Intégration de Logique Reconfigurable dans les Circuits Sécurisés

Nicolas Valette

UNIVERSITE MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

T H E S E

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE MONTPELLIER II

Discipline : Microélectronique

Formation Doctorale : Systèmes Automatiques et Microélectroniques

Ecole Doctorale : Information, Structure et Systèmes

présentée et soutenue publiquement

par

Nicolas Valette

le 6 Mai 2008

Titre :

INTÉGRATION DE LOGIQUE RECONFIGURABLE DANS LES CIRCUITS SÉCURISÉS

JURY

M.	Viktor Fischer	, Rapporteur
M.	Guy Gogniat	, Rapporteur
M.	Lionel Torres	, Directeur de thèse
M.	Gilles Sassatelli	, Codirecteur de thèse
M.	Bruno Rouzeyre	, Examinateur
M.	Frédéric Bancel	, Examinateur
M.	Pierre-Yvan Liardet	, Examinateur

Table des matières

Liste des figures

Liste des tableaux

CONTEXTE ET MOTIVATIONS

Les questions de sécurité sont de plus en plus présentes dans notre société actuelle. Avec le développement de l'ère numérique, la sécurité est souvent accompagnée de l'utilisation de circuits intégrés sécurisés. En effet, c'est sur ces composants que repose bien souvent le contrôle d'accès à des services ou bien la protection d'informations confidentielles. L'illustration la plus parlante est la façon dont les cartes à puce envahissent notre quotidien. Qui ne possède pas de carte bancaire, de carte *SIM* (pour *Subscriber Identity Module*) ou de carte de santé ; sans parler des cartes de transport, carte d'accès et carte de télévision à péage ou les cartes téléphoniques prépayées. De plus, les passeports numériques, ainsi que les nouvelles cartes de santé ou encore les circuits *TPM* (pour *Trusted Platform Module* [TPM07]) intégrés dans les ordinateurs de dernières générations confirment cette tendance. Les circuits sécurisés sont devenus incontournables et la liste des domaines d'application énoncés ci-dessus est loin d'être exhaustive.

Par ailleurs, la plupart de ces composants sont utilisés de façon tout à fait transparente par les utilisateurs finaux. Ces derniers ne se soucient pas de connaître toute la complexité de ces circuits. En revanche, certaines personnes mal intentionnées tentent d'accéder aux données confidentielles contenues dans les cartes à puces ou encore d'usurper une identité. C'est contre ces personnes que les concepteurs de circuits sécurisés tentent de se protéger. Pour cela les industriels investissent des sommes colossales dans la recherche de nouveaux principes toujours plus perfectionnés afin de rendre leurs circuits encore plus inviolables car les fraudeurs font sans cesse preuve d'imagination pour parvenir à leurs fins.

De nouvelles techniques arrivent maintenant à maturité et commencent à représenter un réel intérêt pour les industriels afin de mieux lutter contre certaines attaques. C'est notamment le cas des circuits reconfigurables, encore déclinés sous forme de blocs reconfigurables intégrés dans des SoC (pour *System On Chip*).

Dans ce travail de thèse, nous avons étudié cette nouvelle technologie qu'est la logique reconfigurable et proposé des solutions pour l'utiliser dans le domaine des circuits sécurisés. Cette large problématique d'amélioration de la sécurité grâce à

l'intégration de logique reconfigurable intéresse autant le milieu scientifique académique que le milieu industriel. Pour cette raison, cette thèse s'inscrit dans le cadre d'une collaboration entre un industriel et un laboratoire de recherche scientifique.

Plus particulièrement, ces travaux ont intéressé l'équipe de Product Development, de la division DSA (pour *Digital Secure Access*) faisant partie du groupe MMS (pour *Microcontrollers, Memories and Smartcards*). Ces travaux ont été en grande partie réalisés au sein du site de Rousset (Bouches du Rhône) de la société STMicroelectronics.

C'est dans ce contexte qu'a été défini le sujet de cette thèse dont l'objectif est de proposer une solution pour déjouer les attaques menées contre les circuits sécurisés en investiguant les nouvelles voies offertes par les technologies reconfigurables arrivant à maturité. Dans ce manuscrit, nous pouvons distinguer deux investigations distinctes. La première consistera à contrer les problèmes de clonage et de retro-ingénierie (ou *reverse-engineering*) et sera traitée essentiellement dans le deuxième chapitre. La deuxième orientation consistera à proposer des solutions pour contrer les attaques physiques non invasives, plus particulièrement la DPA (pour *Differential Power Analysis*). Ce point sera traité dans les deux derniers chapitres. Enfin, notons que dans ce mémoire nous nous focaliserons sur l'aspect matériel des circuits sécurisés sans prendre en considération l'aspect logiciel.

Dans un premier chapitre, nous présenterons des notions importantes pour la bonne compréhension de ce manuscrit. Nous ferons un rapide historique sur les composants sécurisés puis décrirons leur architecture interne. La seconde partie de ce chapitre présentera de nombreuses notions cryptologiques en commençant par les principaux algorithmes de chiffrement. Ce chapitre se terminera par un état de l'art des attaques menées contre les circuits intégrés sécurisés.

Le second chapitre de ce mémoire traitera de l'utilisation et de l'intégration des technologies reconfigurables existantes dans les circuits afin d'en augmenter la sécurité, autrement dit nous aborderons la notion de « sécurité flexible ». Nous y démontrerons notamment comment faire face aux problèmes de clonage et de retro-ingénierie que rencontrent les industriels car l'emploi de la logique reconfigurable permet d'apporter des spécificités difficilement reproductibles à ces circuits. Nous complèterons ce chapitre par la présentation d'un protocole de reconfiguration à distance des circuits reprogrammables, apportant à la fois robustesse et rapidité.

Dans le troisième chapitre de ce manuscrit nous nous focaliserons sur l'attaque DPA qui constitue une réelle menace contre les circuits intégrés, mêmes pour ceux qui sont sécurisés. Dans un premier temps, nous établirons un modèle réaliste de mise en évidence de la dépendance entre les données qu'un circuit traite et sa consommation électrique. Ensuite, nous détaillerons l'analyse DPA et prendrons comme exemple l'algorithme DES. Enfin, nous présenterons une classification des principales contre-mesures capables de déjouer cette analyse.

Dans le dernier chapitre, nous présenterons une nouvelle contre-mesure à la fois simple et efficace pour déjouer l'analyse en courant. Nous évaluerons cette solution selon différents modèles de simulations. De plus, nous étudierons comment placer judicieusement ces contre-mesures dans un circuit plutôt que de façon naïve.

Nous évaluerons aussi ces placements selon différents niveaux d'abstraction, notamment en simulant toutes les portes logiques d'un circuit complet.

Enfin, les différents points abordés dans ce manuscrit seront résumés dans une conclusion dans laquelle nous aborderons les perspectives relatives à ces travaux.

CHAPITRE I.

CIRCUITS SECURISES ET NOTIONS DE CRYPTOLOGIE

Dans ce manuscrit de thèse, les notions de circuits sécurisés et de cryptologie seront régulièrement abordées. Pour cette raison, nous présenterons tout d'abord les circuits sécurisés de façon succincte puis passerons à la cryptologie. Nous terminerons par la classification des attaques menées contre ces composants. Ces notions sont nécessaires pour une meilleure compréhension des chapitres suivants. Pour les lecteurs ayant les connaissances sur les notions énoncées, la lecture de ce chapitre n'est pas indispensable.

I.1. Terminologie

La cryptographie est l'art de chiffrer des messages pour les rendre incompréhensibles par des personnes n'ayant pas la connaissance de la technique de chiffrement utilisée. L'étymologie provient du grec *crypto* pour cacher et *graphie* pour écriture.

La cryptanalyse est la science basée sur l'utilisation d'outils mathématiques qui tente à retrouver des secrets. Cette science peut, par exemple, avoir pour objectif de retrouver des clés secrètes ou encore les algorithmes utilisés lors de communications chiffrées.

La cryptologie est la science qui englobe la cryptographie et la cryptanalyse.

I.2. Circuits sécurisés

Dans ce manuscrit, nous entendons par circuit sécurisé, un circuit intégré comprenant au moins un mécanisme qui permet d'améliorer la sécurité d'un système. Généralement, de tels composants sont capables d'effectuer des opérations cryptographiques (chiffrement, signature numérique, fonction de hachage …) de

stocker des données confidentielles, de chiffrer des communications, de certifier des transactions ou encore de générer des nombres aléatoires. En plus de ces aspects, les circuits sécurisés comportent des mécanismes pour se prémunir contre des personnes qui tenteraient de les utiliser de manière frauduleuse et garantir ainsi une résistance élevée face à la cryptanalyse. Par exemple, ces mécanismes peuvent consister à vérifier l'intégrité des données manipulées ainsi que l'intégrité physique du circuit.

L'exemple le plus populaire de circuit sécurisé est la carte à puce [RAN03]. Ces cartes peuvent être utilisées pour valider des transactions comme dans le domaine du bancaire. Elles peuvent aussi autoriser ou interdire des accès à des services ; ceci est notamment le cas dans le domaine de la télévision à péage.

I.2.1. Historique

Initialement, le concept de carte à puce représente un composant renfermant des données confidentielles. Ainsi, ce concept est la première réalisation de circuit sécurisé. Dès 1947 des cartes contenant 64 bits sont utilisées. En 1969, la première mémoire portative est utilisée mais il est difficile de parler carte à puce car il n'existe alors pas de verrou afin de restreindre l'accès aux données, hormis de posséder un lecteur.

L'invention de la carte à puce est généralement attribuée à Roland Moreno qui en 1975 définit dans [MOR75] des moyens de restriction d'accès aux données confidentielles, placés dans le circuit lui même. Ces moyens sont entre autre la comparaison interne d'un code confidentiel, la définition de zones mémoires ou encore le comptage d'erreurs provoquant la réinitialisation du circuit.

Mais les cartes à puces actuelles ne seraient rien sans l'invention de l'allemand Dethloff. Il décrit dans [DET78] comment utiliser un microprocesseur pour restreindre les accès donnant ainsi naissance aux circuits sécurisés tels que nous les connaissons de nos jours. Cette invention offre une plus grande souplesse à ces composants.

Partant de ces principes, les techniques n'ont cessé de se perfectionner durant 30 ans. L'architecture est passée de l'utilisation de processeurs 8 bits à des processeurs de 16 puis 32 bits augmentant considérablement leurs capacités de calculs ; ceci sans tenir compte de l'accroissement de la fréquence de fonctionnement due à l'amélioration des technologies d'intégration. Les mémoires embarquées quant à elles sont devenues réinscriptibles et non volatiles avec les technologies EEPROM [RAN03] et Flash [RAN03]. Leur densité d'intégration est telle qu'actuellement, il existe des composants capables de stocker plusieurs centaines de Kilooctets (ou Ko) de données en mémoire EEPROM ou plusieurs Mégaoctets (ou Mo) en mémoire Flash. Nous sommes loin des 64 bits de mémoire ROM de la carte de 1947. Les techniques de restrictions et les algorithmes cryptographiques ont eux aussi connu des évolutions extraordinaires. Idem pour les interfaces de communication qui pendant de nombreuses années étaient basées sur une liaison série faible débit. L'évolution des besoins en termes d'application donne naissance à des nouveaux protocoles. C'est ainsi que les interfaces sans contact sont apparues pour lesquelles les données transitent par champs électromagnétiques, autorisant des identifications sans contact

avec les lecteurs de carte. D'autres cartes intègrent l'interface USB autorisant des transferts de données élevés.

La norme ISO-7816 [ISO] est divisée en plusieurs parties qui contiennent tous les standards des cartes à puce avec contact. Elles définissent entre autres les caractéristiques physiques, la nature des signaux électriques, les protocoles de communication ou encore certaines règles de sécurité. La norme ISO-14443 [ISO] est similaire à la précédente mais pour les cartes à puce sans contact. De nouvelles standardisations voient le jour notamment au sujet des interfaces et protocoles de communication NFC (pour *Near Field Communication* [NFC]) ou SWP (pour *Single Wire Protocol* [ISOTC].

Historiquement, les circuits sécurisés étaient nommés cartes à puce ou encore smartcards. Par abus de langage, les trois termes cartes à puce, smartcards et circuits sécurisés désignent le même composant.

I.2.2. Description

L'architecture d'une carte à puce peut s'apparenter à celui d'un microcontrôleur. En effet, ce circuit est principalement composé d'un processeur, de mémoires (volatiles et non volatiles), de coprocesseurs, de connecteurs et de périphériques [RAN03].

La différence majeure réside dans le fait que des verrous sont ajoutés afin de garantir l'intégrité du fonctionnement du système. Le processeur fait partie de ces verrous. Il représente l'élément central du système et est utilisé afin d'exécuter le code stocker en mémoire. Les processeurs les plus utilisés sont les processeurs RISC 8 bits car ils présentent un bon compromis entre puissance de calcul, consommation et encombrement. De plus en plus, les processeurs 32 bits sont utilisés afin de répondre à la demande croissante de puissance de calcul.

D'autres mécanismes, désignés par les blocs de protection du circuit de la Figure I.1, sont utilisés afin de garantir la sécurité. Certains de ces mécanismes scrutent l'environnement du circuit et informent le processeur de toute utilisation du système dans des conditions anormales. Il peut par exemple s'agir d'un capteur de température qui s'assure que le circuit se situe bien dans la gamme où le fonctionnement est garanti. Il existe aussi des capteurs capables de vérifier que l'intégrité physique du système n'est pas altérée. D'autres mécanismes ont pour objectif de vérifier que le microprocesseur n'exécute pas d'instructions erronées qui auraient pour conséquence de placer le système dans un état indéterminé. Enfin, les verrous peuvent consister à restreindre l'accès à certaines zones en fonction des privilèges dont bénéficie l'utilisateur.

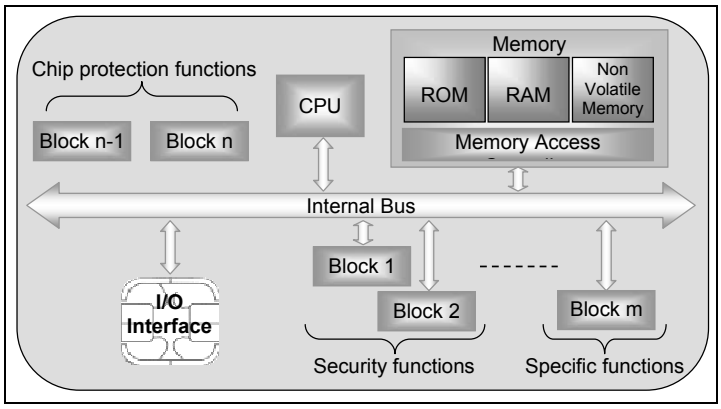

Figure I.1 : Architecture d'un circuit sécurisé.

Les mémoires sont un élément essentiel des circuits sécurisés. Rappelons qu'initialement, les cartes à puces intégraient uniquement cet élément. Les mémoires se décomposent en trois parties :

- la mémoire ROM, accessible seulement en lecture. Elle contient le code correspondant au système d'exploitation du circuit. Elle est figée lors de la fabrication du circuit intégré. Actuellement, il existe sur le marché des cartes qui intègrent 512 Ko de mémoire ROM.
- la mémoire RAM, accessible aussi bien en lecture qu'en écriture et est volatile. Elle contient les données temporaires du système.
- la mémoire non-volatile, accessible en lecture et en écriture dont la caractéristique est de ne pas perdre son contenu lors de la mise hors tension du circuit. Nous distinguons les mémoires basées sur la technologie EEPROM (la plus ancienne) et celles basées sur la technologie Flash. Ces deux technologies diffèrent principalement en terme de densité d'intégration, de courant consommé et de granularité des mots accessibles. C'est dans cette mémoire non-volatile que sont stockées les données confidentielles. Pour cette raison, de gros efforts de conception et d'optimisation sont réalisés sur ces dernières.

Etant donné l'application de tels composants, à savoir le stockage de données confidentielles et l'identification, il est fréquent de retrouver des fonctions communes à toutes les applications, désignées par les fonctions de sécurité de la Figure I.1. Il peut s'agir de fonctions de génération de nombres aléatoires ou encore de vérification des données (CRC pour *Cyclic Redundancy Checking*).

Enfin, il est courant que les circuits sécurisés utilisent des coprocesseurs, différents d'un circuit à l'autre. Ces derniers permettent de répondre à des besoins spécifiques pour une application donnée. Par exemple, selon le besoin, un circuit peut embarquer un coprocesseur capable d'accomplir un chiffrement asymétrique alors qu'un autre embarquera un coprocesseur capable de réaliser un chiffrement symétrique.

I.3. Cryptographie

Nous allons maintenant développer la notion de cryptographie inhérente aux circuits sécurisés.

I.3.1. Chiffrement

L'action de chiffrement (aussi appelé encryption) consiste à transformer un message clair (ou *PT* pour *Plain Text*), c'est-à-dire un message compréhensible de tous, en un texte chiffré (ou *CT* pour *Cipher Text*) compréhensible par les seules personnes en possession de tous les éléments. L'étape inverse qui permet de retrouver le message clair à partir du texte chiffré est appelée décryption (ou déchiffrement). Le chiffrement procure de la confidentialité aux communications.

Le principe du chiffrement est simple. Une personne, Alice, veut transmettre un message secret à Bernard que lui seul pourra comprendre au travers d'un canal non sécurisé. Pour cela, elle va chiffrer son message, et envoyer le message chiffré obtenu. Bernard sera la seule personne capable de déchiffrer le message et de comprendre ce qu'Alice voulait lui communiquer, car il est le seul à posséder tous les éléments nécessaires au déchiffrement. Si une personne, Estelle, espionne les communications afin d'interpréter les messages, elle ne sera pas capable de comprendre la conversation car elle n'a pas connaissance des éléments indispensables. En revanche, elle peut tenter une cryptanalyse. La Figure I.2 représente simplement ce principe. Ainsi, nous pouvons dire que le chiffrement procure de la confidentialité.

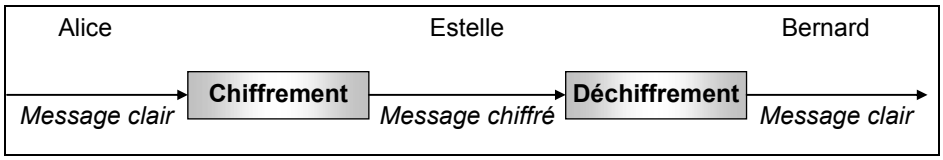

Figure I.2 : Principe du chiffrement.

Les algorithmes de chiffrement sont très anciens. Ils sont apparus presque aussitôt que l'écriture. Le plus ancien document chiffré remonte à environ 1900 av. J-C avec la civilisation égyptienne [KAH97]. Au V^{e} siècle av. J-C, pour transmettre des informations militaires les grecs de la cité guerrière de Sparte utilisèrent la technique surnommée « bâton de Plutarque » basée sur le principe de la transposition [KAH97]. L'empereur romain César utilisa le code qui porte son nom basé sur la substitution [KAH97]. Chaque civilisation a utilisé ses propres techniques de plus en plus perfectionnées afin de transmettre des informations bien souvent liées aux activités militaires. Ainsi, le XX^{e} siècle voit l'apparition de la machine Enigma [MUL07] utilisée par les allemands pendant la seconde guerre mondiale, capable d'automatiser la substitution. Jusqu'à la fin des années 1940, la force des différentes techniques de chiffrement était basée sur le secret de l'algorithme lui-même, par exemple de la table de substitution ou de transposition. La cryptographie moderne repose désormais sur des algorithmes publics utilisant tout de même une clé secrète.

Il existe de nombreux algorithmes de chiffrement, [MUL07] est un très bon support pour se familiariser de manière simple avec tous ces algorithmes. [SCH97] et

[MEN96] sont de bonnes références pour approfondir ce domaine avec une rigueur scientifique.

En plus de la confidentialité procurée par le chiffrement, la cryptographie moderne propose d'autres aspects fortement utilisés dans les applications sécurisées :

- L'authentification : elle permet de vérifier l'origine des données ou l'identité d'un utilisateur afin d'éviter qu'un individu mal intentionné puisse se faire passer pour une autre personne ayant certains droits dans un système. Ne pas confondre avec l'identification qui consiste à connaître l'identité d'un utilisateur.
- L'intégrité : cela consiste à vérifier que les données n'aient pas été modifiées par une personne non autorisée lorsqu'elles sont passées par le canal non sécurisé.
- La non répudiation : c'est le mécanisme qui empêche un expéditeur de nier avoir envoyé un message. La signature numérique est un bon moyen pour appliquer la non répudiation.

Dans cette étude, nous ne nous intéresserons qu'aux algorithmes de chiffrement par blocs. Ces derniers, regroupent les données par blocs de taille fixe avant de les chiffrer. Cette famille s'oppose aux algorithmes de chiffrement à flot continu qui chiffrent les données à la volée, sans les regrouper par blocs. Le chiffrement continu présente l'avantage d'être rapide et nécessite peu de ressources, quelle que soit l'implantation logicielle ou matérielle. En revanche, leur robustesse face à la cryptanalyse est peu élevée ; par conséquent leur utilisation est peu répandue dans les applications qui nécessitent un niveau de sécurité élevé.

La famille de chiffrement par blocs est la plus répandue ; parmi lesquels nous retrouvons notamment le célèbre DES (pour *Data Encryption Standard*), son successeur l'AES (pour *Advanced Encryption Standard*) et le *RSA* (pour *Rivest Shamir* et *Adleman*) pour ne citer que les plus connus.

- **Modes de fonctionnement**

Une faiblesse identifiée de ces algorithmes de chiffrement par blocs est que pour un algorithme et une clé donnés, un texte clair produira toujours le même texte chiffré. Ceci correspond au mode de fonctionnement ECB (pour *Electronic Code Book*).

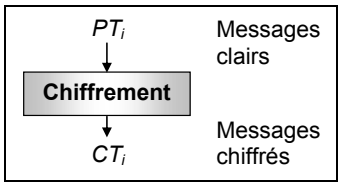

Figure I.3 : Mode ECB.

Il existe un autre mode de fonctionnement appelé CBC (pour *Cipher Block Chaining*). Pour ce mode, le bloc de message clair est combiné avec une donnée à l'aide d'un « OU Exclusif ». Le premier bloc de message clair est combiné avec un vecteur d'initialisation (ou IV pour *Initialization Vector*). Les autres blocs sont

combinés avec le bloc chiffré précédent. Ainsi, un même bloc de message clair ne produit pas le même bloc de message chiffré. Ce mode est souvent utilisé avec les algorithmes de chiffrement par bloc.

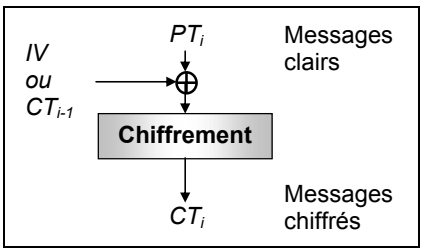

Figure I.4 : Mode CBC.

[NIS98] présente ces modes de fonctionnement ainsi que d'autres que nous n'aborderons pas dans ce manuscrit : CFB (pour *Cipher FeedBack*) et OFB (pour *Output FeedBack*).

I.3.2. Algorithmes symétriques

Les algorithmes symétriques sont ceux qui utilisent la même clé pour le chiffrement et le déchiffrement, aussi appelés algorithmes à clé secrète. Comme expliqué précédemment, nous n'évoquerons que les algorithmes de chiffrement par bloc.

Les algorithmes symétriques de chiffrement par blocs reposent sur deux notions de base, énoncées par Shannon qui sont la diffusion et la confusion [SHA49].

La diffusion consiste à disperser les données du message clair dans le message chiffré afin que le texte en sortie ne donne aucune information sur le texte clair en entrée. Pour une diffusion idéale, un changement d'un bit dans le message clair entraîne un changement de tous les bits en sortie avec une probabilité de ½. Ceci est appelé l'effet d'avalanche, souvent employé en cryptographie. Les transpositions et permutations sont des moyens de diffusion.

La confusion consiste à supprimer au maximum les relations entre les données chiffrées et la clé de chiffrement. Les algorithmes de César, Vigenère et Enigma en sont des exemples parfaits. La substitution est un moyen de confusion couramment utilisé dans les algorithmes symétriques.

Ces deux notions se retrouvent dans les réseaux de Feistel [FEI73]. Un réseau de Feistel est composé de plusieurs étages répétés de façon itérative, appelés rondes. Dans ces réseaux, les données sont séparées en deux blocs de taille identique et inversés pour chaque ronde. De plus, pour chaque itération, un des deux blocs est combiné avec une clé secrète puis avec l'autre bloc (voir Figure I.5).

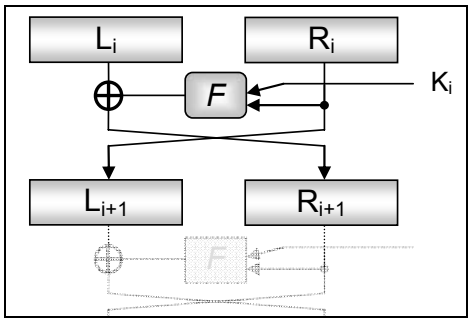

Figure I.5 : Ronde dans un réseau de Feistel.

L'avantage de cette construction est que le chiffrement et le déchiffrement sont structurellement identiques. Il suffit de répéter les itérations dans l'ordre inverse.

I.3.2.1. DES

Le DES (pour *Data Encryption Standard*) a été adopté par le NIST en 1977 et a été officialisé par [NIS77]. Son fonctionnement est entièrement public et est présenté dans la Figure I.6. Sa dernière mise à jour date de 1999 [NIS99].

Cet algorithme utilise la même clé pour le chiffrement et pour le déchiffrement. Ces deux opérations sont données par les deux équations suivantes :

$$CT = DES_K(PT) \qquad \textbf{(Equ. 1)}$$

$$PT = DES_K^{-1}(CT) \qquad \textbf{(Equ. 2)}$$

où CT représente le texte chiffré (ou *Cipher Text*), PT le texte clair (ou *Plain Text*) et K la clé (ou *Key*).

Il traite des blocs de 64 bits de texte clair avec une clé de 56 bits afin de générer, après 16 itérations successives d'un réseau de Feistel, un texte chiffré lui-même sur 64 bits. La Figure I.6 représente les opérations de chiffrement et déchiffrement.

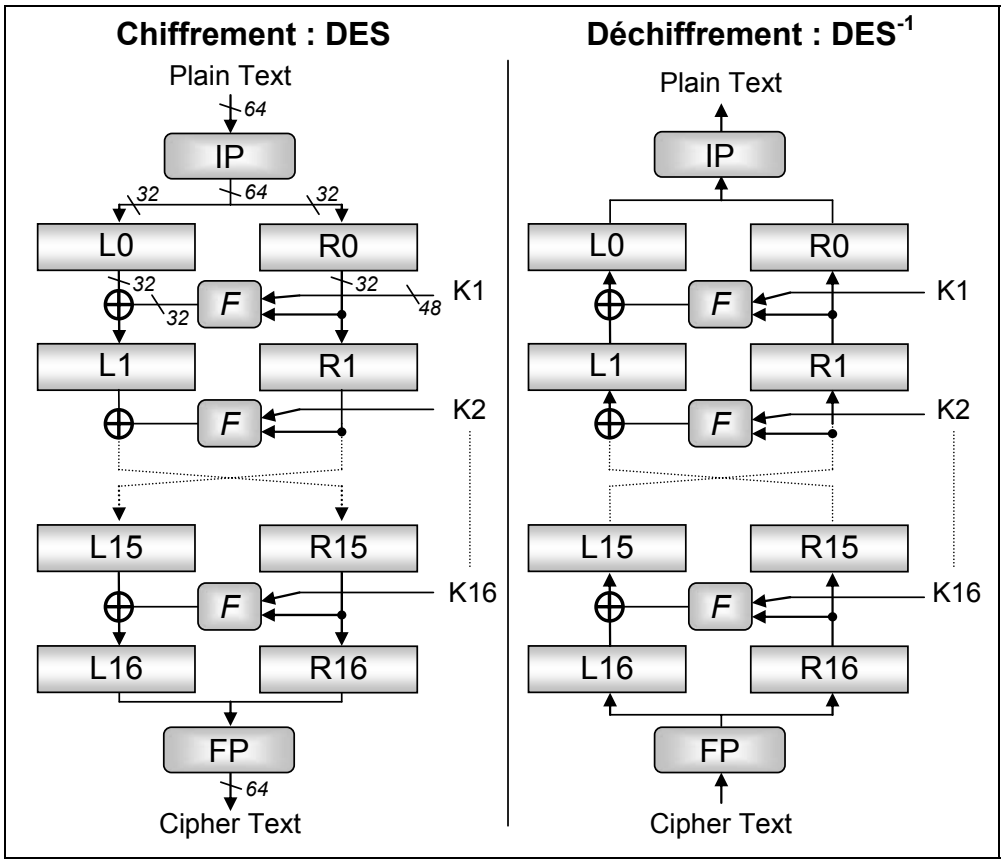

Figure I.6 : Opération DES et DES^{-1}.

- **Permutation initiale IP :**

Le bloc de 64 bits correspondant au texte clair subit tout d'abord une permutation initiale (ou *IP* pour *Initial Permutation*). Cette fonction est décrite dans le Tableau I-1. Il doit être interprété de la façon suivante. Le bit 1 sortant de la permutation initiale correspond au bit 58 entrant dans la fonction IP. Le bit 2 en sortie correspond au bit 50 entrant.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tableau I-1 : Permutation initiale IP.

- **Les rondes :**

Les 64 bits sortants de la permutation initiale sont scindés en deux sous blocs de 32 bits, nommés L_0 et R_0 représentant respectivement la moitié gauche et droite des données.

Ces deux variables L_0 et R_0 représentent les données d'entrée de ce qui est appelé une ronde (ou *round*). L'algorithme DES réalise 16 de ces rondes basées sur le schéma de Feistel et qui sont définies par les équations suivantes :

Pour $i \in \{1,16\}$:

$$L_i = L_{i-1} \oplus F(K_i, R_{i-1}) \quad \textbf{(Equ. 3)}$$

$$R_i = R_{i-1} \quad \textbf{(Equ. 4)}$$

Pour $i \in [2,15]$:

$$L_i = R_{i-1} \quad \textbf{(Equ. 5)}$$

$$R_i = L_{i-1} \oplus F(K_i, R_{i-1}) \quad \textbf{(Equ. 6)}$$

où *i* représente le numéro de ronde.

A la fin de l'algorithme, les données L_{16} et R_{16} sont combinées par une permutation finale (ou *FP* pour *Final Permutation*) afin de créer le texte chiffré (ou *CT* pour *Plain Text*) toujours codé sur 64 bits.

- **La fonction F :**

La fonction *F* est la même pour toutes les rondes, aussi bien pour un chiffrement que pour un déchiffrement. C'est une des raisons pour lesquelles cet algorithme a été retenu comme standard car cette propriété simplifie grandement l'implantation matérielle et logicielle.

Le vecteur de 32 bits qui entre dans la fonction *F* est étendu à 48 bits à l'aide d'une permutation expansive. Le "ou exclusif" correspond à un chiffrement de Vigenère.

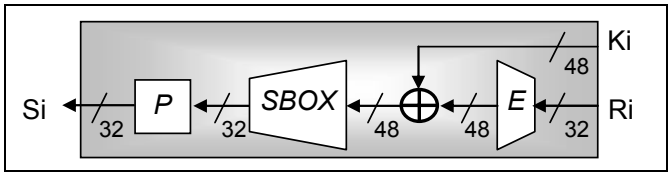

Figure I.7 : Fonction F.

- **La fonction E :**

La fonction *E* est une permutation expansive. Elle transpose un bus de 32 bits en un bus de 48 bits selon le Tableau I-2. En sortie de cette permutation, le bit 1 correspond au bit 32 du vecteur R_i ; le bit 2 sortant au bit 1 entrant …

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tableau I-2 : Permutation expansive E.

- **La fonction SBOX :**

La fonction *SBOX* (pour *Substitution Box*) est en fait composée de 8 sous-*SBOX* structurées comme présenté dans la Figure I.8. Elle est dite compressive car elle réduit un bus de 48 bits vers 32 bits. C'est la seule fonction non linéaire de l'algorithme.

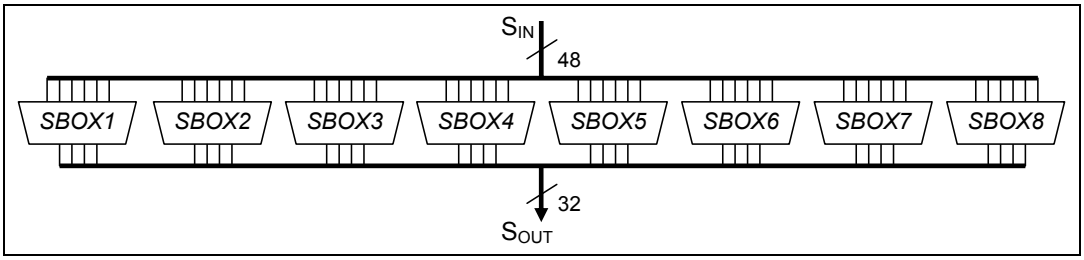

Figure I.8 : Description de la fonction SBOX.

Le Tableau I-3 représente le contenu de la SBOX1. Considérons les 6 bits entrant dans la SBOX1 désignés par [b_6 b_5 b_4 b_3 b_2 b_1]. Les bits b_6 et b_1 sont combinés afin de désigner une ligne du tableau, soit une valeur comprise entre 0 et 3. De la même façon, les bits b_5 b_4 b_3 b_2 désignent la colonne par une valeur comprise entre 0 et 15. La valeur contenue dans la cellule sélectionnée est celle qui sort de la SBOX1, codée sur 4 bits soit comprise entre 0 et 15.

Prenons comme exemple que ces 6 bits sont égaux à [100101]. Dans ce cas là, la SBOX1 fournira en sortie le vecteur [1010] correspondant à la valeur 12 contenue dans la cellule située à la ligne 3, colonne 2

	[b_5 b_4 b_3 b_2]															
[b_6 b_1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tableau I-3 : Table de la SBOX1.

Les tables correspondantes aux 8 *SBOX* sont données en ANNEXE.

- **La fonction P :**

La fonction *SBOX* est suivie d'une permutation, notée *P* et décrite dans le Tableau I-4. Son fonctionnement est similaire à celui de la permutation initiale.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Tableau I-4 : Permutation P.

- **Permutation finale FP :**

A la fin des 16 itérations, les blocs L_{16} et R_{16} sont concaténés avant de subir une dernière permutation (ou *FP* pour *Final Permutation*). Elle est l'inverse de la permutation initiale, soit $IP(X) = FP^{-1}(X)$. La table de permutation est représentée dans le Tableau I-5 dont le fonctionnement est similaire à celui des autres tables de permutation.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tableau I-5 : Permutation finale FP.

- **Génération des sous-clés de ronde :**

Les 16 sous-clés de rondes sont générées selon le principe de la Figure I.9.

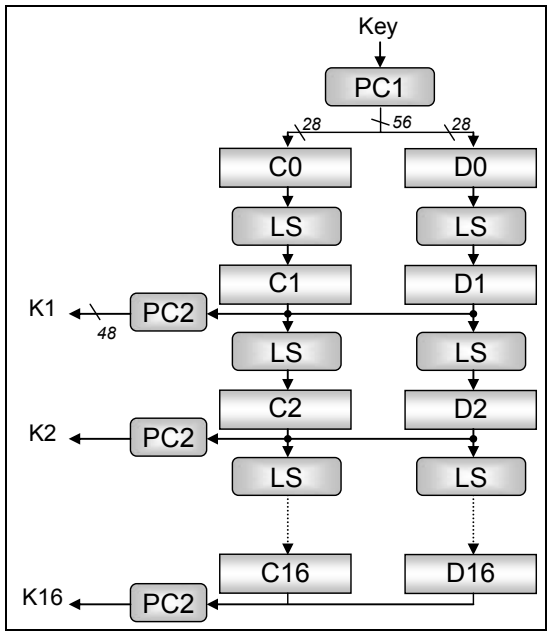

Figure I.9 : Génération des sous-clés de rondes.

Les 56 bits de la clé subissent tout d'abord une permutation PC1 donnée par le Tableau I-6. La clé résultante est séparée en deux parties de 28 bits chacune C_0 et D_0 contenant respectivement les bits de poids forts et les bits de poids faibles.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tableau I-6 : Permutation de clé PC1.

Chaque partie de 28 bits subit un décalage vers la gauche d'une ou deux positions en fonction de la ronde, donné par le Tableau I-7. Cette opération est désignée par la fonction LS dans la Figure I.9.

Itération	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nombre de décalage	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tableau I-7 : Décalage des sous-clés.

Enfin, les sous-clés de ronde sont générées après application d'une permutation compressive nommée PC2.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tableau I-8 : Permutation choisie PC2.

- **Déchiffrement**

La structure du DES étant basée sur les réseaux de Feistel, toutes les itérations (rondes) étant identiques et la permutation initiale étant l'inverse de la permutation finale, la même structure est aussi bien utilisée pour le chiffrement que pour le déchiffrement (voir Figure I.6). La différence réside dans la succession des sous-clés. Pour un chiffrement, les clés K_1, …K_i…, K_{16} sont utilisées alors que pour le déchiffrement elles sont utilisées dans l'ordre inverse, soit K_{16}, …K_i…, K_1.

Actuellement, le DES est utilisé pour des systèmes nécessitant un faible niveau de sécurité car il présente de nombreuses faiblesses [SCH97]. Il a été renforcé par le Triple-DES [NIS99] qui consiste à appliquer successivement trois calculs DES avec deux ou trois clés différentes mais présente toujours une faiblesse élevée face à la cryptanalyse car toujours basé sur l'algorithme DES. De plus ses performances sont limitées.

I.3.2.2. AES

L'*AES* (pour *Advanced Encryption Standard*) présenté en 2001 par le Nist [NIS01], a succédé au *DES* car sa résistance face à la cryptanalyse est depuis longtemps jugée trop faible. Il traite des blocs de 128 bits et peut s'utiliser avec des clés de 128, 192 ou 256 bits qui nécessite respectivement 10, 12 ou 14 tours. Contrairement au DES (et Triple DES), il ne possède pas de clé de chiffrement faible, autrement dit sensible à la cryptanalyse.

I.3.3. Algorithmes asymétriques

Le concept de cryptographie à clé publique a été inventé par W. Diffie et M. Hellman [DIF76a], [DIF76b].

Les algorithmes de chiffrement asymétriques, aussi appelés algorithmes à clé publique, s'opposent aux précédents car ils utilisent une paire de clés : une privée et une publique ; la seconde étant dérivée de la première. Une des particularités de cet algorithme réside dans le fait que la connaissance de la clé publique ne permet pas de retrouver la clé privée. Lorsqu'un utilisateur génère sa paire de clés privée et publique, il conserve sa clé privée secrètement et diffuse largement sa clé publique, il peut même la publier dans un annuaire [MEN96].

Considérons 2 utilisateurs A et B qui souhaitent communiquer par un canal non sécurisé, par exemple par le réseau Internet. Lorsque l'utilisateur B veut envoyer un message à l'utilisateur A, que seul ce dernier soit capable de déchiffrer, il va

chercher dans l'annuaire la clé publique de A : K_{PBA}. Il envoie son message à A après l'avoir chiffré avec K_{PBA}. L'utilisateur A peut déchiffrer le message car il est le seul à connaître la clé privée K_{PVA} nécessaire au déchiffrement. Si un utilisateur C malintentionné intercepte le message afin de le déchiffrer, il en est incapable car il connaît seulement la clé K_{PBA}. Cet échange est représenté dans la Figure I.10.

Contrairement au chiffrement symétrique, A et B n'ont pas besoin de se rencontrer préalablement afin de s'échanger une clé secrète. Cette propriété représente un gain conséquent.

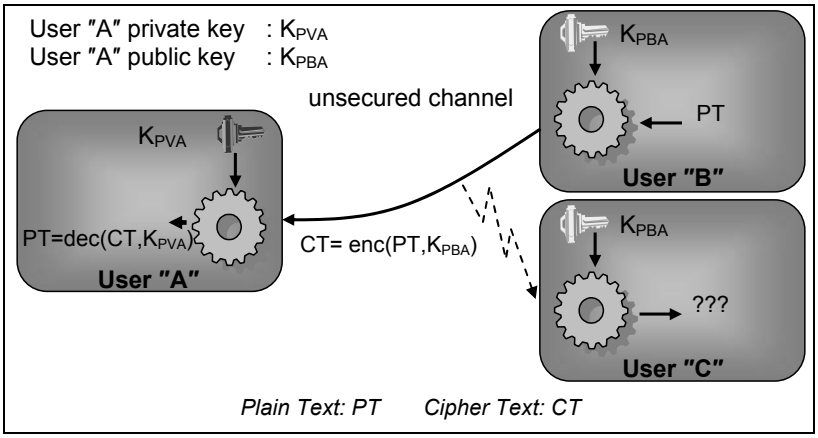

Figure I.10 : Chiffrement avec une clé publique.

Inversement, le chiffrement avec une clé privée (par exemple K_{PVA}) est possible mais ne présente aucun intérêt comparé au chiffrement symétrique pour procurer de la confidentialité. En effet, dans ce cas là, cela signifierait que la clé publique soit distribuée secrètement aux récepteurs voulus. De plus, un chiffrement asymétrique est nettement plus long que son homologue symétrique, typiquement 1 000 fois.

En revanche, le chiffrement à l'aide de la clé privée apporte la non répudiation et l'authentification. Admettons que l'utilisateur A veuille envoyer un message à B et que ce dernier veuille s'assurer que ce soit bien A qui le lui ait envoyé. Dans ce cas-là, B demande à A de chiffrer le message avec sa clé privée K_{PVA} avant de l'envoyer (Figure I.11). B reçoit le message et le déchiffre avec K_{PBA} qu'il avait dans son annuaire. S'il arrive à déchiffrer correctement le message, cela signifie que seul A a pu générer le message car il est le seul à connaître K_{PVA}. B est bien convaincu de l'identité de A et de la provenance du message. Nous pouvons donc dire que ce chiffrement répond aussi au concept de non répudiation car A ne peut nier avoir généré ce message.

Notons que tous les utilisateurs sont susceptibles de déchiffrer le message car ils peuvent eux aussi avoir connaissance de K_{PBA}. Voilà pourquoi le chiffrement avec K_{PVA} n'apporte pas de confidentialité.

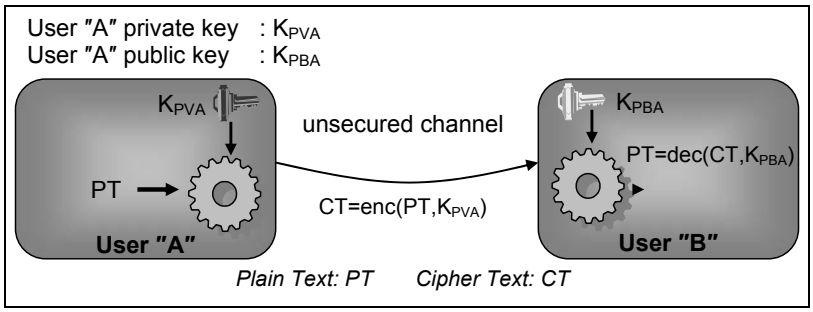

Figure I.11 : Signature avec une clé privée.

Parmi les algorithmes à clé publique, certains se prêtent mieux à la signature numérique, d'autres au chiffrement. Seuls RSA, ElGamal [ELG85] et Rabin [RAB79] sont suffisamment robustes pour répondre à ces deux notions. Leurs temps d'exécution sont beaucoup trop importants, le facteur est souvent supérieur à 1 000. Pour cette raison, ils sont bien souvent utilisés dans des cryptosystèmes dits hybrides [SCH97] afin de s'échanger des clés de session. Dans ce cas là, en reprenant le schéma de la Figure I.10, PT peut correspondre à une clé secrète qui va servir à chiffrer des données avec un algorithme symétrique.

Un des risques de toutes ces communications est de s'assurer que les clés publiques que les utilisateurs s'échangent ne sont pas corrompues. Si un attaquant a pu corrompre une clé publique, il lui devient aisé d'usurper une identité [SCH97].

- **RSA**

RSA du nom de ses inventeurs (R. Rivest, A. Shamir et L. Adleman) est l'algorithme asymétrique le plus répandu. Il a été décrit en 1978 [RSA78] suite à la publication du principe de chiffrement asymétrique [DIF76a]. Il présente une très grande résistance à la cryptanalyse. En effet, depuis bientôt trente ans qu'il est étudié, il résiste toujours. En revanche, il est nécessaire d'utiliser des clés dont la longueur est supérieure à 1 024 bits. Actuellement il est très utilisé dans les applications nécessitant un niveau de sécurité élevé : commerce électronique, Internet …

Le chiffrement RSA n'est pas utilisé systématiquement en lieu et place des algorithmes de chiffrement symétriques car il nécessite une puissance et un temps de calcul importants. En revanche, il est couramment utilisé pour réaliser la signature numérique ou pour échanger des clés de chiffrement symétrique.

La sécurité de RSA repose sur le fait qu'il est relativement facile de multiplier deux grands nombres premiers alors qu'il est beaucoup plus difficile de déterminer les facteurs premiers d'un grand nombre.

I.3.4. Fonctions de hachage

Une fonction de condensation (ou fonction de hachage) à sens unique (ou *One Way Hash Function*) génère un condensé d'une longueur fixe (généralement appelée *Hash* ou *Digest*) qui dépend du message d'entrée. Si un seul bit du message d'entrée change, tous les bits du condensé sont susceptibles de changer avec une probabilité de

½. Ces fonctions sont conçues de telle sorte qu'il est impossible de retrouver le message d'entrée à partir de son condensé.

Il existe des fonctions de hachage qui utilise une clé, appelées *Keyed Hash Functions* [SCH97].

La famille des fonctions SHA (pour *Secure Hash Algorithm*) est très répandue [NIS02] et [RIV92]. SHA-1 génère un condensé de 160, 256, 384 ou 512 bits. La longueur maximale du message est de 2^{64} bits pour SHA-1 et SHA-256, et 2^{128} pour SHA-384 et SHA-512. Le message est traité par blocs de 512 bits.

Le MD5 [RIV92] est une autre fonction de hachage à sens unique qui génère des condensés de 128 bits.

I.3.5. Détection de modification des données

Il est possible de coupler les fonctions de hachage à sens unique avec un algorithme de chiffrement afin de prévenir la modification des données (ou MDC pour *Modification Detection Code*), soit l'intégrité des données [MEN96].

Pour l'expéditeur, cet algorithme consiste à obtenir le condensé d'un message donné à l'aide d'une fonction de hachage, à concaténer ce condensé avec le message initial et à fournir cet ensemble à une fonction de chiffrement. Ce message chiffré peut être transmis via un canal non sécurisé.

Dans un premier temps, le destinataire déchiffre le message afin de distinguer le message et le condensé. Dans un second temps, il recalcule le condensé du message et le compare avec le condensé reçu. Si ceux-ci sont identiques, le destinataire considère que l'intégrité du message reçu est valide. Ce fonctionnement est présenté dans la Figure I.12.

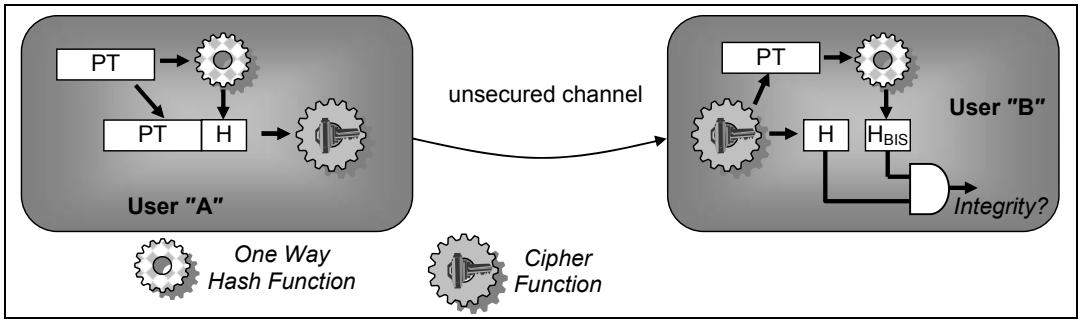

Figure I.12 : Détection de modification de données.

Plusieurs applications sont possibles :

- intégrer le condensé à chaque message émis si l'intégrité de chaque message est importante,
- calculer le condensé de tous les messages envoyés, mais ne l'intégrer qu'au dernier message émis si seulement l'intégrité de l'ensemble des informations est nécessaire.

I.4. Cryptanalyse

La cryptanalyse est le domaine de la cryptologie qui tente de retrouver les secrets, basées sur l'utilisation d'outils mathématiques. Ce secret peut, par exemple, être constitué de clés de chiffrement. Nous pouvons aussi parler d'attaque ou encore de l'action de « casser » un système. La cryptanalyse repose sur les principes de Kerckhoffs [KER83] énoncés au XIX^{e} siècle, en particulier celui stipulant que l'attaquant connaît tous les détails de l'algorithme utilisé ; le secret reposant seulement sur la clé. Ceci n'est pas forcément systématiquement vérifié dans la réalité mais constitue une bonne assertion.

Dans [ABR91], les auteurs présentent la taxinomie des attaquants. Ils sont regroupés en trois catégories :

- Classe I, attaquants isolés (ou *Clever Outsiders*) : Ils disposent de peu de moyens et ont peu de connaissances sur le système. Ils utilisent généralement des brèches dans les systèmes afin de parvenir à leurs fins.
- Classe II, chercheurs (ou *Knowledgeable Insiders*) : Cette catégorie regroupe les attaquants qui ont accès à du matériel relativement coûteux et possèdent une grande expertise technique.
- Classe III, organisations gouvernementales (ou *Funded Organizations*) : Dans cette classe, nous retrouvons les organisations qui possèdent des ressources financières, matérielles et techniques très élevées. Ces personnes sont capables de venir à bout de n'importe quel système.

Dans notre étude, nous souhaitons être résistant contre les attaquants de la classe II et perturber au maximum la tâche de ceux appartenant à la classe III.

Nous distinguons quatre familles d'attaques : exhaustive, logicielle, théorique ou matérielle. Dans ce mémoire, nous nous focaliserons sur la dernière.

I.4.1. Attaque exhaustive

Pour des raisons évidentes, l'attaque exhaustive a été la première utilisée. Elle consiste à tester toutes les combinaisons possibles sans avoir la moindre connaissance théorique sur l'algorithme ciblé. Elle exige une puissance et un temps de calcul élevés qui font que les attaquants de la classe I ne peuvent pas la réaliser.

Prenons l'exemple du DES. Admettons qu'un attaquant dispose d'un texte clair et du texte chiffré associé mais qu'il ne connaît pas la clé de chiffrement. Cela revient à calculer les $2^{56}=7{,}2.10^{16}$ combinaisons possibles et à les tester. Ceci représente un nombre faramineux de combinaisons, surtout avec les moyens dont disposent les attaquants lorsque le DES a été adopté en 1977. A la fin des années 90 quand les attaquants disposent de Pentium III cadencés à 666MHz capables d'effectuer 2 millions de chiffrement à la seconde, 600 années de calcul sont tout de même nécessaires pour obtenir toutes les combinaisons.

En 1998, l'organisation Distributed.net est parvenue à casser DES en 41 jours basé sur une mise en réseau de plus de 100 000 ordinateurs pour obtenir une puissance de calcul de 250 milliards de clés à la seconde.

En 1998, l'EFF (pour *Electronic Frontier Foundation*) proposa une solution matérielle baptisée Deep Crack [EFF98] coûtant 210 000 $ et constituées de 1 536 circuits. Cette machine est capable d'effectuer 92 Milliards de chiffrement DES à la seconde et de trouver une clé en 56 heures. Elle prouve par la même occasion que le niveau de sécurité du DES n'est plus acceptable car la NSA (pour *National Security Agency*) dispose de suffisamment de moyens financiers pour fabriquer des machines équivalentes.

En 1999, l'association de Distributed.net et de Deep Crack parvient même à retrouver une clé en moins de 23 heures achevant ainsi la polémique quant à l'obsolescence de l'algorithme DES.

I.4.2. Attaque théorique

L'attaque théorique, quant à elle, utilise les outils mathématiques et les propriétés des algorithmes de chiffrement afin de retrouver les secrets. De multiples méthodes existent.

En 1980, Hellman présenta un compromis temps / mémoire pour mener à bien une attaque avec moins de ressources et plus d'efficacité qu'une attaque exhaustive [HEL80]. En 1990, fut présentée la cryptanalyse différentielle [BIH90] qui permet de trouver la clé de chiffrement avec seulement 2^{47} textes clairs choisis. Une autre attaque connue est la cryptanalyse linéaire [MAT93] qui n'a besoin que de 2^{43} couples de données.

I.4.3. Attaque logicielle

L'attaque logicielle consiste à exploiter les failles des systèmes d'exploitations (ou OS pour *Operating System*) qui gèrent les opérations cryptographiques. Il peut s'agir d'un OS embarqué dans un circuit sécurisé ou de l'OS d'un système multiprocesseur. Dans ce manuscrit, nous ne traiterons pas de cette attaque. [BER05] et [ACI07] présentent certaines de ces analyses.

I.4.4. Attaque matérielle

Dans les précédentes attaques présentées, l'attaquant a seulement besoin de messages manipulés par un calcul cryptographique pour retrouver la clé qu'il recherche. Lorsque l'attaquant a un accès physique au circuit qui réalise ces opérations, il peut tenter la cryptanalyse dite matérielle. Depuis les premières méthodes présentées [AND93], [KOC96] et [KUH96], ce domaine a connu une véritable effervescence. Pourtant ce genre d'attaque n'est pas systématiquement plus rapide qu'une attaque exhaustive qui rappelons le est capable de casser un DES en moins de 23 heures. Ce qui rend cette attaque redoutable est sa simplicité de mise en œuvre et pour certaines méthodes le peu de moyens nécessaires. De plus, elle permet d'obtenir bien plus d'informations sur le circuit que les précédentes cryptanalyses : topographie, nature du chiffrement, données confidentielles stockées …

Dans ce mémoire, nous nous focaliserons sur les attaques matérielles.

I.4.4.1. Attaques invasives

Nous entendons par attaque invasive, un procédé qui attaque physiquement le circuit ciblé. C'est-à-dire que nous supposons que l'attaquant a extrait le circuit intégré de son boîtier grâce à l'utilisation de produits chimiques. Généralement, le circuit n'est plus fonctionnel après une attaque invasive, mais pas systématiquement.

- **Reconstruction de la topographie**

Cette attaque consiste à acquérir la topographie précise d'un circuit intégré pour en reconstruire tous les masques nécessaires à sa reproduction. Elle est communément appelée retro-ingénierie. Marcus Kuhn et Ross Anderson présentent très bien ce type d'attaque dans [KUH96]. L'analyse des différentes couches, couplée avec un logiciel de reconnaissance de formes est une technique répandue. Cette attaque s'avère redoutable pour révéler le contenu d'une mémoire ROM, comme en témoigne la Figure I.13, [KOM99], susceptible de contenir des données confidentielles ou des clés de chiffrement.

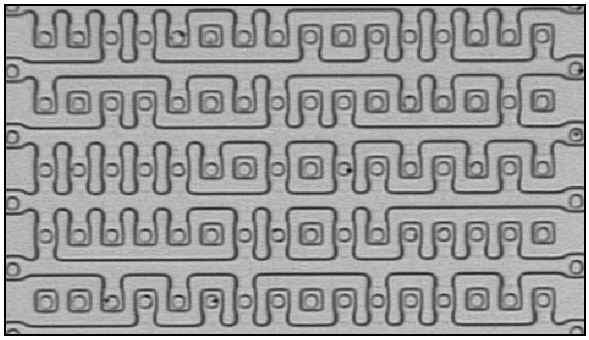

Figure I.13 : Photographie d'une mémoire ROM après retrait de la couche de passivation.

Elle prouve aussi son efficacité dans le repérage et l'indentification de cellules logiques. La Figure I.14 représente une porte NAND avant et après retrait de la couche de passivation [KOM99].

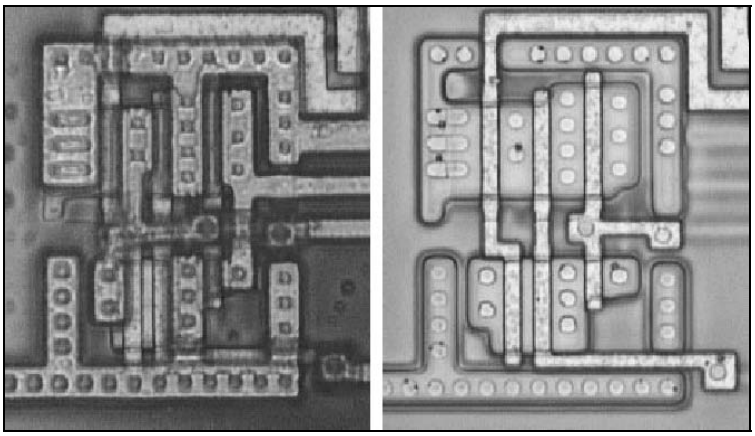

Figure I.14 : Photographie d'une porte NAND avant et après retrait de la couche de passivation.

