Comparaison des stratégies de synchronisation pour circuits intégrés

par

Laurent TREMBLAY

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 19 SEPTEMBRE 2024

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Claude Thibeault, directeur de mémoire
Département de génie électrique à l'école de technologie supérieure

M. René Jr. Landry, président du jury
Département de génie électrique à l'école de technologie supérieure

M. François Blanchard, membre du jury
Département de génie électrique à l'école de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 10 SEPTEMBRE 2024

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Dans un premier temps, je tiens à remercier mon directeur de mémoire, Claude Thibeault, pour son implication accrue dans le projet par sa présence aux rencontres hebdomadaires, pour son mentorat face à mes multiples questionnements et pour sa généreuse aide financière, sans lequel le présent projet n'aurait pas eu lieu.

Je tiens aussi à remercier mon collègue, Michel Kafrouni, pour ses connaissances associées à la réalité du marché du travail et pour son aide lors du développement de la stratégie pseudo-asynchrone.

J'aimerais aussi remercier mes amis du laboratoire d'hydrologie avec qui j'ai pu vivre de belles expériences et qui ont su me guider à travers les spécificités de la maîtrise, ainsi que mes amies du club des ingénieuses de l'ÉTS qui m'ont permis de rester motivé par mon implication dans la vie étudiante.

J'aimerais remercier mon beau-père Mario qui m'a grandement aidé dans la correction de ce travail afin de le rendre plus compréhensible.

Finalement, je tiens à remercier ma famille et mes amis proches de m'avoir soutenu autant dans les moments forts et les moments difficiles du projet.

Comparaison des stratégies de synchronisation pour circuits intégrés

Laurent TREMBLAY

RÉSUMÉ

Le présent mémoire porte sur la comparaison de quatre stratégies de synchronisations retrouvées dans les circuits intégrés pour évaluer leurs performances suite à une implémentation avec les outils de conception assistée par ordinateur utilisés en industrie. On y retrouve trois stratégies reconnues dans la communauté scientifique, la stratégie synchrone, la stratégie asynchrone *bundled-data* et la stratégie du *wave pipelining*, ainsi qu'une nouvelle stratégie proposée par Michel Kafrouni, la stratégie pseudo-asynchrone avec emprunt d'horloge.

La loi de Moore, qui stipule que le nombre de transistors dans un circuit intégré double à chaque deux ans, a entraîné l'effervescence de l'utilisation de la stratégie de synchronisation synchrone dans l'industrie du semi-conducteur. Cependant, au cours de dernières années, on a pu noter un important ralentissement au niveau du développement des procédés de fabrication, limitant l'augmentation des performances d'un système utilisant cette stratégie.

À ce jour, on retrouve plusieurs stratégies de synchronisation proposant des améliorations face à la stratégie synchrone. En premier, on retrouve la stratégie asynchrone *bundled-data*, qui propose l'utilisation de contrôleurs afin d'autoréguler le cadencement du circuit. Ensuite, on retrouve la stratégie du *wave pipelining*, qui tente de pousser les limites des performances en modélisant les délais combinatoires comme une ligne de transmission. Finalement, on retrouve la stratégie pseudo-asynchrone avec emprunt d'horloge, qui propose un hybride entre la stratégie synchrone et la stratégie asynchrone pour exploiter leurs avantages.

Étant donné l'incompatibilité de conception des stratégies non synchrones avec les outils de conception assistée par ordinateur, des méthodes non conventionnelles ont dû être utilisées pour contraindre l'analyse statique des délais. Dans un premier temps, la méthode du *local clock set* a été utilisée pour modéliser la stratégie asynchrone. Par la suite, une nouvelle méthode a été conçue pour modéliser le *Wave Pipelining* en fonction des équations retrouvées dans la littérature scientifique. Enfin, une nouvelle méthode a également été conçue en collaboration avec Michel Kafrouni pour implémenter la stratégie pseudo-asynchrone avec emprunt d'horloge.

Les résultats obtenus dans la présente recherche ont permis de mettre en lumière le potentiel que propose la stratégie pseudo-asynchrone avec emprunt d'horloge. À performance égale aux autres stratégies, elle à démontré une réduction de la surface utilisée ainsi qu'une réduction de la puissance consommée pour les circuits de grande taille. Des améliorations à la méthode développée pour cette stratégie sont recommandées afin d'optimiser la consommation de la puissance associée au réseau de l'horloge sans intervention humaine.

Mots-clés: circuits intégrés, synchronisation

Comparison of Synchronization Strategies in Integrated Circuits

Laurent TREMBLAY

ABSTRACT

This thesis focuses on comparing four synchronization strategies found in integrated circuits to evaluate their performance after implementation with computer-aided design tools used in the industry. It includes three strategies recognized in the scientific community : the synchronous strategy, the asynchronous bundled-data strategy, and the wave pipelining strategy, as well as a new strategy proposed by Michel Kafrouni, the pseudo-asynchronous strategy with clock borrowing.

Moore's Law, which states that the number of transistors in an integrated circuit doubles every two years, has led to the widespread use of the synchronous synchronization strategy in the semiconductor industry. However, in recent years, there has been a significant slowdown in the development of manufacturing processes, limiting the performance improvements of systems using this strategy.

To date, several synchronization strategies offer improvements over the synchronous strategy. First, there is the asynchronous bundled-data strategy, which uses controllers to self-regulate the circuit's timing. Next is the wave pipelining strategy, which pushes performance limits by modelling combinational delays as a transmission line. Finally, there is the pseudo-asynchronous strategy with clock borrowing, which proposes a hybrid between the synchronous and asynchronous strategies to exploit their advantages.

Due to the design incompatibility of non-synchronous strategies with computer-aided design tools, unconventional methods had to be used to constrain static timing analysis. First, the local clock set method was used to model the asynchronous strategy. Next, a new method was designed to model wave pipelining based on equations found in the scientific literature. Finally, a new method was also designed in collaboration with Michel Kafrouni to implement the pseudo-asynchronous strategy with clock borrowing.

The results obtained in this research highlighted the potential offered by the pseudo-asynchronous strategy with clock borrowing. At equal performance to the other strategies, it demonstrated a reduction in the area used as well as a reduction in power consumption for large circuits. Improvements to the method developed for this strategy are recommended to optimize power consumption associated with the clock network without human intervention.

Keywords: Integrated Circuits, Synchronization

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACK	*Acknowledge* (Acquiescement)
ADM	*Adaptive Delay Matching*
IA	Intelligence artificielle
ASIC	*Application Specific Integrated Circuit* (Circuits intégrés à applications spécifiques)
BD	*Bundled-Data*
CE	*Click Element*
CNN	*Convolutionnal Neural Network* (Réseau de neurones convolutif)
CPU	*Central Processing Unit* (Unité de traitement centrale)
CTO	*Clock Tree Optimisation* (Optimisation de l'arbre de l'horloge)
DC	*Design Compiler*
EDA	*Electronic Design and Automation* (Conception électronique automatisé)
FPGA	*Field Programmable Gate Array* (Réseau de portes programmables par l'usager)
GLS	*Gate Level Simulation* (Simulation du réseau de portes logiques)
GPU	*Graphics Processing Unit* (Unité de traitement graphique)
HDL	*Hardware Description Language* (Langage de description matérielle)
ICC	*Integrated Circuit Compiler*
ICC2	*Integrated Circuit Compiler 2*
IRDS	*International Roadmap of Design and Systems*
LCS	*Local Clock Set* (Lots d'horloges locales)
MAC	*Multiply-Accumulate* (Multiplicateur accumulateur)
MEF	Machine à états finis
MMMC	*Multi-Mode Multi-Corners*
NMOS	*N-channel Metal-Oxide-Semiconductor*

NRZ	*Non Return to Zero* (Non retour à zéro)
PMOS	*P-channel Metal-Oxide-Semiconductor*
PNR	*Place and Route* (Placement et routage)
PP	*PrimePower*
PT	*PrimeTime*
PVT	*Process, Voltage and Temperature* (Procédé, tension et température)
QOR	*Quality Of Results* (Qualité des résultats)
REQ	*Request* (Requête)
RTC	*Relative Timing Constraints* (Contrainte de délais relatives)
RTL	*Register Transfer Level* (Niveau transfert de registres)
RZ	*Return to Zero* (Retour à zéro)
SDC	*Synopsys Design Constraints* (Contraintes de design Synopsys)
SDF	*Standard Delay Format*
STA	*Static Timing Analysis* (Analyse statique des délais)
STG	*Signal Transition Graph* (Graphe de transition de signaux)
SV	*SystemVerilog*
TLU	*Table Lookup Unit*
VCD	*Value Change Dump*
WNS	*Worst Negative Slack* (Pire marge négative)
WP	*Wave pipelining* (Pipelinage en vague)
XML	*Extensible Markup Language*

LISTE DES SYMBOLES ET UNITÉS DE MESURE

s	Seconde
μm^2	Micromètre carré
mW	Milliwatt
MHz	Mégahertz
Go/s	Gigaoctets par seconde
Δ	Délai asymétrique
Δ_I	Délai asymétrique d'une bascule en entrée
Δ_O	Délai asymétrique d'une bascule en sortie
Δ_n	Délai asymétrique d'une bascule de l'étage de pipeline n
Δ_{CLK}	Délai asymétrique de l'horloge
T_H	Délai de maintien
T_{SU}	Délai de préparation
T_{CQ}	Délai de propagation d'une bascule
D_{MAX}	Délai de propagation maximal total
D_{MIN}	Délai de propagation minimal total
$T_{COMB(MAX)}$	Délai de propagation maximal à travers un bloc combinatoire
$T_{COMB(MIN)}$	Délai de propagation minimal à travers un bloc combinatoire
pod	Point de divergence
poc	Point de convergence
m	Marge
δ	Marge
$d\text{-}data$	Port de donnée d'une bascule du chemin de donnée
$d\text{-}ctrl$	Port de contrôle d'une bascule du chemin de donnée
$c\text{-}data$	Port de donnée d'une bascule du chemin de contrôle
$c\text{-}ctrl$	Port de contrôle d'une bascule du chemin de contrôle

INTRODUCTION

Depuis quelques années, l'intelligence artificielle (IA) ne cesse de faire ses preuves comme outil essentiel pour répondre aux défis d'envergure du prochain siècle. La possibilité de transmettre la capacité humaine de reconnaître les tendances, de traiter des données et de prendre des décisions appropriées aux ordinateurs permet d'augmenter exponentiellement l'efficacité de traitement de problèmes anciennement considérés comme incommensurables. Parmi ces problèmes, on retrouve des exemples tels que la détection de cancer, l'analyse des changements climatiques et les transports autonomes (A. *et al.*, 2019).

Malgré tous les aspects positifs que propose l'IA, des défis majeurs sont toujours présents au niveau de l'implantation de cette technologie dans le quotidien. Le plus important d'entre eux s'avère être le besoin de haute performance à faible latence. À l'heure actuelle, l'ensemble des données recueillies doivent être traitées dans des centres de données (Lee, Tsung & Wu, 2018), là où les ressources computationnelles pour l'entraînement des modèles sont presque illimitées. Le problème de ce type de traitement est que bien qu'il soit efficient, le délai entre l'envoi des données et la réception des résultats est limité par la bande passante de l'internet.

Ce problème s'aggrave davantage avec l'introduction des systèmes en temps réel, tels les véhicules autonomes, où la quantité de données à traiter peut atteindre deux gigaoctets par seconde (Go/s) (Liu *et al.*, 2019) et où l'accès à l'internet peut être coupé pendant un délai non négligeable. La solution proposée dans la littérature, l'*edge AI* (Singh & Gill, 2023), réside dans la création de ressources computationnelles dédiées à l'implémentation des réseaux de neurones convolutifs (CNN) le plus près de l'endroit où les données sont générées.

La structure d'un CNN est principalement composée de multiplicateurs et d'additionneurs. Ils sont typiquement implémentés de façon logicielle dans des processeurs multi-usages (CPU) ou des processeurs graphiques (GPU) à cause de la flexibilité qu'offre le développement logiciel

(Lin *et al.*, 2018). Cette flexibilité implique cependant un coût élevé lorsqu'il est question de la puissance consommée par ces processeurs à cause de leur structure générique.

Étant donné la tendance de vouloir des appareils de type *edge* alimentés par des batteries (Singh & Gill, 2023), ils auraient plutôt avantage à être implémentés à l'aide de circuits intégrés pour applications spécifiques (ASIC) ou de circuits intégrés programmables par l'usage (FPGA) comme accélérateurs matériels, car ces deux plateformes permettent d'obtenir la meilleure latence sans dépasser excessivement le budget de puissance (Capra *et al.*, 2020).

Le principal défi de l'implémentation matérielle reste au niveau de l'optimisation de l'architecture, ce pour quoi plusieurs travaux sont continuellement effectués pour l'amélioration de la structure du réseau de neurones convolutif (Wang, Lin & Wang, 2018) ou de la structure interne des circuits de multiplication et d'addition (Quinnell, Swartzlander & Lemonds, 2007).

Bien que la majorité des efforts de recherches soient concentrés sur l'optimisation de l'architecture du chemin de données, les efforts quant à l'architecture du chemin de contrôle, agissant comme biais de synchronisation des éléments logiques, se voient souvent être négligés. Cela est effectivement dû au fait que la méthode de synchronisation dominant l'espace numérique est, encore à ce jour, l'utilisation d'une horloge globale, aussi connue sous le de la stratégie synchrone.

La stratégie qui consiste à discrétiser le temps offre une solution élégante au concepteur, ce qui facilite la conception et la vérification, mais se trouve à limiter la performance des systèmes à cause de l'augmentation de la variabilité dans les circuits intégrés dû à la réduction de taille des transistors. Ce fait est observé dans le rapport de l'*International Roadmap for Device and Systems* (IRDS), ce dernier mentionnant qu'un éventuel changement de paradigme devra être effectué pour découpler la performance des circuits intégrés du pire cas d'analyse, soit le cas utilisé pour caractériser les performances de la stratégie synchrone (Device & Systems, 2022).

L'utilisation d'une horloge globale comme moyen de synchronisation nécessite la construction d'une structure en arbre complexe afin de distribuer le signal à toutes les bascules sans induire de délais asymétriques. Plusieurs travaux sont effectués dans l'objectif de développer des méthodes de synthèse pour réduire la puissance et diminuer les délais de ce réseau (Kuzmin, 2023), mais il reste que cette structure nécessite un temps de développement substantiel pour éviter des problèmes liés à la consommation de puissance excessive, à la variabilité causée par les effets de procédé, de tension et de température (PVT), ainsi qu'à l'augmentation exponentielle de la complexité des designs (Roy, Mattheakis, Masse-Navette & Pan, 2014).

À l'opposé, on retrouve dans la littérature scientifique la stratégie asynchrone, qui est décrite comme étant une alternative permettant de résoudre une grande partie des problèmes liés à la stratégie synchrone. Parmi ses bénéfices, on retrouve une grande robustesse aux effets des variations PVT par un cadencement autorégulé, une faible consommation de puissance par un traitement en fonction des évènements et une complexité réduite de conception par l'utilisation d'une synchronisation locale (Nowick & Singh, 2015).

Un hybride entre les deux stratégies, ce que l'on nomme dans ce présent écrit la stratégie pseudo-asynchrone avec emprunt d'horloge (PA), est évalué afin d'observer ses performances face aux stratégies synchrone et asynchrone. Cette stratégie consiste à utiliser un signal d'impulsion pour synchroniser séquentiellement des éléments sur un canal de communication local. Elle possède des caractéristiques semblables au *Wave Pipelining*, soit une stratégie qui permet d'augmenter le débit de données d'un circuit au coût de contraintes plus strictes (Cotten, 1969).

L'objectif de la présente recherche est d'implémenter les stratégies de synchronisation sur des circuits communément utilisés dans les CNN pour justifier leur utilisation en fonction de leur performance.

Pour ce faire, il sera question d'utiliser et de développer des méthodes permettant l'analyse statique des délais (STA), la synthèse logique ainsi que le placement-routage (PNR) des différentes stratégies de synchronisation avec les outils d'automatisation de conception électronique (EDA) commerciaux. Un flux de travail devra aussi être développé afin de dicter les étapes de l'implémentation, de la vérification et la génération de résultats pour les différentes stratégies.

Le circuit choisi pour la comparaison des stratégies sera celui du multiplicateur-accumulateur (MAC) à nombre entier à cause de sa simplicité d'implémentation et de son importante implication dans les CNN. En effet, le circuit combine les opérations mathématiques nécessaires afin d'implémenter le produit scalaire, soit l'opération permettant de calculer le résultat à la sortie d'un neurone par la somme du produit des connexions neuronales et des coefficients synaptiques.

Cette caractéristique relie donc la performance du MAC à la performance du CNN, d'où l'importance de tenter d'améliorer le circuit pour faciliter le développement de puces plus puissantes dédiées à l'IA. Bien que les MAC utilisés dans les CNN sont typiquement conçus pour traiter des nombres à point flottant, le choix d'utiliser des nombres entiers permet, dans le cadre de ce projet de recherche, de mettre l'accent sur le développement des contraintes qui visent l'implémentation des différentes stratégies avec les outils EDA synchrones.

Ce présent mémoire est séparé en trois chapitres. Le premier chapitre consiste en une mise à niveau quant aux concepts de base nécessaires qui permettent la compréhension du travail de recherche effectué. Il est accompagné d'une revue des méthodes et des travaux trouvés dans la littérature scientifique sur les différentes stratégies de synchronisations.

Le deuxième chapitre explique en détail la méthodologie utilisée pour générer les résultats, les contraintes *Synopsys Design Constraints* (SDC) utilisées et développées pour l'implémentation des différentes stratégies ainsi qu'une exposition à la structure des circuits évalués.

Enfin, le dernier chapitre présente une analyse détaillée des résultats obtenus suite à l'implémentation de chaque stratégie de synchronisation, ainsi qu'une discussion récapitulative des tendances observées. Le tout est conclu par des recommandations, les limitations et des pistes de recherche pour les travaux futurs.

Les principales contributions de ce travail de recherche se trouvent dans la mise à jour, l'optimisation et l'extension de la méthode du *Local Clock Set* (LCS), dans la modélisation de la stratégie de synchronisation pseudo-asynchrone (PA) avec emprunt d'horloge avec des contraintes SDC ainsi que dans la modélisation de la stratégie du WP avec des contraintes SDC, permettant l'implémentation des stratégies avec les outils EDA synchrones.

CHAPITRE 1

CONCEPTS DE BASE ET REVUE DE LITTÉRATURE

Dans ce chapitre, les concepts de base liés à chaque stratégie de synchronisation sont exposés sous quatre sections en fonction des informations recueillies dans la littérature scientifique.

La première section traite du fondement de la synchronisation avec la stratégie synchrone, soit la stratégie principalement utilisée dans la conception numérique. On y retrouve des informations sur son fonctionnement accompagné de quelques méthodes couramment utilisées afin d'améliorer ses performances.

La deuxième section traite de la stratégie opposée, soit la synchronisation asynchrone. L'accent est mis sur les circuits utilisant l'encodage *Bundled-Data* (BD) à cause de sa compatibilité avec les outils EDA. On y retrouve une comparaison de différents contrôleurs asynchrones ainsi qu'une comparaison des différentes méthodes utilisées pour permettre leur implémentation avec les outils EDA synchrones.

La troisième section traite de la stratégie du *Wave Pipelining*. Elle permet de maximiser les performances atteignables d'une technologie par la minimisation de la différence entre le délai maximal et le délai minimal de la logique combinatoire. On y retrouve aussi des explications sur les complexités rencontrées lors de son implémentation.

Finalement, la quatrième et dernière section traite de la stratégie pseudo-asynchrone avec emprunt d'horloge, une nouvelle méthode de synchronisation proposée par Michel Kafrouni basée sur l'amalgamation des stratégies synchrones et asynchrones.

1.1 Stratégie synchrone

La nécessité de l'utilisation d'une synchronisation découle de l'invention de la logique séquentielle, basée sur la discrétisation du traitement en plusieurs étapes subséquentes. En temps normal, cette discrétisation se fait par la division du travail en états distincts avec l'aide d'une machine à états finis (MEF) ou par la division du travail en plusieurs étages subséquents par

l'implémentation d'un pipeline. Cette discrétisation se traduit par l'utilisation d'éléments de stockage pour sauvegarder les résultats des étapes intermédiaires, ce qui nécessite une stratégie pour déterminer la séquence d'enregistrement.

