A

FORTIS: A Comprehensive Solution for Establishing Forward Trust for Protecting IPs and ICs

UJJWAL GUIN, University of Connecticut
QIHANG SHI, University of Connecticut
DOMENIC FORTE, University of Florida
MARK M. TEHRANIPOOR, University of Florida

With the advent of globalization in the semiconductor industry, it is necessary to prevent unauthorized usage of third party IPs (3PIPs), cloning and unwanted modification of 3PIPs, and unauthorized production of ICs. Due to the increasing complexity of ICs, system-on-chip (SoC) designers use various 3PIPs in their design to reduce time-to-market and development costs, which creates a trust issue between the SoC designer and the IP owners. In addition, as the ICs are fabricated around the globe, the SoC designers give fabrication contracts to offshore foundries to manufacture ICs and have little control over the fabrication process, including the total number of chips fabricated. Similarly, the 3PIP owners lack control over the number of fabricated chips and/or the usage of their IPs in an SoC. Existing research only partially addresses the problems of IP piracy and IC overproduction, and to the best of our knowledge there is no work that considers IP overuse. In this paper, we present a comprehensive solution for preventing IP piracy and IC overproduction by assuring forward trust between all entities involved in the SoC design and fabrication process. We propose a novel design flow to prevent IC overproduction and IP overuse. We use an existing logic encryption technique to obfuscate the netlist of an SoC or a 3PIP and propose a modification to enable manufacturing tests before the activation of chips which is absolutely necessary to prevent overproduction. We have used asymmetric and symmetric key encryption, in a fashion similar to Pretty Good Privacy (PGP), to transfer keys from the SoC designer or 3PIP owners to the chips. In addition, we also propose to attach an IP digest (a cryptographic hash of the entire IP) to the header of an IP to prevent modification of the IP by the SoC designers. We have shown that our approach is resistant to various attacks with the cost of minimal area overhead.

General Terms: Hardware Security, Trust, Intellectual Property

Additional Key Words and Phrases: IP Overuse, IC Overproduction, Supply Chain, 3PIP, System-on-Chip, Encryption

ACM Reference Format:
U. Guin, Q. Shi, D. Forte, and M. Tehranipoor, 2015. FORTIS: A Comprehensive Solution for Establishing Forward Trust for Protecting IPs and ICs *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 19 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The persistent trend of device scaling has enabled designers to fit more and more functionality on a system-on-chip (SoC) to reduce overall area and cost of a system. As the complexity has grown exponentially, it is fairly impossible to design a complete system by a SoC designer alone. Therefore, the semiconductor industry has shifted gears to the concept of design reuse rather than designing the whole SoC from scratch. Nowadays, the SoC designers obtain licenses for various functional blocks (known as intellectual properties or IPs) for their SoCs to optimize the design process and decrease time-to-market.

In parallel, the increased complexity of the fabrication process has resulted in a majority of SoC designers no longer maintaining a fabrication unit (or foundry) of their own. Building and maintaining such fabs for modern SoCs are reported to cost more than several billions of dollars and

Author's addresses: U. Guin and Q. Shi, Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA; D. Forte and M. Tehranipoor, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.

DOI: http://dx.doi.org/10.1145/0000000.0000000

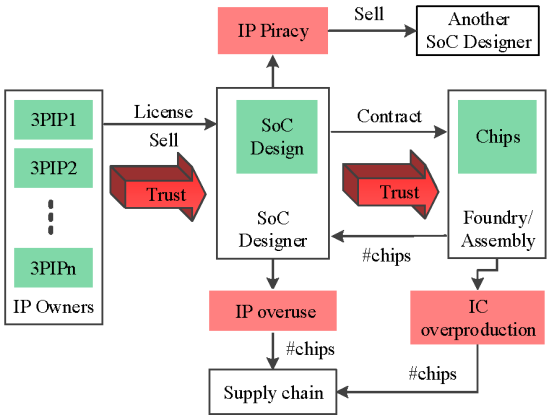

Fig. 1. Lack of trust between various entities involving in SoC design and fabrication.

increasing as technology further scales [Yeh 2012]. Given the increasing cost, the semiconductor business has largely shifted to a contract foundry business model (horizontal business model) over the past two decades. In this business model, the SoC designers first get licenses for 3PIPs to be used in their SoC designs, design the SoCs by integrating the various 3PIPs and then outsource the SoC design to the foundries and assemblies for fabrication and packaging to reduce time-to-market and manufacturing costs.

In the modern SoC design and fabrication flow, forward trust does not exist between the participating entities. The IP owners cannot have complete trust on the SoC designers, whereas the SoC designers may not trust the foundries or assemblies. The lack of transparency and the resulting lack of trust may lead to the following vulnerabilities, as shown in Figure 1.

• *IP overuse*: The SoC designer may produce more ICs and report a lesser amount to the IP owners to reduce the licensing cost. At the same time, the SoC designer may illegally use an IP that was licensed to be used in a different design. In short, the IP owners have no or little means to verify how many chips have been fabricated with their IPs and where they have been used.

• *IP piracy*: A SoC designer may legally purchase a 3PIP core from an IP vendor and then make clones, or illegitimate copies of the original IP [Castillo et al. 2007] [Kahng et al. 2006] [Chakraborty and Bhunia 2009] [Tehranipoor and Wang 2012]. Similarly, untrusted foundries may sell illegal copies of the GDSII files that they receive from SoC designers for fabrication. Further, the integrity of the IP may be at risk. An untrusted SoC designer can add some extra features to those 3PIPs to make them look like a different one and then sell them to another SoC designer. An SoC designer may also modify a 3PIP in order to introduce a backdoor or hardware Trojan into the chip.

• *IC overproduction*: Untrusted foundries and assemblies may produce more than the number of chips they are contracted to manufacture [Koushanfar and Qu 2001] [Roy et al. 2008] [Tehranipoor and Wang 2012]. As no R&D cost are incurred for these chips, they can receive illegitimately larger profits by selling these chips with the name of SoC designer. In addition, they can also overbuild chips practically at zero cost by reporting a lower yield (i.e., percentage of defect-free chips to the total number of chips) to the SoC designer [Contreras et al. 2013] [Rahman et al. 2014].

When an untrusted party overuses the IPs or overproduces the ICs and sells in the open market, the IP owners or the SOC designers lose any possible revenue that could have been gained from those chips. However, an even bigger concern with these ICs is that of reliability. An IC that uses a pirated IP may create a backdoor to leak secret information to the attacker or disable a system at some critical point in time. In addition, overproduced ICs may simply end up in the market with minimal or no testing for reliability and functionality. These ICs may also find their way into the

supply chain for many critical applications, which raises concerns for safety and reliability. Since these ICs have the same name of the SoC designers, their failure would tarnish company reputation.

1.1. Related Work

Existing research cannot fully address the problem and can be classified into three major categories.

• *Logic obfuscation*: This is a technique where a design is transformed to a different one to obfuscate the inner details of the original design, thus preserving the original functionality [Zhuang et al. 2004]. In [Chakraborty and Bhunia 2009], the authors proposed a methodology which can be integrated into the SoC design and manufacturing flow to simultaneously obfuscate and authenticate a design. In this approach, the circuit operates in a normal mode when it receives a predefined sequence of patterns, known as a key, at its input. However, it is not clear how this key will be hidden from the foundries or assemblies as it is necessary to prevent overproduction. In addition, this technique does not address IP overuse.

The authors in [Roy et al. 2008] first proposed to encrypt a netlist by using a lock (a sequence of XOR/XNOR gates) and it can only be unlocked by using a chip unlock key (CUK). The design is not resistant to reverse engineering as key gates are directly related to the key bits (XOR and XNOR gates indicate 0 and 1 at CUK location, respectively) and vulnerable to key sensitization attacks [Rajendran et al. 2012]. The authors in [Rajendran et al. 2012] addressed those problems by proposing different logic encryption techniques. The authors in [Subramanyan et al. 2015] has shown that any logic encrypted circuit can be broken. However, they assume that *an attacker can use scan-chain to read/write the values of all flip-flops in the design.* This assumption does not conform to today's designs. Every design now use test compression architecture to significantly reduce the test cost by reducing test time and test data volume [Synopsys 2015b; 2015d; Nagaraj 2015]. Test responses are compacted many folds before it becomes available for off-chip access. As the modern EDA tools provides diagnostic support (high defect coverage and accurate fault diagnostics) with compression in place [Synopsys 2015b; 2015d; Nagaraj 2015], it is impractical not to incorporate test compression in the design. It is now impossible to access the individual flip-flop values (the output of the combinational circuit, Y for a solution to a QBF) for chips where the design uses test compression. It is impossible to find a key using the approach suggested by the authors by looking at the compacted scan output values. Thus, it is still safe to use the scheme proposed in [Rajendran et al. 2012] to encrypt netlist.

