UNIVERSITÉ DE MONTRÉAL

OPÉRATEURS ET ENGINS DE CALCUL EN VIRGULE FLOTTANTE ET LEUR APPLICATION À LA SIMULATION EN TEMPS RÉEL SUR FPGA

TAREK OULD BACHIR
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE ÉLECTRIQUE)
AOÛT 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

OPÉRATEURS ET ENGINS DE CALCUL EN VIRGULE FLOTTANTE ET LEUR APPLICATION À LA SIMULATION EN TEMPS RÉEL SUR FPGA

présentée par: OULD BACHIR Tarek
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. BRAULT Jean-Jules, Ph.D., président
M. DAVID Jean Pierre, Ph.D., membre et directeur de recherche
M. MAHSEREDJIAN Jean, Ph.D., membre et codirecteur de recherche
M. SIROIS Frédéric, Ph.D., membre et codirecteur de recherche
M. LANGLOIS Pierre, Ph.D., membre
M. LE-HUY Hoang, Dr. Eng., membre

It's simple. We... kill the Batman.

— Heath Ledger, a.k.a The Joker.

Oh ! les gens bienheureux !... Tout à coup, dans l'espace,
Si haut qu'il semble aller lentement, un grand vol
En forme de triangle arrive, plane et passe.
Où vont-ils ? Qui sont-ils ? Comme ils sont loin du sol !

Regardez-les passer ! Eux, ce sont les sauvages.
Ils vont où leur désir le veut, par-dessus monts,
Et bois, et mers, et vents, et loin des esclavages.
L'air qu'ils boivent ferait éclater vos poumons.

— Extraits des oiseaux de passage, poème de Jean Richepin.

Cette thèse est dédiée aux étudiants qui
ont pris massivement la rue en 2012.
Ils ont fait éclore le printemps québécois
et redonné espoir aux gens comme moi.

Elle l'est aussi à toi, mon petit Tristan,
qui as défié l'antiémeute
et ses gaz lacrymogènes
depuis le douillet ventre de maman.

REMERCIEMENTS

Une thèse de doctorat est une longue aventure qui ne se mène pas seul. Les remerciements qui suivent lèvent le chapeau à tous ceux qui ont participé par leur gentillesse, leur temps, leurs discussions, leur amour à ces années de labeur que concluent en quelques pages ce document. Mes remerciements premiers iront aux membres du jury qui ont bien voulu lire ces pages et y porter leurs jugement et sanction. Je les remercie de leur temps et de l'apport indispensable de leur travail. Mes remerciements iront également aux commanditaires de la thèse : les fonds gouvernementaux du CRSNG et du FQRNT et la compagnie Opal-RT Technologies, et plus particulièrement Jean Bélanger dont l'intuition a si souvent été juste, lui qui en plus d'apporter le concours des finances de la compagnie qu'il a fondée, n'a jamais hésité à payer de son temps, d'un verre, d'idées jetées comme autant de destinations où diriger sa barque. Je remercie également mes directeurs qui ont joué un rôle déterminant dans la réalisation de ce travail. Chacun d'eux a contribué d'une façon ou d'une autre à la bonne réussite de ce projet, et je les en remercie chaleureusement.

Je souhaiterais remercier toutes ces personnes à Opal qui m'ont aidé à avancer et à réaliser les différentes contributions de la thèse. Je remercie Handy Fortin Blanchette, aujourd'hui professeur à l'ÉTS et chef de l'équipe eDriveSim au moment où je débutais ma thèse. Handy m'a aidé à voir plus clair dans un domaine qui m'était en grande partie étranger, et il le fit du temps qu'il était employé d'Opal et même après. Je remercie également Christian Dufour qui a pris le relais de Handy dans mon encadrement à Opal. Notre collaboration a été très fructueuse, ponctuée de projets industriels répondant aux commandes des clients, et riche de discussions autour d'un poste d'ordinateur, d'une bière ou par Skype interposé. Je remercie aussi Sébastien Cense qui a aidé à la réalisation matérielle de mes prototypes, particulièrement l'eHS. Je ne manquerai non plus de remercier les amis du laboratoire : pour sa patience et sa pugnacité dans des conversations souvent musclées, Marc-André Daigneault ; pour son cynisme rieur et esprit volontaire, Jonas Lerebours ; pour sa gentillesse et disponibilité, Adrien Larbanet ; pour sa cinéphilie et son humour, Mathieu Allard ; pour son entregent et sa bonhomie, Hani Saad.

Sur une note plus intime, j'aimerais remercier les miens, Mahdi Khelfaoui et Imad Safi, amis de tous les temps qui donnent une tape sur l'épaule et écoutent avec la patience et les encouragements que seuls les amis savent témoigner ; et finalement ma compagne et amie Fanny Lalonde, qui sait l'indulgence lorsqu'arrive une fin de semaine de travail, elle qui, en complétant sa maîtrise et sans compter ses heures pour l'association étudiante, a donné la vie et illuminé la nôtre.

RÉSUMÉ

La simulation en temps réel des réseaux électriques connaît un vif intérêt industriel, motivé par la réduction substantielle des coûts de développement qu'offre une telle approche de prototypage. Ainsi, la simulation en temps réel permet d'intégrer dans la boucle de la simulation du matériel au fur et à mesure sa conception, permettant du même coup d'en vérifier le bon fonctionnement dans des conditions réalistes. Néanmoins, la simulation en temps réel au moyen de CPU, telle qu'elle a été pensée depuis une quinzaine d'années, souffre de certaines limitations, notamment dans l'atteinte de pas de calcul de l'ordre de quelques micro-secondes, un requis important pour la simulation fidèle des transitoires rapides qu'exigent les convertisseurs de puissance modernes.

Pour tenter d'apporter une réponse à ces difficultés, les industriels ont adopté les circuits FPGA pour la réalisation d'engins de calcul dédiés à la simulation rapide des réseaux électriques, ce qui a permis de franchir la barrière de la fréquence de commutation de 5 kHz qui était caractéristique de la simulation sur CPU. La simulation sur FPGA offre à ce titre différents avantages telle que la réduction de la latence de la boucle de simulation du matériel sous test, particulièrement du fait que le FPGA donne un accès direct aux senseurs et aux actuateurs du dispositif en cours de prototypage.

Les paradigmes usuels du traitement de signal sur FPGA font qu'il est d'usage d'y opérer une arithmétique à virgule fixe. Ce format des nombres pénalise le temps de développement puisqu'il requiert du concepteur une évaluation complexe de la précision nécessaire pour représenter l'ensemble des variables du modèle mathématique. C'est pourquoi l'arithmétique à virgule flottante suscite un certain intérêt dans la simulation des réseaux sur FPGA. Cependant, les opérateurs en virgule flottante imposent de longues latences, particulièrement handicapantes dans la réalisation de lois d'intégration (trapézoïdale, Euler-arrière, etc.) pour lesquelles l'utilisation d'un accumulateur à un cycle est cruciale. En cela, la problématique de l'addition et de l'accumulation en virgule flottante forme le cœur de notre travail de recherche. Ce travail a permis l'élaboration des architectures d'accumulateurs, de multiplieurs accumulateurs (MAC) et d'opérateurs de produit scalaire (OPS) en virgule flottante, qui joueront un rôle déterminant dans la mise en œuvre de nos engins de calcul pour la simulation des réseaux électriques.

Ainsi, le travail présenté dans cette thèse propose différentes contributions scientifiques au domaine de la simulation en temps réel sur FPGA. D'une part, il contribue à la formulation d'un algorithme de sommation qui est une généralisation de la technique d'auto-alignement, nantie ici d'une formulation et d'une réalisation matérielle simplifiées. Le travail établit les

critères permettant de garantir la bonne exactitude des résultats, critères que nous avons établis par des démonstrations théoriques et empiriques. La thèse propose également une analyse exhaustive de l'utilisation du format redondant *high radix carry-save* (HRCS) dans l'addition de mantisses larges, format pour lequel deux nouveaux opérateurs arithmétiques sont proposés : un additionneur endomorphique ainsi qu'un convertisseur HRCS à conventionnel.

Une fois l'addition en virgule flottante à un cycle réalisée, la thèse propose de concevoir sur FPGA des engins de calcul exploitant une architecture SIMD (*single instruction, multiple data*) et disposant de plusieurs MAC ou opérateurs de produit scalaire (OPS) en virgule flottante. Ces opérateurs présentent une latence très courte, permettant l'atteinte de pas de calcul de quelques centaines de nanosecondes dans la simulation de convertisseurs de puissance de moyenne complexité. Le travail se conclut par l'analyse de la modélisation de circuits d'électronique de puissance et la présentation d'un engin de calcul versatile permettant de simuler des convertisseurs de topologie arbitraire, pouvant contenir jusqu'à 24 interrupteurs avec des pas de temps en deçà du 1 μs, tout en admettant des fréquences de commutations de plusieurs kilohertz. Cette réalisation a abouti à une commercialisation par le partenaire industriel de la thèse.

ABSTRACT

The real-time simulation of electrical networks gained a vivid industrial interest during recent years, motivated by the substantial development cost reduction that such a prototyping approach can offer. Real-time simulation allows the progressive inclusion of real hardware during its development, allowing its testing under realistic conditions. However, CPU-based simulations suffer from certain limitations such as the difficulty to reach time-steps of a few microsecond, an important challenge brought by modern power converters.

Hence, industrial practitioners adopted the FPGA as a platform of choice for the implementation of calculation engines dedicated to the rapid real-time simulation of electrical networks. The reconfigurable technology broke the 5 kHz switching frequency barrier that is characteristic of CPU-based simulations. Moreover, FPGA-based real-time simulation offers many advantages, including the reduced latency of the simulation loop that is obtained thanks to a direct access to sensors and actuators.

The fixed-point format is paradigmatic to FPGA-based digital signal processing. However, the format imposes a time penalty in the development process since the designer has to asses the required precision for all model variables. This fact brought an import research effort on the use of the floating-point format for the simulation of electrical networks. One of the main challenges in the use of the floating-point format are the long latencies required by the elementary arithmetic operators, particularly when an adder is used as an accumulator, an important building bloc for the implementation of integration rules such as the trapezoidal method. Hence, single-cycle floating-point accumulation forms the core of this research work. Our results help building such operators as accumulators, multiply-accumulators (MACs), and dot-product (DP) operators. These operators play a key role in the implementation of the proposed calculation engines.

Therefore, this thesis contributes to the realm of FPGA-based real-time simulation in many ways. The research work proposes a new summation algorithm, which is a generalization of the so-called self-alignment technique. The new formulation is broader, simpler in its expression and hardware implementation. Our research helps formulating criteria to guarantee good accuracy, the criteria being established on a theoretical, as well as empirical basis. Moreover, the thesis offers a comprehensive analysis on the use of the redundant high radix carry-save (HRCS) format. The HRCS format is used to perform rapid additions of large mantissas. Two new HRCS operators are also proposed, namely an endomorphic adder and a HRCS to conventional converter.

Once the mean to single-cycle accumulation is defined as a combination of the self-alignment technique and the HRCS format, the research focuses on the FPGA implementation of SIMD calculation engines using parallel floating-point MACs or DPs. The proposed operators are characterized by low latencies, allowing the engines to reach very low time-steps. The document finally discusses power electronic circuits modelling, and concludes with the presentation of a versatile calculation engine capable of simulating power converter with arbitrary topologies and up to 24 switches, while achieving time steps below 1 μs and allowing switching frequencies in the range of tens kilohertz. The latter realization has led to commercialization of a product by our industrial partner.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

APR	Additionneur à propagation de retenue
BRAM	Bloc RAM : bloc de mémoire gravé en dur dans le FPGA
clk	Abréviation anglaise de *clock* : référence au signal d'horloge
CS	Carry-Save : format redondant utilisant la sauvegarde de la retenue
CAN	Convertisseur analogique à numérique
CLA	Carry Look-ahead : Anticipateur de retenues
CNA	Convertisseur numérique à analogique
CPU	Central Processing Unit : acronyme anglais servant à désigner un PUG, et par métonymie un ordinateur de bureau
dd	Décalage à droite
dg	Décalage à gauche
DP	Dot-product (operator) : acronyme utilisé dans nos publications anglophones pour désigner un OPS
DSP	Digital Signal Processing : traitement de signal numérique
DUT	Device Under Test : dispositif sous test
ECU	Electronic Control Unit : unité de contrôle électronique, acronyme répandu dans le domaine de l'automobile et de l'aérospatial
eHS	acronyme de eHardwareSolver, nom commercial de l'engin de calcul versatile développé dans le cadre de la présente thèse et commercialisé par le partenaire industriel, Opal-RT Technologies
EMT	Electromagnetic Transient : méthode numériques de modélisation des réseaux électriques employant l'analyse nodale
EMTP	Electromagnetic Transient Program : logiciel de simulation des réseaux électriques de type EMT
E/S	Entrées/sorties
FA	Full Adder : additionneur complet
FAA	Format d'auto alignement, associé à la TAA
FLOPS	Floating-point Operations Per Second : mesure de la puissance de calcul évaluée par la quantité d'opérations en virgule flottante pouvant être effectuées en une seconde

FMA Fused Multiply-Adder : multiplieur additionneur fusionnés

FPGA Field Programmable Gate Array : circuit numérique fait de réseaux pré-diffusés programmables

HIL Hardware-in-the-Loop : réfère à une configuration particulière de la simulation en temps réel où du matériel est connecté au simulateur et se retrouve dans la boucle de simulation

HRCS High Radix Carry-Save : sauvegarde de retenues à base élevée

IEEE Institute of Electrical and Electronics Engineers : société professionnelle regroupant les ingénieurs électriciens et électroniciens en charge notamment d'établir les normes industrielles

IGBT Insulated Gate Bipolar Transistor : transistor bipolaire à grille isolée

ISE Logiciel de synthèse et d'implémentation des FPGA de la compagnie Xilinx

LUT Look-Up Table : table des correspondances

GFLOPS GigaFLOPS : un milliard de FLOPS

Matlab Logiciel mathématique de la compagnie Mathworks

MAC Multiply-acumulator : multiplieur-accumulateur

MANA Modified-augmented nodal analysis : analyse nodal modifiée et augmentée

MEE Méthode explicite d'Euler

MIE Méthode implicite d'Euler

MLI Modulation de largeur d'impulsion, PWM en anglais

MMC Modular Multilevel Converter : convertisseur de puissance multi-niveaux

MNA Modified nodal analysis : analyse nodal modifiée

OPS Opérateur de produit scalaire

PMSM Permanent magnet synchronuous motor : moteur à aimant permanent

PUG Processeur à usage général : processeur tels que ceux que l'on retrouve dans les ordinateurs personnels, par opposition aux processeurs à usage spécifiques

PUS Processeur à usage spécifique : processeur spécialisé destiné à l'exécution d'un d'algorithme ou d'une tâche spécifique

PWM Pulse-width modulation, acronyme anglais de MLI

RAM Random access memory : mémoire à accès aléatoire

ROM Read only memory : mémoire à accès aléatoire permettant uniquement les accès en lecture

RCA	Ripple Carry Adder : additionneur à propagation de retenue
slice	Unité reconfigurable des FPGA de Xilinx, tranche en français
SD	Signed-Digit : format redondant utilisant des chiffre signés
SIMD	Single Instruction, Multiple Data : architecture de processeur où des unités de calcul parallèles exécutent les mêmes instructions sur des données différentes
Simulink	Logiciel de simulation numérique de la compagnie Mathworks
SM	Sous-module : cellule d'un convertisseur MMC
SPS	SimPower System : logiciel de simulation des réseaux électriques commercialisé sous la forme d'une boîte à outils Matlab/Simulink
TAA	Technique d'auto alignement, technique permettant d'accumuler des opérandes en virgule flottante avec une boucle d'accumulation à un cycle
TNA	Transient Network Analyzer : simulateur analogique
ulp	Unit in the Last Place : poids du plus petit bit de la mantisse d'une représentation en virgule flottante
VF	Virgule flottante

INTRODUCTION

La simulation en temps réel des réseaux électriques connaît un intérêt industriel marqué. Cet engouement est motivé par la réduction substantielle des coûts de développement qu'une telle approche permet. Pour avoir une idée de la chose, il suffit de penser à des systèmes tels que les véhicules électriques, les aéronefs modernes ou les réseaux de distribution électrique, domaines d'application où la complexité et le coût élevé des systèmes donnent la préférence à la simulation plutôt qu'au prototypage physique. De plus, la simulation en temps réel permet d'intégrer dans la boucle de la simulation du matériel au fur et à mesure de la conception de ce dernier, ce qui permet d'en vérifier le bon fonctionnement dans des condition réalistes. C'est d'ailleurs là une des spécificités du domaine : la simulation en temps réel permet l'adjonction d'un matériel au simulateur dans une configuration dite de matériel dans la boucle (*hardware-in-the-loop* : HIL) que ne permet pas (ou difficilement) une simulation en temps différé.

Durant la première moitié du XX^{e} siècle, la simulation des transitoires dans les réseaux électriques se fait au moyen de simulateurs analogiques appelés TNA (*Transient Network Analyser*). Ces simulateurs résolvaient les équations différentielles par l'interaction d'unités analogiques interconnectées (additionneurs, intégrateurs) ou en recourant à des modèles réduits permettant la reproduction à l'exact, mais à des échelles moindres, des phénomènes à l'étude. Les simulateurs analogiques eurent leur temps de gloire et continuèrent d'être utilisés jusqu'à l'adoption récente des méthodes numériques de simulation que rendit possible l'apparition des calculateurs puissants que sont nos ordinateurs modernes. Progressivement, les simulateurs numériques se développèrent sous la forme de grappes d'ordinateurs interconnectés par des liens rapides, offrant une puissance de calcul confortable et permettant la prise en charge de modèles de plus en plus complexes.

Néanmoins, la simulation en temps réel au moyen de processeurs à usage général (PUG) souffre de certaines limitations inhérentes à l'infrastructure matérielle et aux systèmes physiques à l'étude. D'une part, la puissance de calcul brute des PUG s'avère parfois insuffisante pour l'atteinte d'un pas de calcul de l'ordre de quelques micro-secondes, ce qui est un requis important pour la simulation fidèle des transitoires rapides telles que celles des réseaux commutés à haute fréquence. Une autre limitation provient des liens de communication liant le PUG à ses périphériques et dont la latence intrinsèque ralentit l'accès aux interfaces (senseurs et actuateurs) du dispositif sous test. Cet état de fait est alors répercuté sur la durée totale de la boucle de simulation, rendant impossible la simulation des transitoires rapides.

Pour tenter d'apporter une réponse à ces difficultés, la pratique industrielle a adopté les circuits reconfigurables de type FPGA (Field Programmable Gate Array) pour la mise en œuvre de processeurs à usage spécifique (PUS) dédiés aux applications de simulation des réseaux électriques. Cette approche offre différents avantages : d'une part, les FPGA sont aujourd'hui suffisamment denses pour accueillir des engins de calcul de grande capacité (quelques dizaines de GFLOPS), ce qui laisse espérer que le modèle à simuler peut être déplacé en partie ou en totalité du PUG vers le PUS ; d'autre part, un FPGA peut disposer d'un accès direct aux entrées/sorties (E/S) du simulateur, réduisant considérablement les latences d'accès aux senseurs et aux actuateurs du dispositif sous test et permettant du même coup la simulation de systèmes à dynamiques rapides.

Il convient de relever que l'utilisation des circuits reconfigurables pour des applications de calcul connaît un vif intérêt. Cet intérêt est justifié d'une part par la croissance impressionnante des ressources matérielles disponibles sur ces dispositifs. De plus, le caractère reconfigurable des FPGA leur confère une flexibilité architecturale qui, dans les cas favorables, permet d'atteindre des performances de calcul meilleures que celles offertes par des processeurs commerciaux. Néanmoins, les paradigmes usuels du traitement de signal sur FPGA introduisent certaines restrictions. Ainsi, il est d'usage d'opérer sur FPGA une arithmétique à virgule fixe qui pénalise le temps de développement puisqu'elle requiert du concepteur (qui, dans le cas de figure qui nous intéresse, n'est souvent pas un expert) un redimensionnement de l'ensemble des variables du modèle mathématique. Une façon de contourner ce problème est d'utiliser une arithmétique à virgule flottante. En effet, il est aujourd'hui possible de disposer sur FPGA d'opérateurs à virgule flottante que l'usager peut obtenir grâce aux librairies commerciales du manufacturier de la puce. Cependant, l'emploi de ces opérateurs impose des latences qui peuvent s'avérer handicapantes, particulièrement dans le cas de l'addition car les techniques de modélisation considérées ici emploient des lois d'intégration (trapézoïdale, Euler-arrière, etc.) pour lesquelles l'utilisation d'un accumulateur est cruciale. Or, les additionneurs en virgule flottantes nécessitent en général plusieurs cycles opératoires, restreignant d'une certaine façon la rétroaction directe de la sortie de l'additionneur à une de ses entrées. La problématique de l'addition et de l'accumulation en virgule flottante forme le cœur de notre travail de recherche et deux chapitres de la thèse lui sont consacrés. Quatre objectifs guideront notre démarche dans l'étude de cette problématique :

1. l'exactitude des calculs ;
2. la compacité des opérateurs d'addition ;
3. la célérité de ces derniers en termes de fréquences et de latence ; et
4. la possibilité de disposer d'une boucle d'accumulation à un cycle.

Ce travail permettra notamment d'élaborer des architectures d'accumulateurs, de multiplieurs accumulateurs (MAC) et d'opérateurs de produit scalaire (OPS) en virgule flottante, qui joueront un rôle déterminant dans la mise en œuvre de nos engins de calcul pour la simulation des réseaux électriques.

Les équations de réseau utilisées pour modéliser les systèmes susmentionnés sont généralement formulées suivant deux approches distinctes : *i*) les équations d'état ; et *ii*) l'analyse nodale. Les deux méthodes sont considérées dans le présent travail. Leur solution repose sur les techniques de résolution des systèmes d'équations différentielles linéaires ou linéaires par parties. Nous démontrons ici que celles-ci s'expriment sous la forme d'un produit matrice-vecteur contraint temporellement sur la rétroaction des états du circuit. Ce qui est entendu par la contrainte temporelle est l'importance d'actualiser au plus vite les variables du système avant de débuter un nouveau pas de temps. La méthode des équations d'états retiendra notre attention surtout pour la modélisation des moteurs électriques d'une part, et du fait qu'elle est au cœur de la librairie SimPowerSystem (SPS) qu'exploite dans ses produits le partenaire industriel de la présente thèse, la compagnie montréalaise Opal-RT Technologies. La méthode nodale quant à elle est caractérisée par la simplicité de sa formulation (facilitant d'autant son automatisation) et sera utilisée dans notre travail pour la modélisation des réseaux à interrupteurs. Nous présentons à cette fin une réécriture des équations de la méthode nodale pour alléger la contrainte sur le chemin de données de nos engins de calcul.

Étant données les contraintes temporelles susmentionnées, il est d'usage de pré-calculer la solution des équations du réseau ou, dans le cas de circuits à interrupteurs, les solutions des différentes topologies du réseau que ces interrupteurs induisent. Ce dernier point nous poussera à étudier des approches permettant de réduire la quantité de mémoire nécessaire à la simulation sur puce des systèmes à l'étude. Il motivera également la recherche de solutions permettant de repousser les limites usuelles imposées aux problèmes considérés dans la simulation en temps réel sur FPGA.

Ainsi, l'implémentation matérielle d'engins de calcul en virgule flottante implique plusieurs défis architecturaux qu'il convient d'étudier et d'évaluer adéquatement. Les paramètres à considérer sont :

1. La taille des ressources reconfigurables disponibles (slices, blocs DSP, blocs RAM) ;
2. La taille de la mémoire embarquée (blocs RAM) nécessaire pour simuler un problème donné et la manière de l'exploiter ;
3. La latence des opérateurs arithmétiques (MAC/OPS) réalisés et leur impact sur la latence globale des engins de calcul ;

4. L'exactitude des calculs, tant au niveau unitaire des opérateurs (MAC/OPS) qu'au niveau système ;
5. Les techniques de modélisation des réseaux électriques, et plus particulièrement des réseaux à interrupteurs.

Notre travail de thèse offre différentes contributions scientifiques que nous pouvons énumérer comme suit :

1. Au domaine de l'arithmétique des ordinateurs, nous avons contribué à la formulation d'un algorithme de sommation qui est une généralisation de la technique d'autoalignement précédemment publiée dans la littérature. Nous avons à ce titre simplifié son expression et sa réalisation matérielle, établi des critères permettant de garantir la bonne exactitude de ses résultats, critères que nous avons énoncés en les appuyant de démonstrations théoriques et empiriques.
2. Au domaine de l'arithmétique des ordinateurs également, nous avons contribué à l'analyse du format redondant *high radix carry-save* (HRCS) et à l'étude de son implémentation sur FPGA. Nous avons également contribué à définir deux nouveaux opérateurs au format HRCS. D'une part, nous avons défini un additionneur endomorphique, prenant en entrée deux opérandes en HRCS et produisant un résultat en HRCS. Ce résultat permet de réaliser des arbres d'additionneurs recevant des nombres de grande largeur (80 bits par exemple) et de les sommer avec une latence réduite. Le second opérateur proposé est un convertisseur HRCS à format conventionnel qui permet de convertir un résultat en HRCS en un format binaire standard. L'avantage de cet opérateur est qu'il a peu d'impact sur le chemin critique de l'additionneur et garantit une latence quasi-constante (un à trois cycles) pour une grande gamme de largeurs du format des nombres (jusqu'à 1024 bits).
3. Au domaine de la simulation en temps réel sur FPGA, nous avons contribué en unifiant l'écriture des équations du réseau, qu'elles soient exprimées selon l'analyse nodale ou les équations d'états. Cette réalisation a permis de formuler une architecture d'engins de calcul de type SIMD (*single instruction, multiple data*) faits à base d'opérateurs MAC et OPS en virgule flottante.
4. Nos contributions au domaine de l'arithmétique des ordinateurs ont permis de réaliser des opérateurs arithmétiques en virgule flottante complexes tels que MAC et OPS qui sont à la base des engins de calcul que nous proposons pour la simulation en temps réel des équations du réseau.
5. Finalement, nos travaux ont permis la réalisation d'un engin de calcul versatile permettant de simuler des convertisseurs de puissance de topologie arbitraire avec des pas

de temps en deçà du 1 μs, tout en admettant des fréquences de commutations de plusieurs kilohertz. Cette réalisation a abouti une commercialisation par notre partenaire industriel.

La présente thèse est organisée comme suit. Le Chapitre 1 propose une revue de la littérature touchant le domaine de la simulation en temps réel, de la simulation sur FPGA, des méthodes de modélisation des circuits à interrupteurs et des techniques rattachées à la représentation en virgule flottante sur FPGA. Le Chapitre 2 s'attaque ensuite à la problématique de la simulation sur FPGA. Les méthodes nodales et des équations d'états y sont présentées, ainsi que les méthodes proposées pour réécrire leur solution sous la forme d'un produit matrice-vecteur. L'analyse du partitionnement d'un tel problème sur plusieurs opérateurs arithmétiques est alors détaillée, démontrant le rôle de l'accumulateur à un cycle, celui du MAC et le rôle primordial joué par l'OPS. Le Chapitre 3 traite de la technique d'auto-alignement (TAA) qui fut proposée au début des années 2000 pour l'accumulation à un cycle d'opérandes en simple précision. Le chapitre en propose une reformulation et la généralisation de la portée algorithmique au format en double précision et aux formats intermédiaires, tout en identifiant les conditions garantissant la qualité de ses résultats. Le Chapitre 4 quant à lui traite du format HRCS, utilisé dans notre travail pour implémenter efficacement la TAA sur FPGA. Le Chapitre 5 exploite les résultats des trois chapitres précédents pour implémenter des engins de calcul destinés à la simulation en temps réel de circuits à interrupteurs modélisés au moyen de l'analyse nodale modifiée et augmentée (*modified-augmented nodal analysis* : MANA). Ce chapitre se termine avec la présentation d'un engin de calcul versatile permettant de simuler des convertisseurs de topologie arbitraire ayant jusqu'à 24 interrupteurs et qui est actuellement commercialisé par le partenaire industriel de cette thèse sous le nom eHS (eHardwareSolver).

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Introduction

La revue de littérature présentée ici tente de faire un point sur l'état de l'art de la simulation des réseaux électriques sur FPGA. Dans un premier temps, nous traiterons des techniques de simulation des réseaux et tâcherons de relever les spécificités associées à la simulation en temps réel. Il est entendu ici par simulation des réseaux la simulation de systèmes aussi variés que les moteurs électriques, les convertisseurs de puissance ou les réseaux de distribution électriques. Cet examen nous portera à traiter des techniques de simulation des réseaux sur FPGA et des considérations qui leur sont propres. Notre revue de littérature portera finalement sur l'emploi de la virgule flottante sur FPGA, où nous veillerons à faire ressortir les paradigmes émergents dans ce domaine, notamment l'utilisation de formats non standards. Cette revue de littérature sera complétée dans les chapitres subséquents afin de couvrir plus en détail certains sujets plus pointus et de conférer au présent chapitre un caractère plus général.

1.2 Contexte

1.2.1 Simulation des réseaux électriques

À partir des années 1930, la simulation des transitoires dans les réseaux électriques se fait au moyen des simulateurs analogiques appelés TNA. Un simulateur TNA est réalisé au moyen de modèles réduits du réseau électrique où les lignes de transmissions sont émulées par des sections en π. Les TNA sont suffisamment sophistiqués pour permettre l'évaluation de scénarios de court-circuit, de fautes ou de soudaine perte de charge. Un de leurs avantages majeurs est leur fonctionnement en temps réel puisqu'il permet une évaluation rapide de différents scénarios de fonctionnement des systèmes à l'étude, sans compter la possibilité d'y connecter physiquement des contrôleurs aux fins de leur validation empirique [73, 115].

L'avénement des ordinateurs numériques et la constante croissance de leurs capacités de calcul fit en sorte que les simulateurs numériques ont progressivement supplanté les TNA dans la pratique industrielle. L'émergence de la simulation numérique des réseaux électriques est souvent attribuée à l'article de Dommel [20] qui appliquait la discrétisation des éléments à l'analyse nodale du réseau. L'analyse nodale recourt à la relation courant-tension aux divers

noeuds du circuit en utilisant la discrétisation numérique des divers composants du circuit. Du point de vue du circuit, cela équivaut à recourir à l'équivalent Norton de l'ensemble des composants du réseau pour ensuite déterminer les tensions inconnues aux différents noeuds. On appelle généralement EMTP (*Electromagnetic Transients Program*) les logiciels utilisant les méthodes numériques pour la simulation des réseaux électriques. De nombreux logiciels de type EMT existent, un des plus connus étant EMTP-RV, développé sous les auspices d'Hydro-Québec sur la base des travaux du Professeur Mahseredjian [72].

Vers la fin des années 1990 et parallèlement au développement d'EMTP-RV, Mathworks commercialisait une boîte à outils Simulink/Matlab appelée SimPowerSystems (SPS), et dont le but est la simulation des réseaux de taille modeste. SPS exploite la représentation d'états pour la modélisation des réseaux électriques. Le chapitre 2 traitera plus en détail de ces deux méthodes de modélisation des réseaux et de leur utilisation potentielle sur FPGA.

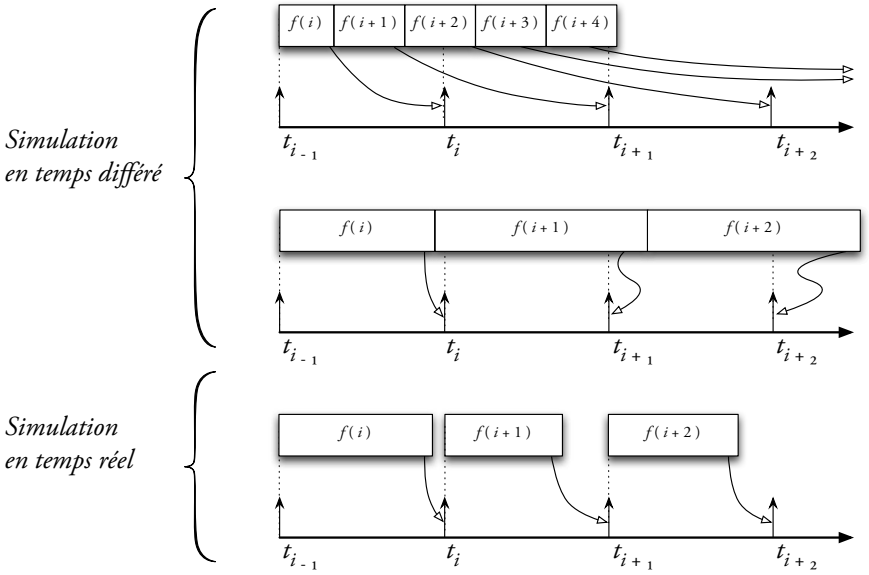

Figure 1.1 – Différence entre la simulation en temps différé et la simulation en temps réel.

1.2.2 Simulation en temps réel

Un des avantages majeurs des TNA est leur fonctionnement en temps réel qui permet d'y adjoindre des contrôleurs physiques. Les méthodes de simulation numériques ne sont pas forcément exécutées en temps réel et peuvent tout autant se concevoir dans le cadre d'une simulation en temps différé. Si l'on se fie à la Figure 1.1, il apparaît qu'une simulation en temps différé produit les états du système à chaque fois qu'une itération termine, ce qui signifie que la simulation peut être soit plus rapide que le temps réel (cas où les calculs ne

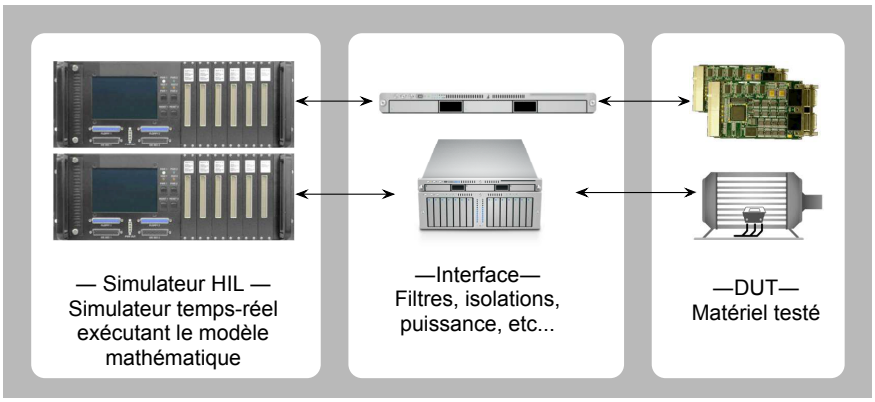

Figure 1.2 – Configuration typique d'un simulateur HIL.

sont pas trop longs), soit plus lente que le temps réel (cas de charge de calcul importante). Dans le cas d'une simulation numérique en temps réel, il est admis que le temps de calcul associé à chaque itération ne doit pas être plus long que le pas de calcul lui-même. De plus, le début de chaque itération est synchronisé sur le début du pas de calcul.

La simulation numérique en temps réel a connu un essor important durant les dix à vingt dernières années, c'est-à-dire depuis l'avénement de simulateurs à base de grappes d'ordinateurs personnels qui permirent d'en réduire le coût tout en offrant des puissances de calcul confortables [68, 90]. Étant donné que ce type de simulateurs sert principalement à interfacer du matériel à des fins de tests et de validation, les simulateurs en temps réel sont souvent appelés simulateurs HIL (acronyme anglais de *Hardware in the Loop*). La Figure 1.2 présente la configuration typique d'un simulateur HIL, où une station de travail est connectée à un dispositif sous test (DUT). Le dispositif sous test peut tout autant être un contrôleur électronique ou un dispositif de puissance (par exemple un panneau solaire). Le dispositif sous test est connecté au simulateur au travers d'une interface d'isolation ou de conditionnement de signal.

Les simulateurs HIL sont largement employés dans les industries automobile et aérospatiale pour valider le bon fonctionnement d'unités de contrôle appelées ECU (*Electronic Control Unit*) sur un modèle mathématique du réseau. L'objectif principal d'une simulation HIL est la réduction des coûts de conception en effectuant des tests sur le prototype réel durant son développement dans des conditions réalistes d'opération [68, 73]. La simulation HIL permet également une couverture de test la plus large possible, incluant les cas limites d'opération qui seraient dangereux voire impossibles à réaliser sur du vrai matériel (cas de faute phase-phase sur une des bornes du moteur, par exemple) [24].

1.2.3 Utilité des FPGA dans les simulateurs HIL

Durant les dix dernières années, les FPGA ont fait leur apparition sur les simulateurs HIL comme une composante importante. Les FPGA sont adjoints à ces simulateurs pour servir divers objectifs :

1. Offrir au modèle simulé sur le PUG un accès aux entrées/sorties (E/S), tant analogiques que numériques ;
2. Servir d'interface d'accès aux cartes de conversion analogiques (CAN/CNA) du simulateur qui emploient généralement des protocoles de communication sérielle tel que SPI ou I^2C ;
3. Générer des fonctions spécialisées telle que des sinus, des signaux de résolveur ou des signaux de modulation de largeur d'impulsion (MLI) ;
4. Détection des événements inter-pas (gâchette de contrôle s'ouvrant ou se refermant au milieu du pas de calcul) pour les besoins de certaines méthodes d'interpolation [23].

La littérature indique qu'il est aujourd'hui envisageable de déplacer, en tout ou en partie, la charge de calcul depuis les CPU du simulateur vers le FPGA [75, 84]. Nous discutons à la Section 1.4 de cette problématique qu'on appellera «la simulation sur puce ».

1.3 Techniques propres à la simulation en temps réel

Les contraintes physiques et temporelles auxquelles est soumise la simulation en temps réel nécessitent certains artifices de modélisation qui méritent d'être considérés plus en détail. Cette section propose de revoir les techniques les plus souvent employées dans le domaine et qui nous seront d'une grande utilité dans le reste du travail. Ce faisant, la section se propose également de brosser un portrait de l'état de l'art du domaine.

1.3.1 Réseaux à interrupteurs

Il est fréquent dans la pratique industrielle de considérer des réseaux à interrupteurs, notamment dans la modélisation des convertisseurs de puissance. Ces réseaux sont de plus en plus présents dans le domaine des réseaux électriques et se retrouvent tout autant dans les systèmes à faible ou moyenne tension que les système à très haute tension. Un réseau commuté pose des contraintes de simulation du fait que la topologie du réseau est modifiée à chaque fois qu'un interrupteur change de statut (ouvert/fermé). Il est généralement admis que la simulation de réseaux commutés peut suivre deux modes distincts : le mode détaillé et le mode comportemental [54]. On retrouve le mode détaillé dans des logiciels tels que

SPICE ou EMTP-RV où le modèle de l'interrupteur est une fonction non linéaire analytique ou linéaire par segments. Ce mode de simulation est très demandant en temps de calcul et se prête mal à la simulation en temps réel. Le mode comportemental quant à lui est consubstantiel à la simulation HIL et se caractérise par une complexité de calcul modérée.

Trois modèles d'interrupteur produisent use simulation en mode comportemental [54] : 1) le modèle idéal ; 2) le modèle de fonction de commutation ; 3) le modèle moyen. Dans le contexte de la simulation en temps réel, la littérature rapporte l'utilisation de chacun de ces modèles pour la modélisation de circuits d'électroniques de puissance. On relèvera cependant que le modèle moyen ne donne aucune information sur le comportement détaillé des transitoires et s'intéresse simplement aux niveaux de tension ou de courant correspondant au fonctionnement au niveau système d'un convertisseur de puissance [95]. Le modèle de fonction de commutation permet au contraire d'atteindre un niveau de détail suffisamment appréciable, avec souvent un niveau de résolution de l'ordre de quelques nanosecondes [24]. Ce modèle est cependant très dépendant de la topologie du convertisseur, ne considère que des convertisseurs avec un nombre d'interrupteurs modeste (moins de 8 [1]) et peut difficilement se généraliser. Il reste finalement le modèle idéal. Ce dernier se traduit soit par un interrupteur idéal, soit par une résistance binaire (R_{on}/R_{off}). Un tel modèle offre différents avantages, notamment de permettre la simulation des cas de faute de court-circuit ou de circuit ouvert [49]. Cependant, un tel modèle nécessite la reformulation des équations du réseau pour chaque combinaison des statuts des interrupteurs [25, 110], ce qui implique généralement de mémoriser les équations de chacune des topologies ainsi induites, limitant de ce fait le nombre d'interrupteurs à 6 ou 7. De plus, dès lors que le circuit inclut des dispositifs à commutation naturelle (des diodes par exemple), la simulation en temps réel nécessite que ce modèle soit adjoint d'une fonction de commutation. Ainsi, le problème retourne à la restriction du modèle d'interrupteur à mode comportemental, soit une limite sur les différentes topologies pouvant être considérées due à la nécessité d'opérer une étude relativement élaborée des différents modes de fonctionnement du convertisseur [49].

Au début des années 1990, un effort conjoint de deux équipes de recherche a abouti à la formulation d'un modèle d'interrupteur qui connaît un regain d'intérêt dans le domaine de la simulation en temps réel [45, 47, 94]. Ce modèle procède en deux temps. D'une part, il choisit de modéliser l'interrupteur soit comme une petite capacité C_{sw} (pour indiquer que l'interrupteur est ouvert), soit comme une petite inductance L_{sw} (pour indiquer que l'interrupteur est fermé). En choisissant adéquatement les valeurs de C_{sw} et L_{sw}, il est possible d'obtenir des équations du réseau totalement indépendantes des statuts des interrupteurs, apportant ainsi une solution aux limitations de mémoire évoquées précédemment. D'autre

1. Cette limite sera justifiée au Chapitre 5.

part, ce modèle d'interrupteur est accompagné d'une règle de mise à jour du statut de l'interrupteur pour différents type de semi-conducteurs [94] exprimée de manière relativement simple puisqu'elle ne dépend que de la gâchette, du statut courant de l'interrupteur et des signes du courant et de la tension aux bornes du dispositif. Ce modèle a été largement utilisé et constitue l'approche algorithmique principalement adoptée dans notre travail de thèse. Il convient cependant de mentionner que cette approche souffre de certaines limitations que nous aurons soin de considérer plus en détails au Chapitre 5.

1.3.2 Découplage du réseau

Le découplage du réseau est une technique qui consiste à découper le réseau en deux ou plusieurs parties dans le but d'atteindre différents objectifs. D'une part, la charge de calcul nécessaire à la simulation d'un réseau complet est souvent supérieure à la charge de calcul que nécessite la simulation de ses multiples sous-réseaux. Dans le cas d'une simulation en temps réel, cet allègement de la charge de calcul par le découpage du réseau peut permettre une réduction éventuelle du pas de temps de la simulation, améliorant par conséquent la précision de cette dernière. D'autre part, le découplage simplifie la parallélisation du calcul sur plusieurs processeurs. Le parallélisme peut tout autant se considérer dans le contexte de simulateurs à base de grappes d'ordinateurs que de simulateurs à base de processeurs multi-coeurs. Un autre intérêt du découplage, dans le cas des réseaux commutés tels les convertisseurs de puissance, provient du fait que le découplage du réseau permet de circonscrire des sous-ensembles d'interrupteurs et de réduire la quantité de mémoire nécessaire pour conserver les équations du réseau.

Une fois le découplage du réseau obtenu — et en supposant une exécution parallèle de la simulation —, il est établi que le pas de temps de la simulation est au minimum le pas de temps le plus lent des pas des sous-réseaux. De plus, tous les processeurs sont synchronisés par un processeur maître qui se charge de cadencer la simulation [73]. Pour ce faire, il faut que tous les échanges d'information entre les processeurs soient conclus avant de débuter un nouveau pas de temps. La durée de la communication dépend de la technologie adoptée : les communications inter-processeurs au travers d'une mémoire partagée (aujourd'hui disponibles sur les ordinateurs de bureau) étant plus rapides que celles exploitant des liens de communication standards telles que PCIe ou Infiniband [90]. Il en ressort que les meilleurs pas de calcul que l'on puisse observer durant une simulation en temps réel avoisinent les 2 à 3 μs [2], mais elles ne sont possibles qu'au moyen d'une mémoire partagée et excluent la présence de matériel dans la boucle. Aussi, dès lors qu'on est en présence de signaux E/S

2. Cet énoncé s'appuie sur l'expérience que l'auteur a eue avec les outils de simulation de la compagnie partenaire de la thèse, Opal-RT Technologies.

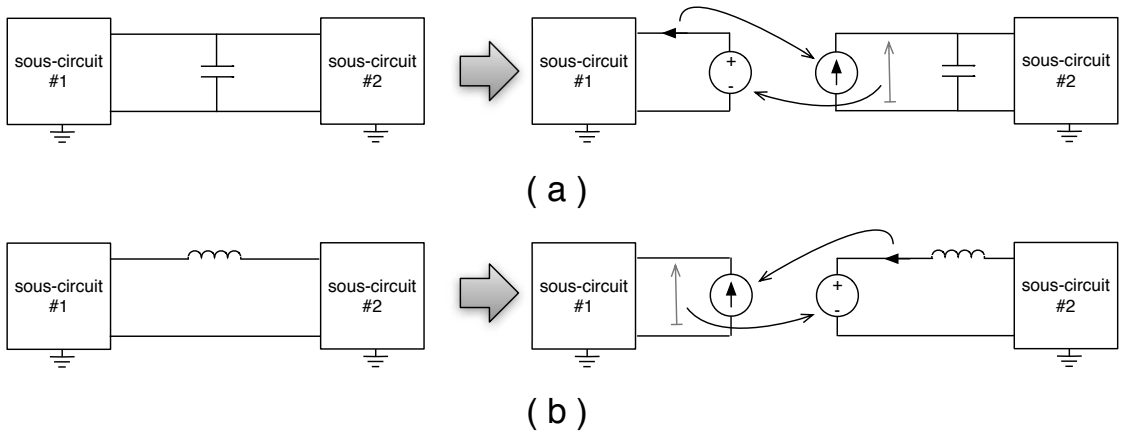

Figure 1.3 – Découplage d'un circuit en deux parties distinctes au travers d'un composant (a) capacitif ; (b) inductif.

interfaçant du matériel, la littérature rapporte des pas de calcul qui sont de l'ordre de 5 à 10 μs au mieux [74, 91].

Le découplage du réseau est traditionnellement effectué au niveau des lignes de transmission. Dans ce cas, la ligne de transmission joue un rôle de délai que peut répliquer avantageusement le délai de la communication [42]. Cette technique est très répandue et couramment utilisée dans la simulations des réseaux de transport de l'énergie électrique, mais elle se prête plus difficilement aux convertisseurs de puissance que l'on retrouve dans des réseaux de moyenne tension.

Une autre technique de découplage consiste à diviser le circuit aux points de variation lente de la tension ou du courant, c.-à-d. les capacités ou les inductances du circuit [46]. Les deux parties du circuit découplé retrouvent de part et d'autre l'information relative à l'autre moitié au moyen d'une source de courant et d'une source de tension, respectivement [49]. Ce procédé est illustré à la Figure 1.3. Contrairement aux techniques de découplage par lignes de transmission, ce type de découplage n'est qu'une approximation du véritable comportement du réseau. Des techniques d'interpolation peuvent être exploitées pour raffiner le résultat [58], mais elles ne sont envisageables que dans le cas d'une simulation en temps différé.

La littérature rapporte également des techniques de déchirement du réseau (*network tearing*) dites diakoptiques [11]. Des variantes de ces techniques ont été récemment proposées et implémentées avec succès dans un contexte de simulation en temps réel [26, 114]. Cependant, ces techniques sont difficilement envisageables sur FPGA car elles requièrent l'inversion de matrices, limitant considérablement le pas de calcul de la simulation et le champ d'application de la technologie.

