TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACE	« Advanced Configuration Environment »
AMD	« Advanced Micro Devices »
ASM	langage Assembleur
ASMBL	« Advanced Silicon Modular BLock »
APU	« Auxiliary Processing Unit »
ATX	« Advanced Technology Extended »
BGA	« Ball Grid Array »
BIPM	Bureau International des Poids et Mesures
CISC	« Complex Instruction Set Computer »
CF	« Compact Flash »
CLB	« Configurable Logic Bloc »
CMOS	« Complementary Metal-Oxide-Semiconductor »
CPU	« Central Processor Unit »
DDR	« Double Data Rate »
DDR2	« Double Data Rate two »
DMIPS	« Dhrystone Million Instruction Per Second »
DSP	« Digital Signal Processing »
DUT	« Design Under Test »
DUV	« Design Under Verification »
DX	« DirectX »
DXUT	« DirectX Utility Library »

ECC	« Error Correction Coding » ou « Error Checking and Correcting »
EDK	« Embedded Development Kit »
ELF	« Executable and Linking Format »
EMC	« External Memory Controller »
ESL	« Electronic System Level »
ÉTS	École de Technologie Supérieure
FIT	« Fixed Interval Timer »
FLOPS	« FLoating point Operations Per Second »
FPGA	« Field Programmable Gate Array »
GF	« giga FLOPS »
GUI	« Graphic User Interface »
HD	Haute Définition
HDL	« Hardware Description Language »
HIL	« Hardware In the Loop »
HILS	« Hardware In the Loop Simulation »
HLS	« Hardware in the Loop Simulation »
HP	« Hewlett-Packard »
HPC	« High Performance Computing »
IBM	« International Business Machine »
IDE	« Integrated Drive Electronics »
IEEE	« Institute of Electrical and Electronics Engineers »
IML	« Integrated Microsystems Laboratory »

IO	« Input/Output »
IP	« Intellectual Property »
IPIF	« Intellectual Property InterFace »
IREQ	Institut de Recherche d'Hydro-Québec
ISE	« Integrated Software Environnement »
ISim	« ISE Simulator »
JTAG	« Joint Test Action Group »
KPOH	« Kilo Power-On Hours »
LACIME	LAboratoire de Communication et d'Intégration de la MicroÉlectronique
LE	Laboratoire d'Expérimentation
LUT	« Look-Up Table »
MAC	« Media Access Control »
MCH	« Multi-CHannel »
MATLAB	« MATrix LABoratory »
MD5	« Message-Digest algorithm 5 »
MUT	« Module Under Test »
PC	« Personal Computer »
PCI	« Peripheral Component Interconnect »
PLB	« Processor Local Bus »
PLD	« Programmable Logic Device »
PowerPC	« Performance Optimization With Enhanced RISC Performance Computing »
PPC	« PowerPC »
RISC	« Reduced Instruction Set Computer »

SATA	« Serial Advanced Technology Attachement »
SDR	« Software-Defined Radio »
SHA	« Secure Hash Algorithm »
SRAM	« Static Random Access Memory »
SDK	« Software Development Kit »
SDRAM	« Synchronous Dynamic Random Access Memory »
TCP	« Transmission Control Protocol »
UART	« Universal Asynchronous Receiver Transmitter »
UUT	« Unit Under Test »
USB	« Universal Serial Bus »
VHDL	« VHSIC Hardware Description Language »
VHSIC	« Very High Speed Integrated Circuit »
XPS	« Xilinx Platform Studio »

LISTE DES SYMBOLES ET UNITÉS DE MESURE (SYSTÈME INTERNATIONAL)

UNITÉS DE TEMPS

Période

h	heure
min	minute
s	seconde
ms	milliseconde
µs	microseconde
ns	nanoseconde
a	année
d	jour

Fréquence

GHz	gigahertz
MHz	mégahertz
kHz	kilohertz
Hz	hertz

UNITÉS GÉOMÉTRIQUES

Longueur

m	mètre
dm	décimètre
cm	centimètre
mm	millimètre
µm	micromètre
nm	nanomètre

Aire

km²	kilomètre carré (= 1 000 000 m²)
hm²	hectomètre carré (= 10 000 m²)
ha	hectare (= 10 000 m²)
m²	mètre carré
dm²	décimètre carré
cm²	centimètre carré
mm²	millimètre carré

Volume

km³	kilomètre cube
m³	mètre cube
dm³	décimètre cube
cm³	centimètre cube
hl	hectolitre (= 0,1 m³)
L	litre (= 1 dm³)
dL	décilitre
cL	centilitre
mL	millilitre (= 1 cm³)

UNITÉS DE MASSE

Masse

t	tonne (= 1 000 kg)
kg	kilogramme
g	gramme
dg	décigramme
cg	centigramme
mg	milligramme
µg	microgramme

Masse surfacique

kg/m²	kilogramme par mètre carré

Masse volumique

kg/m³	kilogramme par mètre cube

UNITÉS CALORIFIQUES

Température

K	kelvin
°C	degré celcius

UNITÉS MÉCANIQUES

Angle plan

rad	radian
gr	grade
r	rotation
°	degré
′	minute
″	seconde

Vitesse

m/s	mètre par seconde
km/h	kilomètre par heure

Vitesse angulaire

rad/s	radian par seconde
r/s	rotation par seconde
r/min	rotation par minute

Accélération

m/s²	mètre par seconde carré

Accélération angulaire
rad/s² radian par seconde carré

Force
N newton

Moment d'une force
N.m newton-mètre

Tension superficielle
N/m newton par mètre

Énergie, travail, quantité de chaleur
MJ mégajoule
kJ kilojoule
J joule

Énergie massique
J/kg joule par kilogramme

Capacité thermique
Entropie
J/K joule par kelvin

Flux thermique surfacique
W/m² watt par mètre carré

Conductivité thermique
W/(m.K) watt par mètre kelvin

Contrainte, pression
MPa mégapascal
kPa kilopascal
Pa pascal

Viscosité cinématique
m²/s mètre carré par seconde

Viscosité dynamique
Pa.s pascal seconde

UNITÉS ÉLECTRIQUES ET MAGNÉTIQUES

Résistance électrique
TΩ téraohm
MΩ mégohm
kΩ kiloohm
Ω ohm
mΩ milliohm
µΩ microhm

Conductance électrique
S siemens

Capacité électrique
F farad
mF millifarad
µF microfarad
nF nanofarad
pF picofarad

Inductance électrique
H henry
mH millihenry
µH microhenry

Différence de potentiel électrique (ou tension)
Force électromotrice
MV mégavolt
kV kilovolt
V volt
mV millivolt
µV microvolt

Intensité de courant électrique
Force magnétomotrice
kA kiloampère
A ampère
mA milliampère
µA microampère

Puissance
Flux énergétique
MW mégawatt
kW kilowatt
W watt
mW milliwatt
µW microwatt
VA voltampère (puissances apparentes)

Champ électrique
V/m volt par mètre

Permittivité
F/m farad par mètre

Perméabilité
H/m henry par mètre

Charge électrique
Quantité d'électricité
C coulomb

Induction électrique
C/m² coulomb par mètre carré

Flux d'induction magnétique
Wb weber

Induction magnétique
T tesla

Intensité de champ magnétique
A/m ampère par mètre

Densité de courant
A/m² ampère par mètre carré

UNITÉS DE QUANTITÉ DE MATIÈRE

Quantité de matière
kmol kilomole
mol mole
mmol millimole
µm micromole

Concentration
mol/m³ mol par mètre cube

Exposition aux rayons X ou Y
Quantité de rayonnement X ou Y
C/kg coulomb par kilogramme

Activité d'un radionucléide
bq becquerel

Dose absorbée
Gy gray

Équivalent de dose ambiant
Sv sievert

UNITÉS OPTIQUES

Intensité lumineuse
cd candela

Luminance
cd/m² candela par mètre carré
cd/cm² candela par centimètre carré

Flux lumineux
lm lumen

Éclairement
lx lux
ph phot

UNITÉS D'INTENSITÉ DU SON

dB décibel

Source : (Bureau international des poids et mesures, 2006).

INTRODUCTION

Le domaine de la microélectronique est vaste, sophistiqué et évolue à une vitesse très rapide en comparaison avec la majorité des autres domaines de génie. En microélectronique, le développement de systèmes numériques suit un processus de conception complexe à plusieurs étapes qui s'échelonne sur différents niveaux d'abstraction.

La vérification des systèmes numériques est une étape de la conception qui prend beaucoup d'importance en termes de temps et de ressources investis selon (Giard, Boland et Belzile, 2008). Cette vérification est cruciale, car elle permet de démontrer le bon fonctionnement du système au niveau comportemental et physique avant sa réalisation. De plus, une vérification comportementale effectuée correctement permet de minimiser le nombre d'itérations pour les étapes subséquentes, dont la synthèse, le placement et le routage.

Pour les systèmes centrés autour de circuits programmables de type FPGA, la vérification est majoritairement effectuée dans un outil de simulation qui peut être intégré ou externe à l'environnement de développement propriétaire utilisé. Ce simulateur doit utiliser des fichiers de test (« testbench ») pour faire les interconnexions, créer et synchroniser les stimuli à envoyer au module conçu à vérifier. Ce fichier de test doit représenter le plus possible la réalité, car il est possible que le fonctionnement réel sur une plateforme matérielle ne donne pas les résultats escomptés alors que la vérification avait préalablement été réussie. Dans ce cas, une nouvelle itération est alors nécessaire et le temps de vérification s'en retrouve inévitablement augmenté.

Il est possible de complémenter le processus de vérification conventionnel, utilisant des fichiers de test, par une méthode qui compare les résultats souhaités à ceux provenant d'une plateforme matérielle qui exécute le module conçu à vérifier. De cette façon, le processus tire profit des modèles à plus haut niveau d'abstraction pour la génération des données et la comparaison des données résultantes. Cette méthode est basée sur une technique nommé

« Hardware In the Loop » (HIL). Cette technique permet d'introduire des éléments matériels dans la boucle de simulation. Elle permet généralement de remplacer certaines opérations logicielles de la simulation numérique par des traitements réalisés par des éléments matériels qui fonctionnent physiquement dans la réalité.

Lorsqu'il est utilisé dans un contexte de simulation, le HIL peut porter le nom de « Hardware In the Loop Simulation » (HILS) ou « Hardware in the Loop Simulation » (HLS). Pour des fins d'homogénéité nous utiliserons le terme HIL au cours de cet ouvrage.

La technique du HIL est en pleine effervescence depuis plusieurs années et est utilisée à des fins très variées dans différents domaines. Par exemple, dans le domaine aérospatial, (Leitner, 1996) utilise le HIL pour valider la capacité d'intégration de certaines composantes critiques à une navette spatiale. Dans cet exemple, le HIL utilise les composants matériels à intégrer et le modèle mathématique de la navette est simulé pour des raisons évidentes. Aussi, dans le domaine de l'automobile, (Short et Pont, 2005) valident la fiabilité et la sécurité d'un ordinateur de bord en utilisant le HIL. L'ordinateur embarqué qui contrôle le véhicule est réel, alors que la dynamique du véhicule est simulée. De plus, dans le domaine de la robotique, (Hamelin et al., 2008) utilisent le HIL pour tester des commandes d'impédance du contrôleur d'un robot virtuel en utilisant les capteurs et les actionneurs d'un robot réel de topologie différente.

Dans le cadre de ce projet de recherche, notre application du HIL s'applique au domaine de la vérification numérique de systèmes microélectroniques. Nous posons l'hypothèse qu'il est possible de faciliter et d'accélérer globalement le processus de vérification numérique en utilisant une approche de vérification comparative qui utilise le HIL.

Le présent mémoire est composé de 4 chapitres qui s'organisent comme suit :

1) le chapitre 1 présente plus en détail la problématique qui amène le projet d'accélération sur matériel de la simulation numérique;

2) le chapitre 2 décrit en détail la méthodologie utilisée pour que le HIL puisse être couplé à la simulation numérique;

3) le chapitre 3 présente plusieurs conceptions numériques qui seront utilisées et comparées pour tester le HIL;

4) le chapitre 4 expose les résultats obtenus par la méthode utilisée et présente l'analyse de ces résultats en fonction des types de conception.

CHAPITRE 1

PROBLÉMATIQUE

1.1 Contexte

Que ce soit pour la conception physique de puces électroniques (ASIC) ou l'utilisation de puces électroniques programmables (PLD), la conception numérique est la pierre angulaire du développement d'applications de traitement de signaux numériques (DSP).

Les PLD que nous utilisons de nos jours sont fabriqués en utilisant un procédé de fabrication de plus en plus fin. Cette diminution du grain peut procurer plusieurs avantages techniques pour les concepteurs, mais entraîne au départ des coûts de développement plus élevés et un risque accru en fonction du marché visé. La Figure 1.1 compare les aspects monétaires du développement en fonction du procédé de fabrication utilisé.

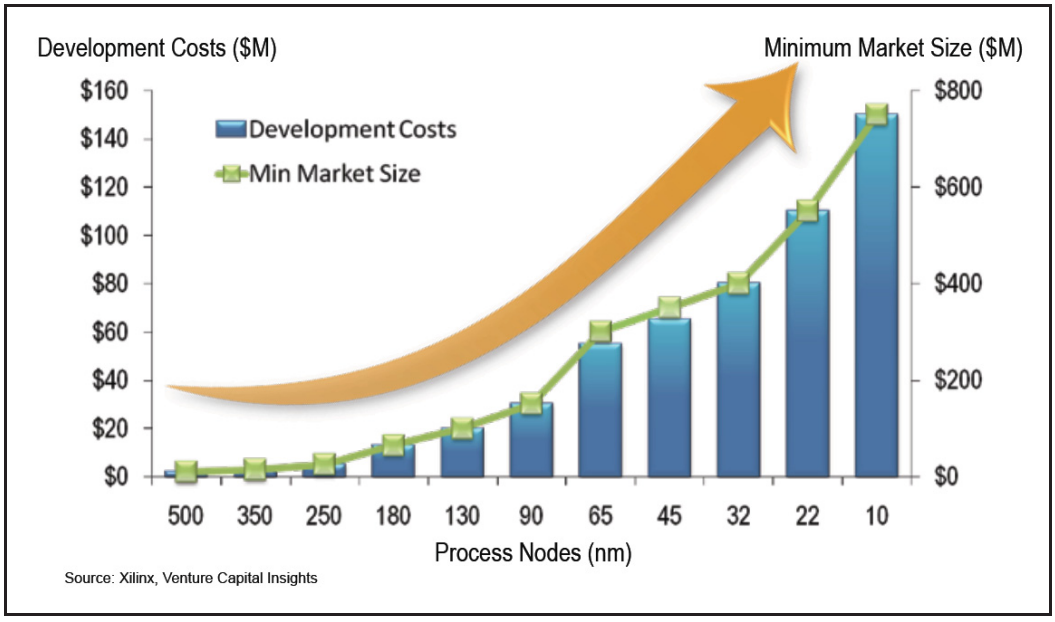


Figure 1.1 Évolution des procédés de fabrication en microélectronique.
Tirée de Xilinx (2009, p.6)

Le nombre de conceptions numériques sur des plateformes programmables de type PLD, dont les FPGA, est en croissance depuis plusieurs années. Cette augmentation se reflète sur la quantité de PLD vendus mondialement, comme le montre la Figure 1.2.

