Intégration et paradigme de calcul d'opérateur à base de mémoire non volatile à l'intérieur de l'architecture d'un processeur

Alban Nicolas

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de l'École Centrale de Lyon

École Doctorale n°160
Électronique Électrotechnique et Automatique (EEA)
Spécialité de doctorat :
Discipline : Électronique

Soutenue publiquement le 28/11/2025, par :
NICOLAS Alban

Intégration et paradigme de calcul d'opérateur à base de mémoire non-volatile à l'intérieur de l'architecture d'un processeur

---

Devant le jury composé de :

Pascal BENOIT	**Président du Jury**
Professeur des universités, LIRMM, Université de Montpellier	
Olivier SENTIEYS	**Examinateur**
Professeur des universités, Inria/Irisa, Université de Rennes	
Lorena ANGHEL	**Rapporteuse**
Professeur des universités, Spintec, Grenoble INP	
Florent BRUGUIER	**Rapporteur**
Maître de conférences, LIRMM, Université de Montpellier	
David NAVARRO	**Directeur de thèse**
Maître de conférences, INL, École Centrale de Lyon	
Cédric MARCHAND	**Co-encadrant de thèse**
Maître de conférences, INL, École Centrale de Lyon	

Remerciements

Ce manuscrit est issu d'une (longue) collaboration de presque cinq ans et d'interactions avec un grand nombre de personnes. Leur aide directe ou indirecte a été précieuse à la rédaction de ce document. Dans un premier temps, je tiens à remercier Cédric Marchand pour l'opportunité qu'il m'a donnée en me proposant de faire cette thèse, ainsi que David Navarro, mon directeur de thèse. Merci à vous deux pour vos directives, conseils, patience et gentillesse qui m'ont guidé tout au long de ce travail.

Ensuite, je remercie les membres de l'équipe Électronique avec qui j'ai pu avoir des discussions enrichissantes sur le monde de la recherche, et tout simplement de bons moments. Merci pour ces moments partagés, ces instants d'échanges sur les méthodes scientifiques et les moments de détente et de rire. Merci à vous Ian O'Connor, Alberto Bosio, Bastien Deveautour, Bertrand Vilquin et Damien Deleruyelle.

Merci ensuite au personnel du laboratoire. Merci à Laurent Carrel et Raphaël Lopez du service technique, qui m'ont aidé à de nombreuses occasions concernant le matériel et les serveurs. Et un immense merci à Patricia Dufaut et Sylvie Goncalves pour leur immense soutien lors des déplacements, des événements, pour l'administratif, leur bonne humeur et leur humour. Sans vous, je n'aurais même pas pu commencer ma thèse vu les aventures administratives nécessaires à son commencement !

Merci aux Enseignants-Chercheurs de l'École Centrale de Lyon pour m'avoir fait confiance avec l'encadrement des étudiants de première année en tant qu'ATER malgré la thèse non terminée. Merci à l'équipe Électronique encore une fois, mais aussi merci à l'équipe du cours de traitement du signal et essentiellement à Gérard Scorletti, Julien Huillery et Arthur Perodou avec qui j'ai pu voir comment s'organisaient vraiment des groupes de cours pour des futurs ingénieurs ! Cette partie de cette aventure est une de mes préférées, échanger avec les élèves et les enseignants a été un vrai plaisir du début à la fin.

Enfin, merci ensuite aux doctorants de l'équipe Électronique pour ces moments de travail sérieux, de conseils, de soutiens et leur enthousiasme lorsque j'en ai eu besoin. Merci aussi pour ces nombreuses pauses café, réflexions (plus ou moins) philosophiques, les afterworks, repas organisés et plein d'autres événements ! Toutes ces choses, ajoutées à vos conseils, ont fait que ces années au sein du laboratoire ont été de bonnes années. Vous êtes vraiment nombreux, je ne veux oublier personne, donc merci à vous tous !

Pour ce qui est des remerciements plus personnels, à Lyon tout d'abord, je souhaite remercier mes nombreux colocataires et amis qui se sont succédé à la Charbonnerie : Solveig, Paul-Antoine, Sophita, Raphaël, Swayam, Martina et Tam. La cohabitation a parfois été mouvementée, entre les soucis liés à la maison, les petites disputes et divers imprévus, mais elle restera avant tout un moment exceptionnel, riche en soirées jeux, soirées films, repas partagés, et longues discussions sur la thèse, la vie et tout ce que l'on peut imaginer. Je remercie ensuite mes amis lyonnais, souvent eux aussi doctorants : Étienne, Mayeul, Iuliia, Mohab, Ali et Ali, pour leur soutien et leur présence. Enfin, je tiens à remercier ma psychologue à Lyon, Mathilde Chanal, pour son suivi exceptionnel, ses conseils et son

écoute. J'ai eu beaucoup de chance de rencontrer une professionnelle aussi compétente.

Merci à tous mes amis, en France, en Bretagne et ailleurs dans le monde, pour leur soutien durant ces années parfois difficiles moralement. Je suis profondément reconnaissant de vous connaître et pour tous les moments partagés ensemble, passés comme présents. Les mots me manquent [1] pour exprimer à quel point ces liens et le temps passé à vos côtés comptent pour moi.

Finalement, je remercie ma famille, et tout particulièrement ma mère, qui a toujours été là pour m'écouter raconter ma vie, mes galères et parler de ma thèse, même si tout n'était pas toujours clair pour elle. Merci pour son soutien constant, tout au long de ces années d'études et au-delà.

Résumé

L'architecture de Von Neumann est actuellement la plus répandue dans la conception des systèmes électroniques. Son organisation repose sur une distinction nette entre les unités de calcul et les zones de mémoire. Cette séparation, combinée à l'évolution beaucoup plus lente des technologies mémoire par rapport à celle des processeurs, entraîne un ralentissement significatif de l'échange des données [2]. Ce phénomène, connu sous le nom de « goulot d'étranglement de Von Neumann » ou « Memory Wall », limite fortement les performances globales des systèmes et est responsable de plus de la moitié de leur consommation énergétique [3].

Cette contrainte technologique devient d'autant plus critique dans le contexte actuel marqué par l'essor de l'Internet des Objets (IoT, Internet of Things). D'ici à 2030, on estime à plus de 32 milliards le nombre de dispositifs connectés [4], dont une majorité reposera sur des capteurs disposant d'un faible apport en énergie. Pour ces systèmes, l'optimisation de la consommation est un enjeu central, rendant indispensable le développement de solutions matérielles capables de réduire les transferts énergivores entre calcul et mémoire.

Dans cette perspective, les mémoires non-volatiles émergentes, et en particulier les transistors ferroélectriques à effet de champ (FeFET) [5], suscitent un intérêt croissant. Compatibles avec les procédés CMOS, ces dispositifs offrent la possibilité d'intégrer directement des capacités de stockage au plus près des unités de calcul. Grâce à leur non-volatilité, ils réduisent les transferts de données nécessaires et apparaissent ainsi comme une voie prometteuse pour atténuer le « goulot d'étranglement de Von Neumann ».

L'intégration de telles technologies se heurte cependant à des difficultés : la conception d'opérateurs exploitant ces technologies encore en cours de développement demeure un processus long et coûteux. Afin de disposer de résultats intermédiaires sur leur pertinence, il est nécessaire de s'appuyer sur des outils d'évaluation adaptés. Ces derniers permettent d'anticiper l'impact de ces nouvelles technologies sur les architectures existantes, tout en guidant les efforts de conception matérielle.

Dans ce contexte, cette thèse propose une plateforme d'émulation fondée sur l'architecture RISC-V. Celle-ci permet de modéliser et d'évaluer le comportement de nouvelles instructions intégrant des composants non-volatiles. Un exemple d'utilisation est présenté autour de l'opération de déchiffrement AES, démontrant comment l'émulation fournit des indications précieuses sur l'impact de l'intégration de FeFET dans le flot d'exécution d'un processeur.

Abstract

The Von Neumann architecture is currently the most widely used in the design of electronic systems and processors. Its organisation is based on a clear distinction between computing units and memory areas. This separation, combined with the much slower evolution of memory technologies compared to processors, leads to a significant slowdown in data exchange [2]. This phenomenon, known as the "Von Neumann bottleneck" or "Memory Wall", severely limits the overall performance of systems and is responsible for more than half of their energy consumption [3].

This technological constraint is becoming all the more critical in the current context marked by the rise of the Internet of Things (IoT). By 2030, it is estimated that there will be more than 32 billion connected devices [4], the majority of which will be low-power sensors. For these systems, optimising energy consumption is a key challenge, making it essential to develop hardware solutions capable of reducing energy-intensive transfers between computing and memory.

In this context, emerging non-volatile memories, and in particular ferroelectric field-effect transistors (FeFET) [5], are attracting growing interest. Compatible with CMOS processes, these devices offer the possibility of directly integrating storage capacities as close as possible to the computing units. Thanks to their non-volatility, they reduce the amount of data transfer required and thus appear to be a promising way of alleviating the "Von Neumann bottleneck".

However, integrating such technologies presents challenges: designing operators that exploit these technologies, which are still under development, remains a lengthy and costly process. In order to obtain interim results on their relevance, it is necessary to rely on appropriate evaluation tools. These tools make it possible to anticipate the impact of these new technologies on existing architectures, while guiding hardware design efforts.

In this context, this thesis proposes an emulation platform based on the RISC-V architecture. This platform makes it possible to model and evaluate the behaviour of new instructions incorporating non-volatile components. An example of its use is presented around the AES decryption operation, demonstrating how emulation provides valuable insights into the impact of integrating FeFET into a processor's execution flow.

Table des matières

1 Introduction **1**
1.1 Contexte de la thèse . . . 3
1.1.1 L'explosion de l'IoT . . . 3
1.1.2 Le problème du ≪ Memory Wall ≫ . . . 6
1.2 L'émergence d'un nouveau paradigme : la Logique en Mémoire (LiM) . . . 8
1.3 Le défi de créer de nouveaux opérateurs dédiés à ces nouvelles technologies 9
1.4 Les objectifs de la Thèse . . . 9
1.5 Plan du manuscrit de Thèse . . . 10

2 État de l'art **11**
2.1 Technologie de mémoires non-volatiles émergentes et intégration dans le Logic-In-Memory . . . 13
2.1.1 Technologies de mémoires non-volatiles émergentes . . . 13
2.1.2 In memory Computing :Logic-in-Memory à l'aide des NVM . . . 24
2.2 Processeur : RISC-V . . . 30
2.2.1 Bref historique des architectures des processeurs . . . 30
2.2.2 Présentation de RISC-V . . . 35
2.3 Méthodes d'évaluation des technologies de mémoires non-volatiles. . . . 37
2.3.1 Approches de tests et d'évaluations . . . 37
2.3.2 Comparaison des méthodes et explication du choix pour la thèse . . 39
2.4 Conclusions . . . 40

3 L'émulation d'opérateurs non-volatiles **41**
3.1 L'élaboration du modèle pour les opérateurs NVM . . . 42
3.1.1 Le modèle réaliste . . . 42
3.1.2 Le modèle comportemental . . . 45
3.2 Le choix du coeur de la plateforme . . . 46
3.2.1 Quel type d'unité de calcul pour la plateforme ? . . . 46
3.2.2 Le choix du processeur . . . 48
3.2.3 COMET RISC-V . . . 51

3.3 La plateforme d'émulation complète 53
3.4 Conclusions 55

4 Tests et Expériences **57**
4.1 Présentation du contexte de l'expérience : l'AES et les Corps de Galois . . . 59
4.1.1 L'Advanced Encryption Standard : Un algorithme de cryptographie impactant 59
4.1.2 Les Corps de Galois : la part mathématique de l'AES 63
4.2 Mise en place des opérateurs non-volatiles à base de FeFETs 67
4.2.1 Création de la carte modèle d'un FeFET 67
4.2.2 Création de la carte modèle d'un FeMFET 69
4.3 Expériences et Résultats 73
4.3.1 La mise en place de l'opération de MixColumn 74
4.3.2 Fonctionnement du simulateur 78
4.3.3 Présentation des résultats 79
4.4 Conclusion 86

5 Conclusions et Perspectives **87**
5.1 Résumé de la contribution Technique 88
5.2 Perspectives 89
5.3 La dissémination du travail 90

Table des figures

1.1 Nombre d'objets connectés entre 2019 et 2030, par secteur [11] 4
1.2 Two numerical solutions 5
1.3 Évolution des mémoires par rapport aux processeurs (d'après [2]) 7
1.4 Architecture de Von Neumann 7

2.1 Schéma explicatif du plan de l'état de l'art 12
2.2 Les différentes catégories de NVM 14
2.3 Différence de progression entre un processeur volatile et un processeur Non-volatile dans le contexte d'un environnement à faible apport d'énergie . . . 15
2.4 (a) : Schéma représentatif du canal formé à l'intérieur de la couche isolante dans une cellule RRAM et (b) : Schéma du même canal dans l'état reset . . 16
2.5 Illustration du changement de phase d'une Cellule PCM 18
2.6 Schéma illustrant une cellule de STT-MRAM 19
2.7 Illustration de l'hystérésis du matériau ferroélectrique 21
2.8 Schéma d'une FeRAM composée d'un transistor et un condensateur (1T-1C) issue de [56] 22
2.9 Schéma classique d'une cellule FeFET 22
2.10 Présentation des différents types de calcul en mémoire avec deux principaux paradigmes qui sont le IMC et le NMC. Sur la droite de la figure, on voit que l'IMC est lui-même divisé en trois avec le CiM, le CwM et la LiM . . . 26
2.11 Schématique d'une cellule TC-MEM pour un bit 29
2.12 Schéma illustrant les deux architectures les plus couramment utilisées dans les processeurs 31
2.13 Frise chronologique de l'évolution des architectures de processeur 33

3.1 Comparaison des hystérésis entre modèle approché et modèle réel 43
3.2 Utilisation des ressources sur le FPGA (DSP ou LUT) en fonction de la méthode choisie 44
3.3 Pipeline du processeur Comet 52
3.4 Schéma complet de la plateforme d'émulation des opérateurs non-volatiles . 53

4.1 Schéma des tours de l'algorithme AES pour le chiffrement (à gauche) et le déchiffrement (à droite) 62
4.2 Schéma des différentes portes FeFET avec (a) un NAND, (b) un XOR, et (c) un AND issu de [5] 68
4.3 Les deux types de FeFET utilisés avec (a) le transistor MFIS et (b) le transistor MFMIS 70
4.4 Schéma des différentes portes FeMFET avec (a) un NAND, (b) un AND, et (c) un XNOR 71
4.5 Schéma du multiplicateur de Galois pour 4 bits sans NVM 74
4.6 Schéma de la cellule de multiplication dans un corps de Galois pour un bit, avec (a) le schéma issu de l'article et (b) la version adaptée aux technologies non-volatiles. 75
4.7 Schéma du multiplicateur de Galois pour 4 bits à base de NVM 76
4.8 Schéma de la cellule pour la multiplication de Galois pour un bit à base de NVM spécifique pour l'AES 77
4.9 Pipeline du processeur Comet, avec les étages modifiés en bleu 79
4.10 Graphique du nombre de cycles du cœur nécessaires au déchiffrement, pour différents types de paramètres utilisés dans l'opérateur non-volatile, où W correspond aux valeurs d'écriture et R aux valeurs de lecture 80
4.11 Graphique représentant le nombre de cycles du cœur du processeur nécessaires pour l'opération de déchiffrement en fonction du nombre de déchiffrements. (a) présente les résultats sur une échelle logarithmique et (b) présente les résultats en valeurs absolues. 83

Liste des tableaux

2.1 Tableau comparatif des différentes technologies de mémoire avec les technologies disponibles indiquées en vert, celles en conception en orange et celles à l'étude en rouge 23
2.2 Comparaison des architectures ARM et x86 32
2.3 Table de comparaison des points avantages et inconvénients de chaque méthode 39

3.1 Éléments présents dans le coeur FPGA Virtex7 XC7VX690T 44

4.1 Informations sur les paramètres utilisés pour les premiers tests 69
4.2 Caractéristiques de consommation énergétique des portes FeMFET 71
4.3 Temps de transition et de propagation des portes logiques FEMFET 72
4.4 Carte Modèle pour les FeMFETs 73
4.5 Paramètres finaux pour l'opération de Mix Column 78
4.6 Différence en termes de cycles consommés par l'instruction simple et les différentes versions de l'opérateur non-volatile 82
4.7 Excédent, exprimé en pourcentage, pour chaque version de l'instruction basée sur la NVM par rapport à la version de base. 82
4.8 Nombre de cycles par opération de déchiffrement pour chaque configuration des paramètres testés 84
4.9 Nombre de cycles par déchiffrement pour chaque configuration des paramètres 84
4.10 Nombre de lecture des portes NVM AND lors des expériences 85
4.11 Valeurs estimées en picojoules pour les différents nombres de déchiffrements effectués 86

Acronymes

AES Norme de Chiffrement Avancée (Advanced Encryption Standard). , 28, 56–65, 67, 74, 76, 77, 81, 85, 89, 90

ARM Advanced RISC machine. , 30, 32–35

ASIC Circuit intégré propre à une application (Application-specific integrated circuit). 37

CBRAM Mémoires à accès aléatoire à pont conducteur (Conductive Bridge RAM). 16

CiM Calcul en mémoire (Computing In Memory). , 16, 25–28, 34, 40

CISC Complex Instruction Set Computer. 32, 33

CMOS Semi-Conducteur à Oxyde Métallique Complémentaire (Complementary Metal-Oxide Semiconductor). , 8, 14, 16, 22, 24, 28, 29, 36, 70, 81, 85, 86, 89

CNFET Transistor à effet de champ à nanotube Calcul (Carbon Nanotube Field Effect Transistor). 16

CnM Calcul proche mémoire (Computing near Memory). 25

CoM Calcul hors mémoire (Computation Outside Memory). 25

CPU Unité centrale de calcul (Central Processing Unit). 6, 8, 54, 55

CwM Calcul avec la mémoire (Computing With Memory). , 25, 26

DPRAM Mémoire à double port (Dual Port RAM. 54

DRAM Mémoire vive dynamique(Dynamic RAM). 17, 20–23, 27

DSP Traitement du signal numérique (Digital Signal Processing). , 44, 45, 88

FeFET Transistor ferroélectrique à effet de champ (Ferroelectric Field Effect Transistor). , 2, 8, 10, 12, 20–24, 28, 40, 42, 45, 57, 58, 67–70, 75, 76, 78, 79, 81, 83–85, 88–90

FeMFET FeFET particulier décrit en sous-section 4.2.2(Ferroelectric Metal Field Effect Transistor). , 57, 69–73, 79, 86

FeRAM Mémoire vive ferroélectrique (Ferroelectric RAM). , 20–24

FM Ferromagnétique. 19

FPGA Réseau de portes programmables sur puce (Field-Programmable Gate Array). , 9, 17, 38, 39, 44, 47, 48, 55, 56, 88, 90

GPU Processeur graphique (Graphics Processing Unit). 17, 34, 36

HLS Synthèse Haut Niveau (High Level Synthesis). 50–52

IA Intelligence artificielle. 5, 27, 34

IMC Calcul en mémoire (In-Memory Computing). , 6, 25, 26, 55

IoT Internet des objets (Internet of Things). , 1–5, 8, 14, 17, 33, 36, 38, 46–49, 51, 55, 58, 73

ISA Architecture du jeu d'instruction (Instruction Set Architecture). 12, 34, 35, 51, 77

LiM Logique en mémoire (Logic in Memory). , 1, 6, 8, 9, 12, 13, 24–30, 36

LUT Table de correspondance (Look-Up Table). , 23, 44

MRAM Mémoire vive magnétique (Magnetoresistive RAM). , 13, 19

MTJ Tunnel de jonction magnétique (Magnétique Tunnel Junction. 19, 28

NMC Calcul proche mémoire (Near Memory Computing). , 25, 26

NVM Mémoire non-volatile (Non Volatile memory). , 2, 11–14, 22–24, 27, 29, 38, 41, 42, 55, 67, 68, 74, 76, 77, 81, 82, 84–86, 88

OxRAM Mémoires à accès aléatoire à oxyde métallique (Metal Oxyde Random Access Memory). 16, 23, 24, 27

PCM Mémoire à changement de phase (Phase Change Memory). , 13, 17, 18, 23, 24

RAM Mémoire vive (Random Access Memory). , 6, 13, 16, 17, 19–22, 44, 54

RISC Reduced Instruction Set Computer. , 32, 34, 35

RISC-V Jeu d'instruction ouvert pour processeur. , 9, 11, 12, 30, 33–36, 40, 42, 48–51, 54, 55, 58, 67, 73, 77, 78, 86, 88

RRAM Mémoire vive résistive (Resistive RAM). , 13, 16, 17, 20, 23, 27, 28

RTL Transfert de registres (Register Transfer Level). 49–52

SECRET SECure and Reconfigurable processor using Emerging Technology, un projet ANR JcJC. 5, 6, 8, 9, 12, 24, 40, 42, 43, 46–49, 58

SRAM Mémoire vive statique (Static RAM). 17, 20, 23, 24, 27

STT-MRAM RAM à couple de transfert de spin magnéticorésistif (Spin-Transfer Torque MRAM). , 19, 20, 23, 24, 28

TCAM Mémoire Adressable par le Contenu Ternaire (Ternary Content-Addressable Memory). 28

VHDL Langage de Description Matérielle pour VHSIC (VHSIC Hardware Description Language. 49–52

VHSIC Circuit intégrés à très grande vitesse (Very High Speed Integrated Circuit).

Chapitre 1

Introduction

1.1 Contexte de la thèse 3
1.1.1 L'explosion de l'IoT 3
1.1.2 Le problème du « Memory Wall » 6
1.2 L'émergence d'un nouveau paradigme : la Logique en Mémoire (LiM) 8
1.3 Le défi de créer de nouveaux opérateurs dédiés à ces nouvelles technologies 9
1.4 Les objectifs de la Thèse 9
1.5 Plan du manuscrit de Thèse 10

Depuis l'explosion de l'Internet des Objets (IoT), qui est un terme pour désigner les objets connectés, en 2024, 18 milliards de systèmes électroniques sont recensés. Ce nombre pourrait atteindre 32 milliards de systèmes électroniques, communicants en permanence entre eux, d'ici 2030 [4]. Cette multitude d'objets pose un certain nombre de problèmes. Parmi ceux-ci, on observe notamment une croissance du nombre de communications et de données transmises. Ces données peuvent être sensibles et ont besoin de protection pour pouvoir être envoyées en toute sécurité, ce qui ajoute un surcoût dans le développement de systèmes électroniques [6].

Le nombre croissant d'objets connectés accentue un problème majeur en électronique : les limites de l'architecture dominante utilisée par les processeurs pour traiter les données. L'architecture de Von Neumann, la plus couramment utilisée, souffre de la séparation entre les unités de calcul et les zones mémoire. Cette dissociation, aggravée par l'évolution des technologies de mémoire, plus lente que celle des processeurs [2], ralentit l'échange des données entre ces parties et engendre le phénomène connu sous le nom de « goulot d'étranglement de Von Neumann » ou « Memory Wall ». Ce problème fait que transférer les données entre ces deux parties représente plus de la moitié de la consommation dans les systèmes électroniques [3].

Le nombre croissant d'objets connectés augmente davantage le surcoût global en énergie dû au « Memory Wall », ce qui oblige le monde informatique à repenser les systèmes, voire à repenser la façon de calculer. Actuellement, de nombreuses technologies sont évaluées pour essayer de venir à bout de ces défis. Les technologies intégrant de la mémoire non-volatile (NVM) au sein des unités de calcul figurent parmi les domaines de recherche les plus explorés [7]. Ces technologies viennent directement fournir une réponse aux défis présentés du fait qu'elles permettent de réduire le nombre de transferts de la mémoire vers les unités de calcul grâce à cette non-volatilité. Cependant, ces technologies doivent être testées afin de savoir si leur fonctionnement peut être efficace dans le système complet. Parmi elles se trouve la technologie des transistors ferroélectriques (FeFET) qui permet d'obtenir une porte logique non-volatile à l'aide d'un simple transistor [5]. Si cette technologie semble prometteuse d'un point de vue théorique, il est nécessaire de l'évaluer dans un système complet. L'un des objectifs de la thèse est de répondre à ce besoin d'évaluation. Dans ce manuscrit sera présentée une plate-forme d'émulation de ces technologies mise au point au cours de la thèse afin d'avoir une première vision de l'impact de ces nouveaux opérateurs non-volatiles au sein d'une architecture de calcul complète.

Dans ce chapitre, seront tout d'abord présentées en Section 1.1 les raisons de l'existence de la thèse en détaillant le contexte dans lequel le travail se situe. La Section 1.2 présente l'une des solutions possibles à la résolution des problèmes posés. La Section 1.3 présente les défis liés au développement de ces nouveaux outils. Enfin, les Sections 1.4 et 1.5 présentent les objectifs de la thèse et détaillent le plan de ce manuscrit.

1.1 Contexte de la thèse

1.1.1 L'explosion de l'IoT

1.1.1.1 L'origine et les chiffres autour de l'IoT

Ce qui est communément appelé « Internet of Things » (IoT) en anglais, ou l'Internet des Objets en français, est un concept inventé en 1999 par Kevin Ashton, un scientifique anglais travaillant au MIT, qui a notamment contribué à la standardisation du protocole pour la radio-identification (Radio-Frequency Identification, RFID en anglais). Une définition possible pour ce terme est la suivante : « Un réseau global et ouvert d'objets intelligents qui ont la capacité de s'auto-organiser, de partager des informations ou des ressources, d'agir et de réagir par rapport à des situations et des changements dans leur environnement de fonctionnement » [8]. De nombreuses définitions sont possibles, mais toutes s'accordent à dire que l'IoT est une deuxième version d'Internet où les données sont créées par des « objets », là où la première version d'Internet découlait des données créées par les utilisateurs humains.

L'appellation a cependant rapidement pris de l'importance au niveau du grand public, du fait de l'immense nombre d'objets du quotidien qui possèdent maintenant des variantes connectées. Les applications de l'IoT sont nombreuses, allant d'un réseau de capteurs permettant de prévoir des séismes [9] à des outils permettant aux personnes âgées de vivre plus confortablement [10]. Il est désormais courant de trouver dans les maisons un certain nombre de ces objets, que ce soit les télévisions, les aspirateurs robots contrôlés via smartphone, ou bien encore les compagnons dotés d'intelligence artificielle, comme « Alexa » proposé par la marque Amazon. Cette utilisation grand public fait que, depuis quelques années déjà, le nombre d'objets connectés a explosé. En effet, on peut voir sur la Figure 1.1 que le nombre d'objets attendu pour 2030 est estimé à environ 30 milliards d'objets connectés simultanément [4], avec parmi ces derniers une bonne moitié d'objets pour les consommateurs, bien devant le secteur tertiaire.

Ce nombre croissant de systèmes connectés représente une opportunité commerciale avec des revenus annuels qui sont actuellement de 293 milliards de dollars en 2023 et qui sont supposés atteindre 621 milliards de dollars d'ici 2030 [12]. Cependant, cette génération de revenus ne se fera qu'en augmentant drastiquement le nombre d'objets disponibles, ce qui induira une croissance du nombre de communications entre les systèmes connectés et donc la quantité d'informations transmises. Il est estimé que le volume de données transmises par l'IoT pourrait atteindre 79,4 zettaoctets d'ici à 2025 [13].

Cette quantité immense de données doit être traitée toujours plus rapidement et efficacement. De plus, parmi ces données se trouvent par exemple des données médicales, hautement sensibles, qui ont besoin d'être sécurisées afin de rester privées. Cette protection est partiellement mise en place au niveau de la communication vers le cloud, car les protocoles de communication demandent une sécurisation de plus en plus importante des

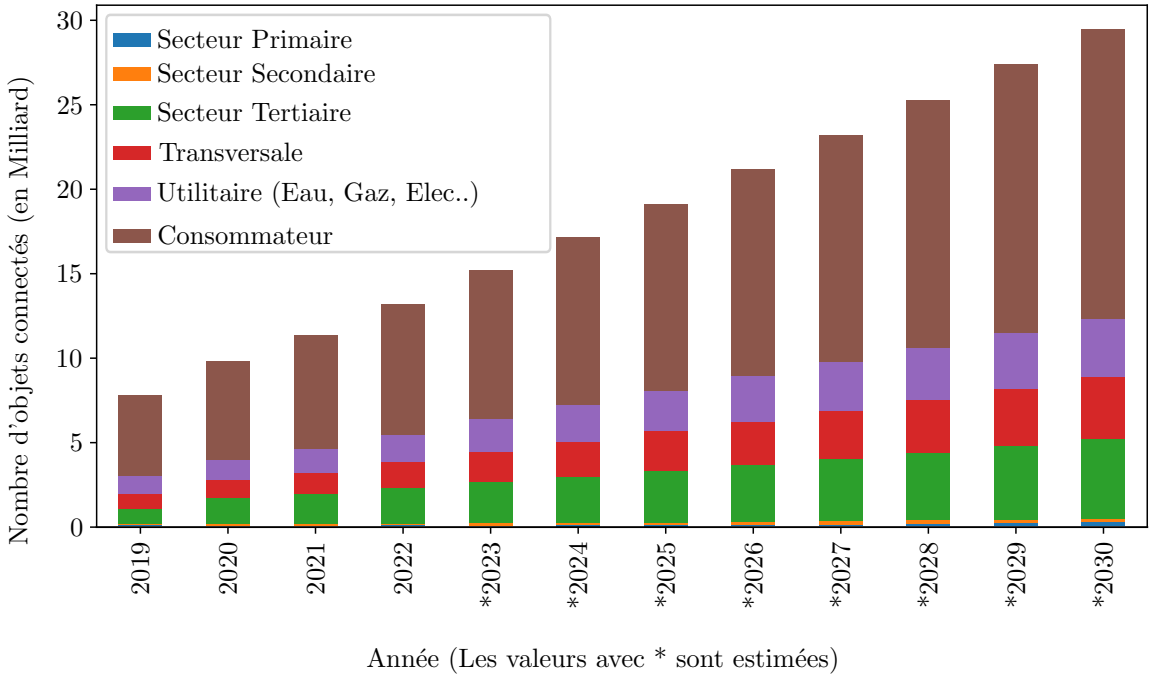
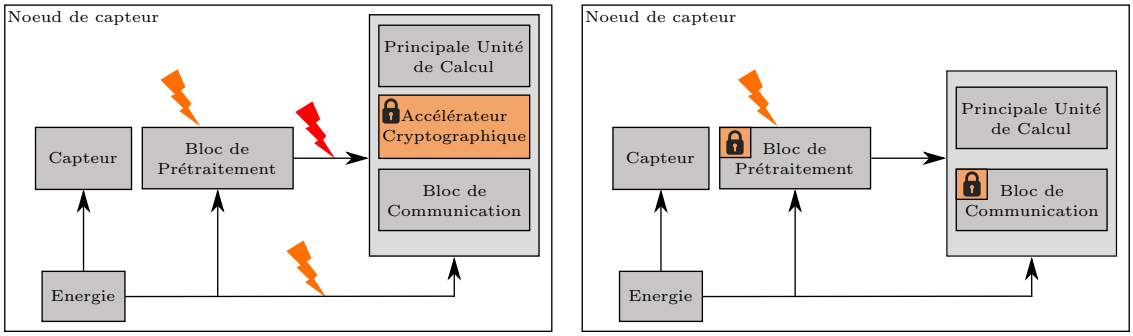


FIGURE 1.1 : Nombre d'objets connectés entre 2019 et 2030, par secteur [11]

données échangées. Cela ne reste pas suffisant et il reste un large espace où des données non protégées sont manipulées.

Parmi ces espaces se situent notamment l'intérieur des systèmes eux-mêmes où les données sont bien souvent brutes (non cryptées ou transformées) et donc vulnérables à de nombreuses attaques existantes. Une façon courante de compromettre les données du nœud IoT est ce qu'on appelle le probing (sondage en français). Cette technique consiste à directement venir observer ce qui se passe sur les éléments du circuit à l'aide de techniques comme celle de la sonde ionique focalisée qui consiste à injecter un matériel permettant d'envoyer des informations, ou de re-router des chemins de données [14]. Un autre type d'attaque est par exemple la « Screaming Channel » [15] qui se base sur une caractéristique spécifique des puces intégrées aux systèmes IoT : la coexistence des transmetteurs radio et des circuits de traitement des données. Cette coexistence entraîne une interaction entre les rayonnements électromagnétiques générés lors des calculs et la fréquence porteuse de l'onde radio. Par conséquent, des informations sensibles, telles que des détails sur le chiffrement des données, peuvent être involontairement transmises. Cette fuite de rayonnement électromagnétique est à l'origine de l'appellation « Screaming Channel », ou « Canaux hurlants » en français. Pour diminuer le risque de fuites de données, des projets informatiques sont nés, dont le projet SECRET.



(a) Schéma classique d'un capteur IoT avec les zones sensibles représentées par les éclairs

(b) Schéma d'un capteur IoT avec la sécurité est déplacée directement dans les zones sensibles

FIGURE 1.2 : Idée de la solution proposée par le projet SECRET avec (a) le capteur IoT avant la sécurisation et (b) le même capteur après sécurisation

1.1.1.2 Le projet SECRET

Le projet SECRET, pour « SECure and Reconfigurable processor using Emerging Technology » (ou Processeur sécurisé et reconfigurable à l'aide de technologies émergentes en français), se place dans le contexte de l'IoT présenté précédemment. L'objectif du projet est de renforcer la sécurité des données collectées par les nœuds des dispositifs IoT. Cette partie figure parmi les plus vulnérables en termes de types d'attaques possibles, du fait de la possibilité d'interagir physiquement avec les nœuds du réseau de capteurs, contrairement au reste du réseau, qui se trouve généralement dans des lieux éloignés ou sur le Cloud.

À l'heure actuelle, les nœuds des réseaux de capteurs ont un déficit de protection et peuvent se faire attaquer en de nombreux endroits, comme le montre la Figure 1.2a.

La zone la plus dangereuse se situe entre le bloc de prétraitement et le reste de la zone de calcul du système. En effet, cette dernière contient les données utiles sensibles non protégées, tout juste mises en forme, qui sont en train d'être transférées vers le reste de la plateforme pour leur traitement.

Ainsi, le but du projet est d'ajouter une couche de sécurité au plus près de la partie captant les données transmises pour réduire fortement toutes ces zones de vulnérabilités. Le but final est d'atteindre une structure plus sécurisée, comme représentée à la Figure 1.2b.

Cependant, l'ajout d'une couche de sécurité supplémentaire possède de nombreux inconvénients. Les solutions de sécurité matérielle et logicielle pour l'IoT sont peu attrayantes [6] car elles entraînent un surcoût en termes de surface du circuit, du fait de l'ajout du circuit correspondant au bloc de sécurisation. De plus, elles augmentent également la quantité d'énergie consommée dans un domaine déjà limité par la taille de ses systèmes. Il est donc primordial d'améliorer l'efficacité énergétique liée à la mise en œuvre d'une solution de sécurité dans les dispositifs concernés. Il est également nécessaire de prendre en compte la croissance rapide de l'IoT et de l'Intelligence Artificielle (IA), qui fait également

exploser la consommation énergétique [16].

Dans cette optique, le projet SECRET souhaite modifier l'architecture de l'unité de calcul située entre les capteurs et l'unité permettant l'envoi des données collectées. Cette modification vise l'intégration de nouveaux opérateurs de calcul dédiés à la sécurité, conçus à partir de technologies émergentes et se servant de concepts connus sous le nom d'« In-memory Computing » (IMC calcul en mémoire) et plus précisément d'une de ces sous-parties, la « Logic in Memory » (LiM, logique en mémoire en français).

L'ajout de mémoire capable d'effectuer des calculs logiques permet également de contrer un problème majeur actuel, connu sous le nom de « Memory Wall » (mur de la mémoire en français).

1.1.2 Le problème du « Memory Wall »

Le concept de « Memory Wall » en électronique a évolué en même temps que les systèmes électroniques en général [17]. Le concept a été popularisé dans les années 90 [18] et désigne la situation où les processeurs et autres unités de calcul se développent plus rapidement, du fait des évolutions de la technologie des semi-conducteurs et de l'amélioration des architectures des processeurs, que les systèmes mémoire comme la RAM (Random Access Memory) et autres cellules mémoires. De nos jours, les processeurs peuvent fonctionner à de grandes vitesses et peuvent même effectuer des tâches en parallèle. Les cellules mémoires ont elles aussi progressé, mais pas à la même vitesse que les processeurs, comme le montre la Figure 1.3. Les deux technologies progressent à partir d'une base d'amélioration commune. Cependant, l'évolution des processeurs est beaucoup plus rapide, atteignant un facteur supérieur à 10 000 sur une période de 30 ans, tandis que la progression des cellules mémoires sur la même période est d'environ un facteur 10 [2].

Cette disparité dans le développement des deux technologies fait que le temps nécessaire au système pour accéder à une donnée dans la mémoire peut devenir significatif par rapport à la vitesse de calcul du processeur. Cela engendre un goulot d'étranglement pour le CPU, et donc une réduction de la vitesse globale du système.

Un autre nom pour ce phénomène est « Goulot d'étranglement de Von Neumann ». Cependant, ce terme est plutôt mal choisi car il fait référence à l'architecture de Von Neumann. Cette dernière est utilisée par la majeure partie des unités de calcul et se caractérise par la séparation entre le processeur et la mémoire qui sont reliés par un seul bus de données, comme le montre la Figure 1.4.

