Linear Complementary Dual Code Improvement to Strengthen Encoded Circuit Against Hardware Trojan Horses

Xuan Thuy Ngo*, Shivam Bhasin*,‡, Jean-Luc Danger*†, Sylvain Guilley*†, Zakaria Najm*
* Institut MINES-TELECOM, TELECOM ParisTech, CNRS LTCI (UMR 5141).
† Secure-IC S.A.S., 80 avenue des Buttes de Coësmes, 35 700 Rennes, FRANCE.
‡ Temasek Laboratories, NTU, Singapore.

***Abstract*—Hardware Trojan Horses (HTH) are a serious threat to semiconductor industry with significant economic impact. We introduced in [10] a method called "encoded circuit", which both prevents and detects HTH. We achieved this goal using Linear Complementary Dual (LCD) codes. In this paper, we achieve a lower overhead and a better tunability by using a Linear Complementary Pair (LCP) of codes, which are not necessarily dual. LCP have two security parameters d_{Trigger} and d_{Payload}, such that the knowledge of strictly less than d_{Trigger} bits of the encoded state reveals no information about the actual state; in addition, any HTH which modifies strictly less than d_{Payload} bits of encoded state, will produce an invalid codeword. The application on an 8-bit processor shows the improvement of the new LCP codes. We also show that it is possible to fully automate CAD flow to generate encoded circuits with LCP codes. Finally we encode a SIMON cryptographic co-processor and test its resistance against physical attacks like Side-Channel Analysis (SCA) and Fault Injection Analysis (FIA).**

***Index Terms*—Encoding, Hardware Trojan Horses (HTH), Side-channel attack, Fault injection attack, Probing attack, Linear Complementary Dual (LCD) codes, Linear Complementary Pair (LCP) of codes, Minimal distance, Dual distance.**

I. Introduction

Nowadays, more and more semiconductor companies outsource their IC designs and fabrication steps because of their high cost and complexity. Nevertheless, this trend opens the door for a dangerous attack named Hardware Trojan Horses (HTH) insertion. A HTH is a malicious module inserted in the original Integrated Circuit (IC) during at design or fabrication stage. One inserted, HTH can effectuate various dangerous task like Denial of Service, leakage of sensible data via circuit outputs, etc [9]. A HTH can be globally seen as a composition of two parts:

- **Trigger**: which *reads* the target circuit state (to trigger its malicious function).
- **Payload**: which *writes* on the target circuit state (to run its malicious function).

Once inserted, one cannot remove a HTH, therefore HTH has become a hot topic in the hardware security field.

Xuan Thuy Ngo is the corresponding author. This project has been funded by the French Government, under grant FUI #14 **HOMERE** 959 (**H**ardware tr**O**jans : **M**enaces et robust**E**sse des ci**R**cuits int**E**grés).

One prominent research direction to fight against HTH relies on detection. Detection is based on several techniques like hardware software co-design [5], reconfigurable logic [1], logic testing [3] and side-channel analysis [2]. But each detection method has its own limitations. For example, logic testing cannot cover all possible test vectors when the IC is complex. The side-channel analysis needs a golden model to build the reference. A standard technique to obtain a golden model is still an open question.

Because of the limitations in detection techniques, prevention methods have emerged as an alternate research axis to fight against HTH. HTH prevention consists in altering the original circuit in order to thwart HTH insertion. In the state of the art, there are a few works on the HTH prevention method. Chakraborty et al. [6] initially presented a prevention method which obfuscates the state machine of the IC registers. But the presented technique protects only the control part, while the data-sensitive part remains attackable. In ODETTE [3], Authors are more intended to raise the HTH activity for a better detectability than a proactive prevention. In [11], authors propose the method named "EPIC" which encodes the combinational logic part. EPIC is based on "security by obscurity" hence probing can be done after configuration to recover the key. This EPIC method is static therefore an attacker can create a HTH which learns the key and subsequently gets activated, hence bypassing the EPIC method. Recently in [10], we propose a prevention technique named "encoded circuit". This technique, based on the paper of Ishai, Sahai and Wagner presented at CRYPTO 2003 [8], encodes and masks all internal registers (including control and data registers) with a Linear Complementary Dual (LCD) code. It ensures that any HTH connected to strictly less than d registers (d is the minimal distance of chosen code used as security parameter) will be ineffective.

