UNIVERSITE CATHOLIQUE DE LOUVAIN

École Polytechnique de Louvain

Département d'Électricité

Synthèse de circuits logiques à ultra-basse consommation en technologie 65nm et régime sous-seuil

Application à un coprocesseur AES pour tag RFID intelligent

Promoteur : Professeur J.-D. Legat

Mémoire présenté en vue de l'obtention du grade d'ingénieur civil en électricité par

Hocquet Cédric

Louvain-la-Neuve

Année académique 2008-2009

Résumé

De nos jours, la technologie RFID est largement répandue. Les tags RFID *intelligents* doivent maintenant présenter des fonctionnalités de plus en plus avancées. Le chiffrement des données, permettant de sécuriser les transmissions qui se font sans-fil, est l'une de ces fonctionnalitées. Par ailleurs, le tag RFID est aussi fortement contraint en terme de consommation, en fonction de son type d'alimentation. L'objet de ce travail étant la synthèse de circuits logiques à très faible consommation en régime sous-seuil, l'étude est faite sur un coprocesseur AES, de manière à répondre aux besoins sus-cités. Dans une première approche, une analyse de l'inverseur de la bibliothèque de *STMicroelectronics* en 65nm est réalisée pour quantifier le comportement de celui-ci, en termes de délai et puissance principalement, à très faible V_{dd}. Le délai augmente d'un facteur 2000 lorsque V_{dd} diminue de 1.2V à 0.2V. La puissance est par contre réduite de six ordres de grandeur. Les effets de la variabilité sont aussi étudiés. À 0.2V, la variabilité gobale (process et température) implique une augmentation d'un facteur 24 du délai. Ensuite, pour synthétiser le circuit AES à très faible V_{dd}, une nouvelle bibliothèque de cellules est créée et caractérisée à différents faibles V_{dd}. À 100kHz, fréquence réaliste de fonctionnement du tag RFID compte tenu du protocole de communication, le coprocesseur AES est opérationnel jusqu'à une tension de 0.3V. Sa consommation vaut 31.3 nW. La variabilité globale (process et température) implique une augmentation de la tension minimum à 0.4V. Le consommation vaut alors 57.5 nW. La variabilité locale, implique une augmentation supplémentaire à 0.42V. Enfin, concernant la synthèse à très faible V_{dd}, deux résultats intérressants sont obtenus. Premièrement, alors qu'à la tension nominale (1.2V), l'outil de synthèse choisi uniquement les portes de plus petite taille pour réduire la consommation, cela n'est pas le cas à très faible V_{dd}. L'usage de portes de taille plus grande permet de diminuer la consommation en réduisant les courants de court-circuit grâce à la (dé)charge plus rapide des noeuds les plus conséquents. Cela permet un gain de puissance 17% sur le cirucit AES. Deuxièmement, la synthèse du circuit à l'aide d'une bibliothèque caractérisée à une très faible tension est plus optimale que celle réalisée à l'aide d'une bibliothèque caractérisée à la tension nominale, si le circuit doit être opéré à cette très faible tension. En effet, le tension minimum de fonctionnement du circuit synthétisé sera plus grande dans le deuxième cas, ce qui a pour conséquence d'augmenter la consommation minimale. Le gain en puissance est de 19% pour le coprocesseur AES.

Abstract

Nowadays, the RFID technology is widely used for a large range of applications. *Smart* RFID tags, exhibit an increasing need of features to be packed within the chip. The encryption of the data, enabling secure wireless transmission, can be one of those features. Moreover, RFID tags are strongly power or energy constrained circuits then presenting a fixed small power or energy budget, depending on the power supply. The aim of the present work beeing the assessment of the synthesis of low-power logical circuits with subthreshold logic, the study is done on an AES coprocessor, in order to address the aforesaid needs. As a first approach, the simulation of the *STMicroelectronics* 65nm library inverter is realized to quantify its evolution when V_{dd} is lowered in terms of delay and power consumption. The delay increases by a factor 20 when V_{dd} is lowered from 1.2V to 0.2V. The power consumption instead is reduced by six orders of magnitude. The effects of the variability are also addressed. At 0.2V, the global variability (process and temperature) induces an increase of the delay by a factor 24. Afterwards, to synthesize the AES circuit at low-V_{dd}, a new cells library is created and characterized at different low-V_{dd}. At 100kHz, which is a realistic operating frequency for RFID tags in consideration of the communication protocol, the coprocessor is operationnal right down to V_{dd} = 0.3V. Its power consumption is 31.3 nW. Global variability (still process and temperature) requires the minimal voltage to be rised to 0.4V. The power consumption is then 57.5 nW. Local variability requires an additional increase of V_{dd} to 0.42V. Finally, two interesting results regarding the logical synthesis at low-V_{dd} are reported. Firstly, while at nominal V_{dd} (1.2V) the synthesis tool chooses only the cells with the smallest driving strength in order to reduce the power consumption, this is not the case at low-V_{dd}. The use of larger cells enables the reduction of the power consumption by decreasing the short-circuit currents thanks to the faster (dis)charge of some larger nodes. A gain of power consumption of 17% is reached for the AES circuit. Secondly, the synthesis of the circuit with a library characterized at low-V_{dd} is more optimal than the synthesis of the circuit with a llibrary characterized at nominal V_{dd}, if the circuit is operated at that low-V_{dd}. Indeed, the minimum supply voltage of the synthesized circuit is higher in the second case, increasing this way the minimum power consumption achievable. There is a difference of 19% in power consumption for the AES coprocessor.

Table des matières

Remerciements

Avant d'entrer dans le vif du sujet, je tiens à remercier plusieurs personnes qui ont contribuées à la réalisation de ce travail. Tout d'abord, je pense à mon promoteur, le Professeur Jean-Didier Legat, qui m'a encadré tout au long de l'année et dont les conseils avisés m'ont beaucoup aidés pour l'accomplissement de ce travail. Je tiens à le remercier tout particulièrement pour m'avoir fait confiance et m'avoir permis de participer aux journées FTFC.

Ensuite, je me dois de remercier David Bol, qui a toujours été là pour répondre à mes nombreuses questions et qui n'a jamais hésité à me consacrer de son temps lorsque cela était nécessaire. Ses conseils judicieux et son souci du détail m'auront beaucoup appris.

Merci aussi à Dina Kamel pour ses relectures et autres conseils utiles ainsi qu'à Bertrand Rousseau pour m'avoir briefé sur les bibliothèques ST. Enfin, merci à Philippe Manet et Thibault Delavallée qui m'ont "hébergé" durant ce quadrimestre et m'ont encouragé par les nombreuses discussions, formelles ou non, que j'ai pu avoir avec eux.

Pour terminer, merci à François Gosset et François-Xavier Standaert pour m'avoir fourni le code VHDL du coprocesseur AES.

En espérant n'avoir oublié personne, il ne me reste plus qu'à vous souhaiter une bonne lecture.

Cédric.

. . . Through the exclusive use of these devices, the high speed switching circuits of the present invention dissipate appreciable power only during the switching transient. . .

Frank M. Wanlass, à propos de l'inverseur CMOS dans son brevet de 1967.

Chapitre 1

Introduction

De nos jours, la technologie RFID (Radio Frequency IDentification) est de plus en plus utilisée. Le tag RFID, composé d'un petit circuit intégré et d'une antenne, le tout encapsulé dans un boîtier souvent très compact, permet, de par sa petite taille, de fournir un moyen pratique d'identification partout où cela est nécéssaire, lorsque ces tags sont utilisés en combinaison à un lecteur adéquat. Ainsi, les exemples ne manquent pas, du forfait de remontées-mécaniques en station de ski aux plaques de voitures électroniques, en passant par des systèmes de traçage des bagages en aéroport ou de gestion de chaînes logistiques en entrepôt. Mais toutes ces applications, de plus en plus nombreuses, nécéssitent bien souvent des fonctionnalités avancées, que se doit de fournir le petit circuit intégré du tag. C'est la raison pour laquelle on peut parler de tag RFID *intelligent*. Une de ces fonctionnalités peut être, par exemple, la possibilité de prendre des mesures de température pour contrôler la chaîne du froid d'un produit surgelé. Comme la technologie RFID échange des données de manière sans-fils, une autre de ces fonctionnalités, très importante, est la possibilité de sécuriser les données transitant par le tag. En effet, dès lors qu'un tag RFID peut être utilisé pour avoir accès à des biens (batiment, voiture) ou encore peut contenir des informations confidentielles (passeport, plaque de voiture), il est de plus en plus indispensable que les données transitant depuis ou vers le tag soient chiffrées. Ce travail, en proposant d'étudier la synthèse d'un module AES [1], présente une telle solution de chiffrement.

De plus, alors que les dimensions du circuit intégré, et donc, en première approximation, sa consommation [2], augmente pour inclure ces nouvelles fonctionnalités, le bugdet de puissance ou d'énergie alloué au tag RFID, lui, n'évolue pas et reste limité. Ce budget est fonction du type d'alimentation qui est en général fournie par une batterie ou via un champ

1. Advanced Encryption Standard, voir Annexe A
2. Nous verrons qu'un circuit de plus petite taille n'implique pas nécéssairement une consommation plus faible et inversement.

électro-magétique. Dès lors, il est indispensable de réduire la consommation du circuit à tout prix. Une des solutions pour parvenir à cette fin consiste à faire fonctionner le circuit à la tension d'alimentation V_{dd} la plus faible possible. Cette tension minimum, lorsque les contraintes imposées au circuit le permettent, peut se situer en dessous de la tension de seuil V_t des transistors. C'est pourquoi, on parle de fonctionnement en régime sous-seuil [3]. C'est dans cette région sous-seuil que l'étude du module AES, proposée ici, est réalisée.

Ainsi, plus généralement, ce travail étudie la synthèse de circuits logiques en régime sous-seuil, le module AES servant de circuit "test" pour comparer les différentes synthèses. Les objectifs fixés à priori pour ce travail ont servi de fil conducteur pour mener à bien l'étude. Ils peuvent être divisés en deux groupes : d'une part, ceux ayant trait à la synthèse logique en générale :

- Déterminer si l'utilisation d'une bibliothèque caractérisée à la tension cible de fonctionnenemt du circuit est utile pour la synthèse de celui-ci. Autrement dit, il s'agit de comparer, en terme de puissance, les résultats de la synthèse du circuit avec une bibliothèque caractérisée à tension V_{dd} nominale à ceux de la synthèse du circuit avec une bibliothèque caractérisée à faible tension V_{dd}, lorsque le circuit est opéré à cette même faible tension V_{dd}. Il s'agit là de l'objectif principal de ce mémoire.
- Observer l'influence de la taille des cellules que contient la bibliothèque sur la synthèse d'un circuit à faible V_{dd}.

et d'autre part, ceux propres au module AES faible consommation :

- Déterminer la tension V_{dd} minimum du module AES. Cette tension est determinée en choisissant d'abord une contrainte temporelle réaliste pour le circuit.
- Estimer la consommation du module AES à cette tension V_{dd} minimum.

Le point de départ de ce travail étant la biliothèque 65nm de *STMicroelectronics* ainsi que le code VHDL du coprocesseur AES, il a fallu passer par différentes étapes pour mener à bien le travail et parvenir à réaliser les objectifs. Détaillons ces différentes étapes. Tout d'abord, il a fallu procéder à l'étude de l'évolution d'une cellule simple de la bibliothèque 65nm lors de la réduction de V_{dd}. Dans un premier temps, les résultats fournis par la simulation de cette cellule simple à l'aide d'un outil de simulation de circuit analogique classique [4] sont comparés aux valeurs trouvées dans la bibliothèque à tension nominale, en guise de vérification. Ensuite, il s'agit de simuler la cellule pour observer et quantifier l'évolution du délai, de la puissance et de l'énergie consommée lorsque V_{dd} diminue. La variabilité est aussi étudiée. La deuxième étape a été le développement d'une méthode de caractérisation. La caractérisation

3. Remarquons ici que le but premier est bien de réduire la consommation du circuit et non pas de le faire fonctionner systématiquement en régime sous-seuil, même si cela sera bien souvent le cas.

4. ELDO

manuelle des cellules à l'aide du même outil de simulation s'est avérée compliqué, dû au grand nombre de paramètres à caractériser. Il a donc fallu trouver une solution réaliste pour caractériser une bibliothèque de cellules complète. C'est finalement l'outil Liberty NCX de Synopsys qui à été utilisé. L'apprentissage de ce dernier fût l'une des étapes majeures de ce travail [5]. La caractérisation à proprement parler des bibliothèques basse-tensions a pû ensuite être réalisée. La méthode de caractérisation étant au point, il s'agit de choisir les cellules qui constitueront la bibliothèque, de les caractériser et de compiler la bibliothèque, cela pour différentes tensions V_{dd}. La dernière étape peut alors être entamée. Il s'agit de la synthèse du module AES avec les différentes bibliothèques recaractérisées. Il est maintenant possible d'observer l'évolution du module AES lorsque V_{dd} diminue. À partir de ces résultats, les conclusions peuvent être tirées quant aux objectifs posés. Il est bon d'indiquer que, pendant toutes ces différentes étapes, de nombreuses publications scientifiques ont dû être lues, analysées et critiquées afin de mieux comprendre la problèmatique étudiée. Enfin, ce travail à également été l'objet d'un papier soumis à l'occasion des huitièmes journées d'études Faible Tension Faible Consommation 2009 [6].

En guise d'avant-goût, nous pouvons d'ores et déjà révéler ici les principaux résultats :

- La simulation de l'inverseur nous montre une augmentation du délai d'un facteur 2000 lorsque V_{dd} passe de 1.2V à 0.2V. La puissance, grâce à la réduction de la fréquence, diminue quant à elle de six ordres de grandeur. La variabilité globale (process et température) provoque une augmentation du délai d'un facteur 24 à 0.2V.
- A faible V_{dd}, la présence de cellules plus grandes dans la bibliothèque permet de diminuer la consommation interne grâce à la réduction des courants de court-circuit dûe à une (dé)charge plus rapide des noeuds. Un gain de l'ordre de 17% est observable. Cela n'est pas valable à la tension nominale.
- La synthèse à l'aide d'une bibliothèque caractérisée à la tension cible de fonctionnement du circuit est effectivement plus efficace et permet de réaliser un gain de l'ordre de 19%. Ce gain provient du fait que le circuit synthétisé à l'aide de la bibliothèque caractérisée à la tension nominale possède un V_{min} supérieur à celui du circuit synthétisé avec la bibliothèque caractérisée à faible V_{dd}.
- Le coprocesseur AES peut fonctionner à une tension d'alimentation de 0.3V avec une fréquence de 100kHz. Sa consommation est de 31.3 nW. En considérant la variabilité globale, il faut monter la tension à 0.4V pour que le circuit reste fonctionnel. Sa consommation augmente à 57.5 nW. La prise en compte de la variabilité locale implique une seconde augmentation de V_{dd} jusqu'à 0.42V.

5. En terme de temps y consacré.
6. Le papier est disponible en Annexe C.

Après ce premier chapitre d'introduction, le second propose un petit tour d'horizon des différents coprocesseurs AES réalisés jusqu'à présent et présentant une faible consommation. Dans le but de faire apparaître plus clairement le contexte dans lequel ce travail a été réalisé, une introduction à la logique sous-seuil et quelques-unes de ses applications récentes sont ensuite exposés. Le début du troisième chapitre est consacré à la présentation des bibliothèques digitales et à la manière dont les différents paramètres des cellules sont modèlisés au sein de celles-ci. Ensuite, la modélisation classique du délai et de la puissance est exposée. La suite du chapitre est consacré à l'étude d'un inverseur de la bibliothèque 65nm, lorsque la tension d'alimentation est diminuée. Plus particulièrement, l'évolution du délai, de la puissance et de l'énergie pas transition est présentée. Le comportement statique de l'inverseur est également montré ainsi que l'effet, négatif, de la variabilité globale et locale, amplifié à faible V_{dd}. Le quatrième chapitre présente les résultats de la synthèse du coprocesseur AES à l'aide des bibliothèques recaractérisées à faible V_{dd}. Les bibliothèques recaractérisées sont d'abord présentées, ensuite des réponses aux questions soulevées dans cette introduction sont proposées, principalement concerant l'utilité de la bibliothèque recaractérisée à faible V_{dd} ainsi que l'effet de la taille de cellules contenues dans les bibliothèques sur la synthèse. Finalement, le cinquième et dernier chapitre conclut le travail.

Chapitre 2

État de l'art

Cette section présente le contexte dans lequel ce travail a été réalisé afin d'en dégager certaines de ses motivations. Cela sera fait en relatant les travaux précédement effectués dans les domaines d'intérêt. La première section de ce chapitre détaille l'avancée des recherches scientifiques réalisées autour de l'implémentation basse consommation d'un coprocesseur AES. Après une rapide présentation de l'architecture générale d'un tel coprocesseur et de l'implémentation basique de ses différentes opérations, les travaux ayant pour objet l'optimisation de la S-Box suivi de ceux concernant l'optimisation du coprocesseur AES dans son ensemble sont exposés. La deuxième partie concerne de manière plus générale la logique sous-seuil. Une introduction à cette logique est d'abord proposée, ensuite quelques résultats récents de circuits utilisant cette logique sont détaillés.

2.1 Advanced Encryption Standard

2.1.1 Introduction

Depuis l'adoption de l'algorithme *Rijndael* comme standard de chiffrement pour l'AES en 2000, et tel que présenté en Annexe A, bon nombre d'implémentations matérielles ont fait leur apparition. Comme pour tout circuit digital, deux grandes approches peuvent être distinguées. Il s'agit de la conception haute performance (en terme de vitesse) et de la conception faible surface/faible consommation. Dans le cadre de ce travail, la conception haute performance ne sera pas abordée, si ce n'est un survol rapide de son architecture générale. Le lecteur curieux peut se référer à [4], [1] et [5] pour les meilleurs résultats concernant la haute performance. Nous ne nous intéresserons donc qu'à la conception faible surface/faible consommation.
Une autre subdivision peut être également faite lorsque que l'on parle d'implémentation ma-

térielle. Celle-ci peut se faire à l'aide d'un circuit logique programmable (FPGA) ou bien directement comme circuit intégré spécialisé (ASIC). Ici, seule cette deuxième solution sera couverte.

La section suivante détaille l'architecture générale des coprocesseurs AES. Ensuite, les optimisations au niveau de la S-Box puis des coprocesseurs AES complets seront abordées au travers des différents travaux réalisés.

2.1.2 Architecture

2.1.2.1 Architecture générale

L'implémentation de base présente une architecture itérative dans laquelle se trouve du matériel pour un seul *round*[1] à la fois. La sortie est routée vers l'entrée du bloc pour réaliser les différentes itérations de la boucle. Cela est présenté à la Figure 1. Cette architecture basique ne permettra pas de réaliser un circuit haute performance à cause du grand nombre de cycles requis. On verra par la suite que ce sera par contre une architecture intéressante pour réduire la consommation.

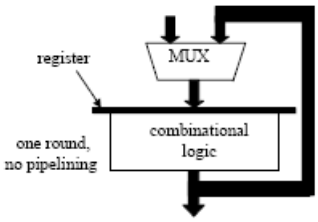


FIGURE 1: Architecture itérative de base (source : [6])

Différentes techniques peuvent être utilisées pour améliorer cette première architecture :

- Déroulage : L'architecture déroulée possède du matériel pour éxécuter plus d'un *round* à la fois. La boucle peut être partiellement ou complètement déroulée selon le nombre d'instance de *round*. Une architecture entièrement déroulée est présentée à la Figure 2. Le déroulage seul ne va pas augmenter la performance du circuit. Cependant, cela va permettre le pipelining.
- Pipelining externe : Le pipelining externe consiste à insérer des registres entre les blocs de l'architecture déroulée (entre les *rounds*). Cela augmente grandement les performances en terme de débit car des blocs de données différents peuvent maintenant être traités simultanément. Cela est aussi visible Figure 2.

1. *round* désigne l'ensemble des quatres opérations (*SubBytes*, *ShiftRows*, *MixColumns* et *AddRoundKey*) appliquées à l'*État* durant le (dé)chiffrement.

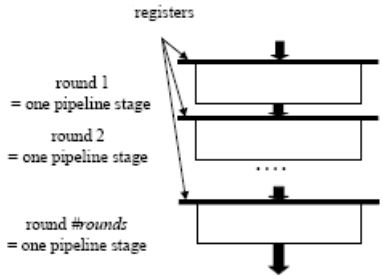


FIGURE 2: Architecture déroulée avec pipelining externe (source : [6])

– Pipelining interne : Pour réaliser du pipelining interne, des registres doivent être insérés à l'intérieur des blocs de chaque *round*. Cette architecture permet d'obtenir un débit encore plus élevé. Une architecture entièrement déroulée avec pipeline externe et interne est présenté Figure 3.

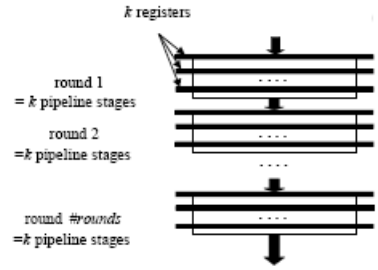


FIGURE 3: Architecture entièrement déroulée avec pipelining externe et interne (source :[6])

2.1.2.2 Implémentations des différentes opérations

– *SubByte* : L'opération *SubByte* est originellement implémentée via une table de correspondance (*Look Up Table*). Cette table de correspondance peut être dans une RAM dédiée ou encore elle peut être générée par l'outil de synthèse à l'aide de portes combinatoires. On verra par la suite que les tables de correspondance, non optimisées pour une faible surface ou consommation, seront vite remplacées par d'autres solutions.
– *ShiftRow* : Cette opération consiste simplement en une réorganisation des bytes de l'*État* entre eux. Cela sera fait par un routage physique ou via un adressage adéquat de la mémoire stockant l'*État*.
– *MixColumns* : L'opération *MixColumns* est implémentée à l'aide de portes combinatoires directement selon sa définition Equation A.3. Cependant, cette définition sera souvent réécrite pour obtenir un circuit plus compact.
– *AddRoundKey* : L'opération *AddRoundKey* consiste en une simple addition XOR.

– *KeyScheduling*[2] : La génération des clés succesives pour chaque *round* peut être vue de deux manières différentes. Soit les clés sont calculées à l'avance et stockées dans une mémoire à cet effet, soit celles-ci sont calculées au fur et à mesure en parallèle à l'exécution des *rounds*[3]. La première solution nécessitant une grande mémoire et donc une grande superficie et consommation, la deuxième approche sera utilisée pour les implémentations ciblant la faible consommation.

2.1.3 Implémentation faible consommation

2.1.3.1 S-Box

La S-Box, utilisée par l'opération *SubByte* et le *KeyScheduling*, étant la partie la plus coûteuse en terme de délai, superficie et consommation (75% de la consommation totale selon [14]), plusieurs auteurs se sont donc interessés uniquement à l'optimisation de celle-ci.
La table de correspondance est sans doute la solution la plus simple à mettre en oeuvre mais présente deux inconvénients majeurs. Premièrement, elle utilise une grande superficie. Deuxièmement, le délai de la table de correspondance est bien souvent le facteur limitant d'une architecture avec pipeline interne car il s'agit du délai le plus important à l'intérieur d'un *round*. Dans le cadre d'une implémentation faible consommation, le pipelining ne sera pas souvent utilisé mais le délai de la S-Box restera tout de même le plus important, toutes architectures confondues, et déterminera donc en grande partie la fréquence de fonctionnement du circuit.

Une alternative à la table de correspondance est l'utilisation de la logique combinatoire seule grâce à la théorie des corps finis (corps de Galois). Cela a été suggéré pour la première fois par V. Rijmen, l'un des auteurs de l'AES, dans [7] mais sans aucune implémentation. A. Satoh *et al.* présentent une première implémentation dans [8]. La méthode est la suivante : les éléments du corps $GF(2^8)$ (que sont les bytes) sont succesivement mappés vers des éléments des corps homomorphiques $GF((2^4)^2)$ et ensuite $GF(((2^2)^2)^2)$, où les opérations sont éffectuées facilement à l'aide de logique combinatoire simple.
La S-Box peut être divisée en trois parties : d'abord une transformation du corps $GF(2^8)$ vers le corps $GF(((2^2)^2)^2)$, ensuite une inversion dans le corps $GF(((2^2)^2)^2)$ et enfin, la transformation inverse du corps $GF(((2^2)^2)^2)$ vers le corps $GF(2^8)$ combinée avec la transformation affine. Cela est représenté Figure 4. L'inversion dans le corps $GF(((2^2)^2)^2)$ est visible Figure 5. Le corps $GF(((2^2)^2)^2)$ est construit en appliquant de multiples extensions de degré 2 au corps $GF(2)$. Chacune de ces extensions requiert un polynôme irréductible servant de base. Il y a plusieurs possibilités pour le choix de ces polynômes, et cela déterminera

2. *KeyScheduling* désigne le mécanisme fournissant les clés nécessaires à l'exécution de l'opération *AddRoundKey* pour chaque *round*.

3. Dans la suite de ce travail, on désignera par *on-the-fly* cette deuxième méthode.

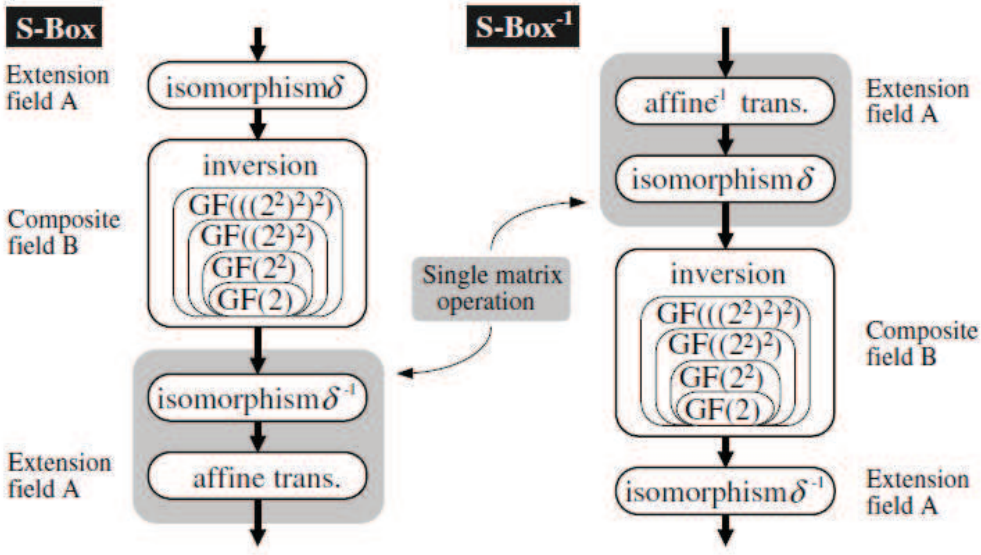


FIGURE 4: Structure de la S-Box de Satoh *et al.* (source : [8])

en partie la complexité du circuit. Ainsi, dans [8], les polynômes choisis sont :

$$\begin{cases} GF(2^2) & : x^2+x+1 \\ GF((2^2)^2) & : x^2+x+\phi \\ GF(((2^2)^2)^2) & : x^2+x+\lambda, \end{cases} \tag{2.1}$$

avec $\phi = \{10\}$ and $\lambda = \{1100\}$. A partir de ces bases, il est possible de créer une matrice pour effectuer la transformation (isomorphisme) qui sera ensuite implémentée en logique combinatoire [18].
Le nombre de portes équivalentes obtenu pour la S-Box lors de la synthèse en technologie CMOS 0.11μm est de 294 et le délai est de 3.69 ns (ou 286 portes équivalentes en technologie CMOS 0.18μm, tel que calculé par Mentens dans [10], voir plus bas). Selon leur calcul, cela représente une taille 4 fois plus petite qu'avec une table de correspondance.