Ce simple bus est donc responsable d'un éventuel goulot d'étranglement. Dans un cas où la mémoire serait aussi rapide que le processeur, ce bus viendrait limiter la vitesse du système complet du fait du manque de bande passante due à la lenteur du bus de données possiblement saturé d'informations (entre les requêtes de données du processeur et les informations à transmettre provenant de la mémoire). Ainsi, l'appellation goulot d'étranglement de Von Neumann est abusive car le problème ne vient pas seulement de l'architecture, bien que ses caractéristiques y contribuent, mais du fait que la vitesse de

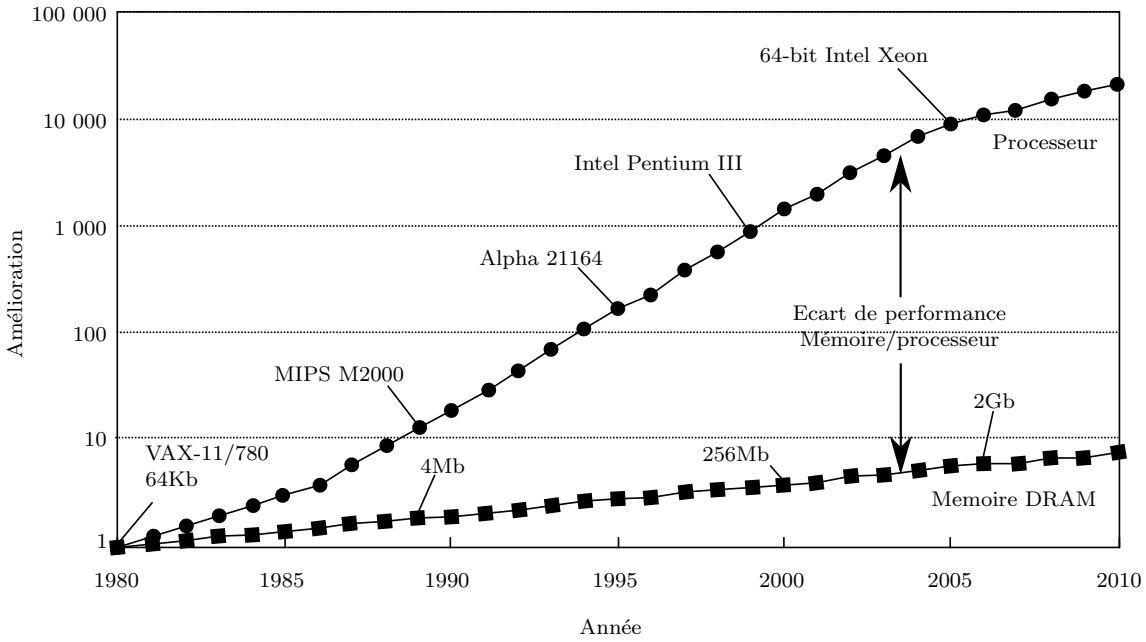


FIGURE 1.3 : Évolution des mémoires par rapport aux processeurs (d'après [2])

développement des processeurs est plus rapide que celle des technologies de la mémoire.

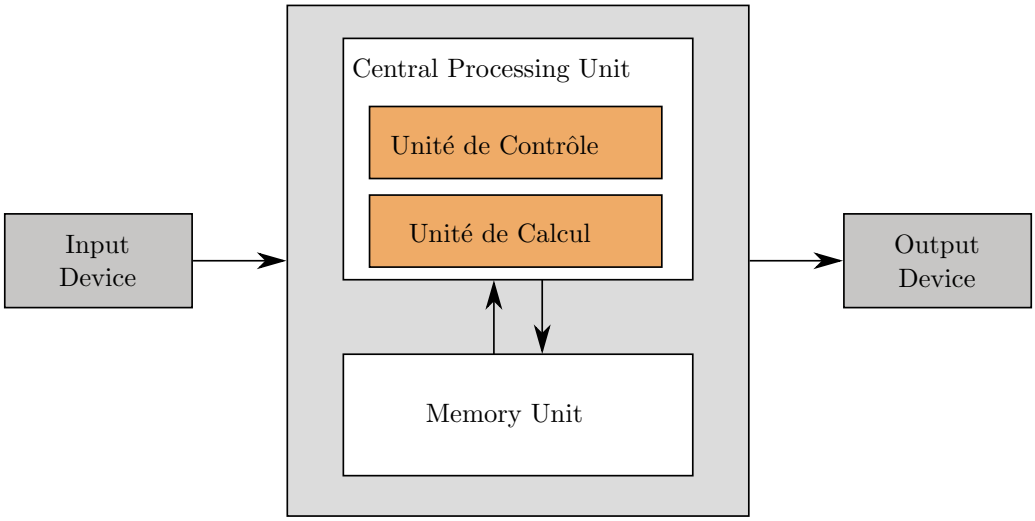


FIGURE 1.4 : Architecture de Von Neumann

Ce « Memory Wall » engendre de nombreux problèmes dans les systèmes électroniques.

Comme la différence de performance entre les mémoires et les processeurs continue de s'accroître, les CPU ont souvent besoin d'attendre pour obtenir les données nécessaires au bon déroulement du programme. Cette attente ralentit le programme et nuit aux performances du système complet. D'autre part, comme le processeur est fréquemment obligé d'attendre pour finir le programme, cela augmente la consommation d'énergie tout en ralentissant son exécution. En effet, le processeur est toujours allumé même s'il ne traite pas activement des données. Cela nuit donc à la durée de vie d'éventuelles batteries ce qui, dans le cas de l'IoT, pose un sérieux problème. À l'heure actuelle, le transfert de la mémoire à l'unité de calcul est ce qui consomme le plus d'énergie dans les systèmes [3].

Cette contrainte est aussi présente dans les systèmes de grande taille comme les centres de données, car les systèmes de mémoire ont plus de transferts à effectuer du fait d'un plus grand nombre de processeurs. La solution actuelle pour palier au « Memory Wall » est l'ajout de mémoires cache, plus rapides que les mémoires principales utilisées. Cependant, ces mémoires conçues pour être plus proches de la zone de calculs sont plus petites et onéreuses que les mémoires principales.

De plus, cette technique n'est pas une solution miracle car il peut toujours y avoir des goulots d'étranglement, surtout dans le cas où la quantité de données travaillées excéde la capacité de la mémoire cache. Ce problème est un défi significatif dans l'élaboration des architectures et il est donc nécessaire de trouver des solutions efficaces pour palier à ces limitations.

1.2 L'émergence d'un nouveau paradigme : la Logique en Mémoire (LiM)

Pour surmonter ces défis causés par le « Memory Wall », de nombreuses solutions sont explorées. L'une d'entre elles est le paradigme Logique en Mémoire (LiM). Cette approche de calcul est apparue dans les années 70 [19]. Le principe est de concevoir une cellule mémoire capable d'effectuer des opérations logiques ou bien, lorsque le système le permet, de réaliser des calculs plus complexes. Cette technique palie efficacement au « Memory Wall » en supprimant le besoin de transférer certaines données. Ce paradigme a beaucoup gagné en popularité ces dernières années [20] grâce aux technologies émergentes non-volatiles qui permettent une intégration étroite avec les technologies CMOS telles que le transistor à effet de champ ferroélectrique (FeFET) [21] ou les jonctions tunnel magnétiques à transfert de spin (STT-MTJs) [22]. Dans le projet SECRET, l'objectif est notamment de concevoir des opérateurs non-volatiles de Logique en Mémoire à base de FeFET [5].

1.3 Le défi de créer de nouveaux opérateurs dédiés à ces nouvelles technologies

Il est impératif de tester les nouveaux dispositifs LiM qui seront conçus sur la base de la technologie non-volatile. Il existe 2 méthodes principales pour tester ces opérateurs : la création de démonstrateur et la simulation.

La création de démonstrateurs pour les nouveaux opérateurs possède plusieurs inconvénients. Le fait que des technologies émergentes soient utilisées demande plus d'efforts pour la préparation des démonstrateurs. Cela s'explique par l'introduction de nouveaux matériaux, dont les exigences techniques diffèrent de celles des transistors conventionnels. L'utilisation de nouvelles techniques pour faire les transistors vient donc rajouter des étapes dans la réalisation et donc engendre un coût supplémentaire [23, 24].

L'autre solution, la simulation, permet d'enlever le coût et le temps pris par le routage ainsi que la création des démonstrateurs car tout se fait à l'aide de logiciels. La simulation présente malgré tout un autre problème. Cette technique de test est très pratique pour de petits systèmes et pour évaluer les opérateurs de manière individuelle du fait de sa rapidité de mise en place. Cependant, pour simuler un système complet, qui est ce que l'on recherche avec le projet SECRET. Cette solution n'est pas viable car le fait de simuler à bas niveau un processeur complet est très chronophage. En effet, d'après [25], il faut un total de huit heures de simulation pour obtenir le fonctionnement d'un processeur sur une seconde.

Compte tenu des inconvénients de ces deux méthodes, nous allons choisir une troisième solution : l'émulation.

L'émulation est la reproduction du fonctionnement d'un composant électronique simplement à l'aide d'un modèle logiciel qui sera inclus dans un système physique. Cela permet dans le cas du projet SECRET de préparer un modèle des opérateurs à évaluer et de les inclure sous la forme d'une nouvelle instruction dans un système composé d'un processeur RISC-V lui même inclut dans une carte électronique FPGA .

1.4 Les objectifs de la Thèse

Cette thèse s'inscrivant dans le cadre du projet SECRET, ses objectifs sont alignés avec ce dernier. La première étape du projet SECRET est d'étudier l'intégration d'opérateurs non-volatiles au sein d'une unité de calcul sans payer le prix d'un démonstrateur sur puce, ni de prendre le temps d'une simulation bas niveau d'un système complexe. Cette thèse s'est donc focalisée sur le développement d'une plateforme d'émulation matérielle FPGA permettant d'évaluer le fonctionnement d'opérateurs de calcul basé sur des technologies émergentes au sein d'un système complet. Cette évaluation se fait en développant des modules imitant le comportement des opérateurs intégrés dans notre plateforme sous la forme de nouvelles instructions utilisées par un processeur RISC-V.

1.5 Plan du manuscrit de Thèse

Le manuscrit est organisé de manière à présenter les travaux de thèse. Dans le Chapitre 2, l'accent est mis sur l'état de l'art et les informations nécessaires à la compréhension du manuscrit.

Le Chapitre 3 se concentre sur la première partie du travail réalisé pendant la thèse. Il détaille l'approche pour émuler les transistors ferroélectriques (FeFET), comprenant la présentation du modèle, le processus de sélection de l'unité de calcul appropriée, et la description de la plateforme dans son ensemble.

Le Chapitre 4 décrit le contexte de l'expérience réalisée et présente les résultats obtenus à la suite de l'utilisation des opérateurs à base de FeFET dans la situation décrite précédemment.

Enfin, le Chapitre 5 conclut le manuscrit en proposant des pistes pour poursuivre le projet de recherche.

Chapitre 2

État de l'art

2.1 Technologie de mémoires non-volatiles émergentes et intégration dans le Logic-In-Memory 13
2.1.1 Technologies de mémoires non-volatiles émergentes 13
2.1.2 In memory Computing :Logic-in-Memory à l'aide des NVM 24
2.2 Processeur : RISC-V 30
2.2.1 Bref historique des architectures des processeurs 30
2.2.2 Présentation de RISC-V 35
2.3 Méthodes d'évaluation des technologies de mémoires non-volatiles. 37
2.3.1 Approches de tests et d'évaluations 37
2.3.2 Comparaison des méthodes et explication du choix pour la thèse . 39
2.4 Conclusions 40

Ce chapitre vient fournir les bases nécessaires pour comprendre le contexte scientifique et technique de cette thèse, ainsi que les choix méthodologiques effectués. Le chapitre présente les éléments de manière croissante, comme présenté dans la figure 2.1, allant de la brique de base que sont les modules de LiM à base de technologies de NVM émergentes jusqu'à présenter le système complet qui est basé sur de l'émulation et qui servira de base à tous nos tests.

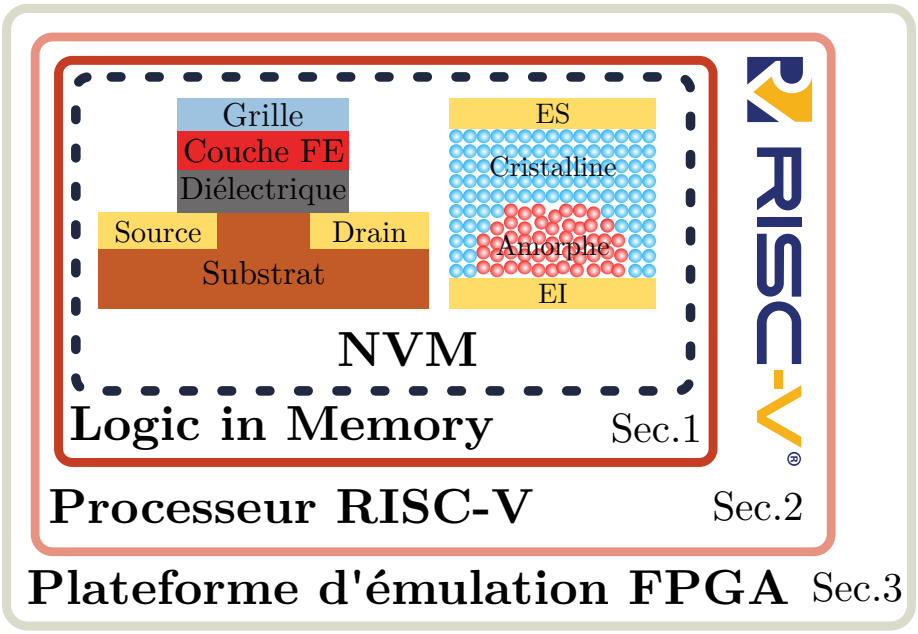


FIGURE 2.1 : Schéma explicatif du plan de l'état de l'art

Le chapitre est formé des trois sections suivantes :

- **Technologies de mémoires non-volatiles et Logic-In-Memory (LiM)**
  Cette section compare les différentes technologies émergentes, avec un focus particulier sur la technologie FeFET utilisée dans le projet SECRET. Elle introduit également le paradigme LiM, en détaillant ses principes théoriques et en présentant des exemples de l'état de l'art.

- **Processeurs RISC-V et leur rôle dans l'intégration**
  Une introduction à l'architecture RISC-V est proposée, avec un aperçu de son historique et de sa structure. L'accent est mis sur l'Instruction Set Architecture (ISA) et sur les avantages qu'elle offre pour l'intégration de nouvelles technologies.

- **Méthodes d'évaluation des technologies**
  Un panorama des approches d'évaluation est présenté, mettant en avant l'émulation

en tant qu'outil clé. Cette méthode est analysée en détail, en soulignant son intérêt et son intégration dans les travaux de la littérature.

2.1 Technologie de mémoires non-volatiles émergentes et intégration dans le Logic-In-Memory

Les progrès technologiques dans le domaine des systèmes électroniques nécessitent des mémoires plus performantes, fiables et à faible consommation d'énergie. Les NVM représentent une solution innovante pour répondre à ces besoins en combinant les avantages des mémoires volatiles et non-volatiles classiques. Cependant, comme leur nom l'indique, ces mémoires émergentes sont toujours en développement et les applications pratiques ne sont pas encore totalement exploitées ou explorées ; leur adoption à grande échelle se heurte à des défis techniques, économiques et architecturaux. Cette partie explore les diverses technologies NVM, leurs caractéristiques distinctives et leur potentiel d'intégration dans les architectures Logic-In-Memory (LiM), qui visent à réinventer les paradigmes du calcul et du stockage des données.

2.1.1 Technologies de mémoires non-volatiles émergentes

2.1.1.1 Introduction aux mémoires non-volatiles émergentes

Ces technologies sont appelées non-volatiles car, lorsque le système de mémoire n'est plus alimenté en énergie, l'information contenue dans les systèmes mémoires n'est pas perdue, à l'opposé des mémoires vives classiques qui, elles, se vident lorsque la source d'alimentation est retirée. Un exemple courant de chacune des mémoires est, par exemple, la mémoire RAM, rapide, contenue dans les ordinateurs pour la mémoire volatile et les mémoires des disques durs qui, eux, sont des exemples de mémoires non-volatiles, mais qui sont lentes.

Comme on peut voir sur la Figure 2.2, il y a d'un côté les technologies résistives qui fonctionnent grâce à un changement de résistivité des matériaux qui les constituent. C'est-à-dire, leur capacité à s'opposer aux courants électriques qui leur sont appliqués. Cette variation est due à un changement de la structure du matériau, de façon pérenne et réversible. C'est ce changement d'état et sa réversibilité qui permettent d'utiliser ces composants pour le stockage d'information. Les principaux représentants de cette famille sont les technologies basées sur ce qui est appelé un memristor [26] : la Résistive RAM [27] (RRAM) qui possède elle-même des sous-classes, La mémoire vive magnétique (MRAM, pour Magnetoresistive RAM en anglais) [28] et les mémoires à changement de phase (PCM pour Phase Change Memory en anglais) [29]. De l'autre côté, la famille ferroélectrique utilise des matériaux du même nom dont la particularité est qu'ils soient naturellement polarisés. Ce phénomène physique a été découvert en 1921 [30]. Les applications de ces matériaux sont nombreuses, notamment dans les domaines de l'électronique et des dispositifs mémoires [31]. Mais c'est

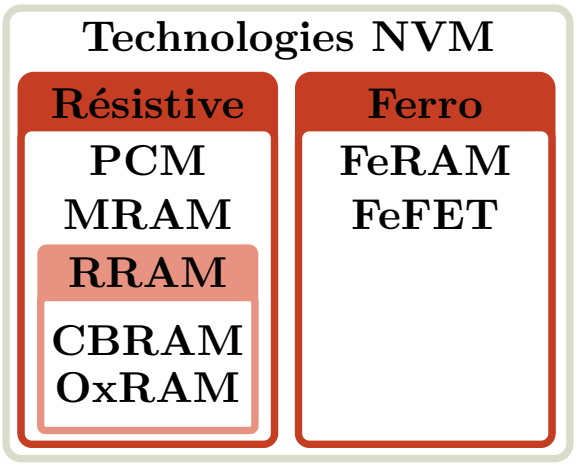


FIGURE 2.2 : Les différentes catégories de NVM

principalement en 2011 grâce à la découverte de la ferroélectricité dans l'oxyde d'Hafnium [32, 33] que les mémoires ferroélectriques ont commencé à être un candidat sérieux pour les mémoires émergentes non-volatiles du fait de la compatibilité avec le procédé de fabrication des circuits CMOS (pour Complementary Metal-Oxide-Semiconductor ou encore semi-conducteur à oxyde métallique complémentaire en français) [34].

De nombreux exemples sont présents dans la littérature pour les intérêts que ces matériaux peuvent avoir. En effet, les NVMs à base de ces matériaux montrent des gains de performance significatifs par rapport aux systèmes classiques, en particulier dans des environnements à faible consommation d'énergie comme le montre la Figure 2.3.

Cette figure issue de [35] montre en effet un fort gain de temps de calcul pour le processeur non-volatile dans un contexte de système IoT fonctionnant uniquement à l'aide de récupération d'énergie passive. Cette nette amélioration est obtenue grâce au fait que le rechargement du contexte est géré dans un environnement plus proche. Grâce à la non-volatilité, les calculs intermédiaires sont préservés, contrairement aux processeurs volatils classiques qui nécessitent souvent de recommencer les calculs depuis le début. Par conséquent, cela permet de réaliser des économies de temps de calcul et d'énergie pour le processeur.

2.1.1.2 Présentation des mémoires non-volatiles émergentes les plus populaires

Les technologies présentées dans la figure 2.2 vont être détaillées ici, une à une, avec une explication du phénomène physique derrière le fonctionnement du système mémoire et des exemples de la littérature scientifique. Les technologies sont présentées sans distinction des familles de mémoires auxquelles elles appartiennent.

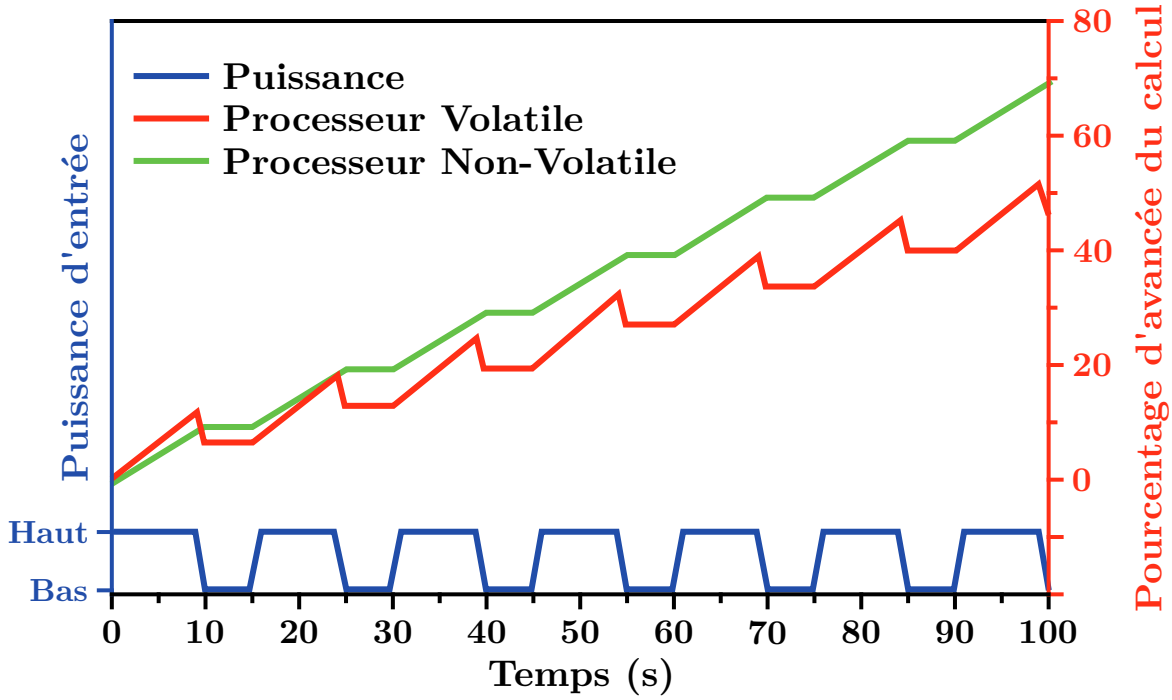


FIGURE 2.3 : Différence de progression entre un processeur volatile et un processeur Non-volatile dans le contexte d'un environnement à faible apport d'énergie

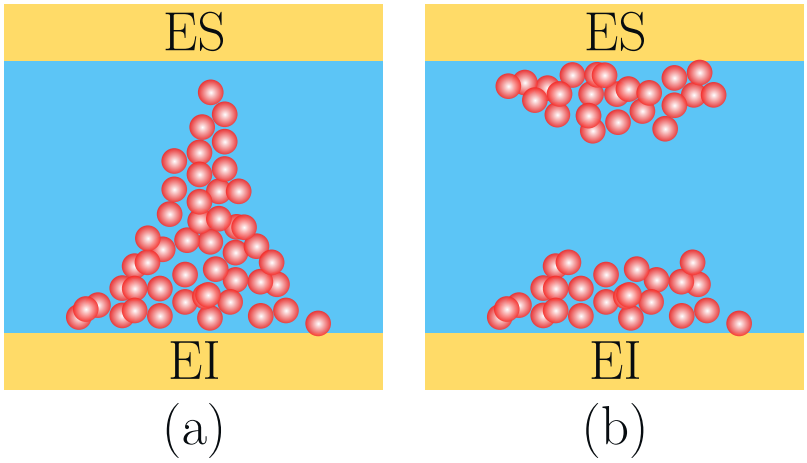


FIGURE 2.4 : (a) : Schéma représentatif du canal formé à l'intérieur de la couche isolante dans une cellule RRAM et (b) : Schéma du même canal dans l'état reset

RRAM (Resistive RAM) : Technologie à base de changement de résistance Les mémoires de type RRAM utilisent le principe de changement de résistance de certains matériaux. On peut simplement décrire la mémoire RRAM comme un sandwich formé de deux électrodes, l'électrode supérieure (ES) et l'électrode inférieure (EI), au milieu desquelles se situe un matériau isolant. La résistance de ce matériau varie entre des états de haute et basse résistances de manière contrôlée afin de stocker des données de façon non-volatile. Comme représenté sur la Figure 2.4, ce changement de résistances se manifeste par l'apparition d'un filament conducteur. Ce changement de résistance est dû à l'application d'une contrainte externe via les électrodes sur la couche isolante qui va influer sur la taille du chemin formé par le filament conducteur et donc augmenter ou diminuer la résistance de la RRAM [28]. La mémoire n'est pas limitée à deux états, haut et bas, il est possible de jouer avec le développement du filament afin de faire un système de stockage multibits [36]. Dans la littérature, les RRAM sont fréquemment classées en deux catégories : les mémoires à accès aléatoire à oxyde métallique (de l'anglais Metal Oxide Random Access Memory (OxRAM)) [37] et les mémoires à accès aléatoire à pont conducteur (Conductive Bridge Random Access Memory [38] (CBRAM)).

La littérature présente des exemples de technologies RRAM dont les performances excèdent celles des technologies conventionnelles. Par exemple, la puce M3D-LIME [39], une intégration monolithique tridimensionnelle (M3D) de mémoire vive résistive (RRAM), a été développée pour l'apprentissage en un coup qui est une technique utilisée dans l'intelligence artificielle. Elle se compose d'une couche de contrôle CMOS (130 nm) pour la gestion des données, d'une couche de calcul en mémoire (Computing In Memory, CiM) avec une RRAM analogique pour l'extraction de caractéristiques, et d'une couche mémoire avec une RRAM numérique et des CNFETs (Carbon Nanotube Field Effect Transistor) pour le stockage et

la correspondance de modèle. Appliquée à l'apprentissage en un coup, la puce a atteint une précision de 96% sur le jeu de données Omniglot, ce qui est équivalent aux solutions actuelles à base de processeur graphique (Graphic processing unit, GPU), avec une efficacité énergétique 18,3× supérieure et un traitement 2,73× plus rapide que les conceptions 2D conventionnelles.

Dans un autre exemple prenant le cas de l'IoT qui exige des systèmes à ultra-faible consommation [40], il est noté le fait que les FPGAs, bien que flexibles, restent limités par leur efficacité énergétique. Ainsi, cette étude analyse une architecture FPGA exploitant la RRAM évaluée via un flux OpenFPGA modifié et des simulations SPICE. Comparée aux FPGAs basés sur les mémoires RAM statiques (Static RAM, SRAM), cette solution réduit la taille de la surface (8%), améliore le temps de calcul (22%) et la puissance (16%) sous tension nominale. En régime proche de Vt, elle double l'efficacité énergétique sans impact sur les performances. La RRAM réduit la consommation en veille et optimise les multiplexeurs de routage. Une analyse de robustesse et des simulations de Monte Carlo confirment sa tolérance aux variations jusqu'à 30%, faisant des FPGAs RRAM une solution prometteuse pour l'edge computing.

PCM (Phase-Change Memory) : Stockage via changement de phase Les mémoires basées sur le changement de phase d'un matériau, connu en anglais sous le nom de « Phase Change Memory » [29] (PCM) sont des matériaux capables de passer d'une phase désordonnée dite amorphe à une phase ordonnée comme une structure cristalline. Chacune de ces phases dispose de résistances électriques différentes, avec la phase amorphe qui a une forte résistivité électrique mais une faible réflectivité optique, tandis que la phase cristalline présente une faible résistivité mais une forte réflectivité. L'utilisation des propriétés de réflectivité des matériaux à changement de phase est assez ancienne, avec notamment la création de dispositifs de stockage comme les Blu-Ray ou des DVDs [41]. La Figure 2.5 illustre le changement de phase à l'intérieur d'une cellule mémoire PCM. Il y a deux actions possibles sur une cellule : la cristallisation qui correspond au passage de la phase amorphe à la phase cristalline et l'amorphisation, inverse de la cristallisation. D'autres termes comme « potentation » et « dépression » [42] peuvent être utilisés pour parler de ces changements.

Comme montré sur la figure 2.5 une cellule PCM est composée d'un volume actif de matériau à changement de phase pris en sandwich entre deux électrodes, un peu à la manière de la RRAM. L'information est stockée dans les mémoires PCM en utilisant le contraste de résistance entre la phase cristalline hautement conductrice et la phase amorphe. Cette information est récupérée en mesurant la résistance électrique du composant.

Les caractéristiques intéressantes des PCM sont la durée de rétention des données (autour de 10 ans à température ambiante) ainsi que la vitesse d'écriture dans cette mémoire (en quelques nanosecondes) [43]. Ces deux points permettent aux PCM d'être utilisées pour le stockage non-volatile tout en fonctionnant à une vitesse équivalente aux mémoires volatiles à haute performance comme les mémoires vives dynamiques (DRAM pour Dynamic

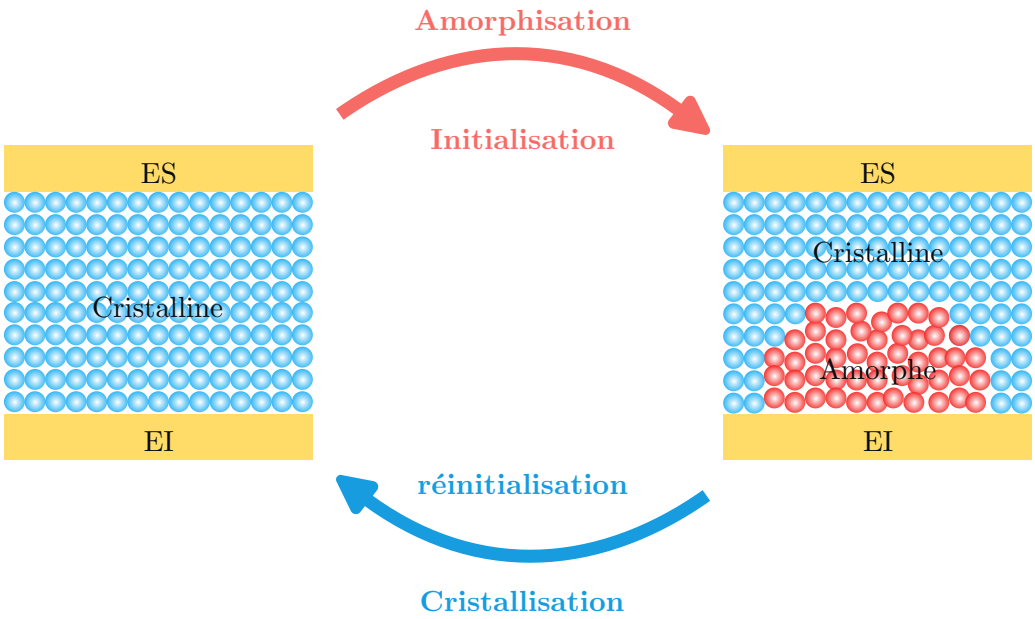


FIGURE 2.5 : Illustration du changement de phase d'une Cellule PCM

Random Access Memory).

Les PCM sont utilisées pour l'inférence d'apprentissage profond [44] car ces dernières permettent la multiplication matricielle. Pour ce faire, les poids synaptiques sont réalisés en faisant varier les résistances des cellules PCM. Les cellules de mémoire à changement de phases sont aussi utilisées pour la formation de Binary Neural Network. Dans [45] les auteurs ont proposé une architecture utilisant ces cellules en représentant les poids synaptiques entre chaque neurone à l'aide de la conductance combinée de N dispositifs PCM. Dans le réseau, une tension d'entrée correspondant à l'activation neuronale est appliquée à toutes les cellules mémoires et la somme des courants de sorties des N dispositifs forme la sortie synaptique. Pour effectuer l'apprentissage du réseau, une cellule est sélectionnée et programmée à la fois (sélection avec un arbitrage basé sur un compteur).

, pour les réseaux de neurones binaires à l'aide de PCM est présentée. Dans cet article

Dans [46], une nouvelle méthode de mappage des données pour les réseaux de neurones binaires à l'aide de PCM, nommée TacitMap, est proposée pour une architecture de calcul en mémoire, maximisant le parallélisme tout en réduisant les transferts de données. Un accélérateur matériel, EinsteinBarrier, basé sur la mémoire PCM optique, est également introduit. L'auteur de l'article explique qu'il exploite TacitMap et intègre un multiplexage en longueur d'onde pour améliorer encore la latence. Les simulations révèlent que, par rapport à l'architecture de référence utilisée [47] dans l'article, TacitMap et EinsteinBarrier réduisent considérablement le temps d'exécution, jusqu'à environ 78 fois et 1205 fois res-

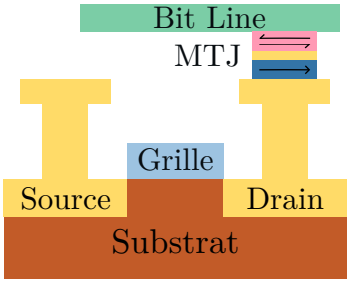


FIGURE 2.6 : Schéma illustrant une cellule de STT-MRAM

pectivement, tout en maintenant la consommation énergétique à 60% de celle du modèle de base.

MRAM (Magnetoresistive RAM) : Utilisation de la magnétorésistance Les cellules de mémoire sont basées sur un composant qu'on appelle une jonction tunnel magnétique (MTJ pour Magnetic Tunnel Junction) [48] qui permet l'utilisation d'un phénomène physique appelé la magnétorésistance à effet tunnel. Ce composant est constitué d'une couche ferromagnétique (FM) fixe qui est appelée « couche de référence », d'une couche fine d'oxyde et d'une couche ferromagnétiques reconfigurable à l'aide d'un courant électrique qui est appelée « couche libre ». En fonction de l'orientation de la couche libre, soit la MTJ est dans ce qu'on appelle le mode parallèle, lorsque les deux couches sont orientées dans le même sens, soit la MTJ est dans le mode anti-parallèle, lorsque les deux couches ont une orientation magnétique opposée. Afin qu'un électron puisse passer la barrière d'oxyde séparant les deux couches FM, son orientation doit être la même que celle de la couche FM située de l'autre côté de la barrière. C'est-à-dire que si la couche FM est polarisée dans le sens opposé (donc anti-parallèle) par rapport à la couche dont provient l'électron, alors celui-ci aura plus de mal à traverser le tunnel. La conductance de la MTJ est plus faible, donc la résistance de la MTJ est plus grande.

Dans la MTJ, lorsque les électrons circulent de la couche de référence vers la couche libre, les spins de ces derniers se polarisent dans la couche de référence et peuvent ainsi changer la direction de l'aimantation de la couche libre, ce qui peut faire basculer l'aimantation de la couche libre dans l'état « 0 » du fait du transfert du moment angulaire du spin vers la couche libre. L'inversion de la direction du courant peut faire passer la couche libre dans l'état inversé « 1 ».

Afin de compléter une cellule MRAM, on associe ce composant de Tunnel avec un transistor. Cela le transforme plus précisément dans un composant appelé RAM à couple de transfert de spin magnétorésistif (STT-MRAM pour Spin-Transfer Torque MRAM en anglais). Dans la Figure 2.6, on peut voir une telle cellule où le MTJ est disposé en « Back End of Line », du fait du placement à la fin de la conception d'un transistor, le situant au sommet du transistor et relié à une Bit-Line.

La STT-MRAM fait partie des mémoires les plus prometteuses. En effet, elle pourrait être utilisée comme unité de mémoire autonome, dans [49] le dispositif a montré une endurance en écriture capable de rivaliser avec les mémoires traditionnelles et des temps de lecture compris entre 30 et 100 nanosecondes. Les STT-MRAMs sont souvent utilisées pour remplacer les mémoires SRAM et DRAM qui nécessitent une alimentation pour conserver les données stockées.

De même que les RRAMs, les STT-MRAM sont intéressantes pour accélérer l'apprentissage rapide dans les réseaux neuronaux binaires. Dans [50], une architecture de calcul en mémoire pour les réseaux neuronaux binaires fonctionnant à l'aide de STT-MRAM est proposée, exploitant une accumulation directe des produits via la tension des lignes sources afin d'optimiser l'utilisation de la matrice et d'améliorer le débit. Contrairement aux approches traditionnelles, où la détection repose sur le courant des lignes de bits, cette méthode réduit la consommation énergétique et améliore l'efficacité du calcul. Pour renforcer la robustesse face aux variations de processus, une détection temporelle (TBS) et un renforcement des lignes de sélection sont introduits, améliorant ainsi la précision du réseau. Les simulations montrent une précision de 80,01% sur CIFAR-10 et 98,42% sur MNIST, avec une perte limitée par rapport aux implémentations logicielles (8,59% et 0,38%, respectivement). L'architecture atteint une efficacité énergétique de 311 TOPS/W (pour Tera Opération par Seconde par Watt) ce qui est une bonne valeur moyenne comparée aux 137,35 TOPS/W d'un réseau de neurones binaire basé sur les RRAM [51] et les 622,35 TOPS/W d'un autre réseau fait à partir de neuristors [52], démontrant ainsi sa pertinence pour les accélérateurs de calcul en mémoire, tout en nécessitant des modifications périphériques minimales.

FeRAM (Ferroelectric RAM) et FeFET (Ferroelectric Field Effect Transistor) : Stockage via utilisation de la ferroélectricité La ferroélectricité est une propriété propre à certains matériaux possédant une polarisation spontanée. Depuis sa découverte en 1921 [30], les applications se sont multipliées. Afin de comprendre la ferroélectricité, il faut connaître les différentes caractéristiques des matériaux qui sont ferroélectriques. Tout d'abord, tous les matériaux ferroélectriques sont des cristaux dotés des propriétés suivantes :

- **La piézoélectricité** : les matériaux ferroélectriques sont tous piézoélectriques, c'est-à-dire qu'ils présentent tous une polarisation lorsqu'ils sont soumis à une contrainte mécanique ou électromagnétique.
- **Pyroélectricité** : une sous-classe des cristaux piézoélectriques. Chacun de ces cristaux présente un axe de polarisation unique, ils présentent une polarisation spontanée due à la présence d'un moment dipolaire au sein des mailles des cristaux. On peut observer ce phénomène en chauffant la surface du cristal, ce qui provoque une circulation de charges depuis la surface vers une autre surface.

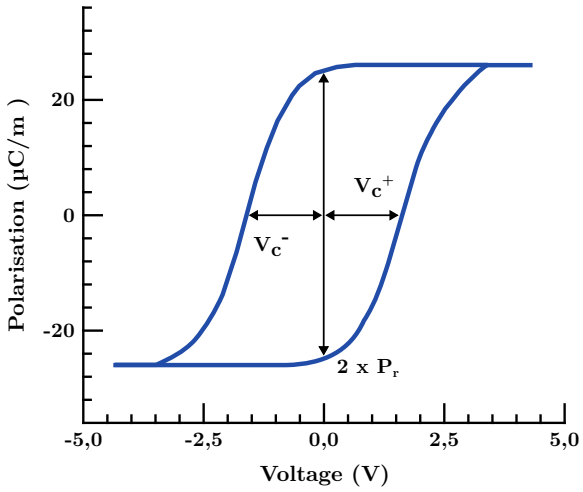


FIGURE 2.7 : Illustration de l'hystérésis du matériau ferroélectrique

- **Ferroélectricité** : les cristaux ferroélectriques sont des cristaux diélectriques, piézoélectriques et pyroélectriques avec deux états d'orientation en présence d'un champ électrique extérieur. On peut définir un matériau férroélectrique comme un matériau possédant une polarisation spontanée dont la direction peut être modifiée lorsque ce dernier est soumis à un champ électrique suffisamment fort.

Ainsi, lorsque ces matériaux sont chauffés à une température appropriée (on appelle cette température la température de Curie), ils vont effectuer un changement de polarisation. Cependant, cette polarisation ne se fait pas de manière homogène ni de manière instantanée pour la polarisation rémanente. Lors du processus de polarisation, il y aura à l'intérieur du matériau une cohabitation de différentes polarités. On appelle ces différentes régions des domaines ferroélectriques et la frontière qui sépare ces domaines est appelée paroi de domaines. Dans la littérature, on peut trouver de nombreuses applications [53, 54] pour ces parois. Il est possible d'impacter leur position et donc, par exemple, la conductivité de ces matériaux [55]. La polarisation globale laisse apparaître une hystérésis dans la courbe $P = f(V)$ comme montrée dans la Figure 2.7.

