

The “Backend Duplication” Method

A Leakage-Proof Place-and-Route Strategy for ASICs

Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet

GET/Télécom Paris, CNRS LTCI
Département communication et électronique
46 rue Barrault, 75634 Paris Cedex 13, France.
{guilley, hoogvorst, mathieu, pacalet}@enst.fr

Abstract. Several types of logic gates suitable for leakage-proof computations have been put forward [1–4]. This paper describes a method, called “backend duplication” to assemble secured gates into leakage-proof cryptoprocessors. To the authors’ knowledge, this article is the first CAD-oriented publication to address all the aspects involved in the backend design of secured hardware. The “backend duplication” method achieves the place-and-route of differential netlists. It allows for 100 % placement density and for balanced routing of dual-rail signals. Wires of every other metal layer are free to make turns. In addition, the method does not require any modification to the design rules passed to the router. The “backend duplication” method has been implemented in 0.13 μm ASIC technology and successfully tested on various ciphers. The example of the design of a DES module resistant against side-channel attacks is described into details.

Keywords: Information leakage, secured backend, differential signals.

1 Introduction: Using Differential Logic to Thwart SCA

It has been shown that sensitive information can be extracted from cryptographic hardware either by spying physical quantities or by injecting faults. The first type of attack is often referred to as “side-channel attack” (SCA [5–7]), whereas the second one is also known as “fault attack” (FA). Two classes of countermeasures against SCA have been put forward. The first idea is to shield the hardware at the algorithmic level: the data manipulated by the cryptoprocessor is masked or protected by secret-sharing methods. The second idea is to build the hardware using only leakage-proof gates, so as make sure that the overall cryptoprocessor is, in turn, leakage-proof.

This article focuses on the implementation of the latter class of countermeasures. Many leakage-proof logic styles have been published. The level of protection the secured gates provide depends upon their specification:

1. SABL [1] is a logic consuming a nearly constant current.

2. WDDL [2] uses dual gates pairs to ensure a constant activity, although the power consumed by each gate of the pair is not the same.
3. Speed-independent (SI) logic presented in [3] features a consumption independent on the input data configuration. It also shields against the leakage of the signal transitions timing by synchronizing the inputs.
4. Refinements [4] of the previous solution also ensure that parasitic capacitances are unconditionally unloaded between two computations.

Some of those methods, for instance methods 3 and 4 above (nicknamed “SI-WDDL” in the rest of this article) can also embed an error-detection feature. The mechanism, based on an alarm propagation, is explained in [3]. Nevertheless, resistance to faults injection is not covered in this paper.

The *logical* part (coding, functionality verification, refinements for synthesis) in a design targeting FPGA or ASIC implementation is called *frontend*. The *physical* part (mainly consisting in place-and-route, but extensive description is provided in Sect. 2) is called *backend*. The common point to the secure gates listed above is the use of differential logic with a 4-phase protocol, such as “return to zero” (RTZ) or any variation [8]. It has already been stressed that the security of individual gates can extend to a netlist of gates only provided that the interconnect is kept differential [9]. Nonetheless, most articles evade the question of the implementation of a secure backend design.

Given the complexity of backend flows in sub-micron technologies, a simple way to realize the secure backend is necessary. We provide in this article a method, called “backend duplication”, that integrates the secure place-and-route into any preexisting backend flow without modifying the design rules.

The rest of the article is organized as follows: the “backend duplication” is presented in Sect. 2. The method is applied to some secured gates primitives in Sect. 3. A case study, namely a DES cryptoprocessor, is provided in Sect. 4. This example was actually fabricated in HCMOS9GP 0.13 μm technology from STMicroelectronics using the method presented in this paper. This section contains an evaluation of the cost and of the security increase provided by the use of the “backend duplication”. Finally, Sect. 5 concludes the article.

2 The “Backend Duplication” Method

2.1 Regular “Place-and-Route” ASIC Design Flow

In a standard cell flow, cryptographic functions are synthesized into a netlist of primitive gates. Then, the gates are placed into rows (see Fig. 1(a)). In each row, the gates are abutted, so that they share the ground (VSS) and the power (VDD) lines. When two gates are not placed side by side, a “filler” cell can be added in-between to ensure the continuity of the supply lines. In sub-micron technologies, there are enough levels of metal to allow the routing of the interconnect over the standard cell rows. For this reason, the rows are themselves abutted. Thus, the supply lines are shared between adjacent rows. This is achieved by flipping upside-down every other row: the ground (resp. the power) of one row

is merged with the ground (resp. the power) of the lower (resp. the upper) row, (see Fig. 1(b)).

Sub-micron technologies allow for 45 degree wire routes, but this feature is not yet implemented in commercial routers: currently, the routing is still Manhattan. Moreover, the most popular routers are also grid-based. Metal wires are only instantiated along a virtual routing tracks superimposed on the floorplan (see Fig. 1(c)). It is thus customary to attribute a preferred direction to every routing layer. However, routers consider the preferred direction only as a recommendation. The convention we use in this paper is that odd metal levels (metal 1, metal 3, and so forth) are preferentially routed vertically, whereas even metal levels are preferentially routed horizontally.

