Path Swapping Method to Improve DPA resistance of Quasi Delay Insensitive Asynchronous circuits

Fraidy Bouesse, Gilles Sicard, Marc Renaudin

TIMA Laboratory, 46 avenue Félix Viallet F38031 Grenoble, France
Fraidy.Bouesse@imag.fr

Abstract. This paper presents a Path Swapping (PS) method which enables to enhance the security of Quasi Delay Insensitive Asynchronous Circuits against Power Analysis (PA) attack. This approach exploits the logical symmetries of the QDI asynchronous blocks, particularly its data-path redundancies, to make all electrical curves used when implementing a PA attacks useless. Indeed, the idea is to average the electrical signatures of a block by randomly exchanging its data-paths during processing. To be able to implement this approach, we adopted a formal model of QDI circuits. Firstly, this formal model enables the designer to formally verify the symmetry of all paths in order to apply a path swapping method. Secondly, it offers the possibility to model the electrical signature of QDI asynchronous circuits. Finally, applying DPA on this formal model allows us to evaluate, in an early phase of the design, the circuit's sensitivity to the relevancy of the approach. Electrical simulations performed on a DES crypto-processor confirm the efficiency of the technique.

Keywords: QDI Asynchronous circuits, Power analysis, Path Swapping (PS).

1 Introduction

One of the most difficult task when designing secure systems is to protect devices from so-called side-channel attacks such as power analysis attacks (DPA, SPA), electromagnetic attacks, timing attacks and differential fault analysis. Since the discovery of these attacks, self-timed circuits have demonstrated their inherent capabilities to increase the security of chips. In fact, the Differential Power Analysis attack, firstly introduced by Paul Kocher [1], uses the correlation between the data processed by the circuitry and an observable power consumption to reveal the confidential information. Secret keys are retraced from the device by observing and monitoring the electrical activity of a device and performing advanced statistical computations.

Additionally to its absence of clock signal which demonstrates the practical way to eliminate a global synchronization signal, self-timed logic is well-known for its ability to decrease the consumption and smooth the current profile. Simon Moore et al. described in [2] techniques for improving chip security against side channel attacks. Their approach to improve chip DPA resistance is focused on the use of an alternative data encoding scheme such as one-hot data encoding (*1-of-N* codes). In the same design context, the Balsa synthesis system was modified to generate circuits with enhanced security against side-channel attacks [3]. The countermeasures that used Self-timed circuit properties are all focused on balancing the operation through special DI Coding scheme. Moreover, Paul Kocher also developed some countermeasures based on the same properties [4], and a new design concept has been presented in [5] by Danil Sokolov et al. who used standard dual-rail logic with a two spacer protocol working in a synchronous environment. The results obtained by exploiting Self-timed logic have been reported in several papers. J. Fournier et al. evaluated and demonstrated in [6] that Speed Independent asynchronous circuits increase resistance against side channel attack and the concrete results of the effectiveness of the QDI asynchronous logic against DPA has been reported in [7].

However, all these papers concluded in terms of DPA that there still exists some residual sources of leakage which can be used to succeed an attack. These residual sources of leakage that are still observable when implementing a DPA attack on balanced QDI asynchronous circuits are addressed by G.F. Bouesse et al. in paper [8]. They show that, the residual sources of leakage of a balanced QDI circuits come from the back end steps which introduce some electrical dissymmetries, especially through the routing capacitances. The solutions implemented in paper [8] and also mentioned in paper [2] in order to remove electrical dissymmetries, consist in constraining the placement and routing. They defined a place and route methodology which enables the designer to control the net capacitances. Contrary to the previous proposed countermeasures mentioned above, the approach described in this paper does not try to get rid of these residual sources of leakage, but instead makes it not exploitable by the DPA attack.

The PS (path swapping) method takes advantage of the structural symmetries that exist in QDI asynchronous circuits or those proposed in [2][3]. In fact, in such circuits many identical structures called paths exist that can be alternatively used to compute a given function. Therefore, the idea is to randomly choose one of the possible paths to compute the function which hence averages the electrical signature over all the paths. The issue lies in succeeding to do so with minimum overhead.

The paper is organized as follows. Section 2 recalls the asynchronous properties that are used to increase DPA resistance, especially the N-rail Quasi Delay Insensitive asynchronous logic together with the four-phase protocol. Section 3 introduces the path swapping technique and section 4 presents the formal approach chosen to implement this technique. The specification of the formal model adopted to represent the circuits is first described, and then formal DPA resistance criteria at logical and electrical levels are defined using this circuit model. It enables us to formally justify the path swapping technique. The approach is validated with the case study described in section 5 and results obtained using electrical simulations are reported in section 6. Section 7 concludes the paper.

2 Previous Works: QDI circuits and security

QDI asynchronous circuits represent a class of circuits controlled by the data themselves. In fact, an asynchronous circuit is composed of individual modules communicating to each other by means of point-to-point communication channels. Therefore, a given module becomes active when it senses the presence of incoming data. It then computes them and sends the result to the output channels. Communications through channels are governed by a handshaking protocol which requires a bi-directional signalling between senders and receivers (request and acknowledge). Among the main classes of handshaking protocols [9] we only consider and describe the four-phase protocol (fig.1) which has an interest in security.

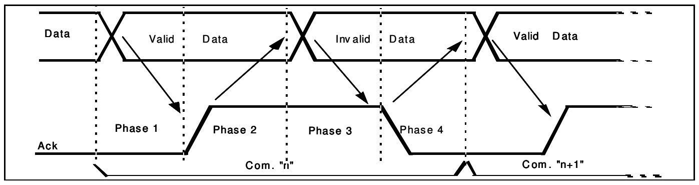


Fig. 1. Four-phase handshaking protocol.

