J3eA - 4 (2005)
DOI : 10.1051/bib-j3ea:2005001

Initiation à la conception de circuit numérique avec un langage de modélisation système : *SystemC*

JEGO Christophe, LE MASSON Jérôme & PIRIOU Erwan

GET/ENST Bretagne
Département électronique, CNRS TAMCIC UMR 2872
email : prénom.nom@enst-bretagne.fr

Résumé

L'ENST Bretagne propose dans le cadre de sa formation d'ingénieur, d'approfondir certains domaines d'enseignement tels que la conception et l'intégration de systèmes de télécommunications. Lors de cette spécialisation, les étudiants participent à un module d'initiation à un langage dédié à la modélisation de systèmes hétérogènes. En effet, les systèmes hétérogènes nécessitent l'utilisation d'un flot de conception mixte logiciel/matériel reposant sur un langage système commun. Actuellement, le langage SystemC est le langage de modélisation système qui est le plus couramment employé. Dans ce contexte, Synopsys a commercialisé un environnement de synthèse reposant sur SystemC pour les descriptions matérielles. L'objectif du module intitulé « conception de systèmes » est de familiariser les étudiants avec un environnement de conception de haut niveau reposant sur le langage SystemC. Dans un premier temps, une initiation expérimentale au langage est organisée. Puis, les élèves suivent un projet d'intégration d'une transformée de Fourier rapide favorisant l'expérimentation d'un flot de conception matériel de haut niveau.

Mots clés

conception de circuit numérique, SystemC, synthèse architecturale et logique, FFT, FPGA

Introduction

Au cours des années 90, la rapide évolution des technologie CMOS vers le submicronique (< 0.6 μm) puis vers le submicronique profond (< 0.25 μm) a favorisé le développement des langages de description du matériel à un haut niveau (les HDL, Hardware Description Langages). Deux d'entre eux ont alors émergé et sont aujourd'hui couramment utilisés : VHDL et Verilog. Aujourd'hui, face à l'émergence des conceptions de SoC (System-on-Chip), au besoin de développement conjoint logiciel/matériel et à l'explosion des temps consacrés à la vérification d'une conception, la demande en langages d'abstraction plus élevés que VHDL et Verilog se fait sentir. Une des voies principales explorée à l'heure actuelle, est l'adaptation du C/C++ à la description matérielle. Ainsi, le standard SystemC 2.0 tout en restant compatible avec le C ANSI, a été adapté à la description matérielle et à la co-vérification pour permettre la conception de systèmes sur puce. Les spécificités d'une description matérielle comme la concurrence ou l'aspect temporel, sont implémentées par des classes C++ (modules, ports, interfaces, canaux et processus) [1].

Un des objectifs principaux du langage SystemC est de permettre la modélisation de systèmes pouvant être intégrés dans des processeurs, des circuits spécifiques (ASIC ou FPGA) ou des SoCs composés de cœurs de processeurs et de modules spécifiques. En fait, ce langage offre la possibilité de spécifier une application à différents niveaux d'abstraction allant du niveau fonctionnel au niveau RTL (*Register Tranfert Level*). L'intérêt principal de ce langage est donc qu'il peut être employé comme langage commun par les concepteurs travaillant au niveau système, sur les processeurs et/ou sur les circuits spécifiques. Pour se faire, le concept de description hiérarchique bien connu dans les langages de modélisation de blocs matériels, existe avec la notion de module en tant que classe C++ dans le langage SystemC.

Dans un premier temps, nous présentons dans ce papier la formation d'ingénieur de l'ENST Bretagne et l'option de spécialisation CIST où est introduite l'initiation au langage SystemC. Puis, nous détaillons l'environnement de conception matériel de haut niveau que nous utilisons. Enfin, la dernière partie concerne le projet d'expérimentation du flot de conception reposant sur le langage SystemC.

I L’option CIST (Conception et Intégration des Systèmes de Télécommmunications) à l’ENST Bretagne

L'Ecole Nationale Supérieure des Télécommunications de Bretagne (ENST Bretagne) offre un large programme de formations (Ingénieurs, Mastères, MsC, DEA, Thèses, Formation Continue) dans les domaines liés aux technologies de l'information. Comme nous allons le voir, au cours de la formation d’ingénieur, les élèves de l’ENST Bretagne ont la possibilité d’approfondir certains domaines d’enseignement tels que la conception et l’intégration de systèmes de télécommunications.

I-1 Organisation de la formation d’ingénieur ENST Bretagne

L’ENST Bretagne a vocation à former des ingénieurs généralistes se destinant aux nombreux domaines composant les technologies de l’information. Le parcours d'un élève ingénieur à l'ENST Bretagne est organisé en 6 phases, d'une durée d'un semestre, désignées par S1, S2, S3, S4…:

S1 est un semestre de tronc commun.

S2, S3, S4 sont des phases durant lesquelles les élèves construisent leur profil de formation, en choisissant certains domaines d'enseignement en majeure et d'autres en mineure.

S5 est un semestre d'option.

S6 est un semestre de stage industriel.

A ces séquences s'ajoutent une formation en langue, et des inter-semestres dans lesquels sont prévues diverses activités (contrôles, préparation des projets du semestre suivant, semaines intensives de langues pour l'inter-semestre d'hiver…), ou des stages (inter-semestre d'été). Cette structure modulaire autorise des possibilités de parcours variés (alternance, semestre à l'étranger…). Enfin, la formation par projets est l’un des axes majeurs de la pédagogie de l’ENST Bretagne [2]. Ainsi, les élèves ingénieurs sont amenés à effectuer au cours des trois années différents projets (projet d’algorithmique et de programmation, projet expérimental, projet ingénieur et projet d’entreprendre). L’objectif est de confronter les élèves à des problèmes concrets pour favoriser l’apprentissage et l’expérimentation de nouvelles connaissances (langages, méthodes, outils…).

I-2 L’option CIST

Au cours de la formation d’ingénieur (semestre 5), les élèves de l’ENST Bretagne ont la possibilité d’approfondir certains domaines d’enseignement tels que la conception et l’intégration de systèmes de télécommunications. Dans une première partie de l’option, les élèves acquièrent une culture approfondie sur les architectures et l’organisation des systèmes de télécommunications. Ainsi, des compléments d’information sur l’électronique, l’électromagnétisme et les communications numériques permettent d’approfondir les fonctions des systèmes de télécommunications.

Des enseignements spécifiques à la conception et l’intégration numérique sont aussi proposés. Ces enseignements s’appuient sur un projet d’intégration numérique ciblant un ASIC ou un FPGA. Toutes les étapes allant de la définition à la réalisation du circuit sont abordées au cours de ce projet. En particulier, le langage de description VHDL est utilisé pour les développements au niveau RTL à l’issue d’une découpe architecturale. Cette étape permet une utilisation concrète d’un langage étudié au cours de l’option.

Des présentations d’enseignants et d’industriels complètent les enseignements spécifiques de l’option. Dans ce cadre, un module intitulé « conception de systèmes » a pour objectif de sensibiliser les élèves à des méthodes et à un flot de conception de haut niveau. Ainsi, une initiation à l’intégration de systèmes hétérogènes est proposée. Ces systèmes nécessitent l'utilisation d'un flot de conception mixte logiciel/matériel (CoDesign). L’apprentissage du langage SystemC dans ce contexte permet d’acquérir une première expérience de conception conjointe logicielle/matérielle.

