Area Optimization of Cryptographic Co-Processors Implemented in Dual-Rail with Precharge Positive Logic

Sylvain Guilley, Laurent Sauvage, Jean-Luc Danger, Philippe Hoogvorst

AREA OPTIMIZATION OF CRYPTOGRAPHIC CO-PROCESSORS IMPLEMENTED IN DUAL-RAIL WITH PRECHARGE POSITIVE LOGIC

Sylvain Guilley, Laurent Sauvage, Jean-Luc Danger and Philippe Hoogvorst

Institut TELECOM / TELECOM ParisTech & CNRS LTCI (UMR 5141),
Département COMELEC, 46 rue Barrault,
75 634 PARIS Cedex 13, FRANCE.
Email: < sylvain.guilley@telecom-paristech.fr >

ABSTRACT

Field programmable gate arrays (FPGAs) become very popular for embedded cryptographic operations. In order to resist side-channel attacks, FPGAs must implement reasoned countermeasures. The most efficient way to mitigate attacks is to adopt a gate-level protection. Two secure gates families exist: those that "hide" and those that "mask" side-channel leakage.

In this article, we detail methods to reduce the size of wave dynamic differential logic (WDDL) implementations. These circuits are designed to hide any physical leak by ensuring a data-independent activity. This study is meant to be generic, and thus applies to any $4 \rightarrow 1$ LUT-based FPGAs. Further optimizations can be reached by taking advantage of some FPGAs proprietary features. Our solutions include RTL code modification, synthesizer usage (potentially in a re-entrant way), and *ad hoc* mapping. For the first time, we point out how sequential parts (*e.g.* registers) of the design can participate to the overall area savings. Also, we show that linear parts of algorithms can be delegated to a synthesizer, but that non-linear parts are better off to be handled with heuristics. We present a 23 % area gain over the state-of-the-art as for the positive WDDL triple-DES symmetric encryption algorithm.

Keywords: FPGA security, cryptographic applications, side-channel attacks mitigation, power-constant logic, positive dual-rail with precharge logic, synthesis optimization.

1. INTRODUCTION

High-end markets often resort to FPGAs because these devices are affordable for small to medium volumes. The applications targeted by these markets require that data are handled with confidentiality. Cryptographic techniques are therefore used to protect the device internal data. However, side-channels [1] attacks are known to and have proved concretely to wreak havoc with any cryptographic scheme. Cryptographic applications thus need to be carefully protected against any kind of sneak attack that exploit whatsoever physical leak. Two families of protections have been put forward in the literature.

Masked logics resort to an external random number generator to mislead any statistical attack. They have been widely studied [2, 3]. However, those "masking" countermeasures have many weaknesses: the random source must be unbiased, second order attacks can be devised and the presence of glitches [4] and the early evaluation [5] still remain a difficult problem to solve.

Hiding logics, such as wave dynamic differential logic (WDDL, [6]), have been seldom studied in FPGAs. Some results on combinatorial functions are reported in [7]. Area performances for a complete algorithm are detailed in [8] although little is said about synthesis optimization.

The goal of this article is to shed a new light on WDDL netlists optimizations by using an improved synthesis flow.

The rest of the paper is structured as follows. The projection of a design into positive WDDL logic is detailed in Sec. 2. The principle of area optimizations is also discussed in this section. We then apply optimizations to the whole DES [9] in Sec. 3. Then, we show in Sec. 4 that non-linear parts of the algorithms can be compacted using heuristics. Finally, the section 5 concludes on the area gain (namely -23 %) obtained with the techniques exposed in the paper.

2. METHODOLOGY

In the article [10], Gaël Rouvroy *et al.* propose methods to compact specifically a DES or a triple-DES hardware description. The authors use manual re-clustering and some peculiarities of the target FPGA (a Xilinx Virtex-II) to achieve area optimizations. For instance, the SSR (usage of LUT memory to store some ROM or some shift registers) functionality, proprietary of Xilinx, is taken advantage of in order to implement the bit shifts needed for the key schedule routine.

