Fab Forensics: Increasing Trust in IC Fabrication

Gedare Bloom,* Bhagirath Narahari, and Rahul Simha
George Washington University, 801 22nd St NW, Washington, DC 20052
E-mail: {gedare,narahari,simha}@gwu.edu, Tel: (202) 994–7181, Fax: (202) 994–4875
* Corresponding author

***Abstract*—Fabrication and design are now performed by different companies as semiconductor fabrication facilities (fabs or foundries) seek to reduce costs by serving multiple clients and consolidating resources. However, lack of immediate control and observation reduces the trust which IC designers have in some fabs. To help fabs increase trust in their processes, we propose an approach for logging forensic information of the fab process and printing the information on chips so that examination of the chip reveals provable deviations from the design. Fab owners can benefit by catching rogue employees and by demonstrating high security standards to their customers. Our proposed solution uses a light runtime system that interacts with a trusted platform module (TPM).**

I. Introduction

Modern integrated circuit (IC or chip) design follows a well structured path from specification to fabrication as shown in Fig. 1. In this supply-chain, engineers in a fabless design house produce a *design* – the Intellectual Property (IP) of the company – that is the starting point for fabrication. Often, this chain starts with the design specified in a high-level description language, followed by synthesis into a gate-level netlist, and then given a physically realizable form by place-and-route tools that generate a geometric representation of the 3D structure of the IC. This geometric representation, called the GDS-II file, is essentially a large database of polygons that drives fabrication processes. At any stage of the design path, a design house may incorporate IP blocks sold by IP vendors. Soft IP is distributed either as source code or as pre-synthesized logic that is supported by a particular tool chain. Hard IP is distributed as mask layers that are merged with the designer's GDS-II and are guaranteed for a particular fab's processes.

Chip designers send a finished design to a semiconductor fabrication plant (fab or foundry) for chip manufacturing. The GDS-II file is often optimized by the fab to match the physical constraints ("rules") of the fab's processes. This involves adding, removing, or modifying polygons within the GDS-II: For example, making lines wider because fab machinery cannot support too fine a resolution. After conforming to the fab rules, the final version of the GDS-II drives the creation of lithographic photomask (mask) layers that block light during the layer-by-layer wafer etching process. Note that the masks commit the IP to its physical form, thus we will focus on the security of the IP between design and mask creation.

The current push in the semiconductor industry appears to be toward collaborative efforts between a fab and various IP vendors, for example Taiwan Semiconductor Manufacturing Company's Open Innovation Platform, or the Common Platform and Joint Development partnership involving IBM, Chartered, Samsung, and others. These efforts aim to form an eco-community centered around fab processes, so that IP vendors can provide hard IP that is well-matched to the fabs in the collaboration. Thus, the fab owners become brokers for both IP blocks and for traditional chip manufacturing.

Fab-oriented eco-communities show potential to benefit the electronic design automation (EDA) community. Fabs benefit because designs and IP vendors become more closely tied to their fab processes. IP vendors benefit by having more straightforward verification and control over the use of their IP. EDA tool suppliers benefit by providing highly-targeted synthesis and routing algorithms that have been shown to work for a particular fab's processes. Fabless design houses benefit by having a consolidated source for both IP blocks and fabrication, and have greater assurance that every tapeout implies a correctly fabricated chip. Thus, the fab has become a natural centerpiece for gathering and distributing IP. Unfortunately, the growth of EDA eco-communities focuses primarily on how to maximize fab and IP vendor revenues, and so far there has been little discussion of security and protection for the fabless design house's IP.

Because the GDS-II file in effect contains the entire design IP, design companies must place great trust in fabs. A single rogue employee at a fab can maliciously tamper with the GDS-II file, can copy and counterfeit the entire GDS-II, or can sell the IP to interested third parties. Although the three attacks (tampering, counterfeiting, and IP theft) can occur at various places in the entire IC supply chain, the centralization of IP at the fab makes it a natural location for both perpetrating attacks and for mounting defenses.

Outsourced chip fabrication was identified as a security risk to US government and industry in 2003 by the US Senate [1] and in 2005 by the Defense Science Board [2]. Industry has also acknowledged these threats [3]. DARPA has sponsored research via the Trust in IC (TIC) program, which was initiated in 2006 [4] and continued in 2007 [5]

The current fab processes are aimed at detecting defects in the mask and wafer production. For example, mask inspection is a well established field for detecting mask defects caused by contaminants. Unfortunately, such techniques are not currently applicable to detecting anomalies introduced by the editing workstations, and provide no deterrence to counterfeiting or IP theft.

