PLL to the Rescue: A Novel EM Fault Countermeasure

Miura, Noriyuki; Najm, Zakaria; He, Wei; Bhasin, Shivam; Ngo, Xuan Thuy; Nagata, Makoto; Danger, Jean-Luc

2016

Miura, N., Najm, Z., He, W., Bhasin, S., Ngo, X. T., Nagata, M., et al. (2016). PLL to the rescue: A Novel EM Fault Countermeasure. Proceedings of the 53rd Annual Design Automation Conference, 90-.

<https://hdl.handle.net/10356/83379>

<https://doi.org/10.1145/2897937.2898065>

© 2016 Association for Computing Machinery (ACM). This is the author created version of a work that has been peer reviewed and accepted for publication by Proceedings of the 53rd Annual Design Automation Conference, ACM. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [<http://dx.doi.org/10.1145/2897937.2898065>].

Downloaded on 26 Apr 2023 16:29:00 SGT

PLL to the Rescue: A Novel EM Fault Countermeasure

Noriyuki Miura[†], Zakaria Najm^{*}, Wei He[‡], Shivam Bhasin[‡],
Xuan Thuy Ngo^{*}, Makoto Nagata[†], Jean-Luc Danger^{*}

[†]Kobe University, Kobe, Japan
{miura, nagata}@cs.kobe-u.ac.jp

^{*}Telecom Paristech, Paris, France
{znajm,xngo, danger}@telecom-paristech.fr

[‡]Nanyang Technological University, Singapore
{he.wei, sbhasin}@ntu.edu.sg

ABSTRACT

Electromagnetic injection (EMI) is a powerful and precise technique for fault injection in modern ICs. This intentional fault can be utilized to steal secret information hidden inside of ICs. Unlike laser fault injection, tedious package decapsulation is not needed for EMI, which reduces an attacker's cost and thus causes a serious information security threat. In this paper, a PLL-based sensor circuit is proposed to detect EMI reactively on chip. A fully automatic design flow is devised to integrate the proposed sensor together with a cryptographic processor. A high fault detection coverage and a small hardware overhead are demonstrated experimentally on an FPGA platform.

CCS Concepts

•Security and privacy → Side-channel analysis and countermeasures;

Keywords

Fault Attacks, Electromagnetic Attacks, PLL, Countermeasure

1. INTRODUCTION

Embedded systems play an important role in a current advanced information society and will become fundamentally critical in a coming Internet of Things (IoT) era. Although IoT could benefit a human life significantly, the hardware security of the IoT embedded system becomes a serious technical issue. Since the IoT devices are distributed anywhere and everywhere in the human life and collect precious private information, the systems could be a potential target of physical attacks.

Side-channel and fault attacks are well-known physical attacks. Side-channel attack (SCA [3]) is passive in nature, which basically search for sensitive information by observing unintentional but natural side-channel leakage of physical parameters, such as power consumption, EM radiation, and operation timing of ICs. Fault injection attack (FIA [2]) is on the other hand active in nature, which intentionally turns the target device in an abnormal operate condition and induces computation faults. This fault operation can be exploited to reveal sensitive information. The abnormal operate conditions can be realized by several methods. The most common and low-cost methods include under-powering, over-clocking, and extreme-heating/cooling. These techniques have a global impact and thus lack precision in location and type of faults. Efficient FIA needs more local fault injection in time and space domains. Sophisticated techniques such as laser injection or focused ion beam (FIB) injection are efficient but very expensive for the attackers and both of them require package decapsulation.

Electromagnetic injection (EMI) is one of the efficient and low-cost methods for fault injection in modern ICs [7]. There are two main advantages in EMI. Firstly, unlike laser injection, EMI can be performed without detailed decapsulation for opening of the target circuit since the EM field can penetrate the package and therefore no dedicated IC chip preparation is required. This reduces the attacker cost significantly by saving the time and money on careful decapsulation and accidental destruction of the chip. Moreover, secure chips often deploy several physical sensors which are triggered upon the opening of the package. Such countermeasures can be easily bypassed by EMI. The second advantage of EMI is high configurability. Depending on the various parameters like probe size, orientation, and injection pulse width/frequency, the fault injection timing and spot can be precisely adjusted.