Les matériaux ferroélectriques servent notamment à créer deux types de mémoires, les mémoires FeRAM ainsi que les FeFET.

Les mémoires RAM Ferroélectriques :

Ces mémoires sont les premières applications effectuées à l'aide des matériaux ferroélectriques. Pour créer ces registres à base de mémoires RAM ferroélectriques, il faut remplacer la couche diélectrique du condensateur d'une cellule modifiée de RAM dynamique (DRAM), composée d'un transistor et d'un condensateur (1T-1C), par une couche d'un matériau ferroélectrique. Le concept de la FeRAM n'est pas nouveau parmi

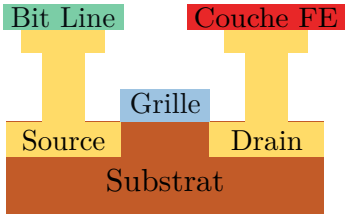


FIGURE 2.8 : Schéma d'une FeRAM composée d'un transistor et un condensateur (1T-1C) issue de [56]

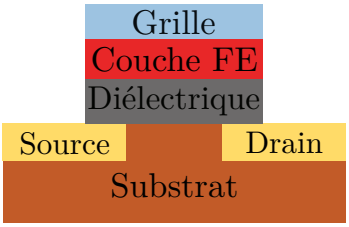


FIGURE 2.9 : Schéma classique d'une cellule FeFET

les technologies de mémoires émergentes non-volatiles [57]. Cependant, l'objectif actuel des chercheurs est d'obtenir des cellules NVM dont les performances ainsi que la taille seraient similaires à celles des DRAM pour obtenir une véritable RAM non-volatile [32].

Les Tansistors à Effet de Champ Ferroélectrique (FeFET) :

Depuis la découverte de la ferroélectricité dans l'oxyde d'hafnium en 2011 [33], l'intérêt pour les technologies de mémoires non-volatiles a fortement augmenté. Cela s'explique notamment par le fait que l'oxyde d'hafnium permet de réaliser des FeFET compatibles avec les procédés CMOS [34]. Cette découverte permet de faire des FeFETs une solution de NVM compétitive du fait de la possibilité de réaliser des portes logiques ainsi que des cellules mémoire à l'aide d'un seul transistor [5]. La particularité des FeFETs est l'ajout d'une couche d'un matériau ferroélectrique au niveau de la grille du transistor. Cela fait que ce dernier peut fonctionner de deux manières : un mode non-volatile, qui se base sur l'hystérésis présenté Figure 2.7 et un mode de commutation qui peut ne pas avoir d'hystérésis [58]. Les FeFETs sont particulièrement attractifs en tant que mémoire grâce à leur capacité de rétention de la polarisation, leur rapport de courant élevé entre les deux états et leur faible tension de fonctionnement. En tant que dispositifs à trois bornes, ils permettent de séparer les chemins de lecture et d'écriture, optimisant ainsi ces opérations simultanément. Contrairement à la FeRAM, dont la lecture est destructive et nécessite une réécriture, les FeFETs offrent un accès non destructif, ce qui les rend plus adaptés aux applications nécessitant des lectures fréquentes [43].

Le FeFET est un dispositif de stockage non-volatile particulièrement intéressant.

Comme proposé dans [59] où il est présenté une architecture de tableau de correspondance (Look-up Table, LUT), le système est organisé en rangées composées de N FeFET, avec un transistor par bit. Les bornes de source des FeFET sont connectées à une ligne de source commune, tandis que leurs grilles sont reliées à une ligne d'écriture/lecture partagée. Selon des simulations basées sur un modèle prédictif FeFET 45 nm, l'utilisation du stockage 1T permettrait de réduire le nombre de transistors d'environ 63%.

2.1.1.3 Comparaison des différentes technologies de NVM

Le tableau 2.1 propose une comparaison des différentes technologies non-volatiles entre elles ainsi qu'avec les technologies déjà existantes. Le tableau regroupe les informations sur le fonctionnement de ces mémoires à partir d'un certain nombre de textes, avec comme éléments centraux le tableau et les textes issus de [43, 60, 61] ainsi que quelques compléments ajoutés via les articles [42, 62-69].

Technologie	eSRAM	eDRAM	FG Flash	SONOS	RRAM (OxRAM)	PCM	STT-MRAM	FeRAM	FeFET
Structure	6T	1T-1C	1,5T	2T	1T-1R	1T-1R	1T-1R	1T-1C	1T
Taille de la cellule	120-150F²	40F²	50F²	60F²	60F²	60F²	50F²	50F²	20-30F²
Ratio On/Off	N/A	N/A	$> 10^4$	$> 10^4$	10^6	10^6	10	10^4	10^6
Energie/bit	∼1fJ	∼1pJ	100pJ	10pJ	∼1pJ	10pJ	∼100fJ	∼100fJ	∼1fJ
Temps d'écriture	<1ns	>10 ns	0.1-1ms	10-100ns	<10ns	∼ 50ns	>10ns	>10ns	∼ 1ns
Endurance	10^{16}	10^{10}	10^4-10^5	10^4-10^6	10^5-10^8	10^6-10^7	10^{15}	10^{14}	10^5-10^9
Rétention	Volatile	Volatile	10 ans	10 ans	10 ans	10 ans	10 ans	10 ans	10 ans

TABLE 2.1 : Tableau comparatif des différentes technologies de mémoire avec les technologies disponibles indiquées en vert, celles en conception en orange et celles à l'étude en rouge

On peut voir dans le tableau Tab 2.1 un récapitulatif des technologies utilisées de manière générale pour les systèmes embarqués, noté par le « e » devant les technologies pour la SRAM ainsi que la DRAM. Ces deux technologies n'ont pas les mêmes usages dans les systèmes, avec la DRAM, plus lente et dense, utilisée comme mémoire principale et la SRAM, plus rapide, utilisée pour les caches CPU. Ensuite, les technologies FG (à Grille Flottante) Flash [70] ainsi que SONOS [71] sont des technologies non-volatiles classiques n'utilisant pas de technologies émergentes. On peut voir que bien qu'elles soient intéressantes pour cet aspect, ces technologies ont des points faibles assez importants, comme le temps d'écriture pour la FG Flash, ou bien encore l'endurance qui est plus faible que les endurances atteignables via l'utilisation des technologies émergentes. Ces technologies ont cependant l'intérêt d'avoir une surface de cellule bien inférieure à celle de la SRAM et donc permettent le stockage de plus d'informations sur une surface équivalente. Le Ratio On/Off, qui compare la valeur en sortie entre le moment où la cellule est « Allumée » et « Éteinte » à savoir le moment où on lit soit un 0 soit un 1, est aussi intéressant, car il permet de savoir à quel point la distinction entre ces deux valeurs est simple pour le

système et, pour ces deux technologies, le ratio a une valeur très comparable aux valeurs des technologies émergentes, ce qui indique une utilisation simple. Ce qui vient freiner l'utilisation de ces technologies est la faible endurance dont elles font preuve, notamment dans le contexte actuel où l'utilisation de données est intensive. C'est pour cela que les technologies non-volatiles émergentes sont intéressantes. On voit que pour les technologies présentées, l'endurance est d'au moins un ordre de grandeur supérieur aux technologies SONOS et FG Flash. On voit aussi qu'en termes d'espace utilisé, ces technologies sont plus compactes que les technologies classiques. La STT-MRAM, bien que très compétitive sur les temps d'écritures et en terme d'énergie par bits, est un peu en retrait sur son Ratio On/Off qui est très faible. Les autres technologies que sont la OxRAM et les PCM sont moins compactes, avec les $60F^2$ par rapport aux 20-$30F^2$ des FeFETs et les $50F^2$ de la FeRAM. Enfin, ces deux technologies ont aussi une consommation d'énergie par bit et un temps d'écriture d'au moins un ordre de grandeur au-dessus des technologies FeRAM et FeFET.

Mais ce qui démarque les technologies ferroélectriques des autres est l'endurance avec des valeurs plus que correct (10^{14} pour la FeRAM et 10^9 pour les FeFETs) ce qui en fait des technologies robustes avec les utilisations actuelles. Il faut ajouter à cela le fait que les FeFETs ont un processus de lecture non-destructif, ce qui permet de ne pas avoir à réécrire le résultat à chaque utilisation. De plus, on constate aussi que les transistors ferroélectriques consomment une énergie de l'ordre du femtojoule pour les écritures, tout en ayant un temps d'écriture équivalent à la technologie SRAM qui est la plus efficace si on oublie le côté rétention de l'information. La fabrication des cellules FeFETs est aussi simplifiée du fait que la ferroélectricité avec le hafnium est compatible avec les processus CMOS existants, contrairement aux autres matériaux qui nécessitent des processus plus complexes pour leur fabrication.

Ces avantages font des technologies ferroélectriques des candidates idéales pour de nouvelles architectures orientées traitement des données. La suite des travaux dans la thèse, conjointement au projet SECRET, se fera en utilisant des technologies ferroélectriques basées sur la technologie 28 nm SLP de GLOBALFOUNDRIES comme technologie principale pour nos tests [21]. C'est dans ce contexte que s'inscrit la suite de cette étude, qui s'intéresse à l'intégration de ces mémoires dans des architectures de logique en mémoire (Logic-in-Memory, LiM).

2.1.2 In memory Computing :Logic-in-Memory à l'aide des NVM

2.1.2.1 Introduction au In-memory Computing et au LiM

Pour comprendre le LiM, il faut tout d'abord définir le plus grand groupe auquel ce paradigme appartient, qui est ce que l'on peut appeler les architectures centrées sur la mémoire. Ces architectures sont classées selon deux critères : tout d'abord la distance entre les éléments de mémoire, puis les éléments de calcul et les technologies utilisées pour

fabriquer ces éléments de mémoire. Grâce à ces critères, on peut distinguer deux catégories d'architecture, à savoir celle pour le calcul proche mémoire (NMC pour Near Memory Computing en anglais) où les unités de calcul sont proches, mais hors de la mémoire, et l'architecture calcul en mémoire (IMC pour In Memory Computing en anglais) où cette fois-ci les calculs sont embarqués dans les zones mémoire du cœur. On peut détailler le modèle un peu plus avec cette fois-ci quatre catégories :

- **Le calcul en mémoire (CiM pour Computing in Memory en anglais) :** Le calcul est réalisé en lisant les données de la mémoire par l'intermédiaire d'amplificateurs de détection. L'accès à la mémoire est modifié afin de prendre en charge le calcul des fonctions booléennes. Par exemple [72], il est possible de faire en sorte que la lecture d'information concerne une colonne de données ou plus au lieu d'une seule. Cela fait que le système détecte comme donnée le résultat d'une fonction booléenne (comme un NOR) des valeurs qui sont stockées en mémoire.
- **Le calcul avec mémoire (CwM pour Computing with Memory) :** C'est par exemple un système qui va venir stocker en mémoire les résultats précalculés et dont les adresses mémoire correspondent aux opérandes [73] .
- **Le calcul proche mémoire (NMC) :** L'unité de calcul se trouve à l'extérieur de l'unité de mémoire. Les opérandes doivent être lus à partir de la mémoire.
- **La logique en mémoire (LiM) :** Il est possible d'accéder aux cellules de la mémoire en mode ≪ calcul ≫ pour réaliser une fonction booléenne donnée, comme le montre l'article définissant ce concept en 1970 [19] ou bien l'architecture Magic qui effectue la même chose à l'aide de memristors [74] .

Le schéma Fig. 2.10 vient résumer ces catégories en une image. Des techniques plus détaillées de classifications ont également été proposées. Dans [75], il est proposé un classement des architectures centrées sur la mémoire suivant l'emplacement de production des résultats dans le système. Dans cette classification, on appelle calcul toute opération logique ou arithmétique élémentaire. Ainsi, comme dans la figure Fig. 2.10, on peut distinguer les calculs selon leur proximité avec la mémoire. Si les calculs sont faits à l'intérieur de la mémoire, on définit l'architecture comme étant CiM. Si le résultat est produit hors de la mémoire, l'architecture est simplement CoM avec le O qui provient de Outside en anglais. Cela est le premier niveau de détail, on peut aller plus loin en détaillant comment sont produites les données. Pour la classe CiM on aura donc, les CiM-A (pour Array) si les données sont produites en matrices par exemple, et pour les CoM on distinguera la distance à la mémoire, avec les CoM-N (N pour Near, proche en anglais) que l'on peut associer aux CnM définies précédemment, et les CoM-F (Far, lointain en anglais). Ensuite, on continue cette classification en précisant quelle technologie de mémoire est utilisée pour l'implémentation des matrices mémoires. Ces mémoires sont celles présentées dans la sous-section sur les mémoires non-volatiles, à l'exception notable des mémoires ferroélectriques,

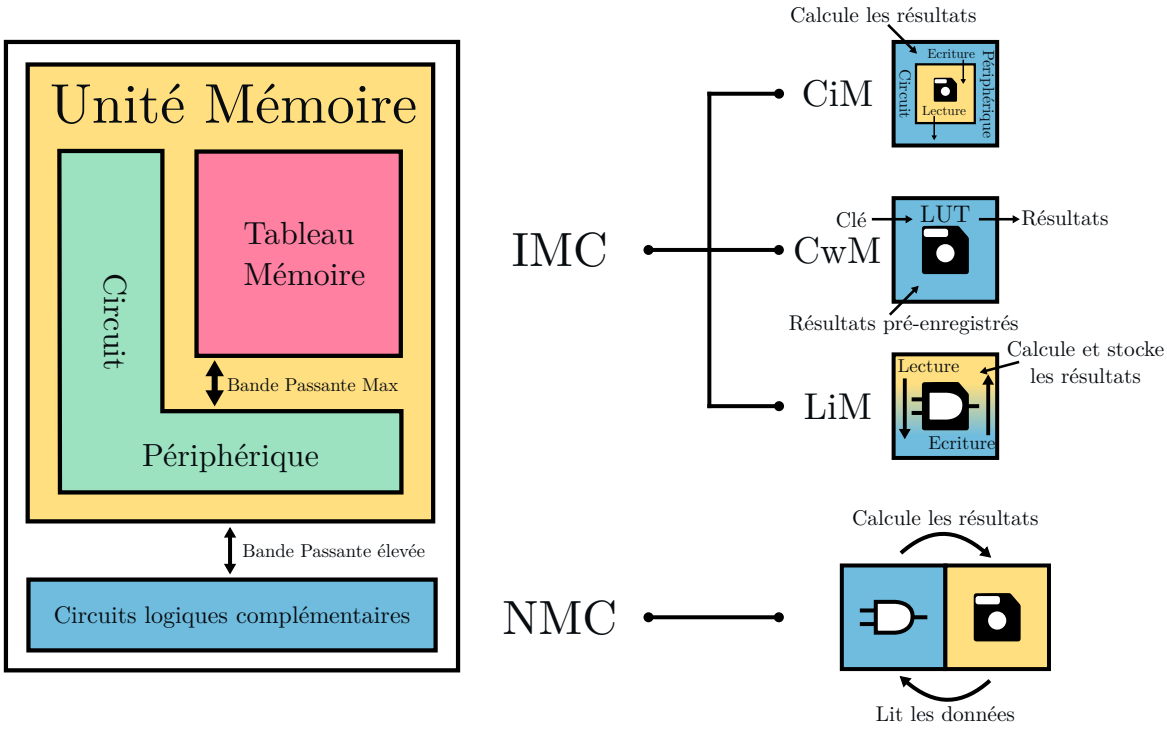


FIGURE 2.10 : Présentation des différents types de calcul en mémoire avec deux principaux paradigmes qui sont le IMC et le NMC. Sur la droite de la figure, on voit que l'IMC est lui-même divisé en trois avec le CiM, le CwM et la LiM

ainsi que des traditionnelles DRAM [76] et SRAM [77]. Finalement, les architectures sont classées selon leur type de parallélisme. On peut en distinguer trois différents. Le premier est le parallélisme de tâche. L'architecture intègre plusieurs unités de contrôle (telles qu'indiquée Fig. 2.12) indépendantes et plusieurs mémoires de données. On retrouve notamment ce type de parallélisme dans les processeurs multi-cœurs actuels pour la gestion de plusieurs tâches concurrentes. On a ensuite le parallélisme de données qui se distingue par des architectures ne contenant qu'une unité de contrôle utilisée pour envoyer la même instruction simultanément sur un ensemble de données. Le troisième et dernier est le parallélisme d'instruction où une seule unité de contrôle est utilisée pour exécuter plusieurs instructions simultanément. Cependant, cette caractérisation n'est pas parfaite non plus. On peut critiquer le fait d'intégrer les technologies utilisées comme critère de classification. En effet, cette dernière ne parle pas des mémoires ferroélectriques. De plus, il faut trois critères afin de pouvoir classer une architecture, mais l'axe technologie est un critère trop large puisque ces technologies peuvent faire partie de plusieurs classes de proximité et de parallélisme.

Maintenant que le concept de CiM a été présenté, il faut raffiner l'analyse par rapport à ce qui sera utilisé pendant la thèse, à savoir la partie de Logic-In-Memory (LiM). Comme précisé précédemment, ce paradigme existe depuis les années 70 [19]. L'objectif principal de ce paradigme est donc de produire des mémoires capables d'effectuer directement des calculs plutôt que de devoir effectuer des transferts importants de mémoire. Cette solution est l'une des approches envisagées pour passer outre le souci du « Mur de la mémoire » [20] tout en réduisant la consommation des circuits [58]. Le regain d'intérêt pour ce paradigme est lié au développement des nouvelles technologies de mémoire émergentes présentées précédemment, car ces dernières permettent de développer des solutions rivalisant en vitesse avec les technologies DRAM et SRAM actuellement utilisées, tout en permettant la non-volatilité.

2.1.2.2 Intégration des NVM dans les architecture LiM

L'intégration des NVM émergentes dans des circuits utilisant le paradigme LiM est un point de départ intéressant pour les défis actuels que sont les applications d'IA qui ont besoin d'une quantité énorme de données, ou encore pour contrer le phénomène du goulot d'étranglement de Von Neumann. En effet, utiliser les nouvelles technologies permet d'avoir de nouveaux outils pour lutter contre ces problèmes.

L'architecture 1T/1R pour les cellules RRAM a notamment permis de construire un cube mémoire d'une grande densité [78]. Les cellules dans ce projet, en associant les cellules OxRAMs à des transistors à feuillets nanométriques empilés (FinFET), permettent d'avoir une densité de cellule de 23,9 F^2/N avec N le nombre de feuillets empilés, ce qui donne la possibilité d'une grande intégration. Le fait que ces cellules puissent se superposer permet à l'architecture d'avoir un haut niveau de parallélisme pour l'écriture ainsi que pour la lecture, ce qui est un critère important pour la Logic-in-Memory. De plus, les simulations

SPICE réalisées dans l'article montrent aussi que l'approche utilisée est environ deux fois plus efficace que les logiques traditionnelles.

Chacune des technologies propose ses solutions, à l'aide d'optimisations des cellules mémoire. Les calculs vectoriels par exemple, peuvent être eux aussi optimisés à l'aide de technologies non-volatiles. La STT-CiM [79] est une utilisation des mémoires STT-MRAM afin d'obtenir une unité de mémoire capable d'effectuer des opérations logiques, vectorielles ou encore arithmétiques. Dans les tests effectués au niveau d'un système complet, les résultats montrent une amélioration des performances d'un facteur 4 tout en réduisant la consommation énergétique du système mémoire d'un même facteur.

Le développement des mémoires non-volatiles a aussi permis l'optimisation de circuits déjà existants et intéressant dans le contexte de LiM. En 2017, une équipe de l'université de Notre Dame aux États-Unis a présenté un design de cellules TCAM (mémoires adressables par contenu ternaire), utilisées par exemple dans les réseaux de neurones, basé sur des FeFETs [80]. Le concept de leur cellule TCAM utilisant 4 transistors et 2 FeFETs est de se servir non pas de différentes valeurs de résistance pour stocker les informations binaires, mais d'utiliser la particularité de la courbe d'hystérésis des FeFET afin de pouvoir stocker de l'information à l'intérieur d'un seul transistor. Cette amélioration permet un gain de surface d'environ 42% lorsqu'elle est comparée à une architecture classique de TCAM composée de 16 transistors CMOS avec des énergies et des délais de recherche de contenu similaires. Cependant, ce n'est pas la seule amélioration que permet ce design, car il offre aussi une réduction de la consommation par rapport à des designs utilisant des RRAM/MTJ de respectivement 7,5 et 149 fois. Cette idée de TCAM basée sur les FeFETs est poussée plus loin dans un autre article présentant cette fois-ci le concept de TC-MEM [81]. L'idée est d'utiliser le fait que les FeFETs ont déjà été utilisés à la fois comme cellule de mémoire pour réaliser de la LiM mais aussi des cellules TCAM. Il est donc présenté ici une cellule faisant office à la fois de cellule mémoire mais aussi de cellule TCAM. Cette cellule a notamment une utilité dans des algorithmes de chiffrement comme l'AES, qui utilisent des tables de transformation communes pour chiffrer et déchiffrer des données.

Dans le circuit présenté Fig. 2.11, il a fallu composer les deux fonctions de mémoire et de mémoire adressable par contenu. Cela est fait en ajoutant un simple transistor nMOS entre les deux FeFETs connectés en parallèle, représenté par l'entrée M. Lorsque l'entrée M est égale à 0, le transistor nMOS est à l'état bas. Cela implique que seul le FeFET 2 peut changer la sortie de la ML (match line pour la TCAM)/ WL (word line pour la mémoire simple). Cela correspond donc au mode mémoire. Quand l'entrée M est à l'état haut (M=1), les FeFETs sont connectés en parallèle et produisent une sortie correspondant à une fonction logique XOR, qui est le composant essentiel d'une mémoire TCAM.

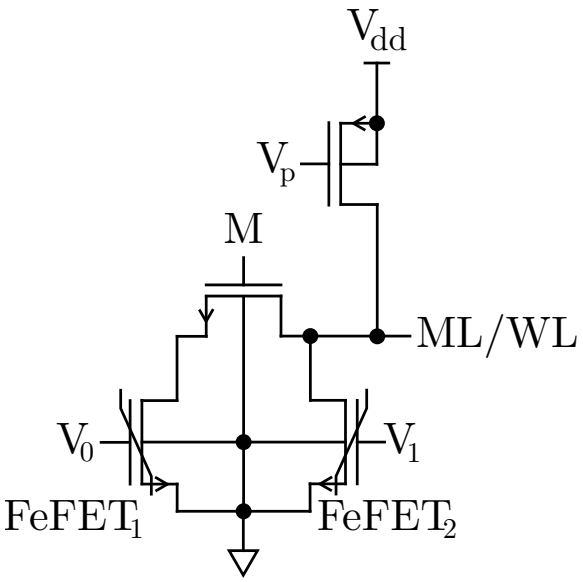


FIGURE 2.11 : Schématique d'une cellule TC-MEM pour un bit

2.1.2.3 Défis et problématiques liés à l'intégration des NVM

L'intégration de ces technologies émergentes permet d'avoir accès à de nouveaux types de circuits comme les TC-MEM ou, comme l'illustrent les autres exemples, d'améliorer les performances de circuits déjà existants. Il faut cependant pour cela passer outre les nombreux défis liés à l'intégration de ces nouvelles technologies. En effet, bien que les performances soient améliorées dans certaines conditions présentées précédemment, les mémoires émergentes non-volatiles ont encore de nombreux points à développer, notamment le fait que ces cellules mémoires sont bien souvent écrites de manière individuelle, ce qui fait que certaines cellules peuvent avoir un nombre d'écritures bien plus important à faire et donc se fatiguer plus vite. Cette contrainte, ajoutée à la faible endurance de certaines technologies pose une limitation importante [82].

De plus, les technologies de mémoires CMOS se sont développées au point d'aller en dessous de 5 nanomètres. Il est donc compliqué d'intégrer de nouvelles technologies non-volatiles à ces systèmes, car ces dernières sont encore à un stade assez immature et de tailles beaucoup plus importantes [83].

Ces points sont deux exemples pour lesquels les mémoires non-volatiles émergentes permettent de combiner des fonctions de stockage et de traitement au sein d'une même unité de mémoire, mais qui ont besoin d'un environnement modulaire pour s'intégrer aux cas d'utilisation actuels. La mise en œuvre concrète des approches Logic-in-Memory (LiM) nécessite donc une articulation étroite entre les structures mémoires émergentes et les unités de traitement, imposant ainsi des exigences spécifiques en matière de flexibilité

architecturale. Dans ce contexte, l'architecture RISC-V, par sa nature libre, modulaire et extensible, offre un cadre particulièrement pertinent pour explorer et intégrer ces nouveaux paradigmes computationnels. La section suivante s'attache donc à présenter les fondements de cette architecture et à en justifier le choix dans le cadre de nos travaux.

2.2 Processeur : RISC-V

Les architectures de processeurs jouent un rôle central dans le développement des systèmes électroniques. Pendant longtemps, les modèles propriétaires, comme ceux développés par Intel (x86) ou ARM, ont dominé le marché. Cependant, ces solutions, bien que performantes, posent des limites significatives, notamment en termes de flexibilité, de coûts et de dépendance technologique. C'est dans ce contexte qu'émergent des initiatives comme RISC-V, une architecture ouverte et modulaire qui transforme le paysage grâce à son accessibilité (licence libre) et sa capacité à s'adapter aux besoins spécifiques des nouvelles technologies. Cette section explore comment RISC-V s'inscrit dans l'évolution des architectures et pourquoi elle représente un atout majeur pour les projets expérimentaux et innovants, en particulier ceux liés aux mémoires non-volatiles et aux systèmes Logic-In-Memory (LiM).

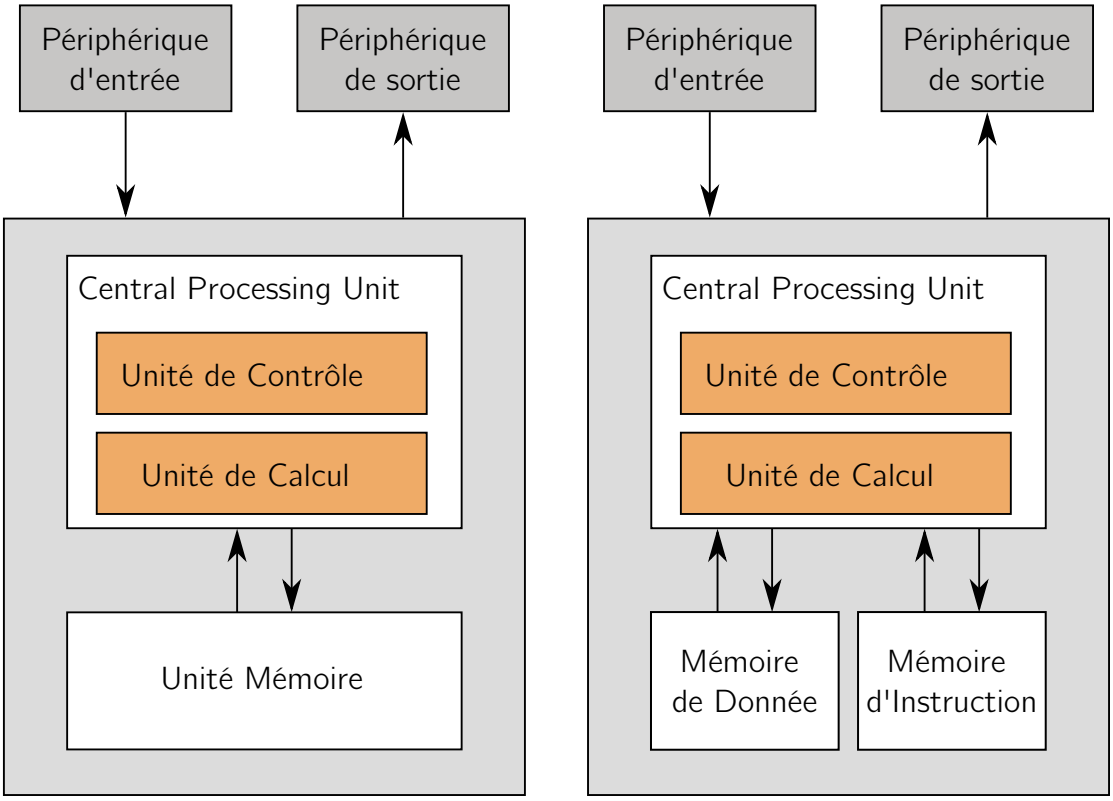
2.2.1 Bref historique des architectures des processeurs

Afin de bien comprendre l'intérêt de l'architecture RISC-V, il est utile de revenir sur l'historique de l'évolution des processeurs. Dans cette partie, il va tout d'abord être présenté un résumé de l'évolution de l'architecture des processeurs, puis une présentation de l'émergence de « l'open hardware ».

2.2.1.1 Historique de l'architecture des processeurs

À l'origine des architectures utilisées actuellement dans nos systèmes se trouvent deux architectures qui ont posé les bases des processeurs modernes. Tout d'abord, l'architecture de Harvard mise au point en 1939 a été introduite avec l'ordinateur Harvard Mark 1 conçu par Howard Aiken [84]. Cette architecture se caractérise par la séparation des mémoires d'instructions et de données, ainsi que par des chemins d'accès distincts, permettant un accès simultané aux données et aux instructions, comme illustré dans la Figure 2.12b. Cependant, le principal inconvénient de cette architecture est que la conception de machines l'utilisant est plus complexe et demande souvent plus d'espace que l'architecture de von Neumann, également émergente à cette époque.

En parallèle, l'architecture de John Von Neumann, comme montrée Fig 2.12a, propose une machine séquentielle avec une mémoire commune pour les instructions et les données [85]. Cette architecture a rapidement été préférée au modèle d'Harvard pour plusieurs raisons. Tout d'abord, elle permet de considérer les programmes comme des données,



(a) Architecture de Von Neumann (b) Architecture d'Harvard

FIGURE 2.12 : Schéma illustrant les deux architectures les plus couramment utilisées dans les processeurs

ce qui permet une reprogrammation dynamique de ces derniers sans modifier la machine. D'autre part, du fait de la mémoire partagée, l'architecture de Von Neumann est aussi moins coûteuse en termes de surface et de complexité de conception. Ces points ont fait que l'architecture de Von Neumann a pris le pas sur celle d'Harvard, bien que celle-ci puisse proposer de meilleures performances.

Par la suite, les premiers ordinateurs ENIAC [86] ont été fabriqués, mais c'est principalement à partir des années 1970 que les modèles de microprocesseurs actuels ont fait leur apparition. En 1971, la société Intel introduisit le premier microprocesseur 4 bits avec la puce Intel 4004 [87]. Quelques années plus tard, apparaît l'architecture x86, encore utilisée de nos jours avec la puce Intel 8086 [88]. Ces processeurs introduisent un jeu d'instructions permettant des calculs plus complexes et efficaces que les anciens modèles. il s'agit aussi d'un processeur à utilisation générale. Cela fait que cette série de processeurs d'Intel devient dominante dans le monde des ordinateurs personnels à partir de 1981. Cette architecture

est un processeur de modèle CISC du fait d'un grand nombre d'instructions différentes et spécifiques, CISC signifiant « Complex Instruction Set Computer » en anglais. Cependant, à partir des années 80, apparaît nouveau type d'architecture avec les architectures dites RISC pour « Reduced Instruction Set Computer » [89] qui vient de la réalisation du fait que pour 80% du temps seuls 20% des instructions étaient utilisées.

Ce faisant, les constructeurs ont développé de nouvelles puces suivant ce précepte avec notamment l'arrivée des puces ARM, pour « Acorn RISC Machine » du fait de l'entreprise d'origine qui est ensuite devenue « Advanced RISC Machine » lorsque l'architecture s'est démocratisée. L'intérêt de cette architecture est qu'elle se concentre sur un petit nombre d'instructions pour avoir une exécution plus rapide ainsi qu'une meilleure efficacité énergétique que son homologue x86. Cela en fait un choix intéressant pour tous les systèmes embarqués. Le tableau Tab 2.2 ci-dessous compare ces deux architectures.

Critère	**Type d'architecture**	**Consommation énergétique**	**Performance**	**Génération de chaleur**	**Place dans le marché**
ARM	CISC	Faible	Optimisé pour l'efficacité	Moins importante	Systèmes embarqués mobiles
x86	CISC	Élevée	Optimisé pour les performances	Plus importante	PC et serveurs

Table 2.2 : Comparaison des architectures ARM et x86

Comme le montre Tab 2.2, ces deux architectures sont complémentaires et ont des cas d'utilisation bien différents. Les processeurs de type x86 sont intéressants pour les ordinateurs personnels, et les processeurs ARM, qui, du fait de leur faible consommation énergétique, sont utilisés pour les systèmes embarqués, qui seront les modèles dominants pour les années à venir. Ces architectures sont à l'heure actuelle toujours dominantes avec un plus grand nombre de modèles, comme les modèles Ryzen de la marque AMD, qui sont des architectures x86, ou encore les modèles d'Apple M1, qui utilisent l'architecture ARM qui sont également utilisés dans les ordinateurs personnels. Cette évolution des architectures est rappelée Figure 2.13 afin d'avoir une vue graphique de cette temporalité.

Ces architectures posent cependant des limites dans le contexte technologique actuel, ce qui va faire naître un besoin d'ouverture et donc aider à la naissance de ce qu'on peut appeler l'« Open Hardware ».

2.2.1.2 L'émergence de l'open hardware

À partir de 1999, un groupe s'intéresse déjà au fait de se défaire des architectures propriétaires, OpenCores [90] fondé par Damjan Lampret [91]. Ce groupe cherche à contrer

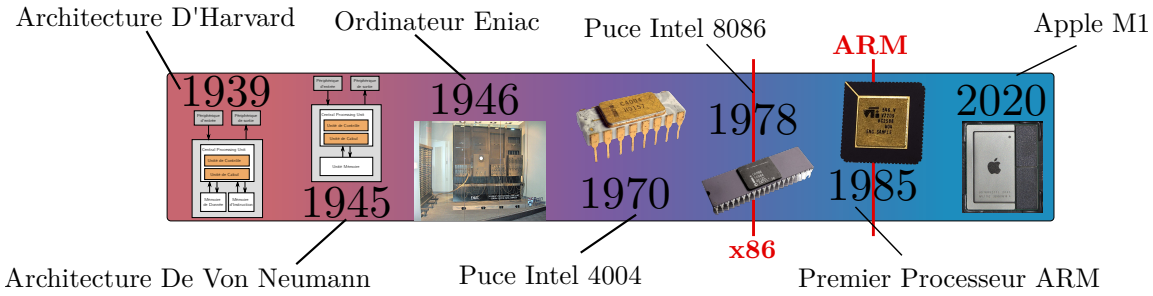


FIGURE 2.13 : Frise chronologique de l'évolution des architectures de processeur

les soucis liés aux architectures propriétaires qui ont émergé petit à petit avec l'évolution du monde numérique.

En effet, même si les architectures x86 et ARM maintiennent leur domination, leur nature propriétaire entre de plus en plus en conflit avec les besoins changeants du secteur en matière de flexibilité, d'efficacité et d'innovation. À mesure que l'industrie va « au-delà de la loi de Moore » [92], où l'efficacité énergétique et la spécialisation deviennent de plus en plus critiques, les architectures ouvertes offrent une voie d'avenir intéressante, permettant une plus grande personnalisation sans les restrictions imposées par les fournisseurs en place.

Un autre de ces soucis est que cette complexité gonfle les coûts de développement, où les processeurs x86 actuels prennent en charge environ 3600 instructions contre seulement 150 pour l'architecture ouverte RISC-V. Ce jeu d'instructions étendu implique un décodage plus complexe, ce qui augmente la latence et la consommation d'énergie. De plus, ce grand nombre d'instructions nécessite une surface de puce plus importante et une consommation accrue de silicium. En outre, les exigences en matière de rétrocompatibilité entraînent des inefficacités supplémentaires. La prise en charge d'un code vieux de plusieurs décennies fait que x86 conserve des instructions à longueur variable et des circuits redondants, augmentant ainsi la consommation électrique et la production de chaleur. Certaines fonctionnalités héritées, comme l'extension d'adresse physique (PAE) 32 bits, continuent de consommer des ressources qui seraient mieux utilisées pour les charges de travail plus modernes. De telles inefficacités sont parmi les raisons des limitations thermiques et de puissance des architectures propriétaires. Contrairement à l'architecture ouverte RISC-V et à ARM, la prise en charge héritée de x86 et l'architecture complexe CISC entraînent une consommation d'énergie plus élevée, ce qui rend x86 inadaptée aux appareils mobiles et IoT. En outre, le besoin de refroidissement actif dans les systèmes x86 limite encore leur utilisation dans des environnements à puissance limitée.

Ces limitations ont un impact direct sur les défis informatiques actuels. Les architectures propriétaires sont limitées dans l'atténuation du goulot d'étranglement de Von Neumann et le mur de mémoire. L'architecture x86 introduit la latence, et l'écosystème fermé d'ARM dissuade l'adoption généralisée des techniques de calcul en mémoire proche. L'efficacité énergétique demeure un défi permanent. Alors que l'ARM domine les appareils mobiles

avec son design RISC à faible consommation d'énergie, son modèle de licence restreint la personnalisation, ce qui limite son potentiel pour les charges de travail futures.

Cette inefficacité est également aggravée par une stagnation de l'innovation. Les ISA propriétaires donnent la priorité aux intérêts des fournisseurs plutôt qu'à l'adoption plus large de nouveaux paradigmes informatiques. L'adoption forcée d'ARMv8 par ARM, par exemple, a brisé la compatibilité héritée, tandis que les architectures propriétaires ont été lentes à intégrer des réponses comme le CiM pour combattre le mur de mémoire. Les ISA ouvertes comme le RISC-V permettent d'expérimenter et de s'adapter à un rythme beaucoup plus élevé, formant une innovation plus réactive en termes de technologie.

Outre la sophistication technique, l'enfermement des fournisseurs aggrave également le problème. Les architectures propriétaires se traduisent par des dépendances importantes, avec des migrations vers de nouvelles plateformes lourdes et coûteuses.

Les écosystèmes logiciels basés sur x86 et ARM nécessitent une émulation lors du passage à de nouvelles plateformes, ce qui entraîne des pertes de performance et de compatibilité. Les contraintes matérielles jouent également un rôle, car les fournisseurs contrôlent les mises à jour des fonctionnalités et les coûts associés. Par exemple, le projet Itanium d'Intel a entraîné des pertes financières monumentales dans l'industrie, montrant comment la décision d'un seul fournisseur peut affecter l'ensemble de l'industrie.

