



HAL
open science

Flot de conception pour l'ultra faible consommation : échantillonnage non-uniforme et électronique asynchrone

Jean Simatic

► **To cite this version:**

Jean Simatic. Flot de conception pour l'ultra faible consommation : échantillonnage non-uniforme et électronique asynchrone. Micro et nanotechnologies/Microélectronique. Université Grenoble Alpes, 2017. Français. NNT : 2017GREAT084 . tel-01758184

HAL Id: tel-01758184

<https://theses.hal.science/tel-01758184>

Submitted on 4 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Nano Électronique & Nano Technologies**

Présentée par

Jean SIMATIC

Thèse dirigée par **Laurent FESQUET**
et co-encadrée par **Rodrigo POSSAMAI BASTOS**

Préparée au sein du **Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés (TIMA)** dans l'**École Doctorale Électronique, Électrotechnique, Automatique & Traitement du Signal (EEATS)**

Flot de conception pour l'ultra-faible consommation

Échantillonnage non uniforme et électronique asynchrone

Thèse soutenue publiquement le **07 décembre 2017**,
devant le jury composé de :

Monsieur Frédéric PÉTROU

Professeur des Universités, Grenoble INP, Président

Monsieur Bruno ROUZEYRE

Professeur des Universités, Université de Montpellier II, Rapporteur

Monsieur Olivier SENTIEYS

Professeur des Universités, Université de Rennes I, Rapporteur

Monsieur Paolo IENNE

Professeur, École polytechnique fédérale de Lausanne, Examineur

Monsieur Laurent FESQUET

Maître de Conférence, Grenoble INP, Directeur de thèse

Monsieur Rodrigo POSSAMAI BASTOS

Maître de Conférence, Université Grenoble Alpes, Co-Encadrant de thèse



Table des matières

Remerciements	xiii
1 Introduction	1
2 État de l'art	5
2.1 Échantillonnage basé sur des évènements	7
2.1.1 Schémas d'échantillonnage non uniforme	7
2.1.2 CAN à traversée de niveaux (CAN-TN)	9
2.1.3 Placement des seuils	10
2.1.4 Traitement des signaux échantillonnés non uniformément (NUS)	11
2.2 Circuits asynchrones	15
2.2.1 Classes de circuits asynchrones	16
2.2.2 Modélisation de circuits asynchrones	18
2.2.3 Protocoles de communication	23
2.3 Synthèse de haut niveau (HLS)	27
2.3.1 Techniques et architecture typiques synchrones	28
2.3.2 Machines à états finis asynchrones (AFSM)	30
2.3.3 Techniques et outils de HLS asynchrones	30
2.4 Conclusion	32
3 Le flot de conception ALPS	33
3.1 ALPS : un flot général pour la conception de systèmes basse consommation basés sur des évènements	35
3.1.1 Vue d'ensemble du flot	35
3.1.2 Choix de l'échantillonnage et de l'algorithme	36
3.1.3 Synthèse de haut niveau	36
3.1.4 Réalisation physique	36
3.1.5 Validation et vérification	37
3.2 ALPS-HLS : Synthèse de haut niveau asynchrone	37
3.2.1 Architecture cible et chemins de données	37
3.2.2 Désynchronisation de la partie de contrôle	38
3.3 ALPS-RTC : Contraintes temporelles et insertion de retards	42
3.3.1 Points de contrôle de temps de l'AFSM	42
3.3.2 Contraintes temporelles de l'AFSM	43

3.3.3	Méthode de résolution	46
3.4	Études de cas	48
3.4.1	Choix du schéma d'interpolation pour l'ASA	48
3.4.2	Dimensionnement d'un filtre RIF non uniforme	51
3.5	Conclusion	58
4	Optimisation de la consommation des micropipelines	59
4.1	Évaluation des protocoles tardifs et nouveaux protocoles	61
4.1.1	Caractéristiques temporelles des pipelines à données groupées . . .	61
4.1.2	Nouveaux protocoles tardifs	65
4.1.3	Comparaison des protocoles tardifs et précoces	66
4.1.4	Perspective : pipelines multi-protocoles	72
4.2	Modèle comportemental à base de réseaux de Petri	73
4.2.1	Modèle « canal équivalent » : fragments élémentaires des primitives	73
4.2.2	Méthodologie de génération par simplification d'un STG de référence	76
4.2.3	Exemple : Conception et vérification d'un AES	81
4.3	Conclusion	83
5	Conclusion	85
	Glossaire	89
	Publications de l'auteur	101

Table des figures

1.1	Objectifs du travail à l'intersection de trois paradigmes.	1
1.2	Le flot de conception ALPS.	3
2.1	Échantillonnage par traversée de niveaux.	8
2.2	Architecture du CAN-TN suiveur.	10
2.3	Algorithme de Sélection d'Activité (QAISAR, SIMATIC et FESQUET 2017).	12
2.4	Calcul d'un échantillon de sortie dans le cas non uniforme.	14
2.5	Modèle de circuit synchrone.	15
2.6	Classes de circuits asynchrones (adapté de KONDRATYEV et LWIN 2002).	16
2.7	Modèle de circuit QDI.	17
2.8	Modèle de circuit micropipeline.	18
2.9	Approches de conception directe et par primitives.	19
2.10	Traduction de primitives en différents modèles formels.	22
2.11	Conventions de validité possibles (PEETERS et BERKEL 1995).	24
2.12	Graphes d'états des protocoles dans l'environnement de BIRTWISTLE et STEVENS (2008).	25
2.13	Protocoles tardifs existants.	26
2.14	Positionnement en termes de découplage des protocoles proposés par rapport aux protocoles tardifs existants.	27
2.15	L'algorithme du PGCD avant et après HLS.	28
2.16	Architecture résultante de la HLS synchrone.	29
2.17	Stratégies existantes de HLS asynchrone.	32
3.1	Le flot de conception ALPS.	35
3.2	Architecture cible du flot ALPS.	37
3.3	Structure d'un étage de l'AFSM.	38
3.4	Chronogramme de l'activation d'un état.	40
3.5	Structures des contrôleurs d'états modifiées pour gérer efficacement les transferts avec l'environnement.	41
3.6	Réalisation logique des composants de sélection.	41
3.7	Points de contrôle de temps dans l'AFSM.	43
3.8	Reformulation des contraintes temporelles avec des sommes de délais.	47
3.9	Formulation du problème d'optimisation du dimensionnement des retards.	48
3.10	Processus de ré-échantillonnage de l'ASA.	49

3.11	Signaux de test analogique et échantillonnés.	53
3.12	Architecture d'un filtre RIF uniforme synchrone optimisé en surface.	53
3.13	Conception manuelle du RIFI.	54
3.14	Surfaces des filtres.	56
3.15	Temps de calcul des filtres.	57
3.16	Consommation en énergie des filtres.	57
4.1	Structure et temps caractéristiques d'un pipeline asynchrone.	61
4.2	STGs de protocoles précoces et 2 phases.	63
4.3	Chemins critiques des protocoles tardifs existants.	64
4.4	Relations entre les protocoles tardifs.	65
4.5	Chemins critiques des nouveaux protocoles tardifs.	65
4.6	Réalisations physique et logique des portes de Muller.	68
4.7	Réalisations logique des contrôleurs tardifs.	68
4.8	Pipeline en anneau pour le banc de test.	68
4.9	Temps de propagation et surface.	69
4.10	Temps cycle minimum.	70
4.11	Débit en fonction de la taille du chemin critique et du taux d'occupation.	71
4.12	Énergie consommée et score de mérite (<i>Figure Of Merit</i>) (FOM).	71
4.13	Correspondance comportementale entre <i>buffers</i> et transitions.	74
4.14	Fragments de réseau de Petri associés aux unions, fourches et <i>buffers</i>	75
4.15	Fragments de réseau de Petri associés aux multiplexeurs et démultiplexeurs.	76
4.16	Simplification des places.	80
4.17	Simplification de transitions avec des poids sur les arcs.	80
4.18	Exemples de modèles obtenus à partir d'un même circuit.	81
4.19	Exemple de conception et vérification d'un AES.	82
5.1	Bilan du travail effectué sur ALPS et perspectives.	85

Liste des tableaux

2.1	Exemple de famille de primitives avec leur représentation en structures statiques de flot de données (SPARSØ et FURBER 2002, Fig. 3.3).	20
3.1	Définitions et couts grossiers des schémas d'interpolation	49
3.2	Surface et consommation en fonction du schéma d'interpolation.	50
3.3	Nombre d'échantillons produits par le dépassement du compteur de temps en fonction de la profondeur du compteur.	52
4.1	Propriétés des protocoles tardifs comparés aux protocoles précoces.	67
4.2	Positionnement du modèle proposé.	73
4.3	Simplification de transitions.	79

*À Milène
sans qui le protocole Maximus
n'aurait pas vu le jour*

The clock-free design paradigm must
eventually prevail. It fits physics.

Tyranny of the clock
IVAN SUTHERLAND

TANSTAAFL

ADAGE ANGLAIS

Le Père Noël n'existe pas !

LUANA PATRICIA CUNHA CARNEIRO

Remerciements

La thèse a été pour moi un grande aventure. Je tiens à remercier par ces quelques lignes celles et ceux qui en ont été acteurs et qui ont participé à ce qu'elle soit pour moi si passionnante et enrichissante.

Un grand merci à Laurent pour m'avoir donné l'opportunité de mener cette thèse, pour son ouverture d'esprit, pour m'avoir tantôt supporté, tantôt aiguillé, tantôt poussé. Merci à Rodrigo pour ses relectures approfondies et pour nos débats fructueux.

Aux collègues de l'équipe CDSI¹ pour l'ambiance chaleureuse, d'échange et de curiosité qui y règne, merci beaucoup. J'ai notamment des pensées particulières pour Pauline et son adage « dans le doute, reboot », pour Karim, sa gentillesse et la pertinence de ses réflexions, pour Rodrigo (Pudu) et les heures passées à essayer de refaire le monde, pour Assia qui m'a appris à couper le temps en quatre, pour Leonel, mon complice dans le label RES, pour Thiago et ses découvertes musicales, pour Leonel et sa cuisine, pour Otto et nos cours matinaux, pour Raphael et Ricardo pour m'avoir laissé la majeure partie du tableau du bureau, mais aussi Amani, Sophie, Tugdual, Marvin, Ali, Grégoire, Matheus, Alice et Fabrice. Merci également à mes collègues à Phelma et au CIME de m'avoir permis de me former à leur contact. Un énorme merci à François Cayre pour sa bonne humeur, son support moral et pédagogique.

Non, la thèse ne s'arrête pas une fois la porte du labo fermée. Ainsi, je tiens aussi à remercier mes amis pour leur participation à leur façon à cette aventure. Merci aux fanfarons pour les fantastiques souvenirs, la musique, la fête. Merci en particulier à Martinouche, Clem trompette, Juju, JB, le jeune, Marie clar et à la Com'Apéro en général pour l'amitié partagée. Merci aux membres de l'orchestre des campus, en particulier Pierre pour ses nombreux conseils, Émérance, Rémi, Quentin et Célestin pour les répétitions au Clos des Marronniers², mes voisins de pupitre pour leur indulgence et Milène en particulier pour m'avoir fait découvrir le festival du film de montagne et le ski de fond.

Au terme de mes remerciements, merci à ma famille, mes parents, mon frère et mes sœurs pour m'avoir soutenu depuis le début, pour leur disponibilité et pour leurs oreilles attentives. J'ai enfin une pensée pour tous les professeurs, camarades et amis dont j'ai croisé la route et sans qui je n'aurais jamais commencé cette thèse. Sans qui je ne serais peut-être jamais allé à Grenoble. Sans qui je n'aurais pas trouvé sur mon chemin une petite tornade qui secoue ma vie et la secouera encore longtemps j'espère.

Merci !

1. Un bien étrange acronyme pour *Design of Integrated Circuits and Systems*.

2. Coup de pub éhonté.

Introduction

Aujourd’hui notre société numérique échange de plus en plus d’informations. La quantité de données échangées est incroyablement élevée et le futur promet un accroissement du trafic avec l’arrivée du transfert de données entre équipements tels que robots, capteurs, etc. Cette orgie de données consomme une quantité très importante d’énergie et contribue à une approche non écologique de nos systèmes électroniques et de communication. Les estimations selon les sources varient entre 2 et 10% de la consommation d’électricité dans le monde mais il est évident que cette part va croître à l’avenir avec l’internet des objets (*Internet of Things*) (IoT).

Afin de limiter la consommation des systèmes électroniques, il existe déjà des solutions matérielles mais qui sont trop limitées pour endiguer à elles seules la croissance de ce secteur. Il faut donc envisager des mesures plus radicales comme réduire le flot de données produit à la source. Contrairement à l’échantillonnage classique uniforme où la fréquence d’échantillonnage reste constante, les techniques d’échantillonnage non uniforme permettent d’adapter la cadence d’échantillonnage en fonction de l’activité du signal afin de ne pas produire d’échantillons superflus. Ces techniques ont déjà démontré qu’elles pouvaient amener à une réduction importante du nombre d’échantillons produits, surtout lorsque le signal d’entrée est sporadique.

En agissant à la source et en utilisant des schémas d’échantillonnage non uniforme et parcimonieux, il est possible de réduire sensiblement la consommation du système à condition d’exploiter les données par une électronique numérique adaptée. En effet, l’approche usuelle tout synchrone ne se prête pas bien à cette approche où les données sont produites irrégulièrement. Il faut donc se tourner vers les logiques événementielles utilisées dans les circuits asynchrones.

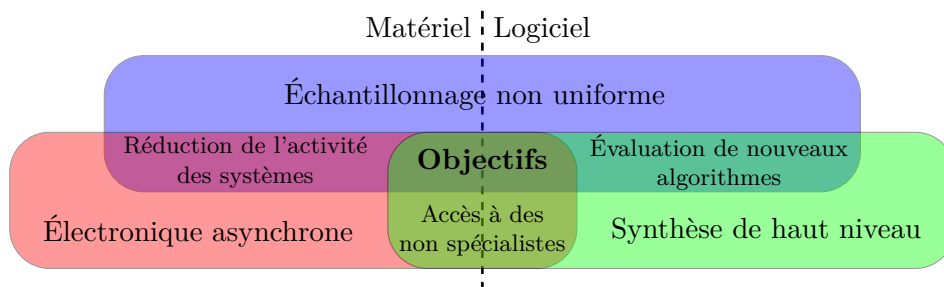


FIGURE 1.1 – Objectifs du travail à l’intersection de trois paradigmes.

Ainsi, comme l'indique la FIGURE 1.1, l'utilisation conjointe d'un schéma d'échantillonnage non uniforme et de l'électronique asynchrone permet d'envisager la réduction de la consommation des systèmes électroniques par la diminution de l'activité des systèmes.

Dans le but de faciliter la conception de ces systèmes par des non spécialistes, la synthèse de haut niveau (*High-Level Synthesis*) (HLS) est utilisée pour déterminer une description matérielle à partir d'une spécification algorithmique du calcul réalisé par le système. L'architecture n'étant pas fixée au départ comme pour la synthèse logique, le processus de HLS explore les architectures possibles. Par ailleurs, la description algorithmique permet une meilleure réutilisation du code (car plus compacte que la description matérielle) et aussi un partitionnement plus facile entre logiciel et matériel. Enfin, dans le cadre des circuits asynchrones, la HLS est intéressante car le concepteur n'a pas besoin de maîtriser les particularités de la conception asynchrone.

Dans le cadre de ce travail, l'environnement de synthèse et de simulation va permettre d'évaluer rapidement les algorithmes de traitement des signaux échantillonnés non uniformément. En effet, le traitement des données échantillonnées non uniformément ne peut être fait efficacement avec les mêmes algorithmes que pour les données échantillonnées uniformément. La traduction directe des algorithmes du cadre uniforme vers le cadre non uniforme entraîne l'augmentation de la complexité de l'algorithme afin de prendre en charge l'irrégularité du flux. En revanche, pour un problème donné, l'élargissement du cadre d'échantillonnage peut permettre l'utilisation de nouveaux algorithmes beaucoup plus simple que ceux classiquement utilisés.

Un flot de conception dédié aux systèmes électroniques basés sur les évènements

Cette thèse introduit un flot de conception pour les systèmes d'acquisition et de traitement du signal basés sur des évènements. Il est appelé ALPS : outils architecturaux pour systèmes basse consommation basés sur les évènements (*Architectural tools for Low-Power event-driven Systems*). Ce flot, représenté sur la FIGURE 1.2, doit permettre 1) à haut niveau, d'aider le concepteur à déterminer les paramètres d'échantillonnage pertinents pour l'application visée et à concevoir l'algorithme de traitement adéquat en s'aidant éventuellement des évaluations fournies par le flot, 2) de générer le Convertisseur Analogique/Numérique (CAN) correspondant aux paramètres et 3) de synthétiser à partir de sa description algorithmique la description matérielle du système de traitement afin de pouvoir évaluer les performances du système matériel.

Il est important de noter ici que pour atteindre la plus grande efficacité possible, l'échantillonnage et l'algorithme sont conçus en visant une application particulière. Cette connaissance *a priori* de l'application se présente sous la forme de base de données de signaux ou de propriétés statistiques sur le signal.

Ce mémoire présente une version du flot dédiée à l'échantillonnage non uniforme et au traitement de signaux unidimensionnels. Le flot de conception pourra être étendu à terme

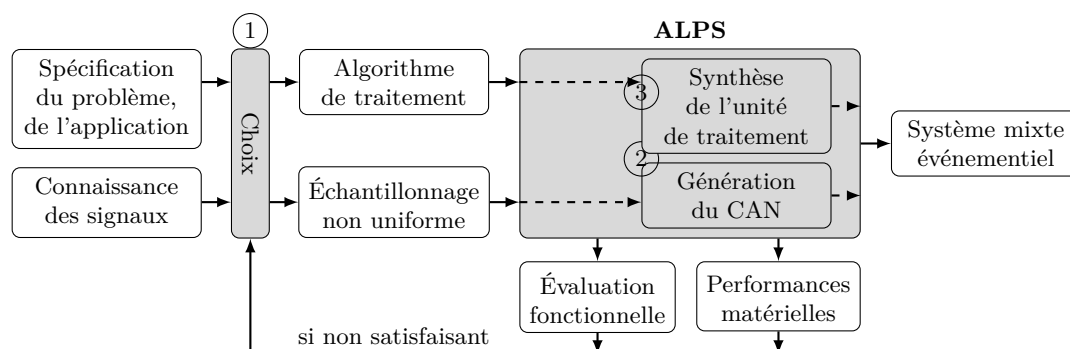


FIGURE 1.2 – Le flot de conception ALPS.

pour y inclure la comparaison des différentes méthodes d'échantillonnage non uniforme. De façon plus générale, le cadre de ce flot de conception peut aussi être utilisé dans le cas de l'échantillonnage parcimonieux d'images et du traitement du flot de données associé.

Plan du mémoire

Le CHAPITRE 2 présente l'état de l'art de l'échantillonnage non uniforme, des circuits asynchrones et de la synthèse de haut niveau. La contribution principale de cette thèse, le flot de conception ALPS, est discutée dans le CHAPITRE 3. Les autres contributions, relatives à la spécification, la vérification et l'optimisation des circuits asynchrones, sont décrites dans le CHAPITRE 4. Enfin, le manuscrit est conclu en indiquant que la méthode et l'outillage développé dans cette thèse est un pas pour proposer des systèmes peu énergivores utilisables pour l'IoT par exemple.

État de l'art

Sommaire

2.1	Échantillonnage basé sur des évènements	7
2.1.1	Schémas d'échantillonnage non uniforme	7
2.1.1.1	Échantillonnage à traversée de niveaux (TN)	7
2.1.1.2	Échantillonnages issus de la dérivée	8
2.1.1.3	Quantification des amplitudes et délais	8
2.1.2	CAN à traversée de niveaux (CAN-TN)	9
2.1.2.1	CAN-TN Sigma/Delta (SENAY et al. 2010)	9
2.1.2.2	CAN-TN flash (AKOPYAN, MANOHAR et APSEL 2006)	9
2.1.2.3	CAN-TN suiveur (ALLIER et al. 2005)	9
2.1.3	Placement des seuils	10
2.1.4	Traitement des signaux échantillonnés non uniformément (NUS)	11
2.1.4.1	Algorithme de sélection d'activité (ASA)	11
2.1.4.2	Filtre à réponse impulsionnelle finie interpolé (RIFI)	13
2.2	Circuits asynchrones	15
2.2.1	Classes de circuits asynchrones	16
2.2.1.1	Circuits à validité des données intégrée dans l'encodage	17
2.2.1.2	Circuits à validité des données groupée	17
2.2.2	Modélisation de circuits asynchrones	18
2.2.2.1	Contexte d'utilisation des modèles formels de contrôleurs asynchrones	19
2.2.2.2	Modèles formels à base de réseaux de Petri	21
2.2.3	Protocoles de communication	23
2.2.3.1	Signalisation des données valides	24
2.2.3.2	Environnements de travail sur les protocoles	25
2.2.3.3	Protocoles tardifs existants	26
2.3	Synthèse de haut niveau (HLS)	27
2.3.1	Techniques et architecture typiques synchrones	28
2.3.2	Machines à états finis asynchrones (AFSM)	30
2.3.3	Techniques et outils de HLS asynchrones	30
2.3.3.1	Traduction dirigée par la syntaxe	30
2.3.3.2	Décomposition en processus pipelinés	31
2.3.3.3	Flots basés sur l'ordonnancement	31
2.4	Conclusion	32

Le flot ALPS utilise l'échantillonnage non uniforme, l'électronique asynchrone et de la HLS pour faciliter la conception et l'évaluation de systèmes basse consommation basés sur des événements. Ce chapitre présente l'état de l'art de ces domaines.

2.1 Échantillonnage basé sur des événements

Afin de préserver l'information spectrale d'un signal, le théorème de Shannon-Nyquist affirme que la fréquence d'échantillonnage de ce signal doit être supérieure à deux fois la fréquence maximale du spectre du signal. De ce fait, l'échantillonnage uniforme produit beaucoup d'échantillons inutiles lorsque le signal est inactif ou variant lentement. Les signaux échantillonnés non uniformément (*Non-Uniformly Sampled*) (NUS) ont pour propriété de ne pas avoir une fréquence d'échantillonnage fixe. En introduisant une dépendance entre l'activité du signal et la cadence (variable) de capture des échantillons, les techniques NUS permettent de réduire le nombre d'échantillons à traiter.

2.1.1 Schémas d'échantillonnage non uniforme

Un schéma d'échantillonnage détermine les instants auxquels les échantillons doivent être capturés. Le cas uniforme est un cas particulier puisqu'il s'agit d'instants régulièrement espacés. L'espacement en question est la période d'échantillonnage. Il est intéressant de noter que ce cas est le cas idéal. En pratique, de légères variations sur le signal d'horloge (gigue ou *jitter*) rendent l'échantillonnage non uniforme. Les instants d'échantillonnage sont alors définis statistiquement indépendamment de l'activité du signal.

De nombreux schémas existent pour lier la cadence d'échantillonnage à l'activité du signal à échantillonner. Les paragraphes suivants décrivent trois des principaux schémas. BIDÉGARAY-FESQUET et FESQUET (2016) présentent d'autres possibilités et les comparent.

2.1.1.1 Échantillonnage à traversée de niveaux (TN)

Introduit par MARK et TODD (1981), le schéma par traversée de niveaux (TN) requiert de définir un ensemble de seuils (aussi appelés niveaux). Un échantillon est alors capturé à chaque fois que le signal analogique traverse un des seuils.

Un mécanisme d'hystérésis¹ peut être introduit afin d'éviter des échantillons superflus si le signal fait des petites oscillations autour d'un seuil. Avec l'hystérésis, un échantillon est capturé seulement lorsque le signal croise un seuil strictement au-dessus ou au-dessous du dernier niveau croisé (voir FIGURE 2.1).

Comme indiqué sur la FIGURE 2.1, deux informations sont à enregistrer pour chaque échantillon : l'amplitude du seuil traversé et le délai par rapport à l'échantillon précédent. Notons que si les seuils restent à des amplitudes fixées, l'information sur l'amplitude peut

1. « L'hystérésis est la propriété d'un système qui tend à demeurer dans un certain état quand la cause extérieure qui a produit ce changement d'état a cessé. » (Wikipédia)

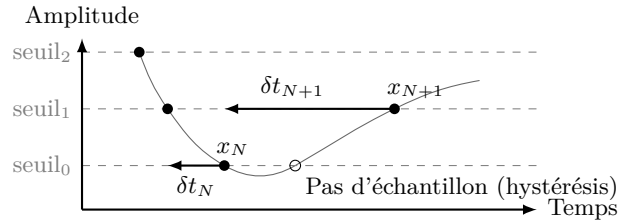


FIGURE 2.1 – Échantillonnage par traversée de niveaux. Pour l'échantillon k , x_k désigne son amplitude et δt_k le délai par rapport à l'échantillon précédent.

être remplacée par la direction binaire de la traversée : le seuil croisé est-il au-dessus ou au-dessous du dernier seuil croisé ?

Si les seuils sont uniformément distribués, le schéma TN avec hystérésis est équivalent au schéma *send-on-delta* (MISKOWICZ 2006), où un échantillon est produit lorsque l'écart entre la valeur du signal et celle du dernier échantillon atteint un quanta déterminé.

2.1.1.2 Échantillonnages issus de la dérivée

Lors de variations de fortes amplitudes, l'échantillonnage TN peut produire beaucoup d'échantillons alors que le signal est presque linéaire entre les extrémums. Une solution est d'échantillonner le signal seulement aux extrémums (GREITANS et al. 2011). Autrement dit, le signal est échantillonné à chaque fois que la dérivée temporelle du signal s'annule. Plus généralement, la dérivée du signal peut être échantillonnée TN (MARTÍNEZ-NUEVO, PATIL et TSIVIDIS 2015). L'information sur l'amplitude se retrouve alors par intégration.

2.1.1.3 Quantification des amplitudes et délais

La conversion des amplitudes et des délais en valeurs quantifiées est nécessaire pour un traitement numérique du signal. Le traitement numérique offre notamment une meilleure insensibilité au bruit et est de conception plus aisée que le traitement analogique (IFEACHOR et JERVIS 2002).

L'échantillonnage TN n'induit pas d'erreur de quantification sur les amplitudes car les amplitudes des seuils sont définies sans erreur. En revanche, les autres schémas (GREITANS et al. 2011 ; MARTÍNEZ-NUEVO, PATIL et TSIVIDIS 2015 ; BIDÉGARAY-FESQUET et FESQUET 2016) nécessitent une quantification pour pouvoir effectuer des calculs sur les amplitudes et introduisent donc une erreur lors de cette quantification.

La quantification des délais est donc la seule composante du bruit de quantification de l'échantillonnage TN (ALLIER et al. 2005). Il est possible de ne pas quantifier les délais mais cela requiert des éléments de retard coûteux en matériel (SCHELL et TSIVIDIS 2008).

2.1.2 CAN à traversée de niveaux (CAN-TN)

Trois architectures de Convertisseurs Analogique/Numérique à Traversée de Niveaux (CANs-TN) ont été proposées dans la littérature. Ces architectures sont similaires à leurs équivalents classiques du cas de l'échantillonnage uniforme.

2.1.2.1 CAN-TN Sigma/Delta (Senay et al. 2010)

La sortie d'un CAN-TN à modulation Sigma/Delta est binaire (tension haute ou basse). L'amplitude du signal est codée par le rapport cyclique de la sortie, c'est à dire la moyenne locale de la valeur du signal durant un cycle. Cette moyenne locale est une estimation de la valeur du signal durant ce cycle. Une constante d'intégration permet de sélectionner des cycles courts ou longs et donc la précision du CAN.

Cette méthode est intéressante car l'échantillonnage s'adapte de lui-même au signal. Toutefois, les informations sur l'amplitude et les délais sont difficiles à calculer pour le traitement du signal.

2.1.2.2 CAN-TN flash (Akopyan, Manohar et Apsel 2006)

Dans ce CAN-TN, une « échelle verticale » de N contrôleurs asynchrones traite la comparaison de l'entrée analogique avec N niveaux de référence. Un jeton unique circule entre les contrôleurs. Ainsi, un contrôleur recevant le jeton l'envoie au contrôleur suivant vers le haut ou vers le bas en fonction du résultat de la comparaison. Le contrôleur envoie aussi une requête contenant la direction de réexpédition du jeton. Les requêtes de tous les contrôleurs convergent vers la sortie du CAN, codant ainsi l'amplitude. Le délai est de ce fait codé implicitement par le temps séparant deux requêtes (temps continu).

2.1.2.3 CAN-TN suiveur (Allier et al. 2005)

L'architecture de ce CAN-TN est décrite dans la FIGURE 2.2. L'entrée analogique est comparée à deux seuils de référence générés par deux Convertisseurs Numérique/Analogique (CNAs). S'il n'y a pas de croisement, les signaux *incr* et *decr* sont inactifs et le compteur de niveaux ne change pas de valeur. Si le signal d'entrée franchit le seuil supérieur, le signal *incr* fait feu et le compteur de niveaux est incrémenté. De même, si le signal d'entrée franchit le seuil inférieur, le signal *decr* fait feu et le compteur de niveaux est décrémenté.

Le changement de valeur du compteur de niveau a pour effet de produire un échantillon contenant l'amplitude du niveau traversé et le délai par rapport au dernier échantillon. Ce dernier est calculé par un compteur de temps remis à zéro à chaque nouvel échantillon (temps quantifié). Le changement de valeur du compteur de niveau a aussi pour effet de mettre à jour les tensions en entrée des CNAs et donc les seuils de référence. GRIMALDI, RODRIGUEZ et RUSU (2011) ont proposé une version à temps continu du CAN-TN suiveur.

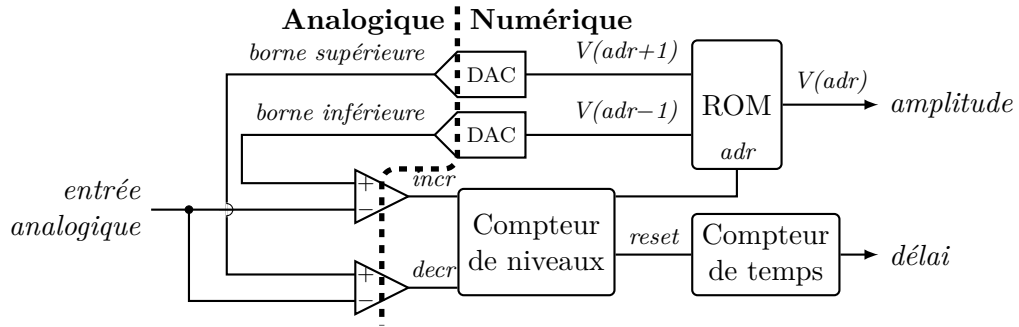


FIGURE 2.2 – Architecture du CAN-TN suiveur.

Dans la suite, le CAN-TN suiveur d'ALLIER et al. (2005) est préféré car il est compact, permet potentiellement des adaptations en ligne ou hors ligne d'un nombre arbitraire de seuils (LE PELLETER 2015). En outre, les amplitudes et délais des échantillons de sortie sont quantifiés, ce qui est adéquat pour le traitement numérique.

2.1.3 Placement des seuils

Ces architectures ont été originellement présentées avec des seuils distribués uniformément sur la dynamique d'entrée. Des versions plus récentes, comme le CAN-TN de GUAN, KOZAT et SINGER (2008), permettent d'ajuster arbitrairement les seuils et ainsi explorer le compromis entre la consommation en énergie et le nombre d'échantillons disponibles pour le traitement.

Dans le cas spécifique de la reconnaissance de formes de signaux physiologique, LE PELLETER (2015) a montré que 4 seuils pouvaient suffire pour une reconnaissance satisfaisante². Dans ce cas particulier, l'utilisation de ces seuls 4 seuils a permis de réduire de 3 ordres de grandeur la consommation en énergie par rapport à un système utilisant un échantillonnage uniforme et une convolution pour la reconnaissance. Les seuils sont sélectionnés par un algorithme d'apprentissage appliqué à un ensemble de traces réelles. Dans ce cas, la sélection minutieuse des seuils aide à préserver les traits caractéristiques des signaux permettant de les distinguer plutôt que de prendre un grand nombre d'échantillons pour garantir une reconstruction précise du signal. Cet exemple montre comment une approche spécifique à une application donnée peut contribuer à diminuer la puissance consommée.

Sans aucune connaissance *a priori* sur le signal et étant donné seulement un nombre de seuils, GUAN, KOZAT et SINGER (2008) a proposé un algorithme séquentiel en ligne qui converge vers l'ensemble de seuils minimisant l'erreur quadratique entre le signal d'entrée et le signal reconstruit (interpolation à l'ordre 0). Étant donné une base de données des signaux et une erreur limite, cet algorithme pourrait être utilisé hors ligne

2. ROC $\sim 0,95$. ROC est l'intégrale de la fonction d'efficacité du récepteur (*Receiver Operating Characteristic*), c'est-à-dire pour un classificateur, le ratio de vrais positifs sur les faux positifs. Le ROC d'un classificateur idéal est égal à 1.

pour sélectionner un ensemble de seuils de cardinal minimal.

À notre connaissance, aucune méthode générale de choix des seuils n'a été proposée pour sélectionner les seuils. L'existence d'un tel environnement est incertain si ce n'est comme collection de méthodes chacune applicable à une certaine classe de problèmes.

2.1.4 Traitement des signaux échantillonnés non uniformément (NUS)

La littérature sur l'échantillonnage par traversée de niveaux traite principalement des algorithmes de reconstruction du signal (FONTAINE et RAGOT 2001 ; MARVASTI 2001). Les imprécisions sont calculées et la qualité de l'échantillonnage examinée au regard d'une application spécifique telle la compression. Les filtres ont aussi été amplement étudiés, notamment les filtres à réponse impulsionnelle finie (RIF) (à temps continu (SCHELL et TSIVIDIS 2008) et à temps quantifié (AESCHLIMANN et al. 2004)) et infinie (SAYINER, SORENSEN et VISWANATHAN 1996 ; FESQUET et BIDÉGARAY-FESQUET 2010a ; BIDÉGARAY-FESQUET et FESQUET 2011).

Le flot présenté dans ce mémoire sera testé avec deux études de cas : un Algorithme de Sélection d'Activité (ASA) (QAISAR, FESQUET et RENAUDIN 2014) (SECTION 3.4.1) et un filtre RIF (AESCHLIMANN et al. 2004) (SECTION 3.4.2).

2.1.4.1 Algorithme de sélection d'activité (ASA)

Afin de permettre le traitement des signaux NUS par des algorithmes classiques, QAISAR, FESQUET et RENAUDIN (2014) proposent de sélectionner des fenêtres pendant lesquelles le signal est actif puis de ré-échantillonner chaque fenêtre uniformément à une fréquence optimale réduisant le nombre d'échantillons. La FIGURE 2.3a montre l'enchaînement des opérations d'échantillonnage non uniforme, de l'ASA puis du ré-échantillonnage.

La fenêtre utilisée ici est une fenêtre rectangulaire de longueur temporelle variable.

L'ASA est décrit par la FIGURE 2.3b. $T_0 = \frac{1}{f_{\min}}$ est la période fondamentale du signal filtré $x(t)$. La période T_0 et le délai mesuré δt permettent de détecter les parties actives du signal NUS. Si $\delta t > \frac{T_0}{2}$, alors le signal est considéré comme inactif. Cette condition assure que le critère d'échantillonnage de Shannon-Nyquist est vérifié pour f_{\min} .

