



HAL
open science

Contribution à la conception de circuits intégrés sécurisés : l'alternative asynchrone

Ghislain Fraidy Bouesse

► **To cite this version:**

Ghislain Fraidy Bouesse. Contribution à la conception de circuits intégrés sécurisés : l'alternative asynchrone. Micro et nanotechnologies/Microélectronique. Institut National Polytechnique de Grenoble - INPG, 2005. Français. NNT : . tel-00011457v2

HAL Id: tel-00011457

<https://theses.hal.science/tel-00011457v2>

Submitted on 25 Jan 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

T H E S E

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : Microélectronique

préparée au laboratoire **TIMA** dans le cadre de
l'**Ecole Doctorale d'« Electronique, Electrotechnique, Automatique,
Télécommunications, Signal »**

présentée et soutenue publiquement

par

Ghislain Fraidy BOUESSE

Le 1 décembre 2005

Titre :

**CONTRIBUTION A LA CONCEPTION DE CIRCUITS INTEGRES
SECURISES : L'ALTERNATIVE ASYNCHRONE**

Directeur de Thèse : Marc Renaudin
Codirecteur : Gilles Sicard

JURY

| | |
|----------------------------|------------------------|
| M. Yassine Lakhnech | , Président |
| M. Jean Jacques Quisquater | , Rapporteur |
| M. Lionel Torres | , Rapporteur |
| M. Marc Renaudin | , Directeur de thèse |
| M. Gilles Sicard | , Codirecteur de thèse |
| M. Alain Merle | , Examinateur |

A mon père
A ma mère

REMERCIEMENTS

Les travaux de cette thèse ont été réalisés au sein du laboratoire TIMA, sur le site Viallet de l'Institut National Polytechnique de Grenoble. Que son directeur M. Bernard Courtois trouve ici tous mes remerciements pour son accueil au sein de son laboratoire.

Toute ma gratitude est adressée à Marc Renaudin, mon Directeur de thèse, professeur à l'Institut National Polytechnique de Grenoble, qui a accepté de me prendre comme stagiaire DEA (2000 – 2001) pour finalement me prendre en thèse en 2002. Tous mes remerciements pour ces belles années de stage et de thèse.

Je tiens à exprimer toute ma profonde reconnaissance à mon co-encadrant Gilles Sicard, maître de conférence à l'université Joseph Fourier, pour sa patience, ses conseils, ses corrections, de son soutien moral et surtout de la semaine de folie que nous avons passée avant ma soutenance.

Je voudrais remercier vivement les membres de mon jury de thèse :

M. Lionel Torres, professeur à l'université de Montpellier II, pour avoir accepté de rapporter ma thèse et de la corriger.

M. Jean Jacques Quisquater, professeur à l'université Catholique de Louvain, pour m'avoir fait l'honneur de participer à mon jury de thèse en tant que rapporteur.

M. Yassine Lakhnech, professeur à l'université Joseph Fourier, pour m'avoir fait l'honneur d'être le président de mon jury de thèse.

M. Alain Merle, responsable du Centre d'Evaluation des Technologies de l'Information (CESTI-LETI), pour avoir accepté d'être examinateur de ma thèse.

Je tiens aussi à remercier Laurent Fesquet, maître de conférence à l'Institut National Polytechnique de Grenoble, pour ses encouragements. Merci Laurent pour toutes ces heures d'enseignement.

A Alain Guyot, maître de conférence à l'Institut Polytechnique de Grenoble, j'adresse mes remerciements pour ses nombreux conseils et surtout pour m'avoir présenté à Marc.

Je souhaite remercier chaleureusement tous les membres (ex-membres) de l'équipe CIS du laboratoire TIMA. D'abord les anciens avec JB pour sa sagesse et surtout pour toutes les corrections apportées sur ce manuscrit. A frère Bob pour sa disponibilité, Dhanistha pour m'avoir appris à utiliser « design analyzer », Manu et Kamel pour m'avoir toujours mis dans l'œil du cyclone, Momo pour ses délires, Jérôme pour ces aventures et Amine pour sa discrétion. Les nouveaux en commençant par Arnaud Baixas pour ses cours de Salsa autour des FPGAs, Caroline Biasi pour son éternel beau sourire, João mon prof. Brésilien de snowboard, Fabien le meneur d'hommes, Bertrand et Yannick mes guides d'Australie, David pour les repas mexicains, Cédric pour sa simplicité, Aurélien pour sa gentillesse, Salim qui me rappelait mes 5 ans passés en Algérie, Yann pour son franc parler et pour son aide en programmation, Estelle pour son écoute, nos longues conversations et surtout pour son soutien moral et Isabelle pour l'ambiance et sa gentillesse. Très gros merci à Sophie pour tous ces circuits que nous avons pu envoyer dans les temps et surtout pour ce magnifique pot de thèse fait avec la petite mexicaine Livier. A Nico et ses belles photos, Adrien pour notre virée à Toulouse, Greg pour toutes ses soirées, à Philippe M. compagnon de bureau pendant une petite mais très riche période.

Je ne saurais oublier Damien (le Canadien), Yanick (le petit), Greg (la ciboule), Seb, Fabrice, Emeline, Clarisse, Patrick, pour toutes ces soirées, dînés, restaurants, sorties de ski et mariages toujours bien arrosés. Toute ma reconnaissance à Eleftherios pour son temps, ses conseils, son assistance et surtout pour ce magnifique voyage effectué en Grèce.

Je voudrais également remercier les administrateurs du CIME, Alexandre Chagoya, Robin Rolland et Bernard Bosc, et tout le personnel administratif du laboratoire TIMA. Tous mes remerciements à Kholdoun Torki pour son aide et à Jean François Paillotin.

Je tiens également à dire merci aux membres du CCDS, du CCREG et à mes amis qui ont pris la peine de venir assister à ma soutenance et tout particulièrement à Marlene L.

Je souhaite remercier également toutes les personnes que j'ai rencontrées dans les différents projets de recherche qui ont contribué à valoriser ce travail, à Edith Beigne et Bruno Robisson du CEA-LETI, à Jacques Sonzogni, Solenn Prevost et Pierre Yvan Liardet de STMicroelectronics Rousset, à Carine Raynaud, Jean Christophe Courrage et Benoît Feix du CESTI-THALES et à Fabien Germain de la SGDN.

Je remercie énormément ma famille, à mon père et à ma mère, à mon oncle (Nguba Martial) qui a su croire en moi, à mes frères et sœurs et particulièrement à Ya Ngoma, Ya Solange et Ya Willy présent le jour de ma soutenance et Pat D. qui a su m'accueillir à Grenoble. Au père Hiss pour son soutien indéfectible.

Une pensée particulière à tous ceux qui de près ou de loin m'ont donné un jour ce petit rayon de soleil synonyme d'amour et de lumière par sa chaleur et son éclat.

TABLE DES MATIÈRES

| | |
|---|----------|
| INTRODUCTION GENERALE : CONTEXTE ET MOTIVATIONS | 1 |
| CHAPITRE I | |
| CRYPTANALYSE MATERIELLE : ATTAQUES PAR CANAUX CACHES..... | 7 |
| 1.1 Introduction | 7 |
| 1.2 La Cryptologie : Notion de Base et terminologie | 8 |
| 1.2.1 La Cryptographie | 8 |
| 1.2.1.1 Notion de Clé cryptographique..... | 8 |
| 1.2.2 Cryptage symétrique | 9 |
| 1.2.2.1 Le Standard de chiffrement de données : le <i>DES</i> | 9 |
| 1.2.3 Cryptage Asymétrique | 11 |
| 1.2.3.1 Le RSA..... | 11 |
| 1.3 Définition de la Cryptanalyse | 12 |
| 1.4 La cryptanalyse dite théorique..... | 14 |
| 1.4.1 La cryptanalyse linéaire | 14 |
| 1.4.2 La cryptanalyse différentielle | 14 |
| 1.5 La cryptanalyse dite logicielle | 15 |
| 1.5.1 Le cheval de Troie | 15 |
| 1.5.2 Attaque par débordement de pile (Buffer overflow)..... | 15 |
| 1.6 La cryptanalyse matérielle..... | 16 |
| 1.6.1 Les attaques intrusives | 16 |
| 1.6.1.1 Attaque par reconstruction du layout..... | 16 |
| 1.6.1.2 Attaque par analyse sous pointes..... | 16 |
| 1.6.2 Les Attaques non intrusives | 17 |
| 1.6.2.1 Les Attaques par injections de fautes | 17 |
| 1.6.2.1.1 Génération et modèles de fautes | 17 |
| 1.6.2.1.2 Exploitation des fautes..... | 18 |
| 1.6.2.2 Les attaques par analyses des signaux compromettants..... | 21 |
| 1.6.2.2.1 Les attaques temporelles..... | 22 |
| 1.6.2.2.2 Les attaques électromagnétiques..... | 24 |
| 1.6.2.2.3 Les attaques en puissance ou attaques par analyse du courant..... | 27 |
| 1.7 Conclusion | 28 |

| | |
|--|-----------|
| CHAPITRE II | |
| LES ATTAQUES EN PUISSANCE : ETUDE ET ANALYSE..... | 29 |
| 2.1 Introduction..... | 29 |
| 2.2 Mesures du courant..... | 29 |
| 2.3 Attaque par simple analyse du courant | 32 |
| 2.4 Attaque par analyse différentielle du courant | 33 |
| 2.4.1 Analyse théorique de l’algorithme : la fonction de sélection..... | 34 |
| 2.4.2 Collecte d’information : mesure et mémorisation des courbes de courant | 38 |
| 2.4.3 Analyse des corrélations..... | 38 |
| 2.5 Attaque par collisions | 43 |
| 2.6 Attaque par corrélation..... | 45 |
| 2.7 Réalisation d’une attaque en puissance sur un cryptoprocasseur synchrone | 47 |
| 2.7.1 Architecture et caractéristiques du circuit : Cryptoprocasseur DES | 47 |
| 2.7.2 Analyse différentielle du cryptoprocasseur..... | 48 |
| 2.8 Conclusion | 53 |
| | |
| CHAPITRE III | |
| ANALYSE DETAILLEE DES ATTAQUES EN PUISSANCE : ETAT DE L’ART SUR LES | |
| CONTRE-MESURES..... | 55 |
| 3.1 Introduction..... | 55 |
| 3.2 Analyse du profil de courant dans un circuit intégré | 56 |
| 3.2.1 Analyse du courant d’une porte inverseuse..... | 56 |
| 3.2.1.1 Effet de la rampe d’entrée..... | 58 |
| 3.2.1.1.1 Les rampes rapides..... | 58 |
| 3.2.1.1.2 Les rampes lentes | 59 |
| 3.2.1.2 Effet du délai..... | 60 |
| 3.2.1.3 Effet de la capacité..... | 61 |
| 3.2.1.4 Effet des commutations en sortie | 62 |
| 3.2.2 Analyse du courant des portes simples et complexes..... | 63 |
| 3.2.3 Analyse du courant dans un bloc..... | 66 |
| 3.2.4 Sensibilité électrique d’un bloc | 69 |
| 3.3 La cryptanalyse matérielle : les contre-mesures | 70 |
| 3.3.1 Les contre-mesures contre les attaques invasives | 70 |
| 3.3.2 Les contre-mesures contre les attaques par injection de fautes..... | 71 |
| 3.3.3 Les contre-mesures contre les attaque temporelles | 72 |

| | |
|--|----|
| 3.3.4 Les contre-mesures contre les attaques électromagnétiques | 72 |
| 3.3.5 Les contre-mesures contre les attaques en puissance..... | 72 |
| 3.3.5.1 Les méthodes de masquage et de duplication | 73 |
| 3.3.5.2 Méthodes par injection de bruit | 74 |
| 3.3.5.3 Les méthodes par lissage et uniformisation du profil de courant..... | 76 |
| 3.4 Conclusion | 77 |

CHAPITRE IV

| | |
|---|-----------|
| LA LOGIQUE ASYNCHRONE ET LA SECURITE MATERIELLE : EVALUATION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS | 79 |
| 4.1 Introduction | 79 |
| 4.2 La logique asynchrone..... | 80 |
| 4.2.1 Les opérateurs asynchrones | 80 |
| 4.2.2 Le contrôle local | 81 |
| 4.2.3 Protocole de communication..... | 82 |
| 4.2.4 Codage des données..... | 82 |
| 4.2.5 Signaux de fin de calcul..... | 84 |
| 4.2.6 Les différentes classes de circuits asynchrones | 84 |
| 4.2.7 Les circuits Quasi Insensibles aux Délais (QDI) | 85 |
| 4.2.8 Les circuits micropipeline..... | 85 |
| 4.2.9 La porte de Muller | 87 |
| 4.2.10 Conception des circuits asynchrones | 87 |
| 4.3 Les circuits asynchrones et les attaques non intrusives | 88 |
| 4.3.1 Choix du type de circuit asynchrone..... | 88 |
| 4.3.2 Les circuits QDI et les analyses en puissance..... | 88 |
| 4.3.2.1 Le Codage de données <i>en p parmi n</i> | 89 |
| 4.3.2.2 Le protocole de communication 4 phases..... | 90 |
| 4.3.2.3 Chemins de données équilibrés | 90 |
| 4.4 Evaluation de la logique asynchrone <i>QDI</i> : Cas d'un cryptoprocasseur asynchrone | 93 |
| 4.4.1 Architecture du cryptoprocasseur | 93 |
| 4.4.1.1 Le banc de registres | 93 |
| 4.4.1.2 Les Interfaces | 94 |
| 4.4.1.3 le Bloc de chiffrement asynchrone | 95 |
| 4.4.2 Caractéristiques des circuits..... | 96 |
| 4.4.3 Résultats des attaques en puissance | 98 |

| | |
|----------------------|-----|
| 4.5 Conclusion | 101 |
|----------------------|-----|

CHAPITRE V

ANALYSE FORMELLE DES CIRCUITS QUASI INSENSIBLES AUX DELAIS 103

| | |
|--|-----|
| 5.1 Introduction..... | 104 |
| 5.2 Représentation formelle des circuits asynchrones <i>QDI</i> | 104 |
| 5.2.1 Graphe orienté d'un circuit asynchrone <i>QDI</i> | 104 |
| 5.2.1.1 Matrice d'adjacence | 106 |
| 5.3 Analyse au niveau logique : symétrie des chemins de données | 107 |
| 5.3.1 Notion de chemin d'exécution | 108 |
| 5.3.2 Les graphes isomorphes | 111 |
| 5.4 Analyses au niveau électrique : Identification des fuites d'informations | 112 |
| 5.4.1 Modèle électrique d'un circuit asynchrone <i>QDI</i> | 113 |
| 5.4.2 Application de l'attaque DPA sur le modèle formel | 115 |
| 5.4.3 Validation par simulation électrique | 116 |
| 5.5 Conclusion | 118 |

CHAPITRE VI

PROTECTION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS : FLOT DE CONCEPTION SECURISE.....121

| | |
|---|-----|
| 6.1 Introduction..... | 121 |
| 6.2 Flot de conception sécurisé des circuits <i>QDI</i> | 122 |
| 6.2.1 Critère d'évaluation de la dissymétrie électrique | 123 |
| 6.2.2 Placement routage contraint | 123 |
| 6.2.3 Cellules asynchrones | 124 |
| 6.3 Les cryptoprocresseurs asynchrones <i>AES</i> | 126 |
| 6.3.1 Le banc de registres et les interfaces | 127 |
| 6.3.2 Le bloc de chiffrement (<i>AES_core</i>) et le bloc de génération des clés (<i>AES_key</i>) | 128 |
| 6.3.3 Caractéristiques des circuits | 131 |
| 6.3.4 Résultats des analyses en courant..... | 134 |
| 6.4 Conclusion | 137 |

| | |
|--|------------|
| CHAPITRE VII | |
| PROTECTION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS : | |
| CONTRE-MESURES..... | 139 |
| 7.1 Introduction | 139 |
| 7.2 Méthode par transpositions aléatoires des chemins de données | 140 |
| 7.2.1 Transpositions aléatoires des chemins de données : Définition..... | 140 |
| 7.2.2 Application de la <i>DPA</i> sur le modèle formel avec transpositions des chemins de données | 143 |
| 7.2.3 La fonction de transposition des chemins de données | 144 |
| 7.2.4 Discussion..... | 145 |
| 7.2.5 Etude de cas : le cryptoprocresseur asynchrone DES | 146 |
| 7.2.6 Validation par simulations électriques..... | 149 |
| 7.3 Méthode par décalage aléatoire des signaux d’acquittements | 152 |
| 7.3.1 Les signaux d’acquittements..... | 152 |
| 7.3.2 Décalage aléatoire des signaux d’acquittements : Analyses formelles..... | 153 |
| 7.3.3 Discussion..... | 155 |
| 7.3.4 Réduction des émissions electromagnétiques..... | 158 |
| 7.3.4.1 Résultats des analyses : simulations électriques..... | 159 |
| 7.3.5 Etude de cas : cryptoprocresseur <i>DES</i> en logique asynchrone <i>QDI</i> | 161 |
| 7.4 Conclusion..... | 164 |
| CONCLUSION ET PERSPECTIVES.... | 165 |
| BIBLIOGRAHIE | 169 |
| BIBLIOGRAHIE DE L’AUTEUR..... | 177 |

LISTES DES FIGURES

| | |
|---|----|
| Figure 1 : Flots de conception supportés par l’environnement <i>TAST</i> | 3 |
| Figure 2 : Le projet <i>TAST</i> | 4 |
| Figure 1.1 : Schéma de Feistel et la fonction f du Standard de chiffrement de données..... | 10 |
| Figure 1.2 : Les attaques cryptanalytiques et les compétences requises | 13 |
| Figure 1.3 : Vue en coupe des transistors <i>CMOS</i> de type <i>N</i> et <i>P</i> sur un substrat | 25 |
| Figure 2.1 : Schéma de mesure et d’acquisition automatique des courbes de courant..... | 30 |
| Figure 2.2 : Mesures différentielles de la première itération de l’algorithme du <i>DES</i> sur un cryptoprocresseur synchrone. Courbes de courant moyennées par 100. | 31 |
| Figure 2.3 : Courbe de courant réalisée par <i>GEMPLUS</i> sur une implémentation <i>RSA</i> utilisant la méthode binaire présentée au chapitre I (cf. § 1.6.2.2.1). | 33 |
| Figure 2.4: Analyse des dépendances du courant consommé dans une architecture en fonction des valeurs d’entrées m_i | 42 |
| Figure 2.5 : Architecture du cryptoprocresseur synchrone <i>DES</i> | 47 |
| Figure 2.6 : Layout du cryptoprocresseur Surface du cœur: $370\mu\text{m} \times 370\mu\text{m}$, Surface avec plots: $1050\mu\text{m} \times 1050\mu\text{m}$ | 48 |
| Figure 2.7 : Profil de courant du cryptoprocresseur synchrone <i>DES</i> | 49 |
| Figure 2.8 : Courant moyennée et instantané de la première itération du cryptoprocresseur synchrone | 50 |
| Figure 2.9 : Analyse des corrélations entre les données manipulées et les courbes de courant | 50 |
| Figure 2.10 : Signal <i>DPA</i> sur le bit 2 en sortie de la <i>SBOX1</i> | 51 |
| Figure 2.11 : Chemin de données du bit 2 en sortie de la <i>SBOX1</i> | 52 |
| Figure 2.12 : Signal <i>DPA</i> sur le bit 3 en sortie de la <i>SBOX1</i> | 52 |
| Figure 2.13 : Signaux <i>DPA</i> sur les bits 1 et 4 en sortie de la <i>SBOX1</i> | 53 |
| Figure 3.1 : Tensions d’entrée/sortie et courants des transistors <i>P</i> et <i>N</i> dans un inverseur <i>CMOS</i> asymétrique ($k=W_p/W_n=1,79$) sur front montant. La capacité de charge est de 16fF. | 57 |
| Figure 3.2 : Zones de fonctionnement de la porte inverseuse. a- correspond à la zone surtension; b- à celle de cour circuit et c- correspond à la zone de décharge..... | 58 |
| Figure 3.3 : Effet d’une rampe rapide sur le profil de courant. L’intervalle de variation de la rampe est tel que $5\text{ps} \leq \tau_{in} \leq 40\text{ps}$ | 59 |

| | |
|--|----|
| Figure 3.4 : Effet d'une rampe lente sur le profil de courant. L'intervalle de variation de la rampe est tel que $80 ps \leq \tau_{in} \leq 160 ps$ | 60 |
| Figure 3.5 : Effet de la capacité de charge sur le profil de courant et sur la rampe de sortie dans le domaine des rampes rapides ($\tau_{in} = 40 ps$) pour $C: 80 fF \leq C \leq 320 fF$ | 61 |
| Figure 3.6 : Effet de la capacité de charge sur le profil de courant et sur la rampe de sortie dans le domaine des rampes lentes ($\tau_{in} = 50 ns$) pour $C: 10 fF \leq C \leq 80 fF$ | 62 |
| Figure 3.7 : Profils des courants des transistors P et N des portes inverseuses symétrique et asymétrique | 63 |
| Figure 3.8 : Schématique d'une porte $CMOS NAND$ à 3 entrées | 64 |
| Figure 3.9 : Profils de courant des transistors en série d'une porte $NAND$ à 3 entrées et variations du délai dans le domaine des rampes rapides | 64 |
| Figure 3.10 : Profils de courant des transistors en série d'une porte $NAND$ à 3 entrées et variations du délai dans le domaine des rampes lentes | 65 |
| Figure 3.11 : Schématique du bloc $SBOX1$: Chemin de donnée du bit de sortie 2 | 67 |
| Figure 3.12 : Courants et variations du nombre de commutations en fonction du poids de Hamming des entrées dans le bloc $SBOX1$ | 68 |
| Figure 3.13 : Courant de la $SBOX1$ avec et sans décalage implémenté sur l'écriture dans la moitié des registres d'entrées | 76 |
| Figure 4.1: Communication de type requête - acquittement | 81 |
| Figure 4.2: Principe du protocole 4 phases | 82 |
| Figure 4.3: Codage trois états | 83 |
| Figure 4.4: Protocole de communication 4 phases "données groupées" ou Bundled Data | 83 |
| Figure 4.5: Différentes classes des circuits asynchrones | 84 |
| Figure 4.6: Structure de base des circuits Micropipelines | 86 |
| Figure 4.7: Structure Micropipeline avec traitement et registre | 86 |
| Figure 4.8: Table de vérité, symbole et exemple d'implémentation d'une porte de Muller à 2 entrées à l'aide d'une porte complexe | 87 |
| Figure 4.9: Codage en double rail d'un digit | 89 |
| Figure 4.10: Opérateur logique Xor' en double rail implémentant un protocole 4 phases | 91 |
| Figure 4.11: Porte logique And en double rail | 91 |
| Figure 4.12: Architecture globale du cryptoprocasseur | 93 |
| Figure 4.13: Interface sur 1 bit | 94 |
| Figure 4.14 : Architecture du Bloc de chiffrement asynchrone | 95 |
| Figure 4.15: Layout et caractéristiques des circuits | 96 |
| Figure 4.16: Temps de chiffrement et courant consommé en fonction de la tension | 97 |
| Figure 4.17: Profils et spectres du courant des cryptoprocasseur asynchrone et synchrone | 98 |

| | |
|---|-----|
| Figure 4.18 : Courbes <i>DPA</i> réalisées en attaquant le bit 1 en sortie du <i>Xor32</i> du cryptoprocasseur asynchrone (V2)..... | 99 |
| Figure 4.19 : Nombre de courbes utilisées pour réaliser des attaques <i>DPA</i> sur tous les bits de sortie du bloc <i>XOR32</i> du cryptoprocasseur asynchrone (V2) lors de l'exécution de la première itération. .. | 100 |
| Figure 5.1: Netlist de portes <i>VHDL</i> d'une description de la porte <i>And</i> en double rail..... | 105 |
| Figure 5.2: Graphe orienté annoté G_{And} d'une porte <i>And</i> en double rail | 105 |
| Figure 5.3: Matrices du digraphe orienté G_{And} | 106 |
| Figure 5.4: Matrices et digraphes orientés de chacun des sommets des sorties de G_{And} | 108 |
| Figure 5.5: Présentation des notions de divergence et convergence en 'et' et en 'ou' | 109 |
| Figure 5.6: Matrices et digraphes de chacun des chemins d'exécutions du digraphe G_{And} | 110 |
| Figure 5.7: Digraphe de la porte <i>And</i> en double rail équilibré. | 112 |
| Figure 5.8: Signature électrique la porte <i>And</i> en double rail $C_{ij}=8$ fF. | 116 |
| Figure 5.9: Signatures électriques de la porte <i>And</i> en double rail. Les capacités de routage varient de $C_d=8$ fF à 32fF..... | 117 |
| Figure 6.1: Synoptique du flot de conception | 122 |
| Figure 6.2: Comparaison d'une implémentation d'une fonction ou exclusif en double rail à l'aide des portes standards et des portes complexes d'une bibliothèque dédiée aux circuits asynchrones. | 125 |
| Figure 6.3: Architecture globale du circuit <i>AES</i> | 127 |
| Figure 6.4: Le banc de registres | 128 |
| Figure 6.5: Algorithme de chiffrement du Rijndael..... | 128 |
| Figure 6.6: Ronde de l'algorithme | 129 |
| Figure 6.7: Architecture des blocs chiffrement des composants..... | 130 |
| Figure 6.8: Layouts et caractéristiques des circuits..... | 131 |
| Figure 6.9: Floorplan et Architectures détaillées des composants | 132 |
| Figure 6.10: Variations du courant et du temps de chiffrement en fonction de la tension d'alimentation. Profil du courant..... | 133 |
| Figure 6.11 : Profil de courant de l' <i>AES</i> 128 bits de clé à 0.5 volt, courbes de courant moyennée sur 128 et instantanée des premières rondes | 134 |
| Figure 6.12: Analyse des 256 courbes correspondant aux 256 valeurs de l'octet 15 du message d'entrée pour une clé fixe..... | 135 |
| Figure 6.13: Pic <i>DPA</i> du bit de poids fort en sortie de la <i>ByteSubs4</i> qui correspond au calcul de l'octet 15. 8192*128 courbes sont utilisées..... | 136 |
| Figure 6.14: Attaque par corrélation sur le bit 0 en sortie de la <i>ByteSubs4</i> . 1024*128 courbes utilisées. | 137 |
| Figure 7.1: Schéma d'implémentation et digraphe de la fonction <i>Xor</i> en double rail : Chemins d'exécutions identiques..... | 141 |

| | |
|---|-----|
| Figure 7.2: Transpositions des chemins de données dans une implémentation d'une fonction logique <i>Xor</i> en double rail. | 142 |
| Figure 7.3: Architecture du <i>DES</i> implémentant la méthode de transpositions des chemins de données... | 147 |
| Figure 7.4: Profil de courant du cryptoprocresseur <i>DES</i> | 149 |
| Figure 7.5: Signal <i>DPA</i> sur le bit 4 en sortie de la <i>SBOX1</i> sans activation de la contre-mesure..... | 151 |
| Figure 7.6: Signal <i>DPA</i> sur le bit 4 de <i>SBOX1</i> avec activation de la contre-mesure..... | 151 |
| Figure 7.7: Half-buffer à un bit implémentant un protocole 4 phases entre deux modules <i>QDI</i> | 152 |
| Figure 7.8: Implémentation du décalage aléatoire sur un signal d'acquittement | 154 |
| Figure 7.9: Architecture du <i>DES</i> et gestion des signaux d'acquittements..... | 157 |
| Figure 7.10: Profils des courants des différentes architectures | 160 |
| Figure 7.11: Spectres en courant des trois architectures..... | 160 |
| Figure 7.12: Profils de courant des premières itération du <i>DES</i> avec et sans activation du décalage des signaux d'acquittements | 162 |
| Figure 7.13: Signatures électrique lors de l'attaque du bit 4 en sortie de la <i>SBOX1</i> sur la première itération du <i>DES</i> | 163 |

LISTE DES TABLEAUX

| | |
|--|-----|
| Tableau 3.1 Variations du nombre de transitions logiques lors du chiffrement d'un <i>DES</i> en fonction du poids de Hamming des messages. Le poids de Hamming des messages est incrémenté en partant de 0 ($R=0$). Les messages et les clés sont pris aléatoirement. | 69 |
| Tableau 3.2 Variations des amplitudes et du temps d'occurrences des pics de courant en fonction des paramètres environnementaux d'un bloc logique. | 69 |
| Tableau 4.1 Suppression des paramètres n'influençant plus le profil de courant en logique asynchrone <i>QDI</i> | 92 |
| Tableau 4.2 Variations du nombre de transitions en fonction des données dans le circuit (V2). | 99 |
| Tableau 5.1 Paramètres influençant l'analyse différentielle en courant dans un circuit <i>QDI</i> | 118 |
| Tableau 6.1 Caractéristiques du composant 1. | 132 |
| Tableau 7.1 Tableau de permutation du premier bloc de substitution du <i>DES</i> (<i>SBOX1</i>) | 148 |
| Tableau 7.2 Réorganisation de la fonction Sbox1 du DES | 149 |

INTRODUCTION GÉNÉRALE : CONTEXTE ET MOTIVATIONS

Dans un contexte mondial d'ouverture, dans lequel l'omniprésence des nouvelles technologies d'information et de communication est sans équivoque, les notions de confidentialité, nécessaires pour rendre les informations indéniables et secrètes, sont devenues des enjeux majeurs auxquels doivent faire face tous fournisseurs de systèmes de communication sécurisées. La confidentialité des informations, transmises dans les réseaux ou contenues dans les terminaux de communication, est une exigence essentielle notamment dans l'industrie de la carte à puce où l'authentification, l'intégrité et le non désaveu des informations sont maîtres mots.

De nos jours, la conception de tels systèmes dits sûrs exige la prise en compte, tant au niveau logiciel que matériel, des nouvelles problématiques de sécurité récemment développées dans le domaine de la cryptanalyse matérielle. En effet, la cryptanalyse fait l'objet d'une attention particulière des concepteurs de systèmes sécurisés depuis l'apparition des attaques matérielles. Ces types d'attaques, classées en attaques intrusives et en attaques non intrusives, exploitent les faiblesses et les défauts de l'implémentation matérielle des circuits cryptographiques pour obtenir des informations confidentielles.

Contrairement aux attaques intrusives, les attaques non intrusives, à savoir les attaques temporelles, les attaques électromagnétiques, les attaques par injection de fautes et les attaques par analyse de courant, ne sont pas destructives. Encore appelées attaques par canaux cachés, elles représentent une nouvelle classe d'attaques dont le principe est basé d'une part sur l'analyse des corrélations entre les signaux compromettants des composants (signal d'alimentation, signal d'horloge etc.) et les données manipulées et d'autre part, sur l'analyse du comportement des circuits en présence de fautes.

Ces différents types d'attaques représentent de nouveaux défis auxquels les concepteurs doivent apporter des solutions. Pour cela, plusieurs contre-mesures ont été proposées allant de l'utilisation de capteurs divers (détection des variations de tension, du signal d'horloge, des variations de températures, de la lumière, etc.) à la modification des algorithmes cryptographiques (masquage, duplication etc.). A cela s'ajoute, l'introduction de bruits dans les composants et l'utilisation de générateurs aléatoires pour

supprimer ou rendre inexploitable les corrélations existantes entre les données traitées et les canaux cachés.

La prise en compte de tous ces paramètres lors de la conception font des circuits de véritables forteresses de sécurité pénalisant parfois les performances des systèmes et ne garantissant aucune fiabilité face aux nouvelles variantes de ces attaques. C'est dans ce contexte que nous proposons l'utilisation de la technologie asynchrone dont les propriétés intrinsèques (notamment des circuits quasi insensibles aux délais) offrent de manière globale un niveau de sécurité plus élevé que leur équivalent synchrone face à des attaques non intrusives. L'absence de signal d'horloge, la redondance des chemins logiques et l'usage d'un codage de données de type *1 parmi n* associé à un protocole de communication de type poignée de mains comprenant 4 phases dont une phase d'initialisation des données avant chaque calcul, sont des atouts incontestables pour résister aux attaques non intrusives.

Les objectifs dans ce domaine sont donc doubles, d'une part évaluer les technologies asynchrones face à des attaques non intrusives afin de définir une méthodologie de conception de circuits sécurisés et d'autre part, intégrer ces méthodes dans le flot de conception automatique des circuits asynchrones que nous développons dans le groupe *CIS (Concurrent Integrated Systems)* du laboratoire *TIMA (Laboratoire des Techniques de l'Informatique et de la Microélectronique pour l'Architecture des Ordinateurs)*.

En effet, la disponibilité d'outils automatisés apparaît naturellement essentiel pour l'exploitation et la valorisation des travaux sur l'amélioration de la sécurité par la technologie asynchrone. Parce que les circuits asynchrones ne peuvent être la cible des outils commerciaux, nous développons un environnement de conception spécifique aux circuits asynchrones pour la sécurité. Cet environnement que nous voulons inclure dans le flot de conception des circuits asynchrones *TAST* se veut être aussi compatible que possible avec les outils industriels standard.

TAST est l'acronyme de "*Tima Asynchronous Synthesis Tools*". C'est un environnement de conception principalement composé d'un compilateur-synthétiseur capable de cibler différents types de logiques asynchrones à partir d'une description de haut niveau exprimée dans un langage proche de *CSP (Communicating Sequential Processes)*. Après analyse, les programmes sont représentés en interne dans l'outil par des réseaux de Petri associés à des graphes de type flot de données. Ce type de représentation, largement utilisé pour représenter la spécification de circuits synchrones, est particulièrement adéquat dans le domaine des circuits asynchrones. Nous avons développé un langage propriétaire pour la modélisation et la synthèse des circuits asynchrones. Il est basé sur le langage *CHP (Communicating*

Hardware Processes) lui-même dérivé de *CSP*. Notre langage inclut en particulier la définition des protocoles de communication, la définition du codage des données, des choix déterministes et indéterministes, la gestion de la hiérarchie, et la génération de traces. Une forme synthétisable du langage, dénommée *DTL (Data Transfer Level)*, a été définie pour permettre la génération automatique de différents styles de circuits asynchrones : les circuits insensibles aux délais (*QDI*) et les circuits Micropipeline. *TAST* permet ainsi la génération de trois modèles différents à partir d'un programme *CHP*:

- Un modèle *VHDL* comportemental destiné à la simulation.
- Un modèle de circuit Micropipeline dont les chemins de données sont décrits en *VHDL RTL* et les contrôleurs par une netlist de portes en *VHDL*.
- Un modèle de circuit *QDI (Quasi Delay Insensitive)* décrit par un réseau de portes en *VHDL*.

La figure 1.1 présente les flots de conception supportés par l'environnement *TAST*.

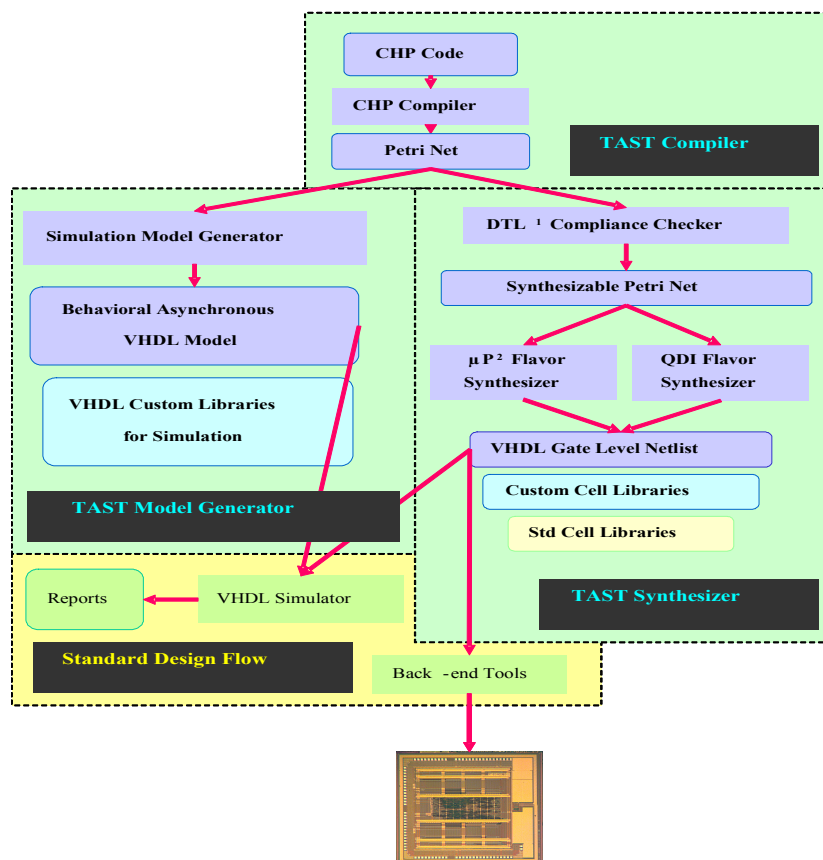


Figure 1 : Flots de conception supportés par l'environnement *TAST*.

Mais *TAST* n'est pas seulement un outil d'aide à la synthèse des circuits. Il permettra à l'avenir d'améliorer les performances des circuits dans le domaine de la surface [Mauri03], de la consommation [Slim04, Essa02], des émissions électromagnétiques [Pany04] et de la sécurité notamment contre les attaques non intrusives [Boue04c, Monn05, Rena04]. La figure 2 montre l'envergure et les possibilités d'aide à la conception offertes par *TAST*.

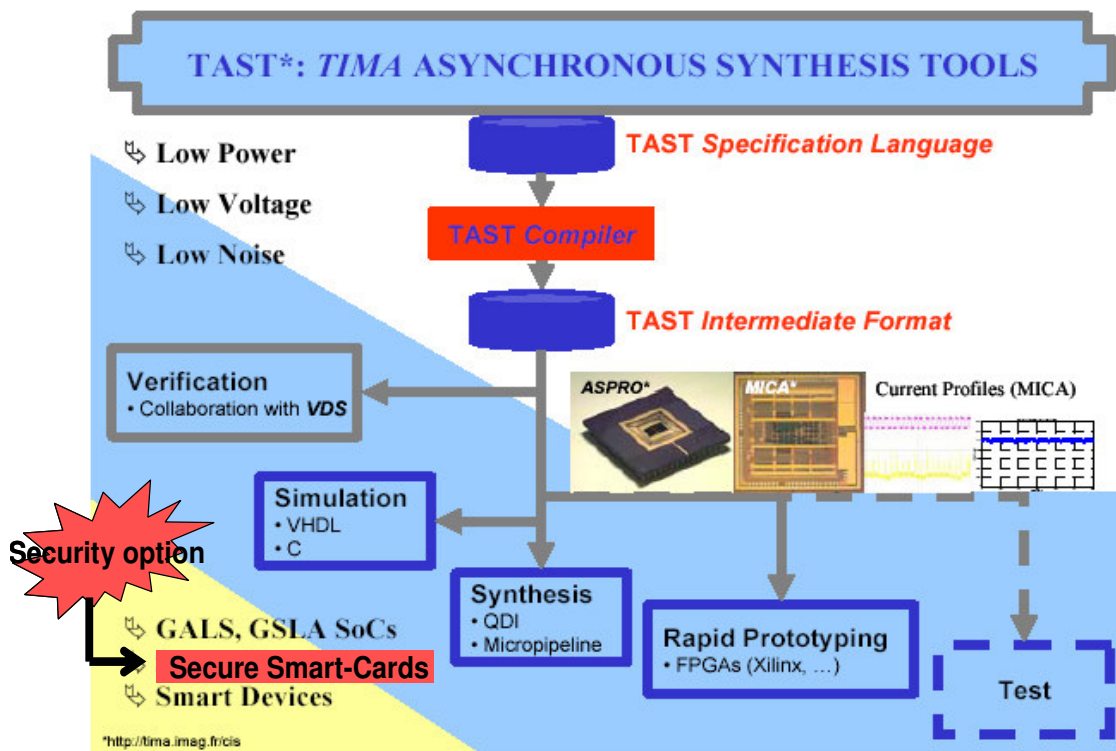


Figure 2 : Le projet *TAST*

Les travaux de recherche présentés dans ce manuscrit sont concentrés sur les options de sécurité du projet *TAST*. Afin d'atteindre ces objectifs, nous devons avant tout, analyser séparément chacune des attaques non intrusives sur les circuits asynchrones de manière à évaluer la sensibilité et la résistance des circuits asynchrones.

C'est dans ce contexte que ce travail de thèse s'est focalisé sur l'évaluation de la logique asynchrone face aux attaques en puissance qui sont parmi les attaques non intrusives les plus simples à mettre en œuvre et les plus efficaces. Dans ces attaques par analyse de courant, on inclut les attaques

électromagnétiques car elles sont le reflet dans le domaine fréquentiel de l'activité électrique d'un composant. La logique asynchrone offre un grand nombre d'alternatives pour la conception de circuits logiques. Nous nous proposons de les étudier dans le but de définir un style, voire plusieurs styles de circuits asynchrones qui améliorent sensiblement la résistance des circuits aux attaques par analyse différentielle de courant (*Differential Power Analysis*). La technologie asynchrone apparaît particulièrement prometteuse pour cette étude car elle permet la conception de circuits dont on peut contrôler finement l'activité électrique et donc la consommation dynamique et le profil de courant. Par ailleurs, elle s'avère très bien adaptée aux applications sans contact car les circuits asynchrones sont peu sensibles aux variations de la tension d'alimentation et fonctionnent à une vitesse auto régulée par l'énergie reçue.

Le premier chapitre du manuscrit présente donc les différents types d'attaques non intrusives existantes. Nous décrivons, dans ce chapitre, les attaques par canaux cachés en rappelant dans la première partie, quelques notions de base en matière de cryptographie et cryptanalyse.

Le second chapitre est focalisé sur les analyses formelles des attaques en puissance. Nous présentons dans ce chapitre le travail de formalisation que nous avons effectué pour étudier et analyser les fuites d'informations par canaux cachés sur les circuits intégrés. Cette étude débouche sur l'implémentation d'une attaque sur un cryptoprocresseur synchrone implémentant l'algorithme du *DES* (*Data Encryption Standard*).

Une analyse détaillée des fuites d'information dans les circuits *CMOS* est présentée dans le chapitre III ainsi que l'état de l'art des différentes contre-mesures proposées dans la littérature. Les résultats d'analyses expliquant la dépendance entre les données manipulées et le courant consommé dans les circuits synchrones sont présentés dans la première partie de ce chapitre. La seconde partie présente les effets des contre-mesures développés dans le domaine et introduit le choix de l'alternative asynchrone.

Le chapitre IV est consacré à l'évaluation de la technologie asynchrone. Les propriétés de faible consommation, de faibles pics de courant, de nombre réduit de pics de courant, d'une consommation indépendante du nombre de transitions logiques grâce à un codage de données de type *1 parmi n* et d'un séquençement de nature flot de données, des circuits asynchrones sont analysés. Les résultats de ces évaluations sont illustrés par la présentation d'attaques effectuées sur de prototypes des cryptoprocresseurs *DES* réalisés en logique asynchrone *QDI* avec pour référence le cryptoprocresseur du *DES* synchrone analysé au chapitre II.

Les fuites d'informations observées dans les circuits *QDI* qui permettent de réussir des attaques en courant, sont formellement analysées dans le chapitre V. Pour ce faire, un modèle formel de représentation des circuits *QDI* est présenté. Ce modèle a permis de modéliser le courant consommé par ce type de circuit et d'appliquer, sur ce modèle, une attaque en courant afin d'identifier l'origine de ces fuites d'information.

De ces analyses formelles de fuites d'information réalisées dans le chapitre V, on déduit dans le chapitre VI une méthodologie de conception des circuits asynchrones sécurisés. Ce flot, permet de contrôler à chaque phase de conception (niveaux logiques et électriques) la sensibilité des circuits à des analyses différentielles de courant. Ce flot a été validé par la réalisation de plusieurs prototypes de circuits asynchrones implémentant l'algorithme de Rijndael (*AES*).

Afin d'accroître la résistance intrinsèque des circuits *QDI*, nous proposons dans le chapitre VII de nouvelles méthodes de protection. Ces méthodes basées sur l'introduction de bruit sur le domaine temporel et spatial sont formellement analysées et les résultats obtenus, par simulations électriques, démontrent toute leur pertinence.

Enfin les différents points abordés dans ce manuscrit sont résumés dans une conclusion qui présente aussi les perspectives de ces travaux de thèse.

CHAPITRE I

CRYPTANALYSE MATERIELLE : ATTAQUES PAR CANAUX CACHES

1.1 Introduction

Longtemps resté un domaine réservé aux services secrets des gouvernements et des services d'espionnage (contre-espionnage) des militaires, le domaine de la cryptanalyse est devenue depuis les années 70, un domaine de recherche grand public porté par de nombreux laboratoires académiques.

En effet, le développement considérable des réseaux de communications a favorisé l'expansion de nouveaux moyens de paiements à distance avec notamment la carte à puce, le e-commerce, et le e-banking. Ces nouveaux types de communications nécessitent de plus en plus de moyens de transactions dits sûrs, pouvant résister à diverses attaques cryptanalytiques. De cette lutte effrénée entre d'une part, les cryptographes qui mettent en place des systèmes ou algorithmes cryptographiques et d'autre part, les cryptanalystes qui développent des techniques pour briser les systèmes de sécurité, un net avantage est donné au second depuis le développement de la cryptanalyse matérielle. Ces nouvelles méthodes de cryptanalyse dites attaques par canaux cachés ont montrées leur efficacité sur de nombreux produits commerciaux de type carte à puce.

Ce chapitre est consacré à l'analyse des attaques par canaux cachés. Nous allons dans la première partie introduire quelques notions de base de la cryptologie avant de présenter les différents domaines de la cryptanalyse et un état de l'art sur les différents types d'attaques matérielles existantes.

1.2 La Cryptologie : Notion de Base et terminologie

Etymologiquement composée des deux mots grecs ‘crypto’ (caché) et ‘logie’ (science), la cryptologie est une branche des mathématiques englobant aussi bien l’ensemble des procédés cryptographiques que celles des méthodes cryptanalytiques.

1.2.1 La Cryptographie

La cryptographie est un art et une science consacrée à la protection des données en les rendant incompréhensibles sauf pour son destinataire. Elle a pour objectif la conception et l’analyse des mécanismes permettant d’assurer l’*intégrité*, l’*authenticité*, la *confidentialité* et le *non désaveu* des données et des communications. Pour cela elle utilise des processus cryptographiques.

Un processus cryptographique est une transformation d’un message appelé *texte en clair* en un message incompréhensible appelé *texte chiffré*. Ce processus est nommé *chiffrement* (*cryptage* ou *encryption*) et lorsque le processus inverse est réalisé, on parle alors de *déchiffrement* (*décryptage*).

$$\left. \begin{array}{l} \text{Chiffrement} \quad E(M) = C \\ \text{Déchiffrement} \quad D(C) = M \end{array} \right\} D(E(M)) = M$$

E est la fonction de chiffrement,

D la fonction de déchiffrement,

C le texte chiffré et M le texte en clair.

Les fonctions de chiffrements ou de déchiffrements représentent des fonctions mathématiques encore appelées *Algorithmes cryptographiques* (*cryptosystèmes*). La sécurité d’un cryptosystème dépend de deux paramètres : la sûreté de l’algorithme et la longueur de la clé utilisée. On entend par sûreté le fait qu’il n’existe pas de meilleur moyen de casser l’algorithme que d’utiliser une *attaque exhaustive* (essais de toutes les clés possibles).

1.2.1.1 Notion de Clé cryptographique

Le concept de clé cryptographique a été créé pour augmenter la robustesse des algorithmes cryptographiques. Elle est généralement mixée avec les données d’entrées des algorithmes

cryptographiques par l'opérateur logique ou exclusif. Plus la longueur de la clé est importante, plus il est difficile de la briser par une attaque exhaustive. Pour une clé de 56 bits, il faut en moyenne $3,6 \times 10^{16}$ tentatives. En faisant l'hypothèse qu'un superordinateur peut essayer 1 million de clés par seconde, il faudra environ 1000 ans pour trouver la bonne clé. De nos jours, les cryptosystèmes utilisent au minimum des clés de 128 bits.

La notion de clé cryptographique permet de définir deux schémas de base des cryptosystèmes ou algorithmes cryptographiques. Les algorithmes à clé secrète ou *cryptage symétrique* et les algorithmes à clés publiques ou *cryptage asymétrique*.

1.2.2 Cryptage symétrique

Dans un système de cryptage symétrique, la même clé est aussi bien utilisée pour le chiffrement que pour le déchiffrement des messages. On parle alors de chiffrement à clé secrète. Parmi les cryptosystèmes à clé secrète les plus utilisés nous trouvons le **DES** (*Data Encryption Standard*) [FIPS46a].

Le système de chiffrement du **DES** a été élaboré dans les années 70 par IBM. Il a été adopté en 1977 par le gouvernement Américain comme standard de protection d'informations non secrètes mais confidentielles.

1.2.2.1 Le Standard de chiffrement de données : le **DES**

L'algorithme du **DES** est basée sur le principe des schémas de Feistel [Schn01]. En effet, les travaux de Feistel ont permis de construire des fonctions bijectives aléatoires utilisées dans la majorité des algorithmes de chiffrement à clé secrète. L'algorithme de chiffrement prend comme entrée des blocs de $2n$ bits qu'il partage en deux blocs de n bits (bloc de gauche **G** et bloc de droite **D**). Les images des blocs (**G,D**) dans la fonction de Feistel sont les blocs (**L,R**) telles que : $L = D$ et $R = G \text{ xor } f(D)$. Cette fonction est une fonction bijective, car le couple (**G,D**) est retrouvé par les relations suivantes : $D = L$ et $G = R \text{ xor } f(L)$.

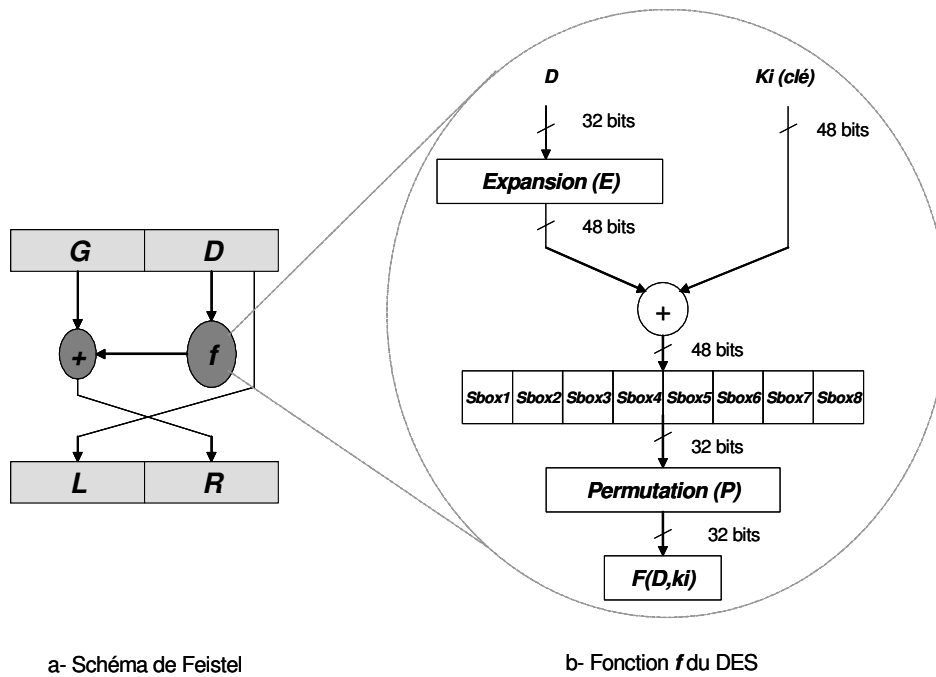


Figure 1.1 : Schéma de Feistel et la fonction f du Standard de chiffrement de données

Comme la partie droite n'est pas modifiée alors le schéma est exécuté plusieurs fois; on parle alors d'*itération* ou de *ronde*.

Le *DES* possède 16 itérations et permet de chiffrer (déchiffrer) des mots de 64 bits avec une clé effective de 56 bits (clé de 64 bits avec les bits de parité). À chaque ronde, le *DES* utilise une nouvelle sous clé calculée en fonction de la précédente clé. La fonction f du *DES* est constituée de 8 fonctions de substitutions ($SBOX_i$; i avec $1 \leq i \leq 8$) qui sont les seules fonctions surjectives de l'algorithme.

Afin d'améliorer la fiabilité du chiffrement du *DES* (clé trop faible), une variante du *DES* en *triple DES* a été adopté [FIPS46b]. L'algorithme reste inchangé sauf que le processus de chiffrement (déchiffrement) s'effectue avec deux clés de 64 bits selon le schéma suivant :

$$\text{Triple DES Chiffrement} : DES_{\text{Chiffrement}}^{Clé1} \rightarrow DES_{\text{Déchiffrement}}^{Clé2} \rightarrow DES_{\text{Chiffrement}}^{Clé1}$$

$$\text{Triple DES Déchiffrement} : DES_{\text{Déchiffrement}}^{Clé1} \rightarrow DES_{\text{Chiffrement}}^{Clé2} \rightarrow DES_{\text{Déchiffrement}}^{Clé1}$$

Depuis 2001, l'algorithme du **DES** est appelé à être remplacé par le nouveau standard de chiffrement avancé des données (**AES** pour *Advanced Encryption Standard*). Ce nouveau standard utilise l'algorithme de **Rijndael** qui n'est plus basé sur un modèle de Feistel mais sur des calculs dans un corps fini de Galois [FIPS197].

1.2.3 Cryptage Asymétrique

Dans le mode de cryptage asymétrique, deux clés sont utilisées, une clé publique et une clé secrète. On parle alors de chiffrement à clé publique. Ce type de chiffrement a été inventé en 1976 par W. Diffie et M. Hellman [Schn01] pour résoudre le problème de gestion des clés posés par les chiffrements symétriques. Dans ce type de chiffrement, tout le monde peut utiliser votre clé publique pour vous envoyer des messages chiffrés que seul vous pouvez déchiffrer avec votre clé secrète. Il offre également la possibilité de créer des signatures numériques.

Parmi les algorithmes asymétriques standards de type **DSS** (*Digital Signature Standard*) avec notamment l'algorithme de signature numérique **DSA** (*Digital Signature Algorithm*) et l'algorithme de signature numérique par courbes elliptiques **ECDSA** (*Elliptic Curve Digital Signature Algorithm*)[FIPS186], le **RSA** reste le plus populaire et le plus utilisé.

1.2.3.1 Le RSA

Le **RSA** a été inventé dans les années 70 par R. Rivest, A. Shamir et L. Adleman [FIPS186]. Sa sécurité est basée sur la difficulté de factoriser des grands nombres entiers en leurs produits de deux nombres premiers.

Soit p et q deux nombres premiers et d un entier tel que d soit premier avec $(p-1)*(q-1)$. Les nombres (p,q,d) représentent la clé secrète. La clé publique est composé du couple (n,e) telle que :

$$\begin{aligned} n &= p * q \\ e &= \frac{1}{d} \bmod((p-1).(q-1)) \end{aligned} \quad (1.1)$$

Si M est le message à chiffrer, alors le processus de chiffrement est réalisé en exécutant le calcul suivant :

$$C = M^e \bmod n \quad (1.2)$$

Pour déchiffrer le message chiffré C , il faut utiliser la clé secrète d tel que :

$$M = C^d \bmod n \quad (1.3)$$

Plus le nombre n est grand, plus il est difficile de retrouver ces facteurs premiers p et q . A ce jour le ***RSA*** est implémenté avec des valeurs de plus 1024 bits.

1.3 Définition de la Cryptanalyse

La cryptanalyse est le domaine de la cryptologie consacré à l'étude des méthodes ou techniques permettant de restituer un message chiffré en clair sans pour autant connaître la ou les clés utilisée(s). L'objectif de cette science est de pouvoir mettre en évidence les faiblesses d'un cryptosystème. On parle alors de tentatives de déchiffrement ou d'attaques cryptanalytiques.

Il existe cinq types d'attaques génériques parmi les quelles nous avons par ordre d'efficacité:

- l'attaque avec un texte chiffré : le cryptanalyste dispose du texte chiffré de plusieurs messages obtenus avec le même algorithme.
- l'attaque avec un texte en clair connu : le cryptanalyste dispose des textes chiffrés et des textes en clairs correspondants.
- l'attaque avec un texte en clair choisi : le cryptanalyste à accès aux textes chiffrés et aux textes en clair, mais il peut choisir les textes en clair à chiffrer.
- l'attaque avec un texte adaptif : le cryptanalyste peut choisir les textes en clair, mais il peut également adapter ses choix en fonction des textes chiffrés précédemment.
- l'attaque avec texte chiffré choisi : le cryptanalyste peut choisir différents textes chiffrés à déchiffrer.

Ces attaques reposent sur le principe de *Kerckhoff* [Stin96] stipulant que la connaissance complète et détaillée du cryptosystème n'est pas un secret pour le cryptanalyste. Ainsi, la sécurité et la robustesse de tout algorithme cryptographique ne repose pas sur la non connaissance de cet algorithme, mais essentiellement sur les propriétés mathématiques de l'algorithme et sur la longueur des clés utilisées.

De nos jours, la conception de tel système nécessite la collaboration de concepteurs de différents domaines (informatique, mathématique, microélectronique etc.) car le domaine de la cryptanalyse est un domaine très ouvert faisant appel à des compétences aussi large que possible. La figure 1 représente les classes et les sous classes des différentes méthodes cryptanalytiques en fonction des compétences qu'elle met en œuvre. Dans chacune de ces méthodes on peut utiliser chacun des cinq modèles d'attaques génériques définis ci-dessus.

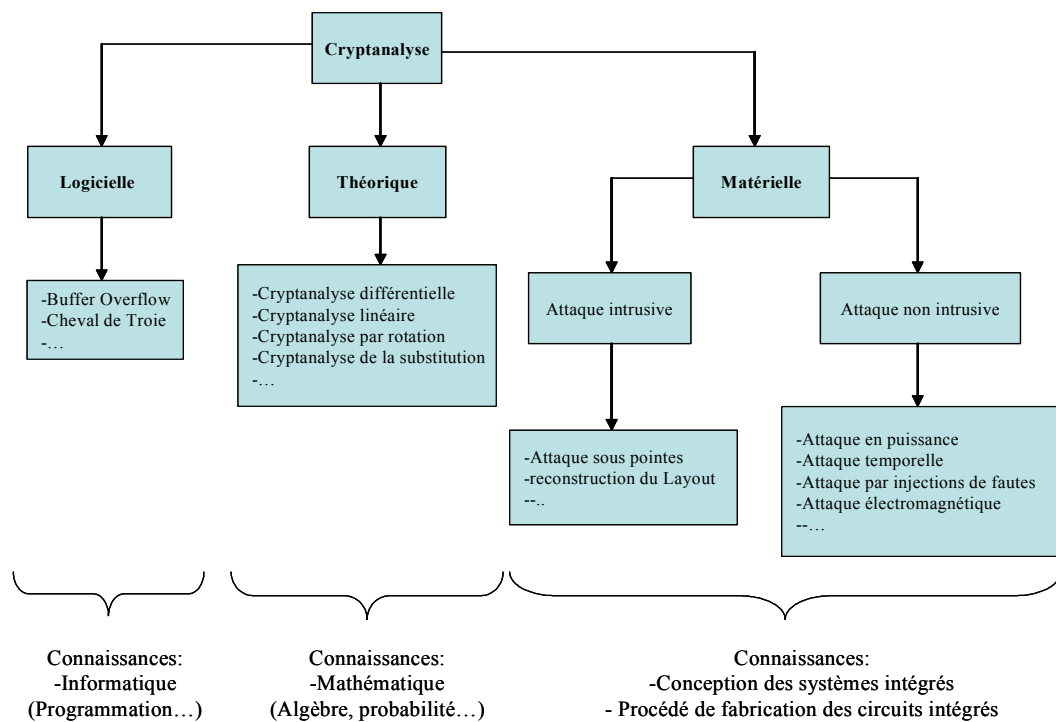


Figure 1.2 : Les attaques cryptanalytiques et les compétences requises

Il existe trois grandes familles d'attaques cryptanalytiques, les attaques dites théoriques, les attaques dites logicielles (Software) et les attaques dites matérielles (Hardware).