Grâce à ses avantages au niveau de la simplification de l'implémentation et de la vérification, le signal utilisé pour contrôler la séquence de traitement a été défini comme étant une horloge globale, donnant naissance à la stratégie de synchronisation synchrone (Wirth, 1995).

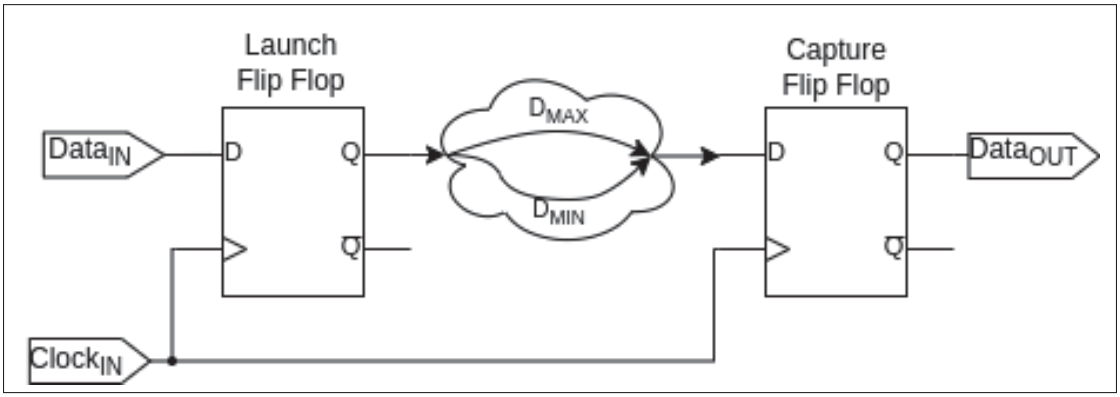


Figure 1.1 Modèle d'analyse d'un circuit synchrone

De par la nature des transitions du signal d'horloge, le choix de l'élément de stockage s'est naturellement porté en faveur de la bascule sensible au front montant. Pour fonctionner correctement, cette dernière nécessite que la donnée soit stable pendant une certaine ouverture de synchronisation définie par une contrainte de préparation et une contrainte de maintien. Cette ouverture est exposée dans le chronogramme de la figure 1.2, ce dernier étant basé sur le modèle d'analyse de la synchronisation synchrone démontré par la figure 1.1.

La première contrainte, présentée par l'équation 1.1, est celle du temps de préparation. Elle établit qu'une donnée lancée doit se propager à travers la bascule (T_{CQ}) et se propager à travers le plus long chemin combinatoire ($T_{COMB(MAX)}$)−ce que l'on définit comme le délai maximal ($D_{MAX} = T_{CQ} + T_{COMB(MAX)}$)−et qu'elle doit arriver avant une certaine marge de préparation

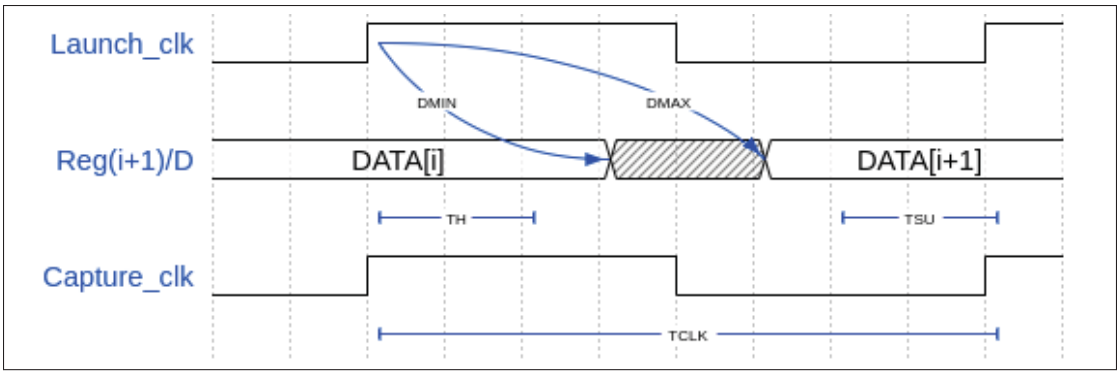


Figure 1.2 Chronogramme typique pour un circuit synchrone

(T_{SU}) pour être correctement capturée par le prochain front montant de l'horloge (T_{CLK}).

$$T_{CLK} \geq D_{MAX} + T_{SU} \tag{1.1}$$

La deuxième contrainte, présentée par l'équation 1.2, est celle du temps de maintien. Elle établit que le temps de propagation à travers la bascule (T_{CQ}) et du plus petit chemin combinatoire entre deux bascules ($T_{COMB(MIN)}$)−que l'on définit comme le délai minimal ($D_{MIN} = T_{CQ} + T_{COMB(MIN)}$)−doit être plus long que le temps durant lequel la donnée doit être maintenue stable (T_H) lors de la réception du front montant de l'horloge.

$$D_{MIN} \geq T_H \tag{1.2}$$

La simplicité du paradigme synchrone vient du fait que, étant donné la nature globale de l'horloge, ces deux équations à elles seules sont nécessaires pour contraindre les échanges de données entre les éléments séquentiels. En effet, cela ce contraste aux autres paradigmes qui nécessitent des contraintes en fonction des différents chemins du circuit.

Cette nature globale permet entre autres de caractériser les performances d'un système synchrone en fonction des caractéristiques du signal d'horloge. La première variable de performance est la fréquence de l'horloge, cette dernière indiquant le débit auquel les données sont traitées. Pour

augmenter la performance, le délai combinatoire doit être réduit par l'insertion de bascules intermédiaires, coupant ainsi la logique en plusieurs étages de pipeline.

La deuxième variable de performance est la latence L. Il s'agit de la différence entre le temps de lancement de la donnée dans le pipeline et le temps d'observation du résultat en sortie. Dans un système synchrone, on l'approxime généralement comme étant le nombre de cycles d'horloge nécessaire avant de pouvoir échantillonner la valeur. Cette relation est présentée par l'équation 1.3, soit le nombre de registres (N_{REG}) multiplié par la période d'horloge (T_{CLK}).

$$L \approx N_{REG} \cdot T_{CLK} \tag{1.3}$$

1.1.1 Délai asymétrique utile

Une des hypothèses faites par les équations 1.1 et 1.2 du modèle théorique de la synchronisation synchrone est qu'on suppose que le signal de l'horloge atteint toutes les bascules du circuit en même temps. Cette contrainte implique un délai asymétrique global nul, ce qui nécessite la construction d'un réseau de distribution de l'horloge entièrement symétrique pour assurer une distribution égale du signal.

Cetta caractéristique s'avère être un défi grandissant étant donné que les structures logiques ne suivent pas nécessairement un motif fractal. En effet, cela implique que des délais seront nécessairement induits lors du PNR par l'entremise des effets parasites. Cela se traduit donc en délais asymétriques (Δ), ce qui est présenté dans l'équation 1.4, qu'on définit comme étant la différence entre le délai de propagation de l'horloge de capture (Δ_O) et celui de l'horloge de lancement (Δ_I).

$$\Delta = \Delta_I - \Delta_O \tag{1.4}$$

L'emplacement de ces délais est schématisé par la figure 1.3. On retrouve deux niveaux du délai asymétrique : les délais locaux et les délais globaux. Le premier fait référence au délai entre deux bascules qui ont un lien immédiat, par exemple les bascules constituant un registre. En

contrepartie, le deuxième fait référence au délai entre deux bascules qui se trouvent dans le circuit, peu importe leur lien, étant donné qu'ils sont contrôlés par le même signal d'horloge.

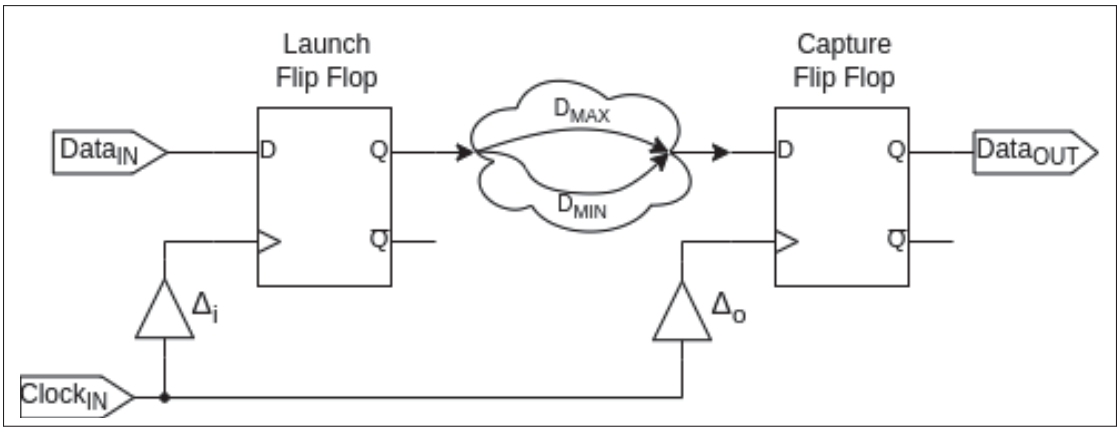


Figure 1.3 Modèle d'analyse d'un circuit synchrone avec délai asymétrique

La figure 1.4 présente le chronogramme qui résulte de l'ajout de délais asymétriques sur le réseau de l'horloge, ce qui permet de modifier les équations 1.1 et 1.2, qui contraignent l'analyse pour la préparation et le maintien de la donnée, afin de prendre en compte le nouveau délai de synchronisation.

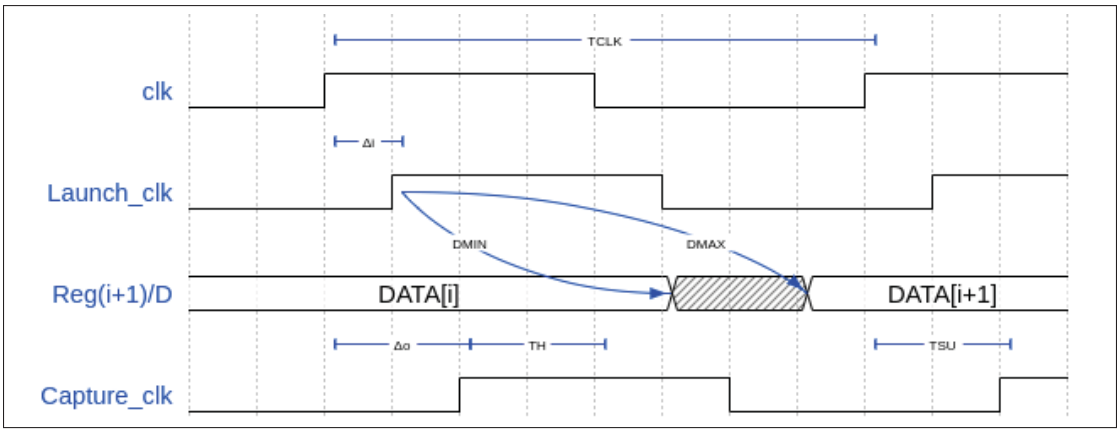


Figure 1.4 Chronogramme du délai asymétrique

Tout d'abord, l'équation 1.5 est dérivée de l'équation 1.1 pour la contrainte du temps de préparation. Elle présente les deux représentations possibles de l'équation.

$$T_{CLK} + \Delta_O \geq D_{MAX} + T_{SU} + \Delta_I \iff T_{CLK} + \Delta \geq D_{MAX} + T_{SU} \tag{1.5}$$

Dans le même ordre d'idées, le délai asymétrique est ajouté à l'équation 1.2 sur la contrainte de maintien pour devenir l'équation 1.6.

$$D_{MIN} + \Delta_I \geq T_H + \Delta_O \iff D_{MIN} \geq T_H + \Delta \tag{1.6}$$

Comme on peut l'observer, l'ajout du terme Δ permet, lorsque positif, de diminuer la période de l'horloge T_{CLK} sans avoir à modifier le délai du chemin combinatoire le plus long D_{MAX}, ce qui permet d'atteindre de plus grandes performances. Cette optimisation vient cependant au coût d'une potentielle augmentation de la surface du circuit dans la mesure où le délai du chemin le plus court D_{MIN} n'est pas assez long pour répondre aux besoins de la nouvelle contrainte de maintien.

Un autre avantage de cette méthode est que l'ajout de délais de synchronisation réduit le pic de courant qui résulte de l'arrivée d'un front montant sur l'ensemble des bascules du réseau, permettant d'éviter des défectuosités dans certaines puces où la dégradation de l'alimentation serait trop importante (Dai & Staepelaere, 2002). Cependant, ces délais insérés dans l'arbre d'horloge, majoritairement par l'utilisation de tampons, augmentent l'activité parasitique du réseau, ce qui provoque une plus grande consommation de puissance moyenne (Kim, Do & Kang, 2017).

Malgré cela, les outils de PNR utilisent le délai asymétrique utile pour optimiser les chemins critiques, ce qui permet de diminuer l'effort nécessaire lors de l'implémentation de l'arbre d'horloge tout en atteignant les objectifs de performance requis par les contraintes du concepteur.

1.1.2 Recadencement

Une des difficultés de la conception de circuits numériques synchrones se trouve au niveau du balancement du délai combinatoire entre les étages d'un pipeline. En effet, comme démontré dans l'équation 1.1 sur la contrainte de préparation, la période de l'horloge est limitée par le plus long délai combinatoire parmi tous les étages du pipeline. Pour augmenter le débit du circuit, le concepteur doit manuellement segmenter la logique combinatoire en ajoutant des bascules intermédiaires à des endroits stratégiques dans le but de balancer ce délai combinatoire entre les différents étages.

La complexité de cette opération est une conséquence de la nature de haut niveau du style de programmation du niveau du transfert de registres (RTL), décrit avec les langages de description matériels (HDL). Ce style de programmation, qui laisse la majorité du travail à l'outil de synthèse, est désavantagé, par rapport au style de programmation structurel, par son absence de granularité au niveau des structures internes. Il compense cependant par le fait qu'il ne nécessite pas l'implémentation de l'entièreté du circuit avec des équations booléennes, ce qui augmente énormément la productivité, d'où son utilisation dans le cadre de la présente recherche.

Pour résoudre cette limitation du RTL, les outils de synthèse sont munis d'un algorithme d'optimisation optionnel nommé le recadencement (*retiming*), schématisé par la figure 1.5. Le recadencement permet à l'outil de déplacer les registres en amont ou en aval de la logique combinatoire afin d'égaliser les délais entre chaque étage du pipeline, ce qui permet ainsi de minimiser la période de l'horloge. Cette minimisation entraîne cependant une augmentation du temps de synthèse, une potentielle augmentation de la surface si la structure du circuit nécessite un dédoublement des éléments séquentiels, une perte de l'encodage des états d'une MEF et une modification dissemblable de la structure de réinitialisation.

Le recadencement est un champ de recherche en constante évolution depuis sa formulation théorique par Leiserson & Saxe (1981) à cause de la nature polynomiale de l'algorithme proposé. La grande majorité des recherches proposent différentes heuristiques pour améliorer l'algorithme de synthèse (Lin & Zhou, 2006), pour augmenter la compatibilité avec des structures

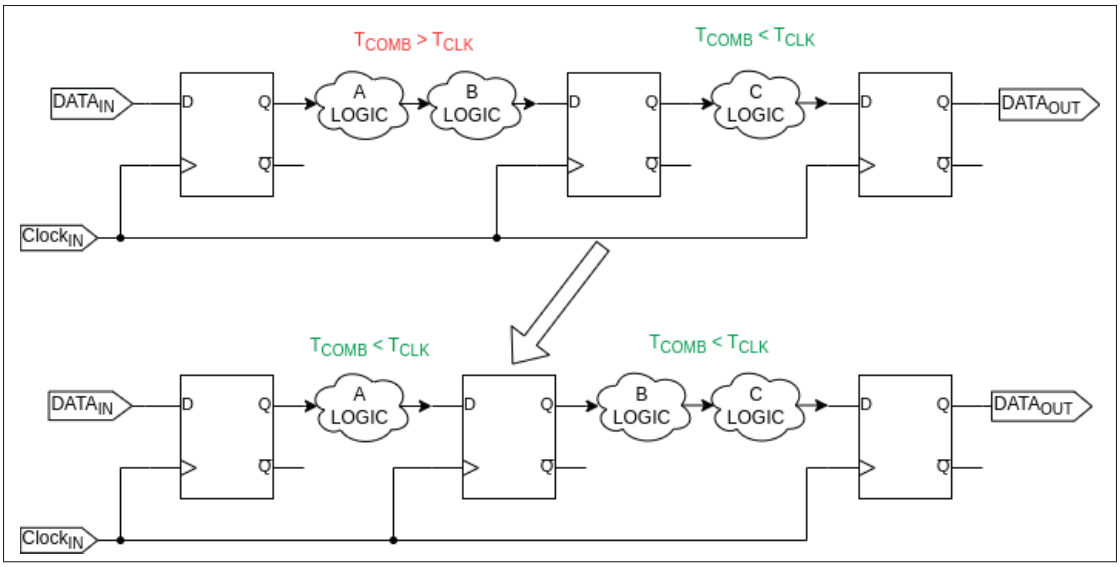


Figure 1.5 Procédure du recadencement

communément utilisées (Meher, 2016) et pour accélérer la vérification de l'équivalence du circuit recadencé (Jiang & Brayton, 2006), le tout en respectant les deux objectifs principaux, soit la réduction de la période de l'horloge et ainsi que la minimisation de la surface occupée par les registres dus au potentiel dédoublement (Shenoy, 1997).

1.2 Stratégie asynchrone

À l'opposé de la stratégie synchrone, on retrouve la stratégie asynchrone. Cette dernière n'utilise aucun signal de référence temporel global pour synchroniser les éléments séquentiels, mais plutôt une stratégie locale pour détecter la validité des résultats. Dans la littérature scientifique, on retrouve deux principaux encodages, illustrés dans la figure 1.6, pour implémenter cette stratégie : l'encodage multirails et l'encodage BD (Sparsø, 2020).

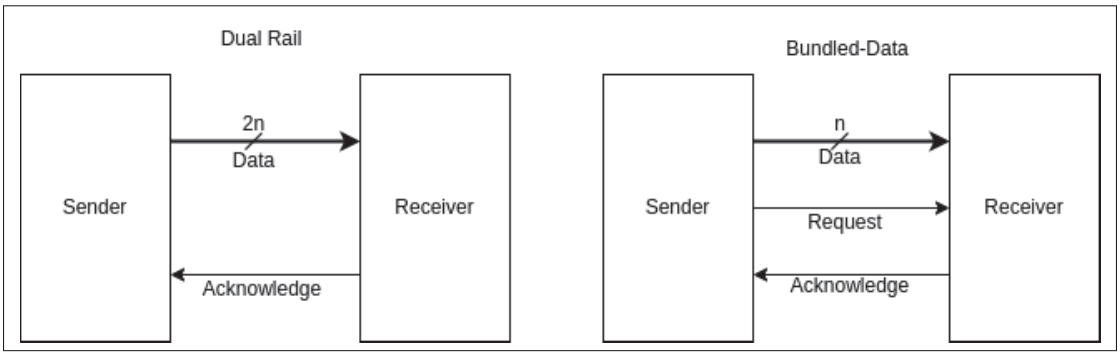


Figure 1.6 Encodage deux rails (gauche) et BD (droit)

L'encodage multirails encode la validité du résultat directement dans le chemin de données en utilisant un encodage de N bits par vecteur de M bits, où $N > M$. L'encodage le plus populaire dans la littérature scientifique est celui sur deux rails, qui utilise deux bits N par bit d'information M. Il est utilisé afin de représenter un état de repos, un état logique haut et un état logique bas. La relation que propose cet encodage entre le chemin de contrôle et le chemin de données le rend indépendant aux délais du circuit, au coût d'une augmentation de la surface due au dédoublement des bits d'information.

