Les Systèmes Embarqués Reconfigurables enseignés à des automaticiens

Benoît Vigne, Pierre-Jean Lapray

Les Systèmes Embarqués Reconfigurables enseignés à des automaticiens

Benoît Vigne[2] and Pierre-Jean Lapray*[1,2]

[1]Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
[2]École Nationale Supérieure d'Ingénieurs Sud Alsace (ENSISA), Mulhouse, France

RÉSUMÉ

Dans le cadre de la formation d'ingénieurs automaticiens à l'ENSISA de Mulhouse, nous avons conçu un groupement d'unités d'enseignement portant sur la conception de systèmes numériques sur FPGA (Field Programmable Gate Array). L'offre comprend les bases de l'électronique numérique, la synthèse de systèmes combinatoires et séquentiels, et la description matérielle en langage VHDL. Il y a lieu de rapporter une expérience pédagogique, en ce qu'elle concerne des étudiants non électroniciens de formation. Nous détaillerons l'organisation des modules, les problématiques de pédagogie, et les possibles améliorations à apporter.

1 INTRODUCTION

Un système embarqué est une architecture matérielle et logicielle. Il est dans la plupart des cas spécialisé, c'est à dire qu'il est dédié à une tâche spécifique se déroulant en permanence. Son objectif peut être aussi rudimentaire qu'un contrôleur pour un système d'arrosage de jardin, ou aussi complexe qu'un contrôle dans un avion à réaction. Les principes sous-jacents sont pourtant les mêmes. La conception logicielle est étroitement liée à la conception matérielle, c.à.d. qu'elle découle du concept d'adéquation algorithme-architecture. Par rapport à un système non embarqué, le systèmes embarqué dispose de contraintes applicatives qui impactes directement la conception. En d'autres termes, le système embarqué peut se définir comme le support de l'application à forte(s) contrainte(s). Ces contraintes sont généralement définies au moment de l'élaboration du cahier des charges du système embarqué. Elles peuvent être catégorisées comme suit :

- Contraintes **physiques** : encombrement, proximité des périphériques.
- Contraintes **énergétique** : faible puissance électrique, autonomie, coût/FLOP.
- Contraintes d'**adaptabilité** : flexibilité de maintenance/reconfigurabilité, capacité de déploiement en masse.
- Contraintes **temporelles** : débit de données, déterminisme, latence, gigue, niveau de parallélisme.
- Contraintes de **sécurité**.

Le FPGA (Field Programmable Gate Array) est l'une des architectures matérielles pouvant être utilisée pour répondre à une ou plusieurs des contraintes énoncées ci-dessus. Les domaines d'application actuels sont multiples, et vont de la communication radio-fréquence (4G/5G) au contrôle-commande, en passant par l'accélération matérielle pour le "machine learning/big data". On pourra citer des cas des exemples d'applications tels que l'imagerie temps-réel [1, 2, 3], le contrôle-commande de moteur (haute dynamique) [4], le déploiement de satellites astronomiques spatiaux [5], ou le "cloud computing" (parallélisme accru) [6]. L'intégration de FPGA dans certains processeurs multi-cœurs Xeon, prouve son intérêt sur le marché actuel. De plus, l'essor du principe d'éco-conception impose de reconsidérer les architectures de calcul pour une utilisation plus efficace de l'énergie électrique [7].

La multiplication des applications à fortes contraintes se traduit concrètement par l'évolution du marché du travail, c.à.d. à une demande accrue en compétences transversales pour les ingénieurs. Les automaticiens doivent, par exemple, être en mesure d'anticiper, lors de la modélisation, l'implémentation de leurs algorithmes de contrôle dans un système électronique spécifique. Pour cela, nous avons conçu un groupement d'unités d'enseignement au sein de la filière intitulée "Automatique et Systèmes Embarqués" (ASE) de l'ENSISA à Mulhouse, qui portent sur la conception de systèmes numériques sur FPGA.