1.4 La cryptanalyse dite théorique

Les attaques dites théoriques sont basées sur des analyses utilisant des théories mathématiques pour briser des algorithmes cryptographiques. Elles exploitent les faiblesses de conception théorique des cryptosystèmes pour générer des attaques. Ces types d'attaques permettent aux cryptanalystes d'évaluer théoriquement la force ou la fiabilité d'un algorithme cryptographique.

Parmi les attaques les plus connues ont trouvé la cryptanalyse linéaire et la cryptanalyse différentielle [Schn01].

1.4.1 La cryptanalyse linéaire

Présentée pour la première fois en 1993 par Matsui [Mats94], elle est théoriquement l'une des attaques les plus redoutables sur l'algorithme du *DES*. La cryptanalyse linéaire est basée sur une modélisation linéaire de la fonction de chiffrement ou de déchiffrement. Cette modélisation est faite en réalisant des attaques en textes clairs connus. Le principe consiste à réaliser une approximation linéaire de la fonction cryptographique par des analyses statistiques des couples de texte clair et de texte chiffré avec une clé identique.

1.4.2 La cryptanalyse différentielle

Développée à la fin des années 80 par Eli Biham et Adi Shamir [Biha90], la cryptanalyse différentielle consiste à étudier les variations des sorties d'un algorithme cryptographique en fonction des variations des entrées. Le principe de l'attaque est basé sur une analyse des propagations des différences des données d'entrées sur les différences des données en sorties. Ce type d'attaque est très utilisée sur des algorithmes de chiffrement itératif par bloc comme c'est le cas du *DES* ou de l'*AES*.

En 1994 M. Hellman et K. Langford introduisent la cryptanalyse différentielle-linéaire qui est une combinaison des deux approches. Cette attaque permet de retrouver 10 bits de clé sur un *DES* à 8 rondes [Lang94].

1.5 La cryptanalyse dite logicielle

Elle regroupe toutes les techniques qui permettent d'accéder aux informations confidentielles d'un cryptosystème en exploitant les failles des systèmes d'exploitation des composants.

Les attaques bien connues comme le cheval de Troie ou le débordement de pile (pour ne citer que celles là) développées et utilisées sur les systèmes d'exploitations des ordinateurs personnels, ont montrés toutes leurs efficacités sur des systèmes d'exploitation moins complexes comme ceux embarqués dans des cartes à puce.

1.5.1 Le cheval de Troie

Tiré de la mythologie grecque et la fameuse guerre de Troie, le cheval de Troie est défini en informatique comme un programme exécutant des instructions ou opérations malicieuses à l'insu de l'utilisateur. L'idée est de cacher dans un programme principal des lignes de codes qui permettant d'ouvrir des fenêtres de vulnérabilité du système lorsque ce dernier est exécuté. Il permet soit de sortir des mots de passe, des copies des données sensibles ou d'exécuter toutes autres actions affaiblissant la sécurité du système [Ross01].

1.5.2 Attaque par débordement de pile (Buffer overflow)

Le principe de l'attaque par débordement de pile est d'écraser une partie des données de la pile d'un processeur (notamment le pointeur d'instructions) par du code dont l'exécution permettra de contrôler le système. L'écrasement des données d'une pile est réalisé en écrivant dans la pile plus de données qu'elle peut en contenir [Boue04]. L'attaque consiste donc à forcer dans un système d'exploitation, l'exécution de codes facilitant des accès aux zones sensibles du système (zones protégées, l'octroi du droit d'administration etc.). L'efficacité de cette attaque repose sur l'accès restreint sur le système qu'elle nécessite pour monter l'attaque, mais elle reste uniquement à la portée des programmeurs avertis.

1.6 La cryptanalyse matérielle

Cette branche de la cryptanalyse regroupe l'ensemble des techniques cryptanalytiques exploitant les vulnérabilités matérielles des circuits intégrés dédiés à des applications sécurisées. On distingue deux catégories d'attaques matérielles : les attaques intrusives et les attaques non intrusives.

1.6.1 Les attaques intrusives

Les attaques intrusives sont directement réalisées sur le silicium du composant. Elles commencent par un désassemblage de la puce de son boîtier. La puce ainsi mise à nue est attaquée par des solvants chimiques pour retirer les différentes couches de passivations, d'isolations (etc.) utilisées dans les procédés de fabrication des circuits intégrés. Ces accès physiques sur la puce, permettent soit de reconstituer le layout de la puce ou de réaliser des analyses sous pointes. Ces attaques sont destructives.

1.6.1.1 Attaque par reconstruction du layout

Ce type d'attaque a pour objectif l'identification et l'analyse des zones sensibles d'une puce par reconstitution de layout. Le layout de la puce est reconstruit en réalisant, dans le sens inverse, les différentes étapes des procédés de fabrication des circuits intégrés [Ross 96]. Ainsi les différentes couches de résines, des métaux, sont enlevées les unes après les autres par des solutions chimiques pour refaire le layout. Ce type d'attaque permet de lire par reconstitution les valeurs d'une mémoire [Samy 02]. Elle permet également d'analyser les mécanismes de sécurités implémentés dans une puce afin d'orienter et définir des schémas d'attaques (logicielles ou matérielles).

1.6.1.2 Attaque par analyse sous pointes

L'identification des zones sensibles par la méthode de reconstitution du layout permet de réaliser des analyses en utilisant des pointes de test. L'utilisation des sondes ou pointes généralement consacrées aux tests de wafer (test sous pointes) [Moor 00] permet la lecture des valeurs transitant sur des bus (données ou adresse) ou sur des entrées et sorties des cellules, bloc etc. A l'aide de ce type de matériel, on peut forcer des valeurs logiques sur certains nœuds de la puce afin de valider des instructions de test ou de comparaison donnant accès à des informations protégées. Dans certains composants, elle peut permettre de rétablir le fusible de programmation de la puce [Kömm 99, Gueu 98].

De manière générale, les attaques intrusives nécessitent des moyens considérables et coûteux notamment avec l'utilisation d'un microscope optique, du matériel sous pointes (*FIB : Focused Ion Beam*) ou même d'une salle blanche. Comme ces attaques sont destructives, elles nécessitent également au cryptanalyste de disposer de plusieurs composants.

Les attaques intrusives permettent de collecter des informations sur les systèmes de protection matérielle embarqués dans les puces. Selon les fiabilités ou défaillances observées dans ces mécanismes de protection, les cryptanalystes mettent alors en place des méthodes d'attaque moins coûteuses, plus simple et surtout non destructive pour briser les composants possédants les mêmes systèmes de protection.

1.6.2 Les Attaques non intrusives

Contrairement aux attaques intrusives, les attaques non intrusives ne sont pas destructives. Selon les méthodes d'analyses mises en place, les attaques non intrusives peuvent être rangées en deux classes : les attaques par injection de fautes et les attaques par analyses des signaux compromettants encore appelées attaques par canaux cachés.

1.6.2.1 Les Attaques par injections de fautes

Le principe des attaques par injections de fautes consiste à générer délibérément des fautes dans un composant en fonctionnement, dont l'exploitation (des fautes générées) permettra de briser les sécurités du composant. Les fautes sont générées en faisant fonctionner le composant dans des conditions anormales. L'implémentation des attaques par injections de fautes se déroule en deux phases : la génération de la faute et l'exploitation de la faute générée pour accéder aux données confidentielles du système.

1.6.2.1.1 Génération et modèles de fautes

Les fautes générées dans les composants sont produites par des modifications anormales des paramètres externes du circuit à savoir :

- variations intempestives des signaux d'alimentation.

- variations intempestives du signal d'horloge.
- variation de la température.
- exposition du composant à des champs de radiation.
- exposition du circuit à des faisceaux de lumière de différentes longueurs d'ondes.

Ces conditions anormales de fonctionnement permettent de produire plusieurs types de fautes. Quelque soit le phénomène mise en œuvre pour injecter la faute dans le circuit, on considère l'effet de la faute uniquement au niveau logique en se référant aux modèles de fautes développés dans le domaine du test. En effet, il existe plusieurs modèles de fautes définis et modélisés dans le domaine du test. Parmi ces modèles, les plus utilisés en cryptanalyse sont les fautes transitoires de type **SET** (*Single Event Transient*), **SEU** (*Single Event Upset*) et des fautes dites «soft error».

- **Les fautes transitoires de type SET** : ce type de faute entraîne une modification temporaire du niveau logique d'un signal. On parle alors de variation intempestive de la valeur logique du signal ou de «glitch». Quelque soit son origine, ce type de faute disparaît dans le temps.
- **Les fautes transitoires de type SEU** : elles correspondent au basculement ou à l'inversion d'un point mémoire.
- **Les fautes dites «soft error»** : Ce type de faute correspond à la mémorisation d'une faute transitoire de type **SET**. La propagation d'une faute **SET** peut conduire à la mémorisation d'un ou de plusieurs bits erronés. Lorsque cette configuration se produit, on obtient alors une erreur classiquement dénommée «soft error».

Chacun de ces modèles peuvent être étendus en fautes multiples en ajoutant l'attribut de multiplicité [Monn 04]. Les techniques d'injections de fautes et les modèles de fautes utilisées en cryptanalyse doivent être sans effet destructeur afin de permettre leurs exploitations.

1.6.2.1.2 Exploitation des fautes

La formalisation d'une attaque exploitant un modèle de fautes à été introduite pour la première fois en 1996 par Boneh, Demillo et Lipton avec l'attaque dite de «Bellcore» du nom de leur groupe de

recherche [Bone 97]. Cette attaque exploite à la fois un modèle de fautes et des faiblesses d'implémentation des algorithmes cryptographiques comme c'est le cas de l'implémentation du **RSA** en utilisant le théorème des restes chinois [Stin96].

En effet, nous avons vu dans le paragraphe 1.2.3.1 que l'algorithme du **RSA**, qui est l'algorithme à clé publique le plus utilisé pour des communications ou transactions sécurisées, utilise un modulo n de 1024 voir 2048 bits. Pour garantir la sûreté de l'information, il est nécessaire que n soit suffisamment grand pour que sa factorisation en p et q soit calculatoirement impossible. Une telle implémentation est coûteuse en temps de calcul. Afin d'optimiser la vitesse d'exécution, des algorithmes adaptés aux calculs des produits modulaires sur des grands nombres sont utilisés avec notamment l'algorithme de Montgomery [Mont 85] [Kak 96, Jeon 97] et l'implémentation par le théorème des restes chinois [PCKS1]. Cette dernière implémentation est particulièrement vulnérable aux attaques par injection de fautes.

*** Rappel sur le théorème des restes chinois:**

Soient m_1, m_2, \dots, m_n des entiers naturels supérieurs à 2, premiers entre eux deux à deux, et a_1, a_2, \dots, a_n des entiers. Le système d'équation est :

$$\begin{aligned} x &= a_1 \bmod m_1 \\ x &= a_2 \bmod m_2 \\ &\dots\dots \\ x &= a_n \bmod m_n \end{aligned} \tag{1.4}$$

Il admet une solution unique modulo $M = \prod_i^n m_i$ de la forme :

$$x = \sum_i^n a_i M_i y_i \bmod M$$

$$\text{avec } M_i = \frac{M}{m_i} \text{ et } y_i = M_i^{-1} \bmod m_i \text{ pour } 1 \leq i \leq n \tag{1.5}$$

L'application de ce théorème sur **RSA** permet de déduire l'implémentation suivante : on calcule dans un premier temps les valeurs intermédiaires S_p et S_q telles que :

$$\begin{aligned} S_p &= m^{d_p} \bmod p \\ S_q &= m^{d_q} \bmod q \end{aligned} \quad \text{avec} \quad \begin{cases} d_p = d \bmod (p-1) \\ d_q = d \bmod (q-1) \end{cases} \quad (1.6)$$

Le résultat de chiffrement S (ou déchiffrement) est obtenu par l'expression:

$$\begin{aligned} S &= (a.S_p + b.S_q) \bmod n \\ \text{avec } a &= \frac{n}{p} (q^{-1} \bmod p) \quad \text{et } b = \frac{n}{q} (p^{-1} \bmod q) \end{aligned} \quad (1.7)$$

L'apparition d'une faute sur le calcul de S_p ou sur celui de S_q permet de déterminer les nombres premiers p et q de n . Soit S_p^* la valeur obtenue avec l'erreur. Le résultat de chiffrement (déchiffrement) avec l'erreur est donné par :

$$S^* = (a.S_p^* + b.S_q) \bmod n \quad (1.8)$$

On déterminant le *PGCD* de n et S^*-S , on trouve la valeur de q :

$$PGCD(S^* - S; n) = q \quad (1.9)$$

En effet,

$$\begin{aligned} S^* - S &= a.S_p^* + b.S_q - a.S_p - b.S_q = a.(S_p^* - S_p) \\ S^* - S &= (q).(q^{-1} \bmod p).(S_p^* - S_p) \end{aligned}$$

Comme les termes $(q^{-1} \bmod p)$ et $(S_p^* - S_p)$ sont inférieures à q et que $n=pq$, alors le Plus Grand Commun Diviseur de n et S^*-S est égal à q .

Il existe plusieurs variantes de cette attaque qui permettent de briser la majorité des cryptosystèmes à clé publique (*PKCs* pour Public Key Cryptosystems) [Bao 97, Mahe 97, Biha 97, lens 97]. La plus proche du modèle de Bellcore est celui de Lenstra [lens 97]. Dans ce modèle seule la connaissance du message à chiffrer M et du résultat S^* sont nécessaires pour trouver les facteurs de n .

Des modèles d'attaques sur des protocoles cryptographiques symétriques (à clé secrète) ont été également développés par des cryptanalystes. Biham et Shamir ont développés en partant du principe du modèle de Bellcore l'attaque par analyse différentielle de fautes encore appelée **DFA** (Differential Fault Analysis) [Biha 97]. L'attaque **DFA** consiste à analyser les différences entre deux résultats de chiffrement (ou déchiffrement) S (correct) et S^* (incorrecte par apparition d'une erreur) en utilisant le même message d'entrée et la même clé. En simulant des fautes dans une implémentation du **DES**, ils ont démontrés qu'il est possible de retrouver toutes les clés de l'algorithme en utilisant 50 à 200 messages chiffrés. Une seconde approche consiste à utiliser les propriétés des mémoires non volatiles. L'idée est de changer la valeur (les valeurs) d'un bit d'une mémoire (ROM ou registre) à l'aide d'un laser. Par exemple, lors de la dernière itération du **DES** le cryptanalyste peut forcer à zéro certains bit du registre de gauche (L) et déterminer les valeurs d'entrées et de sorties de la fonction f du **DES** [Biha 97].

Toutefois, ces attaques font toutes une hypothèse sur le modèle et la localisation précise de l'apparition de l'erreur. C'est le cas de la **DFA** appliquée sur le **DES** dans laquelle la faute doit apparaître sur les dernières itérations (14, 15 et 16) pour retrouver la clé ou encore du **RSA** en **CRT** (Chinese Remained Theorem) dans laquelle, l'erreur ne doit apparaître que dans l'un des calculs intermédiaires de S_p ou S_q .

Des modèles d'injection de fautes plus simples et moins complexes dans leurs mises en œuvre ont été exploitées par Ross et Kuhn dans [Ross 96]. Des variations brusques (intempestives) du signal d'horloge, l'exposition à des champs électromagnétiques ou à des champs de radiation, des variations intempestives du signal d'alimentation, provoquent un fonctionnement anormal du circuit autorisant des accès à des données confidentielles. On parle alors d'attaques par «Glitch». Ces types d'attaques sont particulièrement redoutables sur des instructions de saut, sur des compteurs, des boucles, des tests par comparaisons et des branchements sur conditions.

1.6.2.2 Les attaques par analyses des signaux compromettants

Les attaques par canaux cachés sont basées sur la recherche et l'exploitation de toute corrélation entre les données manipulées par un circuit et ces signaux externes. Ces signaux, sont soit les signaux d'alimentation, les signaux d'horloge, soit les signaux électromagnétiques d'un circuit. Ils sont appelés signaux compromettants ou canaux cachés du fait qu'ils peuvent faire fuir des informations indésirables.

En effet, la notion de canal est définie comme un endroit par lequel transite de l'information, et il est caché quand son utilisation est détournée de son fonctionnement initial.

Les attaques utilisant des signaux compromettants et notamment les signaux électromagnétiques sont bien connues des militaires et des agences d'espionnages depuis la seconde guerre mondiale [Ross01] avant de devenir un domaine de recherche académique publique dans les années 90.

De nos jours, les études sur des corrélations entre les données traitées par un système cryptographique et les signaux compromettants ont donné lieu à de nombreuses méthodes ou techniques d'attaques. En fonction du signal compromettant utilisé, on les classe en attaque temporelle, en attaque électromagnétique et en attaque en puissance.

Nous allons dans cette partie décrire chacune de ces attaques afin d'introduire les attaques en puissance qui seront développées au chapitre 2.

1.6.2.2.1 Les attaques temporelles

Les attaques temporelles exploitent les dépendances temporelles entre les données manipulées et le temps nécessaire au calcul des données. Le principe de l'attaque est de déterminer toute corrélation entre le temps d'exécution d'une fonction (en terme de cycle d'horloge) et les données traitées.

Deux phases sont nécessaires à la réalisation de l'attaque. Dans la première phase, on détermine des corrélations entre les données et le temps de calcul. Cette analyse peut être faite sur deux niveaux d'abstraction : Au niveau architectural (schéma d'implémentation) et au niveau algorithmique. Lorsque des corrélations sont observées, elles sont analysées dans la seconde phase afin d'examiner si elles sont exploitables pour retrouver des informations confidentielles.

Certain choix d'implémentation d'algorithme cryptographique sont particulièrement vulnérables à ce type d'attaque comme c'est le cas de l'implémentation par la méthode binaire du calcul de l'exponentielle modulaire utilisée dans le ***RSA***.

En effet, le calcul de chiffrement (déchiffrement) ***RSA*** qui est de la forme $S = M^e \bmod n$ (cf.§ 1.2.3.1) peut être implémenté par l'algorithme suivant :

Soit l le nombre de bit de la clé e :

$$e = (e_0, e_2, \dots, e_{i-2}, e_{i-1}).$$

$$0 \leq i \leq l-1 \quad e_i \in \{0,1\}.$$

Le calcul de S par la méthode binaire s'effectue comme suit:

1. $S \leftarrow 1$
 2. pour $i = l-1$ jusqu'à 0 fais
 3. $S = S^2 \bmod n$
 4. si $e_i = 1$ alors $S = S.M \bmod n$
 5. sortir (S)
- (1.10)

Dans ce type d'implémentation, le temps de calcul de chaque itération est directement dépendent des valeurs des bits de la clé e . Selon la valeur du bit e_i , l'instruction de la ligne 4 (**L4**) introduira ou non un temps de calcul supplémentaire dans chaque itération. Ainsi une simple analyse du temps d'exécution de chaque itération permet de deviner la clé [Koch 96, Dhem 98].

Une méthode d'attaque exploitant ce type de dépendance à été introduite dans [Koch 96] par Paul Kocher. Cette approche est basée sur le calcul de la variance du temps d'exécution de chaque itération.

Soit $T = e_r + \sum_{i=0}^{l-1} t_i$ la somme des temps d'exécution de chaque itération t_i et de e_r , une variable aléatoire de l'erreur de mesure du temps. Soit b le nombre de bits à déterminer, en soustrayant à T les valeurs des temps de calculs correspond aux b bits on obtient l'expression $e_r + \sum_{i=0}^{l-1} t_i - \sum_{i=0}^{b-1} t_i = e_r + \sum_{i=b}^{l-1} t_i$ dont la variance est définie par :

$$Var(e_r) + (w - b)Var(t). \quad (1.11)$$

La valeur de la variance tendra vers sa valeur minimale lorsque les bits devinés seront corrects, sinon elle tendra vers sa valeur maximale. Les propriétés de la variance définie pour la quantification de la

dispersion des échantillons autour de leur moyenne sont ainsi utilisées pour la détermination des bits de la clé.

Les algorithmes possédant des fonctions de test, de branchement, de saut tant sur le plan logiciel (ligne d'instructions) que matériel (architecture de conception) dépendant des données, sont vulnérables aux analyses temporelles. D'autres modèles d'attaque sur le *DES* et sur l'*AES* ont été proposés respectivement dans [Hevi 99] et dans [Koeu 99]. Le premier définit une méthode d'attaque qui permet de déterminer le poids de Hamming des bits de clé du *DES* et le second présente une attaque sur la fonction *Mixcolumns* de l'*AES* et plus particulièrement sur la sous fonction *Xtime* [FIPS197]. La connaissance du poids de Hamming p d'une clé de n bits permet de réduire une attaque par recherche exhaustive à A_n^p

tentatives avec : $A_n^p = \frac{n!}{(n-p)! p!}$

1.6.2.2.2 Les attaques électromagnétiques

Les possibilités de capture et d'analyse des émanations électromagnétiques des systèmes électroniques (écran de télévision, ordinateurs etc.) sont bien connues des scientifiques depuis les années 50. Les recherches dans le domaine afin de diminuer les interférences électromagnétiques des systèmes électroniques sont restés longtemps classées secret défense par les grandes nations. Il a fallu attendre les années 85 avec la publication des travaux de Win van Eck [Eck85] sur les conséquences des interférences des appareils électroniques sur la sécurité des systèmes pour porter le sujet dans le milieu académique. Dans ses travaux, il démontre la possibilité de reconstruire un signal vidéo par analyse et décodage de interférences électromagnétiques émis par récepteur vidéo.

De nos jours, les effets d'interférence des composants électroniques sur les appareils électroniques sont standardisés. Une directive de compatibilité électromagnétique (*CEM* ou *EMC* en anglais pour ElectroMagnetic Compatibility) à été rendu obligatoire depuis le 1^{er} janvier 1996 [Dire89]. Elle permet de caractériser l'aptitude d'un composant ou d'un équipement à fonctionner de manière satisfaisante dans un environnement et sans y occasionner lui-même des perturbations gênantes. Le terme de *CEM* regroupe deux concepts :

- L'*émissivité* (*EMI* pour *ElectroMagnetic Interference*) qui caractérise les émissions électromagnétiques produit par un appareil électronique dans un environnement.

- L'*immunité* (*EMS ElectroMagnetic Susceptibility*) qui caractérise la susceptibilité électromagnétique d'un appareil électronique dans un environnement.

Les études et les analyses des propriétés électromagnétiques ont permis aux cryptanalystes de développer des techniques d'attaques exploitant le rayonnement électromagnétique des circuits intégrés pour briser des codes cryptographiques.

* les attaques électromagnétiques sur des circuits intégrés

Les attaques par analyses électromagnétiques sur les composants dédiés à des applications cryptographiques ont connues un développement considérable depuis la publication il y a quelques années par la *NSA (National Security Agency)* de rapports scientifiques classés secrets défense sur la cryptanalyse matérielle par effets électromagnétiques [NSA00].

La majorité des circuits numériques sont cadencés par un signal global d'horloge qui impose une synchronisation des traitements dans toutes les parties du circuit. Durant chaque cycle d'horloge (front montant ou descendant), tous les signaux sont mémorisés dans des registres (ou mémoires) et des nouveaux calculs sont alors exécutés entraînant des appels en courant non négligeables. Dans ce type de circuit, le spectre électromagnétique est essentiellement dû à l'activité électrique des éléments logiques et des points mémoires constituant le circuit. Ces éléments (logiques et mémoires) sont constitués de transistors fonctionnant comme des commutateurs commandés par une tension de grille. Lorsque la tension de grille est présente, un transfert de charge entre le drain et la source s'effectue à travers le canal du transistor. Ce transfert de charge a pour effet de favoriser un appel en courant et d'engendrer une émission électromagnétique.

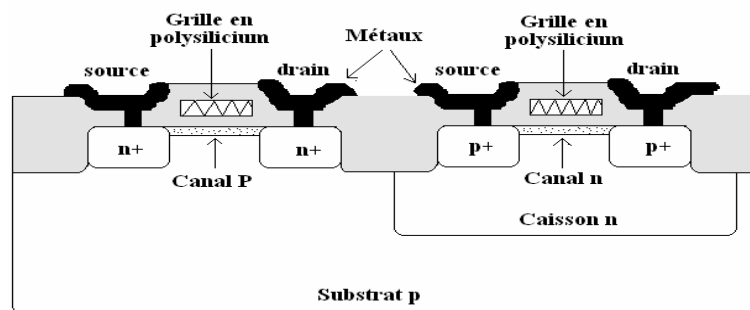


Figure 1.3 : Vue en coupe des transistors *CMOS* de type *N* et *P* sur un substrat

L'activité électrique des circuits intégrés est due aux transferts de charges lors de la commutation des éléments logiques de 0 vers 1 et de 1 vers 0. Plus le niveau d'intégration des transistors augmente, impliquant des fréquences de fonctionnement de plus en plus élevées et des délais de portes de plus en plus faibles, plus les variations du signal d'alimentation croissent augmentant ainsi les émissions électromagnétiques [Pany04].

Les caractéristiques du champ électromagnétique (en terme de fréquence et amplitude) sont donc relatives à l'activité électrique et à la nature des données manipulées dans le circuit. Par conséquence, le principe de l'attaque électromagnétique est d'exploiter cette dépendance afin de déterminer toute corrélation entre les données traitées et les émissions électromagnétiques.

* Mise en œuvre des attaques

La mise en œuvre des attaques électromagnétiques commence par des mesures du champ électromagnétique du circuit. Pour cela, les modèles développés dans le domaine de la **CEM** sont généralement utilisés [Pany04]. Deux types de mesures peuvent être réalisées : des mesures d'émissions conduites ou des mesures d'émissions rayonnées.

- Les mesures d'**émissions conduites** sont effectuées sur les signaux d'alimentation des composants. En terme de cryptanalyse, ces analyses sont quasi identiques aux analyses réalisées sur les signaux d'alimentation. Ici, le signal de courant est remplacé par son spectre électromagnétique.
- Les mesures **rayonnées** permettent d'obtenir plus de précision et sont plus efficaces pour des analyses de corrélations. Elles permettent d'établir une cartographie de l'activité électromagnétique des circuits aidant ainsi à identifier les zones les plus sensibles du circuit sur lesquelles seront focalisées les attaques. Pour cela, des capteurs ou sondes électromagnétiques munis d'inductance sont utilisés et selon leurs caractéristiques (taille de la sonde, bande passante, sensibilité), ils peuvent mesurer l'activité électromagnétique d'une dizaine de portes de la puce [Gand01].

Cette précision dans les analyses, rend ce type d'attaque particulièrement redoutable sur des instructions de branchement sur conditions, de saut, de test aux accès mémoires ou sur des bus de données (d'adresse) en déterminant leurs poids de Hamming.

Des méthodes d'attaques simples ou différentielles par analyses électromagnétiques ont été introduites pour la première fois par Quisquater sous les termes de **SEMA** (Simple Electromagnetic Attack) et de **DEMA** (Differential Electromagnetic Attack) [Quis00]. Ces attaques utilisent les mêmes algorithmes d'analyse que ceux utilisés par des attaques en puissances (**SPA** : Single Power Analysis et **DPA** Differential Power Analysis), sauf que les analyses sont faites sur des courbes d'émissions électromagnétiques. Rao et Rohatgi ont montrés dans [Rao01] que dans bien des cas, la **SEMA** s'avérait bien plus efficace que la **SPA** pour la détermination des valeurs traitées dans les instructions de test, de décalage ou de branchement sur condition. Par contre la **DEMA** reste aussi redoutable que la **DPA**. Toutefois, certaines implémentations sécurisées contre la **SPA** ou la **DPA** ne le sont pas forcément contre la **SEMA** ou la **DEMA** et vice versa.

1.6.2.2.3 Les attaques en puissance ou attaques par analyse du courant

Contrairement aux attaques électromagnétiques qui exploitent l'activité électrique des circuits intégrés sous forme d'émissions électromagnétiques, les attaques en puissances exploitent l'activité électrique des composant en mesurant leur profil de courant. Elles sont focalisées sur la recherche de corrélation entre les données manipulées et les caractéristiques des signaux d'alimentation (**Vdd** ou **Gnd**) du circuit (variations en amplitudes, pics de consommation, décalage temporel, puissance etc.). Une fois des corrélations observées, des méthodes d'analyses sont mises en œuvre afin d'exploiter ces dépendances pour briser le circuit. Les bases des attaques en puissance reposent sur l'hypothèse suivante :

En technologie CMOS la puissance consommée par un élément logique (porte) est différente selon qu'elle charge ou décharge sa capacité de sortie.

$$P_{charge} \neq P_{décharge} \Rightarrow i_{charge} \neq i_{décharge}$$

Ces différences qui sont directement liées à la nature des données manipulées sont exploitées pour établir des corrélations entre les courbes de courant et les données. Plusieurs techniques et méthodes ont été développées en vue d'exploiter cette dépendance pour être appliquée avec succès sur la majeure partie des cryptosystèmes. Ces méthodes permettent de retrouver directement les clés cryptographiques.

Les attaques par analyses du courant sont parmi les attaques matérielles, les plus efficaces, les plus faciles à mettre en œuvre et les moins coûteuses. Elles sont à la portée de tout laboratoire possédant un oscilloscope, une chaîne d'acquisition du courant et un ordinateur de bureau.

1.7 Conclusion

De nos jours, la conception des systèmes intégrés sécurisés nécessite non seulement l'utilisation d'algorithmes cryptographiques résistant à la cryptanalyse logicielle (cryptanalyse linéaire, différentielle etc.) mais également l'utilisation de méthodes d'implémentation matérielle dites robustes contre les attaques par canaux cachés. Le domaine de la conception de tel système requiert une étroite collaboration entre concepteurs de divers domaines, allant des mathématiques à l'informatique en passant actuellement par la microélectronique avec les attaques par canaux cachés.

Nous avons dans ce chapitre, présenté les différentes attaques cryptanalytiques menaçant la sécurité des systèmes intégrés en fonction des domaines et des compétences mises en œuvres. Parmi les différentes attaques matérielles présentées dans ce chapitre, les attaques en puissance sont parmi les plus efficaces et les plus faciles à implémenter. Nous allons dans ces travaux de recherche nous focaliser uniquement sur ces attaques en puissance afin de pouvoir d'une part évaluer la technologie asynchrone face à ce type d'attaque et d'autre part de proposer des contre-mesures.

CHAPITRE II

LES ATTAQUES EN PUISSANCE : ETUDE ET ANALYSE

2.1 Introduction

Les premières notes techniques proposant l'exploitation des variations des signaux d'alimentation en cryptanalyse furent introduites dans les années 90 par Paul Kocher [Kocher 99]. Parmi les méthodes proposées par ce dernier, on notera les attaques par simple analyse du courant (*SPA* pour *Single Power Analysis*) et les attaques par analyse différentielle du courant (*DPA* pour *Differential Power Analysis*). Exceptées les différences liées aux méthodes d'analyse et de détermination des informations confidentielles mises en place dans chacune de ces approches, ces deux types d'attaques sont basées sur des analyses de corrélation entre les données manipulées et le courant consommé.

Nous allons dans ce chapitre présenter le travail de formalisation effectué pour analyser les attaques utilisant les fuites d'information au niveau du courant. Les différentes méthodes développées pour exploiter ces fuites d'informations sont exposées avec notamment l'illustration des résultats d'une attaque différentielle sur un cryptoprocresseur synchrone implémentant l'algorithme du *DES*.

2.2 Mesures du courant

Cette phase correspond à la phase d'acquisition des courbes de courant. Les vecteurs utilisés pour les mesures sont choisis ou aléatoires. Le schéma de la figure 2.1 présente la plate forme utilisée pour effectuer et acquérir automatiquement les mesures de courant. Les mesures sont généralement réalisées

sur la masse en effectuant une mesure différentielle aux bornes d’une résistance placée entre la masse du composant et celle du système d’acquisition. Le même type de mesure peut être fait sur le signal V_{dd} du composant, dans ce cas, ce signal doit être isolé du reste du système pour limiter des effets parasites de bruit.

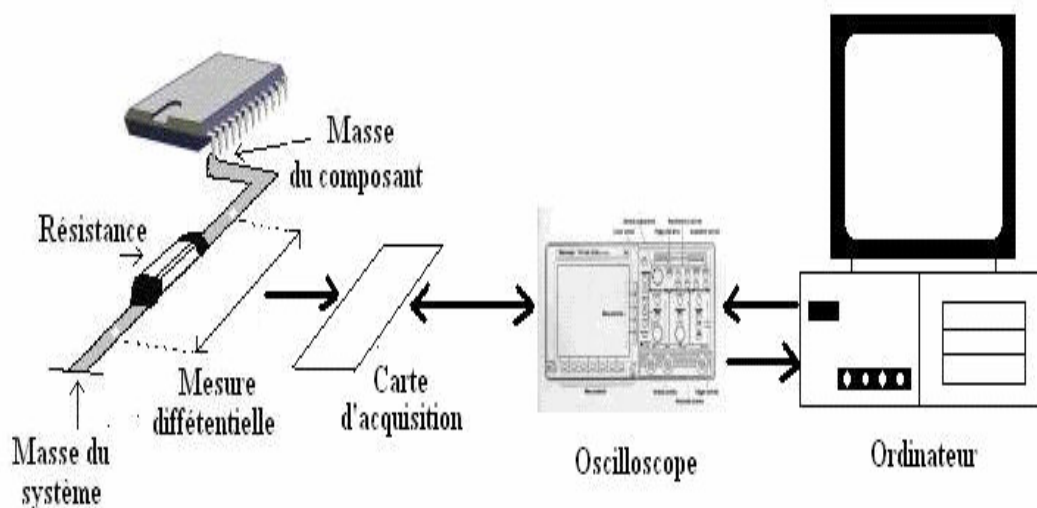


Figure 2.1 : Schéma de mesure et d’acquisition automatique des courbes de courant

Trois facteurs sont essentiels dans les mesures du courant pour la réalisation d’une attaque en puissance : l’amplitude, le nombre de pics de courant relatifs à l’activité électrique du composant et le rapport signal sur bruit.

Afin d’améliorer la qualité du signal du courant (amplitude et nombre de pics), une résistance de faible valeur est utilisée (inférieure à 10 ohms). Plus la résistance est faible, plus les amplitudes et le nombre de pics de courant sont importants et observables.

$$\begin{array}{ll}
 I = \frac{V}{R} \rightarrow \infty & I = \frac{V}{R} \rightarrow 0 \\
 R \rightarrow 0 & R \rightarrow \infty
 \end{array} \tag{2.1}$$

Les effets de la valeur de la résistance sont illustrés sur les figures 2.2. Les mesures réalisées avec des résistances inférieures ou égales à 2 ohms (figure 2.2-a et figure 2.2-c) présentent plus de détails (en termes de nombre et d’amplitude de pics de courant) que celles réalisées à l’aide des résistances supérieures à 10 ohms (figure 2.2-b et figure 2.2-d). Une seconde approche consiste à faire varier le signal

d'alimentation notamment en réduisant la tension d'alimentation du circuit. Une faible alimentation du circuit entraîne une augmentation des délais, ce qui permet d'accroître les pics relatifs à l'activité électrique des fonctions du circuit. Toutefois, la réduction de la tension d'alimentation rapproche le niveau du courant de celui du bruit. Le rapport signal sur bruit est amélioré en utilisant l'approche par moyennage des courbes. En effet, le rapport signal sur bruit est proportionnel à la racine carrée du nombre de courbes mesurées (N):

$$SNR = \sqrt{N} \frac{S_{Signal}}{\sigma_{bruit}} \quad (2.2)$$

σ_{bruit} est la variance du bruit

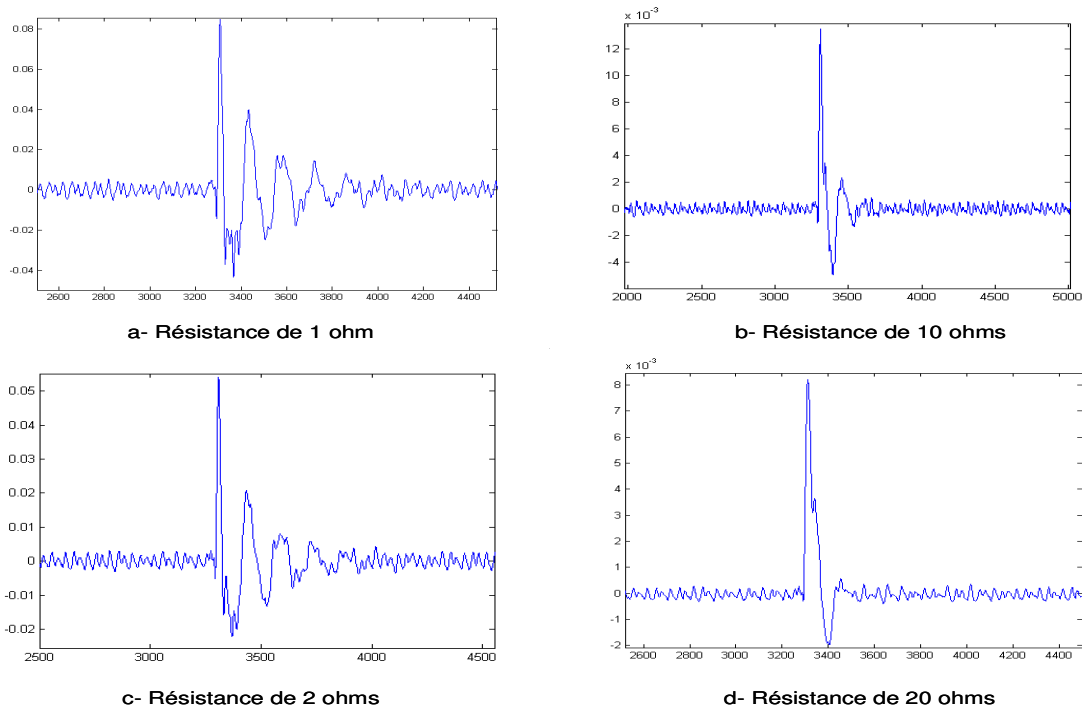


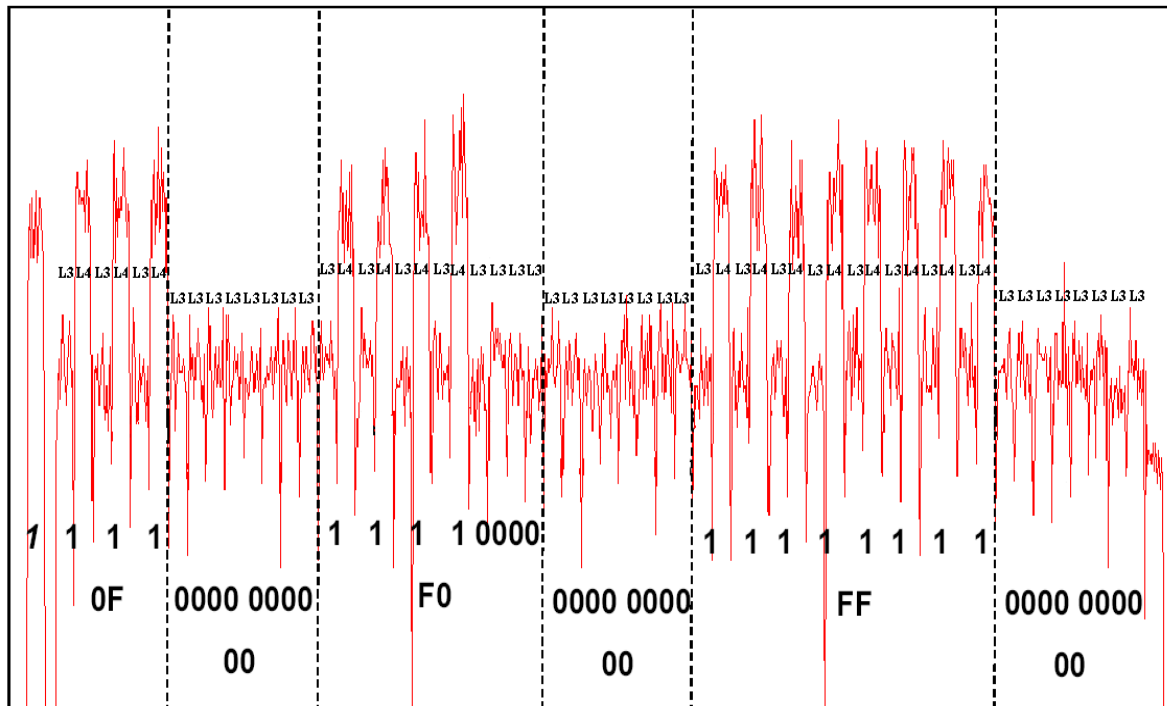
Figure 2.2 : Mesures différentielles de la première itération de l'algorithme du DES sur un cryptoprocresseur synchrone. Courbes de courant moyennées par 100.

Ainsi, la mise en place d'un flot d'acquisition automatique des courbes de courant doit permettre de mesurer avec le maximum de précision possible, les amplitudes des pics de courant nécessaires à la réussite des attaques en puissance. Pour cela, des filtres de bruit et des amplificateurs d'instrumentation peuvent être rajoutés dans la chaîne d'acquisition.

2.3 Attaque par simple analyse du courant

Cette attaque permet par une simple analyse du courant, d'établir des corrélations entre le profil de la forme d'onde du courant et les données manipulées. Selon l'algorithme et le type d'implémentation utilisés, elle permet de retrouver des clés secrètes, ou d'identifier la consommation (pic de courant) de blocs particuliers, d'instructions et de déterminer le poids de Hamming des bus et des registres (données et adresses).

Différents types d'implémentation des algorithmes cryptographiques sont vulnérables aux attaques par analyse du courant ou à ses variantes. Les blocs de génération de sous clés des cryptosystèmes symétriques (*DES* et *AES*) sont particulièrement exposés à ce type d'attaque, car les analyses sont directement effectuées sur les valeurs de la clé. Les articles [Biha99, Man02] présentent des méthodes qui permettent de réussir des attaques sur le bloc de génération des sous clés de l'*AES*. Mais, l'exemple le plus probant est l'implémentation naïve du *RSA* (méthode binaire) définie dans le chapitre I (cf. §1.6.2.2.1), utilisant des instructions conditionnelles dépendantes des bits de clé. En effet, lorsque le test de la quatrième ligne (*L4*) est réussi, le profil de courant pendant l'itération est marqué par un appel en courant supplémentaire dû à l'exécution de la multiplication modulaire $S.M \bmod n$. Dans le cas contraire, le profil de courant de l'itération est uniquement composé de l'appel de courant dû au calcul du carré $S^2 \bmod n$ de la troisième ligne (*L3*). Les mesures de courant d'une telle implémentation ont été réalisées par GEMPLUS [GEM01]. La figure 2.3 représente le profil de courant obtenu. Durant chaque itération, la consommation du courant due à l'instruction de multiplication de la ligne 4 (*L4*) est facilement identifiable comparé aux autres itérations dans lesquelles cette instruction n'est pas exécutées.



Clé évaluée : 0F 00 F0 00 FF 00

L3: Instruction de ligne 3: $S=S^2 \bmod n$

L4: Instructions de la ligne 4: $S=S.M \bmod n$

Figure 2.3 : Courbe de courant réalisée par *GEMPLUS* sur une implémentation *RSA* utilisant la méthode binaire présentée au chapitre I (cf. § 1.6.2.2.1).

2.4 Attaque par analyse différentielle du courant

Plus redoutable et plus efficace que les attaques temporelles et les attaques par simple analyse du courant, les attaques par analyse différentielle du courant sont basées sur des analyses statistiques des courbes de courant mesurées pour un grand nombre de valeurs avec la même clé. Elle exploite astucieusement à la fois les dépendances temporelles et électriques observées dans les attaques temporelles et dans les analyses du courant. Sa mise en œuvre se déroule en trois phases. La première phase est consacrée à l'analyse théorique de l'algorithme et de son implémentation. La seconde phase correspond à la phase de collecte d'information (notamment des courbes de courant) et la dernière phase consiste à analyser les informations collectées pour déterminer la clé.

2.4.1 Analyse théorique de l'algorithme : la fonction de sélection

Dans l'objectif de simplifier et d'accroître l'efficacité des analyses, l'attaque par analyse différentielle est réalisée séparément sur des fonctions de l'algorithme dépendants d'une partie de la clé k (ou de ses sous clés) et des données m . Les fonctions ainsi définies sont nommées fonction de sélection D . Le but de cette phase est donc de pouvoir déterminer par analyse de l'algorithme et de son implémentation les fonctions de sélections D pouvant être utilisées pour réaliser les attaques. Elle est définie par l'expression suivante :

$$D : (m, k) \mapsto \sum_{i=0}^{n-1} d_i 2^i \quad (2.3)$$

Avec n le nombre de bit en sortie de la fonction D et d_i les bits de sorties.

La fonction de sélection doit posséder les propriétés suivantes :

- La valeur de k (le nombre de bits de k), utilisée dans la fonction de sélection, doit être la plus petite possible. Cette propriété permettra de réaliser des analyses exhaustives sur toutes les valeurs possibles de clé k tant que k reste faible (voir la notion de clé dans le §1.2.1.1).
- elle doit permettre de définir une fonction (critère) de répartition des sorties en fonction des entrées. Soit $E_{entrées}$ et $S_{sorties}$ respectivement les ensembles de valeurs d'entrées et de sorties de la fonction D . Soit E_k l'ensemble de toutes les valeurs possibles de k . La fonction D possède une fonction de répartition si et seulement si pour tout k appartenant à E_k il existe une fonction de répartition f des sorties qui permet de classer les éléments de l'ensemble $E_{entrées}$ en des sous ensembles Sf_n distincts. Cette répartition est unique pour chacune des valeurs de la clé k de D . Le choix de la fonction de répartition dépend de la fonction de sélection D . L'idée est de pouvoir mettre en évidence des différences de consommation lorsque un ou plusieurs bits en sortie de D commutent (de 0 vers 1 ou de 1 vers 0). Elle peut être définie par le poids de Hamming des bits de sortie de D ou simplement par la valeur prise par un ou plusieurs bits de sortie.

Voici, en guise d'exemple, quelques fonctions de sélection que l'on peut définir dans l'algorithme du **DES** :

Fonction de sélection 1 :

$$D(m, k) = \sum_{i=0}^3 d_i 2^i = SBOX1(m \oplus k) \quad (2.4)$$

La **SBOX1** correspond à la première fonction de substitution du **DES**, elle dépend de 6 bits de la clé et de 6 bits de données et sa sortie est représentée sur 4 bits. La taille des clés manipulées (6 bits) permet de calculer pour chacune des valeurs de clé k , et pour tous les d_i , les ensembles de répartition Sf_k . Les propriétés de non linéarités des fonctions de substitutions de l'algorithme du **DES** sont particulièrement intéressantes pour garantir le critère de répartition. En effet, ces fonctions sont conçues de manière à ce qu'il ne puisse y avoir aucune corrélation entre les bits d'entrées et les bits de sorties dans chacune des fonctions [Copp94]. Ainsi en prenant la valeur du bit d_i (avec $i=0$) comme fonction de répartition $f(d_i) = d_i$ on définit les 2 sous ensembles $S0_{kd_0}$ et $S1_{kd_1}$ contenant respectivement l'ensemble des valeurs d'entrées m qui mettent le bit d_0 à 0 et à 1 en utilisant la clé k .

$$\begin{aligned} S0_{kd_0} &= \{m \in E_{entrées} / d_0 = SBOX1(m \oplus k) = 0\} \\ S1_{kd_1} &= \{m \in E_{entrées} / d_0 = SBOX1(m \oplus k) = 1\} \end{aligned} \quad (2.5)$$

Lorsque les valeurs de plusieurs bits en sortie de la fonction de sélection sont considérées, on définit les 3 sous ensembles $S0_{kd_i..d_j}$, $S1_{kd_i..d_j}$ et $S2_{kd_i..d_j}$ tels que :

$$\begin{aligned} S0_{kd_i..d_j} &= \{m \in E_{entrées} / d_0 = .. = d_j = SBOX1(m \oplus k) = 0\} \\ S1_{kd_i..d_j} &= \{m \in E_{entrées} / d_0 = .. = d_j = SBOX1(m \oplus k) = 1\} \\ S2_{kd_i..d_j} &= \{m \in E_{entrées} \text{ et } m \notin S0_{kd_i..d_j}, S1_{kd_i..d_j}\} \end{aligned} \quad (2.6)$$

j représente le nombre de bits considérés dans la fonction de répartition $j \leq 4$.

Thomas Messerges a démontré dans [Mess02] l'intérêt de ce type de répartition sur le rapport signal sur bruit lors du calcul du signal **DPA**.

D'une manière générale, les fonctions conçues pour résister à la cryptanalyse linéaire et différentielle sont particulièrement adaptées aux critères des fonctions de sélection [Guil04]. Afin de les rendre résistants face à ces attaques, les bits de sortie de telles fonctions doivent être indépendants entre

eux, aux bits des entrées et doivent être uniformément répartie en fonction de toutes les valeurs possibles des sorties. Ces propriétés permettent d'obtenir des ensembles de répartition unique par clé et contenant un nombre d'élément égal.

Fonction de sélection 2 :

$$D(m, k) = \sum_{i=0}^5 d_i 2^i = XOR_f(m, k) \quad (2.7)$$

Cette fonction correspond à un ou exclusif entre 6 bits de données et 6 bits de la clé. La fonction de répartition peut être définie en prenant le poids de Hamming des bits de sortie. En effet, pour une telle fonction (ou exclusif) le choix du poids de Hamming permet de garantir pour chacune des clés k , une répartition unique des valeurs des entrées en des sous ensembles Sf_k . Soit f la fonction de répartition telle que :

$$f = H(d_0, d_1, d_2, d_3, d_4, d_5) \quad (2.8)$$

H représente le poids de Hamming des bits de sortie et f est comprise entre 0 et 6.

Les différents sous ensembles Sf_k sont définies par les relations suivantes :

$$\begin{aligned} S0_k &= \{m \in E_{entrées} / f = 0\} \\ S1_k &= \{m \in E_{entrées} / f = 1\} \\ S2_k &= \{m \in E_{entrées} / f = 2\} \\ S3_k &= \{m \in E_{entrées} / f = 3\} \\ S4_k &= \{m \in E_{entrées} / f = 4\} \\ S5_k &= \{m \in E_{entrées} / f = 5\} \\ S6_k &= \{m \in E_{entrées} / f = 6\} \end{aligned} \quad (2.9)$$

Soit m_x une valeur d'entrée de la fonction D ($m_x \in E_{entrées}$) et d_x le résultat du ou exclusif entre m_x et la clé k ($d_x = xor(m_x, k)$). En faisant varier k de sorte qu'il prenne toutes les valeurs possibles de E_k (64 valeurs possibles), le poids de Hamming de d_x variera sur l'ensemble des valeurs possibles de f (entre 0 et 6). Ainsi la valeur m_x sera répartie en fonction de k sur l'un des sous ensemble Sf_k correspondant à une valeur du poids de Hamming de H .

$$\begin{aligned}
& m_x \text{ fixe ; pour tout } k \in E_K \\
& \Rightarrow 0 \leq H(d_x) \leq 6 \text{ avec } d_x = m_x \oplus k \\
& \Rightarrow m_x \text{ appartiendra à chacun des sous ensembles } Sf_k (f_k = H(d_x))
\end{aligned}$$

En réalisant une extension de cette analyse sur chacune des valeurs d'entrées, on démontre l'unicité des sous ensembles Sf_k obtenus pour chacune des clés k .

Cette unicité n'est toujours pas vérifiée pour certaine fonction de répartition. En considérant la fonction de sélection précédente, le choix de la valeur d'un bit en sortie de D comme fonction de répartition, ne remplit pas la condition d'unicité.

En considérant par exemple la valeur du bit d_0 , et les sous ensembles $S0_k$ et $S1_k$ tels que $S0_k$ regroupe toutes les valeurs d'entrées qui mettent le bit d_0 à 0 et $S1_k$ celles qui mettent le bit d_0 à 1 pour une valeur de k donnée, on vérifie assez facilement que quelques soient les valeurs de k pour lesquelles le premier bit de k (k_0) reste inchangé (32 valeurs possibles), les sous ensembles $S0_k$ et $S1_k$ restent identiques. Mieux encore, pour les valeurs de k dont le bit 0 est inversé, il suffit d'inverser les éléments des deux sous ensembles pour obtenir les nouveaux sous ensembles.

$$\begin{aligned}
\forall k = \sum_{i=0}^5 k_i 2^i + k_0 2^0 ; \text{ avec } k_0 = 0 & \Rightarrow \begin{cases} S0_k = S0_0 \\ S1_k = S1_0 \end{cases} \\
\forall k = \sum_{i=0}^5 k_i 2^i + k_0 2^0 ; \text{ avec } k_0 = 1 & \Rightarrow \begin{cases} S1_k = S0_0 \\ S0_k = S1_0 \end{cases}
\end{aligned} \tag{2.10}$$

Des fonctions de sélections identiques à celles présentées pour le **DES** sont également définissables sur biens d'autres algorithmes (**AES**).

Ces répartitions sont utilisées pour établir, par la suite, des analyses de corrélation entre les courbes de courant correspondant à l'activité électrique de la fonction sélectionnée et les éléments (messages) des sous ensembles formés. A la notion de fonction de répartition, s'ajoute celle du temps d'occurrence de la fonction de sélection D lors de l'exécution de l'algorithme. Cette notion dépend des algorithmes et permet de définir les points de synchronisation pour la phase d'acquisition des courbes de courant.

2.4.2 Collecte d'information : mesure et mémorisation des courbes de courant

Cette phase est consacrée à la mesure et à la mémorisation des courbes de courant. Dans un premier temps, on identifie par une analyse *SPA* la zone dans laquelle la fonction de sélection est exécutée. Cette identification permet de limiter l'intervalle de la zone d'acquisition des courbes de courant.

Soit n le nombre de messages M_i considérés (messages aléatoires ou choisis). Les messages M_i sont numérotés de 0 à $n-1$. On mesure et mémorise pour une clé identique K et pour chacun des n messages, les courbes de courant $C_i(t)$. $C_i(t)$ correspond à la courbe de courant de la fonction de sélection D obtenue en exécutant le i^{eme} message M_i au temps t . Les données ainsi collectées sont mémorisées sous forme de tableau ou de matrice T_D :

$$T_D = \begin{array}{c|cccc} & N^{\circ} & Messages & Courant & Résultat \\ & i & M_i & C_i(t) & C_i \\ \hline & 0 & M_0 & C_0(t) & C_0 \\ & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & \cdot \\ & n-1 & M_{n-1} & C_{n-1}(t) & C_{n-1} \end{array}$$

En tenant compte des propriétés de la fonction de sélection et surtout de l'instant auquel elle est exécutée dans le calcul du chiffrement (déchiffrement), les textes en clair M_i ou les textes chiffrés C_i sont utilisés pour l'analyse des corrélations. Pour des raisons de simplicité, les répartitions sont effectuées sur les textes en clair lorsque la fonction de sélection est exécutée au début du chiffrement et sur les textes chiffrés lorsqu'elle est exécutée en fin de calcul.

2.4.3 Analyse des corrélations

Dans cette phase, on exploite les dépendances existantes entre les données manipulées et le profil de courant du circuit. Nous avons, dans les étapes précédentes, créé des sous ensembles Sf_k regroupant selon le critère de répartition de la fonction de sélection D , les messages M_i en fonction des valeurs de la clé k . Nous avons mesuré et mémorisé sous forme de tableau T_D pour chacun des messages M_i , les courbes de courant $C_i(t)$ relative à l'activité de la fonction D . L'objectif de cette phase est donc de déterminer des fuites d'information suffisamment importantes pour retrouver la clé [Mess99].

Pour chacun des sous ensembles créé Sf_k , on calcule la moyenne de courant M_{Sf_k} en associant, à chacun des messages M_i des sous ensembles Sf_k , et sa courbe de courant correspondante $C_i(t)$. En considérant n_{Sf_k} comme le nombre d'éléments du sous ensemble Sf_k , la moyenne de courant est exprimée par :

$$\text{Pour tout } M_i \in Sf_k ; M_{Sf_k} = \frac{1}{n_{Sf_k}} \sum_{i=0}^{n_{Sf_k}-1} C_i(t) \quad (2.11)$$

Le signal de biais de l'analyse différentielle S_{DPAk} (le signal DPA) est obtenu en soustrayant l'une des moyennes M_{Sf_k} à la moyenne globale M_c de toutes les courbes de courant.

$$M_c = \frac{1}{n} \sum_{i=0}^{n-1} C_i(t) \quad (2.12)$$

$$S_{DPAk} = M_c - M_{Sf_k}$$

La détermination de la clé est obtenue en analysant les courbes S_{DPAk} . La bonne clé k correspond à la courbe DPA qui présentera des pics de courant importants. S'il n'apparaît aucune courbe possédant des pics de courant important on parlera alors d'échec de l'attaque.

En créant les sous ensembles Sf_k en fonction de la fonction de répartition $f(d_i)$ (qui peut être, soit la valeur d'un bit soit le poids de Hamming des bits de sorties de D), on met en évidence toutes les dissymétries électriques dues à la commutation (charge ou décharge) des bits en sortie de D . Elle correspond à toutes les commutations des portes logiques de la fonction D qui contribuent à la mise à zéro ou à un des bits d_i .

Soient la fonction de sélection D et la fonction de répartition $f(d_0) = d_0$ telle que pour une clé k donnée l'on ait :

$$S0_k = \{M_i \in E_{entrées} / f(d_0) = 0\}$$

$$S1_k = \{M_i \in E_{entrées} / f(d_0) = 1\} \quad (2.13)$$

Les sous ensembles $S0_k$ et $S1_k$ possèdent respectivement n_{0_k} et n_{1_k} éléments. Soit $q_{i0}(t)$ la contribution en courant du bit d_0 dans $C_i(t)$ lorsqu'il est mis à 0 et $q_{i1}(t)$ celle due à sa mise à 1. $Q_i(t)$ est la quantité de courant représentant l'activité électrique des autres bits du composant excepté celle du bit d_0 telle que : $q_{i0}(t) + Q_i(t) = C_i(t)$. Si $Q_i(t)$ est entièrement indépendant de $q_{i0}(t)$, alors on vérifie la relation suivante :

$$C_i(t) - q_{i0}(t) = C_i(t) - q_{i1}(t) = Q_i(t) \quad (2.14)$$

Lorsque la valeur de k est correcte (correspond à la valeur utilisée pour mesurer les quantités $q_{i0}(t)$ et $q_{i1}(t)$), les expressions des moyennes M_{Sf_k} de courant peuvent être définies par :

$$\begin{aligned} M_{S0_k} &= \frac{1}{n_{0_k}} \sum_{i=0}^{n_{0_k}-1} q_{i0}(t) + Q_i(t) \\ M_{S1_k} &= \frac{1}{n_{1_k}} \sum_{i=0}^{n_{1_k}-1} q_{i1}(t) + Q_i(t) \end{aligned} \quad (2.15)$$

Le signal DPA est obtenu par l'équation suivante :

$$\begin{aligned} S_{DPAk} &= M_c - M_{S0k} \text{ avec } M_c = \frac{1}{n} \left(\sum_{i=0}^{n_{0_k}-1} q_{i0}(t) + \sum_{i=0}^{n_{1_k}-1} q_{i1}(t) \right) \\ n &= n_{0_k} + n_{1_k} \end{aligned} \quad (2.16)$$

Si $n_{0_k} = n_{1_k}$ ce qui est souvent le cas lorsqu'en fait des attaques en messages choisis sur les fonctions de substitution du DES , on obtient l'expression suivante du signal de DPA :

$$S_{DPAk} = \frac{1}{2} (M_{S1k} - M_{S0k}) \quad (2.17)$$

A travers cette équation, on exprime l'amplitude maximale que peut prendre le signal DPA lorsque la clé est correcte et que cette différence est uniquement due aux dissymétries introduites lors de la commutation à 0 ou à 1 du bit attaqué.