Wolkerstorfer *et al.* ont également réalisé dans [9] l'implémentation d'une S-Box en utilisant la théorie des corps finis. Cependant, ils se sont limités à représenter les éléments du corps $GF(2^8)$ en une extension du corps $GF(2^4)$, comme proposé initialement par Rijmen dans [7]. Leur implémentation, qui combine la S-Box et la S-Box inverse (pour le processus de déchiffrement) contient 406 portes équivalentes pour une superficie de 0.108mm^2 et une fréquence de 70MHz en technologie CMOS 0.6 μm d'*AMS*. Pour une S-Box implémentée à l'aide d'une table de correspondance, ils obtiennent une superficie de 0.200mm^2 avec la même technologie.

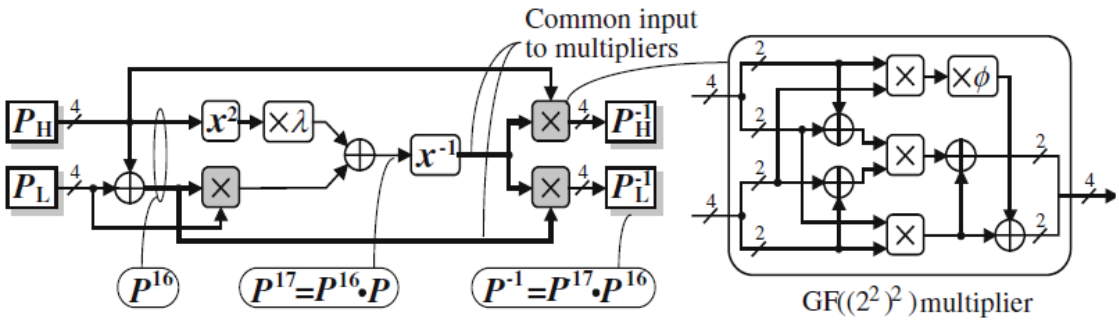


FIGURE 5: Inversion dans le corps $GF(((2^2)^2)^2)$ (source : [8])

Macchetti *et al.* présentent dans [12] une autre implémentation basée sur la décomposition en sous-corps. Ils se limitent également à une décomposition dans $GF(2^4)$ mais en utilisant une représentation comme décrite par Rudra *et al.* dans [13]. La synthèse de leur circuit en technologie CMOS 0.25μm 1.8V de chez *STMicroelectronics* donne une superficie de 0.075mm^2 pour un délai de 7.94ns. Les auteurs donnent également ici des informations concernant la puissance. Ils ont synthétisé un circuit de *round* complet en utilisant leur S-Box. La consommation du circuit est de 2mW pour un délai de 12 ns. On peut remarquer ici que le délai de la S-Box intervient pour 66% du délai total.

Dans [10], Mentens *et al.* améliorent le travail de Satoh en étudiant de manière exhaustive le choix des polynômes de base. Ils montrent qu'un meilleur choix peut amener à une réduction de superficie de 5% par rapport à celui de Satoh. En particulier, en choisissant $\lambda = \{1000\}$, la S-Box contient maintenant 272 portes équivalentes pour une synhtèse en technologie CMOS 0.18μm.

Dans [11], Canright va un cran plus loin en incluant les bases normales en plus des bases polynomiales lors du choix des bases pour les extensions. De plus, il choisit correctement les portes utilisées (en en remplaçant certaines par des NOR qui sont plus petites dans la bibliothèque CMOS 0.13μm utilisée) et il utilise une meilleure méthode (algorithme de recherche par arbre) pour optimiser les isomorphismes obtenus (éliminer la redondance). Cela lui permet de diminuer la superficie de 20% par rapport à Satoh [8].

En 2003, Morioka et Satoh [14] présenteront pour la première fois une implémention de la S-Box optimisée spécifiquement en vue d'une faible consommation. Ils remarquent d'abord que la taille du circuit à moins d'effet que prévu sur la consommation. Particulièrement, ils montrent qu'une implémentation directe de la relation entrée-sortie via une structure type *somme des produits* consomme moins qu'une implémentation à l'aide des corps composés, pourtant de plus faible superficie. Cela est expliqué par des transitions non-désirées (1) à la suite des différences entre les temps d'arrivée des signaux à l'entrée d'un module

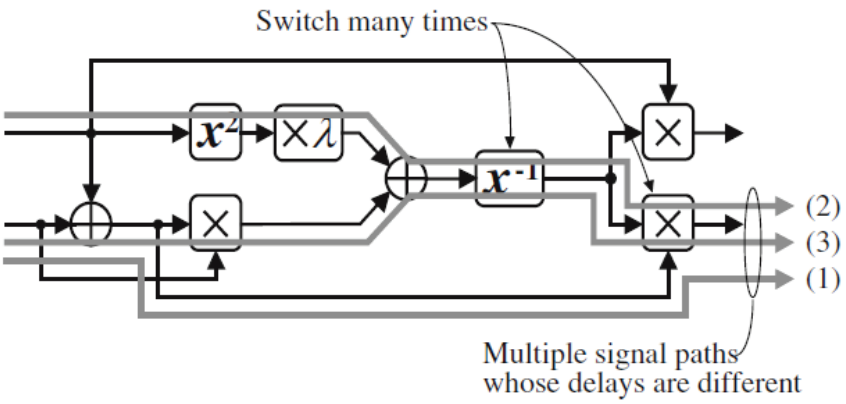


FIGURE 6: Différence entre les temps d'arrivée des signaux dans la structure présentée Figure 5 (source : [14])

(Figure 6) et (2) à l'utilisation de beaucoup de portes XOR qui ont une probabilité de propagation des transitions valant 1 au lieu de 0.5 pour les autres portes. L'implémentation de la S-Box qu'ils proposent est donc optimisée de manière à réduire ces transitions indésirables. Pour ce faire, ils se base sur la structure de décomposition en sous-corps finis déjà développée par Satoh dans [8]. Ils divisent cette structure en trois étages, chacun implémenté via un réseau de type *PPRM* (Positive Polarity Reed-Muller). Les temps d'arrivée des signaux entre les étages sont égalisés par des chaînes de délai quand cela est nécessaire. Synthétisé en technologie CMOS 0.13μm 1.5V, leur S-Box consomme 29μW à une fréquence de 10MHz. Une version combinée des S-Box directe et inverse consomme en moyenne 74.5μW pour un délai de 2ns et un nombre de portes de 725, toujours dans cette même technologie.

Un deuxième travail sur l'implémentaion faible consommation de la S-Box a été réalisé par Bertoni et Macchetti dans [15]. Leur contribution porte sur l'optimisation au niveau logique de la la fonction non-linéaire qu'est la S-Box. La méthode utilisée est la suivante : la S-Box, à N bits d'entrée et N bits de sortie, est vue comme une fonction de permutation sur un ensemble S, un isomorphisme. Si les éléments de S sont représentés par un codage de type *one-hot*, le nombre de bits d'entrée et de sortie va grandir à 2^N mais l'isomorphisme sur S ne sera plus qu'un simple réarrangement des bits de l'entrée vers la sortie, donc ne consommant aucune puissance. Un décodeur placé à l'entrée permet de réaliser le codage *one-hot* (de N bits vers 2^N bits) et un encodeur à la sortie permet de retrouver le codage initial à la sortie (de 2^N bits vers N bits).
Par souci de confidentialité, les auteurs ne dévoilent pas leurs résultats absolus. Par contre, avec une synthèse en technologie CMOS 0.18μm 1.8V de chez *STMicroelectronics*, ils rapportent une réduction de puissance de 57% par rapport à une S-Box traditionelle implémentée comme une table de correspodance pour une augmentation de superficie de 11%, cela à une fréquence de 1GHz. Ils rapportent également un meilleur produit puissance-délai par

rapport aux autres implémentations précédement réalisées, y compris celle de Morioka et Satoh, mais sans donner de chiffres.

Tillich *et al.* ont réalisé en 2006 une comparaison des différentes S-Box existantes en terme de superficie, délai et puissance [16]. Pour ce faire, ils ont re-synthétisé les différentes S-Box en technologie CMOS 0.35μm de chez *AMS*. Les résultats sont :

1. la plus petite superficie est obtenue avec la S-Box de Canright [11] suivi de près par celle de Wolkerstorfer *et al.* [9] et celle de Satoh *et al.* [8]. Il s'agit des S-Box utilisant l'arithmétique des corps finis.
2. la plus faible puissance revient à la S-Box de Bertoni *et al.* [15]. Il est intérressant de remarquer que les S-Box qui consomment le plus sont celles de plus petite superficie sus-citées.
3. une dernière mesure reportée, fort utile dans le cas de circuits tels que les tags RFID, est le produit puissance-superficie. Les résultats sont les mêmes que ceux pour la puissance.

Il est à noter que la S-Box de Morioka *et al.* et celle de Mentes *et al.* ne faisaient pas partie de la comparaison.

Enfin, le travail le plus récent [17] révèle une autre approche. Alors que tous les précédents travaux présentent des optimisations au niveau logique, ici c'est au niveau du transistor que l'étude est faite. Reprenant la S-Box de Mentes *et al.* pour sa faible taille, les auteurs étudient l'évolution de la consommation de la S-Box en fonction de différents paramètres technologiques (tension de seuil et taille des transistors) et fonctionnels (fréquence et tension d'alimentation). Il y est expliqué comment le concepteur peut réduire la consommation du circuit en choissant de manière adéquate la technologie. Ainsi, avec une technologie CMOS 65nm 1.2V LP SVT, la consommation de la S-Box est réduite à 90nW à 100kHz. De plus, si la tension V_{dd} est réduite à 0.8V, une réduction de 60% supplémentaire est possible pour atteindre 37nW. Il est intérressant de remarquer que ces choix technologiques, proposés dans ce travail, modifient principalement la consommation statique du circuit et non la consommation dynamique, qui est la principale cible de tous les autres travaux.

Le lecteur interressé peut se référer à [18] pour plus de détails concernant l'arithmétique des corps finis et son application aux circuits logiques.

2.1.3.2 Co-processeur AES complet

Les premières implémentations matérielles de l'algorithme *Rijndael* ont été faites durant la deuxième étape de sélection pour les différents candidats à l'AES. Ainsi, dans [19] l'algorithme Rijndael a été synthétisé dans une technologie CMOS 0.5μm propre à la *National*

Security Agency. Les résultats sont fournis pour une version itérative et une version avec pipeline externe intégral. Il s'agit d'une synthèse basique sans aucune optimisation spécifique. Les S-Box sont implémentées par des tables de correspondance. La version itérative du circuit présente une superficie de 46 mm^2 ($\sim$ 1000000 portes) et un débit de 443Mbits/s. La version pipelinée a une superficie de 471 mm^2 ($\sim$ 7000000 portes) et un débit de 5.163Gbits/s. Dans [20], la *National Institute of Standards and Technology* a fait le même travail en technologie CMOS 0.35μm de chez *Mitsubishi Electric*. Ici, il s'agit d'une architecture entièrement déroulée mais sans pipeline. Le circuit présente une superficie de 612000 portes et un débit de 1.95Gbits/s.

Parmi les premiers à s'être attelés à l'optimisation du circuit, on peut citer le travail de Kuo *et al.* dans [21]. Partant du fait que l'algorithme est fortement symétrique comme le montre la Figure 7, les auteurs ont minimisé la superficie et le chemin critique par un design rigoureux. Pour balancer correctement performance et superficie, ils choisisent une architecture basique itérative. Un *round* est éxécuté en un cycle d'horloge. Aucun déroulement ni pipeline n'est utilisé. Les clefs succesives sont générées *on-the-fly*. Le chemin de données interne a une largeur de 256 bits. Seul le chiffrement est implémenté ici. Les choix pour

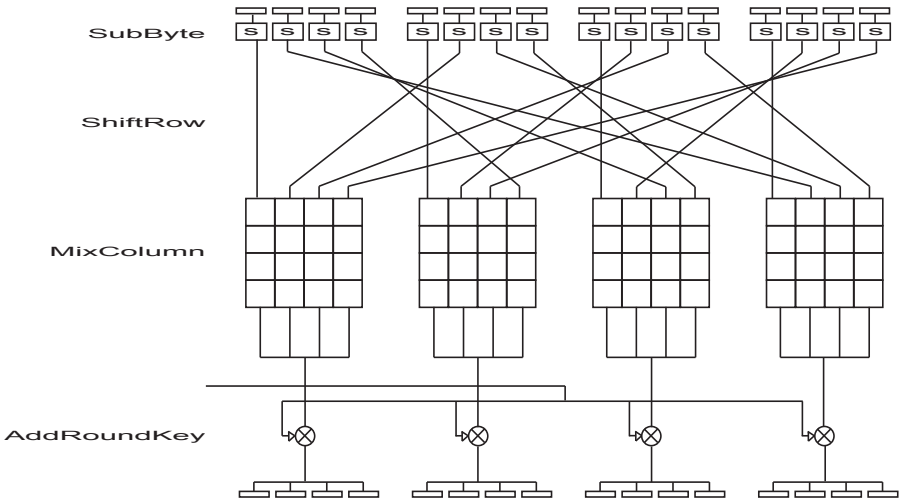


FIGURE 7: Symétrie dans l'algorithme de *Rijndael* (redessiné d'après [21])

l'implémentation des différentes opérations sont les suivants. Les S-Box, pour l'opération *SubByte*, sont implémentées via des tables de correspondances en logique combinatoire. Il y a 48 S-Box au total (32 pour un *round* et 16 pour le *KeyScheduling*). L'opération *ShiftRow* utilise également une table de correspondance pour déterminer l'étendue du décalage. L'opération *MixColumn* est implémentée directement d'après sa définition (Équation (A.3)). Enfin, l'opération *AddRoundKey* est une simple addition XOR. Le *KeyScheduling* est conçu de manière à pouvoir utiliser toute les combinaisons taille de bloc de données/taille de la clé

possible. La constante de *round* est également fournie via une table de correpondance.
Synthétisé en technologie CMOS 0.18μm de chez *National Semiconductor* en condition normale (1.8V et 16°C) le circuit a une superficie de 3.96mm^2 (173000 portes) et présente un débit max de 1.83Gbits/s à 100MHz.
Ainsi, même si ce travail ne présente pas une implémentation faible superficie à proprement parler, cette première optimisation montre déjà une réduction notable de superficie comparée aux précédents résultats.

Dans [22], Lu *et al.* présentent un circuit adapté aux applications contraintes par la taille telles que les *smart card*, PDA ou encore téléphones mobiles. L'architecture est de type itérative. Le chemin de données est de 128 bits. Deux cycles sont requis par *round*. Un (dé)chiffrement nécéssite 21 cycles. Les auteurs proposent de réutiliser le maximum de matériel pour effectuer le chiffrement et le déchiffrement. Ainsi, chacun des modules servant à effectuer les quatres opérations ainsi que le *KeyScheduling* peuvent réaliser l'opération directe ou inverse choisi par un bit de sélection présent sur les modules.
La S-Box utilise une table de correspondance (commune au chiffrement et déchiffrement) pour calculer l'inverse tandis que la transformation affine est réalisée en combinatoire. L'opération *MixColumn* est obtenue en décomposant la matrice de l'Equation A.3 en deux matrices puis en factorisant les termes communs.
La synthèse en technologie CMOS 0.25μm de chez *TSMC* produit un circuit de 32000 portes offrant un débit de 609Mbits/s à 100MHz. Ainsi, les auteurs rapportent qu'en partageant efficacement le matériel entre le chiffrement et le déchiffrement, la taille du circuit est réduite de 68% par rapport aux autres designs.

Grâce notamment à la S-Box entièrement combinatoire déjà présentée précédemment dans ce texte, Satoh *et al.* réduisent de manière significative la superficie du circuit dans [8]. L'architecture est ici aussi itérative. Le chemin de données est de 32 bits et le circuit dispose de quatre instances de la S-Box. Pour encore réduire la taille, ces quatres S-Box sont aussi utilisées pour le *KeyScheduling*, qui est réalisé *on-the-fly*. Ainsi, un *round* est réalisé en cinq coups d'horloge (pour une taille de bloc de données et de clé de 128 bits) et le chiffrement complet nécéssite 54 cycles ($10\times$ 5 + 4 cycles pour le premier *round* dont la clé est directement disponible sans utiliser les S-Box). Enfin, le circuit permet le chiffrement et le déchiffrement.
Les optimisations spécifiques aux opérations sont : *SubByte* est réalisée à l'aide de la S-Box déjà détaillée plus haut dans ce texte, *MixColumn* est optimisée de la même manière que dans [22] et enfin, le *KeyScheduling* utilise un générateur de constante de *round* évitant ainsi le recours à une table de correspondance.
Le circuit est synthétisé en technologie CMOS 0.11μm. La superficie obtenue est de 0.052mm^2 (5398 portes) et le débit est de 311Mbits/s à 131MHz. Une architecture 1 cycle/*round*, obtenue en augmentant la taille du chemin de données à 128 bits et en multipliant par quatre les instances de chaque bloc, a aussi été synthétisée. La superficie est mainteannt de 0.205mm^2

(21337 portes) pour un débit de 2.6 Gbits/s à 224MHz.

Reprochant au design de Satoh *et al.* d'avoir des chemins fortement non-balancés et possédant des fonctions de sélection qui gaspillent temps et superficie, Mangard *et al.* présentent une nouvelle approche dans [23]. Les clés du design sont : (1) des chemins balancés et courts, permettant d'utiliser efficacement chaque cycle et réduisant le risque de transitions non-désirées, ce qui est profitable pour un design basse consommation ; (2) une grande régularité dans l'architecture permettant un placement et routage plus efficace qui fournira un circuit plus compact ; (3) une architecture évolutive pouvant être mise à l'échelle facilement en fonction des besoins tels qu'une augmentation des performances au détriment de la taille, une utilisation de clé de 128, 192 ou 256 bits.
L'unité de données est présentée Figure 8. Chaque cellule comprend un registre 8 bits ainsi que de la logique combinatoire pour effectuer certaines opérations. Contrairement aux autres circuits qui stockent l'*Etat* en cours dans un registre de 128 bits unique, ici l'*Etat* est stocké dans 16 registres de 8bits. Les différentes opérations sont idéalement implémentées grâce à cette disposition en faisant transiter les données d'une cellule à l'autre. Toutes les opérations, à l'exception de *SubByte*, sont réalisées à l'intérieur des cellules.

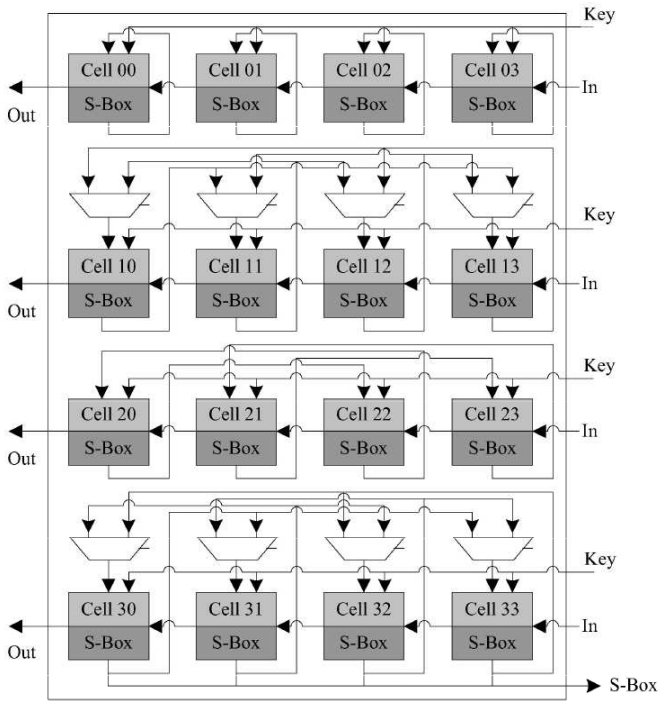


FIGURE 8: Architecture très régulière de Mangard *et al.* (source : [23])

La S-Box est une implémentation pipelinée à deux étages (afin de balancer correctement les chemins) de la version de Wolkerstorfer *et al.* [9]. Une version haute performance de l'architecture est également présentée et contient seize S-Box (une par cellule) au lieu de quatre.

Cette version permet d'éxécuter un *round* en trois cycles au lieu de six pour la version standard. Le *KeyScheduling* calcule les clés *on-the-fly* et utilise les S-Box de l'unité de données. Les deux versions sont synthétisées en technologie CMOS 0.6μm. La version standard contient 10800 portes et présente un débit de 128Mbits/s. La version haute performance contient 15500 portes et offre un débit de 241 MBits/s. Tous deux fonctionnent à 64MHz. En utilisant la même technologie CMOS que Satoh *et al.*, les auteurs estiment une fréquence de fonctionnement trois fois supérieure à la leur.

Alors que certaines implémentations sur circuits logiques programmables ont déjà été optimisées pour réduire la superficie, tel le travail de F.-X. Standaert *et al.* [24], la première implémentation spécialement optimisée basse consommation sur circuit intégré spécifique, en vue d'une utilisation sur tag RFID, est présentée par Feldhofer *et al.* dans [25]. Compte tenu de l'application, ils estiment qu'un courant de 15μA est disponible pour le circuit AES, qui ne devrait pas dépasser 5000 portes en superficie. De plus, le protocole utilisé pose une contrainte temporelle de 18ms pour le chiffrement d'un bloc de 128 bits de données. Les auteurs rapportent qu'aucun circuit jusqu'à présent ne peut respecter de telles contraintes.

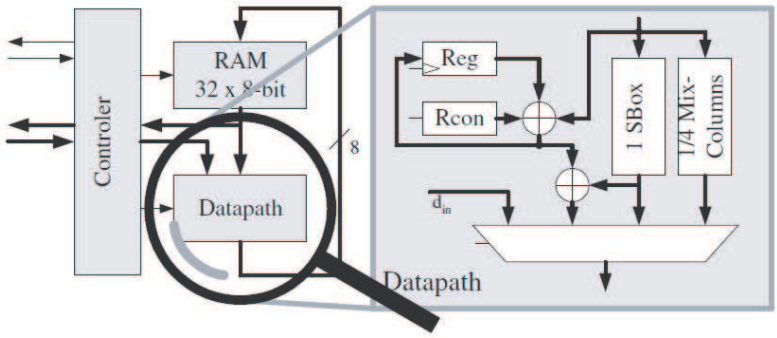


FIGURE 9: Architecture 8 bits proposée par Feldhofer *et al.* (source : [25])

L'architecture proposée, telle que présentée Figure 9, est une architecture itérative avec chemin de données de 8 bits. Cela permet de réduire fortement la consommation comparé aux architectures 128 bits ou même 32 bits. Le module contient une mémoire RAM de 256 bits pour stocker l'*État* ainsi que la *round key*, le chemin de données et une unité de contrôle implémentée par une machine à états. Le chemin de données contient de la logique combinatoire ainsi que des modules pour réaliser les opérations. L'opération *MixColumn* est effectuée byte par byte par un module. L'opération s'effectuant sur une colonne, celui-ci contient trois registres de 8 bits pour stocker temporairement les bytes de la colonne en cours. La S-Box est à nouveau une implémentation pipelinée à deux étages de la version de Wolkerstorfer *et al.* [9]. Seule une instance de la S-Box est présente dans le circuit pour minimiser la superficie. Un chiffrement complet nécessite 1016 cycles d'horloge. Synthétisé en technologie CMOS 0.35μm, la superficie du circuit est de 3595 portes. Une fréquence de 100kHz permet de respecter la contrainte temporelle. Enfin, la consommation du circuit est de 8.15μA.

La deuxième réalisation basse consommation, toujours pour application de type tag RFID, est proposée par Kim *et al.* dans [27]. Reprenant une architecture semblable à celle de Feldhofer *et al.* [25] (itérative, 8 bits), les auteurs proposent ici une optimisation au niveau fonctionnel en réorganisant et regroupant les opérations de l'algorithme comme indiqué sur la Figure 10. Ainsi, un *round* principal est éxécuté en deux étapes fonctionnelles, au lieu de trois dans [25]. La mémoire pour l'*État* et la clé est scindée en deux pour permettre un accès

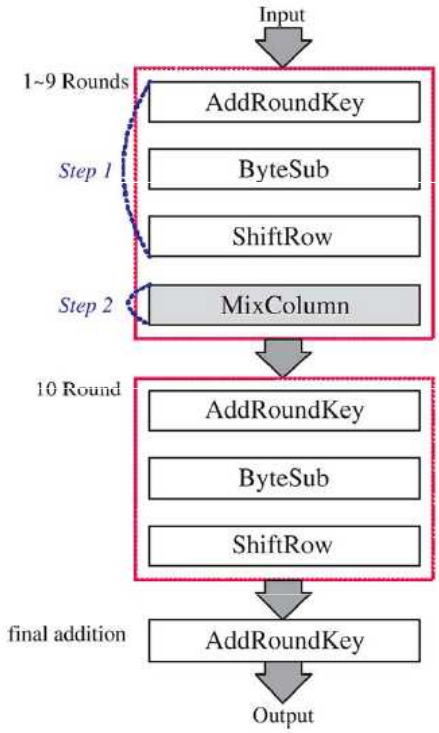


FIGURE 10: Réorganisation des *rounds* par Kim *et al.* (source : [27])

en parallèle. La S-Box est encore une implémentation de la S-Box présentée par Wolkerstofer *et al.* dans [9]. L'opération *MixColumn* se fait également byte par byte de la même manière que dans [25] à la différence près que le module contient ici un registre de 32 bits. Le circuit est synthétisé dans deux technologies CMOS 0.25μm 2.5V de chez *Hynix Corp.* et de chez *Samsung Electronics*. L'architecture proposée nécéssite 870 cycles d'horloge pour chiffrer 128 bits de données. Le circuit compte respectivement 3900 et 3868 portes pour les technologies *Hynix* et *Samsung*. Fonctionnant à 100kHz, la consommation est de 4.85μW (ou 1.94μA) et 21.4μW (ou 8.56μA).