N^i représente le nombre d'échantillons NUS contenus dans la i^e fenêtre W^i . La longueur temporelle de W^i est $L^i = \sum_{n \in W^i} \delta t_n$. Par conséquent, la fréquence moyenne d'échantillonnage F_s^i pour W^i est donnée par $F_s^i = \frac{N^i}{L^i}$. Pour une représentation spectrale correcte, la condition $L^i \geq T_0$ doit être respectée (IFEACHOR et JERVIS 2002 ; QAISAR, FESQUET et RENAUDIN 2008). Afin de satisfaire cette condition dans le pire cas, qui correspond à une sinusoïde d'amplitude maximale et de fréquence f_{\max} , la fenêtre doit avoir une longueur maximale d'au moins N_{\max} échantillons donné par l'équation (2.1) où M est le nombre de seuils (QAISAR, FESQUET et RENAUDIN 2008 ; QAISAR, FESQUET et RENAUDIN 2014). De même, une fenêtre n'est considérée valide que si elle contient au moins une période. Afin de satisfaire cette condition dans le cas d'une sinusoïde

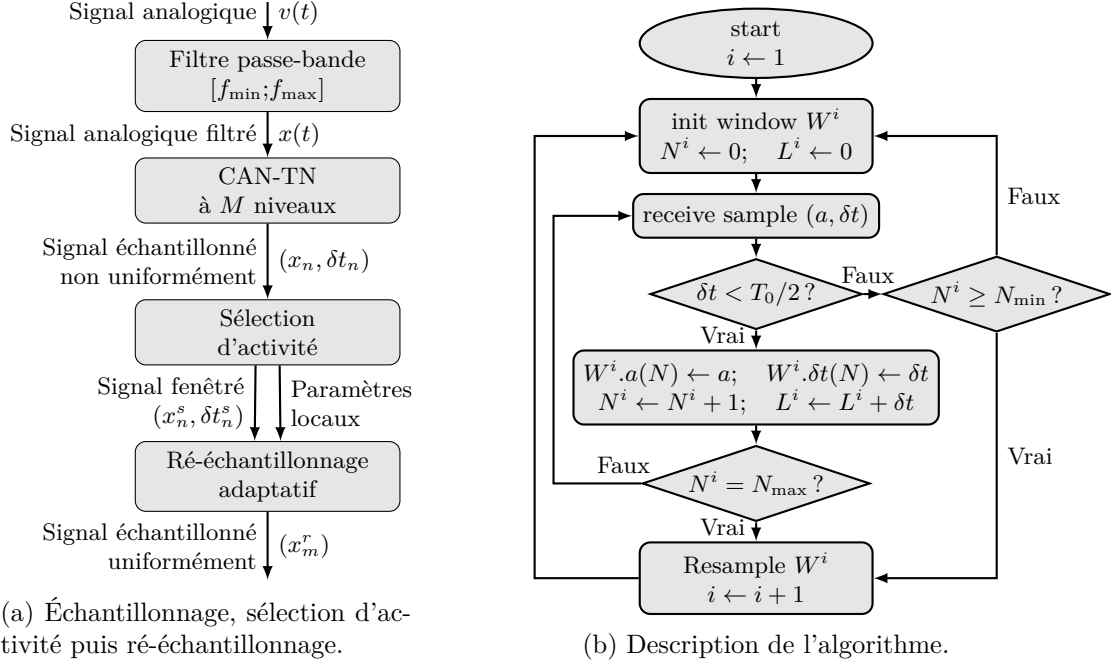


FIGURE 2.3 – Algorithme de Sélection d'Activité (QAISAR, SIMATIC et FESQUET 2017).

d'amplitude maximale et de fréquence f_{\min} , la fenêtre doit avoir une longueur minimale N_{\min} (2.2). Dans ces équations, le facteur $2(M-1)$ est le nombre d'échantillons capturés (avec hystérésis) en une période d'une sinusoïde de pleine amplitude.

$$N_{\max} = 2(M-1) \frac{f_{\max}}{f_{\min}} \quad (2.1)$$

$$N_{\min} = 2(M-1) \quad (2.2)$$

Comme l'indique la FIGURE 2.3b, les échantillons ne sont accumulés dans la fenêtre que tant que le signal est actif ($dt_n \geq \frac{T_0}{2}$). Lorsque la fenêtre est pleine ($N^i = N_{\max}$) ou lorsque le signal redevient inactif et que la fenêtre est suffisamment remplie ($N^i \geq N_{\min}$), la fenêtre est ré-échantillonnée et une nouvelle fenêtre est commencée.

L'un des désavantages du CAN-TN est que des parties intéressantes du signal peuvent être localement sur-échantillonnées comparées au cas classique (jusqu'à un facteur $M-1$ dans le pire des cas). Pour éviter ces échantillons superflus et ainsi améliorer la consommation énergétique du circuit, les fenêtres sont ré-échantillonnées.

Soit F_{ref} la fréquence de référence d'un CAN uniforme de référence et F_{rs}^i la fréquence de ré-échantillonnage de la fenêtre W^i . Selon le théorème de Shannon-Nyquist, F_{ref} doit être supérieur à $F_{\text{Nyq}} = 2f_{\max}$. La fréquence de ré-échantillonnage F_{rs}^i est choisie comme le minimum entre la fréquence de référence F_{ref} et la fréquence moyenne d'échantillon-

nage de la fenêtre F_s^i (2.3).

$$F_{rs}^i = \min \left(F_{\text{ref}}, F_s^i \right) = \min \left(F_{\text{ref}}, \frac{N^i}{L^i} \right) \quad (2.3)$$

En effet, si le CAN-TN produit plus d'échantillons que le CAN classique ($F_s^i > F_{\text{ref}}$), alors F_{rs}^i est choisi égal à F_{ref} . W_i est ré-échantillonné proche de la fréquence de Nyquist.

Dans l'autre cas, le CAN-TN produit moins d'échantillons que le CAN classique ($F_s^i < F_{\text{ref}}$). F_{rs}^i est alors choisie égale à F_s^i . La fréquence de ré-échantillonnage peut ainsi être plus basse que la fréquence de Nyquist. Toutefois, cela ne pose pas de problème de repliement spectral si les seuils du CAN-TN sont suffisamment nombreux et bien placés de telle sorte que le signal soit localement sur-échantillonné par rapport à sa bande passante locale (ALLIER et al. 2005 ; AKOPYAN, MANOHAR et APSEL 2006 ; KOZMIN, JOHANSSON et DELSING 2009 ; GUAN et SINGER 2007).

Différents schémas d'interpolation sont possibles pour le ré-échantillonnage du signal NUS. QAISAR et al. (2016) comparent différents schémas en termes de qualité du signal et de nombre d'opérations. Dans la SECTION 3.4.1, le flot outils architecturaux pour systèmes basse consommation basés sur les événements (*Architectural tools for Low-Power event-driven Systems*) (ALPS) est utilisé pour quantifier l'énergie consommée par chacun des schémas et les surfaces respectives des circuits correspondants. Le concepteur a ainsi plus d'informations pour choisir le compromis approprié pour son application.

2.1.4.2 Filtre à réponse impulsionnelle finie interpolé (RIFI)

Un filtre RIF calcule le produit de convolution y entre un signal d'entrée x et une fonction de transfert h , appelée aussi réponse impulsionnelle car $y = h$ si x est une impulsion (Dirac). La réponse impulsionnelle est finie si la fonction h est limitée dans le temps.

Le produit de convolution analogique est donné par (2.4). Dans le cas d'un échantillonnage uniforme à une période T_s , l'intégrale est décomposée sur des intervalles de largeur T_s . Sur ces intervalles, en supposant une interpolation à l'ordre zéro³, l'intégrale est le produit des amplitudes et de la période. Ainsi, (2.4) devient (2.5) où x_i , y_i et h_i sont respectivement les i^{e} échantillons de x , y et h .

$$y(t) = \int_i x(t-i).h(i)dt \quad (2.4)$$

$$y_k = \sum_{i=0}^N T_s . x_{k-i} . h_i \quad (2.5)$$

Il est possible de renverser les indices de x de telle sorte que, pour calculer y_k , $\tilde{x}_0 = x_k$ est le dernier échantillon reçu, $\tilde{x}_1 = x_{k-1}$ est l'avant dernier échantillon reçu, etc.

3. Autrement dit constante par morceaux ou en escalier.

Cette indexation correspond à l'adressage naturel d'un registre à décalage en matériel. Comme le montre (2.6), la nouvelle indexation enlève toute référence à k dans la formule. Cela signifie que le circuit pourra fonctionner indépendamment du nombre (non borné) d'échantillons reçus.

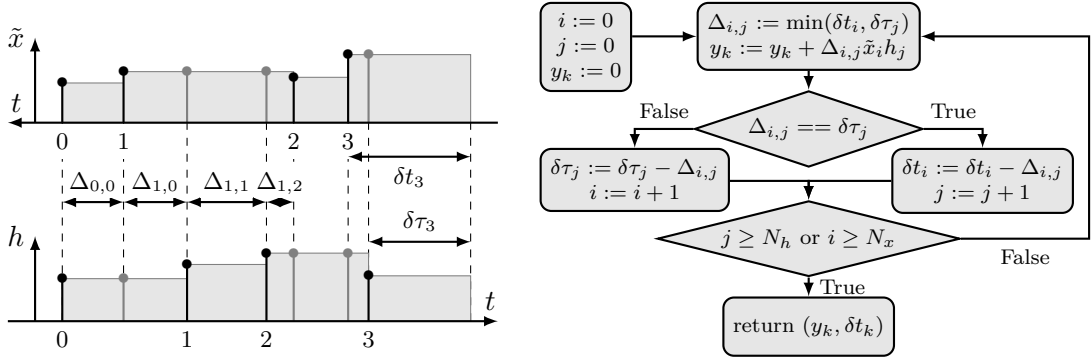
$$y_k \propto \sum_{i=0}^N \tilde{x}_i \cdot h_i \quad (2.6)$$

Dans le cas non uniforme, les échantillons de h et x ne sont pas synchronisés. La décomposition sur des intervalles de taille fixe T_s n'est donc plus possible.

Soient $h = (h_j, \delta\tau_j)$ une fonction de transfert, potentiellement NUS, et $x = (x_i, \delta t_i)$ un signal NUS à filtrer. δt_i et $\delta\tau_j$ sont respectivement les délais séparant les i^e et j^e échantillons de x et h du précédent. Le signal \tilde{x} est la renversée de x comme présenté précédemment.

La FIGURE 2.4a montre la décomposition du domaine d'intégration en intervalles de largeur $\Delta_{i,j}$ sur lesquels les deux signaux sont constants à l'ordre 0. L'intégrale sur l'intervalle $\Delta_{i,j}$ se calcule par (2.7) où « $\hat{\cdot}$ » désigne l'opérateur d'interpolation. À l'ordre 0, l'intégrale est égale au produit des amplitudes et de la largeur de l'intervalle.

$$\int_{\Delta_{i,j}} \tilde{x}(t) \cdot h(t) dt \approx \int_{\Delta_{i,j}} \hat{\tilde{x}}(t) \cdot \hat{h}(t) dt = \Delta_{i,j} \cdot \tilde{x}_i \cdot h_j \quad (2.7)$$



(a) Décomposition en intervalles $\Delta_{i,j}$ sur lesquels \tilde{x} et h sont constants (interpolation à l'ordre 0). L'axe des temps de \tilde{x} est renversé. (b) Algorithme d'AESCHLIMANN et al. (2004) pour calculer le k^e échantillon de sortie. N_h est le nombre d'échantillons de h et N_x le nombre d'échantillons stockés de x .

FIGURE 2.4 – Calcul d'un échantillon de sortie dans le cas non uniforme. L'interpolation est d'ordre 0.

Ainsi, l'amplitude y_k du signal de sortie y avec le délai δt_k (comme le signal d'entrée) est donné par la somme discrète (2.8).

$$y_k = \sum_{i,j} \int_{\Delta_{i,j}} \tilde{x}(t) \cdot h(t) dt \approx \sum_{i,j} \Delta_{i,j} \cdot \tilde{x}_i \cdot h_j \quad (2.8)$$

$\Delta_{i,j}$ peut être formellement défini par (2.9), mais cette définition mène à de nombreux termes nuls. AESCHLIMANN et al. (2004) ont proposé un algorithme qui itère efficacement sur les indices i et j . La FIGURE 2.4b décrit cet algorithme qui, après la capture du k^e échantillon de x , à savoir $(x_k, \delta t_k)$, renvoie le k^e échantillon de y , à savoir $(y_k, \delta t_k)$.

$$\Delta_{i,j} \stackrel{\text{def}}{=} \|[\tilde{t}_{i-1}, \tilde{t}_i] \cap [\tau_{j-1}, \tau_j] \| \quad \text{où} \quad \begin{cases} \tilde{t}_i = \sum_{n=0}^i \delta \tilde{t}_n \\ \tau_j = \sum_{n=0}^j \delta \tau_n \end{cases} \quad (2.9)$$

La condition d'arrêt $i > N_x$ dans la FIGURE 2.4b correspond à la limite matérielle de la mémoire. Afin d'éviter les pertes de précision, la longueur N_x doit vérifier (2.10).

$$\forall k, \quad \sum_{i=0}^{N_x} \delta t_{k-i} \geq \sum_{j=0}^{N_h} \delta \tau_j \quad (2.10)$$

Dans la suite, le filtre RIF pour les signaux NUS est appelé RIF interpolé (RIFI). D'autres schémas d'interpolations sont possibles mais mènent à des calculs d'aires plus complexes (FESQUET et BIDÉGARAY-FESQUET 2010b).

Dans la SECTION 3.4.2, cet algorithme sert de cas d'étude pour comparer des circuits générés avec le flot ALPS à des circuits conçus manuellement ou générés par des outils de synthèse de haut niveau classiques.

2.2 Circuits asynchrones

Les circuits synchrones, comme celui représenté sur la FIGURE 2.5, utilisent un signal global communément appelé « horloge » pour signaler à tous les registres qu'il est temps de faire avancer leurs résultats respectifs. Tous les étages doivent avoir le temps de finir leur calcul au cours d'un cycle d'horloge. Par conséquent, la période de l'horloge doit être plus longue que tous les chemins combinatoires de tous les étages. Le plus long chemin est nommé chemin critique. Cette contrainte temporelle est globale au circuit et détermine le débit du circuit.

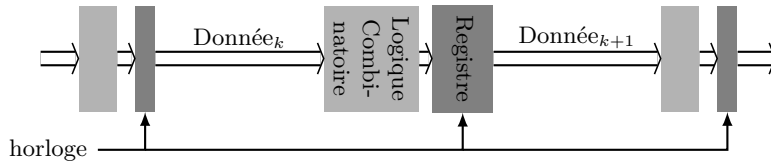


FIGURE 2.5 – Modèle de circuit synchrone.

Le signal d'horloge est un signal global qui doit atteindre tous les registres simultanément (idéalement). Ainsi, l'ensemble du matériel requis pour distribuer correctement ce signal, à savoir l'arbre d'horloge, consomme $\sim 20 - 45\%$ de l'énergie totale d'un circuit (SITIK et al. 2016) et émet un fort rayonnement électromagnétiques aux harmoniques de la fréquence de l'horloge. De plus, l'augmentation de la taille des puces nécessite d'avoir

plusieurs horloges et donc de dépenser quelques cycles de latence supplémentaires pour transférer des données d'un domaine d'horloge à l'autre (GINOSAR 2003).

Le paradigme asynchrone consiste à remplacer le signal d'horloge global par des communications locales entre les éléments du circuit. Tout comme la transition de l'échantillonnage uniforme à non uniforme ouvre sur de nombreux schémas d'échantillonnage, il y a de nombreuses façons de concevoir un circuit asynchrone.

2.2.1 Classes de circuits asynchrones

Comme l'indique SEITZ (1980) au sujet des circuits auto-séquenceés :

*The canonical forms in the time dimension are signalling conventions that are adhered to throughout the system and serve the function of establishing between all parts engaged in a communication an interval or sequence of intervals of time for this communication. If such a scheme is to be regarded as a discipline, it must be possible to state precisely the requirements that the signalling convention places on system interconnections and element timing.*⁴

Ainsi, un circuit asynchrone est avant tout caractérisé par une « convention de signalisation », c'est-à-dire la manière d'indiquer à un bloc l'arrivée d'une nouvelle donnée. La convention choisie induit des « exigences », c'est-à-dire des hypothèses sur le déroulement des séquences d'évènements, appelées *hypothèses temporelles*. La FIGURE 2.6 regroupe les circuits en 5 classes qui se répartissent selon 3 conventions de signalement et 3 types d'hypothèses temporelles.

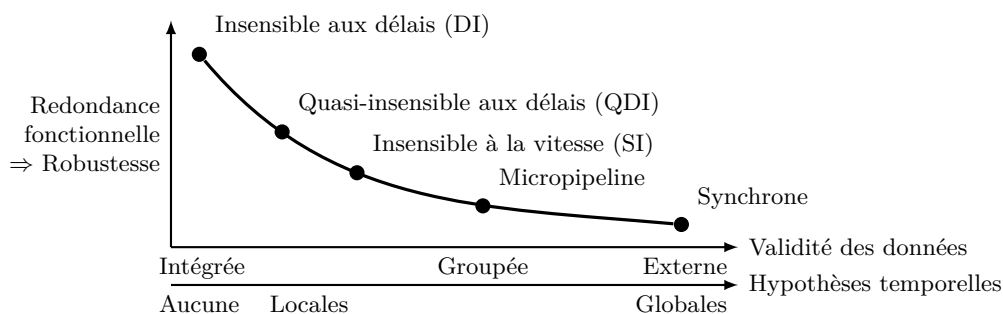


FIGURE 2.6 – Classes de circuits asynchrones (adapté de KONDRATYEV et LWIN 2002).

Aux deux extrémités se trouvent d'une part les circuits synchrones qui utilisent une horloge (externe) pour signaler la validité des données et requièrent de ce fait une contrainte globale sur la taille des chemins combinatoires. À l'inverse, les circuits insensibles aux délais ne requièrent aucune hypothèse temporelle. Ils sont donc plus robustes. Néanmoins, ces circuits sont difficiles à concevoir (MARTIN 1990b).

4. Mise en forme personnelle.

2.2.1.1 Circuits à validité des données intégrée dans l'encodage

Les circuits quasiment insensibles aux délais (*Quasi Delay Insensitive*) (QDI), comme celui représenté sur la FIGURE 2.7, ne nécessitent seulement que certaines fourches soient isochrones, c'est à dire qu'un évènement en entrée de la fourche ait le temps de se propager à toutes les sorties avant qu'un nouvel évènement arrive. Cette hypothèse locale temporelle suffit pour réaliser une machine de Turing (MANOHAR et MARTIN 1995) et donc n'importe quel système de calcul⁵.

Les circuits QDI utilisent un encodage spécifique, typiquement 1-parmi- N , pour indiquer la validité des données. Si les N fils sont au niveau logique bas, cela signifie qu'il n'y a pas de données. Si un des fils est au niveau logique haut, cela signifie qu'il y a une donnée. La valeur de la donnée présente dépend du fil qui est activé. Cet encodage donne aux circuits QDI une forte résilience face aux variations de délais mais nécessite plus de connexions et des portes logiques plus complexes.

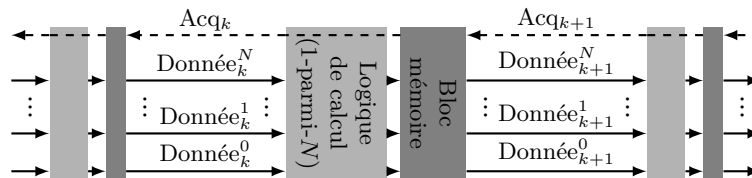


FIGURE 2.7 – Modèle de circuit QDI.

Le bloc mémoire utilise le codage pour détecter l'arrivée d'une donnée valide. Un acquittement est calculé par un arbre de complétion détectant que la donnée a bien été capturée. L'acquittement indique qu'une nouvelle donnée peut être envoyée.

BOUESSE et al. ont montré que les circuits QDI émettaient moins d'émissions électromagnétiques (2007) et étaient plus résistants aux attaques par canaux cachés (2005) que leurs *alter ego* synchrones. En contrepartie, la surface des circuits QDI est plus importante ($\sim 70\%$).

Les circuits insensibles à la vitesse (FIGURE 2.6) sont proches des circuits QDI. Comme hypothèse temporelle, les circuits insensibles à la vitesse supposent que les délais dans les fils sont négligeables. Autrement dit, toutes les fourches sont isochrones. En revanche, les délais dans les portes ne sont pas bornés.

2.2.1.2 Circuits à validité des données groupée

Les circuits à données groupées, comme celui représenté sur la FIGURE 2.8, sont aussi connu sous le nom de micropipelines (SUTHERLAND 1989). Les chemins de données sont similaires aux circuits synchrones mais l'horloge globale est remplacée par des signaux de type « poignée de main » (*handshake*).

5. Par définition de la calculabilité. Selon la thèse de Church, cette définition correspond au sens usuel du terme.

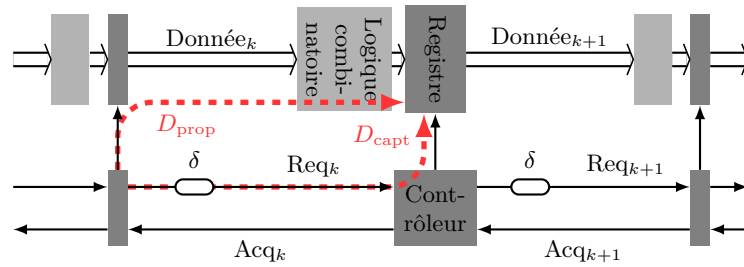


FIGURE 2.8 – Modèle de circuit micropipeline.

Chaque registre (groupe de bascules) est contrôlé par un contrôleur asynchrone. Le contrôleur gère les signaux de *handshake* : les fils de requêtes qui indiquent des données valides et les fils d’acquittements qui indiquent que de nouvelles données peuvent être capturées. L’enchaînement des requêtes et des acquittements obéit à des protocoles présentés dans la SECTION 2.2.3.

À la différence des circuits QDI où la validité de la donnée est intégrée dans l’encodage, les circuits micropipeline adoptent la convention de données groupées. Les signaux de *handshake* sont en effet groupés avec la donnée qu’ils accompagnent. L’ensemble formé par les fils de données et la paire de fils de *handshake* est abstrait en un canal de communication.

Au niveau des hypothèses temporelles, la notification d’une nouvelle donnée valide doit avoir lieu après que la donnée ait fini de se propager à travers le bloc combinatoire précédent. Autrement dit, sur la FIGURE 2.8, le délai de propagation D_{prop} doit donc être plus court que le délai de capture D_{capt} . Des retards sont insérés sur les requêtes pour respecter cette contrainte de données groupées ($D_{prop} < D_{capt}$).

Les circuits à données groupées offrent un compromis intéressant entre compacité du circuit et robustesse face aux variations. En effet, le codage nécessite peu de matériel comparé au QDI et les retards sur les fils de requêtes peuvent être corrélés aux délais dans les chemins de données (CORTADELLA et al. 2010 ; MORENO et CORTADELLA 2017) et ainsi permettre de meilleures performances dans les conditions typiques tout en garantissant le fonctionnement dans les pires conditions.

2.2.2 Modélisation de circuits asynchrones

Les circuits asynchrones sont intrinsèquement concurrents : les différents composants agissent séparément dans le temps et l’espace tout en interagissant les uns avec les autres.

Les langages de description de matériel (*Hardware Description Languages*) (HDLs) tels VHDL et Verilog permettent de décrire des processus concurrents par des primitives à bas niveau de synchronisation, typiquement l’attente d’un changement de valeurs. En revanche, ils ne permettent pas d’abstraire un canal dans un modèle de type producteurs/consommateurs. Pour ce faire, il faut un modèle au niveau transactionnel (*Transaction Level Model*) (TLM) qui permet de décrire le transfert de donnée en rendant

implicite les aspects liés au protocole. Les langages tels CSP (*Communicating Sequential Processes*) (HOARE 1978), son adaptation au matériel CHP (*Communicating Hardware Processes*) (MARTIN 1990a) et les plus récents SystemVerilog et SystemC proposent ce niveau de modélisation TLM.

Les réseaux de Petri (PNs) (PETRI 1966) sont un formalisme basé non pas sur un langage mais sur un graphe. Contrairement aux langages TLM, ils ne permettent pas d'exprimer simplement des opérations arithmétiques. Nous utiliserons toutefois les PNs pour modéliser des contrôleurs asynchrones car ils permettent de décrire toutes les primitives de synchronisations et offrent une visualisation naturelle de l'exécution du modèle.

2.2.2.1 Contexte d'utilisation des modèles formels de contrôleurs asynchrones

Comme le montre la FIGURE 2.9, il existe deux types d'approches dans lesquelles les modèles formels sont utilisés dans la conception un circuit asynchrone. Les deux approches de conception, à savoir directe et par primitives, commencent à partir d'une idée de l'architecture du chemin de données et permettent d'obtenir à la fin la description du contrôleur en portes logiques.

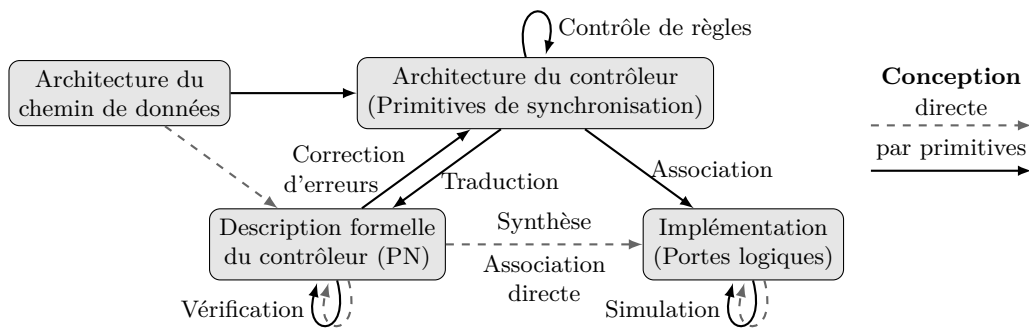


FIGURE 2.9 – Approches de conception directe et par primitives.

Approche de conception directe Dans ce premier cas, le concepteur construit directement le modèle qui convient en fonction de l'architecture du chemin de données. Les vérifications sont directement faites sur le modèle formel.

Ensuite, le circuit est déduit de ce modèle soit par association directe (*direct mapping*) soit par synthèse. Dans le cas de l'association directe, la structure matérielle des éléments du circuit est contrainte car chacun des éléments doit respecter la sémantique des PNs (HOLLAAR 1982; SOKOLOV, BYSTROV et YAKOVLEV 2003). Cela amène soit à diminuer l'expressivité des réseaux, soit à complexifier les contrôleurs.

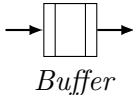
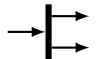

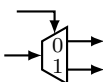
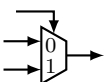
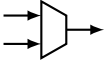
La synthèse de contrôleurs asynchrones à partir de modèles comme les STGs (CHU 1986) (voir SECTION 2.2.2.2, §2) ne passe pas à l'échelle car elle nécessite de calculer le graphe d'états du circuit dont la taille est exponentielle en la complexité du circuit (YAKOVLEV et al. 1996).

Approche de conception par primitives La deuxième possibilité de conception consiste à ne pas spécifier le circuit directement avec un PN mais à utiliser une spécification tierce et générer le PN en fonction de cette dernière. Le circuit est typiquement spécifié par l'assemblage de primitives de base réalisant des fonctions élémentaires de synchronisation. Les différentes familles de primitives offrent différentes granularités et expressivités. Citons par exemple la « famille minimale » des *Static Dataflow Structures* (SDFSs) (SPARSØ et FURBER 2002) et *eXecutable Micro Architectural Specification* (xMAS) (CHATTERJEE, KISHINEVSKY et OGRAS 2012). Les *Handshake components* (BERKEL 1993) ajoutent par exemple une primitive de répétition.

La réalisation physique des primitives d'une famille peut dépendre des objectifs de conception. Par exemple, les canevas *Single Track Full Buffer* (STFB) (FERRETTI et BEEREL 2006) et GasP (SUTHERLAND et FAIRBANKS 2001) visent de hauts débits, tandis que le canevas *Click* (PEETERS et al. 2010) est réalisable avec des bibliothèques standards. Finalement le canevas *Blade* (HAND et al. 2015) et son successeur *Sharp* (WAUGAMAN et KOVEN 2017) sont robustes aux violations temporelles.

La spécification en primitive permet ainsi d'éviter à avoir à synthétiser directement le modèle formel. Le modèle formel sert uniquement à la vérification et est obtenu par assemblage des modèles élémentaires de chacune des primitives. Ces modèles élémentaires peuvent avoir différentes formes décrites dans la SECTION 2.2.2.2. Des vérifications peuvent aussi être faites directement sur les primitives par le contrôle de règles nécessaires à un bon fonctionnement.

TABLE 2.1 – Exemple de famille de primitives avec leur représentation en structures statiques de flot de données (SPARSØ et FURBER 2002, Fig. 3.3).

Type de synchronisation	1 vers 1	1 vers N	N vers 1
Mémorisation	 <i>Buffer</i>		
Non-conditionnelle		 Fourche	 Union
Sélective		 Démultiplexeur	 Multiplexeur
Non déterministe			 Arbitre

La TABLE 2.1 décrit une famille typique de primitives. La mémorisation est assurée par un *buffer*. La sortie du *buffer* peut être pleine ou vide, c'est-à-dire contenir une

donnée ou non. Si l'entrée du *buffer* contient une donnée et que la sortie est vide, alors le *buffer* déplace la donnée de son entrée vers sa sortie. Ainsi, l'entrée est vidée et la sortie est remplie.

La fourche envoie une donnée reçue à son entrée vers toutes ses sorties. Réciproquement, l'union attend une donnée à chacune de ses entrées avant d'en envoyer une vers sa sortie. Le démultiplexeur quant à lui attend une donnée et une valeur de sélection. La donnée est envoyée sur la sortie correspondant à la valeur de sélection. Le multiplexeur est le dual du démultiplexeur.

Enfin, l'arbitre attend une donnée sur l'une de ses entrées. La première donnée arrivant est envoyée sur la sortie. Tous les circuits n'ont pas besoin d'arbitres. En particulier, les circuits dits élastiques (CARMONA et al. 2009) sont caractérisés par des flots de donnée concurrents mais déterministes et donc sans arbitrage. Cette restriction permet de n'utiliser qu'une sous-classe de PNs pour le modèle formel.

Dans ce contexte de conception par primitives, différents modèles formels à base de PNs peuvent être utilisés.

2.2.2.2 Modèles formels à base de réseaux de Petri

Après une courte introduction aux PNs et à leur fonctionnement, nous présentons dans cette section les différentes manières d'utiliser les PNs pour modéliser des circuits asynchrones. Les modèles présentés correspondent à un choix de compromis entre l'expressivité, la taille et la lisibilité du modèle.

Réseaux de Petri (PNs) Les PNs (PETRI 1966) sont un outil général pour représenter des actions soumises à des conditions d'activation. De manière formelle, un PN est un graphe bipartite orienté où des nœuds de types places et transitions alternent. Chaque transition est donc connectée uniquement à des places. De par l'orientation du graphe, les places connectées à une transition sont divisibles en deux classes : les places d'entrée et les places de sortie de la transition.

Les places peuvent contenir des jetons, qui sont interprétés comme des ressources disponibles. Les transitions sont alors des actions qui vont déplacer les ressources selon une règle de transition. Une transition est autorisée à tirer si ses places d'entrée contiennent au moins un jeton. Lorsque la transition tire, elle retire un jeton de chacune des places d'entrée et en rajoute à chacune des places de sortie.

Cette règle de transition a deux extensions principales. La première est d'ajouter des poids aux arcs représentant une multiplicité. Alors, le tir d'une transition déplace autant de jetons qu'indiqué par le poids de l'arc. La seconde extension est de fixer une capacité limite de jetons pour les places. Alors, les transitions ne sont autorisées à tirer que si les places de sortie ont suffisamment de capacité disponible.

Signal Transition Graphs (STGs) CHU (1986) introduit les STGs pour modéliser et analyser les circuits asynchrones. Les STG sont un sous-ensemble des PNs dans les-

quels les transitions marquées $X+$ et $X-$ indiquent respectivement les fronts montant et descendant du signal X . Un STG est valide s'il vérifie les règles correspondant au fonctionnement d'un circuit. Par exemple, deux événements $X+$ ne peuvent avoir lieu consécutivement sans un événement $X-$ entre les deux (règle d'alternance).

Les transitions des signaux du STG sont les transitions du PN. Réciproquement, les arcs du STG sont les places dans le PN. Ils peuvent donc contenir un jeton représenté par un disque. Ainsi, les arcs représentent les conditions relatives d'ordonnancement sur les événements dans le circuit. Lors de la représentation d'un STG dans un environnement, comme dans la FIGURE 2.10c, l'ordonnancement imposé par l'extérieur est représenté par des arcs pointillés.

Comme le montre la FIGURE 2.10c, une spécification de STG conduit à un réseau de grande taille. Or les problèmes de vérification sur les PN sont en général NP-complets (ESPARZA 1998). D'autres modèles plus compacts ont été proposés.

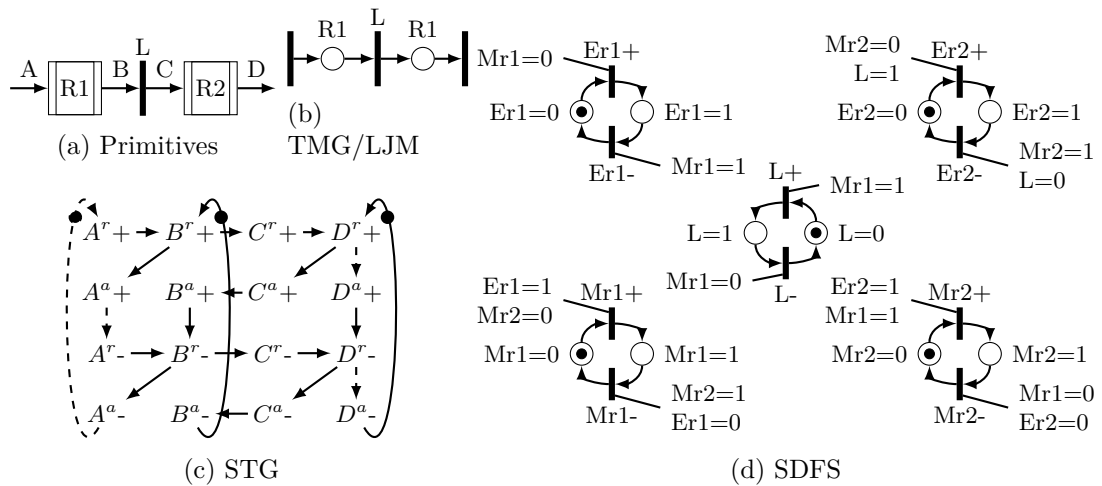


FIGURE 2.10 – Traduction de primitives en différents modèles formels.

Modèles ultra-compacts Dans un *Timed Marked Graph* (TMG) (CARMONA et al. 2009), les nœuds du graphe représentent un module de communication de type union puis fourche. Les arcs représentent les canaux de communication dans lesquels les données sont stockées. Le marquage du graphe représente la distribution des données.

Ce modèle est très compact, l'exemple de la FIGURE 2.10b en témoigne, mais son expressivité limitée ne permet pas de représenter les canaux sans mémorisation. En effet, dans le micropipeline par exemple (SUTHERLAND 1989), une donnée se propage en s'étalant à partir d'un *buffer* à travers un certain nombre de canaux jusqu'à ce qu'elle atteigne le prochain *buffer*.

Le modèle Liaisons/Articulations (*Links/Joints Model*) (LJM) (RONCKEN et al. 2015) suit la même logique que le TMG. Les liaisons (*Links*) sont en charge de la mémorisation et les articulations (*Joints*) synchronisent les données. Sutherland fait remarquer

que *Joint* désigne aussi en anglais un lieu où l'on se retrouve et interagit⁶, ce qui correspond bien à sa fonction. Moyennant cette abstraction et une interface standardisée, RONCKEN et al. (2015) montrent que les *buffers* des différentes familles peuvent naturellement communiquer entre eux.

Comme le montre la FIGURE 2.10b, pour cet exemple simple, le LJM et le TMG mènent à la même représentation. Néanmoins, contrairement au TMG, les articulations du LJM peuvent implémenter une fonction de synchronisation arbitrairement complexe. Cette fonction n'est cependant pas capturée par le modèle de graphe et nécessite une formalisation tierce.

Static Dataflow Structures (SDFSs) SOKOLOV, POLIAKOV et YAKOVLEV (2007) formalisent la sémantique d'étalement du jeton ainsi que d'autres types de jetons (anti-jeton et jeton contre-courant). Ils présentent des PN correspondant qui encodent explicitement la présence ou l'absence de données dans les canaux et les conditions de remplissage ou de vidage du canal. Le modèle utilise des arcs de lectures (CHRISTENSEN et HANSEN 1993) permettant de tester le remplissage d'une place sans la vider lorsque la transition tire. Ainsi, deux places par canal sont nécessaires (une pour chaque état plein ou vide). Sur l'exemple proposé de deux étages de pipeline (FIGURE 2.10a), le modèle SDFS (FIGURE 2.10d) a 10 places et 10 transitions pour 16 états tandis que le modèle STG (FIGURE 2.10c) a 21 places et 16 transitions pour 48 états.