Dans le cas où la clé k utilisée dans la fonction D est incorrecte (ne correspond pas à la clé utilisée pour la mesure des courbes de courant $C_i(t)$), et lorsque la même fonction de répartition $f(d_\theta)$ (identique à la précédente) les moyennes de courant correspondantes aux messages des sous ensembles $S0_k$ et SI_k sont alors définies par les expressions suivantes :

$$\begin{aligned} M_{S0_k} &= \frac{1}{n_{0_k}} \left(\sum_{i=0}^{a-1} q_{i0}(t) + \sum_{i=0}^{b-1} q_{i1}(t) \right) \\ M_{SI_k} &= \frac{1}{n_{1_k}} \left(\sum_{i=0}^{c-1} q_{i0}(t) + \sum_{i=0}^{d-1} q_{i1}(t) \right) \\ M_c &= \frac{1}{n} \left(\sum_{i=0}^{c-1} q_{i0}(t) + \sum_{i=0}^{d-1} q_{i1}(t) + \sum_{i=0}^{a-1} q_{i0}(t) + \sum_{i=0}^{b-1} q_{i1}(t) \right) \\ &\text{avec } n_{0_k} = a + b ; n_{1_k} = c + d ; n = n_{0_k} + n_{1_k} \end{aligned} \quad (2.18)$$

Les valeurs de a et c correspondent au nombre de fois la quantité $q_{i0}(t)$ est représentée respectivement dans $S0_k$ et dans SI_k . Celles de b et d correspondent par contre au nombre de fois la quantité $q_{i1}(t)$ est représentée respectivement $S0_k$ et dans SI_k . Expriment l'expression du signal de DPA :

$$\begin{aligned} S_{DPA} &= M_c - M_{S0_k} = \frac{1}{n} (a \cdot q_{i0}(t) + b \cdot q_{i1}(t) + c \cdot q_{i0}(t) + d \cdot q_{i1}(t)) - \frac{1}{n_{0_k}} (a \cdot q_{i0}(t) + b \cdot q_{i1}(t)) \\ &\text{avec } x \cdot q_{ij} \cong \sum_{i=0}^{x-1} x \cdot q_{ij}(t) \text{ avec } : j \in \{0;1\} \text{ et } x \in \{a,b,c,d\} \end{aligned} \quad (2.19)$$

On obtient après factorisation l'expression suivante :

$$S_{DPA} = q_{i0}(t) \left(\frac{a+c}{n} - \frac{a}{n_{0_k}} \right) + q_{i1}(t) \left(\frac{b+d}{n} - \frac{b}{n_{0_k}} \right) \quad (2.20)$$

En prenant le cas idéal avec $a=b=c=d$, le signal de DPA tends vers zéro.

$$\text{Si } a = b = c = d \Rightarrow S_{DPAk} \rightarrow 0 \quad (2.21)$$

Toutefois, comme la valeur de n est prise très grande pour améliorer le rapport signal sur bruit (cf. § 2.2) alors on déduit que :

$$\begin{aligned} S_{DPAK} &\longrightarrow 0 \\ n &\longrightarrow \infty \end{aligned} \quad (2.22)$$

Ces résultats illustrent la nécessité dans la réalisation d'une attaque *DPA* de considérer un nombre important de messages car elle permet d'une part, d'améliorer le rapport signal sur bruit et d'autre part, elle permet de réduire les pics *DPA* lorsque la clé choisie est incorrecte [Coro00]. En effet, dans certains cas, des clés incorrectes génèrent des pics de courant plus importants que ceux générés par la bonne clé. On parle alors de pics fantômes. Lorsque certains bits de m_i de la fonction de sélection D sont calculés en parallèle dans une autre fonction D' (figure 2.4), certaines fausses hypothèses de clé peuvent générer des pics *DPA* plus importants que ceux générés par la bonne clé.

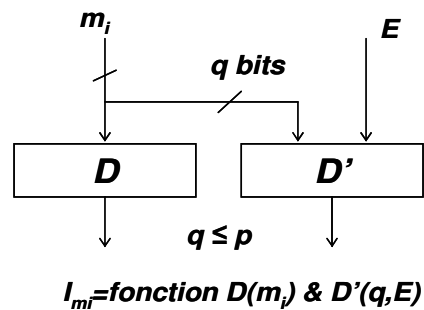


Figure 2.4: Analyse des dépendances du courant consommé dans une architecture en fonction des valeurs d'entrées m_i

Le courant correspondant au calcul de m_i dépend à la fois de la fonction D et des consommations induites par les q bits de m_i qui sont exécutés dans D' . Cela correspond au fait que le courant $Q_i(t)$ n'est pas entièrement indépendant du bit attaqué ce qui peut entraîner une réduction des pics du signal *DPA*.

Ces effets dépendent des algorithmes analysés, de la fonction de sélection définie et des messages utilisés. Des attaques avec des messages choisis sont dans la plupart des cas recommandés pour supprimer ce type d'effets [Beva02].

Le principe de l'attaque par analyse différentielle est donc de deviner la clé correcte en s'aidant des analyses des courbes de courant. Pour cela, on définit un ensemble de vecteurs de tests (messages aléatoires ou choisis) et on mesure pour chacun de ces vecteurs une courbe de courant correspondant à l'exécution de l'algorithme de chiffrement (déchiffrement) avec la clé correcte à déterminer. On définit par la suite les fonctions ou blocs à analyser pour retrouver successivement différentes parties de la clé : se

sont les fonctions de sélections. Pour une fonction de sélection donnée, on réalise une hypothèse sur la clé et on effectue le tri des courbes de courant en fonction d'un critère de répartition. Ces tris sont effectués sur les messages auxquels on associe les courbes de courant correspondantes. La fonction de répartition permettant de réaliser ces tris, doit pouvoir mettre en évidence des dissymétries électriques lorsque la bonne clé est choisie. Une fois les tris effectués, on calcule la moyenne en courant de chacun de ces ensembles triés. La bonne clé est déterminée en réalisant des comparaisons différentielles des différentes moyennes : si la clé choisie est correcte, la différence entre les moyennes n'est pas nulle, sinon elle tendra vers zéro. Dans ce cas, on refait une nouvelle hypothèse de clé jusqu'à la détermination de la bonne clé. Une optimisation de l'attaque consiste à effectuer toutes les hypothèses de clé possibles et de retenir après comparaison celle qui présentera les plus grandes différences en terme de pics de courant.

Il existe plusieurs autres types d'attaques en puissance dérivées des attaques par analyse différentielle ou simple du courant, parmi lesquelles nous avons les attaques par collisions et les attaques par corrélation.

2.5 Attaque par collisions

L'attaque par collisions est basée sur l'analyse des correspondances sur des résultats ou sur des valeurs intermédiaires de chiffrement de différents messages choisis. Soit F la fonction de chiffrement telle que :

$$C_i = F(K, M_i) \quad (2.23)$$

M_i représente le message à chiffrer, C_i le résultat de chiffrement et K la clé.

Soit h une fonction interne de F (h peut être égal à la fonction F) avec :

$$h = (k, m) = c \quad (2.24)$$

k, m et c représentent respectivement une partie de la clé, du message et une partie du résultat intermédiaire.

Si $h(k, m_1) = h(k, m_2) = c$, alors il existe une collision au niveau de la fonction h . Cette collision peut être exploitée pour briser des cryptosystèmes. Les attaques par collisions furent développées pour briser des fonctions de hachage. En effet, la fonction de hachage permet de transformer un message de

longueur quelconque en une empreinte numérique de taille fixe (160 bits pour les algorithmes du DSS) [FIPS186]. Cette empreinte est généralement utilisée pour signer numériquement des messages ou pour l'authentification.

$$\begin{array}{ccccc} M & \longrightarrow & h(m) & \longrightarrow & y = \text{sig}[h(m)] \\ \text{message} & & \text{empreinte} & & \text{signature} \\ & & \text{numérique} & & \text{numérique} \end{array}$$

Le couple de valeur (M, y) représente un message numérique signé valide, mais si on trouve un message M' tel que :

$$M' \neq M \text{ et } h(M') = h(M) \quad (2.25)$$

Alors le couple (M', y) est également valide. Par conséquent, les fonctions de hachage sont fiables si et seulement si toute collision est impossible c'est à dire :

Etant donné un message M , il est calculatoirement impossible d'obtenir M' tel que : $M' \neq M$ et $h(M') = h(M)$.

L'analyse des courbes de courant permet dans certains cas de déterminer des collisions dans des algorithmes cryptographiques. Considérons l'algorithme du *DES* et ces 8 fonctions de substitutions surjectives (*SBOXi*).

Dans chacune des fonctions de substitution une valeur de sortie peut être sélectionnée par 4 différentes valeurs d'entrées. Soit la fonction *SBOX1*, on peut définir les relations suivantes :

$$\begin{aligned} \mathbf{SBOX1(m \oplus k) = SBOX1(z) = SBOX1(0) = SBOX1(5) = SBOX1(36) = SBOX1(55) = 14} \\ \mathbf{z = m \oplus k} \\ \mathbf{00 = 00 \oplus 0} \\ \mathbf{05 = 05 \oplus 0} \\ \mathbf{36 = 36 \oplus 0} \\ \mathbf{55 = 55 \oplus 0} \\ \mathbf{avec k = 0, et m = \{0, 5, 36, 55\}} \end{aligned} \quad (2.26)$$

Le résultat '14' en sortie de la *SBOX1* est obtenu pour différentes valeurs d'entrées. Ce type de relation peut être vérifiée pour chacun des résultats de sortie des fonctions de substitution du *DES*.

Par conséquent, la connaissance de m , permet de déterminer des collisions en sortie de la fonction de substitution et trouver la valeur de k . Pour cela, on calcule la fonction de corrélation entre les courbes de courant mesurées lors de l'exécution des messages m à l'instant t . Si à cet instant les mêmes données intermédiaires sont manipulées, alors les profils de courant seront identiques.

[Schr03] exploite la structure de Feistel du **DES** pour provoquer et déterminer par corrélation des courbes de courant des collisions. L'idée est de créer une collision à l'itération n ($n=I$) et de la détecter à l'itération $n+I$. Pour cela, des masques sont définis dans les registres de droite et de gauche (**R0** et **L0**) afin de ne faire varier uniquement que les bits nécessaires aux analyses des collisions. La détection de la collision est réalisée en calculant le coefficient de corrélation des courbes de courant. Des améliorations de l'attaque par collision sur des structures de Feistel et notamment sur le **DES** sont présentées dans [Ledi04]. Par contre [Schr04] propose une implémentation de l'attaque par collisions sur l'**AES** et en particulier sur la fonction **Mixcolumns**.

2.6 Attaque par corrélation

L'attaque par corrélation est basée sur le calcul du coefficient de corrélation entre deux variables X et Y . La notion de coefficient de corrélation utilisée dans les statistiques permet de mesurer toute relation linéaire entre deux variables. Elle s'exprime par le quotient de la covariance par le produit des écarts-types de chacune des variables. Plus elle est proche de 1 ou -1, plus les variables sont fortement corrélées.

$$r_{XY} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \quad (2.27)$$

[Brie04] a proposé l'utilisation de cette formule statistique pour déterminer toute corrélation entre le courant mesuré et le poids de Hamming d'un vecteur de bits d'un composant. Ainsi, en remplaçant la variable Y par le courant $W_i(t)$ et la variable X par la distance de Hamming $H_{i,R}$ du vecteur considéré, en déduit l'estimation suivante du coefficient de corrélation :

$$r_{WH}(R) = \frac{N \sum W_i H_{i,R} - \sum W_i \sum H_{i,R}}{\sqrt{N \sum W_i^2 - (\sum W_i)^2} \sqrt{N \sum H_{i,R}^2 - (\sum H_{i,R})^2}} \quad (2.28)$$

La distance de Hamming est exprimée par la relation $H_{i,R} = F(M_i \oplus R)$ dans laquelle R représente l'état de référence et F une fonction de l'algorithme. Le calcul de l'estimation est effectué à

chaque instant t des courbes de courant et pour toutes les valeurs aléatoires M_i . Pour une valeur correcte de R , le coefficient de corrélation tendra vers sa valeur maximale (1 ou -1).

La preuve formelle de cette approche a été apportée sur un modèle de courant basé sur la distance de Hamming. En effet, en considérant que la consommation ou l'énergie consommée par un circuit numérique est principalement due aux commutations des bits (de 0 vers 1 ou de 1 vers 0), on peut définir une relation linéaire entre le poids de Hamming et la quantité de courant consommée par le circuit par l'expression :

$$W = aH(V \oplus R) + b \quad (2.29)$$

R représente l'état initial du vecteur (registre), V son état final, H est la distance de Hamming entre V et R , a représente un facteur d'estimation de la quantité de courant consommée en fonction de la distance de Hamming H et b représente une variable indépendante du bruit du composant.

On exprime le coefficient de corrélation entre W et H par la relation :

$$\sigma_{WH} = \frac{\text{cov}(W, H)}{\sigma_W \sigma_H} = \frac{a\sigma_H}{\sqrt{a^2\sigma_H^2 + \sigma_b^2}} \quad (2.30)$$

Si la distance de Hamming H contient m bits indépendants et uniformément repartis, alors sa moyenne et sa covariance correspondent respectivement à $m/2$ et $m/4$. Ainsi on obtient :

$$\sigma_{WH} = \frac{a\sqrt{m}}{\sqrt{ma^2 + 4\sigma_b^2}} \quad \text{avec} \quad \begin{array}{l} \sigma_{WH} \longrightarrow \pm 1 \\ \sigma_b \longrightarrow 0 \end{array} \quad (2.31)$$

Lorsque le bruit est une gaussienne (σ_b), le coefficient de corrélation atteint sa valeur maximale. Considérant maintenant une valeur incorrecte R' comme la nouvelle valeur à évaluer telle que :

$$H' = H(V \oplus R') \quad \text{avec} \quad H(R \oplus R') = k \quad (2.32)$$

Tant que b est indépendant des données manipulées on a :

$$\sigma_{WH'} = \frac{\text{cov}(aH + b, H')}{\sigma_W \sigma'_H} = \sigma_{WH} \frac{m - 2k}{m} \quad (2.33)$$

Cette relation illustre la capacité de cette approche de rejeter des valeurs incorrectes de R . Si un seul bit est différent ($k=1$) sur 8 ($m=8$), alors la valeur σ_{WH} est réduite de $\frac{1}{4}$.

2.7 Réalisation d'une attaque en puissance sur un cryptoprocresseur synchrone

Nous allons dans cette partie présenter des résultats pratiques obtenus en réalisant des attaques en puissance par analyse simple et différentielle de courant. Ces attaques ont été réalisées sur un cryptoprocresseur synchrone implémentant l'algorithme du *DES*.

2.7.1 Architecture et caractéristiques du circuit : Cryptoprocresseur DES*

L'architecture du cryptoprocresseur est composée de quatre grands blocs : un banc de registres, un bloc de contrôle, le bloc de chiffrement et le bloc de génération des sous clés (figure 2.5).

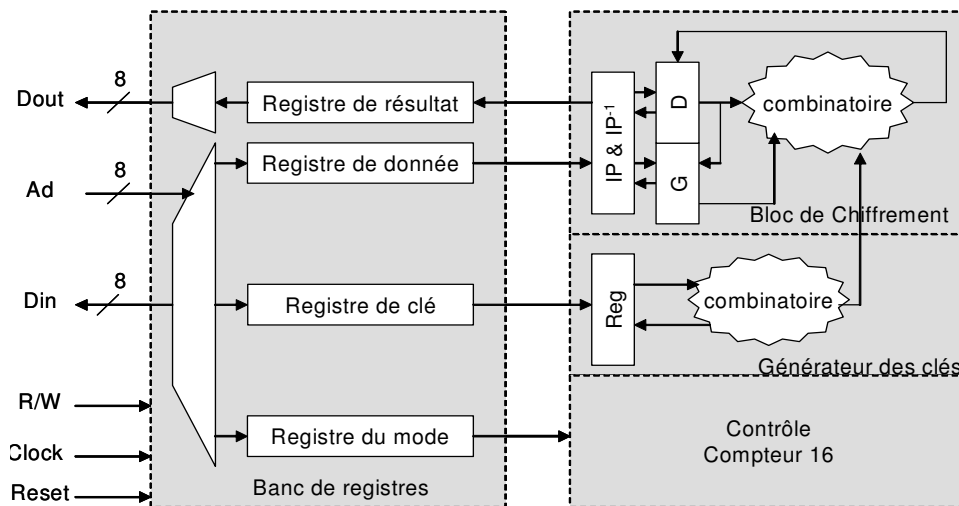


Figure 2.5 : Architecture du cryptoprocresseur synchrone *DES*

- Le **Banc de registres** : il est composé de 25 registres de 8 bits. 8 registres de données, 8 registres de clé, 8 registres de sortie (pour le résultat) et un registre de mode (contrôle). Le

* Ce circuit a été réalisé par le CEA-LETI dans le cadre du projet ESP@SSIS (MEDEA+).

registre de mode permet de gérer le type d'opération à effectuer (chiffrement ou déchiffrement) et le signal de début de calcul.

- Le **bloc de contrôle** comprend un compteur de 4 bascules qui permet de contrôler les 16 itérations de l'algorithme du **DES**.
- Le **bloc de chiffrement** implémente la fonction de chiffrement (et de déchiffrement) du **DES** notamment les deux registres de droite **D** et celui de gauche **G**, la fonction **f**, la fonction **xor** et les différentes fonctions de permutations du **DES**.
- Le **bloc de génération des sous clés** permet de générer à la volée les 16 différentes sous clés utilisées pendant chaque itération du processus de chiffrement ou déchiffrement.

Une fois les registres de données et de clés chargées, l'écriture dans le registre de mode permet de lancer le calcul. Le circuit a été réalisé en technologie **CMOS 0,18 μ m** de **STMicroelectronics**. Il est alimenté en 1.8 volts, il consomme en moyenne 2mA et effectue un calcul de chiffrement (déchiffrement) en 17 cycles d'horloge. Le layout du circuit est représenté sur la figure 2.6 avec une surface de 0.14mm² pour 11400 portes équivalentes.

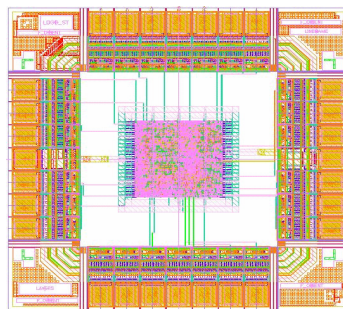


Figure 2.6 : Layout du cryptoprocasseur

Surface du cœur: 370 μ m \times 370 μ m,

Surface avec plots: 1050 μ m \times 1050 μ m

2.7.2 Analyse différentielle du cryptoprocasseur

La réalisation de l'attaque en puissance commence par une analyse de l'algorithme afin de définir la fonction de sélection. Une fois définie, des mesures de courant sont effectuées et des analyses simples ou différentielles sont réalisées pour retrouver les clés.

La fonction de sélection définie pour réaliser l'attaque est la première fonction de substitution du *DES* prise sur la première itération telle que :

$$D(m_i, k) = \sum_{j=0}^3 d_j 2^j = SBOX1(m_i \oplus k) \quad (2.35)$$

Elle correspond à l'exemple de la fonction de sélection définie au paragraphe 1.4.1 (équation 2.4). La fonction de répartition associée à la fonction de sélection *D* est définie en prenant les valeurs des bits en sorties de *SBOX1*. L'objectif de l'attaque est de pouvoir retrouver les 6 bits de la clé *k* utilisée dans la fonction *D*.

Nous allons, dans un premier temps, évaluer les corrélations entre les courbes de courant et les données manipulées dans la fonction *D*. La courbe de courant de la figure 2.7 représente le profil de courant d'une phase complète de chiffrement du cryptoprocresseur. Les 16 rondes du *DES* sont identifiables par les pics de courant dus à l'activité du signal d'horloge.

Les mesures sont réalisées aux bornes d'une résistance de 1 ohm entre la masse du composant et celle de la carte. Les courbes des figures 2.8 représentent un agrandissement de l'activité du courant pendant la première ronde. La courbe du dessus est une courbe de courant moyennée par l'oscilloscope sur 100 acquisitions, par contre celle du dessous représente une courbe de courant instantanée.

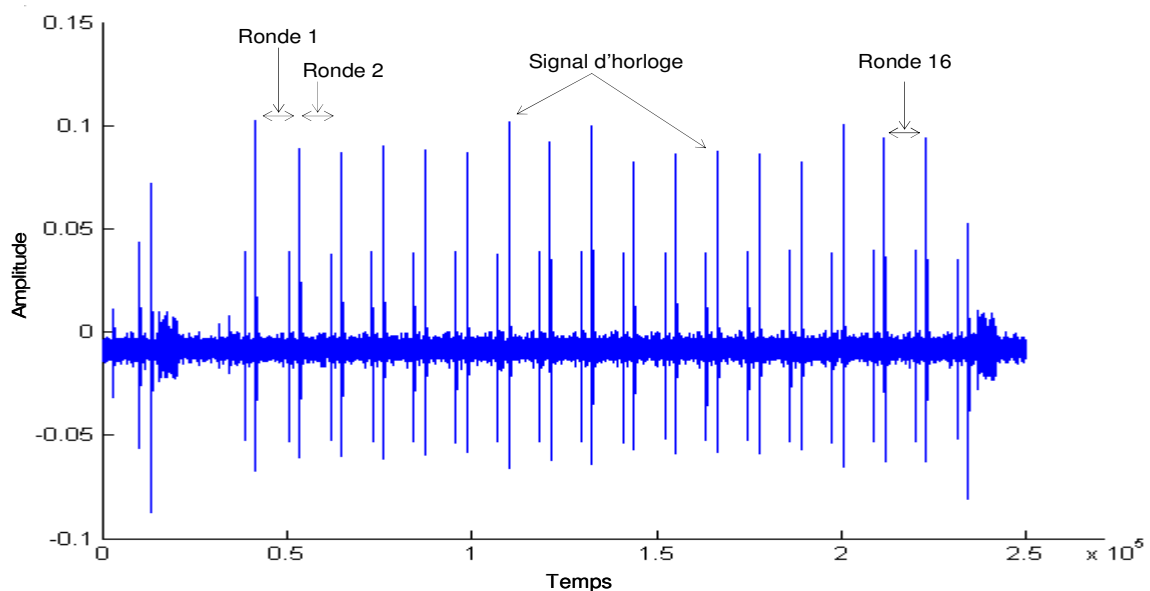


Figure 2.7 : Profil de courant du cryptoprocresseur synchrone *DES*

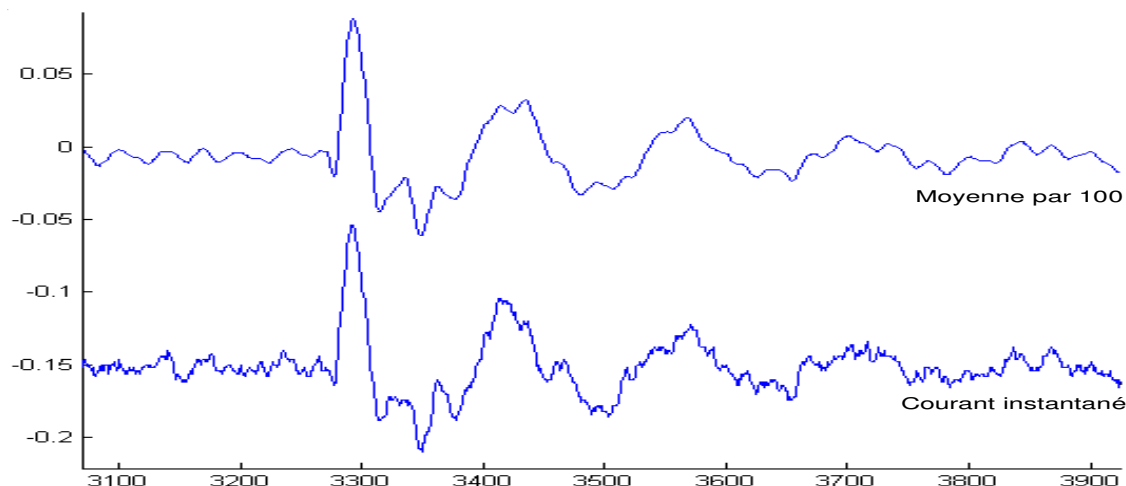


Figure 2.8 : Courant moyennée et instantané de la première itération du cryptoprocresseur synchrone

L'évaluation de la dépendance entre le profil de courant et les données manipulées est faite en comparant visuellement des courbes de courant correspondantes à des messages choisis. Le choix des messages est tel que, seuls les bits de m_i de la fonction D varient. m_i étant sur 6 bits, on obtient 64 textes en clair choisis. La figure 2.9 présente les 64 courbes de courant dont chacune est moyennée par 100. Ce type d'analyse correspond à une analyse *SPA*. Les zones agrandies illustrent les dépendances entre les données traitées et les profils de courant qui peuvent être exploitées par des analyses différentielles. Cela permet d'une part d'analyser la sensibilité de la fonction de sélection et d'autre part d'identifier des blocs sensibles du composant.

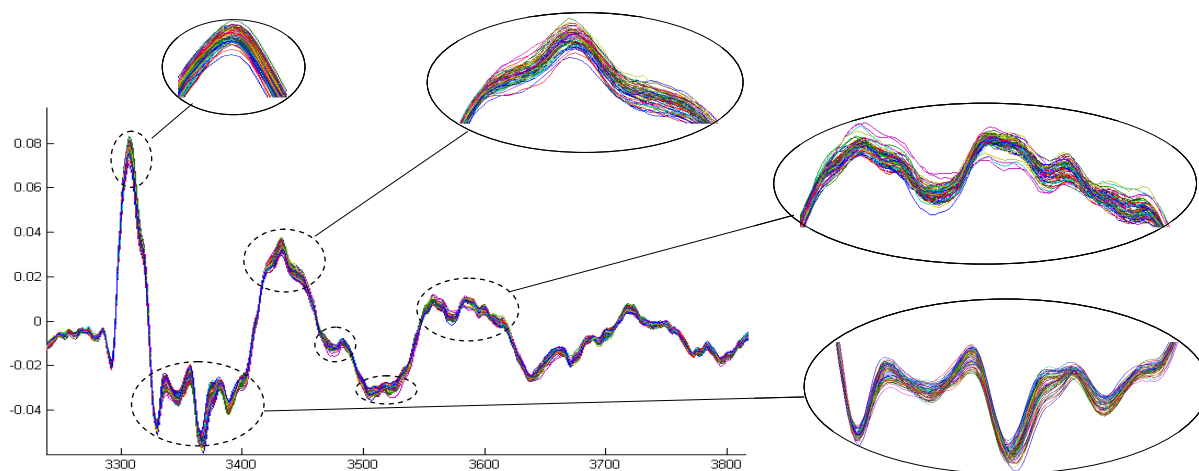


Figure 2.9 : Analyse des corrélations entre les données manipulées et les courbes de courant

Nous avons réalisé une attaque par analyse différentielle du courant en effectuant une recherche exhaustive sur toutes les valeurs possibles de k (64 cas possibles). Chacune des clés est testée en calculant pour chaque bit en sortie de la *SBOX1*, les signaux *DPA* (S_{DPAk}). On analyse par la suite, pour le bit testé, les 64 signaux *DPA* correspondants aux 64 clés possibles. La bonne clé présentera des pics de courant plus importants que les autres signaux. 64 messages clairs choisis sont utilisés. Ils correspondent tous aux 64 valeurs possibles de m_i et pour chacune de ces valeurs, on effectue 100 acquisitions ce qui correspond à un total $64 \cdot 100$ courbes manipulées.

Le schéma de figure 2.10 présente les résultats obtenus sur le bit 2 (d_2). Le signal *DPA* obtenu représente la bonne clé utilisée pour le calcul. Les pics en courant les plus importants apparaissent au début de la deuxième itération pendant et après l'échantillonnage des valeurs de la première itération par le signal d'horloge.

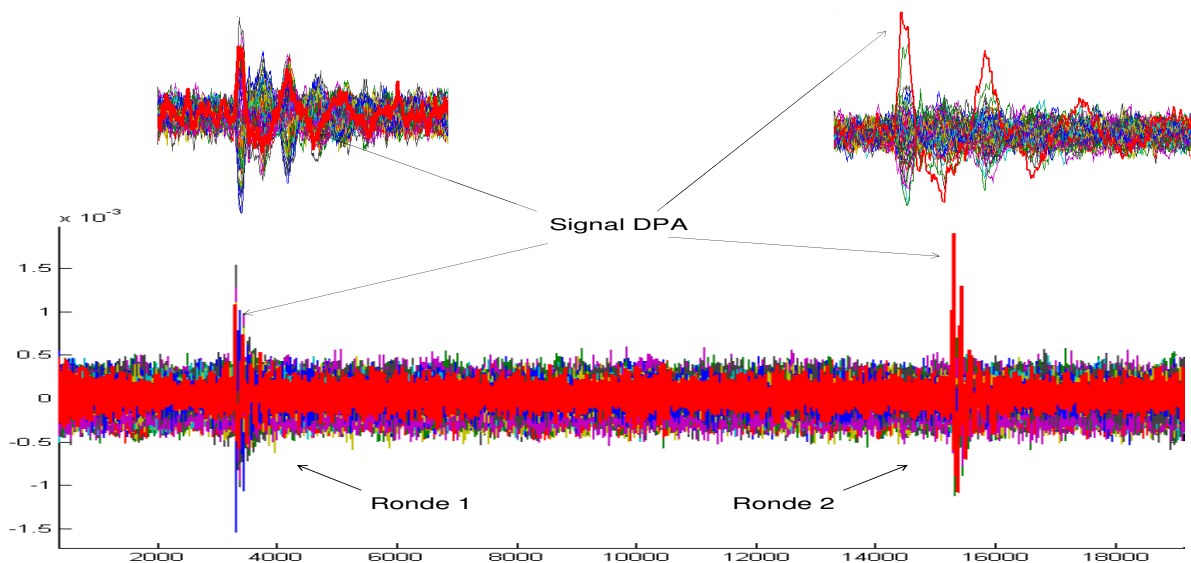


Figure 2.10 : Signal *DPA* sur le bit 2 en sortie de la *SBOX1*

En effet, le chemin de données du bit 2 en sortie de la première *SBOX1* est représenté en gras sur la figure 2.11. La valeur du bit 2, utilisée pour répartir les courbes de courant en sortie de la *SBOX1*, reste correcte jusqu'à la deuxième itération pendant le calcul de la fonction *Xor* avec la deuxième sous clé. Pour cela, nous avons fixé dans chacun des messages m_i , tous les bits de gauches (G) à zéro ($k_{17}=0$). Ce schéma peut être étendu aux différents bits attaqués en sortie des *SBOXi*.

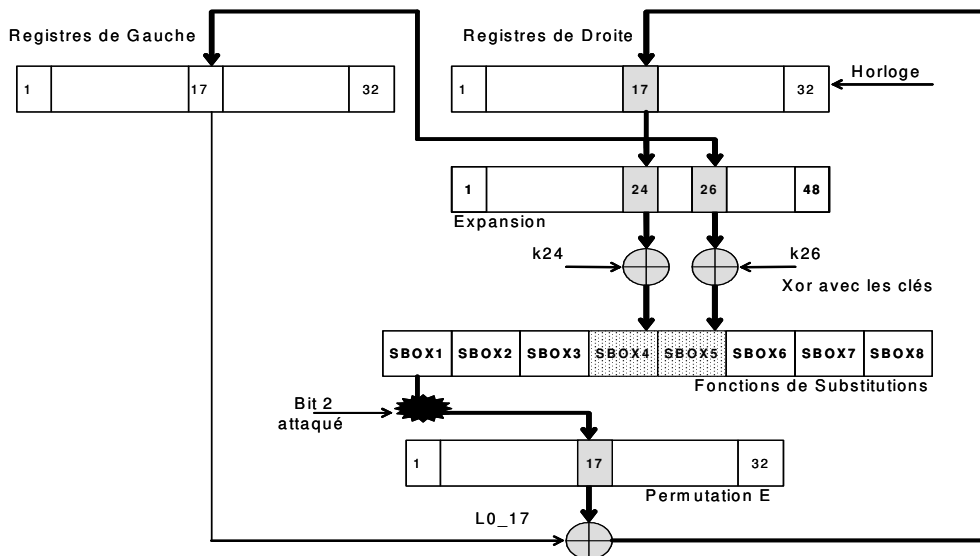


Figure 2.11 : Chemin de données du bit 2 en sortie de la *SBOX1*

La figure 2.12 présente le signal *DPA* obtenu en attaquant le bit 3 en sortie de la *SBOX1*. Comparé au résultat précédent, la signature en courant du bit 3 (signal *DPA*) est moins importante mais permet d'identifier la clé utilisée.

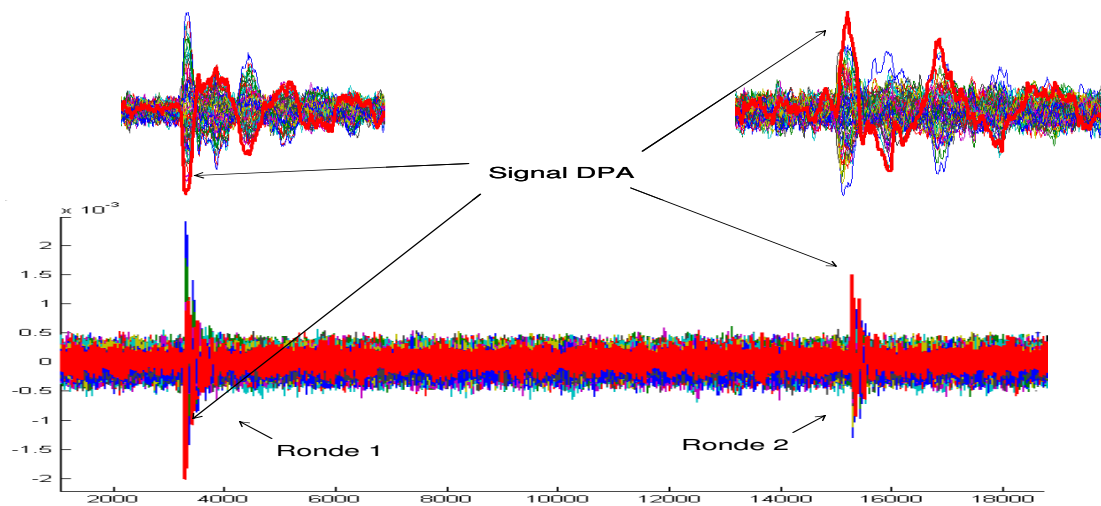


Figure 2.12 : Signal *DPA* sur le bit 3 en sortie de la *SBOX1*

Contrairement aux résultats obtenus sur les bits 2 et 3, les résultats des attaques différentielles sur les bits 1 et 4 ne peuvent être exploités pour trouver les clés. La figure 2.13 présente les signaux *DPA*

obtenus dans lesquels aucune courbe ne présente de différence significative en terme d'amplitude. Cela pourrait s'expliquer par le nombre limité de courbes utilisées pour réaliser les attaques.

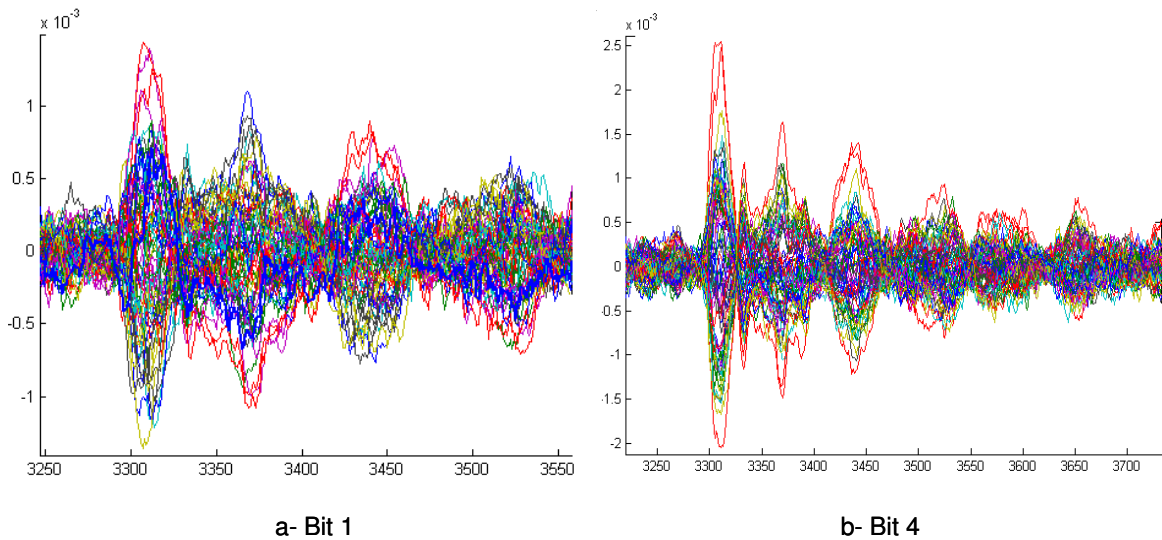


Figure 2.13 : Signaux *DPA* sur les bits 1 et 4 en sortie de la *SBOX1*

2.8 Conclusion

Basée sur l'exploitation des faiblesses d'implémentation des circuits intégrés, les attaques en puissance utilisent des corrélations entre les courbes de courant et les données manipulées pour retrouver des clés. L'existence de ces corrélations a conduit à développer plusieurs méthodes d'attaques dont les plus connues, les plus simples et parmi les plus efficaces sont celles proposées par Paul Kocher avec notamment les attaques par analyse simple et différentielle du courant que nous avons illustrées et appliquées sur un cryptoprocasseur synchrone.

Nous avons dans ce chapitre développé et adopté un formalisme qui nous a permis d'étudier et d'analyser le principe de fonctionnement d'une attaque différentielle. Basée sur une représentation du courant de consommation dû à l'activité électrique liée au bit attaqué, il a permis, d'une part, de démontrer théoriquement l'efficacité de l'attaque et d'autre part, d'introduire l'analyse des origines des fuites d'informations exploitées par les analyses en courant. Ces fuites d'information, observables et susceptibles de briser les codes confidentiels des circuits intégrés dédiés à des applications cryptographiques, sont étudiées en détails dans le prochain chapitre afin d'analyser sur chacune d'elles, l'influence des différentes contre-mesures proposées.

CHAPITRE III

ANALYSE DETAILLEE DES ATTAQUES EN PUISSANCE : ETAT DE L'ART SUR LES CONTRE-MESURES

3.1 Introduction

Nous avons dans les chapitres précédents défini la cryptanalyse et plus particulièrement la cryptanalyse matérielle en se focalisant sur les attaques en puissance. Elles sont rendues possibles uniquement par l'existence des corrélations entre d'une part les données manipulées et d'autre part le courant consommé par le circuit. Nous avons illustré l'efficacité et la simplicité de la mise en place d'une attaque par analyse différentielle du courant que nous avons implémentée sur un cryptoprocésseur synchrone.

Ce chapitre est consacré à l'étude et à l'analyse détaillée des origines des fuites d'informations observées sur le courant d'alimentation des circuits intégrés. Cette étude doit permettre d'explicitier le modèle de fuite d'informations $q_{id_i}(t)$ que nous avons défini dans le chapitre précédent (cf. § 2.3.3) en déterminant les paramètres logiques et physiques influençant la consommation des circuits intégrés en fonction des données traitées. L'objectif étant de pouvoir analyser les effets des contre-mesures proposées dans la littérature sur ces différents paramètres de manière à évaluer théoriquement leur efficacité.

3.2 Analyse du profil de courant dans un circuit intégré

La principale source de dissipation dans les circuits intégrés est la puissance dynamique. Il est bien connu que l'activité électrique dans un circuit intégré est essentiellement due aux commutations des éléments logiques et mémoires constituant le circuit. Les données manipulées dans les circuits entraînent des changements d'états des points mémoires et des nœuds logiques nécessitant des appels de courant. Le courant ainsi dissipé dans le circuit est proportionnel à l'activité des portes ayant commutées. Il est également bien établi qu'une porte logique *CMOS* ne dissipe pas la même quantité d'énergie lorsqu'elle charge ou décharge son nœud de sortie. Les cryptanalystes ont montré que cette différence de consommation est suffisante pour établir des méthodes qui permettent à travers l'analyse du courant, de retrouver les données traitées [Koche99]. Toutefois, ce phénomène ne représente pas l'unique cause expliquant l'origine des fuites d'information dans les circuits intégrés.

L'objectif de cette étude est d'analyser les différents paramètres tant logiques que physiques susceptibles d'affecter le profil de courant d'un circuit. Pour cela, nous avons dans un premier temps au niveau physique, focalisé nos analyses sur le comportement électrique d'une cellule standard et dans un second temps, analyser au niveau logique les effets du nombre de transitions logiques sur le profil de courant dans un bloc de portes logiques.

Cette étude permettra d'une part d'identifier les différents paramètres qui sont à l'origine des fuites d'informations observables sur le courant d'alimentation des circuits intégrés et d'autre part d'évaluer la sensibilité des circuits intégrés.

3.2.1 Analyse du courant d'une porte inverseuse

Les différentes analyses en courant réalisées dans cette partie sont faites sur une porte inverseuse *CMOS* en utilisant les modèles de simulation *Hspice* de la technologie *CMOS 0.13 μ m* de *STMicroelectronics (HCMOS9)*. Ces études sont basées sur les travaux de recherche développés dans [Mau01]. Le choix de l'inverseur permet d'étendre les résultats sur différentes portes logiques *CMOS* du fait que la majorité des portes logiques simples peuvent être ramenées à des inverseurs équivalents [Mau01].

La figure 3.1 illustre les variations de courant circulant dans les transistors P et N de la porte inverseuse ainsi que les tensions d'entrée et de sortie. Un front montant de l'entrée a été considéré sachant qu'il est facile de déduire des résultats similaires sur front descendant de l'entrée.

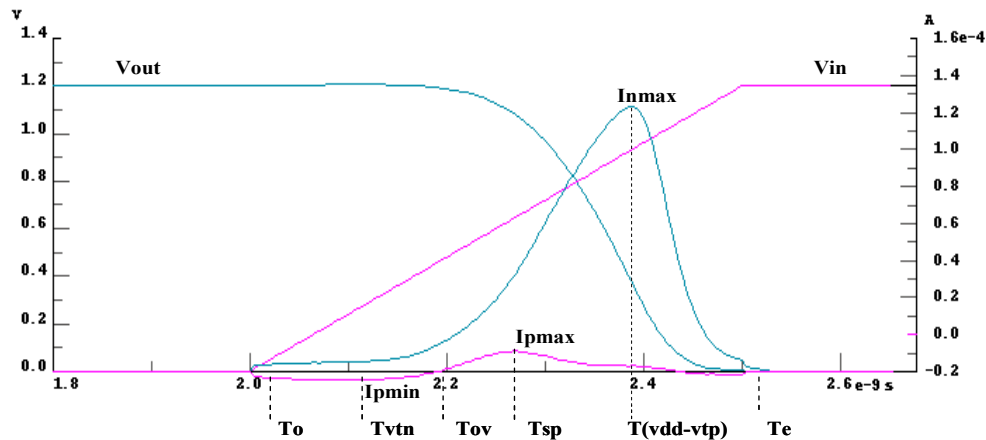


Figure 3.1 : Tensions d'entrée/sortie et courants des transistors P et N dans un inverseur $CMOS$ asymétrique ($k=W_p/W_n=1,79$) sur front montant. La capacité de charge est de 16fF.

Lors de la commutation de la porte, les transistors passent successivement dans différents régimes de conduction qui permettent de déduire les trois zones de fonctionnement suivantes :

- **la zone de surtension :** Comprise entre T_o et T_{ov} , elle correspond à la décharge en sortie du courant directement transmis de l'entrée à la sortie à travers la capacité de couplage (crosstalk interne). Le courant ainsi transféré est évacué par la masse et par l'alimentation (figure 3.2 -a).
- **la zone de court circuit :** Comprise entre T_{ov} et $T_{vdd-vtp}$, elle correspond à l'instant où le transistor P est en mode linéaire et N en mode de saturation fonctionnant comme un générateur de courant (figure 3.2 -b).
- **La zone de décharge :** Elle est comprise entre $T_{vdd-vtp}$ et T_e . Dans cette zone, le transistor P est bloqué et la capacité de sortie se décharge à travers le transistor N (figure 3.2-c).

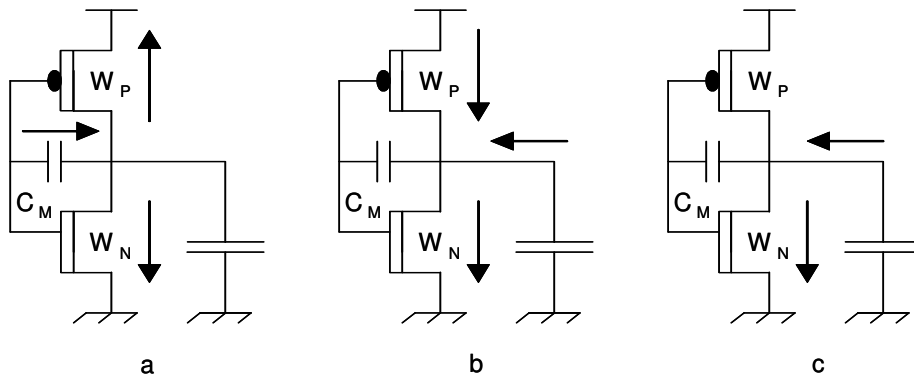


Figure 3.2 : Zones de fonctionnement de la porte inverseuse. a- correspond à la zone surtension; b- à celle de cour circuit et c- correspond à la zone de décharge.

Comme le démontre les différents travaux réalisés en matière de modélisation des portes logiques [Rezz00, Auve00, Niko99], les courants maximaux traversant les transistors P et N sont fonctions des paramètres environnementaux de la porte tels que : les rampes des signaux d'entrée, la topologie de la porte, la taille des transistors, la capacité de sortie etc... Nous allons par la suite, analyser chacun de ces paramètres afin d'examiner leurs influences sur le profil de courant.

3.2.1.1 Effet de la rampe d'entrée

Les effets des rampes des signaux d'entrées sont généralement pris en compte dans la modélisation des délais et dans la détermination du courant maximal des portes. L'évolution du courant dans une porte à permis de déterminer deux domaines de rampes: le domaine des rampes rapides et le domaine des rampes lentes [Mau01].

3.2.1.1.1 Les rampes rapides

Le domaine des rampes rapides correspond à l'intervalle de temps dans lequel la rampe d'entrée atteint sa valeur finale V_{dd} avant que le courant atteigne sa valeur maximale I_{max} . Soit τ_{in} la durée caractérisant la rampe d'entrée. Les effets des variations de τ_{in} sur le profil de courant sont représentés sur la figure 3.3. Comparé aux variations en temps, les amplitudes maximales du courant I_{max} sont indépendantes des rampes d'entrée comme le démontre l'équation (3.1) proposée dans [Mau01].

$$I_{max} = K_n W_n (V_{dd} - V_{thn}) \quad (3.1)$$

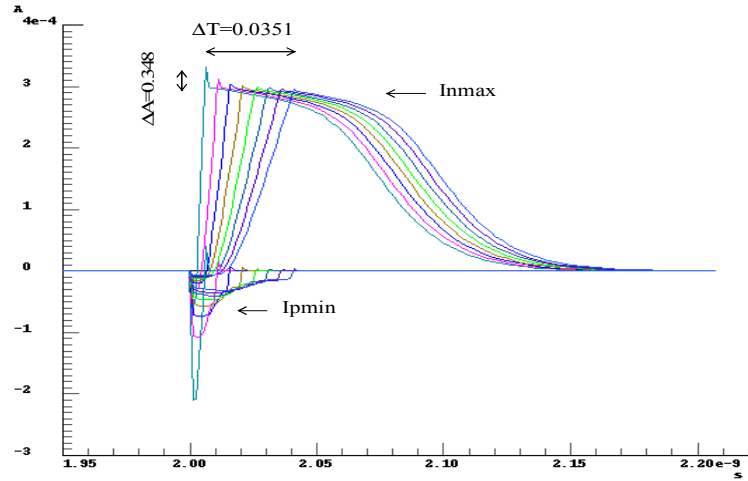


Figure 3.3 : Effet d'une rampe rapide sur le profil de courant. L'intervalle de variation de la rampe est tel que $5 ps \leq \tau_{in} \leq 40 ps$

La variation du courant maximum est de l'ordre de $35 \mu A$ dans un intervalle de $36 ps$. Les variations en amplitudes représentent près de 10% de la valeur maximale du courant, par contre les variations en temps sont proches du délai de commutation d'un inverseur qui est de l'ordre de $40 ps$.

3.2.1.1.2 Les rampes lentes

Contrairement aux rampes rapides, dans le domaine des rampes lentes le courant maximum I_{max} est atteint avant que la tension d'entrée atteigne sa valeur finale V_{dd} . Sa valeur, modélisée dans [Mau01] est inversement proportionnelle à la racine carrée de la durée de la rampe d'entrée τ_{in} :

$$I_{max} = \sqrt{\frac{K_n W_n V_{dd}^2 C}{\tau_{in}}} \quad (3.2)$$

C est la capacité de sortie incluant la capacité de charge C_l , la capacité de routage C_{par} et la capacité de court circuit C_{sc} . Cette équation montre la dépendance entre le profil de courant et la rampe d'entrée. Le schéma de la figure 3.4 illustre cette dépendance.

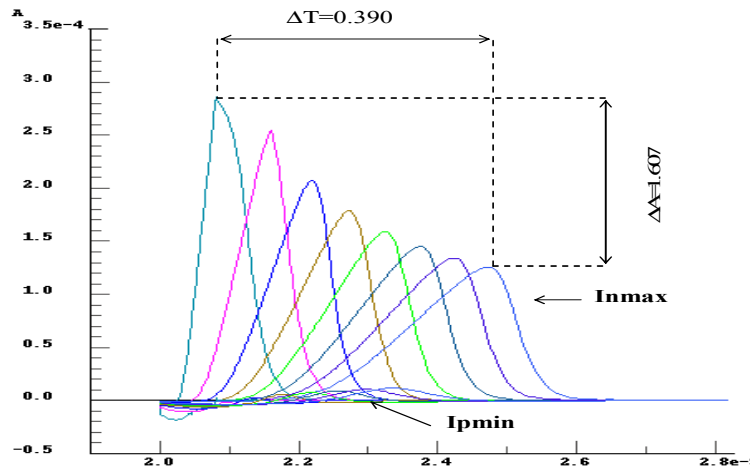


Figure 3.4 : Effet d’une rampe lente sur le profil de courant. L’intervalle de variation de la rampe est tel que $80\text{ ps} \leq \tau_{in} \leq 160\text{ ps}$

Comparé aux variations en amplitude dans le domaine des rampes rapides, les variations en amplitudes et en temps sont plus conséquentes dans le domaine des rampes lentes.

3.2.1.2 Effet du délai

Soit T_{hl} l’intervalle de temps compris entre la tension d’entrée et la tension de sortie lorsqu’elles sont à $V_{dd}/2$. Ce temps représente le délai de propagation de la porte. En se basant sur les travaux [Rezz00], on peut modéliser le délai par l’expression suivante :

$$T_{hl} = \delta_n \cdot v_{in} \cdot \tau_{in} + \frac{C_m \tau_{st} (1 - v_{th})}{C_N} + t_{HLS} (1 - corr) \tag{3.3}$$

τ_{st} Caractérise les performances en vitesse de la technologie et t_{HLS} la réponse indicielle de l’inverseur. Cette équation prend en compte trois facteurs : les effets de la rampe d’entrée (τ_{in}), les effets des différentes capacités de couplage, de charge et dynamique (C_m, C_N) et un facteur de correction incluant les effets non linéaires de la rampe d’entrée ($corr$). Cependant, ce délai n’inclut pas le courant mais dépend des paramètres influençant le courant. En effet, le courant ne varie pas directement en fonction du délai de la porte. Une porte possédant une charge important C et affectée par un signal de rampe rapide peut avoir

le même délai mais pas le même profil de courant lorsqu'elle possède une faible capacité de charge et quelle est affectée par une rampe lente.

3.2.1.1 Effet de la capacité

Elle regroupe toutes les capacités de charge en sortie de la porte. En fonction de la rampe d'entrée, sa valeur affecte différemment le profil de courant. La rampe de sortie correspond au temps mis par la porte pour décharger sa capacité de sortie. Elle dépend des possibilités en courant de la porte et de la charge en sortie de la porte.

$$\tau_{out} = C \frac{V_{dd}}{I_{max}} \quad (3.4)$$

Sa valeur représente la rampe d'entrée de la porte qui la précède $\tau_{out(i)} = \tau_{in(i+1)}$.

* **Cas des rampes rapides** : dans le domaine des rampes rapides la valeur de la capacité en sortie affecte la forme d'onde du courant de décharge (la pente négative du courant). La figure 3.5 illustre les formes d'ondes du courant et celles des rampes de sortie.

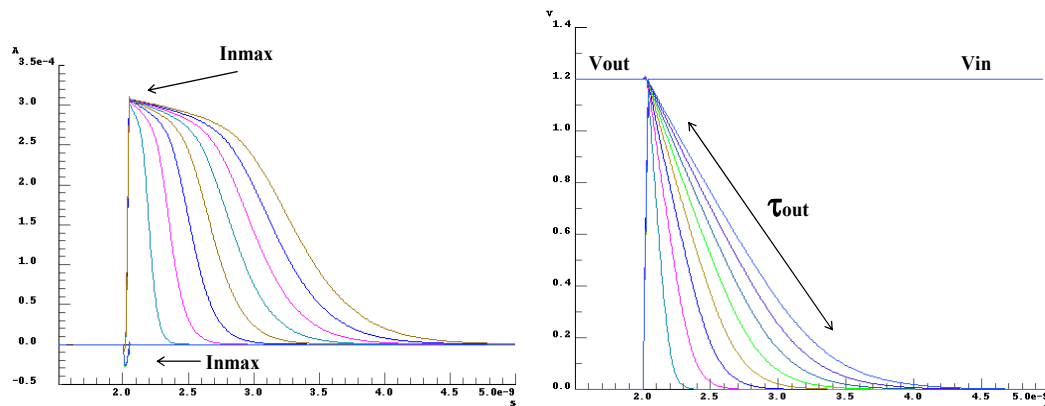


Figure 3.5 : Effet de la capacité de charge sur le profil de courant et sur la rampe de sortie dans le domaine des rampes rapides ($\tau_{in} = 40 \text{ ps}$) pour $C: 80 \text{ fF} \leq C \leq 320 \text{ fF}$

Le courant atteint toujours sa valeur maximale I_{max} et sa pente négative varie en fonction de la capacité en sortie qui est proportionnelle à la rampe de sortie. La valeur de la capacité de sortie C

comprend la capacité de routage C_{par} et la capacité de charge C_l . La capacité de court circuit C_{sc} peut être négligée.

* **Cas des rampes lentes** : Dans le cas des rampes lentes, la valeur de la capacité de sortie affecte aussi bien la pente négative du courant que sa valeur maximale. En remplaçant dans l'expression du courant de l'équation (3.4), l'expression du courant maximum en rampe lente (éq. (3.2)), on obtient :

$$\tau_{out} = \frac{V_{dd}}{\sqrt{\frac{K_n W_n V_{dd}^2}{C * \tau_{in}}}} \quad (3.5)$$

$$\text{avec } C = C_l + C_{par} + C_{sc}$$

La figure 3.6 présente les formes d'ondes obtenues.

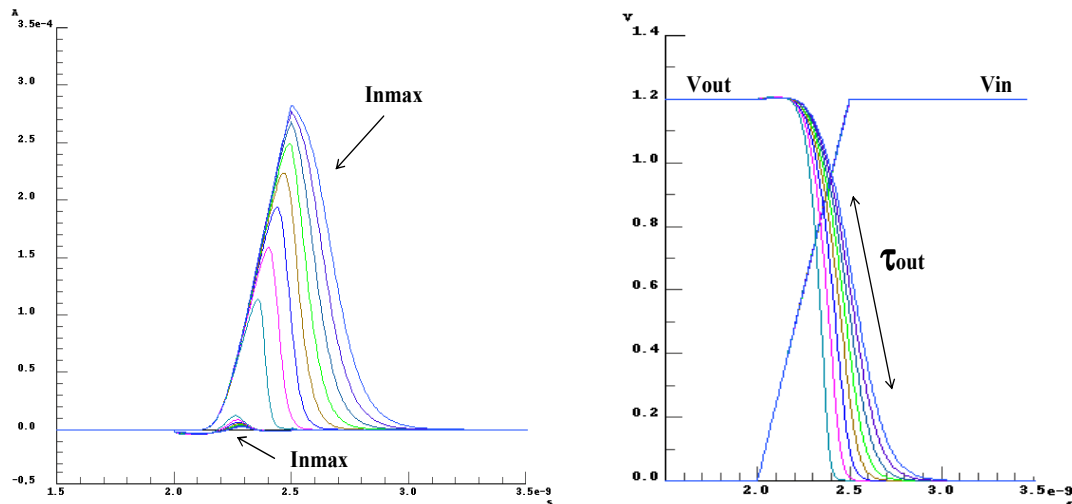


Figure 3.6 : Effet de la capacité de charge sur le profil de courant et sur la rampe de sortie dans le domaine des rampes lentes ($\tau_{in} = 50ns$) pour $C : 10 fF \leq C \leq 80 fF$

Le courant maximum augmente en fonction de la capacité de sortie C tandis que la rampe de sortie décroît.

3.2.1.1 Effet des commutations en sortie

Les effets des commutations en sortie se ramènent à analyser les possibilités en courant des transistors de type P et N . En effet, la mobilité des porteurs dans les transistors N est environ deux fois

plus importante que celle des porteurs libres dans les transistors P . Cette différence introduit une dissymétrie sur les possibilités en courant entre les transistors P et N . En considérant uniquement le domaine des rampes rapides, les expressions des courants de P et N sont données par :

$$\begin{aligned} I_{n\max} &= K_n W_n (V_{dd} - V_{thn}) \\ I_{p\max} &= K_p W_p (V_{dd} - V_{thp}) \end{aligned} \quad (3.6)$$

En posant : $R_\mu = K_n / K_p$; $V_{thp} = V_{thn}$ et $W_n = W_p$ le courant maximum dans P est égal au courant maximum dans N divisé par R_μ : $I_{p\max} = \frac{I_{n\max}}{R_\mu}$.