Nous présentons ici l'organisation des modules

*Corresponding author: pierre-jean.lapray@uha.fr

(Section 2) et les challenges pédagogiques liés à l'enseignement du FPGA à un public peu sensibilisé à l'électronique (Section 3). En conclusion, nous proposerons des pistes exploratoires pour l'évolution du groupe d'enseignement (Section 4).

2 ORGANISATION

2.1 Volume horaire

Le volume horaire pour le groupement d'enseignements est résumé en Table 1.

	Électr. Num.	**Sys. Emb. Reconfig.**	**Sys. Emb. Reconfig. : Blocs IP**	**Projets**
Année	1ère	2ème	2ème	2ème & 3ème
Semestre	*S5*	*S7*	*S8*	*S8* & *S9*
CM	10*h*	12*h*	10*h*	
TD	4*h*	4*h*	0*h*	
TP	4*h*	16*h* (4 × 4h)	20*h* (5 × 4h)	≈ 140*h*

TABLE 1 – Volume horaire du groupement d'enseignement réalisé à l'ENSISA.

2.2 Profil étudiant

Le public concerné par les enseignements dispensés dans la filière ASE dispose d'une forte sensibilisation à la programmation pour cibles de type processeurs. La plupart des architectures de calculs jusque-là utilisées par les étudiants sont des processeurs multi-coeurs (PC fixes/portables), des cartes de développement de types micro-contrôleurs (Arduino) ou "System on Chip" (Raspberry Pi). Les langages assimilés sont donc à haut niveau d'abstraction, tels que Python, Matlab/Simulink, Java, C++, etc.

2.3 Cadre de travail

2.4 1ère année (S5)

Les bases de l'électronique numérique sont introduites durant le programme de 1ère année. Les concepts de représentation de l'information numérique, ainsi que l'analyse et la synthèse logique (combinatoire et séquentielle) sont abordés. Les méthodes de synthèse manuelle par l'utilisation de tableau de Karnaugh, de table de vérité ou de graphe d'états sont traitées sous forme de petites applications concrètes en TD, tels que le système d'alarme d'une maison, ou le contrôle de niveau de liquide dans une cuve. Les étudiants sont rapidement confronté à la nécessité d'avoir un outil de synthèse automatique logiciel...

Aux vues du peu d'heures de TP affectées, nous faisons la conception de fonctions logiques uniquement en simulation, via le logiciel PROTEUS. Le but de du TP de 4*h* est l'étude et la réalisation complète d'un dé électronique à l'aide de composants combinatoires et séquentiels. La décomposition fonctionnelle du dé (voir Figure 1) permet d'appréhender, étape par étape, les éléments vus en cours : les compteurs, le trigger de Schmitt, les bascules D et JK, les verrous et la machine à états finis (FSM).

2.5 2ème année (S7 et S8)

Le langage VHDL est vu en deuxième année et permet aux étudiants d'obtenir les bases syntaxiques pour la description de systèmes numériques à l'aide d'outils EDA (*Electronic Design Automation*). Le cours s'appuie sur l'exemple de l'additionneur complet à 2 entrées ("fil rouge"), afin d'introduire les différents concepts du flux de conception tout au long du cours (voir Figure 2).

Les concepts abordés sont :

1. Introduction à la structure d'un FPGA,
2. Pipeline de conception complet : description RTL, simulation, synthèse, définition de contraintes physiques et temporelles, implémentation, simulation après routage et génération de bitstream.
3. Structure d'un fichier VHDL : librairies/paquetages, entité, architecture.
4. Styles de description : structurelle et comportementale.
5. Les types : *std_logic_vector, integer, signed, unsigned*, conversion de types, etc.
6. La syntaxe : multiplexeur, boucles et conditions, boucle *generate*, processus, déclaration et instantiation de composant, etc.
7. Simulation et bancs de test.
8. Outils EDA : Vivado

Les étudiants sont aussi sensibilisés aux bonnes pratiques du *designer* FPGA : la problématique de métastabilité au travers d'accidents historiques [5], la gestion de certains périphériques (boutons poussoirs, afficheurs), et le respect de contraintes de *timings* via la comparaison entre simulation comportementales et simulation après routage.