The four-phase protocol protocol requires a return to zero phase for both data/requests and acknowledgements. Contrary to synchronous circuits where the shape of the current (current peaks) depends on the previous states and data values, QDI asynchronous logic using a four-phase protocol re-initializes all previously activated nodes before processing a new data. This behaviour enables the designer to precisely control the transitions involved in a given computation. Moreover, because it is based on hazard free logic QDI asynchronous circuits eliminate all current variations caused by glitches.

The implementation of a four-phase handshaking protocol requires sensing the presence of data in phase 1 and their absence in phase 3. In order to do so, dedicated logic and special encoding are necessary for sensing data validity/invalidity and for generating the acknowledgement signal. Considering that one bit has to be transferred through a channel using the four phase protocol, one has to encode three different values: invalid, valid at '1', valid at '0'. Two wires (A0, A1) are then required to encode the three states. This technique is called dual-rail encoding. The acknowledgement signal is generated by taking advantage of the data-encoding. As depicted in figure 2, a Nor gate is usually used to sense the dual-rail encoding output for generating the completion signal.

Dual-rail encoding is easily extended to N-rails. It is called *1-of-N* encoding. This encoding data scheme is useful to reduce the number of electrical transitions involved in a given computation. For the sake of DPA resistance, *1-of-N* encoding ensures that the same number of transitions is required to encode the values *0* to *N-1* and guarantees a constant Hamming weight.

As an example, consider the xor function. Figure 2 shows a dual-rail xor gate implementation. All computations of this dual-rail xor gate involve a fixed and constant number of transitions regardless of the input data. Hence, the opportunity to have data independent power consumption i.e. not correlated to the processed data seems achievable and this is exactly the goal pursued.

However, the QDI implementation of a function is not always balanced. In such cases, the gate structure is modified to ensure that all data-paths and control paths can be balanced and do involve a constant number of transitions [2].

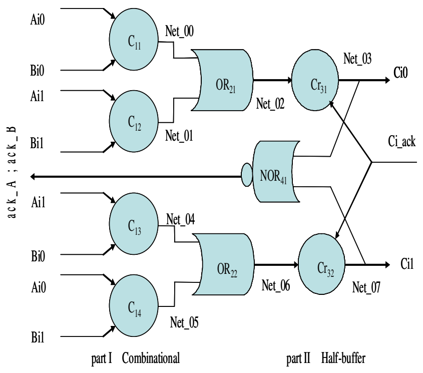


Fig. 2. Dual-rail gate with four-phase protocol. (Cr = Muller gate with a reset)

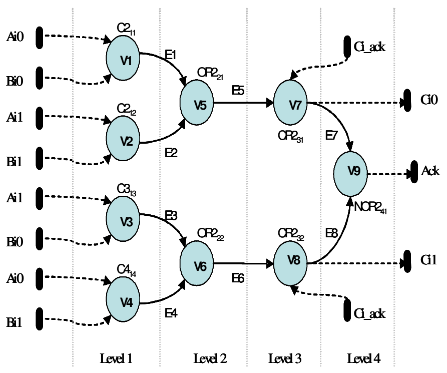


Fig. 3. Annotated directed graph G_{Xor}=(V,E) of the Dual-rail Xor gate of fig.2.

To summarize, the use of QDI logic together with a four-phase protocol and *1-of-N* data encoding enables:

- to control the current by removing all spurious transitions (hazard free logic).
- to equalize the number of transitions using a constant hamming weight representation of data.
- to control the type of all transitions.
- to reduce the dependence between data and power consumption using symmetrical structures of data and control paths.

As described in [2][4] all these properties are suited to implement secure chips against DPA. The design methodology developed in this paper is based on these properties of QDI asynchronous logic, particularly the data path symmetries. They are exploited to randomly average electrical signatures so that the electrical dissymmetries described in [8] and amplified after the back end steps become useless. We now analyse this point by introducing the path swapping method.

III Contribution: Path Swapping method

The goal of this new design approach is to eliminate the electrical effects which enable to succeed the DPA attack on QDI circuits. To do so, we do exploit the circuit structure which exhibits a lot of symmetries. Indeed, in the blocks of such circuits there exist many identical physical paths from their primary inputs to their primary outputs. The idea is to randomly choose one of the possible paths to compute the function. More formally, let's define n_c as the number of output channels using *1-of-N* data encoding. Each output channel i has N rails. We can represent the logical equation of each rail by $f_{ij}(A_x)$ and its dynamic current profile by $P_{ij}(t/A_x)$ when the input value A_x is computed. The value A_x is one element of E_i, the set of all possible input values which activate the channel i. The indexes i and j identify the channel number and the rail number respectively. For each rail there is a data-path from the primary output rail considered to the primary inputs (N data-paths).

If all data-paths are logically symmetric, it means that:

$$\begin{array}{l}\forall \quad A_x \in E_i \quad \Rightarrow \quad f_{i1}(A_x) = \ldots = f_{iN}(A_x) \\ \text{and} \quad P_{i1}(t / A_x) \neq \ldots \neq P_{iN}(t / A_x)\end{array} \qquad (1)$$

This equation (1) shows that for any input value A_x of a QDI asynchronous block we can acquire N different electrical signatures, corresponding to the same computed logical function. The principle of this new design method is to randomly choose among the possible f_{ij} functions and therefore their corresponding electrical signatures in order to make the DPA attack inefficient. To illustrate this, let's now consider the simple xor function and the implementation depicted in figure 2. First, note that this circuit is balanced in the sense that the computation of the xor function always involves the same switching sequence of gates.

$$\text{gate C} \rightarrow \text{gate Or} \rightarrow \text{gate C}_r \rightarrow \text{gate Nor}$$

Besides, the structure is symmetric in the sense that there exist two identical logical paths between the outputs and the primary inputs.

