Security Evaluation of a Balanced Quasi-Delay Insensitive Library (SecLib)

Sylvain Guilley, Florent Flament, Yves Mathieu, Renaud Pacalet

Security Evaluation of a Balanced Quasi-Delay Insensitive Library (SecLib)

Sylvain GUILLEY, Florent FLAMENT, Yves MATHIEU, Renaud PACALET
Institut TELECOM — TELECOM ParisTech, CNRS LTCI (UMR 5141)
Département COMELEC, 46 rue Barrault, 75 634 PARIS Cedex 13, FRANCE

Abstract

This article presents a library of cells enabling the realization of constant-power cryptoprocessors, natively protected against side-channel attacks. The proposed methodology uses a full-custom balanced quasi-delay insensitive (QDI) cell library, called "SecLib". It is suitable for a shielded routing method derived from the "backend duplication", using legacy CAD tools for the backend steps. The discussion is oriented towards the explicitation of topological constraints encountered in highly secure designs. We discuss the impact of intra-die technological mismatch on the security of SecLib.

Keywords: Standard cells design, power-constant logic, side-channel attacks mitigation, transistors mismatch, Monte-Carlo simulation.

1 Introduction

Side-channel attacks are a threat to the security of any electronic device. The seminal article of Paul Kocher [8] introduced several attacks, such as the SPA and especially the DPA, that can defeat cryptoprocessors, whatever the length of the keys. The vulnerability has been identified as an information leakage at the bit-level. Some high-level countermeasures against the DPA, such as duplicating [3] or masking [1], have been put forward. However, given the complexity of the underlying hardware, these solutions can be defeated by exploiting subtle non-logical phenomena, such as glitches [10].

Consequently, many *ad hoc* secured logic styles have been put forward. In the embedded security community, the so-called DPL (Dual-rail with Pre-charge Logic) family is overwhelmingly consensual. The DPL basically divide into two categories: "*power-constant*" and "*masked-power*" styles. In this paper, we investigate the feasibility of implementing optimally secured unmasked logic.

The rest of the paper is organized as follows. The specifications of the balanced QDI secured library "SecLib" is recalled in Sec. 2. Then, the layout challenges of the secured logical gates design are dealt with in Sec. 3. Finally, Sec. 4 concludes the paper and provides some perspectives. The appendices A and B describe the derivation of SecLib gates respectively from a template in GDS2 to build the final gate layout and from a template in VHDL to build the final simulation model.

2 Specifications of SecLib

As the "SecLib" cell library is already extensively described by Guilley *et al.* in [13], only the prominent features are recalled in this section.

SecLib is intended to be compatible, in terms of placement sites, with standard cells. This interoperability enables to reuse legacy cells for non-functional instances. SecLib, like other DPL libraries tailored for highly secured implementations, features security counter-measures at various levels: protocol, architecture, backend.

At the protocol level, a four-phase protocol enables to divide the computations into two steps: the computation proper and the precharge of the netlist. The first step consists in the computation of one iteration, while the second re-initializes all the nets so that the circuit is ready to start a new computation afresh, for instance with all the nets in a same electrical state.

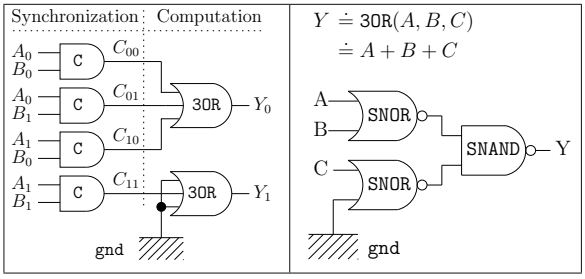

Figure 1: Schematic of the QDI secured AND gate (*left*) and its internal 3OR architecture (*right*).

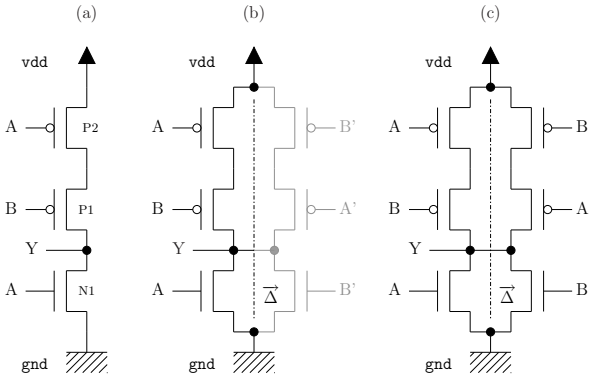

Figure 2: Transistor-level schematic of a SNOR gate.

