Émulation de Systèmes Cyber-Physiques sur FPGA

Maelic Louart, Jean-Christophe Le Lann, Frédéric Le Roy, Abdel Boudraa, Jean-Jacques Szkolnik

Émulation de Systèmes Cyber-Physiques sur FPGA

Maelic LOUART[12], Jean-Christophe LE LANN[1], Frédéric LE ROY[1], Abdel BOUDRAA[2] Jean-Jacques SZKOLNIK[2]

[1]Lab-STICC, UMR CNRS 6285, ENSTA Bretagne, 2 Rue François Verny, 29200 Brest, France

[2]IRENav,EA3634, BCRM Brest CC 600, 29240 BREST Cedex 9, France

(maelic.louart,jj.szkolnik,abdel.boudraa)@ecole-navale.fr,
(jean-christophe.le_lann,frederic.le_roy)@ensta-bretagne.fr

Résumé – Les FPGAs permettent aujourd'hui la conception de systèmes programmables complets sur puce (SoC) capables de couvrir une gamme extrêmement large d'applications, allant des systèmes embarqués classiques à la virtualisation dans le cloud. Toutefois, le potentiel des FPGAs ne s'arrête pas là. Leur flexibilité et leur parallélisme naturel invitent à explorer leur utilisation dans des domaines plus confidentiels. Dans cet article, nous nous intéressons à l'émulation de systèmes cyber-physiques complets sur FPGA. Notre cas d'étude présente l'émulation d'un système constitué d'une multitude de navires communicant selon le protocole d'identification AIS. L'étude démontre un facteur d'accélération de 75 par rapport à l'exécution purement logicielle, et permet d'esquisser un flot de conception adapté au domaine de l'émulation de tels systèmes sur FPGAs.

Abstract – FPGAs today enable the design of complete programmable systems-on-a-chip (SoC) capable of covering an extremely wide range of applications, from traditional embedded systems to cloud virtualization. However, the potential of FPGAs does not end there. Their flexibility and natural parallelism invite to explore their use in more confidential domains. In this paper, we focus on the emulation of complete cyber-physical systems on FPGAs. Our case study presents the emulation of a system consisting of a multitude of ships communicating according to the AIS identification protocol. The study demonstrates a speed-up factor of 75 compared to pure software execution, and allows us to outline a design flow adapted to the emulation of such systems on FPGAs.

1 Introduction

Les FPGAs possèdent plusieurs vertus remarquables [1] qui leur confèrent un statut particulier dans le domaine de l'Electronique. Ils permettent désormais de concevoir des systèmes embarqués complexes complets et sont en particulier attrayants pour de faibles volumes de production. On peut les trouver dans des systèmes embarqués variés, allant du multimedia aux robots interplanétaires, du médical au militaire, etc. Ici, les capacités de reconfiguration dynamique, de versatilité, de synchronisation et d'accélération des calculs ainsi que la faible consommation d'énergie font des FPGAs des plateformes haute performance désormais incontournables [2].

Toutefois, dès les années 2010, des chercheurs ont proposé d'utiliser les FPGAs hors des sentiers battus. En particulier Givargis et Vahid ont proposé d'utiliser des FPGAs comme émulateurs de tout autres systèmes : il s'agissait d'émuler le fonctionnement d'un poumon artificiel [3]. Le rôle du FPGA était alors de permettre le test et la mise au point de ventilateurs externes préexistants : le FPGA jouait le rôle d'un organe artificiel, modélisé par un système d'équations différentielles soigneusement établi avec des spécialistes de la respiration. Il devenait facile de placer ce poumon artificiel dans des situations complexes ou difficiles à gérer pour le ventilateur. De tels ventilateurs rentrent dans la catégorie des systèmes cyber-physiques : ils sont eux-mêmes constitués de microcontrôleurs, de capteurs et d'actionneurs, et sont bien évidemment en interaction forte avec leur environnement (ici le poumon). Ces travaux remarquables n'ont pas selon nous reçu une attention suffisante. Nous proposons une méthode alternative, mais en ligne avec les ambitions de ces auteurs, dans un tout autre cas d'application : il s'agit pour nous de mettre au point un récepteur de communication maritime (AIS). De manière similaire, ce récepteur est un assemblage complexe de blocs de traitement du signal, et est également sensible à son environnement physique. Cet environnement physique est simulé et correspond dans notre cas à un canal de communication hertzien, soumis aux aléas et lois physiques caractéristiques d'un milieu maritime. Notre manière de procéder est toutefois très différente de celles de Givargis et Vahid et Miller : en premier lieu, nous avons la chance de disposer de FPGA contenant une quantité de ressources largement supérieure à celle dont bénéficiait les auteurs à la date de publication. En second lieu, nous allons démontrer l'utilisation possible de la synthèse comportementale (HLS) afin de considérer des modélisations à très grande échelle (celle de plusieurs navires émettant des trames AIS simultanément, voir section 3). Enfin et surtout, notre travail nous permet à la fois de simuler cet environnement physique mais aussi d'émuler différents systèmes embarqués communicants sur le même FPGA. Il s'agit selon nous des premiers travaux qui poussent aussi loin l'utilisation des FPGA à des fins d'émulation et de simulation.

