Chapter 1
FPGA-oriented Security

Mehrdad Majzoobi[†], Farinaz Koushanfar[†], Miodrag Potkonjak[‡]

Abstract Reconfigurable hardware is by far the most dominant implementation platform in terms of the number of designs per year. During the past decade, security has emerged as a premier design metrics with an ever increasing scope. Our objective is to identify and survey the most important issues related to FPGA security. Instead of insisting on comprehensiveness, we focus on a number of techniques that have the highest potential for conceptual breakthroughs or for the practical widespread adoption. Our emphasis is on security primitives (PUFs and TRNGs), analysis of potential vulnerabilities of FPGA synthesis flow, digital rights management, and FPGA-based applied algorithmic cryptography. We also discuss the most popular and a selection of recent research directions related to FPGA-based security platforms. Specifically, we identify and discuss a number of classical and emerging exciting FPGA-based security research and development directions.

1.1 Introduction

The last decade and in the particular the last year were important for FPGAs and even more for FPGA security. For example, for the first time after a decade of no increase, the FPGA revenues grew by more than one third to surpass the $4 B level. Maybe even more importantly, the number of new designs based on FPGA was 110,000. The colossal size of this number can be best seen from the fact that only 2,500 ASIC designs were initiated. At the same time, FPGA has been recognized as an exceptionally efficient platform

[†]Electrical and Computer Engineering Department, Rice University

[‡]Computer Science Department, University of California Los Angeles

due to its flexibility compared to ASICs , and due to its efficiency compared to implementations based on the general purpose microprocessors.

The FPGA security scope is very broad and ranges from technological and architectural issues to applications, from FPGA vulnerability to new types of security primitives and protocols, from relative limitations of FPGA-based systems in terms of security to their strategic and quantitative advantages, and from digital right management (DRM) issues to trusted remote execution. Our objective is to cover various key aspects of this broad space.

Recently several relevant FPGA security surveys have been published, including [1]. We believe that our survey is complementary to the available summaries in the field while it is unique both in terms of the scope as well as the depth of coverage of key issues. In addition, we have a strong emphasis on hardware-based security.

The remainder of the chapter is organized as follows. The next section outlines the steps of the reconfigurable platform's synthesis flow and its vulnerabilities. Section 1.3 discusses the implementation of hardware cryptographic modules on FPGAs and addresses the relevant attacks. The security primitives that can be used as universal mechanisms for many different protection protocols are discussed in Section 1.4. Important primitives such as physical unclonable functions and true random number generation (for the reconfigurable platform) are presented in this section. In Section 1.5 we outline the most challenging directions in the field and early results along those directions. The chapter is concluded in Section 1.6.

1.2 FPGA Synthesis Flow and Its Vulnerabilities

Efficient design and field usage of FPGAs is enabled by sophisticated computer-aided design (CAD) tools that have matured over the years. To make their devices accessible and easy to use, the FPGA vendors and third party providers contribute a full set of programs and tools that allow automatic synthesis and compilation from a high level hardware description language such as Verilog or VHDL to a string of bits, commonly termed a *bitstream*.

The FPGA Synthesis flow is shown in Figure 1.1. The input to the synthesis flow is the hardware specification, design constraints, and sometimes some FPGA-specific commands. The set of inputs is symbolically denoted by HDL (hardware Description Language) on the flow figure but it contains the aforementioned knowledge of design constraints and specifications. The design constraints include the timing bounds between the input and output pads, between the inputs and the registers, and between the registers and the outputs. The designer may also specify additional details such as mutlicycle paths. Another set of possible constraints are location-dependent where a designer may limit the implementation of a specific part of the design to

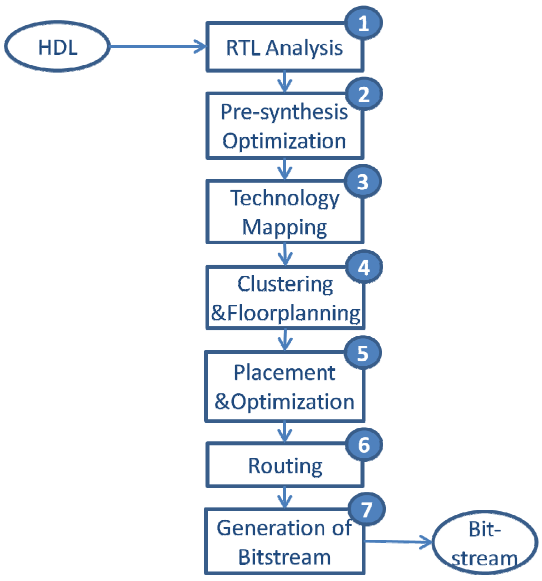


Fig. 1.1 The FPGA synthesis flow.

a target part of the device to achieve a certain design objective or for an optimization reason. The FPGA specific commands specify the selection of the underlying FPGA device that also impacts the design metrics including the timing, power, and cost. Even the same manufacturer often offers a range of devices with differing characteristics that has to be carefully selected for the application at hand.

Although there has been a trend toward using higher level abstraction models such as SystemC and behavioral synthesis tools, they are yet to be widely adopted. The legacy IPs and contemporary designs that are used across a spectrum of applications in industrial, commercial, and defense sectors are predominantly designed at the RTL level. A relatively small number of designs are developed using higher level behavioral languages including but not limited to general purpose languages such as C or SystemC, or domain-specific languages such as Matlab or Simulink. The behavioral-level specifications are not cycle accurate and generally a high level synthesis tool is used for converting the description to HDL level.

Consider the steps of the design flow as shown in Figure 1.1 after the HDL input, design constraints, and the specifications are provided. First, a set of analysis at the register-transfer level (RTL) takes place where the control, memory, and the data path elements are considered. Second, a set of pre-synthesis optimization separately treats each of the identified elements. For example, the datapath optimizations, the control path optimizations including the FSM optimization and retiming, and combinational logic optimizations. Third, the design passes through technology mapping and more detailed optimizations. The control logic is mapped to the basic logic elements. The datapath logic is mapped mostly to dedicated on-chip modules including the multipliers, adders with dedicated carry chains, and embedded memory.

Forth, the location of each element in the floorplan of the mapped netlist is determined. The basic logic elements maybe clustered into logic blocks before the floorplanning. Fifth, the placement is originally done according to the floorplan which is subject to a number of optimization steps. The optimizations are incrementally done post interconnect placement where a better timing profile becomes available. The optimizations at this stage include rewiring, restructuring, and duplication after which typically another round of incremental placement takes place. Sixth, the routing is performed where the signal paths are connected using the available on-chip programmable routing structure. Lastly, the results of mapping, placements, and routing are encoded into a bitstream that can be used to configure the logic and wires to implement the target design. A comprehensive review of the FPGA design automation can be found in [2].

1.2.1 Vulnerabilities

There is a number of possible attacks that can be envisioned on the design flow and the design artifact described earlier in the section. We now briefly mention the plausible adversarial acts and the common generic countermeasures taken against the vulnerabilities. Note that the emphasis of this section is on the attacks that are specific to FPGAs; there is a number of vulnerabilities that apply to most implementations of cryptographic functions, such as system-level attacks on the protocols. In the interest of brevity and conciseness, we focus our discussions to the FPGA domain. Before we delve into discussion, we make a distinction between three types of IPs: soft, firm, and hard IPs. According to the standard definitions since an IP in a hardware description language is still in a program format, is considered to be a “soft IP”. The phrase “firm IP ” is used to refer to an IP that is still pre-synthesis but has a fixed RTL level description. A “Hard IP” refers to the IP that is in form of a full layout post synthesis, placement, and routing, that is ready to be implemented (a bitstream in case of an FPGA) [3].

1.2.1.1 HDL-Level IP Theft and Tampering

Attacks at the HDL level include stealing the IP, inserting malware in an IP to disrupt its operation, or inserting malware/spyware to extract information and data out of the IP.

The common methods for addressing the attacks against stealing of the soft IP cores include watermarking of the soft IP, license agreement, and encryption of the cores that are transferred between parties. Since the soft IP by itself is just a datafile, any other method that is applied to transferring and storage of data files can be used for protecting the transfer and safeguarding of this kind of information. The Trojans/spyware inserted at the HDL level code are either trivial or very hard to detect, based on the availability of designer's information and trust in the designer. It is worth noting that the designer inserted Trojans are very hard to detect in very complex codes, and even the best verification tools may not be able to detect the additional states and functions added by a designer [4, 5]. Often times, the designer does not provide the full specification of the IPs, and therefore, there may not be a basis for comparing the soft IP at hand to a Trojan-free (golden) model. If the designer is trusted, standard cryptographic protocols for integrity checking (e.g., digital signatures) can be applied for ensuring that the original designer's code is not modified. In the final section, we discuss the recent efforts for creation of provably trusted IP.

If the user of an HDL code acquires the program from a certified vendor that has the certificates and can show integrity proofs, there is no need to worry about the HDL-level Trojans. Unfortunately, such proofs and certificates are not always available for third-party IP and reuse scenarios. Therefore, the soft IP trust is a standing complex problem that is not necessarily specific to FPGA; it is a universal problem that threatens almost all soft IP cores that are not from trusted sources or certified vendors.

Aside from classic encryption [6, 7], another set of methods for thwarting the soft IP theft and piracy attacks is based on watermarking [3]. Watermarking hides a hard to forge or remove digital signature in the IP, such that the owner of the datafile can be later recognized based on his/her signature [8]. Methods applied to watermarking during pre- or during synthesis can be directly integrated within the FPGA synthesis tools. Generally speaking, a watermark may be applied at the HDL level, at the netlist level, or at the bitstream level. Depending on the insertion point of the watermark, it can provide a proof of ownership for the legitimate author. For example, an HDL level watermark may be inserted by the core designer, while a bitstream level watermark is likely to be embedded by the tool vendor who is able to easily integrate the watermark within the synthesis flow.

The work in [9] provided the first known methods for FPGA bitstream watermarking and fingerprinting. Fingerprint is a mark that not only identifies the design owner, but also is able to identify the instance of the design. In the FPGA case it can identify the specific device where the design is embedded. Note that the watermark and fingerprint have to satisfy a number of properties including difficulty of forging, difficulty of tampering or removal, uniqueness of the signature sequence and ease of evaluation. A detailed discussion of hardware IP and FPGA core watermarking and fingerprinting is outside the scope of this chapter. We refer the interested readers to excellent comprehensive sources on the topic [10, 3, 1] and Chapter 9 of this book.

1.2.1.2 Synthesis-Level IP Theft and Tampering

By synthesis level IP theft we mean all the stages between the RTL level descriptions to routing (Steps 1-7 in Figure 1.1). Both firm and hard IPs may also be a subject of piracy and malware insertion attacks. A suit of methods based on watermarking can provide ownership proof, but would not be enough to actively deter from piracy. A class of methods that is intended to functionally deter firm IP theft is called active hardware metering [11, 12, 13]. Active hardware metering integrated the behavioral description of a design with the unclonable device-specific signatures such that the IP is only tied to one IC. Transferring the IP to another IC would render the device nonfunctional. For a comprehensive discussion on metering, we refer the interested readers to Chapter 8 of this book.

