

Chapter 2

VLSI Test and Hardware Security

Background for Hardware Obfuscation

Fareena Saqib and Jim Plusquellic

2.1 Introduction

Hardware obfuscation is a technique to conceal the design from malicious insider and outsider adversaries. Obfuscation techniques transform the original design such that the obfuscated version is functionally equivalent to the original design, but it does not reveal the design details and is much harder to reverse-engineer [1]. As discussed earlier in Chap. 1, the business model of distributed and outsourced design, integration, manufacturing, packaging, and distribution channels creates challenges such as intellectual property (IP) piracy, reverse engineering of the netlist from GDSII, integrated circuit (IC) cloning, and counterfeiting opportunities.

Nanometer-sized integrated circuit feature sizes and increased gate density per wafer have been made possible with the advancements in photolithography techniques. However, this has driven the cost and maintenance of fabrication facilities into the billions of dollars, making the business model difficult to justify and sustain. Consequently, many major companies have become fabless and instead outsource their designs to offshore foundries as a cost-effective alternative to owning and operating their own fabs. Unfortunately, the horizontal dissemination of the design process to companies all over the world decreases the trustworthiness and increases the security risks of the design process [2, 3].

This chapter overviews the traditional design flow of integrated circuits and assesses processes in terms of how much information is revealed to aid in reverse engineering the design. We survey the proposed schemes that are designed to enhance the security properties of traditional verification and testing mechanisms to make designs resilient to attacks. This chapter also investigates IP protection schemes that

F. Saqib (✉)
Florida Institute of Technology, Melbourne, FL, USA
e-mail: fsaqib@fit.edu

J. Plusquellic
University of New Mexico, Albuquerque, NM, USA

are designed to prevent illegal modifications and piracy for system-on-chip (SoC) IP reuse-based design flows. This problem is challenging because IP can be distributed as soft (RTL level), firm (netlist level), or hard (GDSII level) and is usually transparent at system design level, in manufacturing facility, and in the distribution chain, making it susceptible to security and privacy attacks. The objective of hardware obfuscation is to make it difficult for an adversary to reverse-engineer the functionality at any level of abstraction throughout the design process.

Threat Models:

IC piracy, cloning, counterfeiting, and sabotage have become major security concerns under the current business model of IP reuse and offshore manufacturing. The following provides a partial list of attack vectors open to an adversary:

- (1) Reverse engineering: GDSII-to-netlist reverse engineering enables the adversary to steal and reproduce the IP.
- (2) Clones: An attacker in the system design flow can steal the IP or IC and make exact clones or, with a few modifications, claim the ownership and make illegal copies.
- (3) Overbuilding: Building more copies of the IC than requested by the customer is referred as overbuilding. Overbuilt ICs can be sold on the black market. Without specialized metering techniques, preventing overbuilding is a challenge.
- (4) Counterfeit chips: Counterfeit chips are intended to deceptively represent an authentic component and can be created from recycled chips or from cloning [4].
- (5) Trojan detection insertion: After reverse engineering the design, the adversary can insert hardware Trojans in a set of counterfeit clones. Hardware Trojans are hidden malicious circuits that can be designed to allow activation through backdoors during the fielded operation of the chip. Activation can involve leaking sensitive information or causing the chip and system to fail catastrophically.

This chapter is organized as follows: Sect. 2.2 introduces the VLSI verification and test concepts and discusses the vulnerabilities, attacks, and countermeasures. Section 2.3 describes the obfuscation techniques that can be integrated into the design flow to make the design more resilient to reverse engineering and summarizes the evaluation metrics for these techniques. Section 2.4 covers the review of nonvolatile memory and emerging technologies and discusses the associated vulnerabilities of key management on nonvolatile memories (NVMs). Section 2.5 introduces the hardware-based cryptographic primitives, physical unclonable functions (PUFs), and true random number generator (TRNG) and their use in hardware obfuscation techniques to improve the resilience against reverse engineering.

2.2 VLSI Verification and VLSI Test Concepts

Very large-scale integration (VLSI) verification is a presilicon procedure to verify the design before fabrication. Random test vectors and formal verification techniques are used to verify design behavior and coverage of generated test vectors. Satisfiability

(SAT) solvers are used in formal verification to find design issues presilicon. Several SAT solver algorithms have been integrated into the electronic design automation (EDA) tools.

In contrast, VLSI testing is applied post-silicon to ensure high quality and reliability in the shipped IC products and to find design problems that affect yield early. The increasing level of complexity and smaller geometries used in modern IC fabrication introduces new failure mechanisms that act to reduce the yield and the quality of shipped products. VLSI testing is critically important to screening defective products, with the ultimate goal of reaching zero defects. VLSI testing also provides important feedback for accelerating product yield ramps and has a direct impact on profitability.

2.2.1 Satisfiability (SAT) Problem

Satisfiability is defined as a condition of boolean expression evaluating to be true for a set of logical values of the variables. Outputs of combinational logic can be expressed as boolean expressions, constituting conjunctions (and) of disjunctive (or) clauses and variables in the form of conjunctive normal form (CNF).

An example of function in the CNF form is as follows:

$$F = (a \vee b \vee c) \wedge (a' \vee c \vee d) \wedge (b' \vee d') \quad (2.1)$$

where a, b, c, and d are the variables that can be '1' or '0'.

The decision problem for SAT, to find a satisfying assignment that makes the function true, is a nondeterministic polynomial (NP) problem. For example, for n variables, 2^n boolean combinations of input variables are examined. Each SAT formula has a polynomial time verifier that takes an input string, a zero, or one assignment for all the variables and outputs a true or false evaluation for the provided inputs.

The SAT problem has exponential complexity in the worst case, but given the importance of the algorithm in CAD, researchers have developed many types of efficient heuristically SAT solvers that provide near-optimal solutions. These SAT solvers have many applications in the electronic design automation (EDA) in verification as well as in the synthesis. The SAT solver algorithms are categorized as conflict-driven clause learning and stochastic local search algorithms. These algorithms have been developed to automatically solve the instances/combinations with large number of variables and clauses. Recent work in the development of efficient SAT solvers includes GRASP [5], Satz [6], and Chaff [7]. These algorithms use SAT solvers in formal or semiformal verification methods.

SAT solvers can also be used by malicious attackers to circumvent logic encryption-based hardware obfuscation by applying SAT-based algorithms to derive the keys [8]. The technique utilizes the approach of iteratively applying input pat-

terns on a set of selected inputs and identifying distinguishing inputs, for which the functions become unsatisfiable. This approach of testing key combinations has proven to weaken the security of hardware obfuscation, and research has focused on countermeasures designed to instill worst-case (exponential) behavior in SAT algorithms.

2.2.2 *Equivalence of Circuits*

Equivalence checking is one approach to functional verification. Equivalence checking is performed at different stages of design flow to verify the functional equivalence of combinational and sequential logic. Equivalence checking takes two descriptions of the design that are structurally different and verifies whether their behavior is functionally equivalent. The designs are compared using formal methods and simulation techniques. Formal methods such as binary decision diagram (BDD) and SAT-based are applied to the compare points in both the reference design and the implemented design, and functional equivalence is verified using simulations. Traditional equivalence checking utilizes the logic cones by analyzing the compare points. Logic cones are generic attributes of digital circuits consisting of reconvergent segments that fan-in to a common output. The input and outputs of cones are connected to the primary inputs, registers, or primary outputs that are also referred as compare points.

The implemented design is verified to prove or disprove the functional equivalence once all the compare points are verified. Several commercial tools such as Cadence Conformal equivalence checker and Synopsys Formality are equivalence checking tools. These tools use the netlist generated from Genus (Cadence) or Design Compiler (Synopsys) synthesis generated netlist to compare with the RTL design description using mathematical models.

Additional research is needed that investigates the equivalence checking in the design flows that implement obfuscation techniques. Equivalence checking of the reference design with the obfuscated netlists shown in Fig. 2.1 is further discussed in another chapter.

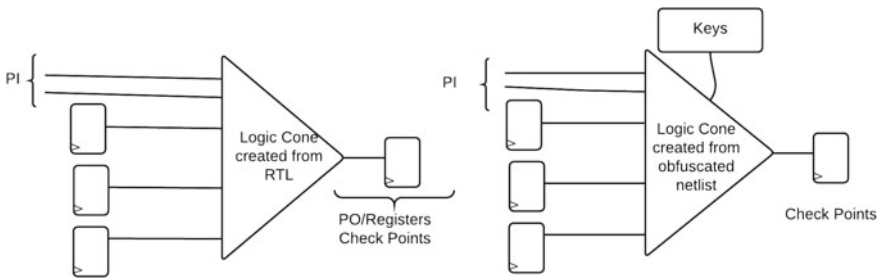


Fig. 2.1 Logic equivalence of reference and obfuscated design

2.2.3 Types of Testing: Functional Testing and Structural Testing

2.2.3.1 Functional Testing

Functional test verifies that the chip performs the correct operations, i.e., can the chip run the Windows operating system or carry out a matrix inverse software operation. ATPG can alternatively and/or additionally be used to generate functional test vectors. For example, vectors can be generated to test the ‘critical paths,’ which are the longest paths in the chip, and test the chip under worst-case power conditions. Therefore, the roles of functional testing also include timing and power verification.

2.2.3.2 Structural Testing

Structural testing refers to techniques that are based on fault models as discussed above. The goal is to check the integrity of the structural characteristics of the chip, i.e., its individual wires and logic gate functions. SSF tests verify that circuit nodes are not shorted to VDD or VSS, while transition and path delay tests verify that the logic gates and selected paths are able to propagate transitions to capture points (flip-flops and POs) using the functional (‘at-speed’) clock frequency. As indicated, ATPG is used to derive the test vectors and automatic test equipment (ATE) is used to apply them. Note that both functional testing and structural testing are constrained by the economics of testing, i.e., a great deal of effort is made to determine the smallest set of test vectors that meets the coverage requirements. This is true because manufacturing tests are applied to every chip, and therefore, to be economical, the test time per chip must be as small as possible.

2.2.4 Fault Modeling

Physical defects can occur in the IC during the manufacturing process, such as interconnect defects or packaging defects, gate–oxide shorts, metal trace bridges, open vias, and shorts to power or ground. Fault modeling is a mechanism to abstract and simplify all the possible ways defects can cause a malfunction in a chip. The most common fault models are single stuck-at fault (SSF), and transition and path delay fault models. The SSF models have also been proposed as a mechanism to strengthen hardware obfuscation techniques, as discussed below. The SSF model represents each defect as a single gate-level pin or net shorted to VDD or VSS. The terms ‘stuck-at-1 (SA1)’ and ‘stuck-at-0 (SA0)’ are used to represent these conditions. The SSF model assumes that only one fault (or no fault) exists in each chip. Figure 2.2a shows 6 gate stuck-at faults for a two-input NAND gate, and Fig. 2.2b shows one instance of SA0 fault in combinational logic.

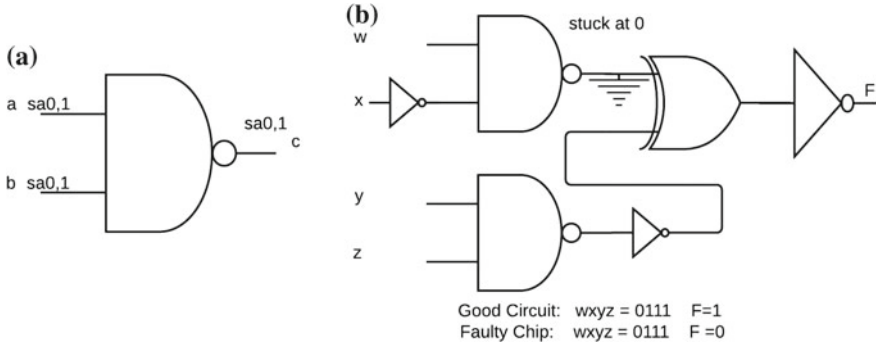


Fig. 2.2 **a** NAND gate inputs and output stuck-at model. **b** Combinational stuck-at fault

A combinational SSF is detected by determining the primary inputs (PIs) input assignments that introduce the appropriate state on the target gate inputs while simultaneously ensuring that the target gate output is observable on one or more primary outputs (POs). A stuck-at fault test determines whether the target node is SA0 or SA1 but also implicitly tests all gate inputs along the path for fault conditions.