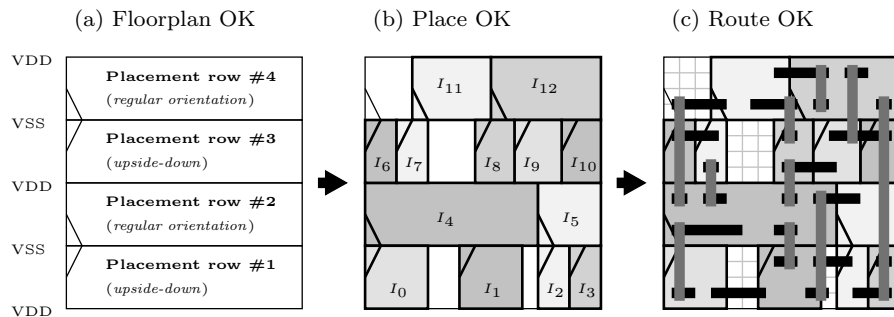


Fig. 1. Illustration of the regular (and insecure) “Place-and-Route” ASIC design flow.

2.2 The “Backend Duplication” Method Overview

The “backend duplication” addresses the strengthening against SCA of sensitive ASICs (smartcards, hardwired cryptoprocessors, etc.) It consists in a single manipulation of the backend layout to ensure the security of its interconnect. However, this method shall not be confused with the tailored duplication method for software or dedicated hardware implementations [10].

The basic idea of the “backend duplication” method is to apply a regular backend flow on a single-ended (as opposed to duplicated) netlist, taking care to leave enough room on the floorplan for the duplication of the placed-and-routed netlist. The duplication basically demands that every other row be kept free, which is typically achieved by obstructing every other row for placement.

The next aspect concerned with duplication is the interconnect. To make it possible to duplicate the interconnect, the vertical wires, that connect every other row, are forced to occupy only one routing channel in two. This ensures that a simple right shift of the vertical routing by a routing pitch (i.e. the distance

separating two routing tracks) does not create electrical shorts. As a consequence, vertical wires must be straight. If they were able to make turns, they would cross the adjacent routing tracks that are kept free for the duplicated vertical routes. On the contrary, wires of the “horizontal routing layers” are left free to make turns, as long as they remain in their placement row. Indeed, if the horizontal routing is confined within one row over two, the duplication of the “horizontal” wires in the upper or the lower rows does not interfere with the wires in the current row.

The constraints imposed to the place-and-route tool summarize as follows: as the design must be translated vertically by the height of one placement row (`ROW_HEIGHT`) for placement reasons and horizontally by one routing pitch (`PITCH`) for routing reasons, the whole placed-and-routed design is scheduled to move by a $(\delta x, \delta y) = (\text{PITCH}, \text{ROW_HEIGHT})$ vector translation. In backend taxonomy, this translation actually coincides with the minimum “placement site”.

At that point, the result of the duplication is two identical netlists interleaved one into the other. Notice that the netlists cannot be “de-interleaved” because they are not independent: some signals must be exchanged locally between abutted gates. As we will see in Sect. 3, it happens for the inverter gate in SABL and WDDL (Tab. 1(b)) and for all gates in SI-WDDL (Fig. 7).

The chip finishing steps shall not delete the indistinguishability of the two netlists. For instance, the dummies generator must be constrained to add dummies (metal pieces added randomly to fulfill the minimum density design rules) only in the rows in which placement is allowed. Afterwards, dummies are duplicated and translated by a placement site: they end up in the same routing environment as initially (no short is created) since the routing was duplicated in the same manner.

2.3 The Constraints Required by the “Backend Duplication”

As mentioned above, the “backend duplication” method is implemented by (1) constraining the design and (2) duplicating the placed-and-routed design. The constraints can be generated automatically by a script setting the following obstructions:

- **placement blockages** one row over two and on the rightmost placement site of the placeable row,
- **routing blockages** of one track channel over two for vertical metals and over the rows already marked obstructed for standard cell placement for horizontal metals.

Figure 8(a) illustrates these constraints on a 16×2 -site piece of floorplan. As far as the routing is concerned, these constraints are more flexible than the ones proposed in the “fat wire” method [9], since only vertical wires are forced to remain strictly straight. The metals whose preferred routing direction is horizontal are free to zigzag, provided they stay within their row. This degree of freedom is not negligible, since there are typically around 12 routing channels per row. This allows for both a more successful and a faster routing.

2.4 Insertion into an Existing Design Flow

As seen in Sect. 2.3, the “backend duplication” method need not redefine the design rules. It only relies on constraints on the CAD software. A typical backend flow includes the steps shown in Fig. 2. The insertion of the “backend duplication” consists in adding three steps (*i*, *ii* and *iii*).

Regular backend flow:	Flow compatible with the “backend duplication”. Added steps:
..... ←	<i>i</i> : Floorplan dimensioning
- Floorplanning ←	<i>ii</i> : Obstructions implementation
- Place-and-route	
- Clock tree generation	
- Scan chain optimization	
- Antenna effects correction	
- Custom steps, like ECO or SI fix	
- Dummies placement	
..... ←	<i>iii</i> : Duplication

Fig. 2. Typical backend flow and modifications (steps *i*, *ii* and *iii*) to implement the “backend duplication” method.

