Protecting Bus-based Hardware IP by Secret Sharing

Jarrod A. Roy†, Farinaz Koushanfar‡ and Igor L. Markov†
†The University of Michigan, Department of EECS, 2260 Hayward Ave., Ann Arbor, MI 48109
‡Rice University, ECE and CS Departments, 6100 South Main, Houston, TX 77005

ABSTRACT

Our work addresses protection of hardware IP at the mask level with the goal of preventing unauthorized manufacturing. The proposed protocol based on chip locking and activation is applicable to a broad category of electronic systems with a primary bus. Such designs include (1) numerous IP offerings for USB, PCI, PCI-E, AMBA and other bus standards typically used in system-on-a-chip designs and computer peripherals, (2) SRAM-based FPGAs that are programmed through an input bus, (3) general-purpose and embedded microprocessors, including soft cores, (4) DSPs, (5) network processors, and (6) game consoles. Our key insight is that such designs can be locked by scrambling the central bus by controlled reversible bit-permutations and substitutions. To securely establish a unique code per chip to control bus scrambling, we employ true random number generators and Diffie-Hellman cryptography during activation.

Categories and Subject Descriptors

K.5.1 [**Hardware/Software Protection**]: Proprietary rights

General Terms Design, Security

Keywords Integrated circuits, Manufacturing, Computer crime, Cryptography

1. INTRODUCTION

As designs become more complex and on-chip transistor counts reach into the billions, the designer's skills, methodologies, and tools are the invaluable assets of semiconductor design houses. This is particularly true for the small- and medium-sized fabless design companies. The soaring costs of building and maintaining state-of-the-art semiconductor manufacturing facilities and nano-scale masks is driving even the large design houses to abandon their home manufacturing and become fabless. For example, Texas Instruments (TI) – the third largest semiconductor company in the world – has announced a new foundry strategy in May 2007, indicating that TSMC, UMC and a third vendor will split the company's 45-nm fabrication. The current trust models and royalty agreements do not fully protect the rights of the designers. The hardware IP providers pay the expenses of masks for their designs, trusting that the foundry would not make additional copies outside the contract. The ready availability of masks, low cost of silicon, and lack of IP owner's control over the manufacturing flow facilitates illegal copying of ICs. Furthermore, IC packaging obscures chip internals and makes it difficult to trace the IP owner.

DAC 2008, June 8–13, 2008, Anaheim, California, USA.

Large-scale integration of millions of nano-scale devices is used in many ICs today, including microprocessors, DSPs , FPGAs, and dedicated graphic chips. A major research challenge is to develop IP protection techniques that are powerful and general enough to handle all these categories. At first, addressing the issue appears impossible because these products have fundamentally different structures. For example, memory-based products, such as flash and FPGA, are so regular and flexible that locking only a small part of a chip will not prevent an IP burglar from using the rest. But locking all parts of the chip leads to unacceptable overhead.

A key insight in our work is that many of the modern IP designs communicate through busses and/or rely on an internal bus to support their primary functions [3]. For example, FPGAs are programmed through an input bus, CPUs rely on several internal busses, GPUs stream pixels and texels through a graphics bus, network processors communicate through an Ethernet bus or antennae, and an entire class of bus-based IP entirely depend on industry-standard busses, such as USB, PCI, PCI-Express, AMBA, etc.[1] While the specific uses of busses and applications may have nothing in common, the functionality of all devices critically depends on a small number of busses. Our work uses this property to lock and unlock the ICs by manipulating their busses.

We propose a novel bus-based IC protection method that can authorize the activation of each individual chip and hence can control the number of working chips that contain the designer's IP. The method leverages bit permutations and substitutions that scramble the bus using a key unique to each IC. The scrambling renders the IC unusable for anyone who does not have the specific *key* for the chip. Only the owner of IP rights with access to design details can compute the shared key required to unlock (unscramble) the bus and to make the IC usable. The key is calculated simultaneously by the designer and the chip using an asymmetric Diffie-Hellman key sharing protocol that does not reveal the key in communications to and from the IC. Thus, an eavesdropper who records all communications between the chip and the unlocking authority cannot compute the key.