Instead, we intend to provide device-agnostic methods

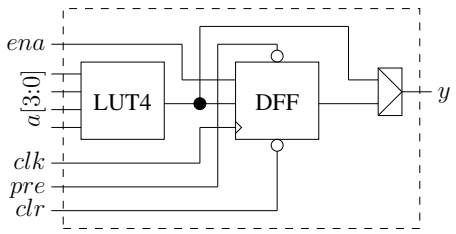

Fig. 1. Minimal structure used in our vendor neutral flow, corresponding to the common denominator of usual FPGAs.

to compact portable design flows. Our flow is suitable for Xilinx, Altera, Lattice and Actel products. The basic structure we assume is that depicted in Fig. 1. It is available for all the FPGAs sold by the four major vendors listed above.

2.1. Related Works

Power-constant logics are secure only provided the backend is also balanced. This has been emphasized in many previous ASIC experiments, such as asynchronous [11] or synchronous [12, 13] logic. However, without routing constraints, place-and-route algorithms (based on simulated annealing and/or `A*`) will treat dual gates and dual wires in the same fashion. Indeed, for each direct gate, one dual gate exists. And for each direct wire, one dual wire also exists. Therefore, statistical algorithm are not expected to favor one gate or one net of a couple for another one. This explains why, at first order, the backend is naturally balanced. Notice however, that at second order, dedicated place and route methods must be used (if available). Such a method has been presented for the first time by Pengyuan Yu and Patrick Schaumont in [14]. The contribution of this paper is to present a design-independent method to obtain secure implementations in FPGA from both logical and physical points of view.

In [15] a method to implement DES in masked logic is described. The bottleneck of secured implementations of symmetric ciphers is the overhead of the transformed substitution box (sbox). It is suggested in [15] to use memories in this respect. Using memories is a wise solution, since these precious resources are otherwise wasted. Furthermore, it enables for sboxes customization without recompiling the design. The characteristics (*e.g.* performances, area footprint) remain the same whatever the sbox. However, using memories forbids to secure the backend, since the pathes to the true and false halves of the memory are unbalanced.

2.2. Synthesis Flow To Get a Positive Differential Netlist

To achieve differential and glitch-free netlists, only monotonic (*e.g.* positive) cells must be used [8]. As FPGA CAD tool cannot be constrained to use only some logical functions, they are not suitable to synthesize secure designs. Instead, ASIC synthesizers are used. These tools require all the same the presence of an inverter. To indicate that it can be freely used, we reduce its to zero. Therefore, the synthesizer does not feel any penalty when using it. This is much desired, since in dual-rail netlists, the inversion of a couple of wires (a_t, a_f) is simply a wire-crossing $\tilde{}(a_t, a_f) = (a_f, a_t)$. Other gates are all equivalent in terms of cost: they are attributed the same (*e.g.* unitary) area. We studied heterogeneous strategies, that turned out to be worse than the uniform area cost. Notably, the combinatorial alone, the "data flip-flop" (DFF) alone, and the combined combinatorial+DFF have the same area, because this is a reality in FPGAs. The generation of a positive library is implemented by `genlut` [16], and the Verilog (IEEE standard # 1364) netlist duplication by `vDuplicate` [17].

2.3. Area Optimization Techniques

We use in this paper several area optimization techniques:

- Source code modification (RTL description style). Indeed, synthesizers are sensitive to the coding style.
- Identify static signals (*e.g.* registers enables), and simplify them (enables are free in FPGAs). This optimization is indeed device-agnostic.
- Pack the registers with orphan logic gates, that were inferred as stand-alone gates by the ASIC synthesizer.

We separate the treatment of linear and non-linear parts of the DES design. In the rest of the article, we take the example of the architecture described in Fig. 6 of [18].

3. DES LINEAR PARTS

3.1. Combinatorial Parts

The ASIC synthesizer Cadence `bgx_shell` infers positive multiplexers as 2×2 LUT4 gates. Using the Cadence `rc` synthesizer, two-input multiplexers are properly inferred as 2 LUT4 gates (refer to Eqn (1)). The other awkward syntheses of `bgx_shell` are studied in Sec. 4 as for non-linear DES parts.

3.2. Sequential Parts

FPGA design is very different from ASIC design, in the sense that DFFs can be considered as free resources in FPGAs. Indeed, every elementary computation grain (cf. Fig. 1) contains one flip-flop. This means that DFFs are much often underutilized in FPGAs.

For this reason, it is tempting to replace single-ended every DFF (either initialized to '0' or '1') by a cluster depicted

in Fig. 2. These structures are indeed suitable for WDDL, because they are differential and have two stages, to memorize simultaneously the precharge token and the actual dual-rail data.