2 Méthode d'émulation de systèmes cyber-physiques sur FPGA

Comme exprimé dans l'introduction, nous proposons une nouvelle approche de simulation de CPSs qui, à partir d'un modèle exprimé en C++, utilise la HLS pour générer le simulateur complet exécutable sur FPGA. Cette approche divise le CPS en composants : un d'entre eux simule l'environnement et les autres émulent des systèmes embarqués, chacun étant un transpondeur AIS. La description des comportements des composants se fait en C++ et l'interaction entre composants est programmée par l'utilisation de la directive *dataflow* fournie par la HLS. Cette directive implémente automatiquement des canaux sous forme de FIFO ou PIPO entre composants. L'intérêt d'utiliser du code C++ est la disponibilité quasi-systématique de codes C/C++ hautement spécifiques à un domaine, capables de simuler des parties du comportement de presque tous les CPS (phénomènes physiques, mécaniques, chimiques, etc.).

L'utilisation de la HLS et de la directive *dataflow* imposent quelques contraintes à la programmation. Tout d'abord, les allocations dynamiques de mémoire, les fonctions récursives et les appels système (printf(), time()) ne sont pas supportés par la HLS, et ne peuvent donc pas être synthétisés. Ensuite, pour que l'optimisation *dataflow* fonctionne, les données doivent circuler dans la modélisation d'un composant à l'autre. Ainsi, les violations de type producteur-consommateur unique, la rétroaction entre composants et l'exécution conditionnelle de composants sont empêchées. En outre, cette méthode ne peut être appliquée qu'à une architecture limitée à un graphe orienté acyclique.

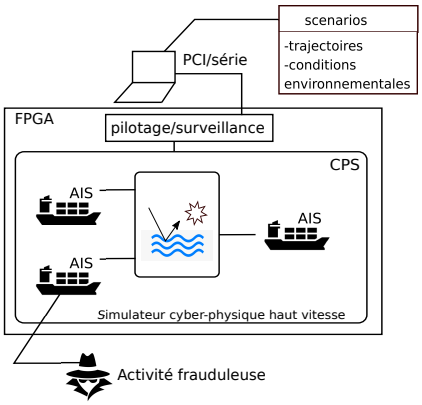

FIGURE 1 – Notre proposition : Les FPGA comme plateformes de simulation de systèmes cyber-physiques.

3 Cas d'application de notre méthode de simulation

3.1 Le système d'identification automatique

Notre étude de cas consiste en un ensemble de navires qui se déplacent et échangent des messages radio en s'appuyant sur un protocole maritime dédié : le système d'identification automatique (AIS). L'AIS fonctionne sur deux fréquences, 161,975 MHz et 162,025 MHz, appartenant à la bande des très hautes fréquences (VHF). Depuis 2002, la convention sur la sauvegarde de la vie humaine en mer (SOLAS) de l'organisation maritime internationale (OMI) exige l'installation d'un transpondeur AIS de classe A pour les navires internationaux de plus de 300 tonnes de jauge brute naviguant dans les eaux internationales et pour les navires à passagers quelque soit leur taille. Ce système améliore la sécurité du trafic maritime en permettant l'échange automatique d'informations de navigation entre les navires (informations dynamiques (position, vitesse, route, cap) et informations statiques (identité, port de départ, etc.)). Cependant, ce système est vulnérable : les informations envoyées par les transpondeurs peuvent être facilement falsifiées ou usur-pées (pour la piraterie, le transport illégal, etc.). C'est pourquoi des algorithmes de détection d'usurpation et de falsification doivent être développés et intégrés dans les transpondeurs AIS de nouvelle génération. Ces algorithmes s'appliquent aux messages et signaux AIS.