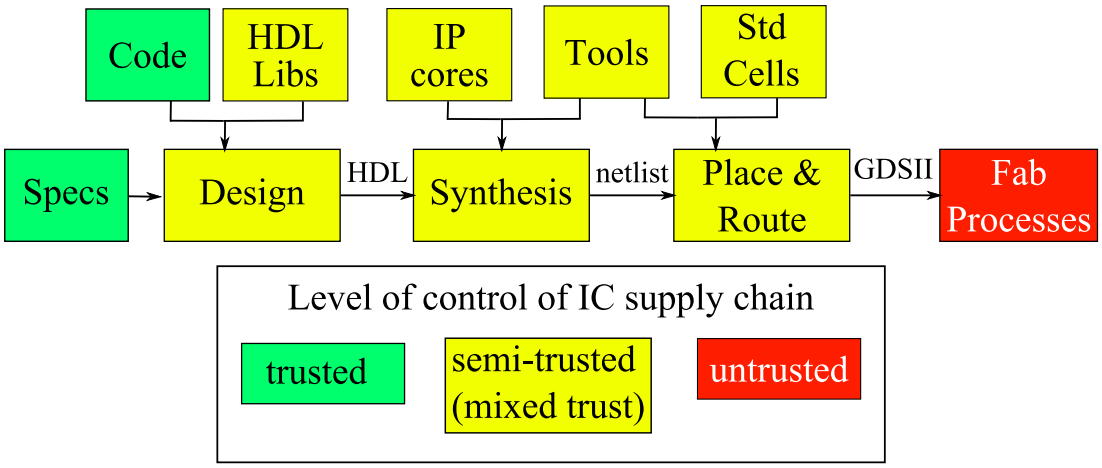

Fig. 1. Modern IC Supply Chain Structure

We consider the question: How can fab owners strengthen the security of their processes? There is a market incentive to consider such an idea because fab owners are just as interested in catching rogue employees as they are in demonstrating security to their clients. In this paper, we describe an idea for improving trust in a fab by instrumenting the supply-chain with added forensic information that is then printed onto masks so that every chip produced by the fab can carry a proof of proper production and so that tampering and counterfeiting are detected.

We propose to retrofit the IC supply chain with a multi-round protocol between the design house and the fab for the purpose of tracking changes made by fab workers. By adding some *trusted hardware* to the fab computers, which edit design files for various proper reasons, we suggest that foundry owners can inject some much-needed trustworthiness to their processes. By including the design house in the fabrication process, alterations can be tracked and later audited.

II. Chip Production

We begin by identifying the flow of information in the most relevant steps toward fabrication, shown in Fig. 1. A design company, as described earlier, encodes its design in the geometric data present in a GDS-II file, which we will call D-GDS (for Design GDS-II). Upon receiving the D-GDS, the fab edits it to conform to fab constraints (rules). After all edits are made, the final GDS is converted to the format used by the mask-printing machines, which then print the masks used in fabricating the layers on each chip. (As mentioned, the mask printer commits the design to a physical form; from this stage onwards additional fabrication processes are used to complete the wafer, which is packaged and returned as a complete chip. For the purposes of forensics, we will not need to consider the steps beyond mask printing.) Note that the GDS files are typically edited using specialized editing software on workstations either co-located with the mask printers or connected by a network. The conversion to mask coordinates (geometric coordinates in the plane) is also performed in software on such workstations.

A. Threats

Some potential attacks include (1) modifying the geometry to insert a malicious (Trojan) circuit to the GDS file, (2) substituting an alternate malicious GDS file for the original, and (3) copying the GDS file for producing counterfeit chips. Additionally, the masks can be stolen even if the other processes are secure. Another attack, which requires collusion among fab workers, would involve producing more chips than are ordered and selling the spares as counterfeits (overbuilding). Numerous other attacks can be imagined and realized in this setting; IP theft is also a possibility, requiring reverse engineering of the GDS.

