Titre: Title:	Validation et caractérisation d'un décodeur LDPC matériel en sous-alimentation
Auteur: Author:	Louis-Normand Ang Houle
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ang Houle, L.-N. (2022). Validation et caractérisation d'un décodeur LDPC matériel en sous-alimentation [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10508/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10508/
Directeurs de recherche: Advisors:	François Leduc-Primeau
Programme: Program:	Génie électrique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Validation et caractérisation d'un décodeur LDPC matériel en sous-alimentation

LOUIS-NORMAND ANG HOULE
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Août 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Validation et caractérisation d'un décodeur LDPC matériel en sous-alimentation

présenté par **Louis-Normand ANG HOULE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Tarek OULD-BACHIR, président
François LEDUC-PRIMEAU, membre et directeur de recherche
Yvon SAVARIA, membre

DÉDICACE

À mon père, Luc, qui nous a quittés trop tôt.

REMERCIEMENTS

Ce projet n'aurait jamais été possible sans la sagesse, la patience et le savoir de professeur François Leduc-Primeau et de Jérémy Nadal. C'est avec la plus grande joie que je leur remets mes remerciements les plus considérés.

Je laisse une place spéciale dans mon coeur à tous mes camarades du laboratoire, Bowen, Hamed, Simon, votre intelligence est pour moi une source d'humilité. À mes charmants amis Benjamin, Géraldine, Hicham, Isabelle, Jean-Maxime, Marc(s)-Antoine(s), Xavier et j'en passe ! Vous m'avez été d'une compagnie précieuse durant ces dernières années, et j'espère pour plusieurs années encore. Enfin, à toute ma fratrie, Jacques-Arthur, Marie-Armande, Paul-Hugo, et à mes très chers parents, Émilia et Luc, merci de votre appui incessant !

RÉSUMÉ

Ce document décrit les étapes entreprises afin de valider et caractériser le fonctionnement d'un ASIC contenant le décodeur LDPC issu du projet EF-FECtive. L'ASIC dans lequel ce dernier a été implémenté appartient au noeud technologique 65nm de TSMC, et a été encapsulé dans un boîtier CPGA à 85 pattes.

Plus précisément, le but de la recherche présentée dans ce mémoire est de déterminer les paramètres optimaux d'opération du décodeur. Ces paramètres incluent la tension d'alimentation, la fréquence d'horloge de la puce, ainsi que la structure du code LDPC utilisé. L'optimisation vise le produit énergie-délai, une mesure de l'efficacité énergétique pondérée par rapport au temps de décodage. Il est présupposé que, vu la nature autocorrectrice de l'opération réalisée par le décodeur, cette dernière pourrait fonctionner en deçà de sa tension nominale ou au-delà de sa fréquence maximale, en mode quasi-synchrone.

Afin d'effectuer des tests détaillés sur cette micropuce, une carte de support capable de contrôler les paramètres visés et de mesurer la puissance consommée par le décodeur a été fabriquée. Cette carte a été conçue de façon à être compatible avec le banc de test utilisé pour la validation de l'architecture, qui en l'occurrence est composé de deux cartes de développement FPGA Xilinx communiquant par l'entremise des ports d'extension FMC présents sur chacune. Le nouveau banc de test est alors composé d'une de ces cartes de développement sur lequel est ajoutée la carte de support de l'ASIC.

Les résultats de ces tests n'ont cependant pas pu être obtenus. Les tests préliminaires, à tension et à fréquence nominale, ont révélé que l'ASIC n'est pas en mesure de compléter l'opération de décodage. L'investigation qui a suivi ces tests pointe vers un problème avec le mécanisme de réinitialisation du décodeur.

Ainsi, ce document a pour but premier de décrire le comportement actuel de l'ASIC, puis d'émettre des hypothèses tentant d'expliquer ce comportement. De plus, il servira de guide pour la réutilisation et l'amélioration de la carte mezzanine dans le futur. Les détails sur son fonctionnement, son assemblage, son utilisation et ses limitations font partie intégrante de ce mémoire.

ABSTRACT

Our team has developed a highly flexible, fast, and energy-efficient LDPC decoder architecture as a part of the EF-FECtive project, which has now been implemented in an Application Specific Integrated Circuit (ASIC) using TSMC 65nm MOSFET and a CGPA85 package. This document will discuss the testing and characterization of this ASIC.

The goal of this project is to study the operation of the decoder architecture using sub-nominal voltage. Reducing the operating voltage of the chip compromises data integrity and gate delays, but considering the self-correcting nature of the decoder's operation, power consumption reductions are achievable using this method. Thus, the work accomplished in this document aims to evaluate the performance of the decoder under different parameters in order to pinpoint its optimal operating conditions.

A carrier printed circuit board (PCB) has been designed to control the frequency and voltage of the ASIC, and to measure its energy consumption. This PCB aims to be compatible with the testbench used to validate the decoder architecture. This setup was composed of two Xilinx FPGA development boards connected through their FMC extension ports. The automated tester contained in one of the development boards is repurposed to control the ASIC with the carrier PCB.

However, preliminary testing at nominal voltage and frequency has revealed that the ASIC is unable to successfully complete a decoding routine. The results of our testing strongly suggest that there is a fault in the design of the chip. Further testing points towards a bug with the reset process of the ASIC.

In this document, the current behavior of the ASIC will be described, and an analysis of this behavior will follow in an attempt to find the fault. This document will also serve as a guide for the reuse and improvement of the mezzanine card in the future.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ASIC	Application Specific Integrated Circuit
BER	Binary Error Rate
BIST	Built-In Self Test
BPSK	Binary Phase-Shift Keying
DRC	Design Rule Check
FER	Frame Error Rate
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
ILA	Integrated Logic Analyzer
JTAG	Joint Test Action Group
LDPC	Low Density Parity Check
LUT	Lookup Table
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
OMS	Offset Min-Sum
QC	Quasi-Cyclic
SPA	Sum-Product Algorithm
TCL	Tool Command Language
VLSI	Very Large Scale Integration

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

L'objectif numéro 7 du programme pour le développement durable établi par les Nations Unies en 2015 est de garantir l'accès à des services énergétiques fiables, durables, modernes, et abordable à tous. L'atteinte de cet objectif nécessite une amélioration moyenne de 3.2% de l'efficience énergétique d'ici 2030 [1]. En contrepartie, une augmentation d'environ 30% du flux de données à travers l'Internet peut être observée chaque année [2]. Le développement de nos infrastructures de télécommunication ouvre la porte à des innovations dans de domaines tels que les véhicules autonomes ou l'Internet des objets [3]. Toutefois, la viabilité économique et environnementale de cette évolution dépend d'avancées technologiques dans l'électronique à basse puissance [4] .

1.1 Définitions et concepts de base

L'intégration a très grande échelle, soit en anglais le Very Large Scale Integration (VLSI), est une technologie qui vise à implémenter dans des circuits intégrés des systèmes de plus en plus complexes dans des surfaces de plus en plus petites. Cette technologie utilise des transistors à effet de champ, connus sous le nom de Metal Oxide Semiconductor Field Effect Transistor (MOSFET), comme unité logique de base. Afin d'accommoder la croissance exponentielle de la taille de ces circuits intégrés, les transistors qui le composent ont dû être miniaturisés. En intégrant des milliards, voire même des dizaines de milliards de ces transistors dans une même puce, cette technologie a permis une amélioration sans précédent de l'efficience énergétique, de la performance, et de l'abordabilité des dispositifs électroniques.

L'évolution qu'ont traversés les circuits intégrés depuis leur apparition a été prédite par Gordon Moore, à ce temps le fondateur de *Fairchild Semiconductor*, dans un article écrit en 1965 [5], d'où la célèbre "Loi de Moore". Le ralentissement de cette tendance autour de 2005 a donné naissance à de nouvelles prédictions pour cette industrie. Parmi les plus pertinents sont la loi de Koomey, qui porte sur l'efficience énergétique des circuits intégrés [6], et la loi de Huang, qui porte sur les processeurs graphiques [7]. La loi de Koomey met en conjonction le nombre de calculs effectué par unité d'énergie utilisée. Cette tendance, observée par Jonathan Koomey, professeur à l'université Stanford, prédit que pour une quantité d'énergie donnée, le nombre de calculs qu'un circuit intégré peut accomplir double a chaque 18 mois. La loi de Huang quant à elle compare la progression entre la densité de transistors des circuits intégrés à usage général par rapport à celle des processeurs graphiques. Jensen Huang, chef exécutif chez Nvidia, observe que la densité des processeurs graphiques augmente plus rapidement que

celle des processeurs traditionnels, d'un facteur d'environ 1,5 par an. Selon lui, une meilleure coopération entre les différents niveaux d'abstraction (système, matériel, logiciel, etc.) d'un processeur spécialisé serait à la cause de cet avantage.

Le VLSI approche toutefois un mur difficile à surmonter. La miniaturisation des transistors atteint bientôt une limite physique au-delà de laquelle il y a une détérioration de la performance plutôt qu'une amélioration [8]. De plus, l'impossibilité de réduire la tension de permutation des transistors MOSFET en deçà de sa zone de saturation a mené à un resserrement des contraintes de dissipation thermiques au point où seulement une fraction d'une puce peut être utilisée à plein régime [9].

L'implémentation d'accélérateurs matériels hautement spécialisés permet d'exploiter la croissance des circuits intégrés tout en prenant en compte les goulots d'étranglement listés ci dessus. Les accélérateurs matériels sont des circuits qui ont pour but de compléter des opérations spécifiques à une application avec des performances plus élevées qu'un processeur classique, au coût d'une flexibilité réduite. Ceux-ci permettent de réduire grandement le temps de traitement ou la consommation d'énergie d'une opération donnée. Un exemple commun d'accélérateur matériel est le processeur graphique, qui est spécialisé dans le traitement d'images. Les tâches parallélisables sont particulièrement avantagées par l'accélération matérielle. Le développement de nouveaux circuits d'accélération matérielle est inexorablement lié au futur du VLSI.

Le projet décrit dans ce document vise à développer un accélérateur matériel pour le décodage de codes Low Density Parity Check (LDPC). Le LDPC est une méthode de codage de correction d'erreurs sans voie de retour qui est utilisé dans plusieurs protocoles de télécommunication moderne, comme la 5G, le Wi-Fi (IEEE 802.11) ou le DVB-S2. Son fonctionnement sera décrit plus en profondeur dans le chapitre 2 de ce document.

En complément de l'implémentation d'accélérateurs matériels, l'ajustement dynamique de la fréquence et de la tension, le Dynamic Voltage and Frequency Scaling (DVFS), est un deuxième principe qui est appliqué aux circuits intégrés afin d'améliorer leurs performances. En ayant le contrôle sur la fréquence et sur la tension d'alimentation d'un circuit, il est possible d'optimiser un système donné en fonction de sa consommation d'énergie ou de sa fiabilité [10]. D'appliquer ce principe en combinaison avec un circuit pour le codage correcteur d'erreur est avantageux, puisque la nature autocorrectrice de l'algorithme permettrait en théorie de préserver une fiabilité du circuit acceptable tout en utilisant une puissance moindre. Il existe par contre un équilibre à maintenir entre le délai du décodeur, le niveau de bruit dans le signal décodé, et la tension d'alimentation du circuit. Une étude de ces compromis fait partie des objectifs de ce projet.

Le flot de conception d'un circuit VLSI commence par l'élaboration du concept. Dans cette étape, les paramètres et les objectifs du projet sont définis, et une méthodologie pour le mener à bien est établie. Une fois cela complété, une description comportementale sera développée de façon à synthétiser un circuit capable de satisfaire ces spécifications. Cette description permettra de générer une liste d'interconnexions (*netlist*) qui, en combinaison avec une librairie de cellules standard, servira de base pour la création d'un schéma du circuit. Ce dessin est utilisé pour la fabrication d'un circuit intégré spécialisé, un Application Specific Integrated Circuit (ASIC). Une fois le système validé, il peut être distribué au grand public sous la forme d'une puce ou de propriété intellectuelle pour être ainsi réutilisé dans un système plus complexe.

En effet, de la conceptualisation du projet jusqu'à son adoption pour un public cible, le développement d'un circuit VLSI comporte plusieurs jalons à franchir, tous nécessitant des processus de tests afin de valider que le système développé fonctionne comme prévu. Ces tests opèrent typiquement en insérant des stimuli dans le circuit à tester, puis en analysant les résultats enregistrés en sorties. L'ensemble des modules permettant ces tests est ce qui compose un banc de test. Bien sûr, afin de pouvoir confirmer que l'opération s'est déroulée comme attendu, les données qui entrent et qui sortent du circuit sous test doivent être connues. Les éléments à tenir en compte dans la création d'un banc de test seront discutés dans la deuxième section du chapitre suivant.

1.2 Éléments de la problématique

Dans un système de communication, l'encodage d'un message avec un code correcteur d'erreurs est la façon la plus efficace d'améliorer la performance d'une transmission. En théorie, un canal de communication peut acheminer une quantité d'information maximale qui est proportionnelle au bruit qui y est présent. Cette limite est nommée la limite de Shannon ou la capacité du canal. Les codes correcteurs d'erreurs modernes, tels que les codes LDPC, permettent d'approcher cette limite [11].

Le décodage LDPC optimal est un problème NP-complet [12]. Cela indique que la validation d'un résultat peut être exécutée rapidement, mais que la recherche de ce résultat prend un effort considérable. Réalistement parlant, il n'est pas utile de réaliser le décodage LDPC en faisant cette recherche exhaustive. Des algorithmes probabilistes et itératifs sont donc utilisés afin de réduire le coût de décodage. Ce processus reste toutefois la partie la plus coûteuse de la réception d'un message, d'où le besoin de développer des architectures de décodeurs plus performantes.

L'architecture proposée tente de résoudre ce problème. Celle-ci a d'abord été modélisée avec un langage de description matérielle [13]. L'implémentation physique a ensuite été réalisée par l'entremise de la fabrication d'une micropuce faite sur mesure, un ASIC, utilisant des transistors de 65nm.

Dans le cadre du projet décrit dans ce mémoire, cette puce a été intégrée à un banc de test composé d'une carte de contrôle programmable et d'une carte de support dans le but de comparer les résultats simulés à des résultats pratiques. La carte de contrôle permet de modifier les stimuli du circuit et d'enregistrer les résultats. La carte de support comprend l'ASIC, les dispositifs nécessaires pour varier la tension et la fréquence de fonctionnement de celle-ci, et des capteurs pour mesurer sa consommation énergétique. La carte de contrôle est programmée en vue de tester le circuit sous différentes conditions, en faisant varier les paramètres mentionnés ci-dessus. La conception de la carte a été faite avec la suite Cadence, qui inclut OrCAD pour la schématisation électrique et Allegro pour le routage du circuit imprimé [14].

La validation de cet accélérateur matériel ouvrirait la possibilité de voir son déploiement dans une variété d'applications. Les domaines qui pourraient bénéficier de son usage sont ceux de la télécommunication et de l'informatique. L'avantage principal de cette architecture est de diminuer le produit énergie-délai nécessaire pour le décodage d'un code LDPC, ce qui permettrait aux systèmes qui l'intègrent d'accéder à une meilleure efficience énergétique. De plus, la plateforme de test créée lors du projet pourra être réutilisée et améliorée dans le futur afin de valider d'autres ASIC développés.

1.3 Objectifs

L'objectif de la recherche est de valider et de caractériser le fonctionnement d'un ASIC contenant un décodeur LDPC. Cela présuppose l'atteinte des objectifs suivant :

- Concevoir une carte permettant d'intégrer la puce et d'automatiser les tests de performance
- Programmer des plans de tests permettant de valider la totalité des fonctionnalités de la puce
- Étudier les sources de bogues bloquant le fonctionnement normal de l'ASIC

1.4 Contribution

Le travail décrit dans ce document contribue principalement à l'infrastructure de test du projet EF-FECtive. Initialement, le décodeur a été conçu avec un langage de description de matériel, puis testé avec un banc de test logiciel. Les simulations issues de ces tests ont démontré le potentiel de l'architecture proposée, et ont mené à la création d'une puce faite sur mesure. Le présent mémoire fait trois contributions au projet : la conception d'une carte de support pour la puce du décodeur, l'ajout d'un mode quasi-synchrone, et la réalisation de tests pour valider le fonctionnement du système.

Précédant la fabrication de la puce, un mode quasi-synchrone a été ajouté au décodeur. Il s'agit d'un mode de test qui permet de suspendre le décodage après un certain nombre d'itérations pour consulter l'état du système. Le mode quasi-synchrone a été ajouté pour faciliter les tests en surfréquence ou en sous-tension, et son fonctionnement sera décrit dans le chapitre 3.

Une carte de support adaptée à la puce a ensuite été conçue pour y effectuer des tests automatisés. Ce circuit contient non seulement les dispositifs nécessaires afin que la puce fonctionne, il permet aussi de faire varier la tension et la fréquence de l'horloge du cœur contenant le décodeur, et de mesurer la puissance consommée par le système. Cette carte de support est conçue de façon à être utilisée avec la carte de développement FPGA VC707.

À l'aide du circuit imprimé, des routines de test ont été programmées puis exécutées sur la puce pour vérifier son fonctionnement. Ces tests ont permis d'observer des comportements qui ne concordaient pas avec le fonctionnement normal du décodeur, et qui limitaient la complétion du plan de test. La comparaison s'est faite par rapport à une émulation du décodeur sur une deuxième carte de développement. Des tests exploratoires ont suivi cette découverte afin de tenter d'obtenir plus d'information sur la puce du décodeur. Une analyse des comportements observés est présentée dans le chapitre 4. La source de ces inexactitudes est encore sujette à recherche.

Les différents éléments qui composent le banc de test de la puce seront détaillés dans ce document. L'infrastructure de test pourra ainsi être améliorée et réutilisée dans des projets futurs.

1.5 Plan du mémoire

Ce document est composé des sections suivantes. Le chapitre 2, divisé en deux parties, expliquera le fonctionnement du décodeur proposé ainsi que les méthodes de validation qui y

seront appliquées. Dans le chapitre 3, le banc de test et la carte de support de l'ASIC seront décrits en détail. Ensuite, les tests effectués sur l'ASIC et son comportement observé seront décrits dans le chapitre 4. Pour conclure, une synthèse des éléments abordés dans ce document, suivie d'hypothèses sur la source du non-fonctionnement de l'ASIC, seront fournies. Une liste d'améliorations potentielles sera aussi incluse dans cette discussion.

CHAPITRE 2 CONTEXTE

Le chapitre qui suit est divisé en deux parties. La première partie détaille le fonctionnement du décodeur LDPC et du code en tant que tel. La deuxième partie explique les considérations relatives au test d'un nouveau circuit intégré, ainsi qu'un résumé des méthodes utilisées lors de ce processus.

2.1 Principe de fonctionnement du décodeur EF-FECtive

La transmission de données à travers des environnements bruyants est un problème fondamental en télécommunication. La dégradation des signaux pendant une transmission peut sévèrement impacter la performance d'une communication si il n'y a aucune façon d'interpréter correctement le message. Les codes correcteur d'erreurs ont été développés comme l'une des méthodes pour surmonter ce problème. Ceux-ci ajoutent un élément de redondance à un message de façon à pouvoir le récupérer dans un signal bruité. En conséquence, l'implémentation de codes correcteurs d'erreurs se fait au prix d'un taux de transmission moins élevé [15].

Une chaîne de communication, qui a pour but d'acheminer une information d'un émetteur à un destinataire, peut généralement être décrite par le modèle suivant, illustré dans la Figure 2.1 ci-dessous. Ce dernier est composé des éléments suivant : l'émetteur, l'encodeur, le modulateur, le canal, le démodulateur, le décodeur et le destinataire. En commençant par l'émetteur, l'information brute sera premièrement traitée par l'encodeur. L'encodeur ajoute la redondance requise afin que le message puisse être correctement décodé par le destinataire. Une fois encodé, le modulateur convertit l'information en un signal adapté au canal, par exemple une onde radioélectrique dans l'atmosphère, un signal électrique dans une paire torsadée, ou un faisceau de lumière dans une fibre optique. Le signal modulé traverse ensuite le canal de transmission, où du bruit est injecté. Après avoir été captée par le récepteur, l'information est reconvertie par le démodulateur, qui transforme le signal modulé en message binaire. Le décodeur détecte et corrige les erreurs dans le message démodulé pour enfin récupérer le message original.

En fonction du canal par lequel le signal modulé est transmis, la distorsion qui l'affecte peut varier grandement. La modélisation du canal est alors nécessaire afin d'uniformiser l'évaluation de la performance d'un système de communication.

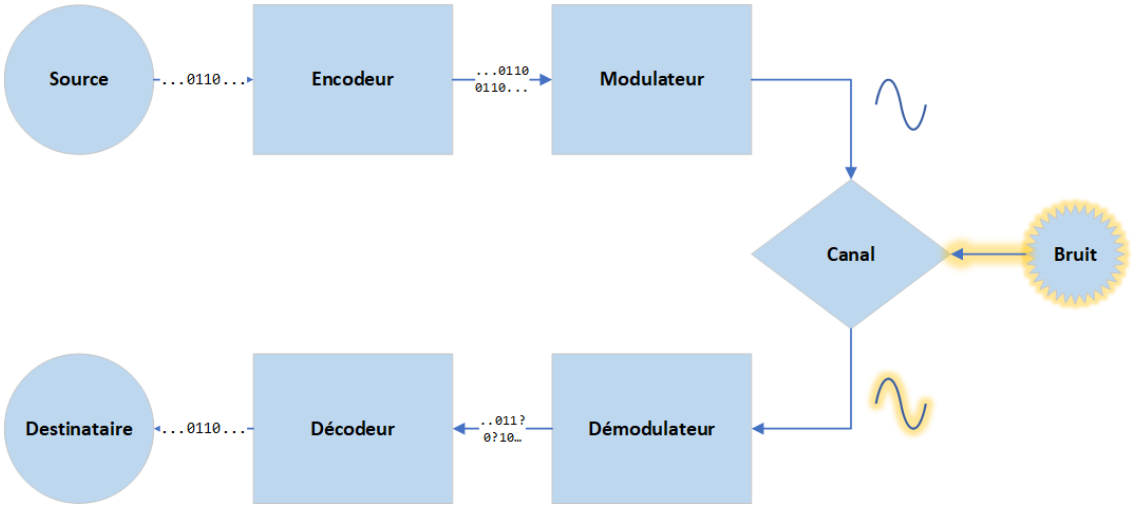


FIGURE 2.1 Modèle d'un système de communication

La performance des codes correcteurs est quantifiée en mesurant le taux d'erreurs post-décodage pour une puissance de bruit donnée. Le taux d'erreurs est mesuré au niveau des bits individuels (Binary Error Rate (BER)) ou au niveau des blocs de code (Frame Error Rate (FER)). Le bruit du canal est représenté comme étant le ratio entre l'énergie contenue par chaque bit (E_b, en joules) par rapport la densité spectrale de puissance du bruit (N_0, en V^2/Hz).

Le LDPC en particulier est une famille de codes correcteurs d'erreurs utilisée dans des protocoles de communication populaires tels que le IEEE 802.11 (Wi-Fi), la 5G, ou le DVB-S2. Le LDPC est aussi utilisé dans des systèmes de stockage de données. Ce code correcteur d'erreurs peut se rapprocher de la limite de capacité d'un canal de communication, d'où son intérêt **??**. La limite de capacité d'un canal de communication est le débit maximal qu'il peut supporter en fonction du bruit qui y est injecté [15].