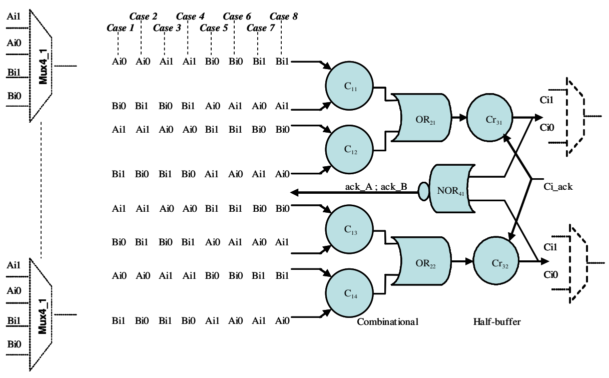
$$\text{- first data path :} \quad \left.\begin{array}{l} gate\ \text{C}_{11} \\ \text{gate C}_{12} \end{array}\right\} \rightarrow gate\ Or_{21} \rightarrow gate\ Cr_{31} \qquad \text{- second data path :} \quad \left.\begin{array}{l} gate\ \text{C}_{13} \\ \text{gate C}_{14} \end{array}\right\} \rightarrow gate\ Or_{22} \rightarrow gate\ Cr_{32}$$

Moreover, each path can be split into two execution-paths which represent an exclusive path that can be used to process a rail.

$$\begin{array}{ll} \text{* execution - paths of the first output rail} & \text{* execution - paths of the second output rail} \\ \text{- gate C}_{11} \rightarrow \text{gate Or}_{21} \rightarrow \text{gate Cr}_{31} & \text{- gate C}_{13} \rightarrow \text{gate Or}_{22} \rightarrow \text{gate Cr}_{32} \\ \text{- gate C}_{12} \rightarrow \text{gate Or}_{21} \rightarrow \text{gate Cr}_{31} & \text{- gate C}_{14} \rightarrow \text{gate Or}_{22} \rightarrow \text{gate Cr}_{32} \end{array}$$

Therefore, as shown in figure 4, different sets of inputs and outputs can be applied in such a structure. For the sake of DPA resistance, it is worthwhile to observe that for constant values at the inputs four different electrical signatures can be obtained using inputs and outputs permutations. We call this method path swapping because interchanging the inputs and/or outputs leads to swap the execution from logical paths to other logical paths inside the circuit.The realization of this technique requires the use of multiplexers/demultiplexers and a random number generator (RNG). Multiplexers/Demultiplexers are used to permute inputs/outputs and are controlled by the random number generator. The use of a random number generator guarantees an equiprobable and unpredictable distribution function of inputs/outputs. Considering the example illustrated in figure 5, if *M* data have to be computed, the random number generator must ensure to randomly activate each execution-path *M/4* times. The specifications of the random number generator and of the Multiplexers/ Demultiplexers blocks are addressed in section 4.6. The path swapping method can only be efficiently implemented with design logic which offers an opportunity to implement symmetrical and balanced circuits as it is the case with QDI asynchronous circuits. This type of logic enables to implement the PS method with a minimum area overhead and by slightly changing the performance of the circuit.

To apply this technique to QDI asynchronous circuits, we have specified a formal design approach which enables us to formally verify the symmetry of the circuit and formally verify at each design level the relevancy of the path swapping approach. This design approach is based on a formal representation of QDI circuits.



This example illustrates all possible permutations of the inputs and outputs that can be applied on a dual-rail xor gate. The first case corresponds to a standard implementation and can be used as a reference one. In the second case, only the rails of the input channel $B_i(B_i0,B_i1)$ are swapped which also requires a permutation of the output rails. In the third case, the rails of input channel $A_i(A_i0,A_i1)$ are swapped and this requires a permutation of the output rails. Both input channel rails $B_i(B_i0,B_i1)$ and $A_i(A_i0,A_i1)$ are swapped in the fourth case without any permutation of the output rails. In the other cases the rails of the input channel $B_i(B_i0,B_i1)$ are permuted with rails of the input channel $A_i(A_i0,A_i1)$.

Fig. 4. Path Swapping method applied to the Dual-rail Xor gate

4 Formal model of QDI asynchronous circuits

The formal model we have adopted to automate and verify at each design phase all QDI properties described above is based on the digraph (directed graph) theory.

4.1 Digraph of QDI asynchronous circuits

A digraph is a graph in which the edges are directed from the initial vertex *(a)* to the terminal vertex *(b)*. If *G=(V,E)* is a digraph, then *V* and *E* are respectively the set of vertices and the set of edges of the digraph *G*. For the purpose of representing the QDI asynchronous circuits as a digraph, we define the two following rules:

* *All gates of the circuit are considered as the elements of the set V (vertices).*

* *All interconnections are considered as the elements of the set E (directed edges).*

For example let's consider the block of figure 2. Its representation in the form of a digraph $G_{Xor}=(V,E)$ is presented in figure 3. Each vertex (V_i) and directed edge E_i are respectively annotated by the name of the corresponding gate and interconnection. All dotted lines represent primary inputs and outputs of the block.

$$M_{BG} = \left(\begin{array}{c|ccccccccc}
 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & V_9 \\
\hline
V_1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
V_2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
V_3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
V_4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
V_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
V_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
V_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
V_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
V_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

a- Boolean Matrix

$$M_{AG} = \left(\begin{array}{c|ccccccccc}
 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & V_9 \\
\hline
V_1 & C2_{11} & 0 & 0 & 0 & OR2_{21} & 0 & 0 & 0 & 0 \\
V_2 & 0 & C2_{12} & 0 & 0 & OR2_{21} & 0 & 0 & 0 & 0 \\
V_3 & 0 & 0 & C2_{13} & 0 & 0 & OR2_{22} & 0 & 0 & 0 \\
V_4 & 0 & 0 & 0 & C2_{14} & 0 & OR2_{22} & 0 & 0 & 0 \\
V_5 & 0 & 0 & 0 & 0 & 0 & 0 & CR2_{31} & 0 & 0 \\
V_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & CR2_{32} & 0 \\
V_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & NOR2_{41} \\
V_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & NOR2_{41} \\
V_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

b- Matrix annotated with labels

Fig.5. Matrix of the digraph $G_{Xor}=(V,E)$

There are several different ways to represent a graph. The two common ways are as an adjacency list or as an adjacency matrix. The adjacency list is appropriate for software implementations. However, for the sake of clarity, we use in this paper an adjacency matrix representation.