The SSF model verifies the structural integrity and truth table description of combinational logic but does not verify whether the chip meets its timing specification. Separate fault models and sets of test vector sequences are required to verify timing. The transition and delay fault models target timing-related defects that cause logic transitions to take longer than expected to propagate through gates and along paths in the chip, i.e., conditions that cause the chip to violate timing constraints. Figure 2.3 shows the examples of delay faults. Defects that affect the drive strength of the gate, transistor doping levels, metal capacitive loading, open vias, and/or resistive gate–oxide shorts can introduce transition and delay faults in the chip.

The delay fault can be represented as a single gate fault, interconnect fault, or path delay fault. A single gate delay fault models the defects that affect gate strength, transistor doping, etc., i.e., anything that causes the timing of the input value at a pin to be slow-to-rise or slow-to-fall. Interconnect delay faults model defects that introduce variations in wire width or cause signal degradation because of resistive shorts to other nodes. Path delay faults model distributed defects, i.e., defects that effect the delay of the entire path. Delay tests are timed two-vector sequences (unlike SSF tests) that are applied at a constant rate using the clock. Such tests are critical for ensuring quality in modern nanometer-sized technologies.

2.2.5 Fault Coverage

Testing methods are evaluated in terms of fault coverage, where it is represented as follows:

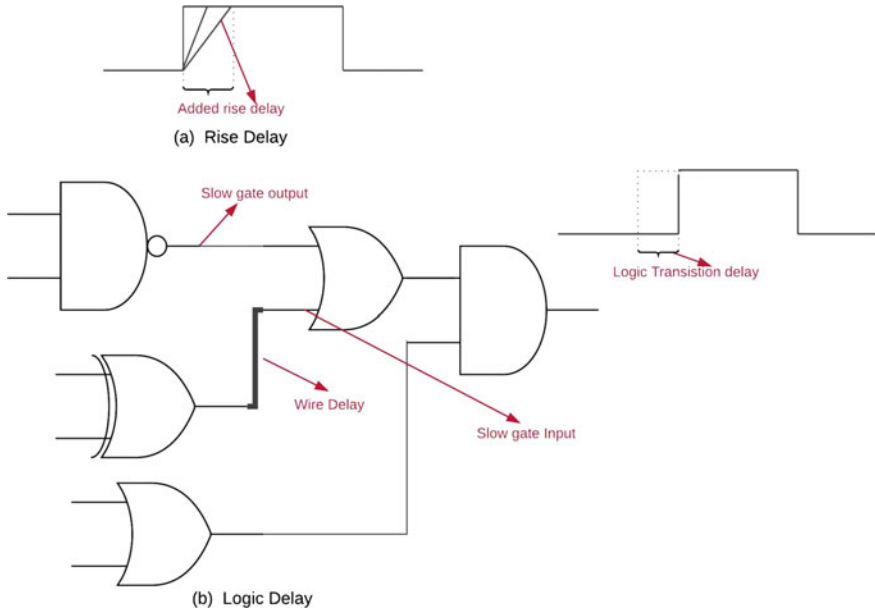


Fig. 2.3 Delay fault model

$$\text{Fault coverage} = \frac{\text{Total detected faults}}{\text{Total fault population}} \quad (2.2)$$

Fault coverage refers to fraction of faults under a given fault model that are covered by the test patterns. Fault coverage is typically computed and reported by automatic test pattern generation (ATPG) tools as these tools derive test patterns to test the faults. Ideally, the coverage should be 100%. Unfortunately, deriving test patterns is an NP-complete problem, and therefore, ATPG algorithms employ heuristics which, in many cases, are not able to find tests for all of the faults. Despite this limitation, test pattern generation using the SSF fault model is able to achieve high levels of coverage, typically 95–99%. It should be noted that fault coverage can be reported differently by different ATPG tools. For example, some tools eliminate the untestable faults from the fault population before applying the equation given above, while others do not. Fault coverage can also be used to identify difficult-to-test nodes and can therefore serve as a basis to guide design-for-testability (DFT) strategies.

2.2.6 Automatic Test Pattern Generation (ATPG)

As indicated above, ATPG is CAD software tool that automatically derives a set of test patterns for a specified list of faults using heuristic algorithms. The fault

model defines the nodes and/or paths in the chip that are the targets of ATPG. ATPG algorithms automatically derive a fault list from the netlist and fault model given as inputs. With the fault list available, a long, incremental process is started in which tests are derived that detect the faults. The faults detected by a test pattern are checked off in the fault list, and fault simulation is typically run to determine other faults that are ‘accidentally’ detected by the test pattern. Commercial vendors provide a variety of different runtime options and support for a fixed set of fault models, including SSF and transition and path delay fault models. ATPG and fault models can be leveraged in hardware obfuscation algorithms to produce strong keys, as discussed in Sect. 2.3. Figure 2.4 shows a typical ATPG flow.

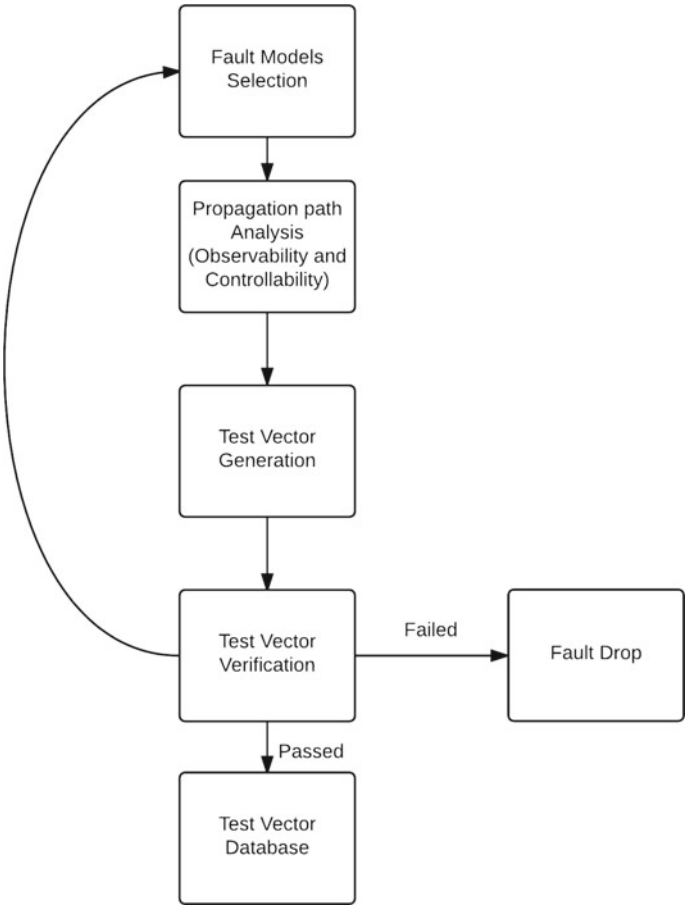


Fig. 2.4 ATPG flow

2.2.7 Testing Metrics: Controllability and Observability

As indicated above, test pattern generation is an NP-complete problem. As is true of many NP-complete problems, the task of generating a test pattern is doable in polynomial time for most of the faults in the fault list. Unfortunately, there are typically a small set of faults that elicit worst-case (exponential) time behavior in the ATPG algorithm. In response to this issue, the manufacturing test community developed a new set of algorithms that compute metrics for each of the faults in advance of ATPG that reflect the likely level of difficulty in generating test patterns for the faults. The metrics are probabilistic measures called controllability and observability. It should be noted that these algorithms also address an NP-complete problem and, like ATPG algorithms, employ heuristics. Unlike ATPG which can fail to find a test pattern for hard-to-test nodes, the heuristics used in algorithms that compute controllability and observability may produce inaccurate information for cases in which the task falls into a worst-case scenario.

Algorithms that compute controllability and observability (first coined by Rutman in 1972) produce numerical estimates regarding the difficulty of setting an internal node to a specific logic value and making an internal node observable on an output of the circuit. Several approaches to computing these testability measures have been proposed including SCOAP [9, 10], CAMELOT [11], TMEAS [12], COP [13], and PREDICT [14]. Testability analysis involves circuit topological analysis. For example, SCOAP traces through the design description and assigns controllability and observability weights to the nodes designated using the following six labels:

1. Combinational 0-controllability CC0(sig),
2. Combinational 1-controllability CC1(sig),
3. Combinational observability CO(sig),
4. Sequential 0-controllability SC0(sig),
5. Sequential 1-controllability SC1(sig), and
6. Sequential observability SO(sig).

The primary inputs (PI-sig) are all set to 1 for both combinational and sequential '0' and '1' controllabilities, i.e., $CC0(PI-sig) = 1$, $CC1(PI-sig) = 1$, $SC0(PI-sig) = 1$, and $SC1(PI-sig) = 1$. Combinational 0 and 1 controllabilities of internal nodes are calculated as the minimum number of combinational node assignments needed to justify a '0' or '1' on the output of a gate driving the node. Sequential 0 and 1 controllabilities on the other hand estimate the minimum number of sequential nodes that must be specified to set the internal node to '0' or '1'. Starting from the primary inputs to primary outputs, node weights are computed such that the circuit depth of the node is factored into the combinational controllability equations. The following rules are used to compute the output combinational controllability

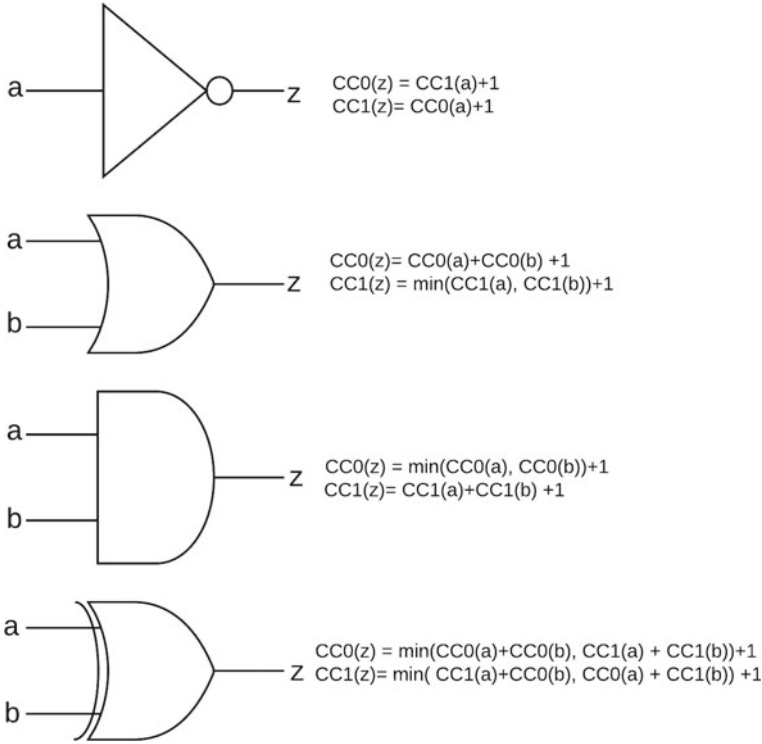


Fig. 2.5 SCOAP controllability calculation

$$Outputcontrollability = \begin{cases} \min(input\ controllability) + 1, & \text{if one input sets gate output} \\ \sum(input\ controllability) + 1, & \text{if all inputs sets gate output} \\ \min(controllabilities\ of\ input\ sets), & \text{if output is determined} \\ & \text{by multiple input sets, e.g., XOR} \end{cases} \quad (2.3)$$