Contribution à un nouveau modèle Dans la SECTION 4.2, nous introduisons un nouveau modèle à haut niveau basé sur les PN pour la spécification et la vérification des circuits à données groupées. Ce modèle, développé en collaboration avec Abdelkarim Cherkaoui (SIMATIC et al. 2017), propose une représentation compacte des canaux sans mémorisation en ne représentant explicitement que les frontières des jetons. Contrairement au SDFS (SOKOLOV, POLIAKOV et YAKOVLEV 2007) et au TMG (CARMONA et al. 2009), ce modèle inclut aussi les composants de sélection (multiplexeurs et démultiplexeurs).

2.2.3 Protocoles de communication

Un protocole est le raffinement de la convention de signalisation pour préciser l'ordonnancement relatif des événements dans les canaux de communication.

Chaque *buffer* est connecté à deux canaux : son canal d'entrée E par lequel le contrôleur reçoit des données et son canal de sortie S par lequel le contrôleur envoie les données. On rappelle que les canaux à données groupées sont constitués d'un chemin de données qui contient la logique combinatoire et de deux fils de *handshake* : une requête qui notifie l'arrivée de données valides et un acquittement qui indique qu'une nouvelle donnée peut être capturée.

6. Une traduction possible de « *joint* » est « *tripot* ».

Dans la suite, X^r et X^a désigneront respectivement les fils de requête et d'acquittement d'un canal X . Par ailleurs, « + » et « - » désignent toujours respectivement les fronts montant et descendant.

2.2.3.1 Signalisation des données valides

Une donnée est valide après la requête et avant l'acquittement. La requête est liée à un évènement sur le fil de requête (L^r+ ou L^r-) tandis que l'acquittement est lié à un évènement sur le fil d'acquittement (L^a+ ou L^a-). La FIGURE 2.11 indique les quatre conventions de validité possibles (PEETERS et BERKEL 1995).

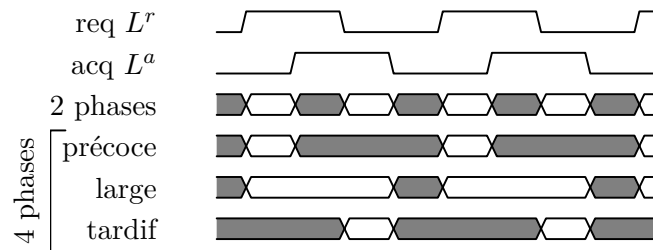


FIGURE 2.11 – Conventions de validité possibles (PEETERS et BERKEL 1995). Les données sont invalides dans les zones grises.

Sur le canal L , la séquence suivante de 4 évènements est répétée en continu : L^r+ , puis L^a+ , ensuite L^r- et enfin L^a- . Autrement dit, entre deux évènements sur un fil de *handshake* (fil de requête ou d'acquittement), il y a toujours un évènement sur le fil de *handshake* en sens inverse (fil d'acquittement ou de requête). Cet enchaînement garantit que le récepteur prend bien en compte chaque évènement.

La première convention est appelée 2 phases car chaque transfert de données nécessite deux transitions. Dans cette convention, L^r+ et L^r- indiquent tous deux la requête et L^a+ et L^a- indiquent tout deux l'acquittement.

Les autres conventions sont appelées 4 phases car chaque transfert de données requiert quatre transitions. Pour indiquer la requête, les conventions précoce et large utilisent L^r+ , tandis que la convention tardive utilise L^r- . Pour indiquer l'acquittement, la convention précoce utilise L^a+ , tandis que les conventions large et tardive utilisent L^a- . Notons que pour les circuits à données groupées utilisant des bascules comme registres, la position de l'acquittement n'influe pas. Les conventions précoce et large sont alors équivalentes.

Les circuits QDI utilisent nécessairement soit la convention 2 phases soit la convention précoce car le front montant du fil de requête L^r+ correspond exactement à l'arrivée d'une donnée valide avec l'encodage en 1-parmi- N . À l'inverse, toutes les conventions sont possibles pour les circuits à données groupées.

2.2.3.2 Environnements de travail sur les protocoles

Un protocole est donc le choix d'une convention de validité et de contraintes sur l'ordonnancement relatif des événements sur les canaux d'entrée (L) et de sortie (R) du *buffer*. Dans la suite, nous appelons protocole précoce (resp. 2 phases ou tardif) un protocole adoptant la convention de validité précoce (resp. 2 phases ou tardif).

Les travaux extensifs de la littérature sur les protocoles spécifient des contraintes assurant un protocole fonctionnel puis proposent une classification des nombreux protocoles possibles. FURBER et DAY (1996) imposent des séquences de base sur les signaux et présentent une série de protocoles, sans souci d'exhaustivité. LINES (1998) propose un ensemble de propriétés pour les protocoles QDI et étudie les 9 protocoles les satisfaisant. Du fait de la nature des protocoles QDI, son travail néglige les événements sur les données.

MCGEE et NOWICK (2005) ajoutent les événements sur les données pour modéliser les protocoles dynamiques aussi. Leur environnement est basé sur un protocole qui offre une concurrence maximale, c'est-à-dire contraignant le moins possible les événements tout en étant fonctionnel. Les autres protocoles sont alors obtenus à partir de ce protocole de concurrence maximale par des transformations réduisant la concurrence. MCGEE et NOWICK mettent ainsi en évidence la structure de treillis⁷ de l'ensemble des protocoles.

BIRTWISTLE et STEVENS (2008) adoptent la même démarche mais pour les protocoles statiques précoces. Ils définissent un protocole de concurrence maximale LC_{max} , construisent son graphe d'états et le représentent sur quatre lignes (FIGURE 2.12a). Les réductions de concurrence sont alors exprimées comme des coupes, c'est-à-dire la suppression d'états, dans ce graphe. Cette démarche permet d'identifier chaque protocole à l'aide de seulement 8 chiffres. Comme l'indique la FIGURE 2.12b, chaque chiffre correspond au nombre d'états supprimés à gauche (L pour *left*) ou à droite (R pour *right*) de chaque ligne.



(a) Graphe d'états du protocole de concurrence maximale. Les lignes sont numérotées de R1 à R4. Les o représentent les états et + l'état initial. (b) Graphe du protocole $L2002oR4264$. Chaque chiffre correspond au nombre d'états coupés (remplacés par des points) dans une ligne.

FIGURE 2.12 – Graphes d'états des protocoles dans l'environnement de BIRTWISTLE et STEVENS (2008).

Cet environnement est dédié aux protocoles précoces ou larges. Néanmoins, la même coupe peut désigner deux protocoles utilisant le même ensemble de contraintes sur l'or-

7. « Ensemble partiellement ordonné dans lequel chaque paire d'éléments admet une borne supérieure et une borne inférieure » (Wikipédia).

donnancement des évènements mais des conventions de validité différentes.

2.2.3.3 Protocoles tardifs existants

Les protocoles tardifs utilisent le front descendant du fil de requête d'entrée (L^r-) pour indiquer l'arrivée d'une donnée valide. À notre connaissance, trois protocoles tardifs ont été étudiés : le protocole des Cellules Universelles pour Séquences Asynchrones (CUSAs) de DAVID (1977), le protocole Burst-mode de YUN, BEEREL et ARCEO (1996), et Early-ack de MANNAKKARA et YONEDA (2010).

Ces trois protocoles appliquent la même réduction de concurrence sur le canal d'entrée L : le plus court chemin de L^r+ à R^r+ est $L^r+ \rightarrow L^a+ \rightarrow L^r- \rightarrow R^r+$. Cela correspond à la coupe à gauche $L2222$. Les trois protocoles présentent en revanche différent découplage et varient donc dans la coupe à droite. La FIGURE 2.13 présente les STGs de ces trois protocoles, décrits dans les paragraphes suivants.

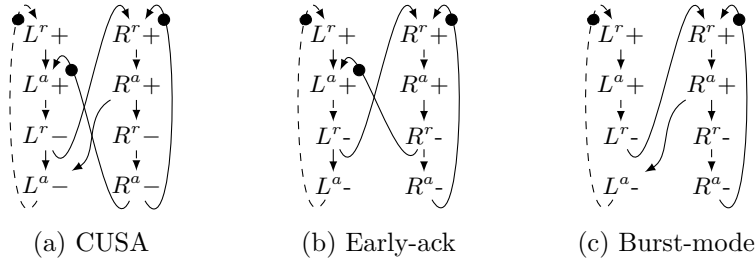


FIGURE 2.13 – Protocoles tardifs existants.

CUSA ($L2222 \circ R2244$) Comme le montre la FIGURE 2.13a, le protocole CUSA est le moins découplé, autrement dit le plus contraint, des trois. Il correspond en revanche au circuit le plus simple.

Early-ack ($L2222 \circ R0040$) Le protocole Early-ack (FIGURE 2.13b) découple la transition L^a- (en retirant la dépendance $R^a+ \rightarrow L^a-$) et augmente la concurrence (en remplaçant $R^a- \rightarrow L^a+$ par $R^r- \rightarrow L^a+$). Cette séquence nécessite toutefois une propagation rapide du front descendant de la requête, sinon une nouvelle donnée pourrait être acceptée avant que l'étage suivant ne lise la donnée courante. Le protocole Early-ack nécessite donc des retards asymétriques avec un court temps de propagation du front descendant.

Burst-mode ($L2222 \circ R2222$) Le protocole Burst-mode (FIGURE 2.13c) fait un choix de découplage symétrique d'Early-ack par rapport à CUSA. Le protocole Burst-mode retire la dépendance $R^a- \rightarrow L^a+$ et découple ainsi la transition L^a+ .

Contribution à deux nouveaux protocoles Dans la SECTION 4.1, deux nouveaux protocoles tardifs sont présentés et étudiés en comparaison avec des protocoles présentés ci-dessus. Comme l'indique la FIGURE 2.14, en termes de découplage, le protocole Late-capture est le symétrique du protocole Burst-mode par rapport au protocole CUSA et le protocole Maximus applique les deux découplages des protocoles Burst-mode et Late-capture. Notons que le protocole Maximus n'a pas d'équivalent dans l'environnement de coupes. Il peut toutefois s'exprimer avec une coupe à indices négatifs qui indiquent les états à rajouter au graphe de la FIGURE 2.12a.

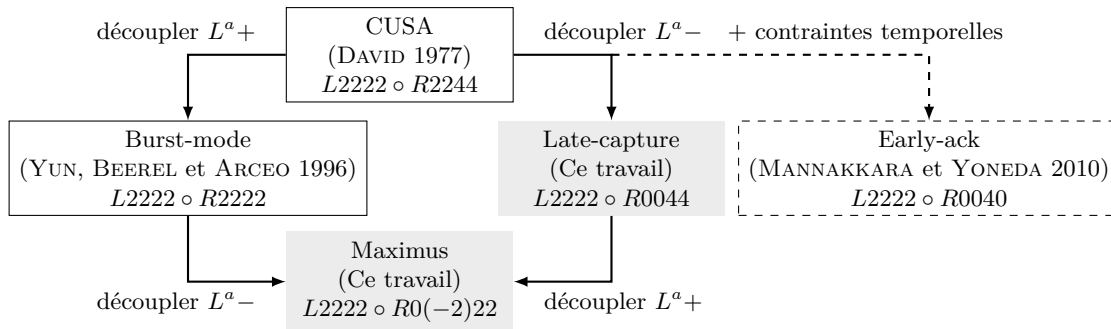


FIGURE 2.14 – Positionnement en termes de découplage des protocoles proposés par rapport aux protocoles tardifs existants.

2.3 Synthèse de haut niveau (HLS)

La HLS traite une description à haut niveau, c'est-à-dire un algorithme, sans informations d'architecture ni temporelles dans la description, pour produire une description synthétisable du matériel. De plus, la HLS est capable d'explorer les architectures possibles afin d'optimiser le circuit selon des critères de performances ajustables.

Grâce à de nombreux outils commerciaux et académiques (dont CatapultC, Forte's Cynthesizer, Vivado HLS, BlueSpec, GAUT et UGH, tous décrit par COUSSY et MORAWIEC (2008)), la HLS de circuits synchrones est un moyen attractif pour concevoir rapidement des circuits implémentant des algorithmes. En outre, la HLS a aussi pour but la vérification et la maintenabilité pour la conception systémique (COUSSY et al. 2009).

Pour les circuits asynchrones, les outils de HLS aident à combler le fossé entre les compétences des concepteurs de circuits et les techniques spécifiques aux circuits asynchrones. Dans les sections suivantes, nous présentons d'abord les techniques typiques de la HLS synchrone et l'architecture de circuit résultante, puis les méthodes et outils de HLS asynchrones.

2.3.1 Techniques et architecture typiques synchrones

Prenons comme exemple l'algorithme du plus grand commun dénominateur (PGCD) reproduit dans la FIGURE 2.15a. Remarquons que les outils (ici AUGH (PROST-BOUCLE, MULLER et ROUSSEAU 2014; *AUGH*)) proposent une syntaxe spécifique pour décrire les types matériels (convention de signe et nombre de bits) et des canaux communications (ici de type premier arrivé premier servi (*First-in First-out*) (FIFO)). L'outil est ensuite capable de générer le matériel pour gérer les transactions sur ces canaux.

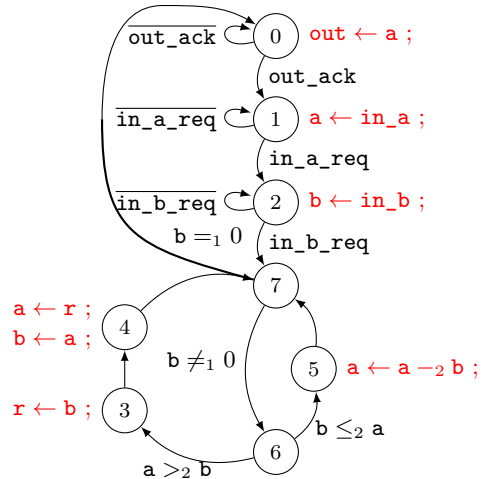
```

/* Interfaces FIFO and registres */
uint8_t in_a, in_b, out;
uint8_t a, b, r;

/* Fonction principale PGCD */
void main(){
  while(1){
    fifo_read(in_a, &a);
    fifo_read(in_b, &b);
    while(b > 0){
      if(a >= b){ a = a - b; }
      else{ r = b; b = a; a = r; }
    }
    fifo_write(out, &a);
  }
}

```

(a) Code source (C).



(b) Graphe de la FSM.

FIGURE 2.15 – L'algorithme du PGCD avant et après HLS. Les instructions de la FSM sont indiquées en rouge et les conditions en noir. Les opérations sont numérotées selon l'opérateur physique correspondant.

Après une analyse syntaxique, la HLS comporte trois étapes principales :

1. L'*allocation* du nombre d'opérateurs arithmétiques de chaque type (multiplicateur, additionneur, comparateur) et de ressources mémoires qui seront nécessaires au calcul.
2. L'*ordonnement* des opérations en respectant les relations de dépendance, la limite du nombre d'opérateurs et le temps de calcul de chaque opération. Dans le cas synchrone, ce temps est un multiple de la période d'horloge. L'ordonnement détermine pour chaque cycle d'horloge quelles seront les opérations réalisées.
3. L'*assignation* qui précise pour chaque cycle quelle variable sera stockée dans quelle mémoire et quelle opération sera effectuée sur quel opérateur.

Dans le cas du PGCD, la FIGURE 2.15b montre le résultat de la HLS après ces trois étapes et avant la génération de la description matérielle. L'allocation a requis deux opérateurs que l'on a numérotés, à savoir le comparateur (n°1) et un soustracteur (n°2), et trois registres, à savoir *a*, *b* et *r*. L'ordonnement mène à 8 états numérotés de 0 à

7 : les états 0 à 2 gèrent la réception des opérandes et l'envoi du résultat tandis que les états 3 à 7 réalisent le calcul à proprement parler.

La formulation du problème de HLS en ces trois sous-problèmes interdépendants mais distincts permet de limiter la complexité de la résolution globale. Cette décomposition a pour conséquence une architecture typique de la HLS composée d'une partie de calcul (chemin de données) et une machine à états finis (*Finite State Machine*) (FSM) pour la partie de contrôle (COUSSY et al. 2009).

Cette architecture est représentée par la FIGURE 2.16. La partie calcul contient les registres pour mémoriser les données, les opérateurs pour effectuer les calculs et des multiplexeurs pour acheminer les opérandes depuis les registres vers les opérateurs et à nouveau vers les registres.

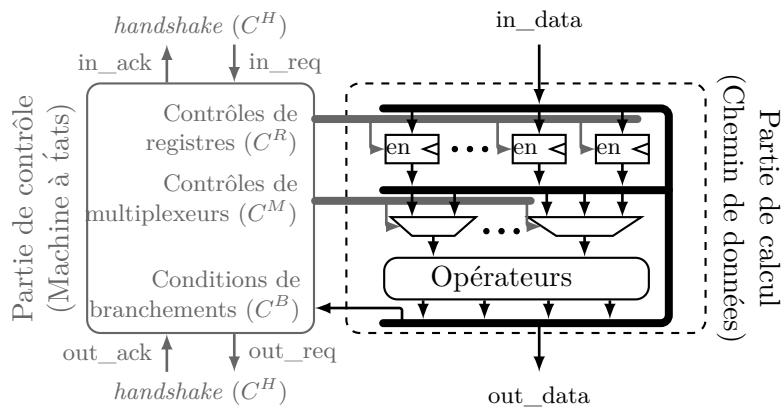


FIGURE 2.16 – Architecture résultante de la HLS synchrone. Les circuit se décompose en partie de contrôle (à gauche) et partie de calcul (à droite).

La partie de contrôle est une FSM qui gère quatre types de signaux de contrôle :

1. Les *contrôles de registre* (C^R) déclenchent la mise à jour des registres. Ces contrôles sont connectés aux signaux *enable* des bascules.
2. Les *contrôles de multiplexeur* (C^M) sélectionnent les entrées des multiplexeurs.
3. Les *conditions de branchement* (C^B) sont des valeurs dont dépend le prochain état de la FSM. Dans la FIGURE 2.15b par exemple, le successeur de l'état 6 dépend de la comparaison entre **a** et **b**.
4. Les *signaux de handshake* (C^H) indiquent la présence de nouvelles donnée (requêtes) et leur bonne réception (acquittements).

Les défis actuels de la HLS synchrone incluent les circuits pipelines, la gestion de chemins multi-cycles et l'exploration de l'espace de conception au niveau système. Pour les circuits asynchrones, la première difficulté est la réalisation de la FSM pour la partie contrôle.

2.3.2 Machines à états finis asynchrones (AFSM)

Les architectures classiques synchrones de FSM, à savoir les machines de Mealy et de Moore, possèdent un registre d'état (codé sur plusieurs bits) qui est mis à jour à chaque cycle d'horloge. La nouvelle valeur de l'état dépend de l'état actuel et des entrées de la FSM.

Dans le cas des machines à états asynchrones (*Asynchronous Finite State Machines*) (AFSMs), pour éviter des problèmes liés aux bits du registre d'état changeant de façon désordonnée, deux mécanismes ont été proposés. Le premier est un bit de complétion (CHENG et al. 2014) qui indique la fin de la mise à jour de l'état et donc que les sorties peuvent être calculées sans danger. Le deuxième est le principe d'AFSM localement synchrone (NOWICK et DILL 1991; CURTINHAS et al. 2014) qui génère en fonction des changements des valeurs des entrées et des sorties de l'AFSM une horloge locale pour mettre à jour le registre d'états. Ces architectures centralisées sont compactes mais peu adaptées au paradigme asynchrone.

À l'inverse, les architectures distribuées (DAVID 1977; HOLLAAR 1982; SOTIRIOU 2001; SHANG, XIA et YAKOVLEV 2002; SAITO et al. 2010) contiennent au moins un élément de mémorisation par état de l'AFSM (encodage *one-hot*). Ainsi, l'état de l'AFSM n'est pas définie par la valeur d'un registre d'états mais par quel élément de mémorisation est actif à ce moment.

Les éléments de mémorisation basés sur des contrôleurs de registres asynchrones (HOLLAAR 1982; SOTIRIOU 2001; SHANG, XIA et YAKOVLEV 2002) permettent de réaliser des pipelines plus naturellement et plus efficacement que ceux basés sur des séquenceurs (bascules Q) (SAITO et al. 2010). Les protocoles tardifs sont intéressants car ils permettent le non-recouvrement des activations des états et améliorent la consommation et la surface des pipelines avec des longs étages de calcul (SIMATIC et al. 2016; OLIVEIRA et al. 2016).

2.3.3 Techniques et outils de HLS asynchrones

Les outils de HLS asynchrones existants suivent l'une des trois stratégies décrites dans les paragraphes suivants.

2.3.3.1 Traduction dirigée par la syntaxe

La stratégie la plus simple possible est d'utiliser un HDL dédié et de traduire les expressions du code en macro-composants matériels. Les outils de synthèse asynchrone TiDE (NIELSEN et al. 2009) et Balsa (BARDSLEY 1998) utilisent respectivement les HDLs de haut niveau Haste et Balsa. Dans ces flots de conception, le processus de synthèse traduit la syntaxe du langage en *Handshake components* (BERKEL 1993). La qualité du circuit généré par cette traduction dirigée par la syntaxe dépend considérablement de l'aptitude du concepteur à écrire le code HDL de façon convenable. Des stratégies d'optimisations telles que la traduction à partir d'un langage plus général (TRACHERO

et al. 2009) peut atténuer ce problème.

2.3.3.2 Décomposition en processus pipelinés

Les outils de HLS utilisent des représentations intermédiaires de l'algorithme pour abstraire et extraire la fonction décrite de façon aussi indépendante que possible de la syntaxe. La décomposition dirigée par les données (*Data-Driven Decomposition*) (DDD) (WONG et MARTIN 2003) et le compilateur CASH du flot *Pegasus* (VENKATARAMANI et al. 2004) utilisent respectivement les formes dynamique et statique d'assignation unique (ANANIAN 1999). Ces approches sont adaptées pour des applications nécessitant un fort débit car elles génèrent des circuits fortement pipelinés. Néanmoins, elles sont peu appropriées pour les applications typiques de l'IoT. En effet, celles-ci requièrent des circuits à basse consommation et peu chers qui peuvent s'accommoder d'une faible vitesse.

2.3.3.3 Flots basés sur l'ordonnancement

La décomposition du cœur de la HLS en allocation, ordonnancement et assignation permet d'optimiser le processus de synthèse selon des critères ajustables. Les méthodes pour résoudre ces problèmes diffèrent selon l'objectif et les contraintes de l'optimisation. Toutefois, la décomposition en parties de calcul et de contrôle qui en découle pose deux difficultés dans le cas des circuits asynchrones.

La première, la réalisation de l'AFSM, a été discutée dans la SECTION 2.3.2. La seconde difficulté est la continuité de l'échelle de temps pour l'ordonnancement. En conséquence, l'ordonnancement asynchrone optimal peut être différent du cas synchrone. HANSEN et SINGH (2010) ont proposé un algorithme efficace qui résout cette question pour deux objectifs d'optimisation : la minimisation de la latence sous contrainte de surface et la minimisation de surface sous contrainte de latence.

À notre connaissance, seul le flot BUDASYN (GARCIA 2015) intègre une solution aux deux problèmes. L'architecture de l'AFSM est localement synchrone (centralisée) et générée à partir d'une spécification XBM (YUN et DILL 1999). L'allocation et l'ordonnancement utilisent l'algorithme de (HANSEN et SINGH 2010).

La FIGURE 2.17 représente graphiquement les différentes approches asynchrones présentées. Par rapport au flot BUDASYN, nous préférons utiliser des outils synchrones de HLS, plus matures, pour le flot ALPS. Seule la partie de contrôle est remplacé par une AFSM. Par rapport à la désynchronisation (CORTADELLA et al. 2006), la HLS permet de disposer d'informations à plus haut niveau sur la fonctionnalité du circuit et donc faciliter l'obtention d'un circuit plus efficace, notamment par l'utilisation de composants de sélection (multiplexeurs et démultiplexeurs) dans le contrôle.

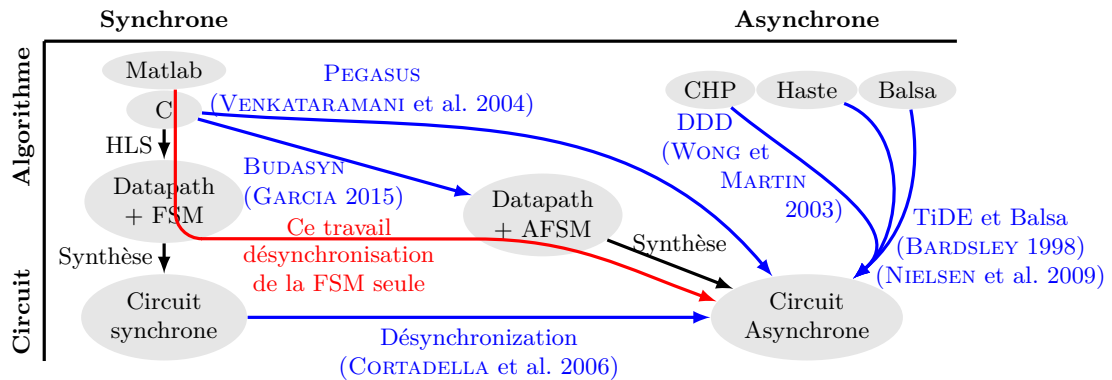


FIGURE 2.17 – Stratégies existantes de HLS asynchrone.

2.4 Conclusion

Dans ce chapitre, nous avons présenté les techniques d'échantillonnage non uniforme, leurs intérêts pour la réduction du nombre d'échantillons. Les opérations de fenêtrage et de filtrage ont illustré le besoin de nouveaux algorithmes et circuits. La synthèse de haut niveau (HLS) permet d'automatiser la traduction d'un programme en matériel. La HLS de circuit synchrone nécessite trois étapes principales, à savoir l'allocation, l'ordonnancement et l'assignation, et mène à une architecture matérielle composée d'une partie de contrôle et d'une partie de calcul (FSM). La principale difficulté de la HLS de circuits asynchrones est la réalisation de cette partie de calcul (AFSM). Nous proposons une solution à ce problème dans le CHAPITRE 3.

Nous avons également défini les conventions de validité des données et les protocoles dans les circuits asynchrones. Les protocoles tardifs sont apparus comme largement moins étudiés par rapport aux protocoles précoces ou 2 phases. Le CHAPITRE 4 étudie les protocoles tardifs. Il introduit aussi un nouveau modèle pour les circuits asynchrones qui répond à la problématique de la modélisation compacte de circuits asynchrones avec des structures conditionnelles.

Le flot de conception ALPS

Sommaire

3.1	ALPS : un flot général pour la conception de systèmes basse consommation basés sur des évènements	35
3.1.1	Vue d'ensemble du flot	35
3.1.2	Choix de l'échantillonnage et de l'algorithme	36
3.1.3	Synthèse de haut niveau	36
3.1.4	Réalisation physique	36
3.1.5	Validation et vérification	37
3.2	ALPS-HLS : Synthèse de haut niveau asynchrone	37
3.2.1	Architecture cible et chemins de données	37
3.2.2	Désynchronisation de la partie de contrôle	38
3.2.2.1	Contrôleurs d'états pour l'activation des chemins	39
3.2.2.2	Contrôleurs de démultiplexeurs et multiplexeurs	40
3.3	ALPS-RTC : Contraintes temporelles et insertion de retards	42
3.3.1	Points de contrôle de temps de l'AFSM	42
3.3.2	Contraintes temporelles de l'AFSM	43
3.3.2.1	Formulation avec les points de contrôle de temps	43
3.3.2.2	Reformulation pour les outils de conception	45
3.3.3	Méthode de résolution	46
3.4	Études de cas	48
3.4.1	Choix du schéma d'interpolation pour l'ASA	48
3.4.2	Dimensionnement d'un filtre RIF non uniforme	51
3.4.2.1	Application de test : électrocardiogramme	51
3.4.2.2	Comparaison de trois circuits RIF	52
3.4.2.3	Performances : surface, temps de calcul et énergie	55
3.4.2.4	Conclusion de l'étude de cas	58
3.5	Conclusion	58

La principale contribution de cette thèse est le flot de conception ALPS pour des systèmes exploitant l'échantillonnage non uniforme et l'électronique asynchrone.

Ce chapitre dédié à ALPS s'organise de la façon suivante : nous présentons d'abord la vue d'ensemble du flot puis nous intéresserons à deux étapes clés, à savoir la HLS asynchrone et les contraintes temporelles relatives (*Relative Timing Constraints*) (RTCs) associées.

3.1 ALPS : un flot général pour la conception de systèmes basse consommation basés sur des évènements

Le flot ALPS vise la conception, la synthèse et l'évaluation de circuits basse consommation. Pour ce faire, ALPS utilise deux paradigmes évènementiels : l'échantillonnage non uniforme réalisé par un CAN-TN et l'électronique asynchrone utilisée par la partie de traitement.

3.1.1 Vue d'ensemble du flot

Le flot de conception ALPS est divisé en trois phases principales présentées par la FIGURE 3.1 et détaillées dans les paragraphes suivants.

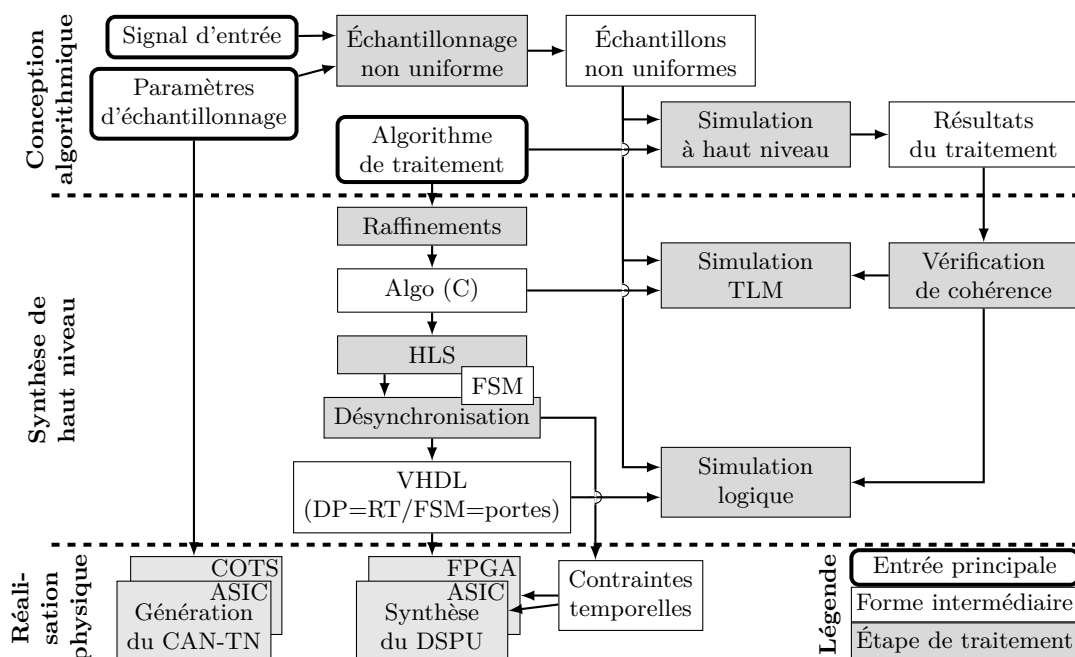


FIGURE 3.1 – Le flot de conception ALPS. Il propose d'une part la conception itérative de l'algorithme et la sélection des paramètres d'échantillonnage. D'autre part, la HLS génère le matériel asynchrone.

3.1.2 Choix de l'échantillonnage et de l'algorithme

Tout d'abord, le concepteur choisit les paramètres d'échantillonnage, à savoir les positions des seuils et la période du compteur de temps. Il conçoit aussi l'algorithme de traitement. Cet algorithme est testé à haut niveau avec le flot d'échantillons NUS résultant des paramètres d'échantillonnage et des signaux caractéristiques de l'application visée.

Les étapes de conception algorithmiques utilisent la bibliothèque Matlab existante SPASS (FESQUET et BIDÉGARAY-FESQUET 2010b) ou son équivalent Python en cours de développement pySPASS (SIMATIC et BIDÉGARAY-FESQUET 2017). Ces boîtes à outils rassemblent les fonctions classiques pour l'échantillonnage et le traitement non uniformes.

3.1.3 Synthèse de haut niveau

Ensuite, la phase de HLS génère la description au niveau des transferts de registres (*Register Transfer Level*) (RTL) de l'unité de traitement numérique du signal (*Digital Signal Processing Unit*) (DSPU) à partir de l'algorithme. La partie contrôle du DSPU (c'est-à-dire la FSM) est désynchronisée de manière spécifique par traduction directe des éléments du graphe d'états de la FSM vers des contrôleurs asynchrones. Cette AFSM confère un comportement asynchrone à tout le circuit. Notons que la méthode de désynchronisation de la FSM en AFSM est applicable dès que l'architecture du DSPU est décomposée en parties de contrôle et de calcul. Ainsi, il est théoriquement possible d'utiliser différents outils de HLS. En pratique, nous avons automatisé le processus pour l'outil académique AUGH (PROST-BOUCLE 2014) et avons réussi à désynchroniser manuellement des FSMs produites par l'outil industriel CatapultC.

3.1.4 Réalisation physique

Finalement, la description de circuit au niveau portes logiques est obtenue en synthétisant le DSPU et en générant le CAN-TN. La synthèse du DSPU suit un flot standard auquel sont ajoutées des contraintes temporelles spécifiques à vérifier.

Selon l'architecture décrite sur la FIGURE 2.2, le CAN-TN requiert deux CNAs et deux comparateurs analogiques. Pour la conversion analogique/numérique de la sortie des comparateurs, il faut mettre un filtre de métastabilité telle la porte « SAMPLE » (KHOMENKO et al. 2017) pour permettre d'avoir une sortie numérique propre. Ces composants sont réalisés avec des composants sur étagère (*Components On The Shelf*) (COTS) pour un cible sur *field-programmable gate array* (FPGA) ou dimensionnés selon la méthode d'ALLIER et al. (2005) pour une cible *application specific integrated circuit* (ASIC). Les compteurs de niveaux, de temps et la mémoire en lecture seule (*Read Only Memory*) (ROM) sont générés en fonction des paramètres choisis¹. Ils suivent un flot classique de conception numérique FPGA ou ASIC.

1. Cette ROM peut être remplacée par une mémoire à accès aléatoire (*Random Access Memory*) (RAM) pour permettre la reconfiguration en ligne des seuils.

3.1.5 Validation et vérification

Dans toutes les phases décrites précédemment, le flot ALPS inclut des étapes de simulation. Au niveau algorithmique, la simulation permet d'évaluer la fonctionnalité du traitement et de générer un résultat de référence. La simulation du modèle au niveau transactionnel (*Transaction Level Model*) (TLM), permet de vérifier la cohérence entre les résultats produits à haut niveau (Matlab ou Python) et ceux produits par la version de l'algorithme en C (compatible avec la HLS). La simulation logique vérifie le fonctionnement de la DSPU synthétisée. Les transitions des fils sont aussi comptées pour permettre une meilleure estimation de l'énergie consommée en pondérant la contribution de chaque fil par sa capacité et son nombre de transitions.

Si la consommation énergétique n'est pas satisfaisante, le concepteur modifie le code C et relance la HLS. Si elle est satisfaisante, le flot se termine par les étapes classiques de placement et routage.

3.2 ALPS-HLS : Synthèse de haut niveau asynchrone

3.2.1 Architecture cible et chemins de données

L'architecture cible du flot ALPS (FIGURE 3.2) est identique à celle de la HLS synchrone car elle en est issue. Pour la partie contrôle, la FSM est remplacée par une AFSM. Quant à la partie calcul, elle reste inchangée hormis la connexion des bascules : les signaux de contrôle de registre (C^R) ne sont plus connectés aux signaux *enable* mais directement aux signaux d'horloges.