4.1.1 Adjacency Matrix

The adjacency matrix of a digraph *G* is the *n-by-n* matrix $(M_G(n,n))$ where *n* is the total number of vertices of the graph *G*. If there is an edge from vertex *(a)* to vertex *(b)*, then the element *((a),(b))* of the matrix is 1, otherwise it is 0. It is called the boolean matrix of G $(M_{BG}(n,n))$ (figure 5-a). In order to represent in the Boolean matrix the input vertices which are defined as the terminal vertices of all input edges (inputs dotted lines in fig. 3), we define the following property:

** For any element (a) of the set V, if ((a),(a))=1, then the vertex (a) is considered as an input vertex.*

From this boolean matrix of the digraph *G*, we can associate an annotated matrix $(M_{AG}(n,n))$ of the annotated digraph *G,* where the directed edge between vertex *(a)* and vertex *(b)* are represented by the name of the terminal vertex as illustrated in figure 5-b. The number of elements in each row *"i"* gives the number of elements connected to a vertex (V_i) and the number of elements in each column "*j"* gives the number of inputs of the vertex (V_j), except for the output vertices. In fact, output vertices which generate the output signals are differently labelled in the matrix (in bold) in order to facilitate their identification. This makes easier finding the sub digraphs which compute each output rail of the block, and then to evaluate their data-path symmetries.

4.2 Logical symmetry of data-paths

The data-path symmetries are analyzed by extracting in the digraph all subdigraphs that generate each output rail. This extraction is done by the exploration of the matrix of the block. The exploration starts by the identification of all output vertices, then collecting for each identified vertex, all its ascendant vertices. Each vertex output is then considered as an anti-root of the tree towards which directed edges are oriented. Let's consider the matrix of the digraph of figure 5-b. The matrix of the subdigraph $G_{Ci0}=(V_{Ci0},E_{Ci0})$ of rail *Ci0* and the matrix of the subdigraph $G_{Ci1}=(V_{Ci1},E_{Ci1})$ of rail *Co1* are presented in figure 6.

$$M_{AGCo1} = \left(\begin{array}{c:cccc} & V_3 & V_4 & V_6 & V_8 \\ \hdashline V_3 & C2_{13} & 0 & OR2_{22} & 0 \\ V_4 & 0 & C2_{14} & OR2_{22} & 0 \\ V_6 & 0 & 0 & 0 & \mathbf{CR2_{32}} \\ V_8 & 0 & 0 & 0 & 0 \end{array}\right) \qquad M_{AGCo0} = \left(\begin{array}{c:cccc} & V_1 & V_2 & V_5 & V_7 \\ \hdashline V_1 & C2_{11} & 0 & OR2_{21} & 0 \\ V_2 & 0 & C2_{12} & OR2_{21} & 0 \\ V_5 & 0 & 0 & 0 & \mathbf{CR2_{31}} \\ V_7 & 0 & 0 & 0 & 0 \end{array}\right)$$

a- rail Ci1 b- rail Ci0

Fig. 6. Matrix of the subdigraph of the digraph G.

To be able to define the symmetry of the data-paths between N-rail of the encoding bit, let us introduce the notion of execution-path.

** The execution-path is defined as any exclusive path that can be used to process one output rail.*

One property of the QDI asynchronous logic is to offer the opportunity to use convergence gates. At each cycle, these gates guarantee the exclusivity of one of its inputs, i.e only one input of the convergence gate is activated. This property enables us to deduce the execution-paths by the exploration of the matrix. The OR gate of both subdigraphs is used as a convergence gate. It means that for each output rail there are two execution-paths described by the subdigraphs and their equivalent matrices in figure 7.

$E_{GC}(1)=(G_{E1},E_{E1})$ *with* $G_{E1}=\{V_1,V_5,V_7\}$ *and* $E_{E1}=\{E_1, E_5\}$
$E_{GC}(2)=(G_{E2},E_{E2})$ *with* $G_{E2}=\{V_2,V_5,V_7\}$ *and* $E_{E2}=\{E_2, E_5\}$
$E_{GC}(3)=(G_{E3},E_{E3})$ *with* $G_{E3}=\{V_3,V_6,V_8\}$ *and* $E_{E3}=\{E_3, E_6\}$
$E_{GC}(4)=(G_{E4},E_{E4})$ *with* $G_{E3}=\{V_4,V_6,V_8\}$ *and* $E_{E3}=\{E_4, E_6\}$

$$M_{EGC(i)} = \begin{array}{c|ccc} & V_j & V_k & V_l \\ \hline V_j & C2 & OR2 & 0 \\ V_k & 0 & 0 & CR2 \\ V_l & 0 & 0 & 0 \end{array}$$

Fig. 7. Subdigraphs $E_{GC}(i)$ and their equivalent matrices $M_{EGC(i)}$.
$(V_J,V_k,V_l) \in [(V_1,V_5,V_7);(V_2,V_5,V_7);(V_3,V_6,V_8);(V_4,V_6,V_8)]$

One way to formally analyze the data-path symmetry is to analyze the symmetry of each execution-path, by processing the digraph isomorphism.

