A Survey of Recent Results in FPGA Security and Intellectual Property Protection

François Durvaux[1⋆], Stéphanie Kerckhof[1⋆⋆],
Francesco Regazzoni[1,2], François-Xavier Standaert[1⋆⋆⋆†]

[1] UCL Crypto Group, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.
[2] ALaRI Institute, University of Lugano, Switzerland.

e-mails: stephanie.kerckhof@uclouvain.be; francois.durvaux@uclouvain.be;
fstandae@uclouvain.be; regazzoni@alari.ch

Abstract. Field Programmable Gate Arrays (FPGAs) are reconfigurable devices which have emerged as an interesting trade-off between the efficiency of Application Specific Integrated Circuits (ASICs) and the versatility of standard microprocessors [81]. Progresses over the last 10 years have improved their capabilities to the point where they can hold a complete System on a Chip (SoC) and thus become an attractive platform for an increasing number of applications (e.g. signal processing, image processing, aerospace, ...). In view of the important data manipulated by these devices, but also of the high amount of Intellectual Property (IP) they may contain, security-related questions have arisen. First, can we use FPGAs as security devices for e.g. securely and efficiently encrypting sensitive data (in particular when compared to software solutions)? Second, how can we guarantee that the IP corresponding to FPGA designs is protected (i.e. cannot be easily counterfeited)? Such questions have been the target of a large number of papers in literature, including several surveys, e.g. [13, 71, 83]. In this chapter, we take another look at them and review a number of important recent results related to security IPs and IP security in modern reconfigurable devices. The chapter is structured in three main sections. First, we briefly describe the structure of recent FPGAs. Next, we discuss security IPs in FPGAs, taking the example of symmetric encryption with the AES Rijndael, and including their performance evaluations and resistance against physical attacks. Finally, we emphasize recent trends for improving IP security in FPGAs, including bitstream security, the use of code watermarking techniques and the exploitation of Physically Unclonable Functions (PUFs).

⋆ PhD student funded by the Walloon region MIPSs project.
⋆⋆ PhD student funded by a FRIA grant, Belgium.
⋆⋆⋆ Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).
† Work funded in part by the ERC project 280141 (acronym CRASH).

1 FPGAs: an overview

In this section, we introduce the major features of FPGAs for the non-familiar reader. First, we present the overall structure of the devices. Next, we briefly describe the different steps of an FPGA design flow. Finally, we discuss the different technologies of reconfigurable devices that are publicly available.

1.1 Structure

For convenience, we will focus on the main two FPGA manufacturers: Altera [3] and Xilinx [84][1]. The Configurable Logic Block and the Logic Array Block (CLB and LAB) are the basic logic cells for the most recent Xilinx (e.g. Virtex-7) and Altera (e.g. Stratix-V) FPGAs. Such devices typically contain an array of these cells connected together through a configurable routing matrix. The CLB is composed of multiple slices (Fig. 1) and the LAB is composed of multiple Adaptive Logic Modules (ALMs, Fig. 2). The main components of slices and ALMs are the Look-Up Tables (LUTs) and the registers. In the latest FPGAs (Xilinx Virtex-7 or the Altera Stratix-V), the LUTs are 6-bit input, 2-bit output functions generators. They can also be configured as small embedded memories or shift registers. In older technologies, the LUT input was generally limited to 4 bits. Slices and ALMs combine the LUTs with a chaining logic in order to allow efficient arithmetic operations. Next, this combinatorial part of the logic cells is followed by registers used to generate synchronous logic. Efficient FPGA designs essentially try to take advantage of these resources in the best manner in order to perform some algorithmic task. Hence, slices, LABs, ALMs, LUTs and registers are the typical figures of merit used to evaluate the performances of FPGA implementations (see the next section).

In addition, it is worth mentioning that the routing matrix of the FPGAs also has a major impact on final performances. These routes, allowing the connection between the different computational blocks and memories, are generally structured according to the connection lengths. Quite naturally, small wires have much lower effective capacitance than long ones, hence explaining significant variations of the computation delays, when efficient routing cannot be ensured with "local" connections [69]. Finally, most modern FPGAs combine reconfigurable logic elements with a variety of "embedded blocks", i.e. specific elements that are hardwired in the devices. Typical embedded blocks include memories, multipliers and processors.

1.2 Design flow

Configuring FPGAs for different applications is carried out in several steps, as when designing ASICs. The first step is to define the behavior of the circuit with

[1] These two manufacturers produce mainly "volatile" FPGAs in which the configuration is stored in non-volatile memory devices like EEPROM or Flash. Non-volatile FPGAs also exist but are out of the scope of this paper.

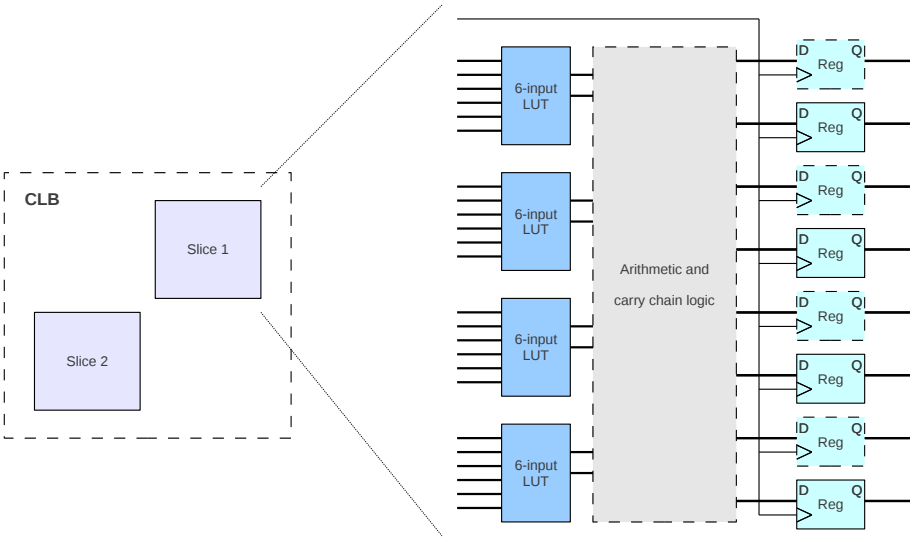

Fig. 1. Xilinx Configurable Logic Block and detailed Slice (dashed registers are optional, they can be bypassed).

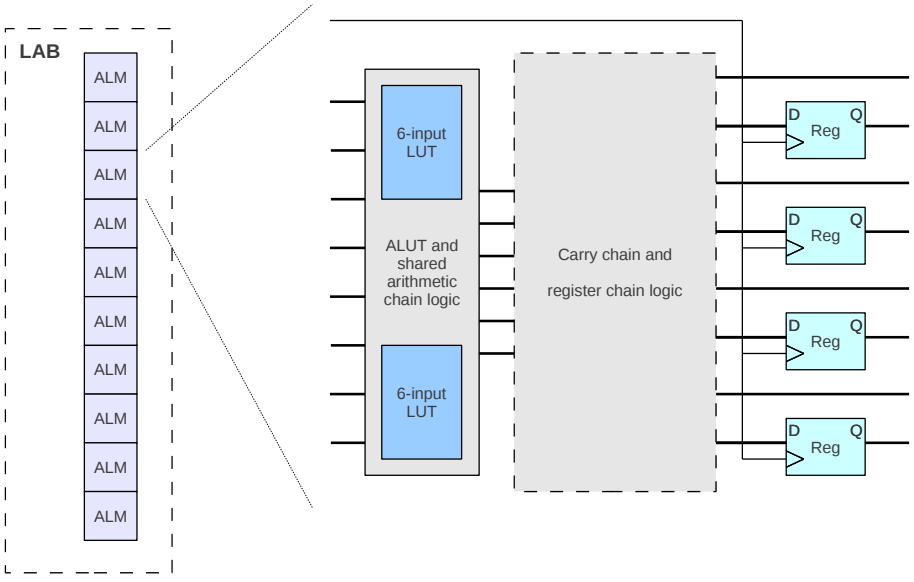

Fig. 2. Altera Logic Array Block and detailed Adaptive Logic Module.