Additionally, most secured cells rely on a dual-rail encoding: every logical bit is in fact carried by two wires. Many representations exist; however, a common one consists simply in associating the value *false* (0) to a wire and the value *true* (1) to the other. The rationale is to make any transition on the two wires indiscernible.

In dual-rail, every Boolean variable A is represented by a couple of two wires (A_0, A_1); when A is valid, $A = 0 \Leftrightarrow (A_0, A_1) = (1, 0)$ and $A = 1 \Leftrightarrow (A_0, A_1) = (0, 1)$. When A is invalid, $A_0 = A_1$. SecLib is optimized for $A_0 = A_1 = 0$.

The overall architecture of a representative SecLib gate (Fig. 1) is classical to the QDI logic [4]. The inputs synchronization disables anticipated evaluation. The gate timing is thus unconditional to the data. This feature protects the gate against the signature differences of unsynchronized DPL caused by variations of input delay time [12]. the inputs configuration decoding $(A, B) \mapsto (C_{00}, C_{01}, C_{10}, C_{11})$ is well suited for an indiscernible processing. Notice that, for unbalanced functions, the computation part is forced to be symmetric by the use of dummy gates (*cf.* Fig. 1 schematic on the *right*). SecLib is close to the logic described in this patent [2]; however, as shown in the sequel, SecLib is much easier to design and to dimension electrically due to the absence of bidirectional signals.

3 Layout of SecLib

3.1 Topological Issues Encountered in the Layout of SecLib

This section analyzes topological issues met when designing a library of dual-rail secured cells. It details the layout requirements arising from the *true* $\leftrightarrow$ *false* symmetry need. The layout issues can be circumvented to the sole SecLib instances, since non-functional gates (based on standard cells) do not leak any information. All layouts are realized in a 130 nanometers technology.

The structure of a balanced NOR (called SNOR, for Secured NOR) is shown in Fig. 2(c). The layout challenge consists in porting the symmetry from the schematic to the masks. The basic steps are illustrated in Fig. 2. First of all, an half-gate is designed (a). Then, two halves are instantiated, one in regular orientation R0, and the other in the mirrored orientation MY (b). This transformation allows for respect of an axial symmetry (the axis is denoted $\overrightarrow{\Delta}$.) The last step, (b) $\rightarrow$ (c), consists in the inner routing. It raises a topological problem, illustrated in Fig. 3. It is impossible to connect the couples (A, A') and (B, B') without a short-circuit, which results in a functionally invalid solution. This concern is not specific to SecLib cells, but indeed inherent to any geometrical balancing strategy.

An approximation is provided with in Fig. 4. Minimum sized polysilicium segments (130 nm $\times$ 180 nm), pointed out by arrows, connect the opposite nets: they are selected in Fig. 4 (c). Those four

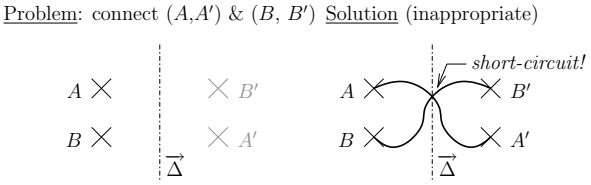

Figure 3: $\overrightarrow{\Delta}$-symmetry topological problem (*left*); invalid solution (*right*).

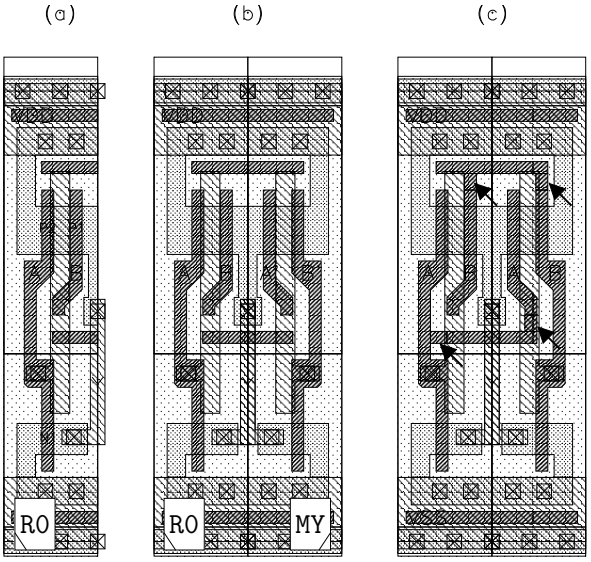

Figure 4: Construction of a quasi-symmetric `SNOR` gate layout (*cf.* corresponding schematic in Fig. 2).

segments constitute the sole symmetry violation.