4.2.1 Isomorphism of a digraph

Two digraphs G_1 and G_2 are isomorphic if there is a one-to-one correspondence between their vertices and directed edges. If there is a directed edge between two vertices of G_1, then there is a directed edge between the two corresponding vertices in the digraph G_2. More formally,

** For any directed edge ((a),(b)) of G_1, G_2 is isomorphic to G_1 if and only if F((a),(b)) is a directed edge of G_2 (F is an isomorphic function).*

In terms of matrices, if A_1 and A_2 are respectively the matrices of G_1 and G_2, the digraph G_1 is isomorphic to the digraph G_2 if there is a classification of the vertices of G_2 such as the boolean matrix of A_1 and A_2 are equal.

** If $A_1=A_2$ then G_1 and G_2 are isomorphic*

Thus, the analysis of the data-paths symmetries is equivalent to determinate the isomorphism of block subdigraphs (each subdigraph represents one execution-path of the block).

** Data-paths are symmetrical at logical level if and only if their digraphs are isomorphic*

Therefore, blocks are balanced if their data-paths are symmetric. If not, the module is said unbalanced. From the previous example, as the matrices $E_{GCo}(1)$, $E_{GCo}(2)$, $E_{GCo}(3\ )$ and $E_{GCo}(4)$ are equal, then their digraphs are isomorphic, so that the digraph $G_{XOR}(V,E)$ is a balanced structure. However, the QDI implementation of a function is not always balanced, in such a case, the digraph is analyzed and modified to ensure that all data and control paths are balanced [2][10]. The directed graph representation adopted in this design flow is well suited to formally analyze the design symmetries. It offers the opportunity to formally analyze the data-paths symmetries of the design and then balance the asymmetric data-paths if necessary. After that, we apply the path swapping method and formally verify that the structure of the circuit is still well balanced at the logical level.

Let's then apply the DPA attack on this type of circuit in order to evaluate the chip's DPA sensitivity. This starts by defining the electrical model of balanced QDI asynchronous logic.

4.3 Electrical Model of balanced QDI asynchronous logic

The electrical model of balanced QDI asynchronous circuit used in this paper is based on the model developed in [8]. It proposes a current model of QDI block implementing a fix number of logical transitions regardless of the input data. As it represents about 85% of the CMOS gate power dissipation, the paper only considers the Dynamic power dissipation (P_d) which is defined as the power required to charge and discharge the capacitive load of the gates. Hence, the block dynamic current profile is expressed by:

$$P_{dc}(t) = \sum_{i=1}^{Nc} \left[\sum_{j=1}^{N_{ij}} I_{ij}(t_i) \right] + P_{dn}(t) \quad \text{with} \quad I(t) = C\,\frac{dV}{dt} \tag{2}$$

$I_{ij}(t_i)$ represents the dynamic current dissipated by the *jth* gate of level *i* and P_{dn} is a dynamic noise function. N_c is the number of gates along the critical data-path. It represents the maximum number of gates in series in the execution path of the block and also corresponds to a number of logical level used to divide a block in N_c logical levels as illustrated in the digraph representation (figure 6). N_{ij} is the number of gates switching at each logical level (N_c). The values N_c and N_{ij} are determined by a simple analysis of the block digraph representation. *C* is the total charge of the output gate node, defined by: $C=C_l+C_{par}+C_{sc}$ in which C_l, represents the load capacitance (gate and routing capacitance), C_{par} is the parasitic capacitance, and C_{sc} is the Short-circuit equivalent capacitance. Let's again consider the block of figure 4. We deduce through the digraph exploration the values of N_c and N_{ij} : $N_c=4$; $N_{11}=N_{21}=N_{31}=N_{41}=1$.

Therefore, the block dynamic current at each phase (evaluation phase and return to zero phases) is given by:

$$P_{dc\,xor}(t) = \left(I_{1j}(t_1) + I_{2j}(t_2) + I_{3j}(t_3) + I_{41}(t_4)\right) + P_{dn}(t) \tag{3}$$

Equation (3) represents, in a first approximation, the profile of the dynamic current of the Dual-rail Xor gate.

This formal current modelling can be extended to all balanced QDI asynchronous block. Its application enables to evaluate with high accuracy the effectiveness of our new secure design approach on balanced QDI asynchronous circuits.

4.4 Applying DPA on the formal model

We have adopted the formalization proposed by Thomas S. Messerges et al. in [11] to apply DPA on this formal model. Before that, let's first review the basis of the attack. DPA attack is performed by computing *M* random values of plain-text-input (PTI_i). For each of the *M* plain-text-input, a discrete time power signal S_{ij} and cipher-text-output are collected. The index *i* of power signal S_{ij} corresponds to the PTI_i that produced the signal and the *j* index corresponds to the time of the sample. According to a DPA algorithm, the S_{ij} are split into two sets by a separating function *D*.

$$S_0 = \{S_{ij} | D = 0\} \qquad S_1 = \{S_{ij} | D = 1\} \tag{4}$$

The average power signal of each set is given by:

$$A_0[j] = \frac{1}{|m_0|} \sum_{i=1}^{n_0} S_{ij} \qquad A_1[j] = \frac{1}{|m_1|} \sum_{i=1}^{n_1} S_{ij} \tag{5}$$

Where $|m_0|$ and $|m_1|$ represent the number of power signals S_{ij} respectively in set S_0 and S_1. The DPA bias signal is obtained by:

$$T[j] = A_0[j] - A_1[j] \tag{6}$$

If the DPA bias signal shows important peaks, it means that there is a strong correlation between the *D* function and the power signal. Selecting an appropriate *D* function is then essential in order to guess a good secret key.

