Security-Aware SoC Test Access Mechanisms

Kurt Rosenfeld
Google, Inc.
New York, NY
kuro@google.com

Ramesh Karri
Polytechnic Institute of NYU
Brooklyn, NY
rkarri@duke.poly.edu

***Abstract*—Test access mechanisms are critical components in digital systems. They affect not only production and operational economics, but also system security. We propose a security enhancement for system-on-chip (SoC) test access that addresses the threat posed by untrustworthy cores. The scheme maintains the economy of shared wiring (bus or daisy-chain) while achieving most of the security benefits of star-topology test access wiring. Using the proposed scheme, the tester is able to establish distinct cryptographic session keys with each of the cores, significantly reducing the exposure in cases where one or more of the cores contains malicious or otherwise untrustworthy logic. The proposed scheme is out of the functional path and does not affect functional timing or power consumption.**

I. Introduction

SoC design cycles are getting shorter and designs are getting more complex. Pressure for more productivity per designer per day has led to reuse of design modules and, in many cases, obtaining modules from external sources of intellectual property (IP). It is impractical, and sometimes impossible, for SoC designers to manually assess the security of each of the cores they use. Modular design methodologies hide the internals of the modules, which boosts productivity, but unfortunately it shifts designers from *knowing* the logic of the chip to merely *hoping* that each piece actually operates according to its interface specifications. Continued progress in VLSI requires that we not let complexity and short design cycles undermine our ability to produce chips that are trustworthy. In the long run, we need to work toward designing in such a way that the security failure of one module does not result in the security failure of the entire system. One step in that direction is the topic of this paper, to reduce the risk of cascading security failure in the test subsystems of chips we design.

Test access mechanisms (TAMs) are present in every complex chip and are used for a large and growing variety of purposes. Their original purpose was to enable efficient structural testing of digital systems instead of the notoriously inefficient process of black-box functional testing. As system design has evolved, the role of TAMs has been extended to include invoking BIST, programming nonvolatile memory, debugging embedded microprocessors, initializing the configuration of FPGAs, initializing volatile run-time configuration registers, and enabling and disabling system components including the TAM itself. A security-aware TAM improves upon conventional TAM concepts by protecting the data that traverses it, thus reducing the risks arising from untrustworthy cores.

A. Assumptions

The threat model addressed in this paper is that one or more untrustworthy cores intercept or modify the test data that passes between the tester and a core. We make the following assumptions:

1) The SoC contains many cores, some of which are untrustworthy.
2) The SoC contains trustworthy inter-core functional wiring.
3) The SoC contains trustworthy test access wiring.
4) Cores are connected to the tester by shared wiring. This can be either a daisy-chain scheme or a bus scheme.
5) Some or all cores, including their test wrappers, are opaque.
6) The SoC design is known to the attacker as much as it is known to the designer. If a crypto key is embedded in the design, the key is known to the attacker.

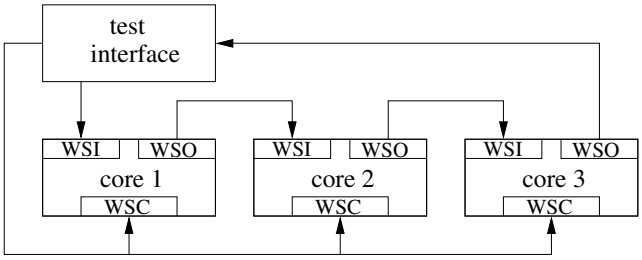

Fig. 1. Data sent from the test controller to core 2 passes through core 1, giving core 1 an opportunity to intercept it. Likewise, data passing from core 2 back to the tester passes through core 3, giving core 3 an opportunity to intercept it.

Some examples of TAM-related threats in a daisy-chained TAM architecture, as shown in Figure 1, include the following:

1) Core 1 sniffs data passing from tester to core 2. Core 1 leaks the data. Core 2 can be an FPGA or microprocessor, and the data can be its bitfile or executable program. This attack results in leakage of intellectual property which can result in monetary losses or expose the leaked data to reverse engineering and additional security exposure.
2) Core 1 modifies data passing from the tester to core 2. Core 2 can be a crypto core and the data can be a key. This attack can create a back door into the system.

B. Constraints

One way to reduce TAM-related risk is to use a star topology for test data. The star topology avoids the threats listed above by avoiding the placement of two cores on the same test wiring where they might interfere with each other. This topology is seldom used, since it results in high wiring cost. As the number of cores in SoC designs increases, star-topology TAM wiring becomes less and less practical. Instead, we reject that technique in favor of cryptography, which allows us simultaneously have the low cost of the bus topology and the good security of the star topology.

