Approche complète de développement des IPs pour les SoCs

A complete approach of IPs development for SoCs

AhceneYoucef Benabdallah* & Rachid Boudour

Laboratoire des Systèmes Embarqués(LASE), Université Badji Mokhtar, BP12, 23000, Annaba, Algérie.

Soumis le : 24/05/2014 *Révisé le : 10/01/2016* *Accepté le : 12/01/2016*

الملخص:

تزايد الفجوة بين إنتاج النظم و التطور التكنولوجي في الآونة الأخيرة, ساهم بالضرورة إلى إعادة استعمال المركبات المصممة و المفحوصة مسبقا تدعى بالمركبات الإفتراضية. إلا أن أن استعمال هذه المركبات لا يتم بدون مواجهة بعض الصعوبات نذكر منها غياب منهج توافقي لتصميم و إنشاء هذه المركبات,مما يجعل دمجها و الإتصال فيما بينها صعب الوصول إليه في نفس التطبيق, و كذلك الإهتمام و تركيز العمل على بعض أنماط هذه المركبات فقط دون غيرها سواء السوفت فقط او الهارد فقط, وكذلك وجود مناهج لتطويرها إلا أنها غير كاملة في مراحلها.
نقدم في هذا العمل منهج كامل لتطوير المركبات الافتراضية, انطلاقا من Métamodèle في UML و الذي يجمع الوصف البرمجي و الكياني مستعملين C++ كلغة برمجة للسوفت و System C كلغة وصف للكيان.
منهجنا تم تجريبه على وحدة الحساب و المنطق ذات معاملات باستخدام SystemC و UML. استعمالنا لـ Cofluent و Quartus II كأداتين من اجل التطوير سمح لنا بالحصول على عدة نماذج, و النتائج كانت مشجعة للغاية.

الكلمات المفتاحية: *مركبات افتراضية- نظام على رقاقة- إعادة الاستعمال- UML-Système C.*

Résumé

Le fossé séparant la production des systèmes et l'évolution technologique n'a cessé de croître ces derniers temps, conduisant à une réutilisation de composants préconçus et pré-vérifiés appelés IP (Intellectual Property). L'essor de ces derniers ne va pas sans rencontrer quelques difficultés. Nous citons entre autres un manque de standards pour la réalisation des IPs, rendant une intégration voire une communication difficile à établir entre les mêmes composants d'une application, un intéressement seulement à certains types d'IPs matériel ou logiciel et une approche de développement généralement incomplète.
Dans ce papier, nous présentons une démarche complète de développement des IPs en partant d'un métamodèle en UML prenant en charge les deux descriptions matérielles et logicielles en utilisant C++ comme langage de programmation et SystemC comme langage de description matérielle.
Notre approche a été expérimentée sur une unité arithmétique et logique paramétrée avec SystemC et UML. Les outils Cofluent et Quartus II nous ont servi à obtenir les différents modèles. Les résultats sont encourageants.

Mots clés : *IPs –SoC – Réutilisation – UML -SystemC*

Abstract

The gap separating between the production systems and the technological development has not ceased from thriving in recent times, leading to the reuse of predesigned components and others pre-verified so-called IP. The growth of the latter is not going to proceed without encountering some difficulties, we can cite among others a lack of standards for the implementation of IPs, making the integration especially communication difficult even for being established between the same components of an application, only one incentive to some types of hardware or software IPs and generally incomplete development approach. In this paper, we present a comprehensive approach for the development of IPs starting from a UML metamodel that supports both hardware and software descriptions using C++ as a programming language and SystemC as a hardware description language. Our approach was tested on an arithmetic and logical unit set by UML and SystemC. CoFluent and Quartus II tools have helped us to get the different models. The results are encouraging.

Key words : *IPs – SoC - Design and Reuse – UML – System C*

* *Auteur correspondant : benabdallah.ahcene@yahoo.com*

1. INTRODUCTION

L'évolution rapide de notre société a conduit à l'élaboration d'équipements(systèmes) de plus en plus complexes, capables de traiter des flots d'informations de nature et d'origine très diverses et dont l'accès doit rester néanmoins simple et rapide. La conception et l'intégration de ce type d'équipements demande aujourd'hui encore des développements longs, fastidieux, coûteux et par conséquent en inadéquation avec cette évolution et flexibilité que doivent respecter les matériels électroniques. Ainsi, de nouvelles techniques de conception doivent-elles être trouvées. Les contraintes à prendre en compte sont typiquement les suivantes :

- Réduire au maximum le temps d'arrivée d'un produit sur le marché (time to market)
- Garantir un résultat de bonne qualité (performance, surface, consommation, …)
- Fiabiliser le cycle de développement (augmenter la complexité d'un circuit rend sa vérification moins aisée) [1].

Face à ces contraintes, les concepteurs essaient de favoriser la réutilisation de code, en s'orientant vers l'assemblage de blocs préconçus et pré-vérifiés désignés sous le terme de composants virtuels. Ce concept, né dans le milieu des années 90, a donné lieu à plusieurs vocables pour désigner ces blocs réutilisables : composants réutilisables, composants virtuels, IPs, ou plus simplement macros.

Afin de privilégier la réutilisation de composants, d'architectures, ou de favoriser l'interconnexion entre des systèmes modélisés avec des notations différentes, il apparaît évident qu'une notation consensuelle est nécessaire. De ce constat est née la notation UML (Unified Modeling Language).

Le travail que nous présentons dans ce papier a permis de définir une approche complète de conception et de réutilisation afin de construire et d'utiliser des IPs dans le SoC (System on Chip) basé sur un méta modèle des IPs en UML.

Le papier est divisé en sections. Dans la suite, nous rappelons en section 2 les concepts préliminaires autour des IPs. La section 3 est consacrée aux travaux antérieurs. La section 4 met l'emphase sur notre approche de développement. Quant à la section 5, elle décrit de la sémantique d'un concept UML en ajoutant de nouvelles règles ou en modifiant les anciennes. les détails de l'implémentation suivie d'une évaluation sur un exemple réel. La section 6 s'achève en proposant de futurs travaux.