Ce genre d'attaque nécessite des moyens importants en matériels et en ressources, qui se retrouvent principalement dans les organisations gouvernementales (Classe III).

- **Analyse sous pointes**

Cette attaque invasive nécessite que le circuit soit encore fonctionnel après décapsulation du circuit intégré et retrait de la couche de passivation [KUH96], [HAN99], [ISH03]. Elle consiste à isoler les bus sur lesquels transitent des données sensibles et à observer les signaux à l'aide de microsondes et d'un analyseur logique. Tout comme l'attaque précédente, elle requiert des moyens et connaissances importants.

I.4.4.2. Attaques semi-invasives

Les attaques semi-invasives consistent à perturber le circuit pour le faire basculer dans un état dégradé afin d'obtenir des indices sur les informations confidentielles.

Dans les attaques semi-invasives l'intégrité physique du circuit ciblé n'est pas atteinte. En revanche, contrairement aux attaques invasives, les personnes utilisant cette technique interagissent avec le circuit.

- **Attaques par injection de fautes : DFA**

L'attaque par injection de fautes (ou *DFA* pour *Differential Fault Analysis*) est la plus connue des attaques semi-invasives [BON97]. L'injection de faute tente de modifier le fonctionnement du circuit en modifiant certains paramètres. La liste ci-dessous présente quelques unes des techniques les plus connues :

- Variation importante et anormale de la source d'alimentation, du signal d'horloge ou des signaux d'entrée / sortie,
- Modification de la fréquence de fonctionnement,
- Variation de la température,

- Attaque laser [SKO02],
- Irradiation du circuit,

L'injection de fautes constitue un large domaine que nous n'aborderons pas dans ce mémoire. Elle n'est toutefois pas négligeable car de petits moyens permettent de casser des systèmes. [BIH97], [BIE00] et [BAR06] constituent une bonne base pour les lecteurs souhaitant en connaître plus sur ce sujet.

I.4.4.3. Attaques non invasives

Les attaques non invasives se distinguent des précédentes par le fait qu'elles n'interagissent pas avec le circuit ; une attaque invasive altère l'intégrité d'un circuit intégré et une attaque semi invasive modifie l'environnement du système.

Les attaques non invasives se contentent d'observer et d'analyser les informations qui fuient du circuit par l'intermédiaire de canaux cachés ; informations qui sont corrélées avec les données que celui-ci traite. Ces canaux cachés sont multiples et intrinsèques à l'utilisation de chaque technologie (CMOS, EEPROM, Flash, FPGA …).

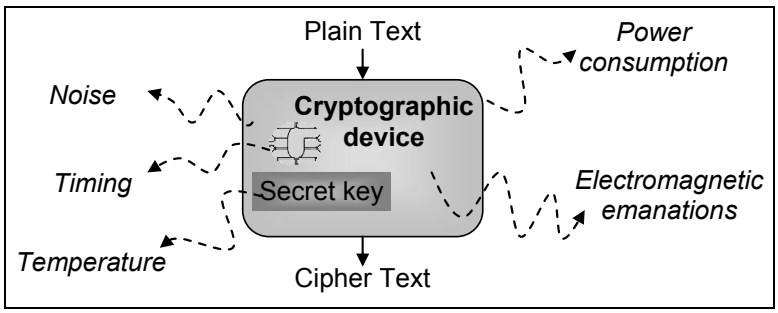

Figure I.15 : Fuite d'information par canaux cachés.

La puissance de cette attaque est qu'elle se contente seulement d'observer le système, ce qui signifie qu'un circuit ne peut détecter qu'une analyse est en cours. Pour cette raison, le défi des concepteurs de circuits intégrés pour se prémunir de ce genre d'analyse est de diminuer au maximum les fuites d'informations. Malheureusement, ces canaux ne peuvent bien souvent pas être supprimés ; dans le meilleur des cas, ils peuvent être diminués, voire décorrélés des données qui transitent. Parmi les analyses qui exploitent les canaux cachés, les principales sont les suivantes :

- Analyse acoustique : [SHA04] présente comment exploiter les émanations acoustiques afin de retrouver des informations sur le système, [CHA99].
- Analyse temporelle : Cette analyse exploite la dépendance qui existe dans certaines implantations entre le temps d'exécution et les données traitées [DHE98], [SCH00]. Dans [KOC96], l'auteur présente entre autre, une attaque réussie de RSA.
- Analyse électromagnétique [GAN01] : Cette méthode relativement récente présente des résultats intéressants. Elle consiste à observer les

rayonnements électromagnétiques émis par le circuit lorsque celui-ci manipule des données.

- Analyse de la puissance consommée : La plus répandue car elle possède un très bon rapport efficacité/simplicité. Nous distinguons deux analyses distinctes.
  - L'analyse SPA (pour *Simple Power Analysis*) [GEM01] qui consiste à observer des variations d'amplitudes de la puissance consommée et à les interpréter pour obtenir des informations sur les données manipulées. Elle est très efficace contre les circuits qui ne sont pas sécurisés, mais de simples contre-mesures sont suffisantes pour déjouer cette attaque. De plus, elle nécessite une bonne connaissance du système étudié.
  - L'analyse *DPA* (pour *Differential Power Analysis*) qui applique un traitement statistique à la puissance consommée par le circuit. Elle présente des résultats surprenants même sur des circuits sécurisés, sans posséder de détails sur le circuit attaqué. Nous reviendrons ultérieurement sur cette attaque dans le CHAPITRE III.

I.4.5. Synthèse

Certaines de ces attaques (retro-ingénierie, DFA, …) permettent d'obtenir un grand nombre d'informations sur les fonctionnalités internes du circuit. Elles sont tellement précises que les personnes mal intentionnées qui les détiennent sont capables de reproduire tout ou partie du composant. Cette reproduction illégale est désignée par le terme clonage. Les industriels dont les revenus sont basés sur la vente de composants sécurisés investissent de grosses sommes d'argent dans la recherche de techniques permettant de se prémunir de ce manque à gagner.

Par ailleurs, afin de leur permettre d'évaluer de façon objective leurs composants, les industriels ont à leur disposition différentes solutions. Les Critères Communs [CC] (ou *Common Criteria*) permettent d'évaluer la méthodologie de conception ainsi que la résistance d'un système face à la cryptanalyse.

I.5. Synthèse

Ce premier chapitre met en avant le fait que les circuits sécurisés sont les composants électroniques les plus utilisés afin de garantir la sécurité d'un système. Il nous a permis de mieux comprendre l'architecture interne des ces circuits. Ainsi, nous sommes maintenant en mesure de distinguer les principaux blocs constituant de tels circuits et la façon dont ils s'interfacent entre eux. Cette première partie est importante car par la suite nous nous baserons régulièrement sur cette architecture pour étayer nos démonstrations.

Dans un second temps, nous avons abordé des notions de cryptologie en commençant par la cryptographie. Nous avons présenté les principales techniques de chiffrement et nous avons particulièrement insisté sur l'algorithme DES. La

compréhension de cet algorithme est essentielle car il nous servira d'exemple dans la suite de ce manuscrit. Tout d'abord, nous l'utiliserons pour présenter une application de la sécurité flexible. Ensuite, lorsque nous présenterons l'attaque *DPA*, nous la détaillerons sur le DES. Enfin, nous présenterons des contre-mesures qui permettent de sécuriser des blocs logiques, applicables entre autre à ce même algorithme de chiffrement.

La dernière partie de ce chapitre concerne la cryptanalyse. Elle constitue un rapide état de l'art des attaques menées contre les circuits sécurisés. Nous nous sommes tout particulièrement attardés sur les attaques matérielles. En effet, rappelons que l'ensemble de nos travaux s'est focalisé sur l'aspect matériel des circuits sécurisés. Nous pouvons ainsi mieux comprendre les enjeux de la conception de tels circuits maintenant que nous prenons conscience de l'ingéniosité développée par certains pour extraire les informations confidentielles qu'ils renferment. Cet état de l'art met aussi en avant le fait qu'il est nécessaire pour un industriel de considérer toutes les attaques possibles (cryptanalyses théoriques, applicatives, physiques …) lorsque celui-ci veut concevoir un circuit avec un niveau de sécurité élevé. En effet, il semble peu judicieux d'utiliser l'algorithme le plus résistant à la cryptanalyse théorique, si celui-ci est implanté sur un circuit qui laisse fuir des informations pertinentes par l'intermédiaire de canaux cachés. C'est à cette problématique que nous allons tenter de répondre dans les chapitres à venir.

Dans ce manuscrit, nous allons proposer des solutions pour tenter de déjouer ces attaques, essentiellement la cryptanalyse matérielle, basée sur l'utilisation de logique reconfigurable.

CHAPITRE II.

SECURITE FLEXIBLE

Dans ce chapitre, nous allons proposer les concepts d'amélioration de la sécurité flexible répondant au mieux à nos contraintes ainsi que le protocole de reconfiguration associé. Rappelons que pour les concepteurs de circuits sécurisés, les besoins sont les suivants :

- Flexibilité afin de pouvoir modifier le circuit pendant sa durée de vie pour mieux répondre à l'obsolescence d'un algorithme ou d'une fonction.
- Pouvoir apporter des particularités à un circuit au niveau matériel, afin de pouvoir déjouer les attaquants qui tenteraient de reproduire un circuit (*cloning*). En d'autres termes, il s'agit de personnalisation d'un circuit.
- Se prémunir d'attaquants qui essaieraient de retrouver la fonctionnalité du circuit (retro-ingénierie).
- Etre capable de modifier la fonctionnalité d'un circuit à distance avec un niveau de sécurité élevé.

Par sécurité flexible, nous entendons la possibilité de faire évoluer un circuit sécurisé. Il peut s'agir d'une évolution suite à l'obsolescence d'un algorithme implanté dans un circuit. Cela permet au propriétaire du circuit de conserver un circuit performant d'un point de vue de la sécurité.

Dans un premier temps, nous étudierons rapidement les technologies reconfigurables en identifiant les solutions envisageables. Ensuite, nous proposerons une solution adaptée pour adresser nos besoins et présenterons un exemple d'application. Finalement, nous étudierons comment l'implanter dans un circuit afin de ne pas dégrader le niveau de sécurité.

II.1. Technologies reconfigurables

De nombreuses technologies sont disponibles afin d'intégrer de la logique reconfigurable. La plus répandue est la technologie dite FPGA (pour *Field Programmable Gate Array*) [CAR86]. Elle est composée de deux niveaux d'abstraction : une couche opérative et une couche de configuration, comme représentées dans la Figure II.1.

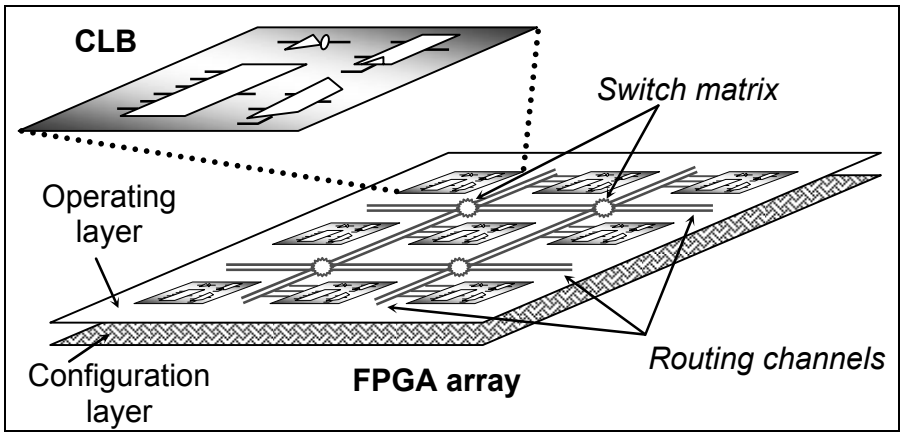

Figure II.1 : Description d'un FPGA.

La couche opérative est constituée d'éléments de base appelés CLB (pour *Configurable Logic Block*) eux même constitués de cellules logiques. Ces cellules logiques sont généralement des portes logiques élémentaires (NAND, OR, XOR, NOT, MUX …), des portes logiques évoluées (tables de vérités ou LUT (pour *Look-Up Table)*) et des bascules (FF pour *Flip-Flop*). L'architecture du CLB varie selon les fabricants, en fonction des besoins. En revanche, tous les CLB d'un même FPGA sont bâtis sur le même modèle. La différence fonctionnelle entre les CLB d'un même FPGA est réalisée par la couche de configuration. En effet, c'est cette dernière qui définit la fonctionnalité des portes complexes ainsi que les connexions entre les cellules logiques au sein même de la CLB. De plus, elle paramètre les *switch matrix* (littéralement matrice d'interrupteurs) dont la fonctionnalité est de connecter les différents CLB entre elles, au travers des canaux de routage (ou *routing channels*).

La couche de configuration s'apparente à un plan mémoire dont chaque bit vient configurer un élément de la partie opérative. Tout comme les mémoires, cette couche peut présenter différentes caractéristiques : volatile ou non, réinscriptible ou non. Par abus de langage, nous désignons un FPGA par le plan mémoire dont il est constitué. Trois types de plans mémoires distincts sont utilisés.

- Réinscriptible et volatile : Généralement, une mémoire de type SRAM (pour *Static Random Access Memory*) est utilisée (voir Figure II.2), [VIR07], [STR07]. Elle a besoin d'une source d'alimentation pour mémoriser la donnée. Le principal avantage est qu'il est possible de changer autant de fois que voulu la valeur stockée. En revanche, il est nécessaire de programmer entièrement la matrice de configuration à chaque mise sous tension avec le contenu d'une mémoire placée à coté du circuit. Cette contrainte entraîne des failles sécuritaires importantes : retro-ingénierie, clonage … De plus, dans [PAA03], les auteurs font références à des attaques réussies contre cette famille de FPGA.

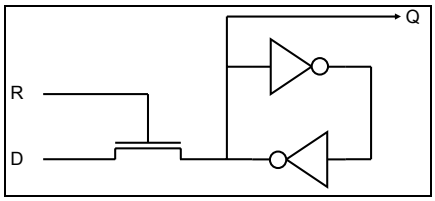

Figure II.2 : Point mémoire SRAM.

- Réinscriptible et non-volatile : le plan mémoire conserve son contenu même en l'absence de source d'alimentation et il est réinscriptible à volonté. Dans ce cas là, une mémoire de type Flash est employée, basée sur les transistors à grille flottante (voir Figure II.3), [PRO07]. Cette technologie présente une densité élevée et des caractéristiques de consommation de courant plus adaptées à l'utilisation des FPGA que les mémoires EEPROM. Cette solution présente un niveau de sécurité élevé contre le clonage en s'affranchissant de la faille des FPGA SRAM. En revanche, une attaque invasive qui tenterait de déterminer l'état d'un transistor à grille flottante (*reverse-negineering*) nécessite des moyens importants, offrant ainsi une résistance aux attaquants de classe III, mais reste réalisable.

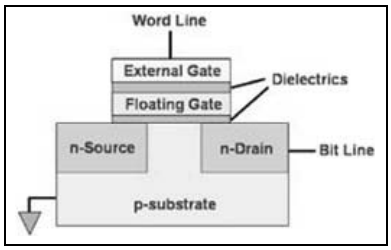

Figure II.3 : Point mémoire constitué d'un transistor à grille flottante.

- Non réinscriptible et non-volatile : le plan mémoire ne peut être programmé qu'une seule fois et conserve définitivement son contenu. Ainsi, nous parlons de FPGA configurable une seule fois. Seule la technologie antifusible présente ces caractéristiques (voir Figure II.4) [AXC07]. Cette solution présente un niveau de sécurité très élevé. En effet, une fois configuré, le plan mémoire n'est plus accessible évitant le clonage. De plus, une attaque invasive de type retro-ingénierie est incapable de déterminer l'état logique d'un antifusible (voir Figure II.5), [ACT02].

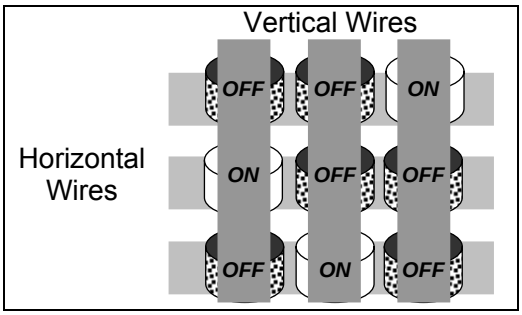

Figure II.4 : Connexions réalisées avec des antifusibles.

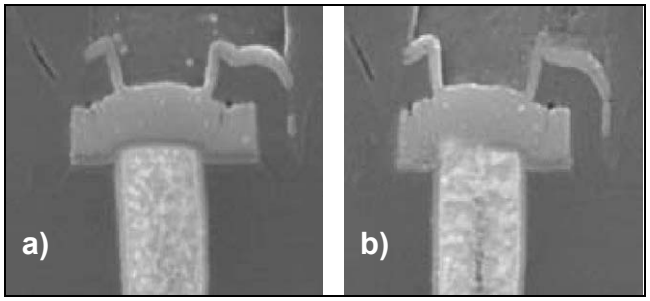

Figure II.5 : Antifusible programmé (a) et non-programmé (b).

La description faite dans cette partie sur les technologies reconfigurables est succincte mais permet de mieux appréhender la suite des explications. Pour les lecteurs qui souhaitent obtenir plus de renseignements, [CAR86], [BRO92], [CHO99a] et [CHO99b] contiennent de nombreuses informations détaillées sur les FPGA.

II.1.1. Circuits dédiés

Ces différentes technologies FPGA sont répandues sous forme de composants dédiés (ou *stand-alone*). Les matrices actuelles représentent l'équivalent de plusieurs Millions de portes implantées dans un circuit spécifique (ou ASIC pour *Application Specific Integrated Circuit*), [VIR07].

Il est établi que les FPGA *stand-alone* commercialisés actuellement ne permettent pas d'atteindre un niveau de sécurité suffisamment élevé [FIS06] et [PAA03]. De nombreuses techniques ont été proposées afin d'élever le niveau de sécurité [BOS04]. Mais dans tous les cas, les applications sont trop restreintes.

Xilinx, le fabriquant de FPGA, avec sa dernière génération de Spartan [SPA07] propose une alternative intéressante pour s'affranchir des attaques de clonage et de retro-ingénierie. Elle consiste à embarquer dans le FPGA une mémoire Flash non-volatile contenant la configuration (ou *bitstream*) et à transférer le contenu de cette mémoire dans la matrice reconfigurable lors de chaque mise sous tension [CRO07]. Lors de cette phase de transfert du *bitstream*, un certain nombre de vérifications et d'authentifications sont réalisées. En revanche, le contenu de la mémoire Flash n'est pas chiffré. Comme expliqué dans [BOS04], cela signifie qu'une attaque qui consiste à extraire le FPGA de son boîtier et à observer les données qui

transitent entre la mémoire Flash et le FPGA à l'aide de microsondes, permettra de reconstruire le *bitstream* et de cloner ce FPGA.

Retenons seulement les trois points suivants :

- Les FPGA antifusibles ont le niveau de sécurité le plus élevé mais ils ne permettent pas de changer la fonctionnalité du circuit.
- Les FPGA Flash présentent des aspects intéressants pour des systèmes nécessitant des niveaux élevés de sécurité, notamment une résistance élevée au reverse engineering. En revanche, l'utilisation de cette technologie nécessite des étapes supplémentaires lors de la fabrication, entraînant un surcoût important.
- Les FPGA SRAM présentent trop de faiblesse sous leur forme *stand-alone*.

Imaginons que malgré tous les efforts fournis, un attaquant parvienne à s'approprier le *bitstream* d'un composant. Il lui est possible de se procurer des composants identiques dans le commerce et de dupliquer le circuit, nous parlons alors de clonage. Ce risque n'est pas acceptable pour un industriel qui place tout son savoir faire dans la fonctionnalité de ce circuit et non pas dans le circuit lui-même. Pour cette raison, l'utilisation d'un circuit courant présente un risque supplémentaire auquel nous préfèrerons un circuit dont une partie au moins est spécifique.

II.1.2. Cœurs reconfigurables

Il existe une autre façon qui permet de créer un système sécurisé flexible. Elle consiste tout simplement à réutiliser une architecture et à lui ajouter un bloc reconfigurable. Nous parlons alors de cœur ou encore d'IP (pour *Intellectual Property*) embarquée dans un système sur puce ou SoC (pour *System On Chip*). Au final, une partie du système est figée selon le principe d'un ASIC et une autre partie est flexible (voir Figure II.6). Ceci représente un intérêt certain pour un industriel qui possède un savoir faire dans le domaine de la sécurité et dont l'architecture est éprouvée.

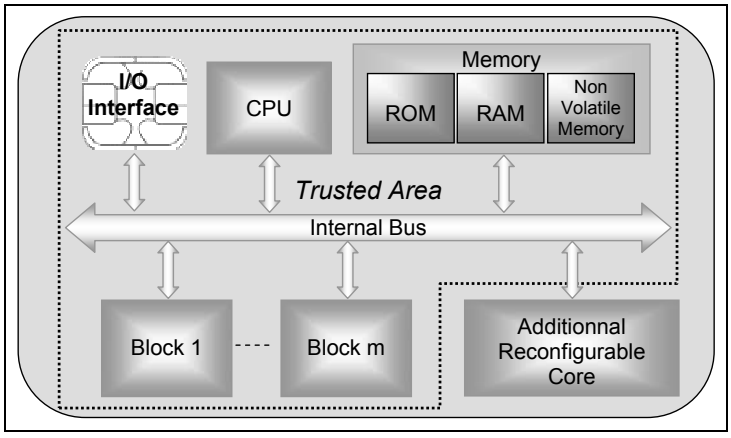

Figure II.6 : Ajout d'un cœur reconfigurable dans un système existant.

Ainsi, l'utilisation de cœurs est préférable aux composants FPGA dédiés car sous cette forme toute la sécurité du système ne repose pas sur la seule matrice

reconfigurable. En revanche, une méthodologie d'intégration doit être respectée. Nous considérons que l'architecture existante constitue la zone de confiance (ou *Trusted Area*) et que c'est sur cette dernière que repose le contrôle du cœur. En effet, dans le cas où le contenu de la zone reconfigurable serait corrompu (par un cheval de Troie ou un virus), l'accès de cette zone au reste du système doit être contrôlé.

Il faut porter une grande attention à l'intégration de cette IP reconfigurable dans le système afin de ne pas baisser le niveau de sécurité du système mais au contraire l'élever. Pour cela, elle doit être englobée par des verrous contrôlés par des fonctions présentes dans la zone de confiance.

De plus, si cette zone additionnelle est basée sur une mémoire volatile, il convient d'utiliser une mémoire non-volatile interne au système pour stocker le *bitstream* afin de se prémunir des attaques menées contre les composants *stand-alone* [FIS06]. Cette solution sous entend une redondance de la mémoire nécessaire au stockage de la configuration. En revanche, si le plan mémoire de cette zone est basé sur une mémoire non-volatile, cette redondance est évitée.

De plus, l'accès à la mémoire de stockage de la configuration (la mémoire non-volatile du système ou directement le plan de configuration de la zone additionnelle) peut être réalisé de différentes façons. Il est possible de créer une interface dédiée contrôlée par le système ou d'utiliser les ports de communications déjà existants dans le système. Dans tous les cas, l'accès au plan mémoire ne doit pas être assuré par la zone mémoire elle-même mais par la zone de confiance.

Par le passé, de nombreux industriels se sont lancés dans la conception de cœurs FPGA embarqués [BUR02], [CAT02], [WIL02], [WIL03]. Certains processeurs intègrent même des cœurs reconfigurables [BOR03], [STM05]. A ce jour, seulement deux sont commercialisés avec une maturité suffisante. Nous allons étudier s'il est possible de les utiliser dans un circuit sécurisé et quelles en sont les restrictions.

II.1.2.1. eFPGA

Le concept d'eFPGA (pour *embedded FPGA*) est de proposer un bloc reconfigurable dont l'architecture est similaire aux FPGA dédiés. Actuellement, la solution la plus avancée est proposée par la société M2000 avec son cœur FlexEOS dont l'architecture est hiérarchisée, [M2000], [MEN]. L'élément de base MFC (pour *Multi-Function Cell*) est constitué d'une LUT configurable à 4 entrées et d'une bascule FF. Un nombre important (quelques milliers) de MFC est connecté par un réseau d'interconnexion local afin de créer un cluster. Ces clusters sont ensuite connectés entre eux par un réseau d'interconnexion global. Ce cœur comporte aussi des ports d'entrée / sortie configurables ainsi qu'un contrôleur connecté au reste du bloc par le réseau global (Figure II.7). La flexibilité de cette solution est élevée.

Le nombre de portes équivalent ASIC est défini par le nombre de MFC et de clusters utilisés. Ce cœur est décliné en différentes tailles mais toujours assemblé selon la même architecture.

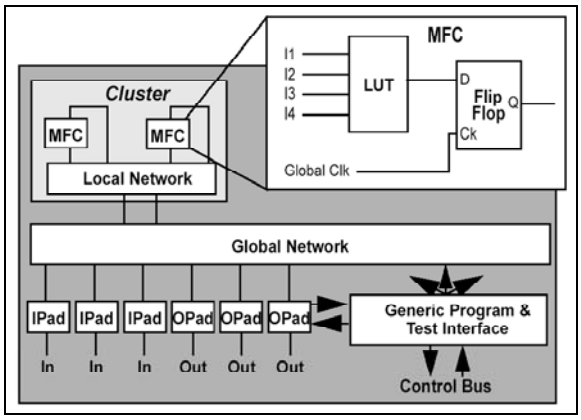

Figure II.7 : Architecture du cœur FlexEOS.

La configuration est stockée dans un plan mémoire volatile. Le fonctionnement s'apparente donc à un FPGA SRAM.

Un cœur FlexEOS 4k (équivalent à 4 000 portes d'un ASIC) présente les caractéristiques suivantes lorsqu'il est basé sur une technologie CMOS 0,13µm :

- Surface : 6,38 mm². Cette surface est relativement élevée mais nécessaire. Une logique reconfigurable occupe toujours plus de surface qu'une logique basée sur des cellules standard. Pour les circuits sécurisés, cette surface peut être rédhibitoire.
- Taille du bitstream : ~62 Ko. Ce bitstream doit être stocké dans une mémoire non-volatile interne au système. Cela représente un coût non négligeable.

Les deux points précédents représentent un surcoût important mais tout de même envisageable car le produit ainsi réalisé possède une valeur ajoutée évidente.

Au niveau de la sécurité, ce cœur n'intègre aucune sécurité particulière et représente une faiblesse puisque le seul moyen est de le sécuriser en périphérie. Ce point est bloquant car si, par exemple, le concepteur du circuit veut se prémunir de la DFA, il lui est indispensable d'agir à l'intérieur de cette zone logique.

De plus, une surface aussi importante est facilement repérable lors d'une attaque de retro-ingénierie. Nous pouvons imaginer qu'un attaquant peut aisément repérer la zone et plus finement l'interface de configuration afin de récupérer le contenu du *bitstream* par *micro-probing*. Il peut ensuite l'analyser pour le reproduire sur un autre circuit reconfigurable (clonage).

II.1.2.2. Structured Asic

Une autre vision s'oppose aux eFPGA. Elle consiste à utiliser une matrice prédéfinie de fonctions complexes (CLB) et un réseau d'interconnexions paramétrables selon le même principe que les FPGA (voir Figure II.1). La différence réside dans le fait que la configuration n'est plus réalisée par des points mémoires reprogrammables, mais par les niveaux de métaux les plus élevés [VIA], [EAS]. [LI08] répertorie les solutions actuelles. Ainsi, la fonctionnalité du bloc n'est pas reprogrammable mais figée par construction ; le terme *mask-configurable* est aussi

utilisé [COO03]. Ces solutions ne répondent pas à nos besoins de reprogrammation à distance pendant la durée de vie du produit et ne seront donc pas étudiées.

En revanche, la société eASIC propose un compromis entre eFPGA et *structured* ASIC. Elle consiste à employer des connexions *mask-configurables* figées entre les cellules de bases (ou eCells) et à l'intérieur de ces eCells. En revanche, ces eCells sont constituées de deux LUT3 dont les fonctionnalités peuvent être figées par construction ou programmées à la manière des FPGA SRAM (voir Figure II.8). Il s'agit de solutions bien distinctes. Cette dernière solution peut éventuellement satisfaire nos besoins en sécurité flexible.

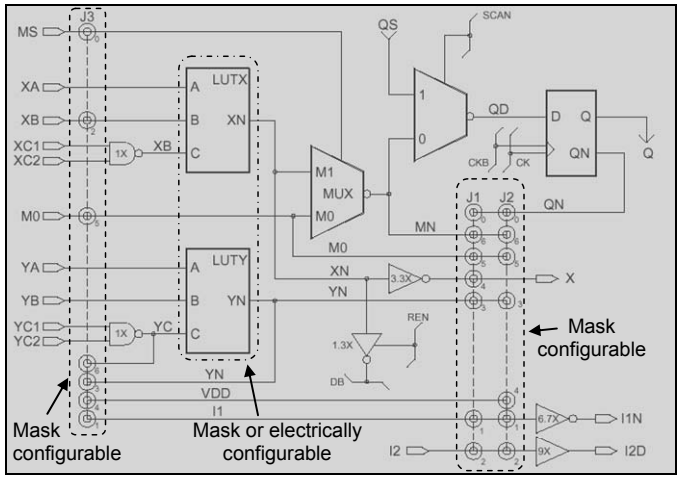

Figure II.8 : Cellule de base du cœur eASICore.

Le cœur eASICore, équivalent à 25 000 portes ASIC, présente les caractéristiques suivantes pour une technologie CMOS 0,13 µm :

- Surface : 0,5 mm². Ce cœur présente une densité plus élevée que l'eFPGA. Ceci est dû au fait que le routage n'est pas paramétrable électriquement mais dès la conception seulement. De plus, cette IP est basée sur l'utilisation de 6 niveaux de métaux. Le surcoût engendré par ces nivaux supplémentaires n'est pas négligeable.
- Taille du *bitstream* : 4 Ko. Pour la même raison que précédemment, la taille du *bitstream* est réduite car il correspond seulement à la configuration des 4 096 LUT3 (1 octet par LUT3).

Bien que plus dense que l'eFPGA, la conception d'un circuit embarquant un cœur reconfigurable basé sur cette solution entraîne un surcoût dont il faut tenir compte avant la commercialisation.

Nous retrouvons aussi les mêmes inconvénients que précédemment à savoir la sécurisation en périphérie seulement et la faiblesse face à la retro-ingénierie.

II.1.3. Synthèse

L'utilisation des cœurs reconfigurables existants sur le marché n'est pas la solution la plus adaptée pour répondre à nos besoins en sécurité flexible. Les caractéristiques sont reprises dans le Tableau II-1.

Certaines raisons sont sécuritaires et inhérentes au simple fait d'utiliser des IP du marché. En effet, il est possible de cloner le contenu de notre cœur dans un autre circuit qui contient la même IP. De plus, l'architecture de tels cœurs est figée et il est donc seulement possible de sécuriser la périphérie de ce bloc. Pour cela, l'utilisation d'un cœur propriétaire est préférable. L'architecture ne serait pas présente dans des produits commercialisés par d'autres industriels et intègrerait les contre-mesures maîtrisées.

D'autres raisons sont économiques. La granularité des cœurs commercialisés est élevée et non sécable ainsi que le surcoût engendré par l'utilisation d'un grand nombre de niveaux de métaux n'est pas applicable dans un contexte de circuits dont le prix de commercialisation doit être faible. Un industriel qui veut intégrer de la logique reconfigurable doit avoir le choix entre disposer la logique reconfigurable de façon répartie ou regroupée dans un bloc. Pour cette raison, il est préférable pour un industriel de disposer d'une granularité fine.

	eFPGA [M2000]	**Structured ASIC [EAS]**
Description	Electrically programmable LUTs Electrically programmable routing	Electrically programmable LUTs Mask programmable routing
Density (Gates/mm²)	6.22 K	50 K
Security features	No	No
Metal Layers	4	6
Bitstream (bits/Kgates)	12.5 K	1.3 K
Flexibility	High	Low
Minimum Area (mm²)	6.3 (for 25 Kgates)	0.5 (for 25 Kgates)
Frequency (MHz)	100	400

Tableau II-1 : Caractéristiques des différents cœurs reconfigurables.

L'utilisation d'un cœur reconfigurable propriétaire représente un investissement trop important pour les circuits sécurisés (développement de la technologie reconfigurable ainsi que des outils associés). De plus, cette solution est vulnérable contre la retro-ingénierie car la zone reconfigurable est facilement repérable dans un circuit intégré. Enfin, le taux d'occupation d'une matrice, même propriétaire, n'atteint jamais 100% mais plutôt 60 à 70%.

Pour ces raisons, nous allons étudier l'ajout de logique reconfigurable répartie, placée discrètement dans les parties sensibles du circuit, de façon transparente pour l'utilisateur final, afin d'ajouter de la sécurité flexible tout en s'affranchissant des problèmes sécuritaires et économiques énoncés jusqu'ici.

II.2. Logique reconfigurable répartie

Nous allons dans cette partie proposer une solution pour répondre aux besoins des industriels en flexibilité pour faire face au clonage et à la retro-ingénierie.

Pour répondre aux contraintes économiques, nous allons étudier une solution basée sur l'utilisation de cellules standard. Ceci évite le développement d'une nouvelle technologie, permet de s'adapter à n'importe quelle technologie d'intégration CMOS et est compatible avec les outils de conception numérique *front-end*. Nous baserons notre étude sur une technologie CMOS 0,13 µm. De plus, nous n'allons pas poursuivre le concept de cœur mais plutôt de logique reconfigurable répartie.

Ainsi, nous allons étudier un élément reconfigurable simple qui puisse être instancié autant de fois que nécessaire à différents emplacements. Cet élément doit permettre d'ajouter de la flexibilité simple ou élevée dans une macro-cellule ou même sur les bus d'interconnexion (voir Figure II.9). Le niveau de flexibilité dépendra du nombre d'éléments reconfigurables instanciés dans l'IP.

L'intérêt est que l'utilisation est parfaitement adaptée aux besoins avec un taux d'occupation de la logique de 100%. En revanche, la flexibilité sera faible mais ciblée. Ce qui induit une étude préalable de l'architecture afin de définir correctement l'emplacement de la flexibilité. En effet, il appartient au concepteur du circuit d'identifier les zones qui nécessiteraient d'être flexibles. De nombreuses possibilités sont envisageables, nous allons en présenter quelques unes.

Il peut s'agir de personnaliser le système afin de le rendre non reproductible. Par exemple, cela peut consister à utiliser un cœur de processeur standard et à rendre flexible sa fonction de décodage d'instructions ou tout autre sous fonction. Concrètement, cela permet de modifier dynamiquement le jeu d'instructions du processeur. De la logique reconfigurable peut aussi être instanciée sur le bus interne afin de permuter les bits de données et/ou adresses. Dans la même idée, l'interface qui permet d'accéder aux mémoires peut être flexible afin d'intégrer un chiffrement simple (substitution) et dynamique.

Nous pouvons implanter cette cellule reconfigurable dans une macro-cellule afin de la personnaliser par rapport à la même cellule qui peut se retrouver dans un autre circuit.

Il est même possible d'intégrer un grand nombre d'éléments flexibles dans un même bloc afin d'offrir une grande souplesse à cette fonction.

L'avantage d'instancier de la sécurité discrète transparente pour l'utilisateur est qu'elle est difficilement décelable ; un utilisateur mal intentionné n'aura potentiellement pas connaissance de ces éléments. Quand bien même il parviendrait à en connaître l'existence et l'emplacement, il doit être capable de les reproduire.

Cette liste non exhaustive présente quelques utilisations possibles afin de personnaliser un circuit pour le rendre difficilement reproductible, renforçant ainsi sa résistance au clonage. De plus, un attaquant a besoin de l'emplacement précis de ces éléments ainsi que le *bitstream* associé (codé selon un format propriétaire) pour comprendre les fonctionnalités, ce qui signifie que le système présentera une résistance accrue face à la retro-ingénierie.

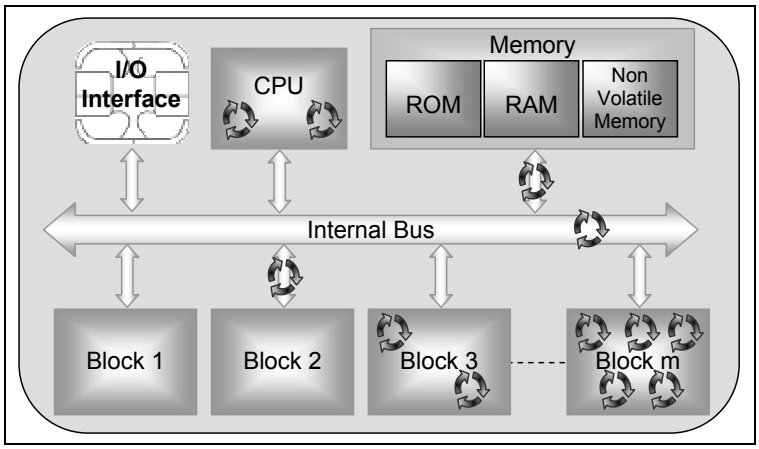

Figure II.9 : Ajout de logique reconfigurable répartie.

Afin d'ajouter de la flexibilité avec une granularité fine, nous retenons l'élément le plus basique qui soit, la table de vérité (ou LUT pour Look-Up Table). Ce choix permet de réaliser n'importe quelle fonction logique combinatoire : simple ou complexe. Nous n'intégrons pas de Flip-Flop en sortie de LUT car ces éléments ont une surface trop importante et ne sont pas systématiquement utilisés. Au besoin, le concepteur peut connecter les cellules qu'il souhaite en sortie de LUT : une autre LUT, une FF, une cellule logique, un port d'entrée /sortie … L'avantage de n'utiliser que des éléments simples (granularité fine) est que nous ne sommes pas pénalisés par la surface des éléments non utilisés présents lorsque la granularité est épaisse.

Dernier point, l'ajout de ces cellules ne remet pas en question les contre-mesures déjà utilisées et éprouvées ; cela permet de les réutiliser si celles-ci sont compatibles.

La solution que nous proposons possède les caractéristiques reprises dans le Tableau II-2.

	Proprietary Distributed Reconfigurable Logic
Description	Electrically programmable LUTs Mask programmable routing
Density (Gates/mm²)	Well-adapted
Security features	Yes, re-use
Metal Layers	Well-adapted
Bitstream (bits)	8 / LUT3
Flexibility	Well-adapted
Minimum Area (mm²)	~ 300 µm² / LUT3
Frequency (MHz)	Well-adapted

Tableau II-2 : Caractéristiques du bloc reconfigurable propriétaire.

II.2.1. Look-Up Table

Il est possible d'utiliser différentes dimensions de LUT. Afin d'optimiser le temps de configuration, il est possible d'adapter leurs dimensions à la longueur de données que le système peut traiter. Par exemple, pour une plate-forme 32 bits, nous pouvons utiliser des LUT5 qui nécessitent 32 bits de configuration ; une plate-forme 16 bits sera adaptée à l'utilisation de LUT4 (16 bits de configuration) ; ainsi de suite. En revanche, l'utilisation de LUT de grandes dimensions (supérieures à 4 entrées) risque de diminuer le taux d'occupation de la logique flexible. Le meilleur choix est encore de choisir la dimension des LUT en fonction de la fonctionnalité que celles-ci doivent accomplir.

Admettons que notre plate-forme traite des données de 8 bits, nous utilisons dans ce cas là des LUT3 constituées d'un multiplexeur (MUX8) et de 8 points mémoires comme représentées dans la Figure II.10. Dans ce cas là, les 3 bits d'entrée permettent de commander le multiplexeur afin d'aiguiller vers la sortie, un des bits contenu dans les éléments de mémorisation.

Le multiplexeur peut se décomposer en sous multiplexeurs extraits de la librairie de cellules disponibles si le MUX8 n'est pas disponible.

Pour stocker la configuration, plusieurs solutions sont envisageables :

- SRAM : utilisation d'un bloc SRAM de 8 bits dédié pour chaque LUT3. Ce bloc doit pouvoir être chargé en série ou en parallèle et diriger en permanence le contenu des 8 bits vers le multiplexeur. L'inconvénient est que cette solution implique la conception d'un bloc adapté à la taille de la LUT avec la logique de décodage et de chargement nécessaire. Cela représente une surface importante.
- Mémoire non-volatile : un bloc de mémoire non-volatile (EEPROM ou Flash) de 8 bits pour chaque LUT. Les inconvénients sont les mêmes que

pour l'utilisation d'une mémoire SRAM. De plus, une mémoire NVM demande plusieurs cycles pour générer les données.

- D-Latch : une cellule par point mémoire, soit 8. Ces cellules font généralement partie des cellules standard. L'inconvénient de l'utilisation de tels éléments est que le chargement série est impossible. Cela implique donc d'intégrer une interface de configuration qui chargera les LUT en parallèle. Cette solution est réalisable.
- Bascule Flip-Flop : une FF par point mémoire, soit 8. Ces cellules sont standard. Dans ce cas là, il est possible de chaîner les FF entre elles afin de réaliser une configuration série. Cette façon de programmer nécessite un nombre de cycle plus important mais simplifie grandement le contrôleur de configuration. En effet, il est possible de chaîner toutes les LUT entre elles et de n'utiliser qu'un seul contrôleur.

Nous opterons pour la dernière approche car elle est plus souple. Elle permet de chaîner autant de LUT que nécessaire sans besoin de redimensionner le contrôleur en fonction du nombre de cellules instanciées et offre la possibilité d'implanter le test logique par *scan* [CHE89].

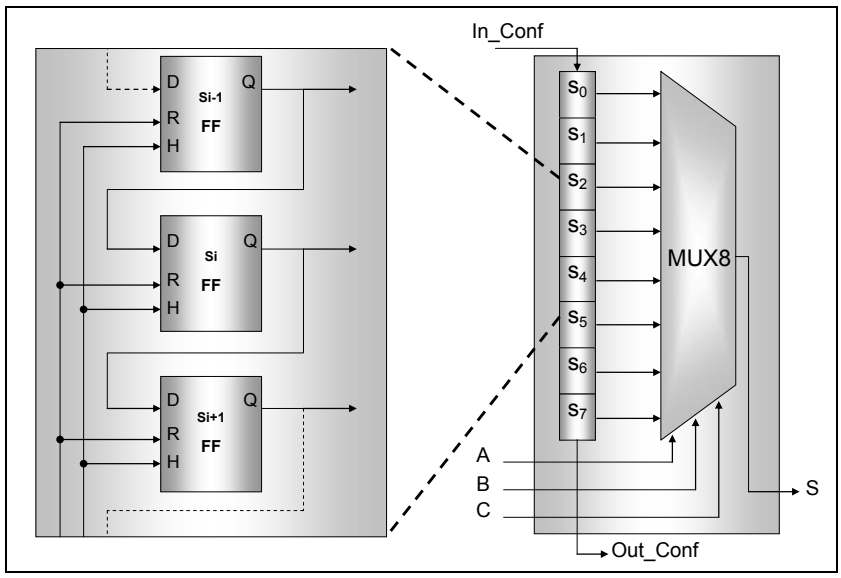

Figure II.10 : Bloc élémentaire reconfigurable (LUT3).

Dans une technologie CMOS 0,13 µm, cette cellule occupe une surface d'environ 300 µm².

La grande flexibilité de cette cellule couplée avec un routage adapté permet de remplacer n'importe quelle fonction logique combinatoire figée par une fonction logique reconfigurable.

II.2.2. Application au DES

Afin d'illustrer l'utilisation d'une telle cellule, nous allons l'instancier dans une fonction afin de rendre cette dernière personnalisable. Admettons que le concepteur d'un circuit souhaite rendre personnalisable sa fonction de chiffrement

DES afin d'obtenir un algorithme de chiffrement propriétaire similaire au DES mais différent du standard.

Avant de définir l'emplacement de nos cellules reconfigurables, étudions tout d'abord l'algorithme de chiffrement DES (voir § I.3.2.1). Ce chiffrement est caractérisé par le nombre d'itérations, les tables de permutations, les tables de substitutions. Le simple fait de modifier un des paramètres énoncés permet de changer complètement le résultat du chiffrement. La modification du nombre d'itérations est assez simple à réaliser et ne consiste pas à elle seule à personnaliser de façon importante la fonction DES initiale. En revanche, si nous nous attachions à rendre reconfigurables les tables utilisées pour chaque ronde, le niveau de personnalisation obtenu serait élevé. De plus, dans une implantation matérielle, ces tables SBOX correspondent à de la logique combinatoire, ce qui se prête tout à fait à l'utilisation de cette solution.

A titre d'exemple, étudions comment personnaliser les tables SBOX. Une fonction SBOX est une permutation compressive de 48 vers 32. Ce qui signifie qu'une combinaison de 48 bits en entrée donne une combinaison de 32 bits en sortie. Décomposons la fonction SBOX afin de définir une des architectures possibles. La fonction SBOX est composée de 8 sous-SBOX (combinaisons 6 vers 4) que nous nommons SBOXi. Chaque SBOXi peut se décomposer en 4 CLB (pour *Customizable Logic Block*) qui représentent une combinaison de 6 vers 1. Ce CLB peut lui-même se décomposer en 8 LUT3. La Figure II.11 représente une architecture capable de recréer la fonction CLB. L'assemblage de 4 CLB permet de produire la fonctionnalité SBOXi. Cette structure SBOXi peut elle-même être dupliquée huit fois afin de créer la fonction SBOX du DES (combinaison 48 vers 32), à la différence près que ce bloc est maintenant reconfigurable à volonté et capable de reproduire n'importe quelle table de substitution 48 vers 32. Nous nommerons ce bloc RSBOX pour *Reconfigurable* SBOX.

Remarquons que dans cette architecture, tous les ports de configuration sont chaînés entre eux. Le *bitstream* que nous devrons générer sera donc un *bitstream* série, dépendant de l'ordre des éléments de cette chaîne. Ceci est possible car les éléments de mémorisation de la configuration sont basés sur des bascules FF. Ce choix démontre bien que la complexité du contrôleur de configuration ne dépend pas de l'architecture lorsque des bascules FF sont utilisées.

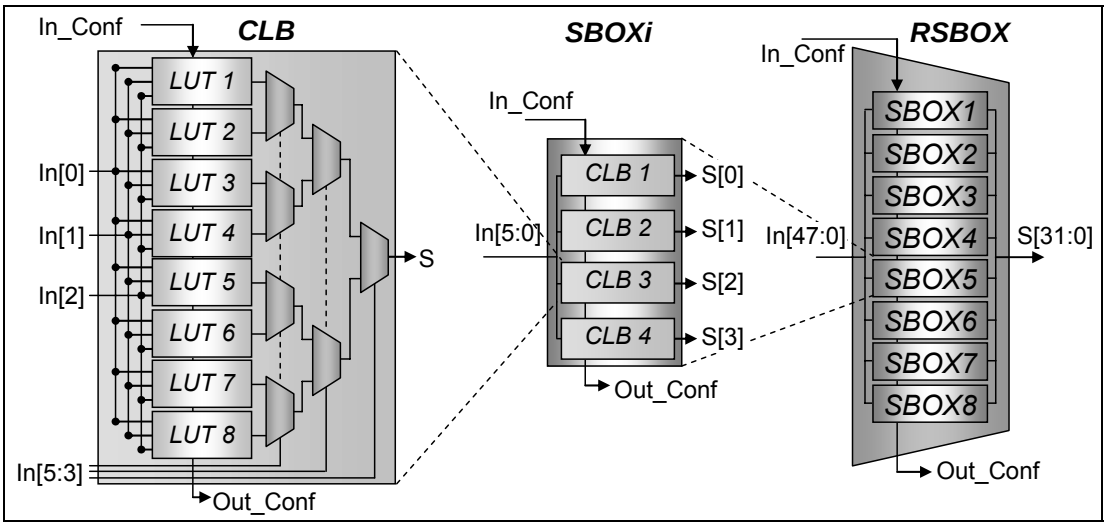

Figure II.11 : Architecture d'une RSBOX.

L'architecture de la RSBOX présentée est composée de 256 LUT3. Cela signifie que pour une configuration série, 2048 cycles sont nécessaires avant de pouvoir utiliser la fonctionnalité SBOX.

Cette architecture a été décrite à l'aide du langage de description matérielle VHDL (pour *Very high speed integrated circuit Hardware Description Language*) au niveau RTL et assemblée avec des « Ou Exclusif » afin de reproduire une ronde du DES. La Figure II.12 représente une simulation logique de cette ronde. Nous retrouvons les 2 048 cycles de configuration soit 204,8 µs pour une fréquence d'horloge de configuration de 10 MHz (Conf_clock). Après la phase de configuration, la logique combinatoire programmée est utilisée pour réaliser 6 rondes en faisant varier les données d'entrée de ronde pour deux clés différentes.

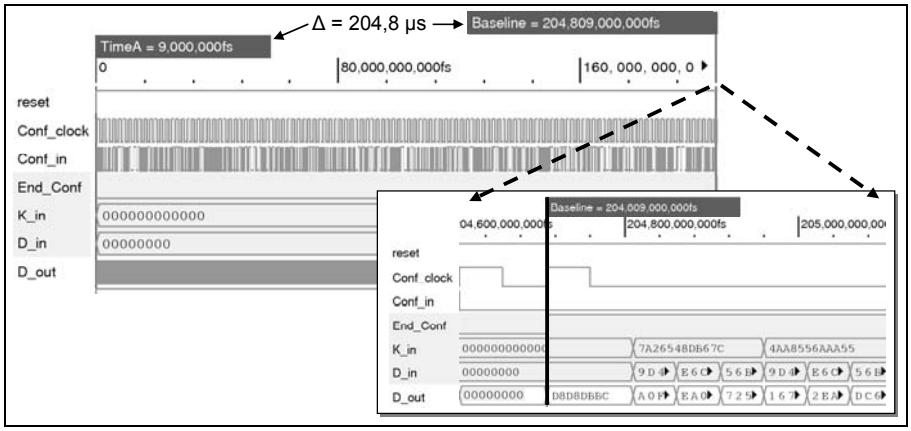

Figure II.12 : Simulation logique de la RSBOX série intégrée dans une ronde du DES.

Cette architecture n'est pas unique. Il est possible d'appliquer une configuration parallèle avec le développement du contrôleur associé. De même, l'utilisation de LUT3 n'est pas obligatoire ; l'utilisation de LUT6 équivalentes à un seul CLB peut lui être préférée.

- **Compromis temps / surface :**

Etant donné que nous avons créé des blocs reconfigurables, il n'est pas obligatoire d'instancier une RSBOX pour remplir la fonctionnalité SBOX. En effet, un bloc SBOXi peut avoir la fonctionnalité d'une SBOX1 à SBOX8 selon la configuration que nous lui appliquons. Il est donc possible de n'instancier qu'une seule SBOXi et de la reconfigurer successivement afin de reproduire les 8 fonctionnalités voulues. La Figure II.13 représente cette idée.

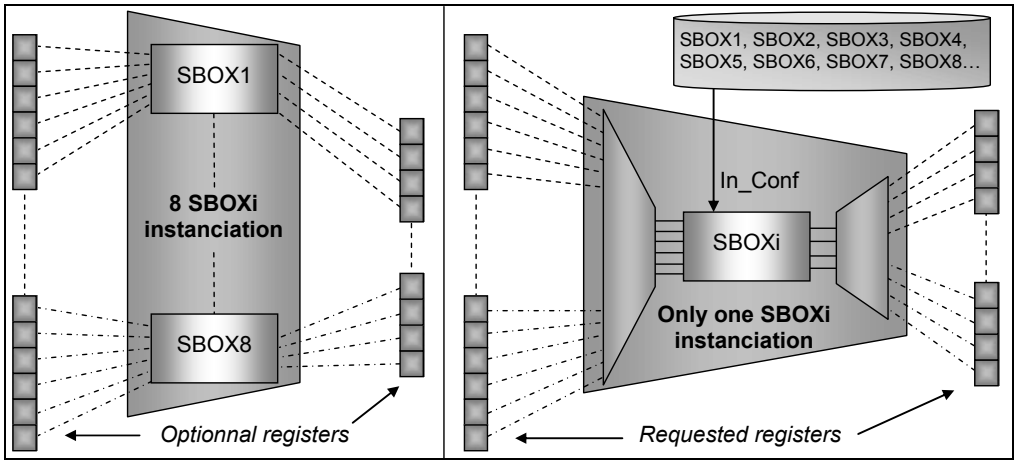

Figure II.13 : Implantation de la fonction SBOX.

L'avantage de cette seconde solution est qu'elle permet de réduire considérablement la surface. En revanche, le temps d'exécution est largement accru, d'autant plus lorsqu'une configuration série est utilisée. De plus, pour la proposition à une SBOXi, il est obligatoire d'englober la fonctionnalité SBOX entre deux séries de registres (80 au total) dont la surface n'est pas négligeable. D'autres compromis sont envisageables. Par exemple, il est possible de n'instancier qu'un seul CLB ou encore une seule LUT3.

Les résultats de ces compromis sont affichés dans le Tableau II-3 lorsqu'une configuration parallèle par mots de 8 bits est appliquée. Afin de mieux comprendre ce tableau, reprenons les résultats correspondants au bloc SBOXi (deuxième colonne). La surface correspond à une estimation suite à la synthèse du bloc décrit au niveau RTL. Cette valeur est 4 fois supérieure à celle du CLB, ce qui est cohérent puisqu'elle comporte 4 CLB. La ligne suivante précise combien de SBOXi sont nécessaires pour réaliser une ronde en un seul cycle, soit 8. La taille du bitstream est directement liée au nombre de LUT3 présentes, soit 32 dans une SBOXi. Cela donne un bitstream de 256 bits qui peut être chargé en 32 cycles avec une programmation par octets. La ligne suivante (T_{SBOX}) représente le nombre total de cycles (configuration et exécution) nécessaires pour réaliser l'équivalent d'une fonctionnalité SBOX qui a lieu pour chaque ronde. Pour la SBOXi, cela consiste à la configurer en SBOX0, à l'exécuter et à recommencer l'opération jusqu'à accomplir les 8 SBOXi. Ce nombre de cycle est donné par la relation suivante :

$$T_{SBOX} = (T_{CONF} + 1) \times N_{INST} \quad \textbf{(Equ. 7)}$$

La ligne suivante (T_{DES}) correspond cette fois à l'exécution d'un calcul DES complet, soit les 16 rondes successives, donné par la relation :

$$T_{DES} = 16 \times T_{SBOX} \quad \textbf{(Equ. 8)}$$

Cette relation n'est pas applicable pour la colonne de gauche car pour cette solution, lorsque la RSBOX est configurée, il n'est plus nécessaire de la reconfigurer.

La dernière ligne donne une indication sur la complexité du contrôleur de configuration qui n'est pas compris dans les résultats de la première ligne.

Granularity	RSBOX	SBOXi	CLB	LUT3
Area (µm²)	~ 100 000	~ 13 000	~ 3 200	~ 400
Instances for one round: N_{INST}	1	8	32	256
Bitstream for one instance (bit)	2 048	256	64	8
Configuration cycles per instance (for 1 byte words): T_{CONF}	256	32	8	1
Total cycle for SBOX execution: T_{SBOX}	257	264	288	512
Total cycle for DES execution: T_{DES}	272	4 224	4 608	8 192
Config. controller complexity	low	middle	high	high

Tableau II-3 : Compromis temps / surface.

A la vue des résultats contenus dans ce tableau, il apparaît clairement que la solution de la colonne de droite est bien trop pénalisante en termes de temps d'exécution. Les solutions présentées dans les deux colonnes du milieu ont des temps d'exécution assez proches mais la surface du CLB est 4 fois inférieure.

La solution de la colonne de gauche est celle qui est la moins pénalisante en termes de temps d'exécution. Plus le nombre de chiffrement DES est important, plus l'intérêt de cette solution devient évident malgré l'augmentation de la surface.

Il appartient au concepteur de choisir la solution qui lui est la plus adaptée.

II.2.3. Reconfiguration avancée

La solution précédente se prête à une amélioration lorsque l'intégralité de la fonctionnalité SBOX n'est pas instanciée. Si nous observions l'enchaînement des reconfigurations, nous constaterions que nous utilisons un nombre fini de configurations différentes. Ceci est visible dans la Figure II.13, dans laquelle nous enchaînons les 8 configurations correspondantes aux 8 SBOXi. Le nombre de configuration est égal à N_{INST} (deuxième ligne du Tableau II-3).

Continuons notre démonstration avec l'instanciation d'un seul bloc SBOXi, il serait intéressant de pouvoir enchaîner les configurations en un seul cycle plutôt que 32 cycles. L'utilisation de registres à décalage permet de répondre à cette attente. En

remplaçant dans la Figure II.10 chaque bascule FF par un registre à décalage (Sreg) de longueur N, il est possible de basculer d'une configuration à une autre (parmi N) en un cycle. Nous nommons la cellule ainsi créée SLUT3 (pour *Shifted* LUT3) représentée dans la Figure II.14.