3.2 Besoin d'émulation

Pour développer un transpondeur fiable contenant les algorithmes développés il faut exécuter un grand nombre de simulations testant le fonctionnement du système. avec un large éventail de scénarios reproduisant les conditions d'utilisation du produit. Cependant, obtenir des flux de tests complets à partir de mesures réelles sur le terrain reste difficile voir impossible. Dans notre travail, pour résoudre ce problème, un cadre de simulation a été créé pour générer une grande variété de scénarios *synthétiques* contenant des messages et des signaux AIS. Les messages générés peuvent être falsifiés et brouillés, et les signaux générés peuvent reproduire les caractéristiques matérielles des transpondeurs (erreurs sur les fréquences des porteuses). Le simulateur émule un nombre variable de transpondeurs AIS qui communiquent entre eux dans un environnement maritime simulé. La simulation fonctionne entièrement sur FPGA et est synthétisée à l'aide de la HLS. Un exemple de l'architecture du simulateur est présenté dans FIGURE 1 où trois transpondeurs communiquent.

3.3 Modèle du CPS simulé

La FIGURE 2 illustre l'architecture générale de notre simulateur CPS dans la même configuration que dans la FIGURE 1. Dans le CPS, trois types de composants sont affichés et ont été modélisés. Un *Émetteur* qui agit comme un transpondeur AIS qui, à partir d'une trame NMEA, envoie un signal AIS modulé en bande de base. Un *Récepteur* qui agit comme un transpondeur AIS, qui reçoit uniquement les signaux AIS en bande de base et les démodule pour en extraire la trame NMEA. Enfin, un *Canal* qui simule l'environnement pour appliquer certaines lois physiques aux signaux AIS échangés (bruit et affaiblissement de propagation) et reproduit certaines imperfections matérielles des *Émetteur* et *Récepteur* (différence fréquentielle

entre leurs fréquences porteuses de modulation). L'*Émetteur* et le *Récepteur* représentent les systèmes embarqués du CPS et le *Canal* représente l'environnement. Nous représentons également sur la FIGURE 2 deux composants spécifiques, un composant *Pilotage* et un composant *Surveillance*, pour, respectivement, piloter la simulation et observer les signaux échangés. Le *Pilotage* gère, à partir d'un réglage initiale effectué à l'initialisation du simulateur, l'évolution des données dynamiques de chaque navire au cours de la simulation. Avec ces données et les informations statiques il crée les trames NMEA transmises à chaque *Émetteur*. Chaque composant contient une chaîne de blocs exécutés séquentiellement. Notez que l'architecture est modélisée par un graphe orienté acyclique : le composant *Pilotage* est le début du graphe et le composant *Surveillance* la fin. Par ailleurs, certaines parties du système sont dédiées à la simulation pure, tandis que certains éléments (comme notre *Récepteur*) peuvent être considérés comme un prototype viable d'un futur dispositif embarqué. En effet, le composant *Récepteur* a parfaitement réussi à décoder des signaux AIS réels enregistrés dans la rade de Brest.

3.4 Synthèse matérielle

Nous avons expérimenté notre méthode de conception du simulateur présentée dans la partie 2. L'ensemble du modèle CPS représente environ 4000 lignes de code C++. Des variables à virgule fixes sont utilisées. Les résultats de la synthèse des trois composants sont présentés dans le tableau 1. Pour le composant *Surveillance* rien n'est présenté car les valeurs sont négligeables. La cible de la synthèse était un FPGA Xilinx Ultrascale+ HBM sur une carte VCU128 (environ 10^6 de cellules logiques du système, 3K tranches de DSP, etc). L'outil HLS était Vitis HLS. Le pourcentage d'occupation pour chaque composant est faible (moins de 2% pour les éléments BRAM, DSP, FF, et LUT). Cette espace libre laisse la possibilité de complexifier l'environnement simulé et d'augmenter le nombre de bateaux simulés en même temps pour tester des scénarios variés. Ainsi, nous montrons dans le tableau 2 de la partie 4.2 qu'au maximum, avec les composants décrits dans la partie 3.3, 18 *Émetteurs* peuvent être simulés en même temps sur le FPGA.

TABLE 1 – Ressources utilisées par chaque composant du simulateur.