2. CONCEPTS PRELIMINAIRES

2.1 Composant Virtuel

Un composant virtuel ou IP est la spécification d'un composant réalisant une fonction bien déterminée pouvant être synthétisée, donc réutilisée, par un utilisateur n'ayant pas participé à la spécification de ce composant. Cette démarche comprend que seule une appréhension de la fonction réalisée est nécessaire, en aucune manière les détails de la spécification [2]. Cependant cette absence de connaissance détaillée ne doit pas empêcher l'obtention par l'utilisateur de performances identiques après synthèse, à celles obtenues par le créateur de l'IP.

VSIA(Virtual Socket Interface Alliance) [3] définit trois classes de composants virtuels matériels en fonction du niveau d'abstraction de leur description synthétisable : hardware, firmware et software. Le premier type est représenté en layout (Dessin des masques après placement/Routage), le deuxième est sous forme de netlist (Réseau de porte logique) et le troisième est représenté en RTL(Register Transfer Level) ou en C++ [4].

2.2 UML pour les Systèmes Electroniques

UML est un langage visuel de modélisation, utilisé pour documenter, spécifier et visualiser graphiquement les aspects d'un système logiciel [5].UML2.0 a apporté plusieurs améliorations significatives pour soutenir les concepts liés à la conception conjointe matérielle et logicielle. Les chercheurs ont pu élargir le champ de modélisation de ce langage du logiciel vers le matériel par le biais des extensions d'UML et la création de profil relatif au domaine tout en utilisant les notions suivantes:

- **Stéréotype**: Chaîne de caractères qui, associée à un élément UML existant, permet de désigner un nouveau type d'élément UML à partir de concepts standards.
- **Contraintes**: Chaîne de caractères qui, associée à un élément UML permet l'extension

Plusieurs profils ont été proposés pour permettre la prise en compte des caractéristiques de base des systèmes

électroniques et embarqués dans un modèle UML, aussi bien sur le plan de la conception et de l'analyse que sur le plan de la validation et de la génération du code exécutable. Parmi ces profils, citons : Profil UML de test (UML Testing Profile)[6], profil UML pour la modélisation de la qualité de Service QoS (Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms)[7], profil UML pour l'ordonnancement, Performance et Temps SPT (Schedulability Performance and Time) [8], profil UML pour les systèmes d'ingénierie SysML[9], profil UML pour leSoC [10], profil MARTE (Modeling and Analysis of Real-Time Embedded Systems)[11] et profil UML-SystemC [12].

2.3 Génération Automatique du SystemC à partir des Profils UML

SystemC [13] est un langage de description de matériel/logiciel, permettant de représenter et de simuler des circuits et des systèmes sur puce à différents niveaux d'abstraction. Conçu comme librairie C++, il bénéficie de types de données et d'environnements de compilation du langage C++.SystemC remplace les HDL(Hardware description Language) comme Verilog et VHDL dans beaucoup de situations [14]. La figure 1 détermine une comparaison du SystemC par rapport aux autres HDL.

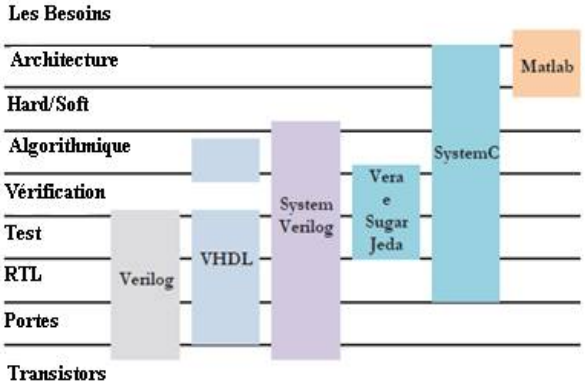

Figure 1 : Comparaison SystemC –HDL

Ceci ne signifie point que ces HDL ne sont plus utiles, mais SystemC supporte une nouvelle approche de conception d'un système. Parmi ses apports, nous citons :

– Il permet de garder un même langage d'un bout à l'autre du flot de conception

– Il a un noyau de simulation très fort pour permettre aux concepteurs de faire des tests et des simulations dessus.

– SystemC soutient la conception à un niveau plus élevé d'abstraction, tandis que la majeure partie des HDL s'appuie sur le niveau RTL.

Ce langage peut être généré automatiquement à partir des notations graphiques d'UML voire des profils en utilisant des règles de correspondance (Mapping rules), ces règles localisent les caractéristiques structurelles et comportementales du langage désiré. La figure 2 illustre la relation entre UML et le code généré

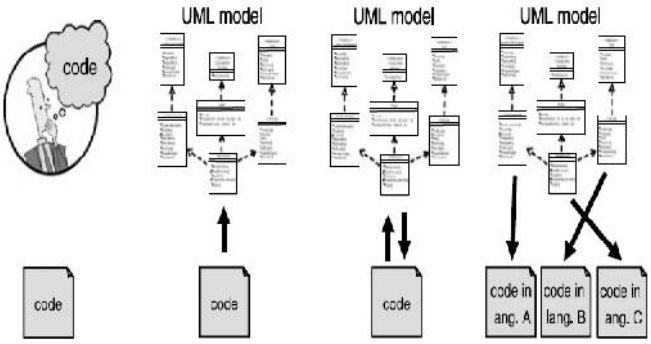

Figure 2 : Relation entre le modèle UML et le code généré [15]

La recherche scientifique a beaucoup travaillé sur cet aspect grâce aux avantages procurés dans la phase de développement des IPs en particulier et dans l'embarqué en général. Des travaux récents sont abordés dans [15, 16, 17].

