Secure Split-Test for Preventing IC Piracy by Untrusted Foundry and Assembly

Gustavo K. Contreras, Md. Tauhidur Rahman, and Mohammad Tehranipoor
Dept. of Electrical & Computer Engineering
University of Connecticut
{gustavo.contreras, tauhid, tehrani}@engr.uconn.edu

ABSTRACT

Counterfeit ICs can have a major impact on the security and reliability of critical applications. This paper presents a method called Secure Split-Test (SST) for securing the manufacturing process to prevent counterfeits, allowing intellectual property (IP) owners to protect and meter their IPs. This is done by requiring test results to be verified by the IP owner and by requiring the IP owner to provide a "key" to unlock the IPs correct functionality. The results and analysis demonstrate that SST can effectively prevent counterfeited ICs from untrusted foundries or assemblies as well as its resilience to attacks and circumvention.

Keywords: Counterfeit ICs, supply chain, IP piracy, untrusted foundry and assembly, IP protection.

I. INTRODUCTION

As the size of the integrated circuit (IC) industry increases, and more ICs are fabricated off-shore, the size of the IC counterfeiting market has increased considerably. Intellectual property (IP) owners have to turn over their full IC design, as well as test patterns and test responses, to foundries to allow them to fabricate and test the ICs. The high cost of IP development puts the parties involved in IC manufacturing and testing in a position where it is possible to profit from exploitation of the IP they have been provided with. One example of such is if a foundry produces more ICs than they were commissioned to make, allowing them to sell these over-produced ICs for the low cost of the materials needed, without having to pay the high cost of the IP development [1]. Another example is if a foundry sells, rather than discards, the defective or out-of-specification ICs that they have produced. It is worth noting that a permanent or parametric defect can be very subtle and difficult to detect, causing the IC to appear functional despite a known error in rare cases [21].

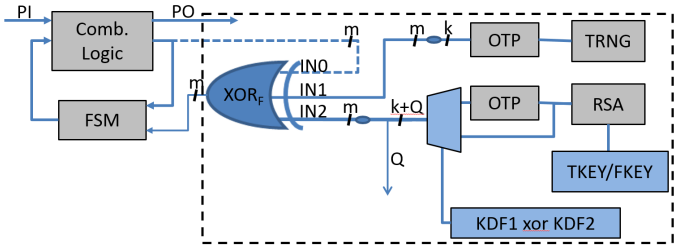
In general, counterfeit ICs represent serious reliability and security concerns, especially with regards to secure, life-threatening, or mission-critical applications [6]. Various techniques have been proposed as ways to combat IC counterfeiting over the past several years. For example, in electronic chip ID (ECID), counterfeits can be detected by giving each IC a unique ID and checking its ID against a database, with ICs not in the database being considered counterfeit. These IDs can be as simple as a bar code sticker [7], or they can be intrinsic to the IC, being produced by exploiting the process variations found in manufactured ICs [8]. Physical Unclonable Functions (PUFs) are a class of silicon hardware structures which produce different outputs in different ICs based on the unique process variations of the ICs they are used in [9]. Ring oscillator (RO) based PUF (RO-PUF) [10] and Arbiter PUF [9] can produce the same kind of static, yet unique and reliable identifiers using a challenge-and-response scheme. This allows the IP owner to maintain a secret challenge, which only they know and use, to identify ICs, making it more difficult for counterfeiters to tamper with or fabricate the identifier.

Another approach to prevent counterfeiting is by requiring that ICs be "activated" by the IP owner after being fabricated by the foundry. Several "active metering" techniques aim at preventing over-manufacturing by requiring that the foundry retrieve "passwords" from the IP owner after fabricating each IC [13] [14] [16]. By requiring the foundry to disclose the existence of every IC they would like to activate, the IP owner is able to "meter" the production of ICs.

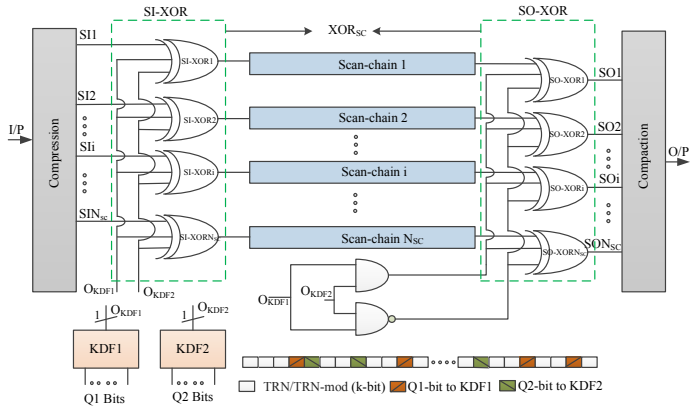
The above techniques address only part of the IC counterfeiting problem by untrusted foundry. Some basic implementations of IC identification techniques are easy to fake; even the more technically advanced PUF-based identification techniques, while making counterfeit detection possible, do little to actually *prevent* counterfeit production. The active metering techniques described above do attempt to prevent counterfeits from being produced; however, these techniques do not prevent production of all types of counterfeits because these techniques require that the IC be activated before it can be tested. The IP owner is required to provide the "key" to the IC before they know that the IC is not defective and is within specification. This can allow the untrusted foundry to ship/sell defective or out-of-spec ICs, which have already been activated by the IP owner. In addition, a foundry can request more keys than necessary from the IP owner by claiming a low yield, thus, the foundry can place many functional (defect-free) ICs in the market. Similar actions can be repeated by the assembly responsible for packaging, testing, and shipping the ICs to the market.

