Logic-Level Countermeasures to Secure FPGA based Designs

Shivam Bhasin

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Électronique et Communication »

présentée et soutenue publiquement par

Shivam BHASIN

le 14 Décembre 2011

Contremesures au niveau logique pour sécuriser les architectures de crypto-processeurs dans un FPGA

Directeur de thèse : **Jean-Luc DANGER**
Co-encadrement de la thèse : **Tarik GRABA**

Jury

M. François-Xavier STANDAERT, Professeur, UCL, Belgique	Président
M. Viktor FISCHER, Professeur, ENSM SE, France	Rapporteur
Mme. Ingrid VERBAUWHEDE, Professeur, KUL, Belgique	Rapporteur
M. Habib MEHREZ, Professeur, UPMC, France	Examinateur
M. Patrick SCHAUMONT, Professeur, Virginia Tech, Etats-Unis	Examinateur
M. Sylvain GUILLEY, Maitre de Conferences Associé, Télécom ParisTech, France	Examinateur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

T H È S E

Abstract

Modern field programmable gate arrays (FPGA) are capable of implementing complex system on chip (SoC) and providing high performance. Therefore, FPGAs are finding wide application. A complex SoC generally contains embedded cryptographic cores to encrypt/decrypt data to ensure security. These cryptographic cores are computationally secure but their physical implementations can be compromised using side channel attacks (SCA) or fault attacks (FA). This thesis focuses on countermeasures for securing cryptographic cores on FPGAs.

First, a register-transfer level countermeasure called "Unrolling" is proposed. This hiding countermeasure executes multiple rounds of a cryptographic algorithm per clock which allows deeper diffusion of data. Results show excellent resistance against SCA. This is followed by dual-rail precharge logic (DPL) based countermeasures, which form a major part of this work. Wave dynamic differential logic (WDDL), a commonly used DPL countermeasure well suited for FPGAs is studied. Analysis of WDDL (DPL in general) against FA revealed that it is resistant against a majority of faults. Therefore, if flaws in DPL namely early propagation effect (EPE) and technological imbalance are fixed, DPL can evolve as a common countermeasure against SCA and FA. Continuing on this line of research we propose two new countermeasures: DPL without EPE and Balanced-Cell based DPL (BCDL).

Finally advanced evaluation tools like stochastic model, mutual information and combined attacks are discussed which are useful when analyzing countermeasures.

Dedicated to my family,
for their unconditional love and support.

Acknowledgements

This PhD has been performed in the auspices of group SEN (Systèmes Électronique Numérique) of Department COMELEC (Communications & Électronique) of Telecom-ParisTech. During these three years of graduate studies, I have learned the essence of research and development. This has been possible by working and interacting with a group of very intelligent, resourceful and kind personalities. Surely without them, my stay in Telecom-ParisTech would not be as fruitful.

First and foremost,I extend my utmost thankfulness to my PhD director, Professor Jean-Luc Danger for his guidance and encouragement. His experience and expertise in the subject are matchless. Having a highly diverse knowledge base, his suggestions were of immense importance. He was of great help in getting out of many complex circumstances.

I would like to express my most sincere gratitude to Dr. Sylvain Guilley, who showed confidence in me and gave me numerous advice in academics, research and career. He is a person of great scientific acumen and admirable human values. He was by my side throughout the three years and taught me how to find paths in the dark alleys of research.

I am extremely thankful to my Co-adviser, Dr. Tarik Graba, for his guidance through various lengthy and productive discussions on the subject. His profound knowledge and extensive experience along with his helping nature proved to be a treasure for me. I take great pride in having worked with these three gentlemen, who are the specialists in the domain of Cryptography and Security. I have learned a lot from them.

I am also thankful to Professor Francois-Xavier Standaert for accepting me as an intern in his team at UCL, Belgium for a period of three months. His profound knowledge in the feild of side-channel analysis helped me learn a lot and widen my research perspectives.

I am extremely grateful to the members of my jury who gave their consent to analyze and consequently validate my work. I feel honored that my work got approved by such well renowned specialists.

The moments spent with many peers is perhaps one of the best memories that I will take with me of my graduate studies. I cherish all the moments of stress and joy spent with my collegues Nidhal, Olive (Olivier), Joe (Youssef), Fhal (Houssem), Max (Maxime), Taufeek, Seb (Sebastian), Zizou (Aziz), Zouha, Nicholas, Amel, Florent, Chantal, Daniel, Zouina and Laurent. I specially admire the efforts of Olivier which he took to teach me french. I have very much enjoyed being in the company of friends like Babu (Sumanta), Shady (Chadi), Mirani (Farhan), Hasham, Mayassa, Marcel, Paulo, Francesco, Lauriane, Nora, Davi, Gutemberg, Khalil, Waqqas, Sami, Tushar, Arif and the list goes on and on. The various ecsapades in Paris with Shady (Chadi), Babu (Sumanta), Hasham and Mirani (Farhan) goes into my ROM.

I am most fortunate to have my family: my parents, my grandmother and sister. Their constant encouragement has pushed me to be the best that I can be. I am certain that this work has been a product of my family's love and support throughout my life.

Last but not the least I am thankful to The French National Research Agency (ANR), who financed this thesis through the ANR-07-ARFU-010 grant "SeFPGA" (Secured Embedded FPGAs). I would also like to thank the DGA project "Rapid BCDL" for its support.

Contents

List of Figures

List of Tables

Glossary

AES:	Advanced Encryption Standard
ASIC:	Application Specific Integrated Circuit
BCDL:	Balanced Cell based Dual-rail Logic
CBC:	Cipher Block Chaining
CLB:	Configurable Logic Block
CPA:	Correlation Power Analysis
DBA:	Differential Behavioral Analysis
DDR:	Double Data Rate
DES:	Data Encryption Standard
DFA:	Differential Fault Attack
DPA:	Differential Power Analysis
DPL:	Dual-rail with Precharge Logic
DPL w/ EPE:	DPL with Early Propagation Effect
DPL w/o EPE:	DPL without Early Propagation Effect
DRSL:	Dual-rail Random Switching Logic
ECB:	Electronic Code Book
ECC:	Elliptic Curve Cryptography
EPE:	Early Propagation Effect
EMA:	Electro Magnetic Analysis
FA:	Fault Attack
FIPS:	Federal Information Processing Standard
FPGA:	Field Programmable Gate Array
GF:	Galois Field
LUT:	Look-Up Table
MDPL:	Masked Dual-rail Precharge Logic
MIA:	Mutual Information Analysis
MTD:	Minimum Traces to Disclose the key
NIST:	National institute of standard and technology
RAM:	Random Access Memory
RSA:	Rivest Shamir Adleman
RTL:	Register Transfer Logic
PC:	Possibility of Combination
PKC:	Public Key Cryptography
SCA:	Side Channel Attack
SKC:	Symmetric Key Cryptography
SPA:	Simple Power Analysis
UART:	Universal Asynchronous Receiver Transmitter
WDDL:	Wave Dynamic Differential Logic

Résumé Français

Le réseau de portes programmable (FPGA) est un circuit intégré qui contient des cellules logiques identiques en tant que composants standards. Chaque cellule logique peut être programmée indépendamment. Ces cellules logiques identiques sont appelées blocs logiques configurables (CLB). La Figure 1 montre la cellule CLB qui se compose d'une "Look- up table (LUT)"" à quatre entrées. Elle permet de mettre en œuvre toute fonction booléenne à quatre entrées et même une bascule active sur front pour les circuits séquentiels. Les cellules CLB individuelles sont reliées entre elles par une matrice de fils et par des commutateurs programmables.

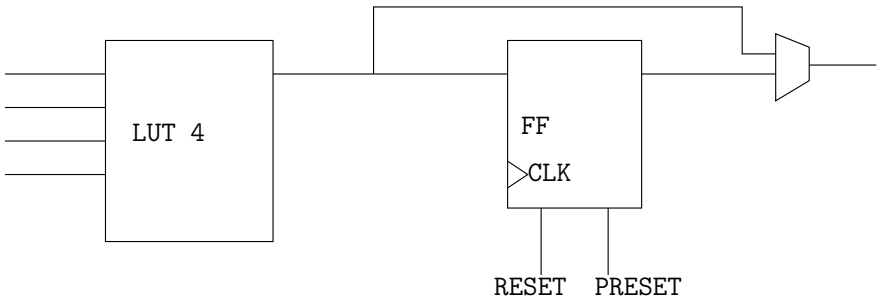


Figure 1: Une cellule logique de base dans un FPGA

Les FPGAs commerciales possèdent des cellules CLB plus complexes par rapport aux blocs logiques de base comme des multiplexeurs, chaînes à retenue rapide, registres en cascade, "Set/Reset" sur des bascules etc. Les CLBs dans un FPGA commercial sont souvent organisées de façon hiérarchique (voir la Figure 2). Les FPGAs comprennent également des macros complexes tel que des multiplieurs, des blocs DSP, RAM.

Les FPGAs ont considérablement évolués au cours de ces dernières années. Les plus modernes sont capables de mettre en œuvre un système complexe sur puce (SoC) et de fournir des performances élevées. Par conséquent, on les utilise dans un large domaine d'applications dans les circuits réels. Un complexe SoC contient généralement un noyau

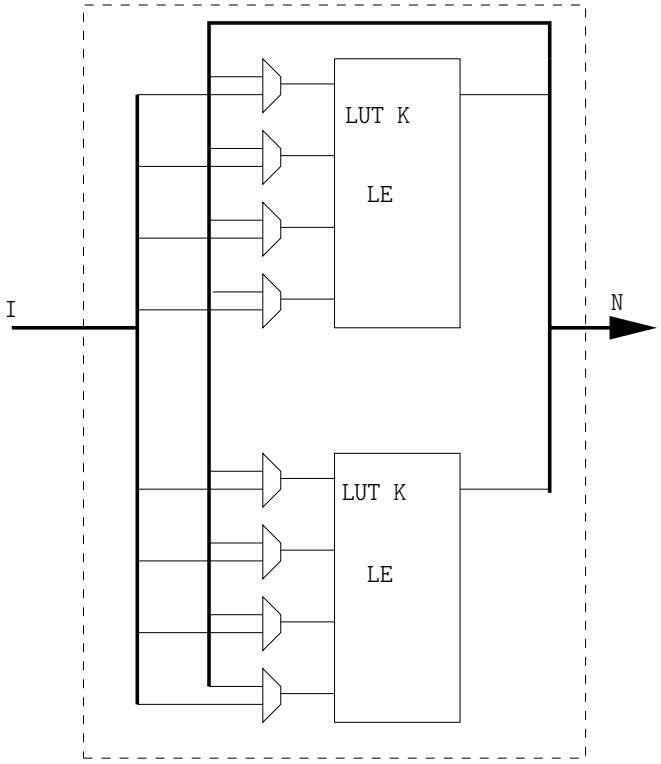


Figure 2: Logic cluster.

cryptographique embarqué pour chiffrer/déchiffrer des données sur le bus système et donc assurer la sécurité. Ces noyaux cryptographiques sont mathématiquement sûrs, mais leurs implémentations physiques peuvent être défaillantes.

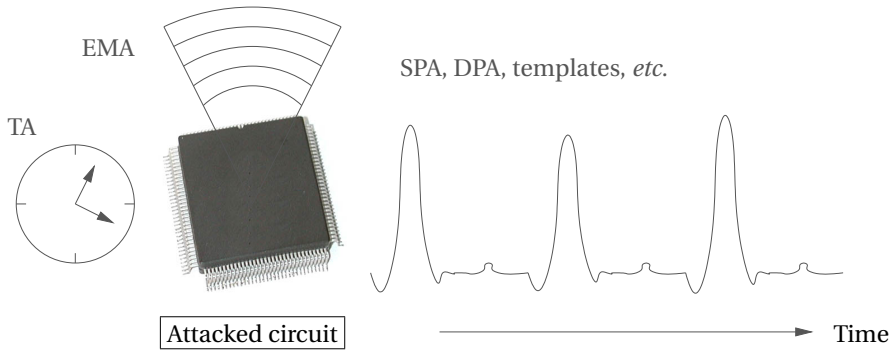


Figure 3: Différentes attaques par canaux auxiliaires

Plusieurs attaques ont été réussies contre de tels noyaux en ciblant leur mise en œuvre

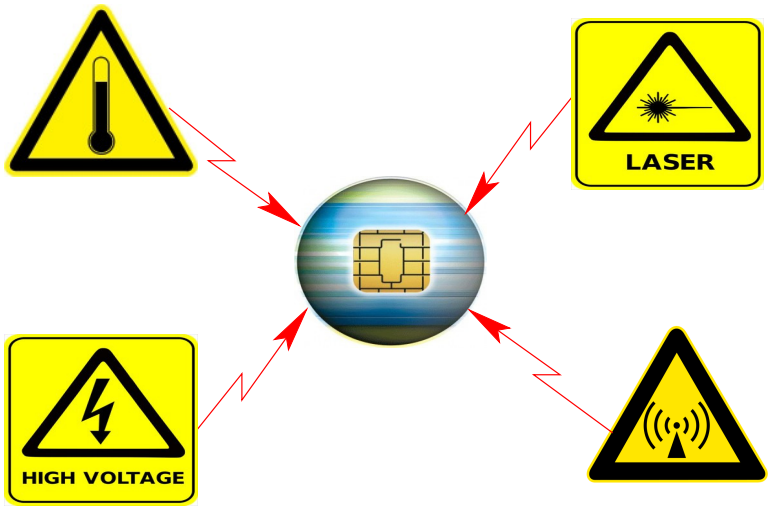


Figure 4: Différentes attaques par injection de fautes

physique. La première attaque physique qui a été publiée est nommée "Timing attack" qui a été présentée par Kocher et al. en 1996 [1]. Dans cette attaque, un adversaire est capable de récupérer une clé secrète utilisée dans un algorithme de signature par la violation du temps d'exécution de diverses opérations. Cette attaque est un exemple d'attaque par canaux auxiliaires (SCA) (voir la figure 3) puisque c'est entièrement passif : le système attaqué n'est pas détruit. D'autres attaques par canaux auxiliaires permettent d'espionner les propriétés du circuit comme la consommation d'énergie et le rayonnement électromagnétique [2, 3]. Leur étude a mobilisé de nombreux chercheurs.

Ces attaques se déroulent en deux étapes : la collecte d'informations par canaux auxiliaires puis leurs analyses. La collecte par canaux auxiliaires est une étape de métrologie, par contre, l'étape d'analyse nécessite des outils sophistiqués théoriques pour être efficace. Ces deux aspects progressent rapidement, comme le témoigne le concours "DPA Contest" [4]. Une autre classe d'attaques physiques connues sous le nom "active" ou attaques par faute (FA, voir la figure 4) fonctionne en modifiant le comportement fonctionnel du dispositif attaqué par des perturbations [5, 6]. Cette thèse porte sur les contremesures SCA et FA pour sécuriser les noyaux cryptographiques dans les FPGAs.

Jusqu'à maintenant, de nombreuses contremesures ont été conçues pour sécuriser les implémentations de systèmes cryptographiques. L'objectif principal de ces contremesures est d'éliminer toute dépendance entre les opérations internes d'un crypto-système et les fuites correspondantes. La plupart des contremesures, si ce n'est la totalité, utilise la redon-

dance pour renforcer le crypto-système contre la SCA et la FA. Ces contremesures peuvent être classées en deux catégories:

1. **masquage de l'information**, [7, Chap. 9], qui vise à rendre la fuite aléatoire
2. **dissimulation de l'information**, [7, Chap. 7], qui vise à cacher l'information secrète présente dans les canaux cachés.

Les contremesures basées sur le masquage de l'information utilisent une valeur générée aléatoirement appelée "masque" de telle sorte que la fuite corresponde au masque sans affecter la valeur du chiffré de sortie comme le montre la figure 5. Une implémentation correcte d'une contre-mesure basée sur le masquage protège contre les attaques SCA de premier ordre. Les attaques d'ordre supérieur peuvent néanmoins réussir.

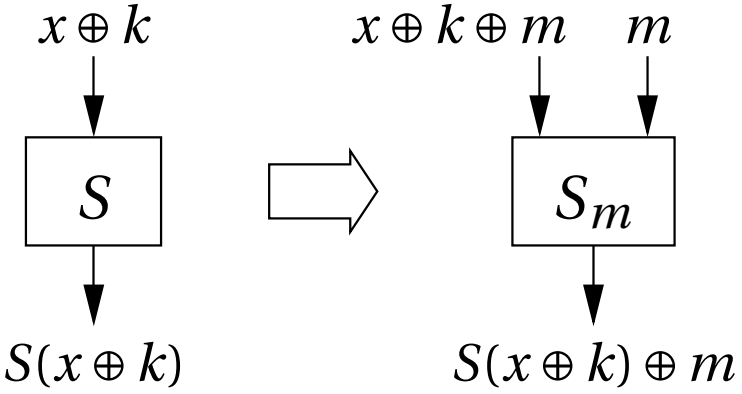


Figure 5: Une S-box masquée en logiciel

Les contremesures utilisant la dissimulation d'information, comme le nom l'indique, tentent de cacher l'information de l'attaquant. La résistance de ces contremesures face aux attaques FA n'a pas été étudiée en profondeur. Dans le travail présenté, l'accent est mis sur l'étude des contremesures basées sur la dissimulation de l'information pouvant être implémentée sur un FPGA commerciale. Mon objectif est d'étudier le bien-fondé de ces contremesures, de les implémenter efficacement sur FPGA et d'effectuer une analyse sécuritaire contre les attaques SCA et FA connues dans l'état de l'art. Les contremesures sur lesquelles je me focalise sont (sous leurs dénominations anglaises): **Loop Unrolling** et **Dual-Rail Precharge Logic (DPL)**.

Dans un algorithme de chiffrement par bloc, les données sont chiffrées en répétant une série d'opérations et en utilisant une clef différente à chaque fois, générée à partir de

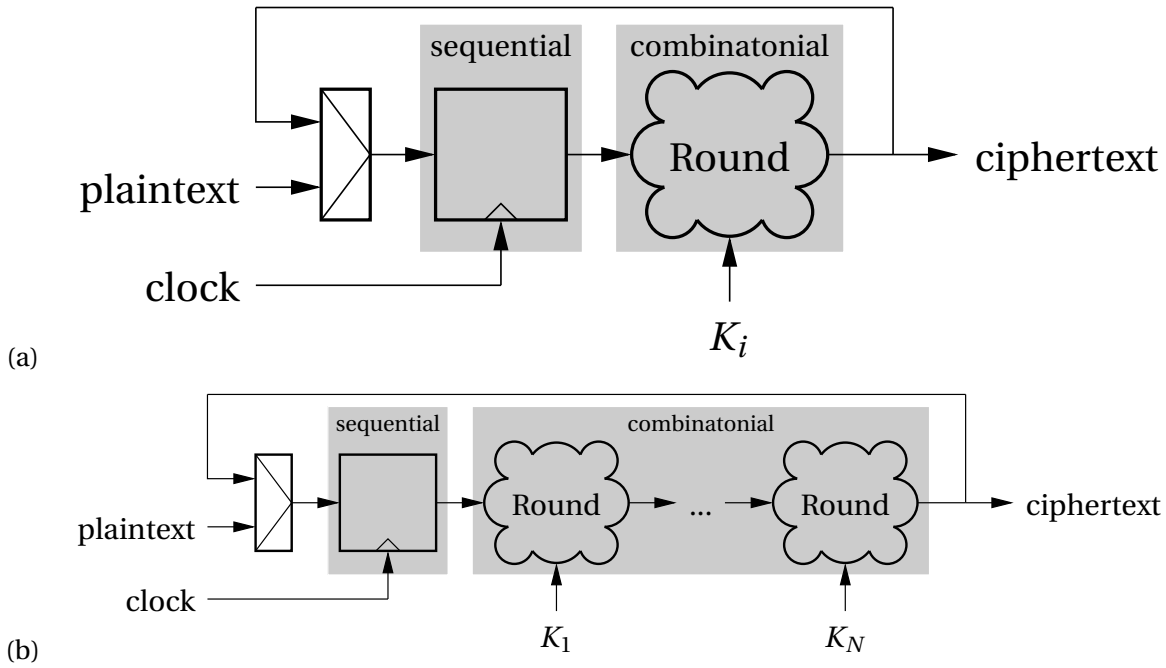


Figure 6: (a) Architecture d'un algorithme cryptographique itératif. (b) Architecture d'un algorithme cryptographique totalement "unrolled".

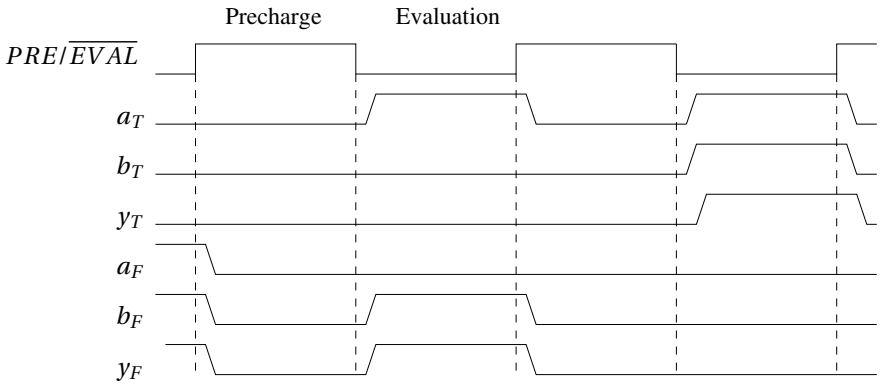


Figure 7: Chronogramme d'une porte ET dans le WDDL.

la clef précédente. Cet ensemble d'opérations est appelé une tour. Le nombre de tours est choisi de telle sorte que les cryptanalyses linéaire et différentielle soient plus difficiles qu'une recherche exhaustive sur l'ensemble des clefs possibles. Les circuits cryptographiques sont généralement conçus pour effectuer soit certaines opérations, soit la tour entière en un seul cycle d'horloge comme le montre la figure 6(a). L'idée du "unrolling" est d'implémenter

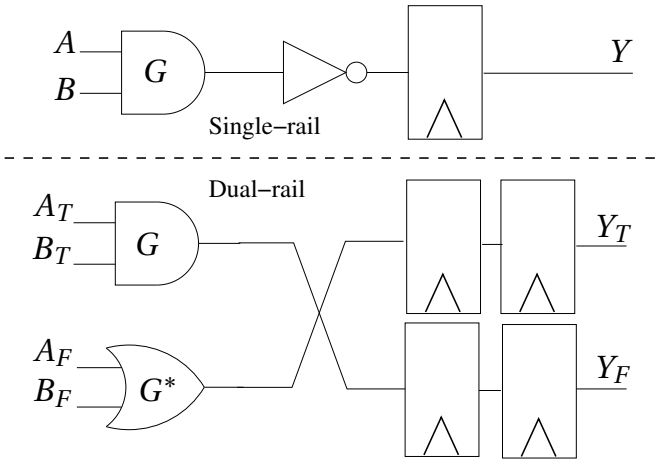


Figure 8: Conversion d'un circuit numérique classique en WDDL.

le circuit de telle manière que les multiples tours soient calculées en un seul cycle comme le montre la figure 6(b). Il s'agit d'une contre-mesure appliquée au niveau RTL (Register Transfer Logic), ce qui la rend totalement indépendante de la plateforme utilisée. Elle peut être classée comme une contre-mesure dissimulant l'information car l'information est toujours présente mais d'une manière cachée. En d'autres termes, il est très difficile de trouver un modèle de fuite permettant d'exploiter l'information présente.

Une contremesure couramment utilisée pour protéger ces systèmes cryptographiques est DPL (Dual-rail Precharge Logic). La première phase de la thèse a porté sur les contremesures DPL proposées précédemment. WDDL (Wave Dynamic Differential Logic) est une contre-mesure DPL couramment utilisée, composée sur toute la logique positive. Elle est ainsi bien adaptée pour les FPGA. La figure 8 montre la conversion d'un simple circuit numérique en WDDL. Ces circuits effectuent un nombre uniforme de transitions par cycle, par le biais de duplications et d'opérations en deux phases (pré-charge et évaluation). Il peut être vérifiée à partir de la figure 7 qu'une porte AND implémentée avec le système WDDL a une consommation d'énergie très équilibrée. Non sans difficulté, il a finalement été montré que le WDDL était vulnérable aux attaques SCA. Comme indiqué dans des diverses publications, cette vulnérabilité vient de l'effet de propagation précoce (early propagation effect (EPE)) et du routage déséquilibré sur le FPGA. Toutefois, WDDL a la remarquable propriété de résister contre la majorité des fautes (fautes asymétriques). Par conséquent, s'il existe un moyen de retirer les défauts mentionnés ci-dessus à la technologie DPL, cela peut devenir une contre-mesure commune contre les attaques SCA et FA. Ainsi

Table 1: Calcul de maques pour une LUT-4 dans DPL w/ EPE

DPL w/ EPE				AND_T		AND_F	
a_T	a_F	b_T	b_F	CCOO		FAFA	
0	0	0	0	0	O	0	A
0	0	0	1	0		1	
0	0	1	0	0		0	
0	0	1	1	0		1	
0	1	0	0	0	O	1	F
0	1	0	1	0		1	
0	1	1	0	0		1	
0	1	1	1	0		1	
1	0	0	0	0	C	0	A
1	0	0	1	0		1	
1	0	1	0	1		0	
1	0	1	1	1		1	
1	1	0	0	0	C	1	F
1	1	0	1	0		1	
1	1	1	0	1		1	
1	1	1	1	1		1	

les nouvelles variantes de DPL évitant l'EPE et le déséquilibre de routage sont devenus un champ de recherche intéressant.

Pour améliorer la contremesure DPL par la contremesure DPL sans EPE en une simple modification dans la table de vérité des portes logiques de base est suffisante. Dans les FPGAs, on peut réaliser cette modification en forçant le masque de la LUT à une valeur précalculée. Les tableaux 1 et 2 montrent respectivement le calcul de masque de LUT pour la contremesure DPL normale et pour la contremesure DPL sans EPE.

L'analyse par information mutuelle (MIA) montre que la suppression de l'effet EPE réduit énormément la fuite du canal auxiliaire comme le montre la figure 9. Ce qui reste de la fuite provient du routage déséquilibré qui est le résultat de "fan-out" élevé des portes logiques. Il est très difficile de contrôler le routage dans les FPGAs pour les grandes conceptions car même si on met des contraintes de routage elles peuvent être ignorées par la plus part des outils de synthèse pendant la phase de l'optimisation. J'ai forcé le routage sur

Table 2: Calcul de maques pour une LUT-4 dans DPL w/o EPE

DPL w/o EPE				AND_T		AND_F		**Input state in the DPL protocol**
a_T	a_F	b_T	b_F	FC80		FAE0		
0	0	0	0	0		0		All NULL0
0	0	0	1	0	0	0	0	Transitional from NULL0
0	0	1	0	0		0		Transitional from NULL0
0	0	1	1	0		0		Faulty
0	1	0	0	0		0		Transitional from NULL0
0	1	0	1	0	8	1	E	All VALID: $(a,b) = (0,0)$
0	1	1	0	0		1		All VALID: $(a,b) = (0,1)$
0	1	1	1	1		1		Transitional from NULL1
1	0	0	0	0		0		Transitional from NULL0
1	0	0	1	0	C	1	A	All VALID: $(a,b) = (1,0)$
1	0	1	0	1		0		All VALID: $(a,b) = (1,1)$
1	0	1	1	1		1		Transitional from NULL1
1	1	0	0	1		1		Faulty
1	1	0	1	1	F	1	F	Transitional from NULL1
1	1	1	0	1		1		Transitional from NULL1
1	1	1	1	1		1		All NULL1

l'FPGA Altera Stratix mais ce la n'a pas apporté beaucoup d'amélioration.

La deuxième contremesure est BCDL("Balanced-cell based DPL"). Cette logique se composent d'un **signal globale PRE** qui est utilisé pour synchroniser toutes les portes ce qui élimine l'effet EPE comme le montre la figure 10. L'autre avantage de l'utilisation de le BCDL est la possibilité d'utiliser les mémoires embarquées avec une taille raisonnable. Cette avantage réduit la taille utilisée du FPGA offre une meilleure performance et un faible "fan-out". J'ai proposé une approche "bottom-up" pour implémenter BCDL dans les FPGAs. Les résultats montrent une résistance effective contre les attaques passives et les attaques actives. Le routage déséquilibré peut être encore un sujet de préoccupation qui peut être contré par la commutation de chemin.

Un algorithme cryptographique comme l'AES peut être implémenté en utilisant quatre primitives:

- Portes XOR,

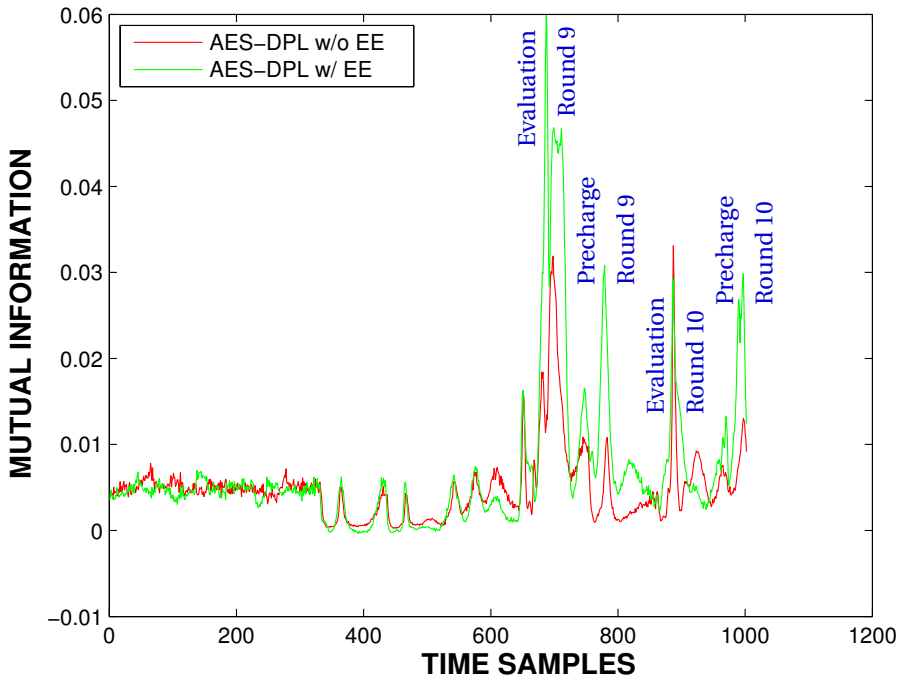


Figure 9: Comparaison de le fuite de l'information mutuelle dans deux AES protégé en utilisant DPL w/ EPE et DPL w/o EPE.

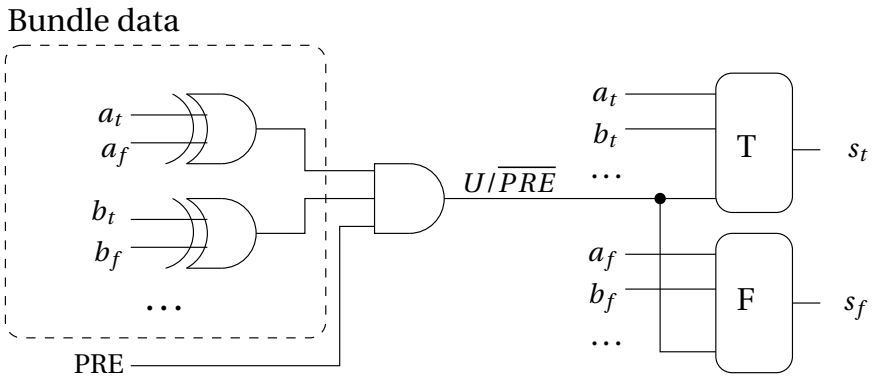


Figure 10: Cellule BCDL à n-entrées.

- Multiplexeurs,
- S-boxes (mémoires)
- Registres (DFF).

Pour convertir un simple AES en BCDL en utilisant l'approche "bottom-up", les étapes suivante doivent être suivies:

1. Le code pour un AES avec rail unique, devrait être écrit d'une façon structurelle en utilisant des primitives. Dans le cas spécifique d'AES, ces primitives sont une bascule D, des mémoires (pour les s-box), des portes XOR et des multiplexeurs.

2. Tous les signaux de données devrait être identifiés et dupliqués. Ce qui veut dire qu'un quelconque signal s devrait être converti en deux signaux s_V et s_F.

3. Les quatre primitives devraient être remplacées par les primitives BCDL. Les primitives BCDL sont double-rail et sont soumises à deux phases d'opérations.

4. Ensuite, un adaptateur qui peut connecter des entrées/sorties simple-rail aux signaux de données double-rail est ajouté. Cet adaptateur assure aussi que les entrées/sorties sont pré-chargées à chaque coup d'horloge, ceci permet d'être conforme au protocol DPL.

5. Puisqu'un circuit DPL fonctionne à moitié fréquence d'un circuit simple-rail, la machine d'état devrait fonctionner durant seulement la phase d'évaluation.

Table 3: Table de vérité d'une porte XOR BCDL à 2 entrées.

PRE	$I1_{\mathbf{T}}$	$I1_{\mathbf{F}}$	$I2_{\mathbf{T}}$	$I2_{\mathbf{F}}$	$O_{\mathbf{T}}$	$O_{\mathbf{F}}$
0	X	X	X	X	0	0
1	0	0	X	X	0	0
1	1	1	X	X	0	0
1	X	X	0	0	0	0
1	X	X	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1

Une implémentation appropriée utilisant les mémoires et les portes XOR est cruciale quant à la sécurité de BCDL. Pour implémenter une porte XOR, le FPGA est forcé à utiliser le même modèle pré-calculé pour chaque porte XOR (table 3). La figure 11 montre une implémentation des mémoires qui utilisent le PRE comme bit le plus significatif de son adresse. Ainsi quand le PRE est au niveau "bas", la sortie est bloquée à "00000000" pour activer la pré-charge.

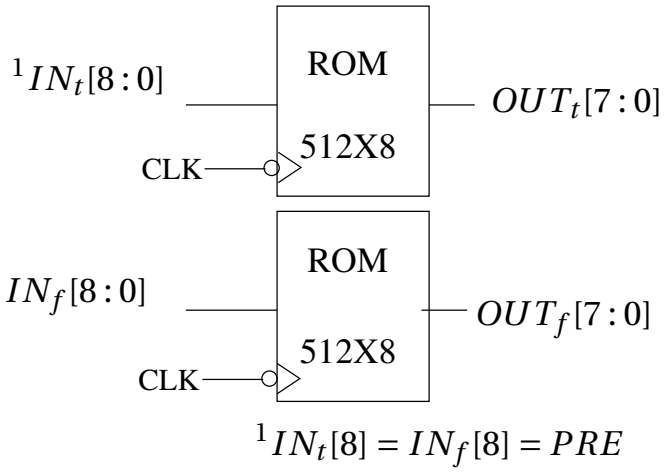


Figure 11: BCDL S-box.

La DPL w/o EPE était attaquable par suite à un routage déséquilibré. Les analyses montrent que des registres avec un grand nombre de sorties, mais aussi un chemin critique long provoquant des fuites exploitables. A partir de ces résultats, j'ai implémenté une version d'AES BCDL dont les registres ont peu de sorties, et un chemin critique plus court. Ceci était possible grâce à la version T-box d'AES. Les T-box sont un ensemble de 8x32 tables, qui combinent le SubBytes, ShiftRows et le MixColumns. Un tour complet d'AES peut être calculé en utilisant les T-box et les portes XOR. Puisque les T-box sont des tables, elles peuvent être implémentées sous forme de bloques mémoires dans le FPGA. Un AES basé sur les T-box partage les même primitives qu'un AES S-box, comme les mémoires, les portes XOR, les multiplexeurs et les registres. J'ai implémenté trois modèles, un AES non protégé, un AES BCDL en utilisant des S-box, finalement un AES BCDL en utilisant les T-box sur un FPGA Altera Stratix II. Les résultats de synthèses sont montrés dans la Table 4. Les résultats de l'attaque MIA montrent clairement la fuite dont chacune des trois implémentation (voir figure 12.

Table 4: Comparaison en coût et en performance d'implémentations AES non protégé et BCDL sous Stratix II.

Architecture	Unprotected AES	AES BCDL (S-box)	AES BCDL (T-box)
ALUT	483	2302	2669
Registers	271	1041	1041
RAMs (4 Kb)	20	40	72

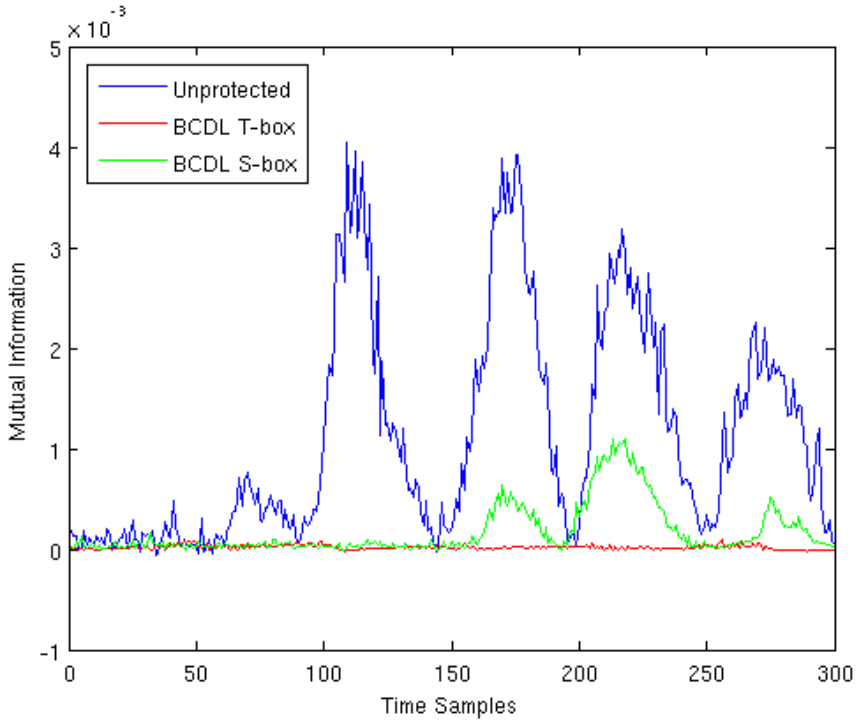


Figure 12: Traces MIA pour la comparaison de fuite de la S-Box 0 pour trois implémentations d'AES.

J'ai également étudié les outils d'évaluation du modèle stochastique, l'information mutuelle et les attaques combinées. Les techniques d'évaluation pour les implémentations non protégées ont été largement étudiées dans la littérature. Cependant, aucune méthode d'attaque formelle n'a jamais été proposé pour les implémentations DPL. Dans cette thèse, j'étudie les attaques profilées comme les attaques par "template" et le modèle stochastique comme technique d'évaluation pour estimer la fuite d'information dans le cas des implémentations DPL et la comparer avec la MIA.

50000 traces ont été utilisées dans l'étape de profilage et 10000 traces pour l'attaque. La figure 13 montre l'information mutuelle calculée en utilisant des templates et les modèles stochastiques de degré 1,2,3 & 4. La figure montre également la MIA calculée en utilisant l'estimation gaussienne sur 90000 traces. L'information estimée par les templates est supérieure à celle estimée par le modèle stochastique. Les résultats montrent également que le modèle stochastique de degré plus élevé fournit une estimation précise. Un avantage de l'estimation par le modèle stochastiques par rapport au template est que les mod-

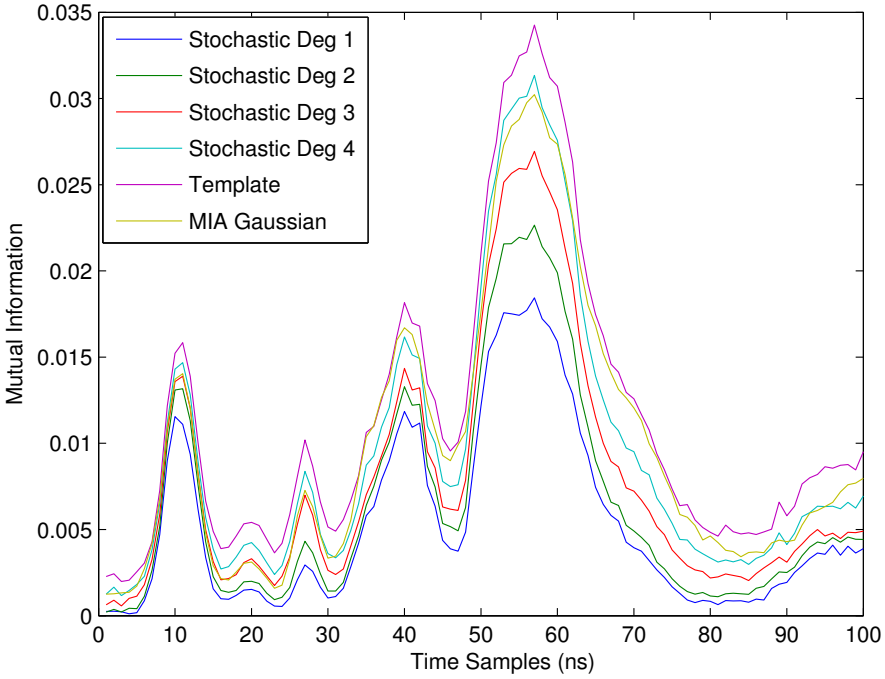
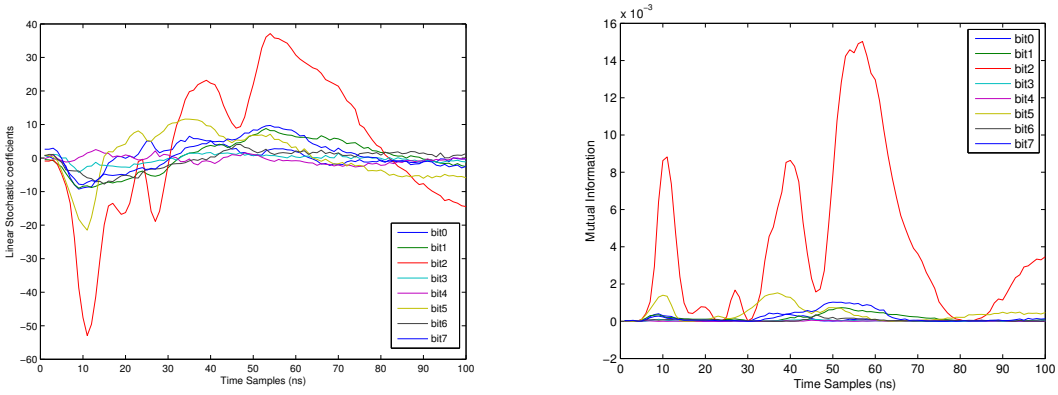


Figure 13: Estimation d'information mutuelle en utilisant "templates", MIA et modèle stochastique de degrés 1,2,3 et 4.

èles stochastique sont capables de localiser la source des fuites (figure 14). L'attaque MIA "mono-bit" peut également repérer exactement la fuite mais il y a un risque d'avoir des résultats erronés avec cette méthode.

Enfin, j'ai étudié les attaques combinées comme une méthode pour accélérer les attaques SCA. Une pratique courante de l'analyse électromagnétique (EMA) est d'acquérir les points de fuite les plus forts sur le circuit. Cependant, il ya d'autres points qui fuient l'information aussi. Nous proposons d'acquérir de multiples fuites simultanées à partir de différents points de fuite. Plusieurs antennes peuvent être utilisées pour acquérir des fuites multiples comme le montre la figure 15.

Avant de tester la combinaison, nous concaténons les traces C_1 et C_2 ensemble, c'est à dire nous nous joignons la trace acquise C_2 à la fin de la trace C_1. L'attaque est lancée sur la trace concaténée qui calcule le coefficient de corrélation de Pearson pour chaque trace sur chacune des deux sections de trace. Pour combiner le coefficient de corrélation, nous utilisons une fonction d'agrégation Ψ. Une fonction d'agrégation est un type spécial de fonc-



(a) (b)

Figure 14: Localisation de fuites en utilisant (a) coefficients stochastique linéaire et (b) MIA Mono-bit.

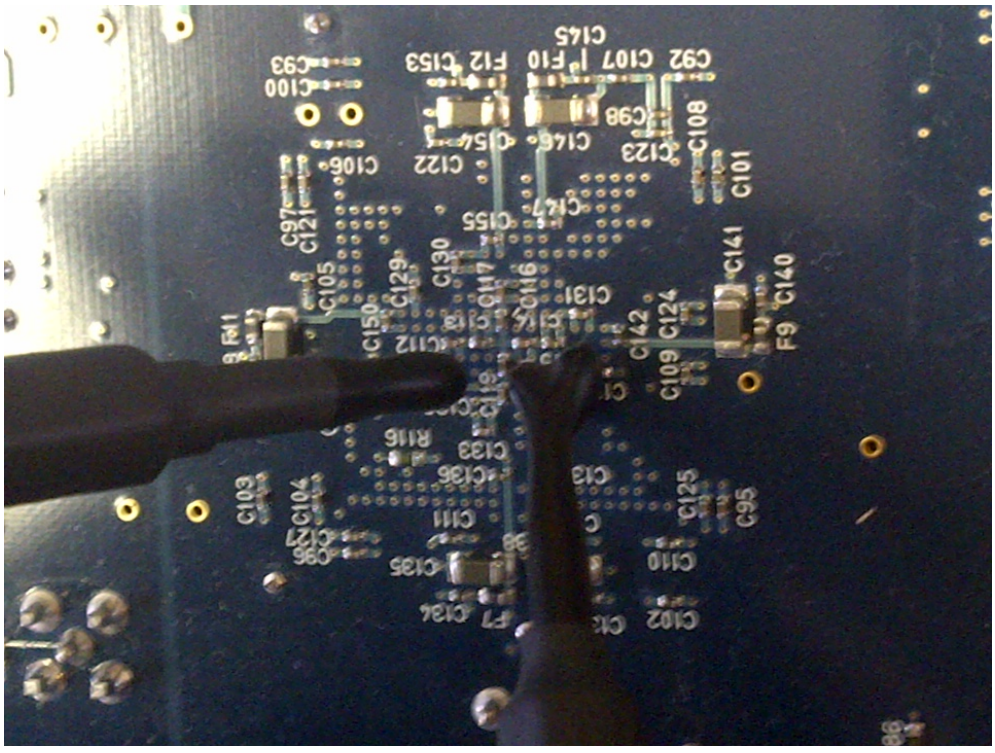


Figure 15: Placement d'antennes pour l'attaque combinée.

tion qui retourne une valeur unique basée sur plusieurs vecteurs de données. La fonction somme() est utilisée comme une fonction d'agrégation dans les expériences suivantes. Les

Table 5: Nombre de traces nécessaires pour attaquer C_1, C_2 et la combinaison des deux.

Sbox No.	C_1	C_2	Sum	**Gain**
0	350	432	212	39.42%
1	943	1073	750	20.46%
2	733	720	397	44.86%
3	400	980	251	37.25%
4	410	176	165	06.25%
5	320	281	270	03.96%
6	548	551	448	18.24%
7	592	192	184	04.16%

résultats sont présentés dans le tableau 5 et on peut observer que l'attaque par combinaison est meilleure que l'attaque en utilisant une seule trace; le gain varie de 3.96 à 44.86%.

Conclusions

Pour résumer, nous avons développé les points suivants dans cette thèse:

- J'ai montré que la logique "Dual Rail" a une excellente résistance contre les attaques par injection de fautes,
- J'ai proposé la logique "Dual rail" sans EPE, et BCDL comme une contre mesure DPL résistant EPE,
- J'ai proposé une architecture BCDL résistante à un déséquilibre de routage, avec un faible "fan-out".
- J'ai aussi proposé comme contre mesure aux canaux auxiliaire, une architecture "Unrolling"
- J'ai montré que le modèle stochastique sont des outils appropriés pour l'évaluation des architectures DPL.
- J'ai proposé une méthodologie pour combiner les mesures pour accélérer les attaques par canaux auxiliaires.

Comme perspectives de ces travaux, je propose une implémentation masquée de la BCDL en utilisant plusieurs bits de masque et qui basculerait entre deux réseaux. La caractérisation des fuites durant les phases de conception est aussi un sujet qui mériterait d'être

approfondi. On pourrait utiliser dans ce contexte une modélisation multivariée stochastique.

Chapter 1

Introduction

1.1 Motivation

Field programmable gate arrays (FPGA) which have significantly evolved during recent years, are capable of implementing complex system on chip (SoC) and providing high performance. Therefore FPGA are finding wide application in complex systems. The main advantages of FPGA over application specific integrated circuits (ASIC) are its reconfigurability, shorter time to market and frequent design updates. They are also economical compared to ASIC when deployed in low volume. Owing to these advantages, FPGA find application in digital signal processing, software-defined radio, medical imaging, computer vision, speech recognition and cryptography. FPGA are also used even in sensitive domains like defence and space. For security needs, SoC embeds cryptographic cores (crypto-cores) to encrypt all the communication in a SoC. Also in applications like network routers, FPGA are deployed to encrypt the data packets before sending them at a high speed. These communications are as secure as the crypto-cores which are generally computationally secure however physical implementations can leak sensitive information.

Several successful attacks against such cores have been put forward which target their physical implementation. The first physical attack to be published was the “timing attack”, presented by Kocher et al. in 1996 [1]. In this attack, an adversary is able to recover a secret key employed in a signature algorithm by spying on the execution time of various operations. This attack is an example of “side-channel attack” as it is completely passive: the attacked system is not aware of the threat. Other side-channel attacks spying on target properties like power consumption and electromagnetic radiation have been reported since then [2, 3], and their study has mobilized many researchers. Such attacks unfold in two stages: side-channel information collection and side-channel analysis (often abridged

SCA). Side-channel collection is a "metrology" step, whereas SCA requires sophisticated theoretical tools to be efficient. Both aspects are advancing rapidly, as attested for instance by the "DPA contest" competitions [4]. Another class of physical attacks known as active or "fault attack" (FA) works by altering the functional behavior of the attacked device by perturbations [5, 6].

Since then many countermeasures have been devised to secure physical implementation of crypto-cores. The primary aim of these countermeasures is to remove any dependency between internal operations of a crypto-core and the corresponding leakages. Most, if not all, countermeasures use redundancy to strengthen the crypto-core against SCA and FA. These countermeasures can be widely classified into two categories:

1. **information masking** [7, Chap. 9], which aims at randomizing the side-channel leakage, and

2. **information hiding** [7, Chap. 7], which aims at hiding the secret information present in the side-channel.

Information masking based countermeasures use a randomly generated value called "mask" in a way that side-channel leakage is corresponding to the mask without affecting the value of the output ciphertext. Proper implementation of masking countermeasures is secure against first order SCA but higher-order SCA might be successful. Information hiding countermeasures, as the name suggest, try to hide the information from an attacker. Resistance of these countermeasures against FA has not been studied deeply.

In the presented work, focus is laid upon the study of information hiding countermeasures which can be implemented on commercial FPGA. Our objective consists in studying the rationale of these countermeasures, their efficient FPGA implementation along with their security analysis against state of the art SCA and FA. The information hiding countermeasures which I focus on are **loop unrolling** of crypto-cores and **dual-rail precharge logic (DPL)**.

1.2 Organization

This thesis is organised as follows:

Chapter 2 gives a general background about cryptography, common cryptographic algorithms, physical attacks on cryptography and corresponding countermeasures. A short introduction to FPGA is also given.

Chapter 3 is divided into two parts. The first part discusses improving the robustness of crypto-cores by proper implementation of some algorithm components. Different implementation of a non-linear component (substitution box) of an AES crypto-processor are compared in terms of implementation cost and security evaluation against SCA and FA. In the second part, I propose an information hiding countermeasure called "Unrolling" which consists of loop unrolling of the cryptographic algorithm. This countermeasure has been implemented on a DES crypto-processor and tested on a FPGA as well as an ASIC. Detailed security evaluation of the ASIC implementation supports this countermeasure.

In Chapter 4, another information hiding countermeasure called dual-rail precharge logic (DPL) is studied. I first present the basic principles of DPL in detail and FPGA implementation of the wave dynamic differential logic (WDDL) [8]: one of the first DPL countermeasure well suited for FPGA. Then, I analyze WDDL against attacks with special focus on FA. This is followed by a complete state of the art of DPL countermeasures for FPGA which were proposed as an improvement over WDDL. Finally, I extend findings regarding fault resistance of WDDL to other DPL in general.

Chapter 5 presents two novel DPL countermeasures for FPGA. DPL w/o EPE (early propagation effect) counters a common DPL flaw of EPE by modifying the truth table of the basic gates which can be implemented in FPGA by forcing pre-computed LUT mask. The second countermeasure is balanced-cell based dual-rail logic (BCDL). It features a global synchronization signal PRE for countering EPE and enables to use embedded memories in FPGA. I show efficient implementation of the two DPL countermeasures on FPGA. Proper evaluation of these countermeasures is provided to demonstrate their strength in terms of security gain.

In Chapter 6, I propose two evaluation techniques. The first evaluation technique applies only to DPL implementations where I use profiled attacks like templates and higher order stochastic models for precise evaluation. The interest of using these attacks are two fold: precise estimation of leakage and pin-pointing the source of leakage. These methods are tested in a simulated environment and on real traces from FPGA implementation of DPL w/o EPE. The second evaluation technique is called combined attacks. The principle of this attack is that multiple measurements of the same activity can be combined to accelerate the attacks on a given implementation. Theoretical background along with practical application is given.

Finally Chapter 7 gives general conclusions and perspectives for future research.

1.3 Contributions

The main contributions of this work are:

1. Comparison of various s-box architectures in an AES co-processor against setup time violation faults [A].
2. Analysis of WDDL countermeasure against setup time violation faults [B] followed by extension of results to DPL in general [C,D].
3. Proposing unrolling of cryptographic algorithms as a side channel countermeasure [E].
4. A novel DPL countermeasure (DPL w/o EPE) for FPGA capable to countering the EPE flaw [F].
5. A novel DPL countermeasure (BCDL)[G] with its efficient implementation on FPGA to counter EPE and technological imbalance [H].
6. Application of advanced evaluation tools like templates and higher order stochastic models for efficient DPL evaluation and implementation [I].
7. A technique to combine multiple measurements of the same activity to accelerate side channel attacks [J].

Chapter 2

General Background

"Cryptology" stems from the Greek root crypto that means "hiding". Cryptology includes two branches "cryptography" and "cryptanalysis". Cryptography is the study of techniques to protect sensitive information from third parties by encoding it into an unreadable form. It is used to transform legible information (plaintext) into a protected form (ciphertext) with the help of secret information (key). According to Kerckhoffs's principle, any information related to cryptographic system can be public except the key. Cryptanalysis comprise of all the methods which can be deployed to obtain the plaintext or the decryption function in a cryptosystem by eavesdropping into the insecure channel.

Cryptography was used throughout the history. Secrecy was used by individuals and governments to hide the true intentions and gain competitive advantage. The first recorded use of cryptography in writing dates back to 2000 BC when the Egyptian used nonstandard hieroglyphs in an inscription. Hieroglyphics were used mainly in the tombs of the pharaohs to tell the story of the life of the deceased, however the intentions was not to hide information. Around 400 BC, the second clue about using cryptography was discovered in Greece. Greeks used *scytale code*, to encrypt information, and that is to write a message on a sheet of papyrus, which was wrapped around a rod. The only way to read it to find a rod with exact dimensions as the source. Julius Caesar used shifting the letters of the alphabet for military purposes.

In the twentieth century, cryptography played a crucial role in the outcome of both world wars. It was widely used in communication systems related to military, diplomatic service and government in general. The overall goal was primarily to protect state secrets and strategies. Indeed, during the World War II, modern cryptographic engines were designed for tactical communications, which greatly improved technology promotion, such as the telegraph and radio communications. For example, the so-called *rotary encoder*, is

an electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code. It was a huge breakthrough in military cryptography. This machine was considered unbreakable for a significant period of time. Moreover, it was the source of the most famous cipher machine in history, designed by the German engineer Arthur Scherbius called *Enigma machine*, which consists of three rotors. Although the Enigma machine was technically difficult for the time, the system was broken by the Polish Cipher Bureau. Broken code was the shared with the British military intelligence, which was used against an attack strategy in Germany and military traffic. It has played a very important role during World War II, and often reported in the literature. After the war, details of Enigma machines are published in [9].

2.1 Modern Cryptography

From the birth of computers and new communications systems in the 1960s, the ability to encrypt using advanced engines expanded exponentially. Computers have provided system designers with an excellent opportunity to improve cryptography. Moreover, there was an increased need for cryptography in the sector of computer technology to provide digital information and ensuring security. In the literature, this new era of cryptography is often referred to as *Modern Cryptography*. Modern cryptography is a mixture of security engineering and mathematics to form the basis of security.

The most famous project in cryptography was developed by IBM and called *Lucifer*. Lucifer was based on complex mathematical functions, which were subsequently adopted and modified by U.S. National Security Agency (NSA), to become the U.S. Data Encryption Standard (DES), called *secret-key cryptography* (SKC). SKC uses one key for encryption and decryption. Thereafter, data encryption standard (DES [10]) was formally adopted by the U.S. Federal government as a standard in 1977. DES is still used for securing e-commerce applications in many financial institutions. Diffie and Hellman also made a major contribution to cryptography in 1976. Their work has presented the basic concepts of *Public-key cryptography* (PKC [11]). PKC uses one key for encryption and another for decryption, and also provided an innovative technique for key exchange. The idea behind this technique was based on unresolved discrete logarithm problem. Even if the authors did not provide a practical implementation of the proposed encryption scheme, as it was impossible to realize at that time, the idea was clear, and it was a source of extensive research in cryptography community.

In 1978, the three cryptographers Rivest, Shamir and Adleman developed the first practical public-key encryption and signature scheme, called RSA [12]. Difficulty of factoring the large numbers was a major mathematical problem, which was used in RSA scheme. Since then, extensive research in both the private key and public key encryption is being conducted.

The so-called 3-DES was the successor to DES. This new algorithm is basically starting from DES, but uses three secret keys for encryption and decryption. Since DES was broken in early 1990s 3-DES was put in place. 3-DES was considered safe and accepted by many financial institutions around the world. In the late 1990s, NIST decided the new standard, Advanced Encryption Standard (AES [13]). AES is an efficient secret-key algorithm with more complex mathematical functions, providing a higher level of security. AES has a lot of options, depending on the *key-space*, which is the range of values that can be used as key. The larger the key space, more available options may be used to represent the different keys, and the more difficult for attackers to find the secret key.

