Architecture de wrapper de synchronisation pour environnement de type GALS/LIS

Pierre Bomel, Eric Martin, Emmanuel Boutillon
LESTER, UBS, Lorient, France
{pierre.bomel, emmanuel.boutillon, eric.martin}@univ-ubs.fr

Résumé

Dans ce papier nous présentons notre contribution en terme d'architecture de wrapper de synchronisation à la méthodologie de conception de SoCs fondée sur la théorie des systèmes insensibles à la latence de Carloni et al.. Cette méthodologie 1) favorise la réutilisation intensive d'IP prédéveloppées, 2) segmente les interconnexions inter-IP avec des stations de relais pour briser les chemins critiques inter-IP et 3) rend les IP robustes aux irrégularités des délais de tranferts de données par encapsulation dans des wrappers. Notre contribution consiste à encapsuler les IP dans un nouveau modèle de wrapper dont la vitesse est optimisée, la surface prédictible et la synthétisabilité garantie. Le principal bénéfice de notre approche est de préserver lors de l'encapsulation les performances individuelles des IP afin de ne pas dégrader les performances globales du SoC.

1. Introduction

Les circuits intégrés modernes, nommés systèmes sur puce (SoC), sont la composition de plusieurs sous-systèmes, ou IPs, qui échangent des données entre eux. La taille des SoCs est devenue telle qu'une stratégie efficace et fiable d'interconnexion est nécessaire pour combiner les sous-systèmes de façon à préserver les performances offertes par les technologies de fabrication submicroniques à un coût de conception acceptable [1]. Cette contrainte de communication peut être satisfaite en employant un media de communication suffisamment rapide entre les IPs.

La topologie de type bus est fréquemment employée car peut coûteuse et facile à mettre en œuvre. Elle introduit néanmoins un goulot d'étranglement car 1) un seul IP peut émettre une donnée à un instant donné et 2) la longueur des pistes du bus et la charge capacitive du grand nombre de module connectés au bus introduisent des temps de propagation qui réduisent la fréquence maximale de fonctionnement. Les topologies de bus hiérarchiques contribuent à repousser ces limites en segmentant le bus en plusieurs sous-bus reliés par des ponts : un meilleur parallélisme de communication et une réduction des longueurs des pistes sont ainsi obtenus.

La nature des SoC hautes performances nécessite maintenant des architectures de type micro-réseaux (NoC) dans lesquelles le degré de parallélisme et le débit sont optimisés en fonction des besoins applicatifs [2]. Le découpage en canaux de communication permet de réduire les temps de transferts sur les interconnexions. Les NoCs intègrent des services supplémentaires tels que le routage de paquets, le contrôle de flux, et éventuellement la détection d'erreurs. Ces services sont nécessaires et ont un coût non négligeable en surface silicium.

L'approche système globalement asynchrone et localement synchrone (GALS) préserve la fréquence de fonctionnement maximale par découpage d'un circuit complexe en sous-systèmes localement synchrones et optimisés en fréquence. La communication inter sous-systèmes est assurée par des liaisons point à point asynchrones. L'interface entre sous-systèmes synchrones et réseau asynchrone est un point déterminant dans l'applicabilité de la méthodologie car elle doit être fiable sans pénaliser les fréquences maximales. Une évolution très prometteuse des GALS est la théorie des systèmes insensibles à la latence (LIS) dont le réseau de communication est synchrone. Elle partage avec les NoCs le haut degré de parallélisme potentiel et avec les bus la simplicité de mis en œuvre.

Ce papier est organisé ainsi. En premier, la section 2 présente l'état de l'art concernant la synchronisation des IPs avec le réseau de communication en environnement LIS. La section 3 justifie notre approche, spécifie notre modèle de wrapper et présente des mesures comparatives avec des FSM classiques. La section 4 présente une expé-

rience de conception d'un système de radiocommunications. La section 5 conclut ce papier et présente les perspectives d'extension des travaux.

