

# Conception de SoC sur la plate-forme OCAE du CIME Nanotech

Stéphane Mancini, Mounir Benabdenbi, Régis Leveugle, Olivier Muller, Frédéric Pétrot

October 29, 2010

## Résumé

Au sein de CIME Nanotech, la plate-forme Objets Communiquant et Applications Embarquées (OCAE) met à disposition des enseignants/chercheurs l'ensemble de la chaîne de conception et prototypage des systèmes numériques intégrés de type System on Chip (SoC) et System on Programmable Chip (SoPC). La filière SLE de l'ENSIMAG utilise cette ressource du CIME dans le cadre de projets de conception de SoC. Les applications visées concernent aussi bien le multimédia que les réseaux d'objets communicant ou encore la sécurité des systèmes embarqués.

L'ensemble des logiciels et plate-formes de prototypage permet aux étudiants de découvrir et mettre en oeuvre une chaîne de conception depuis la spécification haut niveau jusqu'à l'intégration sur système programmable pour prototypage, en passant par des étapes de synthèse haut niveau C-to-RTL, codesign logiciel/matériel et validation. Ainsi, les étudiants sont amenés à concevoir des prototypes de SoC composés de processeurs, IP d'accélération matérielle et logiciel associé-application et système d'exploitation embarqué temps réel ou Linux embarqué.

Les publics visés par la plate-forme OCAE sont toutes les formations qui s'intéressent à l'étude détaillée des interactions logiciel/matériel et aux communications entre systèmes embarqués. Ainsi sont naturellement concernées les filières Systèmes Electroniques Intégrés (SEI) de PHELMA, Systèmes et Logiciels Embarqués (SLE) de l'ENSIMAG mais aussi la filière Signal Image Communication Multimédia (SICOM) de PHELMA et le Master Crypto UJF. Il est à noter que la plate-forme OCAE accueille régulièrement des sessions de formation continue.

## 1 Introduction

L'accroissement de la complexité des System On Chip (SoC) et Multi Processor System on Chip (MPSoC) engendre de nouvelles problématiques méthodologiques, aussi bien techniques qu'organisationnelles. En effet la complexité des systèmes conçus par les ingénieurs nécessite d'une part la mise en place de flots de conceptions complexes, avec de longues boucles de conception/validation, avec des outils très divers, allant de la synthèse de haut niveau, les méthodes formelles de validation, jusqu'au placement-routage de systèmes pouvant contenir plusieurs centaines de millions, voire des milliards, de transistor. Chacune des étapes du flot de conception nécessitant un haut degré de technicité. D'autre part il apparaît que la somme de connaissances et d'expertise impliqué dans de tels projet est telle que plus personne ne peut prétendre en maîtriser l'intégralité et le partage de la connaissance devient un enjeu primordial.

Dans ce contexte, la filière Systèmes et Logiciels Embarqués (SLE) de l'ENSIMAG vise à former des ingénieurs à la conception des SoC. Au sein de ce cursus, les projets "Etude de cas d'implantation d'un SLE" ont pour vocation à faire découvrir les différentes facettes de la conception des SoC par une pratique intensive, puisque ce module recouvre 96H00, dont la moitié encadrées.

Les différents projets ont en commun les outils de développements et couvrent de nombreux domaines applicatifs, depuis le traitement d'image, l'audio, la conception de systèmes d'exploitation ou encore la sécurité des systèmes embarqués. A titre d'exemple, un projet de conception d'accélération du lancer de rayon est présenté en détail.