a Hardware Description Language (HDL). The two most usual HDLs are VHDL and Verilog. They can be used to describe a design from a functional point of view. This part of the design flow is essential as it is where most of the algorithmic optimizations can be introduced. Importantly, one can choose to design for performance or for portability. In the first case, the goal is to take advantage of all the specificities of the target platform, in order to increase performances (e.g. particular logic element configurations, embedded blocks, ...). In the second case, the goal is to have a code that is directly usable on the widest range of devices. The next step consists in generating a technology-mapped netlist. It is done thanks to an electronic design automation tool. The netlist is a description of all the nets linking the basic cells of the target device specified by the user (logic gates, registers, memory, ...). Hence, contrary to the HDL description, the netlist is device-specific. Once the netlist is generated, the different cells are placed-and-routed on the FPGA map, which again takes advantage of an automated design tool. At this step of the design, users perform additional simulations and tests (e.g. timing analysis) in order to validate that the obtained results are functionally correct and fits the target performances. Finally, a binary file describing the design is generated. This file is loaded into the FPGA through, e.g. a serial port like JTAG (Joint Test Action Group, IEEE standard). It is used to configure the FPGA which then behaves essentially like an ASIC.

Note that this design flow illustrates where the performance loss of FPGAs vs. ASICs comes from. Namely, a part of the FPGA resources are consumed to store its configuration (i.e. the description of its functional behavior), whereas all the (hardwired) resources of an ASIC are dedicated to the design processing tasks. However, compared to a software solution, FPGA designs usually allow major performance increases, as they can take advantage of parallel computing and processing units that are specialized for one specific type of computation. Finally, it is also worth mentioning that designing with recent FPGAs offers more and more facilities in terms of pre-designed blocks. That is, a large number of primitives (e.g. embedded memories, or embedded processors like the Microblaze for Xilinx and Nios for Altera) are now made available by manufacturers, which can essentially be used in a HDL design as black boxes.

1.3 Technologies

Different categories of FPGAs are available from different vendors. In the first place, high-end (more expensive) FPGAs are generally distinguished from lower-end ones. The first category features all the latest developments of the manufacturers (e.g. the Xilinx Virtex and Altera Stratix devices) while the second one is essentially optimized for cost (e.g. the Xilinx Spartan or Artix and Altera Cyclone devices). These FPGAs also differ in their fabrication technology ranging from 130 to 28 nanometer in 2011. As an illustration, we next describe a few examples of recent reconfigurable devices.

The most recent high-end FPGA from Xilinx is the Virtex-7. It is part of a new generation of devices built from a 28-nanometer technology and designed for

maximum power efficiency. Virtex-7 FPGAs contain up to 1,954,560 logic cells, which corresponds to 305,400 slices. Each slice contains four LUTs and eight flip-flops, some of them being usable as distributed RAM, for up to 21.55 Mb. The largest device in the family also contains 2,160 DSP slices, which include a pre-adder, an adder, an accumulator and a 25×18 multiplier. It additionally contains 46.51 Mb of RAM blocks that can be instantiated as 18 or 36 Kb blocks. Finally, these FPGAs contain 24 clock management tiles, 4 interface blocks for PCI express, 36 low-power 12.5-Gbps transceivers, 1 analog-to-digital converter, 24 I/O banks, and 1,200 user I/O.

The latest family of high-end FPGAs developed by Altera is denoted as the Stratix-V. They are based on a similar 28-nanometer high-performance process optimized for low power and contain up to 358,000 ALMs. Each ALM is based on two combinational adaptive LUTs (ALUTs) and four registers. Some ALMs can be configured as distributed SRAM, for up to 12.12 Mb. This largest version of the Stratix-V includes 352 27×27 DSP blocks, 704 18×18 multipliers, 52 Mb of RAM blocks, which can be instantiated as 20 Kb blocks, 4 PCI express hard IP blocks, and 48 14.1-Gbps transceivers.

2 Security IPs

Modern reconfigurable devices are complex and complete platforms which provide an appealing and cost effective solution to implement high performance low-to-medium volume custom integrated circuits. FPGA designs are characterized by reduced non-recurring engineering costs and reduced time to market. The cost for a typical mask set to fabricate an ASIC using modern CMOS technology runs in the range of $500K to $700K. By contrast, a system designer can purchase an off-the-shelf FPGA and program it for only a fraction of the cost. Quite naturally, the resulting circuit will be slower, consume more power and utilize significantly more silicon resources than its ASIC equivalent. Still, FPGAs are an attractive platform for several applications nowadays.

In general, and as discussed in the previous section, designing with FPGAs shares a number of similarities with ASIC development. In the first place, the clear definition of the architectural choices and performance goals is a prerequisite in both cases. The requirements of the target application also determine the main figures of merit which will be optimized by the designer. But once these decisions are specified, the task of efficiently designing for FPGAs is different from its ASIC counterpart. In the latter, the designer has full control over the components to implement. By contrast, in the FPGA case, he is forced to use the components that the FPGA vendor selected as the most suitable for a majority of applications. As a result, the strategy to maximize the exploitation of the available resources may significantly depend on the selected FPGAs.

In the remainder of this section, we illustrate this discussion in the context of security IPs. In particular, and as a case study, we first review different implementations of the Advanced Encryption Standard. Next, we take advantage of these examples in order to underline the problem of fairness in the comparison

among different architectures, and the meaningfulness of the metrics currently used for this purpose. Finally, we discuss the specificities of security IPs in FPGAs regarding so-called physical attacks in which an adversary either observes physical emanations of the target devices (side-channel attacks), or tries to induce faults during the cryptographic computations.

2.1 The AES case

The Rijndael algorithm was adopted as the Advanced Encryption Standard (AES) in 2001 [54]. The standard supports a block size of 128-bit and key sizes of 128, 192 and 256 bits. The encryption process, which is illustrated in Fig. 3, starts with the first key addition, followed by a number of round functions which depends on the key size. The round function is composed of four transformations applied to a state of 16 bytes. *ShiftRows* cyclically shifts to the left the bytes in the last three rows of the state, using different offsets; *SubBytes* is a non-linear byte substitution and operates independently on each byte of the state; *MixColumns* multiplies modulo $x^4 + 1$ the columns of the state by the polynomial $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$; finally, *AddRoundKey* adds a round key to the state. All the round keys are generated by a *key schedule* routine, which takes the secret key and expands it as specified in the standard. The decryption algorithm is similar to the encryption one and uses the inverted versions of the basic transformations used during the encryption. The key schedule for decryption is identical to the one used for encryption, but it starts using the last round key and generates the round keys in reverse order. In this context, the typical design decisions that have to be taken include:

- Which key size should be supported (128, 192, or 256)?
- Does the implementation have to compute encryption only, decryption only, or both of them?
- How is the key scheduling computed ("on-the-fly", precomputed on chip before each encryption, or precomputed off chip)?
- Is the algorithm supposed to run in a specific encryption mode (with feedback, without feedback, ...), which would prevent parallelization?

Additionally, depending on the application requirements, the designer also has to select a number of architectural parameters, including the datapath size, the type of architecture (loop or unrolled), the target throughput (high performances or not), the portability of the design among different platforms, the usable area (low cost or not), and, more specific to the reconfigurable world, the type of resources which can be used (BRAMs, DSPs, only LUTs, ...), and the target FPGA.

