

# A Design Methodology for Secured ICs Using Dynamic Current Mode Logic

Mace F.\*, Standaert F.-X., Quisquater J.-J., Legat J.-D.

UCL Crypto Group

Microelectronics Laboratory

Universite Catholique de Louvain

**Abstract.** This paper considers the security of Integrated Circuits (IC's) against power analysis attacks. We present a methodology for the secure design of cryptographic IC's and apply it to the Dynamic Current Mode Logic (DyCML). For this purpose, we first propose to use a Binary Decision Diagram (BDD) approach in order to predict the power consumption of certain critical gates of an encryption algorithm. Then, we validate our model using SPICE simulations. Based on these predictions, we demonstrate that it is only possible to mount power analysis attacks against certain instances of DyCML circuits. We also illustrate that the attack efficiency depends on the quality of the power consumption models, themselves depending on the representation chosen for the targeted boolean function. Relying on these simulation results, we finally propose a complete methodology to (1) find the most secure instance of a boolean function and (2) derive secure ICs from these optimized structures.

**Keywords.** Differential Pull Down Networks, Binary Decision Diagrams, Differential Power Analysis, Side-channel attack.

## 1 Introduction

Cryptographic electronic devices such as smart cards, FPGAs or ASIC's are taking an increasing importance to ensure the security of data storage and transmission. However, they are under the threat of attacks taking advantage of side-channel leakages caused by the physical implementation of any given algorithm. Among these leakages, the use of power consumption proved to be very efficient to recover information about the data handled by any given circuit [1].

To prevent the attacker from efficiently using this information, different countermeasures were proposed. For example, random process interrupts or dummy instructions were used to avoid the sequential execution of the algorithm, but were shown to be inefficient in [2]. Random noise addition was also suggested but does not provide any fundamental countermeasure as the signal is still present and can be recovered by statistical analysis. Masking methods [3] that consist in masking the data with random boolean values are yet another proposal, this time acting at the algorithmic level, but they still leak some information and reduce the implementation efficiency of the algorithm.

Interesting alternatives were presented and proposed to tackle the problem directly at the transistor level, by using specific logic styles. Their purpose is to decorrelate the power consumption of a circuit from the data it handles by obtaining steady activity and power consumption. In [7] and [8], it was proposed to

---

\* This work was supported by the FRIA grant of the FNRS Belgium

use particular dynamic and differential logic styles which achieve a very regular power consumption. Even if they were not able to totally suppress the power consumption variations relative to the data handled, and thus do not provide a theoretical countermeasure against power analysis attacks, nevertheless they help to make the attack significantly harder and weaken its efficiency.

We try here to propose a particular model for the power consumption of such logic styles, and more specifically, we apply our model and predictions to the Dynamic Current Mode Logic. Indeed, to obtain high efficiency attacks, the power consumption behavior of particular implementations of algorithms must be predicted with a good accuracy. So far, predictions were efficiently made on CMOS implementations, using a model based on the number of switching events occurring within the circuits. But, because of their different power consumption behavior, this model is unapplicable to dynamic and differential logic styles [7].

In [12], Binary Decision Diagrams were used as a tool to optimize the design of these logic networks. We show here how these graphs can also be used to predict the power consumption of such gates and, afterwards, choose the structure that achieves the best resistance against power analysis attacks.

This paper is structured as follows. In section 2 we give a short introduction to Binary Decision Diagrams. Next, in section 3, we present some dynamic and differential logic styles and the interest of Dynamic Current Mode Logic. Afterwards, we propose our power consumption behavior model in section 4. Section 5 presents the experiments (validation of the power consumption model, use of this model to mount power analysis attacks) and the achieved results (choice of the most secured implementation) and we conclude in section 6.

## 2 Binary Decision Diagrams

### 2.1 Structure and principles

Binary Decision Diagrams were firstly introduced by Akers in [4] as a method to define, represent, analyze, test and implement large digital functions. These graphs can be easily developed using the Shannon expansion theorem from which a boolean function can be recursively divided into other functions, each one depending on all input variables but one. The graph structure of a function  $f$  is thus obtained by applying the Shannon expansion recursively for each input, until the function to expand is the constant function 1 or 0.