2 Etat de l'art

Les GALS, introduits par [3], combinent des sous-systèmes synchrones qui communiquent entre eux au travers d'un réseau asynchrone. Ils nécessitent la synthèse de systèmes mixtes synchrones/asynchrones [4]. Ils reposent sur l'utilisation d'horloges suspensibles pour synchroniser le fonctionnement des sous-systèmes avec les asynchronismes des flots de données et économiser de l'énergie lors des périodes d'attente.

La théorie des systèmes insensibles à la latence [5] est une évolution des GALS pour laquelle le réseau de communication n'est plus asynchrone mais pseudo-asynchrone. Le réseau est constitué de liaisons point à point pipelinées par adjonction d'unités de stockage nommées stations de relais. Le but est de stocker temporairement les données lors de leur transit et de supporter le protocole de synchronisation de transfert des données entre IPs et stations de relais. Les temps de transit des données entre sous-systèmes ne sont plus quelconques : ce sont des multiples de l'horloge du réseau de communication. La théorie des LIS permet de réutiliser les outils de synthèse RTL classiques aussi bien pour les IPs que pour le réseau de communication pseudo-asynchrone. L'adjonction de composants asynchrones n'est donc plus nécessaire. Cette théorie garantit de plus que le modèle LIS est fonctionnellement équivalent (à la latence près) à son modèle synchrone [6].

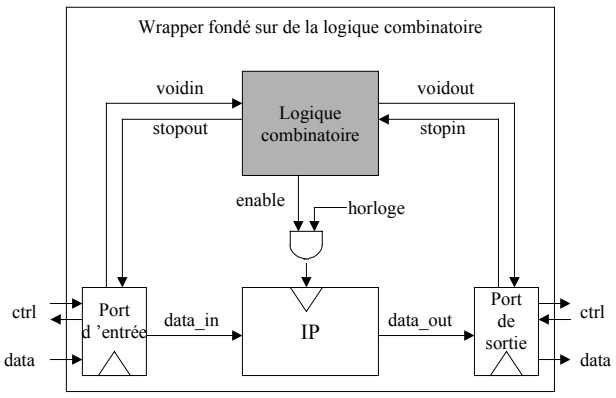

Figure 1 : Modèle de Carloni et al

Les LIS reposent sur le concept de processus patient. Les processus patients [7] sont des IPs synchrones (la perle ou pearl) suspensibles et encapsulés dans un wrapper (la coquille ou shell) dont la fonction est de les rendre insensibles à la latence des entrées-sorties et de piloter leur horloge. La méthodologie de conception est constituée de deux phases. La première consiste à encapsuler les IPs suspensibles avec les wrappers de synchronisation puis de procéder à un placement routage. La deuxième phase consiste à identifier les chemins critiques qui se trouvent sur le réseau de communication initialement synchrone et à insérer des stations de relais. Le réseau devient alors pseudo-asynchrone et peut être à nouveau placé et routé. L'introduction de stations de relais, dont le but est de briser les chemins critiques inter IPs, peut introduire une dégradation des performances du système complet lorsqu'il y a des chemins de retour. [8] modélise formellement ce phénomène de façon à mesurer l'impact de l'insertion et donc à guider l'insertion des stations de relais. Néanmoins, cette approche repose sur une hypothèse simplificatrice très contraignante. Elle considère que tout IP n'est activé que si l'ensemble de ses entrées contient des données valides et l'ensemble des ses sorties dispose de place pour stocker les résultats produits au prochain cycle. Or il est fréquent que seul un sous-ensemble des entrées et des sorties soit nécessaire pour effectuer un pas de calcul synchrone.