Cette dissymétrie est généralement corrigée dans les bibliothèques $CMOS$ en augmentant la taille des transistors P ($W_p = 2.8 * W_n$). La figure 3.7 illustre les différences en courant obtenues entre une porte inverseuse symétrique et asymétrique.

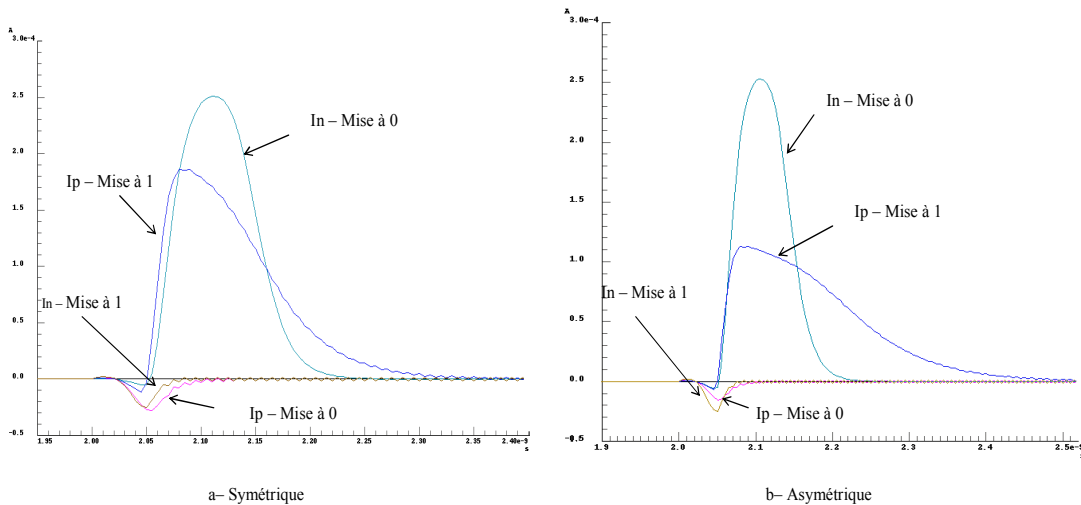


Figure 3.7 : Profils des courants des transistors P et N des portes inverseuses symétrique et asymétrique

3.2.2 Analyse du courant des portes simples et complexes

Dans cette partie, nous étendons les analyses réalisées sur la porte inverseuse sur des portes simples telles que des portes $NAND$ ou NOR et sur des portes complexes de type $AND-OR(AOI)$ ou $OR-AND(OAI)$ et bascule D . L'objectif est de pouvoir évaluer les effets des transistors en série sur le profil de

courant. Pour cela nous allons considérer une porte *NAND* à 3 entrées avec les nomenclatures représentées sur la figure 3.8.

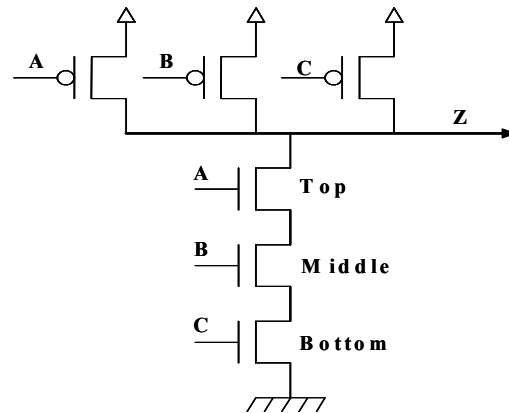


Figure 3.8 : Schématique d'une porte *CMOS NAND* à 3 entrées

Les figures 3.9 et 3.10 présentent les courants de source du transistor Bottom et les tensions d'entrées et de sorties lorsque la commutation de la porte est due soit au transistor A (Top), soit au transistor B(Middle) ou soit au transistor C(Botton) pour les deux domaines de rampes.

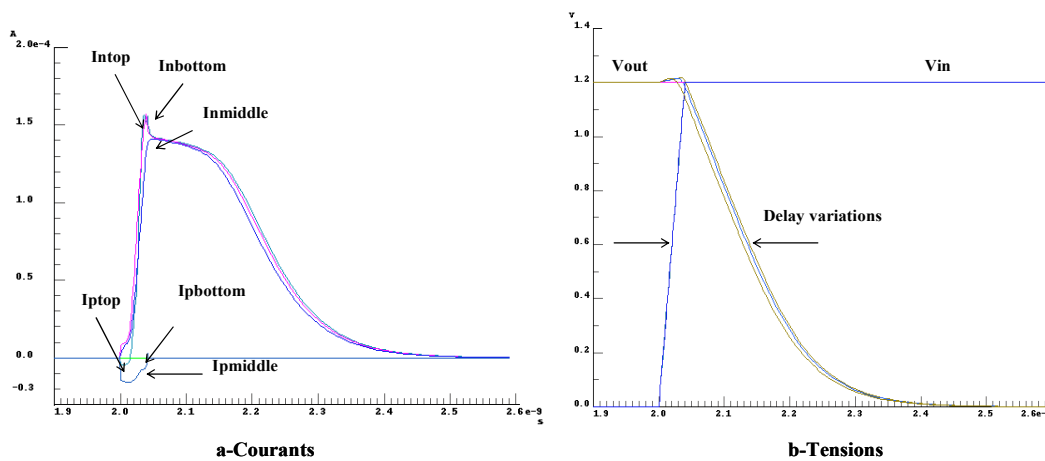


Figure 3.9 : Profils de courant des transistors en série d'une porte *NAND* à 3 entrées et variations du délai dans le domaine des rampes rapides.

Dans le domaine des rampes rapides, les formes d'ondes du courant circulant dans les transistors en série, lorsque les commutations de la porte sont successivement dues aux transistors Top, Middle et Bottom, sont quasi identiques. Toutefois une légère différence apparaît pendant la phase de surtension

lorsque les transistors Top et Middle sont activés. Ce phénomène est essentiellement dû aux capacités de couplages et de la proximité des drains de ces transistors par rapport à la sortie.

Cependant, les différences observées dans le domaine des rampes lentes sont plus importantes. Les courbes de courant ne présentent pas les mêmes amplitudes et sont étalées dans le temps.

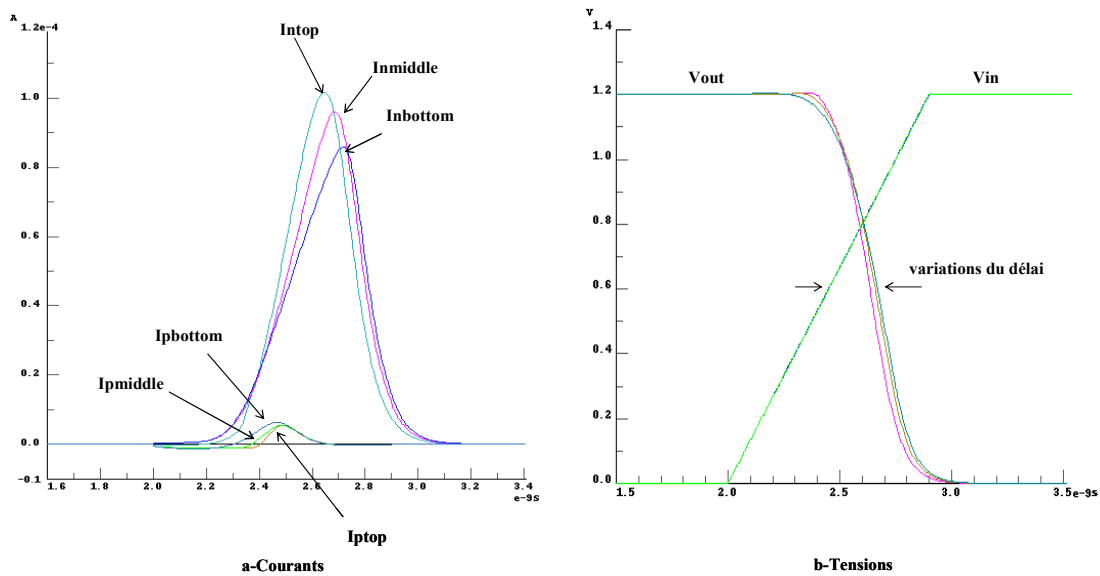


Figure 3.10 : Profils de courant des transistors en série d'une porte *NAND* à 3 entrées et variations du délai dans le domaine des rampes lentes

La variation du délai de la porte est essentiellement due à la durée de la rampe du signal de sortie. Elle augmente en fonction du transistor activé et ses variations sont plus importantes en rampe lente qu'en rampe rapide.

Les portes complexes et les bascules sont composées de plusieurs niveaux logiques dont chacun comprend une ou plusieurs structures de porte simple et inverseur en parallèle ou en série. En réduisant par niveau chacune de ces structures en des inverseurs équivalents, on obtient une chaîne d'inverseurs dont l'analyse en courant est équivalente à l'analyse du courant sur un chemin de données d'un bloc.

3.2.3 Analyse du courant dans un bloc

Après avoir analysé les paramètres influençant le profil de courant d'une porte dans les précédents paragraphes, nous allons, dans cette partie, analyser les effets de chaque porte sur la composition du courant dans un bloc logique.

Le profil de courant dans un bloc est une composition dans le temps des courants de chaque porte du bloc commutant lors d'un calcul. Le courant consommé dépend du nombre de commutations dans le bloc. Ce nombre est caractérisé par la distance du poids de Hamming entre l'état logique précédent et l'état logique courant du bloc.

Soit T_{cc} le temps du chemin critique d'un bloc et T_b le temps auquel le bloc est activé. La forme d'onde du courant du bloc est exprimée en faisant la somme des courants de chaque porte ayant commuté à chaque instant t_i compris entre T_b et T_{cc} .

$$I_{bloc}(t)_M = \sum_{t=T_b}^{t_{cc}} \sum_{i=0}^{N_{porte_a_t}} I_{porte_i}(t) \quad (3.7)$$

$N_{porte_a_t}$ représente le nombre de portes du bloc qui commute à l'instant t et M la valeur d'entrée du bloc. Le nombre $N_{porte_a_t}$ est fonction de l'état logique R précédent du bloc, de l'état logique courant dû à M (la valeur d'entrée). Elle représente le nombre de portes activées à l'instant t .

Afin d'évaluer la dépendance entre le nombre de commutations logiques dans un bloc et le poids de Hamming des données manipulées, nous avons analysé l'impact du poids de Hamming sur le nombre de transitions logiques dans la **SBOX1** du **DES** dont le schéma logique est représenté sur la figure 3.11.

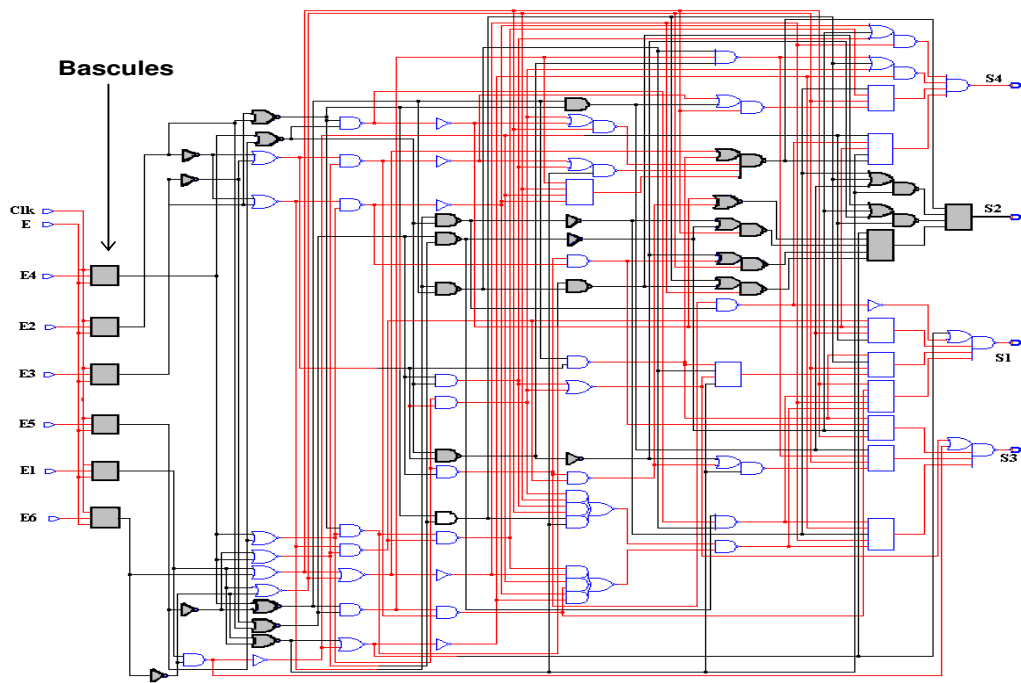


Figure 3.11 : Schématisation du bloc *SBOX1* : Chemin de donnée du bit de sortie 2

Les figures 3.12 illustrent les résultats obtenus en considérant des transitions supérieures ou égales à 100fs (toutes les transitions logiques d'une porte sont fixées à 100fs). La figure 3.12-b présente les variations du nombre total de commutations en fonction du poids de Hamming par contre la figure 1.12-a présente les formes d'ondes de courant obtenues sur la *SBOX1*. L'entrée du bloc étant sur 6 bits, nous avons fait varier en partant de l'état 0 le poids de Hamming des bits d'entrées entre 1 et 6. Comme on peut le constater, à l'instant t correspondant au changement logique des états des bascules, l'amplitude du courant est proportionnelle au nombre de bascules activés et le nombre de commutations dans de telle fonction croît presque linéairement en fonction du poids de Hamming.

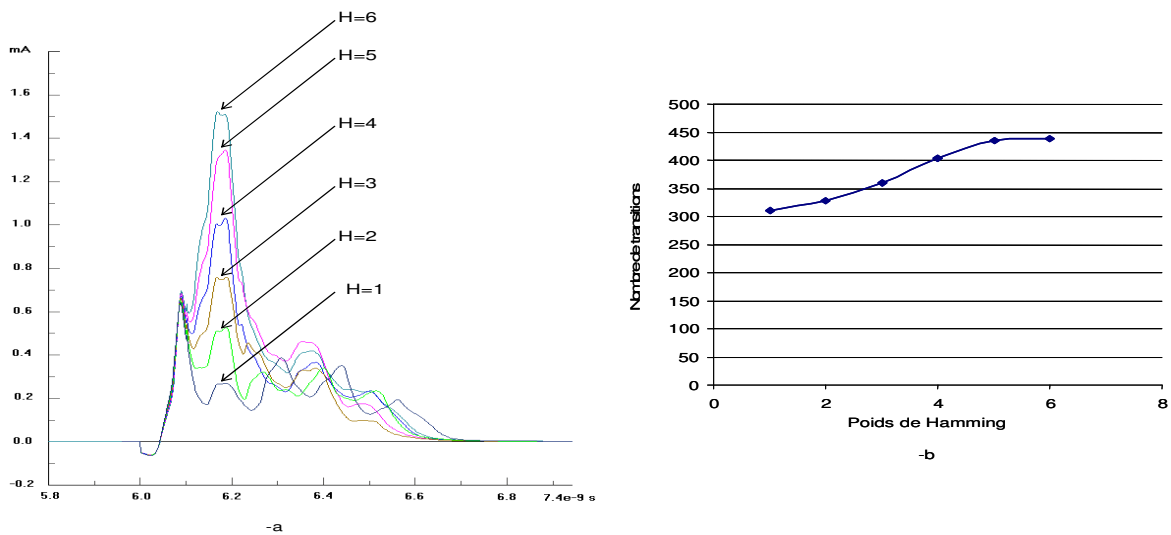


Figure 3.12 : Courants et variations du nombre de commutations en fonction du poids de Hamming des entrées dans le bloc *SBOX1*

Ce point représente l'un des principaux fondements sur lesquels se reposent les attaques en puissance. Comme le confirme le tableau 3.1, le courant consommé dans les blocs des circuits intégrés dépendent de l'état précédent de tous les nœuds logiques du bloc (R) en fonction de l'état courant (M) (valeur courante de la donnée traitée).

Le tableau 3.1 représente un exemple de variation du nombre de commutations logiques en fonction du poids de Hamming sur un chiffrement complet du *DES*. Pour chacune des clés utilisées, on fait varier le poids de Hamming du message à chiffrer. Les variations du nombre de transitions logiques obtenues en fonction des données traitées permettent d'illustrer la dépendance logique entre le courant consommé et la nature des données manipulées dans un bloc.

Tableau 3.1 Variations du nombre de transitions logiques lors du chiffrement d'un *DES* en fonction du poids de Hamming des messages. Le poids de Hamming des messages est incrémenté en partant de 0 ($R=0$). Les messages et les clés sont pris aléatoirement.

| Poids de Hamming des données | 0 | 10 | 20 | 30 | 40 | 50 | 60 |
|--|-------|-------|-------|-------|-------|-------|-------|
| Nombre de transitions Clé: 0000000000000000 | 8389 | 8573 | 8982 | 8506 | 8712 | 8626 | 9222 |
| Nombre de transitions Clé: 133457799BBCDFF1 | 14894 | 15017 | 14606 | 15109 | 14731 | 15298 | 14758 |

3.2.4 Sensibilité électrique d'un bloc

Le tableau de 3.2 résume la sensibilité électrique, en termes de variation d'amplitudes et de temps d'occurrence des pics de courant d'un bloc en fonction de ses paramètres logiques et physiques.

Tableau 3.2 Variations des amplitudes et du temps d'occurrences des pics de courant en fonction des paramètres environnementaux d'un bloc logique.

| | Analyses au niveau logique | Analyses au niveau électrique | | | | | |
|----------------------------|---|-------------------------------------|----------------|-----------------------|--------------------------|-----------------|----------------------|
| | Nombre de transitions logiques: poids de Hamming | Commutation: charge/ décharge | Rampe lente | Capacité en sortie | Structure de la porte | Rampe rapide | Délai de la porte |
| Amplitude des pics | ✓✓✓✓ | ✓✓✓ | ✓✓ | ✓✓ | ✓ | — | — |
| Décalage temporel des pics | ✓✓✓ | ✓ | ✓✓✓✓ | ✓ | ✓ | ✓✓ | — |

✓✓ Grande variations

✓ Variations moyennes

— Indépendant

Les paramètres présentés sur ce tableau, permettent d'expliquer, dans un circuit *CMOS*, les différences de consommation et notamment de profil de courant entre les quantités de courant $q_{i0}(t)$ et $q_{i1}(t)$ correspondant respectivement à la mise à 0 et 1 d'un bit en sortie d'un bloc. Ce tableau montre que le profil de courant dans un bloc, c'est-à-dire le nombre de pics, l'amplitude des pics et leurs occurrences dans le temps sont fortement dépendants du nombre de commutations logiques du bloc. Ce nombre dépend du poids de Hamming des données manipulées dans le bloc. A cela, s'ajoute des dépendances au niveau électrique dues à la nature des commutations mises en œuvre au niveau de chaque porte logique (charge ou décharge), le type de rampe des signaux et les valeurs des capacités en sorties des portes logiques.

En fonction de ces analyses, nous allons maintenant évaluer les effets des différentes contre-mesures proposées.

3.3 La cryptanalyse matérielle : les contre-mesures

De nombreuses contre-mesures ont été développées pour protéger les circuits intégrés contre chacune des différentes catégories d'attaques matérielles. Les enjeux économiques ont rendu ce domaine tellement sensible que les systèmes de sécurité implémentés dans les circuits sont devenus des informations confidentielles (cf. § 1.3). En effet, les fondeurs, notamment de cartes à puce, communiquent très peu sur les moyens de protection développés et implémentés dans leurs produits. Toutefois, les concepts proposés dans les laboratoires académiques publics sont proches de ceux développés dans l'industrie.

3.3.1 Les contre-mesures contre les attaques invasives

Les méthodes de protection développées pour protéger les circuits intégrés contre les attaques invasives sont basées sur l'utilisation de capteurs de détection du désassemblage du circuit, de procédés de placement et de routage aléatoires et sur l'utilisation de couches ou mèches de protection des puces [Kömm99, Kömm 99, Samy 02, Ross 96].

- **Capteur de détection du désassemblage** : l'utilisation de ce capteur permet de détruire les valeurs des mémoires une fois qu'une tentative de désassemblage du circuit est détectée.

- ***Placement et routage aléatoire*** : elle consiste à diffuser dans les blocs logiques les éléments de mémorisation afin d'une part, d'augmenter les difficultés d'identification des registres et des bus lors d'attaques sous pointes et d'autre part, de complexifier les opérations de reconstitution du layout.
- ***Couches de protection*** : certains fabricants insèrent lors des phases de fabrication des puces une grille de protection supplémentaire sur le dessus de la puce afin d'empêcher toute identification des blocs et toute attaque par pointe.

Ces types de protections présentent des coûts en terme de conception et de procédé de fabrication non négligeables.

3.3.2 Les contre-mesures contre les attaques par injection de fautes

Deux types d'approches sont utilisés pour implémenter des circuits résistants aux attaques par injection de fautes.

La première est basée sur l'implémentation de capteurs sur les signaux d'alimentation, le signal d'horloge, ou des capteurs de température. Ils permettent de détecter et de bloquer, le cas échéant, le circuit lors de variations intempestives des signaux d'entrées ou lorsque les conditions de fonctionnement du circuit sont anormales.

La seconde approche est basée sur l'utilisation de la redondance logique communément utilisée dans le domaine spatial pour durcir les circuits contre les fautes générées par l'effet de radiations. La répétition des calculs est également préconisée afin de vérifier les résultats plusieurs fois de suite avant de les délivrer en sortie. A cela s'ajoute l'utilisation des nombres aléatoires pour masquer les données à l'aide de vecteurs d'initialisation aléatoires [Mahe 97, Bone 97].

Cependant, l'utilisation des capteurs reste un problème complexe dans le fonctionnement des circuits car le moindre aléa sur les signaux d'entrées (alimentation, horloge) bloque le circuit et l'implémentation des redondances, des procédures de vérification et des vecteurs aléatoires sont pénalisant en termes de surface, de performance et de consommation des circuits.

Une nouvelle alternative en cours d'évaluation est l'exploitation des redondances logiques contenues dans les circuits asynchrones notamment dans les circuits quasi insensibles aux délais qui les rendent particulièrement intéressants pour le durcissement des circuits [Monn05].

3.3.3 Les contre-mesures contre les attaques temporelles

Pour éviter toute dépendance temporelle entre les données manipulées et le temps d'exécution, les concepteurs doivent à tous les niveaux de conception, éviter d'utiliser des fonctions, des instructions, des tests, des sauts et des branchements qui dépendent directement des données ou de la clé. L'idée est soit d'uniformiser le temps de calcul de chaque fonction ou soit de rendre l'exécution des fonctions totalement aléatoire en rajoutant des instructions aléatoires afin de complexifier l'analyse du temps d'exécution [Koch 96, Dhem 98].

3.3.4 Les contre-mesures contre les attaques électromagnétiques

Les moyens de défense contre les attaques par analyse électromagnétique sont concentrés soit sur la réduction des émissions électromagnétiques (ramener le niveau d'émission proche de celui du bruit), ou soit sur le brouillage des émissions afin de les rendre inexploitable à toute analyse. La seconde approche est généralement la plus utilisée par les concepteurs de systèmes sécurisés car elle permet d'exploiter les méthodes de protection développées contre les attaques en puissance. En effet, les émissions électromagnétiques étant une image (le spectre du courant pour les émissions induites) de l'activité électrique du composant, la majorité des contre-mesures développées pour rendre inexploitable l'activité électrique du composant en fonction des données, brouillent également les analyses électromagnétiques (cf. § 3.3.5)[Boue04b].

3.3.5 Les contre-mesures contre les attaques en puissance

Les méthodes développées pour implémenter des circuits résistants aux attaques en puissance sont focalisées sur la suppression des corrélations entre les données manipulées et les courbes de courant. Les approches développées contre les attaques temporelles à savoir la symétrie dans l'exécution des instructions, la suppression des dépendances directes entre les données manipulées et les fonctions ou

instructions (saut, branchement, test et comparaison) permet de complexifier les analyses simples du profil de courant.

Les contre-mesures proposées pour lutter contre les attaques en puissance notamment contre les attaques par analyses différentielles du courant peuvent être classées en trois catégories : l'ensemble des méthodes permettant de masquer (cacher) les valeurs intermédiaires d'un algorithme, l'introduction du bruit dans les mesures de courant et les méthodes basées sur la réduction des pics de courant. En présentant les différentes contre-mesures contenues dans chacune de ces catégories, nous allons également indiquer celles qui sont susceptibles d'agir sur les émissions électromagnétiques.

3.3.5.1 Les méthodes de masquage et de duplication

Les contre-mesures basées sur le masquage des données consistent à dissimuler les données manipulées de telle sorte qu'il ne soit plus possible de réaliser des corrélations avec le courant consommé. Ces techniques n'agissent pas directement sur la forme d'onde du courant mais sur la nature des données traitées et plus particulièrement sur la fonction de sélection.

Soit la fonction de sélection D dépendant de la donnée m et d'une partie de la clé k : $D(k, m)$. Le principe de la protection est de remplacer la valeur de m par une valeur m' dépendante de m et d'une valeur aléatoire r . Ainsi au lieu de calculer la fonction initiale $D(k, m)$ on calcule la fonction $D(k, m')$ sur laquelle il est impossible de faire des hypothèses de répartition car on ne connaît pas la valeur de m' . Le masque peut être étendu sur la valeur de k telle que D soit fonction de k' et m' : $D(k', m')$.

La nouvelle valeur m' est calculée en fonction de l'algorithme par des opérations dites de masquage. Elle doit pouvoir d'une part conserver les propriétés de l'algorithme, et d'autre part elle doit permettre de retrouver le résultat initial. Son calcul est donc fonction de l'algorithme utilisé. Les opérations de masquage peuvent être groupées en deux classes [Mess00, Coro00], le masquage binaire pour des algorithmes utilisant uniquement des fonctions binaires comme c'est le cas du *DES* et le masquage arithmétique avec *AES*, le *RSA* etc.

$$\begin{array}{ll} \text{masquage binaire} & m' = m \oplus r \\ \text{masquage arithmétique} & m' = (m - r) \bmod 2^p \end{array}$$

p nombre de bit de m

L'objectif de cette approche est donc de masquer aléatoirement les données d'entrées et les clés dans les calculs de chiffrement (déchiffrement) pour empêcher toute analyse de corrélation.

Dans la méthode par duplication [Goubin99], plusieurs valeurs différentes m_i de m tel que $m = f(m_i)$ avec $0 \leq i \leq 2^{p-1}$ sont utilisées à la place de m . En dupliquant le nombre de messages intermédiaires, on augmente ainsi le nombre d'acquisition de courbe nécessaire pour réussir une attaque et le nombre d'hypothèses possibles des clés. En effet, en considérant les blocs de substitution du *DES*, si toutes les entrées de ce bloc sont dupliquées en deux $m = f(m_1, m_2)$, toute attaque en sortie de ce bloc nécessite de considérer les 2^{12} valeurs possibles des entrées (m_1 et m_2 étant chacune sur 6-bit comme m et k).

Ainsi, les méthodes par masquage et par duplication agissent au niveau logique afin de décorréler le poids de Hamming des données manipulées à celui du courant consommé. Cependant, les protections par masquage restent vulnérables faces à des attaques *DPA* de second d'ordre [Mess00a]. En effet, il est toujours possible d'établir une corrélation entre la donnée masquée m' et la valeur du masque utilisé r par analyse du courant lors des opérations de masquages. Ainsi dans une attaque *DPA* de second d'ordre, on considère dans l'analyse des courbes de courant, plusieurs points ou instants correspondant à la fonction de sélection et au calcul du masque.

Afin de renforcer ce type de contre-mesure, elles sont souvent associées aux méthodes par introduction de bruit [Akka01]. Ces deux types d'approches permettent également de renforcer la protection du circuit contre les attaques électromagnétiques en supprimant les corrélations entre l'activité électrique et les données intermédiaires traitées.

3.3.5.2 Méthodes par injection de bruit

Le principe ici est d'introduire du bruit dans le composant afin de rendre les mesures de courant inexploitable sans pour autant perturber le fonctionnement du circuit. Le bruit peut être introduit soit à l'aide d'un jitter sur les signaux d'alimentation ou le signal d'horloge, soit en utilisant un générateur aléatoire. L'objectif est de rendre l'exécution de certaines fonctions (l'occurrence des données), l'écriture et la lecture des registres aléatoires (accès mémoires aléatoires)[Mull01, Oswa01].

Ce type de contre-mesure agit sur le profil de courant du composant et particulièrement sur les amplitudes et les instants d'occurrences des pics. Soit le courant d'un bloc $I_{bloc}(t)$ tel que défini au paragraphe 3.2.3. Si l'instant et le temps d'exécution du bloc sont rendus aléatoires, alors $I_{bloc}(t)$ est de la forme :

$$I_{bloc}(t)_M = \sum_{t=t_{MIN}}^{t_{MAX}} \sum_{i=0}^{N_{porte_a_t}} I_{porte_i}(t) \quad (3.8)$$

avec $[t_{MIN}, t_{MAX}]$ l'intervalle de temps dans lequel la fonction doit être exécutée et $N_{porte_a_t}$ le nombre de portes qui commutent à l'instant t .

En terme d'amplitude de pics : en rendant l'occurrence des données aléatoires, on diminue le nombre $N_{porte_a_t}$ de portes qui commutent à chaque instant t et on les réparties dans l'intervalle de temps d'exécution de la fonction. Ceci à pour conséquence de réduire l'amplitude du courant à chaque instant t . Cette approche est quasi identique aux méthodes de répartition de l'activité du courant utilisées en conception des circuits à faible consommation [Slim04] et à faible émission électromagnétique [Boue05a]. La figure 3.13 présente un exemple de réduction des amplitudes du courant appliqué sur le bloc **SBOX1**. En décalant l'écriture de certains bits (3 bits) dans les bascules d'entrées (figure 3.13-b), on réduit de près de 50% les pics de courant comparés à un calcul sans décalage (figure 3.13-a) pour une même variation du poids de Hamming ($H=6$). Cela permet de réduire le rapport signal sur bruit et les émissions électromagnétiques.

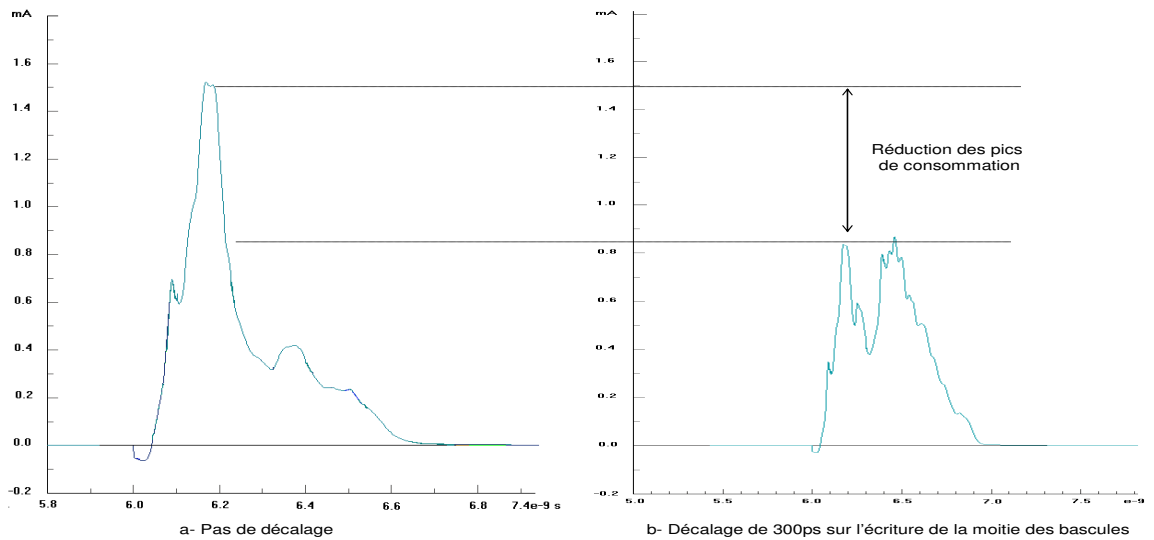


Figure 3.13 : Courant de la *SBOX1* avec et sans décalage implémenté sur l'écriture dans la moitié des registres d'entrées.

Contrairement aux modèles de faible consommation, ici la répartition du courant est aléatoire ce qui permet d'avoir différents instants d'exécution de la fonction.

En terme d'instant d'exécution : plus l'intervalle d'exécution de la fonction est grand, plus il est difficile de localiser l'instant correspondant au calcul du bit (ou des bits) attaqué (bit engendrant la quantité de courant $q_{id_i}(t)$). Ceci a pour conséquence de complexifier la synchronisation pour les acquisitions et le traitement des courbes.

3.3.5.3 Les méthodes par lissage et uniformisation du profil de courant

Elle regroupe toutes les méthodes qui permettent d'une part, de réduire le courant et plus particulièrement les pics de courant et d'autre part, d'uniformiser les formes d'ondes des courants afin de les rendre indépendantes des données. L'objectif est soit de pouvoir réduire les dissymétries dues aux commutations des portes logiques (charge et décharge), soit de les dissimuler. Pour cela on peut les scinder en deux classes: celles qui agissent sur le courant du composant et celles qui agissent sur le profil de courant des portes logiques.

Les techniques qui agissent sur le courant du composant sont basées sur l'utilisation de filtres de courant sur l'alimentation et sur des techniques de découplage des alimentations. L'usage du filtre inséré entre les plots d'alimentation et le cœur de la puce permet de lisser le profil de courant [Rake01]. Par contre les techniques de découplages de l'alimentation externe et interne d'un composant à l'aide des capacités de découplages [Sham00] permet d'uniformiser le profil du courant externe.

Au niveau porte logique, les approches sont focalisées sur la suppression des dissymétries observées lors de la charge et la décharge d'une porte. Tiri dans [Tiri02] propose l'utilisation des portes logiques et des bascules dynamiques de type *SABL* (*Sense Amplifier Based Logic*). Il présente dans [Tiri04], un flot de conception sécurisé basé sur les propriétés de ce type de porte. Elles permettent de garantir quelques soient les commutations en entrées et en sorties, une quantité de courant consommé identique. Cependant, l'utilisation de telles cellules ne serait bénéfique que si le nombre de transitions logiques dans un chemin de données restait invariant quelque soit la nature des données traitées et seulement si les phases de placement et routage n'introduisaient aucune dissymétrie.

Excepté les contre-mesures centrées sur la conception de portes logiques spécifiques qui permettent d'agir sur les dissymétries au niveau électrique, l'ensemble des contre-mesures proposées à savoir : les méthodes de masquage et de duplication, l'introduction de bruit, l'utilisation de générateur aléatoire et l'uniformisation ou le lissage du profil de courant, ne supprime pas l'origine des fuites d'informations, mais permettent de les rendre inexploitable en empêchant toute corrélation entre les données et le courant consommé. Contrairement aux contre-mesures développées en logique synchrone, la logique asynchrone permet d'agir sur les origines des fuites d'informations. En effet, l'utilisation de la logique asynchrone permet de supprimer l'influence des paramètres les plus importants à l'origine des fuites d'information observées sur les circuits intégrés, particulièrement la dépendance de consommation due au poids de Hamming des données et les différences de consommation causées par la nature des commutations (charge et décharge). C'est dans ce contexte que nous présentons la logique asynchrone comme une nouvelle alternative car les propriétés de ce type de logique les rendent particulièrement intéressants pour résister aux attaques par analyse du courant.

3.4 Conclusion

Dans la première partie de ce chapitre, nous avons analysé et présenté les origines des fuites d'information observées sur les circuits intégrés et exploitées dans des attaques de type *SPA* et *DPA*. Les

différents paramètres environnementaux influençant le profil de courant d'un circuit ont été identifiés et classés en fonction de l'importance de leurs effets sur les signaux d'alimentation. Ces observations nous ont permis d'analyser les effets des différentes contre-mesures proposées sur les circuits. En effet dans la seconde partie de ce chapitre, nous avons présenté les contre-mesures existantes et montré leurs impacts sur les différentes causes de fuites d'information permettant de réussir les attaques.

De nos jours, on ne pourrait concevoir un circuit intégré dédié à des applications cryptographiques sans intégrer de contre-mesures contre des attaques matérielles et notamment contre les attaques en puissance. Parmi les différentes contre-mesures développées, nous proposons l'utilisation de la logique asynchrone qui possède des propriétés intrinsèques particulièrement adaptées pour faire face aux attaques non intrusives et plus spécialement aux attaques en courant. Afin d'évaluer son potentiel, nous allons dans le prochain chapitre présenter la logique asynchrone et ses potentialités en matière de sécurité matérielle.

CHAPITRE IV

LA LOGIQUE ASYNCHRONE ET LA SECURITE MATERIELLE : EVALUATION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS

4.1 Introduction

Ce chapitre est consacré à l'évaluation de l'alternative asynchrone. Les propriétés des circuits asynchrones, à savoir une consommation indépendante des données manipulées, l'absence d'un signal d'horloge, de faibles pics de courant (faibles émissions électromagnétiques), l'utilisation d'un séquençement de nature flot de données basé sur un protocole de communication de type poignée de mains, les prédisposent à mieux résister aux attaques comparé à leur équivalent synchrone. Cette étude a donc pour enjeu l'évaluation des propriétés des circuits implémentant la logique asynchrone dans le but de définir des méthodologies de conception de circuits intégrés résistants aux attaques en puissance.

Nous allons, dans la première partie, introduire la technologie asynchrone en se focalisant sur les propriétés liées à la protection des circuits. Ensuite, nous présentons une évaluation des différents types de logique asynchrone afin de déterminer le style de logique asynchrone adéquate pour résister aux attaques. Pour terminer, des résultats des attaques en puissance réalisées sur un cryptoprocasseur asynchrone sont présentés.

4.2 La logique asynchrone

Lorsqu'on parle de circuits synchrones, on fait référence à des circuits qui sont contrôlés et séquencés par un signal périodique, globalement distribué appelé horloge. Les circuits asynchrones représentent une autre classe de circuits dont le contrôle et le séquençement sont assurés par une autre méthode.

En asynchrone, il n'existe pas à priori, de relation temporelle entre les événements au sens large (données ou contrôles). Les circuits asynchrones sont des circuits capables de gérer des signaux asynchrones entre eux et qui ont un comportement bien déterminé. On ne peut avoir un événement (résultat) en sortie d'un bloc, que s'il y a eu un événement (présence des données) sur l'une des entrées, et cela quelques soient les instants d'occurrences. Les systèmes dits asynchrones fonctionnent donc avec la seule connaissance de l'occurrence des événements, sans connaissance de l'ordre. Ce fonctionnement est comparable à celui des systèmes flot de données dans lequel il suffit de décrire l'enchaînement des événements et des opérations sous la forme de graphes de transitions de signaux. A la différence d'un système synchrone qui possède une forte contrainte temporelle, et par le fait que tous les éléments du système évoluent selon le signal d'horloge, un système asynchrone présente une synchronisation localisée où le déclenchement des actions dépend uniquement de la présence des données à traiter.

4.2.1 Les opérateurs asynchrones

Les opérateurs asynchrones sont généralement considérés comme des cellules réalisant certaines fonctions de calcul et de communication avec l'environnement à travers des canaux. Les échanges à travers les canaux peuvent être des données ou des signaux de synchronisation.

Ces opérateurs, sont caractérisés par 4 paramètres :

- Le **temps de latence**, qui correspond au chemin de traitement entre l'entrée et la sortie.
- Le **temps de cycle** : qui caractérise le temps qui sépare la validation de deux entrées successives. Il représente la bande passante de l'opérateur.
- La **profondeur du pipeline**, qui définit le nombre maximum de données ou d'informations que l'opérateur peut mémoriser (profondeur maximale du pipeline).

- Le *protocole de communication*, qui détermine la façon dont des opérateurs asynchrones échangent des informations. Il assure la détection de l'information en entrée et assure la génération d'une signalisation indiquant d'une part, qu'une information a été consommée en entrée et d'autre part, qu'une information est disponible en sortie.

4.2.2 Le contrôle local

En asynchrone, le transfert d'information est géré par une signalisation adéquate. Afin d'assurer une synchronisation des échanges entre les opérateurs connectés qui échangent des données indépendamment des autres opérateurs auxquels ils ne sont pas connectés, le contrôle local, doit assurer les fonctions suivantes :

- surveiller les communications entrantes.
- déclencher le traitement si toutes les informations sont disponibles.
- produire les sorties.

Pour ce faire, il faut utiliser une signalisation bidirectionnelle. Toute action de communication doit être acquittée par le récepteur, afin que l'émetteur puisse émettre à nouveau. Ce type de communication est dit à requête – acquittement (figure 4.1).

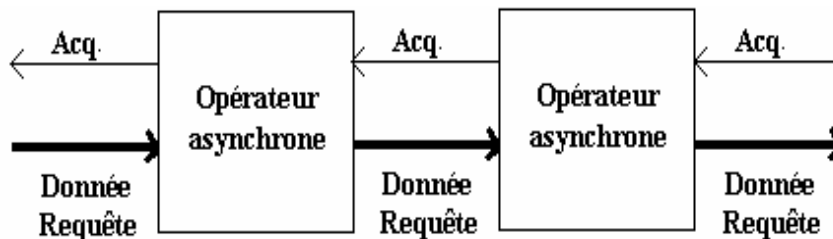


Figure 4.1: Communication de type requête – acquittement

Ces types de circuits sont vulnérables face aux aléas car le temps n'est pas discrétisé. Une attention particulière est donc portée à la conception des circuits asynchrones afin qu'ils soient exempts d'aléas.

4.2.3 Protocole de communication

Il existe deux principaux protocoles de communication, le protocole 2 phases (*NRZ* pour *Non Retour à Zéro*) et le protocole 4 phases (*RZ* pour *Retour à Zéro*). Le protocole le plus utilisé pour des raisons de simplicité d'implémentation [Rena00] est le protocole 4 phases qui se décompose comme décrit sur la figure 4.2.

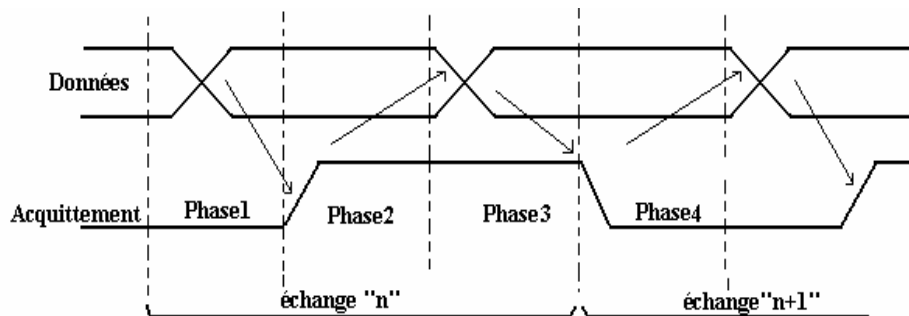


Figure 4.2: Principe du protocole 4 phases

- **Phase 1** : c'est la première phase active du récepteur qui détecte la présence de nouvelles données, effectue le traitement et génère le signal d'acquittement.
- **Phase 2** : c'est la première phase active de l'émetteur qui détecte le signal d'acquittement et émet des données invalides (retours à zéro).
- **Phase 3** : c'est la deuxième phase active du récepteur qui détecte le passage des données dans l'état invalide et place le signal d'acquittement dans l'état initial (retour à zéro).
- **Phase 4** : c'est la deuxième phase active de l'émetteur, qui détecte le retour à zéro de l'acquittement. Il est alors prêt à émettre de nouvelles données.

4.2.4 Codage des données

L'adoption d'un codage particulier, permet de résoudre le problème posé par la détection de la présence des données et de la production du signal de fin de traitement. Le protocole utilise un codage double rail pour chaque bit (deux fils par bit). Avec deux fils par bit, on dispose de 4 états possibles pour

représenter les valeurs logiques «0» et «1». Deux codages sont communément adoptés, l'un utilisant trois états, bien adapté au protocole 4 phases et l'autre utilisant quatre états, bien adapté au protocole 2 phases.

Pour le codage trois états, un fil prend la valeur 1 pour coder une valeur logique «1» et l'autre fil prend la valeur 1 pour coder la valeur logique «0». L'état 11 est interdit et l'état 00 représente l'invalidité d'une donnée. Ainsi pour passer d'une valeur à une autre, on passe toujours par l'état invalide (figure 4.3), ce qui garantit le passage d'un état à un autre par le changement unique d'un bit.

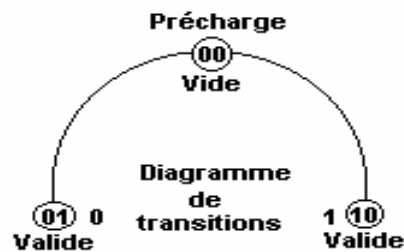


Figure 4.3: Codage trois états

Une technique connue sous le nom de "Bundled Data" ou données groupées permet de limiter le nombre de fils pour transmettre une donnée multibit (figure 4.4).

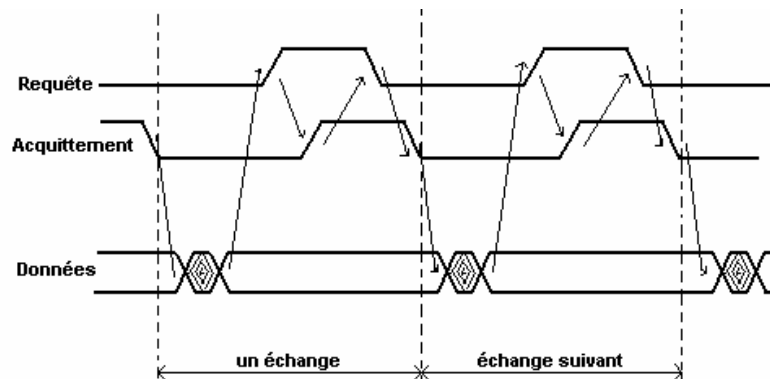


Figure 4.4: Protocole de communication 4 phases "données groupées" ou Bundled Data

Dans cette technique, on crée explicitement des signaux de contrôle uniques, dénommés requête et acquittement, en séparant l'information de contrôle de l'information de donnée proprement dite. Le signal de requête ressemble à une horloge locale qui est utilisé pour déclencher la mémorisation ou le traitement de données qui lui sont associées. L'avantage de ces simplifications est le codage des bits de données sur

un fil. Cependant, il existe une hypothèse temporelle qui doit garantir que l'occurrence du signal de contrôle succède la stabilité des données, et ceci au niveau du récepteur.

4.2.5 Signaux de fin de calcul

Les protocoles décrits ci-dessus, utilisent un signal d'acquiescement que chaque opérateur doit produire. La génération de ce signal, représente une complexité en plus, nécessaire à la réalisation d'un circuit asynchrone. Plusieurs solutions existent, la plus utilisée aujourd'hui est celle où on insère une hypothèse temporelle. Le temps de la chaîne critique est mesuré pour dimensionner le retard à insérer entre la requête et l'acquiescement. Par ailleurs, certaines fonctions possèdent des caractéristiques qui permettent de détecter la fin de traitement.

4.2.6 Les différentes classes de circuits asynchrones

Les circuits asynchrones sont généralement classés suivant les hypothèses faites sur le modèle de délai et en fonction de l'environnement de fonctionnement adopté. La figure 4.5 présente la terminologie habituellement utilisée pour nommer les circuits asynchrones.

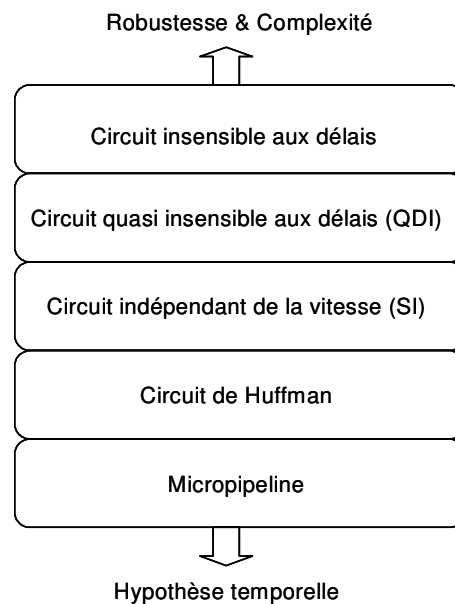


Figure 4.5: Différentes classes des circuits asynchrones

Plus le fonctionnement du circuit respecte fondamentalement la notion d'asynchronisme, plus le circuit est robuste et plus il est complexe.

Conformément à l'outil de conception automatique de circuits asynchrones *TAST* développé au sein du groupe *CIS* du laboratoire *TIMA* [Din02], nous nous intéresserons uniquement à deux classes des circuits asynchrone : les circuits *QDI* et les circuits micropipeline.

4.2.7 Les circuits Quasi Insensibles aux Délais (QDI)

Les circuits *QDI* sont purement asynchrones. Leur fonctionnement est correct quels que soient les temps de propagation dans les fils et les éléments logiques qui composent le circuit. Seules quelques fourches font l'objet d'une hypothèse temporelle. On les nomme fourches isochrones. Une fourche est dite isochrone si les temps de propagation dans les branches de la fourche sont égaux. Le modèle de délai dans ce type de circuit est non borné.

Ainsi, un circuit *QDI* répondra toujours correctement à une sollicitation externe indépendamment du temps nécessaire au calcul. Ceci impose donc au récepteur d'un signal de toujours informer l'expéditeur que l'information a été reçue. Les circuits récepteurs doivent donc être capables de détecter la réception d'une donnée et/ou la fin de son traitement. Les circuits émetteurs quant à eux doivent attendre un compte rendu avant d'émettre une nouvelle donnée.

Cette classe de circuit asynchrone utilise généralement un codage de données de type *1 parmi n* avec un protocole de communication 4 phases.

4.2.8 Les circuits micropipeline

Les circuits micropipelines sont composés de deux parties, une partie de contrôle insensible aux délais et une partie chemin de données qui a un délai borné. Les parties de contrôles (insensibles aux délais) commandent les chemins de données conçus en utilisant le modèle de délai borné. La structure de base de ces circuits correspond à une file de type FIFO composée de portes de Muller (cf. § 4.2.9) connectées tête-bêche (Figure 4.6).

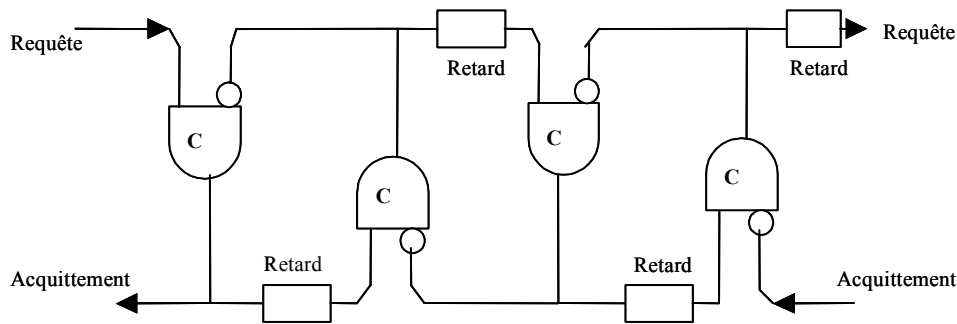


Figure 4.6: Structure de base des circuits Micropipelines

Si on suppose tous les signaux à zéro initialement, la mise à l'état haut de la Requête provoque une transition positive sur l'acquittement qui se propage également à l'étage suivant. Le deuxième étage après un certain délai produit une transition positive qui se propage vers l'étage suivant et d'autre part revient à l'étage précédent, l'autorisant à transmettre une transition négative. Les transitions de signaux se propagent dans la structure tant qu'elles ne rencontrent pas une cellule occupée. Cette structure qui implémente un protocole deux phases est bien insensible aux délais.

On peut l'enrichir d'opérateurs de mémorisations et de traitements combinatoires (figure 4.7).

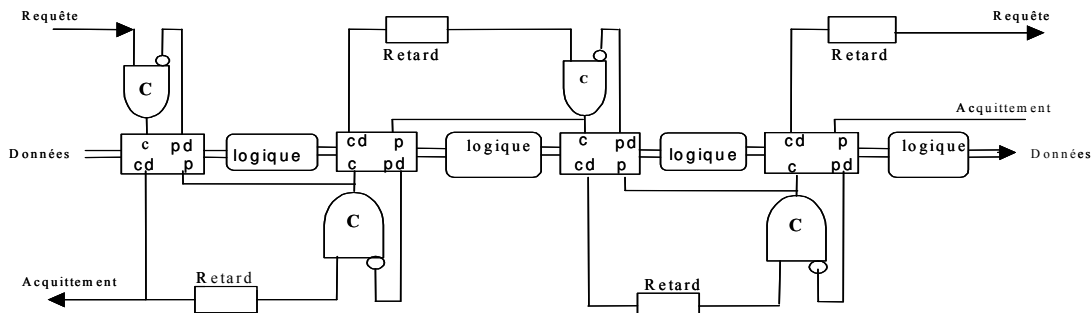


Figure 4.7: Structure Micropipeline avec traitement et registre

Dans cette architecture, les registres capturent les données sur occurrence d'un événement sur le signal **C** (*capture*). Ils produisent le signal **Cd** (*capture done*) une fois la donnée mémorisée dans le registre. Durant cette phase, la donnée précédemment mémorisée est maintenue en sortie du registre. Lorsque **P** (*pass*) est actif, le registre laisse passer la donnée en sortie. Le signal **Pd** (*pass done*) signale que le registre est transparent. Ainsi lors d'une occurrence du signal **Requête**, les données d'entrées sont stockées dans le premier étage. La **Requête** est alors transmise vers l'étage suivant qui stocke le résultat transmis par la logique du premier étage. La propagation de la **Requête** est retardée de façon à vérifier que

la logique combinatoire est stabilisée avant la capture du résultat dans l'étage suivant. Une fois la capture effectuée dans le deuxième étage, le premier registre est rendu passant ce qui permet le traitement dans le premier étage et autorise la capture d'un nouvel événement sur la *Requête*. Le protocole de communication utilisé est de type 'données groupées'.

4.2.9 La porte de Muller

Bien qu'il soit possible d'utiliser exclusivement des cellules de bibliothèques synchrones, les circuits asynchrones sont constitués de cellules spécifiques telles que les portes de Muller. Une porte de Muller ou C-élément est une fonction qui copie le niveau logique des entrées sur la sortie, lorsque ceux-ci sont égaux. Vis-à-vis des transitions, la porte de Muller implémente un rendez-vous. Une transition est propagée en sortie si et seulement si une transition a eu lieu sur chacune des entrées. Cette fonction peut être définie pour un nombre d'entrée quelconque. La figure 4.8 présente la table de vérité, le symbole et un exemple d'implémentation d'une Muller à 2 entrées.

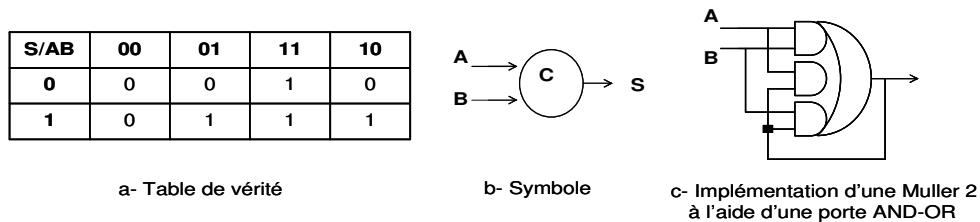


Figure 4.8: Table de vérité, symbole et exemple d'implémentation d'une porte de Muller à 2 entrées à l'aide d'une porte complexe.

4.2.10 Conception des circuits asynchrones

La conception des circuits asynchrones reste encore le domaine de spécialistes utilisant des outils semi-automatiques. Sa diffusion dans le secteur industriel n'est possible que si des outils avec un niveau d'automatisation égal aux outils disponibles actuellement pour les circuits synchrones sont développés et que des ingénieurs soient formés à leur utilisation. Pour cela, le groupe *CIS* du laboratoire *TIMA* développe *TAST* qui permet de décrire et de synthétiser des circuits asynchrones (cf. § Introduction).

4.3 Les circuits asynchrones et les attaques non intrusives

De part leur séquençement de nature flot de données, les circuits asynchrones peuvent jouer un rôle important face aux attaques en puissance. Le fait de ne pas utiliser une horloge globale rend quasiment impossible toute attaque par analyse temporelle et permet d'augmenter les difficultés de synchronisation. L'utilisation de protocole de communication de type poignée de mains (Handshake) permet non seulement aux circuits asynchrones de mieux répartir la consommation du courant dans le temps, mais également de réduire considérablement les appels de courant (pics de courant) ainsi que l'émission électromagnétique (en lissant la forme d'onde du courant).

Cependant certaines classes de circuits asynchrones restent très vulnérables à ces types d'attaques.

4.3.1 Choix du type de circuit asynchrone

Parmi les deux types de circuits asynchrones synthétisés par l'outil *TAST*, les circuits micropipelines sont plus sensibles que les circuits *QDI* aux attaques en puissance.

En effet, comme présenté plus haut, les circuits micropipelines sont constitués de deux parties, une partie contrôle (insensible aux délais) et une partie chemin de données (avec un modèle de délai borné) qui peut comprendre de la logique combinatoire. Cette partie est identique aux blocs logiques des circuits synchrones et présente donc les mêmes faiblesses.

De plus, dans le cas des circuits synchrones, la mesure ou l'analyse des courbes de courant est fortement brouillée ou perturbée par les effets de la consommation des arbres d'horloge qui introduisent des signaux parasites. L'absence d'un signal d'horloge global dans les circuits micropipelines rend l'analyse des courbes de consommation encore plus facile car elle reflète directement l'activité des blocs combinatoires du composant. En supprimant la consommation due à l'horloge, on limite le bruit dans les lignes d'alimentation en augmentant ainsi le rapport signal sur bruit.

4.3.2 Les circuits QDI et les analyses en puissance

Notre approche consiste à exploiter les caractéristiques du codage de données, du protocole de communication et de la structure des chemins de données des circuits asynchrones *QDI*.

4.3.2.1 Le Codage de données en p parmi n

Le type de codage de données adopté dans les circuits asynchrones *QDI* permet de garantir un poids de Hamming constant quelques soient les données manipulées. Nous avons montré dans le chapitre précédent que le poids de Hamming est le facteur logique le plus important sur l'origine des fuites d'information à travers le courant consommé. Par conséquent, en le rendant constant, on supprime toute dépendance entre les données manipulées et le courant consommé lié à ce facteur. Un exemple de codage d'un digit (1 bit) en p parmi n avec ($p=1$ et $n=2$) est présenté sur la figure 4.9.

| 1 bit ($P=1$) | Codé sur 2 rails ($N=2$) |
|--------------------|-------------------------------|
| 0 | 01 |
| 1 | 10 |
| État invalide | 00 |
| Non utilisé | 11 |

Figure 4.9: Codage en double rail d'un digit

Ce type de codage, associé au codage trois états présenté au paragraphe 4.2.4, permet de maîtriser le nombre de transitions logiques dans une structure. En partant d'un état invalide (**00**) à un état valide (**01**) ou (**10**), on effectue toujours une seule transition sur l'un des rails (le résultat est le même en partant d'un état valide à un état invalide).