Another set of IP protection methods based on the use of PUFs attempt at using the inherent and unclonable physical disorders of the device for generating a secret key based on the unclonable device variations. Thorough discussion of IP control based on the PUF signatures is provided in Chapter 7 of this book. A number of defense studies and industrial reports have expressed concerns about the possibility of insertion of hardware malware during the design. Following the suggestion by a Defense Science Board Report [14] and the followup proposal solicitations by DARPA [15], the common trust model in the field became trusted designer (system integrator), untrusted optimization and synthesis tools, untrusted third party cores, and untrusted components-off-the-shelf. The common assumption is that the devices can be trustfully tested for functionality and to ensure they carry on the intended computations, and it can be tested for Trojan detection. A full discussion of Trojan models, detection, and isolation is provided in Chapters 15, 16, and 17 of this book.

1.2.1.3 Bitstream-Level Theft and Tampering

The circuit configuration data is encoded into the bitstream. In the widely used SRAM FPGA technology, because of the underlying volatile memory, at each power up incident the device should read and load the bitstream from an external non-volatile source, typically a Flash device or an EEPROM [6]. The uploaded bitstream typically goes under the functional and parametric tests before being shipped to the users. From this point on, the only active interaction between the provider and the user is via occasional updates by field reconfiguration that can be remotely performed [16]. The common threat model in this area is to assume that the user maybe untrusted [15].

The conventional bitstream uploading methods are independent of the FPGA device, as long as the device is from a certain family and of the same size. Therefore, an adversary could launch an attack targeted at tapping the bitstream during the upload phase and later cloning the stream on other FPGAs. Cloning has been shown to be practically feasible and inexpensive to do for skillful engineers with conventional devices such as probes and logic analyzers. Not only cloning and overbuilding harms the revenue of the original design company, but also the counterfeit devices are often of lower quality and could cause system reliability problems.

Device counterfeiting may also be done at the hardware level, by mislabeling the devices. A common attack is to mislabel a lower quality or an earlier generation device to the current generation. The two generations can be distinguished by structural tests, but such tests are difficult to conduct infield and most customers cannot afford the time and expenses of the testing equipment. The chips are likely indistinguishable based on the functional tests since the input/output specifications (and not performance) of the two chips would be similar. The exact statistics for the percentage of counterfeit components is not exactly known; a few years ago, the Alliance for Gray Market and Counterfeit Abatement (AGMA) estimated that about 10% of the electronic products on the market are counterfeit [17]. It was also reported that the percentage of counterfeit components are growing, emerging as a serious threat to the Integrated Circuits and electronics market.

Another potential form of tampering with the bitstream is *reverse-engineering*. The detailed format of the bitstream for a specific FPGA family is typically considered proprietary to the vendor. Even though the bitstream generation or device configuration details are not commonly published and the complexity of the designs often deters a full reversal of the bitstream, the bitstream alone does not provide any provable security. In some sense, vendor specific bitstream generation only provides a level of obscurity, that is not sufficient for providing protection against reverse-engineering. Given enough time and learning algorithms, bitstream reverse engineering is computationally feasible. Therefore, hiding data and information in the bitstream (i.e., security by obscurity) does not yield a strong protection guarantee.

Full bitstream reversal would expose the IP to unintended parities. Even though the authors are not aware of any tool or method that would offer a full reversal of FPGA bitstream at the time of writing this article, partial reversals of FPGA bitstream were reported earlier. As an example, about 20 years ago, a startup Clear Logic used Altera's bitstreams to produce smaller and cheaper laser programmed devices; however, they had to halt their operations because of a successful lawsuit by Altera [1, 18, 19].

Partial decoding of the bitstream data is also possible by studying the RAM and LUT content [20, 21, 22]. An example of how this can be done is reported by Ulogic project that attempted an iterative process that manipulates the available files in the Xilinx Design Language (XDL) format and partial conversion to bitstream. It is also possible to perform a *read-back* function, which is the process of retrieving a snapshot of an operating FPGA's present state. Note that this snapshot just gives the information about configuration and states in one moment and is different from the original bitstream. However, this mechanism, if repeatedly applied, provides an effective characterization tool for testing, verification, and also partial reverse-engineering [1].

1.3 FPGA-based Applied Cryptography

With the proliferation of personal computing, mobile devices, and Internet, and with the booming of the global information and knowledge, storing and processing of digital functions and data increasingly demands new computing devices. Since many of these devices and services are integrated within our daily lives and our personal data, it is not surprising that protection and security are needed in several key applications, including Internet, secure email, secure wireless access, data centers, electronic financial transactions, and grid computing. As a result, several National and International organizations have been working on developing standards for protecting these applications, such as Advances Encryption Standard (AES), Elliptic Curve Cryptography (ECC) and the recent NIST efforts for standardizing the next generation hash functions [23].

Processing of cryptographic algorithms often takes a large amount of system processing time and resources especially for cases where a large amount of data and information is involved, or where the platform is power constrained to satisfy portability and mobility [23]. Furthermore, many applications require real-time secure processing of data which places additional constraints on the system and processor timing. As a result, in many real world scenarios, the hardware implementation is preferred over software. The comparable high throughput and power efficiency of hardcoded modules compared to their programmable counterparts makes the hardware the natural choice in such scenarios.

It is worth noting that while a software implementation is not the most performance efficient option, it is inexpensive, easy to debug, and induces a short time to market. VLSI hardware solutions provide the high throughput and power efficiency, but they are expensive, they have a long development cycle, and they do not provide much flexibility for design alterations. the reconfigurable hardware has become the platform of choice for many cryptographic modules and security processing tasks. This is because of FPGA robustness, comparative low-cost, and shorter time-to-market compared with the ASICs solutions, simultaneously combined with reconfigurable device throughput and power advantages compared with the software and general purpose computing solutions.

There are a number of other reasons for selecting reconfigurable solutions for cryptography and security applications, including: (i) the effectiveness of the FPGA's cell structure for implementing bit-wise logical operations that are needed in many cryptographic algorithms; (ii) the large amount of memory blocks built-in the state-of-the-art FPGA devices that ease the implementation of memory intensive substitution operation required by the standard encryption algorithms; (iii) the reconfigurable platforms that not only eases interfacing of the security cores to other cores on the same device by allowing reprogrammability, but also provides a flexible solution that can be integrated into a larger platform with other components.

1.3.1 Vulnerabilities

The standard cryptographic algorithms are designed to be secure against algorithmic attacks that target the steps and flows of the security procedure. Unfortunately, while conventional cryptography methods have been resilient to attacks on the security algorithm, they have been demonstrated to be vulnerable to attacks that target some aspects of their implementation, including the side-channels, fault injection, and physical attacks. The security cores programmed as softcore, reconfigured on FPGA, or realized in ASIC have all been target of implementation-level attacks. In the remainder of this subsection, we briefly mention the attacks and provide references for further reading on the subject.

1.3.1.1 Side-Channel Attacks

Once a reconfigurable device is programmed to function as a certain circuit, it is possible to extract external measurable manifestations of the incident computations performed in the circuit. The term side-channel is used to refer to quantities that can be measured from the circuit in operation; those measured external quantities are correlated with the circuit computations, and therefore, could provide additional (side-channel) information about the internal circuit values. Examples of common side-channels used for attacking the secure hardware cores include power analysis, timing analysis, and electromagnetic emanation analysis. In all cases, multiple measurements of the side-channel for different inputs and in different conditions are needed. An important performance measure for the side-channel attacks is the amount of useful information one can get from each round of attack, and the number of required inputs/outputs to successfully accomplish the attack's objectives.
Power analysis The CMOS gates consume two types of power: static and dynamic. The static (leakage) is the power leaked away because of the device imperfections. For each gate, the leakage power is a function of the

gate-type and its incident input vector. The dynamic (switching) power is incurred when the state of one gate transitions from one value to the next. The dynamic power for each gate is also a function of the gate type and the transition input incident to the gate. Both the static and dynamic power can be externally measured by monitoring the current drawn from the circuit's supply pins.

Generic dynamic power measurement results on the widely used SRAM-based FPGAs had demonstrated that a significant portion of the transitional power on those devices is due to the interconnect routing, while the logic switching and clock transitions composed the remaining parts of dynamic power consumed. The leakage power for the logic was not a significant portion in earlier technologies, but the aggressive miniaturization of transistors is drastically increasing the static power significance in newer technologies [24]. The early work in [25] demonstrated that both simple power analysis (SPA) and differential power analysis (DPA) could reveal information about the secret values and operations performed during the execution of cryptographic operations on FPGA. In SPA, the patterns in the power traces incident to individual inputs are processed. In DPA, the differences between the power trace patterns of two or more input sets are processed. A large body of work on attacking the chips based on SPA and DPA has followed, including [26, 27, 28, 29, 30, 31].

Simultaneously, many researchers are working on developing countermeasures against the power analysis attacks [32, 33]. It was shown that if the core is not run in isolation and if there are other sources or cores in the circuit contributing to the power, or even when the core is run in parallel, it is harder to distinguish the contributions of each component. In general, the power analysis attack can be thwarted if the functions that depend on the secret values and information have the same power signature as other operations. Following this principle, two effective countermeasures against the power analysis attacks are: (i) randomization so that the impact of one computation cannot be easily distinguished among the many operations, and (ii) equalization such that all computations consume the same amount of power. For each implementation, both methods incur undesirable power and timing overheads which needs to be mitigated while there is also a need to provide proofs for efficiently obfuscating the secret values. Both overhead mitigation and proof of hiding (randomness) are active research topics.

Timing analysis The gate timing is also a function of its type and internal values. It was shown that by careful path timing signature measurements, one could be able to reveal the secret values that are processed by the gates [34, 35]. The countermeasures for this type of attack are similar in nature to power analysis, and consist of timing equalization and timing randomization. Both methods may incur additional overhead and should be carefully studied and analyzed.

Electromagnetic emanation analysis The movement of electrons during the execution of computations would generate electromagnetic field that can be externally measured by placing antennas outside the chip. Electromagnetic emanation analysis (EMA) was shown to be able to successfully predict the secret values and computations done while executing the security functions [36, 37, 38, 39, 40, 41]. Such attacks were also reported for FPGAs. Most countermeasures against this attack are based on disturbing the EM field by changing the device properties or by adding layers. These methods cannot be directly applied to conventional reconfigurable hardware. The proposed methods for thwarting this attack on FPGA rely on distributing the computations across the FPGA area to avoid localizing the events. Last but not least, we note that it was demonstrated that by combining multiple side-channels, one may be able to launch a much stronger attack [42, 43].

1.3.1.2 Fault Injection Attacks

Several forms of operational faults can be induced in circuits performing the secure processing. A fault maybe generated by a number of methods, including controlling of the voltage, inducing an electromagnetic field close to the device, or exposing the device to radiations. If carefully injected, such faults can reveal aspects of the secret. We briefly mention some of the important ongoing work in this area that are also applicable to FPGAs.