B. Assumptions

In our approach, the fab owner is trusted, but individual employees are not. Our rationale is that fab owners are legally and financially obligated to provide a trustworthy service, whereas employees might be bought by competitors. Thus, the attacker is a rogue employee capable of misusing fab software, replacing editing applications, or loading a new operating system. However, we assume that physical attacks that involve manipulating computer hardware are not feasible for the attacker. Although loading software on the fab workstations is feasible for a rogue employee, we assume that opening the computer chassis and modifying the hardware is not an option. Additionally, the ability to replace software is mitigated somewhat by the use of the trusted platform module (TPM) [6], which can provide integrity checks on system software. Note that our approach is not perfect – a resourceful employee may be able to circumvent the trusted hardware on fab computers. However, our approach represents a first barrier; the trust placed in fab workstations can be strengthened in other ways (surveillance, for example), a topic we do not consider here.

We also assume that it is sufficient to protect the chip design only between the design house and the mask printing. Although an adversary may be able to steal a set of masks, they are unlikely to be usable with different fab technology. Additionally, we assume that editing the mask to manipulate the original chip design or the final, printed forensic information that we add is also not possible for a rogue employee.

III. Trusted Platform Module

Our solution uses TPMs, which provide generic security related features in a hardware module. Using TPMs inside of fab machinery allows a fab owner to provide secure logging of the machines and to facilitate secure communication with chip designers. Note that TPMs are not designed to be physically secure or tamper-resistant. A more secure, more expensive option is to use a secure coprocessor, e.g. the IBM 4758 [7]. Regardless, the functionality provided by a TPM is appropriate for our solution.

A. TPM Functionality

A TPM provides secure storage and security-related functions. First, the TPM provides internal storage for the secret key E_s of a public key encryption pair, for a tree of derived secret keys and other random values, and for a set of platform configuration registers (PCRs) for storing cryptographic hash values. Second, the TPM can be requested to add data to a PCR, which causes the TPM to concatenate the value in the PCR with the new data and to hash the sequence to generate the new PCR value. Third, the TPM can cryptographically sign data using E_s. Fourth, the TPM can *seal* (via encryption with a secret key) a blob of data dependent on a user-given secret passphrase and the state of the PCRs; *unsealing* a blob returns the original data only if the user provides the correct passphrase and the PCRs are in the same state as when the data was sealed. For our scheme, we make use of the first three properties of the TPM; we do not require encryption for protecting local storage, which is the purpose of sealing data, but we do use encryption to protect communication between the fab and the design house.

The primary applications for TPMs are digital rights management and platform authentication. For example, a TPM can start from a known clean state and process a series of values by the chained hashing. If the series is known to a remote authenticator, the TPM provides *remote attestation* of the values that were processed. Some challenges in TPM management include defining the initial clean state and providing a deterministic series of values for hashing. There are also security issues depending on the assumptions made in deploying the TPM, though some of these are corrected using appropriate design assumptions or tamper-resistant hardware. In our setting, we assume the TPM is not physically available to the attacker.

A newer feature of TPMs is the ability to measure the integrity of a region of memory and then execute code stored in that region. This ability, invoked with the `skinit` instruction in AMD processors [8] and with the GETSEC[SENTER] functionality in Intel processors [9], sets up a secure execution environment by disabling interrupts and DMA, verifying single-core execution mode, hashing the region of memory, storing the hash in a PCR, and executing the code in the region of hashed memory. By placing code to verify other regions of memory, the TPM can bootstrap integrity measurement of even very large applications. Thus, `skinit` enables the TPM to verify and execute specific memory contents, which is useful for providing *late launch* of trusted code in isolation from the rest of the system software.

Our approach to supply chain management is inspired by research using TPMs. Flicker [10] uses the late launch feature of `skinit` to provide a secure execution environment with a very small trusted computing base. We protect the software running on fab machinery with a mechanism similar to Flicker's. Paul and Tanenbaum [11] use TPM functionality to protect the software of an electronic voting machine, which builds on the work of Kauer [12].

B. Remote Attestation of Fab Processes

Fig. 2 shows several key fab processes enhanced with our proposed features for increasing security. Each workstation or device is equipped with an additional hardware-software *secure component* that works in concert with the other secure components to provide security by embedding forensic data into the flow of information. The secure hardware component is a TPM, as described above. Secure software is in the form of a runtime environment, which can be verified using the TPM and `skinit` to measure the integrity of the software, thus preventing a rogue employee from completely replacing the runtime with some other environment. This requires that the hash of the editing application is provided by the fab.