2.1.1 Codes de parité à basse densité

L'histoire du LDPC remonte à 1960, quand sa description théorique a été mise sur papier par Gallager [16]. Toutefois, ce dernier n'a pas reçu beaucoup d'attention, car son implémentation était à ce temps très complexe. Le développement d'ordinateurs puissants a permis au LDPC de redevenir un centre d'intérêt en 1995 lorsqu'il a été redécouvert par Mackay et Neal [17]. Depuis cette redécouverte, plusieurs améliorations ont été faites sur le LDPC, autant au niveau de ses algorithmes de calcul que des architectures le supportant.

Comme son nom l'indique, le LDPC fonctionne en calculant la parité de certains sous-ensembles de bits et en l'ajoutant au message encodé. Afin de simplifier le décodage, de petits sous-ensembles sont choisis pour le calcul des parités, d'où la basse densité (*Low Density*) du code.

La structure d'un code LDPC peut être représentée sous deux formes, soit par sa matrice de parité ou par un graphe biparti [18], comme illustré dans la Figure 2.2. Ces deux formes permettent de visualiser les relations entre les différents bits contenus dans un paquet encodé. La matrice de parité $\mathbf{H}$ décrit le système d'équations encadrant un code LDPC et est de format $m \times n$ avec des poids de ligne r et de colonne g. Cette matrice est à basse densité, c'est-à-dire qu'elle contient en majorité des zéros. Un code LDPC est régulier s'il satisfait les conditions $r = g(m/n)$ et $g \ll m$. Si ce n'est pas le cas, le code est irrégulier. Un mot de code v est valide s'il satisfait l'équation $\mathbf{H} \cdot u^T = S = 0$. Le vecteur S correspond au syndrome du mot de code.

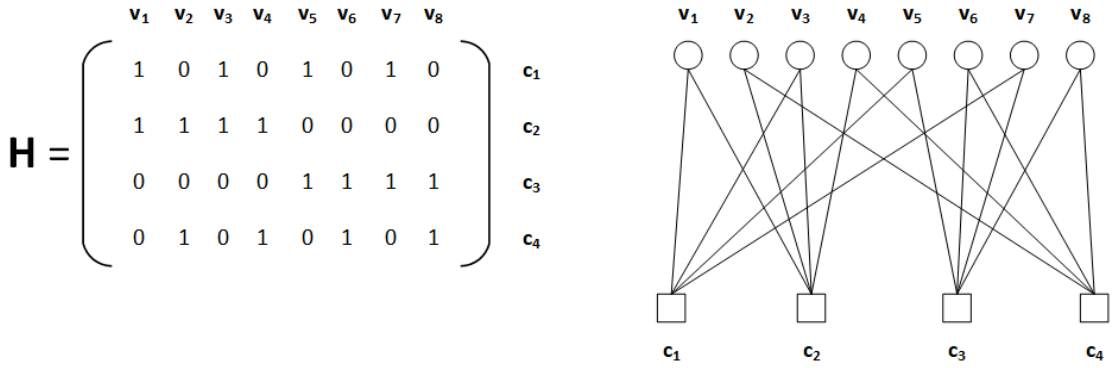


FIGURE 2.2 Représentations d'un code LDPC

Le LDPC est un code correcteur d'erreurs linéaire et en blocs. Un code à blocs encode l'information par paquets, c'est-à-dire que le message contient un nombre fini de symboles. Cela les distingue des codes convolutifs où l'information encodée est habituellement un flux (*stream*). Un code dit linéaire contient des mots de code qui, lorsque combinés avec une opération linéaire, génèrent un nouveau mot de code valide. Cette propriété rend le décodage plus efficace. Dans un code à blocs, un message (u) contenant un nombre de bits k est encodé de façon à ajouter un certain nombre de bits de parité. Le message encodé (v) est composé du message ainsi que des bits de parité calculés (p), et contient n nombre de bits, qui est nécessairement supérieur à k ($n > k$). La génération du message encodé est régie par un ensemble d'équations de parité contenu dans une matrice génératrice $\mathbf{G}$, d'où $u \cdot \mathbf{G} = v$. La différence entre k et n est le nombre de bits de parité insérés durant l'encodage, et le ratio de k sur n quant à lui est le taux de transmission du code. Dans un code LDPC, le taux de

transmission R est déterminé à partir des dimensions de la matrice de parité avec la formule $R \leq 1 - (m/n)$. Il y a égalité s'il n'y a pas de lignes redondantes dans la matrice.

Le graphe biparti décrivant un code LDPC est aussi connu sous le nom de graphe de Tanner [19]. Il est composé d'un côté de nœuds de vérification (*check nodes*, ou CN) et de l'autre de nœuds variables (*variable nodes*, ou VN). Les arêtes reliant les nœuds de vérification aux nœuds variables représentent les contraintes du code, et ils sont équivalents aux éléments non nuls de la matrice de parité.

LDPC quasi-cyclique

Le LDPC quasi-cyclique (LDPC-QC) est un code LDPC pour lequel la matrice de parité est composée de matrices identités permutées circulairement [20]. Les dimensions de la matrice identité équivalent à la taille d'expansion (*lifting size*, z) du code. La taille d'expansion permet de borner le nombre de permutations qui peut être effectuée sur les sous-matrices entre $[-1; z-1]$, où $z = -1$ équivaut à une matrice nulle. Un code LDPC-QC peut ainsi être représenté en fonction de ce paramètre sous la forme d'un base-graphe.

Le LDPC quasi-cyclique a plusieurs avantages en lien avec son implémentation matérielle. Il permet entre autres de simplifier la parallélisation du décodage, car les messages reçus par un nœud de vérification donné sont indépendants des autres nœuds de vérification à l'intérieur d'une itération [21].

2.1.2 Décodage LDPC

Dans la chaîne de communication, la démodulation du signal précède le décodage. Le démodulateur convertit le signal analogique du canal et met chaque bit sous forme d'une probabilité afin d'effectuer un test d'hypothèse sur sa valeur. La valeur issue du démodulateur est la probabilité à priori, ou l'information intrinsèque du bit. Lors du décodage, l'objectif est de déduire la probabilité à posteriori du bit en vérifiant la vraisemblance, ou l'information extrinsèque de ce dernier. L'information extrinsèque est contenue dans le restant du message, c'est-à-dire dans les différents bits de parités auquel le bit concerné est relié, et suit une distribution $f(\cdot)$ qui dépend de l'encodage. Pour un bit b_i contenu dans un message reçu $\tilde{v}$, le calcul de sa probabilité à posteriori suit donc l'équation 2.1 [22].

$$\underbrace{\log \frac{Pr(b_i = 1|\tilde{v})}{Pr(b_i = 0|\tilde{v})}}_{\text{À postériori}} = \underbrace{\log \frac{f(\tilde{v}|b_i = 1)}{f(\tilde{v}|b_i = 0)}}_{\text{Vraisemblance}} + \underbrace{\log \frac{Pr(b_i = 1)}{Pr(b_i = 0)}}_{\text{À priori}} \tag{2.1}$$

Ces probabilités sont mises sous la forme de quotients logarithmiques afin de simplifier les calculs impliqués dans ce processus. Le calcul de la probabilité à priori dépend du canal utilisé, mais la valeur du quotient logarithmique de cette dernière est en général égale à 0, car il est autant probable de recevoir un 0 qu'un 1 lors d'une transmission ($Pr(b_i = 1) \approx Pr(b_i = 0)$).

Un décodage à décision dure (*hard decoding*) fonctionne en considérant uniquement la valeur binaire de l'information issu du canal. Afin d'approcher une transmission optimale, un décodeur à décision douce (*soft decoding*) utilisant une méthode itérative doit plutôt être utilisé [23].

Algorithme somme-produit

Le but principal du décodeur LDPC est de calculer l'information extrinsèque d'un bit encodé. Pour obtenir cette quantité à l'intérieur du réseau complexe de bits dont est composé un code LDPC, l'algorithme de propagation des convictions, aussi connu comme la transmission de messages somme-produit (Sum-Product Algorithm (SPA)) [24] est utilisé. Cet algorithme fonctionne en communicant des messages entre les nœuds variables et les nœuds de vérification pour accumuler progressivement la vraisemblance des bits du message.

À l'initialisation de l'algorithme, la vraisemblance d'un bit (ρ_i) provenant du canal est transmise aux nœuds variables et la valeur stockée dans les nœuds de vérification est remise à zéro. Pour une itération ι, les messages provenant d'un nœud variable i vers un nœud de vérification j sont dénotés $\lambda^{\iota}_{i\to j}$, et inversement les messages provenant d'un nœud de vérification vers un nœud de vérification sont dénotés $\gamma^{\iota}_{j\to i}$. L'ensemble des messages ciblant un nœud de vérification est dénoté $\mathcal{V}_i$, et l'ensemble des messages ciblant un nœud variable est dénoté $\mathcal{C}_i$. La somme de tous les messages provenant des nœuds de vérifications constitue la vraisemblance et la probabilité à postériori $\hat{\rho}^{\iota}_i$ est la somme de la vraisemblance et de la probabilité à priori. Les fonctions utilisées pour calculer ces messages et la probabilité à postériori sont les suivantes, adaptées de [25].

$$\lambda^{\iota}_{i\to j} = \rho_i + \sum_{x\in\mathcal{C}_i\setminus j} \gamma^{\iota}_{x\to i} \tag{2.2}$$

$$\gamma^{\iota}_{j\to i} = 2\cdot\tanh^{-1}\left(\prod_{x\in\mathcal{V}_i\setminus i}\tanh\left(\frac{1}{2}\cdot\lambda^{\iota-1}_{x\to j}\right)\right) \tag{2.3}$$

$$\hat{\rho}^{\iota}_i = \sum_{x\in\mathcal{C}_i}(\gamma^{\iota}_{x\to i}) + \rho_i \tag{2.4}$$

La mise à jour des valeurs de $\lambda_{i\to j}^{\iota}$ et de $\gamma_{j\to i}^{\iota}$ se fait le plus simplement par un ordonnancement parallèle (*flooding schedule*). À chaque itération, cette méthode d'ordonnancement met à jour les messages de tous les nœuds variables en premier, puis tous les nœuds de vérification par après. Un ordonnancement par couches (*layered schedule*) ou séquentiel permet toutefois de converger vers une solution approximativement deux fois plus rapidement [26]. L'ordonnancement par couches sélectionne des sous-ensembles de nœuds, des "couches", pour lequel le calcul des messages est fait et la valeur des probabilités est mise à jour. Une itération complète est alors divisée en sous-itérations pour effectuer le calcul sur tous les nœuds. La subdivision de ces itérations permet de limiter le délai de calcul des messages et de prendre en compte l'information extrinsèque calculé précédemment. L'ordonnancement par couches peut se faire au niveau des nœuds de vérification ou au niveau des nœuds variables.

Le décodeur EF-FECtive utilise un ordonnancement hybride qui se trouve à mi-chemin entre un ordonnancement parallèle et un ordonnancement par couche [13] [27]. Cet ordonnancement s'adapte naturellement en fonction du niveau de parallélisme utilisé lors du décodage. L'utilisation d'un ordonnancement hybride est pris en charge par le matériel. Le mécanisme architectural associé est appelé Δ-*update*. Dans un décodeur standard, les unités de calcul des nœuds de vérification mettent à jour la probabilité à postériori des bits reçus à chaque itération. En ce qui concerne le décodeur étudié, ces unités calculent plutôt la différence Δ entre la valeur passée et la valeur courante de cette probabilité ($\Delta_i^{\iota} = \hat{\rho}_i^{\iota} - \hat{\rho}_i^{\iota-1}$). En conséquence, l'information des messages $\gamma_{j\to i}^{\iota}$ est préservé et ainsi, l'ordonnancement est plus flexible. En effet, lorsqu'un nœud variable est utilisé dans le calcul de plusieurs nœuds de vérification simultanément, l'ordonnancement peut être modifié afin d'éviter un conflit de données. La probabilité à postériori est mise à jour une fois l'itération terminée en additionnant Δ à sa valeur précédente.

Le décodage est mis à terme lorsqu'un certain nombre d'itérations est complété ou lorsqu'un critère d'arrêt est satisfait, par exemple lorsque $S = 0$.

Exemple de décodage

Dans cette section, un exemple de l'algorithme somme-produit appliqué au décodage LDPC sera fourni. À la source, une chaîne de bits u est générée et encodée pour produire le mot de code v. Lors de la réception du signal par le démodulateur, une liste de valeurs $\tilde{v}$ correspondant aux vraisemblances de chaque bits est produite. Ces valeurs sont les entrées du décodeur LDPC. La Figure 2.3 représente visuellement la suite d'opérations que réalise le décodeur pour la propagation des convictions.

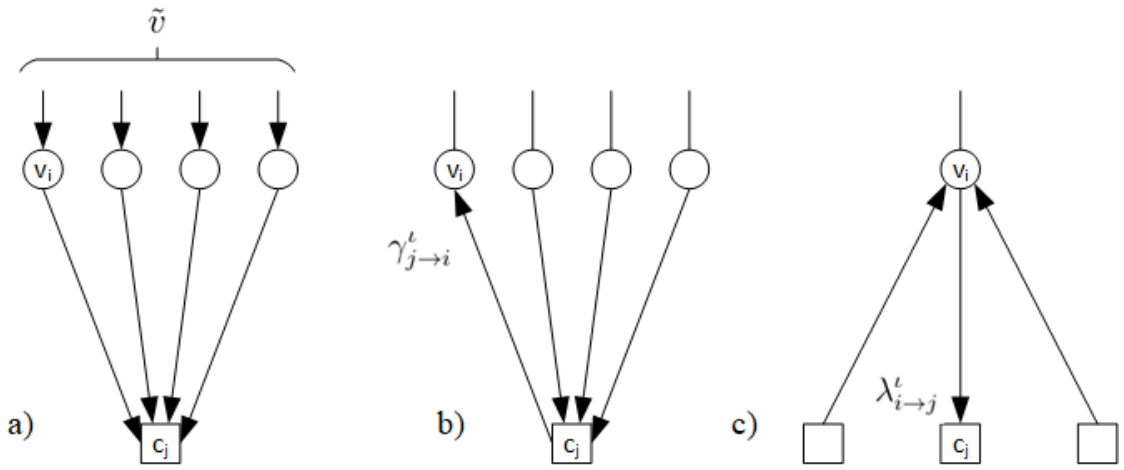


FIGURE 2.3 Propagation des convictions

Le décodage débute avec l'initialisation des nœuds de vérification, l'étape qui est représentée dans la Figure 2.3 a). À la première itération, les valeurs provenant du démodulateur sont transmises directement des nœuds variables aux nœuds de vérification.

La valeur contenue dans les nœuds variables est ensuite mise à jour en calculant les messages $\gamma^{\iota}_{j\to i}$ en utilisant l'équation 2.3. Cette étape est illustrée dans la Figure 2.3 b).

Enfin, dans l'étape c) de la même figure, la valeur des nœuds de vérification est à son tour mise à jour avec le calcul des messages $\lambda^{\iota}_{i\to j}$. Le calcul de ces messages suit l'équation 2.2.

Les étapes b) et c) se répètent jusqu'à l'atteinte d'une condition d'arrêt, ce qui est habituellement à l'obtention d'un mot de code valide, ou après un certain nombre d'itérations. Les valeurs contenues dans les nœuds variables correspondent alors au probabilités à postériori du message. Ces valeurs sont utilisés pour effectuer une décision sur la valeur binaire du message.

Simplification de l'algorithme somme-produit

L'implémentation matérielle de l'algorithme somme-produit n'est pas efficiente sous sa forme présentée dans la section précédente. Afin de réduire la complexité du décodage, le décodeur EF-FECtive utilise une version améliorée du SPA : l'algorithme Offset Min-Sum (OMS) [28]. En estimant la valeur de $\gamma^{\iota}_{j\to i}$, il est possible de réduire la complexité de calcul de l'algorithme en échange d'une baisse de performance modérée [25]. Cet algorithme suppose que l'élément le plus petit domine le résultat du produit présent dans l'équation 2.3. La valeur estimée de $\gamma^{\iota}_{j\to i}$ est donc calculée en remplaçant ce produit par une recherche de minimum, tel que présenté dans l'équation 2.5.

$$\gamma_{j\to i}^{\iota} \approx \left\lfloor \max\left(\min_{x\in\mathcal{V}_i\setminus i}\left(|\lambda_{x\to j}^{\iota-1}|-\beta\right),0\right)\cdot\prod_{x\in\mathcal{V}_i\setminus i}\left(\operatorname{signum}\lambda_{x\to j}^{\iota-1}\right)\right\rfloor \tag{2.5}$$

Cette opération est mieux adaptée à une implémentation matérielle. Les paramètres de biais β sont utilisés afin d'ajuster la performance de l'algorithme.

Architecture du décodeur EF-FECtive

L'un des objectifs principaux du décodeur EF-FECtive est d'opérer sous la tension de seuil. Ce régime d'opération a pour défaut d'augmenter le délai de propagation des portes logiques de la puce [10]. Pour atténuer cet effet, une architecture hautement parallèle utilisant de petites unités logiques est considérée. Une telle architecture supporte une horloge à plus haute fréquence, ce qui permet d'atteindre un débit binaire élevé avec peu de latence.

Le pipelinage est l'une des formes de parallélisme utilisé dans le décodeur EF-FECtive. Afin que cette méthode soit efficace, les dépendances de données entre les différents étages du pipeline doivent être minimisées pour éviter d'y introduire des conflits de mise à jour ou des délais d'attente, d'où l'intérêt d'utiliser le mécanisme Δ-*update* [13]. Ce mécanisme nécessite une unité de décalage additionnelle dans l'architecture du décodeur.

La Figure 2.4 illustre l'architecture du décodeur LDPC EF-FECtive. Deux sections se distinguent dans cette dernière : à gauche se trouve les gestionnaires de mémoire des probabilités à postériori des nœuds variables et les unités de décalage (*shift-*Δ*-mem*), tandis qu'à droite se trouve les unités de calcul des nœuds de vérification, qui génère les messages en amont et en aval de ces noeuds.

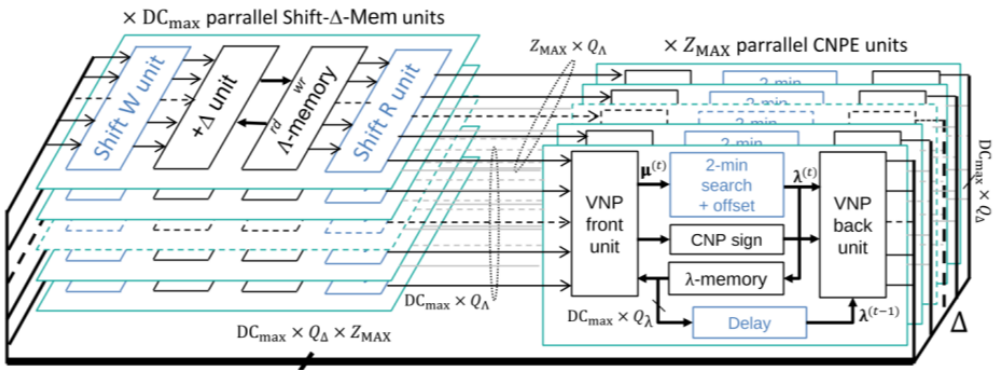


FIGURE 2.4 Architecture du décodeur EF-FECtive

Comme il est illustré dans cette figure, le pipelinage des opérations n'est pas la seule forme de parallélisme utilisée dans ce décodeur. L'ordonnancement hybride permet de calculer jusqu'à

w nœuds simultanément, avec $w \leq z$ où z est la taille d'expansion du base-graphe. Le nombre de nœuds qui peut être traité en parallèle varie en fonction de la densité de la matrice de parité et de la longueur du pipeline.

Les unités de calcul sont composés essentiellement d'un module de recherche de minimum. La recherche se fait sur les valeurs absolues des messages $\lambda_{i \to j}$, tandis que le signe est calculé avec une porte "OU exclusif". Pour effectuer cette opération, la largeur des entrées et des sorties doit être de d_{Cmax}, où d_{Cmax} est le degré maximal des nœuds de vérification.

Selon l'application dans lequel un code LDPC est utilisé, la structure et le format du code peuvent changer. D'une manière similaire, différents paramètres peuvent être ajustés auprès du décodeur. Pour cela, deux registres de paramètres et cinq tables doivent être initialisés dans le décodeur pour compléter un cycle de décodage. Ces tables sont la table de recherche des partitions (*Partition Lookup Table (LUT)*), la table des adresses (*Address Orchestration Table*), la table des activations (*Enable Orchestration Table*), la table des décalages (*Shift-Amount Orchestration Table*), et la table des biais (*OMS Offset Vector*). La table de recherche des partitions indique vers quel gestionnaire de mot de code un groupe de nœuds variable est acheminé. La table des adresses pointe vers l'adresse où doit se faire une lecture pour chaque gestionnaire de mot de code, à chaque sous-itération. La table des activations indique si, pour une sous-itération donnée, un gestionnaire de mot de code est actif. La table des décalages définit le décalage qui doit être appliqué à un groupe de nœuds variable. La table des biais spécifie la valeur du biais (*offset*) à utiliser pour chaque nœud de vérification.

Outre ces tables, les paramètres à définir sont les suivants :

- `curLifting` : Taille d'expansion du code (z)
- `curNbRows` : Nombre de lignes du protographe
- `useET` : Terminaison précoce du décodage
- `ET_iter_cnt` Nombre d'itérations précédant la terminaison précoce
- `nbBusWrdsPerProtVN` : Nombre de mots du bus par noeud variable
- `keepStateDecode` : Maintient de l'état post-décodage
- `MaxIteration` : Nombre d'itérations maximal

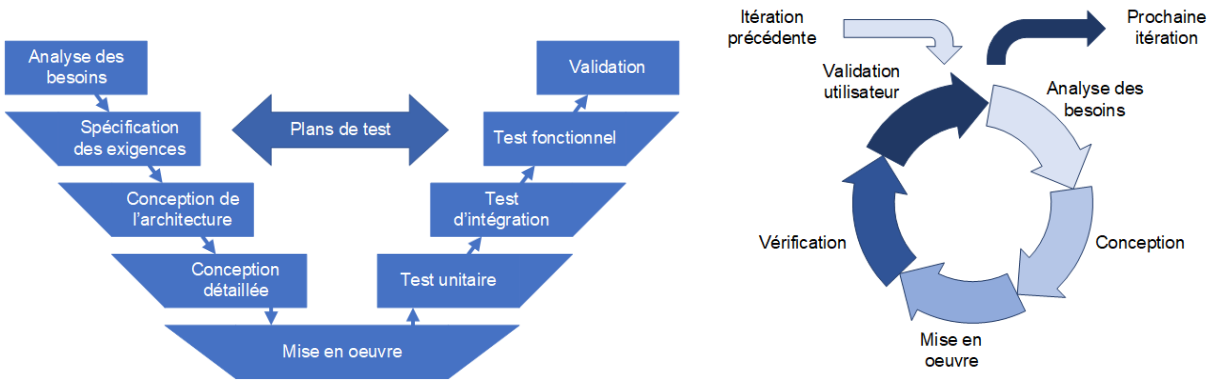
2.1.3 Conception de l'ASIC

L'architecture du décodeur a été implémentée dans une puce faite sur mesure, un ASIC, après avoir démontré en simulation que le pipelinage des opérations de décodage mène à une amélioration du produit énergie-délai [13] [27]. La création de cette puce a pour objectif premier d'étudier la performance et la consommation d'énergie du décodeur lorsque sa tension d'alimentation est réduite.