En contrepartie, l'encodage BD utilise un canal de communication indépendant du chemin de donnée pour indiquer la validité du résultat. Puisque la relation du chemin de contrôle et du chemin de donnée est découplée, des délais doivent être insérés sur le chemin de contrôle pour assurer qu'il arrive avant le délai du chemin combinatoire le plus long.

En temps normal, ce délai est implémenté par l'entremise d'une chaîne de tampons, cependant une récente méthode qui détecte la fin de l'opération en fonction du courant consommé par le circuit a été développée par Huang, Xiao, Li & Yu (2022) pour réduire la surface nécessaire à la synchronisation. Bien que ce type de structure soit en mesure de diminuer la surface du circuit, elle reste moins avantageuse parce qu'elle dépend d'une cellule analogique non standardisée, ce qui diminue sa compatibilité avec les outils EDA synchrones.

1.2.1 Modèle d'analyse

L'implémentation des circuits asynchrones BD se fait par l'utilisation de contrôleurs qui communiquent entre eux avec des signaux de requête et d'acquiescement comme le présente la figure 1.7. Plusieurs protocoles ont été développés pour gérer les transactions, mais deux d'entre eux se démarquent par leur popularité.

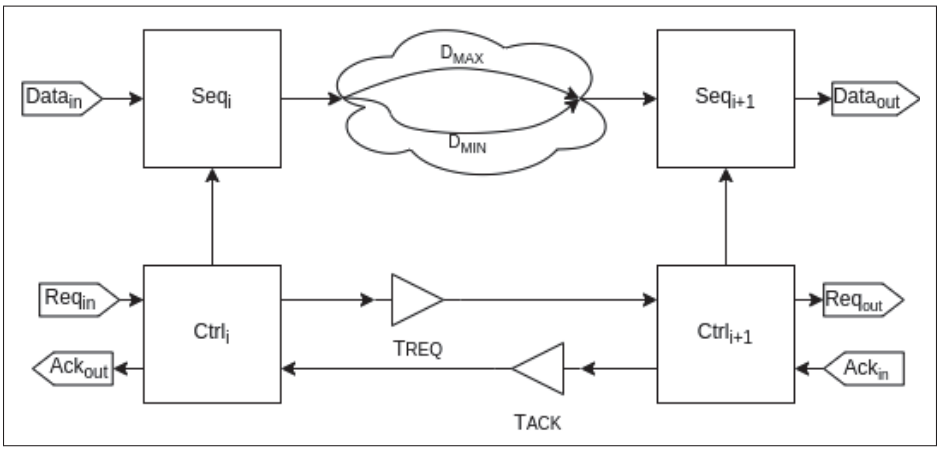


Figure 1.7 Modèle d'analyse pour un circuit asynchrone Bundled-Data

Le premier protocole, connu sous le nom du protocole de retour à zéro (RZ), contient quatre phases. Les deux premières phases consistent à transmettre la donnée, tandis que les deux dernières réinitialisent le canal de communication. Il est illustré par la figure 1.8 .

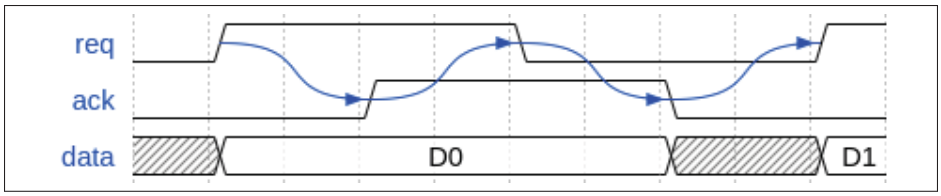


Figure 1.8 Protocole quatre phases (RZ)

Le deuxième protocole, connu sous le nom du protocole du non-retour à zéro (NRZ), contient seulement deux phases : la donnée est transmise lors de la première phase et la deuxième phase réinitialise l'état de la transaction. Il est illustré dans la figure 1.9.

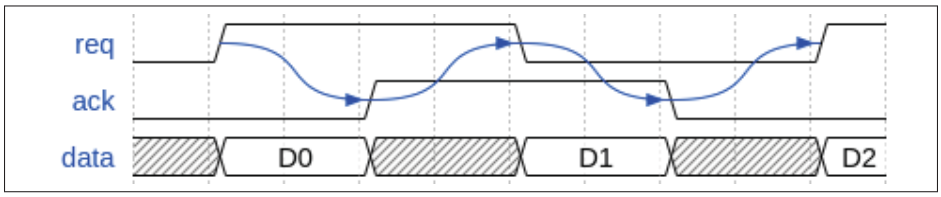


Figure 1.9 Protocole deux phases (NRZ)

Contrairement au protocole RZ, il a l'avantage d'avoir un débit plus grand, au coût d'un circuit de conversion de phase plus volumineux et d'une difficulté accrue quant au débogage (Zhou, 2022).

Une des difficultés de l'analyse des circuits asynchrones se situe lorsqu'il est question de représenter les multiples relations entre chaque signal afin de garantir une séquence précise d'évènements concurrents. En effet, la stratégie asynchrone se voit incompatible avec les modèles d'analyse synchrones, soit les graphes pour MEF et les chronogrammes, car ils transmettent difficilement l'information du séquencement du circuit de contrôle. Par conséquent, le réseau de Pétri a été l'un des premiers outils mathématiques utilisés pour modéliser les transactions asynchrones par sa capacité à modéliser la concurrence par des échanges de jetons.

Malgré sa modélisation fiable des évènements concurrents, une des limitations observées du réseau de Pétri était l'impossibilité de le synthétiser en un circuit logique à cause de son niveau

d'abstraction trop élevé. Cela a donc mené à la création du graphe de transition de signaux (STG) par Chu (1986), une sous-classe du réseau de Pétri décrit par un triplet (T, S, R), où T consiste en un lot de transitions, S représente un lot d'états possibles et R définit la relation de préséance des transitions. La modélisation immédiate des transitions T, qu'elles soient montantes ou descendantes, est ce qui permet au STG d'être directement synthétisé en un circuit logique.

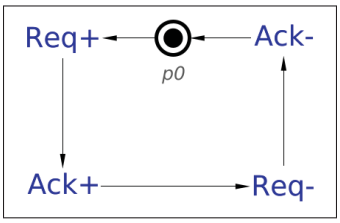


Figure 1.10 Graphe des transitions de signaux pour du protocole asynchrone

La figure 1.10 est une représentation par STG, conçue avec l'outil *Workcraft*, qui décrit la séquence des transactions des signaux de contrôle pour tout protocole asynchrone. Le jeton *p0* permet d'identifier le point de départ ainsi que l'état initial des signaux. Dans un premier temps, il peut être déduit que les signaux REQ et ACK commencent à un niveau logique bas étant donné que leurs prochains évènements seront des fronts montants. Dans un deuxième temps, on peut observer comment la propagation du jeton à travers le STG génère les transitions observées dans les figures 1.8 et 1.9, car ce STG omet l'instant où la donnée est lancée.

1.2.2 Contrôleurs

Comme indiqué précédemment, la synchronisation des circuits asynchrones BD se fait par l'entremise de contrôleurs qui communiquent entre eux avec un protocole de communication prédéfini. Ces contrôleurs sont généralement composés d'un circuit de conversion de phase et d'un élément de mémoire pour sauvegarder l'état de la transaction.

De plus, à cause de la nature non linéaire des structures pouvant se trouver dans le chemin de données, le contrôleur doit être en mesure d'implémenter certaines fonctions non linéaires

telles que la fourche (*fork*), l'union (*join*), la fusion (*merge*) et le multiplexage (*mux*). On retrouve plusieurs contrôleurs dans la littérature scientifique pour la synchronisation des circuits asynchrones BD. Cependant, seulement trois d'entre eux seront présentés : la porte C de Muller, le contrôleur *Mousetrap* et le contrôleur *Click Element* (CE).

Les premières avancées dans le développement des contrôleurs ont été réalisées avec la création de la porte C par Muller (1955) dans son document sur la théorie des circuits asynchrones. Cette théorie découlait du besoin d'un élément capable de synchroniser deux entrées indépendamment de la vitesse du circuit.

Par définition, la sortie de la porte C ne varie qu'au moment où les deux entrées ont la même valeur, sinon l'état en sortie est conservé, tel qu'observé dans la figure 1.11. Cette structure décrit les besoins essentiels des circuits de contrôle, soit un circuit pour gérer la phase des entrées et un élément de stockage pour conserver l'état des transactions.

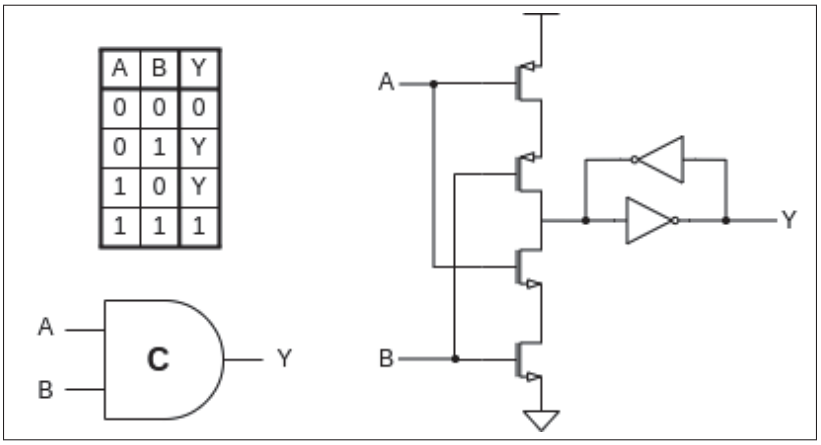


Figure 1.11 Porte C avec table de vérité (gauche) et représentation transistor (droite)

La première utilisation de la porte C comme contrôleur (voir la figure 1.12) dans une stratégie BD a été réalisée par Sutherland (1989) avec l'invention du micropipeline. La structure développée est basée sur une architecture à verrou de type capture-passe. Dans cette structure, on retrouve un nombre pair de verrous, où un d'entre eux est bloqué pour capturer la donnée alors que l'autre

est ouvert pour la laisser passer, permettant ainsi le transfert de données d'un étage vers l'autre sans écrasement. Le grand problème du micropipeline vient de deux constatations. La première constatation est que l'utilisation de la structure capture-passe divise la performance du pipeline en deux à cause du fait qu'un verrou sur deux est bloqué en tout temps. La deuxième constatation vient de l'utilisation de la porte C, qui est une cellule non présente dans les librairies fournies par les manufacturiers de semi-conducteurs.

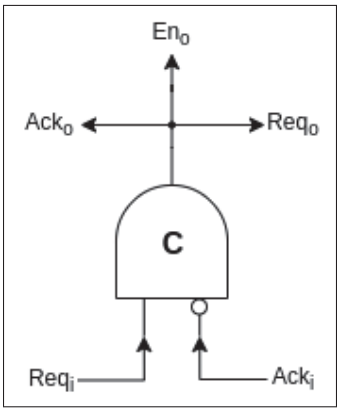


Figure 1.12 Contrôleur basé sur la porte C

Dans un effort pour créer un pipeline asynchrone plus rapide et plus standard que le micropipeline, Singh & Nowick (2007) ont développé un nouveau type de contrôleur nommé *Mousetrap* (voir figure 1.13). Contrairement au micropipeline, le *Mousetrap* n'utilise pas de porte C, mais plutôt un verrou et une porte XNOR, ce qui le rend compatible avec les cellules standardisées typiquement fournies par les manufacturiers de semi-conducteurs.

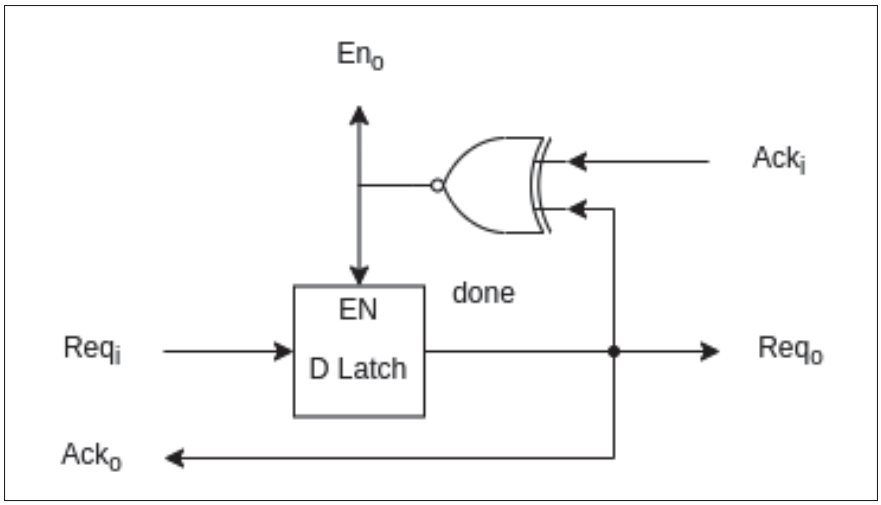


Figure 1.13 Contrôleur *Mousetrap*

Comme son nom l'indique, lorsque le signal de requête est reçu par le contrôleur, le verrou se bloque et capture la donnée à l'entrée de l'étage, tel un piège à souris. En même temps, le verrou de l'étage précédent est débloqué afin de laisser entrer la prochaine donnée dans le pipeline, tel un réarmement du piège à souris. Cette structure augmente le débit du pipeline, car les verrous agissent comme des bascules, ce qui évite l'utilisation d'une architecture de type capture-passe.

Bien que le contrôleur *Mousetrap* résolve les principaux problèmes du micropipeline, il est fondamentalement désavantagé, dans son implémentation avec les outils EDA synchrones, par son utilisation de verrous. En effet, étant donné que ces outils ont été développés dans le but d'implémenter des circuits synchrones, les algorithmes d'optimisations ont été conçus en fonction de l'élément de synchronisation principalement retrouvé dans cette stratégie de synchronisation, soit la bascule sensible au front montant.

Pour tirer avantage de cette particularité, un contrôleur qui utilise une bascule D comme élément de stockage a été développé par Peeters, te Beest, de Wit & Mallon (2010) sous le nom de *Click Element* (voir figure 1.14. La structure proposée pour le circuit du CE, schématisée dans la figure 1.14, a été conçue pour fonctionner selon le protocole deux phases. Toutefois, il est généralisé dans leur article pour permettre son utilisation, peu importe le protocole choisi.

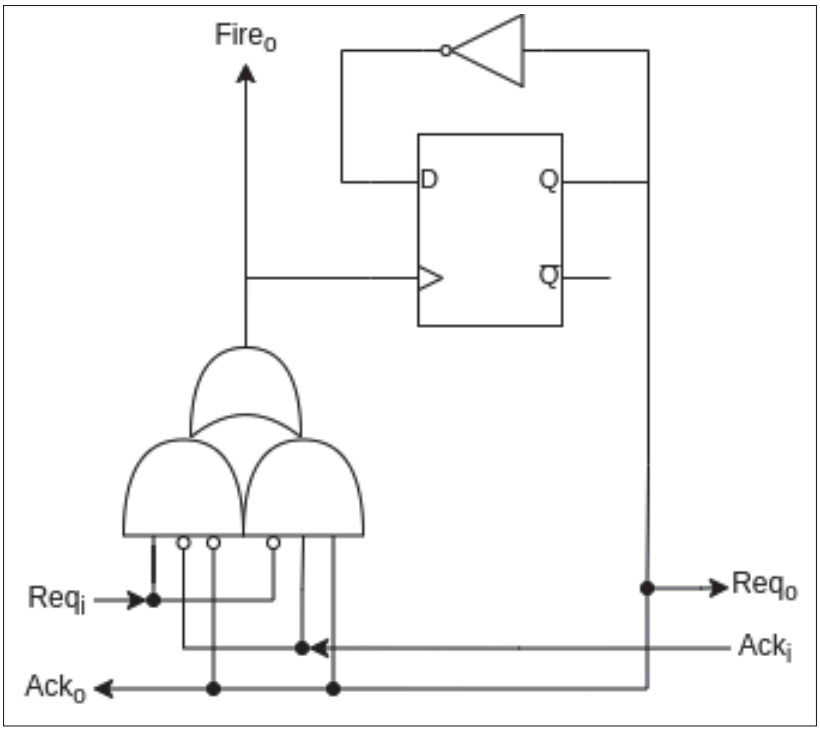


Figure 1.14 Contrôleur *Click Element*

Son implémentation consiste en un circuit convertisseur de phase qui génère une impulsion à transition montante, soit le signal connu sous le nom de *Fire*, afin d'activer le port d'horloge de la bascule et de permettre la propagation entre l'entrée D et la sortie Q. Pour ces raisons, son utilisation est favorisée dans la présente recherche par rapport aux autres contrôleurs.

1.2.3 Analyse statique des délais

Contrairement aux circuits synchrones, les circuits asynchrones ne possèdent pas de signal de synchronisation globale pour contraindre les délais du circuit. L'absence de référence temporelle force le concepteur à supposer un mode d'opération fondamentale afin d'éviter la possibilité d'aléas logiques. Ce mode permet, selon Sparsø (2020), le changement d'état uniquement lorsque le circuit s'est stabilisé, réduisant les performances. Étant donné le besoin de contraintes pour la conception de circuits performants, Stevens, Ginosar & Rotem (1999) ont développé les contraintes de délais relatifs (RTC) pour énoncer mathématiquent la relation temporelle entre

les différents chemins d'un circuit.

$$pod \mapsto poc_1 + m \prec poc_2 \tag{1.7}$$

La définition d'une RTC est régie par l'équation 1.7. Elle définit un point de divergence commun (pod), deux points de convergence différents (poc) et une marge de temps entre les chemins (m). L'équation se lit comme suit : la propagation du chemin $pod \rightarrow poc_1$ doit arriver une durée de temps m avant la propagation du chemin $pod \rightarrow poc_2$.

L'avantage de l'utilisation des RTC vient du fait qu'elles permettent d'assurer qu'un circuit sera performant et robuste si, et seulement si, le concepteur a choisi un bon lot pour contraindre son circuit. Effectivement, de nombreuses RTC peuvent être combinées pour assurer le bon fonctionnement du circuit, ce qui nécessite une analyse des différentes combinaisons possibles avec des critères spécifiques pour choisir un lot optimal (Manoranjan & Stevens, 2016). Pour automatiser cette analyse, des outils externes ont été développés par Lee, Sharma & Stevens (2016) afin d'identifier le lot optimal de RTC à utiliser.

Malgré cet avancement, il reste que l'ajout d'une étape dans le flux de conception n'est pas avantageux pour l'adoption de la stratégie. Pour venir à bout de cela, Gimenez, Simatic & Fesquet (2019) ont développé un lot classifié, générique et réduit de RTC applicables à tous les circuits asynchrones de type BD. Comme ce lot est générique, il revient au concepteur de déterminer l'applicabilité des contraintes du lot à son circuit, mais permet d'éviter l'utilisation de nouvel outil.

Bien que puissant, il reste que l'utilisation des RTC est fondamentalement incompatible avec les outils STA synchrones à cause du fait que ces outils sont conçus avec la prémisse de traiter des circuits synchronisés avec une horloge globale. Heureusement, étant donné la grande diversité de contraintes qui peuvent être appliquées pour assurer la modélisation fiable d'un circuit synchrone, il a été possible de développer des méthodes exploitant les cas limites de ces contraintes afin d'analyser les circuits asynchrones BD. Trois d'entre elles, soit les méthodes du *min-max*, de

l'*Adaptive Delay Matching* (ADM) et du *Local Clock Set* sont présentées pour l'implémentation des RTC.

La première méthode, nommée *min-max*, est développée par (Stevens, Xu & Vij, 2009). Elle est basée sur la description des RTC à l'aide des contraintes SDC pour la spécification du délai minimal et du délai maximal. Elle consiste à identifier une RTC ($pod \mapsto poc_1 + m \prec poc_2$) et à la contraindre avec les contraintes SDC *set_min_delay* pour le chemin le plus long ($pod \rightarrow poc_2$) ainsi qu'avec la contrainte SDC *set_max_delay* pour le chemin le plus court ($pod \rightarrow poc_1$). La méthode est cependant limitée, car elle ne peut analyser les boucles de rétroaction combinatoires, ce qui nécessite l'utilisation de la contrainte SDC *set_disable_timing* pour couper les nœuds de la boucle et obtenir un graphe acyclique.