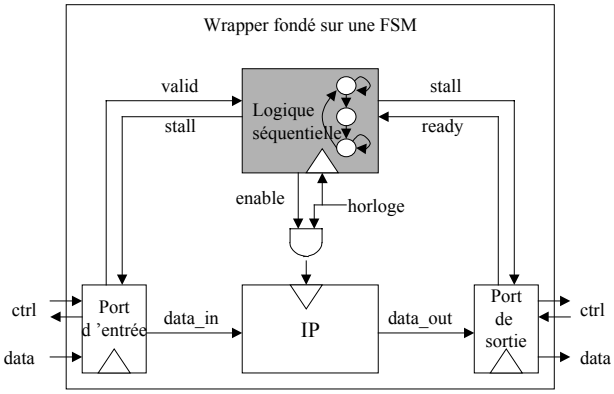

Figure 2 : Modèle de Singh et Theobald

De façon à limiter la sensibilité du processus patient aux seules entrées et sorties nécessaires, Singh et Theobald [9] proposent de remplacer l'horloge sur porte par une machine d'état finis de type Mealy qui teste l'état des seules entrées-sorties pertinentes à chaque cycle. Cette approche est une extension du modèle précédent et a l'avantage de correspondre à un mode de fonctionnement plus réaliste. Elle peut être mise en œuvre si l'on dispose d'une spécification des chronogrammes des entrées-sorties de l'IP qui prouvent que son fonctionnement est bien cyclique et non dépendant des données: c'est à dire qu'il est statiquement prédictible.

Enfin, afin de réduire le coût du matériel, Casu et Macchiarulo [10] prouvent que, s'il est possible de trouver un ordonnancement statique de l'activation des IPs,

alors les stations de relais peuvent être remplacées par des flip-flops et les signaux de synchronisation aval et amont peuvent être supprimés. Le chronogramme cyclique d'activation de l'IP est implanté sous la forme d'un registre à décalage dont le contenu conditionne l'horloge d'activation de l'IP.

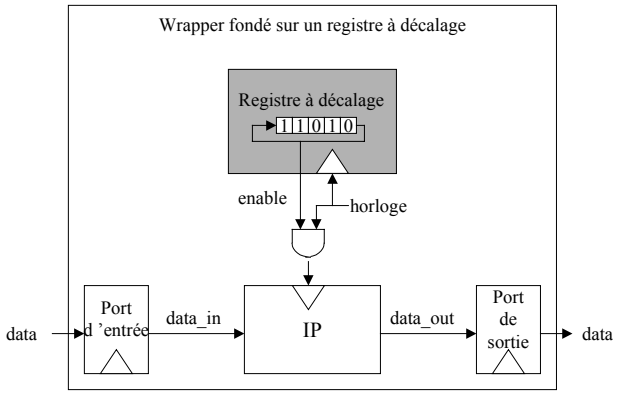

Figure 3 : Modèle de Casu et Macchiarulo

Cette approche repose sur l'hypothèse qu'il n'y a pas d'irrégularités dans les flots de données et qu'il n'est jamais nécessaire de geler aléatoirement les IPs. Elle s'applique donc exclusivement à des systèmes pour lesquels l'environnement produit les données et consomme les résultats suffisamment rapidement.

3. Nouvelle architecture de wrapper

Cette section présente l'encapsulation d'IP dans un nouveau modèle de wrapper. Cette approche permet de produire des wrappers dont la 1) surface est très largement inférieure à celles des wrappers conçus à base de FSM et qui sont 2) synthétisables quels que soient les scénarios de communication.

3.1. Motivation

L'approche de Singh repose sur la synthèse de FSM pour limiter la sensibilité du processus patient aux seules entrées-sorties pertinentes à chaque cycle. La complexité de ces FSMs dépend donc du nombre de cycles et du nombre d'entrées et de sorties de l'IP. Nos expériences de prototypage de systèmes de traitement du signal sur FPGA nous ont montré que les performances et la synthétisabilité des FSM dépend de leur complexité et des outils utilisés. Cette grande sensibilité des outils par rapport à la complexité des FSMs nous a conduit à rechercher un modèle de wrapper qui dépend moins de la complexité des communications.

Pour justifier ce fait, nous avons tout d'abord effectué des tests de synthétisabilité de diverses FSM représentatives de la complexité des scénarios d'IP de traitement du signal. Nous considérons que les IP de traitement du signal qui sont de bons candidats à la réutilisation dans des SoC sont ceux qui ont de forts taux de parallélisme de calcul et de communication. Nous avons choisi les algorithmes DFT/DIT et DCT afin de situer de façon expérimentale ce que nous considérerons ensuite comme l'ordre de grandeur de la complexité de la communication.