La concurrence et l'innovation sont également limitées par des obstacles juridiques. Les brevets déposés par Intel et ARM font qu'il est difficile pour des tiers de mettre au point des solutions de remplacement sans devoir payer des frais de licence très élevés. Une telle exclusivité profite aux grandes entreprises qui ont les ressources nécessaires pour acheter des licences de base personnalisées, mais désavantage les petits acteurs car ces architectures nécessitent des années de développement et des dizaines de millions de dollars d'investissement avant d'atteindre le marché. Même ceux qui obtiennent des licences ne sont pas entièrement protégés de l'exposition juridique, comme on le voit dans des cas tels que Nvidia vs. Qualcomm / Samsung sur les brevets liés au GPU.

La fragmentation de l'écosystème est un autre problème grave. Les ISA propriétaires tels que x86, ARM et MIPS (pour Microprocessor without Interlocked Pipelined Stages) [93], dont l'entreprise fondatrice est passée à RISC-V depuis 2021 [94], ont contribué à la création d'un écosystème logiciel fragmenté, où les changements architecturaux rompent souvent la rétrocompatibilité. Par exemple, le passage d'AArch32 à AArch64 dans ARM a nécessité d'importantes modifications logicielles et les développeurs ont dû faire face à des défis pour maintenir la compatibilité multiplateforme. De plus, les architectures propriétaires limitent la personnalisation et les utilisateurs ne peuvent pas optimiser les processeurs pour des tâches spécialisées telles que l'IA et l'informatique en périphérie.

Pour résoudre ces problèmes, des architectures ouvertes comme RISC-V sont apparues comme une alternative viable. En éliminant les restrictions liées aux brevets et aux licences, le jeu d'instruction RISC-V permet l'innovation et l'accessibilité. En tant qu'architecture ouverte, RISC-V permet l'intégration d'accélérateurs spécifiques à un domaine, par exemple pour des conceptions orientées IA, sans restrictions de la part des ISA pro-

priétaires. L'écosystème collaboratif RISC-V simplifie également la vérification et accélère l'adoption de nouvelles technologies.

2.2.2 Présentation de RISC-V

2.2.2.1 Origines et philosophie

RISC-V (à prononcer RISC five) est un jeu d'instructions (ISA) ouvert sur lequel des étudiants de l'université de Berkeley aux États-Unis ont commencé à travailler à partir de 2010 [95]. Le « V » dans le nom correspond à la cinquième génération de processeurs RISC développés à Berkeley. Cette initiative a été lancée par Krste Asanovic, Yunsup Lee et Andrew Waterman qui avaient pour objectif de s'attaquer aux limitations des architectures propriétaires dominantes et de leurs jeux d'instructions, x86 et ARM par exemple. Comme expliqué précédemment, ces architectures sont fréquemment inaccessibles aux académiciens du fait des prix des licences de ces architectures et des problèmes évoqués plus haut. Ils se sont donc inspirés de ce qui existait déjà au niveau du développement logiciel et avaient pour objectif d'offrir avec RISC-V le même niveau d'ouverture et de flexibilité dans le monde du développement électronique.

La première version du manuel de RISC-V est publiée en 2011, avec un jeu d'instructions minimaliste mais extensible pour permettre une utilisation académique et une utilisation à la fois académique et industrielle. À partir de 2015, la fondation RISC-V est créée avec pour objectif de promouvoir et de favoriser l'adoption à grande échelle du jeu d'instructions RISC-V. L'organisation s'est fortement développée et compte maintenant plus de 200 entreprises membres contribuant à l'écosystème RISC-V.

La philosophie du jeu d'instructions est de promouvoir l'ouverture, la modularité et l'efficacité. Pour cela RISC-V peut s'appuyer sur des caractéristiques clés, qui constituent son principal attrait pour les concepteurs et utilisateurs de cette architecture. Ces dernières sont les suivantes :

- **Une architecture ouverte :** RISC-V a été créé avec comme premier objectif de faire un jeu d'instruction ouvert ne nécessitant pas de payer des frais afin de pouvoir développer sur le jeu d'instruction ou encore d'utiliser les extensions qui ont été validées par la fondation RISC-V
- **Une ISA simple et efficace :** L'objectif de RISC-V est d'être un jeu d'instruction simple et efficace afin de favoriser la rapidité d'implémentation et la portabilité. À titre de comparaison, dans le jeu d'instruction de base de RISC-V, le RV32I comprend 52 instructions distinctes [96]. À l'inverse, le jeu d'instructions de base A32 de ARM comporte au minimum 151 instructions [97].
- **Une grande modularité :** La vraie grande force de RISC-V est sa modularité, les jeux de bases (RV32I ou RV64I) sont extensibles via des modules optionnels qui

permettent de développer simplement des architectures spécialisées dans certaines applications [98] comme par exemple les calculs vectoriels, la gestion des flottants ou encore même des calculs pour la cybersécurité avec l'implémentation des mathématiques pour les champs finis.

- **Un écosystème en pleine expansion :** plus de dix ans après son introduction, la communauté RISC-V continue de s'étendre avec de nouveaux acteurs qui s'intéressent à ce jeu ouvert. Il y a déjà l'existence notable de projets impactants comme par exemple Rocket Chip par Chipsalliance qui permet de développer un système sur puce via leur application [99]. Il est aussi estimé que pour 2025, il y aurait 60 milliards de coeurs RISC-V [98]

2.2.2.2 Intérêt de RISC-V pour les nouvelles technologies de mémoire et le LiM

Les caractéristiques présentées précédemment permettent de nombreux cas d'utilisation de RISC-V dans les technologies actuelles. Un processeur basé sur des memristors ainsi que de la LiM a réussi à atteindre une amélioration de 70% en termes de performance et de consommation d'énergie dans le cas d'utilisation d'un algorithme de hachage lors de communications via des outils IoT [100]. L'intérêt de RISC-V dans ce contexte de ressources limitées est le fait que le jeu d'instructions de base soit simple et peu gourmand en termes de consommation, mais que malgré tout, il puisse aussi s'adapter à un nouveau type de calcul, dans ce cas, l'algorithme de hachage.

Dans un autre exemple, la possibilité d'étendre le jeu d'instructions de RISC-V a permis d'intégrer de la LiM au sein d'un processeur [101]. Cet ajout a permis d'obtenir une réduction de la consommation d'énergie d'au moins 43% lors d'un benchmark avec des technologies CMOS classiques, mais aussi de rendre le code plus parallèle et de réduire le nombre d'opérations mémoire.

Il apparait que RISC-V est très intéressant afin d'utiliser sa possibilité d'extension du jeu d'instructions de base tout en restant un moyen de construire des processeurs peu énergivores. Cependant, ce n'est pas tout, car RISC-V peut aussi servir dans le cas de processeurs cherchant la haute performance, bien qu'il reste encore en retrait par rapport à des architectures établies comme x86. À l'heure actuelle, RISC-V se dote d'extensions permettant la conception d'architectures de type GPU [102]. Il existe également de nombreuses autres extensions, notamment dédiées à la cryptographie post-quantique [103], ainsi que d'autres domaines détaillés dans la littérature [98].

L'introduction des architectures ouvertes comme RISC-V facilite l'exploration de nouvelles approches d'intégration, telles que celles requises par les dispositifs Logic-in-Memory. Toutefois, l'évaluation rigoureuse de ces systèmes, à la fois sur le plan fonctionnel et en termes de performances, impose la mise en œuvre de méthodologies de test adaptées. La section suivante a pour objectif de présenter les différentes stratégies de validation des

systèmes et de nouveaux opérateurs, en mettant en lumière les apports spécifiques de la simulation, de l'émulation et du prototypage.

2.3 Méthodes d'évaluation des technologies de mémoires non-volatiles.

L'évaluation des technologies de mémoires non-volatiles constitue une étape essentielle pour valider leur faisabilité et leurs performances avant une intégration à grande échelle. Face aux nombreux défis posés par ces technologies émergentes, tels que la variabilité des dispositifs, la fiabilité à long terme ou encore leur adaptation aux systèmes complexes, plusieurs approches complémentaires ont été développées. Ces approches, allant des tests physiques sur prototypes aux simulations logicielles, en passant par l'émulation matérielle, permettent d'obtenir une vision globale et détaillée des caractéristiques de ces mémoires dans divers contextes. Cette section explore ces méthodes en détaillant leurs principes, leurs avantages et leurs limitations, avant de démontrer pourquoi l'émulation se distingue comme une approche clé.

2.3.1 Approches de tests et d'évaluations

Parmi toutes les méthodes pour tester les nouveaux opérateurs, il va ici être présenté trois méthodes qui sont parmi les plus courantes pour l'évaluation. Dans l'ordre, vont être présentés le prototypage qui consiste, comme son nom l'indique, à réaliser un prototype des nouveaux opérateurs à l'aide par exemple d'un circuit spécifique à l'application voulue (Application Specific Integrated Circuit, ASIC), puis la simulation qui utilise des logiciels pour évaluer des représentations des circuits et enfin l'émulation qui se sert d'une plateforme matérielle pour imiter le comportement réel des circuits ciblés.

2.3.1.1 Test Hardware : prototypage

Cette approche implique de concevoir physiquement les puces liées à notre nouvelle fonction, puis d'évaluer leurs performances réelles, leur fiabilité et leur compatibilité avec des environnements plus grands [104]. Ce type de test permet d'avoir la plus grande fidélité dans les résultats, car il est possible d'observer à la fois le fonctionnement du circuit, mais aussi les variations lors de la fabrication des prototypes, les facteurs environnementaux et le stress opérationnel. Habituellement, cette approche est utilisée comme méthode d'évaluation finale permettant de faire un benchmarking avant de pouvoir envoyer les nouveaux circuits vers une production de masse. La raison pour laquelle cette méthode est utilisée en dernier est notamment due aux coûts de fabrication des puces prototypes ainsi qu'au temps nécessaire à leur élaboration. En effet, chaque nouvelle itération de la puce demande de créer de nouveaux prototypes. Un autre point négatif de cette méthode est

le fait que l'observation du fonctionnement du système ainsi que le débogage sont plus complexes du fait de la faible flexibilité du circuit et du coût éventuel des outils de test.

2.3.1.2 Simulation du circuit

La simulation est une approche orientée logicielle qui permet de modéliser le comportement des cellules d'opérateurs à tester à divers niveaux d'abstraction, comme la cellule elle-même, le circuit l'intégrant ou encore tout un système comme un processeur. Le concept est alors d'utiliser les modèles physiques ou comportementaux des opérateurs désirés. Cette méthode est généralement privilégiée dans les premières étapes de conception d'un nouvel opérateur car elle permet une itération rapide des paramètres du circuit sans avoir besoin de fabriquer de nouveaux prototypes. À l'aide de la simulation, il est possible de capturer en détail le fonctionnement des circuits et d'obtenir des informations concernant l'endurance, la durée de rétention dans le cas des mémoires, ou encore les défaillances de fonctionnement de l'opérateur. La simulation est cependant limitée par la vitesse à laquelle elle peut fonctionner [25] où, pour simuler une seconde de fonctionnement d'un processeur mono-cœur, cela peut prendre jusqu'à 8 heures. Une autre limitation est la précision des résultats obtenus qui dépend de la définition du modèle utilisé. Ces modèles peuvent ne pas prendre en compte la totalité de l'impact du monde réel et, par exemple, les défauts de fabrication.

2.3.1.3 Émulation matérielle sur FPGA ou plateformes spécifiques

L'émulation est une technique qui se situe à l'interface des deux précédentes. Le principe est d'utiliser une plateforme physique reconfigurable, comme un FPGA [105], dans le but d'imiter le fonctionnement ainsi que de reproduire les timings des cellules testées à un niveau proche du matériel. L'avantage principal de l'émulation est sa capacité à exécuter des processus rapidement malgré leur complexité comparée à la simulation. Cela permet donc de tester de nombreuses configurations pour le circuit émulé, malgré la complexité apparente de l'application considérée, tout en restant dans un environnement contrôlé et flexible. L'émulation, dans le cadre des systèmes mémoires par exemple, permet de tester les interactions entre les NVM et les autres composants du système avant de passer à la fabrication réelle. La reconfiguration des FPGAs permet des mises à jour rapides du système à évaluer alors que le prototypage implique de refaire fabriquer de nouveaux circuits.

Un avantage des FPGA pour le test des mémoires non-volatiles est la possibilité d'émuler une coupure d'énergie [106]. Cette capacité est particulièrement intéressant dans le contexte actuel de l'IoT. Le point faible de cette méthode est qu'il est nécessaire de configurer un système complet,même lorsqu'il s'agit de tester un opérateur élémentaire. De plus, il faut prévoir un modèle adapté afin de ne pas avoir de conflit d'espace interne au FPGA en termes d'espace mémoire, ce qui peut réduire la précision des résultats obtenus.

2.3.2 Comparaison des méthodes et explication du choix pour la thèse

Technique	Simulation	Emulation	Prototypage
Vitesse	Lente	Rapide	Circuit réel
Flexibilité	Elevée	Elevée	Basse
Fidélité	Dépendant du modèle	Comportementale ou structurelle	Exacte
Coût	Faible	Moyen	Elevée
Débogage	Visiblité complète du système	Monitoring, sondes, Outil FPGA	Limité par les Sondes

TABLE 2.3 : Table de comparaison des points avantages et inconvénients de chaque méthode

Le tableau Tab 2.3 permet de comparer chacune des méthodes et d'expliquer le choix méthodologique retenu dans cette thèse. Tout d'abord, une première comparaison peut être établie du point de vue des performances. La simulation, selon le niveau d'abstraction souhaité, est facilement la plus lente des méthodes avec la difficulté à simuler des architectures complexes comme des processeurs complets à un niveau RTL. Cependant, cette lenteur est bien compensée par le fait qu'il s'agit de la méthode offrant la plus grande flexibilité pour les tests, avec la possibilité de modifier absolument tous les paramètres imaginables. Le recours exclusif à des outils logiciels rend aussi le débogage plus simple qu'avec les autres méthodes comme l'émulation, qui nécessite malgré tout un contrôle du FPGA. Cependant, de son côté, l'émulation possède une vitesse proche de la vitesse réelle du circuit grâce à l'adaptation du modèle matériel visant à reproduire son comportement. En revanche, les résultats obtenus seront uniquement comportementaux, du fait de la possible grande abstraction du modèle utilisé. Le prototypage est le plus rapide avec une vitesse réelle ; en revanche, il s'agit de l'approche la moins flexible et la plus difficile à tester. En termes de coût, le prototypage constitue l'approche la plus onéreuse, suivi de l'utilisation d'un FPGA, tandis que la simulation, reposant uniquement sur des outils logiciels, demeure la méthode la plus économique. D'un point de vue général, chaque méthode a son intérêt et elles sont complémentaires. La simulation est souvent utilisée pour obtenir les premiers résultats afin de savoir s'il est intéressant de poursuivre la recherche, tandis que le prototypage permet de consolider et de valider l'intégration finale du système.

L'émulation s'impose progressivement comme une méthode intermédiaire afin d'obtenir des informations dans des systèmes plus complexes que le circuit seul. C'est dans ce contexte que l'émulation a été retenue comme méthodologie principale pour cette thèse. Des résultats préalables ont déjà été obtenus avec les circuits et l'émulation permet ainsi une évaluation rapide dès les premières étapes de conception des opérateurs fondés sur des technologies émergentes. Elle reste en effet plus économique et moins chronophage que la fabrication de démonstrateurs matériels destinés à vérifier la fonctionnalité.

2.4 Conclusions

La solution prometteuse du CiM afin de pallier le « Memory Wall » rejoint la littérature présentée précédemment et s'impose comme une brique importante du développement de la thèse. Son développement est particulièrement pertinent au regard des multiples bénéfices proposés, notamment une augmentation des performances en termes de vitesse de fonctionnement ainsi qu'une réduction significative de la consommation énergétique, comme illustré par la Figure 2.3. Le projet SECRET s'inscrit dans ce contexte d'utilisation du paradigme CiM afin de développer de nouveaux outils pour la cryptographie.

Ces nouveaux opérateurs basés sur les mémoires FeFET doivent être testés à la suite de simulations donnant des résultats intéressants. Dans cette optique, la suite du manuscrit présente une plateforme d'émulation reposant sur un cœur de processeur RISC-V, comprenant de nouvelles instructions permettant d'effectuer l'émulation des opérateurs nécessaires.

Chapitre 3

L'émulation d'opérateurs non-volatiles

3.1 L'élaboration du modèle pour les opérateurs NVM 42
3.1.1 Le modèle réaliste 42
3.1.2 Le modèle comportemental 45
3.2 Le choix du coeur de la plateforme 46
3.2.1 Quel type d'unité de calcul pour la plateforme ? 46
3.2.2 Le choix du processeur 48
3.2.3 COMET RISC-V 51
3.3 La plateforme d'émulation complète 53
3.4 Conclusions 55

L'essor des technologies NVM, en particulier celles basées sur des dispositifs émergents comme les FeFETs, nécessite la mise en place de méthodologies d'évaluation adaptées à leurs particularités. En effet, leur intégration au sein d'architectures de calcul, notamment dans le cadre du Logic-in-Memory, implique des défis d'analyse complexes, tant au niveau technologique que fonctionnel.Pour répondre à ces enjeux, l'émulation se présente comme une approche efficace permettant d'analyser le comportement de circuits de calcul reposant sur des dispositifs émergents, désignés par la suite comme des opérateurs,dans un environnement réaliste, sans supporter les coûts associés à la fabrication de prototypes matériels, ni les limitations de scalabilité rencontrées avec les simulations traditionnelles.Ce chapitre propose ainsi de détailler le processus de mise en place d'une plateforme d'émulation dédiée aux opérateurs non-volatiles, visant à permettre leur intégration, leur configuration et leur analyse dans le contexte d'un cœur processeur RISC-V.

Les sections de ce chapitre s'articulent autour des étapes clés suivantes, avant de conclure par une synthèse des choix techniques opérés, de la structure finale obtenue, et des perspectives pour les expérimentations à venir :

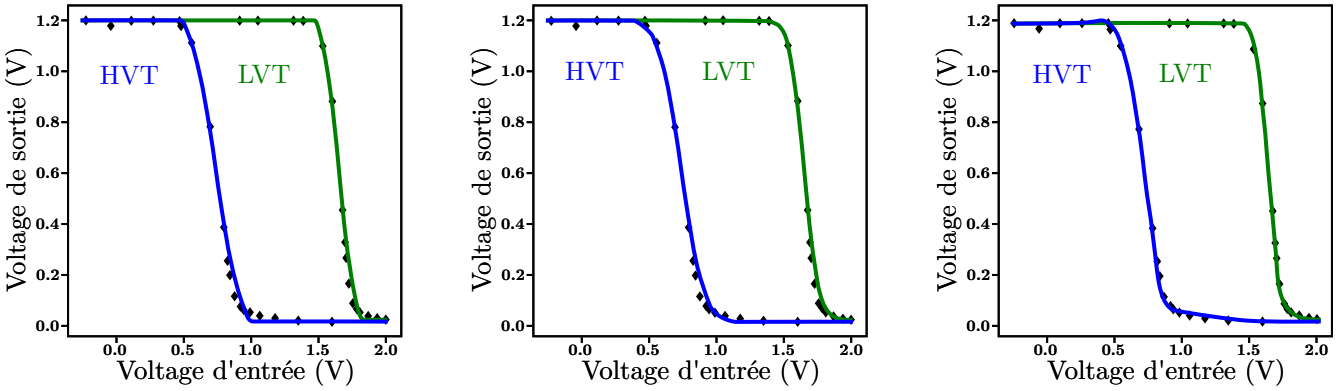
- **Choix pour l'élaboration de la plateforme d'émulation**
  Présentation des spécificités de l'émulation appliquée aux technologies émergentes, ainsi que de la sélection du cœur de processeur cible en fonction des contraintes d'intégration et de flexibilité.
- **Implémentation des opérateurs non-volatiles**
  Description de la forme logique et fonctionnelle retenue pour les opérateurs, puis intégration pratique au sein du cœur processeur via des extensions matérielles ou des instructions spécifiques.
- **Construction de la plateforme complète d'émulation**
  Assemblage des différents blocs nécessaires au bon fonctionnement de la plateforme, intégration des outils de test, et préparation à l'évaluation.

3.1 L'élaboration du modèle pour les opérateurs NVM

Dans cette section, sont présentées les réflexions et les choix effectués dans le cadre de l'émulation des opérateurs. Dans un premier temps, la piste de l'approche dite réaliste est introduite avec l'exemple de l'émulation des FeFET, puis le choix de la carte de développement retenue est détaillé au regard des facteurs clés identifiés.

3.1.1 Le modèle réaliste

Le projet SECRET, tel qu'expliqué précédemment, vise à concevoir de nouveaux opérateurs exploitant des technologies émergentes non-volatiles. Au cours du projet, l'approche adoptée consistait à utiliser les caractéristiques d'hystérésis des transistors



(a) Arctangente
LVT : $E_R = 4.7\%$
HVT : $E_R = 4.2\%$

(b) Tangente hyperbolique
LVT : $E_R = 4.5\%$
HVT : $E_R = 4.3\%$

(c) Spline cubique
LVT : $E_R = 2.1\%$
HVT : $E_R = 3.2\%$

FIGURE 3.1 : Comparaison des hystérésis entre modèle approché et modèle réel

ferroélectriques, obtenues à partir de tests sur des transistors ferroélectriques de 28 nm réalisés par Namlab en Allemagne. Ces tests ont fourni des informations cruciales sur la consommation de puissance et la vitesse de ces transistors. L'idée sous-jacente était de développer un système centré sur un processeur intégrant une section dédiée à l'émulation transistors ferroélectriques, permettant ainsi d'observer leur comportement dans le contexte global d'un processeur.

Cette approche présente plusieurs avantages. Tout d'abord, comme rappelé dans le chapitre précédent, elle offre un gain de temps significatif par rapport à la simulation complète d'un processeur, qui peut prendre jusqu'à 8 heures pour simuler une seconde de fonctionnement à une vitesse de 200 000 instructions par seconde [25]. De plus, elle élimine la nécessité de développer des cartes d'essai physiques, réduisant le temps total de conception ainsi que les besoins en matériaux. En concentrant l'émulation sur la dynamique des transistors ferroélectriques, cette méthode permet une analyse plus ciblée et efficace des performances, contribuant ainsi à accélérer le processus de développement du projet SECRET.

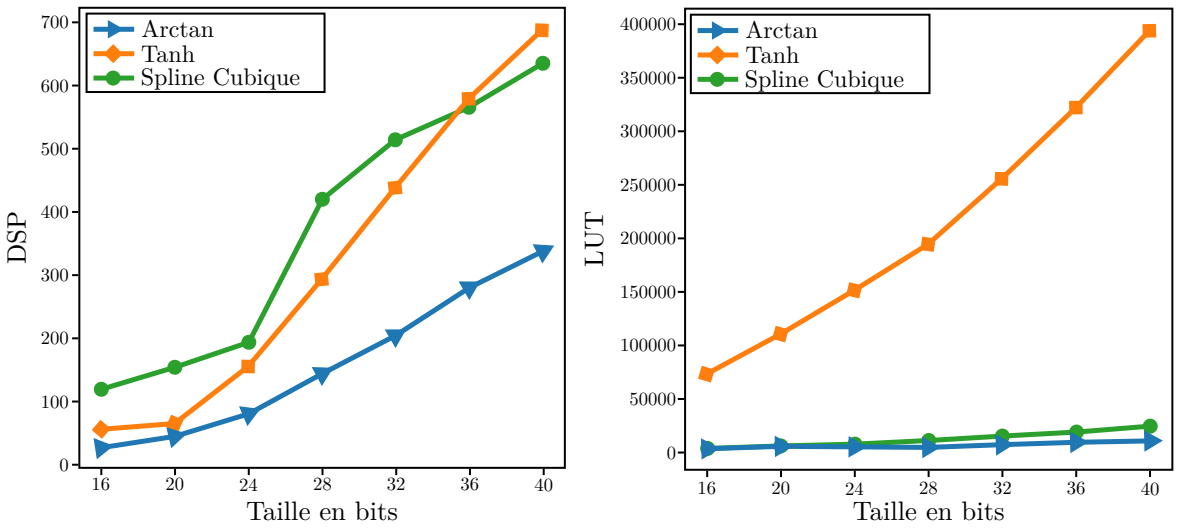
Cette technique d'émulation permet d'avoir une vision précise du fonctionnement des transistors étudiés afin de pouvoir obtenir les données les plus réalistes lors des tests de ces derniers en termes de vitesse d'exécution et de consommation. Ce type d'émulation a été réalisé au sein de l'équipe du projet SECRET par un autre doctorant [107] et il obtient les résultats montrés Figure 3.1 sur le degré de précision par rapport à l'hystérésis du fonctionnement des transistors.

On observe sur la Figure 3.1 que la reproduction des équations à l'aide des techniques d'approximations classiques des équations donne des résultats très proches des courbes issues de mesures expérimentales, avec des erreurs comprises entre 3% et 4% pour les

courbes des hautes tensions (HVT) et de 4% et 2% pour les courbes des basses tensions (LVT). Ces résultats montrent que le modèle approché permet d'obtenir des valeurs très proches du comportement réel des transistors testés.Cependant, l'émulation des transistors à l'aide de cette approche se révèle particulièrement gourmande en ressources matérielles sur la carte FPGA utilisée dans le cadre de ces travaux, dont les capacités sont récapitulées dans le Tableau 3.1.

Cellules logiques	Blocs logiques reconfigurables	LUT	Blocs DSP	Blocs RAM(en KB)
693 120	108 300	433 200	3600	52 920

TABLE 3.1 : Éléments présents dans le coeur FPGA Virtex7 XC7VX690T



(a) Utilisation en DSP

(b) Utilisation en LUT

FIGURE 3.2 : Utilisation des ressources sur le FPGA (DSP ou LUT) en fonction de la méthode choisie

La Figure 3.2 présente la consommation des ressources du FPGA lors de l'émulation d'une unique courbe d'hystérésis. Il apparaît que l'exploitation des ressources du cœur Virtex-7 est particulièrement élevée. À l'exception de la méthode basée sur la tangente hyperbolique, l'occupation des LUT n'apparaît pas comme le facteur limitant dans ce contexte, comme le démontre la Figure 3.2b lorsque les résultats sont comparés aux ressources disponibles sur le coeur Virtex-7. En revanche, la génération d'une seule courbe requiert environ 50 unités de traitement du signal numérique (DSP) sur la carte FPGA, conformément aux résultats illustrés à la Figure 3.2a.

La contrainte principale provient du fait qu'un transistor complet exige l'émulation de quatre courbes distinctes : les courbes de tension et d'intensité associées à l'hystérésis

montante, ainsi que ces mêmes courbes pour l'hystérésis descendante. Cela conduit à une mobilisation d'environ 200 unités de DSP par transistor, limitant ainsi l'implémentation à un maximum de 18 transistors ferroélectriques, sans sans même considérer la consommation des ressources dédiées au processeur et aux autres éléments de la plateforme d'émulation. Cette approche ne permet donc pas d'évaluer de manière exhaustive l'impact des transistors ferroélectriques intégrés au sein du processeur, notamment dans le cas d'opérateurs regroupant typiquement bien plus d'un ou deux éléments de ce type.

Afin de poursuivre cette démarche d'évaluation à l'échelle du système, il devient nécessaire d'identifier une version simplifiée du modèle. Une telle simplification impliquerait inévitablement une réduction de la précision des estimations obtenues, mais constituerait un compromis nécessaire afin de rendre l'exercice d'émulation réaliste à l'échelle du système envisagé.

3.1.2 Le modèle comportemental

Pour répondre aux contraintes d'utilisation des ressources liées à l'émulation précise d'un transistor ferroélectrique, il a été choisi de développer un modèle abstrait, sous la forme d'un ensemble de caractéristiques, désigné par la suite comme une carte modèle afin de bien imiter le comportement des FeFET, sans avoir à représenter explicitement les courbes caractéristiques. Les paramètres jugés pertinents pour la définition de ce modèle sont les suivants :

- La consommation énergétique des transistors pour chacune des opération, , à savoir la lecture et l'écriture, en fonction de la valeur manipulée (0 ou 1), afin de permettre l'estimation de la consommation associée aux calculs,
- Les délais nécessaires à la lectures d'une donnée lors d'un passage au travers des transistors,
- les délais nécessaires à l'écriture d'une donnée lors d'un passage au travers des transistors

L'utilisation de ces paramètres permet de concevoir un code de test destiné à l'évaluation des opérateurs au sein de nos fonctions. L'utilisation de ces paramètres permet de concevoir un code de test destiné à l'évaluation des opérateurs au sein de nos fonctions.

Par exemple dans le cas de la lecture pour un opérateur basé sur des FeFETs, il faudra adapter le temps d'exécution associé à l'utilisation de la porte logique constituée du transistor ferroélectrique à la fréquence de fonctionnement de l'unité de calcul. Dans le cas d'une porte NAND à l'aide d'un unique transistor ferroélectrique [5], dans le pire cas, à savoir lorsque la valeur stockée dans le transistor est 1 et qu'on vient mettre l'entrée à 1 aussi comme indiqué dans la table, le temps de lecture d'une donnée prend 260,3 ns. Si le processeur possède une vitesse de 100 MHz soit l'équivalent d'un cycle toutes les 10 ns, ce délai équivaut à devoir effectuer une lecture de la donnée contenue dans notre porte NAND

non-volatile en prenant un total de 27 cycles processeurs. Bien que la valeur soit proche de 260 ns, il est nécessaire de prendre en compte le temps complet du passage à travers la porte et donc ajouter un cycle processeur aux 26 cycles si on arrondissait.

Concernant la consommation, il faut prendre en compte le passage dans chaque porte en fonction de la table de vérité de cette dernière. Étant donné que les différentes portes logiques utilisées dans le système sont connues, la plateforme permet de suivre le nombre de traversées de chaque porte, ainsi que le résultat logique associé à chacune d'elles. Par exemple, dans le cas de la porte NAND, le système surveille le nombre de passage par la porte logique tout en vérifiant dans quel état est la porte logique quand on y passe. Pour ce faire, un système de compteurs est utilisé. Une fois le total de passages dans la porte effectué par le programme obtenu, les valeurs des compteurs sont renvoyées sur le PC pour effectuer le calcul de consommation totale. Il faut faire de même pour l'écriture en tenant compte cette fois-ci du passage d'une valeur à l'autre. Combien de fois passe-t-on de 0 à 1, de 1 à 0 ou bien même de 0 à 0 ainsi que de 1 à 1.

Maintenant que la méthodologie d'implémentation des opérateurs est définie, en implémentant un délai et en préparant des compteurs, il faut choisir une plateforme adaptée à cette utilisation. Pour cela, il faut choisir un système simple et peu gourmand afin de coller au contexte de l'IoT, et modifiable en profondeur pour pouvoir insérer nos opérateurs. Le choix du langage de description ou de programmation n'est pas un critère déterminant, mais la disponibilité d'un langage simple et facilement exploitable constitue néanmoins un atout. Dans cette optique, la section suivante est consacrée à l'identification du composant central du système qui sera utilisé comme base de la plateforme d'émulation.

3.2 Le choix du coeur de la plateforme

3.2.1 Quel type d'unité de calcul pour la plateforme ?

Pour répondre à l'orientation du projet SECRET vers l'IoT, il est impératif de sélectionner un processeur adapté à nos exigences. Cela a nécessité une analyse approfondie pour identifier l'unité de calcul la plus appropriée. Deux options principales se présentaient dans notre cas : l'utilisation d'un microcontrôleur ou le choix d'un processeur généraliste.

- Pour le cas du microcontrôleur, il s'agit en général d'une architecture peu complexe qui possède à la fois un coeur de calcul ainsi que tout un ensemble de périphériques dont une mémoire embarquée. Cela autorise une utilisation quasi immédiate du système, la seule contrainte résidant dans la prise en compte, lors de l'écriture du code, des différents registres nécessaires au bon fonctionnement de l'ensemble des périphériques. Dans le cadre de l'IoT certains peuvent être optimisés afin d'offrir une utilisation ultra basse consommation. Le principal inconvénient des architectures de type microcontrôleur réside toutefois dans leurs capacités de calcul limitées. En effet, Le fait d'avoir un coeur de calcul simplifié et une mémoire de taille limitée ne permet

pas un grand nombre de calculs différents. De plus, la programmation du microprocesseur imposerait un code bien précis ne pouvant répondre qu'à une application spécifique.

- Dans le cas des processeurs, ces derniers ont l'avantage d'être génériques et permettent un nombre plus important de calculs. Contrairement aux microcontrôleurs, les processeurs ne disposent généralement pas de périphériques ni de mémoire embarquée, en dehors d'éventuelles mémoires internes telles que les registres ou la mémoire cache. Il est donc nécessaire de concevoir un environnement matériel complet autour du processeur choisi, tout en veillant à respecter les contraintes propres au contexte de l'IoT.

Que ce soit dans le cas d'un microcontrôleur ou d'un processeur, l'accès à l'architecture interne de l'unité de calcul reste en général limité, ces architectures étant bien souvent la propriété d'entreprises. Toutefois, le processeur offre une flexibilité accrue, même si cela implique d'implémenter également les blocs périphériques nécessaires à son fonctionnement. Ce choix est d'autant plus pertinent que la modification de l'architecture du cœur de calcul constitue une transformation lourde, plus aisée à réaliser avec un processeur, notamment grâce aux possibilités offertes par son implantation sur un FPGA.

Dans le cas d'un microcontrôleur, les mécanismes d'interruptions permettent effectivement d'exploiter un certain parallélisme entre l'unité de calcul et les blocs matériels associés. Cela constitue un avantage notable pour de nombreuses applications embarquées. Cependant, dans notre contexte, l'intégration d'opérateurs non-volatiles nécessite une modification directe du flot d'exécution et de l'architecture du cœur, ce qui ne peut pas être géré efficacement uniquement par des interruptions. Ces dernières restent adaptées pour orchestrer des événements matériels, Les interruptions restent adaptées à l'orchestration d'événements matériels, mais elles ne permettent pas de redéfinir le pipeline d'exécution ni le jeu d'instructions, éléments centraux de notre étude.

Ainsi, le choix d'un processeur, pour lequel il est possible d'implémenter directement de nouvelles instructions au sein du pipeline, s'est imposé comme la solution la plus adaptée à nos objectifs.

L'étape suivante a consisté en la définition du type de processeur le plus adapté aux besoins du projet. L'idée initiale reposait sur la conception d'un processeur entièrement dédié, fondé sur un jeu d'instructions minimaliste que nous aurions pu enrichir progressivement. Une telle approche aurait offert un contrôle total sur la définition et l'évolution du jeu d'instructions, ainsi qu'une connaissance exhaustive de l'architecture matérielle sous-jacente. Toutefois, cette piste a rapidement été écartée. En effet, le développement complet d'un processeur représente un investissement considérable en temps et en ressources, notamment pour assurer la validation fonctionnelle de l'architecture, la mise en place d'un flot de compilation adapté et la conception d'un jeu d'instructions à la fois cohérent, flexible et évolutif. Dans le cadre temporel restreint du projet SECRET, un tel investissement

s'avérait difficilement compatible avec les objectifs de recherche visés. Par conséquent, afin de disposer d'une base stable et éprouvée, il a été décidé de s'appuyer sur un jeu d'instructions existant, offrant à la fois un écosystème logiciel déjà établi et la possibilité d'y intégrer de nouvelles instructions de manière plus pragmatique.

C'est pourquoi le jeu d'instructions RISC-V, tel que présenté dans le chapitre précédent, constitue un choix particulièrement pertinent pour l'avancement de notre projet. En effet, en tant que jeu d'instructions ouvert, et donc libre d'accès à l'utilisation pour tout le monde, cela permet de développer ses propres architectures de calcul sans avoir à se soucier de son appartenance à une entreprise. De plus, RISC-V a gagné une popularité croissante à l'échelle internationale [108]. Il bénéficie d'une vaste communauté regroupant à la fois des chercheurs et des acteurs industriels [109], garantissant l'existence d'un large éventail de processeurs susceptibles de répondre aux besoins de notre projet. Par ailleurs, le jeu d'instructions RISC-V repose sur des principes simples et offre une grande flexibilité pour l'ajout d'instructions supplémentaires. Cette caractéristique confirme donc son adéquation à nos besoins. En choisissant le jeu d'instructions RISC-V, le projet SECRET s'intègre à une communauté dynamique dont le nombre de membres est en constante augmentation. Cela nous permet d'échanger simplement avec les membres de cette dernière pour mieux se placer et pouvoir avoir directement des informations par rapport aux processeurs développés. De plus, cela permettra aussi à la communauté de réutiliser librement notre travail.

3.2.2 Le choix du processeur

Après avoir décidé de retenir un processeur déjà existant ainsi qu'un processeur basé sur le jeu d'instructions RISC-V, il convient désormais d'identifier le processeur avec lequel les travaux seront menés. La principale difficulté réside dans le fait que, parmi tous les processeurs proposés librement, la majorité dispose d'une documentation très détaillée concernant leur utilisation et la mise en place de l'environnement nécessaire à leur fonctionnement.En revanche, en ce qui concerne la modification des cœurs et la description de l'organisation des fichiers, la documentation se révèle généralement insuffisante, les explications étant souvent très succinctes et se limitant à un simple schéma décrivant l'architecture globale. Cette situation rend particulièrement complexe l'appropriation de l'architecture interne du processeur, en raison du manque de commentaires dans la documentation décrivant le cœur, ainsi que de l'absence fréquente d'un code exhaustivement commenté expliquant le rôle et le fonctionnement des différents fichiers sources.

Il faut donc choisir un coeur de processeur qui répond bien à ces exigences ainsi qu'à celles données par le projet SECRET. Le fait que le projet s'inscrive dans le contexte de l'IoT implique de choisir une unité de calcul efficace énergétiquement mais aussi simple et peu gourmande en termes de ressources sur le FPGA. Avec ces informations, on peut établir les points suivants :

1. Une architecture simple et peu gourmande qui effectue le travail d'un processeur et de ses fonctions basiques sans trop utiliser de ressource que ce soit en termes de silicium

ou bien d'un point de vue consommation énergétique afin de coller aux architectures présente dans les technologies utilisés en IoT.

2. Une architecture suffisamment bien documentée et/ou simple à comprendre afin de pouvoir la modifier et de pouvoir la partager avec la communauté tout en incluant les solutions que nous proposons.

Ces points font que les processeurs que nous allons chercher au sein de l'environnement RISC-V seront ceux suivant le jeu d'instruction RV32-I qui est le set élémentaire de la fondation RISC-V. De plus, dans l'idéal, le processeur trouvé n'aurait pas de mémoire cache afin de garder la simplicité du processeur.

Les coeurs présentés ci-après sont ceux sur lesquels une période de réflexion a été effectuée.