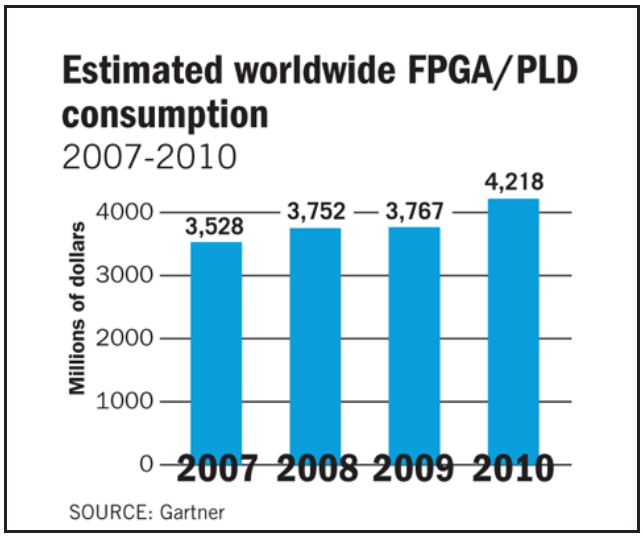


Figure 1.2 Consommation mondiale estimée de FPGA/PLD.
Tirée de McGrath (2009, p.2)

Bien qu'historiquement, on puisse dénombrer beaucoup de démarrages de compagnie dans le domaine de la fabrication de PLD, seulement quelques compagnies ont établie une niche stable dans ce domaine. La Figure 1.3 présente l'ensemble de cet historique.

Il est à noter que les quatre compagnies ayant les parts de marché les plus élevées pour la vente de PLD, pour l'année 2008, totalisent 98.6 % du marché. Ces compagnies sont dans l'ordre, selon (McGrath, 2009, p.1) :

1) Xilinx avec 51.2 %;
2) Altera avec 35.5 %;
3) Lattice Semiconductor avec 6 %;
4) Actel avec 5.9 %.

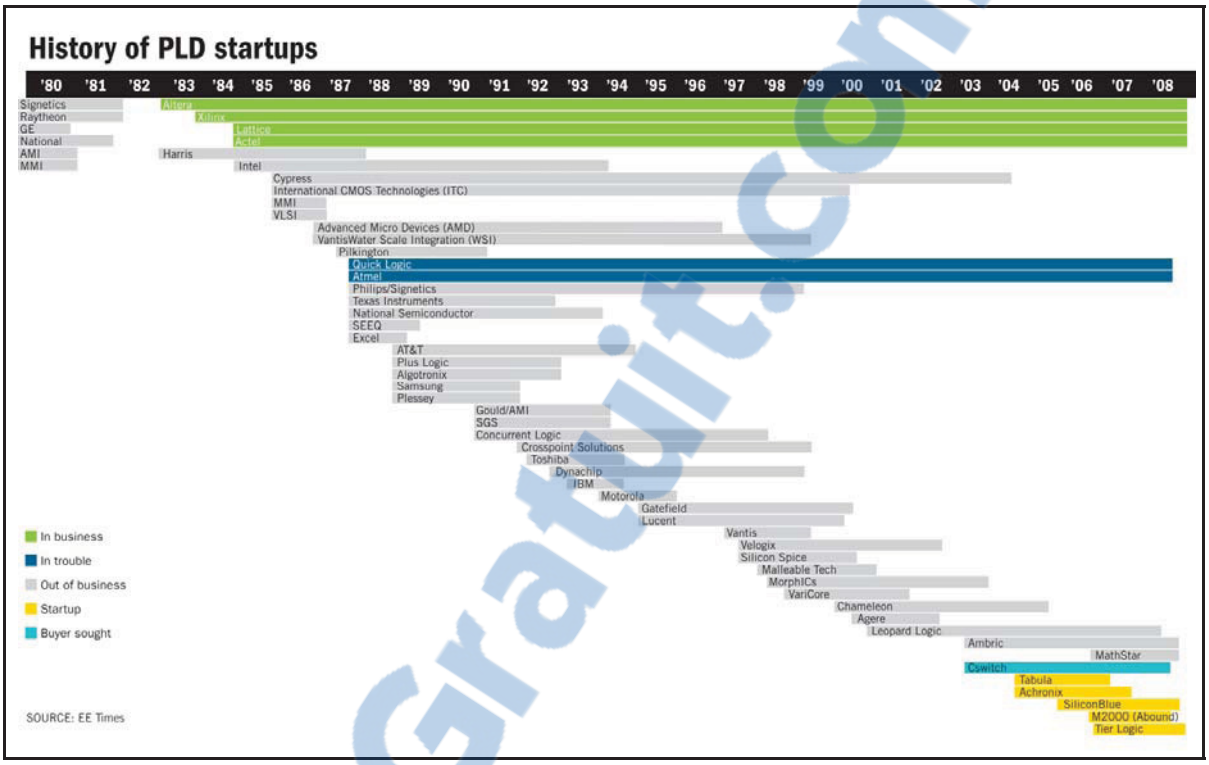


Figure 1.3 Évolution des compagnies de PLD.
Tirée de McGrath (2009, p.1)

Les conceptions numériques sur des plateformes PLD suivent un processus de conception complexe qui peut être représenté par une suite de plusieurs étapes, parfois itératives, tel qu'illustré à la Figure 1.4.

Le processus de conception numérique se répartit sur différents niveaux d'abstraction échelonné du plus haut niveau (les spécifications du système étudié) au plus bas niveau (le circuit interne de la puce électronique utilisée). Du côté droit de la Figure 1.4, on peut voir les différents niveaux d'abstraction traversés au cours du développement.

La validation permet de s'assurer du respect des spécifications de performance du système au niveau architectural. On valide ainsi que le système fait la bonne fonction, le bon traitement ou la bonne opération.

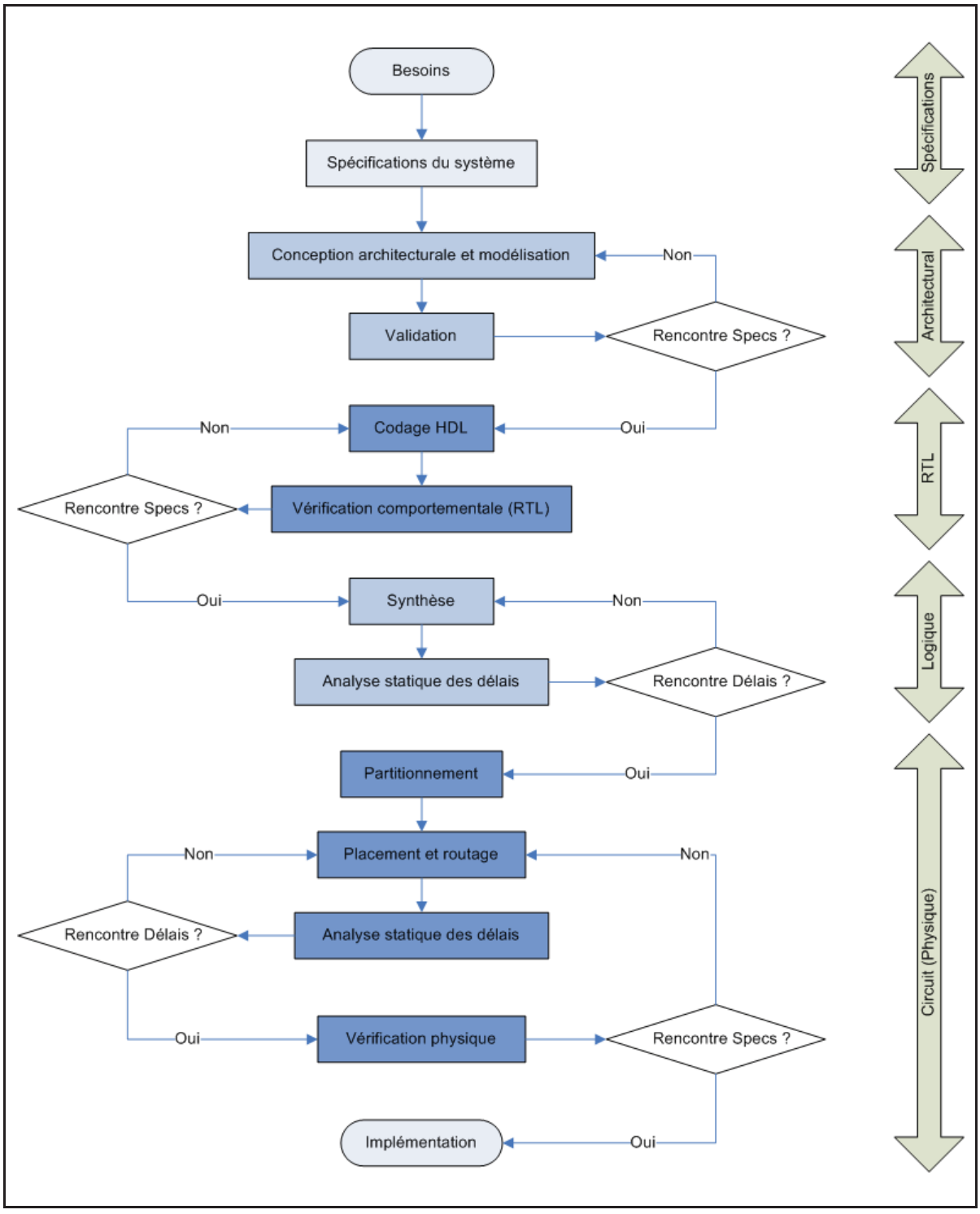


Figure 1.4 Processus global de la conception numérique.

La vérification permet de s'assurer que l'on respecte toutes les spécifications de façon cohérente suite au passage d'un niveau supérieur à un niveau inférieur. On vérifie ainsi que le système fait bien ce qu'on pense qu'il fait ! Pour se faire on utilise habituellement des simulations logiques ou paramétriques, accompagnées d'analyses statiques des délais.

Les vérifications au sein du processus de conception numérique doivent être effectuées avec minutie et en prenant compte des caractéristiques physiques des composantes matérielles utilisées (ex. : CLB pour un FPGA Xilinx). Cette rigueur est importante, car les simulations que composent ces vérifications ne sont aussi bonnes que l'ensemble des vecteurs qui les composent.

Il est important de comprendre que ces vérifications sont essentielles, car la garantie de fonctionnement d'une conception numérique ne pourra pas être donnée sans elles.

1.2 Identification du problème

La technologie reliée aux procédés de fabrication en microélectronique progresse de jours en jours. Ainsi, à chaque nouvelle génération, le nombre de portes logiques au sein des FPGA augmente. Au Tableau 1.1, on peut voir l'évolution des ressources disponibles pour les FPGA Virtex-4, Virtex-5, Virtex-6 et les futurs Virtex-7 de la compagnie Xilinx. Cet accroissement des capacités est bienvenu, car il permet aux concepteurs de réaliser des conceptions numériques de tailles plus importantes et de complexités accrues.

Les experts de l'industrie nous apprennent que 60 % à 80 % de l'effort de conception numérique serait attribuable à la vérification, selon (Bergeron, 2006).

La problématique que nous adressons provient de la combinaison des deux énoncés précédents. Ils montrent que la vérification demande une portion de temps importante au sein de l'effort de conception et que cet investissement de temps sera de plus en plus grand à cause de la complexité grandissante des conceptions futures.

Tableau 1.1 Comparaison des ressources pour les FPGA Virtex de Xilinx
Données tirées de Xilinx (2007, 2010a, 2010b et 2010c)

Ressources	**Virtex-4**	**Virtex-5**	**Virtex-6**	**Virtex-7**
Cellules logiques	14k – 200k	20k – 332k	74k – 567k	286k – 1955k
LUT des CLB	11k – 178k	12k – 207k	46k – 354k	179k – 1222k
Bascules des CLB	11k – 178k	12k – 207k	92k – 708k	358k – 2443k
RAM (Mb)	0.6 – 9.9	0.9 – 18.6	5.6 – 32.8	14.8 – 64.8
Blocs DSP	32 – 512	24 – 1056	288 - 2016	700 – 3960
SelectIO	320 – 960	172 – 1200	360 - 1200	600 - 1200

Plus particulièrement, la fiabilité de la vérification repose principalement sur les fichiers de test utilisés. L'écriture de fichiers de test est problématique, car elle présente plusieurs irritants :

- elle consomme beaucoup de temps;
- il est difficile de représenter exactement la réalité;
- différents stimuli doivent être reproduits.

En conception numérique, le processus de vérification est donc une tâche itérative qui demande beaucoup de rigueur pour garantir l'ensemble des résultats. Ce travail laborieux amène malheureusement certains concepteurs à bâcler consciemment ou non cette tâche.

1.3 Objectifs

Au cours de ce projet, notre but est de faciliter et d'accélérer globalement le processus de vérification numérique en utilisant une approche de vérification comparative qui utilise le « Hardware In the Loop ».

Le HIL permet de complémenter l'approche standard utilisée pour la vérification comportementale. L'écriture du fichier de test et la simulation logique sont donc bonifiés par l'utilisation du matériel dans la boucle. De plus, l'utilisation du HIL peut amener la détection, dans une première phase, de certaines erreurs normalement perçues dans la phase de vérification physique qui est la dernière étape de la conception numérique avant l'implémentation finale. Les erreurs marquantes (« show stopper ») sont donc détectées de manière plus précoce, ce qui permet de sauver une ou des itérations de vérifications ultérieures. Voici quelques exemples de problèmes détectables :

- traitements non-conformes par rapport au modèle haut-niveau de référence;
- manque de ressources physiques;
- problèmes relatifs aux délais;
- mauvaise synchronisation.