Dans [28], Feldhofer *et al.* présentent une version améliorée de leur précédente réalisation [25]. Le circuit permet maintenant de chiffrer ou déchiffrer les données. L'architecture est presque identique à celle de [25]. Un module d'entrée/sortie est rajouté pour permettre d'utiliser le circuit comme un coprocesseur. Le clock gating est rigoureusement appliqué à

toutes les cellules séquentielles du circuit. Lorsque la S-Box n'est pas utilisée, son entrée est isolée et une valeur de {0x52} y est appliquée pour annuler la sortie. Les opérations *MixColumn* et *InvMixColumn* sont intégrées dans un même module. D'après les équations :

$$\begin{aligned} c(x) &= \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \\ c^{-1}(x) &= \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \\ c^{-1}(x) - c(x) &= \{08\}x^3 + \{0c\}x^2 + \{08\}x + \{0c\} \\ c^{-1}(x) - c(x) &= \{08\}(x^3 + x) + \{0c\}(x^2 + 1) \\ c^{-1}(x) &= c(x) + \{08\}(x^3 + x) + \{0c\}(x^2 + 1) \end{aligned}$$

les auteurs remarquent que l'opération inverse est facilement obtenue à partir de l'opération directe et que du matériel peut être économisé en intégrant les deux opérations dans le même module comme le montre leur implémentation à la Figure 11.

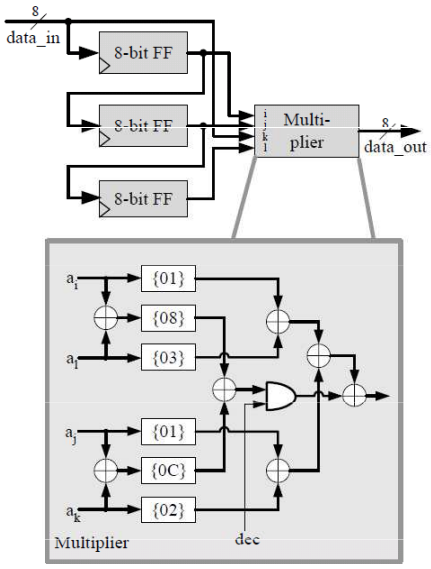
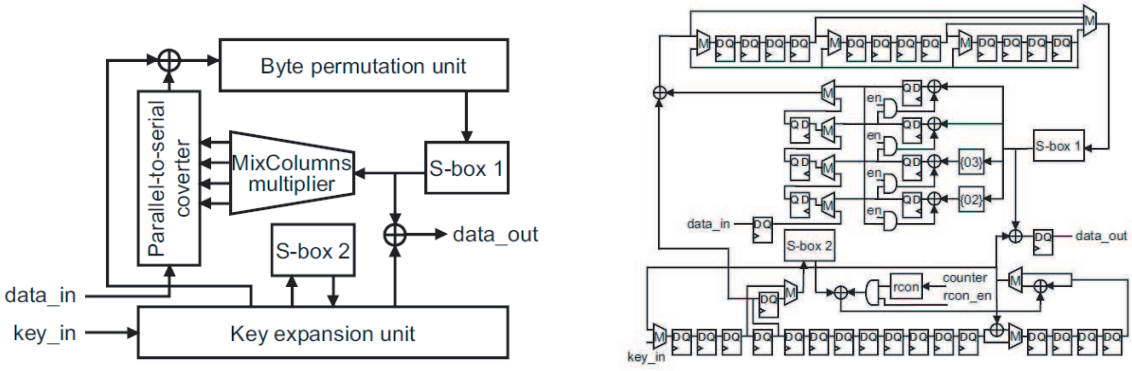


FIGURE 11: Module réalisant les opérations *Inv/MixColumn* (source : [28])

Le circuit est synthétisé et réalisé en technologie CMOS 0.35μm de chez *Philips Semiconductors*. Le chip fonctionnel obtenu comporte 4400 portes pour une superficie de 0.25mm^2. La consommation du circuit est directement mesurée sur le chip. A 100kHz et avec une tension d'alimentation de 1.5V, le courant est de 3μA. Les auteurs reportent que le circuit est fonctionnel jusqu'à une tension minimum de 0.65V.

Enfin, une dernière implémentation basse consommation est réalisée par Hämäläinen *et al.* dans [30]. Encore une fois, la taille du chemin de données est de 8 bits. Mais une toute autre approche concernant l'architecture est montrée ici. A la différence des autres travaux

[25], [27] et [28] qui réalisent les différentes opérations séquentiellement via une machine à états, les opérations sont exécutées en parallèle.



(a) (b)

FIGURE 12: (a) Diagramme haut-niveau, (b) Chemin de données (source : [30])

La Figure 12 présente le diagramme haut-niveau en (a) et l'implémentation de celui-ci en (b). Chacunes des opérations possèdent leur propre module qui sont disposés les uns à la suite des autres. Avec une telle architecture, la S-Box du chemin de données principal est continuellement utilisée. Dès lors, une deuxième S-Box est présente sur le circuit pour éviter de devoir arrêter l'exécution des *round* lors du calcul des clés par le *KeyScheduling*. Etant implémentée selon le design très compact de Canright [11], cette S-Box supplémentaire n'augmente pas de beaucoup la taille du circuit. Ce design permet de réaliser le chiffrement en 160 cycles d'horloge.
Le circuit est synthétisé en technologie CMOS 0.13μm 1.2V. En optimisant la synthèse pour la plus faible consommation, le circuit comporte 3200 portes et consomme 13μW/MHz. La fréquence maximum est de 130MHz ce qui permet un débit de 104Mbits/s.

On retiendra de ces travaux qu'ils ont tous été réalisés avec une technologie CMOS supérieure à 0.11μm et qu'aucun n'utilise la logique sous-seuil, pourtant très efficace pour réduire la consommation comme le présente la section suivante. Aussi, si la réduction de la consommation est souhaitée à tout prix, on privilégiera une architecture avec chemin de données étroit (8 bits) et de type itérative, au détriment des performances.

2.2 Logique sous-seuil

2.2.1 Introduction

Etant donné l'étendue des recherches dans le domaine de la logique sous-seuil, il sera bien difficile ici d'être exhaustif. Cependant, un certain nombre de publications récentes présentant des circuits à très faible consommation grâce à une tension d'alimentation V_{dd} sous la tension de seuil V_t seront exposées ici. Cela permettra de décrire les différentes techniques utilisées dans la logique sous-seuil mais aussi de mettre en évidence le gain non négligeable de consommation apporté. Mais voyons d'abord pourquoi la logique sous-seuil est intéressante lorsqu'une faible consommation est requise.

L'énergie disipée par une porte logique peut être divisée en deux catégories : énergie statique et énergie dynamique. Le premier terme correspond à l'énergie consommée lorsque la porte est au repos. Cette dissipation est dûe uniquement aux différents courants de fuite. Le second terme correspond à l'énergie dissipée lorsque qu'un noeud est chargé ou déchargé. Les expressions de ces deux termes peuvent s'écrire comme :

$$E_{dyn} \propto C_{eff} V_{dd}^2 \tag{2.2}$$

$$E_{stat} = \frac{I_{leak} V_{dd}}{f_{clk}} \tag{2.3}$$

avec C_{eff} la charge effective de la porte (contenant C_L, la capacité de charge du noeud, ainsi que des facteurs tenant compte des courants de court-circuit et du facteur d'activité), f_{clk} la fréquence d'horloge et I_{leak} les courants de fuite.
Lorsque que l'on réduit la tension d'alimentation V_{dd}, on observe que les deux composantes de l'énergie telles que décrites ci-dessus vont être diminuées, comme le suggère les Équations (2.2) et (2.3). Cependant, cette diminution de tension a d'autres conséquences sur le circuit logique. En effet, en observant l'expression du délai d'une porte :

$$T_{del} \propto \frac{C_L V_{dd}}{I_{on}}, \tag{2.4}$$

I_{on} étant le courant actif qui croît exponentiellement avec V_{dd} en régime sous-seuil, on remarque que cette diminution de tension va faire augmenter le délai de manière significative. Cette augmentation de délai, qui correspond à une réduction de la fréquence, pourra être acceptable pour les applications demandant de faibles ou moyennes performances. Par contre, un deuxième regard à l'Équation (2.3) nous indique que cette réduction de fréquence va à son tour conduire à une augmentation de l'énergie statique et qui sera malheureusement bien plus importante que la réduction provoquée par la diminuton de V_{dd}. Ainsi, on comprend que d'une part la réduction de l'énergie dynamique et d'autre part l'augmentation de l'énergie statique donnera lieu à un point d'énergie minimum. Ce point d'énergie minimum n'est pas une constante mais varie en fonction de la technologie utilisée [43], des caractéristiques du

circuit et des conditions de fonctionnement [32]. Il est possible de le calculer en additionnant les deux composantes de l'énergie et en calculant la dérivée par rapport à V_{dd}. Le développement est fait dans [31] et [32]. Les résultats montrent que la tension V_{dd} correspondant à ce point d'énergie minimum, V_{min}, ne dépend pas de V_t et se situe dans la région $V_{dd} < V_t$. Cela s'explique par le fait que, dans la région sous-seuil, le période pendant laquelle le circuit "fuit" (délai) croît exponentiellement alors que les courants de fuite ne sont pas réduits aussi rapidement lorsque que V_{dd} diminue.

Ainsi, on voit l'interêt de la logique sous-seuil lorsqu'il s'agit de concevoir des circuits à (très) faible consommation. La fin de ce chapitre présente les techniques utilisées en logique sous-seuil et les résultats de quelques circuits récents opérant à une tension V_{dd} située sous la tension de seuil V_t et affichant des consommations très faibles.

2.2.2 Bibliothèque de cellules

Lors de la synthèse de circuits en régime sous-seuil à l'aide de bibliothèques de cellules standards, une sélection et/ou modification de celles-ci est nécessaire. En effet, lorsque que la tension d'alimentation devient trop faible, les cellules peuvent ne plus fonctionner correctement. Considérons le cas d'un inverseur. Celui-ci fonctionnera correctement que si son rapport I_{on}/I_{off} est suffisament grand. Ce rapport est fortement réduit en régime sous-seuil dû à la dépendance exponentielle de I_{on} avec V_{dd}. Dès lors, en considérant les variations technologiques (*corners* FS et SF [4]), il est calculé dans [33] que, pour un inverseur en technologie 0.18μm, la tension de fonctionnement minimum, $V_{dd,limit}$, est de 195 mV, lorsque qu'il est idéalement dimensionné. Pour les dimensions standards de la bibliothèque, la tension minimum est de 220mV. Cependant, cette tension minimum de fonctionnement, $V_{dd,limit}$, sera en général, pour les cellules simples, inférieure à la tension correspondant au point d'énergie minimum, V_{min}.
Il en va autrement pour des cellules plus complexes ou des cellules séquentielles. La Figure 13 montre le cas d'un flip-flop en technologie 0.18μm [32]. Les inverseurs sont tous de tailles égales et minimales, sauf I_3 qui est plus grand, diminuant ainsi $t_{clk \rightarrow Q}$. Lorsque la tension est diminuée, le PMOS de I_6 ne parvient pas à garder le noeud N_3 à l'état haut à cause des courants de fuite des NMOS de I_6 et I_3 qui sont trop importants. Une réduction de la taille de I_3 et une augmentation de I_6 est nécéssaire pour pouvoir utiliser ce flip-flop jusqu'à une tension de 300mV.
Un autre exemple est donné dans [39] pour un multiplexeur. Les auteurs proposent de transformer un multiplexeur 4:1 *one-hot* en un multiplexeur 4:1 *encodé* à l'aide de multiplexeurs 2:1 de manière à réduire les transistors "fuiants" en parallèle. Cette méthode permet d'obtenir une tension du niveau logique de sortie diminuée de 10% au lieu de presque 30% par rapport V_{dd} lorsque V_{dd} = 200mV.

4. F pour Fast, S pour Slow ; la première lettre se rapportant au NMOS, la deuxième au PMOS.

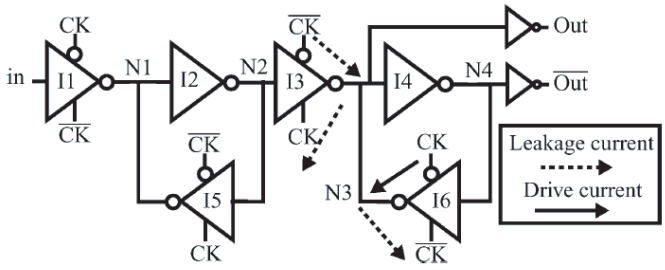


FIGURE 13: Flip-flop (source : [32])

Plus généralement, plusieurs structures sont à éviter. Les cellules comportant des transistors en parallèle auront un rapport I_{on}/I_{off} réduit à faible tension. En effet, les courants de fuite des transistors en parallèle vont être aussi importants que le courant actif. La Figure 14 montre une porte XOR standard (a) (technologie 0.18μm) ainsi que les fuites dans les transistors en parallèle lorsque l'entrée A est basse et l'entrée B est haute (b). Alimentée à 100mV, la tension de sortie ne sera que de 55mV dans ce cas là, ce qui rend la cellule inutilisable [33].

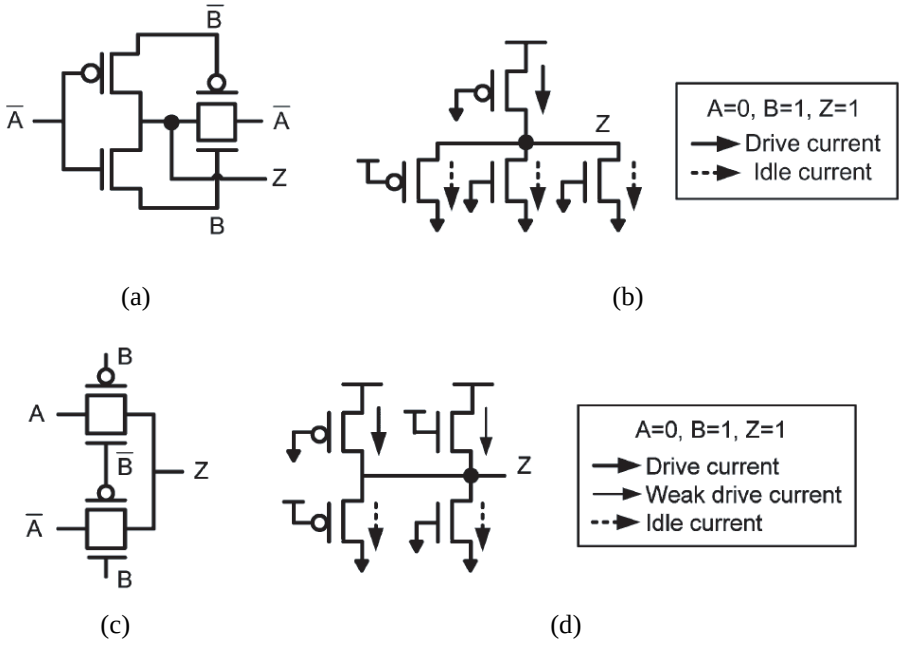


FIGURE 14: (a) Porte XOR standard, (b) Fuites dans les transistors en parallèle quand {A=0, B=1}, (c) Porte XOR avec transistors de transmission, (d) Courants actifs et courants de fuite balancés (source : [33])

Une solution consiste à balancer correctement les courants de fuite avec les courants actifs par l'utilisation de transistors de transmission, comme sur la Figure 14 (c) et (d). Dans ce cas là, la porte est fonctionnelle à 100mV.
Les cellules comportant des transistors en série sont aussi à éviter. Cela à pour effet de réduire le courant actif. La tension de seuil des transistors empilés va aussi être augmentée, réduisant encore plus les courants actifs.

Dans tous les cas, la clé des designs en régime sous-seuil proposés jusqu'ici consiste à réduire la tension d'alimentation V_{dd} au point d'énergie minimum. Par ailleurs, quand le facteur de mérite important est la puissance instantannée, le concepteur veillera à rapprocher V_{min} au plus près de $V_{dd,limit}$, pour obtenir le circuit le plus efficace qu'il soit en terme de puissance [34].

2.2.3 Variabilité

Les circuits logiques sont soumis à différentes variations. Il peut s'agir de variations globales opérationnelles (V_{dd}, température), de variations globales au niveau du process (longueur de grille, épaisseur d'oxide) ou encore de variations aléatoires comme la fluctuation du dopage dans le canal. Un effet important de ces variations va être de modifier V_t. Or, en régime sous-seuil, les portes logiques sont soumises à une sensibilité exponentielle à V_t. Dès lors, des disfonctionnements peuvent apparaitre. Ceux-ci sont classés en deux catégories. On parle de disfonctionnement fonctionnel [35](la porte n'est plus capable de charger sa sortie à la tension d'alimentation) et de disfonctionnement paramétrique (violation de contrainte temporelle) [36] [38].

Lors de la conception d'un circuit, il est donc important de tenir compte des variations globales en prenant certaines marges pour s'assurer ainsi que le circuit reste 100% fonctionnel. Malheureusement, cela va écarter le circuit de son point de fonctionnement minimisant l'énergie. Différentes solutions ont donc été étudiées pour palier à ce problème. Une de ces solutions est le *body biasing*, qui consiste à appliquer une tension V_b au substrat du circuit et qui aura pour effet de modifier V_t. Ainsi, le courant I_{on} et donc le délai sont controlés. Cette tension peut être auto-régulée, comme dans [40].

Une autre solution consiste à faire varier V_{dd} – *voltage scaling* – qui, de la même manière que V_t, va permettre de contrôler I_{on} et donc le délai du circuit. Dans [37], une boucle de régulation de V_{dd} est proposée pour un filtre FIR en technologie CMOS 65nm. Le système permet d'adapter la tension d'alimentation comprise entre 0.25V et 0.7V ainsi que la fréquence en fonction de la charge du circuit. La méthode permet d'obtenir une réduction d'énergie de l'ordre de 50 à 100 %.

Une comparaison de ces deux techniques est faite dans [44]. Il en ressort que le *body biasing* est plus efficace que le *voltage scaling* car il donne lieu à un E_{min} plus faible.

2.2.4 Résultats récents

Kaul *et al.* [39] présentent un accélerateur d'estimation de mouvement en technologie CMOS 65nm. En plus d'une architecture spécialement étudiée pour un fonctionnement à basse tension, les auteurs proposent un ajustement de la tension d'alimentation de +/-50mV. Cette compensation permet d'annuler les effets des variations de température et de process lorsque le circuit travaille à une tension de 320mV. A la tension nominale de 1.2V, le circuit consomme 48mW pour une fréquence de 2GHz. Lorsque qu'il est utilisé en région sous-seuil à une tension de 320mV, qui représente le point d'énergie minimum, la consommation chute à 56μW et la fréquence à 23MHz. Cette consommation peut encore être diminuée jusqu'à 14.4μW en diminuant la tension à 230mV et qui permet d'utiliser le circuit à une fréquence de 4.3MHz.

Pu *et al.* présentent dans [41] un coprocesseur JPEG en technologie CMOS 65nm. Pour balancer correctement la tension de seuil des PMOS et NMOS du circuit soumise à des fluctuations dûes aux différentes variations (telles que décrites plus haut), ils utilisent un système de *body biasing* qui applique une tension à l'un des deux *well* du chip. La tension appliquée est auto-régulée via un petit module mesurant la différence de V_t des deux types de transistors. A la tension nominale de 1.2V, le circuit présente une consommation energétique de 7.5pJ/opération. Cette consommation est réduite à 0.9pJ/opération lorsque le circuit travaille en régime sous-seuil à une tension de 0.4V.

Un deuxième circuit proposé par Kaul *et al.* utilise la logique sous-seuil [42]. Il s'agit ici d'un accélérateur de traitement vectoriel quadrivoie à double alimentation reconfigurable. Les techniques utilisées pour réduire la consommation sont (1) une double alimentation qui, en alimentant les parties du circuit les plus rapides avec la tension la plus basse, permet de réduire la consommation tout en gardant la même fréquence et (2) le *power gating* qui déconnecte de l'alimentation les parties du circuit non-utilisées. La consommation du circuit est de 161mW à 1.1V à une fréquence de 2.3GHz. En régime sous-seuil, la consommation chute à 87μW à 230mV pour une fréquence de 8.8MHz. L'efficacité énergétique maximale est obtenue à 300mV avec 494GOPS/W.

Pour terminer cette section, il est intéressant de se demander ce que devient le comportement en régime sous-seuil des circuits pour les technologies plus avancées. Le travail de D. Bol *et al.* [46] étudie les performances en régime sous-seuil de circuits, du noeud technologique 0.25μm jusqu'au noeud 32nm. Une évolution vers les noeuds plus avancés permet de réduire grandement la consommation dynamique grâce à la réduction de la capacité C_L ainsi que de la largeur de grille. Par contre, la consommation statique va augmenter à partir du noeud 180nm principalement à cause de la variabilité. La conclusion est que les noeuds

65/45nm en technologie LP [5] sont les plus intéressants pour la logique sous-seuil.

Ainsi, on voit que la logique sous-seuil permet de réduire de plusieurs ordres de grandeur la consommation des circuits. Cela se fait évidemment avec une nette diminution des performances. Cependant, pour les circuits ayant besoin d'une très faible consommation et dont les contraintes temporelles sont relativement lâches, les performances obtenues en régime sous-seuil sont encore acceptables. Enfin, il apparait que la technologie 65nm LP semble apporter le compromis idéal entre consommation dynamique et statique en régime sous-seuil.

2.3 Conclusion

À la lumière de ce chapitre, on apperçoit une partie des motivations de ce travail. La première section nous a permis de remarquer que les coprocesseurs AES proposés jusqu'à maintenant n'utilisent pas la logique sous-seuil. De plus, les technologies utilisées dans ces travaux ne sont en général pas les plus avancées. La deuxième section nous a montrer que la logique sous-seuil se présente comme une solution idéale pour réduire la consommation des circuits. Aussi, une technologie plus avancée, telle la technologie 65nm LP, est plus adaptée, en terme d'efficacité énergétique, à la conception de circuits fonctionnants en régime sous-seuil.

Alors que la plupart des coprocesseurs AES réalisés jusqu'à présent souffrent d'une consommation encore fort élevée pour être utilisés sur des tags RFID, la logique sous-seuil semble bien adaptée à de tels circuits. Ce type de circuits ne présente pas de contrainte stricte sur la vitesse de fonctionnement mais est par contre très limité concernant sa consommation. En effet, le volume d'information échangé par les tags RFID est en général assez faible et le temps imparti pour l'échange peut s'élever à plusieurs centaines de millisecondes. D'un autre coté, l'alimentation de ces circuits donne une borne maximale pour la consommation énergétique (dans le cas d'une alimentation par batteries) ou pour la puissance (dans le cas d'une alimentation par champ électro-magnétique). L'utilisation d'un coprocesseur AES pour tag RFID en régime sous-seuil semble donc approprié pour réduire au maximum sa consommation. Ainsi, c'est à juste titre que ce travail propose cette étude.

5. pour *Low Power*. Cette technologie apporte une augmentation de la longeur de grille, de l'épaisseur de l'oxyde de grille et du dopage du canal en vue de diminuer les courants de fuite.

Chapitre 3

Caractérisation des cellules

Les bibliothèques de cellules sont indispensables à la conception de circuits digitaux. Comme les fondeurs fournissent leurs bibliothèques uniquement caractérisées à la tension nominale, la conception et l'optimisation de circuits en logique sous-seuil n'est pas possible. En effet, sans ces bibliothèques caractérisées à basse tension, le comportement d'un circuit en régime sous-seuil ne peut être que mesuré, une fois le circuit réalisé à l'aide des bibliothèques caractérisées à la tension nominale. Dès lors, il apparait indispensable de posséder des bibliothèques caractérisées à basse tension pour pouvoir prédire le comportement du circuit et surtout pouvoir l'optimiser. Cette section étudie l'évolution des cellules lorsque V_{dd} diminue en vue de caractériser une bibliothèque à basse tension. La première section propose une introduction aux bibliothèques digitales et à leur contenu. Ensuite, l'étude d'un inverseur lorsque V_{dd} diminue est présentée au niveau du délai et de la consommation. Le problème de la variabilité est également abordé. Enfin, la méthode de caractérisation est présentée.

3.1 Introduction aux bibliothèques digitales

Le flot de design d'un circuit intégré digital nécessite l'utilisation de bibliothèques de cellules digitales. Ces bibliothèques contiennent, pour chacune des cellules, la description complète de celles-ci. Principalement, il y est décrit : les délais de propagation, les contraintes (pour les cellules séquentielles), les temps de montée/descente en sortie, la puissance de fuite pour chaque vecteur d'entrée, la puissance interne pour chaque transition ainsi que les capacités de chaque entrée. Tout ceci est détaillé dans cette section. La caractérisation d'une cellule consiste à obtenir les valeurs de ces différents paramètres. Ensuite, ces bibliothèques seront utilisées principalement pour synthétiser le circuit et pour vérifier la consommation

de celui-ci. Il est donc essentiel de travailler avec une bibliothèque correctement caractérisée au niveau des délais pour être sûr de respecter la contrainte temporelle imposée au circuit. Notons que la caractérisation de la consommation des cellules est en général moins critique car cela n'influence pas la synthèse du circuit. La suite de cette section décrit brièvement la manière dont les délais et consommations sont modélisés au sein des bibliothèques.

3.1.1 Délai dans la bibliothèque

Différents modèles peuvent être utilisés pour décrire le comportement temporel des cellules. Le plus courant est le modèle non-linéaire. Dans ce modèle, le délai total, de l'entrée d'une porte à l'entrée de la porte suivante est :

$$D_{total} = D_{propagation} + D_{connexion}, \tag{3.1}$$

avec $D_{propagation}$, le délai de propagation de la cellule, définit comme le délai entre l'instant où V_{in}=0.4V_{dd} et le point V_{out}=0.6V_{dd} [1] (pour une transition descendante à la sortie, et inversément pour un transition montante) et $D_{connexion}$ qui représente le délai dû aux interconnexions entre la sortie de la porte et l'entrée de la porte suivante. Ces deux composantes se calculent comme suit :

$D_{propagation}$ Le délai de propagation est obtenu à l'aide d'une recherche dans une table de correspondance indexée par le temps de transition à l'entrée et par la capacité de sortie. Pour obtenir une valeur particulière, une interpolation est réalisée sur les quatre points les plus proches (voir Figure 15).

$D_{connexion}$ Le délai de la connexion se calcule comme un délai de type *RC* en considérant que le réseau est balancé (les ports d'entrée se partagent la résistance et la capacité du fil à part égale) et vaut :

$$D_{connection} = \frac{R_{net}}{N}\left(\frac{C_{net}}{N} + C_{port}\right), \tag{3.2}$$

avec R_{net} la résistance totale et C_{net} la capacité totale d'interconnexion, C_{port} la capacité d'entrée de la porte suivante et N le fan-out.