A l'aide de l'outil de synthèse de haut niveau GAUT[11] nous avons synthétisé pour différentes contraintes (nombre de points et cadence d'arrivée des échantillons) ces algorithmes pour une cible commune : le VirtexE de Xilinx cadencé à 100 MHz.

	Points	Cadence (µs)	Ports	Cycles d'attente	Cycles de calcul
FFT	64	10,2	3	192	53
		4,7	9	127	37
	128	8,4	16	197	43
		5,1	25	156	33
	256	38,4	6	515	85
		19,2	9	381	50
	512	87,1	6	1027	85
		43,8	9	765	57
DCT	4	1,0	4	39	26
		0,5	8	19	26
	8	3,4	9	91	37
		2,0	16	69	29

Tableau 1 : Résultats de synthèse de FFT & DCT

Un extrait de la campagne de test est présenté dans le tableau 1. Les colonnes contiennent pour chaque algorithme, respectivement de gauche à droite : le nombre de points, la cadence à laquelle arrive les échantillons, le nombre de ports d'entrée-sortie, le nombre de cycles nécessitant d'attendre la présence d'une donnée valide en entrée ou la disponibilité de place en sortie sur au moins un des ports, et le nombre moyen de cycles de calcul sans communication avec le réseau. Ces mesures montrent, dans un contexte particulier, que la communication des IP peut comporter de très nombreux points de synchronisation et que le nombre de ports augmente sensiblement si l'on contraint l'algorithme en temps. Des valeurs de quelques dizaines de ports et quelques centaines de points de synchronisation sont alors tout à fait envisageables.

Ayant obtenu expérimentalement un ordre de grandeur de ce que nous appelons la complexité du scénario

de la communication, nous avons synthétisé des FSM représentatives. Pour cela nous avons mis en place un banc de test dont le but est de générer et de synthétiser avec l'outil de synthèse physique ISE des FSM de synchronisation dont le modèle pseudo-VHDL est décrit par la figure 4.

```
process (rst, clk)
begin
    if rst = '1' then
      enable <= '0' ;
      s := S0 ;
    elsif clk = '1' and clk'event then
      case s is
      when <a synchronization state> =>
        if <check_data> and
           <check_room> then
          state := <nextstate> ;
          enable <= '1' ;
          <pop_push_fifos> <= '1' ;
        else
          enable <= '0' ;
          <pop_push_fifos> <= '0' ;
        end if ;
      when <a computing state> =>
        state := <nextstate> ;
        <pop_push_fifos> <= '0' ;
      end case ;
    end if ;
end process ;
```

Figure 4 : pseudo-VHDL FSM model

Ce modèle de FSM pilote l'horloge combinatoire de l'IP avec le signal *enable* et se synchronise avec l'environnement (qui fonctionne comme des fifos) à l'aide des signaux génériques *pop_push_fifos* qui activent les entrées-sorties pertinentes à chacun des cycles. Les expressions *check_data* et *check_room* ont pour but de vérifier respectivement si les données sont présentes et si les sorties sont libres.

Un extrait des résultats de synthèse physique est présenté dans le tableau 2. Les colonnes contiennent respectivement les nombres de ports, de cycles d'attente et de cycles de calcul, suivis par la surface en slice (unité d'allocation de surface dans le VirtexE) et la fréquence maximale de fonctionnement. Ce banc de test illustre le fait que la surface augmente avec la complexité de la communication (ce qui n'est pas surprenant) mais surtout que la synthèse devient impossible au delà d'une limite qui se situe dans la gamme des complexités des algorithmes testés précédemment.

Ports	Cycles d'attente	Cycles de calcul	Taille (slices)	Freq. max (MHz)
4	4	2	15	162
8	8	4	36	148
16	16	8	110	123
32	32	16	429	105
64	64	32	Impos-sible.	Impossi-ble.
128	128	64	Impos-sible.	Impossi-ble.