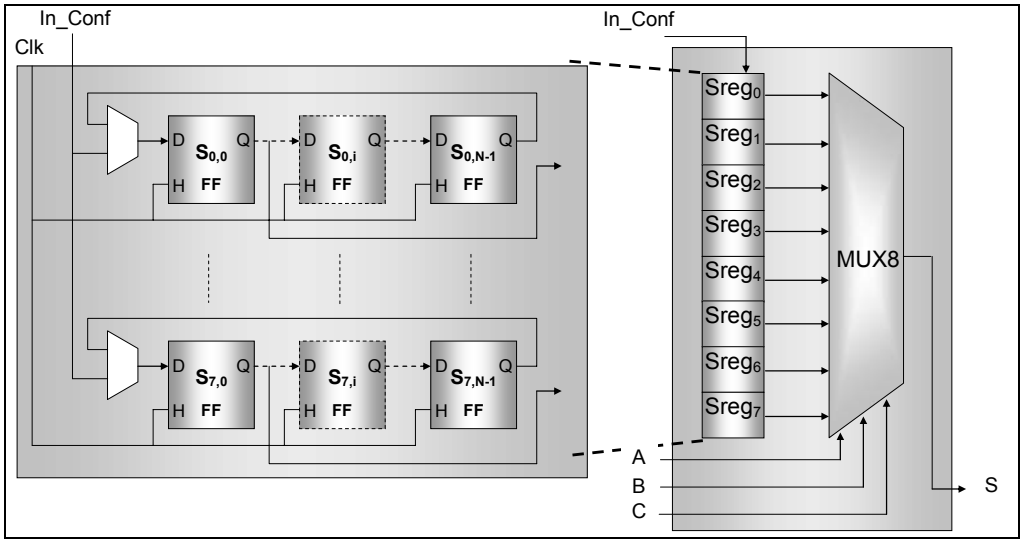

Figure II.14 : Utilisation de registres à décalage dans une SLUT3

Toujours dans le cas où nous ne souhaitons instancier qu'un seul bloc SBOXi, nous pouvons utiliser des registres à décalage de longueur 8 afin d'enchaîner les 8 configurations possibles (égal à N_{INST}). En assemblant ces SLUT3 selon la Figure II.15, nous parvenons à recréer une fonctionnalité RSBOX légèrement différente de celle présentée dans la partie précédente puisqu'elle nécessite un cycle de latence entre chaque exécution, due à la reconfiguration.

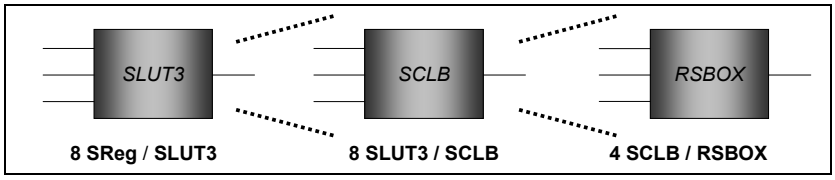

Figure II.15 : Architecture de RSBOX.

Le Tableau II-4 présente les caractéristiques d'une RSBOX avec et sans registre à décalage. L'utilisation de registres à décalage permet de diminuer la surface de 20% pour un temps d'exécution supérieur de 30%.

A titre d'information, la colonne de droite correspond à une SBOX non configurable basée sur des portes logiques CMOS 0,13 µm.

Encore une fois, le choix de la solution la plus adaptée revient au concepteur qui doit comparer le gain de performance par rapport au surcoût en surface.

Granularity	RSBOX (with SLUT3)	RSBOX (with LUT3)	Non configurable SBOX
Area (µm²)	~ 80 000	~ 100 000	~8 000
Instances for one round: N_{INST}	1	1	N/A
Bitstream for one instance (bit)	2 048	2 048	N/A
Configuration cycles per instance (for 1 byte words): T_{CONF}	256	256	N/A
Total cycle for SBOX execution: T_{SBOX}	264	257	1
Total cycle for DES execution: T_{DES}	384	272	16
Config. controller complexity	low	low	N/A

Tableau II-4 : Comparaison des caractéristiques des RSBOX avec et sans registre à décalage.

Bien que nous ayons pris le parti de ne pas développer de nouvelles cellules afin de limiter au maximum les coûts de conception, il serait préférable de concevoir une librairie de cellules LUTn ou SLUTn. En effet, cet effort de conception permettrait d'optimiser la surface ainsi que la consommation induite par l'utilisation de la logique reconfigurable.

Le choix de points mémoires basés sur des bascules FF est adapté si la reconfiguration est fréquente. Si la configuration change peu au cours de la durée de vie du produit, il pourrait être intéressant de s'intéresser au développement d'une mémoire non-volatile dédiée, dont les points correspondraient aux bascules FF. En effet, une NVM a un temps d'accès ainsi qu'une consommation importants. De plus, cela demande un effort de conception conséquent afin de créer une mémoire ayant les dimensions adaptées : nombre de bits stockés et nombre de points adressables.

II.3. Reconfiguration sécurisée

La programmation des éléments reconfigurables doit être sécurisée. En effet, imaginons qu'une personne mal intentionnée parvienne à identifier et modifier la configuration de la RSBOX (voir § II.2.2). Ceci représente une faille majeure car la cryptanalyse d'un algorithme de chiffrement similaire au DES avec des tables SBOX choisies permettrait de retrouver très facilement la valeur de la clé secrète.

II.3.1. Sécurité interne

Le DES reconfigurable présenté dans la partie précédente (voir § II.2.2) est basé sur un plan mémoire volatile (FF) qui perd son contenu pour chaque mise hors tension. Ce qui signifie que la configuration doit être présente dans le circuit dans une mémoire non-volatile (Flash ou EEPROM) et transférée lors de chaque mise sous

tension de la mémoire non-volatile vers le plan volatile. Cette opération se déroulant dans la zone de confiance du circuit, nous considérons qu'elle est sécurisée.

De même, si le plan mémoire est non-volatile, aucun transfert de données n'a lieu, ce qui signifie que le niveau de sécurité est élevé.

Dans les deux cas, si nous considérons que le stockage initial de la configuration est réalisé dans un environnement contrôlé, le haut niveau de sécurité est maintenu. Il n'est pas nécessaire de chiffrer le bitstream stocké.

En revanche, lorsque le propriétaire du circuit souhaite faire une mise à jour à distance de la configuration stockée dans le circuit, une attention particulière doit être portée afin que le bitstream ne soit pas modifié.

II.3.2. Sécurité externe

La sécurité externe correspond à la protection des données échangées qui ont lieu entre le propriétaire (ou *customer*) et l'utilisateur du circuit (ou *end-user*). Lors d'une mise à jour de son circuit, le propriétaire doit être capable d'accomplir cette tâche sans rapatrier tous ses circuits : premièrement, pour des raisons évidentes de coûts ; deuxièmement, pour ne pas perturber l'utilisateur final.

Cet échange est réalisé par l'intermédiaire de canaux non sécurisés comme représenté dans la Figure II.16.

Il existe une multitude de média très hétérogènes (débit, identification, protocoles, directions …). Afin d'être compatible avec tous ces médias, l'échange d'informations doit fonctionner pour des communications unidirectionnelles (mode simplex).

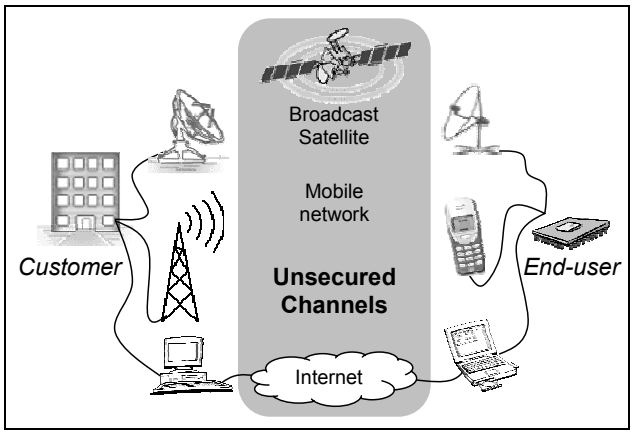

Figure II.16 : Média de communication.

De nombreuses propositions ont été faites afin de protéger le bitstream de circuits FPGA *stand-alone* mais aucune n'est applicable dans notre cas [PRO07]. Pour certaines, la mise à jour du *bitstream* n'est pas applicable car elles nécessitent que toute configuration ait lieu dans un environnement sécurisé [KEA01] et [BOS04]. D'autres sont trop restrictives au niveau des applications [KES00]. Par exemple, [VIR07] propose une solution mais une batterie doit en permanence être connectée au circuit.

De plus, aucune proposition ne se prémunit de l'attaque dite de « l'homme du milieu » qui consisterait, sans même chercher à le déchiffrer, à modifier le bitstream pour placer la matrice reprogrammable dans un état indéterminé. L'authentification permet de s'en prémunir.

II.3.3. Protocole de reconfiguration sécurisée

Nous allons proposer une solution de transfert sécurisé qui puisse s'appliquer à n'importe quel média (prise en compte des transmissions simplex) et dans laquelle les notions suivantes seront présentes :

- Confidentialité : La confidentialité permet de dissimuler les données échangées au regard des utilisateurs non autorisés. Elle est fournie par l'utilisation d'un algorithme de chiffrement (voir § I.3.1).
- Authentification : L'authentification de l'expéditeur est nécessaire afin d'empêcher que n'importe quel utilisateur puisse accéder aux zones reconfigurables (voir § I.3.3).
- Intégrité : L'intégrité est employée afin d'éviter qu'une partie d'un message soit modifiée intentionnellement ou non. Cette fonction est accomplie par l'utilisation d'un MDC (pour *Modification Detection Code*) (voir §I.3.5).
- Rapidité : Potentiellement, la quantité de données échangées est importante. Par conséquent, cela se traduit par l'utilisation d'un protocole hybride (chiffrement asymétrique et symétrique).

Ce protocole hybride doit être réalisé en deux étapes distinctes. Dans un premier temps a lieu l'échange de la clé de session générée par le *customer* et dans un second temps l'envoi du *bitstream* chiffré.

- **Echange de la clé de session K_S**

Pour ce faire, nous utilisons le principe MAC,[MEN96], qui associe un algorithme de signature numérique avec un algorithme de chiffrement asymétrique réalisé deux fois (voir Figure II.17). Tout d'abord, il convient de générer le condensé, *H*, de la clé de session à l'aide d'une fonction de hachage à sens unique. Ce condensé est ensuite concaténé avec les données de la clé de session et fournie à une fonction de chiffrement asymétrique utilisée avec la clé privée du *customer* (K_{PVC}). Cette étape ajoute une notion d'authentification à la donnée transmise en prouvant l'identité de l'émetteur car seul un utilisateur possédant la clé K_{PVC} est capable d'avoir généré ce message, soit le *customer*. Dans un second temps, le résultat est chiffré avec la clé publique de l'utilisateur final (K_{PBU}) afin de garantir la confidentialité, autorisant la transmission de ce message sur un canal non sécurisé. En effet, le résultat de ce chiffrement s'il est intercepté, ne pourra être interprété sans avoir connaissance de la clé K_{PVU} que seul l'utilisateur final possède.

Après réception du message transmis sur le canal non sécurisé, l'utilisateur final, utilise sa propre clé privée (K_{PVU}) afin de retrouver le message chiffré contenant le texte clair et son condensé. Ensuite, il applique sur ce message la clé publique du *customer* (K_{PBC}) pour obtenir la clé de session et son condensé associé. Il calcule de

nouveau le condensé de cette clé, H_{BIS}, et la compare avec celle qui était présente dans le message afin de vérifier l'intégrité de K_S.

Cela implique que le *customer* connaisse la clé publique de l'utilisateur final car il est possible que l'utilisateur ne puisse pas envoyer d'information au *customer*. De même, l'utilisateur final doit connaître la clé publique du *customer*.

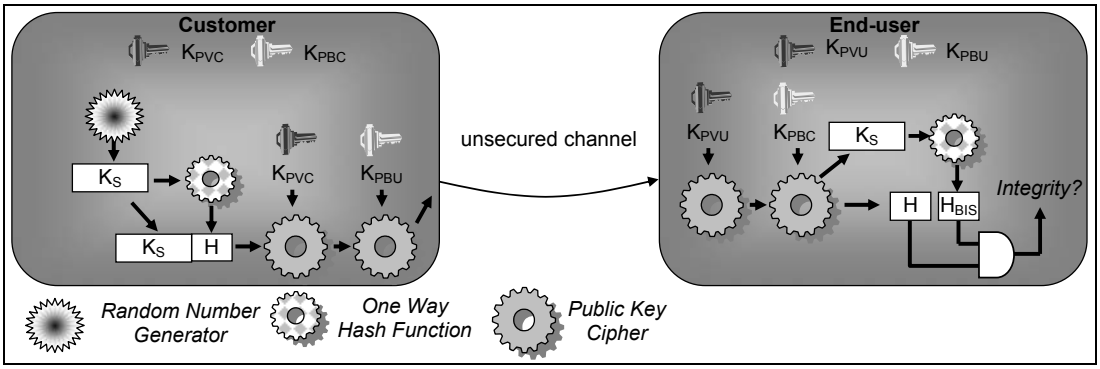

Figure II.17 : Transmission de la clé de session.

- **Transfert du *bitstream* chiffré**

L'utilisation d'algorithmes de chiffrement asymétriques nécessitant un temps de calcul très important, nous préférons utiliser un chiffrement symétrique pour envoyer le bitstream afin de gagner en rapidité.

Le flot de données correspondant au bitstream est chiffré avec la clé de session K_S que les deux utilisateurs se sont échangés dans l'étape d'échange de clé. Tout comme précédemment, nous appliquons le principe MAC aux messages transmis sur le canal non sécurisé en ajoutant le condensé au *bitstream* (voir Figure II.18) mais cette fois-ci, une seule étape de chiffrement est réalisée. Le récepteur compare toujours le condensé intégré aux messages reçus avec celui qu'il génère lui-même afin de vérifier l'intégrité du *bitstream*.

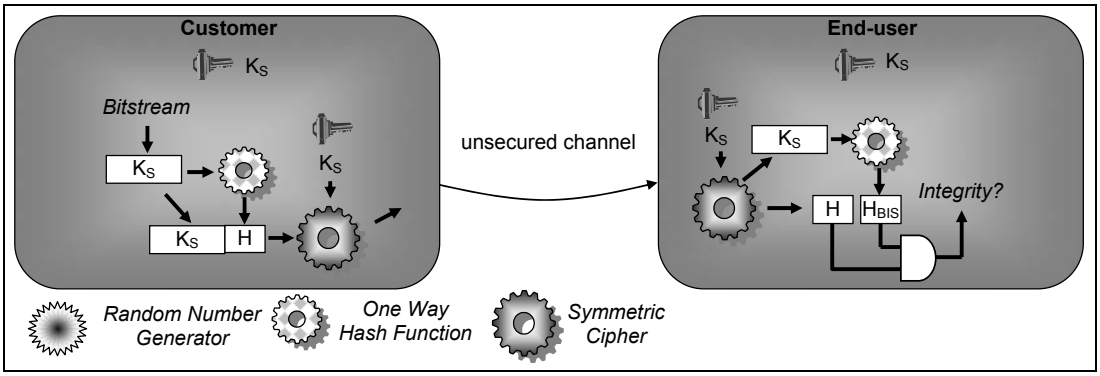

Figure II.18 : Chiffrement et transmission du bitstream.

L'utilisation d'un tel protocole hybride permet de changer régulièrement de clé de session. Ceci permet de limiter le nombre de messages chiffrés avec une même clé et de perturber un attaquant qui tenterait de collecter un grand nombre de messages chiffrés.

Les fonctions utilisées dans ces deux phases (fonction de hachage, chiffrement asymétrique et symétrique) sont généralement présentes dans un circuit sécurisé soit sous la forme de macro-cellules, soit à l'aide du CPU.

De multiples algorithmes et longueur de clé peuvent s'appliquer à ce protocole ; nous pouvons les choisir en fonction de ce que la plate-forme utilisée est capable d'accomplir. Par exemple, nous pouvons employer :

- pour l'échange de clés : compression, RSA avec une clé de 2048 bits en mode de fonctionnement CBC, SHA-1 avec un condensé de 160 bits, et un remplissage des messages avec des valeurs aléatoires.
- pour l'envoi de données : compression, AES avec une clé de 128 bits en mode de fonctionnement CBC, SHA-1 avec un condensé de 160 bits, et un remplissage des messages avec des valeurs aléatoires.

Il convient de compresser les données avant de les chiffrer afin de supprimer la redondance des données dans un message car cette propriété est exploitée par la cryptanalyse.

II.4. Synthèse

Dans ce chapitre nous avons présenté les concepts nécessaires à l'intégration de logique reconfigurable dans un circuit sécurisé afin de répondre au mieux aux besoins des industriels en sécurité flexible.

Dans la première partie de ce chapitre, nous avons réalisé un rapide état de l'art des technologies reconfigurables existantes, sans oublier les inévitables FPGA, en concluant que l'utilisation de circuits dédiés est une solution qui ne possède pas un niveau de sécurité suffisamment élevé. Ce choix présente d'importantes failles, notamment lors de la mise à jour de la matrice reconfigurable. L'utilisation de cœurs reconfigurables n'est, elle aussi, pas envisageable pour des raisons économiques essentiellement mais aussi sécuritaires. Dans un contexte industriel de conception de circuits sécurisés, le paramètre financier occupe une place importante dans le choix des solutions. En effet, le marché des circuits sécurisés n'est pas demandeur de sécurité flexible si le coût en est trop élevé.

Pour cette raison, dans la suite de ce chapitre, nous avons étudié l'utilisation de logique reconfigurable répartie qui ne possède pas les inconvénients des deux solutions énoncées précédemment. En revanche, elle ne peut s'utiliser qu'avec une plate-forme déjà existante. Ce qui en soit est un avantage si l'industriel qui souhaite implanter de la sécurité flexible possède déjà une plate-forme robuste et éprouvée. De nombreuses directions ont été étudiées tant au niveau de l'architecture, des dimensions de l'élément reconfigurable de base, du type de reconfiguration (parallèle ou série) ainsi qu'au niveau des compromis temps d'exécution / surface. Les choix doivent être pris par le concepteur du circuit sécurisé en fonction des besoins auxquels il souhaite répondre ainsi que des contraintes économiques. La puissance de la solution que nous proposons est qu'elle peut s'adapter à n'importe quelle partie combinatoire d'un circuit. De plus, elle s'adapte parfaitement aux domaines des

circuits sécurisés en répondant parfaitement aux problèmes énoncés dans l'introduction de ce chapitre.

Enfin, nous avons achevé ce chapitre en présentant un protocole de reconfiguration des éléments programmables. La logique reconfigurable couplée avec ce protocole robuste et rapide permet aux concepteurs de circuits sécurisés d'investiguer de nouvelles voies pour déjouer les tentatives de clonage et de retro-ingénierie.

Dans la suite de ce manuscrit, nous allons maintenant nous focaliser sur un autre problème que rencontrent les industriels du marché des circuits sécurisés : les attaques par analyse des informations qui fuient par l'intermédiaire des canaux cachés, et plus particulièrement les analyses du courant consommé.

CHAPITRE III.

ANALYSE DIFFERENTIELLE DU COURANT CONSOMME

Dans ce chapitre, nous traiterons des attaques menées contre les circuits intégrés par analyse différentielle du courant consommé, nommée *DPA* (pour *Differential Power Analysis*). Cette attaque, basée sur le fait que la puissance consommée par un circuit intégré est fortement corrélée avec les données que celui-ci traite, permet aux attaquants de retrouver les clés de chiffrement confidentielles stockées dans ces circuits. Dans un premier temps, nous nous attacherons à établir un modèle réaliste mettant en évidence cette dépendance de la consommation vis à vis des données. Le modèle que nous utiliserons sera plus proche de la réalité que ceux présents dans la littérature ([MES99], [AIG07], [GUI04]) car il prend en compte plus de capacités parasites. Suite à cela nous présenterons en détail l'analyse *DPA* et prendrons l'algorithme DES comme exemple. Enfin, nous présenterons les contre-mesures possibles pour déjouer cette attaque en nous attachant tout particulièrement aux contre-mesures matérielles.

III.1. La technologie CMOS

III.1.1. Les transistors MOS

Les circuits intégrés actuels sont principalement basés sur la technologie *CMOS*, qui consomme peu et dont les procédés de conception sont largement maîtrisés. En effet, les transistors *CMOS* consomment seulement lors de la commutation. Autrement dit, dans le cas le plus répandu des circuits numériques synchrones, la principale consommation apparaît lors des fronts d'horloge.

Les transistors *MOS* les plus répandus sont les transistors à Effet de Champs (*MOSFET* pour *MOS Field Effect Transistor*) à enrichissement. Ils sont composés de trois ports : la grille, le drain et la source (voir Figure III.1.a). La caractéristique de ces transistors est que la grille métallique est complètement isolée du substrat par un diélectrique de type *SiO2* (Dioxyde de Silicium). Lorsque la tension adéquate est

appliquée sur la grille, le champ électrique ainsi généré crée un canal conducteur dans le substrat (voir Figure III.1.b).

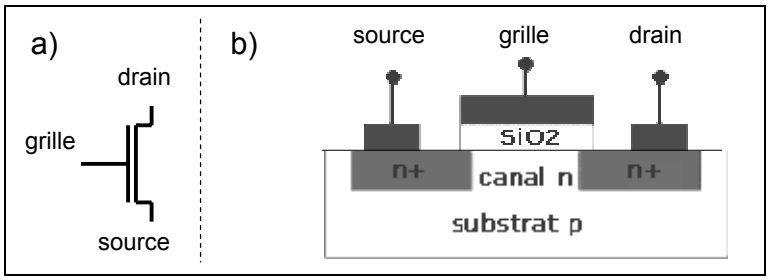

Figure III.1 : Transistor NMOS : a) symbole et b) implantation.

Ils fonctionnent globalement comme un interrupteur. Dans les transistors dits de type *N*, le canal qui contient un excédent d'électrons devient conducteur lorsque la tension de grille est positive et supérieure à la tension de seuil. Inversement pour les transistors de type *P*, le canal qui contient un excédent de trous devient passant lorsque la tension de grille est nulle. Un transistor est dit passant lorsque le canal est conducteur et bloqué lorsque le canal n'est pas conducteur.

III.1.2. Porte inverseuse

En technologie *CMOS*, les transistors sont implantés de façon complémentaire pour réaliser les fonctions logiques. La porte la plus élémentaire est l'inverseur *CMOS* représenté dans la Figure III.2. Il est composé de deux transistors *MOSFET* : un *PMOS* dont la source est reliée au potentiel V_{dd}, et un *NMOS* dont la source est reliée au potentiel V_{ss}. Dans cette figure, V_i et V_o représentent respectivement les tensions d'entrée et de sortie.

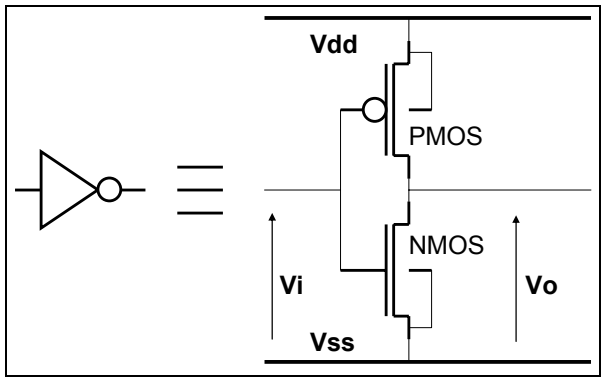

Figure III.2 : Inverseur CMOS.

- **Comportement statique**

Etudions le comportement d'une telle porte tout d'abord du point de vue de la consommation électrique. La Figure III.3 représente schématiquement à l'aide d'interrupteurs, le fonctionnement statique de cet inverseur. C'est-à-dire lorsque la porte est dans l'un de ses deux états d'équilibre. Dans cette représentation, il est

clairement visible qu'il n'existe aucun chemin entre V_{dd} et V_{ss} donc qu'aucun courant n'est consommé.

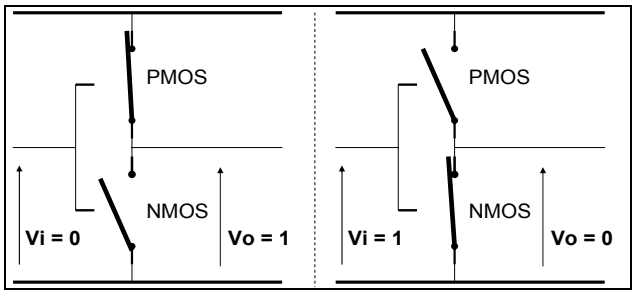

Figure III.3 : Représentation du comportement statique d'un inverseur CMOS.

La Figure III.4 représente la fonction de transfert d'un inverseur équilibré. Un inverseur équilibré est un inverseur dans lequel le rapport $K_P=W_P/L_P$ du transistor *PMOS* est deux fois supérieur au rapport $K_N=W_N/L_N$ du transistor *NMOS*. Ceci est dû au fait que les zones dopées *P* sont deux fois plus résistives que les zones dopées *N*. Nous retrouvons les deux états où $V_i = 0$ et $V_i = 1$, respectivement dans les phases 1 et 5. L'observation du courant entre V_{dd} et V_{ss}, noté *I* fait apparaître une consommation dans les phases 2 à 4. Ceci est dû au fait que V_{dd} et V_{ss} sont reliés ; ce phénomène est appelé courant de court-circuit. Il est à noter que dans la fonction de transfert idéale de l'inverseur, les zones 2 et 4 sont réduites et que la zone 3 disparaît.

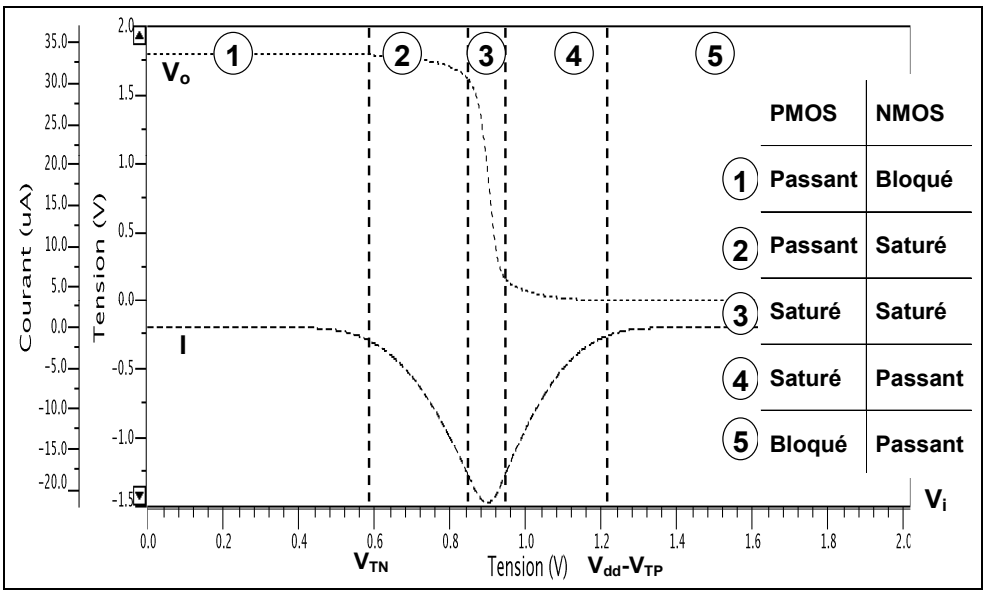

Figure III.4 : Fonction de transfert d'un inverseur.

- **Comportement dynamique**

Dans cette partie, nous allons établir un modèle du comportement dynamique de cette même porte sensiblement différent des modèles couramment utilisés afin de mettre en évidence la relation entre les données traitées et la consommation. Nous pourrons ainsi visualiser les consommations électriques et l'importance des courants de court-circuit par rapport à la consommation globale.

En régime dynamique un phénomène supplémentaire est à prendre en considération : les capacités parasites intrinsèques à la technologie *MOS*. La grille d'un transistor *MOS* se comporte comme une capacité composée de :

- une capacité entre la grille et la source : C_{GS},
- une capacité entre la grille et le drain : C_{GD},
- une capacité entre la grille et le substrat : C_{GB}.

Nous prenons l'hypothèse simplificatrice de remplacer ces trois capacités par une seule et unique. Ainsi, nous retrouvons dans la modélisation d'un inverseur *CMOS* (Figure III.5) deux capacités parasites : respectivement C_P et C_N pour les transistors *PMOS* et *NMOS*. La valeur de ces capacités parasites étant proportionnelle aux paramètres *W* et *L* du transistor, C_P est deux fois plus importante que C_N dans le cas d'un inverseur équilibré. Dans la littérature, il est courant de négliger C_N, cependant nous conserverons cette capacité parasite afin de produire un modèle plus réaliste. Cette différence aura un impact sur les courants électriques observés dans la Figure III.6.

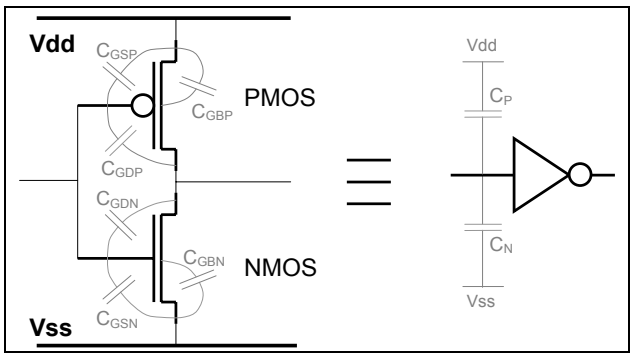

Figure III.5 : Capacités parasites en entrée de l'inverseur CMOS.

La Figure III.6 montre la transition montante et descendante en sortie de l'inverseur, respectivement sur la partie haute et basse de la figure. Notre modèle est constitué d'un transistor chargé par les capacités parasites d'entrée de l'étage suivant. Ces simulations ont été réalisées avec le simulateur électrique eldo [MENb].

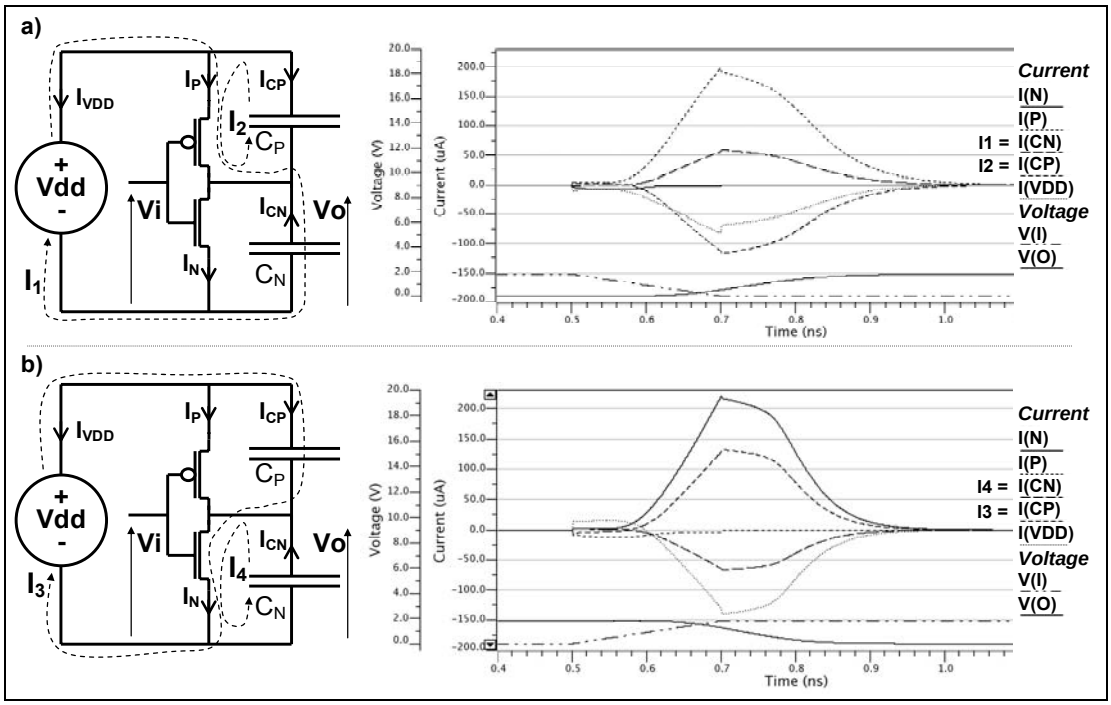

Figure III.6 : Transition montante (a) et descendante (b) en sortie d'un inverseur.

Cette figure met en évidence les charges et décharges des capacités de sortie. Lors de la transition montante en sortie (partie haute de la figure), un courant I_{CN} apparaît pour charger la capacité C_N et traverse le transistor P ; ce courant I_{CN} est égal à I_1. De la même manière, un courant I_{CP} apparaît lors de la transition descendante (partie basse de la figure) de la sortie pour décharger C_P à travers le transistor N ; ce courant I_{CP} est égal à I_3. Dans un inverseur équilibré, la résistivité des transistors P et N étant équivalente, la différence entre les deux courants I_1 et I_3 est due au fait que la capacité C_P est deux fois supérieure à C_N.

Il est important de noter que si nous avions supprimé cette capacité C_P, le courant I_{CP} serait nul. Cela se traduirait par une grande différence entre les courants I_1 et I_3. Cependant, notre modèle se rapproche au mieux de la réalité en conservant cette capacité parasite C_P.

Ainsi, en observant seulement les courants qui traversent les transistors (I_P et I_N), nous pouvons clairement voir une asymétrie et distinguer les transitions montantes et descendantes. Par conséquent, si nous avions un accès illimité aux nœuds internes du circuit, nous pourrions aisément distinguer les transitions montantes des transitions descendantes en sortie de l'inverseur. Malheureusement, cette assertion faite dans bien des modèles n'est aussi pas réaliste.

Par souci d'établir un modèle de consommation le plus réaliste possible, nous considérerons que nous n'avons accès qu'au courant $I_{VDD}(t)$.

Ainsi, en observant seulement le courant $I_{VDD}(t)$, nous remarquons une différence notable dans les profils de consommation mais qui ne permet pas à elle seule de distinguer une transition montante d'une transition descendante. $I_{VDD}(t)$ ne

présente pas d'asymétrie suffisamment importante pour une bonne raison : dans les deux cas I_2 et I_4 ne se retrouvent pas au niveau du courant $I_{VDD}(t)$.

Nous pouvons donc dire que la consommation de la porte est liée au fait qu'une transition se soit produite. Nous verrons par la suite que ce phénomène se produit toujours lorsque plusieurs transistors sont simulés.

Si nous avions utilisé un modèle classique sans la capacité parasite C_N, l'observation du courant I_{VDD} permettrait de distinguer les charges et décharges du nœud de sortie et de deviner de façon évidente l'état du nœud de sortie.

Grâce à la Figure III.6, nous pouvons faire une dernière constatation. Le courant de court-circuit, décrit dans la partie précédente, est négligeable comparé aux courants de charge et décharge des capacités. Ce courant qui se retrouve dans le transistor *N* (respectivement *P*) dans le cas d'une transition montante (respectivement descendante) en sortie, représente moins de 10% du courant I_{VDD}.

III.1.3. Réseau d'inverseurs

Modélisons maintenant un réseau d'inverseurs *CMOS* afin de généraliser les observations que nous venons de faire.

Dans ce cas là, d'autres capacités sont à prendre en compte, et principalement les capacités induites par le routage entre les portes dites capacités de *cross-talk*. Ces dernières sont liées au fait que les canaux de routage sont proches les uns des autres. Les capacités de *cross-talk* entre des fils de connexion n'ont pas le même impact. Elles dépendent des niveaux de métallisation des fils considérés. En effet, deux fils situés sur le même niveau de métal auront une capacité de *cross-talk* importante tandis que deux fils situés sur deux niveaux de métaux adjacents auront une faible capacité. Ainsi, si nous étudions le fil 22 de la Figure III.7 (*wire 22*), les capacités C_{CT1} et C_{CT3} sont bien supérieures aux capacités C_{CT2} et C_{CT4}. Dans notre modèle, nous négligerons ces deux dernières capacités.

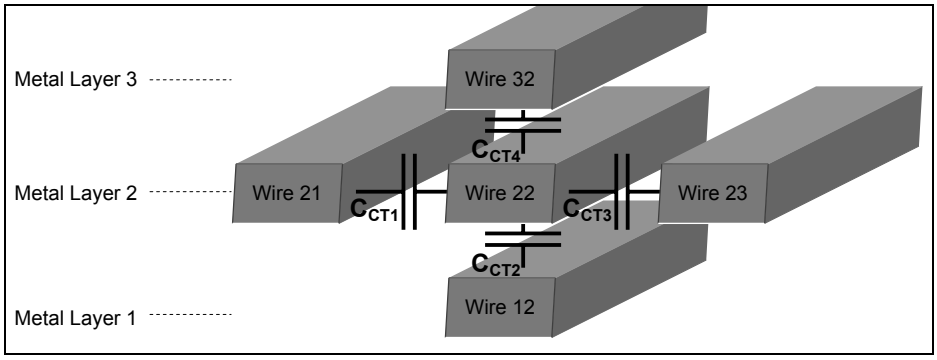

Figure III.7 : Capacités de cross-talk.

La Figure III.8 représente le modèle de réseau à quatre inverseurs que nous allons utiliser pour mettre en évidence le phénomène de corrélation entre les données et la consommation. Nous retrouvons dans ce modèle les inverseurs CMOS, les capacités de *cross-talk* et les capacités parasites d'entrée de l'étage suivant. Les valeurs des éléments correspondent aux valeurs classiques d'une technologie *CMOS* ayant une largeur de grille de 0,13 µm. Les valeurs des capacités parasites C_P et C_N de

chaque inverseur sont respectivement de 6 fF et 3 fF, pour tenir compte de la topologie de l'étage suivant pour lequel nous prenons l'hypothèse d'inverseurs équilibrés. Arbitrairement, les valeurs des capacités de *cross-talk* (CTi) sont les mêmes entre chaque fil de connexion alors que ceci n'est pas tout à fait vrai dans la réalité. Nous faisons aussi apparaître la résistivité induite par le routage (Ri) avec des valeurs réalistes pour une technologie d'intégration de 0,13 µm.

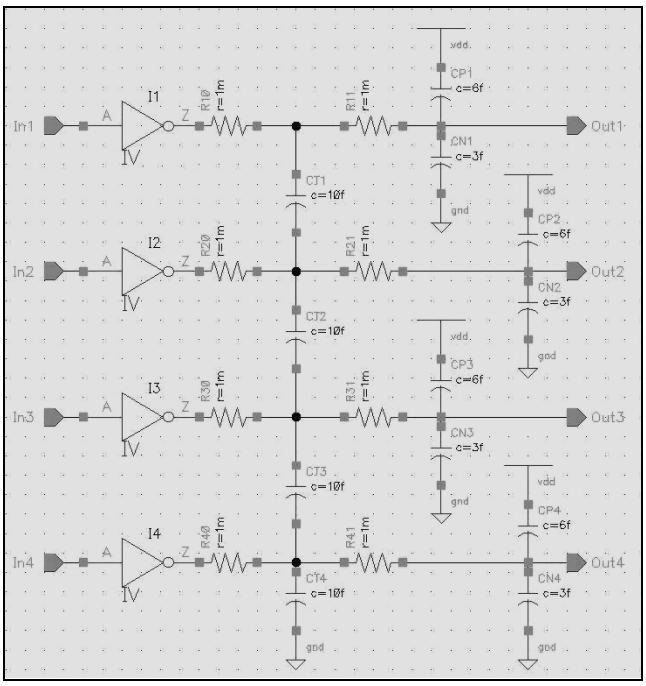

Figure III.8 : Modélisation d'un réseau d'inverseurs.

Comme vu dans la partie précédente, la consommation d'un inverseur est liée à l'occurrence d'une transition quelle qu'elle soit en sortie. Nous allons vérifier par simulation si ce phénomène se retrouve dans un réseau qui comporte des capacités de *cross-talk*. Pour cela, simulons toutes les transitions possibles dans ce réseau d'inverseur. Il existe $2^4 \times 2^4 = 2^8 = 256$ combinaisons possibles. La Figure III.9 représente la superposition des 256 courbes de consommation instantanée $I_{VDD}(t)$ correspondantes aux 256 simulations électriques.

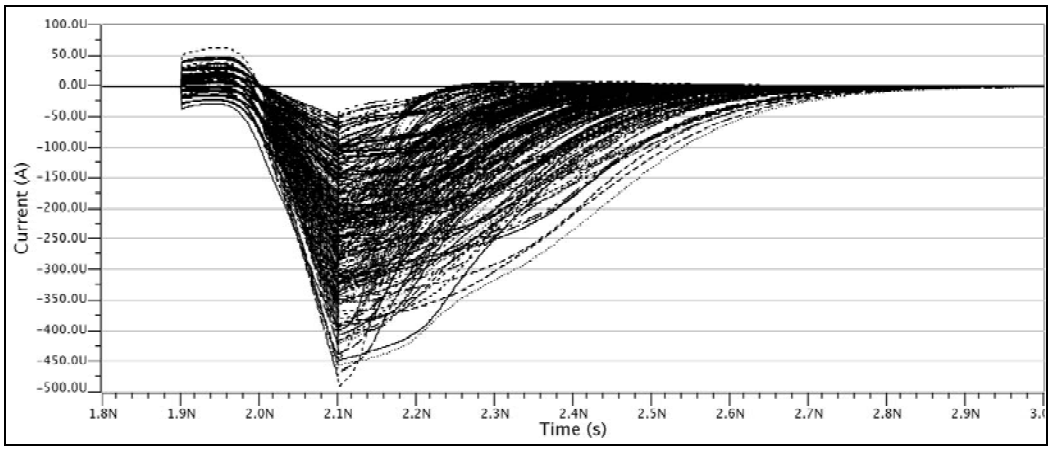

Figure III.9 : Superposition de toutes les courbes de consommation.

Cette superposition ne fait rien apparaître car la dispersion des courbes est élevée. Nous pouvons seulement constater qu'il existe un écart relativement important entre la transition qui induit le pic de consommation le moins important et celle qui induit le plus grand pic.

III.1.3.1. Influence des transitions sur la consommation

Ordonnons les courbes de façon à faire apparaître le nombre de transitions qui apparaissent. Pour cela, appelons T le nombre de transitions qui apparaissent en sortie du réseau, défini par la relation suivante :

$$T = \sum_{n=0}^{n=m-1} \left(b_{n,t-1} \neq b_{n,t} \right) \qquad \textbf{(Equ. 9)}$$

où m représente le nombre de bits du vecteur de sortie et t le temps.

Par exemple, la simulation d'une transition du vecteur de sortie de la valeur "*1101*" vers "*0100*" présente deux bits différents. La courbe correspondante sera donc placée dans l'ensemble pour lequel $T=2$. Dans la partie gauche de la Figure III.10, nous avons classé toutes les courbes $I_{VDD}(t)$ en fonction de cette valeur T, soit en 5 ensembles allant de 0 à 4. Ce premier classement fait apparaître des regroupements effectivement liés au nombre de transitions qui apparaissent dans le réseau. En effet, la dispersion des courbes dans chaque ensemble est moins importante que celle des courbes de la Figure III.9.

Dans un second temps, nous avons calculé les courbes moyennes pour chaque ensemble puis nous les avons superposées dans la partie droite de la Figure III.10. Il est intéressant de noter que les courbes moyennes sont relativement démarquées les unes par rapport aux autres.

Ce regroupement fait apparaître que les courbes peuvent être regroupées en fonction du nombre de transitions qui sont intervenues. Cela permet de mettre en évidence le phénomène de dépendance entre les données traitées et la consommation.

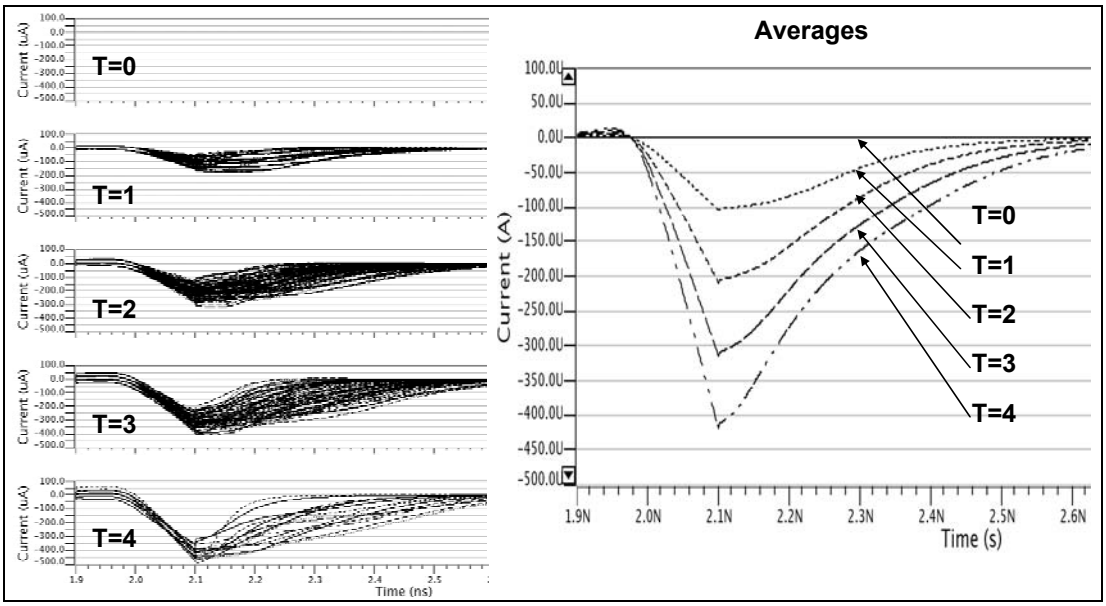

Figure III.10 : Classement des courbes en fonctions de l'activité en sortie des inverseurs.

Raisonnons maintenant en sens inverse ; est-il possible de retrouver le nombre de transitions qui ont lieu dans le réseau en observant seulement le courant $I_{VDD}(t)$ de ce circuit ? Les propriétés énoncées précédemment à savoir faible dispersion des courbes et bonne différentiation des moyennes, sont intéressantes pour retrouver le nombre de transitions. En effet, nous pouvons déduire avec une forte certitude que dans une simulation comportant un pic de consommation à -200 µA, deux transitions se sont produites. Certes ce genre d'affirmation peut sembler inutile mais dans le cas d'un circuit sécurisé, cela peut s'avérer très utile pour un attaquant. Nous verrons par la suite comment cette fuite d'information peut être utilisée par des attaquants.

III.1.3.2. Influence de la distance de Hamming

Etudions d'autres règles de classement des courbes de consommation pour voir si nous pouvons obtenir des informations plus subtiles sur l'état interne d'un circuit.

Classons les courbes de consommation en fonction de la distance de Hamming en sortie du réseau, notée *D*, *D* variant de -4 à 4 et définie par l'équation suivante :

$$D = \sum_{n=0}^{n=m-1} \left(b_{n,t-1} = 1\right) - \sum_{n=0}^{n=m-1} \left(b_{n,t} = 1\right) \qquad \textbf{(Equ. 10)}$$

La déviation des courbes dans chaque ensemble est importante (voir Figure III.11, partie gauche). De plus, les courbes moyennées (voir Figure III.11, partie droite) ne se démarquent pas bien les unes des autres. Ceci s'explique par le fait que la distance de Hamming classe dans le même ensemble une simulation dans laquelle aucune transition en sortie n'a eu lieu et une simulation pour laquelle quatre transitions sont apparues. Par exemple, dans l'ensemble D = 0 nous retrouvons les simulations "0000"→"0000" et "0101"→"1010".

Ces résultats correspondent à la constatation faite dans la partie III.1.2 dans laquelle nous présentions qu'il n'est pas évident de distinguer une transition montante d'une transition descendante. Ainsi, l'observation du courant $I_{VDD}(t)$ ne permet pas de retrouver la distance de Hamming en sortie du réseau avec une certitude suffisante.

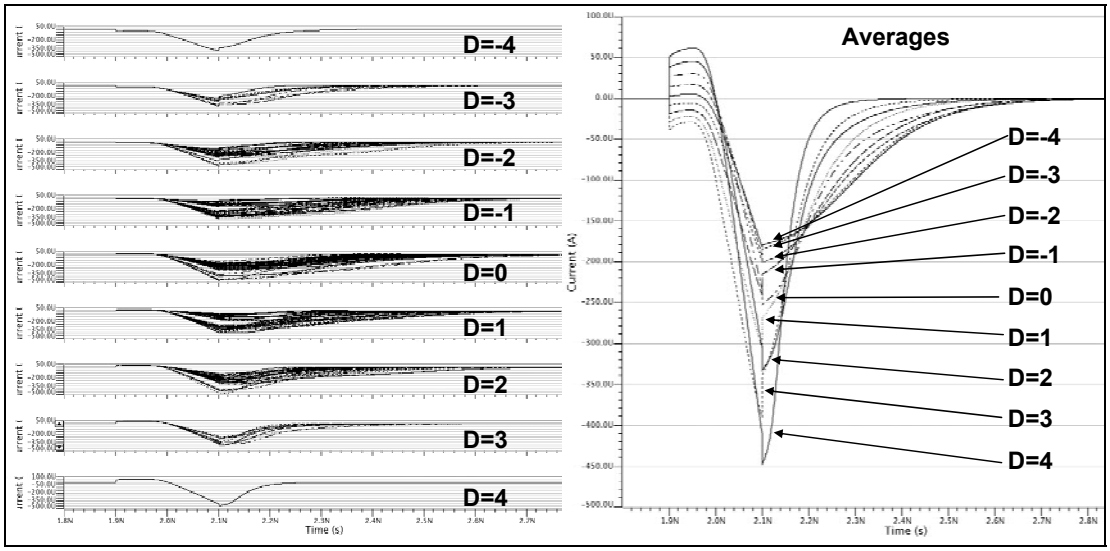

Figure III.11 : Classement en fonction de la distance de Hamming en sortie du réseau.

Etudions un dernier classement se basant sur la valeur absolue de la distance de Hamming des bits de sortie du réseau, défini par la relation suivante :

$$|D| = \left| \sum_{n=0}^{n=m-1} \left(b_{n,t-1} = 1\right) - \sum_{n=0}^{n=m-1} \left(b_{n,t} = 1\right) \right| \qquad \textbf{(Equ. 11)}$$

La dispersion des courbes dans chaque groupe est assez importante (voir Figure III.12, partie gauche). De plus les courbes moyennées (voir Figure III.12, partie droite) pour $|D|=0$ et $|D|=1$ présentent un écart assez faible. Cela signifie que l'observation du courant $I_{VDD}(t)$ d'un tel réseau ne permet pas de retrouver le poids de Hamming en sortie de ce réseau pour les mêmes raisons que le classement précédent.

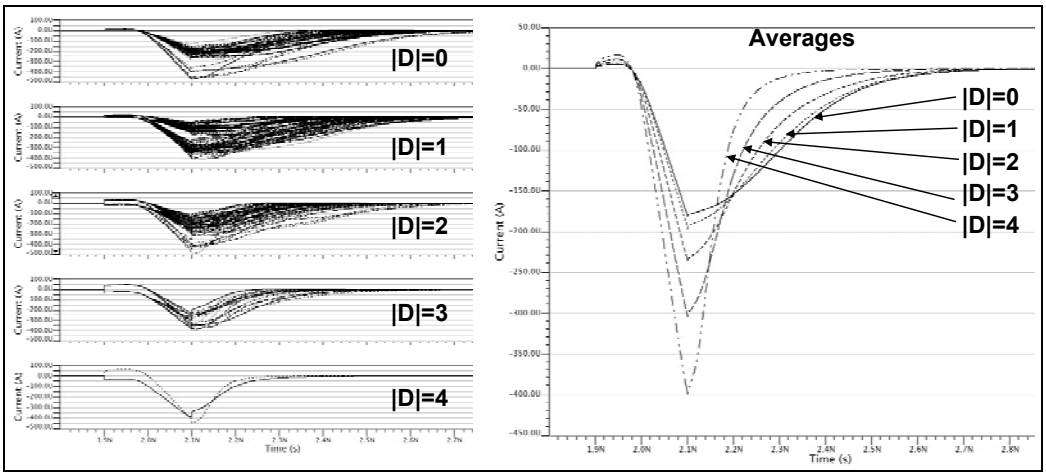

Figure III.12 : Classement en fonction de la valeur absolue de la distance de Hamming en sortie du réseau.

De tous les classements que nous venons d'étudier, le plus pertinent pour retrouver des informations sur l'état interne des nœuds reste celui basé sur le nombre de transition en sortie du réseau (voir Figure III.10).

III.1.3.3. Influence de la symétrie de l'inverseur :

Dans la partie précédente, nous avons montré que le courant consommé par un réseau d'inverseurs est intimement lié aux données que celui-ci traite. Il est notamment possible de déduire des informations relatives à l'activité en sortie du réseau (voir Figure III.10). Etudions maintenant l'influence de l'asymétrie des inverseurs sur cette assertion. La Figure III.13 représente les courbes moyennes pour les différentes valeurs possibles de T (voir § III.1.3.1). La partie gauche correspond à un réseau d'inverseurs équilibrés et la partie droite à des inverseurs symétriques. Rappelons qu'un inverseur symétrique est constitué de transistors *P* et *N* de mêmes dimensions. Les résultats sont similaires à ceux observés précédemment. Il est toujours possible de déduire des informations relatives à l'activité du circuit en fonction de l'observation du courant $I_{VDD}(t)$ seulement.

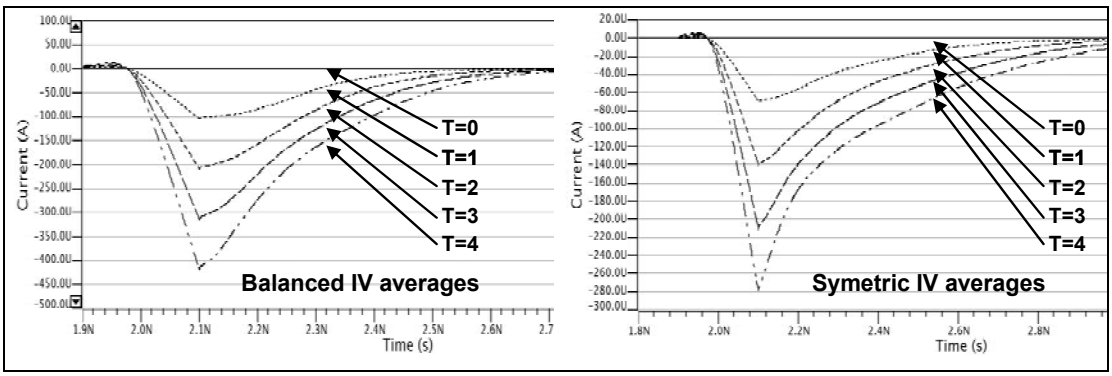

Figure III.13 : Comparaison de réseaux d'inverseurs équilibrés et symétriques.

Il est établi que dans un circuit *CMOS* le modèle de consommation de toute porte logique peut s'assimiler à la consommation d'un inverseur [NIK99], [AUV00]

et [MAU01]. Ainsi, nous pouvons conclure que le phénomène que nous observons dans un réseau de quatre inverseurs peut se généraliser à l'ensemble d'un circuit *CMOS*.

III.2. Attaque DPA

L'attaque par analyse différentielle de la puissance consommée, communément appelée *DPA*, d'un circuit a été présentée pour la première fois par Paul Kocher en 1999 [KOC99]. Elle repose sur une hypothèse faite sur la valeur d'une partie de la clé secrète interne du circuit à un instant donné et sur une analyse statistique des courbes de consommation mesurées, en fonction de cette hypothèse. A la fin de l'analyse, la vérification de l'hypothèse est réalisée ; si elle s'avère être fausse, l'analyse est recommencée avec une nouvelle hypothèse. Elle exploite le fait que dans un circuit intégré, le courant consommé est lié à l'activité des noeuds internes, autrement dit aux données manipulées.

III.2.1. Concepts

L'idée générale (voir Figure III.14) est de deviner une valeur inconnue interne au circuit à l'aide d'une hypothèse et de comparer si l'activité du circuit corrobore bien avec cette hypothèse. Nous allons dans cette partie présenter les concepts nécessaires à la compréhension de cette attaque.

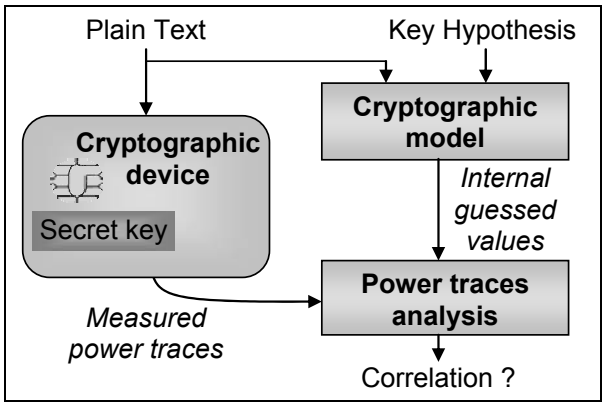

Figure III.14 : Principe de l'analyse DPA.