C. Core Test Wrappers

To facilitate design automation, modular design, and efficient reuse of cores, the cores are often delivered to SoC integrators in a *wrapped* form, which means that the complexity of their internal test structures is hidden behind some wrapper logic, which exposes a simplified, standardized interface to the outside. IEEE 1500 [1] defines this interface. Although wrapping can improve the productivity of SoC integrators, it requires the SoC integrator to trust the wrapper implementations provided by the core vendors. Wrapped versus unwrapped cores is a complex decision with security ramifications.

II. Prior Work

The threat of malicious inclusions in chip designs has been discussed at a technical level in the security and VLSI communities, and at a policy level in defense communities. DARPA has led US DoD research in the area, funding the TRUST program [2], which aims to ensure that critical government operations are able to source trustworthy chips. King [3] shows the construction of malicious modifications to a CPU design that give the attacker the flexibility to choose the details of the attack at run time, in the field, after deployment. Wang [4] presents a taxonomy of malicious hardware. Kim [5] examines the risks malicious modules in an SoC abusing their bus-master capability. They provide a security-enhanced bus arbiter that traps the bus transactions of rogue modules and inhibits them from further action by cutting power to the offending module. The risks of having malicious chips on a JTAG [6] chain were studied in [7], and the authors developed countermeasures.

The SoC TAM security problem differs significantly from the JTAG security problem. Primarily, the difference is that all cores of an SoC are fabricated together. The JTAG security solutions proposed in [7] assume that each chip exists first in isolation, gets packaged and tested, and then shipped to the customer. While in isolation, before being shipped, keys can be set up for use by the tester for secure communication over the untrustworthy test bus in the field. Since this isolated stage does not exist for SoC cores, the TAM security solutions developed for JTAG do not apply to SoCs.

III. Proposed Approach

We propose TAM enhancements that leverage the SoC designer's control over the inter-core wiring. The result is that untrustworthy cores are prevented from sniffing communication on the test bus. As with most communication security schemes, key exchange is a pivotal issue. Many cores are to a single test bus, and to single out a target core for communication, the TAM must provide the tester with a mechanism for securely distinguishing the target core from the rest of the cores. We propose that the SoC integrator construct a scan chain outside of any of the wrapped cores, and connect each core to the output of one of the cells in the scan chain. The foundation of the security of our system are the assumptions that:

1) A core, however malicious and devious it may be, cannot affect the intercore wiring of the SoC.
2) A core cannot control where in the intercore wiring its terminals are connected.

Thus, the tester securely distinguishes each core by which scan cell it connects to, as shown in Figure 2.

As stated in Section I-B, TAMs are under significant pressure to minimize their cost. For this reason, SoCs use shared wiring to connect the tester with the cores. During testing, the target core is addressed using any of various schemes, while the other cores are expected to be passive. We describe three scenarios where untrusted cores are placed on the shared TAM wiring. For each scenario, we discuss how the economy of shared wiring can be retained while protecting sensitive test data from sniffing as it passes by or through malicious cores. The first scenario is where a wrapped cores is obtained already containing the security-aware TAM enhancements described in this paper. The second scenario is where a wrapped cores is obtained without any TAM security enhancements, and has sensitive test data. The third scenario is where a core is obtained that is untrusted but no sensitive test data will be exchanged with it.

A. Security-enhanced Test Wrapper

We enhance the security of the standard core test wrapper by using cryptography to protect the data. Standard crypto primitives are used and the details of their design and implementation are outside the scope of this paper. Here, we focus on the practical aspects of key exchange and issues specifically relevant to the SoC test problem.

A common technique for session key establishment is to use the Diffie-Hellman [8] protocol. In Diffie-Hellman, each party generates a random number, sends a message, and does some arithmetic, and the result is that the two parties agree on a secret that was never explicitly transmitted over the wire. Although Diffie-Hellman is a powerful building block, for our application, we can obtain better security at lower cost by taking advantage of practical constraints on what a malicious core can do. Keys can be generated by the test controller or external tester, and distributed to each core at test initialization time as shown in Figure 2. Cheap and secure, this is our preferred key distribution scheme.