	BRAM	**DSP**	**FF**	**LUT**
Simulateur	329(8%)	36(0%)	22564(0%)	40806(3%)
Pilotage	1	12	2336	5917
Émetteur	65	0	8532	15746
Canal	100	19	3039	7376
Récepteur	99	5	7855	11241

4 Simulations de CPS

4.1 Présentation des scénarios et environnements simulés

Pour illustrer la capacité de notre simulateur à simuler plusieurs navires à la fois, trois scénarios ont été simulés. Un contenant un seul navire, un autre deux et un autre 5. A chaque fois 1000 messages sont transmis au récepteur. Facilement, des falsifications similaires à celles de [4] peuvent être ajoutées aux messages transmis. De plus, pour montrer l'intérêt de notre composant *Canal*, les performances de notre *Récepteur* ont été évaluées statistiquement pour plusieurs valeurs de SNR et de décalage fréquentielle entre la porteuse de l'*Émetteur* et du *Récepteur*. Pour simuler les trajectoires de plusieurs bateaux en même temps il suffit d'ajouter des composants *Émetteur* en plus d'un composant *Canal* et d'un composant *Récepteur* sur le FPGA. En plus, il faut modifier le composant *Canal* pour lui permettre d'accepter des signaux provenant du nombre de composants *Émetteur* souhaité et le composant *Pilotage* pour générer des trames AIS à tous les *Émetteurs* émulés.

4.2 Résultats de la simulation et analyse

Nous comparons, dans le tableau 2 le temps de simulation du logiciel au temps de simulation du matériel pour les trois scénarios présentés ci-dessus. Le logiciel était un processeur Intel Core I5 standard à 1.7GHz avec 16Gb de RAM et le matériel le FPGA présenté ci-dessus. Dans ce tableau, nous affichons aussi le taux d'occupation du FPGA et nous prédisons les performances que l'on aurait obtenu si nous avions simulé 18 *Émetteurs* en même temps. Pour ce nombre d'*Émetteurs* le taux d'occupation des BRAM atteint 97%, le FPGA ne peut pas émuler plus d'*Émetteurs* en même temps. Pendant chaque simulation, le *Récepteur* reçoit 1000 messages. Les temps de simulation des trois scénarios sont très courts et restent les mêmes alors que le nombre d'*Émetteurs* augmente. Cela souligne la capacité des FPGAs à paralléliser les calculs et leur scalabilité. Dans un environnement réel un message est reçu au maximum tous les 26.6 ms alors qu'il nécessite 9.303 ms pour être traité avec notre composant *Récepteur*. Cela laisse une marge pour intégrer des algorithmes dans le *Récepteur* tout en conservant la caractéristique temps réel.

L'utilisation de variables à virgule fixe introduit des erreurs qui influencent la précision des données dynamiques. Ces erreurs peuvent être négligées. En effet, une analyse statistique sur 1000 valeurs montre que ces erreurs suivent une distribution gaussienne centrée avec un écart-type de $\sigma_{lat} = 4.0m$ pour la latitude et $\sigma_{lon} = 4.2m$ pour la longitude. L'écart-type de ce bruit est du même ordre de grandeur que la précision de la mesure GPS utilisé par l'AIS pour connaître la position des bateaux. Ainsi, les positions reçues par le récepteur ont une précision suffisante pour être utilisées pour entraîner les algorithmes de détection d'usurpation et de falsification.

Pour finir, afin de montrer la capacité du simulateur à modifier les conditions environnementales, nous calculons les performances du démodulateur GMSK contenu dans le composant *Récepteur*. Pour ce faire, la courbe de la FIGURE 3 montre l'évolution du taux d'erreur binaire (BER) en fonction du SNR et du décalage en fréquence des porteuses. Seul des erreurs de fréquence positives sont représentées car nous obtenons les mêmes résultats avec des erreurs négatives. Les performances obtenues sont similaires aux autres démodulateurs GMSK ren-