1.4 Simulation sur FPGA

Une des grandes difficultés rencontrées dans la simulation en temps réel provient de la non-linéarité du réseau en présence des discontinuités dues aux commutations des interrupteurs. Comme nous avons pu le voir à la Section 1.3.1, différentes approches existent pour répondre à cette question. Dans les industries de l'automobile et de l'aérospatiale (très demanderesses de la méthodologie de prototypage HIL), la simulation en temps réel des convertisseurs de puissance nécessite des pas de calcul inférieurs à la microseconde. De tels contraintes temporelles sont dues aux fréquences de commutations de ces convertisseurs qui se trouvent dans l'intervalle des 10 kHz à 200 kHz [38, 123]. Aussi, si l'on se fie aux directives de bonne pratique admises par la communauté scientifique [35], les pas de calcul devraient être 10 à 20 fois plus petits que la période de la fréquence de commutation. Cela conduit à des pas de calcul dans l'intervalle de 0.25 μs à 5 μs. Comme nous avons pu le mentionner précédemment, de tels pas de calcul sont inconcevables dans un contexte de simulation sur CPU. C'est ainsi que les FPGA sont devenus de plus en plus utilisés dans le domaine de la simulation HIL, avec des pas de calcul rapportés dans l'intervalle 0.1-1 μs [97, 100].

Ainsi, la croissance de la taille des FPGA durant les récentes années a rendu la simulation sur puce une solution à la fois envisageable et attrayante. Elle suscite de ce fait un important effort de recherche, comme le démontre la riche littérature s'y rapportant [2, 75, 74, 98]. La méthode de modélisation des interrupteurs à matrice constante [45, 47, 94] y a trouvé un terreau fertile [13, 66, 75, 85] pour des raisons évidentes de rareté de la mémoire embarquée. Nous reviendrons à cette technique avec plus de détails au Chapitre 5.

1.4.1 Représentation des nombres réels sur FPGA

Une des difficultés imposées par la simulation sur FPGA provient de la représentation des réels. La représentation en virgule fixe est naturelle dans ce contexte, mais elle impose des limitations sur les pas de calcul dues à des raison de quantifications [75]. Il est possible d'éviter ces écueils en utilisant une approche de modélisation sans unité (p.u.) [24, 52]. Cette solution demeure néanmoins fastidieuse tant du point de vue de la conception que du point de vue de l'entretien et du déverminage. Le format à virgule flottante offre un potentiel intéressant. Avec la disponibilité des ressources matérielles sur les FPGA modernes, l'idée de recourir à la représentation en virgule flottante devient réaliste. Dans le contexte de la simulation des réseaux électriques, on mentionne une telle implémentation aussi tôt qu'en 2004 [52], où une approche mixte (combinant virgule flottante et virgule fixe) était utilisée pour la simulation d'un moteur d'induction. Aujourd'hui, les exemples d'implémentation en virgule flottante sont légion [2, 10, 74, 98], démontrant l'intérêt pour cette solution.

1.4.2 Opérateurs arithmétiques en virgule flottante

Afin de comprendre les enjeux portés par l'utilisation de la virgule flottante sur FPGA, nous allons procéder à un bref survol des concepts importants qui y prévalent. Nous aurons donc à cœur de discuter des quelques généralités qu'il sied de connaître, les librairies d'opérateurs à la disposition du concepteur ainsi que les paradigmes émergents.

La représentation des nombres en virgule flottante est une convention qui consiste à représenter le réel x par $x = (-1)^s \cdot 2^e \cdot m$, où s est le signe de x, m sa mantisse et e son exposant. En normalisant la mantisse ($1 \leq m < 2$), la représentation de x en virgule flottante devient unique. Le norme du format à virgule flottante et son utilisation dans les machines ordinées sont encadrés par la norme IEEE-754 [48]. Les architectures des opérateurs de base sont largement couverts dans les ouvrages spécialisés [29, 93]. Ainsi, la norme admet au moins deux formats pour la représentation binaire de la virgule flottante, soit la simple précision (8 bits pour l'exposant, 24 bits pour la mantisse) et la double précision (11 bits pour l'exposant, 53 bits pour la mantisse). La quadruple précision (15 bits d'exposant, 113 bits de mantisse) a récemment été normalisée dans la révision 2008 de la norme [48].

La construction matérielle des opérateurs en virgule flottante suit le plus souvent le schéma suivant :

1. Dépaquetage des différents éléments (signe, exposant et mantisse) des opérandes entrants vers un format de traitement interne ;
2. Réalisation de l'algorithme de l'opération requise (addition, multiplication, etc.) ;
3. Empaquetage du résultat vers le format standard après normalisation et arrondi.

La réalisation de l'addition et de la multiplication sont relativement évidents. L'implémentation de la division ou de la racine carrée requiert plus de soins [1, 63, 67]. La norme IEEE de l'arithmétique flottante [48] impose aux opérateurs arithmétiques de base ($+$, $-$, $\times$, $\div$) de garantir une erreur absolue inférieur à $\frac{1}{2} ulp(x_{fp} \odot y_{fp})$, où $\odot$ est une opération arithmétique de précision infinie, et où $ulp(\cdot)$ est la fonction *unit in the last place* (ulp) [77].

La littérature rapporte diverses implémentations d'opérateurs de base pour la virgule flottante sur FPGA [53, 57]. Certaines implémentations académiques sont ainsi librement distribuées [15]. Il est aussi possible d'utiliser les librairies commerciales fournies par les compagnies de FPGA (telles que Xilinx ou Altera) puisqu'elles sont optimisées pour les FPGA de leur fabrication. Malheureusement, les librairies commerciales, bien que libres de droits, ne sont pas ouvertes, et il n'est pas possible de connaître le détail d'implémentation de leur technologie.

1.4.3 Tendances émergentes

L'idée qui consiste à affirmer qu'il est préférable de ne pas implémenter sur FPGA des opérateurs en virgule flottante complètement conformes à la norme de l'IEEE s'impose dans la littérature et dans la pratique industrielle comme une règle [14, 60]. On entend par là qu'il n'est pas utile, par exemple, d'implémenter les nombres dénormalisés [3], puisque la perte potentielle de précision occasionnée par ce choix architectural peut être compensée par l'ajout d'un bit supplémentaire à l'exposant, et ce au bénéfice d'une simplification du circuit. D'ailleurs, l'utilisation d'opérateurs customisés et d'un format non standard sur FPGA pour la simulation des réseaux est une avenue de recherche qui s'est avérée fructueuse [87].

Mentionnons également l'approche de fusion d'opérateurs. Il s'agit d'une approche consistant à optimiser le chemin de données d'une architecture DSP par l'élimination des étages d'empaquetage et de dépaquetage des opérateurs intermédiaires. Cette approche est actuellement au coeur d'un produit commercial (*datapath compiler*) en développement chez Altera [64, 65]. Kulisch a démontré qu'une telle approche pouvait améliorer la précision globale de l'opérateur fusionné puisqu'elle élimine les erreurs d'arrondis intermédiaires [62].

Relevons finalement l'idée proposée par Flynn de changement de base dans l'implémentation des opérateurs en virgule flottante [31, 32] que Jabiripur a récemment appliquée avec succès [51]. Cette approche a fait ses preuves dans le domaine de l'implémentation des opérateurs en virgule flottante sur FPGA où elle profite des circuit dédiés à la propagation de la retenue [7, 9].

1.4.4 Accumulation en virgule flottante

Une des difficultés dans l'utilisation de la virgule flottante pour la simulation en temps réel réside dans la latence importante des opérateurs arithmétiques, et plus particulièrement celle de l'additionneur. En effet, la modélisation des circuits électriques requiert l'implémentation d'une loi d'intégration (la méthode trapézoïdale par exemple) qui nécessite une accumulation de plusieurs valeurs réelles. Or, les additionneurs en virgule flottantes requièrent plusieurs cycles opératoires, ce qui exclut d'une certaine façon la rétroaction directe de la sortie de l'additionneur à son entrée telle qu'illustrée par la Figure 1.4.

Une approche simple consiste à effectuer l'accumulation en virgule fixe. C'est l'idée exploitée dans [10, 74] et, de façon plus générale, l'approche promulguée par [18]. Néanmoins, la gamme dynamique des nombres en virgule flottante ne peut être couverte correctement que si la largeur du registre d'accumulation est suffisamment importante (l'accumulateur

3. Les nombres dénormalisés permettent d'agrandir l'éventail des nombres représentés en admettant une mantisse dite dénormalisée ($0 < m < 1$) lorsque l'exposant (biaisé) est nul.

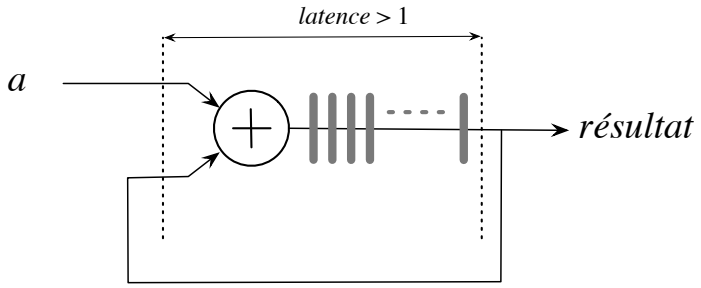

Figure 1.4 – Problème posé par la latence de l'addition dans l'algorithme d'intégration.

dans [10], par exemple, est large de 140 bits), ce qui peut être pénalisant du point de vue de la performance.

Une autre technique pour résoudre le problème de l'accumulation consiste à recourir à des techniques d'entrelacement des opérations par un ordonnancement adéquat des opérations envoyées à un additionneur en virgule flottante. On retrouve cette technique dans le simulateur de moteur à inductance dans [52] et plus récemment (dans un contexte d'application plus général) dans [43, 44, 108, 124]. Le principal problème avec cette approche tient dans le fait qu'elle est dépendante de l'application et reste difficile à employer dans un cadre général. La raison est que, si la quantité d'opérations pouvant être entrelacées est inférieure à la latence de l'additionneur, l'opérateur d'accumulation est sous-exploité.

Une dernière technique est celle proposée dans [69] et adaptée dans [116] dans la réalisation d'un multiplieur-accumulateur (MAC). Son principe consiste à aligner l'ensemble des mantisses des opérandes entrants par rapport à la même limite, suivant les cinq bits supérieurs de leur exposant. Cette approche a été implémentée avec succès sur FPGA dans [89], ainsi que par nos soins [82, 83]. Ces travaux sont couverts aux chapitres 3 et 4.

1.5 Conclusion

Ce premier chapitre a permis de faire un tour d'horizon de la littérature pertinente à notre sujet de recherche, et de pointer vers les avenues que nous avons retenues tout au long de notre travail. Cette revue a cependant tâché de rester assez générale et sera complétée dans les chapitres subséquents lorsqu'un sujet pointu nécessitera l'éclairage scientifique adéquat. Ainsi, la littérature que nous aurons omise pour des raisons d'allégement du texte et de lisibilité sera discutée dans un contexte plus séant et en lui réservant l'espace nécessaire.

CHAPITRE 2

ANALYSE DU PROBLÈME

2.1 Introduction

La conception sur FPGA d'un engin de calcul aux fins de la simulation en temps réel du réseau nécessite la conception d'un PUS pour la résolution des équations différentielles du circuit. De ce qui précède, et étant données les contraintes temporelles associées à ce PUS, il convient de considérer que : 1) un précalcul adéquat de certaines parties du modèle mathématique est de mise pour l'atteinte de haute performance, notamment par l'inversion *a priori* des équations du réseau ; 2) l'utilisation d'opérateurs à virgule flottante customisés et l'utilisation d'un format en virgule flottante non standard offrent une solution élégante et puissante au problème considéré [10, 74] ; 3) la régularité du calcul est une clé essentielle pour obtenir un parallélisme efficace.

Le chapitre qui suit tente de couvrir ces sujets en proposant différentes analyses des équations du réseau et une étude de la mise en œuvre de ces dernières sur FPGA. Aussi, le chapitre débute par une analyse de la manière dont les équations d'état peuvent être réécrites à cette fin avant d'appliquer la même méthode à l'analyse nodale du circuit. Le chapitre propose ensuite une étude permettant de caractériser l'impact des opérateurs sur la performance d'un engin de calcul pour la résolution des équations en temps réel, ainsi que de l'intérêt de disposer d'un opérateur de produit scalaire (OPS) qui sera l'objet de discussion des chapitres 3 et 4.

2.2 Équations du réseau

Dans ce travail, nous considérons tout autant les équations d'états que l'analyse nodale. Ce choix est dicté par le cadre industriel de la thèse et les besoins de la compagnie partenaire. Dans ce qui suit, nous proposons des méthodes de réécriture des équations du réseau pour faciliter leur exécution sur FPGA. Comme nous le verrons, la réécriture des équations mène à un formalisme où les deux méthodes sont confondues. Cette unification des deux méthodes de description des équations du réseau nous permettra de nous concentrer sur les opérateurs arithmétiques nécessaires pour la résolution d'un problème de multiplication matrice-vecteur.

2.2.1 Approche des équations d'états

L'approche des équations d'état est la méthode de choix pour les systèmes électromécaniques et la mise en œuvre de contrôleurs car elle permet notamment l'analyse des vecteurs propres du système [40]. Elle consiste en la réécriture des équations différentielles du réseau (de haut ordre) en un système d'équations de premier ordre. L'expression générale d'un système d'états Σ peut être exprimée ainsi :

$$\Sigma \equiv \begin{cases} \dot{\mathbf{x}}_t &= \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{u}_t \end{cases} \tag{2.1}$$

où $\mathbf{x}_t$, $\mathbf{u}_t$ et $\mathbf{y}_t$ sont respectivement les vecteurs d'états, d'entrée et de sortie du système ; $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ et $\mathbf{D}$ sont respectivement les matrices d'états, d'entrée, de sortie et d'action directe. D'autres écritures existent mais nous retiendrons celle-ci puisqu'elle se retrouve telle quelle dans la librairie SPS exploitée dans le présent travail. Dans le cas où les matrices sont dépendantes du temps, le système est dit variant, autrement le système est dit invariant.

Le système d'équations Σ peut être résolu dans le temps après discrétisation. Différentes techniques de discrétisation (à pas fixe, à pas multiples, explicite ou implicite) existent. Le choix d'une méthode plutôt qu'une autre dépend de différents paramètres tels que la stabilité et l'exactitude de la méthode [22]. Dans le contexte de la simulation en temps réel, les méthodes à pas fixe (telle que la méthode du trapèze) sont généralement préférables car elles réduisent l'effort de calcul. Une précision adéquate peut être garantie lors de la mise en œuvre de méthodes directes si le pas de calcul est suffisamment petit.

Si le pas de calcul Δt est substantiellement plus petit que les constantes de temps du système, $\dot{\mathbf{x}}_t$ peut être considéré comme constant par morceaux, de sorte que la méthode explicite d'Euler (MEE) puisse être considérée, ce qui mène au système d'équations $\Sigma^{\Delta t}_{\text{mee}}$:

$$\Sigma^{\Delta t}_{\text{mee}} \equiv \begin{cases} \mathbf{x}_{n+1} &= \mathbf{A}_d\mathbf{x}_n + \mathbf{B}_d\mathbf{u}_{n+1} \\ \mathbf{y}_{n+1} &= \mathbf{C}_d\mathbf{x}_n + \mathbf{D}_d\mathbf{u}_{n+1} \\ \mathbf{A}_d &= \mathbf{I} + \Delta t\mathbf{A} \\ \mathbf{B}_d &= \Delta t\mathbf{B} \\ \mathbf{C}_d &= \mathbf{C};\ \mathbf{D}_d = \mathbf{D} \end{cases} \tag{2.2}$$

où $\mathbf{I}$ est la matrice identité, $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$ et $\mathbf{D}_d$ sont respectivement les équivalents discrets des matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ et $\mathbf{D}$; $\mathbf{x}_n$, $\mathbf{y}_n$ et $\mathbf{u}_n$ sont les équivalents échantillonnés des vecteurs continus du système Σ.

La MEE s'implémente facilement mais souffre de limites de stabilité et d'exactitude [99]. La méthode du trapèze est généralement préférable car elle est dite A-stable. Elle mène à

l'approximation Tustin du système donnée par :

$$\Sigma^{\Delta t}_{\text{trapèze}} \equiv \begin{cases} \mathbf{x}_{n+1} &= \mathbf{A}_d\mathbf{x}_n + \mathbf{B}_d\mathbf{u}_{n+1} \\ \mathbf{y}_{n+1} &= \mathbf{C}_d\mathbf{x}_n + \mathbf{D}_d\mathbf{u}_{n+1} \\ \mathbf{A}_d &= (\mathbf{I} - \frac{\Delta t}{2}\mathbf{A})^{-1}(I + \frac{\Delta t}{2}\mathbf{A}) \\ \mathbf{B}_d &= (\mathbf{I} - \frac{\Delta t}{2}\mathbf{A})^{-1}\Delta t\mathbf{B} \\ \mathbf{C}_d &= \mathbf{C}(\mathbf{I} - \frac{\Delta t}{2}\mathbf{A})^{-1} \\ \mathbf{D}_d &= \mathbf{C}(I - \frac{\Delta t}{2}\mathbf{A})^{-1}\Delta t\mathbf{B} + \mathbf{D} \end{cases} \tag{2.3}$$

Ainsi, qu'il s'agisse de la MEE ou de la méthode du trapèze, il apparaît que le système d'équations Σ se ramène à un équivalent discret qui peut s'exprimer sous la forme compacte d'une multiplication matrice-vecteur :

$$\left[\begin{array}{c} \mathbf{x}_{n+1} \\ \hline \mathbf{y}_n \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A}_d & \mathbf{B}_d \\ \hline \mathbf{C}_d & \mathbf{D}_d \end{array}\right] \left[\begin{array}{c} \mathbf{x}_n \\ \hline \mathbf{u}_n \end{array}\right] \tag{2.4}$$

On notera à ce point de la discussion certaines spécificités de l'emploi de ces équations sur FPGA. D'une part, on constatera que la méthode du trapèze nécessite l'inversion d'une matrice puisqu'il faut évaluer $(\mathbf{I} - \frac{\Delta t}{2}\mathbf{A})^{-1}$. Ceci pose problème dans le cas où le système est variant, puisque l'inversion d'une matrice se fait en $\mathcal{O}(n^3)$. Des versions parallèles de l'inversion existent, mais elles s'exécutent au mieux en temps $\mathcal{O}(n^3/p)$ sur p processeurs [36]. Cependant, si le système est invariant, les équations discrétisées du réseau peuvent être précalculées, et la méthode du trapèze peut alors être considérée. Le calcul se résume dans ce cas à une multiplication matrice-vecteur, ce qui est envisageable en $\mathcal{O}(n^2/p)$ sur une architecture parallèle à p processeurs.

2.2.2 Circuit compagnon discrétisé

Avant d'aborder l'analyse nodale à proprement parler, il est utile d'évoquer le concept de circuit compagnon discrétisé. Ce concept est exploité dans différentes formulations des équations du réseau, dont l'approche du tableau [37], l'analyse nodale [20], l'analyse nodale modifiée (*modified-nodal analysis* — MNA) [41] ou encore l'analyse nodale modifiée et augmentée (*modified-augmented-nodal analysis* — MANA) [71, 72].

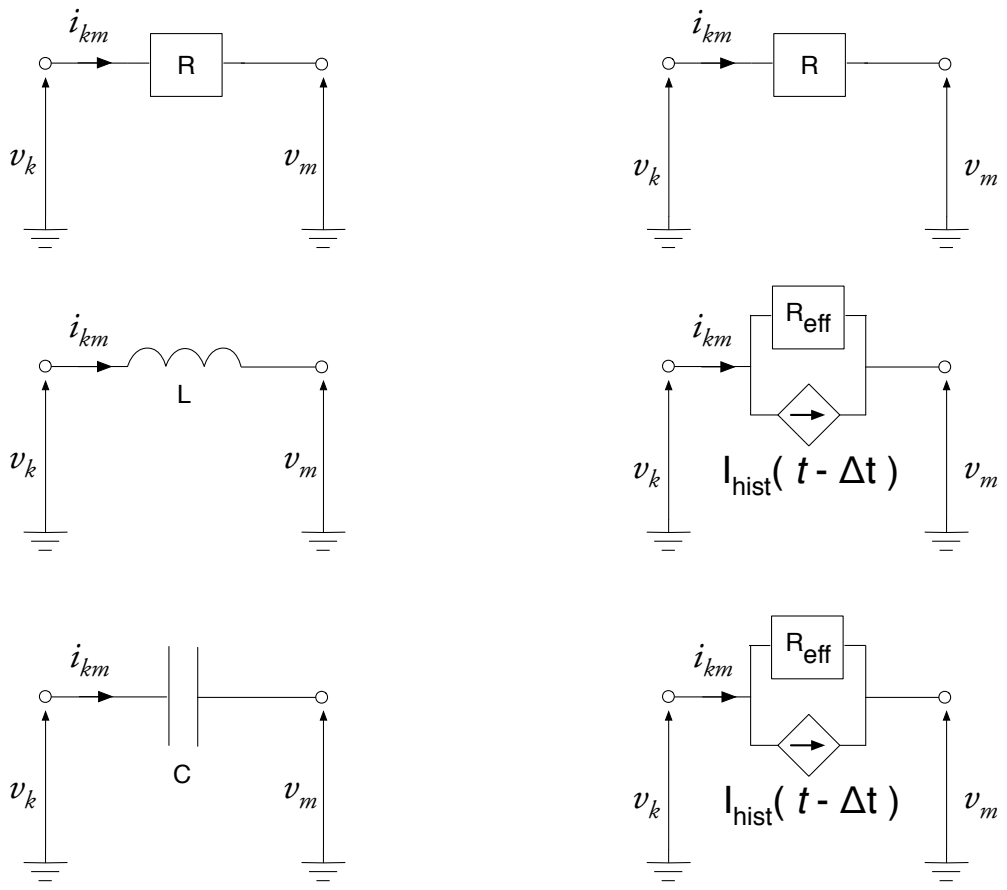

Figure 2.1 – Équivalents Norton (droite) de composants de base (gauche).

Le principe du circuit compagnon discrétisé repose sur l'idée du remplacement de chaque composant du circuit par un équivalent Norton, et d'assembler ces différents composants dans un circuit compagnon dont il convient de décrire les équations du réseau par une des méthodes précitées. La Figure 2.1 donne une illustration de ce que sont les équivalents discrétisés des composants de base, à savoir la résistance, l'inductance et la capacité.

L'équation de la résistance est donnée par la loi d'Ohm et s'exprime sous la forme :

$$i_{km}(t) = \frac{1}{R}\Big(v_k(t) - v_m(t)\Big) \tag{2.5}$$

Sa discrétisation laisse l'équation inchangée et n'implique aucun terme historique.

Dans le cas de l'inductance, nous aurons :

$$\frac{di_{km}(t)}{dt} = \frac{1}{L}\Big(v_k(t) - v_m(t)\Big) \tag{2.6}$$

Il est possible d'utiliser la méthode du trapèze et de poser :

$$i_{km}(t) = i_{km}(t-\Delta t) + \frac{\Delta t}{2L}\Big(v_k(t-\Delta t) - v_m(t-\Delta t)\Big) + \frac{\Delta t}{2L}\Big(v_k(t) - v_m(t)\Big) \tag{2.7}$$

Le terme de courant $i_{km}(t-\Delta t) + \frac{\Delta t}{2L}\Big(v_k(t-\Delta t) - v_m(t-\Delta t)\Big)$ étant associé au passé, on peut le représenter par une source de courant $i_{hist}(t-\Delta t)$, de sorte à retrouver l'équivalent Norton de la Figure 2.1 où le terme $R_{\text{eff}} = 2L/\Delta t$

On écrira alors :

$$i_{km}(t) = i_{hist}(t-\Delta t) + \frac{1}{R_{\text{eff}}}\Big(v_k(t) - v_m(t)\Big) \tag{2.8}$$

Un traitement similaire s'applique au condensateur. Ayant :

$$i_{km}(t) = C\frac{d\Big(v_k(t) - v_m(t)\Big)}{dt} \tag{2.9}$$

la méthode du trapèze donne :

$$i_{km}(t) = \frac{2C}{\Delta t}\Big(v_k(t) - v_m(t)\Big) - \frac{2C}{\Delta t}\Big(v_k(t-\Delta t) - v_m(t-\Delta t)\Big) - i_{km}(t-\Delta t) \tag{2.10}$$

L'équation 2.10 se traduit facilement sous la forme obtenue en (2.8), de sorte que l'on retrouve $R_{\text{eff}} = \frac{\Delta t}{2C}$ et $i_{hist}(t-\Delta t) = -\frac{2C}{\Delta t}\big(v_k(t-\Delta t) - v_m(t-\Delta t)\big) - i_{km}(t-\Delta t)$. Ainsi, le principe de discrétisation des composants suivant la méthode du trapèze peut se généraliser facilement et non sans élégance à l'ensemble des composants du réseau [20].

Tout comme dans l'approche des équations d'états, d'autres techniques de discrétisation existent et peuvent être utilisées. On mentionnera principalement la méthode implicite d'Euler (MIE) qui nous sera utile lorsque nous considérerons les convertisseurs de puissance, comme nous le verrons au Chapitre 5. On notera cependant à ce point de la discussion que, contrairement à l'approche des équations d'états, l'approche du circuit compagnon procède à une discrétisation du circuit au niveau des composants individuels du circuit plutôt qu'à celle de la matrice continue des équations du réseau.

2.2.3 Analyse nodale du réseau

Pour une topologie de réseau donnée, et après la phase de discrétisation de l'ensemble des composants du circuit, il est possible d'appliquer la loi des nœuds de Kirchhoff au réseau afin de le modéliser sous la forme matricielle :

$$\mathbf{G}_t\mathbf{v}_t = \mathbf{i}_t + \mathbf{i}_{hist} \tag{2.11}$$

où $\mathbf{G}$ est une matrice de conductance, $\mathbf{i}_t$ un vecteur contenant les sources de courant externes, $\mathbf{i}_{hist}$ un vecteur contenant les sources de courant constituées des termes passés et où $\mathbf{v}_t$ est un vecteur des tensions nodales inconnues. Notons aussi que les sources de tension idéales impliquent des tensions connues dans le vecteur «d'inconnues ». L'équation 2.11 peut alors être réécrite sous la forme :

$$\left[\begin{array}{c|c} \mathbf{G}_{ii} & \mathbf{G}_{ic} \\ \hline \mathbf{G}_{ci} & \mathbf{G}_{cc} \end{array}\right] \left[\begin{array}{c} \mathbf{v}_{i_t} \\ \hline \mathbf{v}_{c_t} \end{array}\right] = \left[\begin{array}{c} \mathbf{i}_{i_t} \\ \hline \mathbf{i}_{c_t} \end{array}\right] + \left[\begin{array}{c} \mathbf{i}_{i-p} \\ \hline \mathbf{i}_{c-p} \end{array}\right] \tag{2.12}$$

où l'on sépare les termes connus (indice c) des termes inconnus (indice i), de sorte que la solution de $\mathbf{v}_{i_t}$ s'obtient par :

$$\mathbf{G}_{ii}\mathbf{v}_{i_t} = \mathbf{i}_{i_t} + \mathbf{i}_{ip} - \mathbf{G}_{ic}\mathbf{v}_{c_t} \tag{2.13}$$

Cette manipulation algébrique n'est valable que si les sources de tension possèdent un terminal à la masse. Il est possible de contourner cette limitation par des artifices de modélisation (notamment en introduisant des résistances négatives [21]). Il n'en demeure pas moins que la méthode nodale classique souffre de limitations topologiques qu'ont toutefois réussi à surmonter la MNA [41] et la MANA [72]. Dans ce travail, nous nous intéressons uniquement aux équations MANA.

Dans MANA, les équations du réseau sont augmentées de sorte à aboutir à la forme :

$$\mathbf{A}_t\mathbf{x}_t = \mathbf{b}_t \tag{2.14}$$

où $\mathbf{x}_t$ est un vector composite constitué des tensions de nœuds et de courants de branches inconnus. Le vecteur $\mathbf{b}_t$ contient quant à lui les tensions connues et les injections de courant (incluant les termes historiques).

Plus spécifiquement, les équations du réseau s'expriment comme suit [72] :

$$\begin{bmatrix} \mathbf{Y} & \mathbf{V_c} & \mathbf{D_c} & \mathbf{S_c} \\ \mathbf{V_l} & \mathbf{V_d} & \mathbf{D_{VD}} & \mathbf{S_{VS}} \\ \mathbf{D_l} & \mathbf{D_{DV}} & \mathbf{D_d} & \mathbf{S_{DS}} \\ \mathbf{S_l} & \mathbf{S_{SV}} & \mathbf{S_{SD}} & \mathbf{S_d} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{i_V} \\ \mathbf{i_D} \\ \mathbf{i_S} \end{bmatrix} = \begin{bmatrix} \mathbf{i} \\ \mathbf{v_b} \\ \mathbf{d_b} \\ \mathbf{s_b} \end{bmatrix} \tag{2.15}$$

où les indices **l**, **c** et **d** désignent respectivement ligne, colonne et diagonale. La matrice $\mathbf{Y}$ est la matrices d'admittance, $\mathbf{v}$ est le vecteur des tensions inconnues et $\mathbf{i}$ et le vecteur des injections de courant dans le réseau, incluant les termes historiques. La sous-matrice $\mathbf{V_l}$ sert à exprimer les équations des sources de tensions. La sous-matrice $\mathbf{D_l}$ contient les équations des branches dépendantes tandis que $\mathbf{S_l}$ contient les équations des interrupteurs idéaux dans le circuit. Les sous-matrices colonnes sont le plus souvent les transposées des matrices lignes (par exemple, $\mathbf{D_c} = \mathbf{D_l}^T$), tandis que les matrices sur la diagonale sont toutes nulles (exceptées $\mathbf{Y}$ and $\mathbf{S_d}$). Les sous-matrices restantes ($\mathbf{V_{DV}}$, $\mathbf{D_{VD}}$, etc.) sont le plus souvent nulles aussi.

Comme nous l'avions fait pour les équations d'états, nous proposons de reformuler les équations MANA afin d'assurer une exécution performante de ces dernières sur FPGA. Pour ce faire, nous définissons le vecteur $\mathbf{j}_n$ contenant tous les termes historiques dans le circuit compagnon, ainsi que le vecteur $\mathbf{u}_n$ contenant toutes les sources indépendantes du réseau. On définira également le vecteur $\mathbf{y}_n$ contenant les différences de tension et les courants de branche du réseau utiles à l'usager durant la simulation. En posant $\mathbf{b}_n = \mathbf{K}_c[\ \mathbf{j}_n,\ \mathbf{u}_n\ ]^T$, où $\mathbf{K_c}$ est la matrice de connexion, on trouve :

$$\mathbf{x}_n = \mathbf{A}_t^{-1}\mathbf{K}_c \left[\begin{array}{c} \mathbf{j}_n \\ \hline \mathbf{u}_n \end{array}\right] \tag{2.16}$$

En supposant que la matrice MANA est invariante ($\mathbf{A}_t \equiv \mathbf{A}$), il est possible de poser le système d'équations suivant :

$$\left[\begin{array}{c} \mathbf{j}_{n+1} \\ \hline \mathbf{y}_n \end{array}\right] = \left[\begin{array}{c|c} \mathbf{W_{jj}} & \mathbf{W_{ju}} \\ \hline \mathbf{W_{yj}} & \mathbf{W_{yu}} \end{array}\right] \left[\begin{array}{c} \mathbf{j}_n \\ \hline \mathbf{u}_n \end{array}\right] \tag{2.17}$$

Il convient de relever qu'en évaluant $\mathbf{j}_{n+1}$, l'équation 2.17 se trouve à anticiper les termes historiques pour le prochain pas de temps de la simulation. Ceci est possible car les termes historiques $\mathbf{j}_n$ du pas actuel sont toujours exprimés par une combinaison linéaire des tensions aux nœuds ($\mathbf{v}_{n-1}$) et des termes historiques ($\mathbf{j}_{n-1}$) du pas de temps précédent.

On relèvera également que la forme de l'équation 2.17 ressemble beaucoup à celle de l'équation 2.4 que nous avions établie pour les équations d'états. Dans le cas de la méthode

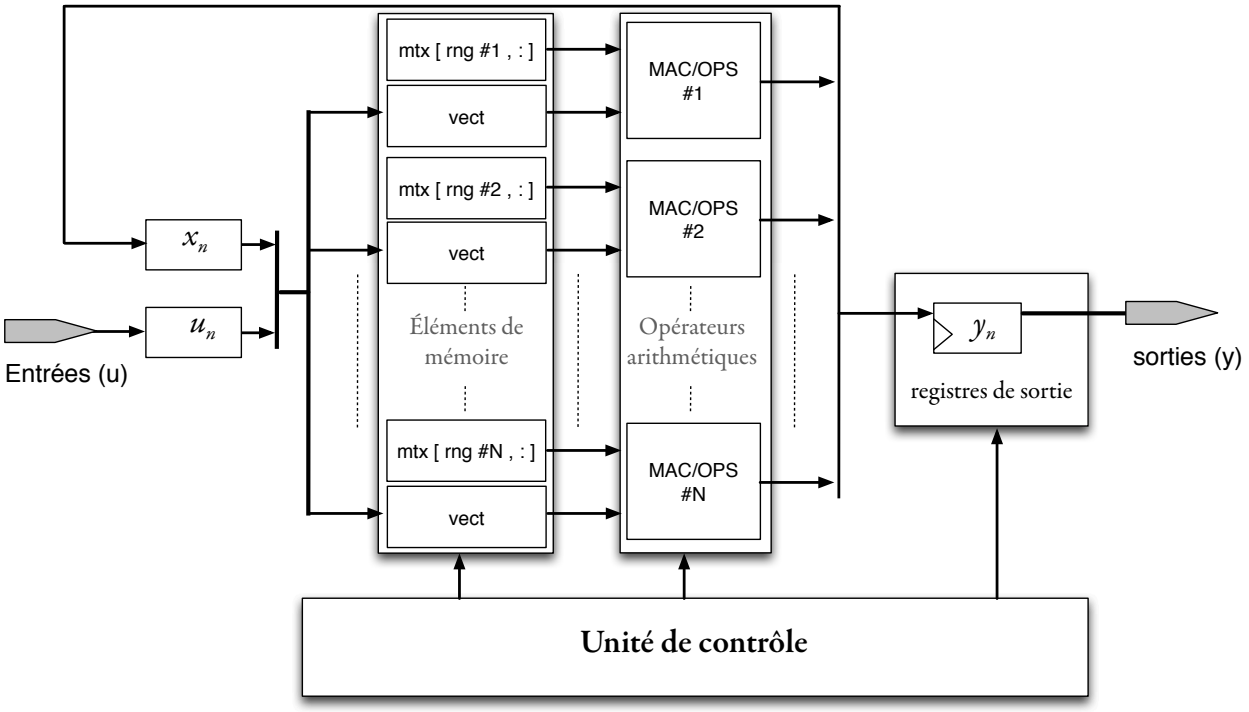

Figure 2.2 – Esquisse d'engin de calcul pour la simulation des réseaux.

nodale, les termes historiques sont en quelque sorte les états du circuit ($\mathbf{j}_n \equiv \mathbf{x}_n$). Ce résultat nous outille pour analyser deux méthodologies de modélisation des réseaux de la même manière. Ce qui vaudra pour l'une vaudra pour la seconde. Aussi, le restant de ce chapitre se concentrera sur les équations d'états pour caractériser les paramètres des engins de calcul que nous voulons réaliser. Nous reviendrons au Chapitre 5 à l'analyse nodale quand il s'agira de considérer la modélisation des convertisseurs de puissance au moyen de la modélisation des interrupteurs à matrice constante [45, 47, 94].

2.3 Caractérisation des engins de calcul

2.3.1 Esquisse d'une architecture d'engin de calcul

Afin de simuler en temps réel les équations du réseau, ce qui précède a démontré qu'il convient de résoudre une équation de produit matrice vecteur. Pour ce faire, nous proposons d'utiliser l'architecture de PUS présentée à la Figure 2.2. Cette architecture sera raffinée et détaillée dans le restant de ce chapitre et les chapitres subséquents.

L'engin de calcul [1] que nous proposons — et que nous avons décliné sous plusieurs formes pour résoudre avec succès maints problèmes de simulation sur puce [25, 27, 30, 49, 81, 84, 85, 86, 87, 88] — se compose des éléments suivants :

1. Entrée accueillant le vecteur $\mathbf{u}_n$;
2. Éléments de mémoire (registres, RAM) servant à entreposer la matrice (équations du réseau) et le vecteur (entrées et états du circuit) qui lui est multiplié ;
3. Registres de sortie permettant d'offrir à l'usager les mesures faites sur le réseau ;
4. Opérateurs arithmétiques disposés en parallèle constitués de MAC ou de OPS ;
5. Unité de contrôle permettant de cadencer les séquences d'opérations.

La séquence des opérations cadencée par l'unité de contrôle se traduit par la lecture des entrées pour le pas de calcul à exécuter, la lecture en séquence des valeurs des équations du réseau et des éléments du vecteur correspondant, la réécriture des états du circuit dans les éléments de mémoire une fois ceux-ci obtenus et l'écriture en sortie des mesures effectuées sur le réseau pour le même pas de calcul.

Le type d'éléments de mémoire utilisés pour entreposer les équations du réseau sont généralement des blocs de mémoire présents dans le FPGA (BRAM). Les BRAM des FPGA (tels que le Virtex 5 ou le Spartan 3 de Xilinx [118, 119] qui ont été exploités dans le présent travail) comportent deux ports de lecture et d'écriture (voir Annexe A). En supposant que le système considéré est invariant, les BRAM sont alors exploitées en mode lecture seule (ROM). Les équations du réseaux sont alors réparties sur autant de BRAM que nécessaire (ce qui correspond au nombre de MAC ou de OPS).

Le vecteur quant à lui peut requérir différents types d'éléments de mémoire. Pour de petits problèmes ($|\mathbf{u}_n| + |\mathbf{x}_n| < 20$) [2], des registres individuels suffisent. Pour des problèmes de taille plus importante, l'utilisation de BRAM peut s'avérer utile. On relèvera toutefois que les mêmes valeurs du vecteur sont lues en même temps par les multiples MAC/OPS. Ainsi, si leur nombre n'est pas excessif, une seule instance de mémoire peut être utilisée et sa sortie acheminée aux différents opérateurs. Dans le cas où le nombre d'opérateurs serait plus élevé, il peut valoir la peine de dupliquer l'instance de mémoire pour le même vecteur. Relevons finalement que pour éviter d'écraser une valeur du vecteur d'états par sa valeur actualisée, nous utilisons le principe de double tampon. Au premier pas de simulation, l'engin de calcul lit le tampon #1 et écrit dans le tampon #2. Au prochain pas de temps, le contrôleur aiguille

1. Nous utilisons les termes PUS et engin de calcul de manière interchangeable.
2. On admettra ici que $|\mathbf{v}|$ signifie dimension du vecteur $\mathbf{v}$ ($\dim(\mathbf{v})$), cette notation ayant été choisie en raison de sa simplicité.

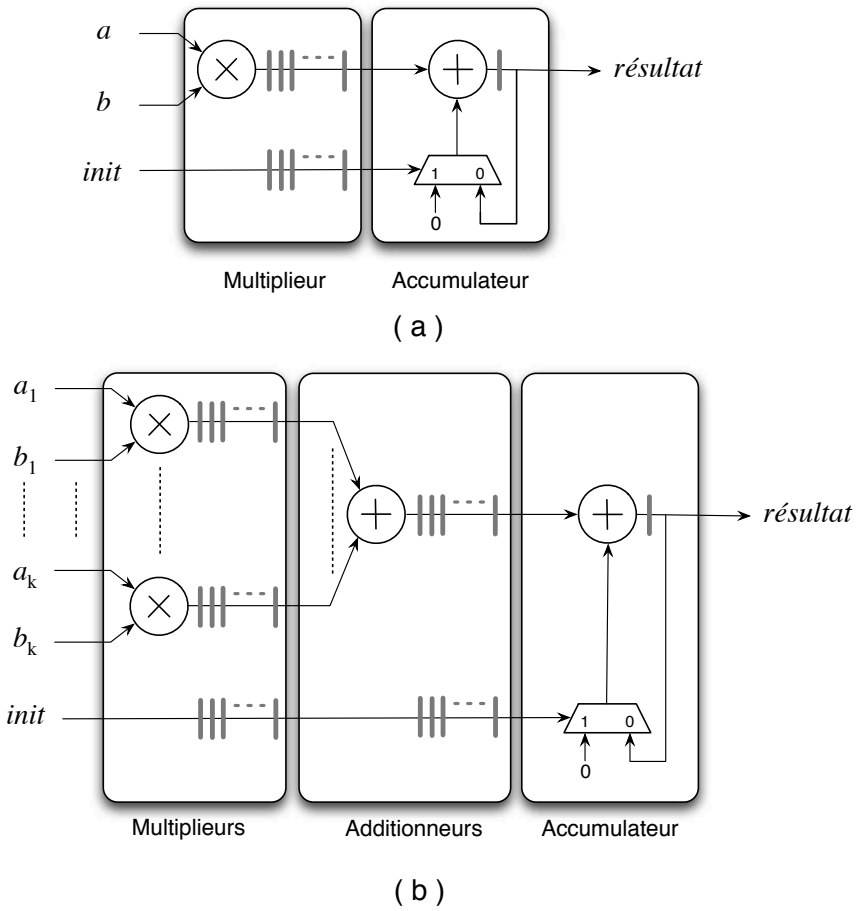

Figure 2.3 – Opérateurs utilisés par les engins de calcul : (a) MAC ; (b) OPSk.

les lectures depuis le tampon #2 et écrit dans le tampon #1. L'alternance se poursuit ainsi tout au long de la simulation.

Les opérateurs de la Figure 2.3 illustrent un MAC et un OPS à k multiplieurs (on écrira souvent OPSk). Le MAC est constitué d'un multiplieur suivi d'un additionneur bouclé sur lui même et dont le but est d'effectuer l'accumulation. L'additionneur dispose d'un mécanisme permettant d'initialiser l'accumulation illustré à la Figure 2.3 par un multiplexeur. Si ledit mécanisme ne requiert pas la mise à zéro d'un registre (contrairement à ce qui se passe dans l'accumulateur du bloc DSP par exemple [120]), il est possible d'épargner un cycle d'horloge par cycle d'accumulation. Généralement, chaque opérateur (MAC ou OPS) réalise plusieurs accumulations par pas de calcul. Les multiples cycles d'initialisation ainsi épargnés permettent alors de réduire le pas de calcul. L'OPS ressemble au MAC, à la différence qu'il dispose de plusieurs multiplieurs à son entrée, suivi d'un circuit de réduction effectuant la somme des résultats des multiplications que complète un accumulateur.

Qu'il s'agisse du MAC ou de l'OPS, la réalisation en virgule flottante (VF) des opérateurs arithmétiques peut s'obtenir de deux manières principalement : 1) l'utilisation d'additionneurs et de multiplieurs élémentaires en VF issus d'une librairie ; 2) l'utilisation d'un opérateur intégré fait «à la main». Dans le premier cas, l'intérêt du procédé se justifie par la facilité d'intégration des opérateurs commerciaux et la garantie d'obtenir un résultat fidèle à la norme IEEE. cependant, cette approche exclut la possibilité d'effectuer l'accumulation en un cycle et nécessite d'entrelacer plusieurs calculs. Cette nécessité d'entrelacement a tendance à réduire l'efficacité du parallélisme de l'engin de calcul, particulièrement pour des problèmes de petite et de moyenne envergures.

La seconde approche a le désavantage de nécessiter la conception *ad hoc* d'opérateurs arithmétiques en VF. Cette tâche peut s'avérer ardue et laborieuse. Surtout elle peut difficilement être confiée à des non spécialistes. Néanmoins, nos travaux ont permis d'emprunter cette approche en offrant les bénéfices suivants : i) un accumulateur à un cycle ; ii) des opérateurs de latence réduite ; iii) une précision supérieure ; iv) une garantie d'obtenir une meilleure qualité d'exactitude des calculs ; v) une compacité de l'opérateur ; vi) un fonctionnement en haute fréquence. Le problème de l'accès à cette technologie par des non spécialistes peut alors être garanti par la réalisation d'un engin de calcul versatile permettant au moyen d'un logiciel approprié de traduire les équations du réseaux en contenu de BRAM, comme ce sera discuté au Chapitre 5.

Il convient de relever que l'architecture de la Figure 2.2 ne nous est pas propre. On la retrouve sous cette forme ou avec de subtiles différences dans les travaux d'autres chercheurs [10, 66, 74, 75, 97, 98]. Ce qui fait l'originalité de notre travail est l'utilisation d'OPS en VF, ce qui constitue une réalisation rendue possible grâce à la méthodologie qui sera élaborée aux chapitres 3 et 4.

2.3.2 Impact des séquences de lecture sur les pas de temps de la simulation

L'analyse que nous présentons ici a été discutée dans [81] où nous avions proposé l'implémentation sur FPGA d'un modèle de moteur sans balais. Il s'agissait alors d'utiliser des opérateurs élémentaires (additionneurs/multiplieurs) pour réaliser un MAC multi-cycles, c'est-à-dire un MAC dont la boucle d'accumulation est effectuée sur plusieurs cycles : c'est là la première approche proposée à la section précédente pour implémenter un MAC. Dans l'exemple considéré ici, les matrices $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$ et $\mathbf{D}_d$ ont respectivement une taille de 5×5, 5×3, 4×5 et 4×3,. Ceci signifie que le circuit admet 5 états, reçoit 3 entrées et produit 4 sorties. L'engin de calcul quant à lui dispose de 3 MAC multi-cycles et effectue l'accumulation en trois cycles. La Figure 2.4 illustre comment les données et les calculs sont répartis sur les trois MAC.

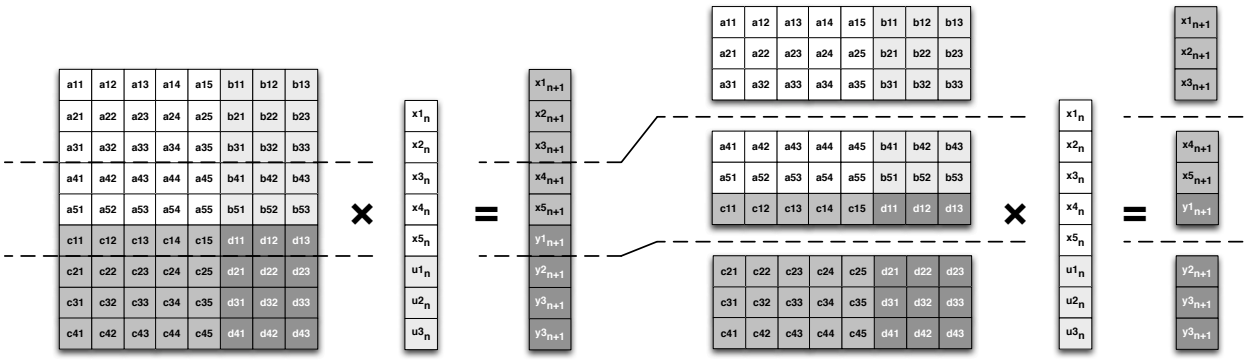

Figure 2.4 – Partitionnement d'un système d'états sur trois MAC.