Figure 2.5 describes the output controllability calculation for a set of standard cells. In contrast, for observability, all the primary outputs (PO-sig) are set to 0. A PO to PI traversal adds 1 to internal nodes as their depth, measured to a PO, is increased. For example, the observability of a gate input is computed using the gate's output observability and the controllabilities on its inputs as follows:

$$Input\ observability = output\ observability + \sum(controllabilities\ of\ all\ other\ input\ pins\ to\ noncontrollable\ value) + 1 \quad (2.4)$$

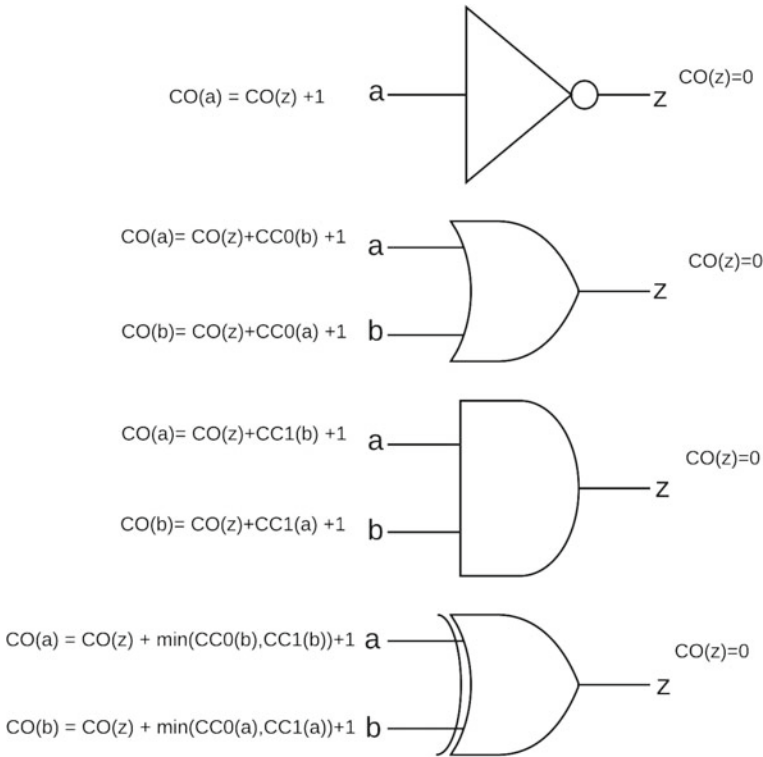


Fig. 2.6 SCOAP observability calculation

Figure 2.6 gives the equations for a common set of logic gates, including NOT, AND, OR, and XOR. The SC0 and SC1 calculations carried out for the sequential gates, e.g., the D-FF, take into account how many times the FF must be clocked to reach a particular output state of '0' or '1'.

The heuristics used in computing testability metrics provide the algorithms with linear runtime complexity. The analysis aids in the design process and can be used to provide guidance to ATPG algorithms as to which nodes are difficult to test. Designs for testability (DFT) techniques can be used in the design flow to add additional nodes as a means of improving the overall controllability and observability of circuit nodes. DFT methods are categorized as ad hoc methods and structured methods. In ad hoc methods, test structures are inserted into designs to target-specific problem areas on a case-by-case basis. Structured DFT methods, on the other hand, include standardized test structures such as scan and built-in self-test (BIST). DFT is typically carried out as an integral component of the design flow to ensure testing requirements can be met. For example, DFT can improve fault coverage and reduce the test generation time.

2.2.8 *Testing and Security*

The goal of manufacturing test is to detect defects that occur during fabrication or packaging before chips are shipped to customers or enter the supply chain. For security and trust, the goal is to provide a high assurance, trusted product. Unfortunately, detecting security and trust problem is much more difficult than providing high-quality, defect-free chips. This is true because the random nature of manufacturing defects makes it possible to find nearly all of them with the test vectors that provide high levels of fault coverage. The adversary for security and trust, on the other hand, will apply sophisticated techniques to break security systems and add malicious components (hardware Trojans) that are nearly impossible to activate and discover using current manufacturing test techniques.

Secondly, traditional approaches that are based on ‘security-by-obscurity,’ where internal design components are manipulated to impair reverse engineering attacks, are in fact partially defeated by DFT structures that assist with manufacturing test. Scan and other ad hoc DFT approaches that increase controllability and observability make it easier for adversaries to obtain internal design details in reverse engineering attacks. In subsequent sections, we discuss the security vulnerabilities introduced by DFT and the proposed countermeasures to allow test engineers to leverage them for finding defects, but simultaneously prevent adversaries from using them for reverse engineering attacks and as a ‘backdoor’ to break security mechanisms. DFT strategies that enable attack vectors include the following:

- (1) Scan,
- (2) Boundary scan, and
- (3) Built-in self-test (BIST)

2.2.8.1 **Scan-Based**

Scan Cells

Most DFT strategies, including scan insertion, are implemented during synthesis. Scan insertion replaces the flip-flops (FFs) in the design with a special scan-based FFs. Scan FFs add a special ‘test mode’ of operation to the FF that allows all or a portion of the FFs to be linked together into a scan chain. The scan input of the scan chain is connected to a primary input, and the scan-chain output is connected to a primary output. This enables the test engineer to set and observe the internal state of the FFs directly and therefore significantly simplifies the task of testing internal combinational blocks for defects.

Figure 2.7 shows the modifications that are made to conventional DFF to convert them into scan FFs. Two styles of scan insertion are shown, called MUXD-SFF and LSSD-SFF. The MUXD-SFF cell now includes a multiplexed input and a control signal that allows functional mode (with normal D input selected) and scan mode

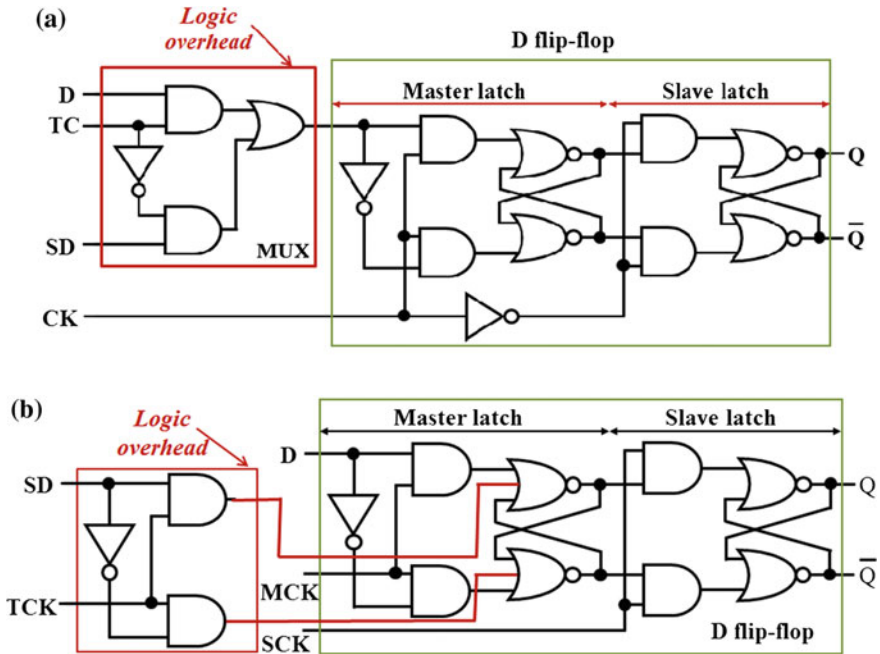


Fig. 2.7 Scan flip-flop design. **a** MUXD-SFF. **b** LSSD-SFF

(with scan input selected). During manufacturing test, the test mode signal called scan enable is asserted to enable the scan operation. The operation of scan-based testing (Fig. 2.8) is a three-step process:

- (a) Configuration of the scan cells with a test vector,
- (b) Application of the system clock to capture the results, and
- (c) Readout of the scan data for analysis.

Built-in self-test (BIST) also adds DFT components to the chip as a mechanism to enable a self-testing mode of operation. BIST can significantly reduce the test costs by making it possible for the chip to self-detect problems, which reduces the dependency and time required on expensive automatic test equipment (ATE).

Scan-Based Attacks on Obfuscation

Scan-based testing is a significant and important tool for reducing cost and improving coverage of manufacturing test, but it can also be used to support noninvasive attacks designed to steal important information such as keys or to bypass security mechanisms and aid adversaries in reverse engineering attacks. Scan chains are easily exploitable by an adversary who has access to the chip and can use it as a ‘side channel’ for malicious activities such as cryptanalysis [15, 16]. Scan-based attacks are categorized broadly into two categories: scan-based observability attacks and scan-based controllability/observability attacks. A scan chain provides the adversary with

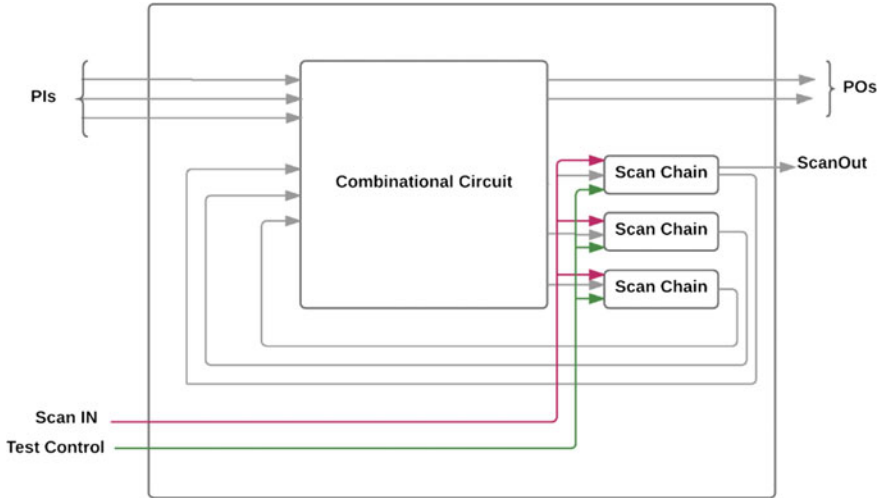


Fig. 2.8 Scan-based testing

the ability to take snapshots of the chip in different states to help reverse-engineer the design. Alternatively, the adversary can set registers to specific values while operating the chip in test mode and hence can access internal secrets, such as key registers, and change key operational modes as a means of bypassing any inserted security mechanisms.

Countermeasures for Scan-Based Attacks

Several techniques are proposed to secure the scan chain, such as disabling scan chain after manufacturing test and scrambling scan chain to make it harder for the adversary to carry out reverse engineering attacks. The scan infrastructure can be secured by blowing fuses to disable scan chain after manufacturing test [17]. In this approach, the protected registers can be made uncontrollable and unobservable by eliminating physical access to them. The disadvantage of this approach includes the fuse-blowing post-processing step and the vulnerability of fuses to focused ion beam (FIB) attacks [18]. FIB tools have been developed to enable ‘circuit edit,’ i.e., the adding and removal of metal at specific regions in the chip. The adversary can use FIB to repair the blow fuses and re-enable access to the secret keys and design details.