II Un environnement de conception matériel de haut niveau reposant sur le langage SystemC

L'un des inconvénients majeurs des flots de conception classiques est la multitude des langages utilisés pour la modélisation d'un système et l'intégration des blocs logiciels et/ou matériels le constituant. En effet, des transcriptions manuelles sont nécessaires et celles-ci introduisent des erreurs. De nombreux travaux de recherche ont été menés depuis une dizaine d'année pour faire face à la demande en langages d'abstraction plus élevés que VHDL et Verilog. Deux voies principales sont explorées à l'heure actuelle : d'une part, l'adaptation du C/C++ à la description matérielle et à la vérification, et d'autre part, l'extension des langages HDL vers la programmation au niveau système. Il existe aussi une troisième solution avec la définition d'un nouveau langage adapté aux nouvelles exigences. Cependant, les travaux en cours n'ont pas encore abouti à des outils commerciaux. Le langage SystemC, qui est une évolution du C/C++, a pris une certaine avance dans ce domaine.

L'objectif de SystemC est de retenir un seul langage d'un bout à l'autre du flot de conception. En particulier dans le cas de systèmes complexes, la modélisation passe par une succession de modèles comportementaux qui sont pour les parties matériels :

- ✓ ***UTF (UnTimed Functional)*** : s'applique à l'interface et à la fonctionnalité d'un modèle. Le modèle ne comporte aucune notion de durée d'exécution, mais seulement un ordre éventuel dans l'exécution des événements. Chaque événement s'exécute en un temps nul. Seul compte l'ordonancement des événements.
- ✓ ***TF (Time Functional)*** : s'applique à l'interface et à la fonctionnalité d'un modèle. Le modèle comporte des notion de durée (temps d'exécution des processus, latence, temps de propagation, ...)
- ✓ ***BCA (Bus Cycle Accurate)*** : s'applique à l'interface d'un modèle. Il signifie que la modélisation des transactions sur l'interface est correcte au cycle près. Un modèle BCA n'apporte aucune information sur les bits (signaux) de l'interface.
- ✓ ***CABA (Cycle Accurate Bit Accurate)*** : s'applique à l'interface d'un modèle et à la fonctionnalité d'un modèle. Il signifie que la modélisation des transactions sur l'interface est BCA et qu'elle porte aussi sur les signaux de l'interface. La modélisation est précise au bit (fil) près.
- ✓ ***RTL (Register Transfert Level)*** : s'applique à l'interface et à la fonctionnalité d'un modèle matériel. Chaque bit, chaque cycle, chaque registre du système est modélisé.

A partir du modèle comportemental BCA, il est possible d'utiliser un outil de synthèse d'architecture. La synthèse d'architecture est un processus qui explore l'espace des solutions architecturales possibles pour trouver l'architecture satisfaisant au mieux un ensemble de contraintes telles que la cadence, la surface ou la consommation. Ce processus permet la génération d'un modèle RTL à partir d'un modèle comportemental de type BCA. L'étape suivante est la synthèse logique dont l'objectif est de transformer une description RTL en une description logique (netlist) composée d'un réseau de portes logiques. Cette synthèse s'accompagne d'un ensemble d'optimisations et s'effectue à partir d'une bibliothèque technologique composée de modèles de portes logiques et de bascules. Il est également possible de raffiner manuellement le modèle comportemental jusqu'à l'obtention d'une description RTL de l'architecture (cas le plus fréquent actuellement). Cependant à ce niveau du flot de conception matériel, les possibilités (généricité, souplesse d'utilisation, temps de conception et exploration de l'espace des solutions architecturales) sont bien moins importantes.

II-1 L'initiation au langage SystemC

L'initiation au langage SystemC s'effectue de manière expérimentale. Cette initiation est précédée dans le cursus par un module de formation sur le langage VHDL. Les élèves sont donc familiarisés aux concepts et aux spécificités d'un langage de description des circuits et des systèmes numériques en abordant l'initiation. C'est pourquoi un cours de seulement une heure trente minutes présente dans un premier temps les notions spécifiques du langage : module, synchronisation, canal, interface et processus. Durant ce cours, des comparaisons avec le langage VHDL mettent en évidence l'intérêt

d'un langage système tel que SystemC. Les trois types de processus (SC_METHOD, SC_THREAD et SC_CTHREAD) sont particulièrement détaillés en SystemC [3]. Si les trois types peuvent être utilisés pour la simulation, seuls les types de processus SC_METHOD et SC_CTHREAD sont synthétisables. Ainsi, un processus SC_METHOD correspond à un processus combinatoire accepté par les outils de synthèse logique. Un processus SC_CTHREAD correspond quant à lui à un processus séquentiel synchrone accepté par des outils de synthèse d'architecture comme celui proposé par Synopsys [4]. Dans un second temps, la structure du langage SystemC (port, signal, horloge, types de données) est explicitée. Une séance d'expérimentation d'une durée de une heure trente minutes est organisée juste après le cours. L'objectif de cette séance est de se familiariser avec le langage et ses caractéristiques. Les élèves modélisent des fonctions volontairement simples aboutissant à la description hiérarchique d'un additionneur 8 bits. Pour se faire, ils décrivent successivement une porte NAND2, une porte XOR2, un additionneur 1 et enfin l'additionneur 8 bits comme le montre la figure suivante :

