Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management

Farinaz Koushanfar, *Member, IEEE*

***Abstract*—In the horizontal semiconductor business model where the designer's intellectual property (IP) is transparent to foundry and to other entities on the production chain, integrated circuits (ICs) overbuilding and IP piracy are prevalent problems. Active metering is a suite of methods enabling the designers to control their chips postfabrication. We provide a comprehensive description of the first known active hardware metering method and introduce new formal security proofs. The active metering method uniquely and automatically locks each IC upon manufacturing, such that the IP rights owner is the only entity that can provide the specific key to unlock or otherwise control each chip. The IC control mechanism exploits: 1) the functional description of the design, and 2) unique and unclonable IC identifiers. The locks are embedded by modifying the structure of the hardware computation model, in the form of a finite state machine (FSM). We show that for each IC hiding the locking states within the modified FSM structure can be constructed as an instance of a general output multipoint function that can be provably efficiently obfuscated. The hidden locks within the FSM may also be used for remote enabling and disabling of chips by the IP rights owner during the IC's normal operation. An automatic synthesis method for low overhead hardware implementation is devised. Attacks and countermeasures are addressed. Experimental evaluations demonstrate the low overhead of the method. Proof-of-concept implementation on the H.264 MPEG decoder automatically synthesized on a Xilinx Virtex-5 field-programmable gate array (FPGA) further shows the practicality, security, and the low overhead of the new method.**

***Index Terms*—Active hardware metering, anti-piracy, chip overbuilding attack, chip protection, hardware obfuscation, integrated circuit (IC) piracy protection, integrated circuit (IC) security and trust, provably secure hardware metering, third-party intellectual property (IP) protection.**

I. Introduction

THE escalating cost of updating and maintaining silicon foundries has caused a major paradigm shift in the semiconductor business model. Many of the key design houses are entirely fabless (i.e., without a fabrication plant), outsourcing their fabrication to third-party providers. Several design companies that have traditionally fabricated their designs inhouse, have either formed alliances to share the cost, or have moved parts of their fabrication offshore to third-party providers.

In the earlier vertical market model, inhouse fabrication together with the clandestine nature of the packaged chips were enough for protection of design intellectual property (IPs). In the new business model, however, fabrication outsourcing requires revealing the design IP to external entities, creating many opportunities for IP infringements. The problem is exacerbated by contracting the offshore foundries to lower the labor and manufacturing cost, since many such fabrication plants are in countries with malpractice of IP enforcement laws [1]. The Alliance for Gray Market and Counterfeit Abatement has estimated that about 10% of the leading edge technology products available on the market are counterfeits [2]. Several government and industry task forces with members from the leading semiconductor companies are actively working to address the important problem of counterfeiting [3]–[5].

To enable tracking of the designer IPs and integrated circuits (ICs), a suit of new security mechanisms and protocols called *hardware metering* was introduced [6]–[9]. Metering enables the designers to have postfabrication control over their designed IPs by passively or actively monitoring or counting the number of produced ICs, monitoring their properties and usages, and by remote runtime enabling/disabling. The term hardware metering was originally coined by Koushanfar, Qu, and Potkonjak to refer to methods for unique functional identification of ICs made by the same mask [6], [7]. Their metering methods were the first that could be used for specific functional tagging of ICs, or for monitoring and estimating the number of fake components in case of piracy detection. Methods for unique identification of chips based on process variations were available earlier, but the hardware metering was the first to integrate the unique chip identifiers into the IC's functionality.

While passive methods for metering by postsilicon processing or by adding programmable parts were proposed about a decade ago, the first set of active methods for metering or tracking ICs was more recently introduced [9]. Several subsequent metering methods have been proposed since [10]–[15]. For a comprehensive review and novel classification of metering, we refer the interested readers to recent surveys on the topic [16], [17].

In a typical hardware metering scenario, each chip is uniquely and unclonably identified, for example by using a physical unclonable function (PUF) module [18]–[20]. A PUF typically extracts the unique delay or current variations on each chip to assign a set of unclonable identifiers (IDs). To meter, the IDs are linked to parts of the IC's functional components, e.g., the combinational part or the sequential part of the computer circuitry. This way, a part of the design functionality is uniquely tailored to the unclonable properties (fingerprint) of the IC and is used

Manuscript received January 20, 2011; accepted April 06, 2011. Date of publication July 29, 2011; date of current version January 13, 2012. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant W911NF-07-1-0198, in part by the Office of Naval Research (ONR) under Grant R16480, and in part by the National Science Foundation CAREER Grant R3A530. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Miodrag Potkonjak.

The author is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: farinaz@rice.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2011.2163307

to form a unique lock for each IC's functionality [9]. Only the designer who has the knowledge of the high level design would be able to find the specific key to unlock each IC. Security of the current active control methods is ensured by one of the two approaches: 1) Expanding the finite state model of the functional specification such that the added states and transitions are hidden in high-level design and are only known to the designers [9], [21]; or 2) employing a known cryptographic primitive such as public key cryptography, secret sharing, or Advanced Encryption Standard (AES) [10]–[12]. The recent surveys on active metering distinguishes between the two methods [16], [17]. The first set of methods based on introducing locks embedded in the behavioral description of the design are called *internal active IC metering*. The second set of methods based on embedding locks within the design and interfacing (controlling) the access by an external cryptography function are termed *external active IC metering*.

This paper presents a detail description and comprehensive analysis of the first known active metering method originally introduced in [9] along with several new results. The locking and controlling methods introduced in this paper are internal, and sequentially designed. We emphasize that although combinational-only internal locks have been used in the context of external active hardware metering [10]–[12], those locks are interfaced with cryptographic hardware that is often sequentially implemented. Therefore, the control method as a whole is a sequential design. Furthermore, the power/area overhead of the cryptographic module interfaced with the combinational locks is greater than the overhead of the internally embedded control circuitry.

A novel aspect of this paper compared to previously published active hardware metering methods, including [9], is introducing secure construction of the active metering by extending the chip's behavioral specification in the finite states domain. We show that our construction of locks by finite state manipulation and compilation during the hardware synthesis and interfacing to a unique PUF state is an instance of an efficiently obfuscatable program under the random oracle model. Even though heuristic methods for finite state machine (FSM) obfuscation were proposed earlier, e.g., [22] and [23], no provable security for such a construction was available. The significance of our construction and security proofs for the obfuscated FSM goes beyond hardware metering and extends to the earlier work in information hiding and obfuscation of sequential circuits, e.g., [22] and [23]. The experimental evaluations presented in this paper are all new as they implement the novel lock construction method. Our contributions are as follows.

1) We comprehensively present the first known method for active IC metering and IC piracy prevention. The method uniquely locks each manufactured IC at the foundry.
2) The locking structure is embedded during hardware synthesis by FSM modifications. The locks are designed such that the IC would not be functional without a proper chip-specific passkey that can only be computed by the designer (IP rights owner).
3) We show the analogy between the hardware synthesis transformations and program compilation. We pose the problem of extending the FSM for hiding the locks as an instance of the classic program obfuscation problem.

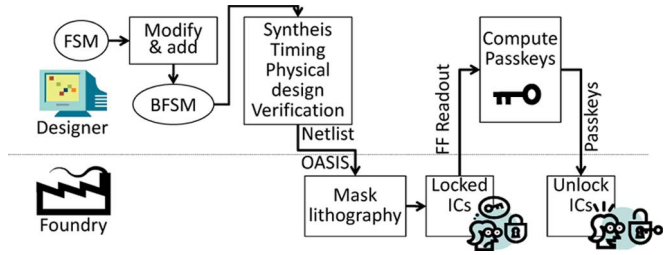

Fig. 1. Global flow of the novel IC activation approach.

4) We demonstrate a construction of the locks within FSM as an instance of a general output multipoint function family. This family is known to be effectively obfuscatable in the random oracle model. Therefore, the locks can be efficiently hidden.
5) Automatic low-overhead implementation of secure metering lock structure during synthesis is demonstrated by obfuscatable topology construction, secure passkey selection, and iterative synthesis.
6) Potential attacks and security of the presented method against each attack are discussed.
7) Experimental evaluations and the overhead of the new hardware metering construction is demonstrated on standard benchmark circuits.
8) Proof-of-concept implementation of the active hardware metering for an H.264 MPEG decoder on field-programmable gate array (FPGA) is done and the overhead is reported. To the best of our knowledge, this is the first hardware implementation of any known active hardware metering methods clearly showing applicability and practicality of our method.

The remainder of this paper is organized in the following way. Section II introduces the global flow. The necessary background is outlined in Section III. The related literature is surveyed in Section IV. Details of the active hardware metering method is presented in Section V, where secure hiding of the locks within the FSM structure is discussed. We also briefly mention a number of potential applications of active metering. Section VI discusses the details of the automatic synthesis and implementation. Attacks and countermeasures are discussed in Section VII. Experimental evaluations and hardware implementation are presented in Section VIII. We conclude in Section IX.