Glitch analysis The objective of such analysis is to force a device to execute faulty operation(s), or to leave the device in a state that can lead to leaking of secret information. The common techniques for induction of glitch include changing the external clock, and altering the supply voltage. Such attacks were shown to be successful on microcontrollers [44], and if not carefully considered, they can be adopted for FPGA and ASIC implementations. An effective countermeasure against this attack is to ensure that all the states are properly defined in models and in implementation, and to verify that the glitches cannot alter the execution order of the events. Another class of countermeasures is to avoid fault injection by implementing tamper detection mechanisms that would report (or prevent, or even correct) altering of clock pulses or voltage levels.

Ionizing radiation analysis Radiation-induced faults have shown to cause single-event upsets in the CMOS circuits [45, 46, 47]. Such single (or multiple) event upsets may cause transient delay faults, or may cause the memory bits to flip (known as soft errors). Since the FPGAs are SRAM-based, such memory flips would alter the device's functionality. Ionizing radiation is a way to induce the faults and hence, change the memory content. If targeted accurately, it could be used for changing the secret, or to trace back a secret. The complexity of the integrated circuits and small size of the individual components renders this attack very difficult. Many methods for detection and removal of soft-errors are in development which could additionally deter this type of attacks.

1.3.1.3 Physical Attacks

For an attacker with access to costly and high precision measurement and testing equipment, it is possible to physically probe or alter the device so that the secret information can be extracted [48]. There are at least two major hurdles in performing such an invasive probing. First, very costly higher precision Focused Ion Beam (FIB) measurement equipments are needed to precisely target the specific parts of the chip [49]. Second, the device has to be depackaged and the passivation layers that protect the metal interconnects from oxidation needs to be removed. Depackaging and delayering is challenging for certain class of package technology and interconnect deposition methods. Miniaturization of CMOS to nanometer scales and the added layers of interconnect are rendering this attack extremely difficult for newer technology nodes.

There is also a possibility of performing a *semi-invasive* physical attack. These attacks also need the device packaging to be removed, but then they adopt techniques from thermal analysis, imaging, and other side-channel studies to conclude the properties of the chip [48]. Unlike the invasive attacks that need very costly equipments mainly owned by governments or mega-companies, the semi-invasive attacks are much less costly and more accessible to general public. It is worth noting that both invasive and semi-invasive attacks pose real threats to electronics and new methods for thwarting and circumventing these attacks are under research and development.

1.4 FPGA Hardware Security Primitives

Security on reconfigurable platforms has emerged as a challenging security paradigm in system design. Systems implemented on FPGAs like any other systems could require secure operations and communications. However, as we discussed in the previous section, on reconfigurable systems in addition to concerns regarding the compromise of data confidentiality and integrity, the system itself can be subject to malicious architectural alterations to the hardware and to design theft during the operation or even before the design is loaded. As a result, it is critical to establish security of configuration data and maintain design integrity against malicious changes. Several existing solutions govern different trade-offs between security and the market requirements on cost and performance. In this section, we discuss a number of mechanisms and protocols that can be used as the underlying primitives for many FPGA security protocols and modules.

Every FPGA relies on certain programming technology that enables the control and configuration of programmable switches inside the FPGA which in turn program the functionality of the underlying logic. Historically used programming technologies include EPROM [50], EEPROM [51, 52], flash [53], static memory (SRAM) [54], anti-fuse [55, 56]. Among these technologies, mainly the flash memory, the static memory, and the anti-fuse are used in modern FPGA devices.

The dominant family of FPGAs is realized using volatile SRAM-based memories. Upon power-up, the FPGA is configured by loading and storing the desired functionality inside the SRAM memory cells. The SRAM cell values define the logic functions by means of initializing a set of truth tables or *lookup tables (LUT)* and by enabling/disabling connections through switch matrices. Once the FPGA is powered off, the content of the SRAM cells is lost. In other words, the SRAM-based FPGAs must be constantly powered to retain the configured functionality and they need to be reprogrammed every time the power is lost.

The lack of non-volatile embedded storage mechanisms on SRAM-based FPGAs thwarts permanent storage of secret keys which is required to establish a secure channel for sending the configuration data. Without the use of encryption, the configuration bitstream has to be communicated to the FPGA at start-up through a non-secure channel. This is specially important in applications in which systems and IPs must be protected against piracy or unauthorized read-out as well as against malicious changes to tweak the system functionality.

Integration of non-volatile memory on SRAM-based FPGAs is costly because integration of state-of-the-art non-volatile technologies on standard CMOS process requires more complicated fabrication steps and wafer processing. As a result, non-volatile storage is often not available on lower-end devices [6]. In order to store keys on SRAM-based FPGA, an external battery is typically attached to the device to constantly provide energy to the SRAM cells containing the secret key (s). The concept is shown in Figure 1.2.

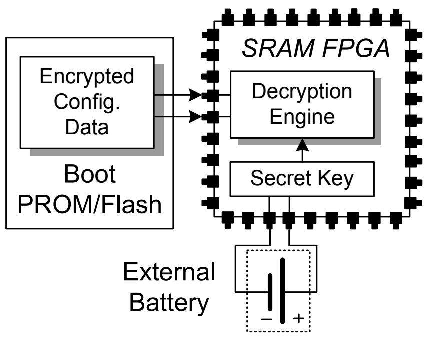


Fig. 1.2 Embedded key storage on SRAM-based FPGAs.

Antifuse technology uses a layer of amorphous silicon in the via, which causes an isolation between the metal layers [57]. In the un-programmed state, the amorphous silicon has very high resistance, thus isolating the metal layers. After programming voltage is applied, the amorphous silicon resistance drops significantly, creating a metal to metal interconnect. Compared to other technologies and even ASICs, the antifuse FPGAs enjoys the highest level of security, because of the following reasons: (i) Since the FPGA can be configured once and shipped by the system designer to the end-user, there's no need to transfer the configuration over an insecure channel. (ii) The fabric of the FPGA (i.e., the interconnection, routing and placement of the programmable elements) reveals no information about the design (in contrast with ASICs). This is due to the fact that all the design data is internal to the device and it is stored at programmable links. Invasive reverse engineering methods such as etching that take away the surface will only reveal the top of the vias and not the state of the amorphous antifuse silicon; thus, such techniques do not expose much information on the chip functionality. Non-invasive attacks that use advanced imaging and probing techniques such as SEM theoretically might have a chance to monitor the device. The imaging technique attempt to determined the state of antifuse links by looking for any deformations in the amorphous silicon vias. With millions of links on each device, it is still not an easy task to scan every single link of the FPGA. For example, Actel's AX2000 antifuse FPGA contains approximately 53 million antifuse elements.

Since anti-fuse FPGAs can only be programmed once, it takes away a great advantage of in-field FPGA reconfigurability feature. Table 1.1 summarizes the properties of different programming technologies.

	SRAM	Flash	Anti-fuse
Volatile?	Yes	Yes	No
Reprogrammable?	Yes	Yes	No
Area	High	Moderate	Low
Power	High	Low	Low
Manufacturing Process	Standard CMOS	Flash Process	Special development
Programming yield?	100%	100%	> 90%
Security	Low	Moderate	High

Table 1.1 Comparison of current programmable technologies.

In the rest of this section, we focus our attention on SRAM FPGAs since they currently have the largest market share in the reconfigurable hardware domain.

1.4.1 Physical Unclonable Function (PUF)

Physical Unclonable Functions (PUFs) provide an alternative mechanism for key storage on SRAM-based FPGAs. PUFs overcome the inherent vulnerability of key storage on non-volatile memories against various attacks as well as the extra technology cost overhead of nonvolatile memory integration onto SRAM-based devices. PUFs use the inherent and embedded nano- and micro-scale randomness in silicon device physics to establish and define a secret which is physically tied to the hardware. The randomness is introduced by the existing uncertainty and lack of precise control during the fabrication process that lead to variations in device dimensions, doping, and material quality. The variation in device physics transfers itself into variations in electrical properties, such as transistor drive current, threshold voltages, capacitance and inductance parasitics. Such variations are unique for each IC and device on each IC. PUFs typically accepts a set of input challenges and map them to a set of output responses. The mapping is a function of the unique device-dependent characteristics. Therefore, the responses two PUFs on two different chips produce to the same set of inputs are different. A comprehensive review of PUF concept and literature is provided in Chapter 7 of this book. In the remainder of this chapter, we focus on the work covering the FPGA PUFs. Our discussions are complementary to the material presented in the earlier chapter.

A common way to build a PUF in both ASICs and FPGAs is by measuring, comparing, and quantifying the propagation delays across the logic elements and interconnects. The variations in delays appears in forms of clock skews on clock network, jitter noise on the clock, variations in setup and hold times of flipflops, and the propagation path delays through the combinational logics.

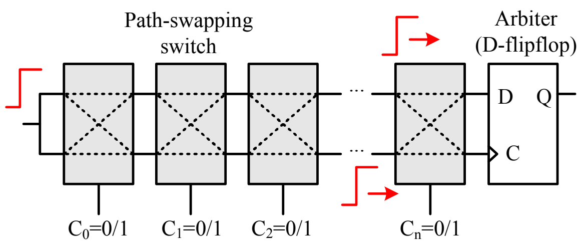
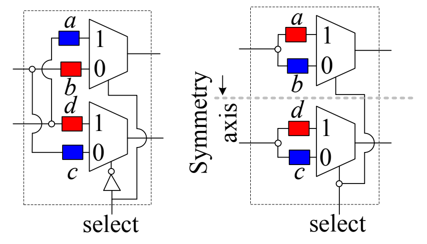


Fig. 1.3 Arbiter-based PUF introduced in [58].

The work in [58] was the first to exploit the unique and unclonable delay variations of silicon devices for PUF formation. The PUF, known as arbiter PUF or delay-based PUF, is shown in Figure 1.3. The PUF uses the analog differences between the delays of two parallel paths that are identical in design and prior to fabrication, but the physical device imperfections make the delays different. Beginning the operations, a rising transition is exert at the PUF input producing a racing condition on the parallel paths. An arbiter at the end of the paths generates binary responses based on the signal arrival times. To enable multiple path combinations and generate an exponential number of challenge/response pairs, the paths are divided into multiple subpaths interleaved by a set of path swapping switches. The challenges to the PUF control the switches and, therefore, how the varying paths are formed.



(a) Asymmetric path-swapping switch (b) Symmetric non path-swapping switch

Fig. 1.4 Two implementation of path selecting switches.

A successful implementation of this type of PUF was demonstrated on ASICs platforms [59]. It is critical to note that the differences in delays should be solely coming from manufacturing variation and not from design-induced biases. To obtain exact symmetry on the signal paths and to equalize the nominal delays, careful and precise custom layout with manual placement and routing is required for implementation on ASICs. The lack of a fine control over arbitrary placement and routing on FPGA has resulted in difficulty in balancing the nominal delays on the racing paths within the arbiter-based PUF. Implementation on FPGA was troubled because of the constraints in routing and placement imposed by the rigid fabric of the FPGA as studied in [60, 61].