***i*. Floorplan dimensioning.** As a matter of fact, the floorplan of an design block is made up of two parts: the *core*, devoted to the standard cells placement and the *die*, that covers the core and an extra channel surrounding it. It is used for example to route a supply ring. The core horizontal dimension must be an even number of the routing PITCH and the vertical dimension an even number of ROW_HEIGHT. This condition ensures that the placement and the routing within the core do not extend out of the core after duplication.

The core can either be checked and repaired if one of the figures is odd or generated automatically. To end up with a core of density d and of aspect ratio r , the first step is to generate a core of density $d/2$ and of aspect ratio $r/2$ before duplication. Then the core dimensions (x, y) are retrieved, and a new core with the dimensions:

$$x' = \left\lceil \frac{x}{2 \times \text{PITCH}} \right\rceil \times 2 \times \text{PITCH}, \quad y' = \left\lceil \frac{y}{2 \times \text{ROW_HEIGHT}} \right\rceil \times 2 \times \text{ROW_HEIGHT}$$

is regenerated. Its density is slightly less than d and its aspect ratio roughly equal to r .

***ii*. Obstructions instantiation.** The constraint script described previously in Sect. 2.3 can be generated automatically as soon as the floorplan dimensions are known. This script is sourced after floorplanning and before place-and-route.

iii. Duplication. As far as standard cells are concerned, the duplication consists in a translation by a placement site followed by an horizontal flipping of each row.

The routing duplication is a bit more complex than a mere translation. Indeed, the design pins extend over the core to reach the die boundary. If the routing was simply translated, the duplicated design would have pins both inside and outside the die. To avoid this shortcoming, the routing extremities (u, v) of every wire undergo this transformation:

- if (u, v) belongs to the core, then $(u', v') = (u + \text{PITCH}, v + \text{ROW_HEIGHT})$,
- otherwise $(u', v') = (u, v)$.

Additionally, to prevent shorts, the constraints described in Sect. 2.3 actually extend till to die limits and the routing channels that are entirely outside the core are obstructed. These transformations are illustrated on Fig. 8(b).

The information needed to apply the duplication is the orientation and position of standard cells and the routing coordinates. The design exchange format (DEF) typically contains all this information. Given the simplicity of the DEF syntax and the availability of parsers [11], the duplication can be implemented easily.

It is also a good idea to apply the duplication on the Verilog netlist: it consists in duplicating all wires and all leaf instances (i.e. standard cells). Verilog parsers are easy to write, even from scratch. The key benefit of generating the duplicated Verilog netlist is to enable LVS verification.

2.5 Comparison with Related Works

K. Tiri [12] noticed that the balancedness of the routing is crucial to effectively protect a differential circuit against SCA. The solution put forward in [9] is based on “fat wires” routing: a large wire is first routed and then split into two minimum-sized wires. This method implies that:

- Specific design rules must be written for the “fat wires”.
- The only way for a wire to turn is to change layers.
- For the “fat wire” to access the pins of standard cells, their layout must be redefined.

The “backend duplication” implies none of these assumptions.

The experimental DPA [6] of F.G. Bouesse *et al.* [13] also showed that the weakest nodes in a differential layout correspond to unbalanced pairs. The backend correction flow described in [14] is iterative: the design is successively routed and analyzed, until every dual-rail pair is balanced. The analysis consists in the collection for every node of the sum of the parasitic elements extracted after every routing (more details in Sect. 4.2). This method requires a complex strategy to constrain the router and a non trivial algorithm to guide the iterative process towards a convergence point. On the contrary, the routing generated by “backend duplication” is balanced by design. However, the “backend duplication” only handles pairs of signals, whereas the iterative method [14] can route both dual and single-rail signals (data is dual-rail; acknowledge is single-ended.)

3 Suitability of the “Backend Duplication” Method with some Logic Styles

3.1 Backend Duplication for WDDL

The wave dynamic differential logic (WDDL, [2]) is a design style that uses standard cells by pairs, in such a way that at any step of the computation, one and only one of the two gates has a transition. This behavior masks the fluctuations of the power consumption due to irregular activity: the activity of a WDDL circuit is constant. The computations are split into successive *precharge* and *evaluation* steps. A Boolean function $e_{i \in \{0,1,\dots\}} \mapsto f(e_i)$ is computed using the two dual gates $f_T(e_i)$ and $f_F(e_i)$ that satisfy:

$$\begin{cases} \text{During precharge: } \exists i, & f_T(e_i) = \overline{f_F(e_i)}, \\ \text{During evaluation: } \forall i, & f_T(e_i) = f_F(\overline{e_i}). \end{cases} \quad (1)$$

Table 1(a) provides some examples of dual gates pairs suitable for WDDL. If the condition on the precharge in (1) cannot be met, the identity shown in Tab. 1(b) solves the problem out. The truth table of two dual gates (refer to

Table 1. Duality: definition, examples (a) and WDDL identity for the inverter (b).

(a)	Regular gate	Dual gate
Definition	$f(e_i)$	$\overline{f(\overline{e_i})}$
Examples	NOT	NOT
	NAND	NOR
	$\Pi \Sigma e_i$	$\Sigma \Pi e_i$

(b) Dual inverter

Regular inverter



Table. 2) shows a symmetry, that can also be observed at the transistor level, as shown in Table. 3.