II. Flow

Fig. 1 shows the global flow of the first known active hardware metering approach as described in [9]. Similar flows were later adopted for both internal and external active ICs metering. There are typically two main entities involved: 1) a design house (a.k.a., *designer*) that holds the IP rights for the manufactured ICs, and 2) a foundry (a.k.a., *fab*) that manufactures the designed ICs.

The steps of the flow are as follows. The designer uses the high level design description to form the design's behavioral model in the FSM format. Next, the FSM is modified so that the extra locking structure composed of additional states and transitions is integrated within the FSM. The term boosted finite state machine (BFSM) is used to refer to the modified FSM. The subsequent design phases (e.g., RTL, synthesis, mapping, layout, and pin placement) take their routine courses.

The foundry would receive the OASIS files (or GDS-II) and other required information for fabricating the chips, and also the test vectors. The test vectors include the challenge set (input) to be applied to the PUF unit on each IC [20]. The design house typically pays the foundry an up-front cost for a mask to be lithographed from the submitted OASIS files and for the required number of defect-free ICs to be fabricated.

Building a mask is a costly and complex process, involving multiple fine steps that should be closely controlled [24], [25]. Once the foundry lithographs a mask, multiple ICs could be fabricated from this mask. Because of the specific PUF responses integrated within the locks on the chips, each IC would be uniquely locked (nonfunctional) upon fabrication. During a startup test phase, the fab inputs the challenge vectors to the chips that would run through the scan chains. The states stored in the chip FFs (a.k.a., *power-up state* or the *startup state*) would be scanned at this point. The scanned FF values are sent back to the design house (IP rights owner) who has the full specifications of the hidden states. The design house is the only entity who could compute the unlocking sequence for each locked chip. This paper introduces provably secure BFSM construction methods so the transitions from each power-up state to the original reset state can be efficiently hidden. Additionally, the designer often computes the error correcting code (ECC) to adjust for any further changes to the startup state because of the noise or other sources of physical uncertainty. The ECC is very important since a few of the PUF response bits may be unstable and altered at a later time because of noise, environmental conditions (e.g., temperature), or circuit instability [26]. The key and the ECC would then be sent back to the fab.

The nonvolatile on-chip memories would be used to store the power-up test vectors (PUF challenges), the unlocking sequence (passkeys), and the relevant ECC bits on each pertinent activated IC. From this point on, every time the IC starts up, it would automatically read the passkey and ECCs and use them for traversing the chip to a working state. As it can be easily seen, active metering is nicely integrated within the regular phases of the hardware design, synthesis, and tape-out flow. The only added phase is the key exchange protocol for unique activation of each IC. The common functional and structural tests would be done on the unlocked ICs.

We note that if a defect or a fault affects the states and sequences traversed during the startup state or the traversal from the locked startup state to the functional state, the IC cannot be unlocked and would be defective.

III. Background and Assumptions

We introduce a number of general terms and concepts that are used throughout the paper. More specific definitions would be described as necessary.

Physical unclonable function. PUFs are an emerging primitive for many device level security and privacy protocols [18], [19], [27]. PUF is a physical function that provides a mapping from its inputs to outputs based on the unique fluctuations in the unclonable device material properties. The PUF input is typically called a *challenge* and the PUF output is commonly called a *response*. To ensure security, the mapping should be such that the responses can be rapidly evaluated, but they are hard to model, characterize, or reproduce. PUFs have been classified to a few categories, including weak PUFs and strong PUFs [20]. A weak PUF (also called a physically obfuscated key) can only generate a very limited (fixed) number of independent outputs. It can be used for generation of secret keys and is secure as long as the adversary does not have a way to read out the challenge/response pairs for the structure. In contrast, a strong PUF has many possible challenge/response pairs and can be used in applications other than secret key generation such as authentication and integrity checking.

Note that even though our described metering scenario includes a type of PUF, our technique is independent of the hardware ID extraction method. Many proposed structures for PUF can be used for ID generation. For example, a weak PUF can be used, or a strong PUF can be used for making a weak PUF [20]. As long as we can extract a set of unclonable digital identifiers (fingerprints) from the device based on its physical disorders, our method is secure. The only assumption made about the PUF is that with a very high probability it produces a unique random number on each device, and that responses are reasonably stable. The PUF should also be hard to tamper/remove.

Design description. We consider the case where the sequential design represents a fully synchronous flow. The description of the design input/output functionality is publicly available. We assume that the functional description is fully fixed, and the I/O behavior is fully specified. Our metering technique is applicable when the IP is available in structural HDL description, or in form of a netlist that may or may not be technology dependent. Thus, it can protect both firmware and hardware [28]. During the IC design flow, the designer maps the circuit behavioral description to a specific technology provided by the target foundry. Several logic-level optimizations (including timing closure, power optimizations, and synthesis transformations) are applied by the designer. Very often, a designed IP is integrated within a larger circuit or a system-on-a-chip.

Finite state machine. Digital sequential circuits are commonly modeled by a finite set of states, transitions between the states, and actions that can be abstracted by a FSM computational model. A deterministic FSM is often formally defined by a tuple FSM $= (\Sigma, \Delta, S, s_0, \delta, \lambda)$ where

$\Sigma \neq 0$	is a finite set of input symbols;
$\Delta \neq 0$	is a finite set of output symbols;
$S = \{s_0, s_1, \ldots\} \neq 0$	denotes a finite set of states;
s_0	is the FSM "reset" state;
$\delta(s, i)$	is transition function on s and $i(S \times \Sigma \rightarrow S)$;
$\lambda(s, i)$	is output function on s and $i(S \times \Sigma \rightarrow \Delta)$.

In δ and λ definitions, s is the state and i is the input. To represent the transitions and output functions of the FSM, we use the state transition graph (STG) with nodes corresponding to states and edges defining the transition conditions based on the current state and the edge inputs. Throughout the paper, we use the terms STG and FSM interchangeably.

Assumptions. The hardware metering objective is to protect the ICs so they cannot be pirated or overbuilt by the foundry. There are a number of realistic assumptions that we make about the potential adversary (foundry). First, the foundry has access to layout (e.g., OASIS or GDS-II files) and the netlist but does not have access to FSM because: 1) the layout files, the netlist, and the test vectors are sufficient for fabricating the chip and a high level behavioral description is not needed, and 2) the details of the FSM behavioral description is a key part of the designer's IP and trade secret that maintains their revenue in the competitive semiconductor market. Second, a significant design modification would impact power, yield, and most importantly, timing. In such scenarios, redoing timing, physical design, verification, and debugging would require an effort equivalent to designing a new IC and a new mask. Therefore, the attack cost and complexity does not justify its benefits. Third, the scan chains are available and it is possible to scan and readout the FF values storing the FSM states on each chip. Fourth, the designer's objective is to protect her/his design from piracy and other related tampering. The designer's objective is not to protect the overall functionality. For example, while designing an H.264 media player, the different submodules are known at the protocol level and at the block level. However, the specific details of a design are to be protected by the design house (so it cannot be readily reproduced) for competitive advantage reasons.

IV. Related Work

We survey the related literature along the lines of ongoing efforts for counterfeit detection, unique circuit identification and PUFs, passive and active metering, hiding secret in FSM, and provable obfuscation (hiding information) for programs.

Counterfeiting has been identified as a major challenge by several major U.S. design companies. The Semiconductor Industry Association (SIA) has formed an anti-counterfeiting task-force (ACTF) with experts from 17 of its 95 member companies [29]. The main goal of ACTF is to support actions and countermeasures against counterfeiting and IP violations with both government and private agencies. Several methods, tools, and guidelines are being developed to reduce counterfeiting. Currently, ACTF is collaborating with SEMI (an industry association and standard generating body for the semiconductor manufacturing supply chain) for developing standards for the authentication service providers in the supply chain. Another key international organization in this domain is the Alliance for Gray Market and Counterfeit Abatement (AGMA) [4].

The first passive hardware metering approach by integrating the uniqueness into the functional description was introduced about a decade ago [6], [7]. The idea was to assign a unique signature to each IC's functionality by integrating a small flexible part to the ASIC. Each IC would be unique with a different control mechanism dictated by the various schedules that were all implementing the same design. The flexible part could be an added programmable component, or it might be a function of unclonable IC variations. One possible advantage of having a part of the chip programmable is that the designer could retain a part of the schematic as a design secret. However, there are at least two limitations to this approach. One has to do with the overhead and cost of augmenting programmable components on a typical ASIC technology because of the additional mask layer requirements. The other one is related to reproducibility. For example, even if a part of the schematic is retained secret during fabrication, the foundry can overbuild the unprogrammed circuits. Getting hold of even one reproducible postprogrammed circuit instance would be enough for overbuilding attacks.

