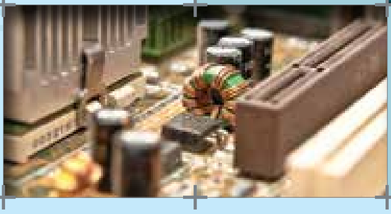
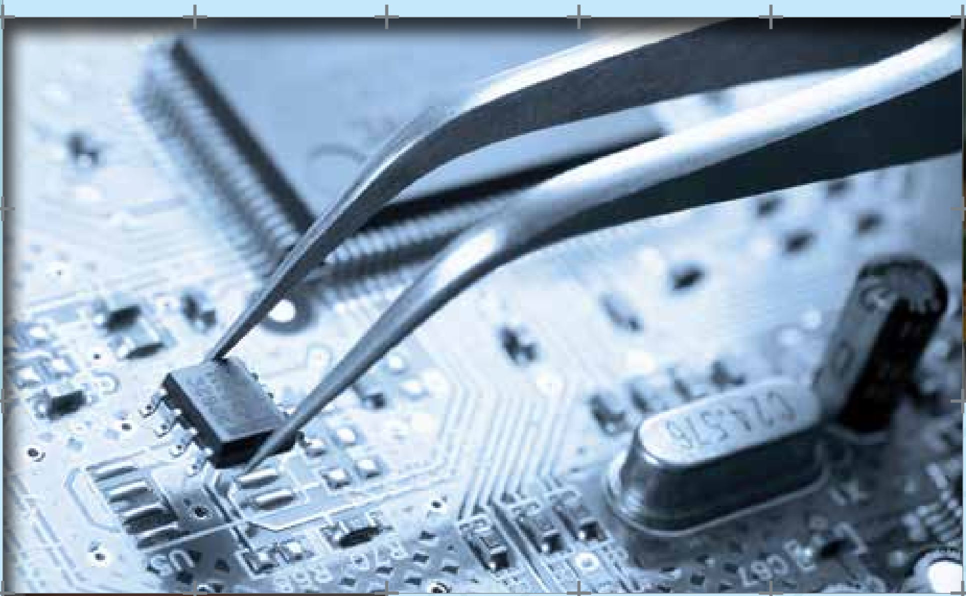
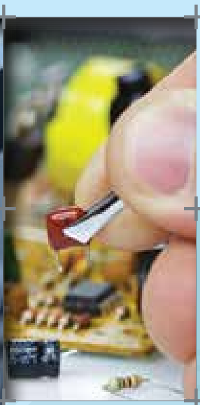
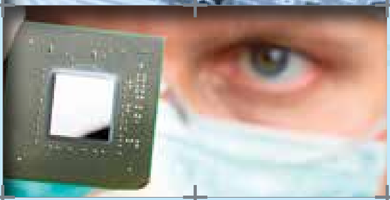
COMPUTING COMMUNITY CONSORTIUM CCC

Semiconductor Research Corporation



Research Needs for SECURE, TRUSTWORTHY, and Reliable SEMICONDUCTORS

www.src.org www.cra.org/ccc/

EXECUTIVE SUMMARY

Virtually all aspects of our lives are touched by semiconductor technology and the integrated circuits that enable our 'smart' and networked world. Yet the same trends that enable more powerful and functional devices and systems also increase certain risks that may compromise their trustworthiness and security. In addition, counterfeit parts and malicious attacks for economic or political gain are a growing threat, especially to government and critical infrastructure systems. The convergence of these trends makes it imperative that government and industry invest in the science and engineering that will strengthen the trustworthiness and security of semiconductors and provide a hardware foundation of trust.

Design and manufacture of today's complex semiconductor circuits and systems requires many steps and hundreds of engineers, typically located at multiple companies worldwide. Detailed specifications are converted into schematic and then physical designs that may include billions of transistors. Considerable resources are invested along the design and manufacture path to verify that the product performs as intended. However, the process is focused primarily on answering the question, "Does the chip do what it was designed to do?" not on the question, "Does it do anything else?"

The software community faces similar challenges. Academic research in the past decade has led to advances in semantics-based program analysis tools and specification languages in addition to verification methods for functional correctness, information flow security, and verifiable protection mechanisms even in the absence of full functional verification. Can some of these methods inspire new methods and tools for semiconductor verification, especially at higher levels of abstraction and closer to the hardware/software interface?

This report is based on a workshop in January 2013 that brought together academic, industry, and government experts from the semiconductor and software communities to discuss approaches, based on experience in their respective fields, for improving trustworthiness of semiconductors. The workshop highlighted gaps in the current semiconductor design and manufacture processes and identified areas where research is needed to provide greater assurance that semiconductors do what they are supposed to do, *and nothing else.*

The overarching need is for research in "Design for Security," with the objective of decreasing the likelihood of errors that cause incorrect behavior, increasing resistance and resilience to tampering, and improving the ability to provide authentication. Design for security requires new strategies for architecture, specification, and verification, especially at the stages of design where formal methods are weak or absent. Design for Security "bakes in" security at the earliest stages rather than attempting to add later or waiting until a problem is identified, which is both riskier and costlier. Design for Security includes techniques that enable low-cost authentication so as to thwart counterfeits.

The following are areas in which research is needed.

1. Functional specification of circuits and systems at the architecture level, including techniques that support formal reasoning and efficient verification.
2. Properties that, if specified and enforced, can provide assurance that a chip is secure when used in a particular application.
3. Strategies and techniques for 'functional + security' verification (i.e. provide a level of confidence that a design does what is desired, and nothing else), for example, including through use of properties identified above, *within levels, in particular at the more abstract levels, prior to the Register Transfer Level (RTL).*
4. Strategies and techniques for functional + security verification, including through use of properties identified above, *at the transition between steps or levels.*
5. Strategies and techniques for functional + security verification, including through use of properties identified above, *between on-chip modules and at the interface with third-party intellectual property (IP).*
6. Metrics/benchmarks for measuring "trustworthiness". Such metrics are critical for assessing various strategies and analyzing cost/benefit.
7. Strategies that allow customers/users to nondestructively authenticate the provenance of a semiconductor at low cost. Ideally, such strategies should be based on features or functions that are "unclonable" and not proprietary or sensitive.

The research envisioned in this report is multidisciplinary, drawing upon expertise in semiconductor design from the architecture to physical level, in semiconductor manufacture, and in software security and assurance. Success will depend on the development of a broad industry and academic community that collaborates and interacts to inform and guide the research direction, to provide researchers with access to real world data and tools, and to provide pathways for results to transition to practice.

The Semiconductor Research Corporation (SRC) offers a successful model for industry-academic collaboration in partnership with government sponsors. In fact, many of the design techniques and tools used by the semiconductor industry today, including for verification, have their origins in university research funded by SRC together with the Federal government. SRC is leading an industry working group on Trustworthy and Secure Semiconductors and Systems (T3S) that provides a framework based on the SRC model for partnering between industry and government to sponsor university research.

Now is the time to launch a collaborative program of research with industry and government support in "Design for Security." Such a program will create an interconnected community of experts that advances the field of hardware security, provide the basic elements for future industry-scale tools, and educate the scientists and engineers who will become the security leaders in academia, industry, and government.