- Le coeur Rocket [110, 111] : Les coeurs Rocket sont relativements simple à générer à partir de l'environnement Rocket Chip. Malheureusement, même si cette solution permet de générer des coeurs de processeur qui pourrais convenir à SECRET : RV32-I ; ils sont écrits en Chisel et le RTL généré est en Verilog. De plus, le coeur possèderait une mémoire cache d'après la documentation. Cela implique l'apprentissage de deux nouveaux langages : Chisel et Verilog. Enfin, bien que la documentation soit relativement fournie, elle ne permet pas de connaitre l'architecture interne du coeur généré et impliquerait une étape d'ingénierie reverse afin de comprendre comment modifier ce coeur pour l'adapter au projet.

- la plateforme PULP [112] : Les solutions de la plateforme PULP 1 : L'environnement proposé par la plateforme PULP est assez vaste. Malheureusement, le plus petit coeur proposé (zero-RISC-V) possède déjà les extensions de multiplication et de division, ce qui n'est pas souhaité dans l'immédiat. De plus, les fichiers RTL disponible sont en verilog. La solution est donc écartée pour le moment. Néanmoins, elle reste intéressante si jamais nous souhaitons changer de coeur au cours du projet ou comparer notre solution avec un coeur de processeur léger qui pourrait convenir dans un contexte IoT.

- la solution Neorrv32 [113] : Cette solution est un SOC complet et ne convient pas au projet SECRET. Cependant, il est possible de télécharger la base d'un coeur RISC-V RV32I configurable avec un grand nombre d'extensions (ce qui ne nous intéresse a priori pas). Cette solution sera envisagée si le projet a besoin d'un coeur alternatif car le code RTL est en VHDL et partiellement commenté : il sera possible d'adapter l'architecture après une étape d'ingénierie inverse.

- La solution potato [114] :Il s'agit d'un petit processeur développé en VHDL qui pourrait convenir au projet. Son seul défaut est de présenter un code non commenté réparti sur un grand nombre de fichier. Cela fait que pour pouvoir travailler sur le

processeur une lourde étape d'ingénierie inverse est nécessaire pour comprendre le code. De plus cette solution est laissée de côté car elle n'est plus mise à jour et il n'y a presque pas de suivi.

- le coeur VexRISCV [115] : Ce coeur RISC-V a été développé dans un langage basé sur Chisel appelé SpinalHDL. Sa force est de pouvoir générer du Verilog ou du VHDL alors que chisel ne permet que de générer du Verilog. L'inconvénient majeur de cette solution réside dans le fait qu'il faut apprendre un langage supplémentaire. De plus, comme pour la majorité des coeurs disponible, celui proposé de base est très peu commenté pour des modifications. Cette solution a toutefois été explorée dans un premier temps car le fait de pouvoir générer VHDL et Verilog est très intéressant pour toucher une plus grande communauté internationale. Cependant, la solution a été écarté face à la difficulté d'apprentissage du langage mais aussi car la génération du RTL n'est pas hiérarchique : cela implique de retravailler les fichiers générés pour clarifier le code.

- le processeur COMET [116] : Ce coeur RISC-V a été développé à l'aide d'un langage utilisant le paradigme orienté objet du C++, offrant une flexibilité et une lisibilité notablement supérieures aux approches basées sur les langages de description matériels classiques (HDL). Cette spécification unique permet, via des méthodes de synthèse haut niveau, de générer à la fois du Verilog et du VHDL, alors que des solutions telles que Chisel se limitent à la production de Verilog, ce qui élargit le potentiel d'adoption internationale du cœur COMET. L'utilisation du language C++ permet aussi de créer à partir du même code source, tant le design matériel cible que des simulateurs performants et précis par compilation logicielle. La modification, l'expansion et la vérification du modèle processeur se trouvent ainsi facilitées par l'utilisation de méthodologies standards du développement logiciel. Toutefois, cette approche requiert l'utilisation d'une ressource logicielle supplémentaire pour la génération du cœur, et il faut noter que, comme pour la plupart des cœurs disponibles, la documentation en vue de modifications reste limitée. Malgré ces réserves, la possibilité de générer facilement des versions matérielles et simulées à partir du même code source représente un atout significatif pour l'évolution rapide de l'architecture, même si une simplification et une amélioration de la hiérarchisation du code généré pourraient encore optimiser cette méthode.

On choisit donc de se concentrer sur le processeur COMET comme plateforme de référence lors de notre projet. Ce coeur a été développé par l'équipe de l'Institut National de Recherche en Informatique et en Automatique (INRIA) de Rennes. Le fait qu'il s'agisse d'un laboratoire français a permis d'établir des interactions étroites, et a notamment conduit à un séjour de trois semaines en septembre 2022 afin de se familiariser avec l'utilisation du logiciel CATAPULT HLS pour la synthèse de descriptions matérielles en VHDL ou en Verilog, à partir d'une base de fichiers en C++. Ce séjour a également permis

de se familiariser avec l'agencement des fichiers et de comprendre le fonctionnement du simulateur du coeur COMET. Ce simulateur constitue un élément central de la démarche, car il permet de tester les fonctionnalités ajoutées au cœur RISC-V tout en fournissant des résultats préliminaires sur les opérateurs évalués. Les résultats présentés dans le chapitre suivant sont d'ailleurs issus de ce simulateur.

Afin de mieux appréhender les capacités et les choix de conception du cœur COMET, il est essentiel d'en examiner en détail le fonctionnement interne. La section suivante se propose ainsi de présenter les caractéristiques architecturales clés de ce processeur RISC-V, ainsi que les spécificités de son implémentation et de son pipeline, qui constituent le socle des développements menés dans ce projet.

3.2.3 COMET RISC-V

Le cœur RISC-V COMET a été conçu dans le but d'exploiter au maximum le potentiel de la synthèse haut niveau (High-Level Synthesis, HLS). Ce paradigme a pour objectif de venir contrer un problème du design RTL qui provient du fait de la complexification des systèmes embarqués, qui deviennent de plus en plus complets, comprenant de plus en plus d'élements à développer et à tester [117]. L'objectif du HLS, en utilisant un langage haut niveau comme par exemple le C est de permettre d'obtenir la description d'un système complet à partir du fichier décrivant son fonctionnement comportemental dans ce langage haut niveau ainsi que d'un outil traduisant ce fichier dans les langages fréquemment utilisés qui sont le Verilog ou encore le VHDL. Ce projet, initié et développé à l'INRIA Rennes, met en évidence l'intérêt de la HLS dans le domaine des architectures processeur. En plus de la rapidité de prototypage et de validation, la description haut niveau offre une souplesse importante, en permettant à la fois la simulation fonctionnelle et la génération automatique d'une description matérielle équivalente en RTL (VHDL ou Verilog). Cette approche réduit considérablement les contraintes liées à la maintenance d'un code RTL et permet des gains notables en productivité.

Le cœur COMET est un coeur totalement décrit à l'aide du langage C++, qui implémente l'ISA RISC-V RV32I, auquel il est possible d'ajouter certaines extensions, comme l'extension M introduisant les opérations de multiplication. Le processeur peut également intégrer une mémoire cache, bien que cette fonctionnalité ne soit pas recherchée lors du développement de la solution du fait de l'objectif d'avoir un système simplifié au maximum afin de représenter de petits objets issus de l'IoT. L'interface mémoire est conçue de manière générique : elle repose sur l'héritage d'une classe abstraite qui définit un ensemble d'entrées/sorties standardisées. Grâce à ce mécanisme, le cœur reste indépendant du type de mémoire utilisé, ce qui permet de substituer facilement des caches, des scratchpads ou encore une combinaison des deux, selon les besoins expérimentaux.

Le processeur fonctionne avec un pipeline à cinq étages, dont chacun est explicitement décrit par une fonction dédiée en C++. La Figure 3.3 illustre l'organisation interne du pipeline du cœur COMET, dont le fonctionnement repose sur une organisation séquentielle

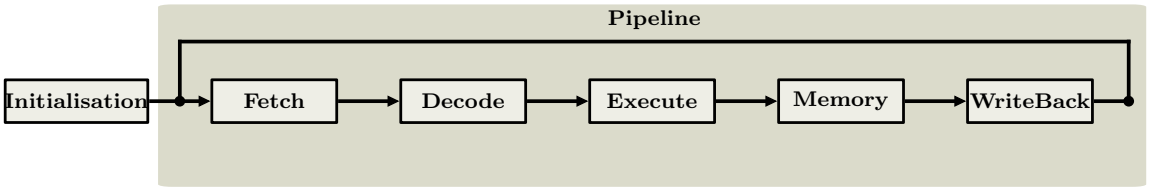


FIGURE 3.3 : Pipeline du processeur Comet

des étapes :

1. Fetch : récupération de l'instruction en mémoire.
2. Decode : décodage de l'instruction, réalisé à l'aide de structures switch-case adaptées aux différentes catégories d'instructions.
3. Execute : exécution de l'opération, incluant notamment les opérations arithmétiques et logiques, lui aussi réalisé à l'aide de structure switch-case.
4. Memory : accès mémoire pour les instructions de chargement et de stockage.
5. Write Back : écriture du résultat dans le registre de destination.

Un soin particulier a été porté à la gestion des dépendances et à la mise en place des mécanismes de forwarding, indispensables pour garantir le fonctionnement correct et performant du pipeline.

La description C++ du cœur peut ensuite être synthétisée à l'aide de l'outil Catapult HLS (Mentor) et de Design Compiler (Synopsys). L'utilisation de ces outils permet la génération automatique de fichiers VHDL ou Verilog directement à partir du code C++ haut niveau. Il est cependant important de noter qu'afin d'ajouter de nouvelles opérations, il faut effectuer une vérification manuelle du temps d'exécution desdites opérations, afin de s'assurer qu'elles n'introduisent pas de ralentissement significatif dans le pipeline, en créant par exemple un étage superflu non désiré dans la pipeline. Cependant ce n'est pas tout : comme le coeur est totalement décrit à l'aide de C++, il est possible de compiler le coeur et de l'utiliser sous la forme d'un simulateur C++ qui correspond strictement à la description matérielle générée. En effet, le simulateur ainsi produit est précis au cycle et au bit d'information prêt, ce qui facilite grandement le débogage (réalisé avec des outils standards tels que GDB) en comparaison avec une approche RTL classique, beaucoup plus laborieuse.

La conception du cœur COMET met en avant sa simplicité et sa modularité. Sa description, composée d'environ un millier de lignes de C++, peut être facilement comprise et modifiée, ce qui rend l'ajout de nouvelles instructions ou de mécanismes de suivi (par exemple le comptage des occurrences d'instructions) particulièrement aisé. De plus, le projet tire profit de pratiques modernes de développement logiciel, telles que l'intégration

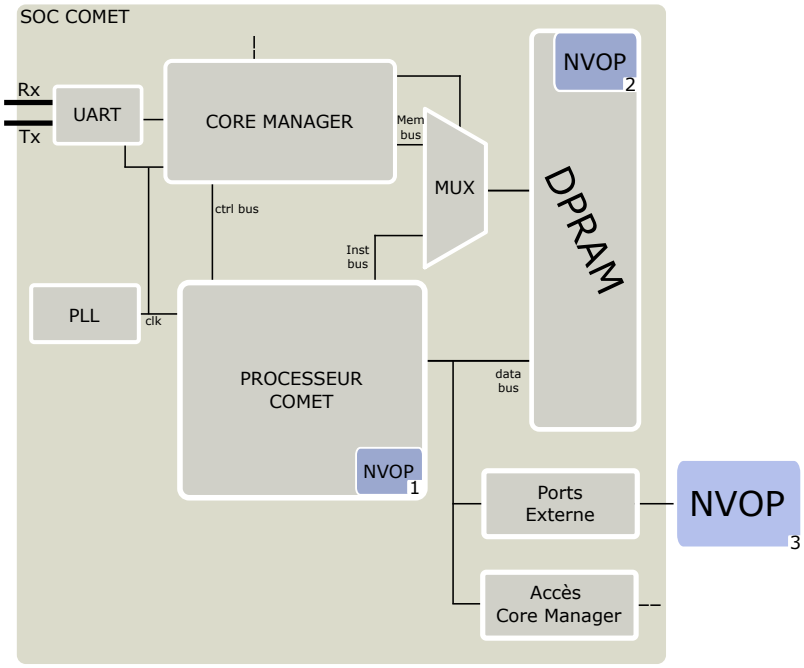


FIGURE 3.4 : Schéma complet de la plateforme d'émulation des opérateurs non-volatiles

continue, permettant de vérifier à chaque modification que le simulateur fonctionne correctement et qu'il est toujours synthétisable en matériel.

L'intérêt principal du cœur COMET réside ainsi dans sa flexibilité et son accessibilité : basé sur le jeu d'instructions RV32I, et pouvant intégrer ou non une mémoire cache, il constitue une base simple mais efficace pour l'expérimentation. Le fait que l'ensemble du pipeline soit décrit dans un unique fichier C++ facilite la prise en main et la modification de certains étages, condition indispensable à l'intégration d'instructions personnalisées dans le cadre de nos travaux.

Maintenant que le processeur COMET, élément central de la plateforme d'émulation matérielle, a été présenté, il est temps d'utiliser celui-ci dans la plateforme d'émulation et de présenter son fonctionnement global dans la prochaine section de ce chapitre.

3.3 La plateforme d'émulation complète

Maintenant que tous les éléments de la plateforme ont été présentés et que la méthode d'émulation des opérateurs a aussi été présentée, il convient maintenant de décrire la plateforme dans son ensemble ainsi que ses différents blocs. L'ensemble de la plateforme est représenté Figure 3.4, et chaque bloc va maintenant être présenté plus en détail.

Outre les principaux blocs de la plateforme qui sont le processeur COMET qui a pour

objectif d'effectuer tous les calculs et qui contient les opérateurs non-volatiles, et le Core Manager, qui, lui, est le contrôleur de la plateforme, la plateforme possède des blocs secondaires, présentés ci-dessous.

Le bloc horloge : Le bloc PLL pour Phase Lock Loop, boucle à verrouillage de phase en français, est le bloc servant d'horloge dans la plateforme.

Le bloc UART : Ce bloc permet la communication avec l'ordinateur connecté à la plateforme via une liaison UART, qui signifie receveur/émetteur asynchrone universel (Universal Asynchronous Receiver Transmitter). L'ensemble des données reçues par ce bloc est envoyé au Core manager et c'est aussi à l'aide de ce bloc que sont transférées les données relatives aux opérateurs non-volatile.

La DPRAM : La mémoire RAM double port (Dual-Port RAM) est la mémoire principale de la plateforme d'émulation, elle contient toutes les informations relatives aux données qui seront utilisées par la plateforme ainsi que l'ensemble des instructions du programme.

Le bloc des Ports Externe : Cette partie de la plateforme permettra dans un temps futur de se connecter à des périphériques nécessitant l'utilisation des technologies de mémoire non-volatile.

Le bloc d'Accès Core Manager : Ce bloc permet de faire la liaison entre le Core manager, qui contrôle toute la plateforme, et le processeur COMET à l'aide d'instructions ajoutées au set d'instructions de base RISC-V.

le bloc MUX : Un multiplexeur qui va faire en sorte que soit la DPRAM reçoit les données envoyées par le core manager, soit la DPRAM va envoyer la suite d'instructions du programme au processeur COMET.

Core Manager : L'objectif de cette partie de la plateforme est donc de contrôler l'ensemble de la plateforme d'émulation. Pour ce faire, le Core manager possède 4 commandes de contrôle sur 8 bits lui permettant d'interagir avec le reste de la plateforme. Ces commandes sont les suivantes :

- Obtenir l'ensemble du programme de la part de l'ordinateur via la connexion UART et l'envoyer vers la DPRAM,
- Envoyer tout ou partie de la mémoire de la plateforme d'émulation vers l'ordinateur,
- Mettre le CPU COMET en route en envoyant un signal d'activation via le control bus reliant le CORE Manager au processeur.

- Mettre le CPU en pause si un signal est reçu dans le Core manager via le bloc d'accès au Core manager

Ces quatre commandes permettent une utilisation du processeur COMET. De plus, la dernière instruction, permettant de réaliser une mise en pause du processeur à n'importe quel moment, permet d'obtenir les informations dans la mémoire de la plateforme, comme par exemple après un calcul critique de la part de la plateforme.

Le processeur COMET RISC-V : Le bloc principal de la plateforme, comme présenté précédemment, le processeur COMET est un processeur RISC-V RV32I auquel on a rajouté de nouvelles instructions afin de pouvoir effectuer nos opérations d'émulation, ou de simulation lorsqu'on utilise le simulateur fourni, et de tests de nos opérateurs non-volatiles. Comme l'objectif de la thèse est de tester nos opérateurs avec le processeur le plus simple possible du fait du contexte de l'IoT, la mémoire cache a été retirée.

Les opérateurs non-volatiles : Ce bloc peut être implémenté de trois manières distinctes. Tout d'abord, le bloc numéro 1 qui n'est pas là en lui-même étant donné qu'il est inclus sous la forme d'instructions dans notre processeur COMET RISC-V. Chacune des nouvelles instructions sera dédiée à l'utilisation d'un nouvel opérateur. En pratique à l'utilisation, elles sont placées dans le code C à l'endroit où les opérateurs vont être utilisés à l'aide de l'assembleur inline. Utiliser cette méthode permet d'enlever l'étape de modification du compilateur pour utiliser les instructions préparées. C'est avec cette méthode que sont présentés les résultats du chapitre suivant. Ensuite, le bloc numéro 2 vient représenter une implémentation dans la cellule mémoire pour effectuer du IMC. Pour cette méthode, les informations à traiter seront référencées et placées directement à des adresses mémoires spécifiques qui effectueront les calculs et écriront le résultat de ces opérations dans une autre adresse dédiée à l'opérateur. Enfin le Bloc 3 vient représenter la méthode utilisant un autre élément connecté au SOC. Cela peut représenter un autre FPGA utilisant une émulation nécessitant plus de ressources matérielles que disponibles sur une seule puce ou bien encore des modèles physiques des opérateurs construits afin de tester leur fonctionnement à la suite de leur élaboration plus poussée.

3.4 Conclusions

Au cours de ce chapitre, la plateforme d'émulation a été détaillée dans son ensemble, afin de comprendre la démarche suivie pour sa conception. Son fonctionnement, ainsi que la manière dont les opérateurs basés sur les mémoires non-volatiles (NVM) sont implémentés et émulés, ont été présentés dans le but de permettre leur évaluation.

Le choix du processeur RISC-V COMETpermet d'intégrer au sein de la plateforme un processeur développé en France, reposant sur le jeu d'instructions RISC-V, aujourd'hui

largement adopté par la communauté scientifique. La plateforme présentée fonctionne actuellement en simulation, à l'aide du processeur COMET, et est destinée à être déployée ultérieurement sur FPGA.

À terme, l'objectif est de disposer d'un processeur pleinement fonctionnel sur la plateforme matérielle et d'exploiter les ports externes afin de connecter des périphériques, dans le but d'élargir le champ des tests réalisés, notamment vers des systèmes plus complexes. De plus, le cœur COMET permet d'obtenir des résultats préliminaires par simulation, facilitant ainsi la validation des choix architecturaux avant l'implémentation matérielle.

Ces premiers résultats seront présentés dans le chapitre suivant, au travers d'un cas d'étude portant sur l'algorithme de sécurisation des données AES.

Chapitre 4

Tests et Expériences

4.1 Présentation du contexte de l'expérience : l'AES et les Corps de Galois 59

4.1.1 L'Advanced Encryption Standard : Un algorithme de cryptographie impactant 59

4.1.2 Les Corps de Galois : la part mathématique de l'AES 63

4.2 Mise en place des opérateurs non-volatiles à base de FeFETs . 67

4.2.1 Création de la carte modèle d'un FeFET 67

4.2.2 Création de la carte modèle d'un FeMFET 69

4.3 Expériences et Résultats 73

4.3.1 La mise en place de l'opération de MixColumn 74

4.3.2 Fonctionnement du simulateur 78

4.3.3 Présentation des résultats 79

4.4 Conclusion 86

La mise en place d'une plateforme d'émulation dédiée aux opérateurs non-volatiles, détaillée au chapitre précédent, constitue une première étape vers l'évaluation concrète de l'intégration de ces opérateurs dans des architectures de traitement standards. Afin de valider l'intérêt de la méthode présentée, il est nécessaire de s'appuyer sur un cas d'usage représentatif à la fois des contraintes réelles du monde applicatif et des exigences fonctionnelles d'un traitement embarqué. C'est dans ce cadre que s'inscrit ce chapitre, qui s'attache à tester et analyser l'impact de l'intégration d'opérateurs non-volatiles, inspirés des technologies émergentes comme les FeFET, à travers l'exécution partielle d'un algorithme cryptographique de référence : l'AES (Advanced Encryption Standard) [118].

Le choix de l'AES comme base expérimentale se justifie par plusieurs raisons. Tout d'abord, comme présenté en introduction du manuscrit dans le contexte de l'IoT avec le projet SECRET, la confidentialité des données échangées est cruciale. Les objets connectés génèrent un volume croissant de données sensibles, qu'elles soient médicales, bancaires ou industrielles, qui nécessitent une couche supplémentaire de sécurisation, ce qui se traduit généralement par des mécanismes de chiffrement sur des modules spécifiques comme la méthode proposée en Figure 1.2b. Cet algorithme occupe une place centrale dans les protocoles de sécurité modernes. En effet, en tant qu'algorithme standardisé et largement adopté, il constitue de ce fait un cas d'étude pertinent.

D'autre part, l'AES présente une structure modulaire composée de plusieurs opérations mathématiques, dont certaines se prêtent particulièrement bien à une implémentation matérielle dédiée, voire à une réécriture sous forme d'opérateurs spécialisés. En particulier, les opérations définies sur des corps de Galois, au cœur du fonctionnement de l'AES, nécessitent des calculs algébriques spécifiques qui offrent un terrain favorable à l'étude de blocs logiques alternatifs, qu'ils soient volatiles ou non-volatiles.

L'objectif de ce chapitre est double. Il s'agit d'une part de démontrer la capacité de la plateforme d'émulation à prendre en charge l'exécution d'opérateurs complexes intégrés dans un cœur RISC-V à l'aide de la méthodologie présentée précédemment. D'autre part, il permet d'analyser l'influence de ces opérateurs sur le comportement global du système. Ce travail est mené à l'aide d'un simulateur logiciel, dans lequel les opérateurs non-volatiles sont représentés par des modèles comportementaux insérés dans le flot d'instructions du processeur.

Le chapitre se découpe de la manière suivante. Tout d'abord, une présentation du contexte cryptographique de l'AES, de ses fondements algébriques, et des opérations ciblées pour l'intégration d'opérateurs spécialisés. La partie suivante décrit la mise en œuvre concrète de ces opérateurs au sein de l'environnement d'émulation, avant de conclure par une analyse des résultats obtenus, illustrant les apports et les limites de cette approche dans un cadre d'expérimentation réaliste.

4.1 Présentation du contexte de l'expérience : l'AES et les Corps de Galois

L'AES, pour Advanced Encryption Standard, est l'un des algorithmes de cryptographie les plus utilisés au monde, qui a été standardisé en 2001. Cet algorithme de cryptographie symétrique moderne assure la protection des données sensibles à l'échelle mondiale. Sa conception repose fortement sur l'utilisation des mathématiques des corps de Galois, qui fournissent la structure algébrique nécessaire à la réalisation de ses transformations fondamentales. Un point fort majeur des algorithmes de cryptographie est que nombre d'entre eux utilisent des constantes publiquement connues qui restent les mêmes pour une longue période de temps et l'AES ne fait pas exception. Au vu des capacités de nos futurs opérateurs à conserver en mémoire pour une longue durée les informations, cela fait de l'AES un candidat idéal pour tester ces capacités de stockage. Cette section propose une présentation approfondie de l'AES, puis explore les fondements mathématiques des corps de Galois, et se conclut par des exemples concrets illustrant leur mise en œuvre dans l'algorithme.

4.1.1 L'Advanced Encryption Standard : Un algorithme de cryptographie impactant

4.1.1.1 Introduction à l'AES

La norme AES (Advanced Encryption Standard) est un algorithme de chiffrement par blocs symétrique devenu la norme mondiale en matière de chiffrement des données sensibles. Développé en réponse aux vulnérabilités de son prédécesseur, le DES (Data Encryption Standard), l'AES a été mis au point par le NIST (National Institute of Standards and Technology) aux États-Unis et publié officiellement sous le nom de FIPS PUB 197 en 2001 [119]. L'AES est né d'un concours public lancé par le NIST à la fin des années 1990 pour remplacer le DES, qui était devenu vulnérable aux attaques par force brute. L'algorithme gagnant, connu sous le nom de Rijndael [120] (développé par Joan Daemen et Vincent Rijmen), a été choisi pour sa combinaison de sécurité, de performance et de flexibilité.

L'AES est considéré comme hautement sécurisé et, à ce jour, aucune attaque mathématique pratique n'a été trouvée qui puisse compromettre l'algorithme complet lorsqu'il est utilisé avec des clés fortes. Sa conception résiste à toutes les formes connues de cryptanalyse, y compris la cryptanalyse linéaire et différentielle, grâce à sa structure de réseau de substitution-permutation, à la conception solide de la « S-box » et au mélange minutieux des clés. La principale menace qui pèse sur l'AES est la recherche de clés par force brute, qui est irréalisable en pratique pour les clés de 128 bits et plus avec la technologie actuelle.

Depuis son adoption, l'AES a été mis en œuvre dans d'innombrables applications. Sa

capacité à être mis en œuvre à la fois dans le logiciel et dans le matériel le rend adapté à un large éventail d'applications. Son efficacité lui permet d'être utilisé dans des environnements à ressources limitées tels que les systèmes embarqués et les cartes à puce, ainsi que dans des systèmes à haut débit tels que les réseaux privés virtuels et le chiffrement des disques. L'AES est la norme de sécurisation des informations classifiées du gouvernement américain et son utilisation est obligatoire dans de nombreux protocoles et normes internationales, notamment dans les domaines des communications sécurisées, du chiffrement de fichiers et des modules de sécurité matériels. Aujourd'hui, l'algorithme est largement reconnu comme la référence en matière de chiffrement des données dans les secteurs gouvernementaux et commerciaux en raison de sa sécurité robuste, de son efficacité et de sa polyvalence.

Les principaux avantages de l'AES sont sa grande sécurité, son efficacité et sa flexibilité sur toutes les plateformes. Sa principale limite est la nécessité d'une gestion sécurisée des clés, car la sécurité du chiffrement symétrique dépend entièrement du secret de la clé. Si la clé est compromise, les données chiffrées le sont aussi. En outre, bien que l'AES soit très résistant à la cryptanalyse classique, l'avènement de l'informatique quantique pourrait menacer les chiffrements symétriques à l'avenir, bien que l'AES-256 soit considéré comme résistant à la quantique dans une certaine mesure.

L'Advanced Encryption Standard est la pierre angulaire de la cryptographie moderne, alliant une sécurité robuste à une efficacité pratique. Son adoption en tant que norme mondiale témoigne de sa résistance aux attaques et de son adaptabilité à divers environnements technologiques. Pour toute personne travaillant dans le domaine de la sécurité de l'information, une compréhension approfondie de l'AES est essentielle, car il sous-tend la confidentialité et l'intégrité des communications numériques et du stockage des données dans le monde contemporain.

4.1.1.2 Le fonctionnement de l'AES

Maintenant que l'AES a été présenté dans la partie précédente, son fonctionnement est présenté ici. L'AES est un algorithme à clé symétrique, ce qui signifie que la même clé secrète est utilisée pour le chiffrement et le déchiffrement. Cela contraste avec les algorithmes asymétriques, qui utilisent des clés publiques et privées distinctes. En tant que chiffrement par bloc, l'AES traite les données par blocs discrets de 128 bits (16 octets) à la fois. Il prend en charge trois longueurs de clé de chiffrement : 128, 192 et 256 bits. Le choix de la longueur de clé affecte directement la sécurité de l'algorithme et le nombre de tours de transformation effectués lors du chiffrement et du déchiffrement.

L'algorithme fonctionne avec trois tailles de clé possibles :

- **l'AES-128** : utilise une clé de 128 bits et effectue 10 tours de transformation,
- **l'AES-192** : utilise une clé de 192 bits et effectue 12 tours de transformation,
- **l'AES-256** : utilise une clé de 256 bits et effectue 14 tours de transformation,

Chaque tour consiste en une série de transformations cryptographiques conçues pour obscurcir complètement la relation entre le texte en clair et le texte chiffré. Plus la longueur de la clé est importante, plus le nombre de tours est élevé, ce qui renforce la sécurité mais augmente également la complexité des calculs à effectuer.

Le processus de chiffrement de l'AES est très structuré et se compose de plusieurs étapes clés :

1. Expansion de la clé : La clé secrète d'origine subit un processus d'expansion de clé, générant une série de clés rondes. Ces clés rondes sont dérivées de la clé d'origine à l'aide d'un algorithme de planification des clés, garantissant que chaque tour de chiffrement utilise une clé unique. Le nombre de clés rondes générées dépend de la longueur de la clé et du nombre de tours.

2. Tour initial (AddRoundKey) : Avant le début des tours principaux, le bloc de texte en clair est combiné à la clé de chiffrement lors d'un tour initial, à l'aide d'une opération OU exclusif (XOR) bit à bit. Cette étape permet d'intégrer la clé dans les données dès le départ, ce qui constitue une première couche de sécurité.

3. Boucle des tours principaux : Chaque tour principal de l'AES consiste en quatre transformations fondamentales appliquées au bloc de données, qui est représenté en interne sous la forme d'une matrice de 4×4 octets appelée « tableau d'état » :

- **SubBytes :** Il s'agit d'une étape de substitution non linéaire au cours de laquelle chaque octet du tableau d'état est remplacé par un autre octet à l'aide d'une boîte de substitution (S-box). La S-box est soigneusement conçue pour résister aux attaques cryptographiques en offrant une forte non-linéarité et une grande confusion.
- **ShiftRows :** Les lignes du tableau d'état sont décalées cycliquement par différents décalages. La première ligne reste inchangée, la deuxième ligne est décalée d'un octet, la troisième de deux octets et la quatrième de trois octets. Cette étape permet de disperser les données dans les colonnes, ce qui améliore la diffusion.
- **MixColumns :** Chaque colonne du tableau d'état subit une transformation linéaire utilisant la multiplication matricielle sur un corps fini (corps de Galois $GF(2^8)$). Cette opération mélange les octets dans chaque colonne, diffusant davantage les données et les rendant plus résistantes aux attaques.
- **AddRoundKey :** Le tableau d'état est combiné avec la clé du tour en cours à l'aide d'un XOR, en intégrant le matériel de la clé à chaque étape.

Le nombre de tours principaux dépend de la taille de la clé, l'AES-128 en a 9, l'AES-192 en a 11 et l'AES-256 en a 13.

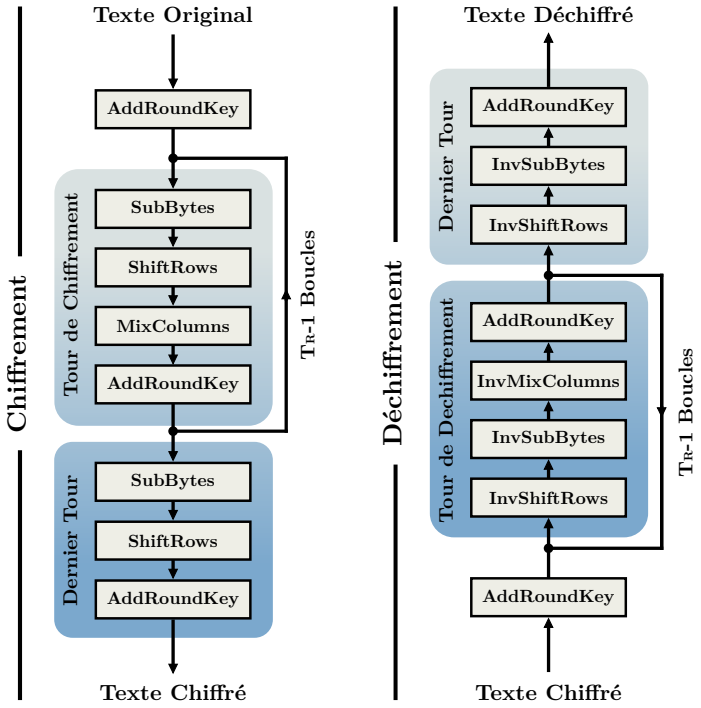


FIGURE 4.1 : Schéma des tours de l'algorithme AES pour le chiffrement (à gauche) et le déchiffrement (à droite)

4. Dernier tour : Le dernier tour d'AES omet l'étape MixColumns, mais inclut SubBytes, ShiftRows et AddRoundKey. Cela permet de s'assurer que le texte chiffré est entièrement brouillé et bénéficie d'un niveau de protection élevé.

Pour résumer toutes ces transformations, un schéma est proposé en Figure 4.1, présentant sous forme de frise l'ensemble des étapes du chiffrement.

Le déchiffrement de l'AES est essentiellement l'inverse du processus de chiffrement. Il s'agit d'appliquer les opérations inverses de chaque transformation dans l'ordre opposé, en utilisant le même jeu de clés rondes, également appliquées en sens inverse. Les opérations inverses sont InvSubBytes, InvShiftRows, InvMixColumns et AddRoundKey. Cette symétrie entre le chiffrement et le déchiffrement est une caractéristique des chiffrements par blocs symétriques et contribue à l'efficacité de l'algorithme.

Bien que l'AES soit le principal algorithme de chiffrement par bloc, il est souvent utilisé avec différents modes de fonctionnement pour chiffrer des flux de données ou des fichiers plus volumineux qu'un seul bloc. Les modes les plus courants sont l'Electronic Codebook (ECB), qui correspond à la forme la plus simple d'utilisation de l'algorithme, le Cipher Block Chaining (CBC) [121], le Counter (CTR) [122] et le Galois/Counter Mode

(GCM) [123]. Chaque mode offre des compromis différents en termes de sécurité, de propagation des erreurs et de parallélisme. Le mode GCM, par exemple, assure à la fois la confidentialité et l'authenticité et est largement utilisé dans les protocoles de communication sécurisés modernes.

4.1.2 Les Corps de Galois : la part mathématique de l'AES

4.1.2.1 Présentation des corps de Galois

Un Corps de Galois (également connu sous le nom de corps fini) est une structure mathématique composée d'un nombre fini d'éléments, dotée de deux opérations, l'addition et la multiplication, satisfaisant les axiomes d'un champ. Les corps de Galois sont notés $GF(p^n)$, où p est un nombre premier (la caractéristique du corps), et n est un entier positif. Les corps les plus courants en cryptographie, tels que ceux utilisés dans l'AES, sont de la forme $GF(2^n)$, ce qui signifie qu'ils contiennent 2^n éléments et opèrent sur des nombres binaires. Dans $GF(2^n)$, chaque élément peut être représenté par un nombre binaire de n bits ou, de manière équivalente, par un polynôme de degré inférieur à n avec des coefficients dans $GF(2)$ (c'est-à-dire 0 ou 1).

Par exemple : l'octet 11001010 dans $GF(2^8)$ correspond au polynôme $x^7 + x^6 + x^3 + x$. Cette représentation polynomiale est essentielle pour comprendre comment l'arithmétique, en particulier la multiplication, est effectuée.

La multiplication des orps de Galois est fondamentale pour la cryptographie, les codes correcteurs d'erreurs et les communications numériques. Dans l'AES, elle joue un rôle central dans la transformation MixColumns, qui assure la diffusion en mélangeant les octets d'une manière mathématiquement robuste. L'utilisation de $GF(2^8)$ permet d'effectuer des opérations efficaces, sûres et prévisibles sur des octets de 8 bits, ce qui correspond à la structure des systèmes numériques modernes. La multiplication des corps de Galois est une pierre angulaire de la cryptographie moderne et des systèmes numériques. En représentant les éléments comme des polynômes et en effectuant une multiplication suivie d'une réduction modulaire avec un polynôme irréductible, $GF(2^n)$ permet des opérations mathématiquement solides. C'est cette structure qui permet à l'AES et à des algorithmes similaires d'atteindre un niveau de sécurité et de performance élevés dans la pratique.

4.1.2.2 Fonctionnement des opérations dans le corps de Galois

La présentation des corps finis étant faite, il est maintenant important d'expliquer le fonctionnement des différentes opérations dans le corps de Galois.

L'addition dans $GF(2^n)$ est simple : elle est réalisée en ajoutant les coefficients correspondants modulo 2, ce qui est équivalent à l'opération XOR bit à bit. Par exemple, dans $GF(2)$:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 1 = 0$

On peut constater à travers cet exemple que l'addition et la soustraction sont la même opération. De plus, cette simplicité rend l'addition très efficace dans les implémentations matérielles et logicielles.

La multiplication dans un corps de Galois est plus complexe que l'addition, en particulier dans les champs de la forme $GF(2^n)$. Le processus comporte deux étapes principales :

- **1. Multiplication polynomiale :** Multiplier les deux polynômes représentant les éléments comme vous le feriez en algèbre ordinaire, mais avec toute l'arithmétique des coefficients effectuée modulo 2 (c.-à-d., 1 + 1 = 0).
- **2. Réduction modulaire :** Réduire le polynôme résultant modulo un polynôme irréductible de degré n. Cela garantit que le résultat reste dans le corps, c'est-à-dire qu'il s'agit d'un polynôme de degré inférieur à n.

Par exemple : Supposons qu'il faille multiplier deux éléments, A et B, dans $GF(2^8)$:

- La première étape est d'exprimer les deux valeurs sous la forme de polynômes : Soit $A = a_7x^7 + \cdots + a_0$ et $B = b_7x^7 + \cdots + b_0$.
- Ensuite, il faut calculer le produit $C(x) = A(x) \cdot B(x)$. Cela peut donner un polynôme de degré inférieur ou égal à 14.
- Enfin, afin de retourner dans le corps, à savoir un polynôme de degré inférieur à 8, il faut diviser $C(x)$ par le polynôme irréductible du corps choisi $P(x)$ et garder le reste. Cela garantit que le résultat obtenu est un polynôme de degré inférieur à 8, correspondant à une valeur de 8 bits.

Le choix du polynôme irréductible utilisé dans la multiplication définit la structure du corps. Dans l'AES ($\mathrm{GF}(2^8)$), le polynôme irréductible standard est : $P(x) = x^8 + x^4 + x^3 + x + 1$ ou encore 0x11B en hexadécimal. Cela permet de s'assurer que le résultat ne dépasse pas les 8 bits d'information avec la présence de x^8.

Ce polynôme ne peut pas être factorisé en polynômes de degré inférieur sur $GF(2)$, ce qui est essentiel pour que le corps ait les propriétés mathématiques souhaitées.

Dans la pratique, en particulier dans l'AES, la multiplication est effectuée à l'aide d'un algorithme « sans retenue » qui peut être résumé de la manière suivante :

On effectue une boucle pour chaque bit d'un des opérandes, B dans notre cas en allant du bit le moins important au plus important :

- Si le bit est à 1, faire un XOR de la valeur actuelle de A dans le résultat.