This paper presents, complements, and advances the first-known active metering method in [9]. Compared to the earlier work, a suit of new concepts, analyses, security guarantees, implementation methods, and results are demonstrated. Another possible FSM-based metering is based on integrating a few new states, but manipulating the transitions and adding an exponential number of transitions in the circuit [21]. Because of the differences in the FSM manipulation method and the continuous checking provided by the multiple distributed locks and paths, there is very little overlap between this paper and the work in [21]. For example, our method only affects the activation and disabling phases and does not incur a significant timing overhead during the circuits regular functionality.

A number of subsequent work in active locking and unlocking of ICs included using asymmetric cryptography and secret sharing [10]–[12]. Those methods are useful for larger chip sets, since it is well known that public key cryptography (PKC) has a high overhead on small chips and in embedded systems [30]. The overhead is much larger than the FSM-based protection presented in this paper. Note that the PKC is usually implemented using sequential circuits. Thus, even though it is possible to only lock the combinational part of the circuit to be protected, the combinational locks need to be integrated with a sequential cipher, resulting in an overall sequential behavior.

Oliveira was the first to use FSM for hiding a secret watermark in a sequential circuit [28]. The watermarking was performed by manipulating the STG to exhibit a chosen state transition that was extremely rare in a nonwatermarked circuit. Simultaneously, the watermark did not change the circuit's functionality. Yuan and Qu have pursued the idea of hiding information in the FSM with the goal of minimizing the overhead in concealed data [22]. To achieve this, the hidden information is embedded within the "don't care" states. The FSM is manipulated post state minimization using the knowledge of maximum set of redundant specification. While we also utilize the concept of hiding data in FSM, our approach is drastically different than watermarking or information hiding. We add an exponential number of states to the FSM to create a secure multipoint function with a generalized output.

It is also worth emphasizing the differences between IC watermarking as a proof of design authorship and IC metering. Watermarking is radically different since it embeds a unique signature in each IP, unlike metering that assigns a specific signature to each IC. The authorship of an IP can be then proved in legal settings if needed. A watermark cannot track or monitor, passively or actively, the number of copies that were fabricated from one mask. We refer the readers interested in watermarking to key papers and books in this area, including [22], [28], and [31]–[34].

There are multiple new concepts and enhancements in this paper compared to [9] including: 1) new construction of BFSM according to the provable obfuscation of multipoint function

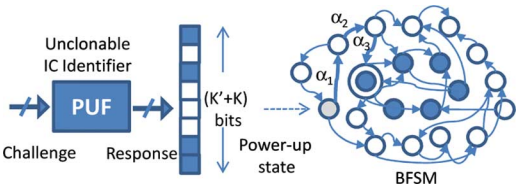

Fig. 2. PUF response is fed to the FFs storing the states of the BFSM. The original states are shown in dark, and the added states are demonstrated in white color on the STG that represents the BFSM.

with a generalized output, for secure concealing of the locks within the original design; 2) comprehensive discussion of the security of the new provably secure construction and providing countermeasures against a new possible set of attacks; 3) new experimental evaluations that yield a different overhead result because of the new provable obfuscation; and 4) results and proof-of-concept low overhead implementation of the new secure methods on state-of-the-art H.264 MPEG decoder on Virtex-5 Xilinx FPGA.

V. Secure Active Hardware Metering Methodology

During the IC design flow, a designer devises the behavioral specification in an FSM format. At this stage, the FSM will be modified to include multiple added states and transitions. This modified FSM is called the BFSM. The initial power-up state of the BFSM is determined by the PUF and is unique for each IC.

A. Designing a BFSM

We first demonstrate the BFSM modification and active metering mechanisms using a small example. We then discuss parameter selection for ensuring the randomness and uniqueness of each activated IC. Comprehensive details of the parameter selection for a secure BFSM construction are discussed in later subsections.

On the small example in Fig. 2, the original FSM states are shown in dark color on the right side of the figure. The BFSM includes the original FSM, along with a number of added states that are shown in white. Assume that the original FSM has $|S|$ states. Therefore, it can be implemented using $K = \log|S|$ FFs. Now assume that we add to the number of states in FSM to build a BFSM with $|S'| + |S|$ states that can be implemented by $K'' = log\{|S'| + |S|\}$ FFs. Observe that for a linear growth in the number of FFs denoted by $K' = K'' - K$, the number of states exponentially increases.

On the left side of Fig. 2, there is a PUF unit that generates random bits based on the unclonable process variations of the silicon unique to each chip. A fixed challenge is applied to the chip upon power up. The PUF response is fed to the FFs that implement the BFSM. Since there are $K'' = \log\{|S'| + |S|\}$ FFs in the BFSM, one would need K'' response bits from the PUF for a proper operation.

As can be seen in Fig. 2, upon the IC's power up, the initial values of the design's FFs (i.e., *power-up state*) is determined by the unique response from the PUF on each chip. The PUF challenges are determined by fixed test vectors given by the designer. For a secure PUF design, the probability of the response should be uniformly distributed over the possible range of values [35]. The number of added FFs can be set such that the value $2^{K''} \gg 2^K$. In other words, the value K'' is set by the designer such that for a uniform probability of selecting the state, the probability of selecting a state in the original FSM is extremely low. We will leverage more on this point in Section V-B.

Because there are exponentially large numbers of added states, there is a high probability that the unique PUF response on each chip sets the initial power-up state to one of the added states. Note that unless the design is in one of the original states, it would be nonfunctional. Therefore, the random FF states driven by the PUF response would place the design in a nonfunctional state. One would need to provide inputs to the FSM so it can transition from this nonfunctional initial power-up state to the functional *reset state* of the original FSM shown by double circle on the example.

For the IP rights owners with access to the BFSM state transition graph, finding the set of inputs for traversing from the initial power-up state to the reset-state (shown by double circle on the figure) is easy. All that is needed is to form a path on the graph and use the input values corresponding to the path transitions (from the STG description) so the chip transitions to the reset state. However, there is only one combination from an exponentially large number of possibilities for the input corresponding to each edge transition. Thus, it would be extremely hard for anybody without access to the BFSM edge transition keys to find the exact inputs that cause traversal to the original reset states. The access to the full BFSM structure and the transition function on its edges is what defines the designer's secret. The passkey for unlocking the chip is the sequence of inputs that can traverse the BFSM states (describing the control component of the chip) from the initial random power-up state to the reset state. Note that although the initial power-up state is random, the assumption is that for a given PUF input (challenge), the response remains constant over time for one chip.

A set of passkeys $(\alpha_1, \alpha_2, \alpha_3)$ required for traversal from the power-up state to the reset state is shown in Fig. 2. This locking and unlocking mechanism provides a way for the designer to *actively control (meter)* the number of unlocked functional *(activated)* ICs from one blueprint (mask), and hence the name active hardware metering. In Section V-C, we will describe a number of other important applications of this active control method.

While constructing the BFSM for hardware metering purposes, a number of requirements must be satisfied. The first set of requirements has to do with the probability of randomly powering-up in a state that was not in the original FSM. Let us assume that by design, we require this probability to be lower than a given value ϵ. This low probability is satisfied by the following two conditions:

1) The value $|S'|$ should be selected such that the probability of not powering-up in an added state is smaller than ϵ:

$$P(\text{power} - \text{up} \in S') = \frac{S' - S}{S' + S} \geq 1 - \epsilon. \quad (1)$$

2) The value $|S'|$ should be selected so that the probability of two ICs having the same startup states is extremely low. Assume that we need to have m distinct ICs each

with a unique startup state. Fortunately, for a linear increase in the number of FFs, we obtain an exponential increase in the number of states. The unclonable response from the PUF is used to set each IC in a unique random state. To achieve completely random and independent states, one can employ the Birthday paradox to calculate this probability and to set it to low values. Consider the probability $P_{\text{collision}}(S', m)$ that no two ICs out of a group of m will have matching startups out of $|S'|$ equally possible states. Assume $S' \gg S$. Start with an arbitrary chip's startup. The probability that the second chip's startup is different is $2^{K''} - 1/2^{K''}$. Similarly, the probability that the third IC's startup is different from the first two is $2^{K''} - 1/2^{K''}.\ 2^{K''} - 2/2^{K''}$. The same computation can be extended through the $2^{K''}$th startup. More formally

$$P_{\text{collision}}(K'', m) = \frac{2^{K''}-1}{2^{K''}} . \frac{2^{K''}-2}{2^{K''}} \cdots \frac{2^{K''}-(m-1)}{2^{K''}} = \frac{2^{K''}!}{(2^{K''}-m)!2^{m.K''}}. \quad (2)$$

For a large value of $2^{|S'|}$, the formula can be asymptotically approximated [36]

$$m = \sqrt{2^{|S'|+1} \times ln\left(\frac{1}{1-P_{\text{collision}}(|S'|, m)}\right)}. \quad (3)$$

This equation yields the approximate closed formula

$$P_{\text{collision}}(|S'|, m) = 1 - e^{-m^2/2^{|S'|+1}}. \quad (4)$$

Another set of important requirements has to do with the BFSM edge traversal and state reachability. For an STG with the initial reset state (denoted by s_0), we call this graph *reset state reachable* if and only if for each and every state in STG there is at least one sequence of inputs $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ of arbitrary cardinality such that it can be applied to the pertinent state, and the final transition destination (after applying the input sequence) is the reset state. From this definition, it is clear that the desired BFSM needs to be reset-state reachable.