Ainsi, pour pouvoir calculer les délais d'un circuit logique, les bibliothèques contiennent :

- des tables de correspondance indexées par le temps de transition à l'entrée et par la capacité de sortie pour les délais de cellule des transitions montantes et descendantes,
- des tables de correspondance indexées par le temps de transition à l'entrée et la capacité de sortie pour les temps de transition à la sortie,

1. Les seuils de 0.4V_{dd} et 0.6V_{dd} sont ceux utilisés par *STMicroelectronics* dans leurs bibliothèques et nous avons donc gardé cette convention. En général, un seuil unique de 0.5V_{dd} à l'entrée et à la sortie est courament utilisé.

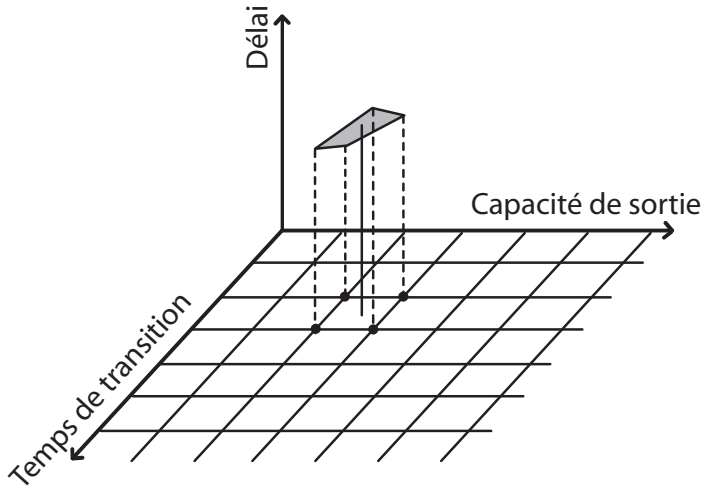


FIGURE 15: Table de correspondance pour le calcul du délai, indexée par le temps de transition à l'entrée et la capacité de sortie

- des modèles d'interconnexion (avec R_{net} et C_{net} en fonction de la longueur de l'interconnexion),
- les valeurs des capacités des ports d'entrée des cellules.

3.1.2 Puissance dans la bibliothèque

L'équation modélisant la puissance dissipée dans une cellule est la suivante :

$$P_{total} = P_{statique} + P_{dynamique}, \tag{3.3}$$

avec $P_{statique}$ la puissance dissipée lorsque la cellule est au repos et $P_{dynamique}$ la puissance dissipée lorsqu'un noeud de la cellule voit sa tension modifiée suite à un stimulus à ses entrées (entraînant ou non un changement d'état à la sortie).

$P_{statique}$ La puissance statique est due aux différents courants de fuite indésirables. Elle se calcule selon :

$$P_{statique} = I_{leak}\, V_{dd}, \tag{3.4}$$

avec I_{leak} le courant de fuite au repos. Comme ce courant varie en fonction de l'état de la cellule, il est calculé pour chaque vecteur possible.

$P_{dynamique}$ La puissance dynamique peut être à nouveau scindée en deux composantes : puissance de commutation et puissance interne.

$$P_{dynamique} = P_{commutation} + P_{interne} \tag{3.5}$$

- Puissance de commutation
  La puissance de commutation est la puissance nécessaire à la charge et la décharge

de la capacité présente sur le noeud de sortie de la cellule. Elle s'exprime par :

$$P_{commutation} = \frac{1}{2} C_{charge} V_{dd}^2 TR, \tag{3.6}$$

avec TR le taux de commutation et C_{charge} la capacité composée de la capacité de charge de la sortie et de la capacité des interconnexions.

– Puissance interne
La puissance interne est la puissance consommée par les changements d'état des noeuds internes à la cellule ainsi que par les courants de court-circuit. Cette puissance peut être dissipée même sans changement d'état à la sortie. Elle est calculée selon :

$$P_{interne} = \sum_{port} E_{port} \times AF_{port}, \tag{3.7}$$

avec E_{port} l'énergie dissipée par transition pour le port *port* et AF_{port} le facteur d'activité pour le port *port*. L'énergie E_{port} est obtenue par recherche dans une table de correspondance indexée par le temps de transition à l'entrée et par la capacité de sortie.

Ainsi, pour pouvoir calculer la puissance d'un circuit logique, les bibliothèques contiennent :

– les puissances de fuite pour chaque vecteur,
– les valeurs des capacités de chaque port,
– les tables de correspondance indexées par le temps de transition à l'entrée et la capacité de sortie pour l'énergie par transition de chaque port.

3.2 Modélisation du délai et de la puissance

Avant d'aller plus loin dans l'étude des cellules à très basse tension, il est bon de se rappeller les relations liant puissance et délai à V_{dd}. Commençons par le délai. Comme vu plus haut, le délai d'une porte logique est donné par :

$$T_{del} \propto \frac{C_L V_{dd}}{I_{on}}, \tag{3.8}$$

où C_L est la charge de la porte et I_{on} est le courant actif. La charge C_L, constituée des capacités parasites en sortie de la porte et de la capacité du noeud de sortie (capacité du fil et capacité d'entrée de la porte) est propre à la technologie utilisée et au circuit et ne dépend pas de V_{dd}. Le courant actif I_{on} est lui dépendant du régime de fonctionnement de la porte. En régime de forte inversion ($V_{dd} > V_t$), ce courant, noté $I_{on,fi}$, est défini par :

$$I_{on,fi} = \mu C_{ox} (W/L) (V_{dd} - V_t)^{\alpha} \tag{3.9}$$

avec α prenant des valeurs entre 1 et 2 en fonction de la saturation de la vitesse (*velocity saturation*) [45]. Ainsi, lorsque V_{dd} est supérieur à V_t, on remarque une dépendance quasi linéaire à V_{dd} du courant actif I_{on} (en fonction de α). Lorsque la tension V_{dd} diminue par delà le seuil que représente V_t, le courant actif sous-seuil, noté $I_{on,ss}$, est maintenant défini par [46] :

$$I_{on,ss} = I_0 \; 10^{\frac{V_{dd}(1+\eta)}{S}} \left(1 - e^{\frac{-V_{dd}}{U_{th}}}\right) \tag{3.10}$$

avec I_0 regroupant les termes invariables, S la pente sous-seuil, η le coefficient d'abaissement de la barrière de potentiel par le drain (DIBL) et U_{th} le potentiel thermique. Sachant que le potentiel thermique vaut 26mV à température ambiante, le second terme de l'équation devient vite négligeable lorsque V_{dd} dépasse ce potentiel thermique ($\sim V_{dd}$ >0.1V). Ainsi, comme la tension limite de fonctionnement des portes logiques CMOS sera supérieure à 0.1V, nous pouvons considérer l'équation suivante pour le courant sous-seuil :

$$I_{on,ss} = I_0 \; 10^{\frac{V_{dd}(1+\eta)}{S}}. \tag{3.11}$$

À la différence du courant actif en régime de forte inversion, le courant sous-seuil dépend de V_{dd} de manière exponentielle. Le délai de la porte, tel que présenté Equation 3.8, augmentera donc de manière quasi linéaire lorsque V_{dd} >V_t mais de manière exponentielle lorsque V_{dd} <V_t.

Concernant la puissance, ses deux composantes, dynamique et statique, sont définies par :

$$P_{dyn} \propto C_{eff} \; V_{dd}^2 \; f_{clk} \tag{3.12}$$

$$P_{stat} = I_{leak} \; V_{dd} \tag{3.13}$$

Le terme C_{eff} représente la capacité effective et tient compte des courants de court-circuit et du facteur d'activité en plus de la capacité de charge C_L. Ainsi, les deux composantes de la puissance dynamique, puissance interne et puissance de commutation, telles que définies dans les bibliothèques, se retrouvent ici regroupées en un seul terme via C_{eff}. La puissance dynamique P_{dyn} est fortement réduite lorsque V_{dd} diminue. D'une part directement par cette réduction de V_{dd}, d'autre part par la réduction de fréquence liée à cette diminution de V_{dd}, comme vu ci-dessus (augmentation de T_{del}).
Le terme I_{leak}, présent dans la deuxième équation, représente le courant de fuite lorsque V_{gs} = 0. Il est principalement constitué du courant de conduction sous-seuil $I_{off,ss}$, du courant de grille par effet tunnel I_{gate} et du courant de jonction en polarisation inverse *drain-susbtrat* I_{junc}. Le courant de conduction sous-seuil, lorsque V_{gs} est nul, est donné par :

$$I_{off,ss} = I_0 \; 10^{\frac{\eta V_{dd}}{S}}. \tag{3.14}$$

Il diminue légèrement avec V_{dd}, dû à l'effet d'abaissement de la barrière de potentielle par le drain. Le modélisation du courant de grille I_{gate} est plus complexe. On retiendra seulement

que celui-ci diminue aussi avec V_{dd}. Enfin, le courant de jonction I_{junc} est également plus complexe à modéliser. À température ambiante, celui-ci est très faible comparé aux deux précédents et peut être négligé. La puissance statique P_{stat} va donc diminuer avec V_{dd}, par sa relation directe avec celle-ci et par la réduction du courant de fuite I_{leak}.

3.3 Cas de l'inverseur

Notre étude de l'évolution du délai et de la consommation des circuits digitaux lors de la diminution de V_{dd} commence par l'investigation de l'inverseur. L'inverseur est la cellule la plus simple de la bibliothèque mais son étude en régime sous-seuil donne déjà une bonne vue d'ensemble du comportement des celulles en régime sous-seuil.
Les résultats présentés dans cette section, et dans tout le restant de ce travail, proviennent de simulations Spice opérées par ELDO [2]. Les simulations utilisent les modèles de transistors BSIM4 ainsi que les sous-circuits reprenant les éléments parasites de la librairie CMOS 65nm LP SVT 1.2V de chez *ST Microelectronics*.

Tout d'abord, la Figure 16 présente les courbes caractéristiques (V_{out} en fonction de V_{in}) de l'inverseur *INVX2* $((W/L)_N = 0.2/0.06); (W/L)_P = 0.28/0.06)$ pour différents V_{dd}. L'utilité de ce graphe est de montrer que l'inverseur est bien fonctionnel à très basse tension, ce qui a bien tout son intérêt ici. On considère que l'inverseur est fonctionnel tant que le gain

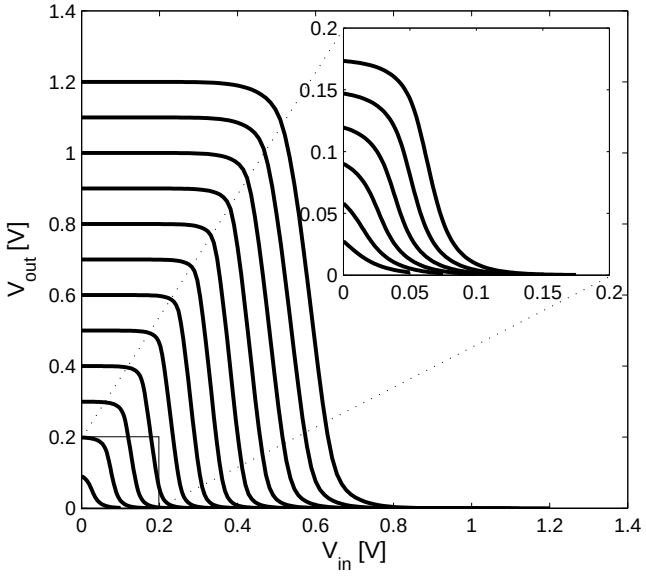


FIGURE 16: Courbes caractéristiques de l'inverseur *INVX2* pour différents V_{dd}

2. Eldo de Mentor Graphics est un simulateur de circuit intégré compatible Spice.

au point V_{in}=V_{out} est inférieur à -1. Il apparait clairement sur la figure que la tension V_{dd} peut être diminuée jusqu'à une tension inférieure à 100mV. Cependant, en dessous de 150mV, la tension du niveau logique haut de sortie n'est plus égale à V_{dd}. Dès lors, la tension d'alimentation V_{dd} devra être supérieure à 150mV.

3.3.1 Délai

L'évolution du délai T_{del} d'un inverseur *INVX2* en fonction de la tension d'alimentation V_{dd} est présentée Figure 17. Le délai est défini comme étant le temps compris entre les points à 40% et 60% de la tension V_{dd} respectivement pour l'entrée et la sortie pour une transition descendante à la sortie et le temps compris entre les points à 60% et 40% de la tension V_{dd} respectivement pour l'entrée et la sortie pour une transition montante à la sortie. Le délai moyen pour les deux types de transition est présenté sur la Figure 17. L'inverseur est simulé avec un temps de transition à l'entrée égal au temps de transition à la sortie de cet inverseur. Enfin, l'inverseur est chargé par une capacité à peu près égale à deux fois la capacité d'entrée de l'inverseur (1.4 fF). On constate, comme prévu, l'augmentation du

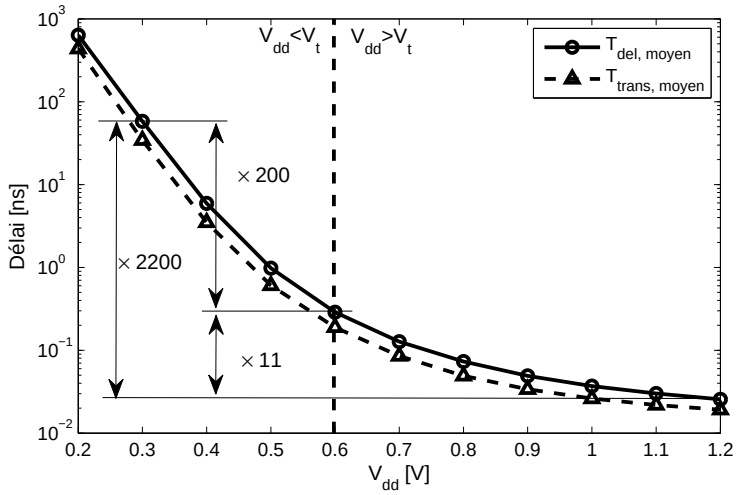


FIGURE 17: Délai et temps de transition moyen de l'inverseur INVX2 en fonction de V_{dd}

délai lors de la réduction de V_{dd}. De plus, les régions de forte inversion et sous-seuil sont bien marquées avec une augmentation bien plus prononcée pour V_{dd} <0.6V = V_t. Ainsi, en diminuant la tension de 1.2V, la tension nominale, à 0.6V, une augmentation du délai d'un facteur 11 est observable. En diminuant encore la tension de 0.6V à 0.3V, le délai subit une augmentation d'un facteur 200.

La Figure 17 présente également le temps de transition à la sortie de l'inverseur T_{trans} en fonction de V_{dd}. Ce temps est défini comme étant le période comprise entre les points

V_{out} = 0.2V_{dd} et V_{out} = 0.8V_{dd} pour une transition montante à la sortie et entre les points V_{out} = 0.8V_{dd} et V_{out} = 0.2V_{dd} pour une transition descendante à la sortie. On peut observer sur la figure que le comportement de T_{trans} est identique à celui de T_{del} lorsque V_{dd} diminue.

Les valeurs des délais et temps de transition moyens de l'inverseur *INVX2* pour différents V_{dd} sont indiqués à la Table 1.

TABLE 1: Délai et temps de transition de l'inverseur *INVX2*

V_{dd}	1.2V	0.8V	0.6V	0.4V	0.2V
$T_{del,moyen}$	25.62 ps	73.03 ps	287.93 ps	5.94 ns	636.07 ns
$T_{trans,moyen}$	19.14 ps	49.12 ps	188.9 ps	3.49 ns	436.29 ns

3.3.2 Puissance et énergie

Avant d'étudier l'évolution de la consommation de l'inverseur en fonction de V_{dd}, il est bon de regarder d'où provient cette consommation. La Figure 18 présente les profils temporels des tensions, courants et puissances instantanées de l'inverseur. L'inverseur à l'étude est commandé et chargé par deux inverseurs identiques au premier. Le graphe supérieur (a) de la figure montre les tensions d'entrée et de sortie de l'inverseur. On remarque le petit dépassement (overshoot) au début des transitions dû au couplage capacitif direct entre l'entrée et la sortie.

Le deuxième graphe (b) présente le courant fourni par l'alimentation ainsi que le courant présent à la sortie de l'inverseur. Lors d'une transition descendante, le courant de l'alimentation I_{vdd} présente deux pics. Le premier, négatif, représente un courant "absorbé" par l'alimentation et est dû au couplage capacitif direct entre grille et drain. En réalité, dans un circuit complet comportant plusieurs milliers de portes, ce courant sera utilisé par d'autres portes transitant au même moment si bien que l'alimentation, vue de l'extérieur, n'"absorbera" aucun courant. Le deuxième pic, positif, représente le courant fourni par l'alimentation pendant la décharge du noeud de sortie. Ce courant correspond au courant de court-circuit circulant lorsque le PMOS et le NMOS sont tous deux passants. Le pic de courant I_z visible à la sortie représente le courant permettant de vider le noeud de sortie et est dirigé vers la masse. Lors d'une transition montante, I_{vdd} présente un unique pic. Ce pic est constitué du courant actif permettant de charger le noeud de sortie ainsi que du courant de court-circuit, identique à celui de la transition descendante. Le courant actif chargeant le noeud de sortie est visible sur le graphe car il s'agit du pic de I_z.

Le dernier graphe (c) présente la puissance totale instantanée de l'inverseur ainsi que la puissance de commutation instantanée. Le profil de la puissance totale est identique à

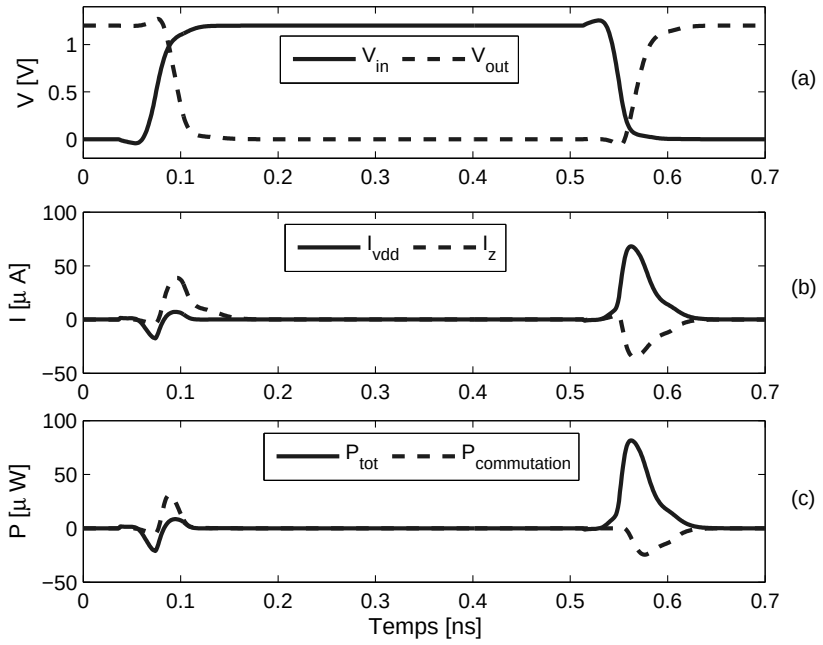


FIGURE 18: Profils temporels pour une transition descendante et montante de l'inverseur de : (a) Tension d'entrée (V_{in}) et de sortie (V_{out}) ; (b) Courant fourni par l'alimentation (I_{vdd}) et courant à la sortie de l'inverseur (I_z) ; (c) Puissance totale instantanée (P_{tot}) et puissance de commutation instantanée ($P_{commutation}$).

celui du courant d'alimentation à un facteur V_{dd} près. En effet, $P_{tot} = I_{vdd}V_{dd}$. Le profil de la puissance de commutation est quant à lui un peu différent car le courant I_z est ici multiplié par V_{out} qui n'est pas constant : $P_{commutation} = I_z V_{out}$.

Regardons maintenant l'évolution de ces puissances lorsque V_{dd} diminue. La Figure 19 présente l'évolution des trois composantes de la puissance telles que définies dans les bibliothèques, à savoir puissance interne, puissance de commutation et puissance statique. Celles-ci sont calculées de la manière suivante. La puissance statique $P_{statique}$ est directement fournie par une simulation DC. La puissance de commutation $P_{commutation}$ est l'intégrale de la puissance de commutation instantanée ($P_{commutation}$ sur la Figure 18 (c)) divisée par la période de la transition (largeur du pic de la puissance totale instantanée $\approx$ délai) (Équation (3.15)). Enfin, la puissance interne $P_{interne}$ est l'intégrale de la puissance totale instantanée (P_{tot} sur la Figure 18 (c)) divisée par la même période de la transition et à laquelle est soustraite la puissance de commutation et la puissance statique (Équation (3.16)). Cela s'écrit :

$$P_{commutation} = \frac{\int_{\Delta T} P_{commutation,inst}\,\mathrm{d}t}{\Delta T}, \tag{3.15}$$

$$P_{interne} = \frac{\int_{\Delta T} P_{tot,inst}\,\mathrm{d}t}{\Delta T} - P_{commutation} - P_{statique} \tag{3.16}$$

avec ΔT la largeur du pic de la puissance totale instantanée.

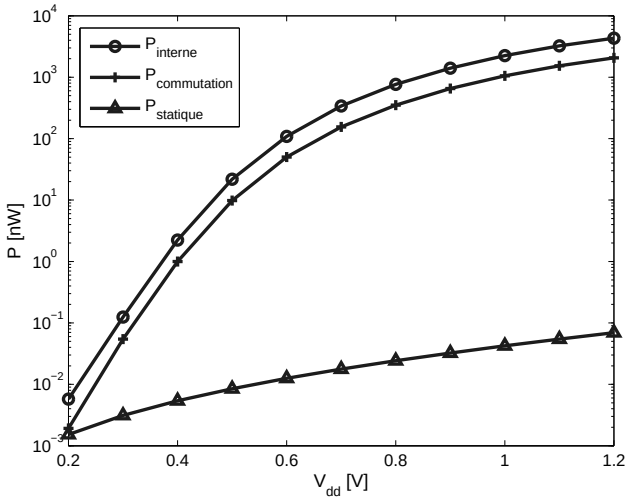


FIGURE 19: Évolution de la puissance interne, la puissance de commutation et la puissance statique d'un inverseur en fonction de V_{dd}.

Comme suggéré par l'Équation (3.12) et (3.13), les trois composantes de la puissance diminuent lorsque V_{dd} diminue. La puissance statique est réduite de manière exponentielle par un facteur total de 45 lorsque V_{dd} passe de 1.2V à 0.2V. La courbe de la puissance dynamique, diminuant aussi de manière exponentielle, présente un net changement lorsque V_{dd} passe sous le seuil. Cela est dû à la forte augmentation du délai dès que $V_{dd} < V_t$. Ainsi, la puissance dynamique est réduite d'un facteur 40 entre V_{dd} =1.2V et V_{dd} =0.6V tandis qu'une réduction d'un facteur 21000 apparait entre V_{dd} =0.6V et V_{dd} =0.2V. Il est à noter que les puissances ci-dessus sont les puissances moyennes pour une transition montante et descendante. La puissance de commutation, qui n'est fournie par l'alimentation que lors d'une transition montante, est donc réduite de moitié.

Les valeurs des puissances internes, de commutation et statiques se trouvent indiquées Table 2 pour différents V_{dd}.

TABLE 2: Puissance de l'inverseur *INVX2*

V_{dd}	1.2V	0.8V	0.6V	0.4V	0.2V
$P_{interne}$	4.32 μW	763.22 nW	108.95 nW	2.23 nW	5.74 pW
$P_{commutation}$	2.06 μW	351.01 nW	50.13 nW	998.89 pW	1.92 pW
$P_{statique}$	69.39 pW	24.21 pW	12.52 pW	5.4 pW	1.53 pW

Alors que la puissance est une donnée importante lorsque l'alimentation du circuit est extraite d'un champ électro-magnétique, comme c'est le cas pour un tag RFID, l'énergie

par opération est par contre le paramètre essentiel à étudier lorsque l'alimentation provient d'une batterie. De plus, le choix de la tension d'alimentation sera dictée par la consommation d'énergie, comme introduit dans le chapitre de l'état de l'art. L'étude de l'énergie par transition de l'inverseur est donc également présentée ici.

La Figure 20 présente l'évolution de l'énergie dynamique et statique pour l'inverseur en fonction de V_{dd} (trait continu). À nouveau, le comportement est bien celui attendu au regard

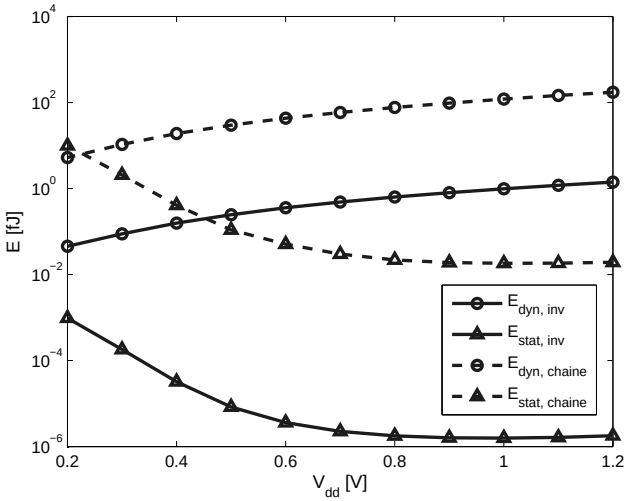


FIGURE 20: Évolution de l'énergie statique et dynamique par transition d'un inverseur (trait continu) ainsi que d'une chaine de 121 inverseurs (trait tireté) en fonction de V_{dd}.

des Équations (2.2) et (2.3). L'énergie dynamique décroît quadratiquement avec V_{dd}. Une réduction d'un facteur 31 apparait lorsque V_{dd} passe de 1.2V à 0.2V. Par contre, l'énergie statique, qui est donnée par intégration de la puissance instantanée de fuite sur une période correspondant au délai de la porte, augmente lorsque V_{dd} diminue. Cette augmentation est encore plus marquée lorsque $V_{dd} < V_t$. Cela est à nouveau dû à la forte augmentation du délai dans cette région. Ainsi, la diminution de V_{dd} de 1.2V à 0.6V provoque une augmentation de l'énergie statique d'un facteur 4. Lorsque V_{dd} passe de 0.6V à 0.2V, l'énergie statique augmente d'un facteur 270.