La section suivante porte sur les considérations à peser lors du test de l'ASIC, de la création du plan de test jusqu'à la validation du fonctionnement du décodeur. Dû à la complexité de la puce, l'atteinte des objectifs cités requiert une infrastructure de test spécialisée qui permettra de générer les données qui prouveront la performance du décodeur. Pour ce faire, nous avons suivi la méthodologie de test décrite ci-après.

2.2 Méthodologies de test

La méthodologie de test est une partie intégrante du cycle de vie d'un projet. Elle permet d'évaluer les performances d'un produit et d'ajouter des boucles de rétroaction dans le flux de travail du projet, pour ainsi améliorer la qualité du résultat. Ce processus est codifié dans différents modèles de gestion de projet. Le modèle en cascade (*Waterfall*), son successeur, le modèle en V, le modèle en spirale [29] ou encore le modèle Agile [30] sont des exemples modernes de méthodologies servant à encadrer le développement d'un produit. Illustrés dans la Figure 2.5 sont le modèle en V et le modèle Agile.



(a) Cycle de vie en V

(b) Cycle de vie Agile

FIGURE 2.5 Modèles de cycle de vie

Le modèle Agile, souvent utilisé en développement logiciel, prévoit un processus itératif, dans lequel un produit fini est atteint à chaque stade de développement, puis amélioré dans la prochaine. La méthodologie utilisée dans ce projet se rapproche plus du modèle en V. Ce dernier décrit le développement d'un produit par rapport à deux axes. Le premier, horizontal, décrit la progression du projet. Le deuxième, vertical, montre le niveau d'abstraction des différentes étapes du projet, du plus abstrait, l'analyse de la problématique, au plus concret, la mise en œuvre de la solution. Dans ce modèle, un projet est décomposé en trois phases. De gauche à droite, il y a la définition du projet, puis la mise en œuvre, et enfin la validation. La définition du projet permet de mettre sur papier les fonctionnalités à vérifier lors de la

phase de validation une fois que la mise en œuvre est complétée. Liant ces deux phases est le plan de test, qui liste les opérations qui seront effectuées afin de vérifier ces fonctionnalités.

2.2.1 Validation, vérification, et tests

La validation d'un système peut se faire par différents moyens. Un système peut être validé, par exemple, en prouvant mathématiquement son égalité au modèle qui le représente par une méthode formelle [31]. Par contre, pour un système complexe, exposé à des contraintes matérielles, il est typiquement plus accessible d'effectuer des tests pour observer le comportement de ce dernier et de le comparer au comportement attendu. Pour un circuit logique simple, la totalité des entrées possibles peut être testée pour obtenir une couverture complète du fonctionnement de ce dernier. Par contre, le nombre de cas à tester grandit exponentiellement en fonction du nombre d'entrées. Pour un circuit complexe, les tests effectués doivent être conçus de façon à produire et observer un ensemble de comportements spécifiques qui permettent de vérifier suffisamment le fonctionnement du système sous test [32]. Un modèle de référence approprié doit aussi exister de façon à connaître le comportement attendu. Illustrée dans la Figure 2.6 est une infrastructure générale de test.

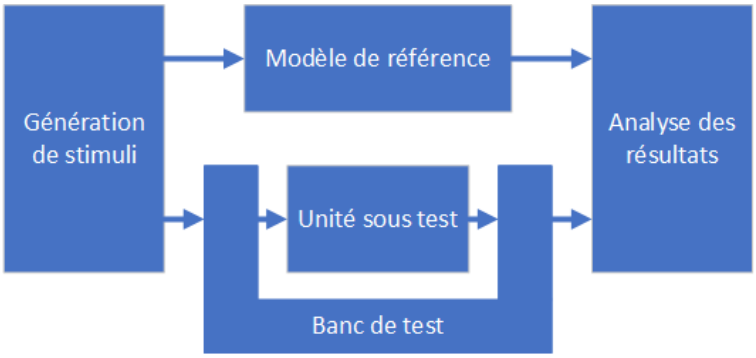


FIGURE 2.6 Unité sous test

La conception des tests commence avec la création d'un banc de test approprié. Le banc de test est un circuit spécialisé adapté à une unité sous test et a pour but de générer les entrées de l'unité puis d'en récupérer les résultats une fois que cette dernière a accompli sa tâche [33]. Ce dernier reçoit des vecteurs d'entrées d'un générateur externe et transmet en sortie les résultats du test pour effectuer une comparaison par rapport à un modèle de référence.

La création du plan de test dépend, comme il est mentionné dans la section précédente, des paramètres qui ont été établis lors de la définition du projet et du niveau d'abstraction sur lequel le test est effectué. Lors du développement d'un système, des tests sont effectués

sur les modules le composant de façon isolée afin d'éviter qu'un bogue se propage sans la connaissance du concepteur. Ces tests sont connus sous le nom de tests unitaires. Lors du test d'un circuit intégré, trois niveaux d'abstraction peuvent être différenciés : la puce, le circuit imprimé, et le système. Pour chacun de ces niveaux, les outils et les méthodes de test diffèrent. Une vérification fonctionnelle est réalisée une fois qu'un système sous test est complet. À ce point, une liste des fonctionnalités que doit accomplir le système est dressée basée sur les spécifications plus du projet.

2.2.2 Conception en vue de tests

La validation d'un système par tests repose sur les principes d'observabilité et de contrôlabilité, qui ensemble composent la testabilité de ce dernier [34]. La contrôlabilité décrit la capacité de l'utilisateur de modifier les stimuli et l'état du système sous test. L'observabilité décrit, elle, la capacité de l'utilisateur de récupérer des données en sortie de ce dernier. Une meilleure testabilité permet de rendre la validation d'un circuit intégré plus rapide et plus rigoureux. En conséquence, afin de rendre un système plus testable, un concepteur doit implémenter certains mécanismes lors de la réalisation de ce dernier [32].

Bien sûr, une application méthodique de principes de conception se doit d'être faite préalablement à la conception en vue de tests. La vérification des règles de conceptions (Design Rule Check (DRC)), par exemple, est une étape incontournable du développement d'un système. Le DRC est typiquement effectué automatiquement par les outils de conception une fois que les règles y ont été définies. L'utilisation de simulations comportementales et post-synthèse pour vérifier la justesse des circuits conçus est aussi un principe efficace pour limiter le nombre de défauts introduit au cours de développement.

Hormis cela, la façon la plus simple d'augmenter la testabilité d'un circuit intégré est d'ajouter des signaux d'entrée ou de sortie à des nœuds d'intérêt. Évidemment, cette méthode a des limites pratiques, mais peut être appliquée facilement lors de la simulation d'un circuit.

Afin d'augmenter le nombre de noeuds accessibles par l'utilisateur sans insérer une trop grande complexité au circuit, il est possible d'intégrer plutôt des cellules de test périphériques (*boundary scan cells*) et de former des chaînes de test nécessitant peu de signaux externes. Cette méthode est bien encadrée par le protocole Joint Test Action Group (JTAG) [35]. Ce dernier est communément utilisé pour vérifier la conformité d'une carte après montage et le fonctionnement des puces qui y sont.

Enfin, une autre méthode est d'ajouter des circuits logiques permettant à la puce d'effectuer en partie ou en totalité le rôle du banc de test. Cela est connu sous le nom de Built-In Self Test (BIST), et nécessite des ressources plus importantes au niveau matériel.

CHAPITRE 3 CONCEPTION DU BANC DE TEST

La validation de la puce requiert un contrôleur qui génère ses signaux d'entrée et récupère les résultats en sortie, en d'autres mots, un banc de test. Un banc de test logiciel et un banc de test émulé étaient utilisés pour effectuer des tests sur le décodeur avant la création de l'ASIC. Dans le contexte de ce mémoire, une carte de support pour l'ASIC est conçue afin d'y effectuer des tests, et l'infrastructure existante est mise à jour.

Le banc de test émulé est constitué de deux cartes de développement Xilinx VC707 interreliés par des cartes mezzanine XM105. La première carte de développement, appelée dans ce document l'émulateur de l'ASIC, contient le décodeur tandis que la seconde carte contient le contrôleur de test, appelée ici le testeur FPGA. Lors d'un test, ces cartes sont programmées avec leurs architectures correspondantes, puis contrôlées par l'entremise du bus AXI JTAG. Ce banc de test rend possible l'émulation du comportement physique des circuits synthétisés dans ce projet, et est décrit dans la Section 3.1.

Une fois les puces ASIC fabriquées, celles-ci sont placées sur une carte de support dont la conception est décrite à la Section 3.2. Le protocole d'utilisation du décodeur et la génération des routines de test sont décrits dans la Section 3.3. La simulation du banc de test joue ici un rôle clé, car les commandes qui sont envoyées au testeur FPGA proviennent d'un fichier généré par ce processus.

3.1 Description du banc de test

La Figure 3.1 montre le banc de test au niveau système. L'émulateur ASIC est attribué au port FMC2 et la carte mezzanine est attribuée au port FMC1. L'utilisation des cartes VC707 est décrite dans le document UG885 de Xilinx [36].

Comme illustré dans la Figure 3.1, la communication entre l'ordinateur et le décodeur doit se faire à travers le testeur FPGA. Depuis l'ordinateur, le testeur FPGA est programmé puis contrôlé par un port JTAG. En fonction des commandes qui lui sont transmises, le testeur FPGA communique avec le décodeur par le bus les reliant.

Dans la Figure 3.2 est une photo du banc de test. Les différents composants du banc de test sont visible. Dans la partie supérieur, le testeur FPGA est connecté à une carte d'extension à gauche et à la carte mezzanine à droite, sur lequel l'ASIC est visible sur son support. Dans la partie inférieure est la carte d'émulatation du décodeur, qui est relié au testeur par sa propre carte d'extension.

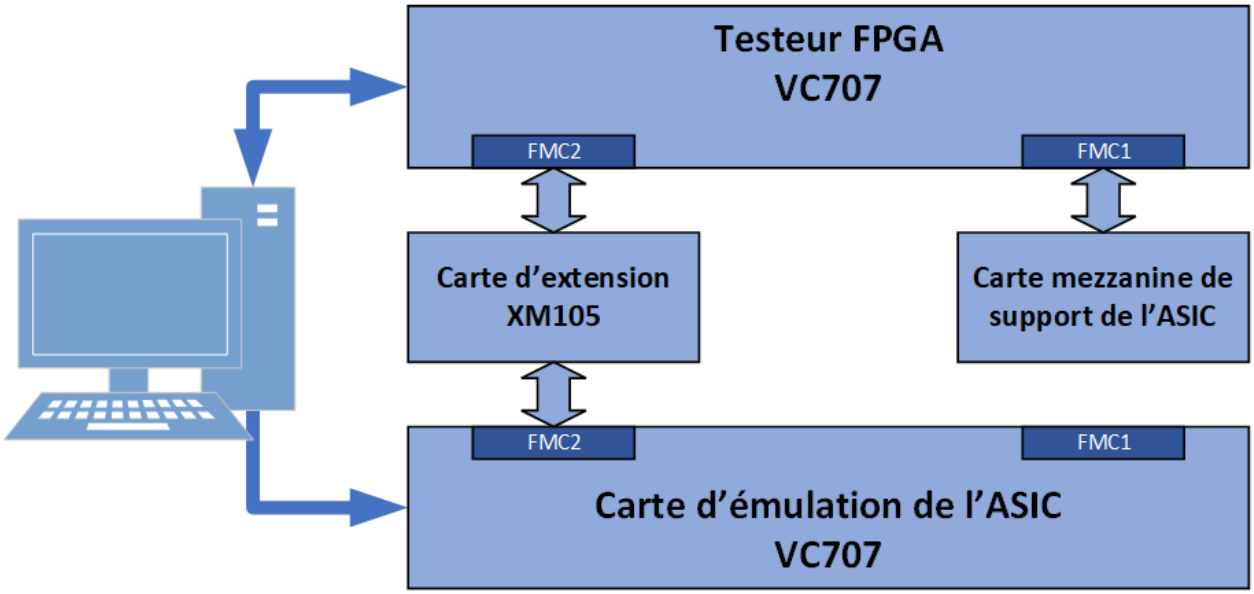


FIGURE 3.1 Schema bloc du banc de test

3.1.1 Testeur FPGA

Le testeur FPGA sert d'interface de transfert de données entre l'ordinateur et le décodeur. Dans la Figure 3.3 est illustré l'architecture haut-niveau du testeur. Le testeur FPGA est conçu dans le langage de description matérielle Verilog, et les blocs de propriété intellectuelle qui le composent sont gérés dans un projet Vivado, contenu dans le répertoire `hw/fpga_tester/ip`. Une fois programmé sur une carte, le testeur reçoit des commandes de l'ordinateur par un port USB à travers un module JTAG-AXI. L'information du bus sériel JTAG provenant du port USB est ici convertie pour être adaptée au bus AXI, un bus propriétaire de 32 bits. Le bus est relié à une mémoire RAM, au banc de registres, et au contrôleur du bus I^2C. Les adresses de base auxquels ces modules sont attribués sont 0x00000000, 0x00010000, et 0x00020000, respectivement.

La carte de support de l'ASIC utilise une version du testeur adaptée à elle. Les interconnexions entre le port FMC-HPC, l'ajout d'un signal de réinitialisation externe, et l'ajout du module I^2C sont les éléments qui diffèrent entre le testeur du décodeur émulé et le testeur de l'ASIC.

La mémoire RAM a une taille de 64 kilo-octets et agit comme mémoire tampon pour les tables et les mots de code. Le banc de registres contient le statut du testeur, une copie des paramètres du décodeur ainsi que les codes d'opération utilisés pour commander le testeur. Le contrôleur GPIO accède au banc de registre et à la mémoire RAM une fois que ceux-ci sont initialisés afin de générer les formes d'ondes appropriées pour communiquer avec le

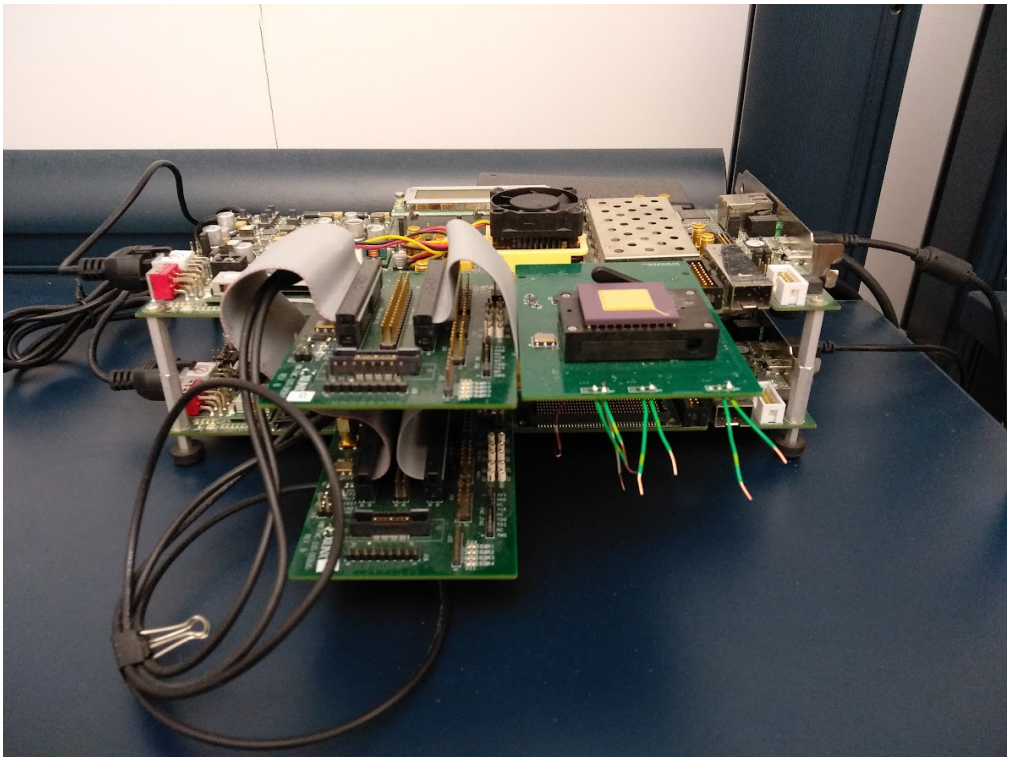


FIGURE 3.2 Photographie du banc de test

décodeur. Le contrôleur GPIO est une machine à états finis pour lequel la transition d'état est dictée par le code d'opération reçu. Un résumé de l'attribution des adresses et des codes d'opérations du testeur FPGA est présenté dans le Tableau 3.1.

Le testeur FPGA contient deux domaines d'horloge, qui sont générés par un bloc de propriété intellectuelle contenu dans le module horloge et reset. L'horloge rapide, `core_clk`, est programmée à 200MHz et est utilisée par le bus AXI et les composantes qui y sont connectées. L'horloge lente, `gpio_clk`, est utilisée pour la communication entre le testeur et le décodeur. L'horloge lente a une fréquence de base de 50MHz qui peut être subdivisée davantage en définissant une valeur au registre 0x0001000A. Une mémoire FIFO sépare le banc de registre du contrôleur GPIO pour synchroniser le passage de données entre les deux domaines. Les deux horloges du testeur sont acheminées au décodeur.

En aval du contrôleur GPIO sont des circuits tampons qui servent à lier les signaux internes du testeur à des signaux externes, en l'occurrence aux connecteurs FMC de la carte VC707. Les signaux externes sont associés à des noms et des configurations qui sont définis dans un fichier de contraintes appelé lors de la synthèse : (`hw/fpga_tester/syn/xc707_physical.xdc`). L'attribution des signaux internes au connecteur FMC est décrit dans le tableau en annexe C.

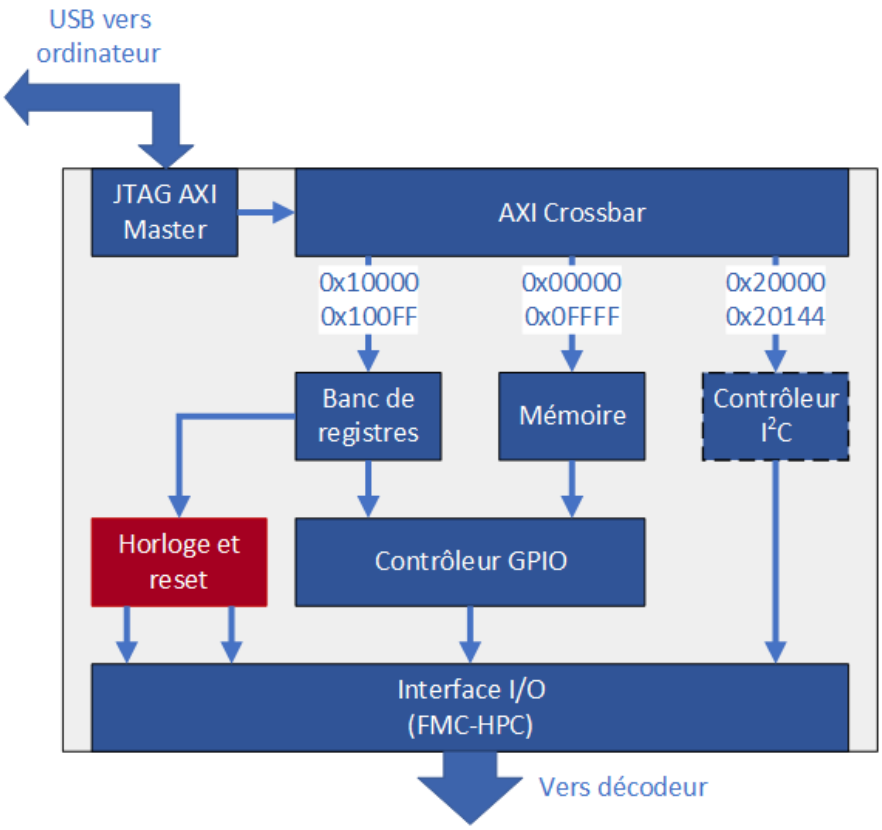


FIGURE 3.3 Schema bloc du testeur FPGA

3.1.2 Communication avec le décodeur

Tout comme le testeur FPGA, la communication avec le décodeur se fait à partir d'un bus de 32 bits et utilise des codes d'opération pour contrôler son état. Les adresses du bus et les codes d'opération du décodeur sont décrits dans le tableau 3.2.

L'adressage du décodeur se fait sur quatre bits, et les données du bus sont synchronisées avec les signaux de contrôle *write enable* et *read enable* en entrée, et *data valid*, *interrupt* en sortie. Les signaux *write enable* et *read enable* sont activés par le testeur FPGA pour indiquer au décodeur la nature de la transaction. Dans le cas d'une lecture, *data valid* est activé par le décodeur pour signaler au testeur la validité des valeurs affichées par le bus de données. Le signal *interrupt* est levé lorsque le décodage est mis à terme, et est remis à zéro avec le code d'opération 6.

3.1.3 Composantes I²C

La lecture des valeurs de courant traversant chaque rail de tension et le contrôle de la fréquence du coeur de l'ASIC se fait avec le bus I²C du connecteur FMC. Pour contrôler les composantes I²C de la carte mezzanine, un module AXI I²C a été ajouté au testeur FPGA. Les différentes composantes qui figurent sur la carte ont des adresses propres à elles, qui sont listées dans le tableau 3.3. La fréquence maximale du bus est de 400kHz, et la tension du bus en sortie de la carte de développement est de 3.3V. Le testeur FPGA pour l'émulation de l'ASIC ne comprend pas le contrôleur I²C.

L'opération du contrôleur I²C est décrite dans le document PG090 de Xilinx [37]. Les fonctions qui ont été conçues pour effectuer des transaction sur le bus I²C avec le contrôleur sont présentes dans le fichier `hw/verif/ldpc_top/ldpc_tb_pkg.sv`. Trois fonctions haut-niveau s'y retrouve : `iic_init`, `iic_write`, et `iic_read_n`.

La première fonction, `iic_init` initialise le contrôleur et ouvre le canal I²C du port FMC. La carte de développement a un multiplexeur (PCA9548A) qui permet d'activer différents canaux du bus I²C. Le bus du connecteur FMC1 est branché au canal 2 du multiplexeur. La deuxième fonction, `iic_write`, permet d'effectuer une écriture dans un registre spécifique d'une composante I²C. Finalement, la troisième fonction, `iic_read_n`, fait une lecture séquentielle de n octets d'une composante I²C à partir d'une adresse de base. Ces fonctions lisent en continu le registre d'interruption du contrôleur I²C lorsqu'une transaction est initiée. Une interruption est générée lorsqu'une transaction est conclue.

Pour tester ces fonctions avec le banc de test, un module simple qui émule le comportement d'une composante I²C a été créé pour retourner des signaux d'accusé de réception (*acknowledge*), sans quoi le contrôleur reste indéfiniment piégé dans une boucle. La description matérielle de ce module est accessible dans le dossier `hw/fpga_tester/hdl/iic_slave`.

À l'initialisation du banc de test, les capteurs de courant sont programmés pour retourner la valeur en sortie de leur convertisseur analogique à numérique amplifiée avec un gain de 8x. Cette valeur peut ensuite être lue lorsque nécessaire.

3.1.4 Génération du fichier de commande

Le banc de test logiciel, présent dans le répertoire `hw/verif/ldpc_top`, contient la routine de test du décodeur qui est décrite dans la Section 3.3. Il permet de simuler le comportement du décodeur avec le testeur FPGA et génère le fichier de commande pour le banc de test émulé. Tel qu'écrit dans le tableau 3.4, la simulation peut se faire à différentes étapes du flot de conception ainsi que sur différentes plateformes. Il est possible aussi d'utiliser le modèle

post-synthèse du circuit pour obtenir une simulation tenant compte des délais de propagation des portes et d'enregistrer les transitions prenant place dans le décodeur sous un fichier VCD ou SAIF lors de son opération.