In this paper, we propose a novel technique which can both detect and prevent the creation and sale of over-produced, defective, or out-of-spec ICs by untrusted foundry and assembly. This technique called Secure Split-Test (SST) reestablishes trust into IC fabrication and test process by reintroducing the IP owner in the IC testing procedure without requiring them to be physically present at the foundry/assembly. By adding cryptographic functionality, unique binary identifiers, and combinational locking logic to an IC design, IP owners can create ICs which can be tested by the foundry and assembly, but whose test results can only be verified by the IP owner. Additionally, the additions that we have made to the IC design make it so that only the IP owner can generate the correct "key" that will unlock the IC's full functionality. These additions allow the IP owner to control the exact number of fully functional ICs released to the market, and ensures unauthorized ICs are non-functional and can be easily detected if they are shipped to the market. The proposed on-chip structure can also prevent cloning (the chip is reverse engineered or IP is pirated for fabrication in a different facility) and remarking (assigning a different grade to the chip) because of the unique keys stored on-chip. SST can also point to the source of the problem in terms of the entity involved in shipping such counterfeited parts.

The rest of this paper is organized as follows: Section II gives a general overview of the Secure Split-Test technique and how hardware and communication interact to secure the ICs' testing flow. Section III presents SST's two main hardware components. Section IV describes the communication between the IP owner and foundry/assembly in order to make the IC fabrication and test flow secure. Section V analyzes different aspects of SST such as area overhead, test coverage, test time, and attacks. Finally, we conclude the paper in Section VI.



(a) Functional-locking block with $m = pk$ expansion.



(b) Scan-locking block.

Fig. 1. Secure Split-test structure.

II. Secure Split-Test Overview

Typically, IP owners have little interaction with their ICs after they provide the foundry with the IP in the form of a GDSII file, test patterns, and corresponding responses. Tested dies are sent to the assembly to be packaged and tested again. Often, the ICs are sent directly from assembly to the market or to the IP owner's vendors. It is the responsibility of the assembly to ship only good ICs to the market in the volume requested by the IP owner. Provided this background, our main objective in this work is to develop a new technique that allows IP owners to control the test process without being physically present at the foundry or assembly. Reasserting control over these processes ensures that no overproduced ICs, defective ICs, or out-of-spec ICs can be sent to the market. The Secure Split-Test technique described in this work ensures that, if those ICs are sent to the market, not only will they be non-functional but they will also be easily detected by either the IP owner or the system integrators using those ICs.

SST introduces trust into the fabrication process by addressing the untrusted production flow in two dimensions, (i) adding the SST structure to the original design and (ii) enhancing communication between foundry/assembly. Through added structure, SST secures against *cloning* and with the added communication between the IP owner and foundry, SST directly involves the IP owner in the testing process to prevent shipping *over-produced*, *defective*, and *out-of-spec ICs*. The added communication allows the IP owner to "see" the ICs as they are being tested and gives the IP owner control over the decision of which ICs pass and which should be discarded. The IC will only become functional once the IP owner has decided that it has passed the necessary tests at assembly. The IP owner merely needs to start the SST process and the process of testing, deciding, and activating ICs will run automatically.

SST secures the test processes of ICs by inserting two main hardware blocks in the design; the functional-locking block and scan-locking block. The functional-locking block's purpose is to ensure that only unlocked ICs will have the correct functionality. IC's will only function correctly when the IP owner sends the correct functional key. The functional-locking block is made up of three smaller hardware blocks, an XOR_F mask for functional locking, true random number generator (TRNG), and RSA decryption logic, as shown in Figure 1(a). SST inserts the XOR_F mask on non-critical paths in the circuit. These XORs have three inputs connected to circuit paths (IN0), TRNG output (IN1), and RSA output (IN2). An XOR_F is transparent only when both inputs coming from the TRNG and RSA outputs are the same; otherwise it acts as an inverter. The TRNG output is different for each IC but it is constant throughout the IC's lifetime since it is stored in a one-time programmable (OTP) memory. The RSA component receives an encrypted key from the IP owner, this key is decrypted and connected to IN2. The circuit only becomes functional when the appropriate RSA input is applied; this gives the IP owner control over when/whether to activate the IC. The encryption and transmission of the key make up the communication component of SST. A server owned by the IP owner receives TRNs for each die and creates test keys (TKEYs), it then receives test outputs and compares them to the expected outputs to determine which dies have passed or failed. The same procedure is followed during assembly. The server decides whether an IC has passed the necessary tests after assembly and only then, it sends the functional key (FKEY) to activate and unlock the IC. The following sections provide details about the SST structure and communication.

The scan-locking block ensures an untrusted party cannot scan out functional results from an IC in an attempt to modify, bypass, or attack the SST hardware. Even when the IC has received its functional key and is functioning correctly, the scan-locking block will prevent any attacker from applying patterns and observing the IC's responses. The scan-locking block is made up of XORs at the scan chain inputs and outputs and two odd-logic functions to determine the key being applied to the circuit.

III. SST Structure

A. Functional-Locking Block

XOR Mask: The "XOR_F mask" is a series of m 3-input XOR gates which are inserted into non-critical paths of an, otherwise, unmodified circuit. XOR_Fs have three inputs IN0, IN1, and IN2. While IN0 is connected to circuit paths, XOR_Fs receive m-bit inputs as IN1 and IN2 with potential to modify the circuit. If the two inputs, IN1 and IN2, are the same, that particular part of the XOR_F mask will act as a buffer. If the two inputs are different, the XOR_F will act as an inverter. The placement of these XOR_Fs into the circuit dictates how they will affect the circuit. Hence, XOR_Fs are placed at the inputs of the scan flip-flops in the circuit. Random XOR_Fs at the scan flip-flop inputs, as shown in Figure 2, invert the circuit's response as it is being captured by the scan flip-flops. The effect of having the inverting XOR_Fs at the scan flip-flop inputs is that some scan flip-flops may be capturing an inverted value of the actual response. Exactly which flip-flops are affected is determined by the two m-bit inputs IN1 and IN2. This property is useful as it means that the ICs can still be tested, and the test results are related to IN1 and IN2. Knowing the values for IN1 and IN2 allows the IP owner to know which test outputs should be inverted. Finding the correct test responses requires simple bit flipping and has negligible test time overhead.