Pour complémenter la vérification en utilisant le HIL, nous allons utiliser des stimuli provenant d'un logiciel de haut niveau, transmettre ceux-ci à la plateforme matérielle, recevoir les résultats de la plateforme et par la suite afficher les résultats dans le même logiciel pour des fins de comparaison. Le module conçu à vérifier, nommé DUV (« Design Under Verification »), est exécuté physiquement par la plateforme matérielle à travers l'infrastructure HIL. Certains auteurs dans la littérature, comme le montre (Short, 2009), utilisent aussi les termes « Unit Under Test » (UUT), « Design Under Test » (DUT) ou « Module Under Test » (MUT). Pour des fins d'homogénéité nous utiliserons le terme DUV dans l'ensemble de cet ouvrage.

Voici les objectifs visés en utilisant une approche de vérification comparative par HIL :

1) détecter de manière précoce les obstacles insurmontables;
2) accélérer globalement le processus de vérification;
3) rendre le processus de vérification moins fastidieux et plus convivial;
4) exécuter les fonctionnalités sur une plateforme matérielle proche de la plateforme utilisée après la conception (principe du HIL).

1.4 Travaux antérieurs

Ce projet se veut un ajout aux travaux de M. Jean-François Boland sur l'utilisation de MATLAB et Simulink dans un environnement de vérification centré sur le langage SystemC (Boland, Thibeault et Zilic, 2005). Ces travaux ont de multiples applications dans plusieurs domaines. On peut citer en exemple son utilisation relié au traitement de signaux numériques pour une radio à composantes logicielles (Boland et al., 2004).

Le projet de contrôle du HIL est en fait un travail en parallèle avec les travaux de M. Boland et permet d'implémenter et de tester le HIL en utilisant une interface de connexion minimale avec MATLAB. Il est possible de remplacer éventuellement la connexion actuelle avec MATLAB par une liaison à l'environnement de vérification centré sur le langage SystemC de M. Boland.

Cette fusion des deux projets amènerait alors une bonification de la vérification grâce à la combinaison des travaux précédents de M. Boland sur l'environnement de vérification SystemC et les travaux actuels sur le HIL.

1.5 Contributions

Nos contributions au projet de développement d'une infrastructure pour l'accélération sur matériel de la simulation numérique se résume à :

1) la sélection d'une plateforme matérielle FPGA appropriée;

2) la création d'un projet de développement FPGA supportant le HIL;
3) la création d'un cœur d'encapsulation du DUV;
4) le développement d'un logiciel de gestion des données;
5) le développement d'un logiciel de contrôle du HIL et d'interface graphique;
6) les tests, les mesures et l'analyse des résultats du HIL.

CHAPITRE 2

MÉTHODOLOGIE

2.1 Contexte

Dans le but de minimiser l'effort en termes de temps et de ressources investis pour la vérification d'une conception numérique, le DUV peut être testé en utilisant une infrastructure basée sur le principe du HIL. Pour ce faire, les stimuli provenant d'un logiciel de haut niveau peuvent être envoyés à une plateforme matérielle exécutant physiquement le DUV. Les résultats obtenus peuvent ensuite être renvoyés au logiciel de haut-niveau pour des fins de comparaison avec les résultats produits lors de la simulation haut-niveau.

2.2 Infrastructure

L'infrastructure que nous proposons comporte une plateforme matérielle, ainsi qu'un ordinateur connecté à celle-ci pour des fins de programmation et de contrôle du processus de HIL. La plateforme matérielle doit minimalement contenir les composantes suivantes pour que l'infrastructure du HIL puisse être supportée :

- un FPGA de la compagnie Xilinx avec un processeur intégré (PowerPC);
- de la mémoire RAM;
- de la mémoire Flash;
- une interface de communication réseau (Ethernet 10/100);
- un lien de programmation JTAG.

La Figure 2.1 montre l'ensemble des interactions entre la plateforme matérielle et l'ordinateur. Les parties de couleurs cyan et bleu sont des composantes matérielles, et celles en vert sont des logiciels commerciaux. Les parties en gris sont des modules électroniques fournis par la compagnie Xilinx, et celles en jaune et en orange sont des modules programmés pour les fins du projet.

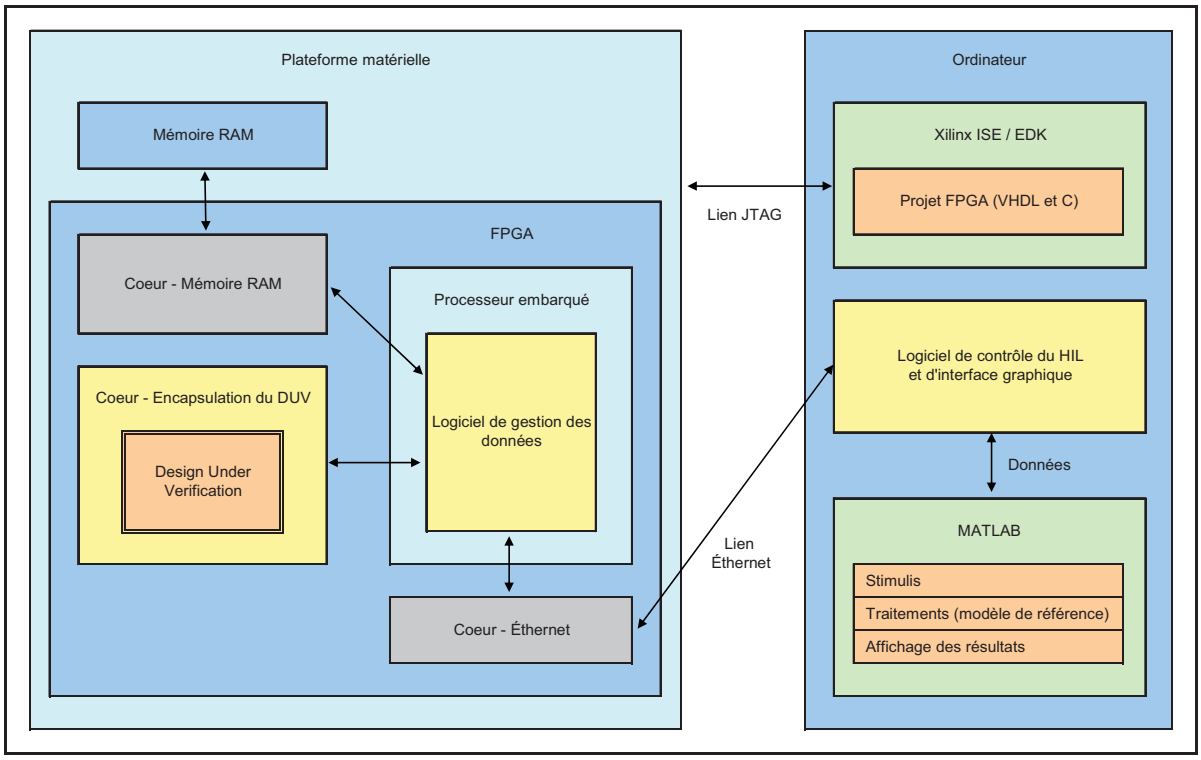


Figure 2.1 Infrastructure du système HIL.

Les différentes composantes matérielles et les logiciels utilisés dans le projet seront expliqués plus en détails dans les sections qui suivent.

2.3 Évaluation des outils

Au début du projet, afin de tester les différents outils reliés au développement du HIL, la plateforme matérielle ML310 de la compagnie Xilinx a été utilisée. Cette carte était performante et conçue dans un format ATX, similaire aux cartes maîtresses des ordinateurs PC. On peut voir sa composition à la Figure 2.2.

La carte ML310 était dotée d'un FPGA Virtex-2-Pro avec un processeur PowerPC intégré, 256 MB de mémoire DDR, des fentes PCI, une interface de communication réseau Ethernet à 10/100 Mbps, ainsi qu'une interface de communication sérielle RS-232 et JTAG. Elle comporte aussi une interface de transfert System ACE de Xilinx, permettant d'utiliser une

carte de données CompactFlash, et bien d'autres fonctionnalités facultatives en fonction de notre projet (interface PCI, IDE, audio, etc.).

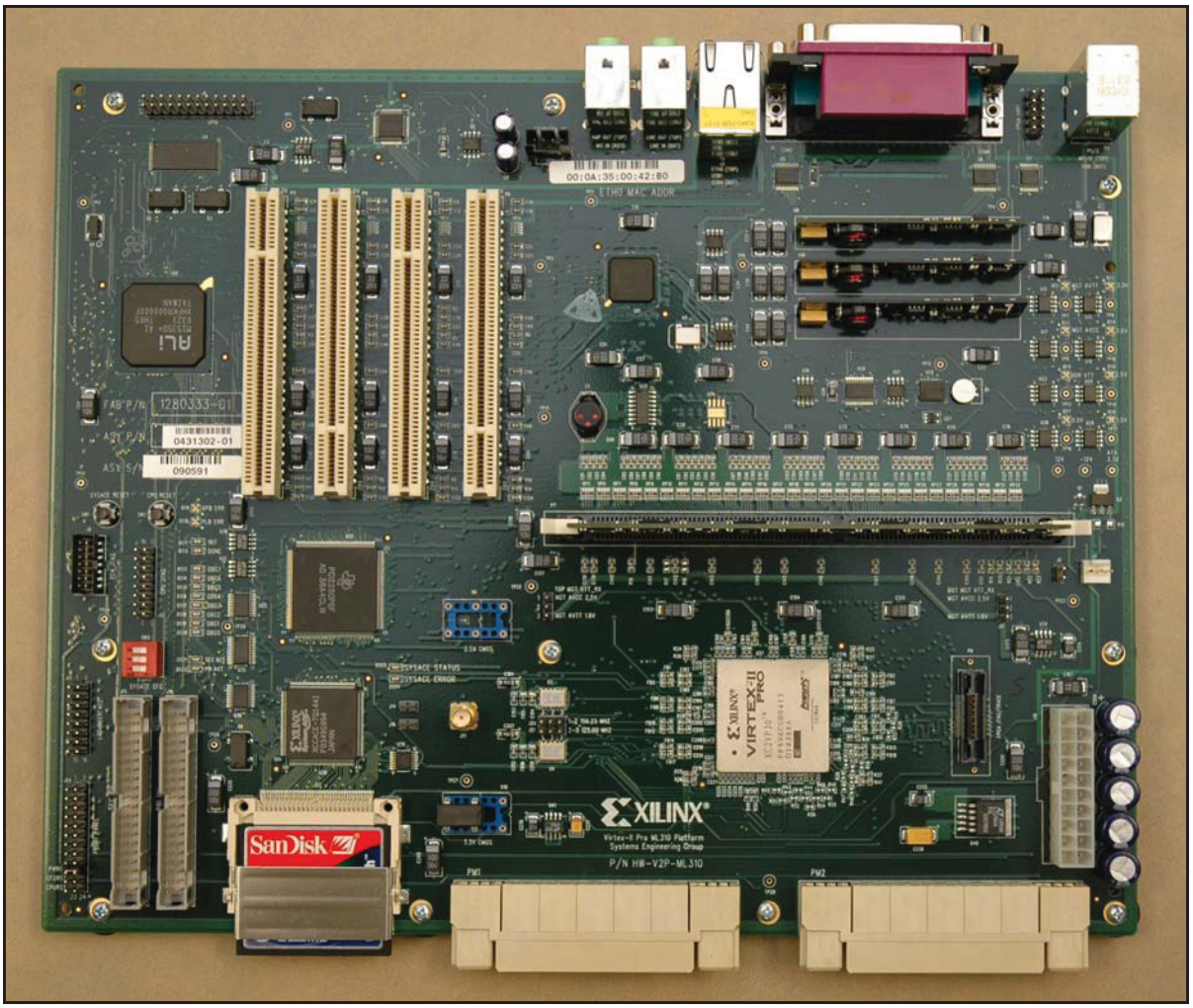


Figure 2.2 Carte de développement Xilinx ML310.
Tirée de Xilinx (2010d)

En utilisant les outils de Xilinx de l'époque (version 6), des problèmes furent rencontrés lors de l'utilisation du cœur Ethernet connecté au processeur PowerPC. Cette carte étant en fin de vie à cause de l'apparition de la technologie Virtex-4, la décision fût prise d'évoluer vers une carte FPGA plus récente et mieux supportée à moyen terme par les outils de développement.

2.4 Plateforme matérielle

La plateforme matérielle sélectionnée pour permettre le HIL est la carte de développement ML403 de Xilinx. Comparativement à la carte ML410 (la carte équivalente à la ML310, mais de la génération suivante) ce produit est plus compact, plus flexible et mieux supporté par les outils de conception plus récents (version 8 à 10). De plus, la carte ML403 est plus accessible et présente un meilleur potentiel de vente, à cause de son coût faible, soit moins de 1000 $ en 2009. Ce qui équivaut à un coût plus de trois fois inférieur à ML410.