Dans le banc de test logiciel, les paramètres du test, les tables, et les mots de code sont premièrement importés dans le fichier d'entête `hw/verif/ldpc_top/ldpc_tb_pkg.sv` depuis le répertoire `codes/` par le script `config.tcl`. Dans le fichier principal du banc de test, `hw/verif/ldpc_top/ldpc_tb.sv`, une instance du testeur FPGA et du décodeur, ainsi que les interconnexions nécessaires entre elles sont créés. Le signal d'horloge du système est généré par le simulateur, et le bus AXI du testeur FPGA est pris en charge par un bloc de propriété intellectuelle AXI VIP (Verification IP) dont l'utilisation est décrite dans le document PG267 de Xilinx. Les fonctions `axi_write`, `axi_read` et `axi_trans` utilisent ce bloc pour effectuer une écriture ou une lecture dans le bus, ou encore pour initier une transaction entre le testeur et le décodeur avec les codes d'opération du testeur. Dans la fonction `axi_trans`, le code d'opération 31 (Requête de transmission) est activé, puis rabaissé manuellement une fois que le registre de statut du testeur indique que la transaction est accomplie.

La génération du fichier de commandes du testeur FPGA se fait à travers ces fonctions. Le fichier de commandes fait appel à des fonctions analogues à ceux qui se trouvent dans le banc de test logiciel. Ces fonctions sont définis dans le script TCL qui ouvre le serveur JTAG pour communiquer avec les cartes de développement, `hw/fpga_tester/scripts/start_jtag_server.tcl`. Il est aussi possible d'écrire directement des lignes de code dans le fichier de commande du testeur à partir du banc de test logiciel avec la fonction `log_write`.

3.2 Carte mezzanine

Avec la fabrication de l'ASIC, la carte de développement contenant le circuit synthétisé n'est plus utilisée. Une carte mezzanine sur mesure est conçue pour supporter cette puce, varier sa tension d'alimentation, varier sa fréquence d'horloge, et mesurer sa consommation énergétique. Cette section décrit la carte de support et son utilisation.

La puce que doit supporter la carte contient deux domaines d'horloge et deux domaines de tension, comme illustré dans la Figure 3.4. Cela permet entre autres de varier les paramètres du décodeur tout en maintenant une interface de communication fiable. Entre ces deux domaines, il y a une mémoire FIFO pour synchroniser les transferts de données et des dispositifs de décalage de niveau (level shifters) pour adapter la tension des signaux. En opération nominale, l'ASIC est alimenté avec des tensions de 2.5V et de 1V, une horloge d'entrée-sortie fonctionnant à 50MHz et une horloge pour le cœur fonctionnant à 200MHz. Selon l'analyse

de temporisation statique (static timing analysis) de l'ASIC post-synthèse, il reste une marge de 9 picosecondes pour l'horloge du cœur à partir de cette fréquence.

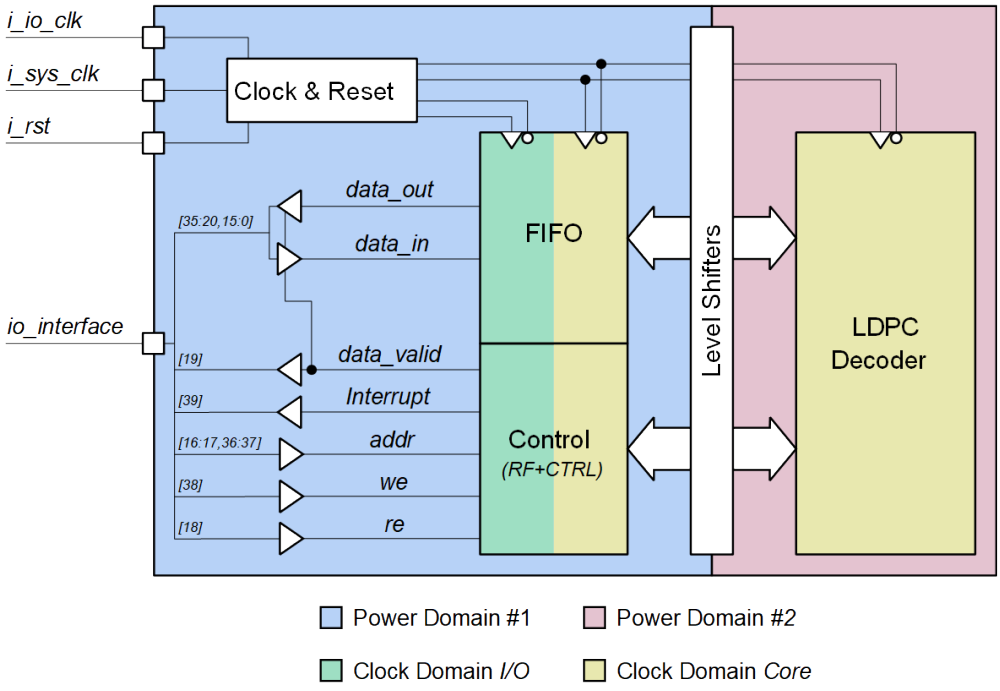


FIGURE 3.4 Vue de haut niveau de l'ASIC du décodeur

La puce communique avec un contrôleur par l'entremise d'un bus de données de 32 bits. L'adressage se fait sur 4 bits, avec les signaux de contrôle *read enable*, *write enable*, *interrupt*, et *data valid*. La réinitialisation se fait avec un signal synchrone actif haut.

L'ASIC est contenu dans un boîtier CPGA85 (Ceramic Pin Grid Array 85). Ce boîtier a 85 pattes distribuées dans une matrice carrée 11x11 distancé de 100 millièmes de pouce. L'attribution des pattes de l'ASIC est décrite en annexe B.

3.2.1 Critères de conception

Les critères principaux de conception de la carte sont les suivants :

— Compatibilité avec le testeur FPGA
— Rails d'alimentation fixe de 2.5V et de 1V
— Rail d'alimentation ajustable de 0.5V to 1V
— Horloge ajustable de 100-400MHz
— Capacité de mesurer la puissance de l'ASIC automatiquement
— Possibilité d'interchanger l'ASIC

À ces critères s'ajoutent les limitations physiques reliées aux capacités de fabrication du manufacturier. La facilité d'assemblage de la carte est aussi prise en compte dans le choix des composantes.

3.2.2 Conception de la carte

La carte de support qui a été conçue en réponse à ces critères réutilise le connecteur FMC-HPC qui reliait les deux cartes de développement VC707 à travers les cartes d'extension. Trois régulateurs de tension linéaires sont présents sur la carte pour pourvoir l'ASIC avec les rails d'alimentation nécessaires. Parmi ceux-ci est un régulateur configurable qui est utilisé pour varier la tension du cœur de l'ASIC. La carte contient aussi une horloge programmable avec une interface I²C pour contrôler la fréquence d'horloge du cœur de l'ASIC. Un support à puce permet d'interchanger l'ASIC sur la carte. La carte contient aussi des translateurs de niveau afin d'adapter les tensions d'entrée-sortie de la carte de développement à celles de la puce. Le schéma bloc ci-dessous illustre les différentes parties du circuit.

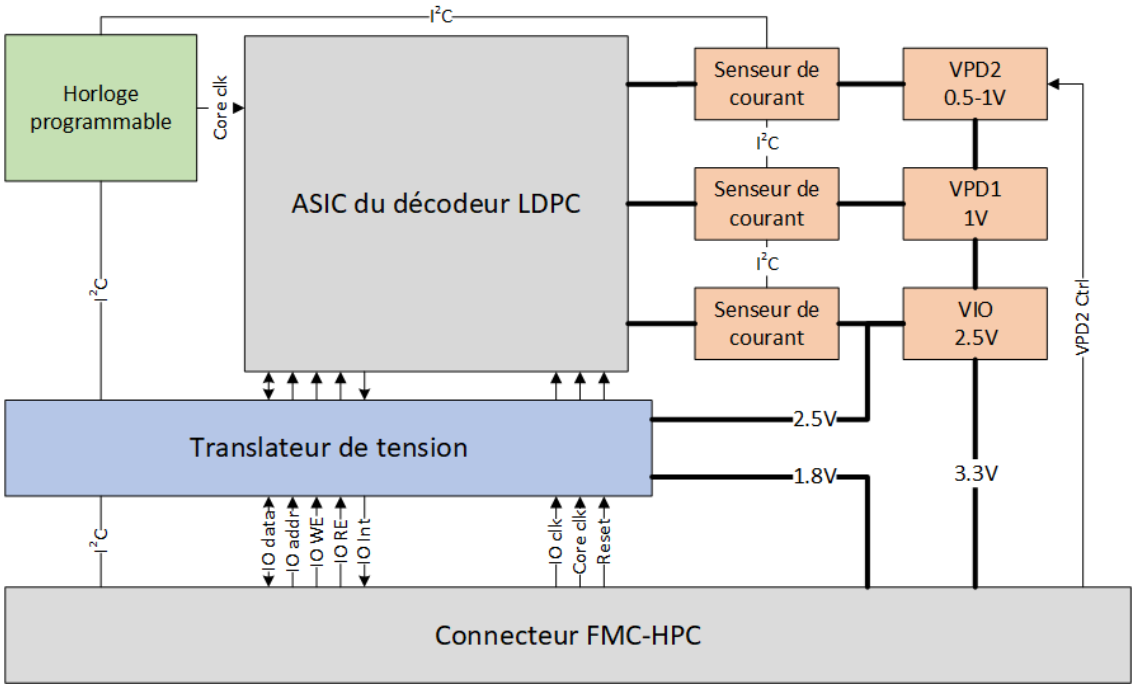


FIGURE 3.5 Schéma bloc de la carte mezzanine

Alimentation

Pour l'alimentation, des convertisseurs de tension linéaires ont été choisis. La carte mezzanine est alimentée depuis la carte de développement avec un rail de 1.8V et de 3.3V. La

tension de 3.3V est convertie en 2.5V, en 1V et en 0.5-1V ajustable. La tension de 1.8V est utilisée uniquement pour alimenter les dispositifs de décalage de niveau du côté de la carte de développement. L'alimentation de 3.3V peut supporter un courant allant jusqu'à 10A.

La puce choisie pour obtenir les tensions de 1V et de 0.5-1V ajustable est le TPS7A8401ARGRR, et a des entrées permettant de sélectionner sa tension de sortie. Ces pattes peuvent être activées en les attachant à la référence, ajoutant ainsi à la tension de sortie la valeur qui est associée à la patte, en plus d'une tension de base de 0.5V. La tension de 0.5 à 1V ajustable est prévue fonctionner en utilisant les ports d'entrée-sortie de la carte de développement. Une patte d'entrée inutilisée doit être gardée flottante.

Un circuit à diodes est présent en sortie afin de limiter la tension en sortie de la puce. Le circuit limiteur de tension utilise deux diodes Schottky qui ont une tension de seuil en polarisation directe de 0.55V. Une résistance limite le courant circulant dans les diodes. Lorsque la tension en entrée de ce circuit dépasse 1.1V, le rail est lié à la mise à la terre.

Ci-dessous est le schéma électrique du convertisseur 0.5-1V ajustable. À gauche se trouve le convertisseur de tension linéaire et à droite le capteur de courant, qui est décrit dans la Section 3.2.2. Les condensateurs d'entrée et de sortie ont été choisis selon les recommandations du manufacturier.

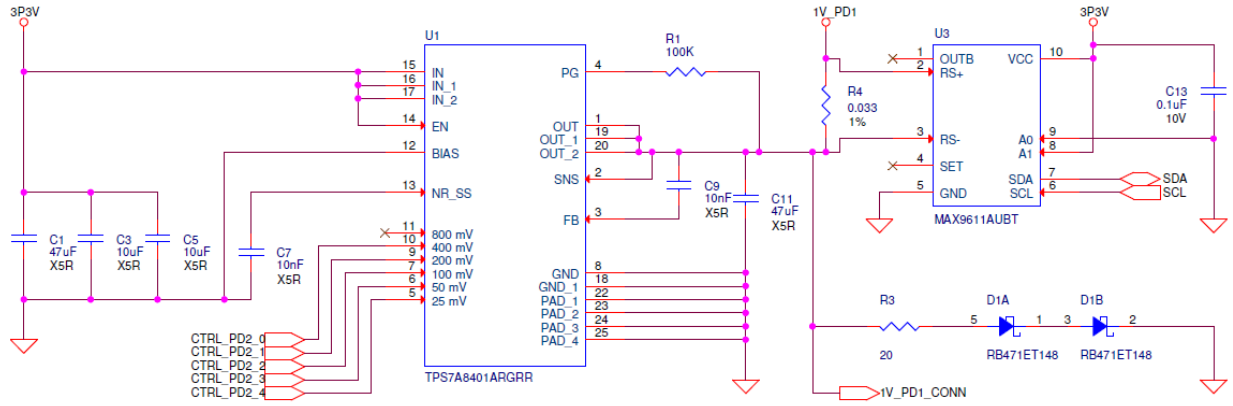


FIGURE 3.6 Schéma électrique de l'alimentation 1V ajustable

Afin de respecter les tensions d'entrée de l'ASIC, un décalage du niveau logique des signaux est nécessaire. La tension des signaux provenant du FPGA sont à 1.8V à niveau haut tandis que celle de l'ASIC est de 2.5V. Trois dispositifs de décalage de niveau, deux à 32 bits et un pour le bus I^2C sont présents sur la carte afin d'effectuer cette fonction. Le translateur de niveau du bus I^2C contient des résistances de polarisation à l'alimentation (*pull-up*) de 10kΩ.

Des condensateurs de découplage de 0.1uF ont été ajoutés à proximité de chacun des circuits logiques de la carte afin de limiter le bruit généré par la permutation des transistors sur les rails d'alimentation. Des condensateurs additionnels de 10nF sont présents pour l'ASIC.

Mesure de puissance

Pour effectuer des mesures de puissance, seulement la mesure de courant est critique, vu que les tensions sont connues. Le courant qui circule dans les différents rails de l'ASIC est déduit en déterminant la chute de tension à travers une résistance faible. Cette tension est amplifiée puis mesurée par une puce MAX9611AUBT, qui intègre un amplificateur opérationnel et un convertisseur analogue à digital par intégration à double rampe. L'amplificateur opérationnel intégré dans cette puce a un gain programmable de 1x, 4x ou 8x avant d'entrer dans le convertisseur, ce qui correspond à des plages de conversion de 440mV, 110mV, ou 55mV. La puce communique par le bus I^2C, et peut produire 500 échantillons de 12 bits par seconde.

La résistance choisie a une valeur de 0.033Ω avec une tolérance de 1%. Selon l'analyse de puissance post-synthèse de l'ASIC, le coeur consomme environ 0.6W lors de son fonctionnement. Pour une tension nominale de 1V, cela correspond à un courant de 0.6A. Considérant ces valeurs, la chute de tension mesurée à travers la résistance équivaut à 19.8mV. La mesure du courant nécessite donc environ 2% de la tension d'alimentation du coeur. Avec l'amplification maximale de 8x, 36% de la plage de conversion est utilisée.

Horloges

Deux horloges sont utilisées par l'ASIC : l'horloge du GPIO et l'horloge du cœur. Ces deux horloges peuvent être contrôlées par la carte de développement FPGA. L'horloge du cœur peut aussi être lue par la carte de développement afin de confirmer le fonctionnement de la puce d'horloge programmable. La puce d'horloge peut être programmée pour avoir une fréquence de 10MHz jusqu'à 810 MHz. La direction du signal est contrôlée depuis le translateur de niveau par une sortie provenant de la carte de développement (`sys_clk_dir`). L'horloge du coeur provenant de la puce ainsi que celui provenant de la carte de développement peuvent tous les deux être mis en état d'haute impédance (*tristated*) avec les signaux `ext_clk_oe` et `sys_clk_oe` respectivement, afin d'éviter des conflits.

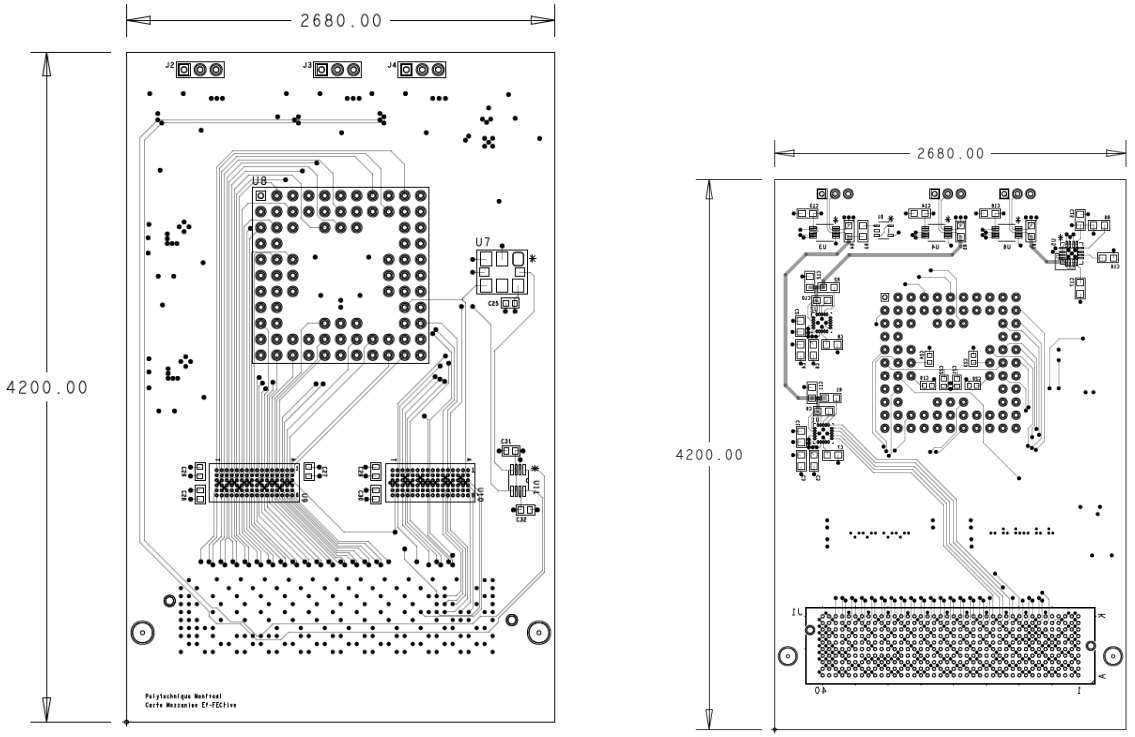
Simulation

La simulation de la carte mezzanine a été considérée une fois que la conception était complétée. Différentes méthodes de simulation ont été considérées, mais l'idée a été abandonnée

à cause de contraintes de temps, du manque de documentation, et de la difficulté de trouver des librairies de simulation appropriées. Une cosimulation Simulink-AMS semblait toutefois être la méthode la plus prometteuse.

Circuit imprimé

Le circuit imprimé a été conçu sur 4 couches de conducteur afin de faciliter le routage des signaux. Les rails d'alimentation et la référence (*ground*) sont contenus dans les deux couches internes. Les couches extérieures contiennent le restant des interconnexions, soit le bus de données, le bus d'adresse, les signaux I^2C, etc. Ci-dessous sont les dessins des couches externes de la carte.



(a) Dessus de la carte mezzanine (b) Dessous de la carte mezzanine

FIGURE 3.7 Couches externes de la carte mezzanine

Le manufacturier sélectionné pour la fabrication du circuit imprimé est JLCPCB.

3.2.3 Assemblage

Les instruments nécessaires pour l'assemblage de la carte sont les suivantes.

- Fer à souder ajustable
- Four à soudage par refusion (*reflow oven*)
- Pistolet à air chaud ajustable
- Pochoir de pâte à souder
- Raclette d'application de pâte à souder
- Pinces de précision anti-statiques

La liste des matériaux est présente en annexe A.

Certaines composantes de la carte de support ont des empreintes qui peuvent difficilement être installées avec un fer à souder, en particulier les composantes à matrice de billes comme le connecteur FMC-HPC ou les translateurs de tension. La méthode idéale pour poser ces pièces est d'utiliser un four à refusion. Pour cette raison, un pochoir pour l'application de pâte à souder a été commandé auprès du fabricant pour accompagner les circuits imprimés. Par contre, vu qu'il y a des composantes de ce type sur les deux côtés de la carte, un pistolet à air chaud doit aussi être utilisé.

Il est mieux de commencer l'assemblage de la carte par le côté opposé à l'ASIC, où se trouve le connecteur FMC-HPC (J1) ainsi que les composantes d'alimentation (U1 à U6). Ci-dessous est le diagramme d'assemblage ce ce côté de la carte, qui indique la position et les désignations de référence des pièces s'y trouvant.

La pâte à souder est premièrement appliquée sur la carte avec l'aide du pochoir et de la raclette d'application. Les pièces qui se trouvent sur ce côté de la carte sont ensuite placées à leur position spécifiée, puis la carte est insérée dans le four à soudage avec le profil de température approprié.

Une fois cela complété, le résultat de la cuisson est vérifié visuellement. Il est possible à partir d'ici d'effectuer l'ensemble de tests T1 décrit dans la Section 4.1 pour confirmer la validité des soudures.

Avec un premier côté assemblé, la carte est retournée pour y ajouter les pièces se trouvant sur le côté opposé. Ci-dessous est le diagramme d'assemblage du dessus de la carte mezzanine.