In the remainder of the section, we describe a representative subset of designs targeting several of these goals. Reported architectures range from high speed to low cost, also including designs which maximize the exploitation of the inner structure of the target FPGA, summarized in Table 1.

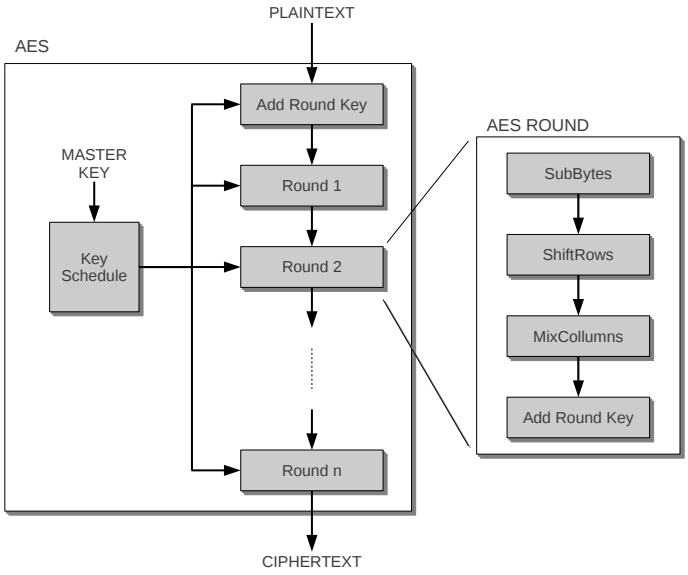

Fig. 3. AES encryption block diagram.

Area efficient designs of AES were implemented using reduced datapath. The 8-bit architecture proposed by Good et al. [21] features a datapath consisting of two processing units, one to perform the SubBytes transformation and the second to compute the multiply and accumulate operations needed by MixColumn. The processor has an instruction set composed of 15 instructions and relies on a pipeline to execute a new instruction every cycle. Low cost implementations were also targeting 32-bit datapaths: possible examples are those of Chodowiec et al. [10] and Rouvroy et al. [63].

Muliple AES implementations were proposed by Helion Technology [78], targeting different purposes: standard encryption, fast encryption, and fast encryption and decryption. The user also has the possibility to choose whether the expansion of the key is performed on-the-fly or precomputed off chip. The fast encryptor (and decryptor) are based on a high-throughput 128-bit datapath version, and are evaluated both on Altera Cyclone-III and Stratix-IV FPGAs, i.e. a low-cost and a high-end FPGA. Several implementations of AES were also proposed by Standaert et al. [75]: the efficiency of their architectures was evaluated at different stages of the design process and the structure of the pipeline which better considered the place and route constraints was discussed.

David Kenney [35] proposed an energy efficient 128-bit implementation in his PhD thesis. It is based on a complete unroll of the rounds and deeply exploits the

pipeline. Examples of this kind of implementation are also reported by Järvinen et al. [30], Hodjat et al. [29] and Chaves et al. [9]

Designs which maximize the specific resources of the target FPGA have also been proposed. Bulens et al. [7] proposed a design which deeper exploits the 8-bit look-up table structure of the Xilinx Virtex-E. Drimer et al. [15] proposed an AES design which is largely implemented on the additional components of the the FPGA, such as DSPs and BRAM, attempting to leave the majority of the programmable logic available for other applications.

To conclude, let us mention that very similar considerations and choices apply to the implementation of other cryptographic algorithms. Typical examples include the implementation of Elliptic Curve Cryptography and hash functions. We refer the interested reader to [11, 17, 26, 28, 27, 87].

2.2 Performance evaluation

Evaluating the performances of a design is a very natural goal. As introduced in the previous sections, each of the resources of an FPGA (slices, LABs, ALMs, LUTs, registers, ...) can be used as figure of merit to carry on the comparison. Unfortunately, producing fair comparisons for FPGA implementations is limited by a number of difficulties that we discuss in this section.

In the first place, it is worth recalling that, as partially illustrated by Table 1, there may be as many different implementations as there are design goals. To make it simple, comparing an 8-bit datapath implementation designed for minimizing the area occupation with a completely unrolled one which aims at high throughput makes little sense. In this respect, even using so-called "efficiency metrics", such as the throughput over area ratio, can be misleading: a compact implementation is indeed inherently less efficient, because of a more involved control part. Besides, and as pointed out by Saar Drimer [14], the basic logic cell of an FPGA varies significantly between vendors and platforms. In addition, the area and performance results are not only affected by the target device and the design itself, but also by the tools used for synthesis and place-and-route and their respective options, the possible presence of a more complete application where the IP core (e.g. cryptographic algorithm) is embedded (which limits the space available for the security IP, thus forces the tool to pack the design), and many other factors.

Eventually, and as previously mentioned, the very goal of optimizing an implementation is highly dependent on the need for portability. Namely, a designer always has the possibility to deeply exploit the inner structure of his target FPGA, in order to achieve higher speed or smaller occupation. However, this comes at the price of a reduced portability, since the inner structure is specific to the device, model, and vendor. Hence, considering such a decision is important when performing comparative analyzes. In this respect, it is worth underlining that academic publications generally tend to focus on device-specific optimizations more than found in industrial IP cores, where portability is usually appreciated for the cost-reduction it allows.

Table 1. FPGA Implementations of the AES algorithm

Device	**Datapath**	**Logical Element**	**Memory blocks**	**Freq. (MHz)**	**Thr. (Gbps)**	**enc/dec**	**Architecture Type**	**Key Scheduling**
Xilinx Spartan-II [21]	8	124 (slices)	2	-	0.0022	enc/dec	loop + pipeline	precomputed on chip
Xilinx Spartan-III [63]	32	163 (slices)	3	71.5	0.208	enc/dec	loop	precomputed on chip
Altera Cyclone-III [78]	32	314 (LEs)	3	170	0.45	enc	loop	precomputed off chip
Altera Cyclone-III [78]	32	603 (LEs)	3	170	0.45	enc	loop	on-the-fly
Xilinx Spartan-II [10]	32	222 (slices)	3	60	0.166	enc/dec	loop	precomputed on chip
Altera Cyclone-III [78]	128	906 (LEs)	10	174	2.02	enc	loop	on-the-fly
Altera Stratix-IV [78]	128	651 (ALUTs)	10	300	3.49	enc	loop	on-the-fly
Altera Stratix-IV [78]	128	1652 (ALUTs)	18	285	3.32	enc/dec	loop	on-the-fly
Altera Cyclone-II [35]	128	3039 (LEs)	18	198.9	2.5	enc/dec	loop + pipeline	on-the-fly
Xilinx Virtex-E [75]	128	1767 (slices)	0	167	2.085	enc	loop + pipeline	on-the-fly
Xilinx Virtex-5 [7]	128	400 (slices)	0	350	4.1	enc	loop + pipeline	on-the-fly
Xilinx Virtex-II Pro [29]	128	5.177 (slices)	84	168.3	21.54	enc	unrolled + pipeline	precomputed off chip
Xilinx Virtex-II 2000 [30]	128	10.750 (slices)	0	139.1	17.8	enc	unrolled + pipeline	on-the-fly
Xilinx Virtex-II Pro [9]	128	3.513 (slices)	80	271	34	enc/dec	unrolled + pipeline	precomputed off chip
Xilinx Virtex-5 [15]	128	321 (slices)	80	413	52.8	enc/dec	unrolled + pipeline	precomputed off chip

Of course, the different limitations discussed here do not mean that it is impossible to compare different designs. They simply underline that any comparison should be carried out with care and the results of the comparison should be well understood. In this context, the typical comparison metrics include working frequency (measured in GHz or MHz), throughput (measured in Mbit/second), hardware occupation (measured in LUTs, registers, ...), and the previously mentioned throughput over area ratio. In addition, improved comparisons can take the reproducibility of the synthesis results into account. For this reason Drimer encouraged the academic and scientific community to favor the publication of implementations which are presented together with the source code [14].