[1]A notable exception to this list is the class of ASIC designs. Our techniques can still be adapted to many ASICs, but such adaptations may have to be application-specific.

The highlights of this work include the following.

- First-of-a-kind research on IP protection for a broad class of integrated circuits, by the novel bus-based protocol for chip locking and activation.
- The idea to leverage the Diffie-Hellman key exchange protocol to lock and activate computing hardware.
- Review and comparative analysis of existing techniques to encode bit permutations and word substitutions.
- Analysis of possible attacks aimed at disrupting the proposed protocols, providing countermeasures against the attacks.
- A new application of chip locking — feature selection — that is possible with our techniques.
- Empirical evaluations of the degree of resiliency against the attacks and probability of success.

The remainder of the paper is organized as follows. In Section 2 we review background in public key cryptography and survey the literature on hardware IP protection. Section 3 outlines the scope of applicability of our IP protection techniques and describes a new application — feature selection. Section 4 introduces the global flow of the proposed approach. In Section 5 we describe the bus-based locking and activation mechanisms in detail. Attacks and countermeasures are discussed in Section 6 followed by experimental evaluations in Section 7. We conclude in Section 8.

2. PRELIMINARIES

Below we review a particular cryptographic public key exchange scheme that is used in our work. Then we survey related research in hardware IP protection, to which our work can be contrasted.

2.1 Diffie-Hellman public key exchange

Public key cryptography protocols provide a way for sharing secrets (keys) between two users such that the shared key is never revealed during inter-user communications. Moreover, an eavesdropper who intercepts messages between the users will be unable to re-construct the secret. The technique is referred to as *asymmetric cryptography* since the information exchanged between the users to construct the shared key is different. Asymmetric cryptography was introduced by Diffie and Hellman (D-H) in 1976. The idea is to use a mathematical "one-way" function that can be quickly computed in polynomial time, but is hard to invert. In this role, the D-H key sharing protocol uses *modular exponentiation*, as we explain below. It considers a multiplicative group of integers C_p modulo p, where p is prime and g is a primitive root mod p. Using D-H, two users A and B can share a key. First, A and B select an element $g \in C_p$; g is not secure since it is transparent to the attackers. Next, user A randomly selects a natural number a and sends g^a to user B.[2] Likewise, B randomly chooses another number b and sends g^b to A. Both users can individually compute the same key, that is calculated as $(g^a)^b$ and $(g^b)^a$ by user A and user B respectively.

[2]Here all operations are performed within C_p, e.g., g^a corresponds to (g^a mod p) in terms of integers.

2.2 Related work

For many years, the only way designers could assert rights to their IPs (outside the royalty agreement) was to embed watermarks in the ASIC or FPGA design [5, 7]. In FPGAs, encrypting the bitstream and exploiting the flexibility of the programmable platform has enabled a number of additional access control primitives [15].

Development of techniques for locking and activation of ICs after manufacturing is a very recent topic. The idea of unique activation of chips by exploiting design features known only to the designer and not extractable at the foundry was first proposed in [1]. In their scheme, the original finite state machine (FSM) of the design is augmented with many new states. The system is engineered so that the power-up state of each chip is non-functional (locked) and determined by the unique random variations of that IC. The designer, having access to the transition function of the FSM is the only entity that can unlock the chip and bring it to the hidden original state. A newer method based on replicating a few FSM states, controlling the sequence logic, and using physically unclonable functions (PUFs) [2] was proposed: the functionality is halted unless the unlocking sequence is entered. Also, a new method based on combinational locking and public key cryptography [11] was introduced. In this work we propose a high-level locking mechanism that radically departs from combinational/sequential logic locking.