On remarque que le point d'énergie minimum est situé en dessous de 0.2V. En effet, la consommation statique d'une porte unique est très faible comparée à la consommation dynamique et l'énergie totale correspond uniquement à l'énergie dynamique. La Figure 20 présente également la consommation énergétique d'une chaîne de 121 inverseurs (trait tireté). Lorsque le nombre de portes du circuit augmente d'un facteur N, l'énergie dynamique sera multipliée par ce même facteur N. Par contre, l'énergie statique va être multipliée par ce facteur N au carré. En effet, en plus de l'augmentation du courant de fuite (qui se voit

multiplié par le facteur N), le délai subit également une augmentation (aussi d'un facteur N). Notons que ceci est valable en considérant un facteur d'activité constant (il vaut 1 ici). Ainsi, on observe sur la Figure 20 le déplacement de la courbe d'énergie dynamique de deux ordres de grandeur vers le haut (plus précisément, elle est multipliée par un facteur 121). La courbe de l'énergie statique est par contre déplacée vers le haut de quatre ordres de grandeur (correspondant à 121^2). Comme la courbe d'énergie statique augmente plus que celle de l'énergie dynamique, le point d'énergie minimum va être déplacé vers la droite. De plus, l'énergie minimum E_{min} correspondante augmente également. Ici, E_{min} vaut 15fJ (pour une transition du noeud à la sortie de la chaine) et est atteint à $V_{dd} \cong 0.2$V.
Même si c'est ici présenté sur un circuit très simple comportant uniquement des inverseurs, cette mobilité du point d'énergie minimum sera semblable sur tous les circuits digitaux. Cependant, il faudra tenir compte de la variation du facteur d'activité.

La Table 3 donne les valeurs de consommation d'énergie dynamique et statique par transition pour différents V_{dd}, cela pour l'inverseur et la chaîne de 121 inverseurs.

TABLE 3: Énergie consommée par transition pour l'inverseur *INVX2* et une chaine de 121 inverseurs.

V_{dd}	1.2V	0.8V	0.6V	0.4V	0.2V
$E_{dyn,inv}$	1.42 fJ	0.63 fJ	0.36 fJ	0.16 fJ	0.05 fJ
$E_{stat,inv}$	1.78e-6 fJ	1.8e-6 fJ	3.61e-6 fJ	32.0e-6 fJ	971.1e-6 fJ
$E_{dyn,chaine}$	174.3 fJ	76.7 fJ	43.15 fJ	19.02 fJ	5.25 fJ
$E_{stat,chaine}$	19.3e-3 fJ	21.9e-3 fJ	50.7e-3 fJ	407.5e-3 fJ	9.95 fJ

3.3.3 Variabilité

Nous avons abordé le problème de la variabilité dans le chapitre sur l'état de l'art. On a notamment remarqué que ce problème se fait d'autant plus ressentir que la tension V_{dd} est faible. Regardons ce qu'il en est sur l'inverseur. Il faut distinguer la variabilité globale de la variabilité locale. La variabilité globale comprend les modifications de certains paramètres sur l'ensemble du circuit. Ces paramètres sont principalement la température ainsi que certains paramètres des transistors (modélisé par les *corners*). La température, semblable à l'ensemble du circuit, aura au premier ordre un effet sur le délai et la consommation. Elle ne modifiera pas fondamentalement le fonctionnement du circuit. L'effet des *corners* sera aussi de modifier délai et consommation du circuit si les variations sont identiques au PMOS

et au NMOS (*corners* SS, TT et FF [3]). Cependant, si les variations diffèrent entre NMOS et PMOS (*corners* "croisés" SF et FS), alors le fonctionnement du circuit va également être modifié (modification de la valeur des niveaux logiques de sortie avec risque de ne plus pouvoir fournir le niveau haut ou bas selon les cas, dégradation des marges de bruit) [32].
La variabilité locale englobe les modifications aléatoires de certains paramètres indépendamment pour chaque transistor. Le paramètre principal est le dopage dans le canal, qui influe sur la tension de seuil V_t. Cela aura des effets sur le délai et la consommation du circuit ainsi que sur son fonctionnement.

La Figure 21 montre la courbe caractéristique typique de l'inverseur pour V_{dd} = 1.2V à 25 ˚C et *corner* TT (trait continu). Superposée à celle-là se trouve également la courbe caractéristique de l'inverseur pour V_{dd} =1.2V à -25˚C et *corner* SS (trait pointillé) (seule une réduction de la température a été considérée ici car cela à pour effet d'augmenter le délai et constitue donc un pire-cas pour le délai, à respecter lors de la synthèse). Ceci permet d'observer l'effet de la variabilité globale. Comme annoncé ci-dessus, la variabilité globale (hormis les *corners* "croisés"), identique à chaque transistor, n'a pas d'effet majeur sur le fonctionnement du circuit et les deux courbes sont donc quasiment identiques. Ensuite, l'effet

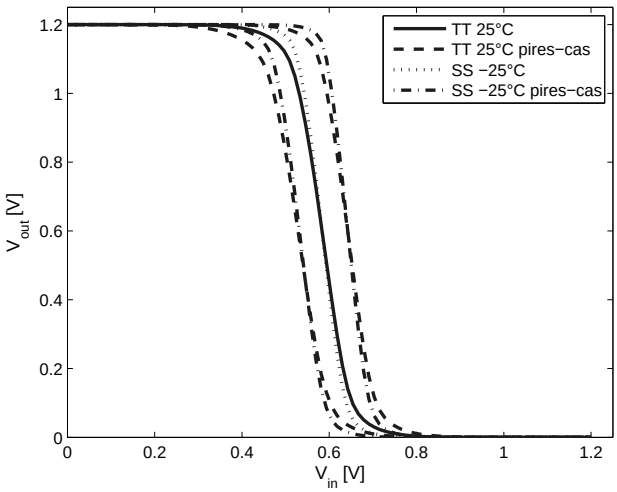


FIGURE 21: Courbe caractéristique de l'inverseur *INVX2* pour V_{dd} =1.2V

de la variabilité locale est également visible sur le graphe. Des simulations Monte-Carlo (500 runs) ont été réalisées sur les deux courbes caractéristiques précédentes. Les résultats extrêmes de la simulation Monte-Carlo pour les deux courbes sont montrés sur le graphe (trait tireté et trait mixte). Les effets de la variation locale sont identiques pour les deux courbes. On remarque une variation de la courbe de l'ordre de 4.2% de V_{dd} (50mV) par

3. S pour Slow, T pour Typical et F pour Fast ; la première lettre s'appliquant au NMOS, la deuxième au PMOS.

rapport à la courbe typique. Cela engendrera une diminution de la marge de bruit.

La Figure 22 montre les mêmes courbes, mais à une tension V_{dd} =0.2V cette fois-ci. Ici encore, le *corner* SS et la réduction de la température n'ont pas d'influence majeure (trait continu et trait tireté). Par contre, la variabilité locale présente maintenant des effets beaucoup plus catastrophiques. Les résultats extrêmes de la simulation Monte-Carlo montrent maintenant une variation de la courbe de l'ordre de 25% de V_{dd} (50mV) (trait mixte). Enfin,

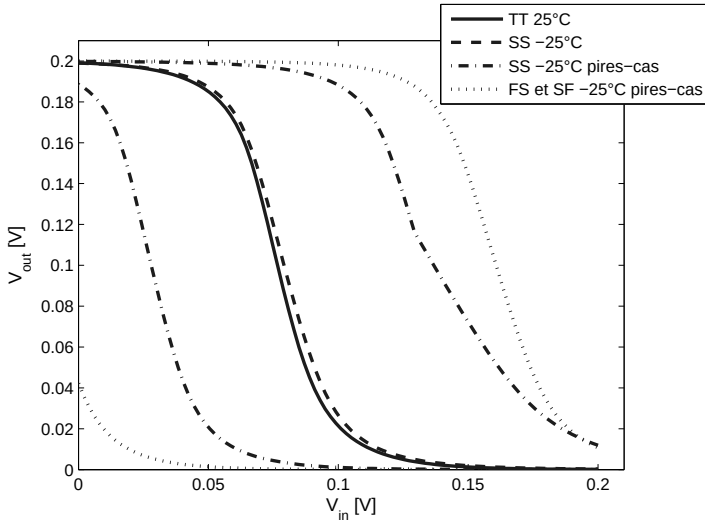


FIGURE 22: Courbe caractéristique de l'inverseur INVX2 pour V_{dd} =0.2V

les simulations Monte-Carlo ont été réalisées aux *corners* "croisés" FS et SF, toujours à -25˚C. Au *corner* FS, le NMOS est plus grand que le PMOS et cela va réduire la force qu'a l'inverseur pour charger sa sortie. Cela se traduit par un décallage de la courbe caractéristique vers la gauche. A l'inverse, au *corner* SF, le PMOS est plus grand que le NMOS et l'inverseur aura plus de mal à décharger son noeud de sortie. Cela se traduit par un décallage de la courbe caractéristique vers la droite. Sur le graphe, les courbes extrêmes (trait pointillé) présentent maintenant une variation de 50% de V_{dd} par rapport à la courbe typique. Ainsi, en considérant les variabilités globales et locales, l'inverseur ne pourra pas être utilisé à V_{dd} =0.2V.

Pour montrer l'effet de la variabilité globale sur le délai, la Figure 23 montre l'évolution du délai en fonction de V_{dd} d'un inverseur à 25˚C, *corner* TT (marqueurs ronds) et à -25˚C, *corner* SS (marqueurs triangulaires). La réduction de température ainsi que le passage au *corner* SS vont tous deux faire augmenter le délai de l'inverseur. Au *corner* SS, les NMOS et PMOS présentant une tension de seuil plus élevée vont directement diminuer I_{on} et donc faire augmenter le délai. En réduisant la température, V_t va être augmenté et va également réduire I_{on} et donc le délai. Ainsi, à V_{dd} =0.2V, le délai va augmenter d'un facteur 24.

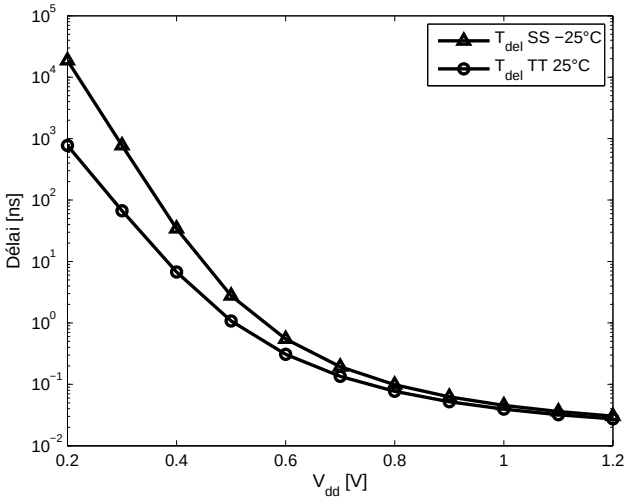


FIGURE 23: Évolution du délai en fonction de V_{dd} d'un inverseur *INVX2* à 25˚C, *corner* TT (marqueurs ronds) et à -25˚C, *corner* SS (marqueurs triangulaires).

Enfin, l'effet de la variabilité locale sur le délai est présenté sur la Figure 24. La figure montre la distribution du délai, normalisé au délai typique, pour une transition descendante (a) et montante (b), cela pour différents V_{dd}. À V_{dd} =1.2V, la distribution du délai se fait selon une normale. En effet, se rappelant l'Équation (3.9) et sachant que la variabilité du dopage du canal est modélisé par — entre autres variations, mais celle-ci étant celle qui a le plus d'effet — une distribution normale de V_t, on voit que I_{on} ainsi que T_{del} seront également distribués selon une loi normale. Lorsque V_{dd} diminue et passe en dessous du seuil V_t, l'Équation (3.9) n'est plus valide mais est remplacée par l'Équation (3.10). La dépendance exponentielle de I_{on} à V_t, au travers de I_0, va être à l'origine d'une distribution lognormale pour ce courant I_{on} ainsi que pour le délai T_{del}. De plus, on remarque également la dérive et l'étalement progressif des distributions lorsque V_{dd} diminue. Alors qu'à 1.2V, la moyenne est équivalente au délai typique et que les pires-cas à 3σ sont limités à 18% du délai typique, il en est autrement lorsque V_{dd} est plus faible. Ainsi, à V_{dd} =0.3V, la moyenne pour une transition descendante est située 68% au delà du délai typique. Les pires-cas à 3σ atteignent maintenant 5.5 fois le délai typique.

Les effets de la variabilité sur la consommation, qui n'auront pas d'influence directe sur la synthèse du circuit, ne sont pas montrés ici.

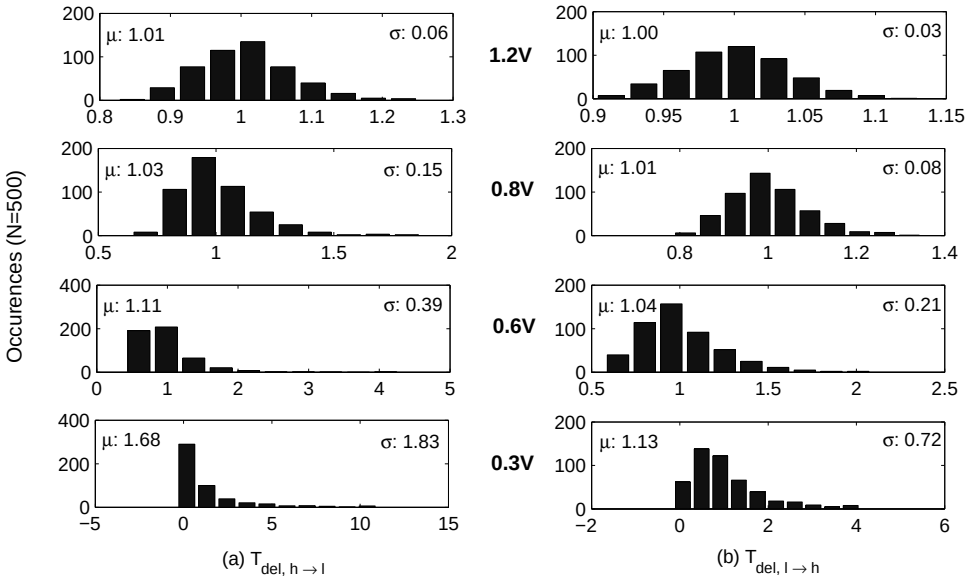


FIGURE 24: Distribution du délai, normalisé au délai typique, de l'inverseur *INVX2* pour différent V_{dd}. Conditions : (*corner* SS, température de -25˚C). (a) Délai pour transition descendante. (b) Délai pour transition montante.

3.4 Flot de caractérisation

Le cas de l'inverseur présenté ci-dessus est relativement simple (compte tenu du faible nombre d'états et de transitions possible) et la caractérisation manuelle de celui-ci est envisageable (c'est-à-dire les simulations SPICE seront peu nombreuses et simples). Cependant, ce n'est pas le cas pour toutes les cellules. En particulier, il est beaucoup plus long et fastidieux de simuler une cellule séquentielle (plus de paramètres à caractériser) ou encore une cellule combinatoire comportant un nombre d'entrées plus élevé (plus d'états et de transitions). Il apparaît dès lors essentiel d'automatiser la caractérisation des cellules. Cette tâche, réalisée pour la première fois en DICE, et qui constitua une grande partie de ce travail, est décrite en détails à l'Annexe B sous forme d'un tutoriel. Ce tutoriel à pour vocation de fournir une aide précieuse à quiconque désirera à l'avenir caractériser une bibliothèque de cellules.

Seul le schéma présentant le flot de caractérisation est montré ici sur la Figure 25. La caractérisation est réalisée à l'aide de l'outil Liberty NCX de Synopsys.

3.5 Conclusion

Après avoir présenté et décrit brièvement le contenu des bibliothèques de cellules, cette section étudie l'évolution du délai et de la consommation des cellules, sur base d'un inver-

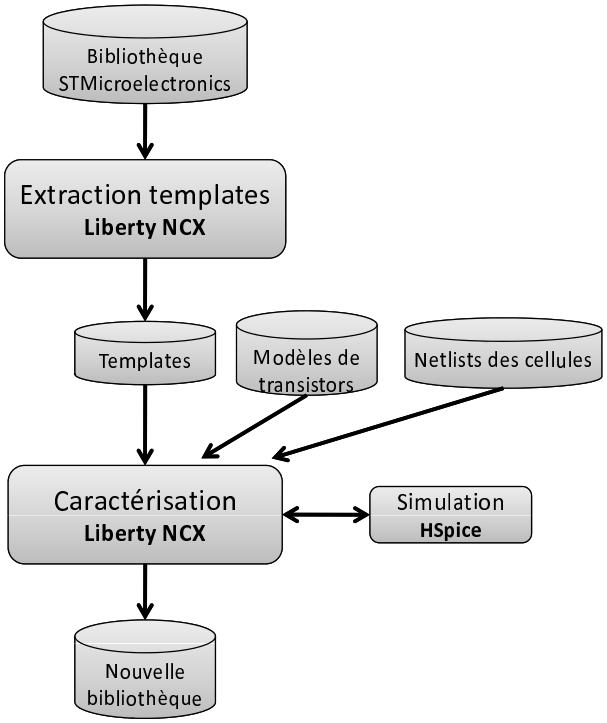


FIGURE 25: Flot de caractérisation d'une bibliothèque de cellules à l'aide de Liberty NCX de Synopsys.

seur, lorsque V_{dd} diminue. Il est montré que le passage de V_{dd} de 1.2V à 0.2V génère une augmentation exponentielle du délai d'un facteur total supérieur à 2000. Le délai à 0.2V vaut 636ns. A l'inverse, la puissance, surtout la puissance dynamique, diminue notablement lorsque V_{dd} diminue. La puissance totale est réduite de six ordres de grandeur pour atteindre 9.19pW à 0.2V. La courbe de l'énergie par transition en fonction de V_{dd} présente, quant à elle, un minimum. Celui-ci est situé en dessous de 0.2V car l'énergie dynamique domine. Cependant, sur une chaine de 121 inverseurs, ce minimum évolue et atteint 15fJ à $V_{dd} \cong$0.2V. Enfin, la variabilité est également étudiée. La variabilité globale a un effet sur le délai. Celui-ci est augmenté d'un facteur 24 en passant du *corner* TT à SS et en diminuant la température de 25˚C à -25˚C. La variabilité locale ainsi que les *corners* "croisés" SF et FS, en plus de modifier le délai, vont influencer le fonctionnement de l'inverseur en réduisant grandement sa marge de bruit. Le processus de caractérisation est montré en fin de chapitre. La caractérisation de la biliothèque à basse tension peut maintenant se faire. Le chapitre suivant présente l'utilisation de cette bibliothèque pour la synthèse d'un coprocesseur AES.

Chapitre 4

Coprocesseur AES faible consommation

Le chapitre précédent a présenté l'évolution de l'inverseur lorsque V_{dd} diminue. Grâce à cela, on a pu quantifier la réduction de puissance qu'il est possible d'obtenir. Cependant, même s'il s'agit d'une bonne indication, cette réduction de puissance ne peut pas s'appliquer tel quel sur le circuit AES. En effet, le circuit possède un grand nombre de portes (modification de la puissance statique) de types différents (modification de la puissance statique et dynamique) et, de par son architecture propre, présente des facteurs d'activité particuliers en chaque noeud (modification de la puissance dynamique). Une méthode de caractérisation a été également proposée dans le chapitre précédent. Ainsi, des bibliothèques de cellules peuvent maintenant être recaractérisées à basse tension, dans le but de synthétiser le circuit AES et d'en rapporter sa consommation. Le début de ce chapitre présente brièvement le coprocesseur AES étudié. Ensuite, les résultats de synthèse pour la bibliothèque nominale à V_{dd} =1.2V sont exposés. À partir de là, une nouvelle bibliothèque est présentée et caractérisée à faible V_{dd}. L'influence de la taille des cellules est décrite et l'utilité de la bibliothèque recaractérisée à faible V_{dd} est prouvée. Enfin, les résultats optimaux, en terme de puissance, sont présentés pour le coprocesseur AES, avec et sans tenir compte de la variabilité.

4.1 Présentation et validation fonctionnelle du coprocesseur AES

Le code VHDL décrivant le coprocesseur AES utilisé dans ce travail provient de F. Gosset et F.-X. Standaert. Il s'agit d'un coprocesseur basé sur une architecture itérative avec un chemin de données d'une largeur de 8 bits. Il permet de chiffrer des blocs de données de 128 bits à l'aide d'une clé de 128 bits. Le coprocesseur peut être divisé en 2 parties, une unité de contrôle et un chemin de données. L'unité de contrôle est implémentée par une machine à

états. Le chemin de données contient des sous-modules tels une S-Box, un module *MixCol*, un module *rcon* ainsi qu'un registre et de la logique combinatoire. Un bloc de donnée nécéssite 1142 cycles d'horloge pour être chiffré. Toutes les synthèses réalisées dans cette section sont celles de ce coprocesseur.

Avant de synthétiser le circuit, le code VHDL est simulé afin de vérifier qu'il décrit bien la fonctionalité voulue. Le document [2], décrivant le standard AES, fournit des vecteurs de test permettant d'accomplir cette tâche. Le code est simulé à l'aide de Modelsim [1]. Une horloge de période 10ns est utilisée (le choix de la période d'horloge est insignifiant ici car seule la fonctionnalité est vérifiée). La Figure 26 représente la fin de la simulation. Sans entrer dans les détails (le design architectural du coprocesseur AES n'étant pas l'objet de ce travail), on peut y vérifier deux choses. Le relevé de la sortie *data_out* montre que le chiffrement s'est effectué correctement (sur base du vecteur de test proposé dans [2]). Ensuite, on voit que 11420 ns se sont écoulées depuis le début du chiffrement. Avec une période de 10 ns, cela correspond à 1142 cycles d'horloge. On considère donc le module comme étant tout à fait fonctionnel.

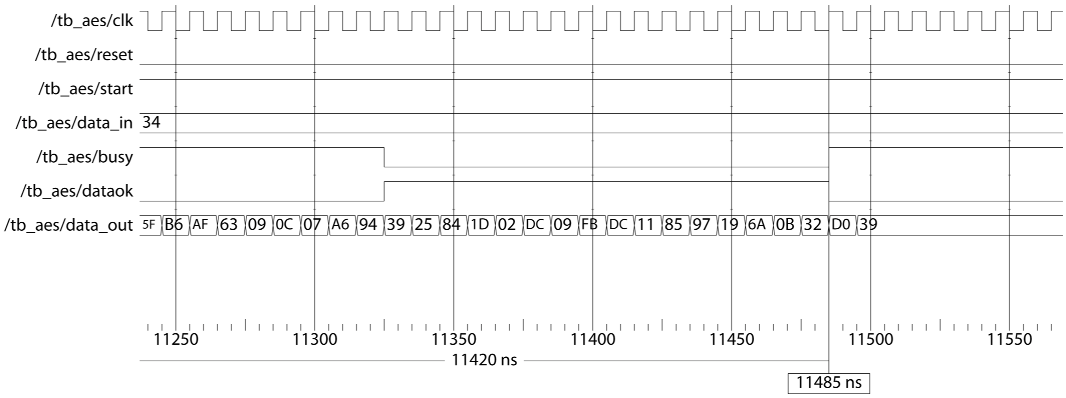


FIGURE 26: Résultat de la simulation fonctionnelle du coprocesseur AES.

4.2 Synthèse à V_{dd} nominal

Pour fournir un point de départ permettant de comparer les résultats obtenus à basse tension, le circuit est d'abord synthétisé avec la bibliothèque 65nm LPSVT [2] de *STMicroelectronics* à sa tension nominale de 1.2V. Les synthèses sont exécutées à l'aide de l'outil Design Compiler de Synopsys. Pour obtenir un circuit plus compact, la hiérarchie est détruite du-

1. Modelsim de Mentor Graphics est un environnement de simulation et débuggage pour HDL.
2. Low Power Standard-V_t, (T_{OX}=1.85nm, L_g=60nm, std-V_T=0.6V).

rant la synthèse. Il s'agit d'une synthèse basique, sans optimisation particulière. Le circuit synthétisé présente une complexité matérielle d'environ 1300 portes pour une superficie de 6400 μm^2.

En plus de celle caractérisée à la tension nominale, *STMicroelectronics* fournit une bibliothèque caractérisée à V_{dd} =1.0V. La Figure 27 présente la répartition de la puissance pour ces deux bibliothèques. Pour chaque barre, la puissance est répartie entre puissance statique, puissance interne et puissance de commutation. La première barre montre le résultat de la synthèse pour la bibliothèque 1.2V SVT avec, comme point de départ, une fréquence de 10MHz. La puissance totale est de 70.3 μW et est très largement dominée par la puissance dynamique (<1% de puissance statique). Comme on l'a déjà remarqué dans l'état de l'art,

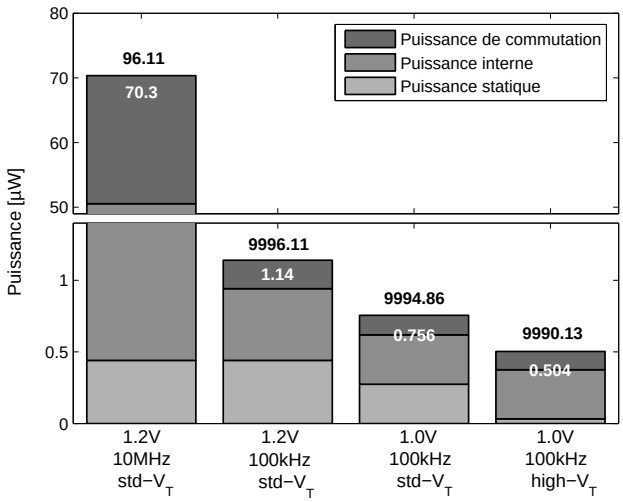


FIGURE 27: Répartition de la puissance du coprocesseur AES pour différentes versions de la bibliothèque 65nm de *STMicroelectronics* avec indication de la puissance totale (caractères blancs) et du slack time [ns] (caractères noirs).

les tags RFID ne sont pas contraints (jusqu'à un certain point) par les performances. Dès lors, une fréquence de 10 MHz n'est pas nécessaire. Comme exemple, nous pouvons vérifier la contrainte posée par un chip RFID largement utilisé de nos jours, l'EM4450 de *EM Microelectronics*. Il s'agit d'un circuit RFID CMOS auto-alimenté, disposant d'une mémoire de 1 kBits pouvant être lue et écrite. Un tel circuit peut se retrouver dans des abonnements de transport en commun ou encore des cartes d'accès aux remontées-mécaniques des stations de ski pour ne citer que ces deux exemples. La porteuse du signal RF de ce circuit est de 125 kHz. Le circuit génère un signal d'horloge de période égale à la période de la porteuse. Selon le protocole de fonctionnement du tag, la lecture de l'entièreté de la mémoire, période d'initialisation comprise, nécessite 52256 cycles d'horloge. À une fréquence de 125 kHz, il faut donc 418 ms pour lire les 1024 bits de données, soit un débit moyen de 2.45 kBits/s. Ainsi, si l'on veut ajouter un coprocesseur de chiffrement AES à un tel tag, le coprocesseur

doit pouvoir fournir un débit de 2.45 kBits/s. Le coprocesseur AES synthétisé dans ce travail peut chiffrer 128 bits de données en 1142 cycles d'horloge, ce qui représente, à 125 kHz, un débit moyen de 14.1 kBits/s. Comme deuxième exemple, le protocole de communication développé dans [25] défini un temps de 18 ms disponible pour le chiffrement de 128 bits de données. Dès lors, le coprocesseur AES proposé, avec un nombre de cycles de 1142 par bloc de 128 bits, peut fonctionner à une fréquence de 100kHz tout en satisfaisant la contrainte.