B. Secure BFSM Construction

In this subsection, we show that our construction of finite state manipulation, compilation by hardware synthesis, and interfacing to the unique IDs coming from the PUF comprise an instance of an efficiently obfuscatable program.

1) Circuit Compilation and Obfuscation for Metering: Let us go through the steps of hardware design and synthesis flow. The designer starts at a functional description level that can be demonstrated by an FSM, whose states and transitions are known. Given the FSM model, for each input, the designer would be able to compute the corresponding output. The specifications are typically implemented in a hardware description language (HDL) format, e.g., VHDL, or Verilog. The register-transfer level (RTL) description of this HDL format along with the technology libraries and design constraints are then input to the hardware synthesis tool. The output of the synthesis tool may be either in the form of a netlist with boolean gate types, or might be mapped to a specific library. The remaining steps of the design process use the resultant netlist.

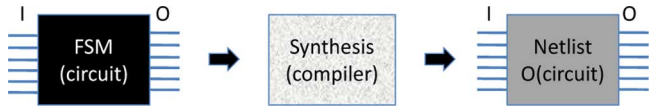

Fig. 3. Active hardware metering as a program obfuscation.

Fig. 3 demonstrates a black box model of the metering synthesis flow from the high level computational model (modeled by an FSM) to the lower level netlist. In effect, hardware synthesis is a compiler. Recall that a compiler provides a transformation from a source program written in a higher level computer language (source language) into another lower level computer language (target language). Here, the input source program is the description of the circuit behavior in finite state automata domain, and the target program is the circuit description in the netlist domain.

The FSM-based hardware metering attempts at hiding information in a source program by modifying the FSM. The modifications add multiple states to the STG, where the modified expanded state-space is used for hiding the power-up state. The objective is to hide information such that the concealed data cannot be extracted from the target program presented in a lower level netlist. The two programs have (almost) the same functionality from the input/output program perspective. In effect, the objective of this type of active hardware metering is to build a *program obfuscation* that would conceal the secret passkey information (shown by $\alpha_i s$ on Fig. 2) so the hidden state and its corresponding traversal cannot be retrieved from the netlist. Informally speaking, an obfuscator is a compiler that transforms a program (e.g., a Boolean circuit) into an obfuscated program (also a circuit), such that the obfuscated program has the same input/output relationships as the original program, but is otherwise obscure (unintelligible).

2) Secure Program Obfuscation: Let us discuss obfuscation in a more formal setting and outline the positive results on this subject. Under a random oracle model,[1] a function family $\mathcal{F}$ can be obfuscated when there is an algorithm $\mathcal{O}$ that takes an input in form of a Turing machine (i.e., a program, a circuit) computing $P \in \mathcal{F}$ and outputs another Turing machine (circuit) such that the *obfuscating requirements* are satisfied [37]. If the requirements are assured, $\mathcal{O}$ is an obfuscator for $\mathcal{F}$, and the obfuscation of the program is shown by $\mathcal{O}(P)$. $\mathcal{O}$ is called *efficient* if its computations are done in polynomial time, and then $\mathcal{O}(P)$ is said to be efficiently obfuscatable. Before we formally outline the requirements, let us define a number of notations. The notation k specifies the *feasibility parameter* that is associated with one family $\mathcal{F}_k$ of functions that we obfuscate; the size of

[1]In cryptography, a random oracle is a mathematical abstraction used in proofs when no implementable function (except for an oracle) could provide the properties required.

$P \in \mathcal{F}_k$ is polynomial in k. The family $\mathcal{F}$ is the union of the families $\mathcal{F}_k$.

a) *Approximate functionality.* There is a negligible function ν such that $\forall k$ and $\forall P \in \mathcal{F}_k$, we would have

$$\mathbf{Pr}[\exists \text{ inputs } x \in \{0,1\}^* : \mathcal{O}(P)(x) \neq P(x)] \geq \nu(k).$$

b) *Polynomial slowdown.* There is a polynomial p such that $\forall P \in \mathcal{F}$, $|\mathcal{O}(P)| \leq p(|P|)$ and for the Turing machine if P spends t time steps to compute on the input $x \in \{0,1\}^*$, $\mathcal{O}(P)$ would spend $p(t)$ time steps at most.

c) *Virtual black box.* For a probabilistic polynomial time Turing machine $\mathcal{A}$, there is another probabilistic polynomial time Turing machine $\mathcal{S}$ and a small negligible function denoted by ν such that $\forall P \in \mathcal{F}$ we have $|\mathbf{Pr}[\mathcal{A}(\mathcal{O}(P)) = 1] - \mathbf{Pr}[\mathcal{S}^P(1^{|P|}) = 1]| \leq \nu(|M|)$, where the probabilities are over $\mathcal{A}$ and $\mathcal{S}$ randomness.

The work in [37] was the first to start a formal theoretical study of obfuscation. In particular, they showed a negative result for the possibility of having a generic obfuscator for all function classes by demonstrating a family of functions that could not be obfuscated. Thereafter, several theoretical studies and results for obfuscation have emerged [38], [39]. The work in [39] provided the first positive results for provable obfuscation of the family of point functions in the random oracle model.

The simplest example for a point function is a program that requires a password to login and operate. The password is typically hidden in the program and should be obfuscated such that nobody else with access to the program would be able to extract the password as long as it cannot be guessed. Password hiding can be modeled as an obfuscation of a point function under the random oracle model. More formally, a point function $\{\mathcal{P}_\alpha\}$ is defined as a function $\mathcal{P}_\alpha$ such that for all inputs x, the function $\mathcal{P}_\alpha = 1$ if the input is equal to the specific access key (password) α. The function would not do anything otherwise.

Reference [39] has demonstrated a point function $\mathcal{P}$ satisfies the three properties specified for an efficiently obfuscatable function. This result was further extended to the family of multipoint functions that have a general output. The function $\mathcal{P}_{(\alpha_1,\beta_1),\ldots,(\alpha_q,\beta_q)}$ on the domain $\{0,1\}^k \rightarrow (\{0,1\}^{\varsigma(k)})^q$ is a q-point function and has a general output of length $\varsigma(k)$ iff

$$\mathcal{P}_{(\alpha_1,\beta_1),\ldots,(\alpha_q,\beta_q)} = \begin{cases} b \in (\{0,1\}^{\varsigma(k)})^q, & \text{where } b_i = \beta_i \\ & \text{if and only if } x = \alpha_i \\ \perp, & \text{otherwise.} \end{cases} \quad (5)$$

This latter proof is given by showing that the multipoint function can be self-decomposed to several smaller point functions. The generalized output multipoint function is a program with q passwords with a string output of length $\varsigma(k)$ (generalizing a single binary output case).

3) Provable Obfuscation of the Locks Within the BFSM: In this subsection, we show that a secure construction for the BFSM used in metering can be modeled as a multipoint function with a general output and thus can be efficiently obfuscated. The hardware metering approach introduced in [9] adds exponentially many states to the FSM such that the probability of powering up in one of the added states is extremely high. Because of the unique chip identifiers coming from the PUF, each chip would power-up in one of the added states. An added state is nonfunctional with a very high probability so the chip would be locked. The designer would be the only entity who can provide the *unique set of unlocking inputs* for transiting between the specific power-up state and the original "reset" state for each chip. In the remainder of the paper, we use the term *passkey* to refer to the set of inputs corresponding to each traversed edge during unlocking. Let us more formally define the problem.

INPUT: Given an original sequential specification of a circuit P in form of an FSM $= (\Sigma, \Delta, S, s_0, \delta, \lambda)$, and given a compiler that transforms this functional specification to a netlist.

OBJECTIVE: Construct a modified FSM denoted by BFSM $= (\Sigma + \Sigma', \Delta, S + S', s_0, \delta + \delta', \lambda + \lambda')$ such that transitions from a state $s' \in S'$ to one of original FSM states $s \in S$ would require "strong" passkeys for transiting the edges. A passkey sequence of length l denoted by $\alpha = \{\alpha_1, \ldots, \alpha_l\}$ applied to the state s' would result in a sequence of transitions before it gets to reset state s. By strong passkeys, we mean they are random and long such that they cannot be guessed by the brute force attack. Without the loss of generality, let us assume that each edge passkey length is fixed to the value k. The reached state s then would be $s = \delta(s', \alpha) = \delta(\delta(\delta(\ldots\delta(s', \alpha_1)\ldots), \alpha_{l-1}), \alpha_l)$. The corresponding output would be $\lambda(s', \alpha) = \lambda(\delta(\delta(\ldots\delta(s', \alpha_1)\ldots), \alpha_{l-1}), \alpha_l)$.