Une approche par tutoriel est utilisée en TD afin d'appréhender rapidement l'environnement de conception Vivado (qui n'est pas toujours très intuitif et dépourvu de bogues), par la mise en pratique d'un flux de conception : *design* structurel/comportemental, simulation, synthèse et implémentation sur carte de compteurs et chenillards.

Les TPs ont longtemps été mis en oeuvre sur des cartes BASYS 2 développées par Digilent.

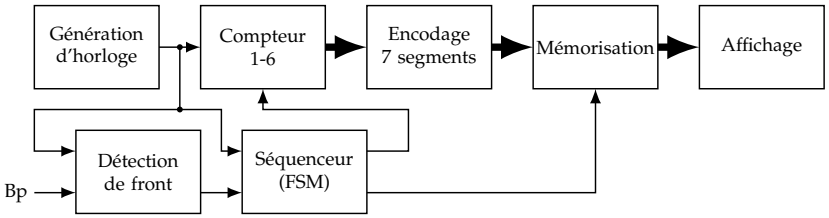


FIGURE 1 – Schéma fonctionnel du dé électronique vu en TP en 1ère année S5. Il met en pratique les concepts de l'électronique numérique et introduit la nécessité d'un synthétiseur automatique (outils EDA).

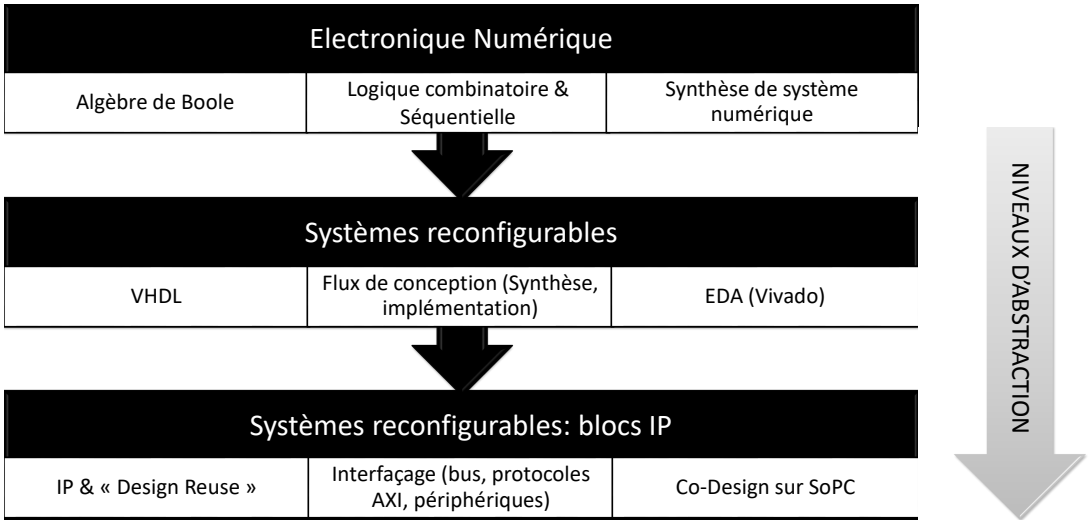


FIGURE 2 – Structure du groupement d'enseignements et niveaux d'abstraction. La couche supplémentaire qui correspond aux projets de 2ème et 3ème années n'est pas représentée ici.

Nous avons fait le choix d'enseigner sur de nouvelles cartes de développement (voir la ZYBO Figure 3(a)), ce qui permet l'utilisation de composant de type SoPC (System on Programmable Chip, Zynq-7000 de Xilinx). Cette évolution s'inscrit dans notre choix d'apporter des compétences en *co-design* durant le semestre 8. Un rapide aperçu de la structure du composant (voir Figure 3(b)) révèle la présence :