La coupe des chemins d'analyse est une étape qui complique la manœuvre par le fait qu'il faut dans un premier temps identifier deux nœuds de la boucle de rétroaction à couper, puis dans un deuxième temps effectuer deux analyses statiques pour pouvoir prendre en compte les délais des nœuds omis. De plus, l'identification des nœuds à couper est une tâche ardue en elle-même, car la spécification d'un mauvais nœud peut omettre des chemins importants lors du STA.

Une autre difficulté des différences de précision des résultats des STA obtenus lors de la synthèse logique et lors du PNR. Étant donné que ces délais changent avec la concrétisation des modèles parasitiques, des heuristiques doivent être utilisées pour modifier automatiquement la valeur des délais spécifiés lors de la synthèse pour les adapter au délais plus réalistes du PNR (Sharma & Stevens, 2020). Un outil externe, nommé ACDC, a été mis au point par Gibiluka, Moreira & Vilar Calazans (2015) pour éliminer ces complexités en automatisant le flux de développement de cette méthode par la description des RTC du circuit à compiler dans un fichier *Extensible Markup Language* (XML).

La deuxième méthode, nommée ADM, est développé dans les travaux de Wu *et al.* (2019) et de Wu *et al.* (2021). Elle propose un flux pour transformer un pipeline synchrone en un pipeline asynchrone en utilisant des CE comme contrôleurs. L'idée principale de la méthode est de déclarer une horloge sur le signal *fire* du premier CE avec la contrainte SDC *create_clock*,

puis de générer des horloges décalées avec la contrainte SDC *create_generated_clock* via l'argument *-edge_shift* pour tous les CE subséquents en fonction du délai du chemin de données. La contrainte SDC *set_min_delay* est ensuite utilisée pour insérer les tampons sur la ligne de requête afin d'égaler le délai combinatoire du chemin de données.

Bien que simple dans son implémentation, la méthode n'a été validée que pour le contrôleur du CE et ne permet pas l'analyse des structures de contrôle non linéaires. De plus, la méthode n'est pas en mesure d'analyser les délais induits par la logique interne du contrôleur, ce qui pousse la contrainte SDC *set_min_delay* à insérer plus de tampons qu'il ne serait nécessaire pour assurer un fonctionnement optimal. Enfin, ces tampons sont insérés pseudo-manuellement lors de la synthèse logique, cependant il serait préférable qu'ils soient instanciés automatiquement par l'outil de PNR lors de l'étape de la construction de l'arbre de l'horloge pour obtenir une meilleur qualité des résulats (QOR).

La troisième et dernière méthode est celle du LCS, développée par Gimenez, Cherkaoui, Cogniard & Fesquet (2018). Elle propose une stratégie pour implémenter directement les équations des RTC à l'aide de définitions de lots d'horloges locales, permettant ainsi l'analyse des circuits asynchrones avec les outils EDA synchrones. Elle se trouve à être largement supérieure aux deux autres méthode pour trois raisons. La première est qu'elle permet l'analyse de structures de contrôle non linéaires. La deuxième est qu'elle permet l'analyse des boucles de rétroaction combinatoires sans désactiver des chemins d'analyse. La dernière est qu'elle est intrinsèquement adaptée aux caractéristiques des outils de STA synchrone à cause de la manière dont les horloges sont traitées.

Bien que la méthode se concentre sur l'étape du STA, elle propose trois stratégies pour l'implémentation des tampons : une méthode manuelle et simpliste qui utilise la commande *insert_buffer*, une méthode quasi automatisée qui utilise la contrainte SDC *set_min_delay*, tout comme les méthodes *min-max* et ADM, puis une méthode entièrement automatisée qui utilise les capacités de l'outil de PNR pour l'insertion des délais lors de l'étape de la construction de l'arbre de l'horloge (Gimenez, 2021). Comme cette dernière est la plus précise, c'est elle qui a

été utilisée dans le cadre des présents travaux pour l'implémentation des tampons. Cependant, comme elle a été développée pour son utilisation dans l'outil *Integrated Circuit Compiler* (ICC) de Synopsys, il a été nécessaire de faire une adaptation des commandes pour l'outil utilisé dans ces présents travaux : *Integrated Circuit Compiler 2* (ICC2).

Bien que la méthode LCS soit puissante, quelques désavantages sont toujours présents lors de son utilisation. Dans un premier temps, la méthode ne spécifie pas comment contraindre les ports des entrées et des sorties du circuit pour vérifier le fonctionnement aux frontières.

Ensuite, elle nécessite la déclaration d'une grande quantité d'horloges pour décrire un chemin de contrôle, ce qui entraîne la création d'un très grand fichier de contraintes, même pour un circuit simple. De plus, ces horloges doivent être spécifiées sur les broches des cellules du circuit, rendant le fichier entièrement dépendant de la structure de contrôle et du nom des broches des cellules.

Finalement, la méthode qu'elle propose pour manuellement contraindre les délais du chemin de donnée est erronée, ce qui empêche l'optimisation du circuit par l'absence de convergence lors de la synthèse et génère un circuit trop gros et très énergivore.

Des optimisations sont donc proposées pour permettre l'analyse aux frontières ainsi que pour permettre la convergence lors de la synthèse.

1.2.4 Analyse des performances

Pour ce qui est de la mesure des performances de la stratégie asynchrone, étant donné l'absence d'une horloge globale permettant de dicter la performance du circuit, il a été nécessaire de redéfinir les variables de performances afin de les dissocier de ce signal. Les variables affectées, relatées par Sparsø (2020) de Williams (1994), sont la latence L, la période T et le débit D.

En premier, la latence L d'un système se décrit généralement comme étant le délai entre l'application d'un stimulus en entrée et l'observation du résultat en sortie. Contrairement à une

synchronisation synchrone, qui est approximée par l'équation 1.3, l'absence de synchronisation globale nécessite une adaptation aux caractéristiques de la stratégie.

Dans un système asynchrone, la validité d'une donnée est déterminée en fonction des signaux de contrôle générés par le contrôleur. Étant donné la nature bidirectionnelle des transactions entre les contrôleurs, on retrouve deux définitions pour la latence : la latence de propagation directe L_F et la latence de propagation inverse L_R. La première fait référence au temps total de propagation d'un signal de requête (REQ), tandis que la deuxième fait référence au temps total de propagation d'un signal d'acquiescement (ACK). Il est utile de noter que seule la latence de propagation directe L_F est utilisée étant donné qu'il s'agit du seul type de latence présent dans un circuit synchrone ($L = L_F$).

Les deux dernières variables, soit la période et le débit, font référence à la quantité de résultats que le circuit est en mesure de générer par unité de temps. Dans un système synchrone, on peut les comparer respectivement à la période et à la fréquence de l'horloge. Pour la stratégie asynchrone cependant, la période P (équation 1.8) est définie comme étant la moyenne à long terme de la différence du temps entre deux échantillonnages subséquents du résultat en sortie. On y retrouve la variable N qui représente le nombre de données capturées, et la variable $T_{D[n]}$, qui représente le moment où la donnée n est valide.

$$P = \frac{1}{N} \sum_{n=1}^{N} T_{D[n]} - T_{D[n-1]} \tag{1.8}$$

Quant à lui, le débit D (équation 1.9) se définit comme étant l'inverse de la période.

$$D = \frac{1}{P} \tag{1.9}$$

Ces définitions sont utilisées pour caractériser la performance de toutes les stratégies de synchronisation.

1.2.5 Octasic

Une autre stratégie de synchronisation asynchrone, que l'on retrouve dans la littérature scientifique, est celle développée par la compagnie Octasic (Laurence, 2012), laquelle est schématisée par la figure 1.15. Elle repose sur l'utilisation d'une impulsion pour contrôler les éléments séquentiels via le jumelage du chemin de données et du chemin de contrôle par l'entremise d'une *Table Lookup Unit* (TLU). La caractéristique de cette unité est qu'elle permet de sélectionner le moment où l'impulsion de capture *done* est générée en fonction de l'opération *OP* effectuée dans le chemin de données.

La structure permet, dans un premier temps, de réduire la surface étant donné que l'ensemble de la logique est implémenté de façon combinatoire, puis, dans un deuxième temps, de réduire la consommation de puissance par le retrait du signal d'horloge.

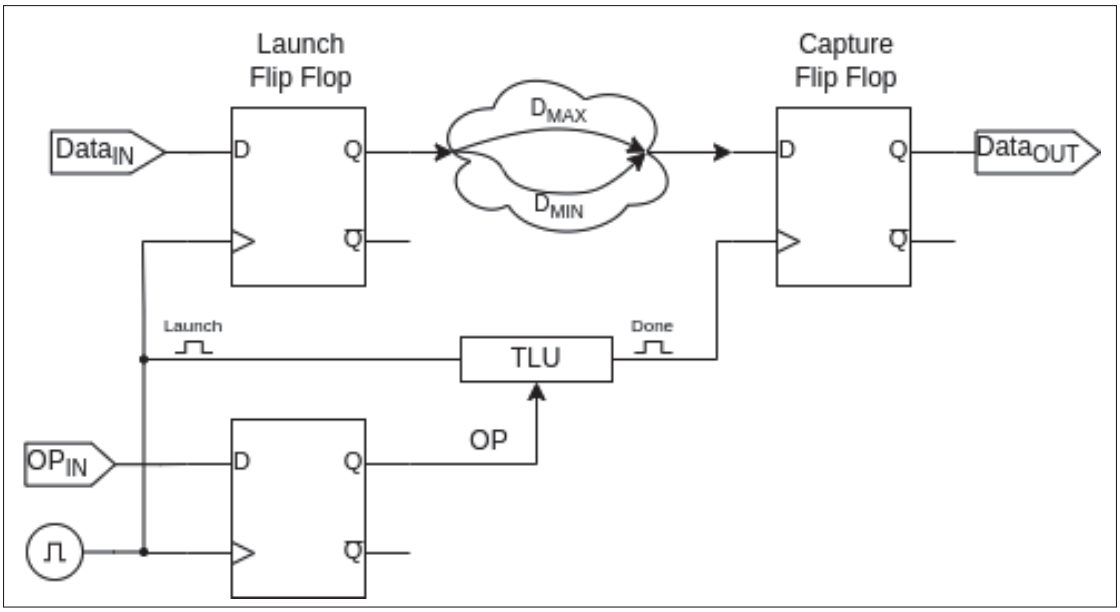


Figure 1.15 Synchronisation de type Octasic

Par définition, la synchronisation ne permet d'exécuter qu'une seule opération à la fois, cette dernière étant contrôlée par un signal d'impulsion généré par un système externe sous forme d'anneau, ce qui limite la période équivalente au délai de la logique combinatoire. Cependant,

comme plusieurs opérations *OP* différentes peuvent être exécutées pour la même logique combinatoire, la période du bloc combinatoire devient égale à la moyenne pondérée des latences des différents chemins en fonction de leur proportion d'exécution.

Néanmoins, comme une seule opération est implémentée dans la logique combinatoire, soit l'opération du MAC, les avantages obtenus de l'utilisation d'une TLU deviennent futiles. Ce faisant, la stratégie à donc un débit comparable à une stratégie synchrone sans étage de pipeline. Pour ces raisons, cette stratégie n'est pas évaluée.

1.3 Stratégie du *Wave Pipelining*

Le *Wave Pipelining*, aussi connu sous le nom de pipeline à débit maximal (*maximum rate pipeline*), est une stratégie développée par Cotten (1969) dans le but de pousser la performance des circuits intégrés au-delà de ce qui est possible avec la stratégie de synchronisation synchrone.

Les caractéristiques du modèle d'analyse seront initialement présentées pour comprendre les principes de la stratégie. L'idée du balancement des délais sera ensuite développée avec les méthodes présentes dans la littérature scientifique. La faible tolérance à la variabilité sera finalement présentée pour expliquer la raison de sa faible popularité.

1.3.1 Modèle d'analyse

Le principe du fonctionnement de cette stratégie se réduit à modéliser la logique combinatoire comme une ligne de transmission ayant un délai de propagation T_D, tel qu'illustré dans la figure 1.16, permettant ainsi la transmission de plusieurs vagues de données en parallèle.

Cette parallélisation temporelle avait comme caractéristique de générer des performances substantielles pour l'époque, en l'occurrence, par la génération d'un débit de deux à trois fois plus vite par rapport à son équivalent synchrone, et ce pour une plus petite surface (Burleston, Cotton, Klaus & Ciesielski, 1994).

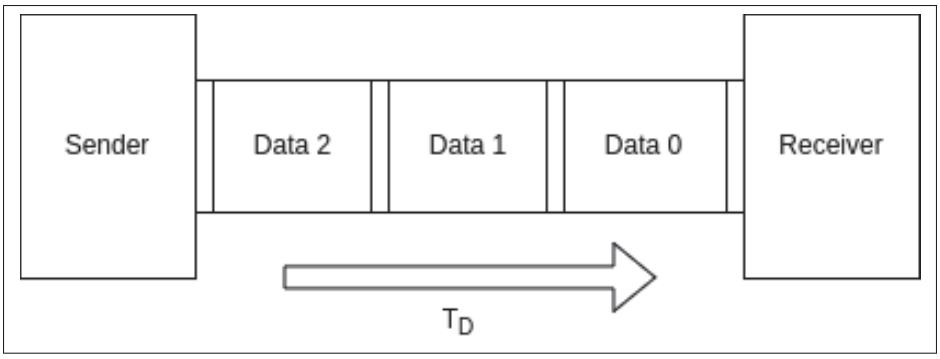


Figure 1.16 Fonctionnement du *Wave Pipelining*

Malgré cela, l'utilisation de la technologie demeure très peu fréquente à cause de la difficulté de son implémentation. La structure fondamentale de la stratégie, présentée par la figure 1.17, est composée de deux registres, un en entrée et un en sortie, le tout entourant la logique combinatoire du module.

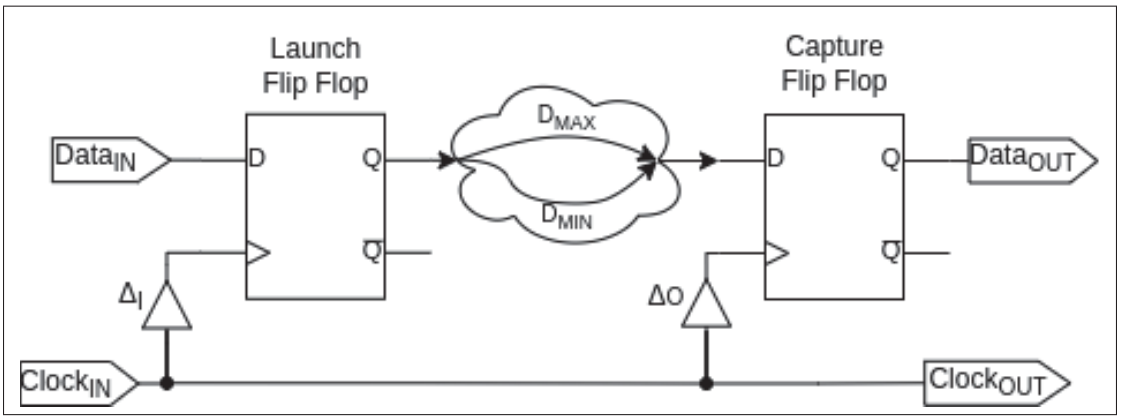


Figure 1.17 Modèle de synchronisation du *Wave Pipelining*

On retrouve principalement deux méthodes pour synchroniser les circuits utilisant le WP (Zhang & Sridhar, 1994). La première méthode, originalement développée par Cotten (1969), est de compenser le délai combinatoire avec un nombre prédéfini de cycles d'horloges ($K \cdot T_{CLK}$) sans aucun délai asymétrique. La deuxième méthode, obtenue par généralisation des contraintes originales par Gray, Liu & Cavin (1994), est d'ajouter une ligne à retard (Δ_O) sur le signal de contrôle du registre en sortie afin de compenser une partie ou l'entièreté du délai combinatoire.

La combinaison de ces deux paramètres donnait ainsi naissance à l'équation 1.10, qui définissait la relation entre le délai de propagation T_D et la stratégie de compensation.

$$T_D = K \cdot T_{CLK} + \Delta_O \tag{1.10}$$

En tout égard, à notre époque, il y a lieu d'apporter une légère correction à l'équation 1.10 proposé par Gray *et al.* (1994). En effet, leur équation suppose que le délai asymétrique de l'horloge de lancement $\Delta_I = 0$, ce qui ne permet pas de généraliser la stratégie de compensation. Ce délai est plutôt pris en compte dans la définition des contraintes de préparation et de maintien par l'utilisation explicite des termes Δ_I et Δ_O. La modification proposée est de remplacer le terme Δ_O de l'équation 1.10 par le terme Δ, où $\Delta = \Delta_O - \Delta_I$, ce qui donne l'équation 1.11.

$$T_D = K \cdot T_{CLK} + \Delta \tag{1.11}$$

Cela permet de représenter correctement l'effet du délai asymétrique entre l'horloge de capture Δ_O et l'horloge de lancement Δ_I. Pour valider cette modification et dériver l'équation des contraintes, le chronogramme présenté dans la figure 1.18 démontre un exemple de pipeline en vague où le paramètre $K = 1$.

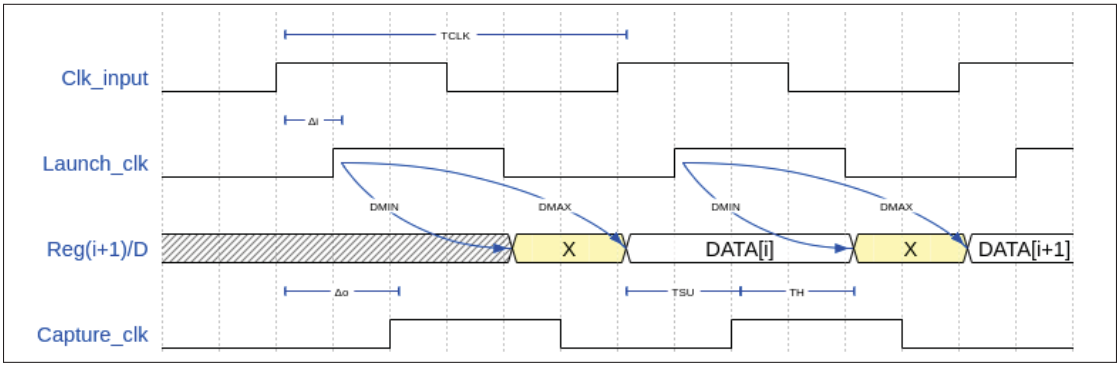


Figure 1.18 Chronogramme du pipeline en vague avec $K = 1$

Pour modéliser la contrainte de préparation, le WP spécifie qu'il faut que le délai combinatoire le plus long (D_{MAX}), incluant le temps de préparation (T_{SU}), doive être compensé par le délai de

la stratégie de contrôle T_D, comme indiqué par l'équation 1.12.

$$T_D \geq D_{MAX} + T_{SU} \tag{1.12}$$

Au niveau de la contrainte de maintien, il est logiquement possible de concevoir que la seule façon d'écraser l'ancienne donnée $(i-1)$ est d'en propager une nouvelle (i), ce qui se modélise par l'addition du prochain coup d'horloge T_{CLK} et du délai combinatoire le plus rapide D_{MIN}. Étant donné que l'ancienne donnée $(i-1)$ est seulement capturée après le délai de la stratégie de contrôle T_D, la nouvelle donnée (i) doit arriver après ce délai additionné au temps de maintien (T_H) de l'élément de stockage, comme indiqué par l'équation 1.13.

$$T_{CLK} + D_{MIN} \geq T_D + T_H \tag{1.13}$$

Les contraintes 1.12 et 1.13 peuvent ensuite être combinées en l'équation 1.14 pour isoler ce qui contraint la période de l'horloge T_{CLK}.

$$T_{CLK} \geq D_{MAX} - D_{MIN} + T_{SU} + T_H \tag{1.14}$$

Dans cette équation, on constate que la période de l'horloge, contrairement à la stratégie synchrone où elle dépend du plus long chemin combinatoire, dépend de la différence entre le délai combinatoire maximal (D_{MAX}) et le délai combinatoire minimal (D_{MIN}). Il devient donc avantageux de réduire cette différence en utilisant des méthodes pour balancer la logique combinatoire et ainsi augmenter le débit du circuit.