In this paper, we study the performance of the "encoded circuit" method using a Linear Complementary Pair (LCP) of codes. This is a generalization of LCD, with two different security parameters: d_{Trigger} and d_{Payload}. These parameters are the dual distance of code D and minimal distance of code C used to encode the random mask and the IC sensitive data respectively. The parameter d_{Trigger} is used to foil the HTH "Trigger" part. It ensures that HTH, if connected to strictly

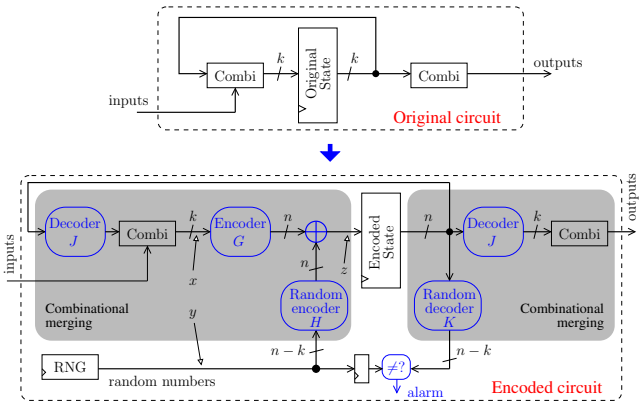

Fig. 1. Architecture of "Encoded Circuit", exemplified on a canonical Moore machine [10].

less than d_{Trigger} registers will not be effective. The parameter d_{Payload} is used to impede the HTH "Payload" part. It ensures that HTH, which tries to modify strictly less than d_{Payload} registers, will create a wrong codeword. These two parameters give us a larger possible choice for different applications hence reducing the overhead comparing with the previous method based on LCD [10]. Next, we propose a fully automated design flow to encode any digital circuit. The presented design flow uses the standard CMOS process and can be used by any hardware designers who have little knowledge of security. Furthermore we analyze the effectiveness of encoded circuits against physical attacks (side-channel and faults attacks). This analysis is performed on an encoded SIMON [4] crypto-processor implemented on an FPGA.

The rest of this paper is structured as follows. Sec. II gives the summary of "encoded circuit method" and the definition of the two new security parameters. Sec. III shows the construction of the new LCP codes. Sec. IV details the automated design flow used to encode ICs. Sec. V presents some case studies and also the improvement of the new LCP codes comparing with LCD used in [10]. Sec. VI evaluates the security of the method against other physical attacks. Finally we conclude in Sec. VII.

II. Security Parameters

A. Encoded Circuit Principle

The "encoded circuit" method, presented in [10] is based on the following observation: **all ICs are composed of 2 distinct parts: combinational and sequential part**. The sequential part, including data and control registers or flipflops, are easier to recognize on the IC layout and netlist because of their bigger size in comparison with combinational logic gates. Therefore, it is easier for an attacker to connect the HTH to the sequential part. Based on this observation, we propose to encode and mask all sequential registers with a LCD code [10]. The principle of this method is presented in the Fig. 1.

A linear Boolean code C (with generator matrix G) of size k and length n is used to encode an original state x of k bits. Then the encoded state is masked with the random number y, which serves as a pool of entropy, to obtain an encoded and masked state **z=xG** $\oplus$ **yH**. The mask y is also encoded by a code D (with generator matrix H) of size $(n-k)$ and length n. Any HTH, which probes the state z, will not be effective. In order to retrieve x and y from z, C and D must be supplementary. G, H, J and K are respectively the generating matrix of encoding and decoding of C and D. They allow to protect all sequential part (including control and data) of ICs against HTH. Then using the "flatten" option for synthesizing the encoded circuit, compiler tools will optimize/merge the combinational part of original circuit with the coding logic part thus obfuscating also the IC combinational part [10].

B. Proposed Security Parameters

In the "encoded circuit" method, we use the security parameter d which is *at the same time* the dual distance of D and the minimal distance of C. This parameter ensures that any HTH connected to strictly less than d bits of z will not be effective. We notice that C and $D = C^{\perp}$ are Quadratic Residue codes (QR).