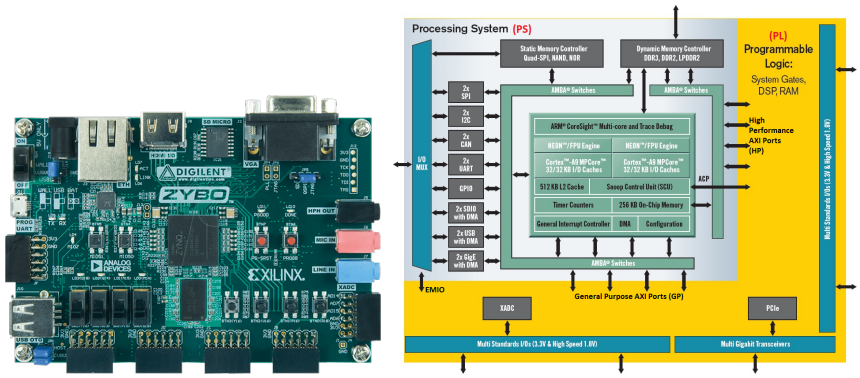
- d'une partie Processing System (PS) comportant 2 coeurs ARM Cortex A9
- d'une partie Programmable Logic (PL) configurable.

Les travaux pratiques du semestre 7 portent sur le développement de 2 applications : la modulation de largeur d'impulsion pour moteur (PWM), et le *chipset* graphique à faible consommation ($\approx 100mW$).

La première application consiste à concevoir un générateur de deux signaux PWM complémentaires à la fréquence de 16 kHz dont le rapport cyclique est réglable avec 2 boutons poussoirs. L'énoncé précise les fonctions mises en oeuvre pour concevoir de tels signaux (voir Figure 4). Outre la synthèse et l'implémentation sur carte, les étudiants doivent procéder à la validation fonctionnelle par simulation en utilisant un *testbench*.

La deuxième application consiste à développer un contrôleur VGA qui doit générer les signaux de synchronisation vidéo verticale et horizontale ainsi que les composantes RGB. La carte intègre 3 convertisseurs numérique/analogique élémentaires basés sur des réseaux R-2R et qui délivrent les signaux analogiques RGB. Le résultat final de l'application est la génération d'un flux vidéo VGA à faible consommation, redirigé vers un écran. Les étudiants peuvent y ajouter une sélection de couleurs par interrupteurs et une animation spatiale/temporelle.

Les travaux pratiques du semestre 8 permettent aux étudiants de construire/configurer une architecture mixte PS/PL et de comprendre les mécanismes de transferts des données entre les entités. Cette construction se veut progressive et sur 3 séances de 4 heures : Lors de la première séance, les étudiants commencent par la configuration de ports d'entrée/sorties parallèles pour piloter un chenillard (sens et vitesse de défilement). Lors de



(a) (b)

FIGURE 3 – (a) Carte de développement Zybo de Digilent, possédant un SoPC Xilinx (Zynq-7000). (b) Structure interne d'un Zynq-7000, composé d'une partie PS ("*Processing System*" avec processeur ARM) et d'une partie PL ("*Programmable Logic*" avec cellules logiques).

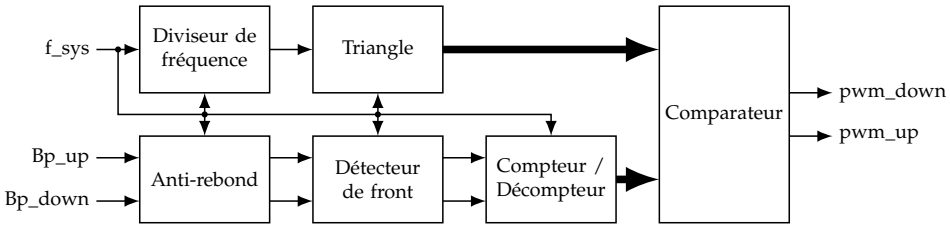


FIGURE 4 – Structure interne de la fonction PWM construite en TP de 2ème année.

la deuxième séance, la temporisation logicielle est remplacée par la configuration d'un timer. Enfin, la dernière séance permet d'appréhender la gestion des interruptions. Ces applications s'inspirent de celles décrites dans "The Zynq Book Tutorials" ([8]), et fait appel à des blocks IP.