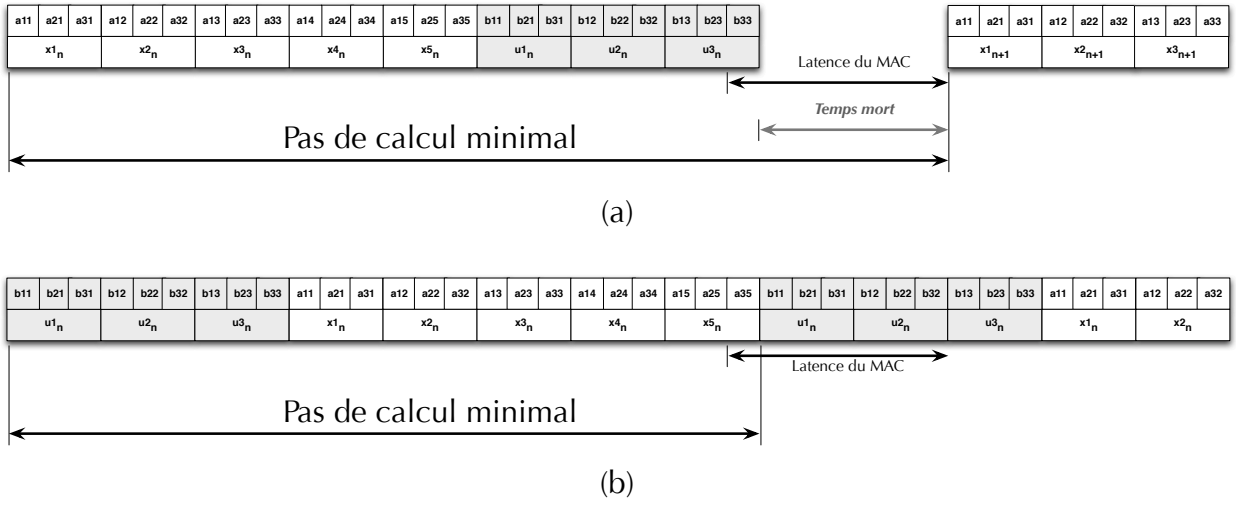

Figure 2.5 – Principe d'optimisation dune résolution multi-MAC.

L'idée d'effectuer l'accumulation sur plusieurs cycles en présence d'un additionneur pipeliné a précédemment été exploitée dans [28]. Nous allons montrer dans ce qui suit comment il est possible d'éviter certains écueils spécifiques à notre formulation du problème. Le principe de départ consiste à lire les éléments de la matrice par colonnes. Puisque le nombre de lignes assignées à un MAC est exactement égal à la latence de l'accumulateur (cas idéal de 3 lignes pour 3 cycles par accumulation), les niveaux de pipeline de l'additionneur jouent le rôle de mémoire où les résultats partiels des produits scalaires sont entreposés. Une fois que toutes les lignes ont été lues, les trois nouveaux résultats apparaissent en sortie du MAC décalés d'un cycle chacun, et après un délai égal à la latence total du MAC, comme l'illustre la Figure 2.5.a.

Dans le contexte de la simulation en temps réel, un nouveau cycle de calcul ne peut débuter tant que les nouvelles valeurs des états du circuit ne sont pas disponibles. La Figure 2.5.a illustre la pénalité subie par le temps de calcul résultant d'une lecture des matrices suivant l'ordre naturel de leur écriture. La latence du MAC fait en sorte que l'engin de calcul est obligé de subir des cycles de temps mort où aucun traitement n'est effectué. Il est possible de remédier à cette situation en privilégiant le vecteur $\mathbf{u}_n$ à $\mathbf{x}_n$, comme l'illustre la Figure 2.5.b. De cette façon, le nouveau cycle de lecture peut débuter plus tôt et la latence du MAC est recouverte par les cycles où les données rattachées à $\mathbf{u}_n$ (celles des matrices $\mathbf{B}_d$ et $\mathbf{D}_d$) sont lues. Dans l'exemple de la Figure 2.5.b, cette approche fait en sorte que plus aucun cycle de temps mort n'existe. Ce résultat n'est bien entendu pas toujours possible selon le problème rencontré et la façon de le traiter (accumulation à un cycle ou à plusieurs cycles). Néanmoins, le principe qui consiste à privilégier $\mathbf{u}_n$ à $\mathbf{x}_n$ s'applique toujours.

2.3.3 Relation entre la taille du modèle et les opérateurs arithmétiques

Dans cette section, nous discutons de la relation qui existe entre la taille du modèle à résoudre et le type d'opérateurs arithmétiques utilisés. La Figure 2.6 présente l'évolution du pas de calcul pour différents tailles de problème en fonction du type et du nombre d'opérateurs utilisés. Cette analyse posait certaines difficultés nécessitant les simplifications que nous allons détailler ici. La difficulté principale de cette analyse relève du grand nombre de paramètres à considérer. D'une part, la matrice des équations du réseau, telle que nous l'avons définie précédemment, a une taille $(|\mathbf{x}_n| + |\mathbf{y}_n|) \times (|\mathbf{x}_n| + |\mathbf{u}_n|)$ et dépend par conséquent de trois paramètres ($|\mathbf{x}_n|$, $|\mathbf{y}_n|$ et $|\mathbf{u}_n|$). D'autre part, le pas de temps dépend du type d'opérateurs (MAC, OPSk) et du nombre (N) de ceux-ci au sein l'architecture de la Figure 2.2. Il est cependant possible de réduire considérablement le nombre de paramètres en tenant compte du fait que le pas de calcul est une fonction monotone ; elle est décroissante en fonction des paramètres de l'engin de calcul, elle est croissante en fonction de la taille du problème. Aussi, pouvons-nous considérer, sans perte de généralité, que $|\mathbf{u}_n| = |\mathbf{y}_n|$ et que $k = N$ (OPS1 $\equiv$ MAC). En supposant une horloge de 200 MHz (5 ns) et une latence (l) de 20 cycles par opérateur (ce sont là des hypothèses raisonnables comme nous le verrons dans les chapitres subséquents), on peut poser l'équation déterminant le pas de calcul (t_s) :

$$t_s = 5 \left(\left\lceil \frac{(|\mathbf{x}_n| + |\mathbf{y}_n|)}{N} \right\rceil \left\lceil \frac{(|\mathbf{x}_n| + |\mathbf{u}_n|}{k} \right\rceil + l \right) \ ns \tag{2.18}$$

La Figure 2.6 considère six tailles de problème ($8 \times 8, 16 \times 16, \ldots, 256 \times 256$) et 16 types/nombre d'opérateurs ($k = N = 1, 2, \ldots 16$). On vise une limite de pas de calcul de 1 μs pour les raisons évoquées au Chapitre 1. Plusieurs conclusions préliminaires peuvent

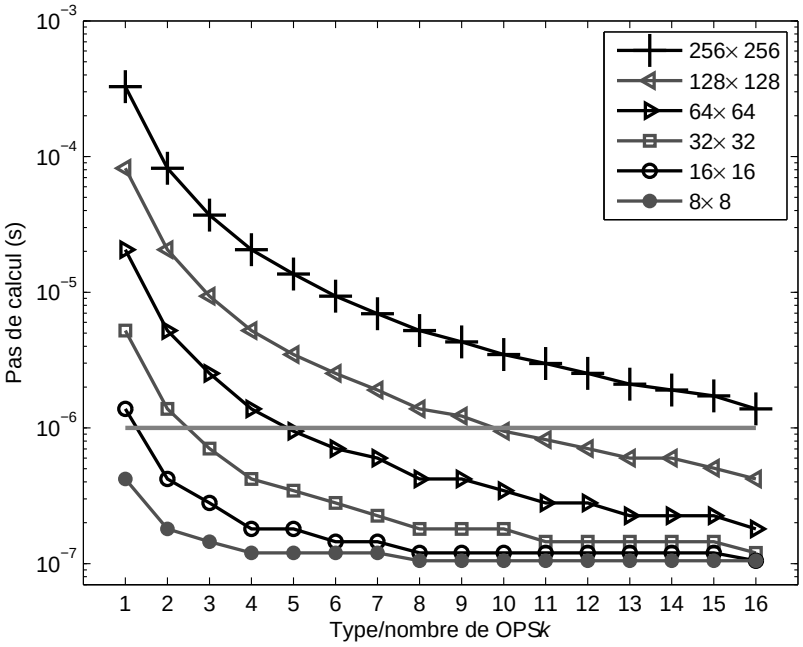

Figure 2.6 – Pas de calcul fonction du type/nombre d'opérateurs OPSk.

être tirées de cette étude. D'une part, il apparaît que si l'on se limite à un pas de calcul de 1 μs, la taille du problème ne peut dépasser un certain seuil critique. Ainsi, aucune des topologies d'engin de calcul considérées n'a permis de passer sous la limite de 1 μs pour un problème de taille 256×256. D'autre part, il apparaît que si un seul MAC ($k = N = 1$) est utilisé, seuls des problèmes de très petite taille peuvent être considérés. On remarquera également que tous les pas de calcul tendent asymptotiquement vers la limite de 100 ns. Cette limite résulte de la latence de l'opérateur (20×5 ns $= 100$ ns). Finalement, on voit clairement l'intérêt d'exploiter des OPS. Plus ces opérateurs sont larges (grandes valeurs de k), plus l'engin de calcul dispose d'une puissance de calcul à même de l'aider à résoudre des problèmes de grande taille.

Afin de quantifier l'intérêt d'utiliser les OPS, nous proposons de considérer l'efficacité des engins de calculs selon la taille du problème et le type d'OPS considéré. L'efficacité d'une architecture parallèle à p processeurs est définie comme le ratio sur p de l'accélération obtenue par ladite architecture [36]. Ici, les accélérations prennent comme référence une architecture à un seul OPS1. Nous considérons dans nos calculs que p est donné par $p = kN$. Cette évaluation est raisonnable puisque un OPSk possède k fois plus d'additionneurs/multiplieurs qu'un OPS1. L'accélération quant à elle sera le ratio du pas de calcul de l'architecture à un OPS1 sur celui d'une architecture à N OPSk.

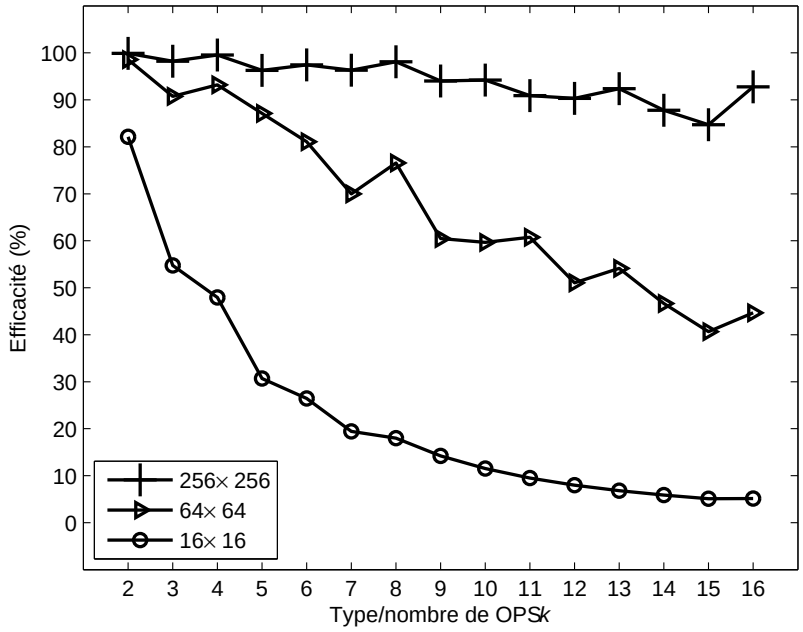

Figure 2.7 – Efficacité des opérateurs OPSk.

L'efficacité (*Eff.*) d'un engin de calcul est donc donnée par :

$$Eff.(N,k) = \frac{t_s(1 \text{ OPS1})}{p \cdot t_s(N \text{ OPS}k)} = \frac{t_s(1 \text{ OPS1})}{k \cdot N \cdot t_s(N \text{ OPS}k)} \tag{2.19}$$

Les résultats de cette analyse sont reproduits à la Figure 2.7, où l'efficacité (en pourcentage) apparaît en ordonnée. La Figure 2.7 considère les cas où $k = N$ afin réduire le nombre de paramètres. Trois tailles de problème sont étudiés : $16 \times 16, 64 \times 64$ et 256×256. Les résultats pour des tailles intermédiaires peuvent se déduire par interpolation. La Figure 2.7 montre clairement que l'efficacité se détériore avec la réduction de la taille du problème et l'augmentation de la taille des opérateurs. Ainsi, il apparaît que l'efficacité de l'engin calcul de la Figure 2.3 est presque parfaite pour une matrice de taille 256×256 pour toutes les tailles d'opérateurs, alors qu'elle devient rapidement sous-efficace pour une petite taille de matrice. Ce résultat est cohérent puisque pour un problème de petite taille, une architecture de parallélisme excessif est sous utilisée. Considérons plus spécifiquement le cas d'une matrice 16×16 qui serait exécutée sur une architecture à 16 OPS16. Le parallélisme en entrée fait qu'un seul cycle d'horloge est nécessaire pour effectuer toutes les lectures. Cependant, l'engin de calcul doit attendre que les résultats du calcul soient disponibles avant de passer au prochain pas de temps. Ainsi, plus la latence de l'opérateur est basse, meilleure est l'effi-

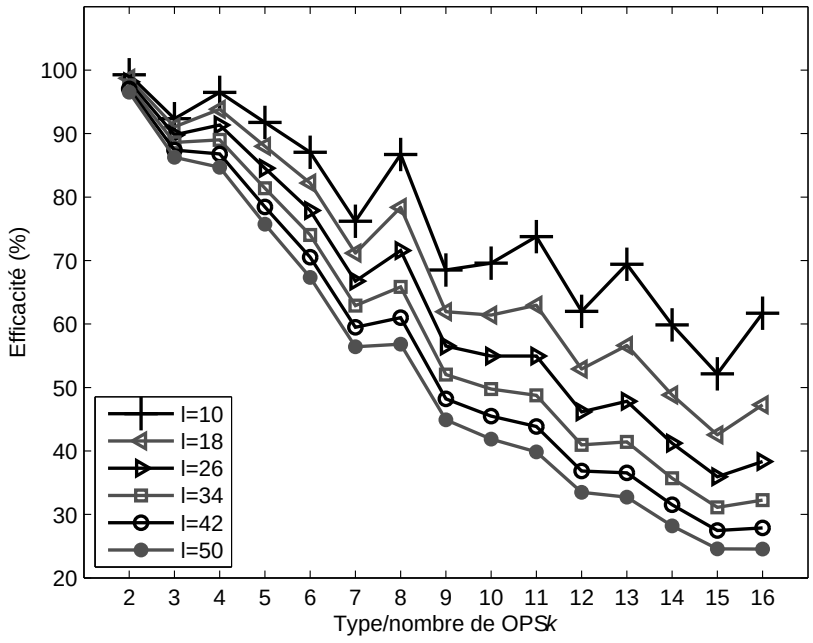

Figure 2.8 – Impact de la latence de l'OPSk sur l'efficacité des engins de calcul.

cacité de l'engin de calcul. La section suivante propose justement de discuter l'influence de la latence des opérateurs arithmétiques sur les performances de l'engin de calcul.

2.3.4 Influence de la latence des opérateurs

Nous proposons d'évaluer l'impact de la latence des opérateurs arithmétiques sur les performances de l'engin de calcul de la Figure 2.3. Pour ce faire, nous considérons uniquement une taille de problème de 64×64 pour laquelle nous faisons varier la taille et le nombre des opérateurs OPS de 1 à 16 (un OPS1 reste l'architecture de référence). La Figure 2.8 présente l'évolution de l'efficacité de l'architecture pour différentes latences, allant de 10 à 50 selon un incrément de 8. Nous avons choisi le spectre le plus large possible afin de donner une idée suffisamment précise de l'évolution de l'efficacité. Nous avons également choisi de nous concentrer sur le cas spécifique d'un problème 64×64 car il est suffisant pour contenir diverses topologies de convertisseurs de puissance, comme nous le verrons au Chapitre 5.

Il apparaît assez clairement de la Figure 2.8 que l'efficacité de l'engin de calcul se dégrade avec la latence et l'excès de parallélisme offert. L'explication de ce phénomène a été discutée précédemment, et les résultats présentés ici la confirment : plus le parallélisme est grand, plus l'engin de calcul est porté à effectuer toutes les lectures rapidement et à attendre que

les résultats lui parviennent, durée qui correspond à la latence de l'opérateur OPS. Ainsi, plus le nombre de cycles de lecture est bas, plus la latence tend à dégrader l'efficacité de l'engin de calcul.

Néanmoins, il ne faut pas perdre de vue les résultats de la Figure 2.6. L'objectif principal de l'engin de calcul est de minimiser le temps de calcul. Ainsi le choix du nombre et type d'opérateurs à adopter est un compromis de conception qui dépend des topologies de circuit et des pas de calcul visés. Aussi, nous remarquerons que l'efficacité d'une architecture à deux OPS2 varie entre 85% et 95%, mais elle ne permet d'obtenir un pas sous la barre du 1 μs que pour de petits problèmes. Cependant, une architecture à quatre OPS4 ou à huit OPS8 permet d'atteindre un pas de calcul inférieur à 1 μs pour une large gamme de problèmes bien que leur efficacité soit inférieure à celle d'une architecture à deux OPS2.

2.4 Conclusion

Ce chapitre a réalisé une analyse de problème portant sur le formalisme de la modélisation des réseaux selon les équations d'états et l'analyse nodale. Il a permis de ce fait d'unifier les deux méthodes de modélisation et a rendu possible une étude du problème circonscrite à une seule et même formulation, à savoir celle du produit matrice-vecteur contraint dans le temps. Cet acquis a ensuite permis d'esquisser une architecture d'engin de calcul fait de MAC/OPS parallèles. Le chapitre s'est alors attardé à évaluer l'impact des différents paramètres de l'architecture (type d'opérateurs, latence) sur les performances de l'engin de calcul en termes de pas de calcul (avec pour objectif de minimiser ce dernier) et l'efficacité globale du PUS. Les Chapitres 3 et 4 vont porter sur les moyens de réaliser une addition/accumulation à un cycle, de sorte à réduire la latence globale des OPS et de garantir une bonne exactitude des résultats. Le Chapitre 5 s'attardera quant à lui sur la manière d'exploiter ces opérateurs dans la simulation en temps réel des réseaux commutés.

CHAPITRE 3

OPÉRATEURS DE SOMMATION EN VIRGULE FLOTTANTE

3.1 Introduction

Les chapitres 1 et 2 ont permis de mettre en évidence l'importance de disposer d'opérateurs en VF de faible latence et de pouvoir opérer des réductions de somme de façon compacte et rapide. Afin de mieux saisir la nature du défi que représente l'implémentation de tels opérateurs en matériel, nous proposons de faire un bref retour sur les bases de l'addition en VF dans un premier temps, puis de considérer plus généralement le problème de la réduction de somme et de la possibilité de la réaliser avec la technologie FPGA. Ce chapitre nous permettra donc de découvrir comment il est possible de réaliser en matériel un accumulateur en virgule flottante, c'est-à-dire un opérateur permettant d'effectuer la boucle d'accumulation en un cycle d'horloge. Les contributions de ce chapitre élargissent les concepts utilisés dans la réalisation de la fonction d'accumulation aux autres formes d'opérateurs où la sommation joue un rôle important, tels que l'arbre d'additionneurs ou l'OPS.

3.2 Problématique

3.2.1 Additionneur en virgule flottante

La Figure 3.1 illustre l'algorithme de base de l'addition en virgule flottante [29]. L'étape I consiste à calculer la différence des exposants, de sorte à permuter les opérandes à l'étape II si l'exposant du premier opérande est inférieur à l'exposant du second. Ainsi, à l'étape III, seule la mantisse du second opérande est décalée à droite si les exposants ne sont pas égaux. Ces trois étapes sont illustrées à la Figure 3.2 à laquelle nous reviendrons plus loin. Une fois que les mantisses sont correctement alignées, on procède à leur addition ou soustraction à l'étape IV. Il est à noter que lors d'une soustraction de nombres de magnitudes relativement égales, la mantisse résultant de cette opération peut subir une perte de résolution. C'est alors qu'entrent en ligne de compte les étapes V et VI qui consistent à détecter le bit le plus significatif de la mantisse résultante et de décaler à gauche cette dernière aux fins de normalisation. L'étape VII procède à l'arrondi de la mantisse. L'étape VIII ajuste l'exposant en conséquence afin de refléter les modifications apportées aux étapes V à VII. L'étape IX traite les cas spéciaux de l'arithmétique flottante, soit Zéro, Inf., NaN, etc.

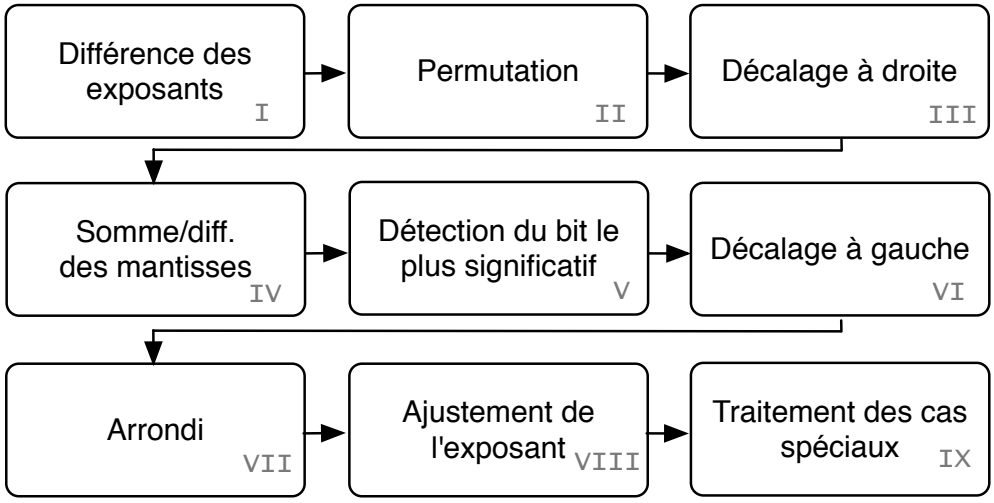

Figure 3.1 – Algorithme d'addition en virgule flottante.

La Figure 3.2 présente deux exemples d'addition illustrant ces étapes. Dans l'exemple 1, les exposants sont différents, de sorte que l'exposant non biaisé du premier opérande ($e_a = 0$) est plus petit que celui du second opérande ($e_b = 3$). Les opérandes sont donc permutés à la phase #2 avant d'être alignés à la phase #3 puis additionnés à la phase #4.

Dans l'exemple 2, on procède plutôt à l'addition de deux nombres de signes différents, et de magnitudes relativement égales. On voit alors qu'à la phase #2, le résultat perd plusieurs bits de résolution. La mantisse est par conséquent décalée à gauche pour préparer la normalisation à la phase #3. L'exposant est ajusté en conséquence.

Il est important de comprendre que les décaleurs variables coûtent cher en latence et en ressources logiques. Ainsi, il est impossible de concilier haute fréquence de fonctionnement et accumulation à un cycle si l'addition en virgule flottante suit une lecture stricte de l'algorithme de la Figure 3.1. Afin de réduire la latence de l'additionneur, il est possible d'utiliser deux chemins de données parallèles, l'un effectuant le décalage de l'étape III, le second celui de l'étape VI [107]. Le principe de cette approche provient du fait que les deux étapes sont mutuellement exclusives. Ainsi, la latence de l'additionneur est réduite au prix d'un peu de logique supplémentaire. Cependant, une telle approche ne suffit pas pour offrir une solution au problème de l'addition à un cycle.

3.2.2 Exactitude de la sommation en virgule flottante

Nous appelons sommation l'opération de réduction de somme de deux opérandes ou plus. La sommation en virgule flottante souffre de problèmes d'exactitude dus à l'accumulation

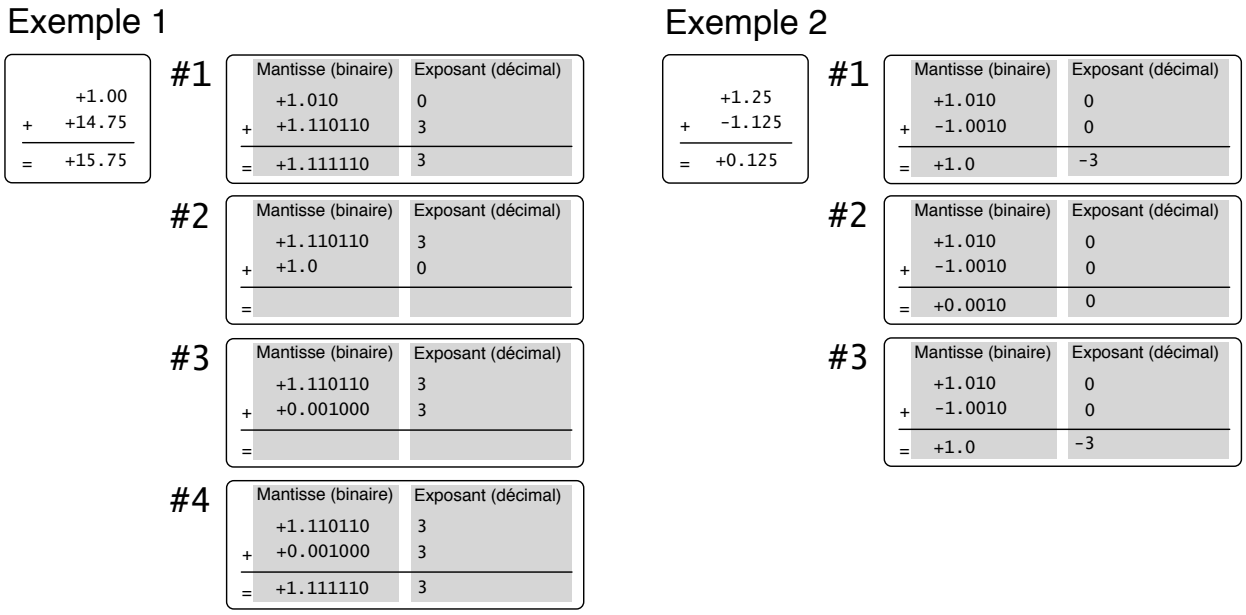

Figure 3.2 – Exemples d'addition en virgule flottante.

des erreurs d'arrondis des additions successives qui la composent. Plus spécifiquement, étant donnée la non associativité de l'addition en VF (il est possible pour trois nombres en VF a_{vf}, b_{vf} et c_{vf} d'obtenir $(a_{vf} + b_{vf}) + c_{vf} \neq a_{vf} + (b_{vf} + c_{vf})$). En effet, partant du principe que la norme IEEE de l'arithmétique flottante [48] impose aux opérateurs arithmétiques de base ($+$, $-$, $\times$, $\div$) de garantir une erreur absolue inférieur à $\frac{1}{2}ulp(x_{vf} \odot y_{vf})$, où $\odot$ est une opération arithmétique de précision infinie, et où $ulp(\cdot)$ est la fonction *unit in the last place* (**ULP!**) [77], il devient clair que chaque nouvelle addition opérée dans l'exécution de la sommation vient accroître l'erreur totale. C'est pourquoi la normeIEEE évoque l'exactitude de la sommation (*sum*) en virgule flottante sans lui imposer de contrainte réelle :

> *Sums are computed in a manner that avoids overflow or underflow in the calculation and the final result is determined from that intermediate result.*

Le travail de Higham [39] a permis d'établir une borne supérieure à l'erreur relative d'une sommation en virgule flottante. En notant S_N le résultat exact de la sommation de N nombres flottants x_i $(1 \leq i \leq N)$ et en notant $\hat{S}_N$ son approximation, on obtient :

$$\frac{|\hat{S}_N - S_N|}{S_N} \leq \frac{(N-1)u}{1-(N-1)u} R_N \equiv \gamma_{(N-1)} R_N \tag{3.1}$$

où $R_N = \sum_{i=1}^{N} |x_i| / |\sum_{i=1}^{N} x_i|$ est le conditionnement de la somme (une grande valeur de R_N indique un mauvais conditionnement) et $u = \frac{1}{2}ulp(1.0)$ (u vaut 2^{-24} en simple précision, 2^{-53}

en double précision). Si $N \ll 1/u$ (ce qui est généralement le cas), alors $\gamma_{(N)} \simeq Nu$, de sorte que l'erreur relative soit de l'ordre de Nu fois R_N. Il convient de mentionner cependant que l'équation 3.1 ne donne qu'une bonne supérieure à l'erreur relative — une borne pessimiste par ailleurs puisque l'erreur relative est généralement bien moins importante.

Techniques logicielles pour réduire l'erreur de la sommation

Il existe différentes techniques permettant d'améliorer l'exactitude d'une sommation. La plus connue d'entre elles est certainement la technique de Kahan [55], dont l'erreur relative est bornée par $(2u+O(Nu^2))R_N$, ce qui signifie que la sommation est aussi exacte que si elle avait était opérée avec une précision double de la précision utilisée, puis que le résultat avait été arrondi "conformément" [1] : ainsi si le calcul était réalisé en simple précision, le résultat serait aussi exact que si la sommation avait été opérée en double précision, puis arrondie conformément à la simple précision.

Les techniques modernes dites de distillation sont plus élaborées que l'algorithme de Kahan bien qu'elles en respectent l'esprit. On les appelle techniques de distillation car elles procèdent en transformant les opérandes de départ pour des opérandes aux meilleures propriétés. Ainsi, la distillation substitue les opérandes y_j $(1 \leq j \leq M)$ aux opérandes x_i $(1 \leq i \leq N)$ de sorte que $\sum_{i=1}^{N} x_i = \sum_{j=1}^{M} y_j = S_N$, tout en veillant à ce que le conditionnement du nouvel ensemble d'opérandes soit meilleur : $\sum_{j=1}^{M} |y_j| / |\sum_{j=1}^{M} y_j| \ll \sum_{i=1}^{N} |x_i| / |\sum_{i=1}^{N} x_i|$.

C'est ainsi qu'une K-tuple exactitude est rendue possible au moyen de la technique de distillation proposée dans [80] : une K-tuple signifie ici que le résultat de la sommation est aussi exact que s'il avait été mené avec K fois la précision employée, puis arrondi à la précision de travail (la technique de Kahan offre une K-tuple exactitude, où $K = 2$). L'algorithme de distillation ultime a par ailleurs été récemment proposé dans [103], garantissant une sommation exacte conformément arrondie, c.-à-d. que l'erreur relative est bornée par $2u$, indépendamment de R_N.

Techniques matérielles de sommation

Les approches matérielles ne souffrent pas des contraintes que subissent les approches logicielles où il n'est possible de faire autrement que d'employer les précisions disponibles sur la machine. Ainsi, il existe différentes formes d'opérateurs matériels de sommation. Certaines sont conventionnelles tandis que d'autres présentent des originalités topologiques assez frappantes. Ainsi, un opérateur de sommation peut prendre différentes formes. Par exemple. il

1. Arrondi conforme est une traduction libre du concept de *faithful rounding* présenté dans [103]. À toutes fins utile, cela signifie que l'on substitue la contrainte arrondi$(x) \leq \frac{1}{2} ulp(x)$ à arrondi$(x) < \frac{1}{2} ulp(x)$.

peut se concevoir comme un additionneur à plusieurs entrées dans lequel toutes les opérandes sont traitées simultanément [112, 117]. Il peut également se concevoir comme un accumulateur où les opérandes arrivent séquentiellement [62, 69, 89]. Il est également possible de fusionner l'additionneur multi-opérandes et l'accumulateur afin d'augmenter le débit du sommateur [76, 125]. D'autres approches plus ou moins exotiques sont également rapportées dans la littérature [56, 111].

L'exactitude d'une sommation matérielle dépend bien évidemment de la stratégie adoptée. Ainsi, il est relativement aisé d'assurer un résultat exactement arrondi au prix d'une mantisse interne très large [5, 62, 112]. Il est également possible de borner l'erreur par des approches topologiques. Par exemple, un arbre binaire d'additionneur a une erreur relative bornée par $\gamma_{\log_2(N)} R_N$ [39]. Notre contribution dans cette thèse est une méthode de sommation matérielle inspirée de la technique dite d'auto-alignement [69, 116] qui se traduit par une exactitude K-tuple, exprimée par une logique de contrôle fort simple tout en permettant la réalisation d'une accumulation à un cycle en simple et en double précision.

3.2.3 Accumulation à un cycle par l'auto-alignement des mantisses

La technique d'auto-alignement (TAA) des mantisses est une méthode permettant l'accumulation à un cycle de nombres en VF [69]. Elle procède en minimisant l'interaction entre la somme en cours et les nouveaux opérandes en procédant à l'alignement de chaque nouvelle mantisse par rapport à un repère commun à toutes les mantisses, repère qui sera défini par leur exposant, comme nous allons le voir. La TAA permet par conséquent d'opérer tous les décalages des mantisses en dehors de la boucle d'accumulation dont le chemin critique est par conséquent réduit. Une version modifiée de la TAA, où des considérations touchant à la perte de précision sont traitées, fut proposée dans [116]. Des implémentations FPGA de la technique furent proposées dans [82, 89].

La Figure 3.3 présente les étapes principes d'un accumulateur exploitant la TAA. Dans un premier temps, l'entrée a est dépaquetée, c.-à-d. défaite de ses parties constituantes : le signe s_a, l'exposant e_a de w_e bits et la mantisse m_a de w_{um} bits. La mantisse m_a est alors convertie vers un format signé en complément à 2 puis décalée à gauche selon la valeur du nombre formé par les l bits les moins significatifs de l'exposant e_a. l bits sont alors tronqués de l'exposant. Cette étape produit un exposant réduit e_i de largeur w_{re} bits ($e_i = e_a[w_e - 1 : l]$, $w_{re} = w_e - l$) et une mantisse élargie m_i de w_{em} bits ($2^l + w_{um} \leq w_{em}$). À l'étape suivante, les e_i et m_i consécutifs sont accumulés, produisant une somme cumulée exprimée par un exposant réduit e_f et une mantisse élargie m_f. La dernière étape de post-accumulation sert à convertir le résultat du format interne vers un format standard, après arrondi et normalisation. Dans [116], les auteurs ne considéraient que la simple précision

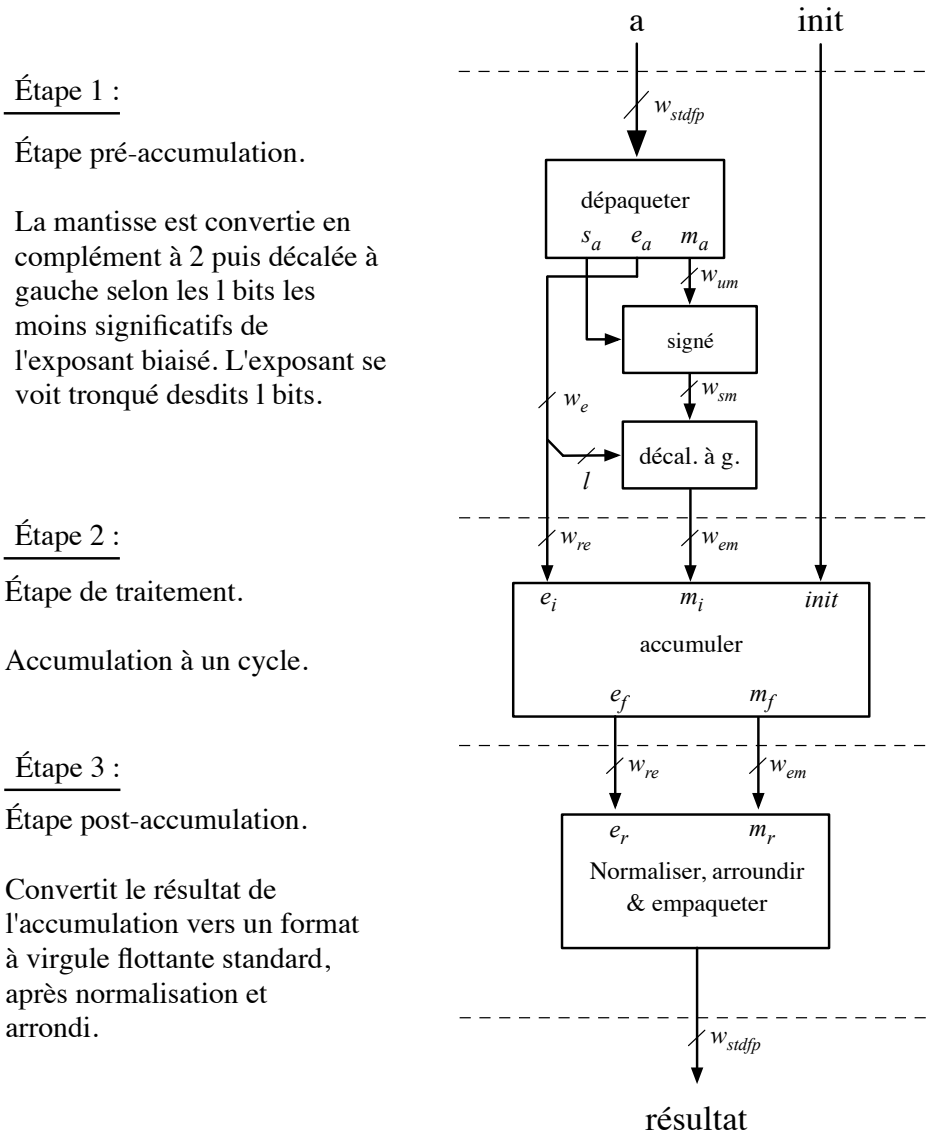

Figure 3.3 – Étapes principales d'un accumulateur TAA.

($w_{stdfp} = 32$) et utilisaient les paramètres $l = 5$, $w_{em} = 57$ et $w_{re} = 3$. Ces valeurs sont choisies de façon que $w_{em} \leq 2^{l+1}$ afin de limiter les décalages à droite de la mantisse à l'intérieur de la boucle d'accumulation. Nous verrons à la Section 3.3 comment nos travaux ont permis de généraliser cette approche de sorte à permettre l'exploitation de la TAA dans la réalisation d'accumulateurs à double précision, mais aussi dans la réalisation d'opérateurs où l'addition est une opération critique tels que le MAC et l'OPS.

L'implémentation de la boucle d'accumulation peut être obtenue en adoptant le chemin de données de la Figure 3.4. Nous avons proposé ce chemin de données dans [82] où fut

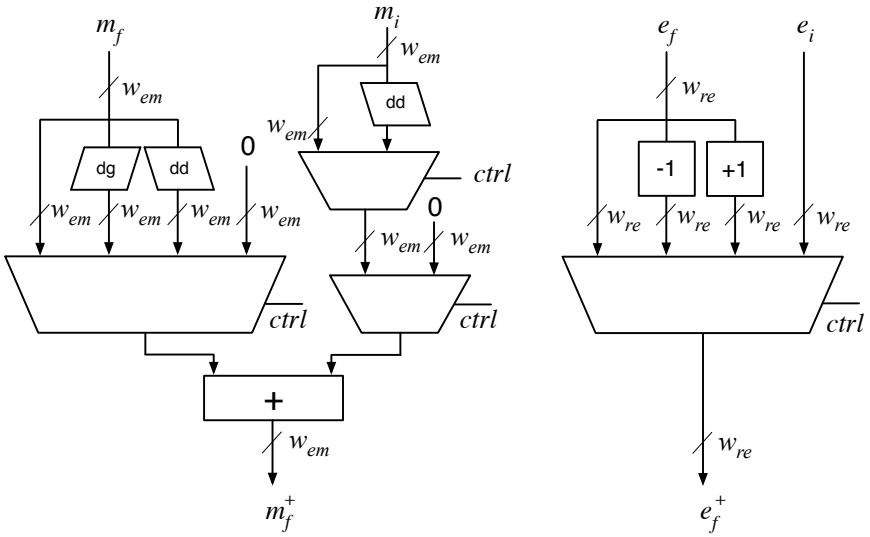

Figure 3.4 – Chemin de données de la boucle d'accumulation tel que proposé dans [82].

Tableau 3.1 – Logique de contrôle de l'accumulateur à virgule flottante.

Mise à jour de la mantisse :
$m_f^+ \leftarrow \mathrm{dd}(m_f^+)$ en cas de prémisse de débordement de m_f^+

$e_f - e_i$	nil	prémisse de perte de précision	autres
≤ -2	m_i	m_i	m_i
-1	m_i	m_i	$\mathrm{dd}(m_f) + m_i$
$+0$	m_i	$m_f + m_i$	$m_f + m_i$
$+1$	m_i	$\mathrm{dg}(m_f) + m_i$	$m_f + \mathrm{dd}(m_i)$
$+2$	m_i	$\mathrm{dg}(m_f) + \mathrm{dd}(m_i)$	m_f
$\geq +3$	m_i	m_f or $\mathrm{dg}(m_f)$	m_f

Mise à jour de l'exposant :
$e_f^+ \leftarrow e_f^+ + 1$ en cas de prémisse de débordement de m_f^+

$e_f - e_i$	nil	prémisse de perte de précision	autres
≤ -2	e_i	e_i	e_i
-1	e_i	e_i	$e_f + 1$
$+0$	e_i	e_f	e_f
$+1$	e_i	$e_f - 1$	e_f
$+2$	e_i	$e_f - 1$	e_f
$\geq +3$	e_i	e_f or $e_f - 1$	e_f

également proposée la logique de contrôle décrite à la Table 3.1. Le cas *nil* renvoie au cas où l'accumulateur est initialisé (init = 1) ou le cas où $m_f = 0$. La prémisse de perte de précision est déterminée par le fait que m_f possède plus que 2^l zéros ou uns consécutifs à sa gauche. Les opérateurs dd() et dg() sont respectivement des décalages de 2^l positions vers la droite et la gauche. Finalement, avant de registrer e_f^+ et m_f^+, il convient de vérifier la prémisse de débordement de m_f^+. On entend par là qu'on veut vérifier que la valeur de m_f^+ est tellement grande qu'elle risquerait de déborder si on y ajoutait une nouvelle valeur, ce qui peut être déterminé par le fait que les deux bits les plus significatifs de m_f^+ sont différents. Dans un tel cas, e_f^+ est incrémenté ($e_f^+ \leftarrow e_f^+ + 1$) et m_f^+ est décalé à droite de 2^l positions ($m_f^+ \leftarrow \text{dd}(m_f^+)$).

Le travail que présente ce chapitre permettra de déceler des lacunes dans la formulation classique de la TAA, notamment concernant l'exactitude de ses résultats. Aussi aurons nous procédé en la formalisant de sorte à mieux pouvoir la généraliser et en étendre la portée à d'autres types d'opérateurs matériels où la sommation a un rôle déterminant.

3.3 Généralisation de la TAA

La Section 3.2.3 a permis de couvrir la TAA et d'en saisir les limitations potentielles. En effet, il est apparu que la logique de contrôle repose sur plusieurs vérifications effectuées sur la mantisse m_f et m_f^+ (prémisse de débordement, zéro et prémisse de perte de précision). Comme ces vérifications doivent toutes être réalisées en un cycle, il est évident que le chemin critique en est affecté. Cela est d'autant plus vrai lorsque la double précision est considérée. Il est possible de remédier à ce problème en adoptant la stratégie que nous avons proposée dans [82], et que nous avons formalisée dans [83].

3.3.1 Le format d'auto-alignement

Nous proposons de généraliser la sommation de type TAA en définissant un format d'auto-alignement (FAA) auquel sont associés trois paramètres entiers : l, le nombre de bits pris de l'exposant standard pour l'auto-alignement de la mantisse ; w, la largeur de la mantisse étendue ; b, une valeur de biais permettant de situer la valeur du réel représenté en TAA. Ainsi, un réel x exprimé en TAA(l, w, b) est donné par une paire d'entiers (e, m), de sorte que :

$$x = m \cdot 2^{2^l e - b} \tag{3.2}$$

La Figure 3.5 illustre les différentes parties d'une mantisse étendue en FAA. À l'extrême gauche, on retrouve $\lceil \log_2(N) \rceil$ bits dont le rôle est d'éviter tout débordement durant la sommation de N opérandes. Ces $\lceil \log_2(N) \rceil$ bits ne représentent pas une limitation dans l'implémentation d'un accumulateur basé sur la TAA puisque N peut demeurer élevé ($N \gg 10^6$) à un coût matériel raisonnable. Cette affirmation sera vérifiée à la Section 4.4.4. Suivent ensuite $2^l - 1$ bits bits permettant de contenir les décalages potentiels que subit la mantisse durant l'auto-alignement. On trouvera à leur droite w_{sm} bits permettant d'accueillir la mantisse signée du format standard. Ainsi, w_{sm} vaut 25 en simple précision et 54 en double précision. Finalement, g bits de garde se retrouvent à l'extrême droite de la mantisse étendue afin d'assurer l'exactitude K-tuple, comme nous allons le démontrer ci-après.

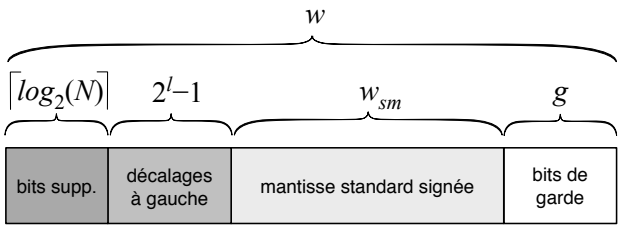

Figure 3.5 – Composition de la mantisse élargie en FAA.

Contrairement à ce que nous avions vu jusqu'ici, w n'est plus limitée à 2^{l+1} bits. Rappelons que cette limitation était imposée dans la formulation originale afin de réduire le nombre de décalages à granularité large à deux potentialités. Il convient de mentionner que le FAA ne nécessite pas une mantisse normalisée, ce qui en fait un format redondant dès lors que $w > 2^l$. Ainsi, chaque réel admet $\lceil w/2^l \rceil$ représentations redondantes dans FAA. Par exemple, $x = 10.78125$ peut être représenté des différentes manières suivantes :

$$\begin{aligned} x &= 10.78125 = 1.34765625 \cdot 2^3 = 690 \cdot 2^{-6} \\ &= (690) \cdot 2^{(2^4)(9)\text{-}150} \equiv (9, 690)_{\mathrm{FAA}(4,64,150)} \\ &\equiv (9, \text{0x0000 0000 0000 02B2})_{\mathrm{FAA}(4,64,150)} \\ &\equiv (8, \text{0x0000 0000 02B2 0000})_{\mathrm{FAA}(4,64,150)} \\ &\equiv (7, \text{0x0000 02B2 0000 0000})_{\mathrm{FAA}(4,64,150)} \\ &\equiv (6, \text{0x02B2 0000 0000 0000})_{\mathrm{FAA}(4,64,150)} \end{aligned} \tag{3.3}$$

Il importe de relever que la valeur du biais b s'obtient par $b = 23 + 127 = 150$, où 23 est le nombre de bits fractionnaires dans la mantisse standard de la simple précision (qui a servi dans l'expression de 690) et 127 est le biais de la même précision. Ici nous avons une mantisse

de $w = 64$ bits, de sorte que l'on obtient $\lceil w/2^l \rceil = 64/2^4 = 4$ représentations redondantes de la même quantité. Nous le verrons plus loin, cette redondance complexifie l'évaluation de l'erreur de la sommation suivant la TAA puisque, selon la position du premier bit actif de poids fort dans la mantisse, le bit de poids faible aura un poids différent au regard de l'*ulp* de la précision de travail.

Cette problématique nous porte à définir le concept de FAA premier. Un réel est exprimé en FAA premier si le premier bit actif de poids fort (position où l'extension de signe débute dans la mantisse élargie) se trouve dans l'intervalle de positions $w_{sm} + g - 1$ à $w_{sm} + g + 2^l$, tel qu'indiqué à la Figure 3.6.