Ainsi, dans le but de réduire la consommation au maximum, la fréquence de fonctionnement est réduite à 100kHz pour la deuxième barre de la Figure 27. La puissance statique n'est évidemment pas modifiée par une réduction de fréquence. Par contre, la puissance dynamique est fortement réduite (d'un facteur 100). La puissance totale vaut maintenant 1.14 μW. Nous l'avons vu dans le chapitre précédent, les trois composantes de la puissance sont réduites lorsque V_{dd} diminue. Ainsi, à V_{dd} =1.0V, la troisième barre présente une réduction de la puissance totale de 33% pour atteindre 0.76 μW. Les trois composantes sont réduites d'un même facteur, cela en concordance avec la Figure 19 qui présente une même pente pour les trois courbes entre 1.2V et 1.0V. On remarque la contribution de plus en plus importante de la puissance statique, dûe à la réduction de la fréquence. Sa part s'élève à 36% de la puissance totale. Pour réduire les courants de fuite, *STMicroelectronics* propose également une version à haut-V_t de sa technologie. Ainsi, la dernière barre de la Figure 27 présente les résultats de la synthèse avec la bibliothèque 65nm LPHVT [3] 1.0V. L'augmentation de V_t, en réduisant les courants de fuite I_{leak}, va permettre de réduire la puissance statique. Celle-ci est réduite de 88% et la puissance totale vaut désormais 0.5 μW.

Comme l'indique les slacks time sur la Figure 27 (caractères noirs), la réduction de la puissance s'accompagne d'une augmentation du délai. Entre 1.2V et 1.0V, le délai augmente de 24% en passant de 3.89 ns à 5.14 ns. Avec la bibliothèque haut-V_t, le délai augmente encore de 48% et atteint 9.87 ns. Cependant, la fréquence de 100kHz offre une période de 10000 ns. On voit donc que le délai du circuit peut encore largement augmenter tout en respectant la contrainte. La réduction de V_{dd} en dessous de 1.0V est donc possible. Pour la suite de ce travail, la technologie haut-V_t ne sera plus utilisée, à cause de l'augmentation trop importante du délai qu'elle impose. En effet, la réduction de V_{dd}, et donc de la puissance, sera vite limitée par cette augmentation du délai. Ceci à été vérifié en extrapolant pour le coprocesseur AES (à partir du comportement de l'inverseur haut-V_t) le V_{dd} minimum atteignable ainsi que la puissance correspondante. Cependant, cette estimation ne permet pas de conclure quant à la non-efficacité des transistors haut-V_t en régime sous-seuil. Cela doit être vérifié et pourrait être l'objet d'un travail futur. La section suivante présente l'évolution du coprocesseur AES lorsque V_{dd} diminue, obtenue grâce à l'usage de bibliothèques recaractérisées à basse tension.

3. Low Power High V_t, high-V_t =0.7V

4.3 Synthèse à très faible V_{dd}

Un des objectifs principaux de ce travail est de vérifier s'il est utile de synthétiser un circuit à l'aide d'une bibliothèque caractérisée à la tension cible d'alimentation du circuit ou non. Autrement dit, est-ce que la synthèse d'un circuit à l'aide d'une bibliothèque caractérisée à une certaine tension V_1 est aussi efficace que la synthèse du même circuit à l'aide d'une bibliothèque caractérisée à une tension V_2, V_2 étant plus petite que V_1, si ce circuit est ensuite alimenté à une tension V_2. De plus, nous aimerions également vérifier que le nombre et la taille des cellules présentes dans la bibliothèque peuvent influencer sur la consommation du circuit synthétisé. Enfin, nous voulons savoir jusqu'à quelle tension d'alimentation le coprocesseur AES est fonctionnel est quel sera le gain de consommatiton réalisé à cette tension par rapport à la tension nominale. Cette section tente de répondre à ces questions en présentant les résultats de synthèses du coprocesseur AES à l'aide de bibliothèques recaractérisées à différentes tensions V_{dd}. En guise d'avant goût, on peut se demander ce que deviendrait la consommation du coprocesseur en se basant uniquement sur les simulations de l'inverseur montrées précédemment. Partant des valeurs du délai et de la puissance du circuit synthétisé à 1.2V, il semblerait qu'une puissance de 52 nW peut être obtenue avec V_{dd} =0.3V, tout en respectant la contrainte posée par la fréquence de 100kHz puisque le délai serait de 9.2 μs. Nous verrons par la suite que cette estimation est assez réaliste.

Dans cette section, les puissances sont calculées par Power Compiler après annotation du facteur d'activité de chaque noeud. Ces facteurs d'activité ont été calculés en simulant, à l'aide de ModelSim, la description matérielle (en Verilog) du circuit obtenu après la synthèse. La simulation est réalisée pour vingt chiffrements avec des clés et blocs de données différents et aléatoires pour chaque chiffrement. Cette méthode permet d'obtenir des résultats plus précis [4].

4.3.1 Bibliothèque recaractérisée à basse tension

4.3.1.1 Choix des cellules

Toute les cellules de la bibliothèque standard de *STMicroelectronics* n'ont pas été recaractérisées. Seule une sélection de certaines de ces cellules font partie de la bibliothèque recaractérisée. En effet, d'après le travail de J.-M. Masgonty *et al.* [47], il est montré qu'un ensemble restreint et bien choisi de cellules résulte en une synthèse plus efficace. Les choix qui ont été fait sont les suivants :

4. Sans annotation, des facteurs d'activité par défaut de 0.5 sont insérés aux entrées du circuit et propagés vers les différents noeuds.

1. Le nombre de fonctions différentes est réduit au minimum. Ces fonctions ont été choisies en analysant le résultat d'une synthèse (bibliothèque ST LPSVT 1.0V) optimisée en contraignant la puissance. Les fonctions étant les plus utilisées sont sélectionnées pour la nouvelle bibliothèque.
2. La force des cellules est limitée. Vu que la contrainte temporelle n'est pas critique, les cellules les plus petites sont sélectionnées pour réduire la consommation. Ainsi, dans un premier temps (voir plus bas), la bibliothèque ne contient que la cellule la plus petite implémentant chaque fonction.
3. Les cellules présentant un fan-in plus grand que 2 sont éliminées. Cela permet d'éviter d'avoir des cellules avec trop de transistors empilés ou en parallèle, pour garantir un niveau logique de sortie et une marge de bruit suffisant à très faible V_{dd}, comme indiqué dans l'état de l'art et dans [48].

Ainsi, la bibliothèque recaractérisée (que nous appelons petite bibliothèque dans la suite de ce travail) contient les neuf cellules suivantes :

DFPQX4 : flip-flop

DFPHQX4 : flip-flop avec enable

DFPHQNX4 : flip-flop avec enable et sortie inversée

INVX2 : inverseur

NAND2X2 : NAND à 2 entrées

NOR2X2 : NOR à 2 entrées

OAI12X2 : NOR à 2 entrées dans NAND à 2 entrées

OAI21X2 : NOR à 2 entrées dans NAND à 2 entrées

MUXI21X2 : multiplexeur 2 :1 à sortie inversée

4.3.1.2 Influence de la taille des cellules

Nous venons d'indiquer au paragraphe précédent que, dans le but de réduire la consommation, seules les plus petites cellules ont été conservées. Cette affirmation n'est en réalité pas toujours correcte. La réduction sera effective uniquement à haut V_{dd}. Lorsque la contrainte temporelle est insignifiante, ce qui est le cas à haut V_{dd} pour le coprocesseur AES présenté ici à 100kHz, l'utilisation des plus petites cellules permet en effet de réduire la consommation grâce à la réduction des capacités. Cependant, quand la réduction de V_{dd} est telle que le délai devient proche de la contrainte, il est plus intérressant d'utiliser des cellules plus grandes pour certains noeuds, comme le prouve la Figure 28. La synthèse du coprocesseur AES à été réalisée avec la petite bibliothèque caractérisée à 0.3V ainsi qu'avec une seconde bibliothèque (que nous appelons bibliothèque moyenne dans la suite de ce travail)

qui contient, en plus des cellules citées précédement pour la petite bibliothèque, des cellules réalisant les mêmes fonctions mais de tailles plus grandes (deux à trois (petites) tailles par fonction, résultant en une bibliothèque de 27 cellules), toujours à 0.3V. La première barre (a) montre la répartition de puissance pour la petite bibliothèque. La puissance totale est de 37.7 nW. La deuxième barre (b) présente la répartition de puissance pour la biliothèque moyenne. La puissance totale est diminuée de 17% pour atteindre 31.3 nW. La principale différence vient de la réduction de la puissance interne. En effet, en ajoutant des cellules plus grandes à la bibliothèque, les temps de transition des noeuds les plus importants peuvent être diminués, ce qui va réduire les courants de court-circuit et donc la puissance interne. La puissance de commutation est par contre légèrement augmentée, à cause des charges plus importantes que présentent les cellules plus grandes. Enfin, on note une petite diminution de la puissance statique, car l'utilisation de cellules plus grandes permet de réduire le nombre de cellules utilisées et donc de diminuer les courants de fuites.

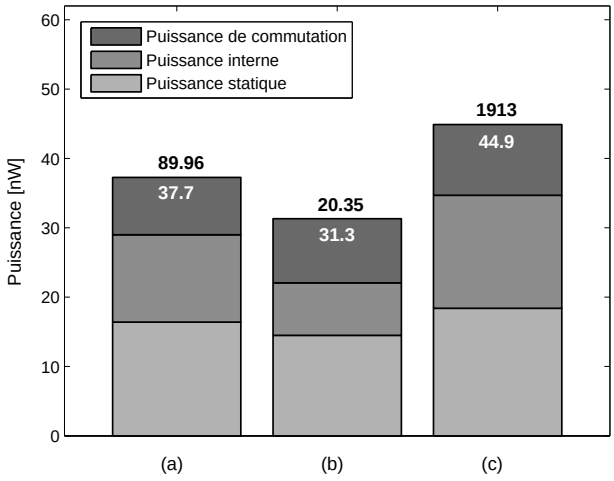


FIGURE 28: Répartition de la puissance du coprocesseur AES synthétisé avec (a) la petite bibliothèque caractérisée à 0.3V, (b) la bibliothèque moyenne caractérisée à 0.3V et (c) la petite bibliothèque caractérisée à 1.2V et circuit fonctionnant à 0.32V. La puissance totale est indiquée (caractères blancs) ainsi que le slack time [ns] (caractères noirs).

4.3.1.3 Influence de la bibliothèque recaractérisée à très faible V_{dd}

Comme annoncé plus haut, nous tenons à vérifier qu'il est bien nécessaire de réaliser la synthèse à la tension réelle de fonctionnement du circuit. Cela a donc été testé de la manière suivante. Le coprocesseur a été synthétisé avec la petite bibliothèque caractérisée à 1.2V. Ensuite, la puissance est calculée en ayant pris soin de remplacer toutes les cellules

par des cellules caractérisées à 0.3V (de manière à pouvoir calculer la puissance à cette tension et non à 1.2V). Cependant, après vérification du délai du circuit obtenu, il est apparu que le circuit ne respectait plus la contrainte temporelle à 0.3V. Il a donc fallu monter la tension V_{dd} à 0.32V. Le résultat est visible sur la barre (c) de la Figure 28. La puissance totale est de 44.9 nW. Comparée au circuit synthétisé directement à 0.3V (barre (a) de la figure), cela représente une augmentation de 19%. Cette augmentation est dûe principalement à l'augmentation de V_{dd} de 0.3V à 0.32V. Nous pouvons donc conclure que la synthèse du circuit est plus efficace si elle est réalisée à la tension d'alimentation cible de ce circuit.

4.3.2 Influence de la variabilité

Un des autres objectifs premiers de ce travail est d'estimer la consommation minimale du coprocesseur AES [5]. Comme on l'a vu, il s'agit de déterminer la tension minimale d'alimentation du circuit. La contrainte pour la réduction de V_{dd} est la vitesse. Nous voulons que le circuit puisse fonctionner à une fréquence de 100kHz. Ainsi, la tension V_{dd} minimum qui pourra faire fonctionner le circuit à 100kHz est la tension que nous recherchons. Pour choisir cette tension minimum, tout en étant sûr que le circuit soit fonctionnel dans tous les cas, il faut considérer la variabilité. Comme on l'a vu plus haut, celle-ci a un impact considérable sur le délai du circuit et son effet est d'autant plus important que la tension V_{dd} est faible. Dans un premier temps, nous avons considéré la variabilité globale. Le pire-cas que nous avons retenu pour le délai est une dérive du process de type SS et une température de -25˚C. L'évolution du délai de l'inverseur dans de telles conditions était présenté Figure 23 comparé au cas typique. On avait remarqué une augmentation d'un facteur 24 à V_{dd} =0.2V. À 0.3V, l'augmentation se fait encore d'un facteur 11.5. Dès lors, il faut augmenter V_{dd} de manière à compenser ce facteur. On remarque sur la Figure 17 qu'à 0.3V, une réduction du délai d'un facteur de 11.5 se fait approximativement en augmentant V_{dd} jusqu'a 0.4V. Cela sera confirmé par la synthèse du module AES entier. La Figure 29 présente les délais du coprocesseur après synthèse avec des bibliothèques caractérisées à différentes tensions.
En plus de la courbe du délai pour les conditions typiques, *corner* TT et 25˚C (marqueurs triangulaires), la figure présente le délai pour les conditions *corner* SS -25˚C pour des tensions V_{dd} aux alentours de 0.4V (marqueurs ronds). On voit que la contrainte fixée par la fréquence de 100kHz est respectée, pour les conditions pire-cas qu'à V_{dd} = 0.4V, alors qu'elle l'est à V_{dd} = 0.3V pour les conditions typiques.

Ensuite, la variabilité locale du coprocesseur AES a également été étudiée. Le chemin critique du circuit synthétisé a été extrait [6] et une simulation Monte-Carlo de 100 runs sur

5. En réalité, plutôt que de vouloir baisser à tout prix la consommation du circuit, on voudrait que celle-ci "rentre" dans le budget de puissance ou d'énergie fixé par les batteries ou le champ électro-magnétique qui alimentent le circuit.

6. Il contient 46 portes.

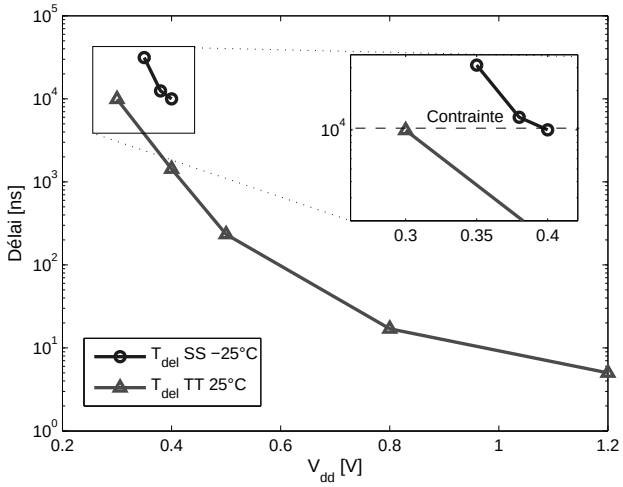


FIGURE 29: Délai après synthèse du coprocesseur AES pour des conditions typiques TT 25˚C (marqueurs triangulaires) et des conditions pire-cas SS -25˚C (marqueurs ronds).

le délai du chemin à été opérée, pour les conditions *corner* SS et température de -25˚C à différents V_{dd}. Par exemple, la Figure 30 présente la distribution du délai qui résulte de la simulation pour la charge et la décharge du noeud de sortie lorsque V_{dd} = 0.42V. À partir de ces résultats, il est possible de calculer le pire-cas à 3σ pour le délai du circuit. Pour V_{dd} = 0.4V, la variabilité locale implique une augmentation du délai d'un facteur de 1.77 (3σ). Comme le délai typique est de 9600ns à cette tension, la contrainte temporelle ne sera pas respectée. Ainsi, il faut monter V_{dd} à 0.42V pour garantir un délai inférieur à 10μs. À cette tension, le rapport entre délai pire-cas et délai typique est de 1.73 tandis que le délai typique vaut 5500ns. Le délai pire-cas est donc de 9515ns, ce qui est bien inférieur à 10μs.

4.3.3 Résultats

En condition typique, *corner* TT et température 25˚C, le coprocesseur AES est fonctionnel jusqu'à une tension V_{dd} de 0.3V (voir Figure 29, marqueurs triangulaires). Synthétisée avec la bibliothèque moyenne, la consommation du circuit est alors de 31.3 nW, comme présenté sur la Figure 31 (a). On voit que l'estimation faite au début de ce chapitre, basée sur l'évolution de l'inverseur, était relativement bonne. Pour rappel, une tension minimum de 0.3V était estimée avec une consommation de 52 nW. Cependant, si le circuit n'est équipé d'aucun mécanisme de mitigation des variabilités (telles le *adaptative voltage scaling* ou le *adaptative body biasing*, comme cité dans le Chapitre 2), le coprocesseur ne pourra pas fonctionner à une tension si basse. En considérant la variabilité globale, *corner* SS et tem-

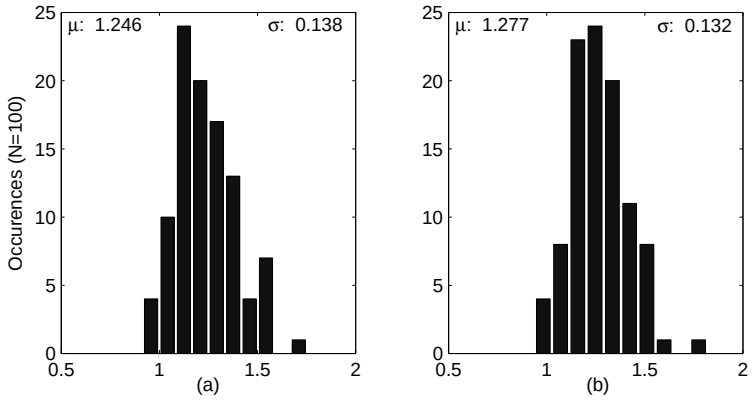


FIGURE 30: Distribution du délai du chemin critique du coprocesseur AES soumis à la variabilité locale pour les conditions SS -25˚C à V_{dd} = 0.42. Transition montante à gauche et descendante à droite.

pérature -25˚C, le coprocesseur AES sera fonctionnel jusqu'à une tension V_{dd} de 0.4V (voir Figure 29, marqueurs ronds). Cela implique une augmentation de la consommation de 84% pour atteindre 57.5 nW (toujours avec la biliothèque moyenne) (Figure 31 (b)).

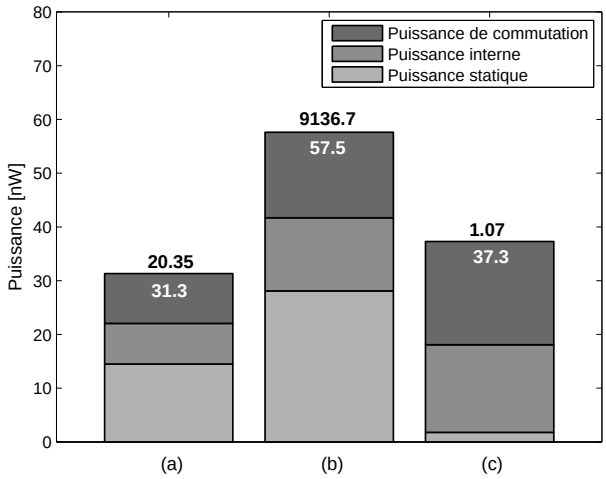


FIGURE 31: Consommation du coprocesseur AES synthétisé en conditions typiques à (a) 0.3V, (b) 0.4V et (c) en conditions pires-cas à 0.4V avec la bibliothèque moyenne. La puissance totale est indiquée en caractères blancs tandis que le *slack time* [ns] est indiqué en caractères noirs.

La dernière barre (c) de la Figure 31 présente la consommation du circuit, synthétisé avec la même bibliothèque que pour la barre (b), mais avec le circuit étant ici opéré effectivement en conditions pire-cas *corner* SS et température -25˚C cette fois-ci. On voit que la puissance statique est grandement diminuée. En effet, une telle variation de process et de température à

pour effet d'augmenter V_t et les courants de fuite sont donc réduits. Par contre, cela va aussi augmenter le délai (car le courant actif est réduit également) et donc la puissance interne va être plus importante. Au total, le circuit consomme 37.3 nW.

La Figure 32 présente les répartitions des puissances, en pourcentage, pour les différentes synthèses du coprocesseur réalisées. La courbe de la puissance totale est superposée à l'histogramme. Les observations suivantes peuvent être faites :

- La puissance totale diminue bien avec V_{dd} et présente une réduction d'un facteur 25 entre V_{dd} = 1.2V et V_{dd} = 0.3V (avec la biliothèque moyenne).
- La synthèse avec une même bibliothèque (petite ou moyenne) en condition typique présente une répartition semblable des puissances indépendamment de V_{dd}.
- A très faible V_{dd}, la synthèse avec la bibliothèque moyenne présente une puissance interne réduite comparée à la synthèse avec la petite bibliothèque pour un même V_{dd}.
- A haut V_{dd} (1.2V), la synthèse avec la bibliothèque moyenne présente une puissance (presque) identique comparée à la synthèse avec la petite bibliothèque pour un même V_{dd}. La différence observée ici est dûe à l'outil de synthèse qui n'optimise pas toujours le circuit de la même façon (ce qui a pu être vérifié).
- La puissance statique est grandement réduite en condition d'opération pire-cas.

Ces observations concluent ce travail.

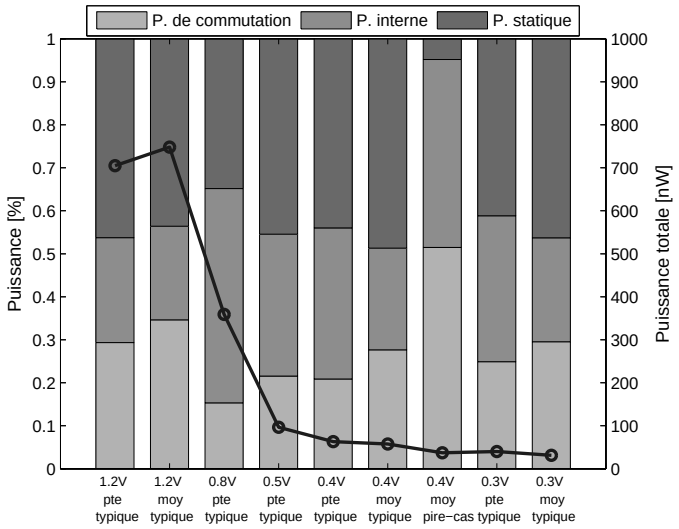


FIGURE 32: Délai après synthèse du coprocesseur AES pour des conditions typiques TT 25˚C (marqueurs triangulaires) et des conditions pire-cas SS -25˚C (marqueurs ronds).

4.4 Conclusion

Ce chapitre a présenté l'évolution, lors de la réduction de V_{dd}, du coprocesseur AES. Grâce à une petite bibliothèque, dont les cellules ont été choisies de manière optimale pour le fonctionnenent en régime sous-seuil, recaractérisée à différentes faibles tensions, le coprocesseur a pu être synthétisé et, la puissance et le délai ont pu être extraits. À la tension nominale de 1.2V, le coprocesseur consomme 1.14μW à 100kHz. Le grand slack time qui en résulte indique qu'une réduction de V_{dd} est possible. La tension minimum de fonctionnement du coprocesseur est de 0.3V et sa consommation chute à 31.3 nW. S'il faut tenir compte de la variabilité globale, une augmentation de V_{dd} à 0.4V est nécessaire. La consommation devient alors de 57.5 nW. Enfin, la variabilité locale demande une augmentation de la tension de 0.02V supplémentaire pour arriver à V_{dd} = 0.42V.

Il a aussi été démontré que la présence de cellules plus grandes dans la bibliothèque permet, à faible V_{dd}, de diminuer la puissance interne grâce à la réduction des courants de court-circuit résultant de la charge plus rapide des noeuds. Un gain total de puissance de 17% est montré. Aussi, l'intérêt de posséder une bibliothèque caractérisée à la tension de fonctionnement du circuit est expliqué. Synthétisé à 1.2V, le circuit présente une tension minimum de fonctionnement de 0.32V, avec une consommation de 44.9 nW. Cela est 19% de plus qu'avec la synthèse directement à 0.3V. La cause principale de cette augmentation est le passage indispensable de V_{dd} de 0.3V à 0.32V, de telle sorte que le circuit soit encore fonctionnel.

Chapitre 5

Conclusion

Ce travail aborde la synthèse de circuits logiques en régime sous-seuil dans une technologie 65nm. Spécifiquement, il met l'accent sur l'ultra faible consommation pouvant être obtenue grâce à la réduction de la tension d'alimentation. Le circuit sur lequel l'étude est réalisée est celui d'un coprocesseur AES pour tag RFID intelligent. À l'heure où les tags RFID connaissent un succès sans cesse grandissant, il devient important de veiller à limiter la consommation de leur circuit intégré. En effet, parmi les nombreuses applications utilisant la technologie RFID, plusieurs d'entre elles nécessitent que le tag, alors qualifié d'*intelligent*, puisse présenter certaines fonctionnalités plus avancées que la simple identification électronique. Ainsi, il faut désormais stocker et transmettre tout type de donnée via le tag. La présence d'un système de sécurisation des données est donc indispensable. L'algorithme de chiffrement AES est couramment utilisé à cette fin. Par ailleurs, le mode d'alimentation du tag RFID le contraint grandement quant à sa consommation. Pour que les nouvelles fonctionnalitées puissent être intégrées au circuit, il faut en réduire sa consommation. La diminution importante de la tension d'alimentation, exposée dans ce travail, permet une telle réduction.

Le coprocesseur AES à faible consommation a déjà été l'objet de précédents travaux. Une architecture itérative présentant un chemin de données étroit (8 bits) semble être la plus adaptée en vue d'une faible consommation. Cependant, aucune des implémentations présentées n'utilise la logique sous-seuil et, dans tous les cas, une technologie CMOS supérieure à 0.11μm est employée. Par ailleurs, les travaux concernant la logique sous-seuil présentent des gains de consommation importants et il apparait de plus que des technologies plus avancées permettent de réaliser des circuits optimaux en terme de consommation.