In this article, we propose a generalization of LCD codes, namely LCP codes. A pair of codes C and D are LCP if they are complementary. We define the minimal distance of C and the dual distance of D as 2 distinct security parameters d_{Payload} and d_{Trigger}. Clearly, if C and D are LCP and dual, then they are LCD, with $d_{\text{Payload}} = d_{\text{Trigger}}$.

A characterization of the two parameters of LCP codes is the following:

- $d_{\textbf{Trigger}}$: insures that HTH, which probes $d_{\text{Trigger}} - 1$ (or less) bits of the encoded & masked state z, does not disclose any information on x.
- $d_{\textbf{Payload}}$: insures that HTH, which modifies $d_{\text{Payload}} - 1$ (or less) bits of the encoded & masked state z, cannot produce a valid codeword.

This is feasible, as stated in the following properties.

Property II.1. *The encoding of x as $z = xG \oplus yH$, where y is a uniformly distributed mask in $\mathbb{F}_2^{n-k}$ does not reveal any information on x provided up to $d_{Trigger} - 1$ bits of z are known, if and only if D is of dual distance $d_{Trigger}$.*

Property II.2. *Let us consider the encoding of x as $z = xG \oplus yH$, where y is a uniformly distributed mask in $\mathbb{F}_2^{n-k}$. Any fault on z of Hamming weight strictly smaller than $d_{Payload}$ can be detected, if and only if C is of minimal distance $d_{Payload}$.*

Now, how to determine the parameters d_{Trigger} and d_{Payload}? The rationale is the following: large HTH can be detected by various means (optical inspection of the chip, SCA, etc.). So, the minimal size of a HTH that would be difficult to identify is captured by distances d_{Trigger} and d_{Payload}. A stealthy HTH below those distances would have uncontrollable trigger and would certainly be captured red-handed when executing its payload.

III. Construction of LCP Codes

A. LCP Codes Properties

For the construction of LCP codes, we need to create two space vectors C and D (seen as linear codes) that are supplementary, i.e., $C \oplus D = \mathbb{F}_2^n$, with those additional constraints:

1) D must be of dual distance d_{Trigger};
2) C must be of minimal distance d_{Payload}.

In our application, C is used to encode original state of k bits therefore the dimension of C is k and the dimension of D is $n-k$. So, for a given k, we search for the smallest $n \geq k$ such that:

1) there exists a code D of parameters $[n, n-k]$ and of dual distance d_{Trigger}, i.e., there exists a code $C' = D^{\perp}$ of parameters $[n, k, d_{\text{Trigger}}]$,
2) there exists a code C of parameters $[n, k, d_{\text{Payload}}]$.

We write the generating matrix G of C in a systematic form $G = \left(\begin{array}{c|c} I_k & M \end{array} \right)$, where M is a $k \times n-k$ matrix. Similarly, we write the generating matrix H of D as $H = \left(\begin{array}{c|c} N & I_{n-k} \end{array} \right)$, where N is a $(n-k) \times k$ matrix.

Proposition 1. *The three following statements are equivalent:*

1) *The matrix* $\left(\begin{array}{c} G \\ \hline H \end{array} \right) = \left(\begin{array}{c|c} I_k & M \\ \hline N & I_{n-k} \end{array} \right)$ *is invertible;*
2) *The matrix* $I_k \oplus MN$ *is invertible.*
3) *The matrix* $I_{n-k} \oplus NM$ *is invertible.*

Corollary 1. *When it is invertible (see Proposition 1), the inverse of matrix* $\left(\begin{array}{c|c} I_k & M \\ \hline N & I_{n-k} \end{array} \right)$ *is given by:*

$$\left(\begin{array}{c|c} I_k & M \\ \hline N & I_{n-k} \end{array} \right)^{-1} = \left(\begin{array}{c|c} (I_k \oplus MN)^{-1} & M(I_{n-k} \oplus NM)^{-1} \\ \hline N(I_k \oplus MN)^{-1} & (I_{n-k} \oplus NM)^{-1} \end{array} \right).$$

Remark 1. *If* $N = M^{\mathsf{T}}$*, then* $GH^{\mathsf{T}} = 0$*, which is equivalent to say that each LCP code is a LCD code.*

B. Algorithmic Construction of LCP Codes

This section presents an LCP codes construction example for $d_{\text{Trigger}} > d_{\text{Payload}}$ based on the properties presented in Sec. III-A. The different steps of LCP construction are:

- Choose the shortest best known linear code (BKLC), termed C', of dimension k and minimal distance at least d_{Trigger}. This can be done by known constructions (see MAGMA [12]). *Nota bene*: in listing 1, C' is Cp.
- Call D the code $C'^{\perp}$, and complement D with k vectors e_i, for $1 \leq i \leq n$ where e_i are the canonical basis vectors of $\mathbb{F}_2^n$, so as to get the basis of a new code C.
- Generate matrix H of code D.
- Generate matrix G of C. This code C has the required parameters $[n, k, d_{\text{Payload}}]$.
- Generate the decoding matrices J and K.