The symmetry illustrated in Table. 3 suggests that standard cells are ready to be used in a WDDL flow using the “backend duplication” method. This is actually only partially true: the structures in transistors indeed perfectly superimpose, but in practice, PMOS (*symbol*: $\overline{\text{P}}\text{MOS}$) are drawn wider than NMOS (*symbol*: $\text{N}\overline{\text{MOS}}$). For this reason, in a commercial standard cell library, the pins of a gate (regular orientation: \square or R0) and of the X-symmetric (orientation: \square or MX) of its dual do not match exactly. Nevertheless, as they are located on the routing grid, they usually overlap.

Fortunately, it is easy to work around this difficulty. The procedure begins with an enlargement of the pins. Then, the pins are merged considering the intersection of the enlarged pins. The routing obstructions are basically made up of the metal not included in the union of the newly created pins:

$$\begin{cases} \text{PIN} = \text{PIN}(\text{NAND}) \cap \text{PIN}(\text{NOR}), & (\square \text{ in Fig. 5}) \\ \text{OBS} = (\text{OBS}(\text{NAND}) \cup \text{OBS}(\text{NOR})) \cup (\text{PIN}(\text{NAND}) \triangle \text{PIN}(\text{NOR})). & (\blacksquare \text{ in Fig. 5}) \end{cases}$$

Table 2. Truth table of the two dual functions NAND/NOR.

		NAND		NOR	
					
e_0	e_1	$\overline{e_0 \cdot e_1}$		$\overline{e_0 + e_1}$	
0	0	1		1	
0	1	1		0	
1	0	1		0	
1	1	0		0	

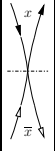
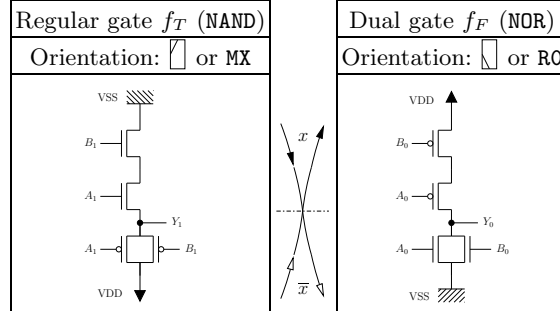


Table 3. Illustration of the NAND/NOR dual gate couple symmetry $\{f_T, (N, P), \text{MX}\} \leftrightarrow \{f_F, (P, N), \text{RO}\}$.



This procedure can be applied on the sole *abstract* view of the standards cells. Thus a simple LEF parser [11] can be used turn a standard cell library into a WDDL-compliant library. Instead of describing the parser into details, a graphical example on the NAND/NOR and AND/OR gate couples is shown in Fig. 5.

As far as cell placement duplication is concerned, the method presented in step *iii* (refer to Sect. 2.4) demands that, in addition to the duplication and the flipping, the gate be replaced by its dual.

3.2 Backend Duplication for Other Logic Gates

In order to apply the “backend duplication” method to SABL or SI-WDDL, the gates must be split into two parts: one computing *true* values, the other *false* values.

The splitting is straightforward for SABL, as shown in Fig. 6.

As for SI-WDDL, the division is a bit less trivial, but is sane since it forces the symmetry of the transistor schematic to be kept in layout view. The placement of each building block of the cell along with the indication of their orientation is provided in Fig. 7.

For both SABL and SI-WDDL, the gate pins must be designed in such a way they are left unchanged in a symmetry $y \leftarrow \text{ROW_HEIGHT} - y$ (or $\text{RO} \leftrightarrow \text{MX}$). This condition ensures that a connection to the pin of a regular gate (placed first) also arrives on a pin of the other half of the gate (placed while duplicating the backend at step *iii*). Additionally, the routing converges faster if the pins are placed on every other vertical routing track: the pins are better accessed if they are not below a vertical routing obstruction.

4 Implementing a Duplicated Netlist

4.1 The Example of a Secured DES Cryptoprocessor Design

In this section, we explain how a placed-and-routed netlist obtained by the “backend duplication” method can be embedded into a whole design. First of all, let us notice that after duplication, even global signals are duplicated: the duplicated backend has two clocks and two resets, that must be shorted together. The two scan chains can either be joined or be considered independently.

Most often, the whole cryptoprocessor need not be secured. The reason is that when implementing a non proprietary algorithm such as DES, the computation steps are public. As a consequence, the control leaks non confidential information. In most designs, the control (algorithm steps) can be clearly dissociated from the datapath (data processing).

It is relevant to derive the control of the duplicated datapath (dual-rail encoding, RTZ protocol) from the original control of the insecure datapath (single-ended, no RTZ): it allows to debug a single-ended control, which is easier to understand and faster to simulate. The method to update the regular control to make it compatible with the duplicated datapath requires that:

- The state machine can be frozen: it has an `enable` input. This enable forces the state machine to work twice as slow as initially to mimic RTZ.
- The control is wrapped by a converter single-to-dual rail for the datapath inputs and dual-to-single rail for its outputs. In addition to converting the control signals exchanged between the datapath and the control, the control wrapper also converts the datapath input and output data. Thus, seen from the outside, the cryptoprocessor keeps a single-ended interface. However, the internal architecture of the datapath is dual-rail RTZ secure logic obtained by “backend duplication”.