True Random Number Generator: Various true random number generators (TRNGs) have been designed [2] [4] for insertion in ICs. In digital circuits, TRNGs use physical phenomena such as clock jitter, temperature, power supply noise, etc., as a source of entropy. True Random Numbers (TRNs) cannot be predicted or algorithmically created by an attacker, even if the attacker has access

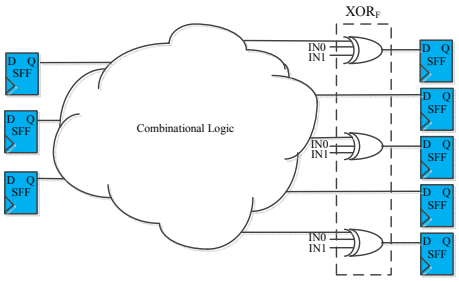


Fig. 2. XOR mask of functional-locking block (XOR_F) inserted at flip-flop inputs.

to the design. An important quality of a TRNG is its unpredictable randomness. Since TRNGs have no stability requirement, they are usually smaller, and less complex than PUFs. TRN will be different for each IC but must remain constant throughout the IC's lifetime. However, TRNGs output different TRNs every time they are accessed and PUFs will not provide a stable and unique output every time it is activated due to its sensitivity to noise, temperature, and aging [9][10]. To address this issue, the first time TRNG is accessed after manufacturing the TRN value will be stored into a one-time programmable memory (OTP) or polyfuse; the XOR_F input IN1 will therefore be connected to this memory rather than TRNG directly. Using OTP to store TRN solves the issue of stability with TRNG or PUF since the value in memory will always be constant. With an m-bit XOR_F, mask IN1 must be an m-bit random value. In order to reduce memory size and area overhead, a smaller k-bit random value can be used, k-bit value is then be repeated p times to enable m XOR_Fs ($m = p.k$).

In addition to the SST implementation, TRNGs can also be used for passive identification as a unique identifier (e.g. ECID) for each IC. Unique identifiers are used to track ICs throughout their lifetime and also verify their authenticity once they are already in the market.

RSA Asymmetric Encryption: The third input of XOR_Fs, IN2, is connected to the output of the RSA block. The RSA block is important to secure the IC and prevent the foundry from figuring out how the decision of passing or failing an IC is made. The RSA asymmetric cryptographic algorithm is a public-key cryptographic system, which means that the encryption and decryption processes are performed using different keys. Hence, we use an industry-verified secure implementation of the RSA algorithm that implements the many security standards relating to the algorithm, such as the PKCS #1 standard [3].

The on-chip RSA performs modular multiplication to decrypt the access key generated by the IP owner. During manufacturing, an RSA public key is embedded into the design in read-only memory; this public key will be the same for all circuits. Due to the asymmetric nature of RSA, using the same public key or allowing the foundry to read the public key does not pose a risk to SST as [19] and [20] have proven that computing the RSA private key from its corresponding RSA public key is as complex as factoring RSA's modulus n into its prime factors.

RSA gives the IP owner control over the XOR_Fs' IN2 inputs; the IP owner can take TRN and modify it in such a way that only the IP owner knows. This new modified TRN (TRN-mod) will be encrypted using RSA and thus invisible to the foundry, this test key (TKEY), once applied to the IC will invert some of the responses coming from XOR_Fs with different IN1 and IN2. Since only the IP owner knows what TRN-mod is, only the IP owner can tell which response bits will be inverted.

Once an IC has passed testing at assembly, the IP owner can encrypt the unmodified TRN into a functional key (FKEY). This FKEY will be decrypted by RSA, the results will be stored in OTP memory (see Figure 1(a)), to make it permanent, and the correct IN2 values will be applied to the XOR_F inputs, thus making the IC functional.

A detailed description of the function of TKEY and FKEY is given in Section IV.

B. Scan-Locking Block

The details of a scan-locking block is presented in Figure 1(b). In scan-locking block, the inputs to some scan chains are inverted when FKEY is applied and make the input bits transparent when TKEY is applied. At the same time the output from some of the scan chains can be inverted when TKEY is applied and transparent otherwise. This block is introduced so an attacker cannot extract any information from an unlocked IC. The scan-locking block consists of 3-input XOR gates inserted at scan chain inputs (SI-XOR) and outputs (SO-XOR) and two key-determining functions (KDFs). Each KDF is composed of XOR gates forming an XOR odd-function circuit which outputs $'1'$ if the input is odd (odd number of $'1's$) or $'0'$ if it is even (even or zero number of $'1's$). Two KDFs, KDF1 and KDF2, are used to detect the type of key has been provided by IP owner and determine the function of the scan-locking mechanism. The output of KDFs, O_{KDF1} and O_{KDF2} are fed to the 3-input XOR gates through some logic gates to make the XORs transparent (when $KDF1 \neq KDF2$) or inverting (when $KDF1 = KDF2$). The Q bits needed to control each KDF are added to TRN and TRN-mod by the IP owner. Q is divided into two parts Q1 and Q2 as inputs for KDF1 and KDF2, respectively. The position of Q additional bits is (see Figure 1(b) random and known only by the IP owner since TRN-mod and TRN are encrypted and only TKEY and FKEY are visible by the foundry.