De nombreuses contre-mesures basées sur le codage p parmi n , ont été présentées et développées dans différents types de circuits et logiques. Simon Moore [Moor 02, Moor03] montre comment l'utilisation d'un codage des données de type 1 parmi n permet :

- de réduire les dépendances entre les données manipulées et le courant consommé,
- de résister aux attaques par injection de fautes en utilisant l'état invalide (**11**) du codage de données pour générer des alarmes en cas de détection de fautes,
- de disperser les dépendances temporelles.

Dans la même optique, l'outil de synthèse logique de circuits asynchrones *Balsa* (basé sur l'outil *Tangram* de *Philips*) développé à l'université de Manchester et initialement destiné à la synthèse logique

des circuits asynchrones micropipelines, a été modifié pour la sécurité des circuits contre les attaques en puissance. Les modifications ont portées essentiellement sur l'intégration d'un codage *p parmi n* [Plan02].

Cependant, ce type de codage n'est pas exclusivement utilisé dans un environnement asynchrone, Sokolov propose dans [Soko04] l'utilisation d'un codage double rail (*1 parmi 2*) dans un environnement synchrone. Pour cela, il a développé un outil qui permet de convertir une description *RTL (Register Transfer Level)* d'un circuit synchrone en une netlist synchrone double rail afin de garantir un poids de Hamming constant.

4.3.2.2 Le protocole de communication 4 phases

Contrairement aux circuits synchrones dans lesquels la consommation du courant dépend de la valeur précédente (de l'état logique précédent), l'utilisation d'un protocole 4 phases assure une remise à zéro de tous les nœuds logiques avant chaque phase de calcul. Ceci, permet au concepteur de tels circuits, de précisément contrôler la nature des transitions. En effet, de l'état invalide on part toujours vers un état valide par une transition positive (0 vers 1), c'est la phase d'évaluation des données. Dans la phase de remise à zéro la transition est orientée dans le sens inverse (1 vers 0).

Parce que ce type de logique doit être fonctionnel sans aléa, tous les appels de courant dû aux phénomènes transitoires des signaux dans un circuit synchrone, sont éliminés dans les circuits asynchrones *QDI*.

4.3.2.3 Chemins de données équilibrés

Les circuits asynchrones *QDI* offrent la possibilité de contrôler avec précision le nombre de transitions logiques dans chaque bloc de calcul. Cela permet, selon le cas, d'équilibrer les chemins de données afin de toujours avoir une consommation indépendante du nombre de transitions logiques. Par exemple, l'opération logique *Xor* (ou exclusif) représente l'opération la plus sensible dans les systèmes cryptographiques symétriques, car elle manipule directement la clé. La figure 4.10 schématise l'implémentation en double rail d'un opérateur *Xor* utilisant un protocole 4 phases. On assure la commutation d'un nombre constant de portes, quelques soient les données.

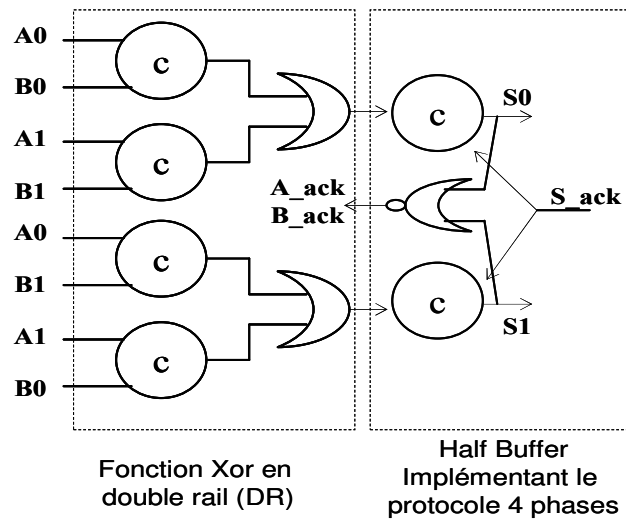


Figure 4.10: Opérateur logique *Xor* en double rail implémentant un protocole 4 phases

Si l'élément logique ne présente pas une structure équilibrée, il est possible d'ajouter des portes pour forcer la symétrie. Le schéma de la figure 4.11 illustre l'implémentation d'une porte *And* double rail équilibrée.

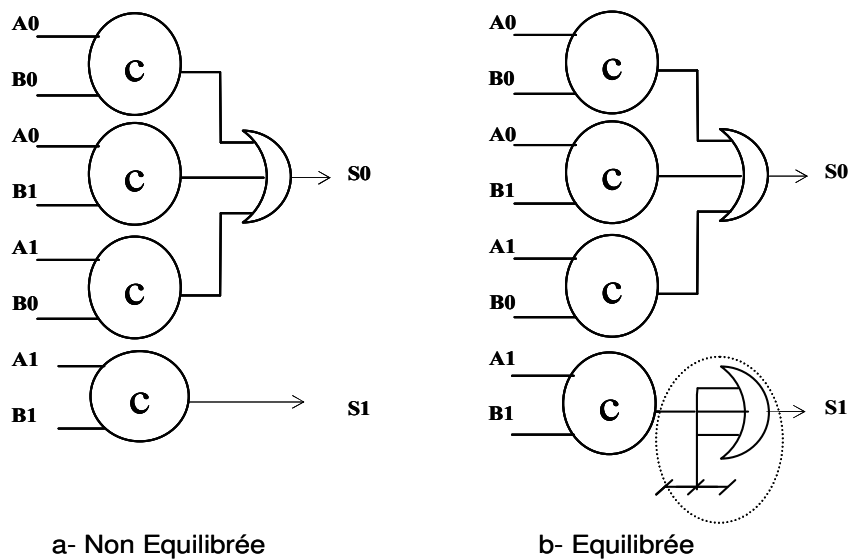


Figure 4.11: Porte logique *And* en double rail

Nous savons aujourd'hui implémenter tout type de fonctions de façon équilibrée (fonction de calcul, registres, machines à états). On peut donc très finement équilibrer le nombre de transitions relatif

aux traitements de zéros et de uns. Le principal résultat obtenu est qu'il est toujours possible de concevoir un circuit **QDI** seulement composé de chemins logiques équilibrés, c'est-à-dire qui effectue un traitement donné avec un nombre de transitions logiques constant quelques soient les valeurs traitées. Ce type de logique permet donc de concevoir des circuits dont le nombre de transitions logiques ne varie pas en fonction des données traitées. En rendant l'activité électrique du circuit indépendant des données, on supprime également les dépendances possibles avec les émissions électromagnétiques. Ainsi, les circuits **QDI** sont théoriquement plus résistants aux attaques **DPA** et électromagnétiques.

Le tableau 4.1 permet de résumer l'ensemble des propriétés de la logique asynchrone **QDI** susceptibles d'améliorer la résistance des circuits face à des attaques en puissance. En reprenant le tableau 3.2 (cf. § 3.2.4) illustrant l'importance des effets des paramètres environnementaux sur le courant dans un circuit synchrone, nous pouvons supprimer, dans le même tableau, et en fonction des analyses réalisées sur les propriétés des circuits asynchrones **QDI**, les paramètres n'influençant plus le profil de courant de ce type de circuit.

Tableau 4.1 Suppression des paramètres n'influençant plus le profil de courant en logique asynchrone **QDI**.

| | Analyses au niveau logique | Analyses au niveau électrique | | | | | |
|----------------------------|---|-------------------------------------|----------------|-----------------------|--------------------------|-----------------|----------------------|
| | Nombre de transitions logiques: poids de Hamming | Commutation: charge/ décharge | Rampe lente | Capacité en sortie | Structure de la porte | Rampe rapide | Délai de la porte |
| Amplitude des pics | /// | /// | /// | /// | /// | — | — |
| Décalage temporel des pics | /// | /// | /// | ✓ | /// | /// | — |

// Grande variations ✓ Variations moyennes — Indépendant

Une étude réalisée sur la signature électrique des circuits **QDI**, présentées dans le chapitre V, a permis de déterminer les influences des paramètres restant du tableau sur le profil de courant.

4.4 Evaluation de la logique asynchrone *QDI*: Cas d'un cryptoprocasseur asynchrone

Dans le but d'avoir d'une part, une première évaluation sur silicium du niveau de sécurité qu'il est possible d'obtenir avec la logique asynchrone *QDI* et de pouvoir d'autre part, comparer en terme de résistance le niveau de sécurité des circuits *QDI* par rapport a un circuit synchrone, nous avons réalisé un cryptoprocasseur asynchrone *QDI* implémentant l'algorithme du *DES* [Boue02a]. Le cryptoprocasseur synchrone présenté au chapitre II (cf. § 2.7.1) est utilisé comme référence.

4.4.1 Architecture du cryptoprocasseur

L'architecture globale du circuit (figure 4.12) a été définie en partant des spécifications de l'algorithme du *DES* et de l'environnement de fonctionnement du circuit. Le circuit est constitué de trois grands blocs : un banc de registres, les interfaces et le bloc de chiffrement et déchiffrement.

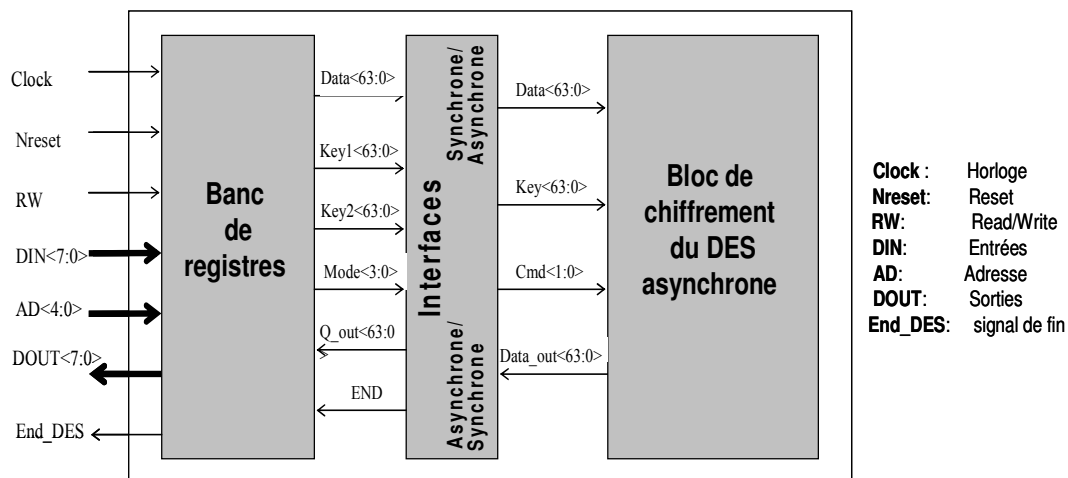


Figure 4.12: Architecture globale du cryptoprocasseur

4.4.1.1 Le banc de registres

Le banc de registres est composé d'un registre de 5-bit (registre de mode) et de 32 registres de 8-bit repartis comme suit :

- 8 registres de 8-bit pour les données.
- 16 registres de 8-bit pour les deux clés nécessaires aux *triples DES*.

- 8 registres de 8-bit pour les sorties.

Le registre de mode permet de configurer le mode opératoire du cryptoprocresseur (*simple* ou *triple DES* en chiffrement/déchiffrement) et de déclencher une phase de chiffrement/déchiffrement. Il contient également un registre dont la valeur permet d'indiquer la fin du processus de calcul. La valeur de ce registre, correspond au signal '*End_DES*' en sortie du composant. Ce signal, et celui de déclenchement du calcul, permettent de synchroniser les acquisitions.

4.4.1.2 Les Interfaces

Afin de faciliter les communications et le contrôle avec un environnement externe standard (FPGA, Microcontrôleur), des interfaces ont été implémentées. On distingue deux types d'interface, les interfaces synchrones/asynchrones et les interfaces asynchrones/synchrones (figure 4.13).

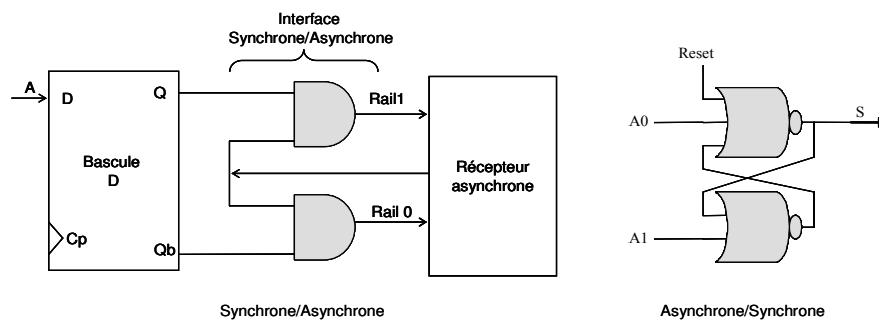


Figure 4.13: Interface sur 1 bit

L'interface synchrone/asynchrone est essentiellement constituée de portes *And*. Dans ce type d'interface, on utilise les sorties *Q* et *Qb* des registres pour générer du double rail. Un signal d'échantillonnage est utilisé sur toutes les portes *And* pour assurer le déroulement du protocole 4 phases. Ce signal est le signal d'acquiescement du bloc qui reçoit la donnée. Par contre, les interfaces asynchrone/synchrone sont composées de registre *RS* avec un signal d'initialisation.

4.4.1.3 le Bloc de chiffrement asynchrone

Ce bloc représente le cœur du *DES* qui implémente ses différentes fonctions de chiffrement/déchiffrement. Comme on peut le voir sur la figure 4.14, il est composé de trois sous blocs distincts : la boucle de chiffrement/déchiffrement, la boucle de gestion des clés et un compteur pour le contrôle. Ces trois sous blocs prennent respectivement en entrée la donnée à chiffrer/déchiffrer, la clé et le signal Crypt/Decrypt signalant le mode ou le type de calcul à effectuer (chiffrement ou déchiffrement).

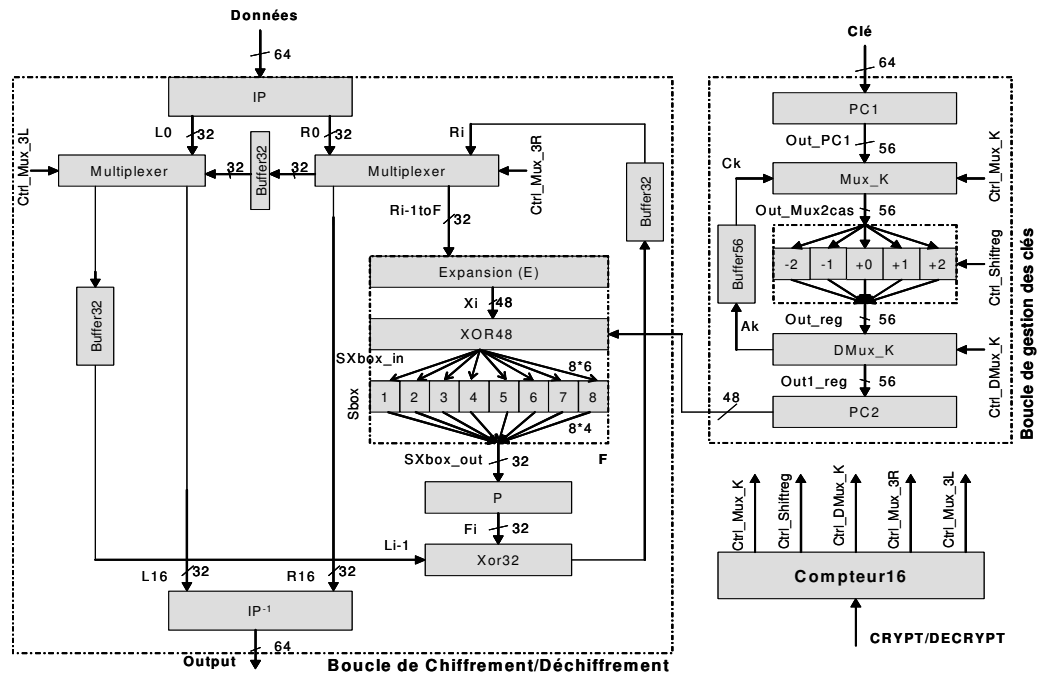


Figure 4.14 : Architecture du Bloc de chiffrement asynchrone

Conformément aux objectifs de sécurité, tous les blocs implémentés dans cette architecture ont des chemins de données réguliers sans aucun test ou branchement conditionnel.

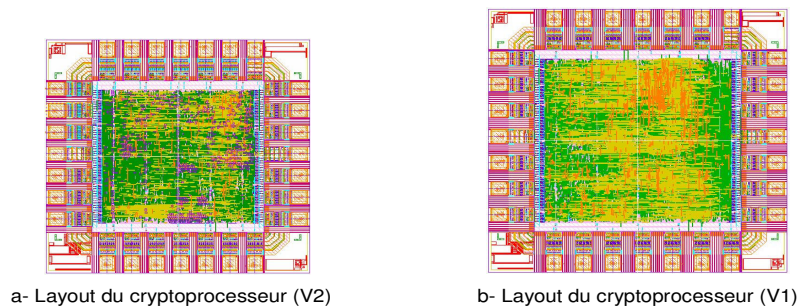
De cette architecture nous avons réalisé deux versions de circuits* :

- une première version n'intégrant pas d'optimisation logique et de pipeline (les blocs Buffer56 et Buffer32 ont été supprimés). Cette version (V1) a été générée par l'outil TAST.
- une seconde version (V2) intégrant du pipeline et une optimisation logique manuelle.

* Ces circuits ont été réalisés en collaboration avec le CEA-LETI et STMicroelectronics dans le cadre du projet (MEDEA+) ESP@SSIS (Enhanced Smartcard Platform for Accessing Securely Services of the Information Society). Ce projet était focalisé sur la mise en place des nouvelles plateformes matérielles pour une nouvelle génération de carte à puces pouvant supporter les besoins et les contraintes des marchés en termes de sécurité et de diversification de services (e-commerce, TV etc.)

4.4.2 Caractéristiques des circuits

Les circuits ont été fabriqués en technologie $0.18 \mu\text{m CMOS}$ de *STMicroelectronics*. Aucune porte spécifique dédiée à la logique asynchrone n'a été utilisée [Mauri03]. Toutes les cellules asynchrones utilisées sont implémentées uniquement à l'aide de cellules standard de la bibliothèque *HCMOS8* [Rig02]. La figure 4.15 présente le layout et les principales caractéristiques des circuits comparés au cryptoprocasseur synchrone.



| Technologie | CMOS 0.18 μm (HCMOS8) de STMicroelectronics 6-LM, Tension Alimentation 1.8 V | | | |
|----------------------------|--|---------------------------------------|--------------------|--|
| Circuits | Descriptions | Surface / Portes Equivalentes | Courant (Moyen) | Vitesse (1.8 V) |
| DES Synchrone | Pas de triple DES | 0.14 mm ² 11400 gates | 2mA | 17 Tclk 2 μs |
| DES Asynchrone (V1) | Pas de Pipeline Pas d'optimisation | 0.71 mm ² 57,7 Kgates | 6 mA | 1400 ns Simple DES 4230 ns Triple DES |
| DES Asynchrone (V2) | Pipeline | 0.426 mm ² 34.65 Kgates | 23 mA | 207 ns Simple DES 650 ns Triple DES |

c- Caractéristiques des cryptoprocasseurs asynchrones et comparaison avec le cryptoprocasseur synchrone

Figure 4.15: Layout et caractéristiques des circuits

La surface du circuit asynchrone *QDI* est 4 fois plus importante que celle du synchrone. Il est bien connu que la logique asynchrone *QDI* est coûteuse en terme de surface. Toutefois, des solutions de réduction de la surface sont en cours d'évaluation dans le groupe de recherche *CIS* du laboratoire *TIMA* avec notamment l'outil *TAST*. Deux aspects sont ciblés, l'utilisation de cellules spécifiques asynchrone [Mauri03, Mauri03a] et une optimisation logique plus agressive pendant la phase de synthèse [Breg04]. L'objectif est de ramener la surface à un facteur 2, comparé au synchrone.

Malgré une surface plus importante, le circuit asynchrone consomme moins et est plus rapide que son équivalent synchrone. En prenant pour référence le même temps de calcul (2 μs) pour les deux circuits (synchrone et asynchrone V1), on est contraint d'alimenter la version asynchrone (V1) à 1.1 volts,

réduisant le courant moyen à 1 mA. Ainsi, pour un même temps de calcul, la version asynchrone (V1) consomme deux fois moins que la version synchrone. Comparé au synchrone, les circuits asynchrones peuvent être alimentés à 0.6 volts. A cette tension, le circuit (V2) consomme 0.45 mA pour un temps de chiffrement de 4 μ s et le circuit (V1) consomme 50 μ A pour un temps de chiffrement de 33 μ s. La figure 4.16 représente le temps et le courant consommé en fonction de la tension des circuits asynchrones.

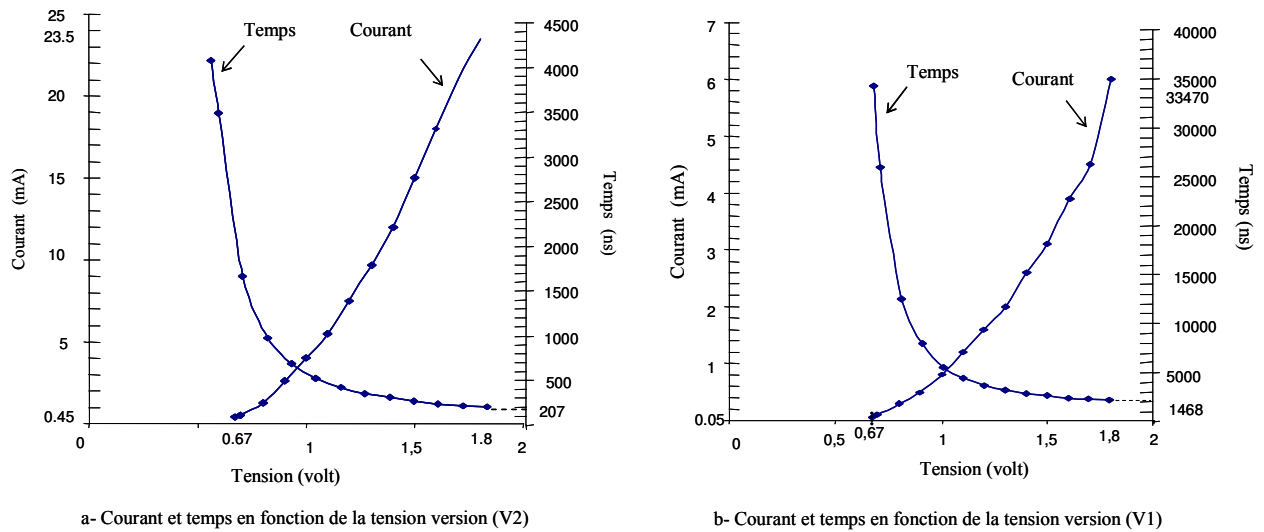


Figure 4.16: Temps de chiffrement et courant consommé en fonction de la tension

En terme d'émissions électromagnétiques, le circuit asynchrone (V1) présente un spectre en fréquences moins large que celui de son équivalent synchrone. Afin de les comparer, nous avons pris comme critère commun aux deux circuits le temps de calcul. La figure 4.17-a représente les profils de courants des circuits effectuant un chiffrement en 2 μ s et leurs spectres en fréquence sont représentés sur la figure 4.17-b. Le spectre du circuit asynchrone (V1) s'étend sur 45 Mhz et présente un seul pic important correspondant à la fréquence d'exécution de chaque boucle de chiffrement et de génération des sous clés. En effet, chaque itération s'effectue en 0.125 μ s ce qui fait 0.0625 μ s pour chacune des deux boucles. Cette valeur correspond à la fréquence de 16 Mhz illustrée sur le spectre.

Par contre le spectre du circuit synchrone est six fois plus large (300 Mhz) et présente plusieurs pics importants parmi lesquels on peut observer le pic à 16 Mhz correspondant à la moitié de la fréquence du signal d'horloge.

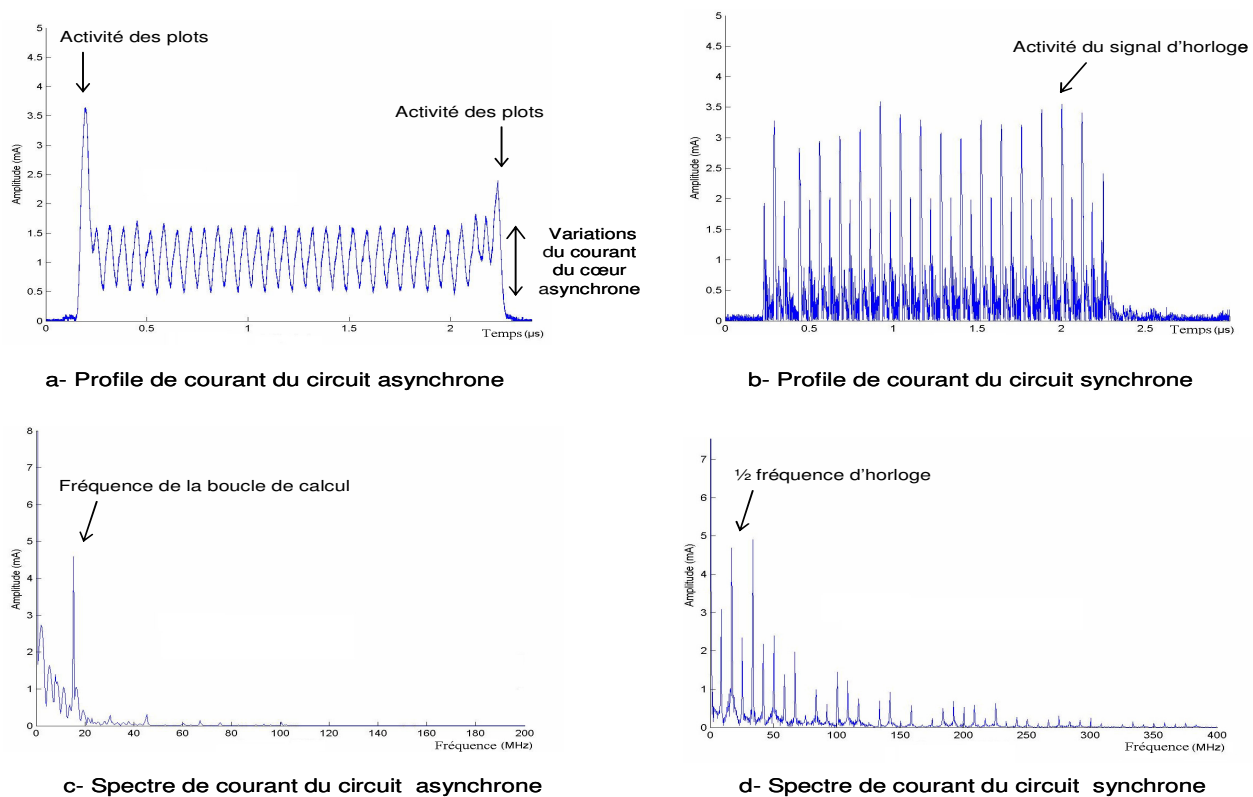


Figure 4.17: Profils et spectres du courant des cryptoprocresseur asynchrone et synchrone

La flexibilité offerte par le circuit asynchrone *QDI* en terme de tension d'alimentation permet de réduire considérablement les émissions électromagnétiques et les pics de courant. Comme le spectre de l'activité électrique ne varie pas en fonction des données manipulées, les caractéristiques d'émission et de susceptibilité de tels circuits sont statiques et peuvent être définies par le concepteur. Ceci a pour conséquence d'une part de faciliter le respect des normes de compatibilité *CEM* et d'implémenter des filtres de courant efficace afin de réduire les émissions [Boue04b].

4.4.3 Résultats des attaques en puissance

Afin de s'assurer de l'équilibrage logique de tous les chemins de données du bloc asynchrone et afin d'avoir un nombre de transitions logiques indépendant des données, nous avons, dans un premier temps, mesuré pour un nombre de vecteurs de test donnés, le nombre de transitions logiques lors du calcul d'un chiffrement. Les vecteurs de test ont été pris de sorte à couvrir tous les poids de Hamming des signaux d'entrées pour une clé fixe. Le tableau 4.2 représente le résultat obtenu. Quelques soient les clés

ou les données utilisées, le nombre de transitions logiques est constant contrairement au circuit synchrone (cf. § 3.2.3).

Tableau 4.2 Variations du nombre de transitions en fonction des données dans le circuit (V2).

| Données | Quelque soit la donnée |
|--------------------------------|------------------------|
| Nombre de transitions logiques | 72512 |

Les analyses *DPA* sur les circuits ont été faites en collaboration avec le *CEA-LETI* dans le cadre du projet MEDEA+@Espass-is [Robin03]. Afin de pouvoir évaluer la résistance du circuit asynchrone en fonction de son équivalent synchrone, nous avons défini un critère de comparaison basé sur le nombre de courbes nécessaires à la réussite d'une attaque. En effet, nous avons montré dans le chapitre II (cf. § 2.2) l'importance d'utiliser un nombre de courbes suffisant pour améliorer le rapport signal sur bruit et pour distinguer le pic *DPA*. Dans un premier temps, nous prenons un nombre considérable de courbes pour réussir l'attaque. Ce nombre est réduit par la suite jusqu'à ce que le pic *DPA* ne soit plus identifiable par rapport aux autres pics. La valeur ainsi obtenue correspond au nombre minimum de courbes nécessaire à la réalisation de l'attaque. Par exemple, la figure 4.18-a présente une courbe *DPA* obtenue avec un ensemble de $6 \cdot 10^6$ courbes sur le circuit (V2). Le pic *DPA* est 25% plus important que le reste des courbes. Lorsque le nombre de courbes est réduit à $4 \cdot 10^5$ (figure 4.18-b), l'amplitude du pic *DPA* descend à 16%. Avec $2 \cdot 10^5$ courbes il devient impossible de le distinguer.

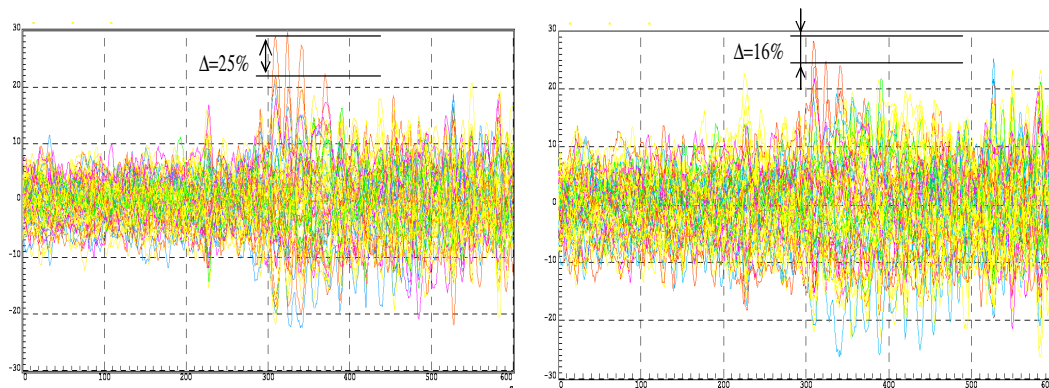


Figure 4.18 : Courbes *DPA* réalisées en attaquant le bit 1 en sortie du *Xor32* du cryptoprocresseur asynchrone (V2).

Cette approche prend en considération la qualité (précision) du type de matériel et du flot d'acquisition mis en place. Comme elle n'est pas absolue, mais tributaire de la qualité des mesures, il est nécessaire de toujours effectuer les mesures dans les mêmes conditions. Cependant, elle permet de comparer le niveau de résistance entre deux circuits et permet d'estimer l'effort et le temps nécessaire pour implémenter une attaque par analyse différentielle du courant.

En fonction de ce critère, nous avons analysé et évalué par rapport au synchrone la résistance du circuit asynchrone. Les attaques réussies sur le circuit synchrone nécessitaient en moyenne 10^4 courbes. Nous avons néanmoins montré des attaques réussies avec $64 \cdot 100$ courbes au chapitre II (cf. § 2.7.2). Les mêmes attaques sur la version asynchrone nécessitent en moyenne vingt fois plus de courbes. La figure 4.19 illustre le nombre de courbes utilisées pour réaliser les attaques sur le circuit asynchrone (V2).

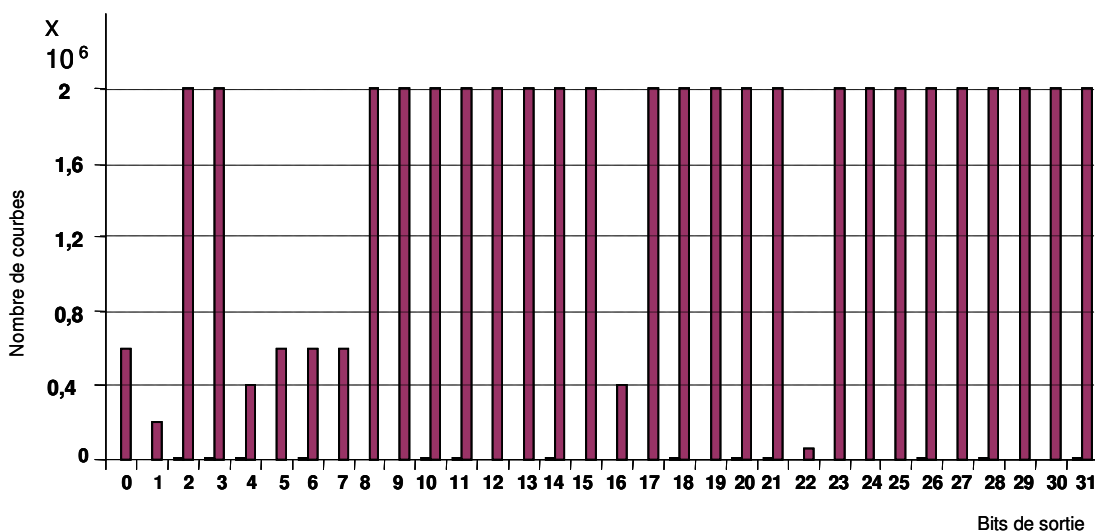


Figure 4.19 : Nombre de courbes utilisées pour réaliser des attaques *DPA* sur tous les bits de sortie du bloc *XOR32* du cryptoprocresseur asynchrone (V2) lors de l'exécution de la première itération.

0.4 Million de courbes sont en moyenne nécessaires pour réussir une attaque tandis que de nombreux bits restent insensibles aux attaques avec l'utilisation de plus de 2 millions de courbes.

Les résultats obtenus montrent le potentiel de la logique asynchrone *QDI* à augmenter la résistance des circuits faces à des attaques en puissances [Plan02, Four03]. Toutefois, ces analyses nous ont permis de mettre en évidence d'autres types de fuites d'informations qui permettent de réussir des attaques sur certains bits. En effet, comme indiqué sur le tableau 4.1, l'utilisation de la logique *QDI* permet de supprimer les paramètres les plus importants expliquant l'origine des fuites d'information dans les

circuits, à savoir : le poids de Hamming et la nature des commutations logiques (charge et décharge). Cela a permis de complexifier les attaques en terme de nombre de courbes nécessaires à réussir une attaque. Cependant les paramètres restants (qui ne sont pas supprimés dans le tableau), avec notamment les valeurs des capacités en sortie des portes logiques et le type de rampe (sachant que la rampe dépend de cette capacité), peuvent à priori expliquer l'origine des dissymétries résiduelles observées qui permettent de réussir des attaques sur certains bits.

Malgré l'existence de ces fuites d'information résiduelle, ces évaluations démontrent le potentiel intrinsèque de la logique asynchrone **QDI** à mieux résister aux attaques en puissance comparé à la logique synchrone [Boue04a].

4.5 Conclusion

Nous avons, dans ce chapitre, introduit la logique asynchrone quasi insensible aux délais et montré que cette logique, associé à un codage de données de type **1 parmi n** et à un protocole 4 phases, permet d'implémenter des circuits dont les chemins de données sont équilibrés. C'est-à-dire, des circuits dont le nombre de transitions logiques est constant quelques soient les données manipulées. Nous avons illustré que les propriétés de tels circuits sont intéressantes pour lutter contre les attaques par analyse de courant, car elles permettent de contrôler l'activité électrique du circuit et d'autre part de lisser et réduire les formes d'ondes du courant. Afin d'évaluer ces propriétés, nous avons réalisé plusieurs prototype dont un de référence synchrone.

Les résultats, des attaques obtenus sur ces prototypes, ont confirmé les prédispositions des circuits asynchrones **QDI** comparé à leur équivalent synchrone. Cependant ces analyses ont mis en lumière des faiblesses dans les circuits asynchrones qui peuvent être utilisées pour réussir des attaques. Nous allons dans le prochain chapitre analyser en détail l'origine de ces fuites d'information résiduelle de manière à proposer de nouvelles contre-mesures.

CHAPITRE V

ANALYSE FORMELLE DES CIRCUITS QUASI INSENSIBLES AUX DELAIS

5.1 Introduction

Nous avons montré dans le chapitre précédent la capacité de la logique asynchrone quasi insensible aux délais à accroître la résistance des circuits vis-à-vis des attaques en puissance. Les différents prototypes réalisés ont permis de démontrer le niveau de résistance élevé obtenu par rapport à une réalisation synchrone en termes de complexité et de mise en œuvre des attaques. Les fuites d'information qui peuvent subsister, même si elles demandent de déployer des efforts très importants pour être exploitées, permettent néanmoins de réussir des attaques en courant.

L'objectif de ce chapitre est d'analyser formellement la logique asynchrone quasi insensible aux délais afin de pouvoir, dans un premier temps, identifier l'origine des fuites d'information résiduelles observées et de proposer par la suite, de nouvelles méthodes de conception en vue de supprimer ces fuites ou de les rendre complètement inexploitable par toutes attaques en puissance.

Pour ce faire, nous avons défini un modèle formel de représentation des circuits asynchrones *QDI* qui permet d'analyser et de vérifier au niveau logique et au niveau électrique, les propriétés de la logique *QDI* bénéfiques à la protection des circuits. Ce modèle permet au niveau logique de formellement vérifier les symétries logiques des chemins de données dans chaque bloc. Au niveau électrique, il a permis de modéliser le profil de courant des circuits *QDI*. L'application de la *DPA* sur ce modèle de courant a permis de formellement identifier les fuites d'information dans ce type de circuit.

5.2 Représentation formelle des circuits asynchrones *QDI*

Le modèle de représentation formelle des circuits asynchrones *QDI* que nous avons adopté est basé sur la théorie des graphes [Hara68, Cogi03]. Les circuits *QDI* sont représentés sous forme de graphe orienté ou digraphe. Ce format permet de manipuler les informations nécessaires à l'analyse de la signature électrique des modules des circuits *QDI* tant au niveau logique qu'au niveau électrique.

5.2.1 Graphe orienté d'un circuit asynchrone *QDI*

Un graphe orienté ou digraphe est un graphe qui possède des arcs (ou arrêtes) orientés d'une extrémité initiale a (ou origine) à une extrémité finale b . Si $G = (V, E)$ est un graphe orienté, alors V et E sont respectivement des ensembles non vides et finis des sommets et des arcs orientés de G .

Afin de pouvoir représenter les circuits asynchrones *QDI* sous forme de graphes orientés, nous avons défini les deux propriétés suivantes :

- *Toutes les portes du circuit sont des éléments de l'ensemble V des sommets du digraphe G .*
- *Toutes les interconnexions du circuit sont des éléments de l'ensemble E des arcs orientés du digraphe G .*

L'orientation des arcs, entre les portes dans un module, est définie par le sens de propagation de l'information: de la sortie d'une porte v_0 vers une entrée v_1 . v_0 est alors le sommet ascendant de v_1 . La représentation graphique est faite en partant d'une netlist de portes *VHDL* ou *Verilog* fournie par l'outil *TAST* après synthèse du circuit. La figure 5.1 représente la description sous forme de netlist *VHDL* d'une porte *And* en double rail. Cette description correspond au schéma de la porte *And* présentée sur la figure 4.11 (cf.§ 4.3.2.3) suivie d'un Half-buffer.

```

ENTITY exemple IS
PORT (
    Resetb: in std_ulogic;
    A: in std_ulogic_vector (1 downto 0);
    A_ack: out std_ulogic;
    B: in std_ulogic_vector (1 downto 0);
    B_ack: out std_ulogic;
    S: out std_ulogic_vector (1 downto 0);
    S_ack: in std_ulogic);
END exemple;

ARCHITECTURE exemple_arch OF exemple IS
    SIGNAL E0, E1, E2, E3, E4: std_ulogic;
    SIGNAL E5, E6, E7 : std_ulogic;
BEGIN
    Inst_Name_00: MULLER2 port map (E0, A(1), B(1));
    Inst_Name_01: MULLER2 port map (E1, A(0), B(0));
    Inst_Name_02: MULLER2 port map (E2, A(0), B(1));
    Inst_Name_03: MULLER2 port map (E3, A(1), B(0));
    Inst_Name_04: TAST_NOR3 port map (E4, E1, E2, E3);
    Inst_Name_05: MULLER2_R port map (Resetb, E5, E0, S_ack);
    Inst_Name_06: MULLER2_R port map (Resetb, E6, E4, S_ack);
    Inst_Name_07: NOR2 port map (E7, E5, E6);

    S(1) <= E5;
    S(0) <= E6;
    A_ack <= E7;
    B_ack <= E7;

END exemple;
    
```

Figure 5.1 : Netlist de portes VHDL d’une description de la porte *And* en double rail

De cette netlist de portes, on obtient le graphe orienté $G_{And} = (V, E)$ décrit sur la figure 5.2. Chacun des sommets et des arcs orientés du graphe est annoté en fonction des portes et des interconnexions auxquelles elles correspondent dans la netlist. Les entrées et les sorties du bloc, sont représentées par les arcs en pointillés.

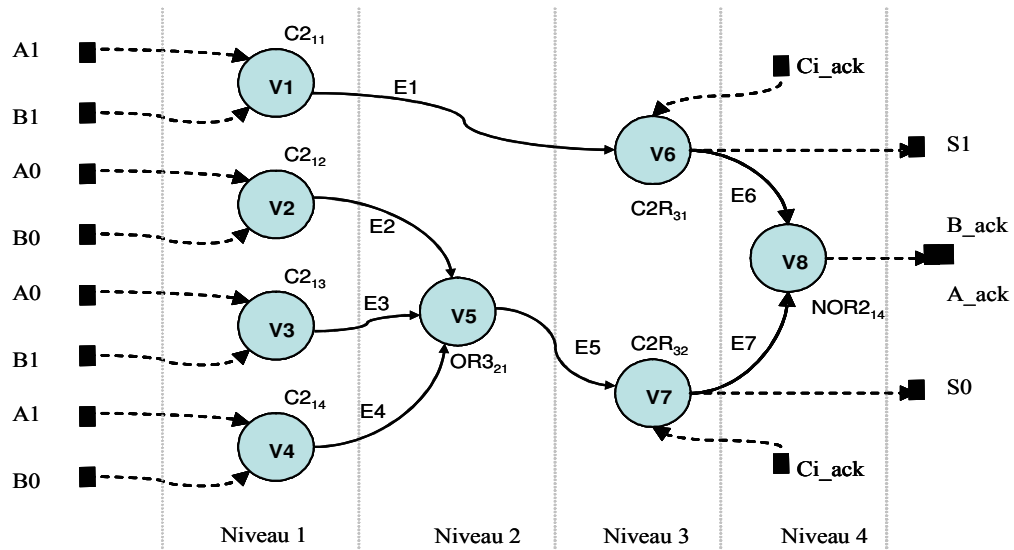


Figure 5.2: Graphe orienté annoté G_{And} d’une porte *And* en double rail

De cette représentation, est déduit une représentation formelle du graphe pour faciliter les analyses et la mémorisation. Suivant le nombre de sommets et d’arcs du graphe, on choisit soit une matrice d’adjacence, soit une liste d’adjacence. Pour des raisons de simplicité et d’illustration nous allons, pour la

suite, utiliser la matrice d'adjacence dite également matrice creuse car elle contient beaucoup d'éléments inutiles. Toutefois, la liste d'adjacence est utilisée pour l'implémentation car elle prend en compte uniquement les éléments nécessaires à l'analyse du digraphe, ce qui réduit ainsi l'espace mémoire nécessaire à la mémorisation du digraphe.

5.2.1.1 Matrice d'adjacence

La matrice d'adjacence du digraphe G est une matrice carré M de dimension $n \times n$ (n colonnes et n lignes) dans laquelle n représente le nombre total de sommets du graphe. S'il existe un arc orienté partant d'un sommet a (origine) à un sommet b (extrémité finale), alors l'élément du couple (a,b) , représentant le croisement entre la ligne contenant le sommet d'origine a et la colonne contenant le sommet d'extrémité finale b , est mise à 1 dans la matrice, si non il prend la valeur 0. Une telle matrice est également appelée matrice booléenne du digraphe G . Afin de représenter les sommets correspondants aux portes d'entrée d'un bloc (toutes les portes du niveau logique 1 qui reçoivent les signaux d'entrées) dans la matrice, nous avons défini la propriété suivante :

- Pour tout sommet a appartenant à l'ensemble V de G , si $(a,a)=1$, alors le sommet a est considéré comme un sommet d'entrée.

$$M_{BG} = \begin{pmatrix}
 \begin{array}{c|cccccccc}
 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \\
 \hline
 V_1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 V_2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
 V_3 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 V_4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 V_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 V_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 V_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 V_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array} \\
 \end{pmatrix}
 \qquad
 M_{AG} = \begin{pmatrix}
 \begin{array}{c|cccccccc}
 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \\
 \hline
 V_1 & C2_{11} & 0 & 0 & 0 & 0 & C2R_{31} & 0 & 0 \\
 V_2 & 0 & C2_{12} & 0 & 0 & OR3_{21} & 0 & 0 & 0 \\
 V_3 & 0 & 0 & C2_{13} & 0 & OR3_{21} & 0 & 0 & 0 \\
 V_4 & 0 & 0 & 0 & C2_{14} & OR3_{21} & 0 & 0 & 0 \\
 V_5 & 0 & 0 & 0 & 0 & 0 & 0 & C2R_{32} & 0 \\
 V_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & NOR2_{41} \\
 V_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & NOR2_{41} \\
 V_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array} \\
 \end{pmatrix}$$

a - Matrice Booléenne
b - Matrice annoté des noms des portes

Figure 5.3: Matrices du digraphe orienté G_{And}

De la matrice booléenne M_{BG} , on déduit une matrice M_{AG} du digraphe G annoté dans laquelle, les différentes valeurs binaires des couples (v_i, v_j) sont remplacées par les paramètres utilisés pour annoter le digraphe G en fonction des analyses à réaliser. En effet, les analyses au niveau logique et électrique à effectuer nécessitent de manipuler dans les digraphes les paramètres suivants :

- au **niveau porte logique** : la nature de la porte (fonction), l'instance de la porte, le délai de la porte et sa charge en sortie.
- au **niveau interconnexions** : le nom du net, son délai, la valeur de sa capacité et la valeur de sa résistance.
- au **niveau bloc logique** : la connectivité des portes, la profondeur logique du bloc (nombre de niveaux logiques), l'identification des sorties et entrées du bloc, les différents chemins de données entre les sorties et les entrées.

La figure 5.3-b représente l'exemple de la matrice annoté M_{AG} du digraphe annoté G dans laquelle chaque couple (v_i, v_j) à 1 est remplacé par le nom de la porte du sommet v_j . On peut ainsi, en fonction des analyses à réaliser, collecter les informations dans les différentes phases de conception et re-annoter la matrice booléenne.

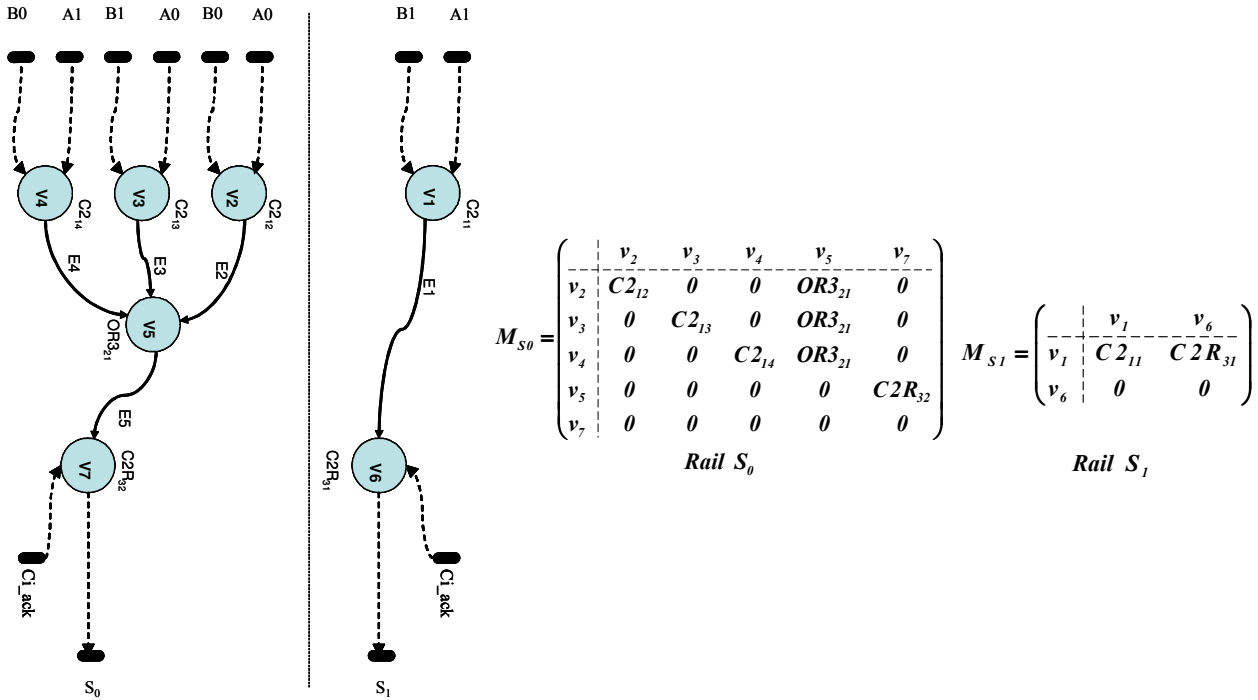
La somme des éléments sur chaque ligne ' i ' permet de déterminer le nombre des arcs ayant leur extrémité initiale au sommet v_i ($d^+(v_i)$), et la somme des éléments sur chaque colonne ' j ' donne le nombre des arcs ayant leur extrémité terminale au sommet v_j ($d^-(v_j)$), excepté pour les sommets représentant des portes de sorties (les portes qui génèrent les signaux de sortie du bloc). Ces sommets sont identifiés en parcourant la netlist et sont étiquetés différemment dans la matrice.

Les analyses au niveau logique et électrique sont effectuées sur les chemins de données de chacune des sorties d'un bloc. L'idée est de créer les sous digraphes correspondant au chemin de données de chaque de sortie d'un bloc afin d'analyser leur signature électrique.

5.3 Analyse au niveau logique : symétrie des chemins de données

Les analyses au niveau logique sont focalisées sur la vérification formelle des symétries des différents chemins de données d'un bloc. L'analyse des chemins de données est réalisée en extrayant du graphe orienté tous les chemins de données dépendants des sommets des sorties. Le parcours du graphe orienté commence par l'identification des sommets des sorties. De chacun de ces sommets, on forme un sous graphe orienté constitué de tous ses ascendants. Chaque sommet de sortie est alors considéré comme une anti-racine de l'anti-arbre du sous graphe orienté formé. Dans cet anti-arbre tous les arcs orientés sont dirigés vers le sommet de sortie. En considérant l'exemple précédant, on déduit les deux sous graphes

orientés $G_{S_0} = (V_{S_0}, E_{S_0})$ et $G_{S_1} = (V_{S_1}, E_{S_1})$ (figure 5.4) correspondant aux deux sorties S_0 et S_1 de la netlist de portes. Dans ce cas présent, nous n'avons pas tenu compte de la sortie S_ack dont le sous digraphe est équivalent au digraphe initial.



a- Digraphes des chemins de données b- Matrices des digraphes des chemins de données

Figure 5.4: Matrices et digraphes orientés de chacun des sommets des sorties de G_{And}

Afin de prendre en compte la symétrie logique entre les chemins de données d'un digit codé en rail (1 parmi n), on introduit la notion de chemin d'exécution.

5.3.1 Notion de chemin d'exécution

L'analyse du chemin d'exécution dans les circuits asynchrones **QDI** est faite en étudiant les fonctions élémentaires des circuits **QDI** à savoir : la transition logique, la divergence en 'et', la convergence en 'et' et la convergence en 'ou'.

- **La transition logique** : Elle correspond au chemin comprenant au moins un sommet transmetteur ($d^+(v_i) = d(v_i) = 1$). Dans ce type de structure, le calcul de l'ascendant est trivial (figure 5.5-a).

- **La divergence en ‘et’** : La divergence en ‘et’ correspondant à un sommet v_i dont $d^+(v_i) > 1$. Chacun des descendants du sommet v_i possède le même ascendant. Une porte est dite divergent en ‘et’ si et seulement si sa sortie est une fourche (figure 5.5-b).
- **La convergence en ‘et’** : Elle correspond à un sommet qui possède plusieurs ascendant ($d^-(v_i) > 1$). Elle est associée aux portes réalisant des fonctions de convergence en ‘et’. Une porte est dite convergente en ‘et’ si et seulement si elle est activée lorsque toutes ces entrées sont valides (exemple de la porte de Muller) (figure 5.5-c).
- **La convergence en ‘ou’** : Elle est associée aux portes réalisant des fonctions de convergence en ‘ou’ et correspond à un sommet à plusieurs ascendant ($d^-(v_i) > 1$). Une porte est dite convergente en ‘ou’ si et seulement si toutes ses entrées sont exclusives (exemple d’une porte or) (figure 5.5-d).

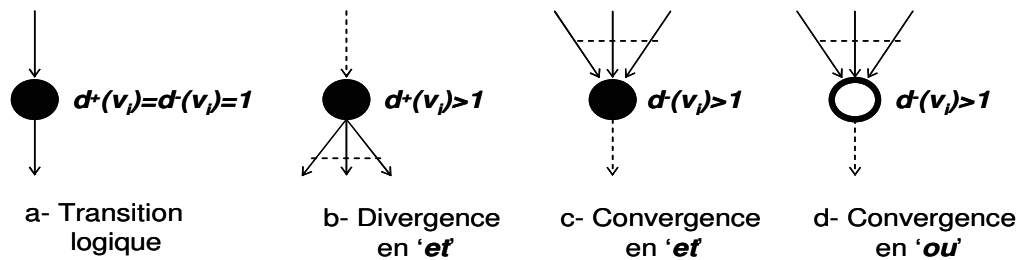


Figure 5.5: Présentation des notions de divergence et convergence en ‘et’ et en ‘ou’

Cette notion d’exclusivité des entrées d’une porte (porte convergente en ‘ou’) est utilisée afin de déterminer les chemins d’exécution d’un bloc. Tout chemin de données entre les entrées et les sorties d’un bloc pouvant évaluer l’une de ses sorties est un chemin d’exécution. On construit un chemin d’exécution à l’aide des propriétés suivantes :

- *Chaque entrée exclusive d’une porte convergente en ‘ou’ ne peut être représentée que dans un seul chemin d’exécution.*
- *Toutes les entrées d’une porte convergente en ‘et’ sont représentées dans un chemin d’exécution.*
- *Toute porte divergence en ‘et’ est représentée dans chacun des chemins possédant chacune de ses branches.*

En considérant le sous graphe orienté G_{S_0} , on déduit à travers l'utilisation de la porte $OR3_{31}$ (convergence en 'ou'), les trois sous graphes orientés $G_{S_{01}}=(V_{S_{01}},E_{S_{01}})$, $G_{S_{02}}=(V_{S_{02}},E_{S_{02}})$ et $G_{S_{03}}=(V_{S_{03}},E_{S_{03}})$ tels que :

$$\begin{aligned}
 G_{S_{01}} &= (V_{S_{01}}, E_{S_{01}}) \text{ avec } V_{S_{01}} = \{v_2, v_5, v_7\} \text{ et } E_{S_{01}} = \{E_2, E_5\} \\
 G_{S_{02}} &= (V_{S_{02}}, E_{S_{02}}) \text{ avec } V_{S_{02}} = \{v_3, v_5, v_7\} \text{ et } E_{S_{02}} = \{E_3, E_5\} \\
 G_{S_{03}} &= (V_{S_{03}}, E_{S_{03}}) \text{ avec } V_{S_{03}} = \{v_4, v_5, v_7\} \text{ et } E_{S_{03}} = \{E_4, E_5\}
 \end{aligned}$$

Chacun de ces digraphes, représente un chemin d'exécution du rail S_0 (figure 5.6).

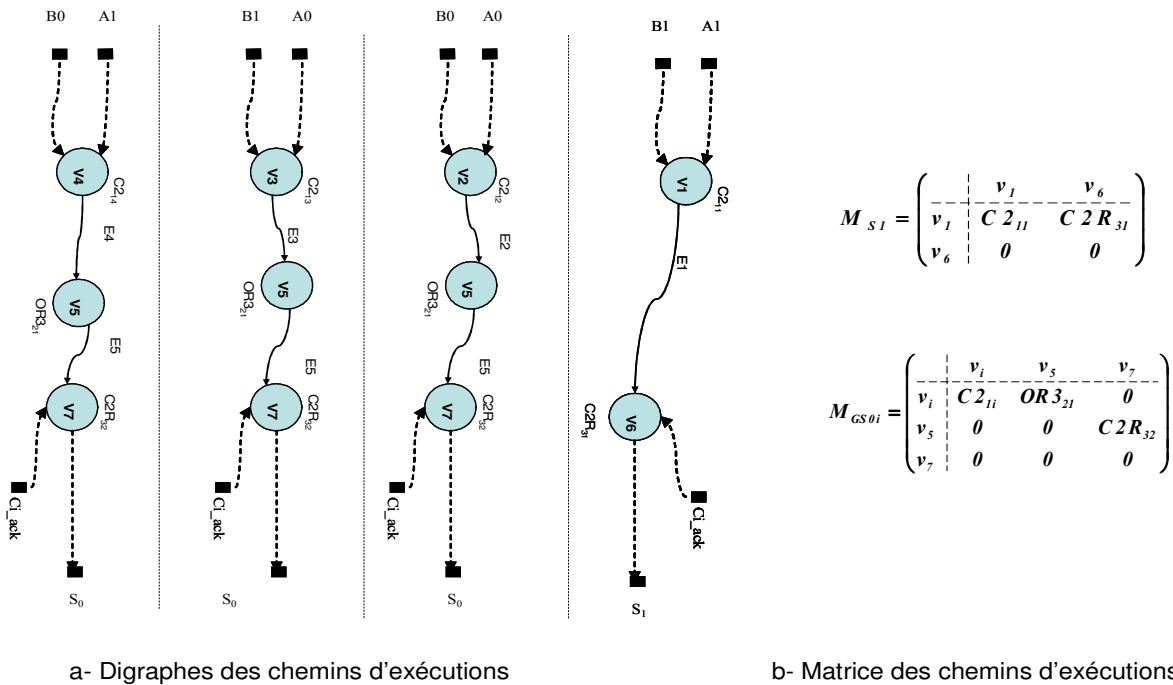


Figure 5.6: Matrices et digraphes de chacun des chemins d'exécutions du digraphe G_{And}

Ainsi, l'analyse de la symétrie des chemins de données dans un bloc au niveau logique équivaut à effectuer les analyses des différents chemins d'exécutions des bits de sorties de ce bloc. Cette analyse est réalisée formellement en utilisant la notion d'isomorphisme des graphes.