- Décaler A d'un bit vers la gauche. Si le bit le plus à gauche (avant le décalage) était à 1, faire un XOR de A avec le polynôme irréductible (sans le terme x^8, puisque le terme le plus élevé est toujours réduit).
- Répéter l'opération pour les 8 bits.

Ce processus est efficace et bien adapté au matériel binaire.

Finalement, il reste à parler des propriétés de la multiplication des corps de Galois. Tout d'abord le produit de deux éléments quelconques du corps est également dans le corps. La multiplication est à la fois associative et commutative. Il existe un élément d'identité unique qui est la valeur 1. Enfin, chaque élément non nul a un inverse de telle manière que le produit de l'inverse et du nombre donne l'élément neutre.

4.1.2.3 Un exemple de Multiplication dans les Corps de Galois

Afin de montrer étape par étape l'algorithme de multiplication classique qui va être utilisé dans l'expérience, un exemple détaillé va être fait dans cette sous-section. Pour ce faire, on va se placer dans le contexte de l'AES en utilisant les deux octets $A = 10111001$ et $B = 00011110$ dans $GF(2^8)$, avec le polynôme irréductible $P(x) = x^8 + x^4 + x^3 + x + 1$ (en hexadécimal 0x11B) :

Étape 1 : Initialisation

- Le premier opérande : $A = 10111001$ (0xB9),
- Le second opérande : $B = 00011110$ (0x1E),
- Le Résultat : $R = 00000000$,
- Le Polynôme irréductible de l'AES : $P(x) = x^8 + x^4 + x^3 + x + 1$ (0x11B) utilisé pour la réduction si le bit de A est défini après le décalage.

Étape 2 : Multiplication itérative : Nous traitons chaque bit de B, du moins significatif au plus significatif (de droite à gauche) :

Bit 0 (b0 = 0) :

- Aucune action (puisque b0 est 0).
- Décalage de A vers la gauche : 01110010, le MSB était 1, donc après le décalage, XOR avec 0x1B : 01110010 XOR 00011011 = 01101001, on note ici le polynôme irréductible 0x1B car le bit de poids fort va toujours finir à 0 du fait que l'opération ne s'effectue que si le MSB de A vaut 1.

- Décalage B à droite : 00001111.

Bit 1 (b1 = 1) :

- R = R XOR A $\Longleftrightarrow$ 00000000 XOR 01101001 = 01101001.
- Décalage A vers la gauche : 11010010 (le MSB était 0, donc pas de réduction).
- Décalage B à droite : 00000111.

Bit 2 (b2 = 1) :

- R = R XOR A $\Longleftrightarrow$ 01101001 XOR 11010010 = 10111011.
- Décalage A vers la gauche : 10100100, le MSB était 1, donc après le décalage, XOR avec 0x1B : 10100100 XOR 00011011 = 10111111.
- Décalage B à droite : 00000011.

Bit 3 (b3 = 1) :

- R = R XOR A $\Longleftrightarrow$ 10111011 XOR 10111111 = 00000100.
- Décalage A vers la gauche : 01111110, le MSB était 1, donc après le décalage, XOR avec 0x1B : 01111110 XOR 00011011 = 01100101.
- Décalage B à droite : 00000001.

Bit 4 (b4 = 1) :

- R = R XOR A $\Longleftrightarrow$ 00000100 XOR 01100101 = 01100001.
- Décalage A vers la gauche : 11001010 (le MSB était 0, donc pas de réduction).
- Décalage B à droite : 00000000.

Bit 5, 6, 7 (tous 0) :

- Aucune action (puisque les bits sont à 0).
- Continuer à décaler A et B, mais aucun autre XOR n'est effectué

Étape 3 : Le résultat final : Une fois que tous les bits de B ont été traités, la valeur dans R est le produit dans $GF(2^8)$. Soit $R = 01100001$ ou 0x61 en hexadécimal. Il s'agit du résultat de la multiplication de 0xB9 (10111001) par 0x1E (00011110) dans le champ de Galois AES à l'aide du polynôme irréductible 0x1B.

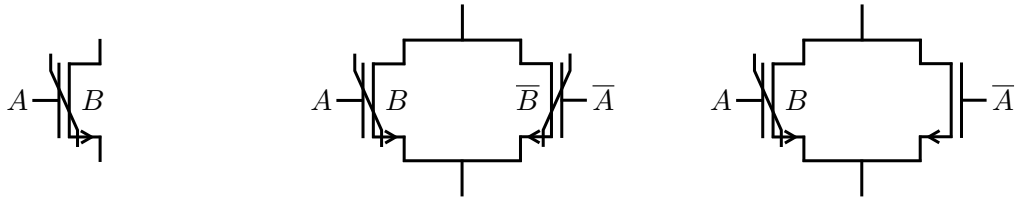
Après l'exemple si on doit résumer l'algorithme, pour chaque bit de l'opérande (ici B), si ce dernier est à la valeur 1, le résultat R est XOR avec la valeur actuelle de A, sinon aucune action n'est nécessaire. Ensuite, il faut décaler A de 1 bit vers la gauche, si le bit de poids fort de A valait 1 avant le décalage, il faut XOR A avec le polynôme irréductible de l'AES. Enfin, il faut décaler B d'un bit vers la droite. Une fois l'étape répétée autant de fois que nécessaire, le résultat est contenu dans R. Cette méthode, connue sous le nom de multiplication sans retenue avec réduction modulaire, est précisément la manière dont AES met en œuvre la multiplication par corps de Galois pour ses opérations MixColumns et S-box.

4.2 Mise en place des opérateurs non-volatiles à base de FeFETs

Après avoir présenté l'algorithme AES et les fondements mathématiques sur lesquels reposent les fonctions ciblées, cette section se concentre sur l'implémentation concrète des opérateurs non volatils au sein de notre architecture. L'objectif est de modéliser le comportement d'un FeFET en tant qu'unité de calcul, puis d'intégrer cette modélisation dans un cœur de processeur afin d'en évaluer l'impact sur l'exécution d'un algorithme cryptographique. La première sous-partie est ainsi dédiée à la construction d'une carte modèle représentant de manière abstraite le fonctionnement d'un FeFET, en tenant compte de ses propriétés structurelles et électriques. La seconde sous-partie décrit la démarche d'insertion de ces opérateurs dans l'environnement d'émulation associé à notre cœur RISC-V, en précisant les adaptations architecturales nécessaires pour garantir une intégration fonctionnelle et cohérente.

4.2.1 Création de la carte modèle d'un FeFET

Comme présenté dans le chapitre précédent, l'émulation à l'aide de modèles réalistes consomme trop de ressources dans les outils matériels utilisés. Il a donc fallu créer un modèle comportemental afin d'obtenir un fonctionnement réaliste des opérateurs utilisés. Ainsi, l'objectif de la carte modèle est de permettre l'intégration d'opérateurs NVM dans un coeur RISC-V sans implémentation physique réelle, tout en conservant un comportement temporel viable et en permettant une évaluation a posteriori de la consommation énergétique de notre carte. L'avantage de cette solution est qu'elle permet aussi une grande flexibilité. En effet, tant que les caractéristiques temporelles sont connues, il est possible



(a) (b) (c)

FIGURE 4.2 : Schéma des différentes portes FeFET avec (a) un NAND, (b) un XOR, et (c) un AND issu de [5]

d'avoir une idée du temps nécessaire pour la progression des calculs pour n'importe quelle technologie de mémoire non-volatile. De plus, en connaissant la construction du circuit, il est possible, si les consommations énergétiques sont connues, d'avoir une estimation globale du circuit lors de l'exécution de calculs.

La façon dont la carte modèle est créée est la suivante : Il est nécessaire de disposer des informations en termes de temps nécessaires pour chacune des situations de l'objet à émuler, ainsi que les consommations pour ces mêmes opérations. Dans le cadre de nos expériences avec les FeFET, nous nous basons sur des portes logiques simples afin d'avoir ces informations. De plus, afin d'avoir des évaluations pessimistes, les valeurs utilisées seront les cas les plus défavorables.

Afin de bien présenter, nous nous basons sur un exemple de situation. Dans le cas de la lecture pour un opérateur NVM basé sur des FeFETs, plusieurs paramètres doivent être pris en compte.. Il est nécessaire, tout d'abord, de récupérer les informations sur le temps mis pour chaque cas dans la lecture. Il faudra adapter le temps pris par l'utilisation de la porte logique constituée du transistor ferroélectrique à la fréquence de notre unité de calcul. Pour le cas de la porte NAND réalisée par un unique transistor ferroélectrique nFET comme indiqué Figure 4.2. De plus, les portes sont toutes constituées soit d'un transistor ferroélectrique, soit d'un maximum de deux transistors en parallèle. Par conséquent, le temps de propagation à travers ces portes est équivalent à celui d'un seul transistor.

L'étape suivante consiste à considérer le pire cas possible pour le temps de lecture, à savoir lorsque la valeur stockée dans le transistor est un 1 logique et que l'entrée est également positionnée à 1. Dans ce cas, le temps de lecture d'une donnée est de 260,3 ns, d'après les valeurs d'émulation obtenues au sein de l'équipe. Le processeur COMET possède une vitesse de 100 MHz, soit l'équivalent d'un cycle toutes les 10 ns. Pour effectuer une lecture de la donnée contenue dans notre porte NAND non-volatile, il faut

Partie Concernée	Lecture dans le FeFET (en termes de cycles du cœur)	Écriture dans le FeFET
Valeurs Actuelles	27	54
Valeurs Prévues	1	10

TABLE 4.1 : Informations sur les paramètres utilisés pour les premiers tests

donc prendre un total de 27 cycles processeur. Bien que la valeur soit proche de 260 ns, il faut prendre en compte le temps complet du passage à travers la porte et du cycle processeur comme ce dernier est de 10 ns précisément, il faut rajouter un cycle processeur aux 26 cycles initialement arrondis. De plus, afin de ne pas perturber le pipeline du processeur, pendant les moments où les opérateurs sont utilisés, le processeur sera mis en état d'attente pendant le nombre de cycles nécessaires pour reproduire le temps requis par les cellules conçues.

Avec une première version des FeFETs, une série de tests est effectuée avec les valeurs du Tableau 4.1. Le tableau présente deux jeux de valeurs. Le premier correspond à celles obtenues à l'aide d'un premier modèle de FeFET qui est basé sur le modèle de FeFET carré de $500nm$ [5] et dont les valeurs de lecture et d'écriture ont été obtenues par des simulations sur le logiciel Cadence. Le second jeu de valeurs correspond à des projections estimées proposées par l'équipe et issues de la littérature. Ces projections indiquent un temps d'écriture d'environ 100 ns [124] et un temps de lecture d'environ 5 ns, ce qui correspond, comme indiqué dans le tableau, à 10 cycles pour l'écriture et 1 cycle pour la lecture.

Deux choses sont remarquables sur ce premier jeu de valeurs : l'absence de consommation dans l'ensemble des valeurs présentées, et la valeur 54, qui correspond au double des 27 cycles de lecture. La raison pour ces deux choses est que pour la première série de modèles de FeFET, les valeurs écrites étaient déjà préenregistrées et avaient pour but de vérifier le fonctionnement du modèle. La valeur correcte est donc les 27 cycles de lecture, correspondant aux 260,3 ns. Ainsi, afin de tester l'écriture, une valeur pessimiste du double du temps de lecture a été retenue pour effectuer la première série de tests. En ce qui concerne la consommation, elle n'a pas été mesurée, car l'objectif principal était d'obtenir un modèle fonctionnel.

4.2.2 Création de la carte modèle d'un FeMFET

Dans un deuxième temps, un autre modèle a été utilisé [125]. Pour cette nouvelle version, le FeFET utilisé précédemment, qui est une version dont le matériau ferroélectrique est directement placé sur une couche d'isolant diélectrique avant le substrat du transistor (aussi appelé MFIS pour « Métal-Ferroélectrique-Isolant-Substrat ») comme montré dans

la Figure 4.3a, est remplacé par une version où une couche métallique est intercalée entre l'isolant et la couche de ferroélectriques comme montré Figure 4.3b. Cette version des FeFETs de type MFMIS pour « Métal-Ferroélectrique-Métal-Isolant-Substrat » [126] est aussi appelée FeMFET.

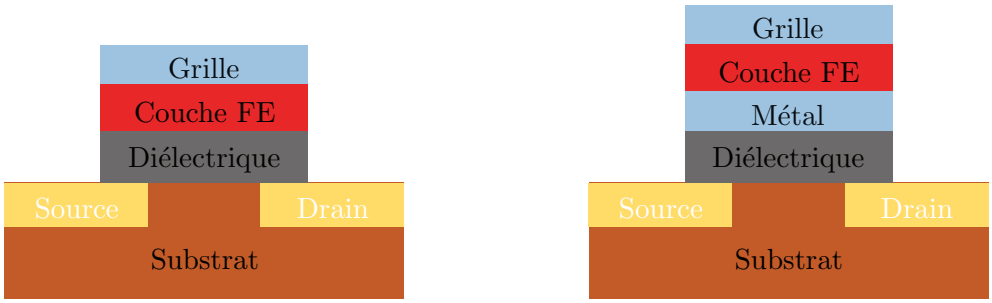


FIGURE 4.3 : Les deux types de FeFET utilisés avec (a) le transistor MFIS et (b) le transistor MFMIS

L'intérêt de ces FeFETs est qu'ils offrent une meilleure rétention des données stockées. De plus, ces FeFETs sont aussi intégrables dans une technologie CMOS ce qui apporte des bénéfices en unissant des paires symétriques de transistors de type n et de type p (par rapport à la charge du substrat). Les bénéfices apportés par ce couplage sont une plus grande résistance au bruit ainsi qu'une faible consommation d'énergie statique [127, 128]. En effet, seul un des transistors est passant, et le circuit ne présente un pic de consommation que lors de la transition de l'état passant à l'état bloquant. Cette réduction de la consommation d'énergie est la principale raison pour laquelle le CMOS est la technologie la plus utilisée dans les puces électroniques.

Pour ce modèle, trois portes, parmi les plus utilisées en électronique, sont caractérisées. Les schémas de ces portes sont indiqués dans la Figure 4.4. Les schémas montrent que les sources des pFeMFET et des pFET sont connectées au Vdd, tandis que les sources des nFeMFET et des nFET sont reliées à la masse. Les drains de tous les transistors sont connectés au même point, correspondant à la sortie de la porte logique non-volatile. Il est également notable que, dans les portes NAND et AND, un transistor MOSFET classique est présent. Ce transistor est issu des technologies actuelles, avec une taille optimisée, et il est donc plus mature que ceux des technologies émergentes. De plus, les transistors FeMFETs sont beaucoup plus gros (considérés comme des carrés de 500 nm de côté). Ainsi, les valeurs présentées dans le tableau sont principalement dues à la consommation et aux temps de propagation des transistors FeMFET. Enfin, pour deux transistors de même type (n ou p), il n'est pas nécessaire qu'ils soient passants simultanément, ces derniers sont donc programmés de manière complémentaire. Il est également impératif d'éviter que deux transistors complémentaires soient passants simultanément, afin de prévenir un

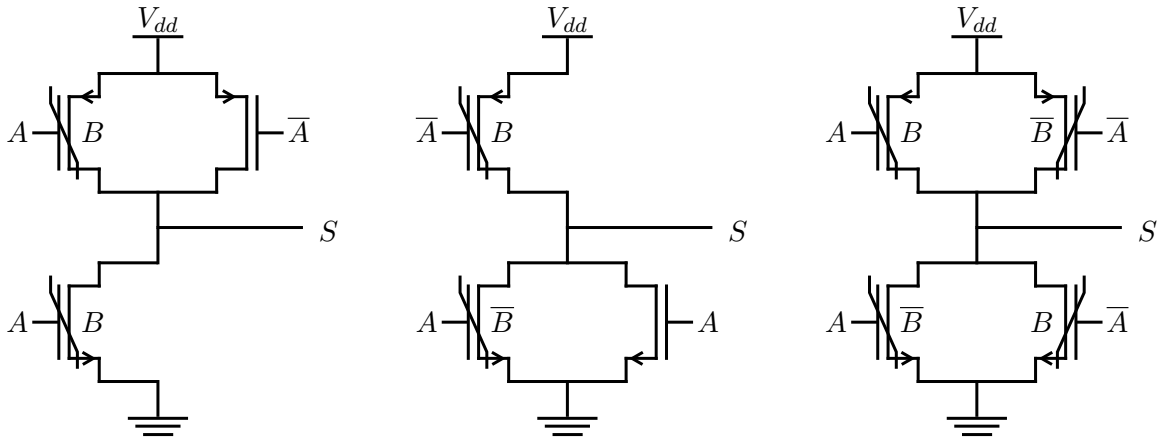


FIGURE 4.4 : Schéma des différentes portes FeMFET avec (a) un NAND, (b) un AND, et (c) un XNOR

Porte	Énergie d'écriture (J)	Énergie de lecture (J)	Puissance Statique (W)
NAND	$35,93 \times 10^{-3}\ pJ$	$1,288 \times 10^{-3}\ pJ$	$14,55 \times 10^{-3}\ pW$
AND	$37,783 \times 10^{-3}\ pJ$	$1,26 \times 10^{-3}\ pJ$	$66,44 \times 10^{-3}\ pW$
XNOR	$11,174\ pJ$	$3,157 \times 10^{-3}\ pJ$	$1,055\ pW$

TABLE 4.2 : Caractéristiques de consommation énergétique des portes FeMFET

court-circuit direct. Cette contrainte est assurée par les programmations complémentaires des transistors. Finalement, au moins un transistor doit être passant afin d'éviter une sortie en état haute impédance.

Pour chacune de ces portes, toutes les informations nécessaires concernant l'écriture et la lecture des données sont disponibles. Ces informations sont réparties en deux tableaux : le premier présente la consommation, tandis que le second donne les délais des cellules.

Le tableau 4.2 présente les caractéristiques énergétiques des portes logiques réalisées à partir de transistors FeMFET, notamment l'énergie nécessaire à l'écriture et à la lecture d'une donnée, ainsi que la puissance statique consommée. Les valeurs indiquées proviennent de simulations effectuées sur la base des paramètres technologiques des dispositifs considérés. Il apparaît que les portes AND et NAND présentent des consommations très proches, ce qui est cohérent avec la similarité de leur architecture interne. En effet, chacune d'elles intègre deux transistors FeMFET, l'un de types n et l'autre de type p, ainsi qu'un MOSFET. La porte XNOR, en revanche, affiche des consommations et des délais plus élevés. Cette différence s'explique par la présence de quatre transistors FeMFET, dont

Porte	Temps de transition (s)	Temps de propagation (s)
NAND	$27,73\ ps$	$268,63\ ps$
AND	$89,54\ ps$	$221,55\ ps$
XNOR	$869,36\ ps$	$928,98\ ps$

TABLE 4.3 : Temps de transition et de propagation des portes logiques FEMFET

les dimensions sont supérieures à celles des MOSFET utilisés. Ces derniers étant optimisés en taille afin de minimiser leur consommation. Il en résulte que la majeure partie de la consommation énergétique provient des transistors FeMFET, tandis que la contribution des MOSFET reste marginale.

Pour déterminer la consommation totale d'un circuit exploitant ces portes logiques, un suivi est effectué sur le nombre d'activation de chaque porte au cours de l'exécution d'un programme. Ce comptage permet, à partir des valeurs unitaires présentées dans le tableau, de calculer la consommation globale en lecture et en écriture. Les informations ainsi collectées sont transférées vers une station de calcul où les traitements nécessaires sont effectués. Cette méthode repose sur l'utilisation de données issues de simulations et s'inscrit dans le cadre d'un modèle de caractérisation énergétique, ce qui permet une adaptation aisée à d'autres technologies de transistors non-volatiles.

Les résultats présentés dans le tableau 4.3 reflètent une cohérence comparable à celle constatée dans l'analyse de la consommation. Il est mis évidence que les temps de transition et de propagation observés pour les portes AND et NAND sont très similaires, comme dans le cas de la consommation énergétique. Ce résultat est peu surprenant compte tenu de la forte similitude entre les architectures des deux portes. La porte XNOR présente également des résultats cohérents avec ceux observés pour la consommation énergétique, lesquels s'expliquent par le nombre plus élevé de transistors FeMFET nécessaires à sa réalisation.

OOn peut constater dans le tableau l'absence des temps d'écriture, qui constituent pourtant l'opération la plus coûteuse dans le cas de ces dispositifs FeMFET. Ceci s'explique par le fait que, pour ces simulations, l'objectif principal était de s'assurer du bon comportement fonctionnel des portes. Pour cela, un temps d'écriture de l'ordre de 1 μs a été utilisé afin de garantir la programmation désirée et d'éviter des erreurs d'états. Néanmoins, la littérature scientifique rapporte des délais nettement plus courts, on peut y voir qu'un temps moyen d'écriture dans un FeMFET est de l'ordre de 100 ns [129].

Dans le cadre de nos expériences, les paramètres d'intérêt sont donc les temps d'écriture ainsi que les temps de propagation au sein des portes. En ce qui concerne les temps de lecture, on constate qu'ils sont tous inférieurs à la période du cycle du processeur COMET cadencé à 100 MHz, utilisé dans ce travail. Ainsi, le temps de lecture des portes FeMFET sera dans tous les cas d'un cycle processeur. Dans le cas de l'écriture, deux scénarios

Porte	NAND	AND	XOR
Temps de Lecture (cycles)	1	1	1
Temps d'Ecriture (Test)	100	100	100
Temps d'Ecriture (Litérature)	10	10	10
Energie de Lecture (J)	$1,288 \times 10^{-3}\ pJ$	$1,26 \times 10^{-3}\ pJ$	$3,157 \times 10^{-3}\ pJ$
Energie d'Ecriture (J)	$35,93 \times 10^{-3}\ pJ$	$37,783 \times 10^{-3}\ pJ$	$11,174\ pJ$

Table 4.4 : Carte Modèle pour les FeMFETs

peuvent être considérés. Avec un temps d'écriture de 1 μs, l'opération nécessite 100 cycles processeur, tandis qu'avec la valeur de 100 ns rapportée dans la littérature, l'écriture requiert 10 cycles processeur.

Avec toutes ces informations, la mise en place de la carte modèle pour les FeMFET est telle que représentée dans le tableau Tab. 4.4. Il est intéressant de constater que, d'après la littérature et les résultats issus des tests des portes logiques, les valeurs retenues pour les temps d'écriture, en dehors des marges de sécurité utilisées pour garantir la programmation des transistors, ainsi que pour les temps de lecture sont cohérentes avec celles considérées lors du premier test utilisant des FeMFETs. L'intérêt de cette approche réside dans le fait qu'en plus des temps de calcul, les consommations associées à chaque porte sont également prises en compte. Cela permet d'obtenir une caractérisation plus complète du comportement des opérateurs dans le cadre de programmes fréquemment utilisés dans l'IoT, ainsi que lors de leur exécution sur un processeur.

Ces cartes modèles constituent l'élément central de la méthode proposée. En effet, elles permettent de modéliser le comportement de n'importe quel opérateur ou composant électronique. Cette flexibilité offre la possibilité soit d'évaluer une technologie donnée, soit d'analyser une même méthode en la confrontant à plusieurs technologies différentes. La section suivante présente l'opération étudiée ainsi que les résultats obtenus pour les deux technologies considérées.

4.3 Expériences et Résultats

Dans cette section sont présentés les expériences menées ainsi que les résultats obtenus. Dans un premier temps, l'opération de MixColumn est adaptée en définissant de nouvelles instructions RISC-V spécifiques et en utilisant comme modèle comportemental un circuit dédié à cette opération de multiplication [130]. La deuxième partie est consacrée à la mise en place pratique de ces instructions dans le cœur du processeur et à la façon dont elles sont appelées. Enfin, la troisième partie expose les résultats des expériences issues de cette intégration ainsi que leur analyse.

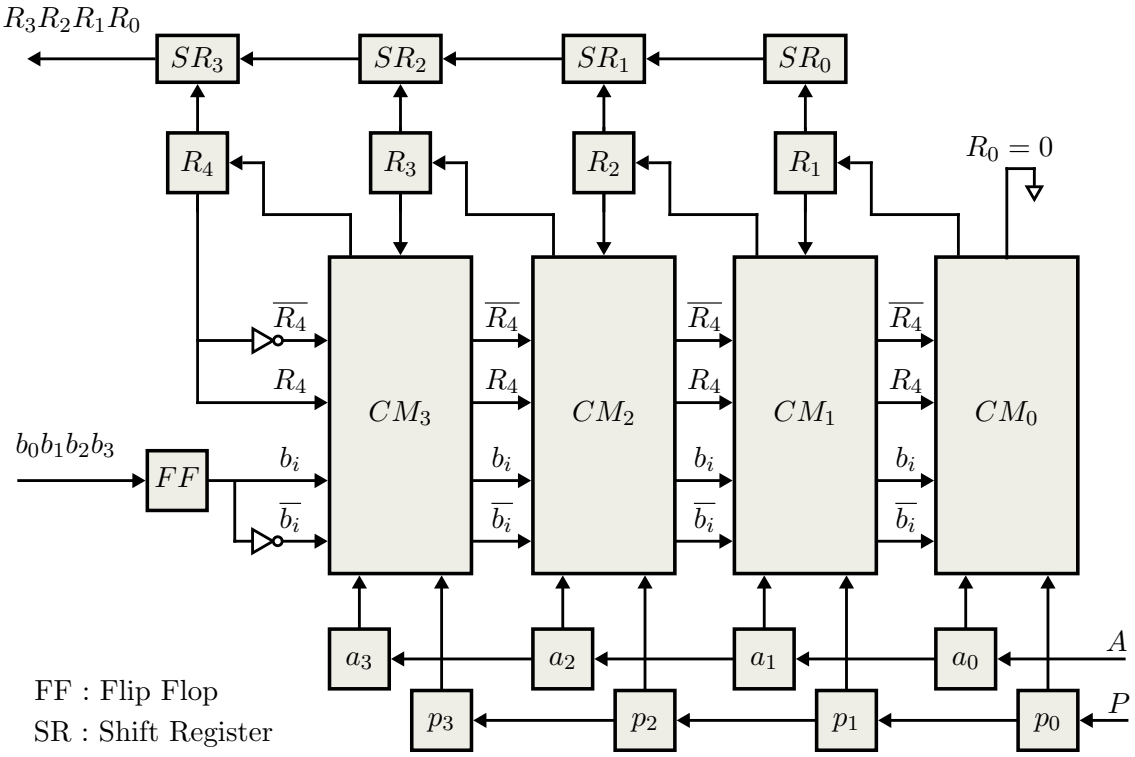


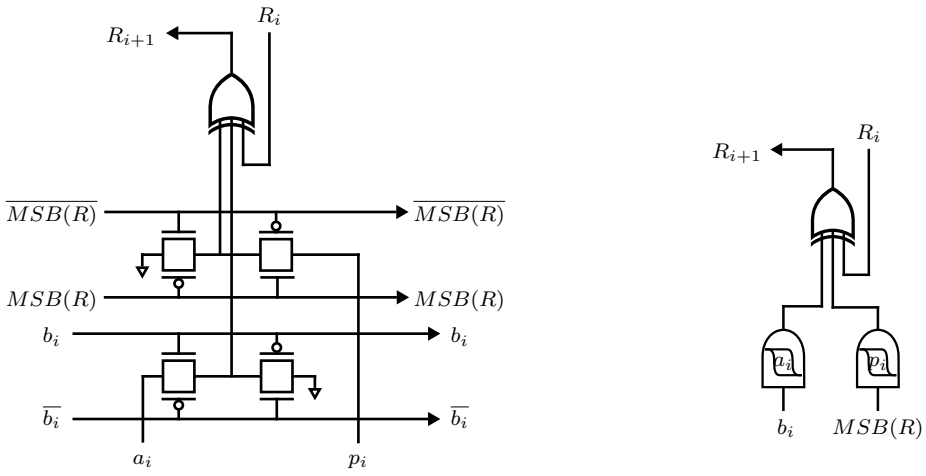
FIGURE 4.5 : Schéma du multiplicateur de Galois pour 4 bits sans NVM

4.3.1 La mise en place de l'opération de MixColumn

Pour les tests effectués, l'opération qui a été choisie est l'opération MixColumns de l'AES. Pour cette opération, chaque colonne du tableau d'état subit une transformation linéaire impliquant une multiplication matricielle sur un champ fini, à savoir un champ de Galois $GF(2^8)$. Cette opération mélange les octets dans chaque colonne, diffusant davantage les données et les rendant plus résistantes aux attaques. Du fait de la multiplication matricielle, les opérations arithmétiques dans les corps de Galois sont utilisées de manière intensive. Dans cette opération, chaque élément est réduit modulo un polynôme irréductible spécifique de l'AES : $P(x) = x^8+x^4+x^3+x+1$, noté 0x11B en représentation hexadécimale. PPour adapter cette opération et rendre son fonctionnement équivalent à celui d'un circuit matériel réel, il a fallu mettre en place un bloc dans le cœur de notre processeur, basé sur un circuit compatible avec les composants. Pour ce faire, un circuit déjà existant [130], dédié à la multiplication dans un corps de Galois fini, a été utilisé après avoir été modifié afin d'inclure des portes à base de NVM.

Le circuit représenté en Figure 4.5 correspond au circuit issu de l'article de référence pour une multiplication sur 4 bits. L'un des points remarquables est que ce circuit est

totalement extensible du fait de la mise en série d'éléments multiplicatifs CM_i traitant un bit, représentés en Figure 4.6a. Sur cet élément, il est possible de distinguer trois blocs. De bas en haut, il y a tout d'abord une opération AND entre un bit de chaque opérande, à savoir a_i et b_i, issus respectivement des opérandes A et B. Au-dessus se trouve un autre opérateur AND qui effectue une opération entre l'élément du polynôme irréductible p_i du polynôme P décrit précédemment, et le bit de poids fort du résultat de l'opération courante R, c'est-à-dire $MSB(R)$. Enfin, une opération XOR est réalisée entre l'élément courant du produit et les résultats des deux blocs précédents. Le fait que la multiplication pour un bit utilise le « produit courant » indique que l'opération s'effectue en plusieurs étapes proportionnelles à la longueur des éléments multiplicatifs. Il en résulte que la complexité de l'algorithme est de l'ordre de la longueur des éléments utilisés. Ainsi, pour huit bits, il suffit de rallonger le circuit de quatre éléments CM, ce qui implique que l'opération de multiplication nécessite huit cycles d'horloge complets pour être réalisée.



MSB : Most Significant Bit (Bit de Poids Fort)

(a) (b)

FIGURE 4.6 : Schéma de la cellule de multiplication dans un corps de Galois pour un bit, avec (a) le schéma issu de l'article et (b) la version adaptée aux technologies non-volatiles.

Dans le projet, l'objectif est cependant d'évaluer le fonctionnement de nouvelles mémoires non-volatiles. Il est donc nécessaire d'adapter l'ensemble de ce circuit aux nouveaux opérateurs. À cet effet, la cellule mono-bit est adaptée dans la version présentée en Figure 4.6b. Les portes AND utilisées précédemment sont remplacées par des versions FeFET développées dans le cadre du projet. L'impact sur le circuit est qu'il peut désormais stocker l'équivalent de deux constantes, ce qui permet d'éliminer certaines entrées logiques. De plus, du point de vue du fonctionnement, comme les deux portes non-volatiles sont en parallèle, l'écriture et la lecture peuvent être réalisées simultanément. Ainsi, lors de l'uti-

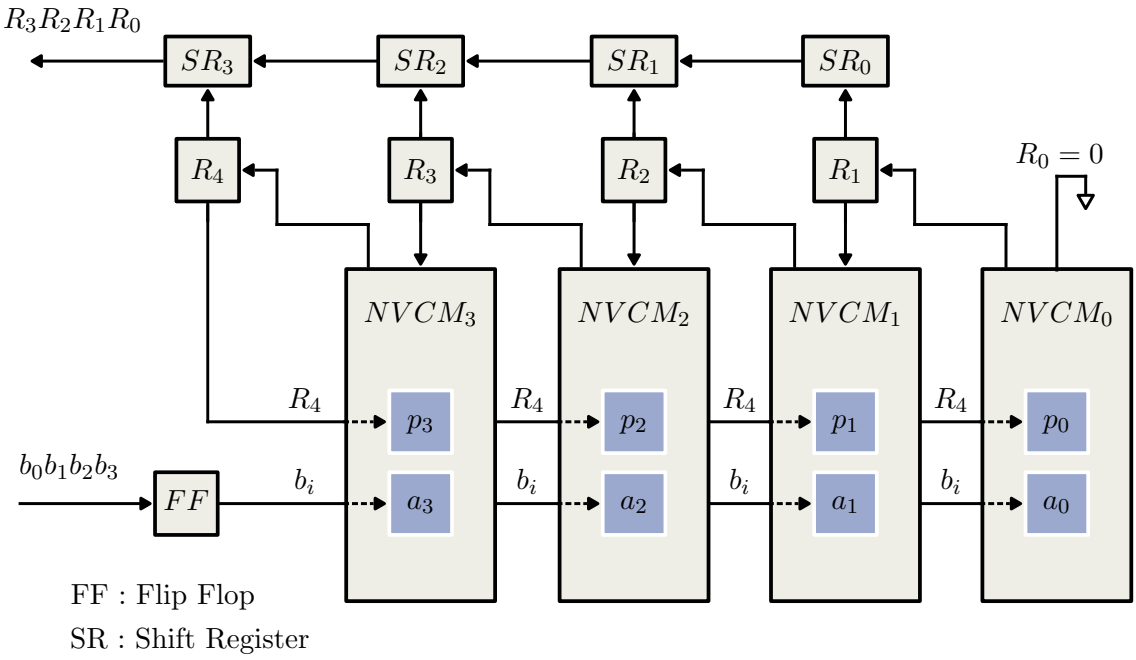


FIGURE 4.7 : Schéma du multiplicateur de Galois pour 4 bits à base de NVM

lisation de ces portes, il est possible de considérer que les deux portes AND en parallèle constituent le principal facteur influençant le temps de calcul.

La version finale du circuit multiplicatif non-volatile pour quatre bits est représenté en Figure 4.7. Ce circuit, issu du travail de l'équipe, constitue la brique de base permettant l'évaluation des opérateurs non-volatile. Comme ce circuit ne modifie pas profondément la structure existante, il reste nécessaire de huit cycles pour effectuer l'opération de multiplication de Galois. Il permet cependant d'améliorer la consommation énergétique du circuit, notamment en termes d'accès mémoire, puisque le besoin de charger le polynôme à chaque étape est supprimé, celui-ci pouvant être stocké jusqu'à un éventuel changement de configuration, et donc d'algorithme.

De plus, dans le cas présent de l'AES, afin d'être optimal, il n'est pas pertinent d'avoir deux portes AND à base de NVM dans les blocs de base du circuit. En effet, le fait de devoir écrire dans la porte contenant les deux opérateurs A et B ralentirait le bon fonctionnement du circuit, en raison du changement fréquent de la valeur A, qui nécessiterait un nombre plus élevé d'écritures avec les NVM, opération la plus coûteuse. Dans ce contexte, le circuit testé est une combinaison des deux architectures présentées précédemment, avec comme brique de base le circuit illustré en Figure 4.8.

Ainsi, pour revenir aux cartes modèle utilisées, il est possible de redéfinir la carte complète du circuit à partir des informations relatives au fonctionnement du circuit multiplicatif. Comme pour les valeurs associées aux FeFETs, la possibilité d'envisager un circuit

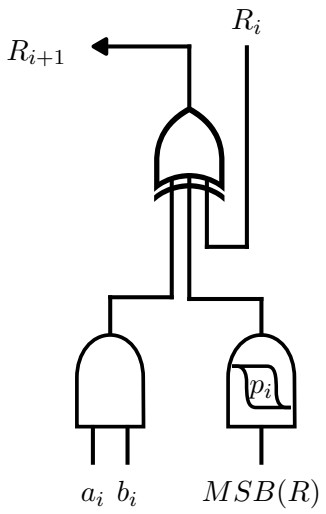


FIGURE 4.8 : Schéma de la cellule pour la multiplication de Galois pour un bit à base de NVM spécifique pour l'AES

multiplicatif effectuant la multiplication en parallèle, et non plus en série, réduisant ainsi le nombre total de cycles nécessaires à un, a été évaluée afin de pouvoir comparer les résultats finaux. Le tableau contenant l'ensemble des valeurs est donc présenté dans le Tableau 4.5. Il contient l'ensemble des valeurs temporelles obtenues lors des tests et prévues avec les différentes technologies ainsi que la consommation pour la porte AND qui est celle utilisée pour l'opération.

L'opérateur ainsi défini trouve principalement son intérêt dans la phase de déchiffrement. En effet, lors du chiffrement, les coefficients par lesquels les opérandes doivent être multipliés sont limités à des entiers compris entre 1 et 3. Dans ce contexte, la multiplication dans le corps de Galois devient moins pertinente, celle-ci pouvant être efficacement remplacée par un simple décalage à gauche, déjà pris en charge dans l'ISA RISC-V.

Afin de rendre cet opérateur exploitable au sein du processeur, il a été nécessaire d'introduire de nouvelles instructions RISC-V au-dessus du jeu RV32-I. Deux instructions principales ont ainsi été ajoutées à l'ensemble d'instructions :

- `LoadPoly` : une instruction configurant les neuf bits du polynôme irréductible sur notre plateforme, pour lequel il existe un registre dédié à cet effet au sein du processeur.

- `GaloisMul` : cette instruction est chargée d'exécuter la multiplication dans le corps de Galois entre deux entiers codés sur huit bits. La précision sur la taille des opérandes

	Valeurs obtenues (cycles)	Valeurs prévues (cycles)
Lecture (FeFET)	27	1
Ecriture (FeFET)	54	10
Lecture (FEMFET)	1	1
Ecriture (FEMFET)	100	10
Circuit de Multiplication	8	1

(a) (Temps de calcul)

Porte	Energie de Lecture (J)	Energie d'Ecriture (J)
AND	$1,26 \times 10^{-3}\ pJ$	$37,783 \times 10^{-3}\ pJ$

(b) Energie

TABLE 4.5 : Paramètres finaux pour l'opération de Mix Column

est essentielle, car elle conditionne directement le nombre de cycles requis pour achever l'opération.

Chacune de ces instructions est déclinée en deux versions : une version volatile et une version non-volatile reposant sur les FeFET. Dans le cas de l'instruction `LoadPoly`, la différence réside dans la prise en compte des transitions de chaque bit stocké dans les FeFET, nécessitant une surveillance systématique de ces changements d'état. Pour l'instruction `GaloisMul`, la distinction se situe au niveau de la durée de blocage du processeur COMET, qui doit être adaptée en fonction des caractéristiques de l'opérateur intégré. Par exemple, pour une multiplication dans le champ de Galois sur huit bits, le processeur doit être maintenu dans son état d'arrêt pendant huit cycles, tout en prenant en considération les délais supplémentaires liés aux opérations de lecture et d'écriture dans les transistors ferroélectriques.

