RICE UNIVERSITY

N-variant Hardware Design

by

Yousra Alkabani

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

[illegible]

Keith D. Cooper, Chair
L. John & Ann H. Doerr Chair in
Computational Engineering; Professor of
Computer Science and Electrical and
Computer Engineering

[illegible]

Moshe Y. Vardi
Karen Ostrum George Professor in
Computer Engineering

[illegible]

Kartik Mohanram
Assistant Professor of Electrical and
Computer Engineering and Computer
Science

Bart Sinclair

Bart Sinclair
Associate Dean of Engineering; Lecturer
on Electrical and Computer Engineering

Houston, Texas
November, 2010

ADVISORS:

Keith D. Cooper

L. John & Ann H. Doerr Chair in Computational Engineering;

Professor of Computer Science and Electrical and Computer Engineering

Farinaz Koushanfar

Assistant Professor of Electrical and Computer Engineering;

Assistant Professor of Computer Science

ABSTRACT

N-variant Hardware Design

by

Yousra Alkabani

The emergence of lightweight embedded devices imposes stringent constraints on the area and power of the circuits used to construct them. Meanwhile, many of these embedded devices are used in applications that require diversity and flexibility to make them secure and adaptable to the fluctuating workload or variable fabric. While field programmable gate arrays (FPGAs) provide high flexibility, the use of application specific integrated circuits (ASICs) to implement such devices is more appealing because ASICs can currently provide an order of magnitude less area and better performance in terms of power and speed. My proposed research introduces the *N-variant hardware* design methodology that adds the sufficient flexibility needed by such devices while preserving the performance and area advantages of using ASICs.

The N-variant hardware design embeds different variants of the design control part on the same IC to provide diversity and flexibility. Because the control circuitry usually represents a small fraction of the whole circuit, using multiple versions of the control circuitry is expected to have a low overhead. The objective of my thesis is to formulate a method that provides the following advantages: (i) ease of integration in the current ASIC design flow, (ii) minimal impact on the performance and area of the ASIC design, and (iii) providing a wide range of applications for hardware security and tuning the performance of chips either statically (e.g., post-silicon optimization)

or dynamically (at runtime). This is achieved by adding diversity at two orthogonal levels: (i) state space diversity, and (ii) scheduling diversity. State space diversity expands the state space of the controller. Using state space diversity, we introduce an authentication mechanism and the first active hardware metering schemes. On the other hand, scheduling diversity is achieved by embedding different control schedules in the same design. The scheduling diversity can be spatial, temporal, or a hybrid of both methods. Spatial diversity is achieved by implementing multiple control schedules that use various parts of the chip at different rates. Temporal diversity provides variants of the controller that can operate at unequal speeds. A hybrid of both spatial and temporal diversities can also be implemented. Scheduling diversity is used to add the flexibility to tune the performance of the chip. An application of the thermal management of the chip is demonstrated using scheduling diversity. Experimental results show that the proposed method is easy to integrate in the current ASIC flow, has a wide range of applications, and incurs low overhead.

Acknowledgment

First I would like to thank my advisor Professor Keith Cooper for the incredible support, help, and advice. I would also like to thank my committee members: Professor Moshe Vardi, Professor Kartik Mohanram, and Professor Bart Sinclair for the great insights they have given me to improve my work.

Professor Joe Warren, the chair of the computer science department has provided me with great support as a student in the computer science department.

I would also like to thank all the professors who taught me classes at Rice university and at Ain Shams university. I would like to specifically thank Professor Luay Nakhleh for being a great teacher.

I would like to thank all the staff in the computer science department. Darnell Price has given me great support, and Bel Martinez has patiently helped me finish all my paperwork.

I also highly appreciate all the help provided to me and all the Rice international students by Dr. Adria Baker and the whole office of international students and scholars (OISS) staff.

I would like to thank my family and friends for their unconditional love and support. I would specially like to thank my mom and dad for giving me all the love, care, and tenderness I needed to keep going through tough times.

At last, I would like to thank the co-authors of my papers for their valuable input on my research during the past years.

Contents

Illustrations

Tables

Abreviations

- **AFSM:** Authentication Finite State Machine
- **ASIC:** Application Specific Integrated Circuit
- **BFS:** Breadth-First Search
- **BFSM:** Boosted Finite State Machine
- **BLIF:** Berkeley Logic Interchange Format
- **CA:** Certification Authority
- **CAR:** Capture And Replay
- **CDFG:** Control DataFlow Graph
- **CFSM:** Control Finite state Machine
- **DAG:** Direct Acyclic Graph
- **DFG:** DataFlow Graph
- **DRM:** Digital Rights Management
- **ECC:** Error Correction Code
- **FF:** FlipFlop
- **FM:** Flexibility Metric
- **FPGA:** Field Programmable Gate Array

- **FSM:** Finite State Machine
- **GD:** Global Diversity
- **HLS:** High-Level Synthesis
- **IC:** Integrated Circuit
- **IP:** Intellectual Property
- **LD:** Local Diversity
- **MV:** Manufacturing Variability
- **NMR:** N-Modular Redundancy
- **NVP:** N-Version Programming
- **NVS:** N-Version Software
- **NVX:** N-version eXecution
- **PKC:** Public Key Cryptography
- **PUF:** Physically Unclonable Function
- **RB:** Recovery Block
- **RFID:** Radio-Frequency IDentification
- **RTL:** Register Transfer Level
- **STG:** State Transition Graph

Chapter 1

Introduction

Lightweight embedded devices have stringent area and power constraints. Such devices need to be portable with long battery life. A designer can consider an ASIC or an FPGA implementation for such devices. The power and area constraints make an ASIC implementation of such devices the natural choice, especially when the mass production of these devices is to be considered. On the other hand, FPGAs can be economical when only a small quantity of the devices is to be manufactured.

Other issues that arise in the design and manufacture of these devices, however, complicate the decision. In many applications, the device would benefit from flexibility that is not available in ASICs derived from conventional designs. For example:

- After manufacturing, the designer might want to tweak the design to improve the performance of an individual chip.
- The designer might also prefer to provide different implementations of the same design that can operate at different points on the energy / performance curve.
- Functionality can be varied (for instance some features can be limited or disabled) depending on the customer who buys the chip.

While an FPGA implementation can easily achieve these goals, it is more difficult to address these points using an ASIC implementation without incurring large overheads. This thesis explores one option to address the problem: embed multiple implementations of the control circuitry in the same design. We call this technique *N-variant hardware design.*

This thesis explores the techniques required to implement N-variant design in a standard ASIC design flow, a necessity if these techniques are to have practical application. It demonstrates how to use N-variant design to achieve two distinct kinds of diversity: state space diversity and schedule diversity. This diversity is useful in its own right, as it allows an ASIC designer to achieve the goals that we laid out earlier. Variants of the control circuitry can be designed to have identical or slightly different functionalities depending on the target application. We can summarize the objectives as follows.

1. Using the standard ASIC flow to implement N-variants gives the N-variant methodology all the performance advantages of ASICs.

2. Embedding multiple variants of the control circuitry of the design on chip, either by manipulating the state space of the design or by manipulating the schedule of the design, can be used to make the chip more robust and resilient against security attacks.

3. The N-variant methodology manipulates the control part of the circuit. Since the control part represents a small fraction of the whole chip (about 1% [1]), the overhead due to the added redundancy is expected to be low.

Given the basic tools for N-Variant Design, we can attack more complex design problems. For example, we can use the tools of N-variant design to design ASICs that have chip-specific locks to protect the IC or to protect individual IP cores on the IC. These locks can be used to require a chip specific key that resets the chip's initial state to an operable state, or to check continuously for the presence of a key

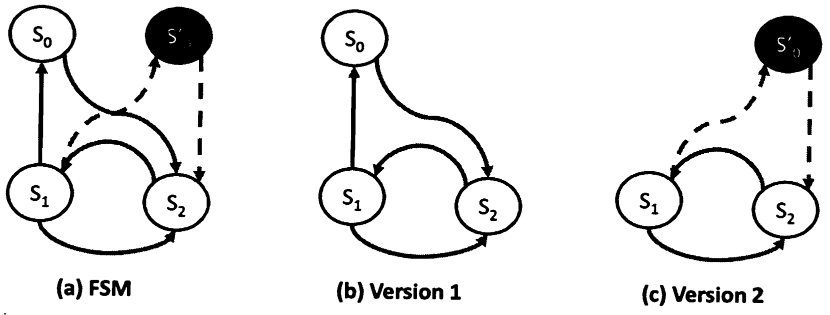


Figure 1.1 : An example of state space diversity.

and protect against unauthorized use. These locks exploit individual variation in the ASIC coupled with multiple variant control circuits and circuit obfuscation to create ASICs that require a unique key per manufactured chip to operate. The computation of the key requires both knowledge of the individual chip, obtained from specific test vectors, and knowledge of the obfuscated design, which the designer holds private.

1.1 Thesis

My thesis is that *by intelligently embedding different implementations of a part of the design in an IC, one can achieve the flexibility and diversity needed to protect an embedded device or tune its performance while maintaining low overhead and without significantly changing the current design flow.*

To support this thesis, we propose two different levels of adding diversity to a circuit: state space diversity, and scheduling diversity. State space diversity is achieved by adding states to the finite state machine (FSM) representing the control part of a system. Figure 1.1(a) shows an example of adding state space diversity to an FSM where we make a copy S'_0 (shown in black) of state S_0 and all the transitions to and from it (duplicated transitions are shown as dashed lines). Figures 1.1(b) and (c) show the two different versions of the FSM. In the first version we use states S_0,

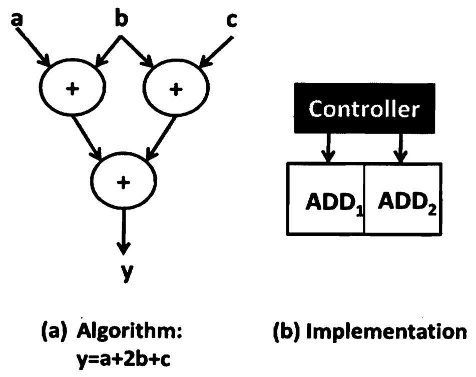


Figure 1.2 : An algorithm and its implementation in hardware.

S_1, and S_2, and in the second version we use S_0', S_1, and S_2. While both versions of the FSM are functionally equivalent, S_0 and S_0' are represented by different state encodings. Thus, if the designer provides means of checking the state encoding at runtime, and after the initial setting each customer (a chip used) is associated with a version, traversing the edge from S_1 to S_0 or S_0' can be used to verify the identity of the chip running the design.

Variant FSMs can have different state encodings [2], have different extra FSM parts added to them [3], or differ in a subset of the states and transitions [4]. We show the application of state space diversity to active hardware metering and IP rights protection in [2, 3, 4, 5].

Scheduling diversity is implemented by manipulating the scheduling and binding algorithms used for high-level synthesis of circuits. In high-level synthesis an algorithm is compiled into a hardware circuit by converting it first into a control dataflow graph (CDFG); then scheduling, allocating, and binding the CDFG, and finally, converting the schedule into an FSM representing the controller and a datapath containing the ALUs, memory and communication logic. Figure 1.2(a) shows an example of a CDFG for a simple algorithm that computes $y = a + 2b + c$. Fig-

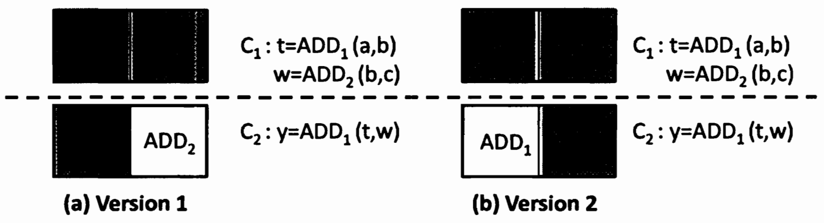


Figure 1.3 : An example of spatial diversity.

ure 1.2(b) shows the hardware implementation of the algorithm using two hardware adders ADD_1 and ADD_2 and a controller. We manipulate the controller to generate designs with different types of scheduling diversity: spatial and temporal.

Figure 1.3 shows how we can add spatial diversity to the implementation of the simple algorithm described above. The control steps are shown as C_1 and C_2. For the first version shown in Figure 1.3(a), the controller uses ADD_1 and ADD_2 to compute the intermediate results t and W respectively. Then it uses ADD_1 in the second control step to compute the final result y. This way ADD_2 is only used in half the control cycles in the first version. In the second version shown in Figure 1.3(b), the controller uses ADD_2 in the second control cycle to compute y. Thus, in the second version ADD_1 is used half the time and we have spatial diversity in the activities of different modules on the chip.

Temporal diversity can be implemented in a similar way as spatial diversity. However, for temporal diversity the controller is manipulated to run at different control steps to generate versions with different delays. Figure 1.4 shows an example for temporal diversity. In the first version shown in Figure 1.4(a), the controller executes the computation of y in two control steps. In the first step it computes t and w, then in the second step it computes the final result y. In the second version shown in Figure 1.4(b), the controller does the computation in three steps. It computes t in the

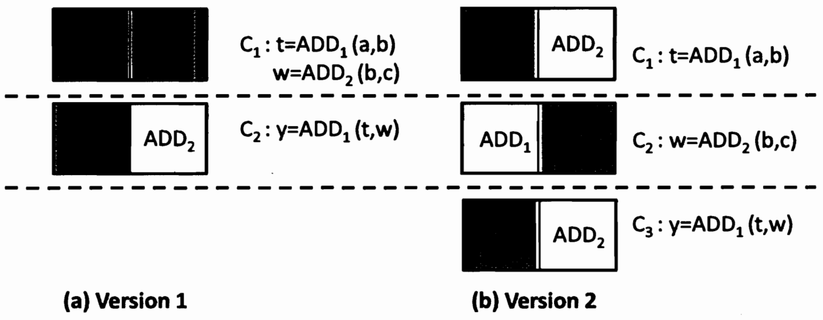


Figure 1.4 : An example of temporal diversity.

first step, w in the second step, and y in the third step. This way, the second version takes an extra control step to compute the result. Temporal diversity can be used in systems that require different deadlines for different tasks. For tasks that have a longer time to execute, one can slow down the system and thus reduce the power consumption by disabling the unused modules. For energy savings, the designer might use the same adder in all three cycles of an operation, thus avoiding any energy cost incurred by enabling and disabling the adders on different cycles. In this scheme, the designer might provide two control FSMs, one that used ADD_1 and the other that used ADD_2. For heat distribution, the design could cycle between them.

1.2 Organization

The rest of this thesis is organized as follows. The use of the N-variant methodology to add state space diversity is presented, along with potential applications, in Chapter 2. That chapter shows a method that automatically replicates the state of a design by duplicating a subset of the logic gates used to implement the design. This method can be applied automatically to any synthesized design that includes an FSM.

Chapters 3 and 4 present two different methods to implement state space diversity. In Chapter 3, we use replication to duplicate states in the controller of a design. The method adds a large number of extra stages to the design by adding extra loops to the original FSM. Manufacturing variability is used in conjunction with state replication to illustrate how to take advantage of state diversity in active hardware metering of ICs. Chapter 4 incorporates state diversity by duplicating a small subset of the states and edges in the design, with remote activation of ICs as an illustrative application.

Chapter 5 shows how to use state diversity to provide a data rights management platform for IP protection.

Chapter 6 illustrates the use of scheduling diversity to control peak temperature of an IC during operation. Finally, Chapter 7 concludes the thesis and suggests future work in this area.

Chapters 2 through 6 have already appeared in print. The content has been reformatted to fit Rice University thesis form, but has not otherwise been changed*. Table 1.1 shows the proper citations for those chapters, including the collaborators who are co-authors on those papers.

*Because Chapters 2 through 6 reproduce published work, they appear unchanged. A careful reader will note that the overall percentage improvements in those chapters are presented as the average of the improvements. In other areas of Computer Science, these numbers might be computed using the aggregate improvement or the geometric mean.

Table 1.1 : List of included published work.

Chapter	**Citation**
Chapter 2	Y. Alkabani and F. Koushanfar, N-variant IC design: methodology and applications, in Design Automation Conference (DAC), pp. 546-551, 2008.
Chapter 3	Y. Alkabani and F. Koushanfar, Active hardware metering for intellectual property protection and security, in USENIX Security Symposium, pp. 291-306, 2007.
Chapter 4	Y. Alkabani, F. Koushanfar, and M. Potkonjak, Remote activation of ics for piracy prevention and digital right management, in IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 674-677, 2007.
Chapter 5	Y. Alkabani and F. Koushanfar,Y. Alkabani and F. Koushanfar, Active control and digital rights management of integrated circuit IP cores, in ACM/IEEE International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES), pp. 227-234, 2008.
Chapter 6	Y. Alkabani, F. Koushanfar, and M. Potkonjak, N-version temperature-aware scheduling and binding, in International Symposium on Low Power Electronics and Designs (ISLPED), pp. 331-334, 2009.

Chapter 2

N-Variant IC Design: Methodology and Applications*

Abstract

We propose the first method for designing *N-variant* sequential circuits. The flexibility provided by the N-variants enables a number of important tasks, including IP protection, IP metering, security, design optimization, self-adaptation and fault-tolerance. The method is based on extending the finite state machine (FSM) of the design to include multiple variants of the same design specification. The state transitions are managed by added signals that may come from various triggers depending on the target application. We devise an algorithm for implementing the N-variant IC design. We discuss the necessary manipulations of the added signals that would facilitate the various tasks. The key advantage to integrating the heterogeneity in the functional specification of the design is that we can configure the variant during or post-manufacturing, but removal, extraction or deletion of the variants is not viable. Experimental results on benchmark circuits demonstrate that the method can be automatically and efficiently implemented. Because of its lightweight, N-variant design is particularly well-suited for securing embedded systems. As a proof-of-concept, we implement the N-variant method for protection of content of portable media players, e.g., iPod. We discuss how N-variant design methodology readily enables new digital

*This work is published in: Y. Alkabani and F. Koushanfar, N-variant IC design: methodology and applications, in Design Automation Conference (DAC), pp. 546-551, 2008.

rights management methods.

2.1 Introduction

N-variant design is the generation of $N \geq 2$ realizations of the same initial design description. The advantage of the technique is that it provides improved flexibility, robustness, attack resiliency, and design diversity. The strength and usefulness of N-variant designs was previously demonstrated for programs [6], virtual machines [7], and for achieving architectural heterogeneity [7]. While the method was initially intended for providing fault-tolerance, recent applications in security of computer systems, software and data has amplified its importance. Many attacks that take advantage of the specific and stationary nature of the underlying platform, may be eliminated by using N-variants [7, 6].

We propose the first methodology for designing N-variant ICs. The method works at the functional specification level, or by scripts that automatically perform pre-synthesis alterations. We construct a single hardware design consisting of multiple variants that are planned to have several exploitation sets. Note that it is also possible to do N-variant design post-synthesis. The drawback is that the design would become vulnerable to removal attacks. The advantage of pre-synthesis alternation is that all of the variants become an integral part of the functionality pertinent design, making the removal attack detrimental to the whole structure.

N-variant IC design has a number of important applications.

(i) Hardware IP protection and digital rights management: New semiconductor business models can be enabled by the N-variant design, e.g., different versions of one design can be sold to various vendors, enabling an automatic way to trace the ICs.

(ii) Usage and content metering: Alternation of the variants can be used for enumer-

ating the usage of IC components that run software/ media files, or in conjunction with unclonable chip IDs for metering the usage of an ICs.

(iii) Security: There is a need to prevent the exploits that target a homogeneous design from working on all the ICs [7]. Selection of different variants provides an effective countermeasure against the attacks.

(iv) Post-silicon optimization: Because of the manufacturing variability, for each IC, the designer can use the testing results to select the variant that has the best power/delay characteristics.

(v) Self-adaptation: Due to the impact of variability in operational conditions, e.g., aging, the IC can be designed such that it can adapt its structure over time.

(vi) fault-tolerance: The inherent redundancy provided by N-variants enables fault-tolerance.

Integration of N-variants at the functional level is done by altering the FSM and adding several states to it. By managing (controlling) the inputs to state-transitions, one would be able to select different variants of the design. The management inputs may come from various triggers, that depend on the target task for the N-variant design. We discuss in detail how the management may be altered for the various applications that we are considering.

The key advantage of using FSM is that it is not extractable from the synthesized design. Thus, even for a party who has access to the synthesized hardware IP, changing the FSM or extracting the original single-variant design would need an effort equivalent to redoing all the stages of design and implementation. For our purposes, the FSM can be safely assumed *inextricable.* Another important benefit of FSM is that certain aspects of FSM are inexpensively *verifiable* post-silicon. The inextricability and verifiability properties of the FSM were previously used for watermarking and for hiding information inside the design, and for remote activation and disabling [8, 9, 4]. As it was shown in the context of watermarking and IC activation/disabling, careful constraint manipulation and don't care planning can greatly reduce the overhead of FSM modifications. The new method is particularly well-suited for lightweight embedded systems applications. This is because the N-variant design enables lightweight mechanisms for protection and security of IP, software and content. The overhead of implementing traditional cryptographic protocols is huge, often overwhelming the constrained resources of an embedded system [10]. As a proof-of-concept, we demonstrate application of N-variant IC design for content usage metering of portable multimedia devices with an embedded MPEG compression module.

2.1.1 Motivational example

Figure 2.1 illustrates a motivational example, where the FSM of the design is shown by a state transition graph (STG) that has four states: s_0, s_1, s_2, and s_3. This FSM is the first variant. We replicate this design three times to have: s'_1, s'_2, s'_3, and s'_4 as the first copy (the second variant), s''_1, s''_2, s''_3, and s''_4 as the second copy (the third variant), and s'''_1, s'''_2, s'''_3, and s'''_4 as the third copy (the fourth variant). Thus, we construct a 4-variant circuit. The different copies can share FFs at the synthesis step to reduce the overhead. By careful state assignment, one can ensure that the FFs needed for each variant to function properly are different in at least one FF. This way, even if a FF is corrupted, there are still other copies that can function properly. In Figure 2.1, we show an example of how the FFs can be shared between the different variants. Assume that we implement the design using 4 FFs. The four FFs denoted by F_1, F_2, F_3, and F_4 are shown in front of each variant. The FFs in white affect the functionality of the variant, while the FFs in gray do not affect the functionality of the variant (although they can keep flipping all the time for obfuscation reasons depending on the application). For instance, the first variant can use F_1, and F_2 shown in white, while F_3, and F_4 do not affect its functionality. However, the second variant is affected by F_1 and F_4. Thus, if F_2 is corrupted, the first variant will not function properly, but the second will. For the circuit to properly function, it is enough to have one correctly functional variant selected. However, depending on the application, we can choose to switch between different variants, or to prefer a variant over the other based on their performances. Thus, the selection function shown in Figure 2.1 is application dependent as discussed in Section 2.4.