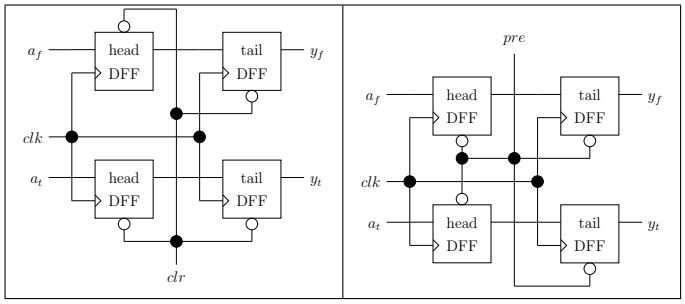

Fig. 2. Structure of the negatively or positively asynchronously initialized WDDL register cells.

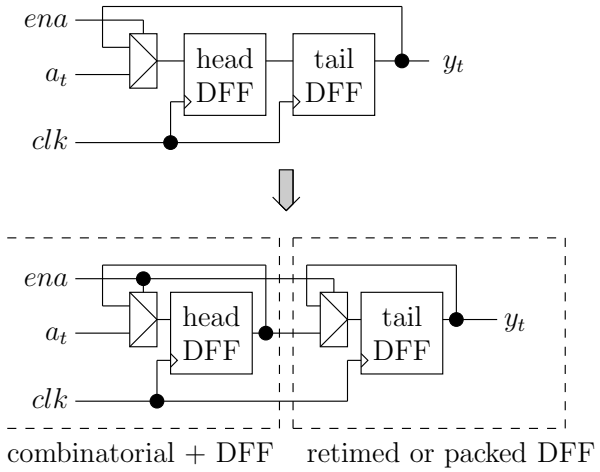

Fig. 3. (Half of a) WDDL register with enable (see Fig. 2), compact in FPGAs complying with the archetype of Fig. 1.

However, this choice is misleading when the DFFs are enabled (*e.g.* to save energy when the module is not working): the built-in enable (*cf* Fig. 1) cannot be used, and an external one is thus required, as shown on top of Fig. 3. It is wiser to have in the ASIC library of cells sequential elements that are "enabled". The adequate description for a compact "DFF with enable" is depicted in bottom of Fig. 3.

The contents of the ASIC library of cells that allow to take advantage of the enable signal present in FPGAs sequential elements is given in Tab. 1. Notice that the description of a pure combinatorial function (item 3) and of a mixed combinatorial+DFF cell (item 4) differs because timing arcs are different. The table 2 provides snippets for the pure AND and the AND+DFF cells, expressed in LIBERTY [20].

Indeed, this strategy allows to save $1,104$ LUT4s:

Table 1. ASIC library contents for +ve WDDL synthesis.

1	Invertor, mandatory for the ASIC synthesizer.
2	Two-input AND gate (also required by ASIC synthesizers for it to start its job).
3	The 166 [19], non-trivial WDDL positive pure combinatorial functions.
4	DFF with the 166 upstream positive combinatorial function.
5	DFF with enable, without upstream combinatorial function. This element is used for the tail register in Fig. 3. It will be optimized by packing with an incomplete purely combinatorial cell (with at least one open input), of by retiming. In the later case, the DFF is automatically swapped with downstream combinatorial logic. This combinatorial logic and the DFF can thus be merged to occupy a single logic element.

- $(64 + 64 + 56) \times 4$ registers (namely LR, CD and IF, see [18]), that are packed with existing logic,
- $(64+64+56)\times 2$ multiplexors (muxs), that are turned into enabled registers (*cf.* Fig. 3)

4. DES NON-LINEAR PARTS

4.1. Synthesis with Legacy ASIC Tools

In [8, § IV], the synthesis in LUT4 positive logic of DES sboxes ($6 \rightarrow 4$ random tables) is shown to always be larger than a manual mapping[1]. We present a technique to reach sometimes more compact DES sboxes using legacy ASIC synthesizers. It consists in:

- using a coarse-level synthesizer, that breaks the design into non-positive gates, and then in
- using a fine-level synthesizer, that remaps all the non-positive instances into a sub-netlist of positive gates. This last step involves an automatic re-clustering to factor positive gates of adjacent sub-netlists that do not use all their inputs.

This re-entrant synthesis technique is illustrated in Fig. 4 for 4 inputs primary synthesis, using `quartus` as a coarse synthesizer.