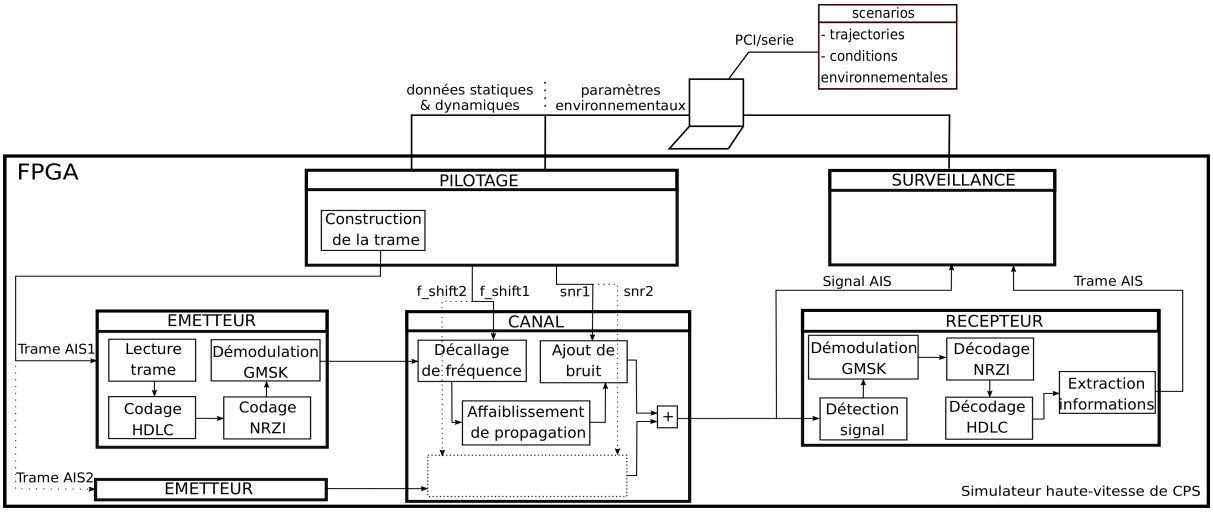

FIGURE 2 – Modèle détaillé du CPS simulé sur FPGA.

TABLE 2 – Comparaison du temps de simulation entre processeurs et FPGA

nb. Emetteurs	Soft.	Hard.	Gain	taux d'occupation
1	921s	15.6s	59	8%
2	986s	15.6s	63	13%
5	1181s	15.6s	75	28%
18 (predict.)	2026s	15.6s	127	98%

contrés dans la littérature ce qui valide la démodulation GMSK mise en oeuvre et le fonctionnement du prototype *Récepteur*.

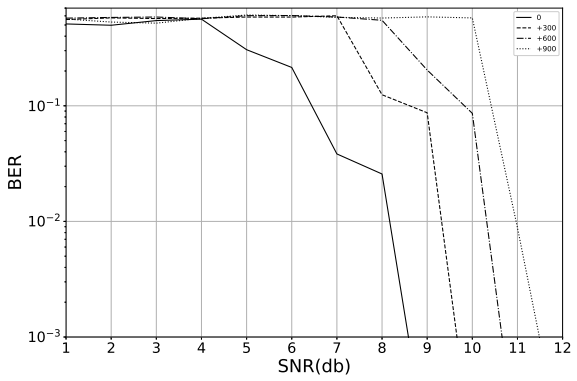

FIGURE 3 – Evolution du BER en fonction du SNR et du décalage de fréquence.

5 Discussion

Notre méthode de conception de CPS est limitée aux CPS dont l'architecture est un graphe orienté acyclique. En dehors de ce type d'architecture, Vitis HLS ne peut pas synthétiser le système. Par exemple, dans notre cas d'application, les transpondeurs AIS communiquent entre eux selon la méthode TDMA (Time-division multiple access) dont l'architecture est un graphe non orienté avec autant de noeuds que de bateaux. Cette méthode TDMA ne peut pas être synthétisée par Vitis HLS.

6 Conclusion

Dans cet article, nous avons proposé l'utilisation du FPGA comme plateforme d'émulation de systèmes cyber-physiques complexes. Notre étude de cas porte sur un ensemble de transpondeurs AIS fonctionnant dans un environnement maritime simulé. Les progrès des outils de HLS et les capacités croissantes des FPGAs rendent une telle approche possible. Par rapport à une émulation purement logiciel nous montrons des gains d'accélération remarquables ($\times 75$). Nous soulignons certaines limites des outils classiques de HLS qui nécessitent des approches alternatives pour pouvoir simuler une large gamme d'architectures. Notre article démontre que le FPGA peut être considéré comme une plateforme de choix pour l'émulation de systèmes cyber-physiques.

Références

[1] T. Grimm, B. Janssen, O. Navarro, and M. Hubner. *The value of fpgas as reconfigurable hardware enabling cyber- physical systems*. 2015 IEEE 20th Conf. on Emerging Technologies & Factory Automat. (ETFA), pp. 1-8.

[2] S. Mojlish, N. Erdogan, D. Levine, and A. Davoudi. *Review of hardware platforms for real-time simulation of electric machines.* IEEE Trans. on Transp. Electrification, vol. 3, no. 1, pp. 130-146, 2017.

[3] B. Miller, F. Vahid, and T. Givargis. *Meds : Mockup electronic data sheets for automated testing of cyber-physical systems using digital mockups.* 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1417-1420, IEEE

[4] F. Katsilieris, P. Braca, and S. Coraluppi. *Detection of malicious ais position spoofing by exploiting radar information.* in Proc. of the 16th Int. Conf. on information fusion, pp. 1196-1203, IEEE, 2013.