The symmetrization methods presented above share the good property that transistors are paired in the same direction. This reduces the devices mismatches in case of mask misalignments during the manufacturing.

3.2 Gate Cocooning

A good cells library is geared towards the routability: the minimum number of metal layers must be used for the internal interconnections. In SecLib, only metals 1 and 2 are reserved for inner routing.

At the backend level, the decoupling between the computing logic and the routing resources is achieved thanks to an imprisonment of the transistors and the local interconnect in a `gnd`/`vdd` cage. The power/ground cage, illustrated in Fig. 5, also provides two interesting benefits. First of all, the cell is a cocoon, where the computation takes place confidentially. The symmetry violation between the cell (*axial symmetry*, hence *odd*) and the routing (*translation*, hence *even* [13]) is thus minimized. Second, the cage is very convenient to connect the cell to the power and ground global nets. In Fig. 5, the metal 2 pins (positive clock CP, input D, output Q, ground `gnd` and power `vdd`) are in bright cyan (▨), whereas obstructions for local interconnect are in low-intensity cyan (▧).

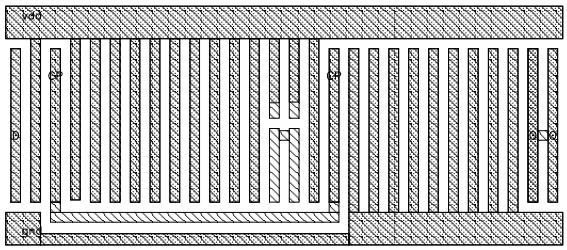

Figure 5: Illustration of the M2 cage, on a D-flip-flop. D and Q pins are made available respectively on the left and right sides of the cell.

3.3 SecLib Gates Interfaces

The position and the shape of the pins is an important issue: in order to be visible from a differential pair, the pins must often be larger than expected. For instance, to comply with the "backend duplication" routing method [6], the pins must respect a vertical symmetry, which increases their extension.

This constraint arises from the conjunction of the two symmetries:

1. translation $\mathrm{T}_{\overrightarrow{v}}$ by a vector $\overrightarrow{v}$ for the routing (upper constraint) and

2. glide reflection $\mathrm{S}_{\overrightarrow{\Delta}}$ around an axis $\overrightarrow{\Delta}$ for the cell two halves (lower constraint),

that must be met concomitantly by the pins, because they constitute the interface between the two symmetries domains. More formally, if pinF (resp. pinT) is the set of points from the floorplan (*i.e.*

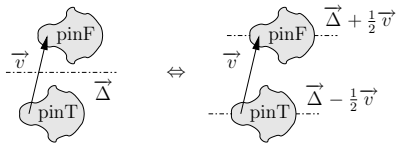

Figure 6: Translation $\mathrm{T}_{\overrightarrow{v}}$ and reflection $\mathrm{S}_{\overrightarrow{\Delta}}$ symmetries to be met by dual pins.

in $\mathbb{R}^2$) that belong to the *false* (resp. *true*) pin, then the symmetries impose that:

$$\begin{cases} \mathrm{pinF} = \mathrm{T}_{+\overrightarrow{v}}(\mathrm{pinT}) & \text{(routing)} \\ \mathrm{pinF} = \mathrm{T}_{+v_X\overrightarrow{e_X}} \circ \mathrm{S}_{\overrightarrow{\Delta}}(\mathrm{pinT}) & \text{(cell)} \end{cases}$$

and reciprocally, that:

$$\begin{cases} \mathrm{pinT} = \mathrm{T}_{-\overrightarrow{v}}(\mathrm{pinF}) & \text{(routing)} \\ \mathrm{pinT} = \mathrm{T}_{-v_X\overrightarrow{e_X}} \circ \mathrm{S}_{\overrightarrow{\Delta}}(\mathrm{pinF}) & \text{(cell)} \end{cases}$$

The second constraint can be simplified as the following local constraints:

$$\begin{cases} \mathrm{pinF} = \mathrm{S}_{\overrightarrow{\Delta}+\frac{1}{2}v_Y\overrightarrow{e_Y}}(\mathrm{pinF}) & \text{(pinF symmetry)} \\ \mathrm{pinT} = \mathrm{S}_{\overrightarrow{\Delta}-\frac{1}{2}v_Y\overrightarrow{e_Y}}(\mathrm{pinT}) & \text{(pinT symmetry)} \end{cases}$$

The proof is given below for pinT (the demonstration for pinF is much similar):

$$\begin{array}{l} \forall (x,y) \in \mathrm{pinT}, (x',y') = (x - v_X, y - v_Y) \in \mathrm{pinF}, \\ \text{thus } (x'',y'') = (x' + v_X, 2\cdot\Delta_Y - y') = \\ (x, 2\cdot(\Delta_Y - \frac{1}{2}v_Y) - y) \in \mathrm{pinT}\,. \end{array}$$

Figure 6 illustrates this "symmetry transportation" result.