3. TRAVAUX ANTERIEURS

De nombreux travaux de développement des IPs ont été répertoriés dans la littérature pour arriver à une standardisation. On peut les classer selon trois facettes:

3.1 Conception et Synthèse

Parmi les travaux liés à cette facette, nous citons:

Le diagramme en Y(**Y-Chart**), introduit par GAJSKI et KUHN [18], reposant sur le fait que la spécification d'un système combine des informations relatives à trois principaux axes de représentation : comportemental, structurel et physique. Dans chacun de ces axes, un système peut être décrit avec une plus ou moins grande richesse de détails, mais les membres du groupe de travail **RTWG** (RASSP Terminology Working Group) ont observé que les trois axes du diagramme en Y restent limités aux aspects matériels d'un système et sont donc insuffisants dans le cadre d'une démarche de conception

conjointe matériel/Logiciel. La taxonomie qu'ils proposent repose sur cinq axes dont les quatre premiers sont dupliqués afin de représenter séparément la vue *interne* et la vue *externe* (interface) d'un système ou d'un composant [19].
Le manuel de réutilisation [20] présente quelques modèles classiques de développement des blocs électroniques. Nous rappelons le Modèle en Cascade basé sur le travail des équipes de spécification en fonction des exigences de fonctionnalités. Une équipe de conception code ces spécifications en effectuant les vérifications fonctionnelles. Ensuite, l'équipe de synthèse effectue la compilation du bloc jusqu'à obtention d'un réseau de portes vérifiant les contraintes temporelles. Dans un quatrième temps, le bloc est placé et routé pour obtenir un prototype fabriqué et testé. Le risque majeur de cette approche est que lorsque le circuit ne satisfait pas à toutes ses contraintes, il faut réitérer l'ensemble du flot par contre le modèle en spiral met en concurrence l'ensemble des équipes que sont : les concepteurs logiciels et matériels, l'équipe de synthèse et de placement et routage. Dans la référence [21], Bernard Laurenta focalisé ses travaux sur la source des erreurs pendant les étapes de développement. Ainsi il a introduit une phase de validation dont l'objectif saillant est d'isoler la compréhension et la validation de la spécification du codage et par conséquent de réduire la distance cognitive entre les éléments du couple spécification-réalisation. Cependant dans les références [22] et [23], Guillaume Savaton et Bertrand Le Gal, ont travaillé sur un axe identique, en proposant un IP comportemental comme un quatrième type des Composants Virtuels, ce qui va à l'opposé de la recommandation VSIA citée plus haut.
Ces approches sont limitées aux aspects matériels.

3.2 Spécification et Modélisation :

Dans la seconde facette, on cite les travaux effectués dans les références [24][25][26] et [27], consistant à modéliser l'IP avec UML et à proposer des types de composants virtuels logiciels en C++ et SystemC. Ces travaux sont liés à quelques types d'IPs logiciels et dont les modèles UML proposés ne peuvent pas être généralisés à l'ensemble des IPs.

3.3 Environnements existants

Dans cette dernière facette, nous citons quelques environnements récents, utilisés dans le développement des IPs.

Marte :
Marte (Modeling and Analysis of Real Time Embedded Systems) profil dédié à modéliser les systèmes embarqués de temps réels, fondé sur quatre paquetages : le paquetage de base, le modèle de conception, le modèle d'analyse, et les annexes [11].
Ce profil n'assure pas la synthèse matérielle, ni les besoins fonctionnels.

Gaspard 2:

Gaspard2 (Graphical Array Specification for Parallel and Distributed), profile UML2.0 vise le domaine de traitement de signal, utilisant le principe MDA et l'approche en Y à différents niveaux d'abstraction [28].
Cette méthode ne capture pas les besoins fonctionnels d'un composant, en plus qu'elle est dédiée à un domaine spécifié.

MoPCoM :

MoPCoM (Méthodologie de Modélisation et Spécialisation de Plateformes et Composant) est une approche récente définie dans le projet MoPCom, et emploie Marte pour la modélisation d'application dynamique, disposant d'un niveau de modélisation détaillé DML (Detailed Modeling Level)[29].
Cette méthodologie ne dispose pas d'un outillage de validation voire de vérification des composants générés.

4. PRESENTATION DE NOTRE APPROCHE

Notre approche de développement des IPs est basée sur la séquentialité des étapes suivantes montrées sur la figure 3.

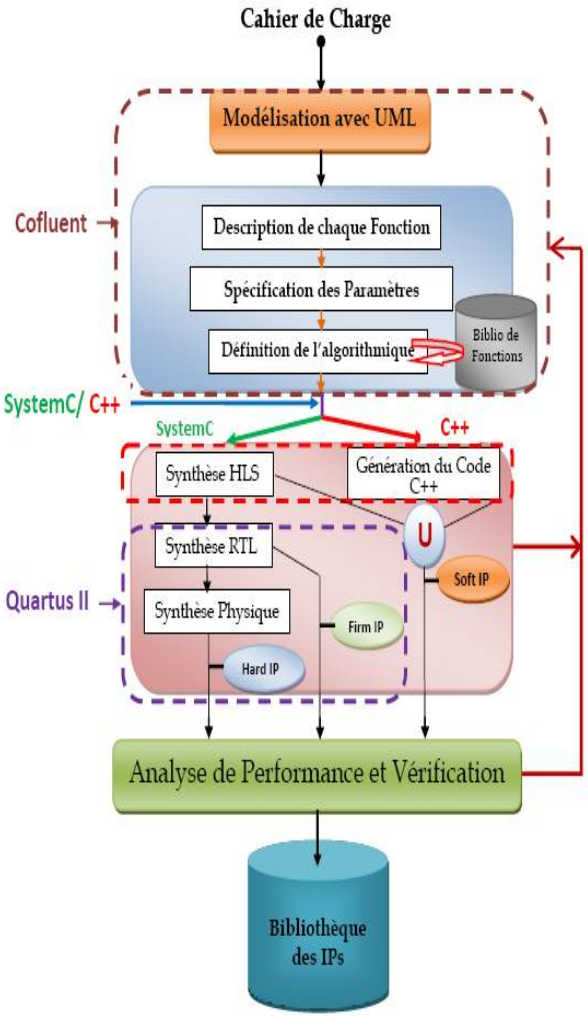

Figure 3 : Approche complète de développement d'un IP

Après une extraction et une analyse des besoins à partir d'un cahier de charge et une formulation des spécifications techniques et fonctionnelles de l'IP, on passe à la modélisation à partir de notre métamodèle puis à la description comportementale et à la génération automatique du code. Ensuite on accède aux différents types de synthèse et enfin la phase d'analyse de performance et de vérification de l'IP. A l'issue de cette ultime étape, l'IP est prêt à être stocké dans notre bibliothèque en vue d'une intégration éventuelle dans un SoC.