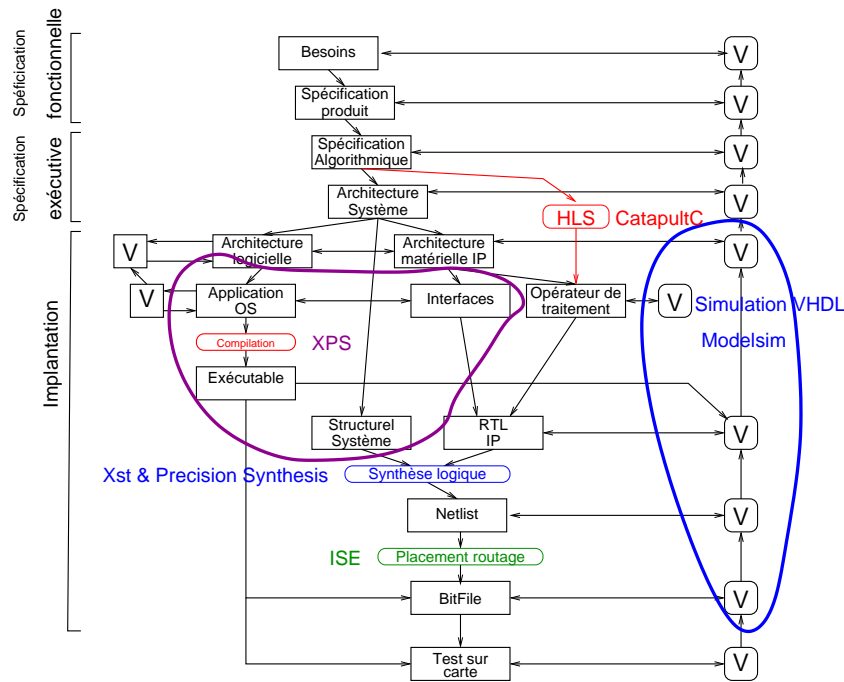


FIG. 1 – Le flot de conception SoPC sur la plate-forme OCAE. Les outils Xilinx peuvent être remplacés par l’environnement Altera.

## 2 Outils et méthodologie

L’objectif des projets proposés aux étudiants est de mettre en oeuvre une chaîne complète de conception, jusqu’au prototypage sur un circuit programmable SoPC. Le choix s’est porté sur l’environnement XPS de la société Xilinx et nous disposons de cartes d’évaluation de différentes générations. Les systèmes conçus comportent tous une partie logicielle et matérielle.

### 2.1 Vue globale du flot de conception

Le flot de conception utilisé par les étudiants est représenté schématiquement sur la figure 1. Ce flot vise à concevoir des systèmes logiciels et matériels dont un exemple est donné figure 3, page 6. Pour la plupart des projets, les étudiants partent d’une spécification de haut niveau, le plus souvent à partir d’une application logicielle de référence ou même de la description fonctionnelle et algorithmique du système à implanter.

Dans une première phase les étudiants sont amenés à spécifier l’architecture système de leur projet et de détailler le partitionnement

logiciel/matériel. Cette première phase est essentiellement “manuelle”, les outils d’aide au partitionnement étant abordés dans d’autres modules de formation.

Dans une seconde phase, les parties logicielles et matérielles sont conçues, d’abord séparément puis ensemble. Le développement matériel suit un cycle de conception/implantation/vérification propre, qui, par exemple, peut être basé sur des tests unitaires extraits de la spécification algorithmique.

Cette phase de développement logiciel/matériel peut être court-circuitée par un flot de synthèse de haut niveau (HLS) qui permet de produire une description RTL d’opérateurs matériels directement à partir d’un code C++, dans l’environnement CatapultC.

Dans les deux cas, une attention particulière est portée aux interfaces entre le logiciel et le matériel. La spécification des méthodes d’accès aux données par les opérateurs de traitement est affinée, en interaction avec l’architecture système (données en mémoire locale, centrale, etc..). Les mécanismes de communication à travers les bus système PLB Coreconnect sont spécifiés, ainsi que les synchronisation par interruptions.

Une fois logiciel et matériel développé, l'environnement XPS permet de produire une description structurelle complète du système. Cette description permet une première validation par simulation des interactions logiciel/matériel. Un Instruction Set Simulator (ISS) permet la simulation VHDL de l'exécution sur la plate-forme matérielle du code exécutable produit par la compilation du logiciel. Cette première validation nécessite une simplification du logiciel applicatif, le plus souvent sans système d'exploitation.

Une fois le système validé, les étapes classique de synthèse logique puis placement/routage permettent de reproduire ces tests unitaires sur les cartes de développement.

## 2.2 OS embarqué

De nombreux projets font appels à des systèmes d'exploitation déployés sur les processeurs présents dans les FPGA Xilinx. Le processeur PowerPC, ainsi que les dernière version de MicroBlaze, permettent l'utilisation du noyau Linux. Ce noyau est déployé en installant "Buildroot", un environnement de cross-compilation et de génération du système de fichier racine basé sur `uclibc`, une version légère de la `libc`. En plus d'un cross-compilateur utilisé pour compiler le noyau, les drivers et les applications, cet environnement génère le système de fichier et tous les services nécessaires au fonctionnement de l'OS.

Cet environnement permet de faire fonctionner sur les cartes de développement tout logiciel s'exécutant sur un OS Linux/Unix sur PC, les performances mises à part.