As for public-key encryption, the existing mechanisms are still evolving. In fact, another class of powerful and practical public-key scheme, also based on the discrete logarithm problem, was developed in 1985, by El-Gamal. In principle, one of the most interesting application of the public key cryptography is the *Digital Signature Algorithm* (DSA). In 1991, the first international standard for digital signatures based on the RSA public key scheme, was formally adopted. In 1994, the new standard, based on the ElGamal public key system was adopted by the U.S. Government. Successor to such plans is *elliptic curve DSA* (ECDSA). In fact, *Elliptic Curve Cryptography* (ECC [14]) provides many features like RSA, such as encryption, digital signatures and secure key distribution.

Another type of encryption introduced recently is *one-way cryptography*. The algorithms are based on one side functions known as *hash function*. They are mainly used to identify the various parties in communication and find wide application in digital signatures. Following provides a detailed outlook on SKC and PKC.

2.2 Symmetric Key Cryptography

Secret-key cryptography is where a single key is shared between the transmitter and receiver. This means that the same key is used for encryption and decryption. For this reason, secret-key cryptography is also called *symmetric key cryptography*. Obviously, the major difficulty of symmetric encryption is the distribution of the secret key that must still be secret,

known only by the sender and receiver. Secret key cryptography is further classified into *stream ciphers* and *block cipher*.

2.2.0.1 Stream Ciphers

Stream ciphers are an important class of encryption algorithms. It is a symmetric key cipher where message (plaintext) is combined with a pseudorandom keystream. Some ciphers use what is known as the *key-stream generator*, leading to a bit stream which is XORed [1] with the plaintext to produce a ciphertext [15]. In the open literature, there are two common stream ciphers used in practice:

- *Self-synchronizing stream ciphers* that computes a key-stream according to few previous ciphertexts. A common problem with encryption is the propagation of errors. Indeed, a slight modification in transmission will result in n bit-change at reception.
- *Synchronous stream ciphers* produces the key stream in a manner independent of the message. However, the same function key stream generation is used for transmission and reception. Such stream ciphers are not affected by transmission errors.

Evaluation of the effectiveness of stream ciphers is done on three grounds: First, no repeating patterns within the main-stream values should exist at least for long periods, second, the key-stream cannot be linearly related to the key and the third, key-stream should be statistically unpredictable. From the perspective of design, stream ciphers are suitable for hardware implementations, and we can expect an increasing use of such ciphers in the coming years.

2.2.0.2 Block Ciphers

Basically, block cipher algorithms are used for encryption and decryption. They aim to divide a message into blocks of bits, which are then processed by several mathematical functions such as substitution and transposition. In fact, block-cipher processes the plaintext into blocks of bits relatively large (i.e. 64 bits, 128 bits ...). Same function is applied to encrypt successive blocks, thus block ciphers are memory-less. In contrast, stream ciphers, when using the encryption function may vary according to the plaintext, which require a large memory. Block ciphers are strong and difficult to break mathematically. Block ciphers are used in five common modes of operations.

[1] XOR stands for exclusive OR

- **Electronic Code Book (ECB) mode** The same block of ciphertext is always generated for a given block of message and key. ECB mode is often used for small block sizes, such as encryption and protection of encryption keys.
- **Cipher Block Chaining (CBC) mode** Each block of plaintext is XORed with the previous encrypted block before being encrypted. In this way, each encrypted block depends on all the blocks processed up to that point. Also, to make each message unique, an initialization vector *IV* is used in the first block.
- **Cipher Feedback (CFB) mode** The data is encrypted in blocks smaller than the size of the initial block. CFB mode works like the self-synchronizing stream cipher.
- **Output Feedback (OFB) mode** An internal feedback mechanism is used to prevent the same plaintext block to generate the same block of ciphertext. This method is technically similar to synchronous stream ciphers.
- **Counter Mode** generates the next keystream block by encrypting successive values of a "counter". The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time.

Typically, each mode is designed to manage how a block cipher will work. The choice of a mode is essentially based on security requirements.

2.2.1 Data Encryption Standard

As presented in the FIPS standard, DES is designed to cipher and decipher blocks of data. The messaage has a block size of 64 bits while the key is 56 bits. Deciphering is done with the same key as for ciphering, but in reverse order of the ciphering process. Ciphering a block starts with an initial permutation IP followed by complex key-dependent computation and finally a permutation which is the inverse of the initial permutation IP^{-1}. Figure 2.1 shows the DES algorithms. The key-dependent computation can be defined by a function f, called the cipher function, and a function KS, called the key schedule [10].

Eavesdroppers of the cipher know the algorithm but do not have the correct key which deprive them from extracting the original data. However, anyone who does have the key and the algorithm can easily decipher the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the secret key for authorized users. A wrong key causes the ciphertext that is produced for any given set of inputs to be different. But with the increase of the computation speed in new computers,

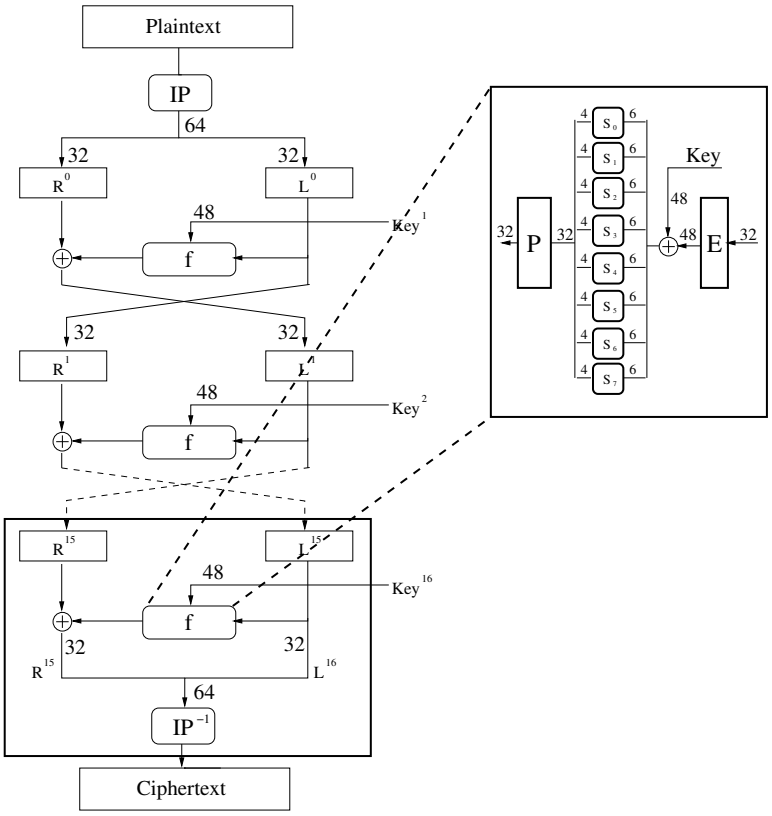


Figure 2.1: Data Encryption Standard

brute force attack have become possible (which means a trial of all possible values of the key). In fact, in June 1997 the DES was cracked by using a network of normal computers and it took 23 hours and 15 minutes [16]. Thus, an urgent need for a more robust encryption algorithm was felt. The first idea was to use 3-DES which consists in using a call of DES, then DES^{-1}, and finally DES.

2.2.2 Advanced Encryption Standard

In 2000, NIST chose the ‘‘Rijndael” algorithm as the algorithm for Advanced Encryption Standard (AES) [13]. AES is an encryption algorithm invented by Joan Daemen and Vincent Rijmen which works on block of 128 bits, and a key of variable length. The length 128, 192, 256 allows a trade off between security and efficiency (speed, computation complexity, implementation cost). AES is an iterative algorithm where the number of rounds depend on the length of the key. There are 10, 12 or 14 rounds for a 128, 192 or 256 bits of key respectively.

Furthermore, AES encryption and decryption are based on four different transformations that are performed repeatedly in a certain sequence. Each transformation maps a 128-bit input state addressed byte-wise into a 128-bit output state. The set of transformations which is repeated every time is called a round. The rounds are slightly different for encryption and decryption. These transformations are described in the pseudo-code given below.

```
Round=0;
AddRoundKey(State,RoundKey);
Round++;
while i<10 {
    Round(State,RoundKey) {
        SubBytes(State);
        ShiftRows(State);
        MixColumns(State);
        AddRoundKey(State,RoundKey);
        };
    Round++;
    };
FinalRound(State,RoundKey) {
    SubBytes(State);
    ShiftRows(State);
    AddRoundKey(State,RoundKey);
    };
```

The 128-bit data block is treated as a 4×4 array of bytes called the state matrix. The algorithm consists of an initial data/key addition,10 full rounds for a 128 bit key (12/14 rounds for 192/256 bits of key). The last round is slightly different from the other rounds. A dedicated key scheduling process is used to generate all the round keys from the input key. Each sub-key is also treated as a 4×4 array of bytes. A full Rijndael round involves four steps:

1. a non-linear substitution that is applied on each byte of the state matrix: “SubBytes”.
2. A circular bytes permutation within the same line: “ShiftRows”.
3. A multiplication in $GF(2^8)$ for each column: “MixColumns”.
4. A simple XOR with the output of the key register: “AddRoundKey”.

SubBytes Transformation

The SubBytes transformation replaces each byte in a block by its substitute from an s-box.

The s-box is an invertible substitution table which is constructed by a composition of two transformations. First, each byte $A_{i,j}$ is replaced by its reciprocal in $GF(2^8)$ polynomial (except 0, which has no reciprocal, is replaced by itself) followed by an affine transformation. The s-box is usually implemented as a look-up table consisting of 256 entries, each entry is 8 bits wide, but it also can be computed in composite fields.

ShiftRows Transformation

Next comes the ShiftRows transformation, each row in a 4×4 array of bytes of data is shifted 0, 1, 2 or 3 bytes to the left in a round fashion, producing a new 4×4 array of bytes.

MixColumns Transformation

The MixColumns transformation, operates on each column individually where each byte is mapped into a new value that is a function of all four bytes in the column. The transformation can be defined as a matrix multiplication on the state. Each column is treated as a polynomial over $GF(2^8)$ and is then multiplied modulo $x^4 + 1$ with a fixed polynomial a(x):

$$a(x) \quad = \quad (0x03)x^3 + (0x01)x^2 + (0x02)x + (0x02)$$

AddRoundKey Transformation

The final transformation is AddRoundKey, it is a bit-wise XOR of the round data with the corresponding round key for the current round.

The Key Schedule

Each round accepts a round key derived from the initial secret key by means of the Key Schedule process.

- "Rotword" operation takes a 32-bit word and rotates it by eight bits to the left.
- "Subword" operation uses s-box table to replace each byte of the columns.
- "Rcon" is a table of constants depending on the round number.

More precisely, if k^0 is the secret key and k^i is the i^{th} round key, then Key Schedule computes $k^i = KS_i(k^{i-1})$ as a function of the previous round key. The functions KS_i themselves depend on the round and on the size of the key. However, the KS_i do not differ much from each other, and for a key size of 128 bits they are all identical.

2.3 Public-Key Cryptography

Symmetric ciphers are computationally strong but key management is an important issue. The secret key needs to be securely exchanged and the encryption of the secret key is not an

option, since it would represent the same problem again and again. If the key gets leaked, then it should be revoked and a new one should be shared. In addition to this, each different secure link requires its own key: every pair of users should be assigned a unique key, known only to the authorized owners, which increases the overall number of keys exponentially with additional users. Even if the key is shared within a single group of people, there would be no way to identify correctly the sender within the group, or have a subset of authorized receivers.

Public-key (asymmetric) crypto-systems were proposed to address such issues. Each user has a pair of keys: a secret key (the private key) and a public key. It is easy to compute the public key from the private key, but the inverse is computationally infeasible. Each user has a public key which is shared with other users. Anyone can use the public key to encrypt the message but it can be decrypted only by one who posses the corresponding private key. This scheme provides some very important properties for secure communications:

- Confidentiality: It is the guarantee that the message will not be read by an unauthorized receiver; it can be achieved by encrypting the message with the public key of the receiver.
- Authentication : It is the proof of the sender's identity, certifying that the sender of the message is actually the one who claims to be, by using one's secret key to encrypt.
- Non-repudiation: It is strictly related to the previous concept and means that the sender can not deny having sent the message.
- Integrity : It guarantees that the message was not modified or tampered with, and it is exactly the message that was transmitted at the source.

2.3.1 RSA

The most common public-key crypto system is RSA [12], based on modular exponentiation in finite ring $\mathbb{Z}_n$. Encryption is computed by exponentiating the message with the public key, decryption is computed again by exponentiation with the other exponent using a secret key. Security of RSA is ensured by the problem of factoring the product of two large primes.

2.3.2 Elliptic Curve Cryptography

Recently public key algorithms have started using elliptic curves [17]. In contrast to RSA, computations take place in a finite additive group. An elliptic curve E over field **K** is defined

Algorithm 1: RSA Algorithm

1. Choose two distinct large primes p and q of the same bit length.
2. Compute $N = p.q$ as the RSA modulus.
3. Let $\varphi(N) = (p-1).(q-1)$ denote Euler's totient function.
4. Choose a public key $3 < e < \varphi(N) - 2$ coprime to $\varphi(N)$
5. Compute as the secret key the unique integer $1 < d < \varphi(N)$ such that $e.d = 1 \ mod \ \varphi(N)$.

- Encryption a message $M \in \mathbb{Z}_n$ Compute $C = M^e \ mod \ N$.
- Decrypting a ciphertext $C \in \mathbb{Z}_n$ Compute $M = C^d \ mod \ N$.
- Signature of a message $M \in \mathbb{Z}_n$, the signature S is created as $S = M^d \ mod \ N$.
- Verification of a signature $S \in \mathbb{Z}_n$ of a message M $S^e = M \ mod \ N$.

by the Weierstrass equation: $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$. The set of points (x,y) $\in K^2$ as a solution of the equation E, together with the point at infinity O form and additive abelian group. The point O is the neutral element of the group denoted as E($\mathbb{F}_p$). The group operation is called addition for two distinct points and doubling for the same point. An elliptic curve group operation consists of many field operations. For a field $\mathbb{F}_p$ with a characteristic other than 2 the equation E can be simplified to $E : y^2 = x^3 + ax + b \ \ a, b \in K$. In order to encrypt message using ECC we have to chose an elliptic curve E defined over a prime field $\mathbb{F}_p$ such that the order of E is divisible by a large prime q, then we choose a base point P on E of order q. Obviously the choice of E and P is crucial for the security of the system. The order of the base point P must be a large prime number.

2.4 Physical Cryptanalysis

Cryptanalysis is concerned with breaking the cryptographic systems. Cryptanalysis involves the study of crypto-systems in order to find weaknesses in them that will permit retrieval of plaintext from the ciphertext, without necessarily knowing the key or the algorithm. In other words, any method that can reduce the number of ciphered samples needed to retrieve a hidden key, compared to brute force is considered as a cryptanalytic method. Cryptanalysis can be widely classified in two categories: mathematical and physical. Modern cryp-

Algorithm 2: ECC Algorithm

1. Choose an elliptic curve **E** defined over $\mathbb{F}_p$.
2. Choose a public base point **P** on **E** of order q.
3. Choose the secret key k $\in \{1, 2, ..., q-1\}$
4. Compute public key **Q**=k.**P** on **E**.

Encryption

- Choose a random session key r $\in \{1, 2, ..., q-1\}$
- Compute the two points **R**=r.**P**=(x_1, y_1) and r.**Q**=(x_2, y_2).
- Compute $C = x_2 + M$ where $M \in \mathbb{Z}_p$ is the message to be encrypted.
- Send out (**R**,C) =(x_1, y_1, C)

Decryption

- Compute $(x_2', y_2') = k.\mathbf{R}$
- Recover $M = C - x_2'$

tographic algorithm are secure against mathematical cryptanalysis however physical cryptanalysis is still a major issue. Basically it is the physical implementation of a cryptographic system which leaks some information about the sensitive variables in one way or the other. In physical cryptanalysis we try to find these techniques and use its power to analyze cryptographic implementation. Physical cryptanalysis can be realized by means of fault attacks (FA) and side channel attacks (SCA).

2.4.1 Fault Attacks

Fault attacks also known as perturbation attacks are a powerful mean for physical cryptanalysis. This class of attacks relies on perturbing a cryptographic operation in order to force a faulty behaviour. Many techniques exist which are capable of exploiting this faulty behaviour to retrieve some information about the cryptographic operation. Basically, FA exploit the physical properties of devices. Fault injection and fault attack are two interesting research directions related to FA. Numerous methods to inject faults in a circuit have been proposed by Bar-El et al. [18].

Power spikes is a commonly used low-cost technique to inject faults. Short duration massive variations of the power supply, which are called spikes, can be used to inject errors into the computation of a device. The error are caused because the device is operated outside the specified rating. Variations in supply voltage during execution may cause a processor to misinterpret or skip instructions. These methods do not require a modification of the device itself but provoke faults by modifying the working conditions.

Clock glitches, like power spikes are low-cost fault induction method which relies on tampering of the clock signal. Clock-signal glitches are the simplest and most practical attacks. Details about glitches can be found in [19, 20, 21]. A clock glitch will reduce the period of one cycle significantly shorter than normal period. This reduced period will cause some erroneous computation which can be used as faults.

Optical faults are an effective way to inject targeted faults but its implementation cost is high. A laser cutter (red or green laser) or focused UV light can be used to change the state of internal signals (transient faults) or even destroy them (permanent faults) in a target device as shown by Kuhn et al. [22]. This allows to inject a great variety of faults. Memory cells used for EEPROM memory and semiconductor transistors are found to be sensitive to coherent light, i.e lasers, and ionizing radiation such as cosmic rays due to photoelectric effects.

Electromagnetic perturbations as shown by Quisquater et al. [23] can be used to inject faults by changing the external electrical field in an electronic device [23]. Here, faults are injected by placing the device in an electromagnetic field, which may influence the transistors and memory cells. However, the main problem using such an approach is to target specific bits. Eddy currents produced by passing alternating current in a passive probing antenna can be strong enough to interfere with the operation of a transistor or memory block.

Several other methods along with the ones mentioned before are used for fault induction. The choice of method is motivated by the cost of implementation and the type of fault required (i.e. fault model). Some fault analysis might need to fault a specific bit or byte which can be done by expensive methods like lasers. On the other hand, some analysis require random faults without strict localization constraint, then power spikes or clock glitches can be used. Faults model is a set of different parameters like the kind of fault (transient vs permanent), affected bits, exploitable or not, duration, location, the time when the fault is applied etc. We propose three main types of parameters which define the faults models:

- **Spatial parameters:** The stress is applied on the whole device “global faults” or a small region “local faults”.

- **Temporal parameters:** The time when the fault occurs can be fully synchronized or completely random.
- **Number of affected signals:** We differentiate between a "single faulty bit","few faulty bits" e.g. in the same Byte "single faulty byte" or random number of faulty bits"multiple faults".
- **Nature of fault:** Is the fault a transient or a permanent fault.

A generic strategy to attack a cryptographic implementation by fault was proposed by Biham et al. and is called DFA (Differential Fault Analysis [6]). The idea is to run a operation on the target device twice with the same input: one without faults and other faulted. From the knowledge of one or multiple couples {correct ciphertext, faulted ciphertext}, some hypotheses on the secret key can be discarded. This kind of attack represents a real threat for the implementation of cryptographic algorithms. Many methods to perform DFA have been proposed in literature which differ from one algorithm to other. It also differs from localization and number of faults in the same algorithm. Some of the common DFA proposed for AES are [24, 25]. In this thesis, we use the DFA proposed by Piret and Quisquater in [26] and therefore we detail this attack.

2.4.1.1 Piret and Quisquater DFA

A common DFA against AES was proposed in [26] which involves injecting the fault between the last and the penultimate MixColumns. With such faults it is possible to obtain a set of candidates for 4 key bytes. If a couple of faults are well located a unique correct candidate can be retrieved.

In this attack the error is injected before the penultimate MixColumns. This way the subsequent MixColumns spreads the faults over the column, thus affecting 4 bytes as shown in Figure 2.2. Further the non-linear substitution layer (SubBytes) is computed, followed by ShiftRows operation. At this point, there are 4 corrupted bytes scattered over different columns. Again, the last MixColumns finally spreads the 4 errors over the whole state, thus infecting all 16 bytes. Such scenario can be exploited to retrieve the whole AES-128 key with only a couple of faults using the algorithm 3. Further improvements to this attacks are proposed by Takahashi et al. in [27].

An error in round 8 yield to four errors in round 9, thus two well located faults are required to recover the whole key. The propagation of the faults in round 8 is shown in Figure 2.3. Although the number of faults are four times high in round 9 but the advantage is

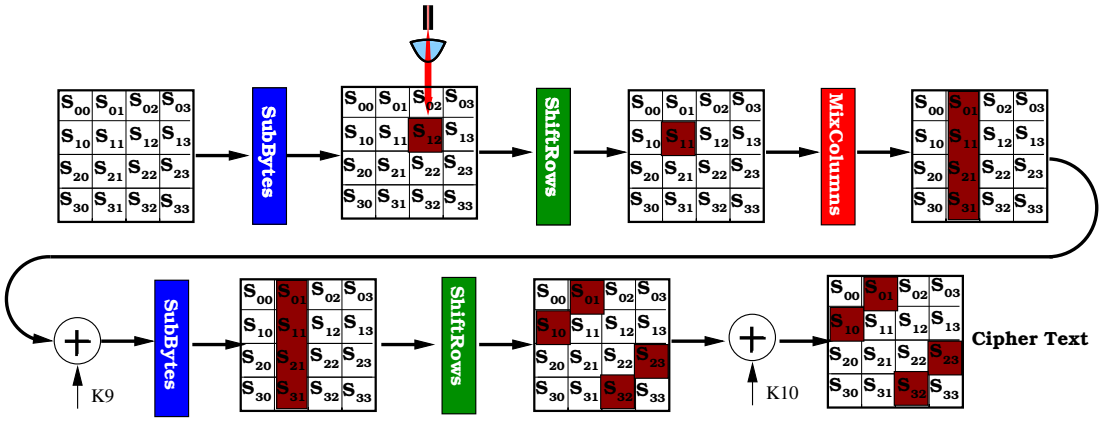


Figure 2.2: Fault effect on round 9 of AES.

Algorithm 3: Piret's DFA Algorithm.

1. Prepare a list L_D that contain 1020 (255×4) different possibility of Mixcolumn of the Round 9.
2. Make an exhaustive search on the $K^{Nround}_{0,d}$, $K^{Nround}_{1,(1-d)mod[4]}$, $K^{Nround}_{2,(2-d)mod[4]}$ and $K^{Nround}_{3,(3-d)mod[4]}$.
3. Compute Δ_t =SubBytes^{-1}((C$\oplus K^{Nround})_{*,d}$) $\oplus$ $SubBytes^{-1}((D \oplus K^{Nround})_{*,d})$.
4. Verify if $\Delta_t \in L_D$.
5. If yes we will add the four bytes of the key to the list C_d of the potential candidates.
6. Return to the second step with another pair of fault until we get only one candidate.

that they can be detected just by observing the ciphertext. A fault in round 8 will need a detailed analysis.

2.4.2 Side Channel Attacks

Side channel attacks (SCA) or observation attacks target directly the physical implementation of a cryptographic system in order to retrieve its secret key. These attacks observe some information leaked in the side channel of the target device and try to exploit it. These side channel leakage can be in terms of timing, power consumption, electromagnetic (EM) radiation. These methods are powerful because they reduce the complexity of recovering the key by several orders of magnitude as compared to brute force.

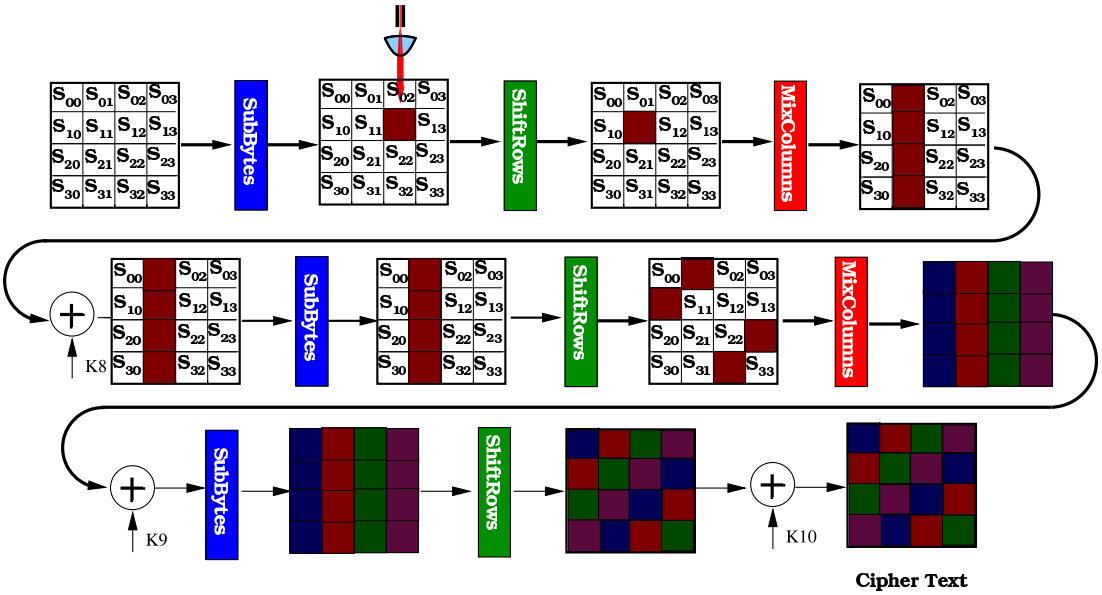


Figure 2.3: Fault effect on round 8 of AES .

2.4.2.1 Timing Attacks

The timing attacks were introduced in [1] by Kocher et al. The basis of timing attacks is that different operations may take different execution time. Thus by observing the execution time of an operation an attacker can have an idea about the secret processed. For example, in case of RSA the circuit will perform a square operation when the key bit is 0 else a square operation followed by a multiplication. Obviously, the execution time of square and multiply is more than just the square operation. Thus the execution time directly reveals the key bits. In other cases, the leakage might not be the exact key bits but a more complex key-dependent function.

2.4.2.2 Power Attacks

Electronic devices are made up of CMOS cells. A CMOS cell as shown in Figure 2.4 derives current from a constant power source V_{DD}. The cell consumes significant amount of power when either transition occurs i.e. $0 \rightarrow 1$ or $1 \rightarrow 0$.

The basic building block of any digital circuit is a CMOS cell. As the power consumption of a CMOS cell is transition dependent, observing the power consumption can reveal significant information about the data processed by the device. This is the principle for power analysis. An attacker collect a large set of power consumption traces each time varying the

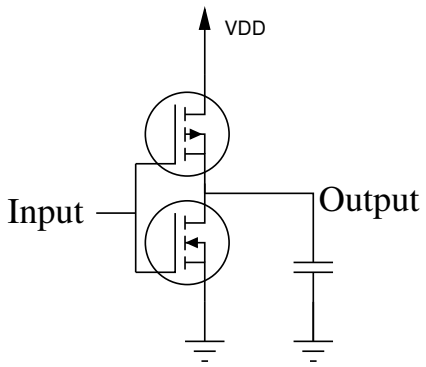


Figure 2.4: Basic CMOS inverter.

plaintext. Thereafter using an appropriate power consumption model and a distinguisher an attacker can extract the correct key candidate. Power consumption model and distinguisher are discussed later.

2.4.2.3 Electromagnetic Attacks

Electromagnetic (EM) attacks are similar to power attacks but the difference lies in method of collecting the leakage. Power attacks observe the power consumption of the device which is a global activity. EM attacks can exploit very localized information. Such attacks generally capture the EM radiation from a device, which depending on the used antenna could be global or localised. When localised radiations are captured, the traces are less noisy and easier to attack. Such attacks are of special interest in case of hardware implementation where large amount of data is processed in parallel and the algorithmic noise is huge. Localised EM can help isolate one operation from another.

2.5 Side Channel Attack Model & Distinguisher

2.5.1 Leakage Model

A basic requirement to mount a power/EM attack on a device is to find an appropriate leakage model. The attack consists of finding a relation between the actual power consumption and the estimation. These estimations are based on the leakage model. The power consumption can be seen as the sum of a deterministic part (depending of data handled) and a random part (or noise). In hardware, power consumption comes mostly from the change of state on each bit in the register. Thus, power consumption can be modeled as follows:

$$C(t) = \sum_{i=1}^{n} H(A_i \oplus B_i) + \mathscr{B}(t)$$

where A_i and B_i are the initial and final state of the i_{th} register in the hardware, n is equal to the length of the register and where $\mathscr{B}$ is a noise, which is supposed to be Gaussian most of time. This model is called the Hamming distance model, considers that the consumption of a 0 to 1 transition is the same as 1 to 0 transition. This consumption is the same for all the bits of the memory. If the device initializes the registers to zero or some countermeasure is deployed which adds a 0 spacer between two computation, the model is not Hamming distance but Hamming weight given by:

$$= C(t) = \sum_{i=1}^{n} H(A_i) + \mathscr{B}(t)$$

Another leakage model used sometimes is the switching distance model. This model is similar to the Hamming distance model but it differentiates a 0 to 1 transition from a 1 to 0 transition [28].

Once the appropriate leakage model is chosen, the next step is to choose a distinguisher. The function of a distinguisher is to detect the dependency between the leakage model and actual side channel information. An efficient distinguisher can differentiate the correct key hypothesis from all the wrong ones with a minimum possible number of traces. In what follows we describe some of the commonly used distinguishers.

2.5.2 Simple Power Analysis

Simple power attack (SPA) aims to recover the secret key by using only a few power consumptions curves. This attack is carried on observation rather than statistical distinguishers. A demonstration of SPA was first given by Kocher et al. [1] on RSA. A single power consumption trace of a naive RSA implementation can reveal the key as power consumption of a square operation is different from square and multiply (Figure 2.5). Apart from revealing key, SPA can aid other attacks in localizing the leaking point etc. However SPA is easy to protect using simple countermeasures like constant time execution. In such cases, more complex distinguisher are used which are generally based on statistical computation like differential power analysis (DPA), correlation power analysis (CPA) and mutual information analysis (MIA).

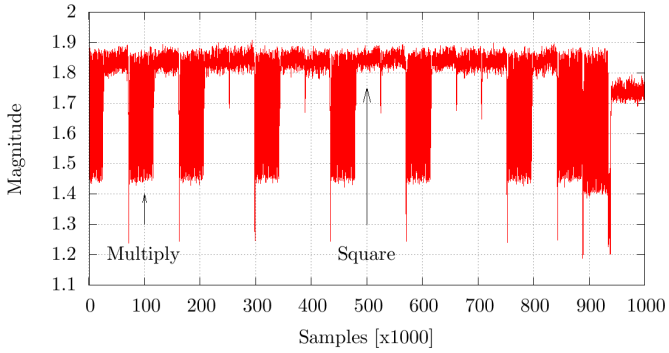


Figure 2.5: SPA on RSA implementation

2.5.3 Differential Power Analysis

Differential power analysis or DPA was introduced by Kocher et al. [2]. The authors introduced a distinguisher called *Difference of Means (DoM)*, which basically involves two partitions based on a single bit activity. The computation of DoM distinguisher is simple and can be expressed as:

$$DoM : \Delta = \mu_1 - \mu_2. \tag{2.1}$$

where μ_1 and μ_2 are the averaged traces of first partition and second partition, respectively. For false key hypotheses the partitioning is more or less random and the differential trace is nearly zero. The secret key can be consequently identified as the one that yields the highest peak in the differential trace. Later this distinguisher was further extended to multi-bit target and was called multibit-DPA. It considers the activity of multiple bits of the target and computes a weighted DoM. It computes centered weights of the considered partitions and calculate the sum over (centered) partitions. In other words multi-bit DPA can be computed using a simple *covariance* between the leakage l and leakage model h and expressed as:

$$DPA : \Delta = Cov(l, h). \tag{2.2}$$

The basic algorithm of a mono-bit DPA attack is described in Algorithm 4.

2.5.4 Correlation Power Analysis

Correlation power analysis (CPA [3]) as proposed by Clavier et al. is the normalized version of DPA. The main advantage of normalization is that it can reduce the noise which affects

Algorithm 4: DPA Algorithm

1. Encrypt many randomly selected plaintext "M"
2. Collect their corresponding power traces "T"
3. Choose selection function "L"
4. Make guess on the key "K"

For (i = 0 to key length)
Collect the average power trace: $\mu(\{L_i(M,K)=0\})$
Collect the average power trace: $\mu(\{L_i(M,K)=1\})$
Compute the differential trace: $D_i = \mu(\{L_i(M,K)=1\}) - \mu(\{L_i(M,K)=0\})$
If D_i has a spike
$K_i = 1$
Else
$K_i = 0$
End For
Output: Cipherkey

the collected traces. CPA is a computation of the *Pearson Correlation Coefficient*ρ between the side channel leakage l and the leakage model h; and generally has the following form:

$$CPA : \rho = \frac{Cov(l,h)}{\sigma_h.\sigma_l}. \tag{2.3}$$

Where σ_l and σ_h are the standard deviations of obtained physical leakages and modelled leakage, respectively.

2.5.5 Mutual Information Analysis

Mutual Information Analysis (MIA [29]) was introduced by Gierlichs at al. as a generic side channel distinguisher using an information theoretic approach. The advantage of MIA over CPA is that MIA is capable of detecting even non-linear dependency between a side channel leakage and leakage model while CPA performs best when the dependency is linear. Therefore MIA extracts the value of the secret key with more flexibility. The *mutual information* is measured in bits and computed between the global observation O (*i.e.* the set of traces acquired) and the leakage model L. L corresponds to leakages partitions involving the couple (l,h).

$$MI(O;L) = \sum_o \sum_l P(o,l) \log \frac{P(o,l)}{P(o)P(l)}. \tag{2.4}$$

$$MI(O;L) = H(O) - H(O|L). \tag{2.5}$$

where l and o are realizations of L and O respectively, $H(O)$ is an estimation of the entropy of O, $P(o,l)$ is the joint probability density function of O and L, $P(o)$ is the marginal probability density function of O and $H(O|L)$ is the conditional entropy of O knowing L. $MI(O;L)$ can be regarded as a positive (*i.e.* $MI(O;L) \geq 0$) and symmetric (*i.e.* $MI(O;L) = MI(L;O)$) measure of the strength of a 2-way interaction between two variables: the observation O and the leakage L that is related to the secret key. But more importantly, the higher the value of the mutual information is, the higher the dependency between O and L is. $MI(O;L) = 0$ when O and L are independent random variables.

The estimation of entropy is a real challenge in calculation of MIA. Several methods to estimate entropy have been compared by Prouff et al. like histograms, kernel density functions, Gaussian parametric estimators *etc.* [30]. In practice, the Gaussian parametric estimation, where the distribution of O, L and the joint distribution of O, L is assumed to be Gaussian, can serve usually as a first approximation for the distributions' shape. In this case entropy can be calculated as a function of standard deviation σ_o of O as:

$$H(O) = -\sum_i p(o_i)\log(p(o_i)) = \log(\sigma_o\sqrt{(2\pi e)}).$$

Moreover, under the Gaussian assumption, it can be shown that mutual information is intimately connected to the Pearson coefficient ρ and can be expressed as follows:

$$MI(O;L) = -0.5\log(1-\rho^2). \tag{2.6}$$

Apart from these distinguishers, there are two other distinguishers commonly deployed are: **template attacks** and **stochastic models**. A disadvantage of these distinguishers is that they require a profiling phase. The ability to profile is a strong assumption for an attacker as it requires access to a clone device with a known secret key or right to vary it. However these attacks are very useful for design evaluation as discussed in Chapter 6.

2.6 Need for Countermeasures

Cryptographic cores are often embedded in the SoC which encrypt/decrypt data on the system bus. Such SoC are as secure as the cryptographic cores are mathematically secure but their physical implementations can be compromised using side channel attacks (SCA) [3, 31] or fault attacks (FA) [6, 26]. Several countermeasures are deployed to protect these cryptographic cores. These countermeasures vary from simple noise generators to sophisticated

masking schemes for SCA. To resist FA, countermeasures deployed usually detect faults and signal it. Some of the commonly used countermeasures against FA and SCA are discussed in the following.

2.6.1 Countermeasures against FA

2.6.1.1 Parity

Parity is widely used in communication theory to detect and possibly correct error in transmitted data. The property of error detection can be used to detect faults in a circuit which can then correct it or stop the circuit depending on the implementation. In [32], Bertoni et al. describe a solution for low cost concurrent error detection in the substitution-permutation network like AES. The detection scheme is based on single parity bit which is propagated through the non-linear and the linear layer of the ciphers. Prediction through the linear layer can be very simple, reducing itself to a bunch of XOR as shown in Figure 2.6. The prediction for the non-linear layer "SubBytes" can be obtained by extending the tables storing the output values. This method can detect only single faults, moreover, the fault coverage is not impressive, since it works for about 96.3% single stuck-at faults and the overhead is about 18%. This countermeasure may not be considered as sufficient to protect against a malicious attacker. Multiple parity bits can be used in order to detect multiple faults [33] for an area overhead of 33% but faults of even order may still be masked with a non-negligible probability.

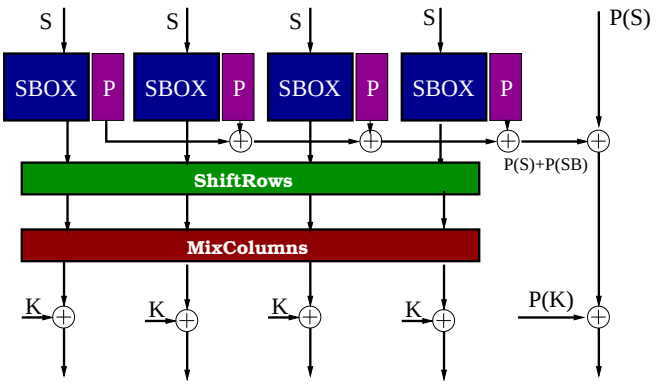


Figure 2.6: Parity based countermeasure

2.6.1.2 Concurrent Error Detection

Karri et al. [34] propose another fault detection method based on the involution property (inverse relationships that exist between encryption and decryption) to check if the condition $f^{-1}(f(x)) = x$ is respected throughout the cipher. Concurrent error detection CED can be implemented at algorithmic level, round level or operation level as shown in Figure. 2.7. The most straightforward methods of performing CED are space redundancy and time redundancy. In space redundancy, multiple copies of the hardware (generally two) are used concurrently to perform the same computation on the same data. At the end of each computation, the results are compared and any discrepancy is reported as an error. The advantage of this technique is that it has minimum error detection latency and it can detect both transient and permanent faults. A drawback of this technique is that it entails at least 100% hardware overhead. Time redundancy is achieved by using the same hardware to compute output for same input data multiple times and compare. This technique has minimum hardware overhead but it entails 100% time overhead.

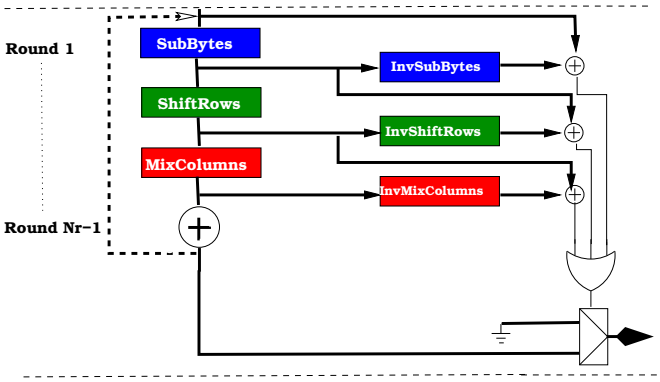


Figure 2.7: Concurrent error detection

2.6.1.3 Cyclic Redundancy Check

Cyclic redundancy check (CRC) is a more complex form of parity bit. This solution [35] by Yen et al. performs fault detection based on the CRC over $GF(2^8)$. The approach uses the linear behavior of each operation in AES to design a detection scheme. It uses a $(n+1, n)$ CRC to detect the errors, where n is $4, 8, 16$ (depending on 8-bit, 32-bit, or 128-bit architecture) and the parity of the output of each operation is predicted. Because AES is byte-oriented and its components are ingeniously designed, the parity of the output can be predicted from a linear combination of the parity of the input. In most cases, the parity is the summation of

Table 2.1: Non-linear Robust code implementation cost.

*****	**Slice**	**Frequency**	**Hardware Overhead**
Unprotected	3856	53.29 Mhz	0%
Robust Non-linear	5205	51.98 Mhz	35%

the input data; also, the schemes are highly scalable and are suitable for 8-bit, 32-bit, or 128-bit architecture. Another advantage of the proposed approach is that the parity calculation between the encryption and the decryption is symmetric because the parity generation in encryption is quite similar to the one in decryption. This is of advantage when encryption and decryption are integrated into one circuit. This method can also be used to protect the KeySchedule.

2.6.1.4 Robust Code

Robust codes were proposed in [36] by Kaprovsky et al. as a fault detection method. First of the two solutions proposed devided AES into 2 parts non-linear and linear, where the non-linear block is the s-box. In order to detect a fault, an inverse multiplication is performed at the s-box output and few bits (typically 2 or 3) of the result are checked. This method is sound because it gives a higher fault coverage at reduced area overhead. The linear layer is protected by exploiting that the sum of the bytes of a single column is not affected by the MixColumns transformation, hence a 8-bit signature is used for each column.

In their second proposal [37], a non-linear robust code is described, based on the addition of two cubic networks, computing $y(x) = x^3$ in $GF(2^8)$, to the previous linear scheme. The method allows to produce r-bit signatures as shown in Figure 2.8 to detect errors. This solution provides good error detection properties against faults of all multiplicities 2^{2-r} where r is the number of redundant bits which are added for data protection. The overall hardware overhead cost is about 35 % for a pipelined architecture of AES that was implemented on Altera Stratix FPGA. The result is shown in Table 2.1. For parallel architecture like the one presented in [37] the total area overhead is about 77%.

2.6.1.5 Double Data Rate

In [38], Maistri and Leveugle show another countermeasure against fault attacks which is similar to CED. The countermeasure is based on time redundancy using double data rate

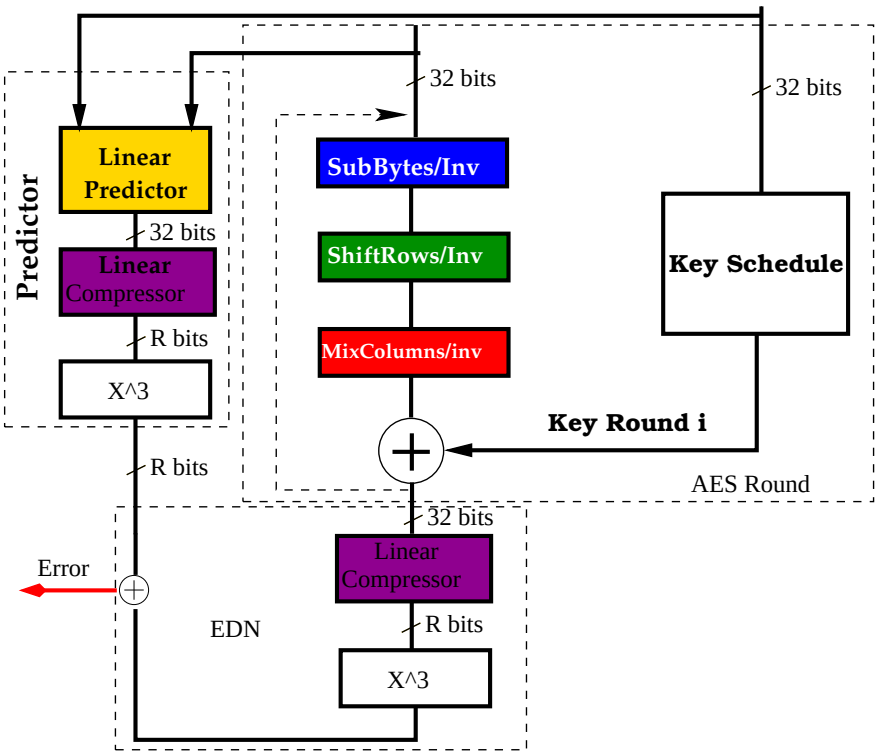


Figure 2.8: Robust code countermeasure.

computation. This countermeasure uses both the rising and the falling clock edges to compute twice the cipher text. In fact, registers are duplicated in order to create two parallel data paths, controlled by the clock edges, while the operation logic is shared between the two paths as shown in Figure 2.9. The area overhead is 36 % for a pipelined architecture and the throughput reduction is between 15% and 55%.

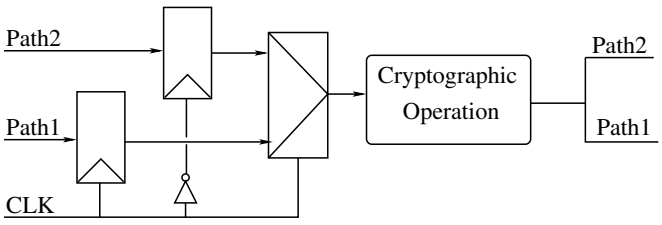


Figure 2.9: Double-Data-Rate as countermeasure

2.6.2 Countermeasures against SCA

2.6.2.1 Noise Generators

Noise generators are often used as countermeasures against SCA which can be located on-chip or off-chip. Noise generators are placed such that they introduce noise into the power consumption signal of the cryptographic cores [39]. This is best measured by signal-to-noise ratio (SNR). Here signal S is the power consumption of the cryptographic core which an attacker tries to exploit and noise N is the additive noise. Noise-generators increase the level of N and reduce the SNR. Reduction of SNR means reduction in the correlation, given by ρ, between the power consumption and corresponding key guesses. In such cases, the attacker needs to acquire more power consumption traces to mount a successful attack. With the advances in acquisition methods a large number of traces can be easily acquired in little time. Therefore noise generators are not considered a sound countermeasure.

2.6.2.2 Random Delay Insertion

Another common countermeasures against SCA is the introduction of random delays. Cryptographic algorithm execute a set a instructions in sequence. To resist SCA, dummy operations are inserted between the defined sequence of operations which means the total execution time of the algorithm will increase. Since the location and the duration of these dummy operations is random, the countermeasure is called random delay insertion. This results in shifting of the peaks across the differential trace due to a desynchronization effect which can be considered as added noise. The spike that actually appears follows a Gaussian distribution, thoroughly characterized by a mean position μ and a variance σ. Like noise generators, random delay insertion increases the number of traces required for a successful attack. A practical attack against such countermeasure is presented in [40] by Clavier et al.

2.6.2.3 Shuffling

Shuffling is similar to random delay insertion. In shuffling, instead of adding dummy cycles the sequence of operation is randomized. Randomizing the execution of the sequence of operations in an algorithm provide additional resistance against SCA. Since the number of operations is small in a cryptographic algorithm, the degree of randomization achieved by shuffling is limited. In general, shuffling and random delay insertion can be combined together to provide a higher resistance.

2.6.2.4 Key Update

Key update at regular intervals was proposed as a countermeasure by Kocher et al. [41]. The idea put forward in this countermeasure is to change the secret key used for encryption and decryption regularly. For example, if the secret key of a target can be retrieved using an attack like DPA in 1000 traces, the value of secret key should be changed before 1000 encryptions. Thus if an attacker get some information about a key, this information would be useless once the key changes. This countermeasures can be effective against SCA but FA can be a concern. In AES, where two well-located faults are enough to retrieve the secret key, key update after every encryption will be impractical. State of the art hardware cryptographic implementations often use physical unclonable functions [42] to generate keys. Key update can not be easily used in such hardware. A solution is proposed by Medwed et al. [43], which suggests use of a fixed secret master key and a session key generated from the master key for each encryption. The scheme proposed in [43] takes into account that the function chosen for generating session key is itself secure.

2.6.2.5 Data Masking

Data masking depends on hiding of internal sensitive variables x by a random mask m to avoid any dependency between the cryptographic device's power consumption and the processed data [44, 45]. This countermeasure is a dynamic research topic and is applicable to both hardware and software. The hardware design involves architecture level modification (RTL level) with less effort at the back-end stage, unlike masking at bit-level [46, 47]. The masking at word level can be vulnerable to second order attacks by Messerges [48]. These implementations could be costly and may need complex architectures in term of number of operations or memories used as look-up tables (LUT) [49, 50, 51].

The internal variable x does not exist as a net in the cryptosystem but can be reconstructed by a pair of signals $(m, x_m = x\ \gamma\ m)$ where x_m is the masked variable and γ is an operation which can be Boolean or arithmetic. The Boolean masking uses the exclusive-or (xor) operation:

$$x_m = x \oplus m,$$

whereas arithmetic masking is made with modulus operation on a finite field:

$$x_m = x + m\ (mod\ n) \qquad \text{or}$$
$$x_m = x * m\ (mod\ n) \quad .$$

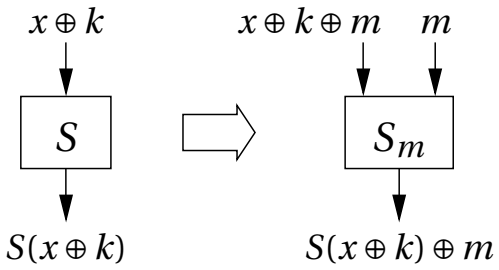


Figure 2.10: S-box for masking by software.

The mask can be applied on internal variables being words (or vectors) of cryptographic functions [44, 49, 52]. Word-level is applicable for hardware as well as software implementation. Other masking schemes use "random pre-charging" which introduce cycle circulating mask m in the design before and after using the internal variable x. This reduces the dependency between the power consumption generally described by the Hamming distance model and x [53].

The implementation of masking is simple. For a function f has the following linearity property:

$$f(x\ \theta\ m)\ =\ f(x)\ \theta\ f(m)$$

The value of $f(x)$ can be reconstructed from the application of f on $x\ \theta\ m$ and m, hence the computation of $f(x)$ can be extracted at the very end of the algorithm. This avoids a direct leakage of information as $x\ \theta\ m$ and m are decorrelated with x.

For non-linear f, the masking is difficult as $f(x)$ cannot be reconstructed mathematically from $f(x\ \theta\ m)$ and $f(m)$. In secret key encryption algorithms, the non-linear component corresponds to the s-boxes S.A common technique applied in software is to use a specific memory acting as a LUT S_m such that $S_m(x \oplus m)\ =\ S(x)\ \oplus\ m$. Consequently the size of this memory to implement the new table increases from 2^n to 2^{2n}, n being the number of bits of the mask. Figure 2.10 illustrates the change when using this masking scheme.

It can be noticed that it is not secure in hardware, because the register transfers demasks the data:

$$\underbrace{x \oplus m}_{\text{initial value}}\ \oplus \underbrace{S(x) \oplus m}_{\text{final value}} = x \oplus S(x)\,.$$

For AES, masking can take advantage of the fact that the s-boxes are calculated by using the inverse in GF(2^8) as proposed in [49]:

$$S(x) = f(x^{-1}) \text{ in GF}(2^8)\,. \tag{2.7}$$

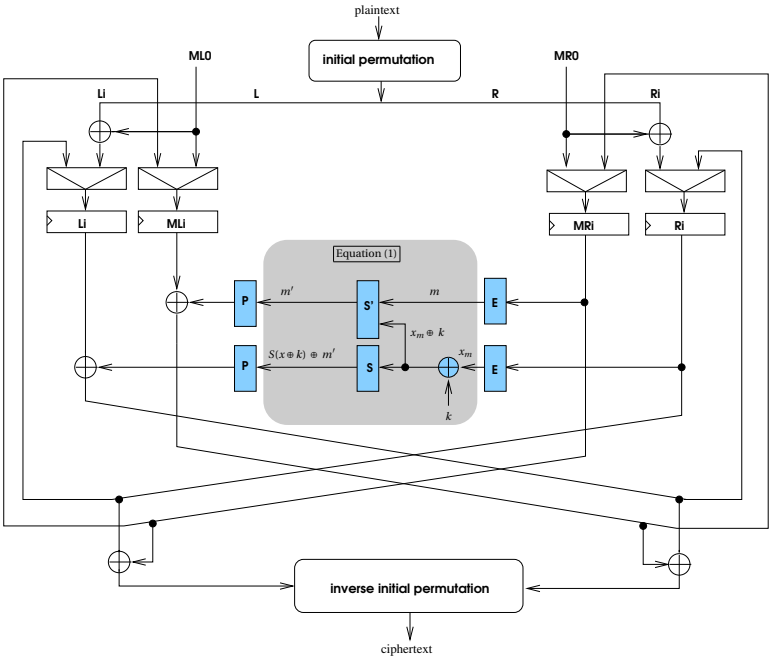


Figure 2.11: Masked DES with S-boxes S and S' in ROM.

The internal variable $S(x)$ can then be rebuilt as:

$$S(x_m) = x \cdot m^{-1} = S(x) * S(m). \tag{2.8}$$

However this implementation is very sensitive to zero-value attack [54] because the power consumption of the value 0 is easily detected. Improvements have been proposed by Oswald et al. in [50] with a slight increase of complexity.

The robustness of masking countermeasures based on Boolean operators is provably secure against first order attacks [55], a first order attack being an attack where only the variable x is considered. But masking logic could be sensitive to second order attacks, where the attack considers two variables x_1 and x_2. For instance when the same mask is applied on x_1 and x_2 and the Boolean masking is used, the second order attack takes advantage of the fact that $x_1 \oplus x_2 = (x_1 \oplus m) \oplus (x_2 \oplus m)$.

In the sequel, we take the example of a simple masked DES which was studied at UCL [56] and its principle is recalled in Figure 2.11; our variable x represents the right half of the LR register.

At each round an intermediate mask ML_i, MR_i is calculated in parallel with the intermediate cipher word L_i, R_i as shown in Figure 2.11. If we ignore the expansion E and the

permutation P, the DES round function f is implemented in a masked way by using an array of 8 functions S and another array of 8 functions S':

$$S(x_m \oplus k) = S(x \oplus m \oplus k) = S(x \oplus k) \oplus m',$$
$$m' = S'(x_m \oplus k, m) = S'(x \oplus m \oplus k, m). \tag{2.9}$$

The variable m' is a new mask reusable for the next round.

The set of functions S contains the traditional s-boxes applied on masked intermediate words. The size of each S is 64 words of 4 bits when implemented with a ROM. S' is a new table which has a much greater ROM size of $4K$ words of 4 bits, as there are two input words of 6 bits. This countermeasure is thus referred to as "zero-offset".

The classical SCA like Differential Power Analysis [57] as well as the Correlation Power Analysis [3], was unable to extract a single S-Box sub-key used by the cryptoprocessor using up to 100,000 traces. This is because the transient demasking observed in [58] occurs only in combinational logic, as opposed to ROM.

2.6.2.6 Data Hiding

Unlike data masking, the motive of data hiding countermeasures is to make the activity of the target device constant at all times. An electronic device which is composed of a CMOS cells consumes significantly more during a transition at input as compared to the case when inputs remain unchanged. This difference is exploited by SCA. A logic gate in a data hiding countermeasure like dual-rail with precharge logic (DPL) consumes almost equal power for each transition. The power consumption is uncorrelated to the data processed which removes the basis for SCA. This characteristic of constant activity makes SCA difficult to mount. A detailed explanation of the DPL rationale is given in Chapter 4.

2.7 Field Programmable Gate Arrays

The field programmable gate array (FPGA) is an integrated circuit that contains identical logic cells as standard components. Each logic cell can be independently programmed. These identical logic cells are called configurable logic blocks (CLB). Figure 2.12 shows a CLB which consists of a 4-input look-up table (LUT), to implement any four input variable Boolean function, and an edge-triggered flip-flop for sequential circuits. In fact, 4-input LUT were used as standard cells in FPGA for a long time because they provide an optimal trade-off between area required for routing resources and LUT size. Recently, advanced research in FPGA has shown a better trade-off between bigger LUT and routing resources

found in state of the art FPGA. Commercial FPGA have more complex CLB which may multiplexers, fast carry chain, register cascading, Preset/Reset functionality for flip flops, synchronous Reset etc. as compared to this basic logic block. CLB in a commercial FPGA are often organized hierarchically as clusters(see Figure 2.13) of these basic logic blocks. FPGA logic also includes hard macros such as multipliers, DSP blocks, RAM.

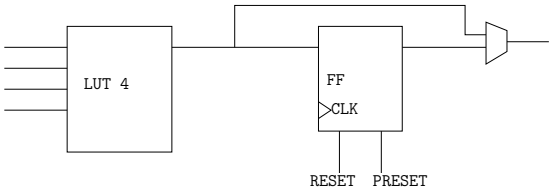


Figure 2.12: Basic logic element in an FPGA.

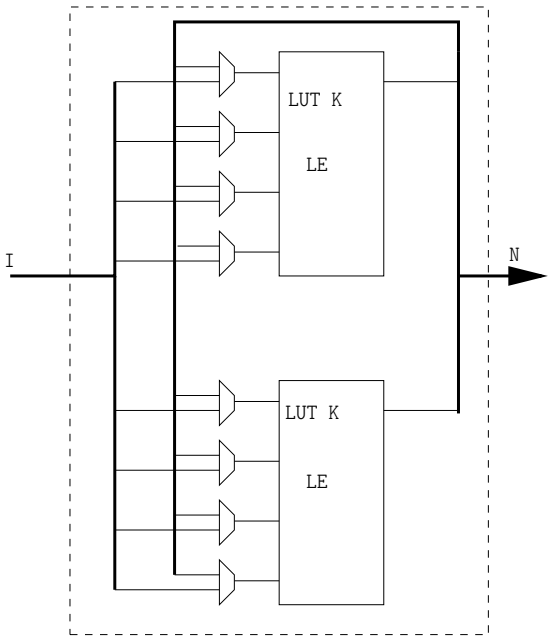


Figure 2.13: Logic cluster.

Individual CLB are interconnected by a matrix of wires and programmable switches. The design is implemented by specifying the logic function of each CLB and selectively closing the switches of the interconnection matrix. Logic cells and interconnect form a network of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit.

2.7.1 Generic FPGA Design Flow

Modern FPGAs may contain hundreds of thousands of logic elements and programmable routing switches. Therefore manual programming is not feasible which calls for design automation to map user design onto FPGAs. In Figure 2.14 the major steps in this process are illustrated. Generally the complex software which is capable of executing all these steps is provided by FPGA vendors.