There are a few research works done to develop countermeasures against this EMI. One of the best known methods is glitch detectors [8]. Zussa et al. proposed to integrate multiple FF-based distributed sensors and their clock networks to detect EMI-induced glitches before inducing the actual fault operation in the protected core circuit. When one of the sensors detects glitches, an alarm signal is raised. The biggest problem is its sensitivity control. Since the detector utilizes simple delay line based sensor, the EMI detection

Figure 1: Block diagram of PLL-based EMI countermeasure and its operation waveforms in (a) lock and (b) unlock state.

threshold is broadly fluctuated over PVT variations. In addition, there is no clear criteria on how many and how finely the sensors should be distributed to guarantee the security margin against EMI.

In this paper, a novel countermeasure to detect EMI is proposed. This countermeasure is based on PLL (phase locked loop). A PLL is a clock control circuit and is found easily in modern ASICs and FPGAs. Whenever, a PLL is started, it needs multiple cycles to generate a stable clock. This state where the PLL generates stable clock is known as a 'locked' state. When the parameters of the PLL is modified during operation, its lock is broken and it again needs multiple clock cycles to come back to the 'locked' state. Proposed countermeasure exploits this property of PLL. Precisely, the PLL monitors an internally generated clock. When an attacker attempts to perform EMI, the internal clock changes phase/frequency which leads to unlocking of PLL and thus detection of disturbances. Since the proposed system requires precise design controls in implementation that cannot be done by generic vendor tools, an automated implementation flow is devised in Xilinx FPGA environment for properly manipulating the bottom layer logic elements, merging the protection system and the security sensitive circuit to be protected.

The rest of the paper is organized as follows: Sec 2 describes the basic functioning of PLL and how it can be used to design EMI countermeasure. Sec 3 details the fully automated design flow for application of the proposed countermeasure in an FPGA platform. Experimental results are shown in Sec 4 and final conclusions are drawn in Sec. 5.

2. PLL-BASED EMI COUNTERMEASURE

2.1 Concept

Phase Locked Loop or PLL, is a clock timing control circuitry which detects phase and frequency difference between two clock sources and feedback-controls to keep synchronization between them. Fundamental building blocks of PLL are a Phase-Frequency Detector (PFD), a Loop Filter (LF), and a Voltage-Controlled Oscillator (VCO) as depicted in Fig. 1. PFD compares two clock input timings: one is an external reference clock $Rclk$ and feedback internal core clock $Fbclk$. Based on the timing comparison, PFD produces up and down pulses (Up , $Down$) whose duration time represents phase and frequency difference between two clock sources.

A simple implementation of PFD consists of only one AND gate and two FFs with reset input. LF generates the control voltage V_{CTRL} for VCO depending on the up/down pulses given from PFD to adjust phase and frequency of VCO for the synchronization. Insertion of a frequency divider ($1/N$) in the feedback loop, enables to generate N -times higher-frequency core clock $Cclk$ synchronized with respect to the external reference clock $Rclk$. PLL is therefore widely used for global clock synchronization in the LSI-based electronic systems and thus almost of all the modern high-performance SoC ASICs and FPGAs integrate PLL on-chip.