As for DES sboxes, we have experienced the performance of two synthesizers (for the re-entrant synthesis), namely `bgx_shell` and `rc` from Cadence, tuned to spend the maximal effort on the area optimization. Additionally, we tested four HDL styles:

[1] This mapping is recalled in Sec. 4.2 at page 4.

```
/** In combinatorial instances, the function is expressed in the "output pin" section */
pin( y ) { direction: output; function: "a0*a1";
    timing() { related_pin: "a1 a0"; intrinsic_rise: 1; intrinsic_fall: 1; }
}
```

```
/** In sequential instances, the function is expressed in the "internal state" section */
ff( iq, iqn )
{
    clocked_on : "clk";
    next_state : "ena*(a0*a1)+ena'*iq";
    clear      : "clr'";
    preset     : "pre'";
}
pin( y ) { direction: output ; capacitance : 0 ; function : "iq";
    timing() { related_pin: "clk"; timing_type: rising_edge; intrinsic_rise: 1; intrinsic_fall: 1; }
}
```

Table 2. Expression of pure combinatorial (*top*) and of mixed combinatorial+sequential (*bottom*) logic in LIBERTY format.

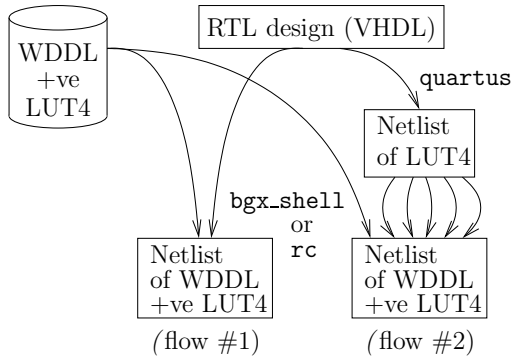

Fig. 4. Alternative synthesis flow (on the right) to end up with a dual-rail WDDL positive netlist of LUT4.

S: NIST original description,

T: The two-bit line decoder is put forward, whereas the four $4 \rightarrow 4$ bijections are left tabulated,

U: `assign` statement in a linearly decoded way,

V: `case` statement in a linearly decoded way.

The synthesis results are shown in Tab. 3, respectively for 4, 3, 2 primary synthesis level. The figures in boldface are better than the state-of-the-art (130 positive LUT4 reported in [8] and in Sec. 4.2). We observe that the HDL description style (either **U**, **V**, **S** or **T**) and the synthesizers choice both impact the quality of the netlist. However, no breakthrough is obtained: the best syntheses end up with 128 gates instead of 130. This is why we continue with a heuristic synthesis methods, that happens to be very efficient.

4.2. Heuristic

A LUT being structured as a tree of 2-input multiplexers, a simple way to get positive functions is to use a tree of "positive" multiplexers in differential logic. A positive multiplexer is such that the selection variable sel is composed of the couple (sel_t, sel_f) where the index$_t$, as "true", indicates the non inverted signal and the index$_f$, as "false", indicates the inverted signal. These two signals are reset at zero during the precharge phase of the differential logic as explained in [21]. The positive WDDL multiplexer equation is then:

$$s_t = x_0 \cdot sel_f + x_1 \cdot sel_t, \tag{1}$$

where x_0 and x_1 are the 2-input of the multiplexer. This positive multiplexer can be easily achieved by a cell composed of a 4-input LUT. Hence one output of a positive sbox is made of positive multiplexers would require 126 LUT4 (63 for the "true" output and 63 for the "false" output).

The heuristic consists in simplifying the first two columns driven by the two LSBs variables a and b associated with their dual rail signals (a_t, a_f) and (b_t, b_f). Actually only sixteen positive multiplexers are enough to generate all the functions of two positive variables. Hence the way to feed the third column inputs driven by c consists in a specific routing from these sixteen functions. Moreover there are six trivial functions which need not to be generated, because they are trivial and can therefore be implemented with pure routing: 0, 1, a_t, a_f, b_t, b_f. This preprocessor, hereafter called "global function generator", needs only ten LUT4 cells and is global to the four sbox outputs. Hence the total number of cells for a complete sbox is: $10 + (8 + 4 + 2 + 1) \times 2 \times 4 = 130$. The originality of this heuristic is to share cells between the true and false parts. This helps save area, without compromising the security. If we note T the set of non-trivial two-input gates, then $\forall f \in T, \overline{f} \in T$, where $\overline{f}$ is the complementary of f. Therefore, the constant activity

Table 3. Synthesis with `bgx(_shell)` and `rc` of the DES sboxes in LUT{4,3,2} positive logic.