Recently, Design Automation Standards Committee of the IEEE developed the standard P1735 [DASC 2014] to provide the guidance for encryption and management of IPs, which has been adopted by most IP and EDA vendors. In the encryption approach, the IP is encrypted with a random symmetric session key. This session key is then encrypted with the public keys of different EDA vendors and attached to the IP such that these vendors can later reconstruct the original IP. Figure 2(a) shows a very simple IP which performs *AND* operation in every clock cycle. To protect from any unwanted modification, the IP is encrypted by using Synopsys *encryptP1735.pl* script [Synopsys 2014]. In this encryption process, the code inside the *'pragma protect* block (encircled in red in Figure 2(a)) will be encrypted. The encrypted IP is shown in Figure 2(b), where the code inside the *'pragma protect* block (encircled in red) is not recognizable to anyone. During decryption, the session keys are decrypted by using the private key of the EDA vendor and then encrypted portions of the IP is decrypted by using this session key. One can find this process in detail in [Synopsys 2014] [Microsemi 2014]. Unfortunately, this encryption approach cannot prevent placing additional features to an existing IP as it does not provide any integrity verification. Figure 2(c) shows this modified encrypted IP where the attacker adds an extra feature (*OR* operation) to the existing one (*AND* operation). We will provide a solution by adding an IP digest resulted from a cryptographic hash function [NIST 2012] in the IP header (see Section 2.4) to prevent any unauthorized modifications. In addition, the encrypted IP does not provide any protection against copying of the whole IP to make an exact clone. As our solution uses an encrypted netlist, copying the entire IP will not help an attacker unless he possesses a valid CUK (see Section 2.1).

```
module secret (a, b, clk, y);
    input a,b,clk;
    output y;
`pragma protect begin
    reg y=0;
always @(posedge clk) begin
    y = a & b;
end
`pragma protect end
endmodule
```

a) An IP.

```
module secret (a, b, clk, y);
    input a, b, clk;
    output y;
`pragma protect data_block
EopI45Xzfd9MCrd6bQoMzZpf0J5X7U0e
Q3SiPiPt3uuZhYazA5xZQS+aD7ySO+rV
Q/msi5/CE5VsxtCSc8SVhFeniUj1sp
IjMKpVHvv+2pw=
`pragma protect end_protected
endmodule
```

b) Encrypted IP.

```
module secret_modified (a, b, clk, y, q );
    input a, b, clk;
    output y, q;
assign q=a | b;
`pragma protect data_block
EopI45Xzfd9MCrd6bQoMzZpf0J5X7U0eQ3SiPiPt3uuZhYazA5xZQS+aD7ySO+rVQ
/msi5/CE5VsxtCSc8SVhFeniUj1spIjMKpVHvv+2pw=
`pragma protect end_protected
endmodule
```

c) Modified encrypted IP.

Fig. 2. Modification of an encrypted IP to add extra features.

• *Hardware watermarking*: This approach has received much attention in the recent years for validating the authorship of an IP. Watermarking techniques uniquely identify an IP by creating a unique fingerprint in it [Charbon 1998] [Kahng et al. 2001] [Qu and Potkonjak 2003] [Castillo et al. 2007] [Lach et al. 2001] [Kirovski et al. 2006]. As the watermarking technique is passive, one cannot use it to prevent IP overuse, IP piracy, and IC overproduction. Rather, it can only be used to verify proof of IP use.

• *IC metering*: The existing metering approaches prevent IC overproduction by attempting to give an SoC designer control over the number of ICs manufactured. These approaches can be either passive or active. Passive approaches uniquely identify each IC and register the ICs using challenge-response pairs. Later, suspect ICs taken from the market are checked for proper registration [Lofstrom et al. 2000] [Lee et al. 2004] [Kumar et al. 2008] [Su et al. 2007] [Suh and Devadas 2007] [Koushanfar et al. 2001].

For passive metering techniques, one major limitation is that they cannot "actively prevent" overproduction. PUF-based detection techniques relies on the matching unclonable IDs/signatures generated by the PUF. The challenge-response pairs are stored in a secure database and verified later whether the responses are listed in the database or not. In the context of our problem, the SoC designers have to count on the foundries/assemblies to send them all defect free chips and trust them blindly on yield information. An untrusted foundry/assembly can hide actual yield information and practically build huge amount of defect free chips. They can literally send these unregistered chips to many different places (subsidiary companies, rogue system integrators, and many others who look for low cost parts). These untrusted entities might not care about the authenticity of ICs.

Active metering approaches lock each IC until it is unlocked by the SoC designer [Roy et al. 2008] [Alkabani and Koushanfar 2007] [Chakraborty and Bhunia 2008] [Alkabani et al. 2007] [Huang and Lach 2008] [Baumgarten et al. 2010]. The PUF-based active metering technique presented in [Koushanfar 2012] has an applicability limitation. In this proposed scheme, foundry needs to capture the initial power-up state through the scan-chains and send that state to the design house to receive the passkeys. The authors did not provide solutions with test compression architecture in place. Test compression is being adopted by the community to significantly reduce test data not out of preference, but of necessity. Every designs now use test compression architecture [Synopsys 2015c]. Test responses (the flip-flop values) are compacted many folds before it becomes available for off-chip access. It is impossible to access the flip-flop values unless there is a bypass of the

compression module. This may create additional overhead. Similar analysis is also applicable to the scheme presented in [Alkabani and Koushanfar 2007; Alkabani et al. 2007].

In [Contreras et al. 2013] [Rahman et al. 2014] the authors proposed secure split-test (SST) to prevent overproduced, out-of-spec/defective, and cloned ICs. SST enables the design house to participate in the manufacturing test process by placing a set of security measures in the design and controlling the test flow. However, the major disadvantage is the back and forth communication between the SoC designer and the foundry/assembly, which increases delay in the test process.

In [Roy et al. 2008], the authors used an on-chip TRNG to generate a public-private key-pair for RSA encryption. This approach suffers from three major issues. First, there is large design overhead due to the on-chip RSA key generation. The keys (e.g., 1024 bit to achieve 80-bit security) are derived by a complex algorithm from two large prime numbers, p and q, which are generally 512 bits long each [Paar and Pelzl 2009]. A prime checker is also required to verify these numbers are indeed prime. Second, the scheme assumes a secure transfer of public key from the chip to the SoC designer *which creates a vulnerability to man-in-the-middle attacks*. The foundry can always intercept the public key from the chip (more interestingly, foundry initiates the communication) and replace it with a new key, which nullifies the objective of creating on-chip key-pairs. Third, the scheme suffers from the key sensitization attacks [Rajendran et al. 2012].

1.2. Contributions

In this paper, we will present ***FORTIS***, a comprehensive solution for establishing forward trust for protecting IPs and ICs against the attacks discussed above. We address each issue as follows:

• *IC overproduction*: We develop a novel communication protocol for activating chips after fabrication. The protocol is similar to Pretty Good Privacy (PGP) by Phil Zimmermann, which is commonly used today in email delivery systems and has demonstrated excellent secrecy over the years [Kurose and Ross 2001]. In our approach, the design is locked by using a set of key gates and can only be unlocked upon receiving a CUK. To encrypt a design by using a CUK was first introduced in [Roy et al. 2008]. An improved version which is resistant to reverse engineering and various attacks, was presented in [Rajendran et al. 2012]. FORTIS uses one of this attack resistant encrypted netlist to prevent IC overproduction.

The major challenge here is to transfer this CUK to the chip from the SoC designer without being intercepted by any untrusted party (including untrusted foundry). Our proposed approach addresses this problem of key transfer from SoC designer to the foundry/assembly. Every chip has two static RSA keys (same for all chips) and a dynamic session key (different for everyone). Our approach does not require on-chip key generation which significantly reduces the area overhead compared to previous techniques.