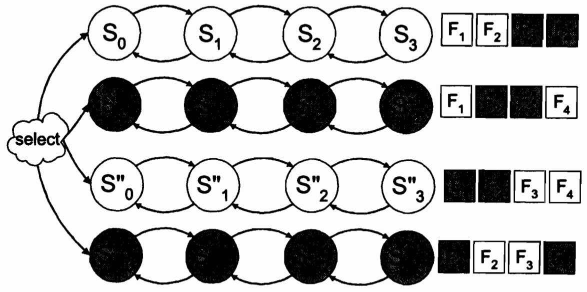


Figure 2.1 : A 4-variant Design

2.1.2 Paper organization

In the next section, we discuss the background and related work along the lines of FSM and the N-variant concept. In Section 2.3 we devise an algorithm for efficient integration of the N-variants. Section 2.4 demonstrates how the variant selection inputs can be managed to facilitate various applications. Experimental results evaluating the overhead of the method on standard benchmarks are presented in Section 6.6. The proof-of-concept implementation for metering multimedia files is also reported. Section 6.7 concludes the paper and outlines a number of future research directions.

2.2 Background

We describe the background and related literature in FSM and N-variant systems that has influenced and inspired our work.

2.2.1 FSM

A FSM is a dynamic discrete system with limited number of states that maps input sequences into output sequences. It can be used to represent a sequential function, e.g., sequential circuits. A FSM is typically defined by a 6-tuple M=(Σ,Δ,Q,q_0,δ,λ), where

- $\Sigma \neq \emptyset$ is a finite set of input alphabets;
- $\Delta \neq \emptyset$ is a finite set of output alphabets;
- Q=$\{q_0,q_1,...\}\neq \emptyset$ is a bounded set of states;
- $q_0 \subset Q$ is the set of initial states;
- δ (q, a) is transition function on input a and set $Q \times \Sigma \rightarrow Q$;
- λ (q, a) is output function for input a and set $Q \times \Sigma \rightarrow \Delta$;

The STG is used to represent the state transitions and input/output relations of the FSM; nodes correspond to states and edges define the input/output conditions for state transitions.

FSMs are used for information hiding and watermarking purposes. Oliveira proposes to alter the STG such that a signature is mapped to a spacial topological property in the sequence of states traversed by a sequence of inputs [8]. Yuan and Qu exploit the existence of redundant transitions and manipulate them to hide information inside the FSM without altering the minimized FSM [9]. Alkabani et al. use a combination of physically unclonable functions and state replication to uniquely lock each IC [4]. The lock is interleaved with combinational and sequential transitions and can be used for enabling/disabling of the IC. The unlocking is done by sequence of inputs that can only be generated by knowing the FSM structure.

The classic way for fault-tolerance FSM design is triple modular redundancy (TMR), where three copies of FSM are concurrently run and the correct output value is determined by voting [11]. Even though TMR may be viewed as a basic N-variant design method, we emphasize that the new N-variant methodology is not a simple replication of the FSMs. Instead, the variants are elegantly interwind within the original design. N-variant design has applications that TMR or other traditional fault-tolerant design methods do not support.

Our N-variant design algorithm is devised to have a low-overhead. A class of methods that is relevant to our work is state assignment and state minimization of large FSM networks [12, 13, 14, 15, 16]. The concepts and methods used in FSM minimization may help to even lower the overhead of N-variant designs in the future.

2.2.2 N-variant method

N-version software development was introduced as early as 1968 [17]. The goal was to achieve fault-tolerance [18, 19]. N-versioning collects N copies of the target software written by *different programmers* and then runs them in parallel, using the redundancy in the results to achieve fault-tolerance. N-versioning is different from N-variant since N-variant systems include multiple copies within the *same design* that appear different. So far, the main usage of N-variants has been for security purposes [7, 6].

The motivating idea is that most systems are homogeneous and the same exploit can attack all instances of the system. The specific machine-level characteristics of the attacked computer is used by the binary exploits, e.g, byte order, calling conventions, program load addresses, etc. If the system is not exactly the same, the exploit would not be able to adjust itself and the attack will be halted.

Achieving a heterogeneous computer system by using randomization was first introduced by Forrest et al [20]. Cox et al. present an architectural framework that systematically uses automated diversity that provides detection and disruption of a large class of attacks [6]. Holland et al. extended the idea of achieving heterogeneity by generating the architecture-dependent parts of kernel and standard C library from machine description [7]. Their approach successfully suppresses the risks of code injection attacks and state corruption attacks.

As we have mentioned in Section 2.1, N-variant IC design can enable many more tasks than just security and attack resiliency. Perhaps the most interesting applications are in the domain of IP protection and enabling new business models, and for post-silicon optimization and self-adaptations of the ICs. Furthermore, because the N-variants are embedded into the hardware, the overhead is smaller than embedding them into the virtual machine and/or the architecture. The low overhead of the N-variant IC renders it suitable for embedded systems applications.

2.3 N-variant IC design

In this section we discuss how the N-variant circuit can be designed. Figure 2.2 demonstrates the flow of our design. The implementation steps can be summarized as follows:

1. A single copy of the FSM is implemented as a group of FFs connected to the

input and output through combinational circuits $logic_1$ and $logic_2$ respectively, as shown in Figure 2.2(a).

2. Modify the FSM as shown in Figure 2.2(b) as follows:

   (a) Select the parameter m representing the number of replications of the FFs in the circuit.

   (b) Partition $logic_1$ into $logic_{1a}$ and $logic_{1b}$ and replicate $logic_{1a}$ for m times.

   (c) Partition $logic_2$ into $logic_{2a}$ and $logic_{2b}$ and replicate $logic_{1a}$ for m times.

   (d) Design the obfuscation logic and the selection logic to maintain the correct circuit's functionality. The obfuscation logic is implemented at the input side and is used to distinguish between different variants, and to generate dummy values in the unused FFs. The selection logic is used at the output side to select the output of the target variant. Note that when a variant is not selected it does not have to generate correct outputs.

Selection of the number of replications and the way the combinational circuits are partitioned yield the state-space of all possible implementations. Each choice requires a minimal complexity of the obfuscation and selection logics. It is worth noting here that the choice is also affected by the application targeted by the N-variant design method.

It is clear that the overhead of the implementation varies greatly with the choice of m, the partitioning of the combinational circuit, and the complexity of both the obfuscation and selection logic blocks. The area and power overheads of the methods are affected mainly by m, and by the partitioning of the combinational circuits. The delay is only affected by the selection and obfuscation logic. At one extreme, one can set $m = N$, $logic_{ia} = logic_i$, and $logic_{ib} = \emptyset$, while implementing the selection logic as

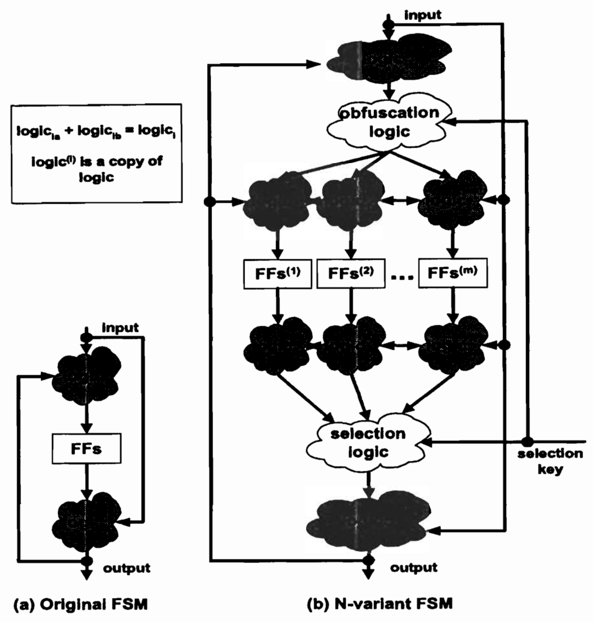


Figure 2.2 : Design of an N-variant circuit.

a multiplexer and discarding the obfuscation logic. This configuration produces the highest area and power overheads, and the lowest delay. At the other extreme, one can set $m = 1$, $logic_{ia} = \emptyset$, $logic_{ib} = logic_i$. This leads to an obfuscated circuit, and produces the lowest area and power overheads, and the highest delays. The potential impact of each of the possible design parameters is as follows.

• *FFs.* The FFs represent the memory of the FSM, they can be replicated m times where $m \geq 1$; $m = 1$ means the FFs are all shared and this removes a big part of redundancy in our implementation. However, the method can still have multiple variants by manipulating the logic blocks.

• $Logic_1$. This circuit component represents the combinational part of the circuit at the input side. It is divided into two parts: $logic_{1a}$ and $logic_{1b}$. $Logic_{1a}$ is to be replicated m times, and $logic_{1b}$ is shared. Increasing the size of $logic_{1a}$ leads to larger area and power overheads. However, it does not affect the delay overhead. Despite the introduction of more area and power overheads, it can be useful for applications with timing constraint.

• *Logic*$_2$. This circuit represents the combinational part of the circuit at the output of the FFs. It is divided into two parts: $logic_{2a}$ and $logic_{2b}$. $Logic_{2a}$ is to be replicated m times, and $logic_{2b}$ is shared. Just like the case in $logic_{1a}$, increasing the size of $logic_{2a}$ leads to larger area and power overheads and can be useful for applications targeting performance improvement and fault-tolerance.

• *Obfuscation logic.* The obfuscation is used to adjust the inputs for different variants. The more FFs are shared between variants, the more complex this part will be. However, in cases where few FFs are shared, this part is used to generate dummy values in the unused FFs. This can be useful for security applications.

• *Selection logic.* This block is responsible for ensuring that the correct output from the target variant is selected. In case of many FF replications, it can be simply implemented as a multiplexer. However, as the number of shared FFs increases, it becomes increasingly important to interleave this block with the obfuscation logic to guarantee that the correct outputs are obtained regardless of the selected variant.

The nature of the original circuit also affects the significance of the overhead. For instance, if the combinational logic is very small in area, power, and delay, even the slightest replications and the simplest obfuscation and selection logic will yield a significant overhead compared to the original one. Note that the area and power overhead of an FSM (representing the control part of a system) is extremely small compared to the overall area and power overhead of the system.

2.4 Application-specific management of the N-variants

In this section, we describe a number of management methods that can enable each of the applications described in Section 2.1. The variant selection contains the following components and structures: • *Trigger.* A trigger prompts selection of a new variant. It may come from various sources, including the clock signals, an internal counter, a sequence of inputs, or an external signal.

• *Storage.* The memory can be used in case selection of a variant is dictated by a stored key, or in case a counter should save its state in presence of rests. Also, many binary attacks may exploit a fixed memory address. To defend against this attack, all or some parts of the memory needs to by duplicated.

• *Driver.* The driver uploads the new variant in case a trigger is activated. A driver may be as simple as a multiplexer uploading the variant selection key from a stored location, or it may be a driver FSM, e.g., an obfuscated counter.

The above components can be used in various ways to manage the variant selection suited for a particular application. A few examples are as follows.

(i) Hardware IP protection and digital rights management. An IC vendor could configure its chips by using a fixed key specific to each customer. This provides a new mechanism to trace back the ICs in the supply chain. Also, many existing hardware IP protection methods can be readily used in N-variant settings. For example, the sequence of traversal among a few variants or within a variant could be utilized as a watermark or for hiding information [8, 9]. As another example, the N-variant structure could be integrated with the unique random number generated on each chip, for fingerprinting or locking the state transitions by the manufacturer [4].

(ii) Usage and content metering. The N-variants can be used to enforce licensing

agreement for the hardware, software, or content usage. Once the license agreement term is over, the IC would enter a nonfunctional state. The IP rights owner is the only entity that can load a new functional variant if a new license is obtained. A trigger by a clock or an internal counter can be used to save the number of uses or the time constraints. In case a counter is used, the states should be saved in an on-chip nonvolatile memory so the chip's reset would not affect the usage metering. Similar triggers were used for licensing of FPGA IPs [21], where it was noted that the trigger is vulnerable to memory reset/clock reset attacks. One possible countermeasure against this attack is to randomize the counter, so that resetting to zero would not traverse to the zero usage state. Many other counter obfuscation methods are possible. For example, one may initialize the counter by using a PUF, so different configurations will be needed depending on the unique IDs of each chip.

(iii) Security. The system could be made heterogeneous by frequently alternating the FSM variant [7]. For example, in case of exploits that use a fixed memory address, the memory can be duplicated creating a master and a slave copy. Every time a new variant is selected, the master and the slave copies would switch. The slave would copy its content from the master. If an attack exploits an address on the master copy, it would halt once the master copy is altered.

(iv) Post-silicon optimization. Because of the variability in CMOS technology, the variant with the best power or delay performance would be different on each IC. A spectrum of new post-manufacturing optimizations can be enabled by using the test data to determine the variant that has the best performance on each IC. For example, one may design the chips to have the maximum possible frequency, by selecting low feature sizes for the gates. Due to variability, some of the delays would be longer than planned, slowing down a few critical paths.

(v) Self-adaptation. The variants could be strategically uploaded using an offline or online monitoring method that manages the upload based on a certain target. For example, to be resilient against aging, one can equalize the amount of usage of the various components. Offline simulation of randomly selected variants can determine which components are used more in one variant. The variants with the largest difference in their usage patterns may be periodically loaded to equalize the aging effect.

(vi) fault-tolerance. The flexibility and redundancy of the N-variant design can be utilized for providing fault-tolerance. For example, if a FF is shown to be faulty during the test phase, the configurations that do not use this FF will be uploaded. The fault-tolerance can be implemented in more sophisticated ways, by adding a monitoring mechanism. As an example, a BIST structure can be included which periodically tests the circuit for faults due to aging, NBTI, or HCI effects. The correct variants could be managed accordingly.

2.5 Experimental results

In this section, we implement the N-variant method described in section 2.3 on benchmark designs and evaluate its performance. A program is written in C to generate N-variants for each design. The Berkeley SIS tool is used to simulate the results and to estimate the overhead. In the first subsection, we show the experimental results for applying the method on standard MCNC'91 benchmarks. In the second subsection, we demonstrate a DCT example as a proof-of-concept for applying the method to embedded multimedia applications. All the resulting circuits are mapped to the MCNC library.

2.5.1 Implementation of the N-variant method

Figure 2.3 shows an implementation of the N-variant method for $m = 2$. Our design targets embedding at least one redundant component for each part, while maintaining low area, power, and delay overheads. We present the results for $m = 2$ and $m = 4$ that translate to doubling and quadrupling the number of FFs respectively. The obfuscation logic is implemented as a block of XOR gates that place different values in the unused FFs for each variant. The selection logic is implemented as a multiplexer block. The number of variants generated by this implementation for a circuit with k FFs is 2^k for $m = 2$, and is 4^k for $m = 4$. Our selection of each parameter of the N-variant design (see Section 2.3) and the impact of the choice on the target applications are as follows:

- *FFs.* We selected $m = 2$ and $m = 4$. Selecting $m = 2$ ensures that the scheme has at least one spare FF for each variant; $m = 4$ provides more redundancy.
- *$Logic_1$.* We selected $Logic_{1a} = \emptyset$, and $logic_{1b}$=$logic_1$. The choices are made to reduce the area and power overheads. However, fault-tolerance and post-silicon optimization are now only dependent on FF redundancies.
- *$Logic_2$.* We devised the following parameters: $Logic_{2a} = \emptyset$, and $logic_{2b}$=$logic_2$. The selection is made to reduce the area and to reduce the delay and power overhead. Again, fault-tolerance and post-silicon optimization are only dependent on FFs.
- *Obfuscation logic.* For $m = 2$ and $m = 4$, only a half, or a quarter of the FFs are shared respectively. We select the obfuscation function to be a block of XOR gates that generate dummy values in the unused FFs. We limit the maximum number of variants to be 2^k (for $m = 2$) and 4^k (for $m = 4$) to keep the function simple and to constrain the delay overhead. Note that some of the variants can be reserved as trap

non-functional FSMs for security and IP protection applications.

- *Selection logic.* This part is implemented as a multiplexer and is kept simple to avoid large delay overheads.

Note that the parameter choices used in one implementation are driven by the designer's target application. The real results may be affected by many factors such as the original shape of the circuit, the gate library used, and different optimizations done on the circuit after modification.

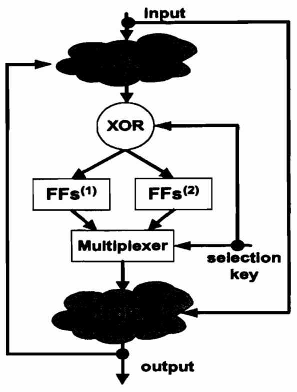


Figure 2.3 : Implementation choice for the experiments.

Tables 6.3 and 2.2 show the respective area, power, and delay overheads for $m = 2$ and $m = 4$. The evaluations on the used benchmarks show a maximum area and power overheads of 20% and 19% respectively, while the delay overhead was at most 17% for $m = 2$. The area and power overhead for $m = 4$ goes up to 48% and 62%, and the maximum delay overhead is 21%. It should be noted that the benchmarks with the highest overhead are very small. It is also interesting to note that sometimes the overhead is negative (the circuit is improved) because the resulting circuit maps to fewer gates in the MCNC library. The average area overhead is -0.65% and 12% for $m = 2$ and $m = 4$. The average power overhead increased from -3% for $m = 2$ to 15%

when $m = 4$. As expected the average delay overhead did not change by increasing m; it remained 10% in both cases for the evaluated benchmarks. All modifications to the circuits are done on the netlist before mapping. The relative values are more important than the absolute values as the values in the library can be scaled down.

It is worth noting here that the delay overhead is the only important metric in real designs. This is because the FSM which constitutes the control part of the design is typically only a small part of the design, significantly less than 1% of the power/area [1]. Thus, even if the FSM's area or power is doubled, it will not significantly affect the overall design.

2.5.2 Embedded multimedia application

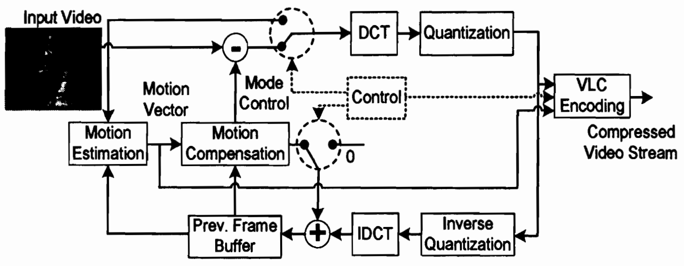


Figure 2.4 : Flow of MPEG video compression flow.

We report the results of implementing the N-variant method for content usage metering for embedded multimedia applications. The key observation that facilitates ultra-low overhead implementation is that the N-variants do not need to be instrumented in all parts of the design; only the critical parts of the design can be strategically selected and augmented. In our multimedia example, we implement N-

	Original				m=2		m=4	
BM	PI	PO	FFs	Area	Area	%	Area	%
dk16	2	3	5	461	482	4.6	535	16.1
keyb	7	2	5	461	490	6.3	535	16.1
planet	7	19	6	887	914	3.0	975	9.9
planet1	7	19	6	887	914	3.0	975	9.9
pma	8	8	5	346	382	10.4	419	21.1
s1488	8	19	6	880	915	4.0	973	10.6
s208	11	2	5	148	178	20.3	219	48.0
s420	19	2	5	148	151	2.0	191	29.1
s820	18	19	5	429	256	-40.3	296	-31.0
s832	18	19	5	427	271	-36.5	316	-26.0
styr	9	10	5	633	662	4.6	716	13.1
tma	7	6	5	287	318	10.8	362	26.1

Table 2.1 : Area overhead of the N-variants implementation.

variants in the DCT part of the design, the essential component for audio, image, and video signal processing. Figure 2.4 shows a typical MPEG video compression flow which contains both DCT and IDCT components.

Table 2.3 shows the results for applying the implementation described above to the DCT control circuit. The original DCT circuit has 9 FFs. We evaluated N-variant implementations with 2^9-variants ($m = 2$), 2^{18}-variants ($m = 4$), and 2^{36}-variants ($m = 8$). The area, power, and delay overheads for $m = 2$ are 6%, 5% and 15.5%, respectively. Changing m and keeping the obfuscation and selection logic the

	Original		m=2				m=4			
BM	Power	Delay	Power	%	Delay	%	Power	%	Delay	%
dk16	1649.8	97.7	1712	3.8	106.6	9.1	1935.1	17.3	107.6	10.1
keyb	1546.5	61.5	1635.5	5.8	67.6	9.9	1837.3	18.8	68.4	11.2
planet	3103.5	187.9	3156	1.7	219.4	16.8	3471.1	11.8	195.6	4.1
planet1	3103.5	187.9	3156	1.7	219.4	16.8	3471.1	11.8	195.6	4.1
pma	1250.3	61	1309.8	4.8	67.9	11.3	1550.4	24.0	73.4	20.3
s1488	3007.3	134.9	3088.7	2.7	152.7	13.2	3378.3	12.3	144.8	7.3
s208	480.4	25.4	573.6	19.4	26.9	5.9	779.1	62.2	30.9	21.7
s420	480.4	25.4	462.9	-3.6	26.9	5.9	667.1	38.9	30.5	20.1
s820	1423.9	33.7	790.7	-44.5	36	6.8	1007.9	-29.2	31.8	-5.6
s832	1445.1	33.6	855.4	-40.8	36.7	9.2	1089.9	-24.6	36.8	9.5
styr	2170.2	128.2	2272	4.7	144.9	13.0	2470.6	13.8	134.9	5.2
tma	989.3	64.8	1062.1	7.4	71.1	9.7	1287.9	30.2	72.7	12.2

Table 2.2 : Power and delay overheads of the N-variant implementation.

same, has a significant impact on the area and power overheads. For $m = 4$ the area and power overheads increase to 16% and 18%, and for $m = 8$ the area and power overheads reach 36% and 42%. However, the change in the delay overhead is insignificant.

2.6 Concluding remarks and future directions

We introduce N-variant design methodology, a novel design method that creates $N \geq 2$ copies of the same design. The strength and usefulness of N-variant design was previously shown for systems, software, and architectural components. However, no

	Orig.	$m = 2$	%	$m = 4$	%	$m = 8$	%
Area	821	871	6.1	955	16.3	1117	36.1
Power	2946	3084	4.7	3490	18.5	4198	42.5
Delay	123.8	143	15.5	140	13.1	139.7	12.8

Table 2.3 : The overhead for 2^9-variants ($m = 2$), 2^{18}-variants ($m = 4$), and 2^{36}-variants ($m = 8$) DCT.

scheme for generation of N-variant designs was introduced, beyond simple replication of the whole structure for fault-tolerance. We devise an algorithm for implementing the N-variant design that works by extending the FSM such that it would include multiple variants of the same design. We demonstrate how the scheme could be utilized to enable a number of important applications, including hardware and content IP protection, IP metering, security, design optimization, self-adaptation and fault-tolerance. We also discuss how the input signals could be managed for enabling each of the target applications. The experimental results on benchmark circuits demonstrate the efficiency of the proposed algorithm in terms of area, power and delay. We also present a proof-of-concept implementation of the method for N-veriant design of MPEG decoders by only manipulating one of its circuit component, namely DCT. Experimental evaluations shows the suitability of the method for embedded systems applications.