L'avantage majeur de la mise en place d'un tel OS est de faciliter les communications avec la station de développement. En effet, une liaison ethernet entre le PC et la carte permet la mise en place de l'échange de fichiers grâce au protocole NFS. Ainsi, les résultats des calculs sur les opérateurs de traitement peuvent être directement stockés dans des fichiers pour comparaison avec des résultats de référence sur la station de développement.

## 3 Exemples de projets

### 3.1 Sécurité et SoC

#### 3.1.1 Système sécurisé à chiffrement asymétrique

L'algorithme RSA est à la base de nombreux protocoles de sécurité. Il requiert toutefois des temps de calcul élevés, lorsqu'il est programmé sur des processeurs typiques des systèmes embarqués. L'optimisation des performances conduit donc souvent à implanter un coprocesseur de chiffrement spécialisé. L'objectif de ce projet est de concevoir un tel accélérateur matériel et de l'insérer dans une plate-forme SoPC afin d'évaluer le gain en performances obtenu.

#### 3.1.2 Système sécurisé reconfigurable dynamiquement

Un nombre croissant de systèmes embarqués doit prendre en compte des contraintes de sécurité, impliquant l'usage de coprocesseurs de chiffrement basés sur différents algorithmes. Une architecture reconfigurable dynamiquement permet de pallier au manque de ressources. Le coprocesseur implanté est alors défini tout au long de l'application de l'exécution en fonction des besoins, et changé lorsque nécessaire. L'objectif du projet est d'implanter une telle architecture permettant de changer l'algorithme de chiffrement à la volée.

### 3.2 Sécurité de fonctionnement

En regard de la complexité des SoC et MP-SoC, il n'est plus possible de garantir que 100% des composants matériels présents sur la puce seront sans défauts. Les défauts peuvent être détectés soit au moment de la fabrication (en usine), soit après mise en service (dans l'équipement), suite à un phénomène de vieillissement prématuré. Il est donc devenu primordial de pouvoir, en cours de fonctionnement, pouvoir collecter différentes informations fournies par les capteurs répartis dans le système, les analyser et prendre la décision qui s'impose.

### 3.2.1 Coprocesseur de monitoring supportant la norme IEEE 1687

Pour répondre au problème de la collecte d'information, un standard définissant une interface d'accès aux différents cœurs intégrant des informations (grandeurs physiques et/ou compteurs d'activité) a vu depuis peu le jour. Il s'agit du standard IEEE 1687 appelé aussi JTAG.

### 3.2.2 Détection d'erreurs transitoires par analyse de signature logicielle (Control Flow Checking) pour systèmes multi-tâches

Le comportement de l'application peut aussi être altéré suite à un changement d'état d'un point mémorisant du système provoqué par une particule ionisante (SEU pour Single Event Upset). Pour éviter un comportement inattendu (voire l'arrêt complet) de l'équipement suite à l'apparition d'une faute, il est donc nécessaire de prévoir des mécanismes distribués de détection en ligne de fautes, d'arrêt synchronisé des tâches et de reprise rapide de l'exécution du programme à partir d'un état sain (on line detection, checkpoint and rollback).

## 3.3 Accélération matérielle et HLS

### 3.3.1 Visionneuse de fractale sur un système processeur-coprocesseur spécialisé

Les fractales sont des courbes irrégulières créées à partir de fonctions itérées ou récursives. Pour calculer un fractal, il faut donc bien souvent répéter un calcul simple, mais néanmoins très précis, un grand nombre de fois. L'objectif de ce projet est de déployer une application de visionneuse de fractale sur un système embarqué sur FPGA et d'utiliser un outil de synthèse d'architecture pour mettre en oeuvre un coprocesseur matériel dédié.

### 3.3.2 Synthèse vocale sur un système processeur-coprocesseur spécialisé

La synthèse vocale permet de recréer une parole à partir des textes fournis en entrée. Le

but de ce projet est de déployer l'application de synthèse vocale *espeak* sur un système embarqué sur FPGA et d'utiliser un outil de synthèse d'architecture pour mettre en oeuvre un coprocesseur matériel dédié.

### 3.3.3 Accélération de la 3D RayTracing

Ce projet est détaillé section 4.

## 3.4 Système d'exploitation

### 3.4.1 Driver Linux pour interface réseau Remote-DMA

Le Remote-DMA (RDMA), est un protocole réseau permettant de réduire les copies subies par un message qui arrive sur un port réseau. Dans un schéma classique, un message IP entrant est copié depuis un buffer vers les couches réseau supérieures pour finalement arriver dans l'espace mémoire virtuel du processus applicatif. Avec RDMA, le message arrive directement à une adresse physique contenue dans l'en-tête du message. Cette adresse peut être établie au préalable par une architecture client-serveur.