DES substitution boxes described as **S**, in LUT4								
bgx	148	134	138	130	132	146	134	138
rc	160	146	152	156	156	154	156	152

DES substitution boxes described as **T**, in LUT4								
bgx	140	132	146	136	130	140	130	138
rc	132	132	**128**	130	136	134	134	132

DES substitution boxes described as **U**, in LUT4								
bgx	140	132	138	132	138	158	134	136
rc	158	156	158	154	160	160	152	154

DES substitution boxes described as **V**, in LUT4								
bgx	140	132	138	130	138	158	134	138
rc	158	156	158	154	160	160	152	154

DES substitution boxes described as **S**, in LUT3								
bgx	148	134	138	130	132	146	134	138
rc	160	148	152	158	154	152	152	158

DES substitution boxes described as **T**, in LUT3								
bgx	140	132	146	136	130	140	130	138
rc	136	132	**128**	**128**	136	134	138	136

DES substitution boxes described as **U**, in LUT3								
bgx	140	132	138	132	138	158	134	136
rc	156	156	164	144	160	162	152	158

DES substitution boxes described as **V**, in LUT3								
bgx	140	132	138	130	138	158	134	138
rc	156	156	164	144	160	162	152	158

DES substitution boxes described as **S**, in LUT2								
bgx	148	134	138	130	132	146	134	138
rc	154	150	152	152	156	154	150	158

DES substitution boxes described as **T**, in LUT2								
bgx	140	132	146	136	130	140	130	138
rc	134	132	**128**	**128**	138	136	134	136

DES substitution boxes described as **U**, in LUT2								
bgx	140	132	138	132	138	158	134	136
rc	154	156	156	144	158	162	150	160

DES substitution boxes described as **V**, in LUT2								
bgx	140	132	138	130	138	158	134	138
rc	154	156	156	144	158	162	150	160

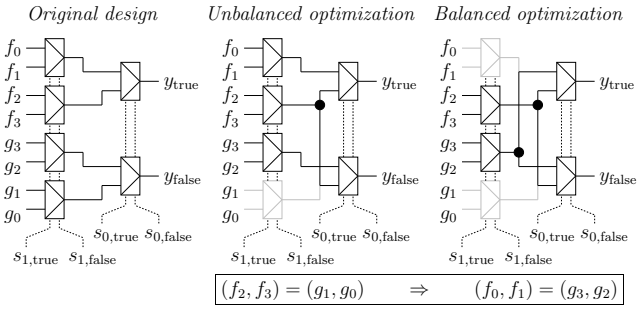

Fig. 5. Optimization B incorrectly (since partially) and correctly implemented on a redundant design.

condition is met.

4.3. Specific Sboxes Optimizations

The tree of cells which is driven by the four MSB variables $cdef$ receives sixteen functions x_i routed specifically from the global function generator. This MSB tree can be optimized by considering the four differential sbox outputs together. These optimizations are implemented in the software `fpgasbox` [22].

4.3.1. Optimizations A & B

The first optimization (A) consists merely in removing the cell of column c if $x_{2i} = x_{2i+1}$. In this case the LUT which is a positive multiplexer can be removed and leave place to the x_{2i} net. The second optimization (B) finds identical cells in column c. Two cells are said to be identical if they have the same inputs. This can be formalized by: $optimize$ = $(x_{2i} = x_{2j})$ and $(x_{2i+1} = x_{2j+1})$. In case of identity, a factorization can be done and one cell is removed.

4.3.2. Security Analysis of the Optimizations

Once again, we notice that the optimizations do not jeopardize the constant-activity condition mandatory for WDDL to be secure. Indeed, if the same Boolean variable f is computed in both the true and the false netlists halves, then the dual g of f exists in the both networks. Therefore, the optimizations consists in transforming $2 \times (f, g)$ (which is obviously functionally redundant) into $1 \times (f, g)$. The last couple remains constant in activity, and so do not disclose their individual value. As an example, the balancedness of optimization B is illustrated in Fig. 5. In this figure, $\forall i \in [0, 3]$, g_i is the dual of f_i. We assume that $(f_2, f_3) = (g_1, g_0)$. By duality, $f_0(x) = \overline{g_0(\overline{x})} = \overline{f_3(\overline{x})} = g_3(x)$, which means that $(f_0, f_1) = (g_3, g_2)$. Hence a pairwise optimization, that does not unbalance the dual networks.