This issue of fair comparisons for FPGA designs has recently received attention, as FPGA implementations are one of the criteria for the selection of the next hash standard [53]. In this context, Gaj et al. proposed a series of guidelines including the definition of suitable performance metrics and the development of uniform interfaces [18], also used in [36]. Furthermore, their evaluation of the candidates was completed on several representative FPGA platforms, from the two major vendors discussed in the first section of this chapter.

2.3 Side-channel attacks & countermeasures

The previous section discussed different implementations of the AES algorithm. However, when the target application involves security IPs, it is important to consider also the security of the designs against various types of physical attacks. Physical attacks exploit the characteristics of the hardware platform on which the algorithm is implemented in order to acquire sensitive information. They are usually classified among two axes: invasive or non-invasive and active or passive. Side-channel attacks are a particular class of passive and non-invasive physical attacks which use the information leaked while data is being processed in order to break some security guarantee, e.g. by deriving the secret key of a cryptographic algorithm [46]. Common examples of this leaking information are the time employed for the computation [38], the corresponding power consumed [39], or the electromagnetic emissions [2, 58].

Among the different types of power-based attacks available in literature, the most common ones are Simple Power Analysis (SPA) and Differential Power Analysis (DPA). In an SPA, an attacker measures the power consumed by a device while performing cryptographic operations and, by observing the traces, attempts to deduce the secret information, e.g. by distinguishing different operations. SPA attacks are typically powerful in context where distinguishing operations directly allows recovering secret information, e.g. in public key cryptographic computations. DPA attacks extend this principle towards data-dependencies and try to recover secret information with some statistical processing. A typical DPA attack consists of four steps. At first, an intermediate key dependent result is selected as the target. Then, the attacker encrypts (decrypts) a certain number of known plaintexts (ciphertexts) and measures the corresponding power consumption traces. Subsequently, hypothetical intermediate values are calculated, based on a key guess. Finally, the hypothetical intermediate values

(and thus the corresponding secret keys) are verified against the measured power traces. If the attack is successful, the right key hypothesis will be clearly visible in correspondence of the time frame where the information is leaked [46].

First investigations of power analysis attacks on FPGA devices were carried out in [56, 76] in 2003. These initial results have been followed by several papers, e.g. [72–74] which improved the efficiency of the attacks and analyzed them in deeper details, considering also the specificity of reconfigurable devices. Examples of tackled problems are the dependency between the attack and used resources or the effects that a pipelined architecture might have on side-channels.

Counteracting side-channel attacks is difficult since the attack is caused by an intrinsic feature of the transistors. Hence, physical security is generally guaranteed by a combination of countermeasures, acting at different abstraction levels (e.g. hardware, algorithm, protocol). Among these solutions, so-called hiding and masking techniques have attracted significant attention for FPGAs. Hiding aims at obtaining independence between the power consumed by a cryptographic device and the secret data being processed, e.g. by ensuring constant power consumption for all inputs. Masking is a technique inspired by cryptographic secret-sharing schemes: the original message to be encrypted is divided into parts called shares that are then encrypted separately. This aims to ensure that only attacks combining the leakage of different shares can be successful. The ciphertext is eventually reconstructed by combining the output shares.

As an illustration, to implement logic function resistant against power analysis attacks, Tiri and Verbauwhede proposed WDDL [79], a differential logic style with precharge, which can be designed from FPGA elements. WDDL essentially translates every gate on a design into "protected gates" having the goal to produce constant power consumption. For this purpose, each gate uses four inputs that are pairwise complementary, and always computes the two complementary outputs. On the Virtex-II FPGAs, WDDL requires 2 LUTs to generate a gate with 2 differential outputs.

Masked implementations of different algorithms were also proposed for FPGAs, incurring different performances and area overheads. Concerning the AES algorithm, an implementation which combines Boolean and multiplicative masking was proposed by Mentens et al. [49]. The area overhead of their secured core compared with the reference unsecured version is approximately 20%, while the speed is degraded by 30%. Kamoun et al. [32] implemented a masked AES S-box on Virtex-4 FPGA which incurs an area overhead of 44% and a frequency decrease of 31%. Specific features of state of the art FPGAs were also exploited for implementing masking: the larger input size of the basic block of Xilinx Virtex-5 FPGA was combined with optimization techniques for S-boxes to obtain an efficient FPGA implementation of the AES algorithm, masked against side-channel attacks [62].

Note that none of the countermeasures investigated so far can guarantee perfect security. In particular, a certain number of physical effects, like glitches in integrated circuits for masking, or early propagation effects for hiding, can

compromise the security of these countermeasures [47, 48, 77]. Overall, providing security against physical attacks remains an active scope of research. For example, one recent trend is to investigate FPGA-dedicated countermeasures [25], taking advantage of the specificities of reconfigurable devices.

Still concerning the power analysis attacks, it is worth mentioning the effort done by the Research Center for Information Security (RCIS) of AIST and Tohoku University in the direction of developing a common platform for standardizing the evaluation of attacks and the comparison of countermeasures. The outcome was the Side-channel Attack Standard Evaluation BOard (SASEBO) [65], which was distributed together with design information to research institutes as common experimental platforms. To date, there are five types of SASEBO boards, based on both Xilinx and ALTERA FPGAs. The FPGA boards have microprocessor features, and thus side-channel attack experiments against cryptographic software can also be performed. Additionally, SASEBO-R is extended with a custom cryptographic LSI that supports all of the block ciphers adopted by ISO/IEC 18033-3, as well as the public-key cipher RSA. The advantage of SASEBO board, besides eliminating the high engineering costs, is to provide a common platform which allows results to be reproduced.

2.4 Fault attacks & countermeasures

More recently, fault attacks have also been applied to FPGAs. Since it is a relatively new area of research, they are not as common as power analysis attacks and there are only few works discussing them, all focusing on attacks against the AES algorithm. In a nutshell, a fault attack consists of a deliberate injection of a fault into a target device, in our case the FPGA where the cryptographic routine is executed. The fault is injected by actively tampering with the device itself. Once the error required by the attack model is produced, the adversary can analyze the differences between the correct and the faulty outputs of the device and extract sensitive information, e.g. about a secret key. Depending on the specific attack used, it can be required that the fault has to appear at a specific point of the computation [16]. Several methods for inducing a fault were proposed and verified to be effective in FPGA. Some of them, such as clocking the circuit at a different speed, or reducing the supply voltage, can be exploited using very inexpensive devices.

Examples of successful attacks mounted on FPGAs were presented by Saha et al. [64], Khelil et al. [37], and Selmane et al. [68]. In Saha et al. [64], the attack induces faults into a diagonal of the AES state matrix that is input to the $8th$ round. Since, to successfully mount a fault attack, it is necessary to introduce exactly the error required by the fault model, the adversary has to perform a preliminary exploration in order to find the sweet spot for the attack. A sweet spot is the area where the degradation induced in the device is sufficient to generate only the needed fault without resulting in a complete corruption of the circuit behavior. In this case, the fault is injected by switching the clock to a faster frequency when the $8th$ round of the encryption function begins.

Other works such as [37, 68] also evaluate the feasibility on FPGAs of the attack proposed by Piret and Quisquater [57]. The attack requires the injection of a fault on one byte of the state before the computation of MixColumn in the $9th$ round of the AES algorithm. In both works, the fault is injected by reducing the supply voltage to induce a setup time violation. This is possible since the propagation delay is inversely proportional to the power supply: when the voltage is reduced, the signals which are propagated into the circuit require more time to stabilize. As a consequence, the values stored in the register might not be computed in time and they produce a faulty result.