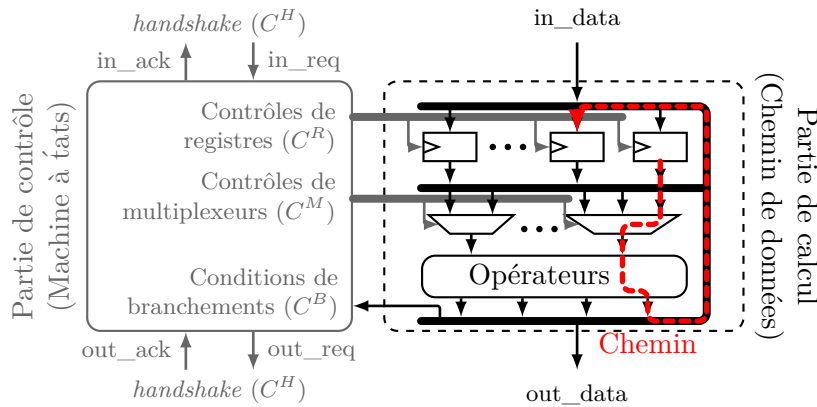


FIGURE 3.2 – Architecture cible du flot ALPS. Les multiplexeurs au centre de la partie calcul orientent les données dans les chemins : depuis les registres en haut vers les opérateurs en bas puis à nouveau vers les registres.

La FIGURE 3.2 introduit aussi le concept de chemin. Un chemin correspond aux portes et aux fils traversés par une donnée lorsqu'elle est utilisée dans un calcul. Ainsi,

3.2.2.1 Contrôleurs d'états pour l'activation des chemins

Dans chaque état, l'AFSM active un ensemble de chemins dans la partie calcul en actionnant des contrôles de multiplexeurs C^M et de registres C^R . Les données se propagent alors le long des chemins activés comme indiqué sur la FIGURE 3.2.

L'AFSM doit idéalement commencer à activer les chemins d'un des états que lorsqu'elle a fini de désactiver les chemins de l'état précédent. Sinon, des conflits entre les chemins risquent de corrompre les données.

Les protocoles tardifs retardent la transmission des requêtes aux états suivants. Ils facilitent donc la séparation des activations des états en offrant 5 événements successifs avant la transmission de la requête : $M+$, $R+$, $A+$, $M+$ et $R-$. Un état est alors défini comme actif si la condition $M + R$ est vérifiée. Un état est alors activé par $M+$ et désactivé par $R-$.

Nous définissons les contraintes suivantes pour les signaux de contrôle :

1. *Établissement* : Pour laisser les données se propager le long du chemin, la capture des données doit arriver avec un retard de δ par rapport à l'activation des multiplexeurs : $C^M+ \xrightarrow{\delta} C^R+$. Il s'agit d'une contrainte d'établissement (*setup*).
2. *Maintien* : La désactivation des multiplexeurs doit succéder à la capture des données : $C^R+ \rightarrow C^M-$. Il s'agit d'une contrainte de maintien (*hold*).
3. *Activation* : L'activation de l'état doit précéder l'activation des multiplexeurs : $M+ \rightarrow C^M+$.
4. *Désactivation* : Pour limiter le chevauchement, la désactivation des multiplexeurs doit précéder la désactivation de l'état : $C^M- \rightarrow R-$.

Les contraintes d'*établissement* et de *maintien* sont des conditions nécessaires. Les contraintes d'*activation* et de *désactivation* sont suffisantes.

La contrainte d'*établissement* requiert d'insérer des retards pour garantir la couverture du chemin de données. Cela va augmenter la surface et la consommation du circuit. Nous choisissons $M+ = C^M+$ pour relâcher la contrainte d'*établissement*. De même, on associe C^R+ à $R-$ pour relâcher la contrainte d'*établissement* et limiter le temps mort avant l'activation de l'état suivant. Le choix précédent associé aux contraintes de *maintien* et de *désactivation* impose d'associer C^M- à $R-$. Le chronogramme résultant est indiqué dans la FIGURE 3.4a. Les flèches pointillées montrent l'association entre les événements d'un protocole tardif (sur L , M , R et A) et les événements générés pour la partie calcul (sur C^M et C^R).

Le défaut de cette association est qu'elle tend à violer la contrainte de *maintien* lorsque le contrôle de registres C^R active beaucoup de bascules. Pour retarder C^M- sans avoir à insérer des retards, nous choisissons d'associer C^M- avec $A-$ plutôt qu'avec $R-$. Néanmoins, les hypothèses temporelles à vérifier pour éviter les interférences entre les états sont plus contraintes (voir SECTION 3.3). Cette nouvelle association est réalisée par la logique du contrôleur d'états de la FIGURE 3.3 : $C^M = M + A$ et $C^R = \bar{R}$.

Parmi les protocoles tardifs existants, les protocoles CUSA et Burst-mode ne sont pas

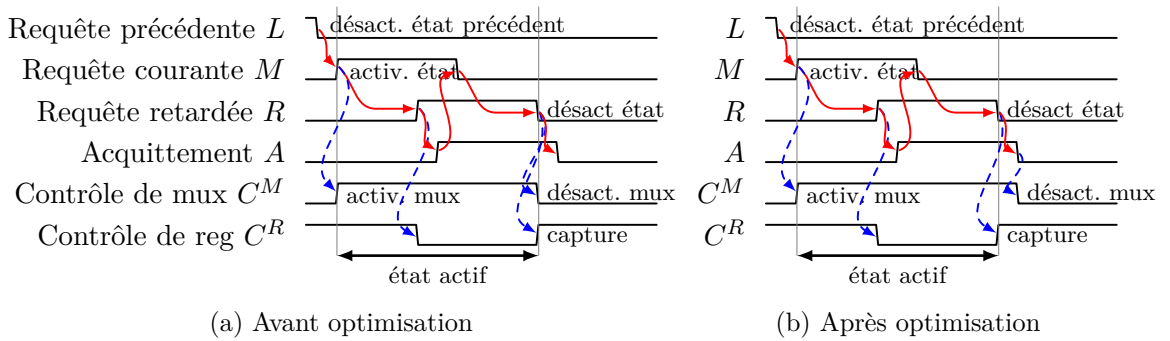


FIGURE 3.4 – Chronogramme de l’activation d’un état. Les dépendances du protocole sont indiquées par des flèches pleines. Les flèches pointillées relient les événements du protocole avec les événements de contrôle de la partie calcul.

efficaces pour cette nouvelle association car la dépendance $R^a+ \rightarrow L^a-$ implique que le front descendant du fil d’acquiescement du canal d’entrée (L^a-) est envoyé après le front montant du fil d’acquiescement sur le canal de sortie (R^a+). De ce fait, le chevauchement est important (au moins $\delta/2$). Le protocole Early-ack nécessite des délais asymétriques qui font perdre l’intérêt en surface des protocoles tardifs (voir SECTION 4.1). En revanche, le nouveau protocole tardif Late-capture convient ².

Envoi et réception de données à l’extérieur du circuit La réception et l’envoi de données avec l’extérieur du circuit nécessite de gérer les signaux de *handshake* (C^H). Ces signaux peuvent être gérés, comme dans le cas synchrone, comme des signaux classiques de contrôle (C^M ou C^R) pour les C^H sortants et des signaux de branchement (C^B) pour les C^H entrants. Par exemple pour envoyer une donnée, la FSM boucle sur l’état d’envoi jusqu’à ce que l’acquiescement *out_ack* soit reçu.

Une AFSM offre la possibilité d’arrêter l’exécution pendant l’attente et ne reprendre pour passer à l’état suivant que lorsque l’acquiescement est reçu. Ainsi, le temps d’attente ne génère pas d’activité dans le circuit.

Les FIGURES 3.5a et 3.5b montrent les structures des contrôleurs d’états modifiés respectivement pour envoyer et recevoir les données. Dans chacun des cas, une seule porte de Muller a été ajoutée.

3.2.2.2 Contrôleurs de démultiplexeurs et multiplexeurs

Les contrôleurs de démultiplexeurs et de multiplexeurs correspondent respectivement au début et à la fin de structures conditionnelles dans l’algorithme. Ces deux contrôleurs nécessitent de trouver une structure logique adaptée aux protocoles tardifs.

². Il s’agit d’ailleurs de la motivation qui nous a mené à le trouver.

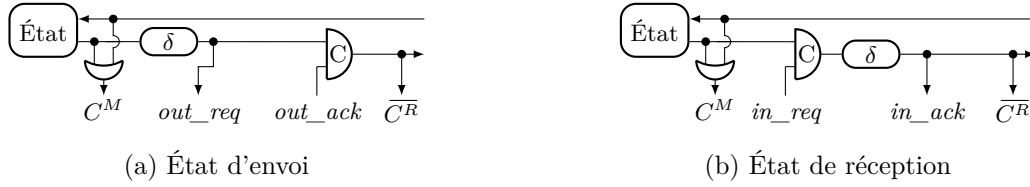


FIGURE 3.5 – Structures des contrôleurs d'états modifiées pour gérer efficacement les transferts avec l'environnement.

Démultiplexeurs Un démultiplexeur est inséré lorsqu'un état a plusieurs successeurs. À la fin de l'activation de l'état courant, un des successeurs est choisi selon la valeur de la condition de branchement C^B . La FIGURE 3.6a montre la structure en portes logiques du démultiplexeur. À cause du protocole tardif, la condition n'est pas encore valide lorsque le fil de requête monte ($R+$). Ainsi, les démultiplexeurs dédiés aux protocoles précoces (SPARSØ et FURBER 2002 ; YAHYA 2009) ne conviennent pas aux protocoles tardifs. Pour les protocoles tardifs, dans notre travail, le démultiplexeur fait suivre la requête montante à tous ses successeurs et attend les acquittements de tout le monde. La condition ne peut être lue en toute sécurité qu'après la chute de la requête. Le démultiplexeur fait alors descendre la requête de la seule branche sélectionnée par la condition.

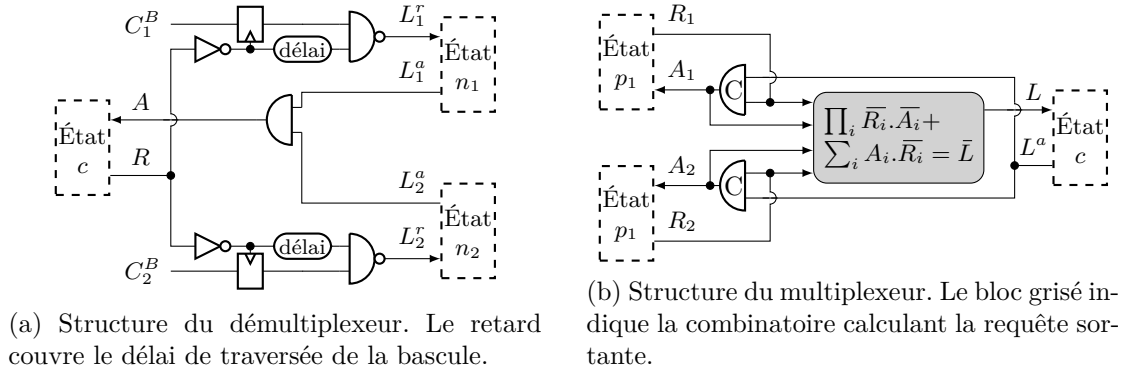


FIGURE 3.6 – Réalisation logique des composants de sélection.

Multiplexeurs Un multiplexeur est inséré lorsqu'un état a plusieurs prédécesseurs, comme montré dans la FIGURE 3.6b. Les acquittements d'entrée A_i utilisent une porte de Muller (M2), comme dans les protocoles précoces (SPARSØ et FURBER 2002 ; YAHYA 2009). Pour la requête de sortie L , il n'est pas possible d'utiliser la disjonction (OU) de toutes les fils de requêtes d'entrées R_i . En effet, les démultiplexeurs font monter les fils de requêtes de toutes leurs branches de sorties. Ainsi, les R_i ne sont pas mutuellement exclusives et L doit valoir 0 si et seulement si tous les canaux sont en attente ($\bar{R}_i \cdot \bar{A}_i$ pour tout i) ou s'il y a un front descendant sur l'une des requêtes ($A_i \cdot \bar{R}_i$ pour un i).

3.3 ALPS-RTC : Contraintes temporelles et insertion de retards

Les circuits asynchrones à données groupées doivent respecter la convention de données groupées : le signal de requête indiquant la validité d'une donnée doit atteindre sa destination après l'arrivée effective de la donnée. Cette contrainte temporelle est relative (*Relative Timing Constraint*) (RTC) puisque l'acheminement de la requête doit être plus rapide que celui de la donnée. Cette section décrit les RTCs requises pour garantir un comportement correct des circuits générés dans ALPS et comment nous ajustons les retards dans le contrôleur pour les vérifier.

Une RTC est formellement composée de trois points de contrôle de temps (MANORANJAN et STEVENS 2016) : 1) un point de divergence pod , 2) un point de convergence avancé poc_0 et 3) un point de convergence retardé poc_1 . L'équation (3.1) décrit une RTC. Elle se lit : « Lorsqu'un évènement (front montant ou descendant) survient au point de divergence pod , ses répercussions doivent atteindre le point de convergence avancé poc_0 au moins un temps δ avant d'atteindre le point de convergence retardé poc_1 . »

$$pod \mapsto poc_0 \stackrel{\delta}{\prec} poc_1 \quad (3.1)$$

Si les contraintes ne concernent que certains fronts, $pod+$ et $pod-$ indiquent respectivement les fronts montant et descendant de pod .

3.3.1 Points de contrôle de temps de l'AFSM

Pour chaque état c , nous définissons les points de contrôle de temps comme indiqués dans la FIGURE 3.7. Soient les points de contrôle suivants :

C_c La sortie de la requête du *buffer* c .

M_c Le point de divergence entre le contrôle de multiplexeur C^M et la requête sortant du *buffer* C_c .

R_c Le point de divergence entre la requête retardée et le contrôle de registre C^R .

A_c Le signal d'acquiescement arrivant des étages suivant.

B_c Le point de divergence entre l'acquiescement et C^M .

Un état c peut contrôler plusieurs sources d'horloges locales pour les registres et plusieurs sources d'activation pour les multiplexeurs. Nous définissons donc pour ces sources des indices distincts de l'indices des états :

C_i^M Une des sources d'activation pour les multiplexeurs.

C_j^R Une des sources d'horloge locale pour les registres.

R_j^D Les données entrantes des bascules activés par C_j^R

R_j^Q Les données sortantes des bascules activées par C_j^R

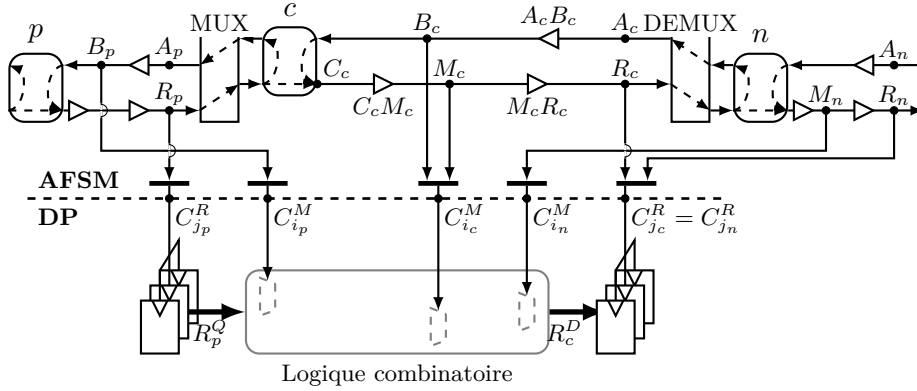


FIGURE 3.7 – Points de contrôle de temps dans l'AFSM. Les flèches représentent les arcs temporels. Les triangles sont les endroits où des retards peuvent être insérés pour résoudre les contraintes relatives temporelles.

Notons que les bascules des démultiplexeurs (FIGURE 3.6a) sont traitées de la même façon que les bascules de la partie calcul. Pour faire le lien entre les états et les sorties de l'AFSM, nous définissons les trois types d'ensemble suivants :

\mathbb{S} L'ensemble des états de l'AFSM.

\mathbb{M}_c L'ensemble des contrôles de multiplexeurs activés dans l'état c . $i \in \mathbb{M}_c$ si et seulement si C_i^M est activé par M_c .

\mathbb{R}_c L'ensemble des contrôles de registres activés dans l'état c . $j \in \mathbb{R}_c$ si et seulement si C_j^R est activé par R_c .

3.3.2 Contraintes temporelles de l'AFSM

Cette section présente les RTCs requises par les circuits générés dans ALPS. Nous utilisons c , n et p pour désigner respectivement les états courant, suivant (*next*) et précédent (*previous*). Les indices i et j désignent respectivement des contrôles de multiplexeurs C_i^M et de registres C_j^R .

3.3.2.1 Formulation avec les points de contrôle de temps

Contraintes de stabilité des multiplexeurs Les signaux de contrôle de multiplexeur $C_{i_c}^M$ doivent rester à l'état haut pendant toute la durée de l'activité de l'état c . En particulier, le front descendant de M_c ne doit pas entraîner de front descendant de $C_{i_c}^M$. On a donc la contrainte $(MS_{c,i_c})^3$.

$$\forall c \in \mathbb{S}, \forall i_c \in \mathbb{M}_c, \quad A_c+ \mapsto C_{i_c}^M+ \prec C_{i_c}^M- \quad (MS_{c,i_c})$$

3. MS pour *mux stability*.

Contraintes de largeur des créneaux des horloges locales À l'intérieur des bascules, les transferts de données entre les verrous (*latches*) maître et esclave doivent avoir suffisamment de temps pour s'exécuter correctement. Ainsi, les parties hautes et basses des créneaux sur les signaux d'horloges doivent être plus larges que deux constantes notées respectivement w_+ et w_- et données par la technologie et les cellules de bascules.

Pour la partie basse, le front descendant du contrôle de registres (C_c^R-) doit arriver w_- avant le front montant (C_c^R+). Le point de divergence est la montée de la requête retardée (R_c+)⁴. On a donc la contrainte (NPW $_{c,j_c}$)⁵.

$$\forall c \in \mathbb{S}, \forall j_c \in \mathbb{R}_c, \quad R_c+ \mapsto C_{j_c}^R- \stackrel{w_-}{\prec} C_{j_c}^R+ \quad (\text{NPW}_{c,j_c})$$

Pour la partie haute, la FIGURE 3.7 montre comment deux états successifs peuvent partager un même contrôle de registre ($C^R = C_{j_c}^R = C_{j_n}^R$). Dans ce cas, le front descendant de l'horloge (C^R-) produit par l'état suivant (n) doit être séparé d'au moins w_+ du front montant (C^R+) produit par l'état courant (c). Le point de divergence est la chute de la requête retardée (R_c-). On a donc la contrainte (PPW $_{c,n,j}$)⁶.

$$\forall c, n \in \mathbb{S}^2, \forall j \in \mathbb{R}_c \cap \mathbb{R}_n, \quad R_c- \mapsto C_j^R+ \stackrel{w_+}{\prec} C_j^R- \quad (\text{PPW}_{c,n,j})$$

Contraintes de maintien (*hold*) La chute de la requête retardée (R_c-) déclenche la capture de la donnée ($C_{j_c}^R+$) et désactive les multiplexeurs de l'état courant c ($C_{i_c}^M-$). Afin de respecter le temps de maintien t_h , les conséquences des événements sur les multiplexeurs doivent se propager aux registres après celles de $C_{j_c}^R+$. On a donc la contrainte (HC $_{c,i_c,j_c}$)⁷.

$$\forall c \in \mathbb{S}, \forall i_c \in \mathbb{M}_c, \forall j_c \in \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_{i_c}^M \text{ à } R_{j_c}^D, \quad R_c- \mapsto C_{j_c}^R+ \stackrel{t_h}{\prec} R_{j_c}^D \quad (\text{HC}_{c,i_c,j_c})$$

C'est cette contrainte qui est relâchée par l'optimisation choisie en SECTION 3.2.2.1. En effet, $C_{i_c}^M-$ arrive plus tard donc la contrainte est plus facile à vérifier.

On remarque aussi que la désactivation de l'état courant (R_c-) va aussi activer les multiplexeurs des états suivants ($C_{i_n}^M+$). On a donc la contrainte (HN $_{c,n,i_n,j_c}$)⁸.

$$\forall c, n \in \mathbb{S}^2, \forall i_n \in \mathbb{M}_n, \forall j_c \in \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_{i_n}^M \text{ à } R_{j_c}^D, \quad R_c- \mapsto C_{j_c}^R+ \stackrel{t_h}{\prec} R_{j_c}^D \quad (\text{HN}_{c,n,i_n,j_c})$$

4. Rappelons que C_c^R est connecté à R_c par un inverseur (FIGURE 3.3)

5. NPW pour *negative pulse width*.

6. PPW pour *positive pulse width*.

7. HC pour *hold current*.

8. HN pour *hold next*.

Contraintes d'établissement (*setup*) Les données doivent être stables aux entrées des registres au moins pendant un temps t_s avant leur capture. t_s est le temps d'établissement.

Clairement, une fois que les multiplexeurs de l'état c sont actifs ($C_{i_c}^M+$), les données se propagent et doivent avoir le temps d'arriver aux registres avant leur capture par le front montant du contrôle de registres ($C_{j_c}^R+$). On a donc la contrainte (SM_{c,i_c,j_c})⁹.

$$\forall c \in \mathbb{S}, \forall i_c \in \mathbb{M}_c, \forall j_c \in \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_{i_c}^M \text{ à } R_{j_c}^D, \quad M_c+ \mapsto R_{j_c}^D \stackrel{t_s}{\prec} C_{j_c}^R+ \\ (\text{SM}_{c,i_c,j_c})$$

De façon moins évidente, les données peuvent aussi se propager depuis les registres qui ont été mis à jour par l'un des états précédents p . On a donc la contrainte (SD_{p,c,j_p,j_c})¹⁰.

$$\forall p, c \in \mathbb{S}^2, \forall j_p \in \mathbb{R}_p, \forall j_c \in \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } R_{j_p}^Q \text{ à } R_{j_c}^D, \quad R_p- \mapsto C_{j_p}^R+ \stackrel{t_s}{\prec} C_{j_c}^R+ \\ (\text{SD}_{p,c,j_p,j_c})$$

Finalement, si certains multiplexeurs dans le chemin courant étaient actifs dans l'un des états précédents p , leur désactivation ($C_{i_p}^M-$) doit avoir le temps de se propager. On a donc la contrainte (SP_{p,c,i_p,j_c})¹¹.

$$\forall p, c \in \mathbb{S}^2, \forall i_p \in \mathbb{M}_p, \forall j_c \in \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_{i_p}^M \text{ à } R_{j_c}^D, \quad R_p- \mapsto C_{i_p}^M- \stackrel{t_s}{\prec} C_{j_c}^R+ \\ (\text{SP}_{p,c,i_p,j_c})$$

C'est cette contrainte qui est resserrée par le retardement de C^M- choisi en SECTION 3.2.2.1.

3.3.2.2 Reformulation pour les outils de conception

Le formalisme de l'équation (3.1) est peu adapté aux outils standards d'analyse temporelle qui manipulent des délais plutôt que des contraintes relatives¹². En conséquence, nous reformulons dans l'équation (3.2) les contraintes relatives temporelles de l'AFSM comme des inégalités entre deux sommes de délais : le délai de propagation du chemin de pod à poc_0 plus une marge δ doit être plus court que le délai de propagation du chemin de pod à poc_1 .

$$\|pod \rightarrow poc_0\| + \delta < \|pod \rightarrow poc_1\| \quad (3.2)$$

Les délais des chemins sont alors décomposés en sommes de délais. Dans la suite, XY désigne le délai du chemin $X \rightarrow Y$. Les délais minimum et maximum sont mesurés par analyse statique de timing (*Static Timing Analysis*) (STA).

9. SM pour *setup mux*.

10. SD pour *setup direct*.

11. SP pour *setup previous*.

12. Sauf dans le cas précis des contraintes d'horloge dont la spécification relative permet de relâcher les contraintes sur l'arbre d'horloge (*skew*).

Pour chaque couple d'états $c, n \in \mathbb{S}^2$, nous avons besoin de calculer le délai $R_c C_n$ entre la désactivation de l'état c (R_c-) et la prochaine activation de l'état n (C_n+). Si n est un successeur direct de c , alors le délai de R_c à travers les éventuels démultiplexeurs et multiplexeurs vers la sortie C_n du *buffer* de n est mesuré directement par STA.

Si n n'est pas un successeur direct de c , il y a plusieurs chemins possibles entre c et n . Soit $P = c \rightarrow s_1 \rightarrow \dots \rightarrow s_{|P|-1} \rightarrow s_{|P|} = n$ un de ces chemins. Les s_i sont les états intermédiaires et $|P|$ la longueur du chemin. On étend alors la définition du délai $R_c C_n$ au cas non direct, noté $R_c C_n$, pour inclure les délais de l'exécution de tous les états intermédiaires.

$$R_c C_n = \min_P \left[R_c C_{s_1} + \sum_{i=1}^{|P|-1} \left(\begin{array}{l} 2C_{s_i} M_{s_i} + 2M_{s_i} R_{s_i} + R_{s_i} A_{s_i} \\ + A_{s_i} B_{s_i} + B_{s_i} C_{s_i} + R_{s_i} C_{s_{i+1}} \end{array} \right) \right] \quad (3.3)$$

Notons que ce délai est utilisé par des contraintes impliquant certaines sorties de contrôle. Soit une contrainte et $E_n \in \mathbb{M}_n \cup \mathbb{R}_n$ un signal de contrôle de l'état n d'intérêt pour cette contrainte. Si l'un des états s_i actionne E_n alors la contrainte entre c et s_i couvrira celle entre c et n . Il en va de même pour un $E_c \in \mathbb{M}_c \cup \mathbb{R}_c$. Il est donc pratique de pouvoir restreindre $R_c C_n$ aux chemins P dans lesquels aucun état intermédiaire n'active un des contrôles d'un ensemble E donné¹³. On notera ce temps $R_c C_n^{\setminus E}$.

Les RTCs reformulées sont données dans la FIGURE 3.8.

3.3.3 Méthode de résolution

Afin de vérifier les RTCs, des retards, représentés par de petits répéteurs (triangles) sur la FIGURE 3.7, doivent être insérés. Il y a trois retards par état, à savoir $A_c B_c$, $C_c M_c$ et $M_c R_c$. Le calcul des tailles de retards requises comporte trois étapes décrites ci-dessous.

1) Ouvrir les boucles temporelles Une boucle temporelle est un circuit fermé d'arcs temporels. Ces boucles posent un problème pour la STA car l'outil estime que des oscillations mènent à un temps maximal de propagation non borné. En réalité, on s'intéresse au délai maximal entre deux oscillations successives. Pour ouvrir les boucles temporelles, certains arcs temporels des contrôleurs d'états, de multiplexeurs et de démultiplexeurs sont coupés de sorte que :

- Les répéteurs sont des points d'ouverture des boucles.
- Les contrôleurs de multiplexeurs et de démultiplexeurs ne fassent que faire suivre les requêtes et les acquittements. Ils se comportent donc comme des fils.
- Les contrôleurs d'état se comportent normalement à l'exception des acquittements entrants qui ne sont pas propagés sur les acquittements sortants.

Les arcs temporels restants sont représentés par des flèches pointillées dans la FIGURE 3.7.

13. En pratique, $E \subset (\{\emptyset\} \cup \mathbb{M}_c \cup \mathbb{R}_c) \times (\{\emptyset\} \cup \mathbb{M}_n \cup \mathbb{R}_n)$.

$$\begin{aligned}
 & \forall c \in \mathbb{S}, \forall i \in \mathbb{M}_c, \quad B_c C_i^M < B_c C_c + C_c M_c + M_c C_i^M && (\text{MS}_{c,i}) \\
 & \forall c \in \mathbb{S}, \forall j \in \mathbb{R}_c, \quad w_- + R_c C_j^R < R_c A_c && (\text{NPW}_{c,j}) \\
 & \quad \quad \quad + A_c B_c + B_c C_c + C_c M_c + M_c R_c + R_c C_j^R \\
 & \forall c, n \in \mathbb{S}^2, \forall j \in \mathbb{R}_c \cap \mathbb{R}_n, && (\text{PPW}_{c,n,j}) \\
 & \quad \quad \quad w_+ + R_c C_j^R < R_c C_n^{\setminus \{j\}} + C_n M_n + M_n R_n + R_n C_j^R \\
 & \forall c \in \mathbb{S}, \forall i, j \in \mathbb{M}_c \times \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_i^M \text{ à } R_j^D, && (\text{HC}_{c,i,j}) \\
 & \quad \quad \quad t_h + R_c C_j^R < R_c A_c + A_c B_c + B_c C_i^M + C_i^M R_j^D \\
 & \forall c, n \in \mathbb{S}^2, \forall i, j \in \mathbb{M}_n \times \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_i^M \text{ à } R_j^D, && (\text{HN}_{c,n,i,j}) \\
 & \quad \quad \quad t_h + R_c C_j^R < R_c C_n^{\setminus \{i,j\}} + C_n M_n + M_n C_n^M + C_i^M R_j^D \\
 & \forall c \in \mathbb{S}, \forall i, j \in \mathbb{M}_c \times \mathbb{R}_c \text{ t.q. } \exists \text{ chemin de } C_i^M \text{ à } R_j^D, && (\text{SM}_{c,i,j}) \\
 & \quad \quad \quad t_s + M_c C_i^M + C_i^M R_j^D < M_c R_c + R_c A_c \\
 & \quad \quad \quad + A_c B_c + B_c C_c + C_c M_c + M_c R_c + R_c C_j^R \\
 & \forall c, n \in \mathbb{S}^2, \forall i, j \in \mathbb{M}_c \times \mathbb{R}_n \text{ t.q. } \exists \text{ chemin de } C_i^M \text{ à } C_j^R, && (\text{SP}_{c,n,i,j}) \\
 & \quad \quad \quad t_s + R_c A_c + A_c B_c + B_c C_i^M + C_i^M R_j^D < R_c C_n^{\setminus \{i,j\}} + C_n M_n + M_n R_n + R_n A_n \\
 & \quad \quad \quad + A_n B_n + B_n C_n + C_n M_n + M_n R_n + R_n C_j^R \\
 & \forall c, n \in \mathbb{S}^2, \forall j_c, j_n \in \mathbb{R}_c \times \mathbb{R}_n \text{ t.q. } \exists \text{ chemin de } R_{j_c}^Q \text{ à } R_{j_n}^D, && (\text{SD}_{c,n,j_c,j_n}) \\
 & \quad \quad \quad t_s + R_c C_{j_c}^R + R_{j_c}^Q R_{j_n}^D < R_c C_n^{\setminus \{j_c,j_n\}} + C_n M_n + M_n R_n + R_n A_n \\
 & \quad \quad \quad + A_n B_n + B_n C_n + C_n M_n + M_n R_n + R_n C_{j_n}^R
 \end{aligned}$$

FIGURE 3.8 – Reformulation des contraintes temporelles avec des sommes de délais. Les délais $A_c B_c$, $C_c M_c$ et $M_c R_c$ correspondant aux retards doivent être choisis pour vérifier les équations. Les autres délais sont fixés.

2) Mesurer et contraindre les chemins sans retards Pour éviter trop d'itérations de placements et routages, les chemins où aucune insertion de retards n'est prévue sont contraints. Ainsi, les délais des chemins suivants sont mesurés après la synthèse et contraints autour de la valeur mesurée : $(R_c \rightarrow C_j^R)$, $(M_c \rightarrow C_i^M)$, $(B_c \rightarrow C_i^M)$, $(R_c \rightarrow A_c)$, $(B_c \rightarrow C_c)$ et $(R_c \rightarrow C_n)$. La partie calcul est contrainte de façon similaire : $(R_j^Q \rightarrow R_j^D)$ et $(C_i^M \rightarrow R_j^D)$.

3) Résoudre les contraintes par programmation linéaire Avec ces conditions, le choix de la taille des retards correspond au problème d'optimisation linéaire de la FIGURE 3.9.

Trouver	$ \begin{aligned} A_c B_c &\in [0, +\infty[\\ C_c M_c &\in [0, +\infty[\quad \text{for all } c \in \mathbb{T} \\ M_c R_c &\in [0, +\infty[\end{aligned} \tag{3.4} $
Minimisant	$ \text{Surface} = \sum_{c \in \mathbb{T}} A_c B_c + C_c M_c + M_c R_c \tag{3.5} $
Avec les contraintes de la FIGURE 3.8.	

FIGURE 3.9 – Formulation du problème d'optimisation du dimensionnement des retards.

Notons que la minimisation de la surface (3.5) peut être remplacée par la minimisation du temps total d'exécution (3.6). Les poids w_c sont alors l'estimation de la fréquence des états durant le calcul.

$$\text{Latence} \approx \sum_{c \in \mathbb{T}} w_c \cdot (A_c B_c + 2C_c M_c + 2M_c R_c)
 \tag{3.6}$$

Finalement, les retards sont ajustés dans le circuit selon la solution trouvée. Si l'outil de synthèse logique n'arrive pas à respecter les contraintes, on peut soit redéfinir les contraintes autour du nouveau circuit soit élargir les marges des contraintes et résoudre à nouveau le problème.

3.4 Études de cas

3.4.1 Choix du schéma d'interpolation pour l'ASA

Le problème est le choix d'un schéma d'interpolation pour le ré-échantillonnage de signaux NUS vers un échantillonnage uniforme. Le processus de ré-échantillonnage (FIGURE 3.10) parcourt la fenêtre W^i et produit des échantillons aux instants $(t_{rs})_{n \in \mathbb{N}}$ alors

que les échantillons d'entrée ont $(t_s)_{n \in \mathbb{N}}$ comme instant d'échantillonnage. Il faut donc interpoler le signal.

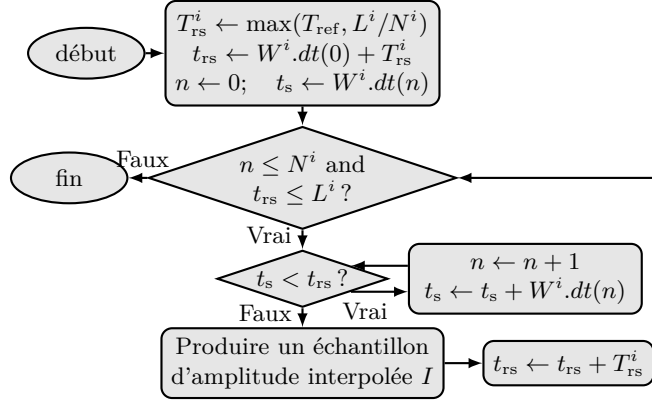


FIGURE 3.10 – Processus de ré-échantillonnage de l'ASA. T_{rs}^i est la période de ré-échantillonnage de la fenêtre i . t_{rs} est l'instant auquel le prochain échantillon de sortie doit être calculé. t_s est le n^e instant d'échantillonnage du signal NUS. I dépend du schéma d'interpolation.

Pour le choix de l'échantillonnage, on se limite aux schémas d'ordre inférieur ou égal à 1, c'est-à-dire ne prenant en compte que 2 échantillons d'entrée (ceux juste avant et après) pour calculer l'amplitude de sortie. Ces schémas sont décrits dans la TABLE 3.1.

TABLE 3.1 – Définitions et couts grossiers des schémas d'interpolation d'ordre inférieur ou égal à 1 (WAELE et BROERSEN 2000; QAISAR et al. 2016).

Schéma	Nom complet	Équation de l'amplitude interpolée I	Cout abstrait
ZOH	Constant d'ordre zéro (<i>Zero Order Hold</i>)	$W^i.a(n-1)$	0
NN	Plus proche voisin (<i>Nearest Neighbour</i>)	$\begin{cases} W^i.a(n-1) & \text{if } t_s - t_{rs} > \frac{W^i.dt(n)}{2} \\ W^i.a(n) & \text{otherwise} \end{cases}$	$2\oplus$
SLI	Linéaire simplifié (<i>Simplified Linear</i>)	$\frac{W^i.a(n) + W^i.a(n-1)}{2}$	\oplus
LIN	Linéaire	$W^i.a(n) - \frac{(W^i.a(n) - W^i.a(n-1))(t_s - t_{rs})}{W^i.dt(n)}$	$3\oplus + \otimes + \odot$

Nous utilisons ALPS pour évaluer le compromis entre la complexité du ré-échantillonnage et la qualité du signal reconstruit. Notamment, nous cherchons une estimation de la complexité du calcul en termes de surface et de consommation du circuit plutôt qu'une complexité abstraite (TABLE 3.1).