In terms of matrices, this construction consists in writing the generating matrix of D as $H = \left(\begin{array}{c|c} N & I_{n-k} \end{array} \right)$, and in choosing $G = \left(\begin{array}{c|c} I_k & M \end{array} \right)$ for the generating matrix of C. When C and D form a LCP, $I_k \oplus MN$ and $I_{n-k} \oplus NM$ are invertible (by Proposition 1).

The Magma script (see listing. 1) shows the computation of C, D codes and also the generating matrix G, H, J and K for an initial state x of 37 bits, $d_{\text{Trigger}} = 13$ and $d_{\text{Payload}} = 10$.

```
k         := 37;  // Can be adapted
d_Trigger := 13;  // Can be adapted
d_Payload := 10;  // Can be adapted

Cp := BestLengthLinearCode(GF(2), k, d_Trigger);
n  := Length(Cp);
D  := StandardForm(Dual(Cp));
H  := GeneratorMatrix(D);
N  := ColumnSubmatrix(H, n-k+1, k);
H  := HorizontalJoin(N,IdentityMatrix(GF(2),n-k));
D  := LinearCode( H );

repeat
repeat M := RandomMatrix( GF(2), k, n-k );
until Rank( IdentityMatrix(GF(2), n-k) + N*M )
eq n-k; // Item 3) of Proposition 1.
G := HorizontalJoin( IdentityMatrix(GF(2), k), M);
C := LinearCode(G);
until MinimumWeight(C) ge d_Payload;

Dimension( C meet D ); // Must be zero
Dimension( C +    D ); // Must be equal to n

// Application of Corollary 1.
Inv1 := ( IdentityMatrix(GF(2),    k) + M*N )^-1;
Inv2 := ( IdentityMatrix(GF(2), n-k) + N*M )^-1;
J    := VerticalJoin( Inv1, N*Inv1 );
K    := VerticalJoin( M*Inv2, Inv2 );
VerticalJoin( G, H ) * HorizontalJoin( J, K )
eq IdentityMatrix( GF(2), n ); // Sanity check
```

Listing 1. Generation of matrices G, H, J, K

For the case where $d_{\text{Trigger}} < d_{\text{Payload}}$, we compute directly C code by choosing the shortest best known linear code (BKLC) $C' = C$ of dimension k and minimal distance at least d_{Payload}. Then we calculate D code with the parameters $[n, n-k, d_{\text{Trigger}}]$ using the supplementary of C.

IV. Automated Design Flow for Encoded Circuit

We briefly described the theory of encoded circuits in previous section. The method to encode a standard digital circuit is straightforward, which makes it easy to automate. The fully automated design flow for encoding a given hardware to protect against HTH insertion is shown in the Fig. 2. The flow can be divided into six distinct steps which are as follows:

a) Logic Synthesis: This step is native to any design flow. The user synthesizes a HDL description of the design with a synthesis script (in TCL), which constraints the tool to flatten the netlist (ex. "`ungroup -flatten -all`" in Encounter RC from Cadence). This steps ensures that we enter into the paradigm of the Moore machine such that all sequential elements are gathered into a global state. Next, we check that the design is coded in a way such that there is no logic from the clock and reset inputs till the flip-flops. Last, the synthesizer is constrained not to use flip-flops that "compute", e.g., flip-flops with an enable or two inputs (it is usual to find these gates in standard cell libraries, because they are dedicated to the *test* of the circuit). So, we use only non-test flip-flops which can be enforced by the "`set_attribute avoid true libcell libcell_location`" TCL constraint in