METHODOLOGY: To address the above problem, we demonstrate a BFSM construction such that the addition of states and transitions to the original graph is an instance of a general output multipoint function that is efficiently obfuscatable. Using this result, we devise a strong passkey mechanism to build a secure hardware metering.

To show the obfuscatable BFSM construction, we form a set of extra state-transition relations that are added to the original FSM. The addition is such that for reaching the states in the original FSM from each potential power-up state $s' \in S'$, one has to traverse one or more of the added states or transitions. Let us add a number of transitions to each state $s' \in S'$ such that each s' has at most t outgoing transitions denoted by $\delta'(s', \alpha_{s1}), \ldots \delta'(s', \alpha_{st})$. The upper bound t on the number of transitions from one added state is set such that there is a sequence of transitions to traverse from each added state to one of the states in the original FSM (our implicit assumption is that the "reset" state is reachable from all the states in the FSM). In other words, $\forall s' \in S', \exists \alpha = \{\alpha_1, \ldots, \alpha_l\}$ such that $s = \delta''(s', \alpha) = \delta''(\ldots\delta''(s, \alpha_1)\ldots), \alpha_{l-1}), \alpha_l)$, where δ'' maybe either δ (transition on the original state machine) or δ' (transition on the added state machine).

Assume that the BFSM is at the state $s' \in S$ and the user inputs a vector of inputs $\{x_1, \ldots, x_l\}$. Let us first define a function $W(x_1, \ldots, x_l)$ as follows:

$$W(x_1, \ldots, x_l) = \begin{cases} s \in S, & \text{if } \exists s'_0, \ldots, s'_l \text{ s.t. } s'_0 \equiv s' \text{ and} \\ & s'_l \equiv s, \text{ and } s_j = \delta''(s_{j-1}, x_j) \\ \perp, & \text{otherwise.} \end{cases} \quad (6)$$

Consider the family of functions $\mathcal{W}(.)$ over the set of all $W(.)$ that can be defined for the BFSM with parameters S' and t and

the transition functions δ''. Our interest is in cases where there is a polynomial number of traversed edges in terms of the parameters S' and t. We notice that there are exponentially many possible valid sequences of input transitions that can traverse from the initial power-up state to one of the original states. Therefore, the function $\mathcal{W}(.)$ cannot be immediately shown to be obfuscatable by directly using the theoretical results for obfuscating general output multipoint functions.

Instead of discussing the $W(.)$ functions on the general BFSM structure, we decompose the transition path to state-to-state transition edges. Each state s_j can be represented by its incident transition functions denoted by $\delta''^{(t')}_j(s_j, \alpha_j^{(t')})$ for traversal to the "neighboring states." The neighboring states, denoted by $s_j^{(t')}$ are those reachable by applying a single passkey, i.e., $s_j^{(t')} = \delta''^{(t')}_j(s_j, \alpha_j{}^{(t')})$, $0 < t' \leq t$. For each state $s_j^{(t')}$ there is a unique passkey $\pi(t')$ from $\{0,1\}^k$ (recall that k is the valid passkey length for the BFSM graph). Define the function $\omega(.)$ as follows:

$$\omega\left(s_j^{(t')}, \pi'', j\right) = \begin{cases} (s_j, \delta''^{(t')}_j(s_j, \pi(t'))), & \text{if } \pi'' = \pi(t') \;\&\; \exists s_j \text{ incident to } s_j^{(t')} \; s.t. \; s_j^{(t')} = \delta''^{(t')}_j(s_j, \pi'') \\ \perp, & \text{otherwise.} \end{cases}$$

The obfuscation of $W(.)$ consists of obfuscation of $\omega(.)$, which is a multipoint function with at most $t|S'|$ points where the output is not $\perp$ and hence can be efficiently obfuscated. Informally, this is an efficient obfuscation since the adversary (with a high probability) cannot guess a set of randomly selected keys that would result in a valid transition. Reference [39] has shown that a composition of obfuscatable functions is also efficiently obfuscatable under the random oracle model. Therefore, we conclude that $W(.)$ is also efficiently obfuscatable since it can be written as a composition of obfuscatable functions.

More importantly, it was demonstrated that for such functions that can be written by decompositions, exploring parts of the passkeys for transitions on the composite structure (in our case BFSM) would not reveal the remainder of passkeys, as long as the two passkey sets are not correlated and do not include all the same compositional unit (in our case the same state) [38], [39]. Therefore, as long as the traversal paths from the power-up state to the original set of states for a locked IC are at least different in one state from a previously unlocked IC, obfuscation of a multipoint function remains secure.

C. Additional Considerations and Applications

So far, we have described the construction of a BFSM such that the structure can be used for obfuscating the initial power-up state. It is clear that to ensure randomness and protection of the scheme, it is desired to have an unbiased PUF that can generate an output bit of 0 or 1 with completely equal probabilities. The diversity of the power-up state is guaranteed if the PUF output is completely random. The other important factor to consider is stability of PUF responses and its vulnerability to operational and temperature conditions. As we mentioned earlier, to ensure robustness and full operability in the presence of fluctuating operational and environmental conditions, we use ECCs in conjunction with PUF responses. As long as the PUF response is unclonable, the security of our method is not based on keeping the PUF output secret (it is the passkeys on the transition graph that are secret). Therefore, the ECC syndromes would not reveal much beyond the PUF value. The use of ECC for PUF is not new; several other works have used ECC in conjunction with PUF to ensure its robustness. We refer the readers to the related work in this domain for the studies of added overhead and security of different ECC methods [26]. Another important issue that we address is storage of the passkeys. Once the passkey is given for unlocking one chip, its value would be stored on NVM inside the chip along with the ECC code. Therefore, every time the IC is powered up, it automatically reads the passkeys to transition to the original reset state.

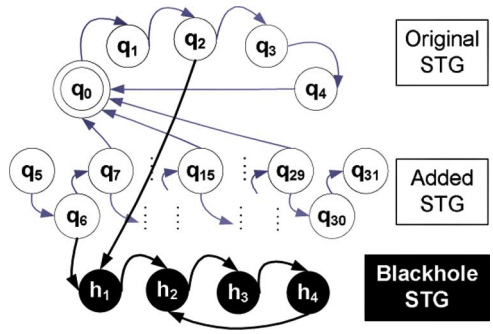

Fig. 4. Example of a blackhole FSM that can be used for online controlling and disabling the ICs. They can also provide a countermeasure against the brute force attacks (Section VII).

An interesting and important observation is that the mechanisms described in the previous section for hiding a number of states within the FSM such that only the designer knows about the traversal to/from those states could be used as a basis for a suite of other security protocols. For example, this state hiding can be used for remote authentication or identification by the designer, online integrity checking, and real-time monitoring, controlling, enabling, or disabling the chip [21]. An application of the method for trusted integration of multiple IP cores was demonstrated in [40].

An example structure for disabling the chip, in case of tamper detection, is the creation of *blackhole* states. The blackholes are states that cannot be exited regardless of the incident input sequence. Their design is very simple as shown in Fig. 4, where the blackhole states do not have a route back to the original reset state. A special case is the creation of trapdoor *grayhole* FSM. Grayhole FSMs are designed in such a way that only a long sequence of input signals (known by the passkey transition holder) can bring the control out of these states to the original functional states of the design. A grayhole construction is similar to a blackhole, with the exception of at least one-edge in the grayhole that can take the design back to its functional states.

VI. Automatic Synthesis and Implementation

In this section, we present the details of the implementation of secure BFSM construction that was introduced in Section V.

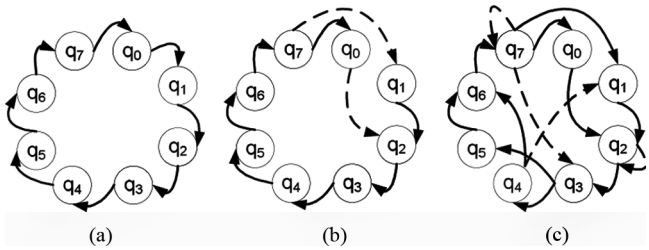

Fig. 5. Partition with $2^3 = 8$ states formed by random perturbations on a ring counter to store connectivity. (a) Ring counter. (b) Reconnecting a state. (c) Adding a few edges.

There is a need to devise a low overhead automatic construction for the high level specification. Since the conventional tools are not optimized for synthesizing a relatively large state-space, our solution for a low overhead implementation is careful selection of the BFSM topology by presynthesis and then performing *automatic iterative synthesis*. Our secure BFSM implementation method has three steps. The first step is BFSM graph topology construction described in Section VI-A. Second, transition (edge) passkey assignment is outlined in Section VI-B. Finally, Section VI-C presents the iterative synthesis method.