The N-variant design methodology has a great potential to impact various applications and lays the ground for much more research. Potential future research directions include implementation of the method on real ICs, careful development of the digital rights management methods that exploit the N-variants as the underlying mechanisms, finding more efficient methods for implementing the N-variants, devising on-chip monitoring mechanisms so that the N-variants could be automatically used for self-adaptation method, and design and implementation of post-silicon optimization scheme that exploit the multiplicity of the variants. Last but not least, better identification and classification of possible attacks against the introduced security and protection mechanisms, particularly those for monitoring the vulnerability of the system against the binary attacks must be performed.

Chapter 3

Active Hardware Metering for Intellectual Property Protection and Security*

Abstract

We introduce the first *active hardware metering* scheme that aims to protect integrated circuits (IC) intellectual property (IP) against piracy and runtime tampering. The novel metering method simultaneously employs *inherent unclonable variability* in modern manufacturing technology, and *functionality preserving alternations* of the structural IC specifications. Active metering works by enabling the designers to lock each IC and to remotely disable it. The objectives are realized by adding new states and transitions to the original finite state machine (FSM) to create boosted finite state machines(BFSM) of the pertinent design. A unique and unpredictable ID generated by an IC is utilized to place an BFSM into the *power-up state* upon activation. The designer, knowing the transition table, is the only one who can generate input sequences required to bring the BFSM into the *functional initial (reset) state.* To facilitate remote disabling of ICs, *black hole* states are integrated within the BFSM.

We introduce nine types of *potential attacks* against the proposed active metering method. We further describe a number of counter measures that must be taken to preserve the security of active metering against the potential attacks. The implemen-

*This work is published in: Y. Alkabani and F. Koushanfar, Active hardware metering for intellectual property protection and security, in USENIX Security Symposium, pp. 291-306, 2007.

tation details of the method with the objectives of being low-overhead, unclonable, obfuscated, stable, while having a diverse set of keys is presented. The active metering method was implemented, synthesized and mapped on the standard benchmark circuits. Experimental evaluations illustrate that the method has a low-overhead in terms of power, delay, and area, while it is extremely resilient against the considered attacks.

3.1 Introduction

In the dominant horizontal semiconductor business model, piracy (illegal copying) and tampering of hardware are omnipresent. In the horizontal business model, hardware IP[†] designed by the leading edge designers are mostly manufactured in untrusted offshore countries with lower labor and operational cost. This places the designers in an unusual *asymmetric* relationship: the designed IP is transparent to the manufacturers, but the fabrication process, quantity and added circuitry to the manufactured integrated circuits (ICs) by the foundry are clandestine to the designers and IP providers.

The security threat, financial loss and economic impacts of hardware piracy which have received far less attention compared to software, is even more dramatic than software [22, 23]. Software piracy has received more attention compared to hardware also because it requires low-cost resources that are available to the general public. Protection of hardware is also crucially important because the ICs are pervasively used in almost all electronic devices and the potentially adversarial fabrication house has the full control over the hardware resources being manufactured. It is estimated

[†]In this paper, the term IP is used to refer to the integrated circuits design specifications that is available to the fabrication house.

that the computer hardware, computer peripherals, and embedded systems are the dominant pirated IP components [22].

Several other issues make the IC protection problems truly challenging: (i) very little is known about the current and potential IC tampering attacks; (ii) numerous attacking strategies exist, since tampering can be conducted at many levels of abstraction of the synthesis process; (iii) the most likely hardware adversaries are financially strong foundries and foreign governments with large economic resources and technological expertise; (iv) the adversary has full access to the structural specification of the design and most often also to the manufacturing test vectors; (v) the internal part of manufactured ICs are intrinsically opaque. While it is possible to tomographically scan an IC, the dense metal interconnect in 8 or more layers of modern manufacturing technology greatly reduce the effectiveness of such expensive inspections.

IC *metering* is a set of security protocols that enable the design house to gain post-fabrication control by passive or active count of the produced ICs, their properties and use, or by remote runtime disabling.

Our strategic goal is the *development, implementation, and quantitative evaluation of symmetric mechanisms and protocols for hardware protection procured by untrusted synthesis, manufacturing, and/or testing facilities.* The term *symmetric* emphasizes that both the designers and the foundry will be protected by the new methods. The symmetry is warranted by the unique variabilities and the key exchange mechanism that is based on the agreement of both parties for unlocking each IC.

Hardware metering is important from both commercial and military point of views. For example, without metering, a foundry can produce numerous copies of one design without paying royalties, or, as another example, the sensitive defense designs may become available to adversaries. The *passive hardware metering* schemes work by giving a unique ID to each chip [24, 25, 26]. The *first ever active hardware metering* method introduced in this paper, provides not just mechanisms for detection of illegal copies, but more importantly, ensures that no manufactured IC can be used without the explicit consent of the designer.

The proposed methods employ two generic security mechanisms: *(1) uniqueness of each IC due to manufacturing variability; and (2) structural manipulation of the design specification while preserving behavioral specification.* While the first mechanism has been already proposed and used for unique IC identification, the second is novel. Even more novel is the integration of two mechanisms, a task that requires a great deal of creativity and formation of solutions to a spectrum of challenging technology, synthesis and optimization problems, with a greater impact than the sum of the powers of the individual techniques.

The integration to the functionality is performed by interwinding the unique unclonable IDs for each chip into the FSM of the design. The integrated control part is denoted by BFSM, and is built by adding new states and transitions to the original FSM, while preserving the original functionality of the circuit. To bring the BFSM into the functional initial (reset) state, knowledge of the transition table is required. Since the designer is the only one who knows this information, no one else can generate a key with a finite amount of resources to unlock the IC. Using a combination of BFSM and newly added black hole states, remote disabling of the ICs can be made possible. We outline several possible attacks against the introduced active hardware metering method and provide mechanisms that neutralize the impact of those attacks. For example, we show how addition of the black hole states disable the random guessing attacks.

The remainder of the paper is as follows. After describing the background, flow and the state-of-the-art in the next two sections, we represent the active metering method in Section 5.4. In Section 5.5, we show a low-overhead implementation and obfuscation of active metering. Section 5.6 introduces potential attacks and the counter measures that needs to be taken to be resilient against the attacks. We present experimental evaluation of the prototype implementation on several standard design benchmarks in Section 3.7. We outline a number of potential applications in Section 5.8 and conclude in Section 6.7.

3.2 Preliminaries

In this section, we describe the necessary background required for understanding the active hardware metering approach. The aim is to make the paper self-contained for the readers who are not familiar with the hardware design and synthesis process. Next, we describe the global flow of the active hardware metering approach.

3.2.1 Background

Manufacturing variability (MV). The intense industrial miniaturization of CMOS devices has been driven by the quest for increasing computational speed and device density, while lowering cost-per-function, as predicted by Moore's law. CMOS variations result in high variability in the delay and the currents of the VLSI circuits. The variations might be temporal or spatial. The temporal variations may occur across nanoseconds to years [27]. Spatial variation is due to lateral and vertical differences from intended polygon dimensions and film thicknesses . Spatial variation may be intra-die, or inter-die [28]. Aside from device variations, the circuit response and its variability are correlated with circuit topology. We will utilize the spatial variations in our benefit, while we address the problem of alleviating temporal variability. Bernstein et al. provide a classification of device variations (beyond 65nm) [29].

Design descriptions. We consider the case in which the sequential design in question represents a fully synchronous flow and that the description of its functionality from an input/output (I/O) perspective is publicly available. We assume that the functionality is fully fixed, in that the I/O behavior is fully specified. Therefore, we utilize unique unclonable identification to embed a distinct mark in the functionality of each IC, without altering the functionality in terms of the normal I/O behavior of the circuit. Our technique is applicable to the case where the piece of IP is available in structural HDL description, or in form of a netlist that may or may not be technology dependent. The description uniquely defines the sequential circuit's behavior and the state transition graph (see the next subsection) of the design.

During the design flow, the user will take such a description and if required, will map it to a specific technology. Typically, logic level optimizations such as retiming are performed at this stage. Most often, the circuit is used as a part of a more complex design.

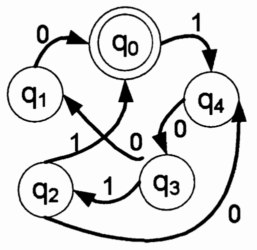


Figure 3.1 : Example of a STG with five states.The inputs required for state-to-state transition are shown next to the edges.

Finite state machine (FSM). FSM is a discrete dynamical structure that translates sequences of input vectors into sequences of output vectors. FSM can represent any regular sequential function. It appears in different forms, e.g. case statements in VHDL and Verilog HDL. The FSM is defined by the tuple M=(Σ,Δ,Q,q_0,δ,λ), where $\Sigma \neq 0$ and $\Delta \neq 0$ are a finite set of inputs and outputs symbols respectively; Q=$\{q_0,q_1,...\}\neq 0$ is a finite set of states while q_0 is the "reset" state; and the transition function is denoted as $\delta(q,a)$ on the input a and the set $Q \times \Sigma \rightarrow Q$, while the output function is denoted as $\lambda\ (q,a)$ on the set $Q \times \Sigma \rightarrow \Delta$.

To represent the state transitions and output functions of the FSM, we use the state transition graph (STG), with nodes corresponding to states and edges defining the input/output conditions for a state-to-state transition. An example STG is shown in Figure 3.1, where there are five states $\{q_0,\ q_1,\ q_2,\ q_3,\ q_4\}$, q_0 is the reset state, and there is a one-bit input controlling the state-to-state transitions. In the remainder of the paper, we use the terms STG and FSM interchangeably to refer to the control part of the design.

3.2.2 Global flow

As a motivational example for our problem, consider the scenario in which a given hardware intellectual property (IP) that belongs to its legitimate owner (Alice) is made available to a fabrication house (Bob). Alice pays for and demands N_A ICs implementing its design. Bob, utilizes the IP description to construct a mask that implements the design. Bob employs the mask to make $N_A + N_B$ copies of the design, where the illegal N_B copies do not encounter much additional cost due to the availability of the mask. Bob may sell the N_B illegal copies and make a lot of profit with negligible additional overhead.

The novel active metering helps Alice to protect her design against piracy by manipulating the STG of the original design, with the objective of creating a locked

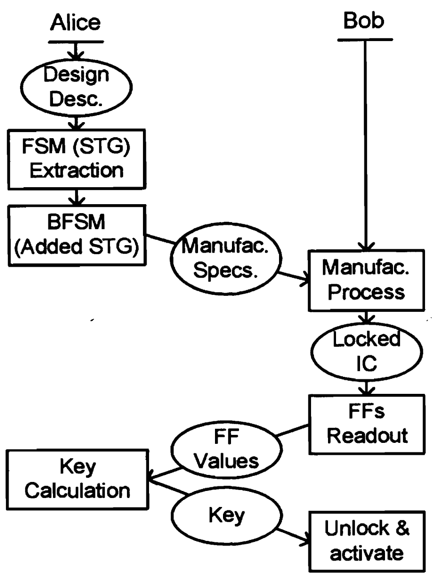


Figure 3.2 : The global flow of the active hardware metering approach.

state, that is unique for each of the ICs manufactured from the design with a very high probability. Upon manufacturing by Bob, each device will be uniquely locked (i.e., rendered non-functional), unless Alice is contacted by Bob to provide the particular key to unlock the IC. The scheme gives the full control over the manufactured parts and operational devices from the IP to Alice.

The global flow of the active hardware metering method is shown in Figure 6.1. We now describe the figure step by step. Alice takes the high level design description and synthesizes it to get the FSM of the design. Next, she constructs the BFSM by adding extra states. After that, she sends the detailed manufacturable design specifications to Bob who makes the mask and manufactures multiple ICs implementing the design. The manufactured ICs are locked (nonfunctional) at this stage. For each IC, Bob reads out the values in its flip flops (FFs) and sends the values to Alice. FF values can be read nondestructively, and the values are unique for each IC. Alice, knowing the BFSM structure, computes a specific key that can be used as input to that IC for unlocking it. The key is then sent back to Bob who utilizes it to activate the IC.

3.3 Related work

We survey the related literature that has influenced and inspired this work along four main lines of research: variability-based ID generation, authentication and security by variability-based IDs, intellectual property protection of VLSI designs, and invasive and noninvasive hardware attacks.

A number of authors have proposed and implemented the idea of addition of circuitry that exploits manufacturing variability to generate unique random sequence (ID) for each chip with the same mask [24, 26, 30]. The IDs are unclonable and separated from the functionality and do not provide a measure of trust, as they are easy to tamper and remove. Loftstrom et al. proposed a method for mismatching the devices based on changing the threshold of the circuits by placing the impurity of random dopant atoms [24]. Maeda et al. proposed implementing the random IDs on poly-crystalline silicon thin film transistors [26]. The drawback of the two described approaches is that they both need specialized process technology, and are easily detectable. Very recently, Su et al. have proposed a technique to generate random IDs by using the threshold mismatches of two NOR gates that are positively feeding back each other [30]. We will exploit their technique for the random ID generation.

A team of researchers has explored the idea of using variability-induced delays for authentication and security [31, 32, 33]. They use Physically Unclonable Functions (PUFs) that map a set of challenges to a set of responses, based on an intractably complex physical system. PUFs are unique, since process variations cause significant delay differences among ICs coming from the same mask. For each IC, a database of challenge-response sets is needed. Authentication occurs when the IC correctly finds the output of one or more challenge inputs. PUF-based methods solely utilize manufacturing variability as their security mechanism. *In contrast, our proposed methods introduce a paradigm shift in hardware security by adding new strong mechanisms: integration into circuit functionality at the behavioral synthesis level. Further more, even though the active metering methods can be utilized for authentication, its main target is addressing the hardware piracy problem.*

Koushanfar et al. have introduced the first hardware metering scheme that gives unique IDs to each IC [34]. The scheme was to make a small part of the design programmable so that one could upload different control paths post fabrication. They further described how to generate numerous different instances of the same control path with the same hardware [25]. They have also provided probabilistic proofs for the number of identical copies and probability of fraud for the proposed metering schemes [34, 25]. All metering schemes were passive. Indeed, no active metering scheme has been proposed to date. The prior work in trusted IC domain also includes introduction of several *watermarking* schemes that integrate watermarks to the functionality of the design at the behavioral synthesis level [35, 36, 37, 8, 38, 39, 40, 41]. *Watermarking is a fundamentally different problem when compared to metering. It addresses the problem of uniquely identifying each IP and not identifying each IC, so the existence of the same mask does not affect the watermarking results.* Fingerprinting for unique identification of programmable platforms has been proposed [42], but the techniques are not applicable to application specific designs (ASICs) due to the existence of a unique mask. Qu and Potkonjak provide a comprehensive survey of the watermarking, fingerprinting and other hardware intellectual property protection methods [38].

Even though many strong cryptographical techniques are available in hardware and software, their attack resiliency has been only verified by classical crypto-analysis methods. A class of attacks that is very challenging to address consists of physical techniques. Physical attacks take advantage of implementation-specific characteristics of cryptographical devices to recover the secret parameters. Koeune and Standaert provide a tutorial on physical security and side-channel effects [43]. The physical attacks are divided into invasive and noninvasive [44]. Invasive attacks depackage the chip to get direct access to its inside, e.g., probing. Noninvasive attacks rely on outside measurements, e.g., from the pins or by X-raying the chip, without physically tampering it.

There are multiple ways to attack an IC, including probing, fault injection, timing, power analysis, and electromagnetic analysis. Invasive attacks are typically more expensive than the noninvasive ones, since they need individual probing of each IC. Note that, according to the well-established taxonomy of physical attacks, attacks by the funded organizations (e.g., foundries) are the most severe ones, since they have both the funding and technology resources [45, 46, 44].

3.4 Active hardware metering

In this section, we present the details of the active hardware metering approach. Active metering is *integrated into the standard synthesis flow*, and is *low overhead*, *generalizable*, and *resilient against attacks.* By generalizable, we mean that the lock can be implemented on structures that are common to all designs. By attack-resiliency, we mean the cryptographic notion of a lock: that an attacker that does not have infinite computational power should not be able to unlock the IC without the knowledge of a key. To be generalizable, the method proposed here aims at protecting the design by boosting the design's FSM (and creating a BFSM) common to the widely used class of sequential designs. In this section, we describe the BFSM construction and introduce the locking mechanism. Implementation details are discussed in the next section.

3.4.1 Method

Random Unique Block (RUB). Perhaps the most important component of the proposed security mechanism is the existence of the unclonable unique ID for each IC. The IDs are a function of the variability present at each chip and are therefore, specific to the chip. RUB is a small circuitry added to the design, whose function is to generate the unique ID. It is desirable that the RUBs do not change and remain stable over time. Recently, a few paradigms for designing unique identification circuitry was proposed [24, 26, 30]. The resulting IDs are mostly stable, and we will later show how to extract a nonvolatile ID from the RUB, even in presence of a few unstable bits.

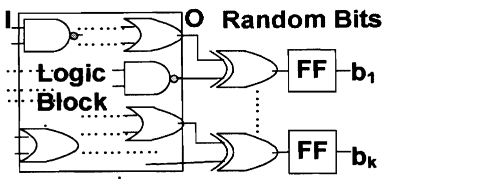
Addition of the BFSM. The key idea underlying the proposed active metering scheme can be described in a simple way. Assume that the original design contains m distinct states. Further assume that the state of STG are stored in k, 1-bit flip flops (FFs). The FFs represent a total of 2^k states, out of which m states correspond to the original design and $(2^k - m)$ states are dont cares. The metering mechanism adds an extra part to the FSM of the design. The added states are devised such that there are a number of transitions from the states in the added STG to the reset state q_0 of the original design.

In our scheme, the power-up state of each IC is built to be a function of the manufacturing variability and thus, will be unique to each instance. Furthermore, we select k such that $2^k - m >> m$. This selection ensures that when the circuit is powered up, its initial state will be in one of the added states in BFSM. Assume that the IC is powered up in the added state q_{a0}. During the standard testing phase, the manufacturer can read the state of the design, e.g., by scanning and reading the FF's. However, unless the foundry has the knowledge of the STG, finding the sequence of inputs required for the correct transition from the state q_{a0} to the reset state q_0 is a problem of exponential complexity. Essentially, there will be no way of finding the sequence other than trying all the possible combinations.

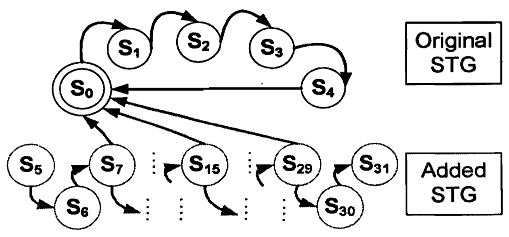
More formally, assume that the sequence of I primary inputs denoted as $\alpha_I=\{a_1, a_2, \ldots, a_I\}$ applied to the state q_{a0} is one correct sequence of states that starts from q_{a0} traverses I states denoted by $Q_I=\{q_{a1}, q_{a2}, \ldots, q_{a(I-1)}, q_0\}$, i.e., $q_0=\delta(q_{a0}, \alpha)$. Assuming that the input is b bits and there are cycles in STG, finding the correct input sequence that would result in I consecutive correct transitions is a problem with exponential complexity with respect to b and is thus, intractable.

As an example, consider the STG shown in Figure 3.3(b) that consists of the original STG that has five states ($\{q_0,q_1,q_2,q_3,q_4\}$) with augmentation of twenty seven added states ($\{q_5,q_6,\ldots,q_{31}\}$). Edges are incorporated to the added states to ensure that there are paths from each of the added states to the reset state of the design. The block shown in Figure 3.3(a) is a RUB.

The output of the RUB defines k random bits that will be loaded into the FF's of the augmented STG upon start-up. Now, an uninformed user who does not have the information about the transition table (e.g., foundry), can readout the data about the initial added state q_{a0}, but this information is not sufficient for finding the sequence of primary input combinations to arrive at the reset state q_0. However, the person who has the information about the structure of the STG, upon receiving the correct state, would exactly know how to traverse from this locked state to q_0. In other words, the



(a) RUB initially loading the STG's FFs



(b) The original and enhanced STG

Figure 3.3 : *The boosted FSM (BFSM).*

owner of the FSM description is the only entity who would have the *key* to unlock the IC.

An interesting application of the proposed BFSM construction method is in remote disabling. Alice will save the RUBs and the keys for all the ICs that she has unlocked. Using the chip IDs that are integrated within the functionality, she can add mechanisms that enable her to monitor the activities of the registered chips remotely, for example, if they are connected to the Internet. She can further add transitions from the original STG to untraversed states, to lock the IC in case it is needed. Remote disabling has a lot of applications. For example, it can be used for selective remote programming of the devices, and royalty enforcement.

3.4.2 Ensuring proper operation

The following issues and observations ensure proper operation and low-overhead of active hw metering:

(i) Storing the input sequence (key) for traversal to the initial state q_0. During testing, once Bob scans out the FF values and sends them to Alice, she provides the key to Bob. He includes both the original RUB and the key in the chip, for example, in a nonvolatile memory. This data is utilized along with the unclonable RUB circuit, for transition to the reset state. Since the power-up state is unique for each IC, sequence of inputs (key) that traverse the power-up state to the reset state is also specific to each IC. One needs to store the key which performs the traversal at the power-up state on each chip. There are many ways to accomplish this. For example, the designer could add a small programmable part to the design which needs to be *coded* with the unique sequence (key) before each IC is in operation. Coding ensures protection of keys against other software attacks. As an alternative, the sequence might not be included in the memory and just used as a permanent password to the IC.

(ii) Powering up in one of the added states. This condition can be easily guaranteed by selecting a large enough k. Assuming that all the states have an equal probability, the probability of starting in one of the added states is $(2^k - m)/2^k$. For a given m, we select k such that the probability of not being in one of the added states is smaller than a given probability. For example, for $m = 100$ and $k = 30$, the probability of starting up in an original state is less than 10^{-7}.

(iii) Diversity of power-up states (unique IDs). k should also be selected so that the probability of two ICs having the same ID becomes very low. Assume that we need to have d distinct ICs each with a unique ID. Assuming that the IDs are completely random and independent, we utilize the Birthday paradox to calculate this probability and to make it low. Consider the probability $P_{ICID}(k, d)$ that no two

ICs out of a group of d will have matching IDs out of 2^k equally possible IDs. Start with an arbitrary chip's ID. The probability that the second chip's ID is different is $(2^k - 1)/2^k$. Similarly, the probability that the third IC's ID 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 ID. More formally,

$$\begin{aligned} P_{ICID}(k,d) &= \frac{2^k-1}{2^k}.\frac{2^k-2}{2^k}.\dots.\frac{2^k-(d-1)}{2^k} \\ &= \frac{2^k!}{(2^k-d)!2^{dk}} \end{aligned} \tag{3.1}$$

Thus, knowing d, the number of required distinct copies, and setting a low value for P_{ICID}, we would be able to find k that satisfies the above equation.