The number of $'1's$ in $Q1$ and $Q2$ determines whether it is an FKEY (odd number of $1's$ in either Q1 or Q2 and even number of $'1's$ in the other) or a TKEY (either odd or even number of $1's$ in both Q1 and Q2). The length of $Q1$ and $Q2$ is an important factor to increase the level of security. The key can be scrambled with high possibilities if the length of $Q1$ or $Q2$ is increased. Adding one more bit in $Q1$ or $Q2$ requires an additional 2-input XOR in odd-function logic circuit, which increases the possible combinations exponentially and hence improves the security significantly. A FKEY (O_{KDF1}=0/1 and O_{KDF2}=1/0) inverts the $SI-XOR$s to invert the input of all scan chains and invert the response by $SO-XOR$s. As a result, an attacker cannot get the correct response from an unlocked IC with FKEY. Even flush patterns will come out of scan chains unaltered, making it difficult to predict the scan-locking block's functionality. On the other hand, a TKEY(both O_{KDF1} and O_{KDF2} are $'0'$ or $'1'$) transfers the unaltered pattern into the scan chains but the response is inverted as $SO-XOR$ is inverting leading to a wrong response. The scan-locking structure does not have any impact on circuit timing because it is not a part of the functional circuitry.

The complete operation of SST is summarized in Table I. It shows different functionality of XOR blocks when TKEY and FKEY are used. With TKEY, $Inv.(F_a(I))$ is performed. The unaltered input pattern, I, is applied to the altered combinational circuit, F_a, with active XOR_Fs. The output responses are captured and inverted by $SO-XOR$s. With FKEY, $Inv.(F_u(inv(I)))$ is performed. The input pattern, I, is inverted before it is applied to the unaltered combinational circuit, F_u. The outputs are again inverted by selectively $SO-XOR$s.

In order to add another level of security, both $SI-XOR$s and $SO-XOR$s can be activated in a random number of scan chains. Suppose, both $SI-XOR$ and $SO-XOR$ are inserted in total N_{SI-XOR_T} among

TABLE I
SUMMARY OF SST OPERATION

KEY	SI-XOR	XOR_F	SO-XOR	Operation
FKEY	Inv.	Tran.	Inv.	$Inv.(F_u(Inv.(I)))$
TKEY	Tran.	Inv.	Inv.	$Inv.(F_a(I))$

Inv.: Inverting, Tran.: Transparent

N_{SC} scan chains. Without knowing the exact $N_{SI-XORr}$, an attacker needs to try

$$\sum_{r=0}^{N_{SI-XORr}} \binom{N_{SC}}{N_{SI-XORr}}$$

different attempts to find out the location of XORs. The design is supposed to provide better hamming distance as only $N_{SI-XORr}$ scan chains alter the input pattern and output pattern depending on the type of key.

In addition, to lower the area overhead, inverter, SO-INV, can be used in place of $SO-XOR_{SC}$ as all $SO-XOR_{SC}$s invert the response further. And for that, no additional logic circuitry is required after KDFs.

IV. SECURED COMMUNICATION

Secure Split-Test enhances communication between the IP owner and foundry. The IP owner gets test results from the foundry and determines whether an IC is operating correctly. IC design becomes more secure because it gives the IP owner the decision over passing or failing ICs without the need of being physically present. Figure 3 shows the communication flow between the foundry and the IP owner. The entire process runs unassisted, the automation is explained at the end of this section.

The first level of communication which is present in all manufacturing process is the transfer of the GDSII file to the foundry. The foundry then proceeds to create masks from the GDSII file provided and uses them to fabricate the wafers. The IP owner provides the foundry with the test patterns at this stage. For each die under test, the TRNG will be activated and a TRN for the die can be retrieved and stored in OTP; the foundry then sends the TRN to the IP owner. The IP owner stores the TRN value and creates TRN-mod. TRN-mod is a modified version of TRN with random bits inverted. The location of the inverted bits are only known by the IP owner. Additionally, Q bits needed to control the scan-locking block are inserted randomly into TRN-mod as well. TRN-mod is encrypted using the IP owner's private RSA key to create a Test Key (TKEY) as shown in Figure 3.

TKEY is sent to the foundry to be used on the die. The IC's internal RSA structure decrypts TKEY using the embedded public RSA key to obtain TRN-mod and the inserted Q bits. The on-chip bus coming from RSA's output separates the Q bits and applies them to the scan-locking block; consequently, TRN-mod is applied to the functional-locking block. In the functional-locking block, the values going into IN1 (TRN) and IN2 (TRN-mod) of XOR_F are different due to the bits the IP owner inverted. XOR_Fs with different IN1 and IN2 will, therefore, have inverted outputs. The scan-locking block uses the Q bits to determine the type of key being applied to the circuit, in this case, TKEY. The foundry can proceed to test the die and collect its perturbed results.

Since output response is perturbed by TRN, TRN-mod and SO-XORs, the foundry cannot determine whether the die is functioning correctly. The response for each die must be sent to the IP owner to be verified. The IP owner, knowing TRN and TRN-mod, and SO-XORs, knows which outputs should be inverted and can therefore compare those to the expected responses.

In order to reduce the size of data being transferred between the IP owner and the foundry or assembly, a signature of the circuit

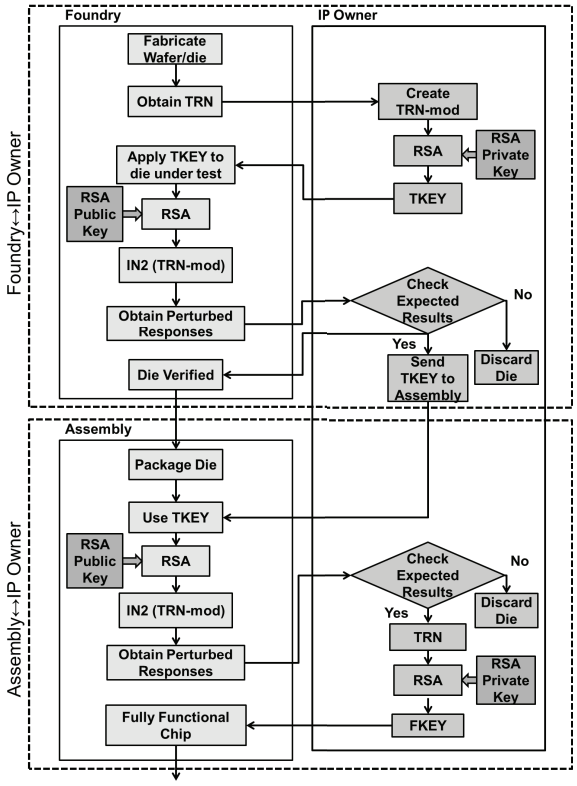


Fig. 3. Communication flow between IP owner, foundry, and assembly.