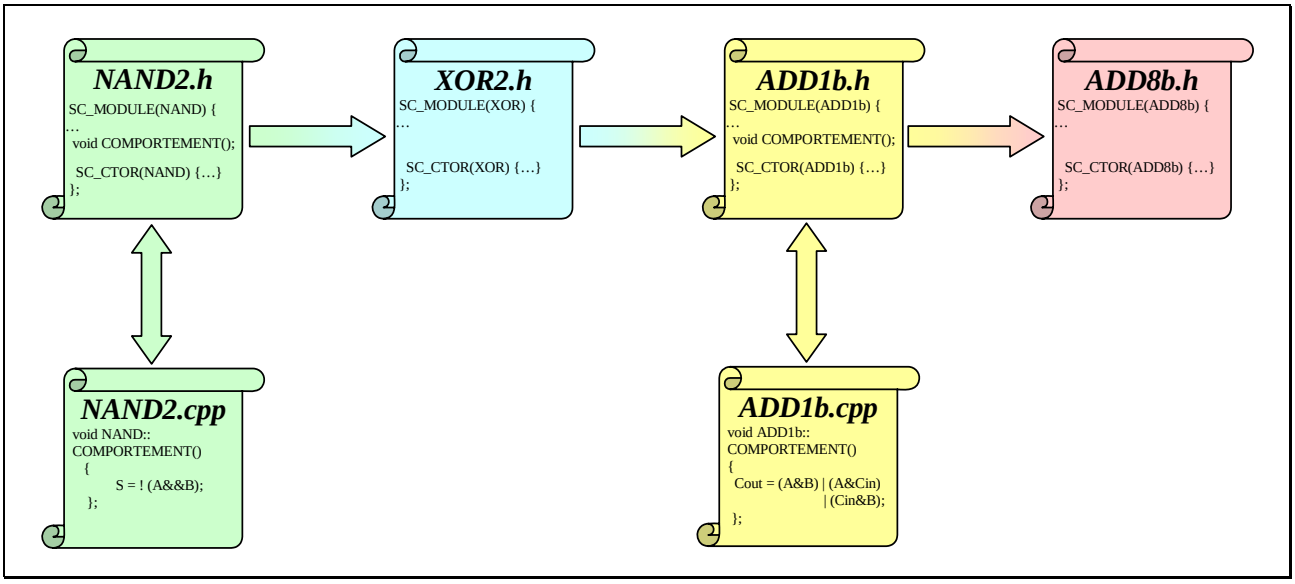


Figure 1 : modélisation d'un additionneur 8 bits en SystemC

II-2 L'environnement de conception matériel de haut niveau

Le langage SystemC est une extension du langage C ANSI. C'est pourquoi, son utilisation en simulation ne nécessite qu'un compilateur C/C++ tel que *gcc* auquel est ajouté la bibliothèque *systemc.h* contenant les extensions spécifiques à ce langage. Cependant, l'exploitation du langage SystemC pour la conception de circuit matériel implique l'utilisation d'un outil de synthèse l'acceptant comme langage de description au même titre que les langages VHDL et Verilog. Actuellement, seul SystemC Compiler de Synopys offre cette possibilité [4].

L'environnement de conception que nous avons utilisé est présenté figure 2. La modélisation des applications est faite dans le langage SystemC sous SystemC_Win (outil disponible gratuitement sur internet à l'adresse *http://www.geocities.com/systemc_win/*). Il est à noter que l'outil ModelSim de la société Mentor Graphics permet la simulation d'une application décrite en SystemC dans sa version 6 disponible depuis septembre 2004. Nous envisageons de remplacer SystemC_Win par ModelSim dans les prochaines sessions. Un raffinement virgule flottante/fixe de la description SystemC est effectué dans un deuxième temps. Puis, une exploration architecturale et la synthèse logique sont réalisées sous SystemC Compiler. Enfin, les solutions architecturales retenues sont intégrées sur un FPGA de la famille Stratix à l'aide de l'environnement Quartus II de la société Altera.

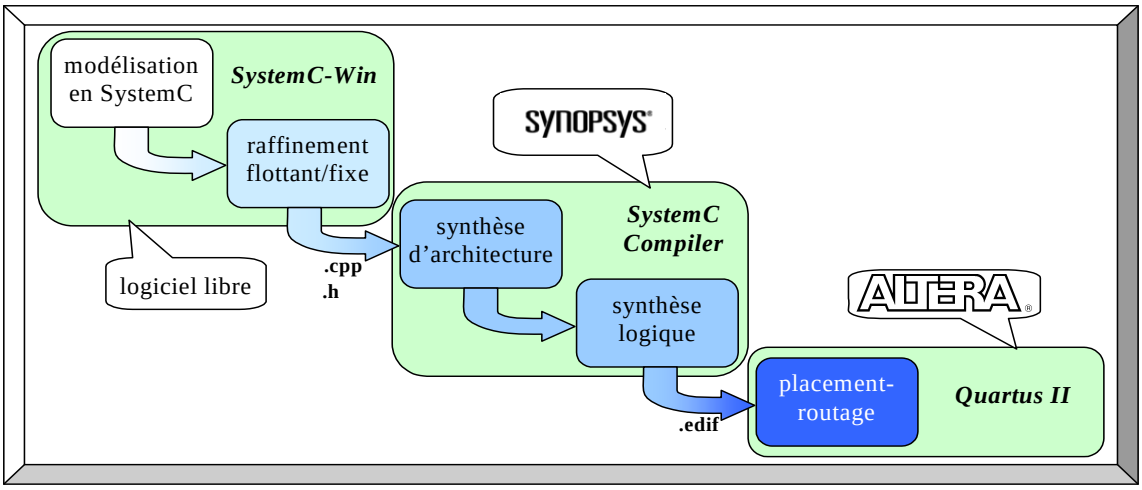


Figure 2 : environnement de conception matériel de haut niveau

III Expérimentation du flot de conception matériel de haut niveau reposant sur le langage SystemC

L'objectif du projet final du module intitulé « conception de système » est d'expérimenter ce qui peut être considéré comme un flot de conception matériel de haut niveau. L'application retenue est la transformée de Fourier rapide. La durée de ce projet est de 9 heures.

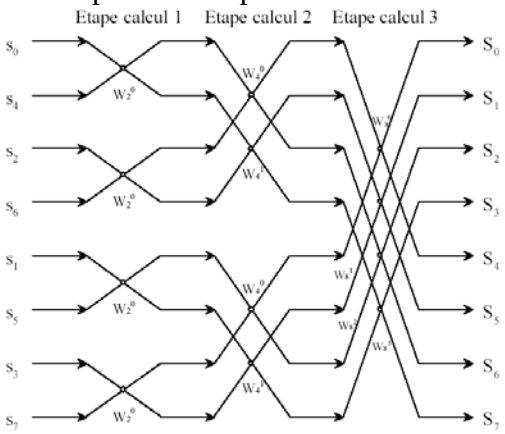
III-1 Application cible : la transformée de Fourier rapide FFT

La transformée de Fourier rapide (Fast Fourier Transform : FFT en anglais) est un algorithme qui réduit la complexité et donc facilite l'implémentation de la transformée de Fourier discrète. Cette opération mathématique est l'une des plus courantes dans le traitement du signal et de l'image. Elle permet de calculer la réponse fréquentielle d'un système à partir de sa réponse impulsionnelle. La FFT est une transformation présente dans de nombreux blocs constituant les systèmes de communications numériques comme la compression, la modulation multi-porteuses (OFDM), l'égalisation… Au niveau calcul, la FFT est une répétition du papillon de Cooley-Tuckey [5]. Si *x(n)* est un échantillon d'entrée, alors *X(k)*, l'échantillon de sortie, est donnée par :

$$X(k)=\sum_{n=0}^{N-1} x(n)\times W_N^{nk} \quad \text{avec} \quad W_N^{nk}=e^{-2\pi j\frac{nk}{N}}$$

où N est le nombre d'échantillons nécessaires pour calculer *X(k)*.