Le signal de test est composé de trois impulsions constituées à partir d'une superposition de deux sinusoïdes (une à basse fréquence et une à haute fréquence). L'équation (3.7) décrit les trois parties actives qui couvrent 25% du signal et donnent une bande passante $[f_{\min}, f_{\max}] = [50 \text{ Hz}; 1 \text{ kHz}]$.

$$x(t) = \begin{cases} 0,5 \sin(2\pi 50t) + 0,4 \sin(2\pi 1000t) & \text{si } 0,5 \text{ s} < t < 1 \text{ s} \\ 0,45 \sin(2\pi 70t) + 0,45 \sin(2\pi 150t) & \text{si } 4 \text{ s} < t < 5 \text{ s} \\ 0,6 \sin(2\pi 60t) + 0,3 \sin(2\pi 100t) & \text{si } 6 \text{ s} < t < 7 \text{ s} \\ 0 & \text{sinon} \end{cases} \quad (3.7)$$

Pour l'estimation de la consommation, le code C définit par une macro `INTERP_SCHEME` et des directives préprocesseur pour passer facilement d'un schéma à l'autre. Les différents circuits sont générés par synthèse haut niveau guidée par l'utilisateur automatisée (*Automated User Guided HLS*) (AUGH), désynchronisés et simulés pour estimer la consommation.

La TABLE 3.2 présente les résultats des estimations de l'erreur, de la surface et de la consommation. Les résultats de LIN (grisés) sont extrapolés à partir des autres (et de leurs équivalents synchrones) car ALPS ne supporte pas encore les appels de fonction générés par AUGH.

TABLE 3.2 – Surface et consommation en fonction du schéma d'interpolation. L'erreur RMS est normalisée par rapport à l'erreur de l'échantillonnage NUS. DL correspond à la mémoire des échantillons. DP\DL correspond au reste du chemin de données (registres intermédiaires et opérateurs combinatoires).

Schéma	Erreur	Surface (μm^2)				Consommation (pJ/échant.)			
		AFSM	DP\DL	DP	Total	AFSM	DP\DL	DL	Total
ZOH	1,15	0,77	1,06	18,5	20,3	1,63	3,24	22,3	27,2
SLI	1,30	0,80	1,10	19,0	20,9	1,60	3,50	24,7	29,8
NN	1,27	0,94	1,32	18,5	20,7	2,00	4,20	23,5	29,7
LIN	1,19	1,21	1,35	19,1	21,7	8,12	13,5	78,8	100,4

Les erreurs sont définies par rapport à l'erreur introduite par l'échantillonnage NUS et après une reconstruction par interpolation linéaire. Le schéma de ré-échantillonnage ZOH introduit le moins d'erreur supplémentaire. Cela est probablement dû à l'hystérésis. Les schémas NN et SLI sont très proches, le SLI étant de 0,3% à 2% moins bon. Enfin le schéma LIN présente une grosse AFSM et une grosse consommation des chemins de données à cause des opérations de multiplication et de division pour chaque échantillon. En revanche, l'erreur est proche du ZOH.

3.4.2 Dimensionnement d'un filtre RIF non uniforme

Notre second cas d'étude est un filtre passe-bas d'ordre 32. Il s'agit de comparer les gains apportés par l'échantillonnage non uniforme et par notre méthode de désynchronisation par rapport à un échantillonnage classique ou des circuits synchrones.

Nous déterminons d'abord les dimensions requises par les RIFs uniformes et non uniformes dans le cas d'un électrocardiogramme. Puis, nous présentons les architectures des filtres comparés et leurs surfaces et consommations respectives.

3.4.2.1 Application de test : électrocardiogramme

Le signal à filtrer est un électrocardiogramme (ECG) de la base de donnée PTB (BOUSSELJOT, KREISELER et SCHNABEL 1995 ; GOLDBERGER et al. 2000). La dynamique d'entrée est $\Delta V = 32$ mV (amplitudes entre ± 16 mV). L'*American Health Association* (AHA) (KLIGFIELD et al. 2007) recommande une fréquence de coupure de 150 Hz pour le filtre. Comme le filtrage n'est pas réalisé sur un signal de longueur infinie, l'AHA recommande une fréquence d'échantillonnage $F_{Nyq} = 500$ Hz. Afin de limiter l'erreur en amplitude, la résolution en amplitude de la conversion analogique/numérique est de 16 bits.

Les seuils de l'échantillonnage non uniforme doivent être choisis de manière à produire suffisamment d'échantillon pour reconstruire le signal. Dans le cas des signaux NUS, la fréquence *moyenne* d'échantillonnage doit être supérieure au double de la fréquence maximale (GRÖCHENIG 1992). On se concentre sur les complexes QRS du signal qui durent dans notre cas 100 ms en moyenne. Sur cette durée, l'échantillonnage doit produire plus de 50 échantillons afin d'obtenir une fréquence moyenne au dessus de $F_{Nyq} = 500$ Hz. Comme la dénivelée cumulée de l'amplitude sur un complexe est 2,7 mV, le quantum entre deux seuils doit être $q < 54$ μ V. Si les seuils sont placés uniformément¹⁴, ce quantum correspond à 593 seuils (9,2 bits). Nous choisissons d'encoder les seuils sur 9 bits ($q = \Delta V/2^9 = 62,5$ μ V).

Dans le cas de l'échantillonnage TN, le bruit de quantification ne dépend que de la précision du compteur de temps dans le CAN-TN (ALLIER et al. 2005). Le rapport signal sur bruit (*Signal-to-Noise Ratio*) (SNR) dû à la quantification dépend seulement de la position des seuils et est donné par l'équation (3.8). $P(x)$ est la puissance moyenne du signal x .

$$\text{SNR}_{\text{dB}} = 10 \log \left(\frac{3P(x)}{P\left(\frac{dx}{dt}\right)} \right) + 20 \log (f_{\text{cnt}}) \quad (3.8)$$

Dans le cas d'une quantification uniforme à 16 bits, le SNR est donné par la formule classique $\text{SNR}_{\text{dB}} = 6.02 \times 16 + 1.761 = 98,08$ dB (IFEACHOR et JERVIS 2002). Pour obtenir

14. Ce choix peut s'expliquer du fait que le signal à analyser comporte une composante quasi-continue de peu d'importance pour le complexe QRS. Le placement uniforme des seuils est le moins variant par translation du signal.

un SNR similaire, la fréquence du compteur de temps doit vérifier $f_{\text{cnt}} \geq 1,25 \text{ MHz}$. Par ailleurs, le compteur de niveaux doit pouvoir suivre les variations du signal. Pour des niveaux sur 9 bits et une borne de bande passante d'entrée $f_{\text{max}} = 150 \text{ Hz}$, la condition de suivi (ALLIER et al. 2005) impose $f_{\text{cnt}} \geq 2\pi(2^M - 1)f_{\text{max}} = 0,5 \text{ MHz}$. Ainsi, pour vérifier les deux conditions, la fréquence du compteur de temps $f_{\text{cnt}} = 1,25 \text{ MHz}$ est choisie.

Le choix de la profondeur du compteur (nombre de bits) est un compromis entre la taille de chaque échantillon et le nombre d'échantillons supplémentaires produits par le dépassement de la capacité du compteur de temps. La TABLE 3.3 indique qu'en dessous d'une profondeur de 17 bits, la réduction marginale¹⁵ de la taille des échantillons est plus faible que l'augmentation marginale du nombre d'échantillons.

TABLE 3.3 – Nombre d'échantillons produits par le dépassement du compteur de temps en fonction de la profondeur du compteur. Sans dépassement, le CAN-TN produit 8036 échantillons. Les variations de taille et nombre sont calculées par rapport à la ligne précédente.

Profondeur du compteur (bit)	Échantillons supplémentaires	Taille des échantillons	Nombre d'échantillons
20	0		
19	0	-5%	+0%
18	144	-5.3%	+1.8%
17	314	-5.6%	+2.1%
16	985	-5.9%	+8.0%
15	2606	-6.3%	+17%
14	6559	-6.7%	+44%

La FIGURE 3.11 montre le signal échantillonné uniformément et non uniformément. Sur la durée totale du signal, l'échantillonnage uniforme produit 57 600 échantillons contre seulement 8340 dans le cas non uniforme.

3.4.2.2 Comparaison de trois circuits RIF

Maintenant que nous avons les paramètres d'échantillonnage, nous présentons les architectures destinées à traiter les flots d'échantillons.

15. La variation marginale d'une quantité V est le rapport $V_M(n) = \frac{V(n+1) - V(n)}{V(n)}$.

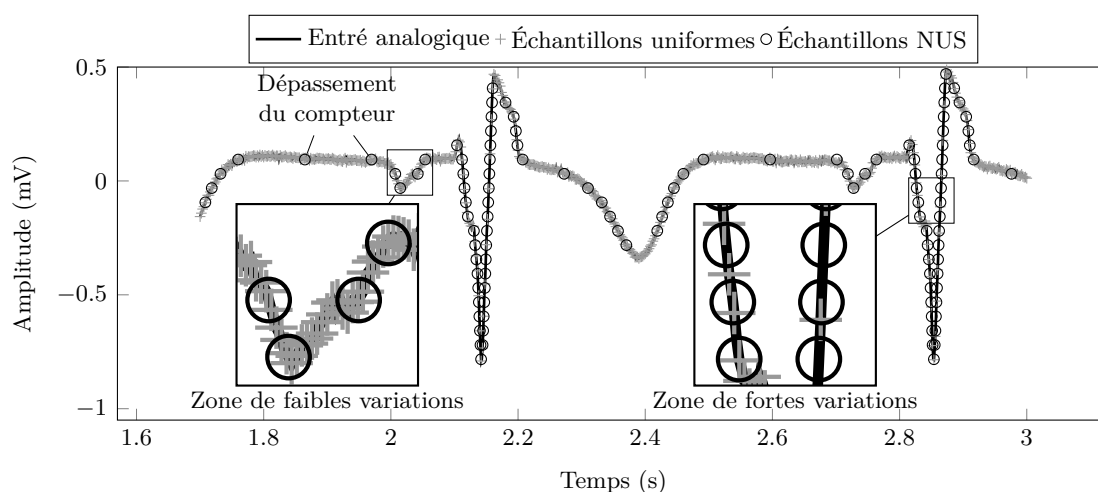


FIGURE 3.11 – Signaux de test analogique et échantillonnés. Dans les zones de variations rapides (le complexe QRS), il y a autant d'échantillons NUS qu'uniformes. Mais dans les zones de variations lentes, la traversée de niveaux élimine les échantillons redondants.

Filtre RIF synchrone conçu manuellement

L'architecture typique des filtres RIF synchrones, composée de N registres, N multiplieurs et $N - 1$ additionneurs, est couteuse en surface de silicium. Pour réaliser le calcul décrit par l'équation (2.6), ce travail utilise une architecture séquentielle plus réduite avec seulement un multiplieur et un accumulateur comme l'indique la FIGURE 3.12.

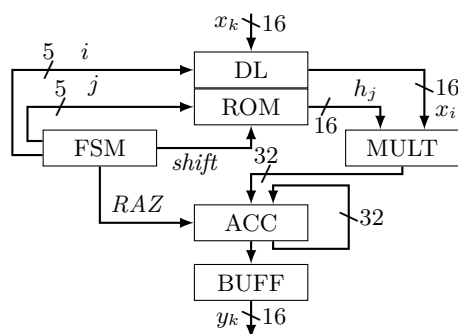


FIGURE 3.12 – Architecture d'un filtre RIF uniforme synchrone optimisé en surface.

Filtre RIF interpolé conçu manuellement

Ce filtre est une version 4 phases à données groupées de l'algorithme RIFI. Il se compose donc de chemins de données similaires au cas synchrone (notamment avec des bascules) (FIGURE 3.12) manœuvrés par un contrôleur asynchrone. Le protocole utilisé

est le protocole Simple (FURBER et DAY 1996)¹⁶.

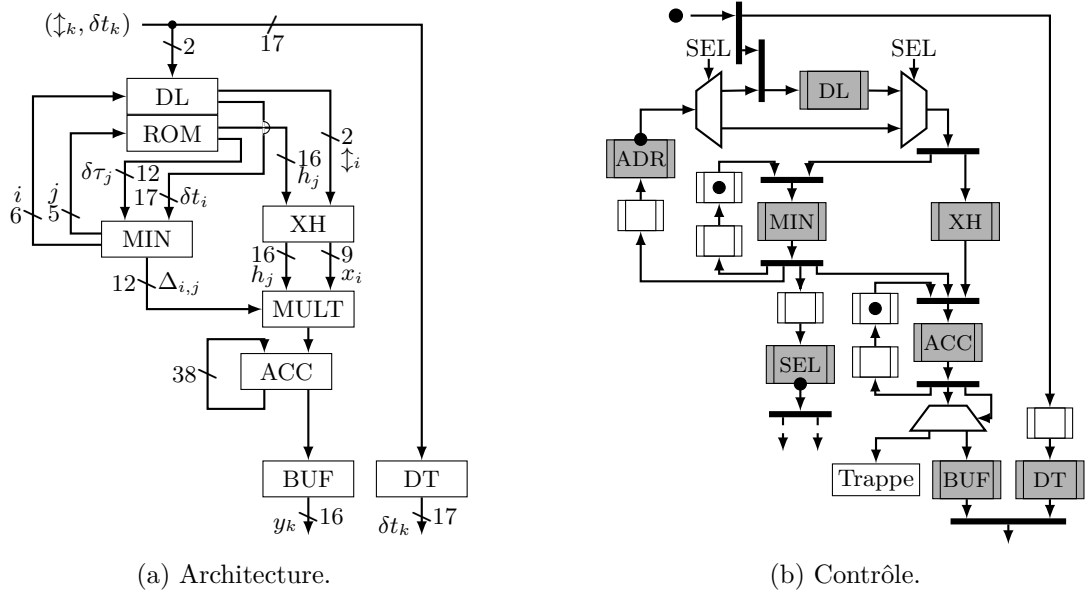


FIGURE 3.13 – Conception manuelle du RIFI. Les *buffers* grisés sont connectés aux registres tandis que ceux blancs permettent d’éviter les blocages.

Un registre à décalage (bloc DL) mémorise les échantillons. Ce bloc contient aussi un registre pour stocker l’amplitude du dernier échantillon \tilde{x}_0 . Ce registre est incrémenté ou décrétementé au moment de la réception d’un nouvel échantillon. La mémoire (bloc ROM) contient les amplitudes et délais des coefficients h .

Le bloc MIN doit déterminer le plus petit délai $\Delta_{i,j} = \min(\delta t_i, \delta t_j)$, incrémenter i ou j en conséquence et indiquer la fin de l’algorithme. Le bloc XH intègre les directions des échantillons (\downarrow_i) de 0 à i pour calculer l’amplitude \tilde{x}_i . Comme les seuils sont distribués uniformément, \tilde{x}_i est encodé sur 9 bits.

Le bloc MULT calcule le produit des amplitudes \tilde{x}_i et h_j par le délai minimum $\Delta_{i,j}$. Les résultats intermédiaires sont accumulés par le bloc ACC. Finalement, le résultat y_k et le délai correspondant δt_k sont enregistrés par BUF et DT.

La FIGURE 3.13b montre le contrôle. Les symboles des primitives sont ceux de la TABLE 2.1.

Filtre RIF interpolé conçu automatiquement

Nous avons utilisé les outils de HLS AUGH et CatapultC pour générer des versions synchrones des filtres RIFI pour les signaux NUS. Les circuits synchrones générés par HLS sont ensuite désynchronisés par ALPS.

16. On remarquera la majuscule qui identifie la dénomination du protocole. Ce protocole est aussi connu sous le nom de « WCHB » (SEITZ 1980) ou « micropipeline 4 phases » (SUTHERLAND 1989).

AUGH L'algorithme de la FIGURE 2.4b est écrit en C. La synthèse d'AUGH crée 1 multiplieur, 5 additionneurs/soustracteurs/comparateurs et une ROM (pour h).

Le cas de la mémoire des échantillons est un cas spécifique pour permettre de décaler les échantillons. Il s'agit d'une boucle `for`. Sans indications, AUGH crée une RAM. En conséquence, le décalage nécessite un état par échantillon à décaler. Deux transformations manuelles résolvent ce problème : le déroulement de la boucle (pour rendre les adresses de sélection statiques), et l'ajout de ports directs de lecture et écriture (pour permettre des lectures et écritures concurrentes). Le résultat est un registre à décalage classique.

CatapultC La version de C utilisé par CatapultC diffère de AUGH seulement dans la déclaration des types physiques et des opérations de lecture et écriture sur les files d'entrée et de sortie. La synthèse de CatapultC génère 2 multiplieurs, 9 additionneurs/soustracteurs/comparateurs et une ROM. L'opération de décalage est automatiquement déroulée.

Les opérateurs supplémentaires comparés à AUGH proviennent de l'ordonnancement. CatapultC choisit de ne passer qu'un cycle d'horloge par itération de l'algorithme de la FIGURE 2.4b.

3.4.2.3 Performances : surface, temps de calcul et énergie

Dans cette section, nous comparons en termes de surface et de consommation en énergie les circuits conçus manuellement, avec ceux générés par les outils de HLS synchrones et leurs versions désynchronisées avec ALPS. Il y a donc 6 filtres :

1. « Uniforme » : Le filtre RIF uniforme synchrone (avec désactivation du signal d'horloge) conçu en RTL.
2. « AUGH » : Le filtre RIFI non uniforme synchrone (avec désactivation du signal d'horloge) conçu en C et traité par AUGH.
3. « AUGH+ALPS » : Le filtre RIFI non uniforme asynchrone obtenu par désynchronisation dans ALPS du circuit généré par AUGH.
4. « Catapult » : Le filtre RIFI non uniforme synchrone conçu en C et traité par CatapultC. Nous ne présentons pas le circuit avec désactivation de l'horloge car la longueur du chemin critique était multipliée par 3 et la surface augmentée de +95% à cause des délais insérés pour respecter les contraintes de timing.
5. « Catapult+ALPS » : Le filtre RIFI non uniforme asynchrone obtenu par désynchronisation dans ALPS du circuit généré par CatapultC.
6. « Manuel » : Le filtre RIFI non uniforme asynchrone conçu en RTL pour la partie calcul et par primitives pour la partie contrôle.

Les outils de HLS AUGH et CatapultC sont configurés pour minimiser la surface du circuit avec une contrainte de fréquence d'horloge de 50 MHz. Nous synthétisons les

circuits sur une technologie TSMC 40 nm CMOS avec une bibliothèque de cellules de Muller fournies par Dolphin Integration.

Surface des cellules

La FIGURE 3.14 montre les surfaces des circuits précédemment décrits. Le changement de paradigme de l'échantillonnage uniforme à non uniforme cause une augmentation de matériel de 35% à 54%. Cela est dû à l'augmentation de la complexité de l'algorithme.

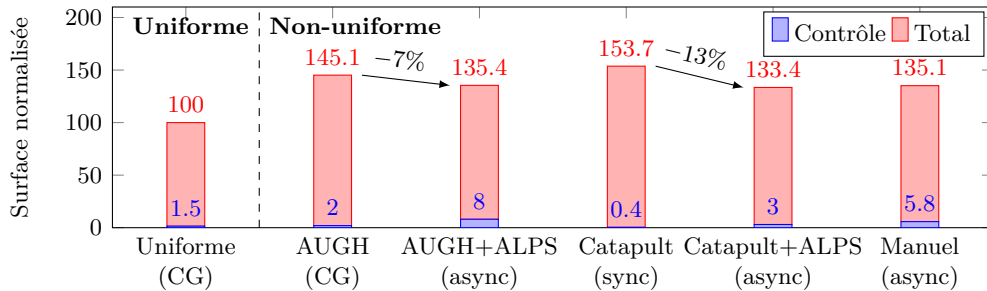


FIGURE 3.14 – Surfaces des filtres. Le filtre « Uniforme » sert de référence. La partie foncée correspond à la surface de la partie de contrôle.

Parmi les filtres RIFI non uniformes, les circuits asynchrones sont entre 7% et 13% plus petits que leurs équivalents synchrones grâce à l'intégration de l'activation parcimonieuse d'horloges locales dans le contrôle asynchrone. En effet, dans le circuit synchrone généré par CatapultC (sans désactivation de l'horloge (*Clock Gating*) (CG)), chaque bascule du registre à décalage doit être précédée d'un multiplexeur pour conserver sa valeur en l'absence de décalage. Avec la CG, les données ne sont décalées seulement lorsque l'horloge locale est activée. La logique de CG peut-être partagée entre les registres.

Temps de calcul

La FIGURE 3.15 montre le temps de calcul pour chaque filtre estimé en simulant le traitement de l'ECG par le circuit après synthèse. La même marge a été appliquée aux chemins critiques synchrones et asynchrones.

Entre les filtres synchrones, la plus grande complexité de l'algorithme de traitement dans le cas synchrone entraîne un temps de calcul par échantillon des filtres RIFIs est entre 3 fois (CatapultC) et 11 fois (AUGH) plus long que pour le RIF. Le temps de calcul pour AUGH est plus long car pour économiser du matériel, AUGH utilise plusieurs cycles là où CatapultC en utilise un seul.

La méthode de désynchronisation d'ALPS permet à chaque état de l'AFSM de fonctionner à la vitesse maximum permise par le chemin critique de l'étage. En comparaison, la cadence des états d'une FSM synchrone est déterminée par le pire chemin critique de tout le circuit. Le gain est particulièrement important (46%) dans le cas du circuit

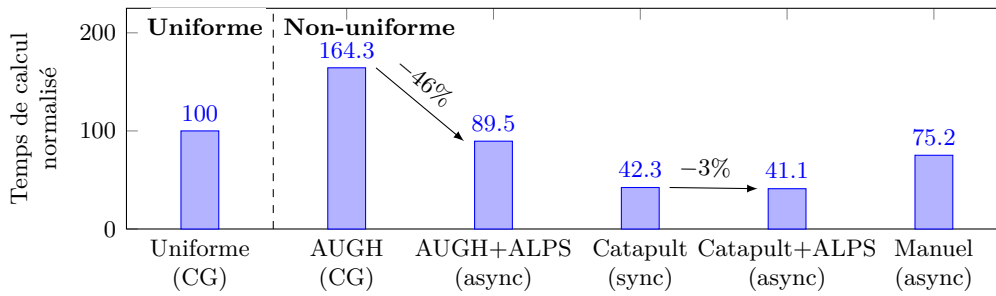


FIGURE 3.15 – Temps de calcul des filtres. Le filtre « Uniforme » sert de référence.

« AUGH+ALPS » car les états de branchement peuvent être exécutés rapidement. Dans le cas de « Catapult+ALPS », le gain est limité car la FSM peut boucler sur le même état alors que dans l'AFSM doit avoir au moins deux états par boucle.

Énergie consommée

Pour estimer l'énergie consommée, nous simulons le traitement de l'ECG au niveau portes logiques et enregistrons l'activité des fils. L'énergie totale consommée est alors la somme des coûts estimés des basculements de chaque fil pondérée par l'activité mesurée.

Les énergies consommées sont données sur la FIGURE 3.16. Dans le cas de l'ECG, l'échantillonnage non uniforme permet de réduire le nombre d'échantillons par 7. Malgré le surplus d'opérations requis pour traiter chaque échantillon, les filtres RIFI non uniformes consomment entre 35% et 75% moins d'énergie que le filtre RIF uniforme pour terminer le traitement.

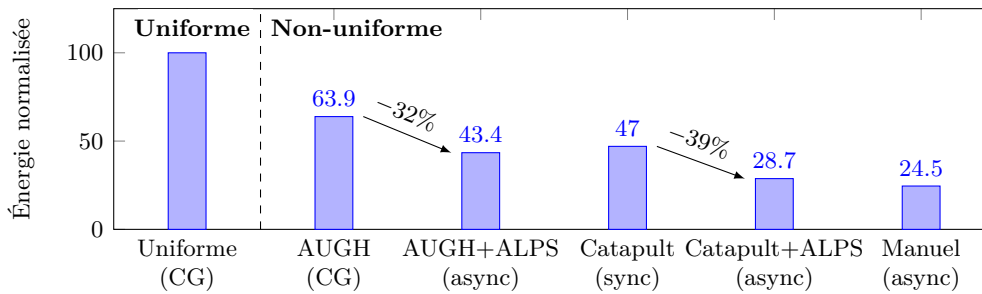


FIGURE 3.16 – Consommation en énergie des filtres. Le filtre « Uniforme » sert de référence.

Le processus de désynchronisation d'ALPS permet de réduire de 32% la consommation d'énergie du filtre généré par AUGH seul. Pour l'outil industriel de HLS CatapultC, le circuit désynchronisé par ALPS (« Catapult+ALPS ») présente une consommation en énergie réduite de 38% par rapport au circuit synchrone original. Le filtre RIFI asynchrone conçu manuellement consomme 15% moins d'énergie que le meilleur des circuits

générés (« Catapult+ALPS »).

3.4.2.4 Conclusion de l'étude de cas

L'échantillonnage non uniforme permet une réduction significative de la consommation en énergie. Le facteur de réduction atteint 4 dans le cas de l'ECG. Comme l'échantillonnage et la performance dépendent grandement du signal d'entrée, il devient important d'optimiser l'échantillonnage pour l'application visée. Ainsi, la généralité est réduite au profit de l'efficacité.

ALPS permet de générer automatiquement des circuits exploitant l'échantillonnage non uniforme. Certes les circuits conçus manuellement à plus bas niveau permettent de meilleurs résultats, mais ils nécessitent un temps de développement plus long et un degré de familiarité plus important avec les méthodes de conception asynchrones. Ainsi, ALPS aide à fournir rapidement un circuit basse consommation et à évaluer la performance des algorithmes adaptés à l'échantillonnage non uniforme. Selon le cas, le circuit généré par AUGH pourra motiver une amélioration de l'algorithme, être utilisé en l'état ou même servir de base pour la conception d'un circuit optimisé en RTL.

3.5 Conclusion

Dans ce chapitre, nous avons présenté le flot ALPS. La conception de l'algorithme de traitement et le choix des paramètres d'échantillonnage sont faits itérativement au niveau algorithmique en s'appuyant sur une connaissance *a priori* de l'application et des signaux. La description matérielle du CAN à traversée de niveau correspondant est générée à partir des paramètres d'échantillonnage. D'autre part, la description matérielle asynchrone de l'unité de traitement est produite automatiquement à partir l'algorithme traduit en C par un outil de HLS synchrone et une désynchronisation de la FSM. Pour remplacer la FSM générée par la HLS, nous avons introduit une nouvelle architecture d'AFSM basée sur des contrôleurs de pipelines asynchrones.

Dans le cas du filtrage d'un ECG, le filtrage non uniforme a permis de diviser le nombre d'échantillons par 7 et de réduire la consommation en énergie de la partie traitement de 36% à 75%. Les circuits asynchrones obtenus en utilisant la méthode de désynchronisation d'ALPS consomment de 32% à 39% moins d'énergie que les circuits obtenus avec les seuls outils de HLS.

Optimisation de la consommation des micropipelines

Sommaire

4.1	Évaluation des protocoles tardifs et nouveaux protocoles . . .	61
4.1.1	Caractéristiques temporelles des pipelines à données groupées .	61
4.1.1.1	Protocoles précoces	62
4.1.1.2	Le protocole 2 phases	63
4.1.1.3	Protocoles tardifs existants	63
4.1.2	Nouveaux protocoles tardifs	65
4.1.2.1	Late-capture	65
4.1.2.2	Maximus	66
4.1.3	Comparaison des protocoles tardifs et précoces	66
4.1.3.1	Résumé de l'analyse formelle	66
4.1.3.2	Banc de test	67
4.1.3.3	Analyse des simulations	69
4.1.4	Perspective : pipelines multi-protocoles	72
4.2	Modèle comportemental à base de réseaux de Petri	73
4.2.1	Modèle « canal équivalent » : fragments élémentaires des primitives	73
4.2.1.1	Fragments associés aux primitives non conditionnelles	74
4.2.1.2	Fragments associés aux primitives de sélection	75
4.2.2	Méthodologie de génération par simplification d'un STG de référence	76
4.2.2.1	Définitions formelles sur les réseaux de Petri (PNs) .	76
4.2.2.2	Stratégie de construction par simplifications itérées .	77
4.2.2.3	Modèles résultant des simplifications	81
4.2.3	Exemple : Conception et vérification d'un AES	81
4.3	Conclusion	83

Dans le chapitre précédent, la recherche d'un protocole adapté à une AFSM (SECTION 3.2.2.1) aboutit au choix d'un protocole tardif. Alors que les 136 protocoles précoces ont été dénombrés (MCGEE et NOWICK 2005; BIRTWISTLE et STEVENS 2008), les protocoles tardifs restent largement inexplorés avec seulement trois protocoles présents dans la littérature (DAVID 1977; YUN, BEEREL et ARCEO 1996; MANNAKKARA et YONEDA 2010).

Dans ce chapitre, nous étudions dans une première section les protocoles tardifs et montrons notamment qu'ils améliorent la surface, le débit et l'énergie consommée des pipelines avec de longs chemins de données. Dans une seconde section, nous présentons un modèle à base de réseaux de Petri pour la conception, l'analyse et la vérification des circuits à données groupées.

4.1 Évaluation des protocoles tardifs et nouveaux protocoles

Les protocoles tardifs utilisent le front descendant du fil de requête pour signaler l'arrivée d'une donnée valide. Parmi les quatre conventions de validité possibles, la convention tardive est la seule propre aux circuits à données groupées et la dernière qui ne soit pas exhaustivement connue.

Dans cette section nous analysons formellement les propriétés temporelles des protocoles précoces et tardifs existants, puis celles deux nouveaux protocoles tardifs. Des simulations au niveau portes logiques servent à valider ces propriétés.

4.1.1 Caractéristiques temporelles des pipelines à données groupées

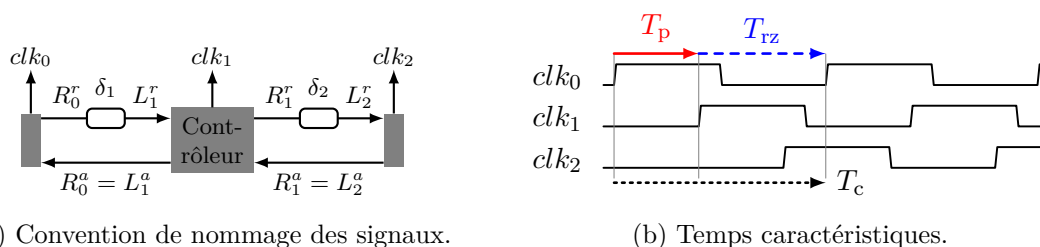


FIGURE 4.1 – Structure et temps caractéristiques d'un pipeline asynchrone.

La FIGURE 4.1a rappelle la structure d'un pipeline à données groupées. Pour chaque étage $k \in \{0, 1, 2\}$, l'horloge locale clk_k contrôle les registres de l'étage dans les chemins de données. Le retard δ_k doit être ajusté en fonction du chemin de données critique pour cet étage. Cette section étudie l'effet de ce retard sur le comportement du pipeline.

Pour chaque étage k , nous définissons 4 temps caractéristiques :

T_{cp} Le *délai critique* est la longueur du chemin critique de la logique combinatoire associée à l'étage. Comme pour les circuits synchrones, l'estimation de T_{cp} doit

prendre en compte les possibles variations environnementales.

T_p Le *temps de propagation* est le temps minimal entre le front d'horloge de l'étage précédent ($clk_{k-1}+$) et celui de l'étage courant (clk_k+). La contrainte de données groupées impose $T_p \geq T_{cp}$.

T_{rz} Une fois que les données sont envoyées vers l'étage suivant, le canal se réinitialise pour se préparer au prochain transfert. T_{rz} est le *temps de retour à zéro (RZ)*.

T_c Après le RZ, l'étage est capable de capturer de nouvelles données. Le temps minimal entre deux fronts successifs d'horloge d'un même étage est le *temps de cycle*. T_c détermine le débit maximal de l'étage.

T_{cp} est une constante pour un chemin de données fixé. T_p , T_{rz} et T_c sont des fonctions du retard sur la requête de l'étage δ_k et du protocole. Pour des raisons de lisibilité, nous faisons les trois suppositions suivantes :

1. Tous les retards δ_k sont égaux au même délai δ . Ainsi, tous les étages sont équivalents et ont le même T_p et T_c .
2. Les délais de propagation dans les portes et les fils sont négligeables par rapport à δ . Cela simplifie les expressions formelles (les égalités sont donc vraies asymptotiquement pour $\delta \rightarrow +\infty$) mais cache l'effet de la complexité du contrôleur qui prédomine lorsque $\delta \rightarrow 0$. Notons que les simulations de la SECTION 4.1.3.3 prennent en compte tous les délais.
3. Il est possible de saturer la contrainte de données groupées ($T_p \geq T_{cp}$), qui devient donc (4.1). En pratique, les variations environnementales obligent à ajouter une marge pour assurer que la contrainte soit vérifiée pour toutes les conditions de fonctionnement (*corners*).

$$T_{cp} = T_p \tag{4.1}$$

4.1.1.1 Protocoles précoces

La FIGURE 4.2 montre les STGs de trois protocoles de FURBER et DAY (1996) : les protocoles Simple, Semi-découplé et Entièrement découplé. Les protocoles découplés introduisent plus de concurrence entre les canaux gauche (L) et droit (R) mais nécessitent des contrôleurs plus complexes. Les trois protocoles vérifient (4.2).

$$T_p = \delta \quad \text{et} \quad T_{rz} = \delta \tag{4.2}$$

T_p est dû à la transmission de la donnée ayant lieu juste après la première transition (L^r+). Dans les protocoles Simple et Semi-découplé, T_{rz} est dû à la donnée devant être propagé à l'étage 2 avant de pouvoir accepter une nouvelle donnée de l'étage 0. Ainsi, au plus un étage sur deux peut contenir des données et ces deux protocoles sont donc appelés des protocoles *half-buffer*. Inversement, le protocole Entièrement découplé est un protocole *full-buffer*. T_{rz} n'est plus limité par la propagation dans le canal suivant mais seulement par le temps RZ du canal 1.

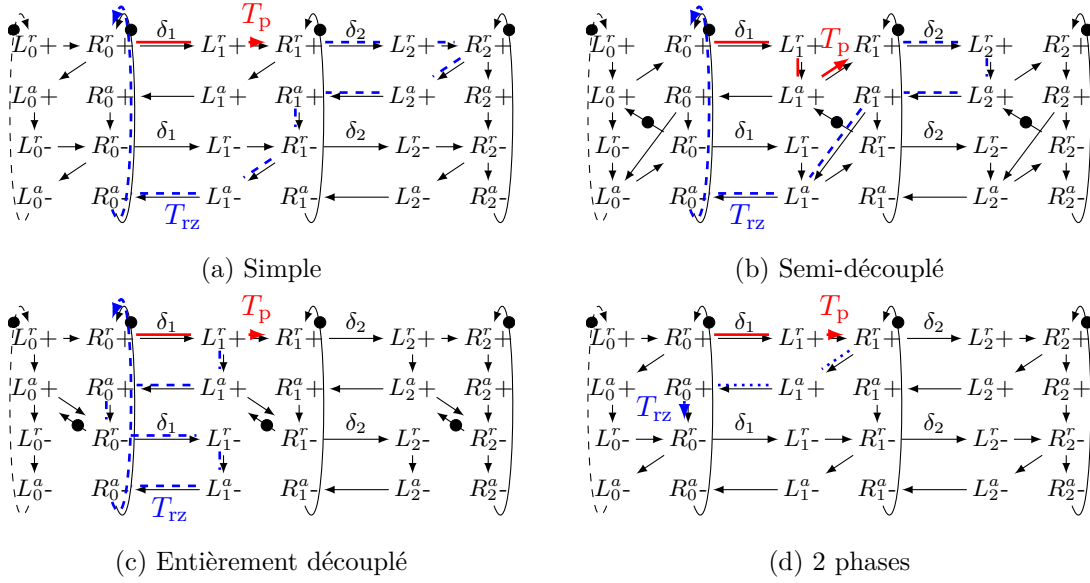


FIGURE 4.2 – STGs de protocoles précoces et 2 phases.

Étant donnée la contrainte de données groupées (4.1), la taille du retard est directement proportionnelle à la longueur du chemin critique associé à l'étage. De plus, le temps de cycle est au moins deux fois plus long que le chemin critique (4.3).

$$\delta = T_{cp} \quad \text{et} \quad T_c = 2T_{cp} \quad (4.3)$$

Les même arguments et propriétés sont valables pour tous les protocoles précoces.

4.1.1.2 Le protocole 2 phases

Le protocole 2 phases (FIGURE 4.2d) a la même spécification que le protocole simple (FIGURE 4.2a). Le contrôleur est donc identique. En revanche, dans le cas 2 phases, les deux fronts $R+$ et $R-$ capturent des données. Ainsi, le chemin de RZ va de R_1^+ à R_0^- et donc $T_p = \delta$ et $T_{rz} \approx 0$ (plus précisément négligeable devant δ ou $\mathcal{O}(1)$).