Whenever possible, the pins are placed on the cell right and/or left sides so that two neighbor cells can be routed directly in metal 2. These recommendations are applied on SecLib gates, as shown on the example of the SecLib `AND` instance in Fig. 7.

The layout of other 2-input gates can be transposed straightforwardly from that of the `AND` gate. For instance, the family $(A,B) \mapsto \{\bar{A}\cdot\bar{B},\ \bar{A}\cdot B,\ A\cdot\bar{B},\ A\cdot B\}$ can be drawn based on the same *template*, specialized by the addition of vias at the relevant places [5]. Some details are provided in appendices A and B. SecLib cells are asynchronous, hence hazard-free: arbitrary Boolean functions can be implemented. Other non-synchronizing logics must restrict themselves to

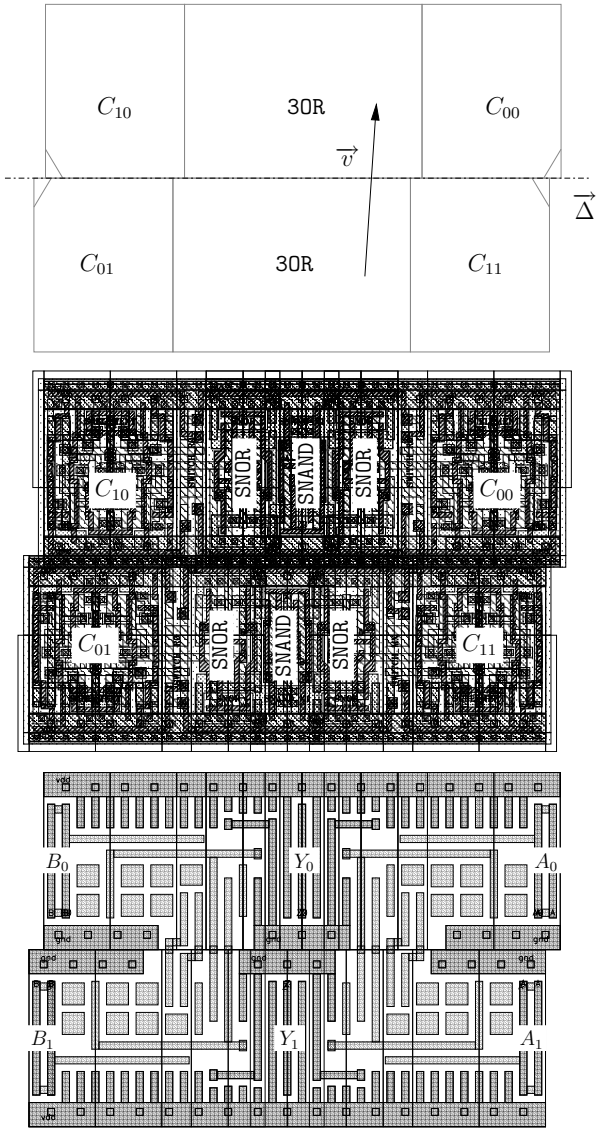

Figure 7: SecLib two-input `AND` gate floorplan (*top*), structure (*middle*) and interface (*bottom*).

positive functions in order not to generate and not to propagate data-dependent glitches. The average density of SecLib is 545 527 transistors/mm^2, versus 766 586 for the standard cells.

3.4 Mismatch Impact on Gates Balancedness

In deep sub-micron technologies, the electrical parameters are subject to local mismatches, that potentially wreak havoc the symmetry of secured gates. The term *mismatch* is defined as the electrical parameter deviation between identically designed components. It is customarily used in analog devices to predict their unbalancedness. The mismatch results from electrical fluctuations induced by nanoscopic variations in physical quantities.

A study on the mismatch in a differential interconnect network is carried out in [7]. This subsection accounts for the threshold voltage mismatch simulation on the instant and average current consumed by secured DPL gates. Both SecLib and WDDL [15] logics are studied, based on the example of an `AND` gate. The comparison is made between those two logic styles because they both use "full-amplitude" signals (from `gnd` to `vdd` volts – as the standard cells provided in founders design kits), which would not be the case for SABL [14] for instance. The testbench is depicted in Fig. 8. The environment is comprised of unitary inverters, of various multiplicities (M=3 or M=8): these values are chosen because they are representative of typical gates neighborhood. The DPL gate is powered by a separate supply, whose current $I(t)$ is extracted. Transistors are provided in 130 nm technology with mismatch models based on Pelgrom's linear characterization [11]. The Monte-Carlo option of electrical simulators is used to launch 500 simulations. The waveforms are represented in Fig. 9 for SecLib and WDDL logics.