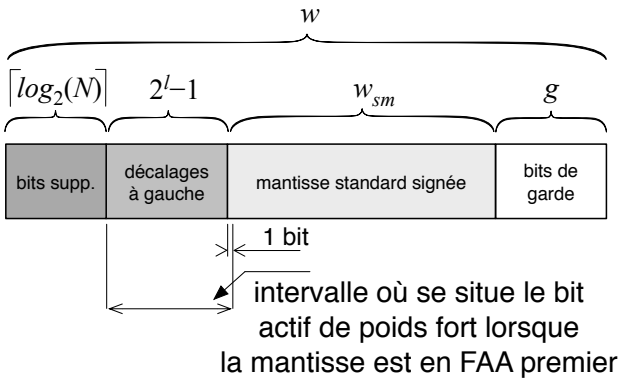

Figure 3.6 – Position du bit de poids fort dans le FAA premier.

3.3.2 Sommation de type TAA d'exactitude K-tuple

L'Algorithme 1 présente notre formulation d'une addition de type TAA. La normalisation et l'arrondi peuvent être effectués sur le résultat pour respecter les exigences de l'encodage binaire du standard IEEE 754. Cet algorithme a la qualité d'une expression concise qui en simplifie la logique de contrôle. Il peut être utilisé dans l'implémentation d'un accumulateur ou d'un additionneur multi-opérandes.

Algorithme 1 $s \leftarrow \texttt{SafSum}(x, y)$

Entrées : $x \equiv (e_x, m_x)$ et $y \equiv (e_y, m_y)$
Sortie : $s \equiv (e_s, m_s)$
$e_s \leftarrow \max(e_x, e_y)$
$q_x \leftarrow e_s - e_x$
$q_y \leftarrow e_s - e_y$
$m_s \leftarrow (m_x \gg (q_x \cdot 2^l)) + (m_y \gg (q_y \cdot 2^l))$
Retourne s

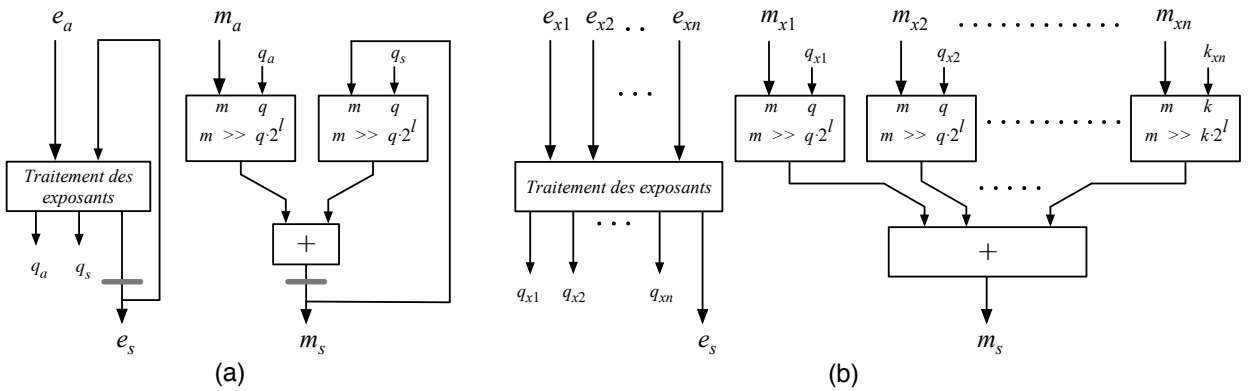

Figure 3.7 – Opérateurs TAA : (a) accumulateur ; (b) additionneur multi-opérandes. Le trait gras indique un niveau de registre.

L'implémentation d'un accumulateur est illustrée à la Figure 3.7.a, tandis que celle d'un additionneur multi-opérandes est donnée à la Figure 3.7.b. Les mécanismes des deux opérateurs sont similaires, à l'exception que l'accumulateur traite les opérandes de manière sérielle, tandis que l'additionneur multi-opérande traite toutes les opérandes d'un coup. Ainsi, l'additionneur multi-opérandes est plus susceptible aux erreurs de troncature puisque celles-ci sont simultanées. C'est pourquoi nous avons mené l'estimation de l'erreur en supposant qu'elle est réalisée par un additionneur multi-opérandes à N entrées. Il en résulte qu'avant la normalisation et l'arrondi, la sommation admet une borne supérieure de son erreur absolue exprimée par :

$$\begin{aligned} |\hat{S}_N - S_N| &\leq (N-1)u2^{-g} \max_{i=1}^{N} |x_i| \\ &\leq (N-1)u2^{-g} \sum_{i=1}^{N} |x_i| \end{aligned} \tag{3.4}$$

où $u = \frac{1}{2}ulp(1.0)$ dans la précision de travail. L'équation 3.4 s'obtient par le raisonnement suivant. Les erreurs de la sommation TAA s'obtiennent des différents décalages à grain large de la mantisse étendue. Par conséquent, la plus grande erreur absolue s'obtient dans le cas où toutes les mantisses sont complètement décalées à droite, excepté pour l'une d'elle.

Ainsi, après normalisation et arrondi, l'erreur relative se trouve bornée par :

$$\begin{aligned}\frac{|\hat{S}_N - S_N|}{|S_n|} &\leq (2u + (N-1)u2^{-g})\frac{\sum_{i=1}^{N}|x_i|}{|\sum_{i=1}^{N} x_i|} \\ &\leq (2u + O(Nu^K))R_N\end{aligned} \tag{3.5}$$

en autant que $\log_2(1/u)(K-1) \leq g < \log_2(1/u)K$ et que tous les opérandes soient exprimés en FAA premier. En vérité, ce résultat est intuitif. Supposons que la précision de travail est la simple précision. Il en résulte que $u = 0.5ulp(1.0) = 2^{-24}$. En ayant g bits de garde à la droite de la mantisse FAA tels que $24(K-1) \leq g < 24K$, on s'assure une exactitude K-tuple de la précision de travail. Ainsi, si l'on admet $g = 24$ bits de garde, on s'assure une exactitude de double précision ($K = 2$). Il apparaît donc bien que l'Algorithme 1 assure une sommation aussi précise que si elle était effectuée en K-tuple précision et que le résultat était arrondi conformément vers la précision de travail.

3.4 Résultats

3.4.1 Exactitude de l'accumulation de type TAA

Il est faux de croire qu'une mantisse interne large suffit à garantir la bonne exactitude du résultat de la sommation. Cette dernière est plutôt assurée par l'algorithme de sommation employé. C'est ce que nous démontrons ici. Pour ce faire, nous avons utilisé des tests numériques connus que rapporte la littérature [39, 78] sur un ensemble d'accumulateurs de type TAA et que résume le Tableau 3.2. Deux approches y sont exploitées.

Tableau 3.2 – Architectures matérielles utilisées pour les tests numériques.

Architecture	**Bits de garde (g)**	**TAA : (l, w, b)**	**N**
TAA original	0	$(5, 64, 150)$	∞
TAA K-tuple	0	$(3, 48, 150)$	2^{16}
TAA K-tuple	8	$(3, 56, 158)$	2^{16}
TAA K-tuple	16	$(3, 64, 166)$	2^{16}
TAA K-tuple	24	$(3, 72, 174)$	2^{16}

1. La première approche est celle de l'accumulateur de [116], utilisant une mantisse interne de 64 bits et le chemin de données de la Figure 3.4 ;
2. La seconde approche est un accumulateur de type TAA d'exactitude K-tuple de la Figure 3.7.a. L'accumulateur est conçu de sorte à pouvoir accepter une séquence

de $N = 2^{16}$ opérandes au plus. Nous avons considéré différentes valeurs de g, soit $g = 0, 8, 16$ et 24.

Les tests numériques ont été menés en suivant la méthodologie suivante :

- Les opérateurs testés ont été conçus en utilisant la boîte à outils System Generator de Xilinx. Il s'agit d'un outil permettant d'effectuer une simulation dans l'environnement Matlab/Simulink précis au coup d'horloge et au bit près. Tous les opérateurs ont été conçus pour travailler en simple précision, de sorte à pouvoir effectuer une évaluation précise de l'erreur en double précision.
- Les erreurs relatives et le conditionnement sont évalués tout au long de la séquence de sommation, c.-à-d. pour toutes les sous séquences initiales, et ce du début à la fin de la séquence de sommation.
- Le résultat exact de la sommation est évalué en employant la double précision en utilisant la routine `FastAccSum` [102] qui est une version accélérée de l'algorithme de distillation ultime présenté dans [103]. La routine `FastAccSum` est librement distribuée avec le logiciel INTLAB [101].
- Les erreurs sont évaluées avant normalisation et arrondi, c.-à-d. sur le résultat en FAA.

Le premier test numérique considère la somme d'une séquence de 64 000 valeurs aléatoires issues d'une distribution normale centrée réduite. Ce test permet d'évaluer la performance de l'accumulateur dans une situation typique. La Figure 3.8.a présente l'évolution du conditionnement de la sommation tout au long de la séquence d'entrée où il apparaît clairement que la valeur de R_N est en général dans l'intervalle $10^2 - 10^4$. Ceci signifie que la sommation est plutôt bien conditionnée. Le trait fin noir de la Figure 3.8.b présente l'évolution de l'erreur relative des résultats produits par l'accumulateur de type TAA original [116]. La même figure indique également comment ces résultats se comparent à ceux d'une sommation réalisée en simple précision par une simple boucle *for* (trait gras de gris clair) ainsi que que ceux de la même sommation réalisée en double précision par une simple boucle *for* (trait gras de gris foncé). Lorsque l'erreur absolue est 0.0, nous avons volontairement forcé l'erreur relative à 10^{-20} pour simplifier l'affichage sur l'échelle logarithmique.

Ce premier test montre que la double précision permet d'obtenir un calcul parfaitement exact puisque l'erreur relative est maintenue à un niveau plancher. L'accumulateur original de type TAA produit des résultats qui sont proches de l'arithmétique en simple précision, parfois meilleurs, parfois moins bons. Il est ainsi démontré qu'une mantisse large ne suffit pas à garantir l'exactitude des résultats, mais aussi que la formulation originale donnée dans [116] présente certaines lacunes.

La Figure 3.8.c présente les erreurs relatives des résultats produits par les accumulateurs de type TAA d'exactitude K-tuple, et ce pour les différentes valeurs de g considérées. Il

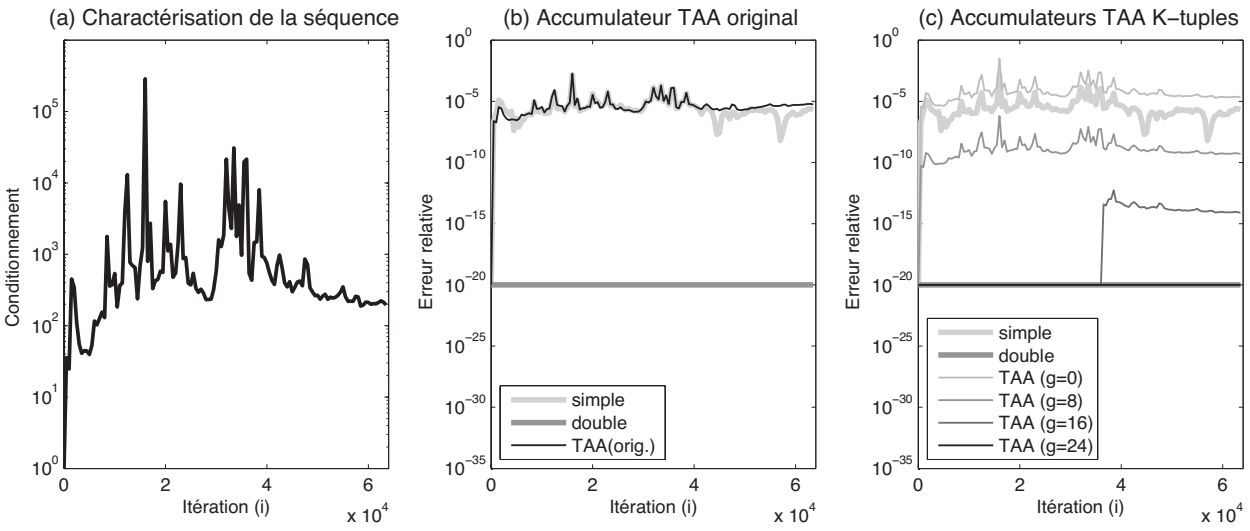

Figure 3.8 – Erreurs relatives et conditionnement d'une séquence de valeurs tirées de la distribution normale centrée réduite.

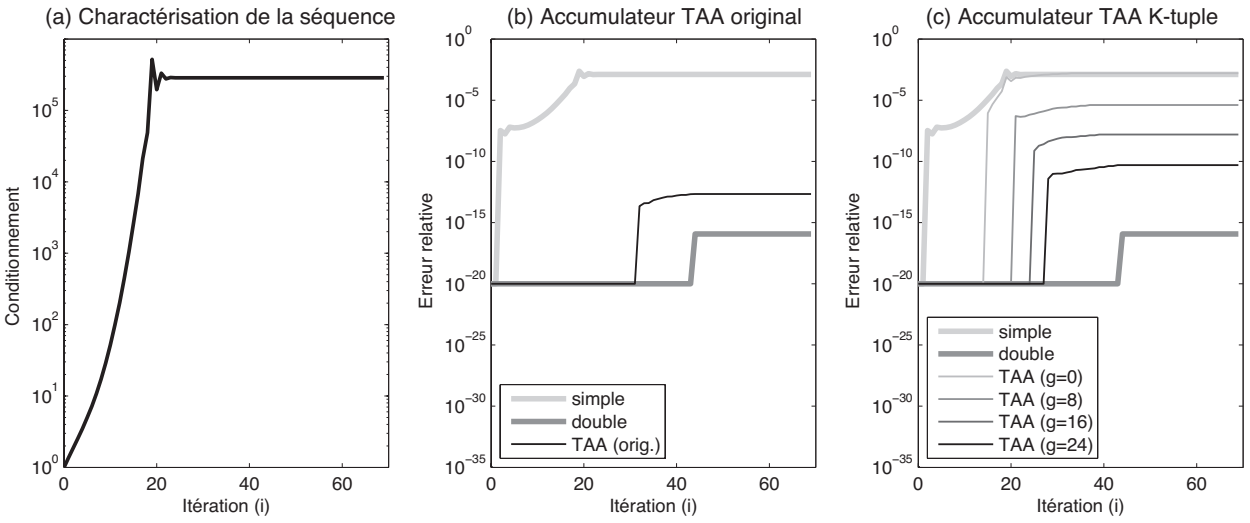

Figure 3.9 – Erreurs relatives et conditionnement d'une séquence de valeurs tirées de la série de Taylor de e^{-x} pour $x = 2\pi$.

apparaît ici que l'erreur relative baisse considérablement quand croît g. Lorsque $g = 0$, les résultats sont moins précis que ceux de la simple précision. Par contre, l'erreur relative est déjà appréciable pour $g = 8$ ($K = 1.33$) puisque l'erreur relative se trouve alors dans l'intervalle 10^{-10} to 10^{-9}. Pour $g = 16$, le résultat est exact au début de l'accumulation, puis l'erreur relative remonte vers 10^{-15}. Pour $g = 24$, les résultats produits sont parfaitement exacts.

La seconde expérience prend en entrée les 70 premiers termes issus de la série de Taylor autour de l'origine de la fonction e^{-x}, évalués pour $x = 2\pi$. Il s'agit là d'un exemple classique d'une sommation de type perte de précision catastrophique (*catastrophic cancellation*), comme l'illustre parfaitement la Figure 3.9.a où l'on voit la progression du conditionnement du problème : R_N atteint rapidement des valeurs très grandes qui dépassent les 10^5, ce qui indique un très mauvais conditionnement. Il est d'usage de ne considérer que les 40 premiers termes de la suite. Nous avons choisi les 70 premiers termes pour illustrer qu'au delà des 40 premiers termes, la double précision ne suffit plus à garantir un résulta exact.

Il apparaît aux Figures 3.9.b et 3.9.c que la double précision ne suffit plus pour obtenir des résultats parfaitement exacts. De plus, on constate à la Figure 3.9.b que l'accumulateur de type TAA original produit des résultats parfaitement exacts jusqu'à l'itération $i = 25$, moment à partir duquel une erreur relative de l'ordre de 10^{-13} apparaît. Ce résultat s'explique par le fait que cet opérateur a été conçu pour résoudre le problème de *cancellation*. Les erreurs relatives des accumulateurs de type TAA que nous avons proposés sont quant à elles présentées à la Figure 3.9.c. On voit que les accumulateurs produisent des résultats parfaitement exact jusqu'à $i = 15$ à $i = 25$ (suivant la valeur de g), moment à partir duquel les résultats présentent une erreur relative non nulle qui est de plus en plus petite au fur et à mesure que g croît — le meilleur résultat étant obtenu pour $g = 24$ où l'erreur relative est de l'ordre de 10^{-11}.

3.4.2 Opérateurs de type TAA

Pour qu'un opérateur exploitant notre formulation de la TAA produise des résultats suffisamment exacts, il convient de disposer de mantisses larges. De larges mantisses ont tendance à ralentir la cadence de l'opérateur, surtout si l'accumulation doit se faire en un cycle et que l'addition ne peut être pipelinée. Nous couvrons au Chapitre 4 l'approche proposée pour résoudre ce problème. En attendant, nous reproduisons ici des résultats d'implémentation d'opérateurs de type TAA rapportés dans [85] qui permettent d'en apprécier la compacité.

Pour cette expérience, nous avons réalisé les opérateurs arithmétiques suivants : un MAC, un OPS2 et un OPS4. Dans un premier temps, l'opérateur est généré en utilisant des noyaux fournis par l'outil CoreGen de Xilinx (additionneurs et multiplieurs en VF). Ces opérateurs arithmétiques ont été générés avec une latence de 4 cycles par opération, de sorte à pouvoir atteindre une fréquence de fonctionnement de 200 MHz au moins. Dans un second temps, nous avons produit les mêmes opérateurs en utilisant la TAA pour effectuer toutes les additions voulues. Bien entendu, en utilisant la TAA, l'accumulation s'effectue en un cycle. Tous ces opérateurs utilisent un FAA(48, 3, 160). Tous les opérateurs produits ont été synthétisés puis implémentés en visant le FPGA Virtex 5 XC5VSX50T-3C [119] avec une contrainte

Tableau 3.3 – Occupation de surface et performance d'opérateurs à virgule flottante implémentés sur un FPGA Virtex 5 (XC5VSX50T-3C [119]) de Xilinx.

	Noyaux commerciaux					**TAA**		
Métriques	add.	mult.	MAC	OPS2	OPS4	MAC	OPS2	OPS4
Latence	4	4	8	12	16	8	9	10
Tranches	210	45	269	321	848	218	284	440
Registres	288	94	383	765	1,529	332	525	912
LUT	493	95	621	1,198	2,378	568	783	1,239
Blocs DSP	0	2	2	4	8	2	4	8
Période min. de clk (ns)	4.970	3.153	4.819	4.855	4.928	4.925	4.827	4.934
Fréq. max. de clk (MHz)	201.2	317.1	207.5	205.9	202.9	203.0	207.1	202.6

d'horloge de 5 ns, ce qui équivaut à une fréquence d'horloge de 200 MHz. Tous les opérateurs ont été synthétisés et implémentés en utilisant la version 10.1 du logiciel ISE de Xilinx.

Le Tableau 3.3 présente l'occupation de surface et la performance en vitesse des opérateurs. Il y apparaît clairement que les opérateurs de type TAA occupent moins de surface que leurs équivalents fait de noyaux commerciaux. Il en est de même pour les latences qui sont aussi importantes ou égales à celles obtenues avec les noyaux commerciaux. Dans le cas de l'OPS2, on constate que l'opérateur de type TAA est plus compact que son équivalent commercial d'environs 11 %. Il l'est de 49 % dans le cas de l'OPS4. Il convient de relever que la grande puissance de calcul de ce dernier opérateur est deux fois supérieure à celle d'un OPS2, quatre fois supérieure à celle d'un MAC.

3.5 Conclusion

Ce chapitre a permis de considérer en profondeur la problématique de la sommation en virgule flottante, de mesurer l'état de l'art sur le sujet et d'apprécier adéquatement les contributions que le travail de thèse a permis d'apporter. Nous avons ainsi montré les limites de la formulation de la TAA originale que proposait la littérature, élargi son applicabilité et démontré formellement et empiriquement la qualité des calculs que de tels opérateurs permettent d'obtenir. Nous avons aussi pu mesurer la compacité des opérateurs ainsi conçus. Le chapitre suivant portera sur la possibilité d'employer des mantisses très larges dans de tels opérateurs sans pour autant en compromettre les performances en fréquence.

CHAPITRE 4

UN SYSTÈME REDONDANT POUR L'ADDITION RAPIDE DE MANTISSES LARGES

4.1 Introduction

Le chapitre 3 a démontré le rôle joué par la largeur des mantisses dans l'exactitude des calculs effectués au moyen de l'algorithme généralisé d'auto-alignement. Notre objectif est d'être en mesure d'effectuer l'addition de mantisses larges en un cycle et à haute fréquence. L'addition en un cycle, comme nous avons pu le voir précédemment, a le mérite de permettre l'implémentation d'accumulateurs. De ce fait, et du fait de la haute fréquence d'opération visée, nous tentons d'améliorer les performances des engins de calculs que nous voulons développer dans ce travail de recherche, et ce sans en compromettre l'exactitude des résultats. Or, la performance des additionneurs est inversement proportionnelle à la largeur des opérandes. C'est pourquoi la solution retenue pour atteindre notre objectif exploite le concept d'additionneurs à temps constant, dont le principe repose sur la représentation redondante des nombres.

Ce chapitre traite de cette question en trois temps. Dans un premier temps, la problématique du chemin critique de l'additionneur est introduite, le principe de la représentation redondante comme solution au problème est présenté et la solution retenue pour notre travail brièvement justifiée. Dans un second temps, le système redondant choisi est plus formellement défini, ses opérateurs étudiés et les contributions de notre travail de recherche présentées, soit *i*) un additionneur redondant endomorphique et *ii*) un convertisseur redondant à format conventionnel. Le troisième et dernier temps de ce chapitre consiste à évaluer empiriquement les performances de nos opérateurs sur FPGA.

4.2 Problématique

4.2.1 Chemin critique de l'additionneur

L'additionneur à propagation de retenue (APR) — *ripple carry adder* (RCA) en anglais — est un opérateur matériel dont la fonction est d'additionner deux nombres entiers. Il est construit suivant une conception dite itérative qui exploite une cellule appelée additionneur complet, *full adder* (FA) en anglais. Un additionneur complet permet d'additionner trois bits : deux bits opérandes et un bit de retenue. En retour, la cellule produit un bit de somme

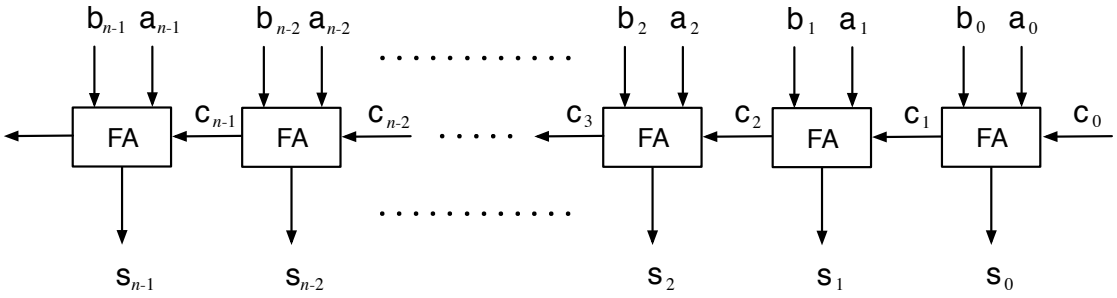

Figure 4.1 – Additionneur construit en chaînant plusieurs cellules d'additionneurs complets.

et un bit de retenue de poids supérieur. Les cellules sont connectées les unes aux autres au moyen des bits de retenue, formant ainsi ce qu'on appelle une chaîne de retenues, tel qu'illustré à la Figure 4.1 où est considéré un RCA à n bits.

La rapidité d'un RCA dépend de la longueur de la chaîne de retenues, et par conséquent du nombre de bits des opérandes. La fréquence d'opération maximale de l'additionneur est par conséquent inversement proportionnelle à la largeur des opérandes. L'additionneur étant un circuit de grande utilité pour les systèmes numériques, les FPGA disposent de chaînes de retenues gravées en dur (voir Figure 4.2) dont le but est d'une part de limiter l'utilisation de la logique reconfigurable dans la réalisation des additionneurs, et d'autre part d'en améliorer les performances. Ainsi, les FPGA modernes (tels que le Virtex 5 de Xilinx) sont en mesure de réaliser des additions sur des opérandes à 64 bits à une fréquence de plus de 400 MHz [12].

4.2.2 L'addition redondante

Il existe différentes stratégies pour réduire le chemin critique d'un additionneur, les plus connues étant probablement celles du circuit à anticipation de retenue (*carry-lookahead adder*) et du circuit à sélection de retenue (*carry-select adder*) [29]. Au début des années 1960, Algirdas Avizienis propose un système de représentation des nombres qui permet d'effectuer l'addition d'entiers en temps constant (c.-à-d. indépendamment de n) en exploitant la redondance de la représentation des nombres [3]. Pour ce faire, l'ensemble des chiffres utilisés est élargi de sorte à enrichir les représentations possibles d'une même quantité. Par exemple, en admettant l'ensemble de chiffres $\{-1, 0, 1\} \equiv \{\bar{1}, 0, 1\}$ et une représentation à quatre chiffres, il est possible de représenter la valeur $+3$ de cinq manières différentes : 0011, $010\bar{1}$, $01\bar{1}1$, $1\bar{1}0\bar{1}$ et $1\bar{1}\bar{1}1$.

Les codes redondants sont abondamment utilisés dans la pratique : on les retrouve par exemple dans la réalisation des multiplieurs signés [8]. Au début des années 1990, Behrooz Parhami suggère une généralisation du concept de représentation redondante des nombres

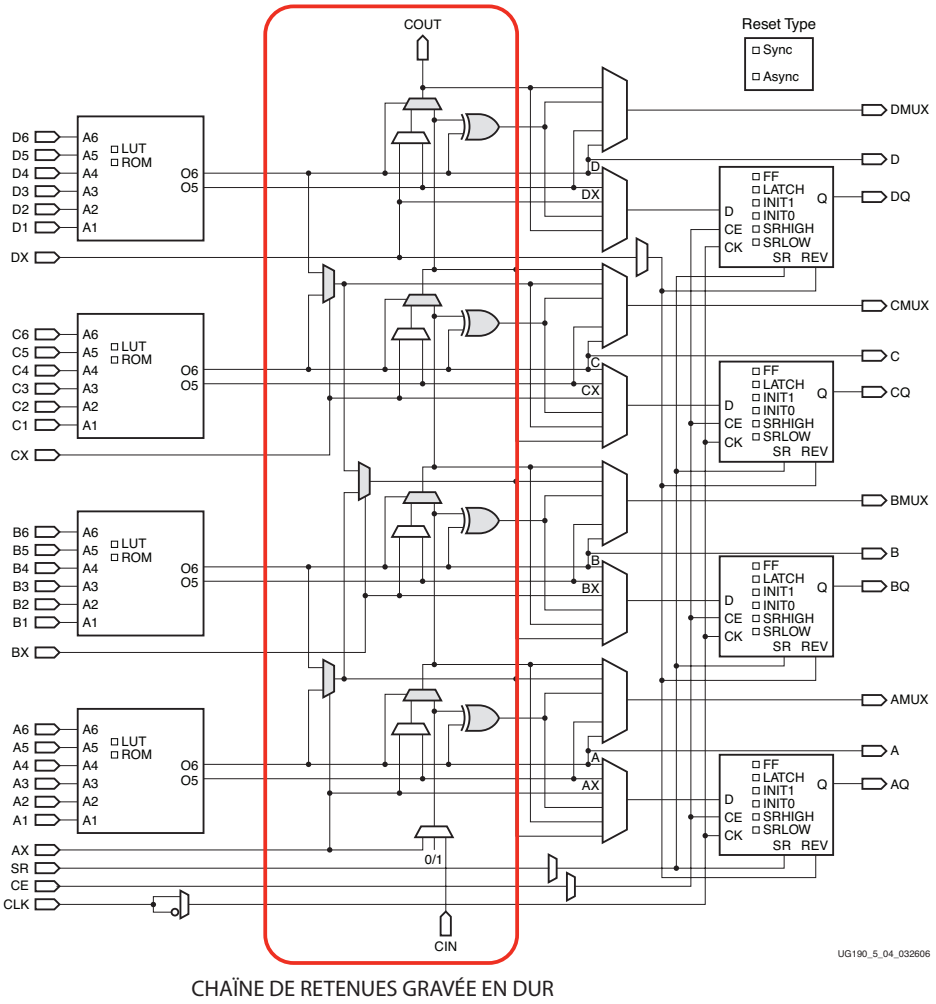

Figure 4.2 – Tranche (*slice*) d'un Virtex 5, avec mise en évidence de la chaîne de retenues gravée en dur. Source : [121].

et établit pour ce faire une classification et une taxonomie des systèmes de numération positionnels [92]. Cette classification identifie deux types distincts d'ensembles de chiffres : 1) les ensembles de chiffres positifs dits à sauvegarde de retenues, *carry-save* (CS) en anglais ; et 2) ensembles de chiffres signés, *signed-digit* (SD) en anglais, dont l'exemple utilisé précédemment (c.-à-d. $\{\bar{1}, 0, 1\}$) fait partie. Il a été établi que ces deux types d'ensembles sont théoriquement équivalents du point de vue matériel, affichant des propriétés communes en termes d'occupation de surface et de consommation de puissance [79]. Il convient cependant de relever une particularité des systèmes de type CS puisque ces derniers admettent une réalisation matérielle employant des blocs standards de l'additionneur binaire, notamment la cellule FA et des RCA [59, 113, 50], ce qui n'est pas le cas des systèmes de type SD.

4.2.3 Additionneurs à virgule flottante avec système redondant

Les circuits arithmétiques à virgule flottante utilisent la représentation redondante pour accélérer ses multiples sous-opérations. C'est là une méthode de conception dont la paternité est attribuée aux travaux de P. Michael Farmwald [33]. La redondance dans ce contexte peut se manifester à différents niveaux de la représentation interne des opérandes : on peut la retrouver au niveau de la représentation de la mantisse [31, 32], au niveau de la représentation de l'exposant [51], ou encore dans la représentation des cas spéciaux de l'arithmétique flottante, c.-à-d. les symboles NaN, Inf, Zero [19]. Le goulot d'étranglement d'un opérateur arithmétique se situe le plus souvent dans la manipulation de la mantisse. La redondance de cette dernière revêt par conséquent une importance primordiale.

La littérature rapporte différents systèmes redondants utilisés pour représenter la mantisse. Le système CS en base 2 est évoqué dans la réalisation d'accumulateurs [69], de multiplieur accumulateur (*multiplier-accumulator* — MAC) [116] et de multiplieur-additionneur fusionnés (*fused multiply-add* — FMA) [61, 109]. L'utilisation du système SD dans la littérature est concomitante à celle de bases élevées ($\gg 2$) [31, 32, 51]. L'intérêt d'utiliser une base élevée provient de l'économie de registres ainsi obtenue. Dans le contexte d'une implémentation FPGA, cet argument est tout aussi valable, mais il est alors préférable d'utiliser un système CS à base élevée (*high-radix carry-save* — HRCS) afin d'exploiter au mieux les chaînes de retenues gravées en dur [18]. La Section 4.3 présente ce format et couvre les innovations que ce travail doctoral a permis d'y adjoindre. Ces dernières seront discutées à la lumières de résultats expérimentaux à la Section 4.4.

4.3 Le système HRCS

4.3.1 Illustration du principe du système HRCS

Avant de considérer le système HRCS en profondeur et de manière plus formelle, il est intéressant d'illustrer son fonctionnement par un exemple pédagogique. Supposons un système de représentation des nombres utilisant la base $b = 4$. En admettant que le système utilise cinq chiffres, les valeurs admissibles vont de 0 à $\sum_{i=0}^{4} (b-1) \cdot b^i = \sum_{i=0}^{4} 3 \cdot 4^i = 1023$.

$$
\begin{array}{rcc}
 & & {\scriptstyle 1\ 1\ 1\ 1} \\
x_1 & & 1\,2\,3\,3\,2 \\
x_2 & + & 0\,3\,3\,0\,2 \\
\hline
\Sigma & = & 2\,2\,3\,0\,0
\end{array}
\qquad
\begin{array}{rcc}
 & & \\
x_1 & & 1\,2\,3\,3\,2 \\
x_2 & + & 0\,3\,3\,0\,2 \\
\hline
s_1 & & 1\,1\,2\,3\,0 \\
c_1 & = & 1\,1\,0\,1\,0
\end{array}
\qquad
\begin{array}{rcc}
 & & \\
s_1 & & 1\,1\,2\,3\,0 \\
c_1 & & 1\,1\,0\,1\,0 \\
x_3 & + & 1\,0\,2\,2\,1 \\
\hline
s_2 & & 3\,2\,0\,2\,1 \\
c_2 & = & 0\,1\,1\,0\,0
\end{array}
$$

(a) (b) (c)

Figure 4.3 – Illustration de l'addition en HRCS.

L'exemple de la Figure 4.3.a montre comment il est possible d'additionner en base 4 les quantités $x_1 = 12332_{(4)} \equiv 446_{(10)}$ et $x_2 = 03302_{(4)} \equiv 242_{(10)}$. L'addition donne $22300_{(4)} \equiv 688_{(10)}$ et implique une propagation de la retenue de la position #1 à la position #4. En utilisant un système à sauvegarde de retenues (CS) pour effectuer cette addition, il est possible de briser la chaîne de propagation et de réduire le chemin critique du circuit. Pour ce faire, le résultat est conservé sous la forme de paires de chiffres incluant une somme et une retenue. Ces paires sont évaluées à chacune des positions des nombres en entrée. Dans l'exemple de la Figure 4.3.b, la retenue à position donnée ne dépend pas des retenues aux positions précédentes (par exemple la retenue à la position #2 ne dépend pas de la retenue à la position #1 ni de celle à la position #0). Ainsi, les chiffres à la position #0 (2 et 2) donnent une somme de 0 et une retenue (de poids supérieur) de 1 ; ceux à la position #1 (3 et 0) donnent une somme de 3 et une retenue de 0 ; ceux à la position #2 (3 et 3) donnent une somme de 2 et une retenue de 1, etc.. Ainsi, le résultat HRCS de l'addition s'exprime comme l'ensemble des paires (somme, retenue) suivant : $((1,1),(1,1),(2,0),(3,1),(0,0)) \equiv 11230_{(4)} + 11010_{(4)} \equiv 688$.

Un des grands intérêts dans l'utilisation du format HRCS est que le résultat peut être réutilisé pour effectuer une nouvelle addition sans conversion préalable vers le format non redondant. Par exemple, nous pourrions ajouter la quantité $x_3 = 10221_{(4)} \equiv 297_{(10)}$ au résultat HRCS de l'addition de x_1 et x_2, c'est-à-dire (s_1, c_1). Cette opération est illustrée à la Figure 4.3.c et donne $((3,0),(2,1),(0,1),(2,0),(1,0)) \equiv 32021_{(4)} + 01100_{(4)} \equiv 905_{(10)} + 80_{(10)} =$

$985_{(10)} = 688_{(10)} + 297_{(10)}$. Cette propriété fait que le système HRCS est avantageux pour la réalisation d'accumulateurs à haute performance, puisque le parallélisme qu'il introduit est au niveau de chaque chiffre, indépendamment du nombre total de chiffres utilisés (c'est précisément là le principe d'un additionneur à temps constant).

Bien entendu, en aval des opérations, le résultat HRCS devra être converti vers un format non redondant. Pour ce faire, il existe différentes techniques permettant cette conversion. Nous couvrirons à la Section 4.3.4 ces techniques, section où nous présenterons également la technique proposée dans ce travail.

4.3.2 Définition du format HRCS

Nous nous intéressons ici au format HRCS en base 2^k. Ce choix de base est motivé par le fait que les chaînes de retenues gravées en dur du FPGA sont exploitées au meilleur de leur potentiel lorsque la base est une puissance de 2. La notation utilisée ici est inspirée de celle présentée dans [6, 7]. Cependant, il convient de relever que ces articles traitaient d'un système à base variable dit à redondance maximale dans la taxonomie de Parhami [92], un système qui plus est servait à la représentation d'entiers naturels, c.-à-d. d'entiers non signés. Le système HRCS utilisé dans notre travail est quant à lui à redondance minimale, il utilise une base fixe et sert à la représentation d'entiers relatifs, c.-à-d. d'entiers signés.

Soit A un entier à n chiffres représenté dans un système HRCS en base 2^k :

$$\begin{aligned} A &= (A_{n-1}, A_{n-2}, \ldots, A_0) \\ &= \left(\left(A_{n-1}^{(s)}, A_{n-1}^{(c)}\right), \left(A_{n-2}^{(s)}, A_{n-2}^{(c)}\right), \ldots, \left(A_0^{(s)}, A_0^{(c)}\right)\right) \end{aligned} \tag{4.1}$$

Chacun des chiffres A_i ($0 \leq i < n$) de A est formé de mots de somme (*sum words*) de k bits — $A_i^{(s)} = A_i^{k-1} A_i^{k-2} \ldots A_i^0$ — et d'un bit de retenue — $A_i^{(c)}$. La valeur entière a_i associée au chiffre A_i est donnée par :

$$a_i = a_i^{(s)} + a_i^{(c)} = \sum_{j=0}^{k-1} A_i^j 2^j + A_i^{(c)} \tag{4.2}$$

Ainsi, chaque chiffre A_i admet $2^k + 1$ valeurs différentes a_i. Par conséquent, le cardinal de l'ensemble des chiffres est $2^k + 1$, soit la base plus un. Cette spécificité en fait un système à redondance minimale [92]. A peut être signé ou non signé. Soit $\tilde{a} = \sum_{i=0}^{n-1} a_i 2^{ik} \bmod 2^{nk}$. Si A est non signé, alors $a = \tilde{a}$. Si A est signé, sa valeur est donnée par la convention de complément à 2^k, c.-à-d. que $a = \tilde{a}$ si $\tilde{a} < 2^{nk-1}$, autrement $a = \tilde{a} - 2^{nk}$.

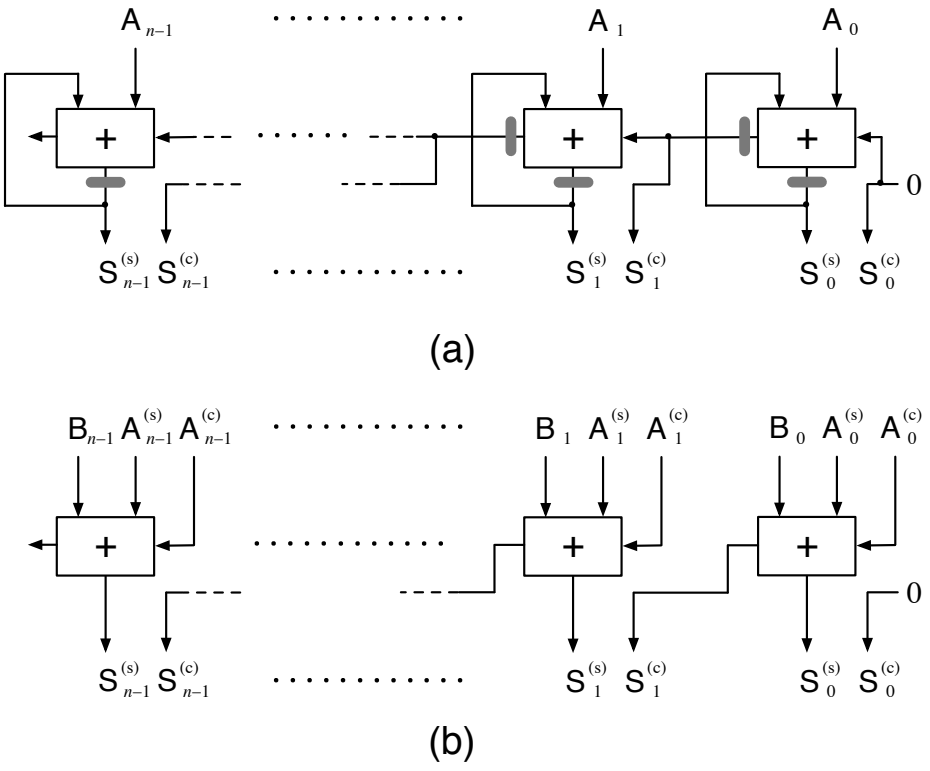

Figure 4.4 – (a) Accumulateur avec sortie HRCS (S) ; (b) Additionneur HRCS hétérogène.

4.3.3 Additions dans le système HRCS

L'intérêt d'utiliser le système HRCS est qu'il permet l'implémentation efficace d'accumulateurs larges. L'accumulateur de la Figure 4.4.a sert à accumuler une entrée en format non redondant de nk bits (ou, vu autrement, des entiers de n chiffres en base 2^k). Une telle topologie permet de réduire le chemin critique de l'accumulateur résultant, passant du traversement de nk FA au simple traversement de k FA. Elle est également très attractive dans un contexte FPGA car elle utilise efficacement les chaines de retenues [7, 18].

Une topologie similaire à celle de l'accumulateur peut être utilisée pour réaliser un additionneur que nous qualifierons d'hétérogène, tel que celui de la Figure 4.4.b. Un additionneur hétérogène accepte en entrée un entier représenté en format conventionnel et un second entier représenté en format HRCS. Il produit en retour un résultat en HRCS.

Une lacune du format HRCS est l'absence d'un additionneur endomorphique, c'est-à-dire d'un additionneur prenant en entrée deux nombres en HRCS et produisant une sortie en HRCS. L'intérêt que nous voyons dans l'utilisation d'un additionneur endomorphique est qu'il permet l'implémentation efficace d'un arbre d'additionneurs, d'autant plus si cet arbre est terminé par un accumulateur, tel que dans l'OPS discuté précédemment. Cependant, il

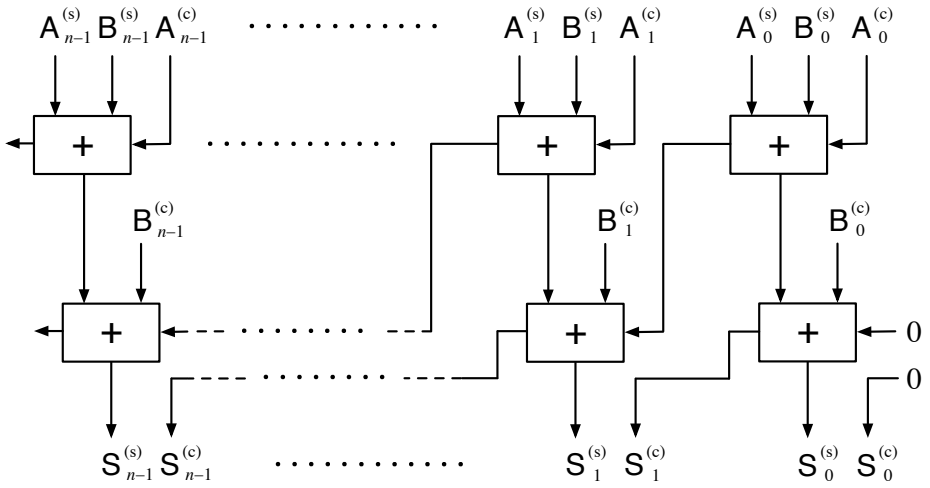

Figure 4.5 – Additioneur endomorphique à deux chaînes de retenues.

n'est pas aisé de concevoir un tel additionneur du fait que l'addition de deux chiffres HRCS peut produire une valeur maximale de $2\max(a_i) = 2(1+(2^k-1)) = 2^{k+1}$, qui est supérieure à la valeur maximale d'un mot de somme et un bit de retenue de poids supérieur, c.-à-d. $\max(a_i^{(s)} + a_{i+1}^{(c)}) = (2^k-1) + 2^k = 2^{k+1} - 1$. Il est possible de pallier à ce problème en employant deux additionneurs hétérogènes, tel qu'indiqué à la Figure 4.5. Cependant, cette solution double le chemin critique, qui passe du traversement de k FA à $2k$ FA.

Notre travail a permis d'établir qu'il est possible de réaliser un additionneur endomorphique à une seule chaîne de retenues au prix de quelques manipulations logiques. La Figure 4.6 présente une vue détaillée des opérations de transformation nécessaires pour parvenir à ce but. Dans cet exemple, les deux opérandes A et B sont des nombres HRCS ayant pour paramètres $n = k = 4$. La transformation est opérée en trois étapes comme suit.

À la première étape, on applique une transformation de type somme et retenue aux $k-1$ bits de poids supérieur des mots de somme $A_i^{(s)}$ et $B_i^{(s)}$, tel qu'indiqué à la Figure 4.6 sous l'intitulé ÉTAPE 1. La transformation consiste à effectuer la somme des paires de bits et de sauvegarder la retenue qui en résulte, ce qui est réalisé par un demi-additionneur et se traduit par la relation logique suivante :

$$\begin{cases} V_i^j = A_i^j \oplus B_i^j, & 0 < j < k \\ W_i^{j+1} = A_i^j \cdot B_i^j, & 0 < j < k-1 \\ \alpha_{i+1} = A_i^j \cdot B_i^j, & j = k-1 \end{cases} \tag{4.3}$$

La seconde étape consiste à prendre les bits aux positions des bits de retenue et organisées en colonnes de 4 à 5 bits sur la Figure 4.6. Chacune de ces colonnes peut être

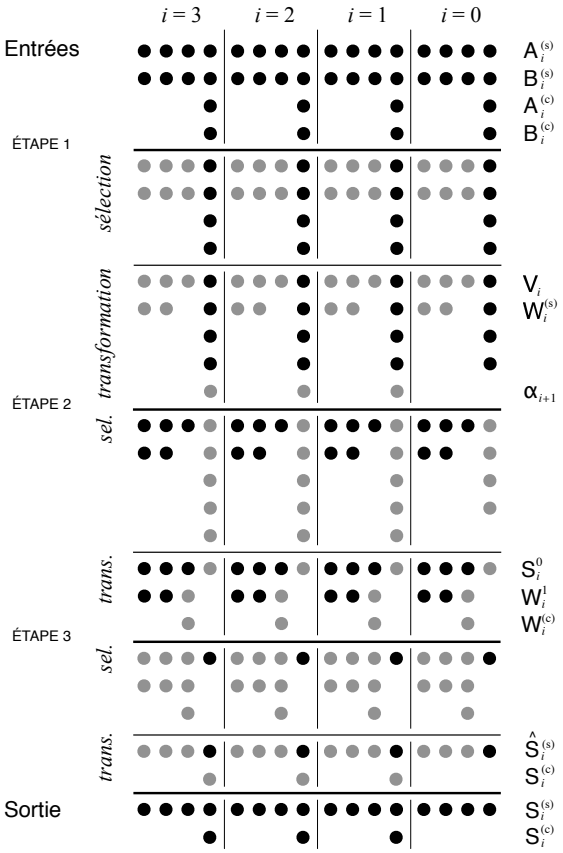

Figure 4.6 – Représentation du détail des transformations permettant de réaliser une addition endomorphique à une seule chaîne de retenues.

remplacée par trois bits équivalents, dont deux de poids supérieurs, tel qu'indiqué sous l'intitulé ÉTAPE 2. Il existe différentes relations logiques pour traduire cette transformation du fait de l'interchangeabilité des deux bits de poids supérieur. Nous proposons d'utiliser la relation suivante :

$$\begin{cases} \beta_i & = & A_i^0 \oplus B_i^0 \oplus A_i^{(c)} \oplus B_i^{(c)} \\ \gamma_i & = & A_i^0 \cdot B_i^0 \cdot A_i^{(c)} \cdot B_i^{(c)} \\ W_i^{(c)} & = & \alpha_i \cdot \beta_i + \gamma_i \\ W_i^1 & = & (A_i^0 + B_i^0 + A_i^{(c)}) \\ & & (A_i^0 + B_i^0 + B_i^{(c)}) \\ & & (A_i^0 + A_i^{(c)} + B_i^{(c)}) \\ & & (B_i^0 + A_i^{(c)} + B_i^{(c)}) \\ S_i^0 & = & \alpha_i \oplus \beta_i \end{cases} \tag{4.4}$$

où α_i est forcé à 0 lorsque $i = 0$.