Using this representation for boolean functions allows their easy manipulation and easy combinations between them. Among the abundant literature presented on BDDs, Bryant proposed in [5] to refine their representation and algorithms used to manipulate them. The basic representation of boolean functions by the mean of such graphs was defined as follows.

A function graph is a rooted, directed, acyclic graph with a vertex set  $V$  containing two types of vertices. The attributes of a nonterminal vertex  $v$  are a particular input  $x_i$  ( $i \in \{0, \dots, m - 1\}$ ) of the implemented boolean function

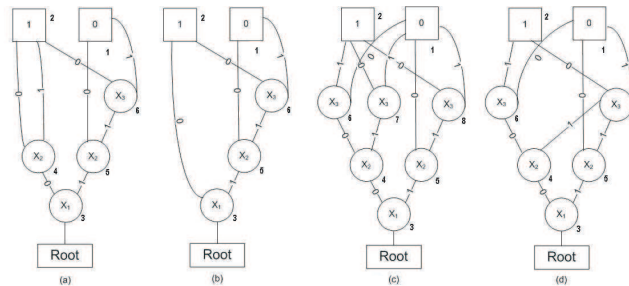
( $f : \{0, 1\}^m \rightarrow \{0, 1\}$ ), an argument index  $index(v) \in \{1, \dots, n\}$  and two children vertices  $low(v)$  and  $high(v)$ . And the ones of a terminal vertex  $v$  are a particular index value  $index(v)$  and a value  $value(v) \in \{0, 1\}$ , representing the function's result. Moreover, if a child vertex of a nonterminal vertex ( $low(v)$  or  $high(v)$ ) is nonterminal, its index should be higher than the one of its parent.

The BDD can be reduced from its first structure based on Shannon expansion. As a matter of fact, two conditions allow a vertex to be removed from the graph or replaced by another vertex, leading to a different structure representing the same function.

- (1) If the two child vertices of a vertex  $v$  are identical ( $low(v) = high(v)$ ), then vertex  $v$  should be removed from the graph and its mother should receive the child of  $v$  in replacement of it.
- (2) The replacement of one or more vertex by a particular one occurs when these two or more vertices define the exact same function as the particular vertex. Then, all the duplicated vertices and all their children should be removed from the graph and replaced by the particular one.

These two principles are shown in figure 1. In this figure, we have represented non-terminal vertices by a circle and terminal vertices by a square.

Sub-figures 1-a and -b illustrate the suppression of one vertex having identical children (the vertex with index value 4), while sub-Figures -c and -d show the fusion of two vertices implementing the same function (vertices 7 and 8). The



**Fig. 1.** Reduction of binary decision diagrams

BDD can also be modified, by changing the data ordering, in order to minimize its number of vertices, and thus reduce the mean path length from the root vertex to the terminal ones. A data ordering, or input ordering, corresponds to the sequence in which the inputs are considered to apply the Shannon's expansion.

## 2.2 Use of BDDs

Binary Decision Diagrams are often used to find minimal representations of boolean functions and manipulate them very efficiently [5], using algorithms like the ITE (If Then Else) algorithm to combine different functions (and thus different BDDs). In the same way, those algorithms have been used to build logic op-

timization systems capable to handle very large circuits with high performances in term of runtime [6].

### 3 Dynamic and Differential Logic Styles

#### 3.1 Their use in cryptographic hardware

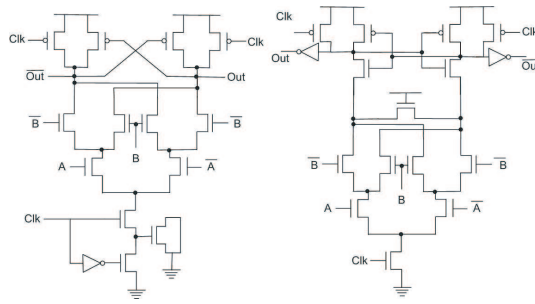
In [7], Tiri *et al.* showed that dynamic and differential logic styles can be used to achieve a power consumption independent from the data handled. In first approximation, the power consumption behavior of CMOS leaks information because it consumes power only for output transitions. On the basis of this model, they recommended to use a dynamic and differential logic style to produce a uniform switching activity and thus ensure a power consumption event for each evaluation cycle, whatever the inputs. They also showed that, in order to really balance the power consumption, we should use a dynamic and differential logic style that achieves a power consumption being the most possible constant. This can be obtained discharging the whole internal capacitance of the gate. It is why they developed Sense Amplifier Based Logic (SABL). The advantage of this logic style is that, by discharging the whole internal capacitance, it produces a power consumption with reduced variations in function of the inputs. Its main drawback is that it yields to a high power consumption.