Les deux dernières séances sont consacrées à un mini-projet sur un lecteur de fichiers audio (voir Figure 5). Les morceaux musicaux sont mémorisés sur une carte SD au format WAVE. Les éléments mis en oeuvre sont :

- Un driver de bus I2C pour la configuration du codec audio,
- un driver de bus I2S pour l'interfaçage du flux audio avec le codec,
- un driver de bus pour carte SD,
- un driver de bus UART pour l'interface IHM,
- une mémoire FIFO pour la gestion du flux audio.

3 RETOUR D'EXPÉRIENCE

"Le fait de coder directement sur le hardware plutôt qu'en soft offre une nouvelle vision de la programmation des systèmes embarqués. En effet, cela nécessite un niveau d'abstraction différent et il est conseillé d'avoir un minimum de connaissances en électronique embarqué. C'est une programmation qui peut être difficile à comprendre au début, mais une fois cette étape franchie, cela devient une méthode de programmation très intéressante. On acquiert une meilleure compréhension et un meilleur contrôle des fonctions réalisées, ainsi qu'un aperçu de toutes les fonctions réalisables. Le seul défaut restant est la durée de compilation qui peut être long, en fonction de la complexité du programme." Anthony CHASSIGNET, étudiant 3ème année ASE.

3.1 Appréhension du langage

Contrairement aux processeurs, les FPGA sont conçus pour des tâches parallélisables, de telle sorte que chaque traitement est indépendant et dispose de ses propres ressources matérielles. L'un des avantages des FPGA par rapport aux systèmes à processeurs est que la logique d'application est implémentée dans des circuits matériels dédiés plutôt qu'une exécution sur une architecture polyvalente de type processeur. Chaque tâche est donc affectée à une section de la puce et peut fonctionner de manière autonome sans influence des autres fonctions logiques. Par conséquent, les performances sont potentiellement inchangées en ajoutant d'autres fonctionnalités au système embarqué ultérieure-

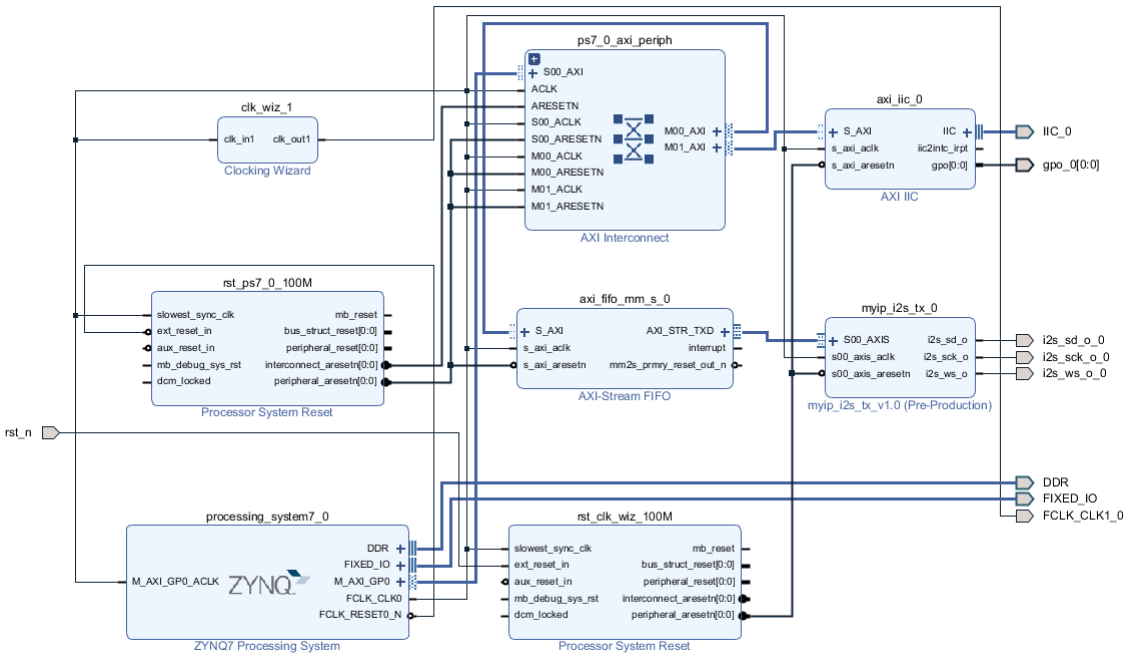


FIGURE 5 – Architecture du projet de lecteur audio.