The SoC designer can choose between on-chip key generation and off-chip key generation. If done on-chip, the external tester (i.e., ATE) does not have access to keys, which

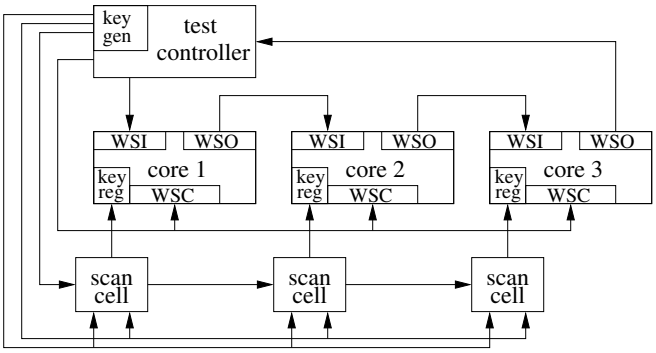

Fig. 2. A chain of scan cells is used for distributing keys to each of the cores. The scan cells are configured not to expose key bits at their outputs while they are being shifted.

improves security in some situations. However, for reasons of cost, as discussed in Section IV, in most circumstances we expect SoC designers to prefer off-chip key generation and cryptography. Key generation entails selecting a key for each core that has a security-enhanced wrapper. The most important characteristic of the keys is that no core should be able to learn the key of another core. As stated in Section I-A, we assume that the design is known to the attacker. This precludes the possibility of hard-coding the keys. Instead, for on-chip key generation, a hardware random number generator is used. Holleman [9] reported a hardware number generator requiring $0.031\,\mathrm{mm}^2$ of die area in $0.35\,\mu\mathrm{m}$ four-metal, double-poly CMOS. If implemented in a current fabrication process with smaller feature sizes, the die area for the circuit would be correspondingly smaller.

When key bits are transmitted to the cores during key setup, they are also stored by the test controller or external tester. The storage of key bits or cipher state is necessary because it is the basis for encrypted communication. However, the SoC designer has a choice of whether to maintain crypto sessions when accessing other cores. For example, if the test schedule involves communicating with core 1 and then with core 2, and then with core 1 again, the question is whether the test controller should maintain the crypto session (cipher state) associated with core 1 while accessing core 2. If it does, then it can resume communications with core 1 without the delay of reinitializing the cipher state associated with core 1. On the other hand, if the on-chip test controller is performing the crypto, then registers must be added to the test controller to store the cipher state, which increases the die area overhead. In the case of off-chip key generation and crypto, storing session state is not a problem. Offloading the crypto to the ATE is consistent with our goals and with the threat model stated in Section I-A.

The scan cells in the key setup scan chain, though very simple, provide two properties that are very important for security. First, as shown in Figure 3, they accept an output inhibit signal, O_INH*, which forces the output to zero when it is pulled low. This has the effect of blocking cores from observing other cores' key bits while they are being shifted in. Second, the logic gates are, for all intents and purposes, unilateral. There is no way for a core to actively force a value onto the flip-flop to affect the key bits that are received by other cores.

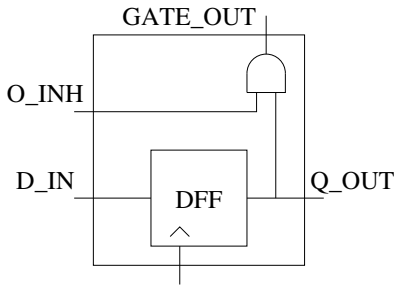

Fig. 3. The key setup scan chain conveys data from the test controller to the core wrapper key registers without allowing it to be sniffed or modified by other cores. Other than the basic distributed shift register functionality, the only extra functionality we require of our scan cell is an output inhibit input (O_INH) to ensure that the key is not leaked during shifting. After the tester has the key bits shifted to their intended location, the tester deasserts the output inhibit signal so that the cores receive their key data.

Communication requirements of cores vary over a wide range, and optimal test access design involves allocation of test buses and scheduling of tests [10]. Cores using BIST exclusively may be interfaced using only a serial test interface. Cores exposing extensive internal scan chains to the tester typically make use of a parallel test bus to increase test speed. In either case standard low-cost symmetric cryptographic modules are placed between the test interface and the core, as shown in Figure 4. The costs of these modules are discussed in Section IV.

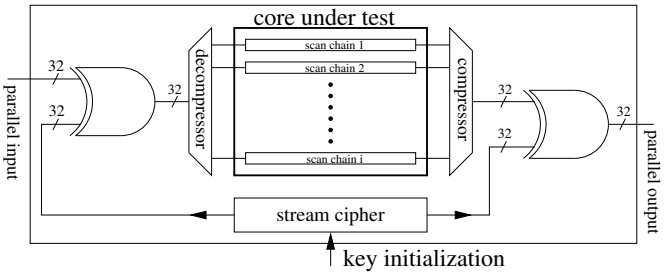

Fig. 4. In a typical security-enhanced wrapped core, a word of compressed test data arrives via the parallel data input, is decrypted instantaneously, decompressed and applied to the inputs of the core's scan chains. The outputs are compressed, encrypted, and sent out. Standard wrapper components are not shown, such as the parallel bypass.