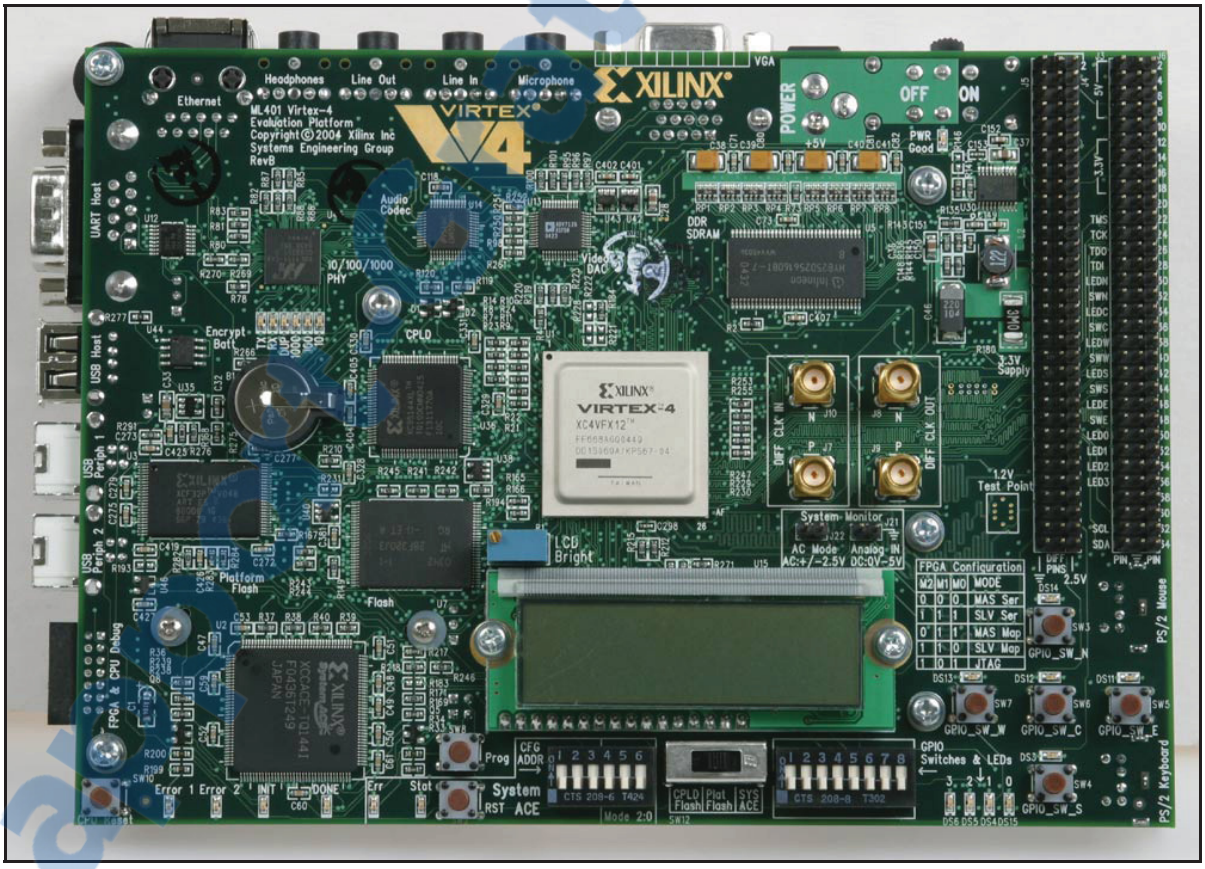


Figure 2.3 Carte Xilinx ML403 (recto).
Tirée de Xilinx (2010e)

La carte ML403, présentée à la Figure 2.3 et à la Figure 2.4, contient toute les fonctionnalités nécessaires au projet de HIL. Soit un FPGA Virtex-4 avec un processeur PowerPC intégré,

8 Mb de mémoire SRAM, 32 Mb de mémoire Flash, une interface de communication réseau Ethernet à 10/100/1000 Mbps, ainsi qu'une interface de communication sérielle RS-232 et JTAG.

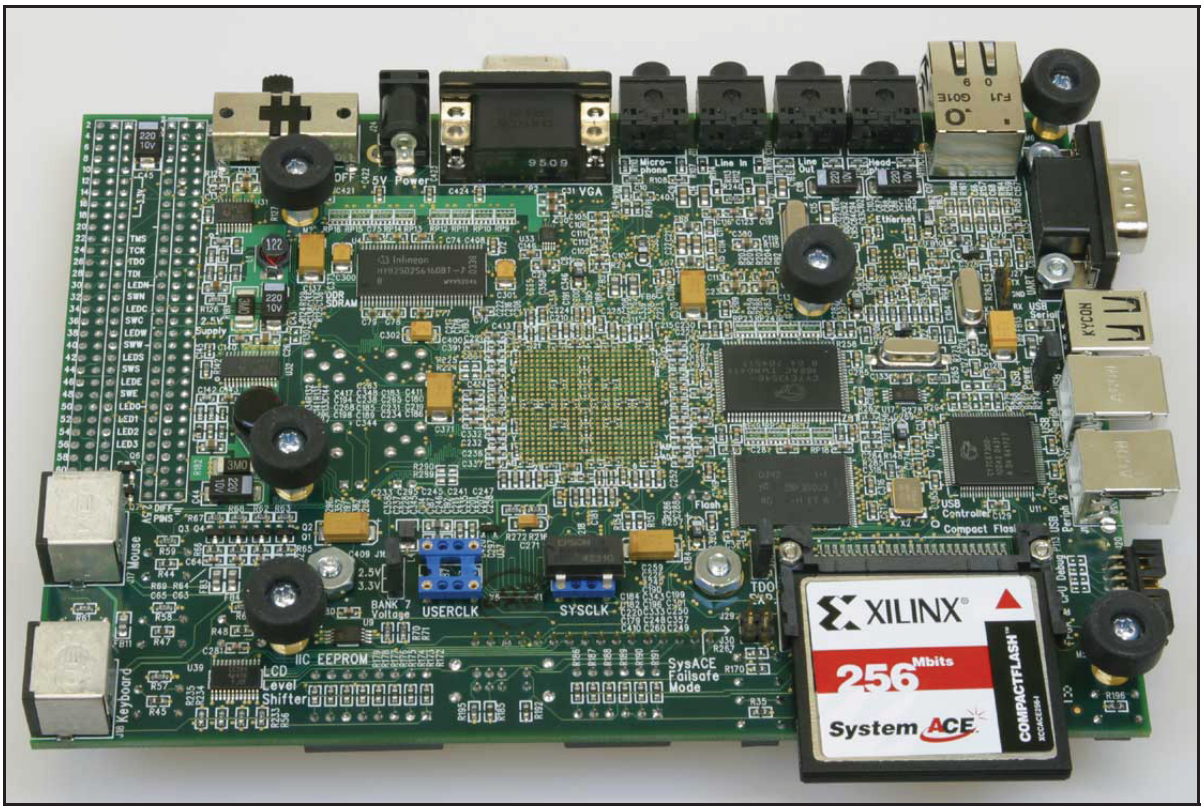


Figure 2.4 Carte Xilinx ML403 (verso).
Tirée de Xilinx (2010e)

Pour le bon fonctionnement du projet, les cavaliers doivent être placés à leur position originale, tel que décrit dans le manuel de l'usager de la carte, voir (Xilinx, 2006).

D'autres périphériques intégrés à la carte pourraient aussi être utilisés pour une utilisation future, soit 64 MB de mémoire DDR SDRAM, 8 MB de mémoire linéaire Flash, une interface de communication USB et une interface de communication System ACE. Cette dernière permet de charger des fichiers provenant d'une carte CF de type 1 ou 2.

2.4.1 FPGA

La carte ML403 contient un FPGA de type Virtex-4 FX12 de la compagnie Xilinx. Celui-ci est le plus petit de la gamme FX, qui cible les applications désirant utiliser une plateforme de traitement embarqué. On peut voir la comparaison des trois gammes qui compose le Virtex-4 à la Figure 2.5.

Le FPGA FX12 a été fabriqué grâce à un procédé de fabrication CMOS à 90 nm utilisant la technologie triple oxyde et l'architecture ASMBL («Advanced Silicon Modular Block»). Son alimentation interne est de 1.2 V, il comporte 11 couches de métallisation et il est intégré dans un boîtier de type BGA FF668.

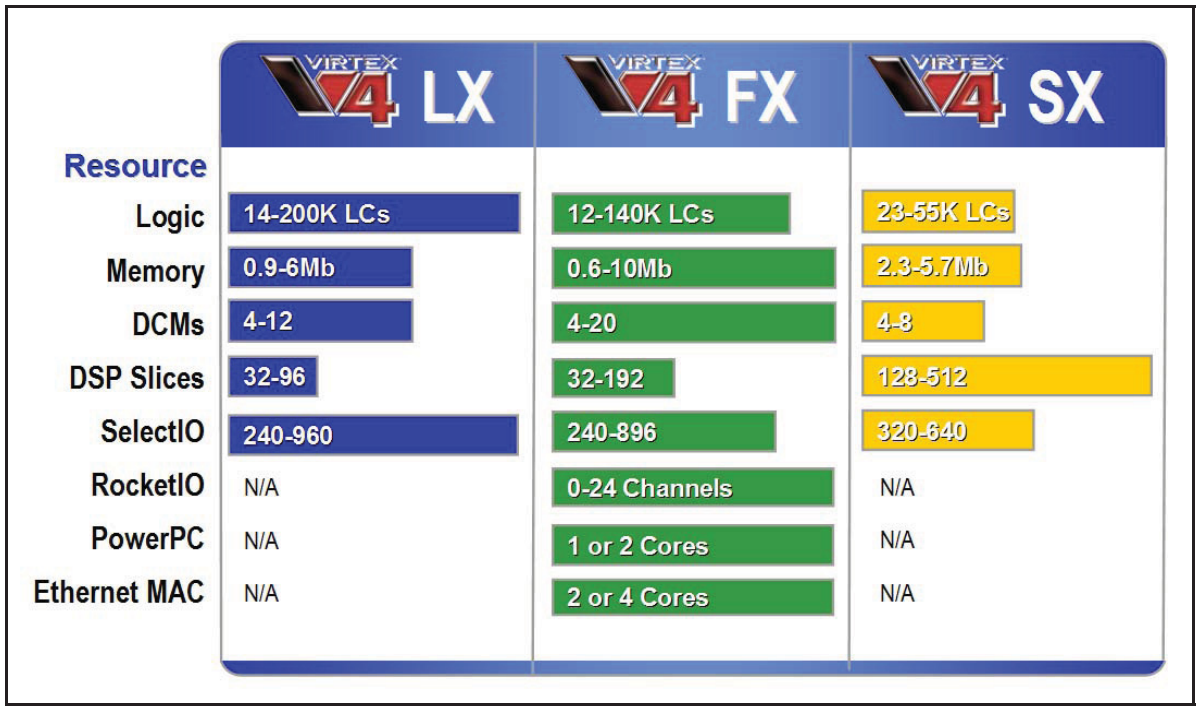


Figure 2.5 Caractéristiques des gammes du Virtex-4 de Xilinx.
Tirée de Xilinx (2005, p. 20)

La puce présente sur la carte utilisée a une côte de vitesse de 10 et son bus système est cadencé à 100 MHz. Elle contient 12312 cellules logiques, 32 blocs DSP, un processeur embarqué PowerPC et deux blocs de communication Ethernet MAC.

2.4.2 Processeur embarqué

Le processeur intégré au Virtex-4 FX est un PowerPC 405 de la compagnie IBM. Le processeur PowerPC utilise un jeu d'instruction réduit (RISC). Ce type de processeur privilégie une plus grande simplicité au niveau des instructions, ce qui résulte en une taille d'unité de calcul réduite en comparaison avec les processeurs qui utilisent des jeux d'instructions complexes (CISC). Ce type d'architecture privilégie la superscalarité (la capacité d'exécuter plusieurs instructions simultanément) au lieu de l'utilisation d'un «pipeline» complexe. Il est à noter que beaucoup de processeurs utilisés dans les PC utilisent une architecture CISC, comme par exemple la famille Pentium de Intel et la famille Athlon de AMD.

La microarchitecture PowerPC est très utilisée sur le marché et a la réputation d'avoir un ratio puissance de calcul sur consommation élevé. On peut voir un résumé de son évolution en fonction des applications et selon les compagnies productrices à la Figure-A I-1 de l'annexe I. En novembre 2009, la famille de processeurs PowerPC représentait 10.4 % de la composition des 500 superordinateurs les plus performants au monde, voir le Tableau 2.1. Cela représente un total de 1 470 752 cœurs répartis sur 52 superordinateurs.

Le PowerPC 405 est un CPU de 32 bits qui emploie un «pipeline» à 5 stages et dont l'optimisation vise les applications embarquées. Pour des fins de comparaison, le Tableau 2.2 illustre les caractéristiques physiques du processeur PowerPC utilisé par le HIL.

Tableau 2.1 Composition des 500 superordinateurs les plus performants
Données tirées de TOP500.org (2009)

Famille de processeur	Nombre d'unité	Proportion des 500	Pmax / unité (GF)	Nombre total de processeurs
Intel EM64T	396	79.2 %	61 849	2 217 612
PowerPC	52	10.4 %	145 671	1 470 752
AMD x86_64	42	8.4 %	194 571	897 175
Intel IA-64	6	1.2 %	57 224	54 512
Sparc	2	0.4 %	76 124	15 104
NEC	1	0.2 %	131 072	1280
autres	1	0.2 %	84480	8192

Il est difficile d'obtenir les mêmes comparatifs pour les processeurs Intel ou AMD présents dans les ordinateurs PC standards. Par contre, on peut utiliser de manière conservatrice, selon (Byme, 2010), un ratio Dhrystone/MHz de 2.4 pour un processeur Intel Atom Z550 de 2.0 GHz. Cela permet donc de comparer la performance du Atom, de 4800 DMIPS (=2.4*2000), à celle du PowerPC, de 456 DMIPS (=1.52*300). On peut donc estimer qu'un PC standard de 2 GHz a un potentiel de calcul d'environ 10.5 fois supérieur au PowerPC de 300 MHz. Il est à noter que pour notre application, ce ratio n'a pas d'impact direct étant donné que les calculs embarqués relié au HIL sont effectués de manière matérielle et non pas par le processeur PowerPC.

Tableau 2.2 Caractéristiques du processeur PPC405
Données tirées de IBM (2006, p. 5)

Performance (Dhrystone 2.1)	1.52 DMIPS/MHz
Performance vs. puissance	9.2 DMIPS/mW @ 1.2 V, 658 MHz
Technologie d'implémentation	90 nm IBM CMOS-9SF et IBM ASIC Cu-08 4 niveaux de metallisation
Fréquence d'horloge du CPU (procédé nominal, 55°C)	658 MHz @ 1.2V
Dissipation de puissance (nominal, actif et statique)	0.145 mW/MHz @ 1.2V, 55°C
	109 mW @ 658 MHz
Plage de voltage en opération	0.7 V à 1.3 V (1.2 V nominal)
Plage de température en opération	-55°C à 125°C
Dimension du cœur	2.17 mm² (avec une cache 16KB/16KB L1)
Temps de vie étendu	200 KPOH

Dans le cadre du projet, la fréquence programmée du processeur PowerPC est de 300 MHz. Aucun système d'exploitation ne sera utilisé, le processeur sera donc programmé en mode natif en utilisant le compilateur intégré à l'environnement de développement EDK. Ce compilateur supporte une combinaison des langages ASM, C et C++. Le logiciel embarqué exécuté sur le PowerPC sera expliqué plus en détail à la sous-section suivante.