Sur ce côté se trouve l'horloge programmable (U7), les translateurs de niveau (U9, U10, et U11), ainsi que le support de l'ASIC (U8). Avec la deuxième moitié du pochoir, la pâte à souder est appliquée sur ce côté de la carte, et les composantes posées. Le pistolet à air chaud est utilisé pour chauffer les composantes jusqu'à l'atteinte du point de fusion de la pâte. Il

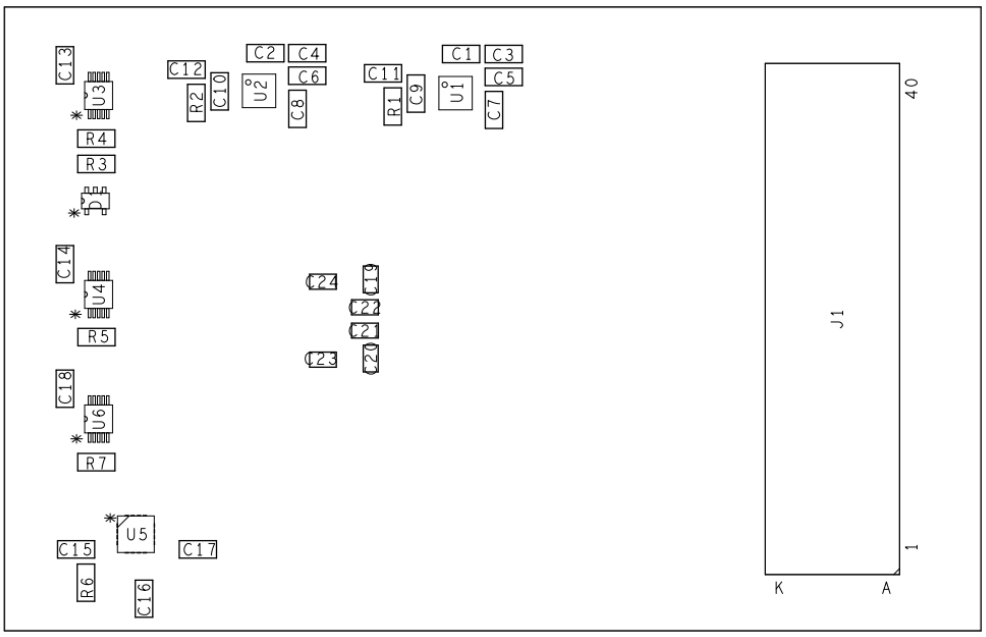


FIGURE 3.8 Assemblage du dessous de la carte mezzanine

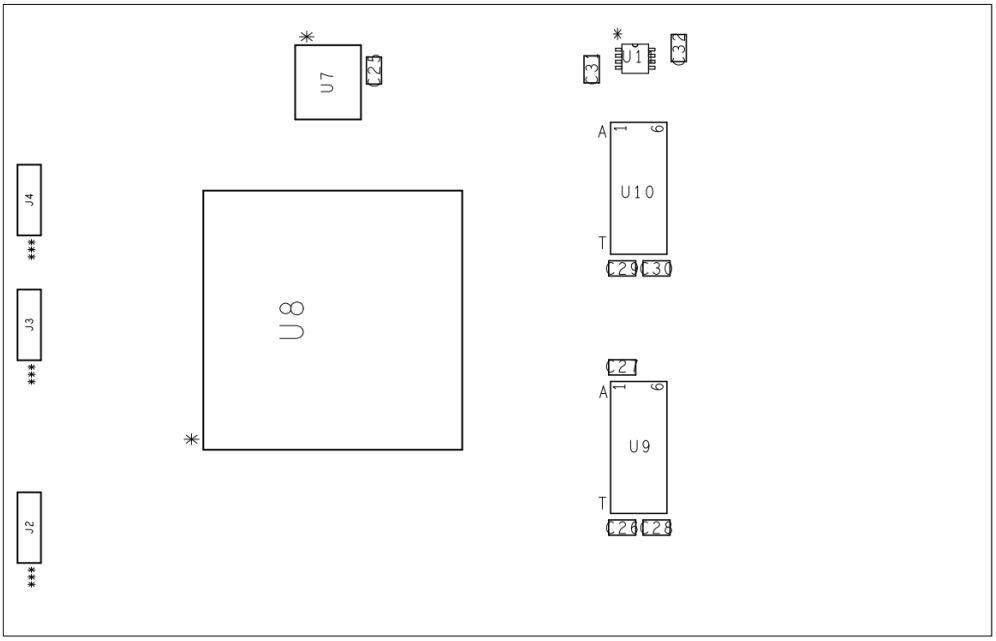


FIGURE 3.9 Assemblage du dessus de la carte mezzanine

faut ensuite assembler le support à puce U8 en y insérant les pattes à leur position désignée, et l'installer sur la carte avec le fer à souder. Une fois cette opération complétée, les tests de l'ensemble T1 sont répétés pour les composantes concernées.

Une fois la carte assemblée, des tests matériels plus avancés peuvent être effectués afin de confirmer la totalité de son fonctionnement.

3.3 Routine de test

3.3.1 Structure du code

Afin d'effectuer des tests sur le décodeur, un réseau de scripts, de vecteurs de tests, de fichiers de synthèse, de fichiers de configuration, et de fichiers de simulation est nécessaire. L'environnement logiciel est organisé en différents répertoires afin de simplifier l'accès aux parties qui sont liées à un projet donné. L'essentiel des éléments discutés dans cette section se retrouve dans le dossier de la portion matérielle du projet (`ldpc_asic/hw`). Par contre, la portion logicielle (`ldpc_asic/sw`) est utilisée pour produire les vecteurs utilisés dans les simulations du banc de test. Les fichiers du projet sont contenus et gérés dans un répertoire Git.

Avant de pouvoir utiliser les programmes contenus dans le projet, il faut premièrement initialiser l'environnement logiciel en lançant le script `hw/setup.csh`. Ce script définit des mots-clés permettant d'accéder plus facilement aux différents répertoires, librairies et licences nécessaires pour l'exécution de différentes tâches.

Une multitude de configurations pour la synthèse et l'exécution des différents bancs de tests coexiste en parallèle à celui de la carte mezzanine. Les options disponibles sont listées dans le tableau 3.4. Ces configurations sont accédées principalement par le façonneur de fichier (*Makefile*) du répertoire `ldpc_asic/hw`. L'appel du Makefile se fait typiquement sous la forme "make <o> TOP=<t> [CFG=<c>] [THREAD=<n>] [DEBUG=<d>]", où <o> est l'opération à exécuter tel que décrit dans le tableau 3.4, <t> est le nom du fichier *top*, <c> est le nom du fichier de configuration du décodeur, <n> est le nombre de processus à lancer, et <d> est l'état du mode debug. Les variables entre crochets sont facultatives.

En fonction de l'opération demandée, le Makefile appelle l'un des scripts contenus dans le fichier `hw/scripts`.

Les fichiers de description matérielle du décodeur sont dans le fichier `hw/hdl`, et ceux du testeur FPGA sont dans le fichier `hw/fpga_tester/hdl`. La synthèse de ces fichiers se fait par des scripts spécifiques aux plateformes appelant les outils Vivado pour la synthèse FPGA et Genus pour la synthèse ASIC.

Tel que mentionné précédemment, il faut premièrement générer les commandes passées au testeur FPGA avant d'exécuter une routine de test sur une plateforme donnée. Ces commandes

sont générés automatiquement sous la forme d'un fichier TCL lors de la simulation de la plateforme en question. Un banc de test logiciel est utilisé afin d'effectuer des simulations sur l'architecture du décodeur. Sa description est contenue dans le fichier `hw/verif/ldpc_top`. Il simule la séquence d'opérations à effectuer afin d'initialiser le système composé du testeur FPGA et du décodeur et d'effectuer le décodage d'un bloc de code LDPC. Une fois que le programme est complété, le banc de test compare les données en sortie du décodeur aux résultats attendus. Les vecteurs de test en entrées et en sortie du décodeur proviennent d'un modèle écrit en C. La section suivante explique l'utilisation du testeur FPGA et du banc de test logiciel.

La routine principale de test est composée des étapes suivantes.

1. Initialisation du testeur FPGA
   (a) Remise à zéro du système
   (b) Initialisation du bloc de vérification AXI
   (c) Initialisation du bus I^2C
   (d) Écriture des paramètre du décodeur
   (e) Écriture de la mémoire des mots de code
   (f) Écriture des tables
2. Test du décodeur
   (a) Transfert des tables et des mots de code
   (b) Relecture des tables et des mots de code
   (c) Décodage
   (d) Vérification des résultats

3.3.2 Initialisation des paramètres

La remise à zéro du décodeur est la première étape de la routine de test. Ceci est fait en basculant la valeur du bit 31 (Réinitialisation du décodeur) du registre de paramètres du testeur FPGA, qui est lié directement à une patte externe du FPGA, qui, elle-même, est liée au signal `rst` du décodeur. Le signal de remise à zéro du décodeur est actif haut, et doit être maintenu dans cet état au moins durant 2 cycles d'horloge.

Avant de commencer un cycle de décodage, les paramètres de l'ASIC sont initialisés avec la configuration appropriée. Deux registres de paramètres existent dans le décodeur, qui contiennent l'information sur la structure du code utilisé et sur le mode de fonctionnement du

décodeur. La valeur de ces paramètres est définie dans le fichier de configuration de la routine de test. Celles-ci sont ensuite écrites dans le testeur FPGA en y accédant directement par le bus AXI. Une fois écrits, les codes d'opérations 8 (Transmission des registres de paramètre du décodeur) et 24 (Réception des registres de paramètre du décodeur) du testeur sont utilisés pour débuter une transaction entre le testeur et le décodeur.

3.3.3 Transmission et réception des tables

Une fois que les paramètres du décodeur sont entrés dans leurs registres respectifs, les tableaux de données contenant les mots de codes et la structure du code LDPC en question sont transmis au testeur, puis à l'ASIC. La transmission de ces tables se fait de façon séquentielle. Le pointeur d'écriture et de lecture est réinitialisé en utilisant les codes d'opération 0 à 5 du décodeur, et le pointeur est incrémenté automatiquement à chaque accès à une adresse.

Au niveau du testeur FPGA, le nombre de mots qui est envoyé est compté lors du transfert des tables. Une fois que le compteur atteint un nombre donné, l'adresse en sortie du testeur est mise à jour pour continuer l'écriture à la prochaine table. La transmission et la réception des tables et des mots de code sont signalées auprès du testeur avec les codes d'opération 9, 10 (Transmission), 24, et 25 (Réception).

Les tables sont relues une fois transmises à l'ASIC pour valider que le données qui y ont été écrites sont conformes. Le décodage est ensuite lancé avec le code d'opération 16 (débute le décodage) pour le testeur FPGA.

3.3.4 Mode quasi-synchrone

Un mode quasi-synchrone a été ajouté pour faciliter les tests de performance en sous-voltage qui sont prévus pour l'ASIC. Le nom quasi-synchrone réfère à d'une part à l'opération d'un circuit synchrone avec des violations des contraintes temporelles, et d'autre part à un mode de validation donnant la possibilité d'interrompre momentanément le décodage à une itération donnée pour inspecter le contenu de la mémoire du décodeur avant de poursuivre le processus.

Dans le banc de test logiciel, la routine de décodage en mode quasi-synchrone est fait avant le décodage en mode normal. Le décodage peut donc être complété normalement une fois que le mode quasi-synchrone a rempli sa tâche. Pour activer ce mode, il suffit d'activer le bit 30 (Maintient de l'état post-décodage) et de désactiver le bit 16 (Terminaison précoce du décodage) du registre de paramètre 1. Une fois que cela est fait, le décodeur s'arrête lorsqu'il atteint un nombre d'itérations maximal, défini dans le champ de bits 9 à 0 du registre de paramètre 2, et maintient son état à l'interne. Ceci permet de suspendre le décodage en cours

et d'examiner le contenu actuel de la mémoire tout en offrant la possibilité de modifier la tension et la fréquence du décodeur à chaque itération.

Au niveau logiciel, pour activer le mode quasi-synchrone dans une routine de test il faut mettre la variable `useET` à faux et définir les vecteurs `QST_iterations` et `QST_frequencies` dans le fichier de configuration. `QST_iterations` définit le nombre d'itérations qui sera exécuté par le décodeur avant de pauser, et `QST_frequencies` définit la fréquence à laquelle ces itérations sont exécutées.

3.3.5 Configuration des sondes ILA

La famille de Field Programmable Gate Array (FPGA) Xilinx donne accès à des outils logiciels permettant d'insérer des points de tests dans des circuits synthétisés et de capturer l'état de noeuds d'intérêt à chaque coup d'horloge. Le module permettant l'enregistrement de ces données est la sonde Integrated Logic Analyzer (ILA), et son opération est décrite dans le document PG172 [38]. La génération des sondes ILA est faite lors de la synthèse du circuit par l'entremise du script Tool Command Language (TCL) `hw/fpga_tester/syn/create_debug_ila`. Dans ce script, les noeuds d'intérêt sont sélectionnés et les sondes sont configurées de façon à utiliser le lancement de capture (*trigger*) avancé. Le lancement de capture avancé permet d'amorcer une capture en fonction de conditions définies par une machine à état. Le fichier `hw/fpga_tester/syn/trigger.fsm` contient les informations reliées à ce mécanisme.

Une fois générées, les sondes ILA sont initialisées dans le script `hw/fpga_tester/scripts/start_jtag_server.tcl` puis l'enregistrement est lancé. Lorsque la capture est complétée, les données accumulées sont enregistrées dans le fichier `probe_data.ila` de la plateforme concernée (`hw/fpga_tester/scripts/mezz`, par exemple). Ce fichier peut être importé dans Vivado à travers le gestionnaire de matériel (*Hardware Manager*) pour visualiser les formes d'ondes ainsi obtenues. Dans la Section 4.4, ces données sont utilisées pour tenter d'identifier la source du problème bloquant le déroulement normal de la routine de test.

Tableau 3.1 Attribution des adresses du testeur FPGA

Adresses	Bits	**Fonction**
0x00000000 à 0x0000FFFF		Mémoire RAM tampon
0x00010000		Codes d'opération
	0	Remise à zéro du pointeur de la table de recherche des partitions
	1	Remise à zéro du pointeur de la table des décalages
	2	Remise à zéro du pointeur de la table des adresses
	3	Remise à zéro du pointeur de la table des activations
	4	Remise à zéro du pointeur de la table des biais
	5	Remise à zéro du pointeur des mots de code
	8	Transmission des registres de paramètre vers le décodeur
	9	Transmission des tables vers le décodeur
	10	Transmission des mots de code vers le décodeur
	16	Débute le décodage
	24	Réception des registres de paramètre du décodeur
	25	Réception des tables du décodeur
	26	Réception des mots de code du décodeur
	31	Requête de transmission
0x00010001		Statut du testeur
0x00010002		Registre de paramètres 1
	7 à 0	Taille d'expansion du code
	15 à 8	Nombre de lignes du protographe
	16	Activation de la terminaison précoce du décodage
	20 à 17	Nombre d'itérations précédant la terminaison précoce
	29 à 24	Nombre de mots par noeud variable
	30	Maintient de l'état post-décodage
	31	Réinitialisation du décodeur
0x00010003		Registre de paramètres 2
	9 à 0	Nombre d'itérations maximal
	18 à 16	Sélection de la vitesse d'horloge du décodeur (Émulateur FPGA)
	31	Sélection de la banque SRAM
0x00010004	15 à 0	Taille de la table de recherche des partitions
0x00010005	15 à 0	Taille de la table des décalages
0x00010006	15 à 0	Taille de la table des adresses
0x00010007	15 à 0	Taille de la table des activations
0x00010008	15 à 0	Taille de la table des biais
0x00010009	15 à 0	Taille des mots de code
0x0001000A	15 à 0	Diviseur de fréquence de l'horloge GPIO
0x00020000 à 0x00020104		Contrôleur de bus I^2C

Tableau 3.2 Attribution des adresses du décodeur LDPC

Adresses	Bits	**Fonction**
0x00000000		Codes d'opération
	0	Remise à zéro du pointeur de la table de recherche des partitions
	1	Remise à zéro du pointeur de la table des décalages
	2	Remise à zéro du pointeur de la table des adresses
	3	Remise à zéro du pointeur de la table des activations
	4	Remise à zéro du pointeur de la table des biais
	5	Remise à zéro du pointeur des mots de code
	6	Remise à zéro du signal d'interruption
	10	Débute le décodage
0x00000001		Registre de paramètres 1
	7 à 0	Taille d'expansion du code
	15 à 8	Nombre de lignes du protographe
	16	Terminaison précoce du décodage
	20 à 17	Nombre d'itérations précédant la terminaison précoce
	29 à 24	Nombre de mots par noeud variable
	30	Maintient de l'état post-décodage
0x00000002		Registre de paramètres 2
	9 à 0	Nombre d'itérations maximal
	18 à 16	Sélection de la vitesse d'horloge du décodeur (Émulateur FPGA)
	31	Sélection de la banque SRAM
0x00000003		Table de recherche des partitions
0x00000004		Table des décalages
0x00000005		Table des adresses
0x00000006		Table d'activations
0x00000007		Table des biais
0x00000008		Temps d'exécution
	15 à 0	Compteur de cycles
	30 à 16	Compteur d'itérations
	31	Validation de la parité
0x00000009		Interruption
	0	Décodage complété
0x0000000F		Mémoire des mots de code

Tableau 3.3 Adresse des composantes I^2C

Composante	**Numéro d'article**	**Adresse**
Multiplexeur I^2C	PCA9548A	0x74
Horloge programmable	570FAB000544DG	0x5C
Capteur de courant (VIO)	MAX9611AUBT	0xE0
Capteur de courant (VPD1)	MAX9611AUBT	0xE6
Capteur de courant (VPD2)	MAX9611AUBT	0xF8

Tableau 3.4 Commandes du façonneur de fichier

Commande	**Description**
sim_asic_beh	Simulation fonctionnelle du modèle comportemental de l'ASIC
sim_asic_syn_saif	Simulation temporelle du *netlist* ASIC avec activité SAIF
sim_asic_syn_vcd	Simulation temporelle du *netlist* ASIC avec activité VCD
sim_fpga_beh	Simulation fonctionnelle du modèle comportemental du décodeur sur plateforme FPGA
sim_fpga_syn_saif	Simulation temporelle du *netlist* FPGA avec activité SAIF
sim_fpga_syn_vcd	Simulation temporelle du *netlist* FPGA avec activité VCD
sim_mezz_beh	Simulation fonctionnelle du modèle comportemental du décodeur sur plateforme mezzanine
sim_mezz_syn	Simulation temporelle du *netlist* ASIC sur plateforme mezzanine
run_valid_fpga	Émulation FPGA du décodeur sur les cartes VC707
run_valid_mezz	Test de la puce ASIC sur la carte mezzanine
syn_asic	Flot de synthèse de la plateforme ASIC
syn_fpga	Flot de synthèse de la plateforme FPGA
syn_mezz	Flot de synthèse du testeur de la carte mezzanine
pwr_asic_saif	Analyse de puissance post-synthèse du *netlist* ASIC avec activité SAIF
pwr_asic_vcd	Analyse de puissance post-synthèse du *netlist* ASIC avec activité VCD

CHAPITRE 4 RÉSULTATS DE TEST

Les tests pour valider les fonctionnalités du banc de test, de la carte mezzanine, et de l'ASIC sont décrits ici. Des plans de tests sont conçus afin de couvrir la totalité des fonctionnalités prévues par les différents éléments du système. Certains de ces tests ont découvert des comportements inattendus. Une analyse approfondie de ces comportements est alors effectuée dans le but d'identifier la cause fondamentale du problème, et possiblement d'y remédier.

4.1 Tests matériels

Un premier plan de test a été établi afin d'assurer que tous les éléments du PCB sont fonctionnels. Le Tableau 4.1 liste les tests effectués sur la carte assemblée avant l'insertion de l'ASIC.

Tableau 4.1 Liste des tests effectués sur la carte mezzanine

	Objectif de test	**Appareil de mesure**
T1.1	Vérifier l'absence de courts-circuits entre les rails d'alimentation et la référence	Ohmmètre
T1.2	Vérifier l'absence de courts-circuits entre les pattes adjacentes des micropuces	Ohmmètre
T1.3	Vérifier les connexions du bus I^2C du port FMC vers les différents dispositifs I^2C	Ohmmètre
T1.4	Vérifier les connexions des apports d'alimentation du port FMC aux convertisseurs	Ohmmètre
T1.5	Vérifier les connexions du bus de données à partir du port FMC	Ohmmètre
T1.6	Vérifier les connexions des signaux d'adressage à partir du port FMC	Ohmmètre
T2.1	Mesurer les tensions d'alimentation des différents rails	Voltmètre
T2.2	Mesurer les tensions d'entrées-sorties de l'ASIC	Voltmètre
T2.3	Varier la tension du domaine de puissance 2	Voltmètre
T3.1	Vérifier l'intégrité des signaux d'horloge	Oscilloscope
T3.2	Vérifier l'intégrité des signaux du bus de donnée	Oscilloscope
T3.3	Vérifier l'intégrité des signaux du bus I^2C	Oscilloscope

Les instruments de mesure utilisés sont les suivants.

— Oscilloscope Tektronix TDS 2024C
— Multimètre Mastech M9502

L'ensemble de tests T1 est effectué durant et après l'assemblage du circuit imprimé. Ils permettent de confirmer qu'aucun court-circuit ne se soit créé au cours de l'assemblage, ce qui pourrait endommager la carte de développement, l'ASIC, ou les composantes de la carte. Une fois ces tests complétés, la carte mezzanine peut être connectée à la carte de développement avec un certain niveau de confiance.

L'ensemble de tests T2 permet de vérifier que, une fois connectées à la carte de développement, toutes les composantes d'alimentation du circuit imprimé soient fonctionnelles. L'ensemble de tests T3 nécessite la programmation de la carte de développement. Un programme simple de test liant les entrées-sorties de la carte mezzanine à des interrupteurs disponibles sur la carte de développement est utilisé pour vérifier la permutation de l'état des signaux, et le fonctionnement des dispositifs de translation de niveau.

4.1.1 Modifications à la carte suite aux tests

Des modifications critiques ont dû être faites à la carte suite à ces tests, telles qu'illustrées dans la Figure 4.1.

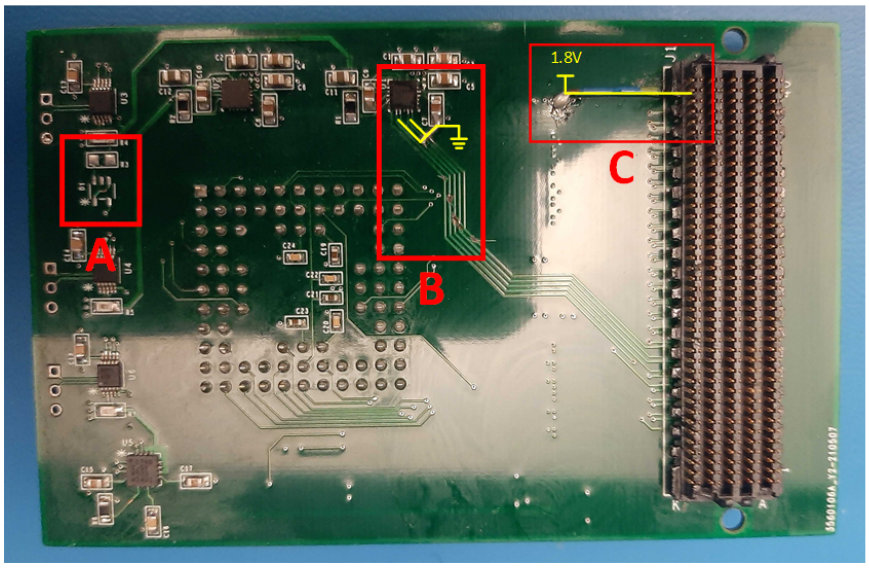


FIGURE 4.1 Modifications faites sur le PCB

La première erreur qui a été détectée suite à ceux-ci est le non-fonctionnement du circuit d'ajustement de la tension du coeur de l'ASIC, détecté lors du test T2.3. La tension à la sortie des circuits avait la valeur attendue lors de l'initialisation de la carte de développement, mais une fois signaux de contrôle configurés, la tension atteignait 2.7V, soit plus de 2 fois la valeur attendue.