outputs is generated using software. The IP owner can compact the captured responses using a MISR-like compactor. When the foundry tests the ICs, a similar software attached to the tester captures the responses and creates a signature in the same manner as the IP owner did. This signature is significantly smaller than the complete set of captured responses and contains the information needed to verify each IC. Creating this signature greatly reduces the data size and test time needed to compare circuit responses on the IP owner side. The IP owner can determine whether the die is working correctly or not by comparing the signatures. Note that different TRN-mods and TKEY for different dies, will result in different output responses and signatures. If the die passes the test, the IP owner lets the foundry know to pass the die; otherwise it asks the foundry to discard it. Passing dies are then sent to assembly to be packaged.

The IP owner sends the assembly the TKEY for each passing die. TKEY can be the same as the one given to the foundry or a new one can be generated using a different TRN-mod. The assembly packages the die and tests the IC with the given TKEY and sends the signature to the IP owner. When an IC passes test, the IP owner generates a functional key (FKEY) based on the correct value of TRN and the corresponding Q-bits indicating FKEY is being applied. FKEY is decrypted by RSA and unlocks the IC's correct functionality. The decrypted FKEY is burned into OTP to permanently unlock the IC. The communication with assembly is shown at the bottom of Figure 3.

FKEY is only sent after testing and therefore gives the IP owner control over how many ICs are activated. The foundry could request additional TKEYs to test the ICs if their yield is low but cannot request FKEYs to unlock the ICs. Any over-produced ICs will not be functional because the proper FKEY can only be obtained from the IP owner using their private RSA key. Defective and out-of-spec ICs cannot be sent into market or marked as passing ICs because the decision of passing or failing an IC is made by the IP owner, and

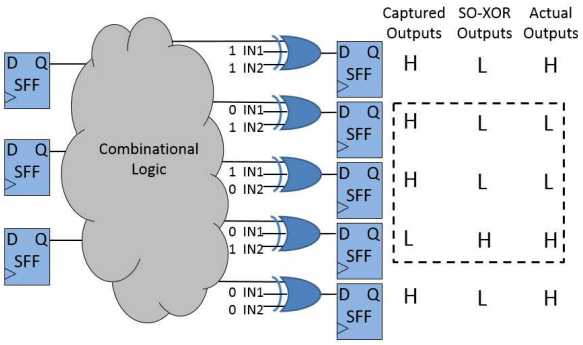


Fig. 4. If IN1 (TRN) and IN2 (decrypted TRN-mod) do not match, the XOR_F mask will flip the corresponding bits, the flipped outputs will get captured. Here, IN1=10100 and IN2=11010.

only those that have passed receive an FKEY. In addition, TKEY and FKEY are dependent on the IC's TRN and will be different for each IC. Modifying each TRN will cause the test outputs collected by the foundry to be different for each IC; an untrusted foundry will not be able to find out the actual test responses because they will differ from IC to IC.

We have fully automated the SST's communication flow. All the handshakes needed between the foundry/assembly and IP owner run from a server and do not require additional resources except for server's memory and minimal processing power used to invert bits, compare values, and encrypt TRN and TRN-mod. Once the server receives the TRN value, the modification to obtain TRN-mod is a simple random bit-flipping procedure. The process of comparing foundry collected test results to the actual test results requires a simple comparison between the signatures obtained by the IP owner and the signatures received from the foundry.

Figure 4 shows how TRN and TRN-mod change the captured outputs from actual outputs. The circuit's TRNG outputs $IN1 = 10100$ as TRN which is stored into OTP. The foundry then sends the TRN to the IP owner's server. The server is programmed to randomly flip bits; here, bits 1, 2, and 3 are flipped to obtain TRN-mod=11010. TRN-mod is encrypted by the server to create TKEY which is sent to the foundry to be applied to the die. The IC's built-in RSA logic decrypts TKEY into TRN-mod which is connected to IN2 at the XOR_Fs. Since $IN1 = TRN = 10100$ and $IN2 =$ TRN-mod $= 11010$, the three middle flip-flops will invert their test responses. Assuming the captured test response to the fully functional circuit is HLLHH, which is only known by the IP owner and is stored in the server, the foundry will see the responses inverted by $SO-XOR$ and the three middle bits flipped by XOR_F, i.e., XOR_F responses: HHHLH, $SO-XOR$ will shift out LLLHL. This test response is sent to the IP owner's server which has the actual response stored. The server compares the received response, LLLHL, and recovers the response HLLHH to compare it to the actual response HLLHH. In this case, the die is marked as a passing die because the responses match. The same flow will be applied after it is packaged and if the test response is the same, a "go" signal is given to assembly and an FKEY will be generated encrypting the actual TRN value to activate the IC.