Let us apply this DPA attack to a balanced QDI asynchronous design without activating the path swapping technique. Choosing an XOR for the *D* function implies to analyse the electrical signature of an Xor gate [8]. Then, the average current signal of both sets of equation (5) is written as follows:

$$\begin{aligned} A_{xor\,0}[t] &= \frac{1}{2}\left(I_{11}(t_1) + I_{12}(t_1) + I_{21}(t_2) + I_{31}(t_3) + I_{41}(t_4) + I_n(t)\right) \\ A_{xor\,1}[t] &= \frac{1}{2}\left(I_{13}(t_1) + I_{14}(t_1) + I_{22}(t_2) + I_{32}(t_3) + I_{41}(t_4) + I_n(t)\right) \end{aligned} \tag{7}$$

Where $I_n(t)$ is a noise signal. The electrical signature is given by:

$$\begin{aligned} S[t] = T[t] = &\left(C_{11}\frac{dVout_{11}}{dt_{11}} + C_{12}\frac{dVout_{12}}{dt_{12}} + C_{21}\frac{dVout_{21}}{dt_{21}} + C_{31}\frac{dVout_{31}}{dt_{31}} + C_{41}\frac{dVout_{41}}{dt_{41}}\right) - \\ &\left(C_{13}\frac{dVout_{13}}{dt_{13}} + C_{14}\frac{dVout_{14}}{dt_{14}} + C_{22}\frac{dVout_{22}}{dt_{22}} + C_{32}\frac{dVout_{32}}{dt_{32}} + C_{41}\frac{dVout_{41}}{dt_{41}}\right) \end{aligned} \tag{8}$$

as $\frac{dVout_{ij}}{dt_{ij}} \cong \frac{\Delta V}{\Delta t_{ij}}$ this expression becomes

$$S[t] = \Delta V.\left(\frac{C_{11}}{\Delta t_{11}} + \frac{C_{12}}{\Delta t_{12}} - \frac{C_{13}}{\Delta t_{13}} - \frac{C_{14}}{\Delta t_{14}}\right) + \Delta V.\left(\frac{C_{21}}{\Delta t_{21}} - \frac{C_{22}}{\Delta t_{22}}\right) + \Delta V.\left(\frac{C_{31}}{\Delta t_{31}} - \frac{C_{32}}{\Delta t_{32}}\right) \tag{9}$$

Δt represents the physical time taken by the gate to charge/discharge its output node. This time also depends on the value of *C*. Recalling that $C=C_l+C_{par}+C_{sc}$.

Contrary to synchronous design where the DPA attack reveals path dissymmetry of the attacked bit (C_i), DPA on the balanced QDI asynchronous design reveals path dissymmetry of all rails that are used to encode the attacked bit. The DPA on dual-rail xor gate requires comparing the electrical behaviour of paths which compute rail C_{i0} and rail C_{i1}. As it is shown in equation 9, the main dissymmetries of such a balanced QDI structure are located on load capacitances which involve gates delay variations between their different paths. Let's now apply the same DPA attack on a dual-rail Xor gate implementing the path swapping. As all data-paths are used to compute outputs, the average current signal of each set of equation (5) contains all gates' currents of the structure. Then, its expression is given by:

$$A_{xor0}[t] = A_{xor1}[t] = \frac{1}{4}(I_{11}(t_1) + I_{12}(t_1) + I_{21}(t_2) + I_{31}(t_3) + I_{13}(t_1) + I_{14}(t_1) + I_{22}(t_2) + I_{32}(t_3) + I_{41}(t_4) + I_n(t)) \qquad \textbf{(10)}$$

This nullifies the electrical signature of the dual-rail Xor gate.

$$S[t] = \approx 0 \qquad \textbf{(11)}$$

Equations (10) and (11) clearly demonstrate that the differential power analysis on such a symmetric data-paths is completely unusable when using the path swapping method.

4.5 The swapping function

As illustrated above, implementing a DPA attack on bit encoded with *1-of-N* encoded data, means analysing the electrical difference between its *N* data-path rails. This fact enables us to reduce the number of possible permutations which are useful to implement the path swapping method. Indeed, let's consider the attacked bit C_i of a selection function *D* encoded with *1-of-N*. E_i represents the set of input values which activate the rail *i* of C_i and m_i represents the number of these input values (elements) in each set E_i. There are two possible approaches to implement the swapping function: a nondeterministic approach and a deterministic approach.

- The nondeterministic approach: in this approach the number of possible input permutations for each input element is computed by the following expression:

$$P_{PE} = \sum_{i=1}^{N} m_i \qquad \textit{P}_{PE}\textit{: number of possible permutation of one input element} \qquad \textbf{(12)}$$

This number highlights two points: first, the elements of the same set E_i can be permuted between them and second, this approach requires for each input element A_i of E_i ($A_i \in E_i$), the use of P_{PE} P_{PE}*-to-1* multiplexers (P_{PE} inputs and 1 output). Hence, the number and the type of multiplexers required for the bit C_i is given by:

$$N_{C_i} = P_{PE}^{\ 2} \quad (P_{PE} - to - 1) \ \textit{Multiplexers} \qquad \textbf{(13)}$$

This number can be reduced if the permutations inside each set E_i are proscribed:

$$N_{C_i} = \sum_{i=1}^{N} \left[m_i \quad \left((\sum_{j=1;j\neq i}^{N} m_j + 1) \to 1 \right) \textit{Multiplexers} \right] \qquad \textbf{(14)}$$

If all m_i are equal (each set E_i has the same number of elements), then:

$$N_{C_i} = P_{PE} \quad (P_{PE} - m_i + 1) \to 1 \ \textit{Multiplexers} \qquad \textbf{(15)}$$

For example, if *N=2* and sets E_0 and E_1 have respectively $m_0=2$ and $m_1=2$ as shown in figure 4, we obtain 4 4-to-1 multiplexers which can be reduced to 4 3-to-1 multiplexers.