3. APPLICATION DOMAINS

Our proposed techniques apply to a wide range of IC designs and electronic systems that satisfy certain assumptions. Coarsely speaking, we require the design to have a primary bus (or a small number of them) such that disabling this bus disables the entire design. This is illustrated in Figure 1 using an SoC architecture as an example. The chip can be locked by disabling the system bus, or it can be cut off from the outside world by disabling its I/O bus. It is also possible to lock specific features of the chip by disabling feature-specific busses, such as the dedicated MPEG bus in Figure 1. The Figure also illustrates that the external bus (in this case a USB bus) can be disabled by placing an external lock. Moreover, our techniques are not restricted to single-chip IP protection, and can be generalized to larger electronic systems. In many such applications, external busses are driven by software, e.g., to program FPGAs, to support bus-based peripherals (USB, PCI-X, Firewire, etc.) through device drivers. In this con-

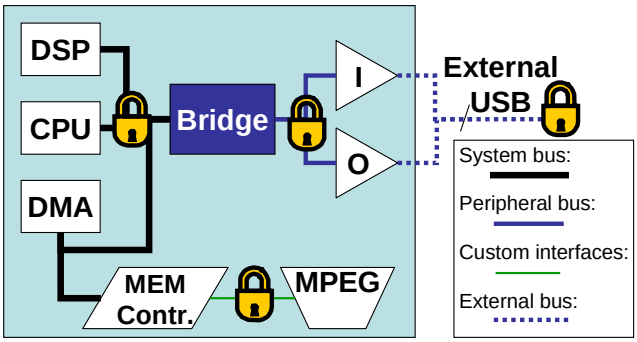

Figure 1: Bus-based IP protection.

text, software plays an important role in hardware IP protection if the activation protocols require that *cryptographic keys* possessed by the software and hardware *match*. Vice versa, by securing hardware, one can improve the environment for securing software.

Placing locks on busses has several distinct advantages. First, many current and pending designs contain busses, including bus-based IP, FPGAs, microprocessors, and DSPs. Second, busses are critical parts of the design, transferring data, chip access information, priorities, clock, and power. Third, in FPGAs and other reconfigurable devices, IP core programming is done by placing an encrypted bitstream on the busses. Fourth, connections to the memory are often accomplished by address busses, creating an opportunity to lock important features of the chips. For example, a microprocessor with a disabled memory interface can still be tested by running carefully selected sequences of commands, but is not commercially viable. Lastly, word-level processing in microprocessor and DSP data-paths also requires an indispensable data bus, which facilitates high-level synthesis, structural layout and leading-edge performance.

More formally, our assumptions can be cast within the communication-based design paradigm [10, 14] or the platform-based design paradigm [6]. In large-scale embedded systems, communication between independent components is of prime concern. Let us denote the output and input components as sender and receiver. Then the sender is modeled as process $S(F_s : I_s \Rightarrow O_s)$ and the receiver by the process $R(F_r : I_r \Rightarrow O_r)$ [14]. The connection implies that the input space of R is restricted to the intersection $O_s \cap I_r$. If the sets S and R have mismatches, there will be three possible scenarios: (1) R discards the inputs and treat them as errors, i.e., a mechanism for error handling must be added; (2) the outputs of S causing mismatches will be removed from S; and (3) signals from S are mapped to signals acceptable by R. In the latter case, an interface is utilized to transform the S output to the domain of R. Such an interface is typically split into two processes that encapsulate S and R and permit communication between the modified behavior over a connection. Connections are implemented using physical channels, i.e., busses. In emerging large-scale designs it is not surprising that the bus-based methodology is gaining so much importance, e.g., in SoC designs [3].

4. THE OVERALL FLOW

Figure 2 shows the overall flow of the new IC locking and activation approach using a timing diagram. There are two parties involved, the design house (user A) and the foundry (user B). The steps of the flow are shown using numbered boxes and arrows that show the communicated messages between the processes. User A employs the technology files available to her and devises the details of the IC, including the logic blocks, address lines, memory, component layouts, pins, and the built-in test structures (Step 1). Next, the GDSII electronic files containing the details of the IC design are sent to the user B. User B builds a mask that fabricates the received design files in silicon (Step 2). Multiple ICs will be implemented using the same mask (Step 3). Each fabricated IC would be powered up for testing (Step 4). Each chip would generate a unique ID, that is random and different from the other chips. The unique ID for the pertinent chip under test would be used as the D-H component b. Herein, we use the notation defined for the D-H protocol in