Considering the relatively recent appearance of fault attacks in FPGAs, no countermeasures have been developed so far, specifically targeting reconfigurable devices. However, initial work has been performed to evaluate the resistance of power analysis countermeasures against fault attacks. In particular, Selemane et al. [67] used the lowering voltage technique to evaluate the intrinsic resistance of WDDL against fault attacks on FPGA implementations. Furthermore, techniques for fault tolerance were already successfully adapted to the needs of fault attack prevention in ASICs [6, 33]. Hence, it is possible to envision the use of similar schemes in reconfigurable devices. As a final note, it is important that the designer also considers the interaction between countermeasures against one attack and the vulnerability to another attack, since it has been shown that several error correcting and detection codes increase the vulnerability of a cryptographic circuit to power analysis attacks [60, 61].

3 IP security

Since FPGAs are volatile and generic platforms, a large number of designs can be implemented on them, ranging from essentially hardware to on chip combinations of hardware and software. Most of the time, these implementations are developed by different designers and the inherent value of their IP can be high. As a result, various design houses base their business on the selling of IPs, and their protection against various types of counterfeiting has become an important issue. In the first place comes the problem of bitstream security. That is, given an FPGA board running an application, how to guarantee that no adversary can recover the bitstream and, from it, clone or reverse-engineer a design? Next, and more critical, comes the problem of design security. That is, given an IP that is sold by a design house, how to guarantee that this IP is not used beyond the terms of a license? And how to combine this requirement with the flexibility constraints of the client (e.g. the need to integrate an IP in a large design, and to simulate it)? In this section, we briefly tackle these two problems and describe some recent trends that are considered to solve them.

3.1 Bitstream security

In this case, the most frequent solution is bitstream encryption. That is, the bitstream is encrypted by the CAD tool with user-defined symmetric (secret)

keys. The same keys are stored on the FPGA, e.g. in a volatile memory with an external battery. During configuration, an on-chip decryption circuit is used to recover the original configuration file. Readback is not allowed when encrypted bitstreams are used. The main drawback of bitstream encryption is the need for an external battery to maintain the keys, and the difficult key management. For example, if a single key is used for all the boards, then a system designer has no opportunity to update the configuration files for only a part of them. Ideally, it should be possible to update the symmetric keys remotely. This could be achieved either by the use of a symmetric master key (but the system security would then depend on this single key) or a public-key mechanism in which each FPGA would come with a private/public key pair stored in a non-volatile memory. Note also that the importance of authenticated encryption for FPGA bitstreams has been discussed in [12, 80]. Finally, let us mention that the on-chip circuitry that is used to perform decryption also has to be secure against physical attacks. As recently discussed in [50], security against side-channel attacks was not considered in certain Xilinx designs.

3.2 Design security

3.2.1. Encrypted netlists. A first solution to protect IP providers' work is exchanging encrypted netlists and simulation models with the system designer. This encryption is done through the Xilinx development tools which embed the key in its software. So, the system designer can easily instantiate the IP core as a black box, i.e. without having access to the implementation details. The problem is that this solution only protects the IP integrity and does not prevent from re-using the IP many times. Moreover, the IP is only protected by a key stored in the CAD software. Hence, it may be potentially recovered by reverse-engineering this tool.

3.2.2. Security chips. Another solution is proposed in the Xilinx documentation [43]. It works as follows: the IP provider sends a pre-programmed security chip with the encrypted netlist to the system manager. To use the IP, the system manager instantiates it and connects the security chip to the FPGA. The encrypted netlist contains the IP core itself and a security module embedding the same identification number as the security chip. When the system starts, the security module checks whether the right security chip is present or not. This solution allows managing the quantity of IP cores sold. However, it faces similar limitations for the key management as bitstream encryption (as the key is contained in an encrypted netlist). Furthermore, this solution still relies on the key of the encrypted netlist. Hence, its main advantage is to allow per-device licensing.

3.2.3. Physically Unclonable Functions (PUFs). In view of the limitations of encrypted netlists and the use of security chips, the development of device-specific identification tools has become an intensive research topic in the

recent years. Physically unclonable Functions are among the frequently considered solutions for this purpose. They are challenge-response systems relying on uncontrollable random features inherent to the manufacturing processes. Their particularities lie in their output, which is easy to measure but assumably hard to predict if it has not been previously queried. Many kinds of PUF have been proposed in the literature. The following lines describe some of them, that are suitable for FPGAs. Implementation details are given in the related references.

1. *SRAM-based PUF:* initially proposed by Guajardo et al. [23], the SRAM-based PUF uses the initialization values of dedicated SRAM blocks. They consider a range of memory locations as challenges and start-up values at these locations as responses. These values depend on the small asymmetry between two cross-coupled inverters, ensuring that the start-up values will always be the same with high probability. Guajardo et al. define this kind of PUF as *intrinsic* because the PUF generating circuit is directly present in the design to protect. The main drawback of SRAM-based PUF with FPGAs is that most FPGA manufacturers initialize the embedded memory blocks to zero before loading the bitstream to avoid shortcuts in the reconfigurable circuitry.
2. *Flip-flop PUF:* proposed by Maes et al. [44], the flip-flop PUF uses flip-flops start-up values as responses similarly to the SRAM-based PUF. Maes et al. imagined this PUF because it is possible to prevent flip-flops from being reset. Hence, this allows having an efficient PUF suitable for every FPGA.
3. *Butterfly PUF:* proposed by Kumar et al. [40], the butterfly PUF is another solution to overcome the SRAM PUF reset drawback. It consists in two cross-coupled latches initialized with two different values to have an unstable operating point. The latches are initialized on an external signal. When this one is released, the stable state depends on the slight differences between the connecting wires which are designed using symmetrical paths on the FPGA matrix. The butterfly PUF needs manual routing to have symmetric paths and its performance highly depends on the targeted FPGA [51].
4. *LUT-based PUF:* proposed by Anderson [4], the LUT-based PUF harnesses the FPGA's LUT structure. It uses LUTs from the same basic logic block (slice or ALM), configured in shift-register, and the carry-chain logic. This PUF relies on delays introduced by the LUTs and the multiplexers. It uses the presence or absence of glitches along the carry-chain to determine the output bit. This PUF has the advantage to be completely described in HDL.
5. *Ring oscillator PUF:* introduced by Gassend et al. [19, 20], it mainly relies on a self-oscillating circuit and a counter. The ring oscillator produces an oscillating signal with a delay-dependent frequency. Besides, the counter measures the number of positive edges over a period of time. The obtained value is a good representation of the ring oscillator intrinsic delay. The main drawbacks of this kind of PUF are the limited number of possible challenges and the significant dynamic power consumption.
6. *Time-bounded PUF:* introduced by Majzoobi et al. [45], the time-bounded PUF relies on three flip-flops placed around the *Circuit Under Test* (CUT) :

the *Launch* FF, the *Sample* FF, and the *Capture* FF. Initially, the flip-flops are set to zero. Then, the *Launch* FF is set to one on the rising edge of the clock. This signal propagates through the CUT and is sampled by the *Sample* FF on the falling edge of the clock. The CUT adds a challenge-dependent delay which may be greater than the half of the clock period. Hence, the sampled value depends on it and is xor-ed with the true launched value to be captured by the *Capture* FF.

As the PUFs lead to unique and device-dependent challenge-reponse pairs, they can be used in many identification-related applications including IP protection. Simpson et al. [70] are among the few to have proposed FPGA-specific IP protection protocols using PUFs (Fig. 4). Their work relies on the assumption that the FPGA manufacturer has embedded a standard security module which contains two different hardware blocks: the PUF itself, used for hardware authentication and key generation, and a block cipher used for symmetric encryption and software authentication.