4.1.1.3 Protocoles tardifs existants

Les protocoles tardifs utilisent le front descendant du fil de requête d'entrée (L^-) pour signaler la validité des données, au lieu de L^+ pour les protocoles précoces. À notre connaissance, seuls trois protocoles tardifs ont été présentés dans la littérature : le protocole CUSA de DAVID (1977), le protocole Burst-mode de YUN, BEEREL et ARCEO (1996) et le protocole Early-ack de MANNAKKARA et YONEDA (2010). Les trois protocoles appliquent la même réduction de concurrence sur le canal L : le plus court chemin de R_0^+ à R_1^+ est $R_0^+ \xrightarrow{\delta_1} L_1^+ \rightarrow L_1^- \rightarrow R_0^+ \rightarrow R_0^- \xrightarrow{\delta_1} L_1^- \rightarrow R_1^+$. Cette

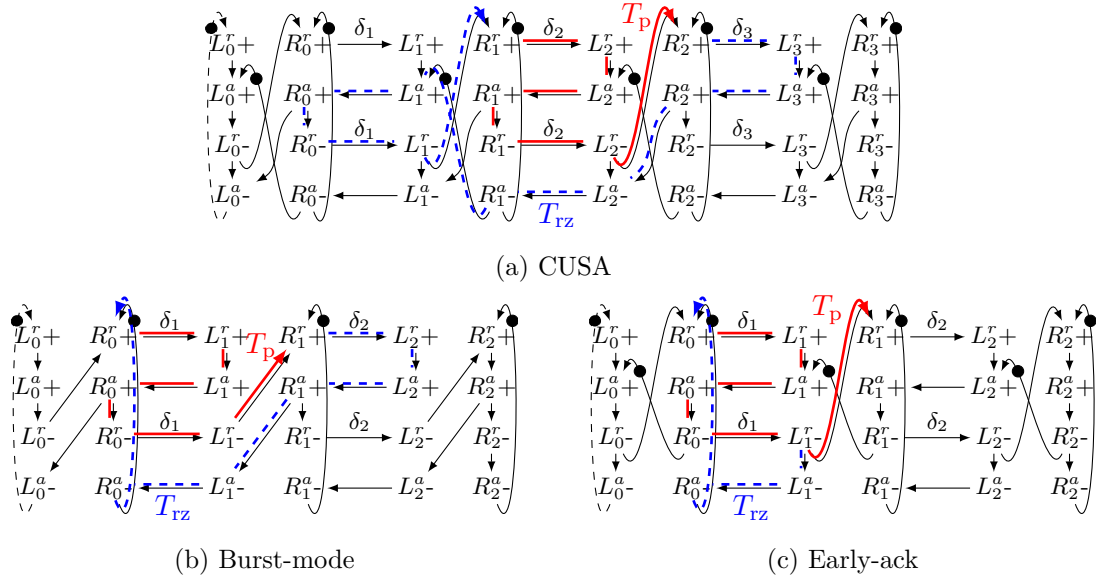


FIGURE 4.3 – Chemins critiques des protocoles tardifs existants.

réduction correspond à la coupe $L2222$ dans l'infrastructure de BIRTWISTLE et STEVENS (2008). Mais les protocoles ont des découplages différents.

CUSA Comme le montre la FIGURE 4.3a, le protocole CUSA est le moins découplé des trois. La requête passe deux fois par le retard δ_2 avant de signaler la validité d'une nouvelle donnée. Le protocole vérifie donc $T_p = 2\delta_2 = 2\delta$.

À cause du faible découplage, le chemin RZ passe par les retards δ_3 et δ_1 , donc $T_{rz} = 2\delta$ pour le protocole CUSA.

Burst-mode Le protocole Burst-mode (FIGURE 4.3b) découple la transition L^a+ en retirant la dépendance $R^a- \rightarrow L^a+$ qui existe dans le protocole CUSA. Comme le STG le montre, $T_p = 2\delta$ et $T_{rz} = \delta$ pour le protocole Burst-mode.

Early-ack Le protocole Early-ack (FIGURE 4.3c) découple la transition L^a- en retirant la dépendance $R^a+ \rightarrow L^a-$ et augmente la concurrence en remplaçant $R^a- \rightarrow L^a+$ par $R^r- \rightarrow L^a+$. Cette spécification nécessite néanmoins une propagation rapide du front descendant de la requête ($R_0^- \xrightarrow{\delta_1} L_1^- \rightarrow R_1^+$) sans quoi une nouvelle donnée pourrait être acceptée ($R_0^- \rightarrow L_0^+ \rightarrow L_0^- \rightarrow R_0^+$) avant que l'étage suivant n'ait lu la donnée courante. Comme ce dernier chemin ne passe par aucun retard, le retard δ_1 sur le premier chemin doit propager rapidement les fronts descendants. On parle alors de retard asymétrique. Ainsi, $T_p = \delta_1$ (délai montant uniquement) et T_{rz} est négligeable.

4.1.2 Nouveaux protocoles tardifs

Comme l'indique la FIGURE 4.5, en terme de découplage, le protocole Late-capture est le symétrique du protocole Burst-mode par rapport au protocole CUSA et le protocole Maximus applique les deux découplages des protocoles Burst-mode et Late-capture.

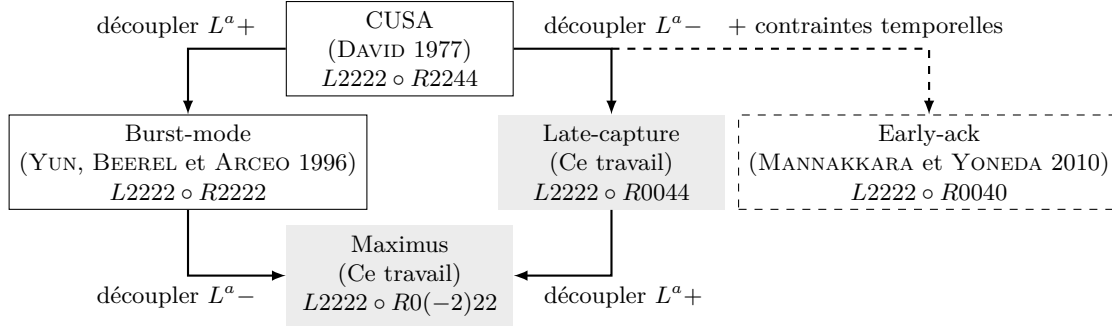


FIGURE 4.4 – Relations entre les protocoles tardifs. Les flèches indiquent une augmentation du découplage.

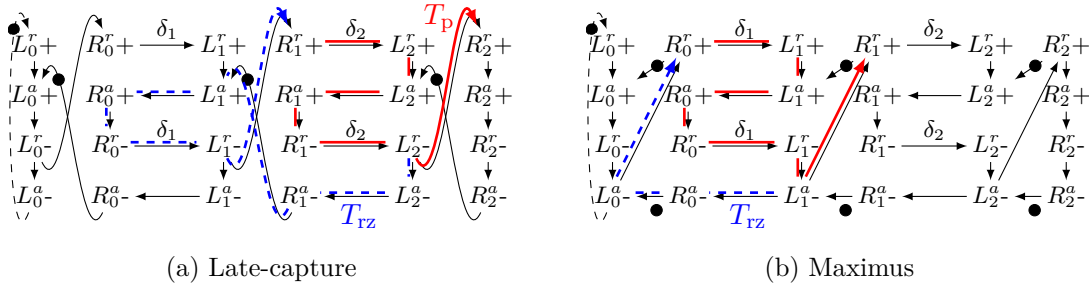


FIGURE 4.5 – Chemins critiques des nouveaux protocoles tardifs.

4.1.2.1 Late-capture

Par rapport au protocole CUSA, le protocole tardif Late-capture découple la transition L^a- en retirant la dépendance $R^a+ \rightarrow L^a-$. Il suit donc le même schéma de découplage que Early-ack mais ne réduit pas la concurrence en conservant la dépendance $R^a- \rightarrow L^a+$. Le canal d'entrée doit donc attendre R^a- au lieu la transition précédente dans le canal de sortie R^r- . Cette réduction de concurrence permet l'utilisation de retards symétriques. Notons que le protocole Late-capture est le symétrique du protocole Burst-mode par rapport au protocole CUSA.

Comme le montre le STG du protocole Late-capture (FIGURE 4.5a), $T_p = 2\delta$ et $T_{rz} = \delta$.

4.1.2.2 Maximus

Nous avons trouvé le protocole Maximus¹ en cherchant à minimiser T_{rz} tout en conservant la possibilité d'utiliser des retards symétriques. Comme le montre la FIGURE 4.4, le protocole Maximus est le plus découplé des protocoles tardifs connus actuellement : les canaux d'entrée et de sortie ne se synchronisent quasiment qu'à la fin de leurs poignées de main respectives ($R^a- \rightarrow L^a-$).

En plus de la synchronisation à la fin des poignées de main, le protocole comprend la dépendance $R^r+ \rightarrow L^a+$. Cette dépendance supplémentaire réduit le nombre de variables internes d'états nécessaires pour encoder correctement tous les états : une seule variable interne suffit. Cela limite donc la complexité du contrôleur. De plus, cette réduction formelle de la concurrence ne dégrade pas la performance. En effet, le chemin $L^a- \rightarrow R^r+ \rightarrow L^a+$ est constitué seulement de transitions internes, et donc prendra moins de temps que le chemin $L^a- \rightarrow \dots \xrightarrow{\delta} L^r+ \rightarrow L^a+$ qui passe par un retard.

Comme le montre la FIGURE 4.5b, $T_p = 2\delta$ et $T_{rz} \approx 0$ pour le protocole Maximus. Les protocoles avec davantage de concurrence auront probablement des contrôleurs plus complexes (plus de variables d'états).

De tous les protocoles tardifs présentés, le protocole Maximus est le premier qui n'a pas d'équivalent précoce. En effet les autres protocoles restent valides si l'on change la convention de validité en conservant les contraintes d'ordonnement des phases. Ainsi, nous prouvons que les environnements existants (MCGEE et NOWICK 2005 ; BIRTWISTLE et STEVENS 2008) basés sur un protocole précoce de découplage maximal ne sont pas suffisants pour énumérer tous les protocoles asynchrones. De fait, la réduction de concurrence sur le canal L des protocoles tardifs permet d'acroître celle sur le canal R .

On peut néanmoins étendre légèrement l'environnement de BIRTWISTLE et STEVENS (2008) pour caractériser le protocole Maximus. Si on autorise de rajouter des état au graphe de départ (indices de coupes négatifs), alors le protocole Maximus est défini par la coupe $L2222 \circ R0(-2)22$. Cette extension permet de caractériser chaque nouveau protocole mais ne suffit pas pour énumérer tous les protocoles. Il faudrait soit trouver le protocole tardif de découplage maximal et s'en servir comme souche, soit formaliser des règles d'ajout d'états et borner par ligne le nombre d'états que l'on peut rajouter.

4.1.3 Comparaison des protocoles tardifs et précoces

4.1.3.1 Résumé de l'analyse formelle

La TABLE 4.1 résume les propriétés des protocoles précoces et tardifs. Les protocoles tardifs ont un temps de propagation T_p égal à 2δ excepté pour le protocole Early-ack pour qui $T_p = \delta$ comme le 2 phases et les protocoles précoces. Le temps de RZ, T_{rz} , varie de 2δ pour le protocole le moins découplé (CUSA) à 0 pour les plus découplés (protocoles Early-ack, Maximus et 2 phases), en passant par δ (protocoles précoces, Burst-mode et

1. D'où son nom en référence à son découplage. « Maximus » signifie « très grand » ou « le plus grand » en latin.

TABLE 4.1 – Propriétés des protocoles tardifs comparés aux protocoles précoces. α est le coefficient de proportionnalité de la surface (en $\text{m}^2 \text{s}^{-1}$).

Protocole	T_p	T_{rz}	Surface (δ)	Période min. T_c ($T_p + T_{rz}$)	Util. de cycle max. (T_{cp}/T_c)
2 phases	δ	0	αT_{cp}	T_{cp}	1
Protocoles précoces	δ	δ	αT_{cp}	$2T_{cp}$	$1/2$
Early-ack	δ	0	αT_{cp}	T_{cp}	1
CUSA	2δ	2δ	$\frac{\alpha}{2} T_{cp}$	$2T_{cp}$	$1/2$
Burst-mode	2δ	δ	$\frac{\alpha}{2} T_{cp}$	$\frac{3}{2} T_{cp}$	$2/3$
Late-capture	2δ	δ	$\frac{\alpha}{2} T_{cp}$	$\frac{3}{2} T_{cp}$	$2/3$
Maximus	2δ	0	$\frac{\alpha}{2} T_{cp}$	T_{cp}	1

Late-capture).

La surface d'un étage du pipeline est la surface du contrôleur plus celle du retard. Cette dernière est proportionnelle à δ . En utilisant l'équation (4.1), il vient $\delta = T_{cp}/2$ pour les protocoles CUSA, Burst-mode, Late-capture and Maximus. Comparés aux protocoles 2 phases et précoces, pour une même longueur de chemin critique T_{cp} , la taille du retard d'un contrôleur tardif est divisée de moitié. Cela réduit la surface totale de la partie contrôle. Pour le protocole Early-ack, les retards asymétriques empêchent de bénéficier de cette réduction de surface ($\delta = T_{cp}$).

Le temps de cycle $T_c = T_p + T_{rz}$ d'un étage de pipeline est l'inverse du débit maximum de l'étage. En conséquence, pour une longueur de chemin critique T_{cp} , maximiser le débit du pipeline revient à faire tendre le rapport d'utilisation du cycle T_{cp}/T_c vers 1. Le protocole CUSA a une utilisation du cycle similaire aux protocoles précoces ($1/2$). Les protocoles Burst-mode et Late-capture améliorent cette utilisation à $2/3$. Enfin, les protocoles 2 phases, Early-ack et Maximus tendent vers l'utilisation de cycle idéale égale à 1 lorsque δ est grand comparé aux temps de traversée des portes des contrôleurs.

4.1.3.2 Banc de test

Nous comparons les protocoles proposés Late-capture et Maximus aux protocoles existants Simple et Burst-mode. Les contrôleurs sont synthétisés sur une technologie TSMC 40 nm CMOS avec une bibliothèque de portes de Muller fournie par Dolphin Integration. Les simulations au niveau portes logiques avec des délais rétro-annotés visent à raffiner les propriétés résumées dans la section précédente.

Logique des contrôleurs La réalisation physique des contrôleurs utilisent des portes de Muller symétriques et asymétriques. La FIGURE 4.6 montre le symbole et la structure

des portes aux niveaux transistors. Comme la bibliothèque de cellules à notre disposition ne contenait que des portes de Muller symétriques, la FIGURE 4.6d montre la réalisation au niveau portes des portes de Muller asymétriques.

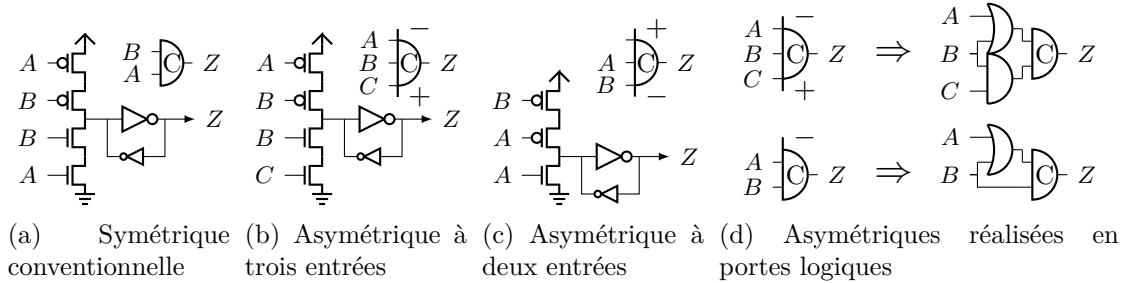


FIGURE 4.6 – Réalisations physique et logique des portes de Muller.

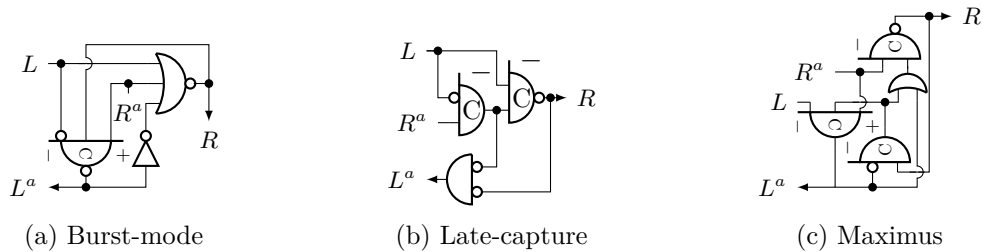


FIGURE 4.7 – Réalisations logique des contrôleurs tardifs.

La FIGURE 4.7 montre les contrôleurs de pipelines synthétisés au niveau portes logiques par l'outil *Petrify*. Il s'agit de circuits indépendants de la vitesse.

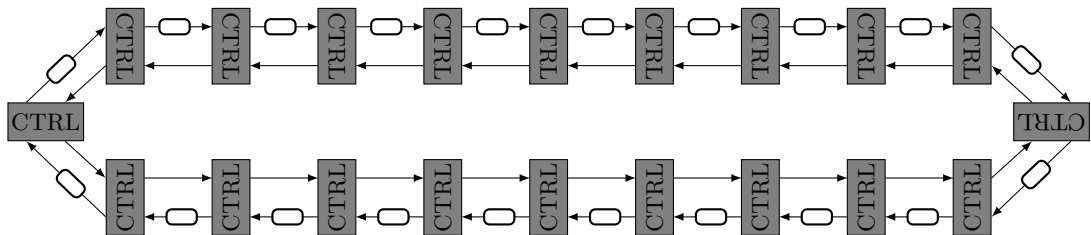


FIGURE 4.8 – Pipeline en anneau pour le banc de test. Le pipeline est formé de 20 étages et peut contenir jusqu'à 19 données se propageant en sens horaire.

Configuration du pipeline en anneau Pour chaque contrôleur, nous simulons un anneau de 20 étages identiques (FIGURE 4.8). La configuration circulaire permet de conserver un nombre constant de données dans le pipeline. Au plus $20 - 1 = 19$ données peuvent circuler dans le pipeline. Ce cas correspond à un rapport du nombre de données sur le nombre d'étage de 95%. Ce rapport est appelé taux d'occupation. Pour pouvoir

contenir 19 données, nous devons mettre deux contrôleurs Simple par étage du pipeline. En effet, les contrôleurs Simple sont des *half-buffers* alors que les autres contrôleurs sont des *full-buffers*.

Nous simulons les pipelines pour différentes longueurs des éléments de délais δ . Tous les étages ont des retards de même longueur. Les retards sont des chaînes d'inverseurs de taille minimale, ce qui donne des retards symétriques.

4.1.3.3 Analyse des simulations

Afin de faire abstraction de la technologie utilisée et de la largeur des données circulant dans le pipeline, l'unité de délai est le délai de porte (*gate delay*) (GD). Il correspond au temps de traversée d'un inverseur de taille minimale connecté à trois inverseurs identiques (*fan-out* 3). De plus, une donnée unitaire (*data item*) (DI) contient un nombre de bits correspondant à l'étage de sortie du pipeline.

Dans la technologie utilisée, $1 \text{ GD} = 420 \text{ ps}$. Ainsi, le débit du pipeline est exprimé en « milli DI par GD » (mDI/GD). Si la sortie de pipeline est de 32 bits, alors, $1 \text{ mDI/GD} = 76 \text{ Mbit/s}$. Le pipeline avec les contrôleurs Simple et aucun retard sert de référence pour les normalisations.

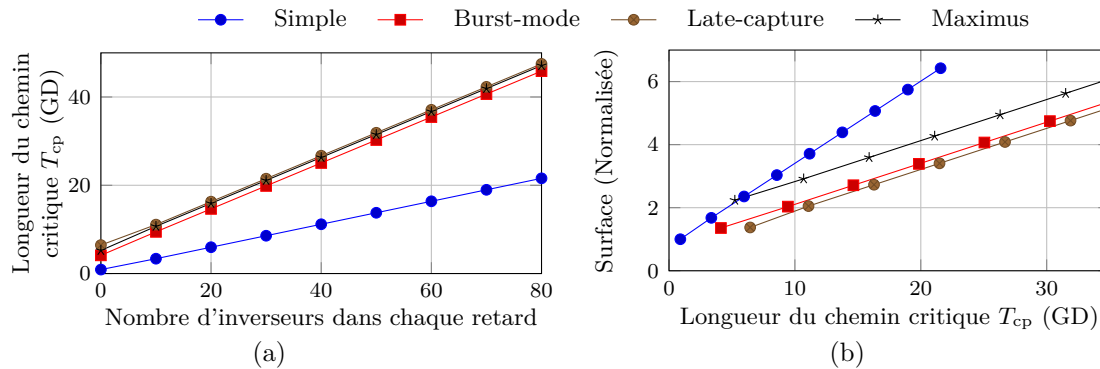


FIGURE 4.9 – Temps de propagation et surface.

Temps de propagation et surface Comme attendu d'après la TABLE 4.1, la FIGURE 4.9a montre que le temps de propagation des protocoles tardifs est deux fois plus long que celui du protocole Simple pour la même longueur de retard δ .

En conséquence, étant donné une longueur de chemin critique T_{cp} , les protocoles tardifs nécessitent d'insérer moins d'inverseurs dans les retards pour respecter la contrainte de données groupées. Néanmoins, comme le montre la FIGURE 4.9b, le contrôleur Simple (même doublé) est plus petit que les autres : les surfaces supplémentaires des contrôleurs Late-capture, Burst-mode et Maximus sont respectivement aussi importantes que 6, 6 et 18 inverseurs.

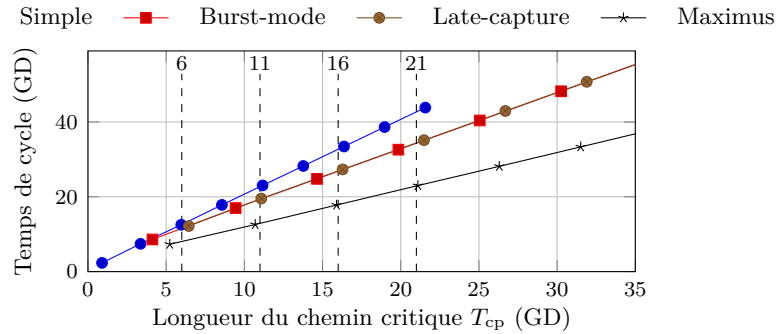


FIGURE 4.10 – Temps cycle minimum.

Temps de cycle La FIGURE 4.10 montre le temps de cycle en fonction du protocole et du temps de propagation. En accord avec la TABLE 4.1, les pentes des droites sont de 2 pour le protocole Simple, $3/2$ pour les protocoles Late-capture et Burst-mode et 1 pour le protocole Maximus. À cause de la complexité supplémentaire des protocoles tardifs, seul le contrôleur Simple permet des temps de propagation et de cycle inférieurs à 4 GD.

Débit en fonction du taux d'occupation des pipelines En pratique, le temps de cycle est plus grand que T_c car le contrôleur attend soit des requêtes (en cas de faible occupation) ou des acquittements (en cas de forte occupation).

Pour une comparaison plus juste, nous choisissons pour chaque protocole une longueur de chaînes de délais telle que le temps de propagation est identique pour tous les protocoles. Les graphes de la FIGURE 4.11 comparent les débits maximums permis par les protocoles pour des temps de propagation de 6, 11, 16 et 21 GD.

Dans la partie gauche des graphes (faible occupation), tous les protocoles ont des débits équivalents. En effet le temps de cycle est limité par la propagation, déterminée par T_p , des quelques données dans le pipeline. Dans la partie droite (forte occupation), le temps de cycle est limité par la phase de RZ, c'est-à-dire la vitesse de propagation de l'information de relâchement des données. Pour le protocole Maximus, cette vitesse est indépendante de la longueur des retards², ce qui mène à de hauts débits, même avec une forte occupation et des retards longs.

Énergie consommée Pour chaque transfert d'une donnée, tous les fils d'un étage basculent deux fois. Ainsi, l'énergie nécessaire peut être décomposée en deux parties : la consommation du contrôleur, fixée lorsque les courants de fuite sont négligeables, et la consommation du retard, proportionnelle à sa longueur.

Comme le montre la FIGURE 4.12a, à cause de leur plus grande complexité, les

2. L'acquittement fait un tour de l'anneau en 24 GD. Cela explique que le débit de Maximus pour une occupation de 95 % est constamment égale à 42 mDI/GD.

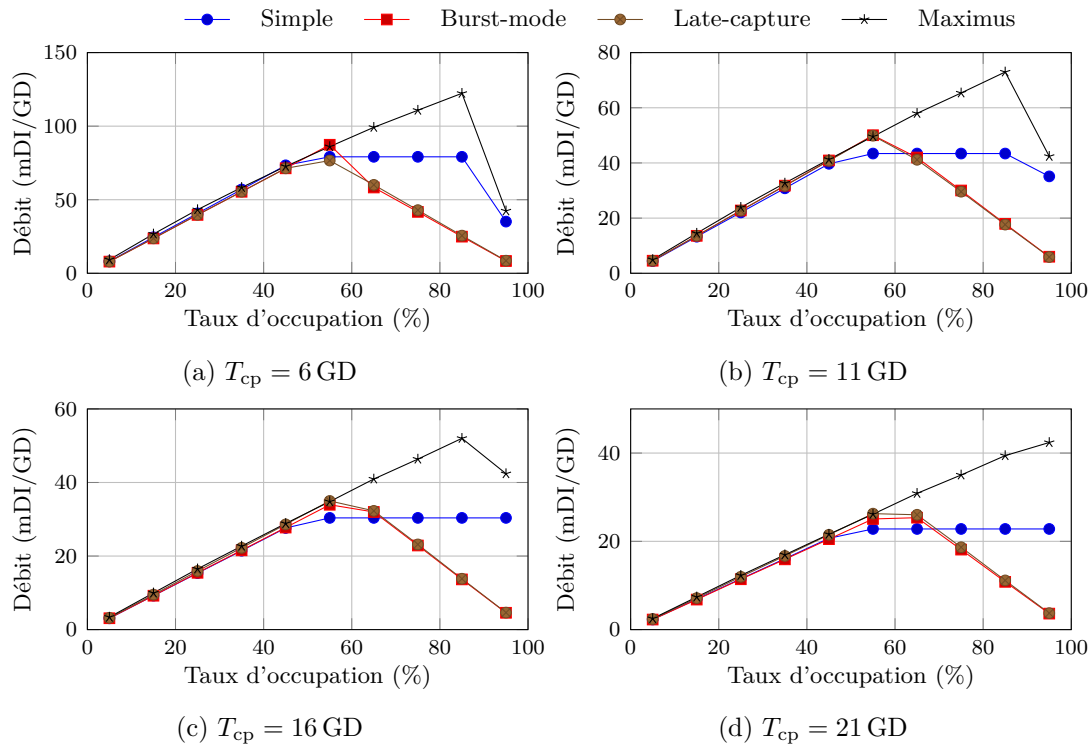


FIGURE 4.11 – Débit en fonction de la taille du chemin critique et du taux d'occupation.

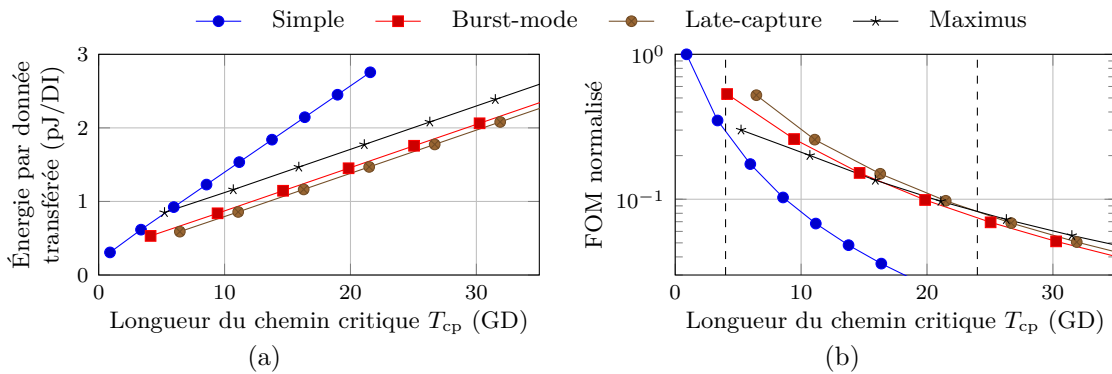


FIGURE 4.12 – Énergie consommée et score de mérite (*Figure Of Merit*) (FOM).

contrôleurs Burst-mode et Late-capture (resp. Maximus) consomment 2 fois (resp. 3 fois) plus d'énergie que le contrôleur Simple. Néanmoins, l'utilisation efficace des retards atténue l'effet de cette consommation supplémentaire pour des chemins critiques longs. Par exemple, pour un chemin critique long de 14 portes, la consommation énergétique des étages utilisant les protocoles Late-capture et Maximus sont respectivement réduites de 45% et 30% comparés au protocole simple.

Score de mérite Afin de choisir entre les protocoles, nous définissons le score de mérite (*Figure Of Merit*) (FOM) suivant comme une fonction du temps de cycle minimum (T_c), du temps de propagation minimum (T_p), de la surface des étages (A) et de l'énergie par transfert (E) :

$$\text{FOM} = \frac{T_p}{T_c A E} \quad (4.4)$$

La FIGURE 4.12b trace le FOM normalisé pour chaque contrôleur en fonction de la longueur du chemin critique. Pour des chemins critiques courts (inférieurs à 4 GD), les protocoles tardifs sont trop lents pour couvrir au plus juste le délai du chemin critique.

Pour des chemins critiques de longueur intermédiaire (entre 4 et 24 GD), le protocole Late-capture ressort car il est lent avec une petite surface. L'étude des autres protocoles précoces et tardifs et l'inclusion de retards asymétriques pourrait équilibrer les protocoles précoces et tardifs dans cette zone intermédiaire. En effet, les retards asymétriques raccourcissent la phase de RZ des protocoles précoces car le front descendant du fil de requête se propage rapidement. En revanche, les retards asymétriques ont une surface plus importante.

Finalement, pour des chemins critiques longs (supérieurs à 24 GD), le protocole Maximus obtient le meilleur FOM grâce à sa capacité à maintenir un haut débit.

4.1.4 Perspective : pipelines multi-protocoles

Le protocole Maximus repousse la frontière des protocoles connus. De nouvelles infrastructures devront être développées afin de terminer une bonne fois pour toutes l'exploration des protocoles asynchrones. YAHYA (2009) a montré que les pipelines asynchrones pouvaient être améliorés en sélectionnant pour chaque partie du pipeline le protocole le plus adapté. Ainsi, ajouter de nouveaux protocoles permet d'offrir plus d'opportunités pour rendre les circuits asynchrones des solutions compétitives en termes de surface et de performance.

Pour optimiser le choix des protocoles, une exploration heuristique s'appuyant sur des simulations est possible mais potentiellement non optimale. L'utilisation d'un modèle formel permet d'envisager une optimisation réelle en particulier avec l'apport de méthodes efficaces de recherche opérationnelle.

Dans la section suivante, nous présentons un modèle comportemental à base de réseau de Petri pour la conception, l'analyse et la vérification des circuits micropipelines.

4.2 Modèle comportemental à base de réseaux de Petri

L'objectif de ce modèle est d'aider le concepteur à analyser et corriger le contrôleur d'un circuit micropipeline. Le concepteur utilise l'approche par primitives (SECTION 2.2.2.1) et décrit son contrôleur par l'interconnexion de primitives de synchronisation. Un modèle formel est ensuite généré et vérifié. En cas de problème, par exemple avec un interblocage, les séquences fautives sont analysées pour permettre des corrections. Finalement, lorsque l'exécution du contrôleur est validée fonctionnellement, le modèle peut être utilisé à des fins d'optimisations, notamment lors de l'insertion des retards et de l'analyse des goulots d'étranglement.

Ainsi, nous recherchons les propriétés suivantes pour notre modèle :

1. Expressivité suffisante pour modéliser les primitives déterministes usuelles, en particulier les multiplexeurs et démultiplexeurs.
2. Compacité pour la rapidité de la simulation, de la détection d'erreurs et de l'analyse de performances.
3. Facilité d'interprétation. La compacité du modèle ne doit pas diminuer la capacité pour un concepteur ou un programme à l'interpréter.

Comparés aux modèles existants (TABLE 4.2), nous cherchons un modèle d'une compacité similaire à celle des *Timed Marked Graphs* (TMGs) mais plus expressif : il s'agit notamment de pouvoir modéliser des canaux sans mémorisation (comme le permettent les *Signal Transition Graphs* (STGs) et les *Static Dataflow Structures* (SDFSs)) mais aussi des primitives de sélection. En revanche, nous ne cherchons pas à modéliser des circuits non déterministes (ce que permettent les STGs).

TABLE 4.2 – Positionnement du modèle proposé.

Modèle	Expressivité	Compacité	Interprétabilité
<i>Signal Transition Graphs</i> (CHU 1986)	++	--	-
<i>Static Dataflow Structures</i> (SOKOLOV et al. 2007)	+		+
<i>Timed Marked Graphs</i> (CARMONA et al. 2009)	-	++	+
Cible de ce travail	+	+	+

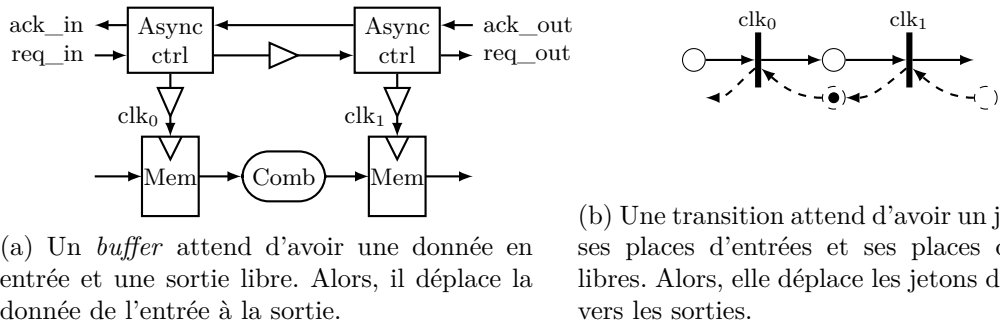
4.2.1 Modèle « canal équivalent » : fragments élémentaires des primitives

Tout comme le contrôleur est construit par interconnexion de primitives, le modèle est présenté ici comme l'interconnexion des fragments élémentaires associés à chacune des primitives. Dans la SECTION 4.2.2, nous justifions la construction de ce modèle.

4.2.1.1 Fragments associés aux primitives non conditionnelles

Afin de répondre aux objectifs, le modèle proposé est un PN qui reflète la structure des canaux dans le circuit. Les requêtes sont ainsi représentées de façon détaillée tandis que les acquittements sont modélisés de façon minimale.

L'idée de base est de remarquer la similarité entre le comportement d'un *buffer* dans un circuit asynchrone et la règle stricte de transition d'exécution d'un PN (FIGURE 4.13). Dans la règle stricte de transition, une transition n'est sensibilisée que lorsque toutes ses places d'entrées ont un jeton et que ses places de sorties sont vides. Une transition sensibilisée qui tire déplace les jetons de ses entrées vers ses sorties. Dans le cas d'un *buffer*, lorsqu'il a une nouvelle donnée en entrée et que sa sortie est libre de données, il déplace la donnée de son entrée à sa sortie. Ainsi, nous cherchons à représenter un *buffer* dans le circuit par une seule transition dans le PN.



(a) Un *buffer* attend d'avoir une donnée en entrée et une sortie libre. Alors, il déplace la donnée de l'entrée à la sortie.

(b) Une transition attend d'avoir un jeton dans ses places d'entrées et ses places de sorties libres. Alors, elle déplace les jetons des entrées vers les sorties.

FIGURE 4.13 – Correspondance comportementale entre *buffers* et transitions.