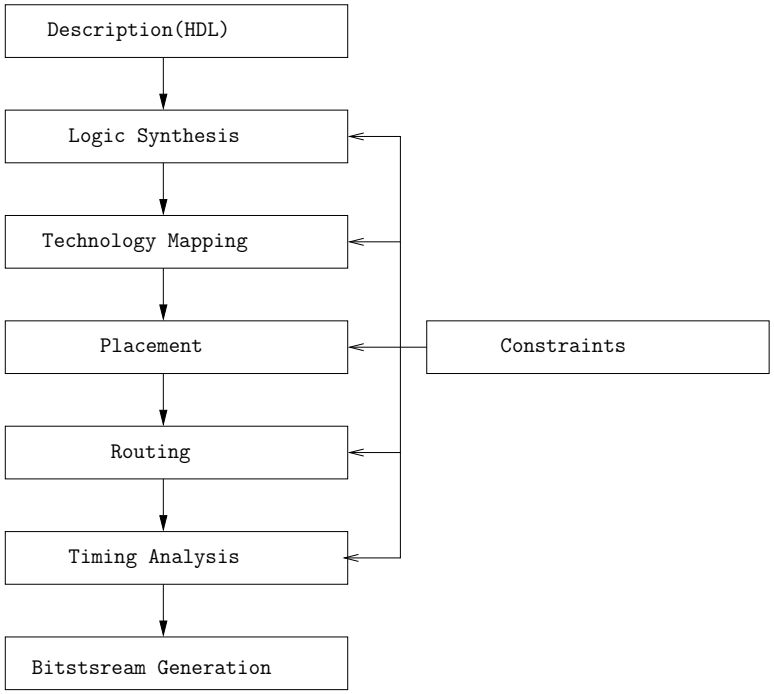


Figure 2.14: Generic FPGA CAD flow.

Any FPGA design begins with a circuit description. Designers use high-level hardware description language (HDL) like VHDL or Verilog to describe the behaviour of the design. Modern FPGA tools also provide graphical interfaces to describe the circuit using standard logic elements from a library.

Synthesis follows circuit description. In this step the software reads the circuit description and translates it further into basic logic blocks. The whole circuit is described in a set of logic equations at this step. The output of this step is called netlist. Till this step the netlist is platform independent. This step is often hidden in modern FPGA tools and merged with technology mapping.

Technology mapping converts a platform independent netlist to a platform specific netlist. Each logical equation in the generic netlist are replaced by FPGA specific logic elements (LE). These LE are equivalent to a CLB described before which differs from one FPGA to another. Therefore a technologically mapped netlist is platform specific.

Then comes placement. In this step, the CAD tool determines the placement of LE defined in the netlist on the actual FPGA fabric. Various optimizations are done in this step to find an appropriate placement scheme for the given netlist. Optimization can be done for area reduction, speed maximization, low power consumption etc.

Once the placement of LEs is done, they are connected to each other by the interconnection matrix and programmable switches. This process of choosing appropriate connection between various LEs is called routing. Optimization can also be done at this step.

Timing analysis is done at various steps of the CAD flow. Propagation delays along various paths are estimated using pre-defined models of FPGA cells in order to analyze the performance of the chip and determine parameters like maximum operating frequency, slowest path etc. Such parameters can be used to further optimize the circuit.

Finally, a bitstream is generated by the FPGA CAD tool. This bitstream contains configuration information about LE, placement, routing etc and programs the generated design on the physical fabric of FPGA. Simulations after each step are often performed to verify functional correctness of the design before and after processing.

Chapter 3

Protecting Cryptographic Circuits at the RTL Level

This chapter focuses on a few techniques to better implement crypto-processors with respect to physical security at the register transfer logic (RTL) level. The advantage of securing at the RTL level is that the design is platform independent. Data masking is a SCA countermeasure which is generally implemented at the RTL level. It is a complex countermeasure which can be implemented in various configurations. Here we focus on countermeasures simpler than masking. There are two major contributions in this chapter. The first contribution is about improving the robustness of a crypto-processor by proper implementation of some components. In fact, when few components of a cryptographic algorithm are implemented with special care, it's resistance against attacks can be significantly improved. This often applies to the non-linear part of the cryptographic algorithms. The security of the non-linear part of a cryptographic algorithm commonly known as substitution box (s-box) is important. It is also the most bulky part of the algorithm. We compare three different implementations of AES differing in the s-box architecture. The three architectures are then analyzed for cost and robustness against side channel attacks (SCA) and fault attacks (FA).

The second contribution presents a RTL level countermeasure. Standaert et al. have studied the effect of pipelining on security in [59]. Here we investigate the other trend such that all registers become unpredictable depending on the key (*i.e.* a hypothesis test involves too many key hypotheses). The idea is to implement the design in such a way that the multiple keys are used during a synchronous operation. In other words, multiple rounds of a cryptographic algorithm are executed in a single synchronous operation. We call this countermeasure as "Unrolling". This is a register transfer logic (RTL) level countermeasure which is totally platform independent. It can be classified as an information hiding countermea-

sure because the information is still present but in a hidden manner. In other words, it is very difficult to find a leakage model to exploit present information. No or very little post-synthesis processing is required. The design can easily be transferred to ASIC.

The rest of the chapter is organised as follows. Firstly, the architecture of the AES crypto-processor is presented along with three different s-box architectures. This is followed by their analysis against FA and SCA. Then we present in detail the rationale of unrolling as a countermeasure and a fully unrolled DES. Implementations in different FPGA platforms and ASIC are given followed by the evaluation of the ASIC implementation against CPA and MIA.

3.1 Protecting Cryptographic Implementations at the Component Level

Designing robust crypto-processor can be started at the RTL level. Different implementations of components of a cryptographic algorithm can vary its robustness against SCA and FA. By components we mean the sub-operations performed during a round of cryptographic algorithms. This added robustness can come from several reasons like lesser contribution to the power/EM signature, less delay etc. Please note that higher robustness means that the implementation may still be broken however more effort will be needed. Often different implementation of the non-linear component of the cryptographic algorithm offer different resistance against attacks. We present three different implementations of AES on FPGA which differ in the s-box architecture. Details of AES crypto-processor and different s-box architectures are described in the following.

3.1.1 AES Co-processor

An architecture of the unprotected AES co-processor is shown in Figure 3.1. The co-processor processes all the 128 bits of the AES datapath i.e. one round in one clock cycle (parallel architecture). Each round is comprised of four operations which are SubBytes, ShiftRows, MixColumns and AddRoundKey. Apart from these operation, hardware implementation of the datapath also contains multiplexers and key expansion unit. The key expansion unit computes a key for each round from the input key. The datapath and the control part are totally separated so that the countermeasures can be implemented only on the datapath which carries secret key related computations. The control part can be used in an unprotected mode which ultimately results in less area/cost.

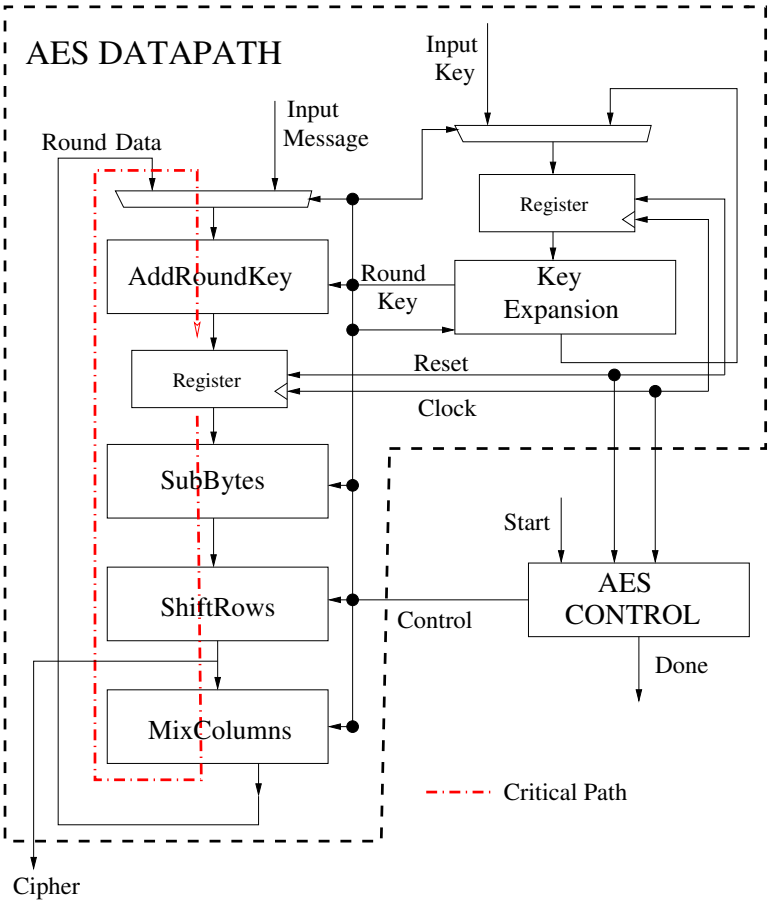


Figure 3.1: AES architecture.

The SubBytes and KeySchedule use 16 and 4 s-boxes respectively. We have developed three different architectures of the datapath with a different s-box in each. In the first and second implementation, s-box is implemented as a table. This table is asynchronous and synthesized in FPGA logic as shown in Figure 3.2. The second implementation uses the same table in synchronous mode. We use the falling edge of the clock to sample the input address of the synchronous table (implemented in block RAM) as in Figure 3.3. Unlike ASIC, RAM are preferred over FPGA logic as the density of RAM is higher. The third implementation is based on finite field arithmetic instead of look up tables. As described by Wolkerstorfer et al. in [60], s-box operation in GF(2^8) can be implemented only with combinatorial logic. The operations in GF(2^8) can be done in GF(2^4) by representing the original polynomial as a linear polynomial with coefficients of four bits each (Figure 3.4). This s-box is implemented completely in FPGA logic.

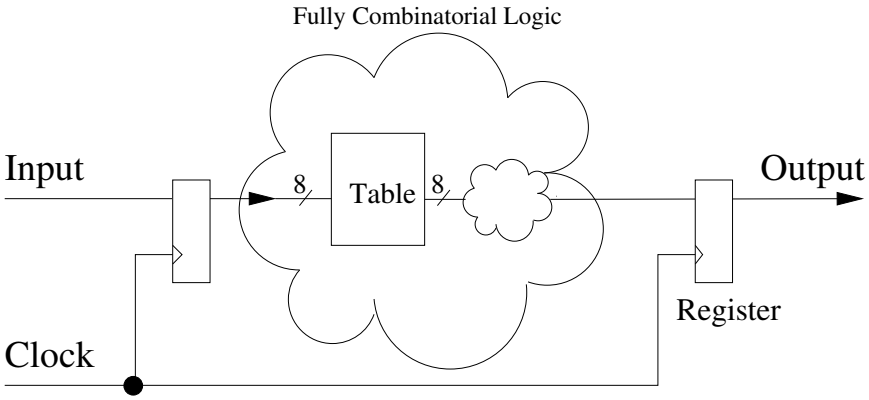


Figure 3.2: S-box as combinatorial tables in LUT.

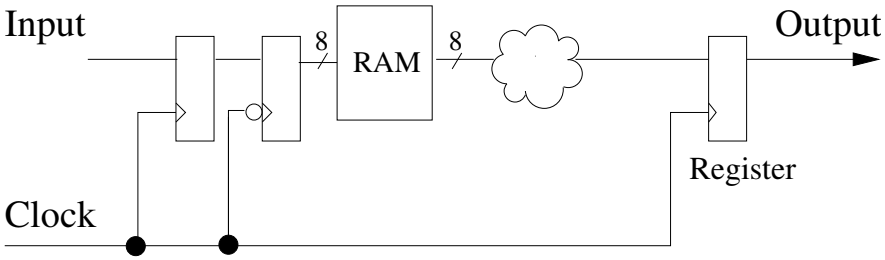


Figure 3.3: S-box as synchronous tables in RAM.

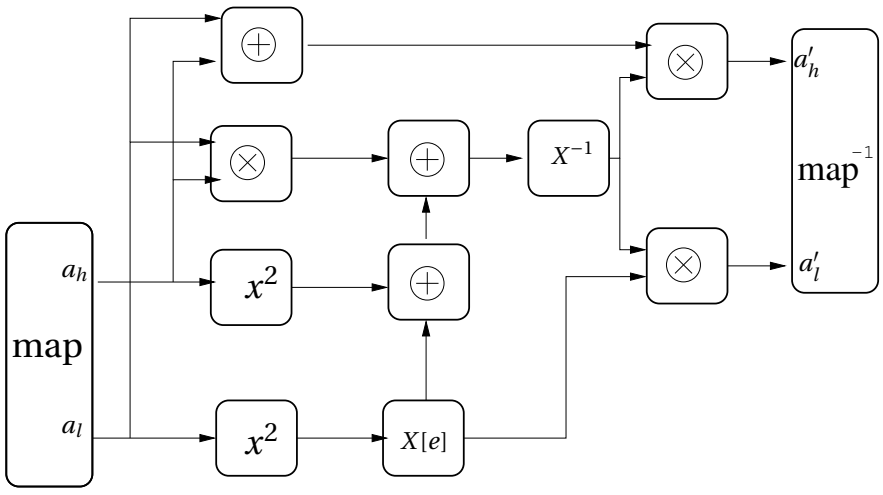
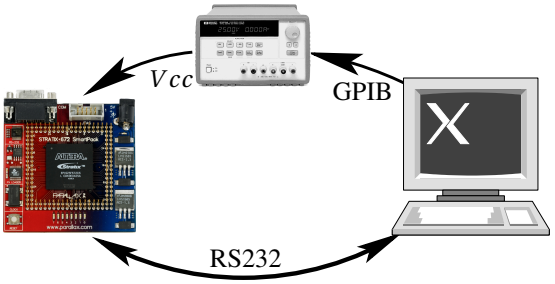


Figure 3.4: S-box using algebraic computation in composite field (GF(2^4)).

3.1.2 Experimental setup and data acquisition

The co-processor along with an UART interface and a controller are synthesized on the FPGA. This design communicates with a monitoring PC via RS-232 serial link. Figure 3.5 sketches the experimental setup.



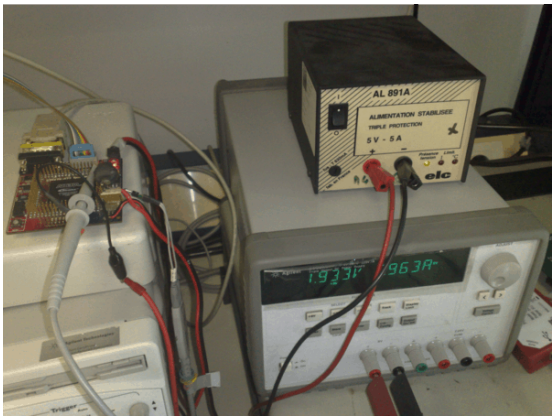


Figure 3.5: Experimental attack platform.

The presented results are obtained with an EP1S25 Altera FPGA soldered on a Parallax evaluation board. For SCA, we measure the EM radiation from the FPGA on an 54855 Infiniium Agilent oscilloscope with a 6 GHz bandwidth and a maximal sampling rate of 40 GSa/s, antennas of the HZ–15 kit from Rohde & Schwarz. We recorded three sets of 10000 side-channel traces (each trace averaged 256 times) related to the activity of the AES crypto-processor, one for each s-box architecture.

As described by Selmane et al. in [61, 62], faults can practically be induced on an FPGA by underpowering the circuit. The power supply is remotely controlled, in order to test a set of non nominal values of V_{cc} successively. When we drive the FPGA at a voltage less than the nominal voltage, the propagation time of the signal increases as illustrated in Figure 3.6. This propagation time of the signal increases because in a CMOS cell when V_{dd} reduces, the

capacitors needs more time to charge and discharge. Such attacks are non-invasive as the attacker does not need access to the silicon die and therefore are easier to implement. We recall that there is no straightforward mechanism to monitor either the power supply level or the frequency in many commodity FPGAs.

Setup time violation on a timing path results in computation of a faulty byte. This fault caused by one or several bit-flips in a byte is called byte-flip fault. It can be observed by monitoring the Hamming weight i.e. the number of non-zero bits in a byte. Here we use the Piret's attack to exploit the faults and retrieve the secret key using the method as described in [26]. Since cryptography involves highly complex computations it is very likely that the critical path is in the cryptographic part [63]. We verified this condition in our setup before starting the analysis. Therefore we do not find any faults on the communication lines.

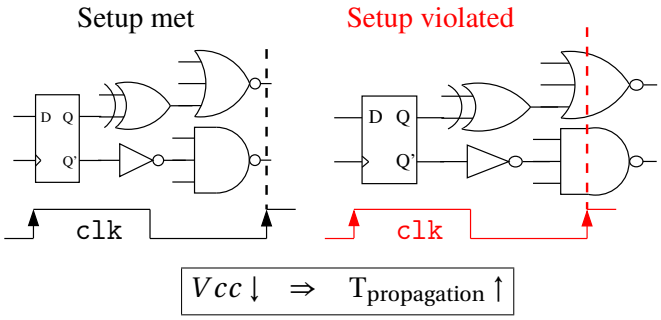


Figure 3.6: Setup time violation caused by a permanent under-voltage.

The nominal voltage of the FPGA running the AES co-processor is 1.5 volts. The operating frequency is 50 MHz. We observe that the FPGA remains functional as we start lowering the V_{cc}. As we continue lowering the V_{cc}, the module starts giving erroneous results. In order to collect faulted ciphertext, for each architecture we have recorded the triples {message, key, ciphertext} for 1,000 encryptions for 100 values of V_{cc} between 1.2 V and 1.5 V. As a result, the entire acquisition campaign consists of 100,000 encryptions for each architecture.

After collecting this set of triples, we analyze the fault. The message and key is considered given while analyzing the faults. The faults is considered only during encryption in the AES datapath and not the communication lines. Since the faults can only affect encryption, only the ciphertext can be affected. We use a off-line module of AES adapted to inject a byte-fault at any round. This method permits the full control of the fault location in terms of space and time. We generate a database of all possible ciphertext and match it with the received ciphertext to find the fault. The software conducts an exhaustive search of single "byte-flip" to compare with received ciphertext. It takes message (m), key (k) and ciphertext

(c) and calls a function AES(). If the output of AES(m,k) is different from c this means that a fault was injected. The output of the software can have two possible faults.

- It will return the location and value of fault against a matched faulty ciphertext. Such fault are “covered”, or
- Render a fault “uncovered”, if no match is found.

A fault could be “uncovered” due to various reasons like multiple faults, fault in the non-cryptographic module, fault in the key expansion etc. As shown in Figure 3.7, fault topology can be identified for a given target. If the fault is ”covered” then we can identify the round and the s-box affected by the fault. For Piret’s Attack, a covered fault in round 8 or 9 (R8 or R9) is considered exploitable.

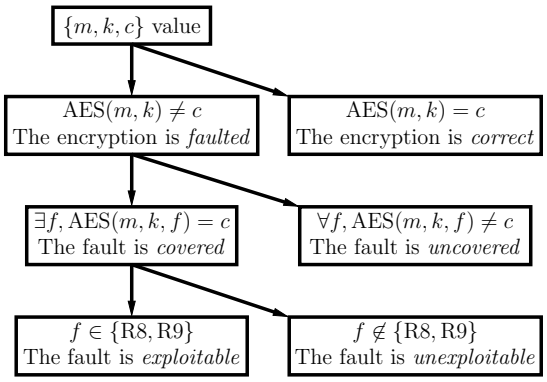


Figure 3.7: AES faults analysis.

3.1.3 Cost Comparison of the Three AES Architectures

For the purpose of cost comparison (in terms of area), we synthesized the AES co-processor with three different s-boxes using Quartus II for Stratix devices. The results are summarized in Table 3.1. AES co-processor with s-box (table) in LUT uses more area while the one with s-box in RAM uses the least area of the FPGA fabric as s-box is the most bulky part. The cost further multiplies because we have used multiple instances of s-box in this parallel architecture. When the s-box is synthesized using GF(2^4), each s-box takes almost 4 times less area than the one in LUT. Hence we reduce the cost in terms of area by changing s-box architecture from LUT to GF(2^4). Every architecture uses 256-bit registers as it memorize the round key and round data once per clock.

Architecture	LUTs	LUTs/S-box	RAM (bits)	Critical Path in Datapath
LUT	5212 (20%)	206	0	13.725 ns
RAM	1061 (4%)	0	40960	19.818 ns
GF(2^4)	2280 (9%)	0	60	17.569 ns

Table 3.1: Cost comparison of the three AES architectures.

3.1.4 Security Evaluation of the Three AES Architectures against SCA

To evaluate the three implementation against SCA, we acquired the traces from Altera Stratix FPGA. The traces were acquired using a EM probe fixed near one of the decoupling capacitors of the FPGA. Thereafter we launched a correlation power analysis (CPA) on the acquired traces. 5000 traces were acquired independently for each implementation. A CPA using Hamming distance model was performed on the last round of AES. The results are presented in Table 3.2, in terms Minimum Traces to Disclose the key (MTD) for the first 8 s-boxes of AES. It can be inferred from Table 3.2 that s-box in LUT are the weakest against SCA while s-box in RAM are the strongest. This is due to the construction of RAM in the FPGA which is foundry-made fixed size hardware and not a programmable block. Therefore the power leakage can be less data dependent. Similar results have been presented by Kaps et al. in [64]. As the EM probe collects localized activity, we find that some s-boxes are easier to break than others. A different position of EM probe might change the scenario.

Table 3.2: Minimum traces to disclose (MTD) the key for first 8 s-boxes of the three AES implementations.

S-box type	**S-box0**	**S-box1**	**S-box2**	**S-box3**	**S-box4**	**S-box5**	**S-box6**	**S-box7**
LUT	163	623	203	263	729	278	208	565
GF(2^4)	207	520	357	340	1002	290	195	717
RAM	367	1022	414	463	1325	467	200	927

3.1.5 Security Evaluation of the Three AES Architectures against FA

In our architecture, the propagation delays in the datapath are larger than the ones in key schedule. Therefore global perturbations on the power lines will affect only the datapath.

At higher voltages (*i.e.* close to the nominal voltage) only single faults occur. As we keep decreasing the voltage, setup times for more than one path are violated to inject multiple faults(uncovered). Our method can be adapted to perform other DFA on AES like the one propose by Kim et al. [65] where a fault is injected in the key schedule.

Figures 3.8, 3.9, 3.10 show the occurrence of faults for the three architectures. Faults are divided in two categories: single *i.e.* faults on one byte of the AES state (datapath register) before SubBytes or multiple *i.e.* faults on more than one byte or in the keypath. In all the three figures the distribution of single faults is in a "bell-shape" which corresponds to a fault model where errors are caused by a setup violation on critical combinatorial path. This is because as the voltage is slightly reduced, single faults appear with low frequency. Further reduction in voltage first causes an increases in the frequency of appearence of single fault followed by a decrease after a certain point. At this point multiple faults start appearing. Since the propagation time of a signal increases with decreasing supply voltage, multiple faults become very likely at lower voltages

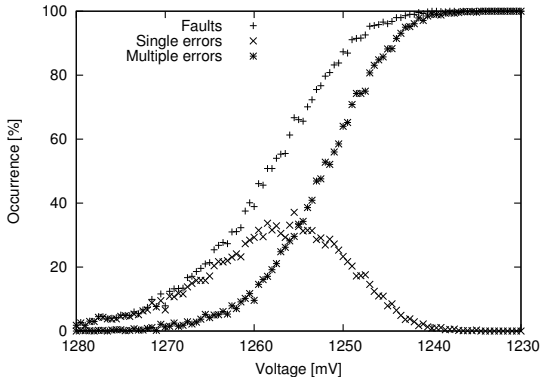


Figure 3.8: Occurrence of faults: s-box in $GF(2^4)$.

Coverage of single faults is shown in figures 3.11, 3.12, 3.13 for the three AES architectures. Coverage is the ratio of single faults detected to all the detected faults. The higher voltage values majority of faults are single as the coverage ratio is close to one. When we further decrease the voltage, the coverage reduces as multiple faults appear. We use the "Piret's Attack" [26] to exploit the faults which use single faults affecting only the two penultimate rounds to retrieve the key. In the following, such faults are called exploitable faults.

Figures 3.14, 3.15 and 3.16 show the Hamming weights of the exploitable byte-flips. The faults are mostly single bit faults (Hamming Weight of the fault=1). This information allows attacker to mount one of the published "Bit Fault" attacks [66].

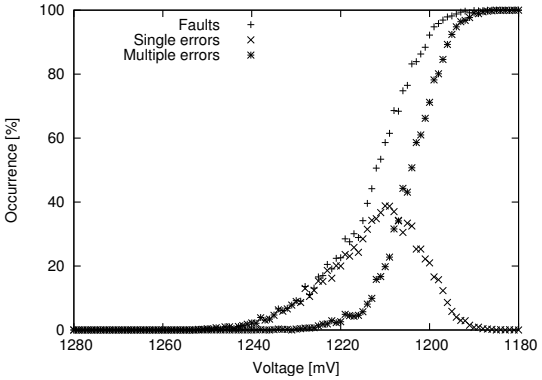


Figure 3.9: Occurrence of faults: s-box in LUT.

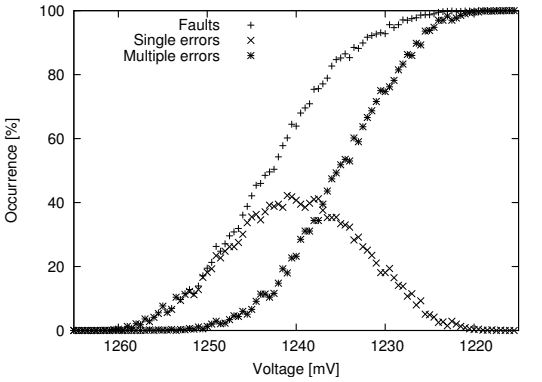


Figure 3.10: Occurrence of faults: s-box in RAM.

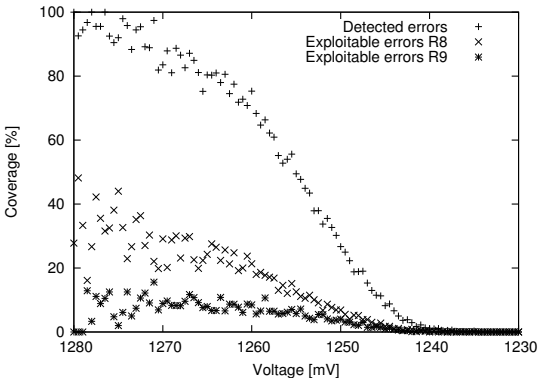


Figure 3.11: Coverage of single faults, and detail of exploitable faults in $GF(2^4)$.

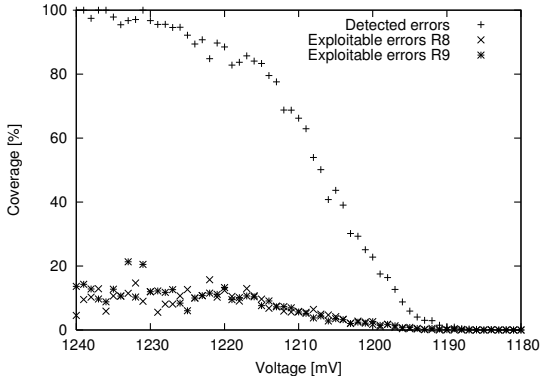


Figure 3.12: Coverage of single faults, and detail of exploitable faults in LUT.

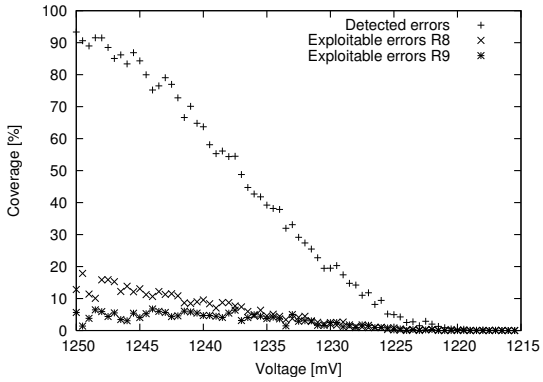


Figure 3.13: Coverage of single faults, and detail of exploitable faults in RAM.

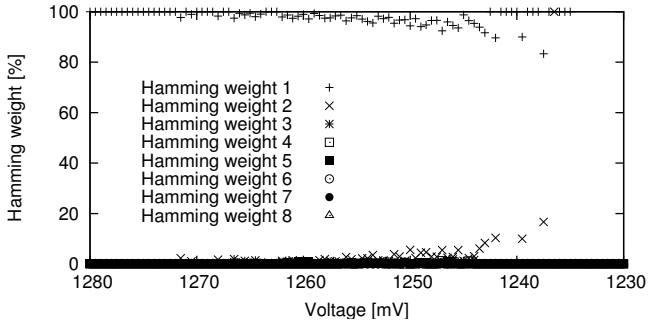


Figure 3.14: Hamming weight of exploitable faults in $GF(2^4)$.

As the software for fault analysis also gives the round and s-box of fault injection, we plot in Figures 3.17 and 3.18 the temporal and spatial localization of the single faults. In

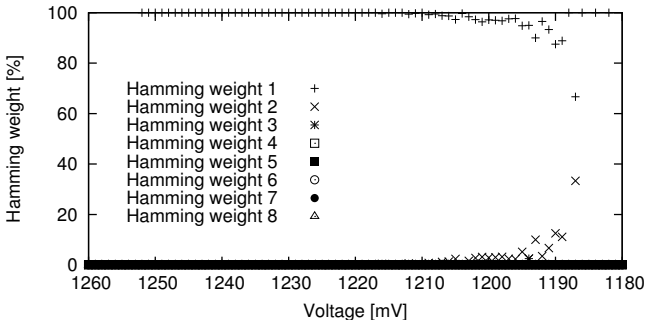


Figure 3.15: Hamming weight of exploitable faults in LUT.

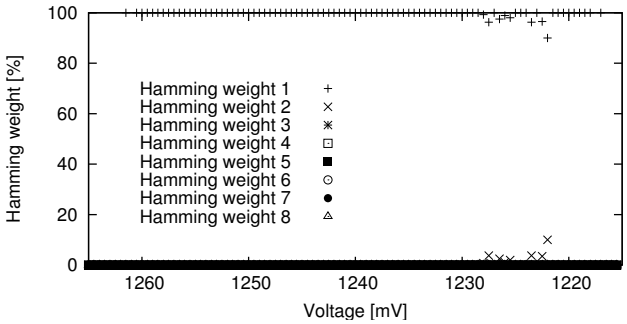


Figure 3.16: Hamming weight of exploitable faults in RAM.

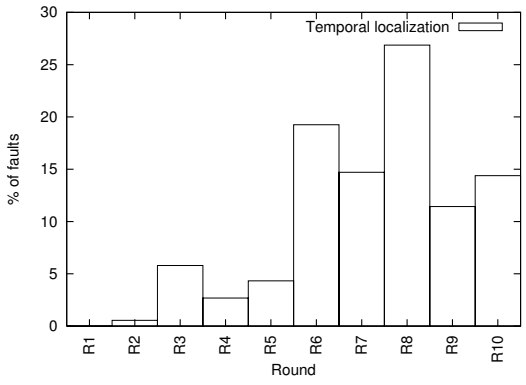


Figure 3.17: Temporal localization of single faults.

Figure 3.17 there is no fault in first round. This is because the first round in AES is comprised only of AddRoundKey operation resulting in a fairly small propagation path. Thus no faults occur in the communication lines. In Figure 3.18, each s-box has different number of faults

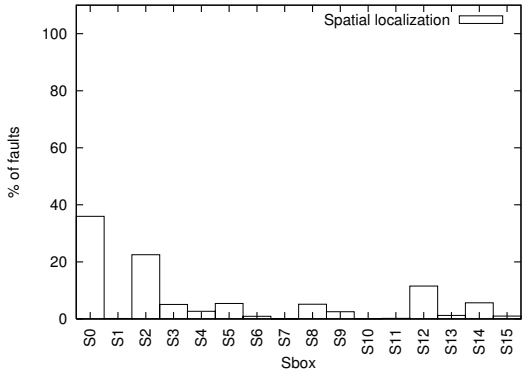


Figure 3.18: Spatial localization of single faults.

over the ten rounds. An uneven temporal and spatial distribution of the faults is observed which may come from data-dependency of the computation time of a logic gate.

We plot the exploitable faults in three architectures on the same diagram as shown in Figure 3.19. Architecture with s-box in GF(2^4) shows the highest peak of the bell shaped distribution . It shows that around 13% of the single faults are exploitable. On the other hand, when s-box is implemented as a table in LUT less than 6% of the faults are exploitable. Thus half the amount of faulty ciphertext are needed when attacking s-box in GF(2^4) as compared to s-box in LUT. These results are in accordance with the results obtained by Francq et al [67], where authors have used the timing analysis of post map netlist of AES co-processor. As per their results, attacking an s-box in LUT is harder than an s-box in GF(2^4) because a "higher attack" frame increases probability of a fault. Thus, we have practically proved the results which were stated theoretically in [67]. An s-box which needs a comparatively larger decrease in voltage to compute faulty results is more secure because at lower voltages some other part of the design may be faulty as well.

Recently, Drimer et al [68] presented an AES design which synthesizes various components of AES in block RAM and DSP to reduce logic utilization in FPGA. We also tested an architecture where s-box is in block RAM. As shown in Figure 3.19, 9% of the faulty ciphertexts are exploitable as compared to 6% in case of s-box in LUT. Security has a cost; this rule of thumb also applies to AES. It is up to the designer to make an intelligent choice.

Figure 3.19 shows that s-box in RAM is more secure than the s-box in GF(2^4). On the other hand in Table 3.1, the critical path timing suggest the opposite. This discrepancy in results can be explained by following arguments. The clock period is 20ns which triggers the state register with the rising edge and RAM with the falling edge. The critical path which is

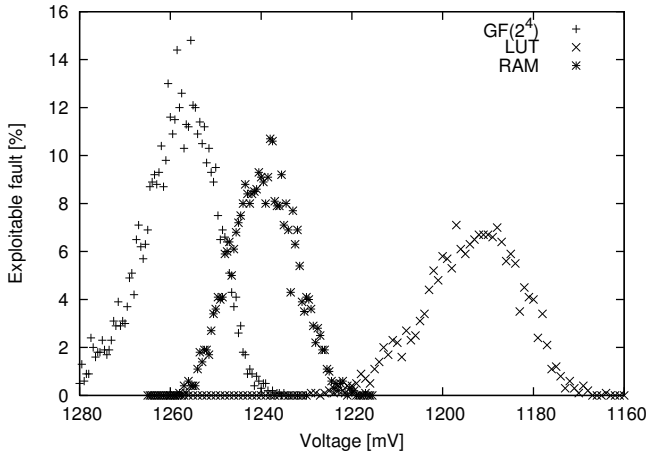


Figure 3.19: Exploitable errors.

computed between the two edges accounts for negative half of the clock cycle. Quartus II normalizes the timing depending on the duty cycle of the clock and displays in terms of one full cycle. This fact was confirmed when reduction in the duty cycle of the clock resulted in a higher maximal frequency of operation. Actual timing is less than 19.818 which should be interpreted for each half of the clock cycle separately.

In case of s-box in GF(2^4), timing information given by Quartus II is trustworthy as all the operation are sensitive to rising edge of the clock. When the s-box is implemented in GF(2^4), as shown in the architecture, The critical path will begin and end at the state register. For s-box in RAM, the critical path is between the RAM and state register. The s-box in GF(2^4) is composed of combinatorial components. Since fault occur dynamically, presence of many combinatorial components increases the probability of faultoccurrence in this architecture. No concrete conclusion can be provided as the construction of RAM and LUT is not precisely known. Due to the difference in form, delays due to underpowering will evolve differently.

3.1.6 Discussion

We have seen the evaluation of the three s-box implementations against SCA and FA. S-box in RAM provide maximum resistance against SCA while s-box in LUT are better suited to resist FA. A designer should make similar analysis to find the best trade-off between area and security. For example, a designer who is constrained by area of FPGA fabric can choose s-box in GF(2^4) or RAM. Another designer concerned with performance can go for s-box in RAM. On the other hand, if security is the prime concern, s-box in RAM is good choice. S-

box in RAM provide good resistance against SCA and FA at a low cost. If the device is located such that SCA is not possible but FA is, s-box in GF(2^4) can be chosen. The main idea is that when a designer designs a cryptographic processor, an analysis of the cost and security requirements should be done and choices should be made at a reasonable trade-off.

3.2 Unrolling Cryptographic Circuits as a Countermeasure

3.2.1 Rationale of the Countermeasure

In a cryptographic block algorithm, data is ciphered by repeating a set of operations with a different key value each time generated from the previous key. This set of operations is called a round. The number of rounds is chosen such that linear and differential cryptanalysis are more difficult than an exhaustive key search. Normally, cryptographic circuits are designed to perform either some operations of a round or the whole round in one clock cycle. Thus the value of the key remains the same for one or more clock cycles. The attacker can guess some of the key bits and correlate it with leakage acquired. A correct guess will give a much higher correlation as compared to wrong guesses when the acquired leakage contains sensitive information.

Data registers, in general, are a soft target for SCA. This is because the leakage from the register is high due to its load and the leakage is synchronised to the clock. In combinatorial logic, the leakage is low and spread over time. If the result of a round is stored in the register at the end of each clock cycle, attacker can easily retrieve the subkey by guessing and correlating. Now, if the key is changed twice during one clock cycle *i.e.* two rounds are executed per clock cycle the key used for one round is further diffused deeper into the design and mixed with the second key. Thus to exploit this property we propose to design the cryptographic coprocessors in such a way that it executes multiple rounds in one clock cycle. We call this as "unrolling the rounds of the algorithm". Also we define the **unrolling factor** as the number of rounds unrolled. Unrolling can be categorised as an information hiding countermeasure. An implementation unrolled twice means that two rounds are performed every clock cycle. A didactic presentation of the loop unrolling technique is given by Kris Gaj and Pawel Chodowiec in the Chapter 10 of "Cryptographic Engineering" [69], along with a discussion about its pros and cons from a performance point of view.

Figure 3.20(a) shows the architecture of one round of a normal iterative cryptographic algorithm while Figure 3.20(b) shows the architecture of an unrolled cryptographic algorithm. An idea of the difficulty to mount a side channel attack on the unrolled version can be estimated from the following discussion. Suppose, we have two implementations of a

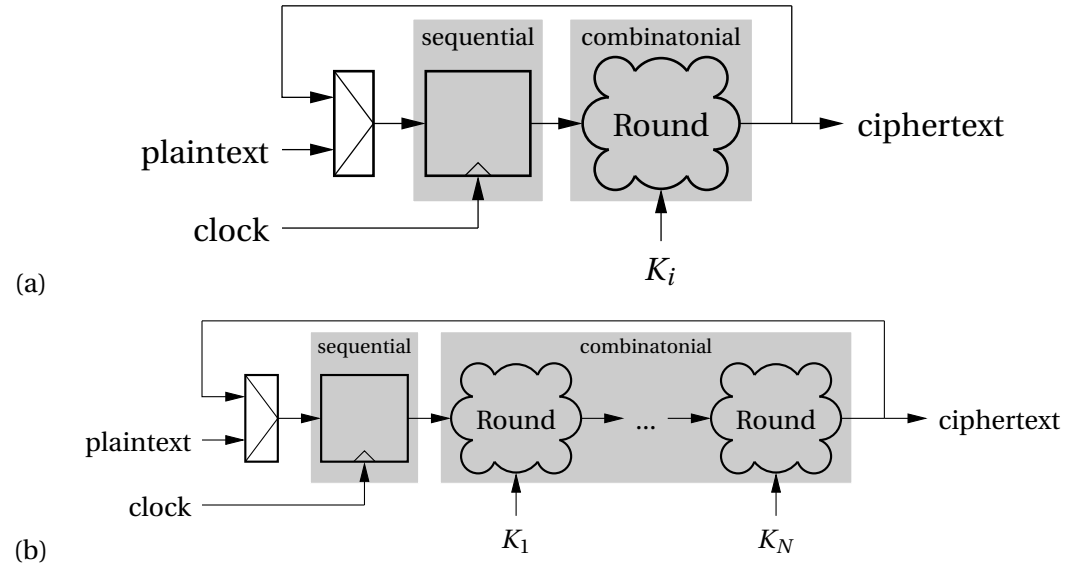


Figure 3.20: (a) Architecture of a iterative cryptographic algorithm. (b) Architecture of a fully unrolled cryptographic algorithm.

cryptographic algorithm: one iterative and the other unrolled with an unrolling factor of 2 as shown in fig 3.20(a) and (b) respectively. Let us examine the signal and the noise when the attack is mounted on 1-bit. In the iterative design, the signal will be the sum of the power activity of all the combinatorial gates and flip-flops involved in calculating that bit. The noise shall be sum of power activity of other gates and flip-flops. In the unrolled design with unrolling factor of 2, an attack on 1-bit in the first of the two rounds will have a signal as the power activity of the gates involved in the computation of that bit. The activity of registers is absent because the register contains the output of subsequent round. The noise shall be twice the previous value as components are doubled. Since the power activity of a combinatorial gates is less than the power activity of a register, the SNR is reduced by a factor of over two.

A rough evaluation of the theoretical complexity of this countermeasure in terms of area is given by the unrolling factor. Thus a design unrolled twice will have double the area of its original design as far as combinatorial part is concerned. In terms of performance, the trade-off is almost null. Unrolling factor of n will multiply the critical path by n times and thus maximum frequency is reduced $1/n$ times. Since n rounds are executed per clock cycle, N/n clock cycles are needed to execute the whole algorithm where N is the total number of rounds. Thus the throughput is approximately the same for original and unrolled design.

The practical results are better than the one described as some of the unnecessary components like multiplexers are removed while unrolling. Thus the area is n times smaller. The maximum operating frequency is reduced by $1/n$ times. We also point out that the unrolling does not impact the possibility of the encrypting block to be used with any mode of operation (CBC, CFB, OFB, *etc.*).

3.2.2 Fully unrolled DES implementation on ASIC

An iterative architecture can be made combinatorial, by removing its register transfers occurring during the rounds as demonstrated by Guilley et al. in [70]. The architecture is based on that described in [71], and the floorplan are depicted in Figure 3.21 and 3.22. Our DES architecture is composed of bit shuffling data flow primitives (permutations, multiplexers and flip-flops) and round logic. LR is a 64-bit register to hold the ciphertext or 16 intermediate messages depending on the architecture. CD is a 56-bit register to hold the key without parity bits to store 16 round keys. Registers LR and CD must be loaded sequentially. In the case of unrolled DES, the algorithm combinatorial depth is thus roughly increased by a factor of sixteen, but the registers LR and CD remain frozen during sixteen clock cycles or clock frequency is reduced by 16 times, which makes up for the delay through the gates. It is a special case of the so called *brutal countermeasure* as described by Roche et al. [72], where the "glued blocks" actually make up the entire datapath. The inputs 1 of the LR multiplexer and 2 of the CD multiplexer play the role of enable for the corresponding registers. The key schedule consists in a sequence of pre-computed circular shifts which can be implemented just by swapping wires and requires no logic. Such a technique is only valid for certain algorithms like DES and the absence of logic in key schedule avoids leakage. Thus attacks similar to the one proposed by Elaabid et al. [73] cannot be mounted anymore.

The synthesizers, in default mode, attempt to fit a timing path into one clock cycle. To synthesize such a design, timing constraints should be relaxed. In the combinatorial DES specific case, the logic driven by LR and CD has time equivalent to sixteen clock cycles to execute. This piece of information cannot be easily inferred, thus user constraints must be set. They basically consist in specifying spare clock cycles for some timing arcs. The timing paths that are concerned thus start at registers LR and CD, plus the Boolean signal originating from the control that tells whether the current operation is a ciphering or a deciphering , where the shifts can be interpreted left or right-wise. The "multi-cycle" constraints listed in Figure 3.23 express the fact that outputs of LR and CD are sixteen times slower that the clock and that the signal to decide between ciphering and deciphering is a false timing path. This last path is indeed never critical because the choice between encryption and decryption is

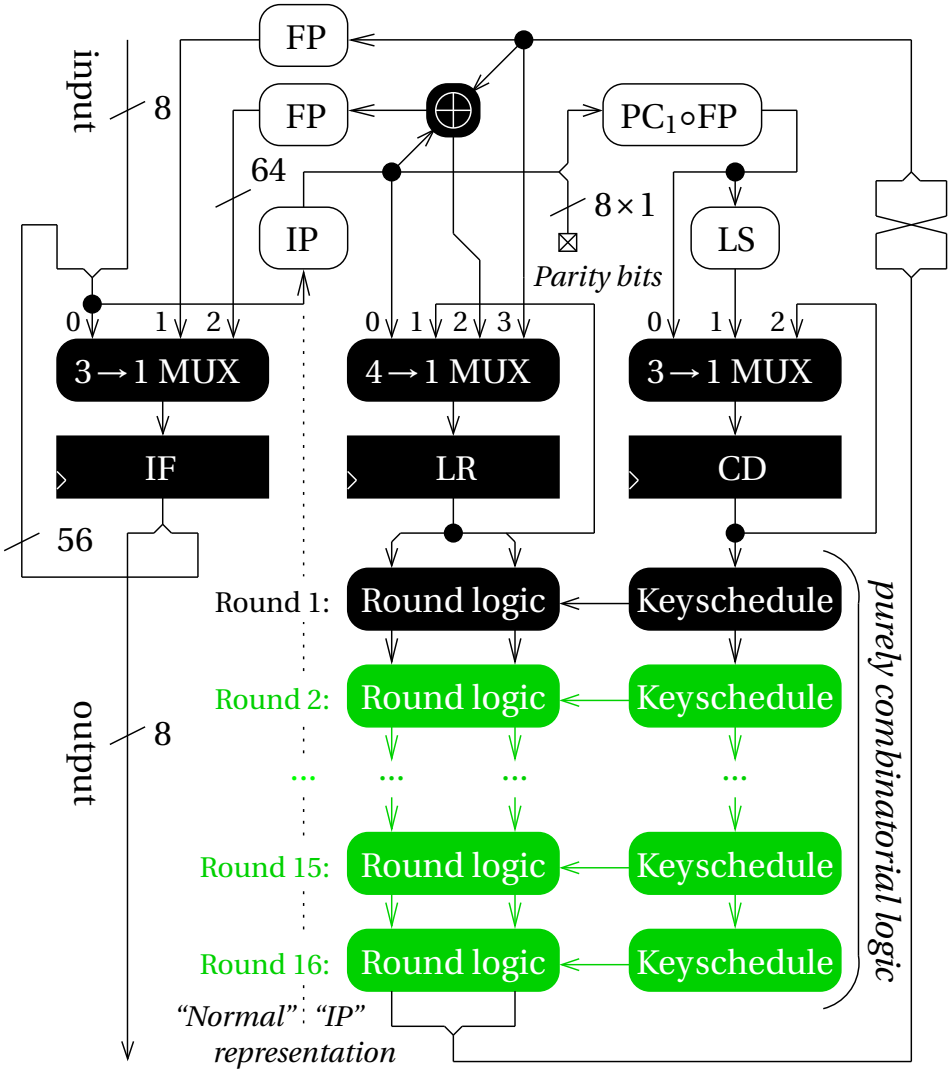


Figure 3.21: Unrolled DES Architecture.

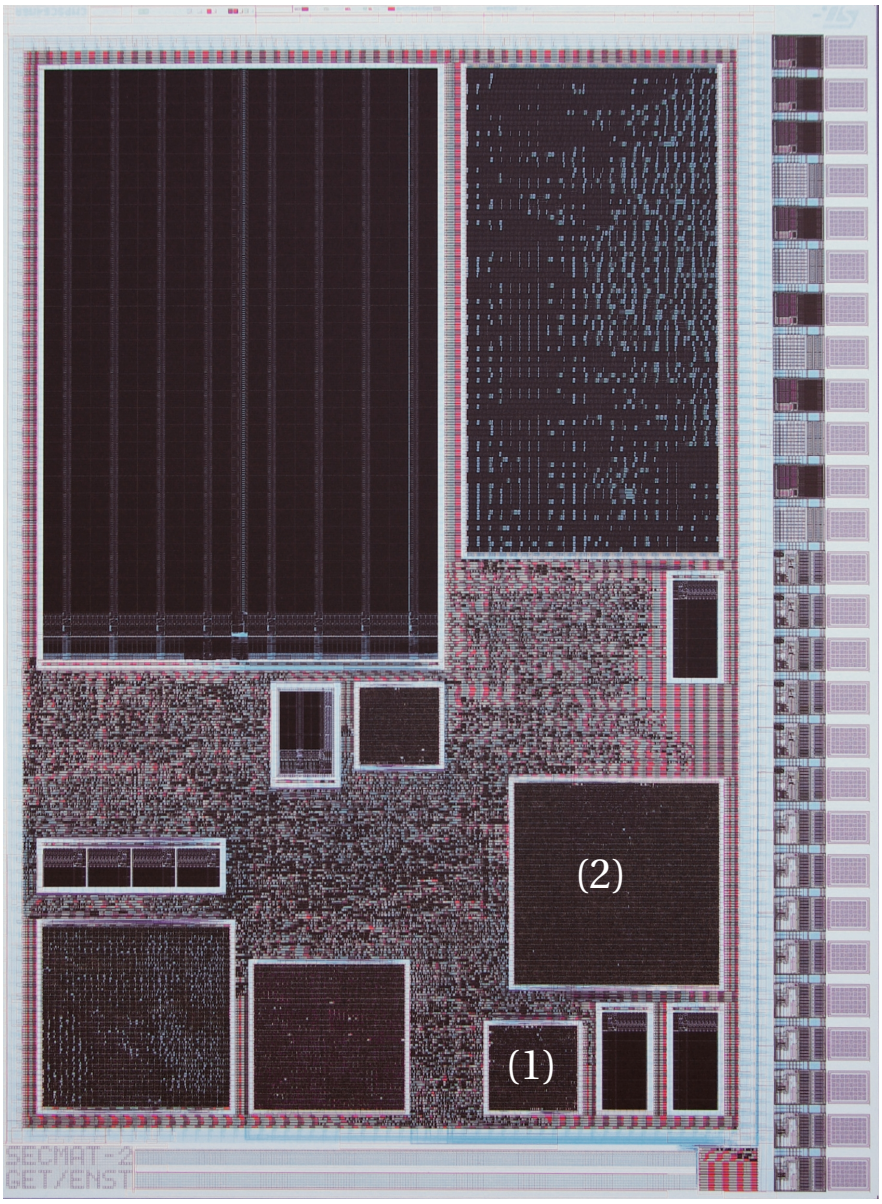


Figure 3.22: Floorplan of the ASIC implementing DES iterative (1) and DES unrolled (2).

```
set_current_module des_datapath_combi_wrapper; # Internal constraints
set_current_instance [find -hier -inst I_REG_LR];
# The following constraint (1+15 cycles allowed for the computation)
# concerns the whole bus:
set_cycle_addition -from [get_info [lindex [find -port q] 0] bus] 15;
set_current_instance [find -hier -inst I_REG_CD];
set_cycle_addition -from [get_info [lindex [find -port q] 0] bus] 15;
set_current_module des_datapath_combi; # External constraint
set_false_path -from [find -port sel_left_not_right]; # Encrypt/Decrypt
```

Figure 3.23: TCL timing constraints crafted for the "multi-cycle" DES combinatorial datapath synthesis for Cadence `bgx_shell`.

not modified during one computation. Similar constraints can be forced for FPGA, however in our experiments the design was functional without specifying the constraint to FPGA. Indeed, every round key in DES is obtained by simply selecting the adequate bits from the 56 bit master key.

We implemented an iterative DES and a fully unrolled DES on SecMatV2: an academic ASIC for security evaluation of cryptoprocessor implemented in 130 nm technology from ST Microelectronics. The placement constraint used for both modules is that their placement density is 95%. Therefore we found that iterative DES consumes an area of 24787 μm^2 while the unrolled DES consumes an area of 139816 μm^2. The actual ratio of the surface area turns out to be 5.64 as compared to 16, the unrolling factor. This reduction in surface area is due to removal of registers, removal of logic involved in the iteration management (multiplexers) and round boundaries optimization. Also the key schedule is completely dissolved in mere routing which is a property specific to DES algorithm. In terms of performance, the iterative DES needs almost 5 times more time for single encryption.

This logic can be directly implemented in FPGA as well. Table 3.3 details the implementation details of this countermeasure on different FPGA.

3.2.3 Security Evaluation of the Proposed Countermeasure

The average side-channel curves as acquired from the SecMatV2 ASIC for one DES encryption are shown in Figure 3.24 and 3.25 respectively for the iterative reference DES and the combinatorial instance. It clearly appears in Figure 3.24,3.25 that the variations increase during the encryption because cryptographic operations consume a lot of system resources.

Side-channel attacks can be roughly divided into two categories. On one hand correlation attacks make the assumption of a known leakage model; several models corresponding

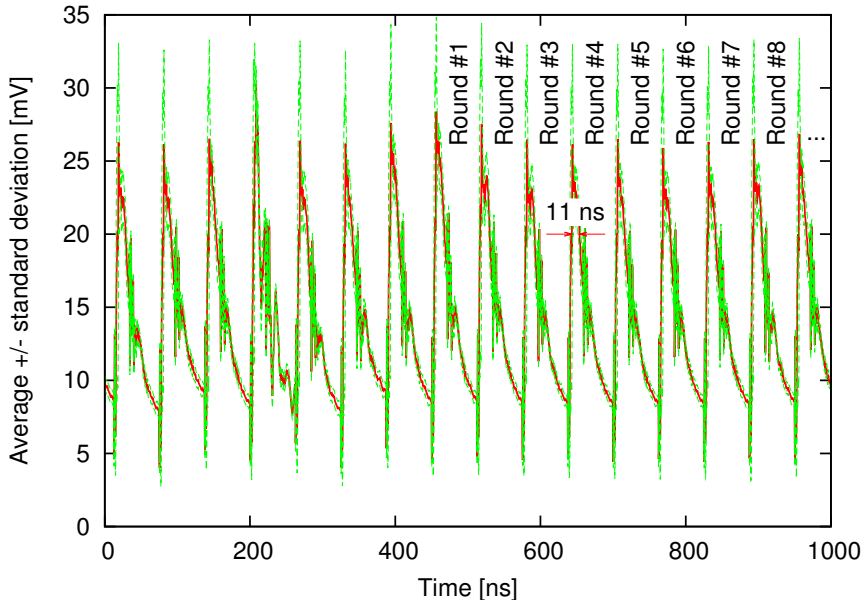


Figure 3.24: Sequential iterative DES encryption signature, with the average variation margin, for statistics collected on 10k measurements.

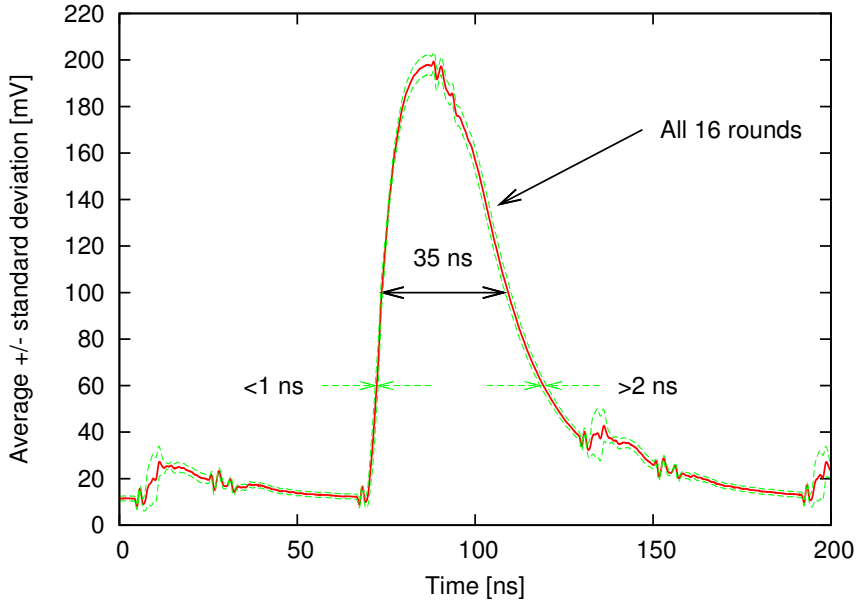


Figure 3.25: Average unrolled DES encryption signature, with the average variation margin, for statistics collected on 100k measurements.

Table 3.3: Cost & Performance Comparison Iterative and Fully Unrolled DES.

Xilinx Virtex 5		
Architecture	Iterative DES	Unrolled DES
LUT	582	2053
Registers	184	184
Freq. (MhZ)	399.93	36.14
Altera Stratix 2		
Architecture	Iterative DES	Unrolled DES
ALUT	324	2024
Registers	184	184
Freq. (MhZ)	336.59	36.78

to different values of the secret are devised. The model that correlates better with the concrete measurements discloses the secret. On the other hand, template attacks are divided into two steps. The first step is done off-line; it consists in pre-characterizing the circuit in an almost blind fashion, for as many representative values of the message and key inputs. Stochastic attacks are a variant where the pre-characterization is made simpler by injecting some partial knowledge about the target's leakage. The second step is the proper on-line attack itself. The attacker attempts to recognize the secret by matching measurements obtained from a fixed albeit unknown secret key.

We show that correlation attacks are made very implausible on a fully combinatorial implementation, due to the signal's desynchronization, even in the early rounds (represented in Figure 3.26). First of all, we apply the same attack that is successful on the iterative reference implementation. It consists in a correlation of the measurements with the consecutive values of the right datapath register R_0, that leaks $\mathscr{L}(initial : R_0, final : L_0 \oplus f(R_0, K_1)) = |R_0 \oplus L_0 \oplus f(R_0, K_1)|$. In our attacks we measure the peak power on the curve. The attack results on iterative and unrolled DES are shown in Tab. 3.4 and 3.5 respectively . Without any surprise, this attack completely fails on the combinatorial instance of DES, since the targeted transition has disappeared in the unrolled implementation. We would like to emphasize that each time an encryption is done, the datapath should be cleared. In other words, the encryption should not start with register in a known state. This can be done like precharge in dual-rail precharge logic (DPL) or by propagating random values without interference from the key. This is because, if two consecutive computations are done then some correlation can be found on the basis of previous computation using the Hamming

distance leakage model.