Etudions comment appliquer ce principe sur un circuit simple (voir Figure III.15) que nous étendrons par la suite à une fonction plus complexe. Supposons un circuit intégré composé de portes logiques. Il possède deux entrées *A* et *B* codées chacune sur 4 bits et un signal interne *S* inaccessible de l'extérieur. Nous considérons dans ce circuit seulement le cône logique ayant la fonctionnalité *F* définie par *S=F(A,B)*.

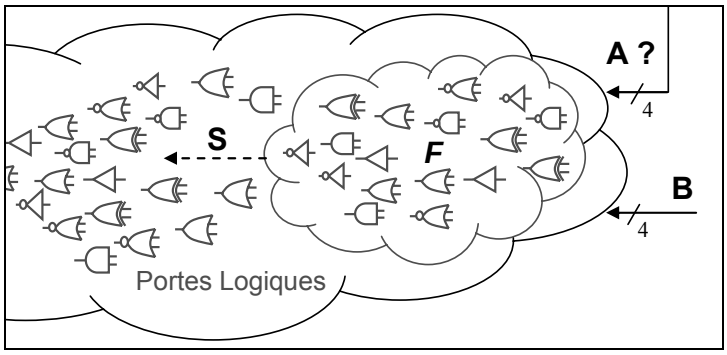

Figure III.15 : Représentation d'un système analysé.

Admettons certaines hypothèses avant de nous lancer dans l'analyse de ce circuit :

- *A* est définie comme la variable inconnue, codée sur 4 bits, que nous cherchons à deviner ; elle est supposée fixe.
- *B* est une variable d'entrée du système, codée sur 4 bits, connue et sur laquelle nous avons un accès illimité.
- *F* est une fonctionnalité connue mais dont l'implantation, elle, est inconnue.
- *S* est un signal interne du système pour laquelle nous n'avons aucun accès. *S* est défini par *S=F(A,B)*. La fonctionnalité *F* étant connue, nous pouvons la reproduire par un autre moyen. Désignons par *s=f(A,B)* un calcul effectué à l'extérieur du circuit, par exemple à l'aide d'un ordinateur et appelons cette fonction *f*, fonction de sélection.
- Enfin, nous supposons que nous pouvons faire fonctionner ce circuit autant de fois que nécessaire avec les stimuli que nous choisissons et que nous pouvons enregistrer les courbes de consommation instantanée notées *I(t)*.

L'analyse se déroule en deux phases. Dans un premier temps a lieu la collecte des échantillons. Elle consiste à réaliser *N* opérations *F* avec une valeur *A* inconnue mais fixe et une valeur *B* différente à chaque fois. Pour chaque calcul, nous enregistrons la trace $I_N(t)$ et nous lui associons la valeur de B_N utilisée afin de remplir la matrice *M* contenant les valeurs des couples $[I_N(t), B_N]$. Il convient de choisir N suffisamment grand.

$$M = \begin{bmatrix} B_0 & I_0(t) \\ \vdots & \vdots \\ B_{N-1} & I_{N-1}(t) \end{bmatrix} \qquad \textbf{(Equ. 12)}$$

Une fois la collecte des échantillons accomplie, nous pouvons procéder à l'analyse proprement dite. Pour cela, définissons deux ensembles vides notés E_0 et E_1 dans lesquels nous regrouperons les courbes de consommation en fonction de la valeur s_N calculée grâce à notre fonction de sélection *f*. Rappelons que $s_N=f(A,B_N)$. E_0 contiendra les courbes pour lesquelles s_N est égal à 0 et E_1 les courbes pour lesquelles s_N est égal à 1, tels que :

$$E_0 = \{I_N(t) / s_N = 0\} \quad \textbf{(Equ. 13)}$$

$$E_1 = \{I_N(t) / s_N = 1\} \quad \textbf{(Equ. 14)}$$

Faisons une hypothèse sur la valeur du signal *A,* notée A_K (par exemple A_K*="0000"*) et calculons pour chaque couple $[I_N(t), B_N]$ la valeur $s_N=f(A_K,B_N)$ correspondante afin de classer les courbes $I_N(t)$ dans les ensembles E_0 et E_1. Nommons E_{0K} et E_{1K} ces ensembles pour une hypothèse A_K donnée.

Définissons ensuite les courbes moyennes de chaque ensemble $M_{E0K}(t)$ et $M_{E1K}(t)$ par :

$$M_{E0K}(t) = \frac{\sum I_n(t) \in E_{0K}}{N_{E0K}} \quad \textbf{(Equ. 15)}$$

$$M_{E1K}(t) = \frac{\sum I_n(t) \in E1K}{N_{E1K}} \quad \textbf{(Equ. 16)}$$

où N_{EiK} représente le nombre d'élément de l'ensemble E_{iK}.

Définissons enfin $\Delta_K(t)$ comme la courbe qui représente la différence de ces deux courbes moyennes pour une hypothèse A_K donnée :

$$\Delta_K(t) = M_{E0K}(t) - M_{E1K}(t) \quad \textbf{(Equ. 17)}$$

Si l'hypothèse A_K est correcte, cela signifie que les valeurs s_N calculées correspondent réellement aux valeurs S_N que le circuit a propagé. Ainsi, le classement en deux ensembles n'est pas quelconque. L'ensemble E_{0K} regroupera toutes les courbes pour lesquelles les transitions internes ont amené à fixer le signal *S* à un 0 logique. Idem, l'ensemble E_{1K} regroupera toutes les courbes pour lesquelles les transitions internes ont fixé le signal *S* à la valeur logique 1. Si nous visualisons les courbes $I_N(t)$ placées dans chaque ensemble, cela donne les chronogrammes de la partie gauche de la Figure III.16. Ainsi, dans chaque ensemble, la signature de la logique sera similaire au moment de l'activation du signal *S*, à l'instant t_S. Dans la Figure III.16, sont aussi représentées en pointillé les courbes $M_{E0K}(t)$ et $M_{E1K}(t)$. Toujours dans le cas où l'hypothèse faite sur A_K est correcte, la trace de $\Delta_K(t)$ laissera apparaître un pic qui correspond à l'instant t_S où les portes logiques activent le signal *S*.

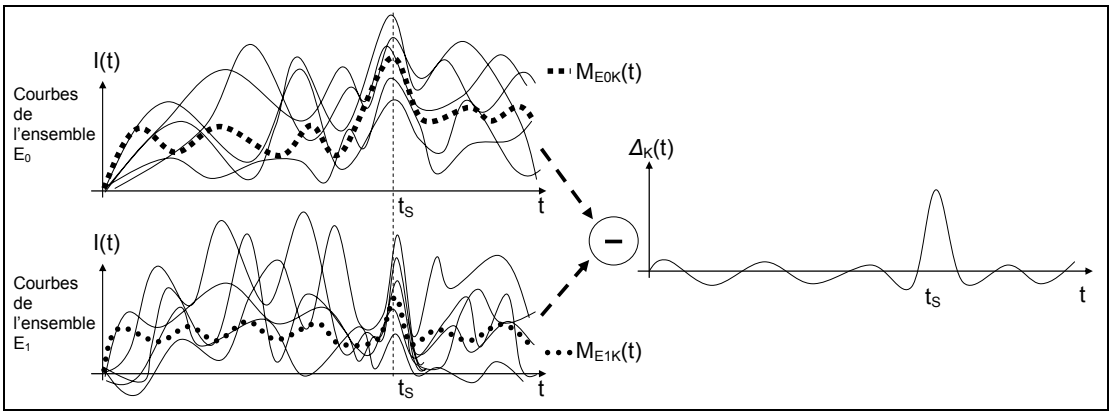

Figure III.16 : Représentation graphique des courbes.

En revanche, si l'hypothèse A_K est fausse, cela signifie que les valeurs S_N et s_N sont différentes. Par conséquent le classement en deux ensembles ne correspond pas avec ce qu'il s'est réellement passé dans le circuit lorsque ce dernier a fixé la valeur du signal *S*. Ainsi, les courbes seront triées de façon aléatoire dans les deux ensembles. Autrement dit, nous ne retrouverons pas dans un ensemble seulement des courbes ayant les mêmes caractéristiques. Lorsque nous calculerons les moyennes $M_{E0K}(t)$ et $M_{E1K}(t)$ de chaque ensemble, nous obtiendrons des courbes avec peu d'amplitude car la moyenne de courbes aléatoires donne une courbe assimilable à du bruit. Le calcul de la courbe différentielle $\Delta_K(t)$ produira donc une autre courbe proche du bruit.

Pour une analyse complète, il est préférable d'effectuer une exploration exhaustive de toutes les valeurs possibles de *A*. Cela donne dans notre cas : $0 \le k \le 15$ car A est codée sur 4 bits. La méthode la plus simple pour tester l'hypothèse de départ est une méthode empirique. Elle consiste à superposer sur un graphique toutes les courbes $\Delta_K(t)$ (voir Figure III.17). La courbe qui présentera le pic le plus important correspondra à l'hypothèse qui a la plus forte probabilité d'être juste.

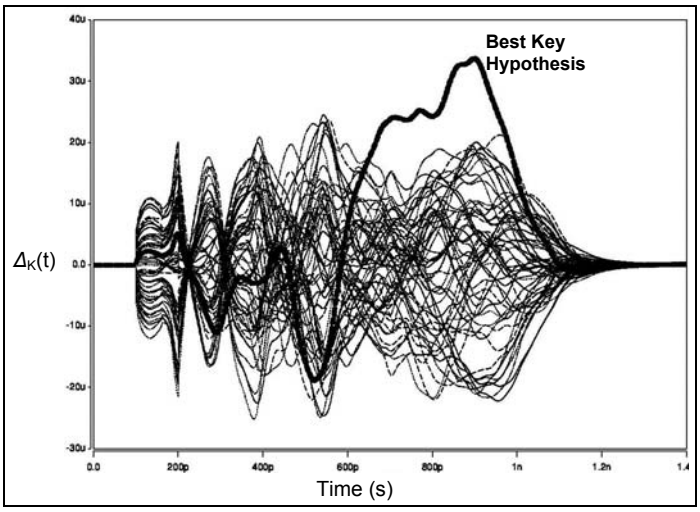

Figure III.17 : Vérification graphique de l'hypothèse.

Malheureusement, il est fréquent que les courbes différentielles ne se démarquent pas bien les unes des autres. Ceci peut être dû à plusieurs choses comme par exemple à de mauvaises mesures ou encore à un circuit basé sur une technologie CMOS à faible consommation de courant.

Notons que l'analyse *DPA* est fortement dépendante de l'implantation. Potentiellement, une modification dans l'implantation donnera des résultats complètement différents. Par exemple, une même fonctionnalité cryptographique (RTL identique) peut donner une résistance à la *DPA* complètement différente en fonction des aspects suivants :

- Modification des paramètres de la synthèse logique (optimisation en surface, en consommation ou en temps d'exécution),
- Placement et routage,
- Changement de technologie CMOS d'intégration,
- Intégration d'éléments perturbateurs (contre-mesures) …

III.2.2. Coefficient de corrélation

Il existe une autre méthode, statistique cette fois, pour tester les hypothèses faites sur la valeur de A, beaucoup plus performante que la méthode empirique.

Elle consiste à calculer le coefficient de corrélation, noté ρ, entre deux séries X et Y de n éléments x_i et y_i. Ce coefficient de corrélation, aussi appelé coefficient de Bravais-Pearson du nom de ses inventeurs, permet de quantifier la dépendance linéaire entre les deux séries. Il est défini par l'équation suivante :

$$\rho = \frac{\mathrm{cov}(X,Y)}{\sqrt{V(X).V(Y)}} \qquad \textbf{(Equ. 18)}$$

où $\mathrm{cov}(X,Y) = \frac{1}{n}\sum x_i.y_i - \bar{x}.\bar{y}$ représente la covariance des séries *X* et *Y*, *V(X)* et *V(Y)* respectivement la variance des séries *X* et *Y*, $\bar{x}$ et $\bar{y}$ respectivement la moyenne des séries *X* et *Y*.

La valeur de ρ est normée entre -1 et 1 (ou -100% et 100%). Plus la valeur absolue se rapprochera de la valeur 1, plus la dépendance linéaire entre les deux séries sera élevée.

Dans notre cas, les séries *X* et *Y* sont respectivement constituées par les valeurs de I_N et s_N pour un instant t et une hypothèse de clé donnés. Il convient de couvrir de façon exhaustive toutes les hypothèses de clé possibles pour tous les instants donnés afin d'obtenir une matrice de coefficients de corrélation à trois dimensions, nommée *ρ(K,t)*. La représentation graphique de cette matrice (voir Figure III.18) fait apparaître que la dépendance linéaire est maximale pour *K=35* et pour t compris entre 100 ns et 400 ns. En effet, c'est entre ces bornes que la valeur absolue de *ρ(K,t)* est la plus importante.

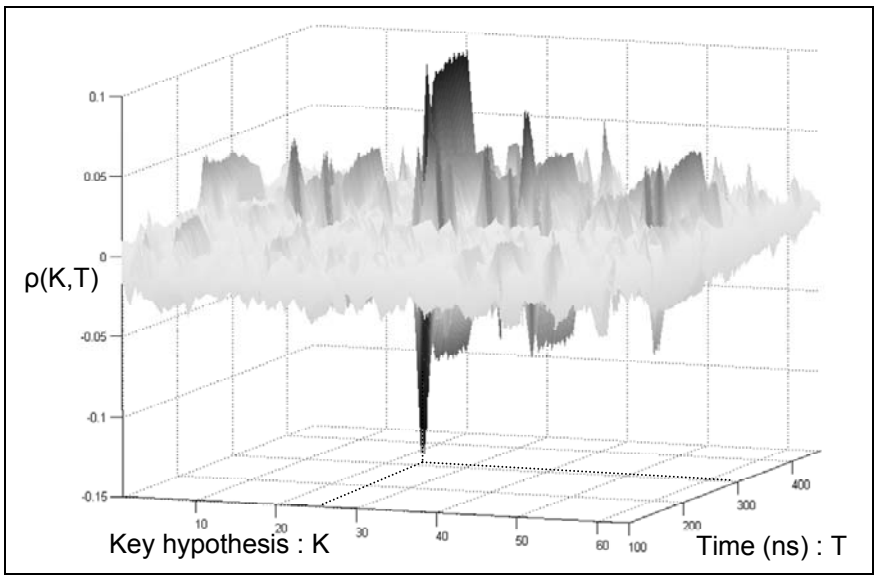

Figure III.18 : Représentation spatiale de la matrice des coefficients de corrélation.

Cependant, cette représentation spatiale n'est efficace que lorsqu'une clé se distingue bien des autres. Par la suite, nous utiliserons peu cette représentation et lui préfèrerons la suivante.

La Figure III.19 donne une représentation sur deux dimensions des coefficients calculés et correspond à une évolution temporelle des différentes hypothèses de clé, soit $\rho_K(t)$.

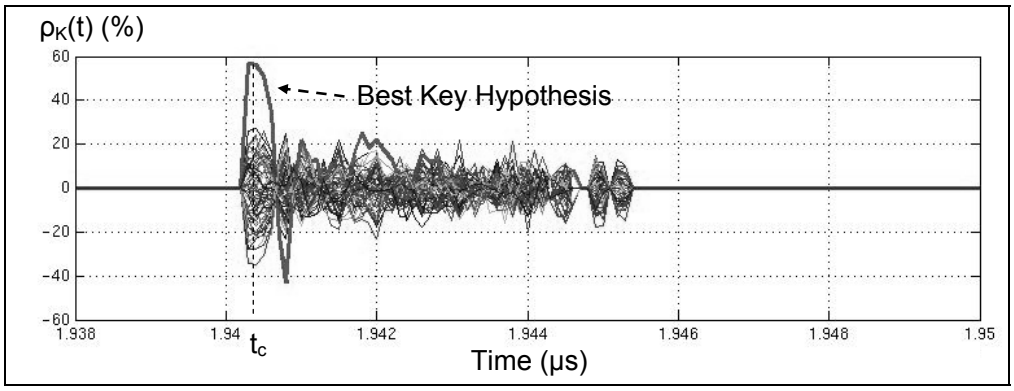

Figure III.19 : Représentation temporelle de tous les $\rho_K(t)$.

Dans cette représentation, tous les $\rho_K(t)$ correspondants à toutes les hypothèses de clés sont superposés. Comme expliqué précédemment, il convient d'interpréter seulement les valeurs absolues de ces coefficients de corrélation. Nous considérons que si la plus grande des valeurs absolues est supérieure d'au moins 10% à la seconde valeur absolue alors l'hypothèse présente une forte présomption d'être correcte ; charge à l'évaluateur de vérifier sur le circuit réel si la clé est exacte. Dans notre représentation, cette condition est remplie à l'instant t_c.

III.2.3. Evolution des coefficients de corrélation :

Dans la partie précédente, toutes les courbes correspondent à une analyse *DPA* pour un nombre d'échantillons donné, par exemple *N=250*. De la même façon que les coefficients de corrélation sont fortement dépendants de l'implantation, les vecteurs qui activent le circuit ont une influence. Il est possible qu'une analyse pour *N=250* révèle une clé différente d'une analyse pour *N=500*.

Nous allons étudier l'évolution du maximum de la valeur absolue du coefficient de corrélation en fonction de ce paramètre *N*, nommé P_K et défini par l'équation suivante :

$$P_K = Max(|\rho_K(t)|) \qquad \textbf{(Equ. 19)}$$

Par exemple, dans la Figure III.19, P_K est égal à 0,6. En calculant P_K pour chaque *N*, nous sommes capables de réaliser l'étude de P_K en fonction de l'élévation du nombre d'échantillons. La Figure III.20 représente cette évolution et permet de visualiser $P_K(N)$.

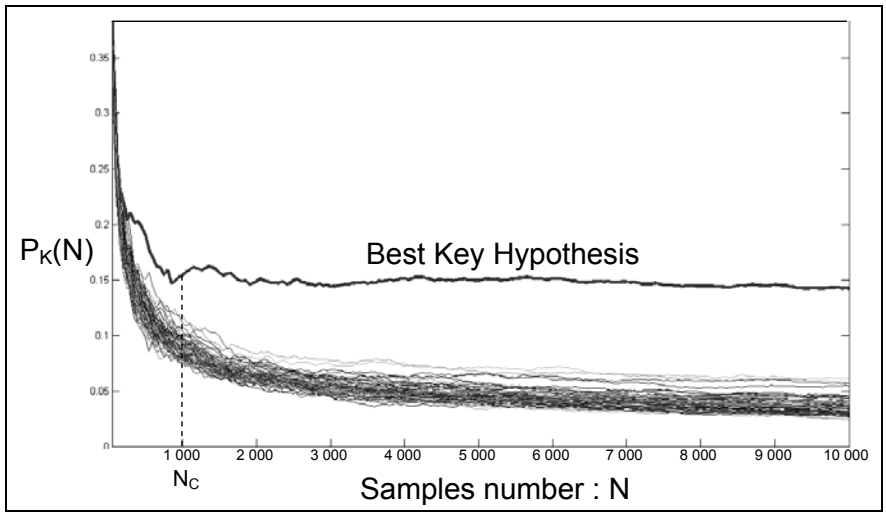

Figure III.20 : Représentation graphique de $P_K(N)$ en fonction du nombre d'échantillons.

Ce dernier graphique nous permet de distinguer deux zones :

- la zone pour laquelle *N<1 000* : Dans cette zone, les maximums des valeurs absolues des coefficients de corrélation sont élevés et différentes hypothèses semblent vérifiées successivement. De plus, toutes les valeurs ont tendance à décroître fortement avec l'augmentation du nombre d'échantillons. Cela signifie que dans cette zone, les valeurs sont fortement dépendantes des vecteurs ayant activé le circuit. Une séquence peut révéler une hypothèse alors qu'une autre séquence sensiblement différente peut en révéler une autre.
- la zone pour laquelle *N>1 000* : Dans cette région, les valeurs en ordonnée sont plus faibles et restent relativement inchangées avec l'augmentation de la collecte d'échantillons. Dans cette région-là, une seule hypothèse de clé est révélée et la valeur correspondante du coefficient de corrélation reste à peu près constante.

Cette représentation fait disparaître l'occurrence temporelle des évènements. Ainsi, lors d'une analyse *DPA*, il est intéressant d'étudier l'évolution de $P_K(N)$ afin de déterminer le nombre d'échantillons nécessaires pour casser un système. Dans un second temps, il convient d'étudier $\rho_K(t)$ afin de déterminer à quel moment intervient l'évènement qui révèle la clé.

Dans l'exemple que nous venons d'étudier, *N=1 000* représente le nombre minimal d'échantillons aléatoires nécessaires à un attaquant pour trouver la bonne hypothèse de clé. Nommons cette valeur limite N_C pour définir ce seuil de certitude. Par la suite, nous utiliserons ce paramètre afin d'évaluer la sécurité d'un système. En effet, l'élévation de cette valeur a pour conséquence directe d'augmenter considérablement le temps, les ressources matérielles ainsi que le coût de l'attaque *DPA*.

III.2.4. Analyse DPA contre l'algorithme DES

La partie précédente présentait les concepts d'une analyse différentielle du courant consommé par un circuit. Ces concepts peuvent s'appliquer à tous les algorithmes de chiffrement à partir du moment où leurs fonctionnalités sont connues. C'est notamment le cas de l'algorithme DES dont la structure est publique. Pour plus de détails sur cet algorithme, reportez vous à la partie I.3.2.1. Nous allons maintenant décrire comment évaluer la résistance d'une implantation de l'algorithme DES, suivant la méthode présentée dans [KOC99]

La première étape à réaliser lors d'une analyse *DPA* est de définir un signal cible et la fonction de sélection associée. Il est essentiel que le signal ciblé soit l'aboutissement d'un cône logique dont un des points de départ est constitué par des bits de la clé recherchée et que ce cône logique soit composé de suffisamment d'étages logiques entre les bits de clé et le bit ciblé ; ceci afin de maximiser l'effet de la différence de la consommation des portes logiques en fonction des bits de clé. Il est possible de cibler un signal situé à l'extrémité d'un cône logique de faible profondeur mais dans ce cas là, le nombre d'échantillons nécessaires pour évaluer l'implantation de l'algorithme sera plus important.

De plus, une bonne fonction de sélection doit présenter certaines propriétés statistiques. En particuliers, elle doit distribuer les échantillons de façon aléatoire si l'hypothèse de classement est fausse. Enfin, elle dépend des données auxquelles l'attaquant a accès ; admettons que le *Plaint Text* soit accessible.

Ainsi, pour évaluer la résistance d'un circuit face à la *DPA*, il est intéressant de choisir comme bit de sélection une des sorties des « ou exclusif » placés en sortie des SBOX. En effet, ces bits représentent la fin du cône logique ayant pour point de départ le « ou exclusif » placé en entrée des SBOX. Une des deux entrées du « ou exclusif » d'entrée de SBOX est constituée par les bits de la clé secrète. Dans la plupart des implantations, les SBOX sont constituées de plusieurs étages logiques.

Il est courant de définir comme bit cible le bit de sortie du « ou exclusif » de la ronde 0 correspondant au premier bit de sortie de la *SBOX1* (voir Figure III.21). Ce choix est arbitraire ; en aucun cas ce bit n'est plus sensible que les autres. Cela réside dans le fait que l'implantation de toutes les SBOX d'un même circuit est similaire. Par conséquent, si l'analyse est capable de révéler la sous-clé correspondante à la SBOX1 de la première ronde, elle révèlera de la même façon les autres sous-clés du système avec la même difficulté. Il est même préférable de cibler successivement tous les bits de sortie de la SBOX choisie afin d'affiner les résultats. Appelons δ_1 cette fonction de sélection et S_1 le signal ciblé.

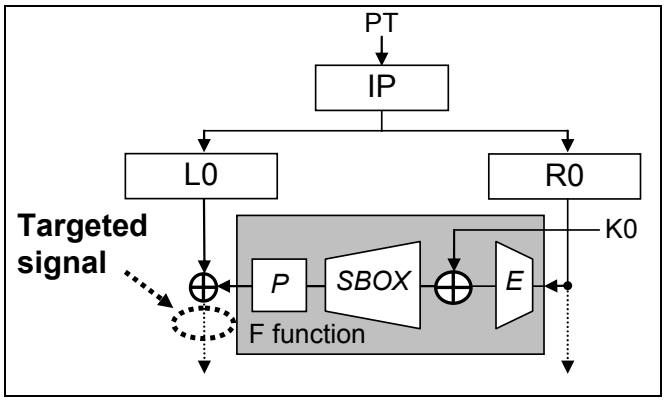

Figure III.21 : Position du bit ciblé par la fonction de sélection S_1.

Dans ce cas là, S_1 est définie par l'équation suivante :

$$S_1 = F(K_0, R_{0,N}) \oplus L_{0,N} = \delta_1(PT_N, K_0) \quad \textbf{(Equ. 20)}$$

avec $$L_{0,N} = L[IP(PT_N)] \quad \textbf{(Equ. 21)}$$

et $$R_{0,N} = R[IP(PT_N)] \quad \textbf{(Equ. 22)}$$

Dans ce système d'équations, la seule valeur inconnue pour calculer le bit cible est la valeur de la sous-clé K_0. C'est donc sur cette variable que nous ferons une hypothèse. De plus étant donné que nous ne ciblons qu'un seul bit de sortie correspondant à la *SBOX1*, seule une partie de la sous-clé K_0 nous intéresse. En effet, l'algorithme DES est constitué de telle sorte que les bits de sortie d'une SBOX sont seulement fonctions de 6 bits de la sous clés K_0 et de 6 bits du *Plain Text*. Désignons les 6 bits de sous-clé de la SBOX1 de la première ronde par $K_{01,h}$ où h représente la valeur de l'hypothèse. Ainsi, $K_{01,h}$ est définie par :

$$K_{01,h} = h \ \textbf{avec} \ 0 \le h \le 2^6 \textbf{, soit} \ 0 \le K_{01,h} \le 63 \quad \textbf{(Equ. 23)}$$

Ce qui représente 64 hypothèses à tester.

Pour chaque message PT_N, les ensembles E_{0h} et E_{1h} regroupent les courbes de consommation $I_N(t)$ définis par :

$$E_{0h} = \{I_N(t) / \delta_1(PT_N, K_{01,h}) = 0\} \quad \textbf{(Equ. 24)}$$

$$E_{1h} = \{I_N(t) / \delta_1(PT_N, K_{01,h}) = 1\} \quad \textbf{(Equ. 25)}$$

Enfin vient l'étape de test des hypothèses h, soit en calculant tous les Δ_{1h} ; soit en calculant la matrice des coefficients de corrélation $\rho(K,t)$.

Cette fonction de sélection S_1 cible seulement 6 bits de la sous-clé K_0 de la première ronde. En reproduisant cette attaque aux autres parties de la sous-clé de la première ronde, nous pouvons reconstruire l'intégralité de la sous-clé K_0. Une fois K_0 reconstruite, nous pouvons utiliser l'inverse de la fonction de génération de clés pour retrouver de nombreux bits de la clé K, la clé de chiffrement DES. En effet, les sous-clés de rondes sont dérivées de la clé de chiffrement K (voir § I.3.2.1). Une fois K_0 connue, nous pouvons reproduire l'attaque sur la deuxième ronde avec les données L_1

et R_1 qui sont maintenant connues. Ainsi de suite, nous renouvelons l'attaque jusqu'à retrouver les 56 bits de la clé DES.

- **Amélioration :**

Si nous avons un accès total au *Plain Text*, nous pouvons adapter les messages clairs afin de limiter les transitions dans le circuit. Nous pouvons, par exemple, figer les bits qui n'atteignent pas la *SBOX1* à des valeurs fixes afin que les transitions dans la partie du circuit qui ne nous concerne pas soient toujours les mêmes pour chaque calcul. L'objectif est de diminuer le bruit généré par les autres portes logiques du circuit dans le but de diminuer au maximum le nombre d'échantillons requis pour révéler la bonne clé. En fait, nous faisons varier seulement les bits dits utiles (ceux qui atteignent la *SBOX1*) pour chaque calcul.

III.2.4.1. Attaque avec texte chiffré connu

Il est possible que le système attaqué n'autorise pas l'accès au texte clair. Dans ce cas là, nous pouvons mener l'attaque à partir du texte chiffré, *CT*, selon le même principe. Il convient de modifier la fonction de sélection afin de cibler cette fois le bit d'entrée du « ou exclusif » correspondant au premier bit de sortie de la première SBOX de la dernière ronde, comme représenté dans la Figure III.22.

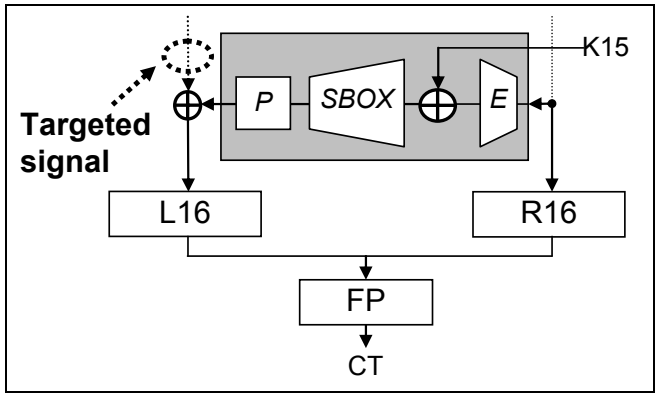

Figure III.22 : Position du bit ciblé par la fonction de sélection S_2.

Dans ce cas là, S_2 est définie par l'équation suivante :

$$S_2 = F(K_{15}, R_{16,N}) \oplus L_{16,N} = \delta_2(CT_N, K_{15}) \qquad \textbf{(Equ. 26)}$$

avec $$L_{16,N} = L\left[FP^{-1}(CT_N)\right] = L\left[IP(CT_N)\right] \qquad \textbf{(Equ. 27)}$$

et $$R_{16,N} = R\left[IP(CT_N)\right] \qquad \textbf{(Equ. 28)}$$

car $CT = FP(L_{16}R_{16}) \Leftrightarrow L_{16}R_{16} = FP^{-1}(CT) = IP(CT)$. Rappelons que la fonction *FP* est l'inverse de la fonction *IP*.

III.2.4.2. Fonction de sélection ciblant plusieurs bits

Une évolution possible consiste à prendre une fonction de sélection qui ne cible pas un seul bit de sortie d'une *SBOX* mais les 4 bits de sortie d'une *SBOX*. Dans ce cas-là, la fonction de sélection est la même que S_1 (ou S_2) mais en ciblant 4 bits au lieu d'un seul. Cette fonction de sélection peut s'appliquer aussi bien sur la première ronde que sur la dernière ronde en fonction des données auxquelles nous avons accès. Elle permet de considérablement réduire le bruit mais nécessite un nombre d'échantillons bien plus grand. Les ensembles E_0 et E_1 sont donc définis de la façon suivante :

$$E_0 = \left\{ I_N(t) / \delta_{3,N} = "0000" \right\} \quad \textbf{(Equ. 29)}$$

$$E_1 = \left\{ I_N(t) / \delta_{3,N} = "1111" \right\} \quad \textbf{(Equ. 30)}$$

Cette méthode est bien décrite dans [MES99].

III.2.4.3. Fonction de sélection basée sur le poids de Hamming

Cette fonction de sélection cible non plus la valeur d'un ou plusieurs bits mais le nombre de bits à la valeur 1 en sortie de la *SBOX*, autrement dit le poids de Hamming d'un bus. De la même façon que précédemment, il est possible de cibler différents signaux en fonction de données auxquelles nous avons accès. Dans [CLA00], les auteurs donnent plus de détails sur cette méthode. La différence consiste à classer les courbes de consommation en deux ensembles de la façon suivante :

$$E_0 = \left\{ I_N(t) / \delta_{4,N} \in \{0000,0001,0010,0100,1000\} \right\} \quad \textbf{(Equ. 31)}$$

$$E_1 = \left\{ I_N(t) / \delta_{4,N} \in \{1111,1110,1101,1011,0111\} \right\} \quad \textbf{(Equ. 32)}$$

III.2.5. La DPA dans la pratique

Dans cette partie nous allons présenter comment évaluer concrètement la sécurité d'un système dans la réalité.

- **Acquisition des données**

Dans la pratique, la première phase consiste à acquérir les données. L'objectif est de disposer d'un nombre suffisamment important de couples : courbes de consommation instantanée - messages (clairs ou chiffrés).

Cela consiste donc à réaliser des calculs cryptographiques sur le circuit ciblé et d'enregistrer les courbes de consommation instantanée pour chaque calcul. Nous considérons qu'une personne qui veut attaquer un circuit cryptographique a accès au texte clair ou au texte chiffré.

Pour ce faire, un attaquant a donc besoin du dit circuit, d'un moyen de communication avec ce circuit (par exemple un lecteur de carte à puce), d'une sonde de tension ou de courant (plus précise mais plus onéreuse), d'un amplificateur, d'un oscilloscope et d'un moyen de stockage de données (un disque intégré à l'oscilloscope ou un ordinateur équipé d'une carte d'acquisition).

Ainsi le coût de l'acquisition à moins de 20 0000 Euros pour le matériel suivant : oscilloscope, sonde, amplificateur de signal, carte d'acquisition et stockage des données. Le coût peut être plus élevé si l'attaquant veut faire des mesures plus précises et s'il veut pouvoir traiter un plus grand nombre d'échantillons.

- **Découpage des échantillons :**

Il convient de repérer le calcul cryptographique afin de limiter l'acquisition aux parties utiles ; ceci afin de diminuer les ressources nécessaires au traitement des données. L'analyse SPA, relativement simple à mettre en place, peut être utilisée pour mettre en évidence les rondes et repérer la fenêtre temporelle à analyser.

Dans l'analyse SPA de la Figure III.23 extraite de [GEM01], les motifs se discernent distinctement et permettent d'identifier qu'il s'agit très probablement d'un chiffrement DES, confirmé par le fait que nous retrouvons des pics qui coïncident avec les décalages de la clé de chiffrement (LS). Seules l'acquisition et l'analyse des premières rondes suffisent pour retrouver la clé secrète.

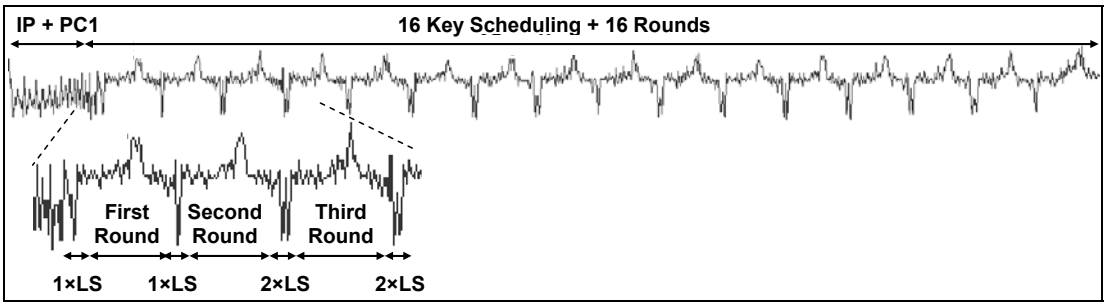

Figure III.23 : Repérage des rondes du DES.

- **Nombre d'échantillons :**

Dans la pratique, les mesures du courant *I(t)* sont perturbées par différentes sources de bruit :

- Le bruit induit lors de la mesure : la sonde de mesure, l'environnement …,
- Le bruit induit par l'activité interne du circuit. L'activité des portes que nous ne considérons pas s'apparente à du bruit,
- Le bruit généré par la technologie CMOS.

Le rapport du signal par rapport au bruit (ou *SNR* pour *Signal to Noise Ratio*) est proportionnel à la racine carrée du nombre d'échantillons traités [MES02] :

$$SNR = \frac{\sqrt{N}}{\sigma} \qquad \textbf{(Equ. 33)}$$

où σ représente le bruit.

Ainsi, avec l'élévation du nombre d'échantillons, la partie utile du signal se détache du bruit, améliorant l'analyse. [CLA00] décrit en détail comment déterminer le nombre d'échantillons nécessaires pour révéler la clé de chiffrement en fonction du bruit.

- **Traitement des données**

La puissance de la *DPA* est qu'il n'est pas nécessaire de connaître l'implantation exacte du circuit. Tout au plus, la connaissance de l'algorithme utilisé permet un gain de temps dans le repérage des opérations cryptographiques. Mais une analyse SPA permet d'apprendre énormément de choses sur le déroulement des opérations internes.

Le coût de l'acquisition des données : Station de travail (PC) et carte d'acquisition est compris entre 5 KEuros et 10 KEuros. Tout dépend de la rapidité, de la précision et du nombre de courbes que l'attaquant souhaite traiter.

Le coût total de l'équipement pour acquérir les données et les traiter est compris entre 10 Keuros et 20 KEuros. Avec cet équipement, il est même possible de venir à bout de circuits pour lesquels les concepteurs ont mis en place des contre-mesures. En effet, la *DPA* est capable d'exploiter la moindre faille de conception. Pour les implantations robustes, ce chiffre peut largement doubler.

En ce qui concerne le temps total de traitement (acquisition et analyse), il est estimé à quelques jours pour une implantation peu robuste avec des moyens matériels puissants, à plusieurs semaines pour une implantation plus robuste avec du matériel moins performant.

- **Algorithme d'analyse DPA**

Dans la pratique, une analyse *DPA* peut se résumer par l'algorithme présenté dans la Figure III.24.

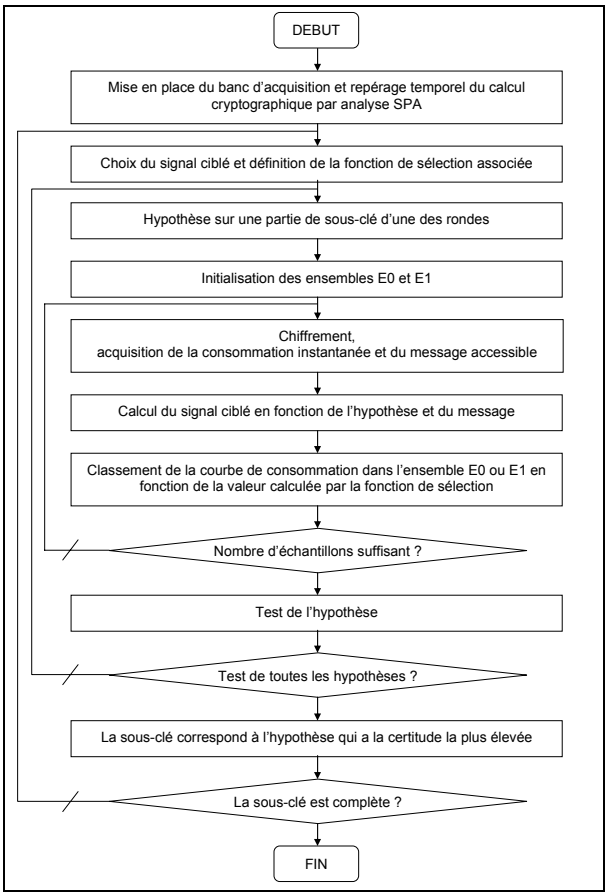

Figure III.24 : Algorithme de l'attaque DPA sur un DES

III.3. Contre-mesures DPA

Au sens large, une contre-mesure est un artifice qui renforce un système face à une attaque dans le but de la déjouer, sans modifier ou altérer le fonctionnement global de ce système. Dans le cas de la *DPA*, nous définissons par contre-mesure *DPA* une méthode qui permet de limiter la corrélation entre la consommation d'un circuit intégré et les données que celui-ci traite afin de compliquer l'attaque, idéalement de la rendre impossible.

Ces techniques peuvent être des modifications d'algorithmes afin de les rendre plus robustes, des ajouts de modules internes, l'encapsulation de parties sensibles, le durcissement de cellules afin de limiter les fuites d'informations …

Nous allons dans cette partie présenter les différents types de contre-mesures existantes pour déjouer la *DPA* selon les niveaux d'abstraction présentés dans la Figure III.25.

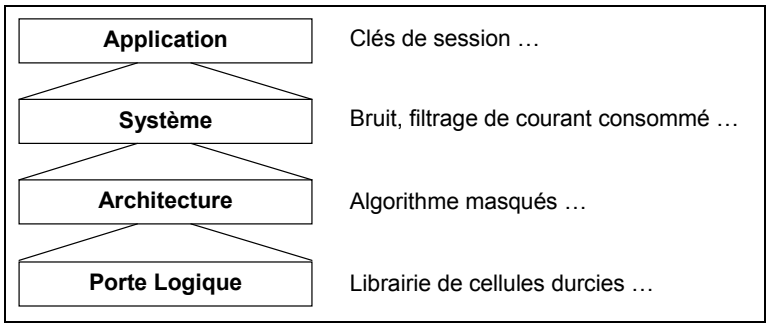

Figure III.25 : Classification des contre-mesures.

III.3.1. Application

La première sécurité à apporter dans un circuit se situe au niveau applicatif. En effet, un attaquant ayant besoin d'un nombre assez élevé d'échantillons, il est souhaitable de changer régulièrement les clés de chiffrement afin de le perturber dans ses acquisitions. C'est le principe des clés de session développé dans [SCH97].

III.3.2. Système

Il est possible de renforcer un système déjà existant en lui ajoutant des modules supplémentaires qui ne perturbent en rien la fonctionnalité globale du système. Voici une liste non exhaustive des principales contre-mesures existantes.

- **Générateur de bruit :**

Une méthode couramment utilisée consiste à ajouter un bloc dont la seule fonctionnalité est de générer un bruit sur l'alimentation. Pour ce faire, ce bloc consomme le courant de façon aléatoire [DAE99]. Il existe plusieurs implantations. Il est par exemple possible de dupliquer certains blocs du circuit et de les faire fonctionner avec des données aléatoires.

Notons que même en ajoutant du bruit sur l'alimentation, une analyse *DPA* reste théoriquement faisable. La conséquence directe est d'augmenter le nombre d'échantillons nécessaires pour effectuer une analyse. Dans la pratique, si cette augmentation est suffisamment élevée, elle peut s'avérer très perturbante pour l'attaquant. En effet, les ressources nécessaires pour stocker les échantillons et pour les traiter peuvent rapidement devenir problématiques et demander un investissement qui n'est pas à la portée de tout attaquant.

- **Horloge dynamique :**

Afin de désynchroniser l'occurrence des événements ciblés par une analyse *DPA*, il est possible de modifier dynamiquement et de façon aléatoire la durée des cycles d'horloge. Bien entendu, la fréquence d'horloge est comprise dans une fenêtre limitée. Cette technique est présentée dans [DAE99]. De la même façon, cette contre-mesure ne fait que ralentir le travail des attaquants.

- **Filtrage actif de l'alimentation**

[RAK00] présente une méthode de filtrage de l'alimentation afin de limiter la fuite d'information. Son efficacité est suffisante mais malheureusement elle engendre une surconsommation de courant ainsi qu'une élévation de la surface. De plus, une attaque invasive déjoue facilement cette contre-mesure.

- **Alimentation détachée :**

Dans [SHA00], l'auteur présente une méthode qui consiste à utiliser une alimentation détachée entre le système et les bornes d'alimentation permettant de lisser la consommation en courant. Cette solution laisse tout de même fuir des informations exploitables par un attaquant. De plus, elle élève considérablement la consommation en courant ainsi que la surface du circuit.

III.3.3. Architecture

Il existe aussi des contre-mesures liées aux architectures. Elles consistent à modifier les parties sensibles d'un système.

- **Fausses Instructions**

Dans les implantations logicielles typiques, les instructions correspondantes à un algorithme de chiffrement donné sont exécutées de façon séquentielle. Par exemple, les évènements liés à la valeur de la clé secrète interviennent toujours au même moment. Une méthode simple pour désynchroniser l'occurrence des événements ciblés est d'insérer de « fausses » instructions. Ainsi, il est beaucoup plus difficile pour un attaquant de réaliser la courbe de consommation moyenne et ainsi d'en extraire des données utiles. L'attaquant obtiendra des moyennes incohérentes [LYO97].

De plus, l'ajout de fausses instructions représente une contre-mesure contre la SPA. En effet, la SPA se base essentiellement sur la synchronisation des évènements.

- **Exécution aléatoire des opérations**

Une autre technique pour désynchroniser l'occurrence des évènements ciblés par la *DPA* consiste à dérouler les instructions non plus de façon séquentielle mais de façon aléatoire, autant que possible. Ceci est plus détaillé dans [CLA00] qui introduit les *RPI* (pour *Random Process Interrupts* ou Interruption Aléatoire de Processus).

- **Algorithmes masqués :**

Il existe une technique qui consiste à masquer les données internes traitées par les algorithmes de chiffrement. Plus exactement, elle consiste à faire en sorte que les informations qui peuvent fuir par canaux cachés semblent aléatoires pour un attaquant.

Dans [GOU99], les auteurs présentent une technique de duplication qui consiste à rendre aléatoire le traitement des données dans un algorithme cryptographique.

Dans [BAJ04], est détaillée l'arithmétique *LRA* (pour *Leak Resistant Arithmetic*). Cette méthode basée sur l'utilisation des *RNS* (pour *Residue Number Systems*) permet d'effectuer les calculs cryptographiques en diminuant les informations qui fuient par les canaux cachés. Cette proposition offre un bon compromis entre exécution aléatoire des opérations et coût d'implantation.

M.-L. Akkar et C. Giraud, quand à eux, ont présenté une technique de masquage des données internes, appliquée dans [AKK01] aux algorithmes de chiffrement *DES* et *AES*. L'idée est de masquer les données à chiffrer (respectivement déchiffrer) avec un masque de 64 bits en début de chiffrement (respectivement déchiffrement), noté *X*, et de dérouler l'algorithme normalement. Il n'est pas nécessaire de modifier les fonctions linéaires car :

$$P(R_1 \oplus X_1) = P(R_1) \oplus P(X_1) \qquad \textbf{(Equ. 34)}$$

En revanche, il convient de modifier les tables *SBOX* qui sont des fonctions non linéaires par des *MSBOX* (pour *Masked SBOX*) définies de la façon suivante :

$$MSBOX(R_1) = SBOX(R_1 \oplus X2) \oplus P^{-1}(X1_L \oplus X1_R) \qquad \textbf{(Equ. 35)}$$

Où $X1 = IP(X)$, $X2 = E(X1_R)$ et P^{-1} représente l'inverse de la fonction *P*.

Cet algorithme DES, dit masqué, nécessite plus d'opérations qu'un DES dit classique. En revanche, il présente un avantage conséquent face à une analyse *DPA* : les données manipulées par les portes logiques ne dépendent plus directement des données traitées, elles dépendent aussi du masque.

Les algorithmes masqués sont tout de mêmes attaquables par des analyses *DPA* plus évoluées comme la *DPA* de second ordre [MES00].

III.3.4. Porte logique

Nous avons présenté dans la première partie de ce chapitre que la consommation d'une porte logique est fortement corrélée avec les données qui activent celle-ci. Ainsi, le dernier niveau d'abstraction à considérer est le niveau des portes logiques. L'objectif est de concevoir des cellules qui ont une consommation indépendante des données afin de rendre le système résistant à la *DPA*. Nous parlons alors de bibliothèque (ou encore de librairie) de cellules renforcées contre la *DPA*. Deux approches sont envisageables :

- La première consiste à concevoir des portes logiques qui consomment un courant constant, quelles que soient les données traitées ; nous parlons alors de portes logiques à consommation cachée (ou *Hiding Logic* ou encore *Blinding Logic*).
- La deuxième approche est sensiblement différente. Elle consiste à concevoir des portes logiques qui ont une consommation indépendante des données traitées. C'est-à-dire qu'elle dépend d'une autre variable que la donnée traitée, par exemple d'une grandeur aléatoire. Dans ce cas là, nous parlons de portes logiques à consommation masquée (ou *Masking Logic*). Ce qui fait que pour deux calculs successifs avec les mêmes données traitées, nous obtiendrons des consommations différentes.

Ces deux méthodes peuvent sembler proches. Pourtant elles présentent des caractéristiques statistiques radicalement opposées. Pour les portes *Hiding Logic*, la variance de la consommation est faible mais la moyenne des consommations est élevée. En revanche, pour les portes en *Masking Logic*, la variance de la consommation est élevée mais la moyenne des consommations est faible. La figure suivante (voir Figure III.26) représente succinctement les profils de consommation des différentes approches.

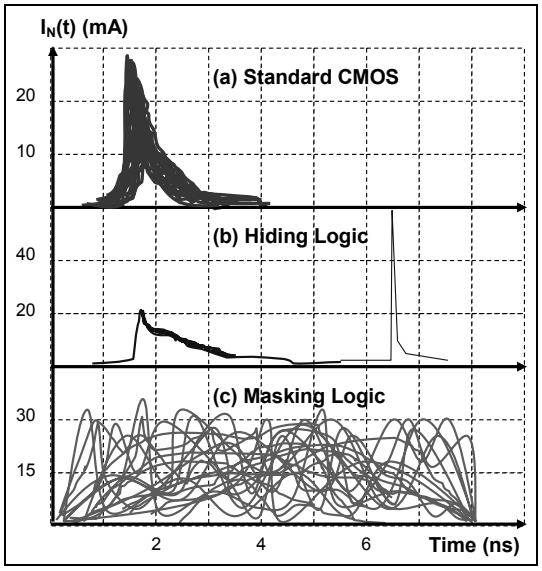

Figure III.26 : Profils de consommation.

Ainsi, les portes basées sur de la logique cachée présentent des caractéristiques statistiques intéressantes pour déjouer la *DPA*. Pourtant les portes en logique à consommation masquée présentent une robustesse face à la *DPA* qui n'est pas négligeable.

III.3.4.1. Logique cachée

De nombreuses techniques existent pour limiter la consommation des portes logiques dont la majorité reposent sur le même principe : le montage différentiel *DCVS* (pour *Differential Cascode Voltage Logic*) présenté par [CHU87].

La Figure III.27 représente le schéma dynamique d'une porte *DCVS*. Dans cette structure, les deux sorties sont préchargées à la valeur « 0 » lorsque le signal d'horloge est à « 0 ». Lorsque le signal d'horloge est à la valeur « 1 », les sorties complémentaires sont évaluées grâce à la fonctionnalité intégrée l'arbre *DCVS*.

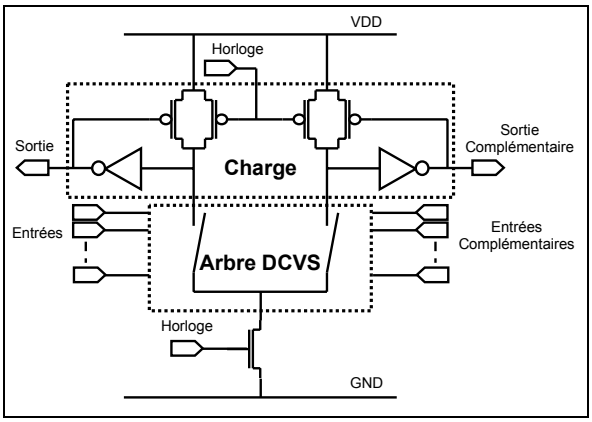

Figure III.27 : Schéma dynamique d'une porte DCVS.

Cette structure n'a cessé d'évoluer pour se décliner en de nombreuses variantes : Domino, DyCML, DDCVSL, LSCML, LVDCSL, CRDL, HRDL, SPDL, ADCVSL … Toutes cherchent à optimiser la consommation et pour certaines à améliorer la rapidité de fonctionnement.

Ces portes différentielles fonctionnent avec des données complémentaires (en entrée comme en sortie). Ces portes fonctionnent selon 2 phases :

- La précharge pendant laquelle les sorties sont toujours préchargées à la même valeur logique,
- L'évaluation durant laquelle les sorties complémentaires sont évaluées en fonction des entrées.

Ce montage est couramment appelé *DRP* (pour *Dual Rail Precharge*). SABL ([TIR02] et [TIR04]) ainsi que TDPL [BUC04] sont les montages les plus connus dérivés de DCVS et utilisés comme contre-mesures *DPA*.

Dans cette famille de portes basées sur le principe *hiding logic* nous retrouvons d'autres techniques qui n'utilisent pas le montage DCVS. Par exemple, la structure WDDL (pour *Wave Dynamic Differential Logic*) fait partie de cette famille [TIR05].

III.3.4.2. Logique masquée

L'approche logique masquée (ou *masking logic*) comme expliquée précédemment, tente de masquer la consommation des portes afin de rendre la consommation dépendante d'autres données que celles traitées.

MDPL (pour *Masked Dual Rail Precharge Logic*) est l'une des techniques les plus connues basée sur ce principe [POP05] et [POP06]. Pour ce faire, les portes traitent des données complémentaires avec de la logique complémentaire. De plus, les données sont masquées pour chaque porte avec des grandeurs aléatoires.

RSL (pour *Random Switching Logic*) présenté par [SUZ04] est équivalente à l'approche MDPL, avec un signal *enable* supplémentaire pour renforcer la robustesse.

La logique asynchrone [MOO02] fait partie de cette famille mais représente un cas particulier. Comme son nom l'indique, elle ne nécessite pas de signal de synchronisation pour traiter les données. En l'absence d'un tel signal, le traitement

repose sur un protocole bien particulier appelé requête-acquittement. Celui-ci est plus rapide que lorsque les opérations sont synchronisées car les portes traitent les données dès que celles-ci sont valides. Ce qui signifie que le temps de traitement des données est variable. D'un point de vue de la *DPA*, ce dernier point est important car il est très difficile pour un attaquant d'extraire des informations utiles de l'observation de la consommation. En contre partie, l'implantation de cette logique engendre une augmentation du nombre de portes et indirectement de la consommation du circuit. [FOU03], [RAZ06] et [BOU05] présentent l'utilisation de telles structures pour déjouer l'analyse *DPA*. Dans [RAZ06], l'auteur introduit une logique STTL (pour *Secured Triple Track Logic*). Il démontre dans ce document que cette logique est sensiblement plus robuste que les autres logiques asynchrones.

III.3.4.3. Synthèse

Dans le Tableau III-1, nous présentons les avantages et inconvénients des différentes contre-mesures existantes au niveau porte logique. Tous les facteurs sont donnés par rapport à une implantation sans contre-mesure basée sur une librairie CMOS standard.

Logique	Avantages	Inconvénients
SABL [TIR04]	Résistance à la DPA	Surface : ×2, Consommation : ×10 Vitesse : /2 Routage équilibré nécessaire, Arbre d'horloge important Développement d'une librairie Développement spécifique
TDPL [BUC04]	Résistance à la DPA	Surface : ×2, Consommation : ×10 Vitesse : /2 Routage équilibré nécessaire, Arbre d'horloge important Développement d'une librairie Développement spécifique
WDDL [TIR05]	Résistance à la DPA Pas de développement de librairie	Surface : ×4, Consommation : ×4 Vitesse : /2 Routage équilibré nécessaire Développement spécifique
MDPL [POP06]	Résistance à la DPA	Surface : ×4, Consommation : ×17 Vitesse : /2 Routage équilibré nécessaire Développement d'une librairie Développement spécifique
Asynchro-nous [BOU05] et [RAZ06]	Vitesse : ×2 Résistance à la DPA	Surface : ×5, Consommation : ×3 Routage équilibré nécessaire Développement spécifique
RSL [SUZ04]	Résistance à la DPA	Surface : ×2, Consommation : ×1,5 Vitesse : /2 Routage équilibré nécessaire Développement d'une librairie Développement spécifique

Tableau III-1 : Comparaison des différentes contre-mesures au niveau porte.

Toutes présentent une résistance plus ou moins accrue face à la *DPA* mais l'inconvénient majeur de ces méthodes est l'augmentation de la surface et de la consommation, augmentation trop importante pour certaines solutions allant jusqu'à un facteur multiplicatif de 17. Ces deux paramètres sont critiques pour les concepteurs de circuits sécurisés. Un autre point important est que pour toutes ces solutions, une grande attention doit être portée sur le routage afin que la charge des portes soit équilibrée. Enfin, toutes les approches, exceptée la logique WDDL, nécessitent le développement d'une librairie de cellules spécifiques ainsi que sa caractérisation. Cette étape représente un investissement conséquent. Dans le chapitre suivant, nous proposons une contre-mesure ainsi qu'une méthodologie de placement qui permet d'augmenter la résistance des circuits sécurisés face à une analyse *DPA* sans rencontrer les inconvénients précités.

III.4. Synthèse

Dans ce chapitre, nous avons tout d'abord décrit les phénomènes de consommation électrique qui interviennent dans les technologies CMOS. Cette

technologie est la plus répandue dans l'industrie des semi-conducteurs. Nous avons particulièrement étudié la structure de base : la porte inverseuse. Partant de cette structure, nous avons élaboré un modèle réaliste permettant d'étudier précisément les courants électriques qui interviennent dans les technologies CMOS. Ce modèle nous a permis de mettre en évidence le lien étroit qui existe entre la consommation électrique d'une porte et les données que celle-ci traite. Enfin, en généralisant notre modèle à un réseau d'inverseurs, nous avons démontré que l'observation de la consommation d'un circuit intégré permet d'obtenir des informations pertinentes sur les données que ce dernier traite. Ce qui dans le cadre des circuits sécurisés représente une faille majeure.

Dans un second temps, nous avons décrit l'attaque *DPA*, une des attaques les plus pratiquées contre les circuits sécurisés, en présentant tout d'abord les concepts à une fonctionnalité logique simple. Ensuite, nous avons détaillé cette attaque sur un algorithme de chiffrement DES en présentant différentes variantes. Dans cette partie nous avons aussi défini un paramètre, noté N_C, qui nous permet de quantifier la robustesse d'une implantation car il est directement lié au nombre d'échantillons nécessaires pour retrouver la clé secrète utilisée dans un algorithme. Par la suite, nous utiliserons ce paramètre pour comparer la résistance des contre-mesures que nous proposerons.