When the control is disabled (`enable = 0`), all the input signals of the datapath (provided by the control wrapper) are set to the precharge state (e.g. 00). This solution emulates the dual-rail RTZ protocol required by the duplicated architecture of the datapath. Moreover, this architecture is well suited for asynchronous gates implementations, such as SI-WDDL, because the datapath inputs (both data and control) are kept behind a register barrier, which guarantees that those signals are glitch-free. This condition is mandatory for SI-WDDL logic to work securely.

The schematic of Fig. 3 shows the secure architecture of a DES module. Let us notice that the control input signals (a simple start command, named `G0` in Fig. 3) is memorized as `G0_Q` over the two phases (precharge and evaluation), to prevent it from being discarded if it arrives when the control is disabled. The `G0` command can actually be activated at any time, because the cryptoprocessor environment is not aware of the RTZ behavior of the secured DES.

4.2 Method Cost and Security Evaluation

The method overhead is assessed below:

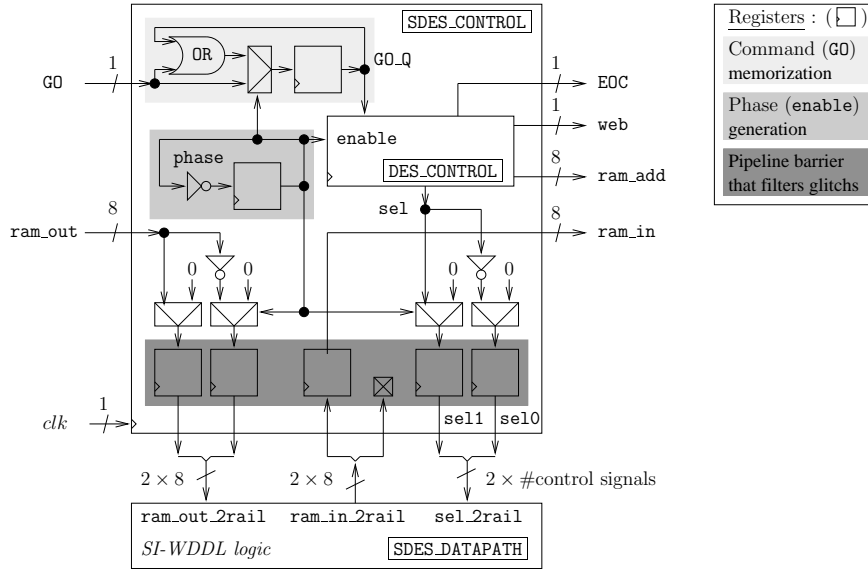


Fig. 3. Secured DES architecture. The duplicated datapath (SDES_DATAPATH), for example implemented in SI-WDDL logic, is obtained according to the method described in Sect. 2. The regular control (DES_CONTROL) is encapsulated into a wrapper (SDES_CONTROL) that can interface to the dual-rail datapath of DES.

- The circuit frequency is unchanged, but every encryption takes twice more time to execute because of the RTZ protocol.
- The area increase of the datapath depends on which secured gates are used. If WDDL gates are chosen, SDES_DATAPATH is simply twice as large as DES_DATAPATH. If SI-WDDL gates are chosen, we obtain a 15 times area increase¹. The overhead of the control area is 14%: the area of the module DES_CONTROL (resp. SDES_CONTROL) is 12 942 μm^2 (resp. 14 788 μm^2 .)

The increase of security can be assessed by the ratio of the two dual lines routing capacitances and resistances. The capacitance “C” accounts for the power dissipation occurring at every transition: $\frac{1}{2} \times C \times (VDD - VSS)^2$. The resistance “R” is responsible for the delay $R \times C$ of the transition propagation. The wire pairs are all the more balanced as the ratios $C(\text{true})/C(\text{false})$ and $R(\text{true})/R(\text{false})$ do not spread much around 1. Figure 4 shows the repartition of those ratios for the 2 211 internal wire couples of SDES_DATAPATH. The three data samples correspond to a dual placed-and-routed design, obtained by the “backend duplication” method, a dual placed and regular routed design, and a regular placed-and-routed design. Both the capacitances and resistances were

¹ The SI-WDDL gates were not optimized: a much better ratio can probably be obtained, even without any trade-off on the gate symmetry.

obtained using the RC extractor tool of Cadence SOC/ENCOUNTER. The technological information was produced by the Cadence COYOTE field solver.

The resistance of a “backend duplicated” circuit against EMA [7] has not been evaluated yet.