1.3.2 Méthode de balancement

À l'époque de l'avènement de cette stratégie, les premiers travaux effectués portaient principalement sur le développement d'outils de synthèse spécialisés dans le balancement de la logique combinatoire (Burleston *et al.*, 1994) pour optimiser l'équation 1.14 et ainsi augmenter le débit. Cependant, les efforts actuels ont la caractéristique de proposer des solutions qui

s'imbriquent dans les outils commerciaux synchrones compte tenu de leur versatilité, de leur maturité au niveau des algorithmes d'optimisation et de leur adoption répandue dans l'industrie du semi-conducteur.

Une de ces approches modernes a notamment permis la proposition d'un nouveau flux de conception qui comporte un script exécuté dans l'outil de synthèse Design Compiler (DC) de Synopsys pour équilibrer les délais combinatoires du circuit. Les résultats qu'elle présentait étaient une réduction de la variation entre D_{MAX} et D_{MIN} inférieure à 20%, ce qui permet la minimisation de l'équation 1.14 (Kim & Kim, 2005).

Plus tard, Kra, Noy & Teman (2020) ont soulevé une limitation de cette première approche. En effet, pour fonctionner, la librairie des cellules standardisées doit être limitée à une quantité restreinte de cellules, causant une explosion de la surface utilisée. Cela les a poussés à développer *WavePro*, un outil externe beaucoup plus robuste qui s'intègre à DC et utilise l'entièreté de la librairie des cellules pour effectuer la même tâche de balancement.

Cette solution proposée dans l'article souffre d'une certaine lacune en ce qu'il omet, tout comme l'article qu'il critique, les résultats quant à la surface utilisée du circuit. Cela rend l'utilisation d'un algorithme de balancement du chemin de donnée difficilement justifiable, étant donné qu'il est impossible de déterminer à quel point la surface est affectée.

De plus, un aspect désavantageux de ces méthodes est qu'elles dépendent d'algorithmes externes aux outils EDA synchrones, impliquant qu'ils n'utilisent pas la puissance d'optimisation des algorithmes fournis par ces outils. Pour venir à bout de cette limitation, l'article de Vireen, Seetharaman & Venkataramani (2008) propose trois procédures incrémentalement manuelles pour permettre la synthèse des circuits WP avec DC au moyen des contraintes SDC.

La première procédure utilise la période de l'horloge pour contraindre itérativement le chemin le plus court jusqu'à la convergence de l'analyse du temps de maintien. La deuxième procédure utilise les commandes SDC *set_max_delay* et *set_min_delay* pour contraindre D_{MAX} et D_{MIN}

respectivement. La troisième et dernière procédure instancie manuellement des cellules ayant un délai égal et les redimensionne à chaque itération pour atteindre un délai $D_{MIN} \approx D_{MAX}$.

Un premier aspect négligé par ces procédures est qu'elles permettent uniquement d'implémenter la stratégie du WP lorsque $\Delta = 0$. Cela est dû au fait qu'elles traitent uniquement des contraintes pour le chemin de données, assumant que le délai combinatoire sera compensé par la période de l'horloge. Ensuite, un deuxième aspect négligé est qu'elles n'explicitent pas la méthode pour traiter des boucles de rétroaction, ces dernières requérant des contraintes spécifiques pour permettre leur analyse statique. Finalement, le dernier aspect négligé par ces procédures est qu'elles ne prennent pas en compte le fait que les outils de PNR possèdent les algorithmes nécessaires pour redimensionner, déplacer et ajouter des cellules afin de balancer la logique combinatoire. En effet, cela fait en sorte qu'il est seulement nécessaire de modéliser les contraintes de préparation et de maintien à l'aide de contraintes SDC et de laisser l'outil de PNR s'occuper de réparer la contrainte de maintien.

Pour ces raisons, des contraintes SDC ont été explicitement développées afin de modéliser et automatiser le flux entier de conception du WP peu importe la forme de l'équation de compensation.

1.3.3 Réponse à la variabilité

Une caractéristique défavorable à la stratégie du WP est que, dans le cas où $K > 1$, une contrainte supplémentaire affectant la période minimale de l'horloge apparaît. Cette contrainte, démontrée par l'équation 1.15, est obtenue en isolant la période de l'horloge T_{CLK} dans les équations 1.12 et 1.13 lorsque le délai de propagation T_D est substitué selon l'équation 1.11.

$$\frac{D_{MAX} - \Delta + T_{SU}}{K} \leq T_{CLK} \leq \frac{D_{MIN} - \Delta - T_H}{K - 1} \tag{1.15}$$

L'équation est approximée dans la littérature scientifique en négligeant les termes Δ, T_H et T_{SU} étant donné leur faible impact sur les délais combinatoires D_{MAX} et D_{MIN}, donnant l'équation

1.16.

$$\frac{D_{MAX}}{K} \leq T_{CLK} \leq \frac{D_{MIN}}{K-1} \tag{1.16}$$

En effet, lorsque l'on a la variable $K > 1$, le WP fonctionne uniquement pour une certaine plage de fréquences, ce qui nécessite l'utilisation de contraintes de conception très strictes pour assurer un fonctionnement valide du circuit (Kim & Kim, 2003). Cette rigidité, quant à la plage de fonctionnement, cause des problèmes lorsqu'on commence à discuter des variations PVT, car les technologies de transistor modernes ont la caractéristique d'avoir une variabilité beaucoup plus élevée que les vieux procédés de fabrication. Cette variabilité est amplifiée lors de l'addition des techniques modernes comme la variation dynamique de l'alimentation et la variation rapide de la température (Burleson, Ciesielski, Klass & Liu, 1998).

Des stratégies comme l'utilisation d'une alimentation s'adaptant aux variations (Nowka & Flynn, 1995) ou l'utilisation d'une horloge variable en fonction des variations (Kim & Kim, 2003) ont été utilisées avec succès dans des projets pour prendre en considération ces difficultés. Ils requièrent cependant une expertise dans le développement de structures analogiques, ce qui n'est pas l'objet de cette recherche.

Une autre solution, non discutée dans la littérature scientifique, serait de limiter le nombre de coups d'horloges à $K \leq 1$ pour éviter la contrainte 1.16 et éliminer la période minimale de fonctionnement. Bien évidemment, ce choix implique une augmentation de la surface du circuit par l'ajout de tampons pour compenser le délai à l'aide de la variable du délai asymétrique Δ à la place du signal d'horloge.

1.4 Stratégie pseudo-asynchrone avec emprunt d'horloge

L'analyse des trois stratégies actuellement utilisées dans la communauté scientifique à révéler clairement que chacune d'entre elles souffrait de lacunes suffisamment importantes pour justifier la création d'une nouvelle stratégie de synchronisation. C'est dans ce contexte que l'on présente la stratégie pseudo-asynchrone avec emprunt d'horloge, cette dernière ayant des caractéristiques intéressantes à étudier.

Ce qu'on définit comme stratégie pseudo-asynchrone fait référence à une catégorie de circuits qui utilise une synchronisation hybride entre les stratégies synchrones et asynchrones. Ce type de synchronisation implique l'utilisation d'une horloge locale pour la synchronisation des éléments de stockage du chemin de données ainsi que l'utilisation de tampons sur le chemin de contrôle pour compenser le délai combinatoire. La stratégie du pipelinage avec emprunt d'horloge développé par Michel Kafrouni (2024), aussi nommé comme étant la stratégie pseudo-asynchrone avec emprunt d'horloge, est développée dans la présente section afin d'en observer les spécificités.

Cette stratégie hybride, montrée à la figure 1.19, se résume à attribuer un budget initial d'une période d'horloge par étage sous forme de tampons. Ce budget initial, qui se traduit en un délai asymétrique de $\Delta_n = T_{CLK}$ pour chaque étage du pipeline, lui donne l'allure d'une synchronisation synchrone. En plus, des tampons sont instanciés en entrée et en sortie pour compenser le temps de préparation T_{SU} (Δ_I) et le délai de propagation de la bascule de sortie T_{CQ} (Δ_O) afin d'assurer un jumelage de la donnée et du contrôle.

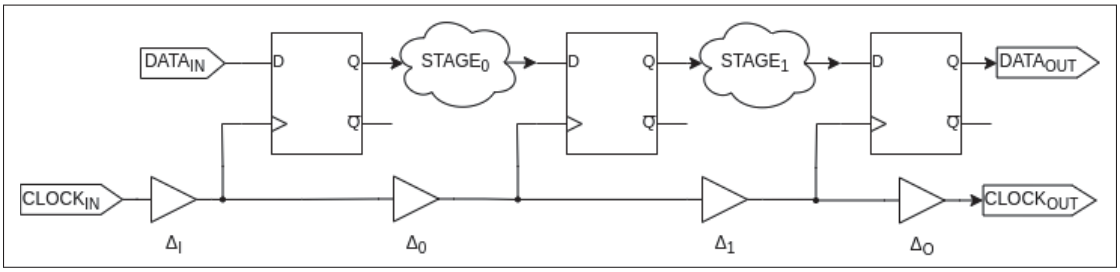


Figure 1.19 Pipeline pseudo-asynchrone avec emprunt d'horloge

Lors de la compilation du circuit, le délai combinatoire du chemin de données de chaque étage se retrouve avec une marge, positive ou négative, par rapport à la période de l'horloge, tout comme un circuit synchrone. La différence se trouve dans le fait que le budget de compensation Δ_n d'un étage n peut emprunter la marge de temps à un autre étage en déplaçant les tampons d'un étage vers l'autre, ce qui peut se comparer à un recadencement manuel.

Cette compensation se décrit par l'ajout d'une nouvelle variable d'emprunt B à l'équation du délai asymétrique en fonction de l'étage n du pipeline $\Delta_n = T_{CLK} + B_n$, où $B_n < 0$ représente un emprunt de marge et $B_n > 0$ représente une réception de marge.

La figure 1.20 présente le chronogramme d'emprunt de marge, tandis que la figure 1.21 présente le chronogramme de réception de marge.

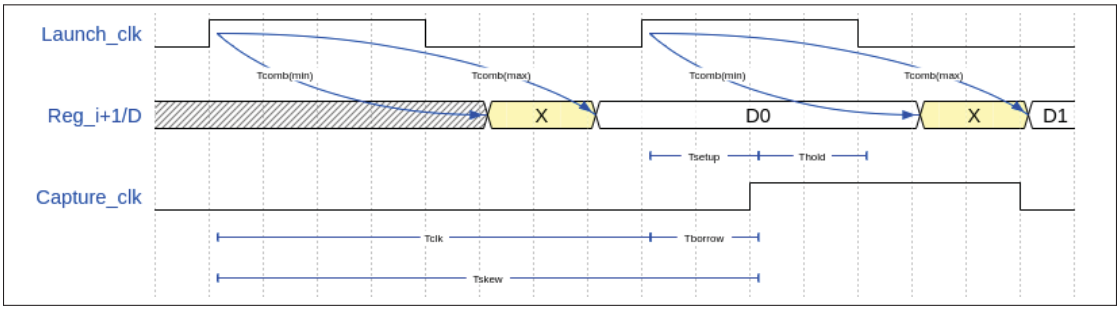


Figure 1.20 Chronogramme d'un étage de la stratégie pseudo-asynchrone empruntant de la marge

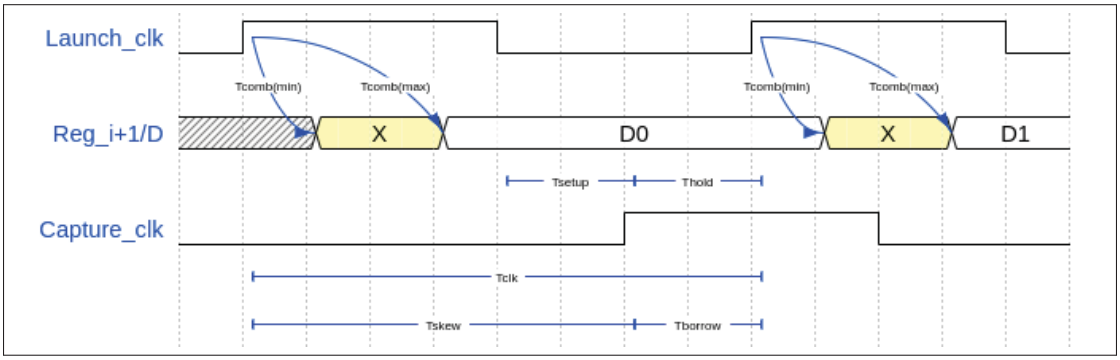


Figure 1.21 Chronogramme d'un étage de la stratégie pseudo-asynchrone donnant de la marge

Pour fonctionner correctement, il est critique que le signal de contrôle arrive à l'élément de capture après l'arrivée de la donnée, ce qui correspond à une structure asynchrone. Cependant, étant donné l'utilisation d'une horloge, il est possible de réutiliser les équations proposées dans l'article de Gray *et al.* (1994) et de modéliser la stratégie du PA comme un pipeline en vague multiétage, où chaque étage n défini le paramètre $K = 0$ de l'équation $T_D = K \cdot T_{clk} + \Delta$ et où

$\Delta = T_{CLK} + B$. On peut aussi intuitivement voir qu'en empruntant de la marge B on obtiendrait au minimum deux données se propageant simultanément dans un même étage, renforçant ainsi sa relation au WP.

Donc, en substituant les équations du WP avec les spécificités de cette stratégie hybride, il est possible de dériver les équations pour les contraintes de préparation et de maintien. Dans un premier temps, la contrainte de préparation est présentée par l'équation 1.17.

$$T_{CLK} + B \geq D_{MAX} + T_{SU} \tag{1.17}$$

On observe que si la période de l'horloge n'est pas en mesure de compenser le délai combinatoire maximal D_{MAX}, un étage peut recevoir de la marge ($B > 0$). Elle indique aussi que pour pouvoir donner de la marge ($B < 0$), la période de l'horloge doit être assez longue par rapport au délai combinatoire le plus long D_{MAX}.

Dans un deuxième temps, la contrainte de maintien est présentée par l'équation 1.18.

$$D_{MIN} - B \geq T_H \tag{1.18}$$

On observe que la quantité de marge qui peut être reçue ($B > 0$) requiert que le chemin combinatoire le plus court (D_{MIN}) soit assez long pour assurer que la capture de la donnée ne soit pas trop retardée dans le temps.

En utilisant ces deux contraintes, on peut ainsi définir le domaine de la variable d'emprunt B avec l'équation 1.19.

$$D_{MAX} + T_{SU} - T_{CLK} \leq B \leq D_{MIN} - T_H \tag{1.19}$$

En plus de ces deux contraintes, un budget est implicitement défini comme étant le nombre de registres de pipeline (N_{RP}) multiplié par la période de l'horloge, soit la relation présentée par l'équation 1.20. Ce budget observable par le fait que chaque étage du pipeline se voit attribuer

un tampon initial de T_{CLK} pour effectuer les opérations d'emprunt.

$$\sum_{n=0}^{N_{RP}} \Delta_n = N_{RP} \cdot T_{CLK} \tag{1.20}$$

Cette contrainte pourrait néanmoins avoir un effet limitant sur la valeur de la période de l'horloge, car elle nécessite que le circuit possède assez de registres de pipeline pour assurer que l'entièreté du délai combinatoire puisse être compensée par le budget alloué. Autrement, la période de l'horloge devra compenser en étant plus élevée, diminuant ainsi le débit du système.

1.5 Sommaire

Dans ce chapitre, les concepts de base associés aux stratégies de synchronisations synchrone, asynchrone et du WP ont été présentés par rapport à l'état actuel de la littérature scientifique. De plus, une stratégie présentement en développement, nommé pseudo-asynchrone avec emprunt d'horloge, a également été présentée afin d'en jauger le potentiel.

Les informations décrites, tels les modèles d'analyses, les méthodes d'implémentation et les particularités de conception, sont approfondies dans le prochain chapitre sur la méthodologie, dans le but de concrétiser la modélisation théorique en fonction des besoins de la présente recherche.

CHAPITRE 2

MÉTHODOLOGIE

Ce chapitre traite des méthodes utilisées pour générer les résultats. Dans un premier temps, le flux de travail est explicitement décrit afin d'indiquer le rôle de chaque outil dans le processus de génération des résultats. Dans un deuxième temps, les contraintes SDC spécifiques à chaque stratégie de synchronisation sont schématisées et expliquées afin de comprendre la façon dont elles sont modélisées par l'outil d'analyse statique. Finalement, les schémas pour chaque implémentation sont décrits afin de comprendre les caractéristiques architecturales des circuits.

2.1 Flux de travail

Le flux indiqué dans la figure 2.1 décrit le processus utilisé pour la génération des résultats. Il s'agit du flux standard de conception des circuits intégrés duquel à été soustraite l'étape du *Design For Test* (DFT). Les étapes du processus vont comme suit :

a. La synthèse logique du code RTL décrit avec un HDL pour générer un réseau de cellules normalisées optimisé ;
b. Le placement-routage du circuit sur puce pour raffiner les estimations parasitiques, insérer des tampons et redimensionner les cellules normalisées ;
c. L'analyse statique de signature, où les délais du circuit sur puce sont estimés ;
d. La simulation post-PNR pour valider, déterminer les performances et générer l'activité de commutation du circuit ;
e. L'analyse dynamique de la puissance pour estimer la puissance moyenne et la puissance crête consommée par le circuit.

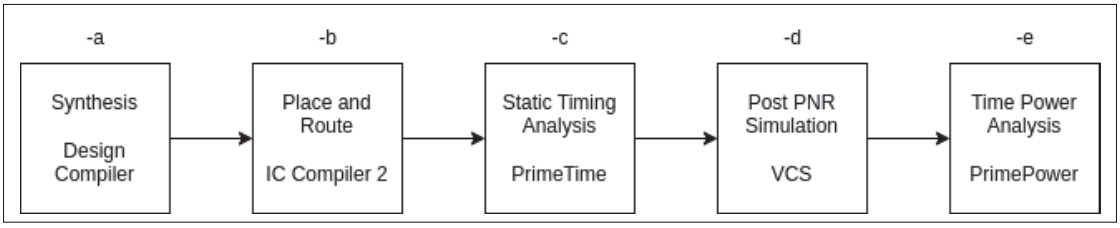


Figure 2.1 Flux de travail utilisé

La suite d'outils utilisée pour le flux de conception ASIC est celle de la compagnie Synopsys. Le choix a été effectué afin de faciliter la cohésion entre les outils. La liste des outils utilisés ainsi que leur version est décrite dans le tableau I-1 situé en annexe. Les étapes du flux de travail sont détaillées individuellement dans les prochaines sous-sections.

2.1.1 Synthèse logique

Pour donner suite à la description du circuit à l'aide d'un HDL, le code est envoyé à l'outil de synthèse DC de Synopsys avec un fichier de contraintes SDC pour la génération d'un circuit optimisé (voir figure 2.2).

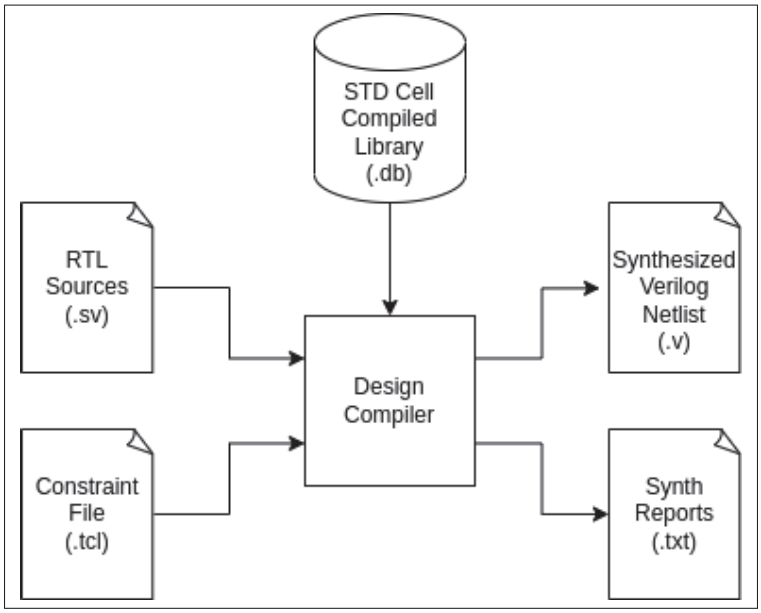


Figure 2.2 Flux de génération des résultats pour la synthèse logique inspiré du manuel d'utilisateur de l'outil DC de Synopsys

À cette étape de conception, le réseau de contrôle est considéré comme étant idéal pour toutes les stratégies de synchronisation, ce qui se traduit par l'utilisation de la contrainte SDC *set_clock_latency* pour modéliser les délais du réseau. Cette modélisation est cruciale dans le cas des stratégies non synchrones, car elle permet la convergence de la synthèse et assure l'entièreté des optimisations sur le circuit.