(iv) Overhead of the added STG. The number of states increases exponentially with adding each new bit, and thus, the scheme has a very low overhead. Note that, in modern designs, the control path of the design (i.e., FSM) is less than 1% of the total area and hence, adding a small overhead to the FSM does not significantly affect the total area [47, 1]. In the next section, we will describe a low-overhead implementation of the proposed method.

(v) Diversity of keys. There is a need to ensure that the keys are distinct in all parts of their sequences, or there is a very small shared subsequence between different keys. This is granted by making multiple paths on the graph from each of the states to the reset state. We will elaborate more on this issue in the attack resiliency section.

3.5 Low overhead implementation and obfuscation

In this section, we discuss the implementation details of the RUB and the BFSM that are the required building blocks for the active hardware metering approach. We start by outlining the desired properties of each block, and then we delve into its implementation details.

3.5.1 RUB implementation

A critical aspect of the proposed security and protection mechanisms is the generation of random ID bits. There are a number of properties that the RUB implementation has to satisfy, including:

• **Low overhead.** The added parts must not introduce a significant additional overhead in terms of delay, power consumption and the area.

• **Distribution of IDs and their correlations.** To have the maximal difference between any two ID numbers (the maximal Hamming distances) the ID bits must be completely random. Thus, no correlation must be present among the ID bits on the same die or across various dies.

• **Indiscernibility.** The IDs must be integrated within the design, such that they cannot be discerned by studying the layout of the circuit. For example, the IDs should not be placed in a memory-like array, where the regularity of the array and its connections to the FFs could be easily detected.

• **Stability.** There is a need to stabilize the IDs over the lifetime of an IC. This is particularly important since studies have shown the temporal changes in CMOS process variations due to many environmental and aging effects including, residual charges, self-heating, negative-bias temperature instability, and hot electron effects [29].

For implementing the random IDs, we employed the recent novel approach proposed by Su et al. [30]. They have designed and tested a new CMOS random ID generation circuit that relies on digital latch threshold offset voltages. Using cross coupling of gates, they report significant improvement in readout speed and power consumption over the existing designs.

Each ID bit is generated by cross-coupled NOR gates. The latch sides are pulled low initially. At the high to low clock transition, the state of each latch is determined by the threshold voltage mismatch of the transistors. Essentially, the approach relies on the positive feedback inherent in the latch configuration to amplify the mismatch. This design removes the need for comparators, low offset amplifiers, or extra dopants needed in previous random ID generation methods [24, 31]. The nominal overhead of the above proposed approach is two NOR gates per bit. The authors have reported 96% stable IDs using this method, while using dummy latches to protect the IDs.

Even though we use the random bit architecture described above, our layout and implementation of random bits are very different. To be indiscernible, we do not place the coupled NOR gates in an array, and instead synthesize them with the rest of the circuit and camouflage them within the sea of gates. based on invariability of the ID bits for an IC. In Subsection 3.6.2, we provide a mechanism that ensures the occasional errors in ID bits do not affect the hardware metering approach.

3.5.2 BFSM implementation

The key design objectives and challenges of the BFSM are as follows:

• **Low overhead.** The addition of the states to the original FSM must have a low overhead in terms of area, power, and delay. This is particularly challenging: as we have computed in Subsection 3.4.2, even under the assumption of having RUBs with Uniform distribution of random bits, the number of added states must be exponentially high to ensure a proper operation.

- **Traversal path.** There must be a path on the BFSM, from each of the power-up states (except for the black hole states that we will describe in Section 3.6.2) to the reset state.

- **States obfuscation.** The states must be completely obfuscated and interchanged to camouflage the added STG and the original STG. Another level of obfuscation is disabling the observability of the FFs, so that similar states on two ICs do not exactly have the same code scanned out from their FFs.

- **Multiplicity of keys.** It is highly desirable to construct the paths on the added STG in such a way that there are multiple paths from each power-up state to the reset state. This will ensure that there are multiple keys for traversal. Now, if the states are obfuscated such that a similar state on two ICs has different codes, and each of them gets a different key for traversal to the original STG, the state similarities will not be apparent, even to a smart observer.

To achieve a low overhead, we have systematically designed STG blocks that are capable of producing an exponential number of states with respect to their underlying hardware resources. The blocks are designed such that there are multiple paths from each of the added states to the reset state and thus, the multiplicity of the keys is satisfied. Our first attempt was to synthesize the added blocks of STG and the original STG together. However, because the synthesis software automatically optimizes the interwoven architecture, it most often ended up with a combined STG that was much larger than the sum of its components. Thus, we decided to first separately synthesize the original and the added STG before we merge them. Next, we employed obfuscation methods that constantly alter the values of the FFs, even those that are not used in state assignment in the current STG. As we will see in attack resiliency section, the introduced obfuscation method has the side-benefit that the adversary cannot exactly distinguish a similar state on two different ICs.

The added STG can be designed to be low overhead; there are exponentially many states for each added FF, ignoring the overhead of the STG edges. However, in real situations, the transitions (edges) require logic. Thus, the added STG is constrained to be sparse to satisfy a low overhead. We have built this block in a modular way. We describe one of our modules here and then discuss systematically interconnecting the modules to have a multi-bit added STG that has a low overhead.

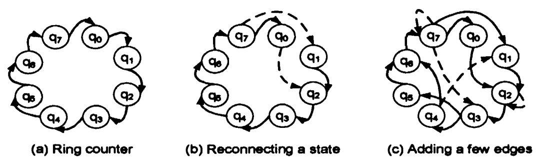


Figure 3.4 : Illustration of steps for building a sparse 3-bit STG.

The first module is a 3-bit added STG. In Figure 3.4, we show three steps for building this module. We start by a ring counter as shown in Graph 3.4(a). Next, we pick a few states and reconnect them to break the regularity. A small example is illustrated in Graph 3.4(b), where the state q_1 is reconnected, such that still there will be a path from each state to any other state. Finally, we add a few transitions (edges) to the STG, like the example shown in Graph 3.4(c); here the states q_1 and q_4 are reconnected, while the edges $\{ q_4 \rightarrow q_1, q_7 \rightarrow q_3, q_7 \rightarrow q_7, q_2 \rightarrow q_2 \}$ are added.

The example is just an illustration. Many other configurations are possible. The various combinations have different post-synthesis overhead. To ensure a low-overhead, we exhaustively searched the synthesized 3-bit structure with various sparse edge configurations like the example above, and selected the configurations with the lowest overhead as our 3-bit modules. As it is apparent from the structure, many low-overhead configurations are possible and we do not need to use the same module multiple times.

After that, we picked the low overhead modules and started to add edges to interconnect them, such that the connectivity property is satisfied, and the interconnected configuration still has low overhead. Furthermore, we need multiple interconnecting paths that can produce multiple keys. This is again done via a modular randomized edge addition and searching the space of the synthesized circuits to find the best multi-bit configurations. Note that, the synthesis program performs state-encoding for the interconnected modules. We have noticed that the distance of the codes assigned to the states does not have a correlation with the proximity of the states. Therefore, even for two RUBs that are only different in 1-bit, typically the power-up states are not close-by on the added STG.

In our experiment, we have tested our approach on 12, 15, and 18-bit added STGs. Now, the original STG has to be glued to the added part. This is done by an obfuscation scheme that ensures the states of FFs that are associated with the original STG keep pseudorandomly changing, even when we traverse the states of the added STGs. Thus, for an observer who studies the values of the interleaved FFs, the activity study would not yield an informative conclusion that can help separating the original and the added states. A simple example for this obfuscation is depicted in Figure 3.5. In this figure, a small original STG with five states is presented. The cloud shown below the original STG indicates the added states.

There are multiple state transitions from the added states to the original state. However, we only show one arrow on the plot not to make it more crowded. In this example, we use the three don't cares of the design for obfuscation purposes. There

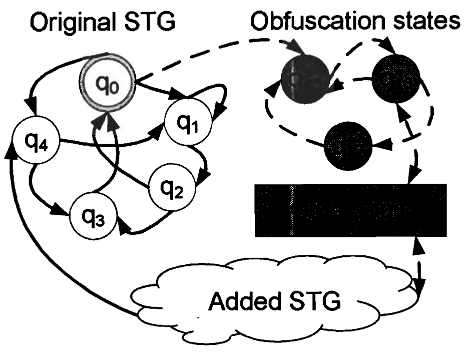


Figure 3.5 : *Obfuscation of the original STG.*

are 3 don't care states that we use to form three new dummy states $q*_5$, $q*_6$, and $q*_7$, illustrated in grey color. The glue logic attaches the inputs and the states of the added STG to the dummy STG. Thus, by carefully designing, one can alter the bits on the dummy STG by changing the input and the states of the added STG without touching the original FSM. If the design does not have sufficient don't cares, we can add a couple of FFs for the dummy states and use the same paradigm. The important requirement for the dummy states is that as a group they should present both 1 and 0 digits in all FFs. The original STG is also connected to the dummy STG and can utilize it as a black hole (described more thoroughly in Subsection 3.6.2), if there is a need to halt the IC.

3.6 Attack resiliency

This section first identifies several types of potential attacks on the active hardware metering approach. Next, we outline a number of mechanisms that must be added to the basic active metering scheme to ensure its resiliency against the suggested attacks.

The adversary (Bob) may attempt to perform a set of invasive or noninvasive attacks on the proposed active metering scheme. Bob may do so by measuring and probing one instance, or by statistically studying a collection of instances. In this section, we first identify and describe the attacks. Next, we propose efficient counter measures that can be taken to neutralize the effect of potential adversarial acts.

We assume that Bob knows all the concepts of the proposed hardware metering scheme, has the complete knowledge of the design at all levels of abstraction provided to the foundry (e.g., logic synthesis level netlist, and physical design GDS-II file, but no behavioral specification), can simultaneously observe all signals (data) on all interconnects and flip-flops (FFs), and can measure, with no error, all timing characteristics of all gates in the ICs.

3.6.1 Description of attacks

The starting point for development and evaluation of the metering schemes is identification and specification of several types of potential attacks:

(i) Brute-force attack. Bob aims to place the pertinent IC into the initial state by systematically applying the input sequences to the BFSM. The systematic application may be a randomized strategy, or may be based on scanning the FFs. Brute-force attack works by randomly changing the inputs in hope of arriving at the reset state. Scanning works by reading out the FF values for a few ICs and storing them. The FFs in the current IC are then monitored for the existence of a common state with the stored ones. In case a state that was read in the previous ICs is reached, Bob uses the same key for traversal to the reset state.

(ii) Reverse engineering of FSM. Bob may try to scan the FFs to extract the STG. The attempt would be to remove the added STG from the BFSM, to separate the original and the added states.

(iii) Combinational redundancy removal. Bob may use the combinational redundancy removal, a procedure that attempts to remove the combinational logic that is not necessary for the correct behavior of the circuit. The proposed techniques of this class often take into account the set of reachable states of the FSM under examination [48]. Note that, the attacks that were described so far can greatly benefit from the ability to simultaneously monitor the multitude of signals/values on the IC using laser reading.

(iv) RUB emulation. The goal of this attack is to create a reconfigurable implementation capable of realizing hardware that has the identical functional and timing characteristics to a RUB for which a legal key is already received.

(v) Initial power-up state capturing and replaying (CAR). Bob knows the initial power-up state of an unlocked IC. He can use invasive methods to load the FFs of other ICs to the same power-up state as the unlocked IC and then utilize the same key to decode the new locks. Note that, unless invasive methods are used, the only way for Bob to alter the values in the FFs is to change the states using the input pins. Without the knowledge of the STG, the change of state can only be done as described in the first attack. This attack and the next two belong to the class of replay attacks.

(vi) Initial reset state CAR. Bob scans the FF of an unlocked IC and reads the code of the reset state. Next, he employs invasive methods to load the FFs of other ICs to unlock them.

(vii) Control signals CAR. In this attack, Bob attempts to bypass the FSM by learning the control signals and attempting to emulate them. Bob may completely bypass FSM by creating a new FSM that provides control signals to all functional units, and control logic (e.g. MUX's and FFs) in the datapath.

(viii) Creation of identical ICs using selective IC release. Bob only releases the ICs with similar characteristics to Alice in the hope of finding the keys by correlations. This attack is probably the most expensive because it involves only a small percentage of manufactured ICs by the untrusted foundry. Only the ICs that have similar RUBs are reported. Hence, if the attack is successful, the design house supplies many keys for ICs with similar RUBs; the birthday paradox shows that one of the keys with relatively high likelihood can be used on the unreported ICs. Note that, the way for Bob to determine closeness of characteristics is by looking at the distances of the initial power-up states.

(ix) Differential FF activity measurement. Bob may start to investigate the differential activities of the FFs of the unlocked designs for the same input, and then try to eliminate the FFs that have different values.

3.6.2 Counter measures

We propose a number of mechanisms to augment the basic active metering scheme and preserve its security against the above attacks. Two important observations are that FSMs in modern industrial design are always a very small part the overall design, well below 1%, and that STG recovery is a computationally intractable problem [47, 1, 8]:

- **Creating black holes FSMs.** Alice may create a black holes FSM inside the BFSM that makes the exit impossible. Black holes are the states that cannot be exited regardless of the used input sequence. Their design is very simple as shown in Figure 3.6, where the black states do not have a route back to the other states. Furthermore,

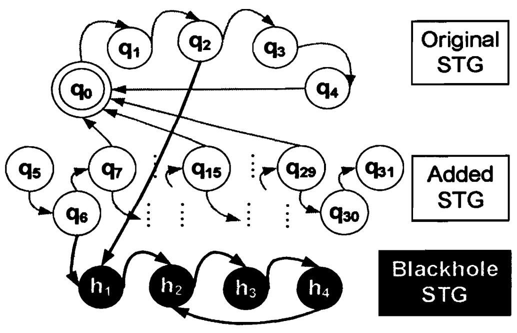


Figure 3.6 : *Example of a black hole FSM.*

a designer can plan the black hole states to be permanent if it is desirable: a small part may be added, so that restarting the IC would not take it out of the black hole states. This measure essentially eliminates the effectiveness of the first two attacks, because no random input sequence leads to the initial state of the functional FSM: once the black hole sub-FSM is entered, there is no way out. A special case is creation of trapdoor black (gray) holes FSMs that are designed in such a way that only long specific sequence of input signals known just to the designer can bring control out of this FSM and into the initial functional state of the overall FSM. An issue that needs to be carefully addressed here is preventing the IC from powering-up in one of the black-hole states. This can be easily ensured by adding extra logic to the black hole parts that would disconnect the black hole states from the power-up states.

- **Merging the functional BFSM with the test and other FSMs,** (e.g. ones that can be used for debugging and authentication). In a typical design, the functional control circuits are not the only FSMs around. Alice, with the the objective to make identification of her functional FSM more difficult, can further intricate the BFSM by co-synthesizing them with others. This augmentation makes the first two and the three CAR attacks less effective. In particular, this merger would distract the ability to simultaneously monitor the multitude of signals/values on the IC using laser reading.

- **Similar FF activity for the unlocked ICs.** The designs would be made such that once an IC exits the locked states and is in its functional states, all its FFs have a deterministic behavior that is the same for all ICs. Thus, the differential FF activity screening would not yield any useful information.

- **Creation of specialized functional FSMs (SFFSMs).** Alice can make the security much tighter by integrating the RUBs not just to assign the initial power-up state, but to alter the structure of the BFSM and make it a SFFSM. Using this method, the reset state for FSM of each IC is a function of its RUB. Each SFFSM operates correctly only if it received a specific stream of signals from the RUBs. Since there are exponentially many states with respect to the number of FFs in FSM, we map a set of blocks that share an identical subset of RUB outputs into a single SFFSM. This counter measure makes the first two attacks (i.e., brute-force attack and FSM reverse engineering) much more difficult and the first two CAR attacks (i.e., initial control signal CAR and initial reset state CAR) almost impossible.

A simple example of this method is presented in Figure 3.7. On this figure, the added STG is shown by the cloud on left, and the original STG is plotted in the right cloud. The original STG has only 3 states: a reset state two other states. Here, the original STG is replicated twice: One replication is denoted by SFFSM' and the other one is denoted by SFFSM". The scheme adds logic to the added STG, so based on the bits in the RUB, it will be categorized into three classes. Each of the classes will transition to one of the reset states in one of the clouds: original FSM, SFFSM' or SFFSM". This scheme will cause confusion in FFs scanning methods that aim at loading the reset state of an unlocked IC in the FFs of a locked chip. Note that, the replicated states need not all be unique, and maybe shared among the replicas to reduce the overhead.

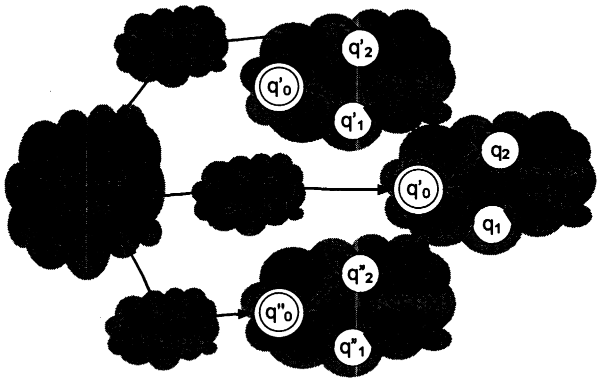


Figure 3.7 : *A simplified SFFSM.*

The example is very small, but one can add the RUB-dependent states at various stages to ensure that the attacker is not able to break the system. A combination of the SFFSM method and the state obfuscation and encoding would ensure a full security of the design against the CAR attacks. Furthermore, using similar methods, the RUBs can be also added to the obfuscation scheme based on the dummy variables like the example presented in Figure 3.5, so that the same inputs would have different random obfuscation patterns.

Another use of SFFSM is for addressing the effect of temporal changes in RUB. Recall that the actual application of the new hardware metering scheme to industrial designs requires mechanisms that ensure resiliency against time-dependent permanent changes of transistors as well as gate-level and transient changes due to the environmental conditions such as temperature and supply voltage fluctuations [29].

The exact reconstruction of the first power-up state of IC (the particular one for which the designer released the key) for the purpose of defeating the variabilities is trivial: Bob can just load the captured and saved outputs of the first power-up RUB for which he has obtained the key. This mechanism makes the design susceptible to reuse attacks, where Bob can reuse the key and the initial RUB for an unlocked IC to decipher another locked IC. However, if Alice included SFFSM in her design, she would be resilient against this attack. The only technical issue that remains to be addressed is to ensure that the SFFSM receives the correct data from the physical RUB, exactly the same as the one that was first received and for which the key is available. Otherwise, the stored key will fail.

In presence of temporal variations, ensuring that each SFFSM receives the correct data from RUB requires error-correction mechanisms. One solution is to employ standard error-correction codes (ECCs). An alternative hardware solution that encounters a lower overhead compared to ECC is to create the specifications of each SFFSM in such a way that it transitions into the correct next states, even when one or up to a specified number of the inputs from the RUB are altered by the environmental conditions. Using the hamming distances of the RUBs, we can group them into similar SFFSMs and synthesize the results such that the error correction mechanisms are inherently present. This mechanism is particularly effective for longer RUBs that are required for present industrial designs. Note that, because the minterms for the combinational logic that implements transitions are now not smaller than for non-resilient versions of the SFFSM, the hardware overhead is often zero or negative at the expense of the lower resiliency against brute force attacks [49]. However, since the probability of brute force attack can be made arbitrarily small with very low overhead (i.e., by using the black holes), this is a favorable trade-off.

- **Resiliency against combinational redundancy removal.** To overcome this attack, Alice must ensure the inapplicability of the attack to typical large circuits and the capability of this method to remove the added states. In general, computing a set of reachable states, can only be done for relatively small circuits, even when the implicit enumeration techniques are used. Thus, the method is only applicable to small circuits of small sizes.

- **Statistical characterization of gates.** Alice can go one step further and attempt to derive the gate-level characteristics of the manufactured ICs by measuring the input/output signals and exploiting the controllability and observability into the design. Essentially, knowing the circuit diagram, she would be able to write a linear systems of equations that can be solved for obtaining the approximate gate-level delay and power characteristics of the gates. She may even go further to use the extracted data to find the distribution of variations across the different chips (e.g., by using methods such as expectation maximization(EM)). Now, if the variations do not have enough fluctuations, then she will get suspicious and can halt the unlocking. This computation would ensure that the selective IC release would not be successful.

- **Obfuscation of state activities and encodings.** The implementation of the BFSM presented in the previous section renders it impossible to tell the difference between the original FSM FFs and the added states FFs. This is because all of the FFs are changing all the time. Therefore, even though two states of BFSM in two ICs might be identical, the attacks based on scanning the FFs would not notice that, since a subset of the bits will be different. In other word, the FFs not used in the added FSM are randomly changing. Another obfuscation method that has already been implemented is that the states in the added STG are not in order and are coded out of sequence by the synthesis tool. Thus, even though there might be a direct transition (edge) between two states, the methods based on FFs readings would not notice the proximity of the two states, since there code words are distant from each other.

Note that, the attacks that were described earlier, even the ones that are computationally very expensive, will not be able to unlock the ICs, if the counter measures described above are in place.

3.7 Experimental evaluations

To test the applicability of the method described earlier, we implemented the active hardware metering on standard benchmark designs. In this section, we present the experimental setup, followed by the overhead of implementing BFSM on the considered benchmarks. After that, we show quantitative analysis of the effectiveness of the brute force attacks. We further show how the addition of black holes can make the scheme resilient against this attack with a minimal overhead. Note that, many of the attacks described earlier are assuming structural counter measures that are hard to quantify and evaluate.

3.7.1 Experiment setup

We used extended set of sequential benchmarks from the ISCAS'89 to evaluate the impact of the active hardware metering method [50]. Even though the ISCAS'89 benchmarks are the latest comprehensive set of the gate-level designs, they are dated compared to the complex circuits in design, production and use today. Recall that following the Moore's law, the size and complexity of the circuits doubles approximately every 18 months. We use the larger benchmarks from the set, and we project the results to more complex circuits. Our projections show that the power, area, and delay overheads diminish as we increase the size and complexity. Simultaneously, the locking complexity and resiliency against the attacks exponentially improves, due the multiplicity of states. We synthesize the benchmarks using the Berkeley SIS tool [51], that given a STG or a logic-level description of a sequential circuit produces an optimized netlist in the target technology (cell library) while preserving the sequential input-output behavior. We have written a C program that modifies the benchmarks by adding the extra states. The program calls SIS to obtain the specifications of the synthesized and mapped original and modified STGs. When evaluating the overhead results, the important observation is that FSMs (i.e., the control circuitry) in modern industrial design are always a very small part of the overall design, well bellow 1% [47, 1]. Thus, even doubling the overhead, will have a minimal impact on the overall circuit that is mostly occupied by memory, testing pins, and data path circuitry.