L'opération papillon est particulièrement adaptée à l'algorithme FFT comme nous le montre l'exemple suivant pour N = 8 et un nombre d'étages k = 3 tel que k = $\log_2 N$ [6].



$$\begin{aligned}
S_0 &= s_0 + s_4\cdot W_2^{0} + W_4^{0}\cdot(s_2 + s_6\cdot W_2^{0}) + W_8^{0}\cdot[s_1 + s_5\cdot W_2^{0} + W_4^{0}\cdot(s_3 + s_7\cdot W_2^{0})]\\
S_1 &= s_0 - s_4\cdot W_2^{0} + W_4^{1}\cdot(s_2 - s_6\cdot W_2^{0}) + W_8^{1}\cdot[s_1 - s_5\cdot W_2^{0} + W_4^{1}\cdot(s_3 - s_7\cdot W_2^{0})]\\
S_2 &= s_0 + s_4\cdot W_2^{0} - W_4^{0}\cdot(s_2 + s_6\cdot W_2^{0}) + W_8^{2}\cdot[s_1 + s_5\cdot W_2^{0} - W_4^{0}\cdot(s_3 + s_7\cdot W_2^{0})]\\
S_3 &= s_0 - s_4\cdot W_2^{0} - W_4^{1}\cdot(s_2 - s_6\cdot W_2^{0}) + W_8^{3}\cdot[s_1 - s_5\cdot W_2^{0} - W_4^{1}\cdot(s_3 - s_7\cdot W_2^{0})]\\
S_4 &= s_0 + s_4\cdot W_2^{0} + W_4^{0}\cdot(s_2 + s_6\cdot W_2^{0}) - W_8^{0}\cdot[s_1 + s_5\cdot W_2^{0} + W_4^{0}\cdot(s_3 + s_7\cdot W_2^{0})]\\
S_5 &= s_0 - s_4\cdot W_2^{0} + W_4^{1}\cdot(s_2 - s_6\cdot W_2^{0}) - W_8^{1}\cdot[s_1 - s_5\cdot W_2^{0} + W_4^{1}\cdot(s_3 - s_7\cdot W_2^{0})]\\
S_6 &= s_0 + s_4\cdot W_2^{0} - W_4^{0}\cdot(s_2 + s_6\cdot W_2^{0}) - W_8^{2}\cdot[s_1 + s_5\cdot W_2^{0} - W_4^{0}\cdot(s_3 + s_7\cdot W_2^{0})]\\
S_7 &= s_0 - s_4\cdot W_2^{0} - W_4^{1}\cdot(s_2 - s_6\cdot W_2^{0}) - W_8^{3}\cdot[s_1 - s_5\cdot W_2^{0} - W_4^{1}\cdot(s_3 - s_7\cdot W_2^{0})]
\end{aligned}$$

Figure 3 : exemple d'application du papillon à la FFT 8 points.

III-2 Raffinement matériel des modèles comportementaux de la FFT 8 points

L'expérimentation de ce flot de conception est effectuée à travers l'étude d'une FFT. Dans notre projet, la transformée permet de calculer les 8 fréquences complexes issues de 8 échantillons complexes. Le développement de cette fonction doit être réalisé en utilisant un banc de test spécifié en SystemC. Nous allons dans la suite de ce paragraphe détaillé le raffinement matériel des modèles comportementaux de la FFT 8 points.

III-2-a Spécification en SystemC de la FFT 8 points en virgule flottante par un modèle TF (Time Functional)

Au niveau de modélisation TF, la description de la FFT comporte une notion de temps qui est l'horloge. Le développement de la FFT doit être réalisé en respectant la modélisation suivante :

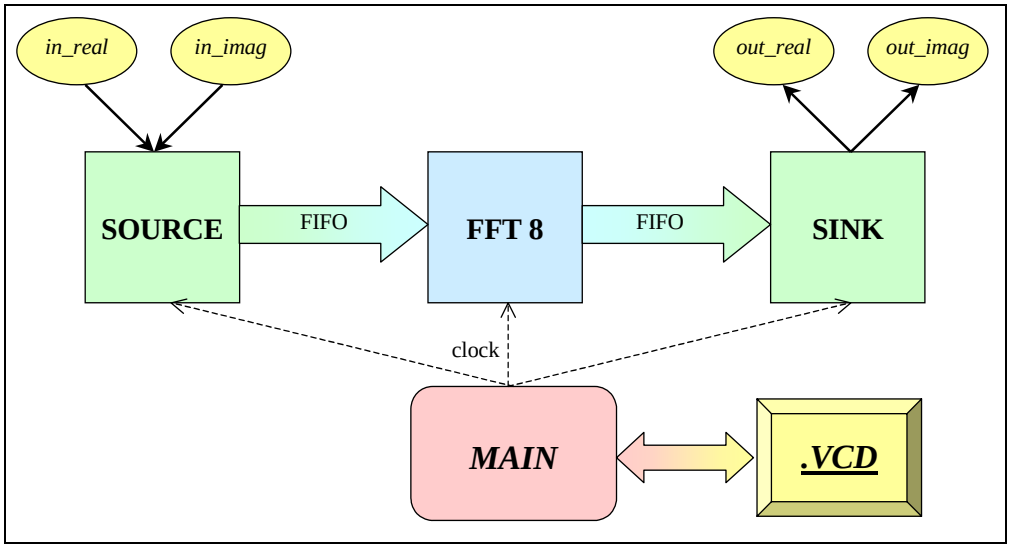


Figure 4 : schéma bloc du banc de test du modèle TF de la FFT 8 points

Cet environnement est constitué de trois composants SOURCE, FFT8 et SINK. Le rôle respectif de ces trois composants est la génération des échantillons, la fonction FFT et la récupération des fréquences. Le composant SOURCE lit les échantillons dans deux fichiers (un pour la partie réelle et un autre pour la partie imaginaire). Le composant SINK écrit les valeurs réelles et imaginaires des fréquences dans deux fichiers distincts. La communication entre les composants est assurée à ce niveau de modélisation par deux FIFO. La FIFO entre les composants SOURCE et FFT8 accumule les données provenant de deux fichiers. Lorsque les données des 8 échantillons complexes sont disponibles, le composant FFT8 calcule les 8 fréquences complexes correspondantes. Puis, les données sont transmises au composant SINK à l'aide d'une seconde FIFO.

Durant cette phase préliminaire, il faut définir les trois composants sans la fonctionnalité FFT. L'objectif est de lire les fichiers *in_real* et *in_imag* et de récupérer l'ensemble des données dans les fichiers *out_real* et *out_imag*. L'ensemble est cadencé par une horloge et les données doivent être échangées entre les trois composants en respectant le protocole de communication. En particulier, le comportement du composant FFT8 doit permettre de récupérer les échantillons et de les stocker dans deux vecteurs sample-real[8] et sample_imag[8].

Il faut ensuite décrire le papillon pour des échantillons complexes dans une fonction particulière. Puis, cette fonction sera réutilisée dans la méthode COMPORTEMENT de la classe FFT. Au sujet des coefficients W_N^n, nous pouvons écrire à partir des caractéristiques de la FFT présentées dans le paragraphe III-1 :

$$W_N^{nk} = e^{-2\pi j \frac{nk}{N}} = \cos(2\pi nk/N) - j\sin(2\pi nk/N) = [W_N^n]_r + j[W_N^n]_i$$

Pour N = 8, nous avons par conséquent :

$$W_8^0 = [W_8^0]_r + j[W_8^0]_i = \cos(0) - j\sin(0)$$
$$W_8^1 = [W_8^1]_r + j[W_8^1]_i = \cos(\pi/4) - j\sin(\pi/4)$$
$$W_8^2 = [W_8^2]_r + j[W_8^2]_i = \cos(\pi/2) - j\sin(\pi/2)$$
$$W_8^3 = [W_8^3]_r + j[W_8^3]_i = \cos(3\pi/4) - j\sin(3\pi/4)$$

De plus, nous savons que $W_N^{2nk} = e^{-2\pi j \frac{nk}{\frac{N}{2}}} = W_{\frac{N}{2}}^{nk}$

Nous obtenons donc pour N = 8 : $W_8^0 = W_4^0 = W_2^0$ & $W_8^2 = W_4^1$

Par conséquent, nous disposons des coefficients nécessaires au calcul de la FFT à échantillons complexes.