• The deterministic approach: the goal of this approach is to constrain the permutation function in order to optimize the use of multiplexers and to guarantee the security. The idea here is to permute one element of set E_i with one element of each of the other sets. Therefore, each element can be permuted N times (as we have N sets) and it requires for the bit C_i, P_{PE} *N-to-1* multiplexers:

$$N_{C_i} = P_{PE} \quad N - to - 1 \ Multiplexers \tag{16}$$

Considering the previous example, we obtain 4 2-to-1 multiplexers.

Even if the swapping function is known, it does not affect the efficiency of the approach because it is randomly executed. In fact, the choice of data-path used to process the data remains random. This point enables us to considerably optimize the use of multiplexers. In addition to this, some optimizations can be applied according to the regularity and the symmetry of the architecture. It is not necessary to implement multiplexers with each block of the architecture (see the case study on paragraph 5). These analyses are also available when using some demultiplexers and can be extended on all data-paths.

4.6 Discussion

Nevertheless, the security brought by this new design approach is fully efficient if and only if these two conditions are fulfilled:
* *Randomizing the path swapping*: the objective is to ensure unpredictable apportionment of path swapping inside a block. The attack is still possible if the hacker knows the random function generator. Indeed, the analysis can be focused on set of data that are processed by the same random value. For example, if the random function is always switching between two cases (case 1 and 2 as described in figure 4). Performing the attack on the first case is equivalent to attack a balanced QDI asynchronous circuit (without path swapping). The situation is the same if one output rail of the bit attacked is always computed in the same data-path. Then, the random generator must be an unpredictable and equiprobable function. The implementation of such a random function is out of the scope of this paper.
* *Implementing multiplexers and demultiplexers in the architecture*. One way to break the random function generator is to apply DPA attack on these blocks. For example, if the multiplexer of channel *Ai* (encoding with two rails: *Ai0* and *Ai1*) presents a significant signature when its rails are swapped, the random function which controls this multiplexer can be predicted. A particular care must be done when implementing these functions [12].

5 Case Study: DES crypto-processor

A chosen example to validate this design approach is a DES algorithm. The asynchronous DES crypto-processor is implemented using a four-phase protocol, *1-of-N* encoded data and balanced data-paths. The architecture used is an iterative structure, based on three self-timed loops synchronized through communicating channels: one loop for the ciphering data-path, the second loop for the key data-path and the third one for the control data-path (a finite state machine) which controls the data-paths along its sixteen iterations (figure 10).

The implementation of multiplexers/Demultiplexers in each block of the architecture could significantly increase the chip's area. This can be done efficiently by taking advantage of the implemented algorithm. As the DES algorithm uses only

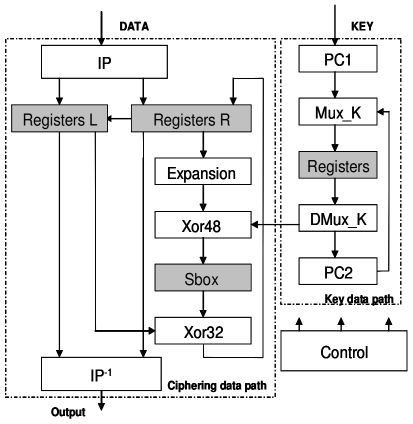


Fig. 8. DES architecture

Table 1. A new ordering of the SBOX1

Output values	Input values				Sets
0	14/0	0/1	15/2	13/3	E0
15	5/0	1/1	8/2	0/3	
1	3/0	7/1	1/2	6/3	E1
14	0/0	4/1	2/2	11/3	
2	4/0	5/1	6/2	3/3	E2
13	2/0	6/1	4/2	15/3	
3	8/0	14/1	12/2	10/3	E3
12	11/0	10/1	9/2	1/3	
4	1/0	3/1	0/2	4/3	E4
11	6/0	11/1	7/2	9/3	
5	12/0	13/1	14/2	8/3	E5
10	9/0	8/1	13/2	12/3	
6	10/0	9/1	5/2	14/3	E6
9	13/0	12/1	13/2	12/3	
7	15/0	2/1	11/2	7/3	E7
8	7/0	15/1	3/2	2/3	

Column number

$\downarrow$

Cx / Rx

$\downarrow$

Row number

four simple types of functions (permutation, Xor, Substitution, Expansion and reduction functions), we only need to implement Multiplexers/Demultiplexers on registers and on the Substitution box (blocks in bold on figure 8). The Substitution function (SBOX) is selected because it is a surjective function (irreversible function).

Indeed, because it is a one way function, it is difficult to trace the information when its inputs/outputs are permuted. Each Substitution Box (SBOX) of the DES algorithm receives 6 bits (64 possible values) in their inputs and generates 4 bits on their outputs (16 possible values), so that, one output value can be selected by 4 different input values [13]. With the dual-rail encoding of the data for each SBOX, we have 8 sets (output rails) of 32 input elements. Applying a nondeterministic permutation, leads to implement 64 64-to-1 multiplexers which is not efficient in terms of area. We used a deterministic implementation exploiting the maximum redundancy of the Substitution function. Let's consider the first substitution box of the DES algorithm. To be able to efficiently swap data-path rails of the SBOX1, we gathered in the same set all input values which generate an output value and its opposite value. For example all input values which generate the output value '0' and its inverse output 'F' are gathered in the same set E_0 as illustrated in table 1.

This representation enables us to observe that, it is only possible to permute in each set, the input values which have the same row number. Therefore it requires 32 2-to-1 multiplexers which increases the SBOX1 area by 30%.

6 Validations: Electrical Simulations

The technology used for implementing the design is the HCMOS9 (0.13µm) from STMicroelectronics. All electrical simulations are performed with *Nanosim* with an asynchronous DES gate Netlist.