However, the recent work in [62] has addressed this problem by demonstrating a working implementation of the arbiter-based PUF on FPGA that utilizes a non-swapping symmetric switch structure as well as a precise programmable delay line (PDL) component to cancel out the systematic delay biases. The path-swapping switch previously used in the arbiter-based PUF of Figure 1.3 can be implemented by two multiplexers (MUX) and one inverter as depicted in Figure 1.4 (b). However, due to cross wiring from the lower half to the upper half (diagonal routing), maintaining symmetry in path lengths for this type of switches is extremely difficult. To avoid diagonal routings, a non-path swapping switch with a similar structure was introduced in [62] which uses two MUXes as shown in Figure 1.4 (a). As it can be seen on the figure, after applying the method the resulting routings and path lengths are symmetric and identical across the symmetry axis (drawn by the dashed line).

Despite using a symmetric switch structure, systematic biases in delay would still exist due to the asymmetries in routing from the last switch to the arbiter flipflop and/or before the first switch. To eliminate such delay skews, a highly accurate programmable delay line (PDL) was introduced in [62]. The PDL was implemented by a single LUT and can achieve a resolution of better than 1 picosecond. The PDL works by slightly increment-

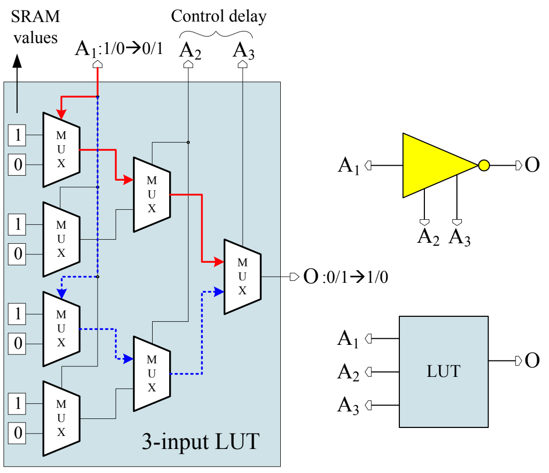


Fig. 1.5 (a) LUT-based programmable delay line (b) symmetric switch structure

ing/decrementing the signal propagation path length inside the LUT. Figure 1.5 shows an example PDL implemented by a three-input LUT. The LUT implements an inverter logic where the output of the LUT reflects negation of A_1. However, the inputs A_2 and A_3 functionally serve as don't-cares while they can change the signal prorogation path inside the LUT and cause slight in the propagation delay.

In contrast to the arbiter-based PUF where racing condition is formed by signal propagation through two independent paths, the FPGA PUF introduced in [63, 64] which referred to as time-bounded PUF, compares the signal propagation speed through a combinational logic against the system clock speed. The time-bounded PUF uses the standard at-speed delay test circuit (delay characterization circuit) shown in Figure 1.6 (a). The at-speed delay test circuit consists of one *launch*, *sample*, and *capture* flipflop. At the rising edge of the clock, the launch flipflop sends a low-to-high signal through the circuit under test (CUT). At the falling edge of the clock the output of the CUT is sampled by the sample flipflop. The steady state output of the CUT is then compared with the sampled value by an XOR logic. If discrepancies exist, it means that the output was sampled before the signal had arrived at the output of CUT. This condition is referred to as timing error. By sweeping the clock frequency in a linear fashion, one can locate the transition point from error free zone to full error zone. The center of the transition corresponds to the delay of CUT.

If the time difference between the sampling time and the signal arrival time is smaller than the setup and hold time of the sample flipflop, then sample flipflop produces non-deterministic outputs. It was shown in [63, 65] the probability of sampling the correct value in this case is a monotonically increasing function of the time difference between the signal arrival time and the sample time. This is depicted in Figure 1.6 (b,c). To estimate the center of this smooth transition curve, statistics on the observed error need to be gathered.

A careful investigation of the characterization circuit reveals that the observability of timing errors go through periodic phases. The measured probability of timing error as a function of half clock period (T/2) on Virtex 5 FPGA is illustrated in Figure 1.7. The two consecutive transitions from 0 to 50% and 50% to 0 (and vise versa) are formed by the differences in propagation delays in rising edge and falling edge signals. The measured probability is the net effect of both transitions. The center and slope of each transition point are unique to each circuit on different FPGAs.

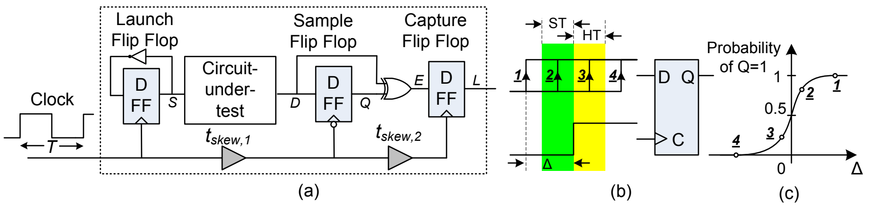


Fig. 1.6 (a) Delay characterization circuit based on at-speed delay testing mechanism (b) sampling signals with different arrival times (c) probability of the flipflop output=1.

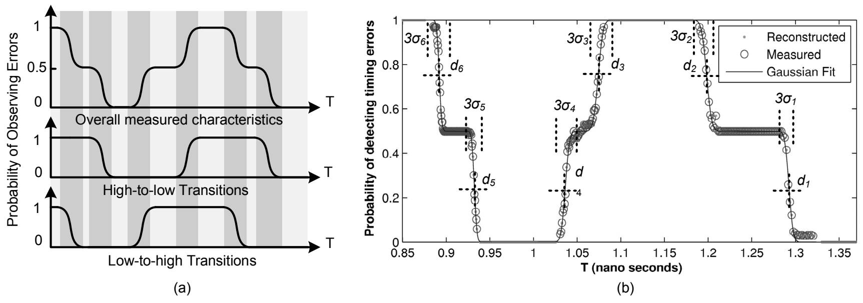
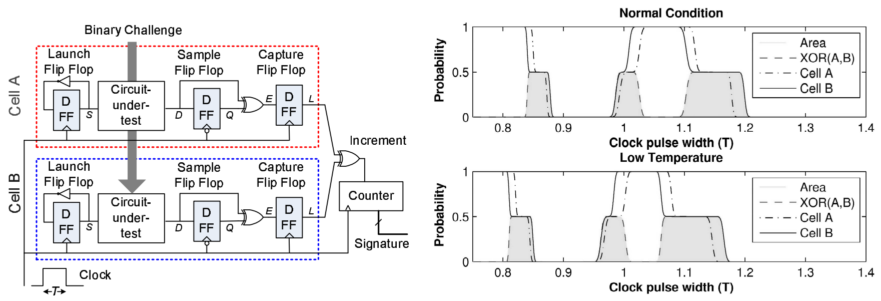


Fig. 1.7 (a) Probability of observing timing error for rising/falling edge transition and both transitions as a function of half clock period (b) Measured probability of transition as a function of half clock period on Virtex 5 FPGAs.

The extracted absolute delay parameters are sensitive to changes in environmental variations. In order to obtain more resilient responses and better signatures against such fluctuations, a method to perform linear calibration of the clock frequency according to the current temperature is introduced in [64]. The operating temperature and voltage are obtained by querying the built-in FPGA sensors, and calibration is performed accordingly on the clock frequency. In addition to frequency calibration, a differential structure is further proposed that cancels out the common effect of environmental variations on delays as shown in Figure 1.8 (a). The differential circuit consists of two at-speed delay test circuit (Figure 1.6) whose outputs are tied to an XOR logic. Since the absolute delays increase/decrease, extracting shift invariant parameters such as the distance between the centers of transition regions (width), or the area under the curve would result in more robust signatures. The circuit in Figure 1.8 (a) measures the area under the XOR probability curve using Riemann sum approximation. As it can be observed on the figure, the area under the measured curves stays the same for low and normal operating temperatures.



(a) Differential timing circuit (b) Symmetric non path-swapping switch

Fig. 1.8 Extracting shift invariant signatures.

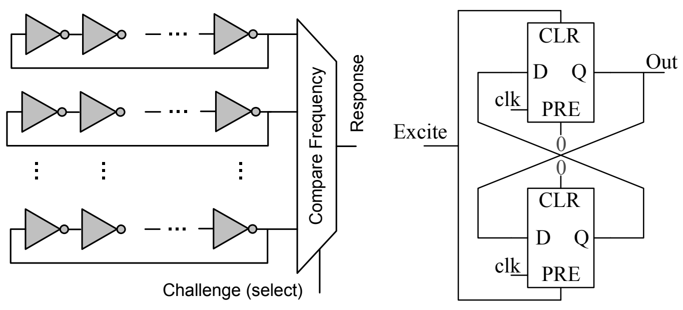
Another family of PUFs amenable to implementation on digital platforms and in particular FPGAs, is based on ring oscillators (RO-PUF). A ring oscillator is composed of an odd number of inverters forming a chain. Due to variations in delays of comprising logic components and interconnects, each ring oscillates at a slightly different frequency. The RO-PUF measures and compares the unique frequency of oscillation within a set of ring oscillators. A typical structure of RO-PUF is shown in Figure 1.4.1 (a). Most of the work around RO-PUFs is focused on post processing techniques, selection, quantization and comparison mechanisms to extract digital responses while achieving robustness of responses and high response entropy.

One of the early papers to consider and study ring oscillators for digital secret generation is [66]. The work proposes a 1-out-of-k mask selection scheme to enhance the reliability of generated response bits. For each k ring oscillator pairs, the pair that has the maximum frequency distance is chosen. It is argued that if the frequency difference between two ring oscillators is big enough, then it is less likely that their difference changes sign in presence of fluctuations in operating temperature or supply voltage.

In order to achieve higher stability and robustness of responses, extra information can be collected by measuring the oscillation frequency under different operating conditions. Methods presented in references [67, 68] use this information to efficiently pair or group the ring oscillators to obtain maximum response entropy. Specifically, frequency measurement is performed at two extreme (low and high) temperatures and a linear model is built to predict the frequency at middle temperature points.