This existing PLL can be utilized for an EMI countermeasure. Fig. 1 describes the basic concept. PLL can be seen as a continuous monitor of internal clock stability. Since the purpose of EMI is an intentional induction of instantaneous abnormal circuit operations by large-power EM pulse, PLL can detect this EMI attack by utilizing feedback clock path as a watch dog signal. In a stable steady state of the PLL loop, both the phase and frequency are locked between the external reference clock and the internal clock sources. Thus PFD only produces almost-no or very short up/down pulses in a locked state (Fig. 1 (a)). Once PLL loses lock due to instantaneous glitches or jitter in the feedback clock path, PLL starts to recover lock by producing explicit up/down pulses for many clock cycles (Fig. 1 (b)). This locked/unlocked state can be easily distinguished by a digital Loop-State Monitor (LSM). By distributing the feedback clock path all over the security sensitive crypto core to be protected, the EMI attack can be easily captured as an unlock event in the PLL loop. In other word, PLL amplifies the instantaneous erroneous operation in the watch-dog clock as a digital attack-warning signal. Once the attack is captured on-chip, the crypto core can be protected by disabling the core or operating it in dummy operation mode, such as in [6].

For further enhancing the security level, the reference clock path is together included in the watch-dog clock paths (Fig. 2). This modified configuration makes it possible to detect the EMI attack before the actual fault operation is induced in the crypto core. In addition, malicious attacks, such as EMI injection into the control loop itself, can be detected since these attacks anyway cause the unlocked state of PLL. A repeater configuration (Fig. 2 (a)) can provide flexibility for the synchronization with the external clock source $Rclk$. This configuration exposes the clock port to the attackers however the fault attack exploiting this clock port is very difficult. Over-clocking or glitch insertion of $Rclk$ can be detected by PLL in advance of the actual fault operation. Also, by carefully designing the maximum operating frequency of the crypto core to be higher than the PLL tuning range, a Fault Sensitivity Analysis (FSA [5]) is also disabled (the attacker only see the fault operation of PLL). A ring-oscillator (RO) configuration (Fig. 2 (b)) hides the clock port on-chip for maximally enhancing the security. No exposed port is available to the attackers. In this configuration, the crypto core asynchronously communicates with other circuits in a hand-shaking manner which is common for the security core design. A test chip in this work integrates the asynchronous crypto core with the EMI countermeasure in RO configuration for the highest security demonstration. The technical challenges lie in (1) how to efficiently implement the crypto core with the countermeasure in a fully automatic design flow; (2) how to adjust the sen-

Figure 2: Security-enhanced implementation of EMI countermeasure with reference clock watch dog in (a) repeater and (b) ring-oscillator configuration.

sitivity of the attack detection to guarantee a high enough security margin i.e., countermeasure raises alarm much before the crypto core is in fault.

2.2 Implementation Details

The proposed countermeasure uses an active ring oscillator (RO) as a watch-dog circuit (Fig 2(b)). The frequency f of a RO can be defined as $\frac{1}{2 \cdot t \cdot n}$, where t is gate delay and n is number of inverters. n is chosen as 1 to allow maximal oscillating frequency. The delay t in detail has two significant components: one is gate delay t_g and the other is routing delay t_r . As shown in Fig. 2(b), RO in our implementation is designed in a way to thoroughly envelop the sensitive module by making multiple loops to guarantee the full coverage of EMI detection. The RO delay is mostly composed of routing delay including buffers delay and therefore t_r cannot be neglected. The operating frequency f can be finally written as $f = \frac{1}{2 \cdot (t_g + t_r)}$. f tends to be as low as several MHz because long single routing is needed for the detection coverage. However, the operating frequency of the crypto core can be still increased by adjusting the clock division rate of the PLL feedback path. The performance degradation of the crypto processing can be minimized.

When an EM injection is made on the chip, the EM glitch will try to disturb the sensitive core. Since the RO is routed over the sensitive core, the glitch will change the routing time delay to a value t'_r . The modified frequency of RO will be $f' = \frac{1}{2 \cdot (t_g + t'_r)}$. This sudden change in frequency will impact the phase of the RO, which in turn will force the PLL to unlock state. Thus, the *LOCKED* signal of LSM raises an alarm.