3.7.2 Overhead of active hardware metering

Our first set of experiments study the overhead of the introduced scheme in terms of area, power, and delay. It is worth noting here that our ultimate goal is to integrate the active hardware metering method in the design flow. Thus we have considered testing the approach on manufactured ICs. However, the prohibitive cost of manufacturing a circuit in aggressive technologies (the quote we got for fabricating a circuit in 65nm was $500K) limits our experiments to synthesizing the benchmarks. Table 3.1 presents the results for the area overhead. Because of the relatively small size of the circuits, we added STGs with 12 FFs and 15 FFs overhead to the original STGs. The first column shows the name of the circuit from the ISCAS'89 benchmark. The second column shows the number of inputs to the circuit. The third column shows the number of outputs to the circuits. Both the number of inputs and the number

Circuit	Original Details				12 FFs		15 FFs	
	In	Out	FFs	Area	Area	%	Area	%
s27	4	1	3	18	224	11.44	278	14.44
s298	3	6	14	244	454	0.86	508	1.08
s344	9	11	15	269	480	0.78	534	0.99
s444	3	6	21	352	554	0.57	609	0.73
s526	3	6	21	445	648	0.46	702	0.58
s641	35	23	17	539	743	0.38	797	0.48
s713	35	23	17	591	793	0.34	847	0.43
s953	16	23	29	743	947	0.27	1001	0.35
s832	18	19	5	769	971	0.26	1025	0.33
s1238	14	14	18	1041	1264	0.21	1318	0.27
s1423	17	5	74	1164	1382	0.19	1436	0.23
s9234	36	39	135	7971	8174	0.03	8228	0.03
s13207	31	121	453	11248	11450	0.02	11504	0.02
s38417	28	106	1463	32246	32448	0.01	32502	0.01

Table 3.1 : Area overhead of active metering for various benchmarks.

of outputs do not change after adding the extra states. The fourth column shows the number of FFs in the original circuit. The fifth column shows the area of the original circuit. Then we show both the new area and the percentage overhead after adding 12 FFs and 15 FFs for the extra states. It can be seen that the percentage area overhead is decreasing as the circuit size increases. Thus, for larger circuit sizes, the area overhead will be even less insignificant.

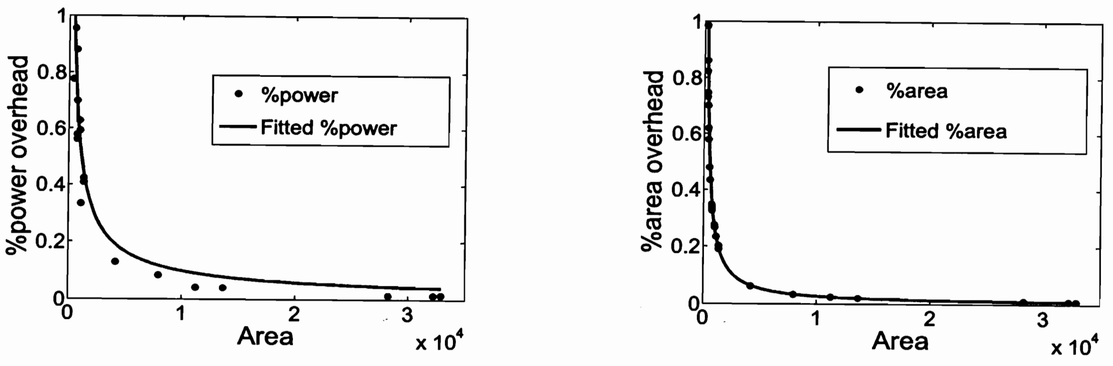
Table 3.2 shows the delay and power overheads. The first column contains the

Circuit	Original Details		12 FFs				15 FFs			
	Delay	Power	Delay	%	Power	%	Delay	%	Power	%
s27	6.60	134.00	14.40	1.18	1418.70	9.59	14.40	1.18	1696.70	11.66
s298	15.00	1167.20	15.00	0.00	2468.60	1.11	15.00	0.00	2746.60	1.35
s344	27.00	1030.00	27.00	0.00	2325.90	1.26	27.00	0.00	2603.90	1.53
s444	17.60	1550.80	17.60	0.00	2815.20	0.82	17.60	0.00	3152.30	1.03
s526	15.20	2065.70	15.20	0.00	3334.30	0.61	15.20	0.00	3664.70	0.77
s641	97.60	1560.60	97.60	0.00	2832.10	0.81	97.60	0.00	3162.40	1.03
s713	100.00	1670.70	100.00	0.00	2935.00	0.76	100.00	0.00	3265.40	0.95
s953	23.60	1816.50	23.60	0.00	3084.20	0.70	23.60	0.00	3414.60	0.88
s832	28.80	2849.60	28.80	0.00	4114.00	0.44	28.80	0.00	4444.40	0.56
s1238	34.40	2709.40	34.40	0.00	4034.00	0.49	34.40	0.00	4312.00	0.59
s1423	92.40	4882.70	92.40	0.00	6226.30	0.28	92.40	0.00	6504.30	0.33
s5378	32.20	12459.40	32.20	0.00	13515.00	0.08	32.20	0.00	14057.50	0.13
s9234	75.80	19385.50	75.80	0.00	20653.30	0.07	75.80	0.00	20983.70	0.08
s13207	85.60	37874.00	85.60	0.00	39138.40	0.03	85.60	0.00	39402.00	0.04
s38417	69.40	112706.80	69.40	0.00	113869.00	0.01	69.40	0.00	114147.00	0.01

Table 3.2 : Delay and power overhead of active metering for various benchmarks.

benchmark names. The second and third columns show the delay and power estimates of the original circuits. These are followed by both the delay and the percentage delay overhead, and the power and the percentage power overhead for adding both 12 FFs and 15 FFs STGs respectively. The delay overheads are universally small. With the exception of s27 that is too small to be considered practical, it is interesting to see that even other small benchmarks encountered no delay overhead after the addition of the new STG. For the small benchmarks that are not realistic compared to the current complex designs, the power increases significantly. As the circuit size increases, the percentage power overhead decreases.

Next, we make a small model of the percentage of area and power overhead versus size of the circuit to extrapolate to more complex designs. The size of the added STG is fixed to 15 FFs. Figures 3.8(a) and 3.8(b) show the overhead data vs. size along with the fitted polynomial models, for power and area respectively. The plots suggest that as the circuit size increases, the percentage of power and area overheads both



(a) % Power overhead vs. size. (b) % Area overhead vs. size.

Figure 3.8 : Percentage of (a) power; and (b) area; overheads vs. size after adding al 5 FFs STG.

decrease. Note that, for more complex designs, it is required to add significantly more than 15 FFs. Even if adding a STG with 100 FFs would add six times the overhead of the 15 FFs case in absolute terms, the overhead would be negligible, while there will be 2^{85} extra states added to the design. Thus, for current and future circuit technologies, the BFSM would have a minimal impact on the performance in terms of power, area, and delay (i.e., it will most likely stay less than 1% of the overall design).

3.7.3 Resiliency against the brute force attack

Most of the attacks described in Section 5.6 can be encountered by devising intelligent design strategies, as described in Subsection 3.6.2. The only attack that we quantitatively study here is the brute force attack. We model this attack by randomly guessing the values on the graph until arriving at the functional reset state of the original FSM.

We simulated the brute force attack on BFSMs with 12, 15, and 18 FFs, varying

	Number of inputs					
bits	**3**	**4**	**5**	**6**	**7**	**8**
12	74385	82708	78939	83156	77028	82490
15	560976	610373	602157	557776	592681	596260
18	933680	932501	938583	918312	N/R	N/R
12 + bh	998000	999000	N/R	N/R	N/R	N/R
15 + bh	N/R	N/R	N/R	N/R	N/R	N/R
12 + 2 bh	N/R	N/R	N/R	N/R	N/R	N/R
12 + 2 bh	N/R	N/R	N/R	N/R	N/R	N/R

Table 3.3 : Average number of attempts needed for the brute force attack to unlock the added STG.

the inputs from 3 to 8. In this experiment, we set an upper bound of 1,000,000 guesses; if the reset state is not reached after this many trials, the original STG is considered unreachable (denoted by N/R) and the brute force attack is reported unsuccessful.

Table 3.3 shows the average number of guesses needed to unlock the BFSM over a 10,000 simulation runs. The first three rows show added STGs with 12, 15, and 18 FFs respectively. The next two rows show the results for STGs with 12 and 15 FFs, after adding 1 and 2 black holes respectively. Although the number of inputs does not affect the overhead, it impacts the resiliency against the brute force attack: the table illustrates that the brute force attacks are less successful if we use more than 3 different inputs. Also, as the size of the added STG increases, more guesses are necessary to unlock the circuit. By adding one black hole to the smaller FSMs, they perform better than the larger FSMs. Adding one or two black holes makes the original STG unreachable for the brute force attack. It is worth noting here that

STGs with 12 and 15 FFs are really small, as they have a total of 4,096 and 32,768 states respectively. If the active metering scheme was to be implemented on current industrial strength designs, the added circuit would have at least a 100 FFs that would create $2^{100} \sim 10^{30}$ states. It would be impossible for a brute force attack to find a key. Furthermore, addition of a few black holes will further make the system resilient against the brute force attack.

Table 3.4 shows area and power overheads for adding a black hole with 2 states to added STGs with 12 and 15 FFs respectively. The overhead of adding a black hole does not exceed 5% even for very small benchmarks. For larger circuits it is unnoticeable. Note that, we often add more than one black hole to the design, to warrant the impossibility of the brute force attacks.

To evaluate the diversity of keys, we studied the number of cycles in the added STGs. For this STG, we form a new graph STG*, that has the same nodes as STG, but reverses the edges. Note that, simultaneously reversing all the edges will not affect the number of cycles in the graph. Since each state on STG has a path to the reset state, the directed acyclic graph (DAG) rooted at the original reset state in STG* will have a path to all states. We find a DAG of STG* by using the Dijkstra's shortest path algorithm. Next, we add the STG* edges to the DAG and see if they form a cycle and combine the cycles into one node; we iteratively continue until the cycles are gone. This approximate method is used to count the number of cycles. Using the method, we roughly guess that the STG with 12 FFs had more than 40 cycles that enables the use to build exponentially many keys for traversal from a certain state. The large number of keys can be easily generated by a combination of cycling and switching between the cycles of the STG.

Circuit	12 FFs % Area	12 FFs % Power	15 FFs % Area	15 FFs % Power
s27	0.05	0.04	0.04	0.03
s298	0.02	0.02	0.02	0.02
s344	0.04	0.02	0.03	0.02
s444	0.03	0.02	0.02	0.02
s526	0.01	0.02	0.01	0.02
s641	0.02	0.02	0.02	0.02
s713	0.01	0.02	0.01	0.02
s953	0.02	0.02	0.02	0.02
s832	0.02	0.01	0.02	0.01
s1238	0.01	0.01	0.01	0.01
s1423	0.01	0.01	0.01	0.01
s5378	0.00	0.00	0.00	0.00
s9234	0.00	0.00	0.00	0.00
s13207	0.00	0.00	0.00	0.00
s38417	0.00	0.00	0.00	0.00

Table 3.4 : Percentage of area and power overheads after adding one blackhole.

3.8 Potential applications

Active hardware metering provides strong anti-piracy mechanisms for hardware IP cores as well as remote-disabling mechanisms for the manufactured parts. Remote disabling can be accomplished if a malicious activity is detected. For example, a designer can add an extra part to the circuit that detects say, the brute force attack where too many invalid inputs are being entered. As another example, the strange activity patterns of the chip may be detected from a network. Upon detecting such a situation, a built-in disabling function would be invoked that transitions the IC into a non-functional state. If this state is a black hole, the IC cannot be used.

Generally speaking, combinations of the two employed security mechanisms, variability-based uniqueness of each IC, and structural manipulation of FSM while preserving the original behavioral specification, provide powerful basis for creating many security and DRM protocols. A few of the many possibilities are: (i) use of a combination of unique functionality and RUB for remote authentication and disablement of smart cards; (ii) certification that a computation was executed on a specified IC in a distributed environment; and (iii) creation of techniques to produce software than can only run on a specific IC, thereby preventing software piracy.

Furthermore, the introduced method has the potential for a broad impact on the IC industry and military use of hardware. As an example, new royalty enforcement systems can be enabled: design reuse has emerged as a dominant strategy, where different IP cores are often supplied by different vendors. The final integrator pays each IP supplier royalties that are proportional to the number of manufactured ICs. All that is needed for royalty enforcement is that each supplier uses its own active metering scheme inside its IP.

3.9 Conclusion

We propose the first active hardware metering scheme that symmetrically protects the IP designer and the foundry by providing a key-exchange mechanism. The active metering method utilizes the unclonable variability-based ID of each silicon circuit (RUB) to uniquely lock the IC at the fabrication house. The FSM of the design is enhanced to include many added states, designed such that the RUB-based state is one of the random states with a very high probability. The state addition was done in such a way that it would not affect the functionality of the original design. The key to the locked IC can only be provided by the designer who knows the state transition graph of the design. We have illustrated the addition of black hole states to the BFSM which can be utilized for remote control and disabling of the ICs. Black hole states are also useful in making the protection scheme highly resilient against the brute force attacks. We presented a low overhead implementation for the hardware metering scheme, identified a comprehensive set of possible attacks, and provided mechanisms that make the scheme much more resilient against the attacks. Experimental evaluations of the proposed metering method on standard benchmark circuits illustrate the low overhead and the applicability of the approach on industrial-size designs and its resiliency against different attacks.

Chapter 4

Remote Activation of ICs for Piracy Prevention and Digital Right Management*

Abstract

We introduce a *remote activation* scheme that aims to protect integrated circuits (IC) intellectual property (IP) against piracy. Remote activation enables designers to lock each working IC and to then remotely enable it. The new method exploits *inherent unclonable variability* in modern manufacturing for unique identification (ID) and *integrate the IDs into the circuit functionality.* The objectives are realized by replication of a few states of the finite state machine (FSM) and adding control to the state transitions. On each chip, the added control signals are a function of the unique IDs and are thus unclonable. On standard benchmark circuits, the experimental results show that the novel activation method is stable, unclonable, attack-resilient, while having a low overhead and a unique key for each IC.

4.1 Introduction

The increasing manufacturing cost of ICs has bolstered the horizontal semiconductor business model, in which design and manufacturing are done by different companies. As a consequence, the sole asset of a design company is the hardware intellectual

*This work is published in: Y. Alkabani, F. Koushanfar, and M. Potkonjak, Remote activation of ics for piracy prevention and digital right management, in IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 674-677, 2007.

property (IP), since designers typically do not control the number of ICs manufactured from a design. Also, hardware piracy, the illegal manufacturing of ICs using IPs, is omnipresent.

With the horizontal business model, digital rights management (DRM) is becoming an increasingly challenging problem due to the following factors. (i) The designers have no control over manufacturing of individual ICs from a unique mask. (ii) The IC internals are intrinsically opaque, and hence there is limited controllability and observability inside manufactured ICs. (iii) Hardware piracy of state-of-the-art IPs may only be done at fabrication facilities (fabs), where there are large resources and access to the most advanced tools and techniques. (iv) Finally, there is an asymmetric relationship between the designer and the fab, since the fab has full access to the design files, netlists and test vectors.

To overcome the hardware IP piracy problem, various watermarking [38] and metering protocols have been proposed [34, 25, 3]. IC *metering* involves a set of security protocols that enables the design house to gain post-fabrication control through passive or active counts of produced ICs, through the monitoring of IC properties and use, and through remote runtime disabling. Note that, IP watermarking is not the same as metering. A watermark uniquely identifies each IP, not each IC, and hence the existence of the same mask does not affect the watermarked IP. With *passive* metering, each IC is uniquely registered into a database, so a suspicious chip's ID can be authenticated against the database. In *active metering*, the IDs lock the chip's functionality. Unless the corresponding key is provided, the chip will not be able to operate properly. We propose a new IC activation scheme that works by not just active metering of ICs during the chip activation, but by also remotely enabling the chip's regular operation. We denote the diverse ID generation circuitry by a Diverse

Random Unique Block (RUB). For each IC, the block generates a unique RUB output for each RUB input vector.

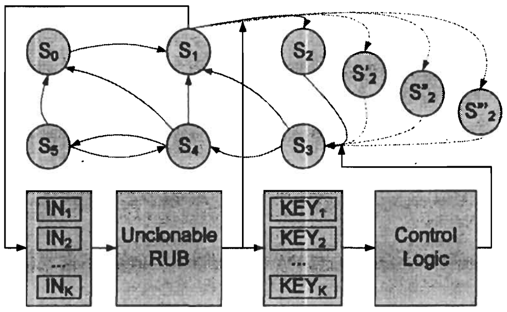


Figure 4.1 : FSM with a lock on the replicated state (S_2).

In Figure 5.1, we illustrate the new remote activation method. The FSM of a design with 6 states S_0 to S_5 is shown, where one of the states (S_2) is replicated three times: S'_2, S''_2,S'''_2. Once the design reaches the state S_1, it will transition to one of the four replicated states, depending on the RUB output (which is a function of its specific input selected by the control circuitry). Unless the correct key corresponding to the diverse RUB is provided, the state cannot transition to S_3 (hence it will be locked).

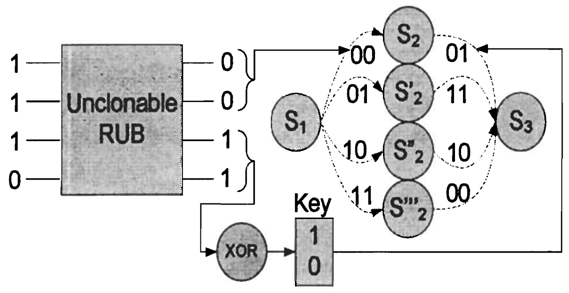


Figure 4.2 : Close-up of the locking/unlocking mechanism.

Figure 4.2 shows a closer look into the locking/unlocking mechanism. Assume that there is a two-bit input controlling the edge transitions of the FSM. Assume further that upon reaching the state S_1, some input key with the value 1110 is selected and the corresponding RUB output of the IC under test is 0011. The first two bits define the transition to one of the next replicated states (S_2 in Figure 4.2). Now, unless the next input is 01, the state cannot reach S_3 and the circuit will be locked. To enable the transition, the next two RUB outputs (11) should be XOR'd with a key that can generate the 01 output (in this case 10 is the key). For each authorized chip, a specific set of RUB inputs will be provided to the fab. Upon manufacturing, the fab will test the output of the specific input set by scanning the FF chains and reporting the stored values. The corresponding output set will be sent to the designer who will provide the corresponding key to unlock the chip. The specific input sets and corresponding unique keys will be stored on the chip to ensure proper operation. In practice, longer inputs and more replicated states will be used to guarantee security.

4.2 Related Work

Manufacturing variability (MV) has been used for generation of unique on chip IDs [24, 26, 30]; however, the IDs were not integrated into the functionality. The first IC metering scheme was proposed in 2001 [34, 25], but the scheme was passive. Variability-induced delays were used for authentication and security [31, 32, 33, 52]. The underlying mechanism was Physically Unclonable Functions (PUFs) that map a set of challenges to a set of responses, based on an intractably complex physical system. Because MV could cause delay differences among ICs made from the same mask, PUFs would be unique. A database of challenge-response sets is formed for each IC. The chip is authenticated if it can retrieve the output of one or more challenge inputs. Note that, PUFs were only used for authentication against a database, and have not been an integral part of the control path like our scheme.

The first active metering was recently proposed [3]. The method added exponentially many states to the original FSM of the design and exploited a static ID to lock the initial state of each IC. The ICs are unlocked with a key that uses the structural knowledge of the FSM graph to guide the transition from the unique start-up state to the initial functional state. The difference between this work and [3] is that, here, we do not add exponentially many states, but instead we replicate very few number of states. Also, unlike [3], our IDs are not static, but instead are a set of diverse unique IDs extracted from each chip. Perhaps most importantly, our scheme does not just lock the initial reset state of a design, but it also ciphers the functional states and their transitions. Thus, the diverse IDs are continuously checked against the transitions.

4.3 Remote IC Activation

Key exchange protocol. The remote activation protocol is designed to protect both the designer and the fab by requiring a key exchange mechanism for IC activation. It can be summarized using the following pseudocode.

1. The designer sends the design files to the fab, test vectors, and the number of required copies to the fab.

2. The fab manufactures the required number of ICs, applies the test vectors and sends the IC outputs to the designer.

3. The designer uses the values sent by the fab and computes a key to operate the chip properly.

4. The fab stores the key on the chip and tests the chip in the operational mode.

The remote activation scheme works in the following way.

A. Modification of the FSM. To modify the FSM we replicate a few states a number of times. For instance, if we have states S_0 to S_n, we can pick state S_i and replicate it 4 times to get S_i', S_i'', S_i''', and S_i''''. Note that adding one bit to the state will exponentially increase the number of states making the process of adding states have a low overhead. For each added state all the transitions to and from the replicated state are connected and are a function of a subset of the bits coming from the RUB. The transitions from the replicated states converge to the same state if they have the correct inputs from the RUB. Half the bits responsible for the transitions to and from the replicated states come directly from the RUB. The rest of the bits have to be keyed to be set to the correct value as shown in Figure 4.2. To maintain proper functionality when the circuit operates, the correct key value is used to cause the transition from one of the copies of S_i to the correct state.

B. The unclonable random unique blocks (RUBs). Lee et al.[32] designed a circuit that generates many unique outputs using selector bits that select various path segments and creates racing paths. We build upon this architecture by adding nonlinearity to make the structure more secure and reverse engineering resilient compared to the original circuit in [32]. The added nonlinearity is not only placed in the delayed paths segments but is also used by the selectors. Figure 4.3 shows the block diagram of a new RUB circuit. The circuit has nine inputs I_1 to I_9 and three outputs O_1 to O_3. The main building blocks of the RUB are selector elements (Sel), delay elements (D), and arbiters (A). A selector element has two input lines and one selection line. Based on the value of the selection line the inputs either pass directly to the output or are

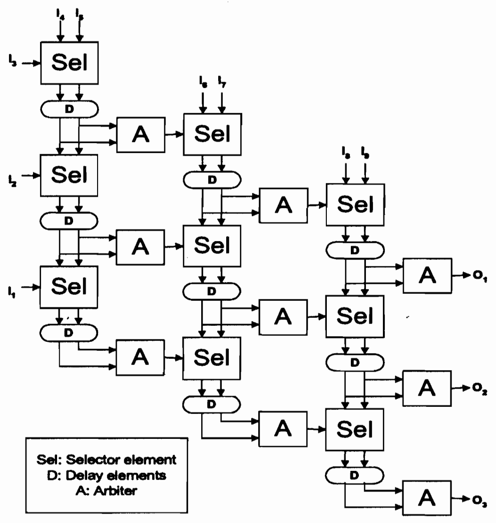


Figure 4.3 : A non-linear unclonable random unique block.

switched. The arbiter gives an output that depends on the input that arrives first (with the shortest delay) at power-up.