The dispersion is important (about 5 %) on the maximum current peak amplitude. The mean relative difference is masked in the standard deviation for both SecLib and WDDL. The standard deviation is greater for SecLib, because the gates belonging to this library are comprised of more transistors than WDDL ones. The statistics on the average current relative difference show that:

- SecLib is more balanced than WDDL

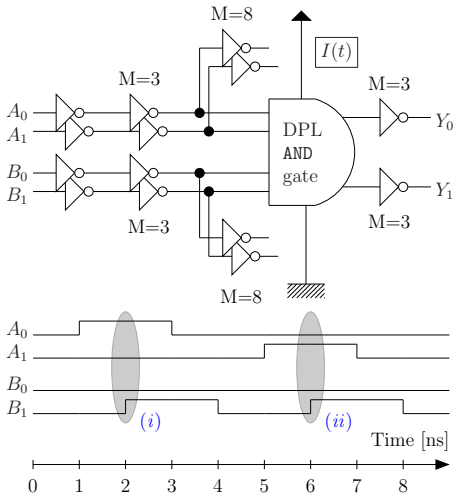

Figure 8: SPICE testbench for DPL gates instant current $I(t)$ extraction.

The relative difference of the instant current $I(t)$ and of the integrated current $\int I(t)\,\mathrm{d}t$ over the transition length are computed between: *(i)* the transition $A = 0$, $B : 0 \rightarrow 1$, and *(ii)* the transition $A = 1$, $B : 0 \rightarrow 1$. This relative difference between these two events is chosen because it is representative of the average unbalancedness that an attacker might exploit. The results are summarized in Tab. 1 in the form: "mean ± standard deviation", expressed in percent.

Table 1: Relative difference of the maximum and the integrated current consumed by two DPL gates.

	SecLib	**WDDL**
$\max I(t)$	(-1.01 ± 5.46) %	(-0.36 ± 4.87) %
$\int I(t)\,\mathrm{d}t$	$(+0.01 \pm 0.33)$ %	$(+1.63 \pm 0.22)$ %

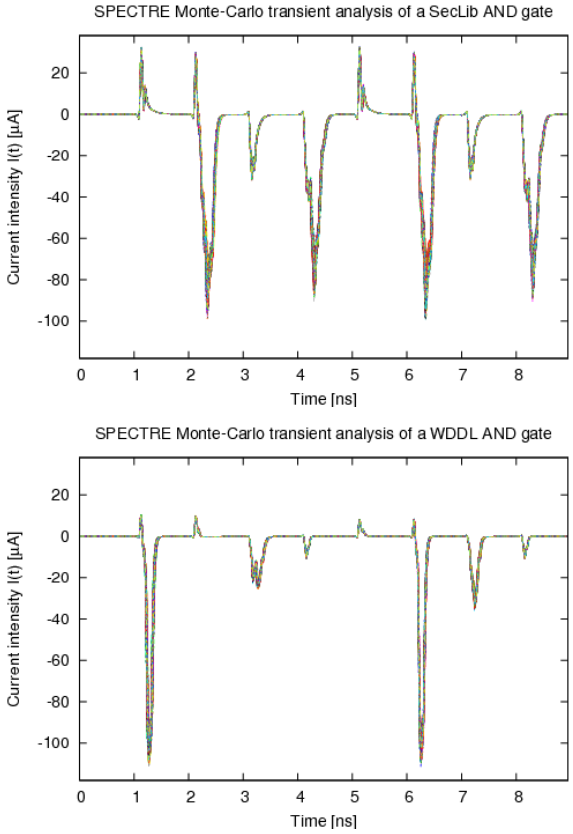

Figure 9: Monte-Carlo simulation results for SecLib (*top*) and WDDL (*bottom*).

($|+0.01|$ % *versus* $|+1.63|$ %),

- the mismatch is the overwhelming source of unbalancedness for SecLib, because the standard deviation is much greater than the mean ($0.33\ \% \gg |+0.01|\ \%$),

- the structural unbalancedness of WDDL is the principal cause of its unbalancedness ($0.22\ \% \ll |+1.63|\ \%$).