2.4.3 Logiciel de gestion des données

Ce logiciel permet de gérer la transition des données échangées entre les différents acteurs du HIL. Dans un premier temps, il gère les stimuli reçus de l'ordinateur par le lien réseau et qui sont enregistrés dans la mémoire RAM de la plateforme HIL. Par la suite, il gère les échanges de données entre la mémoire RAM et le module d'encapsulation du DUV. Finalement, il permet l'envoi des résultats de la mémoire RAM vers l'ordinateur par le lien réseau.

Le logiciel est écrit en langage C en utilisant l'interface de programmation présente dans le logiciel EDK de Xilinx. Le résultat compilé sera placé dans un fichier .ELF qui sera envoyé au FPGA par le lien JTAG.

2.4.4 Cœur d'encapsulation du DUV

Ce cœur permet de contrôler les interconnexions et la synchronisation entre le DUV et le PowerPC. Celui-ci est écrit en VHDL et est synthétisé par l'outil ISE de Xilinx.

Du côté haut niveau, il permet d'établir une interaction avec le logiciel de gestion des données par le bus PLB du processeur PowerPC. Cette interaction est faite en utilisant l'interface bidirectionnelle IPIF de Xilinx, qui dans notre cas inclus 128 registres de 32 bits. Du côté bas niveau, il permet la communication directe avec le DUV par de l'électronique programmable.

Les signaux d'entrées/sorties de l'encapsuleur en relation avec le DUV sont les suivants :

1) 62 registres de 32 bits configurables en entrée (1984 bits);
2) 62 registres de 32 bits configurables en sortie (1984 bits);
3) 1 signal d'horloge sortant;
4) 1 signal 'reset' sortant;
5) 1 signal 'done' entrant optionnel.

Il va de soi que la somme des bits configurables en entrée et en sortie doit être inférieure ou égal à 1984, et cela par tranches de 32 bits. La fréquence du signal d'horloge est basée sur la fréquence du bus et est divisible par un entier d, comme le montre l'équation suivante :

$$f = \frac{100M}{d} \quad ,\forall\left(d \in \left[1,2^{31}\right[\right) \tag{2.1}$$

2.4.5 DUV

Le DUV est la conception numérique que l'on désire vérifier. Le cœur correspondant est écrit en VHDL et est synthétisé par l'outil ISE de Xilinx. Ce cœur est la partie du projet de développement FPGA qui est la plus variable en fonction des conceptions à tester. Il y a donc des limites physiques à respecter en termes de ressources au niveau du FPGA Virtex-4 FX12 qui est utilisé par notre carte matérielle.

Pour les fins du projet, nous assumons que le code VHDL fournit pour le DUV est prêt à être synthétisé, donc exempt d'erreur de syntaxe. Nous allons vérifier trois conceptions différentes de DUV dans le cadre de cet ouvrage, au chapitre 3.

2.5 Ordinateur

Un ordinateur PC standard est nécessaire ayant le système d'exploitation Windows XP 32 bits ou Windows Vista 32 bits. Une mémoire système de 2 GB et un processeur à double cœur sont considérés comme des spécifications minimales pour le fonctionnement du logiciel de contrôle du HIL et MATLAB. Dans le cas où le projet est en développement, de 4 GB à 6 GB de mémoire et un processeur à quatre cœurs sont recommandés. Et cela car il est possible de faire fonctionner en même temps deux instances du logiciel ISE, le logiciel EDK, le logiciel de contrôle du HIL et MATLAB.

L'ordinateur utilisé pour les tests est une station de travail HP xw6600 utilisant les composantes suivantes :

- deux processeurs Intel Xeon 5260 (double cœurs cadencés à 3.33 GHz);
- 4GB de mémoire RAM DDR2 (667 MHz ECC);
- un disque dur SATA de 250GB (7200 rpm);
- le système d'exploitation Windows Vista Entreprise 64 bits.

Nous avons testé les fonctionnalités des outils de Xilinx sur une plateforme Vista 64 bits, mais les tests ont démontré que bien qu'il est possible de faire fonctionner ISE, certains fichiers exécutables de l'environnement EDK ne sont pas compatible avec un système d'exploitation 64 bits. Nous devrons attendre la version 12.1 ou une version ultérieure des outils de Xilinx pour que ce support soit effectif.

Le système d'exploitation utilisé pour faire fonctionner les outils Xilinx et MATLAB pour les fins du projet est Microsoft Windows XP version 5.1 (Build 2600: Service Pack 3). Nous avons utilisé ce système à l'intérieur d'une machine virtuelle VmWare, et cela dans le but d'augmenter la portabilité et la gestion des configurations. La machine virtuelle est exécutée avec VmWare Player 3 sous le système d'exploitation Vista 64 bits.

2.5.1 Outils de Xilinx

La suite d'outils de conception numérique de la compagnie Xilinx est nécessaire pour l'utilisation de la plateforme FPGA. La conception numérique est centrée sur l'outil ISE (bas niveau d'abstraction), qui regroupe tous les outils électroniques nécessaires, comme l'écriture du VHDL, la simulation, la synthèse, le placement et routage, etc. L'outil ISim, intégré à ISE, à été utilisé pour effectuer les simulations nécessaires au projet.

Les parties reliées à l'utilisation d'une carte de développement et/ou aux processeurs embarqués matériels ou synthétisés sont gérés par la suite XPS. Cette dernière comprend l'outil EDK (moyen niveau d'abstraction) qui gère le développement embarqué et l'outil

SDK (haut niveau d'abstraction) qui gère la partie génie logiciel. Pour le projet en cours, nous n'utiliserons pas le SDK, car nous n'utilisons pas de système d'exploitation pour le PowerPC et nous n'avons pas une architecture logicielle complexe à supporter.

Les outils de Xilinx qui ont été installés pour le bon fonctionnement du projet sont :

1) ISE WebPack, version 10.1.03 (Service Pack 3);
2) EDK, version 10.1.03 (Service Pack 3);
3) ISE IP update, version 10.1.03 (Service Pack 3).

Il est intéressant de noter que la suite gratuite de l'outil ISE, nommée WebPack, peut être utilisée dans notre cas, car nous utilisons le plus petit FPGA de la gamme FX du Virtex-4. On peut voir la liste des FPGA supporté par le WebPack à la Figure 2.6.

ISE Design Suite Device Support	ISE WebPACK® Tool	ISE Design Suite Logic Edition Embedded Edition DSP Edition System Edition
Virtex® FPGAs	Virtex-4 FPGAs LX: XC4VLX15, XC4VLX25 SX: XC4VSX25 FX: XC4VFX12	Virtex-4 FPGAs LX: All SX: All FX: All
	Virtex-5 FPGAs LX: XC5VLX30, XC5VLX50 LXT: XC5VLX20T - XC5VLX50T FXT: XC5VFX30T	Virtex-5 FPGAs LX: All LXT: All SXT: All FXT: All
	Virtex-6 FPGAs XCLX75T	Virtex-6 FPGAs All

Figure 2.6 FPGA supporté par la suite d'outil ISE de Xilinx.
Tirée de Xilinx (2010f)

2.5.2 Projet de développement FPGA

Cette partie regroupe l'ensemble des composantes matérielles et logicielles programmées dans le FPGA. Elle est réalisée et implémentée grâce aux outils ISE et EDK de Xilinx.

Les cœurs matériels principaux qui sont synthétisés sont les suivants :

- PPC405 Virtex4 (Cœur de l'encapsulateur du PowerPC matériel);
- XPS UART Lite (Cœur pour la console RS-232);
- XPS MCH EMC (Cœur pour le contrôle de la mémoire SRAM);
- XPS Ethernet Lite MAC (Cœur pour la communication réseau Ethernet 10/100);
- Xilinx FIT (Cœur pour l'interruption de mise à jour des compteurs TCP);
- DUV Wrapper (Cœur d'encapsulation du DUV);
- DUV (Cœur du DUV).

Le cœur Ethernet Lite 10/100 qui est présent gratuitement dans le EDK a été utilisé. Bien que moins performant que les cœurs Gigabits qui utilise le MAC matériel, il est plus facile d'utilisation, car aucune gestion de licence n'est nécessaire.

Sur l'ordinateur, les outils de Xilinx vont produire un fichier .BIT qui contient les informations sur la programmation des cellules logiques du FPGA et un fichier .ELF qui contient le code informatique du processeur embarqué. Ces deux fichiers seront envoyés à la plateforme FPGA par le lien de communication JTAG. Du côté de l'ordinateur, le lien JTAG est possible grâce à un module d'interface USB nommé : « Xilinx Platform Cable USB 2 ».

2.5.3 Logiciel MATLAB

MATLAB permet de générer les stimuli, d'effectuer les traitements théoriques de la conception et d'afficher les résultats. Le processus du HIL permet de remplacer la partie des traitements effectués dans MATLAB par le DUV qui est exécuté sur la plateforme matérielle.

Les traitements théoriques constituent notre modèle de référence (« Golden Model »). Les résultats de ce modèle sont donc comparés aux résultats générés par le DUV à travers l'infrastructure HIL.

Le logiciel MATLAB version 2006b est utilisé sur l'ordinateur. Le détail relatif aux versions des différents modules présents, tels que fournis par la commande 'ver' de la console MATLAB, sont donnés à l'annexe II.

Les temps d'exécution sous l'environnement MATLAB étant dépendant de la puissance et des caractéristiques de l'ordinateur utilisé, nous avons effectué un test de vitesse de la version de MATLAB utilisée. Les résultats fournis par la commande 'bench' de la console MATLAB, sont illustrés à la Figure-A III-1 et au Tableau-A III-1 de l'annexe III. Ces résultats ne sont comparables qu'avec des ordinateurs ayant la même version de MATLAB.

2.5.4 Logiciel de contrôle du HIL et d'interface graphique

Ce logiciel, programmé en C++ sous l'environnement Microsoft Visual Studio 2005, est le point central des communications entre le logiciel MATLAB et la plateforme matérielle. Le logiciel est conçu pour démarrer l'engin de MATLAB afin de pouvoir exécuter des opérations interprocessus à partir du logiciel C++.

Le GUI qui a été conçu pour contrôler le HIL utilise la librairie DXUT de l'engin DirectX. Le développement a été effectué en utilisant la version 9 de DirectX, mais les fonctionnalités sont supportées sans problème dans les versions suivantes de DX. On peut voir un exemple de la représentation graphique à la Figure 2.7.

Dans un premier temps, lors d'un premier démarrage du logiciel, une phase de configuration doit être engagée. L'emplacement des fichiers MATLAB pour la génération des stimuli, le traitement du DUV modélisé et l'affichage des résultats doivent être spécifiés. Il est aussi nécessaire de spécifier l'emplacement des fichiers VHDL du DUV et le futur emplacement

du projet de développement FPGA. Ces opérations sont situées dans la partie supérieure gauche du GUI.

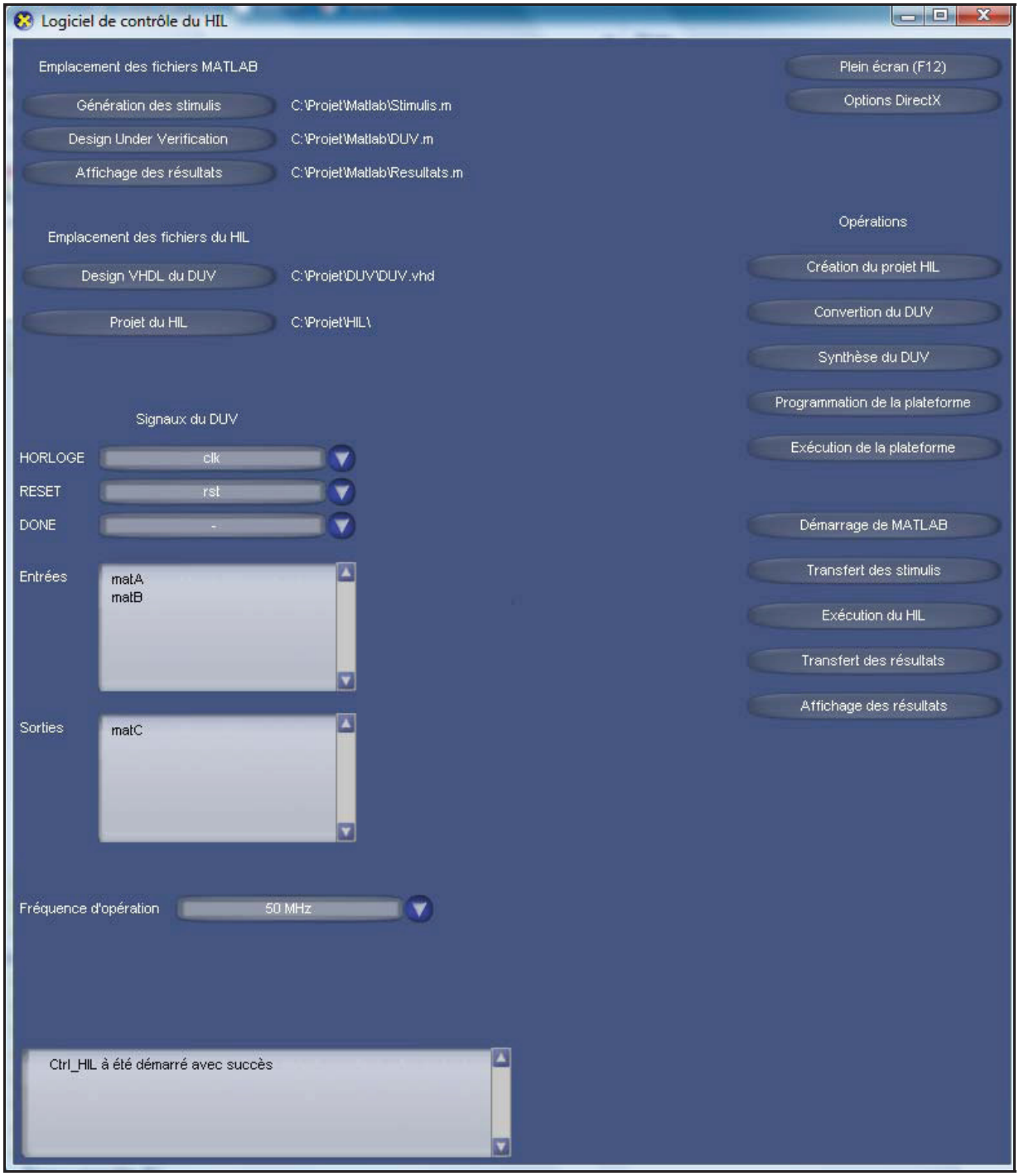


Figure 2.7 Logiciel de contrôle du HIL.