The electrical simulation offers the possibility to analyze without disturbing signals (noise), the electrical behaviour of the design with more details. Hence, the number of necessary messages (M) is minimal. In order to easily evaluate the relevancy of this new countermeasure, the path swapping method is only implemented in the first Substitution function (SBOX1) and to four bits of Register L (figure 8). These four bits are combined with the output bits of the first Substitution function (SBOX1) by an Xor function in block Xor32. To reproduce the effects of back end steps during simulations, a dissymmetry is introduced between rails which compute the fourth bit of the SBOX1. In fact, the load capacitance (*C*) of the first rail ($S_4(0)$) is set to 32 femto-farads. This value includes the gate, the routing, the parasitic and the short-circuit capacitances. It has been estimated after a pre-place and route step with Silicon Ensemble. The defined *D* function for processing the attack is as follows:

$$D(C_4,P_6,K_0)= SBOX1(P_6 \oplus K_0)$$

The attack is done on the fourth bit of the SBOX1 with 64 curves (64 plain-text-inputs). As a reference, the attacks were realized without activating the countermeasure by switching off the random number generator. The results of the attack are displayed in figure 9. The DPA bias signal (*S*) is clearly observable when the correct key is guessed.

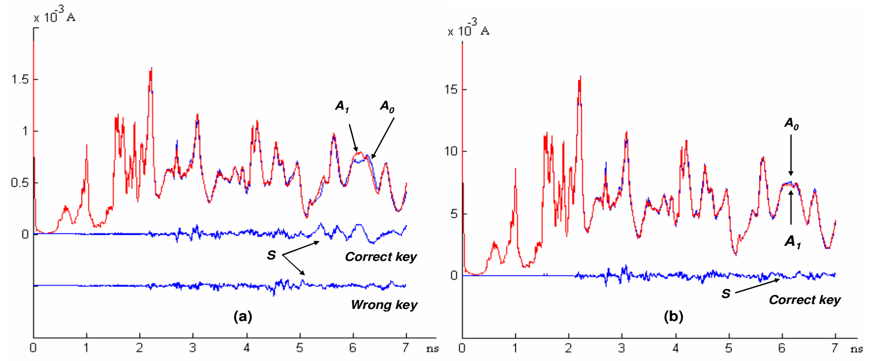


Fig 9. Electrical Signature when performing DPA attack on bit 4 of the SBOX1. Loading charge difference of both rails of this bit is 32 femtoF. Only the first round is considered. (a) – path swapping is not activated (b) – path swapping is activated

The result of the attack when the countermeasure is activated is illustrated in figure 9-b. The DPA bias signal is completely removed as predicted by the equation (11).

All results present in this paragraph demonstrated the relevancy of using the path swapping method on QDI asynchronous circuits which have their data-paths balanced and symmetric.

7. Conclusion

This paper presented a new design technique for enhancing QDI asynchronous circuits' resistance against DPA attack. This design approach which is called Path Swapping exploits all properties of QDI asynchronous logic which are suited to design secure chips, particularly the logical data-path symmetries.

The results obtained from electrical simulations of a DES crypto-processor proved the efficiency of the Path Swapping method in terms of DPA resistance. Current works are focused on the realization of a prototype in order to perform analysis on silicon.

References

[1] P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," Advances in Cryptology - Crypto 99 Proceedings, LNCS Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.

[2] Simon Moore, R. Anderson, P. Cunningham, R. Mullins, G.Taylor, "Improving Smart Card Security using Self-timed Circuits", Eighth International Symposium on Asynchronous Circuits and systems (ASYNC2002). 8-11 April 2002. Manchester, U.K.

[3] L. A. Plana, P. A. Riocreux, W. J. Bainbridge, A. Bardsley, J. D. Garside and S. Temple, "SPA - A Synthesisable Amulet Core for Smartcard Applications", Proceedings of the Eighth International Symposium on Asynchronous Circuits and Systems (ASYNC 2002). Pages 201-210. Manchester, 8-11/04/2002. Published by the IEEE Computer Society.

[4] J. Joshua, P. Kocher, J. Benjamin, Balanced Cryptographic computational method and apparatus for leak minimization in smartcards and others Cryptosystems, EP1088295/WO9967766.

[5] Danil Sokolov, Julian Murphy, Alex Bystrov and Alex Yakovlev,"Improving the Security of Dual-Rail Circuits", CHES 2004, LNCS 3156, pp 282-297, 2004.

[6] J. J. A Fournier, Simon Moore, Huiyun Li, Robert Mullins, and Gerorge Taylor,"Security Evaluation of Asynchronous Circuits", CHES 2003, LNCS 2779, pp 137-151, 2003.

[7] F. Bouesse, M. Renaudin, B. Robisson, E Beigne, P.Y. Liardet, S. Prevosto, J. Sonzogni, "DPA on Quasi Delay Insensitive Asynchronous circuits: Concrete Results", DCIS 2004 Bordeaux, France, November 24-26, 2004.

[8] G.F. Bouesse, M. Renaudin, S. Dumont, F. Germain, « DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement », DATE 2005, Munich, p.424.

[9] Marc Renaudin, "Asynchronous circuits and systems: a promising design alternative", Microelectronic for Telecommunications : managing high complexity and mobility" (MIGAS 2000), special issue of the Microelectronics-Engineering Journal, Elsevier Science, Vol. 54, N° 1-2, December 2000, pp. 133-149.

[10] F. Bouesse, M. Renaudin, F. Germain, "Asynchronous AES Crypto-processor Including Secured and Optimized Blocks", the Journal of Integrated Circuits and Systems (JICS), Volume 1, ISSN 1807-1953,March 2004.

[11] T. S. Messerges and E. A. Dabbish, R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards", USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10-11, 1999.

[12] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "Static Implementation of QDI Asynchronous Primitives", 13th International Workshop on Power and Timing Modeling, Optimization and Simulations, PATMOS2003.

[13] NIST, Data Encryption Standard (DES), FIPS PUB 46-2.