Modélisation des requêtes Comme les requêtes accompagnent les flots de données dans les canaux à données groupées, le modèle s'attache à représenter les requêtes des canaux³. Le modèle proposé associe à chaque *buffer*, union et fourche du contrôleur un fragment de PN représenté sur la FIGURE 4.14. Ainsi, chaque transition du PN correspond à la synchronisation de requêtes dans le contrôleur et chaque canal dans le contrôleur correspond à une place dans le PN.

Modélisation des acquittements Comme un *buffer* ne peut stocker qu'une donnée à la fois, la transition associée au *buffer* ne peut tirer qu'après le tir de tous les *buffers* qui suivent. En conséquence, il faut ajouter des chemins d'acquittements entre la transition, notée t , associée au registre et l'ensemble, noté P_t , des transitions de registres qui précèdent immédiatement t . Les transitions P_t sont donc celles associées aux registres qui sont connectés par un chemin combinatoire au registre de t . Ces chemins d'acquittement (en pointillés sur la FIGURE 4.14) assurent que les unions et fourches peuvent seulement

3. Cette caractéristique motive le nom de modèle « canal équivalent »

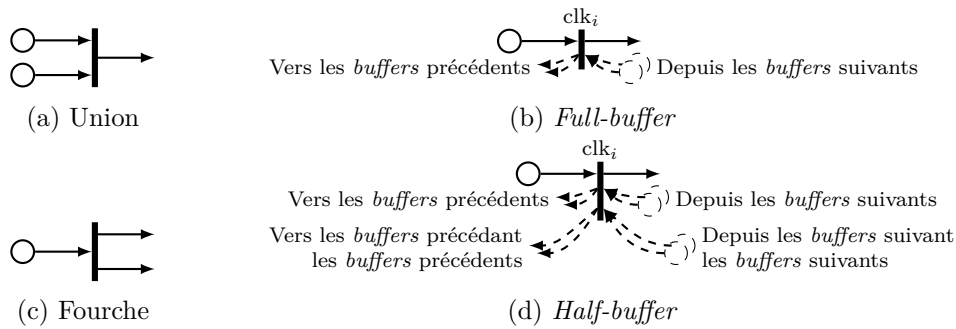


FIGURE 4.14 – Fragments de réseau de Petri associés aux unions, fourches et *buffers*. Les cercles et barres représentent respectivement des places et des transitions. Les arcs et places en pointillés représentent les chemins d’acquiescement.

propager des données sans les stocker (*slack* égal à 0) tandis que les *full-buffers* peuvent stocker au plus une donnée (*slack* égal à 1).

Dans le cas des *half-buffers* (*slack* égal à 1/2), il faut aussi ajouter un chemin d’acquiescement entre t et toutes les transitions de registre qui précèdent immédiatement celles de P_t . Ces chemins d’acquiescement supplémentaires traduisent le fait que deux *half-buffers* successifs peuvent contenir au plus une donnée.

4.2.1.2 Fragments associés aux primitives de sélection

La particularité de notre travail est de proposer une stratégie de modélisation pour les primitives de choix : les multiplexeurs et démultiplexeurs.

Modélisation des requêtes Les FIGURES 4.15a et 4.15c représentent les fragments de PN associés aux chemins de requêtes des multiplexeurs et démultiplexeurs. La valeur du i^e bit du signal de sélection « sel » est notée « sel(i) ». Ces valeurs binaires servent de poids pour les arcs du PN. Les poids sur les arcs modifient les règles de sensibilisation et de tir : la sensibilisation des transitions ne prend pas en compte le marquage des places connectées par des arcs de poids nul et le tir ne change pas le marquage de ces arcs⁴.

Modélisation des acquiescements Les chemins d’acquiescement à travers les multiplexeurs et démultiplexeur sont modélisés par les fragments des FIGURES 4.15b et 4.15d. Pour le multiplexeur, l’acquiescement depuis le canal de sortie est envoyé au canal de la sélection et aux seuls canaux sélectionnés. Ainsi, les poids des arcs de requêtes sont aussi utilisés sur ces arcs d’acquiescement (FIGURE 4.15b). Similairement, dans le cas du démultiplexeur, seul l’acquiescement des canaux sélectionnés sont envoyés au canal d’entrée

4. Dans le cas non binaire, le poids des arcs représente le nombre de jetons déplacés par le tir de la transition. Une transition est alors sensibilisée si et seulement si son tir résulte en un marquage valide.

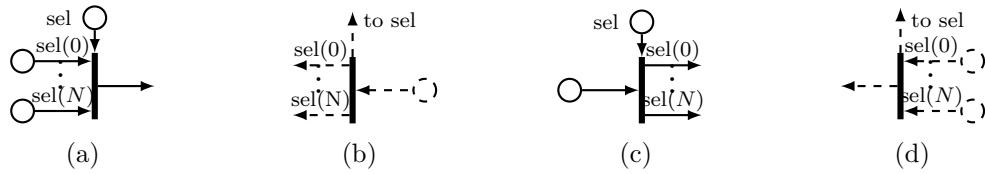


FIGURE 4.15 – Fragments de réseau de Petri associés aux multiplexeurs et démultiplexeurs. (a) et (b) requêtes et acquittements du multiplexeur. (c) et (d) requêtes et acquittements du démultiplexeur. Les « $sel(i)$ » sont les poids des arcs dépendants de la valeur de sélection.

et à celui de sélection. Les arcs d'entrées de la transition sont pondérés en conséquence (FIGURE 4.15d).

Modélisation des valeurs de sélection Les primitives de sélection orientent les flots de données en fonction des valeurs des données de sélection. Ces valeurs sont calculées à l'extérieur du contrôleur dans la partie chemin de données. La difficulté est alors de modéliser ces données et leurs changements sans faire une simulation, coûteuse, des chemins de données.

Dans notre modèle, le concepteur spécifie la suite des données attendue à l'entrée des blocs de sélection et associe un *buffer* ou une entrée à cette séquence. Ce *buffer* ou cette entrée doit correspondre au registre ou au port d'entrée du chemin de données qui contient la donnée de sélection. À chaque tir de la transition associée, les poids du multiplexeur ou démultiplexeur sont mis à jour selon la valeur du prochain élément de la suite.

4.2.2 Méthodologie de génération par simplification d'un STG de référence

Afin de prouver que le modèle est correct, nous le générons à partir du STG du contrôleur (correct par construction) en appliquant successivement des simplifications valides.

4.2.2.1 Définitions formelles sur les réseaux de Petri (PNs)

Le PN représentant le contrôleur est un quadruplet $G = (P, T, F, w)$ avec :

- P l'ensemble des places,
- T l'ensemble des évènements (transitions),
- $F \subset (P \times T) \cup (T \times P)$ l'ensemble des arcs (relation de flot) des places vers les transitions et des transitions vers les places.
- $w : F \rightarrow \{0, 1\}$ une fonction de poids. Pour $(x, y) \in F$, $w(x, y)$ est le poids de l'arc (x, y) .

Étant donné un PN, G , soient :

- $M : P \rightarrow \{0, 1\}$ le marquage représentant le nombre de jeton contenu dans chaque place. M_0 est le marquage initial.
- $\bullet y = \{x \in T \cup P / (x, y) \in F\}$ le pré-ensemble de $y \in P \cup T$, c'est-à-dire l'ensemble des prédécesseurs de y par F .
- $x^\bullet = \{y \in T \cup P / (x, y) \in F\}$ le post-ensemble de $x \in P \cup T$, c'est-à-dire l'ensemble des successeurs de x par F .

La dynamique d'exécution de G définit l'ensemble des transitions tirables pour chaque marquage M . Selon la règle stricte de transition, $t \in T$ peut tirer si et seulement si $\forall p \in \bullet t, M(p) \geq w(p, t)$ et $\forall p \in t^\bullet, M(p) \leq 1 - w(t, p)$. Si t tire, le nouveau marquage M' est tel que $\forall p \in \bullet t, M'(p) = M(p) - w(p, t)$ et $\forall p \in t^\bullet, M'(p) = M(p) + w(t, p)$. Nous définissons les notations suivantes :

- $M \xrightarrow{t} M'$ signifie que t est tirable dans le marquage M et que M' est le résultat du tir.
- Une séquence de tirs $\sigma = t_1.t_2.\dots.t_N$ représente une suite de tirs de transitions t_1, t_2, \dots, t_N s'il existe des marquages M_1, M_2, \dots, M_{N+1} vérifiant $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \dots \xrightarrow{t_N} M_{N+1}$.

Comme notre cadre de travail n'inclue que des primitives déterministes, tous les PN considérés sont des graphes marqués. On a donc $\forall p \in P, |\bullet p| = |p^\bullet| = 1$, où $|\cdot|$ désigne le cardinal d'un ensemble.

4.2.2.2 Stratégie de construction par simplifications itérées

Le modèle doit représenter le contrôleur asynchrone. Ainsi, le modèle doit spécifier avec exactitude les relations entre les événements pertinents du circuit de contrôle. Pour un circuit à données groupées avec des bascules, ces événements pertinents sont les fronts montant des horloges locales des registres. Nous appelons les transitions associées à ces événements « transitions observables ». Les autres événements, à savoir les « transitions internes », sont seulement utiles pour l'ordonnancement qu'elles imposent sur les transitions observables.

Dans la théorie des processus concurrents (MILNER 1983), deux processus sont faiblement (ou observationnellement) équivalents s'il existe une correspondance entre leurs états et transitions respectifs. Le graphe d'état que nous utilisons est sans choix (toute transition tirable sera un jour tirée) et il existe une bijection entre les transitions et les actions observées⁵. Ainsi, dans notre cas, nous pressentons qu'une équivalence de trace (BLOOM, ISTRAIL et MEYER 1988) implique l'équivalence observationnelle.

Parmi les transitions T , les transitions internes sont indistinctement notées 1 et le sous-ensemble des transitions observables est $T \setminus \{1\}$. L'opérateur $\sim : T^* \rightarrow (T \setminus \{1\})^*$

5. D'ailleurs, dans la suite, une transition du PN est assimilée à l'action qu'elle produit dans le circuit.

définit récursivement une séquence observable $\tilde{\sigma}$ à partir d'une séquence σ (4.5)⁶.

$$\tilde{\sigma}.t = \begin{cases} \tilde{\sigma} & \text{if } t = 1 \\ \tilde{\sigma}.t & \text{if } t \in T \setminus \{1\} \end{cases} \quad (4.5)$$

Définition 1 (Modèle correct). *Un modèle correct d'un circuit est tel que le modèle et le circuit ont le même ensemble de séquences observables (équivalence de trace, voir BLOOM, ISTRAIL et MEYER 1988).*

Il est facile de construire un STG correct du contrôleur (YAKOVLEV et al. 1996). On remarquera d'ailleurs que, suivant cette méthode, le STG global est l'interconnexion des STGs élémentaires de chaque primitive.

Dans la prochaine section, nous présentons des règles de simplification d'un PN préservant la correction des modèles.

Règles de simplification

La réduction des PNs est un sujet vastement couvert dans la littérature. Cette section présente quatre règles de simplification qui sont des cas particuliers de règles de réduction connues pour préserver la sûreté, la vivacité (MURATA 1989) et la couverture (*reachability*) (HAN et LEE 2003). Nous avons restreint ces règles pour préserver la correction du modèle. Les trois premières simplifications sont valides pour la règle générale de transition. Dans cette dynamique d'exécution, une transition peut tirer même si les places de sorties ne sont pas vides. Il s'agit de la dynamique du STG. La quatrième simplification est valide dans le cadre de la règle stricte de transition. Ainsi, elle doit être appliquée après toutes les autres.

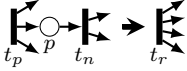
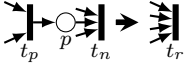
Transformation 1 (Modification du marquage). *Le marquage initial M_0 peut être remplacé par M'_0 si il existe une transition interne $t \in \{1\}$ telle que $M_0 \xrightarrow{t} M'_0$ ou $M'_0 \xrightarrow{t} M_0$.*

Démonstration. Les deux cas sont symétriques par réflexivité de l'équivalence de trace. Supposons que $M_0 \xrightarrow{t} M'_0$. Les graphes marqués sont persistents, c'est à dire qu'une transition tirable sera soit tirée soit restera tirable indéfiniment. Ainsi, pour chaque séquence $\sigma_0.t.\sigma_1$ à partir de M_0 où $\sigma_0 \in (T \setminus \{t\})^*$, la séquence $\sigma_0.\sigma_1$ est une séquence valide à partir de M'_0 . Réciproquement, pour toute séquence σ à partir de M'_0 , $t.\sigma$ est une séquence valide à partir de M_0 . Comme t est interne, les séquences observées sont identiques. \square

Transformation 2 (Simplification de transitions (TABLE 4.3)). *Soient t_p et t_n deux transitions telles que $\exists p \in P, p \in t_p^\bullet \cap \bullet t_n$. Les transitions peuvent être fusionnées si $M_0(p) = 0$ et si les conditions de la TABLE 4.3 sont vérifiées. Après la fusion, la transition résultante est marquée du label de l'éventuelle transition observable.*

6. Nous utilisons ici la notation classique des langages réguliers : $T^* = \bigcup_{n \in \mathbb{N}} T^n$ est l'union des puissances cartésiennes de T , c'est-à-dire, toutes les n -uplets finis ou infinis d'éléments de T .

TABLE 4.3 – Simplification de transitions. La table se lit de la manière suivante : « si t_n est interne, t_p est observable, et t_n possède un unique prédécesseur p , alors p et t_n peuvent être supprimés. Les successeurs de t_n deviennent les successeurs de t_p . »

t_p / t_n	t_n observable	t_n interne
t_p observable	Non applicable	si $\{p\} = \bullet t_n$ 
t_p interne	si $t_p \bullet = \{p\}$ 	si $\{p\} = \bullet t_n$ ou $t_p \bullet = \{p\}$

Démonstration. La fusion des deux actions t_n et t_p indique que pour un observateur externe, il n'y a pas de différence si les conséquences de t_n sont observées avant celles de t_p . Si une transition est observable, alors ses conséquences sont immédiates. Ainsi, la fusion n'est pas possible dans le cas Observable/Observable. On suppose dorénavant que t_n ou t_p est interne et $M_0(p) = 0$.

Supposons que $\{p\} = \bullet t_n$ et $t_n = 1$ (transition interne). Dans ce cas, si t_p tire, t_n devient tirable juste après. t_n étant interne, ses conséquences et celles de t_p peuvent donc être observées dans n'importe quel ordre. Formellement, pour toute séquence de tirs $\sigma = \sigma_0.t_p.\sigma_1.t_n.\sigma_2$ où $\sigma_0, \sigma_1 \in (T \setminus \{t_n\})^*$, la séquence $\sigma' = \sigma_0.t_p.t_n.\sigma_1.\sigma_2$ est aussi une séquence valide et $\tilde{\sigma} = \sigma' = \tilde{\sigma}_0.\tilde{t}_p.\tilde{\sigma}_1.\tilde{\sigma}_2$. L'application récursive de cette même propriété sur σ_2 prouve que $\{p\} = \bullet t_n$ est une condition suffisante pour la fusion dans les cas Interne/Interne et Observable/Interne.

Supposons que $t_p \bullet = \{p\}$ et $t_p = 1$. Dans ce cas, pour toute séquence $\sigma = \sigma_0.t_p.\sigma_1.\tilde{t}_n.\sigma_2$ où $\sigma_0, \sigma_1 \in (T \setminus \{t_n\})^*$, $\sigma' = \sigma_0.\sigma_1.t_p.t_n.\sigma_2$ est aussi une séquence valide et $\tilde{\sigma} = \sigma' = \tilde{\sigma}_0.\tilde{\sigma}_1.\tilde{t}_n.\tilde{\sigma}_2$. Similairement au cas précédent, cela prouve que $t_p \bullet = \{p\}$ est une condition suffisante pour la fusion dans les cas Interne/Interne et Interne/Observable. \square

Les conditions de la TABLE 4.3 sont suffisantes mais pas nécessaires. D'autres simplifications de transitions peuvent préserver les traces. Par exemple, t_p peut être simplifiée si $\bullet \bullet t_n = \{t_p\} \cup \bullet \bullet t_p$.

Transformation 3 (Simplification des places parallèles (FIGURE 4.16a)). Soient $p, p' \in P$ et $t_p, t_n \in T$. Si $p \neq p'$, $t_p \rightarrow p \rightarrow t_n$, $\{p, p'\} \subset t_p \bullet \cap \bullet t_n$ et $M_0(p) = M_0(p')$, alors p' peut être supprimée.

Démonstration. Soit G le graphe marqué et G' celui obtenu en enlevant p de G . Par induction sur la longueur des séquences de tir dans G et G' , pour tout marquage couvert M dans G , il vient que la valeur de $M(p) \oplus M(p')$ est préservée, que le marquage M' de G' obtenu à partir de M en supprimant p est aussi couvert et que t_n (resp. t_p) est tirable dans G' si et seulement si elle est tirable dans G . \square



FIGURE 4.16 – Simplification des places.

La dernière transformation permet de simplifier davantage le modèle mais nécessite d'utiliser la règle stricte de transition. Dans le cas d'un PN sûr, les règles stricte et générale sont équivalentes. En revanche, comme la dernière transformation ne préserve pas la sûreté, il n'est en principe pas possible de revenir à la règle générale.

Transformation 4 (Simplification des places complémentaires (FIGURE 4.16b)). *Soit $p, p' \in P$ et $t_p, t_n \in T$. Si $p \neq p'$, $p \in t_p \bullet \cap \bullet t_n$, $p' \in t_n \bullet \cap \bullet t_p$ et $M_0(p) = 1 - M_0(p')$, alors p' peut être supprimée.*

Démonstration. La preuve est similaire à celle de la simplification des places parallèles (Transformation 3). \square

Extension aux primitives de sélection

Pour modéliser les primitives de sélection, les poids des arcs deviennent des fonctions de N valeurs de sélections. Formellement, pour $(x, y) \in F$, le poids $w(x, y)$ peut soit être constant (0 ou 1) soit dépendre d'une valeur de sélection w_i (w_i ou $1 - w_i$) pour un $i \in [0, N - 1]$. La valeur de sélection w_i est définie comme suit :

- $t_i \in T \setminus \{1\}$ est la transition observable associée à la valeur de sélection.
- $k_i \in \mathbb{N}$ est un compteur initialisé à 0 puis incrémenté de 1 à chaque tir de t_i .
- S_i est une séquence binaire. La valeur de sélection $w_i = S_i(k_i)$ est la k_i^e valeur binaire de la séquence.

La simplification des transitions ne change pas. Les poids des transitions après fusion sont indiqués sur la FIGURE 4.17. La simplification de places parallèles nécessite que les poids soient identiques. La simplification de places complémentaires n'est pas définie avec des poids différents de 1.



FIGURE 4.17 – Simplification de transitions avec des poids sur les arcs.

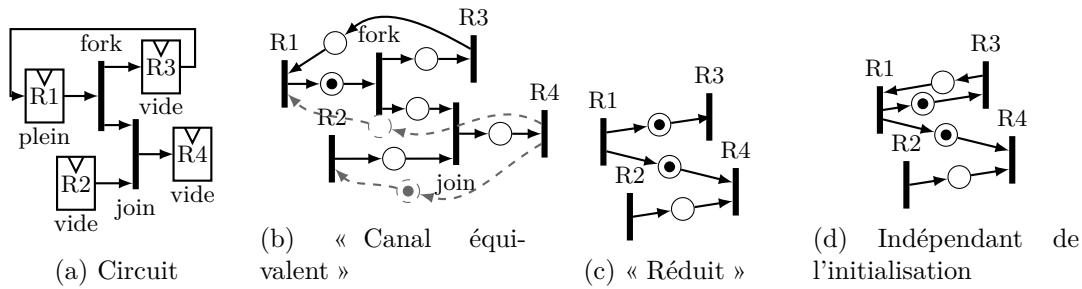


FIGURE 4.18 – Exemples de modèles obtenus à partir d'un même circuit.

4.2.2.3 Modèles résultant des simplifications

Le modèle « canal équivalent » présenté en « canal équivalent » est obtenu à partir du STG en appliquant les transformations sur les chemins d'acquittements jusqu'à ce qu'aucune transformation ne soit possible. La FIGURE 4.18b montre le modèle dérivé de la spécification de la FIGURE 4.18a. Le modèle « canal équivalent » contient une place par canal et les chemins d'acquittements (pointillés) sont simplifiés.

En terme de taille, ce modèle est plus gros que les TMGs (CARMONA et al. 2009) mais est plus expressif car il permet des canaux (arcs dans les TMGs) sans élément de mémorisation. Notre modèle est en revanche plus compact que les STGs (CHU 1986) et les SDFSs (SOKOLOV, POLIAKOV et YAKOVLEV 2007).

Un modèle « réduit » peut être obtenu en appliquant les règles de simplifications à tout le circuit (chemins de requête et d'acquittement) à partir du marquage initial. Si le circuit ne comporte pas de structures de choix, le modèle « réduit » consiste simplement à ajouter des places entre les transitions observables connectées par un chemin combinatoire (FIGURE 4.18c). Le résultat est un graphe dual d'un TMG. Le modèle « réduit » est compact mais moins lisible et aussi dépendant du marquage initial (initialisation du circuit).

Nous envisageons donc un troisième modèle indépendant de l'initialisation. Il serait obtenu en appliquant seulement les règles de simplification qui sont valides pour tous les marquages initiaux correspondant aux initialisations possibles du circuit. La FIGURE 4.18c montre un modèle indépendant de l'initialisation pour notre exemple.

4.2.3 Exemple : Conception et vérification d'un AES

Pour illustrer l'utilisation de notre modèle, nous nous intéressons à la conception d'un AES asynchrone. La FIGURE 4.19a représente la partie de chemins de données. Le message et la clé entrant effectuent 9 rondes dans la boucle avant de sortir pour subir les dernières opérations.

Dans l'approche de conception par primitives, c'est au concepteur d'élaborer la description du contrôleur (FIGURE 4.19b). Le modèle peut ensuite être automatiquement généré. Le PN de la FIGURE 4.19c est un modèle indépendant de l'initialisation. Notons

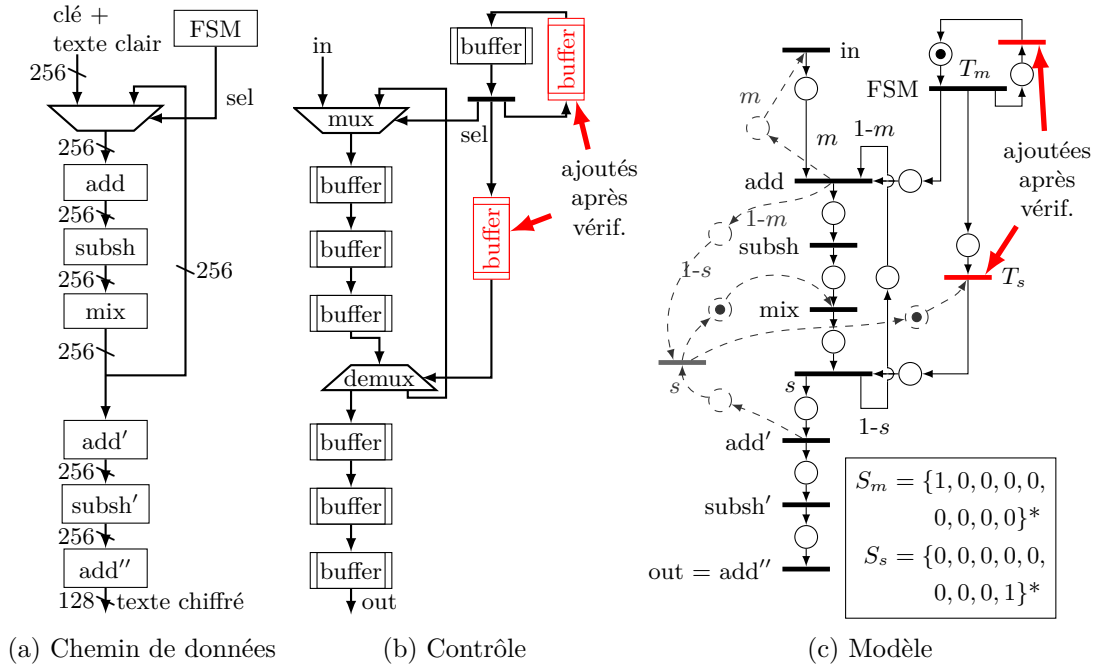


FIGURE 4.19 – Exemple de conception et vérification d'un AES.

que toutes les transitions à l'exception des transitions de requête et d'acquittement du démultiplexeur sont observables. La transition de requête du multiplexeur a été fusionnée avec celle du « add » et la transition de la FSM a été fusionnée avec celle de la fourche. Initialement, la FSM est associée aux deux séquences S_m et S_s . Elles régissent respectivement les poids du multiplexeur et du démultiplexeur. Le modèle obtenu est lisible, compact et permet de vérifier le contrôle.

Le modèle initial présente deux blocages. Premièrement, le registre de la FSM ne peut pas boucler sur lui-même. Deuxièmement, le jeton produit par la FSM est libéré lorsque la donnée a fini de traverser le multiplexeur et le démultiplexeur. Or, lorsque la donnée traverse le multiplexeur, le canal de la FSM vers le multiplexeur est libéré. En revanche, la donnée ne peut pas franchir complètement le démultiplexeur car elle est bloquée dans la boucle par le multiplexeur qui attend pour sa part un nouveau jeton de la FSM. Nous avons donc ajouté deux *buffers* pour corriger ces problèmes. La transition T_s ajoutée devient la transition associée à la séquence S_s .

Le blocage de la boucle de la FSM aurait pu facilement être détecté et corrigé directement au niveau des primitives. En effet, une boucle contenant N données doit contenir au moins $N + 1$ *full-buffers* (ou $N + 2$ *half-buffers*). Pour les blocages tels le deuxième exemple, la correction automatique est beaucoup plus délicate. Néanmoins, l'automatisation de la correction pourrait être utile pour trouver la solution corrigeant le circuit au plus petit coût en matériel et en ralentissement.

Dans cette section, nous avons souligné un mécanisme générique de génération de

modèles à base de PN. Selon les besoins, différents ensembles de transitions observables peuvent être définis et mener ainsi à différents modèles. Le modèle utilisé peut être choisi selon l'étape du flot de conception. Le modèle « réduit » est pratique pour une vérification rapide du circuit. Un modèle indépendant de l'initialisation permet d'explorer les initialisations possibles sur le même réseau de Petri. Finalement, le modèle « canal équivalent » est bien adapté à l'interprétation des traces fautives et à l'évaluation des performances en utilisant des transitions temporalisées dans les chemins de requêtes.

4.3 Conclusion

Ce chapitre compare formellement les caractéristiques temporelles des différentes classes de protocoles asynchrones. Cette analyse, corroborée par des simulations, montre que les protocoles tardifs permettent d'améliorer la surface, le débit et la consommation des pipelines asynchrones lorsque les chemins critiques sont longs.

Pour aider la vérification et l'analyse des circuits asynchrones à données groupées, nous avons présenté un nouveau modèle de représentation des circuits asynchrone exploitant un réseau de Petri qui représente de manière compacte les canaux de communication. Ce modèle propose aussi un mécanisme pour représenter les valeurs des données des structures de sélection. Enfin, ce modèle formel, simplifiable, permet de représenter efficacement nos circuits, de les vérifier et de faire aisément la synthèse des contrôleurs pour les circuits micropipelines.

Conclusion

Pour résumer, l'échantillonnage non uniforme et l'électronique asynchrone sont deux techniques événementielles connues pour permettre des réductions de la consommation d'un à trois ordres de grandeur. Afin d'exploiter ce potentiel, nous avons élaboré le flot de conception ALPS : outils architecturaux pour systèmes basse consommation basés sur les événements (*Architectural tools for Low-Power event-driven Systems*). La FIGURE 5.1 résume les étapes du flot que nous avons traitées et celles à venir.

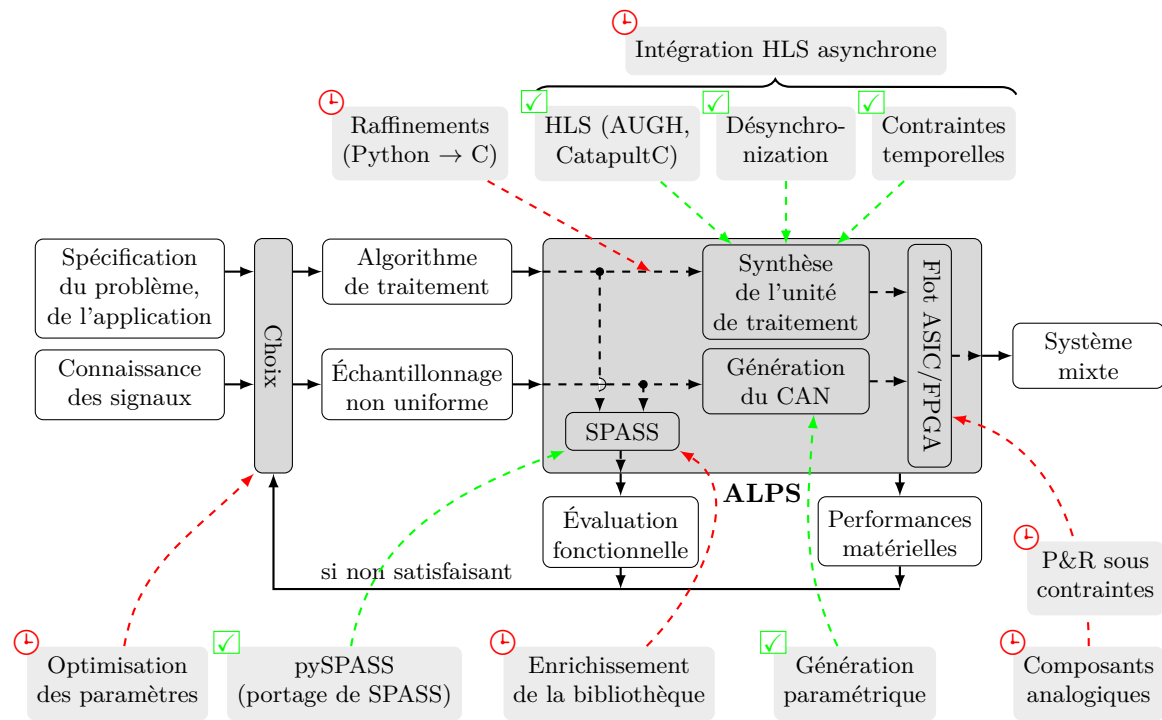


FIGURE 5.1 – Bilan du travail effectué sur ALPS et perspectives. Les étapes effectuées sont en vert (✓) et les travaux à venir sont en rouge (⊕).

Conception au niveau algorithmique

La conception de l'algorithme de traitement et le choix des paramètres d'échantillonnage sont faits itérativement au niveau de la spécification et validés par simulation. Cette

partie à haut niveau s'appuie d'une part sur une connaissance *a priori* de l'application visée, soit sous la forme de base de données de signaux, soit sous la forme de propriétés statistiques. Le développement s'appuie d'autre part sur les bibliothèques pySPASS (Python) ou SPASS (Matlab) implémentant les principales primitives d'échantillonnage et de traitement du signal dans le cadre non uniforme. Ces boîtes à outils permettent l'évaluation par simulation de l'échantillonnage et du traitement dédié à l'application.

Dans le cas d'un ASA, ALPS permet de quantifier l'impact du schéma d'interpolation sur la qualité du signal ré-échantillonné (niveau algorithmique) et la consommation en énergie (niveau circuit). Nous avons ainsi pu déterminer que le schéma de ré-échantillonnage ZOH était celui qui introduisait le moins d'erreur et correspondait au circuit le plus petit et le moins consommant.

Synthèse de haut niveau asynchrone

Une fois les paramètres d'échantillonnage choisis, la description matérielle du CAN-TN correspondant est générée. La description matérielle asynchrone de l'unité traitement est alors produite automatiquement à partir l'algorithme traduit en C. Cette HLS s'appuie sur des outils de HLS synchrone, comme AUGH ou CatapultC, qui génère un chemin de données et une FSM synchrones. Pour remplacer la FSM originelle générée par les outils de HLS, nous avons introduit une nouvelle architecture d'AFSM basée sur des contrôleurs de pipelines asynchrones.

Dans le cas du filtrage d'un ECG, le filtrage non uniforme permet de diviser le nombre d'échantillons par 7 et de diviser la consommation en énergie de la partie traitement par des facteurs de 1,5 à 4. Le filtrage est ici considéré seul, sans connaissance des traitements suivants. Ainsi, les choix des paramètres sont conservatifs et les gains en énergie par conséquent limités.

La méthode de transformation de la partie contrôle (de FSM à AFSM) proposée dans ALPS est applicable à de nombreux outils de HLS de pointe. Les circuits asynchrones obtenus en combinant la méthode de désynchronisation d'ALPS avec les outils de HLS AUGH et CatapultC consomment respectivement 32% et 39% moins d'énergie que les circuits obtenus avec les seuls outils de HLS.

Perspectives pour ALPS

La complétion de l'automatisation du flot passe par deux étapes à court terme : l'implantation sur FPGA et la traduction de l'algorithme du langage à haut niveau (Matlab ou Python) vers C. L'implantation sur FPGA nécessite des techniques spécifiques pour implanter les portes de Muller (HO et al. 2002) et pour satisfaire les RTCs, par exemple itérativement (GIBILUKA, MOREIRA et CALAZANS 2015). La traduction de l'algorithme est plus ardue, notamment pour déterminer automatiquement la convention de signe, le nombre de bits et la position de la virgule à partir de flottants faiblement typés.

À moyen terme, les résultats d'ALPS peuvent être améliorés par une interaction plus forte entre les étapes de HLS et de désynchronisation. En particulier, l'outil de

HLS devra pouvoir intégrer la génération de l'AFSM et adapter les fonctions de cout au cas asynchrone. Pour ce faire, les outils libres de HLS tels AUGH ou LegUp seront à privilégier.

Enfin, les bibliothèques à haut niveau pourront être étendues pour inclure davantage d'algorithmes de traitement et de schémas d'échantillonnage. Le flot devra être étendu pour générer les CANs correspondants. Plus généralement, le flot pourra être adapté pour d'autres flots de données non uniformes, par exemple pour traiter les données venant de capteurs d'image événementiels.

Protocoles tardifs Late-capture et Maximus

Notre comparaison des protocoles précoces et tardifs a montré que les protocoles tardifs sont une opportunité intéressante pour la réduction de la surface, l'amélioration du débit et de l'efficacité énergétique des micropipelines, en particulier dans les pipelines avec des étages où les chemins critiques sont longs. Pour un étage avec un chemin critique long de 15 portes logiques, les protocoles Late-capture et Maximus ont respectivement permis de réduire la surface du contrôleur de 47% et 25% et l'énergie consommée de 45% et 30% comparés au micropipeline typique. Les débits ont été quant à eux augmentés respectivement de 10% et 40%.

Le protocole Maximus montre que les environnements existants dédiés aux protocoles précoces ne sont pas suffisants pour étudier les protocoles tardifs. Le principe des environnements existants consiste à trouver le protocole précoce de concurrence maximale et les réductions de concurrence autorisées pour en déduire l'ensemble des protocoles précoces. Une méthode similaire pourra être appliquée aux protocoles tardifs et ainsi compléter la palette des protocoles à disposition des concepteurs.

Conception, vérification et optimisation des micropipelines

Les modèles formels de circuits asynchrones existants sont soit trop complexes et donc longs à vérifier, soit pas assez expressifs car ils ne permettent pas de modéliser les structures de choix. Le modèle « canal exact » présenté offre un compromis entre expressivité et compacité. De plus, la méthode formelle de génération du modèle permet de facilement ajuster le compromis en fonction du besoin.

Pour être utilisable en pratique, le modèle devra être généré automatiquement, par exemple, à partir d'une description de l'interconnexion des primitives. De plus, l'outil d'analyse du réseau de Petri devra supporter la mise à jour dynamique des poids des arrêtes telle que proposée dans notre modèle.

Outre la vérification fonctionnelle des micropipelines, les transformations proposées peuvent être étendues pour conserver les propriétés temporelles du réseau de Petri. Ainsi, le modèle pourra servir à analyser et à optimiser le débit du circuit, sa consommation, ses émissions électromagnétiques avec la répartition temporelle des pics de courant (GERMAIN, ENGELS et FESQUET 2017) en jouant sur le mélange des protocoles et le choix des retards.