Par la suite, cinq opérations successives sont nécessaires pour rendre la plateforme FPGA fonctionnelle. Celles-ci sont situés dans la partie supérieure droite du GUI, soit :

1) la création du projet HIL;
2) la conversion du DUV;
3) la synthèse du DUV;
4) la programmation de la plateforme;
5) la mise en route de la plateforme.

Enfin, il reste cinq opérations qui peuvent être exécutés suite à la complétion des cinq étapes précédentes. La première permettant le démarrage contrôlé de MATLAB. Ensuite, le fichier MATLAB de création des stimuli est exécuté, puis ces données sont envoyées au logiciel de gestion des données situé sur la plateforme matérielle. Suite à l'exécution des opérations du processus HIL, les données résultantes du DUV sont ensuite retournées à MATLAB pour être affichées et comparées avec le résultat des traitements originaux produits dans MATLAB. La concordance des résultats entre MATLAB et le HIL permet de conclure en une réussite de la vérification du DUV.

CHAPITRE 3

CONCEPTIONS À L'ESSAI

3.1 Cas soumis aux essais

Plusieurs types de conceptions différentes seront testées dans le cadre du projet. Et cela dans le but de comparer la performance du HIL en relation avec les traitements théoriques présents dans les différentes conceptions.

Dans le cas du présent document, nous présentons trois cas de figure reliés à différents domaines. Les sections qui suivent expliquent plus en profondeur les cas de figure testés, chacun représentant un DUV.

3.1.1 Cas 1 – Calcul matriciels

Ce premier cas est le plus simple des trois cas à l'essai. Cet essai propose le produit matriciel signé de deux matrices carrées de dimension 3x3.

Voici un rappel de la théorie :

> Si A = (a_{ij}) est une matrice de type (m,n) et B = (b_{ij}) est une matrice de type (n,p), alors leur produit, noté AB = (c_{ij}) est une matrice de type (m,p) donnée par :

$$\forall i, j, n \in \mathrm{N}^*, \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \ldots + a_{in} b_{nj} \tag{3.1}$$

L'algorithme du DUV que nous allons vérifier, ainsi que le contenu des matrices utilisées sont les suivants :

$$C = A \times B \tag{3.2}$$

$$A = \begin{bmatrix} 41 & -58 & -63 \\ -29 & 33 & 36 \\ -54 & 26 & -41 \end{bmatrix} \tag{3.3}$$

$$B = \begin{bmatrix} 23 & 96 & 28 \\ 17 & 70 & 30 \\ 63 & -86 & 70 \end{bmatrix} \tag{3.4}$$

Les éléments des matrices A et B comportent des entiers signés 8 bits, tandis que des entiers signés 16 bits contiennent le résultat de la multiplication matricielle dans la matrice C.

Ce test représente un calcul de base présent dans beaucoup d'opérations effectuées avec le logiciel MATLAB. La quantité de données à transférer de la plateforme logicielle à la plateforme matérielle est faible et le nombre d'opérations nécessaires est faible.

3.1.2 Cas 2 – Filtre de Sobel

Ce deuxième cas présente un algorithme de détection de contour nommé 'Filtre de Sobel', couramment utilisé en vision numérique. Ce domaine d'application demande une grande quantité de calculs lors du traitement d'images numériques. Avec l'accroissement de la taille des images capturées et l'avènement des caméras de format HD, la capacité de calculs nécessaire demandera de plus en plus l'utilisation de plateforme matérielle (FPGA) au lieu des solutions conventionnelles.

L'algorithme de Sobel est un opérateur qui utilise des matrices de convolution. La convolution est une opération mathématique simple qui est fondamentale pour de nombreux opérateurs de traitement d'images numériques. La convolution fournit un moyen de multiplier ensemble deux matrices, généralement de tailles différentes, mais de la même dimension, pour produire une troisième matrice de la même dimension. Dans notre contexte, la première matrice est l'image d'entrée, la deuxième matrice est le noyau de convolution et la troisième matrice est l'image de sortie. La valeur du pixel de sortie est la somme locale pondérée de chacun des pixels d'entrée, où les poids proviennent du noyau de convolution.

En vision numérique, la convolution utilise les huit points voisins du point visé en plus de celui-ci, pour un total de neuf points, ce qui amène un noyau de convolution d'une taille de 3x3. Pour calculer les approximations des dérivées horizontales et verticales, on utilise deux matrices de convolution différentes pour détecter les contours horizontaux et verticaux. On peut donc obtenir le gradient résultant en x et en y en utilisant les noyaux suivants (où G_h est le gradient horizontal, G_v est le gradient vertical et P est représenté comme étant l'image) :

$$G_h = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * P \tag{3.5}$$

$$G_v = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * P \tag{3.6}$$

On peut obtenir la norme du gradient (G) en combinant le gradient horizontal et le gradient vertical comme suit :

$$G = \sqrt{G_h^{\,2} + G_v^{\,2}} \tag{3.7}$$

Pour obtenir une détection de contour plus générale et pour optimiser les calculs, nous utiliserons un calcul simplifié pour le gradient :

$$G = |G_h| + |G_v| \tag{3.8}$$

L'algorithme du DUV que nous allons vérifier sera le suivant :

Si $P = (p_{ij})$ est un matrice de type (m,n), et que c_h et c_v sont des matrices de type (3,3). De plus si $G_h = (h_{ij})$ et $G_v = (v_{ij})$ sont des matrices de type (m-2,n-2), alors le gradient résultant noté $G = (g_{ij})$ est une matrice de type (m-2,n-2) donnée par :

$$c_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.9}$$

$$c_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$h_{ij} = \sum_{x=1}^{3}\sum_{y=1}^{3} c_{h_{xy}}\, p_{(i+x-1)(j+y-1)} \quad , \forall\left(i \in [1, m-2] \wedge j \in [1, n-2]\right) \tag{3.11}$$

$$v_{ij} = \sum_{x=1}^{3}\sum_{y=1}^{3} c_{v_{xy}}\, p_{(i+x-1)(j+y-1)} \quad , \forall\left(i \in [1, m-2] \wedge j \in [1, n-2]\right) \tag{3.12}$$

$$g_{ij} = \min\left(|h_{ij}| + |v_{ij}|, 255\right) \tag{3.13}$$

Étant donné qu'une image utilisant des pixels de 8 bits est utilisée dans notre cas, les éléments de la matrice G doivent être limités à la valeur 255 en utilisant une segmentation manuelle à borne supérieure. Le DUV gère des fenêtres de dimension 12x12 en entrée, soit un total de 1440 bits, et retourne une image filtrée résultante de dimension 10x10 en sortie. La différence entre la fenêtre d'entrée et la sortie provient du fait que l'algorithme de convolution utilise chacun des pixels voisins au pixel évalué. Le pixel de sortie (1,1) correspond donc au pixel d'entrée (2,2) entouré de ces huit voisins : (1,1), (2,1), (3,1), (1,2), (3,2), (1,3), (2,3) et (3,3).

Pour les fins du test nous utiliserons à la base l'image représentée à la Figure 3.1, de dimension 262x102 et représentant le logo du laboratoire LACIME de l'ÉTS. Cette image est préalablement convertie en 256 tons de gris avant son utilisation par le système.

Figure 3.1 Image d'entrée utilisée pour le cas 2.
Tirée de l'École de technologie supérieure

Il faudra plusieurs itérations pour appliquer le filtre de Sobel à une image standard comme celle choisie. Cette image demandera 260 itérations du DUV pour procéder à la complétion des traitements résultant en l'image finale. Le résultat sera une image de taille 260x100, où le pixel de contour de l'image original sera perdu à cause de l'opération de convolution.

En résumé, ce test représente un calcul plus complexe que le cas précédent. La quantité de données à transférer de la plateforme logicielle à la plateforme matérielle est élevée et le nombre d'opérations nécessaires est moyen.

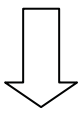
3.1.3 Cas 3 – Décodage par attaque en force

Ce troisième cas présente un algorithme de décodage par attaque en force (« brute-force attack »). Ce type de décodage demande une très grande quantité de calculs qui résulte en un temps de décodage assez long lorsque exécuté sur des PC conventionnels. Ce type d'application sera de plus en plus utilisé sur des plateformes matérielles de type FPGA, au lieu des solutions conventionnelles, ce qui en fait un bon cas d'étude.

L'hypothèse de départ de ce test est la connaissance du processus d'encodage et du message encodé. On cherche alors la première valeur donnant le message correspondant en essayant toutes les possibilités. Le code du DUV développé fait l'essai d'une possibilité, l'encode et compare le résultat de l'encodage avec le message à décoder. Lorsque les deux valeurs sont identiques, le processus est arrêté, la valeur du message original est assignée en sortie et le signal de fin est activé.

L'algorithme utilisé permet d'encoder sans collision une valeur 32 bits sur 32 bits et s'exécute en deux étapes. Premièrement, les bits du message original sont permutés selon l'ordre suivant (le bit le moins significatif est du côté droit et porte le numéro 1) :

(32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1)



(30,9,11,18,14,20,25,32,27,21,5,12,17,10,22,2,6,4,31,19,28,13,16,29,8,7,26,24,1,23,15,3)

Ce résultat subit deuxièmement un 'OU exclusif' avec la valeur binaire qui suit :

00010110 11011010 11111011 10000100

Pour des fins de simplicité dans le cadre du projet, un algorithme de base réversible a été utilisé, mais on peut aisément considérer l'implantation d'algorithmes de hachage cryptographique non-réversibles connus et plus complexes tel que le MD5 et le SHA.

Pour ce test nous utiliserons le message encrypté suivant (en format hexadécimal et binaire) :

9E830987 (10011110 10000011 00001001 10000111)

En résumé, ce test représente un calcul plus complexe que le cas précédent. La quantité de données à transférer de la plateforme logicielle à la plateforme matérielle est faible et le nombre d'opérations nécessaires est très élevé.

CHAPITRE 4

RÉSULTATS OBTENUS

4.1 Détection des anomalies

Avant l'atteinte de résultats, nous avons pu constater que l'infrastructure de vérification utilisant le HIL nous a permis de détecter dès les premiers essais les non-conformités du code VHDL du DUV. Par exemple, il a été possible de détecter rapidement l'utilisation d'un DUV nécessitant des ressources trop importantes pour la taille du FPGA utilisé. Lorsque les résultats du DUV ne correspondaient pas à ceux de notre modèle de référence de MATLAB, il était nécessaire de réexaminer le code VHDL du DUV. Cela nous a permis de manière plus précoce de détecter, par exemple, des erreurs relatives à la mauvaise compréhension du modèle de référence.

Bien que le temps économisé à détecter toutes ces erreurs plus rapidement est difficile à quantifier avec précision, il est évident que cela contribue à accélérer le processus de vérification dans sa globalité.

4.2 Présentation des résultats

Les sections suivantes présentent les résultats obtenus pour les trois cas de figures à l'essai, tel qu'expliqué au chapitre 3.

Les temps inscrits pour des fins de comparaison sont des temps moyennés (T_{moy}) calculés à partir des temps mesurés (t) moyennés sur n essais, comme suit :

$$T_{moy} = \sum_{i=1}^{n} \frac{t_i}{n} \quad , \forall n = 3 \tag{4.1}$$

4.2.1 Cas 1 – Calculs matriciels

Le résultat obtenu par le HIL sur la plateforme matérielle équivaut à celui effectué dans MATLAB et produit la matrice suivante :

$$C = \begin{bmatrix} -4012 & 5294 & -5002 \\ 2162 & -3570 & 2698 \\ -3383 & 162 & -3602 \end{bmatrix} \tag{4.2}$$

Le produit matriciel des deux matrices de taille 3x3 prend un temps moyen de 24 µs dans MATLAB. Et cela en relation avec les performances données à l'annexe II pour la version de MATLAB qui est exécutée sur l'ordinateur utilisé pour les tests.

Pour calculer le temps nécessaire équivalent en utilisant le HIL il est nécessaire d'additionner plusieurs temps, car on doit considérer le temps d'extraction des stimuli à partir de MATLAB, le temps de communication aller-retour vers la plateforme embarquée, le temps nécessaire au DUV et le temps d'envoi des résultats à MATLAB. Pour un DUV cadencé à 50 MHz, les temps moyens des opérations effectuées pour le traitement par le HIL sont indiqués au Tableau 4.1.

Tableau 4.1 Temps d'opération du traitement par le HIL du cas 1

Opérations	**Temps moyens d'opération**
DUV - Calcul matriciel	20 ns
Encapsulation du DUV	2 µs

Logiciel de gestion des données	231.7 µs
Communication TCP	531.2 µs
Logiciel de contrôle du HIL	11 µs

Le temps moyen requis par le traitement par le HIL du cas 1 est de 775.9 µs. Cette valeur sera comparée à celles des autres cas étudiés et sera analysée plus en détails à la section 4.3.

Évidemment cette valeur est élevée en rapport avec le temps nécessaire au DUV pour effectuer son opération. Bien que cette valeur soit aussi plus élevé que le temps nécessaire par MATLAB pour effectuer le traitement, il ne faut pas oublier que ce temps est négligeable par rapport à celui que nécessite la vérification complète de la conception numérique étudiée.

4.2.2 Cas 2 – Filtre de Sobel

Le résultat de traitement d'image obtenu par le HIL sur la plateforme matérielle équivaut à l'image résultante que l'on obtient dans MATLAB, soit l'image de dimension 260x100 que l'on peut voir à la Figure 4.1.

Figure 4.1 Image de sortie résultante du cas 2.

On peut voir très nettement les contours associés au logo du LACIME. Et cela tant pour les arêtes verticales que les arêtes horizontales.

L'opérateur de Sobel appliqué sur cette image prend un temps de calcul de 11 msec dans MATLAB. Pour un DUV cadencé à 50 MHz, les temps moyens des opérations effectuées pour le traitement par le HIL sont indiqués au Tableau 4.2.