Tableau 2 : Résultats de synthèse de FSM

Ce fait, extrêmement restrictif pour nos expériences, a motivé la recherche d'une nouvelle architecture de wrapper assurant la synchronisation de l'IP avec le réseau pseudo-asynchrone qui soit toujours synthétisable, quel que soit l'outil de synthèse employé.

3.2. Modèle d'architecture du wrapper

La solution que nous proposons est équivalente fonctionnellement aux FSM. Il s'agit d'un processeur qui exécute cycliquement des opérations qu'il va lire dans une mémoire asynchrone. La figure 5 illustre la nouvelle structure des processus patients LIS à base de processeurs de synchronisation.

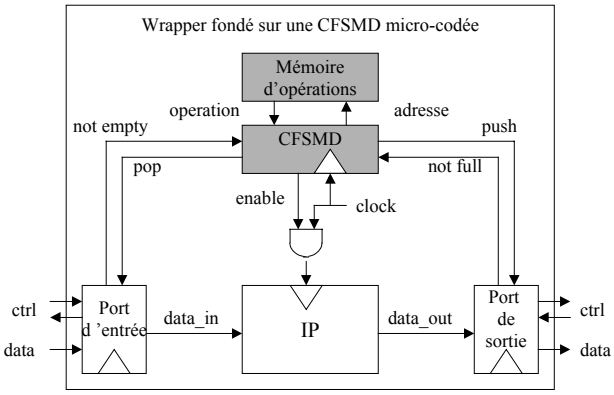

Figure 5 : Modèle de wrapper fondé sur une CFSMD micro-codée

Le wrapper remplace la FSM de Singh et communique avec les ports sur la base de signaux pilotant des FIFOS qui implantent les ports d'entrée-sorties. Ces signaux sont équivalents aux signaux *valid*, *ready* et *stall* de [9]. Le nombre de ports et le nombre de cycles peuvent

être quelconques. Le wrapper est spécifié par une machine d'état finis à trois états, concurrente de l'IP et qui contient un chemin de donnée. Il s'agit d'une CFSMD. La figure 6 présente la spécification de cette CFSMD. Le chemin de données est constitué des variables *pc* (le compteur programme), et *cpt* (le compteur de cycles de calcul restants). L'activation de l'IP encapsulé se fait par le signal *enable* qui est à 1 lorsque l'on doit exécuter un pas de calcul synchrone et 0 autrement. Les expressions *I(pc), O(pc)* et *N(pc)* implantent le décodage des opérations stockées en mémoire et représentent respectivement le masque des ports d'entrées significatifs, le masque des ports de sorties significatifs et le nombre de cycles de calcul immédiatement après la détection de la condition d'activation de l'IP.

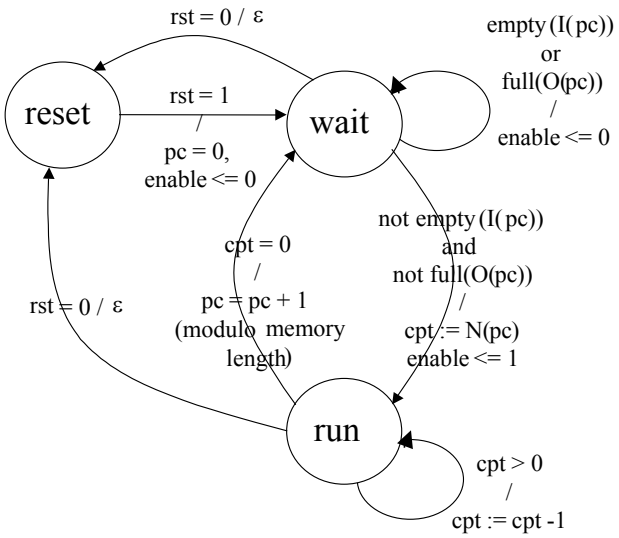

Figure 6 : Spécification du wrapper

Le wrapper est dans l'état *reset* à la mise sous tension et oscille ensuite entre les deux états *wait* et *run* selon qu'il doit se synchroniser avec le réseau ou pas. Afin de ne pas rendre le PS inutilement complexe, la mémoire d'opérations est à lecture asynchrone ce qui réduit les signaux de contrôles entre le PS et la mémoire à deux bus, un pour l'adresse de l'opération à exécuter, un pour la valeur de l'opération immédiatement lue. La mémoire d'opérations étant exclusivement lue, elle peut être implantée sous la forme d'une ROM plutôt que synthétisée sur portes. Elle peut ainsi bénéficier des densités supérieures des mémoires et réduire d'autant l'impact de la complexité de la communication sur la surface du circuit.