Table 3.4: Key recovery attack on the iterative reference DES using a CPA over 10K traces.

S-box index	Key		Lock_t $0 \leq \cdot \leq 10\,000$	SNR	Max CPA [%]
	Actual	Guessed			
1	56	56	4 314	4.38603	8.40
2	11	11	7 848	3.94818	5.68
3	59	59	1 247	5.29027	6.81
4	38	38	3 555	5.09747	5.94
5	0	0	2 272	7.25941	8.86
6	13	13	3 868	4.52662	8.10
7	25	25	4 399	4.69634	6.28
8	55	55	273	6.81590	14.68

Table 3.5: Key recovery attack on the unrolled DES using a CPA over 100K traces.

S-box index	Key		Lock_t $0 \leq \cdot \leq 100\,000$	SNR	Max CPA [%]
	Actual	Guessed			
1	56	58	87 976	1.83827	3.25
2	11	21	75 073	3.04394	1.52
3	59	17	97 462	2.07826	2.69
4	38	25	71 369	1.63005	4.85
5	0	53	70 590	3.45533	2.18
6	13	26	99 982	3.01725	1.18
7	25	22	70 433	2.07131	3.37
8	55	47	74 552	2.78395	3.26

3.2.4 Attack on the Unrolled DES

Now let us observe a case when the previously described constraints are not respected i.e. two encryption are done without clearing the datapath. We explore the Hamming distance (HD) leakage model, on two positions of the algorithm, namely the Feistel function output (P1) and the round output right half (P2). We find that the HD on P1 completely discloses the key. The results are given in Tab. 3.6. We can see that for all the eight broken substitution boxes, the signal-to-noise ratio (SNR) is much smaller than for the case of the reference circuit. The results for the s-box 4 are printed in italics, because actually two keys are guessed

Table 3.6: Key recovery attack using a CPA with a Hamming distance model (with respect to the previous encryption) over 100K traces.

S-box	Key		Lock_t	SNR	Max CPA
index	Actual	Guessed	$0 \leq \cdot \leq 100\,000$		[%]
1	56	56	16 557	2.20267	2.17
2	11	11	44 092	2.15008	2.09
3	59	59	36 090	2.50697	2.22
4	*38*	*9*	*3 291*	*3.73242*	*5.01*
5	0	0	27 164	1.96649	2.28
6	13	13	20 138	2.13591	2.65
7	25	25	17 862	2.11245	2.86
8	55	55	37 317	2.77701	2.75

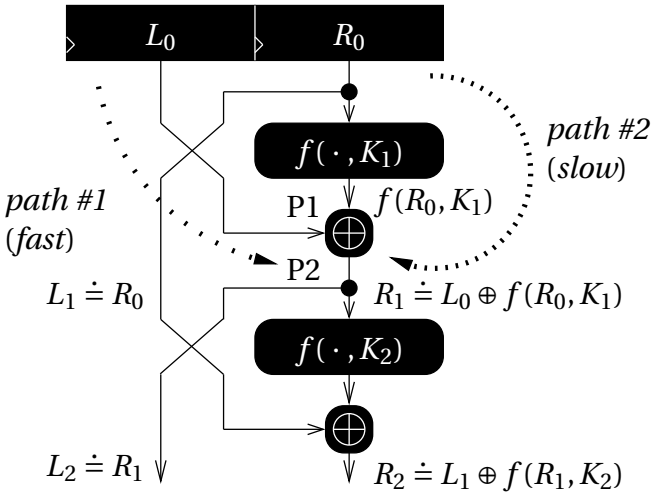


Figure 3.26: Notations used to describe the combinatorial DES leakage functions.

simultaneously in a unrolled implementation, due to a mathematical property of this s-box. The fourth s-box S_4 of DES has the following property: $\forall x, y \in \{0,1\}^6$, $S_4(x) \oplus S_4(y)$ and $S_4(x \oplus \texttt{0x2f}) \oplus S_4(y \oplus \texttt{0x2f})$ are palindromic. This fact can be shown by computing exhaustively the two expressions and comparing them.

Therefore, we have a remarkable Hamming distance conservation property: $\forall x, y \in \{0,1\}^6$, $|S_4(x) \oplus S_4(y)| = |S_4(x \oplus \texttt{0x2f}) \oplus S_4(y \oplus \texttt{0x2f})|$. As a conclusion, in a Hamming distance model, two keys are retrieved in pairs: the correct one and one another (false), equal to the correct key translated by `0x2f`.

To show that the correlations of the s-boxes output (locus P1) are very disrupted due to their combinatorial nature, we have shown the DPA peaks in Fig. 3.27. We favor DPA over CPA, because, as explained by Guilley et al. in the technical article [74], the covariance used by DPA extracts the activity of some nets in the netlist, which is interesting for leakage characterization. As for the CPA, it is more suitable for attacks, because the normalization by the trace standard deviation corrects the fact that the leakage is not necessarily maximum at the times where the side-channel is [75]. The DPA covariance $|f(R_r^{-1}, K_{r+1}) \oplus f(R_r, K_{r+1})|$ for all $r \in [0,6]$ are plotted in Fig. 3.27.

We have also added the transition in R_0 between two consecutive messages, because it indicates the computation beginning and its end. The beginning consists of the R_0 register sampling at the rising edge of the clock. The end corresponds to the other transition (final → initial), in the R_0 register input latches, that are transparent, and that dissipate even in the absence of a clock event. We observe that the DPA covariances do not especially show peaks ordered in time. This indicates the link between the data and the side-channel measurement is destroyed as early as the first couple of rounds.

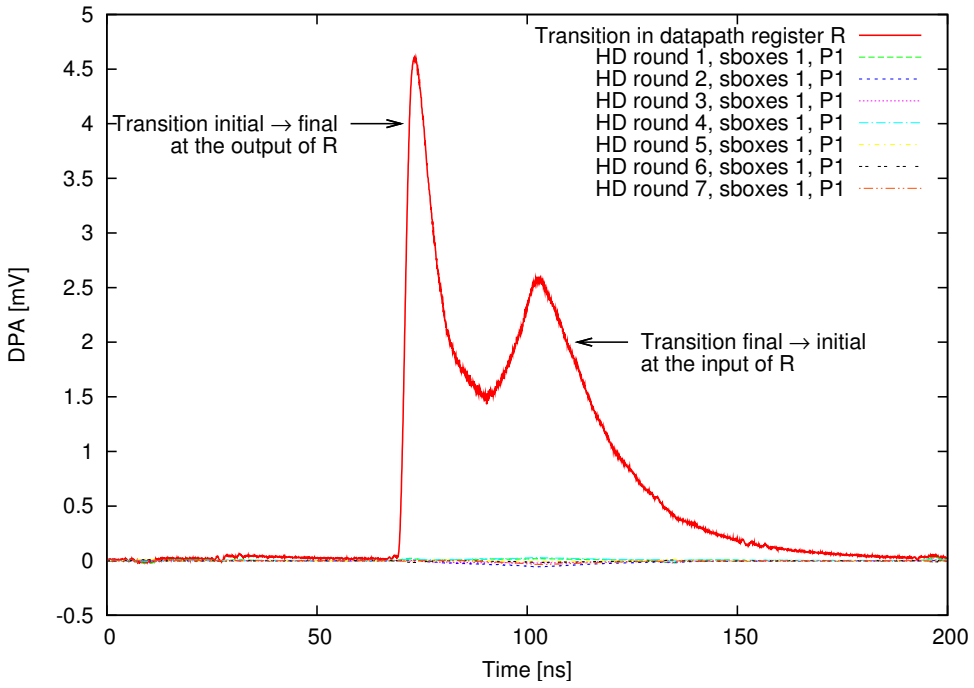


Figure 3.27: DPA covariance for the register transfer R_0, and round correlations for the first s-box outputs.

3.2.5 Evaluation Based on Mutual Information Metric

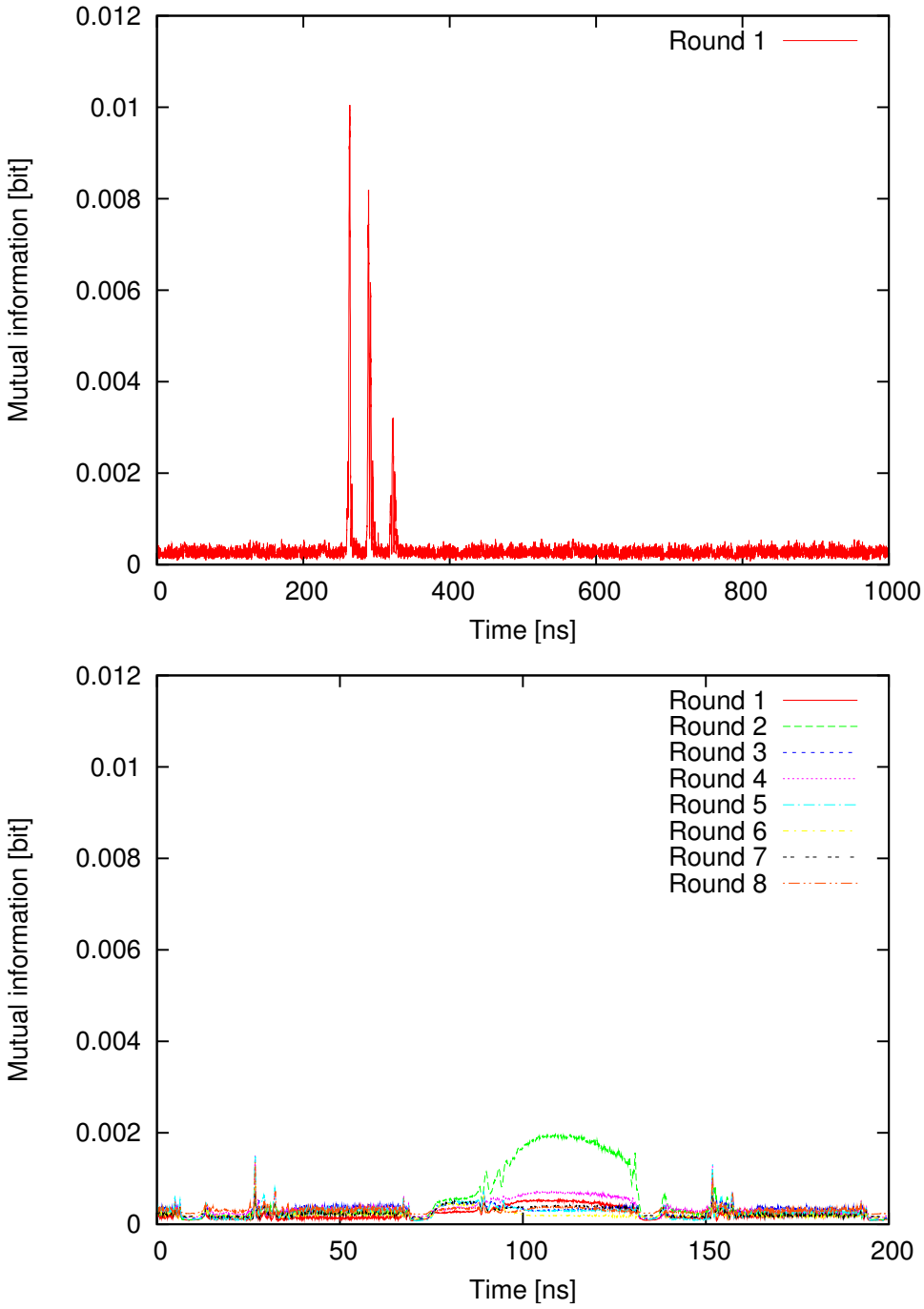


Figure 3.28: Mutual information metric for sequential (*top*) and combinatorial (*bottom*) DES.

Mutual information analysis (MIA) has been introduced by Gierlichs et al. [29] and further discussed by Prouff et al. in [30]. This analysis captures whatsoever dependence between measurements and a leakage model. It is thus a tool suited for an information leakage evaluation, as pointed out by Standaert et al. in [76]. The default leakage model does not assume any device-specific knowledge. Therefore it considers plain dependency with one sensitive and predicable word within the device. The notions of sensitivity and predictability have been defined in [77]. Basically, a variable is sensitive if it depends on one secret, and predictable if testing all the hypotheses for this variable is computationally tractable. The leakage-agnostic approach is the one employed in template attacks [78].

We have computed the mutual information (MI) between the right half of the datapath for s-box #1 and each point of our experimental traces. The results are plotted in Fig. 3.28 for the 80k traces of the iterative DES module and the 100k traces of the unrolled one. In the iterative circuit, the MI is roughly the same for each round. However, it depends on the round index for the combinatorial circuit; therefore we represent a couple of them in Fig. 3.28. It appears clearly that the sequential circuit is leaking more information about the first round than the combinatorial. Hence even with MIA we demonstrate that our proposed countermeasure is sound. It is difficult to find a precise leakage model for such architecture which increases the SCA resistance.

3.3 Conclusions

In this chapter, we first present the security evaluation of three different AES implementations against SCA and FA. These implementations differ in the s-box architecture. S-box in RAM is a good choice in terms of cost and security. S-box in GF(2^4) can be used when the whole design has to be implemented in LUT or ASIC. It provides appropriate resistance at reasonable cost.

Next, we propose a new information hiding countermeasure which comprises unrolling of rounds of a cryptographic algorithm to execute during a single clock. We implemented a fully unrolled DES crypto-processor on a 130-nm ASIC. Results show that unrolling is secure against power attacks with a constraint of clearing the datapath after each encryption. We also evaluated mutual information metric on the design and results show that unrolled DES is less vulnerable. Further work involves testing this countermeasure with other algorithms like AES, *etc.* Also it could be interesting to partially unroll the algorithm for the rounds which are soft targets from attacker's point of view. Finally, unrolling also resists, to an ex-

tent, against some fault attacks. For instance, it is impossible to inject faults via a setup time violation [61, 79, 80], produced by either under-powering the unrolled module.

In the next chapter we shed some light on some complex information hiding countermeasures quite different from unrolling. Such countermeasures are called dual-rail precharge logic (DPL) and attempt to make the device activity constant and independent from the data processed.

Chapter 4

DPL Countermeasures for FPGA

A general background on the countermeasures generally deployed against SCA and FA is given in Chapter 2. In this chapter, we focus on one specific countermeasure i.e. data hiding or information hiding. Information hiding at the bit level can be achieved by a large variety of *ad hoc* encoding and protocols. However, the most convenient ones rely on a so-called dual-rail with precharge representation. The basic idea of dual-rail circuits is to add redundant logic to end up with a constant activity when sensitive bits are manipulated. The building element of all electronic devices is a CMOS gate. It is well known that a CMOS gate consumes significant power during a transition. When there is no transition at the input, the consumption is almost negligible. This difference can be seen on the power consumption curves which forms the basis for SCA. To counter SCA, dual-rail precharge logic are capable of providing a constant power consumption by adding complementary logic to the existing circuit. Along with this a two-phase operation ensures a single activity per cycle per gate such that the operations are not distinguishable on side-channel.

In what follows, the rationale of dual-rail precharge logic (DPL) is presented. Then we present wave dynamic differential logic (WDDL) in detail, one of the first DPL countermeasures well suited for FPGA implementation. The structure of WDDL, its implementation in FPGA and evaluation against attacks are discussed. An FPGA implementation of an AES crypto-processor protected with WDDL is detailed followed by its security analysis with special focus on fault attacks (FA). This is followed by a complete state of the art of DPL countermeasures deployed on FPGA. Finally, we provide an analysis of DPL in general against FA.

4.1 Dual-Rail with Precharge Logic

4.1.1 Dual-Rail with Precharge Logic Protocol

In DPL protocol, every bit a involved in the algorithm is actually mapped into a couple of nets, named (a_F, a_T), called respectively the 'false' and 'true' halves of the dual-rail variable a. The couple (a_T, a_F) alternates between two values:

1. $(0,0)$ or $(1,1)$, called NULL0 or NULL1, and designated as a NULL token, playing the role of spacer, and

2. $(1,0)$ or $(0,1)$, called VALID0 or VALID1, and designated as a VALID token, carrying the value of a.

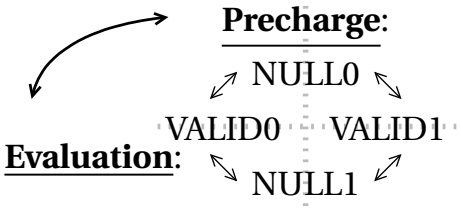


Figure 4.1: The DPL protocol showing two phase operation.

One DPL computation alternates NULL and VALID tokens, with the remarkable property that exactly one bit toggle occurs in each transition. This can be seen as a two phase operation. During precharge phase only NULL tokens are propagated through the circuit or forced at each step by a special gate. Evaluation phase comprises of propagation of VALID tokens. A pair of gates (f_F, f_T) respects the DPL convention if:

- It propagates the NULL values, *i.e.*, if all the inputs are NULL, then (f_F, f_T) is also NULL.

- It propagates the VALID values, *i.e.*, if all the inputs are VALID, then (f_F, f_T) is also VALID.

A DPL convention will ensure that there is exactly one transition per cycle for a DPL gate irrespective of the inputs. A DPL gate consists of two logic functions: a logic function (G) and its complementary OR gate (G^*), satisfying $G^*(x) \doteq \overline{G(\overline{x})}$. Some DPL allows all logical functions while others use only positive functions. A positive logic gate is one that computes an output always greater than or equal to the inputs. Such logic gates are resistant to

glitches. In case of DPL using positive gates, the inverted operations are realized by crossing between the true and false part. This category of DPL which uses positive functions are glitch free but have a cost overhead.

For sequential circuits, each flip-flop is normally replaced by a pair a flip-flops to enable two-phase operation. While the circuit is in precharge phase, 0 is propagated throughout the combinatorial part of the circuit which is then stored in the first flip-flop. The result from the previous computation is shifted from first to second flip-flop. When the circuit switches from precharge to evaluation phase, the value now stored in the second flip-flop serves as input to the combinatorial part. The corresponding output is stored in the first flip-flop (see Figure 4.5). The zero which was already present in the first flip flop is shifted to the second flip-flop, to propagate 0 during the subsequent precharge phase. The two phase operation is done in the true and false nets resulting in quadrupling of flip-flops in the DPL design. However, sometimes single flip-flop can be overused (i.e. during the rising and falling edge) in some DPL along with special precharge circuit to limit the cost overhead to two times.

Theoretically DPL is a sound countermeasure as it balances the target circuit completely and does not leak data-dependent information. In practice, it is not possible to remove each and every imbalance and thus some information is always leaked. In ASIC, most of the imbalance can be reduced to bare minimum as the designer has extensive control over synthesis, gate-level netlist, placement and routing of the circuit. Balancing is harder in FPGAs because of the fabric structure and limited control to designer over the synthesis tool. Certain constraints provided by the designer might be overridden during optimization. The common flaws in DPL implementations are discussed in Section 4.1.2 and 4.1.3.

4.1.2 DPL Flaw: Early Propagation Effect

When a DPL gate enters evaluation phase, input signals change from NULL to VALID. At this point one of the two complementary variable evaluates to '1' to attain a VALID value. Even though the gates are balanced, valid input signals can arrive at different times due to difference in logic path. Since the gate is combinatorial, the gate will evaluate without waiting for all the signals to be valid. This causes skew in the circuit further leading to data-dependent leakage on the power traces. This phenomenon is known as early evaluation (EE) [81, 82] or generally early propagation effect (EPE) because same bias may exist during precharge phase.

If the signal a is advanced or delayed compared to b, there may be an early assessment as shown in Figure 4.2. We show the delay between a and b is reflected at the output of the

DPL gate by the difference between the logical AND(G) and OR(G^*) function. Delays Δt_1 and Δt_2 can signal whether the a is 0 or 1.

Specifically, in Figure 4.2, b is always faster than a. As b is 0 (first half of the timing) or 1 (second half), the output evaluates faster(Δt_2) or less (Δt_1), suggesting its value. Also like **early evaluation**, there is a **early precharge** that makes some signals to switch to '0' faster than others.

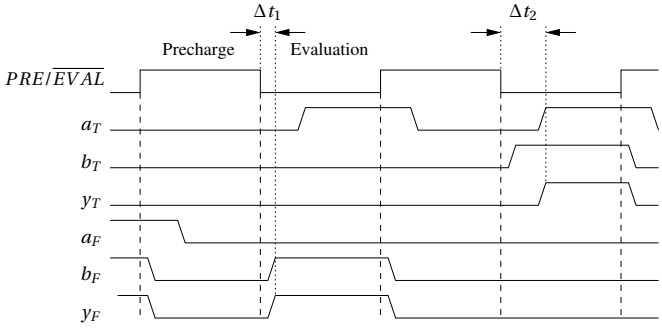


Figure 4.2: DPL AND gate timing with early evaluation.

There are two reasons why a faster than b:

1. Data b_T and b_F ago directly from a register to the gate while a_T and a_F go through a series of interconnecting channels. Therefore *switches* late, or

2. Data a_T and a_F pass through intermediate gates while b_T and b_F arrive directly.

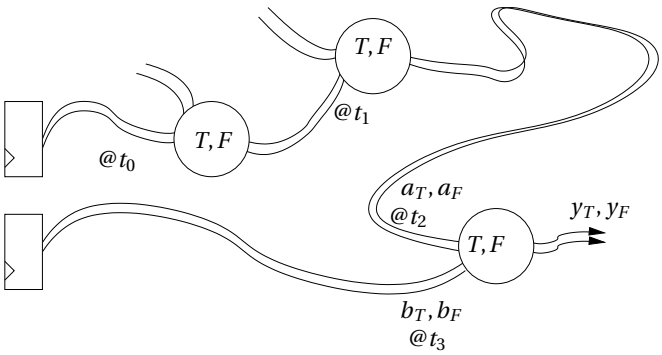


Figure 4.3: Figure to explain different propagation time of signals a and b.

Figure 4.3 illustrates the differences in paths and thus propagation delays may exist between the 2 signals a and b along the blocks T,F and interconnection lines in dual rail. The input of 2 paths is a D flip-flop synchronized on the same clock. The solution to counter

EPE is based on synchronization. If a DPL gate waits for all the signals to be VALID before evaluating, EPE can be countered. Some DPL variants deploy synchronization schemes to counter EPE which are discussed in Section 4.5.

4.1.3 DPL Flaw: Technological Imbalance

Technological imbalance in DPL comes from two major sources. The first kind of technological imbalance is a result of difference in power consumption between two the complementary gates. For example, a DPL AND gate is composed of an AND(G) gate and an OR(G^*) gate. If the consumption of the AND gate is different from that of the OR gate, there will be leakage in the side channel. For example, when the input a of a DPL AND gate is 1, the gate conducts if b is 1, if not its an OR gate. It is therefore possible to infer the value of b as AND and OR do not consume same power.

The second source of technological imbalance in DPL is the imbalanced routing within the dual-rail pair. The imbalance is caused by automatic place and route. Ideally, the true and false part of a DPL gate should be placed and routed identically. A designer does not have precise control over the placement and routing tools in a FPGA. Some routing constraints can be given to the FPGA tool but for complex designs like a cryptographic core these constraints might be overridden during optimization. Therefore to achieve a perfectly balanced routing is difficult in FPGA.

In Section 4.5, different DPL variants which have been proposed for FPGA are discussed. It also throws some light on the techniques used by these DPL variants to counter the two aforementioned implementation flaws.

4.2 Wave Dynamic Differential Logic

4.2.1 Basic theory of WDDL

Power consumption of a standard CMOS cell is dependent on the transition of its input. Thus for a DPA-resistant design, a possible solution could be to introduce a family of cell which consumes constant power like DPL. Wave Dynamic Differential Logic (WDDL) which meets all the logical constraints of a DPL was one of the first DPL logic introduced by Tiri et al. [8]. True and false representations of each signal is used (I/O of each cell). For power consumption with little or no dependency on the processed data, the gate should perform equal number of transition every cycle. To ensure this condition, alternate cycles of precharge and evaluation are used. During precharge, all signals are brought to the same logic level (*e.g.* 0

in WDDL) and exactly one of the two complementary signal is evaluated (=1) during evaluation. These characteristics are shown in Figure 4.4 with the help of timing diagram of a WDDL AND gate. We can see that during precharge all signals are put to logic 0. During evaluation, exactly one of the two complementary outputs evaluates to 1.

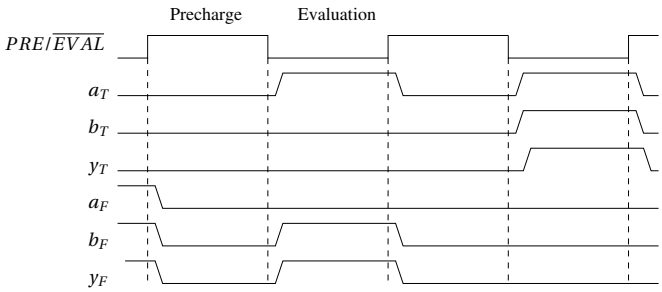


Figure 4.4: Timing diagram for a WDDL AND gate.

Glitches in a DPL design can make it vulnerable to attacks as shown by Mangard et al. [83]. If the inputs arrive at different moments, glitches can be observed. Glitches can be avoided if all the gates in the design are positive in nature. This condition is ensured in WDDL by using a synthesis library of only positive gates (like AND, OR) [84]. As shown in Figure 4.5, a WDDL AND gate is composed of an AND gate (G) and a complementary OR gate (G^*, satisfying $G^*(x) \doteq \overline{G(\overline{x})}$). As stated before each flip-flop is replaced by a pair a flip-flops to allow the propagation of precharge wave through the design. Inverters in WDDL are implemented by wire crossing between the true and false signals of a variable.

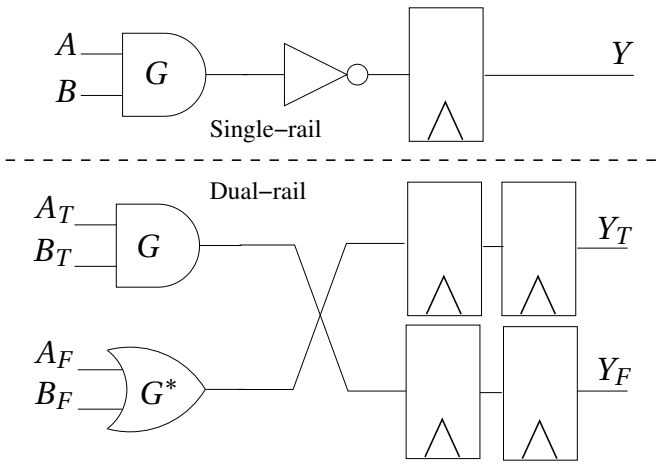


Figure 4.5: WDDL building block.

The initial state in WDDL is propagated by a wave of (0,0) couples through the netlist thanks to the use solely of positive gates. The fact that exactly one half of the gates evaluate results from the duality between the true and false networks. This duality also make WDDL especially area-efficient: each gate receives only one half of the dual-rail signals. Such a DPL logic style is called seperable where a gate takes as input either of true or false part of a DPL signal. Put differently, WDDL is a separable logic, where the instances of each dual network are not subject to a doubling overhead in terms of fan in. In addition, the positivity of WDDL ensures the absence of glitches in the complete netlist. Notice that a variant of WDDL with gates propagating the NULL spacer but without being positive is easily broken in practice, as shown by Guilley et al [84]. Presence of non-positive gates in a circuit leads to glitches which causes data-dependent leakage in the power signature. Therefore such implementations of WDDL with non-positive gates are vulnerable to SCA. Other DPL which use non-positive gates as well are sensible to glitches. To avoid glitches, these DPL logic use some synchronization schemes which monitor all the inputs from the true and the false networks. This involvement of true and false signals in the synchronization makes DPL non-separable i.e. each gate requires the true and false part of DPL signal as input. Non-separability does not pose a security threat but it is difficult to implement.

In our implementation, we use a different way to ensure all positive logic. Instead of using positive gates, we use a library containing all look-up tables (LUT) which implement a positive function. This technique was called WDDL+ as presented by Guilley et al. in [85].

4.2.2 Design Flow for WDDL Implementation on FPGA

We separate the cryptographic coprocessors into control and datapath before securing them. Leakage from the controller is not crucial as the secret key is used only in the datapath. It is sufficient to convert only the datapath from single rail to WDDL . This is done to save area as WDDL takes more area than a single-rail design. The step to convert a single rail design into WDDL are shown in figure 4.6. The datapath is first synthesized using an ASIC synthesizer taking advantage of a library with only 4-input positive LUT (the FPGA tool cannot use a custom library or a subset of its own library library therefore an ASIC synthesizer is used with a dummy primitive library). There are 166 positive functions possible with four inputs, we use one function per equivalence class[1] to number of reduce library elements. The inversions are done externally from the LUT by swapping the true and false signals (typical transformation of WDDL), and the FPGA tools can change the LUT mask to map the input

[1] where the functions can be realized by inverting the inputs

pins permutation accordingly. The output of this step is a gate netlist with only positive logic which is then fed to a custom tool (called vDuplicate in Figure 4.6) for converting the netlist from single-rail to WDDL. A wrapper is used to connect the controller with WDDL datapath. Then the FPGA tool perform mapping, placing and routing of the whole design on the FPGA.

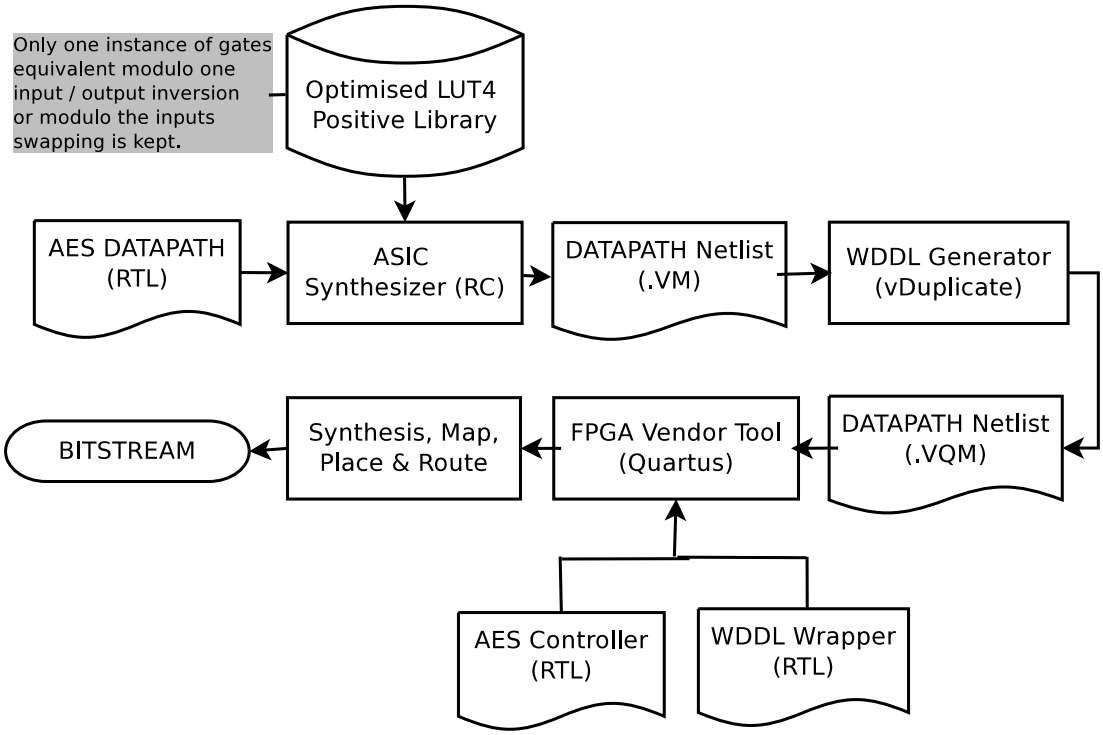


Figure 4.6: WDDL design flow for Altera FPGA.

4.2.3 WDDL Implementation and Synthesis Results

As mentioned earlier, the controller of the cryptoprocessor is not converted to WDDL. To make the same controller work with the WDDL version of the datapath, there is a need to add an extra input to the controller. As the WDDL datapath will precharge in one cycle and evaluate in the next cycle, we require the controller to work every alternate cycle (evaluation) and freeze during the precharge phase. An enable signal driven at half the clock frequency is generated to provide this functionality.

One more modification is required in the design. The I/Os of the WDDL datapath are dual-rail, while the signals from controller to datapath and the global I/Os are single-rail.

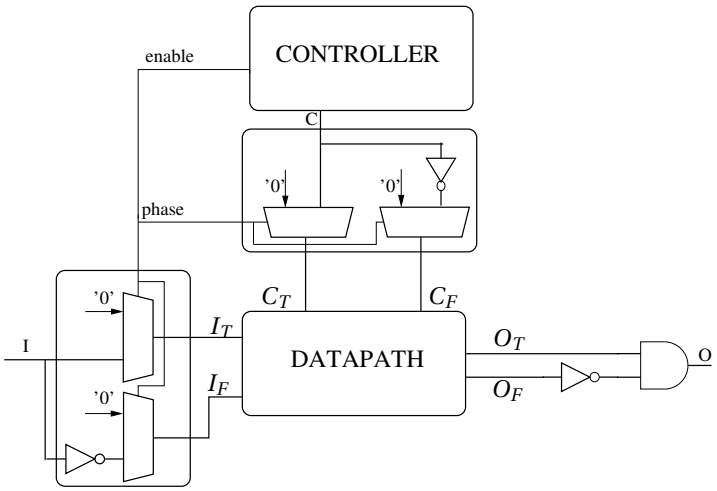


Figure 4.7: Basic architecture of WDDL wrapper.

A wrapper is deployed to make the single-rail and the WDDL parts compatible. As shown in Figure 4.7, all the inputs to the datapath (I & C) are transformed into dual-rail (true and false) signals using inverters. A signal phase is introduced to make the datapath inputs compatible with precharge and evaluation phases. When phase is precharge, both the true and false inputs are discharged to 0. During the evaluation phase, exactly one of the complementary input charges to 1. For the output (O) as shown in Figure 4.7, the true output is `AND`ed with the inverted false output. Only taking the true output while leaving false output unconnected is also an option. The reason for using both outputs is to make sure that the FPGA vendor tool doesn't remove the unconnected false output during optimization in placement and routing steps, as the optimization will create an unbalanced design.

Table 4.1: Area & Performance comparison of WDDL AES datapath with unprotected reference.

Parameters	Unprotected	WDDL	Overhead
Logic Cell (LC)	1958	11356	X5
Logic Cell Registers	256	1024	X4
Max. Frequency (MHz)	32.86	21.34	60%

Synthesis results are summarized in Table 4.1. The logic utilization is increased by 6 times from single rail (unprotected) to WDDL which also reduces the maximum operating frequency by 35%. As expected, the registers are quadrupled.

4.3 Security Evaluation of WDDL against SCA

Security evaluation of WDDL against SCA has been widely studied in literature. The vulnerability in WDDL has been studied by various researchers [81, 82, 86]. The main reason that renders a WDDL attackable by SCA is due to imbalance in its basic gate. It has been shown by simulating the power characteristics of basic gates along with validation on FPGA that the early evaluation effect comes from the difference of delay between two variables of a same gate. Fig. 4.8 illustrates the EE flaw when variable a is in advance to variable b. In this case the output does not switch at the same time.

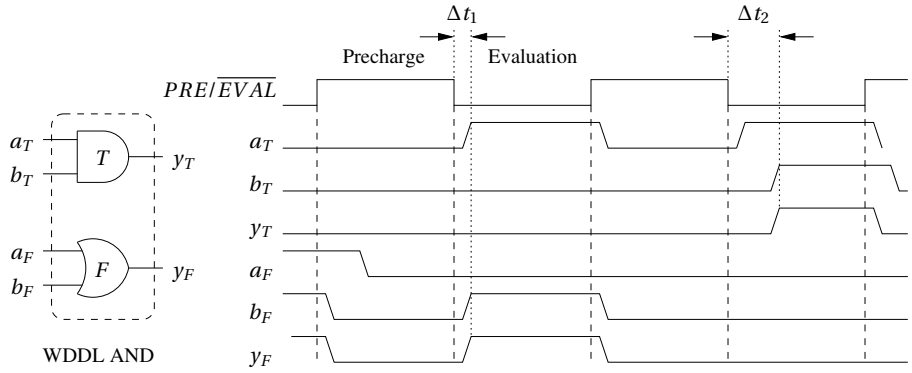


Figure 4.8: WDDL AND gate with the Early Evaluation flaw.

Moreover, the dual networks are not necessarily balanced, since the transistor structure of $x \mapsto f(x)$ and of $x \mapsto \overline{f(\overline{x})}$ differ. Those two issues have made possible some attacks on WDDL circuits, as described for instance by the authors of WDDL themselves in an ASIC [87] or independently in an FPGA [88]. Therefore, either incremental improvements or radically novel strategies have shown up. This was followed by an attack on a WDDL DES coprocessor with balanced placement and routing [89]. A significant gain was seen with balance placement and routing in terms of security but still the design is attackable. On the other hand, very little has been done to test the resistance of FA on WDDL or DPL in general. We try to analyze the resistance of WDDL against fault attacks in the following.

4.4 Security Evaluation of WDDL against FA

Various approaches have been proposed to generate fault in a target circuit. These approaches can be low cost like perturbation with power/clock lines or expensive like using

lasers or focused-ion beam (FIB). Here we analyze WDDL using the same setup as used in Section 3.1.2 i.e. making perturbing the power lines of FPGA.

4.4.1 Experimental Results

Here fault analysis is used to find the occurrence of a single byte fault that affects the state matrix of AES. Attacks on single-rail implementation was shown in the last chapter. We compare those results with similar faults on WDDL versions as tested against setup violation faults. Faults detected are those occurring only in the datapath, while the key schedule is assumed here to be fault-free. Indeed, in our design, the key schedule block is not critical in timing.

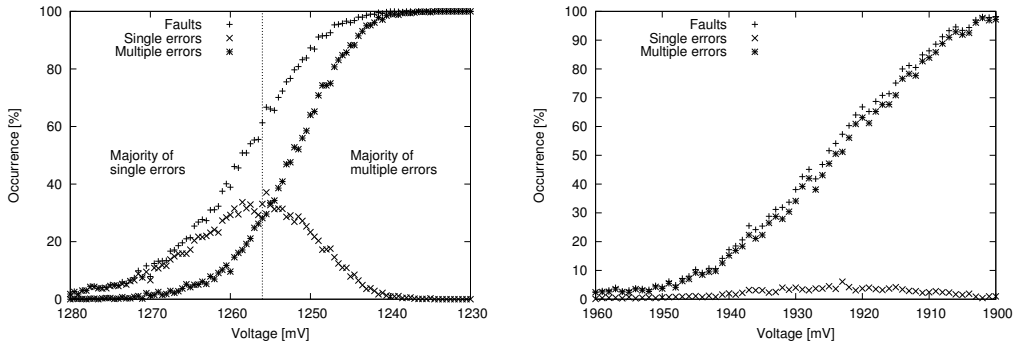


Figure 4.9: Occurrence of fault in (a) Single rail, (b) WDDL.

As already shown in last chapter, Figure 4.9(a) shows the occurrence of faults in single-rail implementation. We can see that the graph of single faults has a bell-shaped distribution. The maximum percentage of single faults is 39% at a voltage of 1.256 V as shown in Figure 4.9(a). All single faults are analyzed in terms of spatio-temporal locality: Figure 4.10(a) and Figure 4.11(a). 26% of single faults occur in round 8 and 12% of them occur in round 9 (refer Figure 4.10(a)). Such faults are exploitable using Piret's Attack. Thus the single-rail implementation of AES with s-box in LUT is not protected against "setup violation" attacks.

For AES WDDL, the results are shown in Figure 4.9(b). Since we use only positive LUT to implement WDDL, there are no glitches in the circuit. When we run the fault attack campaign on WDDL design, less than 2% of the detected faults are single. All the single faults detected are located in the last round of AES as shown in Figure 4.10(b). Such faults are not exploitable by Piret's attack or any other DFA against AES, leading to failure in retrieving the secret key. The software for fault analysis allows us to see faulted bytes and its corresponding

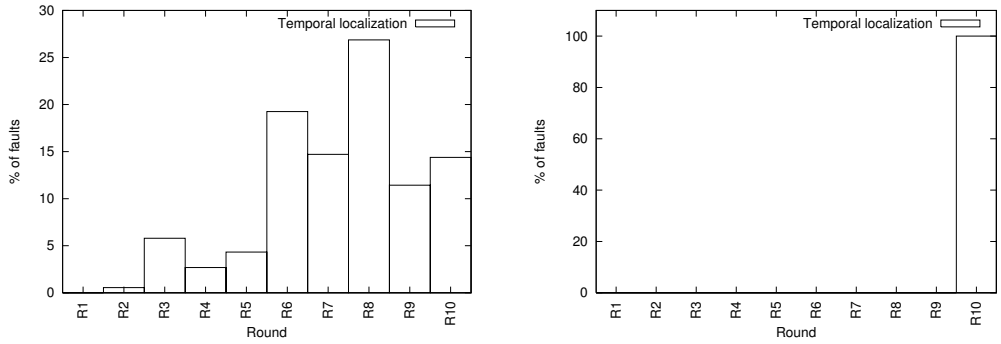


Figure 4.10: Temporal localisation of fault in (a) Single rail, (b) WDDL.

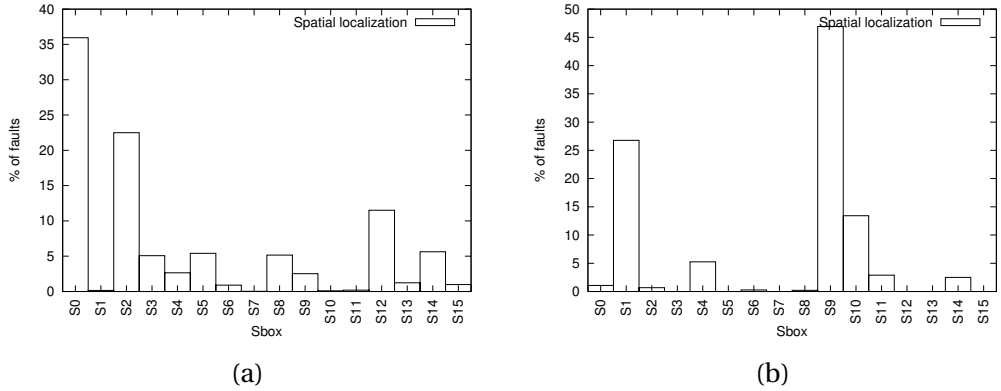


Figure 4.11: Spatial localisation of fault in (a) Singlerail, (b) WDDL.

correct value (value of byte if not faulted). We find that every time a fault occurs, the faulted value C^* is less than its corresponding correct value C, in a bit-wise sense: $C \,\&\, C^* = C^*$. This comes down to using the partial order $\preceq$, defined bit by bit in the following truth table:

C	C^*	$C^* \preceq C$
0	0	1
0	1	0
1	0	1
1	1	1

This means that all the faults are caused when an expected '1' takes a value equal to '0'. Tables 4.2, 4.3 and 4.4 show some examples of practical faults.

We have checked that the bytes faulted at value **0x00** are not due to any transmission problem of the ciphertext to the PC through the UART. Indeed, the critical path is by far in the AES and not in the UART.

Table 4.2: Single fault in round 10.

key	00000000000000000000000000000000
message	093c7b78f4fa44baff2f67fc2d259dd0
ciphertext	96296994aba80db3ea81b491230985db
ciphertext*	96296994aba80db3ea81b491230900db

Table 4.3: Single fault in round 9.

key	00000000000000000000000000000000
message	c4968c64c72bbcb88acb744253f51be7
ciphertext	43720bee23f577a8311bf769f58e97e7
ciphertext*	00720bee23f57700311b0069f50097e7

Table 4.4: Fault strictly before round 9.

key	00000000000000000000000000000000
message	be6d1ddeb2406e9a8546efc65284c4e7
ciphertext	fa73bc0ffb30e9209ec8bfe8f77b96f4
ciphertext*	00000000000000000000000000000000

We found that the presence of single faults only in the last round is not normal so we analyzed these faults. The analysis showed that detection of single faults only in the last round of AES was owing to the presence of XOR gate in the design. An XOR gate which is negative by nature can be implemented in positive logic by a combination of AND, OR gates and inverters. Inverters are required to make the inverted inputs available to the gate. Since inverters are replaced by wire crossing between the true and the false part, presence of inverters in XOR gate yield a mixture of true and false signals. Thus a fault occurring in a true part is further corrupted by mixing with the false part and vice versa. As a MixColumns operation contains many XOR operations, a fault before a MixColumns will corrupt the fault which cannot be detected. Absence of MixColumns in the last rounds make the faults occurring during last round detectable but not exploitable. We observe that for each fault on a byte in the datapath, a null byte in the ciphertext at the corresponding place. Successful single fault injection while encryption with fair control over location of the fault, does not

reveal any information to the attacker as the faulted value is erased. The results observed are easily reproducible. This means that for a particular voltage lower than the nominal voltage, if the ciphertext and input message are constant, the fault is often in the same s-box. This feature gives us better flexibility for complete analysis of these faults. Therefore, a WDDL design is naturally secure against setup violation faults. This has been further explained in the Section 4.4.2.

4.4.2 Theoretical Fault Analysis

4.4.2.1 Fault Model

In an under-powering or over-clocking attack, faults arise from a setup time violation [61, 80]. Francq et al. [67] show that a glitch on the power supply can increase the propagation times of signals globally, which is similar to the effects of global chip under-powering. As per WDDL protocol the (0, 0) spacer starts the evaluation step with all the nodes voltage equal to zero, the evaluation consists in propagating rising transitions along exactly half of the wires. If an attacker, by using any fault injection method, is successful in triggering a setup time violation, it will result in an asymmetric bit flip: only 1 to 0 errors are considered. Therefore, the consequence of the fault is that the value of one or more dual-rail signals will remain in precharge state (0,0), while the rest dual-rail signals acquire valid (0, 1) or (1, 0) evaluation state.

As already discussed in Sec. 4.4.1, it is more probable to fault only a few dual-rail signals when the stress level is level. This corrupted data will then pass to the next round logic. Four cases are possible:

1. the protocol error can turn into functional errors on the data or not, and

2. the protocol errors can vanish while flowing through the combinatorial logic (self protocol healing), or, at the opposite, be amplified.

Next we show that functional errors correspond to bits erasure. The erasure rate also increases as the fault propagates further i.e. an error at round input will trigger many erasure. We show that in a reasonable cryptographic algorithm, no computation is done uselessly and the erasure rate increases. As a consequence after some percolation in the combinatorial logic, majority of the values are erased.

4.4.2.2 Propagation of Faults

Lets take an example of two basic logic gates: AND and XOR functions, each having two inputs a and b. We assume that the fault occurs on input a and b is fault free. In evaluation, instead of having $(a_T, a_F) = (0,1)$ when $a = 0$ and $(a_T, a_F) = (1,0)$ otherwise, we have a NULL value on a i.e. $a_T = a_F = 0$. The logic equation of a WDDL AND gate is $(y_T, y_F) = (a_T \cdot b_T, a_F + b_F)$. When a is faulty, the Table 4.5 function degenerates to AND$(a^*, b) = 0$ if $b = 0$, and NULL otherwise.

Table 4.5: Modified functionality of an AND gate in the presence of erasure faults.

Correct computation								
a	b	a_T	a_F	b_T	b_F	y_T	y_F	c
0	0	0	1	0	1	0	1	0
0	1	0	1	1	0	0	1	0
1	0	1	0	0	1	0	1	0
1	1	1	0	1	0	1	0	1
Faulted computation								
a	b	a_T	a_F	b_T	b_F	y_T	y_F	c
NULL	0	0	0	0	1	0	1	0
NULL	1	0	0	1	0	0	0	NULL

Similar analysis for a WDDL XOR gate is shown in Figure 4.12. The logic equation of a WDDL XOR gate is $(y_T, y_F) = (a_T \cdot b_F + a_F \cdot b_T, (a_F + b_T) \cdot (a_T + b_F))$. According to this equation, a faulty input $(a_T = a_F = 0)$ makes the output of an XOR gate NULL $(y_T = y_F = 0)$. Thus the XOR gate has a maximum error propagation since the error is propagated for any value of b as shown in Table 4.6.

Table 4.6: Modified functionality of an XOR gate in the presence of erasure faults.

Faulted computation								
a	b	a_T	a_F	b_T	b_F	y_T	y_F	c
NULL	0	0	0	0	1	0	0	NULL
NULL	1	0	0	1	0	0	0	NULL

Now, for any function f, we have this property:

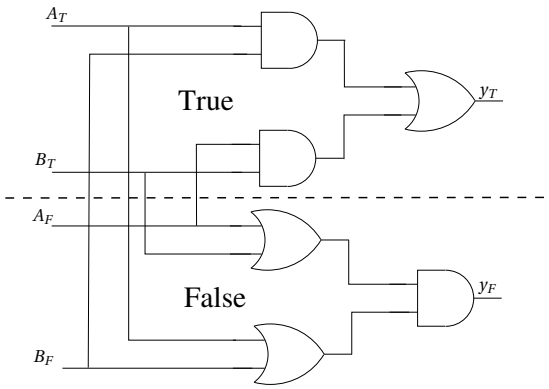


Figure 4.12: WDDL implementation of the XOR gate.

Definition Let f be a positive Boolean function with inputs (a, b) then its WDDL equivalent F can be defined as:

$$\begin{cases} F_T(a_T, b_T) &= f(a_T, b_T), \\ F_F(a_F, b_F) &= \overline{f(\overline{a_F}, \overline{b_F})}. \end{cases}$$

The output of f is correct when f does not depend on the faulty input, and erased otherwise.

The proof is straightforward. If the output does not depend on the faulty input, the output of the gate is calculated correctly for both the true and the false outputs and the fault does not interfere with the result. Otherwise, for a given non-faulty input b if F is dependent on the faulty input bit, then four cases are possible:

1. $F_T = F_F = 1$: impossible since F is positive and the inputs are lower than a legal value, that is either (1, 0) or (0, 1),
2. $F_T = 1$ and $F_F = 0$. In this case, $1 = f(0, b)$ [equation for F_T] and $0 = \overline{f(\overline{0}, \overline{b})} = \overline{f(1, b)}$ [equation for F_F] , *i.e.* $1 = f(1, b)$. Therefore $f(0, b) = f(1, b)$. However, we assumed that F does depend on the first faulty input, hence a contradiction.
3. $F_T = 0$ and $F_F = 1$: for the same reason, this case is incompatible with the input configuration such that F does depend on the faulty input.
4. Consequently, the only possibility is that $F_T = F_F = 0$, hence a NULL propagation.

Let us now study a function modeling the byte substitution table (SubBytes) of the AES. A single bit fault at the input, then:

- for one half of the input data, a specific output bit will depend on this input, and

Table 4.7: Equations for the bytes transformations ×01, ×02 and ×03.

a'	$a \times 01$	$a \times 02$	$a \times 03$
a'_7	a_7	a_6	$a_7 \oplus a_6$
a'_6	a_6	a_5	$a_6 \oplus a_5$
a'_5	a_5	a_4	$a_5 \oplus a_4$
a'_4	a_4	$a_3 \oplus a_7$	$a_4 \oplus a_3 \oplus a_7$
a'_3	a_3	$a_2 \oplus a_7$	$a_3 \oplus a_2 \oplus a_7$
a'_2	a_2	a_1	$a_2 \oplus a_1$
a'_1	a_1	$a_0 \oplus a_7$	$a_1 \oplus a_0 \oplus a_7$
a'_0	a_0	a_7	$a_0 \oplus a_7$

- for the other half, the targeted output bit does not depend on the input.

One half of the output bits are erased to NULL statistically. Notice that this result is independent of the exact functional decomposition in a positive dual gates netlist. Similarly, for two erased inputs 3/4 outputs will be NULL. And of course, when seven or eight errors are present at the input, all the output bits become NULL.

We have already shown that with XOR gates the fault propagation is maximal. The MixColumns transformation is a multiplication of a polynomial over $GF(2^8)$ with the fixed polynomial $a(x)$[4.1], reduced modulo x^4+1.

$$a(x) = (\texttt{0x03})x^3 + (\texttt{0x01})x^2 + (\texttt{0x01})x + (\texttt{0x02}) \tag{4.1}$$

The equations for the byte multiplications involved in this multiplication are written in Table 4.7. Hence we see that the MixColumns operation is implemented as a tree of XOR gates. This ensures a maximum propagation of NULL.

In an SPN (substitution permutation network) like AES, the fault number can only grow at each step. Indeed, for every block f, if a fault is stopped, then: $f(\texttt{'U'}, x)$ is certain, for a given input x. Now, this means that $f(\texttt{'0'}, x) = f(\texttt{'1'}, x)$, and this implies that f is not bijective. Therefore, differential attacks become difficult as the attacker observes an erased value, and cannot backtrack from the faulty ciphertext. The best case being when all the output bits are erased and thus no information that can be useful to retrieve the key is available.

Unlike byte-flips induced by a laser, the setup time violation on WDDL causes no computation to be wrong. Instead, when an input is partially NULL, the logic evaluates the bits

Table 4.8: Truth table for the universal gate AND.

AND	'0'	'1'	'U'
'0'	'0'	'0'	'0'
'1'	'0'	'1'	'U'
'U'	'0'	'U'	'U'

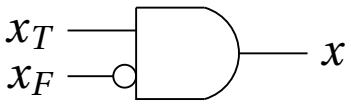


Figure 4.13: Dual-to-single rail circuitry usable in the case of a NULL0 spacer.

that can be correct for sure, but answers NULL if it cannot decide. Therefore, the propagation model is that of 'U' in VHDL [90]. The logic tries to evaluate bits that would not be wrong if any correct value ('0' or '1') were used instead of 'U'. We recall in Table 4.8 the extended truth table of the universal gate AND over $\{\text{'0'},\text{'1'},\text{'U'}\}^2$.

As shown in Fig. 4.13, the conversion of the dual-rail signals to single-rail turns a NULL into a '0'. This circuit makes use of both true and false signal halves, so as to prevent the CAD tool from simplifying half of the logic and balance the true and false networks. Therefore, if a fault occurs during the computation, it can be observed. This difference could be exploited by an attack, as done in the attack of Piret. However, the computed differential will not disclose any information about the last round key, since the XOR function used to mix it propagates a NULL.

All the considerations detailed regarding WDDL rely on the fact the gates are positive. Indeed, the gates will stick to zero when NULL values are produced at the input. Notice that in WDDL the results are independent of the type of spacer used. It can be $NULL0 \doteq (0,0)$, $NULL1 \doteq (1,1)$, used as constants or interleaved alternatively or randomly.

4.4.2.3 Generalization to Arbitrary Fault Models

We consider two categories of faults:

1. **Asymmetric faults**, where bits can only be flipped from 1 to 0. This type of faults is typical encountered in WDDL circuits stressed by a global perturbation, such as under-voltage or over-clocking. Glitch attacks can lead to the same symptom, because it manifests in adding a delay globally to all wires. Flash of white light has been

reported in [91, Ch.12, page 163] to zero selectively the output of some operations. Equally, laser shots on SRAM-based FPGAs tend to favor $1 \rightarrow 0$ bit-flips over $0 \rightarrow 1$ [92]. Notice that in DPL with a (1, 1) spacer, the opposite transition occurs when trying non-invasive attacks. We do not detail this situation as it is the exact opposite of the 1 to 0 case.

2. **Symmetric faults**, where bits are susceptible to toggling in both directions. Laser shots can trigger both 1 to 0 and 0 to 1 transitions. This fault is thus semi-invasive, as opposed to the previous ones. Therefore, it models a more powerful attacker, at least able to chemically prepare the sample to attack.

In the context of asymmetric faults, DPL circuits are natively protected as such. In this respect, it is interesting to compare the pros and the cons of synchronous and asynchronous circuits. When exposed to under-voltage, asynchronous circuits will continue to work, down to a voltage value where the gates will not be supplied enough to produce a strong 1. Below this threshold, errors of type "stuck at zero" will propagate as in the case of synchronous circuits. Overclocking is not an attack that applies to asynchronous circuits that are, by definition, clock-less. Therefore a synchronous circuit might be faulted at fairly high volatages owing a long critical path which makes it less reliable than asynchronous circuit which would work till low voltage perturbations. However, we have noticed that this perturbation is ineffective for exposing secrets. A trade-off between the two approaches can be reached by considering synchronous circuits with jitter on the clock. The jitter can have a large variance, since even if it triggers a setup time violation, the secrets remain safe. Therefore, with DPL, it is secure when used in addition with aggressive clock jitter.

If the attacker has the means to inject symmetric faults, then three types of protections must be considered:

1. When the stress level for fault induction is low, single bit flips is the most likely fault model. In this case, even if the fault is a 0 to 1 transition occurring during the evaluation stage, the only risk is to create a (1, 1), also called NULL1. However, in a dual way of the case study of the propagation of NULL0 values, we can show that the propagation of NULL1 consist in an erasure of the data, so that the syndrome does not convey any single bit of information about the faulty circuit internal state. DPL style thus forces the attacker to be less furtive.

2. With a more intense stress, the attacker will start to induce multiple faults with low multiplicity. In this case, a DPL gate can output completely false values. For instance,

an AND gate for which the inputs are NULL0 and NULL1 evaluates to the correct value 0 (with respect to WDDL valid states), even if the two non faulty inputs were both equal to 1. To protect the implementation against those attacks, additional detection hardware must be added so as to cross-check the computation. A little gain can however be obtained: As the DPL style is protected against single faults, a datapath of n bits can be checked with code words of only $n-1$ bits without risking to weaken the security level. A protection method at the technological level such as the one presented in [93] could be extended from SRAM points to DFFs and combinatorial gates. By using high-VT P transistors (those that compute the `'1'`) and low-VT N transistors (those that compute the `'0'`), the designer could make the faults $1 \rightarrow 0$ much more likely than the opposite $0 \rightarrow 1$.

3. When the stress is very strong, then we expect the faults to be very frequent. Hence the recommendation to use sensors spread on the chip surface.