To overcome this problem, it was proposed in [8] to use the Dynamic Current Mode Logic (DyCML) developed by M. Allam *et al.* [9]. It achieves the required steady power consumption with better performances (in terms of power consumption and delay). Thanks to its dynamic current source, DyCML gates produce a low output swing of which the value is a function of the total (intrinsic and extrinsic) output load capacitance and of the size of one transistor acting like a virtual ground.

In [8], it has been shown that, even if the two implementations of circuits present the same high security margins, according to criterions defined in [7], DyCML was recommended because of its better performances. Moreover, DyCML gates can be directly connected through an asynchronous scheme, which is usually considered to improve the security of implementations of cryptographic algorithms [10]. As an illustration, we give the structure of 2 inputs XOR gates implemented in both DyCML and SABL logic style in Figure 2. As SABL needs a domino-interconnect structure between gates to counteract the charge sharing effect, we presented the gate with the needed output inverters.

#### 3.2 Gate design

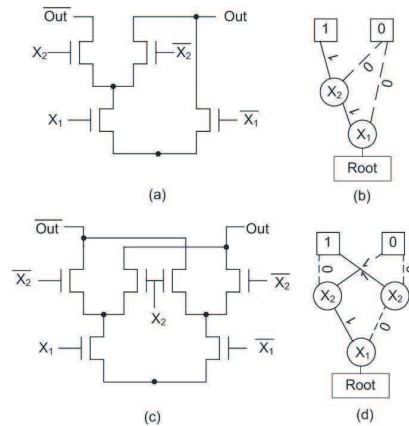
One important step, while designing dynamic and differential gates, consists in deriving an efficient structure for the part of the gate that effectively computes the implemented function. Depending on the type of logic style used, this can be done using several methods like using the Karnaugh Map (K-Map Procedure) or the tabular methods proposed in [11]. However, these methods do not always



**Fig. 2.** Structure of XOR gate: a- DyCML, b- SABL

lead to a structure using differential pairs (Differential Pull Down Network - DPDN ) like it is needed for DyCML or SABL.

A method proposed in [12] exploits the isomorphism existing between DPDN and the BDD implementing the function. A boolean function implemented with a DPDN is computed by the mean of logical switches, which determines which one of the two outputs of the gate will be discharged. Each logical switch is composed of a transistor pair, with connected sources. One of the transistor pair's gate is driven by the logical input related to the vertex and the other one by the complement of this logical input. The drain of the transistors are connected to the sources of particular switches of the structure. The isomorphism is clear to see if you assimilate a switch of the DPDN to a particular non-terminal vertex of the BDD. To illustrate this, figure 3 gives the implementation of two basic functions (AND/NAND and XOR/XNOR) with the corresponding BDD. This isomorphism between the structures of the graph and the DPDN can be used to design very efficient DPDN with optimization of their performances in terms of area, delay [12] and power consumption.



**Fig. 3.** Isomorphism between DPDN and BDD for functions AND (a-b) and XOR(c-d)

#### 4 BDD based tool

This isomorphism gave us the idea to use the BDD not only as a tool of optimization of classical performances of the gates, but also as a tool to predict its

power consumption and the most secure structure among the ones implemented in the gate.

Indeed, we are wondering whether this graph could not be used to determine the structure having the lowest leakage of information.