3.3. Implantation et résultats

L'implantation VHDL de la CFSMD est triviale. Nous ne donnons ici que les résultats de synthèse physique et les comparons avec des FSM de complexités comparables. La figure 7 donne un extrait de synthèse d'un wrapper de 16 ports d'entrée et 16 ports de sorties. Sa surface est de 16 slices et sa fréquence maximale de fonctionnement est de 124 MHz. Par rapport à une FSM équivalente (FSM 16x16x16), la fréquence est identique et le gain en surface est de 95 %. Enfin, la fréquence du wrapper ne dépendant pas du nombre de cycles de l'IP, il est plus rapide que les FSM de complexité supérieure.

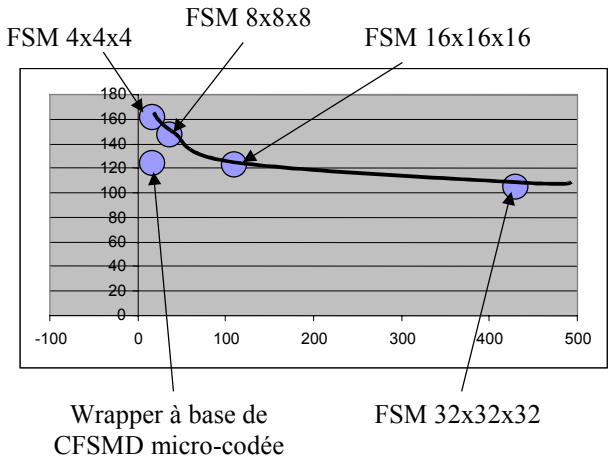

Figure 7 : Comparaison vitesse/surface entre CFSMD micro-codée et FSMs

Notre wrapper a une caractéristique essentielle : sa complexité ne dépend plus du nombre de cycles de calcul de l'IP mais seulement du nombre de ports de l'IP. Par conséquent sa fréquence et sa surface sont constantes, à nombre de ports constant

4. Expérience de synthèse de haut niveau

L'accroissement des besoins en terme de débit et les contraintes de mobilité motivent le développement d'applications de diffusion d'informations via le canal satellite (DVB/DSNG). Les codes de contrôle d'erreurs du type Reed-Solomon et Viterbi sont largement utilisés dans les systèmes de communication numériques pour lutter contre le bruit du canal. Les méthodes traditionnelles de conception de composants numériques sont fondées sur le niveau dit RTL et souffrent de limitations en terme de complexité algorithmique et de flexibilité requise par les différents profils requis. La synthèse de haut niveau sous contraintes de temps accélère la conception de ces composants. Néanmoins, la composition de com-

posants synthétisés de grandes tailles exhibe le même compromis vitesse/fréquence que la réutilisation d'IPs dans les SoCs.

Pour cette raison, nous avons intégré notre wrapper de synchronisation dans l'outil GAUT pour 1) pouvoir interfacer nos composants avec un réseau pseudo-asynchrone et 2) obtenir des systèmes de tailles raisonnables et synthétisables. De plus, le découplage introduit nous permet d'intégrer naturellement des FIFOs de profondeurs variables pour absorber les aléas de communication. Cela a été appliqué à la conception de blocs Reed-Solomon et Viterbi. Le tableau 3 donne les résultats comparatifs entre approches à base de FSMs et à base de wrappers de synchronisation. Il montre que des gains très importants en surface et en vitesse peuvent ainsi être obtenus.