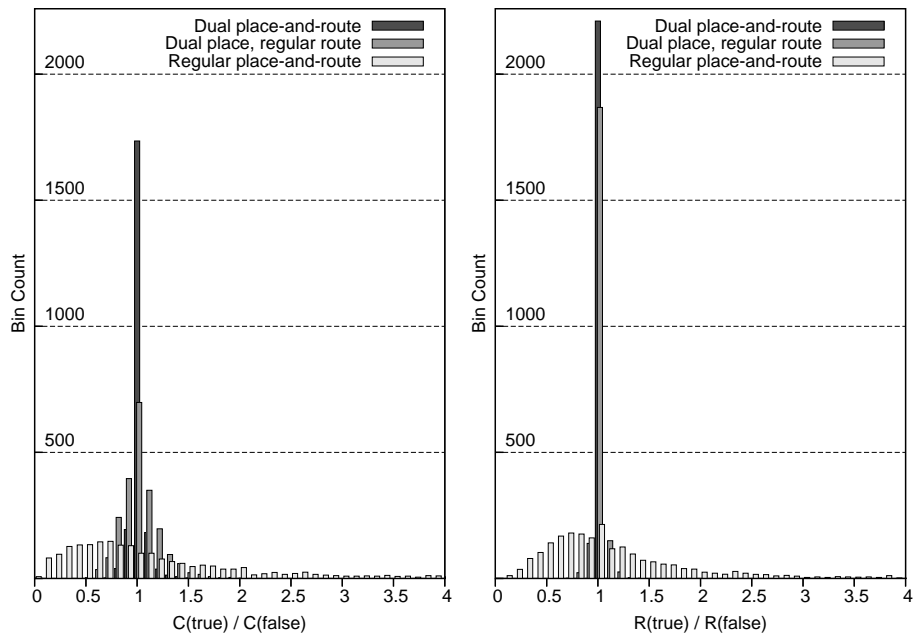


Fig. 4. Ratio of the capacitances and the resistances of SDES_DATAPATH dual nets.

5 Conclusion

Securing a cryptoprocessor against physical attacks (either SCA or FA) can be done at the algorithmic or at the implementation level. This paper focuses on the countermeasures on the hardware implementation. Many types of primitive gates suitable for secure computation have been proposed [1–4], but the issue of building cryptoprocessor out of them is seldom addressed. To the authors’ knowledge, only the “fat wire” method [9] partially tackles this problem.

We provide a complete description of a backend flow compatible with all of the above-mentioned gates. The method we describe can apply to all existing flows and requires no modification of the design rules.

The “backend duplication” method is illustrated on the example of a DES cryptoprocessor. This example also shows that the method is compatible with a secure partitioning of the design: only the datapath is duplicated. The emphasis

is placed on the insertion of the duplicated datapath into the whole DES, whose interface remains unchanged. This case study proves that the hardening of a cryptoprocessor can be fully automated and that the integration of the “backend duplication” method into an existing flow is seamless.

Acknowledgements

This work has been partially funded by the “conseil régional Provence Alpes Côte d’Azur” and the Research Ministry, through ACI SI MARS. The authors are also grateful to the AST division of STMicroelectronics (Rousset, France), for help in the design and the fabrication of the secured DES ASIC prototype.

References

1. Tiri, K., Akmal, M., Verbaughede, I.: A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In: Proceedings of ESSCIRC’02. (2002) pp 403–406.
2. Tiri, K., Verbaughede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: Proceedings of DATE’04. (2004) pp 246–251.
3. Moore, S., Anderson, R., Cunningham, P., Mullins, R., Taylor, G.: Improving Smart Card Security using Self-timed Circuits. In: Proceedings of ASYNC’02. (2002) pp 211–218.
4. Guilley, S., Hoogvorst, P., Mathieu, Y., Pacalet, R., Provost, J.: CMOS Structures Suitable for Secured Hardware. In: Proceedings of DATE’04. (2004) pp 1414–1415.
5. Kocher, P., Jaffe, J., Jun, B.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Proceedings of CRYPTO’96. Volume 1109 of LNCS., Springer (1996) pp 104–113.
6. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis: Leaking Secrets. In: Proceedings of CRYPTO’99. Volume 1666 of LNCS., Springer (1999) pp 388–397.
7. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Proceedings of CHES’01. Volume 2162 of LNCS., Springer (2001) pp 251–261.
8. Sokolov, D., Murphy, J., Bystrov, A.: Improving the Security of Dual-Rail Circuits. In: Proceedings of CHES’04. LNCS, Springer (2004) pp 282–297.
9. Tiri, K., Verbaughede, I.: Place and Route for Secure Standard Cell Design. In: Proceedings of CARDIS’04. (2004) pp 143–158.
10. Goubin, L., Patarin, J.: DES and Differential Power Analysis (The “Duplication” Method). In: Proceedings of CHES’99. LNCS, Springer (1999) pp 158–172.
11. LEF/DEF parsers: (website) <http://openeda.si2.org/projects/lefdef/> or <http://www.cadence.com/partners/languages/languages.aspx>.
12. Tiri, K., Verbaughede, I.: “Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In LNCS, ed.: Proceedings of CHES’03. Volume 2779 of LNCS., Springer (2003) pp 125–136.
13. Bouesse, G., Renaudin, M., Robisson, B., Beigné, E., Liardet, P.Y., Prevosto, S., Sonzogni, J.: DPA on Quasi Delay Insensitive Asynchronous Circuits: Concrete Results. In: Proceedings of DCIS’04. (2004) Bordeaux, France.
14. Bouesse, G., Renaudin, M., Dumont, S., Germain, F.: DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement. In: Proceedings of DATE’05. (2005) pp 424–429. Munich, Germany.

A Appendix: Graphical Illustrations of the “Backend Duplication” Method

Figures 5, 6 and 7 show how WDDL, SABL and SI-WDDL gates must be transformed prior to being used in the “backend duplication” design flow.

Figure 8 illustrates the “backend duplication” (steps *ii* and *iii*) on a floorplan suitable for the duplication (step *i* was already executed: the floorplan dimensions are even.)