Systematic process variation can adversely affect the ability of RO-PUF for generation of unique responses. A method to improve uniqueness of ring oscillator PUF responses is discussed in [69]. A compensation method is used to mitigate the effect of systematic variation by (i) placing the group of ROs as close as possible (ii) picking the physically adjacent pair of ROs while evaluating a response bit. Large scale characterization of an array of ROs on 125 FPGAs (Spartan3E) is performed in [70]

The existing inherent race conditions in combinatorial logics with feedback loop are also used in development of other types of PUFs. For instance, a loop made of two inverter gates can have two possible states. At the power-up, the system enters into a metastable state that settles onto one of two possible states. In fact, the faster gate will dominate the slower gate and determine the output. The idea of back-to-back inverter loops is used in SRAM memory cells. SRAM-based PUFs based on the inherent race condition and variations in component delays produce unique outputs at startup. Unfortunately, in SRAM-based FPGAs, an automatic internal reset mechanism prevents using the unique startup value. A more practical implementation that is based on the same concept but uses the logic components on FPGA rather than the configuration SRAM cells, is referred to as a butterfly PUF. The basic structure of a butterfly PUF is shown in Figure 1.4.1 (b). Butterfly PUF is made of two D-flipflops with asynchronous preset and reset inputs. The flipflops are treated as combinational logics. The work in [71] presents a comparative analysis of delay based PUF implantations on FPGA. The work particularly focuses on the requirements of maintaining symmetry in routing inside the building blocks of Arbiter-based PUF, Butterfly PUF, and RO-PUF.



(a) RO-PUF (b) Butterfly PUF

Fig. 1.9 Other delay based PUFs

1.4.2 True-Random Number Generator (TRNG)

FPGAs are also suitable platforms for implementing True-Random Number Generators (TRNG). TRNGs are important security primitives that can be used to generate random numbers for tasks such as (i) secret or public keys generation, (ii) initialization vectors and seeds for cryptographic primitives and psuedo-random number generators, (iii) padding bits, and (iv) nonces (number used once). Since modern cryptographic algorithms often require large key sizes, generating the key from a smaller sized seed will significantly reduce the effectiveness of the long keys. In other words, by performing a brute-force attack only on the seed that generated the key, one can break the crypto system. Therefore, it is essential to generate the keys from a high entropy source.

Numerous TRNG designs have been proposed and implemented. Each design uses a difference mechanism to extract randomness from some underlying physical phenomena that exhibit uncertainty or unpredictability (or probably a behavior not well-understood). Examples of sources of randomness include thermal shot noise in circuits, secondary effects such as jitter and metastability in circuits, Brownian motion, atmospheric noise, nuclear decay, random photon behavior. In this chapter, we only focus on TRNGs that are implementable on digital platforms and FPGAs.

In general, TRNGs are evaluated using the following typical parameters and measures: (i) entropy source (source of randomness), (ii) design footprint (area and energy per bit), (iii) predictability of the generated bitstream and its statistical properties, (iv) security and robustness of the generated bits against attacks, and (v) ease of implementation.

As discussed in the previous section, one measurable analog quantity on digital platforms is the signal propagation delay. The circuit noise (thermal, shot, and flicker noise) can exhibit their effect on propagation delays. The noise manifest itself as the jitter and phase noise on the systems clock by causing temporal variations in the oscillator frequency.

The approach in [72] uses sampling of phase jitter in oscillator rings to generate a sequence of random bits. The output of a group of identical ring oscillators are fed to a parity generator function (i.e., multi-input XOR). The parity generator output is then constantly sampled by a D-flipflop driven using the system clock. In absence of noise and identical phases, XOR output would be constant (and deterministic). However, in presence of jitter in phase, glitches with varying non-deterministic lengths appear at the XOR output.

Another type of TRNG is introduced in [73] that is based on the basic arbiter-based PUF structure. Unlike PUFs where reliable response generation is desired, the PUF-based TRNG goal is to generate unstable responses. This is achieved by driving the arbiter into the metastable state essentially through violating the setup/hold time requirement of the arbiter. The PUF-based random number generation method searches for challenges that result

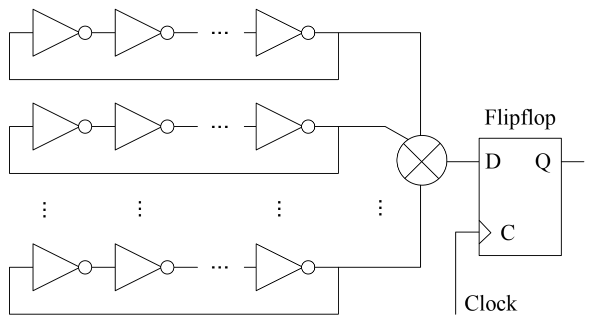


Fig. 1.10 TRNG based on sampling the ring oscillator phase jitter.

in small delay differences at the input of the arbiter which in turn cause highly unreliable response bits.

In order to improve the quality of the output bitsteam and increase the randomness, various post-processing techniques are often performed. [72] introduces resilient functions to filter out deterministic bits. The resilient function is implemented by a linear transformation through a generator matrix commonly used in linear codes. The hardware implementation of resilient function is demonstrated in [74] on Xilinx Virtex II FPGAs. The TRNG after post processing achieves a throughput of 2Mbps using 110 ring oscillators with 3 inverters in each. A post-processing may be as simple as von Neumann corrector [75] or may be more complicated such as extractor function [76] or even a one-way hash function such SHA-1 [77]. Von Neumann method is a well-known post-processing technique to removed localized biases in the generated bit sequence. It looks at pairs of bits in the bitstream. If both bits in the pair are identical, the corrector removes both of them from the sequence. If the bits are different, then it uses only one of them (e.g. the second bit). The bit rate as a result will be reduced to about 1/4 of the input bit rate on average (this is for the optimistic case where 0s and 1s are equally likely).

Besides improving the statistical properties of the output bit sequence and removing biases in probabilities, post-processing techniques increase the TRNG resilience against adversarial manipulation and variations in environmental conditions. An active adversary attacker may attempt to bias the probability of the output bits in order to reduce the entropy of the generated keys. Post-processing techniques typically govern a trade-off between the quality of the generated bit versus the throughput. Other online monitoring techniques may be used to ensure higher quality of the generated random bits. For instance, in [73], the probability of the generated bits are constant monitored and as soon as a bias is observed in the bit sequence, the search for a new challenge vector that produces unreliable response bits is initiated.

Although it is almost impossible to analytically and mathematically prove the unpredictability of the generated bit stream, a simple system design, insight on underlying physical randomness, as well as a thorough examination of the bitstream statistical properties and randomness are fundamental to justify the security of the TRNGs. In other words, it is necessary, although not sufficient, to perform a comprehensive set of statistical tests on the generated bit sequence. A well-known and common suites of randomness tests are outlined in DIEHARD [78] and NIST Test Suites [79].

1.5 Top FPGA Security Challenges

In this section we identify and analyze dominant FPGA research and development challenges and opportunities. The challenges are dictated by technological, application, business models, and tools development trends. We start by discussing our top 15 challenges and finish by analyzing uniqueness of FPGA security requirements and degrees of freedom with respect to ASIC and general purpose and application specific programmable processors. As we already stated, it is obvious that that each platforms has certain advantages or limitation depending on the security threats and goals as well as a set of overall security desiderata. However, it is important to emphasize that flexibility and configuration capabilities of FPGAs may be instrumental for creation of unique classes of security primitive and protocols.

1.5.1 Algorithmic Cryptographic Security

Algorithmic (mathematical) cryptography is one of the most elegant and effective computer science fields. Numerous ingenious and surprising primitives and protocols have proposed, analyzed, and are in practical use [80, 81, 82, 83, 84]. The algorithmic (mathematical) foundations for some protocols such as public-key communication and storage are solid although rarely actual consistent proofs are available. Nevertheless, the chances of breaking modern protocols such as Advanced Encryption Standard (AES) using algorithmic attacks are relatively very small.

However, it is well known that the computer engineering basis of algorithmic security is far less reliable. It has been reported that a variety of physical and side channel attacks easily using inexpensive equipment easily break essentially all algorithmic cryptography protocols. Development of engineering techniques for protection of information leakage is a popular research direction. These techniques are often much less effective for FPGA platforms due to factors such requirements for highly regular routing, relatively sparse and publicly available IC structure, and higher difficulty and cost of TRNG. However, the greatest impediment to any masking technique in particular at the gate level is process variation that prevents matching of gates. In principle, FPGA platforms have here advantage due to their reconfigurability.

Once when the physical security of an FPGA platform is ensured, it would have a high potential for significant energy efficiency, protocols flexibility, and even speed advantages over programmable platforms. In addition, FPGA's ability to facilitate reuse can greatly improve actual security. There is little doubt that at least in short time periods, cost, energy, low latency, and high throughput of algorithmic cryptographical protocols will be of the primary importance.

1.5.2 Hardware-based Cryptography: Primitives and Protocols

Hardware-based security techniques have been going through several partly overlapping phases. Initially, the emphasis was on creation of unique ID. In the next phase, IDs were used for protection of the platform and applications related to the hardware or software running on the platform including -= hardware metering, remote enabling and disabling, and similar tasks. Silicon PUFs initiated a revolution in hardware security [58, 73]. However traditional PUF technique utilize only secret key-based cryptography. More recently several schemes redefined ways how PUFs are constructed and used to enable a variety of public key security protocols. They have been developed under the names of PPUF [13], SIMPL [85, 86], and timed authentication [13, 87, 63]. While the public key PUF approaches have been proposed several years ago, now more and more realistic schemes are analyzed. For example, PPUF-based scheme include not just authentication and public-key private key communication, but also time stamping, place stamping, device stamping, and more demanding protocols such as coin flipping and oblivious transfer. The crucial observation is that FPGAs are ideal platform for many type of primitives and security protocols due to their reconfiguration capabilities.

In addition to hardware primitives, device-level characterization and conditioning play important enabling roles. For example, it has been demonstrated that leakage energy measurement can be used for fast and very accurate gate-level characterization and for provably comprehensive hardware trojans detection [88, 89, 90, 91, 92, 93]. Furthermore, it has been demonstrated that in addition to side channels ways for collecting information, there are device conditioning techniques that can be used to organize accurate and diverse measurements. For example, localized heating can be used for breaking correlation in linear systems of equations in such a way that all gates can be characterized [91]. As another example, very accurate detection bounds based on the submodularity of the objective function can be achieved [90]. These techniques are universal, but the presence on unused hardware on FPGA ICs can additionally facilitate their effectiveness [63, 64]. Finally, hardware primitives can be used for creation of a great variety of security protocols.

It is important to note that they have sharply different principles than the identical algorithm-based security protocols.

1.5.3 Digital Right Management (DRM) of ICs and Tools

There are dominant approaches for protecting hardware IPs. The first is watermarking [94, 95, 96, 97, 98, 99, 100, 8, 101]. Numerous hardware watermarking techniques have been proposed at essentially all levels of design abstraction. The current emphasis in on using of side channels for efficient watermark detection [102]. It is important to emphasize that several of early hardware watermarking techniques enable easy watermark detection through minimal modification of outputs or through augmentation of finite state machines [8, 101]. It is easy to see that IP watermarking is often much more difficult task for FPGAs than for ASICs. This is in particular true for techniques that embed watermarks by superimposing additional constraints on design specification. The main reason is that watermarks on higher levels of ASIC synthesis are naturally protected by the non-recurring engineering (NRE) cost and time-to-market delay. Hardware watermarking is covered in Chapter 9 of this book.