ment. Toutes ces nouveautés posent quelques difficultés d'appréhension de la part des étudiants, puisque il est coutume d'utiliser des opérations séquentielle pour la conception fonctionnelle, c.à.d. par la programmation logicielle (C++, Java, Python, etc.). L'assimilation des pratique de *design* VHDL (entité/architecture, composants, signaux, etc.) peut donc être déroutante de prime abord. Il faut donc désapprendre et réacquérir certains mécanismes de conception liés spécifiquement à la description matérielle. Un problème récurrent est le manque de soin apporté à la première phase qui est la modélisation (définition des entrées sorties, mappage des composants et simulation). La description "à vue" par tâtonnement est très contre-productive en description matérielle et donne rarement un système robuste et fonctionnel.

3.2 Niveau d'expertise atteint

On dit généralement qu'il faut dix fois plus de temps pour concevoir un *design hard* par rapport à un *design soft* pour une même application. Par conséquent, la complexité des applications réalisées en TP est particulièrement faible par rapport aux applications utilisant les processeurs. La difficultés pédagogique ici est de sensibiliser les étudiants aux contraintes d'application (utilisation d'énergie, déterminisme, sécurité, etc.). Ce sont des paramètres fondamentaux, trop souvent entrevu qu'en entreprise, et qui feront préférer les FPGA aux processeurs dans certains cas. Il en résulte une panoplie d'erreurs récurrentes faites par les étudiants telles que l'utilisation massive de processus asynchrones, l'utilisation de signaux asynchrones comme horloge système, la lecture d'une sortie, l'affectation concurrentielle dans plusieurs processus, la déclaration multiple de signaux/ports, les erreurs de syntaxe pour les affectations bit/bus, etc. A cela viennent s'ajouter les difficultés d'ordre purement techniques liées à l'abstraction logicielle, l'aspect contre-intuitif des logiciels EDA, les bogues et la multiplication des interfaces utilisateur (*IP integrator*, *IP catalog*, SDK, l'environnement de *debug*, etc.).

3.3 Projets étudiants

A la suite des unités d'enseignement précédemment détaillées, 3 binômes d'étudiants ont réalisé leur projet sur FPGA en 2ème et 3ème année (sur la période 2018/2019).

3.3.1 Projet 2ème année : Détection de végétation

Le but du projet visait à développer un algorithme de détection de végétation et à le tester sur FPGA. Une caméra à 2 capteurs (bandes spectrales couleurs+proche infrarouge) a été utilisée comme source de données. La couverture végétale dans une scène se détermine par calcul de l'indice de végétation (principe NDVI [9]). Le *de-*

sign fût d'abord simulé dans un environnement logiciel (Matlab/Simulink), puis implémenté sur FPGA. Les outils Matlab/Simulink ont servi à comparer et évaluer les 2 implémentations (logicielle et matérielle). Les étudiants ont en outre conçu un pipeline de simulation matérielle de type "hardware in the loop" utilisant les outils de développement Matlab intitulés "HDL Coder" et "HDL Verifier". Le déploiement final s'est réalisé sur un FPGA de chez Intel (Cyclone V). L'évaluation des performances s'est faite en termes de débit de calcul, de débit d'interfaçage, et de qualité d'image. Les étudiants ont clairement montré une véritable autonomie pendant toute la durée du projet, ce qui tend à valider que le groupement d'enseignement proposé en amont a été suffisant et pertinent pour leur permettre d'avancer. Nous avons aussi eu un retour de satisfaction personnelle des étudiants, qui ont témoigné sur l'intérêt et l'utilité du FPGA dans le cas d'un traitement vidéo temps-réel.