If an untrusted core was obtained in an unwrapped state, the test wrapper described in this subsection should be added by the SoC integrator. Purely from a security standpoint, it is good to obtain cores in an unwrapped state. However, it adds significant work for the SoC integrator, losing the productivity benefits of test integration standards like IEEE 1500. Obtaining cores prewrapped with security-aware wrappers is probably the option most SoC integrators would prefer.

An untrustworthy core can be provided to the SoC integrator prewrapped with a security-enhanced wrapper. The presence of the security-enhanced wrapper does not make the core trustworthy, even if the wrapper itself is free of malicious features. However, the untrustworthy core, even if its wrapper contains malicious features, cannot undermine the TAM security of the SoC. This is in stark contrast with the conventional daisy-chain TAM architecture, where a single untrustworthy core breaks the security of all other cores on the chain.

B. A Security Overwrapper for Prewrapped Cores

In cases where an important module is only available as a conventionally wrapped core, a security *overwrapper* can be used. The functionality provided by the overwrapper, when combined with the core's included wrapper, is equivalent to that of the of the security-enhanced wrapper discussed in Section III-A. The security overwrapper contains functional blocks for decrypting input test data and encrypting output test data.

C. Interoperability with Noncompliant Cores

The TAM security scheme described here does not require that all cores in an SoC comply. Noncompliant wrapped cores can be used as they are, provided that no security-critical data passes over their test interfaces. The presence of noncompliant cores on the test bus does not undermine the security guarantees provided to the compliant cores.

IV. Costs

The SoC TAM security enhancements presented in this paper were designed to be efficient in terms of die area, test time, and effort for the SoC integrator.

A. Die Area Cost

The security enhancements contribute to die area in three ways. To a certain extent, these costs can be traded off, and the optimal choice depends on economics.

1) Wiring Cost: The extra wiring cost of the security enhancements is the cost of three extra wires on the test bus. In a typical SoC with serial control of the core wrappers and a 32-bit path for test data, there is a minimum of 40 wires without the enhancements, and 43 with the enhancements, a 7.5% overhead.

2) Core Wrapper Area: Each core whose test data the SoC integrator decides to protect needs to have its own crypto hardware, whether provided by the core supplier or by the SoC integrator. Assuming a 32-bit parallel test data path, the wrapper must decrypt 32 bits of input test data while encrypting 32 bits of output test data. To implement this with no additional latency, a stream cipher is used. 64 bits of keystream are required for each cycle of the test clock. This is achieved by using a keystream generator that produces multiple bits per clock cycle. The main additional hardware requirements for the security-enhanced core wrapper are:

- 32 XOR gates to decrypt the input test stimulus
- 32 XOR gates to encrypt the output test response
- a stream cipher that generates 64 bits of keystream per cycle of test clock

The Trivium [11] stream cipher meets the requirements. In its 64-bit form, it is equivalent to 5504 NAND gates. Assuming the XOR gates are equivalent to 2.5 NAND gates, the XOR gates used in each core wrapper are equivalent to 160 NAND gates. The total area overhead is therefore approximately 5700 NAND gates. For an SoC with n cores with security-enhanced wrappers, the gate count cost is

$$overhead = n \times 5700 \tag{1}$$

The percentage gate count overhead is

$$\left(\frac{\sum_1^n (g_i + 5700)}{\sum_1^n g_i} - 1 \right) \times 100\% \tag{2}$$

where g_i is the number of NAND gate equivalents in core i. For example, for an SoC an average of 100,000 NAND equivalent gates per core, the area overhead is 5.7%.

3) Test Controller Area: The cryptography can be handled by the test station or by the on-chip test controller. If it is handled by the test controller, it must have hardware for generating random key bits, and for storing them. It also needs the encryptor and decryptor blocks. If the cryptography is handled by the test station, the test controller's complexity is essentially the same as without the security enhancements.