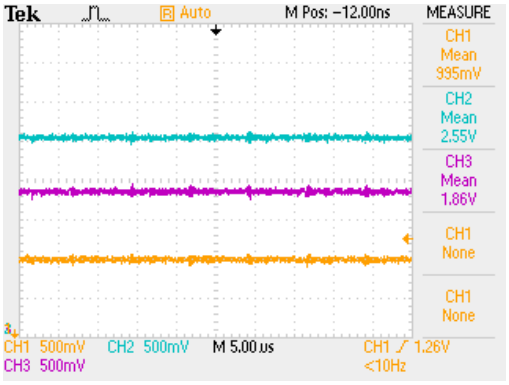
Les sources d'erreurs possibles sont les suivantes :

1. Circuit intégré du régulateur linéaire ajustable
2. Courts-circuits causé par les soudures
3. Tension en entrée du régulateur
4. Signaux d'entrées provenant de la carte FPGA
5. Circuit limiteur de tension

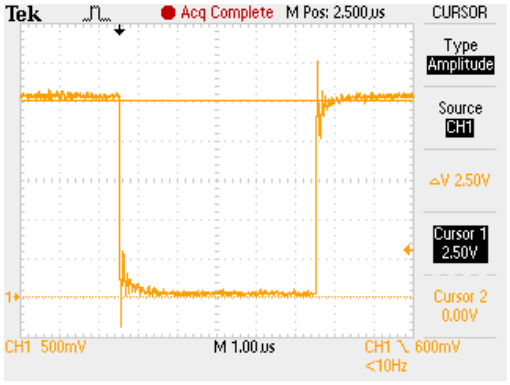
Il est possible d'éliminer les sources d'erreurs 2 et 3 car ils ont déjà été vérifiés lors des tests T1.2 et T2.1. De plus, le circuit de diodes (source d'erreur 5) qui était destiné à limiter cette tension a été enlevé (Figure 4.1, case A) afin d'isoler le circuit principal. Les résistances de polarisation (*pullup*, *pulldown*) internes aux sorties du FPGA ont été désactivées dans son fichier de configuration. Lorsque configurée ainsi, la résistance mesurée entre les signaux de contrôle et la référence est supérieure à 200MΩ et la résistance mesurée entre les signaux de contrôle et la tension source est de 3.3MΩ. Cela pourrait potentiellement être la cause du problème, car à l'intérieur du circuit intégré ces signaux sont reliés à un comparateur contrôlant la tension de sortie du régulateur. En conséquence, la tension a été fixée physiquement à 1V en coupant les traces reliant les pattes de contrôle du régulateur et la carte FPGA (Figure 4.1, case B).

Deuxièmement, lors du test T3.2 une partie des signaux du bus de données restait inactive lorsqu'une permutation était tentée. La source de ce problème a été retracée à la banque 36 d'entrées-sorties de la carte VC707, qui doit être alimentée par la carte mezzanine via les pattes FMC1_VIO_B_M2C du connecteur FMC. Un fil a été ajouté pour lier ce nœud au rail d'alimentation 1.8V (Figure 4.1, case C).

Suite à ces modifications, les mesures suivantes ont été faites avec l'oscilloscope, validant ainsi le fonctionnement du système d'alimentation de la carte mezzanine. Sur la Figure 4.2a sont le rail de 1V sur le canal 1, le rail de 2.5V sur le canal 2, et le rail de 1.8V sur le canal 3. Sur la Figure 4.2b est une capture prise de la patte G10 de la puce, du signal de sortie "o_data_valid" changeant d'état lors d'une écriture dans un registre de paramètre. Comme démontré dans la figure, le niveau haut logique a une tension de 2.5V tandis que le niveau bas logique a une tension de 0V.



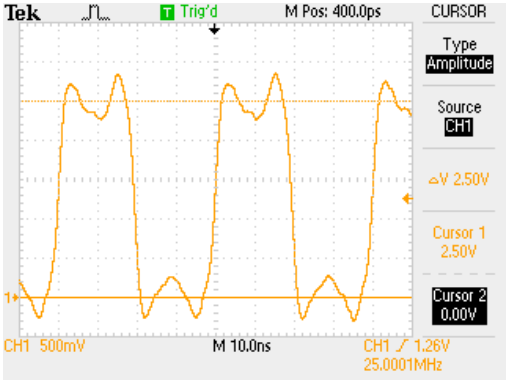
(a) Rails d'alimentation de la carte imprimée



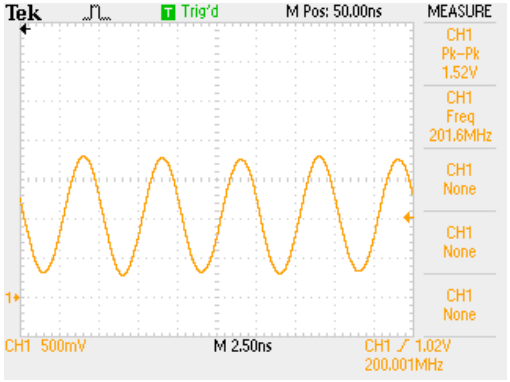
(b) Signal de sortie de l'ASIC

FIGURE 4.2 Mesures de tensions à l'oscilloscope

Les signaux d'horloge, de bus de données, de bus d'adresse, et de bus I²C ont été vérifiés de façon semblable. Ci-dessous sont des captures d'écran de l'oscilloscope mesurant les signaux d'horloge. Ces signaux n'ont pas une forme carrée idéale à cause des limitations matérielles de l'oscilloscope utilisé, qui a une fréquence de coupure de 200MHz. En conséquence, les harmoniques de l'horloge de 200MHz sont filtrés par l'oscilloscope, mais la fréquence fondamentale reste bien représentée.



(a) Horloge d'entrée-sortie (25MHz)



(b) Horloge du coeur (200MHz)

FIGURE 4.3 Mesures à l'oscilloscope des signaux d'horloge

4.2 Tests sur les composantes I²C

Les tests prévus pour les périphériques I²C présentés dans le Tableau 4.2. L'ensemble de tests T4 est présent pour vérifier le fonctionnement des composantes I²C.

TABLEAU 4.2 Liste des tests effectués sur les circuits I²C de la carte mezzanine

	Objectif de test	**Module visé**
T4.1	Changer et mesurer la fréquence de l'horloge programmable	Horloge programmable
T4.2	Lire les valeurs de tension aux bornes des résistances *current sense*	Capteur de courant

Le contrôleur AXI I²C du testeur FPGA est programmé à une fréquence de 200kHz pour ces tests, soit la moitié de la fréquence maximale du multiplexeur. L'intégrité du bus et le fonctionnement du contrôleur sont premièrement vérifiés en capturant à l'oscilloscope une transaction avec le capteur de courant du domaine de puissance 2 (U3) de la carte mezzanine, qui est à l'adresse 0xF8. Cette capture est illustrée dans la Figure 4.4. Cette opération permet simultanément de vérifier que l'initialisation du multiplexeur de la carte de développement se fait correctement.

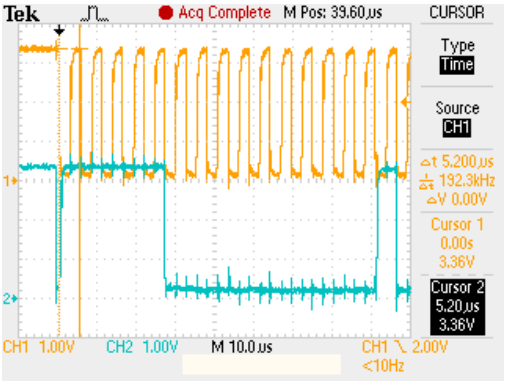


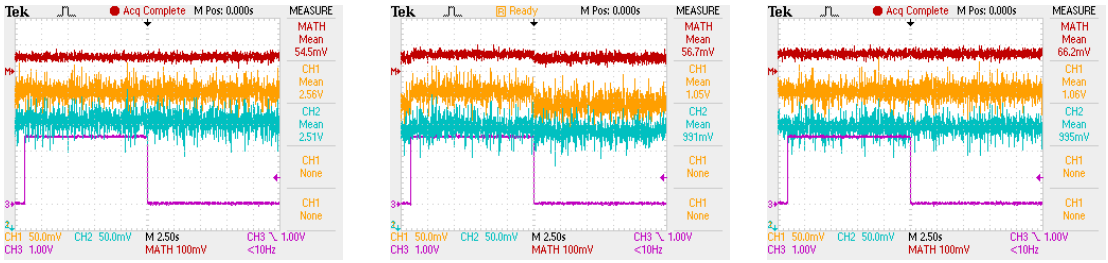
FIGURE 4.4 Transaction I²C

Dans cette figure sont les deux signaux composant un bus I²C : *serial clock* (SCL) sur le canal 1 et *serial data* (SDA) sur le canal 2. La transaction est initiée lorsqu'une transition de niveau haut à niveau bas sur SDA est détectée quand SCL est à niveau haut. L'état de SDA est ensuite capturé à chaque front montant de SCL. Les 7 bits d'adresse du capteur de courant, suivi d'un bit pour indiquer le type de transaction (0 pour une écriture, 1 pour une lecture) et d'un bit pour l'acquittement provenant du module esclave, sont capturés dans la figure.

Afin de comparer les données lues des capteurs de courant aux valeurs réelles, les tensions aux bornes des résistances de mesure de courant ont été mesurées à l'oscilloscope lorsque le signal `i_re` est activé. Toutefois, un biais significatif entre le canal 1 et le canal 2 de l'oscilloscope

est observé. Cette différence a été évaluée pour les trois rails de tension en mesurant avec les deux canaux la tension en amont de la résistance de mesure de courant. Cela est illustré dans la figure en annexe G, où les deux canaux et leur différence, capturée dans le canal mathématique, sont illustrés. Pour une mesure à 1V il y a un biais de 66mV et pour une mesure à 2.5V il y a un biais de 51mV.

Les mesures prises lors de l'activation du signal `i_re` sont illustrées dans la Figure 4.5, où le canal 1 est la borne négative de la résistance, le canal 2 est la borne positive de la résistance, le canal 3 est le signal `i_re` et le canal mathématique est la différence entre le canal 1 et le canal 2. Tel qu'attendu, le courant consommé par le domaine de puissance contenant les registres de paramètres (PD1) augmente lors de la lecture, comme le montre la Figure 4.5b. En augmentant l'échelle du canal mathématique, on y mesure une augmentation de 16mV (Fig. 4.6), ce qui correspond pour une résistance de 33mΩ à 484mA. À partir de ces figures, le courant consommé lorsque les rails de tension sont en repos (*idle*) peut aussi être calculé. Selon ces mesures, le rail de 2.5V consomme 103mA, le rail de 1V consomme 94mA, et le rail de $1V_{adj}$ consomme 15mA.



(a) R_{sense} 2.5V (b) R_{sense} 1V (c) R_{sense} $1V_{adj}$

FIGURE 4.5 Tensions aux bornes des résistances de mesure de courant

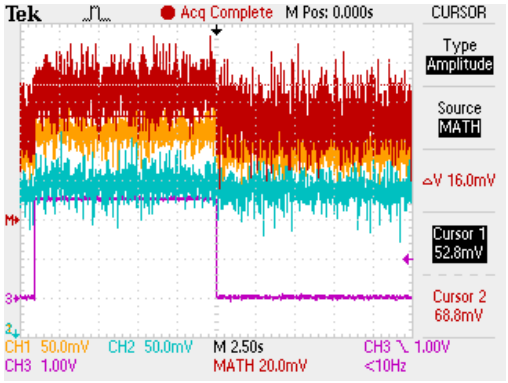


FIGURE 4.6 Courant consommé par PD1 lors d'une lecture

Les mesures prises par les capteurs de courant I^2C sont inscrites dans le Tableau 4.3. Une hausse de la consommation de courant est affichée lorsqu'une lecture est en cours, ce qui indique une certaine activité dans l'ASIC.

Tableau 4.3 Mesures de courant I^2C

Mesure	**Valeur reçue**	**Courant (mA)**
VIO au repos	0x0A6	67,5
VPD1 au repos	0x033	20,7
VPD2 au repos	0x041	26,4
VIO lecture des registres	0x0AA	69,2
VPD1 lecture des registres	0x03D	24,8
VPD2 lecture des registres	0x03F	25,6

4.3 Tests sur l'ASIC à conditions nominales

Les tests effectués sur l'ASIC afin de confirmer ses fonctionnalités et de mieux encadrer les bogues détectés seront détaillés dans cette section. La fonction principale de l'ASIC est d'effectuer le décodage de codes LDPC, mais afin d'accomplir cela, l'initialisation du contenu en mémoire doit fonctionner aussi. De façon à confirmer cela des tests à tension et fréquence nominale ont été exécutés. Cette section liste les fonctions qui sont couvertes lors des tests, et les résultats qui en découlent.

Les tests de l'ensemble T5 permettent de vérifier le fonctionnement de l'ASIC à tension et à fréquence nominale. Enfin, les tests T6 ont pour but d'observer le comportement du décodeur en mode quasi-synchrone.

Les tests T5 décrits dans le Tableau 4.4 ont pour objectif de déterminer si l'ASIC a le comportement attendu lors de l'exécution de la routine de test décrit dans la Section 3.1.2. La routine a échoué lors du test T5.1 : la relecture des registres de paramètre 1 et 2 de l'ASIC renvoie une valeur différente de celle qui y était écrite.

Une série de tests additionnels ciblant cette étape du test a été effectuée afin de tenter de déterminer la source de l'erreur.

Tableau 4.4 Liste des tests effectués sur l'ASIC du décodeur

	Objectif de test	**Module visé**
T5.1	Écrire et relire une valeur dans le banc de registre du décodeur	Décodeur
T5.2	Compléter une routine de décodage en utilisant les configurations de base du décodeur	Décodeur
T5.3	Mesurer la puissance consommée par l'ASIC lors d'une routine de décodage	Décodeur
T6.1	Affecter une valeur au bit d'activation du mode quasi-synchrone	Décodeur
T6.2	Compléter une routine de décodage en mode quasi-synchrone	Décodeur
T6.3	Mesurer la puissance consommée par l'ASIC à chaque cycle de décodage quasi-synchrone	Décodeur

4.4 Tests exploratoires

Dans cette section sont décrits les tests qui ont été effectués dans le but d'obtenir de l'information supplémentaire sur le comportement de la puce, en particulier par rapport aux opérations sur les registres de paramètres. Dans la table 4.5 est la liste des tests effectués.

Dans les tests T5.1a à T5.1e, les valeurs reçues du décodeur sont, à un bit ou deux près, de 0x761EFF3E pour le registre de paramètres 1 de 0x800703FB pour le registre de paramètres 2. Cette valeur correspond à la configuration décrite dans le Tableau 4.6. Ces tests ont été répétés sur une deuxième puce afin de confirmer que le comportement restait le même.

Comme le démontre le Tableau 4.6, les valeurs relues de l'ASIC ne concordent manifestement pas à la configuration voulue.

4.4.1 Signaux de contrôle

Afin d'approfondir la comparaison sur les résultats obtenus, des chronogrammes ont été capturés sur la plateforme FPGA du décodeur. Les figures suivantes soulignent les différences entre les chronogrammes obtenus de la plateforme FPGA à ceux obtenus de l'ASIC. Le chronogramme du FPGA se retrouve dans la partie supérieure de la figure tandis que celui de l'ASIC est dans la partie inférieure. Ces chronogrammes sont issus du protocole de test T5.1a.

Tableau 4.5 Liste des tests ciblant les registres de l'ASIC

	Description du test
T5.1a	Lecture des registres de paramètres après écriture
	Ce test est sensiblement la même que le test T5.1, et sert de base pour les tests suivants. La fréquence de l'horloge des entrées-sorties est fixée à 1MHz, et celle de l'horloge du coeur est fixée à 200MHz. Les valeurs 0x08023c20 et 0x00000019 sont écrites aux registres 1 et 2 respectivement, et une lecture est faite directement après. Aucune autre opération n'est effectuée sur l'ASIC.
T5.1b	Lecture des registres de paramètres après la mise sous tension de l'ASIC
T5.1c	Lecture des registres de paramètres après la réinitialisation de l'ASIC
T5.1d	Lecture des registres de paramètres après l'écriture avec une horloge ralentie
	La lecture des registres est faite après l'écriture des paramètres comme dans le test original, mais avec une horloge des entrées-sorties fixée à 250kHz et une horloge du coeur fixée à 1MHz.
T5.1e	Cycles lecture-écriture des registres de paramètres dans le cas d'échec de relecture
	La valeur des registres est retransférée à l'ASIC lorsque la valeur lue est erronée
T5.1f	Lecture des registres de paramètres après une écriture et une réinitialisation
	La valeur des registres est écrite. L'ASIC est ensuite réinitialisé, et les registres sont lus. Ce test a été ajouté suite à l'observation d'un comportement particulier : la valeur lue par le testeur était bonne si, après l'écriture des valeurs, le signal de réinitialisation était activé.
T5.1g	Lecture des registres de paramètres avec le signal de réinitialisation actif

Tel qu'illustré dans la Figure 4.7, les chronogrammes capturés lors de l'écriture des paramètres sont presque identiques. Par contre, dans la Figure 4.8, le signal `i_txn_req` est activé environ 6ms après l'écriture des paramètres dans l'ASIC, ce qui diffère du comportement du décodeur en simulation et dans le banc de test FPGA.

TABLEAU 4.6 Comparaison des valeurs de paramètres de l'ASIC

Paramètre	**Valeur écrite**	**Valeur lue**
Registre 1	0x08023C20	0x761EFF3E
Taille d'expansion du code	32	62
Nombre de lignes du protographe	60	255
Terminaison précoce du décodage	0	0
Nombre d'itérations précédant la terminaison précoce	1	15
Nombre de mots par noeud variable	8	54
Maintient de l'état post-décodage	0	1
Réinitialisation du décodeur	0	0
Registre 2	0x00000019	0x800703FB
Nombre maximal d'itérations	25	1019
Sélection de la vitesse d'horloge du décodeur	0	7
Sélection de la banque SRAM	0	1

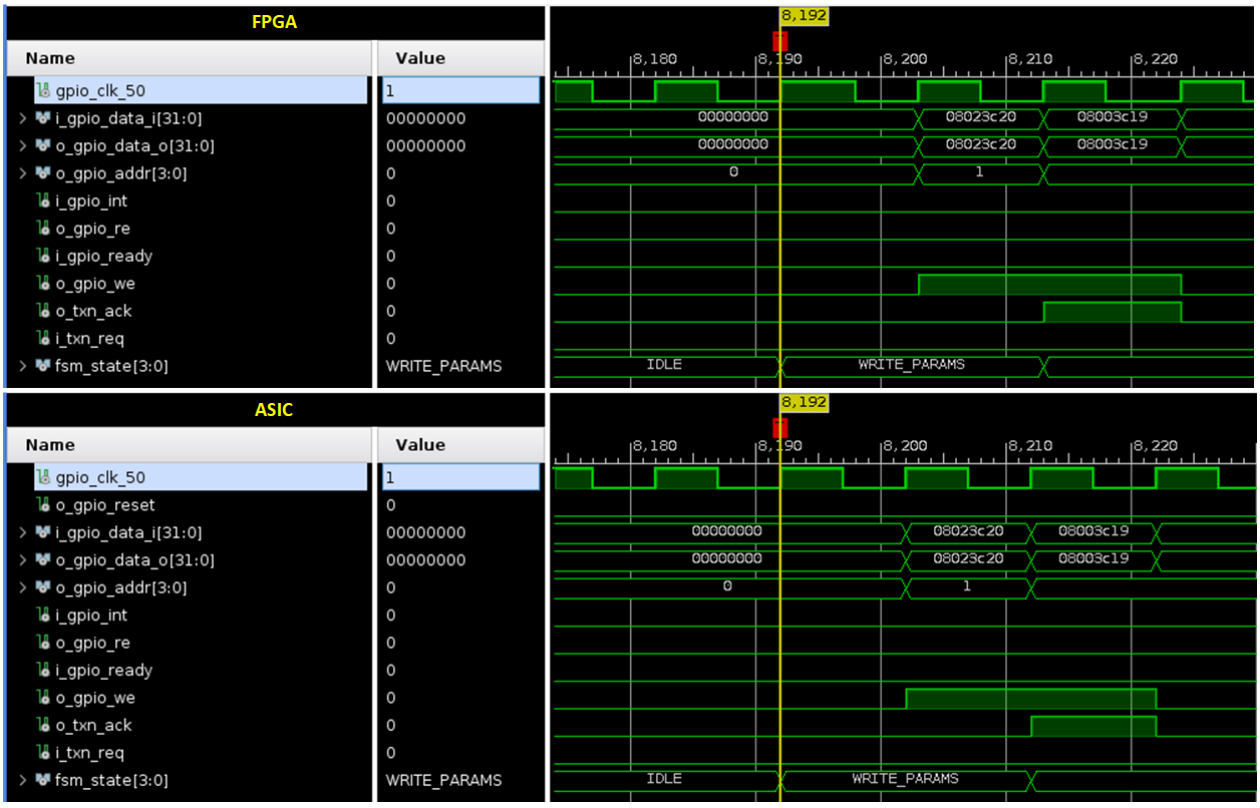


FIGURE 4.7 Comparaison des chronogrammes - Écriture de paramètres 1

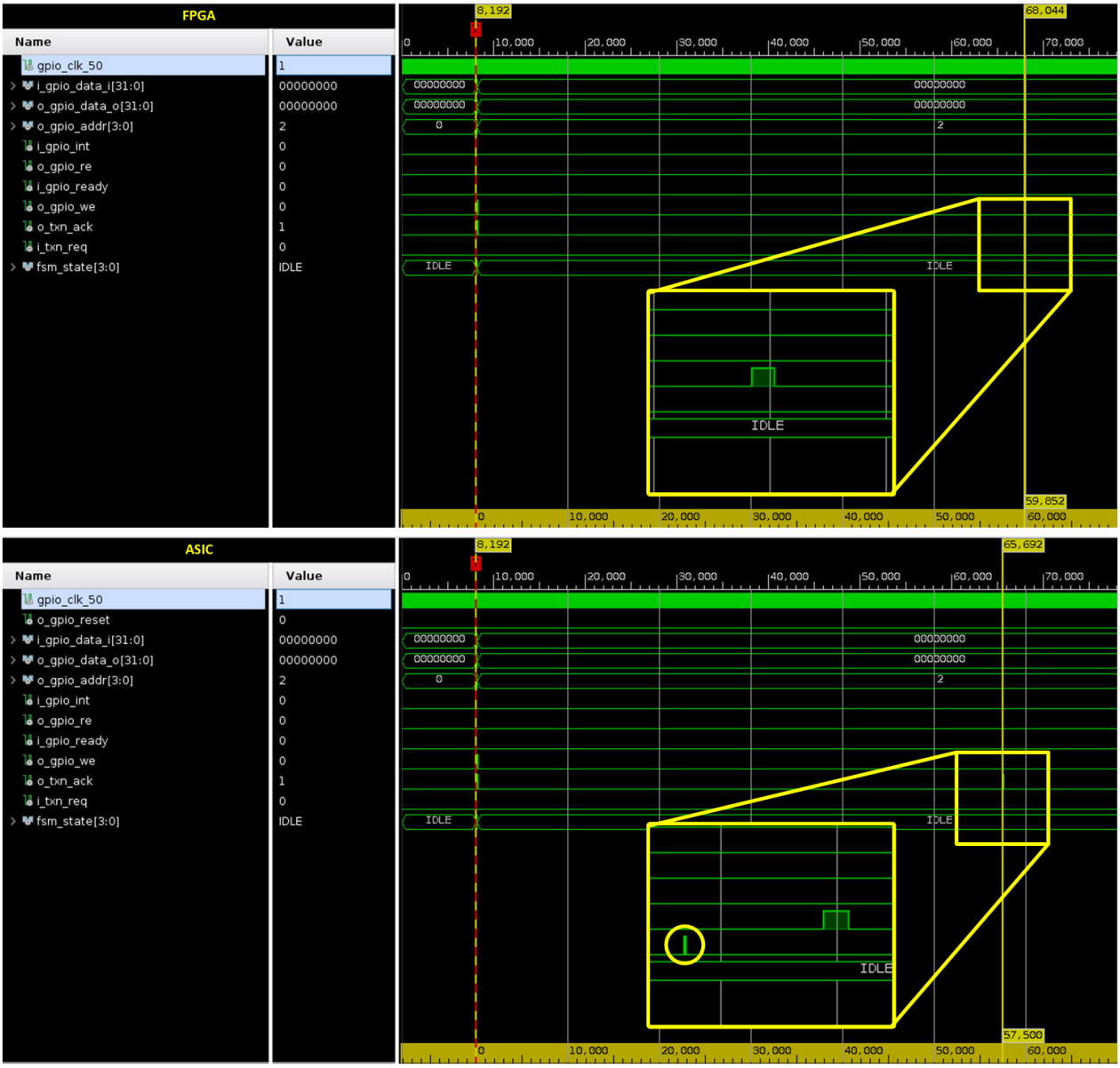


FIGURE 4.8 Comparaison des chronogrammes - Écriture de paramètres 2

La Figure 4.9 montre les signaux provenant de l'émulateur FPGA et de l'ASIC lors de la lecture du registre de paramètres. Lors de la lecture, les valeurs observées sur le bus de données de l'ASIC ne correspondent pas à ceux du FPGA. Dans le rectangle A, la valeur attendue est de 0x00000000 mais la valeur lue est 0x08003408. Dans le rectangle B, la valeur attendue est la valeur écrite dans le registre de paramètres 1 (0x08023C20) mais la valeur lue est 0x0000000. Enfin, dans le rectangle C la valeur attendue est la valeur écrite dans le registre de paramètres 2 (0x00000019) mais la valeur lue reste nulle.