	Complexité	FSM		Wrapper		Gain (%)	
	Port/wait/run	Sli.	Fr.	Sli	Fr.	Sli.	Fr.
Viterbi 64	5 / 4 / 198	494	105	24	105	-95	0
RS syndrom decoder	4 / 2957 / 1	2610	71	24	105	-99	+47

Table 3 : Résultats applicatifs

5. Conclusion et extensions

Ce papier présente notre contribution en terme de modèle de wrapper pour processus patients dans les LIS. Ce modèle, fondé sur une CFSMD micro-codée, permet de mieux préserver la fréquence maximale de fonctionnement d'un SoC. Les mesures comparatives de synthèse physique du wrapper de synchronisation par rapport aux FSMs équivalentes prouvent qu'il apporte un gain en surface non négligeable sans dégrader la vitesse maximale de fonctionnement. Une expérience de conception d'application de type DVB/DSNG illustre le fait que l'adjonction du wrapper permet aussi de composer plus facilement des IP complexes synthétisés par l'outil de synthèse de haut niveau GAUT en vue d'obtenir un système complet qui s'auto-adapte naturellement aux irrégularités de traitement et de communication avec son environnement.

Plusieurs extensions à nos travaux sont envisagées : optimisation de la surface, application du concept avec une technologie ASIC et extension de la fonctionnalité aux systèmes dynamiquement reconfigurables.

- Nous avons validé le concept du wrapper sur FPGA VirtexE et poursuivons actuellement l'expérience avec une technologie 0.12 µm de STMicroelectronics.
- Dans le cadre de systèmes reconfigurables constitués d'IPs ayant plusieurs comportements sélectionnables, le stockage en RAM des opérations du wrapper permet d'adapter la synchronisation au comportement de l'IP par rechargement d'un nouveau jeu d'opérations.

Remerciements

Ces travaux ont été financés par le "*Conseil Régional de Bretagne*", le "*Conseil Général du Morbihan*" et la "*Communauté de communes du pays de Lorient*" dans le cadre du projet PALMYRE et par le projet RNRT ALIPTA.

Références

[1] International Technology Roadmap for Semiconductors, 2003 edition.

[2] L. Benini and G. De Micheli, ''Networks on Chips: A New SoC Paradigm,''. in IEEE Computer, Jan. 2002, pp. 70-78.

[3] D. M. Chapiro, ''Globally-Asynchronous Locally-Synchronous Systems,'' PhD Thesis, Stanford University, Oct. 1984

[4] A. Chakraborty and M.R. Greenstreet, ''Efficient self-timed interfaces for crossing clock domains,'' in Proceedings of the International Symposium on Asynchronous Circuits ans Systems (ASYNC'03), Vancouver, May 2003.

[5] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli, ''Theory of Latency-Insensitive Design,'' in. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 20(9) :18, Sept. 2001

[6] L. P. Carloni and A. L. Sangiovanni-Vincentelli, ''A Formal Modeling Framework for Deploying Synchronous Designs on Distributed Architectures,'' in First International Workshop on Formal Methods for Globally Asynchronous Locally Synchronous (FMGALS'03), Pisa, Sept. 2003.

[7] L. P. Carloni and A.L. Sangiovanni-Vincentelli, ''Coping with Latency in SoC Design,'' in IEEE Micro, Special Issue on Systems on Chip, 22(5) :12, Oct. 2002.

[8] L. P. Carloni and A. L. Sangiovanni-Vincentelli, ''Performance Analysis and Optimization of Latency Insensitive Systems,'' in Proceedings of the 37th Design Automation Conference (DAC'00), June 2000.

[9] M. Singh and M. Theobald, ''Generalized Latency-Insensitive Systems for Single-Clock and Multi-Clock Architectures,'' in Proceedings of the Design Automation and Test in Europe Conference (DATE'04), Paris, Feb. 2004.

[10] M. R. Casu and L. Macchiarulo, ''A New Approach to Latency Insensitive Design,'' in Proceedings. of the Design and Automation Conference (DAC'04), San Diego, June 2004.

[11] GAUT, http://web.univ-ubs.fr/gaut/, LESTER, France