A. BFSM Graph Topology Construction

The BFSM inputs are termed *Primary Inputs (PI)* and its outputs are termed *Primary Outputs (PO)* since they are the same as the PI and PO to the original design. The output at the PO would be correct for an unlocked chip. This value would be incorrect for a locked IC with a very high probability.

The number of states is increased to ensure randomness and uniqueness of the startup state, as described in earlier equations. For a BFSM $= (\Sigma + \Sigma', \Delta, S + S', s_0, \delta + \delta', \lambda)$ with S' added states, our method first uses a partition approach. A partitioning method was introduced in [9], where the low overhead implementation of smaller partitioned FSMs are determined by presynthesizing and evaluating various random configurations of small FSMs (e.g., with 2^4 or 2^5 states). The partitions are then combined by randomly added edges to form the added state-space and transitions. We refer to the added edges and logic for connecting and mixing the original FSM with the new partitions by the term *glue logic*.

An example for a partition is shown in Fig. 5, where the steps for construction of a partition STG with eight states is demonstrated. A ring counter is used as the starting point, as shown in Fig. 5(a). Next, a few states are randomly selected and reconnected. Say on Fig. 5(b), the state q_1 is reconnected, such that still there will be a path from each state to every other state. Finally, a few random transitions are added to STG. In the example shown in Fig. 5(c), the states q_1 and q_4 are randomly reconnected, and the edges $\{q_4 \rightarrow q_1, q_7 \rightarrow q_3, q_2 \rightarrow q_2, q_7 \rightarrow q_7\}$ are randomly added. Therefore, the random perturbations on the ring counter are organized such that the graph connectivity is preserved. Note that our partitioning approach is more systematic and guided than [9] but it uses similar principles. The key difference is because our secure construction method constraints the number of directed edges incident to one added state to be t, such that t ensures the graph connectivity and multiplicity of keys.

In our implementation, connectivity in each group is ensured by construction, such that every state in the partitioned group of states is reachable from every other state. Next, transitions are added among the partitioned FSMs such that the reset state is reachable from all other states. For checking the connectivity of the added group of new states on a generic graph to the original reset state, we devise an algorithm as follows: For a state transition graph (STG) demonstrating the BFSM in graph format, form a corresponding graph STG$''$ whose vertices are a one-to-one mapping of the vertices in STG, but reverses the direction of the edges. This may be a cyclic graph. Next, use a spanning tree algorithm (e.g., Kruskal's algorithm [41]) to extract a directed acyclic graph (DAG) rooted at the initial reset state. If all nodes are reachable from the root node on this DAG (can be checked by a breadth first search (BFS) on the graph), then the connectivity condition is satisfied. Otherwise, more edges between the unreachable states and the reachable states are needed. For a graph with $|S|+|S'|$ vertices and E edges, the complexity of the Kruskal algorithm is $O(E \log\{|S| + |S'|\})$ and the complexity of the BFS is $O(|E| + |S| + |S'|)$. Since $\log\{|S| + |S'|\}$ is typically larger than $(1 + |S| + |S'|/E)$, the overall complexity is on the order of $O(E \log\{|S| + |S'|\})$.

Now, a natural question that may arise is that for traversing to the original reset state, one has to pass one of its neighbors so there could only be at most t distinct passkeys which do not share the states. As we mention in our attacks and countermeasures section, this attack would be effectively avoided as long as there are enough added states and the passkeys for some parts of the unlocking sequence are unique and hard to guess. Another important class of attacks that we discuss is the capturing and removal attacks.

B. Selecting and Computing the Transition Passkeys

Selecting strong passkeys is integral to the security of active hardware metering. Generally speaking, a random passkey is a vector of symbols of a certain length taken from a set of symbols by a random selection process. Each symbol must be equally likely to be selected. The strength of random passkeys depends on the entropy of the underlying random number generator. In our case, the selected passkeys can also serve a dual purpose: the designer can use them as a proof of ownership in addition to using them for unlocking.

For this reason, in order to generate the passkey (presynthesis in software), we use the hash value of the designer generated words that are signed by a private key (PrK) of a public key cryptographic (PKC) system. Now, others with access to the public key (PuK) of the same system can verify the ownership of the designer upon unlocking. However, an eavesdropper who can unlock the chip using a stolen BFSM structure would not be able to claim ownership.

The steps for selecting the transition passkeys for the edges on the added graph (E) are as follows. First, the IP rights owner selects E symbol strings and uses its own PrK to sign each of the strings. The strings must be long enough to be resilient against the brute force guessing attack. Next, a strong hash function (e.g., SHA-2) is applied to the signed strings so a fixed length digest message is obtained. Note that the edge transition passkeys

are independent of the state encodings performed by the synthesis software. Therefore, reading out the state values from the FFs would not help in revealing the incident edge passkeys. Note that since for each chip a new transition passkey has to be found as a path on the BFSM, this computation needs to be efficient. To provide efficiency, binary decision diagrams (BDDs) can be used to store the BFSM and computing the key pairs.

C. Iterative Automatic Synthesis

Once the BFSM is selected and the transition passkeys are determined, the structure has to be synthesized. To perform the synthesis, we first modify the original FSM adding extra control inputs. These inputs represent the transitions from the BFSM with the correct passkeys. Next we synthesize each partition of the BFSM with an output representing the incident edge transitions. After synthesizing each component separately, we connect the extra inputs of the original FSM to the outputs of the partitioned FSMs and resynthesize the whole system. This way, if the original design has K FFs and the BFSM has K'' FFs, then $|S| \leq 2^K$ and $|S'| = 2^{K''} - 2^K$ which implies $|S'|$ is much larger than $|S|$.

VII. Attacks and Countermeasures

In this section, we state the attacks identified on active hardware metering and discuss how the newly proposed method is secure against the proposed attacks.

1) *Brute force attack*. The adversary attempts at randomly generating inputs to randomly traverse from the power-up state until it reaches a state that was reached on the previously unlocked ICs, or it reaches the reset state.
   - — Under the construction of general-output multipoint function, probability of finding the correct passkey resulting in a valid edge transition is extremely low, and therefore, this class of attacks would not be able to break the security. Another effective countermeasure against the brute force attack is creation of blackhole states as discussed in Section V-C. All what is needed is to strategically place the blackhole states and their adjacent edges such that the probability of randomly entering them is high. To avoid the problem of starting up in one of the blackhole states, one can use the grayholes that can be exited by a long sequence of passkeys.
2) *BFSM reverse-engineering*. The adversary uses the states revealed by unlocking each chip to gradually build a BFSM model to enable finding the passkeys to unlock the new ICs.
   - — Our construction method is secure against this attack by the decomposition property (Section V-B) [39]. The way to ensure security is by finding paths that are at least not intersecting on one edge. As an example, for a 128-bit input, each passkey has 2^{128} possibilities, so even finding the passkey string for one edge is of exponential complexity.
3) *PUF removal/tampering attack*. The adversary removes/tampers the PUF on a locked IC and instead places a piece of SRAM that contains the PUF responses from a previously unlocked chip. Now, the passkeys for unlocking the previous chip can be used for unlocking a new IC.
   - — To protect against PUF removal, there are multiple measures that can be taken. One such measure is to use the time bound and single-cycle property of an authentic integrated PUF device (a memory look up takes more cycles) [27], [42]–[46]. The PUF signal timing can be designed to be an integrated part of the timing path of the main functional description. Therefore, removing PUF would affect the circuit timing. Recall that in our attack model, redoing the timing closure is as hard as designing a chip from scratch and is not a valid attack. Another measure is to add obfuscated states within the FSM for PUF checking. The self-check signals from this test states would verify the existence of the PUF during unlocking and while the chip is in operation by testing for randomness (by adding a randomness test modules) or by supplying challenges such that the PUF response can be continuously checked or its timing is taken into account [27], [42]–[46]. Also the prohibitive cost of mask/retiming should be considered when targeting removal of an integrated PUF that is on the design's timing path.
4) *State capture and replay attack(s)*. The adversary captures the states from a previously unlocked chip and would try to force a new chip to power-up in a similar state, or it forces the unlocked IC to start at the original reset state.
   - — To overcome this attack, in addition to the power-up states, also the traversal passkeys can be set to be a function of PUF. Now, the passkeys for one chip would not work on the next because of the uniqueness of PUF responses. All that is needed is to have a number of challenges (inputs) for the PUF and their associated ECC for the corresponding traversals. Since the responses from the PUF cannot be cloned on another chip, we have safeguarded our method against this attack. The PUF removal/tampering attack was already discussed.

VIII. Experimental Evaluations

In this section, we evaluate the new active hardware metering using both simulations and hardware implementation. We report the overhead in terms of area, power, and timing of the synthesized circuits from the ISCAS benchmark suite. The hardware overhead is evaluated on the H.264/MPEG-4 or Advance Video Coding (AVC) decoder circuit, synthesized and implemented on FPGA.