V. Results and Analysis

A. Overhead Analysis

Area Overhead Analysis: The physical size of each block added to the circuit is related to the RSA key size k since RSA can only encrypt data as large as its key size. A block implementing a k-bit RSA process takes k-bit inputs and produces k-bit outputs. The k-bit output is what is being used to configure the XOR_F mask along with the TRNG block. This means that the TRNG block needs to produce a TRN output of k bits as well. However, the number of XOR gates used for the XOR_F mask can be greater than k, the XOR_F mask can be of size m as long as $m = pk$ where p is an integer. We can broadcast the k-bit TRN and RSA outputs and repeat them p times to connect all m XOR_Fs in the mask. Choice of m will be implementation-specific depending on the size of the circuit we are introducing this technique to. Finally, we need ROM storage for the RSA public key which takes k bits of storage. In practice, there are restrictions on the values of k that we can choose. Typically, RSA modules are large powers of two. Additionally, researchers have been able to factor RSA keys of up to 768 bits, meaning that the minimum recommended key size is currently 1024 bits. As an example, we examine the area requirements of a 1024-bit implementation of this technique. RSA implementations have radically different areas based on their goals and features. Here, we use an RSA implementation which is optimized to minimize area overhead. One such implementation required 14K gates [15].

TABLE II
Area overhead for $k = 1024$ and $N_{SC} = 1000$

		Area Overhead		
$p = m/k$	m	1M gates	10M gates	100M gates
1	1024	2.0%	0.20%	0.020%
2	2048	2.1%	0.21%	0.021%
5	5192	2.4%	0.24%	0.024%
10	10240	2.9%	0.29%	0.029%

The ring-oscillator based TRNG presented in [4] is implemented here. The size of TRNG does not need to increase if a larger k is used, the additional bits only require a longer time to generate. An additional $k = 1024$ bits of OTP memory is needed to store the TRN value.

In scan-locking block, a Q1-input XOR is required to implement KDF1 and a Q2-input XOR is required for KDF2. Total $2N_{SC}$ 3-input XORs, 1 inverter, 1 AND gate and 1 NAND gate are required in scan-locking block. Area overhead for different p values (assuming $k = 1024$) and design sizes ($1M$ to $100M$ gates) when $N_{SC} = 1000$ scan chains are shown in Table II. In Table II, Column 1 is the broadcasting ratio p (see Figure 1(a)), and Column 2 represents the size of the XOR_F mask; The area overhead in Columns 3, 4, and 5 are only affected by the size of the XOR_F mask while RSA, TRNG, and memory remain the same (each implemented to support an RSA key size k). For large designs, it is not necessary to use a larger TRNG or RSA; their outputs can simply be expanded using a higher broadcasting ratio p.

Coverage Analysis: Due to the added circuitry, there will be additional faults introduced to the IC. Fault simulation done on RSA has shown that this circuit can achieve 95.83% test coverage with only 960 random patterns. These results show that the same random tests used on the IC in built-in self test (BIST) mode can be used to test RSA and obtain a high fault coverage. Note that generally it is recommended to use BIST for testing cryptographic hardware in a secure IC because of its increased security over test application from tester [22].

As for TRNG, no test is necessary to detect faults in it. The purpose of TRNG is to create random values, these values are stored in memory. Any faults in TRNG do not matter since its output is random. The XOR_F masks will introduce four faults per XOR after fault collapsing. In the given example of an XOR_F mask of 1024 XORs, the number of faults added will be 4096. Since the XOR_F mask is inserted at the input of scan flip-flops, i.e., at the output of the combinational logic, these faults are highly observable , which indicates they are closest to the scan flip-flops capturing their responses and are easy to detect.

Test Time Overhead Analysis: SST can increase test time due to the additional communication with the IP owner's server as described

in Section IV. It is required that the foundry or assembly testing the ICs send the TRN and test signature back to the IP owner. It is also required that the IP owner sends back keys to test the ICs with a pass or fail responses depending on the test results. IP owner also needs time to compute the test keys and to determine whether or not the test results are correct.

The communication automation tool was designed using MATLAB R2012a [18]. The communication was simulated on a computer with a 2.30 GHz Core i7 Intel processor with 8GB of RAM. The simulation was repeated 1000 times to obtain an average value of the total simulation time. The MATLAB tool showed an average of 339ms to read, generate keys, and encrypt TKEY from TRN-mod. Bit flipping to obtain TRN-mod had negligible time. The additional time needed to encrypt FKEY at the server is not considered because FKEY can be encrypted by the server while the chip is being tested at the assembly. Other processes such as generating the actual responses should not require any additional time because they can be generated while the ICs are being fabricated or tested. For multiple ICs, total test-time overhead can be reduced if the tool runs on a server with multiple threads because each thread can be used for one chip.

At the foundry, the first part of the SST flow is generating TRN right after fabrication. The implemented TRNG has a rate of 2.5Mbps, at 40MHz a 1024-bit TRN will take 0.41ms. On the chip, the implemented RSA working at 40MHz can decrypt the message, TKEY, in 325ms. The total test-time overhead for a TRN of size 1024 is $0.41 + 325 = 325.41ms$ at the foundry. The assembly will have an additional 325ms due to the time needed to decrypt TKEY and FKEY by the built-in RSA. Assuming test times of 1, 2, 5, and 10 seconds for four different chips, the test time overhead will be 32.5%, 16.3%, 6.5%, and 3.25% respectively.

We acknowledge that test application time can significantly affect the total test cost, and the time-to-market of an IC. We believe that, for modern large designs with high volumes, the reduction in counterfeiting, the increase in system reliability, and the decrease in device returns created by the use of this technique will justify the increase in test time.