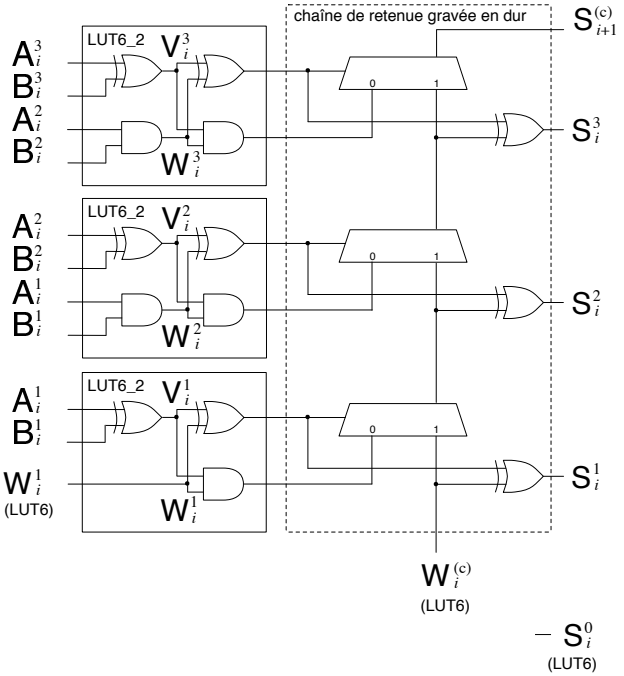

Figure 4.7 – Réalisation des transformations menant à l'additionneur endomorphique à simple chaîne de propagation en exploitant les LUT à six entrées et à deux sorties.

À la troisième étape, les résultats des transformations des deux étapes précédentes sont combinées pour effectuer la propagation de la retenue. On somme pour ce faire le bit de retenue $W_i^{(c)}$ et les deux mots de somme V_i et $W_i^{(s)}$ — $V_i = V_i^{k-1} \dots V_i^1$ et $W_i^{(s)} = W_i^{k-1} \dots W_i^1$. Cette addition produit alors un mot de somme partiel $\hat{S}_i^{(s)} = S_i^{k-1} \dots S_i^1$ et un bit de retenue de poids supérieur $S_{i+1}^{(c)}$. Le mot de somme du résultat final $S_i^{(s)}$ est obtenu par concaténation :

$$S_i^{(s)} = \hat{S}_i^{(s)} S_i^0 \tag{4.5}$$

La Figure 4.7 illustre comment un tel additionneur endomorphique est associé aux primitives des FPGA modernes, soit la LUT à six entrées et à deux sorties, ainsi que la chaîne de propagation qui permet de réaliser efficacement un RCA. La LUT à six entrées et deux sorties permet d'implémenter une fonction à six variables ou deux fonctions à cinq variables (les variables sont communes aux deux fonctions). Pour des raisons de lisibilité, nous considérons dans cet exemple $k = 4$.

On relèvera que les signaux W_i^j et V_i^j sont implicitement générés au sein même des LUT. Ils sont ensuite dirigés vers un demi-additionneur, dont les deux sorties (somme et retenue) vont alimenter la chaîne de retenues gravée en dur. Quant aux signaux W_i^1, $W_i^{(c)}$ et S_i^0, il est clair, en étudiant leur expression donnée à l'équation 4.4, qu'il s'agit de fonctions à six

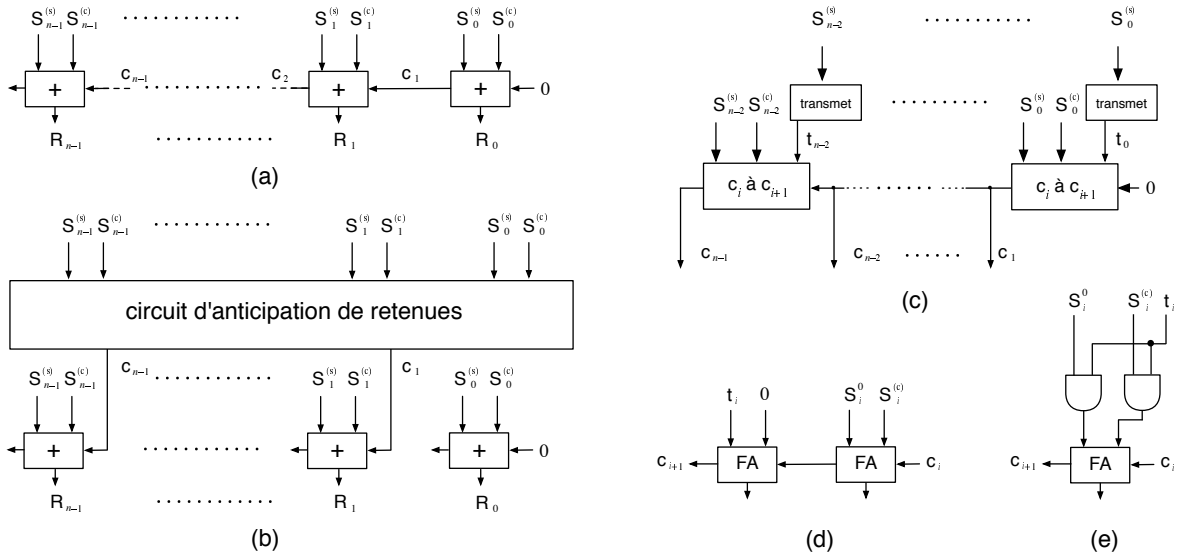

Figure 4.8 – (a) Convertisseur HRCS à format conventionnel utilisant un large RCA ; (b) Convertisseur HRCS à format conventionnel utilisant un circuit d'anticipation de retenues ; (c) Topologie du circuit d'anticipation de retenues ; (d) Implémentation possible de la cellule c_i à c_{i+1} ; (e) Implémentation alternative de la cellule c_i à c_{i+1}, plus avantageuse pour les FPGA disposant de LUT à six entrées.

variables et moins, et que chacune d'elles peut être implémentée dans une LUT à 6 variables. À la Section 4.4.2, nous montrons que les FPGA permettent de réaliser l'additionneur endomorphique de façon efficace.

4.3.4 Conversion

Nous avons pu le voir précédemment, les entiers représentés en format HRCS doivent tôt ou tard être convertis vers un format conventionnel. Il existe différentes façons de réaliser une telle conversion. Dans le cas où cette conversion est effectuée à la fin d'un accumulateur tel que celui de la Figure 4.4.a, il est possible de dédier n cycles à la fin de l'accumulation pour propager les bits de retenue durant lesquels des entrées nulles alimentent l'opérateur [18]. Une telle solution a l'avantage de ne nécessiter aucun surcoût matériel, mais elle impose une pénalité temporelle qui est difficilement tolérable dans notre contexte d'application, étant données les contraintes de la simulation en temps réel auxquelles sont assujettis les engins de calcul que nous voulons développer.

Une autre solution existe pour réaliser la conversion et consiste à utiliser un long RCA de nk bits pour sommer les différentes paires de $S_i^{(s)}, S^{(c)i}$, tel qu'indiqué à la Figure 4.8.a. Un tel convertisseur peut être vu comme n RCA de k bits connectés les uns aux autres au

moyen de bits de retenue. Chaque RCA à la position $\#i$ reçoit un bit de retenue c_i de l'étage précédent ($c_0 = 0$) et produit un mot de somme R_i ainsi qu'un bit de retenue c_{i+1} de poids supérieur qu'il achemine au RCA à la position $\#i + 1$. Là encore, la vitesse de la conversion est limitée par la longueur de la chaîne de retenues. Il est bien entendu possible de *pipeliner* le RCA et d'effectuer la conversion en n cycles. Mais alors, en plus de la pénalité temporelle, similaire à celle précédemment évoquée pour la solution de [18], vient s'ajouter un surcoût matériel.

La solution que nous préconisons part du RCA de la Figure 4.8.a. Elle brise la chaîne de propagation au moyen d'un circuit d'anticipation des retenues c_i, tel qu'illustré à la Figure 4.8.b. Une telle approche est classique dans la réalisation des additionneurs rapides. L'originalité ici vient de la topologie utilisée pour réaliser le circuit d'anticipation, schématisée à la Figure 4.8.c. où des fonctions logiques appelées "transmet" génèrent des signaux notés t_i et décrits par la relation :

$$t_i = \underset{j=1}{\overset{k-1}{\text{AND}}} S_i^j, \qquad 0 \leq i < n-1 \tag{4.6}$$

Les blocs "c_i à c_{i+1}" qui sont chaînés produisent les bits de retenue c_{i+1} en utilisant la relation :

$$c_{i+1} = t_i S_i^0 S_i^{(c)} + t_i c_i (S_i^0 \oplus S_i^{(c)}), \quad 0 \leq i < n-1 \tag{4.7}$$

Les FPGA ont de la difficulté à réaliser des fonctions logiques à multiples entrées et multiples sorties avec l'efficacité espérée d'un circuit d'anticipation de retenues. Néanmoins, comme nous avons pu l'évoquer à maintes reprises jusqu'ici, ces mêmes FPGA disposent de chaînes de retenues gravées en dur qui peuvent être exploitées adéquatement pour réaliser de telles fonctions logiques avec la célérité désirée, en s'assurant que lesdites fonctions puissent être exprimées d'une façon où l'usage de la chaîne de retenues est explicité. C'est précisément ce qui est fait dans les Figure 4.8.d et Figure 4.8.e, où nous proposons deux implémentations possibles de la cellule "c_i à c_{i+1}". La cellule de la Figure 4.8.d vise une implémentation sur les FPGA des générations précédentes (Virtex II Pro, Spartan 3) qui offrent des LUT à quatre entrées seulement. Notons qu'une cellule similaire a été proposée dans [16] pour la réalisation sur FPGA de RCA pipelinés. La cellule de la Figure 4.8.e peut quant à elle être exploitée sur les FPGA disposant de LUT à six entrées et résulte en une économie d'occupation de surface et de meilleures performances en vitesse.

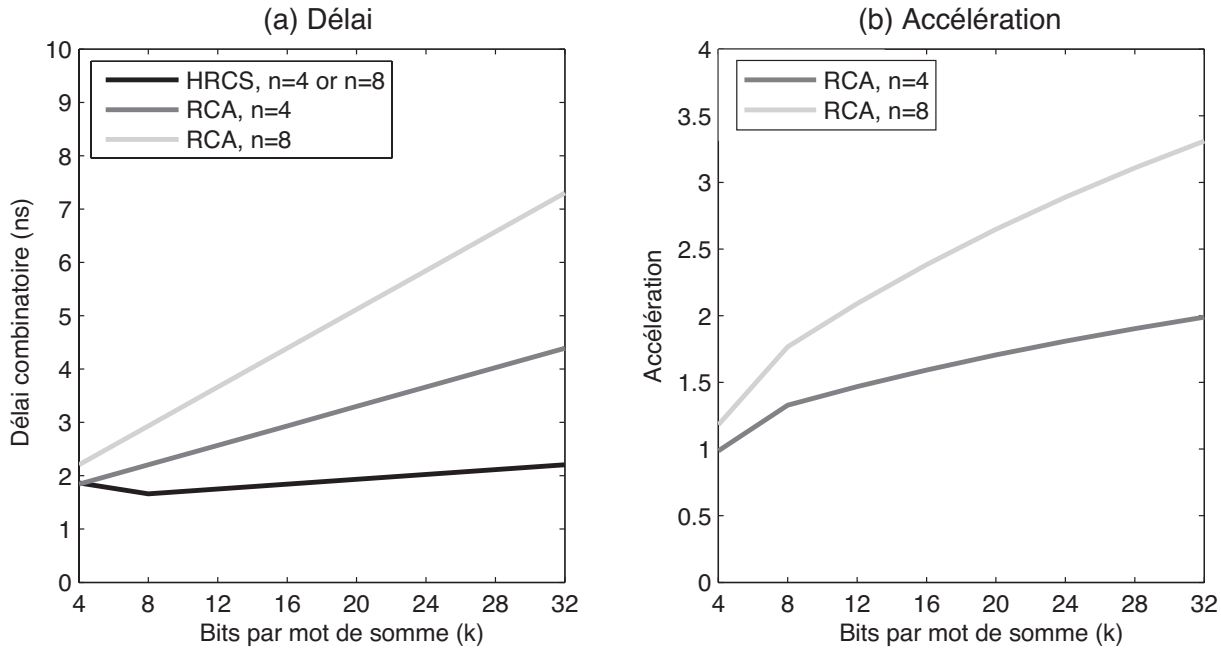

Figure 4.9 – Comparaison du délai de l'additionneur hétérogène et du RCA.

4.4 Résultats

4.4.1 Additionneur hétérogène

Afin de donner une idée de la réduction du chemin critique obtenue en employant l'additionneur hétérogène de Figure 4.4.b, nous avons comparé les performances de ce dernier à celles d'un RCA de taille nk en variant la valeur de k entre 4 et 32, et ce pour des valeurs de $n \in \{4, 8\}$. Figure 4.9 compare les longueurs des chemins combinatoires de ces additionneurs telles qu'estimées par la version 13.1 du logiciel ISE, où le Virtex 5 XC5VSX50T-3C de Xilinx est le FPGA cible.

Sans grande surprise, il apparaît que le délai de l'additionneur hétérogène ne varie pas avec n et ne dépend que de k suivant une relation linéaire, tandis le délai du RCA croît linéairement avec nk. Il est intéressant de relever que la Figure 4.9.a indique que la réalisation d'un additionneur hétérogène avec $k = 4$ donne des résultats sous optimaux puisque le délai est presque aussi important que celui obtenu pour $k = 16$. Cette observation s'explique par le fait que le synthétiseur subit un effet de bord dans l'implémentation de petits additionneurs où il n'exploite pas la chaîne de retenues.

Nous rapportons à la Figure 4.9.b le gain en vitesse obtenu en passant d'un RCA à l'additionneur hétérogène. On note que l'accélération obtenue est dans l'intervalle $1,32 - 1,99$

pour $n = 4$ et dans l'intervalle $1,76 - 3,31$ pour $n = 8$, au lieu des facteurs 4 et 8 respectivement escomptés. Cette observation s'explique par le fait que le délai de l'additionneur endomorphique est donné par $\tau_{\text{lut}} + k\tau_{\text{cc}}$, alors que le délai du RCA est donné par $\tau_{\text{lut}} + nk\tau_{\text{cc}}$, où τ_{lut} est le délai de traversement d'une LUT et τ_{cc} est le délai unitaire (c.-à-d. par bit) du traversement de la chaîne de retenues. On obtient par conséquent une accélération exprimée par :

$$\frac{\tau_{\text{lut}} + nk\tau_{\text{cc}}}{\tau_{\text{lut}} + k\tau_{\text{cc}}} = 1 + \frac{k\tau_{\text{cc}}}{\tau_{\text{lut}} + k\tau_{\text{cc}}}(n-1) \tag{4.8}$$

Le terme à droite de l'équation 4.8 se lit comme suit : plus le ratio $k\tau_{\text{cc}}/(\tau_{\text{lut}} + k\tau_{\text{cc}})$ s'approche de 1.0 (100%), plus l'accélération s'approche de l'optimum théorique. Nos résultats indiquent que pour des valeurs de k dans l'intervalle $8 - 32$, le coefficient $k\tau_{\text{cc}}/(\tau_{\text{lut}} + k\tau_{\text{cc}})$ croît progressivement de 11% à 33% ; il ne dépasse 50% que pour des valeurs de $k > 64$. Bien que cette observation relativise l'efficacité de l'approche HRCS, elle ne remet pas en question son utilisation. Le système HRCS demeure une solution très attractive comme le démontre clairement les estimés de délai de la Figure 4.9.a. Il convient simplement d'être conscient des ordres de grandeur des gains obtenus.

4.4.2 Additionneur endomorphique

Figure 4.10 compare l'occupation de surface et le délai de l'additionneur hétérogène et des additionneurs endomorphiques, celui à deux chaînes de retenues (voir Figure 4.5) et celui à chaîne de retenues unique, désignés respectivement par "double" et "simple". Les Figures 4.10.a et 4.10.b donnent les valeurs en absolu, tandis que les Figures 4.10.c et 4.10.d reproduisent ces valeurs en relatif, où la référence est l'additionneur hétérogène. Si on écarte les effets de bord du synthétiseur pour $k = 4$, il apparaît que la double chaîne de retenues impose un facteur 2 à l'occupation de surface et au délai du chemin combinatoire. La chaîne de retenues simple quant à elle ne requiert qu'un supplément de 15% à 35%, tant en surface qu'en délai. Ce délai supplémentaire et l'occupation de surface sont entièrement attribuables aux trois LUT implémentant les fonctions W_i^1, $W_i^{(c)}$ et S_i^0. C'est pourquoi le surplus baisse quand k croît. Il convient de relever ici que l'additionneur endomorphique à une chaîne de retenues est d'autant plus attrayant que son utilisation permet d'éviter de convertir une des deux opérandes vers un format conventionnel, d'équilibrer plus facilement les étages de pipeline dans une topologie d'arbre d'additionneurs, tout en réduisants sa profondeur.

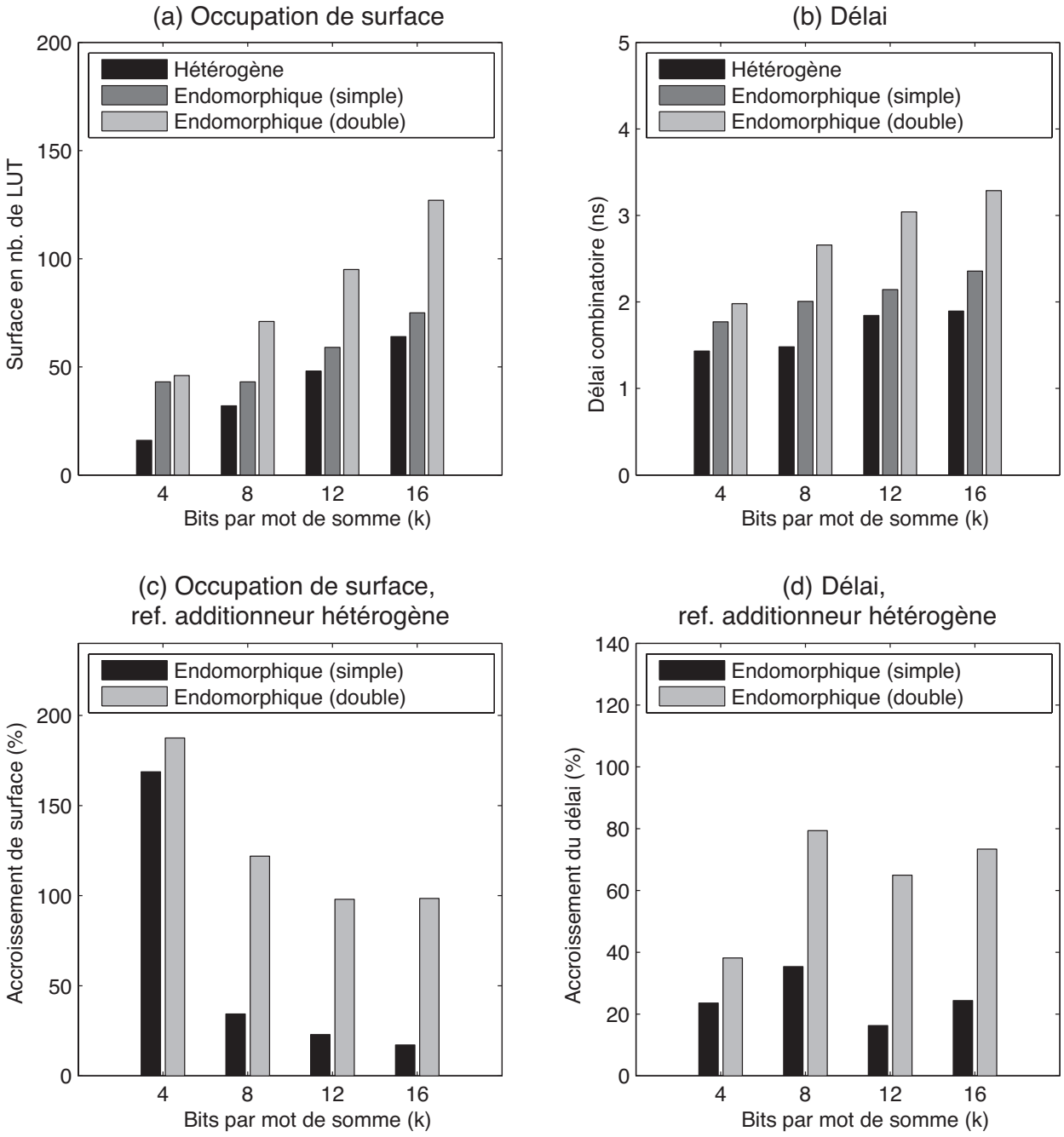

Figure 4.10 – Comparaison des additionneurs hétérogène et endomorphique.

4.4.3 Convertisseur HRCS à format conventionnel

Afin d'évaluer les performances du convertisseur HRCS à format conventionnel, nous avons implémenté trois architectures de convertisseur de façon purement combinatoire. La Figure 4.11 donne l'occupation de surface et le délai associé à chacune de ces architectures.

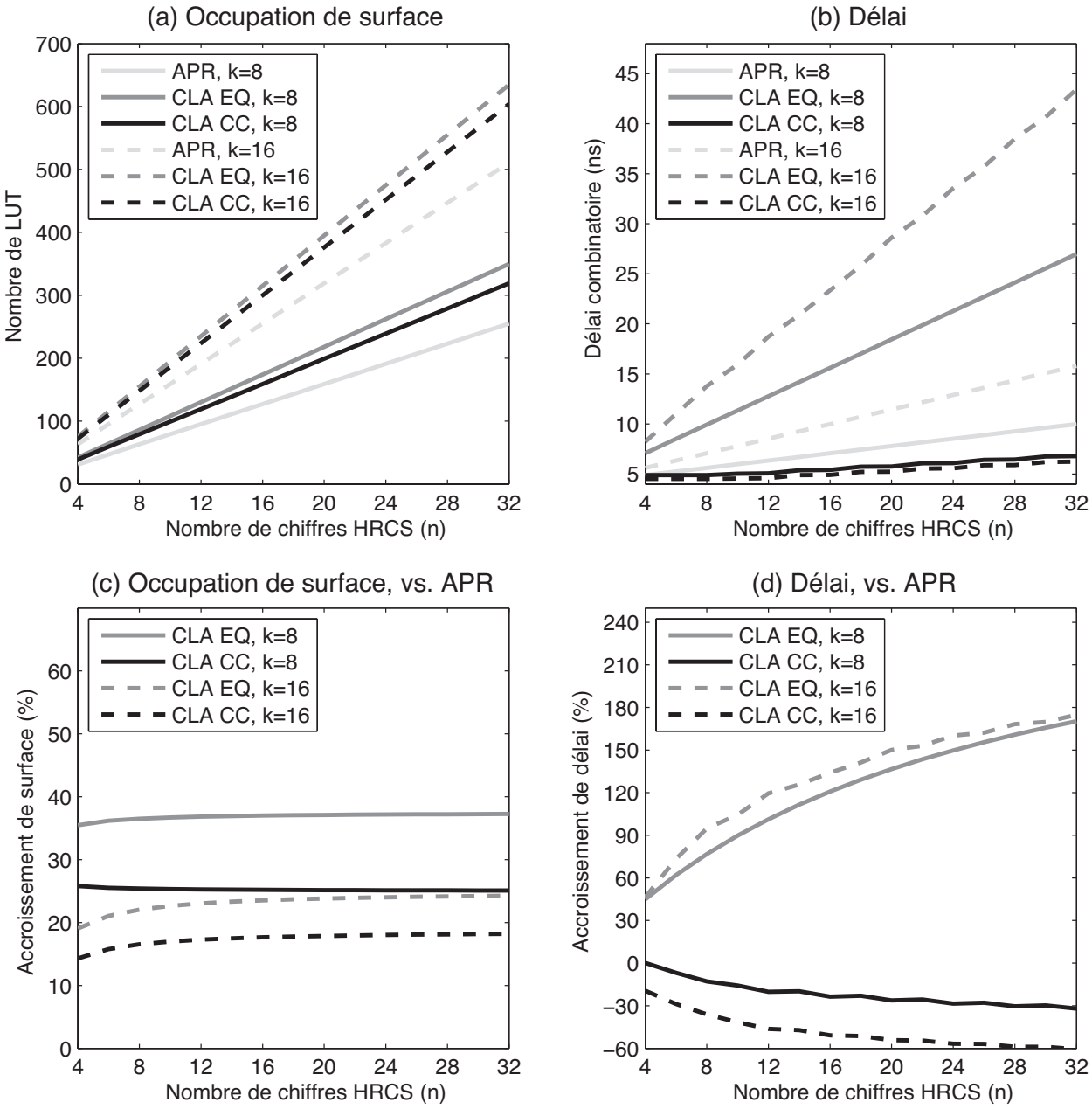

Figure 4.11 – Comparison de trois types de convertisseur HRCS à format conventionnel.

- Le convertisseur noté "RCA' est celui consistant d'un long RCA de nk bits, tel que celui de la Figure 4.8.a ;
- Le convertisseur noté "CLA EQ" est celui utilisant l'anticipation des retenues, où la cellule "c_i à c_{i+1}" est réalisée en transcrivant en VHDL l'équation 4.7 ;
- Le convertisseur noté "CLA CC" est celui utilisant l'anticipation des retenues, où la cellule "c_i à c_{i+1}" est réalisée en utilisant la cellule de la Figure 4.8.e.

Pour cette expérience, le nombre de bits par mot de somme (k) a été fixé successivement à 8 et à 16, soit des valeurs raisonnables pour une implémentation où la performance en vitesse est visée. Pour chacune de ces valeurs, le nombre de chiffres (n) a été varié de 4 à 32, ce qui correspond à une largeur des registres (nk) allant de 16 à 512.

L'option "RCA" démontre un comportement linéaire parfait, tant du point de vue du délai combinatoire que de l'occupation de surface. Il en est de même avec l'option "CLA EQ", à la différence que le délai et l'occupation de surface sont ici supérieurs à l'option "RCA", contrairement à ce qui serait espéré d'un circuit à anticipation de retenue. Ce résultat est bien évidemment explicable par le fait que la chaine de retenue gravée en dur offre de meilleures performances que ce que l'on peut espérer de l'implémentation d'une fonction multivariée ($\gg 6$ variables) au moyen de LUT. Finalement, la dernière option ("CLA CC") offre l'accélération recherchée (le délai est réduit de 30% à 60%), au prix d'une occupation de surface supplémentaire de 15% à 25%. On notera que l'amélioration est meilleure pour $k = 16$, ce qui s'explique par le fait que $\tau_{\text{lut}} \gg \tau_{\text{cc}}$.

4.4.4 Implémentation HRCS des opérateurs de type TAA

Afin d'évaluer l'utilité du format HRCS dans l'implémentation d'opérateurs à virgule flottante exploitant l'algorithme d'auto-alignement que nous avons proposé, nous avons réalisé plusieurs de ces opérateurs et évalué leur performances. Nous avons également varié les différents paramètres entrant en jeu dans la constitution de ces opérateurs afin d'en mesurer concrètement l'impact. Les opérateurs adoptent différentes topologies dont l'opération de base est l'addition, de sorte que l'implémentation matérielle de l'Algorithme 1 est adoptée sous différentes formes.

Les Figures 4.12.a à 4.12.c présentent les trois opérateurs considérés ici, respectivement l'additionneur, l'accumulateur (de K-tuple exactitude) et un additionneur terminé par un accumulateur K-tuple, c.-à d. la forme la plus simple de l'arbre d'additionneurs terminé par un accumulateur. Les Figures 4.12.d à 4.12.f présentent quant à elles l'architecture interne de l'implémentation de type TAA de ces opérateurs, et les différents formats internes utilisés. On distinguera les chemins de données où les opérandes sont en format à virgule flottante standard, en format FAA avec mantisse conventionnelle (non-redondante) et en format FAA avec mantisse redondante (HRCS), tel qu'indiqué par la légende de la Figure 4.12. Ces formats déterminent le type d'additionneur utilisé, soit un additionneur hétérogène (indiqué par une zone grisée) ou un additionneur endomorphique (indiqué par une zone grisée et hachurée).

Avant de discuter les résultats d'implémentation, il convient d'indiquer que ceux-ci sont obtenus après placement et routage et sans contrainte de placement, en utilisant la version

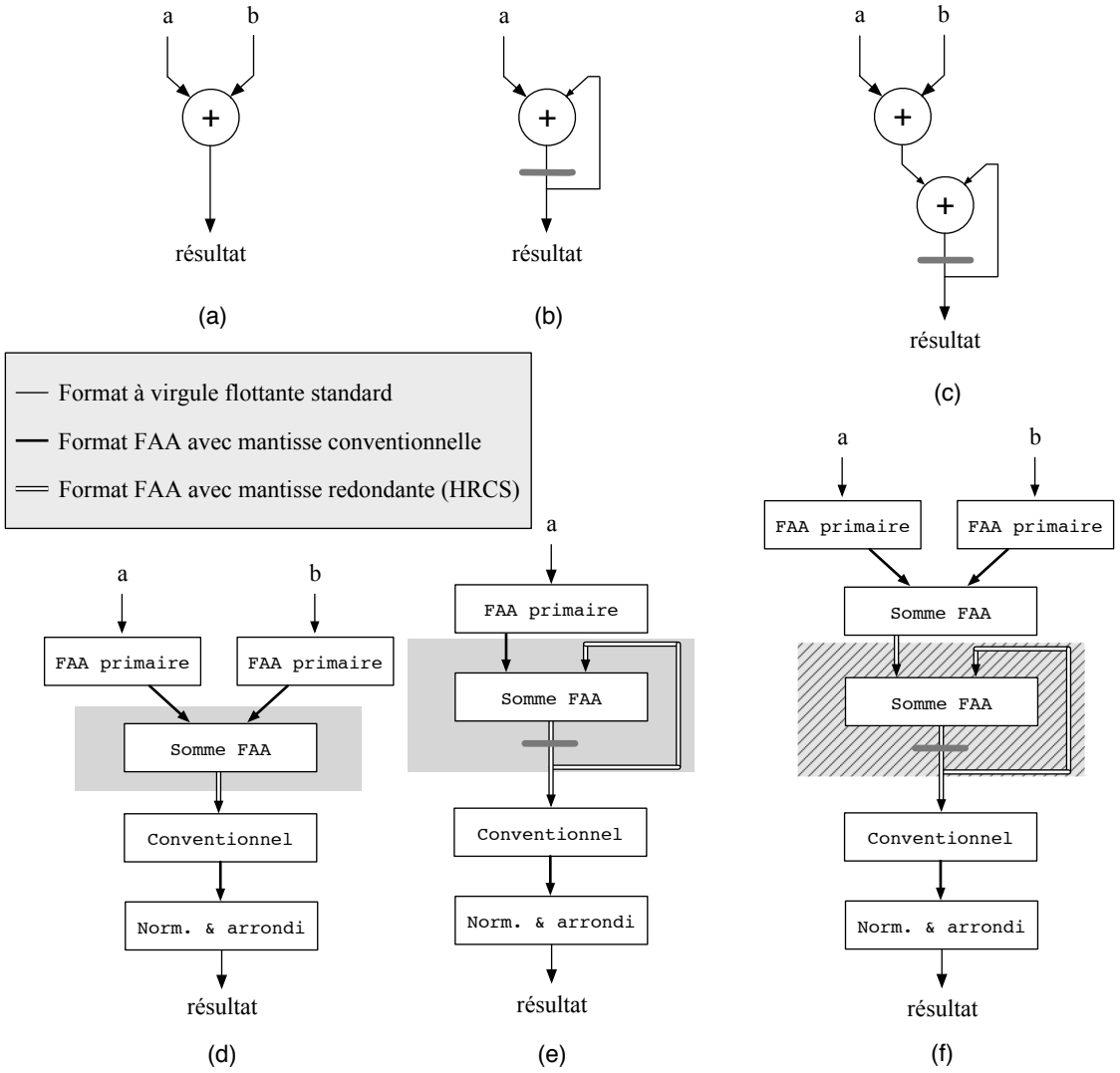

Figure 4.12 – Operateurs à virgule flottante considérés : (a) Additionneur binaire ; (b) Accumulateur ; (c) Additionneur binaire terminé par un accumulateur ; (d) Implémentation de (a) de type TAA ; (e) Implémentation de (b) de type TAA ; (f) Implémentation de (c) de type TAA.

13.1 de l'outil ISE de Xilinx, avec le FPGA Virtex 5 XC5VSX50T-3C pour cible. Ces résultats peuvent varier sensiblement, en mieux ou en pire qui plus est, simplement en changeant le FPGA cible ou en incluant des contraintes de connexion aux pattes. Par conséquent, il faudra manipuler les chiffres présentés avec une certaine précaution, et les utiliser à titre indicatif et comme mesures relatives.

La Table 4.1 présente les différents paramètres et caractéristiques des opérateurs utilisés. Un opérateur utilisant l'approche TAA/HRCS pour effectuer des additions doit fixer l'ensemble des paramètres (w, k, g, N, l), où :

- w est la largeur de la mantisse (il est fonction des autres paramètres) ;
- k est le paramètre du format HRCS de la mantisse ;
- g est le nombre de bits de garde ;
- N est le maximum d'opérandes pouvant être traités dans une même sommation ;
- l détermine le nombre de décalages à fine granularité dont résulte la mantisse FAA.

Il importe de mentionner que le paramètre l a un impact direct sur la quantités de décalages potentiels que la mantisse étendue du format FAA peut subir. Plus précisément, il existe $\lceil w/2^l \rceil$ décalages différents pour une mantisse de largeur w dans un format FAA de paramètre l. En même temps, l limite les valeurs admissibles pour k puisque 2^l doit être un multiple de k afin que les décalages de la mantisse redondante soient effectuées sur des multiples de la base 2^k.

Afin de bien comprendre comment un concepteur doit manipuler ces paramètres, considérons ce qui suit. Supposons que nous voulions traiter des opérandes en simple précision, dont la mantisse signée comporte $w_{std} = 25$ bits. Supposons que nous choisissions $g = 16$ et $N = 2^{16}$. Fixer l à 3 mène à une largeur de mantisse étendue de $w = w_{std} + g + \lceil \log_2(N) \rceil + (2^l - 1) = 25 + 16 + 16 + 7 = 64$, ce qui implique également $\lceil w/2^l \rceil = 8$ décalages potentiels à large granularité. Si on fixait l plutôt à 4, la largeur de la mantisse étendue serait de $w = 25 + 16 + 16 + 15 = 72$, et elle impliquerait $\lceil w/2^l \rceil = 5$ décalages à large granularité potentiels. Bien entendu, plus le nombre de décalages potentiels est grand, plus complexe est l'implémentation d'un accumulateur à un cycle. La relation est inverse suivant la largeur de la mantisse.

Dans le but de disposer d'une référence quant à l'occupation de surface et la fréquence d'opération d'un additionneur, nous avons généré au moyen de l'outil Coregen de Xilinx (version 13.1) deux additionneurs binaires pour chacune des deux précisions usuelles de la virgule flottante. Ces opérateurs sont identifiés comme items #1 et #2 dans les Tableaux 4.1 et 4.2. Les autres opérateurs énumérés dans ces tableaux ont été réalisés par nous en exploitant au mieux le format FAA/HRCS. Il a ainsi été trouvé que choisir $k = 8$ était préférable pour les opérateurs #3 et #4 (additionneurs binaires au format FAA), tandis que $k = 16$ était préférable pour les items #5 à #12.

Le Tableau 4.2 détaille l'occupation de surface et la fréquence d'opération maximale de chacun des items précités lorsque le FPGA cible est Virtex 5 XC5VSX50T-3C [119]. Les opérateurs #1 et #2, sont optimaux pour des latences de 9 et 12 respectivement : ils présentent une fréquence d'opération maximale supérieures à 400 MHz pour une occupation de surface

Tableau 4.1 – Paramètres et caractéristiques des opérateurs évalués.

Item	Architecture	Précision	K	w	k	l	g	N	Latence
Coregen									
#1	Add.	simple	1.00	-	-	-	-	2	9
#2	Add.	double	1.00	-	-	-	-	2	12
Opérateurs TAA									
#3	Add.	simple	1.00	35	8	3	2	2	9
#4	Add.	double	1.00	72	8	4	2	2	11
#5	Acc. K-tuple	simple	1.67	64	16	4	16	2^8	9
#6	Acc. K-tuple	double	1.60	128	16	5	32	2^{11}	11
#7	Add.+acc.	simple	1.67	64	16	4	16	2^8	11
#8	Add.+acc.	double	1.60	128	16	5	32	2^{11}	12
#9	Acc. K-tuple	simple	1.67	96	16	4	16	2^{40}	10
#10	Acc. K-tuple	double	1.60	160	16	5	32	2^{43}	11
#11	Add.+acc.	simple	1.67	96	16	4	16	2^{40}	12
#12	Add.+acc.	double	1.60	160	16	5	32	2^{43}	12

minimale. De telles fréquences d'opération sont également possibles lorsque l'additionneur binaire est réalisé en format FAA (items #3 et #4). Les fréquences d'opération des items #5 à #12 sont légèrement plus faibles car ces opérateurs comportent un accumulateur à un cycle. Néanmoins, on appréciera que ces opérateurs sont capables de traiter les données à des fréquences allant de 324 MHz à 338 MHz, et ce même si la largeur des mantisses est de 160 bits (ce qui est le cas des items #8 et #10). Cette performance est entièrement attribuable au format HRCS ainsi qu'à l'efficacité et la simplicité de l'Algorithme 1.

Comme nous avons pu le constater au Chapitre 3, les opérateurs de type TAA garantissent une meilleure exactitude des calculs au prix de mantisses internes plus larges. Lorsque les mantisses larges sont normalisées lors de la conversion de FAA à standard, il convient d'effectuer une détection du bit le plus fort, opération qui est d'usage coûteuse en occupation de surface et qui peut causer un surcoût matériel. Cependant, ce surcoût n'est pas visible lorsque l'on compare les items #3 et #4 aux références #1 et #2 (il est même contredit dans le cas de l'opérateur #4) car les mantisses étendues dans ces cas n'ont pas à être trop larges. L'argument prend son importance par contre dès lors qu'il s'agit d'additionner plus de deux nombres, tel que discuté au Chapitre 3.

Ainsi, si l'on compare les opérateurs #5 à #6 (accumulateurs) aux références #1 et #2 (additionneurs simples), on constatera qu'ils occupent respectivement 35.3% et 29.4% plus d'espace que les références. Néanmoins, ce surcoût est fort raisonnable si on considère le fait qu'un accumulateur nécessite plus de logique qu'un simple additionneur [111]. De plus, ce surplus est largement compensé par le fait, d'une part, que la boucle d'accumulation est

Tableau 4.2 – Estimés d'occupation de surface et de fréquence maximale des opérateurs à virgule flottante exploitant le format HRCS sur le Virtex 5 (XC5VSX50T-3C) de Xilinx. Les résultats sont obtenus après placement et routage.

Item	Arch.	Précision	K	Slices	Reg.	LUT	Délai (ns)	Freq. (MHz)
Coregen								
#1	Add.	simple	1.00	153	445	415	2.225	449.44
#2	Add.	double	1.00	425	1,035	718	2.306	433.65
Opérateurs TAA								
#3	Add.	simple	1.00	188	448	587	2.177	459.35
#4	Add.	double	1.00	401	1,026	1,253	2.266	441.31
#5	Acc. K-tuple	simple	1.67	207	397	634	2.933	340.95
#6	Acc. K-tuple	double	1.60	550	1,120	1,630	2.973	336.36
#7	Add.+acc.	simple	1.67	310	661	887	2.954	338.52
#8	Add.+acc.	double	1.60	638	1,367	1,959	3.066	326.12
#9	Acc. K-tuple	simple	1.67	317	652	906	2.954	338.52
#10	Acc. K-tuple	double	1.60	657	1,271	1,985	3.057	327.12
#11	Add.+acc.	simple	1.67	388	854	1,215	2.969	336.81
#12	Add.+acc.	double	1.60	799	1,642	2,468	3.079	324.78

réalisée en un cycle (offrant une marge de manœuvre supérieure que ce qui serait possible avec un simple additionneur), et que l'exactitude est K-tuple de la précision des opérandes, avec dans ces cas-ci $K \geq 1.6$.

En ce qui a trait aux opérateurs #7 et #8 (arbres d'additionneurs minimaux terminés par un accumulateur), il faut considérer que ces opérateurs ont la même puissance de calcul que deux additionneurs. Par conséquent, le fait que ces opérateurs occupent respectivement 182% et 150% de la surface des références Coregen #1 et #2 au lieu des $> 200\%$ permissibles démontre que ces opérateurs offrent un gain en surface plutôt qu'un surcoût. Ce fait est d'autant plus marquant que, là aussi, nous bénéficions d'une exactitude des résultats qui est K-tuple de la précision des opérandes, avec $K \geq 1.6$. Ces observations plaident en faveur de l'utilisation du patron de conception à "chemin de données fusionné" (*fused-path*).

Finalement, afin de mesurer l'impact de dédier $\lceil \log_2(N) \rceil$ bits à la gauche de la mantisse pour sommer N opérandes, nous avons ré-implémenté les opérateurs #5 à #8 avec des valeurs plus importantes pour N. Ainsi, tandis que N valait 256 pour les opérateurs en simple précision (#5 et #7) et 2048 pour les opérateurs en double précision, nous avons fixé N à 2^{40} pour les opérateurs en simple précision (#8 et #10) et 2^{43} pour les opérateurs en double précision : additionner $N = 2^{40}$ opérandes à 300 MHz correspond à une heure d'accumulation, à 8 heures pour $N = 2^{43}$.

Là encore, les résultats du Tableau 4.2 sont éloquents et démontrent que le surcoût

nécessaire pour accommoder les 32 bits supplémentaires est très acceptable. En effet, il est de 53% (item #5→#9) et de 39.1% (item #7→#11) en simple précision ; il est de 19.4% (item #6→#10) et 25.2% (item #8→#12) en double précision. Ces observations sont cohérentes avec les résultats précédents et indiquent qu'il est plus rentable d'employer des mantisses larges en double précision qu'en simple précision. Il y a fort à parier que cela est dû au fait que les opérateurs de Xilinx sont plus optimisés en simple précision qu'en double précision.

Une analyse temporelle sur certains opérateurs a été menée afin de départir la part du délai qui se situe dans la logique de celle qui se trouve dans les ressources de routage. Il apparaît que le routage comprend entre 50% et 80% du délai total. Les parties les plus critiques (ayant un taux élevé de routage) se situent dans le bloc de conversion FAA à VF.

Cette analyse indique que les résultats d'implémentation de nos opérateurs, bien que fort encourageants, pourraient bénéficier d'une optimisation supplémentaire pour les améliorer davantage, par exemple par un placement manuel. Il convient cependant de relever qu'une telle tâche peut s'avérer ardue. Les résultats actuels étant très convenables pour le contexte d'application de la présente thèse, ainsi que pour la démonstration que nous désirions faire dans ce chapitre, nous n'avons pas procédé à de telles optimisations.

4.5 Conclusion

Ce chapitre a permis de couvrir le format redondant HRCS pour lequel nous avons développé de nouveaux opérateurs matériels dont la performance a été démontrée par nos évaluations empiriques, et dont l'utilité a été rendue claire par les résultats de la Section 4.4.4. Dans le chapitre suivant, nous allons explorer l'intégration des résultats des chapitres précédents dans la réalisation d'engins de calcul à haute performance pour la simulation en temps-réel aux circuits d'électronique de puissance.

CHAPITRE 5

ENGINS DE CALCUL EMPLOYANT L'APPROCHE TAA/HRCS

5.1 Introduction

Ce dernier chapitre porte sur la réalisation d'engins de calcul employant la technique d'auto-alignement pour la simulation de circuits d'électronique de puissance. Pour ce faire, nous débutons notre propos en considérant la question de la disponibilité de la mémoire embarquée. Ce travail nous portera à revenir sur la modélisation des interrupteurs à matrice fixe. Ainsi, nous présenterons les manipulations des équations MANA que nous proposons pour permettre l'exécution efficace de l'algorithme sur nos engins de calcul. Nous procéderons ensuite à l'évaluation de certains paramètres d'implémentation des opérateurs arithmétiques tels que la précision du format des nombres et la consommation des blocs DSP. Cette discussion se terminera par la présentation d'un engin de calcul versatile permettant de simuler des convertisseurs de puissance de topologie arbitraire et disposant d'un maximum de 24 interrupteurs. Le chapitre se conclut avec la présentation d'études de cas où le modèle résistif de l'interrupteur (R_{on}/R_{off}) a été préféré au modèle à matrice constante. Ces études de cas illustrent comment les techniques de conception d'opérateurs en VF que nous avons développées pour la conception sur FPGA ont permis d'atteindre des performances estimables dans la simulation en temps réel de convertisseurs de puissance tirés de contextes industriels réalistes. Ainsi, le chapitre recoupe les résultats applicatifs rattachés à diverses de nos publications [27, 49, 84, 85, 87, 88, 105].

5.2 Usage de la mémoire embarquée

Les ressources en mémoire embarquée sur un FPGA sont de nos jours relativement appréciables. Néanmoins, pour de nombreux problèmes de simulation en temps réel, elles demeurent insuffisantes. Considérons par exemple le cas de circuits d'électronique de puissance pour lesquels nous voudrions conserver les équations du réseau pour toutes les combinaisons possibles des statuts des N interrupteurs qui le composent. La quantité de mémoire nécessaire serait alors de l'ordre de 2^{N} fois celle nécessaire pour représenter les équations du réseau linéaire équivalent (N = 0). Cette croissance exponentielle rencontre très rapidement les limites physiques des FPGA, de sorte que d'autres techniques de modélisation des réseaux commutés doivent être considérées.