A. Evaluation Setup

The new hardware metering method (BFSM construction) is implemented by modifying the inputs to the ABC synthesis tool and by iterative calls to the tool. The BFSM modifications are done using a combination of Matlab and C programming languages. The application of the method for modifying the ISCAS

benchmarks in simulation is straightforward, directly following the algorithm described in Section VI. To generate the input transitions (as described in Section VI-B), we used the SHA-2 hashing algorithm.

For evaluating the H.264/MPEG-4 decoder on hardware, we used a combination of the Xilinx ISE synthesis package and our hardware metering implementation described above. The benchmark in our study was in Verilog hardware description language. To ensure operability with the ABC tool, we used the Altera Quartus package to translate the Verilog descriptions to the BLIF netlist format that is compatible with our BFSM construction methodology. Similar to our earlier simulations, the BFSM modifications are done in Matlab and C and the resynthesis is performed via iterative calls to the ABC tool. The final modified H.264/MPEG-4 decoder is then reconverted from the BLIF netlist format to Verilog description that is in turn synthesized on Xilinx FPGA by the ISE suite.

B. Benchmark Simulation Results

The FSM description is commonly used for realizing the control path of the circuitry. It is important to note that in modern designs, the control path is only a small part of the overall structure ($\ll$1%) [47]. Therefore, even doubling or tripling the control part would not have a significant impact on the overall design overhead in terms of area or power. The timing increase would impact the design delay though. As we will see in our evaluations, the timing overhead of our method is negligible and it could be even further suppressed by alternative synthesis methods.

The BFSM construction used in our simulations adds 20 flip flops to the original design. As we mentioned earlier, the number of new states ought to be large. Let us assume that K is the number of FFs in the original design corresponding to $S = 2^K$ states, and $K + 20$ is the number of FFs in the modified design corresponding to $S' = 2^{K+20} - 2^K$ states. It is clear that the $S' \gg S$ condition is satisfied. The length of the edge transition passkeys in our evaluations is set to 64 bits. An unlocking sequence on one chip would find a path of at least eight edges on the BFSM state transition graph. Therefore, the minimum length of a passkey for a chip is 512 bits. This number could be fixed, or could be any multiple of 64 bits.

The table in Fig. 6 demonstrates comprehensive performance overhead evaluations on the ISCAS benchmark suite. The first column denotes the benchmark circuit name (sorted by the benchmark number order). The next three columns (Columns 2–4) show the original design properties: the number of primary inputs, the number of primary outputs, and the number of flip flops postsynthesis. Columns 5–7 demonstrate the design area (in terms of the number of gates in the ABC tool) in the following order: the original postsynthesis area, the added area post BFSM synthesis after applying our method, and the ratio between the two former metrics (in %). The original design's power postsynthesis, the newly added power post-BFSM synthesis after applying the hardware metering, and the ratio between the two powers (in %) are reported in Columns 8–10. The postsynthesis delay of the original design, and the ratio between the added delay (post-BFSM construction and synthesis) to the original delay are shown in the last two columns, respectively.

Circuit	In	Out	FFs	Orig. Area	Added Area	OH (%)	Orig. Power	Added Power	OH (%)	Orig. Delay	OH (%)
s820	18	19	5	769	+1411	186	2773	+5924	213.6	28.2	0
s1196	14	14	18	1009	+1524	151	2558	+8223	321.4	35.8	3
s1238	14	14	18	1041	+1472	141	2709	+7153	264.0	34.4	1
s1423	17	5	74	1164	+1393	120	4883	+5564	114.0	92.4	0
s1488	8	19	6	1387	+1261	91	3859	+2804	72.7	38	1
s1494	8	19	6	1393	+1591	114	3913	+9632	246.1	38.4	2
s5378	35	49	164	4212	+1203	28.6	12459	+1874	15.0	32.2	3
s9234	36	39	211	7971	+1425	17.9	19386	+6312	32.6	75.8	0
s13207	31	121	669	11241	+1571	14	37844	+9334	24.7	85.6	1
s15850	14	87	597	13659	+1281	9.3	40003	+3404	8.5	116	0
s35932	35	320	1728	28269	+1383	4.9	122048	+5279	4.3	299.4	0
s38584	12	278	1452	32910	+1226	3.7	112707	+2357	2.1	94.2	2

Fig. 6. Metering overhead on the benchmark suite.

Let us start by analyzing the results for the area overhead. As can be seen in Column 6, the added area by the BFSM seems to be independent of the benchmark circuit size, with a standard deviation of 131 around its mean of 1395. Observe that the circuits on the top of the table are the smaller ones in the benchmark set, with a limited number of inputs and outputs and FFs, occupying a small area (in the original circuit synthesis). Given this observation, it is natural that the overhead (%) is much more significant on the smaller circuits compared to the larger ones. The mean of the percentage overhead in the area is about 73%, with a standard deviation of 67% indicating large fluctuations among the circuits. Looking at the bottom half of the table that includes the larger designs, the mean of the overhead (%) is set at a much lower 13%, with still a relatively large standard deviation of 9%. Our results show that for most industrial designs that are of larger complexity (making them worthwhile to protect), the area overhead for the BFSM construction is quite low, especially since the control path is a small part of the overall design.

Next, we analyze the reported results for the power (Columns 8–10). Note that the units for this report are the same units reported by the ABC synthesis tool. We see that the added power (Column 9) has a large variation that seems not to follow the benchmark size. It has a standard deviation of 2650 around its mean of 5655. For the small circuits in the suite that are shown in the upper half of the table, the overhead ratio is very large, with a mean of 205%, and standard deviation of $\sim$95%. The overhead ratio is much lower for the larger circuits in the benchmark set in the lower half of the table, with a mean of 14.5% and a standard deviation of 12%. Since the circuits in the benchmark set are small compared to the industrial strength circuitry in design and use today, it is safe to say that the power overhead of the metering method is low, in particular since the control path by itself is very small compared to the entire design.

Last but not least, we evaluate the impact of the BFSM construction methodology on the circuit timing (Columns 11–12). The unit for timing is the same unit reported by ABC. The ratio of the added critical path delay overhead compared to the original delay seems to be independent of the circuit size, with a mean of 1% and standard deviation of 1.15%. Therefore, we see that the overhead in the critical path delay introduced by our method is rather low.

	Original H.264	(+20 states) Key: 1024b	(+40 states) Key: 2048b	(+64 states) Key: 5120b
Number of Gates	381,176	407,068	429,075	457,574
Gate overhead	-	6.79%	12.56%	20.04%
Number of LUTs	26,485	29,996	33,160	37,106
LUT overhead	-	13.25%	25.19%	40.09%

Fig. 7. Metering overhead for H.264/MPEG4 on FPGA.

It is also worthwhile to compare the results reported in this paper to the hardware metering methodology introduced in [9]. In comparison, the secure BFSM construction method introduced in this work produces a visibly higher overhead. This is because the latest methodology introduced in this paper follows the theoretical guarantees described in the previous sections. The heuristic-only solutions offered in [9] could not provide as strong proof of security.

C. Hardware Implementation Overhead

The H.264/MPEG-4 Part 10 or Advanced Video Coding (AVC) is a video compression standard. It is presently one of the most used formats for recording, compressing, and distributing high-definition videos. It is widely used in Blu-Ray systems, youtube, iTunes, and other Internet video distribution centers. Designing efficient H.264/MPEG-4 is both a big challenge and an opportunity, especially with the move towards smart phones and other low-power portable systems with video decoding capabilities. We selected the H.264/MPEG-4 for our hardware implementation and evaluation purposes. The original design was written in the Verilog hardware description language. The tool flow for synthesizing this design was described earlier in this section. Note that we do not report the overhead of implementing PUFs on the FPGA in this paper. The overhead for implementing PUF on FPGA on Xilinx boards is readily available in the contemporary literature [19], [26], [27], [44], [48].

The table in Fig. 7 shows the design area after synthesis to the FPGA in terms of the number of equivalent gates and the number of occupied lookup tables (LUTs), along with the percentage overhead. The second column reports the number of equivalent gates and the number of LUTs for the original design. The third column shows the overhead for a BFSM with 20 new FFs and a key length of 1024. It can be seen that the percentage of added equivalent gates is about 6.7%, and the percentage of added LUTs is about 13%. We also experimented with two other cases with 40 and 64 added FFs, as shown on the fifth and sixth columns, with key lengths of 2048 and 5120, respectively. As we mentioned earlier, the key length is determined by the number of edge transitions required for unlocking and the passkey on each edge. The passkey on each edge is set to the number of inputs on the edge, $\alpha = 64$ bits. A key length of 1024 means that we require 16 edge transitions in our unlocking sequence.

We see that with the increase in the number of added FFs, the percentage overhead in terms of the number of equivalent gates and the number of LUTs increases linearly. Since adding to the number of FFs could yield exponentially stronger proofs for security, our protection method is relatively low overhead for very secure construction. As noted earlier, for MPEG4 decoding and many other date-intensive applications that are implemented in hardware for efficiency reasons, the bottleneck is in the data path optimization and not in the control path which is a much smaller part of the overall design. Note that the earlier work in hardware metering were only evaluated on benchmark simulations. To the best of our knowledge, this paper presented the first hardware implementation and overhead evaluation results for hardware metering.