We next describe our solutions in detail, describing the steps at each fab-related stage of the IC supply chain depicted in Fig. 2.

IV. TPM-based Logging

We present a TPM-based scheme using both cryptographic hash functions and cryptographic signatures. If chip designers only want to detect tampering, they can use our solution with a fixed on-chip resource cost but reduced forensic information. We also provide a more complete forensic solution, which comes at a greater cost. A spectrum of solutions exist between the two extremes, and chip designers can, in agreement with the fab, configure an optimal solution.

The first two stages we instrument, edits to conform with fab rules and conversion from GDS to mask overlay, are identical for all of our proposed solutions. In the last stage, when the mask is printed, we introduce both two options for *how* forensic information gets placed on-chip and two options for *what* information gets placed on-chip.

A. GDS-II Editing

Fig. 2 shows the process of editing the D-GDS to conform with the fab rules. As a pre-processing step, the design company uses the public key of the editing workstation to

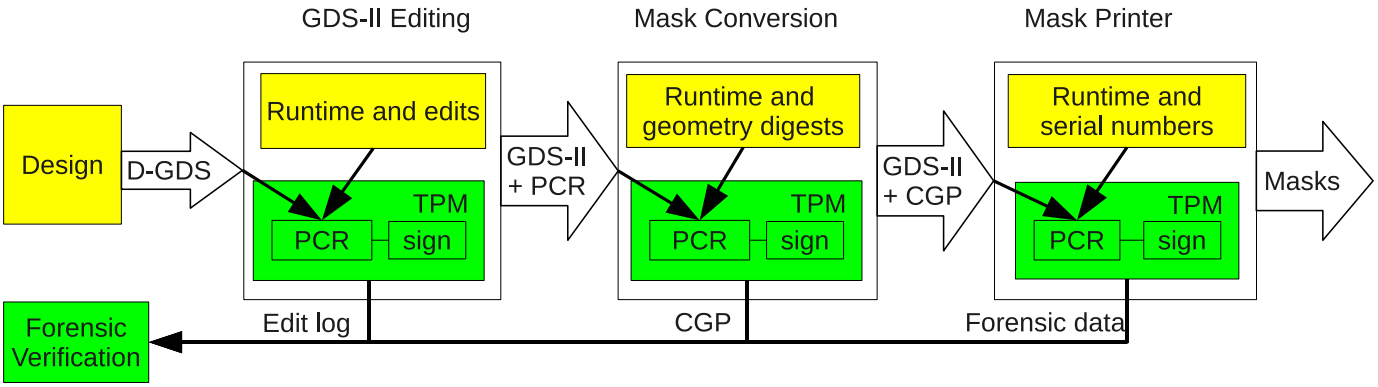

Fig. 2. IC supply chain with security features added to untrusted (uncontrolled) processes. The security features include a trusted platform module (TPM) and runtime software. The platform configuration register (PCR) of the TPM provides a cryptographic hash function used to construct a secure log of the edits made to a design by the fab machinery.

encrypt an asymmetric private key along with the D-GDS in order to establish a private, authenticated channel with the fab. We omit the details of establishing and maintaining the secure channel, and assume that any further communication between the design house and the fab machines is secure.

When the D-GDS file arrives at the fab, the TPM decrypts the file and then measures the integrity of the runtime by using the `skinit` instruction. The computed hash initializes a PCR, and the runtime environment is executed. The runtime instructs the TPM to add the hash of the D-GDS to the hash of the runtime. Other PCRs are set to zero and unused. As the D-GDS is edited, the edits are logged by the runtime on the edit workstation. The runtime also tracks *provenance* information such as user login, location, and timestamps. To prevent unbounded growth of this provenance information, we require that user logins, locations, and timestamps be unambiguously represented as x-, y-, and z-bit numbers respectively.

When a user commits edits made, the runtime requests the TPM to hash the edits and provenance information with the PCR used previously and to sign the result. The edits, provenance information, and signed PCR are encrypted and sent back to the design company, where the data can be re-hashed along with the publicly available hash of the runtime software and the original D-GDS. The computed hash is then compared with the signed hash. The edits made to the GDS-II are thus authenticated. These steps may repeat until the GDS-II description matches the fab rules. By receiving the edits and provenance information, the design company is able to verify the actions made by fab machines.

B. Mask Conversion

The next phase of fabrication is the construction of individual mask geometries from the GDS-II description. As depicted in Fig. 2, augmentations to the mask conversion process are similar to the editing stage.