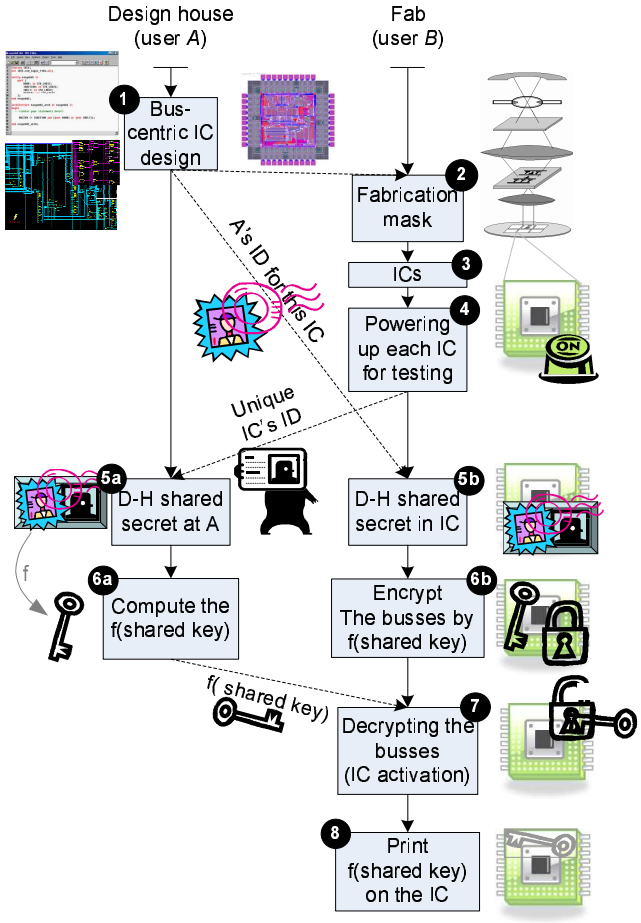

Figure 2: The new IC activation protocol.

Section 2. The value $g^b \bmod p$ is then communicated to user A, who has also generated a unique ID a corresponding to the chip under test. In turn, A sends $g^a \bmod p$ to B.

Both users can now compute the D-H shared key without needing to communicate it or any information that would expose it (Steps 5a and 5b). User A holds a secret one-way function f, which is implemented in hardware and is included on the chip. User A would compute the shared key in software (Step 6a). The shared key is the input to f; the output of f is used for scrambling the bus lines of the chips (Step 6b). Since the function f*(shared key)* is one-way, an attacker who does not know f or the *shared key* cannot generate the unlocking sequence. If the activation authority agrees to activate the chip, it then transmits the activation key, or f*(shared key)*, to user B who activates the chip (Step 7); B also encloses the activation code on the chip for sales (Step 8). To prevent multiple attempts at chip activation, we propose to "burn" random bits into irreversible fuses. Depending on the desire of the IC designer, after activation, the activation code can be used to unlock the IC for each subsequent use of the chip or burned into fuses so that the IC can be used without needing the code. The activation code must be entered into vendor-supplied software for operating the IC, as is the case with FPGAs and device drivers for PCs. Alternatively, the code can be entered into the IC through a dedicated activation pin, using a simple signaling protocol; even an existing pin can be multiplexed for activation, given that normal pins cannot be used before activation.

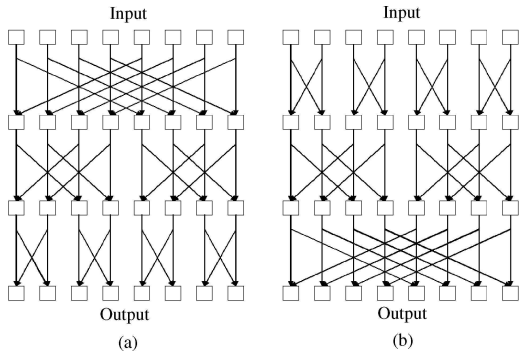

Figure 3: An 8-bit Benes network is made of two components: (a) the butterfly network and (b) the inverse butterfly network. In this network, bus wires are the labeled inputs and outputs and four different bus-key inputs per level (not shown) control the 2-to-1 MUX gates, for a total of 24 key-bits. The computation is pipelined so that one permutation completes per cycle by applying (a) on the first cycle and (b) on the second. Image from [8, Figure 3].