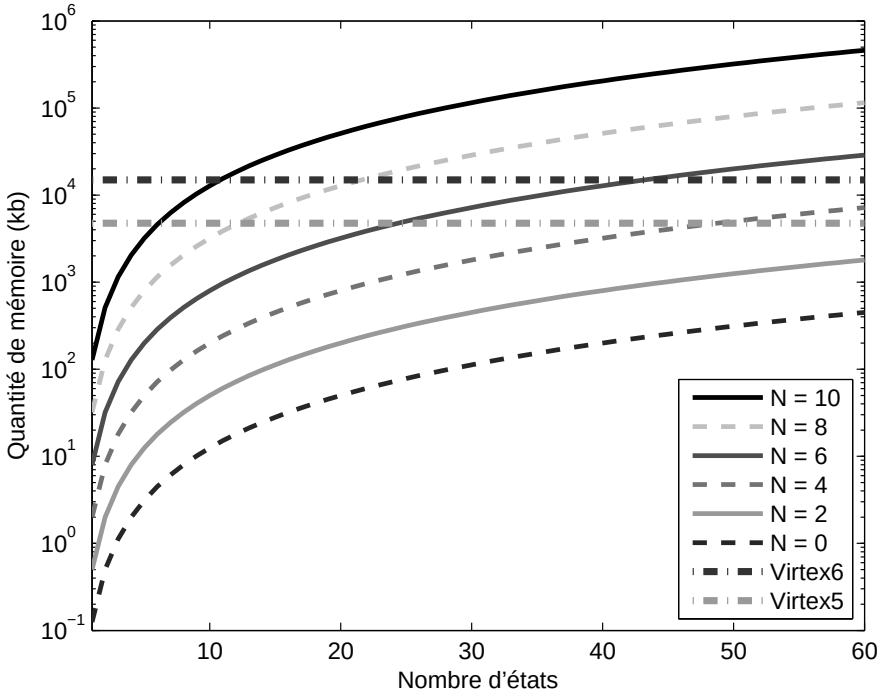

Figure 5.1 – Utilisation de la mémoire embarquée pour différentes tailles du réseau, fonction du nombre d'états et du nombre d'interrupteurs dans le circuit (N donne le nombre d'interrupteurs dans le réseau).

Afin de donner une idée suffisamment précise des limitations dont nous parlons, la Figure 5.1 propose de rapporter sur un graphique la quantité — en kilo-bits (kb) [1] — de mémoire utilisée en fonction du nombre d'états (on posera que $|\mathbf{x}_n| = |\mathbf{u}_n| = |\mathbf{y}_n|$) et le nombre d'interrupteurs dans le réseau (N). Sont également rapportées sur le graphique la quantité de mémoire disponible sur le Virtex 5 utilisé comme FPGA cible dans la présentation des résultats tout au long de ce document (on parle ici du XC5SX50T [119]), ainsi que la quantité de mémoire disponible sur le Virtex 6, le FPGA que fournit actuellement le partenaire industriel sur ses nouveaux simulateurs (il s'agit cette fois du XC6LX240T-1C [122]).

On constate dans la Figure 5.1 que le nombre d'interrupteurs pouvant être considérés sature assez rapidement à 6 à 8 au maximum, et ce pour un nombre d'états dépassant difficilement les 20. Dans un certain nombre de convertisseurs de puissance (le convertisseur triphasé à deux niveaux par exemple), les besoins sont bien en deçà de telles grandeurs. Néanmoins, l'ambition de ce travail est d'offrir une solution sur FPGA qui puisse satisfaire les besoins de nombreuses topologies de convertisseurs, et une grande variété de quanti-

1. On rappellera que 1 kb = 1024 bits.

tés d'interrupteurs. Cette ambition est notamment motivée par les besoins industriels de notre partenaire commercial. À l'issue de ce développement, on comprendra d'autant plus facilement l'intérêt du modèle à matrice fixe présenté à la Section 5.3.

5.3 Modèle d'interrupteur à matrice fixe

Comme nous avons pu le voir au Chapitre 1, plusieurs techniques de modélisation des interrupteurs existent. Étant données les contraintes de disponibilité de la mémoire embarquée, la technique qui a retenu notre attention pour le travail de thèse est celle à matrice fixe, proposée au début des années 1990 [45, 47, 94]. Ce modèle procède en deux temps. D'une part, il choisit de modéliser l'interrupteur soit comme une petite capacité (pour indiquer que l'interrupteur est ouvert), soit comme une petite inductance (pour indiquer que l'interrupteur est fermé), mais en assignant des valeurs C_{sw} et L_{sw} telles que les équations du réseau deviennent invariantes et que seule la façon de calculer l'historique associé à l'interrupteur dépende du statut de ce dernier. D'autre part, ce modèle d'interrupteur est accompagné d'une règle de mise à jour du statut de l'interrupteur pour différents type de semi-conducteurs [94] exprimée de manière relativement simple.

5.3.1 Formulation mathématique du modèle

Comme nous l'avons évoqué au Chapitre 2, les méthodes nodales, et particulièrement l'approche MANA, procèdent à la discrétisation de chacun des composants du circuit au moyen des méthodes numériques connues, puis assemblent les équations du circuit compagnon. Pour les circuits d'électronique de puissance, la méthode d'Euler arrière (MIE), méthode implicite d'ordre 1, est préférable car elle n'occasionne pas les oscillations que la méthode du trapèze tend à générer. Chaque composant aux bornes des nœdus k et m se voit alors associer une conductance équivalente (g_{eq}) et un courant historique (j_n), de sorte que

$$g_{\text{eq}}(v_n^k - v_n^m) = j_n + i_n^{km} \tag{5.1}$$

où i_{n+1}^{km} est le courant qui traverse le composant en passant du nœud k au nœud m, et $(v_n^k - v_n^m)$ est la chute de tension à ses bornes. La discrétisation MIE associe une conductance équivalente g_C à une capacité C exprimée par $g_C{=}C/h$, tandis qu'elle associe une conductance équivalente g_L à une inductance L exprimée par $g_L{=}h/L$, où $h{=}\Delta t$ est le pas fixe de la simulation. Le terme historique de la capacité pour le prochain pas de la simulation est donné par :

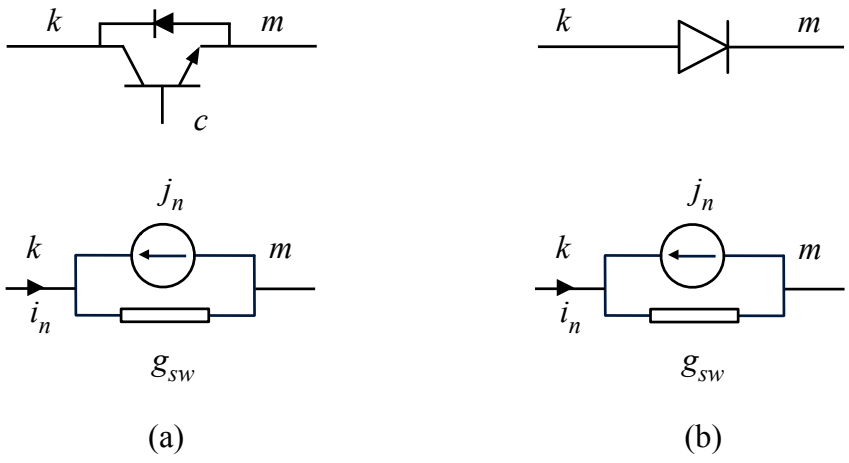

Figure 5.2 – Exemples de discrétisation d'interrupteurs suivant le modèle à matrice constante : (a) IGBT avec diode anti-parallèle ; (b) diode.

$$j_{n+1} = g_C(v_n^k - v_n^m) \tag{5.2}$$

Celui de l'inductance est donné par :

$$j_{n+1} = -i_n^{km} = j_n - g_L(v_n^k - v_n^m) \tag{5.3}$$

Le modèle à matrice fixe procède en représentant l'interrupteur ouvert par une petite capacité tandis que l'interrupteur fermé est une petite inductance. En posant $g_{C_{sw}} = g_{L_{sw}} = g_{sw}$, la matrice MANA $\mathbf{A}_t$ de l'équation 2.14 devient invariante et ce, quel que soit le statut des interrupteurs, de sorte que seul le vecteur $\mathbf{b}_t$ en est affecté en employant, selon le cas, l'équation 5.2 ou l'équation 5.3.

Une des difficulté dans la simulation des réseaux à interrupteurs provient dans l'actualisation des statuts des interrupteurs, particulièrement en présence d'interrupteurs à commutation naturelle (par opposition aux interrupteurs à commutation commandée). La Figure 5.2 présente deux interrupteurs et leur modèle équivalent, à savoir un transistor bipolaire à grille isolée (IGBT) avec diode anti-parallèle (où les deux semi-conducteurs sont vus comme un seul interrupteur) et la diode. On retrouve ces deux types de semi-conducteurs dans la majorité des convertisseurs de puissance modernes. D'autres types d'interrupteurs sont considérés dans [94] et traités de manière analogue.

La loi d'actualisation du statut de la paire IGBT/diode est donnée par :

$$s_n = c_n + s_{n-1}(i_n \leq 0) + \overline{s_{n-1}}(v_n < 0) \tag{5.4}$$

où c_n est la gâchette commandant l'IGBT au nouveau pas, s_n, v_n et i_n sont respectivement le statut de l'interrupteur, le voltage à ses bornes et le courant qui le traverse.

La loi d'actualisation du statut de la diode, elle, se traduit par :

$$s_{n+1} = s_{n-1}(i_n \geq 0) + \overline{s_{n-1}}(v_n \geq 0) \tag{5.5}$$

5.3.2 Manipulation des équations MANA

Nous avons vu au Chapitre 2 que les équations MANA doivent être manipulées afin d'obtenir une formulation sous forme d'un produit matrice-vecteur. Dans ce qui suit, nous allons voir comment il est possible de parvenir à une forme similaire à celle de l'équation 2.17 malgré les quelques spécificités de la méthode considérée ici. Nous verrons également comment il est possible de réduire la quantité de variables à évaluer pour déterminer le statut des interrupteurs.

À chaque pas de calcul, l'engin de calcul aura pour mission d'anticiper le vecteur des historiques $\mathbf{j}_{n+1} = [\mathbf{j}^{s_0}_{n+1}, \mathbf{j}^{s_1}_{n+1}, \mathbf{j}^{d}_{n+1}]^T$ qui contient l'historique des tous les interrupteurs à l'état ouvert ($\mathbf{j}^{s_0}_{n+1}$), l'historique de tous les interrupteurs à l'état fermé ($\mathbf{j}^{s_1}_{n+1}$) ainsi que l'historique des éléments dynamiques (L/C), que l'on note $\mathbf{j}^{d}_{n+1}$. Pour le pas courant, on définira le vecteur des historiques courant $\mathbf{j}^{\sigma}_{n}$ qui est cohérent avec la combinaison des statuts des interrupteurs (notée σ). Le vecteur $\mathbf{j}^{\sigma}_{n}$ contient le vecteur des historiques $\mathbf{j}^{d}_{n}$ ainsi que des éléments des vecteurs $\mathbf{j}^{s_0}_{n}$ et $\mathbf{j}^{s_1}_{n}$, selon le statut effectif de chaque interrupteur. Ainsi, on réécrira l'équation 2.17 sous sa nouvelle forme :

$$\begin{bmatrix} \mathbf{j}_{n+1} \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{W_{ju}} & \mathbf{W_{jj}} \\ \mathbf{W_{yu}} & \mathbf{W_{yj}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_n \\ \mathbf{j}^{\sigma}_{n} \end{bmatrix} \tag{5.6}$$

À chaque pas de temps, l'engin de calcul part du vecteur $\mathbf{j}_n$, détermine l'état des interrupteurs suivant les équations 5.4 et 5.5 et forme le vecteur $\mathbf{j}^{\sigma}_{n}$ pour résoudre l'équation 5.6. Il reste alors à observer que les équations 5.4 et 5.5 dépendent du signe des courant/tensions de chaque interrupteur. Or, puisque l'interrupteur est modélisé comme une capacité ou une inductance (selon qu'il est ouvert ou fermé), il est possible de connaître le signe de la tension à ses bornes et celui du courant qui le traverse à partir du signe des historiques $j^{s_0}_{n}$ et $j^{s_1}_{n}$, respectivement, comme l'indique les équations 5.2 et 5.3. Par conséquent, plutôt que de calculer quatre variables pour chaque interrupteur dans le réseau (soit $j^{s_0}_{n}$, $j^{s_1}_{n}$, v_s et i_s), seuls les historiques à l'état ouvert et fermé suffisent. On remarquera finalement que d'après les équations 5.2 et 5.3, il est possible de poser $j^{s_1}_{n+1} = j^{s}_{n} - j^{s_0}_{n+1}$. Ainsi, il nous suffira de calculer $j^{s_0}_{n+1}$ pour déduire $j^{s_1}_{n+1}$ par simple soustraction. Ceci est très avantageux d'autant plus que

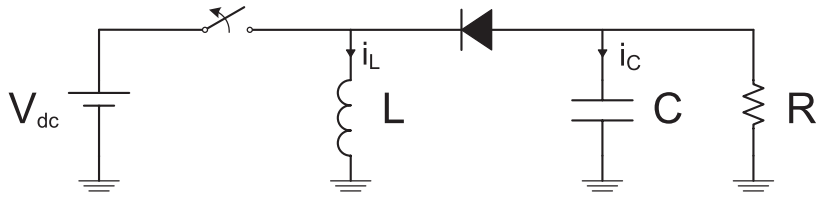

Figure 5.3 – Convertisseur buck-boost servant à démontrer les limites du modèle de l'interrupteur à matrice fixe.

Paramètres	Mode continue		Mode discontinu	
R	8	Ω	80	Ω
L	0.01	mH	0.01	mH
C	100	μF	100	μF
V_{dc}	8	V	8	V
v_C initial	−24	V	−37.5	V
i_L initial	+9	A	0	A
Fréquence de la MLI	100	kHz	100	kHz
Rapport cyclique	75	%	75	%

Tableau 5.1 – Paramètres du buck-boost.

les additions/soustraction en virgule flottante peuvent être effectuées en un cycle d'horloge au moyen du FAA que nous avons couvert au Chapitre 3. Pour ce faire, nous ajouterons des éléments de mémoire à l'engin de calcul de la Figure 2.2 pour conserver le vecteur $\mathbf{j}_n^\sigma$ au format FAA premier.

5.3.3 Limites du modèle à matrice fixe

Il existe certaines limitations au modèle d'interrupteur à matrice fixe qu'il convient de connaître. Pour ce faire, nous proposons de considérer le circuit de la Figure 5.3. Ce circuit d'exemple est emprunté à [106]. Il présente différentes caractéristiques et modes d'opération (mode continu, mode discontinu) qui en font un bon cas d'étude. Le Tableau 5.1 détaille les paramètres utilisés pour initier les deux modes de fonctionnement.

La Figure 5.4 présente les signaux extraits d'une simulation du convertisseur (code Matlab) en mode continu sur une durée 0.2 ms. Une référence SPS est utilisée pour valider les résultats. On fera remarquer que le pas de calcul utilisé pour la référence SPS et le modèle est de 1 ns. Un tel pas de calcul est nécessaire à cause de la haute fréquence de commutation du modèle d'exemple, mais il s'avère également nécessaire pour que la référence SPS donne des résultats corrects. De plus, un pas de calcul de 1 ns facilite le choix du g_{sw} pour le modèle d'interrupteur à matrice fixe : on pose $g_{sw} = 1$, de sorte que $L_{sw} = 1$ nH et $C_{sw} = 1$ nF.

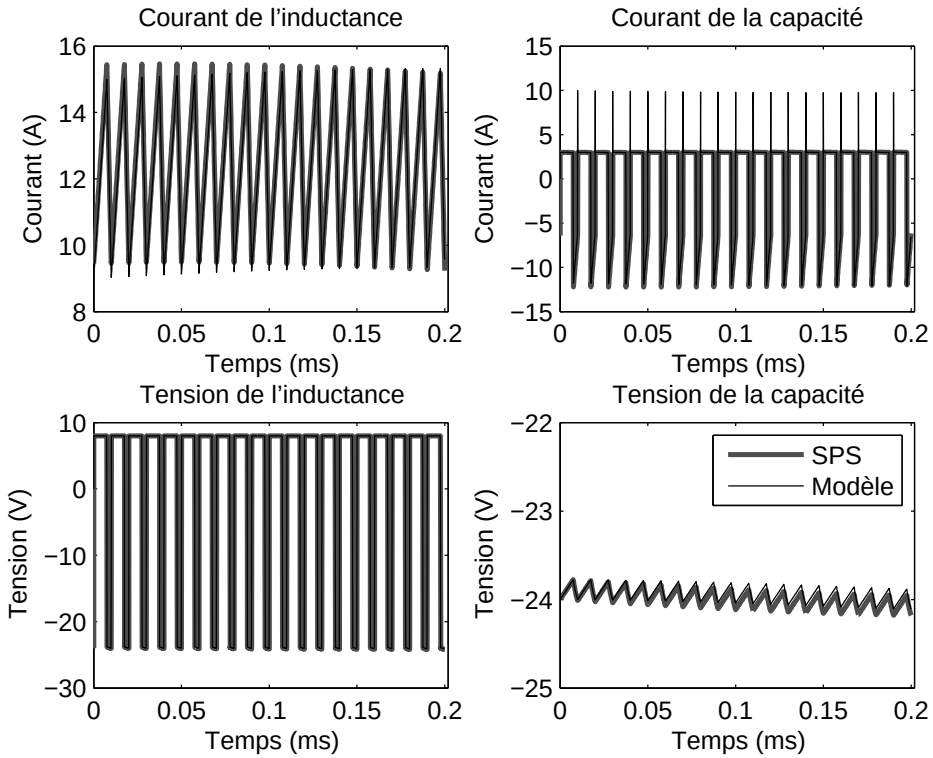

Figure 5.4 – Résultats de simulation du convertisseur buck-boost en mode continu.

Les résultats de la simulation sont plutôt précis : des différences apparaissent au fil de la simulation sur la tension v_c et sur le courant i_L, mais ces différences sont relativement négligeables (< 1 %). Cependant, on voit apparaître des impulsions sur le courant i_C qui sont absentes de la référence. De telles impulsions sont générées par le modèle au moment des commutations commandées. Il a été démontré dans [94] que ces impulsions sont en fait des oscillations convergentes du courant (ou, selon le circuit, de la tension) autour de sa valeur finale après la commutation. Plus ces oscillations sont courtes, plus leur amplitude maximale est importante, et vice et versa. La durée et l'amplitude de ces impulsions dépendent de la valeur de g_{sw} et du pas de calcul. Par conséquent, si le pas de calcul de la simulation est trop grand, il peut être difficile de trouver une valeur de g_{sw} qui assure des oscillations faibles et une convergence rapide, surtout lorsque la fréquence de commutation tend à augmenter.

La Figure 5.5 présente les signaux extraits d'une simulation du convertisseur en mode discontinu sur une durée de 0.05 ms. Ici aussi la référence SPS sert à valider les résultats. Les pas de calcul du modèle et de la référence sont de 1 ns. On constatera que les résultats de la simulation pour le mode discontinu sont tout aussi précis. Mais en plus des oscillations observées sur i_C, des perturbations apparaissent sur i_L et v_L. Contrairement aux oscillations

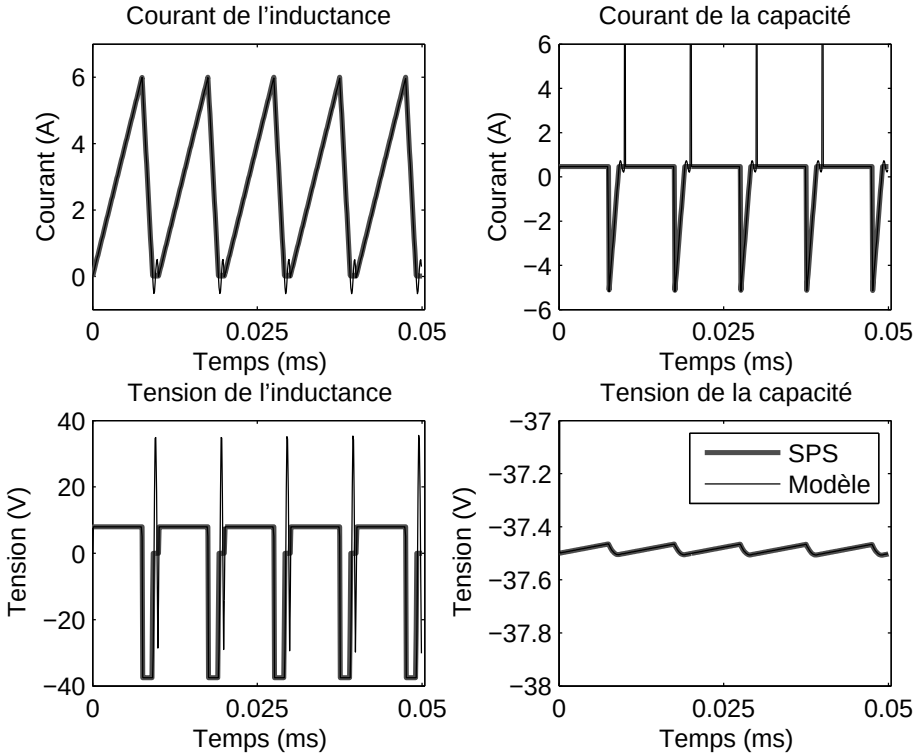

Figure 5.5 – Résultats de simulation du convertisseur buck-boost en mode discontinu.

précédentes, les oscillations observées dans le mode discontinu (qui apparaissent durant le temps où les deux interrupteurs sont ouverts) ne convergent pas. Elles s'expliquent par le fait que lorsque les deux interrupteurs ouverts sont remplacés par de petites capacités C_{sw}. Ainsi, durant ce court temps, le circuit se comporte comme un circuit LC avec un petit courant initial (i_L) qui démarre une oscillation de fréquence $1/(2\pi\sqrt{2\mathrm{C}_{sw}\mathrm{L}})$.

5.3.4 Méthode pour surmonter les limites du modèle à matrice fixe

Il existe différentes méthodes pour réduire l'impact des oscillations observées durant la convergence du modèle à matrice fixe. La plus simple à implémenter est celle proposée dans [47] et qui consiste à ajouter en parallèle de l'interrupteur un amortisseur numérique (*snubber*). Une approche similaire a été proposée dans [70] ; elle consiste à modifier le modèle de l'interrupteur pour qu'il utilise une petite inductance pour modéliser l'interrupteur à l'état fermé, et un circuit RC série pour modéliser un circuit ouvert, de sorte que $g_{\mathrm{L}_{sw}} = g_{\mathrm{RC}_{sw}}$. Cette seconde approche ne permet cependant pas la réduction du nombre d'équations à utiliser que nous avons proposée (les historiques au lieu des historiques plus v_s et i_s). À

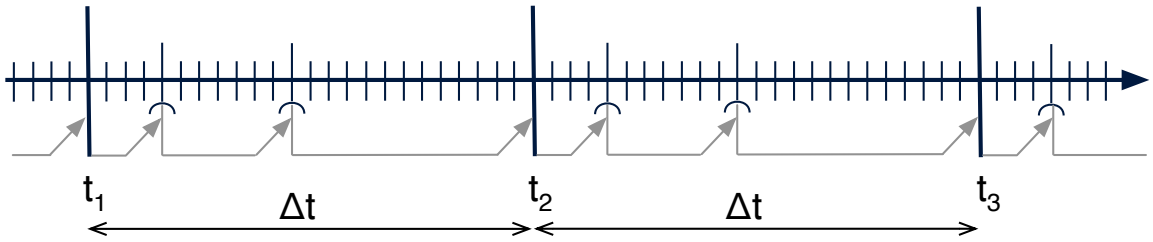

Figure 5.6 – Illustration du principe de la simulation par multiples sous-pas.

ce titre, la première méthode présente davantage d'intérêts, d'autant plus que l'amortisseur numérique peut offrir des résultats satisfaisant tout en étant entièrement résistif (ce qui permet de faire l'économie du calcul d'un historique de capacité pour le *snubber*).

Dans ce qui suit, nous proposons une nouvelle méthode pour surmonter les limites du modèle à matrice fixe. L'approche proposée permet d'absorber les oscillations convergentes et les oscillations parasites observées durant le mode discontinu. Cependant, la méthode souffre de certaines limitations que nous aurons soin de préciser. Ces limitations nous ont poussé à ne pas publier cette méthode. Néanmoins, elle présente un certain intérêt qui lui mérite d'être consignée dans la présente thèse.

La méthode que nous proposons part du constat que le modèle à matrice fixe est plus précis lorsque le pas de calcul est très petit (≤ 1 ns), d'autant plus si un amortisseur résistif est employé. Cependant, avec la technologie FPGA actuelle, il est difficile de concevoir un pas de calcul en bas des 100 ns. Or il convient de constater qu'à l'intérieur d'une fenêtre de temps $\Delta t = 100$ ns, et pour une fréquence de commutation de quelques dizaines de kilohertz, les interrupteurs changent peu de fois de statut. Ainsi, comme l'illustre la Figure 5.6, nous proposons de subdiviser le pas de calcul Δt en petits sous-pas de durée fixe δt ($\Delta t = 24\delta t$ dans l'exemple). Dès lors, en supposant que les interrupteurs ne changent pas de statut durant k sous-pas consécutifs, il est possible de se déplacer du temps t au pas $t + k\delta t$ en exploitant certains manipulations algébriques que nous présentons ci-après. Il est suggéré alors de procéder à un nombre de ces actualisations par pas de calcul (l'exemple de la Figure 5.6 en compte trois, avec $k_1 = 4$, $k_2 = 7$ et $k_3 = 13$).

Si les interrupteurs demeurent dans une combinaison σ donnée durant un sous-pas de temps δt, on peut poser :

$$\begin{bmatrix} \mathbf{j}_{n+1}^{\sigma} \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{\mathbf{ju}}^{\sigma} & \mathbf{W}_{\mathbf{jj}}^{\sigma} \\ \mathbf{W}_{\mathbf{yu}} & \mathbf{W}_{\mathbf{yj}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_n \\ \mathbf{j}_n^{\sigma} \end{bmatrix} \tag{5.7}$$

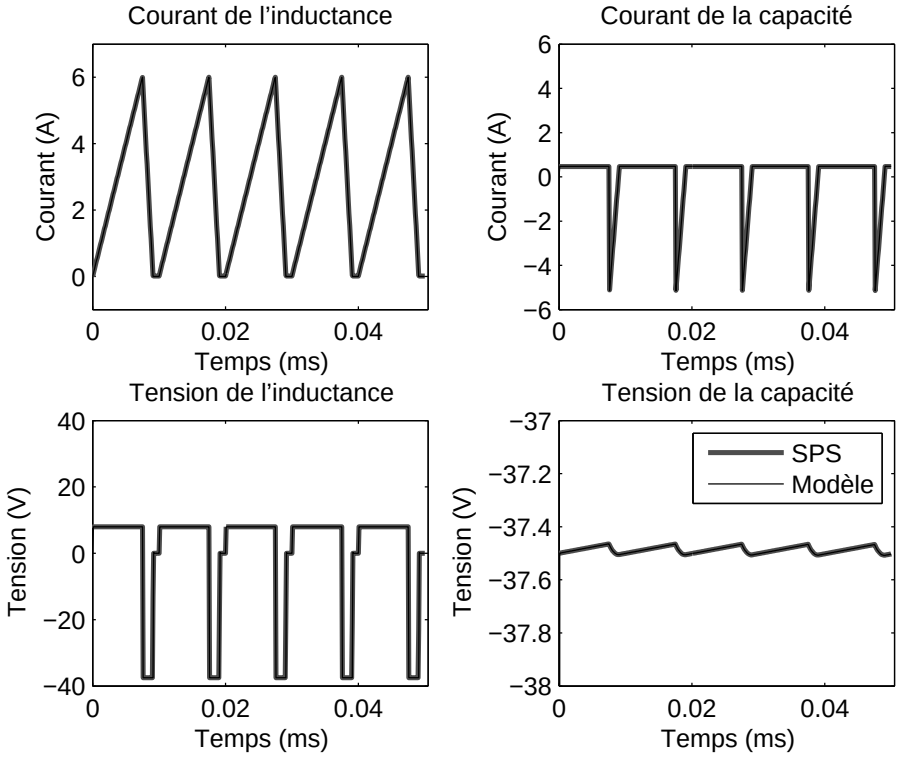

Figure 5.7 – Résultats de simulation corrigés du convertisseur buck-boost en mode discontinu.

En assumant que les interrupteurs ne changent pas de statut durant k sous-pas consécutifs (et que l'entrée $\mathbf{u}_n$ est constante par morceaux sur cet intervalle), il est alors possible d'écrire :

$$\left[\begin{array}{c} \mathbf{j}_{n+k+1} \\ \hline \mathbf{y}_{n+k} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{W_{ju}} + \mathbf{W_{jj}}\mathbf{X}_k^\sigma\mathbf{W_{ju}^\sigma} & \mathbf{W_{jj}}(\mathbf{W_{jj}^\sigma})^{k-1} \\ \hline \mathbf{W_{yu}} + \mathbf{W_{yj}}\mathbf{X}_k^\sigma\mathbf{W_{ju}^\sigma} & \mathbf{W_{yj}}(\mathbf{W_{jj}^\sigma})^{k-1} \end{array}\right] \left[\begin{array}{c} \mathbf{u}_n \\ \hline \mathbf{j}_n^\sigma \end{array}\right] \tag{5.8}$$

où $\mathbf{X}_k^\sigma = \mathbf{I} + \mathbf{W_{jj}^\sigma} + \ldots + (\mathbf{W_{jj}^\sigma})^{k-2}$, et $k \geq 2$.

La Figure 5.7 illustre le résultat de la simulation du buck-boost, obtenu en exploitant la méthode proposée. Le pas de temps est $\Delta t = 100$ ns, le sous-pas de temps est $\delta t = 1$ ps, et trois actualisation sont utilisées : $k_1 = 1$, $k_2 = 100$ et $k_3 = 99899$. Comme on peut le constater, les oscillations de convergence et les oscillations parasites ont complètement disparu et les résultats concordent parfaitement avec la référence SPS. Ce résultat est d'autant plus remarquable que le pas de calcul du modèle est cent fois plus lent que celui de la référence (SPS utilise un pas de calcul de 1 ns).

Néanmoins, il convient de relever que cette approche, bien que prometteuse, a certaines limites. Tout d'abord, comme l'indique l'équation 5.8, la matrice utilisée à chaque actualisation dépend de la combinaison des statuts des interrupteurs σ. De plus, puisqu'à l'intérieur d'un pas de temps Δt, la méthode nécessite des actualisations qui ne sont pas de même durée, il devient nécessaire de précalculer autant de matrices qu'il y a de blocs d'actualisation, et pour chacune d'elle 2^N combinaisons des statuts des N interrupteurs.

Ce requis en mémoire embarquée est plus important que celui des méthodes standards de pré-calcul des équations du réseau pour toutes les combinaisons des interrupteurs. L'avantage qu'elle offre est qu'elle dépend peu de la topologie du convertisseur de puissance, et permet de formuler les lois d'actualisation des statuts des interrupteurs de manière simple. Jumelée à des techniques de découplage efficaces, elle pourrait avoir une utilité pratique.

5.4 Implémentation FPGA

Dans cette section, nous discutons l'implémentation FPGA des engins de calcul dont nous avions esquissé l'architecture au Chapitre 2. Pour ce faire, nous débutons la discussion sur le format des nombres, à savoir le format en VF. Nous montrons que la double précision est pour le moment trop coûteuse pour la technologie FPGA. Néanmoins, en utilisant une largeur de mantisse non standard de précision intermédiaire, il est possible d'obtenir des résultats suffisamment exacts pour que la simulation présente une erreur relative inférieure à 1 %. La section se termine par la présentation d'une architecture d'engin de calcul versatile, capable de simuler des circuits d'électronique de puissance de topologie arbitraire. Trois circuits sont présentés pour illustrer cette versatilité.

5.4.1 Le format des nombres

La virgule flottante admet deux format principaux : la simple et la double précision. Dans le premier cas, la partie fractionnaire de la mantisse compte 23 bits, 52 bits dans le second cas. Afin d'exploiter adéquatement les multiplieurs signés des FPGA, il convient de prendre en compte la mantisse plus son signe et le bit caché, ce qui mène à 25 bits pour la simple précision et 54 bits pour la double précision. Il a été démontré dans [4, 17] qu'il est nécessaire, pour implémenter un multiplieur en double précision, d'exploiter entre 6 et 10 blocs DSP. Ceci reste trop exigeant pour nous qui ambitionnons de réaliser un engin de calcul comprenant 4 OPS4, ce qui représente 16 multiplieurs, soit un total dans l'intervalle de 96 à 160 blocs DSP. L'approche que nous proposons ici permettra de réduire ce nombre à 32 blocs DSP, tout en assurant une qualité appréciable des résultats de simulation.

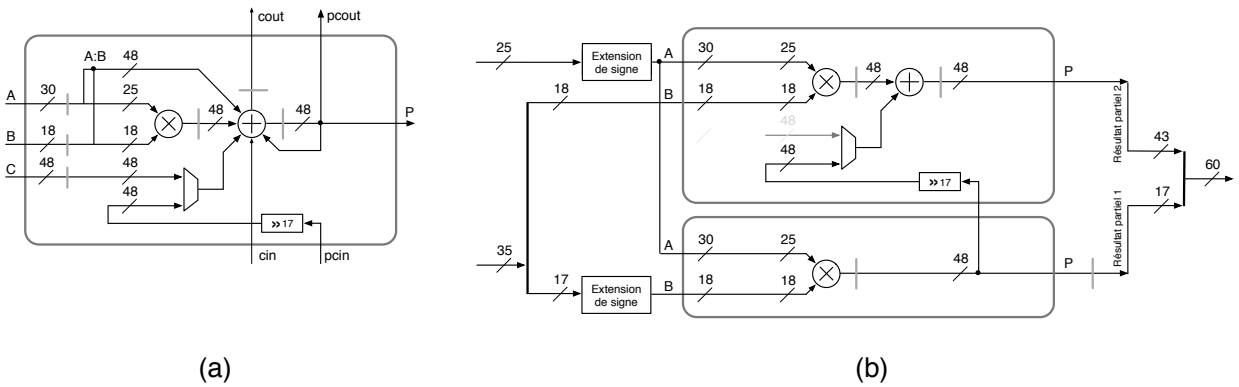

Figure 5.8 – Réalisation d'un multiplieur large au moyen de deux blocs DSP : (a) schéma simplifié du bloc DSP de la famille Virtex à compter de la 5^e génération ; (b) Multiplieur 25×35 signé formé en combinant deux blocs DSP.

La Figure 5.8.a présente sous une forme simplifiée un bloc DSP de la nouvelle génération des FPGA Virtex de Xilinx. Ce dernier comporte un multiplieur asymétrique 25×18 signé suivi d'un additionneur qui, par configuration statique ou dynamique, permet de réaliser une fonction de multiplieur additionneur fusionné ou de multiplieur accumulateur (MAC) en virgule fixe. Les blocs DSP des nouvelles générations de FPGA Virtex sont faits de telle manière qu'il est possible de les combiner pour former des multiplieurs plus larges, et ce en exploitant l'additionneur du bloc DSP pour effectuer la somme des produits partiels. Ainsi, le choix du multiplieur asymétrique 25×18 est une évolution des technologies Xilinx (qui comportaient jusqu'à la 4^e génération des Virtex des multiplieurs symétriques 18×18) pour faciliter l'implémentation de multiplieurs en simple précision. La Figure 5.8.b illustre comment il est possible de produire un multiplieur 25×35 en combinant deux blocs DSP.

Un multiplieur en simple précision n'exploite pas les dix bits supplémentaires obtenus en combinant les deux blocs DSP. Notre idée est de profiter de ces 10 bits pour élargir la mantisse des vecteurs $\mathbf{u}_n$ et $\mathbf{j}_n$. Dans les blocs DSP des anciennes générations de Virtex (ou encore dans les FPGA Spartan de Xilinx), les multiplieurs sont 18×18. Aussi, pour construire un multiplieur 25×25, il faut d'abord construire un multiplieur 35×35. Dans un tel cas de figure, nous proposons d'élargir aussi la mantisse des valeurs des matrice de 10 bits additionnels. Dans la section qui suit, nous étudions l'intérêt d'une telle approche dans l'implémentation des engins de calcul.

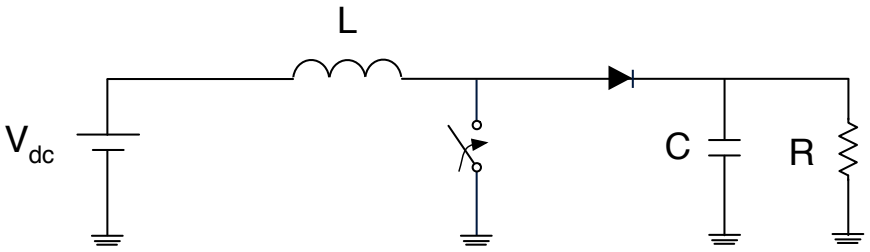

Figure 5.9 – Convertisseur boost servant à évaluer la performance de l'engin de calcul.

Paramètres	Valeurs	
R	8	Ω
L	50	μH
C	50	μF
V_{dc}	200	V
Fréquence de la MLI	1	kHz
Rapport cyclique	30	%

Tableau 5.2 – Paramètres du boost.

5.4.2 Implémentation d'un convertisseur boost

Nous considérons maintenant l'exemple de circuit boost donné dans [94] pour évaluer l'intérêt d'exploiter 10 bits additionnels dans la représentation de la mantisse du format en VF de l'engin de calcul. La Figure 5.9 présente le circuit boost considéré. Le Tableau 5.2 donne le détail des paramètres du circuit et de son opération.

Les engins de calcul utilisés pour cette démonstration ont été implémentés sur un Virtex 5 (XC5SX50T [119]) et sur un Spartan 3 (XC3S5000 [118]), deux FPGA auxquels le partenaire industriel nous donnait accès. Dans [94], le pas de calcul est de 100 ns. Un engin de calcul atteignant un tel pas de calcul peut être obtenu sur le Virtex 5, où la période de l'horloge est 5 ns, mais il est difficile à obtenir sur le Spartan 3 où seul un pas de calcul de 200 ns est possible (l'horloge a une période de 10 ns). Ce sont néanmoins des valeurs très honorables qui suffisent à la simulation du circuit, eu égard à la fréquence et au rapport cyclique de la MLI (voir Tableau 5.2).

Les engins de calcul implémentés exploitent uniquement des MAC. Pour pouvoir atteindre le pas de calcul désiré, nous avons utilisé au total 8 MAC. Les MAC exploitent un FAA$(4, 80, 183)$, ce qui signifie que la mantisse est large de 80 bits. Le Tableau 5.3 donne la surface occupée ainsi que la préformance en fréquence des MAC et des engins de calcul. La latence du MAC sur le Virtex 5 est de 11 cycles, elle est de 12 cycles sur le Spartan 3. Comme on peut le voir, les MAC atteignent la fréquence d'opération ciblée ; il en est de

Tableau 5.3 – Résultats d'implémentation pour le convertisseur boost

Métriques	Spartan 3	disponibles	Virtex 5	disponibles
Opérateur MAC				
Tranches	937 (2 %)	33,280	360 (6 %)	8,160
Blocs DSP	4 (3 %)	104	2 (1 %)	288
Chemin critique	9.912 ns	N/A	4.940 ns	N/A
Fréquence maximale	100.88 MHz	N/A	202.43 MHz	N/A
Engin de calcul comprenant 8 MAC				
Tranches	6,921 (20 %)	33,280	2,768 (34 %)	8,160
Blocs DSP	32 (30 %)	104	16 (5 %)	288
Blocs RAM	9 (8 %)	104	5 (3 %)	132
Chemin critique	9.985 ns	N/A	4.973 ns	N/A
Fréquence maximale	100.15 MHz	N/A	201.09 MHz	N/A

même pour les engins de calcul. Ceci est notamment rendu possible grâce au format HRCS utilisé pour l'accumulation à un cycle des mantisses de 80 bits. On remarquera aussi que les engins de calcul occupent 20 % et 34 % du Spartan 3 et du Virtex 5 respectivement. Ceci signifie que les FPGA disposent encore de suffisamment de place pour accueillir des engins de calcul supplémentaires.

La Figure 5.10 présente la tension à la charge obtenue lors de la simulation du convertisseur boost sur une durée de 30 ms. La figure présente également l'évolution de l'erreur relative en pourcentage pour les deux engins de calcul, où une simulation en double précision est utilisée comme base de référence. La Figure 5.10 donne à titre de référence l'erreur relative résultant d'une simulation où la simple précision standard est utilisée. Comme on peut le constater, l'erreur est toujours inférieure à 1 %. Dans le cas de la simple précision standard, elle est proche de 0.1 %. L'engin implémenté sur le Virtex 5 quant à lui affiche une erreur relative variant entre 0.001 % et 0.01 %. L'amélioration de la qualité de la simulation par rapport à la simple précision attribuable au FAA utilisé dans l'accumulation, mais également aux dix bits supplémentaires dans la représentation des valeurs des vecteurs. Ceci est d'autant plus évident lorsque l'on considère l'erreur relative générée par la simulation sur le Spartan 3 qui varie entre 0.0001 % et 0.001 %

5.4.3 Engin de calcul versatile

L'engin de calcul que nous présentons ici est à la base un prototype de produit commercial qui a servi à une démonstration faite à un client de Opal-RT Technologies, la compagnie GE aviation. Le prototype est aujourd'hui commercialisé par Opal-RT sous le nom eHS pour

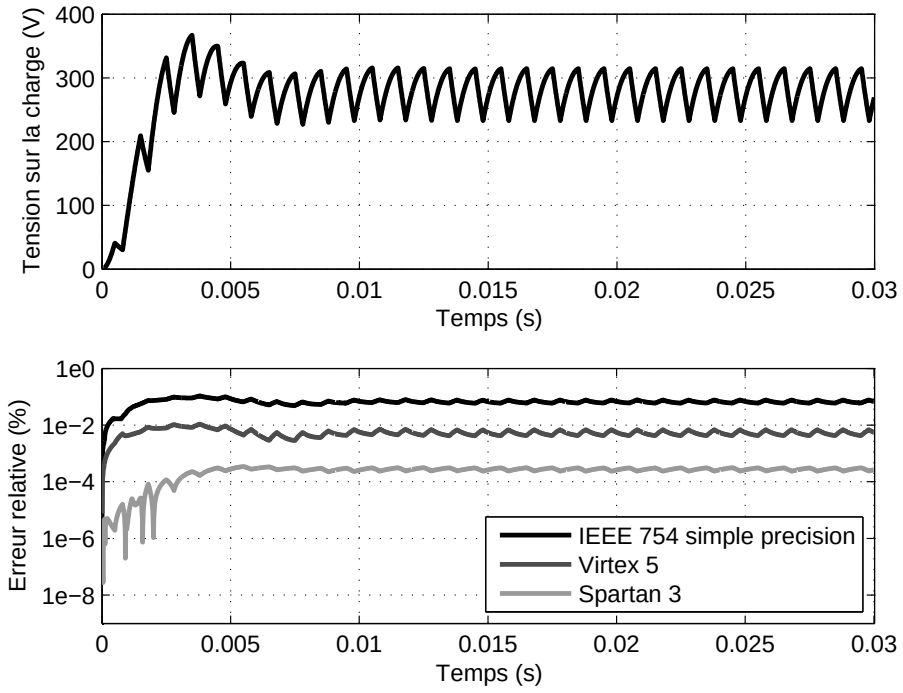

Figure 5.10 – Tension de charge et erreur relative dans la simulation du convertisseur boost.

eHardwareSolver. Sa versatilité a permis d'en faire un simulateur sur FPGA que l'usager peut exploiter en concevant un convertisseur depuis l'interface graphique de SPS [2]. L'eHS permet de simuler des convertisseurs de puissance de topologie arbitraire disposant d'un maximum de 24 interrupteurs, de 16 entrées, 8 sorties (la version commerciale offre jusqu'à 16 sorties) et plusieurs dizaines (jusqu'à 60) composants L/C. Le nombre de résistances dans le réseau est illimité. Nous avons publié les détails de ce travail dans [85], publication dont nous reprenons ici une large partie des résultats. Il est à noter que la version commerciale de l'eHS est uniquement destinée aux FPGA Virtex tandis que les résultats présentés ici considèrent également une implémentation de l'eHS sur Spartan 3.

La Figure 5.11 illustre l'architecture générale de l'eHS. On reconnaît ici l'esquisse que nous en avions donné à la Figure 2.2, avec cette différence qu'un module supplémentaire est ajouté pour le support de l'actualisation des interrupteurs du modèle à matrice constante, selon les lois énoncées précédemment. L'eHS utilise 4 OPS4 parallèles. Ces opérateurs ont été présentés au Tableau 3.3 et discutés au Chapitre 3. Les opérateurs OPS4 utilisés diffèrent légèrement de celui que l'on peut observer à la Figure 2.3.b, en cela qu'un soustracteur TAA

2. Le lecteur intéressé pourra trouver une démonstration vidéo du fonctionnement de l'eHS à l'adresse `www.youtube.com/watch?v=Cfce0FN9kSw`.

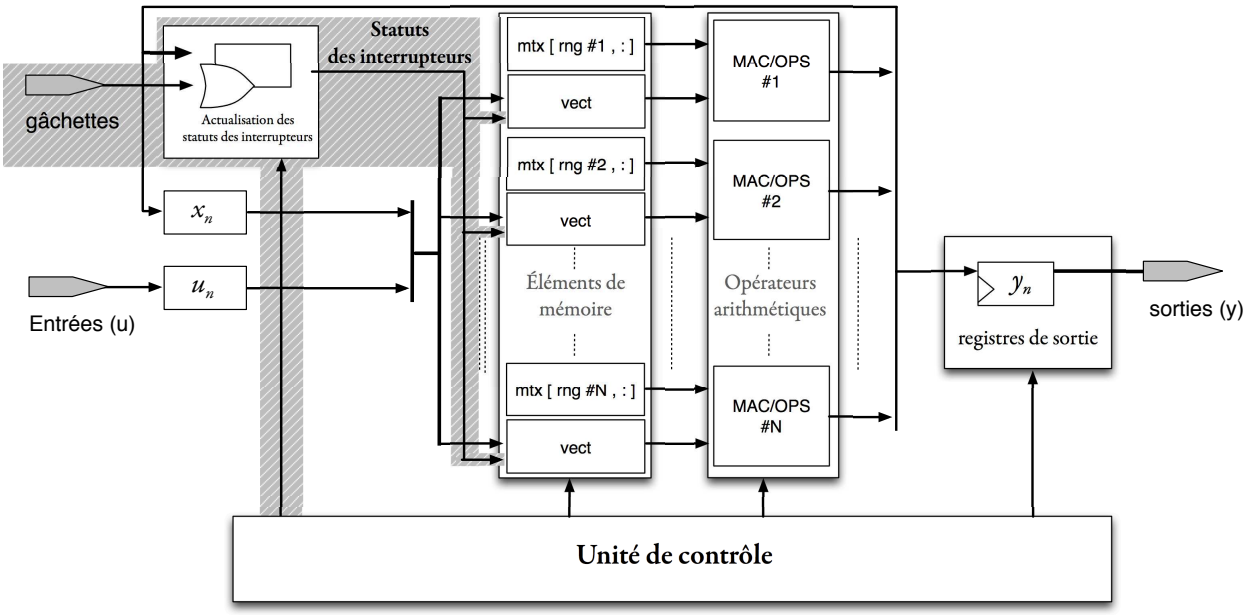

Figure 5.11 – Topologie d'un engin de calcul pour résoudre les équations MANA. La zone grisée est spécifique aux circuits d'électronique de puissance.

suit la boucle d'accumulation pour réaliser la relation $\mathbf{j}_{n+1}^{s_1} = \mathbf{j}_n - \mathbf{j}_{n+1}^{s_0}$. La version Virtex 5 (XC5SX50T [119]) de l'eHS cible une fréquence de fonctionnement de 200 MHz, la cible est de 100 MHz pour le Spartan 3 (XC3S5000 [118]). Avec de telles fréquences, on peut déduire que l'eHS offre une puissance de calcul de 6.4 GFLOPS sur le Virtex 5, la puissance de calcul étant de 3.2 GFLOPS sur le Spartan 3. Le Tableau 5.4 présente les résultats d'implémentation de l'eHS tels qu'obtenus avec la version 10.1 du logiciel ISE de Xilinx. On y constate d'une part que les fréquences cibles sont effectivement atteintes. De plus, on relèvera que les engins de calcul occupent 32 % et 35 % des ressources reconfigurables des FPGA Spartan 3 et Virtex 5 respectivement. Ce résultat est d'autant plus remarquable que les nouvelles générations de la famille Virtex offrent des FPGA beaucoup plus denses. Par exemple, le Virtex 6 XC6LX240T-1C [122] disponible sur la carte ML605 que l'on retrouve sur les nouveaux simulateurs de la compagnie Opal-RT disposent de $37,680$ tranches (*slices*), ainsi que de 768 blocs DSP, comparativement à $8,160$ tranches et 288 blocs DSP sur le Virtex 5 que nous considérons dans nos résultats.