IX. Conclusion

We have developed the first active hardware metering approach. The method uniquely locks each IC implementing the design using unique chip identifiers coming from a physical unclonable function (PUF). The designer (IP rights owner) who has access to the full state encoding of the design is the only entity who can provide the passkeys for unlocking the chip. The unique identifiers are integrated within the states of the design's FSM. The FSM is boosted—in a way that does not alter the functionality of the original design—to include many added states.

In this paper, the first set of provable security guarantees for active hardware metering was demonstrated. We devised a new construction for active hardware metering by modifying the behavioral description of the design in the FSM domain such that the modifications form an instance of a general-output multipoint function that was shown to be efficiently obfuscatable. The hardware synthesis that takes the behavioral-level specification and transforms it to a netlist must be an obfuscating compiler for the metering to be secure so the passkeys cannot be guessed or attacked. The theoretical results on obfuscating the family of point functions were exploited to ensure security of the new construction in the random oracle model. Automated synthesis methods for integration of the new secure metering construction was derived. We discussed the attacks and presented safeguards. The efficiency and practicality of the methods were demonstrated by experimental evaluations on sequential benchmarks and by proof-of-concept hardware metering implementation on a H.264 MPEG decoder on Xilinx Virtex-5 FPGA.

Acknowledgment

The author would like to thank Dr. G. Ghiaasi-Hafezi and A. Mirhoseini for help with reading and editing this paper.

References

[1] Defense Science Board (DSB) Study on High Performance Microchip Supply [Online]. Available: http://www.acq.osd.mil/dsb/reports/2005-02-hpms_report_final.pdf

[2] M. Pecht and S. Tiku, "Bogus! electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum*, vol. 43, no. 5, pp. 37–46, May 2006.

[3] S. Pope, "Trusted integrated circuit strategy," *IEEE Trans. Compon. Packag. Technol.*, vol. 31, no. 1, pp. 230–235, Mar. 2008.

[4] Managing the Risks of Counterfeiting in the Information Technology Industry. A White Paper by KPMG and the Alliance for Gray Market and Counterfeit Abatement (AGMA).

[5] Defense Industrial Base Assessment: Counterfeit Electronics, 2010. A Report by Bureau of Industry and Security's (BIS) Office of Technology Evaluation (OTE) [Online]. Available: http://www.agma-global.org

[6] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in *Proc. Int. Workshop on Information Hiding (IH)*, 2001, pp. 81–95.

[7] F. Koushanfar and G. Qu, "Hardware metering," in *Proc. Design Automation Conf. (DAC)*, 2001, pp. 490–493.
[8] F. Koushanfar and M. Potkonjak, "CAD-based security, cryptography, and digital rights management," in *Proc. Design Automation Conf. (DAC)*, 2007, pp. 268–269.
[9] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security Symp.*, 2007, pp. 291–306.
[10] J. Huang and J. Lach, "IC activation and user authentication for security-sensitive systems," in *Proc. Int. Symp. Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 76–80.
[11] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending piracy of integrated circuits," in *Design Automation and Test in Europe (DATE)*, 2008, pp. 1069–1074.
[12] J. Roy, F. Koushanfar, and I. Markov, "Protecting bus-based hardware ip by secret sharing," in *Proc. Design Automation Conf. (DAC)*, 2008, pp. 846–851.
[13] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: A nondestructive hidden characteristics extraction approach," in *Proc. Information Hiding Conf. (IH)*, 2008, pp. 102–117.
[14] F. Dabiri and M. Potkonjak, "Hardware aging-based software metering," in *Proc. Design, Automation & Test in Europe Conf. Exhibition (DATE)*, 2009, pp. 460–465.
[15] A. Wei, M. Nahapetian, and M. Potkonjak, "Robust passive hardware metering," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 2011, pp. 802–809.
[16] F. Koushanfar, "Hardware Metering: A Survey," in *Introduction to Hardware Security and Trust*. New York: Springer, 2011.
[17] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: An overview," in *Proc. Great Lakes Symp. VLSI (GLSVLSI)*, 2011, pp. 449–454.
[18] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. Conf. Computer and Communications Security (CCS)*, 2002, pp. 148–160.
[19] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. Design Automation Conf. (DAC)*, 2007, pp. 9–14.
[20] U. Rührmair, S. Devadas, and F. Koushanfar, "Security Based on Physical Unclonability and Disorder," in *Introduction to Hardware Security and Trust*. New York: Springer, 2011.
[21] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 2007, pp. 674–677.
[22] L. Yuan and G. Qu, "Information hiding in finite state machine," in *Proc. Information Hiding Conf. (IH)*, 2004, pp. 340–354.
[23] R. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 2008, pp. 674–677.
[24] C. Mouli and W. Carriker, "Future fab: How software is helping intel go nano-and beyond," *IEEE Spectrum*, vol. 44, no. 3, pp. 38–43, Mar. 2007.
[25] B. Santo, "Plans for next-gen chips imperiled," *IEEE Spectrum*, vol. 44, no. 8, pp. 12–14, Aug. 2007.
[26] M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan./Feb. 2010.
[27] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable pufs," *ACM Trans. Reconfig. Technol. Syst. (TRETS)*, vol. 2, no. 1, pp. 5:1–5:33, 2009.
[28] A. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1101–1117, Sep. 2001.
[29] K. Chatterjee and D. Das, "Semiconductor manufacturers' efforts to improve trust in the electronic part supply chain," *IEEE Trans. Compon. Packag. Technol.*, vol. 3, no. 3, pp. 547–549, Sep. 2007.
[30] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Trans. Embedded Comput. Syst. (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.
[31] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 1998, pp. 194–198.
[32] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Design*. Norwell, MA: Kluwer, 2003.
[33] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 10, no. 3, pp. 523–545, 2005.
[34] F. Koushanfar and Y. Alkabani, "Provably secure obfuscation of diverse watermarks for sequential circuits," in *Proc. Int. Symp. Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 42–47.
[35] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. Int. Test Conf. (ITC)*, 2008, pp. 1–10.
[36] S. E. Ahmed and R. J. McIntosh, "An asymptotic approximation for the birthday problem," *Crux Mathematicorum with Mathematical Mayhem*, vol. 26, no. 3, pp. 151–155, 2000.
[37] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (im)possibility of obfuscating programs," in *Proc. Int. Cryptology Conference (CRYPTO)*, 2001, pp. 1–18.
[38] S. Goldwasser and G. Rothblum, "On best-possible obfuscation," in *Proc. Theory Cryptography (TCC)*, 2007, pp. 194–213.
[39] B. Lynn, M. Prabhakaran, and A. Sahai, "Positive results and techniques for obfuscation," in *Proc. Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004, pp. 20–39.
[40] Y. Alkabani and F. Koushanfar, "Active control and digital rights management of integrated circuit IP cores," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2008, pp. 227–234.
[41] T. Corman, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
[42] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Proc. Information Hiding Conf. (IH)*, 2009, pp. 206–220.
[43] U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, "Towards electrical, integrated implementations of SIMPL systems," *Inf. Security Theory Practices*, vol. 6033, pp. 277–292, 2010.
[44] M. Majzoobi and F. Koushanfar, "Time-bounded authentication of FPGAS," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pt. 2, pp. 1123–1135, Sep. 2011.
[45] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public, physically unclonable functions: Architecture and applications," in *Proc. Design Automation Conf. (DAC)*, 2011, pp. 242–247.
[46] S. Meguerdichian and M. Potkonjak, "Matched public PUF: Ultra low energy security platform," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, 2011, pp. 45–50.
[47] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. San Mateo, CA: Morgan Kaufmann, 2006.
[48] S. Meguerdichian and M. Potkonjak, "Device aging-based physically unclonable functions," in *Proc. Design Automation Conf. (DAC)*, 2011, pp. 288–289.

Farinaz Koushanfar (S'99–M'06) received the Ph.D. degree in electrical engineering and computer science, and the M.A. degree in statistics, both from the University of California Berkeley, in 2005, and the M.S. degree in electrical engineering from the University of California Los Angeles.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering at Rice University, Houston, TX, where she directs the Texas Instruments DSP Leadership University Program. Her research interests include adaptive and low power embedded systems design, hardware security and trust, and design intellectual property protection.

Prof. Koushanfar is a recipient of the Presidential Early Career Award for Scientists and Engineers, the ACM SIGDA Outstanding New Faculty Award, the Office of Naval Research (ONR) Young Investigator Program Award, the Defense Advanced Project Research Agency (DARPA) Young Faculty Award, the National Science Foundation CAREER Award, MIT Technology Review TR-35, an Intel Open Collaborative Research fellowship, and a Best Paper Award at Mobicom.