The "integrated current" metric is believed to be the most representative of measurements that an attacker might realize concretely: as a matter of fact, every measurement is low-passed filtered, because of the on-chip power grid and of the on-package decoupling capacitances [9, p. 33]. In conclusion, simulations tend to show that, from the pure computational standpoint, the level of security of SecLib logic is limited by the mismatch, while WDDL is still limited by its intrinsic dissymmetry.

4 Conclusion & Perspectives

This paper revisits the design of statically secured cells suitable for constant-power custom cryptographic ICs. Most previously proposed gates are vulnerable to a power attack exploiting the inputs skew. Therefore, this article focuses on a logic style (SecLib) in which gates inputs are systematically resynchronized. A method to port the symmetry constraints from the schematic to the layout is explicited. We emphasize the topological issues raised by the symmetric routing constraints. The question of the positions of the pins is extensively discussed. This issue is indeed crucial since it allows the gates to support balanced differential routing. The paper concludes positively on the feasibility of industrial-strength secured cells libraries. One strong contribution of this paper is to show that secured logics based on standard cells, such as WDDL, are limited by the unbalanced design, but that the balancedness of SecLib is limited only by the intra-die technological mismatch.

Future works will focus on the study of sequential gates (such as memory elements) and of complex circuits (comprised of more than one single gate).

References

[1] M. Akkar and C. Giraud. An Implementation of DES and AES secure against Some Attacks. In Springer-Verlag, editor, *Proc. of CHES'01*, volume 2162, pages 309–318, 2001.

[2] Loïc Duflot, Philippe Le Moigne, and Fabien Germain. Device Forming a Logic Gate for Minimizing the Differences in Electrical or Electromagnetic Behavior in an Integrated Circuit Manipulating a Secret, November 9 2006. Patent from the État Francais, représenté par le secrétariat général de la défense nationale, WO/2006/117391, `http://www.wipo.int/pctdb/en/wo.jsp?WO=2006117391`.

[3] Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In *CHES*, volume 1965 of *LNCS*, pages 158–172, 1999.

[4] S. Guilley, Ph. Hoogvorst, Y. Mathieu, R. Pacalet, and J. Provost. CMOS Structures Suitable for Secured Hardware. In *DATE*, pages 1414–1415, February 2004.

[5] Sylvain Guilley. *Geometrical counter-measures against side-channel attacks*. PhD thesis, GET / ENST, CNRS LTCI (UMR 5141), january 2007. `http://pastel.paristech.org/2562/01/phd.pdf`.

[6] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet. The "Backend Duplication" Method. In LNCS, editor, *CHES*, volume 3659, pages 383–397, August 2005. Edinburgh, Scotland, UK.

[7] Makoto Ikeda, Hiroshi Yamauchi, and Kunihiro Asada. Tamper Resistivity Analysis for Nano-meter LSI with Process Variations. In *ICECS*, pages 387–390, 2006.

[8] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.

[9] Huiyun Li. *Security evaluation at design time for cryptographic hardware*. PhD thesis, University of Cambridge, UK, April 2006. (Report UCAM-CL-TR-665, `http://www.cl.cam.ac.uk/techreports/`).

[10] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In *CT-RSA*, volume 3376, pages 351–365, 2005. San Francisco, CA, USA.

[11] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid State Circuits*, 24(5):1433–1439, 1989. DOI: 10.1109/JSSC.1989.572629.

[12] Daisuke Suzuki and Minoru Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In *CHES*, volume 4249 of *LNCS*, pages 255–269, 2006. `http://dx.doi.org/10.1007/11894063_21`.

[13] Sylvain Guilley and Florent Flament and Renaud Pacalet and Philippe Hoogvorst and Yves Mathieu. Secured CAD Backend Flow for Power-Analysis Resistant Cryptoprocessors. *IEEE Design & Test of Computers, special issue on "Design and Test of ICs for Secure Embedded Computing"*, 24, November-December 2007.

[14] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *ESSCIRC'02*, pages 403–406, Sept 2002.

[15] K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE*, pages 246–251. IEEE Computer Society, February 2004. Paris, France.