4.1. Modélisation à partir d'un méta modèle

On commence par métamodéliser le composant virtuel à partir de spécifications en utilisant le méta modèle proposé en figure 5, afin d'obtenir un modèle UML pour le composant voulu en recourant à l'un des outils supportant les profils UML. La finalité étant de modéliser le composant et de générer automatiquement son code SystemC à partir des modèles UML [30].

Représentation graphique d'un composant virtuel

La figure 4 montre une représentation graphique d'un IP.

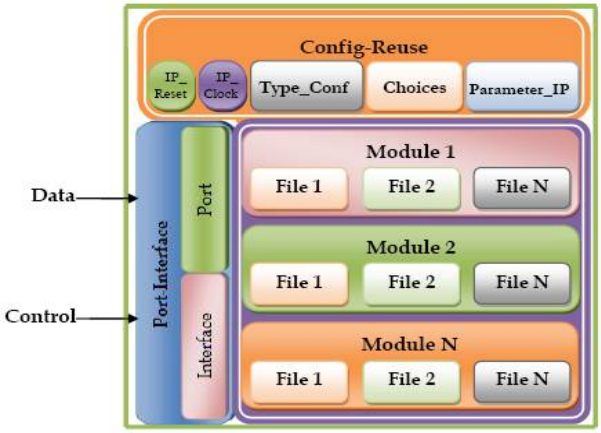

Figure 4 : Représentation graphique d'un IP

Selon cette figure, le composant virtuel est constitué essentiellement des éléments suivants :

- **Config-Reuse** : Cet élément a pour mission de configurer l'IP, on trouve plusieurs champs :
  - **Parameter_IP** : Paramétrisation de l'IP par rapport au système avec des valeurs ;
  - **Choices** : Paramétrisation de l'IP par rapport au Système avec des fonctions ;
  - **Type_Conf** : Configurer l'IP comme maitre, esclave, système ;
  - **IP_Clock** : Horloge de l'IP dans la description matérielle ;
  - **IP_Reset** : Réinitialisation de l'IP pendant le fonctionnement.
- **Port-Interface :** Cet élément a pour but d'échanger les données entre l'IP et le système. Il est composé de :
  - **Port :** lieu d'échange de données et de messages dans la description matérielle.
  - **Interface :** lieu d'échange de données et de messages dans la description logicielle.
- **Module :** Il représente les composants internes de l'IP. Chaque module est constitué de plusieurs codes Files assurant une fonctionnalité dans l'IP.

- Data : Il désigne un ensemble de données échangées entre le composant et le système.
- Control : Il représente un ensemble de fonctionnalités permettant le contrôle et l'adaptation de l'IP à l'environnement d'intégration.

Le métamodèle proposé des IPs

Notre méta modèle couvre les composants virtuels de différentes descriptions matérielles et logicielles. La figures 5 présente le métamodèle. La figure 6 illustre les stéréotypes utilisés dans ce métamodèle.

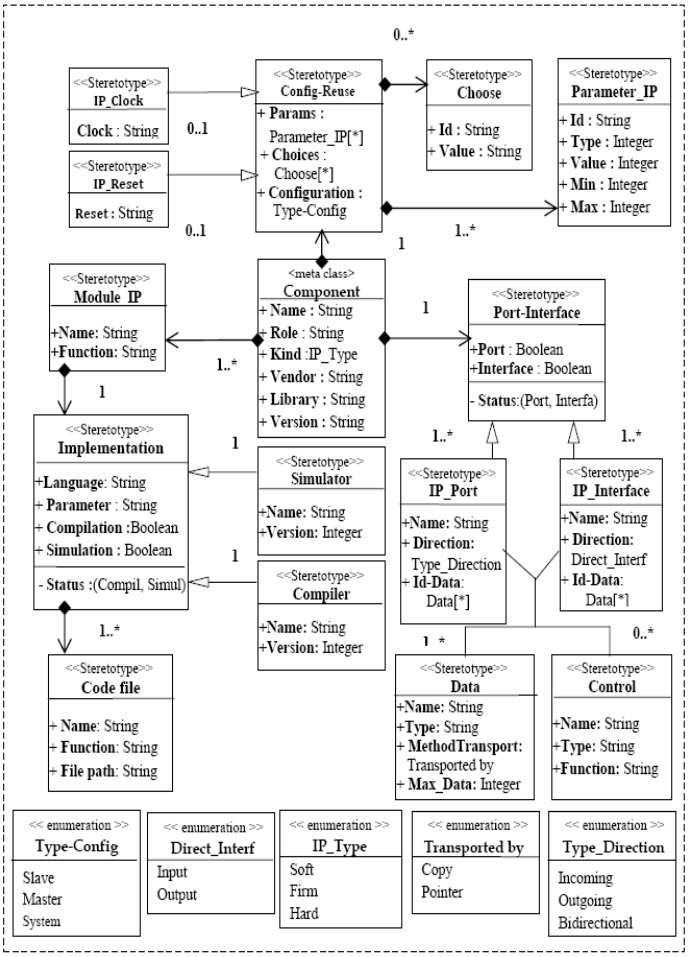

Figure 5: Métamodèle proposé pour les IPs

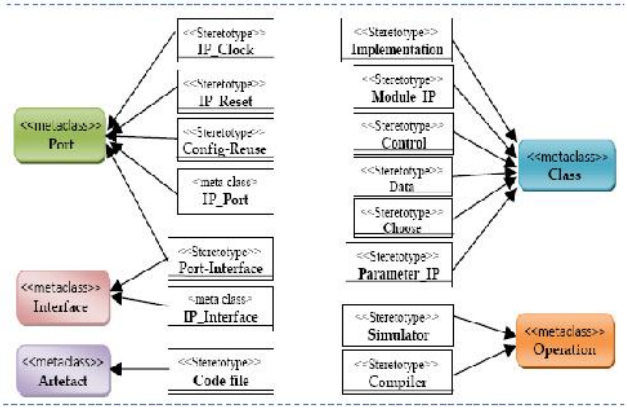

Figure 6: Stéréotypes utilisés

4.2 Description Comportementale

Le comportement du futur composant virtuel (ou de l'un de ses modules), peut être décrit comme suit:

- oit par un organigramme
- oit sous la forme d'un algorithme

Le fait de décrire le comportement et non la structure d'un composant, devrait être considéré comme une simplification du processus de conception. La description comportementale repose sur les notions suivantes :
Description de la fonction : Dans cette étape, on la décrit dans un format de haut niveau qui peut être un graphe, un pseudo algorithme ou en langage naturel
Spécification des paramètres : Le choix du sous ensemble des paramètres est effectué selon le degré de généralité fixé au préalable. Ce dernier influe sur la complexité de l'algorithme.
Définition des propriétés de chaque paramètre : Les paramètres peuvent être continus ou discrets, en rapport avec l'exigence du problème.
Définition de l'algorithme : L'algorithme est conçu selon la description de la fonction souhaitée, de la spécification des paramètres et de leurs propriétés.

4.3 Synthèse et Génération du Code

Génération du code

Après la génération du modèle UML du composant virtuel, ainsi que sa description comportementale, il ne reste plus qu'à faire générer le code voulu en SystemC [31].
Synthèses

Il existe trois types de synthèses dans l'architecture matérielle [21] :

- **Synthèse de haut niveau**

C'est le premier type de synthèse du composant virtuel. Il accepte comme entrée une description comportementale en langage HDL, le SystemC dans notre cas. En d'autres termes, c'est la sortie de l'étape précédente de notre approche de conception (étape de génération du code) afin de produire des IPs logiciels sous une description RTL.

- **Synthèse logique**

C'est le deuxième type de synthèse d'un composant virtuel afin de générer un IP de type microprogrammé (firmware). La synthèse logique permet la transformationd'une description RTL d'un système en une description au niveau portes logiques (Gate netlist).

- **Synthèse physique**

Durant ce processus, les différentes parties d'un circuit sont automatiquement placées et connectées en fonction du problème à résoudre. Le placement-routage est en particulier employé dans le cadre des FPGA (Field Programmable Gate Array).

4.4 Analyse de Performance et vérification

Après l'obtention de différents types de transformation, il ne reste plus qu'à vérifier le bon fonctionnement de l'IP, afin de détecter d'éventuelles erreurs et aussi de s'assurer de la préservation de l'objectif initial de cette conception [32, 33]. Les sources d'erreurs peuvent être dues à la conception ou à la réalisation. Le but de la vérification est de s'assurer que le produit final réalise bien la fonction souhaitée dans des conditions de fonctionnement prévues et qu'il ne provoque pas de conséquences graves dans les autres conditions.

5. IMPLEMENTATION

5.1 Outils utilisés

CoFluent Studio pour la Modélisation et la génération automatique du code

CoFluent Studio [34] est un outil visuel de modélisation et de simulation des stystèmes embarqués, basé sur Eclipse. La figure 7 présente les fonctinnalités de cet outil.

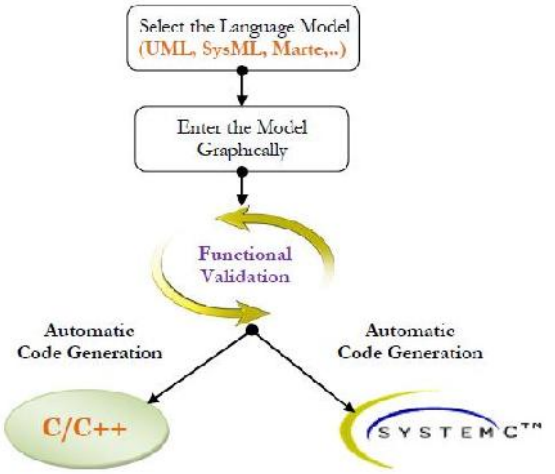

Figure 7 : Les fonctionnalités offertes par Cofluent Studio [34]

Les modèles sont capturés dans des diagrammes graphiques en utilisant les notations standard d'UML ainsi que les notations des profils SysML et Marte afin de générer automatiquement les deux langages de description logicielle C++ et la description matérielle SystemC.

La première étape permet de décrire les fonctionnalités du système en faisant abstraction de l'implantation matérielle future. Le modèle, intégrant des contraintes temporelles dès cette étape, est alors simulé, ce qui permet d'observer le comportement fonctionnel du système, avec une possibilité de vérifier la synchronisation des données.
La seconde étape consiste à choisir les éléments de l'architecture matérielle. Il s'agit d'unités de traitement : des processeurs, des FPGA, des structures de mémoire, … Une fois ce choix fait, le logiciel effectue le partitionnement souhaité par l'utilisateur. On obtient alors un modèle architectural complet du circuit, dans lequel chaque fonction est traitée par une entité logique (processeur ou composant matériel) [34].

Quartus II pour synthèse et vérification

Quartus est un logiciel développé par la société Altera [35], permettant la gestion complète d'un flot de conception CPLD ou FPGA. Apparu à la fin des années 90s, ce logiciel permet de faire une saisie graphique ou une description HDL (VHDL, RTL, …) d'architecture numérique, d'en réaliser une simulation, une synthèse et une implémentation sur une cible reprogrammable [36]. Ce processus est schématisé dans la figure 8.