INTRODUCTION

As a society and as individuals, we rely in myriad ways on semiconductors—the silicon-based integrated circuits (ICs) and systems that are fundamental to today's smart, networked, information-driven world. They are not only in our phones, computers, and televisions, but in our cars, home appliances, and medical devices. They are embedded in and control industrial, healthcare, energy, defense, and transportation systems. Our ability to trust and rely on these systems and devices requires that each component is secure, including both the software and the hardware on which it runs.

This report is the result of a workshop held January 15-16, 2013 in Arlington, Virginia at which experts from the semiconductor and software/programming language communities convened to discuss approaches in their respective fields to improving reliability, trustworthiness, and security. The workshop organizing committee, agenda, and attendees are shown in Appendices to this report. The objective was to identify research based on the state of the art in each community that can lead to a "foundation of trust" in our world of intelligent, interconnected systems built on secure, trustworthy, and reliable semiconductors. While the workshop allowed the communities to learn about the approaches and capabilities for ensuring correctness in their respective areas, it also revealed the differences in focus, priorities, and cultures between the semiconductor industry and software academic researchers.

This report highlights the growing challenges and threats to semiconductor security and identifies research that can help strengthen semiconductor-enabled systems in light of approaches developed by the software community.

SEMICONDUCTOR TRENDS POSE CHALLENGES

The use of semiconductors is predicted to continue to grow at an increasing rate. According to one estimate, the number of transistors (the individual "switches" that store 1's and 0's) will increase by a factor of 200 between 2005 and 2015, reaching 1,200 quintillion (1.2×10^{21}) worldwide[1].

The ever growing number of—and number of uses for—semiconductors is made possible in part by Moore's Law, the exponential trend toward smaller-sized features and more functionality over time. Today, features on an integrated circuit are measured in nanometers, and a single microprocessor can have more than a billion transistors. Many applications rely on a "system on a chip" (SoC), which may combine in a single package multiple functions, such as sensors, data storage, and computation, along with video, audio, and other data communication functions.

Not only are semiconductor products complex, the process of designing and manufacturing state of the art semiconductor "chips" is as well, requiring 12–24 months and involving hundreds of engineers. Moreover, the trend is toward less vertical integration, with separate companies performing various tasks and providing software and hardware tools, materials, components, and IP. From abstract design to the packaged final product, the process of making a chip today typically involves companies and individuals located worldwide [Figure 1.].

Trends toward increasing complexity and pervasiveness of semiconductors along with the lengthening and global supply chain and widespread use of design components, or "IP blocks", from third parties pose a number of risks, including for security. The sources of these risks may be grouped into two categories: (1) accidental errors or weaknesses, i.e., "bugs"; and (2) maliciously inserted or modified functionality, or Trojans.

In this report, the term bug refers to an error, flaw, failure or fault in either software or hardware (e.g., circuit design) that causes an incorrect or unintended result or behavior. A bug can result in security vulnerabilities by causing a system to malfunction or by allowing unauthorized access or control of the software or hardware. For the private sector, including individuals and businesses, unauthorized access may lead to theft of financial assets, identity, or intellectual property. For the government, such access may threaten national and homeland security.

Whereas bugs are unintentional, hardware Trojans are intentional and malicious modifications or additions to a circuit. A hardware Trojan can be introduced during the design or manufacture processes[2]. Once triggered, a Trojan may allow unauthorized access or control of the system, or may disable or destroy the system. A detailed and up-to-date taxonomy of hardware Trojans is maintained by the University of Connecticut Center for Hardware Assurance, Security, and Engineering.[3] The opportunity for and therefore the risk of malicious attack via deliberate tampering or insertion of Trojan circuits during design or manufacture is elevated by today's lengthy and global supply chain for semiconductors. Such risks are of particular concern in military, intelligence, and critical infrastructure (e.g. energy, industrial, financial, and transportation) systems.

[1] http://download.intel.com/newsroom/kits/idf/2011_fall/pdfs/2011_IDF_Otellini_Opening_Keynote.pdf [accessed 3/8/2013]

[2] Pham, S., J.L. Dworak, and T. W. Manikas, "An Analysis of Differences between Trojans inserted at RTL and at Manufacturing with Implications for their Detectability," IEEE North Atlantic Test Workshop, May 9-11, 2012 (Woburn, MA, USA).

[3] https://www.trust-hub.org/taxonomyResources/Taxonomy.pdf (accessed on 4/7/13)

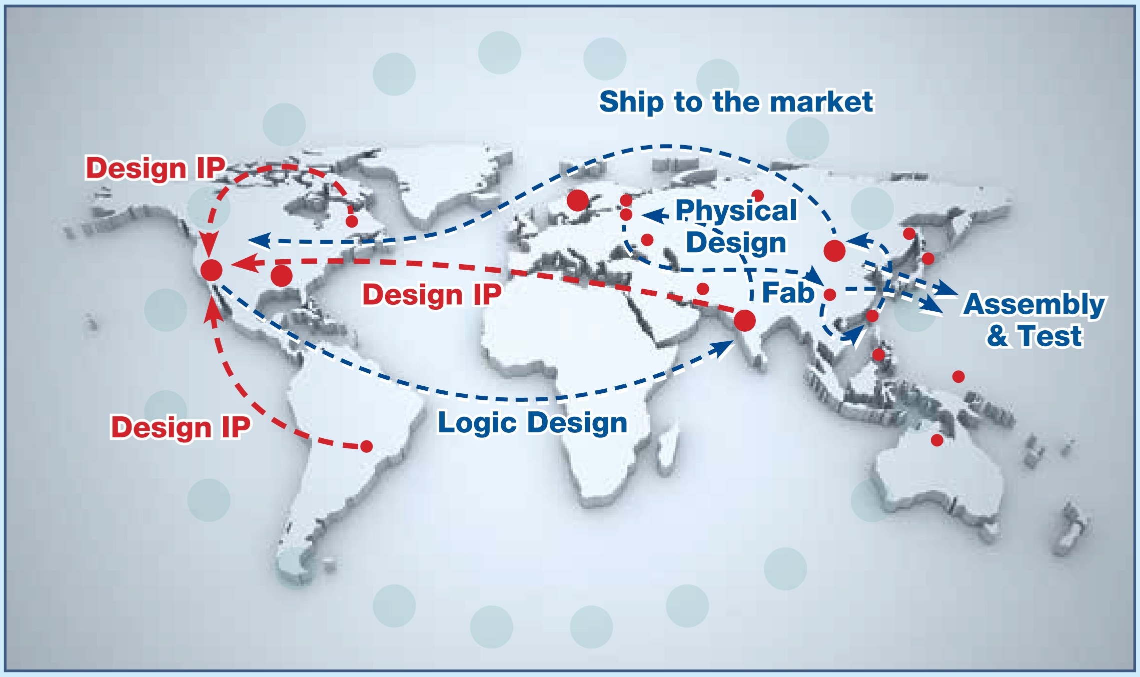


Figure 1. Example of the global path of semiconductor design and manufacture. [Source: M. Tehranipoor, U. Connecticut]

Inserting a Trojan is more difficult in circuits that are more complex and of diminishing feature size, however the likelihood of errors or other bugs goes up with complexity. Moreover, growing complexity along with the increase in pervasiveness of semiconductors makes it more likely that a chip somewhere sometime will fail to perform, with potentially high impact if it is a chip in a critical or life-supporting system. From a security perspective, some bugs may open the door to certain attacks. For example, inputs for which behavior is unspecified and outputs other than those in a specification (such as electrical or thermal responses) may provide an unintended window into chip operation.