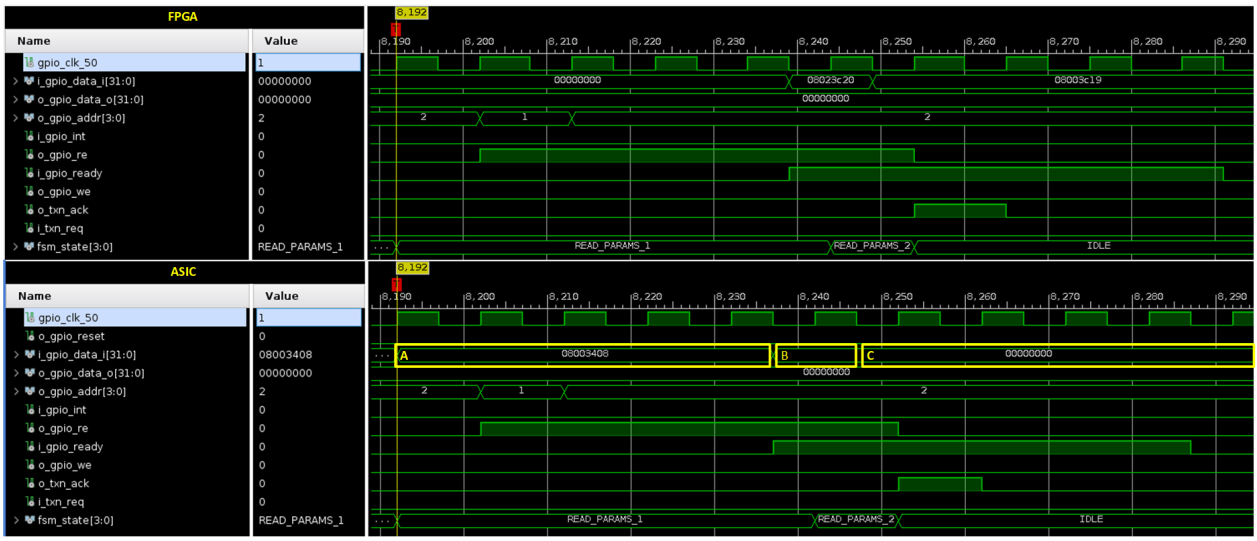


FIGURE 4.9 Comparaison des chronogrammes - Lecture de paramètres 1

Dans la Figure 4.10, le comportement opposé à l'écriture des paramètres est observé au niveau du signal `i_txn_req` lors de la lecture des paramètres. En simulation et avec l'émulateur FPGA, le signal `i_txn_req` est activé environ 7ms après la lecture des registres de paramètres. Sur l'ASIC par contre, ce signal reste inactif.

Les signaux de contrôles i_txn_req et o_txn_ack sont utilisés pour la coordination des transactions entre le banc de registre et le contrôleur des entrées-sorties du testeur FPGA. Dans les chronogrammes des Figures 4.7, 4.8, 4.9, et 4.10, ces signaux sont pris à l'interne du contrôleur des entrées-sorties. Ces deux modules sont séparés par des registres de synchronisation, car ils appartiennent à des domaines d'horloge différents.

Le signal `o_txn_ack` est mis à 1 à la fin d'une transaction avec le décodeur par le contrôleur des E/S, c'est-à-dire à chaque fois que l'état du contrôleur retourne à IDLE. Le signal `i_txn_req` est reçu par le contrôleur des entrées-sorties du banc de registre, et est dépendant du code d'opération 31 (Requête de transmission) du testeur FPGA.

Le bloc utilisé pour synchroniser un domaine d'horloge à l'autre est le XPM CDC (*Xilinx Parameterized Macro Clock Domain Crossing*) avec un signal de validation de transfert (*handshake*) [39]. Dans la Figure 4.11 est illustré le diagramme de ce bloc indiquant ses entrées et ses sorties.

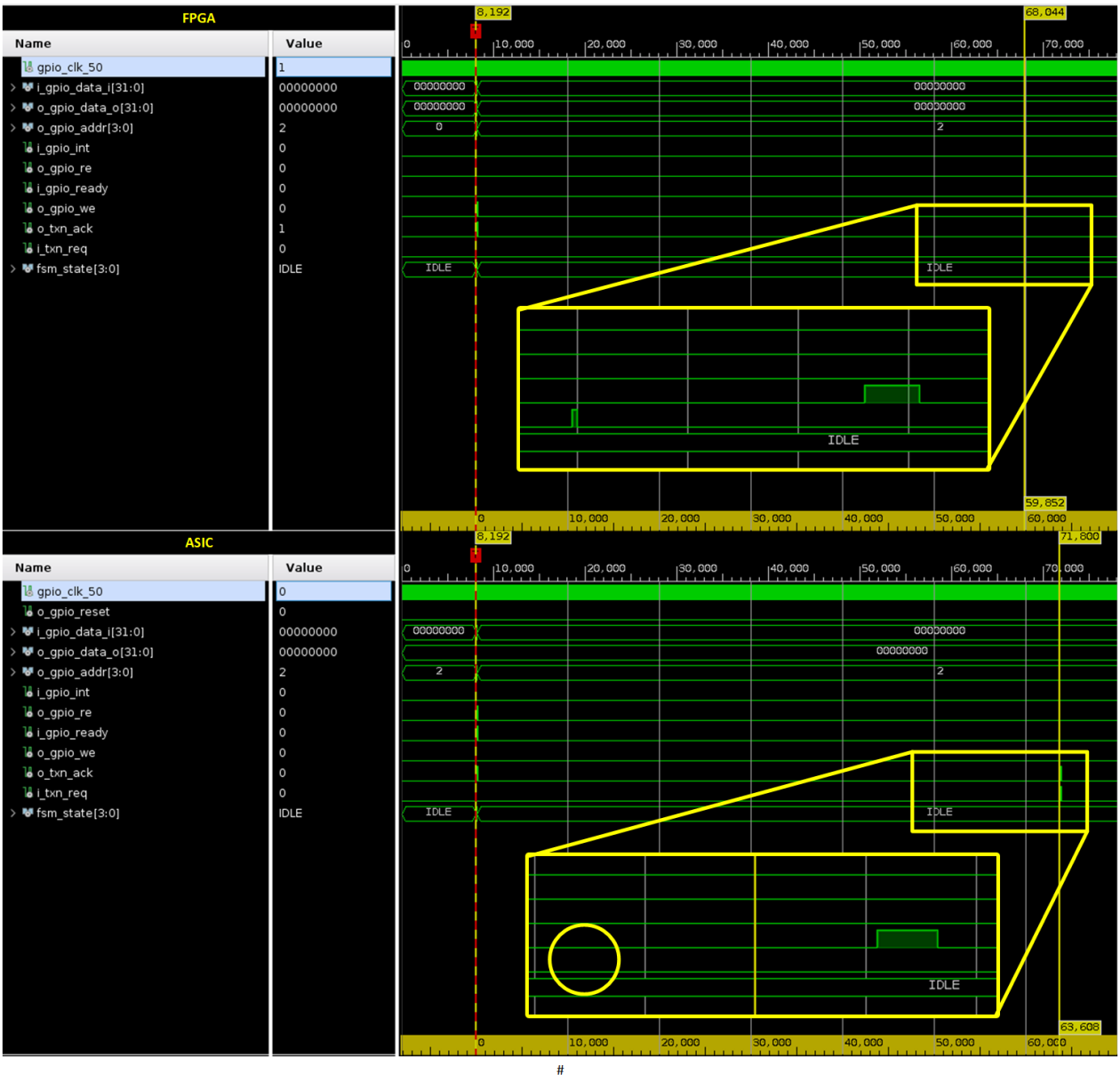


FIGURE 4.10 Comparaison des chronogrammes - Lecture de paramètres 2

Le fonctionnement de ce bloc est décrit par le chronogramme présenté dans la Figure 4.12. Lorsqu'un transfert est disponible, la source active le signal `src_send`. Le destinataire envoie alors une requête avec le signal `dest_req`, et ensuite lit les données transférées. Le transfert est clos une fois que le signal `src_rcv` est activé, puis rabaissé une fois que le signal `src_send` retourne à un état inactif.

Dans le testeur FPGA, l'entrée `src_send` du bloc CDC du contrôleur des entrées-sorties est liée au signal `i_txn_req` tandis que le signal `o_txn_ack` est lié à l'entrée `src_send` du bloc

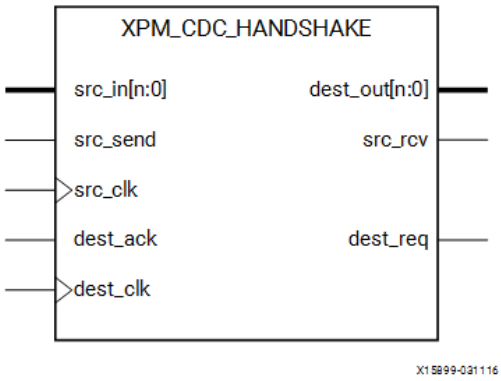


FIGURE 4.11 Diagramme du bloc XPM CDC

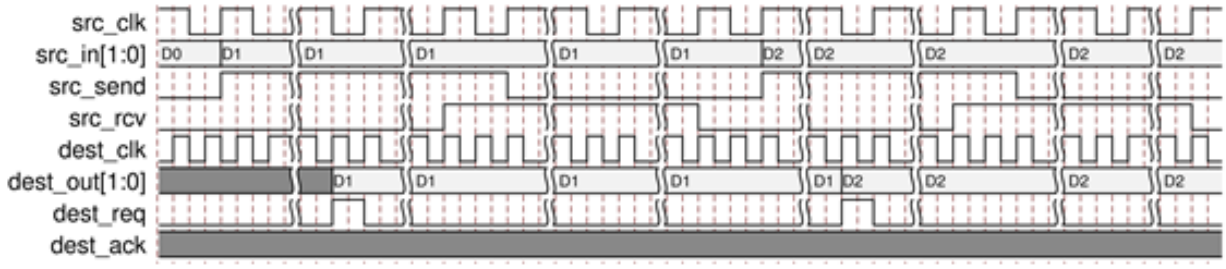


FIGURE 4.12 Chronogramme du bloc XPM CDC

CDC du banc de registre. Un signal intermédiaire sépare ces signaux de contrôle aux entrées des blocs CDC pour y appliquer des opérations logiques. Dans le banc de registre comme dans le contrôleur des entrées-sorties, ce signal porte le nom `txn_data_sync_req`.

4.4.2 Réinitialisation

Ces tests ont permis de révéler que le système de réinitialisation ne semble pas fonctionner tel qu'attendu. Ceci est rendu évident lorsqu'une lecture est faite sur un registre de paramètre suite à un cycle de réinitialisation, tel que fait dans le test T5.1c. Selon la description du banc de test de l'ASIC, la valeur attendue de cette lecture est de 0x00000000 pour ces deux registres suite à cette opération, mais la valeur obtenue est de 0x761EFF3E pour le registre de paramètres 1 de 0x800703FB pour le registre de paramètres 2. Un autre comportement observé est que le registre de paramètre 2 maintient sa valeur en lecture même lorsque le signal de réinitialisation est actif.

L'ASIC a pour particularité de retourner une donnée valide après une réinitialisation : après qu'une écriture soit faite sur un registre, puis que le signal de réinitialisation est basculé, la donnée lue sur ce registre est la donnée qui y a été écrite. Il semble qu'il y ait un problème provenant du système de réinitialisation. Le schéma 4.13 suit le parcours du signal de réinitialisation à travers l'ASIC.

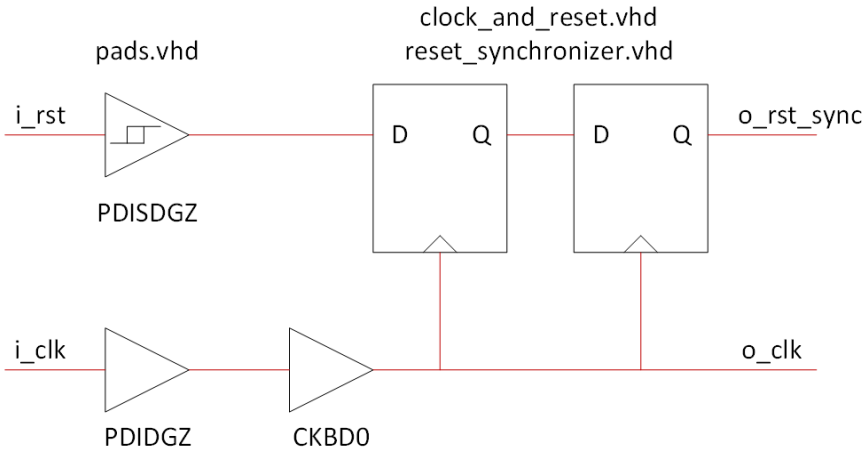


FIGURE 4.13 Chemin du signal de réinitialisation de l'ASIC

La patte de réinitialisation est premièrement isolée par un circuit tampon (*buffer*) du reste de l'ASIC. Le circuit tampon provient de la librairie pourvue par le manufacturier (TSMC) et est associé au code de référence PDISDGZ. Ce circuit tampon contient aussi une bascule de Schmitt pour assurer la stabilité du signal. Une fois que le signal externe a traversé le circuit tampon, il est synchronisé à l'horloge du coeur. Le signal est ensuite transmis aux deux domaines de puissance de l'ASIC, puis distribué aux différents modules présents dans chacun. Le signal de réinitialisation est actif à un niveau haut.

Le circuit d'horloge est l'une des différences principales entre l'ASIC et l'émulateur. Dans l'émulateur, un gestionnaire d'horloge configurable est utilisé pour générer des signaux à différentes fréquences. Ces signaux sont ensuite acheminés à un multiplexeur contrôlé par le paramètre "Sélection de la vitesse d'horloge du décodeur" (bits 18 à 16 du registre de paramètres 2). Dans l'ASIC, les signaux d'horloge externe `i_clk` sont simplement isolés par les circuits tampons PDIDGZ puis CKBD0. La cellule PDIDGZ omet la bascule de Schmitt à cause de leur limitation fréquentielle.

Ce problème semble être la source la plus probable des erreurs encourues lors du test de l'ASIC.

4.4.3 Décalage des tableaux

Un deuxième comportement particulier qui a été détecté se rapporte à la relecture des tables, où il semble y avoir un décalage lors de l'opération. La réinitialisation de l'ASIC suite à l'écriture des tables permet d'accéder à des valeurs partiellement correctes lors de la relecture, comme c'est le cas pour les registres de paramètre. Ces valeurs sont toutefois décalées par rapport à leur position attendue. La Figure 4.14 illustre ce décalage dans la mémoire du testeur FPGA suite à une relecture. Dans cette mémoire, les différentes valeurs écrites et lues sont transposées en une couleur associée à son code hexadécimal en tronquant les 8 bits les plus significatifs. Cela permet de percevoir un décalage de plus ou moins une position parmi les données sélectionnées. La valeur du paramètre 1 (0x08023c20) est présente à l'adresse 0x00004138, mais ne fait pas partie de l'ensemble des valeurs écrites dans les tables. Ce problème pourrait être relié à la remise à zéro du compteur d'écriture ou de lecture des tables.

RESULT	ADDRESS	READ	WRITE	RESULT	ADDRESS	READ	WRITE	RESULT	ADDRESS	READ	WRITE
ERROR	0x00004118	#00001A	#168080	CORRECT	0x0000413B	#000000	#000000	CORRECT	0x0000415E	#000000	#000000
ERROR	0x00004119	#01018D	#00001A	CORRECT	0x0000413C	#000000	#000000	CORRECT	0x0000415F	#000000	#000000
ERROR	0x0000411A	#000000	#01018D	CORRECT	0x0000413D	#000000	#000000	CORRECT	0x00004160	#000000	#000000
ERROR	0x0000411B	#C00007	#000000	CORRECT	0x0000413E	#000000	#000000	CORRECT	0x00004161	#000000	#000000
ERROR	0x0000411C	#3D8000	#C00007	CORRECT	0x0000413F	#000000	#000000	CORRECT	0x00004162	#000000	#000000
ERROR	0x0000411D	#000000	#3D8000	CORRECT	0x00004140	#000000	#000000	CORRECT	0x00004163	#000000	#000000
ERROR	0x0000411E	#0002C7	#000000	CORRECT	0x00004141	#000000	#000000	ERROR	0x00004164	#000000	#000200
ERROR	0x0000411F	#D00000	#0002C7	CORRECT	0x00004142	#000000	#000000	ERROR	0x00004165	#000200	#000010
ERROR	0x00004120	#0001C0	#D00000	CORRECT	0x00004143	#000000	#000000	ERROR	0x00004166	#000010	#000000
ERROR	0x00004121	#00601D	#0001C0	CORRECT	0x00004144	#000000	#000000	CORRECT	0x00004167	#000000	#000000
ERROR	0x00004122	#000000	#00601D	CORRECT	0x00004145	#000000	#000000	ERROR	0x00004168	#000000	#000000
ERROR	0x00004123	#CABC02	#000000	CORRECT	0x00004146	#000000	#000000	ERROR	0x00004169	#000000	#000000
ERROR	0x00004124	#06A3C0	#CABC02	CORRECT	0x00004147	#000000	#000000	CORRECT	0x0000416A	#000000	#000000
ERROR	0x00004125	#E23406	#06A3C0	CORRECT	0x00004148	#000000	#000000	CORRECT	0x0000416B	#000000	#000000
ERROR	0x00004126	#0E03E0	#E23406	CORRECT	0x00004149	#000000	#000000	CORRECT	0x0000416C	#000000	#000000
ERROR	0x00004127	#000000	#0E03E0	CORRECT	0x0000414A	#000000	#000000	ERROR	0x0000416D	#000000	#000040
ERROR	0x00004128	#F065EA	#000000	CORRECT	0x0000414B	#000000	#000000	ERROR	0x0000416E	#000040	#000000
ERROR	0x00004129	#7FC4BD	#F065EA	CORRECT	0x0000414C	#000000	#000000	ERROR	0x0000416F	#000000	#000400
ERROR	0x0000412A	#0085C0	#7FC4BD	CORRECT	0x0000414D	#000000	#000000	ERROR	0x00004170	#000400	#000000
ERROR	0x0000412B	#00576D	#0085C0	CORRECT	0x0000414E	#000000	#000000	CORRECT	0x00004171	#000000	#000000
ERROR	0x0000412C	#000000	#00576D	CORRECT	0x0000414F	#000000	#000000	ERROR	0x00004172	#000000	#000000
ERROR	0x0000412D	#0F88BA	#000000	CORRECT	0x00004150	#000000	#000000	ERROR	0x00004173	#000000	#000000
ERROR	0x0000412E	#5D800A	#0F88BA	CORRECT	0x00004151	#000000	#000000	CORRECT	0x00004174	#000000	#000000
ERROR	0x0000412F	#71813F	#5D800A	CORRECT	0x00004152	#000000	#000000	ERROR	0x00004175	#000000	#040000
ERROR	0x00004130	#1FBC7E	#71813F	CORRECT	0x00004153	#000000	#000000	ERROR	0x00004176	#040000	#000000
ERROR	0x00004131	#000000	#1FBC7E	CORRECT	0x00004154	#000000	#000000	ERROR	0x00004177	#000000	#400000
ERROR	0x00004132	#4A0339	#000000	CORRECT	0x00004155	#000000	#000000	ERROR	0x00004178	#400000	#000000
ERROR	0x00004133	#804C76	#4A0339	CORRECT	0x00004156	#000000	#000000	CORRECT	0x00004179	#000000	#000000
ERROR	0x00004134	#004F76	#804C76	CORRECT	0x00004157	#000000	#000000	CORRECT	0x0000417A	#000000	#000000
ERROR	0x00004135	#164014	#004F76	CORRECT	0x00004158	#000000	#000000	ERROR	0x0000417B	#000000	#200000
ERROR	0x00004136	#000000	#164014	CORRECT	0x00004159	#000000	#000000	ERROR	0x0000417C	#200000	#000000
ERROR	0x00004137	#164014	#000000	CORRECT	0x0000415A	#000000	#000000	CORRECT	0x0000417D	#000000	#000000
ERROR	0x00004138	#023C20	#000000	CORRECT	0x0000415B	#000000	#000000	ERROR	0x0000417E	#000000	#000080
CORRECT	0x00004139	#000000	#000000	CORRECT	0x0000415C	#000000	#000000	ERROR	0x0000417F	#000080	#000000
CORRECT	0x0000413A	#000000	#000000	CORRECT	0x0000415D	#000000	#000000	CORRECT	0x00004180	#000000	#000000

FIGURE 4.14 Décalage de la relecture des tables

L'écriture et la lecture des données dans les tableaux se font avec des pointeurs internes, qui sont incrémentés à chaque accès à la mémoire. Les codes d'opérations 0 à 5 sont utilisés pour remettre à zéro ces pointeurs.

Lorsque l'un de ces codes d'opération est envoyé à l'ASIC, il est transmis au banc de registres, qui par la suite remet à zéro le pointeur indiqué.

La table des adresses et la table des décalages de l'ASIC utilisent le même module générique décrit dans le fichier `Tables.vhd`, illustré dans la Figure 4.15. Ce module a une entrée nommée `resetCnt` utilisée pour remettre à zéro le pointeur d'adresse de la mémoire. Les trois autres tables, soit la table de recherche des partitions, la table des activations et la table des biais, utilisent des processus locaux au banc de registres.

Le signal `resetCWcounter` lie le banc de registres à l'unité de contrôle du décodeur, qui contient la mémoire des mots de codes (`CWMem`). Ce signal remet à zéro le compteur *ioWordCnt*, le pointeur d'adresses de la mémoire des mots de codes.

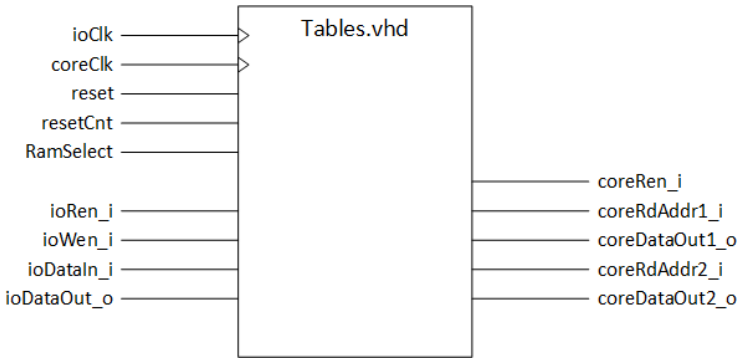


FIGURE 4.15 Module Tables.vhd

Il serait improbable de faire une lecture dans le registre des paramètres en accédant aux adresses réservées aux tables. Une hypothèse serait que le code d'opération envoyé par le testeur FPGA est erroné.