As discussed above, prior approaches also have major limitations when testing is performed. Either the chip has to be unlocked [Roy et al. 2008] or test responses to be sent to the SoC designer [Contreras et al. 2013] [Rahman et al. 2014] [Tehranipoor et al. 2015] create additional vulnerabilities in the design flow. In our proposed approach, it is not required to provide CUK during test pattern generation (see Section 2.1). This helps us to perform manufacturing tests without unlocking the chips. Our proposed approach does not impact manufacturing tests and prohibit unwanted activation of ICs during test.

• *IP overuse*: We address IP overuse by introducing a trusted authentication platform (TAP) in the SoC. This TAP is trusted by all parties involved in the SoC design, and can be imported as a trusted third party IP. In our proposed approach, each IP is locked with key gates. The synthesis and test pattern generation flow is very similar as before. To the best of our knowledge, our proposed IP metering approach addresses the third party IP (3PIP) metering problem for the first time in a forward trust manner.

• *IP piracy*: We use IP encryption [DASC 2014] in our design flow to obfuscate the netlist. We propose IP integrity verification (see Figure 9) to make it resistant to modification, whereby the malicious SoC designer/foundry cannot modify a 3PIP by adding/disabling features. Along with this the netlist is locked by using a set of key gates to prevent the cloning of IPs. One question

that could arise is that if a 3PIP is locked by using a secret CUK, then how an SoC designer will simulate an SoC which uses that 3PIP. We address this issue by attaching CUK to the IP header and then encrypt it by using EDA tool's public key such that the tool can retrieve the CUK during simulation.

Table I. Comparison of different approaches to ensure forward trust.

Scheme	IP Overuse	IP Piracy		IC Overproduction		Resistant to Attacks
		Detection	Prevention	Detection	Prevention	
Logic Obfuscation	✗	✓	✗	✗	✗	Low
Hardware Watermarking	✗	✓	✗	✗	✗	Low
IC Metering	✗	✗	✗	✓	✓	Low
FORTIS	✓	✓	✓	✓	✓	High

Table I shows the summary of our contributions compared to existing research. IP piracy and IC overproduction are both categorized into detection and prevention categories. An IP/IC can only be detected as pirated/overproduced by the detection approaches, whereas prevention approaches prevent pirated IPs or overproduced ICs from entering into the supply chain. Our proposed approach, FORTIS, addresses the challenges for establishing forward trust, whereas the other approaches try to address the problem partially. The 3PIP owners protect their IPs from untrusted SoC designers and foundries when they use FORTIS in their design flow. Similarly, the SoC designers protect their SoCs from untrusted foundries. Further, FORTIS is the only approach that prevents modifications to any 3PIPs. Our proposed design flow inherently assures forward trust in the SoC design and fabrication process.

The rest of the paper is organized as follows. In Section 2, we describe our proposed approach, FORTIS, to address IC overproduction and IP overuse. We also address IP piracy by untrusted SoC designers and foundries. We present our results and attacks analysis in Section 3. We conclude the paper in Section 4.

2. FORTIS: ARCHITECTURE AND COMMUNICATION

With increasing SoC design complexity, design reusability has become an integral part of the SoC design process. Unfortunately, this creates the risk of overuse of 3PIPs by untrusted SoC designers and foundries. In addition, the SoC designers lose profits once an untrusted foundry/assembly overproduces chips and sells them under their name. Thus, forward trust is extremely important to the entities involved in the SoC design. The IP owners need to trust the SoC designer, whereas the SoC designers must trust the foundries and assemblies. Our proposed design flow automatically ensures this forward trust among these entities.

Figure 3 shows our proposed design flow for establishing forward trust between various entities involved with the SoC design process. The design flow is very similar to the existing IC design flow except for the lock insertion and functional activation steps. Our design process starts with the insertion of locks by using a set of key (XOR/XNOR) gates using an existing secure logic encryption technique [Rajendran et al. 2012], where the authors already has investigated how/where to insert these gates. The circuit produces functionally correct output when it receives a chip unlock key (CUK). The number of XOR or XNOR gates depends on the level of security one wants to achieve. We now modify the gate level netlist to enable manufacturing tests before the activation of chips.

Each 3PIP owner inserts key gates to lock their design and then generates test patterns. The SoC designer receives all these locked IPs and integrates them in the design. The SoC designer also inserts a lock in one of the in-house IP to protect against IC overproduction. The SoC designer collects all the test patterns from different IP owners and stores them in a pattern repository for future wafer and package tests. As all the 3PIPs are locked, the simulation may be a challenge for a SoC designer. We address this in Section 2.4.

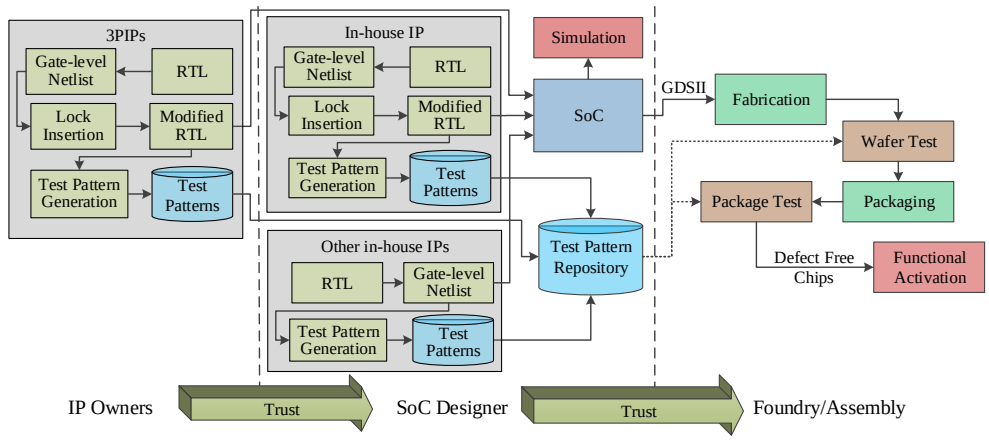

Fig. 3. FORTIS for enabling IC/3PIP metering to ensure forward trust in the SoC design and fabrication.

The GDSII file corresponding to the SoC design is now sent to the foundry. The foundry first processes wafers, which generally contains hundreds of dies in a single wafer. Foundry then performs wafer test to inspect dies to find gross defects. If there are too many dies on a wafer that are defective, the foundry sometimes rejects the whole wafer. After wafer tests, the defect-free dies are sent to assembly for packaging. The good chips are then sorted out by using package tests and the chips that have been damaged during the packaging process are discarded. Our proposed design flow does not modify the existing fabrication, packaging and test processes. Finally, each chip is unlocked using a valid CUK by the entity who perform the final manufacturing test (foundry, assembly, or SoC designer) before supplied to the market.

2.1. Enabling Structural Test without Activation

It is absolutely necessary to activate the chips after the tests have been performed, which will prevent an untrusted foundry/assembly to pile up defect free ICs by hiding actual yield to the SoC designer. In this section, we will present an architecture that enables structural tests before the activation of chips.

In the previously proposed architectures, the structural test patterns are generated considering a predetermined CUK value. This is due to the existent of forward implication of the key gates. A forward implication exists when the inputs of a logic gate are assigned in a way that the output is uniquely identified [Bushnell and Agrawal 2000]. For a two input XOR gate, the other input must be specified to either 1 or 0. If we do not assign a value at $CUK[i]$, the ATPG tool will consider this input as X, and all the faults before the gate k_i (logic cone shown in shaded grey color) will be untestable due to the non-existence of the forward implication.