To demonstrate this phenomena, simple experiments are performed on Spartan-6 FPGA. PLL_ADV block on Spartan-6 FPGA is used as a PLL module, which receives external clock as input. The CLK_FB_OUT is feedback to CLK_FB_IN for self-calibration, as shown in Fig. 3(a). The *LOCKED* output of the PLL is observed on an external pin. The results are shown in Fig. 3(b). When the injected pulse disturbs the internal clock routing, the PLL is unlocked and *LOCKED* signal (in green) moves to zero. EMI pulse (in yellow) appears multiple cycles after the trigger (in pink) i.e., triggering delay. As soon as the EMI pulse appears, the *LOCKED* signal goes down asynchronously and raises an alarm. The EMI pulse disturbs the phase of the CLK_FB_IN, which leads to breaking of the lock. A disturbance on the (asynchronous) reset pin of the PLL can also have similar affects. When the EM pulse is removed, the PLL starts to stabilize itself and the *LOCKED* signal stays low during this period for multiple cycles.

A similar behavior is observed by changing the phase of externally-input reference clock CLK_IN. However a gradual and slow change in frequency of CLK_IN does not triggers

Figure 3: Validation of the countermeasure principle on Spartan-6. (a) Configuration of PLL, (b) Observed signal on scilloscope with trigger in pink, EMI pulse in yellow, CLK_OUT in purple and LOCKED in green.

the alarm. The attacker can possibly exploit the external clock to perform a FSA [5] in the crypto core. By carefully designing the maximum operating frequency of the crypto core to be higher than the tuning range of PLL, FSA can be disabled. However this is not always possible for any crypto core and existing PLL pair. The clock generated by the RO is therefore utilized as the reference clock input to derive the clock source for the crypto core. This configuration enhances the PVT variation tolerance of the countermeasure. Since a change in process, supply voltage, and temperature will impact both the crypto core and RO in a similar way, susceptibility (sensitivity) against EMI is also changed similarly. In addition, the precise fault control by intentional disturbance of supply voltage and temperature becomes difficult.

An EM attack sensor [6] was presented as a countermeasure against a passive side-channel attack. This sensor was build on a LC coil acting as a probe sensor by detecting frequency shift with respect to another calibrating coil. It can detect micro EM probes which approach the circuit at a proximity within 0.1 mm. In our approach, the EM-probe detection range is as long as few cms as it detects active EM probes. These two techniques can be together integrated to cover the protection against a passive proximity EM analysis attack and an active EMI.

3. DESIGN AUTOMATION

An automatic design flow is devised to implement the proposed countermeasure together with the cryptographic processor to be protected. The detailed explanation is separated into two parts. First, the procedure to automatically place and route the watch-dog RO in an FPGA is explained. Second, the detailed design flow to integrate RO and PLL together with the crypto core.

3.1 Controllable RO Routing Flow

Due to limited routing freedom in FPGAs, routing manipulation is almost impossible in an automated manner with standard vendor tools. To implement the RO in the proposed countermeasure, the routing from inverter output to its own input should be sufficiently long to thoroughly cover all the sensitive block. The EM probe trying to inject fault will therefore interfere with RO. However, using the standard commercial router only results in very uncontrollable and short loops, as the tools are designed for resource op-

timization by default. In order to properly control the RO routing path, a customized tool is constructed over RapidSmith library [4] to manipulate the logic and routing in a Xilinx environment. The proposed design flow mainly comprises of 4 distinct stages.