During a preliminary step, the FPGA manufacturer and the IP provider have sent authentication information to the trusted third party (TTP) including a challenge-response pair vector from the PUF ($\overline{\text{CRP}}$). Then, the protocol is a 3-player game between the system designer (SYS), the IP provider (IPP) and a TTP. First, the SYS sends a request to the TTP to obtain a particular IP. Then, the TTP forwards this request to the IPP attached with a PUF response that is used as encryption key. In the same time, the TTP sends the IP authenticity information attached with a PUF challenge that allows the SYS to recover the PUF response (i.e. the encryption key). Finally, IPP sends his encrypted IP to the SYS which is able to decrypt it into the device without having access to the IP itself. Hence, the SYS can not use the data for another platform without asking the IP provider. The strength of using PUFs in this kind of IP protection protocol comes from the hidden, secret, and non-volatile aspects of PUFs.

Starting from the Simpson et al.'s work, Guajardo et al. [23] proposed a protocol where the TTP cannot access the IP block exchanged between the IPP and the SYS. Indeed, as the TTP knows all the challenge-response pairs and since the SYS-IPP channel is public, he is able to access to the IP core. To avoid this, they introduced a public-key based operation. In a following work [24], the same team studied the advantages that asymmetric cryptography provides in this context. It allows that secret information from PUF never has to leave the FGPA unlike in previous works. This results in increased security guarantees.

In general, PUFs are interesting objects for dealing with security questions in non-volatile FPGAs, in particular when these devices do not contain any non-volatile memory that could be used to store a key. As far as IP protection is concerned, it can also allow per device licensing, without the need of any security chip. However, it also faces limitations, as the protected designs have to include the processing of the PUFs, which could possibly be removed by an adversary, e.g. if the security of the netlists is compromised.

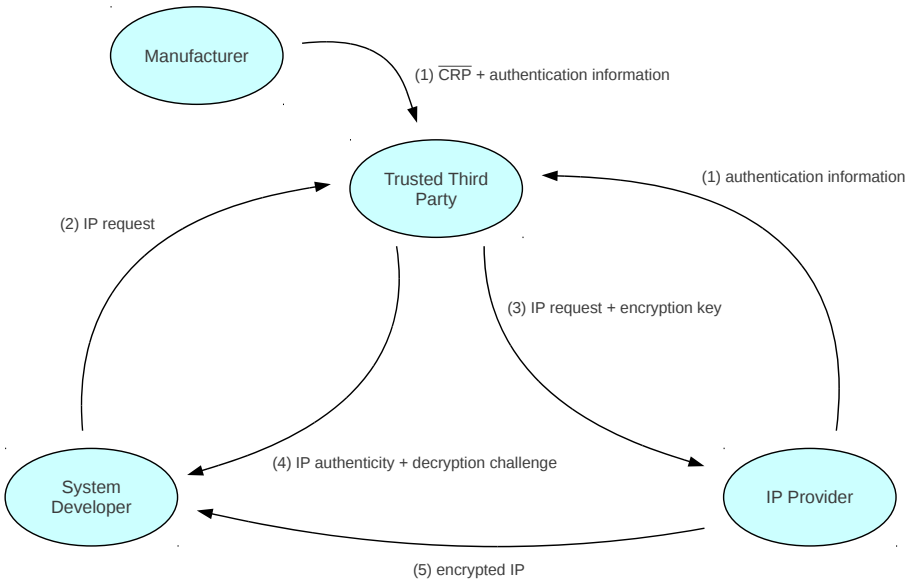

Fig. 4. Simspon et al.'s IP protection protocol based on PUFs.

3.2.4. IP watermarking Digital watermarking is the process of embedding an evidence of ownership into all types of digital content. The embedded information must be very difficult to remove. Hence, each copy of the content will also include the watermark information. This process is largely used in multimedia content by slightly modifying the data in an unperceivable way from a user point of view (e.g. hiding a signature in the highest frequencies of a picture). Similar techniques have been proposed for IP protection. However, watermarking for IP protection is more difficult as most of the times, when a chip needs to be tested, it is not available before returning completely assembled and packaged from the manufacturing process. Hence, e.g. the constraint-based watermarking proposed by Kahng et al. [31] and Narayan et al. [52] cannot capture such post-production features. Also, the main difference with other IP protection techniques lies in the fact that the watermarking is an *a posteriori* solution. That means that, unlike other solutions, the watermarking is searched and checked only if the IP owner has doubts on the IP authenticity (i.e. it allows detecting counterfeiting, but does not directly prevent it).

When watermarking IPs, care must first be taken that the functional correctness of the core is preserved. Also, in order to be efficient, an IP watermarking strategy should only use the usual design tools, not affect the performances of the core, be robust to removals or modifications, and be sufficient as proof-of-ownership. In this context, the two main goals of IP watermarking are the detectability and the proof-of-ownership [5]. The detectability means that, given

an IC (Integrated Circuit), the IP core designer is able to determine whether his IP core is used. The proof-of-ownership means that, given an IC, the IP core designer is able to prove to a third party that he owns the IP core used. A general survey and analysis of watermarking techniques is provided by Abdel-Hamid et al. in [1], and complete states-of-the-art for IP watermarking can be found in Drimer's and Ziener's PhD theses [14, 85].

A watermark can be embedded at the three different levels of the design flow, with pros and cons: the behavioral HDL description, the netlist generation, and the bitfile generation. Including the watermark in the HDL or the netlist allows protection of individual IP cores as they can be handled independently. However, if the adversary is able to break the netlist encryption, he can then (as for bitstream encryption and PUFs) easily remove the watermark. Including it in the bitfile may prevent the netlist encryption issue, but can only protect the whole system as the bitfile is generated by the system designer and not by the IP provider.

Next, different ways to recover a watermark signature exist, among which we find:

1. *Bitfile :* the bitfile can be read either using the readback command if it is activated, either by wire tapping the bus between the PROM (Programmable Read-Only Memory) and the FPGA if the bitfile is not encrypted.
2. *Ports :* some ports of the FPGA can be dedicated to the watermark reading. However, in the case of IPs integrated in larger systems, the system designer could deliberatly choose to disconnect those ports from the IP.
3. *Power :* introduced by Ziener et al. [86], as in cryptographic side-channel attacks, the clock frequency and the toggling logic can be extracted from the measured power traces.
4. *Electromagnetic radiations :* the EM radiations are easy to measure and offer a similar information as the power traces if the chip is not protected.
5. *Temperature :* proposed by Kean et al. [34], the temperature just needs a thermo-couple to measure it through the chip package without having access to the power pins. Although it is a rather slow process, it is currently the only commercially available approach.

As this chapter focuses specifically on FPGAs, we now report a few examples of watermarking techniques that are or could be applicable in this case.

Lach et al. [41, 42] propose to include a watermark signature in the unused LUTs at the bitstream level. Furthermore, Schmid et al. [66] improve it by tightly integrating the watermark with the LUTs of the design, so that simply removing the mark carrying components would damage the IP core. The watermark extraction is done knowing their positions and performing a readback of the bitfile.

Oliveira [55] proposes a Finite-State-Machine-based watermarking while Castillo et al. [8] propose a technique using unused parts of the distributed RAM memory. These two techniques offer the ease to be implemented in HDL but the main drawback is that they need FPGA ports to extract the watermark.

A side-channel based technique is introduced by Becker et al. [5] and consists in hiding a watermark signature below the noise floor in the power traces, by including leakage generating circuit depending on an identification number. The signature can be easily recovered with DPA. This solution is well-suited to tag netlist cores but it can hardly be applied to HDL IP cores, since the identification and the suppression of the watermarking circuit would then be easy.

Finally, as previously mentioned, Kean et al. [34] proposes a solution which consists in hiding the watermark signature in the chip's temperature. The underlying working is quite the same as the Becker et al.'s work but, instead of measuring power consumption leakages, they measure the heat generated by their circuit. This one is composed of multiple ring oscillators that are switch on/off according to the identification number. The tag signature can be easily found with cross-correlation.