Let us illustrate this point with an example by considering a fault D, shown in Figure 4(b). This fault will be testable if it is being propagated to the output Y_{1m}. If $CUK[i]$ is 1 then the output of the gate $k1$ becomes $\bar{D}$, otherwise it becomes D. The corresponding Y_{1m} will be $\bar{D}$ or D depending on the $CUK[i]$.

$$Y_{1m} = \begin{cases} D & if\ CUK[i] = 0 \\ \bar{D} & if\ CUK[i] = 1 \end{cases}$$

To maintain a forward implication, we need to provide a CUK value during test pattern generation. Thus, the previously proposed designs need a CUK (for example, $CUK[i] = 0$ or $CUK[i] = 1$) before the structural test pattern generation to test all the faults before the key gate. It is now necessary to load the same key into the chips before the manufacturing test begins. If we activate the chips before manufacturing tests then the objective of preventing overproduction will

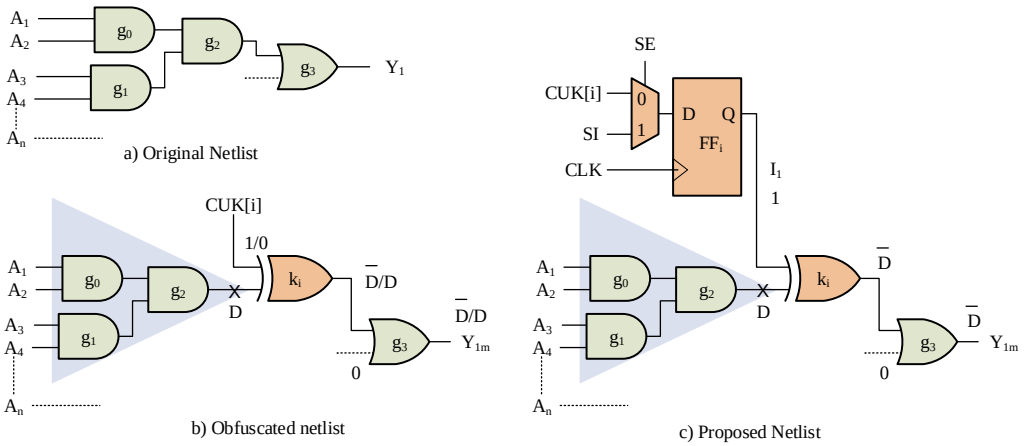

Fig. 4. An obfuscated netlist with two XOR/XNOR gates.

be unsatisfied. An untrusted foundry/assembly can overbuild chips by asking for more keys and reporting a lower yield to the SoC designer.

In our proposed netlist, the ATPG tool assigns a unique value at the I_1 input to maintain the forward implication (assign 1 for example) for the key gate to transfer the fault $\bar{D}$ to the output Y_{1m}. Thus, the ATPG tool can generate test patterns without knowing the key. In this paper, we refer structural or scan test patterns as patterns. These patterns will be used later during wafer and package tests to find defect free chips from a manufacturing unit.

Figure 4(c) shows our proposed netlist, where the key bit $CUK[i]$ is connected to a scan flip-flop (FF_i). The output of FF_i drives the key gate k_1. In the test mode, when the scan enable (SE) signal is asserted, this flip-flop becomes a part of the scan chain. The ATPG tool generates test pattern for this modified netlist with $n+1$ inputs $(A_1, A_2, ..., A_n, I_1)$ rather than the original netlist (Figure 4(a)) with n inputs $(A_1, A_2, ..., A_n)$ or obfuscated netlist (Figure 4(b)) with n inputs $(A_1, A_2, ..., A_n)$ and $CUK[i] = 0/1$.

Let us now consider key sensitization attack presented in [Rajendran et al. 2012]. In key sensitization attack, the key bits are treated as Xs and propagated to the output. As the unlocked chips contain 0 or 1 at a key bit location, these key values are visible at the output and the attacker can recover the key. For traditional DFT, where there is no compaction of test responses, the key sensitization attack works. However, this attack may not be feasible in any design, which uses an on-chip test response compaction module. On-chip test response compaction is very common in today's designs [Synopsys 2015b; 2015d; Nagaraj 2015]. Almost every chip uses response compaction to significantly reduce test data not out of preference, but of necessity.

Figure 5 shows an example of a compressor logic structure with a compression ratio 2. The effect of Xs (FFs captured the key bits) will be suppressed at the output $dout$ if at least two of the inputs of the XOR gates in the compressor are Xs. In this example, we can select scan chain 3,4, and 5. At i^{th} clock cycle three key bits $(k-1, k, k+1)$ will be at $dout$ simultaneously and their individual effect cannot be separated.

$$
\begin{aligned}
dout[0] &= din[4] \oplus din[3] \oplus din[2] \oplus din[0] = k \oplus (k-1) \oplus \ldots = X \oplus X \oplus \ldots \\
dout[1] &= din[5] \oplus din[3] \oplus din[2] \oplus din[1] = (k+1) \oplus (k-1) \oplus \ldots = X \oplus X \oplus \ldots \\
dout[2] &= din[6] \oplus din[5] \oplus din[4] \oplus din[2] = (k+1) \oplus k \oplus \ldots = X \oplus X \oplus \ldots \\
dout[3] &= din[7] \oplus din[5] \oplus din[4] \oplus din[3] = \ldots \oplus (k+1) \oplus k \oplus (k-1) = \ldots \oplus X \oplus X \oplus X
\end{aligned}
$$

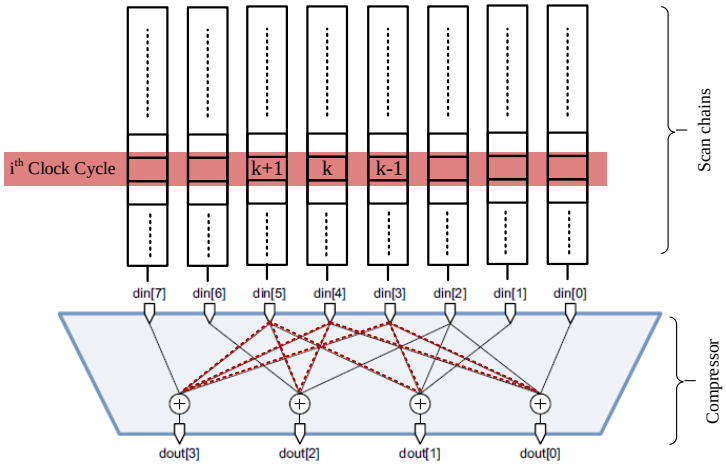

Fig. 5. Compressor logic structure example for 8-to-4 compressor [Synopsys 2015c].

The key propagation will be failed as there is no forward implication for these XOR gates. Thus, by selecting the scan chains carefully and place key gates at the same location on these scan chains, we can circumvent key sensitization attack.

One could argue that the diagnostics done for failure analysis may be impacted due to the compressed test responses. However, modern EDA tools provide diagnostic support (high defect coverage and accurate fault diagnostics) with compression in place [Synopsys 2015b; 2015d; Nagaraj 2015]. The compacted responses collected during the test can be used for diagnostics without going back to the traditional DFT (without compressions). So with this added feature, we do not see any reason why the SoC designers will not use test compression in their SoCs.

It is worthwhile to mention here that our proposed key insertion flow does not impact the test process using JTAG [IEEE Standards Association and others 2001] in the field as the test patterns are generated after the insertion of the key gates and has no impact on CUK. No modifications to the design are made after test pattern generation.

2.2. FORTIS for Preventing IC Overproduction

The success of the proposed design flow lies in the secure transfer of CUKs to the chips without interception by any untrusted entity in the supply chain. In the following, we will describe the transfer of CUK from SoC designer to the chips to prevent IC overproduction by the untrusted foundry. Then we will extend this communication protocol from 3PIP owners to prevent IP overuse by the untrusted SoC designer.

To ensure the safe transfer of CUK from the SoC designer to the chips, the following are required:

— *Message integrity*: The SoC designer must ensure the integrity of the request received from the chips. If the SoC designer detects an altered request, either modified by an attacker or errors in the transmission, it is necessary to stop the transmission of the encrypted CUKs.
— *End-point authentication*: The SoC designer must verify that the request was initiated by the chips and not by an untrusted foundry or any other entity in the supply chain. As the chip cannot communicate by its own, the foundry only gets the information from the chip and forwards it to the SoC designer.
— *Confidentiality*: Only the SoC designer and the chip should understand the contents of the transmitted messages.

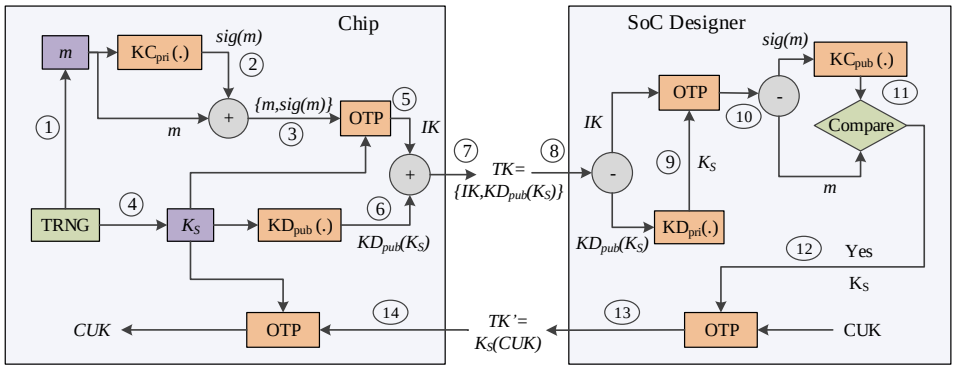

Fig. 6. Architecture and communication flow of FORTIS to prevent IC overproduction.