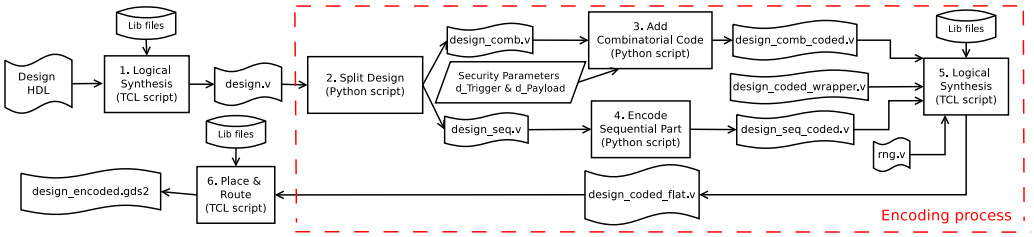

Fig. 2. Design flow for encoding method integration.

Cadence Encounter Compiler. Besides, we make sure the design has no latches in it. The synthesis exports the netlist as `design.v`.

b) Split Design: This is the first step of modified design flow: We identify and separate the sequential part of the design from the combinational part. For the sequential part, it is also important to keep the initial value at reset for each flip-flop. In our tools, we found it convenient to simply use regular expressions to identify the flip-flop gates. Namely, their name matches `HS65_LS_FDR*` (the DFF name in the target technology). The final wildcard comprises the various loads. The number of DFF is k, and their initial state is denoted as $x_0 \in \mathbb{F}_2^k$. For the combinational part of the circuit, it is sufficient to remove all the flip-flops, followed by addition of a `from_seq` input and a `to_seq` output bus (of bitwidth k). The automation is achieved with Python. This step generates two files: `design_comb.v` and `design_seq.v`.

c) Add Combinational Code: In this step, the user inputs the security parameters d_{Trigger} and d_{Payload}. Using the value k derived from the previous step, the script generates HDL code for matrices G, H, J and K for a suitable n. Next the file `design_comb.v` is connected with the HDL of matrices G, H, J and K as shown in Fig. 1. The connection between matrices and the combinational circuit is done automatically using a Python script. This step generates `design_comb_coded.v` at the output and the hierarchical structure of the file is kept intact at this stage. Also, at this stage, the Random Number Generator (RNG) which produces $(n-k)$ bits of random numbers at every clock period, is considered as a black-box.

d) Encode Sequential Part: The input of this step is `design_seq.v`. This step comprises of regeneration of with data input / output as a bus of bitwidth n, and programmed with encoded initial state x_0G at reset. In other words, k flip-flops in the uncoded state are replaced by n flip-flops in the encoded state, keeping the equivalent state at reset.

e) Synthesize Encoded Design: This step takes netlists `design_comb_coded.v`, `design_seq_coded.v` along with a RNG description `rng.v` and a wrapper circuit `design_coded_wrapper.v` as inputs. The function of the wrapper circuit is to connect the combinational and sequential part of the encoded circuit, while keeping the same interface as original design. All the files are fed to a logic synthesizer to generate a flattened netlist of the encoded design i.e. `design_coded_flat.v`.

f) Place & Route: The rest of the design flow is same as the standard design flow. In this step, the designer gives the synthesized netlist `design_coded_flat.v`. The design is then placed and routed to generate the final layout (GDSII).

V. Case Studies

A. 8 Bits Processor

In this section, we encode practical circuits using the proposed design flow and compare with the code used in [10]. In order to compare the new code with the once used in [10], we use the same target circuit, namely the nanoprocessor. This is a 8-bit processor without pipeline which requires 3 clock cycle to execute every instruction [13]. It has 16 basic instructions, and operates using an external 256 byte memory. The unprotected processor gives the following after synthesis 37 **sequential cells (flip-flops) and** 199 **combinational gates.** Then we applied 2 new LCP codes **[81,37,17,12]** and **[81,37,17,8]**, whose parameters are interpreted as:

- 81 is the length of codeword (n),
- 37 is the length of initial state (k),
- 17 is d_{Trigger},
- 12 and 8 are two values of d_{Payload}.

The results are presented in Tab. I. We can notice that with the new code, we can reduce the overhead from 9717 (old code [10]) to 7377/7324 μm^2 (new codes) with the same d_{Trigger} parameter. We also noticed that the area is reduced from 7377 to 7324 μm^2 by reducing the d_{Payload} from 12 to 8 in the new codes. So with a smaller d_{Payload}, we can reduce the overhead of encoded method. Another new code example ([49,37,5,3]) for the nanoprocessor is also proposed. The overhead of this code is $< 3\times$. It could be acceptable for certain applications.