CHAPITRE 5 CONCLUSION

Dans la partie finale de ce document, une discussion portant sur les limitations du système conçu et des améliorations qui pourraient y être apportées sera faite. Un résumé du travail accompli dans le cadre de ce projet précédera cette analyse.

5.1 Synthèse des travaux

Au cours de ce projet de recherche, un système automatisé pour le test d'un ASIC contenant un décodeur LDPC a été créé. Ce système comprend une portion logicielle, qui simule le comportement du décodeur et du banc de test qui le contrôle, ainsi qu'une portion matérielle. La portion matérielle comprend principalement une carte imprimée pour supporter l'ASIC ainsi permettant son utilisation, le contrôle de sa fréquence d'horloge et de sa tension, et la mesure de sa consommation énergétique. Cette carte est destinée à être utilisée de pair avec une carte de développement FPGA VC707.

Avec cette infrastructure de test, la validation de l'ASIC est possible. Une variété de comportements inusités ont été observés au cours du test de la puce. Ils ont entre autres révélé un problème qui semble provenir du mécanisme de remise à zéro de la puce.

5.2 Limitations de la solution proposée

Avec le système présent, l'automatisation des tests pose toujours un certain défi.

Au niveau du logiciel de test, l'exécution de routines de tests nécessite préalablement la simulation et possiblement la synthèse de la plateforme désirée. Le temps de compilation nécessaire pour compléter ce processus peut parfois atteindre une heure, ce qui introduit un délai significatif aux tests. La stabilité des tests est potentiellement un autre élément à améliorer. Par exemple, lorsque la condition de lancement de capture des sondes ILA n'est pas atteinte, le programme de test reste piégé dans une boucle infinie. Dans sa forme actuelle, le testeur FPGA ne peut pas traiter efficacement des communications provenant de différentes sources ou faire usage d'interruptions pour mieux ordonnancer ses instructions.

La carte de support bénéficierait elle aussi d'une révision. Le circuit permettant de varier la tension reste non fonctionnel dans l'état où il est en ce moment. Les circuits de mesure de courant permettent d'obtenir des données générales sur la consommation énergétique de la puce, mais ces résultats sont limités dû à la précision et au délai de conversion de ceux-ci.

Il est aussi difficile de prendre des mesures à l'oscilloscope ou de faire des modifications au circuit. Pour l'assemblage de la carte, il manque de marqueurs et d'ancrages pour aligner le pochoir de la pâte de soudage. De plus, le connecteur PGA 11x11 est en fin de vie et ne sera plus disponible à partir de juin 2022, et le translateur de tension 32 bits a maintenant une nouvelle empreinte physique.

Finalement, l'état interne de la puce ASIC est difficile d'accès, ce qui rend complexe son débogage.

5.3 Améliorations futures

Le système conçu comporte des lacunes qui peuvent être corrigées. Pour conclure ce document, des recommandations seront émises pour améliorer les résultats obtenus dans une itération future du projet.

L'ASIC est le premier point qui devrait être abordé. Il serait utile d'y implémenter un registre de statut pour avoir davantage d'information sur le décodeur. Les signaux d'horloge devraient avoir une interface différentielle pour améliorer leur résistance au bruit. Pour réduire la complexité du système, sélectionner une technologie avec des tensions adaptées à celles de la carte de développement serait avantageux. Intégrer le protocole JTAG à l'ASIC serait un choix judicieux, car cela permettrait d'automatiser davantage les tests matériels et les tests unitaires. Une nouvelle empreinte physique pour la puce devrait être considérée vu que la production du support à puce utilisé a cessé.

Pour la carte de support, il faudra réviser le circuit pour varier la tension. Cette fonction s'est avérée non fonctionnelle et une modification au circuit de contrôle ou le remplacement de la puce de régulation de la tension pourrait être nécessaire. L'ajout de points de test pour des nœuds d'intérêt serait essentiel pour rendre la prise de mesure sur la carte plus accessible. Des marqueurs et des ancrages devraient être ajoutés pour l'alignement du pochoir de la pâte à soudure. D'autres détails à changer, portant en particulier sur le placement des trous de montage, la taille de la sérigraphie, ou sur le schéma de conception de la carte, sont listés dans le fichier `hw/pcb/README.md` du répertoire du projet.

Auprès du testeur, l'ajout d'un micro-contrôleur pour le contrôle du décodeur serait un choix envisageable. Cela permettrait de simplifier l'automatisation des routines de test et d'avoir une plus grande flexibilité. L'accès à des librairies de pilotes, la gestion des interruptions matérielles, le chronométrage des opérations et l'utilisation d'outils de déboguage intégrés sont des avantages liés à l'ajout d'un micro-contrôleur dans le testeur FPGA.

RÉFÉRENCES

[1] IEA, IRENA, UNSD, World Bank et WHO, "Tracking SDG 7 : The Energy Progress Report," World Bank, Washington D.C., Rapport technique, 2022. [En ligne]. Disponible : https://trackingsdg7.esmap.org/data/files/download-documents/sdg7-report2022-full_report.pdf

[2] IEA, "Data Centres and Data Transmission Networks," IEA, Paris, Rapport technique, 2021. [En ligne]. Disponible : https://www.iea.org/reports/data-centres-and-data-transmission-networks

[3] I. F. Akyildiz, A. Kak et S. Nie, "6G and Beyond : The Future of Wireless Communications Systems," *IEEE Access*, vol. 8, p. 133 995–134 030, 2020, conference Name : IEEE Access.

[4] C.-L. I, S. Han et S. Bian, "Energy-efficient 5G for a greener future," *Nature Electronics*, vol. 3, n°. 4, p. 182–184, avr. 2020, number : 4 Publisher : Nature Publishing Group. [En ligne]. Disponible : https://www.nature.com/articles/s41928-020-0404-1

[5] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, n°. 8, p. 4, 1965.

[6] J. Koomey et S. Naffziger, "Moore's Law Might Be Slowing Down, But Not Energy Efficiency," *IEEE Spectrum*, mars 2015, section : Computing. [En ligne]. Disponible : https://spectrum.ieee.org/moores-law-might-be-slowing-down-but-not-energy-efficiency

[7] T. S. Perry, "Move Over, Moore's Law : Make Way for Huang's Law," *IEEE Spectrum*, avr. 2018, section : Computing. [En ligne]. Disponible : https://spectrum.ieee.org/move-over-moores-law-make-way-for-huangs-law

[8] D. Mamaluy et X. Gao, "The fundamental downscaling limit of field effect transistors," *Applied Physics Letters*, vol. 106, n°. 19, p. 193503, mai 2015, publisher : American Institute of Physics. [En ligne]. Disponible : https://aip.scitation.org/doi/10.1063/1.4919871

[9] M. B. Taylor, "Is dark silicon useful ? Harnessing the four horsemen of the coming dark silicon apocalypse," dans *DAC Design Automation Conference 2012*, juin 2012, p. 1131–1136, iSSN : 0738-100X.

[10] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester et T. Mudge, "Near-Threshold Computing : Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, n°. 2, p. 253–266, févr. 2010, conference Name : Proceedings of the IEEE.

[11] S.-Y. Chung, J. David Forney, T. J. Richardson et R. Urbanke, “On the design of low-density parity-check codes within 0.0045dB of the Shannon limit,” *IEEE Commun. Lett.*, vol. 5, n°. 2, p. 58–60, Feb. 2001.

[12] E. Berlekamp, R. McEliece et H. van Tilborg, “On the inherent intractability of certain coding problems,” *IEEE Transactions on Information Theory*, vol. 24, n°. 3, p. 384–386, mai 1978, conference Name : IEEE Transactions on Information Theory.

[13] J. Nadal, M. Fiorentino, E. Dupraz et F. Leduc-Primeau, “A Deeply Pipelined, Highly Parallel and Flexible LDPC Decoder,” dans *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*, juin 2020, p. 263–266.

[14] CMC Microsystems, “Logiciel Cadence Allegro/OrCAD.” [En ligne]. Disponible : https://www.cmc.ca/fr/logiciel-cadence-allegro-orcad/

[15] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, n°. 3, p. 379–423, juill. 1948, conference Name : The Bell System Technical Journal.

[16] R. G. Gallager, “Low-Density Parity-Check Codes,” *IRE Transactions on Information Theory*, p. 8, 1962.

[17] D. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Transactions on Information Theory*, vol. 45, n°. 2, p. 399–431, mars 1999, conference Name : IEEE Transactions on Information Theory.

[18] W. E. Ryan et S. Lin, *Channel codes : classical and modern.* Cambridge ; New York : Cambridge University Press, 2009, oCLC : ocn401147470.

[19] R. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, n°. 5, p. 533–547, sept. 1981, conference Name : IEEE Transactions on Information Theory.

[20] J. H. Bae, A. Abotabl, H.-P. Lin, K.-B. Song et J. Lee, “An overview of channel coding for 5G NR cellular communications,” *APSIPA Transactions on Signal and Information Processing*, vol. 8, n°. 1, 2019. [En ligne]. Disponible : http://www.nowpublishers.com/article/Details/SIP-122

[21] M. Fossorier, “Quasicyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Transactions on Information Theory*, vol. 50, n°. 8, p. 1788–1793, août 2004, conference Name : IEEE Transactions on Information Theory.

[22] J. Hagenauer, E. Offer et L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Transactions on Information Theory*, vol. 42, n°. 2, p. 429–445, 1996.

[23] R. Jose et A. Pe, “Analysis of hard decision and soft decision decoding algorithms of LDPC codes in AWGN,” dans *2015 IEEE International Advance Computing Conference (IACC)*, juin 2015, p. 430–435.

[24] F. R. Kschischang, B. J. Frey et H.-A. Loeliger, “Factor Graphs and the Sum-Product Algorithm,” *IEEE Transactions on Information Theory*, vol. 47, p. 498–519, 1998.

[25] E. Eleftheriou et A. Dholakia, “Efficient implementations of the sum-product algorithm for decoding of LDPC codes,” dans *in Proc. IEEE Globecom 2001*, 2001.

[26] E. Sharon, S. Litsyn et J. Goldberger, “Convergence analysis of serial message-passing schedules for LDPC decoding,” dans *Turbo Codes and Related Topics ; 6th International ITG-Conference on Source and Channel Coding (TURBOCODING), 2006 4th International Symposium on*. Institute of Electrical and Electronics Engineers Inc., 2006. [En ligne]. Disponible : https://cris.tau.ac.il/en/publications/convergence-analysis-of-serial-message-passing-schedules-for-ldpc

[27] E. Dupraz, F. Leduc-Primeau et F. Gagnon, “Low-Latency LDPC Decoding Achieved by Code and Architecture Co-Design,” dans *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, déc. 2018, p. 1–5, iSSN : 2165-4719.

[28] L. Schmalen, “Offset Min-Sum decoding of LDPC codes,” EP Brevet EP2 892 157A1, juill. 2015. [En ligne]. Disponible : https://patents.google.com/patent/EP2892157A1/en

[29] K. Forsberg et H. Mooz, “The Relationship of System Engineering to the Project Cycle,” *INCOSE International Symposium*, vol. 1, n°. 1, p. 57–65, 1991, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.1991.tb01484.x. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.1991.tb01484.x

[30] R. Khan, A. K. Srivastava et D. Pandey, “Agile approach for Software Testing process,” dans *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, nov. 2016, p. 3–6.

[31] G. Swamy, “Formal verification of digital systems,” dans *Proceedings Tenth International Conference on VLSI Design*, janv. 1997, p. 213–217, iSSN : 1063-9667.

[32] L.-T. Wang, C.-W. Wu et X. Wen, édit., *VLSI test principles and architectures : design for testability*, ser. The Morgan Kaufmann series in systems on silicon. Amsterdam ; Boston : Elsevier Morgan Kaufmann Publishers, 2006, oCLC : ocm64624834.

[33] D. Biederman, “An overview on writing a VHDL testbench,” dans *Proceedings The Twenty-Ninth Southeastern Symposium on System Theory*, mars 1997, p. 384–388, iSSN : 0094-2898.

[34] L. Goldstein, “Controllability/observability analysis of digital circuits,” *IEEE Transactions on Circuits and Systems*, vol. 26, n°. 9, p. 685–693, sept. 1979, conference Name : IEEE Transactions on Circuits and Systems.

[35] Texas Instruments, “IEEE Std 1149.1 (JTAG) Testability Primer,” 1997. [En ligne]. Disponible : https://www.ti.com/lit/an/ssya002c/ssya002c.pdf

[36] Xilinx, “VC707 Evaluation Board for the Virtex-7 FPGA User Guide,” 2019. [En ligne]. Disponible : https://www.xilinx.com/support/documents/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf

[37] Xilinx, “LogiCORE IP AXI IIC Bus Interface v1.02a,” 2012. [En ligne]. Disponible : https://docs.xilinx.com/api/khub/documents/AtgxlWBVjH1a43Hij30Hfw/content?Ft-Calling-App=ft%2Fturnkey-portal&Ft-Calling-App-Version=3.11.43&filename=pg090-axi-iic.pdf

[38] Xilinx, “Integrated Logic Analyzer v6.2,” 2016. [En ligne]. Disponible : https://docs.xilinx.com/api/khub/documents/w1BckBqYd0w2WsdXTPu1og/content?Ft-Calling-App=ft%2Fturnkey-portal&Ft-Calling-App-Version=3.11.43&filename=pg172-ila.pdf

[39] Xilinx, “XPM CDC Generator v1.0 LogiCORE IP Product Guide,” févr. 2021. [En ligne]. Disponible : https://docs.xilinx.com/r/en-US/pg382-xpm-cdc-generator

ANNEXE A LISTE DES MATÉRIAUX DE LA CARTE MEZZANINE

Tableau A.1 Liste des matériaux de la carte mezzanine

Composante	**Qté.**	**Réf.**	**Valeurs**	**Numéro d'article**
Condensateur	4	C1 C2 C11 C12	47uF, X5R	JMK212BBJ476MG-T
Condensateur	4	C3 C4 C5 C6	10uF, X5R	CL21A106KOQNNNE
Condensateur	4	C7 C8 C9 C10	10nF, X5R	QMK212BJ103KG-T
Condensateur	3	C13 C14 C18	0.1nF	08056D104KAT2A
Condensateur	2	C15 C17	4.7uF	CL21A475KPCLNNC
Condensateur	1	C16	1nF	CL21B102KBANFNC
Condensateur	11	C19 C21 C23 C25 C26 C27 C28 C29 C30 C31 C32	0.1uF	TMK107BJ104KA
Condensateur	3	C20 C22 C24	10nF	06035C103M4T2A
Diode Schottky	1	D1		RB471ET148
Connecteur FMC-HPC Mâle	1	J1		ASP-134488-01
Header Pins	9	J2 J3 J4		
Resistance	3	R1 R2 R6	100KΩ	CRCW0805100KJNEA
Resistance	1	R3	10Ω	ERJ-6GEYJ200V
Resistance	3	R4 R5 R7	0.033Ω, 1%	KRL1220E-M-R033-F-T5
Régulateur de tension 1V ajustable	2	U1 U2		TPS7A8401ARGRR
Capteur de courant I2C	3	U3 U4 U6		MAX9611AUBT
Régulateur de tension 2.5V	1	U5		ADP1740ACPZ-2P5-R7
Horloge programmable	1	U7		570FAB000544DG-ND
Support de CI 11x11 PGA	1	U8		3M5067-ND
Broches à insérer	86	U8		3M5074-ND
Translateur de tension 32 bits	2	U9 U10		SN74AVC32T245ZKER
Translateur de tension I2C	1	U11		TCA9406DCTR
Pâte à soudure	1			TS391AX50

ANNEXE B ATTRIBUTION DES PATTES DE LA PUCE ASIC

TABLEAU B.1 Attribution des pattes de la puce ASIC

Signal	***Die***	**Boitier**	**Signal**	***Die***	**Boitier**	**Signal**	***Die***	**Boitier**
i_addr[0]	62	C10	io_data[20]	2	C2	VDD_PD2	6	D1
i_addr[1]	61	B11	io_data[21]	83	B3	VDD_PD2	11	F3
i_addr[2]	60	C11	io_data[22]	82	A2	VDD_PD2	53	F9
i_addr[3]	57	E9	io_data[23]	79	A4	VDD_PD2	58	D11
i_io_clk	46	J11	io_data[24]	78	C5	VDD_PD2	72	A7
i_re	56	E10	io_data[25]	75	A6	VDD_PD2	76	A5
i_rst	54	F11	io_data[26]	74	C6	VDDPST	12	G3
i_sys_clk	47	H10	io_data[27]	73	C7	VDDPST	20	J2
i_we	55	E11	io_data[28]	70	B6	VDDPST	31	K6
io_data[00]	40	L10	io_data[29]	69	A8	VDDPST	44	J10
io_data[01]	39	L9	io_data[30]	66	A10	VDDPST	52	G9
io_data[02]	38	K8	io_data[31]	65	B9	VDDPST	67	A9
io_data[03]	35	K7	NC	1	B2	VDDPST	81	A3
io_data[04]	34	L7	NC	21	L1	VSS	5	D2
io_data[05]	33	J7	NC	22	K2	VSS	16	H1
io_data[06]	30	L5	NC	41	K9	VSS	26	K4
io_data[07]	29	K5	NC	42	L11	VSS	37	L8
io_data[08]	28	J5	NC	43	K10	VSS	48	H11
io_data[09]	25	L3	NC	63	A11	VSS	59	D10
io_data[10]	24	L2	NC	64	B10	VSS	71	B7
io_data[11]	23	K3	NC	84	A1	VSS	77	B5
io_data[12]	18	J1	NC	85	C3	VSSPST	13	G1
io_data[13]	17	H2	o_data_valid	50	G10	VSSPST	19	K1
io_data[14]	10	F2	o_interrupt	14	G2	VSSPST	32	J6
io_data[15]	9	E1	VDD_PD1	15	F1	VSSPST	45	K11
io_data[16]	8	E2	VDD_PD1	27	L4	VSSPST	51	G11
io_data[17]	7	E3	VDD_PD1	36	L6	VSSPST	68	B8
io_data[18]	4	C1	VDD_PD1	49	F10	VSSPST	80	B4
io_data[19]	3	B1						

ANNEXE C ATTRIBUTION DES PATTES DU CONNECTEUR FMC-HPC

Tableau C.1 Attribution des pattes du connecteur FMC-HPC

Signal	**Émulateur**	**Mezzanine**	**Signal**	**Émulateur**	**Mezzanine**
gpio_addr[0]	HA16_P	HA02_P	gpio_data[22]	HA11_P	HB10_P
gpio_addr[1]	HA16_N	HA02_N	gpio_data[23]	HA11_N	HB10_N
gpio_addr[2]	HA17_P	HA07_P	gpio_data[24]	HA12_P	HB15_P
gpio_addr[3]	HA17_N	HA07_N	gpio_data[25]	HA12_N	HB15_N
gpio_data[00]	HA0_P	HA14_P	gpio_data[26]	HA13_P	HB14_P
gpio_data[01]	HA0_N	HA14_N	gpio_data[27]	HA13_N	HB14_N
gpio_data[02]	HA1_P	HA17_P	gpio_data[28]	HA14_P	HB18_P
gpio_data[03]	HA1_N	HA17_N	gpio_data[29]	HA14_N	HB18_N
gpio_data[04]	HA2_P	HA18_P	gpio_data[30]	HA15_P	HB17_P
gpio_data[05]	HA2_N	HA18_N	gpio_data[31]	HA15_N	HB17_N
gpio_data[06]	HA3_P	HA21_P	gpio_int	HA19_P	HA03_N
gpio_data[07]	HA3_N	HA21_N	gpio_re	HA18_N	HA06_N
gpio_data[08]	HA4_P	HA22_P	gpio_ready	HA19_N	HA03_P
gpio_data[09]	HA4_N	HA22_N	gpio_we	HA18_P	HA06_P
gpio_data[10]	HA5_P	HA23_P	rst		LA02_P
gpio_data[11]	HA5_N	HA23_N	io_clk	CLK1_M2C	LA00_CC
gpio_data[12]	HA6_P	HB01_P	sys_clk		HA00_CC
gpio_data[13]	HA6_N	HB01_N	sys_clk_dir		LA03_N
gpio_data[14]	HA7_P	HB00_P	sys_clk_oe		LA03_P
gpio_data[15]	HA7_N	HB00_N	ext_clk_oe		HA04_P
gpio_data[16]	HA8_P	HB07_P	ctrl_pd2_0		HA10_N
gpio_data[17]	HA8_N	HB07_N	ctrl_pd2_1		HA10_P
gpio_data[18]	HA9_P	HB06_P	ctrl_pd2_2		HA11_N
gpio_data[19]	HA9_N	HB06_N	ctrl_pd2_3		HA11_P
gpio_data[20]	HA10_P	HB11_P	ctrl_pd2_4		LA02_N
gpio_data[21]	HA10_N	HB11_N			

ANNEXE D HIÉRARCHIE DES FICHIERS DU TESTEUR FPGA

FIGURE D.1 Hiérarchie des fichiers du testeur FPGA

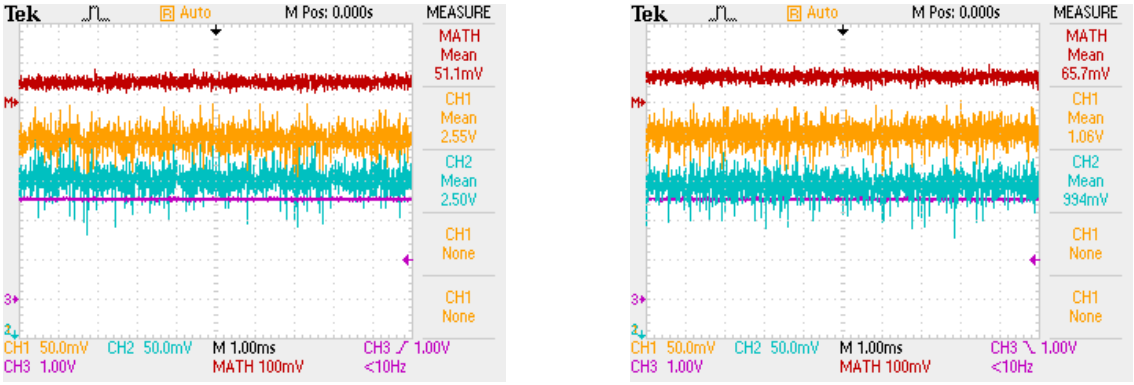
ANNEXE E HIÉRARCHIE DES FICHIERS DE L'ÉMULATEUR FPGA

FIGURE E.1 Hiérarchie des fichiers de l'émulateur du décodeur

ANNEXE F HIÉRARCHIE DES FICHIERS DE L'ASIC

FIGURE F.1 Hiérarchie des fichiers de l'ASIC du décodeur

ANNEXE G BIAIS DES CANAUX DE L'OSCILLOSCOPE



(a) Biais des canaux de l'oscilloscope à 2.5V (b) Biais des canaux de l'oscilloscope à 1V

FIGURE G.1 Biais des canaux de l'oscilloscope