All these can be achieved by using a combination of asymmetric and symmetric key encryption. The widely used Rivest-Shamir-Adleman (RSA) algorithm [Rivest et al. 1978] is used as the asymmetric key encryption algorithm to provide message integrity and end-point authentication. Note that Discrete logarithm or elliptic curve algorithms [Paar and Pelzl 2009] can also be used instead of RSA. Depending on the area budget, one can select one of the algorithms from the above. One-time-pad (OTP) [Paar and Pelzl 2009] is used for symmetric key encryption to provide the confidentiality. OTP has low area overhead as it only requires a simple XOR network for the encryption and decryption.

Figure 6 shows our proposed protocol to securely transfer CUK from SoC designer to the fabricated chips. To achieve this we need the keys (public key of the SoC designer (KD_{pub}) and private key (KC_{pri}) of the design) to be embedded in the design. Thus all the fabricated chips have the same CUK, KD_{pub}, and KC_{pri}. The SoC designer has the other two keys, KD_{pri}, and KC_{pub}. The steps for transferring the CUK from the SoC designer to the chip are listed below:

1) The on-chip TRNG generates a message (m) which is unique for each and every chip.

2) The message m is encrypted with the private key KC_{pri} stored in the chip to form a signature, i.e., $sig(m) = KC_{pri}(m)$. This signature will be used to validate message integrity and verify end-point authentication.

3) The message m and its signature $sig(m)$ are concatenated.

4) The TRNG generates a random session key (K_S), which is unique for every communication. This session key can be stored in a non-volatile memory for future decryption to receive CUK. If the entire activation is performed while the chips are powered on, we can even store K_S in a volatile memory. This unique session key helps us to prevent replay attacks.

5) A one-time-pad (OTP) encrypts the concatenated message (m) and its signature ($sig(m)$) with K_S.

$$IK = K_S(\{m, sig(m)\}) = K_S \oplus \{m, sig(m)\}$$

6) The session key, K_S, is encrypted with the public key, KD_{pub}, of the SoC designer.

7) The transmission key is formed by concatenating encrypted K_S and IK. $TK = \{KD_{pub}(K_S), IK\}$. The foundry receives TK from the chip and forwards it to the SoC designer.

8) Upon receiving the TK from the foundry, the SoC designer separates encrypted K_S and IK.

9) Session key K_S is retrieved by decrypting $KD_{pub}(K_S)$ with KD_{pri}.

$$K_S = KD_{pri}(KD_{pub}(K_S))$$

10) A one-time-pad is used to decrypt IK to retrieve the concatenated m, and its signature $sig(m)$.

$$IK \oplus K_S = K_S \oplus \{m, sig(m)\} \oplus K_S = \{m, sig(m)\}$$

11) The SoC designer retrieves the message from the signature by using chip's public key, KC_{pub}.

$$KC_{pub}(sig(m)) = KC_{pub}(KC_{pri}(m)) = m$$

12) A comparison is performed to match m and decrypted signature $sig(m)$. This step verifies the integrity of m and end-point authenticity. The SoC designer now knows that the TK is originally coming from the chip if m equals to the $KC_{pub}(sig(m))$, not from an attacker.

13) After verifying the authenticity of the sender, the SoC designer encrypts CUK by using an OTP with the session key K_S and sends another transmission key (TK') to the foundry.

$$TK' = K_S(CUK) = K_S \oplus CUK$$

14) The foundry applies this TK' to the chip. The chip now reconstructs the correct CUK after decrypting TK' by using the OTP with its stored session key, K_S.

$$K_S(TK') = K_S \oplus CUK \oplus K_S = CUK$$

This correct CUK is then stored in a non-volatile memory (NVM) [Jeong et al. 2012] to provide inputs to the key gates. The size of the NVM depends on the size of the CUK. One needs to make sure that the CUK values are not accessible by the JTAG [IEEE Standards Association and others 2001] in the field.

2.3. FORTIS for Preventing IP Overuse

The overuse of IP occurs when a SoC designer makes a foundry manufacture extra chips (including IC overproduction) without the knowledge of the 3PIP owners, which results in a loss of revenue. In this section, we will present an approach to prevent 3PIP overuse.

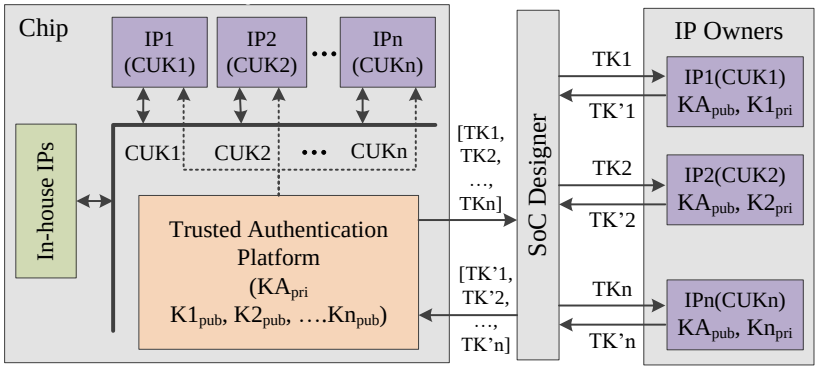

Fig. 7. Architecture and communication flow of FORTIS to prevent IP overuse.

Figure 7 shows our proposed FORTIS architecture to prevent IP overuse. The IC contains a trusted authentication platform (TAP), which is introduced in the SoC design in order to reduce the area of each 3PIP by eliminating individual encryption/decryption blocks for each IP block, and is trusted by all the 3PIPs in that SoC. In addition, TAP can be encrypted by our propose approach (see Section 2.4) such that inner details are hidden to the SoC designer and it is modification resistant. The connection details between the TAP and 3PIPs are also obfuscated by the EDA tool such that SoC designers cannot add additional circuitry to observe $CUKs$ and provide them to the 3PIPs directly. Note that, we assume trusted EDA tools throughout the paper and it cannot be modified to get an undue advantage by the SoC designers.

Each IP contains a lock (i.e., the key gates) which can only be unlocked by using the correct chip unlock key CUK_i of IP i. This CUK_i is only known by the i^{th} IP owner. The IPs only receive CUK_is from the TAP for the activation. TAP holds its own private key (KA_{pri}) and public

keys ($\{Ki_{pub}\}$) for all the IPs in the design. TAP generates the transmission keys ($TK1$, $TK2$, ..., TKn) and sends them to the SoC designer. The SoC designer forwards each transmission key (TKi) to the corresponding IP owner. In return, the IP owners send the encrypted chip unlock key ($TK'i$) to the SoC designer. Upon receiving all the $TK'i$s from the IP owners, the SoC designer sends them to the foundry to unlock each IP in the fabricated chips.

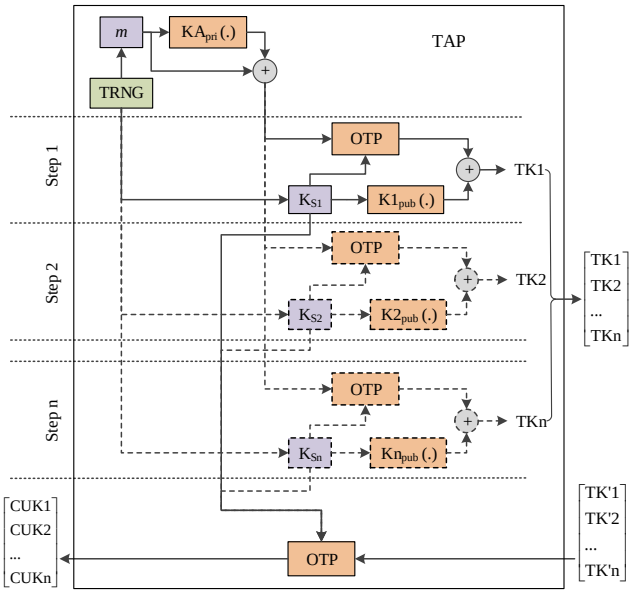

Fig. 8. Architecture of TAP and communication flow to reconstruct $CUKs$ for all 3PIPs in a SoC.