Now, if we consider only asymmetric faults, we could think that power analysis could be made possible by fault injection. Indeed, if DFA does not expose the key, it at least indicates to the attacker that a fault has happened. More precisely, we could imagine to correlate the amount of detected faults to a side-channel, in a view to establish correlations. Indeed, in nominal operation conditions, the activity is constant: half of the gates commute in each clock cycle. When a fault is injected, the activity will become lower:

- in a fault position dependent fashion (for sure), as illustrated in Fig. 4.14,
- but perhaps also in a data dependent fashion.

However, such an attack cannot be mounted, since if a sensitive variable is faulty, irrespectively of its value, the fault will generate a NULL0. Therefore, after the fault, the system has forgotten its value, and computation (in terms of number of toggles) will continue in similar ways. This argument is confirmed by the practical observation of power consumption of WDDL AES as shown in Figure 4.15. We can see that the power consumption of the device is abruptly reduced as soon as the fault occurs approximately at time 2130 ps. The power consumption further reduces after two cycles and remains constant till the end of encryption. It takes exactly 2 cycles (1 ShiftRows and 2 MixColumns) for NULL0 to diffuse through the whole design. This holds even if the DPA protection has a second order flaw, such as early evaluation. The only way to take advantage of such a flaw is to exploit it without faults. Indeed, to rephrase why DFA does not help the DPA, with faults, the distinctions

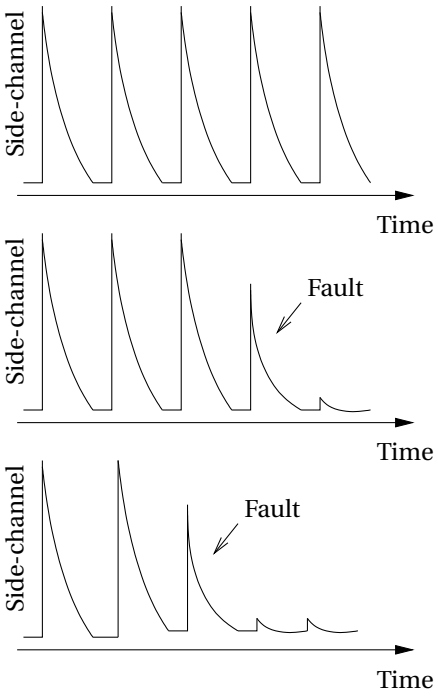


Figure 4.14: Power dependence of a WDDL circuit in the faults.

of power curves at second order simply disappear. We cannot show any experimental curve to illustrate this point since we have no mean to deduce the bit concerned with the fault based on the sole knowledge of the ciphertext.

Finally, we attract the reader's attention to the fact that vulnerability analysis of WDDL against faults exploitation or DPA in the presence of faults has been argued in the precharge to evaluation step. However, it can be transposed without any change to the case of evaluation to precharge step. Indeed, the circuit's behavior is unchanged, except that vulnerable transitions, previously $0 \rightarrow 1$, are replaced with $1 \rightarrow 0$. However the attacker has less insight, since faults occurring in the precharge stage cannot be observed, that are filtered out by the WDDL circuit wrapper.

4.5 DPL: State of the Art

Although WDDL is one of the oldest and most popular DPL style, many other DPL styles have been proposed later, both for ASIC and FPGA implementations. Here we just focus on DPL applicable to FPGA. Some of the these DPL styles differ from WDDL in the precharge circuit, while other add special circuits to overcome the flaws like imbalanced routing. There

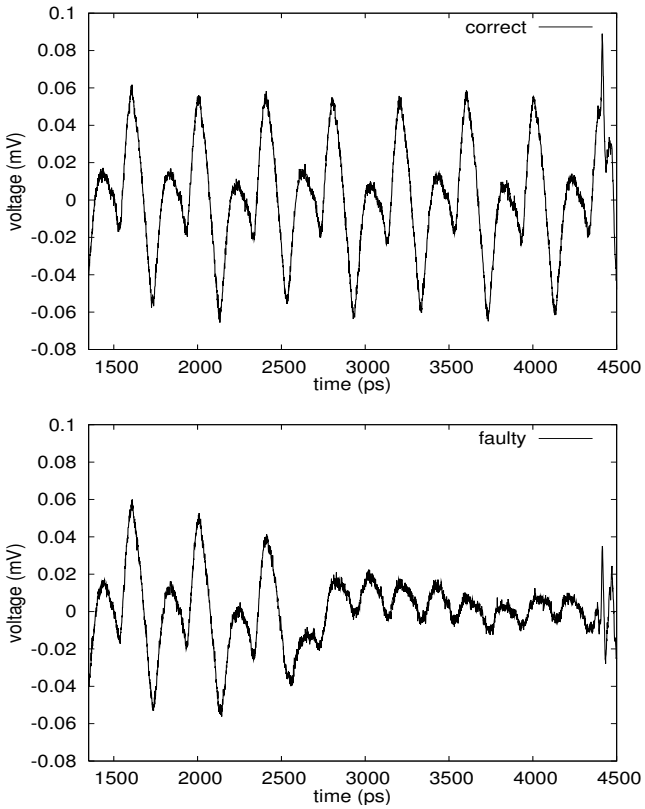


Figure 4.15: Practical power consumption of a WDDL circuit in the presence of faults.

are other DPL styles which are a direct extension of WDDL to reduce either complexity or routing imbalance etc.

4.5.1 WDDL Variants

Some variants of WDDL have also been devised to ease the balance of WDDL networks. However, as already explained in the subsection devoted to MDPL, it is known that balancing the WDDL interconnect does not solve the early evaluation (EE) inherent to this logic [94]. Nevertheless, we introduce them here because some of these logic styles have unexpected positive side-effects on their security w.r.t the EE.

4.5.1.1 DWDDL

Double WDDL (DWDDL) is introduced in [95] by Yu et al. to counterbalance one unbalanced network. As the name suggests, DWDDL proposes to route two WDDL designs (one

direct and other inverted) adjacent to each other. This solution is sound in theory. The overhead associated with the further duplication of hardware in DWDDL can be reduced by substitution box (s-box) as designer in [96]. These s-boxes are similar to the WDDL in BDD-style presented in [97], which allows for a separation between the true and false halves thus allowing for a copy-and-paste routing to guarantee the same back-end.

4.5.1.2 WDDL with Divided back-end duplication

WDDL is area-efficient since the number of combinatorial logic gates is merely doubled with respect to the unprotected initial version. Indeed, the transformation from unprotected single-rail to protected dual-rail consists, for the combinational instances $x \mapsto f(x)$, to substitute them with a couple of dual gates $(x \mapsto f(x), x \mapsto \overline{f(\overline{x})})$ [98]. However, the counterpart is that the gates must be non-inverting, so as to propagate the precharge wave throughout the netlist. Thus, inverting gates are replaced by their non-inverting counterparts, while the undesirable inverters are replaced by wire crossings between the true and false wires. This procedure is described in [84]. This makes it impossible to separate WDDL into two unconnected netlist.

The "divided back-end duplication" technique [99] is an attempt to separate WDDL into two halves that do not communicate one with each other. This way, the design can be placed-and-route for one half, and then copied and pasted for the second half, where gates are replaced by their dual function. For this purpose, the inverters are not replaced by wire-crossings, but by controlled inverters; such gates are:

- inverters in the evaluation phase, and
- buffers in the pre-charge phase.

Thus, the netlist is properly precharged. For this behavior to be implementable, a global phase signal is required. It is called EVAL, and must be faster than all the data changes. Indeed, its role is to cadence the DPL protocol. It is equal to 1 in the evaluation phase and to 0 in the precharge phase. The inverter $y_T = \overline{a_T}$ is thus replaced by a gate that computes a_T when EVAL = 0 and $\overline{a_T}$ when EVAL = 1. This gate is thus an exclusive or gate: $y_T = a_T \cdot \text{EVAL} + \overline{a_T} \cdot \text{EVAL} = a_T \oplus \text{EVAL}$. The transformation of an unprotected function into a WDDL and a divided back-end duplication function is illustrated in Fig. 4.16, for the example of the mapping $y = a \cdot \overline{(b+c)}$.

Nonetheless, this strategy is not secure, since the fundamental hypothesis of constant activity logic is violated. We illustrate this on the previous example, where (a, b, c) evaluates

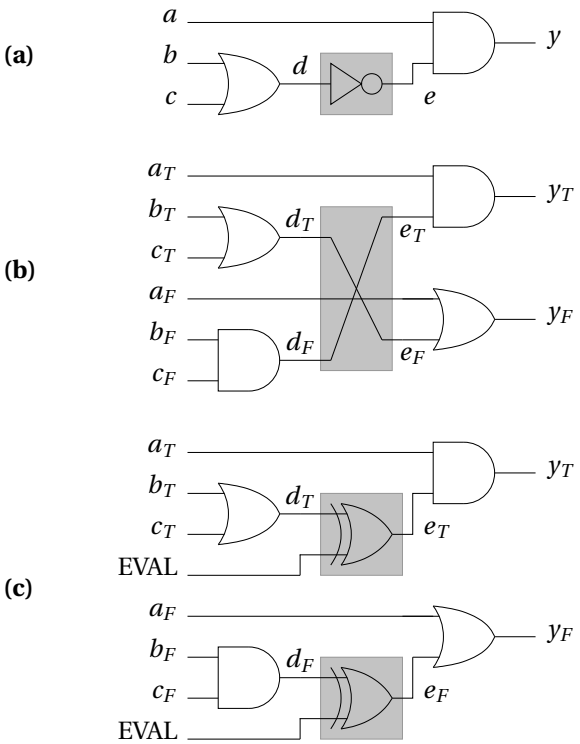


Figure 4.16: Combinatorial function $y = a \cdot \overline{(b+c)}$ in **(a)** unprotected standard cell synthesis, **(b)** WDDL and **(c)** divided back-end duplication. The inverters are highlighted in a grey box.

to either $(0,1,1)$ or $(1,1,1)$. In other words, the toggling count of this small netlist depends on the sensitive variable a, which permits in principle a side-channel attack. As shown in the simulation depicted in Fig. 4.17,

- the net y_T has a non-functional $0 \rightarrow 1 \rightarrow 0$ transition (*aka* glitch) if and only if $a = 1$ in precharge [event noted (a)], and
- has the same bias in precharge [event noted (b)], with in addition an early (*respectively* late) transition if and only if $a = 1$ (*respectively* $a = 0$) [event noted (c)].

Thus, this example shows that some configurations of some netlist leak more than an unprotected implementation. Indeed, the difference of toggling count is $|1-0| = 1$ when the netlist is unprotected whereas it is $|(1+2)-(1+0)| = 2$ when protected. Furthermore, the EPE is still present in this netlist. All in one, the security of the netlist is degraded, since it has conditional glitches. Regarding the data-dependent delays, the divided back-end duplication method has helped fixed the second-order issue of true/false pairs routing balancing while nonetheless keeping the first-order flaw related to EPE.

Similar flaws can be observed when the netlist evaluates to $(0,0,0)$ or $(1,0,0)$. As shown in Fig. 4.18,

- when $a = 1$, there is a glitch on y_F during evaluation [event noted (a')] and an early evaluation of y_T [event noted (b')],
- and the same happens in return to precharge: when $a = 1$, there is a glitch on y_F [event noted (c')] and an early evaluation of y_T [event noted (b')].

In order to properly balance the delays, two conditions shall be fulfilled:

1. Along the interconnection network, when the true and false signals are synchronized at the beginning, they should remain synchronized at the end of the path; This condition is made explicit in Fig. 4.19.
2. The true and false values shall be outputted at the same date for different input data, which is all the more important as the dual-rail inputs do not arrive simultaneously at the gate entrance. This requirement is illustrated in Fig. 4.20.

The first condition is definitely enhanced by the divided back-end balancing logic. However, the second condition is definitely not respected, as it inherits from the EPE issue of WDDL. The non-respect of this condition is more dramatic than the non-respect of the first condition, because it induces a bias that gets worse and worse with the logical depth of the datapath.

4.5.1.3 IWDDL

Eventually, isolated WDDL (IWDDL) [100] is a different strategy to separate a WDDL netlist into two unconnected halves. Here, inverters are kept but potential glitches are stopped by systematically inserting one register after it. This strategy is expensive in terms of area and requires a redesign of the controller. Additionally, the design becomes much more pipelined, which requires much higher clock frequencies to maintain an acceptable throughput. However, the benefit of this approach is to stop also the propagation of the EE wave. Apart from the very poor performance of IWDDL, this method is however very strong from a pure security standpoint. Only one point is questionable: isn't the complete separation of the netlist opening the door to well located EMA attacks, that can record selectively the activity from only one half of the netlist, thus defeating the activity invariability property. This issue is all the more stringent as the netlist is much larger in IWDDL than in WDDL, because of the large quantity of registers added for the pipeline.

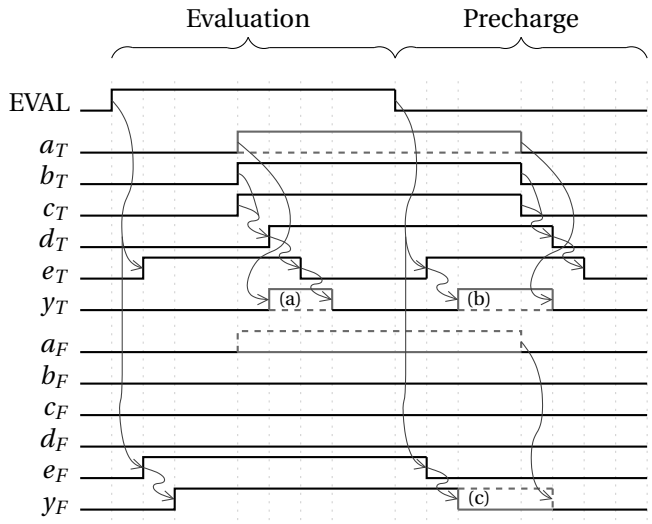


Figure 4.17: Timing Diagram of execution of the netlist depicted in Fig. 4.16**(c)** for $(a,b,c) = (0/1,1,1)$.

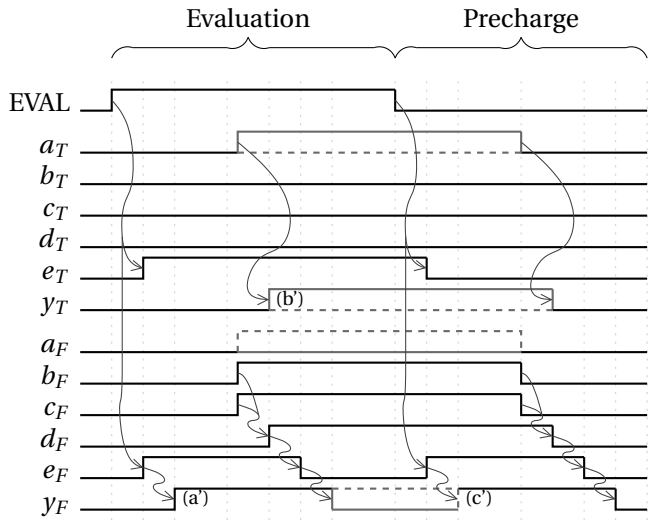


Figure 4.18: Timing Diagram of execution of the netlist depicted in Fig. 4.16**(c)** for $(a,b,c) = (0/1,0,0)$.

4.5.2 SDDL

Simple Dynamic Differential Logic (SDDL [8]) allows the usage of negative logic thus making it less complex and more flexible than WDDL. The problem in SDDL is that the precharge wave is stopped when a negative logic appears in the circuit. All the nets beyond a negative

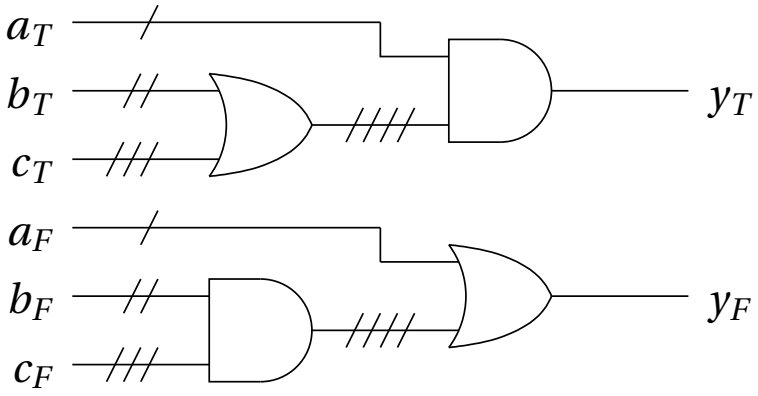


Figure 4.19: Pairwise balance of dual-rail pairs in a DPL netlist.

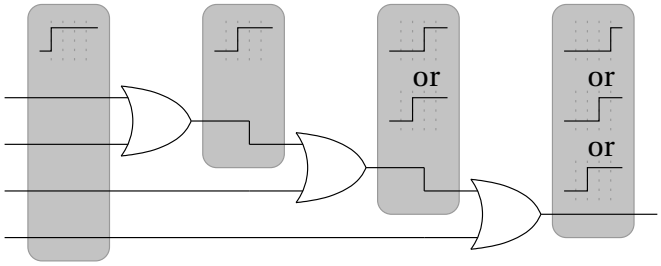


Figure 4.20: Example of netlist (*e.g.* one half of a WDDL netlist) prone to early evaluation effect.

logic gate are not precharged. To fix this issue, a dedicated pre-charge circuit is added to each negative logic gate so that the precharge wave could affect all the nets. SDDL does not have any cross connection and its separable which allows direct and complementary logic to be routed symmetrically. Another weak point of this logic style also comes from usage of negative logic. Negative logic can produce glitches therefore SDDL cannot guarantee one switching event per clock cycle. WDDL is considered to be a more secure logic style than SDDL because it is glitch free. Area efficient FPGA implementations of SDDL which can also use Block RAMs are addressed in [64]. Also copy-and-paste nature of this logic is a security loop-hole. A well equipped attacker may isolate the direct logic from complementary logic using precise EM probes which makes the attack possible.

4.5.3 Partial DDL

Recently a low cost-variant of SDDL called Partial DDL was introduced in [101] by Kaps et al. A normal approach in designing DPL cryptographic cores is to implement datapath which

contains key-dependent data in dual-rail while the control is left in single-rail. A wrapper is deployed to connect the secured dual-rail datapath to single-rail controller. Partial DDL further reduces the part of circuit which needs to be secured. Authors propose to divide the datapath according to leakage model. Indeed there are some parts of the circuit which are easier to model while others are not. To reduce implementation cost, only the part of the datapath which can be modelled easily is converted into dual-rail. Rest of the circuit remains single-rail. The net cost of partial DDL is less than twice of single-rail design. Authors have chosen SDDL as the DPL used to protect the part of datapath. As the direct and complementary part of the protected datapath are placed on different sides of unprotected datapath attack by isolation is possible as in SDDL.

4.5.4 MDPL

Popp et al. proposed masked dual-rail with precharge logic (MDPL [46]) as an attempt to fix the imbalance of WDDL. The assumption is that, in some conditions, it can be difficult to constrain a router to balance the differential interconnect. Indeed, the two solutions available in the literature, namely the fat wire [102] and the back-end duplication [103] methods, apply primarily to ASIC. The transposition to FPGA is possible, albeit with less fine-grain control over the result [96]. For this reason, MDPL proposes to swap the true and the false routes randomly, so as to overcome the fatal routing imbalance. By the same token, it makes up for the structural unbalance of the dual pair of gates. The only gates involved in the logic are majority functions, both for the true and the false networks. Nonetheless, MDPL fails to provide a solution to the early evaluation and precharge of WDDL. Another problem with MDPL and MDPL-based logic is that they use a single bit of mask which adds only 1-bit entropy against SCA. Higher order attacks can be used to attack such logic as shown in [104, 105, 106].

4.5.5 DRSL

The primary focus of Dual-rail Random Switching Logic (DRSL [107]) which was introduced by Chen et al. is to make the evaluation and the precharge gates data-independent. For this reason, one pairwise unanimity gate[1] computes the validity of all inputs prior to allowing the gate from delivering any result, thus avoiding the EE flaw. On the contrary, the unanimity makes it possible for the overall DRSL gate to always anticipate the precharge. However, in the original design of DRSL, the functions are not required to be positive. The example of

[1] The pairwise unanimity Boolean gate performs the following computation: $(x_T, x_F, y_T, y_F, \ldots, z_T, z_F) \mapsto (x_T + x_F) \cdot (y_T + y_F) \cdot \ldots \cdot (z_T + z_F)$.

the AND function is sketched in Fig. 4.21(a). Hence the presence of data-dependent glitches in the return to precharge phase.

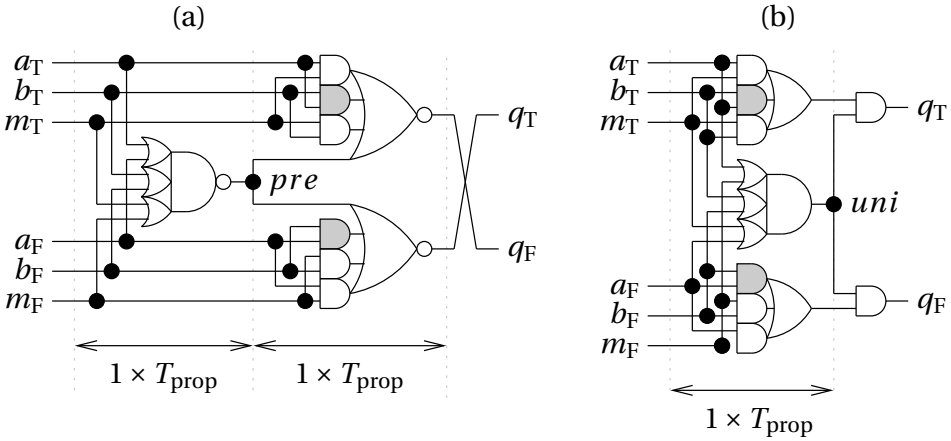


Figure 4.21: (a) Genuine DRSL AND gate and (b) Glitch-free variant.

We carried out an extensive simulation of the DRSL AND gate when it returns to precharge. Table 4.9 shows the situation where the mask is the fastest to return to NULL. More precisely, we assume that m returns to precharge first, followed by a and b in this order, which we abbreviate as $t_{m\to\text{NULL}} < t_{a\to\text{NULL}} < t_{b\to\text{NULL}}$. It happens that the DRSL AND gate glitches iff $a \oplus b = 1$, irrespective of the mask value.

Notice that it could have been anticipated that the glitch property does not depend on m if the mask is particular (*e.g.* the fastest signal) and a and b are equivalent. Indeed, when $m = 0$, there will be a glitch pattern for the DRSL AND gate computing in the direct convention, whereas when $m = 1$, the glitch pattern will correspond to a complemented interpretation for the functional signals a and b. As, by design of DRSL, the attacker cannot make the difference between a transition occurring on a true or a false wire, the glitches will be observed without any distinction for each value of m. As a return to NULL with a glitch consists in three transitions (one functional plus two non-functional), whereas a return to NULL without a glitch consists in a single transition, a correlation of the traces with the value $a \oplus b$ will yield a peak. Assuming that $a \oplus b$ is sensitive and predictable, this correlation is a means to test hypotheses.

We also studied other types of transitions ordering. In the cases where $t_{a\to\text{NULL}} < t_{m\to\text{NULL}} < t_{b\to\text{NULL}}$ or $t_{a\to\text{NULL}} < t_{b\to\text{NULL}} < t_{m\to\text{NULL}}$, the DRSL gate also features glitches, when $b \oplus m = 1$, irrespective of the value of variable a. But given that m is an unknown quantity, these glitches do not convey any information about the value of b. The glitches are thus innocuous in these cases. There is however a possible flaw if b is known (*e.g.* it is a primary

Table 4.9: Exhaustive simulation of all the return to NULL cases in the DRSL NAND gate Glitches are indicated in **boldface font**.

Mask=0: Direct function					**Mask=1: Dual function**				
$a=0$	$b=0$	$m=0$	pre	q	$a=0$	$b=0$	$m=1$	pre	q
0 1	0 1	0 1	0	01	0 1	0 1	1 0	0	01
0 1	0 1	0 0	0	01	0 1	0 1	0 0	0	01
0 1	0 1	0 0	1	01	0 1	0 1	0 0	1	01
0 0	0 1	0 0	1	00	0 0	0 1	0 0	1	00
0 0	0 0	0 0	1	00	0 0	0 0	0 0	1	00
$a=0$	$b=1$	$m=0$	pre	q	$a=0$	$b=1$	$m=1$	pre	q
0 1	1 0	0 1	0	01	0 1	1 0	1 0	0	10
0 1	1 0	0 0	0	**1**1	0 1	1 0	0 0	0	1**1**
0 1	1 0	0 0	1	**1**1	0 1	1 0	0 0	1	1**1**
0 0	1 0	0 0	1	00	0 0	1 0	0 0	1	00
0 0	0 0	0 0	1	00	0 0	0 0	0 0	1	00
$a=1$	$b=0$	$m=0$	pre	q	$a=1$	$b=0$	$m=1$	pre	q
1 0	0 1	0 1	0	01	1 0	0 1	1 0	0	10
1 0	0 1	0 0	0	**1**1	1 0	0 1	0 0	0	1**1**
1 0	0 1	0 0	1	**1**1	1 0	0 1	0 0	1	1**1**
0 0	0 1	0 0	1	00	0 0	0 1	0 0	1	00
0 0	0 0	0 0	1	00	0 0	0 0	0 0	1	00
$a=1$	$b=1$	$m=0$	pre	q	$a=1$	$b=1$	$m=1$	pre	q
1 0	1 0	0 1	0	10	1 0	1 0	1 0	0	10
1 0	1 0	0 0	0	10	1 0	1 0	0 0	0	10
1 0	1 0	0 0	1	10	1 0	1 0	0 0	1	10
0 0	1 0	0 0	1	00	0 0	1 0	0 0	1	00
0 0	0 0	0 0	1	00	0 0	0 0	0 0	1	00

input, such as one bit of the plaintext). In this case, the value of the cryptoprocessor-wide mask bit (*i.e.* only one bit of entropy is used for the routing randomness) can be estimated by classifying the traces according to their intensity, as in [104].

However, the situation where the mask is the fastest to return to zero is the most likely, for at least two compelling reasons:

1. As the mask is global (shared by all the protected gates), it is amplified and therefore propagates very fast, in a similar way as a clock signal.

2. Also, the mask is directly available at one register's output, whereas the data signals can traverse many other DRSL instances prior to arriving at the gate's inputs.

Two solutions can be imagined to patch the glitching problem of DRSL. The first one consists in adding buffers to delay the signals so as to balance the paths within the DRSL gate. This solution is however technology-dependent. Another option consists in implementing DRSL in positive logic, as shown in Fig. 4.21(b). This solution has a cost in CMOS logic, because inverting gates are smaller than non-inverting ones (actually realized in practice by the sequential[1] composition of an inverting gate with an inverter [108]). However, this is not constraining in FPGA. A loss in area is nonetheless expected, as the functionality can only consist in positive gates, thereby limiting the degree of freedom of the logic synthesizers. In this case, the new logic, that we name DRSL+, consists in MDPL augmented with a synchronization by an unanimity cell. The equation for the AND gate becomes:

$$\begin{cases} q_T &= \underbrace{(a_T + a_F)\cdot(b_T + b_F)\cdot(m_T + m_F)}_{\text{shared unanimity cell}}\cdot \\ & \quad \underbrace{(a_T \cdot b_T + b_T \cdot m_T + m_T \cdot a_T)}_{\text{masked direct functionality}}. \\ q_F &= \underbrace{(a_T + a_F)\cdot(b_T + b_F)\cdot(m_T + m_F)}_{\text{shared unanimity cell}}\cdot \\ & \quad \underbrace{(a_F \cdot b_F + b_F \cdot m_F + m_F \cdot a_F)}_{\text{masked dual functionality}}. \end{cases} \tag{4.2}$$

DRSL+ gate is not an implementation-level correction of DRSL. Instead, DRSL+ really changes the functionality of DRSL (*i.e.* the Boolean equations for (q_T, q_F) are not the same). It is straightforward to check that the DRSL AND gate is not positive, whereas equation (4.2)

[1] *Sequential* in the sense of "one after the other" and not opposed to *combinational* – the assembly of the two gates do remain combinational.

is, given the sole use of Boolean AND ($\cdot$) and OR ($+$) functions involved in the expression[1]. Surprisingly enough, this correction comes with hardly significant overhead. Indeed, the DRSL+ style does not forbid the use of inverting CMOS standard cells. As a matter of fact, the Boolean functionality of DRSL+ can be mapped entirely with standard cells.

Another attack against DRSL is presented in [109]. Actually, this attack puts forward a vulnerability that is common to all masked DPL styles. The idea is that the masking of the gates allows to make up for the routing unbalance. However, the mask signal is itself differential and therefore unbalanced. As it is not balanced (since this is the hypothesis when resorting to masked DPL), it paradoxically opens the door to an attack on itself.

4.5.6 STTL

Secure Triple Track Logic (STTL) was presented in Soreas et al. [110] eludes any glitching risk by waiting to evaluate and to precharge until all the inputs are either valid or NULL. This incurs useless delays in the return to precharge phase, which is however only detrimental to performance, not to security. The main drawback of STTL is the requirement to route one synchronization signal slower than the dual-rail, while granting a balanced routing within the dual-rail pair. However, the known methods to balance signals (fat wire and back-end duplication) operate on a full netlist, and are therefore difficult to adapt on heterogeneous netlist, in which single-ended and dual-rail signals are mixed up.

4.5.7 DPL styles Comparison

Table 4.10 draws up a comparison of the main DPL styles, in terms of principle, design constraints and performance, highlighting most of the known advantages (masking, synchronization) and drawbacks (primitives and back-end constraints, and technological bias) of such countermeasures.

Masking allows to greatly reduce the technological bias, but also results in a significant increase of area. As a matter of fact, it requires at least a transformation of 2-input operations into 3-input majority function (MDPL) or into a 4-input RSL gate (DRSL).

Synchronization on both precharge and evaluation is mandatory to avoid glitches and early propagation effects.

Primitive constraints induce a higher complexity, by reducing the panel of usable functions (like in WDDL where only positive functions are allowed), or by binding the designer

[1] This remark also explains why we preferred to use (X_T , X_F) notations instead of (X_m, $\overline{X_m}$), where it is not clear that $X_F \leftrightarrow \overline{X_m}$ is a positive monomial.

Table 4.10: DPL performance and security features overview.

Logic	Mask	Synchro		Constraints		Tech	Speed
		Pre	Eval	Primitives	Back-end	Bias	
WDDL	no	✗	✗	positive funct only	balanced place&route	high	$< 1/2$
SDDL	no	✗	✗	no	copy & paste	low	$< 1/2$
Partial DDL	no	✗	✗	no	copy & paste	low	$< 1/2$
MDPL	yes	✗	✗	MAJ †	no	no	$< 1/2$
STTL	no	✓	✓	no	delay on sync signal	very low	$< 1/4$
DRSL	yes	✓	✗	no	no	no	$< 1/2$
IWDDL	no	✓	✓	no	netlist post-processing	low	$< \frac{1}{2 \cdot n_i}$ ‡

† MAJ stands for the majority gate: $MAJ(a,b,c) \doteq a \cdot b + b \cdot c + c \cdot a$.
‡ n_i is the maximum number of inverters amongst all combinatorial paths.

to use specific functions that can be more area-consuming or slower than basic ones (Seclib, MDPL, DRSL).

Back-end constraints generate extra design work as the P/R stage has to meet specific requirements to achieve a good balance between the T and F networks. It can also cause a loss of performance, like in STTL where the synchronisation signal must be manually made slower than the others, by adding delay elements between each gate, in order to ensure that it always switches last.

Technological bias corresponds to the imbalance between the True and False networks. It encompasses the load, interconnect and CMOS structure differences. This is a significant source of information leakage, and must therefore be as low as possible to ensure a secure countermeasure.

Based on the study of the state of the art of DPL countermeasures we can classify DPL into two distinct categories:

- DPL with Early Evaluation (DPL w/ EE) are the DPL styles which suffer from EE flaw. In general, its EPE which occurs during precharge or evaluation. The phenomenon is more common during evaluation. DPL styles of this kind may evaluate an output without knowledge of all the inputs and create imbalance. WDDL, SDDL, Partial SDDL, MDPL, IWDDL and DWDDL belong to this category.

- DPL without Early Evaluation (DPL w/o EE) are DPL styles which depend on a certain synchronization scheme to counter EE. Such DPL styles can evaluate only when all

the inputs are known. DRSL and STTL fall under this category.

In the next section, we extend the results of resistance against faults from WDDL to DPL in general.

4.6 Security Evaluation of DPL against FA

4.6.1 Fault Model

Most, if not all, fault attacks reported in the literature, use a single perturbation source to generate faults within the FPGA. Basically, the perturbation responsible to place the target device out of specified operating conditions is either global or local. Global perturbations consist in varying one environmental variable, such as the power supply, the clock frequency or the external temperature. The perturbation can be steady or transient. But in either case, the source of faults is not precise: the complete circuit is faulted altogether. Local perturbation are more difficult to create, because they require an access to the silicon die surface. This condition means that a mechanical/chemical preparation of the circuit must be done beforehand. Such a step is reserved to advanced laboratories that have access to specialized facilities. Moreover, the preparation cannot be achieved with 100% success probability, which drastically increases the cost of the attack. Nonetheless, even if open samples are available, equipment able to inject a localized fault is often large. For instance, a laser source and its focal dimensions limit the minimum distance between two faults. We would like to underline that it is anyway very difficult if not impossible to resist against coherent multiple faults injection. By coherent multiple faults injection we mean symmetrical fault on a dual-rail pair. Any protection mechanism, based on either spacial or temporal redundancy can be abused. Similarly, when a parallel path uses an encoding to check for the data integrity, consistent faults can be injected to change a code word for another one. We assume in the sequel that multiple faults can be generated locally (by means of a laser or an electromagnetic injection [111]), but decorrelated one from each other.

However, if we imagine that it is possible with some sophisticated equipment to inject related multiple faults, which has by the way never been published so far, it is not sure that the antinomic bit-flips (i.e. opposite bit-flips on a two signal) can be obtained. Indeed, the only way to trick the DPL w/o EE logic is to replace a VALID token (0, 1) by another VALID (1, 0). If a VALID token is replaced by NULL, DPL w/o EE will propagate NULL. Now, with two spatially close injectors, it is far from obvious that the faults will not negatively interfere. Indeed, the way to flip a 0 into a 1 is to inject energy at the correct wavelength in a N^+ doped

region whereas to flip a bit the other way round, the energy shall be injected in another well, possibly at a different wavelength. If we take the example of electromagnetic (EM) injection [111] with micro metric probes, it is expected that opposite fields must be generated to trigger contrary bit flips. However, this also means that the perturbation merely cancels itself due to the proximity of the two regions to excite. In any case, given the lack of literature about this subject and without any proof-of-concept experimental feedback, it is hard to further speculate on the feasibility of such coherent fault injections. Therefore it is safe to consider such a vulnerability as highly implausible, and it can thus be ignored in a short to medium term. In summary, we continue our analysis by assuming that multiple faults can be generated locally, but decorrelated one from each other. Nonetheless, some DPL styles, such as the divided back-end duplication [99] or the Isolated WDDL [100] separate on purpose the two DPL circuit halves. This strategy, that eases the designer, also helps the attacker as dual fault injection becomes easier.

4.6.2 Faults Transformation

In Section 4.4, it is shown that WDDL is immune against multiple asymmetric faults such as those caused by setup time violations. Basically, the idea is that asymmetric faults are able to turn any VALID token into a given NULL one. For instance, the fault can induce a mutation from any VALID to the NULL0 spacer. The NULL token can propagate until the outputs, being even amplified. However, the NULL wave propagation acts as an eraser, which means that the outputs have eventually lost any information about the faulted values. A parallel is done in Section 4.4.2.2 between asymmetrical faults and the logical propagation of `'U'` value in the 9-valued type `std_ulogic` of VHDL (IEEE standard number 1076).

We add that all DPL styles are actually protected against setup violation attacks. Indeed, they never disclose the faulty result in the presence of a setup violation. Instead, they have two different kinds of behavior:

1. WDDL and MDPL compute results given the inputs, and propagate NULL spacers for the outputs whose values are non decidable. This is the logic behavior of `'U'` in VHDL. One could say that faults in these logic are recessive w.r.t. VALID values.

2. iMDPL, DRSL, STTL propagate the NULL state on the fault , even if a VALID value could have been deduced. This is the logic behavior of `'X'` in VHDL. Faults in this second class of logic are dominant, or rather contaminating, as their propagation is indeed an unexpected avalanche effect.

The implication is that DPL in itself does not provide a good protection against symmetrical faults. As a matter of fact, it can filter out a NULL (see Fig. 4.22(a)) and generate a faulted VALID from NULL tokens (see Fig. 4.22(b)). In contrast, the DPL styles that are EE-free propagate the NULL unconditionally. Additionally, the NULL (behaving like an `'X'`) always absorbs other VALID faults, as shown in Figure 4.23.

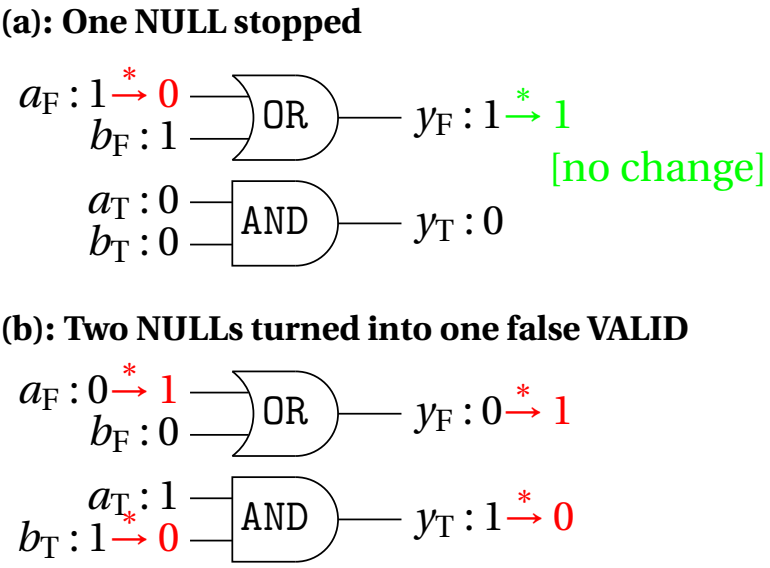


Figure 4.22: Two DPL w/ EE drawbacks to fight DFAs, illustrated on the example of a DPL AND gate. In this figure and in the subsequent ones, the asterisk character (*) symbolizes the faults.

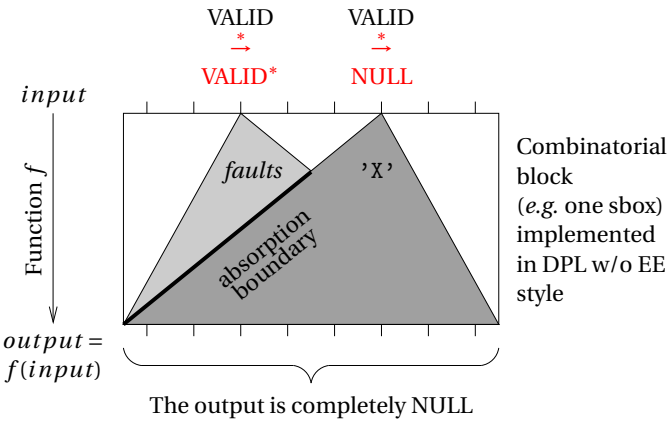


Figure 4.23: Illustration of the absorption of VALID faults by a salvo of NULL tokens in two inter-penetrating logic cones in a DPL w/o EE netlist.

Table 4.11: Number of NULL tokens propagated on average through the s-boxes of AES ($8 \rightarrow 8$) and DES ($6 \rightarrow 4$) in DPL with EE.

Fault	AES Sbox	DES Sboxes							
multiplicity	(SubBytes)	#1	#2	#3	#4	#5	#6	#7	#8
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	4.04	2.48	2.53	2.65	2.46	2.53	2.60	2.63	2.50
2	7.04	3.88	3.90	3.92	3.93	3.91	3.93	3.93	3.91
3	7.94	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
4	8.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
5	8.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
6	8.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
7	8.00	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
8	8.00	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.

4.6.3 Propagation of NULL Values Through Substitution Boxes

The fault propagation in logic with EE is exploding in substitution boxes (s-boxes). The average number of NULL tokens at the output of various s-boxes when one or several NULL tokens of the same type (either NULL0 or NULL1) are at the input has been computed in Table 4.11 for AES and DES, for any logic style subject to EPE, such as WDDL or MDPL.

In DPL styles which resist EE, the propagation is also independent on the implementation. It is also more straightforward as it does not depend on the data: the propagation through a gate occurs iff the output depends on the given input. This is case of all non-trivial gates. Notably, any fault, even single, on the input of an sbox, corrupts the entire sbox output: the propagation is maximal.

4.6.4 Analysis of the FA Protection of DPL

Single bit faults are inefficient against DPL because they turn a VALID data into a NULL token, that propagates and leads to an non-exploitable error since it hides the faulted value. Multiple faults generate randomly a large quantity of NULL values along with some unlikely but exploitable bit-flips. However, as NULL values are systematically propagated, they proliferate very quickly after some combinatorial logic layers traversal. And as they have the nice property to contaminate VALID values, the risky coherent bit-flips (simultaneous $0 \xrightarrow{*} 1$ and $1 \xrightarrow{*} 0$ in one dual-rail couple), they jam their propagation hopefully before they reach

the algorithm output. This absorption property is all the more efficient as the number of NULL generated by the multiple faults is high. Therefore, the only way to inject a poisonous fault is to stress the circuit sufficiently enough to have multiple faults, without nonetheless creating too many faults so as to leave a chance for them not to be absorbed during their percolation towards the outputs. But, hopefully, in this opportunity window of low stress (generation of 2, 3, or maximum 4 errors because of the high diffusion of cryptographic algorithms), efficient coding schemes can be used in supplement to the DPL w/o EE protection.

To be more accurate, we present a simple model that provides a convincing proof of our assertion. Let us consider a dual-rail circuit that is attacked with a perturbation of $2n$ wires, and that has an intensity sufficient enough to cause $m \leq 2n$ simultaneous faults. We also make the optimistic hypothesis that the m faults are equally distributed over the $2n$ wires, and that the flips are truly symmetrical, *i.e.* it is as likely to flip to a 0 as to a 1. Those conditions model a worst case from the defense view point, because they foster coherent bit-flips susceptible to turn a VALID value into a VALID* one, by the mean of two antinomic flips on two wires pertaining to the same dual-rail couple. To further simplify the modelization, we also assume that the attacked block has a perfect diffusion: in practice, this is not exactly true for one round of an algorithm, but for at least two of them (and exactly two in the case of AES). Nevertheless, it helps us grasp more intuitively the idea of the proof without introducing overcomplicated considerations. Therefore, for a fault to successfully propagate through the round, no single NULL shall be generated. Otherwise, the NULL wave catches the fault, because of the perfect diffusion, as already depicted in Fig. 4.23. The main requirement for VALID faults to be generated, m must be even. Indeed, they are generated by pairs. If, on the contrary, m is odd, then at least one NULL (bit-flip of one wire in a pair) is generated, leading to the VALID fault absorption. Then, a VALID fault is generated iff, given a unique fault, a second one occurs in the paired wire. For $m = 2$ faults, this happens with probability $1/(2n-1)$. Then, the probability to generate at least one VALID fault that survives until the output is equal to:

$$p(2n,m) \doteq \begin{cases} \binom{n}{m/2} \Big/ \binom{2n}{m} & \text{if } m \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}$$

The computation of the probability to have only VALID bit-flips is a matter of events counting. The space we are working on is made up of $2n$ objects, each of which belongs to a pair. It is illustrated in Fig. 4.24. We choose m different objects, that are going to be the

targets of exactly m induced faults. There are $\binom{2n}{m}$ ways to do so; furthermore, we assume that each configuration is equally likely. Now, we are interested in the subset of events that consists in having no single object selected. In other words, if one object is selected, then its other half in the pair must be selected too. This comes down to counting the number of different selections of complete pairs (the two elements together). Given that there are n different pairs and that we are going to choose $m/2$ amongst them. There are $\binom{n}{m/2}$ ways to select complete pairs only. Therefore, the probability to select complete pairs is defined by the ratio between:

- the number of complete pairs selection, and
- the number of possible selections.

It is equal to $p(2n,m) = \binom{n}{m/2} \Big/ \binom{2n}{m}$.

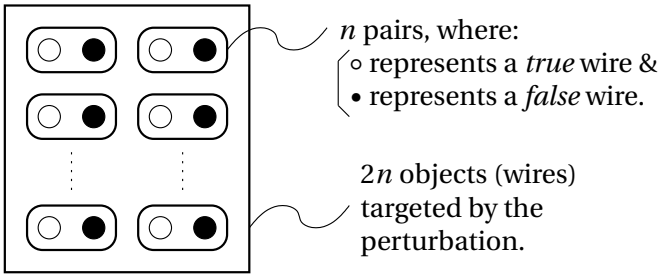


Figure 4.24: Set of $2n$ wires that are coupled and will be subject to m faults.

Traditionally, the difficult cases to handle in terms of protection are when the faults multiplicity m is large. The worst attack consists in flipping one-half of the bits, *i.e.* $m = n$. Notice that having $m > n$ in a sustained manner would imply to flip more than half of bits, which means that the fault injection is correlated with the objects' value. This is usually not the case, since faults injection are never coherent in practice. So, we study $p(2n,m)$ only for $m < n$. Anyway, because of the following symmetry property: $p(2n,2n-m) = p(2n,m)$, the study for the $m > n$ deduces directly from that of $m < n$. We emphasize that the two interesting features of $p(2n,m)$ regarding DFA-resistance are that:

1. it is zero if m is odd, which makes any attack infeasible,
2. it is decreasing fast otherwise, when m increases in $[2,4,6,\cdots,2\lfloor n/2 \rfloor]$.

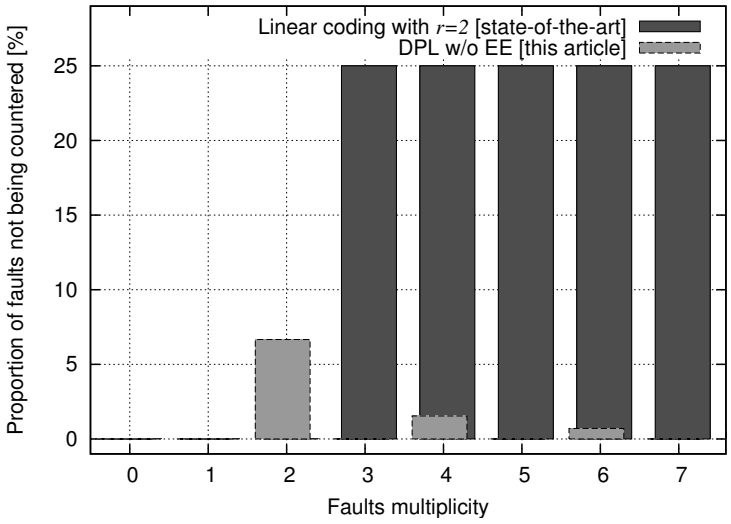


Figure 4.25: Probability that m faults injected on n wires be innocuous due to the protection conveyed by two different countermeasures: either a *detection* by an informational redundancy scheme or an *annihilation of the faulted data* by one or several VALID $\xrightarrow{*}$ NULL token transformations.

As a proof for the second characteristic of $p(2n,m)$ when $m \ll n$, we underline this asymptotical equivalence by :

$$p(2n,m) \approx_{n\to+\infty} \frac{1}{2^{m/2}} \cdot \frac{m!}{(m/2)!} \cdot \frac{1}{n^{m/2}} = \mathcal{O}\left(\frac{1}{n^{m/2}}\right).$$

This probability becomes very small starting from a multiplicity of 4 when m increases up to n[1]. This is to be contrasted with schemes involving a coding with error detection. They are basically able to detect:

- all the faults of multiplicity smaller than the error detection capability r (faults of multiplicity $m \leq r$ mutate a code word into a non-code word), but
- only a ratio of $1 - 1/2^r$ faults for $m > r$.

The Figure 4.25 compares the rate of successful faults injection depending on the multiplicity, for an $n = 8$ set of wires, respectively for the proposed scheme based on DPL w/o EE and for a classical integrity check with a linear code detecting $r = 2$ bits of error.

[1] When m is too large, starting from n, the probability increases, because of the property: $p(2n,m) = p(2n,2n-m)$.

As a matter of fact, usual schemes, based on spatio-temporal redundancy or coding, can be defeated with high probability if the number of faults is greater than the detection capacity. The implementations using DPL w/o EE take advantage of three properties that all contribute to destroy the VALID faults:

1. faults are likely to alter only one wire in a DPL pair, especially if the stress is badly localized, thus creating much more NULL tokens than wrong VALID pairs,

2. because of the protection against EE, NULL values win against VALID ones, hereby hiding in particular VALID fault propagation,

3. as the algorithms implement cryptography, they have a high diffusion, which helps the NULL values meet the possibly faulted VALID values still alive.

4.6.5 Low-cost countermeasure against setup time violation attacks

A straightforward countermeasure against non-invasive global attacks consist in inserting some logic in charge of detecting abnormal situations before the critical parts of the designs become faulty. For instance, the Figure 4.26 presents a setup consisting of a series of buffers followed by an inverter, making up a delay line, inserted between two registers. The source register value passes through the series of buffers, inverted and then stored in the destination register at each clock cycle.

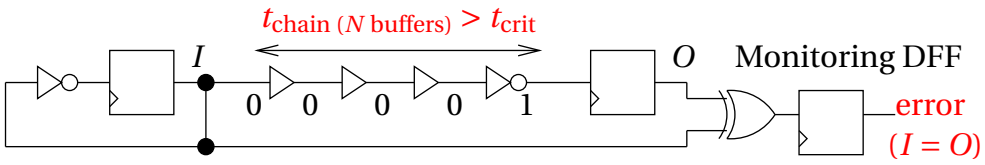


Figure 4.26: Countermeasure based on the insertion of a monitoring logic with a propagation time higher than the critical path of the rest of the circuit.

The source register also receive the complement of its current value every clock cycle. For a circuit working in nominal conditions, the outputs of the two registers should be complementary. If this condition is violated, the circuit is faulty and should be immediately stopped. The number of buffers are chosen such that the delay of this chain becomes a higher than the critical path of the targeted circuit. If the operating voltage is reduced, the delay chain will be violated before the actual circuit and an alarm is raised before the cryptographic parts of the design become faulty.

The chain should be implemented in such a way that it operates at the same clock as the protected circuit and driven by the same source voltage. We implemented this countermeasure on an Altera Stratix FPGA. We used “lcell”, an Altera primitive cell which ensures that synthesis tool will not remove or shorten the length of the chain while optimization.

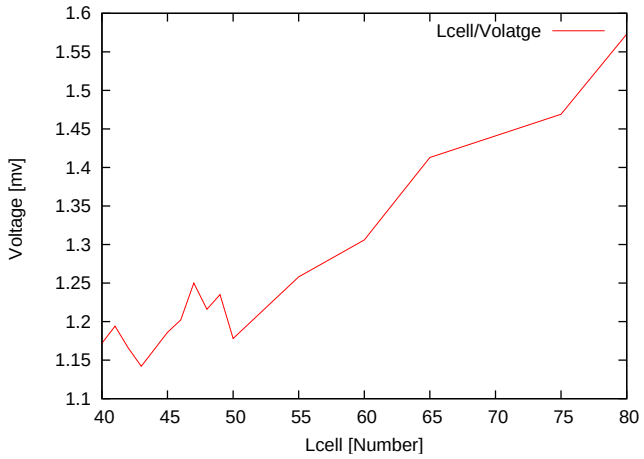


Figure 4.27: Chain Voltage/lcell.

We analyzed the chain in order to find a relation between the length of the chain and the faulty voltage. Figure 4.27 shows the voltage of the setup violation as a function of the number of lcell used in the chain. The nominal voltage for the used FPGA is 1.5V. It is clear that the violation voltage increases more or less linearly with the number of lcell (buffers).

4.7 Conclusions

In this chapter, we presented different DPL logic styles aiming at hiding the cryptoprocessor activity to thwart SCA. It permits the understanding of different DPL styles and compare them. DPL also resist against DFAs in addition to increasing the resistance against SCAs. Indeed, single faults consist in turning a VALID token into a NULL one, which conceals the value of the (sensitive) data before corruption. The DPL styles that protect against the EPE side-channel analysis ensure in addition that the NULL propagation contaminates all the data it crosses in the combinatorial logic cones. Thus, in the case of multiple faults, both VALID faults and NULL tokens are generated, but the NULL tokens destroy the VALID faults prior they arrive at the algorithm observable outputs. Even the "differential behavioral attack" (DBA), where a simultaneous observation of the faulty message and corresponding power curve are used by an attacker. In the case of DPL the attacker cannot learn anything

from power curve corresponding to a faulty encryption. Finally, we also propose a low-cost countermeasure to detect setup time violation faults.

Although the DPL logic is based on an elegant manner to obtain secure implementations, flaws exist at logical and physical level. The different logic styles are more or less able to counteract these negative effects but often with higher complexity or effort at back-end design. Research on new DPL styles is still active to improve the robustness and keep a good compromise with complexity and performances requirements. In the next chapter we propose two new DPL countermeasures for FPGA with aim to thrawt SCA at a reasonable cost.

Chapter 5

Novel DPL countermeasures for FPGA

In Chapter 4, we showed that DPL has a excellent property of resisting asymmetrical faults by construction. Therefore if the flaws in DPL i.e. EPE and technological imbalance are removed, we can arrive at a DPL which is a common countermeasure against SCA and FA. One device can be claimed tamper-resistant only if it is protected, at least to some extent, against both SCA and DFA simultaneously. Generally countermeasures against SCA and DFA are implemented separately. DFA countermeasures act at the algorithmic level, usually introducing space or time redundancy in data representation and processing. However, the effective protection against SCA is more subtle. There is a need for removal of any source of leakage which could provide some sensitive information through physical side-channels. Therefore, a widespread methodology is to use balanced logic gates along with *ad hoc* back-end steps. As we know how to resist against DFA before the logic synthesis and to resist against SCA after synthesis, it is implicitly considered obvious that the protection against DFA and SCA should be built independently. To avoid such complex countermeasures DPL is a good solution given it can provide required SCA resistance at reasonable cost. In this context, we present various methods to improve DPL implementations on FPGA.

Two different DPL countermeasures for FPGA are proposed in this chapter. Each of the two countermeasures deploy different techniques to counter flaws like EPE or routing imbalance. Detailed analysis of these new DPL countermeasure against SCA is also presented.

5.1 DPL without Early Propagation Effect (DPL w/o EPE)

5.1.1 Rationale of the Proposed Logic

A DPL cell, in general, observes one transition per cycle which is favourable for a DPA resistant logic style. As shown in Figure 5.1, a basic DPL AND gate consists of an AND gate

(G) and a complementary OR gate (G^*, satisfying $G^*(x) \doteq \overline{G(\overline{x})}$). This basic DPL gate does not has a synchronization scheme and is prone to early propagation effect. We denote such DPL styles by DPL with early propagation effect (DPL w/ EPE). Its truth table is shown in Table 5.1. The activity of a true, false pair of a signal x can be shown as:

- $x_T = x_F$ in precharge phase and
- $x_T = \overline{x_F}$ in evaluation phase (where the horizontal bar represents the complement).

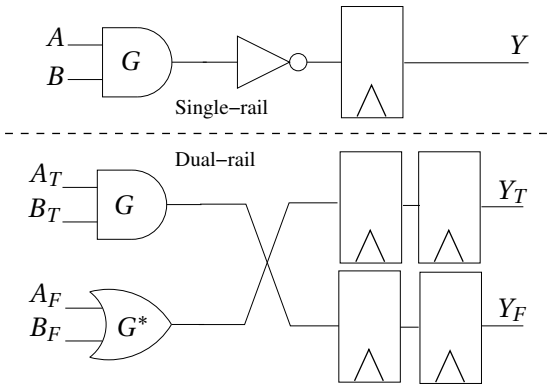


Figure 5.1: DPL building block.

Now let us closely observe Table 5.1. The Table 5.1 shows that for some inputs (a_T, a_F, b_T, b_F) values like (0,0,0,1), (0,0,1,1), (0,1,0,0) the AND_F gate evaluate to '1'. For these values AND_F has evaluated despite the fact that either of a or b is not VALID. This is indeed logical as an OR gate outputs '1' when at least one of the inputs is '1'. Thus the DPL AND gate evaluates early before a and b have acquired a VALID state. Such phenomena can result in a data dependent leakage. The problem of imbalanced routing stays independent of the early evaluation effect which comes either from imbalance by automatic place and route or due to construction of FPGA fabric.

As stated earlier, if DPL is made EPE free then it can be used as a common countermeasure against both SCA and DFA. To cure DPL of EPE, we implement the truth tables of DPL logic such that it propagates a VALID output only if all the inputs are VALID. This behavior can be achieved by a purely combinatorial gate, as depicted in Table 5.2 for an AND gate where the encoding mask is changed from (`CC00, FAFA`) to (`FC80, FAE0`). Similarly, we can calculate the encoding mask for a DPL OR (`T, F`) gate which are changed from (`FFCC, A0A0`) to (`FEC0, F8A0`). This implies that an AND_T gate which was previously implemented

Table 5.1: Look-up-Table (LUT) masks encoding for 4-input LUT implementing the AND function in DPL with early propagation effect (DPL w/ EPE)

DPL w/ EPE				AND_T		AND_F	
a_T	a_F	b_T	b_F	CC00		FAFA	
0	0	0	0	0	0	0	A
0	0	0	1	0		1	
0	0	1	0	0		0	
0	0	1	1	0		1	
0	1	0	0	0	0	1	F
0	1	0	1	0		1	
0	1	1	0	0		1	
0	1	1	1	0		1	
1	0	0	0	0	C	0	A
1	0	0	1	0		1	
1	0	1	0	1		0	
1	0	1	1	1		1	
1	1	0	0	0	C	1	F
1	1	0	1	0		1	
1	1	1	0	1		1	
1	1	1	1	1		1	

by an encoding mask CC00 (see Table 5.1) will now be implemented with an encoding mask FC80(see Table 5.2). By changing the encoding mask we ensure the following:

- The gate outputs NULL when the inputs are
  NULL or transitional from this value.
- The gate outputs VALID only when all the inputs are VALID.
- In case of inconsistent values w.r.t. DPL convention, the gate outputs an arbitrary NULL value.
- The overall 4-input gate is positive.