The mask conversion workstation first receives the GDS-II and PCR value from the editing workstation and uses the signed PCR for authentication and integrity checking. As before, the TPM initializes a PCR to the hash of the runtime, and then hashes in the PCR received from the editing workstation. Sometimes the same machine used for editing is also used for mask conversion, in which case passing and initializing is a no-op. (Conversely, multiple machines might be involved in each of these phases, but we simplify to a single machine per phase.)

During the conversion of the GDS-II to mask overlays, the runtime constructs a digest that summarizes the geometry of the mask overlays. This geometric digest, together with the provenance information associated with digest creation, is joined together by the runtime to form the chip geometry package (CGP). Then the CGP is hashed into the PCR and signed, after which the CGP, signature, and hash are passed to both the mask printer and the design company. The CGP is a compact representation of all the changes made to the D-GDS, together with provenance information.

C. Mask Printing

We provide two parameters that can be configured based on the needs of chip designers. These two parameters are *how* and *what* forensic information is placed on to the chip. We first describe the how. Regardless of the configuration, logging of the mask printing is identical, as shown in Fig. 2.

On receiving the CGP from the mask conversion workstation, the mask printer's TPM initializes a PCR with the hash of the runtime. The runtime then verifies the signature on the CGP and adds the received hash value to the PCR, before requesting the TPM for a unique serial number. The serial number, generated from a pool given to the fab, identifies a single production run at a particular fab. Thus a serial number provides identification of the foundry of origin. Next, the runtime adds the serial number to the CGP and requests the TPM both to hash the serial number and to sign the hash. Finally the CGP, hash, and signature are sent back to the design company. The hash pre-image is now all the forensic information gathered thus far: the provenance data, edits log, conversion digest, printer digest, and serial number.

The next step is to put forensic data on the chip, for which we provide two options.

- Imprint with mask. The masks can be marked and layered such that previously unused areas of the chip will store the forensic information. During photolithography, the information will be transferred from each mask to its respective layer in the chip. Such etching of non-functional information is just like printing a picture (a silhouette) on to the chip. Later, the information can be recovered using non-destructive imaging techniques.
- Register imprint. This option involves more design and takes up significantly more chip area but does not need imaging to recover the forensic data – they can be read from one of the chip's pins. We describe the idea for a single mask; it is easy to replicate the idea for each mask for which a hash is desired. Suppose we wish to store a hash of k bits. The idea is to include a k-bit read-only register in the original design such that every register bit is set to 1. Now, the data lines from the register are routed along the target layer (corresponding to the target mask) with a deliberate gap so that no bit can be read from the register unless the gap is closed. Then, to store a particular bit pattern, the gap is closed for the 1's in the bit pattern by including gap-closing rectangles in the mask. In this way, the register's output can be programmed by the mask. Finally, the register can be set up so that it can be read out JTAG-style from one of the chip's pins.

Both options maintain the economy of scale by requiring just a single set of masks per production run. Our solution does not provide unique per-chip tracking, for which effective solutions exist in research on IC activation, IC authentication, and hardware metering [13]–[19]. Such techniques should be straightforward to incorporate in our proposed scheme, as they tend to focus on design-side solutions.

D. On-chip Costs

One concern for chip designers is the cost of our solution in terms of consumed fabric space. We provide a fixed-price solution that reduces forensics and a solution with full forensic information but a variable cost, for which we provide a measure. Solutions between the two are also possible.

First, just the final hash value can be put on-chip. The advantage of just printing one hash value is a fixed, low fabric space cost while still providing tamper-evident fabrication. The cost is that if a chip is tampered with, there is no forensic information available for tracing the deviation to its source.

Second, the provenance data and all of the hashes, along with the production runs serial number, can be put on-chip, in addition to the final hash value. The cost then becomes a function of the size of the provenance information per edit times the number of edits, which is on the order of $(x + y + z + h) * n)$ transistors, where x, y, z, and h are the number of bits in the login ID, location ID, timestamp, and hash function output respectively, and n is the number of logged edits. Note that x, y, z, and h are fixed cost, and so the space cost will vary with n. n may be anywhere from 1 to arbitrarily large, where a value of 1 indicates a single machine with one user providing editing, conversion, and serial number assignment. In the more likely scenario, multiple fab workstations (and workers) are involved in each phase, so n will vary. Note that we have not included the cost of the access circuitry needed to read the register.