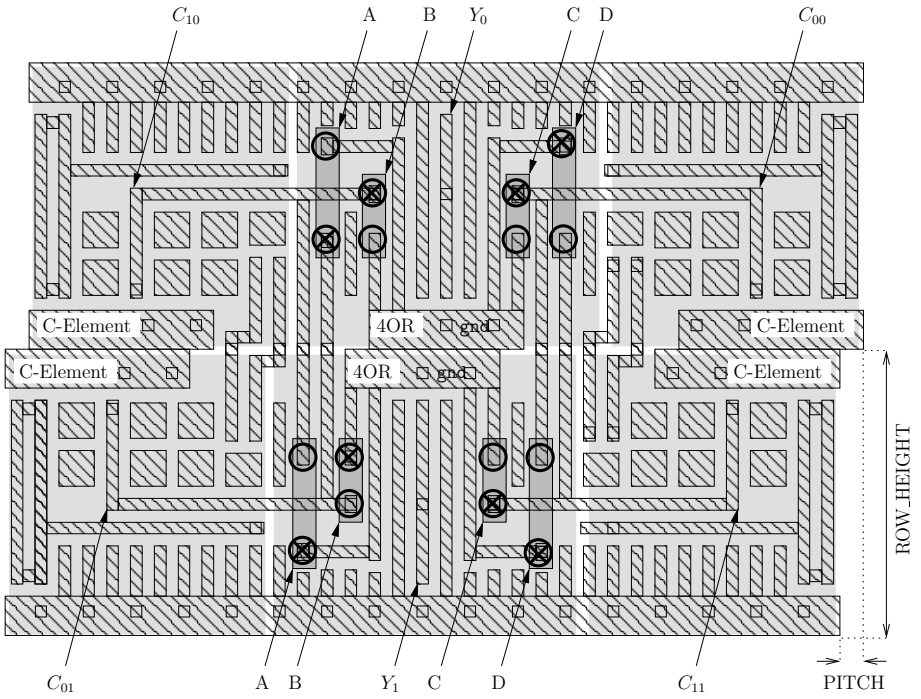

Figure 10: Layout of an unfinished SecLib gate: vias can be added at positions spotted by circles.

Table 2: Connectivity within a two-input SecLib template gate to specialize it to a `AND`.

Input \ `4OR`	`4OR` **driving** Y_0	`4OR` **driving** Y_1
A	`gnd` *or* $\underline{C_{01}}$	$\underline{\texttt{gnd}}$ *or* C_{10}
B	`gnd` *or* $\underline{C_{10}}$	$\underline{\texttt{gnd}}$ *or* C_{01}
C	`gnd` *or* $\underline{C_{00}}$	`gnd` *or* $\underline{C_{11}}$
D	$\underline{\texttt{gnd}}$ *or* C_{11}	$\underline{\texttt{gnd}}$ *or* C_{00}

A Generation of the Layout of Two-Input SecLib Gates from one Template

This section explains how to generate multiple two-input gates from one single GDS2 template. The structure given in Fig. 1 involves a `3OR` gate. For symmetry reasons, this `3OR` gate is actually one `4OR` with one input shorted to the ground. We provide in Fig. 10 with the layout (only metal-2 is shown) of an unbalanced (`XOR` and `XNOR` are excluded) two-input gate template.

The four inputs A, B, C & D of the `4OR` can be connected either to the ground or to one C-Element gate output. The connection points are indicated by a circle in Fig. 10. One via is instantiated to choose the adequate connection; it is represented by a cross in Fig. 10. Table 2 summarizes the connection possibilities; the vias are indicated by underlining the selected input for each input of the two `4OR` gates driving the differential output (Y_0, Y_1).

B Generation of the Behavioral Description of Two-Input SecLib Gates from one Template