This logic, which we call as DPL without early propagation effect (DPL w/o EPE) does not evaluate early by design, and propagates errors: if any input is stuck to NULL or if the input

Table 5.2: Look-up-Table (LUT) masks encoding for 4-input LUT implementing the AND function in DPL without early propagation effect (DPL w/o EPE)

DPL w/o EPE				AND_T		AND_F		**Input state in**
a_T	a_F	b_T	b_F	FC80		FAE0		**the DPL protocol**
0	0	0	0	0		0		All NULL0
0	0	0	1	0	0	0	0	Transitional from NULL0
0	0	1	0	0		0		Transitional from NULL0
0	0	1	1	0		0		Faulty
0	1	0	0	0		0		Transitional from NULL0
0	1	0	1	0	8	1	E	All VALID: $(a,b) = (0,0)$
0	1	1	0	0		1		All VALID: $(a,b) = (0,1)$
0	1	1	1	1		1		Transitional from NULL1
1	0	0	0	0		0		Transitional from NULL0
1	0	0	1	0	C	1	A	All VALID: $(a,b) = (1,0)$
1	0	1	0	1		0		All VALID: $(a,b) = (1,1)$
1	0	1	1	1		1		Transitional from NULL1
1	1	0	0	1		1		Faulty
1	1	0	1	1	F	1	F	Transitional from NULL1
1	1	1	0	1		1		Transitional from NULL1
1	1	1	1	1		1		All NULL1

is out of specifications, then the output always remain to NULL too. An advantage of DPL w/o EPE over certain other DPL is that all kind of logic can be used instead of only positive logic since output is calculated if all inputs are VALID and **does not generate glitches**.

For a complex circuit, we need a set of logic functions in order to achieve an optimised synthesis. Table 5.3, provides the encoding mask for all two input functions synthesized with 4-input LUT. The table list encoding masks when a particular function is implemented in DPL w/ EPE and its equivalent when the same function is implemented in DPL w/o EPE. The rows printed in gray correspond to trivial functions (those that depend on only one input variable), that can be implemented as routing in DPL. The constant functions 0 and 1 are also ruled out as they do not propagate the precharge, and because they are simplified out by the synthesizer anyway: they should not exist in the same netlist. These logic as presented are specific to FPGAs directly targeting the 4 input LUT rather than basic gates.

The LUT mask for a bigger LUT can be calculated in a similar way when targeting a modern FPGA.

Table 5.3: LUT masks for the $2 \rightarrow 1$ gates in DPL w/ and w/o EE styles.

$f(a,b)$, **when** $(a,b)=$				**DPL-EE**		**DPL-noEE**		**Function**
(1,1)	(1,0)	(0,1)	(0,0)	**True**	**False**	**True**	**False**	**Name**
0	0	0	0	0000	FFFF	F880	FEE0	0
0	0	0	1	A0A0	FFCC	F8A0	FEC0	$\overline{a} \cdot \overline{b}$
0	0	1	0	C0C0	FFAA	F8C0	FEA0	$\overline{a} \cdot b$
0	0	1	1	F0F0	FF00	F8E0	FE80	$\overline{a}$
0	1	0	0	AA00	FCFC	FA80	FCE0	$a \cdot \overline{b}$
0	1	0	1	AAAA	CCCC	FAA0	FCC0	$\overline{b}$
0	1	1	0	EAC0	FCA8	FAC0	FCA0	$a \oplus b$
0	1	1	1	FAFA	CC00	FAE0	FC80	$\overline{a} + \overline{b}$
1	0	0	0	CC00	FAFA	FC80	FAE0	$a \cdot b$
1	0	0	1	ECA0	FAC8	FCA0	FAC0	$\overline{a \oplus b}$
1	0	1	0	CCCC	AAAA	FCC0	FAA0	b
1	0	1	1	FCFC	AA00	FCE0	FA80	$\overline{a} + b$
1	1	0	0	FF00	F0F0	FE80	F8E0	a
1	1	0	1	FFAA	C0C0	FEA0	F8C0	$a + \overline{b}$
1	1	1	0	FFCC	A0A0	FEC0	F8A0	$a + b$
1	1	1	1	FFFF	0000	FEE0	F880	1

5.1.2 Implementation on FPGA

An AES coprocessor is implemented in two DPL styles for evaluation. We realize a proof-of-concept experiment where targeted logic are implemented in a similar manner. The two implementations differ only in the LUT mask encoding, having exactly the same back-end as optimised by the tool. In another implementation, we try to balance the placement and routing of the two designs by providing placement constraints, the rest being unchanged. Hence we implement DPL w/ EPE and DPL w/o EPE with and without balanced placement and routing (total four implementations). In FPGA, our chosen technology target, the configuration files remain the same, but for the LUT masks. Thus, by subsuming the individual issues of robustness against SCA and DFA into a unique problem, we arrive at an original

solution that is economic in resources because of its duality w.r.t. both the SCA and the DFA threats. The main objective of this study is to demonstrate the effect of early evaluation, which to our knowledge, has only been studied theoretically before [81].

Thus to assure security of the design it is sufficient to secure the datapath only. The design flow to implement a DPL w/o EPE cryptographic coprocessor on an FPGA is the same as shown for WDDL in Section 4.2.2. Since DPL designs are redundant by nature, we have to use customised tool for processing. The goal of this synthesis is to remove the unnecessary logic redundancy while keeping the redundancy needed for DPL style. This cannot be achieved by a standard design flow. An ASIC synthesizer is used to synthesize the design with a library containing all the gates considered in Table 5.3. Then the output netlist is processed using a custom tool which converts a single-rail netlist into a DPL netlist. The controller is then connected to the datapath using a wrapper. Thereafter, a legacy FPGA vendor tool does synthesis, mapping, placing & routing for the whole design on the FPGA. Although the design flow is shown for Altera FPGAs, it has also been tested for Xilinx FPGAs.

As stated earlier, to secure a design against SCA and DFA we can use a DPL style which is free from EPE by limiting the library to two-input gates, implemented as in Tab. 5.3. We start with an RTL model of single-rail AES coprocessor synthesized using QUARTUS II for EP1S25B672C7 device from Altera Stratix FPGAs as a reference design. For DPL logic, this coprocessor is synthesized using a library of all non-trivial two input logic functions. Once the synthesis of DPL w/o EPE logic is complete, the routing constraint file (rcf) is extracted from the design. Then we implement an implementation using DPL w/ EPE logic which is based on WDDL but its very different from WDDL because DPL w/ EPE is not limited to positive gates. We change the LUT masks in the previously obtained dual-rail netlist as in Table 5.3. Again this netlist is connected to other non-sensitive part of the design and synthesized using QUARTUS II. The routing is constrained by the previously extracted **rcf**. We are able to conserve 99.96% of the routing from one DPL logic to another. Notice that no effort has been put in placing and routing the design. The placing and routing is done automatically for the first design by the FPGA vendor's tool. The second design is forced to keep the same routing. Considering the cost of the design, it is evident that a design in original WDDL will be smaller than a design in DPL w/o EPE. However, in our implementations we force the DPL w/ EPE design which is also based on the principles of WDDL to be placed and routed exactly as the DPL w/o EPE design. We agree that this is not an optimal implementation for DPL w/ EPE but can be interesting for observing the effect of early evaluation.

Finally we have two identical netlist, implementing the same circuit and having the same placement and routing, differing only in LUT masks. Thus one netlist has LUT masks

Table 5.4: Area & Performance comparison of DPL w/o EPE AES datapath with unprotected reference.

Parameters	Unprotected	DPL w/o EPE	Overhead
Logic Cell (LC)	1958	14574	X7
LC Registers	256	1024	X4
Max. Frequency (MHz)	32.86	19.70	60%

Table 5.5: CPA results on S-box 0 for two DPL variants of AES.

Implementation	No. of Traces							
S-box BIT index	0	1	2	3	4	5	6	7
DPL w/ EPE	5124	X[1]	1496	X	2376	X	25432	X
DPL w/o EPE	X	X	2332	X	X	X	X	X

which suffer from early evaluation while the other netlist is immune. Such a setup will allow us to demonstrate the effect of the early propagation effect.

5.1.3 Evaluation of Early Propagation Effect

5.1.3.1 Using Correlation Power Analysis

We analysed the two DPL variants against differential power analysis. We took power consumption measurements ($traces$) at 5GSa/s, using an *electromagnetic probe* capturing the field of a leaking capacitor on the back-side of the FPGA core with a *54855 Infiniium oscilloscope* from Agilent Technologies. In order to reduce acquisition noise, each trace was average of 64 acquisitions. We performed a mono-bit correlation power analysis (CPA) on the first s-box of the two DPL implementations. The power consumption difference can be exploited on one bit individually. When the activity for multiple bits are added as in correlation power analysis on unprotected circuits, the residual biases between activity of (0,1) and (1,0) is not consistent from bit to bit; it is not straightforward which method to combine them would be suitable.

The performed attack was partially successful. The results are shown in Table 5.5. We were able to find the right key of the first s-box for three bits in DPL w/ EPE implementation

[1] In this table, X signifies failure of the mounted attack after 40 000 traces.

and only one bit in DPL w/o EPE. Given this data, it can be said that DPL w/o EPE implementation provides higher robustness than DPL w/ EPE. Thus keeping everything constant, countering early evaluation has improved the robustness of the design. Next we try to measure the side channel leakage due to early evaluation, using mutual information metric.

5.1.3.2 Using Mutual Information

The main advantage of mutual information metric (MIM) lies in the value of $I(S;O)$ which can be used to compare implementations. Here $I(S;O)$ is the mutual information between two random variables S and O, where S represents set of sensitive variables used by the target device and O are the observations measured during computations of such device. For two similar implementations, the one leaking more information from the side channel during a particular time instant will generate a higher $I(S;O)$, computed knowing the correct key $k = k^0$ than the one leaking less. We use this technique to compare the two DPL variants of AES. Incidentally, DPL was also taken as an example where mutual information performs well in the MIA original paper [29].

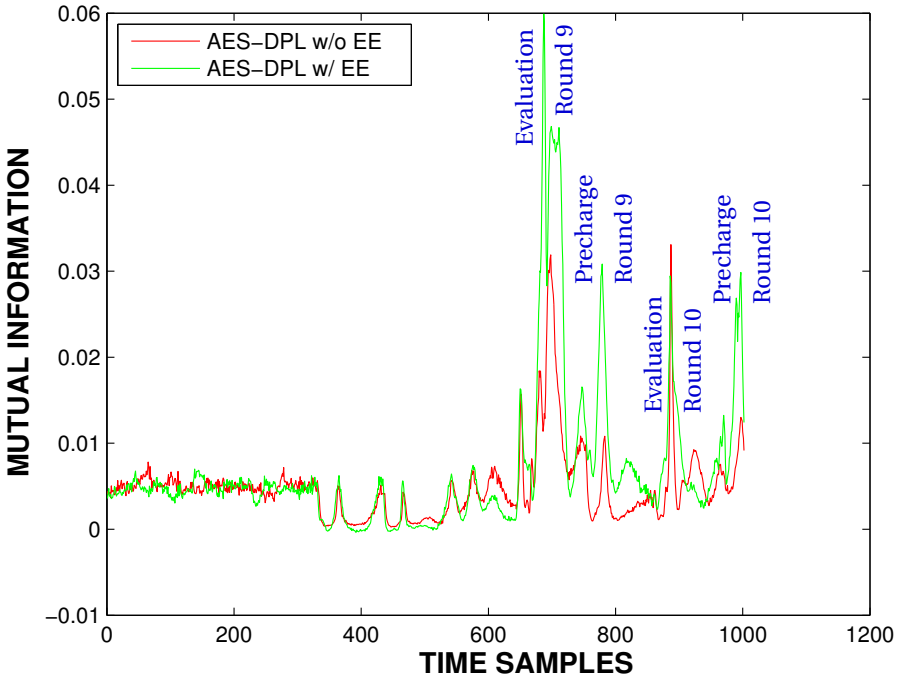


Figure 5.2: Comparison of Mutual Information leaked from an AES protected using DPL w/ EPE and DPL w/o EPE.

We use the same traces as for CPA and compute the mutual information for the correct key at the first s-box (one byte). As shown in Figure 5.2, the mutual information for a DPL w/ EPE is almost twice as that for DPL w/o EPE. Since everything else is kept unchanged, the leakage is reduced due to removal of early propagation effect from DPL w/ EPE. The remaining leakage could be due to technological bias (routing imbalance of the complementary logic) which is common to the two implementations. Indeed a detailed analysis of the floorplan revealed that the leaking bits are routed using long routing channels. On the other hand, non-leaking bits are connected to adjacent LUT thus avoiding routing channels. Therefore it can be said that imbalanced routing introduce delays in the circuit exploitable by attackers. Controlling routing of a design, to our knowledge, is difficult because the routing algorithms are kept secret by the FPGA vendors. Nevertheless, some options to control routing are available which are shown in Section 5.1.4.

Every combinatorial block in DPL receives input and stores output by a pair of registers in master-slave mode. If we consider the circuit at various stages, the connection between master and slave is pretty much balanced as very few wires are used. Similarly, for the connection between a combinational output and master register input, the routing is balanced. For the connection between slave register and input of the combinational part, the fanout is high. It is at this point (denoted Y_t and Y_f in Fig. 5.1) that the circuit suffers from high imbalance in the routing of the two circuits. In our analysis, the two circuits are routed almost identically. Assuming equal leakage due to routing from both circuits, countering the early evaluation has alone reduced the leakage by half. Thus early evaluation is a major flaw in dual-rail logic in FPGA and reducing this effect could considerably reduce the leakage.

In Figure 5.2, we see that the leakage is occurring over a period of 4 clock cycles. We correlate the traces with the value of the secret in penultimate round. In DPL, each round is composed of two clock cycles: precharge and evaluation. Since, the last round does not have MixColumns and the SubBytes is bijective, therefore a byte in round 9 is correlated to a byte in round 10 located at the corresponding spot. This means that if we observe correlation with a byte in round 9, we should also observe the correlation in round 10. Hence we see 4 peaks for precharge and evaluation cycles in round 9 and round 10, in Figure 5.2, for each of the DPL w/ EPE and DPL w/o EPE.

We wish to illustrate that the leakage put forward by the MIM indeed opens the door to successful key recovery attacks, and that the attack is easier if the MIM indicator is high; this means that the MIM value, for any characterization (*e.g.* target bit in a sensitive AES state byte) already indicates the speed of the attack: roughly speaking, the speed of the attack is related to the MIM value. In corollary, the different values of the MIM give a precious

feedback information to the designer: the bits that yield the largest MIM should indeed be corrected with priority in the design.

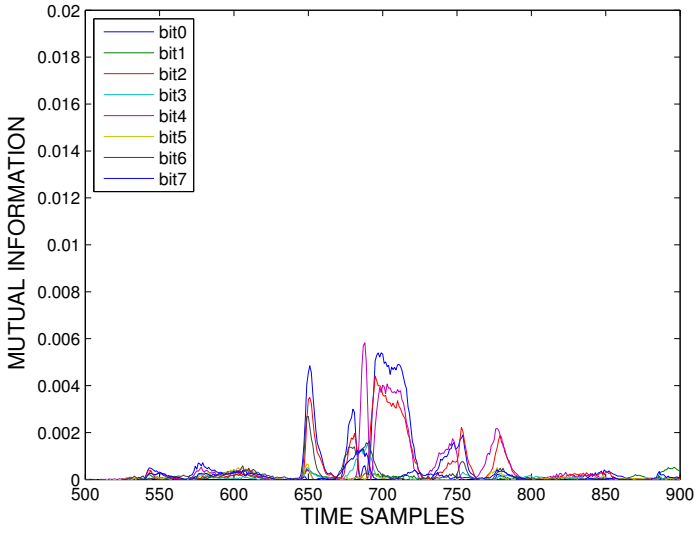


Figure 5.3: Bitwise leakage of $S-box0$ in DPL w/ EPE.

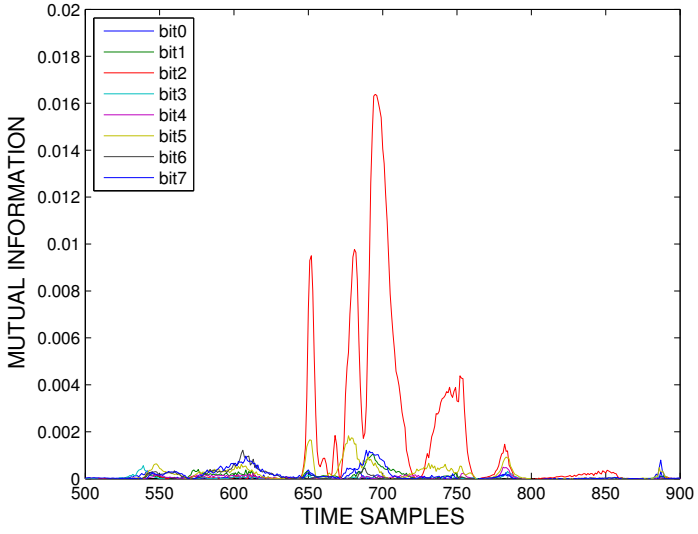


Figure 5.4: Bitwise leakage of $S-box0$ in DPL w/o EPE.

We also aim to test the real vulnerability of the flaws identified in the two designs. To

reach this goal, we compare two attacks: CPA and MIA. The CPA is known to be appropriate if the leakage model is correct, whereas the MIA is both matching the MIM analysis (both are based on PDF estimation) and more resilient to leakage model imperfections.

Comparing Table 5.5 with figures 5.3, 5.4, we see that the bits broken using CPA are the ones leaking in the MIM. Although this observation seems intuitive and justifies the relevance of robustness metrics, such as the MIM, we emphasize that, to our best knowledge, it is the first time a relationship between the amount of leakage and the speed of an attack is put forward experimentally. Roughly, lesser the MIM value of a leaking bit, higher is the number of traces needed for a successful CPA. On the other hand, bits not leaking in MIM are also not broken by CPA. Also in Figure 5.4, bit 2 is leaking much more than the other 7 bits which are leaking almost nothing. A detailed analysis of the post synthesis netlist showed that the synthesizer didn't respected the DPL w/o EPE encoding mask for this bit during the optimisation process. However the other bits, which we checked were compliant with the DPL w/o EPE logic. Thus the unexpected leakage observed is due to some unwanted optimization by the tool and imbalanced routing. From a designer point of view, such information is very important as the designer can cover the loop holes once known.

5.1.4 Balanced Placement

In commercial FPGAs, one issue often faced by designer is that they do not have enough control over placement and routing procedures. A designer could provide certain constraints to the FPGA vendor's tool but sometimes it is difficult to achieve such constraints to desired level. In cryptographic circuits which are implemented on FPGAs, imbalanced placement and routing is also a source of leakage. We would like to test a technique which could be used to achieve balanced placement and routing. We explain this solution with reference to Altera Stratix FPGAs but could be easily applied to other FPGAs. In Stratix FPGAs, each slice contains 10 LUT. While synthesizing the design we provide as constraint a file which places a gate and its complementary gate into the same slice (refer Figure 5.5). Given the two complementary gates placed in proximity, it could be expected that the routing would be more or less balanced. We would like to specify that it is difficult to precisely control the routing for Altera FPGA by user-defined constraints.

Hence we synthesize a DPL w/o EPE netlist as before but we include the placement constraints this time. Let us call this version as "DPL w/o EPE-BB" where BB stands for balanced back-end of the design. Then we extract the routing constraint file (rcf) file of this design which is used to synthesize DPL w/ EPE netlist with balanced back-end referred as "DPL w/

```
set_instance_assignment -name LL_MEMBER_OF Region_13 -to "MUX_2_297:mux_1|g2115_true" -section_id Region_13
set_instance_assignment -name LL_MEMBER_OF Region_13 -to "MUX_2_297:mux_1|g2115_false" -section_id Region_13
set_global_assignment -name LL_RESERVED OFF -section_id Region_13
set_global_assignment -name LL_MEMBER_STATE LOCKED -section_id Region_13
set_global_assignment -name LL_SOFT OFF -section_id Region_13
set_global_assignment -name LL_AUTO_SIZE OFF -section_id Region_13
set_global_assignment -name LL_STATE FLOATING -section_id Region_13
set_global_assignment -name LL_HEIGHT 1 -section_id Region_13
set_global_assignment -name LL_WIDTH 1 -section_id Region_13
set_global_assignment -name LL_ORIGIN LAB_X1_Y1 -section_id Region_13
```

Figure 5.5: Placement constraints crafted for the placing complementary gates in same slice when synthesized by Altera `Quartus`.

EPE-BB". Since only the placement has changed the area utilization and performance of the design with balanced back-end are the same as the design without balanced back-end.

5.1.5 Evaluation of Balanced Placement

Here we analyze the implementations using correlation power analysis (CPA). The traces are normalized before applying CPA. It can be observed that normalization improves the attack. There are 4 different implementations of AES analyzed and compared which are protected using: DPL w/ EPE (Figure 5.6), DPL w/ EPE-BB (Figure 5.7), DPL w/o EPE (Figure 5.8), DPL w/o EPE-BB (Figure 5.9). The correct key guess is (`0xa3` = 163) and we plot the rank of the correct key. In these graphs, the color red indicates that the attack has recovered the correct key.

Now if we compare the implementations, improvements achieved using back-end balancing is negligible. In fact balancing the placement does not necessarily improve the routing. Present results show minor improvement due to back-end balancing but overall the results are not impressive. These results cannot be considered generic. There may be cases where forcing placement might degrade the routing balance. Also, here results are specific to Altera Stratix. Better balancing might be possible in other FPGA or with other tools.

5.2 Balanced-Cell Based Dual-Rail Logic (BCDL)

In the previous section, we presented DPL w/o EPE as a countermeasure which deploys a synchronization scheme built into the logic. Evaluations demonstrated that the leakage due to EPE was reduced by deploying the synchronization scheme. However, the countermeasure was costly in terms of area and slow in performance. Imbalanced routing of DPL w/o EPE is partially due to high cost or complexity. This is because for high number of dual-rail signals present in the netlist there will be more difficulty in balancing them symmetrically. The idea of balanced-cell based dual-rail logic (BCDL) was proposed by Guilley and Danger

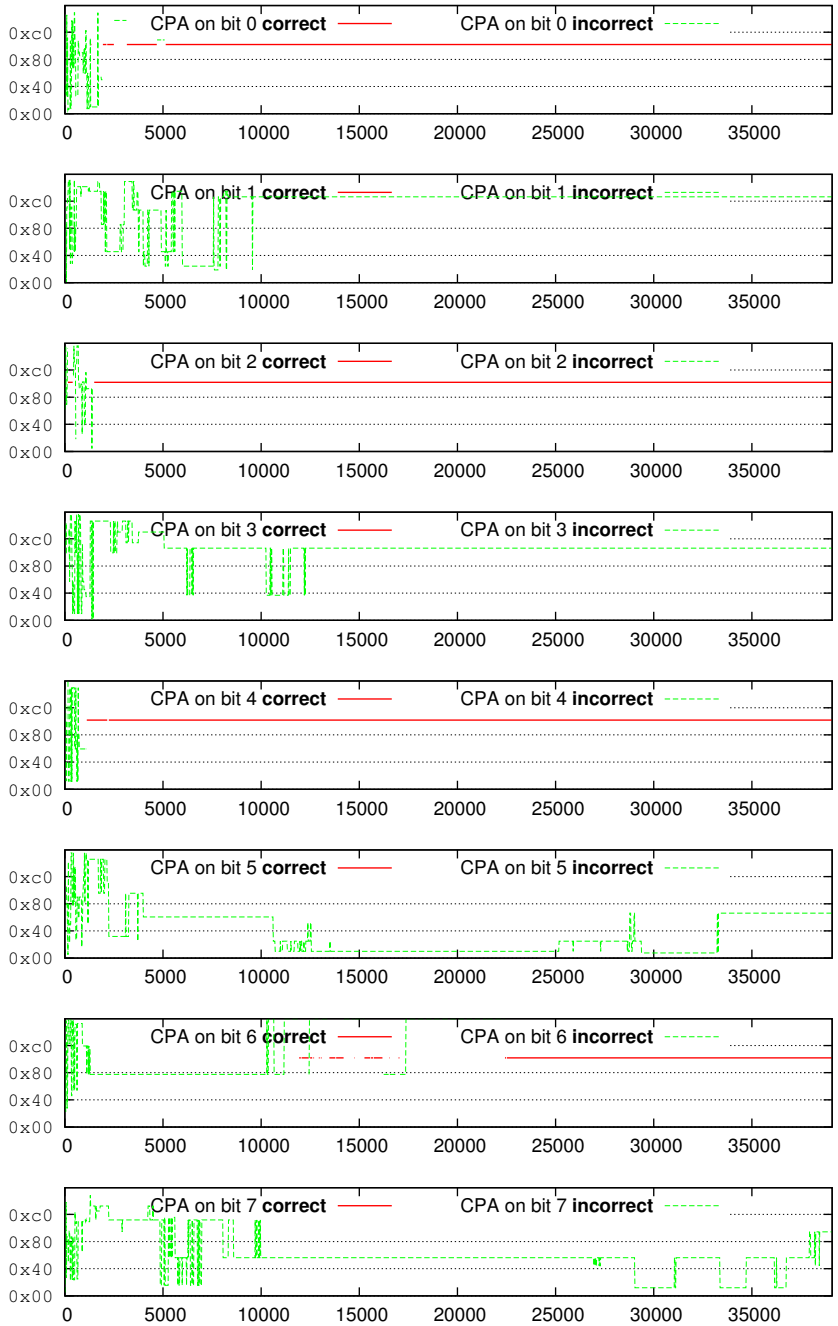


Figure 5.6: Key value retrieved by CPA for DPL w/ EPE.

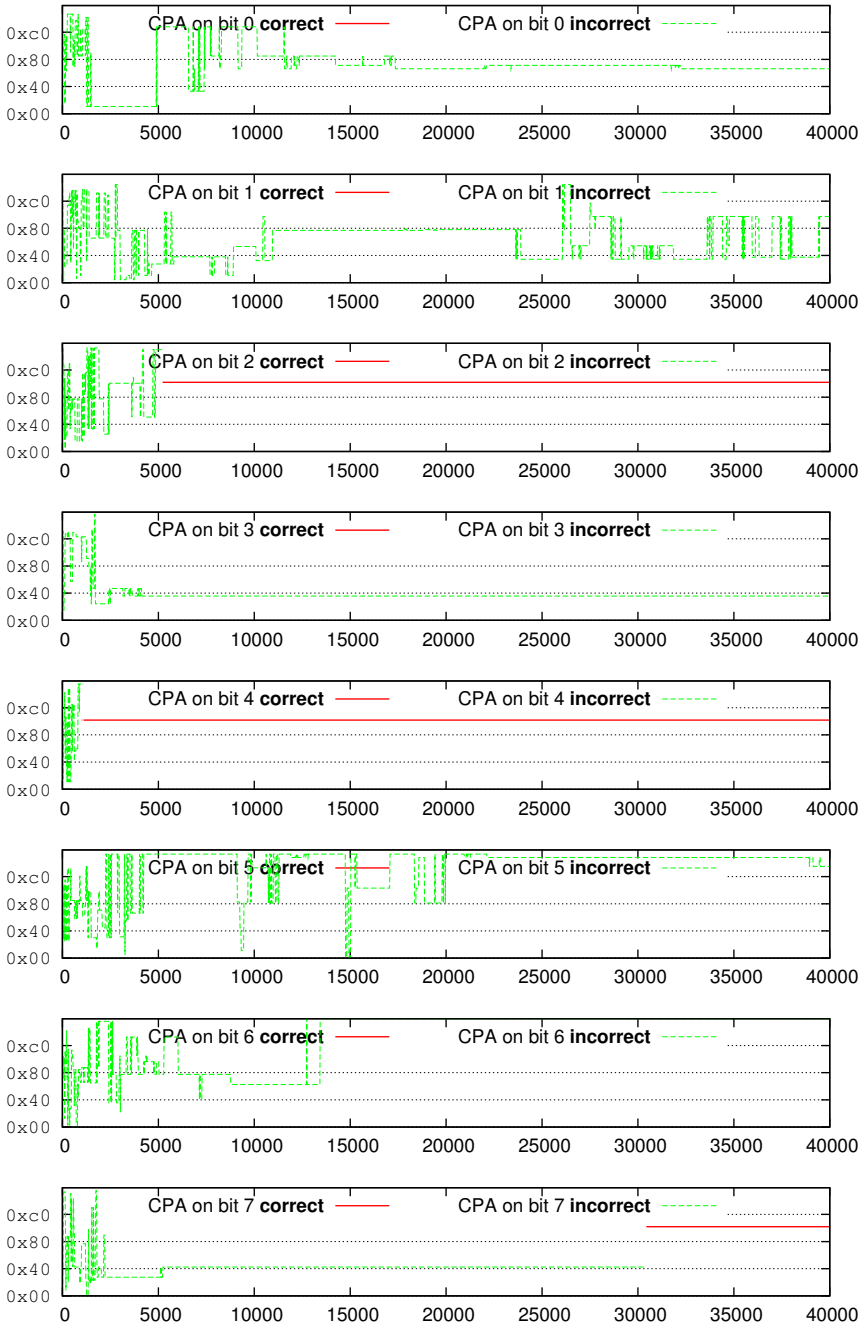


Figure 5.7: Key value retrieved by CPA for DPL w/ EPE-BB.

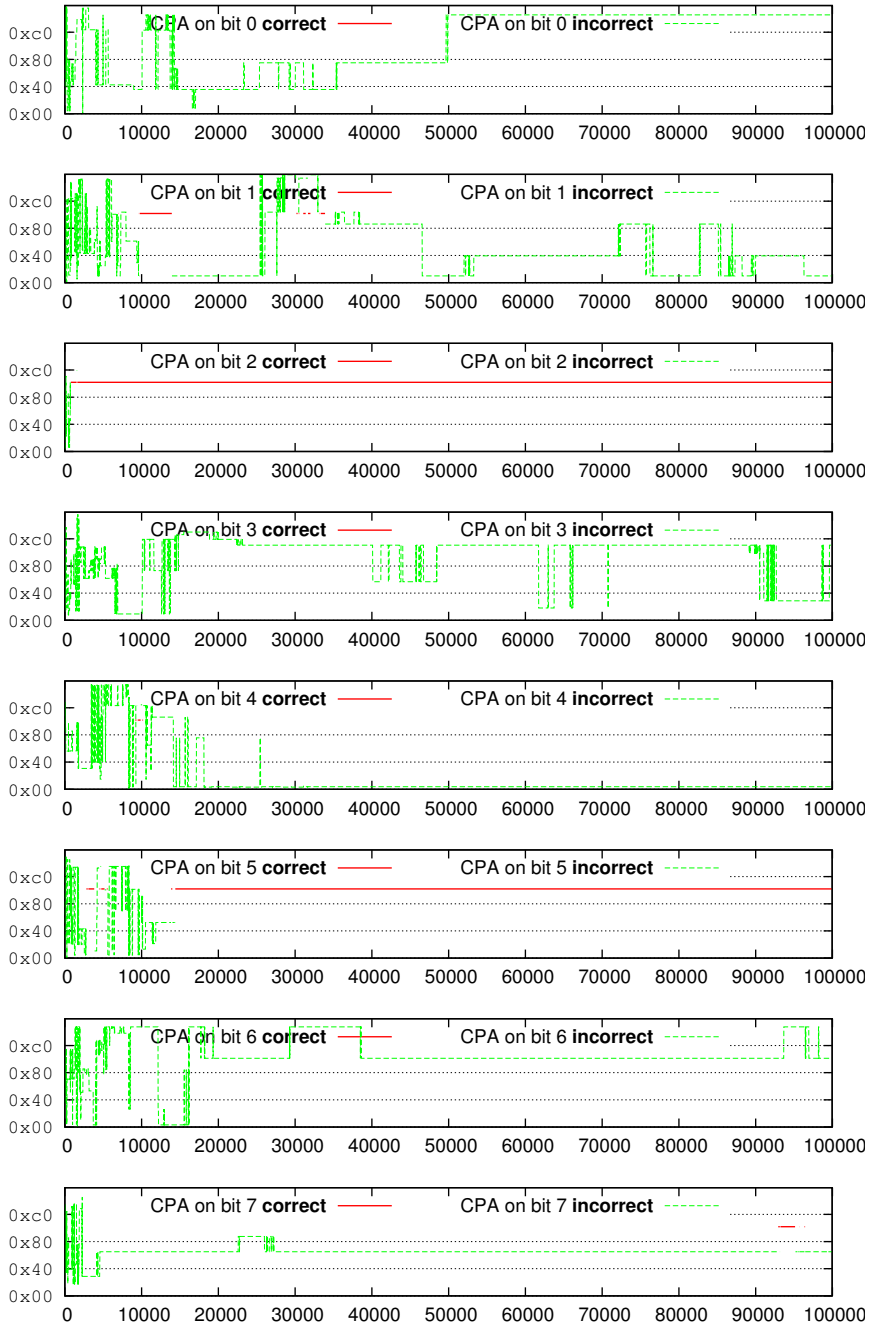


Figure 5.8: Key value retrieved by CPA for DPL w/o EPE.

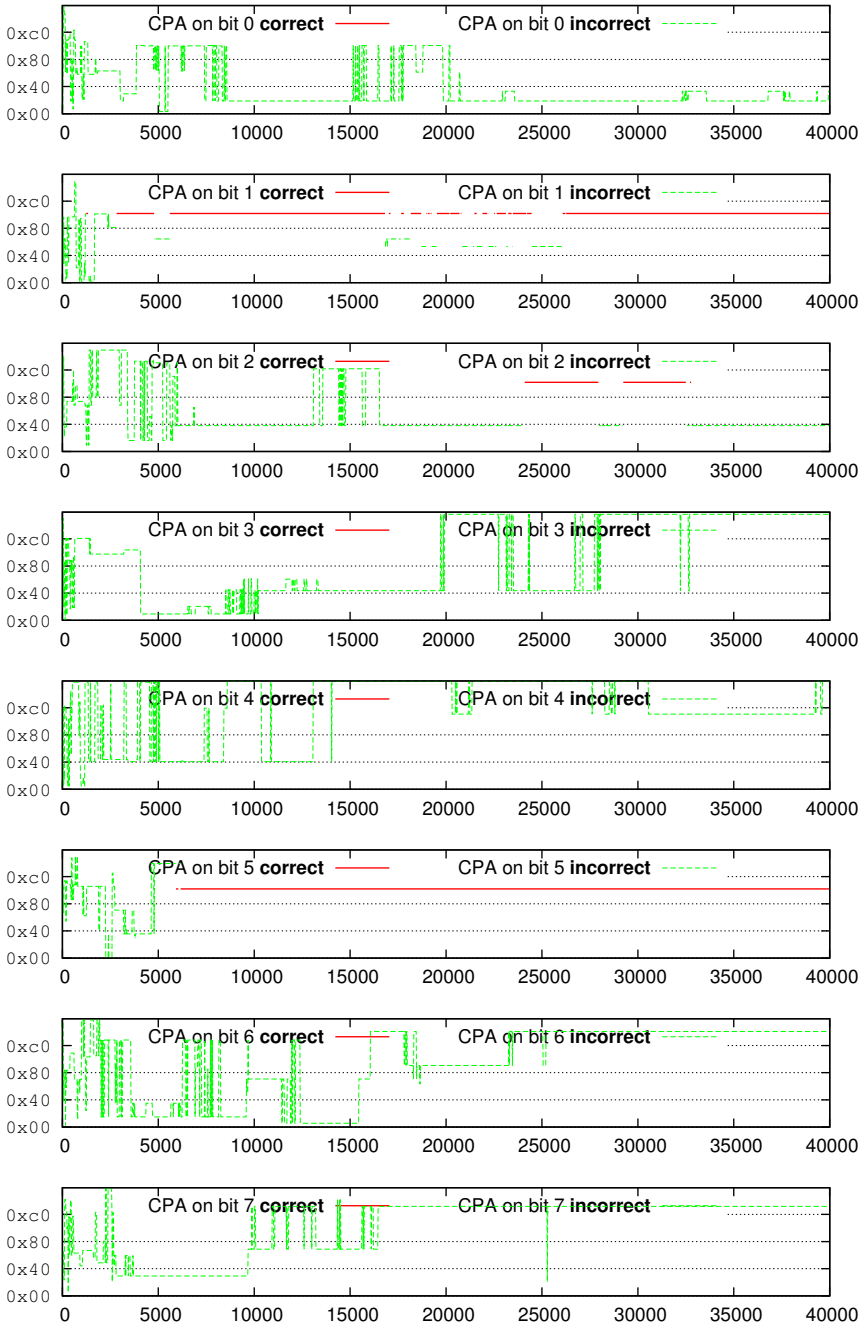


Figure 5.9: Key value retrieved by CPA for DPL w/o EPE-BB.

in [112] and I will introduce various methods to implement BCDL on an FPGA. The purpose to introduce BCDL is to find a DPL logic style which can provide a evident robustness at a reasonable cost and performance trade-off on FPGAs. The main goal of BCDL [112] is to avoid most of the vulnerabilities in current DPL by using synchronization schemes, while keeping a reasonable complexity. Therefore it is based on two principles:

1. A specific **synchronization** scheme is added to all logic gates, before the actual precharge or evaluation.

2. The synchronization is performed on **a block of data** (which is well adapted to FPGA LUT).

5.2.1 Synchronization to counter EPE

5.2.1.1 Basic Principle

The idea of synchronization in BCDL is taken from asynchronous circuits. In clock-less circuits, synchronization is usually performed between 2 signals with "C-element". A C-element is a memory that only changes its state when there is unanimity (to 0 or 1) on its inputs. In BCDL, special signals called unanimity to 0 and 1 (U_0 and U_1) are deployed for synchronization on bundled data. Unanimity signals have the following properties:

- U_1 is the signal authorizing the evaluation. It rises to 1 when all signals have left the precharge state defined by equation (5.1):

$$U_1(x,y,\ldots) \doteq \begin{cases} 1 & \text{if } x \neq (0,0) \text{ and } y \neq (0,0), \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

- $U_0 = 1$ when all the inputs are in the precharge state, as shown in equation (5.2):

$$U_0(x,y,\ldots) \doteq \begin{cases} 1 & \text{if } x = y = \cdots = (0,0), \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Actual calculation only starts if there is unanimity (U_1 or U_0 valid) and is frozen otherwise. In other words, BCDL circuit evaluates only when either U_1 or U_0 are valid otherwise forced to precharge. Figure 5.10 shows a schematic diagram for synchronization of bundle data.

Calculation of the precharge requires that output of each cell is forced to 0. The actual computation of signals carrying information is done in the evaluation phase. Based on this

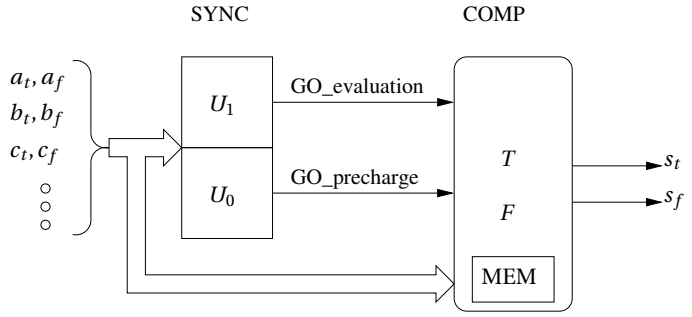


Figure 5.10: Synchronization and "bundled" data in BCDL.

property, we can optimize our model by using a simplified scheme with a **global precharge signal**, PRE. It is used to induce the precharge state globally, in a very short amount of time. Thus, it allows the designer to reduce the complexity (and increase the performances) of the BCDL cell, by replacing the "unanimity to 0" (of Figure 5.10) by a logical "AND" between the PRE signal and the output of the "unanimity to 1" (see Figure 5.11).

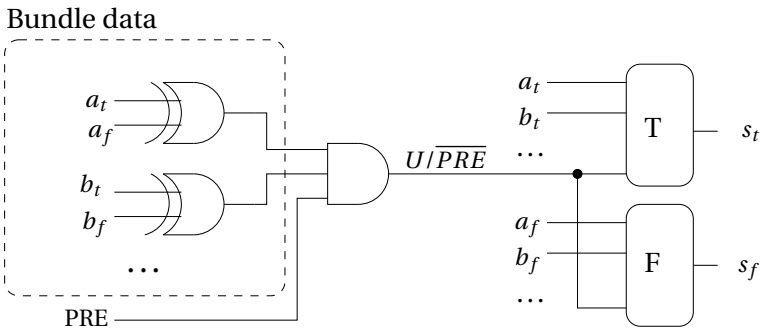


Figure 5.11: n-input BCDL cell.

The actual computation is then synchronized by the $U/\overline{PRE}$ signal as follows:

- When $U/\overline{PRE}$ falls to 0 (just after the signal PRE), the precharge is forced, independently from the inputs. Using a clock path to route global signal PRE in FPGA ensures us that the precharge signal will always be faster than any input. As a matter of fact PRE takes advantage of the FPGA global lines which are specific, fast and sized to broadcast heavy loaded signals. Moreover frequency of PRE is half that of the clock signal which can be generated from a FPGA PLL without any skew with respect to the clock.

- When $U/\overline{PRE}$ rises to 1, indicating that, on one hand, the signal PRE is valid and, on the other hand, that the synchronization of inputs is over, the evaluation phase

begins.

Precise temporal relationships between signals of a 2-input OR gate (where (a_t, a_f), (b_t, b_f) are the inputs and (s_t, s_f) the output) are shown in Figure 5.12.

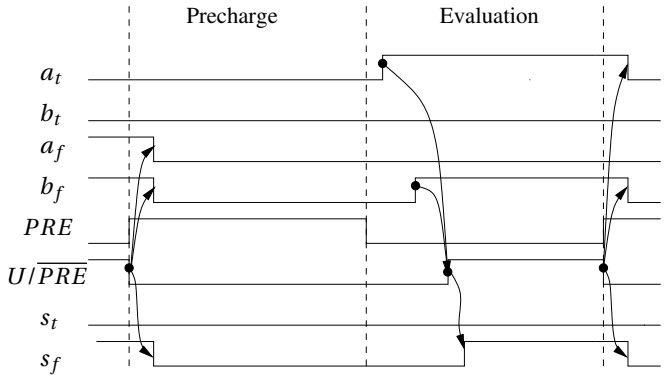


Figure 5.12: Temporal relationships of a 2-input BCDL OR gate signals.

5.2.1.2 Application to FPGA

Based on the above properties, our logic is able to overcome EPE on a global scale (between each cell of the circuit). However, an FPGA is build of look-up tables (LUT) and the synchronization should be applied at LUT-level. FPGA LUT can be imagined as a tree of multiplexers as shown in Figure 5.13 because the actual architecture is never known. Local synchronization is achieved by applying the two following constraints:

- The $U/\overline{PRE}$ signal is assigned to the first column of this tree.
- The inputs e_t and e_f are plugged on the same pin respectively on T and F cells.

Due to these constraints, we can obtain significant properties regarding local robustness which are:

- **No glitches:** As the $U/\overline{PRE}$ signal is the first to switch before the precharge state, the internal nets are all forced to '0' without any glitch, regardless of the inputs. Likewise, it is the last one to switch prior to the evaluation after the other inputs are already positioned.
- **Reduction of the technological bias:** The total number of switching events for T and F gates does not change according to the inputs. It is a constant = $(2^n - 1)$ for a n-input LUT. It is therefore difficult to discriminate the activity of T from that of F as

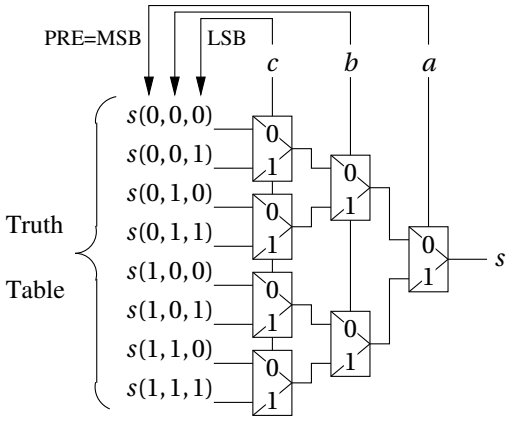


Figure 5.13: Structure of T and F LUT.

the consumption profile is identical regarding the couple T , F . This is illustrated in Figure 5.14, which describes all the combinations of a 2-input XOR, when $U/\overline{PRE}$ switches. Bold nets correspond to multiplexers outputs that are switching. There is, thereby, an overall balance in terms of switching time as well as energy consumption (number of simultaneous switching).

- **No local early evaluation or precharge:** Indeed, $U/\overline{PRE}$ is always delaying the evaluation (switching to '1' last) and forcing the precharge (falling to '0' before any other signal). In other words, the evaluation is always **delayed** and the precharge always **anticipated**, regardless of the data.

5.2.2 Area Optimization

Area-consumption is a limitation for most DPL countermeasures on FPGA. Even if the main goal is to achieve the best robustness, it could prove useless if it is actually too complex to implement on a real device. Thereby, one of the main objectives of BCDL is to keep a reasonable complexity. Thanks to our synchronization schemes we obtain three significant properties:

- **Reduced S-Box area:** As stated before, in various DPL logic style, one basic 8-input s-box (2^8 byte) could be merely duplicated into two T and F 16-input s-boxes (512×2^8 byte). Another approach is to use a true and a false RAM of size $2^8 \times 8$ each with special circuitry at the output to enable dual-rail operation as shown in Figure 5.15. The net cost of RAM is increased by a little over two. However, this low-cost implementation

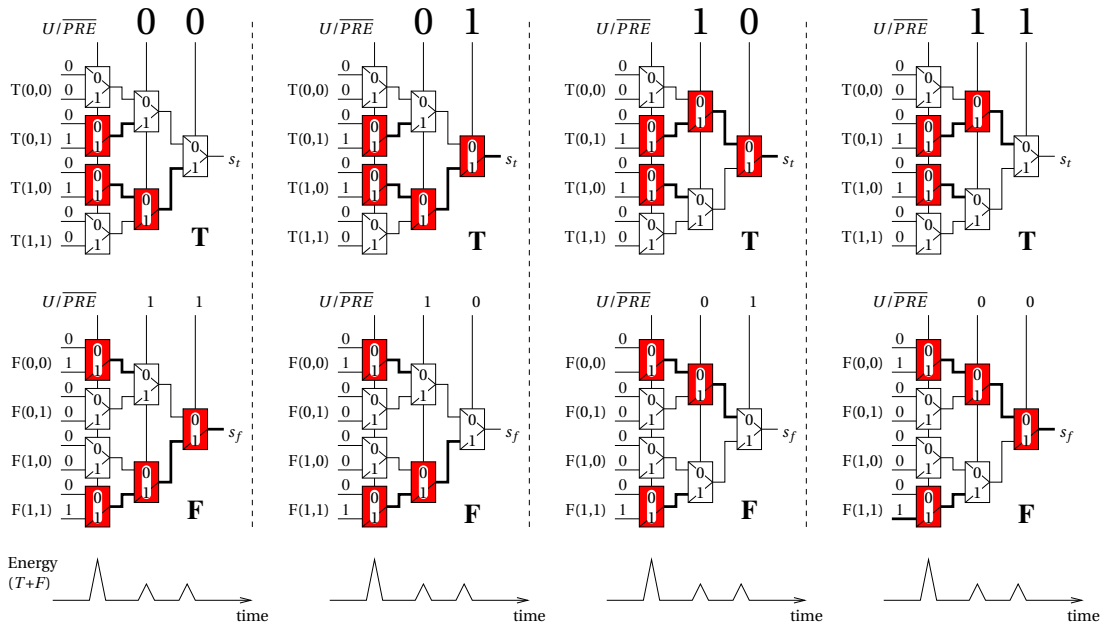


Figure 5.14: Local switching balance in BCDL: LUT3 example.

(in Figure 5.15) is vulnerable to glitches and the input of AND gate can leak information if not implemented properly. In Figure 5.15, the input of the AND gate can also be attacked. Also special routing is required for the precharge signal to the RAM output which is generally limited at the global DPL inputs. This huge size can be reduced by building a local precharge signal but it might induce glitches due to the lack of synchronization. BCDL takes advantage of its global precharge signal to reduce the RAM size to only **4 times** the basic one without any glitch risk. Indeed there will be T and F S-Boxes, which will only have one more input than the basic implementation,

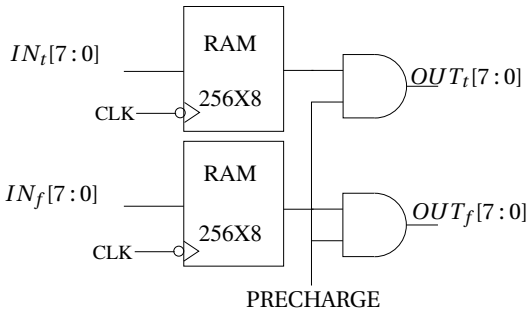


Figure 5.15: A low cost DPL S-box in RAM.

that is the $U/\overline{PRE}$ signal. If the memories are synchronous, PRE can be used directly instead of $U/\overline{PRE}$ signal.

- **Reduced complexity:** Due to the absence of glitches within LUT, BCDL is not limited to positive functions, and can use all 2^{2^n} existing functions (for a n-input LUT), which provides many optimization opportunities. Though an extra signal PRE has to be routed to each part of the design.

- **Integrated synchronization:** We can exploit the recent FPGA technologies to make this optimization. For a 2-input function, if we use an FPGA with 5-input LUT ($LUT5$) or more, we can directly integrate the synchronization scheme in T and F cells. Indeed, if the $true$ and $false$ signals as well as $U/\overline{PRE}$ are inputs of the same LUT, the rendezvous can be computed without additional logic cells. It is then possible to implement a 2-input BCDL function with only 2 $LUT5$.Some FPGA offer bigger LUT with multiple output which can accommodate the whole BCDL cell in a single LUT.

5.2.3 Performance Optimization

As of now, all DPL-based countermeasures have about the same performances, and that is a speed at least two times slower than the unprotected architecture. This is mainly the result of the typical 2-phase behavior ($precharge$, $evaluation$) which is common to all DPL. Most of the times, the precharge has roughly the same duration as the evaluation, in WDDL for example, it must last long enough for the '0' to go through all the logic. On the other hand, due to the global precharge signal, BCDL can be optimized to be faster than any other DPL. As a matter of fact, the global signal being extremely fast and homogeneously distributed throughout the device, the duration of the precharge state can be quite short. More time is then given to the evaluation, which dictates the speed of the design. We propose to achieve speed optimization by using a non-regular clock, as shown in Figure 5.16. Using this scheme, we can speed up the system $\sim 1.3 - 1.5$ times the basic one.

5.2.4 Implementation of AES BCDL on FPGA

BCDL can be implemented using the top-down approach like DPL w/o EPE where the datapath is synthesized with a restricted library and then duplicated using scripts. Another approach to perform this task is the **bottom-up approach**. The main advantage of this approach is that we can implement the design by identifying and securing basic primitives. Another benefit of this approach is that we only duplicate or in other words secure only

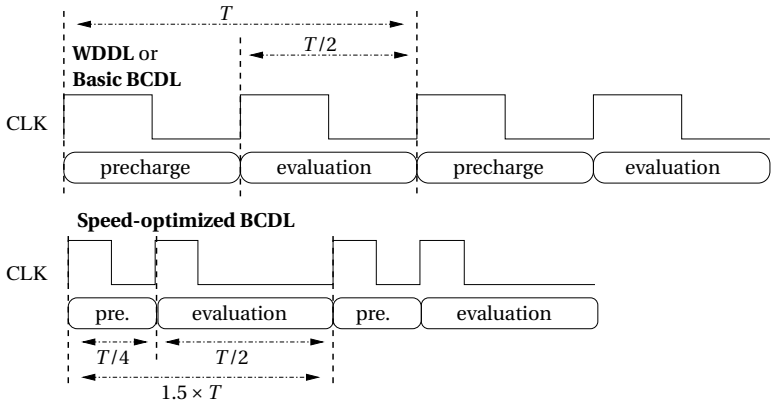


Figure 5.16: Basic BCDL *versus* speed-optimized BCDL timings.

data signals only which carry the actual secret in cryptographic circuits. The control signals are not carrying any important information and can be used without duplication. Thus the circuit occupies less resources.

Implementation of AES cryptographic co-processor in BCDL using bottom-up approach starts with identifying the primitives used. AES is an iterative algorithm where a round is calculated several times depending on the size of the key. AES can be implemented using four primitives which are **XOR gates, multiplexers and s-boxes (memories) and registers (DFF)**.

To convert a AES single rail module to BCDL using bottom-up approach following steps should be followed:

1. The AES single rail code should be written in a structural manner using the primitives. In case of AES, these are DFF, memories (s-box), XOR and multiplexers.

2. All the data signals should be identified and duplicated. This means that a signal a is converted to a_T and a_F.

3. The 4 primitives should be replaced by BCDL primitives. The BCDL primitives are dual-rail and comply with two phases operation.

4. Then a wrapper which can connect single rail I/Os to the dual rail data signal is added. The wrapper also ensures that the I/Os are precharged every other clock cycle to follow DPL protocol.

5. Since a DPL circuit works at half the frequency of its single rail equivalent, the FSM should work only during evaluation phase.

Most of these steps are pretty simple to implement. However depending on the circuit under consideration it might be difficult to identify all the primitives. If the number of distinct primitives are high, too much effort might be required to implement this approach.

The four BCDL primitives of AES are explained below:

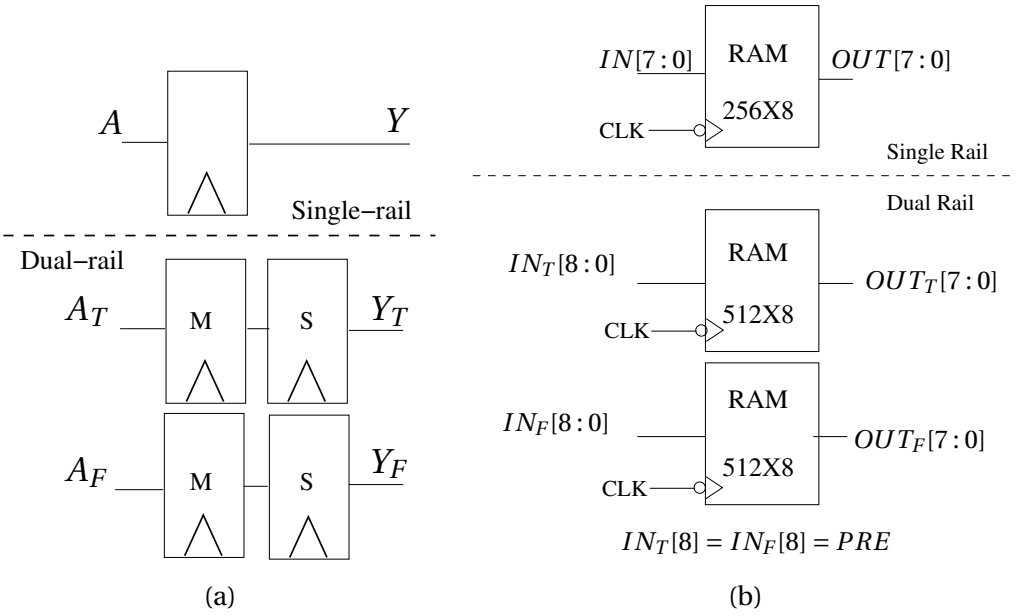


Figure 5.17: (a) BCDL DFF, (b) BCDL S-box.

1. **BCDL Registers (DFF):** A BCDL register is comprised of four DFFs. These are similar to WDDL or DPL w/o EPE where two registers in master-slave configuration are used in each true and false part of the datapath as shown in Figure 5.17(a). Such arrangement of DFFs allow two phase operation of BCDL.

2. **BCDL S-box:** An AES s-box takes a byte as input and returns a byte output which can be implemented in $2^8 \times 8$ RAM. To make s-box BCDL compliant, we add 1-bit input as the MSB called PRE such that when PRE is low, the output is stuck to "00000000" else the output receives the computed value. By choosing PRE as the MSB, we divide the memory into two halves. With this extra input we now need a $2^9 \times 8$ RAM. Taking duplication into account the size of the memory is quadrupled. This is a remarkable

as for other DPL like DPL w/o EPE, the cost is increased by 256 times (1MB). We do not need a synchronization as the RAM we use are synchronous to clock. This will limit the maximum frequency of the circuit as majority is done during half of the clock but the problems like EPE are solved. Figure 5.17(b) shows how a single rail s-box is converted to a BCDL s-box.

3. **BCDL XOR:** XOR gates are non-positive in nature. Such gates are normally avoided in DPL logic because they block the precharge signal from flowing, therefore sources of glitches. The XOR has to be implemented in such a way that it allows the precharge phase to propagate and does not generate glitch. To achieve this we try to modify the truth table of the XOR gate in a special way as shown in Table 5.6. Doing this, we keep the basic functionality of an XOR unchanged while the intermediate states are all made to propagate zeros. This behaviour will remove glitches and EPE at the same time. We always use two input XOR gates in our design because using LUT5, only 2-input BCDL gates are possible. If we need to use three or more input XOR gates the gate is implemented in several LUT and the interconnections of these LUT can cause a problem. Virtex V possess $LUT6\_2$ which can be used as a 6-input 1-output LUT or as a 5-input 2-output LUT. Similarly Stratix II has ALUT which is capable of implementing two 5-input 1-output LUT if two inputs are common. A whole 2-input BCDL cell with true and false output is synthesized in a single $LUT6\_2$ or ALUT. Figure 5.18(a) shows a complex BCDL XOR gate which uses two LUT, one to implement XOR and other for XNOR.

   Note that this complex XOR gate should be implemented at bit level i.e. for every single bit XOR in single rail, we have a complex XOR gate which comprises of a XOR and a XNOR of single bit inputs. Also we need to make sure that this complex XOR is not simplified or modified by the optimizer. For Altera we can use "lcell" primitive for each of the outputs. When Quartus II finds an lcell declared, it makes sure that particular instance is kept and not merged with other components during optimisation and implemented using a two LUT5 or one ALUT. For Xilinx LUT primitives are used to ensure that BCDL XOR gate is not optimised.