Les travaux ont été valorisés pendant la journée des projets et de l'innovation en 2018. Le poster présenté par les 2 étudiants est disponible sur le web [10].

3.3.2 Projet 3ème année : La pédale multi-effets pour guitare électrique

Le but du projet est de concevoir un multi-effets audio en utilisant un FPGA. Les étudiants ont repris un projet existant de captation et restitution de son par codec audio, et ont rajouté des blocs de traitement du signal audio, dans le domaine temporel et dans le domaine fréquentiel. Les effets de fuzz (écrêtage) et de pitch ont pu être implémentés sur une Zedboard. Les étudiants ont notamment été confrontés au différents standards de bus (AXI, AXI-Stream), mais aussi au *co-design* (configuration du codec audio) et à l'élaboration de FIR (Finite Imulse Response) à l'aide de blocs IP. Le projet fût un succès, et sera repris par d'autres étudiants pour de futurs projets.

3.3.3 Projet 3ème année : Intégration d'un Soc sur une plateforme robotique mobile

Les robots mobiles utilisent de plus en plus des capteurs extéroceptifs évolués (caméra, lidar) afin de percevoir au mieux leur environnement et accroître ainsi leur autonomie [11]. Ces capteurs nécessitent des traitements gourmands à la fois en temps de calcul mais aussi en consommation. L'apport d'un SoC (System on Chip) intégrant processeurs+fpga peut permettre de réduire ces contraintes temporelles et énergétiques [12], afin de rendre le système plus efficace.

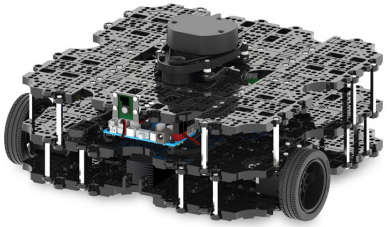


FIGURE 6 – Turtlebot3 (modèle Waffle).

L'école dispose de petits robots mobiles Turtlebot3 (voir Figure 6) de type unicycle développés par la société ROBOTIS [13]. Ces robots s'appuient sur une architecture matérielle composée de :

- 1 carte Raspberry Pi3 modèle B assurant l'échange de données avec un PC,
- 1 carte OpenCR basée sur un micro-contrôleur STM32 assurant l'interfaçage avec les capteurs proprioceptifs et les moteurs,
- 1 lidar 2D,
- une caméra (optionnelle).

L'ensemble de la commande est assurée par un noyau logiciel ROS (Robotic Operating System). Sur le système originel, les données du lidar sont envoyées par une liaison Wifi vers le PC maître afin d'être traitées pour de la localisation et cartographie (SLAM).

Le projet des étudiants consistait à remplacer la carte Rasberry par une carte Zybo afin de traiter dans un premier temps in situ les données du lidar puis dans un deuxième temps la traitement d'images issues de la caméra. L'installation d'une distribution linux adaptée [14] à la carte a pris beaucoup de temps mais a permis de valider le fonctionnement de ROS sur la carte Zybo. L'interfaçage de la partie *Programmable Logic* (PL) avec le noyau linux n'a été abordée que partiellement et nécessite d'être approfondi.

4 AMÉLIORATIONS POSSIBLES ET CONCLUSION

Le temps de développement est par définition beaucoup plus long en environnement matériel bas niveau, ce qui conduit l'étudiant à se sentir frustré de n'avoir réalisé qu'un allumage de LEDs à la fin d'un TP de 4h. Tout le challenge pédagogique repose ici sur le fait de faire prendre conscience de la complexité qu'a un algorithme à haut niveau, montrer les faiblesses du processeur, et proposer une optimisation via le FPGA. L'argument selon lequel un simple affichage VGA aurait pu être fait de manière haut niveau en quelques minutes à l'aide d'un processeur est souvent avancé par

les étudiants. Cet argument pourrait être contrebalancé en imposant des contraintes fortes pour l'application demandée en début de TP. L'étudiant s'imposerait de par lui-même alors le choix du FPGA comme architecture de calcul. Cet objectif garantirait la recherche de performance énergétique/calculatoire comme but premier, plutôt que la recherche d'une fonctionnalité en particulier. Plusieurs avantages pourraient être démontrés par l'expérimentation ; par exemple, l'utilisation abusive de processeurs pour la réalisation de tâches simples peut conduire à une surconsommation.