Tableau 4.2 Temps d'opération du traitement par le HIL du cas 2

Opérations	**Temps moyens d'opération**
DUV - Filtre de Sobel 10x10	1.3 μs (1 itération)
Encapsulation du DUV	515.7 μs
Logiciel de gestion des données	181.18 ms
Communication TCP	140.62 ms
Logiciel de contrôle du HIL	7.8 ms

Le temps moyen requis par le traitement par le HIL du cas 2 est de 330.12 ms. Ce temps est évidemment plus élevé qu'au cas 1, car la complexité du DUV est plus importante et la quantité de données à traiter est bien supérieure.

4.2.3 Cas 3 – Décodage par attaque en force

Le résultat de décodage obtenu par le HIL sur la plateforme matérielle équivaut à la valeur résultante que l'on obtient dans MATLAB, soit (en format hexadécimal et binaire) :

6015E82E (11000000 00101011 11010000 0101110)

Le nombre d'itérations nécessaires dépend de la position à partir de 0 de la solution à l'intérieur de l'intervalle $[0,2^{32}[$. Pour en arriver à ce résultat, le DUV exécuté par MATLAB ou par la plateforme matérielle doivent faire 1 612 048 430 itérations avant de converger vers la solution.

Le décodage par attaque en force sur le message encodé prend un temps de calcul interpolé de 163984 s dans MATLAB, soit 45 h, 33m et 4 s. Ce qui est considérable en fonction de la puissance de calcul des ordinateurs actuels.

Tableau 4.3 Temps d'opération du traitement par le HIL du cas 3

Opérations	**Temps moyens d'opération**
DUV – Décodage par attaque en force	20 ns (1 itération)
Encapsulation du DUV	32.24 s
Logiciel de gestion des données	167.1 µs
Communication TCP	630.6 µs
Logiciel de contrôle du HIL	7 µs

Pour un DUV cadencé à 50 MHz, les temps moyens des opérations effectuées pour le traitement par le HIL sont indiqués au Tableau 4.3.

Le temps moyen requis par le traitement par le HIL du cas 3 est de 32.241 s.

4.3 Interprétation des résultats

Pour les trois cas de figure qui étaient à l'essai, les résultats de la vérification ont été positifs. Le HIL a donc permis de vérifier que le DUV avait un comportement conforme aux traitements théoriques testés sous MATLAB. Ce qui est un point excellent à lui seul, sans considérer les autres avantages potentiels.

Idéalement, il serait possible de toujours obtenir une accélération due au fait que l'on exécute les calculs sur une plateforme matérielle. Mais malheureusement, le temps de communication qui doit être ajouté peut amener le temps global du HIL à être plus élevé dans certains des cas étudiés.

4.3.1 Comparaison des performances du HIL vs MATLAB

Au Tableau 4.4, on peut voir la comparaison des performances pour les 3 cas que nous avons mis à l'épreuve. On peut y voir que dans tous les cas testés les performances de calculs sont grandement améliorées en utilisant une plateforme matérielle versus l'utilisation standard du logiciel MATLAB. Dans le cas où beaucoup de données doivent être transférées et que le temps de calcul est relativement court, on voit que la latence du HIL influence à la baisse le ratio global de performance (voir le cas 1 et 2). Par contre, dans le cas où l'on effectue des calculs plus complexes et que la quantité de données est restreinte, on voit le ratio global de performance augmenter à un niveau impressionnant (voir le cas 3).

Tableau 4.4 Comparaison des performances entre le HIL et MATLAB

Cas	Temps de calcul du HIL	Latence du HIL	Temps de calcul de MATLAB	Ratio de calcul HIL/MATLAB	Ratio global HIL/MATLAB
1	2 µs	773.9 µs	24 µs	12	0.031
2	517.7 µs	329.60 ms	11 ms	21.25	0.033
3	32.240 s	804.7 µs	163984 s	5086.35	5086.23

Les cas 1 et 2 montrent des performances plus modestes au niveau du ratio global. Différemment, le cas 3 se démarque avec une performance étonnante de 5086x, digne d'une plateforme de traitement matériel FPGA. Étant donné que nous avons testé des cas extrêmement opposés, la marge est importante entre les deux premiers cas et le cas 3. Il est donc raisonnable de penser que beaucoup de DUV futurs pourraient obtenir un ratio global de performance supérieur à 1. Ce qui nous permettrait d'obtenir un gain de performance direct et donc d'augmenter l'accélération globale, ce qui est l'un des avantages visés.

Il est à noter que, même en présence du cas 1 ou 2, l'accélération globale du processus de vérification engendrée par la HIL est tout de même présente. Et cela, car il ne faut pas oublier que nous n'avons pas eu à écrire et valider un programme de test VHDL. De plus, certaines itérations de conception ont pu être sauvées en effectuant la vérification utilisant le HIL. Le temps économisé pour ces différentes raisons est en général non-négligeable, car il correspond à plusieurs heures de travail.

CONCLUSION

Nous avons posé l'hypothèse qu'il serait possible de faciliter et d'accélérer globalement le processus de vérification en utilisant une approche de vérification comparative qui utilise le « Hardware In the Loop ».

Dans ce mémoire, nous avons montré le bon fonctionnement d'un système de vérification de conception numérique utilisant une méthode comparative utilisant le HIL et le logiciel MATLAB. Il est maintenant possible d'affirmer que grâce à ce système le HIL peut permettre une accélération du temps total de traitement des données.

Bien qu'il soit intéressant d'obtenir une accélération tangible au niveau du temps de traitements du modèle étudié, il faut voir l'accélération globale visée à un niveau d'abstraction plus élevé. En fait, à lui seul, le temps d'écriture des stimuli exacts du 'testbench' qui a été sauvé, n'a presque pas de prix pour un concepteur numérique. Sauver cette étape et pouvoir comparer les résultats sans avoir à faire de quelconques étapes fastidieuses est en soit un avantage intéressant de l'approche par le HIL.

Le fait d'utiliser les mêmes stimuli et de pouvoir comparer dans le même environnement (MATLAB) les résultats des 2 méthodes permet de faciliter le processus de vérification. Et cela en comparant avec la méthode traditionnelle utilisant l'approche du 'testbench'.

L'accélération globale est obtenue en sauvant, lors de l'écriture du 'testbench', la reproduction des stimuli judicieux à l'environnement de vérification. De plus l'accélération potentielle du traitement des données et la vérification sur la plateforme réelle visée par l'application sont des avantages qui bonifient l'accélération globale. À cela s'ajoute un autre aspect qui n'a pas été exploité pendant ce projet et qui se rapporte à l'accélération potentielle de la phase de validation. En effet, le HIL pourrait bonifier les simulations quelques fois trop longues et fastidieuses effectuées pendant cette phase.

En conclusion, nous pouvons affirmer qu'il est possible de faciliter et d'accélérer globalement le processus de vérification et de validation en utilisant une approche de vérification comparative qui utilise le HIL.

RECOMMANDATIONS

Nous recommandons la continuité des travaux portant sur la vérification de conceptions numériques en utilisant une approche de HIL. Il serait intéressant de combiner l'infrastructure HIL et l'infrastructure SystemC développée par M. Jean-François Boland. Pour se faire, il serait possible de remplacer l'interface de connexion actuelle envers MATLAB par un lien à l'infrastructure SystemC. Les fonctionnalités et les possibilités futures de vérification de conceptions numériques seraient alors grandement bonifiées. De plus, cette infrastructure cadre bien avec la méthodologie ESL en utilisant SystemC comme langage de modélisation de haut niveau.

Dans le futur, les outils utilisés de la compagnie Xilinx pourraient être migrés à la version 12.1 pour une simplicité et des performances accrues. Une plateforme matérielle utilisant un FPGA plus récent, le Virtex-5 FX30T, pourrait aussi être utilisée pour plus de performance et de versatilité.

ANNEXE I

ÉVOLUTION DE L'ARCHITECTURE POWERPC

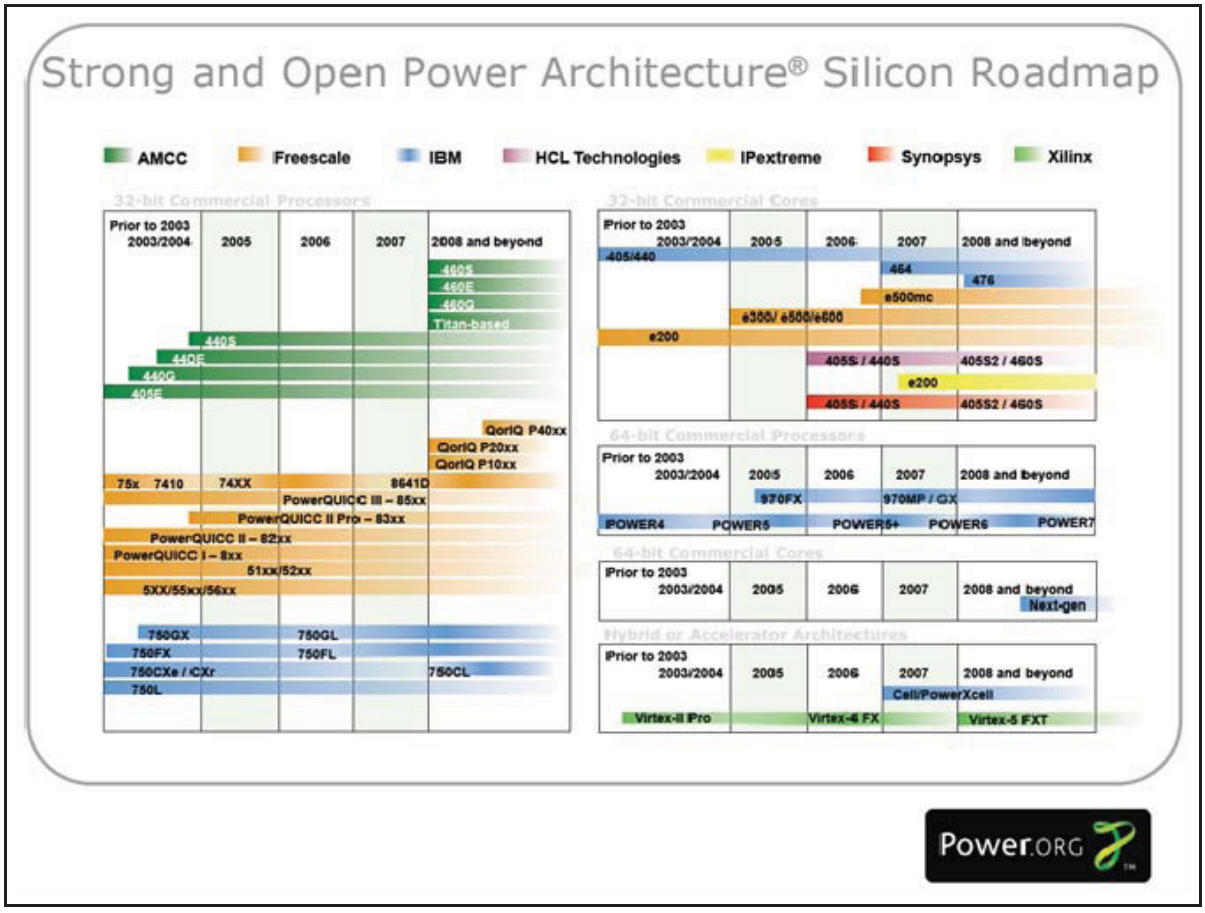


Figure-A I-1 Évolution des processeurs PowerPC.
Tirée de Behmann (2009, p. 10)

ANNEXE II

VERSIONS DE MATLAB ET DE SES MODULES INTÉGRÉS

MATLAB	Version 7.3.0.267	(R2006b)
Simulink	Version 6.5	(R2006b)
Bioinformatics Toolbox	Version 2.4	(R2006b)
Communications Blockset	Version 3.4	(R2006b)
Communications Toolbox	Version 3.4	(R2006b)
Control System Toolbox	Version 7.1	(R2006b)
Curve Fitting Toolbox	Version 1.1.6	(R2006b)
Data Acquisition Toolbox	Version 2.9	(R2006b)
Database Toolbox	Version 3.2	(R2006b)
Distributed Computing Toolbox	Version 3.0	(R2006b)
Embedded Target for TI C2000 DSP(tm)	Version 2.1	(R2006b)
Embedded Target for TI C6000 DSP(tm)	Version 3.1	(R2006b)
Filter Design HDL Coder	Version 1.5	(R2006b)
Filter Design Toolbox	Version 4.0	(R2006b)
Financial Toolbox	Version 3.1	(R2006b)
Fixed-Point Toolbox	Version 1.5	(R2006b)
Fuzzy Logic Toolbox	Version 2.2.4	(R2006b)
Image Acquisition Toolbox	Version 2.0	(R2006b)
Image Processing Toolbox	Version 5.3	(R2006b)
Instrument Control Toolbox	Version 2.4.1	(R2006b)
Link for Code Composer Studio	Version 2.1	(R2006b)
MATLAB Compiler	Version 4.5	(R2006b)
Mapping Toolbox	Version 2.4	(R2006b)

Neural Network Toolbox	Version 5.0.1	(R2006b)
Optimization Toolbox	Version 3.1	(R2006b)
Partial Differential Equation Toolbox	Version 1.0.9	(R2006b)
Real-Time Workshop	Version 6.5	(R2006b)
Real-Time Workshop Embedded Coder	Version 4.5	(R2006b)
Robust Control Toolbox	Version 3.1.1	(R2006b)
Signal Processing Blockset	Version 6.4	(R2006b)
Signal Processing Toolbox	Version 6.6	(R2006b)
SimMechanics	Version 2.5	(R2006b)
SimPowerSystems	Version 4.3	(R2006b)
Simulink Control Design	Version 2.0.1	(R2006b)
Simulink Fixed Point	Version 5.3	(R2006b)
Spline Toolbox	Version 3.3.1	(R2006b)
Stateflow	Version 6.5	(R2006b)
Statistics Toolbox	Version 5.3	(R2006b)
Symbolic Math Toolbox	Version 3.1.5	(R2006b)
System Identification Toolbox	Version 6.2	(R2006b)
Virtual Reality Toolbox	Version 4.4	(R2006b)
Wavelet Toolbox	Version 3.1	(R2006b)
xPC Target	Version 3.1	(R2006b)