Dans la dernière partie, nous avons décrit les contre-mesures existantes pour déjouer les analyses *DPA* selon différents niveaux d'abstraction. Il en résulte que pour éviter de concevoir un circuit avec d'importantes failles sécuritaires face à la *DPA*, il est indispensable de considérer tous les niveaux d'abstraction. Nous avons particulièrement détaillé les contre-mesures basées sur des portes logiques en présentant les avantages et les inconvénients. L'inconvénient majeur de la majorité des solutions présentées est qu'elles nécessitent le développement de nouvelles cellules et que toutes accroissent considérablement la surface et la consommation du circuit intégré dans lequel elles sont placées. La conclusion de ce chapitre est qu'il est primordial de respecter deux règles essentielles. Au niveau des portes logiques, il convient de réduire au maximum la consommation des éléments qui traitent les données sensibles. De manière générale, la génération de bruit permet de perturber au maximum l'acquisition et la synchronisation des courbes de consommation.

Dans le chapitre suivant, nous allons nous focaliser sur l'ajout de contre-mesures au niveau des portes logiques, compatibles avec l'utilisation de contre-mesures à tous les autres niveaux d'abstraction et qui ne présentent pas les inconvénients énoncés dans ce chapitre.

CHAPITRE IV.

CONTRE-MESURES BASEES SUR LA RECONFIGURATION DE CHEMINS

Dans ce chapitre, nous allons proposer une nouvelle contre-mesure, assimilable au principe de la logique masquée (voir § III.3.4.2) qui permet de déjouer les analyses en courant. Tout d'abord, nous définirons le principe générique de cette contre-mesure. Ensuite, nous évaluerons différentes règles d'utilisation de cette technique dans un circuit intégré. Nous nous baserons sur des simulations électriques d'une partie sensible du circuit. Enfin, nous réaliserons des simulations logiques d'un circuit complet, couplées avec un outil d'analyse de la consommation, afin de caractériser au mieux la résistance induite par les contre-mesures que nous proposons face à une analyse *DPA*. Toutes les évaluations seront déterminées par le nombre de vecteurs nécessaires pour retrouver la clé secrète : le paramètre N_C décrit dans le § III.2.3.

IV.1. Description du Path Jitter

La maturité des contre-mesures présentées dans le chapitre précédent n'est pas encore assez importante. En effet, l'intégration de ces cellules dans un flot de conception standard n'est pas aisée. De plus, pour la plupart, elles nécessitent le développement d'une nouvelle librairie de cellules, seule la logique WDDL s'en affranchit. La solution que nous proposons s'affranchit de ces problèmes car elle utilise les cellules logiques d'une librairie standard, ce qui est crucial dans un contexte industriel.

De plus, notre solution peut s'appliquer à une partie seulement d'un circuit en ajoutant les contre-mesures que nous proposons de façon distribuée, car il n'est pas nécessaire d'adapter les signaux qui les activent. Ceci n'est pas possible avec les logiques présentées dans le chapitre précédent qui fonctionnent toutes avec des données complémentaires, en entrée comme en sortie.

Notre approche consiste à modifier la topologie du circuit au cours de son exécution afin de modifier aléatoirement et dynamiquement la charge des nœuds internes ainsi que les délais de propagation des données. Dans la pratique, cela consiste à perturber l'obtention d'informations utiles suite à l'acquisition de la consommation en ajoutant du bruit au niveau des portes logiques. Nous désignons cette contre-mesure par le terme générique de « *path jitter* ».

Pour cela, le *path jitter* que nous proposons remplace une fonctionnalité déjà présente dans le circuit par *n* fonctionnalités identiques mais basées sur des implantations différentes. Sa description est donnée dans la Figure IV.1.

Les *n* implantations différentes sont englobées par deux blocs de sélection de chemin. Ces deux blocs permettent d'activer un seul des *n* chemins en fonction du signal de sélection de chemin, désigné par *Path Selection Signal*. Il est nécessaire que ce signal soit aléatoire afin que l'activation des chemins soit elle-même aléatoire. L'élément *In Path Selection Block* consiste à aiguiller les *i* signaux d'entrée vers un des *n* chemins de *i* signaux ; tandis que *Out Path Selection Block* connecte les *o* signaux d'un des *n* chemins vers les *o* signaux de sortie.

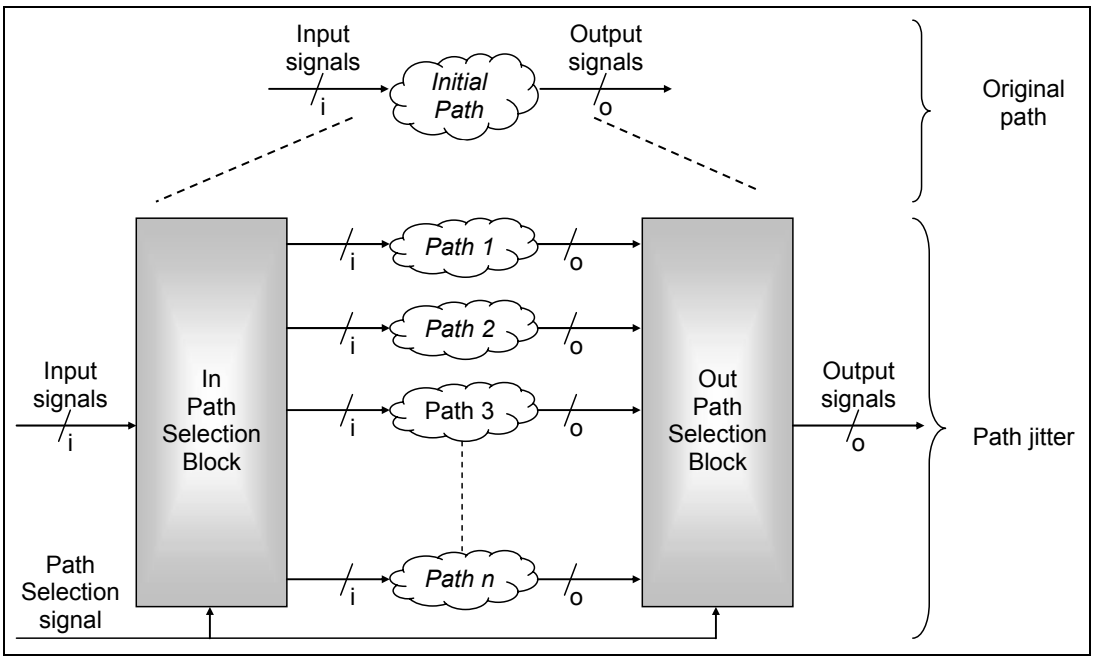

Figure IV.1 : Description d'un path jitter.

L'utilisation de cette architecture est parfaitement compatible avec les outils standards de conception de circuits intégrés : synthétiseur, simulateur logique, analyseur de délais, analyseur de puissance, analyse formelle… En revanche, l'inconvénient de cette logique est qu'elle multiplie la surface globale du circuit par le nombre de chemins implantés.

Le principal avantage de cette approche, hormis qu'elle masque les informations utiles, est qu'elle peut s'utiliser de façon distribuée dans un circuit sur n'importe qu'elle porte logique sans rendre les signaux complémentaires. Elle peut aussi bien s'adapter à des chemins constitués de cellules de délais, de portes logiques que de bascules.

De plus, il est important de noter que cette contre-mesure matérielle est parfaitement compatible avec l'utilisation de contre-mesures des autres niveaux d'abstractions présentés dans le § III.3.

IV.1.1. Chemins constitués de cellules de délais

Dans ce cas là, les chemins sont constitués de délais additionnels, différents pour chaque chemin.

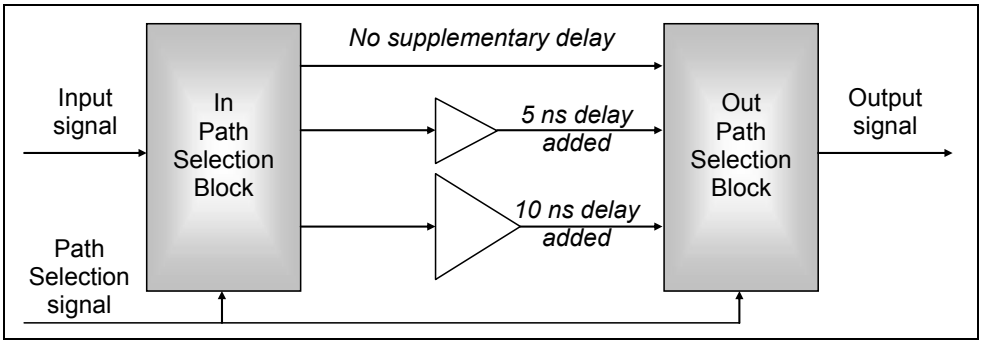

Figure IV.2 : Exemple d'ajout de retard aléatoire.

Cette approche permet de faire varier la propagation des données au travers des différents étages logiques.

La Figure IV.3, réalisée grâce au simulateur électrique eldo, permet d'apprécier comment la consommation peut être perturbée dans une SBOX0 (sous bloc de la fonction SBOX) qui contient seulement 10 path jitter basés sur les retards. Sur cette figure sont superposées plusieurs traces. Chaque trace correspond à la consommation de la SBOX0 pour les mêmes données traitées mais avec une topologie différente des chemins.

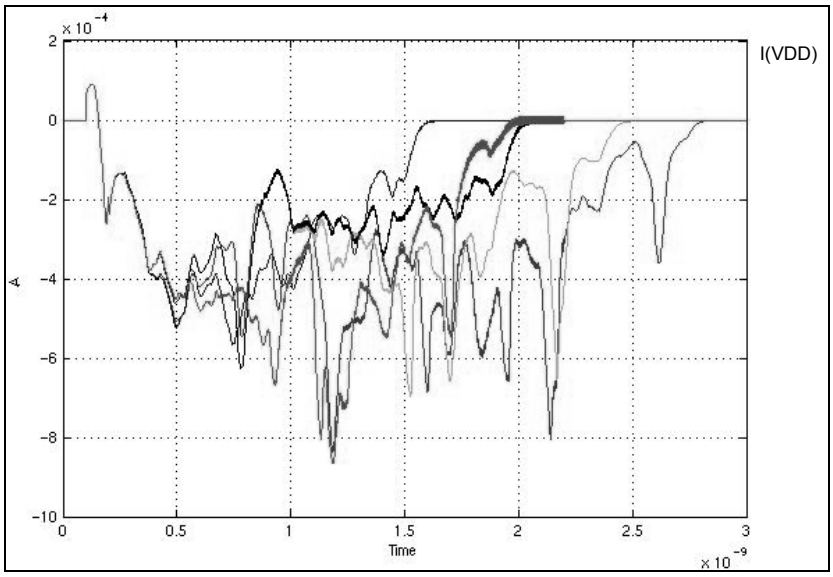

Figure IV.3 : Consommation instantanée lors de simulations de path jitter avec différents délais dans un bloc SBOX0.

IV.1.2. Chemins constitués de portes logiques

Il est possible d'appliquer la structure de la Figure IV.1 à des chemins constitués de fonctions logiques simples ou complexes. Rappelons que les chemins doivent avoir la même fonctionnalité mais basés sur des implantations différentes, pouvant être déduites à partir de la loi de De Morgan. Prenons l'exemple d'une fonction réalisant le « Ou exclusif ». De Morgan nous donne la relation suivante :

$$A \oplus B = A.\overline{B} + \overline{A}.B = \overline{\overline{A.\overline{B}} + \overline{\overline{A}.B}} = \overline{\overline{A.\overline{B}}.\overline{\overline{A}.B}}$$
$$= \overline{(\overline{A} + B).(A + \overline{B})} = \overline{(\overline{A} + B)} + \overline{(A + \overline{B})} \qquad \textbf{(Equ. 36)}$$

La Figure IV.4 présente comment appliquer la structure du path jitter à 2 chemins à une fonctionnalité « Ou Exclusif ». Un des 2 chemins est basé sur la cellule XOR de la librairie. L'autre chemin est constitué d'une combinaison de cellules reproduisant la même fonctionnalité obtenue par l'équation précédente. Pour augmenter l'asymétrie, il est possible d'utiliser dans le chemin additionnel des portes ayant des dimensions différentes ou encore d'insérer des délais supplémentaires. Tout dépend des cellules disponibles dans la librairie.

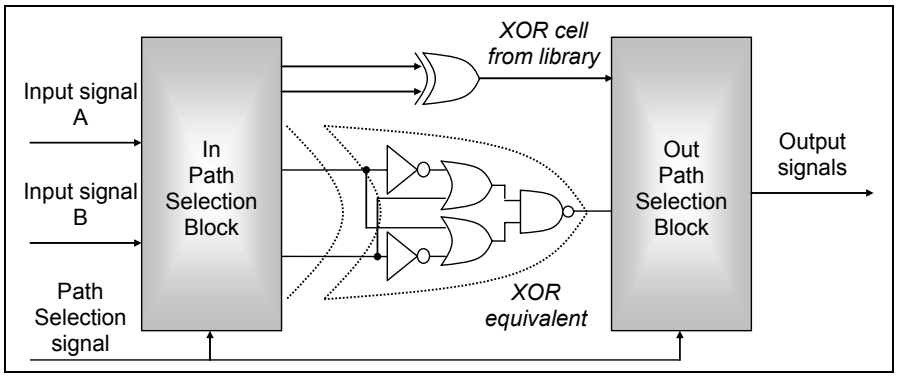

Figure IV.4 : Exemple d'application à des portes logiques.

Afin d'apprécier l'asymétrie de la consommation induite par le path jitter de la Figure IV.4., nous simulons toutes les transitions possibles en entrée. Pour une cellule à deux entrées comme celle que nous utilisons, il y a 14 transitions différentes. Elles peuvent être repérées dans la Figure IV.5 à l'aide des signaux A et B représentés sur la partie inférieure. La partie supérieure présente la consommation instantanée globale dans le cas où la cellule initiale est activée (*SEL=0*) et dans le cas où la cellule équivalente est activée (*SEL=1*). Les chronogrammes de cette figure ont été réalisés à l'aide du simulateur électrique eldo.

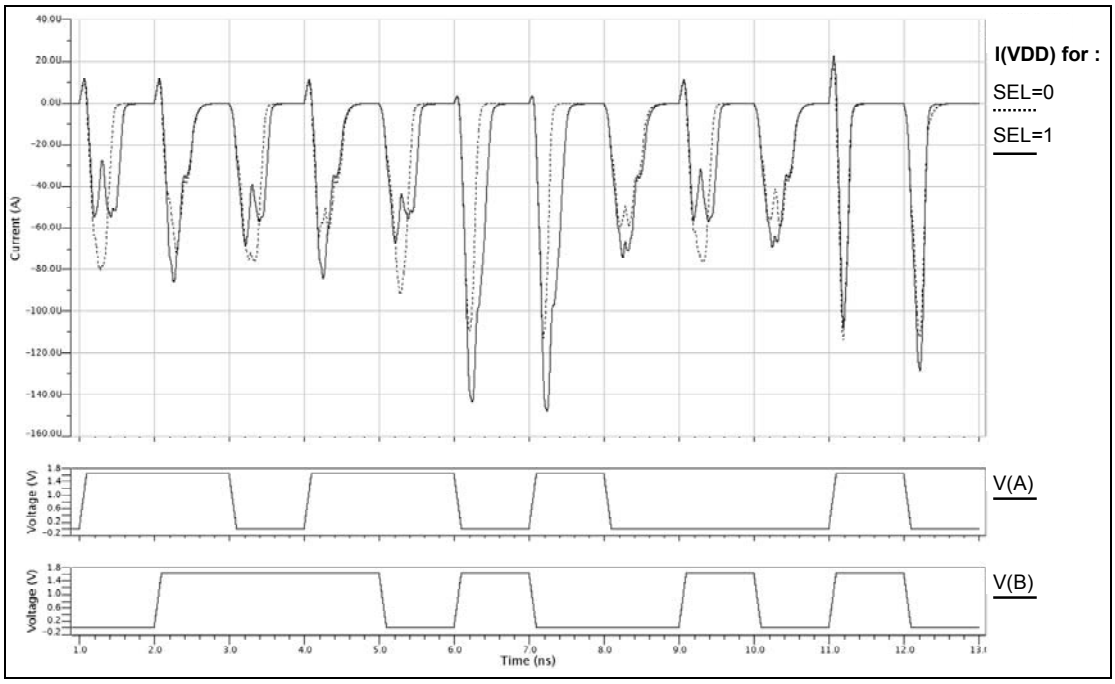

Figure IV.5 : Simulations de toutes les transitions pour un path jitter à 2 entrées.

L'asymétrie visible sur le chronogramme est reprise dans le Tableau IV-1 qui quantifie la différence entre les amplitudes pour chaque transition (en µA et en %) entre les cas où *SEL=0* et *SEL=1*. Notons que les écarts ne sont pas constants et que le chemin qui consomme le plus n'est pas toujours le même, traduits dans le tableau par des valeurs négatives et positives. La différence d'amplitude varie entre -30,1% et +26,9%, ce qui permet de perturber l'obtention d'informations à partir de l'analyse du

courant car l'état du signal de sélection change considérablement l'amplitude d'une même transition.

Transition sur A	0→1	1	1→0	0→1	1	1→0	0→1	1→0	0	0	0→1	1→0
Transition sur B	0	0→1	1	1	1→0	0→1	1→0	0	0→1	1→0	0→1	1→0
Différence d'amplitude (µA)	24,5	-13,7	7,6	-20,2	24,5	-33,9	-35	-15	18,8	-9,8	5,1	-16,2
Différence d'amplitude (%)	26,9	-16,3	10,0	-26,9	26,8	-30,0	-30,1	-25,3	21,6	-16,6	3,8	-14,4

Tableau IV-1 : Asymétrie de la consommation pour chaque transition.

Ces perturbations induites par le *path jitter* au niveau de la consommation sont relativement importantes en pourcentage. Ainsi, avec la propagation des données et le phénomène de diffusion présent dans le DES, la consommation d'une SBOX peut être considérablement perturbée. La Figure IV.6 représente ce phénomène lors de simulations électriques eldo. Elle représente la superposition de consommations instantanées d'un bloc SBOX comportant plusieurs *path jitter*. Le bloc SBOX traite toujours les mêmes données pourtant les consommations ne sont pas identiques.

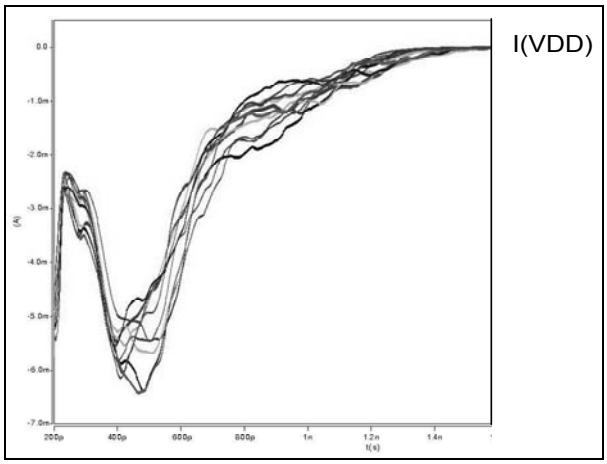

Figure IV.6 : Consommation instantanée d'un bloc SBOX intégrant plusieurs path jitter.

De plus ces courbes de consommations sont obtenues par simulation, dans des conditions idéales. C'est-à-dire que seule la fonction SBOX est simulée, la consommation du reste du circuit intégré ne perturbe pas la courbe de consommation.

IV.1.3. Chemins constitués de bascules

Intéressons nous maintenant à un autre élément essentiel constituant les circuits intégrés synchrones : les registres (ou FF pour Flip-Flop). Ces cellules utilisent relativement plus de surface que de simples portes logiques. Ainsi, il n'est pas envisageable d'assembler plusieurs bascules dans un *path jitter*. En revanche, une bascule FF standard a la propriété de disposer de deux sorties complémentaires. Nous utilisons cette propriété afin de créer une bascule qui échantillonne la donnée présente sur D ou l'inverse de la donnée D en fonction du signal de sélection.

L'échantillonnage a lieu lors d'un front montant du signal d'horloge (Clk). La Figure IV.7 présente une des structures capable de réaliser cette fonctionnalité à partir de la structure du *path jitter* que nous nommons JFF pour *Jittered Flip-Flop*. Lorsque la donnée exacte est échantillonnée, la sortie Q de la Flip-Flop est connectée vers la sortie. Lorsque l'inverse de la donnée est échantillonné, la sortie Qn (complémentaire de Q) est connectée vers la sortie.

La contrainte de cette structure est que le signal de sélection doit rester stable lorsque le chemin est utilisé. En effet, s'il venait à changer pendant que la donnée exacte est échantillonnée, l'inverse de la donnée serait dirigé vers la sortie altérant ainsi la fonctionnalité de la bascule.

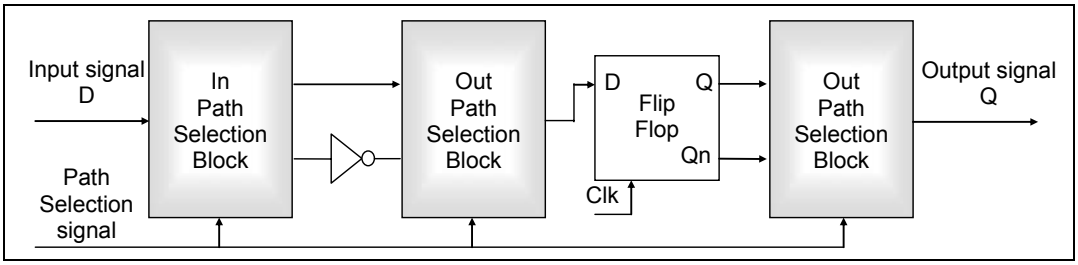

Figure IV.7 : Structure d'une JFF.

La Figure IV.8 présente les résultats de simulation de cette structure ; trois consommations instantanées sont présentées correspondantes à :

- Une FF seule,
- Une JFF configurée en échantillonnage exact (*SEL=0*),
- Une JFF configurée en échantillonnage inverse (*SEL=1*).

Dans le cas d'une bascule, nous étudions les échantillonnages *0→1* (repère 1) et *1→0* (repère 2). Lorsque la donnée échantillonnée est inchangée (*1→1* et *0→0*), les consommations sont identiques pour les trois structures comparées et par conséquent ne sont pas représentées sur la Figure IV.8. Sur la partie supérieure du chronogramme est affiché le signal d'horloge afin de faciliter le repérage de ces transitions. Cette figure a été obtenue avec le simulateur eldo.

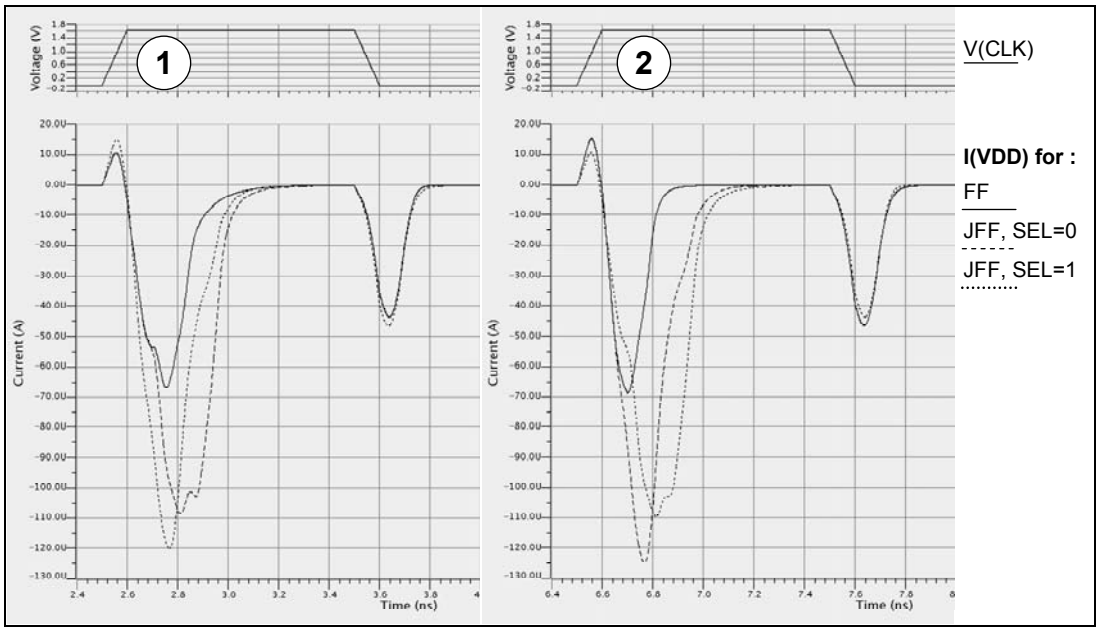

Figure IV.8 : Comparaison des profils de consommation lors des échantillonnages 0→1 et 1→0.

Notons que la consommation lors de la transition *0→1* pour *SEL=0* est identique à la consommation lors de la transition *1→0* pour *SEL=1* (et inversement). Cela prouve qu'il est possible de fausser l'obtention d'informations à partir de l'observation du courant consommé. Le tableau suivant (Tableau IV-2) quantifie l'asymétrie en comparant la différence d'amplitude du courant lorsque *SEL=0* et *SEL=1*.

Echantillonnage	0→1	1→0	1→1	0→0
Différence d'amplitude (µA)	19,5	-15,9	3,2	-3,2
Différence d'amplitude (%)	13,9	-13,3	11,7	-13,3

Tableau IV-2 : Asymétrie de la consommation pour chaque transition.

La différence de +/- 13 % est conséquente. Nous verrons par la suite l'impact de cette asymétrie lorsque cette JFF est instanciée dans un circuit.

IV.1.4. Implantation

L'inconvénient majeur de la structure de nos *path jitter* est qu'elle augmente considérablement la surface des parties logiques (ou glue logique). Le facteur multiplicatif dépend du nombre de chemins différents instanciés dans un *path jitter*. Admettons qu'une cellule *path jitter* avec seulement deux chemins occupe deux fois plus de surface qu'une cellule classique équivalente. Pour cette raison, étudions comment évolue cette surface en fonction des zones que nous voulons sécuriser. Tout d'abord, prenons les hypothèses suivantes quand à l'occupation d'un circuit par les différents blocs :

- 50 % pour les blocs mémoire,
- 10 % pour les blocs analogiques,
- 40 % pour les blocs logiques (glue logique et macro cellules numériques),
  - dont 10 % pour un bloc DES.

Ces estimations sont typiques pour un circuit sécurisé, comme représenté dans la figure suivante.

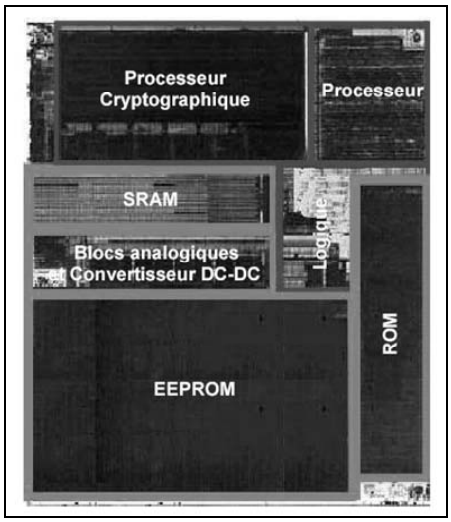

Figure IV.9 : Photographie d'une carte à puce.

Si nous souhaitons appliquer la structure proposée à l'intégralité des portes logiques d'un circuit et en tenant compte de la proportion donnée ci-dessus, la multiplication de la surface est estimée à 1,4 fois la surface initiale. Ce qui est considérable et non envisageable pour des raisons de coût. De plus, cette solution « naïve » ne présente pas un intérêt d'un point de vue sécuritaire car tout le circuit ne manipule pas des données confidentielles.

Si nous appliquons cette structure seulement au bloc DES, ce coefficient tombe à 1,1 %. Ce rapport est convenable mais la sécurisation complète de la macro-cellule n'est certainement pas indispensable.

Si nous l'appliquons aux zones sensibles d'un DES, ce coefficient est estimé à 1,05 %. Ce qui permet de proposer une solution résistante aux attaques avec un encombrement additionnel faible.

Dans la partie suivante, nous allons explorer le placement des contre-mesures et évaluer la résistance des différents placements face aux analyses en courant.

IV.2. Exploration du placement des contre-mesures

Plutôt que de sécuriser tout un circuit ou une macro cellule complète, nous allons étudier le placement réparti de notre structure dans les parties sensibles du

système. Notre structure se prête parfaitement à cette distribution dans le circuit car elle ne manipule pas des données complémentaires, contrairement à la plupart des contre-mesures présentées précédemment (voir § III.3).

Pour évaluer l'efficacité de nos contre-mesures, nous étudierons une fonction DES implantée dans un circuit. Nous partons d'une implantation initiale à laquelle nous ajouterons des *path jitter* placés différemment.

Rappelons que notre objectif est de réaliser un circuit résistant à la *DPA* à moindre coût. Afin d'évaluer l'amélioration de la résistance de notre proposition face à une analyse *DPA*, nous observerons l'évolution du nombre d'échantillons nécessaires pour « casser » ce DES, le paramètre N_C.

Les contre-mesures vont être positionnées selon certaines règles, basées sur des paramètres extraits de l'implantation initiale. Présentons tout d'abord le protocole de placement avant de présenter l'évaluation proprement dite.

IV.2.1. Collecte de paramètres sur le circuit

Pour nous aider à placer les contre-mesures judicieusement, nous avons besoin de collecter suffisamment de paramètres sur l'activité des portes du circuit lors de chiffrement DES : occurrence des transitions, puissance consommée lors des transitions … Pour cela nous utilisons le protocole représenté dans la Figure IV.10.

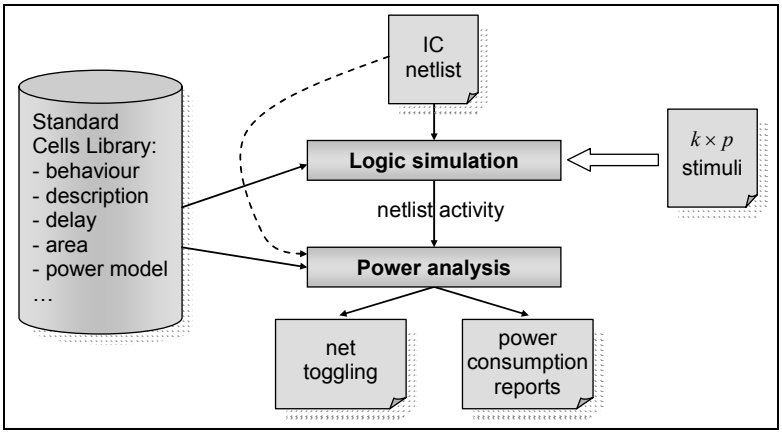

Figure IV.10 : Enregistrement de l'activité du circuit.

- **Description du circuit intégré (ou *IC netlist*) :**

Le point d'entrée de ce flot est constitué d'un fichier contenant la description du circuit intégré. Elle correspond à la *netlist* obtenue après la synthèse logique de toutes les fonctions numériques du circuit. Cette *netlist* post-synthèse est constituée de portes logiques de la librairie de cellules standard. L'étape de synthèse qui génère cette *netlist* à partir de la description RTL ne fait pas l'objet de notre étude.

- **Simulation logique (ou *Logic simulation*) :**

La simulation logique de la macro cellule DES est réalisée au niveau porte. Le logiciel SimVision [CAD] est un bon outil pour réaliser ces simulations. Pour

fonctionner, il est nécessaire de fournir à ce dernier une description des cellules utilisées (fonctions logiques, délais de propagation, temps de montée et de descente …) regroupée dans une librairie de cellules.

La simulation logique est constituée d'une succession de $k \times p$ chiffrements DES, où k représente le nombre de clés utilisées et p le nombre de textes clairs différents. Nous faisons varier la clé car si nous basons notre analyse sur une seule et unique clé donnée pour renforcer le circuit, il est fort probable qu'il ne sera pas renforcé si une clé différente est utilisée. C'est pour cette même raison que nous faisons varier les textes clairs.

La Figure IV.11 représente le chronogramme obtenu par simulation d'un chiffrement DES. Nous retrouvons la permutation initiale (*IP*), la permutation finale (*FP*) ainsi que les 16 rondes (*Ri*). Entre chaque ronde, nous pouvons aussi deviner combien de décalages de clé ont lieu. La structure du DES a été présentée dans la partie I.3.2.1.

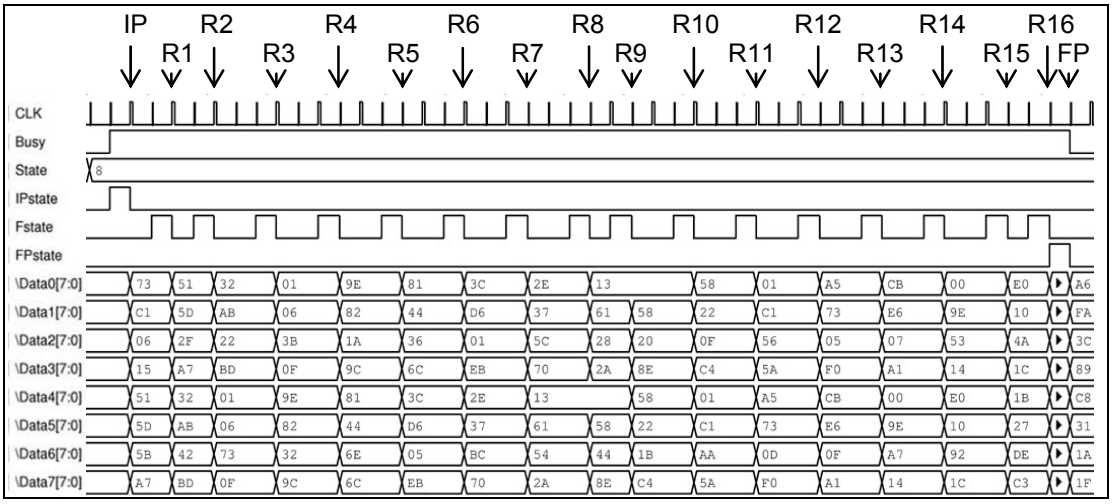

Figure IV.11 : Simulation logique d'un seul chiffrement DES.

Ce chiffrement est reproduit p fois pour k clés différentes avec k et p suffisamment élevés. En revanche, toutes les combinaisons ne sont pas reproduites car cela nécessiterait $2^{56} \times 2^{64} = 2^{120} = 1{,}3.10^{36}$ chiffrements. Dans notre étude nous considérons que $k>20$ et $p>2000$ suffisent à couvrir de nombreuses combinaisons.

Lors de cette simulation logique, l'outil enregistre l'activité de tous les nœuds internes de la *netlist* d'entrée et les stocke dans un fichier au format *VCD* (pour *Value Change Dump*).

- **Analyse de la puissance (ou *Power Analysis*) :**

L'analyse de la puissance peut être réalisée avec l'outil PrimePower distribué par la compagnie Synopsys [SYN]. Cette étape consiste à analyser le fichier contenant l'activité des nœuds internes (fichier *VCD*) avec les informations relatives à la consommation des cellules, informations présentes dans la librairie. L'outil permet d'extraire des paramètres globaux dynamiques ou statiques (puissance instantanée

globale consommée par la *netlist* par exemple) ou précis pour chaque porte et pour chaque nœud. Dans notre cas, nous nous intéressons aux paramètres fins comme :

- Le nombre de transitions de chaque nœud,
- La consommation moyenne dynamique pour chaque porte,
- La puissance moyenne consommée par chaque porte pour chaque transition sur sa sortie,
- La consommation moyenne lorsque des *glitch* apparaissent.

Les paramètres relevés à la fin de ce flot sont intimement liés à l'implantation de la fonction DES : optimisation en surface ou en puissance, technologie CMOS utilisée, architecture du DES …

L'outil est capable de générer des fichiers dans lequel les cellules sont classées en fonction de ces différents paramètres.

Afin d'évaluer l'impact de nos contre-mesures, nous n'étudierons qu'une partie de la fonction DES, la première ronde seulement car toutes les rondes sont similaires. De plus, nous nous focaliserons particulièrement sur les cellules logiques utilisées pour réalisées la fonction SBOX0 car toutes les SBOX sont basées sur la même implantation.

Cette restriction n'est pas fortuite. Nous prenons cette décision pour réduire les temps de traitement. En revanche, le choix de la SBOX0 de la première ronde est arbitraire. En effet, si nous parvenons à renforcer cette SBOX, nous pourrons renforcer les autres de la même façon.

Ainsi, les fichiers générés par notre analyse ne contiendront que les cellules relatives à la fonction SBOX0 de la première ronde.

IV.2.2. Placement des contre-mesures

Nous avons développé un outil capable de placer automatiquement des contre-mesures en fonction d'un fichier contenant les noms des portes à remplacer. Cet outil est représenté dans la Figure IV.12.

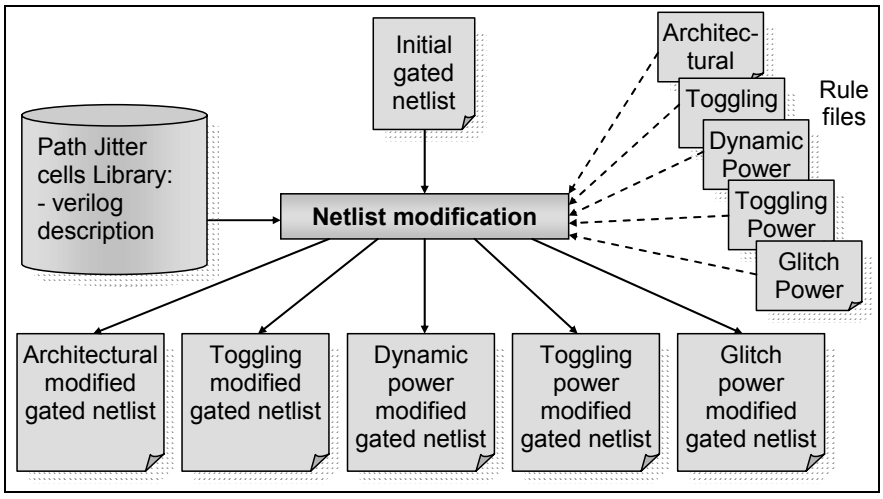

Figure IV.12 : Outil d'analyse et de modification de netlist.

- **Netlist Initiale :**

La *netlist* initiale est le fichier qui correspond à la description Verilog de la première ronde du DES au niveau porte. Dans notre cas, elle correspond à l'architecture de la Figure IV.13 et ne représente qu'une partie de la macro cellule DES.

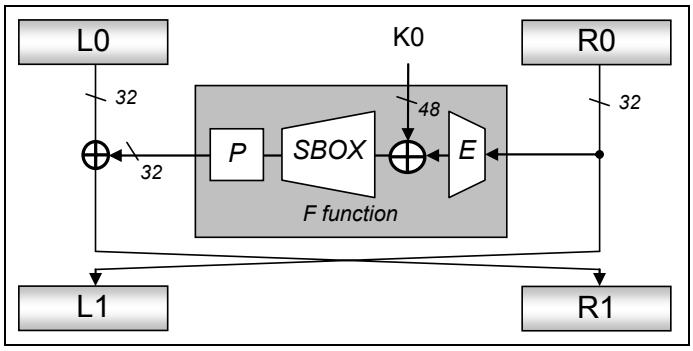

Figure IV.13 : Architecture de la netlist initiale.

- **Librairie de contre-mesures :**

Cet élément contient les différentes structures possibles des *path jitters*. Elle contient donc la description de *path jitters* basés sur des cellules de retard, des JFF ainsi que la structure appliquée à différentes fonctions logiques simples (XOR, AND, NAND, INV, OR3, NAND4 …) ou complexes. Les éléments de cette librairie ont été préalablement validés.

- **Règles de modification :**

Pour fonctionner notre outil a besoin d'un fichier contenant la liste des cellules à modifier. Ces fichiers sont générés à l'aide des paramètres collectés par l'étape décrite précédemment (voir § IV.2.1).

Prenons l'exemple du fichier correspondant à l'activité des nœuds. Il contient seulement les cellules relatives à la fonction SBOX0 de la première ronde dont l'activité est la plus élevée. En considérant la cellule pour laquelle le nombre de transitions est maximum, le fichier ne contient que les cellules dont l'activité est supérieure à 90% de cette valeur. Ce raisonnement de 90% est maintenu pour toutes les règles de placement déduites des paramètres du circuit.

En plus de ces placements consécutifs à l'étude du fonctionnement du circuit, une règle relative à l'étude de l'architecture du circuit est employée. Elle consiste à renforcer les cellules jugées sensibles dans le cas des analyses en puissance.

Ainsi, les règles utilisées sont basées sur :

- l'étude de l'architecture : c'est-à-dire que les cellules « ou exclusif » situées en périphérie de la SBOX0 seront renforcées ainsi que les registres d'entrée et de sortie de ronde.
- le nombre de transitions de chaque nœud : dans cette règle sont regroupées des cellules « ou exclusif » ainsi que des registres. En plus, apparaissent des cellules de la logique combinatoire de la SBOX0. Ces dernières sont situées en fin de cône logique de la SBOX0, juste avant les « ou exclusifs ».
- la consommation moyenne dynamique pour chaque porte : ce placement regroupe essentiellement des cellules appartenant à la fonction SBOX0, équitablement réparties dans le cône logique de cette dernière.
- la puissance moyenne consommée par chaque porte pour chaque changement d'état de sa sortie : tout comme pour le classement précédent, ces cellules logiques sont réparties dans la logique qui réalise la fonction SBOX0. Aucun registre n'est renforcé.
- la consommation moyenne des *glitches* qui apparaissent sur les nœuds : cette règle de placement contient elle aussi essentiellement des cellules de la logique de la SBOX0. Là aussi, aucun registre n'est présent.

- **Modification des cellules :**

L'outil parcourt la *netlist* initiale à la recherche des cellules à modifier contenues dans le fichier précisant la règle utilisée pour les remplacer par les cellules renforcées équivalentes décrites dans la librairie.

- **Netlist renforcées :**

Pour chaque règle utilisée, l'outil génère une *netlist* en description Verilog équivalente à la *netlist* d'entrée mais modifiée de cinq façons distinctes. La fonctionnalité de ces *netlists* est vérifiée à l'aide de simulations logiques et de vérification formelle.

Rappelons que seules les cellules correspondantes à la SBOX0 sont susceptibles d'être modifiées.

IV.2.3. Evaluation des contre-mesures

Dans cette partie nous allons comparer la résistance des différents placements des contre-mesures dans la SBOX0 face aux attaques *DPA*. Nous nous limiterons à des simulations électriques de la structure présentée dans la Figure IV.13. Cette structure représente une partie de la macro cellule DES. Cette macro cellule représentant elle-même une partie d'un circuit intégré.

Présentons tout d'abord notre méthodologie d'évaluation de la résistance avant de procéder à la comparaison proprement dite.

IV.2.3.1. Protocole d'évaluation

Pour évaluer une netlist, il convient dans un premier temps de collecter un grand nombre d'échantillons relatifs au traitement de calculs cryptographiques par cette netlist. Ces échantillons sont des courbes de consommation instantanée. Dans un second temps, ces courbes sont analysées par un outil capable de réaliser la *DPA*.

- **Collecte des échantillons :**

Afin d'être précis dans notre collecte d'échantillons, nous basons notre analyse sur des simulations électriques au niveau transistor, car les modèles sont bien plus précis que ceux utilisés pour une simulation logique. En contre partie, une simulation électrique demande une puissance de calcul plus élevée. Les outils eldo [MENb] ou spice [SPI] font partie des simulateurs électriques les plus connus capables de réaliser ces simulations.

La Figure IV.14 présente le déroulement de cette étape. La *netlist* correspondant à la description Verilog est tout d'abord transformée en un format interprétable par le simulateur électrique.

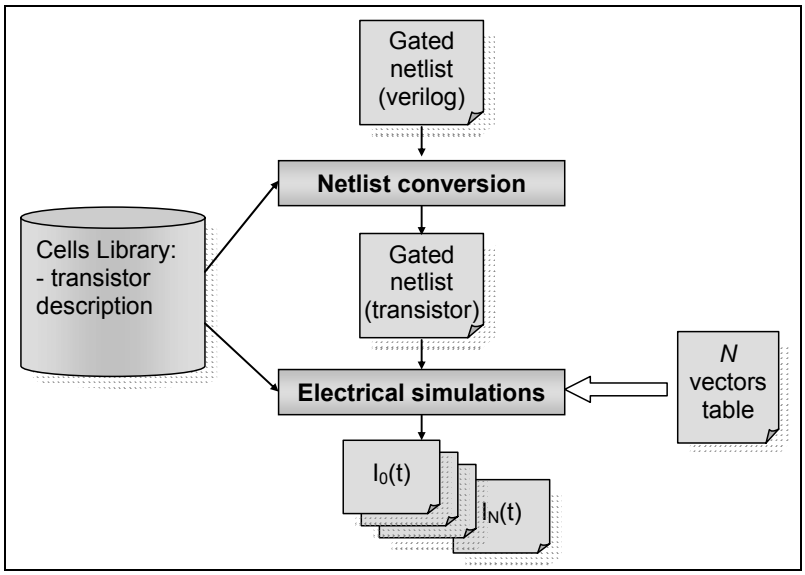

Figure IV.14 : Obtention des courbes de consommation.

Ensuite, nous réalisons *N* simulations, d'un cycle d'horloge chacune correspondant au déroulement de la première ronde. Nous réalisons cette restriction afin d'économiser du temps de simulation.

De plus, seule cette ronde est simulée car si celle-ci peut être cassée, les suivantes le seront aussi avec la même difficulté (voir § III.2.4). Les *n* simulations sont réalisées en utilisant la même clé K_0, par contre les données entrantes (L_0 et R_0) sont aléatoires et toutes différentes, engendrant ainsi des consommations différentes comme le fait apparaître la Figure IV.15. Sur ce chronogramme sont superposées les consommations $I_{VDD}(t)$ pour plusieurs simulations.

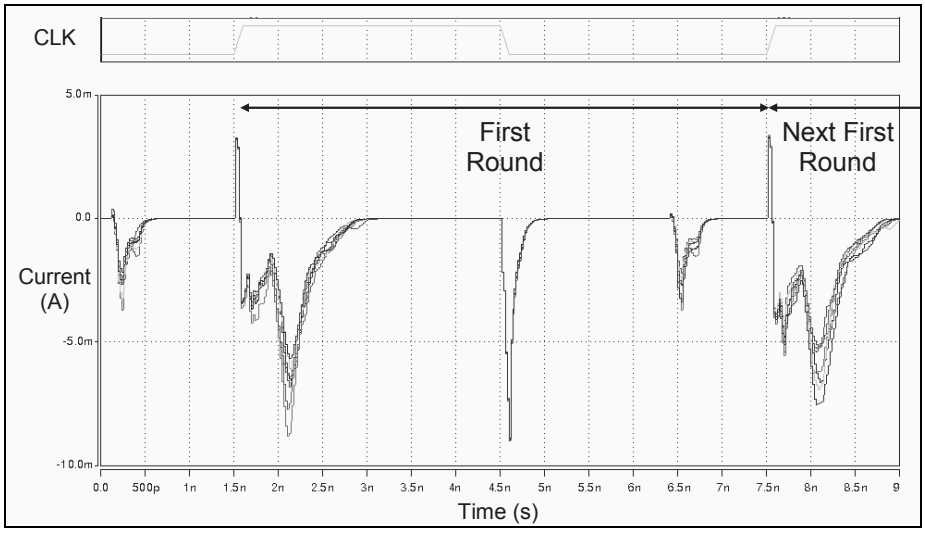

Figure IV.15 : Simulations électriques de la première ronde pour l'implantation initiale et consommations $I_{VDD}(t)$ associées.

Sur cette figure sont visibles trois évènements importants :

- *t=1,5 ns* : front montant de l'horloge (*CLK*). Lors de cet évènement la première série de registres échantillonne les données présentes sur les ports d'entrée. Ces données sont redirigées vers les sorties de ces registres et stimulent la logique combinatoire avec ces nouvelles données. A partir de cet évènement, environ 1,5 ns sont nécessaires pour stabiliser la logique de cette implantation.
- *t=4,5 ns* : front descendant de l'horloge. Durant cet événement un certain nombre de commutations ont lieu dans les registres eux mêmes entraînant un pic de consommation. Toutes les consommations sont identiques.
- *t=7,5 ns* : deuxième front montant de l'horloge. Cet évènement coïncide à l'échantillonnage de nouvelles données par les bascules d'entrée de la ronde. Comme pour *t=1,5 ns*, ces nouvelles données activent la logique combinatoire. Par contre un évènement supplémentaire apparaît. Les registres de sortie capturent les données traitées par la ronde.

Ces trois fronts se retrouveront dans toutes les implantations évaluées. Par contre les délais de propagation des données au travers de la logique combinatoire présentés dans ce chronogramme coïncident avec l'implantation initiale. Ces délais seront légèrement altérés lorsque les implantations seront modifiées.

- **Analyse DPA :**

Les *n* échantillons obtenus par simulation sont fournis à notre outil d'analyse *DPA* (voir Figure IV.16) qui génère les coefficients de corrélation pour toutes les hypothèses de clés de la SBOX0, soit 64 clés différentes. Ces coefficients peuvent être interprétés de multiples façons, comme expliqué dans le CHAPITRE III (temporelle, 3D ou évolution en fonction du nombre d'échantillons). Nous préfèrerons observer l'évolution des coefficients de corrélation en fonction du nombre d'échantillons.

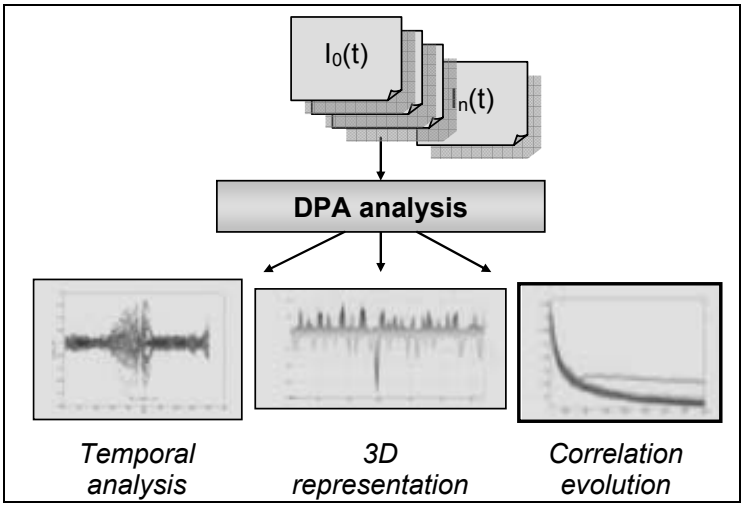

Figure IV.16 : Phase d'analyse DPA.

Pour chaque implantation étudiée, nous procèderons à deux analyses distinctes que notre outil est capable de réaliser :

- DPA_K : basée sur l'analyse décrite par Kocher (§ III.2.4),

- DPA_H : basée sur l'analyse du poids de Hamming (§ III.2.4.3).

Les résultats de ces deux analyses nous permettrons d'évaluer la résistance d'une implantation par rapport aux autres en révélant le nombre d'échantillons nécessaires N_C pour trouver la bonne clé ainsi que le degré de certitude.

IV.2.3.2. Implantation initiale

Le flot complet présenté dans le paragraphe précédent est maintenant appliqué à toutes les implantations. Procédons tout d'abord à l'évaluation de la *netlist* initiale (voir Figure IV.13) dans laquelle aucune contre-mesure n'est présente.

La Figure IV.17, représente l'évolution des coefficients de corrélation pour les analyses DPA_K et DPA_H. La méthodologie qui nous permet d'obtenir ces évolutions est expliquée dans le chapitre précédent. En gras figure la bonne clé. Ces deux analyses révèlent que $N=500$ échantillons sont nécessaires pour casser la clé avec DPA_K et $N=1\,000$ pour DPA_H. Nous ne retenons que la plus faible de ces deux valeurs, soit $N_C=500$ pour cette implantation.

Attention, ce chiffre annoncé n'est pas réaliste. En effet, dans notre cas nous réalisons une simulation. Ce qui signifie que nous nous affranchissons de tout le bruit induit par les mesures et du filtrage induit par les régulateurs de tension présents dans la plupart des circuits intégrés. De plus, seule la logique de la ronde est simulée. Ainsi, la logique du reste du circuit ne perturbe pas le courant que nous observons.

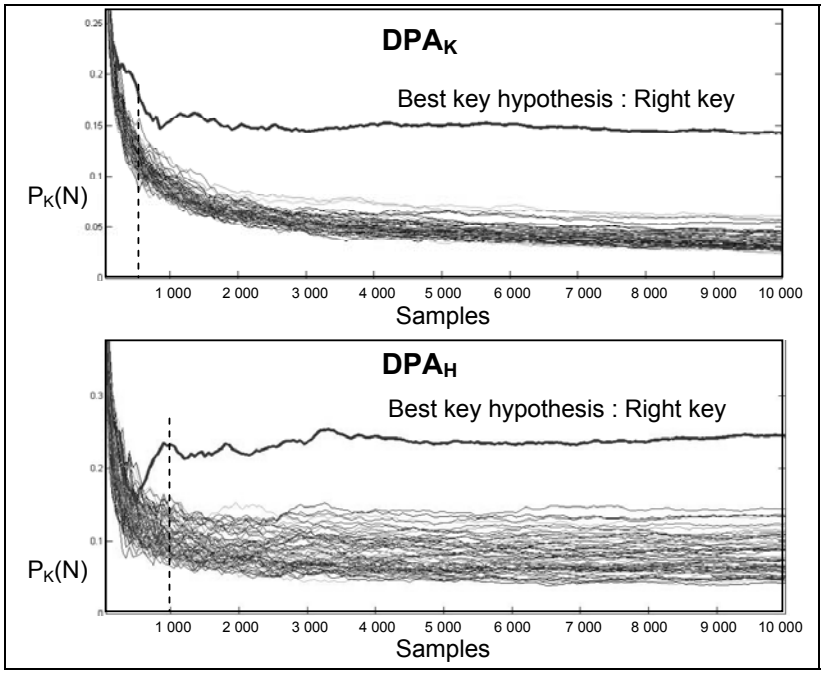

Figure IV.17 : Analyses DPA_K et DPA_H pour la cellule initiale.

La puissance de notre flot d'évaluation est qu'il nous permet de distinguer très précisément l'apparition des phénomènes qui permettent à une analyse *DPA* de trouver la clé. Ceci n'est pas visible sur la figure précédente. Procédons à une représentation temporelle de ces coefficients de corrélation pour un nombre d'échantillons N où la bonne clé est révélée. Fixons par exemple $N=5\,000$ où

P_K=~*0,15* pour DPA_K et P_K=~*0,25* pour DPA_H. Rappelons que P_K correspond à la valeur maximale de $\rho_K(t)$ pour un nombre d'échantillon donné.

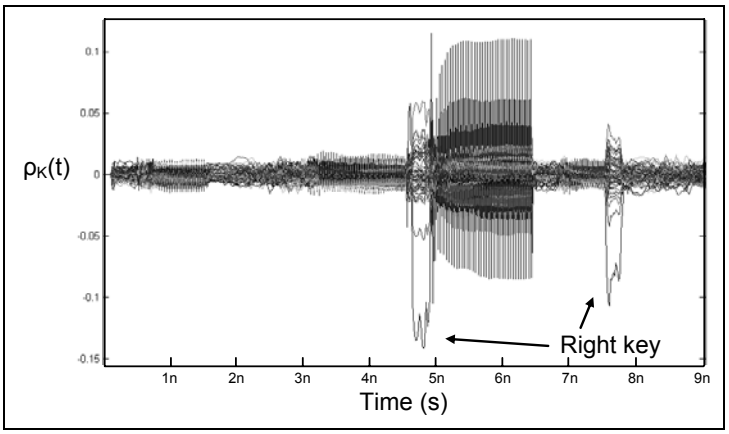

Figure IV.18 : Représentation temporelle de DPA_K pour la cellule initiale.

Dans la représentation de la Figure IV.18, nous retrouvons bien la valeur précédente $Max(|\rho_K(t)|) = \sim 0{,}15$. De plus, elle révèle le phénomène qui permet de casser la clé pour t_{K1}=*4,7ns* et t_{K2}=*7,5ns*. En nous référant à la Figure IV.15, nous constatons que le premier évènement correspond à un front descendant du signal d'horloge ; le deuxième évènement à un front montant.

Les corrélations apparaissant pour t_{K1} et t_{K2} s'expliquent par la structure des cellules FF que nous utilisons ainsi que par le modèle que nous simulons.