In order to do so, we proposed to model the power consumption relative to one single input sequence using the following assumptions. Firstly, we consider that variations in the power consumptions are caused by variations of the number of internal capacitances within the DPDN that are charged/discharged at each evaluation cycle, like it was proposed in [7]. Secondly, we focus our interest on the diffusion capacitances of the transistors. The third hypothesis we make is to consider that drain-gate capacitances and source-drain capacitances are equal and the only ones to play a role. A precise modeling of their value is complex because it evolves along the time, depending on the voltages at the source, drain and gate and on the form of their evolution. It is why to predict the power consumption relative to a particular input sequence, we only counted the number of normalized capacitances that were *activated* (by this, we mean effectively connected to the discharged output of the gate), considering each one equal.

To achieve rapid predictions for all possible implementations (as we show in the experiments, the considered boolean functions are defined as following  $f : \{0, 1\}^4 \rightarrow \{0, 1\}^4$ ), we designed a program taking an hexadecimal truth table for input and producing several outputs (these outputs are produced for each 1-bit function obtained considering one 4-bit function as the combination of four 1-bit functions  $\{0, 1\}^4 \rightarrow \{0, 1\}$ , as a 1-bit function is implemented in a single gate):

- The graph structures for each input ordering
- The Spice netlist relative to the transistor implementation of the graphs
- The predictions of the connected capacitances for each input sequence, for each graph

Indeed, for a function defined on  $\{0, 1\}^4 \rightarrow \{0, 1\}$ , there are  $4! = 24$  possible input orderings in such a graph. The applied procedure consisted thus in building the graphs corresponding to the different input orderings, reducing them, determining the discharged output node for each input sequence and finding the total number of connected normalized capacitances for each particular input sequence. The number of normalized capacitances connected to one vertex has three contributors:

- 2 capacitances for the 2 source-gate capacitances of the transistors forming the switch associated to the vertex.
- As many capacitances as there are mother vertices to this vertex (drain-gate capacitances of the transistors of the switches connected to this vertex).
- All capacitances contributions due to the connected vertices in function of the input sequence.

## 5 Experiments

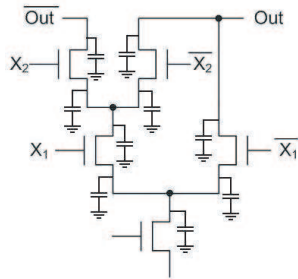
The different experiments involved power consumption extractions carried out using SPICE simulations with a  $0.13\mu\text{m}$  PD-SOI (Partially Depleted Silicon-On-

Insulator) technology, with a power supply of 1.2V. DyCML gates were designed with an output swing of 0.8V and loads for the gate corresponding to one active transistor. We extracted the power consumption by averaging the power consumption of each single gate on the precharge phase following the evaluation corresponding to the input sequence.

### 5.1 Validating the power consumption model

To achieve the validation of our power consumption model, we use the correlation coefficient to measure how our predictions compare to the actual Spice simulations of the power consumption. We selected the correlation coefficient because it is a usual tool to mount practical power analysis attacks [13]. However, other statistical tools could be considered.

We firstly validated our model on a simple 2-inputs DyCML AND gate. The predictions, structure and normalized connected capacitances of the gate are summarized in figure 4 and table 1. Exhibiting a correlation value of 0.9696, the model and predictions match pretty well. We then evaluated the validity of our



$x_2$	$x_1$	f	Discharged Node	Connected Capacitances
0	0	0	$\overline{OUT}$	8
0	1	0	$\overline{OUT}$	8
1	0	0	$\overline{OUT}$	5
1	1	1	OUT	7

**Fig. 4.** Parasitic Capacitances of a AND gate. **Table 1.** Prediction features in function of the input sequence.

model on more complex gates. To do so, we used the function P and Q of a Khazad Sbox [14]. These functions are defined on  $GF(2^4)$ . We thus determined the eight 1-bit functions because each 1-bit function can be implemented in a single gate. For each of these 1-bit functions there are 24 possible implementations (input orderings). We then correlated these predictions and simulated power consumptions to finally obtain an average correlation of 0.8387 and a mean standard deviation of the correlation of 0.1433 between the power consumption and its prediction.

### 5.2 Simulated Attacks

We then evaluated the utility of such a model in the context of power analysis attacks. To do so, we mounted simulated attacks using the predictions of connected capacitances and the simulations of the power consumptions. We ran these on a single Sbox for which the inputs resulted in a XOR operation between plaintexts and a key.