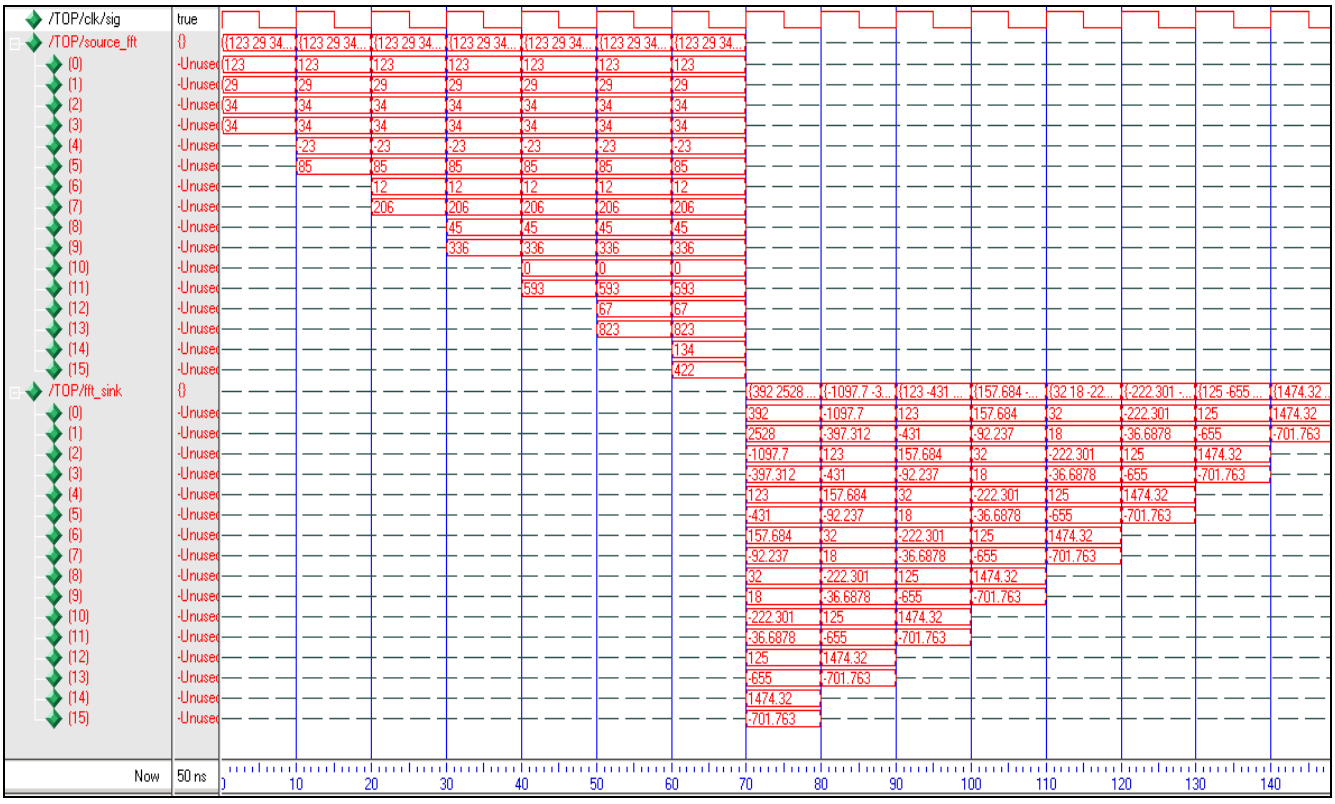


Figure 5 : chronogramme des communications à travers les FIFO pour le modèle TF de la FFT

Le chronogramme de la figure 5 représente la communication à travers les deux FIFO pour le modèle TF de la FFT 8 points. Les 8 échantillons complexes et les 8 fréquences complexes issues de la transformée, sont donnés dans le tableau 1.

échantillons complexes		fréquences complexes	
partie réelle	partie imaginaire	partie réelle	partie imaginaire
123	29	392	2528
34	34	-1097.7	-397.312
-23	85	123	-431
12	206	157.684	-92.237
45	336	32	18
0	593	-222.301	-36.6878
67	823	125	-655
134	422	1474.32	-701.763

Tableau 1 : données échangées à travers les FIFO pour le modèle TF de la FFT

III-2-b Spécification en SystemC de la FFT 8 points en virgule flottante par un modèle BCA (Bus Cycle Accurate)

L'objectif de cette deuxième phase est de raffiner le modèle de communication de la FFT permettant de calculer les 8 fréquences complexes issues de 8 échantillons complexes. En effet, la modélisation des transactions sur l'interface doit être au cycle prêt au niveau BCA. Nous choisissons d'utiliser un protocole de communication « requête/acquittement » pour les communications entre les trois composants. Ce protocole se compose de deux signaux de contrôle avec un bloc qui demande une communication (le maître) et l'autre bloc qui valide la demande (l'esclave). De plus, nous séparons les parties réelles et imaginaires des échantillons et des fréquences (deux bus distincts). La modélisation correspondante est la suivante :

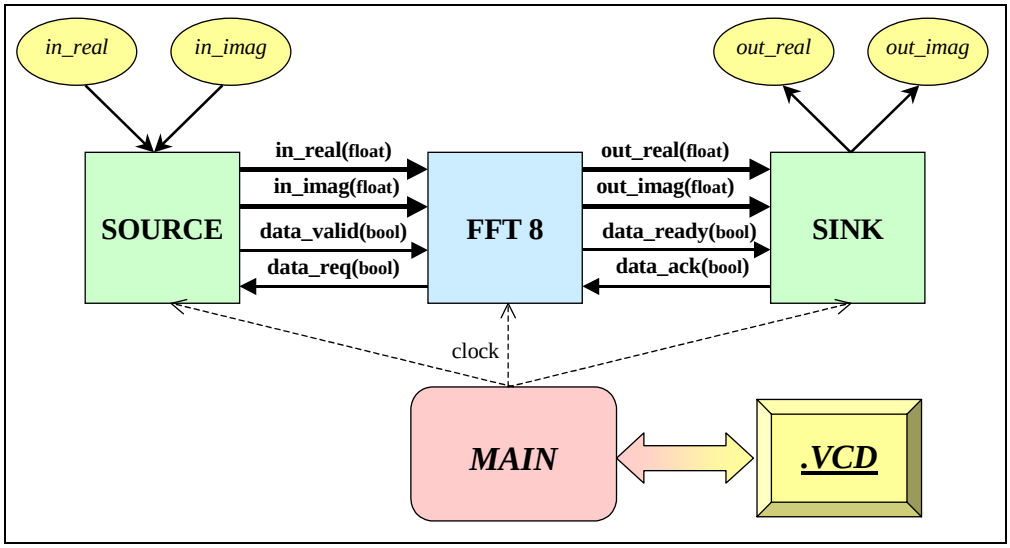


Figure 6 : schéma bloc du banc de test du modèle BCA de la FFT 8 points

Le protocole de communication « requête/acquittement » est le suivant : les signaux data_req et data_valid sont à *false* par défaut. Lors d'une demande de lecture data_req est positionné à *true*. Lorsque le module SOURCE est prêt, il positionne le signal data_valid à *true*. Un échantillon peut alors être lu. Il est lu par le bloc FFT puis data_req est positionné à *false* et le cycle d'horloge suivant est attendu (hypothèse d'exécution : un échantillon lu sur chacun des bus par cycle d'horloge). L'objectif est de lire les 8 échantillons avant de commencer le traitement. Le principe est similaire pour la phase d'écriture en utilisant les signaux data_ready et dat_ack. Dans les deux protocoles de communication, le bloc maître est le composant FFT8.