B. SIMON Cryptographic Coprocessor

In our second study, we use a lightweight crypto-processor SIMON. This cipherblock use a 32-bits plaintext and 64-bits of key and compute 32-bit ciphertext after 32 rounds. The unprotected SIMON gives the following after synthesis:

- 109 sequential cells (flip-flops),
- 300 combinational gates.

Thus we have $k = 109$ for the original SIMON netlist.

It is interesting to compare "encoded circuits" (this paper) with "private circuits" [8]. Private circuits have an overhead quadratic with d_{Trigger}. For example, for $d_{\text{Trigger}} = 2$ (i.e.,

TABLE I
SYNTHESIS RESULTS OF ENCODED CIRCUIT METHOD, AND SECURITY PARAMETERS FOR THE NANOPROCESSOR.

IC (Code)	Gates	Area (μm^2)	n	k	d_{Trigger}	d_{Payload}
Original ([37,37,1,1]†)	199	1181	37	37	1	
Encoded ([86,42,17]) [Prev. work: LCD codes [10]]	1410	9717	86	37	17	
Encoded ([73,37,17,12]) [This work: LCP codes]	1159	7377	81	37	17	12
Encoded ([73,37,17,8]) [This work: LCP codes]	1151	7324	81	37	17	8
Encoded ([49,37,5,3]) [This work: LCP codes]	433	3137	49	37	5	3

†: Notice that the $[37, 37, 1, 1]$ code is the identity function (i.e., no encoding is used).

TABLE II
SYNTHESIS RESULTS OF ENCODED CIRCUIT METHOD, AND SECURITY PARAMETERS FOR THE SIMON CO-PROCESSOR.

IC (Code)	Gates	Area (μm^2)	n	k	d_{Trigger}	d_{Payload}
Original ([109,109,1])	300	1919	109	109	1	
Encoded ([110,109,2,1]) [This work: LCP codes]	560	3567	110	109	2	1
Encoded ([140,109,10,6]) [This work: LCP codes]	3107	20239	140	109	10	6
Encoded ([123,109,5,3]) [This work: LCP codes]	2348	15249	123	109	5	3

resistance to a single probe attack), the overhead is $39.7\times$ in area (results obtained on Virtex5 FPGA). For the sake of comparison, we coded SIMON with LCP codes such that $d_{\text{Trigger}} = 2$ and $d_{\text{Payload}} = 1$. Table II shows that the overhead is only $1.9\times$, for the same level of security, using LCP.

In addition, Tab. II also presents the results of two LCP codes [140,109,10,6] and [123,109,5,3] on SIMON. This application shows that the encoding method can be used for very different ICs. These encoded SIMON circuits are also implemented on the FPGA to evaluate their performance against physical attacks. The analyses against physical attacks are presented in the following section.

VI. SECURITY EVALUATION

In this section, we evaluate the encoded circuit method performance against other physical attacks as Probing Attack, Side-Channel Attack and Fault-Injection Attack.

A. Probing Attack

As encoded circuits are based on private circuits, they directly address the threat of **probing attack**. Probing attacks (front-side or back-side [7]) use tiny probes to monitor the inputs/outputs of internal blocks to directly recover sensitive data. By encoding all internal sequential logic parts and by masking encoded data with random numbers, this prevention method can also protect IC against probing attack with less than $d_{\text{Trigger}} - 1$ probes.

B. Side Channel Attack (SCA)

SCA extracts sensitive information from a circuit using the power consumption, electromagnetic leakage or delay analysis. We applied SCA on the encoded SIMON circuit

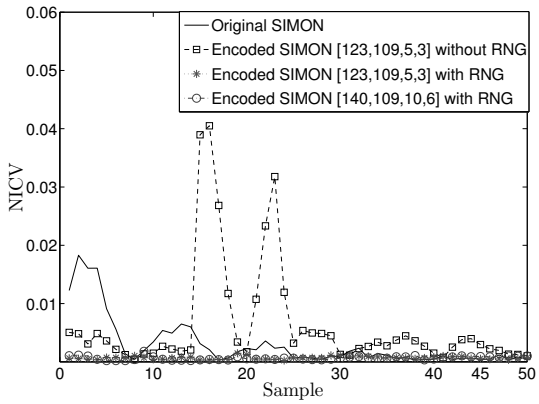

Fig. 3. NICV for encoded SIMON circuit

(as in Tab. I). The platform setup is a Sasebo GII FPGA Board (which contains a Virtex-5 FPGA) running at 24 MHz, $Langer\ RFU\ 5-2$ EM probe. For each design we acquired 200.000 traces with random plaintexts.