Vulnerabilities due to bugs are considered more likely than the malicious insertion of Trojan circuits, in part due to vigorous verification and testing practices already in place. However both are more likely due to the globalization of the supply chain and both might be detected or avoided by integrating security throughout semiconductor design and manufacture.

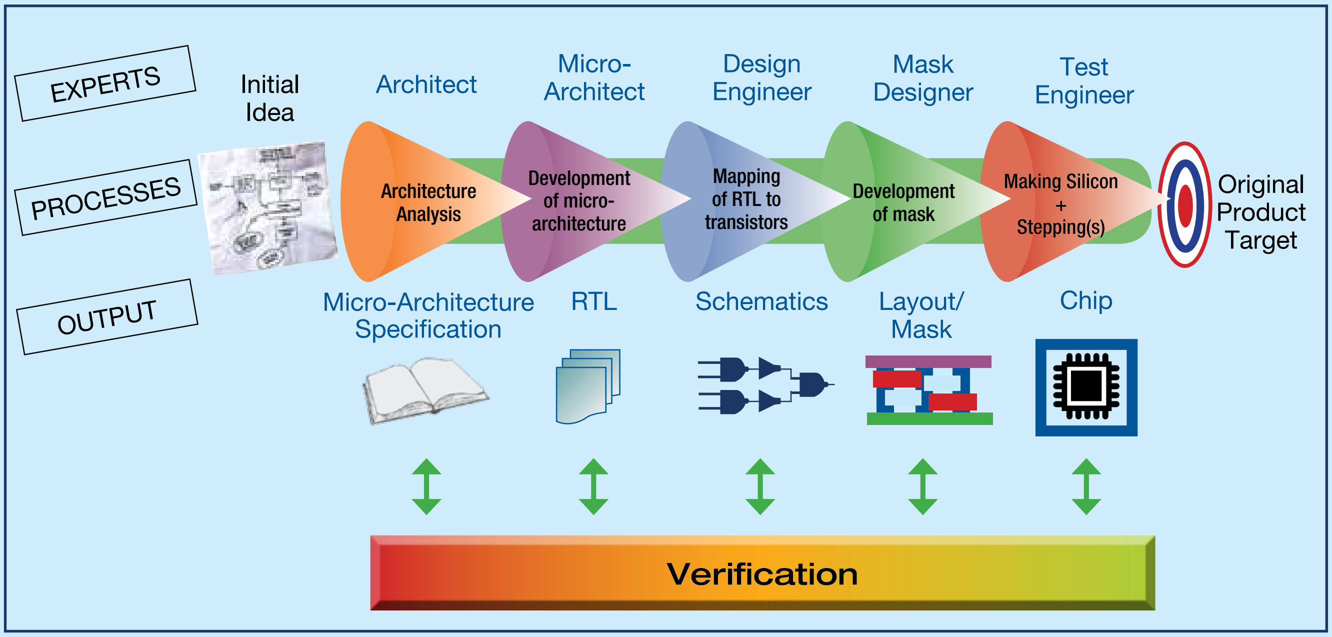


Figure 2. Schematic of the semiconductor design and manufacture processes. [Source: C. Seger, Intel]

SEMICONDUCTOR DESIGN AND MANUFACTURING: CURRENT PRACTICES

The semiconductor design and manufacturing process comprises several stages, shown in a simplified schematic in the figure above (the actual process is less straightforward).

ARCHITECTURE LEVEL

The architectural design process begins with a concept and specification for the functionality, architecture, and key performance attributes of the device to meet the needs of a particular application or market (e.g. computer processor, cellular transceiver, memory chip, microcontroller). If at all complex, the design is broken down into one or more modules, some of which may be imported from a third-party IP provider. Conventionally this specification is in the form of a natural language (text) description, which in successive phases of refinement may run to hundreds or thousands of pages. Large natural-language documents, no matter how well engineered, leave room for inconsistency, incompleteness, or ambiguity that can lead to bugs and vulnerabilities.

Executable specifications are increasingly used – especially for purely digital devices – and are written in special-purpose programming languages that may be input to subsequent design steps in an automated manner. Some examples of these include C/C++ models, SystemC, SystemVerilog Transaction Level Models, Simulink, MATLAB, and architectural description languages (ADL). Such programs can in principle be less ambiguous than natural language. However, an executable specification is essentially a program, and just as programs can have bugs, executable specifications may not match the design intent.

In general this system-level specification describes what the chip is supposed to do; and leaves what it should not do unspecified. In fact, in order to enable as much freedom in implementation and to avoid over-constraining the design early, high-level models are often deliberately vague on many security-related issues.

MICROARCHITECTURE LEVEL

Microarchitecture describes the way an architecture is implemented on a chip, usually in the form of diagrams that depict the interconnections among various elements, from gates and registers to arithmetic logic units. The output of the microarchitecture stage is a RTL description. This step and subsequent design steps are performed with the assistance of various electronic design automation (EDA) tools. The process of confirming that the RTL is correct and that subsequent abstractions are equivalent involves substantial time, effort, and resources in equivalence checking and "functional verification," a field that is described in greater detail later in this report.

LOGIC AND PHYSICAL DESIGN AND LAYOUT

When the RTL is sufficiently debugged, it is translated into lower levels of abstraction, leading to a switch-level design. This translation may be performed automatically by synthesis tools, or by engineers heavily supported by synthesis tools. Automatic formal methods are heavily used during this translation process to ensure that the switch-level design correctly implements the RTL. The switch-level design may be loaded into a programmable-logic device, or it may be translated into a layout of the circuit.

The physical design and layout steps include selecting which gates to use from a library of available logic gates in a particular technology (technology mapping), placing them on the integrated circuit (placement), wiring them together (routing), and converting this design to a set of shapes to be used in chip fabrication (layout) obeying restrictions (design rule checking). These processes are all aided by EDA tools. The output is a machine-readable representation (typically in GDSII format) of the graphical layout for each layer in the manufactured chip. Analog and mixed-signal designs can also be created using a library of components (e.g. amplifiers, converters, input stages, output stages, etc. in a cell library), frequently with manual editing for performance improvements to meet specifications and to ensure matching at the input/output interfaces with other circuits on the chip.

FABRICATION, TEST, AND PACKAGING

Masks are made, typically by a firm that specializes in mask fabrication, for each layer to be deposited and are used to create the integrated circuit on a silicon wafer using a photolithographic process. A single chip may have a dozen or more layers of metallic conductor material separated by layers of dielectric insulator. When the wafers are completed, they are diced into individual chips and assembled in packages that are ready for integration into the end product.

The insertion of Trojans during fabrication, versus during design, is considered more difficult and less likely; manufacturing flaws that can lead to incorrect behavior are a greater possibility. The potential for such flaws increases as feature sizes continue to diminish, requiring manufacturing technologies that are precise on the atomic scale. In fact, a current topic of research is how to design for variability in manufacturing at such small scales. Fabricators may make changes in the as-delivered circuit layout to ensure manufacturability, not intending to alter system behavior. While such revisions generally are made in consultation with the designers, there is the possibility for modifications to be made without the designers' knowledge.