As we mentioned previously, other solutions are possible that will cost between the two we described and will provide differing amounts of forensic information as well. For example, the three phases at the fab can be segmented so that multiple edits are grouped together; perhaps by time or by creating "user groups." Segmenting the phases can provide a fixed cost solution while reducing the ability to precisely identify the source of tampering.

The cost of three phases of edits is straightforward to estimate. First, x would be about 16 bits for a compact, numerical login ID sufficient for more than 64,000 IDs. Next, y would be smaller, depending on the number of workstations, say 8-bits, sufficient for 256 location IDs. The timestamp might be a bit longer, probably 32 bits, enough to differentiate more than 100 years at the granularity of a second. As before, the hash would be around 256 bits. Thus, the fixed cost of $x + y + z + h$ is 312, which would be a transistor count of 936 transistors (per mask) for three logs.

E. Recovering Forensic Data

The unique commitment to hardware can be recovered from every chip, e.g. by non-destructive imaging for the first option or directly reading the imprint register in the second option. Design companies can verify fabricated chips by comparing the recovered data with values stored during the fabrication process. The amount of information printed on chip will determine how well tampering or counterfeiting can be traced to its source. If only the hash was printed, then only tampering will be detected. Correct but duplicated hash values are still useful for determining the source of counterfeit chips. If all of the provenance data is printed, then investigators can find exactly where the data received by the chip designer deviated from the fab process.

The provenance information provides a legal basis for remedial action and, like any strong security measure, may deter attacks in the first place. For counterfeit detection, the values stored in the printing stage can be used, along with the data sent back to the design company, to answer the usual who, when, and where questions. (Of course, we assume the secure components remain secure in all three phases, otherwise the forensic data can be easily spoofed, wiped clean, or forged.)

V. Attacks and Limitations

A naïve Trojan circuit insertion into the GDS-II is prevented by monitoring the edits made at a workstation. Since the editing software cannot be replaced without causing the hash computed during the `skinit` instruction, a rogue employee is unlikely to succeed at implanting a Trojan directly in the GDS-II. Because the secure component records the edits, and the true D-GDS is available to the design house, then, with

the fab owner's assistance, the design house can determine if the edits are malicious. A clever attacker will wait until the final round of edits and modify the GDS-II that is sent to the mask conversion workstation, where the geometric digest will show that the hash of the GDS-II differs from the last round's hash received by the design company. Similarly, if the mask conversion workstation adds the Trojan circuit, then the digest that the mask printer produces and prints to the mask will differ from the digest returned by the mask conversion workstation. An attacker that attempts to simply replace the mask, perhaps by loading a mask produced elsewhere, is frustrated by the digest constructed for printing.

Another attack is to steal masks and produce counterfeit chips at another fab, the so-called counterfeit chip problem (or cloning). However, these chips will contain the mask digest which, if recovered, can become the basis for identifying a rogue employee or for implicating a particular shift at the original fab. A related attack is to simply produce more chips at the fab than were ordered, known as overbuilding. This attack would re-use serial numbers in the mask printer, or produce serial numbers that are beyond an agreed range based on the number of chips ordered. In both cases, the forensic information can serve as the basis for further investigation.

We also acknowledge that our approach has some limitations. First, the approach focuses on detection and does not prevent the insertion of Trojan circuits or the production of counterfeit chips. Second, following suspicion, chip designers need to perform some post-deployment testing and recovery of on-chip forensic information. Third, a resourceful attacker could locate the mask digest and subvert it by superposed etching. Fourth, all of the information used in constructing the provenance information must be saved by the chip designer in order to verify fabricated ICs. Finally, a practical concern is the US export control of cryptographic hardware and software; most foundries exist overseas, so deploying our solution may require some special licensing.

VI. Related Work

The problems of intellectual property (IP) theft, counterfeit chips, and malicious chip alterations (known as Trojan circuits or hardware Trojans) are well-known in the academic realm. Identifying and defending against the threats caused by an insecure IC supply chain is an active research area, with ongoing research investigating the various problems of Trojan circuits, IP theft, and chip counterfeiting.

Trojan circuit detection is relevant to the general concern of securing the IC supply chain. Two primary methods of detection are current research trends in Trojan detection: logic-based testing [20]–[23] and side-channel analysis [24]–[26]. Other literature in the field discusses alternate defense methods [27]–[29] or novel attacks [30].