Pour résumer, la question récurrente et centrale du point de vue pédagogique pour nous serait : Comment susciter l'intérêt des FPGA, et ce au travers d'applications simples (peu coûteuse en développement), et dont le cahier des charges justifierait d'ambler l'utilisation d'un FPGA ? Des pistes pourraient être envisagées en considérant certaines contraintes applicatives telles que :

- Système à latence fixe (fortement déterministe) qui échoue en exécution sur processeur.
- Système hautement parallélisé pour une accélération matérielle inatteignable sur processeur.
- Système très basse consommation, en quantifiant la puissance de calcul en rapport avec la transformation d'énergie.
- Système à sécurité renforcée pouvant être *hacké* sur processeur mais pas sur FPGA.

Nous réfléchissons actuellement à un point de référence applicatif qui pourrait être abordé communément sur une architecture processeur et un FPGA : le développement d'une même application sur les 2 architectures est une piste d'amélioration pour mettre en éveil l'intérêt d'utilisation d'un FPGA plutôt qu'un processeur.

REMERCIEMENTS

Les auteurs souhaitent remercier le CNFM (Coordination Nationale de la Formation en Microélectronique et en nanotechnologies) pour les tarifs préférentiels appliqués sur les cartes Zybo.

RÉFÉRENCES

[1] P.-J. LAPRAY et al. « An FPGA-based pipeline for micropolarizer array imaging ». In : *International Journal of Circuit Theory and Applications* 46.9 (2018), p. 1675-1689.

[2] P.-J. LAPRAY, B. HEYRMAN et D. GINHAC. « HDR-ARtiSt : an adaptive real-time smart camera for high dynamic range imaging ». In : *Journal of Real-Time Image Processing* 12.4 (2016), p. 747-762.

[3] P.-J. LAPRAY et al. « Multispectral filter arrays : Recent advances and practical implementation ». In : *Sensors* 14.11 (2014), p. 21626-21659.

[4] J. EMERY et al. « Design and validation methodology of the control system for a particle beam size measurement instrument at the CERN laboratory ». In : *2017 American Control Conference (ACC)*. Avr. 2017, p. 4221-4228.

[5] D. R. BRANSCOME et al. « WIRE Mishap investigation board report ». In : *NASA, June* 8 (1999), p. 255.

[6] S. A. FAHMY, K. VIPIN et S. SHREEJITH. « Virtualized FPGA Accelerators for Efficient Cloud Computing ». In : *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. Nov. 2015, p. 430-435.

[7] S. TIAN et J. SZEFER. « Temporal thermal covert channels in cloud FPGAs ». In : *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2019, p. 298-303.

[8] M. A. E. LOUISE H CROCKETT Ross A.Elliot et D. NORTHCOTE. *The Zynq Book Tutorials for Zybo and Zebboard*. Strathclyde Academic Media, 2015.

[9] J. ROUSE et al. « Monitoring vegetation systems in the Great Plains with ERTS ». In : *NASA special publication* 351 (1974), p. 309.

[10] E. KERRI et B. KRIT. *Développement d'un algorithme de traitement d'image sur FPGA*. URL : https://e-formation.uha.fr/pluginfile.php/133280/mod_resource/content/1/Posters-2018/Poster_07.pdf.

[11] I. N. R. SIEGWART et D. SCARAMUZZA. *Introduction to autonomous mobile robots*. MIT press, 2011.

[12] T. OHKAWA et al. « FPGA Components for Integrating FPGAs into Robot Systems ». In : *IEICE Transactions on Information and Systems* E101.D.2 (2018), p. 363-375.

[13] URL : http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

[14] URL : http://xillybus.com.