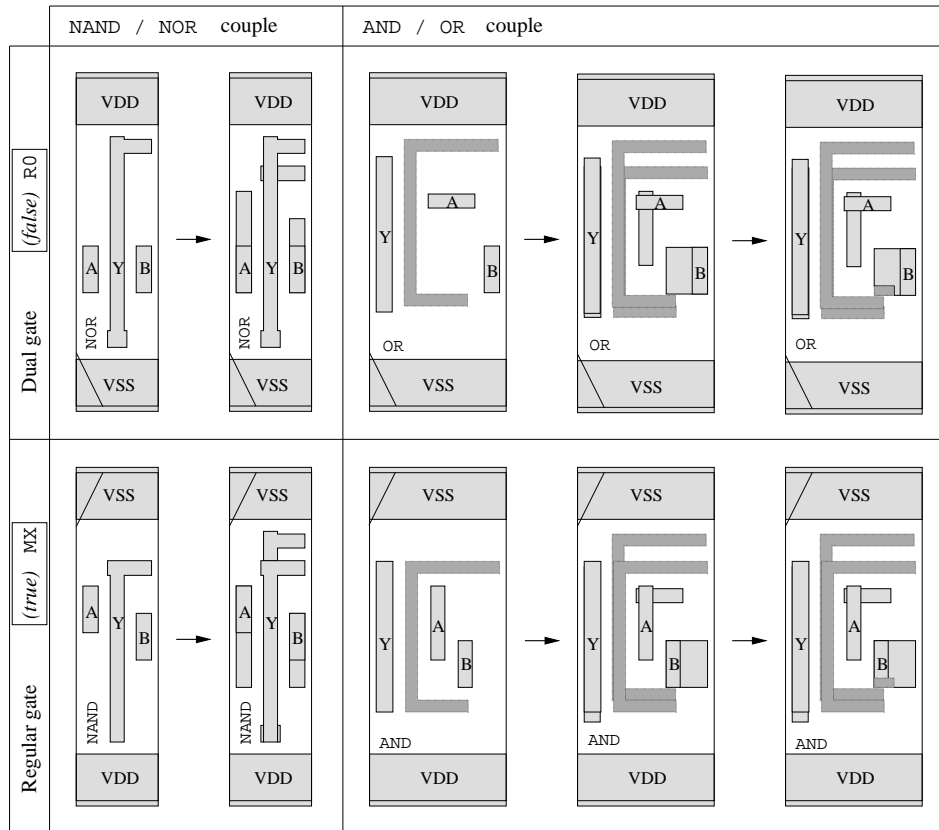


Fig. 5. Transformation on the *abstracted* views of the standard cells to make them WDDL-compliant [2]. This resulting gate couple satisfies the following condition: the abstract couples $\{f_T, (N, P), \mathbf{MX}\}$ and $\{f_F, (P, N), \mathbf{R0}\}$ perfectly superimpose.

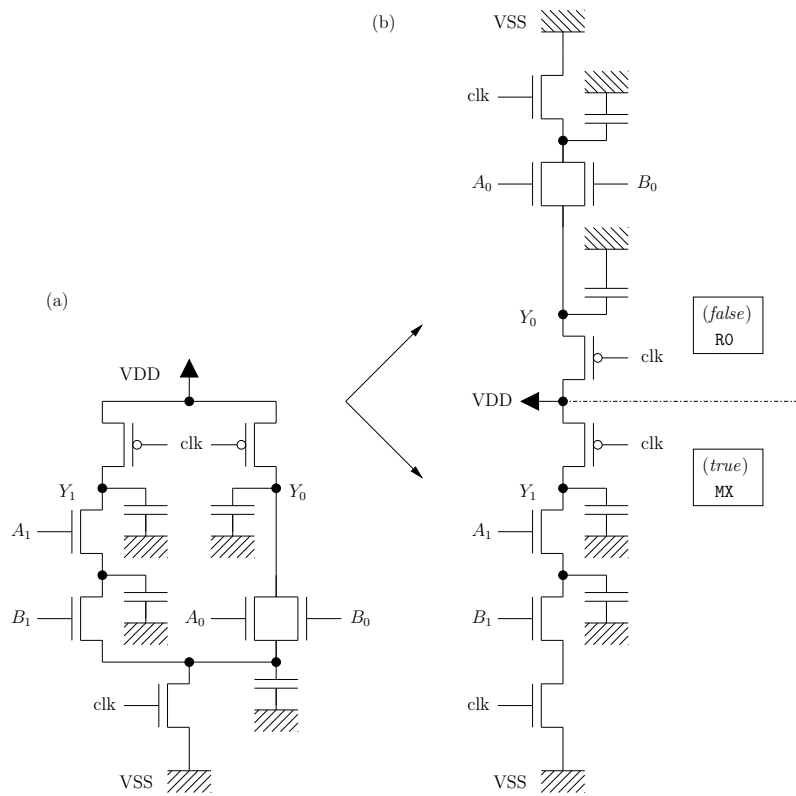


Fig. 6. Transformation of a NAND gate implemented in SABL [1] (a) into two dual gates (b), for subsequent use in the “backend duplication” design flow.