Although some defensive techniques against IP theft and counterfeiting are instrumented in design stages through obfuscation techniques [19], the majority of existing work focuses on authentication, typically by linking each chip to its source or consumer. Three primary approaches to authentication are heavily studied: IP watermarking [31]–[38]; physical device fingerprinting via manufacturing variability and physically unclonable functions (PUFs) [39]–[44]; and IC activation (or deactivation) [14]–[19]. Watermarking is primarily useful for detecting and tracking IP theft, while device fingerprinting and IC activation can be used more aggressively for prevention, similar to digital rights management (DRM) solutions. For example, many of the device fingerprinting and IC activation techniques involve unique identifiers (keys, tokens, or random values) which can provide hardware metering, the ability for IP producers to track IP usage and charge per consumer [13].

Our solution does not cleanly fit in any of the above areas of authentication, because we are attempting to provide a method for foundry owners to prevent IC supply chain attacks. Past work has focused on the fab as an untrusted, untrustworthy black box; we suggest research should also explore how to make the box more transparent. Some of our proposed methods are straightforward and, we suspect, can be improved using techniques borrowed from the existing IC authentication literature.

VII. Conclusions

We have presented an approach to help a fab assert that it is secure and trustworthy. Our approach relies on the existence of some secure components (TPM hardware and a simple runtime system) that are added to the fab's machines to provide edit logging and tracking. The edits and logs are incorporated with geometric digests and a per-production run serial number, all of which are hashed and printed onto every chip. An advantage of printing edits and logs over just stamping serial numbers on to each set of masks is to provide tracking during an audit so that the perpetrator might be discovered. Not only does our solution authenticate the foundry that produces a chip, but it also encodes data about the employees and workstations that were involved in the production run.

A number of open questions remain, such as what makes an effective geometric digest, what detailed information should be logged, and could other useful information be stored on the chip? There are also some practical issues to resolve, such as how much time and money the proposed solution costs and how to balance that cost with the risk of supply chain problems. These complex issues have important and far-reaching impact on the long-term viability of adding forensics capabilities to fab machinery.

Acknowledgment

This work is supported in part by NSF grant CNS-0934725 and AFOSR grant FA9550-09-1-0194.

References

[1] J. Lieberman, "White paper: National security aspects of the global migration of the U.S. semiconductor industry," http://lieberman.senate.gov/documents/ whitepapers/ semiconductor.pdf, 2003.

[2] D. S. Board, "Task force on high performance microchip supply," http://www.acq.osd.mil/dsb/reports/2005-02-HPMS_Report_Final.pdf, 2005. [Online]. Available: http://www.acq.osd.mil/dsb/reports/2005-02-HPMS_Report_Final.pdf

[3] S. Equipment and M. I. (SEMI), "Innovation at risk: Intellectual property challenges and opportunities," http://www.semi.org/ en/ Issues/ IntellectualProperty/ index.htm, 2008.

[4] DARPA, "BAA 06-40 - solicitations - microsystems technology office," http://www.darpa.mil/mto/solicitations/baa06-40/, 2006. [Online]. Available: http://www.darpa.mil/mto/solicitations/baa06-40/

[5] ——, "BAA 07-24 - solicitations - microsystems technology office," http://www.darpa.mil/MTO/solicitations/baa07-24/index.html, 2007. [Online]. Available: http://www.darpa.mil/MTO/solicitations/baa07-24/index.html

[6] "Trusted computer group: TPM work group," https://www.trustedcomputinggroup.org/groups/tpm/, 2009. [Online]. Available: https://www.trustedcomputinggroup.org/groups/tpm/

[7] "Building a high-performance, programmable secure coprocessor," *Comput. Netw.*, vol. 31, no. 9, pp. 831–860, 1999.

[8] AMD, "AMD64 architecture programmers manual volume 3: General-Purpose and system instructions," Nov. 2009.

[9] Intel, "Intel 64 and IA-32 architectures software developers manual volume 2B: instruction set ref., N-Z." Mar. 2010.

[10] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: an execution infrastructure for tcb minimization," in *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*. Glasgow, Scotland UK: ACM, 2008, pp. 315–328. [Online]. Available: http://portal.acm.org/citation.cfm?id=1352625

[11] N. Paul and A. S. Tanenbaum, "The design of a trustworthy voting system," in *Computer Security Applications Conference, Annual*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 507–517.