Cela est principalement dû au fait que les outils de synthèse ont comme objectif principal de minimiser une fonction de coût, cette dernière étant définie par défaut comme la pire marge négative (WNS) du plus long chemin combinatoire. L'algorithme de synthèse consiste à optimiser le pire chemin combinatoire pour qu'il démontre une marge supérieure ou égale à zéro par rapport aux contraintes spécifiées (Synopsys, 2023a).

Dans la mesure où le WNS n'atteint pas une marge supérieure ou égale à zéro pour tous les chemins du circuit, l'étape de synthèse est considérée comme ayant échoué. En effet, l'absence de convergence implique que les efforts d'optimisations étaient focalisés sur la minimisation des délais du circuit, omettant du coup plusieurs étapes d'optimisation au niveau de la surface et au niveau de la puissance.

Trois étapes de synthèse sont itérées pour assurer la convergence du circuit : la lecture des fichiers sources, le *mapping* des cellules normalisées et l'optimisation du circuit. Pour la première étape, chaque fichier source est lu de façon hiérarchique et est traduit en un circuit qui utilise la librairie des portes génériques de l'outil.

Ensuite, pour la deuxième étape, la commande DC *compile* est utilisée, avec un faible effort de compilation, pour transformer les cellules génériques aux cellules de librairie de la technologie utilisée. C'est aussi à cette étape que les contraintes du circuit sont lues, car ces dernières utilisent la nomenclature des broches de la librairie TSMC180 nm.

Finalement, la troisième et dernière étape consiste à utiliser à trois reprises la commande DC *compile_ultra* une première fois sans arguments pour optimiser le réseau de cellules, suivi de deux appels subséquents avec l'argument *-incremental* afin d'améliorer les QOR. Le temps de synthèse totale prend en considération le nombre d'exécutions des trois étapes de synthèse, car chaque itération permet d'obtenir un circuit plus performant.

Pour faire suite à la synthèse, un circuit optimisé de cellules normalisées est généré et accompagné de rapports sur les QOR. Il est à noter que pour les stratégies de synchronisation non synchrones, il est possible que la contrainte de maintien ne soit pas entièrement respectée. En effet, ce

choix a été fait pour qu'elle soit corrigée par l'outil de PNR, car ce dernier est équipé avec les algorithmes nécessaires pour redimensionner les cellules et ajouter des tampons dans le chemin de données pour converger vers un circuit valable.

2.1.2 Placement-routage

Le PNR, effectué avec l'outil ICC2 de Synopsys, consiste en l'implémentation sur puce du réseau de cellules normalisées obtenu suite à la synthèse logique. Il s'agit d'une étape critique, car les résultats sont majoritairement dérivés du circuit qui résulte de l'implémentation physique, et ce pour plusieurs raisons.

Tout d'abord, les outils de PNR sont équipés d'algorithmes d'analyse *multi-mode multi-corner* (MMMC), qui permettent d'optimiser le circuit en fonction des différentes propriétés de chaque coin d'analyse (*design corners*), ce qui permet d'observer l'espace des performances atteignables de façon simultané.

Ces outils sont également équipés avec des algorithmes de redimensionnement pour les cellules et d'ajout de tampons, ce qui permet d'automatiquement réparer les erreurs associées à la contrainte de maintien. Cela est important dans le cadre des stratégies du WP et du PA, car souvent ils se retrouvent dans la situation où le délai minimal n'atteint pas le requis pour cette contrainte.

Enfin, ils possèdent des algorithmes pour implémenter l'arbre de l'horloge, ce qui permet d'ajouter automatiquement des tampons dans la structure de contrôle des stratégies de synchronisation non synchrones.

Quatre étapes distinctes, basées sur le manuel de l'utilisateur (Synopsys, 2023b), sont décrites pour expliquer le flux du PNR dans le cadre du projet. Ces étapes de conception sont accompagnées d'un flux de générations des résultats, présentés dans la figure 2.3.

La première étape est celle du partitionnement de la puce. On y spécifie la taille de la puce, la structure de l'alimentation et la localisation des entrées/sorties. Pour tous les circuits, un cœur

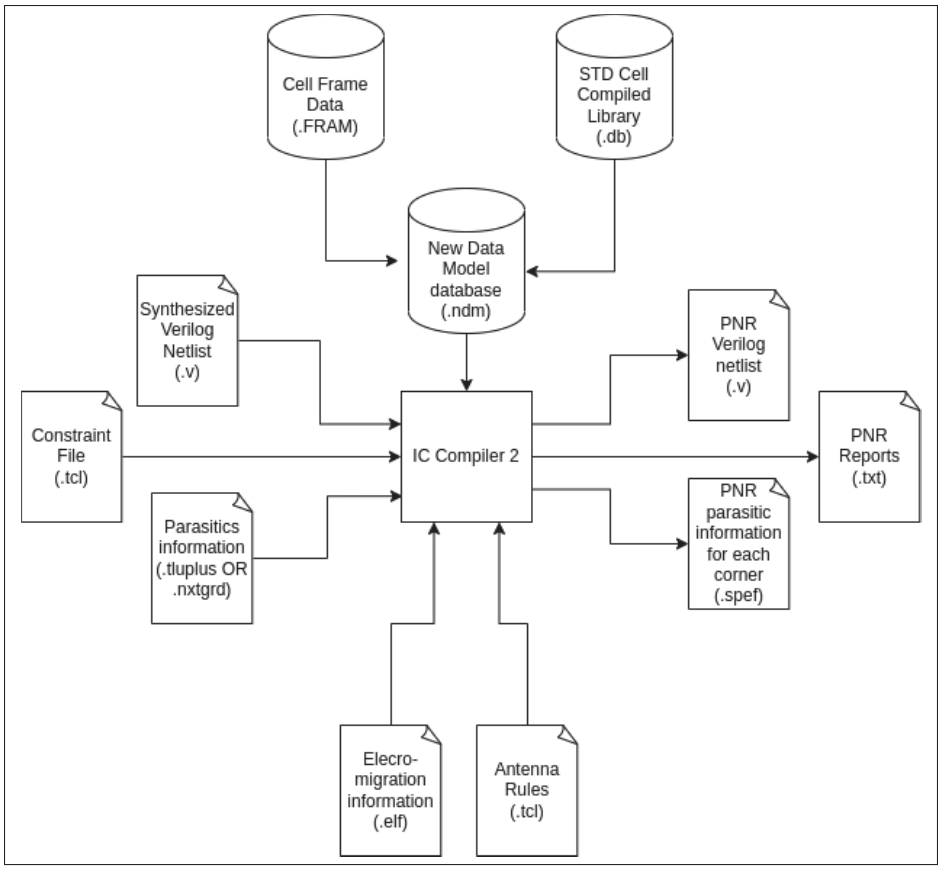


Figure 2.3 Flux de génération des résultats pour le PNR inspiré du manuel d'utilisateur de l'outil ICC2 de Synopsys

en forme de carré est instancié avec un ratio d'utilisation initial de 50%, ce qui permet d'éviter la surutilisation des ressources lors de l'ajout de tampons. Le réseau de l'alimentation est ensuite crée par l'instanciation de deux anneaux pour l'alimentation (VDD) et la masse (VSS), suivi de traces verticales et de rails horizontaux. En dernier, les entrées et les sorties sont distribuées en fonction de leur rôle autour de la puce : les ports de contrôle sont placés à gauche, les sorties du chemin de données sont placées à droite, et les entrées du chemin de données sont distribuées sur les deux côtés restants.

La deuxième étape est celle du placement, où les cellules normalisées sont distribuées à travers la puce en fonction des délais du circuit avec la commande ICC2 *create_placement*. Le placement est ensuite optimisé avec la commande ICC2 *place_opt* avant de passer à la prochaine étape.

La troisième étape est celle de l'implémentation de l'arbre de l'horloge. Cette étape est au cœur de l'implémentation des stratégies de synchronisation, car il s'agit du moment où les délais idéaux spécifiés avec la contrainte SDC *set_clock_latency* sont implémentés avec des tampons. Pour ce faire, la commande ICC2 *set_clock_balance_points* est exploitée avec l'argument *-delay* dans le but de forcer l'ajout de délais asymétriques sur le réseau de l'horloge. Cela permet de répondre aux besoins des contraintes de jumelage entre la donnée et le signal de contrôle pour les stratégies asynchrones. Les spécificités sur la façon dont ces commandes sont utilisées en fonction de chaque stratégie de synchronisation sont expliquées dans la section sur les contraintes.

Finalement, la dernière étape du flux est celle du routage des cellules. Les cellules sont connectées automatiquement entre elles à l'aide de la commande ICC2 *route_auto*, puis un nombre variable d'itérations de l'étape d'optimisation du routage sont lancées avec la commande ICC2 *route_opt*, et ce, jusqu'à la convergence du STA. À chaque itération, un nettoyage est effectué et la latence de l'horloge est recalculée avec la commande ICC2 *compute_clock_latency* pour que l'analyse statique des stratégies non synchrones converge plus rapidement.

Il est à noter que ces trois dernières étapes sont effectuées pour tous les coins d'analyses spécifiés dans le tableau I-2 en annexe.

2.1.3 Analyse statique des délais de signature

Suite au placement et routage, le circuit, combiné aux informations sur ses propriétés parasites, est envoyé dans l'outil de STA PrimeTime (PT) de Synopsys pour calculer les délais de propagation finaux pour chaque coin d'analyse, soit le flux décrit dans la figure 2.4.

Lorsque tous les chemins ont été validés par rapport aux requis spécifiés dans les contraintes, les délais du réseau de cellules normalisés sont exportés dans le fichier *Standard Delay Format* (SDF) pour être utilisés dans la simulation post-PNR.

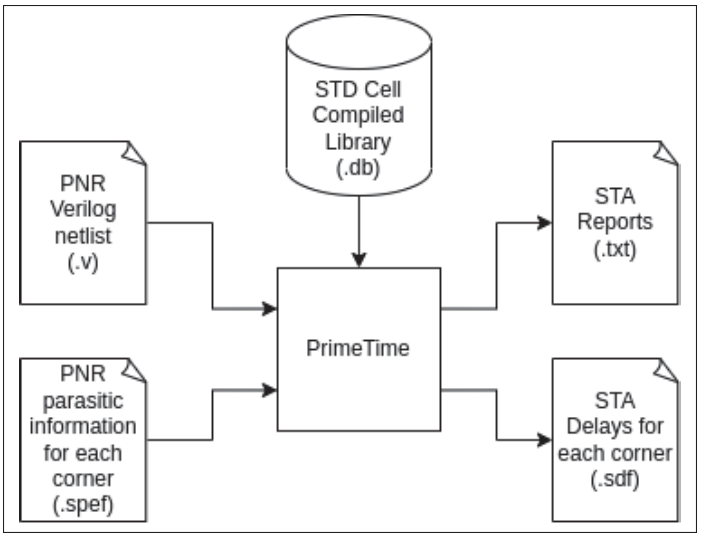


Figure 2.4 Flux de génération des résultats du STA de signature inspiré du manuel d'utilisateur de l'outil PT de Synopsys

Il est à noter que les horloges sont considérées comme étant propagées lors du STA de signature, par l'utilisation de la commande SDC *set_propagated_clock*. Cela permet de prendre en compte les délais induits par les tampons sur le chemin de contrôle lors de la vérification temporelle (Synopsys, 2023c).

2.1.4 Simulation post-PNR

La simulation post-PNR, aussi connue sous le nom de *Gate-Level Simulation* (GLS), est effectuée avec l'outil VCS de Synopsys à l'aide du flux présenté dans la figure 2.5. Elle consiste en la simulation dynamique du circuit physique avec les délais rétroannotés des interconnexions et des cellules provenant du fichier SDF obtenu lors de l'étape du STA de signature. Les résultats qu'elle offre dépendent entièrement des stimuli choisis : c'est pourquoi une stratégie de simulation uniforme a été développée.

Pour toutes les stratégies de synchronisation, les vecteurs en entrée sont lus dans un fichier texte qui contient 10 000 ou 100 000 vecteurs en fonction de la taille du circuit. Ces vecteurs sont

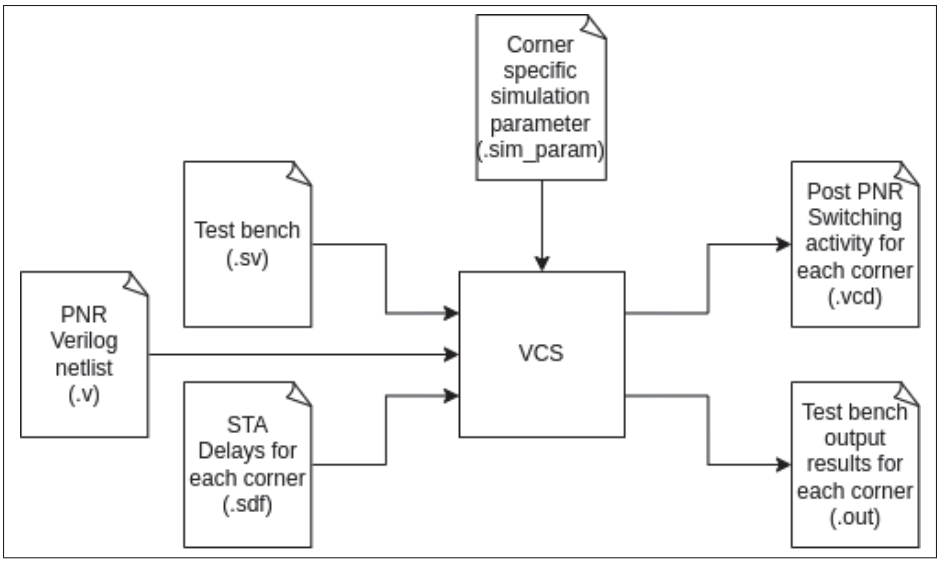


Figure 2.5 Flux de génération des résultats pour la simulation post-PNR inspiré du manuel d'utilisateur de l'outil VCS de Synopsys

générés de façon pseudo-aléatoire par l'utilisation de la semence 3082738027423 et la fonction *random* du langage de programmation Python.

La simulation post-PNR vise à l'atteinte de trois objectifs spécifiques : la validation du circuit, la mesure des performances et la génération de l'activité de commutation.

Le premier objectif vise la validation des fonctionnalités du circuit obtenu suite au PNR. En comparaison avec le STA, qui ne fait que valider le circuit par rapport aux contraintes établies, la simulation post-PNR permet d'observer les réactions dynamiques du circuit face aux stimuli. Dans le cadre des stratégies de synchronisation non synchrones, il s'agit d'un des seuls moments où la structure de contrôle peut être observée pour valider l'allure des signaux, et ainsi garantir la validité du circuit.

Le deuxième objectif vise à mesurer les performances du circuit. Pour les stratégies asynchrones, la complexité du chemin de contrôle implique qu'il est difficile de déterminer ces variables de performance en utilisant uniquement les résultats obtenus du STA de signature. La mesure de

ces deux variables est donc programmée directement dans les bancs de tests selon leur définition respective.

Pour ce qui est du calcul de la période, l'équation 1.8 est utilisée pour calculer la moyenne de temps entre chaque résultat. De son côté, la latence est mesurée par le calcul de la différence de temps où le vecteur injecté est capturé par le registre en entrée et l'apparition du résultat au registre de sortie.

Le dernier objectif vise la génération de l'activité de commutation du circuit, une étape nécessaire afin de permettre une analyse de puissance dynamique. Cela est fait dans le banc d'essai par l'utilisation des directives *SystemVerilog* (SV) *$dumpon* et *$dumpoff*, ces dernières activant et désactivant respectivement l'écriture des évènements dans un fichier *Value Change Dump* (VCD). Pour tous les circuits, l'activation de l'écriture dans le fichier VCD est effectuée immédiatement à la suite de la séquence de réinitialisation.

2.1.5 Analyse de la puissance

La dernière étape du flux de travail est celle de la génération des résultats de la puissance consommée par les circuits à l'aide de l'outil d'analyse de puissance PrimePower (PP) de Synopsys. Cet outil utilise l'équation 2.1 afin d'estimer la puissance moyenne.

$$P_{moy} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot F \tag{2.1}$$

Les variables qu'on y retrouve sont l'activité de commutation α, la charge capacitive C_L, la tension d'alimentation V_{DD} et la fréquence de commutation F. Deux méthodes sont répertoriées pour déterminer la valeur de l'activité de commutation α : la méthode statique et la méthode dynamique (Synopsys, 2024).

La méthode statique utilise un modèle probabiliste afin d'estimer l'activité de commutation α sur les nœuds du circuit. Elle a l'avantage d'être rapide, mais peut donner des résultats imprécis dans la mesure où le concepteur n'indique pas les bonnes prédictions, d'où son utilisation dans les

phases initiales d'un projet. Elle cause aussi un problème quant à l'estimation de la puissance du circuit de contrôle des circuits asynchrones, ces derniers suivant une séquence non probabiliste.

En comparaison, la méthode dynamique utilise un fichier qui contient l'activité de commutation des nœuds d'un réseau de cellules normalisées provenant d'une simulation logique pour dynamiquement adapter $alpha$. Cela la rend plus longue à exécuter par rapport à son équivalent statique, mais elle permet d'obtenir des résultats plus précis pour un mode d'opération voulu. Étant donné l'utilisation de vecteurs générés de façon pseudo-aléatoire en entrée, le mode d'opération simulé est celui du cas typique.

Cette méthode est préférable dans le cas de la simulation de différentes stratégies de synchronisation, car elle permet d'obtenir des résultats fiables plus aisément, d'où son utilisation pour les résultats de la puissance moyenne et la puissance crête. Le flux utilisé pour la génération des données avec la méthode dynamique est présenté dans la figure 2.6.

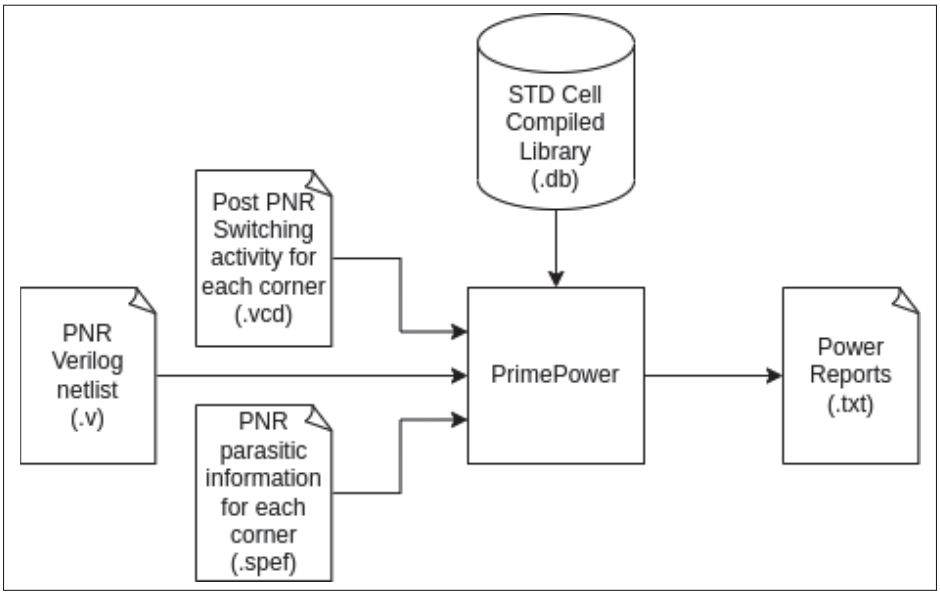


Figure 2.6 Flux de génération des résultats pour l'analyse dynamique de puissance inspiré du manuel d'utilisateur de l'outil PP de Synopsys

2.2 Contraintes

Cette section traite de la façon dont les stratégies de synchronisation sont implémentées avec les outils EDA, ainsi que les contraintes SDC utilisées afin de les modéliser. Les contraintes sont présentées pour la stratégie synchrone avec et sans recadencement, pour la stratégie asynchrone BD, pour la stratégie du WP ainsi que pour la stratégie PA avec emprunt d'horloge.