Partant de la bibliothèque de cellules standards 65nm de *ST Microelectronics*, différentes étapes permettent d'atteindre les objectifs fixés à priori. Ainsi, l'étude d'un inverseur de la bibliothèque a d'abord permis de quantifier l'évolution du délai et de la puissance lorsque V_{dd} diminue. Après, une méthode de caractérisation des cellules est développée et des nouvelles

bibliothèques sont créées en choisissant idéalement leurs cellules constitutives et en les caractérisant à différentes faibles tensions. Le coprocesseur AES est ensuite synthétisé avec ces nouvelles bibliothèques et les valeurs de délai et puissance sont rapportées, permettant de répondre aux objectifs, ce que nous pouvons faire maintenant.

Un gain de puissance de l'ordre de 19% est réalisable lorsque le circuit est synthétisé avec la bibliothèque caractérisée à la tension de fonctionnement. En effet, avec la bibliothèque caractérisée à la tension nominale, la synthèse produit un circuit dont la tension minimum de fonctionnement est plus élevée, impliquant une augmentation de puissance. Nous pouvons conclure qu'il est utile de posséder une bibliothèque caractérisée à la tension à laquelle le circuit doit être opéré lorsque la consommation doit être minimisée.
Une bibliothèque contenant des cellules de plus grande taille, permet, à faible V_{dd}, de synthétiser un circuit ayant une puissance interne réduite. Cela provient da la réduction des courants de court-circuit en conséquence de la (dé)charge plus rapide de certains noeuds plus importants. Le gain en puissance observé est de 17%.
À 100kHz, la tension minimum de fonctionnement du coprocesseur AES est de 0.3V. Avec une telle tension d'alimentation, la puissance est de 31.3 nW. S'il faut considérer la variabilité globale (process et température), cette tension doit être élévée à 0.4V et la puissance vaut alors 57.5 nW. Enfin, la prise en compte de la variabilité locale nécessite une tension d'alimentation de 0.42V pour que le circuit reste fonctionnel à 100kHz.

Finalement, plusieurs pistes d'améliorations ont été enrtevues dans ce travail et mériteraient d'être approfondies. Tout d'abord, l'estimation de la puissance peut être améliorée en effectuant le placement et routage du circuit synthétisé. Cela permettrait de tenir compte plus précisement des interconnexions lors du calcul de la puissance. Dans ce travail, ces interconnexions sont estimées grossièrement par l'outil de synthèse. Deuxièmement, la synthèse du circuit à l'aide des cellules à haut V_t, dont on a brièvement parlé, mériterait d'être étudiée. En effet, même si ces cellules à haut V_t ne permettent pas de diminuer la tension d'alimentation autant qu'avec des cellules standards, cela pourrait être compensé par l'importante réduction de la puissance statique. Enfin, on a vu que les cellules standards présentent des niveaux logiques de sortie et des marges de bruit réduites à très faible V_{dd}. Les effets de la variabilité sont aussi fort importants. Une optimisation des cellules, notamment un redimensionement correct de celles-ci, permettrait de réduire ces problèmes. Ces quelques suggestions, ainsi que les nombreuses publications que j'ai pu lire durant ces derniers mois, indiquent que bien du chemin peut encore être fait dans le domaine de la logique sous-seuil.

Bibliographie

[1] I. Verbauwhede, P. Schaumont, H. Kuo, "Design and Performance Testing of a 2.29-GB/s Rijndael Processor", Dept. of Electr. Eng., Univ. of California, Los Angeles, CA, USA, 2003

[2] "Specification fot the ADVANCED ENCRYPTION STANDARD (AES)", Federal Information Processing Standards Publication 197, 2001.

[3] Jörg J. Bucholz, "Matlab Implementation of the Advanced Encryption Standard", 2001

[4] A. Hodjat, I. Verbauwhede, "Speed-Area Trade-off for 10 to 100 Gbits/s Throughput AES Processor", Dept. of Electr. Eng., California Univ., Los Angeles, USA, 2003

[5] S. Morioka and A. Satoh, "A 10 Gbps full-AES crypto design with a twisted-BDD S-Box architecture", in *Proc. IEEE International Conference on Computer Design : VLSI in Computers and Processors*, pp. 98-103, 2002.

[6] P. Chodowiec, P. Khuon, K. Gaj, "Fast implementations of secret-key block ciphers using mixed inner- and outer-round pipelining", George Mason University, USA, 2001

[7] V. Rijmen, "Efficient implementation of the Rijndael S-box", 2001.

[8] A. Satoh, S. Morioka, K. Takano, S. Munetoh, "A compact Rijdael hardware architecture with S-Box Optimization", in *ASIACRYPT 2001, LNCS 2248*, pp. 239-254, 2001.

[9] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC implementation of the AES S-Boxes", in *Proceedings of the RSA Conference - Topics in Cryptography (CT-RSA)*, number 2271 in *Lecture Notes in Computer Science*, pp. 67-78, February 2002.

[10] N. Mentens, L. Batina, B. Preneel and I. Verbauwhede, "A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box", K.U. Leuven ESAT/COSIC, Belgium, 2005.

[11] D.Canright, "A very compact S-Box for AES", in *CHES 2005*, pp. 441-455, 2005.

[12] M. Macchetti and G. Bertoni, "Hardware implementation of the Rijndael Sbox : A case study", in *ST Journal of system research*, vol. 0, pp. 84-91, 2002.

[13] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Efficient implementation of Rijndael encryption with composite field arithmetic", in *Proceedings of 3rd International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, number 2162 in *Lecture Notes in Computer Science*, pp. 171-184, Paris, France, 2001.

[14] S. Morioka and A. Satoh, "An Optimized S-Box Circuit Architecture for Low Power AES Design", in *CHES 2002, LNCS 2523,* pp. 172-186, 2003.

[15] G. Bertoni, M. Macchetti, "Power-efficient ASIC Synthesis of Cryptographic Sboxes", in *GLSVLSI'04*, 2004.

[16] S. Tillich, M. Feldhofer, J. Großschädl, "Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box", in *SAMOS 2006*, number 4017 in *LNCS*, pp. 457-466, 2006.

[17] D. Kamel, F.-X. Standaert, D. Flandre, "Scaling trends of the AES S-Box low power consumption in 130 and 65 nm CMOS technology nodes",

[18] C. Paar, "Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields", PhD thesis, Institute for Experimental Mathematics, Universität Essen, Germany, 1994.

[19] B. Weeks, M. Bean, T. Rozylowicz, C. Ficke, "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms", National Security Agency white paper, 2000.

[20] T. Ichikawa, T. Kasuya, M. Matsui, "Hardware Evaluation of the AES Finalist", in *The Third AES Candidate Conference*, pp. 279-285, National Institute of Standards and Technology, 2000.

[21] H. Kuo, I. Verbauwhede, "Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm", in *CHES 2001*, number 2162 in *LNCS*, pp. 51-64, 2001.

[22] C.-C. Lu, S.-Y. Tseng, "Integrated Design of AES (Advanced Encryption Standard) Encrypter and Decrypter", in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'02)*, 2002.

[23] S. Mangard, M. Aigner, S. Dominikus, "A Highly Regular and Scalable AES Hardware Architecture", in *IEEE Transactions on Computers*, Vol. 52, No. 4, 2003.

[24] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, "A Methodology to Implement Block Ciphers in Reconfigurable Hardware and its Application to Fast and Compact AES RIJNDAEL", in *FPGA'03*, 2003.

[25] M. Feldhofer, S. Dominikus, J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm", in *CHES 2004*, in number 3156 of *LNCS*, pp. 357-370, 2004.

[26] S.-Y. Wu, S.-C. Lu, C. Sung Laih, "Design of AES Based on Dual Cipher and Composite Field", in *CT-RSA 2004, LNCS 2964*, pp. 25-38, 2004.

[27] M. Kim, J. Ryou, Y. Choi, S. Jun , "Low Power AES Hardware Architecture for Radio Frequency Identification", in *IWSEC 2006, LNCS 4266*, pp. 353-363, 2006.

[28] M. Feldhofer, J. Wolkerstorfer, V. Rijmen, "AES implementation on a grain of sand", in *IEE Proceedings Information Security*, vol. 152, pp. 13-20, 2005.

[29] E. Trichina, T. Korkishko, K. Hee Lee, "Small Size, Low Power, Side Channel-Immune AES Coprocessor : Design and Synthesis Results ", in *AES 2004, LNCS 3373*, pp. 113-127, 2005.

[30] P. Hämäläinen, T. Alho, M. Hännikäinen, T. D. Hämäläinen, "Design and Implementation of Low-area and Low-power AES Encryption Hardware Core", in *Proc. 9th Euromicro Conference on Digital System Design (DSD 2006), Cavtat, Croatia*, 2006.

[31] B. Zhai, D. Blaauw, D. Sylvester, K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling", in *Proceedings. 41st Design Automation Conference, 2004.* , 2004.

[32] B.H. Calhoun, A. Wang, A. Chandrakasan, "Modeling and Sizing for Minimum Energy Operation in Subthreshold Circuits", in *IEEE Journal of Solid-State Circuits,* Vol. 40, No. 9, pp. 1778-1786, 2005.

[33] A. Wang, A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology", in *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 310-319, 2005.

[34] D. Bol, "Pushing Ultra-Low-Power Digital Circuits into the Nanometer Era", Thèse de doctorat, UCL, 2008.

[35] J. Kwong, A. P. Chandrakasan, "Variation-Driven Device Sizing for Minimum Energy Sub-threshold Circuits", in *ISLPED'06*, 2006.

[36] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K.K. Das, W. Haensch, E.J. Nowak, D.M. Sylvester, "Ultralow-voltage, minimum-energy CMOS", in *IBM Journal of Research and Development*, Vol. 50, No. 4/5, pp. 469-490, 2006.

[37] Y.K. Ramadass, A.P. Chandrakasan, "Minimum Energy Tracking Loop with Embedded DC-DC Converter Delivering Voltages down to 250mV in 65nm CMOS", in *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 256-264, 2008.

[38] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, D. Blaauw, "Exploring Variability and Performance in a Sub-200-mV Processor", in *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 881-891, 2008.

[39] H. Kaul, M.A. Anders, S.K. Mathew, S.K. Hsu, A. Agarwal, R.K. Krishnamurthy, S. Borkar, "A 320 mV 56 μW 411 GOPS/Watt Ultra-Low Voltage Motion Estimation

Accelerator in 65 nm CMOS", in *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 107-114, 2009.

[40] J. T. Kao, M. Masayuki and A. P. Chandrakasan, "A 175-mV multiply-accumulate unit using an adaptive supply voltage and body bias architecture", in *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1545-1554, Nov. 2002.

[41] Y. Pu, J. Pineda, H. Corporaal, Y. Ha, "An Ultra Low-Energy/Frame Multi-standard JPEG Co-processor in 65nm CMOS with Sub/Near Threshold Power Supply", in *Proc. the ISSCC*, San Francisco, USA, Feb. 2009.

[42] H. Kaul, M.A. Anders, S.K. Mathew, S.K. Hsu, A. Agarwal, R.K. Krishnamurthy, S. Borkar, "A 300mV 494GOPS/W Reconfigurable Dual-Supply 4-Way SIMD Vector Processing Accelerator in 45nm CMOS", in *Proc. the ISSCC*, San Francisco, USA, Feb. 2009.

[43] D. Bol, D. Kamel, D. Flandre, J.-D. Legat, "Nanometer MOSFET Effects on the Minimum-Energy Point of 45nm Subthreshold Logic", in *ISLPED09*, 2009.

[44] D. Bol, D. Flandre, J.-D. Legat, "Technology Flavor Selection and Adaptive Techniques for Timing-Constrained 45nm Subthreshold Circuits", in *ISLPED09*, 2009.

[45] T. Sakurai, A.R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas", in *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 584-594, 1990.

[46] D. Bol, R. Ambroise, D. Flandre, J.-D. Legat, "Interests and Limitations of Technology Scaling for Subthreshold Logic", in *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, Vol. 17, 2009.

[47] J.-M. Masgonty, S. Cserveny, C. Arm, P.-D. Pfister, C. Piguet, "Low-Power Low-Voltage Standard Cell Libraries with a Limited Number of Cells", PATMOS 2001, 2001.

[48] N. Verma, J. Kwong, A.P. Chandrakasan, "Nanometer MOSFET Variation in Minimum Energy Subthreshold Circuits", in *IEEE Transactions on Electron Devices*, vol. 55, no. 1, 2008.

Annexe A

Advanced Encryption Standard

A.1 L'algorithme

L'algorithme Advanced Encryption Standard (AES), aussi connu sous le nom de algorithme de *Rijndael* (du nom de ses auteurs Joan Daemen et Vincent Rijmen) est le standard de chiffrement adopté par le gouvernement américain au début du siècle pour remplacer le DES (Data Encryption Standard). L'AES est un algorithme de chiffrement à clé symétrique. Il y a une clé unique commune à l'émeteur et au récepteur. La taille du bloc de données est de 128 bits et la taille de la clé peut être de 128, 192 ou 256 bits [1]. L'algorithme suit un schéma itératif. Il y a 10 itérations (pour une clé de 128 bits), appellées *round*. Durant chaque *round*, les mêmes opérations sont répétées sur le bloc de données. Ce bloc de données est représenté par une matrice 4×4 de 16 bytes appelée *Etat*. Toutes les opérations sont donc exécutées sur l'*Etat*. Les éléments de l'*Etat* sont des éléments du corps de Galois $GF(2^8)$ et sont notés par $S_{r,c}$ avec $(0 \leq r,c < 4)$. Le schéma de l'algorithme est représenté sur la Figure 33.

Chaque *round* consiste en une succesion de quatres opérations. L'opération *SubBytes*, l'opération *ShiftRows*, l'opération *MixColumns* et l'opération *AddRoundKey*. L'opération *AddRoundKey* est aussi exécutée préalablement au premier *round*. Enfin, le dernier *round* diffère des précédents en ce sens qu'il ne possède pas l'opération *MixColumns*. Toutes ces opérations sont décrites dans la section suivante.

1. La plupart des implémentations se limitent à une clé de 128 bits.

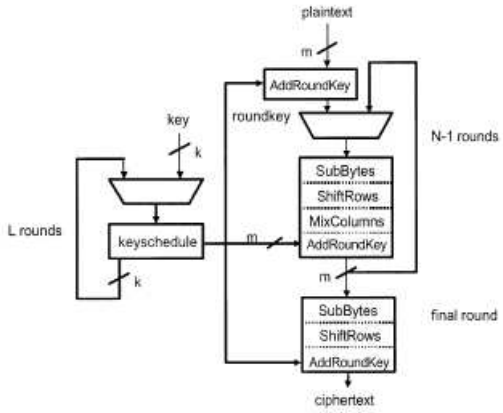


FIGURE 33: Schéma de l'algorithme AES (source : [1])

A.2 Les opérations

A.2.1 L'opération *SubBytes*

L'opération *SubBytes* (Figure 34) est une substitution non-linéaire de byte. Elle consiste à d'abord prendre l'inverse multiplicatif dans le corps $GF(2^8)$ et d'ensuite y appliquer une transformation affine définie par $b'_i = b_i \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$ avec c_i le i^{eme} bit du byte ayant pour valeur $\{63\}$. On appele S-Box la matrice de 256 éléments qui contient tous les éléments de $GF(2^8)$ sur lesquels l'opération *SubBytes* a été appliquée. Ainsi, l'opération peut être exécutée en choisissant correctement l'élément de la S-Box comme décrit par

$$S'_{r,c} = S[S_{r,c}] \tag{A.1}$$

où $S[x]$ est la fonction de substitution.

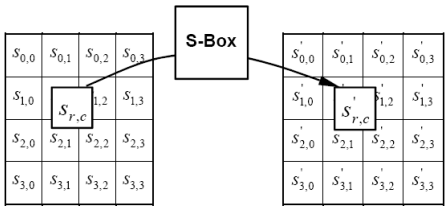


FIGURE 34: Opération *SubBytes* (source : [2])

A.2.2 L'opération *ShiftRows*

Durant l'opération *ShiftRows* (Figure 35), les lignes de l'*Etat* sont cycliquement décallées vers la gauche. Le décallage va de 0 à 3 bytes en fonction du numéro de la ligne. L'opération

est :

$$S'_{r,c} = S_{r,(c+r)mod4} \tag{A.2}$$

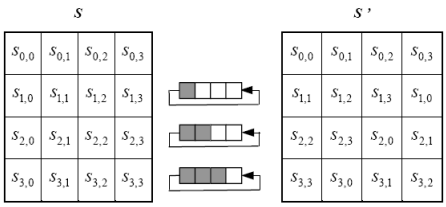


FIGURE 35: Opération *ShiftRows* (source : [2])

A.2.3 L'opération *MixColumns*

L'opération *MixColumns* (Figure 36) est une opération qui s'éxécute colonne par colonne sur l'*Etat* et qui est décrite par le produit suivant :

$$\begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}' = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{A.3}$$

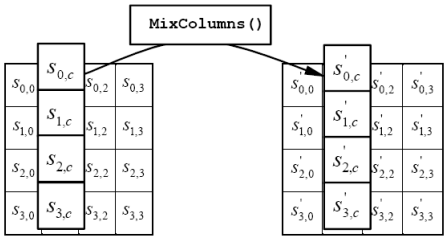


FIGURE 36: Opération *MixColumns* (source : [2])

A.2.4 L'opération *AddRoundKey*

L'opération *AddRoundKey* (Figure 37) réalise une addition colonne par colonne de l'*Etat* avec une clé de *round*. L'opération est définie par :

$$\begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}' = \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \oplus \left[w_{(round*4)+c} \right] \tag{A.4}$$

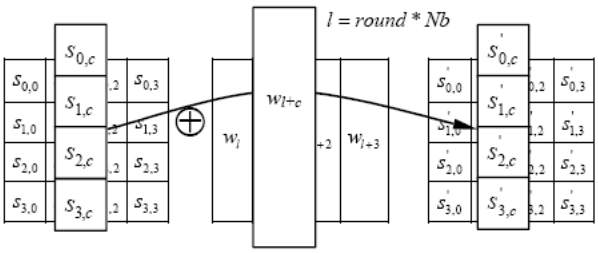


FIGURE 37: Opération *AddRoundKey* (source : [2])

A.2.5 Les clés de *round*

Les clés de *round*, qui sont différentes pour chaque *round*, sont dérivées de la clé originale. Les clés de *round* sont produites par le *KeyScheduling* et sont stockées ligne par ligne dans un tableau. Le schéma suivi pour la production de ces clés est montré Figure 38. Les clés sont créées ligne par ligne. Une nouvelle ligne est formée en additionnant deux lignes précédentes, celle qui la précède directement et celle quatre lignes avant. Les quatres premières lignes sont équivalentes à la clé originale. Toutes les quatres lignes, la ligne qui précède directement est d'abord décallée ensuite substituée, via la S-Box, et enfin additionnée à une constante de *round* avant d'être utilisée pour produire la clé. La constante de *round* est un élément de 32 bits dont le premier byte est une puissance de 2 et les autres sont nuls.

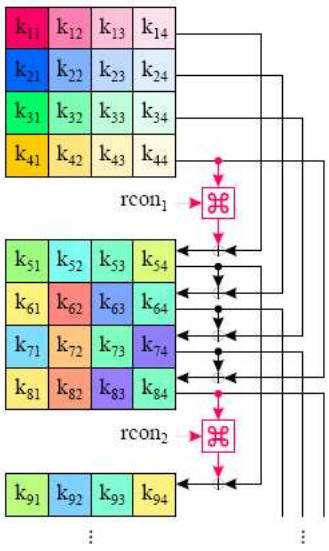


FIGURE 38: Génération des clés de *round* (source : [3])

Enfin, rappellons que les opérations sont effectuées dans le corps $GF(2^8)$ et que le résultat de ces opérations doit toujours appartenir à ce corps. C'est pourquoi, les addition sont éxécutées à l'aide de XOR i.e. modulo 2. De la même manière, une multiplication doit être suivie par une opération modulo $m(x)$ avec $m(x) = x^8 + x^4 + x^3 + x + 1$ comme polynôme irréductible.

Le processus de déchiffrement est très similaire. Il fonctionne selon le même schéma mais avec les opérations inverses éxécutées dans le sens inverse.

Annexe B

Méthode de caractérisation

B.1 Introduction

La présente annexe décrit, sous la forme d'un tutoriel, la méthode permettant de caractériser les cellules telle qu'utilisée dans ce travail. Le schéma général de la caractérisation est présenté à la Figure 39. Elle se fait à l'aide de l'outil Liberty NCX de Synopsys. Ce dernier permet de générer des bibliothèques de cellules au format standard Liberty (`.lib`).

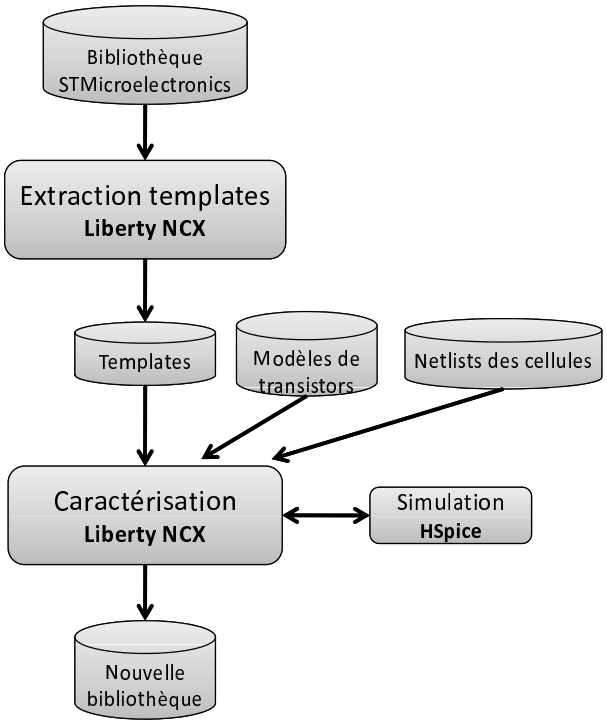


FIGURE 39: Flot de caractérisation d'une bibliothèque de cellules à l'aide de Liberty NCX de Synopsys.

Les étapes à suivre pour réaliser la caractérisation sont :

- Extraction à partir d'une bibliothèque existante, ou création manuelle, des templates des cellules à caractériser qui contiennent les descriptions fonctionnelles de celles-ci ainsi que des commandes propres à Liberty NCX pour réaliser la caractérisation.
- Caractérisation des cellules à l'aide d'un simulateur externe, sur base des templates, des modèles de transistors utilisés par le simulateur et des netlists des cellules à caractériser.
- Compilation de la bibliothèque au format `.db`.

Les éléments nécessaires pour mener à bien la caractérisation sont :

- Une bibliothèque existante à partir de laquelle les templates vont être créés, dans le cas d'une re-caractérisation.
- Des templates de cellules, créés manuellement, dans le cas où il ne s'agit pas d'une re-caractérisation.
- Les fichiers décrivant les modèles d'éléments de circuit utilisés par le simulateur (transistors, diodes,...).
- Les netlists, propres au simulateur utilisé, des cellules à caractériser.

Dans ce tutoriel, la marche à suivre complète est détaillée et les différents scripts nécessaires sont montrés. Par souci de simplicité, les exemples montrés se rapportent à la caractérisation d'une bibliothèque ne contenant qu'un inverseur, mais bien sûr, la méthode est identique pour une bibliothèque complète. Le simulateur utilisé en combinaison avec Liberty NCX est HSpice de Synopsys. Les exemples montrés se rapportent donc à ce simultateur.

B.2 Extraction des templates

Le but étant de re-caractériser les cellules de *STMicroelectronics* à une tension plus faible, nous utilisons la bibliothèque de *STMicroelectronics* comme fichier `.lib` source permettant de créer les templates. L'extrait de la bibliothèque 65nm de *STMicroelectronics* contenant la cellule *HS65_LS_IVX2* (il s'agit du plus petit inverseur) et caractérisée à 1.2V, est présentée au Fragment 1 [1]. Les paramètres et données non-indispensables à la bonne compréhension de ce tutoriel sont supprimés (remplacés par ...).

1. Remarquons ici qu'une port `vdds` apparait dans la bibliothèque. Il s'agit de l'alimentation du body de la cellule qui est séparée de l'alimentation principale. Lors de la re-caractérisation, nous avons éliminé ce port en connectant le body sur l'alimentation principale.