1. 4 single-inverter RO modules are instantiated by a HDL source code, and deployed on the 4 corners of a rectangular region to be covered. 3 of the 4 ROs are actually dummy for positioning the region boundary of the watch-dog clock path. The preliminary design is implemented by generic FPGA design flow to create the post P&R `ncd` file, as shown in Fig 4(a).
2. The `ncd` file is parsed by the customised analysis tools to be converted into `xd1` format, which is human readable and contains all the implementation details mapping to a specific device. The basic routing element in Xilinx environment is “node” that is used to concatenate different instances and populate a complete routing network. The nodes of the 4 ROs are extracted from each RO modules in the `xd1` file [2]. For the main RO, a **source** node i.e., output pin of inverter (labeled node 1.1) and the **sink** node i.e., input node of inverter (labeled node 1.2) are extracted as shown in Fig 4(b).
3. In this step, the dummy ROs are simply removed from the `xd1`, only their nodes are left in the design. On the contrary, the LUT of the main RO is intact, but all the routings are removed, except the node 1.1 and node 1.2.
4. Since the vendor router is just capable of routing a path from **source** to **sink**, a custom **node router** is developed that can partially route a path between the two designated nodes. Following the previous steps, 4 independent routings are performed using the node router: node 1.1 $\rightarrow$ node 2 $\rightarrow$ node 3 $\rightarrow$ node 4 $\rightarrow$ node 1.2. Each individual route follows the generic routing protocol which results in the shortest path between two nodes. Next, a long routing path is completed that is positioned to be a rectangular by the selected corner nodes (1.1, 1.2, 2, 3, 4) as seen in Fig 4(d).

The same flow can be simply extended when the RO has a zig-zag shape (Fig. 3) and more than 4 nodes can be identified.

3.2 Co-Integration Flow of Sensor and Crypto Core

This section presents an automatic design flow to implement the countermeasure with a cryptographic processor core. The module to be protected can be any other sensitive circuit, such as CPU, key storage, etc. In this work, a cryptographic core of Simon block cipher is considered as an design example. All the steps are automated and concatenated using Java, which calls Xilinx tools in command line mode. The `xd1` files are also processed using customised tools [2]. The main procedure is sketched in Fig. 5 and involves the following steps:

1. The flow starts with the HDL description of the cipher i.e., `cipher.v` and the specific placement constraints on logic region. The cipher is synthesized, placed and routed to extract the `ncd`, followed by `xd1` file of the cipher implementation.

Figure 4: Exemplary design flow to implement a RO with restrained routing requirements.

2. The placement constraints from the previous steps are also used to define the corner nodes of the 4 ROs implemented. 3 out of 4 ROs are dummy modules in nature and are eventually removed. Nevertheless, the corner nodes can also be extracted from `cipher.xd1` rather than defining in the constraint file.
3. The EM countermeasure previously explained is also implemented and converted to `xd1` file.
4. The `RO.xd1` is then modified to envelop the cipher module. This procedure is explained in previous subsection (see Fig. 4). An enable signal is also added to the final RO in an automatic way using `fpga_edline` commands at this step.
5. In the penultimate step, the three `xd1` designs are combined (cipher, PLL, modified_RO) and concatenated.
6. The final step is the bitstream generation from the merged design.

A high-level overview of the flow is illustrated in Fig 6. The automation methodology can also be extended to ASIC design flow, of course with different tool tricks.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

The proposed countermeasure is validated on SASEBO-W board, which carries a Spartan-6 FPGA from Xilinx. Spartan 6 FPGA is packaged in frontside (FG) package which makes it easier to perform EM injection as compared to other FPGA that are packaged in flip chip (FF). The chip package stays intact as EMI does not need package removal.

The full setup for EMI test is presented in Fig. 7. The signal generator is used to fine tune the frequency with a precision of 1 ps. A 300W (55dB) broadband, 400MHz class A amplifier allows injection of very short EM pulses of width of 1.5-ns. A remotely controllable XYZ axes table (not shown)

Figure 5: Design flow of automatic co-integration of crypto core and countermeasure.

Figure 6: Conceptual overview of design flow.

is used to perform a full fault cartography with a spacial resolution of $1.0 \mu\text{m}$. The XYZ table scans the entire chip and generates a cartography as a 30×30 array.

4.2 Target Circuit

The countermeasure is validated on a hardened cryptographic module i.e., a block cipher which has fault detection capabilities in-built. Using such hardened cryptographic primitives simplifies the analysis part and it is easier to distinguish exploitable faults from faults which occur on the peripherals. The target circuit is a one cycle per round encryption only implementation of Simon32/64 block cipher implemented in Verilog. It is hardened for fault detection using encoded circuits [1] with a linear code of dual dis-

Figure 7: Picture of experimental setup for fault injection.