2.2.1 Stratégie synchrone

Les contraintes utilisées pour modéliser la stratégie synchrone, telles que présentées dans la figure 2.7, sont celles que l'on retrouve typiquement pour un circuit simple. On y retrouve la définition de l'horloge avec la contrainte SDC *create_clock*, la définition du délai de propagation qui précède les entrées avec la contrainte SDC *set_input_delay* ainsi que la définition du délai qui suit les sorties avec la contrainte SDC *set_output_delay*.

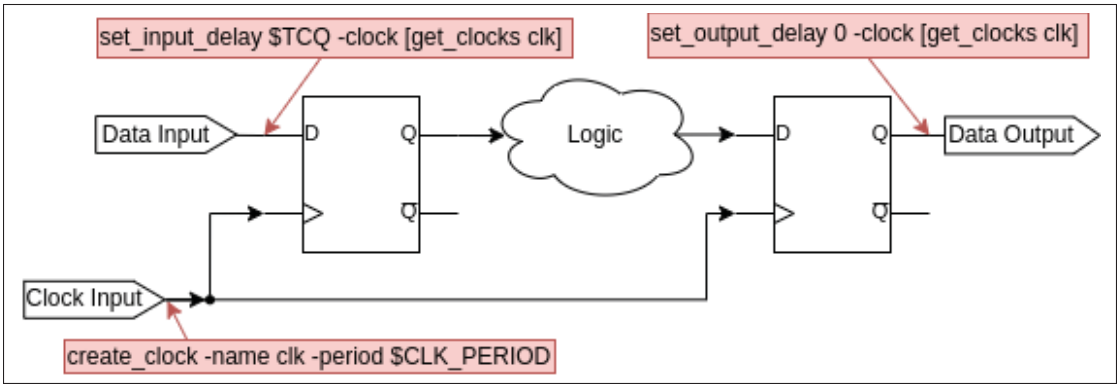


Figure 2.7 Contrainte SDC pour la stratégie synchrone

Le port d'entrée est modélisé de façon à ce que la donnée provienne d'une bascule par l'application d'un délai de propagation externe de T_{CQ}. En contrepartie, pour le port de sortie, un délai nul est appliqué pour modéliser l'absence de logique qui suit le circuit. Comme les circuits évalués sont considérés comme étant idéaux, aucune modélisation de l'instabilité de phase de l'horloge n'est effectuée avec la contrainte SDC *set_clock_uncertainty*.

2.2.1.1 Recadencement

En plus des contraintes synchrones, des indications spécifiques, exposées dans la figure 2.8, sont ajoutées lors de la synthèse logique avec DC pour activer la fonction de recadencement des registres. La fonctionnalité est activée par l'utilisation de la commande DC *set_optimize_registers*, ce qui rend tous les registres du circuit déplaçables lors de l'exécution de la compilation avec la commande DC *compile_ultra* (Synopsys, 2023a).

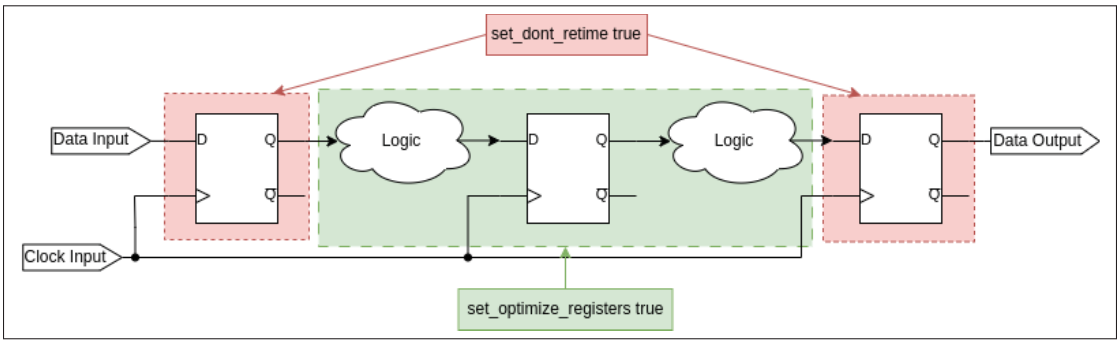


Figure 2.8 Contraintes de syntèse pour le recadencement

Afin d'empêcher l'outil de synthèse de déplacer la logique combinatoire en amont et en aval des registres d'entrées et de sorties, ce qui rendrait la comparaison inéquitable avec les autres stratégies de synchronisation, ces registres sont exclus du recadencement avec la commande DC *set_dont_retime*.

2.2.2 Stratégie asynchrone

Le lot de RTC génériques défini par Gimenez *et al.* (2019) est utilisé pour contraindre les circuits asynchrones BD. Parmi ce lot, deux RTC sont implémentés pour permettre un fonctionnement minimal du circuit. Les symboles utilisés pour représenter les points de convergence de l'équation 1.7, qui définissant la relation de préséance d'une RTC, sont repris de l'article.

On y retrouve les préfixes *d-* et *c-* pour décrire s'il est question d'un élément séquentiel du chemin de données ou du chemin de contrôle ainsi que les suffixes *data* et *ctrl* pour indiquer

s'il est question de la broche de donnée ou de contrôle de l'élément séquentiel. Par ailleurs, la marge de temps m, représentée par le terme δ, est soustraite du chemin long.

La première RTC utilisée est celle de jumelage, définie par l'équation 2.2.

$$pod \mapsto d\text{-}data_i \prec d\text{-}ctrl_i - \delta \tag{2.2}$$

Elle nous informe du fait que la donnée appliquée au port d'entrée d'un élément séquentiel d'un étage i du chemin de données doit arriver un certain temps δ avant le signal de contrôle appliqué à ce même élément séquentiel afin de garantir que la donnée est correctement capturé. Il s'agit de la contrainte équivalente de la contrainte de préparation pour une synchronisation synchrone. L'équation est représentée graphiquement par la figure 2.9.

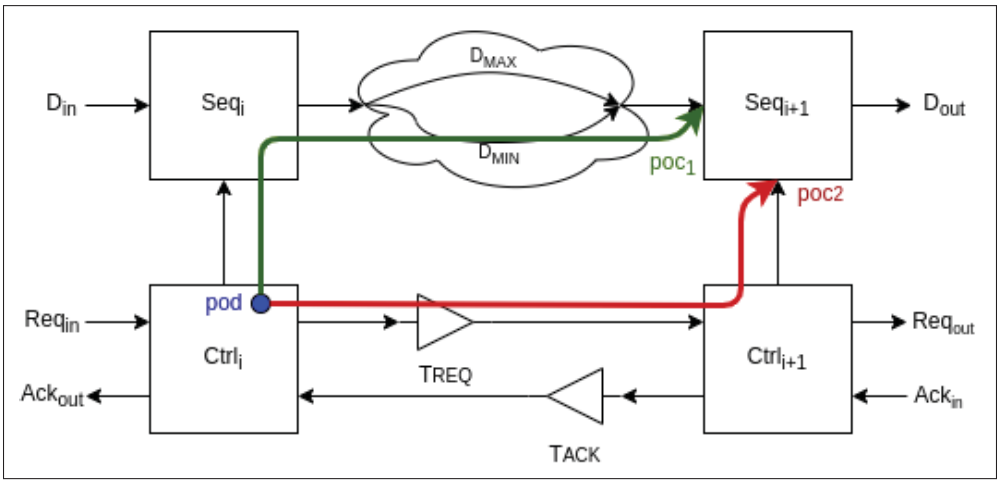


Figure 2.9 Représentation de la RTC de jumelage

La deuxième RTC est celle de l'écrasement de données, définie par l'équation 2.3

$$pod \mapsto d\text{-}ctrl_i \prec d\text{-}data_i - \delta \tag{2.3}$$

Elle indique que la capture de la donnée sur l'élément séquentiel de l'étage i par l'activation de son port de contrôle doit être faite un certain temps δ avant qu'une nouvelle donnée soit

présentée à son port de données. Il s'agit de l'équivalent de la contrainte de maintien pour une synchronisation synchrone. L'équation est représentée graphiquement par la figure 2.10.

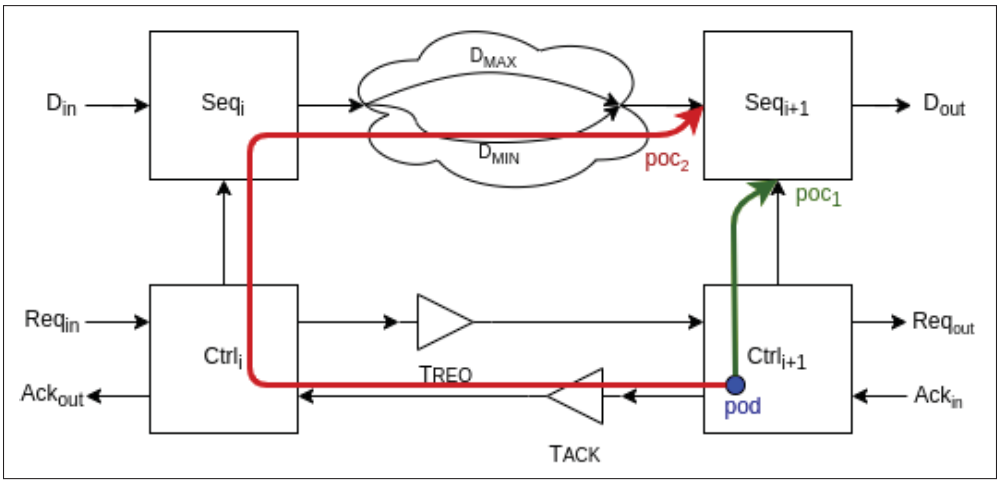


Figure 2.10 Représentation de la RTC d'écrasement de donnée

Les deux contraintes sont implémentées à l'aide de la méthode LCS de Gimenez *et al.* (2018), ce qui permet l'analyse statique avec les outils EDA synchrones. Les figures 2.11 et 2.12 schématisent les différentes séquences de commandes utilisées pour implémenter la contrainte de jumelage et la contrainte d'écrasement de données respectivement.

Il est à noter que ces séquences de commandes sont décrites plus en détail en naviguant le dépôt Git du code source de la méthode LCS (Gimenez, 2025). De plus, les figures représentent uniquement la définition des lots d'horloges pour la phase à transition montante du signal de requête et d'acquiescement. Un lot équivalent d'horloges doit être développé pour le chemin pris par la phase du front descendant.

Des exceptions supplémentaires à ces définitions d'horloges sont indiquées avec les contraintes SDC *set_false_path* et *set_clock_groups -asynchronous*. L'objectif de ces exceptions est de forcer indirectement l'analyse entre l'horloge racine et l'horloge de capture en ignorant les chemins débutants ou arrivant aux horloges transitoires.

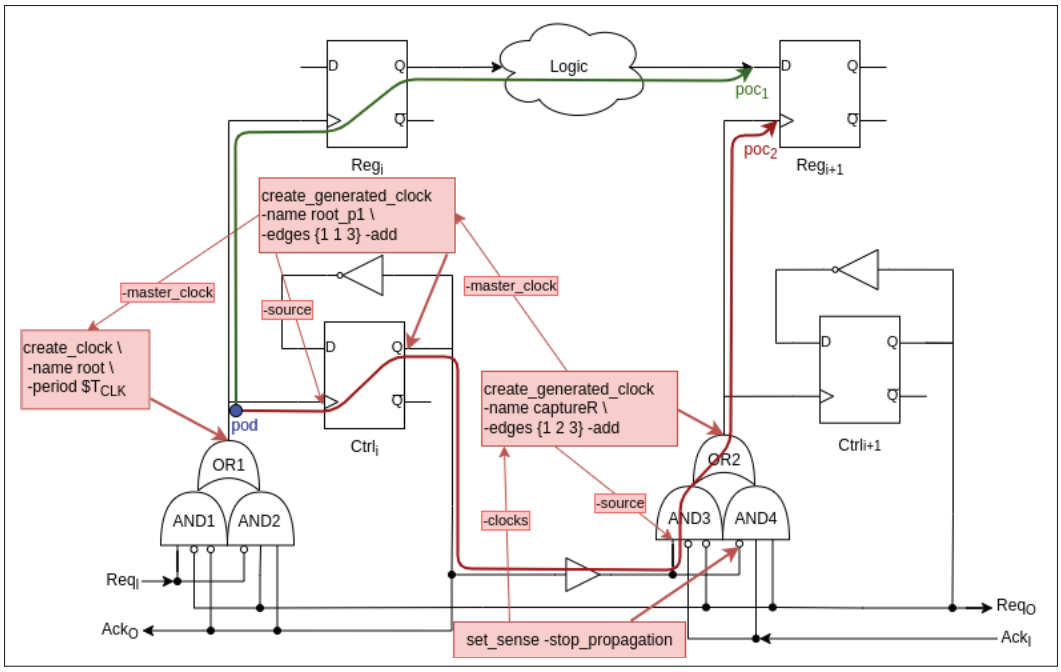


Figure 2.11 Contraintes SDC pour la contrainte de capture pour la stratégie asynchrone

Bien que la méthode soit fonctionnelle pour effectuer l'étape du STA sur des circuits asynchrones BD, quelques modifications y ont été apportées. La première est une mise à jour pour son utilisation avec l'outil ICC2, étant donné son développement dédié à l'outil ICC. La deuxième est une correction proposée pour correctement contraindre le chemin de données, ce qui augmente la qualité des résultats suite à la synthèse. La dernière est une extension proposée pour permettre l'analyse statique des entrées et des sorties, une fonction non décrite dans la stratégie originale.

La première modification porte sur la mise à jour de certaines commandes pour permettre l'implémentation automatique de tampons avec l'outil ICC2 de Synopsys. Dans sa thèse de doctorat, Gimenez (2021) proposé d'une méthode pour implémenter automatiquement des lignes à retard dans l'outil ICC de Synopsys en utilisant la commande ICC *set_clock_tree_exception*.

Cependant, comme cette commande ne se retrouve pas dans la deuxième version de l'outil, il a fallu développer une procédure équivalente, cette dernière illustrée dans la figure 2.13. Elle se résume au remplacement de la commande ICC *set_clock_tree_exception* par la commande ICC2

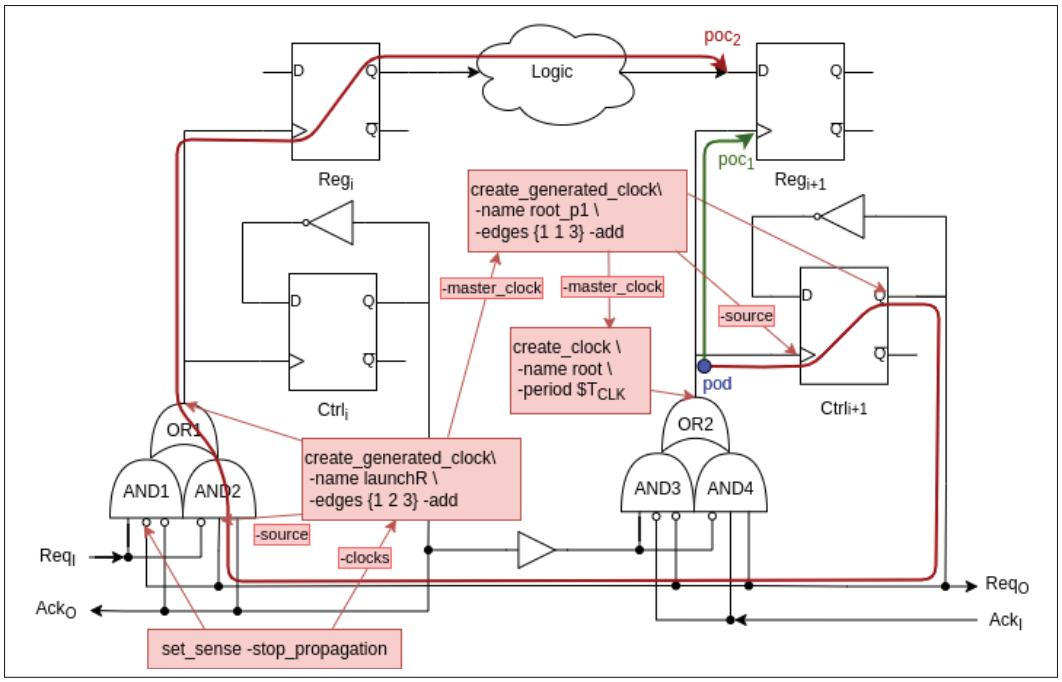


Figure 2.12 Contraintes SDC de la contrainte de lancement pour la stratégie asynchrone

set_clock_balance_points ainsi qu'à l'ajout de contraintes *set_dont_touch* pour éviter l'insertion de tampons sur les mauvais nœuds.

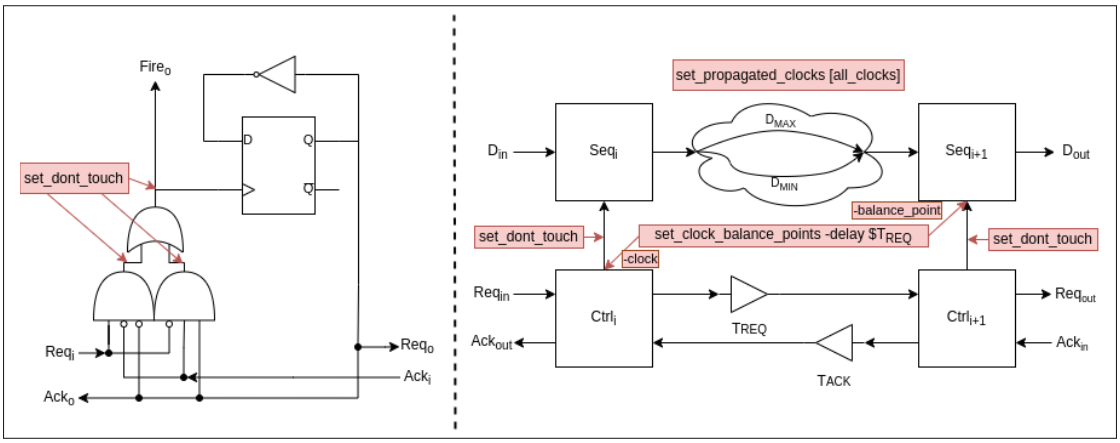


Figure 2.13 Contraintes pour l'insertion de tampons dans ICC2 avec la méthode LCS

On y trouve aussi l'utilisation de la commande *set_propagated_clocks*, cette dernière permettant à l'outil de propager les horloges pour se réguler dans l'étape de l'ajout des tampons. Rappelons que cette première modification n'est pas considérée comme une amélioration de la méthode LCS, puisqu'il s'agit essentiellement d'une mise à jour. De ce fait, les résultats obtenus avec cette modification seront considérés comme similaires à ceux obtenus à partir de la méthode originale (de base). Par contre, les deux modifications suivantes sont, elles, considérées comme des contributions de ce mémoire, car elles améliorent la méthode LCS originale.

La deuxième modification consiste en la correction de la méthode contraignant le délai combinatoire lors de la synthèse. La méthode LCS propose une stratégie qui utilise la période des horloges racines créées avec la contrainte SDC *create_clock* pour contraindre les délais du chemin combinatoire. Cependant, il s'avère que cette stratégie n'est valide que si les périodes choisies se trouvent être des multiples les unes des autres, ce qui rend son utilisation très limitée.