Many tests are performed throughout the fabrication process, typically using automated tools, to verify that the chip behaves according to the specification and that no errors have been introduced during the various steps. The packaged devices are also tested to ensure that they function correctly before being shipped to the customer. It should be noted that digital devices and digital subsystems on a chip are better tested thanks to existing models (fault models, functional models, design-for-test standards and methods), whereas analog and mixed-signal devices frequently must be tested using functional models without accepted test standards and methods. In any case, testing is not designed to detect the presence of Trojan circuits that may have been added at any point along the design or manufacture path.

****

A number of strategies and practices to provide assurance of quality and reliability have been put in place throughout the design and manufacture process. Some are non-technical, e.g. tracking parts, training personnel, etc. Other techniques, such as verification, are highly technical and are the subject of ongoing research and development.

VERIFICATION: THE STATE OF THE ART

Verification is the field focused on confirming that a chip does what the user/designer specified and is performed at points along the design and manufacture path using various static (or formal) and dynamic (or simulation) methods. Formal methods apply mathematical techniques, such as logic, to reason about the possible behaviors of a system whereas simulation is the execution of a system, or of simulated models of a system at some level of abstraction, against real or synthesized inputs. Where practical, formal methods can be much more efficient and reliable than simulation. Dynamic verification techniques include massive simulation of a design, e.g. billions of possible states, and comparison of the results with specifications and formal methods whereby equivalence of designs at different levels and the validity of assertions about behavior can be proved. Formal methods and simulation techniques supported by sophisticated software tools that consume significant resources. Each has advantages and both are applied to varying extents in different parts of the design and manufacture "tool chain".

In the last ten years, formal verification has made significant progress—moving from a subject of academic research to a deployed technology. However, formal methods are used more in some stages of design than others. They are not widely used in the steps leading to the RTL; for example for chip/module architecture specification or for verification that the RTL correctly implements this specification. On the other hand, formal methods are used to verify properties of the RTL, for example that some components cannot dead-lock or that the floating-point addition satisfies the IEEE standard. In the absence of generally usable formal methods prior to the RTL, testing against reference behavior is used to evaluate the correctness of RTL designs. This is adequate to ensure sufficiently correct behavior in practice, but such testing is expensive and generally cannot detect deliberately inserted Trojan functionality that is triggered only under very specific conditions.

The semiconductor industry uses both formal methods and simulation tools for verifying that the switch-level design and physical layout correctly implement the RTL. At the physical design stage, different semiconductor companies use different methods and tools. Some rely on consistency checking and model checking at each level of synthesis, while others use functional equivalence verification (FEV) that crosses levels. The latter provides greater assurance because, in principle, it rules out the possibility of bugs in the translation between levels. However, FEV requires a more sophisticated set-up, because it must reason about the translation of meaning between two “languages” rather than the preservation of meaning in a single language.

Functional verification at various stages of design consumes considerable industry resources and powerful software tools have been developed and continue to evolve and improve. Fundamental research is focused on improved model checking algorithms and satisfiability (SAT) solvers, word-level and higher-level solvers, system-level verification (including microcode and software), and formal methods for analog/mixed-signal verification. However, verification today places little emphasis on providing assurance that additional functionality that could compromise reliability or security is not present.

The general lack of verification beyond that for functionality—i.e. for security— poses a vulnerability that needs research and that may benefit from strategies that have been developed by the software and programming language community.

SOFTWARE ASSURANCE METHODS

Software security vulnerabilities plague interconnected devices and systems, including the Internet of Unix/Windows/Mac computers, the “smart grid”’ of industrial plants and electrical distribution, and the wireless smart-phone network. Most of these vulnerabilities do not come from an “insider threat,” i.e., a malicious programmer who inserts secret trapdoors into the software. Instead, ordinary software bugs are discovered that can be exploited by an attacker who sends carefully designed, unexpected input to trigger the bug in such a way that the attacker can take control of the system, or at least cause it to not function properly (e.g., denial of service).

Researchers in programming languages, software specification and verification, and system specification have in recent years been developing techniques that could be useful for lifting the state of the art in architecture-level specification from “executable specification” to “logical specification.” A logical specification supports formal reasoning, enabling the verification of architectural-level properties, such as cache coherence. It also provides a

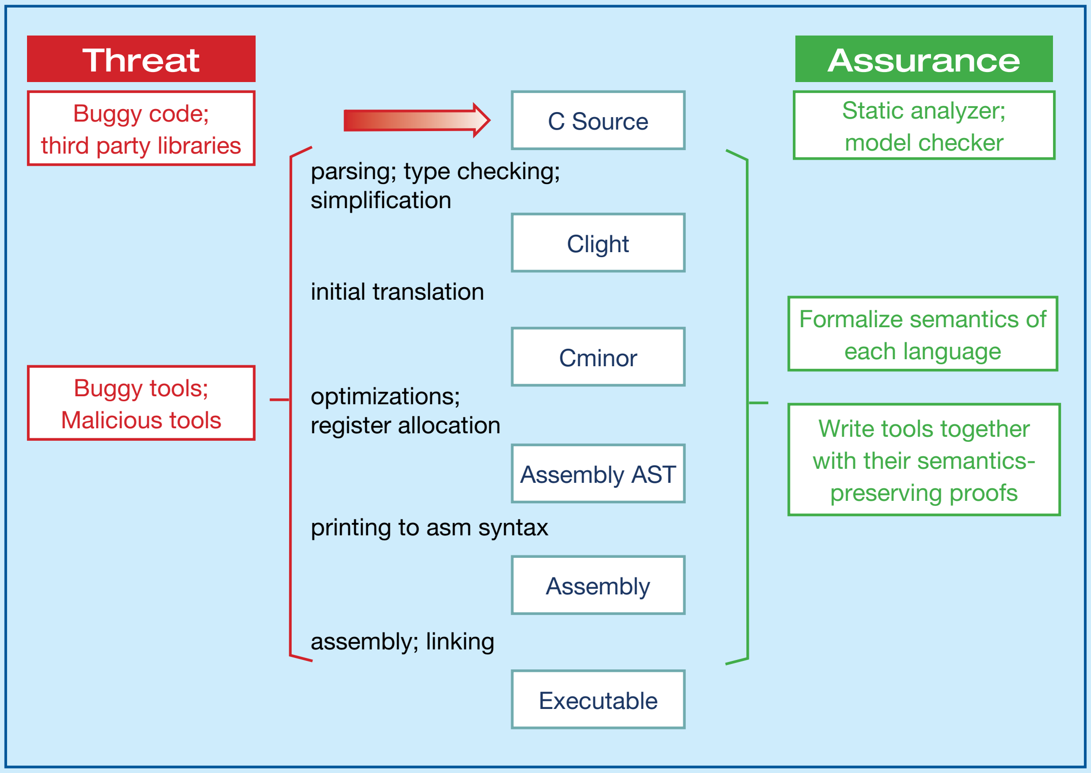

Threat
Buggy code; third party libraries
Buggy tools; Malicious tools
C Source
parsing; type checking; simplification
Clight
initial translation
Cminor
optimizations; register allocation
Assembly AST
printing to asm syntax
Assembly
assembly; linking
Executable
Assurance
Static analyzer; model checker
Formalize semantics of each language
Write tools together with their semantics-preserving proofs

Figure 3. Flow process of a verified C compiler (CompCert)

firm foundation to build software protection mechanisms, from certified compilers to verified hypervisors to malware analysis. Figure 3 illustrates the security threats and assurance strategies in the case of a verified compiler.