Figure 8 shows the generation of transmission keys by the trusted authentication platform. TAP has a built-in TRNG, which generates a message (m) and separate session keys (K_S) for all different IP owners. First, the signature of m is generated and then concatenated with its signature. This ensures the message integrity and end-point-authentication for all the IP owners and also that the request is indeed coming from the TAP and not from a tampered TAP used by an attacker. TAP then generates one transmission key in each step. At step 1, a session key (K_{S1}) for IP owner 1 is obtained from the TRNG. This session key helps to encrypt $\{m, sig(m)\}$ and the encrypted output is concatenated with the encrypted K_{S1} to form $TK1$. At step 2, a different session key (K_{S2}) for IP owner 2 is received from the TRNG. This session key is then used to encrypt $\{m, sig(m)\}$ and the encrypted output is concatenated with the encrypted K_{S2} to form $TK2$. In a similar fashion, all the transmission keys (TKi) are generated. Then the foundry receives all the TKi, sends them to the SoC designer, and waits for the encrypted $CUKs$.

After receiving the transmission keys ($TK'i$s), the foundry applies them to the TAP. TAP decrypts these $TK'i$s by using its session keys, K_Ss, to generate the chip unlock keys, CUK_is, for all different IPs.

2.4. FORTIS for Preventing IP Piracy

To establish a forward trust between the IP owners and SoC designers, foundries, and assemblies, it is necessary to add security measures in the IP to prevent IP piracy, such as, cloning, and modification of IPs by untrusted SoC designers and foundries. FORTIS inherently prevents the cloning of IPs. As each IP is locked by using a set of key gates, even if the attackers copy the netlist completely, they cannot unlock it without the proper CUK. However, simulation of an SoC having these locked 3PIPs needs to be addressed, as these IPs will work properly only upon receiving a proper CUK.

At the same time, it is necessary to protect these $CUKs$ from the SoC designer. Otherwise, there is no point of adding them into the IPs in the first place. Our objective of simulating a 3PIP will be successful if we provide a CUK securely to the simulation tool without interception by the SoC designer.

We also propose IP integrity verification to prevent IP modification by the SoC designer. We use a cryptographic hash function [NIST 2012] to create an IP digest (see message digest [Paar and Pelzl 2009]) to make it resistant against modification. Any modification, including addition or deletion of extra features, to a 3PIP will result in a different IP digest than the original one, which can easily be detected by comparison in an EDA tool.

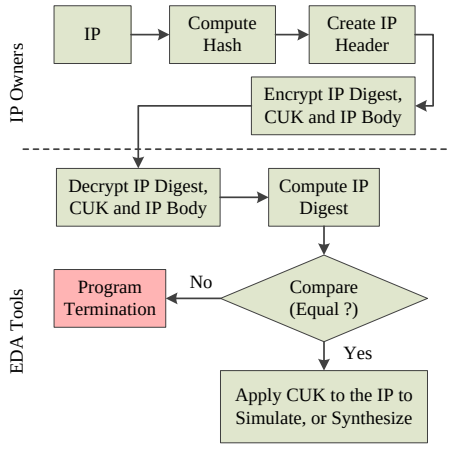

Fig. 9. Proposed flow to prevent IP piracy integrated into FORTIS.

Figure 9 shows our proposed flow to prevent cloning and modification of 3PIPs. The IP owners first compute IP digest which is the hash of the entire locked netlist. An IP header is the created which contains the CUK for the simulation of an SoC and the IP digest. The IP is then encrypted (the code inside the *'pragma protect* blocks) by using a symmetric encryption method (e.g., Advanced Encryption Standard - Cipher Block Chaining (AES-CBC) [Morris Dworkin 2001]) recommended in encryptP1735.pl script [Synopsys 2014]. This symmetric key is now encrypted by the public keys of different EDA vendors such that these vendors can later on decrypt them to get the IP.

We propose a new IP digest comparison flow during synthesis and simulation of SoCs. The EDA tool first needs to decrypt the encrypted portion of the IP header and the IP body. An IP digest has to be calculated from the decrypted IP by using the same hash function used before to form the IP digest. A comparison needs to be performed with the IP digest retrieved from the IP header and newly computed IP digest. If both of them are equal, then it is ensured that no modifications to the program has been made, otherwise, the program has to be terminated.

Figure 10 shows an example of our proposed encrypted IP. We use SHA-512 [NIST 2012] to form an IP digest, which is attached to the IP header along with the CUK. We use Synopsys encryptP1735.pl script [Synopsys 2014] to encrypt the IP header and IP body. Figure 10(a) shows a locked IP. The encryption is carried out in two parts - (i) The IP vendor encrypts the IP data (data block) using its own symmetric key which is called the data key. We use aes256-cbc as symmetric encryption algorithm to encrypt the data block. (ii) The IP vendor then encrypts the data key with its public key by using asymmetric encryption to create a key block. The encryption version, encode

a) Locked IP. b) Locked and encrypted IP.

Fig. 10. IP header insertion for the simulation of a locked IP.

type, key owner, key name and key method need to be mentioned. We use RSA as asymmetric encryption to generate the key block which is attached to the IP header (see Figure 10(b)).

3. RESULTS AND ANALYSIS

3.1. Test Metrics Analysis

The objective of analyzing the test metrics is to provide an evidence of the impact of our proposed modification of the locks to enable manufacturing test before the activation of chips (see Section 2.1). The proposed architecture is evaluated with implementations on large benchmark circuits from ITC99 [Davidson 2015], opencores.org [OpenCores 2015], and OpenSPARC T1 processor core [OpenSPARC T1 2015]. In this evaluation, we have used SAED 32/28nm library [Synopsys 2015a] provided by Synopsys to synthesize all benchmark circuits. Each synthesized benchmark circuit is then locked with a 128 bit CUK. A total of 128 XOR/XNOR gates along with 128 D-flip flops are inserted for each synthesized benchmark circuit. Scan-chain insertion is performed on both unsecured and secured versions of the same circuits to evaluate and compare test metrics, such as test pattern count and test coverage. A comparison between the unlocked and locked versions of these benchmarks are presented in Table II.

Table II. Test Metrics Comparison.

Benchmark	Gate Count	Pattern Count			Test Coverage		
		Before	After	Change (%)	Before	After	Change (%)
des	16,341	60	59	-1.67	100.00	99.78	-0.22
b18	25,397	404	388	-3.96	99.53	99.45	-0.08
ethernet	30,534	627	629	0.32	99.94	99.84	-0.10
b19	40,789	424	418	-1.42	99.34	99.40	0.06
vga_lcd	43,346	1,710	1,721	0.64	100.00	99.94	-0.06
leon3mp	253,050	3,821	3,848	0.71	100.00	99.99	-0.01
sparc	836,865	3,220	3,185	-1.09	99.16	99.09	-0.07

As shown in Table II, the gate count of investigated benchmark circuits range between $16,341$ to $836,865$. We chose these larger benchmark circuits to better represent typical large industrial designs. We also present the pattern count and test coverage figures of the implemented designs before and after applying our proposed architecture, in order to present the impact of proposed architecture on circuit testability. We do not expect any major change in test pattern count and test coverage, but there may be a minor improvement to test coverage as each key gate with a D flip-flop adds a test point in the design. On the other hand, the XOR/XNOR gates and D flip-flops create additional faults in the netlist, which may lead to the reduction of test coverage. The shown pattern count change range between -3.96% and 0.71%, which means at worst the proposed architecture would result in less than 1% overhead in pattern length. Similarly, the change in test coverage ranges between -0.22% and 0.06%. Both impacts are minor and would not significantly impact testability of the secured design.

3.2. Area Overhead Analysis

The area overhead of our proposed FORTIS consists of:

(i) RSA module: The RSA module used in our proposed design to encrypt the session key and generate the signature makes up a major part of the area overhead. This area can be reduced significantly depending on the speed of operation. As speed is not our major concern, we can select a slower, but more area efficient RSA module. It is reported that a minimum size RSA datapath can be implemented by using only 861 gates [Miyamoto et al. 2011].