We computed the total power consumption  $M_i$  of the Sbox, for a plaintext  $i$ .

The second step of this simulated attack consisted in predicting the number of connected capacitances  $P_{k,i}$  using guesses of the inputs obtained by realizing a XOR operation between each possible key  $k$  and the plaintext  $i$ . The last step consisted in calculating the correlation between the simulated power consumption  $M$  and the predictions  $P_k$  for a key  $k$ .

We used the data collected after the SPICE simulations for the measurements and, using this attack methodology, we extracted the correlation after the encryption of a number of plaintexts varying from 1 to 256, and for all the possible keys. We extracted the value of this correlation and also correlation margins for 256 encrypted plaintexts. The correlation margin 1 is defined here as the difference in correlation between the right key guess and the wrong key guess having the highest correlation, while correlation margin 2 is the difference between the correlation for the right key guess and the mean correlation of the wrong ones. This was done for 4 particular implementations of the Sbox:

- the one with the total simulated power of each function P and Q having the largest variance (Implementation 1)
- the one with the total simulated power of each function P and Q having the smallest variance(Implementation 2)
- the one for which the implementation of each 1 bit function is the most correlated with the power consumption prediction (Implementation 3)
- the one for which the implementation of each 1 bit function is the less correlated with the power consumption prediction (Implementation 4).

The simulated attack results are given in table 2. In this table, we give the values of both correlation for the right key guess and correlation margin, for each implementation. A negative value for the correlation margin 1 corresponds to an unsuccessful attack. We also give the mean number of encryptions needed to achieve a discrimination between the right key guess and the wrong ones.

Implementation	Correlation	Margin 1	Margin 2	# of texts
1	0.8393	0.4401	0.8429	15
2	0.6186	0.2855	0.6210	40
3	0.9199	0.5305	0.9235	7
4	0.3462	-0.0037	0.3469	N.A.

**Table 2.** Simulated Attack Results

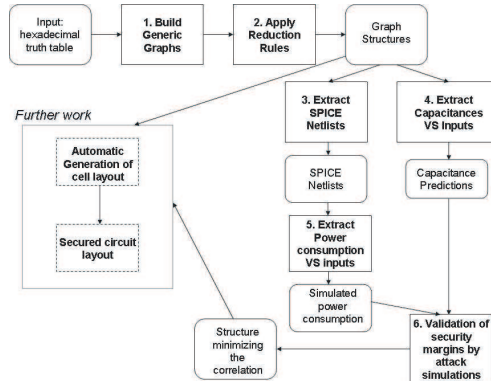
As we can see, the first three implementations were successfully attacked, as they all had a positive correlation margin. We can also see that they all have high correlation and sufficient correlation margin to allow successful attacks, while for the fourth implementation, 2 key guesses (the right one and a wrong one) remained correlated to the key, leading to a negative value of the correlation margin and preventing us from being able to obtain the right key guess. The difference in correlation between the right key guess and the second most correlated wrong one is 0.0453.

### 5.3 Discussion

It is quite obvious that the efficiency of a power analysis is dependent of the obtained correlation for the right key guess and on the difference in correlation

between this right key guess and the wrong ones. Moreover, as it was presented previously, these values are dependent of the power consumption model used to mount the attack and on the chosen algorithm.

These simulated attack results thus clearly emphasize the possibility of choosing the implementation for which the power consumption is the harder to precisely model, and thus being the most resistant towards a power analysis attack based on this model. Indeed, correlation and correlation margin can be significantly decreased in comparison to the other implementations, while the number of cleartexts needed is highly dependent on the predictability of the chosen implementation.



**Fig. 5.** Proposed methodology for the design of secured implementation of cryptographic functions

With all the tools, we get the possibility to automatically choose the implementation of a circuit that limits at the maximum the side channel information leakage of a circuit. Moreover, as suggested in [12], the graph structure generated by the tool can help for automated cell layout generation. This was done in [15], for some non-complementary digital VLSI cells for which transistor placement can be efficiently achieved from a graph structure describing the cell, applying adapted selection algorithms to choose the best structure. The adjunction of all these concepts yields to a design methodology that can be described like in Figure 5.