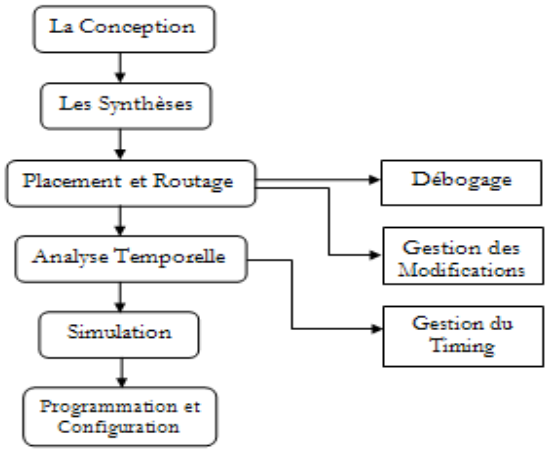

Figure 8: Etapes de développement dans Quartus II

Il comprend une suite de fonctions de conception au niveau système, permettant d'accéder à la large bibliothèque d'IP d'Altera et un moteur de placement-routage intégrant la technologie d'optimisation de la synthèse physique et des solutions de vérification. La version actuelle du Quartus est la 13.1

5.2 Etude de cas : UAL

L'approche proposée est appliquée sur une UAL, dont les paramètres de réutilisation sont :

a. Taille des registres : ce paramètre influe considérablement sur l'UAL en matière de précision de calcul, de coût, de taille de code. La reconfiguration de ce paramètre permet de toucher une large gamme de processeurs(8, 16, 32 ou 64 bits).

b. Jeu d'instructions supporté par le processeur : la sélection d'un ensemble d'opérations supportées par l'UAL est un paramètre qui influe essentiellement sur la fonctionnalité du processeur et la taille du code, et en second lieu sur le coût et l'espace d'embarquement.

Cette UAL assure le calcul de l'addition, la soustraction, la multiplication, la division, et la comparaison de deux nombres selon les figures 9 et 10.

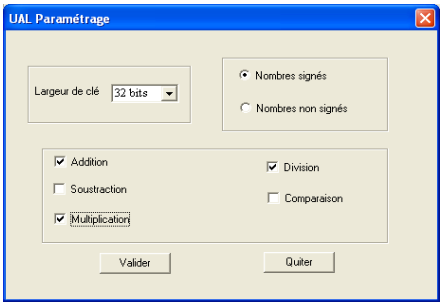

Figure 9 : Interface de l'UAL

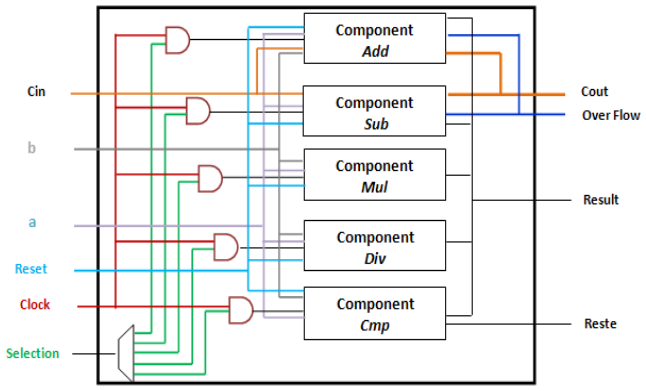

Figure 10 : Schéma de l'UAL

La figure 11 illustre les étapes de notre expérimentation sur une UAL.

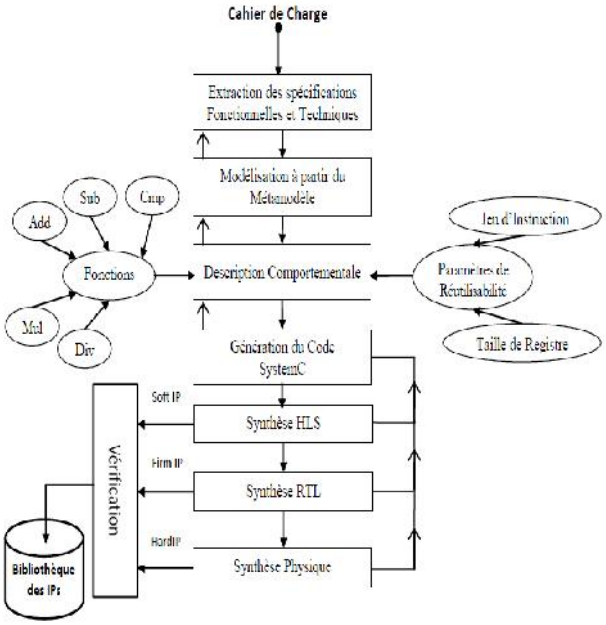

Figure 11 : Les étapes de l'éxpérimentation sur une UAL

Nous choisissons le composant Add avec un seul bit pour dérouler notre approche avec SystemC selon les figures 12,13,14 et 15.

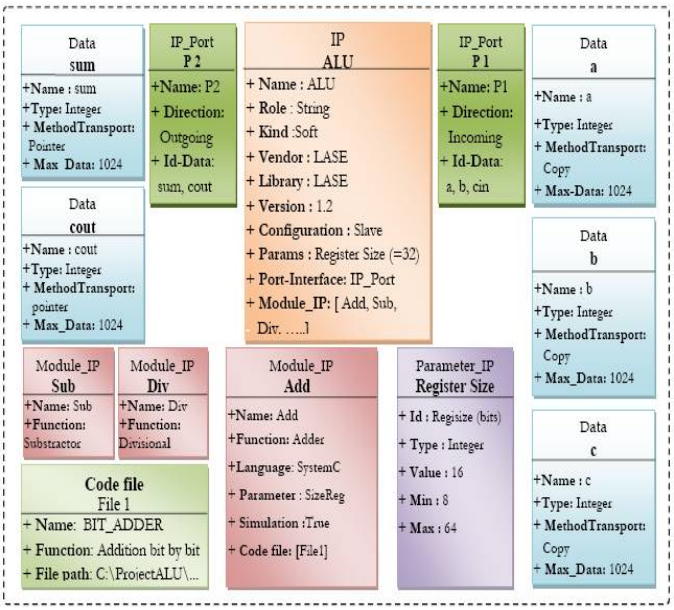

Figure 12 : Modèle UML de l'UAL

```
#include '' systemc.h ''
SC_MODULE (BIT_ADDER)
{
sc_in <sc_logic> a, b, cin  ;
sc_out<sc_logic> sum, cout ;

SC_CTOR (BIT_ADDER)
            {
SC_METHOD ( process 1) ;
sensitive<< a <<b << cin ;
            }

void process 1()
} ;

void  BIT-ADDER : : process 1( )
{
sc-logic  aANDb, aXORb, cinANDaXORb ;

        aANDb = a . read ( ) &  b. read ( ) ;
        aXORb = a . read  ( )  ^  b.read ( ) ;
        cinANDaXORb  =  cin . read  ( )  &  aXORb

       sum = aXORb   ^ cin . read ( )
       cout = aANDb | cinANDaXORb ;
```