The second task is hardware metering where the goal is to ensure that foundry does not sell unauthorized ICs [103, 104]. There are two broad classes of hardware metering. The first is passive hardware metering where already sold ICs are examined in order to detect illegally produced ICs. Passive metering has techniques are equally difficult for both FPGA and ASIC designs. Active metering techniques do not just detect illegal ICs, but directly enforce DRM rights by requiring specific authentication steps that can be provided only by the designer. While obviously active metering techniques are more effective, they also induce higher operational and design overheads in metrics such as energy or delay. Metering is covered in Chapter 8 of this book.

Finally, the most ambitious DRM step is remote control where the designer or an authorized entity can remotely control which action can or can not be conducted by the user [105, 106, 107, 108]. The stated three types of techniques and many other DRM tasks (e.g. auditing) can be performed on all three implementation platforms (FPGA, ASIC, and programmable processor). One advantage of programmable and configurable platforms is that a more refined control can be conducted on them.

1.5.4 Trusted Tools

In modern and pending synthesis flows, usage of design tools is unavoidable. It is easy to embed a variety of malicious circuitry, malicious functionality, and security vulnerabilities using the CAD tools. A key security task is to derive a trusted set of synthesis and analysis tools. There are two types of trusted tools required for FPGA synthesis. The first type are tools used for realization of FPGA structures themselves. The second are tools that are used for implementation of a specified functionality on FPGA chips. In the case when programmable processors are used on FPGA ICs, we need to also secure the compiler(s) and the operating system(s).

Although at the first look the problem may seem intractable, recently it was addressed in a surprisingly simple way using notions of fully specified design (FSD). FSD is a design where user-specified functionality utilizes all the resources at all times. Therefore, a potential attacker does not have means (hardware and clock cycles) to initiate attacks. FSD can be easily realized using regular synthesis tools. The key idea is that the designer develops simple tools for checking the solutions produced by the complex synthesis tools. For example, the designer can keep updating her specified functionality until all functional units are not used in all clock cycles [109].

Additional efforts are needed to ensure that the produced designs are energy efficient and to provide runtime checking. Of course, this first approach will hopefully provide impetus for other conceptually different techniques for trusted synthesis. The final remark is that one can create numerous abstraction of trusted synthesis and that is an interesting and important problem itself.

1.5.5 Trusted IP

In addition to trusted tools, modern design flows require trusted hardware and software IP. Deriving techniques that provide proofs that a particular IP is trustworthy is essential due to the ever increasing silicon-productivity gap. There are two practical options. One is to require that each IP is fully checkable. For example, one can insist that each IP is created using trusted tools; the test vectors could be included for verification. Another maybe even more realistic, but less secure option is to develop security wrappers in form of additional circuit logic that control all inputs and outputs from each IP. In that case, the IP user can export to the otherwise untrusted IP or import from it only the data that is functionally specified. Therefore, IP would not get in possession of the privileged information from other parts of the overall design. It still can produce intentionally incorrect results, but these data would not help the attacker to take a control over the overall design. The

FPGA flexibility makes this platform better suited for inclusion of trusted IP.

1.5.6 Prevention of Reverse Engineering

One of the often promoted FPGA advantages over other implementation and architectural options is its resiliency against reverse engineering [110, 111]. Today, IC reverse engineering is widely used for tasks such as patents enforcement, technological espionage, and market trend tracing. In particular, antifuse FPGA (such as the ones designed by Actel) are identified as reverse engineering resilient devices and are widely used by several US government agencies. The key argument is that antifuse devices have very small feature sizes and it is very difficult to figure out if a particular devices fused and not. Since the number of fuses is large (several hundred thousands) their accurate classification is at least very demanding. In addition, it is often argued that usually only a small percentage of them is actually fused and that, therefore, this makes them even more difficult for reverse engineering. However, this argument is questionable since the entropy of the scenario is much higher than one where a half of fuses is actually burned.

We anticipate that reverse engineering research will pursue two lines of attacks. The first is indeed technological where the goal is to develop technologies that are difficult (ideally impossible) for reverse engineering. The second line will explore diversity in functionality specification and realization that will make each IC unique. Hence, merging imperfect technological information from multiple ICs will not bring results. A prime candidate for this purpose are N-version synthesis techniques [112, 113, 114].

1.5.7 Trojan Detection and Diagnosis

Recently hardware Trojan detection and diagnosis attracted a great deal of attention. Currently a major emphasis is on added ghost circuitry that is used in a variety of detrimental ways to alter the functionality of the initial design. In addition, more subtle attacks that employ device aging or resizing and crosstalk have been proposed and analyzed. The detection techniques can classified in two broad classes: (i) side channel-based; and (ii) ones that use functional or delay testing as their starting points. Silicon foundries are often cited as a major potential security attacker. It was argued that FPGA automatically provide protections against hardware trojans since the designer consequently configures FPGA in such a way that this information is not available to potential attackers. In addition, the regular FPGA structures makes embedding of hardware trojans difficult. However, that is only to a certain extent true because the attacker can also alter non-functional crucial components of designs such as power supply network.

It is important to note that hardware Trojans detection is much more difficult than functional or manufacturing testing because malicious alterations are intentionally conducted such that their analysis is difficult or maybe even infeasible. There are two main conceptual and technical difficulties. The first is that the ratio of the number of gates vs. input/output pins keeps increasing and as a result, the controllability and observability is consistently reduced. The second is that many discrepancies between the measurements and simulations can be easily explained as the impact of process variations.

Nevertheless, one can comfortably state that many types of structural hardware Trojans can be already detected and even diagnosed [4]. We expect that the next generation of functional Trojan horses where malicious circuitry is partly or fully merged with circuitry that is actually used for targeted functionality will create much more severe security requirements. The Trojan related topics are comprehensively addressed in Chapters 15, 16, and 17 of this book.

1.5.8 Zero Knowledge and Oblivious Transfer

There is an interesting and important class of cryptographical protocols that are unfortunately too complex for widespread use in their software implementation. This class includes zero knowledge and oblivious transfer. Hardware-based security primitives such as PPUFs when realized on FPGA have the potential to create ultra efficient realization of these protocols [13, 64]. We expect that a variety of these and similar protocols will not just be proposed but also realized and demonstrated. Interestingly, in many of these applications, protocols and security primitives the role of flexibility is essential. Therefore, FPGAs will be often the preferred implementation platform for those types of protocols.

1.5.9 Self-Trusted Synthesis

A variety of trusted modules and platforms have been developed. In some situations there is not even a specific itemization and quantification of what and who is trusted.

It has been demonstrated recently that one can easily create PUF structures in such a way that all decisions about its parameters and, therefore, the relationship between challenges and responses is completely controlled by the user. The procedure is surprisingly simple. It is sufficient to allow a user to age each PUF delay segment either randomly or to its own specifications [115, 116, 92, 117].

While, of course, the path from the devising and implementing a PUF to designing and implementing an arbitrary design may be complex and complicated, we believe that soon such solutions will be created. The FPGA flexibility is essential for such tasks although one could also create flexible ASIC solutions.

1.5.10 New FPGA Architectures and Technologies

There is a large number of different FPGA architectures in terms of combinatorial logic blocks, interconnects, and embedded memories. There are also several technologies that are used for configuration (e.g. SRAM and fuses). FPGA can be used to implement a great variety of applications. However, it appears that no consideration for security primitives and protocols has been used as the design objectives and/or constraints. After several decades of stable silicon CMOS technology, it seems that we are on the brink of revolutionary changes. For example, technologies such as graphene, III-V and graphene nanotubes, memristors, phase change materials, photonics, and plasmonics may fundamentally alter the design objectives and design process. We already see that the process variation greatly complicates detection of hardware Trojans and enables PUF existence and optimization. These technological changes will greatly impact FPGA trade-offs and architectures. In addition, 3D technologies might qualitatively alter the FPGA architectures and could have influential security ramifications.

1.5.11 FPGA Tools for Hardware-based Security

Development and rigorous analysis of FPGA security tools is a difficult and complex task. For example, process variation (PV) often plays a crucial role. PV impacts all the design metrics and has a complicated nature that keeps changing with each new technological node. For example, the effective channel length depends on several highly correlated factors. On the other hand, the threshold voltage consistently follows an uncorrelated Gaussian distribution. Other models such as device aging are also of high importance. In addition, tools for reverse engineering may have a crucial importance.

In general, one has two options: implementation and/or simulation. As FPGA implementation platform is greatly preferred due to its sharply lower cost and flexibility. There is, at least among one class of researchers, the philosophy that implementation is ultimate proof of any concept and the value of simulation is minimal. There is, obviously, some advantages in the implementation. If nothing else, it implies that the technique works at least on one platform. At the same time, any statistical proof based on one or very few points is at best of questionable value. Also, very little insight and knowledge is obtained from so limited experiments.

Simulation models are widely used in industry and have convincingly demonstrated their practical values. They can be used not only for well established technologies but also for the pending ones. Therefore, simulations are of great theoretical, conceptual, and practical values. Still, in order to obtain maximal benefit, comprehensive and up-to-date modeling and simulation tools are needed. We believe that it is crucial to have sound and fast FPGA security models that are shared among various groups of researchers. There are already activities along these lines, including Trust-Hub[1] that aims to provide both FPGA platforms that are remotely accessible as well as simulation and benchmark capabilities. For example, a collection of most effective attacks may greatly improve the development of security techniques. Finally, it is important to emphasize that these tools must find ways to transparent synthesis and analysis of FPGA-based systems.

1.5.12 Side Channels

Side channels are effective mediums and mechanisms that drastically increase the observability of the inside of the pertinent IC [118]. There is a large and ever increasing number of side channels modalities including delay, power, electromagnetic emanation, substrate noise, and temperature. Side channels greatly facilitate the detection and analysis of malicious circuitry. They also impose an even stronger threat to cryptographical protocols and hardware-based security techniques.

Side channels are in particular effective against FPGA implementations because the structure of the circuitry is known. The density is relatively lower than ASICs, and one can embed additional circuitry to augment the side channels.

1.5.13 Theoretical Foundations

Numerous interesting and sometimes surprisingly elegant hardware security techniques have been proposed. In addition, several classifications for hardware Trojan attacks and defense mechanisms, TRNGs, PUFs, and other hardware security primitives and protocols have been published. Nevertheless, we

[1] http://trust-hub.org/

are still far from establishing sound foundations and identifying the universally beneficial paradigms.

Currently, innovative but ad-hoc techniques dominate the development of hardware-based security techniques for both ASIC and FPGA platforms. More complex structure of FPGA synthesis flow and much higher number of heterogeneous design results in more pressing need for development of sound foundations and standardized ways for analysis of synthesis and analysis flows.

1.5.14 Physical and Social Security Applications

The strong emphasis of both classical algorithmic security and emerging hardware-based security techniques is on data and electronic system protection. These goals are, of course, tremendously important. However, two types of new application classes are of even higher importance. The first type consist of securing physical, chemical, and biological entities [119]. The second is related to personal and social security. Interestingly, hardware based security has the potential to play an essential role in many such application. For example, it has been demonstrated that many objects such as paper, DVD, optical fiber, and wireless radios can be used as PUFs. An important alternative is to integrate silicon (in particular FPGA) PUFs for protection of physical or biological systems. Even parts (e.g. blood vessels) of individual human bodies can be used as PUFs. FPGA-based platforms would be most often used as a staring point due to their low NRE cost.