Le chronogramme de la figure 7 représente le protocole de communication « requête/acquittement » pour le modèle BCA la FFT 8 points. Les 8 échantillons complexes et les 8 fréquences complexes issues de la transformée, sont ceux du tableau 1.

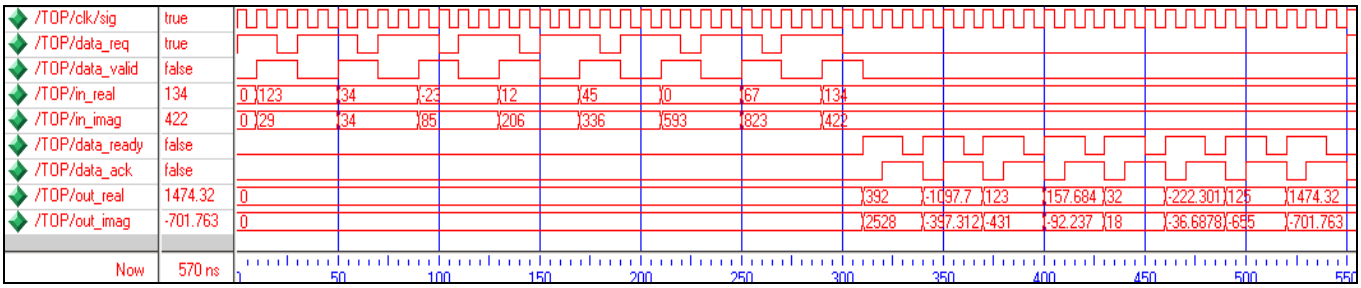


Figure 7 : chronogramme des communications pour le modèle BCA de la FFT

III-2c Spécification en SystemC de la FFT 8 points en virgule fixe par un modèle CABA (Cycle Accurate/ Bit Accurate) :

L'objectif à ce niveau de modélisation est de raffiner la fonctionnalité permettant de calculer les 8 fréquences complexes issues de 8 échantillons complexes. Cette étape correspond au passage flottant vers fixe de l'algorithme de la FFT. En effet, la modélisation des données doit être au bit prêt au niveau CABA. La modélisation correspondante est la suivante :

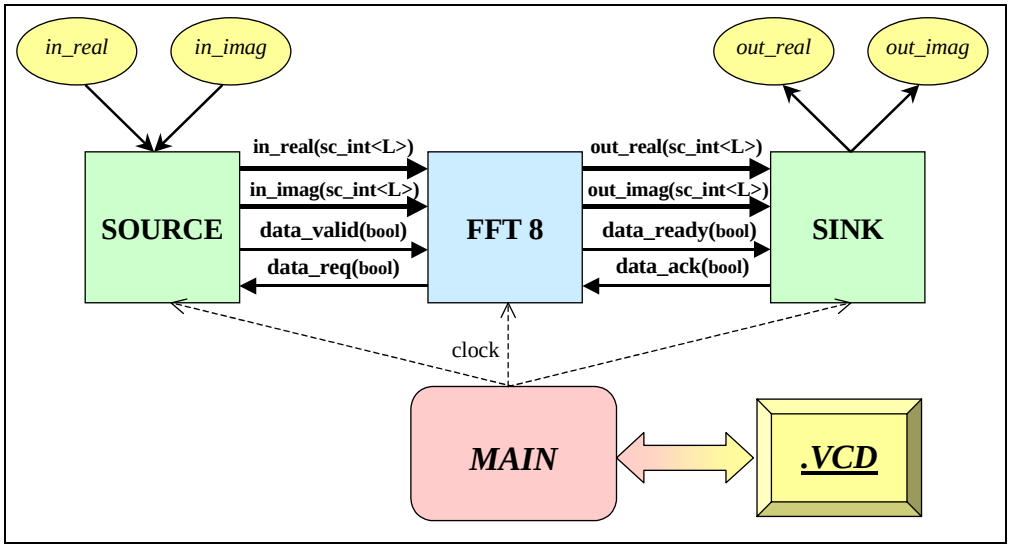


Figure 8 : schéma bloc du banc de test du modèle CABA de la FFT 8 points

Le raffinement de la description FFT en SystemC consiste à convertir toutes les valeurs de type float en valeurs de type sc_int<L> et à adapter le modèle SystemC en conséquence. La simulation permettra de comparer les résultats obtenus pour les deux versions. Le type sc_int<L> désigne un type entier sur L bits comparable à un type bit_vector en VHDL.

Il faut modifier la façon dont sont effectués les calculs de la FFT : les coefficients W doivent être codés sur des entiers 16 bits en complément à 2 au format (11,5), c'est à dire la partie fractionnaire est codée sur 5 bits. Nous effectuons dans le papillon des multiplications de deux mots de 16 bits, le résultat est assigné à un mot de 32 bits. Une troncature est alors nécessaire pour obtenir un vecteur de 16 bits en sortie de l'opération de multiplication. Nous choisissons de retenir les bits 5 à 20 dans le vecteur.

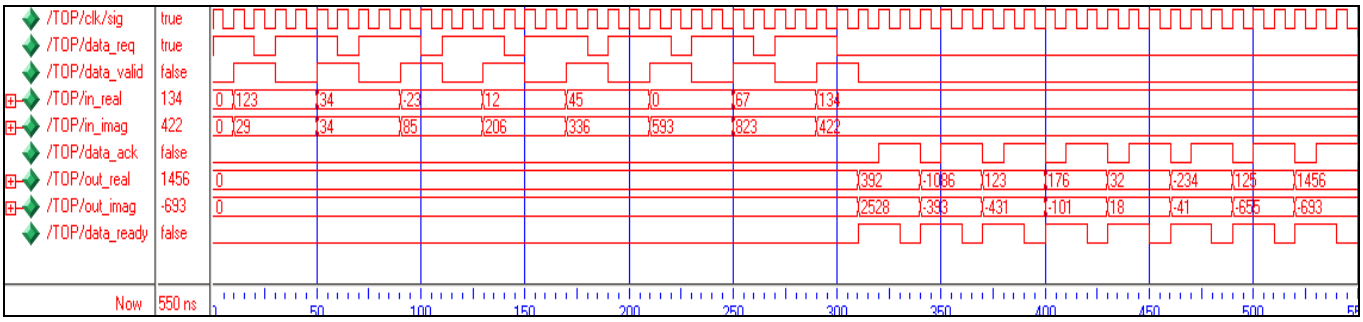


Figure 9 : chronogramme des communications pour le modèle CABA de la FFT

Le chronogramme de la figure 9 représente les résultats de simulation pour le modèle CABA de la FFT 8 points. Les 8 échantillons complexes et les 8 fréquences complexes issues de la transformée, sont donnés dans le tableau 2.