ANNEXE III

TEST DE LA VITESSE RELATIVE DE MATLAB

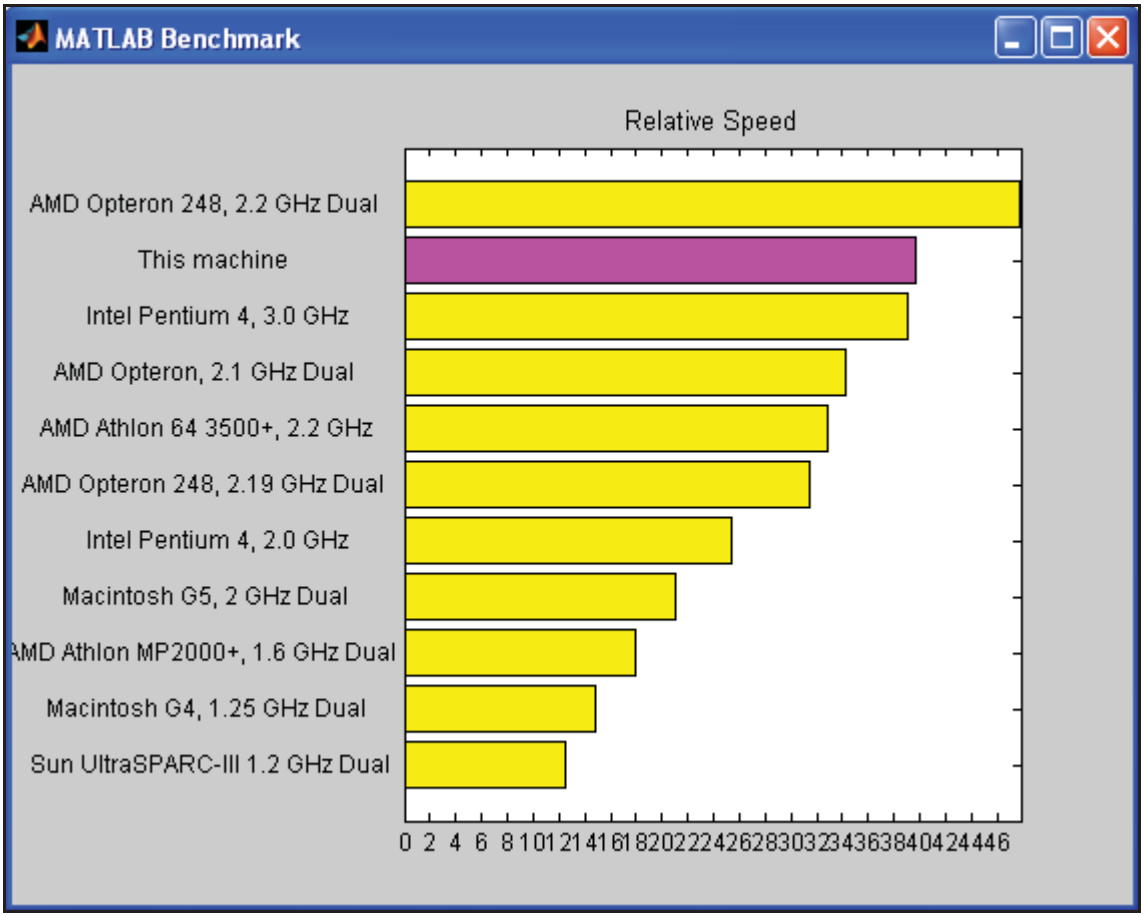


Figure-A III-1 Vitesse relative de la machine hôte en relation avec des machines typiques.

Tableau-A III-1 Temps d'opération d'algorithmes standards exécutés sur la machine hôte et les machines typiques

MATLAB Benchmark

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
AMD Opteron 248, 2.2 GHz Dual	0.2461	0.2842	0.2077	0.5368	0.4977	0.3228
This machine	0.2856	0.3565	0.2108	0.3925	0.5003	0.7726
Intel Pentium 4, 3.0 GHz	0.2403	0.4679	0.3003	0.5523	0.5849	0.4200
AMD Opteron, 2.1 GHz Dual	0.3273	0.3246	0.2315	0.6436	0.3695	1.0286
AMD Athlon 64 3500+, 2.2 GHz	0.2966	0.2858	0.5427	0.5904	0.6163	0.7131
AMD Opteron 248, 2.19 GHz Dual	0.3158	0.3065	0.5338	0.6880	0.6251	0.7166
Intel Pentium 4, 2.0 GHz	0.3892	0.7746	0.4196	0.9116	0.7826	0.6530
Macintosh G5, 2 GHz Dual	0.2284	0.3962	0.3591	1.8114	1.3693	0.5899
AMD Athlon MP2000+, 1.6 GHz Dual	0.6978	0.7124	0.4908	1.1942	0.8679	1.6216
Macintosh G4, 1.25 GHz Dual	0.6362	0.8885	0.6492	1.5170	2.1924	0.8813
Sun UltraSPARC-III 1.2 GHz Dual	0.6419	0.6917	0.8945	1.4503	1.2380	3.1070

Place the cursor near a computer name for system and version details. Before using this data to compare different versions of MATLAB, or to download an updated timing data file, see the help for the bench function by typing 'help bench' at the MATLAB prompt.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Behmann, Fawzi. 19 octobre 2009. « *Power.org Overview and Announcements* ». In *Asia Power Architecture Conference 2009 (PAC 2009)*. (Taiwan, 14-19 octobre 2009). 34 p. En ligne.
< http://www.power.org/events/powercon09/taiwan09/Power_Overview_Announcments_Fawzi_Behmann.pdf >.

Bergeron, Janick. 10 février 2006. *Writing Testbenches using SystemVerilog, 1e éd.*. New York : Springer, 412 p.

Boland, Jean-François, A. Chureau, Claude Thibeault, Y. Savaria, François Gagnon et Z. Zilic. Juin 2004. *« An Efficient Methodology for Design and Verification of an Equalizer for a Software Defined Radio »*. In IEEE *Circuits and Systems 2004 (NEWCAS 2004).* (Montréal, 20-23 Juin 2004). p. 73-76. En ligne.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1359021 >.

Boland, Jean-François, Claude Thibeault, et Zeljko Zilic. Février 2005. « *Using MATLAB and SIMULINK in a SystemC verification environment* ». In *Proceedings* of *Design and Verification Conference 2005 (DVCon 2005).* (San Jose. 14-16 février 2005). En ligne (site de l'IML).
< http://www.iml.ece.mcgill.ca/people/professors/zilic/DVCon05.pdf >.

Bureau international des poids et mesures. 2006. « *Le système international d'unités* », 8e édition. Paris (France) : BIPM. 180 p. En ligne.
< http://www.bipm.org/utils/common/pdf/si_brochure_8.pdf >.

Byme, Joseph. 9 avril 2010. « *ARM Outmuscles Atom on Benchmark* ». The Linley Group. En ligne.
< http://blog.linleygroup.com/2010/04/arm-outmuscles-atom-on-benchmark.html >.

École de technologie supérieure. Laboratoire de Communication et d'Intégration de la Microélectronique. En ligne.
< http://lacime.etsmtl.ca >.

Giard, Pascal, Jean-François Boland, et Jean Belzile. 15 Octobre 2008. « *CORBA Communication Backplane for Design and Verification* ». In *Microsystems and Nanoelectronics Research Conference 2008 (MNRC 2008).* (Ottawa, 15 octobre 2008). p. 121-124. En ligne.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04683393 >.

Hamelin, Philippe, Pascal Bigras, Julien Beaudry, Sylvain Lemieux et Michel Blain. 2008. *« Hardware-in-the-loop Simulation of an Impedance Controlled Robot Using a Direct-Drive Test Bench »*. In *IEEE International Symposium on Industrial Electronics 2008 (ISIE 2008).* (Cambridge, 30 juin - 2 juillet 2008). p. 1281-1286. En ligne.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04677129 >.

Hylla, Kai, Jan-Hendrik Oetjens et Wolfgang Nebel. 2009. *« An Advanced Simulink Verification Flow Using SystemC »*. In *Languages for embedded systems and their applications*, sous la dir. de Radetzki, Martin. p.71-84. Lecture Notes in Electrical Engineering, volume 36, partie I. Stuttgart (Allemagne) : Springer. En ligne.
< http://www.springerlink.com/content/h78162t74782rk28/fulltext.pdf >.

IBM. 2 septembre 2006. *« Product Overview – PowerPC 405 CPU Core »*. En ligne.
< https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/3D7489A3704570C0872571DD0065934E/$file/PPC405_Product_Overview_20060902.pdf >.

Leitner, Jesse. Février 1996. « *Space technology transition using hardware in the loop simulation* ». In *Proceedings of IEEE Aerospace Applications Conference 1996.* (Aspen, 3-10 Février 1996). Volume 2, p. 303-311. En ligne.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00495985 >.

McGrath, Dylan. 27 juillet 2009. *« FPGA startups stare down giants and ghosts »*. *EE Times*. En ligne.
< http://eetimes.com/news/latest/showArticle.jhtml?articleID=218500007 >.

Short, Michael, et Michael J. Pont. Septembre 2005. « *Hardware in the Loop Simulation of Embedded Automotive Control Systems* ». In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems.* (Vienna, 13-16 septembre 2005). p. 226-231. En ligne.
< http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01520052 >.

Short, Kenneth L. 2009. *VHDL for Engineers*. Upper Saddle River : Pearson Education, 685 p.

TOP500.org. Novembre 2009. « *Processor Family share for 11/2009 »*. En ligne. < http://www.top500.org/stats/list/34/procfam >.

Xilinx. Mars 2005. « *Virtex-4 Overview »*, v2.1. En ligne. < http://www.electroniciens.aquitaine-limousin.cnrs.fr/IMG/ppt/CEN3_Virtex4.ppt >.

Xilinx. 24 mai 2006. « *ML401/ML402/ML403 Evaluation Platform, User Guide »*, UG080 v2.5. En ligne. < http://www.xilinx.com/support/documentation/boards_and_kits/ug080.pdf >.

Xilinx. 28 septembre 2007. « *Virtex-4 Family Overview »*, DS112 v3.0. En ligne. < http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf >.

Xilinx. 22 avril 2009. « *Virtex-6 FPGA Extended Overview* ». Xilinx Product Marketing. En ligne. < http://www.opensparc.net/pubs/preszo/09/brussels/12_MD_Virtex_6_Overview .pdf >.

Xilinx. 2010. « *Virtex-5 FPGA Product Table ».* En ligne. < http://www.xilinx.com/publications/prod_mktg/V5_LX__TXT_psm_table.pdf >.

Xilinx. 2010. « *Virtex-6 FPGA Product Table »*. En ligne. < http://www.xilinx.com/publications/prod_mktg/Virtex6_Product_Table.pdf >.

Xilinx. 2010. « *Virtex-7 FPGA Product Table »*. En ligne. < http://www.xilinx.com/publications/prod_mktg/Virtex7-Product-Table.pdf >.

Xilinx. 2010. « *Xilinx ML310 Documentation and Tutorials* ». En ligne. < http://www.xilinx.com/products/boards/ml310/current/ >.

Xilinx. 2010. « *ML403 Evaluation Platform Documentation* ». En ligne. < http://www.xilinx.com/products/boards/ml403/docs.htm >.

Xilinx. 2010. « *ISE Design Suite Product Table* ». En ligne. < http://www.xilinx.com/publications/matrix/Software_matrix.pdf >.

* Toutes les documents en ligne relatifs aux références bibliographiques ont été consultés le 19 mai 2010 pour des fins de vérification.

BIBLIOGRAPHIE

École de technologie supérieure. 3 mars 2010. *« Guide de rédaction d'un rapport de projet, d'un mémoire ou d'une thèse »*, v2.2. 84 p. En ligne.
< http://www.etsmtl.ca/zone2/administration/decanats/formation/etudsup/Deroulement/guides/Guide_redaction.pdf >.

Impulse Accelerated Technologies. 23 juillet 2008. « *Accelerating a Complex FIR Filter on an Avnet Virtex-5 FXT Board* ». En ligne.
< http://www.impulseaccelerated.com/ReadyToRun/AvnetV5FX/ComplexFIR_PPC/AvnetV5FX_ComplexFIR_PPC.pdf >.

Ledin, Jim A. Février 1999. « *Hardware-in-the-Loop Simulation* ». *Embedded Systems Programming*. En ligne.
< http://www.idsc.ethz.ch/Courses/embedded_control_systems/Exercises/Hardware-in-the-Loop.pdf >.

Schulenburg, Jörg. Juillet 2009. « *SPINPACK – parallel performance and possible acceleration using FPGAs* ». European Centre for Theoretical Studies in Nuclear Physics. Trento (Italie). En ligne.
< http://www.ect.it/Meetings/ConfsWksAndCollMeetings/ConfWksDocument/2009/Talks/6_10_July/Schulenburg.pdf >.

Sgandurra, Robert. 1 Mars 2010. *« Understanding Virtex FPGAs ». Embedded Technology Magazine.* En ligne.
< http://www.embeddedtechmag.com/component/content/article/7544 >.

Shenoy, Kunal. 16 décembre 2005. « *Accelerating Software Applications Using the APU Controller and C-to-HDL Tools* », XAPP901 v1.0. Xilinx. En ligne.
< http://www.xilinx.com/support/documentation/application_notes/xapp901.pdf >.

Subramanian, Nikhil. 2009. « *A C-to-FPGA Solution for Accelerating Tomographic Reconstruction* ». University of Washington. En ligne.
< http://ee.washington.edu/faculty/hauck/Tomography/Subramanian_Thesis.pdf >.

Thibeault, Claude. 2006. Note de cours, SYS-834 : Technologies VLSI et ses applications.

Thibeault, Claude. 2006. Notes de cours, ELE-740 : Logique programmable VLSI.

TOP500.org. 13 novembre 2009. « *TOP500 Report for November 2009 »*. En ligne. < http://www.top500.org/static/lists/2009/11/top500_statistics.pdf >.

Woods, Nathan et Bryce Mackin. Novembre 2006. « *FPGA Acceleration in HPC: A Case Study in Financial Analytics* », v1.0. XtremeData. En ligne. < http://www.xtremedatainc.com/pdf/FPGA_Acceleration_in_HPC.pdf >.

Xilinx. 6 février 2009. « *Virtex-5 Family Overview »*, DS100 v5.0. En ligne. < http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf >.

Xilinx. 28 janvier 2010. « *Virtex-6 Family Overview »*, DS150 v2.2. En ligne. < http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf >.

* Toutes les documents en ligne relatifs à la bibliographie ont été consultés le 19 mai 2010 pour des fins de vérification.