Like hardware designs, software is built with a multilayered tool chain, composed of many components with complex interfaces, and subject to bugs and security vulnerabilities. Much research has been devoted to software assurance, with significant results in recent years. Some of these assurance-related techniques, described below, have potential for application to the semiconductor design process.

- **Semantic specification:** One cannot prove anything about the behavior of a program (written in some programming language, or machine language) without a specification of what that programming language (or machine language) means. The past two decades have seen significant progress in the science of operational semantics[4], a method for specifying what programming languages mean.

- **Functional specification:** Proving that a program is correct requires having a *formal specification* of what it is supposed to do. There has been notable progress in logics and methods for formally specifying the behavior of software[5].

[4] Gordon D. Plotkin. A Structural Approach to Operational Semantics. (1981) Tech. Rep. DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark.

[5] Thierry Coquand and Gerard Huet. 1988. The calculus of constructions. Information and Computation. 76, 2-3 (February 1988), 95-120.

- **Generic safety/security:** Without a formal specification of what a program is supposed to do, can we prove anything useful about it? Surprisingly, the answer is yes. We can prove safety properties, such as: this program never crashes; this program never accesses memory outside a specified region. We can prove security properties, such as: the secret data provided at input X never flows to low-security output Y. All such proofs require a formal specification of the semantics of the programming language, but they do not require a functional specification of the particular program.

- **Automatic tools:** It is problematic to do proofs about software manually for several reasons: software is large, its specifications are large, therefore the proofs are large and it's difficult to check that every single step in the proof is correct; software is a moving target, it is always being maintained and modified from one release to the next; and it is difficult to keep handwritten proofs in sync with the software. Therefore the software assurance research (and development) community has built automatic *static analysis* tools that check safety and security properties of software; when the software is modified, one simply runs the tools on the new versions.

- **Functional correctness:** It is more difficult to prove functional correctness of a system (i.e., that the system behaves according to a certain specification of what it is supposed to do) than to prove generic safety/security properties. Nevertheless, considerable progress has been made in this area. There are automatic tools such as software model checkers[6], software verification systems[7], and interactive proof assistants to allow machine-verified proofs of functional correctness[8]. Compared with generic safety/security verification, functional-correctness proofs usually require more (human) developer interaction that is specific to the program being verified.

- **Translation validation:** Programmers find it much easier to reason about source-language programs than about the machine-language translations that come out of the compiler. To permit such reasoning in a software-assurance context, one must prove the correctness of the compiler. Research results in the past decade show that it is now feasible to verify the correctness of high-quality optimizing C compilers, or Java compilers. This allows verification tools to focus on the source language, relying on the correctness proof of the compiler itself for assurance of the compiled program. These research results in compiler validation *could not have been obtained* without the prior research in how to specify operational semantics of programming languages—the specification of what it means for a compiler to be correct is the operational semantics of the language being compiled.

Some of the most significant recent progress in software verification has been in the specification of complex source languages and protocols, and the ability to do functional equivalence verification between source languages and machine language—that is, the formal correctness proof of an optimizing compiler. These methods and formalisms from software verification could offer strategies to address two areas of weakness in the hardware verification tool chain: (1) the weakness between the Architecture level and the Microarchitecture/RTL level, where the architectures are (often) insufficiently formalized and there is no formal verification of the correspondence between architecture and RTL and (2) the weakness in the specification and verification of SoC components at the architecture level.

In software, a large class of security vulnerabilities comes from bugs. Verification has been relatively successful compared to traditional testing in reducing these vulnerabilities. In many cases "lightweight" software verification,

[6] Thomas Ball and Sriram K. Rajamani S. 2002. The SLAM project: debugging system software via static analysis. In POPL 02: Proceedings of the Symposium on Principles of Programming Languages. ACM, New York, 1–3.

[7] Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, & Xavier Rival, A Static Analyzer for Large Safety-Critical Software., In PLDI 2003 — ACM SIGPLAN SIGSOFT Conference on Programming Language Design and Implementation.

[8] For example Coq (http://coq.inria.fr/) and Isabelle (http://www.cl.cam.ac.uk/research/hvg/Isabelle/)

which does not aim to prove full functional correctness, but just assures simple safety properties such as type-checking and array-bounds checking, has been effective in catching common vulnerabilities such as buffer overruns. Even software testing can be driven by formal-methods techniques from verification, such as ensuring "coverage" not just in which lines of source code get executed but in the semantic space.

In addition to adapting strategies from software verification, hardware design and verification may be strengthened with respect to security by using approaches from the information security realm, where experts are trained to "think like the attacker." Such an approach might mean treating other on-chip components as if they were deliberately malicious. Thinking like an attacker may find different types of vulnerabilities compared to other methods that model or test faults and failures.

VULNERABILITIES IN SEMICONDUCTOR DESIGN AND MANUFACTURE

The potential for errors, bugs, or Trojans is present throughout the semiconductor design and manufacturing process (Figure 3). Within each step, there are varying degrees of specification and verification. In general, these processes (1) are relatively weak at the architecture stage and (2) are focused on what the chip does, not what it does not do. Security features are too often added as an afterthought, rather than being incorporated from end to end in the designs and processes. Security-related properties are not generally specified and therefore cannot be checked and verified at later stages.

A number of vulnerabilities are associated with interfaces of various types. First, there are interfaces between stages along the path of design and manufacture. Between some stages there is considerable iteration, whereas between others, such as between physical layout and mask fabrication, there is a formal sign off that occurs and much less modification once completed. As mentioned earlier, the semiconductor design and manufacturing process is global, typically involving many firms worldwide and many more individuals. In the course of shipping designs and even physical artifacts around the globe, there are opportunities for bugs or tampering. Once inserted in a design or as part of a third-party IP block, a bug or Trojan may go undetected throughout the rest of the process due to gaps in verification.

Another interface that can be a source of vulnerability is between modules or components on a chip. Misbehavior can result from insufficient specification/verification between modules. In particular, IP blocks from third parties may include bugs or added functionality, whether malicious or unintentional. Like the chip designers, reputable suppliers of IP use high-assurance RTL-to-circuit-layout synthesis and verification tools, and are willing to provide customers with documentation. However, there is a diverse marketplace for design modules and when using IP from less documented sources, the extent of verification is generally opaque to the integrator.

Another example of interfaces is between hardware and software. Security cannot address hardware or software alone; there are many instances where the two are intimately linked, such as the bit-patterns used to program an FPGA (and the tool chain to create, distribute, and validate this bit stream), the microcode used to implement increasingly sophisticated algorithms in the hardware, device drivers, etc. This firmware, which is often overlooked when addressing software or hardware security alone, is relatively obscure (so source inspections are unlikely to be useful) and has a lot of (unchecked) control of the hardware. It is therefore a particularly unprotected pathway for attackers.