Next we use leakage detection techniques to check traces for any first-order leakage. We precisely compute the Normalized Inter Class Variance (NICV) with respect to the plaintext in the traces. NICV is computed as: $NICV = \frac{\mathsf{Var}[\mathbb{E}[T|X]]}{\mathsf{Var}[T]}$, where T denotes side-channel traces and X represent a chosen nibble of plaintext. The Fig. 3 shows four NICV computations for one key byte of four different designs: uncoded SIMON, encoded SIMON [123,109,5,3] with RNG deactivated, encoded SIMON [123,109,5,3] with RNG activated and encoded SIMON [140,109,10,6] with RNG activated. By observing the results, we notice the following conclusions. First of all, encoded circuit without RNG should not be used, as the side-channel leakage is amplified. This is because during encoding we transform the code linearly without confusion, thus any linear distinguisher can detect the leakage. Secondly, encoded circuit with RNG do reduce side-channel leakage to an extent. The gain in SCA resistance is bounded due to limited entropy. Finally, as we increase d_{Trigger} from 5 to 10, we increase the entropy and therefore reduce the side-channel leakage.

We also compute the inverse security gain i.e. the ratio between the NICV maximum value of encoded SIMON circuits with the one of original SIMON circuit. The inverse security gain w.r.t to original SIMON is 3.7 for encoded SIMON [123,109,5,3] without RNG, 0.0996 for encoded SIMON [123,109,5,3] with RNG and 0.085 for SIMON [123,109,10,6] with RNG. The results shows that the Signal Noise Ratio (SNR) reduces significantly with the encoded circuit and d_{Trigger} can be used as the security parameter of side-channel attacks.

C. Fault Injection Attack

As stated earlier, encoded circuits can also be used to detect **Fault Injection Attacks** (FIA). This can be done by decoding and verifying the random numbers injected to mask the encoded circuit. Precisely, decoding can be done using the

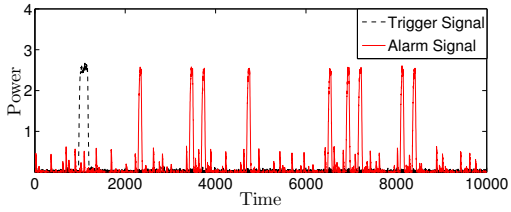

Fig. 4. Global fault injection for one SIMON cipher computation

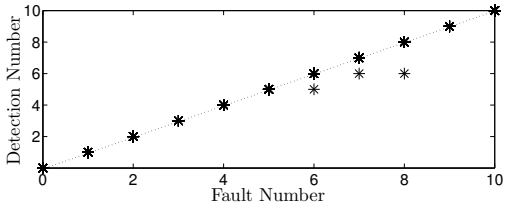

Fig. 5. EM FIA: Number of Detection = F(Number of Fault)

matrix K and the compared as shown in Fig. 1. If the input and output random differ, an "alarm" signal is raised and recovery mechanism like global reset is launched.

To evaluate this aspect of encoded circuit, we implemented the encoded SIMON [123,109,5,3] on the Sasebo-W FPGA board and UART for external communications. Then we perform global and local FIA on this board. The global FIA is done by varying the circuit frequency to inject the faults. The Fig. 4 presents the screenshot of "alarm" signal for one SIMON cipher computation. In this case, "alarm" signal is triggered whenever there is a fault. The results show that encoded circuit can detect the global FIA. Moreover, faults in individual rounds are detected separately.