(ii) OTP module: The size of the one-time pad depends on the size of the CUK. For a 128 bit CUK, we need 128 XOR gates. The same OTP can be used for the encryption of $\{m, sig(m)\}$ and decryption of TK'.

(iii) Keys gates: The size due to keys also depends on the CUK. To implement one key bit, we need one XOR/XNOR gate and a scan flip-flop.

(iv) RSA Keys: Extra storage or logic is needed to keep or generate at least 1024 bit KC_{pri} for chips or KA_{pri} for TAP. We can simply neglect the size of the public keys (KD_{pub} or $Ki_{pub}s$) as they can be as small as number 3 or 17 [Paar and Pelzl 2009].

(v) TRNG: A single TRNG is used for generating the message, m and session keys, K_Ss. We propose the use of an area efficient cryptographically secure pseudorandom number generator [Holcomb et al. 2007] or [Sunar et al. 2007] depending on the implementation choice.

(vi) Non-volatile memory: The size of the non-volatile memory depends of the session keys, K_Ss. We need a 128 bit non-volatile memory to store a 128 bit K_S.

There is no area overhead of any 3PIPs for preventing IP overuse except for the key gates. The trusted authentication platform (TAP) provides the $CUKs$ to all different 3PIPs. The primary motivation for implementing TAP in any design for a SoC designer is that they need to prevent IC overproduction.

Considering all these modules, the total gate count is approximately 10K. Table III shows the overhead analysis. For benchmark circuits, it ranges from 24.52% to 1.19%. However, for industrial designs it becomes less than 1%. For Xilinx Artix-7 and Kintex-7 [Xilinx 2105a] the overhead becomes 0.77% and 0.15% respectively. It becomes negligible for Virtex-7 [Xilinx 2105b]. The area overhead may further be reduced if the original design already contains a TRNG and a RSA module, as is the case for most of the industrial designs.

Table III. Area overhead analysis.

	Design	Gate Count	Overhead (%)
Benchmarks	b19	40,789	24.52
	vga_lcd	43,346	23.07
	leon3mp	253,050	3.95
	sparc	836,865	1.19
Industrial Designs	Artix-7	1.3M	0.77
	Kintex-7	6.6M	0.15
	Virtex 7	20M	0.05

3.3. Security Analysis

The security of our proposed protocol is of prime importance to prevent the overproduction of ICs and overuse of 3PIPs. In the following, we will perform the security analysis of our proposed approach.

Exhaustive key search: The length of a chip unlock key, CUK, should be long enough such that it can withstand exhaustive key search or brute-force attacks. We need to achieve at least 80 bits of security as this is the lower minimum requirement for exhaustive key search [Paar and Pelzl 2009]. To achieve this, we require 80 key gates (XOR/XNOR). However, the key size may be increased up to 256 bits for higher security, which will hardly impact the overall area of a modern design.

Encryption: In our approach, we use RSA to encrypt the session key and generate signature. One can achieve 80 bit of security while the key length is 1024 bits. However, 128 bit security can be achieved with the key length of 3072 bits [Paar and Pelzl 2009]. Depending on the area budget one can select a desired security level of n bits. We have used one-time-pad to encrypt $\{m, sig(m)\}$. As the session keys, K_Ss, are generated from a TRNG, a perfect secrecy can be achieved. Thus, we can achieve an overall RSA equivalent secrecy in our proposed protocol.

Man-in-the-middle attack: As the key-pairs for the RSA are generated by the IP owners and reside in the circuit, no key transfer is required. This prevents an attacker (e.g., untrusted foundry) from becoming a man-in-the-middle.

Replay attack: In this attack scenario, the attacker copies a message between two parties and replays that message to one or more of them. Our proposed protocol is inherently resistant to replay attacks as a new session key, K_S, is generated every time during encryption. Every time, the encrypted message will be different from the previous one. In addition, the message (m) is unique for every chip, which also helps to make a unique transmission key for every chip.

Reverse engineering: As we use a secure logic encryption technique [Rajendran et al. 2012], it is extremely hard for an attacker to find CUK by reverse engineering. Even if we assume that reverse engineering is possible to find the key, an attacker cannot feed the CUK to a chip, as they do not know the private key of the SoC designer (KD_{pri}) to retrieve K_S. As the session key, K_S, is unique for every chip, it is not economical for the attackers to retrieve K_S for each chip by reverse engineering. We also assumed that the attacker cannot model the TRNG to predict its output after observing certain K_Ss. Finally, we believe that it is extremely expensive to perform reverse engineering for modern designs manufactured with $22nm$ or lower technology nodes.

Tampering RSA Keys: In this attack scenario, an untrusted foundry reconstructs new masks to replace the keys, KC_{pri} and KD_{pub}, with its own. This enables the foundry to unlock unlimited number of chips when it receives the CUKs from the IP owners. Fortunately, this attack can easily be prevented by the IP owners. The SoC designer can request only one locked chip and then verify the correct keys. If the foundry replaces KC_{pri} and KD_{pub} by its own, the SoC designer will not be able to unlock the chip and consequently, it can detect mask modification.

Tampering TRNG: An untrusted foundry can modify the masks to bypass the TRNG and write a permanent value for K_Ss and m. Once it knows the CUK, it can unlock any number of chips. Fortunately, this attack can also be detected by the IP owners and can be prevented. Like before, the SoC designer can request few locked chips to monitor the message, m and the session key, K_S. If either m or K_Ss from these chips are the same or biased, it will definitely be the indication of the tampering of TRNG. As it is extremely expensive to design a new set of masks, there is little economic incentive for an untrusted foundry to manufacture a product with two different set of masks.

Tampering IP Digest: In this attack scenario, the attacker tries to tamper the IP digest by replacing the original IP digest with the tampered IP digest. Fortunately, this tampering can be detected. As the attacker does not have the private key of the EDA tool (we assume a trusted EDA tool for synthesis

and simulation), he cannot reconstruct the original IP from its encrypted version. If the attacker try to modify the IP and then compute the digest, it will be different than the original one.

4. CONCLUSION

In this paper, we have presented, FORTIS, a comprehensive solution for establishing forward trust for different entities involved in the SoC design and manufacturing. FORTIS uses a novel communication protocol between the fabricated chips and the SoC designers/IP owners to activate the chips for preventing IP overuse and IC overproduction. FORTIS uses an existing logic encryption technique to obfuscate the netlist of an SoC or a 3PIP and can only be unlocked upon receiving a chip unlock key (CUK). A modification is proposed to the existing encrypted netlist to enable manufacturing tests before the activation of chips which is one of the key requirement to prevent overproduction. Our proposed modification does not have any impact on manufacturing test process.

To address IP overuse, we have introduced a trusted authentication platform in the SoC. This TAP is trusted by the all parties involved in the SoC design. To the best of our knowledge, the metering approach we have presented to prevent IP overuse is the first in the literature. The encrypted IP with additional IP digest check prevents the SoC designer from IP piracy. As the IPs are locked by using a set of XOR/XNOR gates, even if the attackers copy the netlist completely, they cannot unlock it without the proper CUK, which prevents IP cloning. Finally, our proposed design flow is resistant to all known attacks.

References

Yousra Alkabani, Farinaz Koushanfar, and Miodrag Potkonjak. 2007. Remote activation of ICs for piracy prevention and digital right management. In *Proc. of IEEE/ACM international conference on Computer-aided design*. 674–677.

Yousra M. Alkabani and Farinaz Koushanfar. 2007. Active hardware metering for intellectual property protection and security. In *Proc. of 16th USENIX Security Symposium on USENIX Security Symposium*. Article 20, 16 pages.