Ces cellules FF sont construites de telle sorte qu'en entrée, un élément laisse passer les données lorsque *CLK=0* et les bloque lorsque *CLK=1*. Ainsi, lors d'un front descendant de *CLK* (*1→0*), la donnée présente en entrée de la FF active les nœuds internes de la cellule. Dans notre cas précis, les registres incriminés sont les registres de sortie de ronde. En effet, à $t=t_{K1}$ les données traitées par la ronde en fonction de la clé sont présentes sur ladite structure d'entrée des FF de sortie. Lors de la transition descendante de *CLK*, des transitions significatives ont lieu. Ces transitions sont directement liées aux bits sur lesquels l'analyse *DPA* se focalise.

Pour $t=t_{K2}$, un autre phénomène se produit. Les FF utilisées ont une autre particularité intrinsèque à leur fonctionnalité. Elles sont composées de deux étages distincts, séparés par une structure qui laisse transiter la donnée échantillonnée par le premier étage vers le deuxième étage lorsque *CLK=1*. Lorsque *CLK=0*, ces deux étages sont isolés. Ainsi, lors d'une transition montante de *CLK* (*0→1*) le deuxième étage se charge avec la donnée du premier. Pour $t=t_{K2}$, de nouvelles données sont présentes en entrée des registres de sortie de la ronde. Ces nouvelles données, dépendantes des données de la clé chargent les nœuds internes des registres qui appellent du courant de façon significative pour une analyse *DPA*.

Par la suite, nous observerons si l'apparition de ces deux évènements pour $t=t_{K1}$ et $t=t_{K2}$ persistent lorsque nous ajoutons nos contre-mesures.

Etudions maintenant la représentation temporelle de $\rho_K(t)$ lors d'une analyse DPAH basée sur le poids de Hamming (voir Figure IV.19).

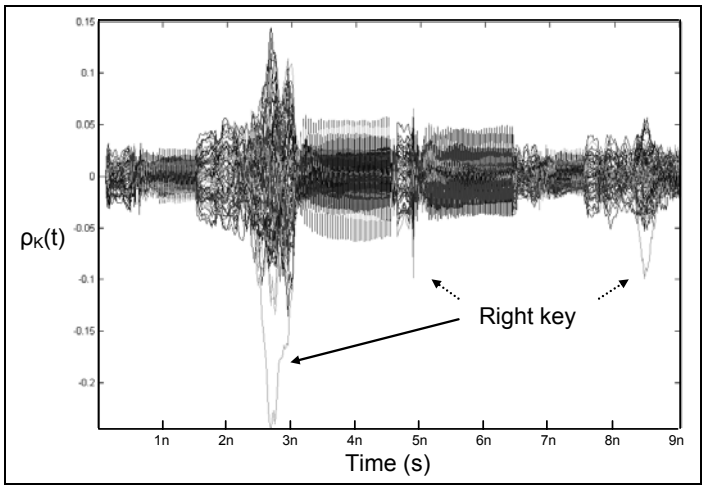

Figure IV.19 : Représentation temporelle de DPA_H pour la cellule initiale.

La Figure IV.19 fait bien apparaître $|\rho K(t)|$=~0,25 que nous avions précédemment dans la Figure IV.17 et nous donne l'information sur l'instant d'apparition du phénomène : t_H=*2,5 ns*. Nous observons aussi des maximums locaux lors du front descendant et du deuxième front montant de *CLK*.

En fait, t_H correspond très précisément au moment où les données échantillonnées par le premier étage de bascules de la ronde ont traversé tous les étages de la logique combinatoire qui constitue la fonction F. En effet, toujours selon la Figure IV.15, nous observons que le circuit ne consomme plus de courant pour *t=2,7 ns*, signifiant que tous les nœuds de la logique interne sont chargés.

Nous étudierons si cet évènement persiste lorsque la cellule contient nos contre-mesures.

Nous pouvons conclure que sur cette cellule DPA_K est plus efficace pour cibler les transitions liées aux registres. DPA_H est plus efficace pour détecter les corrélations apparaissant dans la logique combinatoire.

IV.2.3.3. Implantations avec contre-mesures

Réalisons maintenant le protocole d'évaluation présenté au § IV.2.3.1 sur les cinq *netlists* renforcées décrites dans § IV.2.2. Nous allons ainsi évaluer cinq règles distinctes.

- **Contre-mesures basées sur l'étude de l'architecture :**

La Figure IV.20 représente l'évolution de $P_K(N)$ en fonction du nombre d'échantillon pour l'analyse DPA_K (partie haute de la figure) et DPA_H (partie basse de la figure). Constatons tout d'abord que le nombre d'échantillons nécessaire pour casser la clé utilisée par le système est considérablement élevé par rapport à l'implantation initiale puisqu'un attaquant a maintenant besoin :

- de ~6 000 échantillons pour DPA_K, soit ×12 par rapport à DPA_K de l'implantation initiale,

- de ~2 000 échantillons pour DPA_H, soit ×2 par rapport à DPA_H de l'implantation initiale.

Pour l'implantation initiale et quelle que soit l'analyse pratiquée, $N_C=500$. Pour cette implantation, $N_C=2\ 000$, soit une amélioration relative de N_C par un facteur 4. Nous définissons par F_C ce facteur d'amélioration relative par rapport à l'implantation sans contre-mesure.

Notons aussi que dans les deux cas étudiés, l'écart absolu entre la bonne clé et la seconde hypothèse est beaucoup plus faible que précédemment. Avec une remarque particulière pour DPA_K où l'écart de 12% en valeur relative entre les coefficients de corrélation de la meilleure hypothèse et de la seconde correspond à un écart absolue de seulement 0,005. Cet écart est infime mais la précision de nos mesures permet tout de même de le révéler.

En l'état actuel, nous ne pouvons déterminer avec précision le nombre exact de mesures nécessaires si jamais il s'agissait de résultats relatifs à l'étude d'un circuit réel. C'est pour cette raison que nous ne retenons que le facteur d'élévation relative du nombre d'échantillons : F_C.

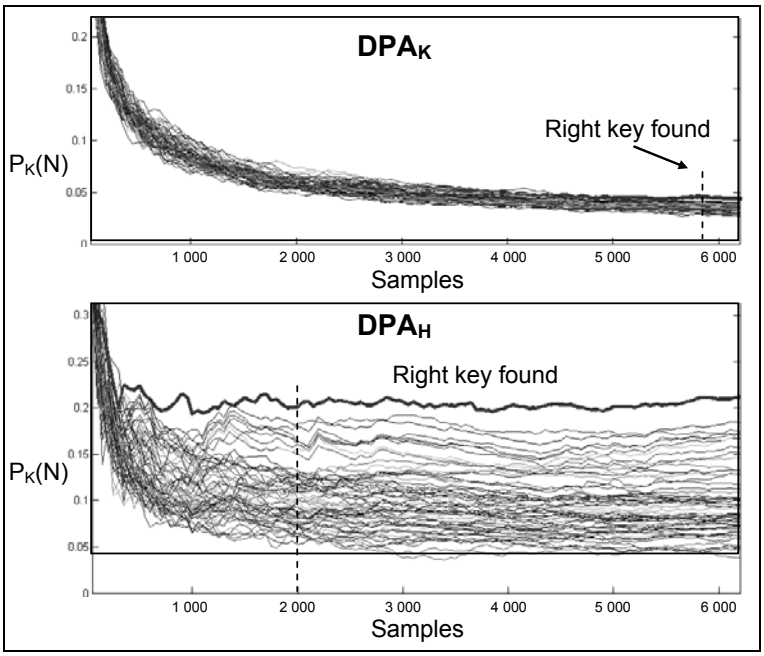

Figure IV.20 : Analyses DPA_K et DPA_H pour la cellule modifiée en fonction de l'étude de l'architecture.

Procédons à l'étude de la représentation temporelle des coefficients de corrélation pour $N=6\ 200$.

Concernant DPA_K (voir Figure IV.21), l'évènement qui révèle la clé se situe à $t_K=8,2\ ns$ et correspond au début de l'activation de la logique lors du deuxième front montant par les nouvelles données échantillonnées par les bascules d'entrée de ronde. La réussite de DPA_K est donc due aux portes logiques et non plus aux registres comme pour l'implantation initiale.

Pour DPA_H, l'évènement a lieu pour $t_H=3,9\ ns$. Comme expliqué précédemment, nous ne pouvons plus nous reporter au chronogramme de la Figure

IV.15 car les délais de propagations des données au travers de la logique combinatoire sont augmentés par les *path jitter*. En fait durant t_H, seule la logique combinatoire est activée révélant ainsi une corrélation entre les données traitées et la consommation $I_{VDD}(t)$.

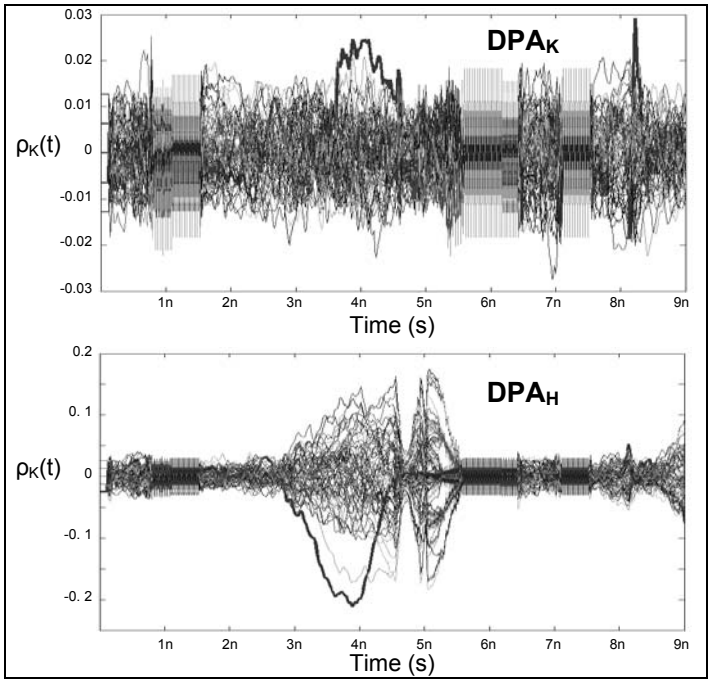

Figure IV.21 : Représentation temporelle de DPA_K et DPA_H pour la cellule modifiée en fonction de l'étude de l'architecture.

- **Contre-mesures placées en fonction de l'activité du circuit :**

Pour cette implantation, les contre-mesures ont été placées en fonction de l'activité du circuit. Les cellules qui subissent le plus grand nombre de transitions sur leurs nœuds de sortie ont été remplacées par des *path jitters*.

La Figure IV.22 présente l'évolution des coefficients de corrélation pour DPA_H seulement car DPA_K n'aboutit pas pour 2 600 simulations (voir Annexe B). Nous ne réalisons pas plus de simulations car DPA_H trouve la bonne clé pour environ $N_C=300$ seulement. Ainsi, ce placement des contre-mesures n'est pas judicieux car il n'améliore pas la résistance de la fonction DES face à une analyse *DPA*.

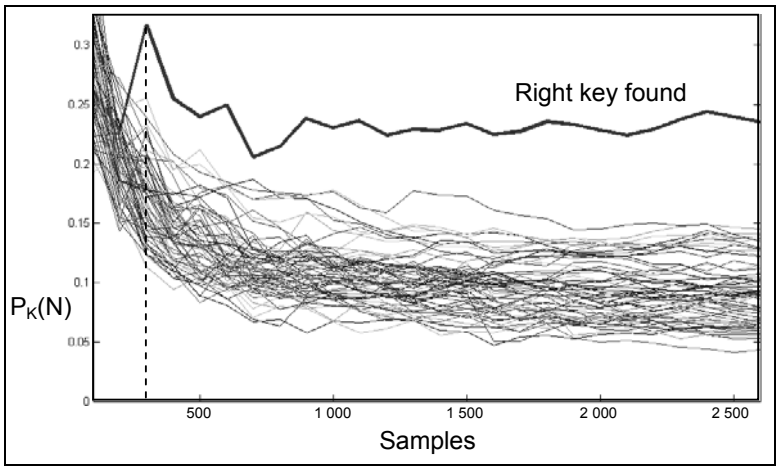

Figure IV.22 : Analyse DPA_H pour la cellule modifiée en fonction de l'activité des cellules.

La représentation temporelle de DPA_H (voir Annexe B) révèle la clé pour t_H=*3,9 ns*. Tout comme pour l'implantation précédente c'est l'activité des portes qui permet à DPA_H d'aboutir.

- **Contre-mesures placées en fonction de la consommation dynamique :**

Pour cette implantation, DPA_H est la première à trouvée la bonne clé pour N_C=*1 200* échantillons. DPA_K ne retrouve toujours pas la bonne clé pour 1 750 échantillons (voir Annexe B). Le facteur d'amélioration relative (F_C) est égal à 2,4.

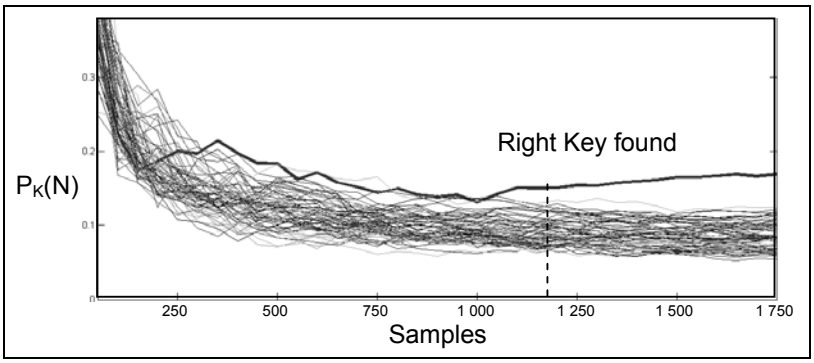

Figure IV.23 : Analyse DPA_H pour la cellule modifiée en fonction de la consommation dynamique.

La corrélation qui permet de déterminer la bonne clé apparaît pour t_H=*3,5 ns* et est due à l'activité des portes logiques de la fonction F (voir Annexe B).

- **Contre-mesures placées en fonction de la puissance moyenne :**

Pour cette implantation, DPA_K (Figure IV.24) parvient à trouver la bonne clé avant DPA_H (voir Annexe B) pour N_C=*1 200*. L'amélioration relative F_C est égale à 2,4.

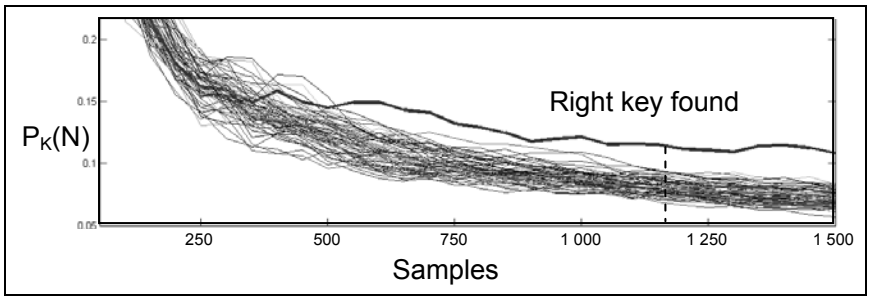

Figure IV.24 : Analyse DPA_K pour la cellule modifiée en fonction de la puissance moyenne.

L'étude temporelle des coefficients de corrélation fait apparaître t_K pour $t=7,5$ *ns* qui correspond au deuxième front montant du signal d'horloge. Cela signifie que le succès de l'analyse est la conséquence de la consommation des registres (voir Annexe B).

- **Contre-mesures placées en fonction de la consommation lors de glitch :**

La *netlist* qui prend en compte l'apparition des *glitch* n'améliore pas la sécurité du DES. En effet, l'analyse DPA_K révèle la clé pour $N_C=500$, soit pour le même nombre d'échantillons que lors d'une analyse de l'implantation initiale. L'analyse DPAH reproduite en Annexe n'apporte rien.

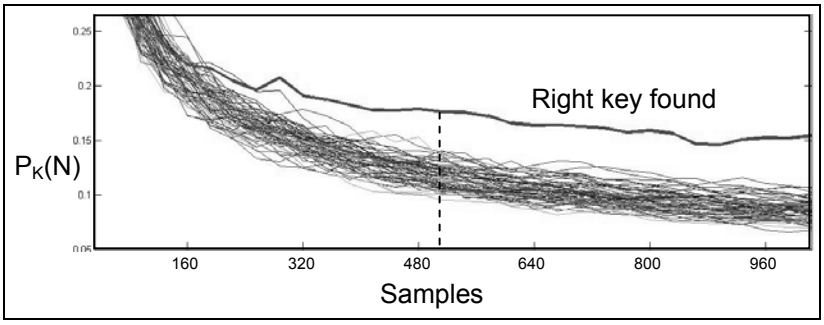

Figure IV.25 : Analyse DPA_K sur l'implantation comprenant des contre-mesures placées en fonction des glitch.

Les évènements t_{K1} et t_{K2} occurrent respectivement pour $t=4,7$ *ns* et $t=7,5$ *ns* (voir Annexe B). Les raisons sont les mêmes que pour l'implantation initiale, à savoir l'activité des registres.

IV.2.3.4. Synthèse

L'exploration du placement des contre-mesures que nous venons de réaliser démontre que le placement basé sur l'étude de l'architecture offre la meilleure résistance à la *DPA*. La Figure IV.26 regroupe les paramètres N_C et F_C pour chaque implantation. Modérons la valeur $F_C=2$ pour la solution basée sur l'étude de l'architecture car l'écart absolue entre la première et la seconde hypothèse est infime. Il y a fort à parier qu'un tel placement nécessiterait beaucoup plus de mesures dans des conditions réelles.

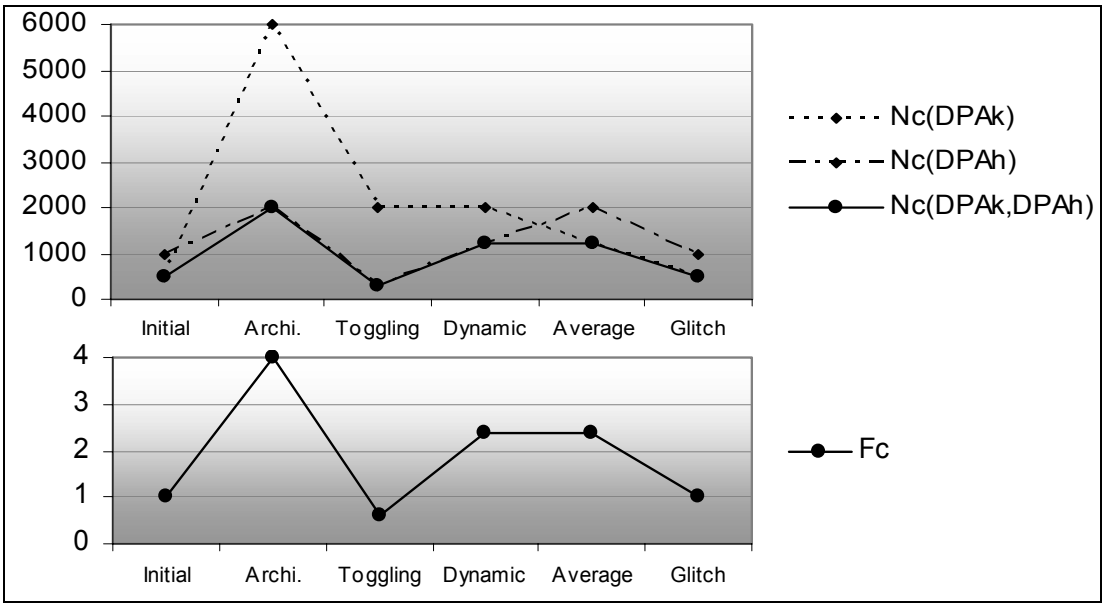

Figure IV.26 : Comparaison des différents placements.

Nous regroupons sur la Figure IV.27 l'apparition temporelle des corrélations qui permettent de casser chaque implantation.

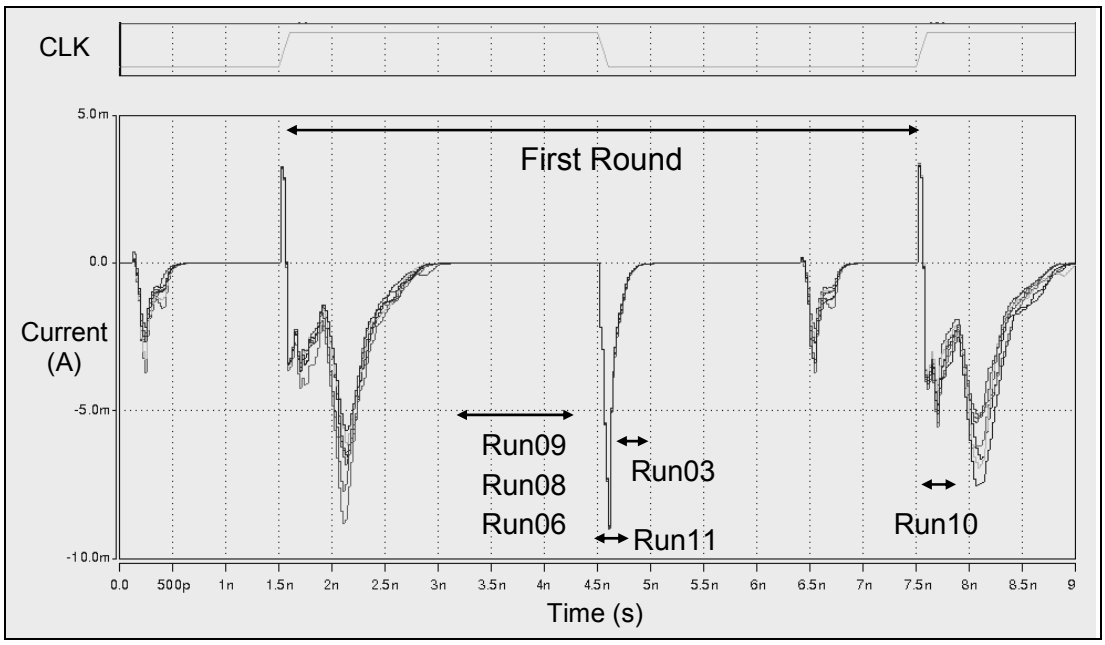

Figure IV.27 : Apparition des évènements pour les différents placements.

L'évaluation réalisée dans cette partie a été menée sur un modèle transistor d'une partie de la fonction DES. Ce protocole présente l'avantage d'être très fin, rapide et de déterminer avec précision l'apparition des évènements qui présentent une corrélation forte entre les données traitées et la consommation $I_{VDD}(t)$.

En revanche, il est relativement long à réaliser. Chaque simulation électrique que nous avons réalisée dure environ 4 minutes. Le calculateur que nous utilisons a obtenu un score de 1,56 pour le critère SPECint_base2006 et 1.52 pour

SPECfp_base2006 [SPE07]. Cela représente tout de même 100 heures de calcul pour obtenir 1 500 simulations. Si l'évaluateur possède plusieurs machines, il peut paralléliser ces simulations. Notons que l'analyse *DPA* ne demande pas un temps de traitement trop élevé.

De plus, comme nous ne simulons qu'une partie de la fonction DES, soit une infime partie d'un circuit final, nous nous affranchissons de la consommation des autres portes présentes dans le circuit Ce qui signifie que nous ne pouvons déduire avec certitude la valeur du paramètre N_C du circuit final. Ce paramètre est essentiel pour un industriel qui souhaite anticiper le niveau de sécurité d'un circuit avant même qu'il ne soit réalisé.

Ainsi, afin d'être plus réaliste et maintenant que nous connaissons le placement le plus efficace contre les analyses *DPA*, nous allons passer à un niveau d'abstraction plus élevé pour évaluer ces contre-mesures : le niveau circuit. Ce niveau ne contient pas l'intégralité du circuit car la simulation demanderait beaucoup trop de ressources. En revanche, elle contient toutes les cellules logiques du circuit intégré, ce qui représente une bonne approche du circuit réel. De plus, nous réaliserons des simulations logiques et non plus électriques afin de limiter le temps de traitement et les ressources nécessaires.

IV.3. Evaluation des contre-mesures au niveau circuit

L'exploration du placement des contre-mesures réalisée dans la partie précédente nous a permis de déterminer le placement le plus efficace pour déjouer la *DPA*. Dans cette partie, nous allons évaluer l'impact de ce placement en simulant toutes les parties logiques d'un circuit intégré, ceci afin de nous rapprocher au maximum du fonctionnement réel du composant et des contre-mesures. Pour ce faire, nous allons raisonner comme précédemment. Nous allons réaliser un nombre N important de simulations afin d'obtenir un nombre élevé d'échantillons $I_N(t)$ que nous analyserons ensuite avec DPA_K et DPA_H.

Cette fois-ci nous utilisons un simulateur logique, moins précis mais plus rapide qu'un simulateur électrique. Ce choix est dicté par le fait que la simulation électrique de toutes les parties logiques d'un système serait beaucoup trop longue et demanderait beaucoup de ressources mémoires. Le flot que nous utilisons dans cette phase, basé sur les travaux de [MER06], (voir Figure IV.28) est appliqué à deux *netlists* distinctes :

- l'initiale, correspondant au circuit intégré sans contre-mesures. Dans ce cas là, la *netlist* est directement fournie au simulateur logique.
- la modifiée dont certaines portes correspondant à la *SBOX0* ont été remplacées par des *path jitters*. La modification de *netlist* est réalisée par notre outil détaillé dans la Figure IV.12. La *netlist* modifiée est alors vérifiée à l'aide d'un outil de vérification formelle. L'outil Formality [SYN] distribué par Synopsys permet de réaliser cette étape.

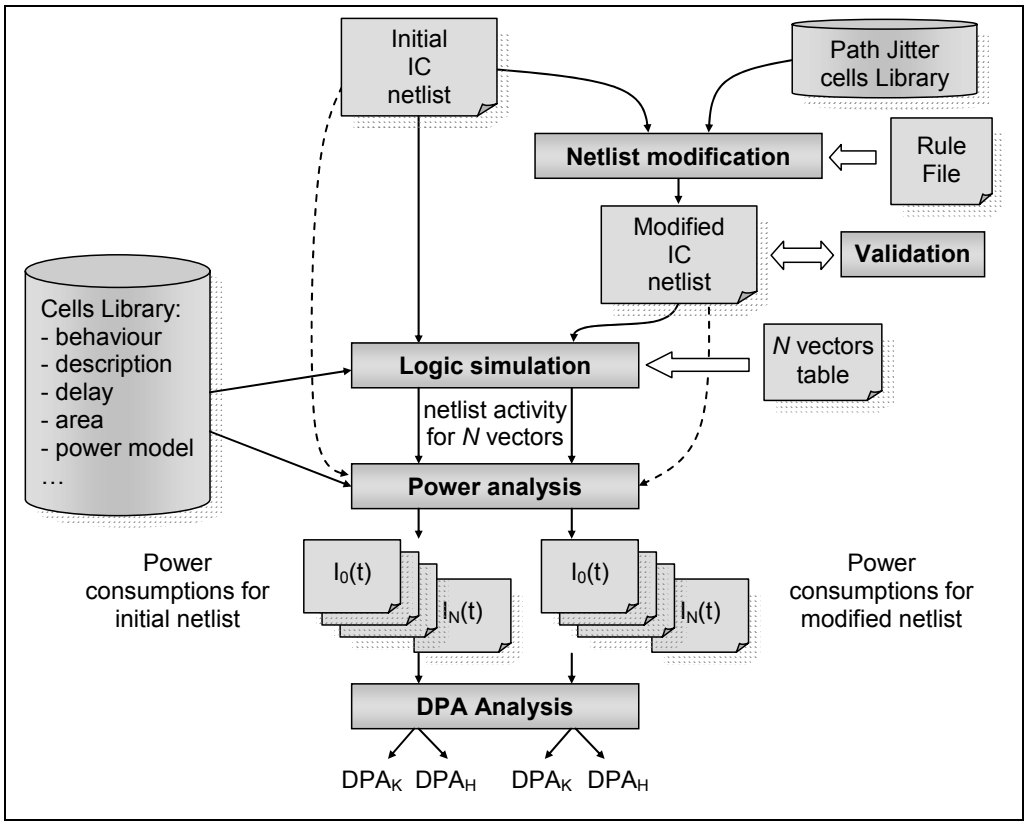

Figure IV.28 : Obtention des courbes de consommation.

- **Simulation logique :**

Les simulations logiques sont réalisées indépendamment sur les deux *netlists* avec exactement les mêmes vecteurs afin de pouvoir comparer correctement l'impact de nos modifications.

Cette étape génère un fichier *VCD* contenant l'activité du circuit pour chaque simulation, soit *N* fichiers pour chaque netlist.

- **Analyse de la puissance :**

Cette étape est équivalente à celle présentée précédemment (voir § IV.2.1). Elle génère un fichier de consommation globale du circuit pour chaque simulation. Au final, nous obtenons un ensemble de traces $I_N(t)$ pour l'implantation sans contre-mesures et un ensemble pour l'implantation avec contre-mesures.

- **Analyse DPA :**

La phase d'analyse *DPA* est identique à celle présentée dans la Figure IV.16. Elle permet de générer pour chaque ensemble de consommation deux matrices de coefficients de corrélation, une pour DPA_K et une pour DPA_H.

IV.3.1. Etude sur une fenêtre temporelle limitée

Il nous est possible d'analyser notre circuit sur une fenêtre temporelle égale à un chiffrement DES complet. Cependant, nous allons limiter la fenêtre temporelle pour concorder avec les simulations électriques réalisées précédemment (voir § IV.2.3). Cette étape va nous permettre d'étudier l'influence de l'activité des portes logiques du reste du circuit.

Ainsi, nous limitons la fenêtre temporelle au cycle d'horloge pendant lequel la première ronde est exécutée, soit *R1* (voir Figure IV.29). Le cycle suivant n'est pas interprété. En effet, dans notre précédent modèle, le cycle suivant coïncide à un nouveau calcul de la première ronde. En fait, dans l'implantation réelle, le cycle suivant correspond à une opération de préparation de la sous-clé de chiffrement pour la prochaine ronde.

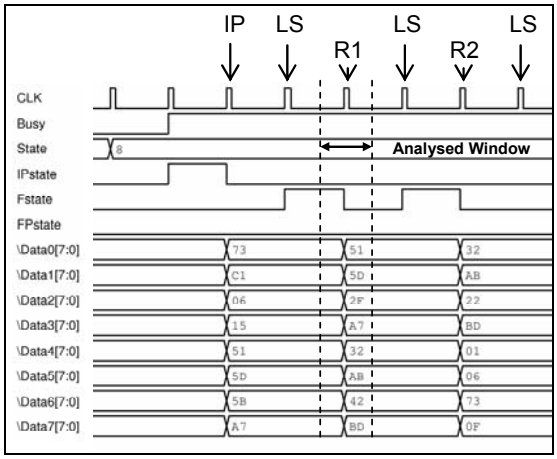

Figure IV.29 : Limitation de la fenêtre temporelle analysée.

La Figure IV.30 permet de visualiser les profils de consommation. Nous observons deux pics de courant, le premier lors du front montant de l'horloge, le second lors du front descendant. Ceci est caractéristique des circuits synchrones. Ces pics sont aussi importants car il s'agit de simulations sans éléments capacitifs et résistifs parasites. C'est-à-dire que les retards des nœuds ainsi que les charges et décharges des capacités parasites induites par le routage ne sont pas simulés. Seuls les retards de traversé des portes logiques interviennent. Nous pouvons constater que lorsque la logique combinatoire est activée, les consommations $I_N(t)$ diffèrent.

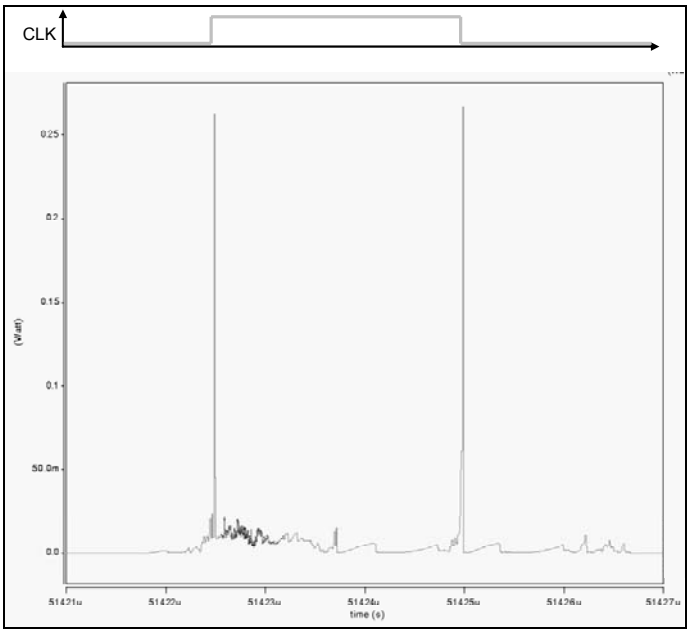

Figure IV.30 : Superposition des courbes de consommation pour plusieurs chiffrement DES.

IV.3.1.1. Evaluation de l'implantation initiale

Etudions l'évolution des coefficients de corrélation en fonction du nombre d'échantillons. La Figure IV.31 fait apparaître que l'analyse DPA_K réussie pour N_C=*1 000*. Il y a un facteur 2 entre cette valeur et celle obtenue par simulation électrique. Ceci est du au fait que notre modèle comporte maintenant beaucoup plus de portes qui perturbent l'acquisition. Cela signifie que nous devons analyser plus d'échantillons afin d'extraire un signal utile du bruit présent.

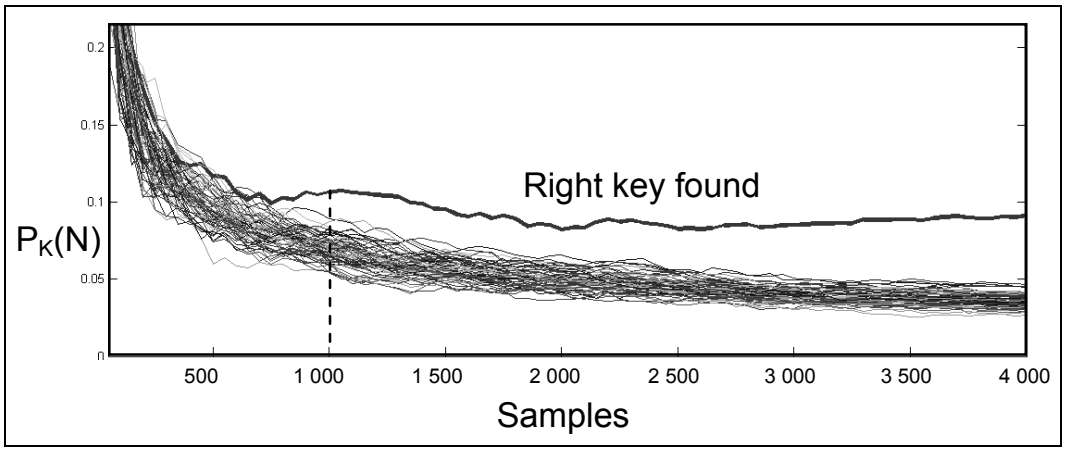

Figure IV.31 : Analyse DPA_K de l'implantation initiale pour une fenêtre temporelle limitée.

L'analyse DPA_H, représentée dans la Figure IV.32, quand à elle nécessite beaucoup plus d'échantillons pour révéler la bonne clé : environ 20 000. Ce qui donne un facteur 20 par rapport au modèle précédent. De plus, plusieurs fausses clés sont

révélées ; une pour *N* compris entre 3 000 et 6 000, et une autre pour *N* compris entre 6 000 et 12 000. [GUI04] traite plus en détail de ce phénomène de fausses corrélations.

Pour cette implantation N_C est égal à 1 000, toutes analyses confondues.

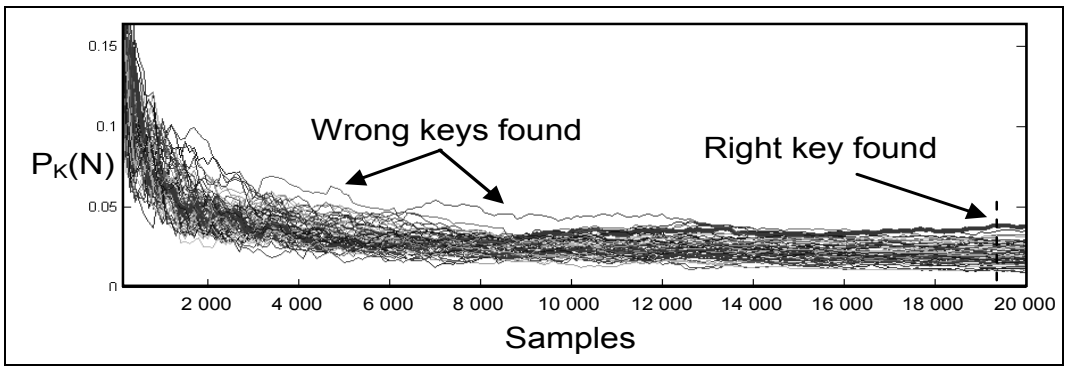

Figure IV.32 : Analyse DPA_H de l'implantation initiale pour une fenêtre temporelle limitée.

La partie gauche de la Figure IV.33 représente temporellement les coefficients de corrélation DPA_K pour $N=4\ 000$. L'évènement t_K apparaît lors du front descendant du signal d'horloge. Cela coïncide parfaitement avec les résultats obtenus lors des simulations électriques et conforte notre modèle précédent.

La partie droite de cette même figure correspond à DPA_H pour $N=19\ 500$. La corrélation apparaît lorsque la logique combinatoire est activée et est tout à fait cohérent avec les résultats obtenus avec le précédent modèle électrique.

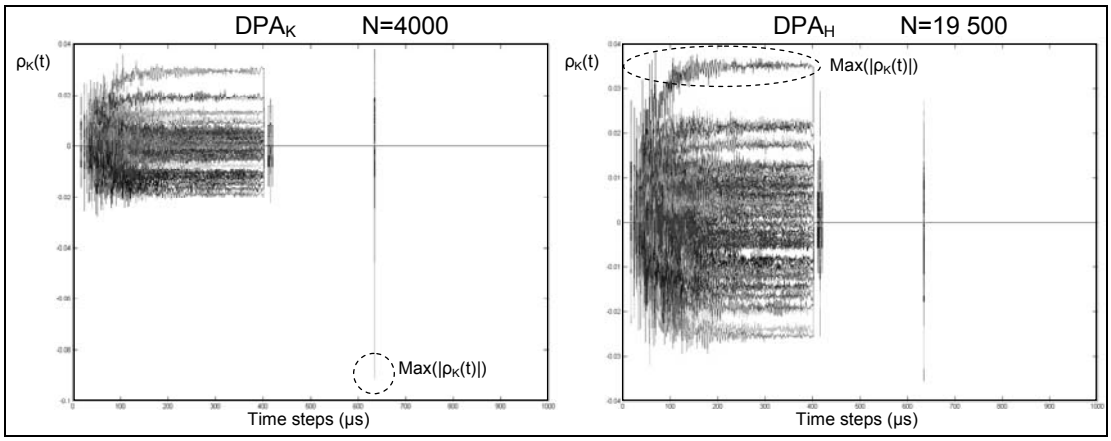

Figure IV.33 : Représentation temporelle DPA_K et DPA_H pour la netlist initiale.

IV.3.1.2. Evaluation de l'implantation avec contre-mesures

Dans cette partie, nous présentons les résultats de l'évaluation de la *netlist* renforcée en fonction de l'étude de l'architecture.

L'analyse DPA_K (Figure IV.34) donne $N_C=5\ 500$ mais ne possède pas une certitude très élevée pour la bonne clé car elle disparaît et ne dépasse que très ponctuellement un écart de 10 % avec la seconde meilleure hypothèse. La bonne clé

apparaît pour *N* compris entre 5 200 et 6 000, pour *N=6 500* et *N=8 600*. L'écart relatif maximum est de 15 % soit un écart absolu de seulement 0,007 entre les valeurs des coefficients de corrélation. Nous pouvons distinguer cet écart car nous sommes en simulation. Très certainement que dans le cas de l'analyse d'un circuit réel, cet infime écart ne pourrait pas être mis en évidence.

Notons que l'analyse DPA_H n'aboutit toujours pas pour *N=12 000*. En considérant ces écarts aussi infimes soient-ils, nous obtenons un facteur d'amélioration par rapport au même circuit sans contre-mesures, $F_C=5$.

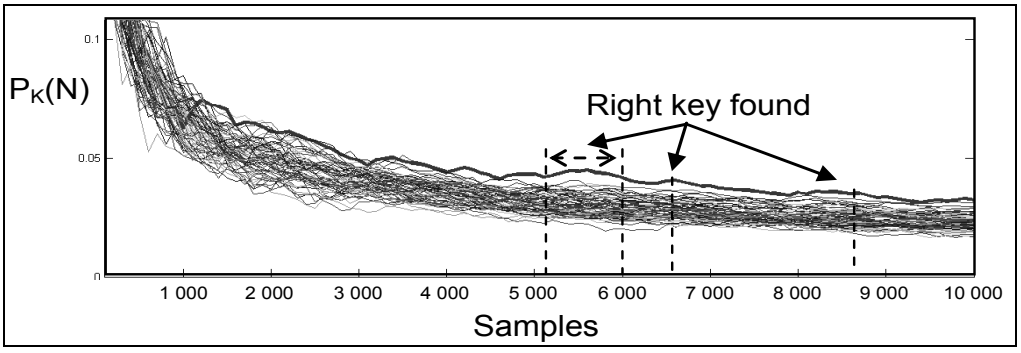

Figure IV.34 : Analyse DPA_K au niveau produit sur l'implantation avec contre-mesures placées intuitivement.

Dans la représentation temporelle de DPA_K (Figure IV.35), l'apparition de t_K est causée par l'activité de la logique combinatoire de la ronde.

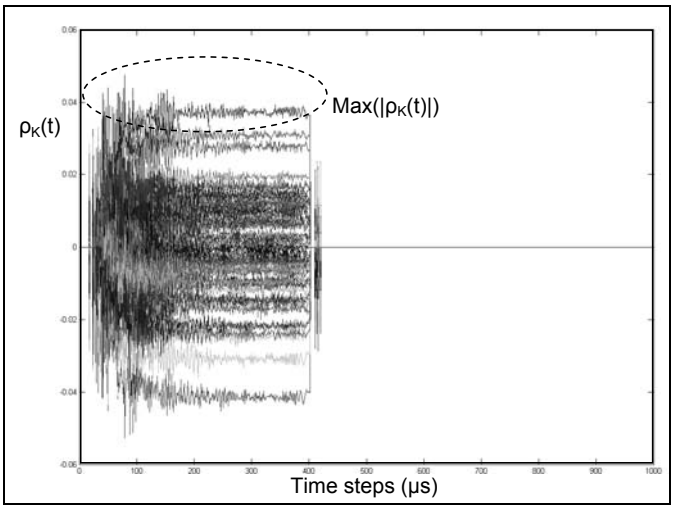

Figure IV.35 : Représentation temporelle de DPA_K pour la netlist avec contre-mesures.

Cette première étude en prenant en compte toutes les portes logiques du circuit donne une amélioration satisfaisante de la sécurité avec une amélioration par 5 du nombre d'échantillons nécessaires pour retrouver la bonne clé. De plus, la certitude que cette clé soit correcte est très faible.

IV.3.2. Etude sur une fenêtre temporelle étendue

Elargissons maintenant notre fenêtre temporelle d'étude afin d'observer si d'autres évènements n'étaient pas pris en considération jusqu'à maintenant et pour nous rapprocher au plus des conditions réelles.

Rappelons que l'outil qui réalise l'analyse *DPA* cible les données en sortie de la première ronde. Pour cette raison, il est inutile d'élargir la fenêtre temporelle à tout un chiffrement DES. En effet, cela exigerait une capacité de stockage trop importante.

Nous nous contentons d'analyser l'activité comprise entre le démarrage du chiffrement et la fin de l'exécution de la quatrième ronde. Nous enregistrons plusieurs rondes car il est possible que des corrélations apparaissent après la première ronde. Le chronogramme de la Figure IV.36 représente la fenêtre temporelle considérée.

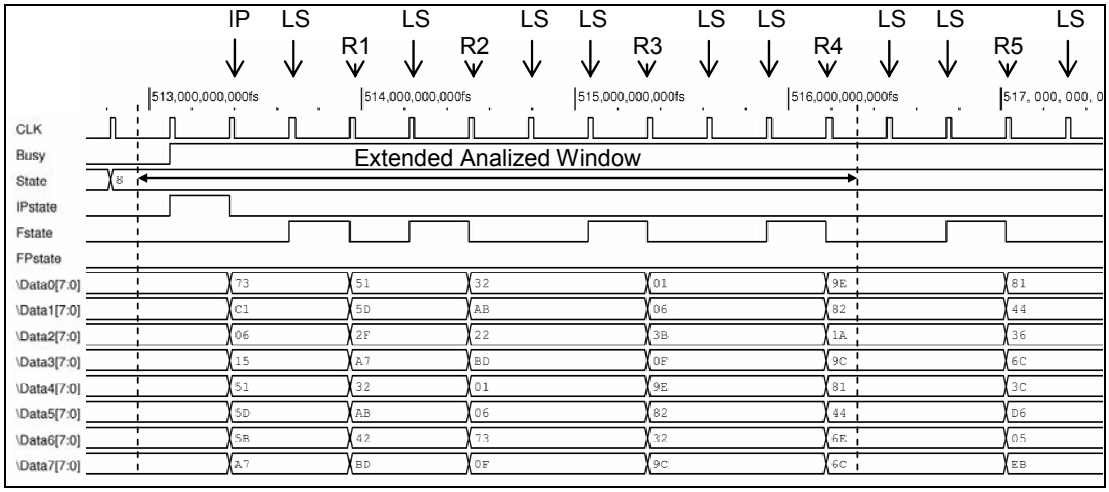

Figure IV.36 : Extension de la fenêtre temporelle analysée.

Le flot d'évaluation est identique à celui utilisé précédemment. La seule différence est que l'outil *DPA* traite toute la fenêtre temporelle acquise ; alors qu'auparavant nous procédions à un découpage des courbes de consommation $I_N(t)$.

La Figure IV.37 représente l'analyse DPA_K pour les deux implantations étudiées.

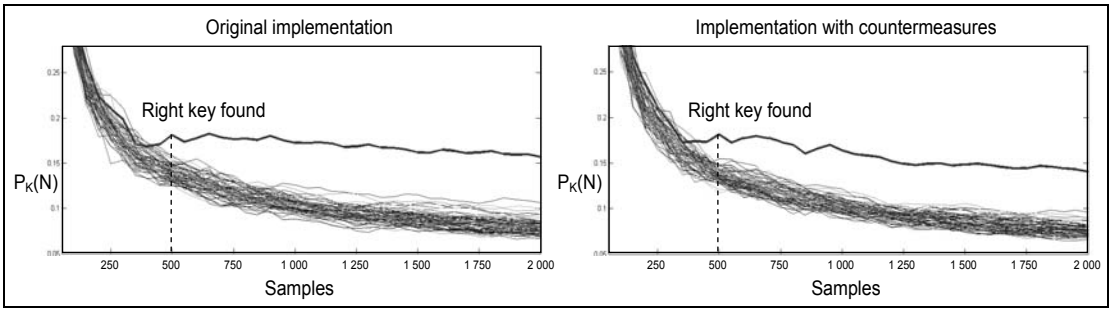

Figure IV.37 : DPA_K pour une fenêtre temporelle élargie sur les deux implantations.

Aussi surprenant que cela puisse paraître, les résultats sont identiques. Notre proposition n'apporte aucune amélioration. Etudions l'occurrence des évènements au

cours du temps afin de comprendre ce résultat. La Figure IV.38 correspond au circuit initial où l'activation de la logique est clairement visible pour chaque front montant de l'horloge de la fenêtre étudiée. Nous pouvons distinguer deux évènements distincts :

- t_{K1} pendant le déroulement de la première ronde,
- t_{K2} pendant l'opération de gestion de clé. Cet évènement est celui qui contribue le plus à la découverte de la bonne clé.

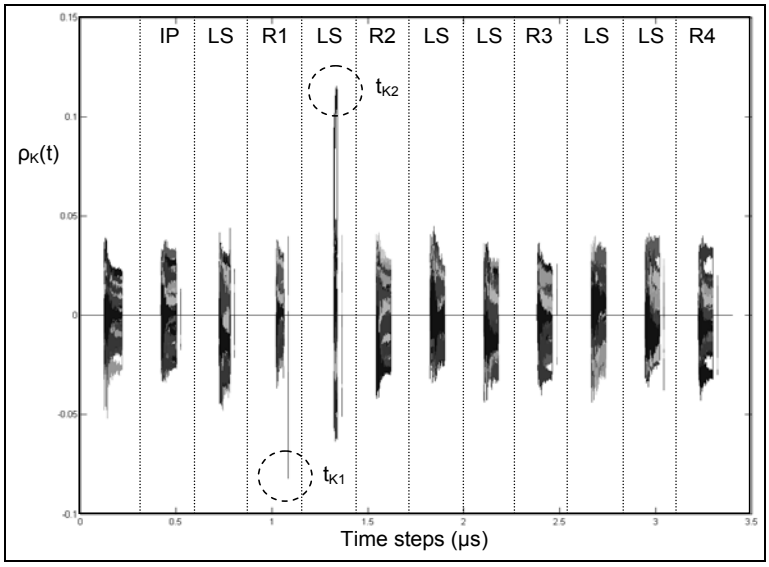

Figure IV.38 : Evolution temporelle des coefficients de corrélation pour DPA_K sur l'implantation initiale.

Observons maintenant la même représentation lorsque les contre-mesures sont instanciées (voir Figure IV.39).

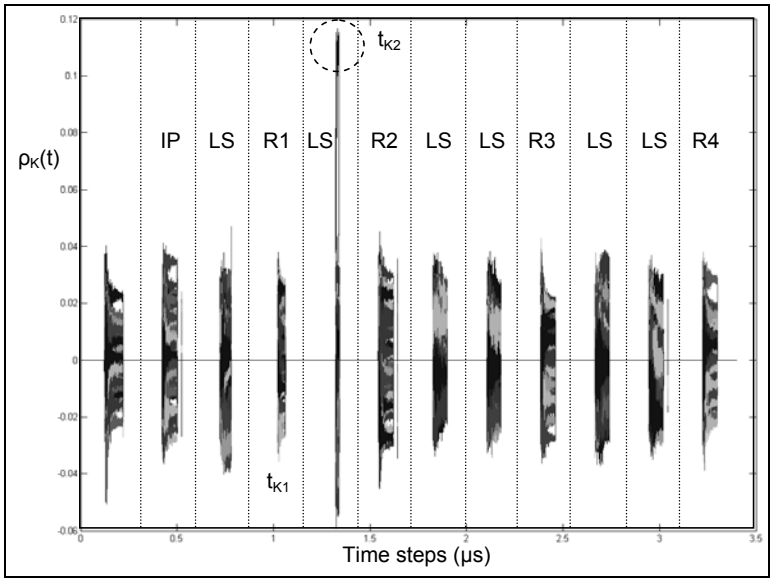

Figure IV.39 : Evolution temporelle des coefficients de corrélation pour DPA_K sur l'implantation avec contre-mesures.

Nous observons que l'évènement t_{K1} a disparu conformément aux attentes.

En revanche, l'évènement t_{K2} qui apparaît pendant la phase de gestion de clé n'est pas modifié. Il provient d'une faille de l'implantation (codage *RTL*) utilisée et que notre contre-mesure n'a que très légèrement modifié. Des manipulations ont permis de vérifier cette hypothèse. Il serait même envisageable de sécuriser le bloc de gestion de clé avec des *path jitters* afin d'achever la sécurisation de l'implantation du bloc *DES* utilisé.

Analysons maintenant la matrice des coefficients de corrélation générée par DPA_H. Pour l'implantation initiale, la bonne clé apparaît dès 5 000 échantillons. Il y a une amélioration significative de la sécurité car nous ne retrouvons toujours pas la clé pour *N=20 000* lorsque les contre-mesures sont présentes.

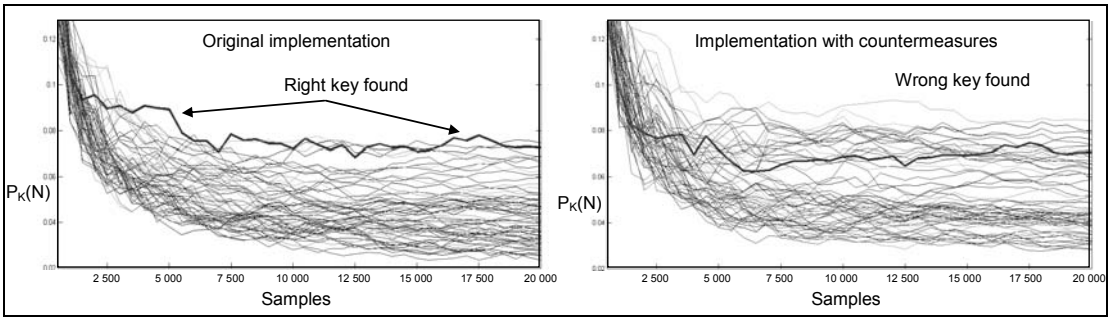

Figure IV.40 : DPA_H pour une fenêtre temporelle élargie sur les deux implantations.

Les représentations temporelles de DPA_H (Figure IV.41) sans et avec contre-mesures montrent que notre proposition a un impact sur la signature de la logique lors de la gestion de clé (*LS*), puisque t_H disparaît au profit d'une mauvaise hypothèse de clé (voir Figure IV.42).

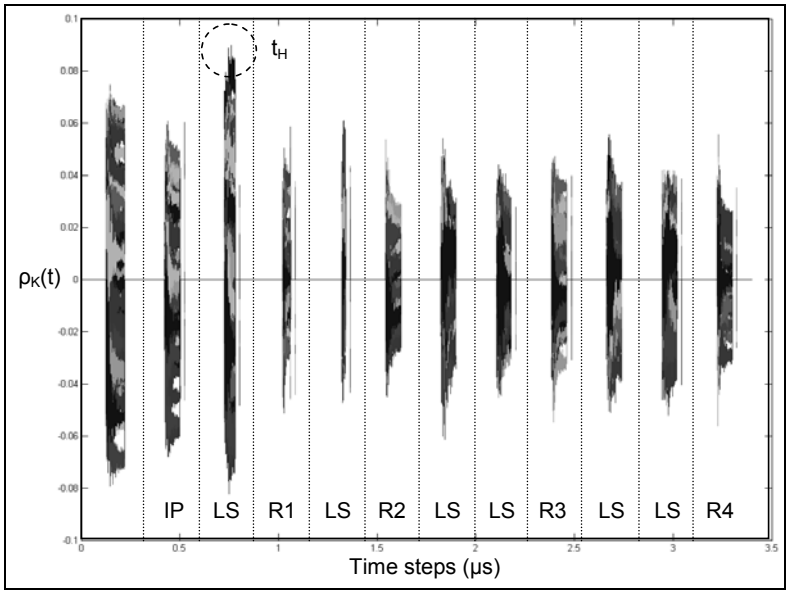

Figure IV.41 : Evolution temporelle des coefficients de corrélation pour DPA_H sur l'implantation initiale (N = 4 500).

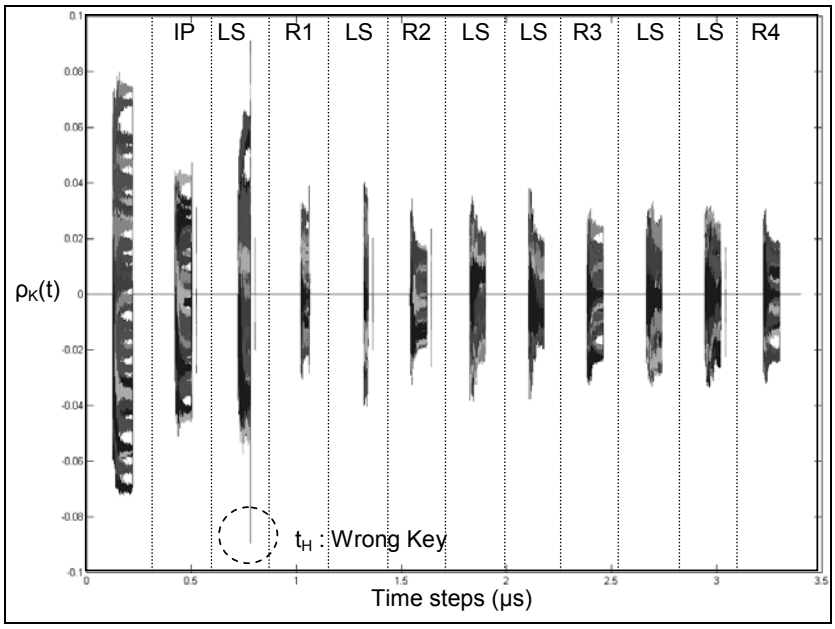

Figure IV.42 : Evolution temporelle des coefficients de corrélation pour DPA_H sur l'implantation avec contre-mesures (N = 12 500).

IV.3.3. Simulation avec éléments parasites

Afin de nous rapprocher toujours plus du fonctionnement réel d'un circuit, nous allons maintenant réaliser les simulations logiques en incluant les capacités parasites ainsi que la résistivité des nœuds d'interconnexion induites par le placement et le routage.