B. Hamming Distance Analysis

The SST structure was simulated using Synopsys' logic verification tool VCS [17]. The IC was a synthesized implementation of the ISCAS'89 benchmark s38417 with 28 inputs, 106 outputs, 8709 gates, 13470 inverters, 1570 scan flip-flops separated into 10 scan chains. The average hamming distance between the actual response and captured response from functional-locking block and scan-locking block is presented in Table III and Table IV, respectively. In the functional-locking block, XOR_F mask of different sizes $m = 64, 128, 192$ and 256 were inserted at the input of scan flip-flops of non-critical paths in the circuit, and each of the 64 ICs had their own unique and distinct values for IN1 and IN2 applied to the XOR_F mask. 157 stuck-at-fault patterns generated by ATPG are applied to scan chains, and the response from each SO is observed for each combination to ensure the inserted XOR_F masks changed the captured test values differently for each combination. These results confirm the effectiveness of the XOR_F mask at locking an IC's functionality and at obfuscating the correct test responses since IC has shown to output different responses when the mask is inverting outputs. Table III shows that m is an important factor in perturbing the correct test response. Response perturbation is enhanced by increasing the size of m as flipping happens more often. Result shows that inverting 16% of total XOR_F is good enough to reach ideal Hamming distance, 50%. On the other hand, Hamming distance improves if XORs are inserted in random scan chains, instead of all scan chains. Result, Table IV, shows that $N_{SI-XORr}$ affects the Hamming distance dominantly which improves the level of security. The result shows that Hamming distance increases with the number of SI-XORs. The result shows that the goal, 50% Hamming distance, can be achieved by inserting SI-XOR and SO-XOR in less than 50% of total scan chains.

TABLE III
HAMMING DISTANCE ANALYSIS FOR FUNCTIONAL-LOCKING BLOCK

m	64	128	192	256
HD	58.02%	57.12%	55.52%	53.87%

TABLE IV
HAMMING DISTANCE ANALYSIS FOR SCAN-LOCKING BLOCK

$N_{SI-XORr}$	HD
1	08.09%
2	15.73%
3	25.16%
4	32.24%
5	40.41%
6	46.46%
7	53.08%
8	60.20%

C. Attack Analysis

Overall, SST significantly increases the security of the IC supply chain. It is worth analyzing the possible attacks on this technique and the security that SST provides. Different attacks would include: (1) attacks on the design of the technique (direct attacks), (2) attacks to modify the netlist (tampering attacks), (3) attacks which attempt to deceive the IP owner or avoid the technique (circumvention attacks), (4) hardware-based attacks that attempt to remove the SST blocks (removal attacks), and (5) Unlocked IC attacks.

Direct Attacks: Secure split-test is relatively resilient to direct attacks. Each IC requires one key to reach a fully functional state, and from a hardware perspective it would be easy to have a single output pin which indicates whether or not the IC is in that state. The problem of finding a key that puts the IC into a fully functional state is equivalent to the problem of bypassing RSA. An attacker who tries to bypass this technique would have two options: (i) randomly generate potential keys in the hopes that they find one which works for a known TRN, or (ii) factor the public modulus into its component primes so that they can find the private key themselves and instantly generate the correct key. Both of these attacks are proven to be difficult, the first one would require billions of iterations since a key of length 1024 would have 2^{1024} possibilities; the second attack is equivalent to attacking RSA which has not been done yet and is extremely difficult as shown in [19][20].

Circumvention Attacks: Attacks which try to bypass SST do not fully defeat the technique. If an attacker has knowledge of which XOR_Fs and SO-XORs in the circuit have been activated by the key/TRN combination used, the attacker could figure out which responses have been inverted. The attacker could change the responses obtained in order to make the IC pass the test. The IP owner would then pass the IC and send a go signal for that IC. This attack could be done at three different stages, at the foundry, at the assembly, or both foundry and assembly working together. If the attack occurs at the foundry, the bad IC will be sent to the assembly, where the same tests are applied using the same TKEY. This time, the IC will fail and the IP owner knows to discard the IC. Additionally, the IP owner will see that the foundry sent a failing IC to the assembly, raising a red flag about the integrity of the foundry. If the attack occurs at the assembly, the IP owner will be able to know results are being changed because the same tests were done by the foundry using the

same TKEY. Any mismatch in results between foundry and assembly can detect an attack on the IC and the IP owner will know to discard and not send a final FKEY for that IC. In the third case of both the attack occurring by both foundry and assembly, using the same TKEY will not prevent the attack. However, if different TKEYs are used for the foundry and assembly, the attacker has to figure out the difference between the TKEYs in order to know which outputs to change, this task is the same as cracking RSA which, as mentioned before, is infeasible for large keys.

Tampering Attacks: An attacker could try to re-route the outputs from TRNG to go directly to the output of RSA, bypassing the RSA decryption needed to activate the IC. We acknowledge that this attack could compromise the system; however, rather than defining separate blocks for each component, the components can be synthesized and optimized with the design in order to obfuscate the components and their functionality, i.e., generate a flat netlist.

The XOR_F mask can also be hidden into the design during synthesis. To prevent the attacker from figuring out the location of the XOR_F mask, other gates or a combination of gates (NAND, NOR, etc.) can be used instead of XORs. Using other gates would not change the design or effectiveness of SST, but would make it almost impossible to find which gates belong to SST.

Obfuscation makes the task of finding the individual SST components difficult and helps prevent attacks. However, it is important to note that attacks to re-route nets or bypass gates from the XOR_F mask require a high degree of sophistication such as high technical background, access to the design files, knowledge of circuits functionality, multiple resources, and time. These aspects of the attack alone deter many of the counterfeit attacks because overproducing or selling defective ICs may not require any level of sophistication.

Removal Attacks: It is possible that the foundry may try to remove some or all of the hardware needed by this technique. Exactly how much they can remove depends on how much they know about the logic design of the IC. For example, they could not blindly remove any XOR gate whose output connects to a flip-flop input; they would have to know whether or not the XOR gate was part of the XOR_F mask. Attacks aimed to tamper with or remove the TRNG block or RSA block would have to be very carefully designed to avoid detection. This is especially true because, as specified, this technique implements a basic metering methodology that requires foundries to report each IC back to the IP owner and requires that the IP owner provide a working key for the IC. Attacks that altered the way that the TRNG or RSA blocks worked would also have to avoid communications with the IP owner, as the TRNG and RSA blocks directly affect the scan output during testing.