Afin de démontrer la versatilité de l'eHS, nous avons ciblé trois circuits d'électronique de puissance différents, que le lecteur retrouvera schématisé à la Figure 5.12. Plus spécifiquement, les circuits considérés sont : *i*) circuit #1, un convertisseur boost ; *ii*) circuit #2, un convertisseur deux niveaux triphasé ; *iii*) circuit #3, un convertisseur boost alimentant un

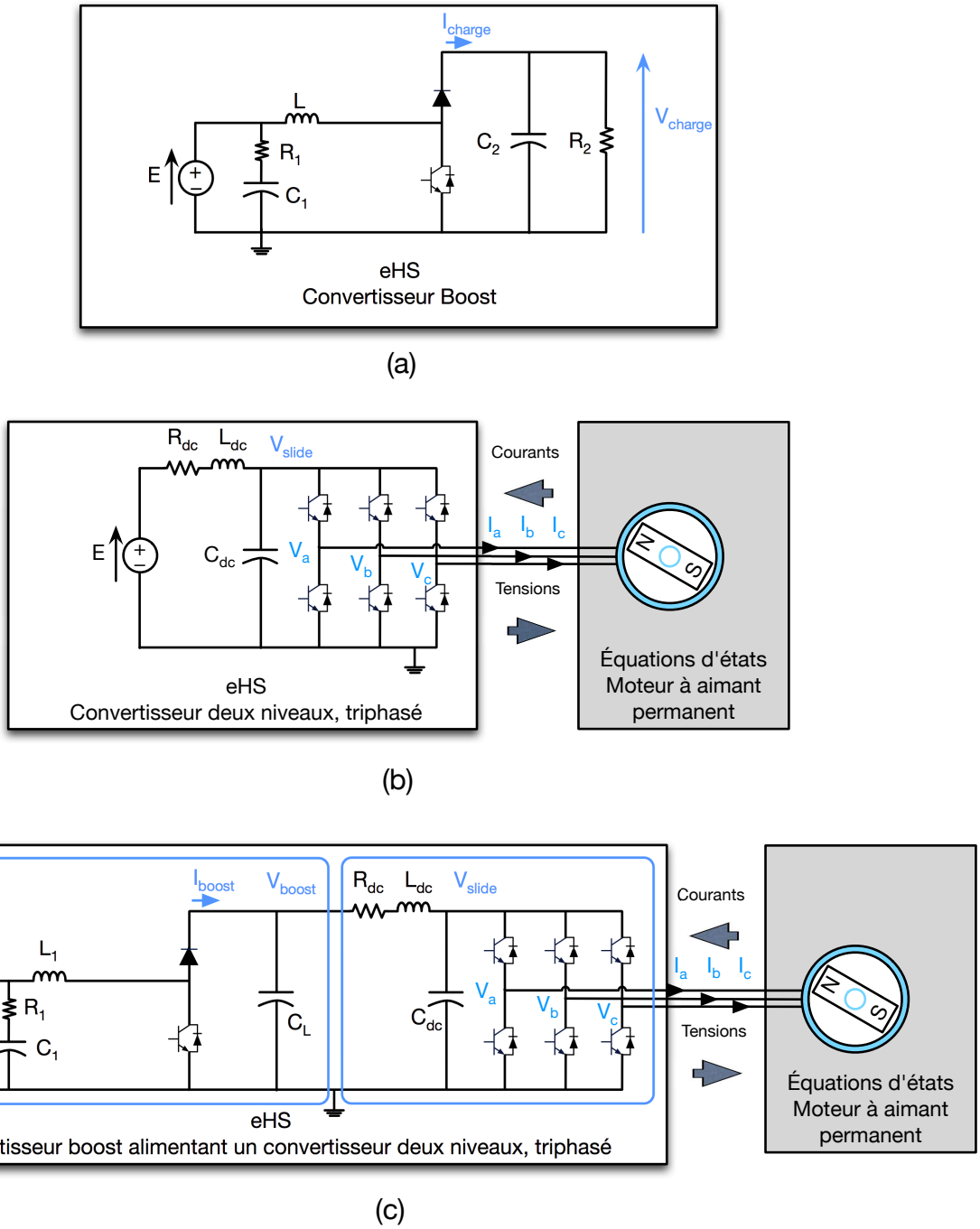

Figure 5.12 – Trois circuits d'électronique de puissance servant à tester l'eHS : (a) Un convertisseur boost ; (b) Un convertisseur deus niveaux, triphasé connecté à un modèle de moteur à aimant permanent ; (c) Un convertisseur boost alimentant un convertisseur triphasé, le tout connecté à un moteur à aimant permanent.

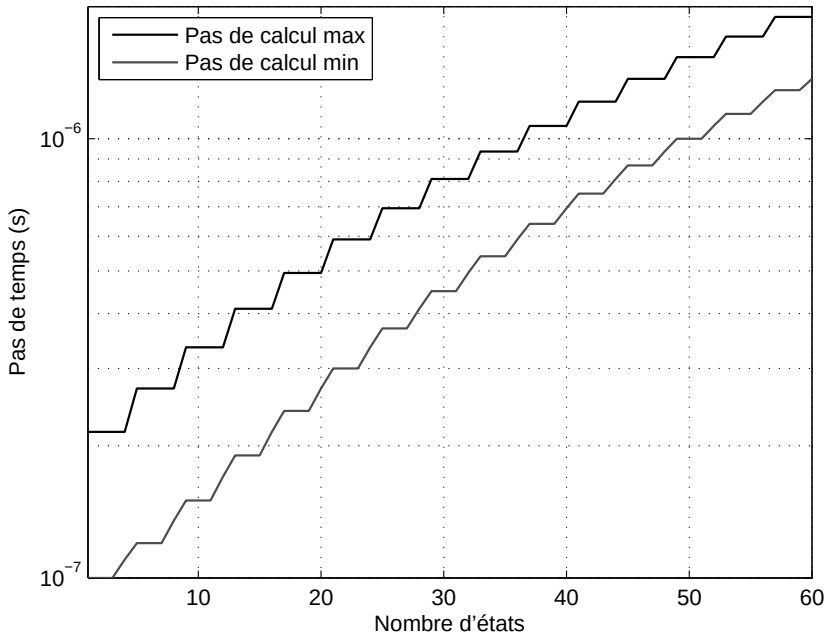

Figure 5.13 – Évolution du pas de calcul de l'eHS en fonction du nombre d'états.

convertisseur deux niveaux triphasé. Les circuits #1 et # 2 sont connectés à des moteurs à aimant permanent (PMSM) dont les équations sont exécutées par un modèle d'équations d'états que réalise un engin de calcul fait d'un seul MAC multi-cycles, tel que décrit dans [84]. Le Tableau 5.5 donne les pas de calcul obtenus pour les différents convertisseurs de puissance considérés selon qu'un Virtex 5 ou un Spartan 3 est utilisé. Le tableau présente également la taille de la matrice à titre de référence.

La Figure 5.13 illustre l'évolution du pas de temps en fonction du nombre d'états ($|\mathbf{j}_n|$),

Tableau 5.4 – Résultats d'implémentation de l'eHS.

Métriques	Spartan 3	disponibles	Virtex 5	disponibles
Tranches	10,776 (32 %)	33,280	2,933 (35 %)	8,160
Registres	9,436 (14 %)	66,560	5,370 (16 %)	32,640
LUTs	16,683 (25 %)	66,560	7,374 (22 %)	32,640
Blocs DSP	64 (61 %)	104	32 (11 %)	288
BRAM	45 (43 %)	104	63 (47 %)	132
Chemin critique	9.979 ns	N/A	4.976 ns	N/A
Fréquence maximale	100.21 MHz	N/A	200.96 MHz	N/A

Tableau 5.5 – Circuits considérés pour démontrer la versatilité de l'eHS

		Pas de calcul	
Circuit	Taille de la matrice	Virtex 5	Spartan 3
Circuit #1	9×6	120 ns	260 ns
Circuit #2	18×11	135 ns	290 ns
Circuit #3	28×17	215 ns	450 ns

selon que le nombre d'entrées sorties soit minimal ($|\mathbf{u}_n| = |\mathbf{y}_n| = 1$) ou maximal ($|\mathbf{u}_n| = |\mathbf{y}_n| = 16$). Comme on peut le constater, le pas de temps est en deçà de la limite de 1 μs pour une grande variété de problèmes. C'était d'ailleurs un des critères de conception de l'eHS lorsque nous l'avions réalisé. On constatera également que le pas de temps évolue suivant des créneaux (paliers de quatre points consécutifs), qui s'expliquent par le fait que les équations sont réparties sur des OPS4.

La manière dont ces circuits sont pris en charge par l'eHS est la suivante. Un script Matlab est utilisé pour décrire les équations MANA du réseau. Un second script Matlab prend en entrée les équations décrites par le premier script et les convertit en instructions à exécuter par l'unité de contrôle et en contenu des mémoires BRAM. Ce second script ajuste le pas de calcul à la taille du problème, de sorte à garantir le plus petit pas de calcul possible. Ainsi, plus le problème est petit, et plus le pas de temps l'est aussi.

Afin de démontrer le bon fonctionnement de l'eHS, nous n'avons retenu que les résultats de simulation du circuit #3 puisqu'il regroupe les deux premiers. La Figure 5.14 présente une simulation de 0.25 s de durée où sont illustrés les courants du moteur ainsi que les voltages à la sortie du boost et sur le rail DC (après le filtre LC). Le boost et le pont à deux niveaux sont stimulés par des MLI de 20 kHz. Ceci est considéré comme une vitesse de commutation très élevée au regard de se qui se fait généralement dans le cadre de la simulation en temps réel, où la plus haute fréquence rapportée dans la littérature est 8 kHz [75]. Pour notre simulation, la vitesse du moteur est fixée à 400 Hz puis subitement changée à 100 Hz vers 0.125 s du temps de la simulation. Ce brusque changement est effectué afin de démontrer un bon comportement en transitoire du circuit. La comparaison avec la la référence SPS a indiqué une bonne exactitude des résultats, avec des erreurs relatives variant entre 0.1 % et 1 %. Ceci est d'autant plus remarquable que la référence SPS utilise une représentation à double précision et que l'intégration y est effectuée à pas variables, tandis que le moteur simulé sur le FPGA est simulé au moyen d'une simple précision et en utilisant une intégration à pas constants suivant une discrétisation d'Euler avant (la discrétisation Euler explicite a été employée car les équations d'états sont variantes [84]).

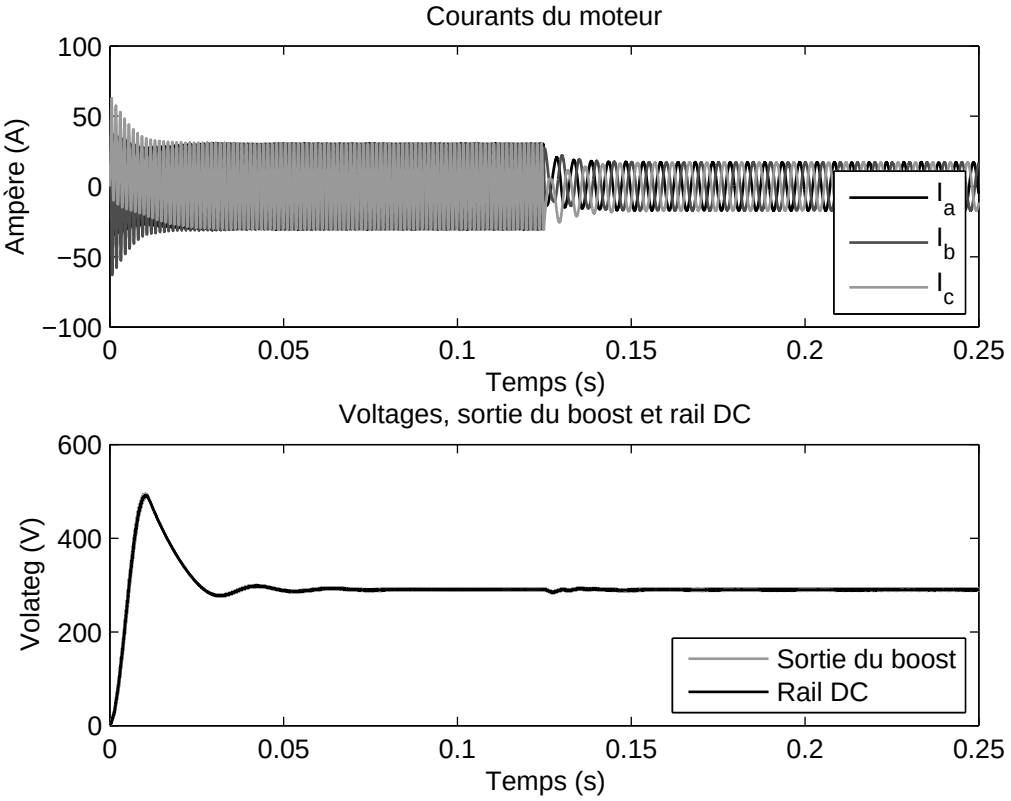

Figure 5.14 – Résultats de simulation du circuit #3 : (a) Courants du moteur PMSM ; (b) Tensions à la sortie du boost et sur le rail DC (après le filtre LC).

5.5 Études de cas utilisant le modèle résistif de l'interrupteur

L'eHS est un outil pratique et facile à utiliser, permettant d'effectuer des études de faisabilité et du prototypage rapide. Cependant, dans certains cas, il est préférable de disposer d'un engin de calcul spécifique à une application. Le modèle résistif de l'interrupteur s'avère alors utile pour garantir une bonne précision au modèle. Cette section considère deux études de cas du genre auxquels nous avons été confronté.

Le premier cas est celui d'un onduleur servant à alimenter un moteur à aimant permanent. Le modèle devait être en mesure de supporter différents scénarios de défaut, sur les interrupteurs, du côté triphasé du convertisseur (moteur) ainsi que du côté à courant continu (batterie). Le modèle devait également pouvoir se connecter à différents types de modèles de moteur. Cette application était particulièrement critique car elle répondait à une commande d'un client industriel du marché automobile et que les fréquences de commutations visées était de plusieurs dizaines de kilohertz.

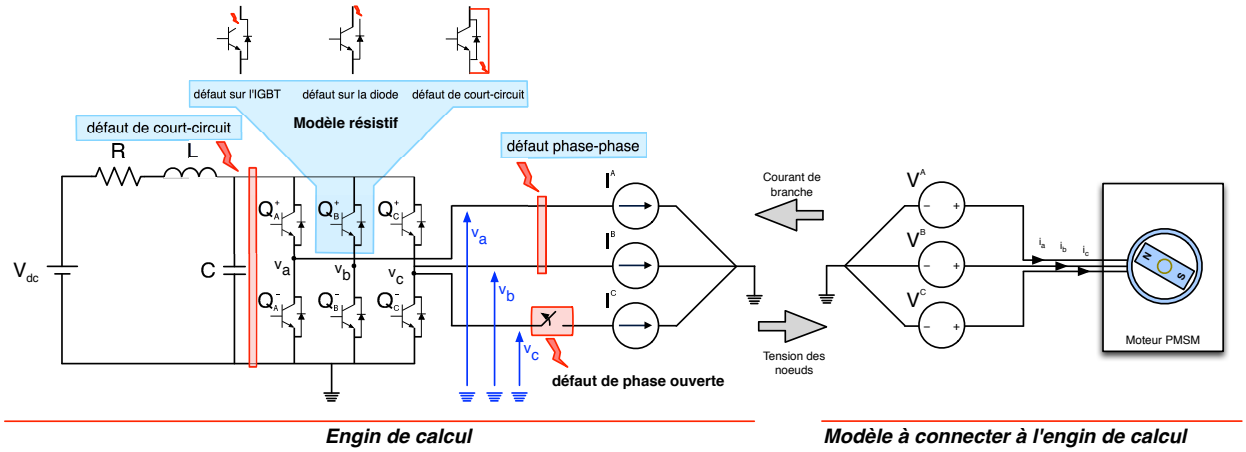

Figure 5.15 – Circuit d'onduleur découplé avec identifications des défauts supportés par le modèle.

Le second cas d'étude est motivé par une problématique rencontrée par un collègue doctorant qui cherchait à simuler en temps réel un convertisseur multi-niveaux (*multi-level converter* — MMC). Ses travaux avaient en effet démontré que pour être en mesure de considérer un MMC à 401 niveaux, le pas de calcul devait être en deçà des 10 μs, ce qui était impossible à réaliser en n'employant que des CPU. Ce cas démontre l'intérêt d'utiliser un FPGA pour ce type d'applications, mais également que le modèle résistif de l'interrupteur peut être considéré dans certains cas de convertisseurs de puissance utilisant plusieurs milliers de semi-conducteurs.

5.5.1 Pont à deux niveaux

Ce cas d'étude provient d'une commande de Toyota à Opal-RT pour des besoins de prototypage de contrôleurs ECU pour ses véhicules hybrides. Opal-RT a ensuite collaboré avec la compagnie Denso pour valider le bon fonctionnement de l'engin de calcul dans différents contextes en y connectant un modèle de moteur réalisé par éléments finis [49]. Cette approche permet d'extraire les caractéristiques spécifiques et détaillées des moteurs électriques visés et d'exécuter des tests de façon très réaliste selon l'approche HIL.

La Figure 5.15 présente le schéma du circuit modélisé. Les différents scénarios de défaut considérés y sont clairement identifiés. On constatera par exemple que trois types de défauts sont admissibles sur la paire IGBT/diode : IGBT ouvert, diode ouverte ou faute de court-circuit. De la même façon, deux défauts sont possibles du côté moteur : un défaut de court-circuit phase-phase et un défaut de phase ouverte (uniquement sur la phase C).

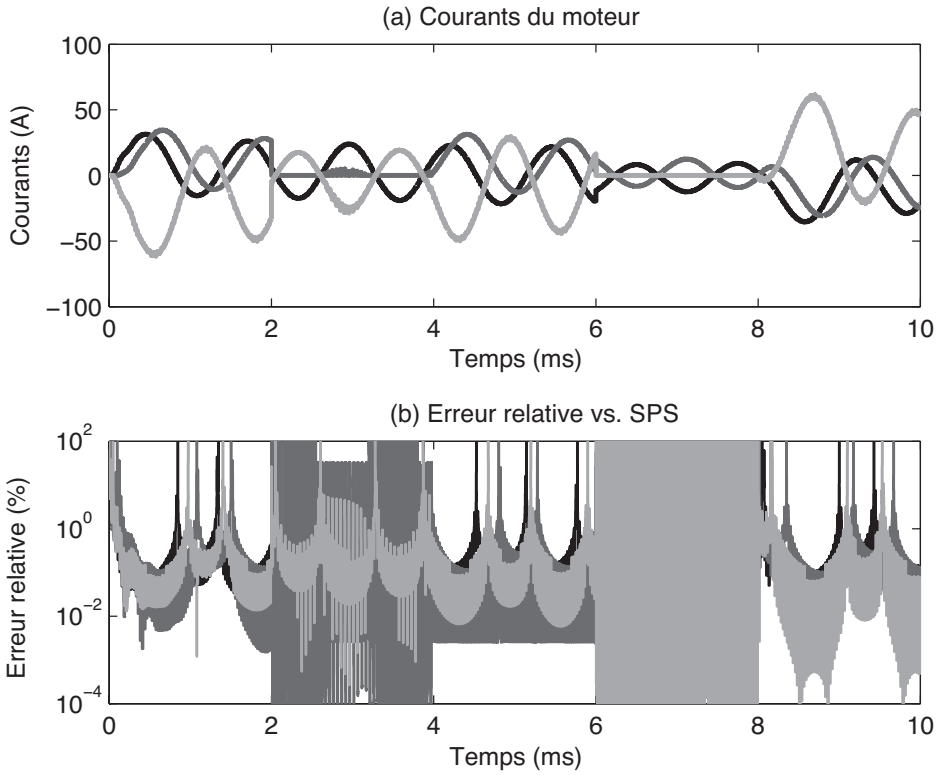

Figure 5.16 – Simulation d'un onduleur alimentant un moteur PMSM.

L'engin de calcul que nous avons réalisé pour cette application comporte 4 MAC. La mantisse TAA est de 64 bits. Les données de la matrice sont représentés avec une mantisse signée de 25 bits tandis que le vecteur des états utilise une mantisse signée de 35 bits. Ce choix a été justifié précédemment à la Section 5.4.1. Le Tableau 5.6 présente les résultats d'implémentation de l'engin de calcul sur le Virtex 6 XC6LX240T-1C [122] après placement et routage. On relèvera que les résultats de consommation de mémoire concordent ici avec les résultats de la Figure 5.1. Le nombre d'états étant de trois [3], on voit bien que la mémoire utilisée par l'engin de calcul est bel et bien inférieure à 1 % des ressources en mémoire embarquée disponibles sur le FPGA Virtex 6.

La validation du bon fonctionnement de l'engin de calcul s'est faite en simulation (modèle précis au bit près et au cycle d'horloge près) ainsi qu'en temps réel, sur les simulateurs d'Opal-RT. Pour la simulation en temps différé, le modèle de moteur est de type DQ, et les résultats de la simulation sont comparés à une référence SPS. Dans nos tests, différents scénarios de fautes ont été considérés pour le travail (un total de 25 pour être exact). La Figure 5.16 présente un de ces cas testés, où les courants du moteur peuvent être observés à

3. La faute du côté DC est réalisée avec un modèle d'interrupteur à matrice fixe.

Tableau 5.6 – Résultats d'implémentation du pont à deux niveau.

Métriques	Virtex 6	disponibles
Tranches	2,000 (6 %)	37,680
Blocs DSP	8 (< 1 %)	768
BRAM	9 (< 1 %)	416
Chemin critique	4.964 ns	N/A
Fréquence maximale	201.45 MHz	N/A

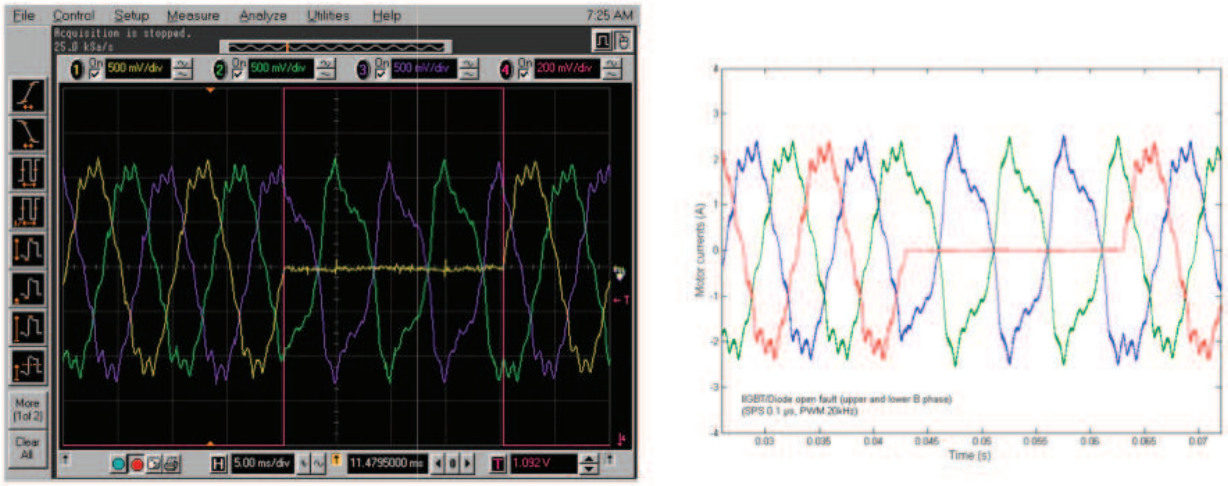

Figure 5.17 – Simulation en temps réel d'un onduleur alimentant un moteur PMSM.

la Figure 5.16.a. La MLI utilisée est commutée à 20 kHz alors que la rotation du moteur est fixée à 400 Hz. Durant l'intervalle de temps allant de 2 à 4 ms, un défaut de type diode et IGBT ouverts est appliqué à l'interrupteur supérieur de la phase B ($Q_{\mathbf{B}}^{+}$). Dans l'intervalle de temps allant de 6 à 8 ms, c'est un défaut de type phase ouverte qui est appliquée sur la phase C du moteur.

La Figure 5.16.b présente l'erreur relative obtenue sur le courant en comparaison d'une référence SPS exécutée en temps différé. On voit qu'aux croisements par zéro, l'erreur relative augmente mais que globalement, en opération normale, elle varie entre 0.01 % et 1 %. Ces niveaux de précision sont généralement jugés comme excellents pour la simulation en temps réel, où des approximations de l'ordre de 5 % sont jugées admissibles. Finalement, remarquons que durant les fautes, les erreurs relatives oscillent beaucoup. Cet état des choses est plutôt acceptable dans le présent cas de figure car les courants durant ce temps sont proches de zéro.

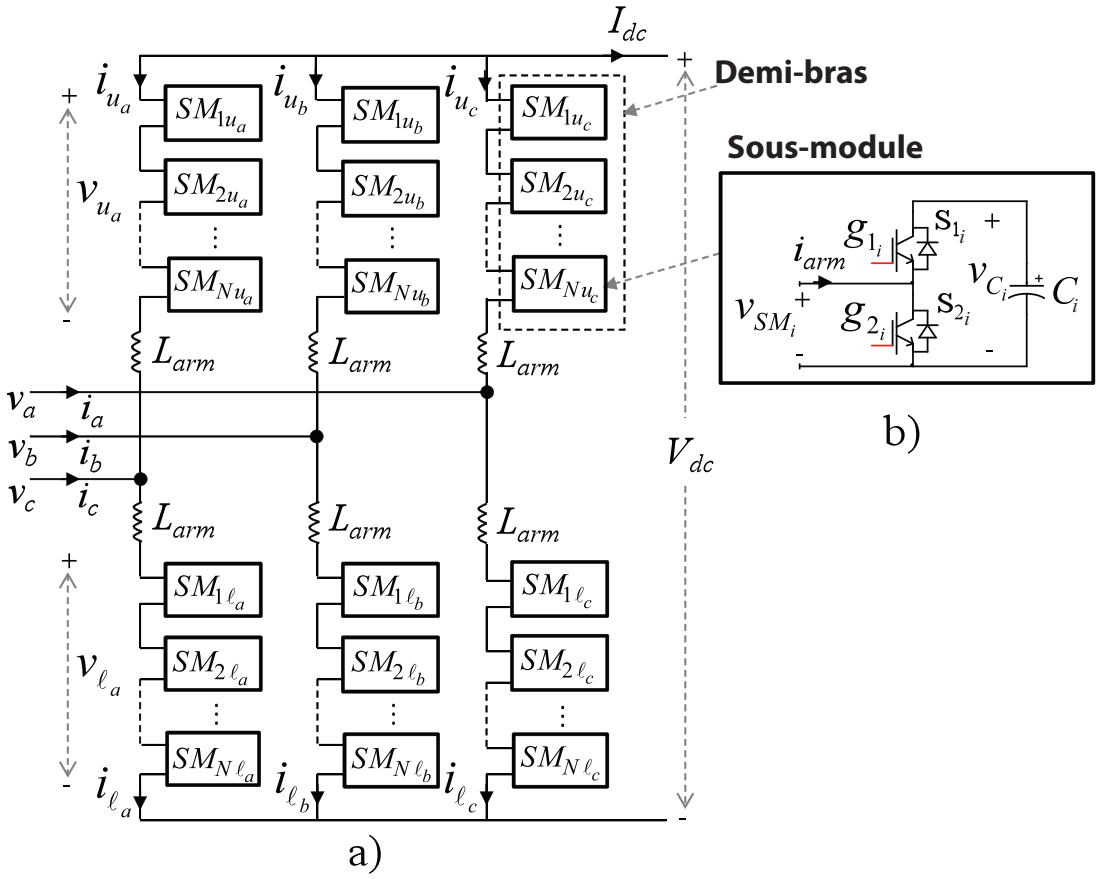

Figure 5.18 – Structure du MMC.

Pour le test en temps réel, le modèle de moteur utilisé est un modèle généré par les techniques d'éléments finis. Nous reproduisons à la Figure 5.17 une capture d'oscilloscope présentée dans [49] et illustrant un défaut de phase ouverte. Une simulation en temps différé tirée de SPS est présentée à droite pour illustrer la correspondance des deux modèles. D'autres cas de fautes sont illustrés dans la publications, ainsi que le résultats de tests sur le modèle de moteur.

5.5.2 Convertisseur multi-niveaux

Le second cas d'étude est celui d'un MMC, convertisseur de puissance utilisé dans les réseaux de très haute tension à des fins d'interface entre un réseau à courant alternatif et une ligne de transport à courant continu. Cette technologie connaît un grand intérêt de par le monde, en font foi les nombreux projets de mise en service de MMC, dont le lien à courant continu sur 70 km entre la France et l'Espagne est une émanation [96]. C'est d'ailleurs dans

le cadre de ce projet que s'inscrit le travail de notre collègue, où un convertisseur MMC de 401 niveaux [4] doit être opéré. Les résultats présentés dans cette section sont tirés d'un article de revue récemment soumis pour publication [105].

La Figure 5.18 présente un MMC où on retrouve le réseau à courant alternatif du côté gauche du convertisseur, et la ligne de transport à courant continu du côté droit. Chaque phase du convertisseur comporte deux demi-bras, chacun de ces demi-bras étant constitué de N sous-modules (SM). Les sous-modules du demi-bras sont des cellules formées de deux paires IGBT/diode ; leur rôle et de relier plusieurs capacités en cascade pour sommer les voltages à leurs bornes. Le nombre de cellules actives peut ainsi varier de 0 et N, de sorte que le convertisseur puisse discrétiser un signal alternatif en $N + 1$ niveaux au plus. Dans ce travail, il est supposé que les SM sont opérés de telle façon qu'en tout temps, une et seulement une des deux paires diode/IGBT conduise.

Malgré le grand nombre d'interrupteurs que comporte un MMC (un MMC à 401 niveaux comprend 4 800 paires IGBT/diode), la modélisation du convertisseur au moyen du modèle résistif de l'interrupteur est néanmoins possible. L'idée a été proposée par [34] où il est suggéré de remplacer chaque demi-bras par son équivalent de Thévenin, et ce après avoir converti chaque SM en son équivalent de Thévenin. Les équations pour effectuer ces manipulations sont relativement simples et se traduisent comme suit. Les interrupteurs S_{1_i} et S_{2_i} sont remplacés par des équivalents résistifs, R_{1_i} et R_{2_i} respectivement. On aura alors la résistance de Thévenin d'un SM_i exprimée par :

$$R^{Th}_{\mathrm{SM}_i} = R_{2_i}\left(1 - \frac{R_{2_i}}{R_{1_i} + R_{2_i} + R_{C_i}}\right) \tag{5.9}$$

où R_{C_i} est la résistance équivalente issue de la discrétisation de la capacité C_i. La tension de Thévenin de la cellule SM_i quant à elle est donnée par :

$$v^{Th}_{\mathrm{SM}_i} = \left(\frac{R_{2_i}}{R_{1_i} + R_{2_i} + R_{C_i}}\right) V_{C_i} \tag{5.10}$$

où v_{C_i} est la tension aux bornes de la capacité C_i.

Les équations 5.9 et 5.10 étant posées, il est possible de déterminer la résistance et la tension équivalentes de Thévenin du demi-bras [5] assez facilement, puisque les SM sont connectés en série.

4. Nous revenons plus loin sur la définition du nombre de nouveaux dans un MMC.

5. Il convient de remarquer que ce que l'on désigne par demi-bras en français se traduit par *arm* en anglais. Le lecteur est donc prié de ne pas s'étonner de la notation qui suit.

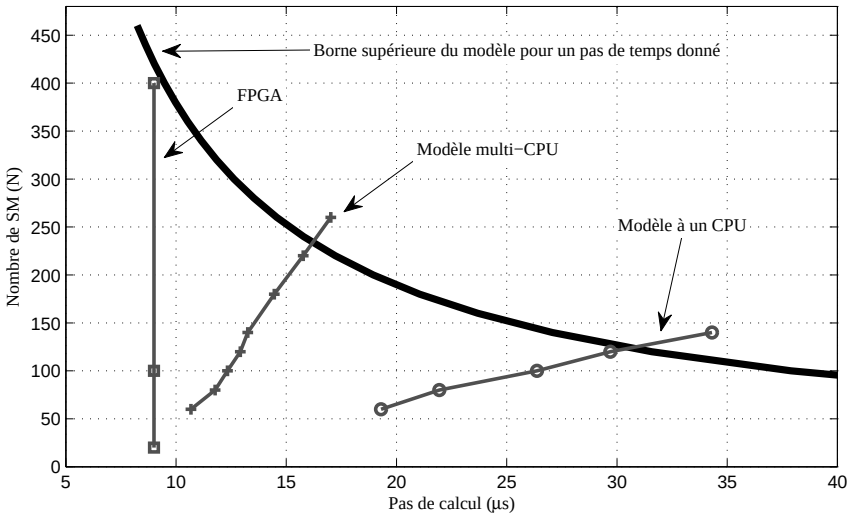

Figure 5.19 – Relation entre le pas de temps et le nombre de SM pouvant être considérés.

Ainsi, la résistance de Thévenin pour le demi-bras est donnée par :

$$R_{arm}^{Th} = \sum_{i=1}^{N} R_{\mathrm{SM}_i}^{Th} \tag{5.11}$$

tandis que la tension de Thévenin du demi-bras est donnée par :

$$v_{arm}^{Th} = \sum_{i=1}^{N} v_{\mathrm{SM}_i}^{Th} \tag{5.12}$$

Ces équations simplifient grandement les calculs pour la simulation du modèle, rendant possible la simulation en temps réel d'un MMC avec près de 100 SM en se servant uniquement de CPU [104]. Cependant, pour un MMC à 400 SM, une simulation sur CPU ne parvient pas à respecter la contrainte que se doit de suivre le modèle, et qui est illustrée à la Figure 5.19. La preuve mathématique de cette limite n'est pas reproduite ici afin d'alléger le texte. Le lecteur intéressé la retrouvera dans [105]. Cependant, si on s'y fie, il apparaît que, pour être en mesure de simuler un MMC à 400 SM, il faut que le pas de temps de la simulation soit entre 9 et 10 μs. Or, la Figure 5.19 indique qu'un modèle simulé sur un CPU sera en mesure de rouler en temps réel en autant que le nombre de SM ne dépasse pas 120. Si le modèle est réparti sur plusieurs CPU, la simulation en temps réel pourrait considérer jusqu'à 230 SM. Ces chiffres doivent cependant être revus à la baisse puisqu'ils ne prennent pas en compte les temps de communication avec un contrôleur externe (le contrôleur est émulé sur CPU).

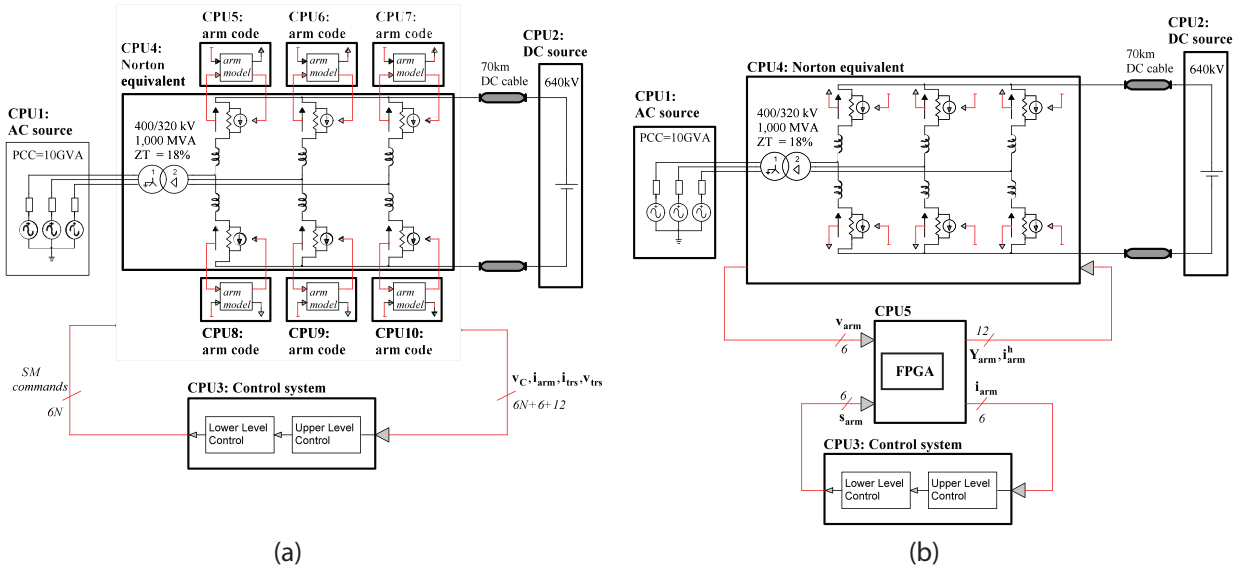

Figure 5.20 – Infrastructure matérielle pour la simulation du MMC : (a) Cas de simulation Multi-CPU où chaque demi-bras est simulé sur un CPU indépendant ; (b) Cas de simulation où le MMC (six demi-bras) sont simulés sur FPGA.

La Figure 5.19 montre que la solution FPGA permet de résoudre ce dilemme. Une implémentation FPGA permettant de ce faire est présentée à la Figure 5.20.b (le pendant multi-CPU — dont les résultats étaient donnés à la Figure 5.19 — est illustré à la Figure 5.20.a). Il s'avère en fait que le MMC est simulé sur le FPGA avec un pas de calcul de 2.5 μs. C'est le temps de communication entre le CPU et le FPGA (fixée à 9 μs) qui limite le pas de simulation totale de cette solution. Ce résultat est intéressant en cela qu'il nous montre que l'amélioration du lien de communication entre le CPU et le FPGA (c.-à-.d. la diminution de la latence de ce canal) a un fort potentiel dans l'amélioration des performances des simulateurs en temps réel. On remarquera dans la Figure 5.20 que les MMC sont représentés par des équivalents Norton et non des équivalents Thévenin. Ce choix est expliqué par des raisons technologiques, à savoir l'utilisation de la boîte à outils SSN de Opal-RT qui nécessite une représentation nodale. Néanmoins, de par la simplicité de la formulation de l'équivalent Thévenin des demi-bras, le FPGA effectue ses calculs selon ce modèle — ces derniers sont ensuite traduits vers des équivalents Norton sur le CPU5 puis intégrés au reste du réseau.

La topologie de l'engin de calcul que nous avions développée jusqu'ici ne peut être exploitée pour effectuer les calculs rattachés à un demi-bras de MMC. Afin de déterminer l'architecture à utiliser pour notre problème, il nous faut faire un sommaire des quantités

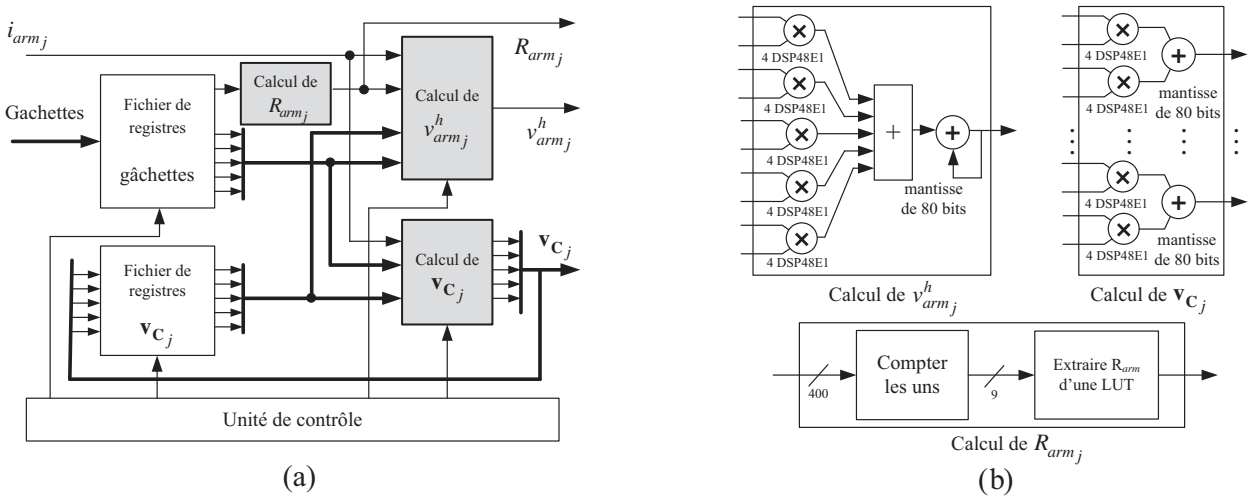

Figure 5.21 – Engin de calcul pour la simulation du MMC : (a) Topologie de l'engin de calcul ; (b) opérateurs arithmétiques associés aux unités de calcul.

devant être évaluées (R_{arm_j}, $v^h_{arm_j}(t)$ et les valeurs des $v_{C_{ij}}(t + \Delta t)$ pour le pas de temps suivant), les valeurs en entrée des unités de calcul (i_{arm_j} et les valeurs actuelles $v_{C_{ij}}(t)$) ainsi que la façon de ce faire. Notons que les indices j servent à distinguer les différents demi-bras entre eux. Relevons à ce titre que :

$$v^h_{arm_j}(t) = R_{arm_j} i_{arm_j}(t) + \sum_{i=1}^{N} \alpha_{ij} v_{C_{ij}}(t) \tag{5.13}$$

où α_{ij} est une constante qui dépend des statuts des interrupteurs $S_{1_{ij}}$ et $S_{2_{ij}}$. De la même façon, on trouvera que :

$$v_{C_{ij}}(t + \Delta t) = R_{eq_{ij}} i_{arm_j}(t) + \beta_{ij} v_{C_{ij}}(t) \tag{5.14}$$

où β_{ij} est également une constante qui dépend des statuts des interrupteurs $S_{1_{ij}}$ et $S_{2_{ij}}$. Quant à R_{arm_j}, son expression (voir les équations 5.9 et 5.11) permet d'entreposer ses valeurs potentielles dans une mémoire LUT adressées par un pointeur correspondant au nombre de SM actifs dans le demi-bras.

La Figure 5.21.a illustre l'engin de calcul issu de cette analyse. On retrouve comme précédemment des fichiers de registres permettant d'emmagasiner les gâchettes (fournies par le contrôleur) et les valeurs des $v_{C_{ij}}(t)$. Ces fichiers de registres permettre la lecture et l'écriture simultanées de cinq valeurs en entrées, et cinq valeurs en sortie afin d'assurer un débit suffisamment élevé. Une unité de contrôle sert à cadencer les séquences opératives tandis que trois unités arithmétique sont utilisées pour calculer respectivement R_{arm_j}, $v^h_{arm_j}(t)$ et les va-

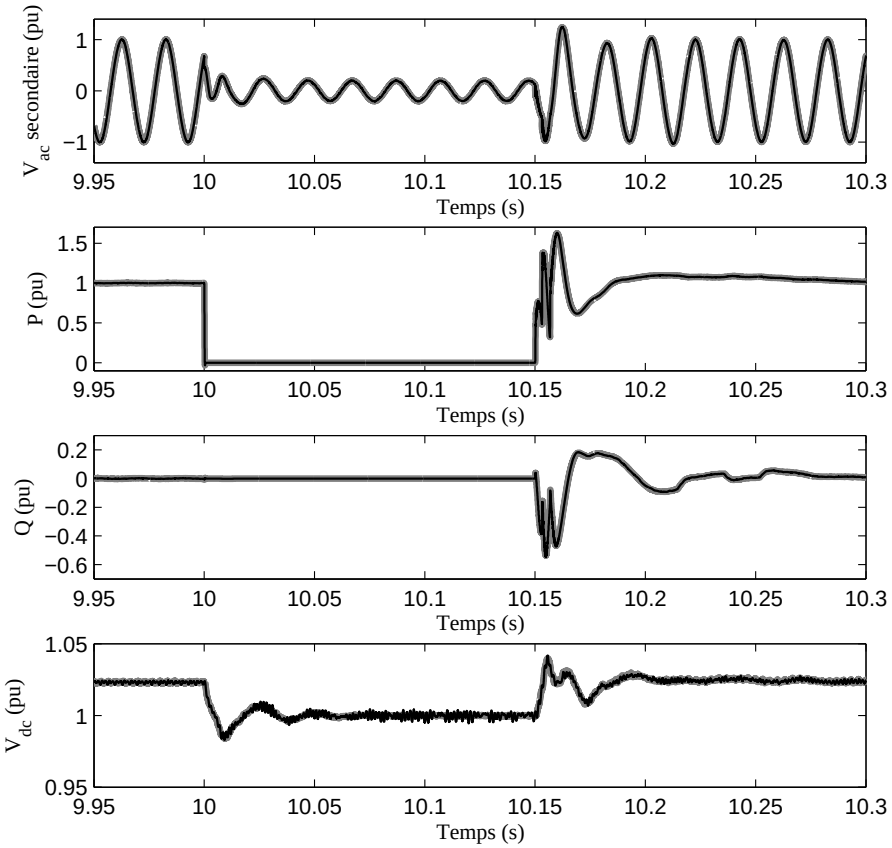

Figure 5.22 – Résultats de simulation du MMC.

leurs des $v_{C_{ij}}(t+\Delta t)$. La façon dont ces unités opératives sont implémentées est illustrée à la Figure 5.21.b. L'unité permettant de calculer R_{arm_j} est une LUT adressée indirectement par les gâchettes ; l'unité en charge de calculer $v^h_{arm_j}(t)$ est un OPS5 ; finalement, l'unité servant au calcul de $v_{C_{ij}}(t+\Delta t)$ utilise des 5 OPS2 (sans fonction d'accumulation) — ses sorties sont redirigées vers les fichiers de registres. Contrairement à ce que nous faisions précédemment, les constantes sont maintenant représentées avec une mantisse signée de 44 bits, tandis que les variables exploitent toujours des mantisses de 35 bits. Cette solution permet d'employer quatre blocs DSP pour effectuer chaque multiplication. Les additions en FAA emploient une mantisse interne de 80 bits. Ces précautions arithmétiques servent à garantir la meilleure exactitude des calculs.

Le Tableau 5.7 présente les résultats d'implémentation de l'engin de calcul sur le Virtex 6 XC6LX240T-1C [122] après placement et routage. On relèvera que les résultats de consom-

Tableau 5.7 – Résultats d'implémentation du MMC.

Métriques	Virtex 6	disponibles
Tranches	14,119 (37 %)	37,680
Blocs DSP	60 (7 %)	768
BRAM	92 (22 %)	416
Chemin critique	4.991 ns	N/A
Fréquence maximale	200.40 MHz	N/A

mation de mémoire sont de beaucoup inférieurs à ce que la taille du circuit pouvait faire croire (nous avons près de 2400 états, et le double de paires IGBT/diode). Cette économie dans la consommation de mémoire est attribuable aux simplifications des équations et aux découplages des demi-bras du restant du réseau électrique, ce qui démontre l'intérêt de ces approches dans la simulation en temps réel. On notera que la consommation des blocs DSP concorde avec la théorie (4 blocs DSP par multiplieur, pour un total de $5 + 5 \times 2 = 15$ multiplieurs, soit $4 \times 15 = 60$ blocs DSP). La consommation des tranches du FPGA (37 %) est relativement importante, mais raisonnable étant donnée la dimension du problème simulé.