Scan-chain scrambling techniques obfuscate the register-to-scan-chain mapping to make it harder to interpret scan data [19]. The technique requires a key to establish the correct assignment of register-to-scan-chain mapping. Incorrect keys randomly map the registers to scan-chain elements, effectively scrambling the data. This technique protects embedded secret information as well as details of the internal design, making it difficult to reverse-engineer the chip.

An alternative is to implement a key separation method that disables access to the secret key register in test mode [16]. The proposed method introduces a mirror key register (MKR) which is muxed-in when scan mode is enabled and which prevents

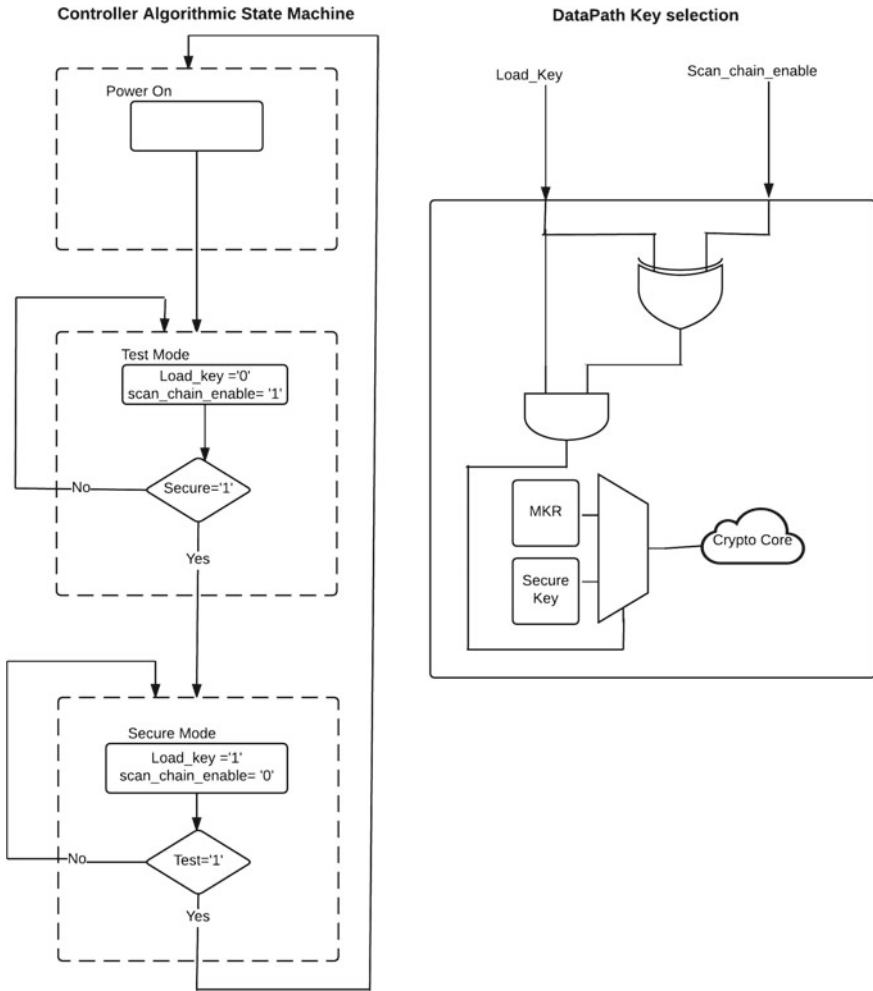


Fig. 2.9 Algorithmic state machine and data path for mirror key

access to the cryptographic key register. This approach enables the cryptographic unit to be tested for manufacturing defects but prevents an adversary from using scan to steal the secret key during functional mode. The control signal to the MKR is the scan-enable signal, so the switch to the MKR is performed automatically when scan is enabled. A block diagram of the proposed method is shown in Fig. 2.9 using the algorithmic state machine and data path. This technique protects the secret key, but still allows the probing of internal design details and therefore does not prevent reverse engineering attacks.

A low-cost secure scan (LCSS) technique is proposed to overcome this deficiency by introducing dummy flip-flops in the scan chain [20]. The proposed architecture is

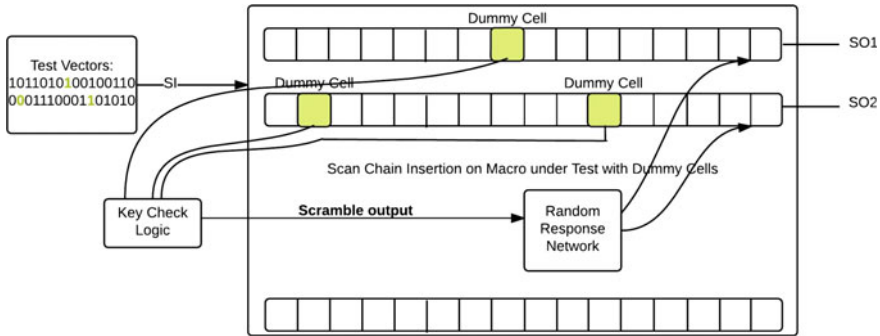


Fig. 2.10 Low-cost secure scan (LCSS)

shown in Fig. 2.10. LCSS requires only small changes to be made in the design flow to accommodate the insertion of additional scan cells and can be used to protect secret embedded keys and the chip’s intellectual property. All the dummy cells are checked with key checking logic (KCL) to determine whether the dummy cells have been programmed with the correct code. Incorrect codes disable access to the scan-chain data and instead enable a q-bit LFSR which generates random data on the scan-chain output.

2.2.8.2 Boundary Scan

Boundary scan is a DFT mechanism for printed circuit board (PCB)-level testing, that is similar to the scan DFT technique used inside the chip. Boundary scan creates a shift register out of the I/O pads of chip and allows the chip solder connections and interconnect on the PCB to be tested for manufacturing defects. JTAG is an IEEE standard 1149.1 developed by a working group called the Joint Test Access Group, along with other scan architectures including IEEE Std. 1500 and IEEE P1687 (IJTAG) for reconfigurable scan networks.

JTAG

JTAG provides a single test interface across heterogeneous components/devices on a printed circuit board (PCB) and hence facilitates testing. Figure 2.11 shows the interface signals of the test access port (TAP).

The test signals for the JTAG interface are defined as follows:

- TCK: Test clock—all the boundary scan cells are shifted with the event on TCK.
- TMS: Test mode select determines the next state. There are 16 states in JTAG.
- TDI: Test data in—test vectors are provided through this signal. Additional JTAG instructions are also provided by TDI.
- TDO: Test data out—scan out the responses.

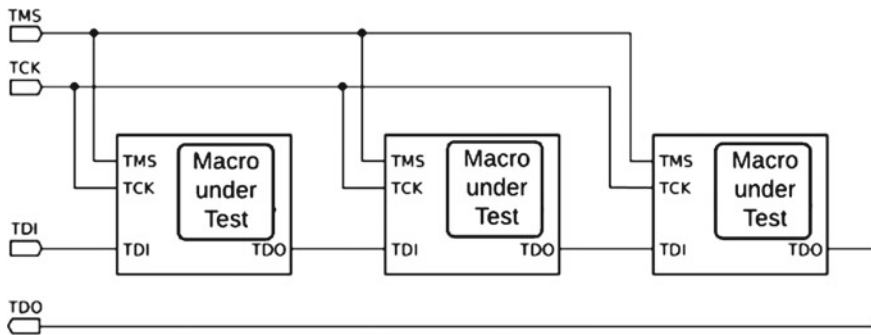


Fig. 2.11 JTAG

JTAG-Based Attacks on Obfuscation

JTAG makes the chips and entire PCB vulnerable to attacks because it does not implement any type of device authentication in its daisy chain topology. Several attacks have been reported that exploit the JTAG interface as a means of stealing secret keys, of carrying out piracy of intellectual property, and to circumvent standard policies. The adversary can also replace genuine chips with counterfeit clones without fear of being detected. Therefore, JTAG has the same type of vulnerabilities as scan-chain design and additionally is vulnerable to the insertion of malicious devices. One such attack model is discussed in [21] explaining that the adversary can hijack the shared resources, such as bus, and launch a denial-of-service attack or spoof information.

Countermeasures Against JTAG Attacks

One countermeasure proposed in [22] is to implement JTAG interface using fuses and electronically destroy it after the completion of manufacturing test, thus eliminating security risks. A better solution is to add a security mechanism to JTAG that is designed to limit the access to authorized users. An authentication mechanism can also be added to allow a controller or centralized trusted server to authorize chips to perform certain tests [23]. An alternative is to allow the controller chip to abort test traffic by introducing security policies [24]. Compact cryptographic modules can be further included to encrypt test data and carry out key-based authentication of chip under test [21]. Keys can be programmed on chip in tamper-evident nonvolatile memory or they can be generated on the fly using physical unclonable functions [25].

2.2.8.3 Built-In Self-Test (BIST)

Built-in self-test is a testing technique that can generate and apply random test vectors on chip and then validate that the results are fault-free, thus eliminating ATE at the cost of additional area overhead on the chip. BIST is commonly used to test embedded memories that do not provide external pins for direct access. An example showing

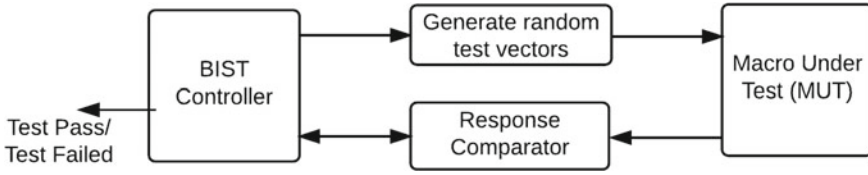


Fig. 2.12 Built-in self-test process

how BIST can be implemented is shown in Fig. 2.12. The controller applies random test vectors to the macro under test (MUT) and verifies the responses on chip. The interface control signals only convey whether the circuit is passed or failed and does not transfer the responses. BIST can be implemented with chips and boards that also include JTAG. Since BIST does not reveal the data or state of the system, it naturally provides obfuscation.

2.3 Hardware-Based Obfuscation Design Primitives

To better understand hardware primitives that are currently used in several obfuscation techniques, we first discuss a classification scheme for obfuscation techniques.

2.3.1 Types of Hardware Obfuscation

Hardware-based obfuscation is broadly categorized as passive hardware obfuscation, active hardware obfuscation, and reconfigurable logic-based obfuscation. Recently proposed method includes active key-based hardware obfuscation schemes that can be further classified as combinational logic obfuscation and finite state machine (FSM)-based obfuscation.

2.3.1.1 Passive Hardware Obfuscation

In keyless or passive hardware obfuscation techniques, the design description is obfuscated and/or encrypted using cryptographic primitives. A register transfer-level (RTL) design obfuscation technique is discussed in [26], which renames signals and reorganizes the code to obscure its meaning to adversaries. Research reported in [27–29] encrypts the hardware description language (HDL) before distributing to untrusted entities in the supply chain. The IP designer provides key to legal customers to decrypt the design for integration or for fabrication.

In passive or keyless hardware obfuscation, the functionality is not modified and only the design file or netlist is obfuscated. Passive hardware obfuscation techniques

do not stop the adversary from using the design as a black box or from distributing or overbuilding the design. Also, passive techniques cannot prevent the customers from distributing the decrypted copy.