Figure 13 : Code source (Code File) d'un additionneur un bit modélisé avec SystemC

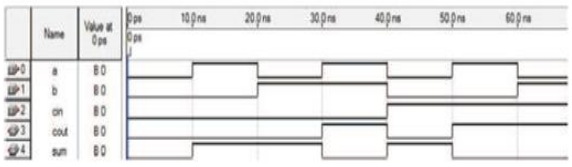

Figure 14 : Fonction de Simulation d'un additionneur-1bit

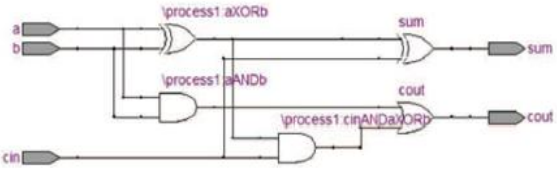

Figure 15 : La description RTL d'un additionneur-1bit.

6. DISCUSSION

Cette approche utilisant la méthode MDA, permet de définir l'architecture et de générer un code standard en description logicielle voire matérielle en utilisant UML2.1. Parmi les avantages de cette approche par rapport à l'existant (Marte, Gaspard 2, MoPCOM,..), elle présente un méta-modèle inspiré de l'architecture et la fonctionnalité d'un Composant Virtuel Réutilisable, définie comme un point de départ commun entre l'ensemble des développeurs de ce type de composant. Cette approche permet aussi de définir les besoins fonctionnels avec une validation et vérification du type de composant généré (Soft, Firm, Hard), comme le nécessitent nos composants IP.

Cette approche est indépendante des outils de modélisation, de synthèse et de vérification, ce qui procure une certaine interopérabilité du composant généré. Par ailleurs, elle nous a permis de faire les observations suivantes :

- Une démarche unique de développement des IPs en utilisant un métamodèle en UML rassemblant les deux types d'architectures matérielle et logicielle des IPs ;
- Une élimination des obscurités en rendant le développement des IPs clair et facile par le biais du langage de modélisation UML et ses profils ;
- Une rapidité dans le développement par l'usage des outils générant du code automatiquement, ce qui va influer sur la durée de développement des SoC et la contrainte du Time To Market.
- Obtention de tous les types d'IP en utilisant seulement deux outils : de la spécification de l'IPs jusqu'à la fonte. La figure 15 illustre les différents niveaux.

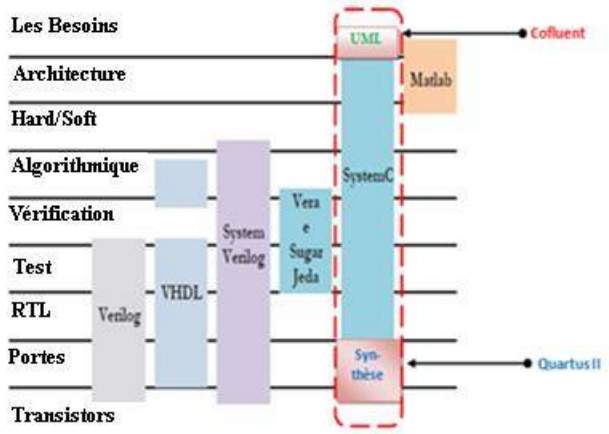

Figure 16 : Les différents niveaux parcourus par notre approche

Cependant, cette approche ne prend pas en charge la définition des besoins non fonctionnels (Non-Functional Properties), point fort de Marte, ni elle est spécifique à un domaine donné, ce qui conduit à un manque de détails du domaine désiré (cas de Gaspard pour le Traitement Intensif du signal).

7. CONCLUSION

Dans ce papier, il est proposé une ébauche comme guide de développement des IPs, recommandée par le VSIA afin de réduire l'écart entre la production des systèmes et l'évolution technologique qui n'a cessé de s'amplifier ces dernières décades. Cette ébauche, outre sa clarté et son interopérabilité, offre la possibilité de créer des composants de bonne qualité, fiables et faciles à être mis en œuvre dans un système. Ainsi, notre approche de développement des IPs, peut être particulièrement perçue comme complète et pratique. Elle consiste à partir d'une spécification axée sur notre métamodèle UML dans un premier temps, à générer automatiquement du code SystemC par l'outil Cofluent dans un second temps et enfin à obtenir une suite de synthèses de modèles produits par Quartus II grâce à sa très riche librairie de composants.
L'expérimentation a été menée avec succès sur une UAL paramétrée.
Nos travaux futurs porteront sur l'enrichissement du métamodèle, et l'introduction de Marte dans notre système afin de créer notre propre outil, qui prend en charge tout le flot de développement de la spécification jusqu'au placement/routage.

REFERENCES

[1] Martin E., Baganne A. & Casseau E., 2000. MILPAT Project Development Methodology for intellectuals properties (IP's) for Telecom Applications: Progress Report 1.1 Behavioral Level: HLS tools and IPs, Lester University of South Brittany, France, 43p.

[2] Savaton G., Casseau E. &Martin E., 2006. Design of a flexible 2-D discrete wavelet transform IP core for JPEG2000 image coding in embedded imaging systems. *Signal Processing*, Vol.8 (7), pp. 1375-1399.