4 Conclusions

FPGAs are useful for the implementation of security algorithms, because of the significant performance gains that they provide compared with software solutions. As they allow customizing and specializing the processing blocks, they also offer interesting opportunities for the design of countermeasures against non-invasive (passive) physical attacks. In particular, their inherent parallelism generally allows noisy measurements in side-channel attacks and less precise fault insertions in active attacks. Overall, cryptographic implementations in FPGAs can be seen as the result of a flexibility vs. efficiency and security trade-off. That is, for security and efficiency, it is best to fully take advantage of each given device architecture. But for flexibility, it is more attractive to have designs able to run on a large amount of devices. A similar statement can be made for the important problem of IP protection in reconfigurable devices. For flexibility reasons, it is desirable that the security relates to the netlists so that IPs can be easily simulated and integrated in larger designs. But for security reasons, the best solution would be to deal directly with bitstreams. IP and bitstream security is also limited by the difficult key management problem. Ideally, the integration of non-volatile keys and public key cryptography facilities in each device would be the best solution to allow the "per device" licensing of the IPs based on bitstreams. But present devices do not offer such facilities. Alternative solutions exist, based on the detection of a security chip, or Physically Unclonable Functions, but are then limited by some other assumptions (e.g. the difficulty to remove the "detection mechanisms from the design"). Watermarking based techniques are yet another way to detect IP theft a posteriori. These questions illustrate the rapidly evolving nature of security issues in reconfigurable computing, for which several important research problems remain open.

References

1. Amr T. Abdel-Hamid, Sofiène Tahar, and El Mostapha Aboulhamid. Ip watermarking techniques: Survey and comparison. In *IWSOC*, pages 60–65. IEEE Com-

puter Society, 2003.
2. Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.
3. Altera. `http://www.altera.com/`.
4. Jason H. Anderson. A PUF design for secure FPGA-based embedded systems. In *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pages 1 –6, jan. 2010.
5. Georg T. Becker, Markus Kasper, Amir Moradi, and Christof Paar. Side-channel based watermarks for integrated circuits. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 30 –35, june 2010.
6. Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Trans. Computers*, 52(4):492–505, 2003.
7. Philippe Bulens, François-Xavier Standaert, Jean-Jacques Quisquater, Pascal Pellegrin, and Gaël Rouvroy. Implementation of the AES-128 on Virtex-5 FPGAs. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 16–26. Springer, 2008.
8. Encarnación Castillo, Luis Parrilla, Antonio García, Antonio Lloris-Ruíz, and Uwe Meyer-Bäse. IPP watermarking technique for IP core protection on FPL devices. In *FPL*, pages 1–6, 2006.
9. Ricardo Chaves, Georgi Kuzmanov, Stamatis Vassiliadis, and Leonel Sousa. Reconfigurable memory based AES co-processor. In *IPDPS*. IEEE, 2006.
10. Pawel Chodowiec and Kris Gaj. Very compact FPGA implementation of the AES algorithm. In Walter et al. [82], pages 319–333.
11. Guerric Meurice de Dormale, Philippe Bulens, and Jean-Jacques Quisquater. Collision search for Elliptic Curve Discrete logarithm over GF(2^m) with FPGA. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 378–393. Springer, 2007.
12. Saar Drimer. Authentication of fpga bitstreams: Why and how. In Pedro C. Diniz, Eduardo Marques, Koen Bertels, Marcio Merino Fernandes, and João M. P. Cardoso, editors, *ARC*, volume 4419 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2007.
13. Saar Drimer. Security for volatile FPGAs. PhD dissertation, University of Cambridge Technical Report UCAM-CL-TR-763, 2009.
14. Saar Drimer. Security for volatile FPGAs. Technical Report UCAM-CL-TR-763, University of Cambridge, Computer Laboratory, November 2009.
15. Saar Drimer, Tim Güneysu, and Christof Paar. DSPs, BRAMs, and a pinch of logic: Extended recipes for AES on FPGAs. *TRETS*, 3(1), 2010.
16. Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on AES. *CoRR*, cs.CR/0301020, 2003.
17. Junfeng Fan, Daniel V. Bailey, Lejla Batina, Tim Güneysu, Christof Paar, and Ingrid Verbauwhede. Breaking Elliptic Curve Cryptosystems using reconfigurable hardware. In *FPL*, pages 133–138. IEEE, 2010.
18. Kris Gaj, Ekawat Homsirikamol, and Marcin Rogawski. Fair and comprehensive methodology for comparing hardware performance of fourteen round two SHA-3 candidates using FPGAs. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 264–278. Springer, 2010.

19. Blaise Gassend. Physical Random Functions. Master's thesis, MIT, USA, 2003.
20. Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *ACM Conference on Computer and Communications Security*, pages 148–160, New York, NY, USA, 2002. ACM Press.
21. Tim Good and Mohammed Benaissa. AES on FPGA from the fastest to the smallest. In Rao and Sunar [59], pages 427–440.
22. Louis Goubin and Mitsuru Matsui, editors. *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*. Springer, 2006.
23. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems Workshop*, volume 4727 of *LNCS*, pages 63–80, September 2007.
24. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. Physical unclonable functions and public-key crypto for FPGA IP protection. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 189 –195, aug. 2007.
25. Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2011.
26. Tim Güneysu and Christof Paar. Ultra high performance ECC over NIST primes on commercial FPGAs. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 2008.
27. Mohamed N. Hassan and Mohammed Benaissa. Efficient time-area scalable ECC processor using μ-coding technique. In M. Hasan and Tor Helleseth, editors, *Arithmetic of Finite Fields*, volume 6087 of *Lecture Notes in Computer Science*, pages 250–268. Springer Berlin / Heidelberg, 2010.
28. Mohamed N. Hassan and Mohammed Benaissa. Small footprint implementations of scalable ECC point multiplication on FPGA. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1 –4, may 2010.
29. Alireza Hodjat and Ingrid Verbauwhede. A 21.54 Gbits/s fully pipelined AES processor on FPGA. In *FCCM*, pages 308–309. IEEE Computer Society, 2004.
30. Kimmo U. Järvinen, Matti Tommiska, and Jorma Skyttä. A fully pipelined memoryless 17.8 Gbps AES-128 encryptor. In *FPGA*, pages 207–215, 2003.
31. Andrew B. Kahng, Darko Kirovski, Stefanus Mantik, Miodrag Potkonjak, and Jennifer L. Wong. Copy detection for intellectual property protection of VLSI designs. In *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, pages 600 –604, 1999.
32. Najeh Kamoun, Lilian Bossuet, and Adel Ghazel. SRAM-FPGA implementation of masked S-Box based DPA countermeasure for AES. In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 74–77. IEEE, 2009.
33. Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(12):1509–1517, 2002.
34. Tom Kean, David McLaren, and Carol Marsh. Verifying the authenticity of chip designs with the DesignTag system. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 59 –64, june 2008.
35. David Kenney. Energy efficiency analysis and implementation of AES on an FPGA. Master's thesis, University of Waterloo, Canada, 2008.