2.3.1.2 Active Hardware Obfuscation

Active hardware obfuscation or key-based techniques, on the other hand, modify the functionality as a mechanism to harden the design against reverse engineering. Logic-based obfuscation involves embedding the key in the functional unit itself and requires the user to provide the correct keys along with the functional inputs to get the correct results. Key integration is accomplished by the insertion of key-based logic in the combinational logic paths and/or finite state machine (FSM) of the design. For example, most proposed techniques add states to the FSM and XOR and XNOR gates to the data path.

Combinational Logic Obfuscation

Logic obfuscation modifies the design by incorporating additional gates such as XOR and XNOR to the data path, which have one or more of their inputs driven by registers that store the key. Timing analysis is typically performed in advance to select insertion points that do not impact the timing characteristics of the design. The insertion points can be selected randomly [30, 31] among the noncritical paths available, or more sophisticated techniques can be employed, such as those based on graph theory [32] or fault model analysis [33]. These techniques are covered in detail in Chaps. 5 and 6.

FSM-Based Logic Obfuscation

FSM-based obfuscation, also referred as IC metering, modifies the circuit design and locks each chip using a unique state transition path that can only be unlocked when the chip receives the correct key from a key management authority or design house. The key ensures the chip follows an unlocking sequence of state transitions when powered up to run in functional mode [34]. These techniques can be designed to require a unique key for each chip, that is either stored in a NVM such as an EEPROM or fuses or be generated on the fly using a physical unclonable function (PUF). The key is paired with an augmented FSM in such a way that only the design house can unlock the chip. Sections 2.4 and 2.5 discuss the key management using NVMs, PUFs, and TRNGs.

Reconfigurable Logic-Based Obfuscation

Reconfigurable logic-based obfuscation technique suggests to make a small component of the design reconfigurable in the chip. This approach hides the functional details of the obfuscation method during the manufacturing process which hinders the untrusted fabrication facility from reverse engineering the design. The technique proposed in [35] utilizes a fingerprinting technique by altering the implementation

slightly as a mechanism to detect clones or overbuilding. The use of embedded reconfigurable logic against code injection attacks on an open source SPARC processor is discussed in [36, 37].

2.3.2 Metrics of Hardware Obfuscation

Active hardware-based obfuscation hardens the design against reverse engineering, but such a scheme is vulnerable to side-channel attacks on keys and simulation-based attacks designed to decode key-gate values. It is assumed that if the malicious user is given enough time and resources, the obfuscation will fail. The authors of [38] propose the following objectives and metrics for hardware obfuscation:

- (I) The size of the input space must be large enough to make brute force attacks on FSM and combinational logic obfuscation infeasible.
- (II) The obfuscation method should attempt to maximize the impact of wrong key guesses, such as the hamming distance between the correct outputs and obfuscated outputs is 50%.

2.4 Volatile and Nonvolatile Memories

Active hardware obfuscation techniques require a key storage mechanism to produce the correct results, and for the case of programmable logic, netlist configuration information must be available at power-on to reconfigure the field programmable gate components. A variety of technologies exist to permanently store the keys including volatile memory and nonvolatile memory (NVM).

2.4.1 Volatile Memory

Key-related information stored in a volatile memory, such as dynamic random-access memory (DRAM) or static RAM (SRAM), is lost over power cycles of the chip unless it is powered from a battery, which represents a cost overhead for the system and reduces its reliability and availability. Other disadvantages of using volatile memory for key storage are that it is vulnerable to ‘cold boot attacks’ and requires the key communication process to be secure.

2.4.2 Nonvolatile Memory

NVM retains data across power cycles and can be categorized according to the writing mechanism that they employ. ROM, EPROM, EEPROM, and FLASH are common NVMs that are read-only, read mostly, and rewritable, respectively.

2.4.2.1 Read-only Memory (ROM)

ROM is a read-only memory that is programmed during the manufacturing process and cannot be changed in the field. ROM is a high-speed, high-density, and low-cost memory, thus making it an attractive medium for low-cost and embedded devices. Keys are permanently stored, are immutable, and are usually the same for all the manufactured devices. Furthermore, ROMs are vulnerable to attacks whereby adversaries can use specialized tools to read out their contents.

2.4.2.2 Reprogrammable Memory

EPROM and EEPROM are reprogrammable memories that can be programmed after manufacturing. These memories utilize floating gate-type technology which allows the data storage transistor to be reprogrammed by changing the trapped charge on the gate input. The floating gate retains the trapped charge across power cycles, and therefore, it does not require a battery. However, specialized hardware is required to add and remove the trapped charge. A benefit of floating gate technologies is that keys can be programmed after manufacturing, thereby preventing the manufacturer from engaging in reverse engineering attacks.

2.4.2.3 One-Time Programmable Memory

Antifuse, e-fuse, and laser fuses are one-time programmable memories and therefore represent a class of fused-based technologies. Fused-based storage is more vulnerable to invasive attacks which probe the layout of the chip as a means of stealing the secret information and bypassing the security mechanisms.

2.4.2.4 Emerging Technologies: RRAM or ReRAM, PCM, and STT-MRAM

Resistive random-access memory (RRAM) or ReRAM is a NVM that stores ‘0’ and ‘1’ by changing the resistance of memristor devices. PCM is similar to ReRAM, which stores information using resistance levels. STT-RAM stores information on ferromagnetic layers using magnetic polarization and has the access speeds close to

the caches. These memories are nonvolatile and therefore do not require an energy source to maintain their contents.

2.4.3 Limitations of Current Key Storage Mechanisms

The advantage of storing information in RAM is that the secret information is lost after a power cycle. On the other hand, the battery-backed RAM introduces reliability issues because data is lost if the battery fails. Conventional key management systems utilize NVMs to store master keys or session keys, but as pointed out earlier, NVMs are vulnerable to invasive and noninvasive attacks.

Invasive attacks such as microprobing and laser-cutting attacks allow the adversary to learn the secrets by decapping the chip and reading the memory cells. To mitigate physical attacks, variants of tamper-resistant NVM include sensors to detect physical access to the device. The sensors require a battery to remain active while power is turned off. Therefore, NVM is not attractive for use in embedded and resource-constraint devices.

Noninvasive attacks that target key extraction from NVM include glitch attacks, fault injection (timing, voltage, temperature, radiation), and power analysis. Mitigation techniques include randomized design flows or design techniques that equalize power consumption. The drawback of these approaches is that it does not follow traditional design flows and increase area overhead to the design. Furthermore, emerging NVM technologies are promising but have not been validated as to whether they provide enhanced security properties over conventional NVMs.

2.5 Design Obfuscation: PUF and TRNG

2.5.1 Physical Unclonable Functions (PUFs)

Physical unclonable function (PUF) is an emerging physical layer cryptographic primitive used in hardware security and privacy protocols. They are embedded structures that utilize inherent manufacturing process variations to extract unique but reproducible secrets. The concept was first introduced as physical one-way functions [39] and later as physical unclonable functions [40]. PUFs measure variations in propagation delays, wire resistances, and other analog circuit parameters to produce digital bitstrings that are random and unique across instances of the chip population. PUFs are unclonable because the random information source on which they are based (the source of entropy) cannot be replicated, i.e., manufacturing process control cannot reduce the physical and electrical variations that occur across and within chips to zero tolerance levels. PUF bitstrings are generated on the fly as needed and therefore eliminate the need for NVM, e.g., EEPROM and e-fuses. This new key

generation mechanism eliminates the probing attack vulnerabilities discussed earlier in reference to NVMs because PUFs do not store digital versions of the bitstrings and the analog nature of the entropy source makes it tamper-evident whereby physical probing changes and/or destroys the ability to regenerate the same bitstring.

2.5.1.1 PUF Operation

(a) Apply Challenges

PUFs are based on a challenge–response pair (CRP) mechanism. The challenge for a PUF is defined as a digital input, usually in the form of a bitstring of ‘0’s and ‘1’s. The output of a PUF is also digital, but for most PUFs, this requires an on-chip mechanism to convert the small analog variations leveraged by the PUF to be digitized. The digitization process occurs automatically for some PUFs, such as the SRAM PUF, where the bitstring is produced immediately after power-up. The challenge to the PUF typically selects a unique set of elements from the entropy source or, as is more common, selects a set of elements that are combined in a unique fashion. The randomness of the entropy source ensures that the CRPs are unique across chips, i.e., the response bitstring produced by the PUF is different for each chip even when using the same challenge. In the best case, 50% of the bits in the response bitstring are uniquely defined by each PUF instance.

(b) Enrollment

PUF-based security applications require an enrollment process. Enrollment is carried out in a secure environment where CRPs are measured and stored by a trusted authority in a secure database. Enrollment can also be done while the chip is in the field as long as the PUF’s existing secrets can be used to securely transmit new CRPs to the trusted authority.

(c) Regeneration

Regeneration is a process that is carried out by a fielded chip usually in response to a request issued by an application that requires a key for encryption or a unique bitstring for authentication. When exact replication of the bitstring is required, e.g., key generation for encryption, PUFs require some type of helper data as a means of fixing or avoiding bit flip errors that occur when the CRPs are reapplied. Helper data is typically stored by the trusted authority during the enrollment process and is transmitted to the fielded device in-the-clear when needed. Therefore, helper data does not leak any, or leaks very little information, about the secret bitstring. In other applications such as authentication, exact reproduction of the bitstring may not be required, and instead, a close match is sufficient to confirm the identity of the chip.

2.5.2 PUF Evaluation Measures and Parameters:

2.5.2.1 Effect of Environmental Variations

Reproducing the bitstring exactly without helper data is challenging for PUFs because of changes that occur to the entropy source when the environment changes, i.e., the outside temperature is high or a low battery causes the supply voltage to drop on the chip. Changes in the environment introduce bit flip errors in the response bitstring, making it difficult to achieve high reliability. Error-tolerant mechanisms can be designed into the PUF architecture to help mitigate these types of adverse effects, e.g., [41]. However, error tolerance is not sufficient in many applications, and helper data must also be used as described above to meet the reliability requirements.

2.5.2.2 Evaluation Metrics of PUF

A survey on the PUF evaluation metrics is reported in [42] and includes techniques designed to measure and quantify the quality of PUF response bitstrings. The most important of these are summarized as follows:

1. Uniqueness

Uniqueness is a quality metric that ensures the responses generated by any two chips for a given challenge should be substantially different. Interchip hamming distance (HD) is used to quantize the difference, and in the ideal case, it is 50%.

$$uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{n-1} \cdot \sum_{j=i+1}^n \left(\frac{HD(R_i, R_j)}{n} \right) \quad (2.5)$$

where ‘R’ is the response, ‘k’ is the number of chips, and ‘n’ is the length of the response bitstring. Interchip (HD) quantifies the number of differences that occur across a set of response bitstrings. Ideally, each bit position of the response bitstring has the equal probability of being a ‘0’ or ‘1’. If some bits are biased to one value or the other, then these bits are ‘more predictable’ from an adversarial point of view. Bit biasing is measured across each bit position in the response bitstrings from a set of chip using hamming weight as given by Eq. 2.3, where ‘i’ is the bit position and ‘m’ is the number of chips.

$$Bit\ aliasing[i] = \frac{1}{k} \sum_{j=1}^k (R_j) \quad (2.6)$$

2. Reproducibility

Response bitstring reliability or reproducibility is measured using intrachip HD. The data used in the analysis is the response bitstring measured under different

environmental conditions from the same chip using the same challenge. The data from other chips is typically factored in by computing the average intrachip HD from the individual analyses. The ideal value is 0%, i.e., no chip has any bit flip errors under any environmental conditions. As indicated above, this ideal result is not possible using the ‘raw’ response bitstrings directly, and instead, helper data is required to achieve error-free regeneration. PUFs that are able to achieve relatively low interchip HDs without helper data can be used in some authentication applications that just require most of the bits in enrollment and regeneration bitstrings to match.

$$\text{Reproducibility} = \frac{1}{m} \sum_{i'=1}^m \left(\frac{HD(R_i, R_{i'})}{n} \right) \quad (2.7)$$

Equation 2.4 gives the formula for computing intrachip HD. It counts the number of bits that are different in the bitstrings collected over ‘m’ different environmental conditions, called temperature–voltage corner conditions.