Figure 8: Sensitivity of the proposed countermeasure against EM pulse injection.

tance of 5. Encoded circuits operate by encoding the internal state of the circuit, using a linear code, which is then mixed with a random number. If a fault is injected inside the encoded circuit, it can be simply detected by decoding the random number and comparing with the initial random number. Any fault in the computation of cryptography will modify the code and hence the decoded random number will differ. More details on this technique can be found in [1]. Encoded circuits based hardening is applicable to any digital circuit. The validation is done on Simon32/64 core, as it stays small in area even after hardening and it is easier to analyze the faults.

Next, the proposed countermeasure is implemented on hardened Simon32/64 core using the design flow detailed in Sec. 3. The RO is routed using 12 nodes and the design flow converges in single iteration, without any conflicts. The overall cost of hardened Simon32/64 is 578 *flip-flops* and 1658 *LUTs* for Simon32/64 and the UART communication interface. Maximal operating frequency of the design is 50 MHz. Proposed countermeasure uses one *PLL_ADV* primitive for the PLL functionality and single *LUT* as an inverter along with routing resources to realize the RO. The circuit is driven close to maximal frequency so that faults are easier to inject. Moreover, the frequency of oscillation of the RO can be estimated from FPGA editor and PLL can be tuned accordingly. The hardened Simon core along with proposed countermeasure is validated on SASEBO-W platform. The analysis is conducted using the EMI platform explained in Sec.4.1. In the next section, the results of EMI injection are presented and analyzed.

4.3 Experimental Results

The experiments are devised to test two parameters i.e., the sensitivity of countermeasure and the spatial coverage of the countermeasure. Sensitivity means that the countermeasure is expected to raise an alarm much before the EM pulse faults the sensitive circuit. This is done by fixing the position of the probe on a point on top of the sensitive core. The impact of injecting EM pulses with varying power intensity is shown in Fig. 8. The proposed countermeasure raises an alarm for EM pulses of power as low as 38dBm. On the other hand, faults in sensitive core (Simon) start appearing at a minimum power of 54dBm. The security margin of the proposed countermeasure can be estimated to 19dBm from Fig. 8. At 64dBm, the faults discontinue, i.e., the EMI setup has reached its practical limit. Moreover, repetition of this experiment at several other points over SIMON32/64 depicts similar trend. None of the performed experiment shows reverse trend i.e., faults in sensitive core are injected without raising the alarm.

Figure 9: EM cartography of the (a) detection rate, (b) undetected faults on Spartan-6 FPGA implementing Encoded Simon32/64 and proposed countermeasure under EMI setup.

Next experiments studies the spatial coverage of the fault injection. The FPGA is scanned by a remotely controlled XYZ table carrying the EM injection probe and the FPGA is scanned using 30×30 grid. The fault detection capability of the EMI countermeasure is shown using a color map in Fig 9(a). The current results are at injection power of 53dBm i.e., in the region of fault injection on Simon. The expected region over Simon core and related I/O (i.e., center of FPGA) reports 100% fault detection. The fault detection probability falls to less than 100% in certain regions. However, the core is totally encircled by the watch-dog RO including some margin, thus such injection do not impact the Simon core.

There were a couple of pixels on the cartography, where some unexpected behavior occurred, as shown in Fig. 9(b). These were around pixel (27,18). Although no part of Simon core or the countermeasure is placed around these pixels but still some faulty output can be observed. Moreover, this faulty output turned out to be non-exploitable. From these observations, it can only be speculated that the EM pulse violates an internal (hidden) sensor which disturbs the FPGA functioning. It could also be due to disturbance of some power delivery or clock network or PLL itself. However, as the internal architecture of FPGA is not known, this hypothesis cannot be confirmed. Referring back to Fig. 8, at higher power some undetected faults are injected with a probability lower than of 0.01. When the the power of EM pulse is high, an injection triggers similar sensors and unexpected behavior is observed.