4. **BCDL Multiplexers:** Multiplexers in bottom-up approach can be sometimes tricky. Here we can distinguish two types of multiplexers depending on the selection signal. When the selection signal is a control signal, multiplexers are pretty easy to implement. We just need to duplicate the multiplexers and make sure that they take the same selection signal. This means for every multiplexer in single rail we have two

multiplexers in BCDL. In our AES design, majority of the multiplexers are of this nature. Figure 5.18(b) shows a multiplexer of this kind. As we see the selection function remains the same but in BCDL, this selection function will select both the true and false part of the same signal.

In the second type, the selection signal is a data signal. When this signal is duplicated, the true signal controls the true part of the multiplexers and the false signal controls the false part. In this case we need to ensure that when the selection input is in precharge, the output should also be in precharge.

Table 5.6: Truth Table of a 2-input BCDL XOR gate.

PRE	$I1_{\mathbf{T}}$	$I1_{\mathbf{F}}$	$I2_{\mathbf{T}}$	$I2_{\mathbf{F}}$	$O_{\mathbf{T}}$	$O_{\mathbf{F}}$
0	X	X	X	X	0	0
1	0	0	X	X	0	0
1	1	1	X	X	0	0
1	X	X	0	0	0	0
1	X	X	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1

Apart from these constraints, the global synchronization signal PRE should be routed through global high-speed clock paths. Special care should be taken to ensure PRE uses clock paths which ensures many advantages to BCDL like:

- PRE with the synchronization stage counters EPE.
- PRE forces the precharge phase which removes the constraint of using only positive gates hence decreases the cost.
- Since PRE is much faster than other signals, the precharge phase can be made faster which results in higher throughput.
- PRE is used to synchronize memories thus block memories of FPGA may be used for designing DPL.

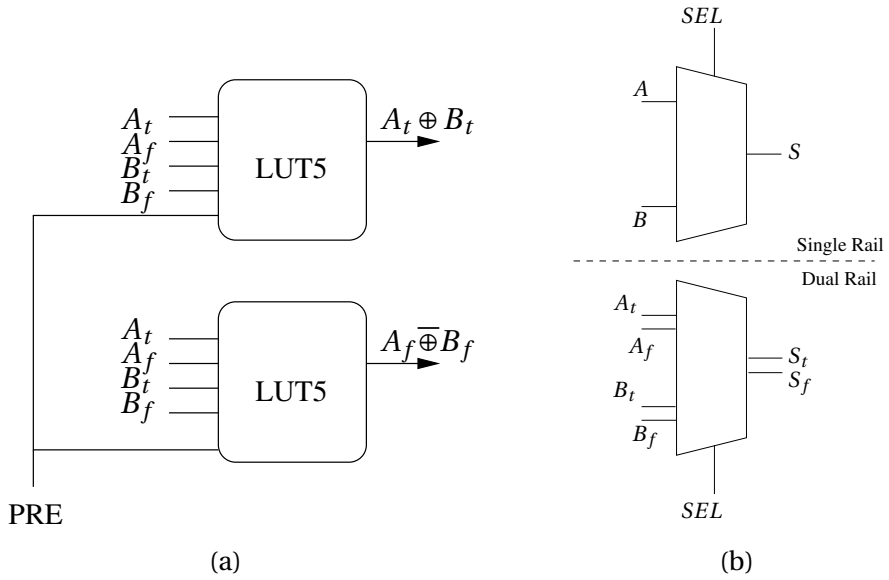


Figure 5.18: (a) BCDL XOR, (b) BCDL Multiplexer.

Table 5.7: Cost Comparison of AES BCDL on Stratix II.

Architecture	Unprotected AES	AES BCDL
ALUT	483	2302
Registers	271	1041
RAMs (4 Kb)	20	40

5.2.4.1 Implementation Cost and Performance Evaluations on Stratix II

We implemented our designs on SASEBO-B Evaluation board [113]. SASEBO-B has two FPGAs soldered on it: two Stratix II ($EP2S15$ & $EP2S30$). The placement and routing is performed automatically by the FPGA tool. Other back-end techniques to balance routing can be added to improve this design.

Single rail AES as reference and AES in BCDL were implemented and tested on Stratix II FPGA ($EP2S30$). AES processes 128 bits of data per cycle. The results are summarised in Table 5.7.

The number of registers are quadrupled from reference AES to AES BCDL as expected. The logic utilization is increased roughly 5 times from single rail to BCDL: the extra cost comes from XOR gate which uses one ALUT per bit of data. Stratix 2 has 4K block RAMs. AES BCDL uses twice the block RAM as compared to unprotected. In BCDL, XOR is a primitive and special care is taken in order to ensure that XOR is not optimised. No such constraints

are applied to the unprotected AES and the FPGA tool is free to optimize it for area. Therefore we see a 5 times logic utilization. Still AES BCDL can have the problem of imbalanced routing.

5.2.5 Improving AES BCDL to reduce routing imbalance on FPGA

DPL w/o EPE was attackable due to imbalanced routing. Further analysis showed that large fanout of registers and high gate count in timing path cause major leakage in the side channel (refer 5.1.3). In DPL w/o EPE, s-box is implemented in logic so the fanout of the register is high. Fanout is also high for diffusion functions. Complex cryptographic algorithms like AES rely on substitution and diffusion function for security. Hardware implementation of substitution and diffusion is done by swapping wires and combinatorial logic where a single gate drives multiple gates. Since multiple operations are done in a round of cryptographic algorithm, the timing path is long. The FPGA placement and routing tool, will place all the gates driven by same input nearby for resource optimization. Therefore it is difficult to provide identical placement and routing to the corresponding gate in the false part which causes routing imbalance. Timing imbalance is also increased with high fanout. It can be roughly expressed as $\Delta T = K \times F$ where K is the constant capacitance and F is the fanout.

A simple way to reduce fanout in a cryptographic circuits is to use memories. Typically, ROM can make up for complex unstructured or structured high algebraic degree combinational blocks. Let us take the example of substitution boxes (s-box) in symmetric ciphers, such as AES. All the s-boxes of AES are the same, and are called SubBytes; they consist in a $8 \rightarrow 8$ bit bijection, defined as $y \mapsto y^{-1}$ in $GF(2)[x]/x^8 + x^4 + x^3 + x + 1$ if $y \neq 0$ or 0 otherwise. In ASIC technology, ROMs are neither efficient in terms of area nor in performance. In FPGA technology, the density of user logic is about 30 times less than that of ROM-based macros. [114]. Therefore, using ROM can drastically reduce the overall design area and power consumption. Such a noting has already be made by Saar Drimer in [68] where authors wanted to design a high-speed AES with low-logic utilization. In our case, ROM are interesting as they can reduce fanout.

Apart from s-box, ShiftRows with MixColumns ensure proper diffusion of the input data. When considering hardware implementation, ShiftRows does not require logic and can be implemented by swapping of wires. MixColumns can be implemented with shift operations and XOR gates. These signals are fed to the state register and then fed back after the next clock. Such structures are area consuming and generate gates with high fanout. For example, a multiplication by 2 in Galois field is done by xtime() function which is used in MixColumns. As shown in Figure 5.19, when xtime() is computed bit B7 has a fanout of 4.

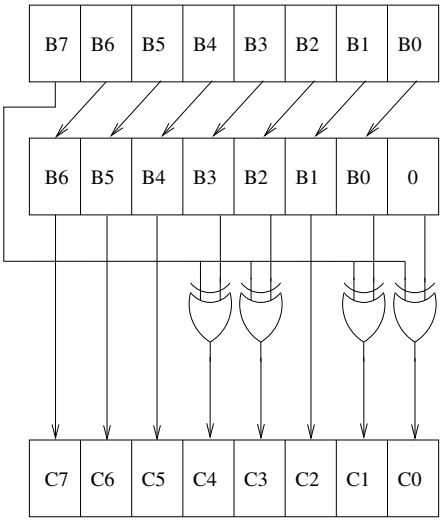


Figure 5.19: Logical diagram of xtime() function.

For a parallel architecture there are 16 bytes processed in parallel. Each byte goes through one xtime() transformation followed by XOR operations This leads to a fanout of 5 for each of these 16 bytes. As a result there are series of gates with high fanout. To counter this problem we use t-boxes. T-boxes were introduced in original Rijndael proposal and its FPGA implementation is discussed in [115]. T-boxes are sets of 8x32 look-up tables which combine SubBytes, ShiftRows and MixColumns. A full AES round can be computed by using t-boxes and XOR gates as shown in Figure 5.20. Since T-Boxes are look-up tables they may be implemented in block memories of FPGA. We also reduce the number of gates in each timing path which increases performance. The XOR network is a set of 4 5-input 32-bit XOR which computes the XOR between 32-bits output of 4 t-boxes with 32 bits of the key to compute one column of the round output each (use Figure 5.21). This architecture has three advantages. The two main advantages are lower logic utilisation and higher operating frequency. The third advantage is the fanout reduction. Also in [116], authors demonstrate that by using memories for DPA resistance is improved. Thus using t-boxes is a good solution. Note that other cryptographic algorithms like DES and PRESENT can be redesigned to possess unitary fanout. In PRESENT if the s-box is implemented in RAM, rest of the circuit has unitary fanout.

As stated above the imbalance in DPL which comes from high fanout and long timing paths can be reduced by using memories. The use of memories is possible due to presence of a global synchronization signal. In other DPL styles, using memories without glitches will have an exponential area overhead. On the other hand, an AES s-box in BCDL needs $2^9 \times 8$

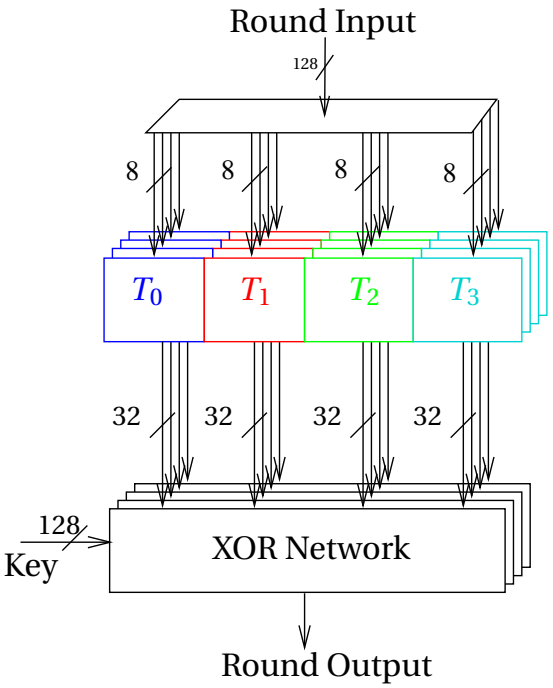


Figure 5.20: Architecture of AES-128 datapath using T-box architecture.

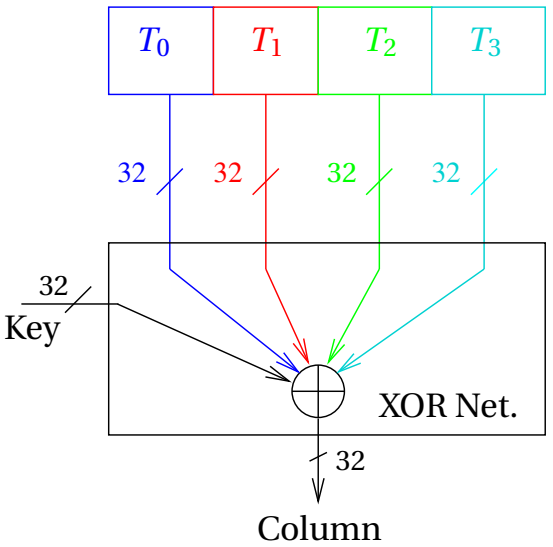


Figure 5.21: XOR network used in the T-box architecture (shown for one column).

bits (4Kb) of memory for $Sbox_{\mathrm{T}}$ and 4Kb for $Sbox_{\mathrm{F}}$. It is due to the global synchronization signal that the memory utilization is increased by 2^{n+2} and not 2^{2n}. Therefore, BCDL is good for FPGAs in terms of cost, speed and security.

5.2.5.1 AES BCDL using T-box

As stated earlier, by using t-box the fanout of register and other gates can be significantly reduced. This property is useful for a DPL countermeasure. An AES round using t-box can be expressed as in (5.3) where j=[0,3] (Figure 5.20).

$$e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j-1}] \oplus T_2[a_{2,j-2}] \oplus T_3[a_{3,j-3}] \oplus K_j \tag{5.3}$$

Since t-box is a pre-computed table, it could be implemented in a block RAM. An AES implemented using t-box has the same four primitives as AES BCDL i.e. memories, XOR gate, multiplexers and registers. Each BCDL t-box uses 16Kb (512×32) of RAM as compared to 8Kb (256×32) for single rail. We would like to remind again that the number of t-boxes is doubled in BCDL.

Table 5.8: Cost & Performance Comparison of AES BCDL using T-box on Stratix II.

Architecture	AES BCDL (S-box)	AES BCDL (T-box)
ALUT	2302	2669
Registers	1041	1041
RAMs (4 Kb)	40	72

Since Stratix II has memory blocks of size 4Kb, Quartus optimises our design to finally use 72 4K block of RAMS: 64 for t-boxes and 8 for s-boxes in KeySchedule. The number of registers is the same as in AES BCDL using s-box. The main concern is logic utilization which is unexpectedly higher than in other cases. We analyzed the synthesized design and found that the cause of high logic utilization is the secure XOR gate. In AES BCDL using t-box, there are four 5-input 32bit XOR operations where each bit of XOR takes at least 4 ALUT. Detailed results are shown in Table 5.8. This optimization is unwanted as the requirement of unitary fanout is not achieved. Still there is a reduction of fanout from the BCDL s-box to BCDL t-box and this can be analyzed by SCA.

5.2.6 Security Evaluation of BCDL against SCA on Altera Stratix II

We implemented three implementation which are single-rail unprotected AES (using s-box) as reference, AES in BCDL using s-box, AES in BCDL using t-box. We acquired 150000 traces from SASEBO-B board where the designs were implemented on the control FPGA ($EP2S30$) by placing an antenna over a decoupling capacitor. The traces were sampled on a Agilent Oscilloscope at a sampling rate of 5 GSa/s. Placement and routing are done automatically

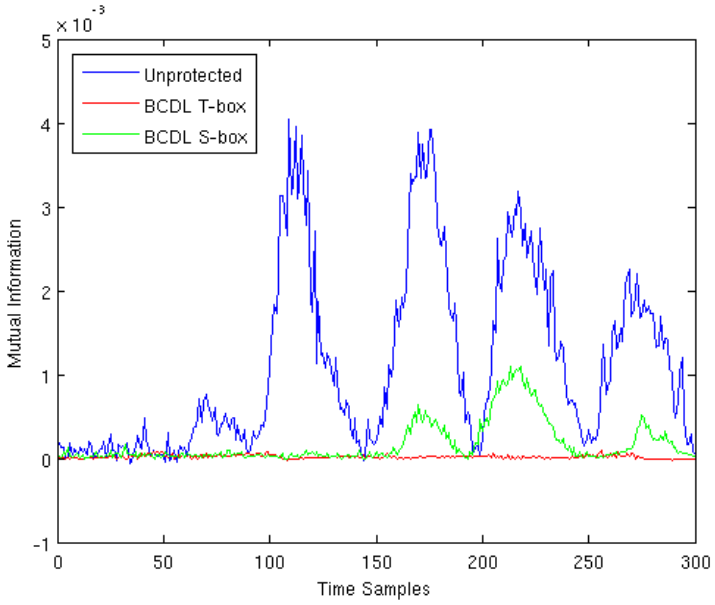


Figure 5.22: MI trace comparing information leaked by three implementations for most significant byte of AES datapath.

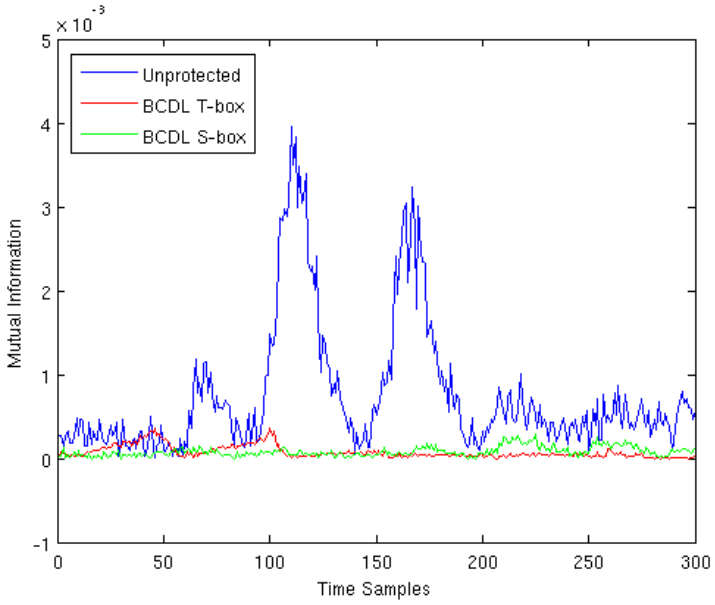


Figure 5.23: MI trace comparing information leaked by three implementations for second most significant byte of AES datapath.

by the FPGA tool because we want to observe the security gain achieved just by reduction of fanout. Other back-end balancing methods can always be applied to further improve the routing. A CPA using Hamming weight model is used to analyze the acquired traces. The minimum number of traces required to disclose (MTD) the key is **135872** traces for AES BCDL using t-box as compared to **9291** traces for AES BCDL using s-box. The reference AES was broken in **850** traces. In [64], authors have had similar observations. They demonstrate that by using BRAM in a implementation of a AES s-box protected with SDDL logic the DPA resistance is increased by a factor of 2.5.

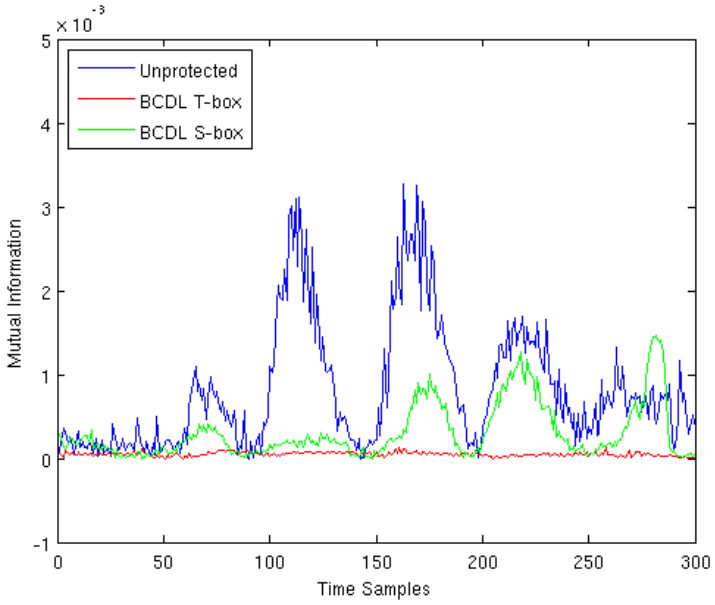


Figure 5.24: MI trace comparing information leaked by three implementations for third most significant byte of AES datapath.

Further analysis was done using MIA. Here the implementation was not attacked but the mutual information (MI) present in the traces is revealed. These result are in accordance with the CPA results calculated for the correct key using a Hamming weight leakage model for BCDL and Hamming distance for single rail. We compute the MIA trace for the four most significant bytes of the AES datapath. The results are shown in figures 5.22- 5.25.

We equally computed a MI trace using the mono-bit model, which shows the information leaked by each bit of the most significant byte of the AES datapath (Figure 5.26). The leakage in AES BCDL using s-box is evident. Another point to note is that leakage is different for different bits. This is owing to the difference in routing imbalance. On the other hand,

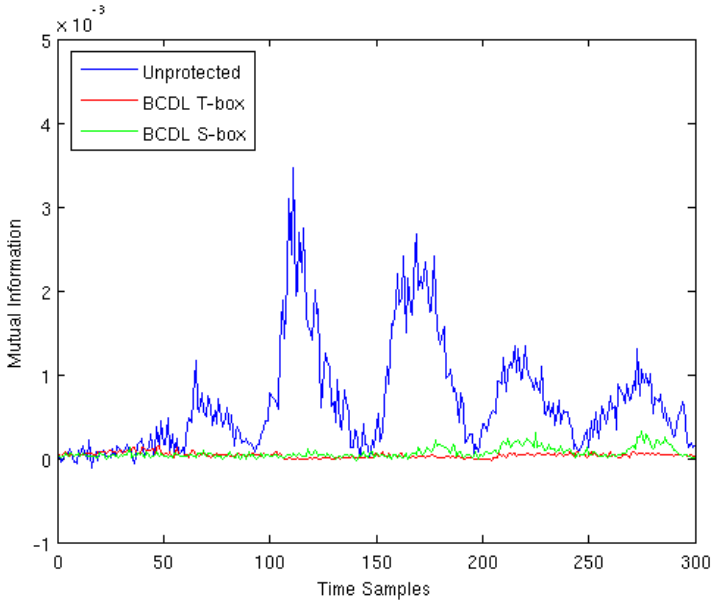


Figure 5.25: MI trace comparing information leaked by three implementations for fourth most significant byte of AES datapath.

the 8 bits in AES BCDL using t-box does not leak significant amount of information. Next we zoom on the MI trace of AES BCDL using t-box for further analysis as shown in . Although the leakage is much less, however some leakage is always present Figure 5.27. This leakage should come from routing imbalance introduced due to optimization of our design.

5.2.7 Initial Results on Xilinx Virtex V FPGA

As mentioned before Stratix II has block memories of 4K each. For unitary fanout we need a block memories of at least 16 Kb. Therefore we decided to move towards Xilinx Virtex V as available on SASEBO-GII. Another reason for moving towards Virtex V FPGA is the possibility of fixing the imbalanced routing manually. SASEBO-GII has two FPGAs soldered on it: a Spartan III ($XC3S400A-4FTG256$) and a Xilinx Virtex V ($XC5VLX50$). We implemented the same three designs as previously. The results are shown in Table 5.9.

The number of LUT and registers are roughly doubled and quadrupled respectively from reference AES to AES BCDL using s-box as expected. Normally, AES BCDL using s-box needs 4 times more RAM than reference but in single rail the block RAM were underused so the net utilization is only doubled. These results are much better than other EPE-free DPL logic where the logic utilization is increased by a factor of 4 to 6. Still AES BCDL using s-box can

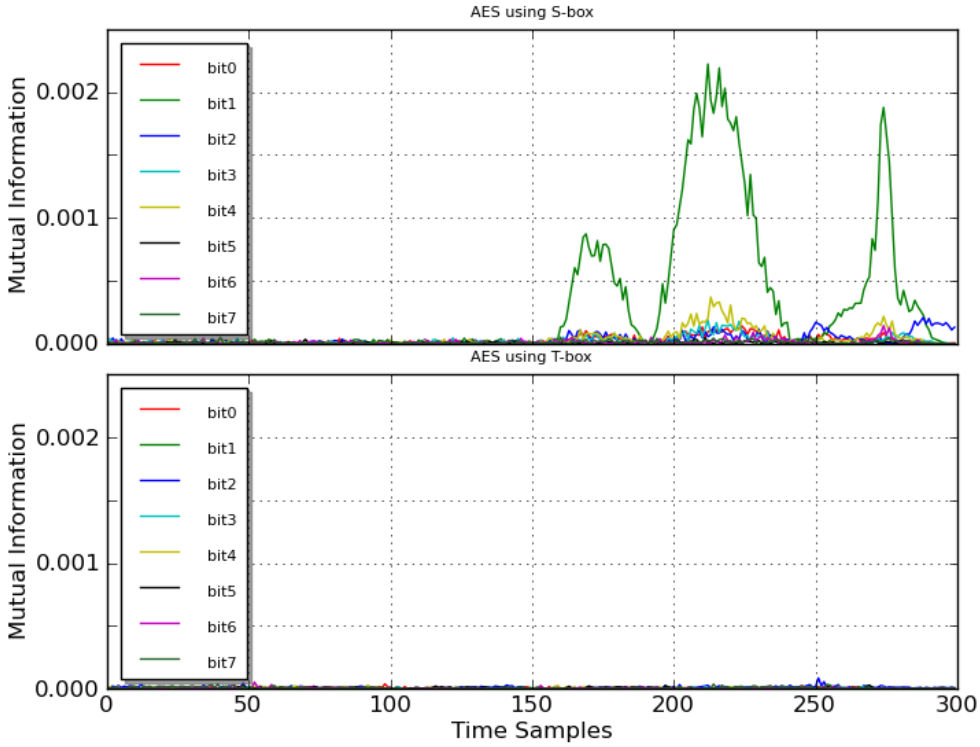


Figure 5.26: MI trace comparing information leaked by two implementations for most significant byte of AES datapath using mono-bit leakage model.

Table 5.9: Cost & Performance Analysis on Xilinx Virtex V.

Architecture	Reference	BCDL (S-Box)	BCDL (T-Box)
LUT	892	1768	2217
Registers	264	1034	1034
RAMs (36 Kb)	5	10	34
Freq. (MHz)	146.8	112.5	127.2

have the problem of imbalanced routing. The floorplan of AES BCDL using s-box and t-box were analyzed to compare the routing difference for two signals with the highest fanout. We found that highest fanout for AES BCDL using s-box is 8 compared to 2 for AES BCDL using t-box. Figure 5.28 (a) shows routing of the true signal which is very different from the false of a DPL signal in AES BCDL using s-box. Clearly there is imbalance which could be a potential leakage source.

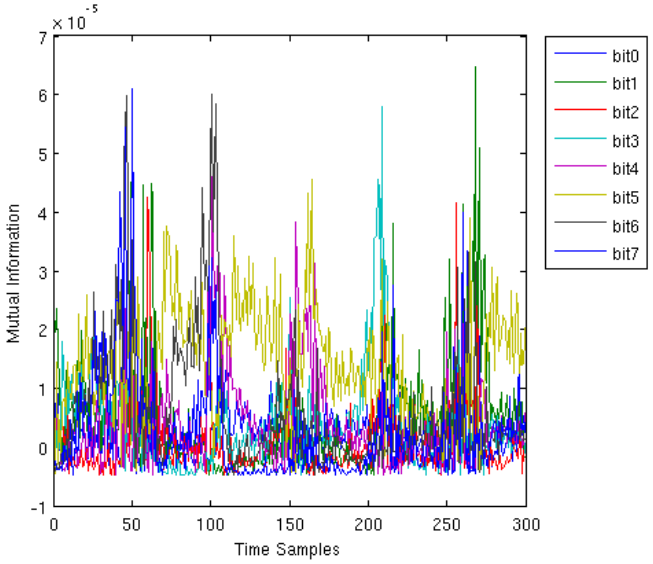
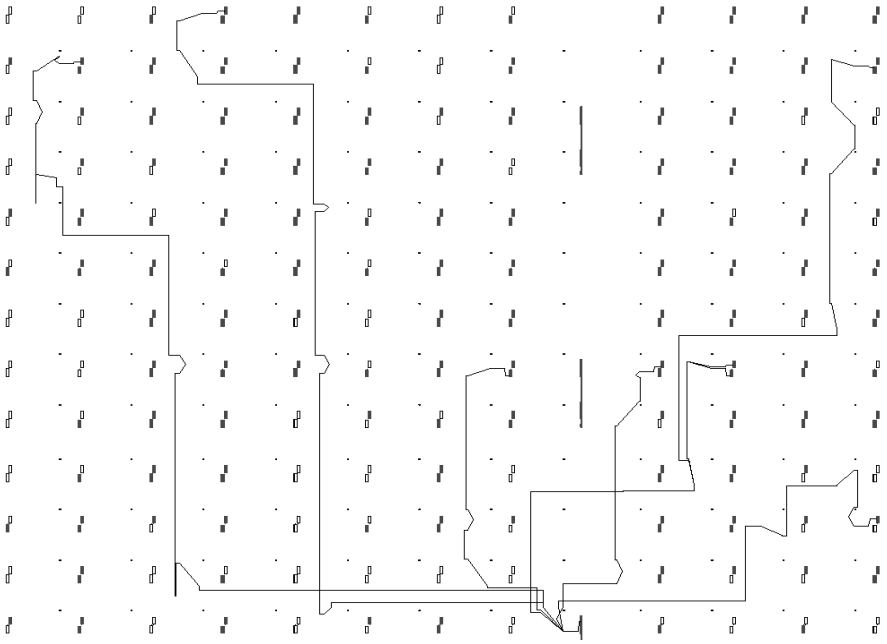
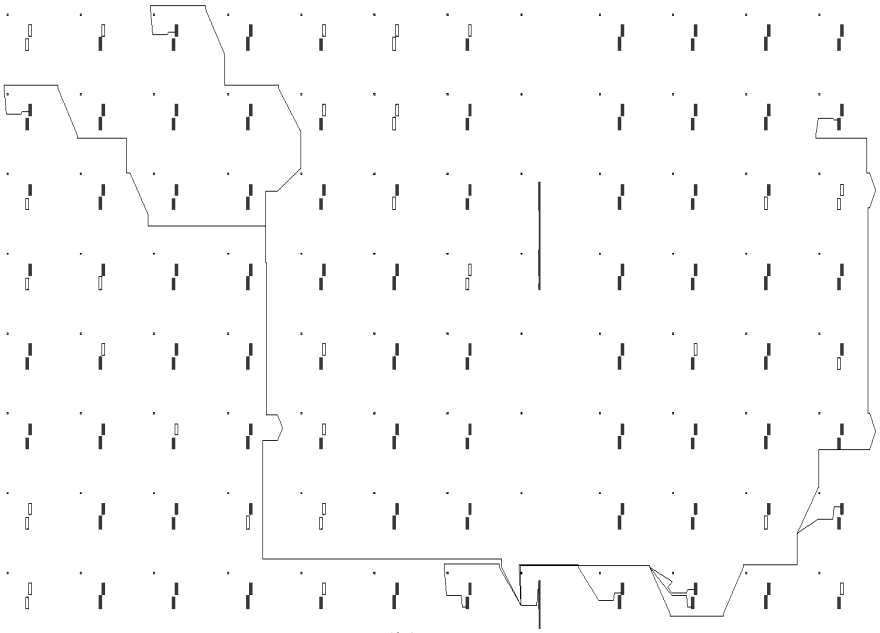


Figure 5.27: Zoom on MI trace of AES BCDL using T-Box showing information leaked for most significant byte of AES datapath using mono-bit leakage model.

Now AES BCDL using t-box, consumes 34 blocks of 36Kb RAM: 32 blocks for 32 t-boxes and 2 blocks for 8 S-Boxes used in KeyExpansion. The RAM utilization can be further reduced to half as the 36Kb block RAM in Virtex V is a true dual port RAM. This means 2 t-boxes of 16 Kb each can be be fitted into same block RAM without any compromise with the fanout. Same area optimization applies to AES BCDL using s-box. The number of registers is the same in both the BCDL implementations. Again the cause of high logic utilization is the secure XOR gate. In AES BCDL using t-box, there are 4 5-input 32bit XOR operations where each bit of XOR takes at least 4 LUT. If this constraint on XOR is removed the design uses around 400 LUT. Nevertheless with this design, the fanout of the circuit is reduced. We found a maximum fanout of 2 with majority of data signal having a unitary fanout. These few signals with fanout 2 had been optimised by the synthesis tool. Analysis of the floorplan revealed that routing, though not perfect, seems to be better than in the previous case (refer Figure 5.29). An analysis of individual path delays can be done for accurate results. We aim to perform such analysis in near future. Such architecture should leak some information but the leakage would be far less than for the s-box architecture. In terms of frequency AES BCDL using t-box lies between other two designs.



(a)



(b)

Figure 5.28: Difference in routing of (a) true and (b) false signal with fanout 8 in AES BCDL using s-boxes (not to scale).

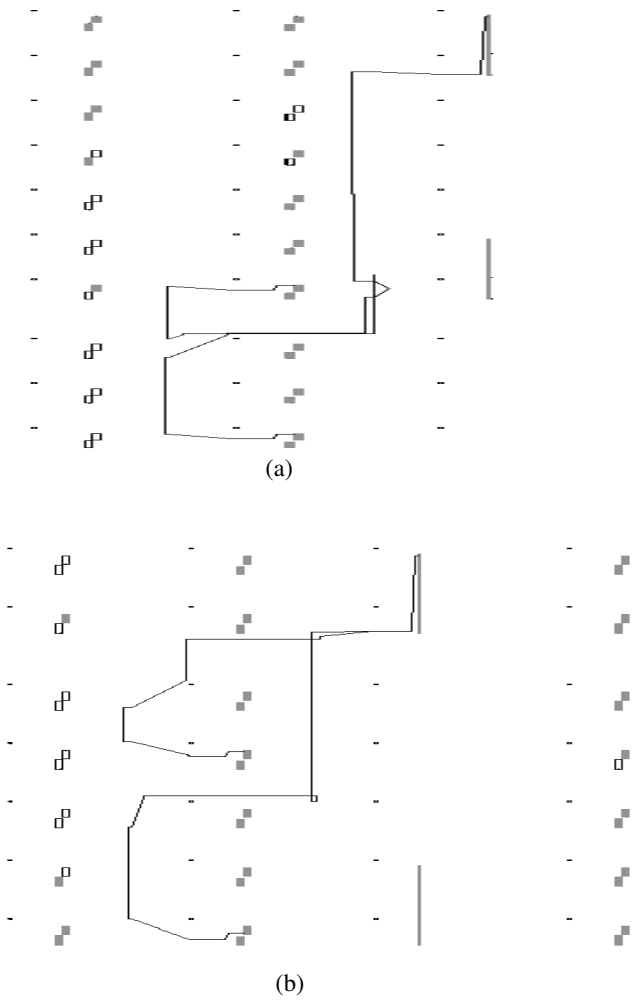


Figure 5.29: Difference in routing of (a) true and (b) false signal with the maximum fanout (of 2) in AES BCDL using T-boxes (not to scale).

5.3 Conclusions

In this chapter, we presented two new DPL countermeasures. DPL w/o EPE is a countermeasures where the truth tables of the basic gates are modified to counter EPE which is possible in FPGA by changing the LUT mask. The technique not only removes the vulnerability but also allows designers to use non-positive gates for DPL synthesis, which can enable better optimization. We demonstrate the effect of early evaluation which, to our knowledge, was only discussed theoretically before. Using mutual information metric we were able to demonstrate that only removing early evaluation and keeping everything else constant reduces the leakage to half. No significant improvement was seen by partial P&R constraints

on Altera Stratix FPGA.

BCDL is the second countermeasure proposed. BCDL is a DPL style which possess a global synchronization signal for countering EPE and enable the use of embedded memories in FPGA. We show efficient implementation of DPL circuit on FPGA. Using memories can counter, to some extent, the imbalance in routing as it reduces the fanout and number of gates in a timing path. This also allows to have an area and speed efficient implementation of DPL in FPGA. We present details of an AES BCDL implementation using s-boxes and t-boxes in Stratix 2 and Virtex V. We equally analyze these implementations on Stratix II against SCA and found that AES BCDL using t-boxes is more resistant than its version using s-boxes. Thus by reducing fanout we increase the robustness of DPL implementations on FPGA.

As we know that routing on FPGA cannot be precisely controlled, even with reduced fanout some parts of the circuit are imbalanced. However modern FPGA tools provide options to change the routing. Since cryptographic circuits are bulky, balancing each and every bit is very difficult. In this context we propose to design circuits with low fanout and to take help of SCA tools to localize the leaking bits. Once this information is known, the routing of the leaking bits can be balanced manually. Such tools are discussed in the next chapter. Another solution is the masked DPL. The idea is to balance the routing statistically by swapping the true and the false paths randomly. These methods to balance routing are interesting directions for further research.

Chapter 6

Advanced Evaluation Techniques

Logic level countermeasures to protect FPGA designs have been the focus of Chapter 3, 4 and 5. Implementation of these countermeasures on FPGA are detailed along with robustness evaluation. In this chapter we shift the focus on evaluation techniques. Evaluation techniques for unprotected implementations have been widely studied in literature. However, a formal attack methodologies has never been proposed for DPL implementations. The mono-bit attack model along with common SCA like DPA, CPA and MIA is the most used analysis technique for DPL [88][117][29]. Such attacks can give the designer a vague idea about the flaws in the design. Sometimes Hamming weight is also used but if the flaw is minor, these attacks might not work at all. As demonstrated in Chapter 5, countering early evaluation results in reduction of the leakage in side channel but some leakage due to imbalanced routing still remains. We used MIA using Gaussian parametric estimation to analyse the countermeasure.

Here we explore profiled SCA as evaluation techniques to estimate the leaked information. Profiled SCA are the most powerful type of side-channel attacks. They are divided in two different stages. The first stage known as the profiling phase uses a training device identical to the target which allows precise characterization of its physical leakage. The second stage which is an online exploitation phase is mounted on the target device in order to perform a key recovery. Standard profiled side-channel attacks include template attacks and stochastic models, respectively introduced in [78] and [118] respectively. Authors in [119][120], compare the two profiled attacks. We analyze the potential of these profiled attacks and compare them with previously used MIA in context of DPL evaluation. This analysis is done in a simulated environment as well as on real traces of a DPL implementation of FPGA.

The second contribution in this chapter is the idea of combining multiple attacks. SCA have come a long way since first introduced. Extensive research has improved various aspects of SCA like acquisition techniques, processing of traces, choice of leakage model, choice of distinguisher etc. When the targeted design is protected, the number of traces required to attack increases many times. Sometimes the encryption key is changed after a few encryptions. Such methods make the number of traces acquired a scarce resource. Few work also report the combination of few of these methods to improve the attack. We propose a different flavor of combined attacks which involve combination of different measurements acquired for the same activity. Limiting the number of encryptions with the same key does not prevent the attacker from acquiring multiple traces of the same activity. These multiple set of traces can be exploited to accelerate the attack. The method has been applied to an unprotected DES crypto-processor running on FPGA which can be simply extended to a protected implementation.

6.1 Evaluation Tools for DPL Implementations

To evaluate a DPL implementation, mono-bit DPA is one of the most used analysis techniques unlike single-rail implementations. Multi-bit DPA when based on Hamming weight or Hamming distance model perform better for single-rail designs. In DPL, the leakage is caused by the imbalance between the true and the false network, and this imbalance could be opposite for targeted bits. The imbalance may reduce or vanish by combining different bits using a leakage model like Hamming weight. Thus the power consumption difference between different bit-flips ($0 \rightarrow 1$ & $1 \rightarrow 0$) can be exploited best on a single bit [88]. Please note that the imbalance is still there but the attack might not be able to exploit it when combined. Another technique used against DPL implementations is MIA. Profiled attacks are also used commonly for evaluations, however to our knowledge, it has not been applied on DPL implementations till now. MIA has been discussed in Chapter 2. In the following, we briefly describe profiled attacks followed by application to DPL.

6.1.1 Template Attacks

Template attacks were introduced by Suresh Chari *et al.* in [78]. The salient feature of template attacks is that it characterizes the noise in measurements, unlike other approaches. The main idea is to capture an amount n of traces $C_{M,k}(t)$ (typically $n = 1000$) on the programmable device for each subkey k and to describe the behaviour of the noise depending on k. Sometimes the profiling is done on a key-dependent variable instead of the key. Each

set of $C_{M,k}(t)$ is averaged, to obtain a new set $\mathbf{A} = \{\mathbf{A}_k,\ \forall k \in K\}$. In order to reduce the profiling time, a set of points of interest has to be selected.

Let $\mathbf{T} = \{t_i,\ 1 \leq i \leq p\}$ be a set of p points of interest. For a given key k, we can now compute a noise vector for all traces $C_{M,k}(t)$ as follows:

$$\mathbf{N}_k(M) = [C_{M,k}(t_1) - \mathbf{A}_k(t_1), \ldots, C_{M,k}(t_p) - \mathbf{A}_k(t_p)] \tag{6.1}$$

Let $\mathbf{N}_{k,t}$ be the vector of all elements of $\mathbf{N}_k$ at the instant t. Now, we can compute the covariance matrix which has its elements defined as:

$$\Theta_k[t_i, t_j] = cov(\mathbf{N}_{k,t_i}, \mathbf{N}_{k,t_j}) \tag{6.2}$$

The couple $(\mathbf{A}_k, \Theta_k)$ is the template for the key k. The profiling phase is finished when a template is computed for each key $k \in K$ or key-dependent variable.

The key extracting phase uses the maximum likelihood principle. For each key k and for each measured trace, we compute a noise vector $\mathbf{n}$ on the points of interest (using $\mathbf{A}_k$). Thereafter we compute $f_k(\mathbf{n})$, where f_k is the multivariate Gaussian distribution, as follows:

$$f_k : \mathbb{R}^p \to \mathbb{R} \qquad f_k(\mathbf{n}) = \frac{1}{\sqrt{(2\pi)^p . |\Theta_k|}} e^{-\frac{1}{2}\mathbf{n}^{\mathrm{T}}\Theta_k^{-1}\mathbf{n}} \tag{6.3}$$

where $|\Theta_k|$ is the determinant of Θ_k. $f_k(\mathbf{n})$ will give the highest value if k is the good guess. It gives the probability of each key candidate. Once the probability of each key candidate or key-dependent variable is known, we can compute the entropy and eventually the mutual information.

6.1.2 Stochastic Model Attack

Stochastic Models are also a type of profiled attacks proposed by Schindler et al [118]. The profiling phase needs only one test key i.e. the power consumption is modeled, at a time t as follows:

$$W_t(x, k) = h_t(x, k) + \mathscr{B}_t \tag{6.4}$$

where x is the plain text and k the key. The first summand h_t is the deterministic part of the power consumption (which depends on x and k) and $\mathscr{B}_t$ a random noise with zero expectation ($\forall t, \mathbb{E}(\mathscr{B}_t) = 0$). The first step of the profiling phase consists in approximating

h_t, followed by estimation of $\mathscr{B}_t$ using h_t. h_t is assumed to have the *EIS* property (Equal Image under different Subkeys), which implies that only one test key is needed for the profiling phase. Let $\tilde{h}_t$ be the best estimation of h_t computed as:

$$h_t(x,k) = \beta_0 + \sum_{i=1}^{u} \beta_{it} g_i(x,k) \tag{6.5}$$

where the g_i are chosen base functions, which depend on x and k, and β_{it} are coefficients, which estimates the system. It is the choice of base functions which define the degree of stochastic models. A linear model takes just a function of individual bits where as a higher degree model considers multiple bits for each coefficient. We assume that β_0 is always equal to 1. As we attack the output of an AES s-box, in the linear model we take the 8-bits at the output of the s-box as base vectors giving base function of length 9 (8 bits and 1 constant). For the second degree base vectors we take the product of all the individual bits which gives us 37 (1+8+28) base vectors. Similarly base vectors of degree 3 and 4 are of length 93 and 163 respectively. The second step of the profiling phase consists of characterization of the noise. First, some relevant instants have to be selected (*e.g.* by using the T-Test or Euclidian norm of the coefficients β_{it}). The noise is characterized by constructing the probability density function of the multivariate normal distribution, using a covariance matrix (computed with a noise random variable associated on each point of interest). When the first device is profiled, attack can be performed using the maximum likelihood principle.

6.1.3 Experimental Results

We conducted two sets of experiments. The first experiment is based on simulated traces. The aim of this experiment is to observe the behavior of each of the three evaluation tools when the environmental noise is varied. Thereafter, we test these tools on real traces acquired from a AES implementation protected with DPL running on a Altera Stratix FPGA.

6.1.3.1 Simulated traces

We generated some simulated traces to study the behavior of evaluation tools under affect of Gaussian noise. The simulation were done in the MATLAB environment. A vector v of length 50000 containing natural numbers between 0 and 255 was generated. We also computed a vector containing the Hamming weight (HW) of each element of v. This was followed by generating a vector of simulated leakages l_i given by:

$$l_i = HW(v_i) + n_i \tag{6.6}$$

where n_i is the Gaussian noise added. We conducted various experiments on simulated traces l_i, each time varying the variance of added noise. Let $v_i = Sbox^{-1}(x \oplus k)$ is the intermediate value i.e. one byte at the output of round 9 of AES based on which the profiling was performed. We built templates as well as stochastic models of degree 1,2,3 and 4 for the value v_i to compute the mutual information. We also compute the mutual information using MIA with Gaussian parametric estimation. Figure 6.1 shows that when the leakage is perfectly linear (l_i), the stochastic models of linear degree is as accurate as template attack.

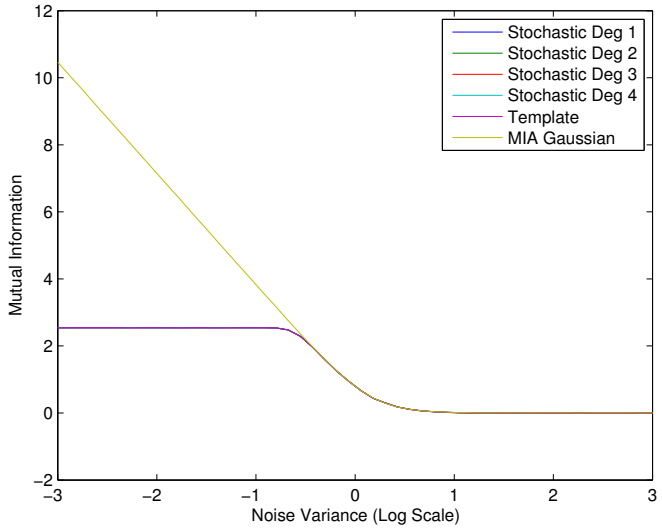


Figure 6.1: Mutual information for simulated traces with linear leakages model as a function of added Gaussian noise.

An interesting observation is that when the added noise is very low, the MIA using Gaussian parametric estimation seems to overestimate the information present in the traces. Figure 6.2 shows four example of distributions for varying value of added Gaussian noise. When the added noise is zero or low, each partitions has a discrete Gaussian distribution which can be estimated by a bigger Gaussian. Thus for small values of additive noise, the basic assumption of MIA using Gaussian parametric estimation does not hold true and gives erroneous results. However, when we increase the level of noise then the mixture of Gaussians can be estimated by a single bigger Gaussian. Therefore the information estimated by MIA for reasonable amount of added noise is reliable and is comparable to templates and stochastic models. Next we apply these methods on real traces to check their validity in a real scenario.

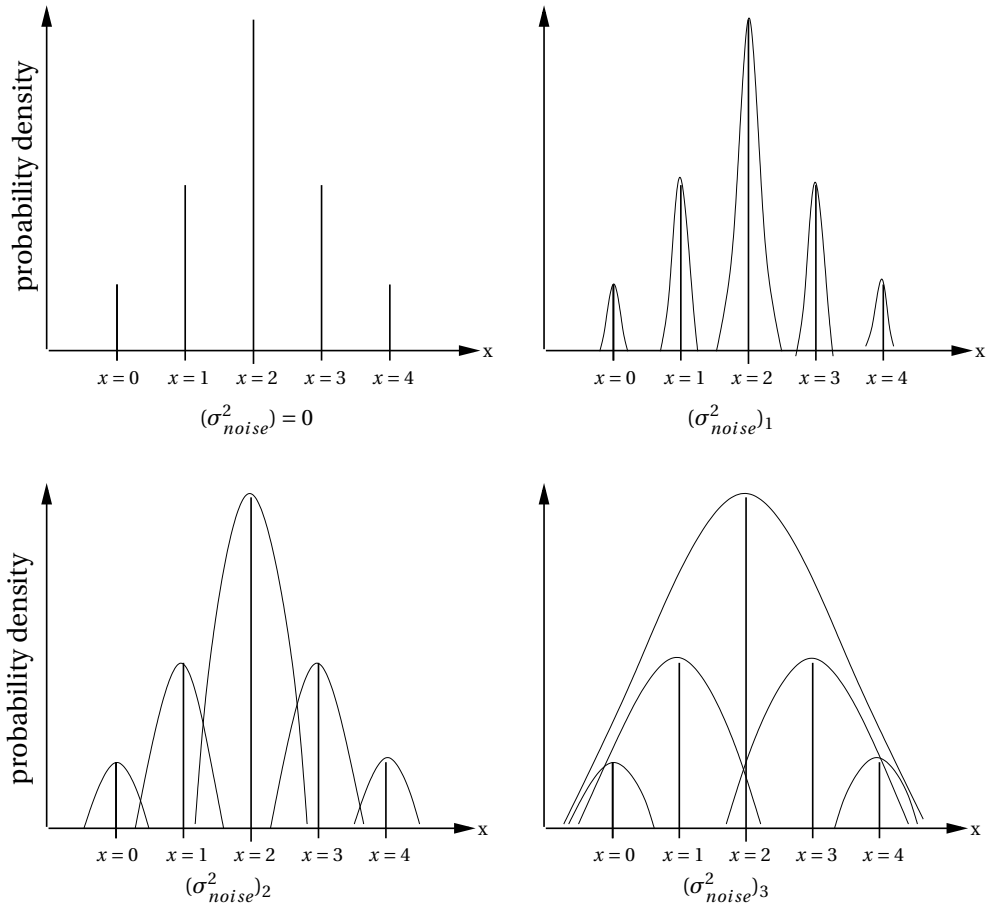


Figure 6.2: Examples of distributions with additive Gaussian noise.

6.1.3.2 Real Traces

After studying the effectiveness of the evaluation tools on simulated traces, we decided to test them on real traces. We attack the traces acquired from an FPGA implementation of AES protected with DPL w/o EPE which are also used in Figure 5.2. When dealing with simulated traces, the Gaussian noise assumption works fine because the introduced noise is perfectly Gaussian. However, this might not be case with real traces. For a low noise case, estimation using a bigger Gaussian distribution is erroneous. We acquired a set of 90000 traces. 50000 traces were used in the profiling stage i.e. building templates and approximating h_t and $\mathscr{B}_t$. For the attack a set of 10000 traces was taken. Figure 6.3 shows the mutual information calculated using templates and stochastic model of degree 1,2,3 & 4. We also computed MIA with Gaussian parametric estimation on 90000 traces.

Information estimated by templates is higher than the stochastic model of degree 1. This

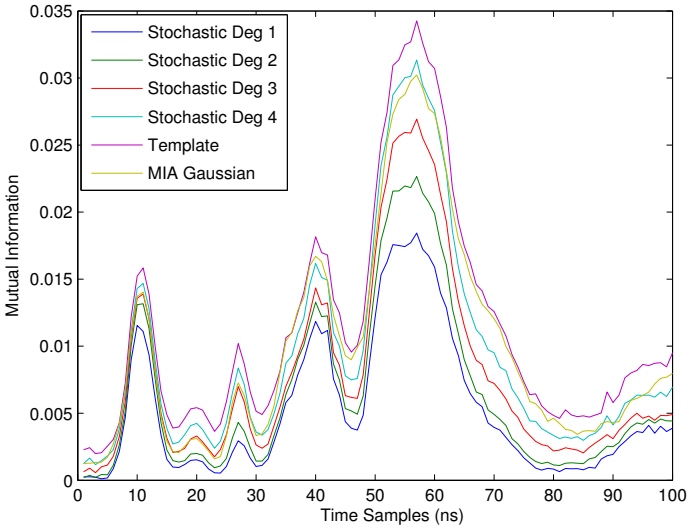
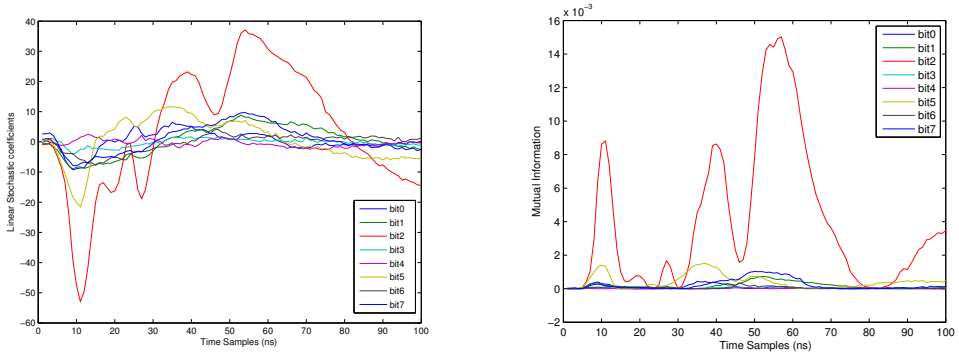


Figure 6.3: Mutual information estimated using templates, stochastic model of degree 1,2,3,4 and MIA.

means that the leakage is not linear. Stochastic model of degree 2,3 and 4 adds further information. Information estimated by stochastic model of degree 4 is approaching the information estimated by templates. Therefore in this case computing a stochastic model of degree 4 which is computationally simple, will be as good as template. An advantage of stochastic models over templates is that stochastic models are capable to localise the leakage source. In this case, coefficients show that the 3^{rd} bit (bit 2) of the targeted s-box is leaking much more than other bits (Figure 6.4). This information was called β-characteristics in [121]. β-characteristics are of use for designers to improve their design. It is clear from Figure 6.4(a) that mostly bit 2 is leaking.

On the other hand, MIA using Gaussian parametric estimation does not overestimates the information. It seems to perform almost as good as stochastic models and templates. We also performed a mono-bit MIA on the same traces. The results are shown in Figure 6.4(b) where it is clear that mostly bit 2 is leaking. Analysis of the FPGA floorplan confirmed that this particular bit was asymmetrically routed. Thus mono-bit MIA has an advantage of pinpointing the leakage like stochastic coefficients.



(a) (b)

Figure 6.4: Localizing the leakage using (a) Linear Stochastic coefficients and (b) Mono-bit MIA.

6.1.4 Discussion

From the experiments presented above we can infer that template attacks provide the best estimation but are not capable of pinpointing the vulnerabilities. On the other hand stochastic models of first degree is as good as template when the leakage is linear. Templates are more efficient when the leakage is not linear which is common in real traces. Higher degree stochastic models can be explored to approach the estimation of templates in such cases. The main advantage of stochastic models in DPL evaluations is that they are capable of pinpointing the vulnerabilities. They can tell exactly which bits are leaking. Stochastic models can also outperform templates when the number of traces are insufficient as stochastic models perform an approximation with fewer degrees of freedom. Finally MIA using Gaussian parametric estimation can sometimes overestimate the information. In our case, the traces acquired were quite noisy and it seemed to work fine nevertheless the risk of overestimation is present. MIA is also able to pinpoint the leakage as in Stochastic models when used in mono-bit mode. An advantage of MIA using Gaussian parametric estimation is that it is easy and faster to mount and no profiling is required. Comparing the three we can say that stochastic models are best suited for evaluating DPL designs because they are capable of correctly estimating information and localizing the leakage.

6.2 Combination of Measurements

A common practice to carry out Electromagnetic Analysis (EMA [122]) is to acquire the strongest and most obvious leakage points on the device. However there are other points

which leak information as well. We propose to acquire multiple simultaneous leakages from different leakage points. Multiple antennae can be used to acquire multiple leakages. These multiple leakages for a single activity could be combined for an efficient SCA. Multi-channel attacks have already been introduced in [123] for mono-bit DPA and template attacks. Here we give a more generic outlook towards combing measurements using any distinguisher. We also provide a metric based on information theory to test if the possibility of combination exists for a given pair of traces.

Cartography is often used to reconstruct a dynamic image of the device using a sensor. An attacker can use this dynamic image in identifying the areas where the information leakage is the most intense [122]. As a matter of fact, the electromagnetic radiations correlated to a given process are not necessarily produced at the exact location of the processing zone. The power lines or clock paths leak more information and therefore power supply and ground networks as well as the clock buffer trees are of special interest. Decoupling capacitors which can leak radiated emanations about an internal process are another interesting source of leakage. An EMA starts with research of a relevant leakage point for capturing EM radiations. An attacker can choose between the two aforementioned techniques to perform a complete cartography of the chip or carefully choosing a decoupling capacitor. When dealing with complex cryptographic circuits which are often bulky, several leakage points are identified. Some of these leakage points provide enough leakage to mount a successful attack, however the speed of attack could vary. A common practice is to choose the point which could lead to the fastest attack. We put forward a methodology to combine leakages from several leakage points in order to accelerate the attack. Thus during a single encryption for a given message and a fixed secret key we use multiple antennae to capture the radiation from different chosen points. A combination of power measurement and EM measurement can also be used.

6.2.1 Theoretical Background

Information gain of a single attribute X with respect to class C, also known as mutual information between X and C, measured in bits is:

$$Gain_c(X) = I(X;C) = \sum_x \sum_c P(x,c) \log \frac{P(x,c)}{P(x)P(c)} \tag{6.7}$$

Equivalently:

$$I(X;C) = H(X) - H(X|C) \tag{6.8}$$

Here $H(X)$ gives the entropy of X and $H(X|C)$ gives the conditional entropy of X knowing C. To simplify the calculation of entropy we consider the distribution of X is Gaussian. In this case entropy can be calculated as a function of standard deviation σ_x of X as:

$$H(X) = -\sum_i p(x_i)\log_2(p(x_i)) \tag{6.9}$$

$$H(X) = \log_2(\sigma_x\sqrt{(2\pi e)}) \tag{6.10}$$

This method might not be ideal for estimating entropy but works well in practical cases as shown previously. Nevertheless other methods of estimating entropy can be applied. Information gain can be regarded as a measure of the strength of a 2-way interaction between an attribute X and the class C. 3-way interactions were introduced as interaction gain [124] which is equivalent to mutual information of 3-variables. Interaction gain is also measured in bits, and can be understood as the difference between the actual decrease in entropy achieved by the joint attribute XY and the expected decrease in entropy with the assumption of independence between attributes X and Y. Interaction gain can be considered equivalent to multivariate mutual information [125]. The Venn diagram representation is shown in Fig. 6.5.

$$\begin{aligned} I(X;Y;Z) &= I(X,Y;Z) - I(X;Z) - I(Y;Z) \\ I(X;Y;Z) &= (D+F+G) - (F+G) - (D+G) = \textbf{-G} \end{aligned} \tag{6.11}$$

As per Eq (6.11), interaction gain is equal to **-G**. If X and Y are independent, I(X,Y;Z) = I(X;Z)+I(Y;Z). This means the interaction gain I(X;Y;Z) is zero. Interpreting from Fig 6.5(a) and (b) combination is possible when the information equal to D is added to I(X;Z) with introduction of Y. This makes I(X,Y;Z) = D+G+F. If D is zero, then introduction of Y is not providing any extra information.