L'intégration de ces nouvelles instructions a nécessité des modifications sur le coeur de processeur COMET utilisé dans le projet. Dans la prochaine partie, les modifications ainsi que le fonctionnement de la plateforme seront présentés.

4.3.2 Fonctionnement du simulateur

Afin d'utiliser pleinement l'environnement RISC-V, plusieurs étapes sont nécessaires en termes d'installation et de configuration. Une fois cela fait, il est nécessaire de comprendre le fonctionnement du processeur. Comme expliqué précédemment, le processeur COMET RISC-V est pipeliné à cinq étages. Le fonctionnement est présenté en Figure 4.9. Il est possible de noter sur cette figure les étapes surlignées en bleu, car ce sont dans ces étapes que les modifications seront effectuées.

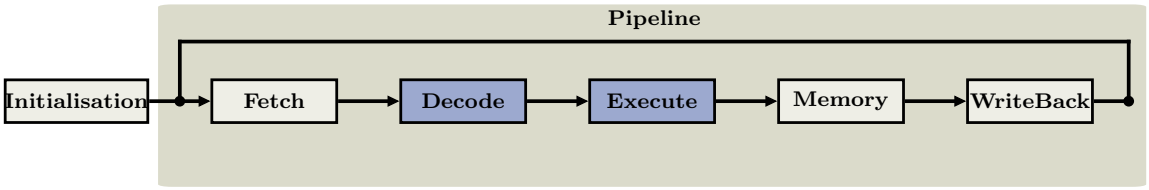


FIGURE 4.9 : Pipeline du processeur Comet, avec les étages modifiés en bleu

Afin d'utiliser ces instructions, il faut indiquer au processeur que de nouvelles options pour effectuer des calculs sont disponibles. Les modifications du cœur de COMET sont effectuées dans l'étape de décodage, chargée d'interpréter les instructions reçues. Il est nécessaire d'initialiser les constantes liées aux calculs à effectuer, lesquelles seront ensuite transmises et utilisées dans les autres étages du pipeline. Ces informations sont reçues par l'étape d'exécution, qui assure l'exécution finale des étapes de fonctionnement du cœur. C'est dans cette partie qu'est implémentée la mise en attente du cœur, une fois que les constantes de temps d'écriture et de lecture ont été prises en compte.

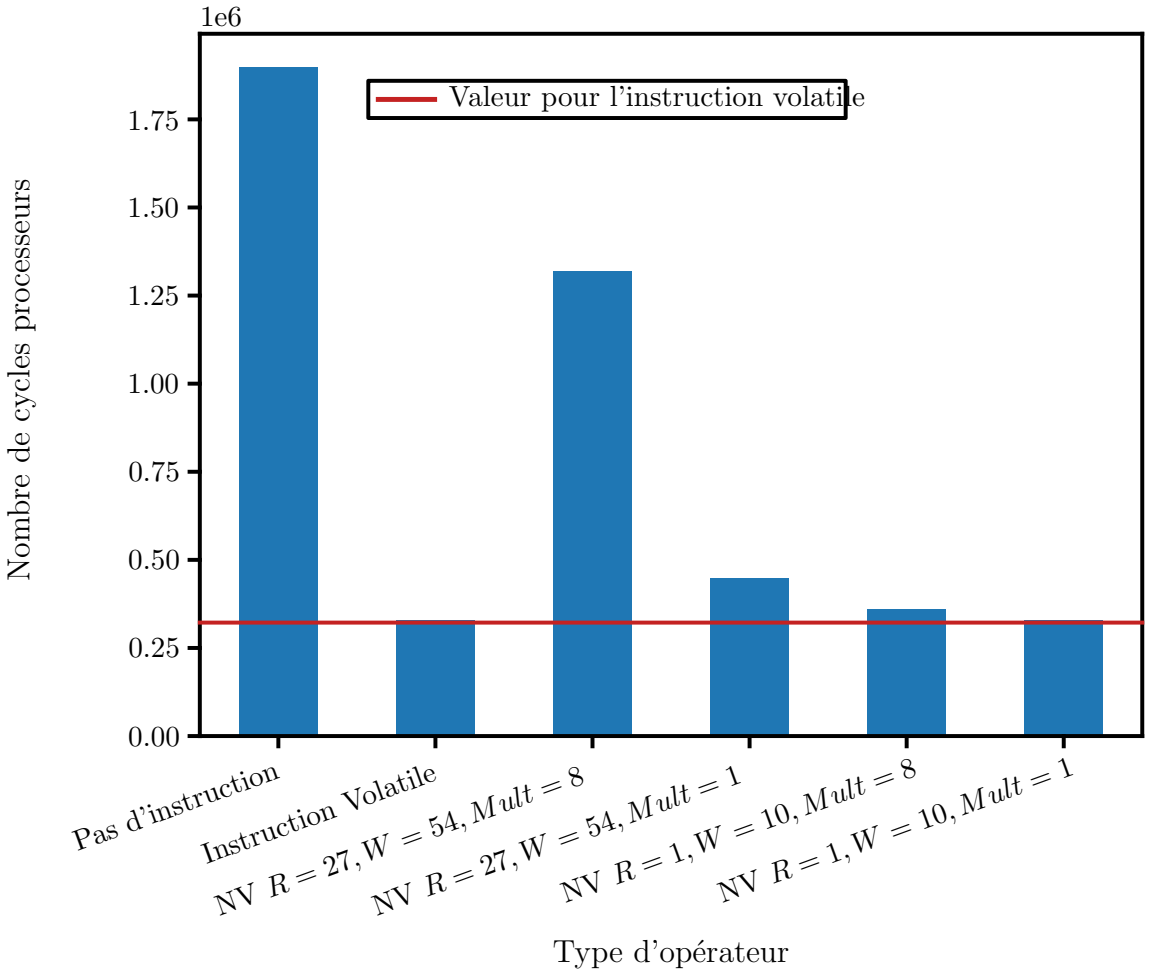
Une fois le cœur mis en place, il faut définir le code de test pour l'expérience. Dans le cas présent, ce dernier est entièrement écrit en C, afin d'être cohérent avec le code utilisé par le processeur COMET. Cependant, il est nécessaire d'indiquer explicitement au processeur qu'il doit utiliser les nouvelles instructions. Pour ce faire, deux solutions ont été explorées. Tout d'abord, la modification du compilateur a été envisagée afin d'obtenir une insertion générale des instructions de manière automatique. La deuxième solution a consisté à utiliser de « l'inline assembly », qui consiste à écrire directement les commandes sous forme assembleur dans le code de test, en appelant des fonctions déjà définies en C. Cette méthode ne permet pas une flexibilité aussi grande que de modifier complètement le compilateur. Cependant, si les positions exactes d'utilisation des nouvelles instructions sont connues, elle garantit que les commandes seront utilisées à l'endroit souhaité, ce qui est le cas dans le projet.

Une fois que cela est fait, il est alors nécessaire de traduire l'ensemble du code en instructions compréhensibles par COMET, puis de lancer le code assembleur via la version logicielle du processeur.

4.3.3 Présentation des résultats

Après avoir effectué les tests avec les combinaisons de paramètres présentées, il est désormais possible d'analyser les résultats obtenus.

La figure 4.10 présente les résultats pour l'ensemble des cas énumérés dans le tableau 4.5a, à l'exception du cas correspondant à un temps d'écriture de 100 cycles, ce dernier étant non compétitif par rapport aux autres. De plus, concernant les différents modèles de FeFET et de FeMFET, les temps d'écriture mesurés expérimentalement et ceux issus de la littérature ont été regroupés.



	Sans intruction	Instruction Volatile	NV R=27 W=54 Mult=8	NV R=27 W=54 Mult=1	NV R=1 W=10 Mult=8	NV R=1 W=10 Mult=1
Nombre de Cycles	1 898 032	327 573	1 318 339	447 427	359 831	327 575

FIGURE 4.10 : Graphique du nombre de cycles du cœur nécessaires au déchiffrement, pour différents types de paramètres utilisés dans l'opérateur non-volatile, où W correspond aux valeurs d'écriture et R aux valeurs de lecture

Tout d'abord, la figure met en évidence qu'un simple ajout d'une instruction dédiée à la multiplication dans le corps de Galois entraîne une réduction significative de la charge de travail du processeur, diminuant le nombre d'instructions nécessaires pour un déchiffrement d'un facteur 5,7×. Ce premier résultat est particulièrement intéressant, puisqu'il montre qu'une unique opération spécialisée peut profondément améliorer l'efficacité du processeur, tant en termes de temps d'exécution que de consommation énergétique. Ensuite, concernant la comparaison entre opérateurs volatiles et non-volatiles, il apparaît que, pour une seule séquence de déchiffrement, la version non-volatile demeure moins performante que son équivalent basé sur des transistors CMOS. Cette différence s'explique par le fait que les technologies non-volatiles n'ont pas encore atteint le même degré de maturité que les solutions CMOS traditionnelles, comme discuté dans les sections précédentes. Néanmoins, il est remarquable de constater que, dans le cas du circuit NVM optimisé, c'est-à-dire lorsque l'écriture nécessite 10 cycles, la lecture nécessite un seul cycle, et que le multiplicateur est exécuté en parallèle plutôt qu'en série, l'écart avec la version volatile n'est plus que de deux cycles processeur seulement.

Cependant, outre le fait que le circuit sera déjà compétitif dès un tour de déchiffrement, le principal intérêt de la technologie provient de sa non-volatilité. À savoir qu'il n'est plus nécessaire de charger le polynôme spécifique à l'AES dans le cas présent. Le Tableau 4.6 présente l'impact du nombre de déchiffrements successifs sans avoir besoin de recharger la valeur du polynôme. Le tableau est structuré de telle sorte que, d'un côté, dans la deuxième colonne, figure le nombre total d'instructions effectuées par le processeur lors des déchiffrements avec l'opérateur simple. De l'autre côté, lors de l'utilisation des opérateurs non-volatiles, ce qui est affiché n'est pas l'ensemble des cycles effectués mais une comparaison par rapport au circuit CMOS. Par exemple, il est remarquable que, dans le cas de l'utilisation d'un opérateur basé sur les FeFETs nécessitant R=27 cycles pour une lecture et W=54 cycles pour une écriture, ainsi qu'un circuit multiplicatif fonctionnant en parallèle, le surcoût en cycles processeur pour un déchiffrement est de 119 854 cycles supplémentaires.

Un résultat particulièrement intéressant est que, dès deux déchiffrements consécutifs, le circuit non-volatile optimisé parvient à surpasser l'implémentation reposant sur les instructions classiques de multiplication dans le corps de Galois. Cet avantage s'accentue progressivement à mesure que le nombre de déchiffrements augmente. Ces observations mettent en évidence que la technologie non-volatile, au-delà de l'élimination des rechargements répétés, permet également de réduire la charge de calcul globale subie par le processeur. L'amélioration en termes de performance demeure toutefois relativement modeste dans ce cas précis, en raison de la faible quantité d'informations stockées, soit seulement 8 bits au total. Néanmoins, ces premiers résultats sont prometteurs et ouvrent la voie à des perspectives d'optimisation, notamment par une exploitation de circuits plus complexes ou par l'augmentation du volume de données stockées.

Dans le Tableau 4.7, le pourcentage de surcoût en instructions par rapport à l'instruction simple est représenté pour chaque ordre de grandeur. Il apparaît que le pourcentage de surcoût ne diminue pas dans la plupart des cas. En particulier, dans le cadre d'un

Nb Decrypt	Simple Instruction (raw value)	NV R=27 W=54 Mult=8	NV R=27 W=54 Mult=1	NV R=1 W=10 Mult=8	NV R=1 W=10 Mult=1
1	327 573	990 766	119 854	32 258	2
3	654 803	1 981 472	239 648	64 500	-12
5	982 033	2 972 178	35 9442	96 742	-26
10	1 800 108	5 448 943	658 927	177 347	-61
30	5 072 408	15 356 003	1 856 867	499 767	-201
50	8 344 708	25 263 063	3 054 807	822 187	-341
100	16 525 458	50 030 713	6 049 657	1 628 237	-691
300	49 248 458	149 101 313	18 029 057	4 852 437	-2 091
500	81 971 458	248 171 913	30 008 457	8 076 637	-3 491
1000	163 778 958	495 848 413	59 956 957	16 137 137	-6 991

TABLE 4.6 : Différence en termes de cycles consommés par l'instruction simple et les différentes versions de l'opérateur non-volatile

Nombre de déchiffrement	Simple Instruction (raw value)	NV R=27 W=54 Mult=8 (en %)	NV R=27 W=54 Mult=1 (en %)	NV R=1 W=10 Mult=8 (en %)	NV R=1 W=10 Mult=1 (en %)
1	327 573	302.5%	36.59%	9.848%	0.0006%
10	1 800 108	302.7%	36.60%	9.852%	-0.0034%
100	16 525 458	302.7%	36.61%	9.853%	-0.0042%
1000	163 778 959	302.8%	36.61%	9.853%	-0.0043%

TABLE 4.7 : Excédent, exprimé en pourcentage, pour chaque version de l'instruction basée sur la NVM par rapport à la version de base.

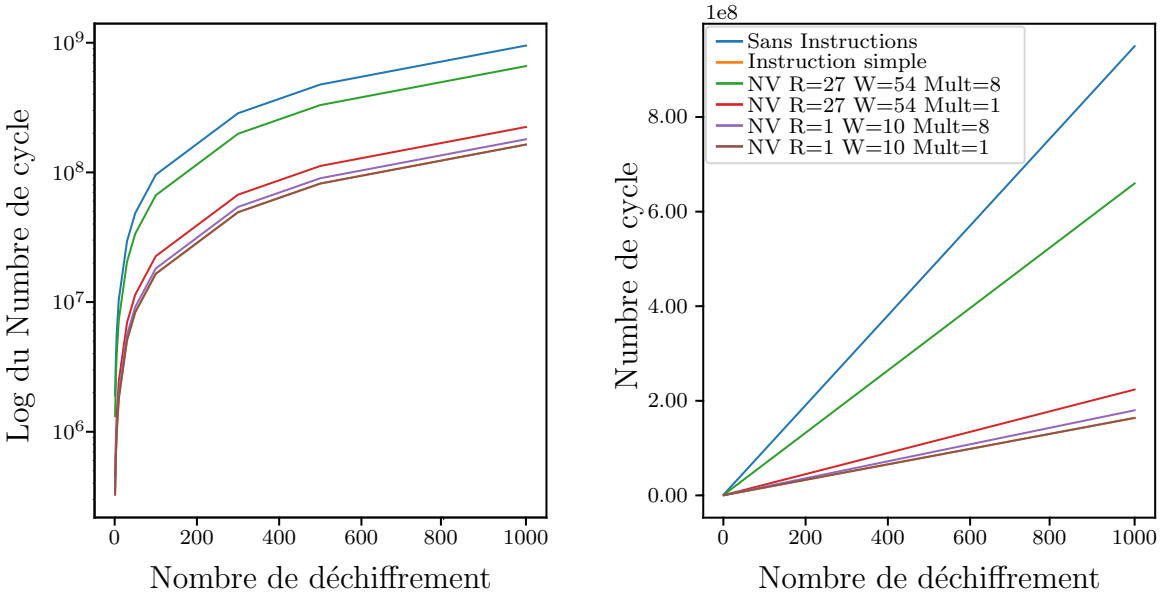


FIGURE 4.11 : Graphique représentant le nombre de cycles du cœur du processeur nécessaires pour l'opération de déchiffrement en fonction du nombre de déchiffrements. (a) présente les résultats sur une échelle logarithmique et (b) présente les résultats en valeurs absolues.

circuit utilisant les valeurs actuelles du FeFET, ce qui est essentiellement dû à la lecture nécessitant 27 cycles, l'écart en pourcentage augmente. Ensuite, dans le cas de l'utilisation d'un multiplicateur en série, nécessitant 8 cycles pour effectuer une multiplication dans le corps de Galois, l'écart augmente également. En effet, même si le polynôme doit être rechargé à chaque nouveau déchiffrement, le besoin de 8 cycles par multiplication représente un coût trop élevé pour être compétitif avec l'instruction volatile. Il ressort du tableau que, si le circuit est entièrement optimisé avec un fonctionnement en parallèle, la version volatile finira par dépasser le nombre d'instructions utilisées par le circuit contenant des opérateurs non-volatiles.

Lorsque le nombre de cycles est tracé en fonction du nombre de déchiffrements, comme illustré dans la Figure 4.11, il apparaît que le comportement du circuit est linéaire. Cette tendance s'explique naturellement par le fait que l'augmentation observée provient de la répétition systématique de la même opération plusieurs fois. Dès lors, il devient possible de déterminer avec précision le coût en cycles d'un déchiffrement isolé, en se basant uniquement sur les opérations elles-mêmes, indépendamment de la couche de contrôle apportée par le processeur.

Pour déterminer ces constantes, une soustraction élément par élément entre deux lignes du tableau de valeurs non transformées a été effectuée, suivie d'une normalisation par le

	Sans intruction	Instruction Volatile	NV R=27 W=54 Mult=8	NV R=27 W=54 Mult=1	NV R=1 W=10 Mult=8	NV R=1 W=10 Mult=1
Nombre de Cycles	948 840	163 615	658 968	223 512	179 736	163 608

TABLE 4.8 : Nombre de cycles par opération de déchiffrement pour chaque configuration des paramètres testés

	Sans intruction	Instruction Volatile	NV R=27 W=54 Mult=8	NV R=27 W=54 Mult=1	NV R=1 W=10 Mult=8	NV R=1 W=10 Mult=1
Nombre de Cycles	949 192	163 958	65 371	223 915	180 095	163 967

TABLE 4.9 : Nombre de cycles par déchiffrement pour chaque configuration des paramètres

nombre de déchiffrements. Les résultats obtenus pour chaque configuration sont présentés dans le Tableau 4.8. Ces chiffres confirment les observations précédentes : ils mettent en évidence que la seule configuration susceptible de surpasser l'instruction volatile correspond au cas d'une instruction non-volatile entièrement optimisée.

Un autre enseignement tiré de ces résultats concerne l'évaluation du nombre de cycles consommés par le processeur pour l'exécution de l'ensemble du code autour de la phase de déchiffrement. Ces valeurs sont présentées dans le Tableau 4.9. Un résultat particulièrement notable apparaît ici : l'écart entre l'instruction volatile et la version optimisée basée sur la NVM n'est que de 9 cycles processeur. Cette observation s'explique de manière cohérente par le fait que, mis à part l'opération d'écriture, les deux configurations sont équivalentes. En effet, dans le cas non-volatile, cette écriture requiert 10 cycles, ce qui justifie pleinement le léger surcoût observé.

En ce qui concerne la consommation énergétique, l'utilisation du système de comptage présenté précédemment permet d'obtenir les résultats présentés dans le Tableau 4.10. À noter que les valeurs liées aux écritures n'y figurent pas, puisque, dans tous les cas, le polynôme étant stocké de manière non-volatile, huit écritures initiales sont nécessaires au démarrage de l'expérience, correspondant à l'insertion des huit bits de données. L'analyse se concentre donc sur les opérations de lecture dans les portes à base de FeFET. Deux configurations sont considérées. Dans le premier cas, lorsque la multiplication est réalisée en série, chaque multiplication nécessite 8 cycles, impliquant à chaque cycle un parcours complet de huit portes AND. Cela conduit à un nombre particulièrement élevé de lectures pour chaque opération. Dans le second cas, correspondant à une implémentation parallèle, et en

Nombre de déchiffrement	Lecture Circuit Parallèle	Lecture Circuit Série
0	0	0
1	18 432	147 456
2	36 864	294 912
3	55 296	442 368
4	73 728	589 824
5	92 160	737 280
10	184 320	1 474 560

TABLE 4.10 : Nombre de lecture des portes NVM AND lors des expériences

supposant que l'architecture globale du circuit reste similaire, le nombre total de lectures est réduit d'un facteur de 8, ce qui améliore significativement l'efficacité de l'opération.

Il est également possible d'estimer la consommation énergétique à partir des valeurs présentées précédemment. Concernant les écritures avec les opérateurs non-volatiles, dans le cadre de l'AES, l'énergie consommée peut être évaluée à $3{,}023 \times 10^{-1}$ pJ. Le Tableau 4.11 synthétise quant à lui les résultats obtenus pour les lectures. Comme pour les mesures de performances en nombre de cycles processeur, les valeurs obtenues suivent une tendance fortement linéaire, ce qui s'explique par la méthodologie employée : le test repose sur la répétition systématique des mêmes calculs plusieurs fois. Il est toutefois notable que, pour un déchiffrement complet, la consommation mesurée atteint 23,224 pJ.

D'un point de vue strictement énergétique, ce résultat demeure très élevé en comparaison de ce qui serait attendu avec des transistors CMOS en technologie 16 nm, dont la consommation se situe typiquement à l'échelle de l'attojoule [131]. Cette différence importante peut néanmoins être expliquée par deux facteurs principaux : d'une part, le manque de maturité des technologies non-volatiles utilisées, et d'autre part, la taille des FeFETs expérimentaux testés (500 nm), qui influence directement la consommation énergétique observée.

Ces premiers résultats mettent clairement en évidence la pertinence de la méthodologie proposée. En effet, celle-ci a permis d'obtenir une première évaluation réaliste de l'impact potentiel de l'intégration d'opérateurs NVM au sein du cœur du processeur. Le choix du cas d'étude, fondé sur l'algorithme AES, apparaît d'autant plus pertinent que le polynôme associé est utilisé depuis plus de vingt ans dans le domaine de la cryptographie, conférant ainsi à l'analyse un cadre solide et représentatif. Toutefois, cette configuration expérimentale présente également une limite importante : la quantité de données rendues non-volatiles reste extrêmement réduite, puisqu'elle se limite à seulement huit bits de données.

Nombre de déchiffrement	Énergie Circuit Parallèle (pj)	Énergie Circuit Série (pj)
0	0	0
1	23	185
2	46	371
3	69	557
4	92	743
5	116	9283
10	232	1857

TABLE 4.11 : Valeurs estimées en picojoules pour les différents nombres de déchiffrements effectués

4.4 Conclusion

Ainsi, les résultats préliminaires obtenus mettent en évidence l'intérêt concret de l'intégration d'éléments issus de technologies non-volatiles. Tout d'abord, l'ajout d'une instruction dédiée à la multiplication de Galois a permis de réduire d'un facteur 6 le nombre de cycles nécessaires pour effectuer un déchiffrement. Par ailleurs, lorsque le circuit exploite un fonctionnement parallèle avec des FeMFET, en considérant un cycle de lecture et dix cycles d'écriture, le Tableau 4.7 montre que le nombre d'instructions supplémentaires induites par les caractéristiques de ces opérateurs décroît à mesure que le nombre de déchiffrements augmente. À partir de deux déchiffrements, les résultats indiquent même une amélioration par rapport à la version basée sur une instruction unique. Certes, ces gains demeurent limités en raison du faible volume de données concerné, soit un polynôme de huit bits. Néanmoins, il est intéressant d'envisager le potentiel de ces opérateurs dans des applications beaucoup plus gourmandes en mémoire, telles que le stockage et la manipulation des poids dans les réseaux neuronaux.

L'objectif principal de ces expériences résidait toutefois moins dans la recherche d'un gain immédiat de performance que dans la validation de la méthodologie proposée pour l'évaluation des technologies non-volatiles. À cet égard, la mise en place d'une carte modèle associée à l'intégration de nouvelles instructions dédiées dans un processeur RISC-V a démontré son efficacité. Cette approche offre en effet une évaluation pertinente et rapide de l'impact de ces nouvelles NVM, y compris lorsqu'elles sont comparées aux technologies volatiles CMOS traditionnelles CMOS, déjà largement optimisées.

Chapitre 5

Conclusions et Perspectives

5.1 Résumé de la contribution Technique 88
5.2 Perspectives 89
5.3 La dissémination du travail 90

Dans un premier temps, il convient de rappeler que l'objectif de la thèse était de trouver une méthode permettant de tester de nouveaux opérateurs à base de NVM dans le contexte de l'exécution sur un processeur, tout en ayant un retour réaliste de l'impact de ces technologies sur le fonctionnement du système. Ce chapitre présente les contributions techniques dans une première section ; dans une deuxième section, les perspectives et pistes d'amélioration sont présentées. Enfin, dans une troisième section, un résumé des contributions et de la dissémination scientifique est présenté.

5.1 Résumé de la contribution Technique

Dans ce manuscrit, ce travail de thèse présente plusieurs développements originaux portant sur les problématiques de test de nouvelles technologies dans le contexte de leur utilisation au sein d'un processeur. La première contribution concerne l'élaboration d'une méthode d'évaluation des opérateurs non-volatiles. Dans un premier temps, un modèle initial, reposant sur des approximations de courbes telles que la tangente hyperbolique, a été proposé. Toutefois, il a été démontré que l'émulation précise des courbes d'hystérésis nécessitait environ 50 DSPs par courbe, soit un total de 200 DSPs par FeFET pour générer l'ensemble des courbes de tension et d'intensité, montantes et descendantes. Ce coût en ressources, particulièrement élevé, s'est révélé prohibitif, même pour un FPGA de grande capacité tel qu'un FPGA de la famille Virtex-7.

Il a donc été nécessaire de proposer une approche alternative permettant d'émuler un plus grand nombre de transistors ferroélectriques. La méthode retenue repose sur l'utilisation d'une carte modèle représentant les caractéristiques essentielles d'un opérateur non-volatile ou d'une porte logique. Cette abstraction est construite à partir de paramètres clés tels que le temps de propagation dans la porte et la consommation énergétique globale du circuit. Elle offre ainsi un compromis entre simplicité et fidélité : suffisamment légère pour être exploitée dans des contextes plus larges, tout en conservant le niveau de précision requis pour analyser de manière pertinente l'impact de ces opérateurs au sein d'un système complet.

À l'aide des cartes modèles développées, l'utilisation du processeur RISC-V COMET a permis de concevoir une plateforme d'émulation spécifiquement dédiée à l'évaluation des opérateurs non-volatiles. Cette plateforme offre une grande flexibilité en permettant de tester différentes technologies émergentes directement dans le contexte d'exécution d'un processeur. L'implémentation dans le cœur RV32-I, dépourvu de mémoire cache, s'effectue par l'ajout de nouvelles instructions dédiées aux opérations exploitant les NVM. Contrairement aux approches purement logicielles ou aux estimations isolées, cette méthodologie replace l'opérateur étudié au sein du flot d'instructions réel d'un cœur RISC-V, ce qui autorise une évaluation précise de son impact sur les performances globales du système. Par ailleurs, cette approche se distingue par sa rapidité : elle permet d'obtenir des résultats dans des délais bien plus courts que la simulation classique, dont la complexité peut nécessiter

jusqu'à huit heures pour reproduire une seule seconde d'exécution dans le contexte d'un processeur [25].

Enfin, une troisième contribution de cette thèse réside dans l'évaluation de l'impact d'opérateurs non-volatiles à base de FeFET dans le cadre d'une application concrète, à savoir l'algorithme AES. L'étude a tout d'abord montré que l'introduction d'instructions dédiées à la multiplication dans les corps de Galois permettait une réduction significative du nombre de cycles, en diminuant ce dernier d'un facteur d'environ six par rapport au jeu d'instructions standard RV32-I utilisé par le cœur COMET.

Dans un second temps, une comparaison a été menée entre l'ajout d'une instruction dédiée classique et l'intégration d'une instruction reposant sur des opérateurs non-volatiles issus de technologies émergentes. Les résultats indiquent qu'à l'heure actuelle, ces dernières nécessitent un nombre de cycles plus élevé que leur équivalent basé sur des technologies CMOS pour effectuer une opération de déchiffrement. Toutefois, l'analyse dans un scénario prenant en compte des extinctions de la plateforme a mis en évidence une évolution intéressante : le surcoût initial en cycles diminue lorsque le circuit de multiplication dans les corps de Galois est optimisé pour fonctionner en parallèle plutôt qu'en série, avec une amélioration d'autant plus marquée que le nombre de déchiffrements augmente.

Enfin, dans une configuration combinant un multiplicateur parallèle optimisé et des portes logiques non-volatiles caractérisées par des paramètres proches des projections théoriques issues de la simulation, il a été observé que la version non-volatile devient plus performante que la version CMOS dès trois déchiffrements, en nécessitant cinq cycles de moins que l'implémentation reposant uniquement sur des instructions dédiées classiques. Ces résultats constituent ainsi une preuve de concept claire du potentiel des opérateurs à base de FeFET dans un contexte cryptographique. Ils soulignent en particulier que le pourcentage de cycles supplémentaires diminue progressivement grâce à la non-volatilité. Néanmoins, il convient de relativiser ces résultats : dans les expériences menées, les cellules utilisées étaient programmées et lues de manière analogue, avec pour seule différence les tensions appliquées. Dans le cadre d'un circuit de grande échelle intégrant une mémoire non-volatile, il serait indispensable d'introduire une surcouche de gestion de la programmation, ce qui pourrait influer sur les performances globales.

5.2 Perspectives

Ce travail ouvre plusieurs perspectives de recherche et de développement :

1. Une première piste est l'évaluation d'autres technologies non-volatiles compatibles CMOS en développant des cartes modèles à utiliser dans la plateforme de test, afin de comparer leurs performances et leur adéquation à différents scénarios d'utilisation. Cette diversification permettrait de positionner la méthodologie utilisée au cours de la thèse comme un outil générique d'évaluation des mémoires émergentes.

2. Une deuxième perspective concerne l'étude d'applications plus intensives en mémoire que le chiffrement, notamment les algorithmes d'intelligence artificielle embarquée. Ces applications, particulièrement gourmandes en échanges mémoire-processeur, constituent un terrain privilégié pour exploiter les avantages de la non-volatilité, par exemple en modélisant un ensemble de poids pour un réseau neuronal à l'aide de ces nouvelles mémoires FeFET, ou, comme indiqué précédemment, à l'aide d'autres technologies de mémoire émergentes, grâce à la simplicité de la mise au point d'une carte modèle.

3. Enfin, une troisième piste d'exploration est l'intégration de la mémoire non-volatile sous d'autres formes architecturales : directement connectée à la mémoire principale de la plateforme, ou encore en tant qu'accélérateur externe relié via les ports de sortie du processeur. Chacune de ces méthodes d'implantation pourrait offrir des compromis différents en termes de performance et de consommation.

4. La plateforme d'émulation sur FPGA pourrait être étendue à l'aide des ports externes pour fonctionner directement au sein de circuits de plus grande envergure. Une telle extension permettrait de renforcer le réalisme de l'évaluation, de réduire l'écart entre l'émulation et les futures implémentations matérielles, et d'ouvrir la voie à des études à plus grande échelle, comme par exemple le test de réseaux de capteurs ou, plus simplement, la mise en œuvre d'un échange réel d'informations entre deux FPGA échangeant des données chiffrées par AES.

5.3 La dissémination du travail

Cette section présente l'ensemble de la dissémination effectuée au cours de la thèse, selon l'ordre chronologique de publication pour les articles et de présentation pour les posters.

1. La première dissémination des recherches fut la présentation d'un poster présentant l'élaboration de la méthodologie et les choix effectués lors de l'école d'hiver FETCH en juin 2022.

2. Un second poster a été présenté lors du colloque annuel du Groupement de recherche SOC² à Strasbourg, sur les systèmes embarqués et les objets connectés, en juin 2022.

3. Un poster a été présenté l'année suivante lors du colloque du GDR SOC² à Lyon, présentant les premiers résultats de la plateforme en juin 2023.

4. Un article de conférence a été publié, en tant que second auteur, intitulé « FeFET based Logic-in-Memory design methodologies, tools and open challenges », en novembre 2023, à la suite de la conférence VLSI-SoC 2023 [132].

5. Un article de conférence a été publié, en tant que premier auteur, lors de la conférence Nanoarch 2023 à Dresde, en Allemagne, présentant les résultats de la plateforme d'émulation d'opérateurs non-volatiles au cours d'une présentation en décembre 2023, ainsi que dans un article en ligne disponible en janvier 2024 [133].

Cela représente un total de trois conférences au cours desquelles les avancées et résultats ont été présentés sous forme de posters, ainsi que deux articles de conférence.

Bibliographie

[1] *Facebook*. URL : https://www.facebook.com/chickentheginger (visité le 12/01/2026).

[2] Danijela Efnusheva, Ana Cholakoska et Aristotel Tentov. « A Survey of Different Approaches for Overcoming the Processor-Memory Bottleneck ». en. In : *AIRCC's International Journal of Computer Science and Information Technology* (avr. 2017), p. 151-163. ISSN : 0975-3826. URL : https://ischolar.sscldl.in/index.php/IJCSIT/article/view/148221 (visité le 23/04/2024).

[3] Shekhar Borkar et Andrew A. Chien. « The future of microprocessors ». In : *Commun. ACM* 54.5 (mai 2011), p. 67-77. ISSN : 0001-0782. DOI : 10.1145/1941487.1941507. URL : https://doi.org/10.1145/1941487.1941507 (visité le 13/03/2023).

[4] *IoT connected devices worldwide 2019-2030*. en. URL : https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (visité le 16/04/2024).

[5] Cédric Marchand et al. « FeFET based Logic-in-Memory : an overview ». In : *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. Juin 2021, p. 1-6. DOI : 10.1109/DTIS53253.2021.9505078. URL : https://ieeexplore.ieee.org/abstract/document/9505078 (visité le 16/04/2024).

[6] Michaël Mahamat, Ghada Jaber et Abdelmadjid Bouabdallah. « Achieving efficient energy-aware security in IoT networks : a survey of recent solutions and research challenges ». en. In : *Wireless Netw* 29.2 (fév. 2023), p. 787-808. ISSN : 1572-8196. DOI : 10.1007/s11276-022-03170-y. URL : https://doi.org/10.1007/s11276-022-03170-y (visité le 09/07/2024).

[7] An Chen. « A review of emerging non-volatile memory (NVM) technologies and applications ». en. In : *Solid-State Electronics*. Extended papers selected from ESSDERC 2015 125 (nov. 2016), p. 25-38. ISSN : 0038-1101. DOI : 10.1016/j.sse.2016.07.006. URL : https://www.sciencedirect.com/science/article/pii/S0038110116300867 (visité le 21/10/2022).

[8] Somayya Madakam, R. Ramaswamy et Siddharth Tripathi. « Internet of Things (IoT) : A Literature Review ». en. In : *Journal of Computer and Communications* 3.5 (mai 2015). Number : 5 Publisher : Scientific Research Publishing, p. 164-173. DOI : 10.4236/jcc.2015.35021. URL : https://www.scirp.org/journal/paperinformation.aspx?paperid=56616 (visité le 17/04/2024).

[9] Jangsoo Lee et al. « A Smart IoT Device for Detecting and Responding to Earthquakes ». en. In : *Electronics* 8.12 (déc. 2019). Number : 12 Publisher : Multidisciplinary Digital Publishing Institute, p. 1546. ISSN : 2079-9292. DOI : 10.3390/electronics8121546. URL : https://www.mdpi.com/2079-9292/8/12/1546 (visité le 17/04/2024).

[10] G. Demiris et B. K. Hensel. « Technologies for an Aging Society : A Systematic Review of "Smart Home" Applications ». en. In : *Yearb Med Inform* 17.01 (2008). Publisher : Georg Thieme Verlag KG, p. 33-40. ISSN : 0943-4747, 2364-0502. DOI : 10.1055/s-0038-1638580. URL : http://www.thieme-connect.de/DOI/DOI?10.1055/s-0038-1638580 (visité le 17/04/2024).

[11] *IoT connected devices by vertical 2030*. en. URL : https://www.statista.com/statistics/1194682/iot-connected-devices-vertically/ (visité le 10/07/2024).

[12] *Worldwide IoT revenue 2030*. en. URL : https://www.statista.com/statistics/1194709/iot-revenue-worldwide/ (visité le 15/07/2024).

[13] This text provides general information Statista assumes no liability for the information given being complete or correct Due to varying update cycles et Statistics Can Display More up-to-Date Data Than Referenced in the Text. *Topic : Internet of Things (IoT)*. en. URL : https://www.statista.com/topics/2637/internet-of-things/ (visité le 15/07/2024).

[14] Huanyu Wang et al. « Probing Attacks on Integrated Circuits : Challenges and Research Opportunities ». In : *IEEE Design & Test* 34.5 (oct. 2017), p. 63-71. ISSN : 2168-2356, 2168-2364. DOI : 10.1109/MDAT.2017.2729398. URL : http://ieeexplore.ieee.org/document/7984893/ (visité le 05/02/2025).

[15] Giovanni Camurati et al. « Screaming Channels : When Electromagnetic Side Channels Meet Radio Transceivers ». In : oct. 2018, p. 163-177. DOI : 10.1145/3243734.3243802.

[16] Mack DeGeurin. *AI companies eye fossil fuels to meet booming energy demand*. en-US. Mars 2024. URL : https://www.popsci.com/technology/ai-power/ (visité le 29/03/2024).

[17] Sally A. McKee. « Reflections on the memory wall ». In : *Proceedings of the 1st conference on Computing frontiers*. CF '04. New York, NY, USA : Association for Computing Machinery, avr. 2004, p. 162. ISBN : 978-1-58113-741-5. DOI : 10.1145/977091.977115. URL : https://doi.org/10.1145/977091.977115 (visité le 23/04/2024).

[18] Wm. A. Wulf et Sally A. McKee. « Hitting the memory wall : implications of the obvious ». In : *SIGARCH Comput. Archit. News* 23.1 (mars 1995), p. 20-24. ISSN : 0163-5964. DOI : 10.1145/216585.216588. URL : https://dl.acm.org/doi/10.1145/216585.216588 (visité le 23/04/2024).

[19] Harold S. Stone. « A Logic-in-Memory Computer ». In : *IEEE Transactions on Computers* C-19.1 (jan. 1970). Conference Name : IEEE Transactions on Computers, p. 73-78. ISSN : 1557-9956. DOI : 10.1109/TC.1970.5008902. URL : https://ieeexplore.ieee.org/abstract/document/5008902?casa_token=sJlYJ9lkNpQAAAAA:IqjeCz3f90GduaHdg56dA124J-LJAhd_ZolVx1LY5ZFXo1GHvTjz6nNbWHGdUuPKOCOoxPWcQ6p_ (visité le 27/04/2024).

[20] Xingqi Zou et al. « Breaking the von Neumann bottleneck : architecture-level processing-in-memory technology ». en. In : *Sci. China Inf. Sci.* 64.6 (avr. 2021), p. 160404. ISSN : 1869-1919. DOI : 10.1007/s11432-020-3227-1. URL : https://doi.org/10.1007/s11432-020-3227-1 (visité le 23/04/2024).

[21] Sven Beyer et al. « FeFET : A versatile CMOS compatible device with game-changing potential ». In : mai 2020, p. 1-4. DOI : 10.1109/IMW48823.2020.9108150.