5. CHIP LOCKING AND ACTIVATION FOR BUS-BASED DESIGNS

In order to implement the overall flow introduced above, one uses a specific circuit-level bus-locking technique. In other words, the bus is equipped with additional *bus-key* inputs such that only a certain key combination activates the bus, while other combinations scramble it. For example, this can be accomplished by XORing key bits with bus lines, but such a naive technique is too easy to circumvent in practice.

5.1 Classes of reversible transformations

We are looking for a reversible transformation on address/data lines that can be efficiently controlled by a key of sufficient length. The transformation must have small hardware overhead and be easy to reverse in device drivers, software used to program FPGAs, etc. To this end, we compare the following categories of reversible transformations.

XOR with a key uses an n-bit key for an n-bit bus.

Arithmetic transformations can be implemented with modular addition and subtraction. Key length is n for one arithmetic operation.

Bit permutations can be implemented using Benes networks where key length is $n \log_2 n$ bits. An example of an 8-bit Benes network is shown in Figure 3. The 8-bit network consists of six levels of eight 2-to-1 MUX gates in a butterfly and inverse butterfly pattern. Each level is controlled by four key bits for a total 24 bits per key. Efficient implementations produce arbitrary permutations in one cycle of a high-performance microprocessor when pipelined [8].

Linear transformations over the field $\mathbf{F}_2$ can be implemented using only NOT and XOR gates. The algorithm from [9] generates near-optimal circuits for them where fanouts are limited to two, which can simplify layout. Key length is $n^2/\log n$ due to [9, Lemma 1 and Theorem 1].

Arbitrary reversible transformations include all of the transformations described above and rather than permute n wires, they permute all possible combinations of functions of n wires. There are $2^n!$ such circuits, making key length $\log_2(2^n!) \approx 2^n(n \ln n - 1) + 1$ (using an integration by parts approximation).

5.2 Selecting an adequate reversible transformation

First, we describe drawbacks of some of the transformations described above. Both XOR locking and arithmetic locking use n-bit keys, which may be too few for small busses. XOR locking leaves many bits unmodified on average, which may make it easy to crack. Arithmetic locking may succumb to some form of differential analysis. Also, adding a small number to the current address combination will not change the most significant bits. Key length for linear transformations and arbitrary reversible transformations is too large, and will incur significant overhead. In addition, optimal circuits to implement any reversible transformation using NOT, XOR and AND gates have been studied in [13], and are generally difficult to find.

Given the choices of a reversible transformation above, bit permutations appear to be the most promising. They have efficient circuit implementations and key lengths are larger than the size of the bus, but not as large as for linear transformations or arbitrary reversible transformations. In Section 7.2, we show how these transformations are sufficient to securely lock modern busses.

A possible concern about using bit permutations for locking is that permutations do not change the number of zeroes in the data along the bus. We are currently unaware of any exploits that could take advantage of this, but if one were found, we propose to use bit permutations and arithmetic locks together — common practice in cryptography.

6. ATTACKS AND COUNTERMEASURES

In this section, we discuss possible attacks and devise countermeasures against them. We consider the following attacks against the proposed bus-based hardware protection.

(i) Brute-force attack. The adversary aims to activate the pertinent IC by applying multiple keys. The attackers' hope is that he would randomly find the unlocking key. In more sophisticated versions of this attack, the attacker may attempt to find a pattern in the already unlocked ICs and build a model that could help him in performing a more efficient search for the unlocking sequence.

(ii) Replication attack. The attacker attempts to copy (clone) the random unique sequence of an authorized IC and then use the key received for the cloned IC to unlock it.