FRAGMENT 1 : Description de l'inverseur telle qu'apparaissant dans la bibliothèque *STMicroelectronics* (fichier `CORE65LPSVT.lib`).

```
/*****************************************************
 Synopsys Technology File

 Time generated:                    Fri Jun  9 14:46:32 2006

 Process values given:
   Library nominal:                 1

 Voltage values given:
   Library nominal:                 1.2

 Temperature values given:
   Library nominal:                 25

 *****************************************************/

 library(CORE65LPSVT){
        delay_model : table_lookup;
  ...

/*****************************************************/

  nom_process : 1;
  nom_voltage : 1.2;
  nom_temperature : 25;

/*****************************************************/

  time_unit : "1ns";
  voltage_unit : "1V";
  current_unit : "1mA";
  pulling_resistance_unit : "1kohm";
  capacitive_load_unit(1,\
          pf);

/*****************************************************/

  default_cell_leakage_power : 2.74352e-08;
  default_connection_class : "default";
  default_fanout_load : 1;
  default_inout_pin_cap : 0.01;
  default_input_pin_cap : 0.01;
  default_leakage_power_density : 1.0552e-08;
  default_max_fanout : 20;
  default_output_pin_cap : 0;
```

```
  default_wire_load_mode : "enclosed";
  ...
  input_threshold_pct_fall : 60;
  input_threshold_pct_rise : 40;
  leakage_power_unit : "1mW";
  output_threshold_pct_fall : 60;
  output_threshold_pct_rise : 40;
  simulation : true;
  slew_derate_from_library : 1;
  slew_lower_threshold_pct_fall : 20;
  slew_lower_threshold_pct_rise : 20;
  slew_upper_threshold_pct_fall : 80;
  slew_upper_threshold_pct_rise : 80;
  smr : "1.1";

/*****************************************************/

  power_supply(){
   default_power_rail : vdd;
   power_rail(vdd, 1.2);
   power_rail(vdds, 1.2);
  }
  operating_conditions(nom_1.20V_25C){
   process : 1;
   temperature : 25;
   tree_type : "balanced_tree";
   voltage : 1.2;
   power_rail(vdd, 1.2);
   power_rail(vdds, 1.2);
  }

/*****************************************************/
        wire_load(area_0Kto1K){
   area : 0;
   capacitance : 0.8;
   fanout_length(1,0.00093825)
   fanout_length(2,0.0018765)
   fanout_length(3,0.00281475)
   fanout_length(4,0.003753)
   fanout_length(5,0.00469125)
   fanout_length(6,0.0056295)
   fanout_length(7,0.00656775)
   fanout_length(8,0.007506)
   fanout_length(9,0.00844425)
   fanout_length(10,0.0093825)
   fanout_length(11,0.0103208)
   fanout_length(12,0.011259)
```

```
   fanout_length(13,0.0121972)
   fanout_length(14,0.0131355)
   fanout_length(15,0.0140738)
   resistance : 5.82357;
   slope : 0.00093825;
  }
  ...

  power_lut_template(power_table_1){
   index_1(" 0.005, 0.12, 0.24, 0.47 ");
   index_2(" 0.0014, 0.21 ");
   variable_1 : input_transition_time;
   variable_2 : total_output_net_capacitance;
  }
  ...

  lu_table_template(table_1){
   index_1(" 0.005, 0.12, 0.24, 0.47 ");
   index_2(" 0.0014, 0.013, 0.026, 0.053, 0.11, 0.21 ");
   variable_1 : input_net_transition;
   variable_2 : total_output_net_capacitance;
  }
  ...

cell(HS65_LS_IVX2){
   area : 1.56;
   cell_leakage_power : 6.94009e-08;
   ...
   leakage_power(){
    power_level : "vdd";
    value : 1.0683e-07;
    when : "!A";
   }
   leakage_power(){
    power_level : "vdd";
    value : 2.7429e-08;
    when : "A";
   }
   leakage_power(){
    power_level : "vdd";
    value : 6.71295e-08;
   }
   ...
   pin(A){
    capacitance : 0.000892;
    direction : "input";
    ...
```

```
 fall_capacitance : 0.000864;
 fall_capacitance_range(0.000705,0.001094);
 input_signal_level : vdd;
 max_transition : 0.47;
 rise_capacitance : 0.000919;
 rise_capacitance_range(0.000724,0.001094);
}
pin(Z){
 capacitance : 0;
 direction : "output";
 ...
 fall_capacitance : 0;
 fall_capacitance_range(0,0);
 function : "!A";
 max_capacitance : 0.035;
 output_signal_level : vdd;
 rise_capacitance : 0;
 rise_capacitance_range(0,0);
 internal_power(){
  power_level : "vdd";
  related_pin : "A";
  fall_power(power_table_7){
   values("-1.769915e-04, -1.464715e-04",\
        "-2.621714e-04, -1.932615e-04",\
        "-1.283615e-04, -1.960515e-04",\
        "2.403386e-04, -1.474814e-04");
  }
  rise_power(power_table_7){
   values("1.661759e-03, 1.717659e-03",\
        "1.623959e-03, 1.714659e-03",\
        "1.771259e-03, 1.716659e-03",\
        "2.151859e-03, 1.771659e-03");
  }
 }
 ...
 timing(){
  ...
  intrinsic_fall : 0.009319;
  intrinsic_rise : 0.010976;
  related_pin : "A";
  timing_label : "A_Z";
  timing_sense : negative_unate;
  cell_fall(table_6){
   values("0.015185, 0.027756, 0.046045, 0.084014, 0.154420",\
        "0.059220, 0.083381, 0.109620, 0.154830, 0.228980",\
        "0.088914, 0.123970, 0.158890, 0.214550, 0.298730",\
        "0.133930, 0.184990, 0.235010, 0.308840, 0.412160");
```

```
    }
    cell_rise(table_6){
     values("0.018832, 0.035666, 0.060115, 0.111060, 0.205030",\
         "0.066128, 0.094405, 0.126500, 0.183650, 0.280240",\
         "0.100520, 0.138820, 0.179620, 0.247480, 0.353590",\
         "0.155310, 0.208620, 0.263320, 0.349250, 0.475030");
    }
    fall_transition(table_6){
     values("0.015621, 0.037357, 0.069158, 0.135800, 0.258610",\
         "0.037651, 0.058423, 0.087681, 0.148780, 0.265160",\
         "0.059342, 0.082724, 0.112670, 0.173730, 0.285850",\
         "0.095164, 0.125930, 0.159140, 0.221600, 0.333950");
    }
    rise_transition(table_6){
     values("0.022167, 0.052919, 0.098224, 0.192200, 0.366190",\
         "0.044289, 0.073743, 0.115100, 0.202440, 0.370800",\
         "0.065103, 0.097706, 0.140590, 0.227080, 0.388590",\
         "0.099482, 0.139100, 0.186410, 0.276180, 0.435460");
    }
   }
  }
 }
}
```

Détaillons les éléments importants :

- Le groupe `library()`, obligatoire, défini la bibliothèque. La description complète de la bibliothèque doit se trouver dans ce groupe. Certains paramètres propres à la bibliothèque ou communs à toutes les cellules sont ensuite inscrits tels que les unités, les seuils pour le calcul des délais, le modèle de délai, les conditions d'opération, les modèles d'interconnexion ou encore les indexes des tables de correspondance.
  - Le groupe `cell()` défini une cellule. Il y a autant de groupes `cell()` que de cellules dans la bibliothèque. Les paramètres généraux de la cellule débutent le groupe comme la superficie ou encore des valeurs par défaut de capacité ou de puissance.
    - Le groupe `leakage_power()` donne la puissance statique en fonction de l'état d'entrée de la cellule. Si la cellule comprend plus d'une entrée, il y a autant de groupes `leakage_power()` que de combinaisons des valeurs des signaux d'entrée possibles.
    - Le groupe `pin()` défini un port d'une cellule. On y retrouve principalement la direction du port et la capacité ou la fonction selon que ce soit un port d'entrée ou de sortie. Il y a autant de groupes `pin()` que de ports à la cellule.
      - Le groupe `internal_power()` donne les tables de correspondance pour la

puissance interne relatif à un port d'entrée pour le port de sortie en question. Il y a autant de groupes `internal_power()` pour un port de sortie qu'il y a de ports d'entrée pouvant provoquer un changement d'état de ce port de sortie. Les indexes des tables sont ceux définis plus haut dans la bibliothèque.

– Le groupe `timing()` donne les délais et temps de transition du port de sortie en question. Il y a autant de groupe `timing()` qu'il y a de "timing arcs" pour ce port de sortie. Les indexes des tables sont ceux définis plus haut dans la bibliothèque.

L'extraction du template pour l'inverseur (et pour la bibliothèque) se fait en invoquant Liberty NCX avec certaines options, situées dans un fichier `template_cmd` dédié, via la commande :

```
> ncx -f template_cmd
```

Le fichier `template_cmd` contient :

```
> set input_library old_lib/CORE65LPSVT.lib
> set output_template_dir template
> set templates true
> set netlist_file CORE65LPSVT.spc
> set netlist_dir netlists
> set farm_type no_farm
```

Les deux premières commandes définissent le chemin vers la bibliothèque d'entrée et le dossier de sortie. La troisième permet d'activer l'extraction des templates. Les quatrième et cinquième commandes indiquent le fichier contenant l'ensemble des netlists ainsi que le répertoire dans lequel les netlists séparées seront déposées [2]. Enfin, la dernière désactive le calcul partagé (qui est activé par défaut).

Le template ainsi généré par Liberty NCX se compose de trois fichiers. Le premier est le template général de la bibliothèque (voir Fragment 2), le deuxième est celui de la cellule [3] (voir Fragment 3) et le dernier contient les indexes des tables de correspondance (voir Fragment 4). Comme on peut le voir, les templates ont plus ou moins la même structure que les bibliothèques. Cependant, aucune valeur ne s'y retrouve (mis à part certaines valeurs par défaut) et certaines commandes propres à Liberty NCX peuvent être rajoutées à différents endroits. Parmi celles-ci, deux sont particulièrement importantes. Elles sont mises en évidence dans le template de la bibliothèque (caractères gras). La première, `switching_power_split_model : false ;`, indique à Liberty NCX la manière dont il doit calculer la puissance interne. En mettant l'option à "false", la puissance interne est

2. En plus de générer les templates, Liberty NCX segmente le fichier contenant l'ensemble des netlists en plusieurs netlists séparées pour chaque cellule.

3. Il y a un fichier séparé pour chaque cellule de la bibliothèque, ici seul l'inverseur est présenté.

calculée comme la puissance totale moins la puissance de commutation, comme décrit au Chapitre 3. La deuxième commande, `clk_width : 1e-5 ;`, permet de spécifier le temps minimum entre deux changements du signal d'entrée pendant la simulation des cellules. Il est particulièrement important de mettre une valeur suffisamment élevée (au détriment de temps nécessaire à la caractérisation), surtout à faible V_{dd}, sous peine d'obtenir des résultats erronés [4], ou même de ne pas pouvoir caractériser une cellule. On remarque également à la fin une commande permettant d'inclure le fichier contenant les indexes des tables de correspondance à créer et enfin, les groupes `do{}` et `dont{}` qui permettent de spécifier les cellules à caractériser ou à ne pas caractériser.

Concernant le template de l'inverseur, seules des commandes sont rajoutées pour spécifier quels indexes utiliser pour créer les tables de correspondance, indexes qui se trouvent dans le fichier inclus.

Les indexes des tables de correspondance sont extraits tel quel de la bibliothèque originale. Cependant, en vue d'une re-caractérisation à faible V_{dd}, il faut modifier manuellement les index des temps de transition (temps de transition beaucoup plus élevés à faible V_{dd}). Liberty NCX peut également créer les indexes des tables automatiquement à partir des valeurs minimum et maximum de ces indexes ainsi que le nombre d'élément voulu dans la table, et ce de manière linéaire ou logarithmique.

FRAGMENT 2 : Template de la bibliothèque (fichier `AESlib_0_3V_rechar.opt`).

```
* Generated by Liberty NCX vZ-2007.09-SP1
delay_model : table_lookup ;
...
nom_voltage : 0.3000000 ;
nom_temperature : 25.0000000 ;
time_unit : 1ns ;
voltage_unit : 1V ;
current_unit : 1mA ;
pulling_resistance_unit : 1kohm ;
capacitive_load_unit : 1.0000000 pf ;
...
input_threshold_pct_fall : 60.0000000 ;
input_threshold_pct_rise : 40.0000000 ;
leakage_power_unit : 1mW ;
output_threshold_pct_fall : 60.0000000 ;
output_threshold_pct_rise : 40.0000000 ;
...
```

4. Par exemple, en analysant de plus près les fichiers de simulation générés par Liberty NCX, on s'appercoit que la puissance statique d'une cellule n'est pas calculée à l'aide d'une simulation DC mais à l'aide d'une simulation transitoire. Dès lors, il faut veiller à ce que le temps soit suffisamment long après une transition sous peine de mesurer une puissance statique trop élevée.

```
slew_lower_threshold_pct_fall : 20.0000000 ;
slew_lower_threshold_pct_rise : 20.0000000 ;
slew_upper_threshold_pct_fall : 80.0000000 ;
slew_upper_threshold_pct_rise : 80.0000000 ;
...
default_wire_load : wl_zero ;
default_wire_load_selection : default_by_area ;

fast_mode : false ;
switching_power_split_model : false ;
clk_width: 1e-5 ;
tran_timestep : 1e-9;
ncx_constraint_search_interval : 3000 ;
ncx_constraint_accuracy : 0.1;

power_supply  {
  default_power_rail : vdd ;
  power_rail : vdd 0.3000000 ;
}
operating_conditions nom_0.30V_25C {
  process : 1.0000000 ;
  temperature : 25.0000000 ;
  tree_type : balanced_tree ;
  voltage : 0.3000000 ;
  power_rail : vdd 0.3000000 ;
}
wire_load ... {
 ...
}

include AESlib_0_3V_rechar.indexes ;

do {
  H65_LS_IVX2
}

dont {
  ...
}
```

FRAGMENT 3 : Template de l'inverseur (fichier `h65_ls_ivx2.opt`).

```
* Generated by Liberty NCX vZ-2007.09-SP1
area : 1.5600000 ;
...
rail_connection : vdd vdd ;
ncx_leakage_power_when : !A A ;
```

```
pin A {
  direction : input ;
  ...
  input_signal_level : vdd ;
}
pin Z {
  direction : output ;
  ...
  function : !A ;
  output_signal_level : vdd ;
  ncx_internal_power_rise_input_transition_time_index : 1 ;
  ncx_internal_power_fall_input_transition_time_index : 1 ;
  ncx_internal_power_rise_total_output_net_capacitance_index : 3 ;
  ncx_internal_power_fall_total_output_net_capacitance_index : 3 ;
  ncx_rise_input_net_transition_index : 1 ;
  ncx_fall_input_net_transition_index : 1 ;
  ncx_rise_total_output_net_capacitance_index : 2 ;
  ncx_fall_total_output_net_capacitance_index : 2 ;
}
```

FRAGMENT 4 : Indexes des tables de correspondance (fichier `AESlib_0_3V_rechar.indexes`).

```
* Generated by Liberty NCX vZ-2007.09-SP1

ncx_input_transition_time_index : 14 336 672 ;
ncx_input_transition_time_index_1 : 14 336 672 1316 ;

ncx_constrained_pin_transition_index : 14 336 672 ;

ncx_input_net_transition_index : 14 336 672 ;
ncx_input_net_transition_index_1 : 14 336 672 1316 ;

ncx_related_pin_transition_index : 14 336 672 ;

ncx_total_output_net_capacitance_index : 0.0014 0.009 0.018 \
 0.035 0.07 ;
ncx_total_output_net_capacitance_index_1 : 0.0014 0.07 ;
ncx_total_output_net_capacitance_index_2 : 0.0014 0.0044 0.0088 \
 0.018 0.035 ;
ncx_total_output_net_capacitance_index_3 : 0.0014 0.035 ;
```

B.3 Caractérisation de la bibliothèque

Une fois les templates créés, la bibliothèque peut être caractérisée. Mais pour cela, certains fichiers supplémentaires sont nécessaires. Tout d'abord, il faut disposer des modèles d'éléments de circuit ainsi que des netlists des cellules que le simulateur va utiliser. Le Fragment 5 présente la netlist de l'inverseur. Cette netlist provient de *STMicroelectronics* et contient les éléments parasites de la cellule.

FRAGMENT 5 : Netlist de l'inverseur (fichier `HS65_LS_IVX2.spc`).

```
.SUBCKT HS65_LS_IVX2 A Z vdd vss
 *COMMENT  Extracted with DK_cmos065lpgp_7m4x0y2z_50A28A@4.1@20060317.0
 *at 08:13 18 Apr  2006
 Xld_D0 vss vdd DNWPS AREA=5.20365 PJ=9.53
 XMM64 Z A vss vss NSVTLP AD=0.041 AS=0.041 L=0.06 PD=0.61 PS=0.61 \
  W=0.2 lpe=1 po2act=0.205
 XMM65 Z A vdd vdd PSVTLP AD=0.0574 AS=0.0574 L=0.06 PD=0.69 PS=0.69 \
  W=0.28 lpe=1 po2act=0.205
 C14 vdd A 3.93903e-17
 C16 Z vdd 4.18569e-17
 C17 Z vss 1.73068e-16
 C18 Z vss 2.01431e-17
 C19 Z A 1.39162e-16
 C2 vss A 2.78265e-17
 C5 vdd A 7.89036e-18
 C9 vss A 9.15628e-17
 Cg1 A vss 2.17049e-17
 Cg10 vss vss 1.19838e-16
 Cg15 vdd vss 4.46402e-18
 Cg20 Z vss 2.14667e-17
 Cg3 vss vss 5.41575e-17
 Cg6 vdd vss 8.14917e-17
.ENDS HS65_LS_IVX2
```

Le Fragment 6 présente le fichier `model` qui contient les liens vers les modèles utilisés par le simulateur. Ce fichier est inclus à chaque fichier de simulation nécessaire à la caractérisation des cellules.

FRAGMENT 6 : Modèles utilisés par le simulateur (fichier `model`).

```
* libraries
.lib "/users/chocquet/libertyncx/spice_lib/LPmos_bsim4_svt.lib" svtlp_TT
.lib "/users/chocquet/libertyncx/spice_lib/common_poly.lib" PRO_TT
.lib "/users/chocquet/libertyncx/spice_lib/common_active.lib" PRO_TT
```

```
.lib "/users/chocquet/libertyncx/spice_lib/common_go1.lib" PRO_TT
.lib "/users/chocquet/libertyncx/spice_lib/diodeiso.lib" diodeiso_typ

.param svtlp_dev = 0
```

Comme pour l'extraction des templates, Liberty NCX est invoqué avec un fichier de commandes mais propre à la caractérisation cette fois-ci. Le fichier `first_char_cmd` contient :

```
> set netlist_dir netlists
> set input_template_dir template
> set work_dir work
> set library_template_file AESlib_0_3V_rechar
> set output_library new_lib/AESlib_0_3V_rechar.lib
> set model_file spice/model
> set simulator_exec /dir/DICE/synopsys/hspice2008/hspice/bin/hspice
> set farm_type no_farm
> set timing true
> set ccs_timing false
> set nldm true
> set power true
> set ccs_power false
> set nlpm true
```

Les trois premières commandes définisent les répertoires contenant les netlists des cellules, les templates et un répertoire de travail temporaire. Les deux commandes suivantes indiquent le nom du template de la bibliothèque à caractériser suivi du nom de la nouvelle bibliothèque de sortie. Le fichier contenant les modèles à inclure pour la simulation est ensuite donné. La septième commande permet d'indiquer quel simulateur sera utilisé en spécifiant son chemin d'accès. Enfin, les six dernières commandes indiquent à Liberty NCX qu'il doit caractériser les cellules en temps et puissance en utilisant les modèles non-linéaires et sans inclure les modèles de type *Composite Current Source*.

La caractérisation est lancée à l'aide de la commande :

```
> ncx -f first_char_cmd
```

Le Fragment 7 présente la bibliothèque re-caractérisée. Ici, aucune donnée n'a été supprimée (mis à part les modèles d'interconnexion et les tables de correspondance).

FRAGMENT 7 : Bibliothèque re-caractérisée à 0.3V (fichier `AESlib_0_3V_rechar.lib`).

```
/*
 * Generated by Liberty NCX vB-2008.12-SP2 on Fri Mar 20 14:16:04 2009
 */
library ("AESlib_0_3V_rechar") {
 define(drc_pg_arcs,library,string);
 define(smr,library,string);
 define(switching_power_split_model,library,string);
 define(fast_mode,library,string);
 define(clk_width,library,string);
 define(tran_timestep,library,string);
 define(driver_model,library,string);
 define(def_sim_opt,library,string);
 define(simulator,library,string);
 define(drc_blockgroup,cell,string);
 define(drc_blocktype,cell,string);
 define(drc_celltype,cell,string);
 define(drc_ground_pin_drives,cell,string);
 define(drc_ground_pins,cell,string);
 define(drc_maxparallel,cell,string);
 define(drc_power_pin_drives,cell,string);
 define(drc_power_pins,cell,string);
 define(drc_restriction,cell,string);
 define(drc_pindrive,pin,string);
 define(drc_pinfunction,pin,string);
 define(drc_pinsigtype,pin,string);
 technology("cmos");
 delay_model : "table_lookup";
 library_features("report_delay_calculation");
 nom_process : 1.000000;
 nom_voltage : 0.300000;
 nom_temperature : 25.000000;
 time_unit : "1ns";
 voltage_unit : "1V";
 current_unit : "1mA";
 pulling_resistance_unit : "1kohm";
 capacitive_load_unit(1.000000, \
                "pf");
 default_connection_class : "default";
 default_fanout_load : 1.000000;
 default_inout_pin_cap : 0.010000;
 default_input_pin_cap : 0.010000;
 default_max_fanout : 20.000000;
 default_output_pin_cap : 0.000000;
 default_wire_load_mode : "enclosed";
 define_cell_area("bond_pads", \
                "pad_slots");
```

```
drc_pg_arcs : "vdd:gnd";
input_threshold_pct_fall : 60.000000;
input_threshold_pct_rise : 40.000000;
leakage_power_unit : "1mW";
output_threshold_pct_fall : 60.000000;
output_threshold_pct_rise : 40.000000;
simulation : true;
slew_derate_from_library : 1.000000;
slew_lower_threshold_pct_fall : 20.000000;
slew_lower_threshold_pct_rise : 20.000000;
slew_upper_threshold_pct_fall : 80.000000;
slew_upper_threshold_pct_rise : 80.000000;
smr : "1.1";
switching_power_split_model : "false";
fast_mode : "false";
clk_width : "1e-5";
tran_timestep : "1e-9";
driver_model : "snps_predriver";
def_sim_opt : "POST=1 PROBE POST_VERSION=2001 INGOLD=2 NOMOD NOPAGE NUMDGT=10 \
 MEASDGT=10 LIMPTS=500000 ICSWEEP=0 AUTOSTOP ALTCC=1 RUNLVL=5 ACCURATE=1 \
 ABSVAR=1.000000e-03 RELVAR=1.000000e-02";
simulator : "HSPICE - Z-2007.03-SP1 64-BIT (May 25 2007)";
voltage_map("vdd",0.300000);
voltage_map("VSS",0.000000);
wire_load ("...") {
 ...
}
...
operating_conditions ("nom_0.30V_25C") {
       process : 1.000000;
       temperature : 25.000000;
       tree_type : "balanced_tree";
       voltage : 0.300000;
}
...
power_lut_template ("...") {
 ...
}
...
lu_table_template ("...") {
 ...
}
cell ("HS65_LS_IVX2") {
 area : 1.560000;
 drc_blockgroup : "stdcell";
 drc_blocktype : "primitive";
 drc_celltype : "combinational";
```

```
drc_ground_pin_drives : "gnd none";
drc_ground_pins : "gnd";
drc_maxparallel : "1.0000000";
drc_power_pin_drives : "vdd none";
drc_power_pins : "vdd";
drc_restriction : "none";
cell_leakage_power : 4.893559e-09;
driver_waveform_rise : "preDrv";
driver_waveform_fall : "preDrv";
leakage_power () {
  when : "A";
  value : "1.3523955e-09";
}
leakage_power () {
  when : "!A";
  value : "4.8935586e-09";
}
pg_pin (VSS) {
  voltage_name : "VSS";
  pg_type : "primary_ground";
}
pg_pin (vdd) {
  voltage_name : "vdd";
  pg_type : "primary_power";
}
pin (A) {
  direction : "input";
  drc_pindrive : "none";
  drc_pinfunction : "normal";
  drc_pinsigtype : "signal";
  input_signal_level : "vdd";
  fall_capacitance : 0.000698;
  capacitance : 0.000735;
  rise_capacitance : 0.000772;
  related_power_pin : "vdd";
  related_ground_pin : "VSS";
}
pin (Z) {
  direction : "output";
  drc_pindrive : "always";
  drc_pinfunction : "normal";
  drc_pinsigtype : "signal";
  function : "!A";
  output_signal_level : "vdd";
  related_power_pin : "vdd";
  related_ground_pin : "VSS";
  internal_power () {
```

```
 related_pin : "A";
 rise_power ("power_outputs_2") {
  index_1("14, 336, 672, 1316");
  index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
  values("8.9958224e-05,9.0848955e-05,8.4128682e-05,8.1043593e-05,7.7846468e-05",\
    "8.6611279e-05,8.5666748e-05,8.2510802e-05,7.8278533e-05,7.3416858e-05",\
    "9.1342833e-05,8.9576725e-05,8.7941140e-05,8.1391077e-05,7.3437488e-05",\
    "0.0001117,9.9934153e-05,9.5111262e-05,8.5488269e-05,7.7873520e-05");
 }
 fall_power ("power_outputs_2") {
  index_1("14, 336, 672, 1316");
  index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
  values("1.7566718e-06,2.0137259e-06,1.8391262e-06,1.2246051e-06,4.8597294e-10",\
    "-5.2181441e-07,-9.6498076e-07,-1.0469904e-06,-1.2507575e-06,-2.3489498e-06",\
    "3.5885141e-06,8.8390299e-07,-4.6569421e-07,-2.0259222e-06,-3.5742104e-06",\
    "1.6376896e-05,9.0964498e-06,4.4240158e-06,-2.2916374e-07,-4.1485237e-06");
 }
}
timing () {
  related_pin : "A";
  timing_type : "combinational";
  timing_sense : "negative_unate";
  cell_rise ("del_1_4_5") {
    index_1("14, 336, 672, 1316");
    index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
    values("41.7667394, 74.3408037, 121.0917588, 219.5416329, 402.1378004", \
      "286.3819191, 358.5690820, 418.0982103, 515.8856311, 697.9718705", \
      "463.3557182, 591.3699397, 697.7726912, 827.1576917, 1008.8973568", \
      "733.2425866, 944.9454410, 1130.4342706, 1360.8967038, 1605.5935248");
  }
  rise_transition ("del_1_4_5") {
    index_1("14, 336, 672, 1316");
    index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
    values("49.0504277, 116.9588728, 215.7385666, 421.9150753, 804.4387982", \
      "140.2664935, 168.5912565, 236.6559215, 425.5592501, 806.2555139", \
      "255.1329601, 297.1168840, 334.5039943, 470.3784384, 811.3843251", \
      "441.9297568, 527.2847261, 588.4608072, 658.4619427, 907.9815868");
  }
  cell_fall ("del_1_4_5") {
    index_1("14, 336, 672, 1316");
    index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
    values("23.1467805, 34.5959563, 51.3852356, 86.7941097, 151.0327081", \
      "190.3033109, 255.3869649, 307.6363271, 371.9690369, 443.4690482", \
      "281.0708679, 396.5011501, 491.9655225, 607.9822015, 731.8360531", \
      "395.5155421, 585.9676548, 753.1945698, 958.9723504, 1178.8190477");
  }
  fall_transition ("del_1_4_5") {
```

```
        index_1("14, 336, 672, 1316");
        index_2("0.0014, 0.0044, 0.0088, 0.018, 0.035");
        values("17.3663537, 41.3436787, 76.4175354, 149.1202681, 283.5765827", \
          "126.2228295, 147.2918001, 163.0914852, 194.2180035, 297.1398771", \
          "226.6253176, 267.6360111, 297.1233357, 329.9759896, 381.9205006", \
          "391.0089390, 469.4636289, 527.4270052, 583.5007642, 643.7893489");
      }
    }
  }
}
```

B.4 Compilation de la bibliothèque

Finalement, pour être utilisable avec les outils de synthèse notamment, la bibliothèque au format .lib doit être compilée au format `.db`. Cela est fait à l'aide de Library Compiler de Synopsys. La commande est [5] :

```
> lc_shell -xg_mode -f compile_cmd
```

où le fichier `compile_cmd` contient :

```
> read_lib AESlib_0_3V_rechar.lib
> write_lib AESlib_0_3V_rechar
```

5. L'option `-xg_mode` n'est normalement pas indispensable mais semble être obligatoire sur les stations de travail en DICE.

Annexe C

Publication

Cédric Hocquet, David Bol, Dina Kamel, Jean-Didier Legat, "Assessment of 65nm sub-threshold logic for smart RFID applications", *8ème Journées d'études Faible Tension Faible Consommation 2009*, Neuchâtel, juin 2009.