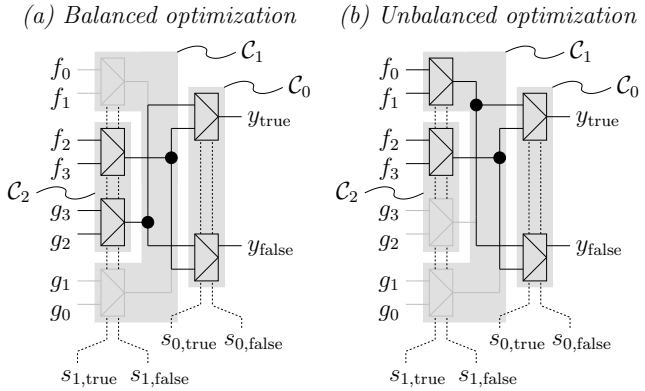

Fig. 6. Coupling of dual multiplexors conservation (a) or destruction (b) by the proposed optimization algorithm.

In order to further improve the security of the netlists, it is desirable to respect at the layout level the symmetry at the netlist level [23]. This requires to form couples between dual instances. We attract the reader's attention to the danger of forming couples before the optimization. Indeed, the optimization consists in removing two multiplexors by the same token, as explained in Fig. 5. However, depending on the optimization implementation (descent in the graph representing the netlist), the two multiplexors can be removed from two different couples. This is illustrated in Fig. 6:

(a) the two related muxs are removed from the same couple $\mathcal{C}_1$, which leaves a secure set of couples,

(b) the two related muxs are removed from two different couples ($\mathcal{C}_1$ and $\mathcal{C}_2$), leave the dual muxs in two different couples, which does not guarantee that they will be placed and routed in a balanced way.

To avoid this pitfall, we therefore suggest to form the couples after the optimization. Afterwards, we recognize multiplexors to place in the same couple by the analysis of the function they compute.

4.3.3. Results

Optimizations are done by permuting the inputs value of each sbox in order to find the optimum. Table 4 indicates the overall gain for every sbox of DES. A total of 292 LUT4s can be saved. The functionality of the netlists has been proved by their actual programmation in a Stratix EP1S25.

5. CONCLUSIONS

The design of cryptographic applications in FPGA require the use of secure logics. We have illustrated how to implement an area-efficient hiding logic style, called positive WDDL. We provide for a LUT usage gain of $1,104$ LUT4s as for linear (registers, cf Sec. 3) logic and 292 LUT4s as for non-linear logic (sboxes only, cf Sec. 4). The overall area saving is equal to: $\frac{6{,}038-1{,}104-292}{6{,}038} = \frac{4{,}642}{6{,}038} = 77\ \%$. This means that a 23 % surface optimization is made possible by packing registers and optimizing sboxes synthesis.

Table 4. Optimization A & B gain for the DES sboxes considered separately, expressed in differential LUT4 positive logic.

	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**Total**
Before	130	130	130	130	130	130	130	130	1,040
After	102	98	98	64	106	98	96	86	748
Gain	28	32	32	66	24	32	34	44	292

6. REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis: Leaking Secrets," in *CRYPTO*, ser. LNCS, vol. 1666, 1999, pp. 388–397.

[2] T. Popp and S. Mangard, "Masked Dual Rail Pre-Charge Logic: DPA Resistance without Routing Constraints," in *CHES*, ser. LNCS, vol. 3659. Springer, September 2005, pp. 172–186.

[3] T. Popp, T. Zefferer, and S. Mangard, "Evaluation of the Masked Logic Style MDPL on a Prototype Chip," in *Proceedings of CHES'07*, ser. LNCS, vol. 4727. Springer, September 2007, pp. 81–94.

[4] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," in *CHES*, ser. LNCS, vol. 3659. Springer, 2005, pp. 157–171, August 29th – September 1st, Edinburgh, Scotland, UK.