Although system attacks via software vulnerabilities have been the easier route for unauthorized access, low-level on-chip software can have errors that are difficult to detect and if in widely deployed devices can be enormously damaging and expensive to replace/repair. Examples of potential vulnerabilities at the hardware-software interface include the following.

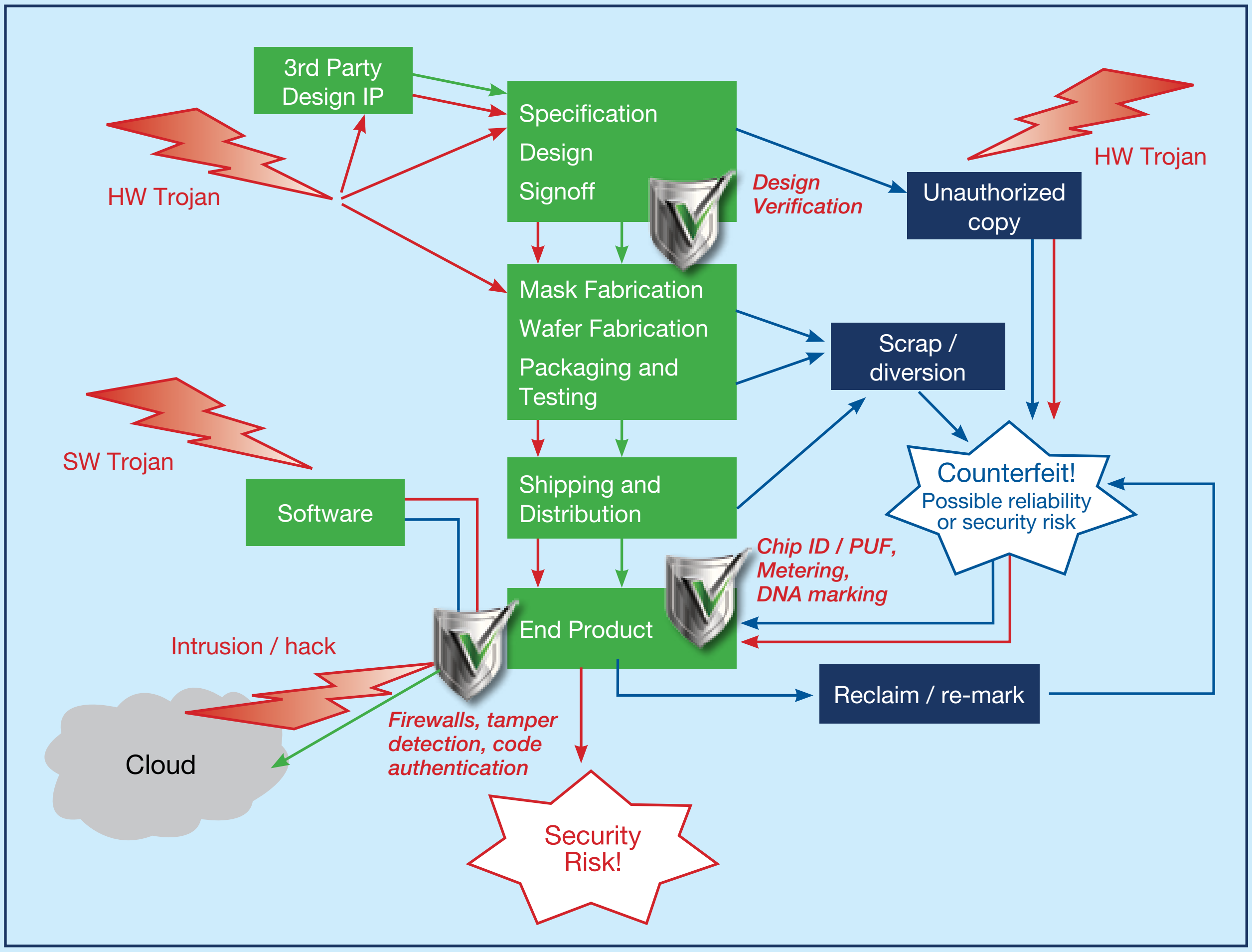


Figure 4. Vulnerabilities in the semiconductor design and manufacture flow to tampering or Trojans (shown in red) and to counterfeiting (shown in blue). [Source: K. Kemp, Freescale]

- Microprocessors (CPU, GPU, or special-purpose processor) provide an instruction-set architecture (ISA) on which the software runs. If this ISA is ill-specified, i.e., contains behaviors that vary from one implementation to another, then software-level vulnerabilities can result.

- Third-party IP modules contain not only transistor circuits ("hardware") but also program software ("firmware"). Like any software, this firmware may contain bugs that can lead to vulnerabilities.

The likelihood of an error or other bug occurring and not being detected along the design and manufacture path is a function of the complexity of the chip, the number of sources of IP blocks, and the coverage and robustness of the specification and verification processes.

The risk of malicious tampering has a different profile. In general, unauthorized electronic access to design files is easier, and therefore more likely, than physical access during manufacture. Although an insider attack can occur at any point in the design/manufacture path, attacks at the design stage (whether by an insider or outsider) can be complex and may be difficult to detect. Attacks at the manufacturing stage are simpler and more likely to be detected during testing.

COUNTERFEITS: ANOTHER TYPE OF THREAT

In addition to bugs and Trojans, a third category of threats that is primarily economically driven, yet impacts reliability and trustworthiness, is counterfeits. The large and lucrative market for semiconductor-based parts, especially those that tend to be in short supply such as obsolete parts that no longer are produced by the original manufacturer, has led to a growing counterfeit problem. The problem is especially significant in the defense and aerospace industries not only because of the many legacy systems, but also because parts that are rated for military or space use command a premium price.

The extent of counterfeit products is not well known, but in 2012 a Senate Armed Services Committee report[9] on counterfeit electronics parts in the Department of Defense supply chain identified roughly 1,800 cases of suspected counterfeit parts in 2009 and 2010 involving more than one million individual parts. The report included specific examples of parts found in a number of Navy and Air Force aircraft. Such counterfeits pose an economic threat to suppliers of legitimate products and can result in a host of problems for customers if the counterfeit does not meet the specifications of the real part, including failure to perform properly and shortened lifespan.

There are several types of counterfeits with different characteristics and risks (shown in blue in Figure 4). The most common pathway is through recycling of used parts that are sold as new or unused, often with fake date codes and other marks. Replacement parts enter the supply chain through many channels and even authorized distributors can end up with reclaimed counterfeits in their inventory.

[9] "Inquiry into Counterfeit Electronic Parts in the Department of Defense Supply Chain," Report of the Committee on Armed Services, United States Senate, May 21, 2012.

Another potential source of counterfeits is from product that fails testing during the manufacturing process. Although foundries work hard to keep yields as high as possible, some wafers and chips do not meet specifications and are scrapped. While processes are in place to account for material that does not pass inspection, an unscrupulous entity could divert, package, and sell such chips as authentic.

The above categories of counterfeits are likely to fail to perform properly because they are already used or flawed in the first place, however other counterfeits may be essentially identical to the original and much harder to detect. A foundry that manufactures chips for others could produce more than were ordered, using the exact same design and materials, and then sell the unauthorized copies themselves. Finally, a skilled person could reverse engineer a legitimate chip and then manufacture "clones" (not shown in Figure 4).