2. Randomness

The PUF architecture, including the circuit structure used as the source of entropy, and measurement technique can induce bias and reduce the randomness in the sequence of bits generated by each chip. Bitstring randomness is measured using statistical tests; for example, NIST has developed statistical testing software for evaluating randomness in bitstrings generated by pseudorandom number generators [43]. The test includes uniformity and frequency tests that count the number of ‘0’s and ‘1’s in a bitstring. The frequency test requires the balance of ‘0’s and ‘1’s in any given bitstring to fall within a tolerance; otherwise, the bitstring fails the test. The test suite includes other tests that look for patterns in a set of bitstring subsequences that occur more often than expected from bitstrings drawn from a truly random source. The NIST software tool suite applies a set of up to 15 different tests to each of the input bitstrings and reports the number of tests that each bitstring passes. Several other randomness evaluation tools have also been developed including DIEHARD [44] and AIS.31 [45].

2.5.3 Classification of PUFs

PUFs are classified into weak and strong PUFs, based primarily on two criteria: the size of their CRP space and the level of resilience they have against model-building attacks. A third related criteria is the size of the entropy source, i.e., how many independent random varying components are used to generate the bitstrings. A second PUF classification uses the terms ‘intrinsic’ and ‘nonintrinsic,’ which relates to whether the PUF is self-contained on the chip (intrinsic) or requires external instrumentation to support it. Yet a third classification uses the terms ‘nonelectronic’ and ‘electronic’ to refer to the underlying structure of the entropy source, e.g., silicon versus a material that exhibits random properties.

2.5.3.1 Weak and Strong PUFs

Strong PUFs can produce a very large, unique set of bits per device and have a very large CRP space to support this characteristic. The very large CRP space makes it impractical for an adversary, who has possession of the PUF chip, to apply them all as a means of building a database, i.e., a ‘digital’ clone of the PUF. Many consider a second property, i.e., model-building resistance, to be equally important to the very large CRP space. Model-building refers to an attack mechanism whereby the adversary uses the machine learning algorithms to derive a system capable of predicting the response of the PUF after being trained on only a small portion of the CRPs. Resistance to model-building attacks is best realized using an entropy source with a large set of randomly varying components, but many proposed PUF architectures, e.g., the Arbiter PUF, only have a small set and instead use cryptographic primitives such as secure hash functions and XOR networks to obscure the CRP interface. The HELP PUF is an example of a PUF which is based on a large entropy source, i.e., the best-case scenario [25].

Weak PUFs on the other hand have fewer CRPs and, in some cases, only one response pair which is the case for the SRAM PUF. Weak PUFs are usually limited to key generation where model-building attacks do not apply because the secret does not leave the chip. Weak PUFs can also serve applications that require only a unique ID from the PUF. Weak PUFs have capabilities similar to those provided by an NVM, but provide a tamper-evident property which enhances their security over NVM. Examples of weak PUFs include physically obfuscated keys (POKs) [41], SRAM PUF [46], and Butterfly PUF [47].

2.5.3.2 Intrinsic and Nonintrinsic PUFs

As indicated above, intrinsic PUFs are completely self-contained architectures on the chip, capable of making measurements, and carry out bitstring generation, whereas nonintrinsic PUFs require benchtop instrumentation, e.g., photonic-based sensors, to implement the measurement components. Intrinsic PUFs are far more popular, and the number of proposed architectures continues to grow. Manufacturing process variations on the chip manifest in many forms on the chip including within-transistor threshold voltages and metal resistance characteristics. For example, PUFs based on variations in delay include the Arbiter [48, 49], Ring Oscillator [41], and HELP [25] PUFs. Delay is popular because there are many well-defined on-chip delay measurement techniques that are available.

2.5.3.3 Sources of Entropy

Examples of nonelectronic PUFs include coating PUFs, optical PUFs, and CD Player PUFs. Nonelectronic PUFs, to date, have not been considered as support primitives for implementing hardware obfuscation functions. Electronic intrinsic PUFs are the

most common class of PUFs proposed for this purpose and include those based on variations in transistor threshold voltages [50, 51], propagation delay (as indicated above, the Arbiter, Ring Oscillator [41], and HELP [25] PUF are examples), and power-up patterns in memory, e.g., the SRAM PUF [46]. The list of electronic intrinsic PUFs keeps growing and includes the ROM PUF [52], leakage current PUF [53], the metal resistance PUF [54, 55], the transistor transconductance PUF [56], and other path delay-based PUFs [57, 58].

2.5.4 PUFs: Candidates for Hardware Obfuscation

In this section, we discuss the following PUFs, both weak and strong, that are considered good candidates for hardware obfuscation applications:

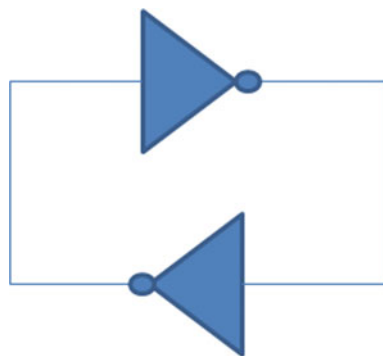
- (1) Memory-based intrinsic SRAM PUF.
- (2) Delay-based intrinsic PUFs including the Arbiter, Ring Oscillator, and HELP PUFs.

(1) SRAM PUF

The SRAM PUF is classified as a weak intrinsic PUF that uses the randomness in the power-up bit patterns of SRAM as source of entropy [46]. The SRAM cell is implemented as a pair of cross-coupled invertors whose geometries are identical (Fig. 2.13). Manufacturing process variations cause mismatches in the transconductance parameters of the inverters resulting in the random power-up states that remain constant for the majority of the cells. The power-up pattern varies from one chip to the next, enabling the SRAM PUF to serve in chip identification roles and in PUF-based hardware obfuscation protocols to map reconfigurable logic to a specific function.

SRAM PUF behavior is affected by the systematic variations, where the number of ‘1’s and ‘0’s can be biased, thus degrading its randomness statistical metric, making it vulnerable to model-building attacks (SRAM PUFs cannot be model-built). SRAM

Fig. 2.13 Cross-coupled NOT gate SRAM cell



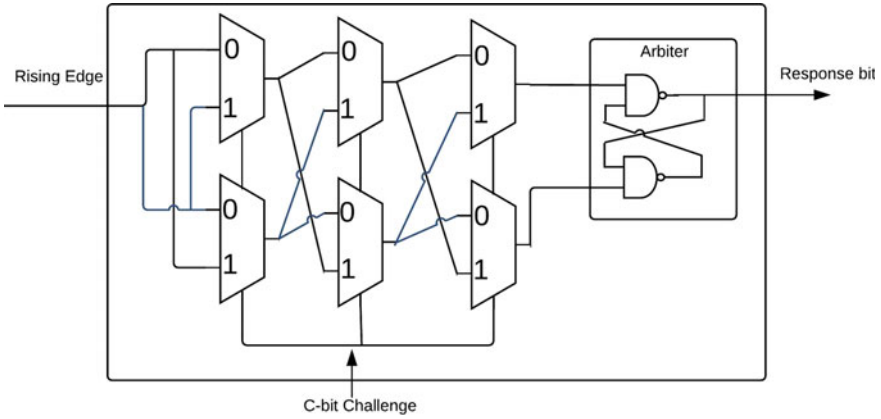


Fig. 2.14 Arbiter PUF

PUFs typically have poor reproducibility, reported as high as 20% or more in some cases.

(2) Arbiter PUF

The Arbiter PUF is a delay-based PUF defined using a sequence of multiplexers and an arbiter, e.g., a cross-coupled NAND latch as a mechanism to provide an unbiased evaluation mechanism as shown in Fig. 2.14 [48, 49]. The PUF leverages the delay variation between two identical paths to generate a bit. Challenge bits select the configuration of the switches that in turn determines the specific configuration of the paths. The pairs of multiplexers serve as switch boxes, either routing the two paths straight through the switches or flipping their connections. For a given challenge, the Arbiter PUF measures the delay of two identical length paths. A rising signal is given input to leftmost pair of multiplexers, as shown in Fig. 2.14. The input signal races along the two delay lines, and the arbiter at the end assigns a '0' or '1' based on which path is faster. The connection of the path endpoints to the D and Clk inputs allows the arbiter gate to automatically compute the result of the race.

The arbiter PUF is vulnerable to model-building attacks because of its linear structure and small number of components. A precise timing model can be constructed to learn the parameters from a relatively small set of CRPs. To reduce the effectiveness of model-building attacks, the authors of [41, 59] propose a parallel Arbiter architecture which includes an XOR obfuscation network on the outputs.

(3) Ring Oscillator PUF

A Ring Oscillator PUF (RO PUF) is a weak PUF composed of identical delay loops and counters [41]. RO PUFs measure path delay variations as differences in the 'ring' frequency of the delay loops (Fig. 2.15). Challenges select a pair of identical oscillators and compare the number of oscillations produced by each oscillator of the pair. Frequency is measured by connecting the output of each RO to a separate

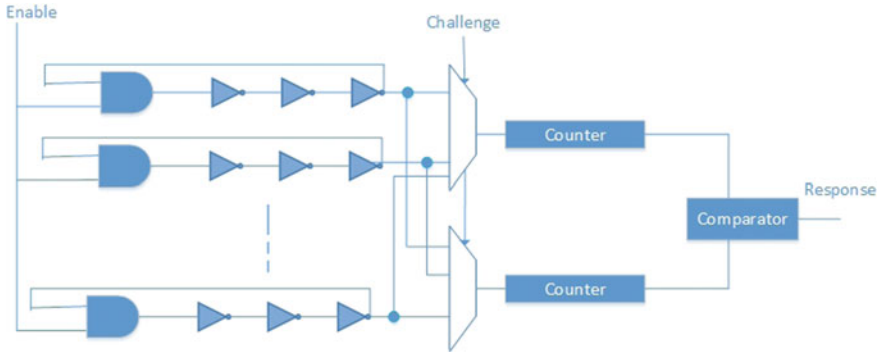


Fig. 2.15 Ring oscillator PUF

counter. The result of the comparison generates a single ‘0’ or ‘1’ bit in the bitstring. Other pairings are used to construct the additional components of the bitstring.

RO PUFs are also subject to model-building attacks in common usage scenarios in which the same RO is used in multiple different pairings. Machine learning algorithms attempt to determine the relative frequencies of all ROs, which, once known, make it possible to predict the response bitstring to any sequence of challenges used to build the bitstring [60].

(4) HELP PUF

A hardware-embedded delay PUF (HELP PUF) proposed in [25, 57] is a strong PUF. It leverages delay variations in existing design functional units and does not require identical structures, unlike other existing delay-based PUFs. HELP also implicitly provides tamper protection of the existing functional unit(s), i.e., any change in the structural characteristics of the functional unit will change the measured path delays.

Figure 2.16 shows the architecture of HELP with the functional unit representing the entropy source. The inputs and outputs of the functional unit are connected to a set of launch row and capture row flip-flops (FFs), respectively. A series of launch–capture clocking events are applied to the functional unit using two clocks, Clk_1 and Clk_2 as shown on the left side of Fig. 2.16. The phase shift between Clk_1 and Clk_2 is adjusted dynamically across the sequence of launch–capture tests, where the digitally selected value of the fine phase shift between the two clocks is referred as the launch–capture interval (LCI). The smallest LCI interval that allows the propagating edge along a path to be captured in the capture FF is used as the digitized timing value for the path.