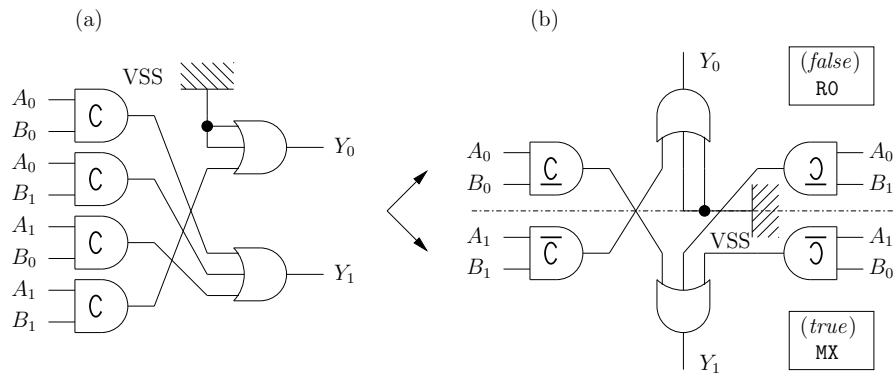


Fig. 7. Transformation of a NAND gate implemented in SI-WDDL [3] (a) into two dual gates (b). Notice that the two halves of the gate exchange signals.

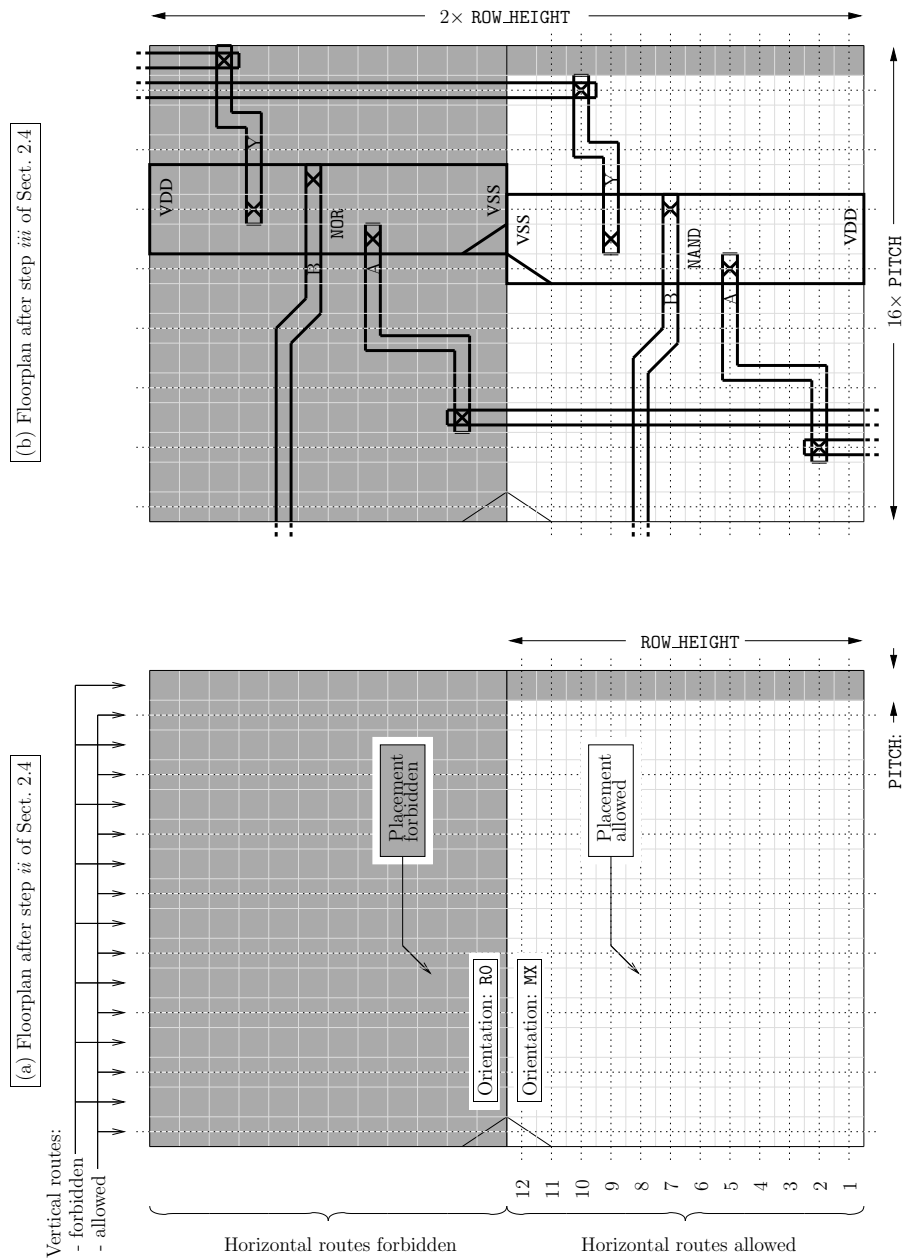


Fig. 8. (a) Place and route constraints, illustrated on a floorplan containing 16×2 placement sites. In PITCH units, the placement site is 1×12 and the routing grid offset is $\frac{1}{2} \times \frac{1}{2}$. (b) Final floorplan containing one single NAND gate (and its dual NOR gate). The horizontal wires can turn (wires connecting ports A, B and Y), whereas the vertical ones are straight. The vias that contact horizontal and vertical wires are noted \boxtimes .