[3]VSI Alliance : http://www.vsi.org

[4]VSI Alliance, 1997.Architecture Document – Version 1.0. Rapport technique.

[5] Booch G., Rumbaugh J. and Jacobson I., 2005.The Unified Modeling Language user guide, 2nd Edition. Addison –Wesley. 496 p.

[6]Object Management Group,.2004.UML 2.0 Testing Profile Specification v2.0.

[7] Object Management Group,.2009.UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms.

[8] Object Management Group, 2005.UML Profile for Schedulability, Performance, and Time (SPT) Specification, v1.1.

[9] Object Management Group., 2005.OMG Systems Modeling Language Specification 1.1.

[10]Object Management Group,.2006.UML Profile for System on a Chip (SoC) Specification, v1.0.1.

[11] Object Management Group,.2009.A UML Profile for MARTE.

[12] Riccobene E., Scandura P., Rosti A., and S. Bocchino, 2005. A SOC Design Methodology Involving a UML2.0 Profile for SystemC., Proceedings of DATE'05, The Design, Automation and Test in Europe Conference and Exhibition, Germany. Vol.2, pp. 704-709.

[13] IEEE 1666 Standard,.2001. SystemC Language Reference Manual, Open SystemC Initiative, [Online] Available: http://www.accellera.org/

[14] Chen Rui,.2011.Synthesizable Systemc To Vhdl Compiler Design. Electronic Theses, The Florida State University, USA. 54p.

[15] Vanderperren Y., Mueller W., Da He Mischkalla, F., & Dehaene W., 2012. UML for Electronic Systems Design : A Code Generation Perspective. Chapter In: *Design Technology for Heterogeneous Embedded Systems*, Springer Science Business Media B.V., pp. 13-39

[16] Mura M., Paolieri M.2007. SC2: State charts to SystemC: Automatic executable models generation. proceedings of Forum on specification & Design Languages FDL07, Spain . Springer *Science+Business Media B.V.*, Vol. 10, pp. 227-239.

[17] Boutekkouk F.,.2010.Automatic SystemC Code Generation from UML Models at Early Stages of Systems on Chip Design, *International Journal of Computer Applications*, Vol. 8, pp. 10-17.

[18] Gajski D.D., Kuhn R., 1983.Guest Editors' Introduction: New VLSI Tools, *IEEE Computer*, vol. 16 (12), pp. 11-14.

[19] Hein C., Carpenter T., Kalutkiemicz P., Madisetti V. , 2000. RASSP VHDL Modeling Terminologyand Taxonomy. Proceeding of 2nd Annual RASSP Conference, pp. 273-281.

[20] Keating M., Bricaud P., 2002. Reuse Methodology manual for System-on-chip designs, Kluwer Academic Publishers, 3th editions, 312p.

[21]Bernard L., 1999. Conception des blocs réutilisables : Réflexion sur la méthodologie, thèse de doctorat en Informatique de l'Institut national polytechnique de Grenoble-France, 156p.

[22] Savaton G., Casseau E., Martin E., Lambert C. - Nebout,.2004. Conception d'un composant matériel réutilisable flexible pour la transformation en ondelettes 2D, I. *Revues : Traitement de Signal*, Vol.21 (2), pp. 157-178.

[23] Le Gal B. ,. 2005. Contribution à la prise en compte des contraintes des applications TDSI dans la synthèse de haut niveau, Thèse de doctorat en informatique , Université de Bretagne Sud, Lorient, France.202p.

[24] Damaševičius R., Štuikys V., 2005. Soft IP Customization Models Based on High-Level Abstractions. *Information Technology and Control*, Vol.34 (2), pp. 125, 134.

[25] Damaševicius R., Štuikys V., .2002. High-Level Design of Soft IP using C++ and SystemC:

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.2154 .

[26] Shimizu N., Ikura M., Wiriya W., & Chivapreecha S., .2009. A new logic circuit design methodology with UMLl , Proceeding of ITC-CSCC'09, Korea ,62-65.

[27] Vidal J., de Lamotte F., Gogniat G., Diguet J.- P., & Soulard P., 2009. IP reuse in an MDA MPSoPC co-design approach, Procedding of, International Conference on Microelectronics ICM'09, Morocco,pp. 256-259.

[28] Piel E., Atitallah R.B., Marquet P., Meftali S., Niar S., Etien A., Dekeyser J.L., Boulet P., 2008. Gaspard2: from MARTE to SystemC Simulation, Proceedings of the DATE'08 workshop on Modeling and Analyzis of Real-Time and Embedded Systems with the MARTE UML profile DATE'08, USA, 26-31.

[29] Vidal J., de Lamotte F., Gogniat G., Soulard P., Diguet J.P., 2009. A Code-design approach for Embedded System Modeling and code generation with UML and MARTE, in: Proceedings of Design Automation and Test in Europe, (DATE'09), Germany, 6p.

[30] Boudour, R., Kimour, M.T.,.2006. From design specification to SystemC, Journal of Computer Science. Vol.2 (2), 201–204.

[31] Aref I., El Gaid K., 2013. System Level Modeling of a Digital Communication Channel Based on SystemC, International Journal of Future Computer and Communication, Vol. 2 (3), pp. 210-214.

[32] Silveira G.S. , Brito A.V. , de A. Oliveira H.F. , and Elmar U. K. Melcher,.2012. Open SystemC Simulator with Support for Power Gating Design Corporation *International Journal of Reconfigurable Computing*. Hindawi Publishing Special issue, Vol. 9, 12p.

[33] D.D. Gajski, A.C. Wu, V. Chaiyakul, S. Mori, T. Nukiyamaet, P. Bricaud, 2000.Essential Issues for IP Reuse. Proceeding of Asia for and South Pacific Design Automation Conference (ASPDAC) – Embedded Tutorial , USA, pp. 37-42.

[34] Cofluent Studio http://www.cofluentdesign.com

[35] ALTERA: http://www.altera.com

[36] Quartus II,.May 2013. Software and Device SupportRelease Notes Version 13.1: http://www.altera.com/literature/rn/rn_qts_dev support.pdf