## 6 Conclusion

In this paper, we investigated a model for the power consumption of Dynamic and Differential Logic styles. These circuits have previously been proved to offer a good alternative to CMOS in terms of security against side-channel attacks. However, their security against such attacks highly depends on the impossibility to efficiently predict their power consumption: a previously uninvestigated problem.

As a main result, we demonstrated that for one specific logic style, namely the DyCML, it is possible to derive a simple and efficient leakage model and build practical attacks on this basis. Then, we showed that it is possible to search exhaustively among the possible representations of a logic function, in order to find

the hardest to predict one. In other words, we looked for the structures offering the best resistance against power analysis. Confirmed by simulated experiments, we finally illustrated that the best resulting implementations allow an improved security, as the original power analysis attacks could not be applied anymore. Remark that, even if it is not discussed in the paper, such circuits also exhibit lower power consumption variances and are consequently harder to measure.

An open question is to know whether and how the methodology presented can be generalized to other DDLs. The improvement of the power consumption models is another concern. Finally, we plan to perform practical experiments in order to evaluate how our simulation-based conclusions relate to practice.

## References

1. P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, In *Advances in Cryptology - CRYPTO'99*, LNCS, vol 1666, pp 388-397, Springer, 1999
2. C. Karlof, D. Wagner, Hidden Markov Model Cryptanalysis, In *CHES'2003*, LNCS, vol 2779, pp 17-30, Springer, 2003 -
3. T.S. Messerges, Using second-Order Power Analysis to Attack DPA Resistant Software, In *CHES'1999*, LNCS, vol 1965, pp 71-77, Springer, 1999
4. S.B. Akers, Binary Decision Diagrams, *IEEE Transactions on Computers*, vol. C-27, No. 6, pp.509-516, June 1978
5. R.E. Bryant, Graph Based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers*, vol. C-35, No. 8, pp. 677-691, August 1986
6. C. Yang, M. Ciesielski, BDS: A BDD-Based Logic Optimization System, In *ACM/IEEE Design Automation Conference - DAC'2000*, pp. 92-97, June 2000
7. K. Tiri, M. Akmal, I. Verbauwhede, A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards, In *The Proceedings of ESSCIRC 2002*, pp.403-406
8. F. Mace, F.-X. Standaert, I. Hassoune, J.-D. Legat, J.-J. Quisquater, A Dynamic Current Mode Logic to Counteract Power Analysis Attacks, In *The Proceedings of DCIS 2004*, pp.186-191
9. M.W. Allam, M.I. Elmasry, Dynamic Current Mode Logic (DyCML): A New Low-Power High-Performances Logic Styles, *IEEE Journal of Solid-State Circuits*, vol. 36, No. 3, pp. 550-558, March 2001
10. J. J.A. Fournier, S. Moore, H. Li, R. Mullins, G. Taylor, Security Evaluation of Asynchronous Circuits, In *CHES'2003*, LNCS, vol 2779, pp 137-151, Springer, 2003
11. K.M. Chu, D.I. Pulfrey, Design Procedures for Differential Cascode Voltage Switch Circuits, *IEEE Journal of Solid State Circuits*, vol. SC-21, no. 6, pp. 1082-1087, December 1986
12. J. Cortadella, Mapping BDDs into DCVSL gates, UPC/DAC Technical Report No. RR 95/04, February 1995
13. E. Brier, C. Clavier and F. Olivier, Optimal Statistical Power Analysis, *IACR e-print archive 2003/152*, <http://eprint.iacr.org>, 2003
14. P. Barreto, R. Rijmen, The KHAZAD Legacy-Level Block Cypher, NESSIE Project Home Page, <https://www.cosic.esat.kuleuven.ac.be/nessie>, 2001
15. M. A. Riepe, K. A. Sakallah, Transistor Placement for Noncomplementary Digital VLSI Cell Synthesis, *ACM Transactions on Design Automation of Electronic Systems*, vol. 8, no. 1, pp. 81-107, January 2003.