The VHDL behavioral description of n-input QDI gates, upon which SecLib gates are built, is listed below. It enables fast functional simulations of SecLib netlists.

```
-- @file   qdi.vhd
-- @brief The behavioral specification of the quasi-delay insensitive (aka QDI)
--        primitives used in SecLib (Secured Library) logic.

library ieee;
use ieee.std_logic_1164.all;

-- Multiple input / single output 1-of-2 four-phase QDI gate behavioral model:
entity qdi is
  generic
  (
    tt: std_ulogic_vector -- Truth table
  );
  port
  (
    -- A(False, True), B(False, True), C(False, True) [if a'length = 6].
    a: in  std_ulogic_vector;
    y: out std_ulogic_vector
  );
begin
  assert a'length mod 2 = 0 and y'length = 2
  report "QDI gate ports are not dual-rail"
  severity failure;
  assert tt'length = 2**( a'length / 2 )
  report "QDI gate truth table has a bad dimension"
  severity failure;
end entity qdi;

architecture beh of qdi is
  signal c: std_ulogic_vector( 0 to 2**( a'length / 2 ) - 1 ); -- ``a'' decoded
  -- Evaluates whether one '1' of the truth table is hit. Models an OR gate:
  function eval( signal c: in std_ulogic_vector )
  return std_ulogic_vector is
    variable result: std_ulogic_vector( 0 to 1 ) := "00";
  begin
    for I in c'range loop
      if ( not tt( I ) and c( I )) = '1' then result( 0 ) := '1'; end if;
      if (     tt( I ) and c( I )) = '1' then result( 1 ) := '1'; end if;
    end loop;
    return result;
  end function eval;
begin
  -- Example on 3 bits:
  --+-------------+---------+
  -- |             | A  B  C |
  -- |     0 1 2   | 01 01 01 | <= "01" means "True,False"
  --+-------------+---------+
  -- | c( "0 0 0" ) | YN YN YN | <= Bits to test (Y=Yes, N=No) against 0 or 1
```

```
  -- | c( "0 0 1" ) | NY YN YN |
  -- | c( "0 1 0" ) | YN NY YN |
  -- | c( "0 1 1" ) | NY NY YN |
  -- | c( "1 0 0" ) | YN YN NY |
  -- | c( "1 0 1" ) | NY YN NY |
  -- | c( "1 1 0" ) | YN NY NY |
  -- | c( "1 1 1" ) | NY NY NY |
  --+--------------+----------+
  G_DECODE: for C_I in c'range generate
    -- Testing concomitant ''all 0'' and ''all 1'' bits:
    P_SEQUENTIAL_C_I: process( a ) -- Models a C-Element with a'length/2 inputs
      -- The type ''boolean_vector'' does not exist in VHDL, unfortunately:
      variable rdv: bit_vector( 0 to 1 );
      -- Tests whether or not the bit at position ''pos'' of the (32-bit)
      -- integer is set.
      function is_set( a: integer; pos: natural ) return boolean is
      begin
        assert pos < 32 -- Portability notice
        report "Integers are often represented as 32-bit strings"
        severity warning;
        if ( a/2**pos mod 2 ) = 0
        then return false;
        else return true;
        end if;
      end function is_set;
      function is_set( a: integer; pos: natural ) return integer is
      begin
        if is_set( a, pos )
        then return 1;
        else return 0;
        end if;
      end function is_set;
    begin
      rdv := "11"; -- By default, a double RdV... now cancelling the bad choices
      for ABC in 0 to a'length / 2 -1 loop -- n iterations for n-input gates
        if A( 2 * ABC + is_set( C_I, ABC )) = '1' -- The bit is set
        then rdv( 0 ) := '0'; -- No rendez-vous to ''0''
        else rdv( 1 ) := '0'; -- No rendez-vous to ''1''
        end if;
      end loop; -- On A, B, C, etc. dual-rail signals concatenated in ''a''
      assert not( ( rdv( 0 ) and rdv( 1 ) )= '1' )
      report "One C-Element reported a rendez-vous to both ''0'' and ''1''"
      severity failure;
      -- Updating ''c'' only if there were actually a rendez-vous:
      if rdv( 0 ) = '1' then c( C_I ) <= '0'; end if;
      if rdv( 1 ) = '1' then c( C_I ) <= '1'; end if;
    end process P_SEQUENTIAL_C_I;
  end generate G_DECODE;

  P_OUTPUT: y <= eval( c );

end architecture beh;

library ieee;
```

```
use ieee.std_logic_1164.all;
use work.all;

-- Two-input QDI gate
entity qdi2 is
  generic
  (
    tt: std_ulogic_vector -- Truth table
  );
  port
  (
    -- a(False, True), b(False, True) => y(False, True)
    A0, A1, B0, B1: in  std_ulogic;
    Z0, Z1:         out std_ulogic
  );
end entity qdi2;

architecture adaptor of qdi2 is
  signal inputs:  std_ulogic_vector( 0 to 3 ); -- A_False A_True|| B_False B_True
  signal outputs: std_ulogic_vector( 0 to 1 ); -- Z_False Z_True
begin
  P_INPUTS: inputs <= A0 & A1 & B0 & B1;
  I_QDI: entity qdi( beh )
    generic map( tt => tt )
    port    map( a => inputs, y => outputs );
  P_OUTPUTS_Y0: Z0 <= outputs( 0 );
  P_OUTPUTS_Y1: Z1 <= outputs( 1 );
end architecture adaptor;
```

Finally, the behavioral description of the SecLib `AND` gate is given below:

```
library ieee;
use ieee.std_logic_1164.all;
use work.all; -- For the visibility of the previously described entity "qdi2"

entity SAN2_X1 is
  port
  (
    A0, A1, B0, B1: in  std_ulogic;
    Z0, Z1:         out std_ulogic
  );
end entity SAN2_X1;

architecture template of SAN2_X1 is
begin
  I_QDI2: entity qdi2( adaptor )
    generic map( tt => "0001" )
    port    map( A0 => A0, A1 => A1, B0 => B0, B1 => B1, Z0 => Z0, Z1 => Z1 );
end architecture template;
```