1.5.15 Recovery and Long-life Enabling Techniques

Faults masking techniques such as built-in-self-repair (BISR) play an important role in enhancing the yield or lifetime of integrated circuits. For example, BISR mechanisms and circuitry are widely used in dynamic random access memory (DRAM) ICs. We anticipate that analogous techniques may be similarly relevant in protection against the Trojan horses. FPGA are ideally suited for BISR Trojan masking due to their reconfiguration capabilities. If there is a Trojan in FPGA hardware, one could place and configure the hardware in such a way that its impact is eliminated. If there is a Trojan in the bitstream, after its detection, characterization, and removal, one can quickly create a new Trojan-free system.

Recently, several powerful aging techniques were proposed for creation of PUFs. Aging techniques provide numerous advantages over process variation-based PUFs including enabling that user herself creates her own PUFs, much higher entropy, prevention of precomputation, and much longer lifetimes in presence of intentional and non-intentional device aging. Currently, only transistor aging is considered, but we can expect that other types of aging including electromigration will be soon pursued.

1.5.16 Executive Summary

It is difficult to consistently and in a uniform way analyze the advantages and limitations of the three principal implementation platforms (ASIC, FPGA, and programmable processors). The various security features have drastically differing requirements. Still, there is a reasonable arguments that FPGAs may emerge as a security platform of choice due to their desirable features including flexibility and post-silicon realization of functionality. While the development of new and practical hardware-based security techniques is still in very early phases, it may result in new and revolutionary ways for both system and data security. In the meantime, support for realization of classical algorithmic protocols and DRM issues will be of primary importance.

1.6 Conclusions

We have surveyed a selection of the most important important issues related to FPGA security. Specifically, we placed emphasize on security primitively (PUFs and TRNGs), analysis of potential vulnerabilities of FPGA synthesis flow, digital rights management, and FPGA-based applied algorithmic cryptography. We also analyzed the most challenging and beneficial research and development techniques related to FPGA and FPGA-based security platforms. While, of course, it is very risky to publicly state firm predictions, we expect that the system and hardware-based security of and by FPGAs is bound to emerge as a premier research and development direction.

References

1. S. Drimer, "Volatile FPGA design security – a survey (v0.96)," April 2008. [Online]. Available: http://www.cl.cam.ac.uk/~sd410/papers/fpga_security.pdf
2. D. Chen, J. Cong, and P. Pan, "FPGA design automation: A survey," Foundations and Trends in Electronics Design Automation, vol. 1, pp. 139–169, January 2006.
3. G. Qu and M. Potkonjak, Intellectual Property Protection in VLSI Design. Springer, 2003.
4. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," IEEE Design & Test of Computers, vol. 27, no. 1, pp. 10–25, 2010.

5. R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware trojans,” IEEE Computer, vol. 43, no. 10, pp. 39–46, 2010.
6. S. Trimberger, “Trusted design in FPGAs,” in Design Automation Conference (DAC), 2007, pp. 5–8.
7. Y. Hori, A. Satoh, H. Sakane, and K. Toda, “Bitstream encryption and authentication with AES-GCM in dynamically reconfigurable systems,” in Field Programmable Logic and Applications (FPL), September 2008, pp. 23–28.
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. J. Lach, W. H. Mangione-Smith, and M. Potkonjak, “Fingerprinting digital circuits on programmable hardware,” in Information Hiding (IH), 1998, pp. 16–31.
10. J. Lach, W. H. M. Smith, and M. Potkonjak, “Fingerprinting techniques for field-programmable gate array intellectual property protection,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 10, pp. 1253–1261, 2001.
11. F. Koushanfar, G. Qu, and M. Potkonjak, “Intellectual property metering,” in Information Hiding (IH), 2001, pp. 81–95.
12. F. Dabiri and M. Potkonjak, “Hardware aging-based software metering,” in Design, Automation and Test in Europe (DATE), 2009, pp. 460–465.
13. N. Beckmann and M. Potkonjak, “Hardware-based public-key cryptography with public physically unclonable functions,” in Information Hiding. Springer, 2009, pp. 206–220.
14. “Defense science board (DSB) study on high performance microchip supply. http://www.acq.osd.mil/dsb/reports/2005-02-hpms_report_final.pdf.”
15. D. A. R. P. A. D. M. T. O. (MTO), “TRUST in ICs,” 2007.
16. S. M. Trimberger and R. O. Conn, Remote field upgrading of programmable logic device configuration data via adapter connected to target memory socket, United States Patent Office, September 2007. [Online]. Available: http://patft1.uspto.gov/netacgi/nph-Parser?patentnumber=7269724
17. “Managing the risks of counterfeiting in the information technology industry. a white paper by kpmg and the alliance for gray market and counterfeit abatement (agma).”
18. Altera Corporation vs. Clear Logic Incorporated (D.C. No. CV-99-21134), United States Court of Appeals for the Ninth Circuit, April 2005. [Online]. Available: http://www.svmedialaw.com/altera%20v%20clear%20logic.pdf
19. Court issues preliminary injunction against Clear Logic in Altera litigation, Altera Corp., July 2002. [Online]. Available: http://www.altera.com/corporate/news_room/releases/releases_archive/2002/corporate/nr-clearlogic.html
20. P. Gutmann, “Secure deletion of data from magnetic and solid-state memory,” in USENIX Workshop on Smartcard Technology, July 1996, pp. 77–89.
21. ——, “Data remanence in semiconductor devices,” USENIX Security Symposium, pp. 39–54, August 2001.
22. S. P. Skorobogatov, “Low temperature data remanence in static RAM,” University of Cambridge, Computer Laboratory, Tech. Rep. 536, June 2002.
23. F. Rodriquez-Henriquez, N. Saqib, A. Diaz-Perez, and C. Koc, Cryptographic algorithms on reconfigurable hardware. Springer, 2007.
24. N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, “Leakage current: Moore’s law meets static power,” IEEE Computer, vol. 36, no. 12, pp. 68–75, 2003.
25. F.-X. Standaert, L. van Oldeneel tot Oldenzeel, D. Samyde, and J.-J. Quisquater, “Differential power analysis of FPGAs : How practical is the attack?” in Field Programmable Logic and Applications (FPL). Springer-Verlag, September 2003, pp. 701–709.

26. L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in Field Programmable Gate Arrays Symposium (FPGA), 2002, pp. 157–164.
27. S. Mangard, E. Oswald, and T. Popp, Power analysis attacks: Revealing the secrets of smart cards. Secaucus, NJ, USA: Springer-Verlag, 2007. [Online]. Available: http://www.dpabook.org/
28. F.-X. Standaert, S. B. Örs, and B. Preneel, "Power analysis of an FPGA implementation of textscRijndael: is pipelining a DPA countermeasure?" in Cryptographic Hardware and Embedded Systems Workshop, ser. LNCS, vol. 3156. Springer, August 2004, pp. 30–44.
29. F.-X. Standaert, S. B. Örs, J.-J. Quisquater, and B. Preneel, "Power analysis attacks against FPGA implementations of the DES," in Field Programmable Logic and Applications (FPL). Springer-Verlag, August 2004, pp. 84–94.
30. F.-X. Standaert, F. Mace, E. Peeters, and J.-J. Quisquater, "Updates on the security of FPGAs against power analysis attacks," in Reconfigurable Computing: Architectures and Applications, ser. LNCS, vol. 3985, 2006, pp. 335–346.
31. F.-X. Standaert, E. Peeters, G. Rouvroy, and J.-J. Quisquater, "An overview of power analysis attacks against field programmable gate arrays," Proceedings of the IEEE, vol. 94, no. 2, pp. 383–394, Febuary 2006.
32. T. S. Messerges, "Power analysis attack countermeasures and their weaknesses," in Communications, Electromagnetics, Propagation and Signal Processing Workshop, 2000.
33. S. Mangard, "Hardware countermeasures against DPA – a statistical analysis of their effectiveness," in RSA Conference, ser. LNCS, T. Okamoto, Ed., vol. 2964. Springer, February 2004, pp. 222–235.
34. P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in Cryptology Conference on Advances in Cryptology, ser. LNCS, vol. 1109. Springer-Verlag, 1996, pp. 104–113.
35. J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack," in International Conference on Smart Card Research and Applications (CARDIS), 1998, pp. 167–182.
36. J.-J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and counter-measures for smart cards," in International Conference on Research in Smart Cards (E-SMART). Springer-Verlag, 2001, pp. 200–210.
37. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in Cryptographic Hardware and Embedded Systems Workshop (CHES), ser. LNCS, vol. 2523. Springer-Verlag, August 2002, pp. 29–45.
38. K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in Cryptographic Hardware and Embedded Systems Workshop (CHES), ser. LNCS, vol. 2162. Springer-Verlag, May 2001, pp. 251–261.
39. V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier, "Electromagnetic side channels of an FPGA implementation of AES," Cryptology ePrint Archive, no. 145, 2004.
40. E. De Mulder, P. Buysschaert, S. B. Örs, P. Delmotte, B. Preneel, G. Vandenbosch, and I. Verbauwhede, "Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem," in International Conference on "Computer as a tool" (EUROCON), November 2005, pp. 1879–1882.
41. E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Power and electromagnetic analysis: improved model, consequences and comparisons," VLSI Journal of Integration, vol. 40, pp. 52–60, January 2007.
42. D. Agrawal, B. Archambeault, S. Chari, J. R. Rao, and P. Rohatgi, "Advances in side-channel cryptanalysis, electromagnetic analysis and template attacks," vol. 6, no. 1, Spring 2003.
43. D. Agrawal, J. R. Rao, and P. Rohatgi, "Multi-channel attacks," in Cryptographic Hardware and Embedded Systems Workshop, ser. LNCS, vol. 2779, September 2003, pp. 2–16.