[12] B. Kauer, "OSLO: improving the security of trusted computing," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. Boston, MA: USENIX Association, 2007, pp. 1–9. [Online]. Available: http://portal.acm.org/citation.cfm?id=1362919

[13] F. Koushanfar and G. Qu, "Hardware metering," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 490–493.

[14] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, 2004, pp. 176–179.

[15] N. Couture and K. B. Kent, "Periodic licensing of FPGA based intellectual property," in *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, 2006, pp. 357–360.

[16] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, 2007, pp. 674–677.

[17] J. Huang and J. Lach, "IC activation and user authentication for security-sensitive systems," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 76–80.

[18] J. Roy, F. Koushanfar, and I. Markov, "EPIC: ending piracy of integrated circuits," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 1069–1074.

[19] R. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, 2008, pp. 674–677.

[20] S. Smith and J. Di, "Detecting malicious logic through structural checking," in *Region 5 Technical Conference, 2007 IEEE*, 2007, pp. 217–222.

[21] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ICs: problem analysis and detection scheme," in *Proceedings of the conference on Design, automation and test in Europe*. Munich, Germany: ACM, 2008, pp. 1362–1365.

[22] R. Chakraborty, S. Paul, and S. Bhunia, "On-demand transparency for improving hardware trojan detectability," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 48–50.

[23] S. Dutt and L. Li, "Trust-Based design and check of FPGA circuits using Two-Level randomized ECC structures," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–36, 2009.

[24] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, 2007, pp. 296–310.

[25] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware trojans using power supply transient signals," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 3–7.

[26] J. Li and J. Lach, "At-speed delay characterization for IC authentication and trojan horse detection," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 8–14.

[27] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 15–19.

[28] G. Bloom, B. Narahari, R. Simha, and J. Zambreno, "Providing secure execution environments with a last line of defense against trojan circuit attacks," *Computers & Security*, vol. 28, no. 7, pp. 660–669, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2009.03.002

[29] G. Bloom, B. Narahari, and R. Simha, "OS support for detecting trojan circuit attacks," in *Hardware-Oriented Security and Trust, IEEE International Workshop on*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 100–103.

[30] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. San Francisco, California: USENIX Association, 2008, pp. 1–8.

[31] J. Lach, W. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998*, 1998, pp. 299–302.

[32] A. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J. Wong, "Copy detection for intellectual property protection of VLSI designs," in *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, 1999, pp. 600–604.

[33] A. Kahng, J. Lach, W. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 10, pp. 1236–1252, 2001.

[34] E. Charbon and I. Torunoglu, "Watermarking techniques for electronic circuit design," in *Digital Watermarking*, 2003, pp. 347–374.

[35] A. Caldwell, H. Choi, A. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. Wong, "Effective iterative techniques for fingerprinting design IP," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 208–215, 2004.

[36] L. Yuan, P. Pari, and G. Qu, "Soft IP protection: Watermarking HDL codes," in *Information Hiding*, 2005, pp. 224–238.

[37] M. Lin, G. Tsai, C. Wu, and C. Lin, "Watermarking technique for HDL-based IP module protection," in *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on*, vol. 2, 2007, pp. 393–396.

[38] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla, and A. Lloris, "IPP@HDL: efficient intellectual property protection scheme for IP cores," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 5, pp. 578–591, 2007.

[39] K. Lofstrom, W. Daasch, and D. Taylor, "IC identification circuit using device mismatch," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, 2000, pp. 372–373.

[40] S. Maeda, H. Kuriyama, T. Ipposhi, S. Maegawa, and M. Inuishi, "An artificial fingerprint device (AFD) module using poly-Si thin film transistors with logic LSI compatible process for built-in security," in *Electron Devices Meeting, 2001. IEDM Technical Digest. International*, 2001, pp. 34.5.1–34.5.4.

[41] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[42] E. Simpson and P. Schaumont, "Offline Hardware/Software authentication for reconfigurable platforms," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, 2006, pp. 311–323.

[43] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: A nondestructive hidden characteristics extraction approach," in *Information Hiding*, 2008, pp. 102–117.

[44] G. Hammouri, E. ztrk, B. Birand, and B. Sunar, "Unclonable lightweight authentication scheme," in *Information and Communications Security*, 2008, pp. 33–48.