For local FIA, we used electromagnetic injection (EMI). For the best evaluation of the encoded circuit against local FIA, we separate the location of each block on FPGA. The encoded SIMON is isolated from UART block and RNG block. Then we inject a single electromagnetic signal on the encoded SIMON. It insures that we will fault only encoded SIMON. We inject faults using an electromagnetic pulse of width of 1.5 ns. The EM pulse is injected from the beginning to the end of SIMON computation with a step of 1 ns. So in total, we perform 2560 steps. For each step, we perform 10 EMI. In each experiment, SIMON computes with the same plaintext and key. The Fig. 5 represents the number of detection in function versus the number of produced faults. We can notice that encoded SIMON can detect almost all produced faults. There are a few case, where the number of detection is smaller than the number of produced faults. This reduction is due to the setup modification between each FIA (setup instability). For 2560 delay steps EMI, there are statistically 2557 cases where the number of detection is equal to the number of faults. So these first experiments demonstrate that encoded circuit method can detect FIA even with a high-cost FIA technique (EMI FIA).

VII. Conclusions

In this paper, we propose Linear Complementary Pair (LCP) of codes to encode the IC hence preventing HTHs insertion. They are based on two quantifiable security metrics: d_{Trigger} and d_{Payload}. Here d_{Trigger} defines the minimum number of connections required to insert an effective HTH and d_{Payload} defines the minimum number of state that HTH needs to modify, to be (hopefully) undetected.

The paper presents the definition and the construction of the new LCP codes and also their application on 2 circuits: a nanoprocessor and a SIMON cryptographic co-processor. The results shows that we reduce the overhead of encoded circuit from $\approx 10\times$ to $\approx 7\times$ with the new LCP codes comparing with LCD codes [10]. By adjusting the two security parameters, we can further reduce the overhead. Several experiments are performed to demonstrate the effectiveness of encoded circuit method against side-channel attack and fault injection attacks. This method reduces significantly the leakage on side-channel and allows to detect reliably both global and local fault injection attack.

References

[1] Miron Abramovici and Paul Bradley. Integrated circuit security: new threats and solutions. In Frederick T. Sheldon, Greg Peterson, Axel W. Krings, Robert K. Abercrombie, and Ali Mili, editors, *CSIIRW*, page 55. ACM, 2009.

[2] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan Detection using IC Fingerprinting. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 296–310, Washington, DC, USA, 2007. IEEE Computer Society.

[3] M. Banga and M. S. Hsiao. ODETTE : A Non-Scan Design-for-Test Methodology for Trojan Detection in ICs. In *HOST, IEEE*, pages 18–23, 2011.

[4] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/2013/404.

[5] Gedare Bloom, Bhagirath Narahari, and Rahul Simha. OS Support for Detecting Trojan Circuit Attacks. In Mohammad Tehranipoor and Jim Plusquellic, editors, *HOST*, pages 100–103. IEEE Comp. Soc., 2009.

[6] R. S. Chakraborty and S. Bhunia. Security against hardware trojan through a novel application of design obfuscation. In *ICCAD, IEEE*, pages 113–116, 2009.

[7] Clemens Helfmeier, Dmitry Nedospasov, Christopher Tarnovsky, Jan Starbug Krissler, Christian Boit, and Jean-Pierre Seifert. Breaking and entering through the silicon. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 733–744. ACM, 2013.

[8] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, August 17–21 2003. Santa Barbara, CA, USA.

[9] Yier Jin, Nathan Kupp, and Yiorgos Makris. Experiences in hardware trojan design and implementation. In *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '09, pages 50–57, Washington, DC, USA, 2009. IEEE Computer Soc.

[10] Xuan Thuy Ngo, Sylvain Guilley, Shivam Bhasin, Jean-Luc Danger, and Zakaria Najm. Encoding the state of integrated circuits: A proactive and reactive protection against hardware trojans horses. WESS '14, pages 7:1–7:10, New York, NY, USA, 2014. ACM.

[11] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. EPIC: Ending Piracy of Integrated Circuits. In *DATE*, pages 1069–1074. IEEE, 2008.

[12] University of Sydney. Magma Computational Algebra System. http://magma.maths.usyd.edu.au/magma/, Accessed on 2014-08-22.

[13] M.J. Wirthlin, B.L. Hutchings, and K.L. Gilson. The Nano Processor: a low resource reconfigurable processor. In *IEEE Workshop on FPGAs for Custom Computing Machines, 1994*, pages 23–30, Apr 1994.