(iii) Read-only access to masks. An attacker who has access to masks may attempt to obtain the secret integer from the chip. The basic premise of the D-H cryptography scheme is that the integers from the two users and the final shared key are not revealed. Reading out the chip's secret facilitates breaking of the D-H secret sharing scheme.

(iv) Removal attack. The adversary may attempt to remove the bus locking circuitry so that the ICs will be unlocked upon manufacturing.

(v) Man-in-the-middle attack. In this attack, the adversary intercepts designer's public value and transmits its own public value to the IC. When the IC sends its public value, the attacker substitutes it with it own and transmits it to the designer. Therefore, the adversary and designer agree on one shared key and IC and attacker agree on another shared key. Now, the attacker can simply decrypt messages transmitted by the designer and the IC, and can read and potentially modify them before re-encrypting with the proper key and sending them to the other party. This

vulnerability is possible because D-H key exchange protocol does not authenticate the users.

(vi) Side-channel attack. Several cryptography protocols including D-H have been shown to be vulnerable to side-channel attacks. An attacker with access to the chip that runs these protocols can externally measure the power/timing of the signals many times. The adversary can use inference techniques on the IC's power/timing data to guess the key with a high probability.

The following countermeasures ensure the resiliency of the proposed method against various attacks:

- **Increasing the key length.** An effective way to eliminate the possibility of attack (i) is to add to the key length. The longer the key, the lower the probability of randomly guessing the correct combination. Note that the use of a long key as a counter measure is not specific to our scheme; all key-based security and cryptography methods, e.g., AES and 3DES, assume that random guessing of the key is computationally infeasible.
- **Unclonability.** A unique and unclonable identification bit string extracted from the IC can be integrated into the one-way function f [1]. Thus, attack (ii) will be ineffective since the key used for unlocking one IC will be a function of its unique variations and cannot be utilized for activating the busses of other chips. The unique and unclonable ID can be a part of the message communicated between the IC and the IP rights owner who will then use it as input to the one-way function. This countermeasure is also effective against attack (iii), since the unclonable IDs are typically a function of unique post-silicon manufacturing variability of each IC that is not available at the mask level.
- **Design complexity.** Miniaturization of devices and active integration of the locking/unlocking circuitry in the bus and address encoding/decoding would impede reverse-engineering. Thus, attack (iv) is not plausible since the attacker cannot distinguish/disintegrate or remove the lock.
- **One-way function.** The use of the one-way function that is integrated into our method deters the effectiveness of attack (v). In other words, even if a man-in-the-middle establishes two different channels one with the IC and one with the designer, he will not be able to compute the key specific to the IC or to use the key given for another chip.
- **Randomization.** To alleviate the correlation between the power and timing signals and the computed key, random timing and power activities can be added. Therefore, the side-channel attack (vi) will not be effective. A number of other methods that are commonly used for removing the information from the side-channel can be adopted, for example, one can equalize the D-H computation such that the peak power or time is not extractable from the differential external pin measurements.

7. VALIDATION

To ensure practicality of our proposed scheme, we demonstrate its low overhead and key-strength.

7.1 Minimizing design overhead

The overhead of the proposed bus-locking scheme can be traced to three components: (1) an implementation of the Diffie-Hellman (D-H) protocol, (2) pins and communication circuits, and (3) circuit-level bus locking. Two major factors help ensuring small overhead. First, the proposed protocol exchanges very few bits (<1000) and therefore does not require high speed. Second, most of the chip remains disabled during activation. Therefore, on-chip resources can be multiplexed and reused during activation. This particularly involves I/O pins, as well as arithmetic and cryptographic modules available on the chip.