A. Baumgarten, A. Tyagi, and J. Zambreno. 2010. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design and Test of Computers* 27, 1 (Jan.-Feb. 2010), 66 –75. DOI: http://dx.doi.org/10.1109/MDT.2010.24

M. Bushnell and Vishwani Agrawal. 2000. *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Springer.

Encarnación Castillo, Uwe Meyer-Baese, Antonio García, Luis Parrilla, and Antonio Lloris. 2007. IPP@HDL: Efficient Intellectual Property Protection Scheme for IP Cores. *IEEE Trans. Very Large Scale Integr. Syst.* 15, 5 (May 2007), 578–591. DOI: http://dx.doi.org/10.1109/TVLSI.2007.896914

R. S. Chakraborty and S. Bhunia. 2008. Hardware protection and authentication through netlist level obfuscation. In *Proc. of IEEE/ACM International Conference on Computer-Aided Design*. 674 –677. DOI: http://dx.doi.org/10.1109/ICCAD.2008.4681649

R. S. Chakraborty and S. Bhunia. 2009. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 10 (October 2009), 1493 –1502. DOI: http://dx.doi.org/10.1109/TCAD.2009.2028166

E. Charbon. 1998. Hierarchical watermarking in IC design. In *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998*. 295–298. DOI: http://dx.doi.org/10.1109/CICC.1998.694985

G. Contreras, T. Rahman, and M. Tehranipoor. 2013. Secure Split-Test for Preventing IC Piracy by Untrusted Foundry and Assembly. In *Proc. International Symposium on Fault and Defect Tolerance in VLSI Systems*.

DASC. 2014. 1735-2014 - IEEE Approved Draft Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP). (2014).

Scott Davidson. 2015. ITC99 Benchmark Home Page. (2015). https://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. 2007. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *In Proceedings of the Conference on RFID Security*.

Jiawei Huang and J. Lach. 2008. IC activation and user authentication for security-sensitive systems. In *Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust*. 76 –80. DOI: http://dx.doi.org/10.1109/HST.2008.4559056

IEEE Standards Association and others. 2001. *1149.1-2001 - IEEE Standard Test Access Port and Boundary Scan Architecture*. IEEE.

Doo Seok Jeong, Reji Thomas, RS Katiyar, JF Scott, H Kohlstedt, A Petraru, and Cheol Seong Hwang. 2012. Emerging memories: resistive switching mechanisms and current status. *Reports on Progress in Physics* 75, 7 (2012), 076502.

A.B. Kahng, J. Lach, W.H. Mangione-Smith, S. Mantik, I.L. Markov, M. Potkonjak, P. Tucker, Huijuan Wang, and G. Wolfe. 2001. Constraint-based watermarking techniques for design IP protection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 20, 10 (Oct 2001), 1236–1252. DOI: http://dx.doi.org/10.1109/43.952740

A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. 2006. Constraint-based Watermarking Techniques for Design IP Protection. *Trans. Comp.-Aided Des. Integ. Cir. Sys.* 20, 10 (Nov. 2006), 1236–1252. DOI: http://dx.doi.org/10.1109/43.952740

D. Kirovski, Yean-Yow Hwang, M. Potkonjak, and J. Cong. 2006. Protecting Combinational Logic Synthesis Solutions. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 25, 12 (Dec 2006), 2687–2696. DOI: http://dx.doi.org/10.1109/TCAD.2006.882490

F. Koushanfar. 2012. Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management. *Information Forensics and Security, IEEE Transactions on* 7, 1 (Feb 2012), 51–63. DOI: http://dx.doi.org/10.1109/TIFS.2011.2163307

F Koushanfar and Gang Qu. 2001. Hardware metering. In *Proc. IEEE-ACM Design Automation Conference*. 490–493. DOI: http://dx.doi.org/10.1109/DAC.2001.156189

Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. 2001. Intellectual property metering. In *Inform. Hiding*. Springer-Verlag, 81–95.

S.S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. 2008. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust*. 67 –70. DOI: http://dx.doi.org/10.1109/HST.2008.4559053

J Kurose and K Ross. 2001. Computer Networks: A Top-Down Approach. (2001).

J. Lach, W.H. Mangione-Smith, and M. Potkonjak. 2001. Fingerprinting techniques for field-programmable gate array intellectual property protection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 20, 10 (Oct 2001), 1253–1261. DOI: http://dx.doi.org/10.1109/43.952741

J.W. Lee, Daihyun Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. 2004. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proc. of Digest of Technical Papers on VLSI Circuits*. 176 – 179. DOI: http://dx.doi.org/10.1109/VLSIC.2004.1346548

K. Lofstrom, W.R. Daasch, and D. Taylor. 2000. IC identification circuit using device mismatch. In *Proc. of IEEE International Solid-State Circuits Conference*. 372 –373. DOI: http://dx.doi.org/10.1109/ISSCC.2000.839821

Microsemi. 2014. Libero SoC Secure IP Flow User Guide for IP Vendors and Libero SoC Users. (2014). http://www.microsemi.com/document-portal/doc_view/133573-libero-soc-secure-ip-flow-user-guide

A. Miyamoto, N. Homma, T. Aoki, and A. Satoh. 2011. Systematic Design of RSA Processors Based on High-Radix Montgomery Multipliers. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 19, 7 (July 2011), 1136–1146. DOI: http://dx.doi.org/10.1109/TVLSI.2010.2049037

Morris Dworkin. 2001. NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation . (2001).

Pradeep Nagaraj. 2015. *Choosing the Right Scan Compression Architecture for Your Design*. Technical Report. https://www.cadence.com/rl/Resources/white_papers/Test_Compression_wp.pdf

NIST. 2012. FIPS PUB 180-4: Secure Hash Standard. (March 2012).

OpenCores. 2015. (2015). https://www.opencores.org.

OpenSPARC T1. 2015. (2015). http://www.oracle.com/technetwork/systems/opensparc/ opensparc-t1-page-1444609.html.

Christof Paar and Jan Pelzl. 2009. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media.

Gang Qu and Miodrag Potkonjak. 2003. *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media.

Md Tauhidur Rahman, Domenic Forte, Quihang Shi, Gustavo K Contreras, and Mohammad Tehranipoor. 2014. CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*. IEEE, 46–51.

J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. 2012. Security analysis of logic obfuscation. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. 83–89.

R. L. Rivest, A. Shamir, and L. Adleman. 1978. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM* 21, 2 (Feb. 1978), 120–126.

J.A. Roy, F. Koushanfar, and I.L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *Proc. on Design, Automation and Test in Europe*. 1069 –1074. DOI: http://dx.doi.org/10.1109/DATE.2008.4484823

Y. Su, J. Holleman, and B. Otis. 2007. A 1.6pJ/bit 96using Process Variations. In *Proc. of IEEE International on Solid-State Circuits Conference*. 406 –611. DOI: http://dx.doi.org/10.1109/ISSCC.2007.373466

P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. 137–143. DOI: http://dx.doi.org/10.1109/HST.2015.7140252

G.E. Suh and S. Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proc. of ACM/IEEE on Design Automation Conference*. 9 –14.

B. Sunar, W.J. Martin, and D.R. Stinson. 2007. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *Computers, IEEE Transactions on* 56, 1 (Jan 2007), 109–119. DOI:http://dx.doi.org/10.1109/TC.2007.250627

Synopsys. 2014. Synopsys FPGA Synthesis Synplify Pro for Lattice: User Guide. (November 2014).

Synopsys. 2015a. 32/28nm Generic Library for Teaching IC Design. (2015). https://www.synopsys.com/COMMUNITY/UNIVERSITYPROGRAM/Pages/32-28nm-generic-library.aspx.

Synopsys. 2015b. Compression for Highest Test Quality and Lowest Test Cost. (2015). https://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/dftmax-ultra-ds.aspx

Synopsys. 2015c. DFT Compiler, DFTMAX^{TM}, and DFTMAX^{TM} Ultra User Guide. (Septermber 2015).

Synopsys. 2015d. High Quality, Low Cost Test. (2015). https://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/DFTMAX.aspx

Mohammad Tehranipoor and Cliff Wang. 2012. *Introduction to Hardware Security and Trust*. Springer.

Mark (Mohammad) Tehranipoor, Ujjwal Guin, and Domenic Forte. 2015. *Counterfeit Integrated Circuits: Detection and Avoidance*. Springer.

Xilinx. 2105a. (2105). http://www.xilinx.com/publications/prod_mktg/zynq7000/Zynq-7000-combined-product-table.pdf.

Xilinx. 2105b. (2105). http://www.xilinx.com/publications/prod_mktg/Virtex7-Product-Table.pdf.

Age Yeh. 2012. Trends in the global IC design service market. DIGITIMES Research. (March 2012).

Xiaotong Zhuang, Tao Zhang, Hsien-Hsin S. Lee, and Santosh Pande. 2004. Hardware Assisted Control Flow Obfuscation for Embedded Processors. In *Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '04)*. ACM, New York, NY, USA, 292–302. DOI:http://dx.doi.org/10.1145/1023833.1023873