échantillons complexes		fréquences complexes	
partie réelle	partie imaginaire	partie réelle	partie imaginaire
123	29	392	2528
34	34	-1086	-393
-23	85	123	-431
12	206	176	-101
45	336	32	18
0	593	-234	-41
67	823	125	-655
134	422	1456	-693

Tableau 2 : données échangées pour le modèle CABA de la FFT

III-3 Synthèse et intégration de la FFT 8 points

Dans cette dernière partie, le flot de conception pour la synthèse et l'intégration d'une solution architecturale est expérimenté. La description en SystemC du modèle comportemental CABA de la FFT est le modèle sur lequel s'appliquent successivement la synthèse d'architecture puis la synthèse logique.

III-3a Synthèse architecturale et synthèse logique du modèle CABA de la FFT

L'outil utilisé pour la synthèse architecturale est SystemC Compiler de Synopsys. Un script de synthèse est nécessaire pour l'utilisation de cet outil. Ce fichier contient : la technologie cible ASIC ou FPGA, les paramètres pour les options de synthèse et les commandes de synthèse architecturale. L'architecture cible est constituée d'un chemin de données, d'une mémoire et d'un contrôleur qui pilote la partie traitement. Le chemin de données est un circuit du type *mux-registre-mux-opérateur* dans lequel toutes les sorties sont mémorisées. Le processus SC_CTHREAD de la description SystemC de la FFT est synthétisé comme un chemin de données spécifique commandé par une machine d'états. Les étapes de la synthèse architecturale de *SystemC Compiler* sont présentés sur la figure 10. Après la description du modèle comportemental CABA de la FFT, l'utilisateur doit spécifier l'horloge de l'architecture. Il doit également guider la synthèse suivant une des trois orientations possibles : minimisation de la surface de l'architecture, minimisation du temps de traversée de l'architecture ou respect de la latence (nombre de cycle). Puis l'outil effectue les étapes d'ordonnancement des opérations, d'allocation des ressources et des registres. Lors de l'allocation, des ressources de calcul et des registres sont assignés respectivement aux opérations et aux variables. Les variables sont ainsi automatiquement allouées à des registres tandis que les tableaux sont regroupés dans des mémoires.

De plus, l'outil génère la machine d'états (sous la forme d'une machine de *Mealy*) de la partie contrôle de l'application. Enfin, la description RTL de la solution architecturale utilise un format interne à *Synopsys* (.db). Il n'est donc pas possible d'effectuer la synthèse logique avec un outil autre que ceux de Synopsys.

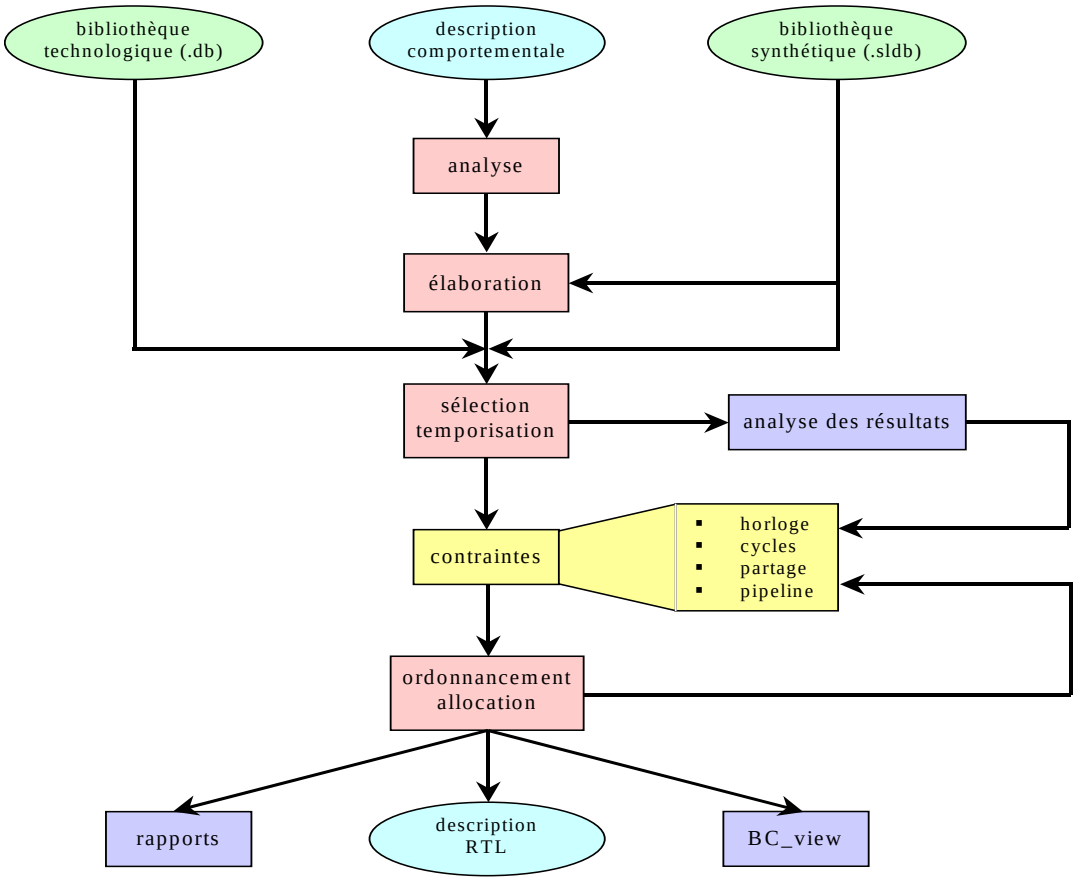


Figure 10 : étapes de la synthèse architecturale de l'outil SystemC Compiler

Plusieurs synthèses ciblant différentes solutions architecturales (compromis débit/complexité) sont effectuées par les étudiants. Pour réaliser ce travail des scripts paramétrables sont fournis. Les caractéristiques de différentes solutions architecturales pour la FFT 8 points sont présentés dans le tableau 3. La période d'horloge est fixée à 30 ns.