Diffie-Hellman circuits. One of the reasons we chose D-H rather than the more recent RSA cryptography is that it is much easier to implement. While modern RSA circuits (e.g., the ones from OpenCores) require on the order of 15,000 standard cells, D-H can be implemented using one tenth of these resources [12]. In fact, D-H circuits are dominated by modular exponentiation which can be implemented efficiently by repeated squaring and modular multiplication. Most textbook multiplication circuits are sufficient in terms of speed, and no pipelining is necessary, because D-H circuits do not lie on critical paths in an activated chip. Area can be further minimized by using a half-sized multiplier with an adder. Moreover, several modern processors, such as Niagara1 and Niagara2 from Sun include support for cryptography and particularly fast modular exponentiation (they include RSA as well). Since the processor remains dormant during activation, it may be possible to use its arithmetic circuits for D-H. Alternatively, if a stand-alone D-H implementation is used, one can turn off (gate) its clock and power trunks when they are not needed.

I/O pins. Given the small number of bits transferred during activation, the entire exchange can be serialized through a single I/O pin using a simple handshaking protocol. Moreover, given that the chip remains largely disabled before activation, one of its existing pins can be multiplexed to support activation. Such multiplexing, however, may entail a small increase in latency during normal use. The communication circuits required for serialization, deserialization and handshaking implement very simple FSMs with only a handful flip-flops each. These only operate during activation and can be turned off during normal use.

Circuit-level bus locking. While circuit-level bus locks do not require as many gates as modular exponentiation circuits for the D-H protocol, they cannot be turned off and may slow down the host bus. Therefore, they are the main source of overhead in the proposed bus-locking scheme. This is why we use permutation circuits from [8] that have already been optimized and implemented within microprocessor designs. Such circuits require only $2n \log_2 n$ MUX gates which is considerably smaller than an n-bit multiplier, for example. Their logic depth, when pipelined, is $\log_2 n$, which is comparable to an ALU for modern designs, producing permutation every cycle [8]. Thus timing is not degraded and only one cycle of latency is added for bus communications.

7.2 Key strength evaluation

The countermeasures described in Section 6 are sufficient to completely defeat attacks (ii)-(vi). The countermeasure for attack (i), brute-force key testing, is to increase key length so as to make the attack infeasible for modern circuits. Thus we perform a comprehensive analysis of Benes networks used in [8] to show that this method is secure.

The circuits described in [8] for arbitrary bit permutations consist of $2 \log_2 n$ stages of n MUX gates each. $n/2$ bits control each stage for a total of $n \log_2 n$ key bits. As there are $n \log_2 n$ bits per key, this gives $2^{n \log_2 n} = n^n$ possible key combinations, which is much larger than the number of permutations of n bits, $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ (by Stirling's

# Valid key combs	# Perms	Total key combs	% Key combs
128 ×	8192 =	1048576	6.25%
256 ×	14336 =	3670016	21.88%
512 ×	12288 =	6291456	37.50%
640 ×	2048 =	1310720	7.81%
1024 ×	2816 =	2883584	17.19%
2048 ×	512 =	1048576	6.25%
4096 ×	128 =	524288	3.13%
Total	40320	16777216	100%

Table 1: Input collisions in an 8-bit Benes network. The first column gives the number of equivalent key combinations for a permutation. The second column counts permutations with that number of equivalent keys. The third column aggregates key combinations (out of 2^{24}) that the row covers.

approximation). If key combinations were mapped to permutations by the circuit uniformly, approximately $\frac{e^n}{\sqrt{2\pi n}}$ key combinations would map to each permutation. Even though the number of keys that map to a permutation grows nearly exponentially, the probability of guessing a valid key combination at random is 1 in $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \approx n!$. Thus a brute-force attacker would need to test $n!$ key combinations on average to find a working key.

Unfortunately, permutation circuits do not map keys to permutations uniformly. We fully analyze an 8-bit Benes network, shown in Figure 3, to see how non-uniform the mapping of keys to permutations is and what ramifications this has on the effective bit-length of permutation keys. To completely analyze the behavior of the circuit, we use ROBDD-based equivalence checking [4] for each of the $8! = 40320$ permutations of 8 bits. We construct a *miter* circuit for each permutation which produces a 1 as output when the circuit produces the *correct* permutation, universally quantify out all of the non-key inputs, and count the number of key combinations that make the miter evaluate to 1 using standard ROBDD operations.