To check this condition we propose a simple test. The possibility of combination (PC) can be calculated as a ratio:

$$PC = \frac{Max(I(X;Z), I(Y;Z))}{I(X,Y;Z)}. \tag{6.12}$$

For a combination to exist, PC should lie between 0.5 and 1, where PC=1 will suggest no combination is possible. In context of combined attacks, interaction gain can be directly applied. This is a profiling step because knowledge of the secret key is required to calculate PC. Alternatively PC can also be used as an distinguisher. Here we focus on combining measurements using a common SCA like CPA.

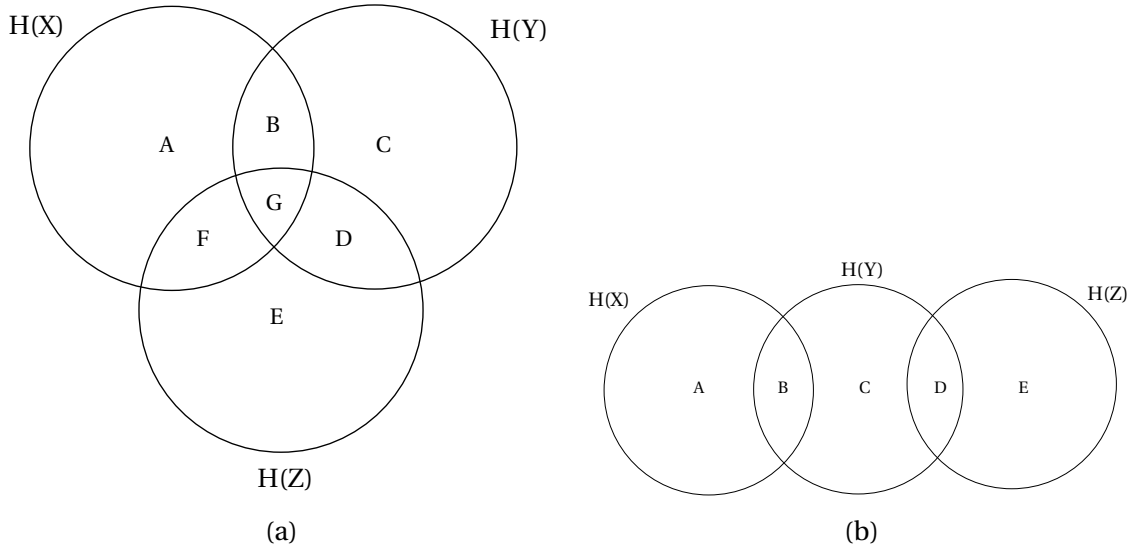


Figure 6.5: Venn diagram representation of a case when combination is (a) possible, (b) not possible.

6.2.2 Practical results

Our measurement setup consists of one Altera Stratix-II FPGA soldered on an SASEBO-B platform, an 54855 Infiniium Agilent oscilloscope with a bandwidth of 6 GHz and a maximal sampling rate of 40 GSa/s, antenna of the HZ–15 kit from Rohde & Schwarz. We recorded 5000 side-channel traces (averaged 256 times) related to the activity of an unprotected DES crypto-processor. We target two decoupling capacitors on the backside of the FPGA which show emanations corresponding to a DES execution. As the number of capacitors on the FPGA are limited, trial-and-error method was efficient for choice of capacitance. We place an antenna close to each of the capacitor as shown in Figure 6.6. We collect two sets of 5000 traces from two chosen capacitors for the same dataset. A crypto-processor is a bulky design and could be spread over different power banks in an FPGA which are terminated by different capacitors. Therefore different capacitor leak more information about a certain part of the circuit. In our circuit, partition can be seen as different s-boxes.

We start with computing the metric PC. Figure 6.7 shows the ratio PC for two cases. We computed the value of PC for Sbox 0 in each case. Figure 6.7(a) considers traces from two capacitors which are leaking relevant information. It can be seen that the value of PC is close to 0.5 when the value of mutual information is relevant. Figure 6.7(b) considers traces from a leaking capacitance and another point which is not leaking. Here the value of PC is close to 1. The value of mutual information of the two measurements is multiplied by 100

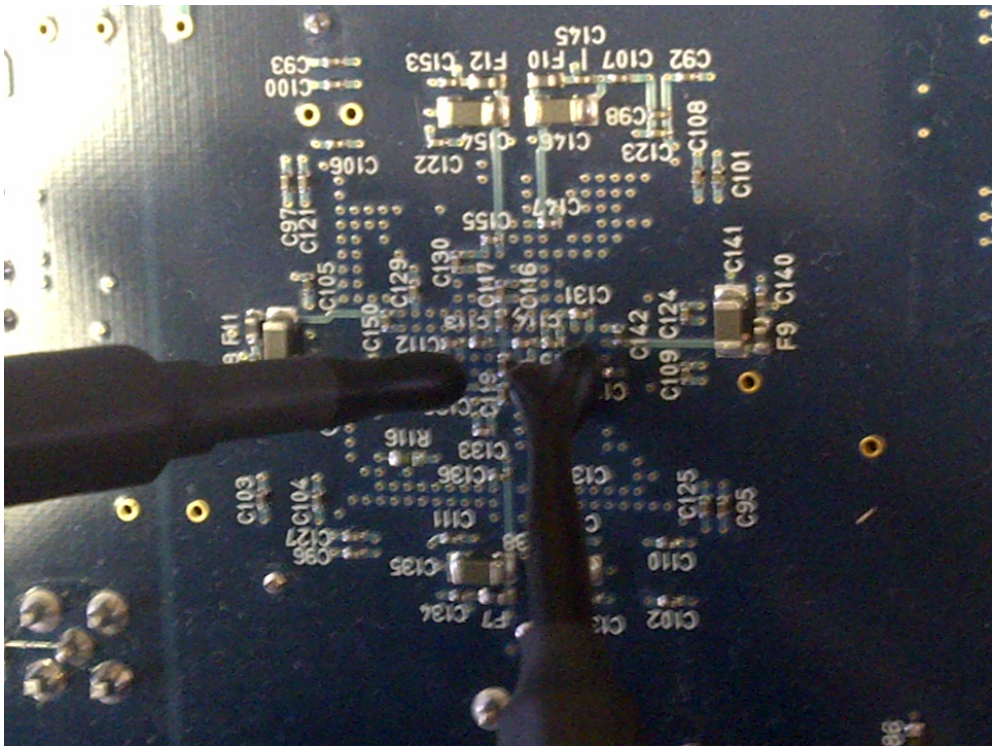


Figure 6.6: Placement of antennae for a combined attack based on combination of measurements.

to depict it on the same scale as PC. This means that combination is possible for the traces in Figure 6.7. Unfortunately we did not had a set of traces to have a case between to the two extremes. In the other parts of the trace there is a noise and the value of PC is randomly changing.

Next step is to observe practical application of combination of measurement using a common attack like CPA. We applied CPA on the traces collected from capacitance C_1 and C_2 independently. Table 6.1 summarises the result of CPA on each set of traces. These results are averaged over 30 attacks. We see that C_1 is better suited for s-box no. 0,1,3 and 7 and C_2 for the rest. Before testing the combination, we concatenate traces of C_1 and C_2 together i.e. we join the the trace acquired from C_2 at the end of the trace acquired from C_1 . Traces can be normalised before concatenation specially when techniques like principle component analysis are applied but if the traces are taken with the same scale then normalisation will not help a lot. In our experiments the traces are taken with the same scale on the oscilloscope therefore normalisation is not needed.

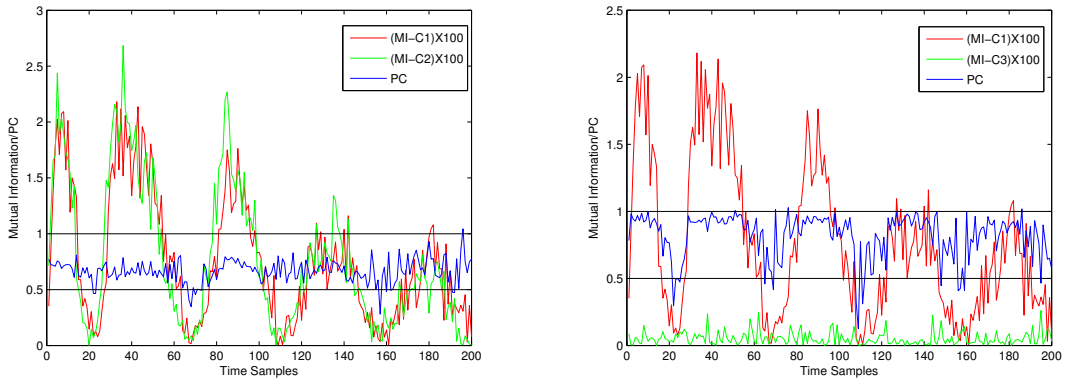


Figure 6.7: Calculation of PC for two cases when combination is (a) possible, (b) not possible.

We launch an attack on the concatenated trace. The attack calculates the Pearson correlation coefficient of the key hypothesis for each trace on each of the two sections of concatenated trace. To combine we use an aggregate function Ψ.

An aggregate function is a special type of operator that returns a single value based on multiple rows of data. For each key hypothesis, k, a new score is generated by computing $\Psi((\Delta_k)_{trace_1},(\Delta_k)_{trace_2})$, which is the aggregate function of $(\Delta_k)_{trace_1}$ and $(\Delta_k)_{trace_2}$. Δ_k denotes an output of the attack which is combined like differential trace, rank of the key, correlation coefficient etc. This way, a new vector of scores, denoted by $\Delta_{vect_{combi}}$ is built. An illustration of the combination mechanism is shown in Fig. 6.8.

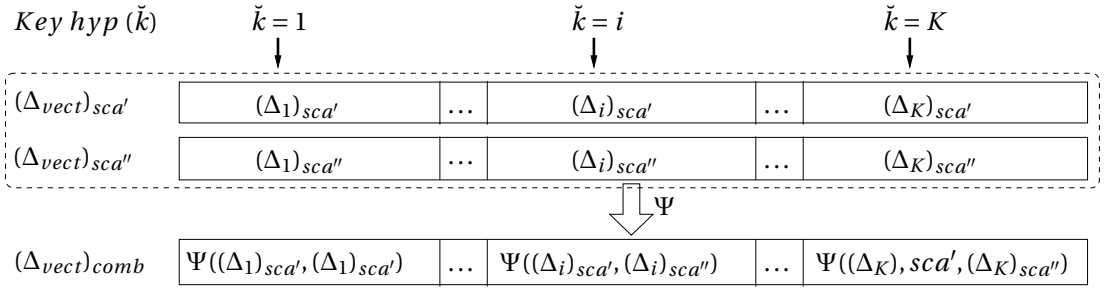


Figure 6.8: The mechanism of combination using an aggregate function Ψ.

Precisely, the used aggregate function in this experiment is the Sum() on the calculated coefficient value. Let S_1, S_2 be two inputs of aggregate function with respective noise of standard deviation σ_1, σ_2. The SNR of the output of Sum is $(S1+S2)/\sqrt{{\sigma_1}^2+{\sigma_2}^2}$. When S_1 and S_2 are equal the SNR of the output Sum is increased by $\sqrt{2}$. It is shown that Sum()

can increase the SNR even if the two traces contain equivalent information. If the amount of information is not equivalent Sum() will further increase the SNR hence a faster or more efficient attack. Next we demonstrate the combination using Sum aggregate function with Pearson coefficient on real traces.

Table 6.1: No. of traces to attack using C_1, C_2 and combination of both.

Sbox No.	C_1	C_2	Sum	**Gain**
0	350	432	212	39.42%
1	943	1073	750	20.46%
2	733	720	397	44.86%
3	400	980	251	37.25%
4	410	176	165	06.25%
5	320	281	270	03.96%
6	548	551	448	18.24%
7	592	192	184	04.16%

The computation complexity of combination is equivalent to processing a trace with twice the number of samples with minor overhead of applying the aggregate function. Two parallel attacks on non-concatenated traces will have similar computation overhead but concatenation makes it easy to manage the key hypotheses and apply aggregate functions. Table 6.1 shows the number of traces required to attack when combination is applied. We find that in each case the combination is better than individual attack and the gain varies from 3.96-44.86%. This also complies with our PC test. For each sbox we found PC smaller than 1 and positive. The scale of PC and gain cannot be compared as each quantity is computed using different method. As mentioned before some countermeasures change encryption key after a specific number of encryption to prevent SCA. Since the number of traces acquired is considered a scarce resource, we demonstrate that multiple measurements can be exploited to speed up the attack.

6.3 Conclusions

In this chapter, we first presented a methodology to evaluate DPL implementations. A careful evaluation can reveal the leakage sources. The information on imbalanced wires is critical for a designer who can remove the security loop holes. We compared common evaluation tools which are template, stochastic models and MIA using Gaussian parametric estimation. On comparing them we found that stochastic models are most appropriate

for evaluating DPL implementations. Templates cannot localize the leakage while MIA can overestimate the information.

Then combination of measurement is presented as a method to accelerate SCA. We provide theoretical background based on information theory metrics to test combination by computing possibility of combination PC. Then it is shown how aggregate functions can be used to combine two measurements in a common attack like CPA. Practical results show a gain of upto 45% using this method. Please note that we intend to propose methodologies to accelerate attacks and not attacks in particular. Depending on the target different aggregate functions can be used to improve the performance of the combined attack. Choice of aggregate functions also depends on the practical behavior of distinguisher.

Chapter 7

Conclusion and Perspectives

7.1 Summary

This thesis is focused on logic-level information hiding countermeasures for FPGA. Countermeasures when implemented at logic level can be used across various platforms with little optimisations. In the context of work presented, various platforms refer to FPGAs from different vendors. In the following I conclude the key points of this work.

Chapter 3 presented a RTL level countermeasure based on loop unrolling of a cryptographic algorithm and known as called "Unrolling". This countermeasure is shown to be robust against SCA at a cost much less than n, where n is the unrolling factor. The only constraint necessary to achieve SCA resistance is to randomly precharge the datapath, before each encryption/decryption. This chapter also compared different implementations of AES s-box w.r.t cost and robustness against SCA and FA.

Chapter 4 summarized DPL countermeasures. I analyzed FPGA implementation of a WDDL AES co-processor against fault attacks. Underpowering of FPGA is used to inject setup-time violation faults. I showed that WDDL is immune to faults except some very well located faults. To break WDDL, the attacker needs to inject a symmetric fault on both the nets of a dual-rail pair which is either unlikely or requires expensive equipments. This property of fault resistance in WDDL is then shown to be inherent in DPL by construction. Recently, a new family of attack was introduced by Li et al. called fault sensitivity analysis (FSA [126]). It would be interesting to analyze DPL against FSA

In Chapter 5, I proposed two new DPL countermeasures. DPL w/o EPE is a countermeasures where the truth table of the basic gates are modified to counter EPE which is possible in FPGA by changing the LUT mask. I showed the effect of early evaluation on side channel leakage experimentally. No significant improvement was seen by partial $P\&R$ constraints on

Altera Stratix FPGA. Then I explained BCDL which features a global synchronization signal to counter EPE and enable use of embedded memories in FPGA. By using memories, specially for an AES with t-boxes, the fanout of the design is drastically reduced which further reduces the number of dual-rail nets to be balanced. Without any balancing at the back-end I demonstrate a security gain of around 14. BCDL with memories also allows to have an area and speed efficient implementation of DPL in FPGA.

In Chapter 3, 4 and 5 DPA, CPA and MIA are used for SCA evaluation. Chapter 6 compared other evaluation techniques like templates and higher order stochastic models against MIA using Gaussian parametric estimation. The MIA is shown to overestimate information in low noise traces but is quite reliable in practical cases where a significant amount of acquisition noise is always present. The templates are the best for estimating information leakage but cannot pin-point leakage source. Stochastic models win over the other two techniques. It can pin-point the information leakage like MIA. Higher order stochastic models are comparable to templates when estimating information. This is followed by a proposition to combine measurement to accelerate SCA. These measurements are taken from two or more different location of the target corresponding to the same activity. I provided theoretical background based on information theory metrics and showed its application to CPA using aggregate functions. Practical results show a gain of about 45% using this method.

Before concluding, lets compare hiding countermeasures against masking countermeasures. Side-channel attacks are only one class of attacks: what is thus the suitability of masking and hiding against the other attack strategies? The suitability of countermeasures to thwart attacks is given in Fig. 7.1.

This figure shows that masking is basically a countermeasure against non-invasive SCA. On the other hand, hiding is a countermeasure which was originally proposed to counter SCA but it was also found resistant against most fault injection attacks since the attacker erases the value stored redundantly in one pair of wires by changing only one of them. It is interesting to see that by associating masking and hiding, the protection extends to semi-invasive and invasive attacks. This association must be realized with care, since otherwise some attacks become possible, such as the “folding attack”proposed by Schaumont et al. [105] or the “subset attack” given by De Mulder et al. [106]. The synopsis of this attack consists in recovering the masking bit and then to defeat the hiding countermeasure. However, by using more than one bit of mask, these attacks become impossible.

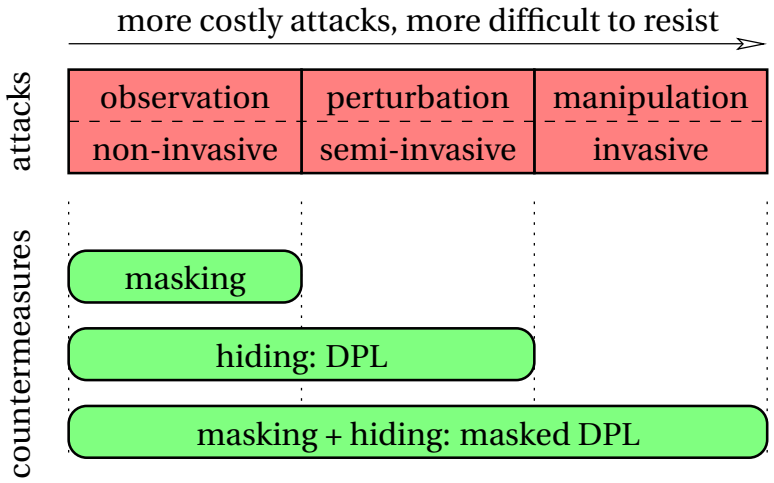


Figure 7.1: Coverage of countermeasures for all physical attacks classes.

7.2 Perspectives

A immediate perspective to this work would be to efficiently implement AES BCDL using t-boxes in new generation FPGA. True dual-port memory of size 32Kb or more are available on state of the art FPGA which can be fully used to implement a BCDL t-box (i.e. a true and a false t-box). Also, latest FPGA tools provide various options to constraint the routing. It would be interesting to check if these constraints can be used to precisely route every dual-rail net symmetrically. Analysis against FSA should also be done.

A second perspective for robust implementations of DPL on FPGA is a four step methodology:

1. Design DPL circuits with low fanout and automatic placement and routing.
2. Use advanced evaluation tools to find the imbalance bit.
3. Route them symmetrically using manual routing tools.
4. Iterate from step 2 until desired security achieved.

Another perspective is the implementation of masked BCDL (MBCDL). In fact, MBCDL can be implemented without complicating the gate structure. Path switching can be done by keeping the logic as unchanged just by switching memories. The value of the mask can be used to switch and select a memory block for a cycle. The mask used should not be one bit otherwise folding attacks are possible.

Finally, a perspective related to Unrolling is to test the robustness of partial or selective unrolling of a cryptographic algorithm. For example in AES, the first and the last rounds are targeted during an attack. In this case, unrolling the first and last two rounds of AES will make the attack difficult.

List of Publications

[A] SHIVAM BHASIN, NIDHAL SELMANE, SYLVAIN GUILLEY, AND JEAN-LUC DANGER. **Security Evaluation of Different AES Implementations Against Practical Setup Time Violation Attacks in FPGAs.** In HOST (Hardware Oriented Security and Trust), pages 15-21. IEEE Computer Society, July 27th 2009. San Francisco, CA, USA.

[B] NIDHAL SELMANE, SHIVAM BHASIN, SYLVAIN GUILLEY, TARIK GRABA, AND JEAN-LUC DANGER. **WDDL is Protected Against Setup Time Violation Attacks.** In FDTC, pages 73-83. IEEE Computer Society, September 6th 2009. Lausanne Switzerland.

[C] SHIVAM BHASIN, JEAN-LUC DANGER, FLORENT FLAMENT, TARIK GRABA, SYLVAIN GUILLEY, YVES MATHIEU, MAXIME NASSAR, LAURENT SAUVAGE, AND NIDHAL SELMANE. **Combined SCA and DFA Countermeasures Integrable in a FPGA Design Flow.** In ReConFig, pages 213-218. IEEE Computer Society, December 9-11 2009. Cancun, Mexico.

[D] NIDHAL SELMANE, SHIVAM BHASIN, SYLVAIN GUILLEY, AND JEAN-LUC DANGER. **Security Evaluation of ASICs and FPGAs against Setup Time Violation Attacks.** To Appear in IET Information Security Journal.

[E] SHIVAM BHASIN, SYLVAIN GUILLEY, LAURENT SAUVAGE, AND JEAN-LUC DANGER.**Unrolling Cryptographic Circuits: A Simple Countermeasure Against Side-Channel Attacks.** In RSA Cryptographers Track, CT-RSA, 5985 of LNCS, pages 195-207. Springer, March 1-5 2010. San Francisco, CA, USA.

[F] SHIVAM BHASIN, SYLVAIN GUILLEY, FLORENT FLAMENT, NIDHAL SELMANE, AND JEAN-LUC DANGER. **Countering Early Evaluation: An Approach Towards Robust Dual-Rail Precharge Logic.** In WESS, ACM, October 24-28 2010. Scottsdale, Arizona, USA.

[G] MAXIME NASSAR, SHIVAM BHASIN, JEAN-LUC DANGER, GUILLAUME DUC, AND SYLVAIN GUILLEY. **BCDL: A high performance balanced DPL with global precharge and without early-evaluation.** In DATE 10, pages 849-854. IEEE Computer Society, March 8-12 2010. Dresden, Germany.

[H] SHIVAM BHASIN, SYLVAIN GUILLEY, YOUSSEF SOUISSI, TARIK GRABA, AND JEAN-LUC DANGER. **Efficient Dual-Rail Implementations in FPGA using Block RAMs.** In ReConFig. IEEE Computer Society, Nov 30-Dec 2 2011. Cancun, Mexico.

[I] SHIVAM BHASIN, SYLVAIN GUILLEY, YOUSSEF SOUISSI, AND JEAN-LUC DANGER. **Efficient FPGA Implementations of Dual-Rail Countermeasures using Stochastic Models.** In NIAT, Sep 26-27 2011. Nara, Japan.

[J] YOUSSEF SOUISSI, SHIVAM BHASIN, SYLVAIN GUILLEY, MAXIME NASSAR, AND JEAN-LUC DANGER. **Towards Different Flavors of Combined Side-Channel Attacks.** In RSA Cryptographers Track, CT-RSA, LNCS, To Appear. Springer, Feb 27-March 2 2012. San Francisco, CA, USA.

[K] YOUSSEF SOUISSI, MAXIME NASSAR, SYLVAIN GUILLEY, SHIVAM BHASIN, AND JEAN-LUC DANGER. **Embedded Systems Security: An Evaluation Methodology Against Side Channel Attacks.** In DASIP. IEEE Computer Society, Nov 2-4 2011. Tampere, Finland.

[L] YOUSSEF SOUISSI, SYLVAIN GUILLEY, SHIVAM BHASIN, AND JEAN-LUC DANGER. **Common Framework to Evaluate Modern Embedded Systems against Side-Channel Attacks.** In HST. IEEE Computer Society, Nov 15-17 2011. Boston, USA.

[M] ZOUHA CHERIF, FLORENT FLAMENT, SHIVAM BHASIN, JEAN-LUC DANGER, SYLVAIN GUILLEY, AND HERVE CHABANNE. **Evaluation of white-box and grey-box Noekeon implementations in FPGA.** In ReConFig, pages 310-315. IEEE Computer Society, December 13-15 2010. Cancun, Mexico.

Bibliography

[1] PAUL C. KOCHER, JOSHUA JAFFE, AND BENJAMIN JUN. **Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems**. In *Proceedings of CRYPTO'96*, **1109** of *LNCS*, pages 104–113. Springer-Verlag, 1996. (PDF). xxi, 1, 19, 21

[2] PAUL C. KOCHER, JOSHUA JAFFE, AND BENJAMIN JUN. **Differential Power Analysis**. In *Proceedings of CRYPTO'99*, **1666** of *LNCS*, pages 388–397. Springer-Verlag, 1999. xxi, 1, 22

[3] ÉRIC BRIER, CHRISTOPHE CLAVIER, AND FRANCIS OLIVIER. **Correlation Power Analysis with a Leakage Model**. In *CHES*, **3156** of *LNCS*, pages 16–29. Springer, August 11–13 2004. Cambridge, MA, USA. xxi, 1, 22, 24, 33

[4] TELECOM PARISTECH SEN RESEARCH GROUP. **DPA Contest (2nd edition)**, 2009–2010. `http://www.DPAcontest.org/v2/`. xxi, 2

[5] DAN BONEH, RICHARD A. DEMILLO, AND RICHARD J. LIPTON. **On the Importance of Eliminating Errors in Cryptographic Computations**. *Journal of Cryptology*, **14**(2):101–119, 2001. xxi, 2

[6] ELI BIHAM AND ADI SHAMIR. **Differential Fault Analysis of Secret Key Cryptosystems**. In *CRYPTO*, **1294** of *LNCS*, pages 513–525. Springer, August 1997. Santa Barbara, California, USA. DOI: 10.1007/BFb0052259. xxi, 2, 17, 24

[7] STEFAN MANGARD, ELISABETH OSWALD, AND THOMAS POPP. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, `http://www.dpabook.org/`. xxii, 2

[8] KRIS TIRI AND INGRID VERBAUWHEDE. **A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation**. In *DATE'04*, pages 246–251. IEEE Computer Society, February 2004. Paris, France. DOI: 10.1109/DATE.2004.1268856. 3, 69, 90

[9] D. KAHN. *Seizing the Enigma: the race to break the German u-boat codes 1939-1943*. Barnes & Noble, 2009. 6

[10] NIST/ITL/CSD. **Data Encryption Standard. FIPS PUB 46-3**, Oct 1999. `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`. 6, 9

[11] WHITFIELD DIFFIE AND MARTIN EDWARD HELLMAN. **New Directions in Cryptography**. *IEEE Transactions on Information Theory*, **22**(6):644–654, November 1976. 6

[12] RONALD L. RIVEST, ADI SHAMIR, AND LEONARD M. ADLEMAN. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. *Commun. ACM*, **21**(2):120–126, 1978. 7, 13

[13] NIST/ITL/CSD. **Advanced Encryption Standard (AES). FIPS PUB 197**, Nov 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`. 7, 10

[14] VICTOR S. MILLER. **Use of Elliptic Curves in Cryptography**. In *CRYPTO*, **218** of *Lecture Notes in Computer Science*, pages 417–426. Springer, August 18-22 1985. Santa Barbara, California, USA. 7

[15] RAINER A. RUEPPEL. *Analysis and design of stream ciphers*. Springer-Verlag New York, Inc., New York, NY, USA, 1986. 8

[16] NIST/ITL/CSD. **Secure Hash Algorithm (SHA). FIPS PUB 180-2**, Nov 2001. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`. 10

[17] GADIEL SEROUSSI IAN BLAKE AND NIGEL SMART. *Elliptic Curves in Cryptography*, **265**. Cambrige University Press, 1999. 13

[18] HAGAI BAR-EL, HAMID CHOUKRI, DAVID NACCACHE, MICHAEL TUNSTAL, AND CLAIRE WHELAN. **The Sorcerer's Apprentice Guide to Fault Attacks**. *Proceedings of the IEEE*, **94**(2):370–382, 2006. DOI: 10.1109/JPROC.2005.862424. 15

[19] *Cryptographic smart cards*, **03**, 1996. 16

[20] *Robust protection against fault-injection attacks on smart cards implementing the advenced encryption stadard*, 2004. 16

[21] *Faults and side-channel attacks on pairing based cryptography*, 2004. 16

[22] *Design principles for tamper-resistant smartcard processors*, 1999. 16

[23] JEAN-JACQUES QUISQUATER AND DAVID SAMYDE. **Eddy current for Magnetic Analysis with Active Sensor**. In *Esmart 2002, Nice, France*, 9 2002. 16

[24] JOHANNES BLÖMER AND JEAN-PIERRE SEIFERT. **Fault based cryptanalysis of the Advanced Encryption Standard**. In SPRINGER, editor, *Financial Cryptography*, **2742** of *LNCS*, pages 162–181, 2003. 17

[25] CHRISTOPHE GIRAUD. **DFA on AES**. In SPRINGER, editor, *Advanced Encryption Standard (AES) 4th international conference, LNCS springer*, **3373** of *LNCS*, pages 27–41, May 2005. Bonn, Germany. 17

[26] GILLES PIRET AND JEAN-JACQUES QUISQUATER. **A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD**. In *CHES*, **2779** of *LNCS*, pages 77–88. Springer, September 2003. Cologne, Germany. 17, 24, 42, 45

[27] JUNKO TAKAHASHI AND TOSHINORI FUKUNAGA. **Differential Fault Analysis on AES with 192 and 256-Bit Keys.** *IACR Cryptology ePrint Archive*, pages 23–23, 2010. 17

[28] THANH-HA LE, CÉCILE CANOVAS, AND JESSY CLÉDIÈRE. **An overview of side channel analysis attacks**. In *ASIACCS*, pages 33–43. ASIAN ACM Symposium on Information, Computer and Communications Security, 2008. DOI: 10.1145/1368310.1368319. Tōkyō, Japan. 21

[29] BENEDIKT GIERLICHS, LEJLA BATINA, PIM TUYLS, AND BART PRENEEL. **Mutual Information Analysis**. In *CHES, 10th International Workshop*, **5154** of *Lecture Notes in Computer Science*, pages 426–442. Springer, August 10-13 2008. Washington, D.C., USA. 23, 63, 116, 149

[30] EMMANUEL PROUFF AND MATTHIEU RIVAIN. **Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis**. In SPRINGER, editor, *ACNS*, **5536** of *LNCS*, pages 499–518, June 2-5 2009. Paris-Rocquencourt, France. 24, 63

[31] PAUL C. KOCHER, JOSHUA JAFFE, AND BENJAMIN JUN. **Differential Power Analysis**. In *CRYPTO*, **1666** of *LNCS*, pages pp 388–397. Springer, 1999. 24

[32] GUIDO BERTONI, LUCA BREVEGLIERI, ISRAEL KOREN, AND PAOLO MAISTRI. **An Efficient Hardware-Based Fault Diagnosis Scheme for AES**. *proceedings of DFT*, **52**:130–138, 2004. 25

[33] G. BERTONI, L. BREVEGLIERI, I. KOREN, P. MAISTRI, AND V. PIURI. **Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard**. *IEEE Transactions on Computer-Aided Design*, **52**(4), April 2003. 25

[34] RAMESH KARRI, KAIJIE WU, PIYUSH MISHRA, AND YONGKOOK KIM. **Concurrent Error Detection Schemes for Fault Based Side-Channel Cryptanalysis of Symmetric Block Ciphers**. *IEEE Transactions on Computer-Aided Design*, **21**(12):1509–1517, december 2002. 26

[35] CHIH-HSU YEN AND BING-FEI WU. **Simple error detection methods for hardware implementation of Advanced Encryption Standard**. *IEEE transactions on computers*, **55**(6):720–731, june 2006. 26

[36] MARK KARPOVSKY, KONRAD J. KULIKOWSKI, AND ALEXANDER TAUBIN. **Differential Fault Analysis Attack Resistant Architectures For The Advanced Encryption Standard**. *DSN 2004*, (9), August 2004. 27

[37] MARK KARPOVSKY, KONRAD J. KULIKOWSKI, AND ALEXANDER TAUBIN. **Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard**. *IEEE Transactions on Computer-Aided Design*, **21**(2), may 2004. 27

[38] P. MAISTRI AND R. LEVEUGLE. **Double-Data-Rate Computation as a Countermeasure against Fault Analysis**. **57**, pages 1528 –1539, nov. 2008. 27

[39] STEFAN MANGARD. **Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness**. In *CT-RSA*, **2964** of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004. San Francisco, CA, USA. 29

[40] CHRISTOPHE CLAVIER, JEAN-SÉBASTIEN CORON, AND NORA DABBOUS. **Differential Power Analysis in the Presence of Hardware Countermeasures**. In *CHES*, LNCS, pages 252–263, London, UK, August 2000. Springer-Verlag. 29

[41] PAUL C. KOCHER. **Leak-resistant cryptographic indexed key update**, March 25 2003. United States Patent 6,539,092 filed on July 2nd, 1999 at San Francisco, CA, USA. 30

[42] G. EDWARD SUH AND SRINIVAS DEVADAS. **Physical Unclonable Functions for Device Authentication and Secret Key Generation**. In *DAC*, pages 9–14, 2007. 30

[43] MARCEL MEDWED, FRANÇOIS-XAVIER STANDAERT, JOHANN GROSSSCHÄDL, AND FRANCESCO REGAZZONI. **Fresh Re-Keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices**. In *AFRICACRYPT*, **6055** of *LNCS*, pages 279–296. Springer, May 03-06 2010. Stellenbosch, South Africa. DOI: 10.1007/978-3-642-12678-9_17. 30

[44] SURESH CHARI, CHARANJIT S. JUTLA, JOSYULA R. RAO, AND PANKAJ ROHATGI. **Towards Sound Approaches to Counteract Power-Analysis Attacks**. In *CRYPTO*, **1666** of *LNCS*. Springer, August 15-19 1999. Santa Barbara, CA, USA. ISBN: 3-540-66347-9. 30, 31

[45] LOUIS GOUBIN AND JACQUES PATARIN. **DES and Differential Power Analysis**. In *CHES*, LNCS, pages 158–172. Springer, Aug 1999. Worcester, MA, USA. 30

[46] THOMAS POPP AND STEFAN MANGARD. **Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints**. In *Proceedings of CHES'05*, **3659** of *LNCS*, pages 172–186. Springer, August 29 – September 1 2005. Edinburgh, Scotland, UK. 30, 92

[47] DAISUKE SUZUKI, MINORU SAEKI, AND TETSUYA ICHIKAWA. **Random Switching Logic: A Countermeasure against DPA based on Transition Probability**, 2004. `http://eprint.iacr.org/2004/346`. 30

[48] THOMAS S. MESSERGES. **Using second-Order Power Analysis to Attack DPA resistant Software**. In *CHES*, **1965** of *LNCS*, pages 71–77. Springer, August 17-18 2000. Worcester, MA, USA. 30

[49] MEHDI-LAURENT AKKAR AND CHRISTOPHE GIRAUD. **An Implementation of DES and AES Secure against Some Attacks**. In LNCS, editor, *Proceedings of CHES'01*, **2162** of *LNCS*, pages 309–318. Springer, May 2001. Paris, France. 30, 31

[50] ELISABETH OSWALD, STEFAN MANGARD, NORBERT PRAMSTALLER, AND VINCENT RIJMEN. **A Side-Channel Analysis Resistant Description of the AES S-box**. In LNCS, editor, *Proceedings of FSE'05*, **3557** of *LNCS*, pages 413–423. Springer, February 2005. Paris, France. 30, 32

[51] ÉRIC PEETERS, FRANÇOIS-XAVIER STANDAERT, NICOLAS DONCKERS, AND JEAN-JACQUES QUISQUATER. **Improved Higher-Order Side-Channel Attacks With FPGA Experiments**. In *CHES*, **3659** of *LNCS*, pages 309–323. Springer-Verlag, 2005. Edinburgh, UK. 30

[52] THOMAS S. MESSERGES. **Securing the AES Finalists Against Power Analysis Attacks**. In *Fast Software Encryption'00*, pages 150–164. Springer-Verlag, April 2000. New York. 31

[53] MARCO BUCCI, MICHELE GUGLIELMO, RAIMONDO LUZZI, AND ALESSANDRO TRIFILETTI. **A Power Consumption Randomization Countermeasure for DPA-Resistant Cryptographic Processors**. In *PATMOS*, **3254** of *LNCS*, pages 480–490. Springer, 2004. 31

[54] JOVAN DJ. GOLIC AND CHRISTOPHE TYMEN. **Multiplicative Masking and Power Analysis of AES**. In *CHES*, **2523** of *Lecture Notes in Computer Science*, pages 198–212. Springer, August 13-15 2002. San Francisco, USA. 32

[55] JOHANNES BLÖMER, JORGE GUAJARDO, AND VOLKER KRUMMEL. **Provably Secure Masking of AES**. In LNCS, editor, *Proceedings of SAC'04*, **3357**, pages 69–83. Springer, August 2004. Waterloo, Canada. 32

[56] FRANÇOIS-XAVIER STANDAERT, GAËL ROUVROY, AND JEAN-JACQUES QUISQUATER. **FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks**. In *proceedings of FPL 2006*. IEEE, August 2006. Madrid, Spain. 32

[57] RÉGIS BEVAN AND ERIK KNUDSEN. **Ways to Enhance Differential Power Analysis**. In *ICISC*, **2587** of *Lecture Notes in Computer Science*, pages 327–342. Springer, November 28-29 2002. Seoul, Korea. 33

[58] STEFAN MANGARD AND KAI SCHRAMM. **Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations**. In *CHES*, **4249** of *LNCS*, pages 76–90. Springer, October 10-13 2006. Yokohama, Japan. 33

[59] FRANÇOIS-XAVIER STANDAERT, SIDDIKA BERNA ÖRS, AND BART PRENEEL. **Power Analysis of an FPGA: Implementation of Rijndael: Is Pipelining a DPA Countermeasure?** In *CHES*, **3156** of *LNCS*, pages 30–44. Springer-Verlag, August 11-13 2004. Cambridge (Boston), MA, USA. 37

[60] JOHANNES WOLKERSTORFER, ELISABETH OSWALD, AND MARIO LAMBERGER. **An ASIC Implementation of the AES SBoxes**. In BART PRENEEL, editor, *CT-RSA*, **2271** of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002. 39

[61] FAROUK KHELIL, MOHAMED HAMDI, SYLVAIN GUILLEY, JEAN-LUC DANGER, AND NIDHAL SELMANE. **Fault Analysis Attack on an FPGA AES Implementation**. In *NTMS*, pages 1–5, Tangier, Morocco, nov 2008. IEEE. DOI: 10.1109/NTMS.2008.ECP.45. 41, 64, 78

[62] NIDHAL SELMANE. *Global and local Fault attacks on AES cryptoprocessor: Implementation and Countermeasures*. PhD thesis, Telecom ParisTech, December 2010. `http://www.iacr.org/phds/?p=detail&entry=565`. 41

[63] OLIVIER FAURAX, ASSIA TRIA, LAURENT FREUND, AND FRÉDÉRIC BANCEL. **Robustness of circuits under delay-induced faults: test of AES with the PAFI tool**. *IEEE International On-Line Testing Symposium*, pages 185–186, July 8-11 2007. Heraklion, Crete, Greece. 42

[64] RAJESH VELEGALATI AND JENS-PETER KAPS. **Techniques to enable the use of Block RAMs on FPGAs with Dynamic and Differential Logic**. In *International Conference on Electronics, Circuits, and Systems, ICECS 2010*, pages 1251–1254. IEEE, Dec 2010. 44, 91, 141

[65] CHONG HEE KIM AND JEAN-JACQUES QUISQUATER. **New Differential Fault Analysis on AES Key Schedule: Two Faults are Enough**. In *CARDIS 2008*, pages 48–60. Springer, 2008. Royal Holloway, University of London, UK. 45

[66] ALESSANDRO BARENGHI, GUIDO BERTONI, LUCA BREVEGLIERI, MAURO PELLICIOLI, AND GERARDO PELOSI. **Fault attack on AES with single-bit induced faults**. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 167 –172, aug. 2010. 45

[67] JULIEN FRANCQ AND OLIVIER FAURAX. **Security of several AES Implementations against Delay Faults**. In *Proceedings of the 12th Nordic Workshop on Secure IT Systems (NordSec 2007)*, October 2007. Reykjavík, Iceland. 49, 78

[68] SAAR DRIMER, TIM GÜNEYSU, AND CHRISTOF PAAR. **DSPs, BRAMs and a Pinch of Logic: New Recipes for the AES on FPGAs**. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 99–108. IEEE, 14-15 Apr 2008. Stanford, Palo Alto, CA. 49, 136

[69] KAYA ÇETIN KOÇ. *Cryptographic Engineering*. Springer US, 2009. 51

[70] SYLVAIN GUILLEY, SUMANTA CHAUDHURI, LAURENT SAUVAGE, JEAN-LUC DANGER, TAHA BEYROUTHY, AND LAURENT FESQUET. **Updates on the Potential of Clock-Less Logics to Strengthen Cryptographic Circuits against Side-Channel Attacks**. In *ICECS*, IEEE,

pages 351–354, December 13–16 2009. Medina, Yasmine Hammamet, Tunisia. DOI: 10.1109/ICECS.2009.5411008. 53

[71] SYLVAIN GUILLEY, PHILIPPE HOOGVORST, AND RENAUD PACALET. **A Fast Pipelined Multi-Mode DES Architecture Operating in IP Representation**. *Integration, The VLSI Journal*, **40**(4):479–489, July 2007. DOI: 10.1016/j.vlsi.2006.06.004. 53

[72] THOMAS ROCHE AND CÉDRIC TAVERNIER. **Multi-Linear cryptanalysis in Power Analysis Attacks: MLPA**. In *Western European Workshop on Research in Cryptology, WEWoRC 2009*, July 7-9 2009. Graz, Austria. 53

[73] MOULAY ABDELAZIZ EL AABID, SYLVAIN GUILLEY, AND PHILIPPE HOOGVORST. **Template Attacks with a Power Model**. Cryptology ePrint Archive, Report 2007/443, December 2007. `http://eprint.iacr.org/2007/443/`. 53

[74] SYLVAIN GUILLEY, PHILIPPE HOOGVORST, RENAUD PACALET, AND JOHANNES SCHMIDT. **Improving Side-Channel Attacks by Exploiting Substitution Boxes Properties**. In PRESSE UNIVERSITAIRE DE ROUEN ET DU HAVRE, editor, *BFCA*, pages 1–25, 2007. May 02–04, Paris, France, `http://www.liafa.jussieu.fr/bfca/books/BFCA07.pdf`. 61

[75] SYLVAIN GUILLEY, LAURENT SAUVAGE, JEAN-LUC DANGER, NIDHAL SELMANE, AND RENAUD PACALET. **Silicon-level solutions to counteract passive and active attacks**. In *FDTC, 5th Workshop on Fault Detection and Tolerance in Cryptography, IEEE-CS*, pages 3–17, Washington DC, USA, aug 2008. (Up-to-date version on HAL: `http://hal.archives-ouvertes.fr/hal-00311431/en/`). 61

[76] NICOLAS VEYRAT-CHARVILLON AND FRANÇOIS-XAVIER STANDAERT. **Mutual Information Analysis: How, When and Why?** In *CHES*, **5747** of *LNCS*, pages 429–443. Springer, September 6-9 2009. Lausanne, Switzerland. 63

[77] FRANÇOIS-XAVIER STANDAERT, ÉRIC PEETERS, GAËL ROUVROY, AND JEAN-JACQUES QUISQUATER. **An Overview of Power Analysis Attacks Against Field Programmable Gate Arrays**. *Proceedings of the IEEE*, **94**(2):383–394, February 2006. (Invited Paper). 63

[78] SURESH CHARI, JOSYULA R. RAO, AND PANKAJ ROHATGI. **Template Attacks**. In *CHES*, **2523** of *LNCS*, pages 13–28. Springer, August 2002. San Francisco Bay (Redwood City), USA. 63, 149, 150

[79] OLIVIER FAURAX, ASSIA TRIA, LAURENT FREUND, AND FRÉDÉRIC BANCEL. **Robustness of circuits under delay-induced faults: test of AES with the PAFI tool**. In *IOLTS*, pages 185–186. IEEE Computer Society, 8-11 July 2007. Heraklion, Crete, Greece. 64

[80] NIDHAL SELMANE, SYLVAIN GUILLEY, AND JEAN-LUC DANGER. **Setup Time Violation Attacks on AES**. In *EDCC, The seventh European Dependable Computing Conference*, pages 91–96, Kaunas, Lithuania, may 7-9 2008. ISBN: 978-0-7695-3138-0, DOI: 10.1109/EDCC-7.2008.11. 64, 78

[81] DAISUKE SUZUKI AND MINORU SAEKI. **Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style**. In *CHES*, **4249** of *LNCS*, pages 255–269. Springer, October 10-13 2006. Yokohama, Japan. `http://dx.doi.org/10.1007/11894063_21`. 67, 74, 114

[82] KONRAD J. KULIKOWSKI, MARK G. KARPOVSKY, AND ALEXANDER TAUBIN. **Power Attacks on Secure Hardware Based on Early Propagation of Data**. In *IOLTS*, pages 131–138. IEEE Computer Society, 2006. Como, Italy. 67, 74

[83] STEFAN MANGARD, NORBERT PRAMSTALLER, AND ELISABETH OSWALD. **Successfully Attacking Masked AES Hardware Implementations**. In LNCS, editor, *Proceedings of CHES'05*, **3659** of *LNCS*, pages 157–171. Springer, August 29 – September 1 2005. Edinburgh, Scotland, UK. 70

[84] SYLVAIN GUILLEY, LAURENT SAUVAGE, JEAN-LUC DANGER, TARIK GRABA, AND YVES MATHIEU. **Evaluation of Power-Constant Dual-Rail Logic as a Protection of Cryptographic Applications in FPGAs**. In *SSIRI*, pages 16–23, Yokohama, Japan, jul 2008. IEEE Computer Society. DOI: 10.1109/SSIRI.2008.31, `http://hal.archives-ouvertes.fr/hal-00259153/en/`. 70, 71, 87

[85] SYLVAIN GUILLEY, LAURENT SAUVAGE, JEAN-LUC DANGER, AND PHILIPPE HOOGVORST. **Area Optimization of Cryptographic Co-Processors Implemented in Dual-Rail with Precharge Positive Logic**. In *FPL (18th IEEE International Conference on Field-Programmable Logic and Applications)*, pages 161–166, Heidelberg, Germany, sep 2008. ISBN: 978-1-4244-1961-6. 71

[86] DAISUKE SUZUKI AND MINORU SAEKI. **An Analysis of Leakage Factors for Dual-Rail Pre-Charge Logic Style**. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E91-A**(1):184–192, 2008. DOI: 10.1093/ietfec/e91-a.1.184. 74

[87] KRIS TIRI, DAVIS HWANG, ALIREZA HODJAT, BO-CHENG LAI, SHENGLIN YANG, PATRICK SCHAUMONT, AND INGRID VERBAUWHEDE. **Prototype IC with WDDL and Differential Routing – DPA Resistance Assessment**. In LNCS, editor, *Proceedings of CHES'05*, **3659** of *LNCS*, pages 354–365. Springer, August 29 – September 1 2005. Edinburgh, Scotland, UK. 74

[88] LAURENT SAUVAGE, SYLVAIN GUILLEY, JEAN-LUC DANGER, YVES MATHIEU, AND MAXIME NASSAR. **Successful Attack on an FPGA-based WDDL DES Cryptoprocessor Without Place and Route Constraints**. In *DATE*, pages 640–645, Nice, France, apr 2009. IEEE Computer Society. 74, 149, 150

[89] LAURENT SAUVAGE, MAXIME NASSAR, SYLVAIN GUILLEY, FLORENT FLAMENT, JEAN-LUC DANGER, AND YVES MATHIEU. **Exploiting Dual-Output Programmable Blocks to Balance Secure Dual-Rail Logics**. *International Journal of Reconfigurable Computing*, page 12, 2010. DOI: 10.1155/2010/375245. 74

[90] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (`HTTP://WWW.IEEE.ORG/`). **IEEE Standard VHDL (Very High Speed Integrated Circuits Description Language) Reference Manual**, ISBN: 0-7381-3247-0 2002. 82

[91] CHRISTOPHE CLAVIER. *De la Sécurité des Cryptosystèmes Embarqués*. PhD thesis, (french). Université de Versailles Saint-Quentin-en-Yvelines, November 23 2007. 83

[92] VINCENT MAINGOT, JEAN-BAPTISTE FERRON, RÉGIS LEVEUGLE, VINCENT POUGET, AND ALEXANDRE DOUIN. **Configuration errors analysis in SRAM-based FPGAs: software tool and practical results**. *Microelectronics Reliability*, **47**(9-11):1836–1840, 2007. 83

[93] GABRIEL TORRENS, BARTOMEU ALORDA, S BARCELÓ, J ROSSELLÓ, SEBASTIA BOTA, AND JAUME SEGURA. **An SRAM SEU Hardening Technique for Multi-Vt Nanometric CMOS Technologies**. In *DCIS*, November 12–14 2008. ISBN: 978-2-84813-124-5, Grenoble, France. 84

[94] SYLVAIN GUILLEY, LAURENT SAUVAGE, FLORENT FLAMENT, PHILIPPE HOOGVORST, AND RENAUD PACALET. **Evaluation of Power-Constant Dual-Rail Logics Counter-Measures against DPA with Design-Time Security Metrics**. *IEEE Transactions on Computers*, **9**(59):1250–1263, September 2010. DOI: 10.1109/TC.2010.104. 86

[95] PENGYUAN YU AND PATRICK SCHAUMONT. **Secure FPGA circuits using controlled placement and routing**. In *CODES+ISSS'07: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 45–50, New York, NY, USA, 2007. ACM. 86

[96] SYLVAIN GUILLEY, SUMANTA CHAUDHURI, LAURENT SAUVAGE, TARIK GRABA, JEAN-LUC DANGER, PHILIPPE HOOGVORST, VINH-NGA VONG, AND MAXIME NASSAR. **Place-and-Route Impact on the Security of DPL Designs in FPGAs**. In *HOST (Hardware Oriented Security and Trust), IEEE*, pages 29–35, Anaheim, CA, USA, jun 2008. 87, 92

[97] TORU AKISHITA, MASANOBU KATAGI, YOSHIKAZU MIYATO, ASAMI MIZUNO, AND KYOJI SHIBUTANI. **A Practical DPA Countermeasure with BDD Architecture**. In *CARDIS*, **5189** of *Lecture Notes in Computer Science*, pages 206–217. Springer, Sept 2008. London, UK. 87

[98] KRIS TIRI AND INGRID VERBAUWHEDE. **Secure Logic Synthesis**. In *FPL*, **3203** of *LNCS*, pages 1052–1056. Springer, August 30 – September 1 2004. Leuven, Belgium. 87

[99] KARTHIK BADDAM AND MARK ZWOLINSKI. **Divided Backend Duplication Methodology for Balanced Dual Rail Routing**. In *CHES*, **5154** of *LNCS*, pages 396–410, Washington, DC, USA, aug 2008. Springer. DOI: 10.1007/978-3-540-85053-3_25. 87, 99

[100] ROBERT P. MCEVOY, COLIN C. MURPHY, WILLIAM P. MARNANE, AND MICHAEL TUNSTALL. **Isolated WDDL: A Hiding Countermeasure for Differential Power Analysis on FPGAs**. *ACM Trans. Reconfigurable Technol. Syst.*, **2**(1):1–23, 2009. 89, 99

[101] JENS-PETER KAPS AND RAJESH VELEGALATI. **DPA Resistant AES on FPGA Using Partial DDL**. In *FCCM: 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 273–280. IEEE Computer Society, May 02–May 04 2010. Charlotte, North Carolina, USA. DOI: 10.1109/FCCM.2010.49. 91

[102] KRIS TIRI AND INGRID VERBAUWHEDE. **Place and Route for Secure Standard Cell Design**. In KLUWER, editor, *Proceedings of WCC / CARDIS*, pages 143–158, Aug 2004. Toulouse, France. 92

[103] SYLVAIN GUILLEY, PHILIPPE HOOGVORST, YVES MATHIEU, AND RENAUD PACALET. **The "Backend Duplication" Method**. In *CHES*, **3659** of *LNCS*, pages 383–397. Springer, 2005. August 29th – September 1st, Edinburgh, Scotland, UK. 92

[104] KRIS TIRI AND PATRICK SCHAUMONT. **Changing the odds against Masked Logic**. In *13th Annual Workshop on Selected Areas in Cryptography*, **4356** of *LNCS*, pages 134–146. Springer, August 17 & 18 2006. Montreal, Canada. 92, 95

[105] PATRICK SCHAUMONT AND KRIS TIRI. **Masking and Dual Rail Logic Don't Add Up**. In *CHES*, **4727** of *LNCS*, pages 95–106. Springer, September 10-13 2007. Vienna, Austria. 92, 166

[106] ELKE DE MULDER, BENEDIKT GIERLICHS, BART PRENEEL, AND INGRID VERBAUWHEDE. **Practical DPA Attacks on MDPL**. In *First International Workshop on Information Forensics and Security (WIFS)*. IEEE Signal Processing Society, December 6-9 2009. London, United Kingdom. Also `http://eprint.iacr.org/2009/231`. 92, 166

[107] ZHIMIN CHEN AND YUJIE ZHOU. **Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage**. In *CHES*, **4249** of *LNCS*, pages 242–254. Springer, October 10-13 2006. Yokohama, Japan, http://dx.doi.org/10.1007/11894063_20. 92

[108] NEIL H.E. WESTE AND DAVID HARRIS. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, 2004. 3 edition (May 11, 2004), ISBN: 0321149017. 95

[109] MINORU SAEKI AND DAISUKE SUZUKI. **Security Evaluations of MRSL and DRSL Considering Signal Delays**. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E91-A**(1):176–183, 2008. DOI: 10.1093/ietfec/e91-a.1.176. 96

[110] RAFAEL SOARES, NEY CALAZANS, VICTOR LOMNÉ, PHILIPPE MAURINE, LIONEL TORRES, AND MICHEL ROBERT. **Evaluating the robustness of secure triple track logic through prototyping**. In *SBCCI'08: Proceedings of the 21st annual symposium on Integrated circuits and system design*, pages 193–198, New York, NY, USA, September 1-4 2008. ACM. 96

[111] JEAN-JACQUES QUISQUATER AND DAVID SAMYDE. **Radio Frequency Attacks**. In HENK C. A. VAN TILBORG, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005. 98, 99

[112] JEAN-LUC DANGER AND SYLVAIN GUILLEY. **Circuit de cryptographie programmable – Logique BCDL (Balanced Cell-based Differential Logic)**, 25 Mars 2008. Brevet Français FR08/51904, assigné à l'Institut TELECOM; WO/2009/118264. 125

[113] AKASHI SATOH. **Side-channel Attack Standard Evaluation Board, SASEBO**. Project of the AIST – RCIS (Research Center for Information Security), http://www.rcis.aist.go.jp/special/SASEBO/. 135

[114] IAN KUON AND JONATHAN ROSE. **Measuring the Gap Between FPGAs and ASICs**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**(2):203–215, February 2007. 136

[115] VIKTOR FISCHER AND MILOS DRUTAROVSKÝ. **Two Methods of Rijndael Implementation in Reconfigurable Hardware**. In ÇETIN KAYA KOÇ, DAVID NACCACHE, AND CHRISTOF PAAR, editors, *CHES*, **2162** of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2001. 137

[116] S. SHAH, R. VELEGALATI, J.-P. KAPS, AND D. HWANG. **Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs**. In *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, pages 274 –279, dec. 2010. 137

[117] AMIR MORADI, MAHMOUD SALMASIZADEH, AND MOHAMMAD T. MANZURI SHALMANI. **Power Analysis Attacks on MDPL and DRSL Implementations**. In *ICISC*, **4817** of *Lecture Notes in Computer Science*, pages 259–272. Springer, November 29-30 2007. Seoul, Korea. 149

[118] WERNER SCHINDLER, KERSTIN LEMKE, AND CHRISTOF PAAR. **A Stochastic Model for Differential Side Channel Cryptanalysis**. In LNCS, editor, *CHES*, **3659** of *LNCS*, pages 30–46. Springer, Sept 2005. Edinburgh, Scotland, UK. 149, 151

[119] FRANÇOIS-XAVIER STANDAERT, FRANÇOIS KOEUNE, AND WERNER SCHINDLER. **How to Compare Profiled Side-Channel Attacks?** In SPRINGER, editor, *ACNS*, **5536** of *LNCS*, pages 485–498, June 2-5 2009. Paris-Rocquencourt, France. 149

[120] BENEDIKT GIERLICHS, KERSTIN LEMKE-RUST, AND CHRISTOF PAAR. **Templates vs. Stochastic Methods**. In *CHES*, **4249** of *LNCS*, pages 15–29. Springer, October 10-13 2006. Yokohama, Japan. 149

[121] MICHAEL KASPER, WERNER SCHINDLER, AND MARC STÖTTINGER. **A stochastic method for security evaluation of cryptographic FPGA implementations**. In JINIAN BIAN, QIANG ZHOU, PETER ATHANAS, YAJUN HA, AND KANG ZHAO, editors, *FPT*, pages 146–153. IEEE, 2010. 155

[122] LAURENT SAUVAGE, SYLVAIN GUILLEY, AND YVES MATHIEU. **ElectroMagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack of a Cryptographic Module**. *ACM Trans. Reconfigurable Technol. Syst.*, **2**(1):1–24, March 2009. Full text in `http://hal.archives-ouvertes.fr/hal-00319164/en/`. 156, 157

[123] DAKSHI AGRAWAL, JOSYULA R. RAO, AND PANKAJ ROHATGI. **Multi-channel Attacks**. In *CHES*, **2779** of *LNCS*, pages 2–16. Springer, September 8-10 2003. Cologne, Germany. 157

[124] ALEKS JAKULIN AND IVAN BRATKO. **Analyzing Attribute Dependencies**. In *PKDD 2003, volume 2838 of LNAI*, pages 229–240. Springer-Verlag, 2003. 158

[125] BENEDIKT GIERLICHS, LEJLA BATINA, BART PRENEEL, AND INGRID VERBAUWHEDE. **Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis**. In *CT-RSA*, **5985** of *LNCS*, pages 221–234. Springer, March 1-5 2010. San Francisco, CA, USA. 158

[126] YANG LI, KAZUO SAKIYAMA, SHIGETO GOMISAWA, TOSHINORI FUKUNAGA, JUNKO TAKAHASHI, AND KAZUO OHTA. **Fault Sensitivity Analysis**. In *CHES*, **6225** of *Lecture Notes in Computer Science*, pages 320–334. Springer, August 17-20 2010. Santa Barbara, CA, USA. 165