Unlocked-IC Attacks: SST is resilient to attacks using unlocked ICs. With the scan-locking block, the IC manufacturers cannot run the test vectors on the unlocked IC to determine the response as the inputs of some scan chains are inverted using $SI-XOR$ if FKEY is applied; which lead to wrong response. At the same time the response of some scan chains are flipped using $SO-XOR$s in some scan chains to change the response intentionally when TKEY is applied in an unlocked IC. As a result, an attacker cannot obtain any information from an unlocked IC.

D. Integrity Analysis

It is worth considering what would happen if a foundry or assembly were to attempt to place ICs utilizing SST into the market without trying to defeat the technique. The most important aspect is that, as we have demonstrated, unauthorized ICs will not have been provided with the correct FKEY that they require to function correctly. Overproduced ICs, ICs with defects, or out-of-spec ICs that a foundry or assembly might try to place into the market will be easy to detect for two reasons. First and most important, any IC sent to the market without a proper FKEY will not function correctly, this would be easily detected as soon as the IC is used for the first time because it will not be usable. This allows for easy detection of counterfeits because the client can contact the IP owner who, by simply obtaining the TRN value, will know that the IC is a counterfeit. Second, the IP owner maintains a database of all known and authorized ICs with respective TRNs. This allows counterfeit detection by randomly sampling the market and checking the TRN values to look for unauthorized ICs, including clones. Certain specifications will not cause failures in the captured responses, these chips could seem like working chips to the IP owner. The use of sensors on chips that can measure these specifications and turn them into digital responses will allow for these specification failures to be represented in the ICs captured responses, the IP owner will then be able to more efficiently prevent out-of-spec ICs from being activated or sent to the market. Finally, using this technique IP owner can tell what entity (foundry, assembly, or a third party cloning at the chip) has placed a counterfeit IC in the market.

VI. Conclusion

We have presented a technique that allows IP owners to be involved in IC manufacturing and testing process. Secure Split-Test eliminates the ability of attackers to create counterfeit ICs. SST introduces security measure at both the beginning of testing and at the end of testing in order to ensure that only the IP owner has control over which chips are sent to the market fully activated. Our result and attack analyses demonstrate the efficiency of the proposed technique.

References

[1] "Defense Industrial Base Assessment: Counterfeit Electronics", http://www.bis.doc.gov, U.S. Department of Commerce Bureau of Industry and Security Office of Technology Evaluation, 2010

[2] B. Jun, P. Kocher, "The Intel random number generator", Intel Corporation, 1999.

[3] "PCKS #1 v2.1: RSA Cryptography Standard", ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, RSA Security, 2002.

[4] B. Sunar, W.J. Martin, and D.R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks", *IEEE Transactions on Computers*, vol. 56, num. 1, pp.109-119, 2007.

[5] H. Livingston, "Avoiding Counterfeit Electronic Components", *IEEE Transactions on Components and Packaging Technologies*, vol. 30, num. 1, pp.187-189, 2007.

[6] J. Stradley and D. Karraker, "The Electronic Part Supply Chain and Risks of Counterfeit Parts in Defense Applications", *IEEE Transactions on Components and Packaging Technologies*, vol. 29, num. 3, pp.703-705, 2000.

[7] K. Chatterjee and D. Das, "Semiconductor Manufacturers Efforts to Improve Trust in the Electronic Part Supply Chain", *IEEE Transactions on Components and Packaging Technologies*, vol. 30, num. 3, pp.547-549, 2007.

[8] K. Lofstrom, W.R. Daasch, and D. Taylor, "IC identification using device mismatch", in proc. *IEEE International Solid-State Circuits Conference (ISSCC)*, pp.372-373, 2000.

[9] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions", in proc. *9th ACM Conference on Computer and Communications Security (CCS '02)*, pp.148-160, 2002.

[10] G.E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation", in proc. *44th ACM/IEEE Design Automation Conference (DAC '07)*, pp.9-14, 2007.

[11] J. Guajardo, S.S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protection", in proc.*9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '07)*, pp.63-80, 2007.

[12] F. Koushanfar, G. Qu, M. Potkonjak, "Intellectual Property Metering", in proc. *4th International Workshop on Information Hiding (IHW '01)*, pp.81-95, 2001.

[13] Y.M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security", in proc. *16th USENIX Security Symposium*, pp.20:1-20:16, 2007.
[14] J.A. Roy, F. Koushanfar, and I.L. Markov, "EPIC: Ending Piracy of Integrated Circuits", in proc. *Design, Automation and Test in Europe 2008 (DATE '08)*, pp.1069-1074, 2008.
[15] Z. Keija, X. Ke, W. Yang, and M. Hao, "A novel ASIC implementation of RSA algorithm", in proc. *5th International Conference on ASIC (ICASIC '03)*, pp.1300-1303, 2003.
[16] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design Test of Computers*, vol. 27, num.1, pp.65-75, 2010.
[17] Synopsys VCS: Functional verification tool, Synopsys; version 2010.03.
[18] MATLAB version 7.14.0. Natick, Massachusetts: The MathWorks Inc., 2012.
[19] J.M. DeLaurentis, "A further weakness in the common modulus protocol for the RSA cryptoalgorithm," *Cryptologia*. pp. 253-259, 1984.
[20] G.L. Miller, "Riemann's hypothesis and tests for primality," *Journal of Computer and Systems Sciences*. pp. 300-317, 1976.
[21] M. Bushnell and V. Agrawal, Essentials of Electronic Testing for Digital, Memory, and Mixed-signal VLSI Circuits, Kluwer Academic, 2000.
[22] D. Karaklajic, M. Knezevic, I. Verbauwhede, "Low Cost Built in Self Test for Public Key Crypto Cores," Workshop on Fault Diagnosis and Tolerance in Cryptography pp.97-103, 2010.