44. R. J. Anderson and M. G. Kuhn, "Low cost attacks on tamper resistant devices," in International Workshop on Security Protocols. Springer-Verlag, 1998, pp. 125–136.
45. T. Karnik, P. Hazucha, and J. Patel, "Characterization of soft errors caused by single event upsets in CMOS processes," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 2, pp. 128–143, 2004.
46. A. Lesea, S. Drimer, J. Fabula, C. Carmichael, and P. Alfke, "The Rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs," IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 317–328, September 2005.
47. J. Fabula, J. Moore, and A. Ware, "Understanding neutron single-event phenomena in FPGAs," Military Embedded Systems, March 2007.
48. S. P. Skorobogatov, "Semi-invasive attacks – a new approach to hardware security analysis," University of Cambridge, Computer Laboratory, Tech. Rep. 630, April 2005.
49. J. M. Soden, R. E. Anderson, and C. L. Henderson, "IC failure analysis: Magic, mystery, and science," IEEE Design & Test of Computers, vol. 14, no. 3, pp. 59–69, July 1997.
50. D. Frohman-Bentchkowsky, "A fully-decoded 2048-bit electrically-programmable MOS ROM," in IEEE International Solid-State Circuits Conference (ISSCC), vol. XIV, 1971, pp. 80–81.
51. R. Cuppens, C. Hartgring, J. Verwey, H. Peek, F. Vollebragt, E. Devens, and I. Sens, "An EEPROM for microprocessors and custom logic," IEEE Journal of Solid-State Circuits, vol. 20, no. 2, pp. 603–608, 1985.
52. A. Scheibe and W. Krauss, "A two-transistor SIMOS EAROM cell," IEEE Journal of Solid-State Circuits, vol. 15, no. 3, pp. 353–357, 1980.
53. D. Guterman, I. Rimawi, T. Chiu, R. Halvorson, and D. McElroy, "An electrically alterable nonvolatile memory cell using a floating-gate structure," IEEE Transactions on Electronic Devices, vol. 26, no. 4, pp. 576–586, 1979.
54. W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze, "A user programmable reconfiguration gate array," in IEEE Custom Integrated Circuits Conference (CICC), May 1986, pp. 233–235.
55. J. Birkner, A. Chan, H. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, and R. Wong, "A very-high-speed field-programmable gate array using metal-to-metal antifuse programmable elements," Microelectronics Journal, vol. 23, no. 7, pp. 561–568, Nov. 1992. [Online]. Available: http://www.sciencedirect.com/science/article/B6V44-4829XPB-7F/2/3e9f92c100b2ab2f2527c5f039547578
56. E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. Eltoukhy, J. Chang, T. Speers, and A. Mohsen, "Dielectric based antifuse for logic and memory ICs," in International Electron Devices Meeting (IEDM), 1988, pp. 786–789.
57. "Design security in nonvolatile flash and antifuse FPGAs," Actel FPGAs, Tech. Rep.
58. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in ACM Conference on Computer and Communications Security (CCS), 2002, pp. 148–160.
59. J. Lee, D. Lim, 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 on VLSI Circuits, 2004, pp. 176 – 179.
60. S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA." Springer, 2010, p. 382387.
61. M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," ACM Transactions on Reconfigurable Technology and Systems, vol. 2, pp. 5:1–5:33, March 2009.
62. M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in IEEE Workshop on Information Forensics and Security (WIFS), 2010, p. in press.

63. M. Majzoobi, A. Elnably, and F. Koushanfar, "FPGA time-bounded unclonable authentication," in Information Hiding (IH), 2010, pp. 1–16.
64. M. Majzoobi and F. Koushanfar, "FPGA time-bounded authentication," IEEE Transactions on Information Forensics and Security, p. in press, 2011.
65. M. Majzoobi, E. Dyer, A. Elnably, and F. Koushanfar, "Rapid FPGA characterization using clock synthesis and signal sparsity," in International Test Conference (ITC), 2010.
66. G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in Design Automation Conference (DAC), 2007, p. 914.
67. C.-E. Yin and G. Qu, "LISA: Maximizing RO PUF's secret extraction," in Hardware-Oriented Security and Trust (HOST), 2010, pp. 100 –105.
68. G. Qu and C.-E. Yin, "Temperature-aware cooperative ring oscillator PUF," in Hardware-Oriented Security and Trust (HOST), 2009, pp. 36–42.
69. A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGA-friendly secure primitive," Journal of Cryptology, pp. 1–23, 2010.
70. A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in Hardware-Oriented Security and Trust (HOST), june 2010, pp. 94 –99.
71. S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in Reconfigurable Computing: Architectures, Tools and Applications, ser. Lecture Notes in Computer Science, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds. Springer Berlin / Heidelberg, 2010, vol. 5992, pp. 382–387.
72. B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Transactions on Computers, vol. 58, pp. 109–119, 2007.
73. C. W. Odonnell, G. E. Suh, and S. Devadas, "PUF-based random number generation," in In MIT CSAIL CSG Technical Memo 481 (http://csg.csail.mit.edu/pubs/memos/Memo-481/Memo-481.pdf, 2004, p. 2004.
74. D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," in Field Programmable Logic and Applications (FPL), 2006, pp. 1 –6.
75. J. von Neumann, "Various techniques used in connection with random digits," von Neumann Collected Works, vol. 5, pp. 768–770, 1963.
76. B. Barak, R. Shaltiel, and E. Tromer, "True random number generators secure in a changing environment," in Cryptographic Hardware and Embedded Systems workshop (CHES). Springer-Verlag, 2003, pp. 166–180.
77. B. Jun and P. Kocher, "The Intel random number generator," in CRYPTOGRAPHY RESEARCH, INC., 1999.
78. G. Marsaglia, "DIEHARD: A battery of tests for randomness," in http://stat.fsu.edu/ geo, 1996.
79. NIST, "A statistical test suite for random and pseudorandom numbers," in Special Publication, 2000.
80. A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.
81. O. Goldreich, Foundations of Cryptography, Volume 1: Basic Tools. Cambridge University Press, 2001.
82. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley, 1996.
83. W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. IT-22, pp. 644–654, 1976.
84. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21 (2), pp. 120–126, 1978.

85. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, "Towards electrical, integrated implementations of SIMPL systems," in Workshop in Information Security Theory and Practice (WISTP), 2010, pp. 277–292.
86. U. Rührmair, "SIMPL systems, or: Can we design cryptographic hardware without secret key information?" in SOFSEM, 2011, pp. 26–45.
87. Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, and U. Rührmair, "Circuit-based approaches to SIMPL systems," Journal of Circuits, Systems, and Computers, vol. 20, pp. 107–123, 2011.
88. Y. Alkabani and F. Koushanfar, "Consistency-based characterization for IC trojan detection," in International Conference on Computer-Aided Design (ICCAD), 2009, pp. 123–127.
89. F. Koushanfar, A. Mirhoseini, and Y. Alkabani, "A unified submodular framework for multimodal IC trojan detection," in Information Hiding (IH), 2010.
90. F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," IEEE Trans. on Information Forensic and Security, 2011.
91. S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: foundations and hardware security applications," in ACM/IEEE Design Automation Conference (DAC), 2010, pp. 222–227.
92. S. Wei and M. Potkonjak, "Scalable segmentation-based malicious circuitry detection and diagnosis," in International Conference on Computer Aided Design (ICCAD), 2010, pp. 483–486.
93. ——, "Integrated circuit security techniques using variable supply voltage," in ACM/IEEE Design Automation Conference (DAC), to appear, 2011.
94. A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking techniques for intellectual property protection," in ACM/IEEE Design Automation Conference (DAC), 1998, pp. 776–781.
95. A. B. Kahng, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Robust IP watermarking methodologies for physical design," in ACM/IEEE Design Automation Conference (DAC), 1998, pp. 782–787.
96. I. Hong and M. Potkonjak, "Technique for intellectual property protection of DSP designs," in International Conference on Acoustic, Speech, and Signal Processing (ICASSP), 1998, pp. 3133–3136.
97. F. Koushanfar, I. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 10 (3), pp. 523–545, 2005.
98. J. Lach, W. Mangione-Smith, and M. Potkonjak, "Enhanced FPGA reliability through efficient runtime fault recovery," IEEE Transactions on Reliability, vol. 49 (49), pp. 296–304, 2000.
99. A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20 (10), pp. 1236–1252, 2001.
100. D. Kirovski and M. Potkonjak, "Local watermarks: Methodology and application to behavioral synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22 (9), pp. 1277–1284, 2003.
101. F. Koushanfar and Y. Alkabani, "Provably secure obfuscation of diverse watermarks for sequential circuits," in International Symposium on Hardware-Oriented Security and Trust (HOST), 2010, pp. 42–47.
102. D. Ziener, S. Assmus, and J. Teich, "Identifying FPGA IP-cores based on lookup table content analysis," in International Conference on Field Programmable Logic and Applications (FPL), 2006, pp. 1–6.

103. Y. Alkabani, F. Koushanfar, and M. Potkonjak, “Remote activation of ICs for piracy prevention and digital right management,” in International Conference on Computer Aided Design (ICCAD), 2007, pp. 674–677.
104. Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, “Trusted integrated circuits: A nondestructive hidden characteristics extraction approach,” in Information Hiding (IH), 2008, pp. 102–117.
105. F. Koushanfar, G. Qu, and M. Potkonjak, “Intellectual property metering,” in International Workshop on Information Hiding (IHW). Springer, 2001, pp. 81–95.
106. F. Koushanfar and G. Qu, “Hardware metering,” in Design Automation Conference (DAC), 2001, pp. 490–493.
107. Y. Alkabani and F. Koushanfar, “Active hardware metering for intellectual property protection and security,” in USENIX Security Symposium, 2007, pp. 291–306.
108. F. Koushanfar, “Active integrated circuits metering techniques for piracy avoidance and digital rights management,” ECE Dept., Rice University, Tech. Rep. TREE1101, 2011.
109. M. Potkonjak, “Synthesis of trustable ICs using untrusted CAD tools,” in ACM/IEEE Design Automation Conference (DAC), 2010, pp. 633–634.
110. J. Wong, D. Kirovski, and M. Potkonjak, “Computational forensic techniques for intellectual property protection,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23 (6), pp. 987–994, 2004.
111. D. Kirovski, D. Liu, J. Wong, and M. Potkonjak, “Forensic engineering techniques for VLSI CAD tools,” in IEEE/ACM Design Automation Conference (DAC), 2000, pp. 580–586.
112. Y. Alkabani and F. Koushanfar, “N-variant IC design: methodology and applications,” in Design Automation Conference (DAC), 2008, pp. 546–551.
113. Y. Alkabani, F. Koushanfar, and M. Potkonjak, “N-version temperature-aware scheduling and binding,” in International Symposium on Low Power Electronics and Design (ISLPED), 2009, pp. 331–334.
114. M. Majzoobi and F. Koushanfar, “Post-silicon resource binding customization for low power,” ACM Transactions on Design Automation of Electronic Systems (TODAES), to appear, 2011.
115. M. Nelson, A. Nahapetian, F. Koushanfar, and M. Potkonjak, “SVD-based ghost circuitry detection,” in Information Hiding (IH), 2009, pp. 221–234.
116. M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, “Hardware trojan horse detection using gate-level characterization,” in ACM/IEEE Design Automation Conference (DAC), 2009, pp. 688–693.
117. M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, “Differential public physically unclonable functions: Architecture and applications,” in ACM/IEEE Design Automation Conference (DAC), to appear, 2011.
118. A. Vahdatpour, M. Potkonjak, and S. Meguerdichian, “A gate level sensor network for integrated circuits temperature monitoring,” in IEEE Sensors, 2010, pp. 1–4.
119. M. Potkonjak, S. Meguerdichian, and J. Wong, “Trusted sensors and remote sensing,” in IEEE Sensors, 2010, pp. 1–4.