Tout comme dans la partie précédente, nous allons comparer l'évolution des coefficients de corrélation pour l'implantation initiale et celle avec les contre-mesures placées en fonction de l'architecture. La Figure IV.43 fait apparaître que le nombre d'échantillon est multiplié par trois entre les analyses DPA_K sans et avec contre-mesures.

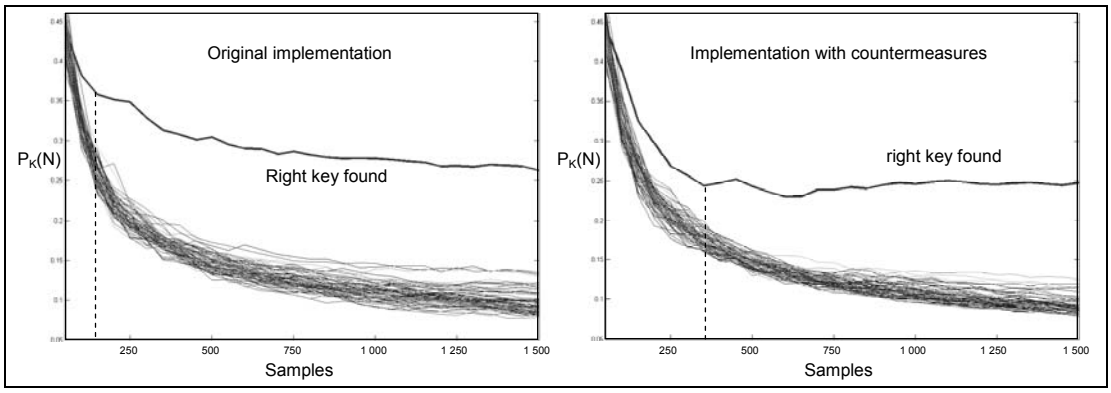

Figure IV.43 : DPA_K pour une fenêtre temporelle élargie avec prise en compte des parasites, sur les deux implantations.

Les résultats sont similaires à ceux obtenus dans la partie précédente (voir § IV.3.2) en ce qui concerne l'implantation initiale. Notons que les *path jitters* sont plus efficaces lorsque les éléments parasites sont inclus dans les simulations.

L'analyse temporelle fait apparaître plusieurs évènements t_K pour l'implantation sans contre-mesures lors de *R1*, lors de *LS* qui suit et lors de *R2*. Ceci est représente en Annexe B.

Pour l'implantation avec contre-mesures, c'est l'étape *LS* qui suit *R1* qui révèle la bonne clé (voir Annexe B).

La figure suivante (Figure IV.44) permet de comparer les résultats de l'analyse DPA_H. Dans les deux cas étudiés, la bonne clé n'est pas révélée. Mieux encore, dans l'implantation initiale aucune clé ne se distingue alors que lorsque les contre-mesures sont présentes, de mauvaises clés sont révélées, dès 500 échantillons.

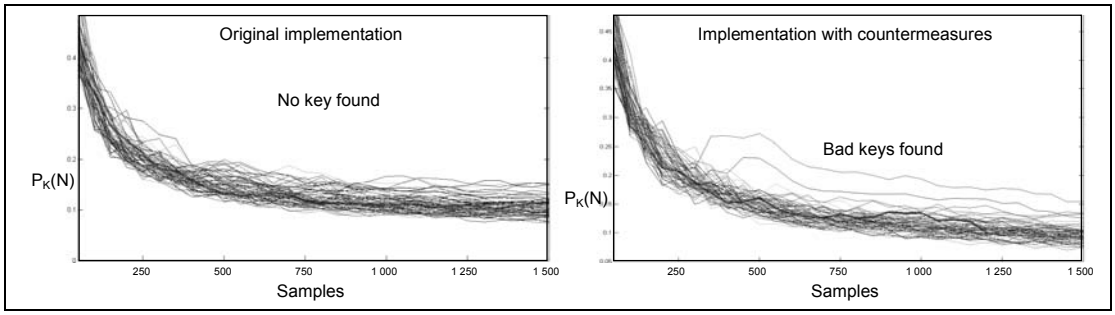

Figure IV.44 : DPA_H pour une fenêtre temporelle élargie avec prise en compte des parasites, sur les deux implantations.

L'analyse temporelle ne fait pas apparaître la clé. Pour l'implantation sans contre-mesure, les plus fortes valeurs des coefficients de corrélation apparaissent pendant le déroulement de *LS* précédent *R1*, sans révéler la bonne clé (voir Annexe B).

De la même manière, l'implantation avec contre-mesures fait apparaître les plus fortes corrélations lors de la phase *LS* précédent *R1* (voir Annexe B).

Au final, lors de la simulation avec prise en compte des éléments parasites, nous obtenons un facteur F_C égal à 3. Ce facteur peut paraître faible mais présente tout de même un intérêt certains. En effet, la puissance de ces *path jitters* est qu'ils

sont compatibles avec d'autres types de contre-mesures. Pour information, l'algorithme DES que nous avons employé tout au long de cette étude ne comporte aucune sécurité particulière contre l'analyse *DPA*. Cela laisse présager que le facteur F_C peut encore être élevé si nous utilisons d'autres contre-mesures placées selon différents niveaux d'abstraction.

IV.4. Synthèse

La première partie de ce chapitre a été consacrée à la description, selon différentes approches, d'une nouvelle contre-mesure élaborée pour déjouer l'attaque *DPA*. Elle s'inscrit dans la famille des contre-mesures qui se situent au niveau des portes logiques, plus particulièrement basées sur le masquage de la consommation. L'avantage de cette proposition est qu'elle ne nécessite pas le développement de cellules supplémentaires. De plus, elles peuvent être réparties dans un circuit dans les parties sensibles seulement afin de sécuriser ce dernier.

C'est pour cette raison que dans une seconde partie, nous nous sommes intéressés au placement judicieux de seulement quelques unes de ces cellules. Nous avons pris comme exemple l'algorithme DES. Et plutôt que de sécuriser toutes la macro cellule réalisant la fonction DES, nous avons mis en place un protocole de placement des contre-mesures selon certains critères. Différents placement ont été évalués à l'aide de simulations électriques. Il en résulte que la stratégie basée sur l'étude de l'architecture est celle qui présente les meilleurs résultats face à une attaque *DPA*.

Enfin, dans une dernière partie, nous avons développé un autre protocole toujours capable d'évaluer le placement des contre-mesures réalisé dans la partie précédente. La différence réside dans le fait que dans cette partie, il s'agit de simulations logiques, non plus électriques, et qu'elles sont appliquées à l'intégralité des portes logiques d'un circuit réel et non plus une infime partie. Dans ce sens, cette partie peut aisément s'intégrer dans le flot industriel de conception *Front End* d'un circuit intégré. Cette évaluation du placement des contre-mesures a démontré que la version renforcée du circuit intégré est plus robuste que la version initiale. De plus, il reste une marge d'amélioration du niveau de sécurité car il est encore possible d'employer des contre-mesures situées à d'autres niveaux d'abstraction.

Enfin, une limitation de notre modèle est qu'il ne permet pas de chiffrer avec exactitude la résistance de la solution que nous proposons si celle-ci était utilisée sur un circuit réel. Il nous indique seulement un facteur d'amélioration relative face une analyse *DPA*. Dans notre cas, nous pouvons tout au plus supposer que la valeur du facteur $F_C=3$ augmentera sur le circuit réel du fait du bruit additionnel engendré par les appareils de mesure et l'environnement. En effet, nous pouvons considérer notre protocole de simulation comme le pire cas favorisant une analyse *DPA*. L'exemple le plus parlant se situe dans la partie IV.3.1.2 dans laquelle nous avons mis en évidence qu'il est fort probable qu'une analyse réelle ne révèle pas la bonne clé pour un nombre d'échantillon raisonnable.

CONCLUSIONS ET PERSPECTIVES

Dans ces travaux de thèse, nous avons traité des problématiques que peuvent rencontrer les concepteurs de circuits intégrés sécurisés afin de rendre leurs produits toujours plus sûrs, et ainsi plus compétitifs. Nous avons notamment abordé les attaques par clonage et retro-ingénierie ainsi que les analyses en courant auxquels ces derniers sont confrontés.

Dans la première moitié de ce manuscrit, nous avons présenté une solution capable d'améliorer la sécurité flexible afin de contrer les attaques par clonage et retro-ingénierie. Cette solution que nous avons proposée consiste à développer des cellules reconfigurables élémentaires ou *Look-Up-Table*. Différentes méthodes d'assemblage et concepts d'intégration ont été débattus. Cette solution offre l'avantage d'être simple à mettre en place et présente de nombreux compromis afin de pouvoir l'adapter à différentes architectures de circuits sécurisés. Une architecture a été proposée alliant flexibilité et faible encombrement pour un haut niveau de sécurité. Afin de ne pas diminuer le niveau de sécurité de cette solution, nous avons aussi abordé le protocole de reconfiguration. En effet, cette phase est critique de par les nombreuses failles sécuritaires qui peuvent apparaître. Le protocole que nous avons proposé est rapide et robuste à la fois. Enfin, il peut facilement être implanté dans tous les circuits sécurisés. Cette première partie de nos travaux ouvre la voie à l'utilisation de la logique reconfigurable dans les circuits sécurisés afin d'augmenter la flexibilité fonctionnelle de ces derniers.

Une autre investigation possible afin d'améliorer la sécurité flexible, consisterait à concevoir entièrement des blocs *Look-Up-Table*, et non plus à utiliser un assemblage de cellules logiques standards. Cela permettrait d'obtenir des cellules reconfigurables optimisées en surface et en consommation. Il serait aussi envisageable de concevoir ces cellules en prenant comme objectif la résistance aux canaux cachés. En effet, prenons le cas des FPGA qui sont par nature basés sur des éléments reconfigurables pour lesquels les attaques DPA sont amplifiées par le fait que le routage des éléments de base n'est pas correctement équilibré. Ainsi, nous pourrions concevoir ces cellules afin de palier à cette faiblesse.

Dans la deuxième moitié de ce manuscrit nous avons abordé les attaques par analyse des canaux cachés et plus particulièrement la DPA qui reste une large menace pour les circuits sécurisés. Au travers d'un modèle réaliste, nous avons mis en évidence les phénomènes exploités par la DPA. Suite à cela, nous avons présenté un

modèle de contre-mesures basé sur la modification dynamique de la topologie du circuit intégré. L'avantage de cette proposition est qu'elle est simple à mettre en œuvre puisqu'elle ne nécessite pas le développement de cellules spécifiques. De plus, elle peut se décliner selon plusieurs variantes et être placée de diverses façons dans les circuits sécurisés. Afin d'évaluer cette proposition et de déterminer le placement le plus judicieux, nous avons été contraints de mettre en place différentes méthodologies. Ainsi, le premier flot que nous avons mis en place, basé sur des simulations électriques a permis de déterminer qu'une étude préalable du circuit est nécessaire pour offrir le niveau de sécurité le plus élevé possible. Enfin, nous avons mis en place un flot, basé sur des simulations logiques, qui permet d'évaluer la résistance d'une architecture face à la DPA avant même la création du circuit intégré. Il permet d'évaluer la résistance à différents niveaux d'abstraction et à différentes étapes du flot de conception d'un produit. Ce qui permet d'anticiper la robustesse d'une solution retenue relativement tôt dans la phase de conception. Ceci nous a permis de mettre en évidence que la solution que nous proposons améliore le niveau de sécurité du circuit que nous avons renforcé.

Une des limitations de notre méthodologie est qu'elle ne permet pas de donner avec exactitude la résistance de la solution que nous avons proposée. En effet, celle-ci reste cloisonnée à des simulations. Idéalement, afin d'affiner cette méthodologie, il serait intéressant de réaliser, dans un premier temps, l'analyse réelle d'un produit existant et, dans un second temps, de réaliser l'analyse réelle sur le même produit possédant des contre-mesures. Ceci nous permettrait de chiffrer précisément l'amélioration qu'apportent les contre-mesures que nous proposons et d'ajuster notre modèle. Il serait aussi intéressant d'évaluer la complémentarité de cette contre-mesure avec d'autres contre-mesures existantes.

BIBLIOGRAPHIE

[ABR91] D. G. Abraham, G. M. Dolan, G. P. Double, J. V. Stevens, "Transaction Security System", *IBM Systems Journal*, Vol. 30, Issue 2, Special issue on Cryptology, pp. 206-229, IBM Corp., 1991, ISSN 0018-88670.

[ACI07] O. Aciiçmez, J.-P. Seifert and C. K. Koç, "Predicting secret keys via branch prediction", Topics in *Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007*, to appear.

[ACT02] "Design Security with Actel FPGAs", Actel White Paper, August 2002, http://www.actel.com/

[AIG07] M. Aigner and E. Oswald, "Power Analysis Tutorial", Seminar paper, Institute for Applied Information Processing and Communications, Graz University of Technology, http://www.iaik.tugraz.at/aboutus/people/oswald/papers/dpa_tutorial.pdf

[AKK01] M.-L. Akkar and C. Giraud, "An Implementation of DES and AES, Secure against Some Attacks", in proceedings of *3rd International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001*, Paris, France, May 14-16, 2001, LNCS 2162, pp. 309-318, Springer-Verlag, 2001, ISBN 3-540-42521-7.

[AND93] R. Anderson, "Why Cryptosystems Fail", in proceedings of the 1st *ACM Conference on Computer and Communications Security*, Fairfax, Virginia, United States, pp. 215-227, 1993, ACM Press New York, NY, USA, ISBN 0-89791-629-8.

[AUV00] D. Auvergne, J. M. Daga and M. Rezzoug, "Signal Transition Time Effect on CMOS Delay Evaluation", *IEEE Transactions on Circuits and Systems, Part I : Fundamental Theory and*

Applications, Vol. 47, n° 9, pp. 1362-1369, 2000.

[AXC07] Actel Corp., "Axcelerator Device Family", *Datasheet*, http://www.actel.com/

[BAR06] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall and C. Whelan, Discretix Technologies Ltd, "The Sorcerer's Apprentice Guide to Fault Attacks", in proceedings of the *IEEE Publication*, Feb. 2006 Vol. 94, Issue 2, pp. 370-382, ISSN 0018-9219.

[BER05] D. J. Bernstein, "Cache-timing attacks on AES". *Technical Report*, April 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[BIE00] I. Biehl, B. Meyer and V. Muller, "Differential Fault Analysis on Elliptic Curves Cryptosystems", in proceedings of *Crypto 2000, Advances in Cryptology*, pp. 131-146, Springer-Verlag, 2000.

[BIH90] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", in proceedings of the *10th Annual international Cryptology Conference on Advances in Cryptology*, Aug. 11-15, 1990, LNCS 537, pp 2-21, ISBN 3-540-54508-5.

[BIH97] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", in proceedings of 17th Annual *International Cryptology Conference, Advances in Cryptology, CRYPTO'97*, Santa Barbara, California, USA, August 1997, LNCS 1294, pp. 513, Springer Berlin / Heidelberg, ISSN 0302-9743

[BON97] D. Bonneh, R. A. Demillo and R. J. Lipton "On the Importance of Checking Cryptographic Protocols for Faults", in proceedings of *EUROCRYPT '97, Advances in Cryptology*, LNCS, pp. 37-51, Springer-Verlag, 1997.

[BOR03] M. Borgatti, L. Calì, G. De Sandre, B. Forêt, D. Iezzi, F. Lertora, G. Muzzi, M. Pasotti, M. Poles and P.L. Rolandi, "A Reconfigurable Signal Processing IC with embedded FPGA and Multi-Port Flash Memory" in proceedings of the *40th Conference on Design Automation, DAC'03*, 2003, Anaheim, CA, USA, pp. 691 - 695, ACM Press, New York, NY, USA, ISBN 1-58113-688-9.

[BOS04] L. Bossuet, G. Gogniat and W. Burleson "Dynamically Configurable Security for SRAM FPGA Bitstreams", in proceedings of *11th Reconfigurable Architectures Workshop, RAW 2004*, 26-27 Avril 2004, Santa Fé, USA.

[BOU05] G. F. Bouesse, « Contribution à la Conception de Circuits Intégrés sécurisés : l'Alternative Asynchrone », *Mémoire de thèse*, TIMA, Grenoble, France, 2005.

[BRO92] S. Brown, R. Francis, J. Rose and Z. Vranesic, "Field-Programmable Gate Arrays". Norwell, MA: Kluwer, 1992.

[BUC04] M. Bucci, L. Giancane, R. Luzzi and A. Trifiletti, "Three-Phase Dual-Rail Pre-Charge Logic", in proceedings of 6th *International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2004.*

[BUR02] D. Bursky, "Fast-Turns Alternatives to FPGAs", in Electronic Design, Online ID#2098, April 15, 2002, http://electronicdesign.com/Articles/ArticleID/2098/2098.html

[CAD] Cadence Inc., http://www.cadence.com/

[CAR86] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo and S. L. Sze, "A user programmable reconfigurable gate array," in proceedings of *Custom Integrated Circuits Conference*, May 1986, pp. 233–235.

[CAT02] A. Cataldo, "Silicon as Task Configurations", in EETimes, April 25, 2002, http://www.eetimes.com/story/OEG20020423S0058

[CC] Common Critera, http://www.commoncriteriaportal.org/

[CHA99] S. Chari, C. S. Jutla, J. R. Rao and P. Rohatgi, "Towards Sound Approaches to Counteract Power Analysis Attacks", in proceedings of *Advances in Cryptology, CRYPTO'99*, LNCS 1666, pp. 398-412, Springer Verlag, 1999.

[CHE89] W.T. Cheng and V. D. Agrawal, "An Economical Scan Design for Sequential Logic Test Generation", in proceedings of the *Fault Tolerant Computing Symposium*, pp. 28-35, 1989

[CHO99a] P. Chow, S. Ong Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The Design of an SRAM-Based Field-Programmable Gate Array - Part I: Architecture", *IEEE Trans. VLSI Syst.*, Vol. 7, n°. 2, June 1999.

[CHO99b] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The design of an SRAM-based field-programmable gate array - Part II: Circuit design and layout," *IEEE Trans. VLSI Syst.*, Vol. 7, n°. 3, September 1999.

[CHU87] Kan M. Chu, DCVSL, IEEE Journal of Solid-State Circuits, vol. SC-22, n° 4, August 1987.

[CLA00] C. Clavier, J.-S. Coron and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures", in proceedings of *International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000*, LNCS 1965, Springer-Verlag, 2000.

[COO03] L. H. Cooke, "Use of Configurable Cores in Platform based SoCs", eASIC White Paper, http://www.easic.com/

[DAE99] J. Daemen and V. Rijmen, "Resistance against Implementation Attacks: A Comparative Study of the AES Proposals", in

proceedings of the *2nd Advanced Encryption Standard Condidate Conference, AES'99*, March 1999.

[DET78] J. Dethloff, "Identification System Safeguarded against Misuse", December 1976, U.S. Patent ref. 4,105,156.

[DHE98] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestre J. J. Quisquater and J. L. Willems, "A Practical Implementation of the Timing Attack", *UCL Crypto Group Technical Report CG-1998/1*, Université Catholique de Louvain, Belgium, 1998.

[DIF76a] W. Diffie and M. E. Hellman, "Multi-user Cryptographic Techniques", in proceedings of *AFIPS National Conference*, pp. 109-112, 1976.

[DIF76b] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, n°6, pp. 644-654, November 1976.

[EAS] eASIC Corp., http://www.easic.com/

[EFF98] Electronic Frontier Foundation, *Cracking DES - Secrets of Encryption Research, Wiretap Politics & Chip Design*, O'Reilly, 1998, ISBN 1565925203

[ELG85] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", in proceedings of *Advances In Cryptology – CRYPTO'84*, pp. 10-18, Springer Verlag, Berlin, 1985.

[FEI73] H. Feistel, "Cryptography and Computer Privacy", *Scientific American*, May 1973, pp. 15-23.

[FIS06] V. Fischer, "Flexible Security and Its Technology Limits", in proceedings of the *2nd International Workshop on Reconfigurable Communication-centric Systems-on-Chip, ReCoSoC 2006*, Montpellier, France, July 2006, Univ. Montpellier II, 2006, ISBN 2-9517461-2-1.

[FOU03] J.J.A. Fournier, S. Moore, H. Li; R.D. Mullins and G.S. Taylor, "Security Evaluation of Asynchronous Circuits" in proceedings of *5th International Workshop Cryptographic Hardware and Embedded Systems, CHES 2003*, Cologne, Germany, September 8-10, 2003, LNCS 2779, Springer-Verlag, 2003.

[GAN01] K. Gandolfi, C. Mourtel and F. Olivier, "Electromagnetic Analysis: Concrete Results", in proceedings of *3rd International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001*, Paris, France, May 14-16, 2001, LNCS 2162, pp. 251-261, Springer-Verlag, 2001, ISBN 3-540-42521-7.

[GEM01] GEMPLUS, "Simple Power Analysis", presented at GEMPLUS, Workshop on Cryptography and Security, March 13, 2001,

http://www.cs.uku.fi/kurssit/ads/09_20-_20SPA.pdf

[GOU99] L. Goubin and J. Patarin, "DES and Differential Power Analysis--- The Duplication Method", in proceedings of 1st *International Workshop on Cryptographic Hardware and Embedded Systems, CHES'99*, LNCS 1717, pp.158-172, London, United Kingdom, Springer-Verlag, 1999, ISBN:3-540-66646-X

[GUI04] S. Guilley, P. Hoogvorst and R. Pacalet, "Differential Power Analysis Model and Some Results", in proceedings of the *sixth Smart Card Research and Advanced Application IFIP Conference, CARDIS 04*, pp. 127-142, August, 2004, Toulouse, France.

[HAN99] H. Handschuh, P. Paillier and J. Stern, "Probing Attacks on Tamper-Resistant Devices", in proceedings of 1st *International Workshop on Cryptographic Hardware and Embedded Systems, CHES'99*, LNCS 1717, pp.303-315, London, United Kingdom, Springer-Verlag, 1999, ISBN:3-540-66646-X

[HEL80] M.E. Hellman, "A Cryptanalytic Time-Memory Trade-off", *IEEE Transactions on Information Theory*, vol.6 (1980), pp. 401-406.

[HAR07] Altera Corp., "Hardcopy Series Handbook", Datasheet, http://www.altera.com/

[ISH03] Y. Ishai, A. Sahai and D. Wagner, "Private Circuits: Securing Hardware against Probing Attacks", in proceedings of *Advances In Cryptology – CRYPTO'03*, LNCS 2729, pp. 463-481, Springer Verlag, 2003.

[ISO] International Organization for Standardization, http://www.iso.org/

[ISOTC] ISO Standards Development, http://isotc.iso.org/

[KAH97] D. A. Kahn, *The Codebreakers*, Scribner Book Company Ed., October 1997, ISBN 0684831309.

[KEA01] T. Kean, "Secure Configuration of Field Programmable Gate Arrays", in proceedings of *11th International Conference on Field-Programmable Logic and Applications, FPL 2001*, Belfast, United Kingdom, 2001.

[KER83] A. Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, vol. IX, pp. 5–38, Janvier 1883, pp. 161–191, Février 1883.

[KES00] D. Kessner, "Copy Protection for SRAM based FPGA designs", 2000, http://web.archive.org/web/20031010002149/http://free-ip.com/copyprotection.html

[KOC96] P.C. Kocher, "Timing Analysis on Implementation of Diffie-Hellman, RSA, DSS and Other Systems", in *Crypto'96*, 1996, LNCS 1109, pp. 104-113, Springer-Verlag.

[KOC99] P.C. Kocher, J. Jaffe and B. Jun, “Differential Power Analysis”, in *19th Annual International Cryptology Conference, Advances in Cryptology, CRYPTO'99*, Santa Barbara, California, USA, August 1999. LNCS 1666, Springer, 1999, ISBN 3-540-66347-9.

[KOM99] O. Kömmerling and M. G. Kuhn, “Design Principles for Tamper-Resistant Smartcard Processors“, in proceedings of the *USENIX Workshop on Smartcard Technology, Smartcard '99*, Chicago, Illinois, USA, May 10-11, 1999,USENIX Association, pp. 9-20, ISBN 1-880446-34-0.

[KUH96] M. Kuhn and R. Anderson, “Tamper Resistance – a Cautionary Note”, in *The Second USENIX Workshop on Electronic Commerce*, Oakland, California, USA, November 18-21, 1996, pp. 1-11, ISBN 1-880446-83-9.

[LI08] M.-C Li, H.-H. Tung, C.-C. Lai and R.-B. Lin, “Standard Cell Like Via-Configurable Logic Block for Structured ASICs”, in proceedings of *ISVLSI 2008*, April 7-9, 2008, Montpellier France, to be published.

[LYO97] R. Lyons “Understanding Digital Signal Processing”, Addison-Wesley, 1997.

[M2000] M2000 Inc., “FlexEOS Embedded FPGA Cores”, Datasheet, 2003, http://www.m2000.fr

[MAU01] P. Maurine, « Modélisation et optimisation des Performances de la Logique Statique en Technologie Submicronique Profond », *Thèse de doctorat*, Université Montpellier II, 23 Avril 2001.

[MEN96] A. J. Menezes, *Handbook of Applied Cryptography*, CRC Press, Oct 1996, ISBN 0-8493-8523-7.

[MENa] Menta S.A.S., http://www.menta.fr/

[MENb] Mentor Graphics Corp., http://www.mentor.com/

[MER06] J. Mercier, “Méthodologie d’Analyse en Courant des cartes à Puce”, *Mémoire de thèse*, Université Aix-Marseille I, 2007.

[MES00] T. S. Messerges, “Power Analysis Attack Countermeasures and their Weaknesses”, in *Communications, Electromagnetics, Prpoagation & Signal Processing Workshop, CEPS 2000*, October 12 2000, Illinois, USA, http://www.iccip.csl.uiuc.edu/conf/ceps/2000/messerges.pdf

[MES02] T. S. Messerges, E.A. Dabbish and R.H. Sloan, “Examining Smartcard Security under the Threat of Power Analysis Attacks”, IEEE TC, Vol. 51, Issue 5, pp. 541-552, May 2002.

[MES99] T. S. Messerges, E.A. Dabbish and R.H. Sloan, “Investigation of Power Analysis Attacks on Smartcards”, in *The USENIX Workshop on Smartcard Technology*, Chicago, Illinois, USA, May 10-11,

1999.

[MOO02] S. Moore, R. Anderson, P. Cunningham, R. Mullins and G. Taylor, "Improving Smart Card Security using Self-timed Circuits" in proceedings of the *8th IEEE International Symposium on Asynchronous Circuits and Systems*, ASYNC 2002.

[MOR75] R Moreno, "Data Storage Systems", October 1975, Patent publication number DE2512935.

[MUL07] D. Müller, *Les Codes Secrets Décryptés*, City Editions, Février 2007, ISBN 2-35288-041-6, http://www.apprendre-en-ligne.net/crypto/

[NFC] Near Field Communication Forum, http://www.nfc-forum.org/

[NIK99] S. Nikolaidis and A. Chatzigeorgiou, "Analytical Estimation of Propagation Delay and Short Circuits Power Dissipation in CMOS Gates", *International Journal of Circuit Theory and Applications*, Vol. 27, Issue 4, pp. 375-392, September 30 1999.

[NIS00] National Institute of Standards and Technology, "Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures", Special Publication 800-20, April 2000.

[NIS01] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS 197, November 26, 2001, http://csrc.nist.gov/archive/aes/index.html

[NIS02] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-2, 2002.

[NIS77] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46, 1977.

[NIS98] National Institute of Standards and Technology, "Modes of Operation Validation System (MOVS): Requirements and Procedures", Special Publication 800-17, February 1998.

[NIS99] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, 1999.

[PAA03] C. Paar and T. Wollinger, "How secure are FPGAs in cryptographic applications?" in proceedings of *Field Programmable Logic, FPL 2003*, LNCS, vol. 2778, pp. 91-100, Springer-Verlag, 2003.

[POP05] T. Popp and S. Mangard, "Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints" in proceedings of *7th International Workshop Cryptographic Hardware and Embedded Systems, CHES 2005*, Edinburgh, Scotland, August 29 - September 1, 2005, LNCS , Springer, 2005.

[POP06] T. Popp and S. Mangard, "Implementation Aspects of the DPA-

Resistant Logic Style MDPL" in proceedings of *International Symposium on Circuits and Systems, ISCAS 2006*, Island of Kos, Greece, May 21-24, 2006, IEEE Computer Society, 2006.

[PRO07] Actel Corp., "ProASIC3 Flash Family FPGAs Datasheet", Datasheet, http://www.actel.com

[QUI00] J.-J. Quisquater and D. Samyde, "A New Tool for non-intrusive Analysis of Smartcards based on Electro-Magnetic Emissions, the SEMA and DEMA Methods", in proceedings of *EUROCRYPT 2000*.

[RAB79] M. O. Rabin, "Digital Signatures ans Public-Key Functions as Intractable as Factorization", *Technical Report MIT/LCS/TR-212*, MIT Laboratory for Computer Science, January 1979.

[RAK00] P. Rakers, L. Connell, T. Collins and D. Russell, "Secure Contactless Smartcard ASIC with DPA Protection", in proceedings of *Custom Integrated Circuits Conference, CICC 2000*, June 21-24, Orlando, FL, USA, pp. 239-242, ISBN 0-7803-5809-0.

[RAN03] W. Rankl and W. Effing, *Smartcard Handbook – 3rd Ed*, Wiley, 2003, ISBN 0-470-85668-8.

[RAZ06] A. Razafindraïbe, « Analyse et amélioration de la logique double rail pour la conception de circuits sécurisés », *Mémoire de Thèse*, LIRMM/UMR 5506, Montpellier, France, 2006.

[RIV92] R. L. Rivest, "The MD5 Message Digest Algorithm", RFC 1321, April 1992

[RSA78] R. L. Rivest, A. Shamir and L. Adleman, "A method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21, n°2, pp. 120-126, February 1978.

[SCH00] W. Schindler, "A Timing Attack against RSA with the Chinese Remainder Theorem", in proceedings of *International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000*, LNCS 1965, pp. 110-125, Springer-Verlag, 2000.

[SCH97] B. Schneier, *Cryptographie Appliquée - 2eme Ed*, Vuibert, 2001, ISBN 2-7117-8676-5.

[SHA00] A. Shamir, "Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies", in proceedings of *International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000*, LNCS 1965, pp. 71-77, Springer-Verlag, 2000.

[SHA04] A. Shamir and E. Tromer, "Acoustic Cryptanalysis", http://www.wisdom.weizmann.ac.il/~tromer/acoustic/, 2004.

[SHA49] C. E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, vol. 28, n°4, pp. 656-715, October 1949.

[SKO02] S. P. Skorobogatov, R. J. Anderson, “Optical Fault Induction Attacks”, in proceedings of *4th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2002*, Redwood Shores, CA, USA, August 13-15, LNCS 2523, pp. 31-48, Springer Berlin / Heidelberg, ISSN 0302-9743.

[SPA07] Xilinx Corp., “Spartan-3A FPGA Family Datasheet”, Datasheet, http://www.xilinx.com/

[SPE07] Standard Performance Evaluation Corporation, www.spec.org/

[SPI] Spice Circuit Simulator, http://bwrc.eecs.berkeley.edu/ Classes/IcBook/SPICE/

[STA04] F.-X. Standaert, S.B. Örs, J.-J. Quisquater and B. Preenel, “Power Analysis Attacks against FPGA Implementations of the DES”, in *Field Programmable Logic 2004, FPL 2004*, Antwerp, Belgium, August 19 - September 01, 2004, LNCS 3203, pp 84-94, Springer-Verlag, 2004, ISBN 978-3-540-22989-6.

[STM05] STMicroelectronics, “GreenFIELD STW21000 - Reconfigurable Micro Controller”, *Technical Article*, 2005, http://www.st.com/

[STR07] Altera Corp., “Stratix III Device Handbook”, Datasheet, http://www.altera.com/

[SUZ04] D. Suzuki, M. Saeki and T. Ichikawa, “Random Switching Logic: A Countermeasure against DPA based on Transition Probability”, *Cryptology ePrint Archive* (http://eprint.iacr.org/), Report 2004/346, 2004.

[SYN] Synopsys Inc., http://www.synopsys.com/

[TIR02] 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 proceedings of the *29th European Solid-State Circuits Conference, ESSCIRC 2002.*

[TIR03] K. Tiri and I. Verbauwhede, “Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology” in proceedings of *5th International Workshop Cryptographic Hardware and Embedded Systems, CHES 2003*, Cologne, Germany, September 8-10, 2003, LNCS 2779, Springer-Verlag, 2003.

[TIR05] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont and I. Verbauwhede, “Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment” in proceedings of *7th International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2005*, Edinburgh, UK, August 29 - September 1, 2005, LNCS, Springer, 2005.

[TPM07] Trusting Computing Group, https://www.trustedcomputinggroup.org/specs/TPM/

[VIA] ViASIC Inc., http://www.viasic.com/

[VIR07] Xilinx Corp., "Virtex-5 Multi-Platform FPGA", Datasheet, http://www.xilinx.com/

[WIL02] R. Wilson, "The Constantly Shifting promise of Reconfigurability", in EETimes, September 12, 2002, http://www.eetimes.com/story/OEG20020911S0066

[WIL03] R. Wilson, "Design Reuse Expands Across Industry", in EETimes, March 27, 2003, www.eetimes.com/story/OEG20030324S0039

[CRO07] G. Crow, "Advanced Security Schemes for Spartan-3A/3AN/3A DSP FPGAs", Xilinx Corp. White Paper, ref 267, http://www.xilinx.com/support/documentation/white_papers/wp267.pdf

CONTRIBUTIONS

[VAL05a] N. Valette, L. Torres, G. Sassatelli, F. Bancel et N. Bérard, "Intégration d'un coeur reconfigurable dans un composant sécurisé", dans les actes des *8ème Journées Nationales du Réseaux Doctoral en Microélectronique*, *JNRDM 2005*, Mai 2005, Paris, France.

[VAL05b] N. Valette, L. Torres, G. Sassatelli, F. Bancel and N. Bérard, "Integration of Reconfigurable Logic on Secure Circuits", in proceedings of *1st Reconfigurable Communication-centric SOCs Conference*, *ReCoSoC 2005*, June 2005, Montpellier, France.

[VAL06a] N. Valette, L. Torres, G. Sassatelli and F. Bancel, "Securing Embedded Programmable Gate Arrays in Secure Circuits", in proceedings of *13th Reconfigurable Architecture Workshop*, *RAW 2006*, April 2006, Rhodes, Greece, ISBN 1-4244-0054-6.

[VAL06b] N. Valette, L. Torres, G. Sassatelli and F. Bancel, "How to Secure Embedded Programmable Gate Arrays ?", in proceedings of *2nd Reconfigurable Communication-centric SOCs Conference*, *ReCoSoC 2006*, June 2006, Montpellier, France.

ANNEXES

ANNEXE A :

Tables SBOX de l'algorithme de chiffrement DES.

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tableau A-1 : Tables SBOX 1 à 4 du DES.

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tableau A-2 : SBOX 5 à 8 du DES.

ANNEXE B :

Courbes des coefficients de corrélation relatifs aux analyses DPA. Du chapitre IV.

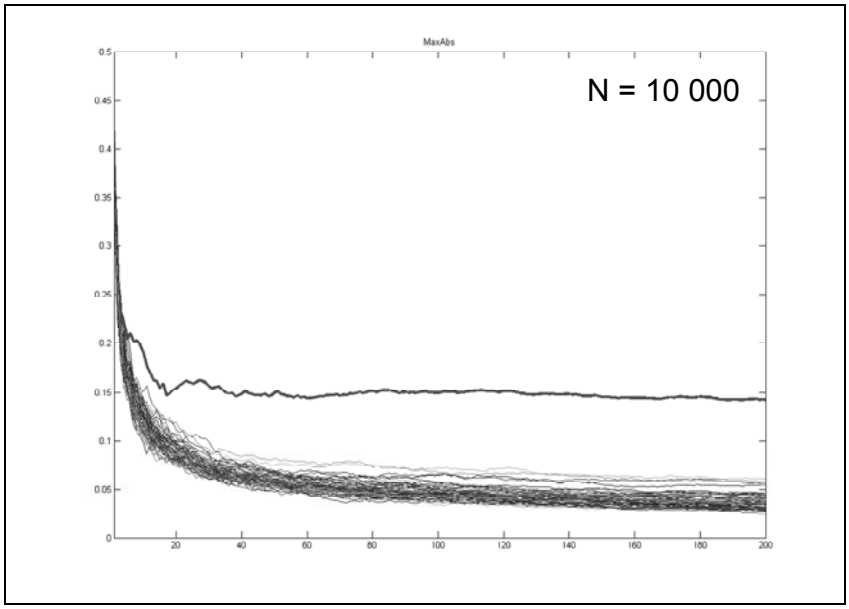

Figure B - 1 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES avec la netlist initiale.

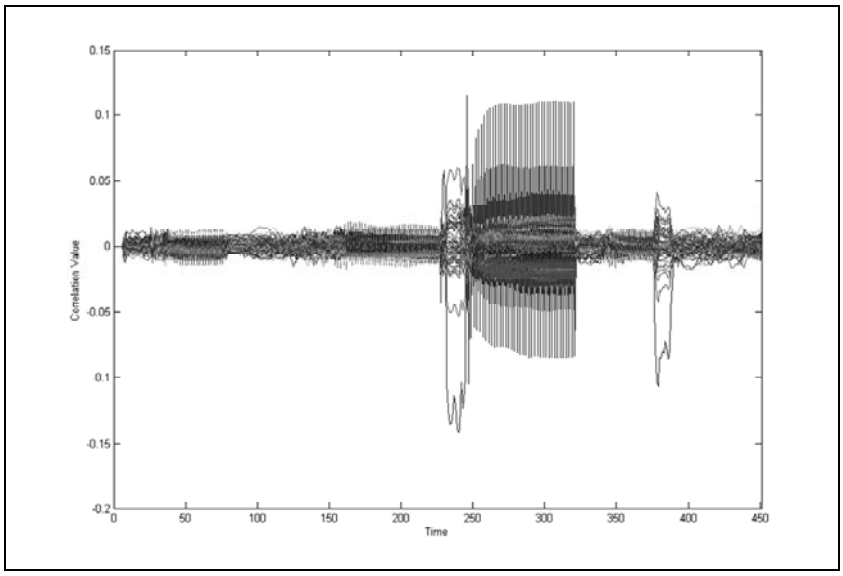

Figure B - 2 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES avec la netlist initiale.

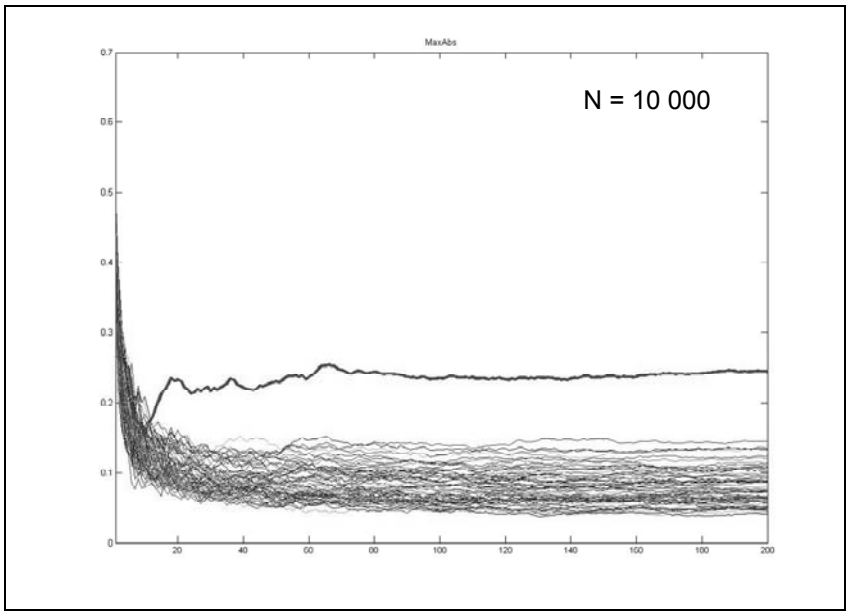

Figure B - 3 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES avec la netlist initiale.

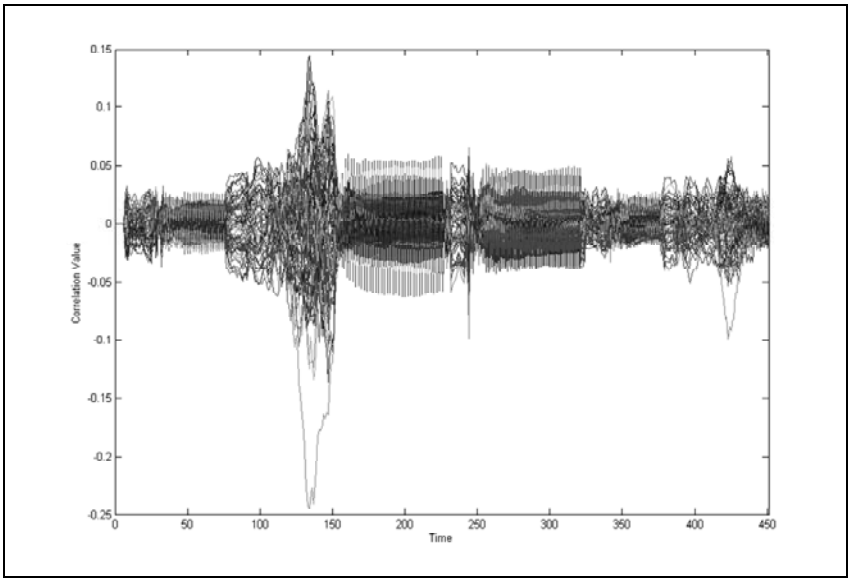

Figure B - 4 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction DES avec la netlist initiale.

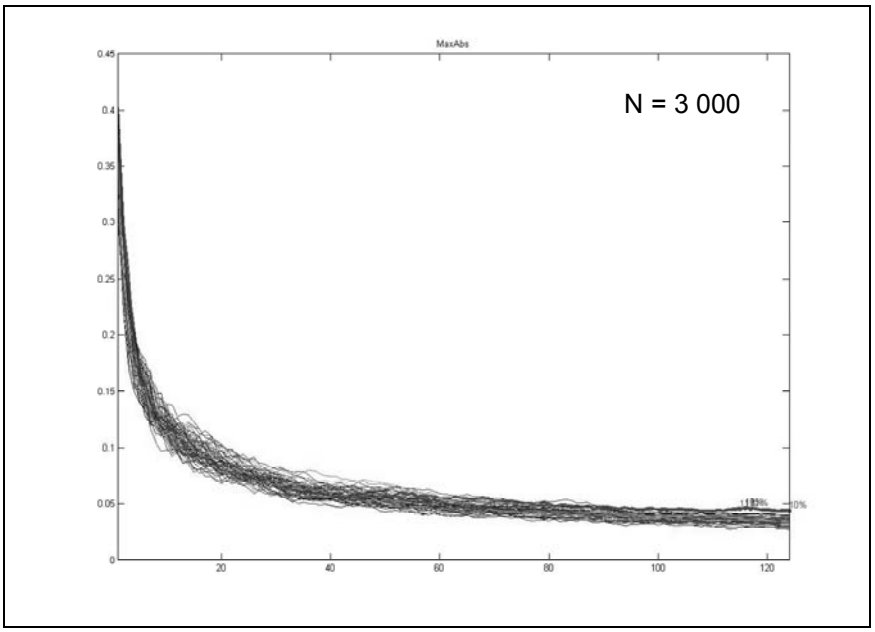

Figure B - 5 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'étude de l'architecture.

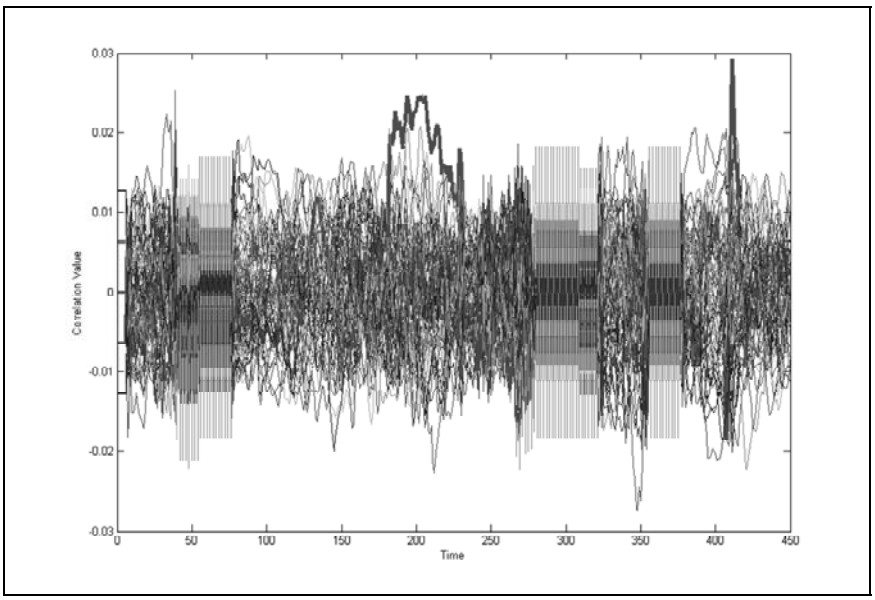

Figure B - 6 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'étude de l'architecture.

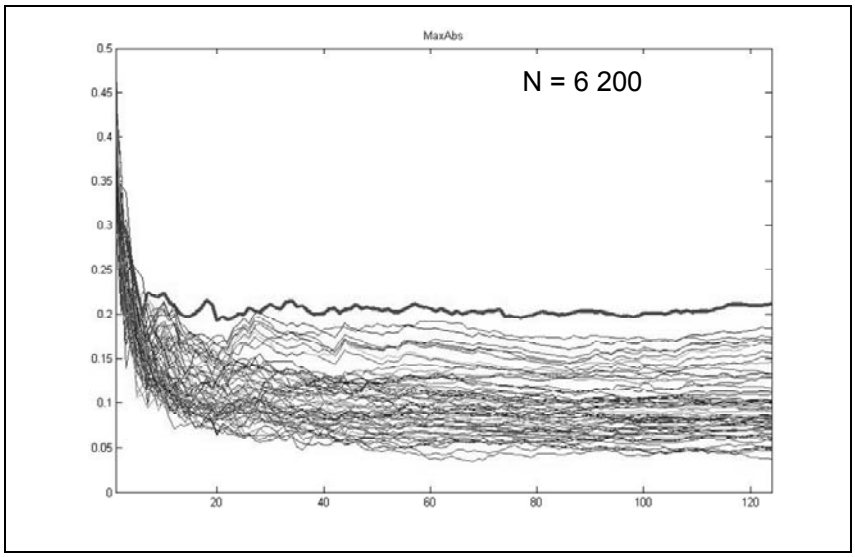

Figure B - 7 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'étude de l'architecture.

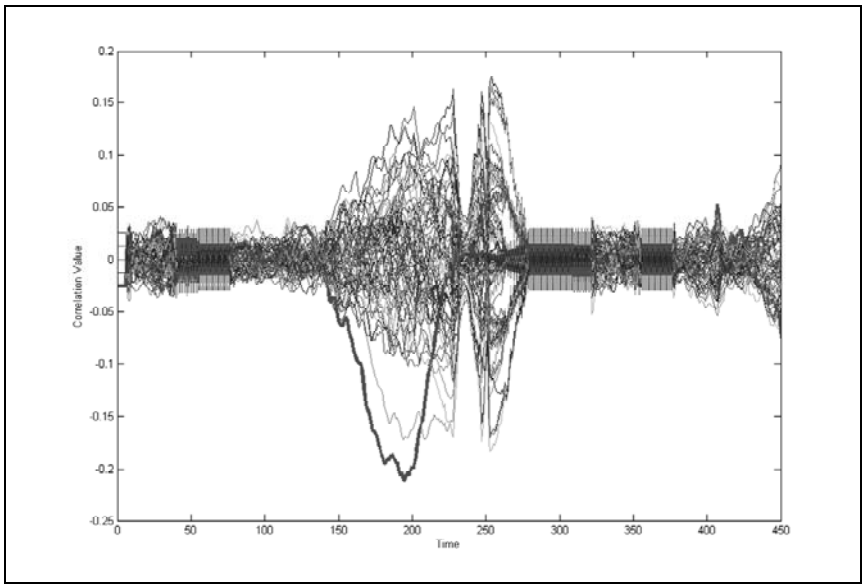

Figure B - 8 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'étude de l'architecture.

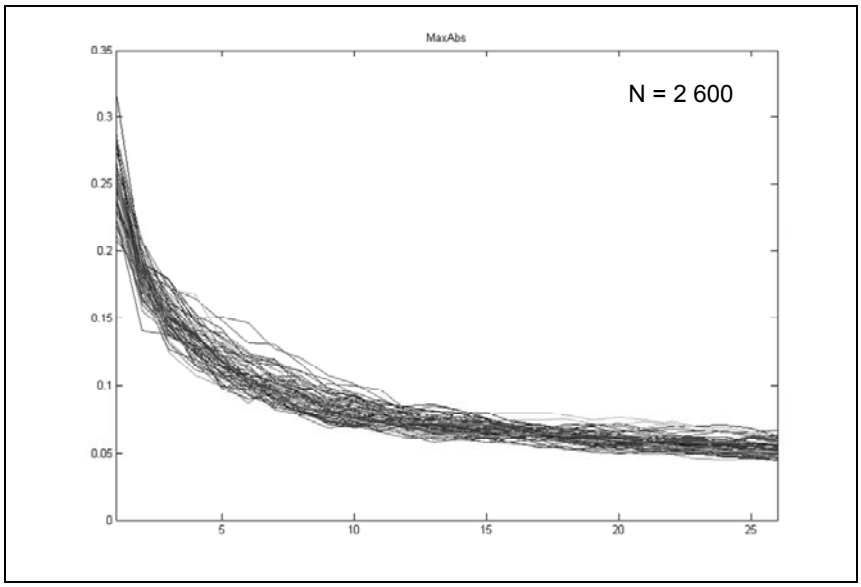

Figure B - 9 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'activité du circuit.

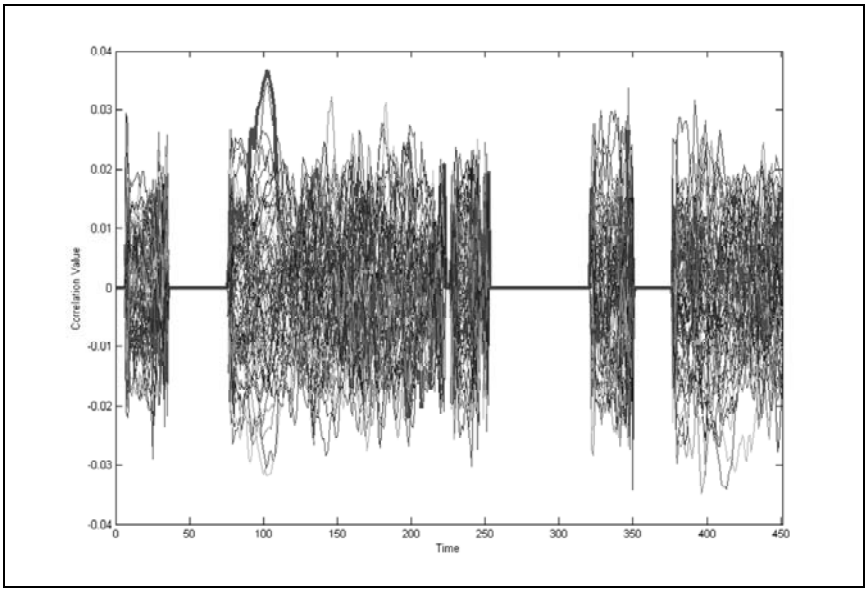

Figure B - 10 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'activité du circuit.

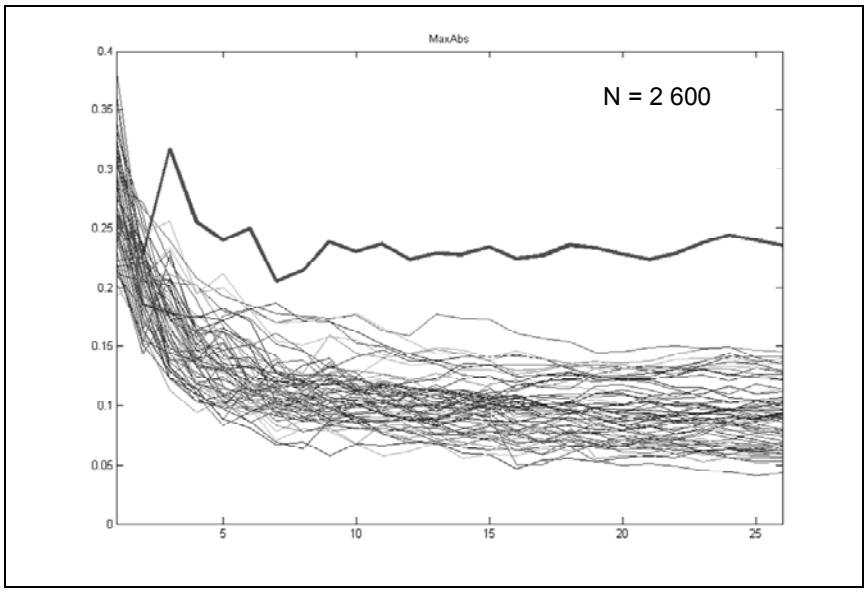

Figure B - 11 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'activité du circuit.

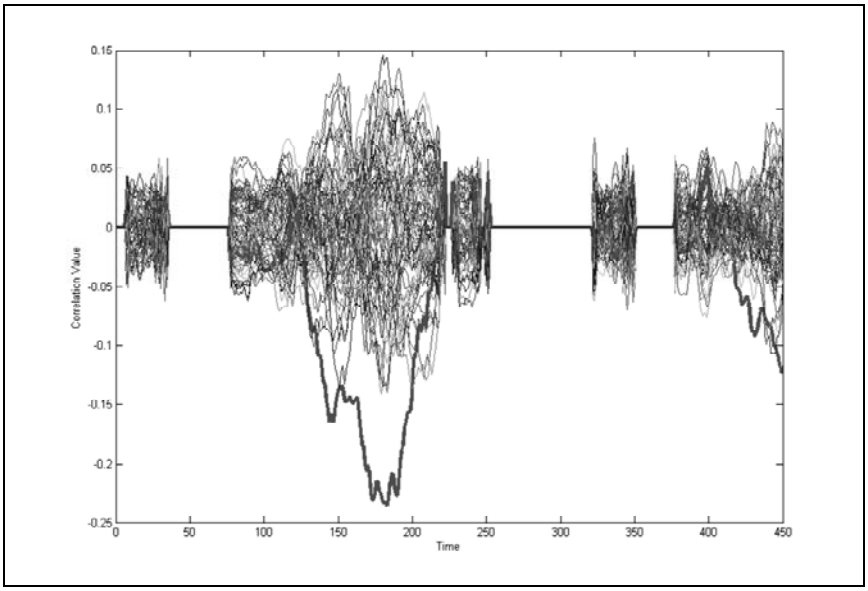

Figure B - 12 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de l'activité du circuit.

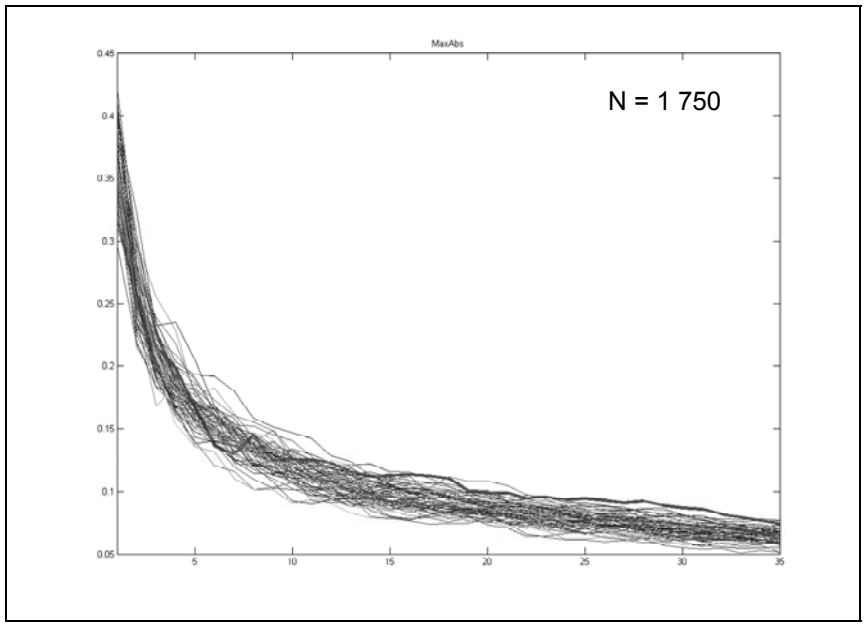

Figure B - 13 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la consommation dynamique.