[22] Esteban Garzón et al. « Reconfigurable CMOS/STT-MTJ Non-Volatile Circuit for Logic-in-Memory Applications ». In : *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*. ISSN : 2473-4667. Fév. 2020, p. 1-4. DOI : 10.1109/LASCAS45839.2020.9069027.

[23] Xiankai Lin et al. « Two-dimensional van der Waals ferroelectric field-effect transistors toward nonvolatile memory and neuromorphic computing ». In : *Applied Physics Letters* 123.18 (oct. 2023), p. 180501. ISSN : 0003-6951. DOI : 10.1063/5.0165837. URL : https://doi.org/10.1063/5.0165837 (visité le 18/07/2024).

[24] Jae Young Kim, Min-Ju Choi et Ho Won Jang. « Ferroelectric field effect transistors : Progress and perspective ». In : *APL Materials* 9.2 (fév. 2021), p. 021102. ISSN : 2166-532X. DOI : 10.1063/5.0035515. URL : https://doi.org/10.1063/5.0035515 (visité le 18/07/2024).

[25] Daniel Sanchez et Christos Kozyrakis. « ZSim : fast and accurate microarchitectural simulation of thousand-core systems ». In : *SIGARCH Comput. Archit. News* 41.3 (juin 2013), p. 475-486. ISSN : 0163-5964. DOI : 10.1145/2508148.2485963. URL : https://doi.org/10.1145/2508148.2485963 (visité le 27/03/2024).

[26] L. Chua. « Memristor-The missing circuit element ». In : *IEEE Transactions on Circuit Theory* 18.5 (sept. 1971). Conference Name : IEEE Transactions on Circuit Theory, p. 507-519. ISSN : 2374-9555. DOI : 10.1109/TCT.1971.1083337. URL : https://ieeexplore.ieee.org/abstract/document/1083337 (visité le 09/03/2025).

[27] Chen YangYin. *ReRAM : History, Status, and Future*. en-US. URL : https://ieeexplore.ieee.org/abstract/document/8961211 (visité le 09/03/2025).

[28] Shimeng Yu. *Resistive Random Access Memory (RRAM)*. en. Google-Books-ID : sonPCwAAQBAJ. Morgan & Claypool Publishers, mars 2016. ISBN : 978-1-62705-930-5.

[29] Scott W. Fong, Christopher M. Neumann et H.-S. Philip Wong. « Phase-Change Memory—Towards a Storage-Class Memory ». In : *IEEE Transactions on Electron Devices* 64.11 (nov. 2017). Conference Name : IEEE Transactions on Electron Devices, p. 4374-4385. ISSN : 1557-9646. DOI : 10.1109/TED.2017.2746342. URL : https://ieeexplore.ieee.org/abstract/document/8048346 (visité le 09/03/2025).

[30] J. Valasek. « Piezo-Electric and Allied Phenomena in Rochelle Salt ». In : *Physical Review* 17.4 (avr. 1921). Publisher : American Physical Society, p. 475-481. DOI : 10.1103/PhysRev.17.475. URL : https://link.aps.org/doi/10.1103/PhysRev.17.475 (visité le 09/03/2025).

[31] Hidemi Takasu. « The Ferroelectric Memory and its Applications ». en. In : *Journal of Electroceramics* 4.2 (juin 2000), p. 327-338. ISSN : 1573-8663. DOI : 10.1023/A:1009910525462. URL : https://doi.org/10.1023/A:1009910525462 (visité le 09/03/2025).

[32] T. Mikolajick, U. Schroeder et S. Slesazeck. « The Past, the Present, and the Future of Ferroelectric Memories ». In : *IEEE Transactions on Electron Devices* 67.4 (avr. 2020). Conference Name : IEEE Transactions on Electron Devices, p. 1434-1443. ISSN : 1557-9646. DOI : 10.1109/TED.2020.2976148. URL : https://ieeexplore.ieee.org/abstract/document/9032362 (visité le 09/03/2025).

[33] T. S. Böscke et al. « Ferroelectricity in hafnium oxide thin films ». In : *Applied Physics Letters* 99.10 (sept. 2011), p. 102903. ISSN : 0003-6951. DOI : 10.1063/1.3634052. URL : https://doi.org/10.1063/1.3634052 (visité le 09/03/2025).

[34] T. S. Böscke et al. « Ferroelectricity in hafnium oxide : CMOS compatible ferroelectric field effect transistors ». In : *2011 International Electron Devices Meeting*. ISSN : 2156-017X. Déc. 2011, p. 24.5.1-24.5.4. DOI : 10.1109/IEDM.2011.6131606. URL : https://ieeexplore.ieee.org/abstract/document/6131606 (visité le 09/03/2025).

[35] Yiqun WANG et al. « A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops ». In : sept. 2012, p. 149-152. ISBN : 978-1-4673-2212-6. DOI : 10.1109/ESSCIRC.2012.6341281.

[36] Furqan ZAHOOR, Tun Zainal AZNI ZULKIFLI et Farooq Ahmad KHANDAY. « Resistive Random Access Memory (RRAM) : an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications ». en. In : *Nanoscale Research Letters* 15.1 (avr. 2020), p. 90. ISSN : 1556-276X. DOI : 10.1186/s11671-020-03299-9. URL : https://doi.org/10.1186/s11671-020-03299-9 (visité le 24/07/2025).

[37] H.-S. Philip WONG et al. « Metal–Oxide RRAM ». In : *Proceedings of the IEEE* 100.6 (juin 2012). Conference Name : Proceedings of the IEEE, p. 1951-1970. ISSN : 1558-2256. DOI : 10.1109/JPROC.2012.2190369. URL : https://ieeexplore.ieee.org/abstract/document/6193402 (visité le 09/03/2025).

[38] C. GOPALAN et al. « Demonstration of Conductive Bridging Random Access Memory (CBRAM) in logic CMOS process ». In : *Solid-State Electronics*. Special Issue devoted to the 2nd International Memory Workshop (IMW 2010) 58.1 (avr. 2011), p. 54-61. ISSN : 0038-1101. DOI : 10.1016/j.sse.2010.11.024. URL : https://www.sciencedirect.com/science/article/pii/S0038110110004107 (visité le 09/03/2025).

[39] Yijun LI et al. « Monolithic three-dimensional integration of RRAM-based hybrid memory architecture for one-shot learning ». en. In : *Nature Communications* 14.1 (nov. 2023). Publisher : Nature Publishing Group, p. 7140. ISSN : 2041-1723. DOI : 10.1038/s41467-023-42981-1. URL : https://www.nature.com/articles/s41467-023-42981-1 (visité le 08/03/2025).

[40] Xifan TANG et al. « A RRAM-based FPGA for Energy-efficient Edge Computing ». In : *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. ISSN : 1558-1101. Mars 2020, 144–a–144-f. DOI : 10.23919/DATE48585.2020.9116478. URL : https://ieeexplore.ieee.org/abstract/document/9116478 (visité le 09/03/2025).

[41] Giriraj NYATI et al. « Development of phase change material for write once Bluray disc ». In : *Materials Science and Engineering : B* 164.3 (oct. 2009), p. 186-190. ISSN : 0921-5107. DOI : 10.1016/j.mseb.2009.09.014. URL : https://www.sciencedirect.com/science/article/pii/S0921510709003924 (visité le 24/07/2025).

[42] Geoffrey W. BURR et al. « Phase change memory technology ». In : *Journal of Vacuum Science & Technology B* 28.2 (mars 2010), p. 223-262. ISSN : 2166-2746. DOI : 10.1116/1.3301579. URL : https://doi.org/10.1116/1.3301579 (visité le 17/04/2025).

[43] Ali KESHAVARZI et al. « FerroElectronics for Edge Intelligence ». In : *IEEE Micro* 40.6 (nov. 2020), p. 33-48. ISSN : 1937-4143. DOI : 10.1109/MM.2020.3026667. URL : https://ieeexplore.ieee.org/abstract/document/9207822 (visité le 10/04/2025).

[44] S. R. NANDAKUMAR et al. « Phase-Change Memory Models for Deep Learning Training and Inference ». In : *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. Nov. 2019, p. 727-730. DOI : 10.1109/ICECS46596.2019.8964852. URL : https://ieeexplore.ieee.org/abstract/document/8964852 (visité le 09/03/2025).

[45] Irem BOYBAT et al. « Neuromorphic computing with multi-memristive synapses ». en. In : *Nature Communications* 9.1 (juin 2018). Publisher : Nature Publishing Group, p. 2514. ISSN : 2041-1723. DOI : 10.1038/s41467-018-04933-y. URL : https://www.nature.com/articles/s41467-018-04933-y (visité le 09/03/2025).

[46] Taha SHAHROODI et al. *High-Performance Data Mapping for BNNs on PCM-based Integrated Photonics*. en. arXiv :2401.17724 [cs]. Jan. 2024. DOI : 10.48550/arXiv.2401.17724. URL : http://arxiv.org/abs/2401.17724 (visité le 23/02/2025).

[47] Tifenn HIRTZLIN et al. « Digital Biologically Plausible Implementation of Binarized Neural Networks With Differential Hafnium Oxide Resistive Memory Arrays ». English. In : *Frontiers in Neuroscience* 13 (jan. 2020). Publisher : Frontiers. ISSN : 1662-453X. DOI : 10.3389/fnins.2019.01383. URL : https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2019.01383/full (visité le 24/07/2025).

[48] Jian-Gang (Jimmy) ZHU et Chando PARK. « Magnetic tunnel junctions ». In : *Materials Today* 9.11 (nov. 2006), p. 36-45. ISSN : 1369-7021. DOI : 10.1016/S1369-7021(06)71693-5. URL : https://www.sciencedirect.com/science/article/pii/S1369702106716935 (visité le 09/03/2025).

[49] Daniel C. WORLEDGE. « Spin-Transfer-Torque MRAM : the Next Revolution in Memory ». In : *2022 IEEE International Memory Workshop (IMW)*. ISSN : 2573-7503. Mai 2022, p. 1-4. DOI : 10.1109/IMW52921.2022.9779288. URL : https://ieeexplore.ieee.org/abstract/document/9779288 (visité le 09/03/2025).

[50] Thi-Nhan PHAM et al. « STT-BNN : A Novel STT-MRAM In-Memory Computing Macro for Binary Neural Networks ». In : *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 12.2 (juin 2022). Conference Name : IEEE Journal on Emerging and Selected Topics in Circuits and Systems, p. 569-579. ISSN : 2156-3365. DOI : 10.1109/JETCAS.2022.3169759. URL : https://ieeexplore.ieee.org/document/9762321?denied= (visité le 09/03/2025).

[51] Xiaoyu SUN et al. « Fully parallel RRAM synaptic array for implementing binary neural network with (+1, 1) weights and (+1, 0) neurons ». In : *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. ISSN : 2153-697X. Jan. 2018, p. 574-579. DOI : 10.1109/ASPDAC.2018.8297384. URL : https://ieeexplore.ieee.org/abstract/document/8297384 (visité le 24/07/2025).

[52] Huawei CHEN et al. « Logic gates based on neuristors made from two-dimensional materials ». en. In : *Nature Electronics* 4.6 (juin 2021). Publisher : Nature Publishing Group, p. 399-404. ISSN : 2520-1131. DOI : 10.1038/s41928-021-00591-z. URL : https://www.nature.com/articles/s41928-021-00591-z (visité le 24/07/2025).

[53] Pankaj SHARMA et al. « Nonvolatile ferroelectric domain wall memory ». In : *Science Advances* 3.6 (juin 2017). Publisher : American Association for the Advancement of Science, e1700512. DOI : 10.1126/sciadv.1700512. URL : https://www.science.org/doi/full/10.1126/sciadv.1700512 (visité le 09/03/2025).

[54] Dennis MEIER et Sverre M. SELBACH. « Ferroelectric domain walls for nanotechnology ». en. In : *Nature Reviews Materials* 7.3 (mars 2022). Publisher : Nature Publishing Group, p. 157-173. ISSN : 2058-8437. DOI : 10.1038/s41578-021-00375-z. URL : https://www.nature.com/articles/s41578-021-00375-z (visité le 09/03/2025).

[55] L. J. MCGILLY et al. « Controlling domain wall motion in ferroelectric thin films ». en. In : *Nature Nanotechnology* 10.2 (fév. 2015). Publisher : Nature Publishing Group, p. 145-150. ISSN : 1748-3395. DOI : 10.1038/nnano.2014.320. URL : https://www.nature.com/articles/nnano.2014.320 (visité le 09/03/2025).

[56] Jun OKUNO et al. « 1T1C FeRAM Memory Array Based on Ferroelectric HZO With Capacitor Under Bitline ». In : *IEEE Journal of the Electron Devices Society* 10 (2022), p. 29-34. ISSN : 2168-6734. DOI : 10.1109/JEDS.2021.3129279. URL : https://ieeexplore.ieee.org/abstract/document/9621226 (visité le 07/05/2025).

[57] Nicolas NAGEL et al. « An Overview of FeRAM Technology for High Density Applications ». en. In : *MRS Online Proceedings Library (OPL)* 655 (jan. 2000), p. CC1.1.1. ISSN : 1946-4274. DOI : 10.1557/PROC-655-CC1.1.1. URL : https://www.cambridge.org/core/journals/mrs-online-proceedings-library-archive/article/abs/an-overview-of-feram-technology-for-high-density-applications/D5D3BC00E21243964089EFB1B1D44534 (visité le 09/03/2025).

[58] Xunzhao YIN et al. « Exploiting ferroelectric FETs for low-power non-volatile logic-in-memory circuits ». In : *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ISSN : 1558-2434. Nov. 2016, p. 1-8. DOI : 10.1145/2966986.2967037. URL : https://ieeexplore.ieee.org/abstract/document/7827698 (visité le 10/03/2025).

[59] Xiaoming CHEN, Michael NIEMIER et Xiaobo Sharon HU. « Nonvolatile Lookup Table Design Based on Ferroelectric Field-Effect Transistors ». In : *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. ISSN : 2379-447X. Mai 2018, p. 1-5. DOI : 10.1109/ISCAS.2018.8351375. URL : https://ieeexplore.ieee.org/abstract/document/8351375 (visité le 10/03/2025).

[60] Asif Islam KHAN, Ali KESHAVARZI et Suman DATTA. « The future of ferroelectric field-effect transistor technology ». en. In : *Nature Electronics* 3.10 (oct. 2020). Publisher : Nature Publishing Group, p. 588-597. ISSN : 2520-1131. DOI : 10.1038/s41928-020-00492-7. URL : https://www.nature.com/articles/s41928-020-00492-7 (visité le 17/04/2025).

[61] Kai NI, Sourav DUTTA et Suman DATTA. « Ferroelectrics : From Memory to Computing ». In : *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. ISSN : 2153-697X. Jan. 2020, p. 401-406. DOI : 10.1109/ASP-DAC47756.2020.9045150. URL : https://ieeexplore.ieee.org/abstract/document/9045150 (visité le 17/04/2025).

[62] Attilio BELMONTE et al. « Analysis of the Excellent Memory Disturb Characteristics of a Hourglass-Shaped Filament in Al2O3/Cu-Based CBRAM Devices ». In : *IEEE Transactions on Electron Devices* 62.6 (juin 2015), p. 2007-2013. ISSN : 1557-9646. DOI : 10.1109/TED.2015.2423094. URL : https://ieeexplore.ieee.org/abstract/document/7103322 (visité le 17/04/2025).

[63] J. Minguet LOPEZ et al. « Elucidating 1S1R operation to reduce the read voltage margin variability by stack and programming conditions optimization ». In : *2021 IEEE International Reliability Physics Symposium (IRPS)*. ISSN : 1938-1891. Mars 2021, p. 1-6. DOI : 10.1109/IRPS46558.2021.9405195. URL : https://ieeexplore.ieee.org/abstract/document/9405195 (visité le 17/04/2025).

[64] G. HU et al. « Spin-transfer torque MRAM with reliable 2 ns writing for last level cache applications ». In : *2019 IEEE International Electron Devices Meeting (IEDM)*. ISSN : 2156-017X. Déc. 2019, p. 2.6.1-2.6.4. DOI : 10.1109/IEDM19573.2019.8993604. URL : https://ieeexplore.ieee.org/abstract/document/8993604 (visité le 17/04/2025).

[65] K.-T. CHEN et al. « Non-Volatile Ferroelectric FETs Using 5-nm Hf0.5Zr0.5O2 With High Data Retention and Read Endurance for 1T Memory Applications ». In : *IEEE Electron Device Letters* 40.3 (mars 2019), p. 399-402. ISSN : 1558-0563. DOI : 10.1109/LED.2019.2896231. URL : https://ieeexplore.ieee.org/abstract/document/8630859 (visité le 17/04/2025).

[66] Korok CHATTERJEE et al. « Self-Aligned, Gate Last, FDSOI, Ferroelectric Gate Memory Device With 5.5-nm Hf0.8Zr0.2O2, High Endurance and Breakdown Recovery ». In : *IEEE Electron Device Letters* 38.10 (oct. 2017), p. 1379-1382. ISSN :

1558-0563. DOI : 10.1109/LED.2017.2748992. URL : https://ieeexplore.ieee.org/abstract/document/8025570 (visité le 17/04/2025).

[67] Valerio MILO et al. « Memristive and CMOS Devices for Neuromorphic Computing ». en. In : *Materials* 13.1 (jan. 2020). Company : Multidisciplinary Digital Publishing Institute Distributor : Multidisciplinary Digital Publishing Institute Institution : Multidisciplinary Digital Publishing Institute Label : Multidisciplinary Digital Publishing Institute Publisher : publisher. ISSN : 1996-1944. DOI : 10.3390/ma13010166. URL : https://www.mdpi.com/1996-1944/13/1/166 (visité le 18/01/2026).

[68] Asif Ali KHAN et al. *The Landscape of Compute-near-memory and Compute-in-memory : A Research and Commercial Overview*. arXiv :2401.14428 [cs]. Jan. 2024. DOI : 10.48550/arXiv.2401.14428. URL : http://arxiv.org/abs/2401.14428 (visité le 24/11/2025).

[69] P. MANNOCCI et al. « In-memory computing with emerging memory devices : Status and outlook ». In : *APL Machine Learning* 1.1 (fév. 2023), p. 010902. ISSN : 2770-9019. DOI : 10.1063/5.0136403. URL : https://doi.org/10.1063/5.0136403 (visité le 24/11/2025).

[70] Danny SHUM et al. « Functionality Demonstration of a High-Density 1.1V Self-Aligned Split-Gate NVM Cell Embedded into LP 40 nm CMOS for Automotive and Smart Card Applications ». In : *2015 IEEE International Memory Workshop (IMW)*. ISSN : 2159-4864. Mai 2015, p. 1-4. DOI : 10.1109/IMW.2015.7150288. URL : https://ieeexplore.ieee.org/abstract/document/7150288 (visité le 17/04/2025).

[71] Akihiko KANDA et al. « A 24-MB Embedded Flash System Based on 28-nm SG-MONOS Featuring 240-MHz Read Operations and Robust Over-the-Air Software Update for Automotive Applications ». In : *IEEE Solid-State Circuits Letters* 2.12 (déc. 2019), p. 273-276. ISSN : 2573-9603. DOI : 10.1109/LSSC.2019.2948813. URL : https://ieeexplore.ieee.org/abstract/document/8879506 (visité le 17/04/2025).

[72] Maha KOOLI et al. « Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution ». In : *J. Emerg. Technol. Comput. Syst.* 18.2 (avr. 2022), 40 :1-40 :26. ISSN : 1550-4832. DOI : 10.1145/3485823. URL : https://doi.org/10.1145/3485823 (visité le 02/04/2025).

[73] Muhammad IRFAN et al. « Reconfigurable content-addressable memory (CAM) on FPGAs : A tutorial and survey ». In : *Future Generation Computer Systems* 128 (mars 2022), p. 451-465. ISSN : 0167-739X. DOI : 10.1016/j.future.2021.09.037. URL : https://www.sciencedirect.com/science/article/pii/S0167739X21003836 (visité le 02/04/2025).

[74] Nishil Talati et al. « Logic Design Within Memristive Memories Using Memristor-Aided loGIC (MAGIC) ». In : *IEEE Transactions on Nanotechnology* 15.4 (juill. 2016). Conference Name : IEEE Transactions on Nanotechnology, p. 635-650. ISSN : 1941-0085. DOI : 10.1109/TNANO.2016.2570248. URL : https://ieeexplore.ieee.org/abstract/document/7471529 (visité le 02/04/2025).

[75] *A Classification of Memory-Centric Computing — ACM Journal on Emerging Technologies in Computing Systems*. URL : https://dl.acm.org/doi/abs/10.1145/3365837 (visité le 02/04/2025).

[76] Sparsh Mittal et Jeffrey S. Vetter. « A Survey Of Techniques for Architecting DRAM Caches ». In : *IEEE Transactions on Parallel and Distributed Systems* 27.6 (juin 2016). Conference Name : IEEE Transactions on Parallel and Distributed Systems, p. 1852-1863. ISSN : 1558-2183. DOI : 10.1109/TPDS.2015.2461155. URL : https://ieeexplore.ieee.org/abstract/document/7181712 (visité le 02/04/2025).

[77] Harekrishna Kumar et V. K. Tomar. « A Review on Performance Evaluation of Different Low Power SRAM Cells in Nano-Scale Era ». en. In : *Wireless Personal Communications* 117.3 (avr. 2021), p. 1959-1984. ISSN : 1572-834X. DOI : 10.1007/s11277-020-07953-4. URL : https://doi.org/10.1007/s11277-020-07953-4 (visité le 02/04/2025).

[78] S. Barraud et al. « 3D RRAMs with Gate-All-Around Stacked Nanosheet Transistors for In-Memory-Computing ». In : *2020 IEEE International Electron Devices Meeting (IEDM)*. ISSN : 2156-017X. Déc. 2020, p. 29.5.1-29.5.4. DOI : 10.1109/IEDM13553.2020.9371982. URL : https://ieeexplore.ieee.org/abstract/document/9371982 (visité le 18/04/2025).

[79] Shubham Jain et al. « Computing in Memory With Spin-Transfer Torque Magnetic RAM ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.3 (mars 2018), p. 470-483. ISSN : 1557-9999. DOI : 10.1109/TVLSI.2017.2776954. URL : https://ieeexplore.ieee.org/abstract/document/8241447 (visité le 18/04/2025).

[80] Xunzhao Yin, Michael Niemier et X. Sharon Hu. « Design and benchmarking of ferroelectric FET based TCAM ». In : *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. ISSN : 1558-1101. Mars 2017, p. 1444-1449. DOI : 10.23919/DATE.2017.7927219. URL : https://ieeexplore.ieee.org/abstract/document/7927219 (visité le 18/04/2025).

[81] Cédric Marchand et al. « A FeFET-Based Hybrid Memory Accessible by Content and by Address ». In : *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 8.1 (juin 2022), p. 19-26. ISSN : 2329-9231. DOI : 10.1109/JXCDC.2022.3168057. URL : https://ieeexplore.ieee.org/abstract/document/9758734 (visité le 18/04/2025).

[82] Saeed KARGAR et Faisal NAWAB. « Challenges and future directions for energy, latency, and lifetime improvements in NVMs ». en. In : *Distributed and Parallel Databases* 41.3 (sept. 2023), p. 163-189. ISSN : 1573-7578. DOI : 10.1007/s10619-022-07421-x. URL : https://doi.org/10.1007/s10619-022-07421-x (visité le 09/05/2025).

[83] Robert STRENZ. « Review and Outlook on Embedded NVM Technologies – From Evolution to Revolution ». In : *2020 IEEE International Memory Workshop (IMW)*. ISSN : 2573-7503. Mai 2020, p. 1-4. DOI : 10.1109/IMW48823.2020.9108121. URL : https://ieeexplore.ieee.org/abstract/document/9108121 (visité le 09/05/2025).

[84] Howard H. AIKEN et Grace M. HOPPER. « The automatic sequence controlled calculator — I ». In : *Electrical Engineering* 65.8-9 (août 1946). Conference Name : Electrical Engineering, p. 384-391. ISSN : 2376-7804. DOI : 10.1109/EE.1946.6434251. URL : https://ieeexplore.ieee.org/abstract/document/6434251 (visité le 23/03/2025).

[85] William ASPRAY. *John von Neumann and the Origins of Modern Computing*. en. Google-Books-ID : srhNEAAAQBAJ. MIT Press, déc. 1990. ISBN : 978-0-262-51885-7.

[86] H. H. GOLDSTINE et Adele GOLDSTINE. « The Electronic Numerical Integrator and Computer (ENIAC) ». en. In : *The Origins of Digital Computers : Selected Papers*. Sous la dir. de Brian RANDELL. Berlin, Heidelberg : Springer, 1982, p. 359-373. ISBN : 978-3-642-61812-3. DOI : 10.1007/978-3-642-61812-3_29. URL : https://doi.org/10.1007/978-3-642-61812-3_29 (visité le 23/03/2025).

[87] W. ASPRAY. « The Intel 4004 microprocessor : what constituted invention ? » In : *IEEE Annals of the History of Computing* 19.3 (juill. 1997), p. 4-15. ISSN : 1934-1547. DOI : 10.1109/85.601727. URL : https://ieeexplore.ieee.org/abstract/document/601727 (visité le 24/07/2025).

[88] Stanley MAZOR. « Intel's 8086 ». In : *IEEE Annals of the History of Computing* 32.1 (jan. 2010), p. 75-79. ISSN : 1934-1547. DOI : 10.1109/MAHC.2010.22. URL : https://ieeexplore.ieee.org/abstract/document/5430762 (visité le 24/07/2025).

[89] A. D. GEORGE. « An overview of RISC vs. CISC ». English. In : *Proceedings The Twenty-Second Southeastern Symposium on System Theory*. ISSN : 0094-2898. IEEE Computer Society, jan. 1990, p. 436, 437, 438-436, 437, 438. DOI : 10.1109/SSST.1990.138185. URL : https://www.computer.org/csdl/proceedings-article/ssst/1990/00138185/12OmNzlUKsE (visité le 24/07/2025).

[90] *OpenCores*. en. Page Version ID : 1260140006. Nov. 2024. URL : https://en.wikipedia.org/w/index.php?title=OpenCores&oldid=1260140006 (visité le 02/04/2025).

[91] *Damjan Lampret*. URL : http://www.lampret.com/ (visité le 02/04/2025).

[92] John SHALF. « The future of computing beyond Moore's Law ». In : *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences* 378.2166 (jan. 2020). Publisher : Royal Society, p. 20190061. DOI : 10.1098/rsta.2019.0061. URL : https://royalsocietypublishing.org/doi/full/10.1098/rsta.2019.0061 (visité le 02/04/2025).

[93] John HENNESSY et al. « MIPS : A microprocessor architecture ». In : *SIGMICRO Newsl.* 13.4 (oct. 1982), p. 17-22. ISSN : 1050-916X. DOI : 10.1145/1014194.800930. URL : https://dl.acm.org/doi/10.1145/1014194.800930 (visité le 24/07/2025).

[94] *MIPS Pivots to RISC-V with Best-In-Class Performance and Scalability — MIPS – RISC-V International*. en-US. URL : https://riscv.org/blog/2022/05/mips-pivots-to-risc-v-with-best-in-class-performance-and-scalability-mips/ (visité le 24/07/2025).

[95] *About RISC-V International*. en-US. URL : https://riscv.org/about/ (visité le 24/07/2025).

[96] *RV32I, RV64I Instructions — riscv-isa-pages documentation*. URL : https://msyksphinz-self.github.io/riscv-isadoc/html/rvi.html# (visité le 03/04/2025).

[97] *Instruction Set Assembly Guide for Armv7 and earlier Arm architectures Reference Guide*. URL : https://developer.arm.com/documentation/100076/0200/a32-t32-instruction-set-reference/a32-and-t32-instructions/a32-and-t32-instruction-summary?lang=en (visité le 03/04/2025).

[98] Enfang CUI, Tianzheng LI et Qian WEI. « RISC-V Instruction Set Architecture Extensions : A Survey ». In : *IEEE Access* 11 (2023), p. 24696-24711. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2023.3246491. URL : https://ieeexplore.ieee.org/abstract/document/10049118 (visité le 09/04/2025).

[99] *chipsalliance/rocket-chip*. original-date : 2014-09-12T07 :04 :30Z. Avr. 2025. URL : https://github.com/chipsalliance/rocket-chip (visité le 03/04/2025).

[100] Xiaoyong XUE et al. « A RISC-V Processor with Area-Efficient Memristor-Based In-Memory Computing for Hash Algorithm in Blockchain Applications ». en. In : *Micromachines* 10.8 (août 2019). Number : 8 Publisher : Multidisciplinary Digital Publishing Institute, p. 541. ISSN : 2072-666X. DOI : 10.3390/mi10080541. URL : https://www.mdpi.com/2072-666X/10/8/541 (visité le 09/04/2025).

[101] Andrea COLUCCIO et al. « RISC-Vlim, a RISC-V Framework for Logic-in-Memory Architectures ». en. In : *Electronics* 11.19 (jan. 2022). Number : 19 Publisher : Multidisciplinary Digital Publishing Institute, p. 2990. ISSN : 2079-9292. DOI : 10.3390/electronics11192990. URL : https://www.mdpi.com/2079-9292/11/19/2990 (visité le 09/04/2025).

[102] Ruobing HAN et al. *Supporting CUDA for an extended RISC-V GPU architecture*. arXiv :2109.00673 [cs]. Sept. 2021. DOI : 10.48550/arXiv.2109.00673. URL : http://arxiv.org/abs/2109.00673 (visité le 09/05/2025).

[103] Pietro NANNIPIERI et al. « A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms ». In : *IEEE Access* 9 (2021), p. 150798-150808. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2021.3126208. URL : https://ieeexplore.ieee.org/abstract/document/9605604 (visité le 09/05/2025).

[104] Ashis Kumar CHAKRABORTY et al. « Hardware and Software Reliability, Verification, and Testing ». en. In : (2023). Sous la dir. d'Hoang PHAM, p. 415-442. DOI : 10.1007/978-1-4471-7503-2_22. URL : https://doi.org/10.1007/978-1-4471-7503-2_22 (visité le 24/07/2025).

[105] Wentao WANG, Zhuoxuan SHEN et Venkata DINAVAHI. « Physics-Based Device-Level Power Electronic Circuit Hardware Emulation on FPGA ». In : *IEEE Transactions on Industrial Informatics* 10.4 (nov. 2014), p. 2166-2179. ISSN : 1941-0050. DOI : 10.1109/TII.2014.2361656. URL : https://ieeexplore.ieee.org/abstract/document/6917006 (visité le 13/05/2025).

[106] Simone RUFFINI et al. « NORM : An FPGA-based Non-volatile Memory Emulation Framework for Intermittent Computing ». In : *J. Emerg. Technol. Comput. Syst.* 18.4 (oct. 2022), 73 :1-73 :18. ISSN : 1550-4832. DOI : 10.1145/3517812. URL : https://dl.acm.org/doi/10.1145/3517812 (visité le 13/05/2025).

[107] Matrangolo PAUL-ANTOINE et al. « Hardware Emulation of FeFET On FPGA ». In : *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. ISSN : 2159-3477. Juill. 2022, p. 380-385. DOI : 10.1109/ISVLSI54635.2022.00085. URL : https://ieeexplore.ieee.org/abstract/document/9912067 (visité le 01/09/2025).

[108] *NASA Makes RISC-V the Go-to Ecosystem for Future Space Missions*. en-us. URL : https://www.sifive.com/press/nasa-selects-sifive-and-makes-risc-v-the-go-to-ecosystem (visité le 01/09/2025).

[109] *RISC-V Members*. en-US. URL : https://riscv.org/members/ (visité le 01/09/2025).

[110] *3.1. Rocket Chip — Chipyard 1.11.0 documentation*. URL : https://chipyard.readthedocs.io/en/stable/Generators/Rocket-Chip.html (visité le 01/09/2025).

[111] Krste ASANOVI, Rimas AVIZIENIS et Jonathan BACHRACH. *https ://aspire.eecs.berkeley.edu/wp/wp-content/uploads/2016/04/Tech-Report-The-Rocket-Chip-Generator-Beamer.pdf*. Avr. 2015. URL : https://aspire.eecs.

berkeley.edu/wp/wp-content/uploads/2016/04/Tech-Report-The-Rocket-Chip-Generator-Beamer.pdf (visité le 01/09/2025).

[112] Francesco CONTI et al. « PULP : A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision ». en. In : *Journal of Signal Processing Systems* 84.3 (sept. 2016), p. 339-354. ISSN : 1939-8115. DOI : 10.1007/s11265-015-1070-9. URL : https://doi.org/10.1007/s11265-015-1070-9 (visité le 01/09/2025).

[113] FRAUNHOFER-IMS. *NEORV32 — Zephyr Project Documentation*. Juin 2025. URL : https://docs.zephyrproject.org/latest/boards/others/neorv32/doc/index.html (visité le 01/09/2025).

[114] Kristian Klomsten SKORDAL. *skordal/potato*. original-date : 2015-11-18T20 :23 :24Z. Août 2025. URL : https://github.com/skordal/potato (visité le 01/09/2025).

[115] *SpinalHDL/VexRiscv*. original-date : 2017-03-08T21 :14 :28Z. Sept. 2025. URL : https://github.com/SpinalHDL/VexRiscv (visité le 01/09/2025).

[116] Simon ROKICKI et al. « What You Simulate Is What You Synthesize : Designing a Processor Core from C++ Specifications ». In : *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ISSN : 1558-2434. Nov. 2019, p. 1-8. DOI : 10.1109/ICCAD45719.2019.8942177. URL : https://ieeexplore.ieee.org/abstract/document/8942177 (visité le 01/09/2025).

[117] Sakari LAHTI et al. « Are we there yet ? A study on the state of high-level synthesis ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.5 (2018), p. 898-911.

[118] Marie A WRIGHT. « The Advanced Encryption Standard ». In : *Network Security* 2001.10 (oct. 2001), p. 11-13. ISSN : 1353-4858. DOI : 10.1016/S1353-4858(01)01018-2. URL : https://www.sciencedirect.com/science/article/pii/S1353485801010182 (visité le 16/07/2025).

[119] National Institute of STANDARDS et al. *Advanced Encryption Standard (AES)*. en. 2001-11-26 00 :11 :00 2001. DOI : https://doi.org/10.6028/NIST.FIPS.197. URL : https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427.

[120] *https ://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf*. URL : https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf (visité le 16/07/2025).

[121] M. VAIDEHI et B. Justus RABI. « Design and analysis of AES-CBC mode for high security applications ». In : *Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014*. Juill. 2014, p. 499-502. DOI : 10.1109/ICCTET.2014.6966347. URL : https://ieeexplore.ieee.org/abstract/document/6966347 (visité le 30/07/2025).

[122] Helger Lipmaa, Phillip Rogaway et David Wagner. « CTR-mode encryption ». In : *First NIST Workshop on Modes of Operation*. T. 39. Citeseer. MD. 2000.

[123] David McGrew et John Viega. « The Galois/counter mode of operation (GCM) ». In : *submission to NIST Modes of Operation Process* 20 (2004), p. 0278-0070.

[124] S Dutta et al. « Monolithic 3D integration of high endurance multi-bit ferroelectric FET for accelerating compute-in-memory ». In : *2020 IEEE International Electron Devices Meeting (IEDM)*. IEEE. Piscataway, NJ USA : IEEE, 2020, p. 36-4.

[125] Paul-Antoine Matrangolo. « Processeur sécurisé et reconfigurable incluant des technologies émergentes ». fr. Thèse de doct. Ecole Centrale de Lyon, avr. 2025. URL : https://theses.hal.science/tel-05150272 (visité le 13/08/2025).

[126] Eisuke Tokumitsu, Gen Fujii et Hiroshi Ishiwara. « Electrical Properties of Metal-Ferroelectric-Insulator-Semiconductor (MFIS)- and Metal-Ferroelectric-Metal-Insulator-Semiconductor (MFMIS)-FETs Using Ferroelectric SrBi2Ta2O9 Film and SrTa2O6/SiON Buffer Layer ». en. In : *Japanese Journal of Applied Physics* 39.4S (avr. 2000). Publisher : IOP Publishing, p. 2125. ISSN : 1347-4065. DOI : 10.1143/JJAP.39.2125. URL : https://iopscience.iop.org/article/10.1143/JJAP.39.2125/meta (visité le 13/08/2025).

[127] George Clifford Sziklai. « Symmetrical Properties of Transistors and Their Applications ». In : *Proceedings of the IRE* 41.6 (juin 1953), p. 717-724. ISSN : 2162-6634. DOI : 10.1109/JRPROC.1953.274250. URL : https://ieeexplore.ieee.org/abstract/document/4051378 (visité le 13/08/2025).

[128] R. Zimmermann et W. Fichtner. « Low-power logic styles : CMOS versus pass-transistor logic ». In : *IEEE Journal of Solid-State Circuits* 32.7 (juill. 1997), p. 1079-1090. ISSN : 1558-173X. DOI : 10.1109/4.597298. URL : https://ieeexplore.ieee.org/abstract/document/597298 (visité le 13/08/2025).

[129] K. Ni et al. « SoC Logic Compatible Multi-Bit FeMFET Weight Cell for Neuromorphic Applications ». In : *2018 IEEE International Electron Devices Meeting (IEDM)*. ISSN : 2156-017X. Déc. 2018, p. 13.2.1-13.2.4. DOI : 10.1109/IEDM.2018.8614496. URL : https://ieeexplore.ieee.org/abstract/document/8614496 (visité le 16/08/2025).

[130] P Scott, SE Tavares et L Peppard. « A fast VLSI multiplier for GF (2 m) ». In : *IEEE Journal on selected Areas in Communications* 4.1 (1986), p. 62-66.

[131] Valeriu Beiu, Azam Beg et Walid Ibrahim. « Atto-Joule gates for the whole voltage range ». In : *2011 11th IEEE International Conference on Nanotechnology*. ISSN : 1944-9399. Août 2011, p. 1424-1429. DOI : 10.1109/NANO.2011.6144399. URL : https://ieeexplore.ieee.org/abstract/document/6144399 (visité le 10/09/2025).

[132] Cédric MARCHAND et al. « FeFET based Logic-in-Memory design methodologies, tools and open challenges ». In : *2023 IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC)*. ISSN : 2324-8440. Oct. 2023, p. 1-6. DOI : 10.1109/VLSI-SoC57769.2023.10321901. URL : https://ieeexplore.ieee.org/abstract/document/10321901 (visité le 01/09/2025).

[133] Alban NICOLAS, Cédric MARCHAND et David NAVARRO. « Non Volatile Operators Emulation Platform ». In : *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*. NANOARCH '23. New York, NY, USA : Association for Computing Machinery, jan. 2024, p. 1-6. ISBN : 979-8-4007-0325-6. DOI : 10.1145/3611315.3633252. URL : https://doi.org/10.1145/3611315.3633252 (visité le 01/09/2025).