5.3.2 Les graphes isomorphes

Deux graphes orientés G_1 et G_2 sont isomorphes si et seulement si pour tout arc orienté (a,b) de G_1 il existe une fonction bijective f telle que l'arc $f((a,b))$ soit un arc de G_2 . f est alors appelé fonction d'isomorphie.

$$\begin{aligned} f : G_1 &\longrightarrow G_2 \text{ telle que} \\ \forall (a,b) \in G_1 &\Leftrightarrow f((a,b)) \in G_2 \end{aligned} \quad (5.1)$$

En terme de matrice, soient M_1 et M_2 respectivement les matrices booléennes des digraphes G_1 et G_2 . Les digraphes G_1 et G_2 sont isomorphes si et seulement si pour tout classement des sommets de G_1 , il existe un classement des sommets de G_2 tel que les matrices booléennes de G_1 et G_2 soient égales.

$$\text{Si } M_1 = M_2 \text{ alors } G_1 \text{ et } G_2 \text{ sont isomorphes} \quad (5.2)$$

Par conséquent, en fonction des graphes et matrices des différents chemins d'exécution construits à partir du graphe d'un bloc, on analyse la symétrie logique du bloc en calculant la fonction d'isomorphisme entre ses différents chemins d'exécution.

- *Les chemins de données d'un bloc sont symétriques au niveau logique si et seulement si les graphes orientés de ses chemins d'exécutions sont isomorphes.*

De ce fait, un bloc est équilibré si ses chemins de données sont symétriques au niveau logique, sinon il est dit non équilibré.

En reprenant l'exemple précédent, comme les matrices M_{S01} , M_{S02} et M_{S03} sont identiques alors les digraphes G_{S01} , G_{S02} et G_{S03} sont isomorphes. Cependant, le digraphe G_{And} est équilibré si et seulement si le digraphe G_{S1} du chemin d'exécution du rail S_1 est isomorphe à l'un des digraphes des chemins d'exécutions du rail S_0 (G_{S01} , G_{S02} ou G_{S03}). Afin d'équilibrer le bloc, le digraphe G_{S1} est modifiés telle que la matrice M_{S1} soit égale aux matrices M_{S0i} (avec $i=1,2,3$).

$$M_{S1} = M_{S01} = M_{S02} = M_{S03} = \begin{pmatrix} & v_i & v_j & v_k \\ v_i & C2_{1i} & OR3_{2j} & 0 \\ v_j & 0 & 0 & C2R_{3k} \\ v_k & 0 & 0 & 0 \end{pmatrix}$$

Le nouveau digraphe de G_{And} , obtenu après modification, est représenté sur la figure 5.7 dans lequel tous les chemins de données sont équilibrés au niveau logique.

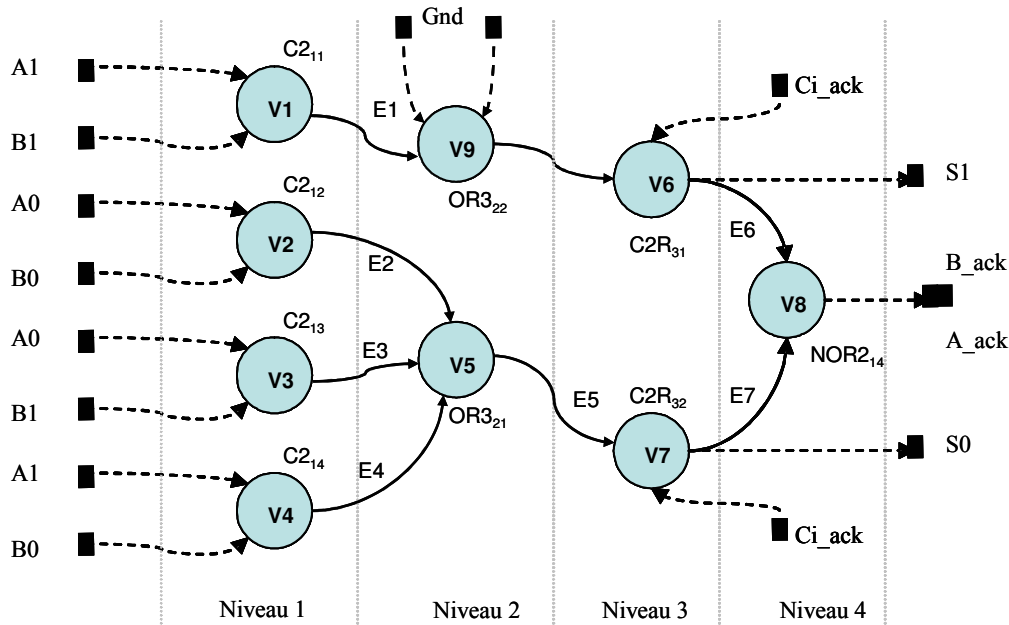


Figure 5.7: Digraphe de la porte *And* en double rail équilibré.

La représentation graphique adopté dans cette approche est bien adaptée à l'analyse formelle de la symétrie dans les circuits asynchrones *QDI*. Elle offre la possibilité de formellement analyser la symétrie des chemins de données du circuit et de corriger l'équilibrage si nécessaire au niveau logique, ceci afin de préserver tous les bénéfices dus à l'utilisation d'un codage *p parmi n* et d'un protocole 4 phases (nombre de transition logique indépendant des données manipulées).

Les analyses au niveau électrique sont effectuées en rétro annotant les digraphes à l'aide des informations collectées lors des phases de placement et de routage.

5.4 Analyses au niveau électrique : Identification des fuites d'informations

Le travail a d'abord porté sur la modélisation de la signature électrique d'un circuit asynchrone *QDI*. Nous avons, par la suite, appliqué une attaque par analyse différentielle du courant sur le modèle de façon à identifier ces fuites d'informations résiduelles.

5.4.1 Modèle électrique d'un circuit asynchrone *QDI*

On s'intéresse ici exclusivement aux circuits asynchrones *QDI* sécurisés au sens décrit dans le paragraphe précédent (cf. § 5.3.2). Ainsi, on suppose que tous les blocs sont équilibrés au niveau logique c'est à dire que tous les calculs impliquent un nombre constant de transitions logiques, indépendamment des données d'entrée traitées.

On part de l'expression de la consommation de puissance d'un bloc logique en technologie *CMOS* statique qui inclut :

- La consommation statique due aux courants de fuite, P_f .
- La consommation de court circuit, P_s .
- La consommation dynamique utile, P_d .

Dans le cadre des applications visées, la consommation dynamique étant prépondérante sur la vitesse, il est raisonnable de négliger la consommation statique et la consommation de court circuit. Un dimensionnement adéquat des transistors des portes permet de réduire leur contribution à moins de 15%. La consommation dynamique utile, qui est dissipée lors de la charge et décharge des capacités présentes à la sortie des portes logiques, a pour expression :

$$P_d = \eta f C V_{dd}^2 \quad (5.3)$$

Où η est le rapport d'activité, f la fréquence de fonctionnement, V_{dd} la tension d'alimentation et C la charge totale à commuter définie par : $C = C_l + C_{par} + C_{cc}$ avec C_l la capacité de charge (portes et routage), C_{par} la capacité parasite et C_{cc} la capacité équivalente de court circuit.

Nous étendons ce modèle aux circuits asynchrones considérés dans cette étude :

$$P_{da} = \eta f_a C V_{dd}^2 \quad (5.4)$$

Avec f_a la fréquence de fonctionnement du bloc, c'est à dire la fréquence des signaux de requête et d'acquiescement. Le nombre déterminé et fixe de transitions nécessaire à un calcul nous permet d'estimer la

consommation dynamique. Soit N_t le nombre de transitions logiques pour un bloc donné, la dissipation dynamique du bloc est alors :

$$P_{db} = \eta f_a \sum_{i=1}^{N_t} C_i V_{dd}^2 \quad (5.5)$$

L'objectif de l'équilibrage des chemins dans un bloc asynchrone est d'obtenir une consommation en courant identique quelque soit les données traitées. Or, le modèle de consommation de l'équation (5.5) montre bien que la charge des portes logiques n'est pas nécessairement égale. Elle dépend en particulier des capacités parasites et de routage. Estimons le courant dynamique consommé par un bloc. Le courant dynamique consommé dans une porte est donné par :

$$I(t) = C \frac{dV}{dt} \quad (5.6)$$

Soit N_c le nombre de portes en série dans les chemins de données. Puisque les circuits sont équilibrés, tous les chemins présentent N_c portes en série, ou encore N_c niveaux logiques. On note alors N_{ij} le nombre de portes j qui commutent à chaque niveau logique i . L'expression du courant dynamique d'un bloc peut alors s'exprimer comme :

$$P_{dc}(t) = \sum_{i=1}^{N_c} \left[\sum_{j=1}^{N_{ij}} I_{ij}(t_i) \right] + P_{dn}(t) \quad (5.7)$$

$I_{ij}(t_i)$ représente le courant dynamique de la j -ième porte au niveau logique i , et P_{dn} est une fonction de bruit. Les valeurs N_t , N_c et N_{ij} sont déterminées par une analyse de la structure logique du bloc. En considérant le digraphe de la figure 5.7 représentant le digraphe d'une implémentation équilibrée de la fonction **And** en double rail. On déduit par analyse du digraphe les valeurs suivantes N_t , N_c et N_{ij} :

$$N_t = N_c = 4 ; N_{1j} = N_{2j} = N_{3j} = N_{4j} = 1$$

Ainsi, l'expression du courant dynamique de ce bloc est donné par :

$$P_{dc_{AND}}(t) = (I_{1j}(t_1) + I_{2j}(t_2) + I_{3j}(t_3) + I_{4j}(t_4)) + P_{dn}(t) \quad (5.8)$$

L'équation (5.8) représente une approximation du profil temporel du courant du bloc de calcul d'une fonction *And*. Cette approche peut être étendue à tout bloc asynchrone *QDI* et nous permet d'analyser la sensibilité du bloc à l'attaque par analyse différentielle du courant.

5.4.2 Application de l'attaque DPA sur le modèle formel

Contrairement aux circuits synchrones pour lesquels la *DPA* révèle les dissymétries relatives à une équipotentielle correspondante au bit attaqué, l'application de la *DPA* aux circuits asynchrones révèle une dissymétrie entre plusieurs équipotentielles correspondantes aux rails utilisés pour encoder les valeurs logiques (0 et 1 dans le cas binaire) du bit attaqué.

En effet, appliquer la *DPA* au circuit de la figure 5.7 nécessite de comparer le comportement électrique des chemins qui effectuent le calcul des rails S_0 et S_I . Ainsi les courants moyens consommés par le circuit en fonction de la valeur du bit calculé sont donnés par :

$$A_{And0}(t) = (I_{11}(t_1) + I_{22}(t_2) + I_{31}(t_3) + I_{41}(t_4) + I_n(t))$$

$$A_{And1}(t) = \frac{1}{2}(I_{12}(t_1) + I_{13}(t_1) + I_{14}(t_1) + I_{21}(t_2) + I_{32}(t_3) + I_{41}(t_4) + I_n(t))$$
(5.9)

Avec $I_n(t)$ le bruit. La signature électrique *DPA* est donnée par :

$$S_{DPA}(t) = \left(C_{11} \frac{dVout_{11}}{dt_{11}} + C_{22} \frac{dVout_{22}}{dt_{22}} + C_{31} \frac{dVout_{31}}{dt_{31}} + C_{41} \frac{dVout_{41}}{dt_{41}} \right) -$$

$$\frac{1}{3} \left(C_{12} \frac{dVout_{12}}{dt_{12}} + C_{13} \frac{dVout_{13}}{dt_{13}} + C_{14} \frac{dVout_{14}}{dt_{14}} + C_{21} \frac{dVout_{21}}{dt_{21}} + C_{32} \frac{dVout_{32}}{dt_{32}} + C_{41} \frac{dVout_{41}}{dt_{41}} \right)$$
(5.10)

Puisque $\frac{dVout_{ij}}{dt_{ij}} \cong \frac{\Delta V}{\Delta t_{ij}}$ l'expression (5.10) devient :

$$S_{DPA}(t) = \Delta V \left(\frac{C_{11}}{\Delta t_{11}} - \frac{1}{3} \left(\frac{C_{12}}{\Delta t_{12}} + \frac{C_{13}}{\Delta t_{13}} + \frac{C_{14}}{\Delta t_{14}} \right) \right) + \Delta V \left(\frac{C_{22}}{\Delta t_{22}} - \frac{1}{3} \frac{C_{21}}{\Delta t_{21}} \right) + \Delta V \left(\frac{C_{31}}{\Delta t_{31}} - \frac{1}{3} \frac{C_{32}}{\Delta t_{32}} \right)$$
(5.11)

Δt représente le temps nécessaire à la porte pour charger/décharger sa sortie. Ce temps dépend de la valeur de la charge C ($C=C_I+C_{par}+C_{cc}$).

L'équation (5.11) montre clairement que bien que les chemins soient équilibrés au niveau logique, les capacités de charge des portes ont un effet sur la signature *DPA*, et rendent donc le circuit sensible aux attaques en puissance.

5.4.3 Validation par simulation électrique

Pour valider et quantifier les effets des capacités sur la *DPA*, nous avons effectué des simulations électriques de circuits conçus en technologie *HCMOS9 (0,13 μm)* de *STMicroelectronics* à l'aide du simulateur *Eldo*. De telles simulations électriques offrent la possibilité d'analyser avec plus de détails et sans signaux de perturbations (bruit) le comportement électrique de chaque porte du bloc.

La figure 5.8 illustre les signatures électriques obtenues pour les circuits *And* équilibrés et non équilibrés dont les digraphes sont respectivement représentés sur les figures 5.7 et 5.2. La charge utilisée est une charge identique pour toutes les capacités C_l ($C_l=C_d=8$ fF) et les phases d'évaluation et de retour à zéro sont analysées. Comme on s'y attendait, le circuit non équilibré présente une signature électrique plus importante que celle du circuit équilibré (au niveau logique). Les quelques pics remarquables sur la signature du circuit équilibré sont dus aux capacités parasites C_{par} et de court circuit C_{cc} .

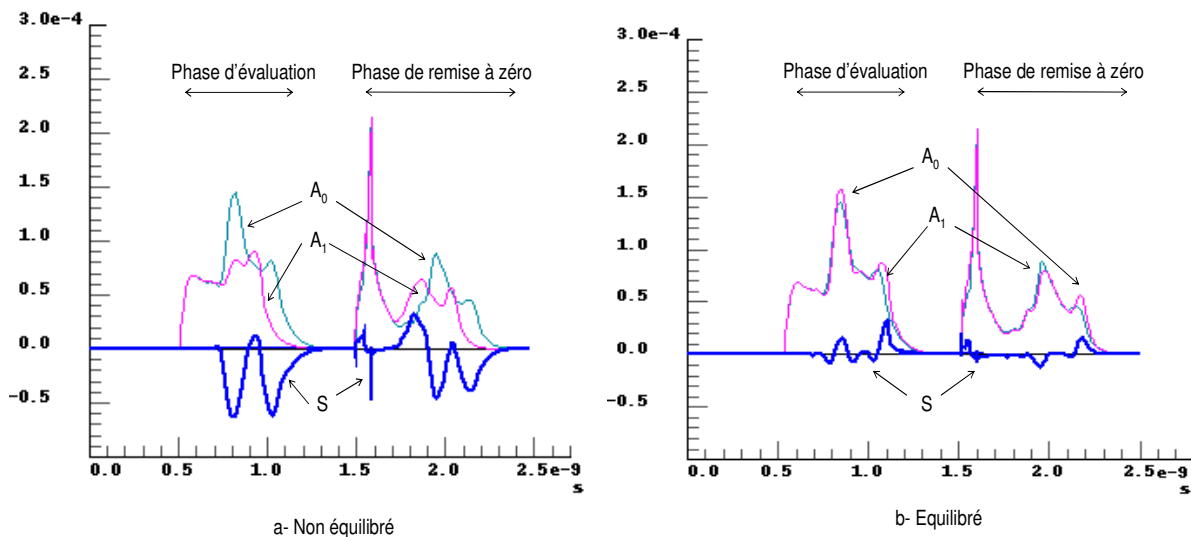


Figure 5.8: Signature électrique la porte *And* en double rail $C_{lj}=8$ fF.

Introduisons maintenant une dissymétrie dans les capacités de routage pour en évaluer l'effet sur la signature *DPA*. La figure 5.9 présente les résultats de simulations. Une capacité de routage de $C_d = 8$ femto Farad est associée à chaque nœud par défaut. La figure 5.9-a représente le profil de courant $S(t)$

lorsque la capacité C_{I31} est deux fois plus importante que C_d . Puisque C_{I31} est à la sortie du bloc, au troisième niveau logique, on voit apparaître un pic à la fin de chacune des phases. Ce phénomène correspond au dernier terme de l'équation (5.11). Lorsque C_{I22} est fixé à $2 * C_d$, deux pics apparaissent dans $S(t)$ (Figure 5.9-b). Cela s'explique par le fait, que cette capacité cause un retard qui décale le profil de courant de tout le bloc. Cela est confirmé par le résultat de simulation présenté sur la figure 5.9-c où C_{I11} est égale à $2 * C_d$. Enfin, la Figure 5.9-d présente une simulation pour laquelle la valeur de C_{I11} est quatre fois plus importante que C_d . Dans ce cas les profils de courant sont complètement décalés et la signature $S(t)$ présente de forts pics de *DPA*.

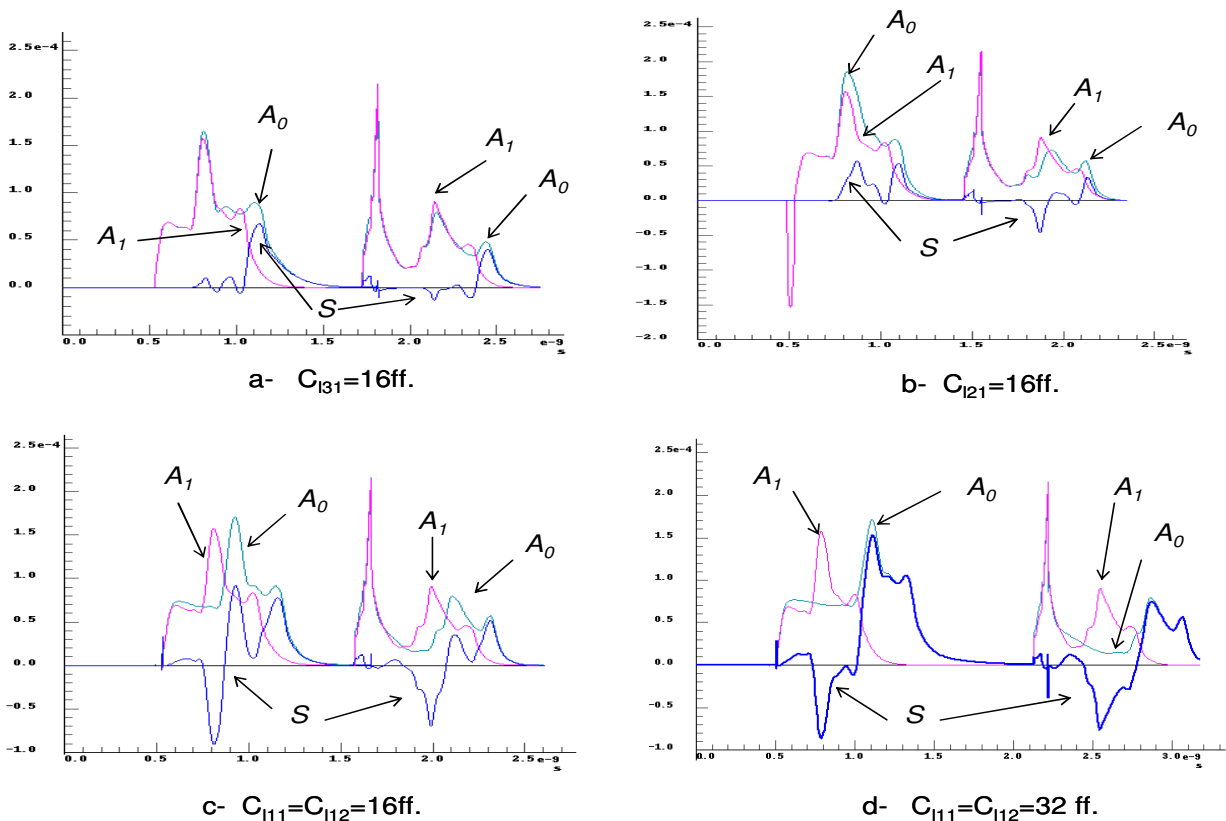



Figure 5.9: Signatures électriques de la porte *And* en double rail. Les capacités de routage varient de $C_i=8\text{fF}$ à 32fF .

Ces simulations confirment les conclusions du modèle théorique et montrent la forte dépendance entre la valeur de la capacité de sortie des portes et la signature électrique [Boue05c].

En reprenant le tableau 4.1 (cf. § 4.3.2), on remarque effectivement que seuls les paramètres dépendant des capacités sont présents à savoir les rampes, la capacité de sortie et le délai. Les délais des portes deviennent aussi importants. Comme nous l'avons illustré, l'analyse différentielle entre deux chemins met en évidence le délai de chaque porte et le délai total de chaque chemin (délai cumulé de chaque porte du chemin). Cela a une incidence sur l'instant d'occurrence du pic de consommation de chaque porte dans un chemin. En reconsidérant les influences de chaque paramètre, on redéfinit ce tableau par le tableau 5.1.

Tableau 5.1 Paramètres influençant l'analyse différentielle en courant dans un circuit *QDI*

| | Analyses au niveau logique | Analyses au niveau électrique | | | | | |
|----------------------------------|---|-------------------------------------|----------------|-----------------------|--------------------------|-----------------|----------------------|
| | Nombre de transitions logiques: poids de Hamming | Commutation: charge/ décharge | Rampe lente | Capacité en sortie | Structure de la porte | Rampe rapide | Délai de la porte |
| Amplitude des pics | ✓✓ | ✓✓ | ✓✓ | ✓✓ | ✓✓ | — | — |
| Décalage temporel des pics | ✓✓ | ✓✓ | ✓✓✓ | ✓✓✓✓ | ✓✓ | ✓✓ | ✓✓✓ |



5.5 Conclusion

Afin d'identifier les fuites d'informations observées dans les circuits *QDI* lors d'une attaque en puissance, nous avons dans ce chapitre analysé formellement la symétrie des circuits *QDI* sur deux niveaux d'abstractions, au niveau logique et au niveau électrique. Pour ce faire, nous avons adopté une représentation des circuits *QDI* sous forme de graphe orienté. De cette représentation, nous avons utilisé les propriétés des graphes pour analyser au niveau logique puis au niveau électrique l'équilibrage des circuits *QDI*.

Ces analyses au niveau logique nous ont permis de confirmer d'une part le potentiel des propriétés de la logique asynchrone *QDI* contre les attaques en puissance et d'autre part de vérifier formellement l'équilibrage logique dans les blocs *QDI*. Au niveau électrique, ces analyses nous ont permis de définir un

modèle de courant de ce type de circuit. L'application de la **DPA** sur ce modèle a permis de mettre en évidence l'origine des fuites d'informations observées sur ces circuits.

En effet, nous avons formellement montré que quelque soit le niveau d'équilibrage obtenu au niveau logique, les signatures électriques sont fortement altérées en fonction des capacités de sortie des portes et ceci notamment à cause des capacités parasites introduites pendant les différentes phase de placement, de routage et d'optimisation électrique du back end. Ces résultats démontrent la nécessité de définir des méthodes et/ou techniques pour limiter cette source de fuite afin d'améliorer la résistance des circuits asynchrones **QDI**.

CHAPITRE VI

PROTECTION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS : FLOT DE CONCEPTION SECURISE

6.1 Introduction

Les analyses formelles réalisées dans le chapitre V sur la signature électrique des circuits asynchrones *QDI* ont permis d'identifier et de localiser l'origine des fuites d'informations. En effet, la suppression de l'influence du poids de Hamming, et l'utilisation d'une logique équilibrée ont favorisé la mise en évidence des dissymétries introduites pendant les phases de 'back-end' qui permettent de réussir des attaques par analyses différentielles du courant de manière à obtenir les clés cryptographiques.

Dans l'objectif de proposer des méthodes pour améliorer la résistance des circuits asynchrones *QDI*, nous présentons, dans ce chapitre, une nouvelle méthodologie de conception sécurisée, basée sur un flot qui permet de contrôler à chaque phase de conception, la sensibilité du circuit à des attaques en puissance. Ce flot s'appuie sur le format de représentation formel adopté au chapitre V pour analyser les circuits *QDI*. Les résultats obtenus dans ce chapitre ont conduit à établir un critère de sensibilité au niveau électrique afin de pouvoir limiter les dissymétries dues aux capacités parasites. Ceci, en imposant des contraintes pendant les phases de placement et de routage. Le flot, ainsi spécifié, a été utilisé pour réaliser deux prototypes de circuits implémentant chacun l'algorithme du Rijndael (*AES*). Les caractéristiques et les résultats des attaques sur ces circuits sont aussi présentés.

6.2 Flot de conception sécurisé des circuits *QDI*

L'objectif de cette approche est de contrôler le flot de conception et notamment les étapes de back-end pour supprimer les dissymétries électriques entre les différents chemins de données. Pour ce faire, nous avons défini un flot de conception qui permet de prendre en compte à toutes les phases de conceptions la symétrie logique et la symétrie électrique du circuit. Ce flot, défini sur la figure 6.1, est basé sur le modèle formel adopté pour représenter les circuits *QDI* en exploitant les propriétés des graphes orientés.

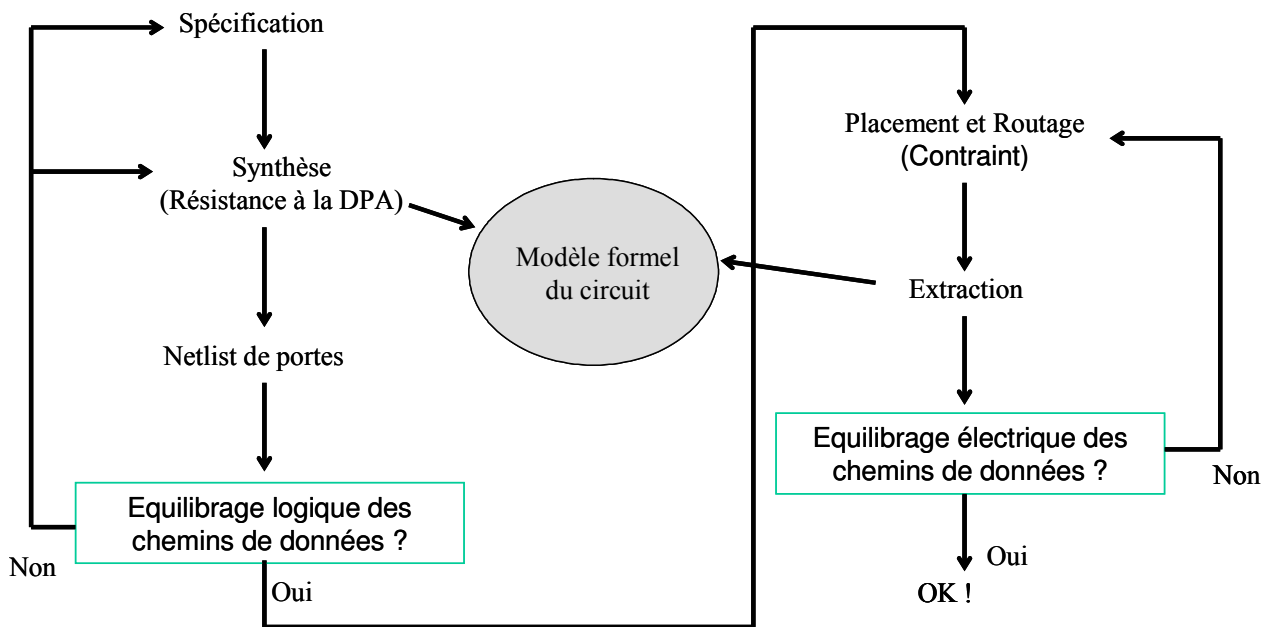


Figure 6.1: Synoptique du flot de conception

La spécification du circuit est réalisée sous la forme de programmes écrits en langage de description de matériel *CHP*. Les programmes sont simulés dans notre environnement de conception *TAST* et synthétisés ensuite sous la forme d'une netlist de portes logiques. La représentation formelle du circuit est construite à partir de la netlist, et permet de vérifier et de garantir la symétrie logique des blocs. Une fois cette étape de vérification effectuée, le circuit est placé et routé. La phase d'extraction permet de réannoter le modèle formel avec les informations des capacités. Il est alors possible de vérifier le niveau d'équilibrage électrique en calculant le paramètre dc_i du critère d'évaluation de la symétrisation électrique et d'itérer sur la phase de placement et routage pour converger vers une solution acceptable.

6.2.1 Critère d'évaluation de la dissymétrie électrique

Les analyses théoriques réalisées et les résultats des simulations ont montrés l'importance de la capacité de routage dans la sensibilité à la **DPA** (cf. § 5.4). Nous avons donc défini un critère pour évaluer après placement routage le niveau de dissymétrie introduit pour un bit donné. Si C_{ij} représente la capacité du rail j ($j \in \{0,1\}$ pour un canal double rail) d'un canal i , alors la dissymétrie du canal i est définie par l'expression :

$$dc_i = \frac{|C_{i0} - C_{i1}|}{\text{Min}(C_{i0}, C_{i1})} \quad (6.1)$$

Comme illustré précédemment, plus la valeur de dc_i est faible, plus le circuit est résistant à la **DPA**. Par conséquent, on peut d'une part, identifier dans une architecture les bits les plus sensibles sur lesquels il serait possible de réussir des attaques en puissance, et d'autre part, contraindre le placement routage en fonction de ce critère.

6.2.2 Placement routage contraint

Habituellement, lorsque le circuit n'est pas trop complexe, comme c'est le cas pour l'**AES**, le placement routage s'effectue "à plat", c'est à dire en travaillant sur le réseau de portes logiques comme un seul bloc. Cette approche permet d'obtenir une surface minimale pour le circuit. Malheureusement, nous avons constaté que cette approche ne permet pas de contrôler le placement et donc les longueurs des fils de routage.

Ainsi, nous avons adopté une stratégie de placement routage hiérarchique de façon à pouvoir contraindre le placement et ainsi agir sur le critère de dissymétrie dc_i . Cependant, comme ce critère n'est pas interne aux outils de placement routage commerciaux (**Soc encounter**, **Silicon Ensemble**), il est difficile, lorsqu'une dissymétrie est observée sur un nœud, de pouvoir la corriger en contraignant le placement routage. En effet, en imposant une contrainte sur ce nœud, on ne saurait garantir lors d'une nouvelle itération de routage qu'aucune nouvelle dissymétrie ne sera introduite. Ces types d'outils ont pour but d'optimiser uniquement la surface, la consommation et surtout les performances en vitesse en bufférisant les chemins de données lents électriquement (chemin possédant de fortes charges capacitives). Lorsqu'une contrainte est imposée sur un nœud, cette contrainte est prise en compte, par l'outil, en

modifiant le placement et le routage des cellules avoisinants le nœud, introduisant ainsi des nouvelles dissymétries.

La possibilité de faire converger les valeurs de dc_i des différents nœuds du circuit vers leurs valeurs minimales est un problème de type NP-complet que l'on saurait résoudre en modifiant les algorithmes d'optimisations de ces outils afin d'y intégrer le critère dc_i (des outils avec les codes sources). De tels outils existent uniquement dans le domaine universitaire comme c'est le cas des outils développés par le LIRMM pour le placement et routage des cellules virtuelles [Pell04].

A défaut de pouvoir contraindre le routage en fonction du critère dc_i , on utilise des cellules dédiées à la conception des circuits asynchrones afin de pouvoir non seulement optimiser la surface et les performances, mais également de contraindre le placement des cellules en utilisant des cellules complexes double rail.

6.2.3 Cellules asynchrones

Nous avons entrepris la conception d'un ensemble de cellules de bibliothèque dites asynchrones. Ces cellules n'existent dans aucune bibliothèque standard et ont été jusqu'à présent réalisées en utilisant exclusivement les fonctions disponibles [Rig02]. Il en résulte bien sûr une pénalité sur la surface, ainsi que sur les performances en vitesse et en consommation.

Les objectifs d'un tel développement sont principalement focalisés sur la réduction de la surface des circuits *QDI*. Toutefois, l'utilisation des portes complexes asynchrones permet de réduire le nombre de fils d'interconnexions ce qui réduit d'une part, les risques de dissymétries électriques et d'autre part, allège le temps d'itération nécessaire à l'évaluation du critère de symétrisation électrique dc_i . Par ailleurs, l'usage des cellules asynchrones double rail (une entrée et une sortie codée en *1 parmi 2*) facilite la symétrisation électrique lors du placement routage. Le fait d'utiliser une porte double rail permet de minimiser quelque soit le placement de la cellule dans le 'floorplan', la différence de charge entre les rails de la porte. La figure 6.2 représente un exemple d'usage de portes complexes et d'une porte en double rail dans une implémentation d'une fonction ou exclusif en double rail.

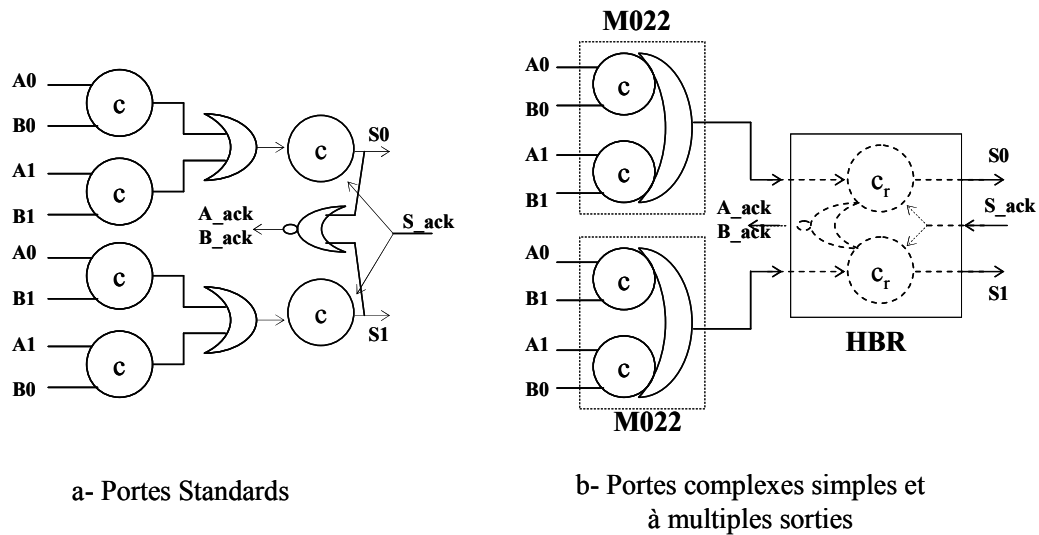


Figure 6.2: Comparaison d’une implémentation d’une fonction ou exclusif en double rail à l’aide des portes standards et des portes complexes d’une bibliothèque dédiée aux circuits asynchrones.

Notre approche est de compléter les bibliothèques existantes en concevant des cellules asynchrones compatibles du point de vue topologique, électrique et physique. Ainsi, il est possible de concevoir des circuits asynchrones en combinant les fonctions disponibles dans les bibliothèques standard et les fonctions des cellules asynchrones ajoutées.

La bibliothèque *TAL-130* est une bibliothèque asynchrone compatible avec la bibliothèque standard *HC MOS9* de *STMicroelectronics*. Les fonctions disponibles sont principalement des portes de Muller, ou *C-Element*. Pour chacune des cellules, les vues requises pour la conception et le placement routage de circuits complexes sont disponibles. Cela inclut le code *VHDL* fonctionnel, le code *Verilog* logico-temporel rétro-annotable, le symbole, le schéma transistor, le schéma électrique après extraction des parasites, la vue *abstract* pour le placement routage, le *layout* et les caractéristiques physiques telles que surface, vitesse et consommation (fichier .lib). L’ensemble de ces vues permet d’appliquer un flot de conception semblable à celui utilisé pour les circuits synchrones. Ces cellules ont été caractérisées avec l’aide de *ST Microelectronics*.

Le développement d’une telle bibliothèque peut être renforcé par la conception de cellules sécurisées. C’est dans ce sens que [Raza04] propose la conception des cellules double rail compacts et sécurisées. Ces travaux sont basées sur la conception des cellules dont la consommation est identique quelles que soient les données d’entrées. L’idée est de concevoir une bibliothèque de cellules protégées

contre les analyses en courant. Cependant, l'efficacité de cette approche dépendra fortement des phases de placement et de routage (capacités parasites).

Afin d'évaluer d'une part le flot de conception sécurisé que nous avons présenté et d'autre part mesurer l'apport de l'utilisation d'une bibliothèque dédiée à la conception des circuits asynchrones, nous avons réalisé deux cryptoprocresseurs asynchrones implémentant l'algorithme de l'*AES*. Ce nouveau standard est appelé à remplacer l'algorithme du *DES*. Les deux cryptoprocresseurs ont été réalisés avec le même flot, mais chaque circuit utilise différents types de cellules et implémente une gestion des signaux d'acquittements différents. Le premier circuit est réalisé à l'aide de cellules standard d'une bibliothèque synchrone avec des acquittements groupés tandis que le second utilise des cellules asynchrones dédiées de la bibliothèque *TAL* avec des acquittements séparés.

6.3 Les cryptoprocresseurs asynchrones *AES*

L'architecture globale des circuits a été définie en partant des spécifications de l'algorithme du Rijndael présentées dans [FIPS197] et de l'environnement de fonctionnement du circuit.

L'architecture des différents blocs des composants *AES* ainsi que leurs signaux d'entrées et sorties sont présentés sur la figure 6.3. L'*AES* est composé de quatre principaux blocs : le banc de registres, les interfaces, le bloc de chiffrement (*AES_core*) et le bloc de calcul à la volée des sous clés (*AES_key*). Les composants *AES* sont destinés à effectuer le chiffrement des blocs de 128 bits de données en utilisant soit des clés de 128, 192 ou de 256 bits. Outre le signal de synchronisation *Ctl* indiquant le début et la fin de la phase de chiffrement, les composants comprennent des fonctions de relecture des différents registres (données, clés et registre de mode).

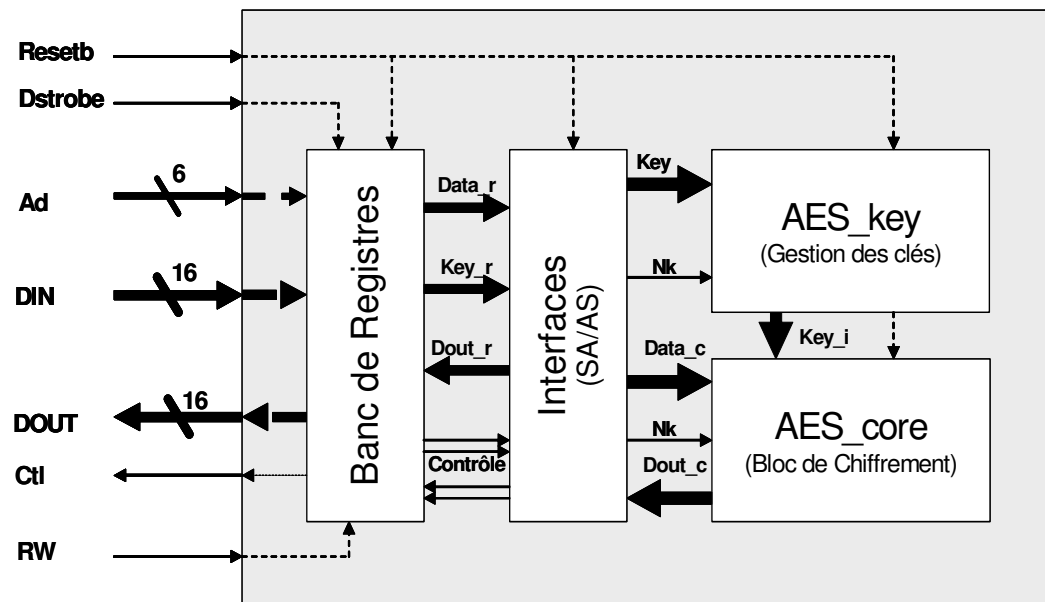


Figure 6.3: Architecture globale du circuit *AES*

6.3.1 Le banc de registres et les interfaces

Le banc de registres permet de mémoriser et d'accéder aux données à chiffrer, aux données chiffrées, à la clé, et au mode. Afin de gérer le début et la fin du chiffrement on implémente le bit de contrôle *Ctl* dans le registre de mode. Après avoir écrit les valeurs de la donnée et celles des clés dans les différents registres, on active dans le registre du mode le bit *Ctl* pour démarrer une phase de chiffrement. Une fois le chiffrement terminé, le circuit désactive le bit *Ctl* pour spécifier la disponibilité du résultat dans les registres de sorties. Le signal *Ctl* décrit précédemment est l'image du bit de contrôle *Ctl* du registre mode.

Le banc de registres est composé d'un registre de 5 bits (registre de mode) et de 32 registres de 16 bits repartis comme suit :

- 8 registres de 16-bit pour les données.
- 16 registres de 16-bit pour la clé, du fait que l'on peut avoir des clés de 128, 192 et 256 bits.
- 8 registres de 16-bit pour les sorties.

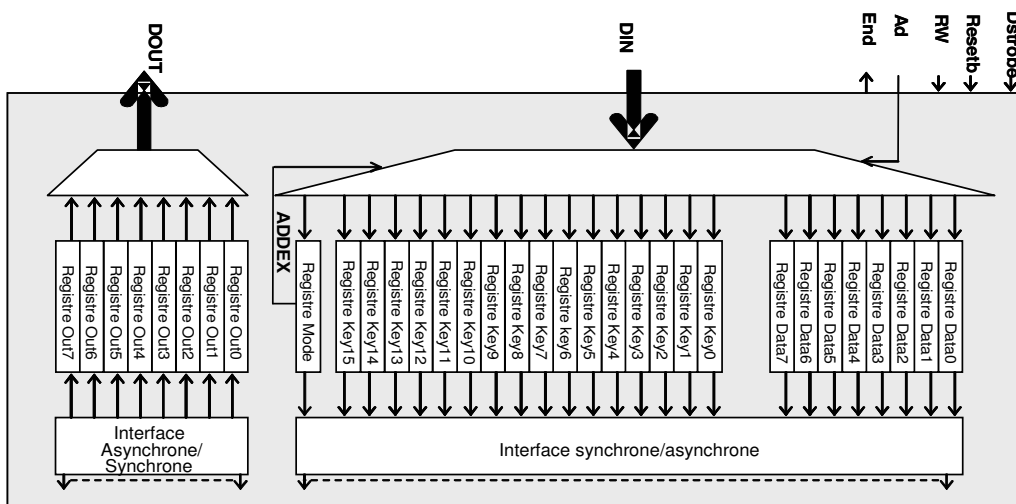


Figure 6.4: Le banc de registres

Les interfaces sont identiques aux interfaces implémentées dans le composant DES (cf. § 4.4.1.2).

6.3.2 Le bloc de chiffrement (*AES_core*) et le bloc de génération des clés (*AES_key*)

L'algorithme de chiffrement du Rijndael est représenté sur la figure 6.5.

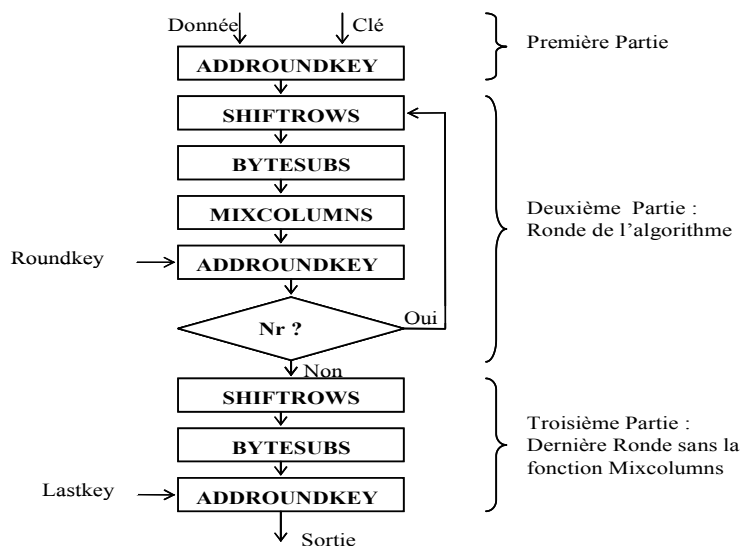


Figure 6.5: Algorithme de chiffrement du Rijndael

Le nombre de bits de la clé permet de spécifier la valeur de Nr correspondant au nombre de rondes ou d'itérations effectuées pour le chiffrement d'un texte de 128 bits. Le schéma des figures 6.6, décrit les chemins et les dépendances des données (octets) au sein de l'algorithme de chiffrement du Rijndael.

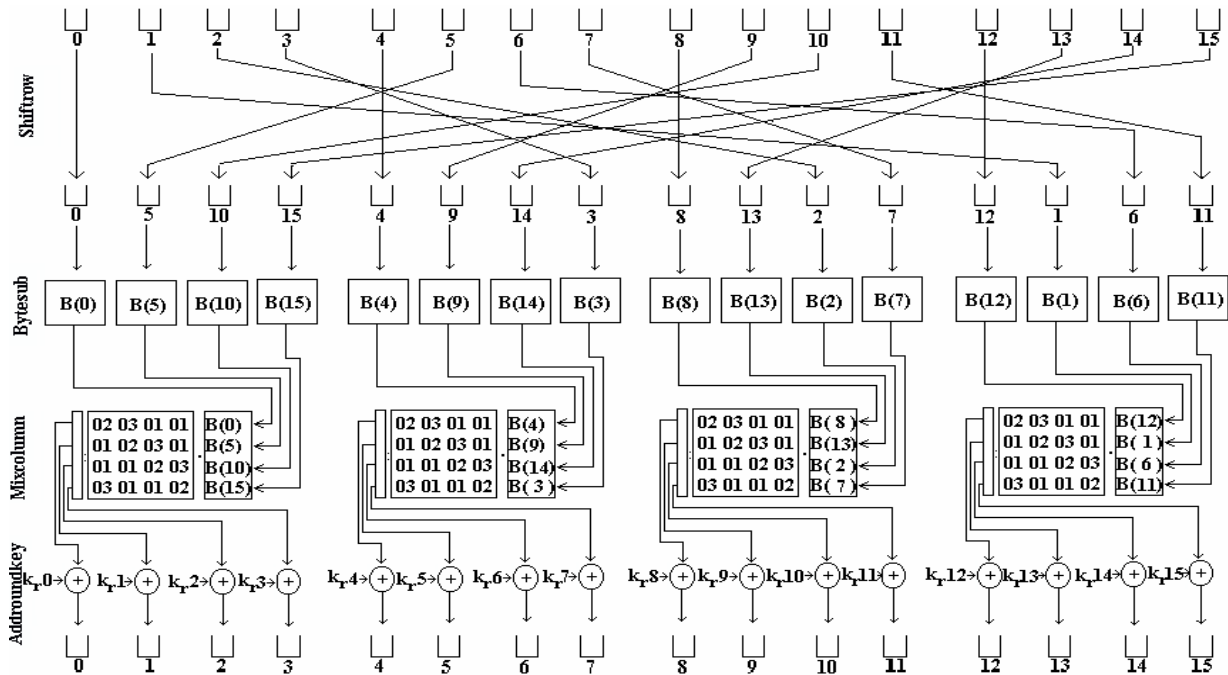


Figure 6.6: Ronde de l'algorithme

Plusieurs possibilités d'implémentation s'offrent selon le mode de fonctionnement de chaque ronde. On entend par là, le nombre d'octets (1, 4 ou 16) à faire circuler entre les blocs. Dans les deux premiers cas (1 ou 4 octets), l'utilisation de registres de mémorisation est nécessaire : au minimum 15 registres d'un octet pour travailler sur 1 octet, et 12 registres pour travailler sur 4 octets. Sans oublier que le nombre d'itérations serait respectivement de 160 et de 40 pour des blocs de clé de 128 bits. La réalisation d'une architecture parallèle sur 16 octets nécessiterait trop de ressources matérielles notamment pour le bloc de substitution (Bytesubs). Le compromis surface/vitesse nous conduit à définir une architecture régulière de chiffrement qui traite des mots de 32 bits. Pour des raisons d'optimisation (surface), un multiplexeur et un démultiplexeur ont été introduit afin d'éviter la duplication des blocs Bytesubs et Shiftrows (figure 6.6).

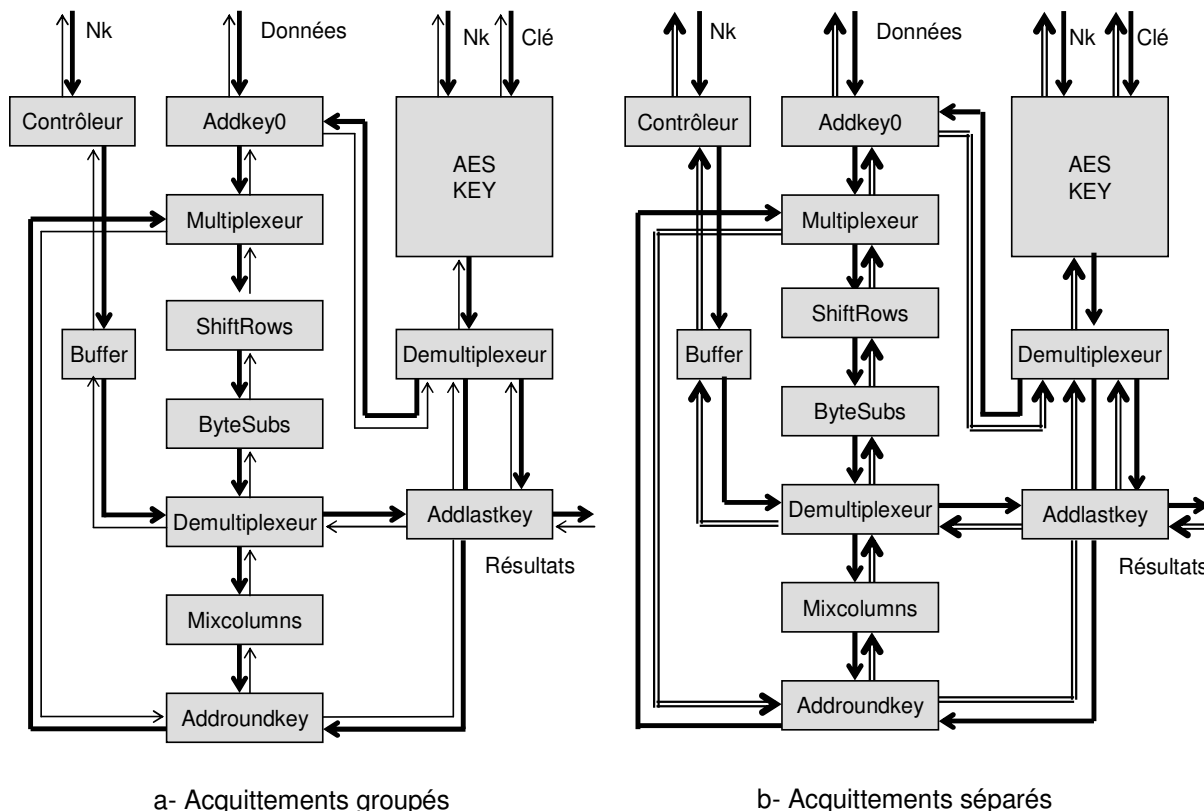


Figure 6.7: Architecture des blocs chiffrement des composants

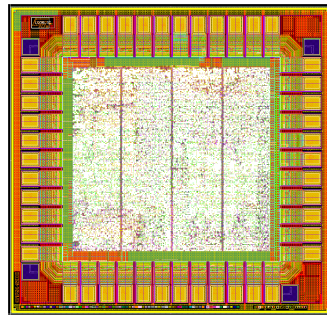
L'architecture du bloc de chiffrement est décrite sur la figure 6.7. Le composant 1 implémente un signal d'acquiescement par vecteur de bits (canaux de N-bit) et le composant 2 utilise pour chaque bit un signal d'acquiescement (canal d'un bit). Le fait d'éclater le signal d'acquiescement permet :

- d'augmenter la vitesse en réduisant la latence de chaque bloc. La latence ne dépend plus d'un signal global d'un bloc, mais uniquement des chemins de données des bits d'un bloc.
- de réduire la surface en supprimant la majorité des arbres d'acquiescement par bloc.
- de désynchroniser les instants de calcul de chaque bloc.

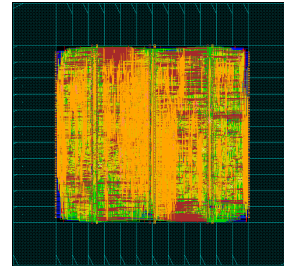
Cette implémentation est proche des techniques de protection proposées dans le chapitre VII (cf. § 7.3). Concernant le bloc *AES_key*, il permet de générer à la volée toutes les sous clés cryptographiques nécessaires au chiffrement d'une donnée. Les choix et contraintes adoptés pour le bloc de chiffrement (*AES_core*) ont été également adoptés dans ce bloc.

6.3.3 Caractéristiques des circuits

Les circuits ont été fabriqués en technologie *CMOS 0.13 μm* de *STMicroelectronics (HCMOS9)*. La figure 6.8 présente les layouts et les principales caractéristiques physiques des circuits.



a- Layout du composant 1



b- Layout du composant 2

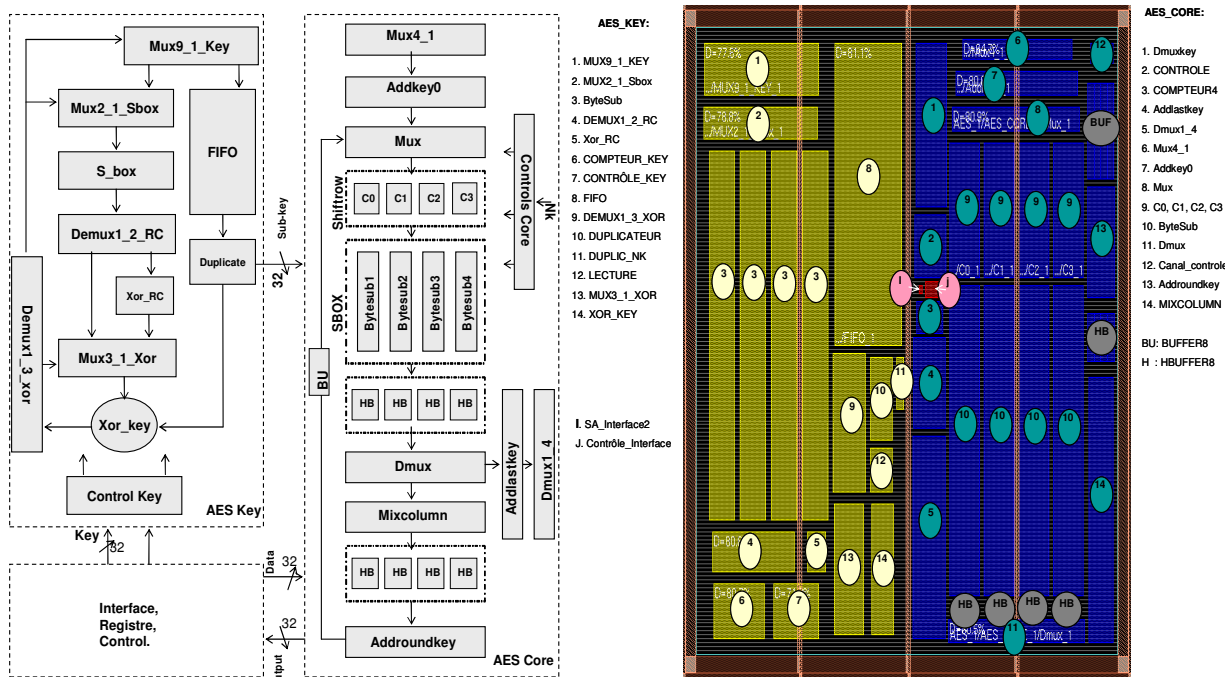
| Technologie | CMOS 0.13 μm (HCMOS9) de STMicroelectronics 6-LM, Tension Alimentation 1.2 V | | | | | | | |
|--------------------|--|---|--------------------------------------|---------------------------|-----------------------|---------------------------------|------------------|--------------------|
| Circuits | Descriptions | Surface core / Surface totale / Nombre de portes équivalentes* | Taux de remplissage/ Densité** | Nombre interconnexions | Longueur de la clé | Temps de Chiffrement (ns) | Energies (nj) | Courant (Moyen) |
| AES Composant 1 | Cellules standards | 0.648 mm ² | 63% | 32860 | 128 bits | 910 | 1.25 | 10mA |
| | | 1.55 mm ² | 63% | | 192 bits | 1.100 | 1.63 | |
| | | 70247 gates | | | 256 bits | 1.400 | 2 | |
| AES Composant 2 | Cellules dédiées | 0.507 mm ² | 60% | 26395 | 128 bits | - | - | - |
| | | 1.217 mm ² | 60% | | 192 bits | - | - | - |
| | | 60336 gates | | | 256 bits | - | - | - |

c- Caractéristiques des cryptoprocresseurs AES
 *porté équivalente : nand2 à 4 transistors (ND2LL)=6.0516 μm^2
 ** nombre de porte équivalente par mm² = 165245.5

Figure 6.8: Layouts et caractéristiques des circuits

Il est à signaler que l'approche hiérarchique de placement routage a un coût, puisque les densités des circuits sont autour de 60% alors qu'elles seraient supérieures à 90% avec un placement routage dit à plat. Enfin, l'utilisation de la bibliothèque asynchrone permet de réduire d'environ 20% le nombre d'interconnexions et la surface du cœur asynchrone du composant 2 par rapport au composant 1 utilisant exclusivement des cellules standard synchrones.

La figure 6.9 présente l'architecture détaillée des blocs de chiffrement et de génération des sous clés ainsi qu'une vue du placement contraint dans laquelle on remarque les différents blocs de l'architecture.



a- Architectures des blocs de chiffrement et de génération des sous clés

b- Floorplan contraint

Figure 6.9: Floorplan et Architectures détaillées des composants

Les tests de fonctionnalités et les mesures du courant ont été réalisés uniquement sur le composant 1 en attendant le retour de fabrication du composant 2. Les caractéristiques en termes de consommation et de temps de chiffrement du composant 1 sont résumés sur le tableau 1.1.

Tableau 6.1 Caractéristiques du composant 1

| Longueur de la clé | Tension d'alimentation : 1.2 volt | | | | Tension d'alimentation : 0.4 volt | | | |
|--------------------|-----------------------------------|--------------|-------------|--------------------|-----------------------------------|--------------|-------------|--------------------|
| | Temps de chiffrement (ns) | Energie (nJ) | Débit | Courant Moyen (mA) | Temps de chiffrement (µs) | Energie (nJ) | Débit | Courant Moyen (mA) |
| 128-bit | 910 | 10.5 | 141 Mbits/s | 12 | 21 | 1.25 | 6.4 Mbits/s | 0.2 |
| 192-bit | 1.100 | 12.5 | 116 Mbits/s | | 23 | 1.63 | 5.5 Mbits/s | |
| 256-bit | 1.440 | 14 | 89 Mbits/s | | 27 | 2 | 4.7 Mbits/s | |

Comme on peut le constater, le circuit reste fonctionnel jusqu'à une tension d'alimentation de 0.4 volt. A cette tension, le courant consommé est divisé par un facteur 60, de 12 mA à la tension nominale (1.2 volt) à 0.2 mA à 0.4 volt. Le temps de chiffrement est ainsi multiplié approximativement par 20 en fonction de la longueur de la clé. Les variations du courant et temps de chiffrement en fonction de la tension sont représentées sur la figure 6.10.