C. Input and key memory. The input to the RUB and the key are stored in memory to maintain the proper operation of the circuit. At the initial stage the input memory is loaded with different values to give read out data to the designer to compute the key. Once the designer computes the key and gives it to the fab, the key in the memory will be loaded to ensure that the transitions that occur as a function of the RUB occur correctly and the chip is functional. Figure 4.2 shows the input and key memory and how they are XOR'd with the output of the RUB for correct functionality.

D. Modification of the FSM control circuitry. The control circuitry of the FSM is modified to ensure that the transitions to the replicated states are function of the output coming directly from the RUB. The transitions from these states are functions of the RUB and the key. For example, they can be combined using a simple XOR as shown in Figure 4.2.

4.4 Attacks

In this section we discuss some of the potential attacks on the remote activation scheme and how we address them.

(i) Brute-force attack. The application of random inputs is not a feasible attack because of the exponentially low probability of correct guessing.

(ii) Reverse engineering of FSM. This attack is infeasible because the extraction of the corresponding state transition graph (STG) is a computationally intractable task.

(iii) Combinational redundancy removal. The attacker may use the combinational redundancy removal, a procedure that attempts to remove the combinational logic that is not necessary for the correct behavior of the circuit. The integration into the functionality of the circuit makes this attack impossible.

(iv) RUB emulation. The goal of this attack is to create a reconfigurable implementation capable of realizing hardware that has the identical functional and timing characteristics of a RUB for which a legal key is already received. However, the state of technology prevents this type of attack.

4.5 Experimental results

We studied the area, delay, and power overheads and the diversity of the keys generated by the RUB. We used sequential benchmarks from the MCNC'91 set, and Berkeley SIS for synthesis.

4.5.1 Area, delay, and power overheads

To modify the original FSM, a C program linked to SIS was written to manipulate the FSM and replicate some states. The number of replicated states was set as a

Table 4.1 : Area overhead for adding six extra states for *Random* and *Heuristic* state selection.

	Original			Random		Heuristic	
BM	**PI**	**states**	**area**	**area**	**%**	**area**	**%**
planet	7	48	888	1,373	54.62	1,191	34.12
s510	19	47	605	1,005	66.12	987	63.14
s1494	8	48	859	2,655	209	1,276	48.54
s1488	8	48	880	2,457	179	1,248	41.82
s298	3	135	2,951	5,924	101	3,769	27.72
dk16	2	27	460	964	109.6	898	95.22
sand	11	32	1,092	3,851	253	1,634	49.63
s820	18	24	430	918	113.5	1,040	141.9
s832	18	24	425	927	118.1	1,135	167.1
styr	9	30	633	1,777	180.7	1,306	106.3
Avg	10.3	46.3	922.3	2,185	138.4	1,448	77.54

parameter. We selected the states for replication using two methods,random selection and by selecting the state with the least number of outgoing edges in original STG. Table 4.1 shows the area overhead for the original FSM, the modified FSM with random state selection *Random*, and the modified FSM with states with least edges selected *Heuristic*. The first column shows the names of ten sequential benchmarks from MCNC'91. The second, third, and fourth columns show the number of primary inputs, the number of states, and the area of the original FSM after optimization using SIS. The remainder of the columns shows the area and the percentage overhead for adding six extra states using the *Random* and the *Heuristic* methods, respectively. All the circuits are optimized using the same parameters. The average overhead is 138.4% and 77.5% for the *Random* and the *Heuristic* methods, respectively.

Table 4.2 demonstrates the delay and power overheads for the *Random* and *Heuristic* methods compared to the original FSMs. It is interesting to note that in many cases the delay overhead is negative. The average delay overhead is 3.7% and -11.4% for the *Random* and *Heuristic* methods respectively, the power overhead is 148.9% and

Table 4.2 : Delay and power overhead for *Random* and *Heuristic* selection methods.

	Original		Random				Heuristic			
BM	delay	power	delay	%	power	%	delay	%	power	%
planet	186.2	3,087	93.90	-49.57	4,963	60.77	67.70	-63.64	4,374	41.68
s510	47.60	2,280	81.10	70.38	3,679	61.34	70.90	48.95	3,489	53.02
s1494	115.60	2,958	143.60	24.22	9,435	218.98	84.60	-26.82	4,560	54.15
s1488	134.9	3,011	91.40	-32.25	8,637	186.86	86.70	-35.73	4,450	47.81
s298	201.50	10,798	134.3	-33.35	26,005	140.81	182.60	-9.38	13,896	28.68
dk16	104.70	1,662	87.50	-16.43	3,595	116.32	74.70	-28.65	3,361	102.26
sand	74.80	3,917	88.80	18.72	14,312	265.37	61.50	-17.78	6,084	55.31
s820	36.30	1,430	54.70	50.69	3,054	113.49	47.30	30.30	3,729	160.71
s832	33.60	1,441	44.20	31.55	3,320	130.37	45.40	35.12	4,014	178.50
styr	128.20	2,170	93.40	-27.15	6,401	194.96	68.90	-46.26	4,718	117.40
Avg	106.34	3,276	91.29	3.68	8,340	148.93	79.03	-11.39	5,267	83.95

84%. The results show that the *Heuristic* method outperforms the *Random* method on average by cutting the overheads into half. Using the *Heuristic* method the average area and power consumption of the modified circuit is less than the double of the original circuit. Because the FSM is usually a very small part of the circuit, a larger area and power overhead is not expected to affect the overall system area and power on a chip.

Figure 4.4 shows the area, delay, and power overhead of adding different numbers of extra states using the *Heuristic* method. It can be seen that the overheads are nonlinear due to the optimization of the circuits during synthesis. Also it can be seen that the delay overhead decreases as the number of the extra states increases.

4.5.2 Diversity of the keys

To study the diversity of the keys produced by the RUB, we simulate a four stages RUB. The RUB has 12 input bits and 4 output bits. We randomly generated 1,000 RUBs and calculated the number of inputs generating different outputs for every RUB. Figure 4.5 shows a histogram of the average number of inputs that have unique

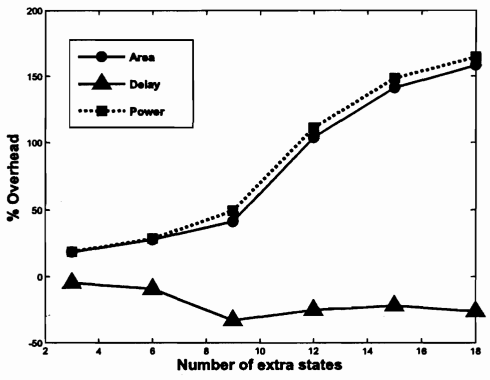


Figure 4.4 : Area, delay, and power overhead for different numbers of extra states for benchmark s298.

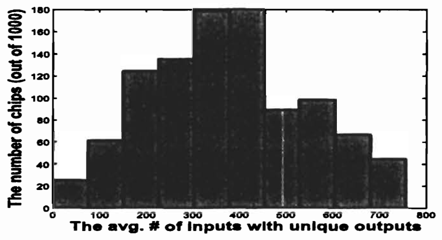


Figure 4.5 : The average number of inputs producing unique output for 1,000 different RUBs.

outputs for the 1,000 simulated RUBs. The figure shows diversity in the keys generated by the different RUBs.

4.6 Conclusion

We have developed a new approach for remote enabling, disabling and metering of integrated circuits. The approach leverages inherent manufacturing variability of modern and pending Si technologies. The key conceptual novelty is that designers can control ICs remotely continuously and concurrently with execution. The approach is evaluated in terms of delay, power, and area overheads as well as in terms of the achieved security.

Chapter 5

Active Control and Digital Rights Management of Integrated Circuit IP Cores*

Abstract

We introduce the first approach that can actively control multiple hardware intellectual property (IP) cores used in an integrated circuit (IC). The IP rights owner(s) can remotely monitor, control, enable, or disable each individual IP on each chip. The approach introduces a paradigm shift in the microelectronic business model, nurturing smaller businesses, and supporting the design-reuse paradigm. The IPs can be controlled by the original designer or by the designers who reuse them. Each IP has a built-in functional lock that pertains to the unique unclonable ID of the chip. A control structure that coordinates the locking and unlocking of the IPs is embedded within the IC. We introduce a trusted third party approach for issuing certificates of authenticity, in case it is required for the applications. We present methods for safeguarding the approach against two attack sources: the foundry (fab), and the reuser. Experimental results show that our approach can be implemented with low area, power, and delay overheads making it suitable for embedded systems. The introduced control method is also low overhead in terms of the added steps to the current design and manufacturing flow.

*This work is published in:Y. Alkabani and F. Koushanfar, Active control and digital rights management of integrated circuit IP cores, in ACM/IEEE International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES), pp. 227-234, 2008.

5.1 Introduction

The state-of-the-art digital ICs are increasingly complex. The progressive demand for multiple applications, performance, and functionality for integrated circuits has resulted in extreme CMOS miniaturization that add to the complexity. Building, operating, maintaining and upgrading silicon fabs for the complex designs is prohibitively expensive, e.g., upgrade to the current technology, 45nm, costs about $4bn [23]. The leading edge design companies are fabless. Even the large semiconductor companies including Texas Instruments (TI) and Freescale that had in-house manufacturing recently started to outsource their fabrication.

Because of the complexity, adapting the design reuse paradigm is the key to address constraints such as low-power, real-time budgets, silicon efficiency, time-to-market, and low cost [53]. A consequence of the current shift towards the fabless business model and design reuse is increased horizontalization of the microelectronic industry. Integration of multiple functionalities, applications, and design techniques has lead to modularity and specialization of design houses.

Many fabless design companies, particularly the specialized IP core designers are small. Their major investment is the technical and engineering staff and human resources who work together to produce the IP product. If the IP is ever exploited the company loses its capital investment. It is also likely that the IPs accidentally or through negligence are misused. For example, a design engineer (who we call a "reuser") may not take the time to check each core's license agreement. The IP-core design companies only receive revenue when their core is licensed to reusers, regardless of the volume and profit of the end product(s) that typically include multiple IPs. The presence of smaller companies is essential for a competitive market, but those companies endanger consolidation in the current business model.

We propose a novel approach that allows the IP core providers to gain post-fabrication control over their IPs on each chip. The approach introduces a paradigm shift in the digital rights management (DRM) of integrated circuits IP cores for vendors, designers and foundries. Depending on the application, the method may be used to control the number of chips that implement the IP, to remotely and actively enable or disable the usage. The misuse of the IP products is not only detected, but also prevented. The method works by uniquely locking the functionality of the IP core embedded in the manufactured chips, such that the rights owner is the only entity who can provide the key to unlock it. Our contributions include:

• Introduction of the first architecture and implementation for individual control of each IP core, in a multi-IP design.

• Integration of locking into each IP's functionality and coordinating the IPs by the reuser's control core.

• Control of the IP cores that may be done by the original core provider, the IP reuser, or both.

• Successful integration of the method within the standard synthesis flow, with a minimal addition of steps.

• Low-overhead and efficient implementation of the approach on chips containing multiple industrial benchmark IP cores.

• Ensuring trustworthiness of the key-exchange protocol by introducing a trusted third party providing certificate of authenticity.

• Discussion of attacks and providing safeguards.

• Introduction of a number of possible applications that are enabled by the new multi-IP protection method.

Motivational Example: Figure 5.1 presents a reuser's design which contains multiple IP cores. The cores denoted by IP_1, IP_2, ..., to IP_K are the protected ones. The functional control unit of each IP is represented by a finite state machine (FSM). The circuit designer (reuser) includes two new modules in her design. One added part is an identification (ID) circuitry that extracts the unique identification bits for the chip using the silicon variability [54, 55, 56]. The other addition is a control module that is embedded within the central controller of the chip. Each protected IP is directly connected to the the ID circuitry. Each of the protected IPs contains a lock within their functional states.

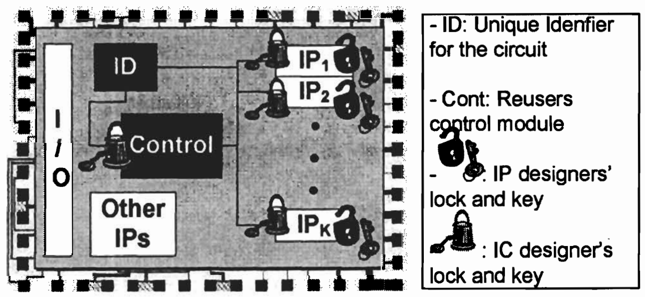

ID
I/O
Control
Other IPs
IP1
IP2
IPK
- ID: Unique Idenfier for the circuit
- Cont: Reusers control module
- : IP designers' lock and key
: IC designer's lock and key

Figure 5.1 : A reuser's design including multiple IP cores. Each IP may be locked/unlocked by the IP designer or the reuser, depending on the application.

The remote enabling/disabling provides two sets of locks and keys, one for the designer and one for the reuser. The locks are embedded within the control structure of each IP that can be represented by a finite state machine (FSM) [4]. There are two major advantages for the selected locking/unlocking mechanism: (i) the IDs come from the variations of the physical structure of silicon and are therefore random and unclonable, and (ii) the locks are integrated within the functional control structure, so removing or tampering the lock would tamper the functionality, rendering the IP unusable. Furthermore, as we will show, modifying the FSM does not result in a significant overhead.

Many protection, security, and DRM protocols can be enabled by the new IP locking/unlocking method. For example, the core providers can protect their IPs against overbuilding a licensed product, since each IP would be locked upon manufacturing. As another example, the reuser who has another set of locks/keys on the IP can select which IPs (or even features) are activated on the chip, e.g., for charging the customers who are willing to pay for added features. In the remainder of the paper we show the details of the new approach, implementation, experiments, and applications.

5.2 Related Work

Methods for digital design reuse and intellectual property trading are emerging [57, 58, 53, 3, 4, 2]. Protection of IPs in the reuse-based design flow is of paramount importance, but the prior work on individual IP protection has been limited. Most of the effort has been focused on FPGA soft IP core protection [59, 60]. A number of watermarking methods for IP identification have been proposed, but unlike our method that is active and uniquely locks each chip, a watermark is passive and is the same on all the chips implementing the same design [8, 38, 9]. A watermark can only be used to solve disputes about illegal usage of a design. It cannot identify, activate or disable individual ICs or IPs.

The inherent and unclonable silicon manufacturing variability has been used to uniquely identify each chip [24, 2]. Delay-based physically unclonable functions (PUFs) were constructed to extract the variability in circuit timing as a function of input (challenge) bits, generating a unique output (response) that can be used for identification and security [61, 55, 56]. PUFs were implemented in both ASICs and FPGAs [32, 62]. Several applications of PUFs are emerging, including RFID, proof of execution on a specific processor, securing processors, and active metering [54, 61, 62, 4, 63].

Recently, securing IPs in an ASIC design by individually tagging each core was proposed [64]. Since the tags are separated from the functionality, they are subject to removal attacks by both the reusers and the foundry. Note that approaches that use traditional implementations of cryptography protocols for securing at the low level are both high overhead and non-secure [65, 66], since the digitally stored keys are subject to physical and side-channel attacks [44].

Our new approach adapts the mechanism in [4], who integrated the unique identifiers of the chip into its control structure. The approach presented here includes several new aspects: First, multiple IP cores are controlled, not just one. Second, we consider interactions among the IP core designers, reusers, and the foundry, whereas the previous work only considered the designer-foundry relation. Third, unlike the previous work that only developed a control mechanism, we create a system-level secure IP integration solution and discuss the supply chain interactions. Fourth, we introduce the role of trusted integrator who will be useful for a secure design flow. Fifth, the reuser's role and possibilities of attacks are discussed for the first time. Lastly, the new approach directly applies to a number of novel system-level security, protection, and DRM methods that can be very useful for embedded systems (Section 5.8).

5.3 Flow of the Active Control for IP Cores

Figure 5.2 shows the overall flow of the new IP protection approach. There are four main entities involved: (i) IP rights owners (IP designers) who design, format and sell the individual IPs, (ii) IC rights owner (reuser) who integrates multiple IPs, including the open IPs and I/O interfaces, into one IC, (iii) The fabrication plant (fab), and (iv) an authorized system verifier; who we call a *certificate authority (CA)*. This entity ensures the trust between hardware IP providers, reusers, and the fab.

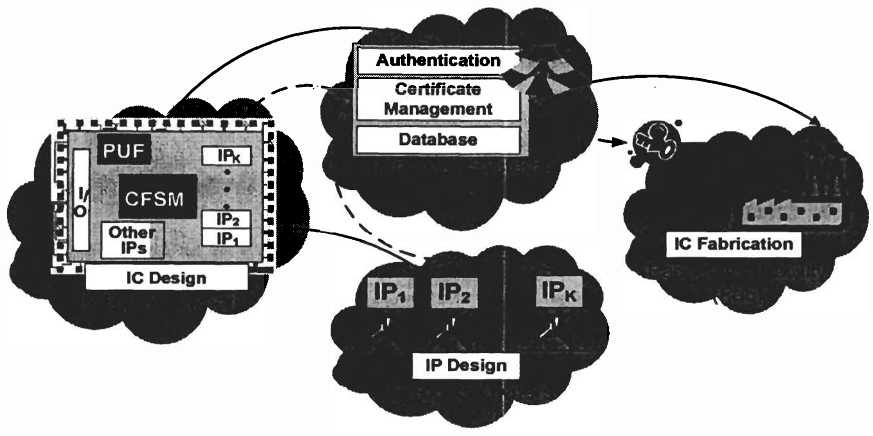


Figure 5.2 : The flow of the active control for integrated circuits' IP cores.

While the first three components are commonly present in the IC design cycle, the last component is new. CA is the trusted third party component for many asymmetric cryptography protocols, including several public key infrastructure (PKI) schemes. The new model is an asymmetric security scheme based on the keys provided by the IP designers and system designer. The CA provides trust by authorizing the parties; preventing possible breaches.

The flow can be described as follows. The IP designer forms the FSM of the design by using the high level design description. Then, the lock(s) are strategically embedded in the FSM. The modified finite state machine is called the boosted finite state machine (BFSM). The reuser may integrate multiple locked IPs, in addition to other components, including her own designs, unlocked IPs, I/O peripherals, memory, and the master identification/control parts. The master identification/control consists of a controlling finite state machine (CFSM) and a PUF. The CFSM interacts and controls the various IPs; it can enable/disable the other components. The PUF provides a mean for identifying each IC implementing the design in a unique and unclonable way. The ready-to-fab designs are shipped to the CA who certifies the IP cores and the reuser. The material is then sent to the fab who makes the masks and produces a number of ICs as specified by the contract. The operations described so far are shown by solid arrows on the figure. The dashed arrows present the steps required for key exchange transactions.

The fabricated ICs are nonfunctional and have locks on the CFSM and on the protected IPs from the providers. For each IC, the fab tests the PUF input and runs it through the flip flops (FFs) scan chain. The state of the IC will be read out from the FFs and sent to the CA who will in turn supply the state of each chip to the authorized reusers and IP providers. Each of the contacted parties will produce the specific keys to unlock the component. Also, the IP provider computes the error correcting code (ECC) for the lock, to mask the possible few changes caused by the fluctuations in the PUF identifiers. The keys are then sent back to the CA, who certifies the consent of the rights owners before sending them to the fab.

5.4 IP Control Method

In this section, we present the main modifications made to a multi-IP design to apply the method.

5.4.1 BFSMs

Each of the IP designers need to modify the FSM of their designed IP such that they embed a lock in it. The modified control structure is the BFSM. The BFSM is designed such that both its states and transitions are a function of the unique chip identifiers. The BFSM attempts to form a unique control path on each of the chips, while all the chips are from the same mask [4].

The BFSM of an IP core should satisfy the following properties.

- It must have incorrect functionality (locked) as long as the key is not provided.

- The key can be easily computed by the party who knows the BFSM structure and difficult to find otherwise.
- Knowing the key for one IC must not help in finding the key for another IC of the same design.
- Once the key is provided, the IP would function correctly.

Note that unlike symmetric cryptography where the keys are used to reverse a trap-door function and revealing the keys tampers the security, the keys here do not convey significant information about the lock. This is because the lock is in the structure of the state transition graph that is only known to the designer.

The same BFSM structure can be exploited to disable the chip during its operation. All what is needed is to modify the locks. For example, changing the PUF challenges will ensure that the functionality is trapped in a locked state.

5.4.2 CFSM

The overall FSM of the design that is devised by the reuse designer is also manipulated such that it embeds locks that allow the chip designer to lock/unlock her designed parts. Next, some states for controlling the other IPs are also included by the IC designer. We refer to these added IP control signals as CFSM. The CFSM gives the chip designer a level of control over the several IPs that are included in the design. For example, the CFSM receives signals from the IP cores about their locked/unlock status. The CFSM can also generate control signals that can enable or disable various IPs on the chip. There are many applications that can benefit from the CFSM (see Section 5.8).

5.4.3 PUF

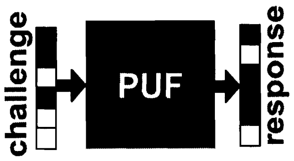


Figure 5.3 : PUF challenge/response pairs.

PUF is the circuitry which generates random unique values per chip. Figure 5.3 demonstrates the high level block diagram of a PUF [54]. The PUF circuit generates a unique response (output) for each input vector (challenge) that is applied to it. Even though the response varies from one chip to the next, the response to the same challenge remains the same over time.

PUF has a much larger overhead compared with BFSM and CFSM. Thus, we share it among the IPs to reduce the overhead. There is a need to ensure that the PUF is properly connected to the IPs so that the IP rights owner receives her proper royalties. The trusted third party (authorized system verifier) ensures the proper interface of PUF to the BFSMs before sending the design files to the fab.

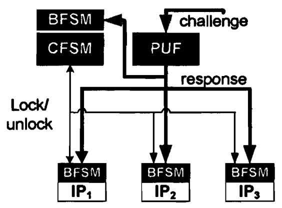


Figure 5.4 : System block diagram.

5.5 Implementation

Figure 5.4 shows the block diagram of the system components described earlier. Let us assume that we have three IPs denoted by IP_1, IP_2, and IP_3. The response of the PUF is connected to the IPs' BFSM, and the CFSM communicates with the BFSMs to control (lock/unlock) them. We outline the implementation of the BFSM, PUF, and CFSM.