All the experiments were performed without any chip decapsulation. So it is hard to point exact location of crypto-core and sensor routing on the cartography. The FPGA die only covers a part of the package. Future work can explore techniques like X-ray imaging for deeper analysis.

4.4 Discussion

This section compares countermeasure proposed in this work with the state of the art. In [8], authors use multiple glitch detectors to detect local EM injection. Their idea is simple and works very well. Now comparing with proposed scheme, glitch detectors can have several shortcomings. First of all, there is no defined methodology to find the required number and placement of glitch detectors, which makes automation difficult. Proposed scheme needs only one RO and an existing PLL, and can be easily automated. The need of PLL on-chip is one limitation of proposed scheme however PLL exists almost of all the modern FPGAs and ASICs. The placement of proposed scheme is easy as it needs to only envelope all the sensitive core

Table 1: Qualitative Comparison of Proposed Countermeasure with State of The Art

Property	Glitch Detector [8]	This Work
Design Flow	Semi-Automatic	Fully-Automatic
Placement	Difficult	Envelope target
PLL Necessity	No Need	PLL (Existing)
Composition	n Detectors	1 PLL + 1 RO
PVT Variation Immunity	Low	High

thoroughly with wide enough security margin. Every glitch detector works as stand-alone environment sensor, and thus every detector has different sensitivity to local PVT and process variation. The sensitivity of proposed countermeasure is majority routing spread over a large area, which make PVT and process variation sensitivity limited. These properties are summarized in Table 1.

5. CONCLUSIONS

EMI has recently emerged as a precise and powerful fault injection technique. In this paper, a circuit level countermeasure against EMI is proposed. It uses a PLL circuit and a free running oscillator. A fully automated design flow is provided for supporting the implementation of the proposed countermeasure system. The protected scheme is validated on Spartan-6 FPGA. Results show that the proposal detects all the faults targeting the sensitive core with significant security margin of 19dBm. Further work will focus on better understanding of unexpected fault behavior from hidden architecture of FPGAs, using techniques like X-ray imaging.

6. REFERENCES

- [1] S. Bhasin, J. Danger, S. Guilley, Z. Najm, and X. T. Ngo. Linear Complementary Dual Code Improvement to Strengthen Encoded Circuit Against Hardware Trojan Horses. In *IEEE HOST 2015*.
- [2] W. He, A. Otero, E. de la Torre, and T. Riesgo. Customized and automated routing repair toolset towards side-channel analysis resistant dual rail logic. *Microprocessors and Microsystems*, 38(8):899–910, 2014.
- [3] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. *CRYPTO '99*, pages 388–397, 1999.
- [4] C. Lavin, M. Padilla, P. Lundrigan, B. Nelson, and B. Hutchings. Rapid prototyping tools for fpga designs: Rapidsmith. In *FPT'10*, pages 353–356. IEEE, 2010.
- [5] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta. Fault Sensitivity Analysis. In *CHES*, volume 6225 of *LNCS*, pages 320–334. Springer, 2010. Santa Barbara, CA, USA.
- [6] N. Miura, D. Fujimoto, M. Nagata, N. Homma, Y. Hayashi, and T. Aoki. EM attack sensor: concept, circuit, and design-automation methodology. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 176:1–176:6, 2015.
- [7] F. Poucheret, K. Tobich, M. Lisart, L. Chusseau, B. Robisson, and P. Maurine. Local and direct EM injection of power into CMOS integrated circuits. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2011.
- [8] L. Zussa, A. Dehbaoui, K. Tobich, J. Dutertre, P. Maurine, L. Guillaume-Sage, J. Clédière, and A. Tria. Efficiency of a glitch detector against electromagnetic fault injection. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014*, pages 1–6, 2014.