36. Stéphanie Kerckhof, François Durvaux, Nicolas Veyrat-Charvillon, Francesco Regazzoni, Guerric Meurice de Dormaele, and François-Xavier Standaert. Compact fpga implementations of the five sha-3 finalists. ECRYPT II Hash Workshop, Talinn, Estonia, May 2011.
37. Farouk Khelil, Mohamed Hamdi, Sylvain Guilley, Jean-Luc Danger, and Nidhal Selmane. Fault analysis attack on an FPGA AES implementation. In *NTMS'08*, pages 1–5, 2008.
38. Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal I. Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *LNCS*, pages 104–13. Springer, Berlin, September 1996.
39. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *LNCS*, pages 398–412. Springer, Berlin, August 1999.
40. Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 67–70, June 2008.
41. John Lach, William H. Mangione-Smith, and Miodrag Potkonjak. Signature hiding techniques for FPGA intellectual property protection. In *ICCAD*, pages 186–189, 1998.
42. John Lach, William H. Mangione-Smith, and Miodrag Potkonjak. Robust FPGA intellectual property protection through multiple small watermarks. In *DAC*, pages 831–836, 1999.
43. Bernhard Linke. Xilinx FPGA IFF copy protection with 1-wire SHA-1 secure memories. `http://www.maxim-ic.com/app-notes/index.mvp/id/3826`, jun 2006.
44. Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, page 17, Eindhoven, NL, 2008.
45. Mehrdad Majzoobi, Ahmed Elnably, and Farinaz Koushanfar. FPGA time-bounded unclonable authentication. In Rainer Bhme, Philip Fong, and Reihaneh Safavi-Naini, editors, *Information Hiding*, volume 6387 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2010.
46. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in Information Security. Springer, New York, 2007.
47. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Rao and Sunar [59], pages 157–171.
48. Stefan Mangard and Kai Schramm. Pinpointing the side-channel leakage of masked AES hardware implementations. In Goubin and Matsui [22], pages 76–90.
49. Nele Mentens, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. An FPGA implementation of Rijndael: Trade-offs for side-channel security. In *IFAC Workshop-PDS*, pages 493–498. Citeseer, 2004.
50. Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks extracting keys from Xilinx Virtex-II FPGAs. Cryptology ePrint Archive, Report 2011/390, 2011. `http://eprint.iacr.org/`.
51. Sergey Morozov, Abhranil Maiti, and Patrick Schaumont. An analysis of delay based PUF implementations on FPGA. In Phaophak Sirisuk, Fearghal Morgan, Tarek El-Ghazawi, and Hideharu Amano, editors, *Reconfigurable Computing: Architectures, Tools and Applications*, volume 5992 of *Lecture Notes in Computer Science*, pages 382–387. Springer Berlin / Heidelberg, 2010.

52. Naveen Narayan, Rexford D. Newbould, Jo Dale Carothers, Jeffrey J. Rodriguez, and W. Timothy Holman. IP protection for VLSI designs via watermarking of routes. In *ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International*, pages 406 –410, 2001.
53. NIST. http://csrc.nist.gov/groups/st/hash/sha-3/index.html.
54. NIST. Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, November 2001.
55. Arlindo L. Oliveira. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 20(9):1101–1117, 2001.
56. Siddika Berna Örs, Elisabeth Oswald, and Bart Preneel. Power-analysis attacks on an FPGA - first experimental results. In Walter et al. [82], pages 35–50.
57. Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In *CHES'03*, pages 77–88, 2003.
58. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
59. Josyula R. Rao and Berk Sunar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
60. Francesco Regazzoni, Thomas Eisenbarth, Luca Breveglieri, Paolo Ienne, and Israel Koren. Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In Cristiana Bolchini, Yong-Bin Kim, Dimitris Gizopoulos, and Mohammad Tehranipoor, editors, *23rd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2008)*, pages 202–210. IEEE Computer Society, 2008.
61. Francesco Regazzoni, Thomas Eisenbarth, Johann Großschädl, Luca Breveglieri, Paolo Ienne, Israel Koren, and Christof Paar. Power attacks resistance of cryptographic S-boxes with added error detection procedures. In Cristiana Bolchini, Yong-Bin Kim, Adelio Salsano, and Nur A. Touba, editors, *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 508–516. IEEE Computer Society, 2007.
62. Francesco Regazzoni, Yi Wang, and François-Xavier Standaert. FPGA implementations of the AES masked against power analysis attacks. In *COSADE 2011*, 2011.
63. G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat. Compact and efficient encryption/decryption module for fpga implementation of the aes rijndael very well suited for small embedded applications. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 2, pages 583 – 587 Vol.2, april 2004.
64. Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita RoyChowdhury. A diagonal fault attack on the Advanced Encryption Standard. Cryptology ePrint Archive, Report 2009/581, 2009. `http://eprint.iacr.org/`.
65. Sasebo. `http://staff.aist.go.jp/akashi.satoh/SASEBO/en/`.
66. Moritz Schmid, Daniel Ziener, and Jürgen Teich. Netlist-level IP protection by watermarking for LUT-based FPGAs. In *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT 2008)*, pages 209–216, Taipei, Taiwan, December 2008.

67. Nidhal Selmane, Shivam Bhasin, Sylvain Guilley, Tarik Graba, and Jean-Luc Danger. WDDL is protected against setup time violation attacks. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on*, pages 73 –83, sept. 2009.
68. Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger. Practical setup time violation attacks on AES. In *Proceedings of the 2008 Seventh European Dependable Computing Conference*, pages 91–96, Washington, DC, USA, 2008. IEEE Computer Society.
69. Li Shang, Alireza S. Kaviani, and Kusuma Bathala. Dynamic power consumption in virtex-II FPGA family. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, FPGA '02, pages 157–164, New York, NY, USA, 2002. ACM.
70. Eric Simpson and Patrick Schaumont. Offline hardware/software authentication for reconfigurable platforms. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 311–323. Springer Berlin / Heidelberg, 2006.
71. François-Xavier Standaert. Secure and efficient symmetric encryption using FPGAs. *Cryptographic Engineering*. Chapter 11, pp 295-320, Springer, 2009.
72. François-Xavier Standaert, François Macé, Eric Peeters, and Jean-Jacques Quisquater. Updates on the security of FPGAs against power analysis attacks. In Koen Bertels, João M. P. Cardoso, and Stamatis Vassiliadis, editors, *ARC*, volume 3985 of *Lecture Notes in Computer Science*, pages 335–346. Springer, 2006.
73. François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power analysis of an FPGA: Implementation of Rijndael: Is pipelining a DPA countermeasure? In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2004.
74. François-Xavier Standaert, Eric Peeters, Gaël Rouvroy, and Jean-Jacques Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2):383–394, 2006.
75. François-Xavier Standaert, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. Efficient implementation of rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs. In Walter et al. [82], pages 334–350.
76. François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, David Samyde, and Jean-Jacques Quisquater. Power analysis of fpgas: How practical is the attack? In Peter Y. K. Cheung, George A. Constantinides, and José T. de Sousa, editors, *FPL*, volume 2778 of *Lecture Notes in Computer Science*, pages 701–711. Springer, 2003.
77. Daisuke Suzuki and Minoru Saeki. Security evaluation of dpa countermeasures using dual-rail pre-charge logic style. In Goubin and Matsui [22], pages 255–269.
78. Helion Technology. http://www.heliontech.com/.
79. Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.
80. Stephen Trimberger, Jason Moore, and Weiguang Lu. Authenticated encryption for fpga bitstreams. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '11, pages 83–86, New York, NY, USA, 2011. ACM.
81. Frank Vahid. The softening of hardware. *Computer*, 36:27–34, April 2003.

82. Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*. Springer, 2003.
83. Thomas Wollinger, Jorge Guajardo, and Christof Paar. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Trans. Embed. Comput. Syst.*, 3:534–574, August 2004.
84. Xilinx. `http://www.xilinx.com/`.
85. Daniel Ziener. *Techniques for Increasing Security and Reliability of IP Cores Embedded in FPGA and ASIC Designs*. Dissertation, University of Erlangen-Nuremberg, Germany, July 2010. Verlag Dr. Hut, Munich, Germany.
86. Daniel Ziener and Jürgen Teich. Power signature watermarking of IP cores for FPGAs. *Signal Processing Systems*, 51(1):123–136, 2008.
87. The SHA-3 Zoo. http://ehash.iaik.tugraz.at/wiki/the_sha-3_zoo.