5.5.1 BFSM Implementation

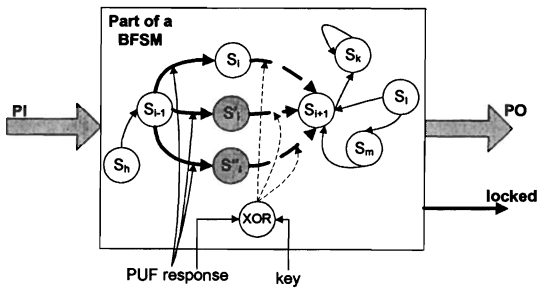


Figure 5.5 : Implementation of the BFSM.

The implementation of BFSM is inspired by [4] but the BFSM was further adapted and modified to include more states and communications with the CFSM. Figure 5.5 shows a part of a BFSM on a sample IP core where a state S_i is replicated twice as S'$_i$ and S"$_i$. The transitions to S$_i$ from S$_{i-1}$ are copied to its replicated states such that based on the PUF response, either S$_i$ or one of its replica is reached. The reached state is only a function of the PUF response. However, the transitions from the replicated states to S$_{i+1}$ are a function of both the PUF response and the key. The key and the response are XOR'd; if the output is correct, the valid state S$_{i+1}$ will be reached. Otherwise, a wrong transition (not shown on the figure) will be taken. PI/PO represent the set of primary inputs/outputs to the BFSM. Whenever a wrong transition is taken, the flag signal from the IP's BFSM is set to 1 to inform the CFSM that the BFSM is still unlocked. The flag value is 0 otherwise. The BFSM implementation steps can be summarized as follows:

1. The n states with the least number of outgoing edges are selected for replication.

2. Each selected state is replicated m times.

3. Transitions to the replicated states are a function of the PUF response and are thus unique to each chip.

4. Transitions from the replicated states are a function of the PUF response and the key. Correct transitions are only taken if the key is properly set. Incorrect random added transitions are taken when the key is wrong.

5.5.2 PUF Implementations

We implement the delay-based PUF introduced in [61]. The response is found by comparing the delay of two parallel paths that must be the same, but vary because of manufacturing fluctuations. The signal starts at the common starting point of the two paths on the left and ends at an arbiter which is inserted at the right end of the two parallel lines. If the signal on the top path arrived earlier, the arbiter output will be

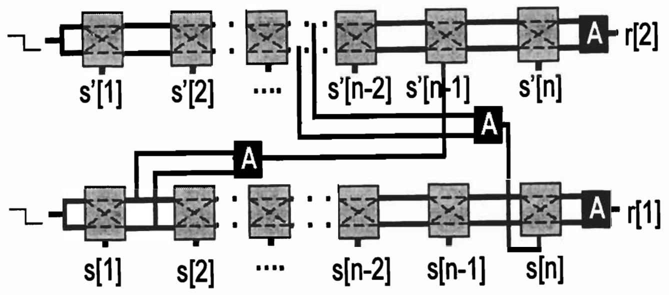


Figure 5.6 : Implementation of the PUF.

zero; otherwise, its output would be one. The parallel paths are divided into multiple segments, such that each segment is controlled by a switch. Different combinations of the path segments are selected by the switches, causing the racing path pair and also the arbiter output (response bit) to change.

The above PUF is vulnerable to modeling attacks because of its linear structure. Feedforward arbiters are used to alleviate this problem [32]. The added arbiters compare the delays of two partial path pairs and use the arbiter output as the selector line for a forward switch in the circuit. Figure 5.6 shows an example of a two bit output delay-based PUF with random feedforward arbiters which may also connect different path pairs. Switches s[1] to s[n] represent the cascade of switches for the first output, and s'[1] to s'[n] are the switches for the second output. From each path pair, we randomly select the output of a few switches and connect them to arbiters, then connect the output of these arbiters to selection lines of other switches constructing a feedforward connection. The selection lines of switches that are not connected in a feedforward (not shown in the figure) represent the challenge to the PUF, while r[1] and r[2] represent the response of the PUF.

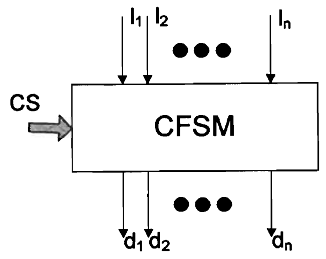


Figure 5.7 : Implementation of the CFSM.

5.5.3 CFSM Implementation

The CFSM is implemented as a finite state machine that is embedded and hidden inside the main FSM (BFSM) of the IC. A block diagram of the CFSM control signals is shown in Figure 5.7. The CFSM inputs can be divided into two groups: (1) inputs coming from the IP cores' BFSMs ($l_{1:n}$), and (2) external inputs (CS) that can be used to control the IC and enable/disable the IPs remotely. $l_{1:n}$ signals from BFSMs inform the CFSM if an IP is locked. The CS signals are used to upload the main key that determines which features (IPs) can be activated on a particular IC. Note that setting this key will not unlock the IPs, however, it will only help preventing the CFSM from disabling the whole chip when the IPs are locked. The output signals from the CFSM are $d_{1:n}$ that represent the disabling signals for the BFSMs in the IC. The CFSM continuously monitors all the BFSMs and if a BFSM that is supposed to be enabled is locked, the CFSM disables all the enabled IPs to detect and fix the control FSM.

5.6 Attacks and Safeguards

We envision two categories of attacks on the proposed method: foundry level attacks and IC designer level attacks. The IC designer (reuser) and the foundry do not have the same knowledge about the design and do not share the same objectives. For example, the reuser may tamper with the communicated signals to the IPs to overrule the owners' rights. The foundry may also overlook the rights of the IC designer.

Foundry level attacks and countermeasures are:

- **Brute-force attack.** This attack can be performed by continuously applying random inputs to each IP until the correct value of the key is found. This attack is not feasible for one IP because the probability of guessing the correct key is extremely low [3]. Having more than one IP locked in addition to the main design renders the attack even more infeasible.
- **Reverse engineering of the BFSMs and the CFSM.** One might try to reverse engineer the BFSM and the CFSM by STG extration. However, the computation of the STG is a computationally intractable task especially that the BFSMs are enlarged versions of the FSMs of the IPs in the system, and the CFSM is obfuscated by hiding its states within the large state-space of the IC's main FSM [8].
- **PUF emulation.** This attack attempts to emulate the behavior of the PUF of one unlocked IC and replicate it on the others. However, this attack is infeasible in the state-of-the-art manufacturing and software emulation is much slower and can be detected [55, 56].
- **Combinational redundancy removal.** Using a combinational redundancy removal software, one can try to remove all the extra states added to the different parts of the design. However, since all the modifications are integrated within the functionality of the different IPs, they are not redundant and this attack will not be successful.

IC designer level attacks and countermeasures are:

- **Bypassing the PUF.** The adversarial reuser may try to bypass the PUF interface to the other IPs so that only one key is needed to unlock different IPs, maintaining only the connections of the PUF to the main BFSM to keep the reuser rights. However, it is the responsibility of the CA to check the interfaces and ensure that the PUF is properly connected to the IPs.
- **Tampering with the PUF.** The designer can tamper with the PUF such that one of the racing paths is much longer than the other. This can cancel out the effect of MV and produce deterministic output for all the ICs. However, the trusted system verifier should also test and certify the PUF's randomness [55, 56].

5.7 Experimental Results

The proposed method is implemented and evaluated using the Berkeley SIS synthesis tool. All the programs are written in C. MCNC'91 sequential benchmarks are used to represent FSMs of different IPs. It should be noted that the FSM that contains the control part of any IP represents a very small fraction of the overall size of the design [1]. Thus, even tripling the overall area or power of these FSMs will not significantly affect the overall area and power of the IP. However, the delay of the FSM can affect the speed of the IP and thus, delay is the most important design metric in our implementation.

We show the overhead for using one and five IPs. Table 5.1 demonstrates the overhead when applying the metering method on one IP. The overhead number includes the overhead due to both the BFSM and the CFSM. The first column represents the benchmark number (C#) which will be used to refer to the benchmark in this section. The second column represents the name of the benchmark circuit. The third column

C#	circuit	PI	states	area	delay	power	area	%	delay	%	power	%
1	planet	7	48	888	186.2	3,087	1752	97	70.2	-62.3	6428	108.2
2	s510	19	47	605	47.6	2,280	1426	136	49.9	4.8	4555.8	99.8
3	s1494	8	48	859	115.6	2,958	1746	103	65.8	-43.1	5178.8	75.1
4	s1488	8	48	880	134.9	3,011	2045	132	68.1	-49.5	6008.8	99.6
5	s298	3	135	2,951	201.5	10,798	5960	102	136.8	-32.1	22358.8	107.1
6	dk16	2	27	460	104.7	1,662	1970	328	49.3	-52.9	5886.8	254.2
7	sand	11	32	1,092	74.8	3,917	1092	0	59.6	-20.3	8584.8	119.2
8	styr	9	30	633	128.2	2,170	2180	244	52.7	-58.9	6218.8	186.6
	Mean						2271	143	69	-39	8,153	131

Table 5.1 : The overhead of BFSM modifications for one IP.

shows the number of primary inputs (PIs) of the benchmark before modification. The fourth, fifth, and sixth column show the area, delay, and power overheads of the original benchmark. The rest of the columns show the area, delay and power of the modified IP and the percentage overhead of each parameter. It can be seen that the area and power overheads are on the average 143% and 131% respectively.Also, the delay overhead is low and is not affected by the number of IPs on the ICs. Thus, we do not report the delay overheads in our subsequent evaluations.

Next, we add a 16 stage random feedforward PUF with 64 cascaded switches per stage. The PUF has a total of 64 challenge bits and 16 response bits since we share the selector inputs that are below each other to keep the number of circuit inputs low. If we include the PUF in the overhead calculation, the overhead would be large. Note that the MCNC benchmarks are only control circuits and they do not include memory and I/O periphery/interfaces that are the area/power consuming components. Thus, the percentage of the added circuitry's overhead is much smaller than demonstrated.

Table 5.2 shows the overhead for integrating five benchmark circuits randomly selected from Table 5.1. It can be seen that the overheads without adding the PUF are almost constant. However, since the PUF's overhead is much larger than the FSM's overhead, adding the value of the overhead of the PUF to the system causes

C#	area	%	power	%
7,4,1,5,8	13,281	106	50,702	121
2,2,4,2,7	7,611	101	29,129	112
3,7,8,1,8	9,161	123	33,545	135
3,5,8,6,1	13,858	139	47,135	128
2,5,6,6,5	17,573	137	62,276	129
6,7,1,2,3	8,187	110	31,528	127
7,2,5,3,8	12,650	106	47,970	117
8,5,1,3,2	13,309	124	45,789	115
mean	11,954	118	43,509	123

Table 5.2 : CFSM overhead for integration of five IPs.

the overhead to decrease as we increase the number of the IPs sharing the PUF. Figure 5.8 shows the decrease of the overhead as we increase the number of IPs sharing the PUF.

5.8 Applications

The ability to uniquely identify each copy of an IP in a design-reuse paradigm enables a range of new applications, inluding:

Protection against foundry overbuilding. The IP control and CFSM control methods eliminate the possibility of overbuilding and hence prevent piracy by requiring the consent of the original designer and IP providers for enabling/disabling of their cores.

Protection against licensed designers' overuse. A reuser may utilize a singly licensed core in multiple designs. Detection of misused IP cores in a large design is

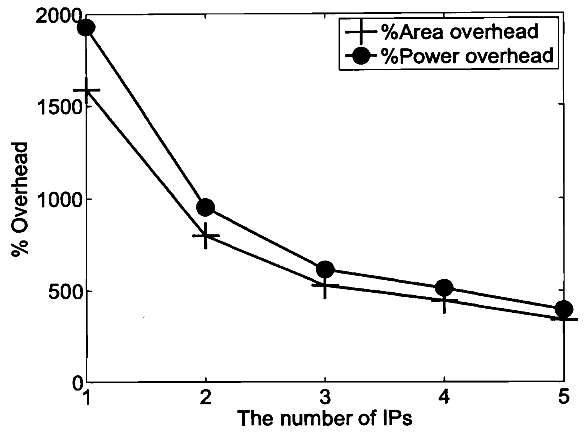


Figure 5.8 : The change of the overhead with increasing the number of IPs sharing the PUF.

a very hard problem. With the new method, no IP will be activated without the consent of its original designer.

Interval licensing by remote enabling/disabling of IPs. Runtime disabling/enabling of IPs can be done since the chips that contain the IPs are identified and can be detected online. A possible application is interval licensing, where the product royalty must be frequently paid for continuous usage of the IP; otherwise, the IP is disabled.

Software/content metering. The unique IP identifiers can be further exploited for controlling the software and content running on the hardware.

Ownership proof. The original key for operating an IP core is given only for one set of PUF responses. A way to prove the ownership of the IC is to change the challenge inputs and then ask the designer to provide a new key which renders this device operational. The designer who has the full information of the STG can easily provide the new key, but other entities cannot. Thus, the IP rights owners can assert their ownership by online checking and authentication.

Multiple levels of protection. The approach introduces symmetry to the current asymmetric business model. Not only the reuser, but also the IP designer and the fab are protected by the symmetry. In addition to preventing piracy, the false accusations of overbuilding or overuse are prevented.

Enabling pay-as-you-configure method for the reuser. The chip designer embeds its locks in the functionality of the IP cores. The reuser can design its chips such that the IPs that provide additional functionality are disabled. Only the customers who pay the proper fees may enable those IPs.

Support for the design reuse paradigm. One of the greatest challenges in reuse-based design is protection of the rights of the IP owners. Since the proposed method targets digital rights management of IPs, it supports the design reuse paradigm that is essential to the development and evolvement of the modern designs and semiconductor industry [67].

5.9 Conclusion

We introduced the first approach, architecture and implementation for actively and uniquely controlling the functionality of each IP, in a multi IP core design and reuse paradigm. The approach protects the rights of the IP core owners, reusers, and the foundry by introducing a key exchange mechanism. The IC and each of its embedded IP cores are uniquely locked upon manufacturing. The method enables the designers and reusers to actively and remotely lock/unlock their IPs on each of the ICs post-manufacturing. We discussed a number of possible attacks, and provided countermeasures against them. Experimental evaluations on standard benchmark circuits demonstrate the low overhead and the applicability of the approach on industrial-strength designs. We introduced a number of newly enabled applications in protection, DRM, and security of the IP cores.

Chapter 6

N-Version Temperature-Aware Scheduling and Binding*

Abstract

Technology scaling to nanometer nodes causes growing increase in power density and especially leakage that in turn result in locally hot regions on the chip. In this paper, we introduce a novel methodology for temperature-aware design. The methodology embeds N-versions of the scheduler and binder such that the thermal profiles of the versions are distant from each other. Next, instead of using only one version of the scheduler and binder, a rotation of N-versions of the scheduler and binder is constructed for balancing the thermal profile of the chip. We propose a linear programming framework that takes the multiple versions as the input, and constructs the thermal-aware rotational scheduling and binding by selecting the N most efficient versions and by determining the duration of each version. Our experimental evaluation shows a very low overhead and an average 5% decrease in the steady-state peak temperature produced on the benchmark designs compared to using a schedule that balances the amount of usage of different modules.

*This work is published in: Y. Alkabani, F. Koushanfar, and M. Potkonjak, N-version temperature-aware scheduling and binding, in International Symposium on Low Power Electronics and Designs (ISLPED), pp. 331-334, 2009.

6.1 Introduction

The intense feature scaling of CMOS has been driven by the growing application demands and pursuit of improved performance, as envisioned by Moore's law. Aggressive scaling lowers the cost-per-function, but it simultaneously escalates the device density and computational speed. The power density (i.e., power consumption per unit area) is also growing. The increased power generates heat on the chip. Since the heat propagation is slow compared with the switching activity of the IC, the heat would be concentrated at local regions, or so called *hotspots* [68, 69, 70]. The heat gradient increase would result in thermal stress that can speed up chip aging due to negative bias temperature instability, electromigration, or gradual dielectric breakdown. Therefore, circuit reliability would degrade.

The excessive increase in design complexity, power density, heat gradient, and unreliability of the scaled CMOS devices, has made thermal-aware design and optimization a strong research focus. In the ASICs domain, several solutions at different levels of design abstraction including scheduling, resource allocation, binding, floorplanning, and placement were developed [71, 72, 73, 74, 75, 76]. The common denominator for the existing work is that by assuming a certain model, they perform optimization (often iterative ones) that finds one optimized solution at the target level of the design abstraction.

In this paper, we introduce a paradigm shift by devising a flexible synthesis methodology that forms N-versions of the scheduling and binding solution each with unique thermal characteristics. The N-versions are simultaneously embedded into one design. During the operation, the versions would rotate such that each version would be used for a predefined time duration.

Our contributions are (1) Introduction of the concept of using rotational N-version scheduler and binder. (2) Designing an algorithm for finding the N-versions. (3) Methodology for construction and low-overhead implementation of the N-versions. (4) Development of a linear program that uses the thermal properties of each version to find the length and duration of N-versions. (5) Evaluation of the new method by comparison of the most balanced scheduler and binder.

6.2 Related work

In the past few years, a number of new approaches for thermal effect modeling and thermal-aware design has emerged. Banerjee et al. introduced a method for designing temperature and reliability aware cost trade-off with respect to power, performance and cooling [70]. Efficient modeling of the chip level and architecture level thermal characteristics have been proposed [68, 69]. The impact of thermal energy on power consumption was also studied [77]. Energy minimization has been addressed for a variety of systems including real-time [78], and under impact of manufacturing variability [63].

Chu and Wong proposed thermal-aware placement using a matrix synthesis method [71]. Tsai and Kang devised a standard cell placement tool for balancing the chip thermal distribution [72]. HotFloorplan is a floorplanning tool that manages the chip lateral heat propagation [73].

Mukherjee and Memik proposed a multistage integrated temperature optimization at the architectural synthesis levels [74]. They develop an iterative optimization method for scheduling and binding that gets feedback the post-floorplan thermal simulations. Ni and Memik studied thermal-induced leakage power optimization by redundant resource allocation [75]. Lim and Kim formulated the thermal-aware binding problem into a problem of repeated utilization of network flow method [76]. Shang et al. explain the challenges the power and temperature optimization pose for high-level synthesis researchers and summarizes the research progress in the field [79]. Zhang and Chatha present approximation algorithms for temperature-aware scheduling; for a set of periodic tasks executing on a processor the latency is minimized subject to thermal constraints [80]. Note that there is also a vast body of literature for thermal-aware design for MPSoC and programmable devices that are outside the scope of this paper. To the best of our knowledge, this is the first work that considers combining multiple schedules and allocation for efficient thermal management at the high level synthesis.

6.3 Flow

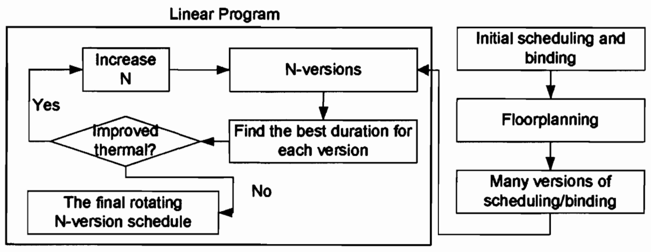

Linear Program
Increase N
N-versions
Yes
Improved thermal?
Find the best duration for each version
No
The final rotating N-version schedule
Initial scheduling and binding
Floorplanning
Many versions of scheduling/binding

Figure 6.1 : Flow of the rotational N-version thermal-aware scheduling and binding.

Figure 6.1 presents the rotational N-version thermal-aware scheduling and binding flow. The initial scheduling and binding is performed by list scheduling with a fixed timing constraint that minimizes the number of resources. A force-directed floor-planner is used after resource selection such that the similar resources are placed far from each other. This is to maximize the number of alternatives for a resource such that the temperature increase is independent among the alternatives. Finally, the maximally constrained minimally constraining rule is adopted for creation of many versions of the scheduler and binder. Next, the linear programming method finds N out of the several available scheduling and resource binding versions and their running durations, such that the selected versions have the smallest peak thermal energy dissipation.

6.4 N-version scheduling / binding

In this section, we describe how we generate the Multi-version schedules. Algorithm 1 shows the main steps for generating the N'-versions (N' $\geq$ N). The inputs to the Algorithm are G the CDFG of the circuit, and N' the number of generated versions. The output is the N' versions of the scheduling and binding method.

The first few steps generate a layout for different resources on a grid. In Step 2, we find the lower bound on each resource type denoted by R_t^{lb} using list scheduling. We do not change the critical path timing because in many of the DSP applications that we target, the throughput must remain constant. Next we choose the number of resources to be used for each module type (denoted by R_t in Step 3). R_t must be greater than or equal R_t^{lb} to guarantee that we do not alter the timing. In Step 4, we generate a force directed layout that ensures that resources of the same type are placed furthest away from each other. Finally, in Step 5, for each resource we determine its coordinates on the grid.

The remainder of the steps of the Algorithm 1 form N' versions of the scheduler and binder. For each version, we do both scheduling and binding using a maximally constrained minimally constraining heuristic described as follows. In Step 8, we select a center resource for each module type that is the most frequently used resource in the pertinent version. In line 10, we compute a priority pair (p_r, n_r) for each module. p_r is proportional to the distance from the center resource and n_r is the number of neighbors on the grid. The center resource of each type gets the highest priority ($p_r = 1$) to be used in a control step (maximally constrained). Further resources with larger number of neighbors have lower priority. We bind the operations to resources in order of priority. To break the ties among the resources with the same pr value, we use the number of neighbors of the resource n_r, where the resource with the smaller value of n_r has a higher priority (minimally constraining).

6.5 Rotational N-version method

Thermal model. The compact thermal model that we use to predict the temperature rise due to using a certain version. The thermal energy of each module can be estimated by considering three different parameters: the power consumed at that module, the activity of the module in the schedule, and the exchange of energy with each of its neighboring modules.

We model the stationary state of the IC, where its produced thermal energy is equal to the energy that it transfers to the environment. The two key assumptions are: (i) the chip is small relative to the environment and therefore, the environment does not change its temperature due to the heat conducted by the IC; and (ii) the rate of thermal change is much slower than the chip's clock frequency f_c and thus, one can consider the usage rate of one module in a certain scheduling round to be a good approximation of the cumulative impact of that module on the temperature.