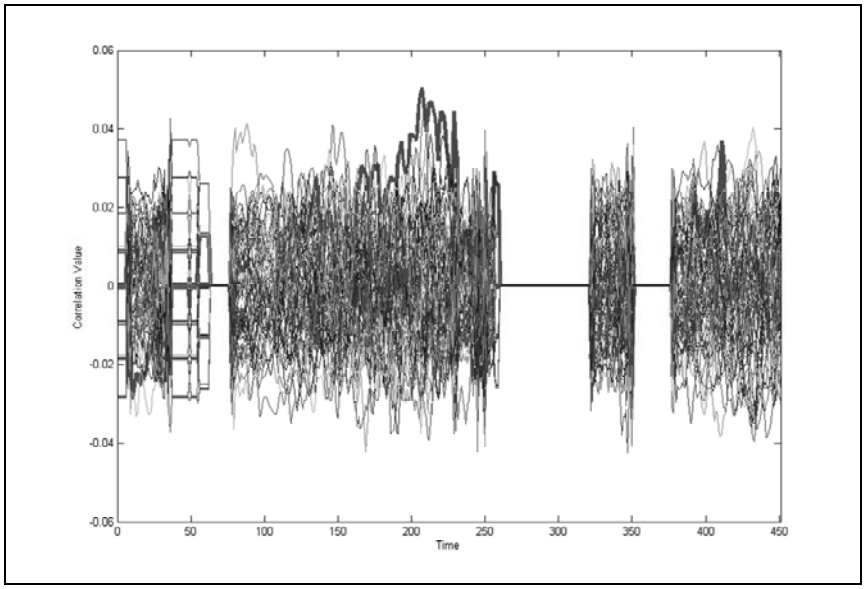

Figure B - 14 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la consommation dynamique.

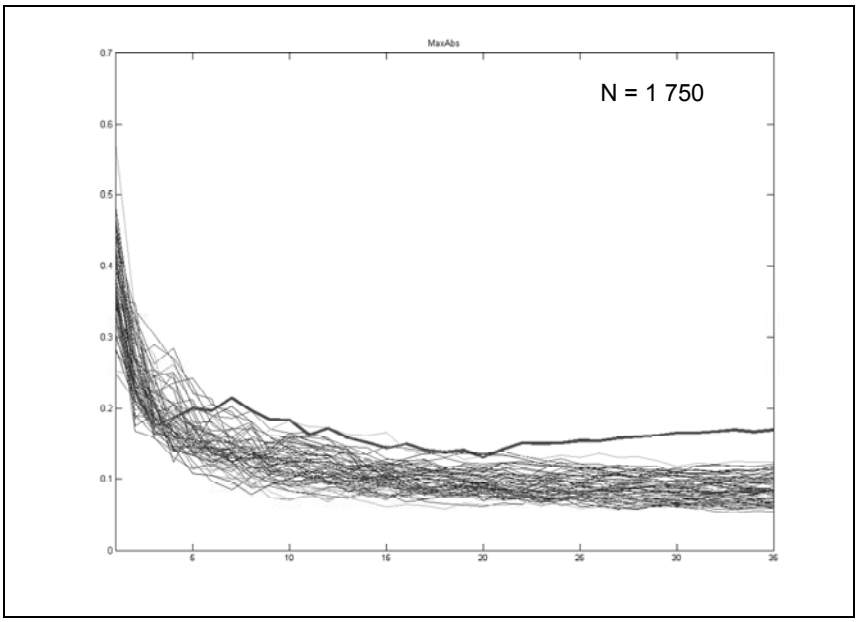

Figure B - 15 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la consommation dynamique.

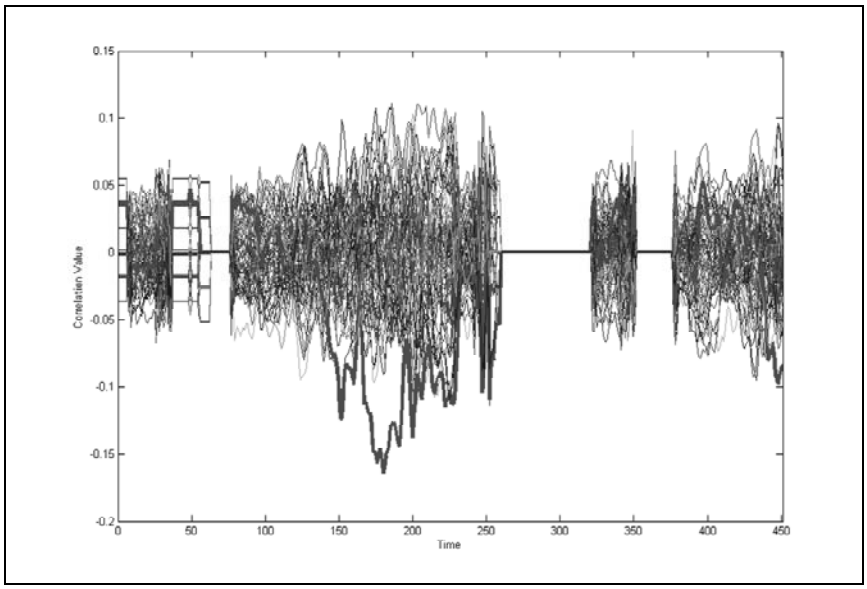

Figure B - 16 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la consommation dynamique.

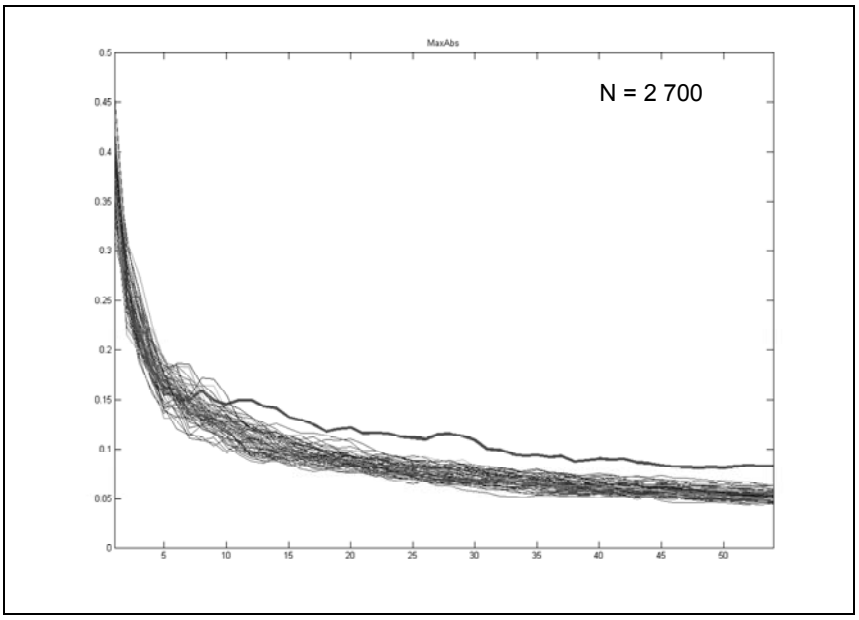

Figure B - 17 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance moyenne.

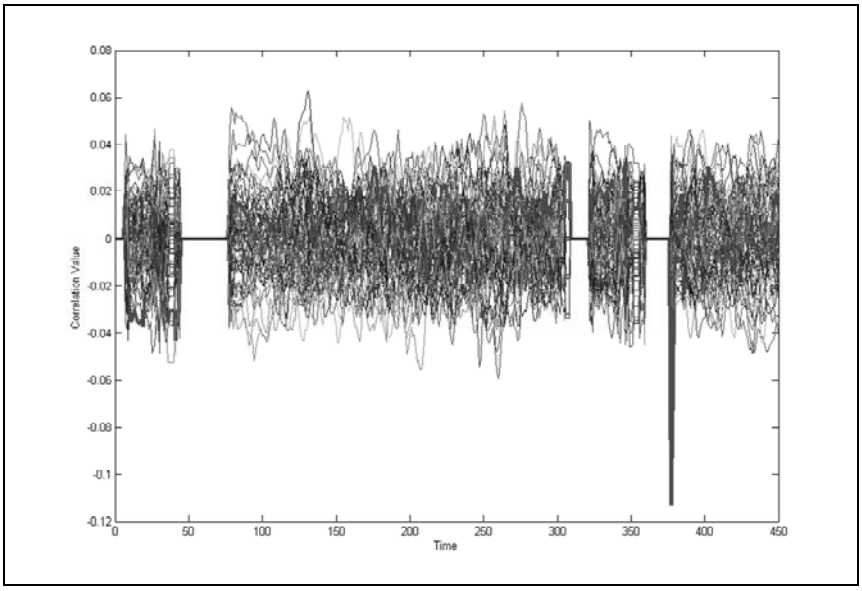

Figure B - 18 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance moyenne.

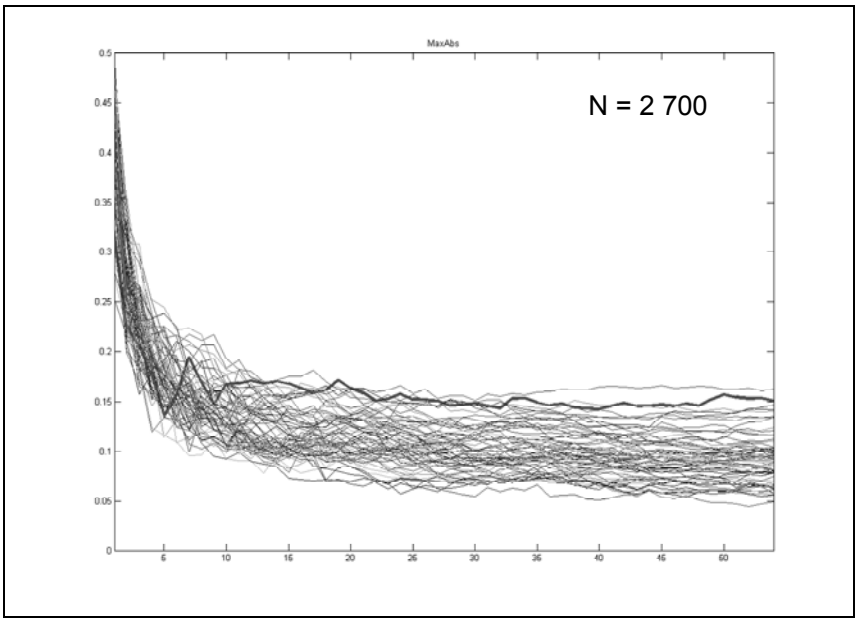

Figure B - 19 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance moyenne.

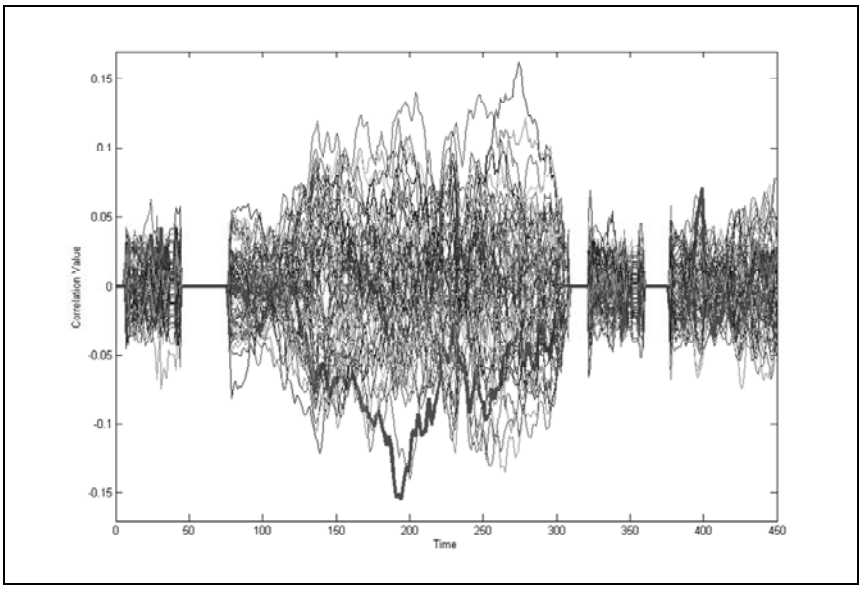

Figure B - 20 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance moyenne.

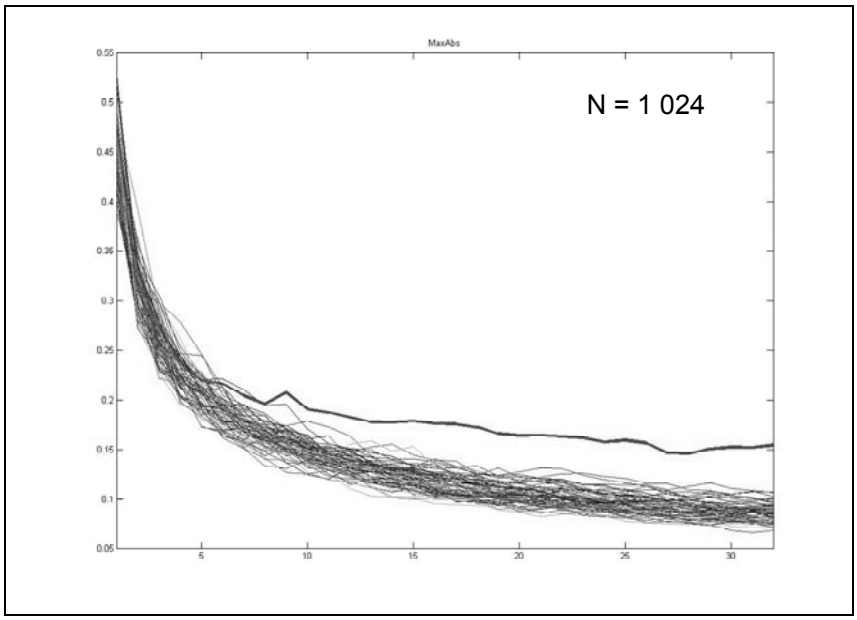

Figure B - 21 : Analyse DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance consommée lors de glitch.

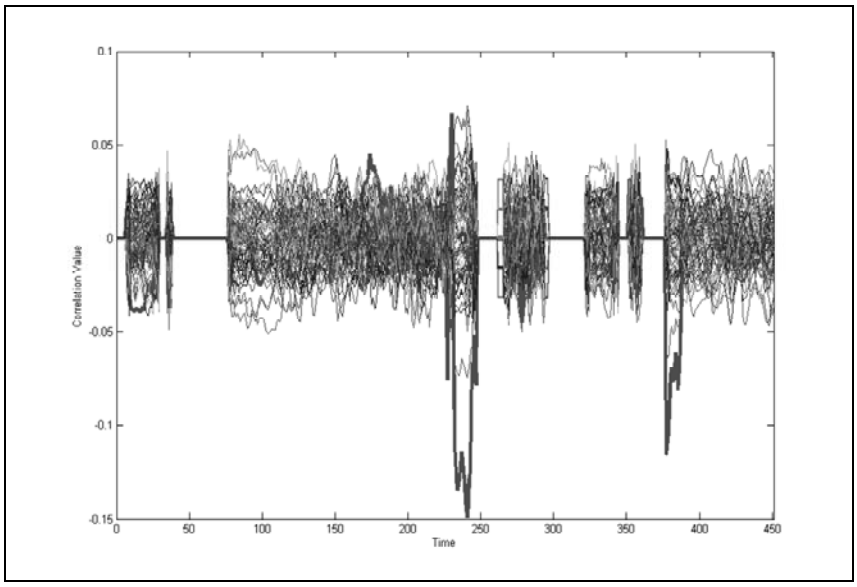

Figure B - 22 : Représentation temporelle de DPA_K lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance consommée lors de glitch.

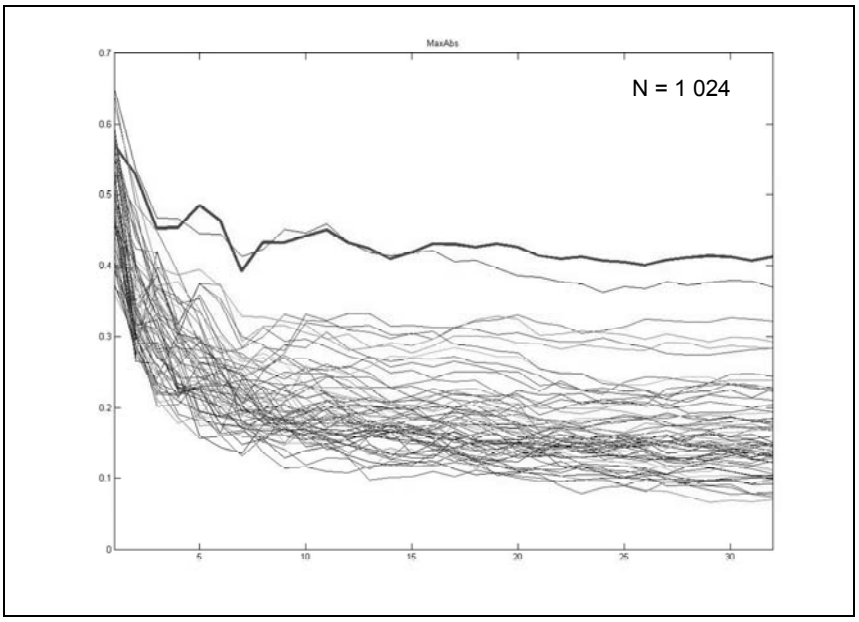

Figure B - 23 : Analyse DPA_H lors de simulations électriques d'une partie de la fonction DES renforcée en fonction de la puissance consommée lors de glitch.

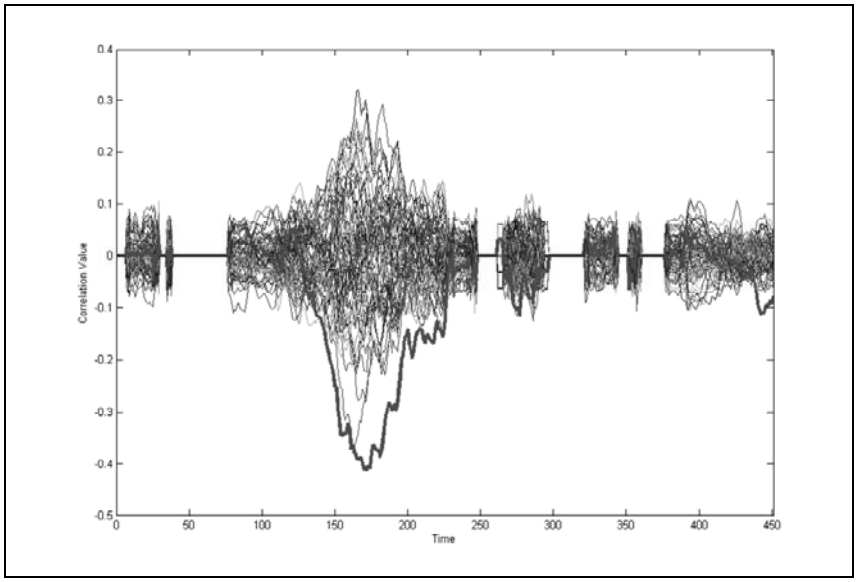

Figure B - 24 : Représentation temporelle de DPA_H lors de simulations électriques d'une partie de la fonction de la puissance consommée lors de glitch.

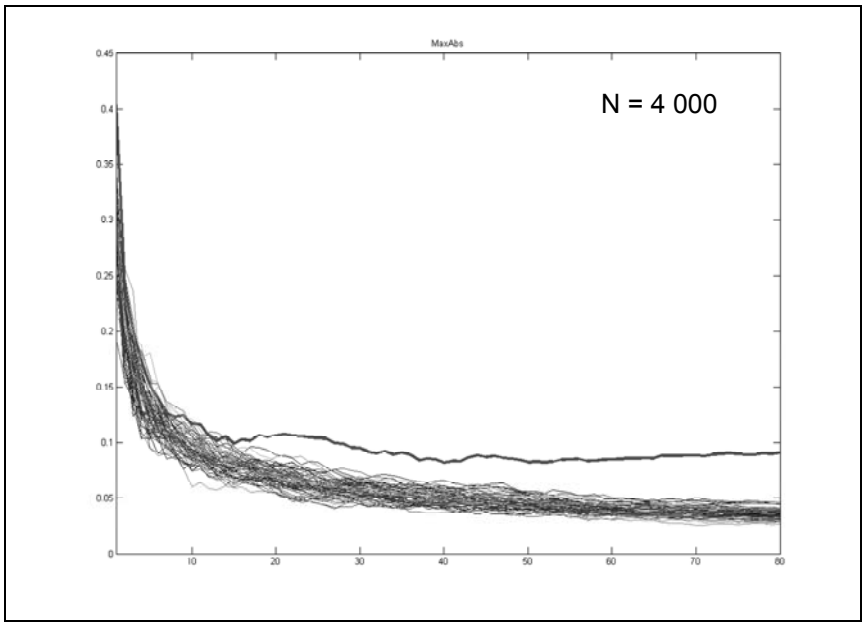

Figure B - 25 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle réduite.

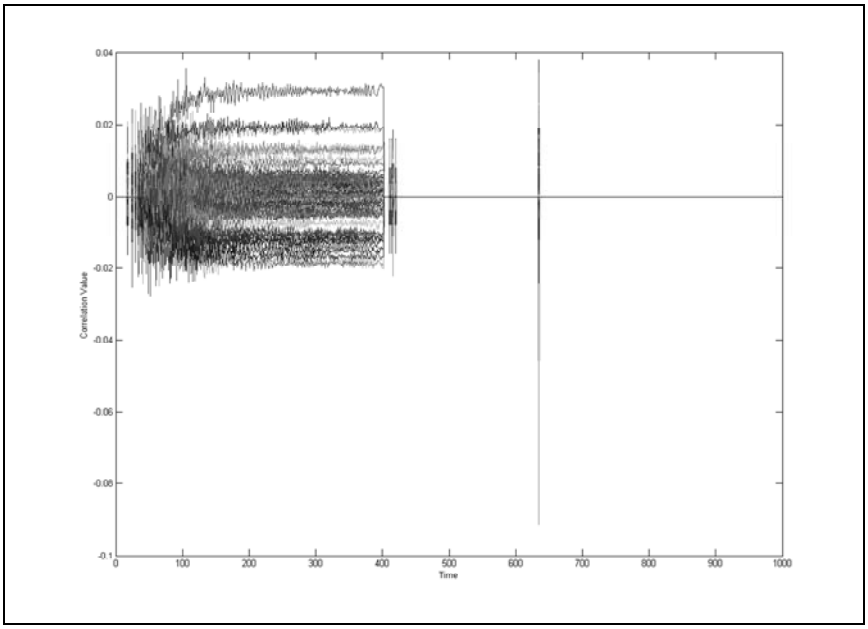

Figure B - 26 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle réduite.

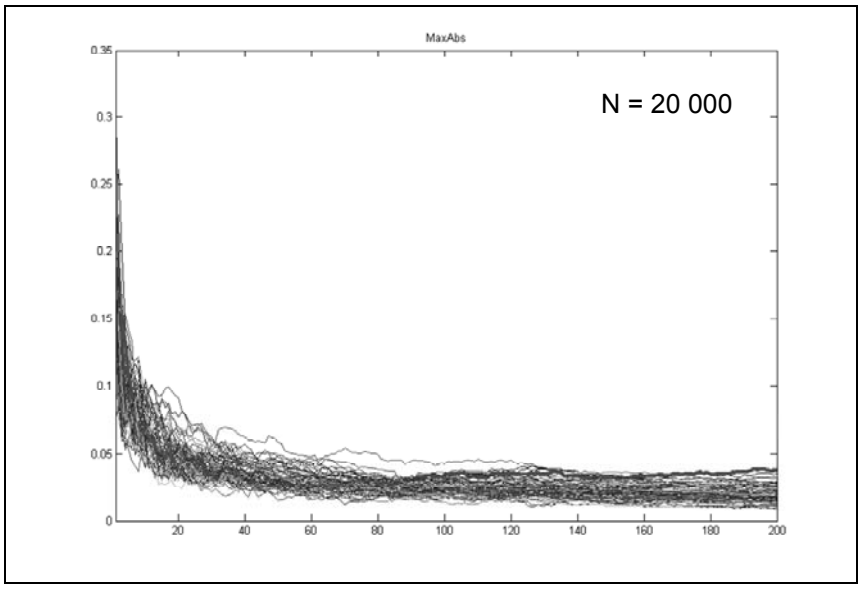

Figure B - 27 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle réduite.

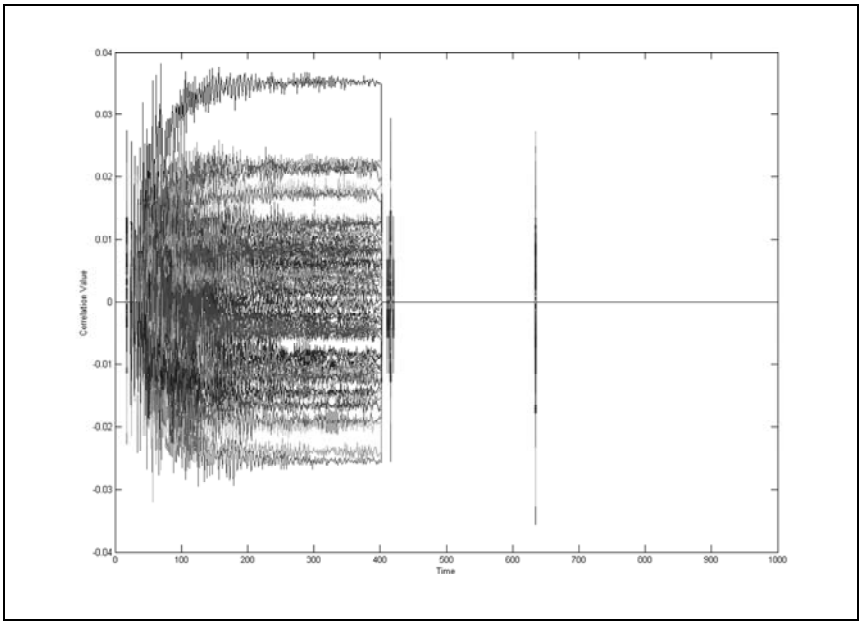

Figure B - 28 : Représentation temporelle de DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle réduite.

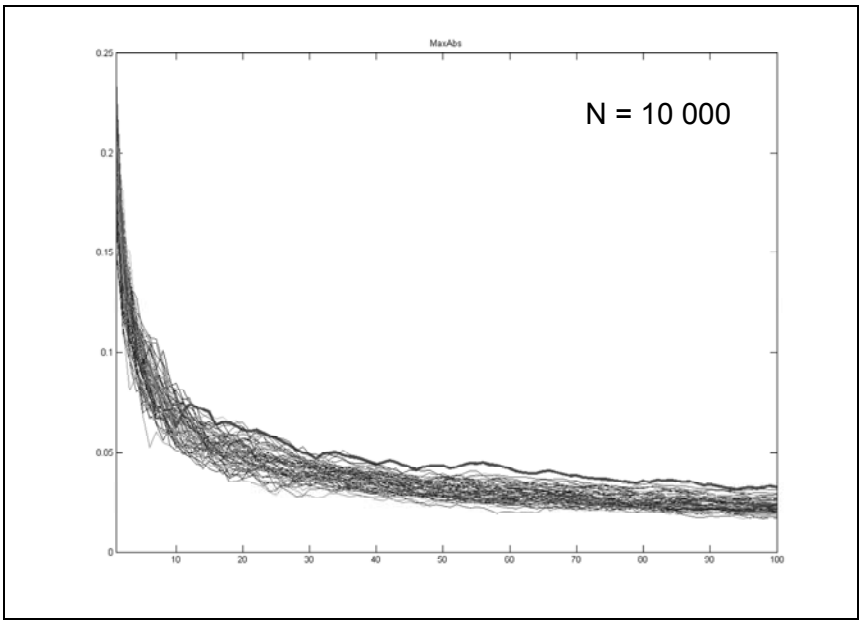

Figure B - 29 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle réduite.

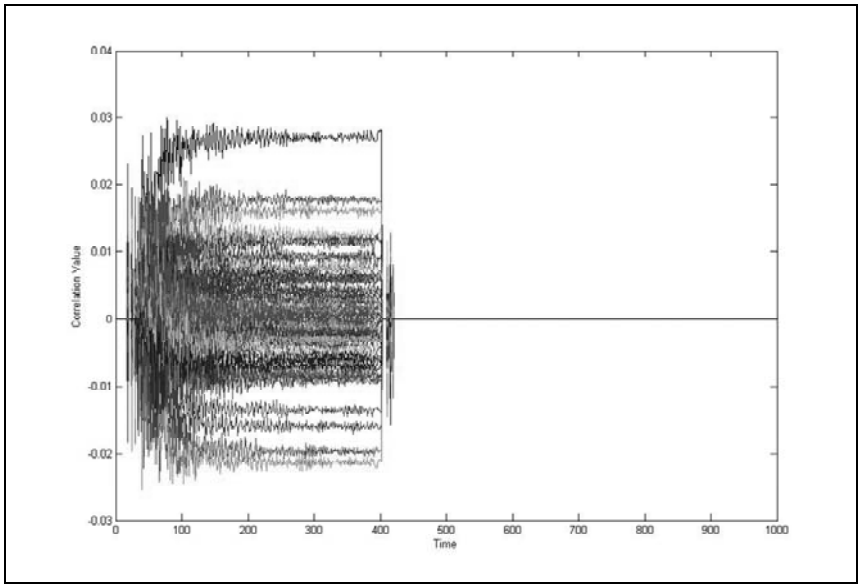

Figure B - 30 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle réduite.

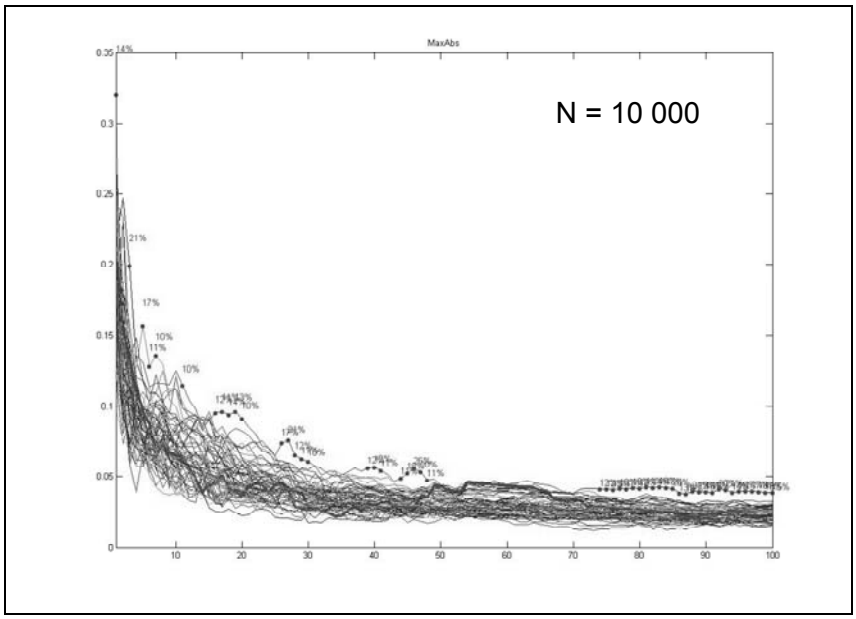

Figure B - 31 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle réduite.

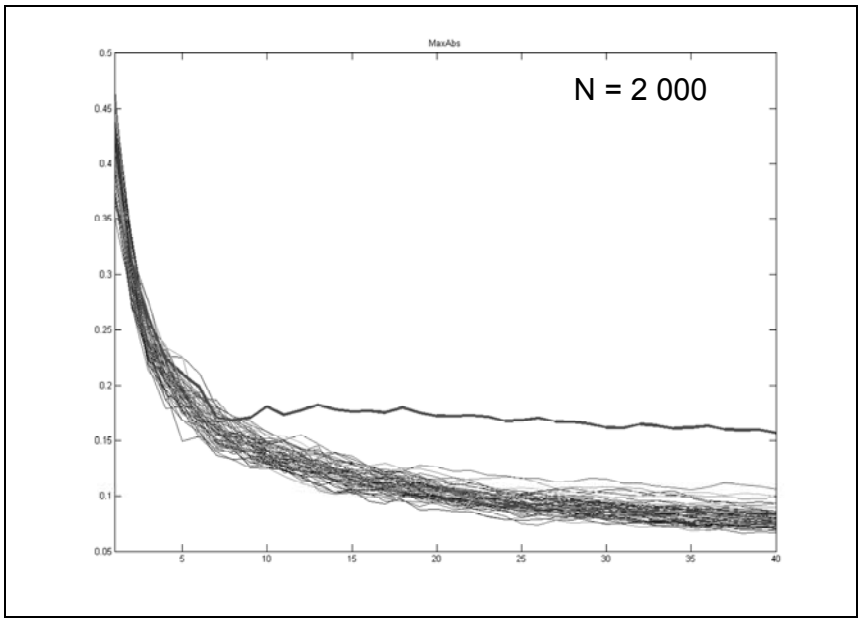

Figure B - 32 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue.

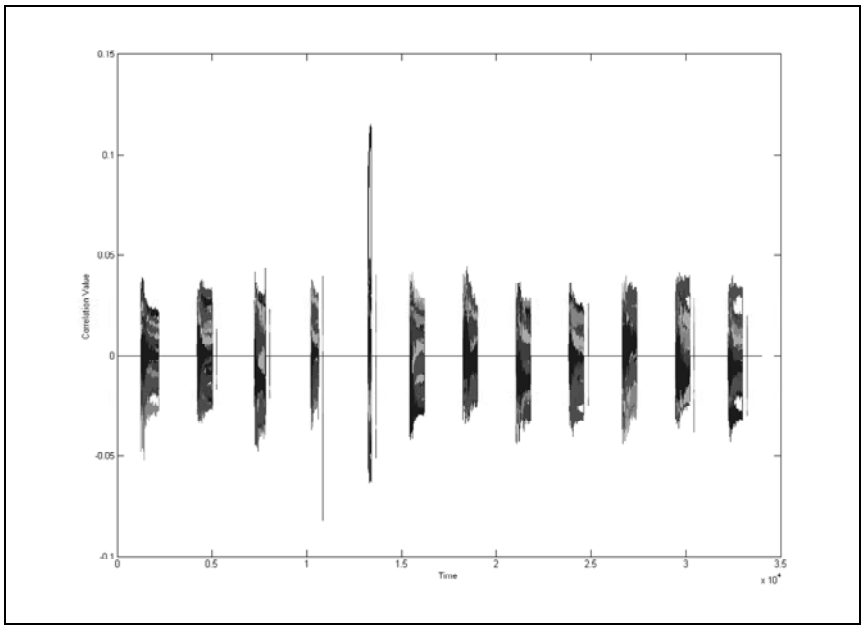

Figure B - 33 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue.

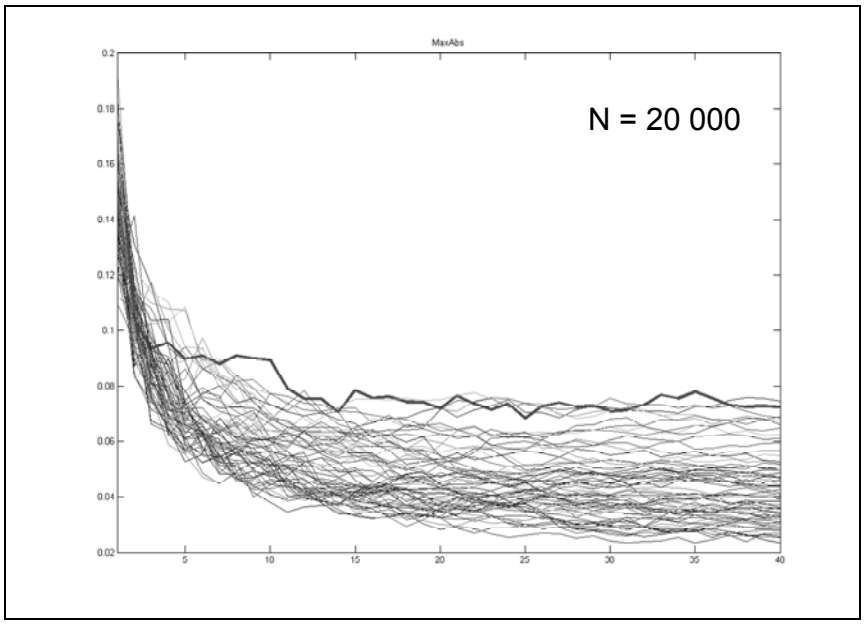

Figure B - 34 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue.

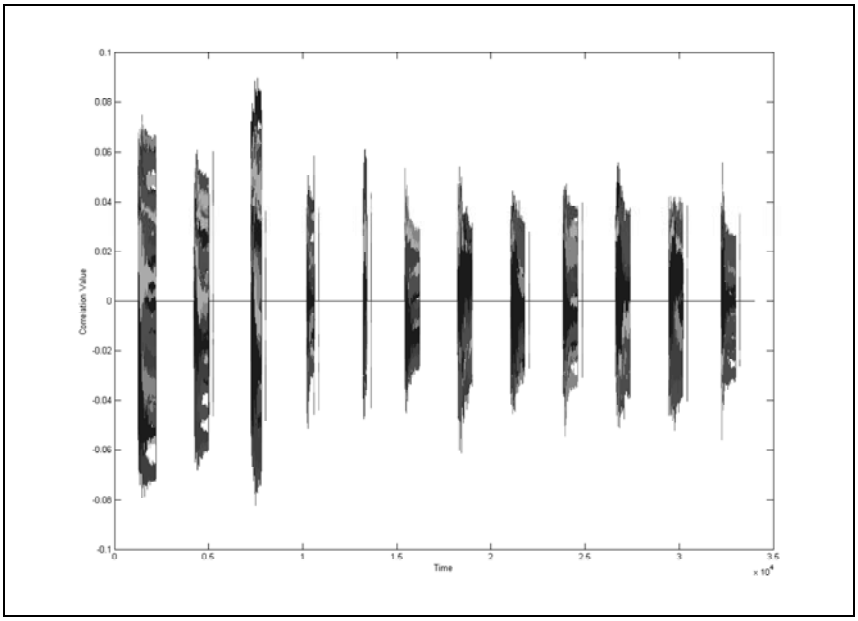

Figure B - 35 : Représentation temporelle de DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue.

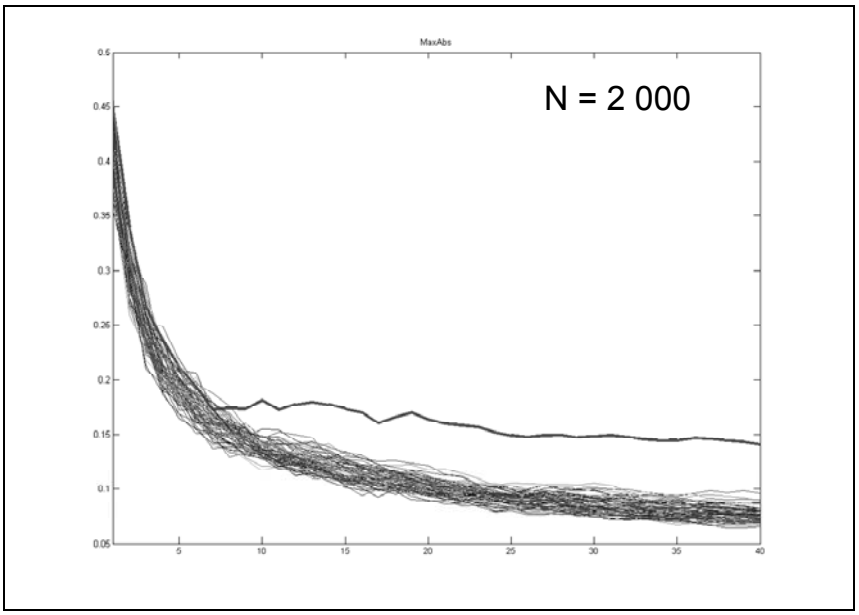

Figure B - 36 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue.

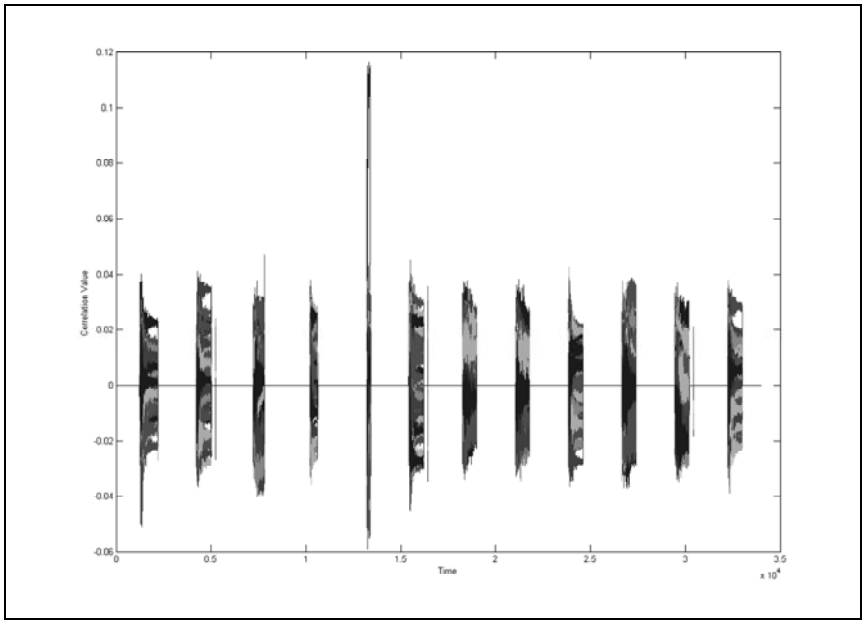

Figure B - 37 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue.

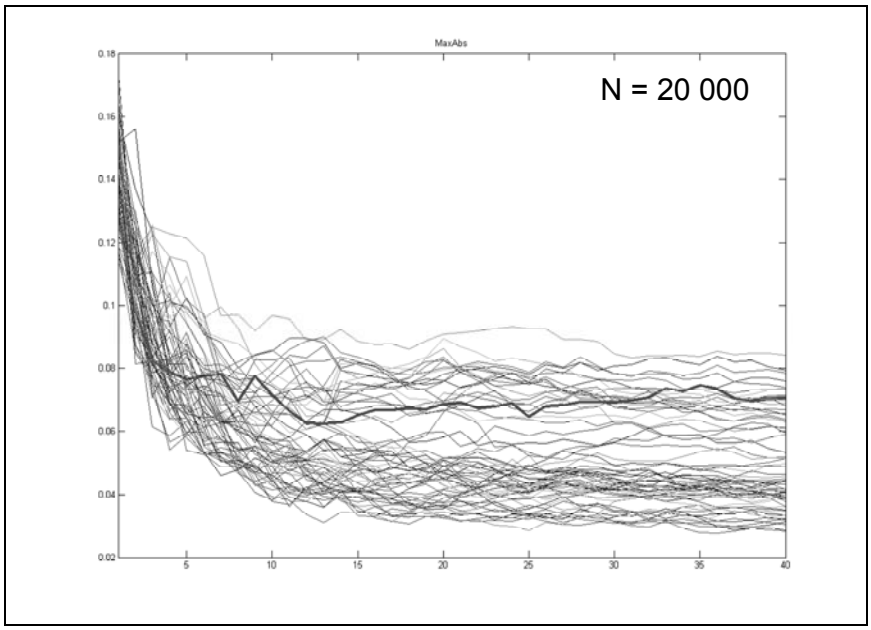

Figure B - 38 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue.

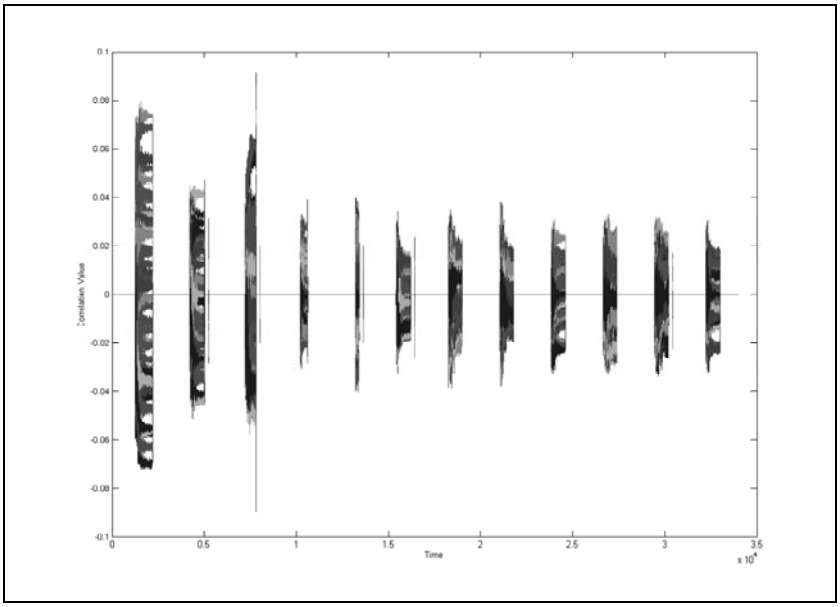

Figure B - 39 : Représentation temporelle de DPA_H lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue.

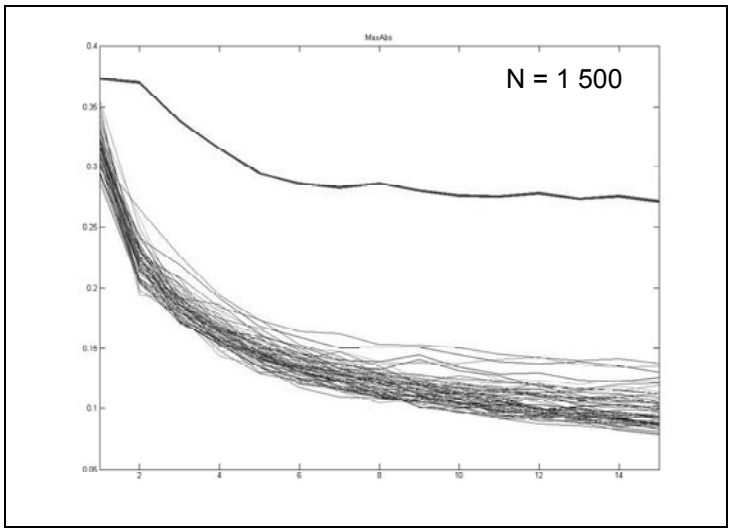

Figure B - 40 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

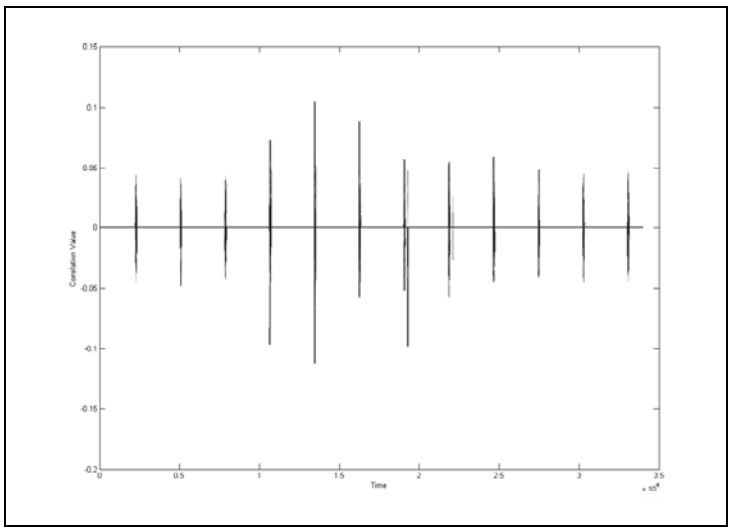

Figure B - 41 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

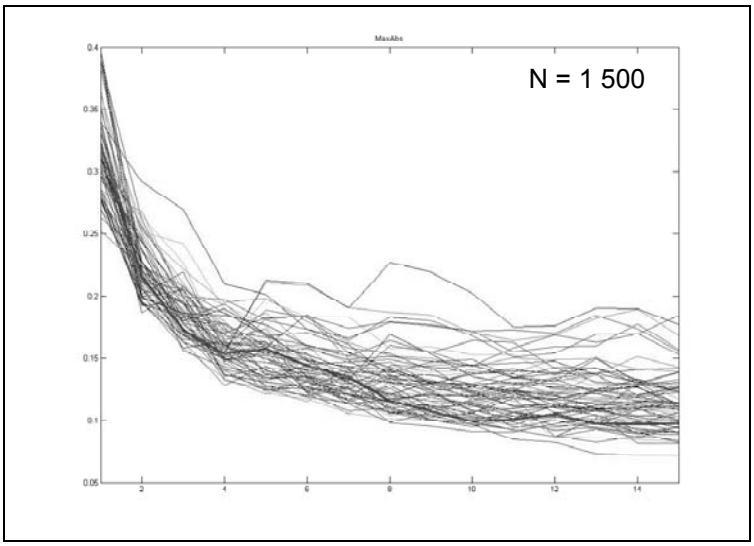

Figure B - 42 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

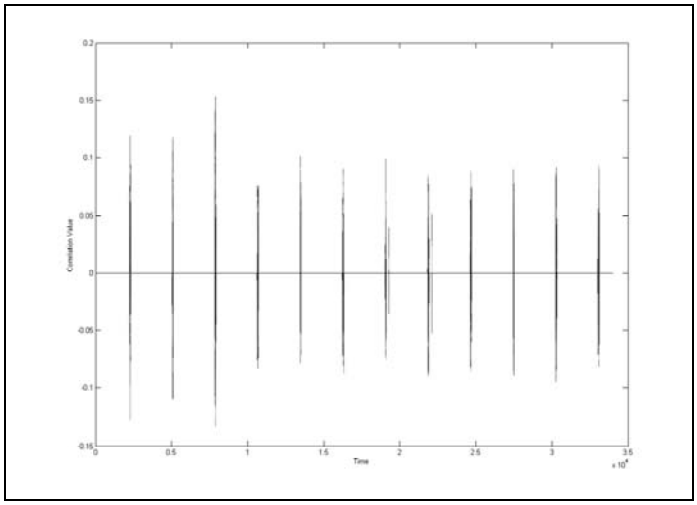

Figure B - 43 : Représentation temporelle de DPA_H lors de simulations logiques du circuit complet avec la netlist initiale, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

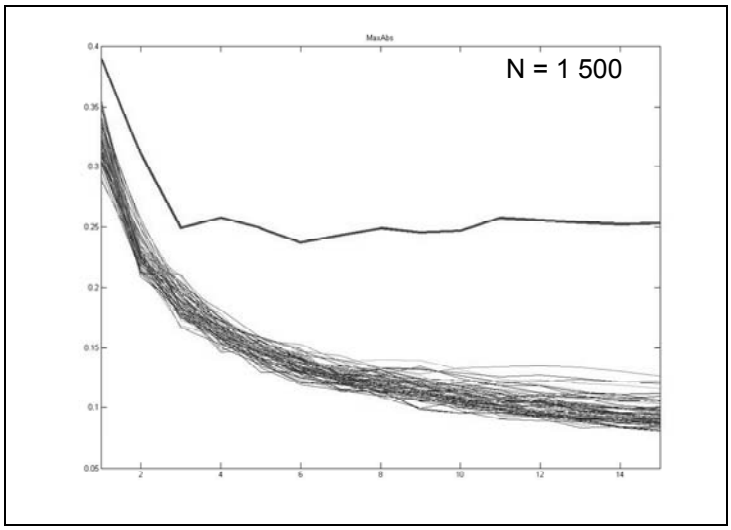

Figure B - 44 : Analyse DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

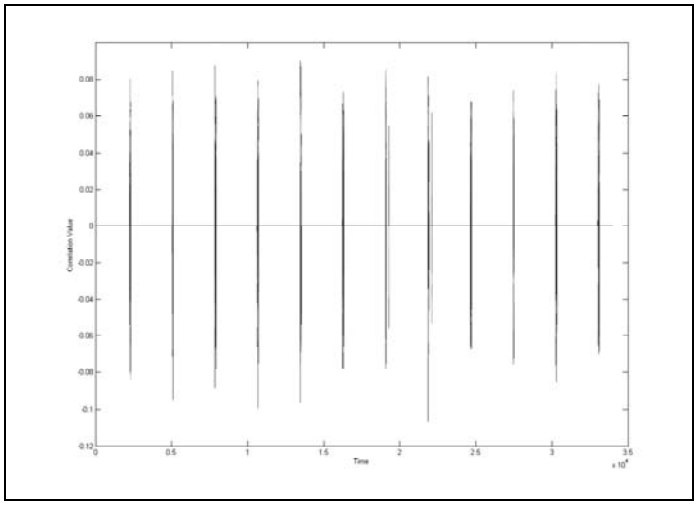

Figure B - 45 : Représentation temporelle de DPA_K lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

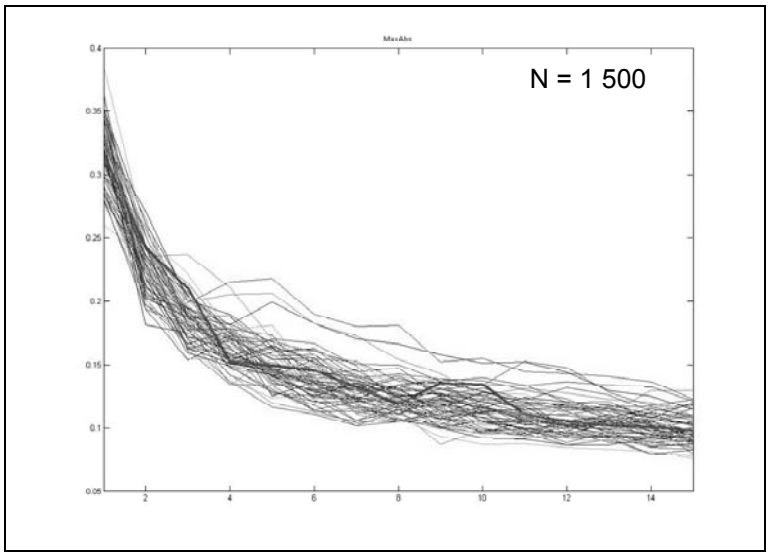

Figure B - 46 : Analyse DPA_H lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

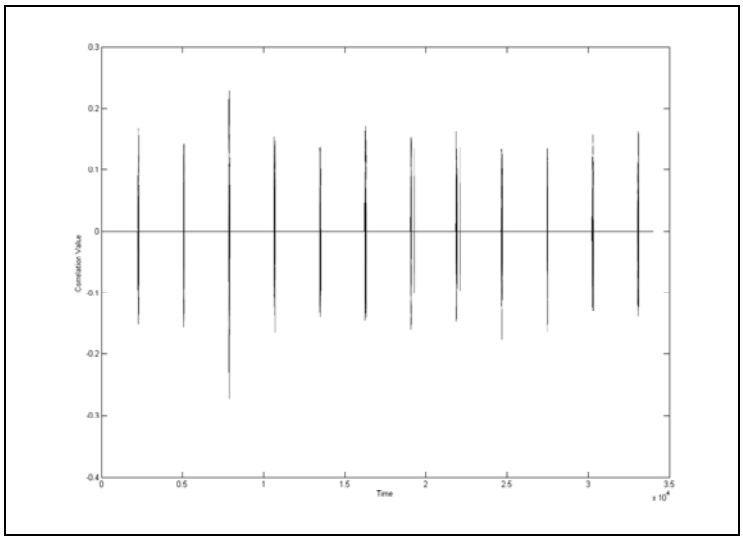

Figure B - 47 : Représentation temporelle de DPA_H lors de simulations logiques du circuit complet avec la netlist renforcée, pour une fenêtre temporelle étendue en tenant compte des éléments parasites.

TITRE:

Intégration de Logique Reconfigurable dans les Circuits Sécurisés

RESUME :

Ces travaux traitent des problèmes de sécurité et de flexibilité dans le domaine des circuits sécurisés. Dans ce manuscrit, après la présentation de notions cryptographiques, nous étudions deux problématiques distinctes. La première concerne les attaques par clonage et retro-ingénierie. Dans ce sens, nous proposons une solution basée sur l'utilisation de logique reconfigurable répartie, et traitons aussi du protocole de reconfiguration associé. La seconde problématique étudiée dans ce manuscrit vise à éviter les attaques par analyse des canaux cachés. Nous suggérons alors une contre-mesure, basée sur la reconfiguration dynamique des chemins de données du circuit intégré. Cette contre-mesure est présentée selon différentes variantes et évaluée selon différents placements et niveaux d'abstraction.

TITLE :

Integration of Reconfigurable Logic into Secure Chips

ABSTRACT :

This work deals with security and flexibility issues for secure chips. We discuss first of cryptographic notions and then study two distinct problem. The first one, concerns cloning and reverse-engineering attacks. In this way, we make a proposition based on use of distributed reconfigurable logic and its associated secure reconfiguration protocol. The other problem concerns the side-channels analysis attacks. We evoke a countermeasure based on reconfiguration of data paths. This countermeasure is studied with different use and evaluated with different placing into chips.

DISCIPLINE :

Microélectronique

MOTS-CLES :

Circuits sécurisés, Cryptanalyse, Attaques matérielles, DPA, Contre-mesures DPA, Circuits reconfigurables.

ADRESSE DU LABORATOIRE :

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)

UMR CNRS / Université Montpellier II, n° C55060

161, rue Ada - 34 392 Montpellier Cedex 5 - France