Table 1 shows complete statistics for an 8-bit Benes network. The degree of non-uniformity in mapping key combinations to permutations is somewhat surprising. In the best case, there are 8192 permutations where only 128 key combinations map to that permutation, which makes for an effective key length of $\log_2(2^{24}/128) = 17$. In the worst case, there are 128 permutations for which 4096 key combinations map to the permutation. These permutations, one being the identity permutation, have an effective key length of $\log_2(2^{24}/4096) = 12$. If we extrapolate the worst case to larger circuits, such that the worse case effective key length is half the total key length, 32-bit busses would be protected by 80-bit keys, and 64-bit busses by 192-bit keys. To crack a 32-bit permutation, an attacker would need to check $2^{80} \approx 10^{24}$ keys. If the attacker had access to one thousand 5GHz processors that can check one key per cycle, it would take over 7000 years to crack a 32-bit permutation; cracking a 64-bit permutation would take over 10^{37} years.

We have also considered extensions of brute-force attacks, such as the use of rainbow tables and birthday paradoxes, but space limitations preclude detailed analysis in this paper. In brief, rainbow tables can be defeated using the standard "salting" technique used to store Linux passwords, and birthday paradoxes provide at most a square-root speed-up over brute-force, which would be insufficient to crack 64-bit permutations according to our calculations.

8. CONCLUSIONS

We propose the first bus-based IC locking and activation scheme, that works by uniquely locking each chip at the manufacturing site. The locking is performed by unique random IDs on each chip and D-H key sharing between the IP rights owner and the chip. We demonstrate the flow of the new scheme, discuss its wide range of applications, devise an implementation based on permutations and one-way functions, and present the attacks and countermeasures. Evaluation results confirm that the locking scheme has a very low overhead while it is highly resilient against attacks.

9. REFERENCES

[1] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security", *USENIX Security*, pp. 291-306, 2007.

[2] Y. Alkabani, F. Koushanfar and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management", *ICCAD*, pp. 674-677, 2007.

[3] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", *Computer* 35(1), pp. 70-78, 2002.

[4] G. D. Hachtel and F. Somenzi. *Logic Synthesis And Verification Algorithms.* Kluwer, 2000.

[5] A. Kahng, J. Lach, W. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection", *DAC*, pp. 776-781, 1998.

[6] K. Keutzer, S. Malik, R. Newton, J. Rabaey and A. Sangiovanni-Vincentelli, "System Level Design: Orthogonalization of Concerns and Platform-Based Design", *IEEE TCAD*, 19(12), pp. 1523-1543, 2000.

[7] J. Lach, W. Mangione-Smith and M. Potkonjak, "Signature hiding techniques for FPGA intellectual property protection", *ICCAD*, pp. 186-189, 1998.

[8] R. B. Lee et al., "Single-Cycle Bit Permutations with MOMR Execution," *J. Comp. Sci. Tech.* 20(5), 2005.

[9] K. N. Patel, I. L. Markov and J. P. Hayes, "Efficient Synthesis of Linear Reversible Circuits", *IWLS*, pp. 470-477, 2004.

[10] J. Rabaey, K. Keutzer, M. Sheets, S. Malik, A. Mihal, A. Sangiovanni-Vincentelli and M. Sgroi, "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", *DAC'01*, pp. 667-672.

[11] J. A. Roy, F. Koushanfar and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," *DATE*, pp. 1069-1074, 2008.

[12] B. Schneier, *Applied Cryptography.* John Wiley & Sons, 1996.

[13] V. V. Shende, A. K. Prasad, I. L. Markov and J. P. Hayes, "Synthesis of Reversible Logic Circuits", *IEEE TCAD* 22(6), pp. 710-722, 2003.

[14] M. Sgroi, L. Lavagno and A. Sangiovanni-Vincentelli, "Formal Models for Embedded System Design", *IEEE Design and Test of Computers* 17(2), pp. 14-27, 2000.

[15] S. Trimberger, "Trusted Design in FPGAs", *DAC*, pp. 5-8, 2007.