### 3.4.2 Portage de DNA sur Microblaze

DNA est un système d'exploitation développé par l'équipe TIMA-SLS. Il implémente toutes les fonctionnalités classiques d'un OS, notamment :

- Le support multiprocesseur (SMP)
- Le multithreading
- Les interruptions & exceptions
- Les entrées/sorties

Il repose entièrement sur un HAL spécifié ultérieurement qui permet de séparer l'OS de l'architecture. Dans ce projet, le HAL sera conçu pour le processeur MicroBlaze.

## 4 Projet "3D Ray-Tracing"

Le lancer de rayon, ou "Ray-Tracing", est une application graphique attractive car les résultats sont directement visualisables. Cette ap-

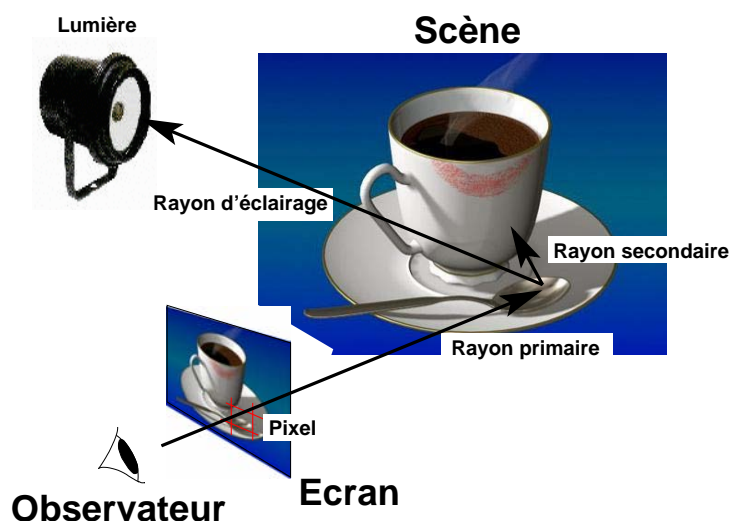


FIG. 2 – Principe du lancer de rayon

plication met en jeu tous les aspects de la conception des SoC, depuis la description algorithmique du traitement jusqu'à la réalisation d'un prototype.

Le lancer de rayon a pour objectif de faire de la synthèse d'image photo-réaliste: par calcul, une image est produite à partir d'une scène composée d'objets, chacun étant représentés par un maillage de triangles. L'image obtenue correspond au point de vue d'une caméra placée dans cette scène, comme illustré par la figure 2. Afin d'obtenir des effets photo-réalistes, l'image est calculée en simulant le trajet de la lumière dans la scène. Plus exactement, pour calculer l'intensité lumineuse d'un point du capteur de la caméra virtuelle, c'est le trajet inverse de la lumière arrivant en ce point qui est simulée. Pour chaque pixel de l'image produite, l'algorithme génère un rayon de lumière passant par ce pixel et la focale de la caméra, ce rayon est dit *rayon primaire*. L'objet vu en ce pixel est trouvé en calculant l'intersection entre le rayon et les objets de la scène, c'est à dire avec tous les triangles qui composent cette dernière. L'objet vu est celui dont l'intersection avec le rayon est la plus proche de la focale. Comme l'intensité lumineuse produite par cet objet dépend de son éclairage, des rayons de lumière sont à nouveau envoyés dans la scène afin déterminer les éventuelles occlusions des sources de lumière, ce sont les *rayons d'éclairage*. Des effets de réflexion et réfraction peuvent aussi être

obtenus en lançant récursivement les rayons réfléchis et réfractés inverses, dits *rayons secondaires*. L'algorithme s'arrête soit sur un critère d'arrêt soit lorsqu'une profondeur de récursion maximum est atteinte.

Il est compréhensible que le calcul d'intersection entre un rayon et un triangle est un des goulots d'étranglement de cet algorithme et que son accélération matérielle est nécessaire. Un premier partitionnement logiciel/matériel conduit à l'architecture représentée figure 3. Un accélérateur (ou IP) d'intersection rayon/triangle est géré par un processeur PowerPC qui fait le reste des calculs (lecture de la scène, génération des rayons, calculs d'éclairage, etc ...). Cet accélérateur prend en entrée une liste de rayons ainsi qu'une liste de triangles et détermine s'il y a intersection entre chacun des rayons et chacun des triangles. Traiter les données en paquets permet un meilleur recouvrement des calculs et des communications logiciel/matériel. En effet, comme cet accélérateur permet le calcul d'une intersection en 4 cycles d'horloge<sup>1</sup>, la durée d'un simple calcul d'intersection serait minorée par le coût de l'écriture des données puis la synchronisation par interruption. Il est à noter qu'une autre stratégie aurait pu consister en l'utilisation du calculateur d'intersection