The compact model is based on the Fourier conduction equations with constant thermal properties that is known to be a linear elliptic boundary value problem. Elliptic boundary value problems are a class of problems which do not involve the time variable, and instead only depend on space variables [81]. The thermal energy of the module at coordinate (i,j) is denoted by $Q_{i,j}$ respectively. Based on the Fourier conduction equations, the Thermal energy ($Q_{i,j}$) of the module (i,j) can be written as:

$$\begin{aligned} Q_{i,j} &= k_{Si/env} * A_{Si/env} * (T_{i,j} - T_{env}) \\ &+ k_{Si/Si} * A_{Si/Si} * (T_{i,j} - T_{i,j-1}) \\ &+ k_{Si/Si} * A_{Si/Si} * (T_{i,j} - T_{i,j+1}) \\ &+ k_{Si/Si} * A_{Si/Si} * (T_{i,j} - T_{i-1,j}) \\ &+ k_{Si/Si} * A_{Si/Si} * (T_{i,j} - T_{i+1,j}). \end{aligned} \tag{6.1}$$

Linear program for selection of the N-versions and their durations. We describe how we generate the rotational schedule using a linear program. The linear program takes as input the N'-versions that are constructed in the previous section. Next it selects the N-versions out of N' for embedding in the chip. The linear program also assigns a duration to each of the N selected final schedules.

The linear program is shown in Algorithm 2. The objective function (shown in Step 1) is to minimize the maximum temperature on the chip. This is followed by four types of constraints. Step 2 shows the first constraint type that represents local Newton heat laws. $Q_{i,j}$ and $T_{i,j}$ are variables representing the thermal energy generated by and the temperature of the resource at the coordinate (i,j) respectively.

The second type of constraints are shown in Step 3. This constraint represents the local thermal energy generation which is a function of the schedules. $P_{k,i,j}$ is a constant representing the average power generated by the resource at (i,j) in version k. p_k is a variable denoting the fraction of time this schedule is to be used. Step 4 shows the global constraints for the maximum temperature on the grid, where each resource temperature must not exceed the maximum temperature. Finally step 5, shows the total activity constraint that sets the sum of all the fractions p_k to 1. To have a low-overhead implementation for the rotational N-version method, we construct the FSM (F_r) of the rotational schedule from the FSM (F_1) of one of the versions. F_r has $logN$ extra inputs added to F_1 that are used as the key to select a version (I_{key}). The number of outputs of F_r are the same as the number of outputs of F_1. The construction is done similar to the method described in [82].

6.6 Experimental evaluations

We evaluate the rotational N-version method on different benchmarks from HYPER extracted from [83]. The benchmark names are shown in the second column of Table 6.1. The benchmarks lee, arai, and dir are 8 point fast discrete cosine algorithms with sharply different structures. Specifically, lee is Lee's recursive sparse matrix factorization algorithm, arai is Arai-Agui-Nakajima algorithm, and dir is the direct generic definition of DCT-I algorithm. feig is Feig's fast 2D 8x8 DCT with provably minimal number of multiplications. The benchmarks aircraft and honda are two industrial strength mechanical controllers.

The CDFG of the benchmarks are extracted in graphviz format. Matlab is used to read the CDFG, implement the multiple-version scheduling and binding and for the LP problem formulation and solving. We also generate HotSpot floorplan and power trace files of the schedules to evaluate the temperatures. We use the default values in HotSpot.

Table 6.1 shows the improvement in the maximum temperature when we use the minimum resources computed by the list scheduler. The comparison is made to the scheduling and binding method that most balances the usage of the modules. This balanced version is found by giving equal priorities to all the modules in Algorithm 1. The first column in the table represent the benchmark name. The second and third columns show the number of ALU operations (*aop*) Multiplication operations (*mop*) in the CDFG of the benchmark. The fourth and fifth columns demonstrate the lower bound on the number of ALUs ($A\#$) and Multipliers ($M\#$). The number of schedules selected by the linear program ($N\#$) is shown in the sixth column. The maximum temperature for the balanced scheduling and binding method (T_b), the maximum temperature in the rotational N-version (T_r) method (both in °C), as well as percentage improvement (I) in the Temperature are presented in the last three columns respectively. Temperatures are computed using HotSpot. The maximum improvement is above 11.7% and on the average the improvement is 4.9%.

Table 6.1 : Max temp. improvement (min resources).

Name	**aop**	**mop**	**A#**	**M#**	**N#**	**T_b(C)**	**T_r(C)**	**I%**
arai	**39**	**5**	8	1	2	69.52	61.41	11.7
lee	**37**	**20**	4	4	2	64.99	62.57	3.7
honda	**70**	**34**	12	8	2	67.56	66.22	2.0
dir	**77**	**47**	11	11	3	64.22	61.45	4.3
aircraft	**147**	**127**	15	16	4	65.67	60.73	7.5
feig_dct	**505**	**78**	48	18	8	74.75	71.67	4.1

We next evaluate the performance of the rotational N-version method compared with the balanced version by adding resources. Table 6.2 shows the maximum temperature improvement when we use 10% extra resources. The first column shows the benchmark name. The second and third columns show the number of ALUs and Multipliers used in each benchmark. The fourth column shows the number of schedules produced by the linear program. The last three columns show the maximum temperatures in the balanced schedule and the rotational schedule in degree Celsius and the percentage improvement. The maximum improvement is about 19%, while the average is 7.4%.

Table 6.2 : Max temp. improvement (add resources).

Name	**A#**	**M#**	**N#**	**T_b(C)**	**T_r(C)**	**I(%)**
arai	9	2	2	69.61	61.4	11.8
lee	5	5	3	63.57	61.22	3.7
honda	14	9	3	67.73	61.62	9.0
dir	13	13	2	68.25	62.05	9.1
aircraft	17	18	6	71.48	57.9	19.0
feig_dct	53	20	3	74.32	72.27	2.8

To study the overhead of embedding multiple schedules, we use ABC synthesis tool to estimate the area overhead of a single schedule and the rotational schedule. The area overhead for Table 6.1 is shown in Table 6.3. The first column shows the benchmark name. The second column shows the area in terms of the number of literals for the chip using a single schedule denoted by *orig*. The third column represents the area for the new schedule denoted by *new*. The maximum overhead is less than 5%. Note that benchmark 9 has zero overhead because it uses only one schedule. On the average the overhead is 1.3%. This shows the very low overhead of rotating among the versions. Note that the power overhead of the N-version method is proportional to its area overhead. The timing overhead is zero since all of the versions satisfy the timing constraint.

Table 6.3 : Area overhead of the N-versions in Table 6.1.

Name	**Orig (lit)**	**New (lit)**	**%**
arai	99738	99738	0.2
lee	83369	83519	0.2
honda	211627	213597	0.9
dir	229201	239662	0.6
aircraft	322173	338011	4.9
feig_dct	712040	738329	3.7

6.7 Conclusion

We introduced a new temperature-aware scheduling and resource allocation method that combines N different versions of scheduling and binding. The combination was done by rotating between the N versions and by running each of the versions for a certain duration of time. Maximally constrained minimally constraining scheduling method was used for the efficient design and implementation of the multiple versions. We presented a linear programming formulation of the scheduling rotation that selects N out of many versions and determines the duration of each version. Evaluation of the method on standard benchmarks showed the low overhead of implementing the multiple versions in one design, and selection of the best value for N. Our experimental results shows that using the new method, an average of about 5% reduction in the peak temperature is obtained on the benchmarks in comparison with the scheduling and binding method where the usage of all resources are balanced.

Algorithm 1 Generation of N'-versions of the schedules.

Input: $CDFG, N'$

Output: S'_n

```
1begin
2       Compute $Rlb_t$- using list scheduling;
3       Choose $R_t$ such that $R_t >= Rlb_t$;
4       Generate layout grid for $R_t$;
5       Compute co-ordinates $(x_t, y_t)$ for $R_t$ on the grid;
6       for $k = 1 : N'$
7           for each resource type $t$;
8               Choose a center resource $n_{c_t}$;
9               for each resource $r \in R_t$;
10                  compute the priority pair $(p_r, n_r)$
11          $time = 1$;
12          $ready$ = operations without predecessors;
13          while There exists an unscheduled operation;
14          begin
15              Schedule a maximum subset of $ready$
                Operations with least mobility are scheduled,
                Resources with min $(p_i, n_i)$ are bound;
16              $time = time + 1$;
17              Add operations whose predecessors
                are done to $ready$;
18          end
19      end
```

Algorithm 2 The linear program to generate the rotational schedule.

```
Input: grid, N'-versions
Output: $S_r$
1       Objective function
        min $T_{max}$;
2       Constraint type 1
        for each resource $r_{i,j}$ on grid
            Satisfy equation 6.1;
3       Constraint type 2
        for each resource $r_{i,j}$ on grid
            $Q_{i,j} = p_1 P_{1,i,j} + p_2 P_{2,i,j} + ... + p'_N P_{N',i,j}$;
4       Constraint type 3
        for each resource $r_{i,j}$ on grid
            $T_i, j <= T_{max}$;
5       Constraint type 4
        $p_1 + p_2 + ... + p'_N = 1$;
```

Chapter 7

Conclusion

In this thesis we have presented the N-variant hardware design methodology. The methodology works by embedding multiple variants of parts of the same design in one IC. The variants are designed to add diversity and flexibility to the otherwise inflexible ASIC. The variants are generated using two different methods: state space and scheduling diversity. Both methods manipulate the control part of the design.

The techniques take advantage of the low overhead of the control circuit of modern designs and thus provide diversity while incuring low area and power overheads by manipulating just the control parts. They can be easily integrated in the current tool flow. Moreover, all the transformations presented can be implemented automatically with minimal input from the designer, while fabrication of the resulting designs is identical to the fabrication of standard ASICs. Thus, these techniques are both inexpensive and easy to implement and do not require a high learning curve to use. These properties make them extremely practical. In what follows, we summarize our contributions and then we present possible future directions for research in this area.

7.1 Contributions

The contributions can be summarized as follows:

- **N-variant hardware design methodology** : We introduced the idea of N-variant hardware design and presented different ways of implementing it. The practicality and usefulness of the methodology was highlighted by illustrating different applications of the method. Proof-of-concept implementations on standard benchmarks showed the method to be not only easy to implement and integrate in the ASIC design flow, but also indicated low overhead of the method in terms of area, power, and delay.

- **Active hardware metering for IC protection**: We presented the first active hardware metering method for IC protection. The method adjusts the asymmetry in the relationship between the IC designer and the fabrication facility by giving the designer control over her chips once fabricated. Taking advantage of manufacturing variability, each IC is uniquely locked and only the designer can compute the key to unlock it. In this application, state space diversity was implemented by adding exponentially many states and generating variants with different starting states on the state transition graph of the design. The overhead of the design was estimated on standard benchmarks and was shown to be low in terms of area, power, and delay.

- **Active hardware metering for IP protection**: In this application, we presented a hierarchical organization that takes advantage of the N-variant methodology to protect not only the IC design owner, but also the IP owners whose IPs are reused by the IC designer. Thus, the method facilitates design reuse of hardware IPs. In this application state space diversity was implemented by replicating a few states and transitions of the design FSM. Each variant used a different subset of the states and transitions. Implementation on standard benchmarks showed the method to have low overhead. We also discussed the security of the method in terms of attacks and countermeasures.

- **Temperature control using N-variant Design**: This application used scheduling diversity to implement different variants of the controller. Rotation between different variants was used to control the peak temperature of the IC. Simulations showed that we could tune the peak temperature and reduce it using the rotational schedule. In addition, estimates of the area, power, and delay overheads on standard benchmarks are shown to be low.

7.2 Future Directions

- **Applications**: The applications of the N-variant hardware design methodology are endless. N-variant designs could be used for power tuning, post-silicon optimization, and fault tolerance. In addition, practical implementation of the method on real applications instead of standard benchmarks is important to validate the method.

- **Methods for adding diversity**: More methods for adding diversity should be explored. At a low level, a new diversity dimension could use different types of gates for different variants, while at a high level different algorithms can be explored to implement different variants.

- **Software**: All the methods dealt only with hardware. However, it would be interesting to study the applicability of these methods on software or a combination of hardware/software.

- **Variant control**: All the applications presented used a static hardware-based method for variant control. However, varying the mechanism for controlling the variants may open up the scope for more interesting applications. For instance, variants could be controlled by software, or dynamically using feedback from sensors. This should enable new methods for the adaptive control and tuning of computer systems.

Bibliography

[1] J. Hennessy and D. Patterson, *Computer architecture: a quantitative approach.* Morgan Kaufmann Publishers, 1996.

[2] Y. Alkabani and F. Koushanfar, "N-variant IC design: methodology and applications," in *Design Automation Conference (DAC)*, pp. 546–551, 2008.

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

[4] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ics for piracy prevention and digital right management," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2007.

[5] Y. Alkabani and F. Koushanfar, "Active control and digital rights management of integrated circuit IP cores," in *ACM/IEEE International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES)*, 2008.

[6] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-variant systems: A secretless framework for security through diversity," in *USENIX Security Symposium*, pp. 105–120, 2007.

[7] D. Holland, A. Lim, and M. Seltzer, "An architecture a day keeps the hacker

away," *SIGARCH Computer Architecture News*, vol. 33, no. 1, pp. 34–41, 2005.

[8] A. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101–1117, 2001.

[9] L. Yuan and G. Qu, "Information hiding in finite state machine," in *Information Hiding Conference (IH)*, pp. 340–354, 2004.

[10] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 3, pp. 461–491, 2004.

[11] S. Pontarelli, G. Cardarilli, A. Malvoni, M. Ottavi, M. Re, and A. Salsano, "System-on-chip oriented fault-tolerant sequential systemsimplementation methodology," in *Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 455–460, 2001.

[12] J. Kim and M. Newborn, "The simplification of sequential machines with input restrictions," *IEEE Trans. on Computers*, vol. 21, no. 12, pp. 1440–1443, 1972.

[13] S. Devadas, "Design automation conference (dac)," in *DAC*, pp. 270–276, 1989.

[14] Y. Watanabe and R. Brayton, "The maximum set of permissible behaviors for FSM networks," in *ICCAD*, pp. 316–320, 1993.

[15] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "A fully implicit algorithm for exact state minimization," in *Design Automation Conference (DAC)*, pp. 684–690, 1994.

[16] J. Rho, G. Hachtel, F. Somenzi, and R. Jacoby, "Exact and heuristic algorithms for the minimization of incompletelyspecified state machines," *IEEE Trans. on CAD*, vol. 13, no. 2, pp. 167–177, 1994.

[17] K. Knowlton, "A combination hardware-software debugging system," *IEEE Transactions on Computers*, vol. 17, no. 1, pp. 81–86, 1968.

[18] B. Randell, "System structure for software fault tolerance," *Software Engineering*, vol. 1, no. 2, pp. 221–232, 1975.

[19] A. Avizienis, "The N-version approach to fault-tolerant software," *IEEE Transactions on Software Engineering*, vol. 11, no. 12, pp. 1491–1501, 1985.

[20] S. Forrest, A. Somayaji, and D. Ackley, "Building diverse computer systems," in *Hot Topics in Operating Systems (HotOS)*, p. 67, 1997.

[21] N. Couture and K. Kent, "Periodic licensing of FPGA based intellectual property," in *Field Programmable Technology (FPT)*, pp. 357–360, 2006.

[22] VSI Report, "Vsi alliance - intellectual property protection development working group, "white paper: The value and management of intellectual assets". http://vsi.org/documents/datasheets/toc_ippwp210.pdf," 2002.

[23] DSB Report, "Defense science board (DSB) study on high performance microchip supply. http://www.acq.osd.mil/dsb/reports/2005-02-hpms_report_final.pdf," 2005.

[24] K. Lofstrom, W. Daasch, and D. Taylor, "IC identification circuits using device mismatch," in *International Solid State Circuits Conference (ISSCC)*, pp. 372–373, 2000.

[25] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in *Information Hiding Workshop (IHW)*, pp. 81–95, 2001.

[26] S. Maeda, H. Kuriyama, T. Ipposhi, S. Maegawa, Y. Inoue, M. Inuishi, N. Kotani, and T. Nishimura, "An artificial fingerprint device (AFD): a study of identification number applications utilizing characteristics variation of polycrystalline silicon TFTs," *IEEE Trans. Electron Devices*, vol. 50, no. 6, pp. 1451–1458, 2003.

[27] S. Roy and A. Asenov, "Where do the dopants go?," *Science*, vol. 309, no. 5733, pp. 388–390, 2005.

[28] A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for* VLSI*: Timing and Power.* Series on Integrated Circuits and Systems, Springer, 2005.

[29] K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, S. Nassif, E. Nowak, D. Pearson, and N. Rohrer, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4/5, pp. 433–450, 2006.

[30] Y. Su, J. Holleman, and B. Otis, "A 1.6J/bit stable chip ID generating circuit using process variations," in *International Solid State Circuits Conference (ISSCC)*, pp. 406–411, 2007.

[31] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, *Concurrency and Computation: Practice and Experience*, vol. 16, ch. Identification and authentication of integrated circuits, pp. 1077–1098. John Wiley & Sons, 2004.

[32] J. Lee, L. Daihyun, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium of VLSI Circuits*, pp. 176–179, 2004.

[33] G. Suh, C. O'Donnell, I. Sachdev, and S. Devadas, "Design and implementation of the aegis single-chip secure processor using physical random functions," in *International Symposium on Computer Architecture (ISCA)*, pp. 25–36, 2005.

[34] F. Koushanfar and G. Qu, "Hardware metering," in *Design Automation Conference (DAC)*, pp. 490–493, 2001.

[35] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *International Conference on Computer Aided Design (ICCAD)*, pp. 194–198, 1998.

[36] 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," in *Design Automation Conference (DAC)*, pp. 776–781, 1998.

[37] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 35, no. 3, pp. 434–440, 2000.

[38] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Design.* Kluwer, 2003.

[39] D. Kirovski and M. Potkonjak, "Local watermarks: Methodology and application on behavioral synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1277–1283, 2003.

[40] J. Wong, R. Majumdar, and M. Potkonjak, "Fair watermarking using combinatorial isolation lemmas," *IEEE Trans. CAD*, vol. 23, no. 11, pp. 1566–1574, 2004.

[41] F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Design Automation of Electronic Systems*, vol. 10, no. 3, pp. 523–545, 2005.

[42] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Fingerprinting digital circuits on programmable hardware," in *Information Hiding Workshop (IHW)*, pp. 16–32, 1998.

[43] F. Koeune and F. Standaert, "A tutorial on physical security and side-channel attacks," in *Foundations of Security Analysis and Design (FOSAD)*, pp. 78–108, 2004.

[44] R. Anderson, *Security Engineering: A guide to building dependable distributed systems.* John Wiley and Sons, 2001.

[45] D. Abraham, G. Dolan, G. Double, and J. Stevens, "Transaction security system," *IBM Systems Journal*, vol. 30, no. 2, pp. 206–229, 1991.

[46] R. Anderson and M. Kuhn, "Tamper resistance - a cautionary note," in *USENIX Workshop on Electronic Commerce*, pp. 1–11, 1996.

[47] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, "Optimizing power using transformations," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12–31, 1995.

[48] H. Savoj and R. Brayton, "On the optimization power of retiming and resynthesis transformations," in *Design Automation Conference (DAC)*, pp. 297–301, 1990.

[49] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for* VLSI *Synthesis.* Kluwer Academic Publishers, 1984.

[50] F. Brgles, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *International Symposium of Circuits and Systems*, pp. 1929–1934, 1989.

[51] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Tech. Rep. UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

[52] F. Koushanfar and M. Potkonjak, "CAD-based security, cryptography, and digital rights management," in *Design Automation Conference (DAC)*, 2007.

[53] M. Jacome and H. Peixoto, "A survey of digital design reuse," *IEEE Design and Test of Computers*, vol. 18, no. 3, pp. 98–107, 2001.

[54] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Computer and Communications Security (CCS)*, pp. 148–160, 2002.

[55] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *International Test Conference (ITC)*, p. in press, 2008.

[56] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUF," in *International conference on computer-aided design (ICCAD)*, 2008.

[57] ""Design and reuse website", http://www.us.design-reuse.com/."

[58] J. Rowson and A. Sangiovanni-Vincentelli, "Interface-based design," in *Design Automation Conference (DAC)*, pp. 178–183, 1997.

[59] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-art implementations and attacks," *IEEE Trans. on Embedded Computing Systems*, vol. 3, no. 3, pp. 534–574, 2004.

[60] S. Trimberger, "Trusted design in FPGAs," in *Design Automation Conference*, pp. 5–8, 2007.

[61] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Delay-based circuit authentication and applications," in *ACM symposium on Applied computing*, pp. 294–301, 2003.

[62] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007.

[63] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: A nondestructive hidden characteristics extraction approach," in *Information Hiding Conference (IH)*, 2008.

[64] C. Marsh and T. Kean, "A security tagging scheme for asic designs and intellectual property cores," *Design & Reuse*, January 2007.

[65] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending piracy of integrated circuits," in *Design Automation and Test in Europe (DATE)*, 2008.

[66] Certicom Report, "http://www.certicom.com/index.php/protect-your-silicon-ip-from-the-gray-market?action=sol,silicon," 2008.

[67] ""The international technology roadmap for semiconductors (itrs)", http://www.itrs.net/."

[68] Y. Cheng, P. Raha, C. Teng, E. Rosenbaum, and S. Kang, "ILLIADS-T: an electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips," *IEEE Trans. on CAD*, vol. 17, no. 8, pp. 668–681, 1998.

[69] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on VLSI*, vol. 14, no. 5, pp. 501–513, 2006.

[70] K. Banerjee, S.-C. Lin, and V. Wason, "Leakage and variation aware thermal management of nanometer scale ICs," in *IMAPS-Workshop*, 2004.

[71] C. Chu and D. Wong, "A matrix synthesis approach to thermal placement," in *ISPD*, pp. 163–168, 1997.

[72] C. Tsai and S. Kang, "Standard cell placement for even on-chip thermal distribution," in *ISPD*, pp. 179–184, 1999.

[73] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *Journal of Instruction-Level Parallelism*, no. 7, 2005.

[74] R. Mukherjee and S. Memik, "An integrated approach to thermal management in high-level synthesis," *IEEE Trans. on VLSI*, vol. 14, no. 11, pp. 1165–1174, 2006.

[75] M. Ni and S. Memik, "Thermal-induced leakage power optimization by redundant resource allocation," in *ICCAD*, pp. 297–302, 2006.

[76] P. Lim and T. Kim, "Thermal-aware high-level synthesis based on network flow method," in *CODES+ISSS*, pp. 124–129, 2006.

[77] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Trans. on CAD*, vol. 24, no. 7, pp. 1042–1053, 2005.

[78] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. on of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1702–1714, 1999.

[79] L. Shang, R. Dick, and N. Jha, *High-Level Synthesis Algorithms*, ch. High-Level Synthesis Algorithms for Power and Temperature Minimization, pp. 285–297. Springer, 2008.

[80] S. Zhang and K. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *ICCAD*, pp. 281–288, 2007.

[81] L. Evans, *Partial Differential Equations.* American Mathematical Society, 1998.

[82] Y. Alkabani and F. Koushanfar, "N-variant IC design: Methodology and applications," in *Design Automation Conference (DAC)*, pp. 546–551, 2008.

[83] K. Rao and P. Yip, *Discrete Cosine Transform.* Academic Press, 1990.