Pour conclure ce cas d'étude, la Figure 5.22 présente certaines variables (en p.u.) du réseau lorsqu'une défaut de type "trois phases à la terre" est appliqué à la 10^e seconde du temps de la simulation. Une superposition du résultat de la simulation en temps réel et d'une simulation en temps différée utilisant SPS est effectuée afin d'illustrer la bonne allure générale des courbes. Les erreurs relatives pour ces courbes varient entre 0.1 % et 3 %, ce qui est très acceptable pour une simulation en temps réel.

5.6 Conclusion

Ce dernier chapitre a permis de regrouper les développements des chapitres précédents et d'en faire la synthèse dans la mise en œuvre d'engins de calcul pour la simulation des circuits d'électronique de puissance. Nous avons pu établir les critères de conception servant à la réalisation de tels engins de calcul (format en VF non standard, accumulation en FAA) et démontré le bon fonctionnement de ceux-ci. Un engin de calcul versatile a été présenté et étudié selon différents critères (utilisation des ressources FPGA, pas de calcul, contexte d'application réaliste, haute fréquence de commutation). Nous avons également considéré des études de cas où le modèle résistif de l'interrupteur a été utilisé. Ces cas d'étude servaient à démontrer que si le modèle résistif souffre de certaines limitations quant à son caractère général du fait de sa gourmandise en mémoire, des engins de calcul spécialisés peuvent être considérés pour des applications de haute performance et demandant une assez bonne

précision.

Ainsi, nous avons pu réunir dans ce chapitre l'ensemble des résultats de nos travaux de recherche, les faisant aboutir à ce qui peut être décrit comme la pointe de l'état de l'art, et ce tout en proposant des prototype de grade industriel. L'eHS et le pont à deux niveaux sont deux cas d'espèce, et ils sont aujourd'hui commercialisés par le partenaire industriel de la thèse, la compagnie Opal-RT Technologies.

CONCLUSION

Au terme de cette thèse, il convient de refaire le sommaire des contributions qui sont les nôtres et de suggérer des voies de recherche pour l'avenir. Ainsi, le travail présenté dans cette thèse a apporté différentes contributions scientifiques au domaine de la simulation en temps réel sur FPGA et, à la base de ces apports, des contributions à l'arithmétique des ordinateurs.

Nous avons ainsi formulé un algorithme de sommation qui est une généralisation de la TAA, nantie d'une formulation et d'une réalisation matérielle simplifiées. Nous avons démontré l'intérêt d'un tel algorithme dans la construction d'opérateurs tels que le MAC et l'OPS, et l'intérêt de pouvoir disposer d'un système redondant pour la sommation rapide de mantisses larges. Le travail a par ailleurs établi les critères permettant de garantir la bonne exactitude des résultats de la sommation, critères que nous avons établis par des démonstrations théoriques et empiriques.

En ce qui a trait au système redondant, la thèse a proposé une analyse exhaustive de l'utilisation du format HRCS dans l'addition de mantisses larges, format pour lequel deux nouveaux opérateurs arithmétiques ont été proposés : un additionneur endomorphique ainsi qu'un convertisseur HRCS à format conventionnel. Ces développements ont été cruciaux pour le développement d'arbres d'additionneurs en virgule flottante à faibles latences ainsi qu'à la fonction d'accumulation à un cycle d'horloge.

Fort de ces résultats, notre travail s'est tourné vers l'implémentation d'engins de calcul faits de plusieurs MAC ou OPS en virgule flottante ayant une très courte latence et permettant de ce fait l'atteinte de pas de calcul de quelques centaines de nanosecondes dans la simulation de convertisseurs de puissance de moyenne complexité. En proposant une analyse de la modélisation de circuits d'électronique de puissance, nous avons pu proposer un engin de calcul versatile permettant de simuler des convertisseurs de topologie arbitraire, et pouvant contenir jusqu'à 24 interrupteurs avec des pas de temps en deçà de la microseconde, tout en admettant des fréquences de commutations de plusieurs kilohertz. Nous avons également considéré des cas pratiques où le modèle résistif de l'interrupteur a été utilisé. Les cas d'études que nous avons considérés sont tirés de contextes industriels où des problématiques modernes étaient à la recherche de solutions qu'ont pu apporter nos travaux. Ces réalisations placent nos travaux à la pointe de l'état de l'art du domaine de la simulation en temps réel sur FPGA.

La thèse a par ailleurs permis de mettre la lumière sur certains faits et d'éclaircir des voies de recherche à venir. Ainsi, il est établi que la simulation des circuits d'électronique

de puissance est réalisable sur FPGA avec des pas de temps qui n'excèdent pas 1 μs. Deux critères fondent cette conclusion : la puissance de calcul sur FPGA et la mémoire embarquée disponible. Pour ce qui est de la puissance de calcul, nous avons pu constater qu'il était possible d'utiliser moins de 10 % d'un FPGA tel que le Virtex 6 et de disposer de plus de 6 GFLOPS de puissance de calcul. Ainsi, des engins de calcul plus puissants que l'eHS (un engin de calcul nanti de 8 OPS8 offrirait plus de 25 GFLOPS) permettrait d'effectuer davantage de calculs dans la même fenêtre de temps. Les résultats du Chapitre 2 ont par ailleurs permis de démontrer qu'un tel engin de calcul serait efficace dans l'exécution de cette tâche.

Cette marge de manœuvre est certainement sujette à amélioration dans un futur proche, étant donnée la croissance impressionnante de la densité des FPGA durant les dernières années. Un tel fait permet d'entrevoir la possibilité d'implémenter des algorithmes itératifs sur FPGA sans trop compromettre le pas de temps de la simulation. Néanmoins, pour ce faire, il convient de réduire les besoins en mémoire embarquée de manière significative. La voie royale pour cela est la formulation d'une méthode de découplage des circuits électriques, permettant par exemple de considérer des topologies arbitraires de convertisseurs de puissance, tout en utilisant le modèle résistif de l'interrupteur. Une telle méthode se doit de ne pas être trop gourmande en calculs et en mémoire, elle doit garantir la possibilité de s'exécuter efficacement sur FPGA (régularité des calculs) et de fournir une solution générale à une large gamme de problèmes. Pour ce faire, la clé du succès est probablement du côté de l'implémentation d'opérateurs de division en virgule flottante, et sur cette base d'engins de calcul servant à résoudre sur puce des systèmes d'équations linéaires.

Une autre question qui mérite réflexion est le constat que nous avons fait des limites imposées par les liens de communication entre FPGA et CPU. Les protocole de communications dits rapides visent principalement un haut débit de communication, atteignant dans le cas du PCIe plusieurs dizaines de Gbps. Or la simulation en temps réel (surtout la simulation HIL) nécessite une communication à très courte latence. Ce requis est d'autant plus important que les techniques de découplage (tant inter-CPU que CPU-FPGA) gagneraient en précision si elles pouvaient être effectuées de manière itérative. Autrement dit, une voie de recherche qui mérite d'être considérée est l'étude de protocoles de communication à faible latence (par exemple par l'implémentation de protocoles non standards). Un autre sujet d'intérêt est une refonte du paradigme de communication des simulateurs multi-CPU (et des simulateurs multi-FPGA que l'on voit se profiler dans un avenir proche) afin d'intégrer l'idée de communications inter-pas, seules garantes de simulations réparties utilisant des méthodes itératives.

PUBLICATIONS DE L'AUTEUR

Chapitre de livre

- DUFOUR, C., OULD BACHIR, T., GRÉGOIRE, L.-A., and BÉLANGER, J. (2012). Real-time simulation of power electronic systems and devices. *In Vasca, F. et Iannelli, L., editors : Dynamics and Control of Switched Electronic Systems*, Advances in Industrial Control, pages 451–487, Springer London.

Articles de revue

- SAAD, H., OULD BACHIR, T., MAHSEREDJIAN, J., DUFOUR, C., DENNETIÈRE, S., and NGUEFEU, S. (2014). Real-time simulation of MMCs Using CPU and FPGA. *IEEE Transactions on Power Electronics — accepté.*

- OULD BACHIR, T., and DAVID, J.-P. (2013). Self-alignment schemes for the implementation of addition-related floating-point operators. *ACM Transactions on Reconfigurable Technology Systems* (TRETS), vol. 6, issue 1, pages 1–21.

- OULD BACHIR, T., DUFOUR, C., BÉLANGER, J., MAHSEREDJIAN, J., and DAVID, J.-P. (2013). A fully automated reconfigurable calculation engine dedicated to the real-time simulation of high switching frequency power electronic circuits. *Mathematics and Computers in Simulation.* vol 91, pages 167–177.

- F.-BLANCHETTE, H., OULD BACHIR, T., and DAVID, J.-P. (2012). A state-space modeling approach for the FPGA-based real-time simulation of power converters. *IEEE Transactions on Industrial Electronics*, vol. 59, issue 12, pages 4555–4567.

Articles de conférence

- DUFOUR, C., CENSE, S., OULD BACHIR, T., GRÉGOIRE, L.-A., and BÉLANGER, J. (2012). General-purpose reconfigurable low-latency electric circuit and drive solver on FPGA. *In Annual Conference of IEEE Industrial Electronics Society* (IECON), pages 3073–3081, Montréal, Canada.

– OULD BACHIR, T., DUFOUR, C., BÉLANGER, J., MAHSEREDJIAN, J., and DAVID, J.-P. (2012). Effective floating-point calculation engines intended for the FPGA-based HIL simulation. *In International Symposium on Industrial Electronics* (ISIE), pages 1363–1368, Hangzhou, China.

– INABA, Y., CENSE, S., OULD BACHIR, T., YAMASHITA, H., and DUFOUR, C. (2011). A dual high-speed PMSM motor drive emulator with finite element analysis on FPGA chip with full fault testing capability. *In European Conference on Power Electronics and Applications* (EPE), pages 1–10, Nottingham, UK.

– OULD BACHIR, T., DUFOUR, C., DAVID, J.-P., and MAHSEREDJIAN, J. (2011). Floating-point engines for the FPGA-based real-time simulation of power electronic circuits. *In International Conference on Power Systems Transients* (IPST), Delft, The Netherlands.

– OULD BACHIR, T., DUFOUR, C., DAVID, J.-P., BÉLANGER, J. et MAHSEREDJIAN, J. (2011). Reconfigurable floating-point engines for the real-time simulation of PECs : a high-speed PMSM drive case study. *In Electrimacs*, Cergy-Pontoise, France.

– OULD BACHIR, T., DUFOUR, C., DAVID, J.-P., and BÉLANGER, J. (2010). Effective FPGA-based electric motor modeling with floating-point cores. *In Annual Conference of IEEE Industrial Electronics Society* (IECON), pages 829–834, Arizona, USA.

– OULD BACHIR, T., and DAVID, J.-P. (2010). FPGA-based real-time simulation of state-space models using floating-point cores. *In International Power Electronics and Motion Control Conference* (EPE/PEMC), pages S2 26–31, Ohrid, Macedonia.

– OULD BACHIR, T., and DAVID, J.-P. (2010). Performing floating-point accumulation on a modern FPGA in single and double precision. *In International Symposium on Field-Programmable Custom Computing Machines* (FCCM), pages 105–108, North Carolina, USA.

BIBLIOGRAPHIE

[1] Anane, M., Bessalah, H., Issad, M., Anane, N. et Salhi, H. (2008). Higher radix and redundancy factor for floating point srt division. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(6):774 –779.

[2] Apopei, B., Mills, A., Dodd, T. et Thompson, H. (2009). Real time simulation in floating point precision using FPGA computing. *In Reconfigurable Computing : Architectures, Tools and Applications*, Lecture Notes in Computer Science, chapitre 40, pages 349–354. Springer-Verlag, Berlin Heidelberg.

[3] Avizienis, A. (1961). Signed-digit number representations for fast parallel arithmetic. *IEEE Transactions on Electronic Computers*, 10(3):389–400.

[4] Banescu, S., de Dinechin, F., Pasca, B. et Tudoran, R. (2010). Multipliers for floating-point double precision and beyond on FPGAs. *In Highly-Efficient Accelerators and Reconfigurable Technologies*.

[5] Baumhof, C. (1995). A new VLSI vector arithmetic coprocessor for the PC. *In Proceedings of the 12th Symposium on Computer Arithmetic*, pages 210–215.

[6] Beguenane, R., Beuchat, J.-L., Muller, J.-M. et Simard, S. (2005). Modular multiplication of large integers on FPGA. *In Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pages 1361–1365.

[7] Beuchat, J.-L. et Muller, J.-M. (2008). Automatic generation of modular multipliers for FPGA applications. *IEEE Transactions on Computers*, 57(12):1600–1613.

[8] Booth, A. (1951). A Signed Binary Multiplication Technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240.

[9] Catanzaro, B. et Nelson, B. (2005). Higher radix floating-point representations for FPGA-based arithmetic. *In IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 161–170.

[10] Chen, Y. et Dinavahi, V. (2009). FPGA-based real-time EMTP. *IEEE Transactions on Power Delivery*, 24(2):892–902.

[11] Chua, L. et Chen, L.-K. (1976). Diakoptic and generalized hybrid analysis. *IEEE Transactions on Circuits and Systems*, 23(12):694–705.

[12] Cosoroaba, A. et Rivoallon, F. (2006). WP245 v1.1.1 : Achieving higher system performance with the Virtex-5 family of FPGAs. White Paper.

[13] Dagbagi, M., Idkhajine, L., Monmasson, E. et Slama-Belkhodja, I. (2012). FPGA implementation of power electronic converter real-time model. *In International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 658–663.

[14] de Dinechin, F., Detrey, J., Trestian, I., Creţ, O. et Tudoran, R. (2007). When FPGAs are better at floating-point than microprocessors. Rapport technique ensl-00174627, ÉNS Lyon. http ://prunel.ccsd.cnrs.fr/ensl-00174627.

[15] de Dinechin, F., Klein, C. et Pasca, B. (2009). Generating high-performance custom floating-point pipelines. *In Field Programmable Logic and Applications*. IEEE.

[16] de Dinechin, F., Nguyen, H. et Pasca, B. (2010). Pipelined FPGA adders. *In International Conference on Field Programmable Logic and Applications (FPL)*, pages 422–427.

[17] de Dinechin, F. et Pasca, B. (2009). Large multipliers with fewer DSP blocks. *In International Conference on Field Programmable Logic and Applications (FPL)*, pages 250–255.

[18] de Dinechin, F., Pasca, B., Creţ, O. et Tudoran, R. (2008). An FPGA-specific approach to floating-point accumulation and sum-of-products. *In International Conference on ICECE Technology (FPT)*, pages 33–40. IEEE.

[19] Detrey, J. et de Dinechin, F. (2007). A tool for unbiased comparison between logarithmic and floating-point arithmetic. *The Journal of VLSI Signal Processing*, 49(1): 161–175.

[20] Dommel, H. (1969). Digital computer solution of electromagnetic transients in single- and multiphase networks. *IEEE Transactions on Power Apparatus and Systems*, 88(4): 388–399.

[21] Dommel, H. (1992). *EMTP Theory Book, 2nd Ed.* MicroTran Power Analysis Corporation.

[22] Dufour, C. (2000). *Deux contributions à la problématique de la simulation numérique en temps réel des reseaux de transport d'énergie*. Thèse de doctorat, Universite Laval.

[23] Dufour, C. et Belanger, J. (2001). Discrete time compensation of switching events for accurate real-time simulation of power systems. *In The 27th Annual Conference of the IEEE Industrial Electronics Society (IECON '01)*, volume 2, pages 1533–1538.

[24] Dufour, C., Blanchette, H. et Belanger, J. (2008). Very-high speed control of an FPGA-based finite-element-analysis permanent magnet synchronous virtual motor drive system. *In 34th Annual Conference of IEEE Industrial Electronics Society (IECON 2008)*, pages 2411–2416.

[25] Dufour, C., Cense, S., Ould Bachir, T., Grégoire, L.-A. et Bélanger, J. (2012a). General-purpose reconfigurable low-latency electric circuit and drive solver on FPGA. *In Annual Conference of IEEE Industrial Electronics Society (IECON)*, Québec, Canada.

[26] Dufour, C., Mahseredjian, J. et Bélanger, J. (2011). A combined state-space nodal method for the simulation of power system transients. *IEEE Transactions on Power Delivery*, 26(2):928–935.

[27] Dufour, C., Ould Bachir, T., Grégoire, L.-A. et Bélanger, J. (2012b). Real-time simulation of power electronic systems and devices. *In* Vasca, F. et Iannelli, L., éditeurs : *Dynamics and Control of Switched Electronic Systems*, Advances in Industrial Control, pages 451–487. Springer London.

[28] Duman, E., Can, H. et Akin, E. (2007). Real time FPGA implementation of induction machine model - a novel approach. *In International Aegean Conference on Electrical Machines and Power Electronics.*, pages 603–606.

[29] Ercegovac, M. et Lang, T. (2004). *Digital Arithmetic*. Morgan Kaufmann.

[30] F.-Blanchette, H., Ould Bachir, T. et David, J.-P. (2012). A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters. *IEEE Transactions on Industrial Electronics*, 59(12):4555–4567.

[31] Fahmy, H. et Flynn, M. (2003). The case for a redundant format in floating point arithmetic. *In Proceedings of the IEEE Symposium on Computer Arithmetic*, pages 95–102.

[32] Fahmy, H., Liddicoat, A. et Flynn, M. (2001). Improving the effectiveness of floating point arithmetic. *In Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers.*, volume 1, pages 875–879.

[33] Farmwald, P. M. (1981). *On the design of high performance digital arithmetic units*. Thèse de doctorat, Stanford University.

[34] Gnanarathna, U., Gole, A. et Jayasinghe, R. (2011). Efficient modeling of modular multilevel HVDC converters (MMC) on electromagnetic transient simulation programs. *IEEE Transactions on Power Delivery*, 26(1):316–324.

[35] Gole, A., Keri, A., Kwankpa, C., Gunther, E., Dommel, H., Hassan, I., Marti, J., Martinez, J., Fehrle, K., Tang, L., McGranaghan, M., Nayak, O., Ribeiro, P., Iravani, R. et Lasseter, R. (1997). Guidelines for modeling power electronics in electric power engineering applications. *IEEE Transactions on Power Delivery*, 12(1):505–514.

[36] Grama, A., Karypis, G., Kumar, V. et Gupta, A. (2003). *Introduction to Parallel Computing (2nd Edition)*. Addison Wesley, 2 édition.

[37] Hachtel, G., Brayton, R. et Gustavson, F. (1971). The sparse tableau approach to network analysis and design. *IEEE Transactions on Circuit Theory*, 18(1):101 – 113.

[38] Hartmann, M., Round, S., Ertl, H. et Kolar, J. (2009). Digital current controller for a 1 Mhz, 10 kW three-phase VIENNA rectifier. *IEEE Transactions on Power Electronics*, 24(11):2496 –2508.

[39] Higham, N. J. (1993). The accuracy of floating point summation. *SIAM J. Sci. Comput*, 14:783–799.

[40] Hinrichsen, D. et Pritchard, A. J. (2005). *Mathematical Systems Theory I : Modelling, State Space Analysis, Stability and Robustness*, volume 48. Springer-Verlag, Berlin.

[41] Ho, C.-W., Ruehli, A. et Brennan, P. (1975). The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, 22(6):504–509.

[42] Hollman, J. et Marti, J. (2003). Real time network simulation with PC-cluster. *IEEE Transactions on Power Systems*, 18(2):563 – 569.

[43] Huang, M. et Andrews, D. (2012). Modular design of fully pipelined reduction circuits on FPGAs. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):1–10.

[44] Huang, M. et Kilic, O. (2010). Reaping the processing potential of FPGA on double-precision floating-point operations : An eigenvalue solver case study. *In IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 95–102.

[45] Hui, S. et Christopoulos, C. (1990). A discrete approach to the modeling of power electronic switching networks. *IEEE Transactions on Power Electronics*, 5(4):398–403.

[46] Hui, S. et Fung, K. (1997). Fast decoupled simulation of large power electronic systems using new two-port companion link models. *IEEE Transactions on Power Electronics*, 12(3):462–473.

[47] Hui, S. et Morrall, S. (1994). Generalised associated discrete circuit model for switching devices. *IEE Proceedings – Science, Measurement and Technology*, 141(1):57–64.

[48] IEEE (2008). IEEE-754, Standard for floating-point arithmetic.

[49] Inaba, Y., Cense, S., Ould Bachir, T., Yamashita, H. et Dufour, C. (2011). A dual high-speed PMSM motor drive emulator with finite element analysis on FPGA chip with full fault testing capability. *In European Conference on Power Electronics and Applications (EPE)*, pages 1–10, Nottingham, UK.

[50] Jaberipur, G. et Gorgin, S. (2010). An improved maximally redundant signed digit adder. *Computers and Electrical Engineering*, 36(3):491–502.

[51] Jaberipur, G., Parhami, B. et Gorgin, S. (2010). Redundant-digit floating-point addition scheme based on a stored rounding value. *IEEE Transactions on Computers*, 59(5):694–706.

[52] Jastrzebski, R., Laakkonen, O., Rauma, K., Luukko, J., Saren, H. et Pyrhönen, O. (2004). Real-time emulation of induction motor in fpga using floating point representation. *In IASTED International Conference*, Applied Simulation and Modelling, pages 226–231.

[53] Jie, S., Ning, Y. et Xiao-Yan, Z. (2008). An ieee compliant floating-point adder with the deeply pipelining paradigm on fpgas. *In International Conference on Computer Science and Software Engineering*, volume 4, pages 50–53.

[54] Jin, H. (1997). Behavior-mode simulation of power electronic circuits. *IEEE Transactions on Power Electronics*, 12(3):443–452.

[55] Kahan, W. (1971). A survey of error analysis. *In IFIP Congress*, pages 1214–1239.

[56] Kapre, N. et DeHon, A. (2007). Optimistic parallelization of floating-point accumulation. *In IEEE Symposium on Computer Arithmetic, 2007 (ARITH)*, pages 205–216.

[57] Karlstrom, P., Ehliar, A. et Liu, D. (2008). High-performance, low-latency field-programmable gate array-based floating-point adder and multiplier units in a virtex 4. *IET Computers Digital Techniques*, 2(4):305–313.

[58] Kato, T., Inoue, K., Fukutani, T. et Kanda, Y. (2009). Multirate analysis method for a power electronic system by circuit partitioning. *IEEE Transactions on Power Electronics*, 24(12):2791–2802.

[59] Kawakami, K., Shigemoto, K. et Nakano, K. (2008). Redundant radix-2^r number system for accelerating arithmetic operations on the FPGAs. *In International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 370–377.

[60] Kilts, S. (2007). *Advanced FPGA Design : Architecture, Implementation, and Optimization*. Wiley-IEEE Press, New York.

[61] Kim, D. et Kim, L.-S. (2009). A floating-point unit for 4D vector inner product with reduced latency. *IEEE Transactions on Computers*, 58(7):890–901.

[62] Kulisch, U. (2002). *Advanced Arithmetic for Digital Computer : Design of Arithmetic units*. Springer-Verlag.

[63] Kwon, T. J. et Draper, J. (2008). Floating-point division and square root implementation using a taylor-series expansion algorithm with reduced look-up tables. *In Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 954–957.

[64] Langhammer, M. (2008). High performance matrix multiply using fused datapath operators. *In Asilomar Conference on Signals, Systems and Computers*, pages 153–159.

[65] Langhammer, M. et VanCourt, T. (2009). FPGA floating point datapath compiler. *In Symposium on Field Programmable Custom Computing Machines (FCCM)*, pages 259–262.

[66] Le-Huy, P., Guerette, S., Dessaint, L. A. et Le-Huy, H. (2006). Real-time simulation of power electronics in power systems using an fpga. *In Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 873–877.

[67] Liddicoat, A. et Flynn, M. (2001). High-performance floating point divide. *In Proceedings. Euromicro Symposium on Digital Systems, Design, 2001.*, pages 354 –361.

[68] Lu, B., Wu, X., Figueroa, H. et Monti, A. (2007). A low-cost real-time hardware-in-the-loop testing approach of power electronics controls. *IEEE Transactions on Industrial Electronics*, 54(2):919–931.

[69] Luo, Z. et Martonosi, M. (2000). Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques. *IEEE Transactions on Computers*, 49(3):208–218.

[70] Maguire, T. et Giesbrecht, J. (2005). Small time-step (<2us) VSC model for the real time digital simulator. *In International Conference on Power Systems Transients (IPST)*.

[71] Mahseredjian, J. et Alvarado, F. (1997). Creating an electromagnetic transients program in matlab : Matemtp. *IEEE Transactions on Power Delivery*, 12(1):380–388.

[72] Mahseredjian, J., DennetiËre, S., DubÈ, L., Khodabakhchian, B. et GÈrin-Lajoie, L. (2007). On a new approach for the simulation of transients in power systems. *Electric Power Systems Research*, 77(11):1514–1520. Selected Topics in Power System Transients - Part II.

[73] Mahseredjian, J., Dinavahi, V. et Martinez, J. (2009). Simulation tools for electromagnetic transients in power systems : Overview and challenges. *IEEE Transactions on Power Delivery*, 24(3):1657–1669.

[74] Majstorovic, D., Celanovic, I., Teslic, N. D., Celanovic, N. et Katic, V. A. (2011). Ultralow-latency hardware-in-the-loop platform for rapid validation of power electronics designs. *IEEE Transactions on Industrial Electronics*, 58(10):4708–4716.

[75] Matar, M. et Iravani, R. (2010). FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients. *IEEE Transactions on Power Delivery*, 25(2):852–860.

[76] Morris, G. et Prasanna, V. (2005). An FPGA-based floating-point Jacobi iterative solver. *In Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*.

[77] Muller, J.-M. (2005). On the definition of ulp(x). Rapport technique RR-5504, INRIA.

[78] Muller, J.-M., Brisebarre, N., de Dinechin, F., Jeannerod, C.-P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D. et Torres, S. (2010). *Handbook of Floating-Point Arithmetic*. Birkhauser Boston.

[79] Nagendra, C., Owens, R. et Irwin, M. (1995). Unifying carry-sum and signed-digital number representations for low power. *In Proceedings of the international symposium on Low power design*, pages 15–20, New York, NY, USA. ACM.

[80] Ogita, T., Rump, S. M. et Oishi, S. (2005). Accurate sum and dot product. *SIAM J. Sci. Comput*, 26:2005.

[81] Ould Bachir, T. et David, J.-P. (2010a). FPGA-based real-time simulation of state-space models using floating-point cores. *In International Power Electronics and Motion Control Conference (EPE/PEMC)*, pages S2 26–31, Ohrid, Macedonia.

[82] Ould Bachir, T. et David, J.-P. (2010b). Performing floating-point accumulation on a modern FPGA in single and double precision. *In International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 105–108, North Carolina, USA.

[83] Ould Bachir, T. et David, J.-P. (2012). Self-alignment schemes for the implemention of addition-related floating-point operators. *ACM Trans. Reconfigurable Technol. Syst.*, 6(1):1–21.

[84] Ould-Bachir, T., Dufour, C., Bélanger, J., Mahseredjian, J. et David, J.-P. (2012). Effective floating-point calculation engines intended for the FPGA-based HIL simulation. *In International Symposium on Industrial Electronics (ISIE)*, pages 1363–1368, Hangzhou, China.

[85] Ould Bachir, T., Dufour, C., Bélanger, J., Mahseredjian, J. et David, J.-P. (2013). A fully automated reconfigurable calculation engine dedicated to the real-time simulation of high switching frequency power electronic circuits. *Mathematics and Computers in Simulation*, 91:167–177. ELECTRIMACS 2011 - PART II.

[86] Ould Bachir, T., Dufour, C., David, J.-P. et Bélanger, J. (2010). Effective FPGA-based electric motor modeling with floating-point cores. *In Annual Conference of IEEE Industrial Electronics Society (IECON)*, pages 829–834, Arizona, USA.

[87] Ould Bachir, T., Dufour, C., David, J.-P., Bélanger, J. et Mahseredjian, J. (2011a). Reconfigurable floating-point engines for the real-time simulation of PECs : a high-speed PMSM drive case study. *In Electrimacs*, Cergy-Pontoise, France.

[88] Ould Bachir, T., Dufour, C., David, J.-P. et Mahseredjian, J. (2011b). Floating-point engines for the FPGA-based real-time simulation of power electronic circuits. *In International Conference on Power Systems Transients (IPST)*, Delft, The Netherlands.

[89] Paidimarri, A., Cevrero, A., Brisk, P. et Ienne, P. (2009). FPGA implementation of a single-precision floating-point multiply-accumulator with single-cycle accumulation. *In IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*, pages 267–270.

[90] Pak, L.-F., Faruque, M., Nie, X. et Dinavahi, V. (2006). A versatile cluster-based real-time digital simulator for power engineering research. *IEEE Transactions on Power Systems*, 21(2):455–465.

[91] Paquin, J.-N., Li, W., Belanger, J., Schoen, L., Peres, I., Olariu, C. et Kohmann, H. (2009). A modern and open real-time digital simulator of all-electric ships with a multi-platform co-simulation approach. *In Electric Ship Technologies Symposium (IEEE/ESTS)*, pages 28–35.

[92] Parhami, B. (1990). Generalized signed-digit number systems : a unifying framework for redundant number representations. *IEEE Transactions on Computers*, 39(1):89–98.

[93] Parhami, B. (2000). *Computer arithmetic : algorithms and hardware designs*. Oxford University Press, Oxford, UK.

[94] Pejovic, P. et Maksimovic, D. (1994). A method for fast time-domain simulation of networks with switches. *IEEE Transactions on Power Electronics*, 9(4):449–456.

[95] Peralta, J., Saad, H., Dennetiere, S., Mahseredjian, J. et Nguefeu, S. (2012a). Detailed and averaged models for a 401-level MMC-HVDC system. *IEEE Transactions on Power Delivery*, 27(3):1501–1508.

[96] Peralta, J., Saad, H., Dennetiere, S., Mahseredjian, J. et Nguefeu, S. (2012b). Detailed and averaged models for a 401-level MMC-HVDC system. *IEEE Transactions on Power Delivery*, 27(3):1501–1508.

[97] Pimentel, J. et Le-Huy, H. (2006). Hardware emulation for real-time power system simulation. *In IEEE International Symposium on Industrial Electronics*, volume 2, pages 1560–1565.

[98] Pimentel, J. C. G. (2006). Implementation of simulation algorithms in FPGA for real time simulation of electrical networks with power electronics devices. *In IEEE International Conference on Reconfigurable Computing and FPGA's*, pages 1–8.

[99] Press, W. H., Teukolsky, S. A., Vetterling, W. T. et Flannery, B. P. (2007). *Numerical Recipes 3rd Edition : The Art of Scientific Computing*. Cambridge University Press.

[100] Rakotozafy, M., Poure, P., Saadate, S., Bordas, C. et Leclere, L. (2011). Real-time digital simulation of power electronics systems with neutral point piloted multilevel inverter using FPGA. *Electric Power Systems Research*, 81(2):687–698.

[101] Rump, S. M. (1999). INTLAB - INTerval LABoratory. *In* Csendes, T., éditeur : *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht.

[102] Rump, S. M. (2009). Ultimately fast accurate summation. *SIAM Journal on Scientific Computing*, 31(5):3466–3502.

[103] Rump, S. M., Ogita, T. et Oishi, S. (2008). Accurate floating-point summation part I : Faithful rounding. *SIAM Journal on Scientific Computing*, 31(1):189–224.

[104] Saad, H., Dufour, C., Mahseredjian, J., Dennetière, S. et Nguefeu, S. (2013). Real-time simulation of MMCs using the state-space nodal approach. *In International Conference on Power Systems Transients (IPST)*, Vancouver, Canada.

[105] Saad, H., Ould Bachir, T., Mahseredjian, J., Dufour, C., Dennetière, S. et Nguefeu, S. (2014). Real-time simulation of MMCs using the CPU and FPGA. *IEEE Transactions on Power Electronics (accepté)*.

[106] Sana, A.-R., Mahseredjian, J., Dai-Do, X. et Dommel, H. (1995). Treatment of discontinuities in time-domain simulation of switched networks. *Mathematics and Computers in Simulation*, 38(4–6):377–387.

[107] Seidel, P.-M. et Even, G. (2001). On the design of fast IEEE floating-point adders. *In Proceedings of the IEEE Symposium on Computer Arithmetic, 2001*, pages 184–194.

[108] Song Sun, Zambreno, J. (2009). A floating-point accumulator for fpga-based high performance computing applications. *In International Conference on Field-Programmable Technology, 2009. FPT 2009.*, pages 493–499.

[109] Swartzlander, Jr., E. et Saleh, H. (2012). FFT implementation with fused floating-point operations. *IEEE Transactions on Computers*, 61(2):284–288.

[110] Sybille, G., Le-Huy, H., Gagnon, R. et Brunelle, P. (2007). Analysis and implementation of an interpolation algorithm for fixed time-step digital simulation of PWM converters. *In IEEE International Symposium on Industrial Electronics*, pages 793–798.

[111] Tai, Y.-G., Lo, C.-T. D. et Psarris, K. (2012). Accelerating matrix operations with improved deeply pipelined vector reduction. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):202–210.

[112] Tenca, A. (2009). Multi-operand floating-point addition. *In IEEE Symposium on Computer Arithmetic (ARITH)*, pages 161–168.

[113] Timarchi, S., Navi, K. et Kavehei, O. (2009). Maximally redundant high-radix signed-digit adder : New algorithm and implementation. *In IEEE Computer Society Annual Symposium on VLSI*, pages 97–102.

[114] Tomim, M., Marti, J. R., De Rybel, T., Wang, L. et Yao, M. (2010). Mate network tearing techniques for multiprocessor solution of large power system networks. *In IEEE Power and Energy Society General Meeting*, pages 1–6.

[115] van der Sluis, L. (2001). *Transients in Power Systems*. John Wiley and Sons.

[116] Vangal, S., Hoskote, Y., Borkar, N. et Alvandpour, A. (2006). A 6.2-GFLOPS floating-point multiply-accumulator with conditional normalization. *IEEE Journal of Solid-State Circuits*, 41(10):2314–2323.

[117] Verma, A., Verma, A., Parandeh-Afshar, H., Brisk, P. et Ienne, P. (2010). Synthesis of floating-point addition clusters on FPGAs using carry-save arithmetic. *In International Conference on Field Programmable Logic and Applications (FPL)*, pages 19–24.

[118] Xilinx (2009a). *DS099 v2.5 : Spartant-3 Family overview*.

[119] Xilinx (2009b). *DS100 v5.0 : Virtex-5 Family overview*.

[120] Xilinx (2009c). *UG193 v3.3 : Virtex-5 FPGA Xtreme DSP Design Considerations*.

[121] Xilinx (2010). *UG190 v5.3 : Virtex-5 FPGA User Guide*.

[122] Xilinx (2011). *DS150 v2.3 : Virtex-6 Family overview*.

[123] Zhang, H. et Tolbert, L. (2011). Efficiency impact of silicon carbide power electronics for modern wind turbine full scale frequency converter. *IEEE Transactions on Industrial Electronics*, 58(1):21–28.

[124] Zhuo, L., Morris, G. et Prasanna, V. (2007). High-performance reduction circuits using deeply pipelined operators on FPGAs. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1377 –1392.

[125] Zhuo, L. et Prasanna, V. K. (2008). High-performance designs for linear algebra operations on reconfigurable hardware. *IEEE Transactions on Computers*, 57(8):1057–1071.

ANNEXE A

TECHNOLOGIE FPGA

Les FPGA sont des circuits reconfigurables caractérisés par une densité de surface continuellement croissante et une puissance de calcul telle qu'elle surpasse celle des microprocesseurs dans certaines applications. On recourt souvent aux FPGA à titre d'accélérateurs matériels dans une configuration de co-processeur reprogrammable. Nous allons brosser ici un portrait rapide de cette technologie et mettre de l'avant les caractéristiques qui retiendront notre attention dans ce travail.

A.1 Marché mondial

Le marché des FPGA en est un lucratif, avec un chiffre d'affaire mondial tournant autour de 2 à 3 milliards de dollars, présentant une certaine maturité caractérisée par une croissance annuelle appréciable ($5 - 8$ %). On retrouve les FPGA dans quatre sections de marché principaux :

1. Les technologies de la communication ;
2. Calcul et traitement de données ;
3. Électronique de consommation et automobile ;
4. Divers autres secteurs industriels.

A.2 Altera vs. Xilinx

Les compagnies Xilinx et Altera dominent le marché des circuits programmables, et plus spécifiquement le marché des FPGA avec respectivement 49 % et 39 % de parts de marché [6] . L'architecture des FPGA de Altera diffère de celle de Xilinx, de sorte que les optimisations pouvant être obtenues pour l'un ne le sont pas forcément pour l'autre.

Dans le cadre de notre travail, nous favorisons les FPGA de Xilinx. Plusieurs facteurs participent à ce choix :

– Depuis plusieurs années, Opal-RT, le partenaire industriel, produit des cartes actives serties des FPGA de Xilinx (Spartan et Virtex) ;

6. Selon les données fournies par Xilinx dans son récent *Investor Factsheet*.

- Pour ces mêmes cartes, le partenaire industriel a développé du logiciel à destination des FPGA de Xilinx, tels que pilotes, librairies Matlab/Simulink, noyaux de calcul, etc.. . . ;
- Le partenaire industriel a développé une carte utilisant un Spartan 3 de Xilinx ;
- Le partenaire commercialise du logiciel permettant l'exploitation d'une carte de développement fabriquée par XiIinx et incorporant un FPGA haut de gamme de la famille Virtex 5 ;
- L'exploitation des outils logiciels du partenaire industriel facilite grandement l'intégration de notre travail dans le cadre d'un simulateur commercial ayant fait ses preuves ;
- Une grande partie de la littérature scientifique porte sur les FPGA de Xilinx, de sorte qu'il est plus facile de comparer nos résultats aux leurs.

A.3 Technologie de FPGA chez Xilinx

Xilinx possède deux gammes principales de FPGA. La gamme Spartan et la gamme Virtex. La gamme Spartan est une gamme de FPGA à bas coût (moins de 100$) avec un réseau d'interconnexion réduit et des blocs spécialisés moins performants. Les fréquences d'horloge atteignables par un Spartan 3 sont modestes (moins de 100 MHz). La gamme Virtex quand à elle est une gamme de FPGA de haute performance. Il lui est associé une gamme de prix équivalente, allant de plusieurs centaines de dollars à quelques milliers. Les FPGA Virtex sont également caractérisés par des fréquences d'opération élevées (>200 MHz) et la présence de blocs spécialisés gravés en dur, tels que processeur, bloc DSP, bloc RAM, PHY, etc. Jusqu'à une date récente, les FPGA de Xilinx étaient constitués de blocs reprogrammables à base de LUT à 4 entrées, c'est-à-dire qu'elles pouvait implémenter une fonction logique de 4 entrées au plus. Pour des fonctions avec davantage de variables, il fallait interconnecter plusieurs de ces LUT. Depuis le Virtex 4, Xilinx a introduit des LUT à six entrées, permettant ainsi l'implémentation efficace de fonctions complexes.

A.3.1 Blocs DSP

Depuis le Virtex 4, Xilinx a introduit un bloc spécialisé dans ses FPGA appelé bloc DSP. Ce bloc est gravé en dur et permet l'implémentation efficace de différentes fonctions telles que multiplication, accumulation, addition larges, etc. Le bloc DSP du Virtex 4 (que l'on retrouve également dans le Spartan 6) possède un multiplieur traditionnel 18x18 bits signés. Les FPGA Virtex 5 et Virtex 6 possèdent des multiplieurs 25x18 bits signés qui facilitent l'implémentation de la multiplication en virgule flottante. Un tel bloc est illustré à la Figure A.1.

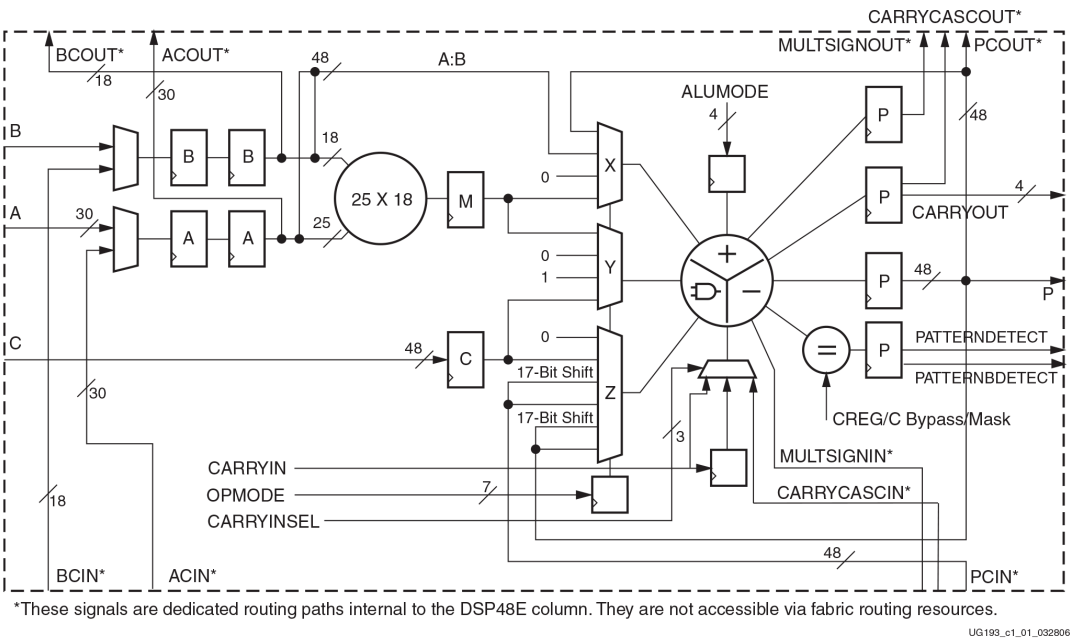

Figure A.1 – Bloc DSP du Virtex 5 [120].

A.3.2 Blocs RAM

Entre autres circuits gravés en dur, les FPGA incorporent de petites mémoires (BRAM). Ces BRAM sont configurables de différentes manières. Elles sont de taille de 18 kilobits (18 kb) ou de 36 kb. Ceci signifie que ces mémoires peuvent être adressées par dix bits d'adresse pour des largeurs de données de 18 bits et 36 bits respectivement. Bien entendu, ces mémoires peuvent être combinées pour former des mémoires de plus grande taille. Ces mémoires possèdent deux ports indépendants de lecture/écriture.

A.3.3 Circuits dédiés à la propagation de la retenue

Les FPGA modernes ont introduit une logique dédiée à la propagation de la retenue permettant d'implémenter des additionneurs de manière efficace. La figure A.2 présente une *slice* (tranche du FPGA) des FPGA Virtex 5. Les FPGA contiennent des milliers de ces blocs et la surface occupée par une architecture est souvent rapportée en nombre de *slices*, comme nous le ferons dans ce travail de thèse.

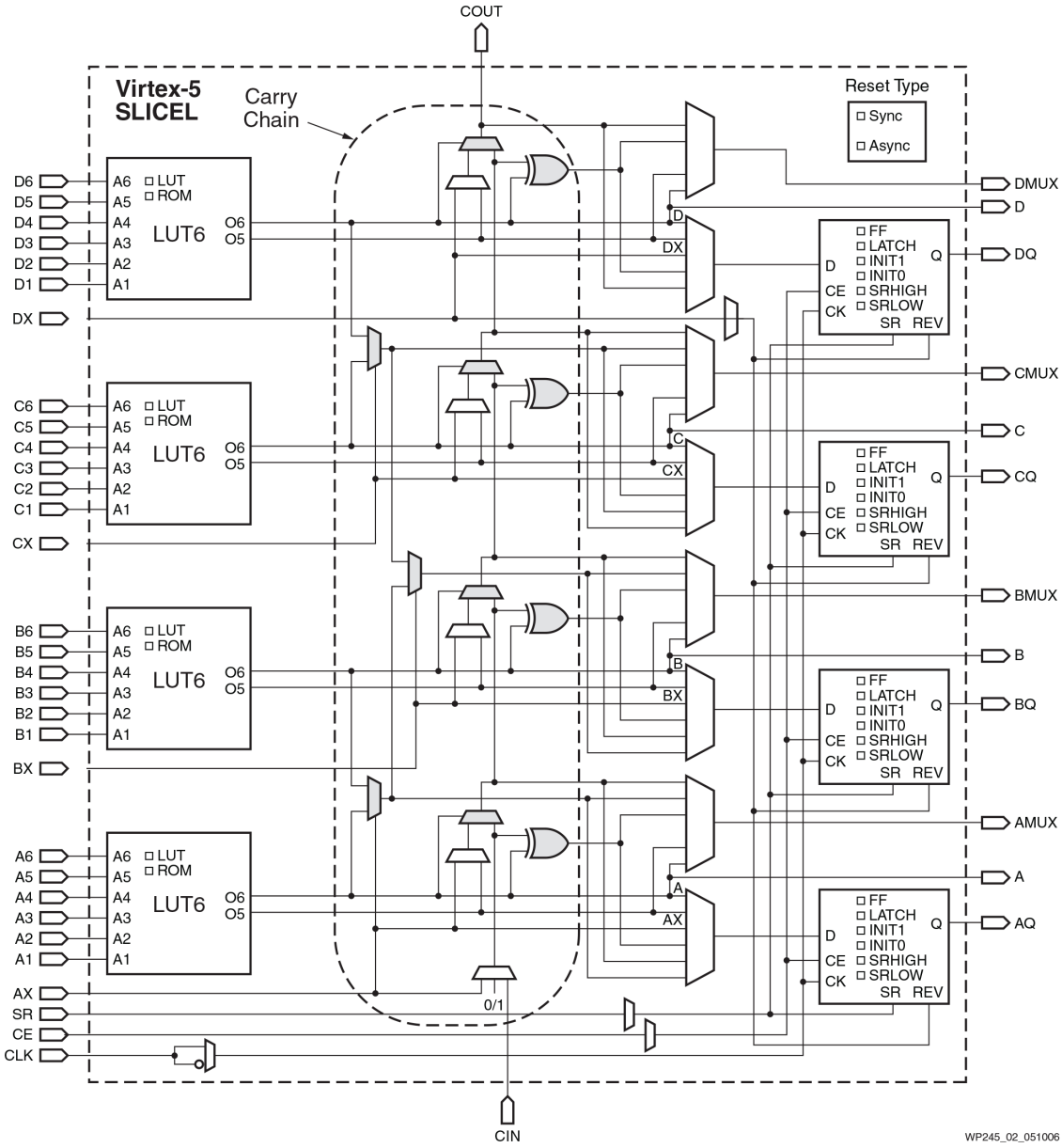

Figure A.2 – Schéma d'une *slice* du Virtex 5.

A.4 Outils de conception

Il existe différents outils pour la conception sur FPGA. Il y a d'une part les traditionnels langages de description matérielle telle que le VHDL et le Verilog. Il existe également d'autres langages plus évolués tels que le SystemC ou SystemVerilog. Un effort industriel et académique tente également d'introduire des paradigmes de développements en langage de haut niveau d'abstraction (tels que HandelC, ImpulseC). Dans le domaine du développement DSP, Xilinx et Altera ont introduit des boîte à outils Simulink pour développer du matériel depuis Matlab. Dans ce projet, nous utilisons également le VHDL et la boîte à outils System Generator de Xilinx.