[5] D. Suzuki and M. Saeki, "Security Evaluation of DPA Countermeasures Using Dual-Rail Precharge Logic Style," in *Proceedings of CHES*, ser. LNCS, vol. 4249. Springer, October 2006, pp. 131–138.

[6] K. Tiri and I. Verbauwhede, "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation," in *Proceedings of DATE'04*, February 2004, pp. 246–251, Paris, France.

[7] ——, "Synthesis of Secure FPGA Implementations," in *International Workshop on Logic and Synthesis (IWLS)*, june 2004, pp. 224–231, Temecula, California, USA.

[8] S. Guilley, L. Sauvage, J.-L. Danger, T. Graba, and Y. Mathieu, "Evaluation of Power-Constant Dual-Rail Logic as a Protection of Cryptographic Applications in FPGAs," in *SSIRI*, Yokohama, Japan, jul 2008, pp. 16–23, http://hal.archives-ouvertes.fr/hal-00259153/en/.

[9] NIST/ITL/CSD, "Data Encryption Standard. FIPS PUB 46-3," 1999.

[10] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple DES," in *FPL*, ser. LNCS, vol. 2778. Springer, September 2003, pp. 181–193.

[11] G. Bouesse, M. Renaudin, B. Robisson, E. Beigné, P.-Y. Liardet, S. Prevosto, and J. Sonzogni, "DPA on Quasi Delay Insensitive Asynchronous Circuits: Concrete Results," in *DCIS*, 24–26 Nov 2004.

[12] K. Tiri and I. Verbauwhede, "Place and Route for Secure Standard Cell Design," in *CARDIS'04*, August 2004, pp. 143–158.

[13] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu and Renaud Pacalet, "The "Backend Duplication" Method," in *CHES*, ser. LNCS, vol. 3659. Springer, 2005, pp. 383–397, Edinburgh, Scotland, UK.

[14] P. Yu and P. Schaumont, "Secure FPGA circuits using controlled placement and routing," in *CODES+ISSS'07: Proc. of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. New York, NY, USA: ACM, 2007, pp. 45–50.

[15] F.-X. Standaert, G. Rouvroy, and J.-J. Quisquater, "FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks," in *FPL*, August 2006, Madrid, Spain.

[16] S. Guilley and A. D. Dao, "Software '`genlut`'," apr 2008, identification by the "Agence pour la Protection des Programmes" (APP): IDDN.FR.001.160027.000.S.P.2008.000.20600, http://www.tsi.enst.fr/publications/enst/misc-2008-8414.pdf.

[17] ——, "Software '`vDuplicate`'," apr 2008, identification by the "Agence pour la Protection des Programmes" (APP): IDDN.FR.001.160028.000.S.P.2008.000.20600, http://www.tsi.enst.fr/publications/enst/misc-2008-8415.pdf.

[18] S. Guilley, P. Hoogvorst, and R. Pacalet, "A Fast Pipelined Multi-Mode DES Architecture Operating in IP Representation," *Integration, The VLSI Journal (Elsevier)*, vol. 40, no. 4, pp. 479–489, July 2007.

[19] N. J. A. Sloane, "Dedekind numbers: number of monotone Boolean functions of n variables or number of antichains of subsets of an n-set," http://www.research.att.com/~njas/sequences/A000372.

[20] Synopsys, "`Liberty` Vol. 1 & 2," dec 2003, "`liberty.pdf`", available from Synopys "tap-in" program website: http://www.synopsys.com/partners/tapin/.

[21] K. Tiri and I. Verbauwhede, "Secure Logic Synthesis," in *FPL*, ser. LNCS, no. 3203, aug 2004, pp. 1052–1056, Antwerpen, Belgium.

[22] P. Hoogvorst, S. Guilley, and T. Graba, "Software '`fpgasbox`'," jul 2008, identification by the "Agence pour la Protection des Programmes" (APP): IDDN.FR.001.280019.000.S.P.2008.000.20600, http://www.tsi.enst.fr/publications/enst/misc-2008-8416.pdf.

[23] S. Guilley, L. Sauvage, T. Graba, J.-L. Danger, P. Hoogvorst, V.-N. Vong, and M. Nassar, "Place-and-Route Impact on the Security of DPL Designs in FPGAs," in *HOST (IEEE Computer Society)*, Anaheim, CA, USA, june 2008, pp. 26–32, DOI: 10.1109/HST.2008.4559042, ISBN: 978-1-4244-2401-6.