The threat of counterfeits puts greater pressure on companies that design, manufacture, and sell chips to provide means by which customers can verify authenticity. Basic documentation showing chain of custody is used, but also can be forged. Various strategies have been suggested for combatting this problem, including embedding IP protection in the design process, techniques that enable rapid authentication by users, and various characterization methods for detecting counterfeits.[10] On-chip metering techniques to measure use have been proposed to detect used/recycled chips being sold as new. Counterfeits are considered in this report from the perspective of novel strategies for authentication and for making semiconductors more resistant to being counterfeited by novel approaches to design and manufacture.

RESEARCH NEEDED TO ENSURE TRUSTWORTHY AND RELIABLE SEMICONDUCTORS

The overarching challenge is to make semiconductors that are reliable, trustworthy, and secure—even as complexity, the globalization and fragmentation of the design and manufacture process, and the number and capability of attackers all continue to rise. The overarching need is for research in "Design for Security," with the objective of decreasing the likelihood of errors, increasing resistance and resilience to tampering, and improving the ability to provide authentication. Design for Security requires new strategies for architecture, specification, and verification. It "bakes in" security at the earliest stages rather than waiting until a problem is identified later on, which is both riskier and costlier. Specific challenges in need of research include the following.

1. Functional specification of circuits and systems at the architecture level.
   The lack of functional specification at the architecture level, typically at the software-hardware interface, impedes the formal assurance of hardware devices. A formal architecture specification enables verification of correspondence between the architecture and the microarchitecture (RTL) of a device. In software development, functional architecture specification is a necessary component for building trustworthy

[10] Koushanfar, F., S. Fazzari, C. McCants, W. Bryson, M. Sale, P. Song, and M. Potkonjak, "Can EDA Combat the Rise of Electronic Counterfeiting?," DAC 2012, June 3-7, 2012 (San Francisco, CA, USA)

software, such as verified hypervisors and certified compilers. This research includes the investigation of architecture-specification techniques that are broadly applicable and that support formal reasoning and efficient verification.

2. Properties that, if specified and enforced, can provide assurance that a chip is secure when used in a particular application.
   This area includes specification and classification of hardware security properties that, if enforced, can rule out specific classes of hardware attacks. One such example is the information-flow security policy. Research is also needed on the specification of hardware components that are designed for enforcing security (e.g., Trusted Platform Module) and their interaction with the rest of the system.

3. Strategies and techniques for 'functional + security' verification (i.e. provide assurance that design does what is desired, and nothing else), including through use of properties identified above, *within levels, in particular at the more abstract levels* (prior to RTL).
   Functional verification, modeling, and testing at the RTL and below is well studied and widely practiced. Above the RTL level there are new research opportunities, and at all levels there are challenging and high impact research problems in security verification.

4. Strategies and techniques for functional + security verification, including through use of properties identified above, *at the transition between steps or levels.*
   The scope of equivalence verification at transitions from one level to the next needs to be broadened. Verification research is especially needed between the architecture and microarchitecture levels. While researchers in academia and industry have been relatively successful at functional verification between the microarchitecture/RTL level and the gate/switch level, even at these lower levels of abstraction growing security concerns require new research.

5. Strategies and techniques for functional + security verification, including through use of properties identified above, *between on-chip modules and at the interface with third-party IP.*
   Should such verifications be done with respect to architectures of these components, or microarchitectures? What kinds of inter-module security verification can be done without full functional specifications?

6. Metrics for measuring "trustworthiness".
   What are indicators of resistance to attack? Metrics are critical for assessing various strategies and analyzing trade-offs. The security landscape of threats and countermeasures is dynamic. Tools are needed that allow for suites of metrics to be assessed and benchmarked on a periodic basis. This topic cross-cuts all of the research topics.

7. Strategies and techniques that allow customers/users to nondestructively authenticate the provenance of a semiconductor.
   Despite research and progress in "design for security," in cases of critical applications, customers want to be able to confirm the authenticity and status (e.g. new vs. used) of a particular chip. Strategies that can be applied to all semiconductors—i.e. are easy and inexpensive—are preferable. Ideally, such authentication strategies should be based on features or functions that are "unclonable" and not proprietary or sensitive.

The order of the questions above does not imply priority. However, there are interdependencies; for example, results from research on the first two topics about specification and security properties will be useful to research topics three through seven. Progress on all of these priority areas will be needed in order to build fundamentally secure chips and hardware.

ADDRESSING RESEARCH THROUGH COLLABORATION

Many of the design techniques and tools used by the semiconductor industry today, including for verification, have their origins in university research, often funded by both the SRC[11] and the Federal government. Driven by industry, SRC was also an early funder of "design for test," "design for manufacture," and other design automation research. Furthermore, much of the basic research on hardware verification, from early foundations to today, has been funded by SRC. Certain aspects of the "design for security" research outlined in this report are a natural extension of the ongoing industry efforts—both internal and through SRC—to improve the tools and processes used to produce correct (bug-free) semiconductors.

The research outlined in this report, however, goes beyond the areas that are the domain of industry research and development (R&D), into areas that heretofore have been largely unexplored, but that no longer can be ignored. Research in these areas will improve not only trustworthiness and reliability but also security, and therefore is relevant to both government and industry and can be accelerated through close public-private collaboration.

In every area in which SRC has invested, collaboration among industry and academic partners has been critical to success; the same is expected to be true in the areas highlighted in this report. In general, interaction among industry experts and academic researchers (faculty and students) is vital to ensuring that the research is continuously informed by current drivers and inputs from those who are on the front lines of semiconductor technology R&D and the dynamic challenges to trust and security. The SRC Industry Liaison Program engages SRC-funded researchers at the project level in order to provide input and feedback in real time, mentor students as they progress toward graduation and ensure efficient technology transfer. Interaction between students and industry engineers and scientists greatly enriches the educational experience, often leading to opportunities for internships and eventually employment. Industry-academic interaction also can facilitate progress by making industry expertise, data, and tools available to academic researchers.

The research envisioned in this report is multidisciplinary, drawing upon expertise in semiconductor design from the architecture to physical level, semiconductor manufacture, and software security and assurance. Success will depend on the development of a broad industry and academic community that interacts, for example, through periodic in-person reviews and on-line seminars, workshops, and other resources. SRC has infrastructure and experience in managing such interactions and already has established an industry group on Trustworthy and Secure Semiconductors and Systems (T3S) to identify long-term research needs and to create partnerships with government to address those needs. The Computing Community Consortium[12] also provides

[11] SRC (www.src.org) is a not for profit industry consortium that invests in university research on behalf of its members, often in partnership with federal agencies.

[12] http://www.cra.org/ccc/index.php

a forum for exchanging information. Together, SRC T3S and CCC are able to engage, convene, and connect the diverse relevant communities.

In addition to posing broad research questions, several of the research areas could benefit from a set of challenge problems that are developed with input and collaboration from industry. For example, research on how to specify SoC components or on how to protect unspecified SoC components from each other could benefit from a set of sample SoC components, such as USB controllers, 802.11 Wi-Fi, or GPS systems. In addition to individual components, examples of whole-system integration such as a smart phone or voting machine would be useful to research teams. In fact, developing open-source benchmarks such as these is itself a valuable form of research.