PUF response bits are computed from delay differences between nonidentical path delays. A modulus technique is proposed as a means of removing the bias in the path delays of the nonidentical paths used in the difference operation while fully preserving the smaller within-die delay variations.

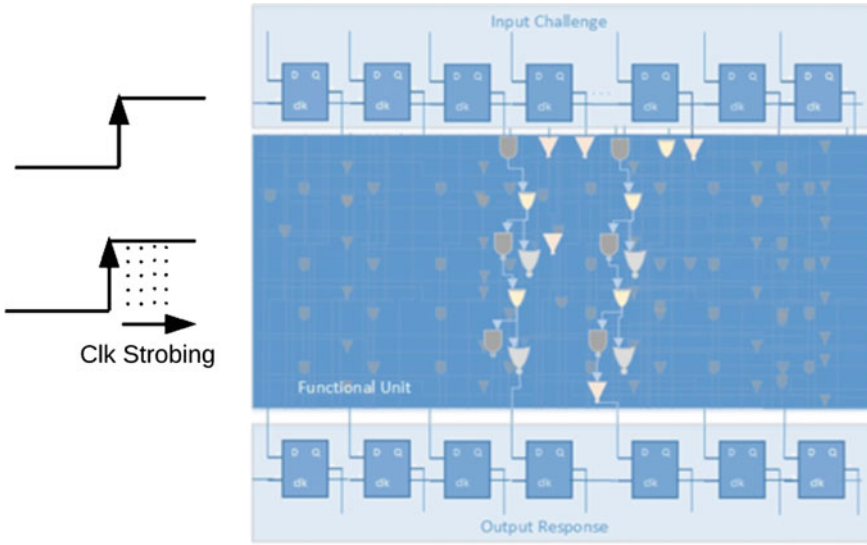


Fig. 2.16 HELP PUF

2.5.5 True Random Number Generator (TRNG) Use in Hardware Obfuscation

True random number generators are hardware primitives that are used in many hardware-based security techniques, including hardware obfuscation. A true random number generator (TRNG) uses randomness and noise to generate secrets that are not reproducible. The randomness or noise should have uniform distribution to avoid bias. The TRNG is an important primitive for cryptographic applications, which is used for generating nonces for authentication protocols, for generating one-time pads and for providing a selection mechanism for primes, as a unique key per device, etc. Hardware obfuscation and hardware metering using a TRNG are proposed in [31] to define randomized chip IDs upon power-up that are then stored in tamper-resistant NVM.

A TRNG can be implemented using on-chip variations [46, 48]. Examples of such TRNG are arbiter-based TRNGs, ring oscillator-based TRNGs, and technology-independent TI-TRNGs [61]. TRNGs are evaluated with respect to randomness and the uniformity of their distribution. Environmental variations such as supply voltage or temperature variation can adversely affect the noise distribution and introduce bias, making the output from the TRNG more predictable.

TRNGs are used to generate unique keys for input to key gates in combinational logic and in obfuscated state machines. The obfuscated data path and control path produce the correct output when the correct key is applied. TRNG-based key generation requires storage of the generated key in a battery-backed RAM or NVM.

The disadvantages of using TRNGs for producing keys are that the stored keys in battery-backed memory or NVM can be stolen and cloned, allowing designs to be reverse-engineered and security features completely eliminated from the design. Additionally, the overhead of manufacturing of NVM requires additional masks and manufacturing steps, thus increasing the costs of the chip. Thus, other alternatives such as physical unclonable functions are better suited for the generation of reproducible secret keys, as long as high reliability to bit flip errors can be ensured.

2.5.6 Applications of PUFs and TRNG in Hardware-Based Obfuscation Techniques

PUFs and TRNGs can be incorporated into logic obfuscation for the chip authentication [31, 34] or for obfuscation of logic [62]. PUFs and TRNGs can use nonelectrical properties such as heat, atmospheric noise, and fiber optics as a source of entropy; however, focus has been on the silicon process variations that can be more easily measured and digitized. PUF-based obfuscation and activation schemes can be used to improve security by allowing each chip to be assigned its own unique challenge-response pairs, thereby allowing each chip to exclusively modify and hide the design and authenticate to allow correct functionality, respectively.

A finite state machine (FSM)-based metering technique described in [34] hides the functionality with an augmented FSM structure known as black hole finite state machine (BFSM). The PUF response directs the state transition from obfuscated states to the valid state, and only valid transitions can bring the chip to a properly functioning operational state. The PUF is used to generate a unique key for the finite state-based activation and hides actual functionality from the adversary, thereby preventing illegitimate overbuilt chips. This augmented FSM can be implemented using reconfigurable logic, where each chip has a unique key based on the chip identifier. A PUF-based BFSM technique is shown in Fig. 2.17.

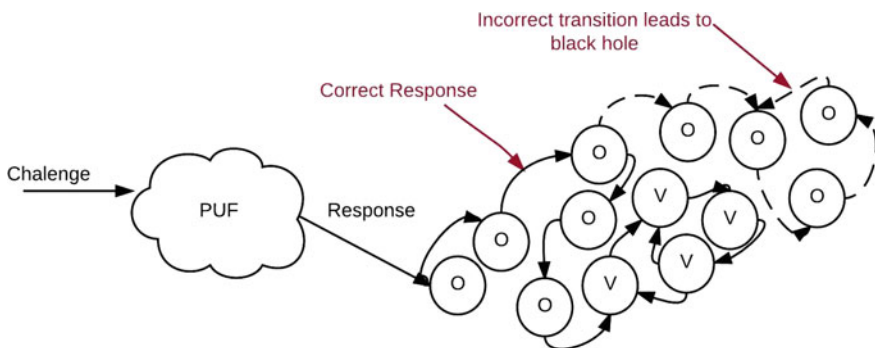


Fig. 2.17 PUF-based BFSM technique

This technique was subsequently modified by [63] using a smaller number of obfuscated states for remote activation of resource-constraint devices. Some valid states are replicated, and the transition through the replicated states is only possible with the correct key.

An FSM-based hardware obfuscation and metering technique using TRNG is described in [31]. As explained earlier, a TRNG can be used to define randomized and unique identification (ID) upon power-up that is burnt into the electrically programmable fuses, such as an electronic fuse unit (EFU).

A PUF is proposed in [62] to implement hardware obfuscation for logic and interconnect obfuscation. The scheme is shown in Fig. 2.18, where each instance of the obfuscated integrated circuit is different, thus making it resilient to reverse engineering. The adversary not only is required to guess the gates but also needs to characterize the PUF responses or use a brute force method to explore all possibilities. Interconnect obfuscation is achieved by using switching gates such as multiplexers to create wire swapping. As shown in Fig. 2.19, only the correct key or PUF response will establish correct connections.

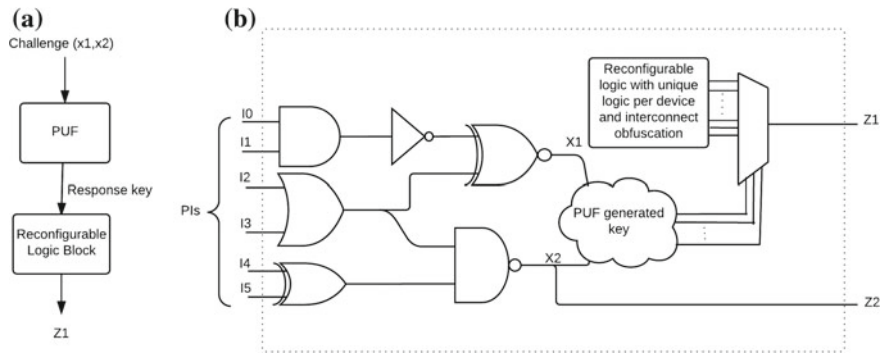


Fig. 2.18 a PUF-based random logic obfuscation. b Integration in design flow

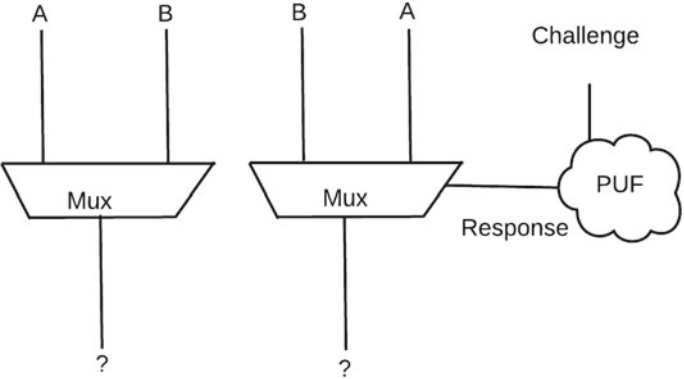


Fig. 2.19 Signal path obfuscation

By choosing PUF-based logic that affects multiple outputs, placement of obfuscated logic with uncontrollable flip-flops can further improve the security of these types of obfuscation techniques. Additionally, the selection of wire pairs to implement swaps between highly correlated pairs can increase the difficulty in reverse engineering. Therefore, hardware obfuscation schemes based on PUFs and TRNGs can effectively mitigate piracy attacks.

Summary

This chapter covered preliminary concepts and techniques of VLSI verification and testing. We describe a set of related vulnerabilities associated with VLSI verification techniques and testing structures that can expose the design details and help reverse-engineer the functionality to compromise the security through obscurity. Proposed changes to existing techniques are discussed that are designed to provide countermeasures against such attacks. The taxonomy of hardware obfuscation techniques is also presented, as well as a set of hardware primitives and related concepts. Hardware obfuscation techniques are motivated because of growing trend of offshoring the fabrication process, where the foundry has the complete knowledge of the design details in the form of GDSII. Obfuscation techniques modify the design and require correct keys as input in order to make the designs functional. Section 2.4 discusses different key storage schemes, such as nonvolatile memory and their vulnerabilities and overhead. Section 2.5 covers hardware-based cryptographic functions, physical unclonable functions (PUFs), and true random number generator (TRNG) as building blocks that further enhance the IC design obfuscation resilience against reverse engineering and mitigate IC piracy attacks. Subsequent chapters further discuss their applications to mitigate IC piracy, cloning, overbuilding, and use of counterfeit chips.