FFT 8 points		solution architecturale		
		rapide	compacte	compromis
nombre de cycle d'exécution		15	105	31
fréquence de fonctionnement MHz		2.22	0.31	1.075
nombre d'opérateurs combinatoires	multiplieur 16_16->32	16	3	5
	additionneur 16_16->16	8	3	6
	soustracteur 16_16->16	16	2	6
	comparateur 4_4->1_1	1	1	1
nombre d'opérateurs séquentiels	registre 128 bits	5	5	5
	registre 16 bits	26	30	28
nombre d'états de la FSM		18	108	34

Tableau 3 : caractéristiques de différentes solutions architecturales

Synopsys propose un outil de synthèse logique (transformation de la description RTL en Netlist) dédié aux circuits FPGA : FPGA Compiler II. L'option *set_FPGA* permet de cibler un FPGA lors de la

synthèse d'architecture sous SystemC Compiler. L'utilisation de cette option implique l'emploi de l'outil FPGA Compiler II pour la synthèse logique. Le circuit FPGA cible dans notre projet est un *STRATIX EP1S40F780C5* de chez Altera. La génération de la description au niveau RTL sous SystemC Compiler permet d'obtenir en plus de la description, un fichier de script pour la synthèse logique. Dans le cadre du projet, il est demandé aux étudiants de faire la synthèse logique de la solution architecturale la plus rapide.

III-3b Implantation d'une solution architecturale sur un circuit FPGA

La Netlist de la solution architecturale retenue est ensuite transmise à l'environnement QuartusII d'Altera. L'implémentation consiste alors à définir un projet sous QuartusII utilisant la Netlist au format *edif* et à effectuer le placement/routage sur la cible FPGA Stratix retenue lors des phases de synthèse. Les résultats en terme d'occupation du circuit FPGA sont les suivants :

- ✓ nombre d'éléments logiques utilisés : 7025 / 41250 (17%)
- ✓ nombre de blocs DSP : 6 / 112 (5%)
- ✓ nombre d'accès :70 / 691 (10%)

Le floorplan de la solution architecturale retenue est présenté sur la figure 11. Nous pouvons observer la répartition des logiques éléments et des blocs DSP occupés.

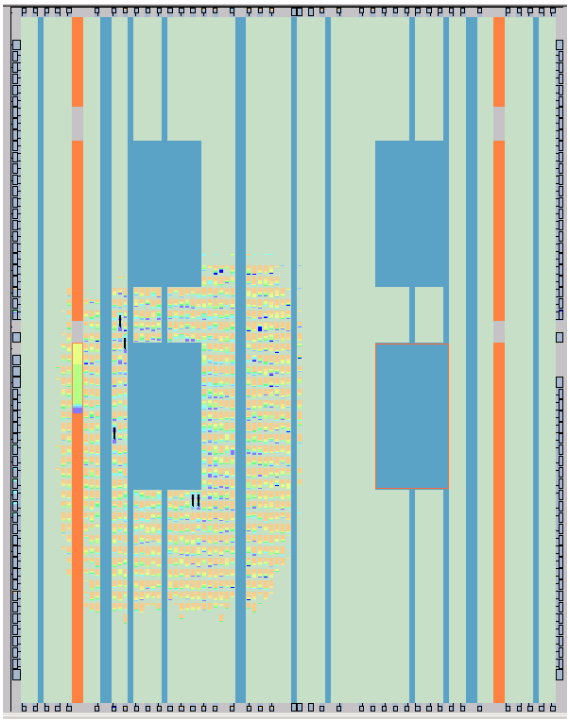


Figure 11 : floorplan de la solution architecturale retenue pour la FFT

III-3c Prototypage de la solution architecturale sur une carte de développement.

Lors de cette ultime étape, le circuit pour la FFT 8 points est implanté sur la plate forme « Nios Development Kit, Stratix Professional Edition » d'Altera (figure 12). Cette carte de prototypage est construite autour d'un FPGA Stratix EP1S40, composé lui-même de plus de 40.000 éléments logiques et de plus de 3 Mbits de mémoire embarqué. Le chargement du FPGA est réalisé via le port JTAG de la carte.

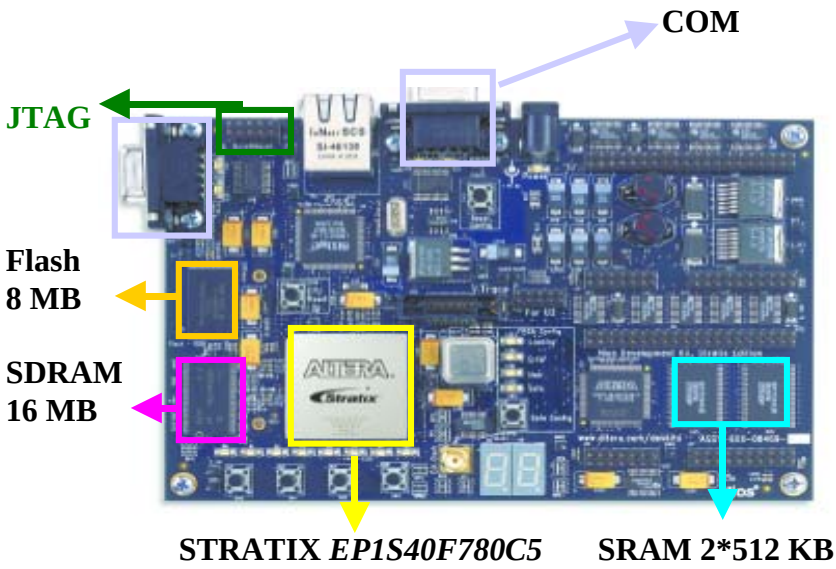


Figure 12 : carte de développement STRATIX de la société Altera

CONCLUSION

Le module d'enseignement présenté dans cet article a pour objectif d'initier les élèves-ingénieurs à un environnement de conception matériel de haut niveau. Pour se faire, une expérimentation d'un langage de modélisation au niveau système est menée dans un premier temps. Puis, un projet d'intégration est réalisé par les étudiants. Ce module intervient dans le cadre d'une spécialisation à la conception et l'intégration de systèmes de télécommunications. Il est précédé par une formation intensive au langage de description matériel VHDL. L'enseignement de ce module permet ainsi de mettre en évidence l'intérêt de la modélisation en SystemC par rapport à une modélisation classique en VHDL (transition plus aisée de l'algorithme vers l'architecture). L'utilisation prochaine d'un simulateur commun pour les descriptions VHDL et SystemC (ModelSim V6) va nous permettre de sensibiliser les étudiants à la co-simulation. Enfin, il est à noter que ce bloc d'enseignement devient essentiel à notre formation car le nombre d'élèves-ingénieurs qui sont amenés à utiliser le langage SystemC au cours de leur stage de fin d'étude et/ou de leur premier emploi est en constante augmentation.

BIBLIOGRAPHIE

[1] OSCI (Open SystemC Initiative) ,*"SystemC 2.0.1 Language Reference Manuel"*, http://www.systemc.org, version 2, 2002.
[2] Groupe de pilotage du projet ingénieur de l'ENST Bretagne, *"VADE-MECUM du projet d'ingénieur 2003-2004 "*, Brest, Janvier 2004.
[3] T. Grötker, S. Liao, G. Martin, S. Swan, *"System Design with SystemC"*, Ed. Kluwer Academic Publishers, 2002.
[4] Synopsys, *"Describing Synthetizable RTL in SystemC™"*, version 1.2, November 2002.
[5] J. W. Cooley & J.W. Tuckey, *"An algorithm for the machine calculation of complex Fourier series"*, Mathematics of computation, pp. 297-301, April, 1965.
[6] M. Kunt, *"Traitement numérique des signaux"*, Ed. Presses Polytechnique Romandes, 1984.