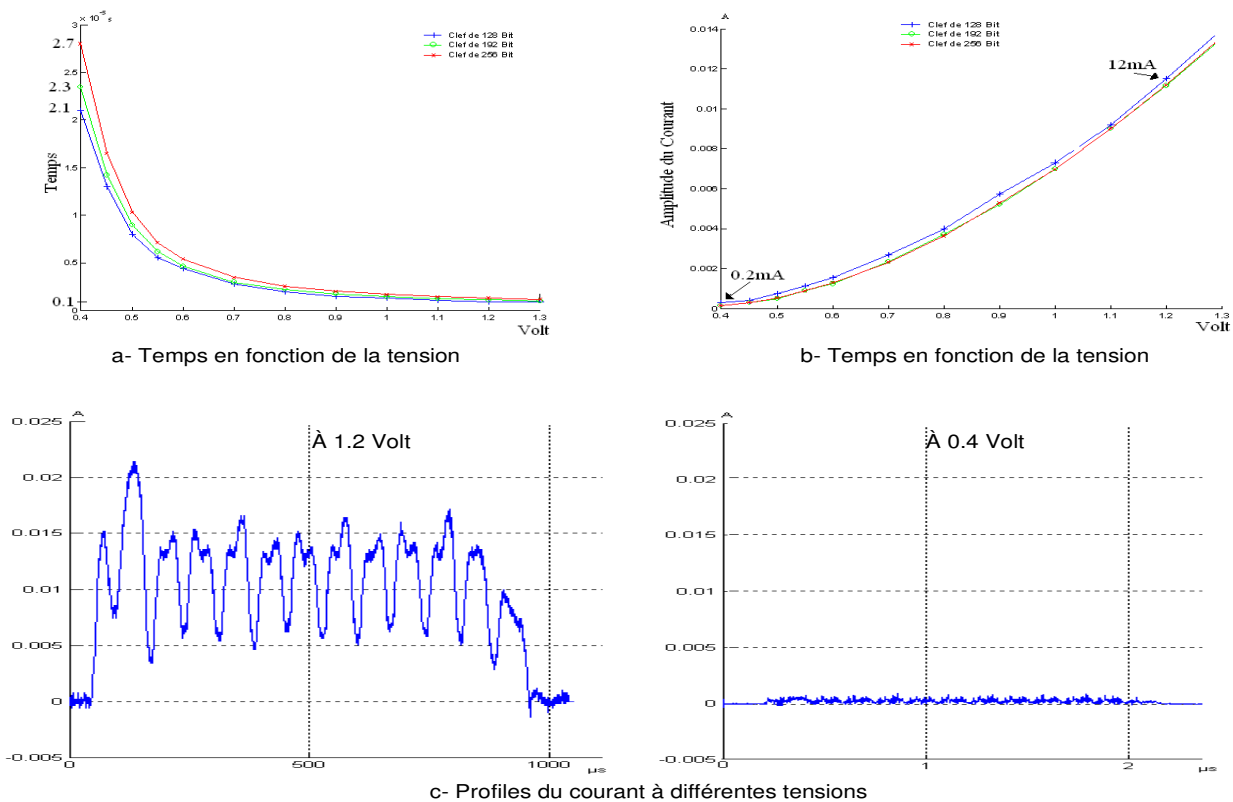


Figure 6.10: Variations du courant et du temps de chiffrement en fonction de la tension d'alimentation. Profil du courant

Parce que la logique asynchrone à travers le principe de l'insensibilité aux délais permet de concevoir des circuits qui sont fonctionnels sans aucune contrainte et hypothèse temporelle, il est possible de réduire considérablement le courant consommé en réduisant la tension d'alimentation au détriment du délai de calcul. La largeur de l'intervalle des tensions d'alimentation possible donne au concepteur une bonne marge de manœuvre afin d'appliquer une politique de faible consommation adéquate. Cette propriété peut être exploitée pour cibler des circuits à très faible consommation ou à faible émission électromagnétique [Boue05].

6.3.4 Résultats des analyses en courant

Les attaques réalisées sur le composant et les premiers résultats d'attaques présentés sur ce composant ont été effectuées par le *Centre d'Évaluation de la Sécurité des Technologies de l'Informations (CESTI)* de Toulouse dans le cadre d'un projet d'étude de la technologie asynchrone avec la *Direction Centrale de la Sécurité des Systèmes d'Informations (DCSSI)*. La première partie de l'attaque consiste à analyser en *SPA* les variations du profil de courant en fonction de différents messages et clés aléatoires, ceci, afin d'identifier des zones de sensibilités dans le circuit. Une fois identifiées, des analyses approfondies sont effectuées pour corrélérer ces zones aux opérations de l'algorithme. On identifie dans la courbe de courant les correspondances entre les pics et les fonctions de l'algorithme. Par la suite, ces différences de consommation à ces instants précis sont exploitées pour réaliser des attaques *DPA* ou *CPA*.

Afin de supprimer les différents bruits, notamment le bruit de désynchronisation des courbes (*Jitter*) pendant la phase d'acquisition du courant, chaque courbe collectée est moyennée pour plusieurs exécutions de l'algorithme avec la même donnée et clé. La figure -6.11 présente le profil de courant de l'*AES* à 128 bits de clé à 0.5 volts dans lequel on peut identifier les 10 itérations de chiffrement. L'agrandissement des deux premières rondes est également illustré entre une courbe moyennée et instantanée.

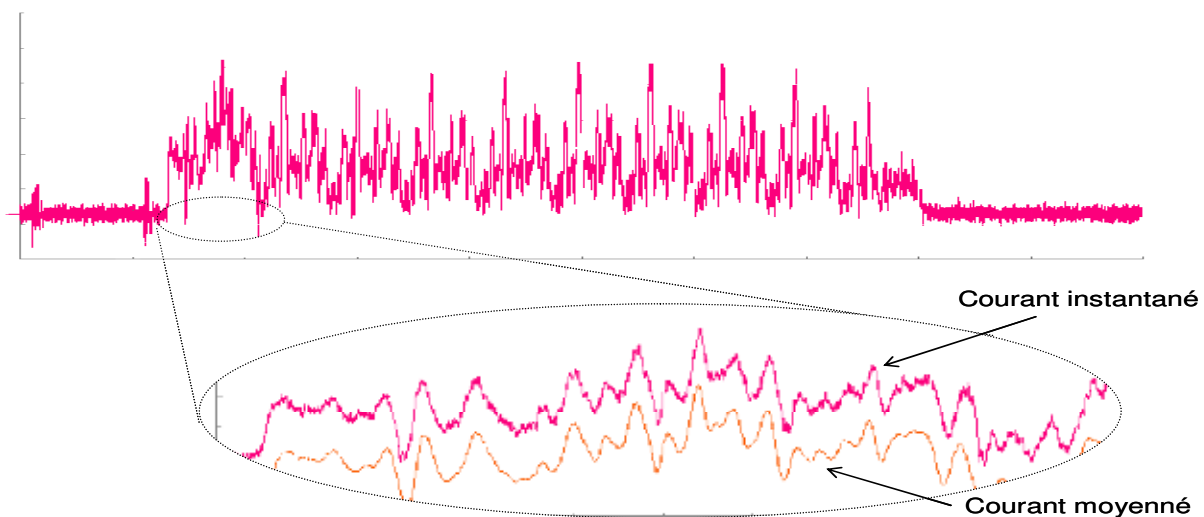


Figure 6.11 : Profil de courant de l'*AES* 128 bits de clé à 0.5 volt, courbes de courant moyennée sur 128 et instantanée des premières rondes

Les analyses *SPA* réalisées sur l'octet 15 des données ont montré des dépendances entre les courbes de courant et les données manipulées susceptibles d'être exploiter pour implémenter une attaque *DPA*. En effet, pour une clé fixe, toutes les valeurs possibles de l'octet 15 du message ont été calculées de sorte à avoir 256 courbes de courant moyennées. L'analyse de ces 256 courbes est présentée sur la figure 6.12 dans laquelle on peut remarquer les variations du courant en fonction des données.

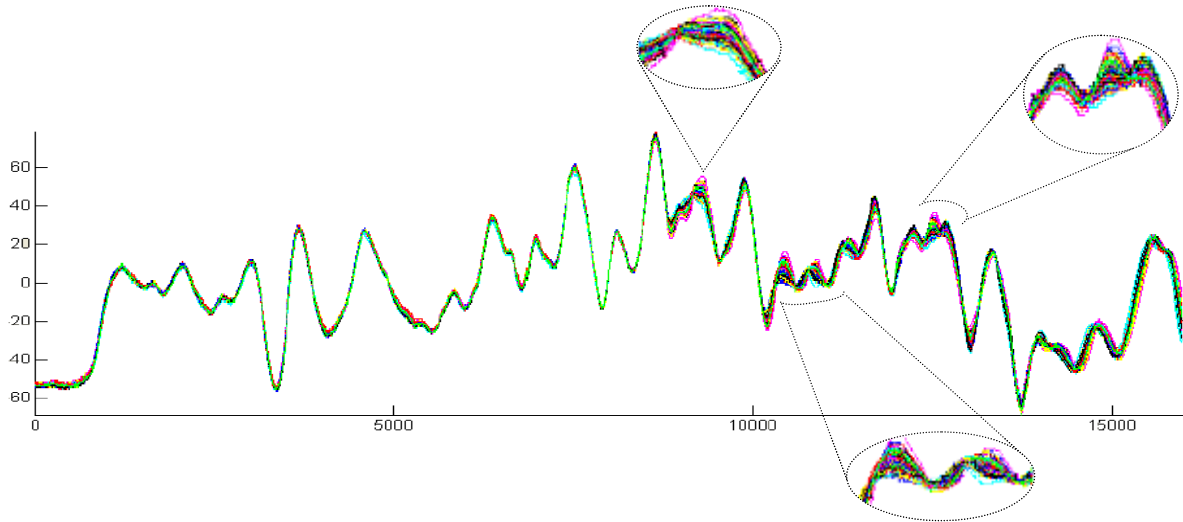


Figure 6.12: Analyse des 256 courbes correspondant aux 256 valeurs de l'octet 15 du message d'entrée pour une clé fixe.

Ces analyses ont permis la réussite des attaques différentielles en courant sur les bits de cet octet. La fonction de sélection est telle que :

$$D(m_{15}, k_{15}) = \sum_{j=0}^7 d_j 2^j = \text{ByteSubs4}(m_{15}, k_{15}) \quad (6.2)$$

La fonction de répartition correspond aux valeurs prises par le bit de sortie d_j du bloc *ByteSubs4* de manière à avoir comme ensemble de répartition :

$$\begin{aligned} S0_{d_j} &= \{C_i / [D(m_{15}, k_{15})] \mid d_j = 0\} \\ S1_{d_j} &= \{C_i / [D(m_{15}, k_{15})] \mid d_j = 1\} \\ &\text{avec } 0 \leq j \leq 7 \end{aligned} \quad (6.3)$$

C_j correspond au courant consommé lors du calcul du message m_j . Les bits des messages considérés sont aléatoires exceptés ceux de l'octet 15 qui sont pris entre 00 et 0xFF. On effectue les acquisitions des 256 courbes de courant dont chaque courbe est mesurée k fois (soit $k=16$ ou $k=32$). Ceci correspond à une attaque **DPA** sur 0.5 Million de courbes ($k=16$) ou 1 Million de courbes ($k=32$), sachant que chaque courbe est une moyenne de 128 exécutions. La figure 6.13 présente le résultat obtenu lorsque le bit de poids fort en sortie de la **ByteSubs4** est attaqué avec plus d'un million de courbes.

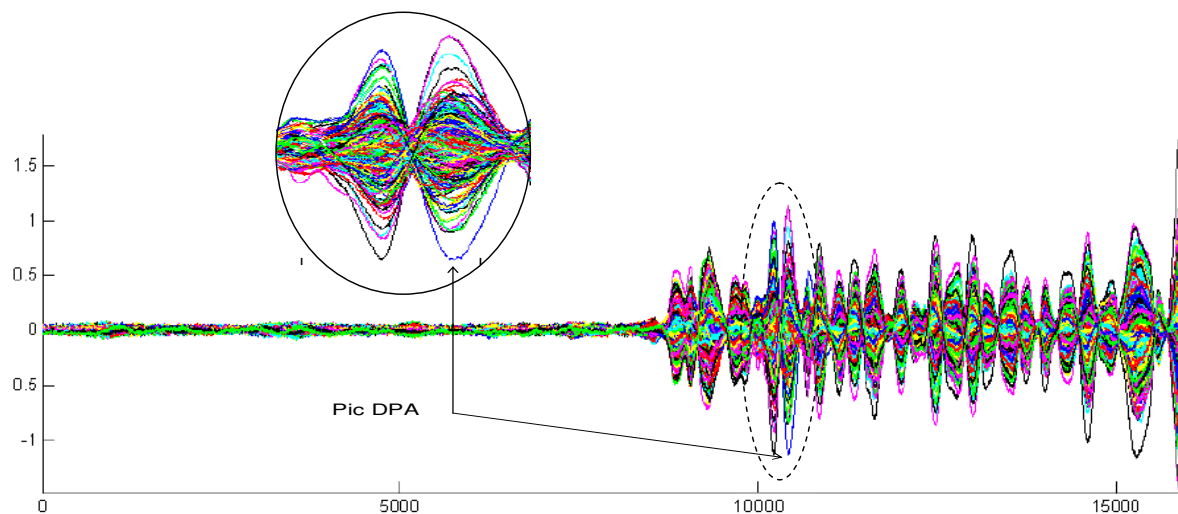


Figure 6.13: Pic **DPA** du bit de poids fort en sortie de la **ByteSubs4** qui correspond au calcul de l'octet 15. 8192*128 courbes sont utilisées.

Le niveau de résistance atteint par cette approche est quasi identique au niveau de sécurité présenté par les circuits **DES** asynchrones. Ces résultats sont confirmés par les attaques par corrélation (**CPA**) réalisées sur le bit de poids faible de l'octet 15. En effet, l'équation de l'attaque par corrélation (eq. 2.27) définie au chapitre II (cf. § 2.6) a été appliquée sur les 2 rails du bit 0 de l'octet 15 en sortie de **ByteSubs4**. Les premiers résultats obtenus montrent qu'il est possible avec ce type d'attaque de réduire considérablement le nombre de courbes nécessaires pour réussir une attaque. La figure 6.14 présente le résultat d'une **CPA** avec près de 1024*128 courbes utilisées.

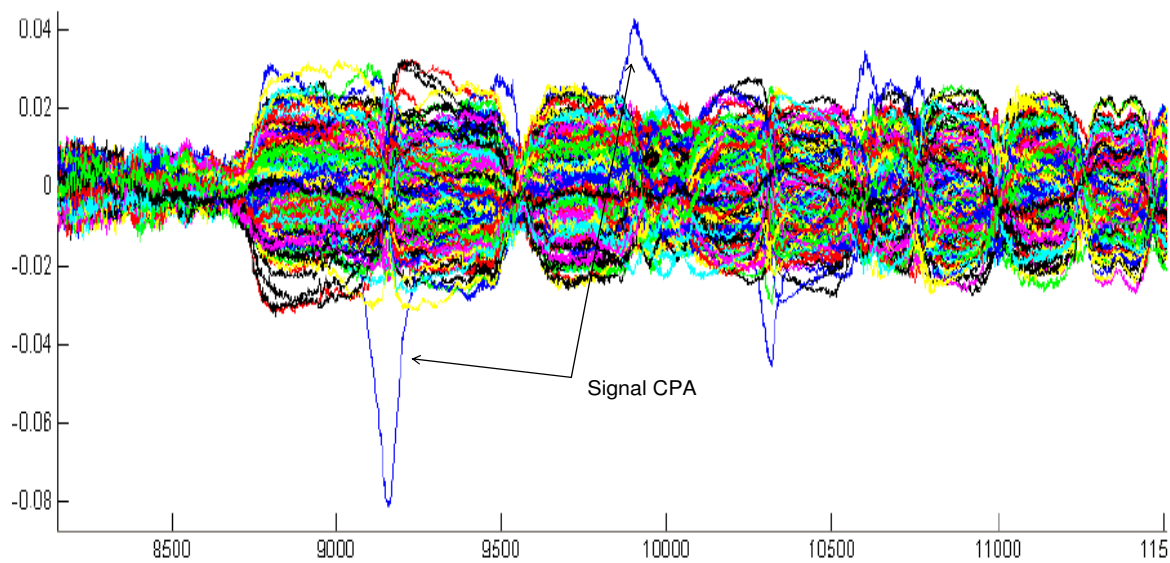


Figure 6.14: Attaque par corrélation sur le bit 0 en sortie de la *ByteSubs4*. 1024*128 courbes utilisées.

L'impossibilité de contraindre les phases de placement-routage, de sorte que le critère de sensibilité électrique (6.1) défini au paragraphe 6.2.1 soit pris en compte, explique la réussite des attaques effectuées sur certains bits de l'architecture. En effet, des dissymétries électriques et logiques introduites pendant les phases de placement et routage et pendant la phase d'optimisation électrique (ajout de Buffer) sont exploitées afin de réussir des attaques en puissance.

Toutefois, ce flot de conception sécurisé a permis de définir un outil automatique qui permet de déterminer les nœuds sensibles (en terme de *DPA*) d'une architecture. Cette sensibilité est détectée en fonction du critère de sensibilité définie pour évaluer la symétrie électrique. Des analyses sont en cours d'évaluation afin de comparer les nœuds sensibles détectés par l'outil et ceux attaqués sur le composant.

6.4 Conclusion

Nous avons, dans ce chapitre, spécifié un flot de conception pour circuit sécurisé contre les attaques par analyses différentielles. Ce flot est basé sur la représentation formelle des circuits asynchrones *QDI* sous forme de graphe orienté afin d'analyser à tous les niveaux de conception la sensibilité du circuit face à ces types d'attaques. La réalisation des circuits de prototypes à partir de ce flot a permis d'évaluer et de pointer les limites d'un tel flot notamment sur les contraintes à imposer lors des phases de 'back-end'.

Les résultats obtenus nous permettent d'une part, d'établir des corrélations entre les chemins de données des bits attaqués et le critère de sensibilité dc_i que nous avons défini. Les résultats de ces analyses vont nous permettre de déterminer, pour une technologie donnée, qu'elle différence en terme de capacité on peut tolérer lors des phases de placement routage. Ces analyses sont en cours et ne pourront pas être présentées dans ce manuscrit. Toutefois, quelques soient les conclusions que l'on obtiendra de ces analyses, les résultats des attaques sur les circuits, montrent la nécessité de développer de nouvelles contre-mesures plus efficaces afin de renforcer la protection des circuits asynchrones **QDI**.

CHAPITRE VII

PROTECTION DES CIRCUITS QUASI INSENSIBLES AUX DELAIS : CONTRE-MESURES

7.1 Introduction

Les résultats des attaques, présentés dans le chapitre précédent sur le cryptoprocresseur *AES* réalisée en logique *QDI* avec un placement des cellules contraint, ont démontré la nécessité d'accroître la sécurité de ce type de circuit en développant des nouvelles contre-mesures. Nous proposons dans ce chapitre d'exploiter les propriétés des circuits *QDI* notamment la redondance de structures logiques et la gestion des signaux d'acquittement pour mettre en place des techniques de protections adaptées à ce type de logique.

En effet, les techniques que nous développons dans ce chapitre pour améliorer la résistance des circuits *QDI* sont basées sur des décalages aléatoires spatiaux et temporels des signatures électriques des blocs des circuits. Ces techniques, à savoir la méthode par transposition aléatoire des chemins de données et la méthode par décalage aléatoire des signaux d'acquittement, ne suppriment pas les dissymétries observées sur les capacités parasites, mais tendent à les rendre inexploitable.

En se basant sur le modèle formel des circuits *QDI* défini dans le chapitre V, nous allons, dans ce chapitre, montrer la pertinence des méthodes proposées. Ces méthodes sont analysées formellement et les résultats obtenus sont illustrés à l'aide de simulations électriques effectuées sur des cryptoprocresseurs *DES*.

7.2 Méthode par transpositions aléatoires des chemins de données

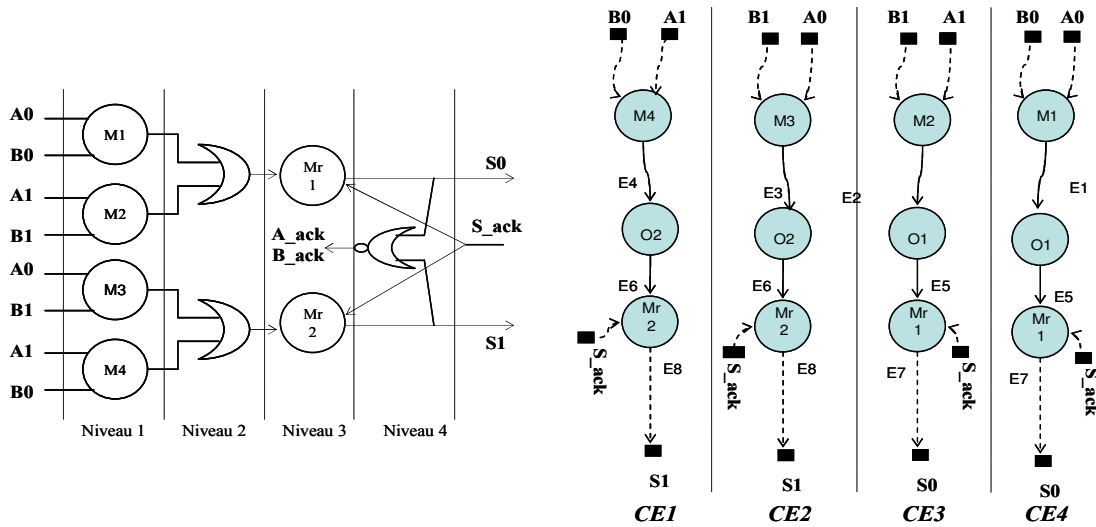
Dans cette partie, nous présentons l'approche par transpositions aléatoires des chemins de données afin d'augmenter la sécurité des circuits *QDI* contre les attaques par analyse en courant. Cette approche exploite les propriétés de symétrie logique des circuits *QDI* et plus particulièrement la redondance de ses chemins de données de manière à rendre toutes les courbes de courant inutilisables lors de l'implémentation d'une attaque en courant. L'idée est de moyenniser toutes les signatures électriques d'un bloc en intervertissant aléatoirement les chemins de données pendant les calculs.

L'implémentation de cette méthode est faite à travers le modèle formel de représentation graphique des circuits *QDI* que nous avons décrit dans le chapitre V (cf. § 5.2). Cette représentation qui permet de formellement vérifier la symétrie logique de tous les chemins de données dans un bloc, est utilisée de manière à mettre en œuvre la méthode de transposition des chemins de données. D'autre part, les possibilités de modélisation des signatures électriques offertes par ce modèle, nous permettent d'appliquer une attaque *DPA* dans le but d'évaluer formellement très tôt dans les phases de conceptions, l'efficacité de l'approche.

7.2.1 Transpositions aléatoires des chemins de données : Définition

L'objectif de cette nouvelle approche est de rendre inutilisable les effets électriques qui permettent de réussir les attaques en puissance sur les circuits *QDI*. Pour cela, nous exploitons les symétries logiques des blocs *QDI* car, dans ce type de bloc, il existe plusieurs chemins physiques identiques entre les entrées et les sorties primaires.

En effet, nous avons introduit, dans le chapitre V (cf. § 5.3.1) la notion de chemin exécution que nous avons utilisé pour définir la symétrie logique dans un bloc. Un bloc est symétrique au plan logique si et seulement si les digraphes de ses chemins d'exécutions sont isomorphes. Par exemple, en reprenant l'implémentation de l'opérateur *Xor* en double rail décrit sur la figure 4.12, on en déduit les 4 chemins d'exécutions identiques représentés sur la figure 7.1-b.



a- Implémentation de la fonction *Xor* en double rail

b- Digraphe de la fonction *Xor*:
CEi : Chemins d'Exécution *i*

Figure 7.1: Schéma d'implémentation et digraphe de la fonction *Xor* en double rail : Chemins d'exécutions identiques.

Chacun de ces 4 chemins d'exécutions est constitué d'une porte de *Muller* au premier niveau logique, d'une porte *Or* à deux entrées au second niveau, d'une porte de *Muller* comprenant un signal d'initialisation au troisième niveau logique et d'une porte *Nor* à deux entrées au dernier niveau. Dans une telle structure, où tous les chemins d'exécutions sont identiques, on peut appliquer sur chacune des entrées de chaque chemin d'exécution les différentes valeurs d'entrées des autres chemins d'exécution (figure 7.2). Ainsi, pour une valeur possible en entrée, 4 différentes signatures électriques peuvent être obtenues en intervertissant les entrées et les sorties. Chacun des couples d'entrées $(A0, B0)$, $(A1, B1)$, $(A1, B0)$, $(A0, B1)$, peut être calculé par chacun des chemins d'exécutions *CE1*, *CE2*, *CE3* et *CE4*. La figure 7.2 représente les différentes possibilités de permutations dans la fonction *Xor* en double rail. Le choix du chemin à utiliser pendant chaque calcul est rendu aléatoire.

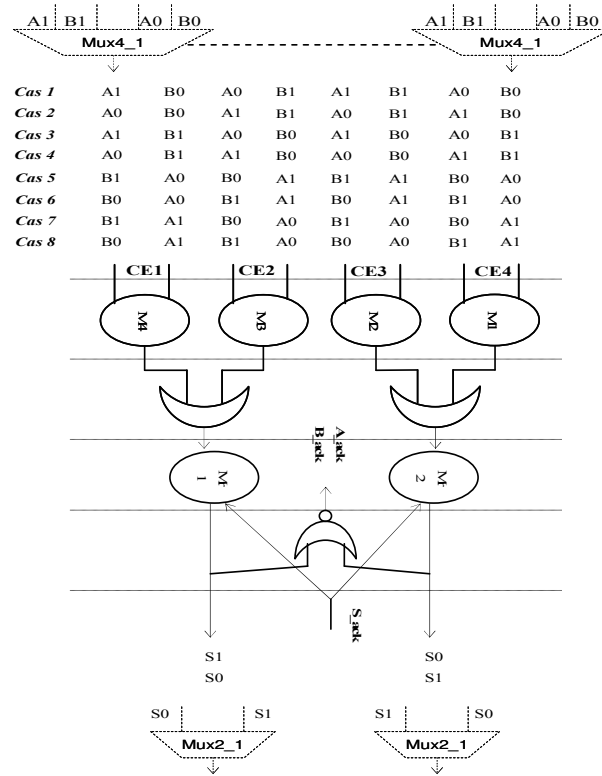


Figure 7.2: Transpositions des chemins de données dans une implémentation d’une fonction logique *Xor* en double rail.

Cette méthode est appelée transpositions aléatoires des chemins de données car le fait d’intervertir aléatoirement les entrées et les sorties d’un bloc équivaut à permuter de façon aléatoire les chemins logiques (physiques) à l’intérieur du bloc.

Plus formellement, soit n_c le nombre de canaux de sortie (bits de sortie) d’un bloc codé en 1 parmi n et n_{id} le nombre de chemin d’exécution du bit i (ou canal de sortie i). On peut représenté l’équation logique d’un chemin d’exécution par $f_{ij}(A_x)$ et son profil de courant dynamique par $P_{cij}(t/A_x)$ avec A_x la valeur à calculer. A_x représente l’une des valeurs possibles de l’ensemble E_i des entrées excitant le canal i et j correspond au numéro du chemin d’exécution ($j \in n_{id}$). Si tous les chemins d’exécutions sont équilibrés logiquement des entrées aux sorties primaires du bloc, alors :

$$\forall A_x \in E_i \Rightarrow f_{i1}(A_x) = \dots = f_{in_{id}}(A_x)$$

et

$$P_{cij}(t/A_x) \neq \dots \neq P_{cn_{id}}(t/A_x)$$

(7.1)

L'équation 7.1 montre que pour toute valeur d'entrée A_x d'une structure symétrique possédant n_{id} chemins d'exécutions identiques, on peut acquérir n_{id} signatures électriques différentes. En rendant le choix des chemins aléatoires, on supprime toute corrélation possible entre les données manipulées et les signatures électriques.

La réalisation de cette approche nécessite l'utilisation de multiplexeurs, de démultiplexeurs et d'un générateur aléatoire. Les multiplexeurs (démultiplexeurs) sont utilisés pour permuter les entrées et les sorties et sont contrôlés par le générateur aléatoire. L'usage d'un générateur aléatoire permet de garantir une distribution équiprobable des entrées sur les différents chemins logiques du bloc. Si M données doivent être calculées par une fonction possédant 4 chemins d'exécutions ($n_{id}=4$), alors chaque chemin doit être excité au moins $M/4$ fois de façon aléatoirement.

Cette méthode de transpositions des chemins de données peut être réalisée efficacement dans des circuits offrant la possibilité d'implémenter des fonctions logiques équilibrées comme c'est le cas des circuits **QDI** sans pour autant augmenter considérablement la surface.

7.2.2 Application de la **DPA** sur le modèle formel avec transpositions des chemins de données

En se basant sur le modèle électrique des circuits **QDI** défini au chapitre V (cf.§ 5.4.1), on applique une attaque **DPA** sur la fonction **Xor** de la figure 7.2 implémentant la méthode d'intervention des chemins de données. Comme tous les chemins de données sont utilisés pour calculer les sorties, alors les moyennes en courant des deux ensembles de répartitions S_0 et S_1 sont identiques et ont pour expression :

$$A_{xor0}(t) = A_{xor1}(t) = \frac{1}{4}(I_{11}(t_1) + I_{12}(t_1) + I_{13}(t_1) + I_{14}(t_1) + I_{21}(t_2) + I_{22}(t_2) + I_{31}(t_3) + I_{32}(t_3) + I_{41}(t_4) + I_n(t)) \quad (7.2)$$

Ce qui a pour effet d'annuler le signal de **DPA**

$$S_{DPA}(t) \approx 0 \quad (7.3)$$

Ces équations démontrent que tout analyse différentielle sur de telles structures équilibrées est complètement impossible lorsque leurs chemins de données sont intervertis aléatoirement. Rappelons que ces structures sont dites équilibrées parce qu'elles possèdent un nombre constant de transitions logiques,

un même nombre de transition, un même type de transitions et un même nombre de portes sur leurs chemins de données.

7.2.3 La fonction de transposition des chemins de données

Comme cela à été démontré dans le chapitre V (cf.§ 5.4.1), la réalisation d'une attaque *DPA* sur un bit utilisant un codage de type *1 parmi n*, revient à analyser les différents chemins de données de chacun de ses *n*-rail. Ce constat permet d'optimiser le nombre de permutations nécessaires à l'implémentation de la méthode par transposition des chemins de données.

En effet, considérons une attaque sur le bit C_i codé en *1 parmi n* de la fonction de sélection D . E_j représente l'ensemble des valeurs d'entrées qui activent le rail j du bit C_i et m_j représente le nombre total d'éléments dans l'ensemble E_j . Il existe deux types d'approches d'implémentation : l'approche non déterministe et l'approche déterministe.

- **L'approche non déterministe** : Dans cette approche, le nombre de permutations possibles d'un élément d'entrée est donné par l'expression suivante :

$$P_{pe} = \sum_{j=1}^n m_j \quad (7.4)$$

P_{pe} le nombre de permutations possibles
d'un élément d'entrée

Ce nombre, met en évidence deux points : premièrement, les éléments d'un même ensemble peuvent être permutés entre eux. Deuxièmement, cette approche nécessite pour chaque élément A_x de E_j ($A_x \in E_j$), l'utilisation de P_{pe} multiplexeurs de type P_{pe} vers 1 (P_{pe} entrées, une sortie). Ainsi, le nombre et le type de multiplexeurs nécessaires à l'implémentation de la méthode sur le canal C_i est donné par :

$$N_{C_i} = P_{pe}^2 \text{ Multiplexeurs de } P_{pe} \text{ vers } 1 \quad (7.5)$$

Ce nombre peut être réduit si les permutations des éléments d'un même ensemble sont supprimées.

$$Nc_i = \sum_{j=1}^n \left[m_j \text{ Multiplexeurs de } \left(\left(\sum_{k=1; k \neq j}^n m_k + 1 \right) \rightarrow 1 \right) \right] \quad (7.6)$$

Si tous les ensembles E_j sont égaux (possèdent le même nombre d'éléments), alors :

$$Nc_i = P_{pe} \text{ Multiplexeurs de } (P_{pe} - m_i + 1) \rightarrow 1 \quad (7.7)$$

Pour une valeur de $n=2$ et pour des ensembles E_0 et E_1 contenant chacun 2 éléments ($m_0=2$ et $m_1=2$) comme c'est le cas avec l'exemple de la figure 7.2, on obtient 4 multiplexeurs de 4 vers 1 qui peuvent être réduits à 4 multiplexeurs de 3 vers 1.

- **L'approche déterministe** : le but de cette approche est de contraindre les permutations de manière à optimiser le type de multiplexeur. L'idée est de permuter un élément de E_j avec un élément des autres ensembles E_k ($k \in n$ et $k \neq j$) du canal C_i . Ainsi, chaque élément peut être permuter n fois (n ensembles) et cela nécessite P_{pe} multiplexeurs de n vers 1.

$$Nc_i = P_{pe} \text{ Multiplexeurs de } n \text{ vers } 1 \quad (7.8)$$

En considérant l'exemple précédent, on obtient 4 multiplexeurs de 2 vers 1. Le fait de connaître la fonction de permutation (les valeurs permutées) n'affecte pas l'efficacité de la contre-mesure, car le choix du chemin pendant chaque calcul est aléatoire. Ce point peut être également utilisé pour optimiser l'usage des multiplexeurs.

De manière générale, on exploite les symétries et les régularités des architectures du circuit pour une mise en œuvre efficace de la méthode. Il n'est pas nécessaire d'implémenter des multiplexeurs dans chaque bloc de l'architecture.

Ces analyses sont entre autre valables pour les démultiplexeurs et peuvent être étendues sur tous les canaux d'un bloc.

7.2.4 Discussion

Quelque soit le niveau de sécurité apporté, cette nouvelle méthode est performante si et seulement si les deux conditions suivantes sont respectées :

- **Exécution aléatoire des transpositions des chemins de données** : l'objectif est d'assurer une interversion imprédictible des chemins à l'intérieur du bloc. Une attaque est toujours possible si la fonction de permutation aléatoire des différents chemins est découverte. Connaissant l'ordre de permutation des chemins de données, l'attaque peut être focalisée sur les calculs utilisant la même fonction de transposition. Par exemple, si la fonction aléatoire commute toujours entre deux transpositions (le cas 1 et le cas 2 de l'exemple décrit sur la figure 7.2), l'implémentation d'une attaque en prenant uniquement en compte les informations du premier cas correspond à l'implémentation d'une attaque standard sans contre-mesure. La situation est la même si l'un des rails du bit attaqué est toujours évalué par le même chemin de données. Ainsi, le générateur aléatoire doit être imprédictible et équiprobable. L'implémentation de tels générateurs n'est pas prise en compte dans le contexte de ces travaux de recherche. Les générateurs pseudo aléatoires actuellement implémentés dans de nombreux circuits intégrés permettraient d'atteindre un niveau de sécurité satisfaisant.
- **L'implémentation des multiplexeurs/démultiplexeurs** : l'une des méthodes pour s'affranchir des effets du générateur aléatoire est l'utilisation de ces blocs (multiplexeurs/démultiplexeurs) comme fonction de sélection pour une attaque *DPA*. Si le multiplexeur d'un canal A_i codé en double rail (A_{i0} et A_{i1}) présente une signature significative quand ses rails sont intervertis, la fonction du générateur aléatoire qui contrôle ce multiplexeur peut être déterminée. Une attention particulière doit être portée dans les phases de conception de ces blocs et notamment dans les phases de back-end afin de limiter au minimum les dissymétries introduites sur les capacités de sorties des portes. En imposant uniquement des contraintes de placement et de routage sur ces blocs de l'architecture, il est alors possible de faire converger le critère de sensibilité d_{ci} vers sa valeur minimale.

7.2.5 Etude de cas : le cryptoprocresseur asynchrone DES

L'exemple choisi pour valider cette approche est le cryptoprocresseur *QDI* implémentant l'algorithme du *DES*. Ce cryptoprocresseur est quasi identique en terme d'architecture à celui décrit et

testé dans le chapitre IV (cf. § 4.4). Ainsi on part de cette netlist initiale pour implémenter et évaluer la contre-mesure.

Nous avons dans cette implémentation, exploité les symétries dans l'architecture entre les différents blocs du *DES* afin de limiter l'usage des multiplexeurs/démultiplexeurs. Le choix de l'architecture adopté pour réaliser l'algorithme nous permet d'implémenter uniquement des multiplexeurs dans 4 différents blocs de l'architecture : dans les *Registre L*, les *Registres R*, les *Registres* de la boucle des sous clés et dans les blocs des fonctions de substitutions *SBOX* (en en gras sur la figure 7.13).

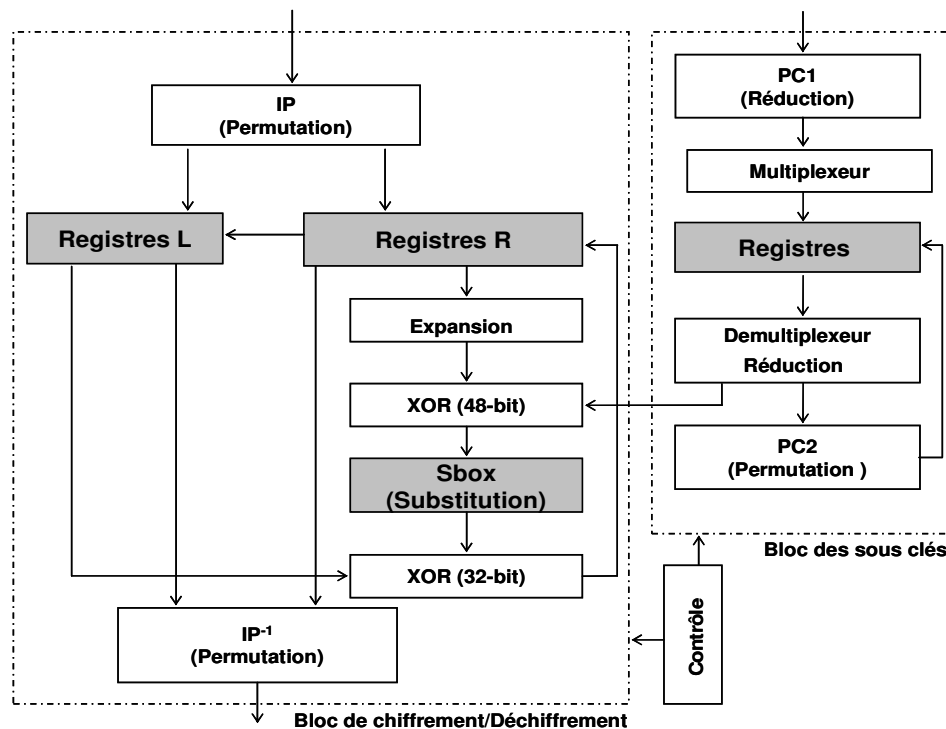


Figure 7.3: Architecture du *DES* implémentant la méthode de transpositions des chemins de données.

Les fonctions de substitutions sont prises en compte car elles représentent les seules fonctions injectives de l'algorithme. Si l'on permute les entrées de ces fonctions, il ne sera plus possible de retrouver l'information en fonction du résultat obtenu. Chaque bloc de substitution comprend 6 bits en entrée ($A_0, A_1, A_2, A_3, A_4, A_5$) et 4 bits en sortie (C_0, C_1, C_2, C_3). Une valeur de sortie peut être sélectionnée par 4 différentes valeurs d'entrées. En double rail, on obtient 8 ensembles de 32 éléments chacun. L'application de l'approche non déterministe équivaut à implémenter 64 multiplexeurs de 64 vers 1, ce qui n'est pas

optimal en terme de surface. Par conséquent, nous avons implémenté l'approche déterministe en exploitant au maximum les redondances de la fonction de substitution.

En effet, considérant la première fonction de substitution du *DES* dont les valeurs sont représentées sur le tableau 7.1. La valeur décimale des bits d'entrées A_0 et A_5 représente le numéro de la ligne de sélection et les valeurs des bits A_1, A_2, A_3 et A_4 représentent la valeur de la colonne.

Tableau 7.1 Tableau de permutation du premier bloc de substitution du *DES* (*SBOX1*)

| | 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 |

↓ Numéro de la colonne

↑ Numéro de la ligne

Afin d'invertir efficacement les chemins d'exécution de ce bloc (*SBOX1*), nous avons regroupé dans les mêmes ensembles toutes les valeurs d'entrées qui génèrent une valeur de sortie et son inverse. Par exemple, toutes les valeurs d'entrées qui génèrent 0x0 en sortie, sont regroupées dans le même ensemble E_0 que les valeurs qui génèrent en sortie 0xF. Le tableau 7.2 présente cette réorganisation de la *SBOX1*.

Cette réorganisation permet d'observer qu'il est possible de permuter uniquement les valeurs d'entrées qui pointent sur la même ligne de sélection dans le même ensemble (dans l'ensemble E_0 , la valeur 14 sera permuté avec la valeur 5 de la ligne 0, celle de 0 avec la valeur 1 de la ligne 1, les valeurs 15 et 8 de la ligne 2 sont interverties entre elles et la valeur 13 est permutée avec la valeur 0 sur la ligne 3). Ainsi on utilise 32 multiplexeurs de 2 vers 1. Ce qui représente une augmentation de la surface du bloc de 30%.

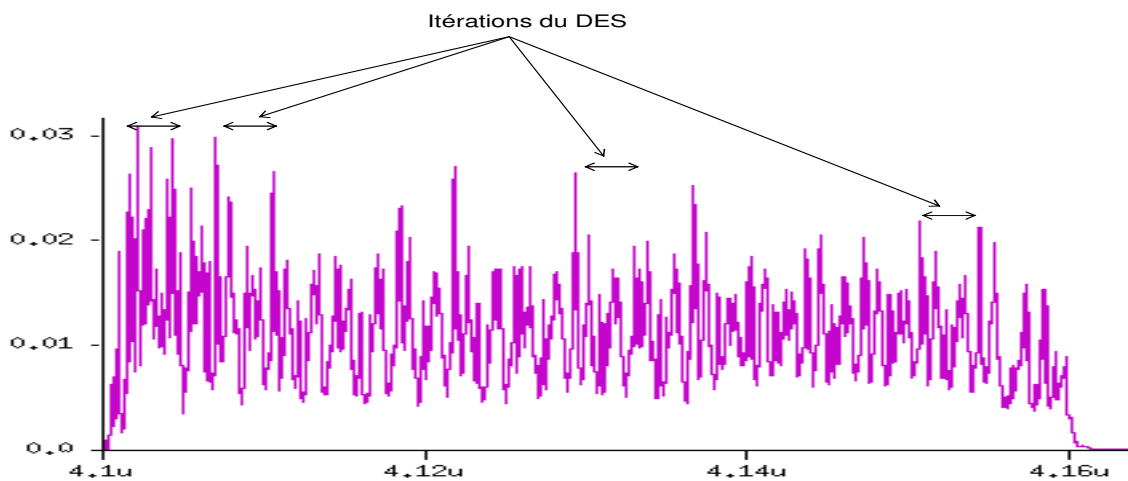
Tableau 7.2 Réorganisation de la fonction Sbox1 du DES

| Sorties | Entrées | | | | Ensembles |
|---------|--------------|--------------|--------------|--------------|----------------|
| 0 15 | 14/0 5/0 | 0/1 1/1 | 15/2 8/2 | 13/3 0/3 | E ₀ |
| 1 14 | 3/0 0/0 | 7/1 4/1 | 1/2 2/2 | 6/3 11/3 | E ₁ |
| 2 13 | 4/0 2/0 | 5/1 6/1 | 6/2 4/2 | 3/3 15/3 | E ₂ |
| 3 12 | 8/0 11/0 | 14/1 10/1 | 12/2 9/2 | 10/3 1/3 | E ₃ |
| 4 11 | 1/0 6/0 | 3/1 11/1 | 0/2 7/2 | 4/3 9/3 | E ₄ |
| 5 10 | 12/0 9/0 | 13/1 8/1 | 14/2 13/2 | 8/3 12/3 | E ₅ |
| 6 9 | 10/0 13/0 | 9/1 12/1 | 5/2 13/2 | 14/3 12/3 | E ₆ |
| 7 8 | 15/0 7/0 | 2/1 15/1 | 11/2 3/2 | 7/3 2/3 | E ₇ |

Numéro de la colonne
 C_x / L_x
 Numéro de la ligne

7.2.6 Validation par simulations électriques

La technologie utilisée pour implémenter le circuit est la *HCMOS9* ($0.13\mu\text{m}$) de chez *STMicroelectronics*. Toutes les simulations électriques réalisées dans cette partie ont été faites avec *NanoSim* à partir d'une netlist de portes. La figure 7.4 représente le profil de courant obtenu lors de l'exécution de l'algorithme du *DES*.

**Figure 7.4:** Profil de courant du cryptoprocresseur *DES*

Il faut noter que la simulation électrique est exempte de bruit et déterministe, et fournit donc des conditions quasi parfaites pour la *DPA*. Il s'agit donc d'une analyse favorable à l'attaque et le nombre de mesures à effectuer (N) est minimum.

Dans le but d'évaluer la pertinence de cette contre-mesure, la méthode par interversion de chemins des données a été uniquement implémentée sur la *SBOX1* et sur 4 bits du bloc *Registre L*. Les bits permutés dans *Registre L* correspondent aux bits qui sont combinés par la fonction *Xor32* avec les bits de sortie de la *SBOX1* (voir figure 7.3).

Afin de reproduire les effets des capacités parasites introduites lors des phases de back-end pendant les simulations, une dissymétrie est introduite entre les rails du bit attaqué (bit 4) en sortie de la *SBOX1*. La capacité de sortie du rail S_{40} est fixée à 32 fF (ce qui correspond à une différence de capacité entre les deux rails S_{40} et S_{41} de 32 fF). Cette valeur inclut la valeur de la capacité de routage, des capacités parasites et celle du court circuit. Elle a été estimée après une phase de pré placement et routage à l'aide de l'outil *Soc Encounter*. L'attaque différentielle est réalisée sur le bit 4 en sortie de la *SBOX1* avec 64 courbes de courant (64 messages d'entrées) sur la première itération de sorte que la fonction de sélection soit :

$$D(m_6 ; k_6) = SBOX1(m_6 \oplus k_6) \quad (7.9)$$

Dans un premier temps, nous avons réalisé une attaque sans activation de la contre mesure implémentée afin d'utiliser le résultat comme référence. Le résultat obtenu est représenté sur la figure 7.5. Lorsque la bonne clé est utilisée, le signal *DPA* fait apparaître des pics de consommation importants contrairement au cas où une mauvaise clé est utilisée.

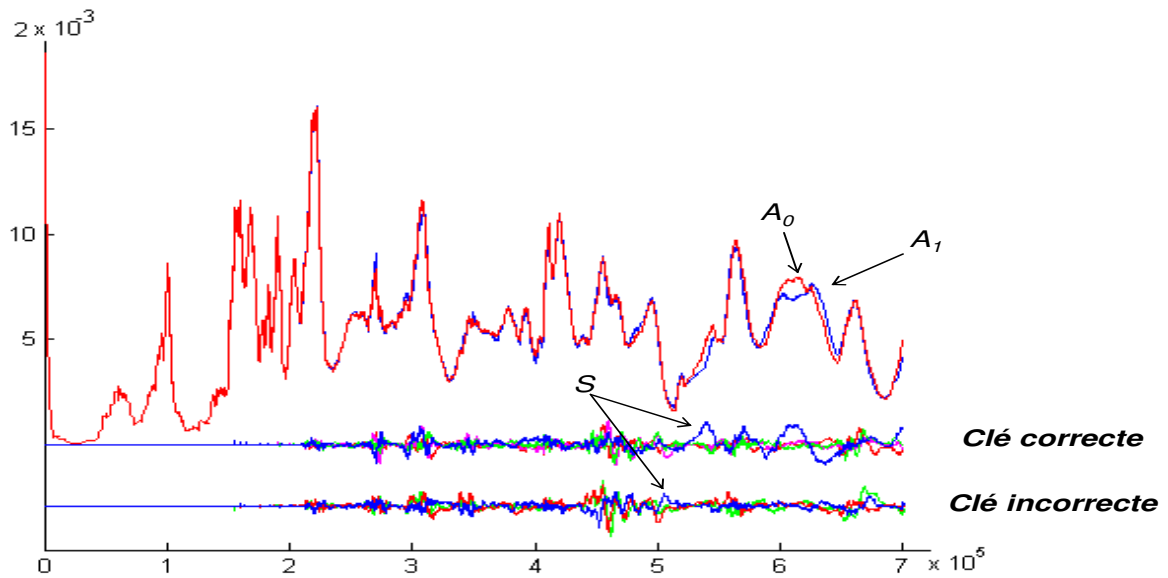


Figure 7.5: Signal *DPA* sur le bit 4 en sortie de la *SBOX1* sans activation de la contre-mesure.

Le résultat de l'attaque lorsque la contre-mesure est activée est représenté sur la figure 7.6. Le signal de *DPA* ne présente plus de pic important conformément à l'expression de l'équation (7.3).

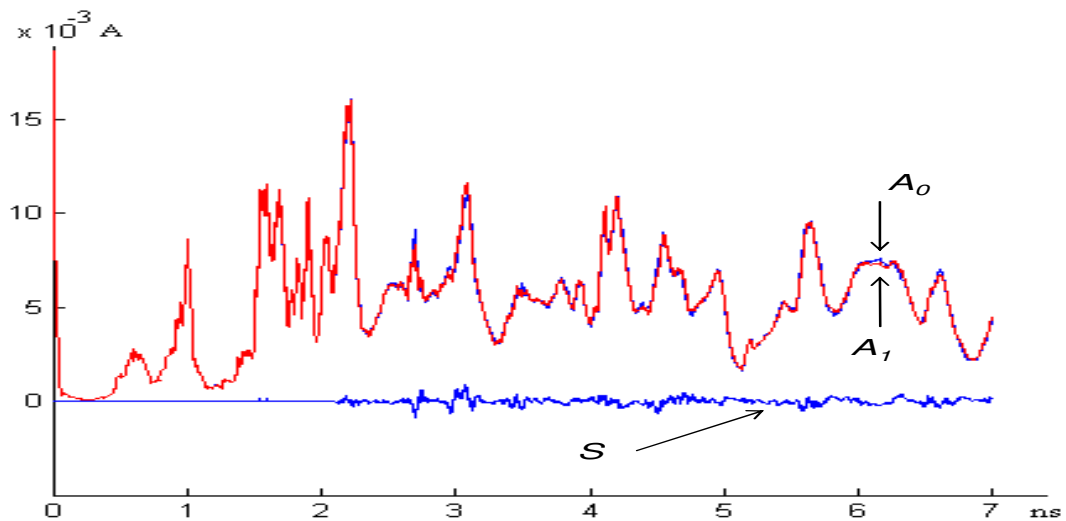


Figure 7.6: Signal *DPA* sur le bit 4 de *SBOX1* avec activation de la contre-mesure.

Les résultats théoriques et pratiques présentés dans cette partie démontrent la pertinence de la méthode par transpositions de chemins des données applicables dans des circuits *QDI* possédant des chemins de données équilibrés.

7.3 Méthode par décalage aléatoire des signaux d'acquittements

La seconde contre-mesure que nous proposons pour accroître la résistance des circuits *QDI* face à des attaques en puissance est basée sur un décalage aléatoire des signaux d'acquittements des blocs *QDI*. Cette méthode exploite les propriétés des signaux d'acquittement pour introduire des variations temporelles de manière à désynchroniser les temps d'exécutions des blocs. La pertinence de l'approche est formellement présentée et analysée. Les simulations électriques réalisées sur un cryptoprocresseur *DES* démontrent les performances de la méthode à travers une réduction considérable des amplitudes des pics de courant du signal *DPA*. Ces réductions de pics permettent non seulement de réduire les émissions électromagnétiques, mais également de complexifier les analyses différentielles en courant [Boue05b].

7.3.1 Les signaux d'acquittements

Le schéma de la figure 7.7 résume le principe de fonctionnement d'un circuit asynchrone *QDI*. Il représente l'implémentation de deux modules *QDI A* et *B* avec leurs éléments de mémorisation appelés Half-buffer. Le Half-buffer permet d'implémenter le protocole de communication 4 phases décrit dans le chapitre IV (cf. § 4.2.3). L'activation du signal d'acquittement du module *B* (*B_ack*) indique la disponibilité du module à recevoir des données. Lorsqu'une donnée est transférée du module *A* vers le module *B*, le module *B* calcule ses sorties et désactive son signal d'acquittement (*B_ack*) signalant ainsi, qu'il est prêt à recevoir une donnée invalide du module *A*. La réception de la donnée invalide par *B* met fin au transfert en réinitialisant le signal d'acquittement (*B_ack*).

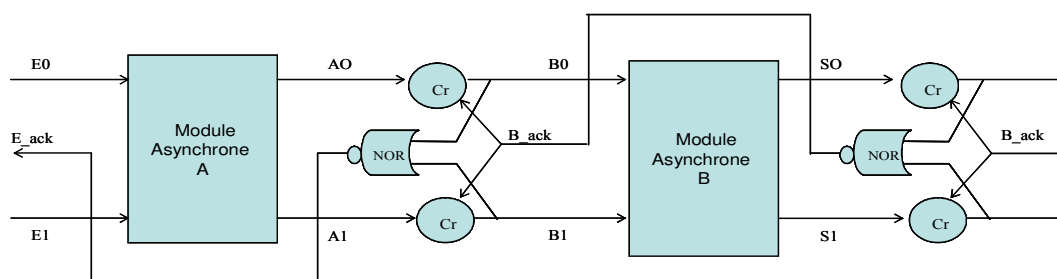


Figure 7.7: Half-buffer à un bit implémentant un protocole 4 phases entre deux modules *QDI*.

Dans ce mode de fonctionnement, le signal d'acquittement peut être considéré comme une horloge locale qui contrôle la mémorisation des données localement dans les Half-buffer. Il est important de rappeler que ces circuits sont fonctionnellement corrects sans aucune hypothèse temporelle, ils sont

uniquement sensibles aux événements. Ainsi, le signal d'acquiescement permet de contrôler l'activation des modules de calcul (temps d'occurrence des modules) dans une architecture **QDI**.

La technique proposée dans cette approche exploite cette propriété en insérant des délais aléatoires sur les signaux d'acquiescement des modules. L'idée est de désynchroniser les courbes de courant entre différents calculs afin de rendre les analyses de corrélations, entre les données manipulées et le courant consommé, pratiquement impossibles.

7.3.2 Décalage aléatoire des signaux d'acquiescement : Analyses formelles

On reprenant les analyses de corrélation définies dans le chapitre II (cf. § 2.4.3), on rappelle brièvement le principe de calcul du signal **DPA**.

Soient $q_{i0}(t_j)$ et $q_{i1}(t_j)$ les contributions en courant du bit d_i lorsqu'il est respectivement mis à 0 et 1 à l'instant t_j . Les valeurs $q_{i0}(t_j)$ et $q_{i1}(t_j)$ représentent la quantité de courant consommée par les chemins de données qui permettent le calcul de la valeur du bit d_i . Les moyennes en courant de chacun des deux ensembles formés par ces deux quantités sont calculées par les expressions suivantes:

$$\begin{aligned} M_0(t_j) = \varepsilon_0(t_j) &= \frac{1}{|n_0|} \sum_{i=1}^{n_0} q_{i0}(t_j) \\ M_1(t_j) = \varepsilon_1(t_j) &= \frac{1}{|n_1|} \sum_{i=1}^{n_1} q_{i1}(t_j) \end{aligned} \quad (7.10)$$

Avec n_0 et n_1 respectivement les nombres d'éléments des ensembles de répartitions \mathcal{S}_{d_i0} et \mathcal{S}_{d_i1} . En comparant ces deux moyennes à l'instant t_j correspondant au temps d'occurrence du bit attaqué, on déduit le signal **DPA** par la relation suivante :

$$\varepsilon_0(t_j) - \varepsilon_1(t_j) = \varepsilon(t_j) \quad (7.11)$$

Afin d'améliorer la qualité du signal **DPA** en terme d'amplitude, on utilise un nombre important de courbes (de messages d'entrées) N . Cela permet d'une part, d'augmenter le rapport signal sur bruit (cf. § 2.2) et d'autre part, de garantir l'excitation de tous les chemins de données qui permettent de faire commuter le bit attaqué. L'objectif est de prendre en considération toutes les quantités possibles $q_{ix}(t_j)$ représentant l'activité électrique du bit attaqué. Comme la probabilité d'activation de tous les chemins est

proportionnelle à N , alors plus la valeur de N est grande, plus la probabilité d'activation des différents chemins de données du bit attaqué est grande.

$$P(\omega) = \frac{N}{m} \quad (7.12)$$

m est généralement inconnu par le hacker et représente le nombre de chemins de données du dit bit.

La méthode que nous proposons consiste à générer du bruit temporel dans le composant en introduisant des délais sur les signaux d'acquittements des blocs du composant. Pour cela, on utilise un générateur aléatoire afin de rendre le contrôle des délais placés sur les signaux d'acquittements aléatoires comme illustré sur la figure 7.8.