References

1. Zhuang X, Hsien-Hsin TZ, Lee S, Pande S (2004) Hardware assisted control flow obfuscation for embedded processors. In: Proceedings of international conferences on compilers, architecture, and synthesis for embedded system, pp 292–302
2. Rajendran J, Sinanoglu O, Karri R (2013) Is split manufacturing secure? In: Proceedings of the IEEE design, automation and test in Europe conference and exhibition (DATE), Grenoble, France, 18–22 March 2013, pp 1259–1264
3. Tehranipoor M, Wang C (eds) (2011) Introduction to hardware security and trust. Springer, New York, p 427
4. Guin U, DiMase D, Tehranipoor M (2014) Counterfeit integrated circuits: detection, avoidance, and the challenges ahead. *J Electr Test Theory Appl (JETTA)* 30:9–23
5. Marques-Silva JAP, Sakallah KA (1996) GRASPVA new search algorithm for satisfiability. In: Proceedings of the ICCAD, pp 220–227
6. Li CM, Anbulagan (1997) Heuristics based on unit propagation for satisfiability problems. In: Proceedings of IJCAI, pp 366–371
7. Malik S, Zhao Y, Madigan CF, Zhang L, Moskewicz MW (2001) Chaff: engineering an efficient SAT solver. In Proceedings of the DAC, pp 530–535. ([62] Marques-Silva JAP, Sakallah KA (1996) GRASPVA new search algorithm for satisfiability. In: Proceedings of the ICCAD, pp 220–227)

8. Subramanyan P, Ray S, Malik S (2015) Evaluating the security of logic encryption algorithms, HOST
9. Goldstein LH (1979) Controllability/observability analysis of digital circuits. *IEEE Trans Circuits and Syst (CAS)* 26(9):685–693
10. Goldstein L, Thigpen E (1980) SCOAP sandia controllability/observability analysis program. In: *Proceedings of the 1980 design automation conference*, pp 190–196
11. Bennetts R (1984) *Design of testable logic circuits*. Addison-Wesley, Reading
12. http://www.eng.auburn.edu/~agrawvd/COURSE/E7250_05/REPORTS_TERM/Kantipudi_Tmeas.pdf
13. Brglez F, Pownall P, Hum R (1984, October) Applications of testability analysis: from ATPG to critical delay path tracing. In: *Proceedings of the 1984 international test conference on the three faces of test: design, characterization, production (ITC'84)*. IEEE Computer Society, Washington, DC, USA, pp 705–712
14. Seth SC, Pan L, Agrawal VD (1985, June) PREDICT-probabilistic estimation of digital circuit testability. In: *Proceedings of the fault tolerant computing symposium*, pp 220–225
15. Yang B, Wu K, Karri R (2004) Scan based side channel attack on dedicated hardware implementations of data encryption standard. In: *Proceeding of the IEEE international test conference 2004 (ITC 2004)*, 26–28 October 2004, pp 339–344
16. Yang B, Wu K, Karri R (2005) Secure scan: a design-for-test architecture for crypto chips. *IEEE Trans Comput Aided Des Integr Circuits Syst* 25(10):2287–2293
17. Ebrard E, Allard B, Candelier P, Waltz P (2009) Review of fuse and antifuse solutions for advanced standard CMOS technologies. *Elsevier Microelectr J* 40(12):1755–1765
18. Young R, Carlson P (2004) (Dual-beam FIB/SEM): a tool for advanced failure analysis. In: *Evaluation engineering*, online magazine September 2004. <http://www.evaluationengineering.com/>
19. Hely D, Flottes ML, Bancel F, Rouzeyre B, Berard N, Renovell M (2004) Scan design and secure chip (secure IC testing). In: *Proceedings of the 10th IEEE international on-line testing symposium*, pp 219–224
20. Lee J, Tehranipoor M, Patel C, Plusquellic J (2007) Securing designs against scan-based side-channel attacks. *IEEE Trans Dependable Secure Comput* 4(4):325–336
21. Rosenfeld K, Karri R (2010) Attacks and defenses for JTAG. *IEEE Des Test Comput* 27(1):36–47
22. Sourgen L (1993) Security locks for integrated circuit. US Patent # 5264742
23. Busky RF, Frosik BB (2006) Protected JTAG. *Proceeding of the IEEE 2006 international conference on parallel processing workshops*. Columbus, OH, USA, pp 407–414
24. Clark CJ, Ricchietti M (2004) A code-less BIST processor for embedded test and in-system configuration of boards and systems. *IEEE test conference 2004*:857–866
25. Saqib F, Arenó M, Aarestad J, Plusquellic J (2014) An ASIC implementation of a hardware-embedded physical unclonable function. In: *IET Comput Dig Tech* 8(6):288–299 (Patent Pending)
26. Thicket family of source code obfuscators. <http://www.semdesigns.com>
27. Batra T, Methodology for protection and licensing of HDL IP. <http://www.us.design-reuse.com/news/?id=12745&print=yes>
28. Goering R, Synplicity initiative eases IP evaluation for FPGAs. <http://www.scdsource.com/article.php?id=170>
29. Xilinx IP evaluation. <http://www.xilinx.com/ipcenter/ipevaluation/index.htm>
30. Chakraborty RS, Bhunia S (2009) HARPOON: an obfuscation-based SoC design methodology for hardware protection. *IEEE Trans Comput Aided Des Integr Circuits Syst (TCAD)* 28(10):1493–1502
31. Roy J, Koushanfar F, Markov I, EPIC: ending piracy of integrated circuits. In: *Proceedings of the design automation and test in Europe (DATE)*, pp 1069–1074
32. Rajendran J, Pino Y, Sinanoghu O, Karri R (2012) Security analysis of logic obfuscation. *ACM/IEEE49th design automation conference (DAC)*, 3–7 June 2012. CA, USA, San Francisco, pp 83–89

33. Ranjendran J, Zhang H, Zhang C, Rose GS, Pino Y, Sinanoglu O, Karri R (2015) Fault analysis-based logic encryption. *IEEE Trans Comput* 64(2):410–424
34. Alkabani Y, Koushanfar F (2007) Active hardware metering for intellectual property protection and security. *Proceedings of 16 USENIX security symposium*, (2007) USENIX Association. Berkley, CA, USA, pp 291–306
35. Koushanfar F, Qu G (2001) Hardware metering. In: *Proceedings of the IEEE design automation conference 2001 (DAC 2001)*, pp 490–493
36. Liu B, Wang B (2014) Reconfiguration-based VLSI design for security. *IEEE J Emerg Selected Top Circuits Syst* 2014 (JETCAS 2014), 5(1):98–108
37. Baumgarten A, Tyag A, Zambreno J (2010) Preventing IC piracy using reconfigurable logic barriers. *IEEE Des Test Comput* 27(1):66–75
38. Rostami M, Koushanfar F, Rajendran J, Karri R (2013) Hardware security: threat models and metrics. In: *Proceedings of the 2013 IEEE/ACM international conference on computer-aided design (ICCAD 2013)*. San Jose, CA, USA, 18–21 November 2013, pp 819–823
39. Pappu R (2001) Physical one-way functions, PhD thesis, Massachusetts Institute of Technology
40. Gassend B, Clarke D, Van Dijk M, Devadas S (2002) Silicon physical random functions. In: *Proceedings of the 9th ACM conference on computer and communication security*, 2002, pp 148–160
41. Suh GE, Devadas S (2007) Physical unclonable functions for device authentication and secret key generation. In: *Proceedings of the 44th ACM/IEEE design automation conference (DAC '07)*. San Diego, CA, USA, 4–8 June 2007, pp 9–14
42. Maiti A, Gunreddy V, Schaumont P (2011) A systematic method to evaluate and compare the performance of physical unclonable functions. *J Int Assoc Cryptogr Res (IACR) ePrint*, 657:22
43. NIST: computer security division, statistical tests. http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html
44. Marsaglia G (1995) Diehard battery of tests of randomness. <http://www.stat.fsu.edu/pub/diehard/>
45. Killmann W, Schindler W (2011) A proposal for: functionality classes for random number generators. In: *AIS*, September 2011, p 133
46. Su Y, Holleman J, Otis B (2007) A 1.6pJ/bit 96 percent stable chip ID generating circuit using process variations. In: *Proceedings of the 2007 IEEE international solid-state circuits conferences (ISSCC)*, pp 200–201
47. Kumar SS, Guajardo J, Maes R, Schrijen GJ, Tuyls P (2008) Extended abstract: the butterfly PUF protecting IP on every FPGA. In: *Proceedings of the IEEE international workshop on hardware-oriented security and Trust*, 2008 (HOST 2008). Anaheim, CA, USA, June 2008, pp 67–70
48. Gassend B, Lim D, Clarke D, Van Dijk M, Devadas S (2004) Identification and authentication of integrated circuits. *Concurrency Comput Pract Exper* 16(11):1077–1098
49. Lee JW, Lim D, Gassend B, Suh GE, Dijk MV, Devadas S (2004) A technique to build a secret key in integrated circuits for identification and authentication applications. In: *Digest of Technical Papers, IEEE 2004 VLSI Circuits Symposium*, 17–19 June 2004, pp 176–179
50. Lofstrom K, Daasch WR, Taylor D (2000) IC identification circuits using device mismatch. In: *IEEE digest of technical papers*, (2000) international solid state circuits conference. IEEE, San Francisco, CA, USA. February, 2000, pp 372–373
51. Puntin D, Stanzione S, Iannaccone G (2008) CMOS unclonable system for secure authentication based on device variability. *Conference on solid-state circuits 2008*:130–133
52. Ruhrmair U, Jaeger C, Bator M, Stutzmann M, Lugli P, Csaba G (2011) Applications of high-capacity crossbar memories in cryptography. *IEEE Trans Nanotech* 10(3):489–498
53. Ganta D, Vivekraj V, Priya K, Nazhandali L (2011) A highly stable leakage-based silicon physical unclonable functions. *IEEE 2011 24th international conference on VLSI design*. Chennai, India, 2–7 January 2011, pp 135–140
54. Helinski R, Acharyya D, Plusquellic J (2009) Physical unclonable function defined using power distribution system equivalent resistance variations. In: *46th ACM/IEEE design automation conference*. San Francisco, CA, USA 26–31 July 2009, pp 676–681

55. Ismari D, Plusquellic J (2014) IP-level implementation of a resistance-based physical unclonable function. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST, 2014). Arlington, VA, USA, 6–7 May 2014, pp 64–69
56. Chakraborty R, Lamech C, Acharyya D, Plusquellic J (2013) A transmission gate physical unclonable function and on-chip voltage-to-digital conversion technique. In: IEEE 2013 50th ACM/EDAC/IEEE design automation conference (DAC, 2013). Austin, TX, USA, 29 May–7 June 2013, pp 1–10
57. Che W, Saqib F, Plusquellic J (2015) PUF-based authentication, invited paper, international conference on computer aided design, November 2015, pp 337–344
58. Zheng Y, Krishna AR, Bhunia S (2013) ScanPUF: robust ultralow-overhead PUF using scan chain. In: IEEE 2013 18th Asia and South Pacific design automation conference (ASP-DAC, 2013). Yokohama, Japan, 22–25 January 2013, pp 626–631
59. Rahman T, Forte D, Fahrny J, Tehranipoor M (2014) ARO-PUF: An aging-resistant ring-oscillator PUF design. In: IEEE design, automation, and test in Europe conference, 2014 (DATE, 2014). Dresden, Germany 24–28 March 2014, pp 1–6
60. Rührmair U, Sehnke F, Sölter J, Dror G, Devadas S, Schmidhuber J (2010) Modeling attacks on physical unclonable functions. In: Proceedings of the 17th ACM conference computer and communications security 2010 (CCS '10), pp.237–249
61. Rahman MT, Xiao K, Forte D, Zhang X, Shi J, Tehranipoor M (2014) TI-TRNG: technology independent true random number generator. In: 2014 51st ACM/EDAC/IEEE design automation conference (DAC 2014). San Francisco, CA, USA, June 2014, pp 1–6
62. Wendt JB, Potkonjak M (2014) Hardware obfuscation using PUF-based logic. In: 2014 IEEE/ACM international conference on computer-aided design (ICCAD 2014). San Jose, CA, USA, 2–6 November 2014, pp 270–271
63. Alkabani Y, Koushanfar F, Potkonjak M (2007) Remote activation of ICs for piracy prevention and digital right management. In: Proceedings of the IEEE/ATM international conference on computer-aided design (CAD), 2007. San Jose, CA, USA, 4–8 November 2007, pp 674–677
64. Eichelberger EB, Williams TW (1977) A logic design structure for LSI testability. In: Proceedings of the design automatic conference (DAC), pp 462–468

Hardware Protection through Obfuscation

Forte, D.; Bhunia, S.; Tehranipoor, M.M. (Eds.)

2017, XII, 349 p. 148 illus., 121 illus. in color.,

Hardcover

ISBN: 978-3-319-49018-2