CONCLUSION

Trends in semiconductor technologies and their use, as well as in how they are designed and made all point toward a need for research on how to make these devices even more reliable and trustworthy. At the same time, the threat of malicious attack is also growing. This report identifies several areas of research that can increase the security, trustworthiness, and reliability of semiconductors. In many cases such research has potential to improve the semiconductor design/synthesis process so as to save time and effort in addition to offering security improvements.

Now is the time to launch a collaborative program of research with industry and government support in "design for security"—for the benefit of those who design and manufacture semiconductors and for the benefit of those who use them. Such a program will create an interconnected community of experts that advances the field, provides the basic elements for future industry-scale tools, and educates the scientists and engineers who will go on to positions in industry, academia, and government, where the need is large and growing for experts in the design and manufacture of trustworthy, reliable, and secure semiconductors.

APPENDIX A: CONTRIBUTORS & WORKSHOP ORGANIZERS

Andrew Appel, Princeton University
Chris Daverse, Semiconductor Research Corporation
Kenneth Hines, Computing Community Consortium
Rafic Makki, GLOBALFOUNDRIES
Keith Marzullo, National Science Foundation
Celia Merzbacher, Semiconductor Research Corporation
Ron Perez, Advanced Micro Devices
Fred Schneider, Cornell University
Mani Soma, University of Washington
Yervant Zorian, Synopsys

APPENDIX B – WORKSHOP ATTENDEES

Rob Aitken, ARM
Nina Amla, NSF
Andrew Appel, Princeton
Kris Ardis, Maxim Integrated
Clark Barrett, NYU
Jay Bhadra, Freescale
Arnett Brown, Booz Allen Hamilton
Furkan Cayci, U. Delaware
Chris Collins, Texas Instruments
Chris Daverse, SRC
Don Davidson, DoD/OSD
Ben Epstein, DARPA
Jeremy Epstein, NSF
Saverio Fazzari, Booz Allen Hamilton
Kathleen Fisher, DARPA
Alex Halderman, U. Michigan
Suzanne Iacono, NSF
Anupam Joshi, U. Maryland / BC
William Joyner, SRC
Ramesh Karri, NYU
Kevin Kemp, Freescale
Fouad Kiamlev, U. Delaware
Farinaz Koushanfar, Rice
Andreas Kuehlmann, Coverity
Richard Kuhn, NIST
Ruby Lee, Princeton
Jeremy Levitt, Mentor Graphics
Patrick Lincoln, SRI
Igor Linkov, US Army Corps of Engineers
Rafic Makki, GLOBALFOUNDRIES
Brad Martin, ODNI
Keith Marzullo, NSF
Carl McCants, IARPA
Celia Merzbacher, SRC
Ron Perez, AMD
David Pichardie, INRIA/Harvard
James Plusquellic, U. New Mexico
Tim Polk, OSTP
Dan Radack, IDA
Farhan Rahman, AMD
Fred Schneider, Cornell
Carl Seger, Intel
Peter Sewell, Cambridge
Mani Soma, U. Washington
Peilin Song, IBM
Gang Tan, Lehigh
Mohammed Tehranipoor, U. Connecticut
Steve Trimberger, Xilinx
Vivek Vedula, Freescale
Ingrid Verbauwhede, UCLA / KU Leuven
Steve Zdancewic, U. Pennsylvania
Fen Zhao, OSTP
Yervant Zorian, Synopsys

APPENDIX C – WORKSHOP AGENDA

January 15	
	Continental breakfast
8:30 AM	**Welcome and Overview of the Workshop** • Keith Marzullo, NSF, Director of Computer and Network Systems Division • Sponsors (Jeremy Epstein, NSF & Celia Merzbacher, SRC)
9:00 AM	**Plenary Session 1:** Overview of Semiconductor Design and Manufacture (with an eye toward the future) Moderator—Ron Perez, AMD
	Semiconductor Manufacture Tools & Processes and Potential Vulnerabilities (15 mins)—K. Kemp, Freescale
	Semiconductor Design Tools & Processes and Potential Vulnerabilities (15 mins)—S. Trimberger, Xilinx
	Top-to-bottom integrative design & verification (15 mins)—C. Seger, Intel
	Incorporation of designs from multiple sources (15 mins)—R. Aitken, ARM
	Panel Discussion (30 mins)
10:30	**BREAK**
11:00	Plenary Session 2: Overview of Software Assurance Methodologies Moderator—Fred Schneider, Cornell
	Thinking about attacks / minimizing trusted base (20 mins)—A. Appel, Princeton
	CompCert as a software tool chain (20 minutes)—D. Pichardie, INRIA/Harvard
	Specifying the HW/SW interface (20 minutes)—P. Sewell, Cambridge
	Panel Discussion (30 mins)
12:30	**LUNCH** *Reality of hardware vulnerability—F. Kiamlev and F. Cayci, U. Delaware*
1:30	**Breakout sessions** (facilitated discussion on specified questions)
	Facilitators: Jeremy Epstein (NSF), Fred Schneider (Cornell) and Yervant Zorian (Synopsys)
5:00	**Preliminary Breakout reports** (5 min each; bring forward one or two issues/ideas for consideration by all)
6:00 – 8:00	**Reception**

January 16	
	Continental breakfast
8:30AM	**Opening remarks:** Overview of Day 2 (Merzbacher/Epstein)
	Plenary session 3: Further food for thought Moderator—Ruby Lee, Princeton
8:35AM	*Counterfeit chips: What are the threats? (20 mins + 5 mins Q&A)—M. Tehranipoor, U. Conn.*
9:00AM	*Why information flow is different from—and harder than—verifying other kinds of properties (20 mins + 5 mins Q&A)—S. Zdancewic, U. Penn.*
9:30AM	Continue Breakout Groups to finalize output (with 15 min break)
12:00PM	Breakout Group reports & discussion—Working lunch
1:30PM	Wrap-up
2:00PM	*Adjourn*

Established in 2006 through a Cooperative Agreement between the National Science Foundation (NSF) and the Computing Research Association (CRA), the CCC serves as a catalyst and enabler for the computing research community. Its goals are to unite the community to contribute to shaping the future of the field; provide leadership for the community, encouraging revolutionary, high-impact research; encourage the alignment of computing research with pressing national priorities and national challenges (many of which cross disciplines); give voice to the community, communicating to a broad audience the many ways in which advances in computing will create a brighter future; and grow new leaders for the computing research community.

Semiconductor Research Corporation

SRC is a recognized leader in managing collaborative research and has developed efficient, effective and proven mechanisms and processes for creating and managing industry consortia, setting direction, managing and coordinating the research, and disseminating the results. SRC's primary objectives are to support the competitiveness of its company members (individually and collectively), explore new technologies, stimulate industry-relevant academic research, promote greater academic collaboration, and sustain a pool of experienced faculty and a pipeline of relevantly educated students. Since its inception in 1982, SRC has managed over $1.8 billion in basic academic research at more than 200 universities worldwide and supported over 10,000 students, who have gone on to become the next generation of leading-edge researchers, technology innovators and industry leaders. Processes and infrastructure developed by SRC identify and communicate industry's collective basic research needs, connect the academic faculty and student researchers with industry "users", support university research with high impact potential, and deliver early results to members.

Semiconductor Research Corporation