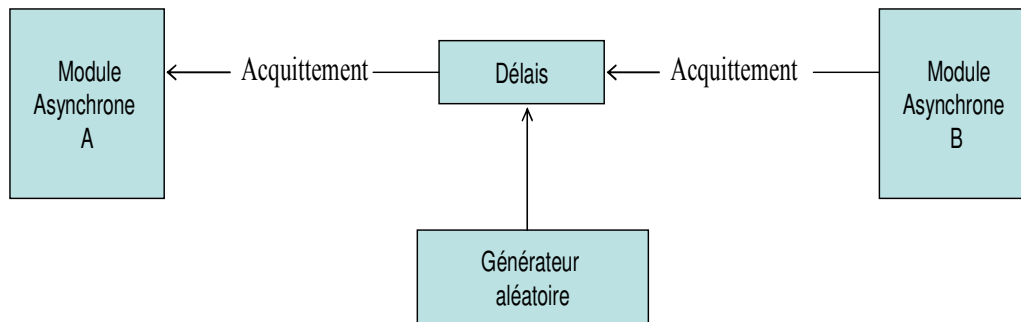


Figure 7.8: Implémentation du décalage aléatoire sur un signal d'acquittement

Soit n le nombre possible de délais que l'on peut implémenter dans une architecture donnée. n dépend du nombre de signaux d'acquittements m disponibles dans une architecture et du nombre de délais del_i implémenté par signal d'acquittement. La valeur de n est déterminée par la relation suivante :

$$n = \prod_{i=1}^m del_i \quad (7.13)$$

Si le signal d'acquittement du bit attaqué peut être décalé aléatoirement à n instant différents, alors sa valeur sera calculée à n temps différents t_j . Soit N/n le nombre de fois que le bit attaqué est calculé au temps t_j et soit $N/2n$ le nombre de fois que les quantités $q_{i0}(t_j)$ et $q_{i1}(t_j)$, de ce bit, contribuent respectivement dans les ensembles S_{di0} et S_{di1} . En considérant une répartition égale des entrées dans les ensembles de répartition S_{di0} et S_{di1} ($n_0=n_1=N/2$), les moyennes de courant peuvent être exprimées par :

$$\begin{aligned}\varepsilon_0(t_j) &= \frac{1}{n_0} \sum_{i=1}^{N/2n} \left(\sum_{j=1}^n q_{i0}(t_j) \right) \\ \varepsilon_1(t_j) &= \frac{1}{n_1} \sum_{i=1}^{N/2n} \left(\sum_{j=1}^n q_{i1}(t_j) \right)\end{aligned}\quad (7.14)$$

Le signal de **DPA** est obtenu par la relation suivante :

$$\begin{aligned}\varepsilon(t) &= (\varepsilon_0(t_1) - \varepsilon_1(t_1)) + \dots + (\varepsilon_0(t_n) - \varepsilon_1(t_n)) \\ &\quad \text{avec} \\ \varepsilon_0(t_n) &= \frac{1}{n_0} \sum_{i=1}^{N/2n} q_{i0}(t_n) \\ \varepsilon_1(t_n) &= \frac{1}{n_1} \sum_{i=1}^{N/2n} q_{i1}(t_n)\end{aligned}\quad (7.15)$$

Cette équation (7.15) met en évidence le fait que, au lieu d'avoir une seule quantité $\varepsilon_x(t_j)$, nous avons n quantités $\varepsilon_x(t_n)$ qui correspondent aux n temps d'occurrence du bit attaqué. Les valeurs de $\varepsilon_x(t_j)$ ainsi obtenues sont divisées par un facteur n comme le démontre les simplifications suivantes :

$$\begin{aligned}\varepsilon_x(t) &= \frac{1}{n_x} (q_{xj}(t_j) + \dots + q_{xn}(t_j)) \cong \frac{q_{xj}(t_j)}{n} \\ &\quad \text{avec} \\ q_{xj}(t_j) &\cong \dots \cong q_{xn}(t_j)\end{aligned}\quad (7.16)$$

Malgré le fait que l'on augmente le nombre de points significatifs par n , on réduit d'un facteur n les amplitudes des pics **DPA**, ce qui favorise la diminution du rapport signal sur bruit. Cette méthode peut être utilisée également pour réduire les émissions électromagnétiques des circuits **QDI** pour cibler des applications nécessitant des faibles rayonnements électromagnétiques [Boue04b].

7.3.3 Discussion

Considérons par exemple une réalisation de l'attaque **DPA** avec 1000 courbes moyennés ($N=1000$ messages). Dans une approche standard dans laquelle le bit attaqué est calculé à un instant donné, on

obtient dans chacun des ensembles de répartition une moyenne sur 500 courbes. En faisant un décalage aléatoire des signaux d'acquittements avec par exemple $n=16$, on obtient 16 points (en terme de temps) d'occurrences différentes où le bit attaqué est calculé. Il existe 62 quantités $q_{ix}(t_j)$ (N/n courbes) où ce bit est calculé à l'instant t_j et chaque ensemble contient 31 quantités $q_{ix}(t_j)$ en moyenne. Lorsque les moyennes en courant de chaque ensemble de répartition sont calculées, les quantités $q_{ix}(t_j)$ sont 16 fois plus petites que dans une approche standard. Par conséquent, la contribution de la quantité $q_{ix}(t_j)$ dans les courbes de courant sont réduites d'un facteur 16.

La réussite d'une attaque dans de telles conditions, nécessite d'augmenter le nombre de courbe d'acquisition (N) ou d'appliquer une fonction d'intercorrélacion qui est exactement l'un des objectifs à atteindre en terme de complexité de l'attaque. En effet, l'intercorrélacion est un moyen efficace de re-synchronisation des courbes. Mais pour être performant, on doit dans un premier temps, identifier une quantité $q_{ix}(t_j)$ de manière à l'utiliser comme référence. Une fois la référence identifiée, des re-synchronisations peuvent être effectuées par la suite pour chacune des N courbes, sachant que les calculs d'intercorrélacion ne peuvent qu'être réalisées sur des courbes de courants instantanées qui sont naturellement très bruitées.

Afin d'accroître la complexité de l'attaque, la valeur de n peut être augmentée en manipulant les valeurs des délais del_i et des acquittements m .

- la valeur de m dépend de l'architecture. Sa valeur peut être augmentée en éclatant les signaux d'acquittements dans une architecture. Chaque bit de chaque valeur intermédiaire peut être acquitté individuellement. Cette technique, permet également de réduire la latence du chemin de données.
- Les valeurs des délais (del_i), dépendent du temps spécifié dans le cahier des charges pour exécuter la fonction à implémenter. Elles sont limitées par le temps maximum que doit prendre la fonction pour effectuer son calcul.

On applique sur l'architecture du **DES** la méthode par décalage aléatoire des signaux d'acquittement, et notamment sur les 5 blocs en gras représentés sur la figure 7.9. Chaque bloc est implémenté avec un signal d'acquittement et chacun d'eux possède 4 différentes valeurs de délais, ce qui correspond à 1024 combinaisons possibles de délais ($n=1024$). En terme de résistance à la **DPA**, cela équivaut à une réduction des amplitudes de pics de courant d'un facteur 1024.

Les décalages peuvent être étendus sur tous les signaux d’acquiescement des différents blocs de cette architecture (14 blocs principaux). La boucle de chiffrement de l’architecture a un chemin de données de 32 bits. Ainsi, chaque bloc possède au moins 32 bits en entrées, excepté le bloc de contrôle. En éclatant ces acquiescements qui sont uniques par bloc de sorte à acquiescer chaque bit séparément, on obtient 32 signaux d’acquiescement par bloc, ce qui dans une telle architecture correspondant à un total approximatif de 400 signaux d’acquiescements. Toutefois, le nombre de délais à introduire dans une architecture et notamment le nombre de combinaisons possible doit tenir compte des contraintes temporelles du système (l’environnement de fonctionnement du système).

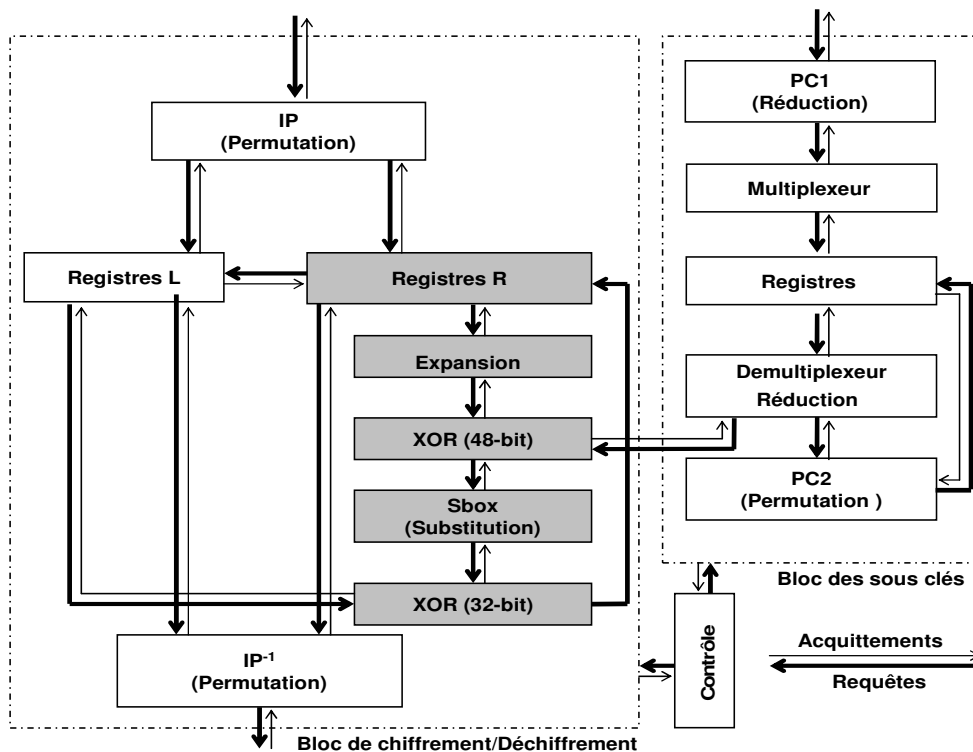


Figure 7.9: Architecture du *DES* et gestion des signaux d’acquiescements

Ainsi, les signaux d’acquiescements des circuits en logique asynchrone quasi insensible aux délais peuvent être exploités pour générer du bruit temporel afin d’accroître leur résistance contre les attaques en puissance. Cependant, cette méthode permet également de réduire considérablement les émissions électromagnétiques des circuits faisant de la logique *QDI* une alternative intéressante dans le domaine de la *CEM*. Nous avons dans le chapitre IV (cf. § 4.4.2) comparé en terme d’émissions électromagnétiques le cryptoprocresseur *DES* synchrone et le cryptoprocresseur *DES* asynchrone. Malgré l’avantage affiché par le

circuit asynchrone, ce circuit n'implémente par une architecture optimisée pour les faibles émissions électromagnétiques. En développant cette contre-mesure, nous avons également déterminé et mis en place des architectures plus adéquates pour adresser le problème des émissions électromagnétiques dans les circuits intégrés.

7.3.4 Réduction des émissions électromagnétiques

Les études que nous avons menées pour la réduction des émissions électromagnétiques ne sont pas entièrement focalisées sur la protection des circuits contre les attaques électromagnétiques, mais plus généralement sur des applications nécessitant de très faibles émissions électromagnétiques. Les analyses réalisées sur les signaux d'acquittements des circuits *QDI* nous ont permis de déduire différents types d'architectures *QDI* dédiés aux faibles émissions électromagnétiques.

En effet, le retard introduit sur les acquittements permet non seulement d'augmenter la latence des blocs mais permet également de réduire le nombre de processus concurrents. Ainsi en retardant et en contrôlant certains signaux d'acquittements dans une architecture on peut forcer une exécution quasi séquentielle de ses fonctions. Cette absence de parallélisme dans l'exécution permet de réduire de plus de 50% les pics de courant dans des architectures *QDI* itératives comme c'est le cas avec les algorithmes cryptographiques [Boue05a].

En considérant l'architecture du *DES* décrite sur la figure 7.9, nous avons évalué trois architectures différentes en terme de réduction des pics de courant.

- La première architecture utilise des signaux d'acquittements groupés (version *AG*). Cela signifie qu'il y a par bloc un seul signal d'acquittement. Le courant consommé par ce type d'implémentation n'est pas négligeable comparé aux autres types de circuits asynchrones. En effet, ces signaux d'acquittement peuvent être considérés comme des signaux d'horloge locaux autorisant le calcul en fonction des événements des signaux d'acquittement des blocs. Dans ce type d'architecture, les possibilités de concurrence entre les différents blocs, activés chacun par un signal, sont accrues, ce qui entraîne une consommation significative.

- La seconde architecture utilise des signaux d'acquittements séparés (version *AS*) entre les blocs. Dans cette architecture on effectue des acquittements bit par bit. Le nombre d'acquittement par bloc dépend du nombre des bits d'entrées et de sorties des blocs à acquitter. Contrairement à la première approche, ici, on réduit la dépendance entre les signaux d'acquittements et le bloc. Chaque signal d'acquittement dépend du chemin de données du bit qu'il acquitte. Ce qui diminue la latence directe du bloc et permet de lisser le profil de courant en distribuant la consommation électrique dans le temps.
- La dernière architecture utilise des signaux d'acquittement séparés et retardés dans le temps (version *ASRT*). L'idée ici est d'éviter des calculs en parallèle des différentes fonctions de l'architecture et tout particulièrement de celles présentant une forte activité électrique. L'activité du courant est répartie dans le temps en utilisant des délais sur les signaux d'acquittement. Ces délais sont contrôlés logiquement ou adaptés statiquement sur les circuits en suivant le flot de conception décrit dans [Pan02].

7.3.4.1 Résultats des analyses : simulations électriques

La figure 7.10 présente les profils de courant des trois architectures du *DES* définies ci-dessus. Comme on s'y attendait, la version *AS* est près de deux fois plus rapide que la version *AG* avec un temps de chiffrement de 45 ns pour la version *AS* contre 80 ns pour la version *AG*. Elle permet (la version *AS*) de réduire les amplitudes des pics de courant de 40% comparé à *AG*. L'utilisation des délais dans la version *AS* permet d'obtenir une réduction de près de 60% entre la version *AG* et celle de *ASRT*. Cependant cette version présente plus de pics de courant (nombre de pics de courant).

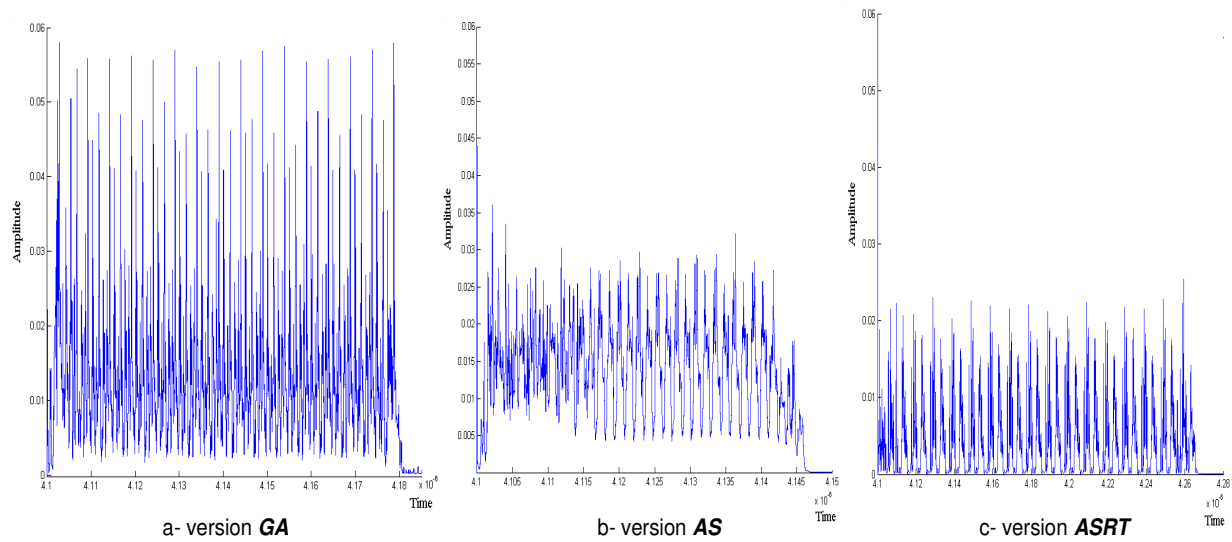


Figure 7.10: Profils des courants des différentes architectures

En effet, comme on le constate sur les spectres en courant de la figure 7.11, la version *ASRT* exhibe plus de pics de courant au delà de 6 GHz (figure 7.11-c). Le fait de supprimer toutes concurrences entre les différentes fonctions de l'algorithme revient à forcer leur exécution de façon séquentielle ce qui permet de dévoiler avec plus de détails les activités électriques de chacune des fonctions implémentées. Mais contrairement aux versions (*AG* et *AS*) les amplitudes de ces pics sont considérablement réduites (d'un facteur de 2,75 entre la version *AG* et la version *ASRT*).

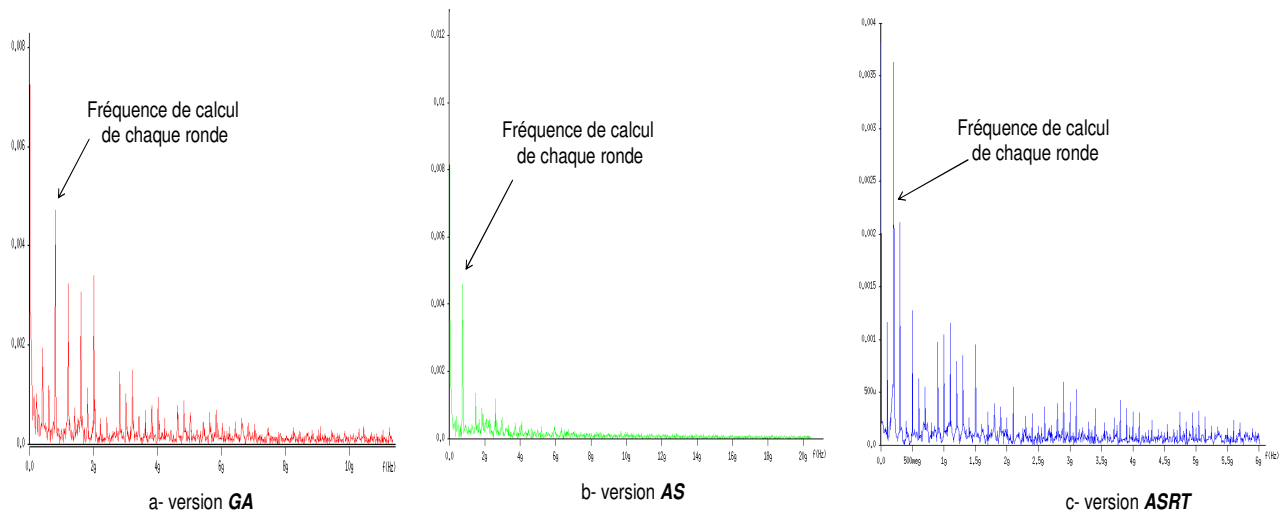


Figure 7.11: Spectres en courant des trois architectures

Ces analyses démontrent les avantages en termes de performance et de consommation des architectures *QDI* utilisant des signaux d'acquittement éclatés. L'application des méthodes de répartitions de l'activité électrique développées dans [Pany04] permet de cibler des applications à très faibles émissions électromagnétiques. Cependant, sur le plan de la sécurité contre les attaques par analyse en courant et notamment contre les attaques électromagnétiques, de telles implémentations restent vulnérables. Mieux encore, les attaques seraient même facilitées avec les détails offerts sur les activités électriques des différentes fonctions de l'architecture (figure 7.10-c). Toutefois, ces méthodes peuvent être combinées aux techniques de décalage aléatoire des signaux d'acquittement.

7.3.5 Etude de cas : cryptoprocasseur *DES* en logique asynchrone *QDI*

Dans cette partie, nous allons présenter les résultats d'implémentation du décalage temporel aléatoire des signaux d'acquittement dans une architecture du *DES* réalisée en logique asynchrone *QDI*. Le flot de simulation utilisé est identique au flot utilisé et décrit au paragraphe 7.2.6.

L'architecture utilise un signal d'acquittement par bloc, et seul l'acquittement du bloc *SBOXI* a été décalé aléatoirement à l'aide de 8 délais différents ($n=8$). Les attaques sont donc réalisées en sorties de la *SBOXI* pendant la première itération de calcul du *DES* et ceci, sur 64 courbes non moyennés ($N=64$ messages qui représentent toutes les valeurs possibles de la *SBOXI*). La figure 7.12 présente le profil de courant des premières itérations du *DES* lorsque le décalage est désactivé et lorsqu'il est activé. Lorsqu'un délai de 13ns est utilisé, le temps nécessaire pour effectuer une itération (figure 7.14-a) correspondant au temps nécessaire pour effectuer 3 itérations (figure 7.14-b) lorsque le délai est de 0 ns, ce qui correspondant à une multiplication du temps de calcul par un facteur 3. Afin d'optimiser aussi bien le temps d'exécution et la surface due à l'implémentation des blocs de délais, le pas ou l'écart entre les délais peuvent être réduit au maximum. Dans une telle technologie (*CMOS 0.13 μ m*) on peut implémenter des délais de l'ordre de quelques picosecondes.



Figure 7.12: Profils de courant des premières itérations du *DES* avec et sans activation du décalage des signaux d'acquittements

Comme la *SBOX1* possède 4 bits en sorties codés en double rails, nous avons potentiellement 8 chemins de données entre les sorties et les entrées (ici nous ne tenons pas compte des chemins d'exécutions) qui permettent de calculer les 8 quantités $q_{ix}(t_j)$ de chaque rail. Considérons par exemple le bit 4 en sortie de la *SBOX1* dont les rails présentent toujours une dissymétrie de 32 femto farads. Comparé à une approche standard (sans activation du décalage) dans laquelle les quantités $q_{40}(t_j)$ et $q_{41}(t_j)$ seraient calculées respectivement 32 fois chacune aux instants t_j , l'utilisation du décalage permet de les calculer 4 fois à des instants t_j différents, si bien que leur contribution dans la moyenne de courant dans chaque ensemble S_{40} et S_{41} est réduite par 8. La figure 7.13 présente ces moyennes de courant utilisées pour calculer le signal de *DPA*.

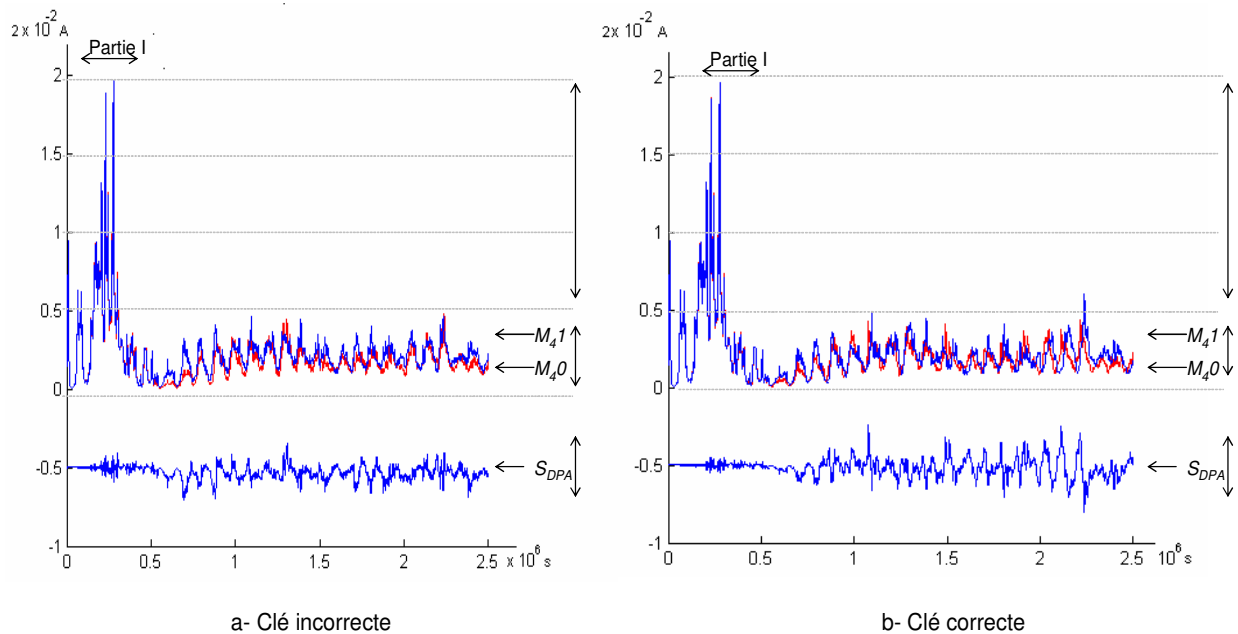


Figure 7.13: Signatures électrique lors de l'attaque du bit 4 en sortie de la *SBOX1* sur la première itération du *DES*.

La première partie de ces courbes présente les premières opérations réalisées avant la prise en compte des effets des retards insérés sur le signal d'acquiescement de la *SBOX1* (voir architecture figure 7.9). En effet, la permutation initiale (*IP*), les registres (*R* et *L*), l'expansion et la fonction *Xor48* (ou exclusif sur 48 bits) exécutées au début de la première itération ne sont pas affectés par le décalage introduit dans la *SBOX1*. Cela explique les amplitudes de courant élevés observées au début des courbes et qui sont par la suite réduites, illustrant ainsi les effets attendus du décalage aléatoire lors des calculs des différentes moyennes. Evidemment, on peut supprimer ce phénomène en commençant l'implémentation du décalage sur le bloc de permutation initiale *IP*.

La réalisation de l'attaque *DPA* dans de telles conditions ne nous a pas permis de sortir des pics de *DPA* suffisamment importants pour réussir l'attaque. Pour cela il faudrait augmenter le nombre de courbe ce qui est l'objectif à atteindre. Par conséquent, on montre par cette approche qu'il est possible de complexifier les analyses de manière à rendre l'implémentation de l'attaque quasi impossible en augmentant les combinaisons des délais.

7.4 Conclusion

Nous avons dans ce chapitre présenté des méthodes de protection des circuits asynchrones *QDI* afin d'augmenter leur résistance intrinsèque face aux attaques par analyses différentielles de courant. Ces nouvelles contre-mesures exploitent les propriétés de la logique *QDI* pour masquer toutes les dissymétries électriques introduites lors des phases de placement-routage pouvant permettre de réussir des attaques.

La première méthode est basée sur une interversion aléatoire des chemins d'exécutions des blocs d'une architecture. Elle exploite les symétries logiques existant dans les blocs des circuits *QDI*. Le choix aléatoire du chemin de données parmi tous les chemins possible du bloc pendant chaque itération ou calcul permet d'obtenir une moyenne identique de courant quelques soient les données manipulées. Des analyses formelles sur la méthode développée ont été proposées ainsi que des résultats concluants d'implémentation sur un circuit *DES*.

La Seconde méthode présentée permet d'introduire du bruit temporel dans le circuit afin de désynchroniser aléatoirement les temps de calcul de chaque bloc de l'architecture. Les analyses théoriques et les résultats de simulations nous ont permis de démontrer toute l'efficacité de cette approche qui permet d'amplifier la complexité des analyses. Ces analyses ont également permis de montrer le potentiel de la logique *QDI* à réduire les émissions électromagnétiques des circuits faisant de cette logique une alternative à explorer.

CONCLUSION ET PERSPECTIVES

De nos jours, on ne saurait concevoir un circuit intégré dédié à des applications cryptographiques sans implémenter des contre-mesures contre des attaques par canaux cachés. Les nouvelles variantes de ces attaques avec notamment la *DPA* de second d'ordre, les attaques par collisions et les attaques par corrélations obligent un développement constant de nouveaux systèmes de protection.

Nous avons montré, dans ces travaux de recherche, le potentiel offert par l'alternative asynchrone pour faire face aux attaques par analyse de courant comparé aux circuits implémentant de la logique synchrone. Après avoir introduit les différents types d'attaques par canaux cachés, nous avons analysé et détaillé l'origine des fuites d'informations dans les circuits *CMOS* implémentés en logique synchrone. Ces études nous ont permis par la suite, d'analyser les propriétés des circuits asynchrones et tout particulièrement des circuits quasi insensibles aux délais dans le but d'examiner théoriquement leur potentiel. Les prototypes des circuits réalisés pour évaluer les propriétés intrinsèques des circuits *QDI* ont confirmées d'une part, les prédispositions de cette logique à améliorer la résistance des circuits contre les attaques par analyse de courant et ont d'autre part, permis de mettre en évidence quelques fuites résiduelles d'informations qui permet de réussir des attaques. Des analyses formelles effectuées sur la représentation graphique des circuits asynchrones *QDI*, ont permis d'identifier l'origine de ces fuites d'informations qui sont essentiellement dues aux dissymétries introduites dans les chemins de données lors des phases de back-end. En partant de ce constat, nous avons proposé une méthode d'évaluation de la sensibilité des circuits aux attaques différentielles pendant les différentes phases de conception, notamment au niveau logique et au niveau électrique. Cette évaluation, que nous avons voulu automatisable, est faite en fonction des critères de sensibilité qui permettent d'agir très tôt dans les phases de conception. Afin d'améliorer le niveau de résistance intrinsèque des circuits *QDI*, nous avons développé deux contre-mesures qui consistent à décaler aléatoirement dans le temps les instants de calcul des blocs sensibles d'une architecture ou à intervertir aléatoirement les chemins de données des blocs sensibles présentant des structures redondantes et symétriques. Chacune de ces méthodes a été analysée et testée par simulations électriques. Les résultats obtenus ont montré la pertinence de ces deux approches.

Les nouvelles contre-mesures que nous avons développées pour accroître la résistance intrinsèque des circuits **QDI** et la spécification de méthode de conception de circuits sécurisés que nous avons mise en oeuvre sont en cours d'intégration dans l'environnement **TAST**. Le but à atteindre étant d'offrir au concepteur la possibilité de définir la sécurité comme une contrainte et un objectif pour la synthèse, afin de concevoir des circuits résistants aux attaques non intrusives avec notamment les attaques par analyses différentielles de courant et les attaques par injections de fautes.

Aujourd'hui, l'outil permet de synthétiser des circuits **QDI** dont les chemins de données sont équilibrés. Ces circuits sont décrits sous la forme de netlist **VHDL** qui peuvent être simulées. Nous avons entrepris, dans le groupe, la conception d'un ensemble de cellules de bibliothèques dites asynchrones afin d'améliorer la surface et les performances des circuits **QDI**. Cette bibliothèque est en cours de finalisation et sera enrichie des cellules spécifiques à la sécurité. De telles cellules permettront d'avoir un courant constant quel que soient les données manipulées. Ces travaux sont menés en collaboration avec le **LIRMM (Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier)**.

Concernant les attaques par fautes, des travaux de recherche sont actuellement en cours dans le groupe sur l'analyse du comportement des circuits quasi insensibles aux délais en présence de fautes [Monn05a]. Ces travaux permettront d'évaluer les capacités intrinsèques des circuits **QDI** à résister aux attaques par injections de fautes. Le principe est de pouvoir exploiter les structures redondantes de la logique **QDI** afin de durcir les circuits contre les fautes sans pour autant perturber les contre-mesures implémentées contre les attaques par analyse de courant.

Ces travaux, ont également ouvert des perspectives intéressantes dans le domaine de la **CEM** et dans le domaine de la conception des générateurs de nombres aléatoires. En effet, les solutions de protection que nous avons proposé dans ces travaux sont basées sur l'utilisation d'un générateur de nombres aléatoires. Nous allons, dans la suite de ces travaux, étudier les possibilités de conception des générateurs aléatoires asynchrones en exploitant les propriétés d'insensibilité aux délais et de concurrence de la logique **QDI**. En terme de **CEM**, nous avons montré que les circuits **QDI** sont particulièrement intéressants pour cibler des applications à très faibles émissions électromagnétiques, cependant aucune analyse de susceptibilité sur ce type de logique n'a été effectuée. La question est de savoir si l'absence d'aléa nécessaire au fonctionnement de ce type de logique est un atout ou un inconvénient en terme de susceptibilité aux radiations électromagnétiques.

Enfin, un prototype de cryptoprocasseur *DES* est en cours de conception afin d'évaluer sur silicium les différentes contre-mesures proposées dans ces travaux de thèse.

BIBLIOGRAPHIE

- [Akka01] M. L. Akkar and C. Giraud, "An Implementation of DES and AES Secure Against Some Attacks," presented at Cryptographic Hardware and Embedded Systems (CHES2001), pp. 309-318, LNCS 2162, Paris, France, May 14-16, 2001.
- [Auve00] D. Auvergne, J. M. Daga, and M. Rezzoug, "Signal Transition Time Effect on CMOS Delay Evaluation," *IEEE Transactions on Circuits and Systems*, vol. 47 N°9, pp. 1362-1369, 2000.
- [Bao97] F. Bao, R. H. Deng, Y. Han, A. B. Jeng, A. D. Narasimhalu, and T. H. Ngair, "Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults," presented at 5th International Workshop on Security Protocols. Springer-Verlag, vol. 1361, pp. 115-124, Paris, France, 1997.
- [Beva02] R. Bevan and E. Knudsen, "Ways to Enhance Differential Power Analysis," presented at ICISC 2002, vol. 2567, pp. 327-342, 2002.
- [Biha97] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," presented at 17th annual International Cryptology Conference on Advances in Cryptology CRYPTO'97, vol. 1294, pp. 513-525, California, USA, 1997.
- [Biha90] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like cryptosystems (Extended abstract)," presented at Advances in Cryptology Crypto'90, Springer-Verlag 1991 ed, vol. 537 of LNCS, pp. 2-21, 1990.
- [Biha99] E. Biham and A. Shamir, "Power Analysis of the Key Scheduling of the AES Candidates," presented at Second AES Candidate Conference (AES2), Rome, Italy, March 22-23, 1999.
- [Bone97] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," presented at EUROCRYPT'97, vol. 1233, pp. 37-51, Berlin, 1997.
- [Breg04] V. Bregier, B. Folco, L. Fesquet, and M. Renaudin, "Modeling and Synthesis of multi-rail multi-protocol QDI circuits," presented at International Workshop on Logic Synthesis, June, 2004.
- [Brie04] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," presented at Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), vol. 3156 of LNCS, pp. 16-29, Edinburgh, Scotland, 2004.
- [Cogi03] O. Cogis and C. Robert, *Théorie des Graphes : Problèmes, Théorèmes, Algorithmes*: Vuibert, 2003. ISBN 2 7117 5321 2.
- [Copp94] D. Coppersmith, "The Data Encryption Standard (DES) and its Strength Against Attacks," Technical report rc 186131994 IBM Thomas J. Watson Research Center December 1994

- [Coro00] J. S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis," presented at Cryptographic Hardware and Embedded Systems (CHES2000), pp. 231-237, LNCS 1965, Worcester, MA, USA, August 17-18, 2000.
- [Coro00] J. S. Coron, P. Kocher, and D. Naccache, "Statistic and Secret Leakage," presented at Financial Cryptography (FC2000). Lecture Notes in Computer Science, vol. 1962, pp. 157-173, 2000.
- [Dhem 98] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestre, J. J. Quisquater, and J. L. Willems, "A Practical Implementation of the Timing Attack," presented at The Third International Conference on Smart Card Research and Applications (CARDIS'98), vol. 1820, pp. 167-182, Louvain la Neuve, Belgium, September 14-16, 1998.
- [Din02] A. V. DinhDuc, J. B. Rigaud, A. Rezzag, A. Siriani, J. Fragoso, L. Fesquet, and M. Renaudin, "TAST CAD Tools," presented at 2nd Asynchronous Circuit Design Workshop of the European Commission's Fifth Framework Programme (ACID'02), Munich, Germany, January, 28-29, 2002.
- [Dire89] "Directive du Conseil Européennes sur la Compatibilité électromagnétique," in *Journal Officiel des Communautés européennes*, vol. 139, 1989, pp. 19-26.
- [Eck85] W. v. Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk ?," *Elsevier Science Publishers*, 1985.
- [Essa02] M. Essalhiene, L. Fesquet, and M. Renaudin, "Dynamic Voltage Scheduling for Real Time Asynchronous Systems," presented at 12 International Workshop on power and timing modeling, optimization and simulation (PATMOS02), Sevilla, Spain, 11-13, September, 2002.
- [FIPS186] NIST, "DIGITAL SIGNATURE STANDARD (DSS)," National Institut of Standard and Technology FIPS PUB 186-2, January 2000
- [FIPS197] NIST, "Advanced Encryption Standard (AES)," National Institut of Standard and Technology, FIPS PUBS 197, November 2001
- [FIPS46a] NIST, "Data Encryption Standard (DES)," National Institut of Standard and Technology FIPS PUB46-2, December 1993
- [FIPS46b] NIST, "Data Encryption Standard (DES and Triple-DES)," National Institut of Standard and Technology FIPS PUB46-3, October 1999
- [Four03] J. J. A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor, "Security Evaluation of Asynchrone Circuits," presented at Cryptographic Hardware and Embedded Systems (CHES2004), LNCS 2779, Cologne, Germany, September, 7-10, 2003.
- [Gand01] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results," presented at Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), pp. 251-261, Paris, France, 2001.
- [GEM01] GEMPLUS, "Single Power Analysis," presented at GEMPLUS, Workshop on Cryptography and Security, March 13, 2001. http://www.cs.uku.fi/kurssit/ads/09_20-_20SPA.pdf.

- [Goubin99] L. Goubin and J. Patarin, "DES and Differential Power Analysis: The Duplication Method," presented at Cryptographic Hardware and Embedded Systems (CHES1999), pp. 158-172, Worcester, MA, USA, 1999.
- [Gueu98] P. Gueulle, *Carte à Puce: Initiation et Applications*, vol. 2, ETSF ed. Paris: Dunod, 1998. 2-10-023997. pp. 16-18.
- [Guil04] S. Guilley, P. Hoogvorst, and R. Pacalet, "Differential Power Analysis Model and some Results," presented at WCC/CARDIS 04, pp. 127-142, Toulouse, France, Aug, 2004.
- [Hara68] F. Haray, R. Z. Norman, and D. Cartwright, *Introduction a la Théorie des graphes orientés: modèles structuraux*: Dunod, 1968.
- [Hevi99] A. Hevia and M. Kiwi, "Strength of Two Data Ebcryption Standard Implementation Under Timing Attacks," *ACM Transaction on Information and System Security (TISSEC)*, vol. 2, pp. 416-437, 1999.
- [Jeon97] Y. J. Jeong and W. P. Burleson, "VLSI array Algorithms and Architectures for RSA modular mutiplication.," *IEEE Transactions on Very Large Scale of Integration (VLSI) Systems*, vol. 5(2), pp. 211-217, 1997.
- [Kak96] C. K. Koç, T. Acar, and B. S. Kaliski, "Analyzing and Comparing Montgomery Multiplication Algorithms," *IEEE Micro*, vol. 16(6), pp. 26-33, 1996.
- [Koch96] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Others Systems," presented at 16th International Cryptology Conference on Advances in Cryptology (CRYPTO'96), vol. 1109, pp. 104-113, Santa Barbara, California, USA, August 18-22, 1996.
- [Koche99] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," presented at the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 99), Lecture Notes In Computer Sciences ed. Springer-Verlag, vol. 1666, pp. 388-397, Santa Barbara, California, USA, August 15-19, 1999.
- [Koeu99] F. Koeune and J. J. Quisquater, "Timing Attack against Rijndael," UCL Crypto Group, Louvain la Neuve June 10 1999. <http://web.engr.oregonstate.edu/~aciicmez/osutass/data/Koeune99.dpf>.
- [Kömm99] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," presented at USENIX Workshop on Smartcard Technology (Smartcard 99), pp. 9-19, Chicago, Illinois, USA,, May, 10-11, 1999.
- [Lang94] S. K. Langford and M. E. Hellm, "Cryptanalysis of DES," presented at RSA Data Security Conference, pp. 353-365, Berlin, Jan 12-14, 1994.
- [Ledi04] H. Ledig, F. Muller, and F. Valette, "Enhancing Collision Attacks," presented at CHES2004, vol. 3156 of LNCS, pp. 176-190, 2004.
- [lens97] A. K. Lenstra, "Memo on RSA signature generation in the presence of faults," Sept 28, 1996. Available from the author arjen.lenstra@citicorp.com.

- [Mahe97] D. P. Maher, "Fault Induction Attacks, Tamper Resistance, and Hostile Reverse Engineering in Perspective," presented at First International Conference on Financial Cryptography, vol. 1318, pp. 109-122, Anguilla, British West Indies, 1997, 1997.
- [Man02] S. Mangard, "A Single Power Analysis (SPA) Attack on Implementations of the AES Expansion," presented at Information Security and Cryptology (ICISC02), 6th International Conference, pp. 343-358, Seoul, Korea, November 28-29, 2002.
- [Mart90] A. J. Martin, *Programming in VLSI: from communicating processes to delay-insensitive circuits*, in C.A.R. Hoare, ed, *Developments in Concurrency and Communication*, UT Year of Programming Series: Addison-Wesley, 1990.
- [Mats94] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," presented at Eurocrypt'93. Springer, vol. 765 of LNCS, pp. 386-397, May 23-27, 1993.
- [Mau01] P. Maurine, "Modélisation et Optimisation des Performances de la logique Statique en Technologie CMOS Submicronique Profond," in *Electronique, Optronique et Systèmes*. Montpellier: Université Montpellier II, 2001.
- [Mess00] T. S. Messerges, "Securing the AES Finalist Against Power Analysis Attacks (FSE2000)," presented at Fast Software Encryption Workshop, New York, USA, April, 2000.
- [Mess00a] T. S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," presented at Cryptographic Hardware and Embedded Systems (CHES2000), pp. 238-251, LNCS1965, Worcester, MA, USA, August 17-18, 2000.
- [Mess02] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining Smart-Card Security under the Threat of Power Analysis Attacks," *IEEE Transactions on Computers*, vol. 51 N°5, pp. 541-552, 2002.
- [Mess99] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards," presented at USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10-11, 1999.
- [Monn04] Y. Monnet, M. Renaudin, and R. Leveugle, "Asynchronous Design Sensitivity to Fault Injection," presented at 10th IEEE International On-line Symposium (IOLT 04), Madeira Island, Portugal, July 12-14, 2004, 2004.
- [Monn05] Y. Monnet, M. Renaudin, and R. Leveugle, "Asynchronous Circuits Transient Faults Sensitivity Evaluation," presented at 42th Design Automation Conference (DAC05), Anaheim, USA, June 13-17, 2005.
- [Monn05a] Y. Monnet, M. Renaudin, R. Leveugle, S. Dumont, and F. Bouesse, "An Asynchronous DES Crypto-processor Secured against Fault Attacks," presented at IFIP International Conference on Very Large Scale Integration (VLSI-SOC 2005), Perth, Australia, October, 17-19, 2005.
- [Mont85] P. Montgomery, "Modular Multiplication without Trial Division," *Mathematics of Computation*, vol. 44 N° 170, pp. 519-521, 1985.

- [Moor00] S. Moore, R. Anderson, and M. Kuhn, "Improving Smartcard Security Using Self-timed Circuit Technology," presented at 4th Asynchronous Circuit Design Workshop (ACiD 2000), Grenoble, France, January 31, February 1st, 2000, 2000.
- [Moor02] Moore Simon, Ross Anderson, Cunningham Paul, Mullins Robert, and Taylor George, "Improving Smart Card Security using Self-timed Circuits," presented at Eighth International Symposium on Asynchronous Circuits and systems (ASYNC 2002). Manchester, U.K., 8-11 april, 2002.
- [Moor03] S. Moore, R. Anderson, R. Mullins, G. Taylor, and J. Fournier, "Balanced Self-checking Asynchronous Logic for Smart-card Applications," *Elsevier Science Publishers, Microprocessors and Microsystems*, vol. 27, pp. 421-430, 2003.
- [Mull01] D. Muller and N. P. Smart, "Random Register Renaming to Foil DPA," presented at Cryptographic Hardware and Embedded Systems (CHES2001), Paris, France, 2001.
- [Niko99] S. Nikolaidis and A. Chatzigeorgiou, "Analytical Estimation of Propagation Delay and Short-circuit Power Dissipation in CMOS Gates," *International Journal of Circuit Theory and Applications*, pp. 375-392, 1999.
- [NSA00] NSA, "NACSIM 5000 Tempest Fundamentals," FORT GEORGE G. MEADE, Maryland 20755, December 31, 2000
- [Oswa01] E. Oswald and M. Aigner, "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks," presented at Cryptographic Hardware and Embedded Systems (CHES2001), pp. 39-50, LNCS 2162, Paris, France, May 14-16, 2001.
- [Pan02] D. Panyasak, G. Sicard, and M. Renaudin, "A current Shaping Methodology for Low EMI Asynchronous Circuits," presented at EMC Compo, Toulouse, France, November, 14-15, 2002.
- [Pany04] D. Panyasak, "Réduction de l'Emission Electromagnétique des Circuits Intégrés: L'Alternative Asynchrone," in *Institut National Polytechnique de Grenoble*. Grenoble, France, 2004.
- [PKCS1] R. Laboratories, " PKCS#1 v2.1: RSA Encryption Standard," June 2002. <ftp://ftp.rassecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf>.
- [Pell04] L. Pellier, "Contribution au placement et à l'interconnexion de cellules virtuelles pour la synthèse topologique de circuit CMOS," in *Electronique, Optronique et Systèmes: Université Montpellier II*, 2004.
- [Plan02] L. A. Plana, P. A. Riocreux, W. J. Bainbridge, A. Bardsley, S. Temple, J. D. Garside, and Z. C. Yu, "SPA-A Synthetisable Amulet Core for Smartcard Applications," presented at Eighth International Symposium on Asynchronous Circuits and systems (ASYNC 2002). pp. 201-210, Manchester, U.K., 8-11 april, 2002.
- [Quis00] J. J. Quisquater and D. Samyde, "A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions, the SEMA and DEMA methods," presented at the rump session of EUROCRYPT'2000, Bruges, Belgium, May 14-18, 2000.

- [Rake01] P. Rakers and T. Collins, "Secure Contactless SmartCard ASIC with DPA Protection," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 559-565, 2001.
- [Rao01] J. R. Rao and P. Rohatgi, "EMpowering Side-Channel Attacks," IACR ePRINT 2001/037, May 11, 2001.
- [Raza04] A. Razafindraibe, P. Maurine, M. Robert, F. Bouesse, B. Folco, and M. Renaudin, "Secured Structures for Secured Asynchronous QDI Circuits," presented at XIX Conference on Design of Circuits and Integrated Systems (DCIS2004), Bordeaux, France, November, 24-26, 2004.
- [Rena00] M. Renaudin, "Asynchronous Circuits and System: a Promising Design Alternative," *Microelectronics for Telecommunications: Managing High Complexity and Mobility (MIGAS2000)*, special issue *Microelectronics-Engineering Journal*, vol. 54, N°1-2, Elviers Science, pp. 133-149, 2000.
- [Rena02] M. Renaudin and F. Bouesse, "On the Design of secure Chips," presented at Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 4 June : <http://tima.imag.fr/cis>, 2002.
- [Rena04] M. Renaudin, F. Bouesse, and Y. Monnet, "Improving DPA and DFA resistance of circuits using asynchronous logic," presented at e-Smart 2004 and e-government & Smartcart International Meeting 5th Edition, Sophie Antipolis, French Riviera, France, September, 22-24, 2004.
- [Rezz00] M. Rezzoug, P. Maurine, and d. Auvergne, "Second Generation Delay Model for Submicron CMOS Process," presented at PATMOS 2000, vol. 1918, pp. 159-167, Göttingen, Germany, September 13-15, 2000.
- [Rigau02] J. B. Rigaud, "Spécification de Bibliothèques pour la Synthèse de Circuits Asynchrones," in *Institut National Polytechnique de Grenoble*. Grenoble, France, 2002.
- [Rives78] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21 N° 2, pp. 120-126, February 1978.
- [Robin03] B. Robisson, E. Beigne, F. Bouesse, and M. Renaudin, "Improvement of Smartcard Security with Asynchronous Logic Chips," presented at International Micro and Nanotechnology meeting (MINATEC2003), Alpes Congrès, Grenoble, France, September, 22-26, 2003.
- [Ross01a] A. Ross, "Security Engineering: A Guide to Building Dependable Distributed Systems," Wiley Computer publishing ed, vol. 1, pp. 153-154, United States of America, 2001.
- [Ross96] A. Ross and M. Kuhn, "Tamper Resistance a cautionary Note," presented at second USENIX Workshop on Electronic Commerce Proceedings, pp. 1-11, November 1996, 1996.
- [Ross01] A. Ross, "Security Engineering: A Guide to Building Dependable Distributed Systems," vol. 1, Wiley Computer publishing ed. United States of America, 2001.
- [Ross01a] A. Ross, "Security Engineering: A Guide to Building Dependable Distributed Systems," vol. 1, Wiley Computer publishing ed. United States of America, 2001, pp. 306-310.

- [Samy02] D. Samyde, S. Skorobogatov, R. Anderson, and J. J. Quisquater, "On a New Way to read Data from Memory," presented at First International IEEE Security in Storage Workshop, Greenbelt Maryland, December 11-11, 2002.
- [Schn01] B. Schneier, *Cryptographie Appliquée*, 2 ed: Vuibert Informatique, 2001. ISBN 2-7117-8676-5.
- [Schr03] K. Schramm, T. Wollinger, and C. Paar, "A New Class of Collision Attacks and its Application to DES," *Fast Software Encryption*, vol. 2887, Lectures Notes in Computer Science. Springer, 2003.
- [Schr04] K. Schramm, G. Leander, P. Kelke, and C. Paar, "A Collision-Attack on AES, Combining Side-channel and Differential-Attack," presented at CHES2004, vol. 3156 of LNCS, pp. 163-175, 2004.
- [Sham00] A. Shamir, "Protecting Smart Card from Passive Power Analysis with Detached Power Supplies," presented at Cryptographic Hardware and Embedded Systems (CHES2002), pp. 71-77, LNCS 1965, San Francisco, USA, 2000.
- [Slim04] K. Slimani, "Une Méthodologie de Conception de Circuits Asynchrones à Faible Consommation d'Energie: Application au Microprocesseur MIPS," in *Institut National Polytechnique de Grenoble*. Grenoble, France, 2004.
- [Soko04] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Improving the Security of Dual-Rail Circuits," presented at Cryptographic Hardware and Embedded Systems (CHES2004), LNCS 3156, pp. 282-297, Boston, USA, 2004.
- [Stin96] D. Stinson, *Cryptographie: Théorie et Pratique*, International Thomson Publishing ed. Paris: International Thomson Publishing France, 1996. 2-84180-013. pp. 106
- [Tiri02] K. Tiri and I. Verbauwhede, "Adynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards," presented at 28th European Solid-State Circuits Conference (ESSCIR2002), pp. 403-406, Fidenza, Italy, September, 24-26, 2002.
- [Tiri04] K. Tiri and I. Verbauwhede, "A logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation," presented at Design Automation and Test in Europe Conference (DATE04), Paris, France, 2004.
- [Wadd04] J. Waddle and D. Wagner, "Towards Efficient Second -Order Power Analysis," presented at Cryptographic Hardware and Embedded Systems (CHES2004), pp. 1-15, LNCS 3156, Cambridge, USA, August 11-13, 2004.

BIBLIOGRAPHIE DE L'AUTEUR

- [Boue02] F. Bouesse, L. Fesquet, and M. Renaudin, "QDI circuit to Improve Smartcard Security," presented at 2nd Asynchronous Circuit Design Workshop (ACID2002), Munich; Germany, 28-29 January 2002, 2002.
- [Boue02a] F. Bouesse and M. Renaudin, "Rapport de Conception et Synthèse du Cryptoprocasseur Asynchrone DES:ADIDES," Groupe CIS du laboratoire TIMA, Grenoble, France Décembre 2002.
- [Boue04c] F. Bouesse, M. Renaudin, and F. Germain, "Asynchronous AES Crypto-processor Including Secured and Optimized Blocks," *Journal of Integrated Circuits and Systems (JICS)*, vol. 1, pp. 5-13, 2004.
- [Boue04] F. Bouesse, M. Renaudin, P. Proust, J. P. Tual, L. Sourgen, and F. Germain, "High Security Smartcards," presented at Design, Automation and Test in Europe Conference and Exhibition (DATE 04), Paris France, February 16-20, 2004, 2004.
- [Boue04a] F. Bouesse, M. Renaudin, B. Robisson, E. Beigne, P. Y. Liardet, S. Prevosto, and J. Sonzogni, "DPA on Quasi Delay Insensitive Asynchronous Circuits: Concrete Results," presented at XIX Conference on Design of Circuits and Integrated Systems (DCIS2004), Bordeaux, France, November, 24-26, 2004.
- [Boueb] F. Bouesse, M. Renaudin, and G. Sicard, "Improving DPA Resistance of Quasi Delay Insensitive Circuits Using Randomly Time-shifted Acknowledgement Signals," presented at IFIP International Conference on Very Large Scale Integration (VLSI-SOC 2005), Perth, Australia, October, 17-19, 2005.
- [Boue05] F. Bouesse, M. Renaudin, A. Witon, and F. Germain, "A Clock-less low-voltage AES crypto-processor," presented at 31st European Solid-State Circuits Conference (ESSCIRC2005), pp. 403-406, Grenoble, France, 2005.
- [Boue04b] F. Bouesse, G. Sicard, A. Baixas, and M. Renaudin, "Quasi Delay Insensitive Asynchronous Circuits for Low EMI," presented at 4th International Workshop on Electromagnetic Compatibility on Integrated Circuits (EMC COMPO4), Angers, France, March 31-April 1, 2004.
- [Boue05a] F. Bouesse, G. Sicard, and M. Renaudin, "Efficient Quasi Delay Insensitive Asynchronous Architectures for Low EMI," presented at 5th International Workshop on Electromagnetic Compatibility on Integrated Circuits (EMC COMPO 05), Munich, Germany, November, 28-30, 2005.

- [Mauri03] P. Maurine, J. B. Rigaud, F. Bouesse, G. Sicard, and M. Renaudin, "Static Implementation of QDI Asynchronous Primitives," presented at 13th International Workshop on Power and Timing Modeling, Optimization and Simulations (PATMOS2003), Torino, Italy, 10-12 September, 2003, 2003.
- [Monn05a] Y. Monnet, M. Renaudin, R. Leveugle, S. Dumont, and F. Bouesse, "An Asynchronous DES Crypto-processor Secured against Fault Attacks," presented at IFIP International Conference on Very Large Scale Integration (VLSI-SOC 2005), Perth, Australia, October, 17-19, 2005.
- [Raza04] A. Razafindraibe, P. Maurine, M. Robert, F. Bouesse, B. Folco, and M. Renaudin, "Secured Structures for Secured Asynchronous QDI Circuits," presented at XIX Conference on Design of Circuits and Integrated Systems (DCIS2004), Bordeaux, France, November, 24-26, 2004.
- [Rena02] M. Renaudin and F. Bouesse, "On the Design of secure Chips," presented at Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 4 June : <http://tima.imag.fr/cis>, 2002.
- [Rena04] M. Renaudin, F. Bouesse, and Y. Monnet, "Improving DPA and DFA resistance of circuits using asynchronous logic," presented at e-Smart 2004 and e-government & Smartcart International Meeting 5th Edition, Sophia Antipolis, French Riviera, France, September, 22-24, 2004.
- [Robin03] B. Robisson, E. Beigne, F. Bouesse, and M. Renaudin, "Improvement of Smartcard Security with Asynchronous Logic Chips," presented at International Micro and Nanotechnology meeting (MINATEC2003), Alpes Congrès, Grenoble, France, September, 22-26, 2003.
- [Boues05c] F. Bouesse, M. Renaudin, S. Dumont, F. Germain, «DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement», Design Automation and Test in Europe Conference and Exhibition (DATE 2005), Munich Germany, March 7-11, 2005, p.424
- [Renau05a] M. Renaudin, F. Bouesse, Y. Monnet "Secure asynchronous circuits for smart-Card applications: Design and Methodologies", MEDEA+ Design Automation Conference, Château des Mesnuls, France, May 24-26, 2005.
- [Renau05b] Marc Renaudin, Fraïdy Bouesse, Yannick Monnet, Laurent Fesquet, "Secure asynchronous circuits design and prototyping", 3rd International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices - CryptArchi 2005, Saint-Etienne, France, June 8 - 11th 2005.
- [Boue05e] Fraïdy Bouesse, Marc Renaudin, "Improving DPA resistance of Quasi-delay Insensitive Asynchronous Circuits", 3rd International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices - CryptArchi 2005, Saint-Etienne, France, June 8 - 11th 2005.
- [Monn05b] Yannick Monnet, Fraïdy Bouesse, Marc Renaudin, "designing resistant asynchronous circuits against malicious fault injection", 3rd International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices - CryptArchi 2005, Saint-Etienne, France, June 8 - 11th 2005.
- [Fesq05] Laurent Fesquet, Fraïdy Bouesse, Yannick Monnet, Marc Renaudin, "Implementing Asynchronous Circuits on LUT Based FPGAs", 3rd International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices - CryptArchi 2005, Saint-Etienne, France, June 8 - 11th 2005

- [Boue05f] F. Bouesse, "Les Circuits Asynchrones Quasi Insensibles aux Délais et la Sécurité Matérielle", Journées Nationales du Réseau des Doctorants en Microélectronique (JNRDM'05), 9-11 mai 2005, Paris, France.
- [Mauri03a] P. Maurine, J. B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, "TAL : une bibliothèque de cellules pour le design de circuits asynchrones QDI", (FTFC'03), 4èmes journées d'études Faible Tension, Faible Consommation, 15-16 Mai 2003, Paris, France, pp. 41-45.

RESUME

Ce travail de thèse s'intègre dans le cadre du développement de nouvelles techniques de protection des circuits intégrés face aux attaques par analyse de courant en exploitant les propriétés de la logique asynchrone. En effet, ces attaques qui exploitent les faiblesses d'implémentation matérielle des composants cryptographiques pour retrouver des informations secrètes, sont parmi les attaques non intrusives les plus efficaces et les plus faciles à mettre en œuvre. Ainsi, nous proposons dans ces travaux l'utilisation de la logique asynchrone Quasi Insensible aux Délais (QDI) pour sécuriser les circuits intégrés contre ce type d'attaques. Les propriétés de la logique QDI apparaissent particulièrement intéressantes pour sécuriser l'implémentation des circuits intégrés car elles permettent de contrôler finement l'activité électrique. Le travail a porté dans un premier temps sur l'évaluation de la résistance des circuits asynchrones QDI. Les résultats obtenus montrent une nette amélioration du niveau de sécurité d'un circuit asynchrone par rapport à son équivalent synchrone, et permettent également d'identifier les limites de cette approche. Nous avons développé dans ces travaux, une méthode d'analyse formelle afin d'évaluer la sensibilité de la logique asynchrone QDI et présentons par la suite, de nouvelles contre-mesures exploitant la topologie de ces circuits. Cette étude a ainsi conduit à spécifier de nouvelles méthodologies de conception de circuits asynchrones sécurisés dans le but de pouvoir les intégrer dans la méthodologie automatisée TAST (TIMA Asynchronous Synthesis Tools).

MOT-CLES

Circuits asynchrones, Circuits quasi insensibles aux délais, Cryptanalyse matérielle, Attaques par canaux cachés, Attaques en puissance, Circuits résistants, Méthodologie de conception.

TITLE

CONTRIBUTION TO SECURE DESIGN OF INTEGRATED CIRCUITS: THE ASYNCHRONOUS ALTERNATIVE

ABSTRACT

This work is focused on the development of new design techniques for protecting integrated circuits against power analysis attacks by exploiting the properties of asynchronous logic. In fact, among non intrusive attacks which exploit the hardware weaknesses of cryptographic devices for retrieving confidential information, the power analysis attacks are the most efficient and the easiest to implement. In this work the countermeasures developed are based on Quasi Delay Insensitive asynchronous logic (QDI) and focused on the protection of integrated circuits against power analysis attacks. The properties of the QDI asynchronous logic are particularly interesting for securing an implementation because it enables the designer to precisely control the current activity. The work was first concentrated on the evaluation of the resistance of asynchronous logic to DPA. The results obtained demonstrate the potentiality of the QDI properties to improving chips' security compared to synchronous logic, and enable us to identify some limits of this approach. We propose a formal analysis to evaluate the sensitivity of QDI asynchronous logic to power analysis and then present new countermeasures that exploit the QDI logic topology. These studies lead to the specification of a new design methodology for implementing secure asynchronous chips which will be integrated in the TAST framework (TIMA Asynchronous Synthesis Tools).

INTITULE ET ADRESSE DU LABORATOIRE

Laboratoire TIMA, 46 avenue Félix Viallet, 38031 Grenoble Cedex, France.

ISBN : 2-84813-077-6

ISBNE : 2-84813-077-6