<sup>1</sup>Une soixantaine d'opérations arithmétiques sont nécessaires au calcul d'une intersection/rayon, ce qui correspond à une centaine d'instructions d'un processeur classique

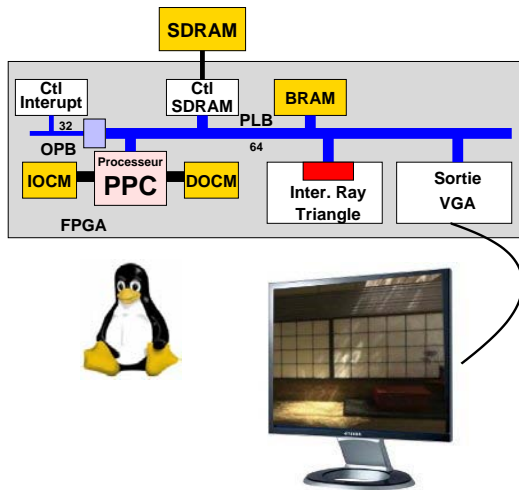


FIG. 3 – SoC pour le lancer de rayon

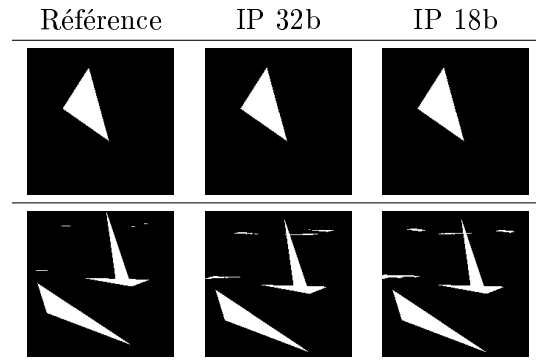
comme co-processeur du coeur PowerPC.

Du fait de la complexité de l'application, les différentes étapes sont abordées par différents binômes, sur plusieurs années. Le tableau 1 donne les résultats de calculs d'intersection produits par une IP conçue dans l'environnement HLS CatapultC par un binôme d'étudiants. Les images sont produites par l'environnement de simulation conçus par les étudiants. Elles illustrent l'effet de la précision des calculs sur la qualité des résultats. Afin de comparer les méthodes de conception classiques et la HLS, d'autres binômes se sont essayés à concevoir une IP en écrivant le RTL d'une architecture pipeline.

L'intégration de l'opérateur d'intersection rayon/triangle dans un système complet est un nouveau projet. L'objectif est de pouvoir récupérer le code de l'application de référence, qui fonctionne sous Linux, et de remplacer les fonctions logicielles de calcul d'intersection par un appel à un driver contrôlant l'opérateur matériel.

## 5 Conclusion

La formation à la conception des SoC par la pratique est rendue possible grâce aux outils de prototypage. La quantité d'informations et de connaissances nécessaires rend difficile la mise en place de projets utilisant l'ensemble du flot de conception. Des projets qui s'étalent sur plusieurs années pourraient aider les étudiants à comprendre les difficultés liées à la con-



TAB. 1 – Comparaison entre références et résultats de la HLS de l'opérateur d'intersection rayon/triangle, pour différentes précisions des calculs (32b/18b)

ception des SoC et à la transmission des connaissances: ils doivent faire face à des codes et rapports précédants, sans que les personnes à leur origine ne soient là puis doivent eux-même produire des documents exploitables par leurs successeurs.

Nous avons pu constater que les étudiants qui prennent en main les outils HLS arrivent rapidement à des résultats du fait de l'environnement intégré de CatapultC. Cet environnement permet la simulation de l'IP dans un programme identique à celui utilisé pour valider l'algorithme initial. De ce fait, il est plus aisé de fournir des données au calculateur puis de récupérer les résultats pour comparaison avec une référence. Cependant, le paramétrage de l'outil HLS et la grande variabilité au code d'entrée, pour un algorithme donné, rend difficile la génération d'une architecture efficace sans un minimum de recul.

La diversité des outils et cartes d'émulation fournies par la plate-forme OCAE permet l'expérimentation de nombreux flots de développements, de la conception jusqu'au prototypage.