Perspectives pour l'utilisation industrielle des techniques événementielles

L'échantillonnage non uniforme et l'électronique asynchrone semblent de bonnes opportunités pour répondre aux défis actuels du nombre croissant de données générées et aux budgets énergétiques limités. Pourtant, elles sont peu adoptées et suscitent un faible intérêt, voire de la défiance. En effet, peu de concepteurs maîtrisent ces techniques événementielles suffisamment pour les utiliser et les évaluer dans leurs applications en un temps raisonnable.

Le flot ALPS vise à fournir facilement et rapidement une estimation ou un démonstrateur permettant d'évaluer l'efficacité de plateformes réalisant l'échantillonnage non uniforme avec l'unité de traitement dédiée adéquate. Notre travail a posé les fondements du flot et a produit des solutions aux problématiques liées au matériel. Il s'agit donc d'un premier pas pour permettre à l'industrie de s'appuyer sur un échantillonnage et une logique basée sur des événements pour produire des systèmes peu énergivores utilisables pour l'IoT par exemple.

Les études de cas menées ont confirmé que pour exploiter pleinement le potentiels de ces techniques événementielles, il faut une connaissance fine des besoins de l'application pour que l'échantillonnage extrait du signal l'information la plus pertinente et que l'algorithme la traite le plus efficacement.

Nous laissons ces problématiques à haut niveau aux spécialistes du traitement du signal, des statistiques ou de l'intelligence artificielle. En revanche, ALPS leur permettra d'évaluer leurs solutions sans nécessiter d'être un spécialiste en microélectronique. Ainsi, plus qu'un premier pas, ALPS est une rampe de lancement pour le développement de techniques événementielles et vers une approche énergétiquement bien plus efficace de nos systèmes électroniques et de communication.

Glossaire

A

AES *Advanced Encryption System*.

AFSM machine à états asynchrones (*Asynchronous Finite State Machine*).

AHA *American Health Association*.

ASA Algorithme de Sélection d'Activité (QAISAR, FESQUET et RENAUDIN 2014).

ASIC *application specific integrated circuit*.

C

CAN Convertisseur Analogique/Numérique.

CAN-TN Convertisseur Analogique/Numérique à Traversée de Niveaux (ALLIER et al. 2005).

CG désactivation de l'horloge (*Clock Gating*).

CMOS semi-conducteur métal-oxyde complémentaire (*Complementary Metal-Oxide-Semiconductor*).

CNA Convertisseur Numérique/Analogique.

COTS composants sur étagère (*Components On The Shelf*).

CUSA Cellule Universelle pour Séquences Asynchrones (DAVID 1977).

D

DDD décomposition dirigée par les données (*Data-Driven Decomposition*).

DSPU unité de traitement numérique du signal (*Digital Signal Processing Unit*).

E

ECG électrocardiogramme.

F

FIFO premier arrivé premier servi (*First-in First-out*).

FPGA *field-programmable gate array*.

FOM score de mérite (*Figure Of Merit*).

FSM machine à états finis (*Finite State Machine*).

H

HDL langage de description de matériel (*Hardware Description Language*).

HLS synthèse de haut niveau (*High-Level Synthesis*).

I

IoT internet des objets (*Internet of Things*).

L

LIN interpolation linéaire.

LJM modèle Liaisons/Articulations (*Links/Joints Model*) (RONCKEN et al. 2015).

N

NN plus proche voisin (*Nearest Neighbour*)

NUS échantillonné non uniformément (*Non-Uniformly Sampled*).

P

PGCD plus grand commun dénominateur.

PN réseau de Petri (*Petri Net*) (PETRI 1966).

Q

QDI quasiment insensible aux délais (*Quasi Delay Insensitive*).

R

RAM mémoire à accès aléatoire (*Random Access Memory*).

RIF à réponse impulsionnelle finie.

RIFI RIF interpolé (AESCHLIMANN et al. 2004).

RMS *Root Mean Square*. $RMS(x) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} x_i^2}$

ROC fonction d'efficacité du récepteur (*Receiver Operating Characteristic*).

ROM mémoire en lecture seule (*Read Only Memory*).

RTC contrainte temporelle relative (*Relative Timing Constraint*).

RTL au niveau des transferts de registres (*Register Transfer Level*).

RZ retour à zéro.

S

SDFS *Static Dataflow Structure* (SPARSØ et FURBER 2002 ; SOKOLOV, POLIAKOV et YAKOVLEV 2007).

SLI interpolation linéaire simplifiée (*Simplified Linear Interpolation*).

SNR rapport signal sur bruit (*Signal-to-Noise Ratio*).

STFB *Single Track Full Buffer* (FERRETTI et BEEREL 2006).

STA analyse statique de timing (*Static Timing Analysis*) (BHASKER et CHADHA 2009).

STG *Signal Transition Graph* (CHU 1986).

T

TLM modèle au niveau transactionnel (*Transaction Level Model*).

TMG *Timed Marked Graph* (CARMONA et al. 2009).

TN traversée de niveaux.

W

WCHB *Weak Condition Half Buffer* (SEITZ 1980).

X

xMAS *eXecutable Micro Architectural Specification* (CHATTERJEE, KISHINEVSKY et OGRAS 2012).

Z

ZOH constant d'ordre zéro (*Zero Order Hold*).

Bibliographie

- AESCHLIMANN, F., E. ALLIER, L. FESQUET et M. RENAUDIN (2004). « Asynchronous FIR filters : towards a new digital processing chain ». In : *10th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 198-206. DOI : 10.1109/ASYNC.2004.1299303.
- AKOPYAN, F., R. MANOHAR et A. B. APSEL (2006). « A level-crossing flash asynchronous analog-to-digital converter ». In : *12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 11 pp.-22. DOI : 10.1109/ASYNC.2006.5.
- ALLIER, E., G. SICARD, L. FESQUET et M. RENAUDIN (2005). « Asynchronous level crossing analog to digital converters ». In : *Measurement* 37.4. 8th Workshop on ADC Modelling and Testing, p. 296 -309. ISSN : 0263-2241. DOI : <http://dx.doi.org/10.1016/j.measurement.2005.03.002>.
- ANANIAN, C Scott (1999). « The static single information form ». Thèse de doct. Massachusetts Institute of Technology.
- BARDSLEY, Andrew (1998). « Balsa : An asynchronous circuit synthesis system ». Mém. de mast. University of Manchester.
- BERKEL, Kees van (1993). *Handshake Circuits : An Asynchronous Architecture for VLSI Programming*. New York, NY, USA : Cambridge University Press. ISBN : 0-521-45254-6.
- BHASKER, J. et Rakesh CHADHA (2009). *Static Timing Analysis for Nanometer Designs : A Practical Approach*. 1st. Springer Publishing Company, Incorporated. ISBN : 0387938192, 9780387938196.
- BIDÉGARAY-FESQUET, B. et L. FESQUET (2016). « Levels, peaks, slopes... which sampling for which purpose ? » In : *2nd International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, p. 1-6. DOI : 10.1109/EBCCSP.2016.7605261.
- BIDÉGARAY-FESQUET, Brigitte et Laurent FESQUET (2011). « Non-Uniform Filter Design in the Log-Scale ». In : *9th International Conference on Sampling Theory and Applications (SAMP TA)*, art-150.
- BIRTWISTLE, G. et K. S. STEVENS (2008). « The Family of 4-phase Latch Protocols ». In : *14th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 71-82. DOI : 10.1109/ASYNC.2008.19.
- BLOOM, B., S. ISTRAIL et A. R. MEYER (1988). « Bisimulation Can'T Be Traced ». In : *15th ACM Symposium on Principles of Programming Languages (SIGPLAN-*

- SIGACT*). POPL '88. San Diego, California, USA : ACM, p. 229-239. ISBN : 0-89791-252-7. DOI : 10.1145/73560.73580.
- BOUESSE, Fraïdy, Marc RENAUDIN et Gilles SICARD (2005). « Improving DPA Resistance of Quasi Delay Insensitive Circuits Using Randomly Time-shifted Acknowledgment Signals ». In : *13th International Conference on Very Large Scale Integration of System on Chip (VLSI-SoC)*. Sous la dir. de Ricardo REIS, Adam OSSEIRAN et Hans-Joerg PFLEIDERER. Boston, MA : Springer US, p. 11-24. ISBN : 978-0-387-73661-7. DOI : 10.1007/978-0-387-73661-7_2.
- BOUESSE, Ghislain Fraïdy et al. (2007). « Asynchronous logic Vs Synchronouos logic : Concrete results on electromagnetic emissions and conducted susceptibility ». In : *6th International workshop on electromagnetic compatibility of integrated circuits (EMC Compo)*. Torino, Italy, p. 99-102.
- BOUSSELJOT, R, D KREISELER et A SCHNABEL (1995). « Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet ». In : *Biomedizinische Technik/Biomedical Engineering* 40.s1, p. 317-318.
- CARMONA, J., J. CORTADELLA, M. KISHINEVSKY et A. TAUBIN (2009). « Elastic Circuits ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.10, p. 1437-1455. ISSN : 0278-0070. DOI : 10.1109/TCAD.2009.2030436.
- CHATTERJEE, S., M. KISHINEVSKY et U. Y. OGRAS (2012). « xMAS : Quick Formal Modeling of Communication Fabrics to Enable Verification ». In : *IEEE Design & Test of Computers* 29.3, p. 80-88. ISSN : 0740-7475. DOI : 10.1109/MDT.2012.2205998.
- CHENG, Fu-Chiung, Yuan-Feng CHEN, Shu-Chuan HUANG et Ching Yang HUANG (2014). « Synthesis of QDI FSMs from Synchronous Specifications ». In : *20th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 61-68. DOI : 10.1109/ASYNC.2014.16.
- CHRISTENSEN, Søren et Niels Damgaard HANSEN (1993). « Coloured Petri nets extended with place capacities, test arcs and inhibitor arcs ». In : *14th International Conference Application and Theory of Petri Nets*, p. 186-205. ISBN : 978-3-540-47759-4. DOI : 10.1007/3-540-56863-8_47.
- CHU, Tam-Anh (1986). « On the models for designing VLSI asynchronous digital systems ». In : *INTEGRATION, the VLSI journal* 4.2, p. 99-113.
- CORTADELLA, J., A. KONDRATYEV, L. LAVAGNO et C.P. SOTIRIOU (2006). « Desynchronization : Synthesis of Asynchronous Circuits From Synchronous Specifications ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.10, p. 1904-1921. ISSN : 0278-0070. DOI : 10.1109/TCAD.2005.860958.
- CORTADELLA, J. et al. (2010). « Narrowing the margins with elastic clocks ». In : *IEEE International Conference on Integrated Circuit Design and Technology*, p. 146-150. DOI : 10.1109/ICICDT.2010.5510273.
- CORTADELLA, Jordi et al. *Petrify : a tool for synthesis of Petri Nets and asynchronous circuits*. Software, <http://www.cs.upc.edu/~jordicf/petrify/> [accessed 2016-04-11].

- COUSSY, P., D.D. GAJSKI, M. MEREDITH et A. TAKACH (2009). « An Introduction to High-Level Synthesis ». In : *IEEE Design & Test of Computers* 26.4, p. 8-17. ISSN : 0740-7475. DOI : 10.1109/MDT.2009.69.
- COUSSY, Philippe et Adam MORAWIEC (2008). *High-Level Synthesis : From Algorithm to Digital Circuit*. 1st. Springer Publishing Company, Incorporated. ISBN : 1402085877, 9781402085871.
- CURTINHAS, T., D.L. OLIVEIRA, D. BOMPEAN, L.A. FARIA et L. ROMANO (2014). « SICARELO : A tool for synthesis of locally-clocked extended burst-mode asynchronous Controllers ». In : *5th IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, p. 1-4. DOI : 10.1109/LASCAS.2014.6820270.
- DAVID, R. (1977). « Modular Design of Asynchronous Circuits Defined by Graphs ». In : *IEEE Transactions on Computers* C-26.8, p. 727-737. ISSN : 0018-9340. DOI : 10.1109/TC.1977.1674910.
- ESPARZA, Javier (1998). « Decidability and complexity of Petri net problems — An introduction ». In : *Lectures on Petri Nets I : Basic Models : Advances in Petri Nets*. Sous la dir. de Wolfgang REISIG et Grzegorz ROZENBERG. Berlin, Heidelberg : Springer, p. 374-428. ISBN : 978-3-540-49442-3. DOI : 10.1007/3-540-65306-6_20.
- FERRETTI, M. et P. A. BEEREL (2006). « High performance asynchronous design using single-track full-buffer standard cells ». In : *IEEE Journal of Solid-State Circuits* 41.6, p. 1444-1454. ISSN : 0018-9200. DOI : 10.1109/JSSC.2006.874308.
- FESQUET, L. et B. BIDÉGARAY-FESQUET (2010a). « IIR digital filtering of non-uniformly sampled signals via state representation ». In : *Signal Processing* 90.10, p. 2811 -2821. ISSN : 0165-1684. DOI : <http://dx.doi.org/10.1016/j.sigpro.2010.03.030>.
- FESQUET, L., J. SIMATIC, A. DARWISH et A. CHERKAOUI (2016). *Event-based design for mitigating energy in electronic systems*. Keynote talk at OAGM & ARW Joint Workshop on “Computer Vision and Robotics”.
- FESQUET, Laurent et Brigitte BIDÉGARAY-FESQUET (2010b). *SPASS 2.0 : Signal Processing for ASynchronous Systems*. Software, <http://ljk.imag.fr/membres/Brigitte.Bidegaray/SPASS/> [accessed 2014-01-28].
- FONTAINE, Ludovic et José RAGOT (2001). « Filtrage de signaux à échantillonnage irrégulier ». In : *Traitement du signal*.
- FURBER, S.B. et P. DAY (1996). « Four-phase micropipeline latch control circuits ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 4.2, p. 247-253. ISSN : 1063-8210. DOI : 10.1109/92.502196.
- GARCIA, Kledermon (2015). « Behavioral synthesis of optimized asynchronous circuits ». (in Portuguese, available at <http://www.bdita.bibl.ita.br/tesesdigitais/69866.pdf>). Mém. de mast. Instituto Tecnológico de Aeronáutica – ITA, Brazil.
- GERMAIN, S., S. ENGELS et L. FESQUET (2017). « Event-Based Design Strategy for Circuit Electromagnetic Compatibility ». In : *3rd International Conference on Event-based Control, Communication and Signal Processing (EBCCSP)*. (to be published).
- GIBILUKA, M., M. T. MOREIRA et N. L. V. CALAZANS (2015). « A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework ». In : *Euro-*

- micro Conference on Digital System Design (DSD)*, p. 79-86. DOI : 10.1109/DSD.2015.104.
- GINOSAR, R. (2003). « Fourteen ways to fool your synchronizer ». In : *9th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 89-96. DOI : 10.1109/ASYNC.2003.1199169.
- GOLDBERGER, Ary L. et al. (2000). « PhysioBank, PhysioToolkit, and PhysioNet ». In : *Circulation* 101.23, e215-e220. ISSN : 0009-7322. DOI : 10.1161/01.CIR.101.23.e215. eprint : <http://circ.ahajournals.org/content/101/23/e215.full.pdf>.
- GREITANS, Modris, Rolands SHAVELIS, Laurent FESQUET et Taha BEYROUTHY (2011). « Combined peak and level-crossing sampling scheme ». In : *9th International Conference on Sampling Theory and Applications (SampTA)*.
- GRIMALDI, R. L., S. RODRIGUEZ et A. RUSU (2011). « A 10-bit 5kHz level-crossing ADC ». In : *20th European Conference on Circuit Theory and Design (ECCTD)*, p. 564-567. DOI : 10.1109/ECCTD.2011.6043596.
- GRÖCHENIG, Karlheinz (1992). « Reconstruction Algorithms in Irregular Sampling ». In : *Mathematics of Computation* 59.199, p. 181-194. ISSN : 00255718, 10886842.
- GUAN, K. M. et A. C. SINGER (2007). « Opportunistic Sampling of Bursty Signals by Level-Crossing - an Information Theoretical Approach ». In : *41st Conference on Information Sciences and Systems*, p. 701-707. DOI : 10.1109/CISS.2007.4298396.
- GUAN, Karen M, Suleyman S KOZAT et Andrew C SINGER (2008). « Adaptive reference levels in a level-crossing analog-to-digital converter ». In : *Journal on Advances in Signal Processing (EURASIP)* 2008, p. 183.
- HAN, Z. et G. B. LEE (2003). « Reduction method for reachability analysis of Petri nets ». In : *Tsinghua Science and Technology* 8.2, p. 231-235.
- HAND, D. et al. (2015). « Blade – A Timing Violation Resilient Asynchronous Template ». In : *21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 21-28. DOI : 10.1109/ASYNC.2015.13.
- HANSEN, J. et M. SINGH (2010). « A Fast Branch-and-Bound Approach to High-Level Synthesis of Asynchronous Systems ». In : *16th IEEE Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 107-116. DOI : 10.1109/ASYNC.2010.14.
- HO, Quoc Thai, Jean-Baptiste RIGAUD, Laurent FESQUET, Marc RENAUDIN et Robin ROLLAND (2002). « Implementing Asynchronous Circuits on LUT Based FPGAs ». In : *12th International Conference on Field-Programmable Logic and Applications (FPL)*. Sous la dir. de Manfred GLESNER, Peter ZIPF et Michel RENOVELL. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 36-46. ISBN : 978-3-540-46117-3. DOI : 10.1007/3-540-46117-5_6.
- HOARE, Charles Antony Richard (1978). « Communicating sequential processes ». In : *Communications of the ACM* 21.8, p. 666-677.
- HOLLAAR, L. A. (1982). « Direct Implementation of Asynchronous Control Units ». In : *IEEE Transactions on Computers* C-31.12, p. 1133-1141. ISSN : 0018-9340. DOI : 10.1109/TC.1982.1675937.
- IFEACHOR, Emmanuel C. et Barrie W. JERVIS (2002). *Digital Signal Processing : A Practical Approach*. 2nd. Pearson Education. ISBN : 0201596199.

- KHOMENKO, V., D. SOKOLOV, A. MOKHOV et A. YAKOVLEV (2017). « WAITX : An Arbiter for Non-persistent Signals ». In : *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 33-40. DOI : 10.1109/ASYNC.2017.8.
- KLIGFIELD, Paul et al. (2007). « Recommendations for the Standardization and Interpretation of the Electrocardiogram ». In : *Circulation* 115.10, p. 1306-1324.
- KONDRATYEV, A. et K. LWIN (2002). « Design of asynchronous circuits using synchronous CAD tools ». In : *IEEE Design & Test of Computers* 19.4, p. 107-117. ISSN : 0740-7475. DOI : 10.1109/MDT.2002.1018139.
- KOZMIN, K., J. JOHANSSON et J. DELSING (2009). « Level-Crossing ADC Performance Evaluation Toward Ultrasound Application ». In : *IEEE Transactions on Circuits and Systems I : Regular Papers* 56.8, p. 1708-1719. ISSN : 1549-8328. DOI : 10.1109/TCSI.2008.2010094.
- LE PELLETER, Tugdual (2015). « Méthode de discrétisation adaptée à une logique événementielle pour l'ultra-faible consommation : application à la reconnaissance de signaux physiologiques ». Thèse de doct. Institut National Polytechnique de Grenoble - INPG.
- LINES, Andrew Matthew (1998). *Pipelined asynchronous circuits*. Rapp. tech. California Institute of Technology.
- MANNAKKARA, Chamika et Tomohiro YONEDA (2010). « Asynchronous pipeline controller based on early acknowledgement protocol ». In : *IEICE Transactions on Information and Systems* 93.8, p. 2145-2161.
- MANOHAR, Rajit et Alain J. MARTIN (1995). *Quasi-Delay-Insensitive Circuits Are Turing-Complete*. Rapp. tech. Pasadena, CA, USA : California Institute of Technology.
- MANORANJAN, J. V. et K. S. STEVENS (2016). « Qualifying Relative Timing Constraints for Asynchronous Circuits ». In : *22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 91-98. DOI : 10.1109/ASYNC.2016.23.
- MARK, J. et T. TODD (1981). « A Nonuniform Sampling Approach to Data Compression ». In : *IEEE Transactions on Communications* 29.1, p. 24-32. ISSN : 0090-6778. DOI : 10.1109/TCOM.1981.1094872.
- MARTIN, Alain J. (1990a). « Programming in VLSI : From Communicating Processes to Delay-insensitive Circuits ». In : *Developments in Concurrency and Communication*. Sous la dir. de C. A. R. HOARE. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., p. 1-64. ISBN : 0-201-17232-1.
- (1990b). « The Limitations to Delay-insensitivity in Asynchronous Circuits ». In : *6th MIT Conference on Advanced Research in VLSI*. AUSCRYPT '90. Boston, Massachusetts, USA : MIT Press, p. 263-278. ISBN : 0-262-04109-X.
- MARTÍNEZ-NUEVO, Pablo, Sharvil PATIL et Yannis TSIVIDIS (2015). « Derivative Level-Crossing Sampling ». In : *IEEE Transactions on Circuits and Systems II : Express Briefs* 62.1, p. 11-15.
- MARVASTI, Farokh (2001). *Nonuniform sampling : theory and practice*. Springer Science & Business Media. DOI : 10.1007/978-1-4615-1229-5.

- MCGEE, Peggy B et Steven M NOWICK (2005). « A lattice-based framework for the classification and design of asynchronous pipelines ». In : *42nd annual Design Automation Conference (DAC)*. ACM, p. 491-496.
- MILNER, Robin (1983). « Calculi for synchrony and asynchrony ». In : *Theoretical Computer Science* 25.3, p. 267 -310. ISSN : 0304-3975. DOI : [http://dx.doi.org/10.1016/0304-3975\(83\)90114-7](http://dx.doi.org/10.1016/0304-3975(83)90114-7).
- MISKOWICZ, Marek (2006). « Send-on-delta concept : an event-based data reporting strategy ». In : *sensors* 6.1, p. 49-63.
- MORENO, Alberto et Jordi CORTADELLA (2017). « Synthesis of All-Digital Delay Lines ». In : *23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. (to be published).
- MURATA, T. (1989). « Petri nets : Properties, analysis and applications ». In : *Proceedings of the IEEE* 77.4, p. 541-580. ISSN : 0018-9219. DOI : 10.1109/5.24143.
- NIELSEN, S.F., J. SPARSØ, J.B. JENSEN et J.S.R. NIELSEN (2009). « A Behavioral Synthesis Frontend to the Haste/TiDE Design Flow ». In : *15th IEEE Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 185-194. DOI : 10.1109/ASYNC.2009.10.
- NOWICK, Steven M et David L DILL (1991). « Synthesis of asynchronous state machines using a local clock ». In : *IEEE International Conference on Computer Design : VLSI in Computers and Processors*, p. 192-197.
- OLIVEIRA, D. L., L. A. FARIA, H. A. DELSOTO et K. GARCIA (2016). « Synthesis of bundled-data asynchronous pipelines with reduced matched delays on FPGAs ». In : *23rd IEEE International Congress on Electronics, Electrical Engineering and Computing (INTERCON)*, p. 1-5. DOI : 10.1109/INTERCON.2016.7815571.
- PEETERS, A. et K. van BERKEL (1995). « Single-rail handshake circuits ». In : *2nd Working Conference on Asynchronous Design Methodologies*, p. 53-62. DOI : 10.1109/WCADM.1995.514642.
- PEETERS, A., F. t. BEEST, M. d. WIT et W. MALLON (2010). « Click Elements : An Implementation Style for Data-Driven Compilation ». In : *IEEE Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 3-14. DOI : 10.1109/ASYNC.2010.11.
- PETRI, Carl Adam (1966). « Communication with automata ». eng. Thèse de doct. Universität Hamburg.
- PROST-BOUCLE, Adrien. *AUGH : Autonomous and User Guided High-level synthesis*. Software, <http://tima.imag.fr/sls/research-projects/augh/>.
- (2014). « Génération rapide d'accélérateurs matériels par synthèse d'architecture sous contrainte de ressources ». Thèse de doct. Laboratoire TIMA, CNRS/Grenoble INP/UJF.
- PROST-BOUCLE, Adrien, Olivier MULLER et Frédéric ROUSSEAU (2014). « Fast and standalone design space exploration for high-level synthesis under resource constraints ». In : *Journal of Systems Architecture* 60.1, p. 79-93.
- QAISAR, S. M., M. AKBAR, T. BEYROUTHY, W. AL-HABIB et M. ASMATULAH (2016). « An error measurement for resampled level crossing signal ». In : *2nd Interna-*

- tional Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, p. 1-4. DOI : 10.1109/EBCCSP.2016.7605241.
- QAISAR, Saeed Mian, Laurent FESQUET et Marc RENAUDIN (2008). « An Adaptive Resolution Computationally Efficient Short-Time Fourier Transform ». In : *Journal of Electrical and Computer Engineering* 2008.
- (2014). « Adaptive rate filtering a computationally efficient signal processing approach ». In : *Signal Processing* 94, p. 620 -630. ISSN : 0165-1684. DOI : <http://dx.doi.org/10.1016/j.sigpro.2013.07.019>.
- RONCKEN, M. et al. (2015). « Naturalized Communication and Testing ». In : *21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 77-84. DOI : 10.1109/ASYNC.2015.20.
- SAITO, H., N. HAMADA, T. YONEDA et T. NANYA (2010). « A floorplan method for asynchronous circuits with bundled-data implementation on FPGAs ». In : *IEEE International Symposium on Circuits and Systems (ISCAS)*, p. 925-928. DOI : 10.1109/ISCAS.2010.5537402.
- SAYINER, Necip, Henrik V SORENSEN et Thayamkulangara R VISWANATHAN (1996). « A level-crossing sampling scheme for A/D conversion ». In : *IEEE Transactions on Circuits and Systems II : Analog and Digital Signal Processing* 43.4, p. 335-339.
- SCHELL, B. et Y. TSIVIDIS (2008). « A Continuous-Time ADC/DSP/DAC System With No Clock and With Activity-Dependent Power Dissipation ». In : *IEEE Journal of Solid-State Circuits* 43.11, p. 2472-2481. ISSN : 0018-9200. DOI : 10.1109/JSSC.2008.2005456.
- SEITZ, C. L. (1980). « System Timing ». In : sous la dir. de C.A. MEAD et L.A. CONWAY. Addison-Wesley. Chap. 7.
- SENAY, Seda, Luis F CHAPARRO, Mingui SUN et Robert J SCLABASSI (2010). « Adaptive level-crossing sampling and reconstruction ». In : *18th European Signal Processing Conference*. IEEE, p. 1296-1300.
- SHANG, Delong, Fei XIA et Alexandre YAKOVLEV (2002). « Asynchronous circuit synthesis via direct translation ». In : *IEEE International Symposium on Circuits and Systems (ISCAS)*. T. 3, p. III-369.
- SIMATIC, J., P. ALEXANDRE, A. CHERKAOUI, R. P. BASTOS et L. FESQUET. *A High-Level Synthesis Tool for Designing Ultra-Low Power Asynchronous Systems*. Submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
- SIMATIC, Jean et Brigitte BIDÉGARAY-FESQUET (2017). *pySPASS : Python Signal Processing for ASynchronous Systems*. Software, <https://forge.imag.fr/projects/pyspass/> [accessed 2017-06-28].
- SITIK, C., W. LIU, B. TASKIN et E. SALMAN (2016). « Design Methodology for Voltage-Scaled Clock Distribution Networks ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.10, p. 3080-3093. ISSN : 1063-8210. DOI : 10.1109/TVLSI.2016.2539926.
- SOKOLOV, D., A. BYSTROV et A. YAKOVLEV (2003). « STG optimisation in the direct mapping of asynchronous circuits ». In : *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, p. 932-937. DOI : 10.1109/DATE.2003.1253725.

- SOKOLOV, D., I. POLIAKOV et A. YAKOVLEV (2007). « Asynchronous Data Path Models ». In : *7th International Conference on Application of Concurrency to System Design (ACSD)*, p. 197-210. DOI : 10.1109/ACSD.2007.45.
- SOTIRIOU, C. R. (2001). « Direct-mapped asynchronous finite-state machines in CMOS technology ». In : *14th Annual IEEE International ASIC/SOC Conference*, p. 105-109. DOI : 10.1109/ASIC.2001.954681.
- SPARSØ, Jens et Steve FURBER (2002). *Principles of Asynchronous Circuit Design*. Springer.
- SUTHERLAND, I. et S. FAIRBANKS (2001). « GasP : a minimal FIFO control ». In : *7th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 46-53. DOI : 10.1109/ASYNC.2001.914068.
- SUTHERLAND, Ivan E. (1989). « Micropipelines ». In : *Communications of the ACM* 32.6, p. 720-738.
- TRANCHERO, M., L.M. REYNERI, A. BINK et M. de WIT (2009). « An Automatic Approach to Generate Haste Code from Simulink Specifications ». In : *15th IEEE Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 175-184. DOI : 10.1109/ASYNC.2009.19.
- VENKATARAMANI, Girish, Mihai BUDIU, Tiberiu CHELCEA et Seth Copen GOLDSTEIN (2004). « C to Asynchronous Dataflow Circuits : An End-to-End Toolflow ». In : *13th IEEE International Workshop on Logic Synthesis (IWLS)*. Temecula, CA.
- WAELE, S. de et P. M. T. BROERSEN (2000). « Error measures for resampled irregular data ». In : *IEEE Transactions on Instrumentation and Measurement* 49.2, p. 216-222. ISSN : 0018-9456. DOI : 10.1109/19.843052.
- WAUGAMAN, Maxwell et William KOVEN (2017). « Sharp - A Resilient Asynchronous Template ». In : *23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. (to be published).
- WONG, C.G. et A.J. MARTIN (2003). « High-level synthesis of asynchronous systems by data-driven decomposition ». In : *Design Automation Conference (DAC)*, p. 508-513.
- YAHYA, Eslam (2009). « Performace Modeling, analysis and optimization of multi-protocol asynchronous circuits ». Thèse de doct. Institut National Polytechnique de Grenoble - INPG.
- YAKOVLEV, A.V., A.M. KOELMANS, A. SEMENOV et D.J. KINNIMENT (1996). « Modelling, analysis and synthesis of asynchronous control circuits using Petri nets ». In : *Integration, the VLSI Journal* 21.3, p. 143 -170. ISSN : 0167-9260. DOI : [http://dx.doi.org/10.1016/S0167-9260\(96\)00010-7](http://dx.doi.org/10.1016/S0167-9260(96)00010-7).
- YUN, K.Y., P.A. BEEREL et Julio ARCEO (1996). « High-performance asynchronous pipeline circuits ». In : *2nd International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, p. 17-28. DOI : 10.1109/ASYNC.1996.494434.
- YUN, K.Y. et D.L. DILL (1999). « Automatic synthesis of extended burst-mode circuits. I. (Specification and hazard-free implementations) ». In : *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 18.2, p. 101-117. ISSN : 0278-0070. DOI : 10.1109/43.743711.

Publications de l'auteur

Publications dans des conférences internationales avec comité de lecture

- EL-HADBI, A., A. CHERKAOUI, O. ELISSATI, J. SIMATIC et L. FESQUET (2017). « On-the-fly and sub-gate-delay resolution TDC based on self-timed ring : A proof of concept ». In : *15th IEEE International New Circuits and Systems Conference (NEW-CAS)*, p. 305-308. DOI : 10.1109/NEWCAS.2017.8010166.
- QAISAR, S. M., J. SIMATIC et L. FESQUET (2017). « High-level Synthesis of an Event-Driven Windowing Process ». In : *3rd International Conference on Event-based Control, Communication and Signal Processing (EBCCSP)*, p. 1-8. DOI : 10.1109/EBCCSP.2017.8022807.
- SIMATIC, J., R. P. BASTOS et L. FESQUET (2016). « High-level synthesis for event-based systems ». In : *2nd International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, p. 1-7. DOI : 10.1109/EBCCSP.2016.7605252.
- SIMATIC, J., L. FESQUET et B. BIDEGARAY-FESQUET (2015). « Correctly sizing FIR filter architecture in the framework of non-uniform sampling ». In : *International Conference on Sampling Theory and Applications (SampTA)*, p. 269-273. DOI : 10.1109/SAMP.2015.7148894.
- SIMATIC, J., A. CHERKAOUI, R. P. BASTOS et L. FESQUET (2016). « New asynchronous protocols for enhancing area and throughput in bundled-data pipelines ». In : *29th Symposium on Integrated Circuits and Systems Design (SBCCI)*, p. 1-6. DOI : 10.1109/SBCCI.2016.7724066.
- SIMATIC, J., A. CHERKAOUI, F. BERTRAND, R. P. BASTOS et L. FESQUET (2017). « A Practical Framework for Specification, Verification, and Design of Self-Timed Pipelines ». In : *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, p. 65-72. DOI : 10.1109/ASYNC.2017.16.
- SKAF, A., J. SIMATIC et L. FESQUET (2017). « Seeking low-power synchronous/asynchronous systems : A FIR implementation case study ». In : *IEEE International Symposium on Circuits and Systems (ISCAS)*, p. 1-4. DOI : 10.1109/ISCAS.2017.8050379.

Publications dans des conférences nationales

- SIMATIC, J., R. POSSAMAI BASTOS et L. FESQUET (2015). « Flot de conception pour l'ultra-faible consommation : échantillonnage non-uniforme et électronique asynchrone ». In : *Journées Nationales du Réseau Doctoral en Micro-nanoélectronique (JNRDM)*.

Flot de conception pour l'ultra-faible consommation : échantillonnage non uniforme et électronique asynchrone

Résumé – Les systèmes intégrés sont souvent des systèmes hétérogènes avec des contraintes fortes de consommation électrique. Ils embarquent aujourd'hui des actionneurs, des capteurs et des unités pour le traitement du signal. Afin de limiter l'énergie consommée, ils peuvent tirer profit des techniques événementielles que sont l'échantillonnage non uniforme et l'électronique asynchrone. En effet, elles permettent de réduire drastiquement la quantité de données échantillonnées pour de nombreuses classes de signaux et de diminuer l'activité. Pour aider les concepteurs à développer rapidement des plateformes exploitant ces deux techniques événementielles, nous avons élaboré un flot de conception nommé ALPS. Il propose un environnement permettant de déterminer et de simuler au niveau algorithmique le schéma d'échantillonnage et les traitements associés afin de sélectionner les plus efficaces en fonction de l'application ciblée. ALPS génère directement le convertisseur analogique/numérique à partir des paramètres d'échantillonnage choisis. L'élaboration de la partie de traitement s'appuie quant à elle sur un outil de synthèse de haut niveau synchrone et une méthode de désynchronisation exploitant des protocoles asynchrones spécifiques, capables d'optimiser la surface et la consommation du circuit. Enfin, des simulations au niveau portes logiques permettent d'analyser et de valider l'énergie consommée avant de poursuivre par un flot classique de placement et routage. Les évaluations conduites montrent une réduction d'un facteur 3 à 8 de la consommation des circuits automatiquement générés. Le flot ALPS permet à un concepteur non spécialiste de se concentrer sur l'optimisation de l'échantillonnage et de l'algorithme en fonction de l'application et de potentiellement réduire d'un ou plusieurs ordres de grandeur la consommation du circuit.

Mots clés – Faible consommation, techniques événementielles, échantillonnage non uniforme, circuits asynchrones, synthèse de haut niveau, machine à états finis, contrôleurs asynchrones, traitement numérique du signal.

Design flow for ultra-low power : nonuniform sampling and asynchronous circuits

Abstract – Integrated systems are mainly heterogeneous systems with strong power consumption constraints. They embed actuators, sensors and signal processing units. To limit the energy consumption, they can exploit event-based techniques, namely non uniform sampling and asynchronous circuits. Indeed, they allow cutting drastically the amount of sampled data for many types of signals and reducing the system activity. To help designers in quickly developing platforms that exploit those event-based techniques, we elaborated a design framework called ALPS. It proposes an environment to determine and simulate at algorithmic level the sampling scheme and the associated processing in order to select the most efficient ones depending on the targeted application. ALPS generates directly the analog-to-digital converter based on the chosen sampling parameters. The elaboration of the processing unit uses a synchronous high-level synthesis tool and a desynchronization method that exploits specific asynchronous protocols to optimize the circuit area and power consumption. Finally, gate-level simulations allow analyzing and validating the energy consumption before continuing with a standard placement and routing flow. The conducted evaluations show a reduction factor of 3 to 8 of the consumption of the automatically generated circuits. The flow ALPS allow non-specialists to concentrate on the optimization of the sampling and the processing in function of their application and to reduce the circuit power consumptions by one to several orders of magnitude.

Keywords – Low power, event-driven techniques, non uniform sampling, asynchronous circuits, high-level synthesis, finite-state machine, asynchronous controllers, digital signal processing.