If the designer prefers to perform the cryptography and key generation using the on-chip test controller, then there are three cases. In the first case, the test controller only maintains information about a single core at a time. This requires a key setup whenever addressing a new core. For this, the die area overhead in the test controller associated with the security-enhanced TAM is the area of the stream cipher and XOR gates, the equivalent of 5700 NAND gates. In cases where all of the key setup is done at initialization time, the die area overhead is $5700 + 12nk$ NAND gates, where n is the number of cores and k is the key length. The third case is where cipher state is maintained by the test controller for each security-enhanced wrapped core. Essentially, this means keeping a copy of the state register of the stream cipher, which is almost the same as the test controller simply having separate instances of the stream cipher module for each security-enhanced wrapped core with which it will communicate. The die area overhead of this is approximately $5700n$ NAND gates. This minimizes test time, but is the most expensive option in terms of die area.

B. Test Time

The security enhancements do not affect the test clock speed, test duration, or test scheduling. However, the security-enhanced cores need to have their key registers initialized before testing can commence. The worst-case test time overhead is the time to program all key registers, back-to-back. For a k-bit key and n cores in the SoC, the worst case key initialization time is

$$t_{init} = kn \tag{3}$$

The key setup clock frequency can be assumed to be the same as the test clock frequency. The percentage test time overhead is

$$\left(\frac{\sum_1^n t_i + k}{\sum_1^n t_i} - 1\right) \times 100\% \quad (4)$$

where t_i is the number of test clock cycles required to test core i. For example, assume an 80-bit key length and 50 cores in the SoC, the worst-case key initialization time is 4000 cycles of the key setup clock. If the average core requires more than 8000 bits of test data, the the test time overhead of the security enhancements is less than 1%.

C. Effort for SoC Integrator

Under the proposed scheme, cores fall into three categories:

1) no security-enhancements,
2) cores shipped with security-enhanced wrappers as described in Section III-A, and
3) cores that were shipped with non-security-aware wrappers and had the security overwrapper added as described in Section III-B.

The effort for the SoC integrator for category-1 cores is zero. The effort for using category-2 cores is very low. The signaling of the cores is consistent, so the per-core additional effort is just assigning the signals to connect the core to the key-setup scan chain. Category-2 is the preferred category, achieving all of the security benefits with minimal effort. Category-3 cores are slightly more effort for the SoC integrator, but still minimal work. The difference is that category-2 cores come with the crypto modules already in place whereas category-3 cores require the SoC integrator to add the crypto module between the core and the test bus terminals.

V. Conclusion and Future Work

We presented a scheme that eliminates the risk of a malicious SoC core sniffing test data. The essential contribution is a straightforward way of establishing cipher keys without any hard-coded secrets in the design. The area overhead is under 6% and the test time overhead is under 1%. For typical chips where a minority of the cores have secrecy-sensitive test data, only those cores need the security-aware wrapper, and the total area overhead can be correspondingly reduced to 3% or less. Additional area savings are available when the functional clock is more than 8 times higher in frequency than the test clock. In such cases, the Trivium stream cipher can be used in a configuration that produces 8 bits at a time instead of 64 bits at a time, and it can be clocked by the functional clock instead of by the test clock. This reduces the size of the main source of area overhead, the stream cipher, by 32%. We showed how the SoC integrator can use his control over the inter-core wiring to maintain the security of the test data. Future work using the same key setup scheme will provide not only secrecy guarantees for the test data, but integrity guarantees as well. As the number of cores in SoCs increases, and cores are obtained from an increasingly wide variety sources, limiting the damage done by a single rogue core has become an important concern with a practical solution.

References

[1] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur, "Overview of the IEEE P1500 standard," vol. 1, sep. 2003, pp. 988 – 997.

[2] D. Dean. R. Collins, "Trust, a proposed plan for trusted integrated circuits," http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA456459.

[3] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

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

[5] L.-W. Kim and J. D. Villasenor, "A system-on-chip bus architecture for thwarting integrated circuit trojan horses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1 –5, 2010.

[6] IEEE Std 1149.1-2001, Test Access Port and Boundary-Scan Architecture.

[7] K. Rosenfeld and R. Karri, "Attacks and defenses for jtag," *Design Test of Computers, IEEE*, vol. 27, no. 1, pp. 36 –47, jan. 2010.

[8] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644 – 654, nov. 1976.

[9] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio, "A 2.92 μW hardware random number generator," *Proceedings of the 32nd European Solid-State Circuits Conference*, pp. 134 –137, sep. 2006.

[10] K. Chakrabarty, "Optimal test access architectures for system-on-a-chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, pp. 26–49, 2001.

[11] C. D. Canniere and B. Preneel, "Trivium specifications," *ECRYPT Stream Cipher Project*, 2006.