Étant donné l'utilisation de la méthode WNS par l'outil de synthèse pour déterminer le chemin à optimiser, l'absence de contraintes, quant au délai attendu de la logique combinatoire, pousse l'outil à optimiser tous les chemins du circuit en vitesse sans jamais converger. Cette priorisation dans l'optimisation des délais cause d'importantes déficiences quant à l'optimisation de la surface utilisée et de la puissance consommée par l'outil.

L'amélioration proposée consiste à ajouter l'utilisation de la contrainte SDC *set_clock_latency* sur les horloges de capture de la méthode LCS avec le délai combinatoire voulu lors de l'étape de la synthèse. L'utilisation de cette contrainte est combinée avec l'utilisation de la commande DC *group_path* pour limiter les chemins à optimiser entre les horloges racines et les horloges de captures. Cela est dû au fait que, pour chaque horloge créée avec la méthode, un nouveau groupe est inféré par l'outil de synthèse (Synopsys, 2023a). Il est donc important de les supprimer avec la commande DC *remove_path_group* pour limiter la quantité de chemins à valider.

La dernière modification étend les capacités d'analyse statique de la méthode LCS afin d'inclure l'analyse des ports d'entrée ainsi que les ports de sortie. Elle est présentée dans la figure 2.14.

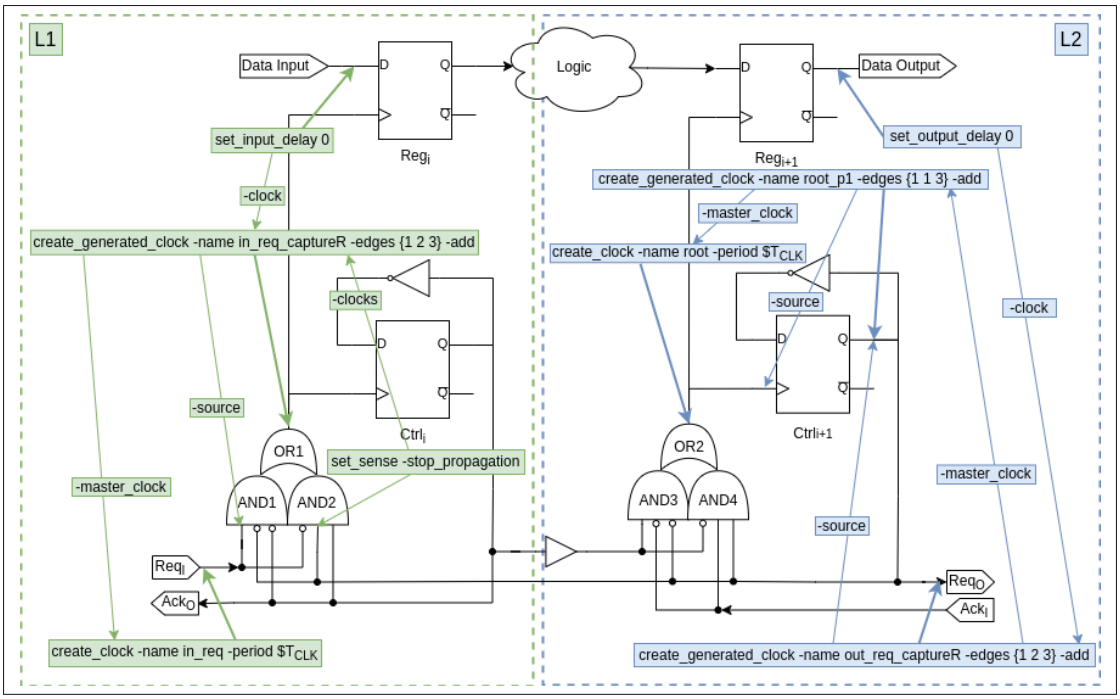


Figure 2.14 Contrainte SDC pour les entrées (gauche) et les sorties (droite) utilisant la méthode LCS

Pour les ports d'entrée, une horloge racine est créée avec la contrainte SDC *create_clock* sur le port de contrôle en entrée, dans ce cas-ci le port de requête Req_i, puis elle est propagée avec l'utilisation d'horloges générées jusqu'à l'atteinte du registre du chemin de données, telle que décrite par la méthode originale. La commande *set_input_delay* est ensuite appliquée avec un délai de zéro aux ports d'entrée du chemin de donnée, ce qui indique qu'elle est jumelée avec le signal de contrôle. La commande est référencée à l'horloge générée qui s'attaque à la broche de contrôle du registre avec l'argument *-clock*, ce qui active la comparaison entre le chemin $DATA_{IN} \rightarrow D\text{-}Reg/D$ et le chemin $Req_i \rightarrow D\text{-}Reg/CLK$.

Pour les ports de sortie, une horloge générée est créée sur le port de contrôle en sortie, dans ce cas-ci Req_O, puis elle est reliée avec l'horloge générée la plus proche afin de la connecter à l'horloge racine qui contrôle le registre en sortie. Un délai externe de zéro est appliqué au port de sortie du chemin de donnée avec la contrainte SDC *set_output_delay* pour indiquer un jumelage entre la donnée et le contrôle. La commande est associée à l'horloge générée sur le

port de sortie pour activer l'analyse entre le chemin $D\text{-}Reg/CLK \rightarrow DATA_{OUT}$ et le chemin $C\text{-}Reg/CLK \rightarrow Req_O$.

Contrairement aux contraintes d'entrée, on retrouve une petite distinction lorsqu'il est question de l'analyse du front descendant du signal de contrôle. En effet, les arguments *-clock_fall* et *-add_delay* doivent être ajoutés à la commande *set_output_delay* pour que le calcul des délais soit fait par rapport à un front descendant du signal de contrôle.

2.2.3 Stratégie du *Wave Pipelining*

Dans cette section, nous présentons les contraintes développées dans le cadre de ce projet pour la stratégie du WP. Comme mentionné, des stratégies sont disponibles pour la synthèse du WP, cependant, au meilleur de nos connaissances, aucun ensemble de contraintes n'est disponible dans la littérature scientifique.

L'objectif principal de la modélisation du WP se situe au niveau de la traduction des équations 1.12 et 1.13 en contraintes SDC de façon à permettre l'analyse par l'outil de STA. Pour ce faire, trois lots de contraintes, illustrés dans la figure 2.15, ont été développés.

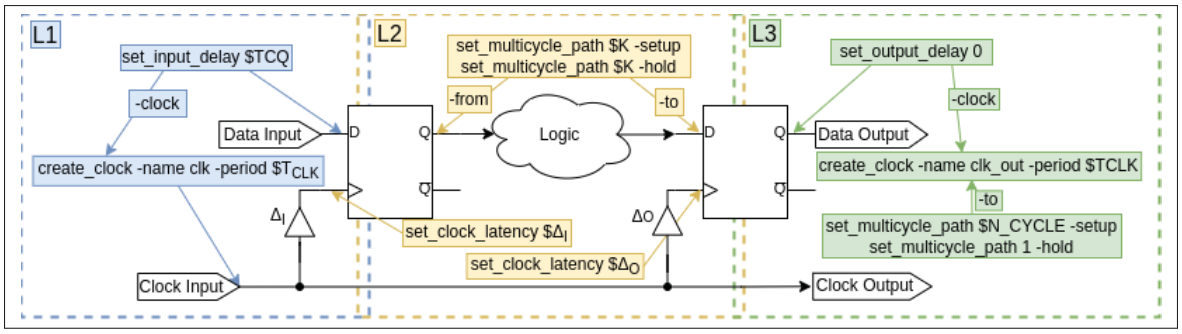


Figure 2.15 Contrainte SDC pour la stratégie WP

Le premier lot, $L1$, modélise une interface synchrone à l'entrée du circuit. Un délai externe de T_{CQ} est appliqué au port de données en entrée avec la commande SDC *set_input_delay* pour indiquer que les données proviennent d'un registre. Ce délai est ensuite référencé à l'horloge

principale du circuit avec l'argument *-clock*, cette dernière ayant été créée à l'aide de la contrainte SDC *create_clock*.

Le deuxième lot, $L2$, modélise l'équation 1.11 du délai de propagation pour compenser la logique interne du circuit. Dans un premier temps, la contrainte *set_multicycle_path* est utilisée pour modéliser le terme $K \cdot T_{CLK}$ de l'équation 1.11 pour les contraintes de préparation et de maintien. En contrepartie, le terme du délai asymétrique Δ de l'équation 1.11 est modélisé par l'utilisation de la commande *set_clock_latency* sur la broche d'horloge de la bascule d'entrée pour définir Δ_I et sur la broche d'horloge de la bascule de sortie pour définir Δ_O.

Normalement, dans une synchronisation synchrone, la contrainte de capture d'un chemin multicycle est vérifiée au cycle K, tandis que la contrainte de maintien est vérifiée au cycle $K - 1$, ce qui est fait automatiquement lorsque la commande *set_multicycle_path* est utilisée sans arguments. Cependant, comme la vérification des deux contraintes se fait au même front d'horloge dans une synchronisation WP, tel qu'observé dans la figure 1.18, les arguments *-setup* et *-hold* doivent être explicités avec la même valeur de K.

Le dernier lot, $L3$, modélise une interface synchrone qui capture la donnée en sortie. Pour ce faire, une horloge virtuelle, nommée *clk_out*, est créée afin de modéliser le signal d'horloge de capture. L'instant où l'horloge capture la donnée en sortie est ajusté pour être un nombre de cycles défini par l'équation 2.4 avec la contrainte SDC *set_multicycle_path* pour la contrainte de préparation.

$$N\_CYCLE = K + \left\lceil \frac{\Delta + T_{CQ} + T_{SU}}{T_{CLK}} \right\rceil \tag{2.4}$$

En contrepartie, pour la contrainte de maintien, l'analyse doit se faire de T_{CLK} à $(N\_CYCLE - 1) \cdot T_{CLK}$, ce qui est impossible à programmer dans l'outil STA. Ce problème à été solutionné en modélisant la forme équivalente de 0 à $(N\_CYCLE - 2) \cdot T_{CLK}$ en posant le nombre de cycles de la contrainte SDC *set_multicycle_path -hold* à 1. Le port en sortie est ensuite associé à l'horloge avec la contrainte SDC *set_output_delay* via l'argument *-clock* et un délai de zéro pour activer l'analyse vers la sortie.

Lors de l'implémentation physique, afin d'éviter l'insertion de délai entre le port d'entrée et de sortie de l'horloge, la commande ICC2 *set_clock_balance_points* est utilisée pour spécifier un délai d'insertion nul au port de sortie de l'horloge.

2.2.4 Stratégie pseudo-asynchrone avec emprunt d'horloge

Cette section présente la deuxième version de l'ensemble des contraintes développées en collaboration avec Michel Kafrouni pour la synchronisation pseudo-asynchrone avec emprunt d'horloge. Elle propose quatre lots de contraintes SDC, présentés dans la figure 2.16, pour implémenter la stratégie dans les outils EDA synchrones, éliminant ainsi les limitations et les complexités observées lors de l'utilisation de la méthode originale développée par M. Kafrouni.

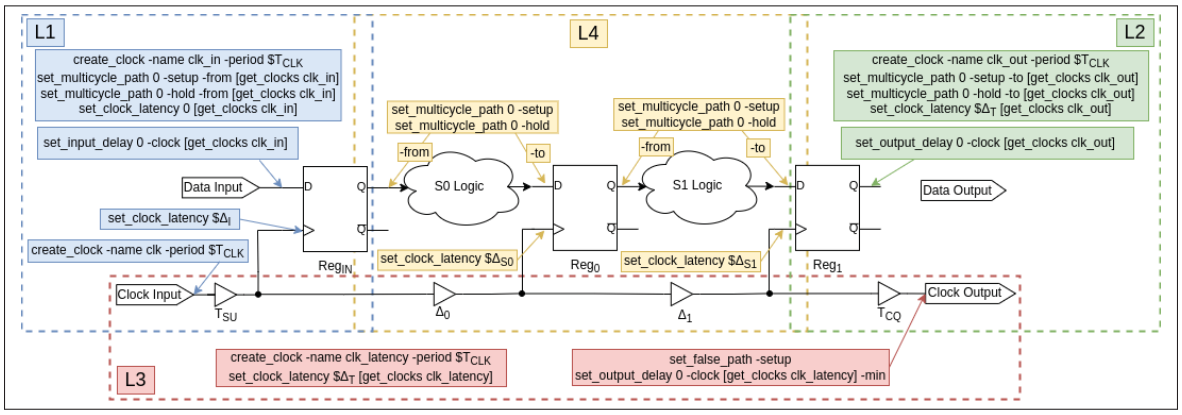


Figure 2.16 Contraintes de la stratégie pseudo-asynchrones avec emprunt d'horloge

Le premier lot de contraintes, $L1$, modélise un port de données en entrée jumelé à un signal de contrôle, soit l'horloge dans ce cas. Dans un premier temps, une horloge virtuelle *clk_in* est créée avec la contrainte SDC *create_clock* pour séparer la latence de lancement et la latence de capture des données. La contrainte SDC *set_multicycle_path* est ensuite utilisée avec un argument de zéro pour indiquer que les données lancées par l'horloge virtuelle *clk_in* sont capturées sur le même front que celui de l'horloge affectant les registres d'entrée. Enfin, un délai externe de zéro est appliqué au port d'entrée avec la contrainte SDC *set_input_delay* et est référencé à l'horloge

virtuelle avec l'argument *-clock*, indiquant que l'horloge et la donnée sont lancées en même temps.

Pour assurer la convergence du STA en synthèse, un délai de $\Delta_I = T_{SU}$ est appliqué au port de contrôle de la bascule en entrée avec la contrainte SDC *set_clock_latency*. Ce délai idéal est retiré lors de l'étape du STA de signature, car il est mesuré avec la propagation des horloges au moyen de la contrainte SDC *set_propagated_clocks*.

Le deuxième lot de contraintes, $L2$, modélise un port de données en sortie jumelé à un signal de contrôle. Au départ, une horloge virtuelle *clk_out* est créée avec la contrainte SDC *create_clock* pour modéliser l'horloge capturant la donnée en sortie. Tout comme en entrée, la contrainte SDC *set_multicycle_path* est utilisée pour indiquer que les signaux sont capturés par l'horloge virtuelle sur le même front que celle de l'horloge de lancement. Un délai de zéro est spécifié sur le port en sortie avec la contrainte SDC *set_output_delay* et est associé à l'horloge virtuelle avec l'argument *-clock*, ce qui modélise le jumelage entre la donnée et le contrôle.

Pour assurer une convergence du STA en synthèse, la latence totale du circuit−soit la somme du temps de préparation, du délai combinatoire et du délai de propagation de la bascule en sortie ($\Delta_T = T_{SU} + N_{RP} \cdot T_{CLK} + T_{CQ}$)−est appliquée à l'horloge virtuelle avec la contrainte SDC *set_clock_latency* pour compenser le délai de propagation total du circuit.

Notons que, tout comme pour le port d'entrée, les délais idéaux indiqués par la commande *set_clock_latency* sont retirés lors du STA de signature. De plus, l'horloge virtuelle *clk_out* est convertie en horloge générée avec la commande *create_generated_clock* sur le port de sortie du signal d'horloge, permettant ainsi de mesurer le délai de la ligne à retard à la propagation des horloges avec la contrainte SDC *set_propagated_clock*.

Le troisième lot de contraintes, $L3$, propose une utilisation contre-intuitive des contraintes SDC dans le but de guider l'outil de PNR lors de l'implémentation de la ligne à retard. Pour ce faire, une horloge virtuelle nommée *clk_latency* est créée afin de séparer ses propriétés de l'horloge virtuelle *clk_out*. Cette horloge est associée au port de sortie de l'horloge avec la commande

set_output_delay avec un délai de zéro et avec l'argument *-min* pour indiquer que l'analyse se fait uniquement pour chemin du délai minimal.

L'analyse du chemin avec le délai maximal est désactivée, et ce pour deux raisons. La première raison est pour éviter que la méthode du WNS utilise ce chemin lors de l'implémentation du circuit en synthèse. La deuxième raison est parce que, contrairement à l'analyse du délai minimal, l'outil d'analyse statique ajoute une demi-période d'horloge fantôme dans le rapport d'analyse statique du chemin maximal, empêchant la convergence des délais.

La contrainte SDC *set_clock_latency* est appliquée sur l'horloge afin d'indiquer que le délai total de la ligne à retard doit être de Δ_T. Notons que, suite à l'implémentation par l'outil de PNR, les délais ajoutés peuvent être observés en exécutant la commande ICC2 *report_timing -delay_type min -to [get_clocks clk_latency]*. Aussi, lors du STA de signature, l'horloge *clk_latency* n'est plus nécessaire, car le délai total peut être mesuré avec l'horloge générée *clk_out* placée au port de sortie de l'horloge. Il est donc recommandé de la supprimer.

Le dernier lot de contraintes, $L4$, décrit la relation interne entre les étages de registres. La contrainte SDC *set_multicycle_path* est utilisée avec un argument de 0 pour indiquer que la capture et le lancement des signaux du chemin de données sont effectués sur le même cycle d'horloge. La contrainte SDC *set_clock_latency* est ensuite utilisée pour définir le délai de l'étage en fonction de l'équation $\Delta_n = T_{CLK} + B_n$, où T_{CLK} est la période d'horloge et B_n est la marge de temps empruntée ou donnée à un autre étage de pipeline.

L'algorithme 2.1 décrit la méthode utilisée dans le fichier de contrainte pour calculer les délais appropriés de chaque étage. La variable Δ_S est utilisée comme accumulateur afin de modéliser l'accumulation des délais Δ_n de la ligne à retard pour chaque nouvel étage du pipeline. Aussi, la variable B_n représente les valeurs emprunté ou donné pour chaque étages, provenant d'un tableau B.

Algorithme 2.1 Algorithme du calcul du délai compensatoire pour chaque étage de la stratégie PA avec emprunt d'horloge

```
# Déclarer un tableau contenant la marge emprunté ou donné pour chaque étages
B = {B_0, B_1, ..., B_{N_REG-1}}
# Application de la latence au registre d'entrée
Δ_S ← T_SU
set_clock_latency Δ_S Reg_I/CLK
# Algorithme calculant la latence des étages en fonction de l'emprunt
for all pipeline stages n ∈ N_Reg do
    Δ_S ← Δ_S + T_CLK + B_n
    set_clock_latency Δ_S Reg_n/CLK
end for
# Application de la latence au port d'horloge en sortie
Δ_S ← Δ_S + T_CQ
set_clock_latency Δ_S clock_output
```

2.3 Multiplicateur-accumulateur

Comme mentionné, dans l'optique de concentrer les efforts sur l'implémentation des différentes stratégies de synchronisation, une architecture RTL du MAC a été développée pour traiter des nombres entiers signés encodés en complément à deux. Ces deux choix ont été effectués pour éviter les pertes d'efficacités liées à une architecture structurelle et éviter les complexités d'un circuit traitant des nombres à point flottant.

Globalement, le circuit du MAC possède deux entrées de N bits, *a_i* et *b_i* et une sortie *y_o* qui provient d'un accumulateur de C bits. L'architecture est décrite avec le langage SV qui utilise une description RTL haut niveau par l'utilisation des opérateurs "$*$" et "+". Le signal de réinitialisation est synchronisé de façon asynchrone pour toutes les stratégies afin d'éviter une augmentation de la surface utilisée pour la stratégie synchrone.

Au niveau de la synchronisation, il possède quelques particularités qui ajoutent un niveau de complexité quant à l'implémentation du circuit de contrôle. La première particularité est qu'il y a une opération pour joindre les deux vecteurs appliqués aux entrées du circuit de multiplication. La deuxième particularité est qu'il y a une boucle de rétroaction pour le circuit d'accumulation.

Les prochaines sous-sections discuteront du design de chaque stratégie accompagné d'explications complémentaires sur les particularités de leur implémentation.

2.3.1 Stratégie synchrone

La figure 2.17 montre le circuit conçu pour les fins du projet pour la synchronisation synchrone. Quatre registres sont utilisés pour découper la logique combinatoire : deux registres pour synchroniser les vecteurs a_i et b_i en entrée, un registre pour séparer la logique combinatoire entre le circuit de multiplication et le circuit d'addition, puis un registre pour synchroniser la sortie y_o.

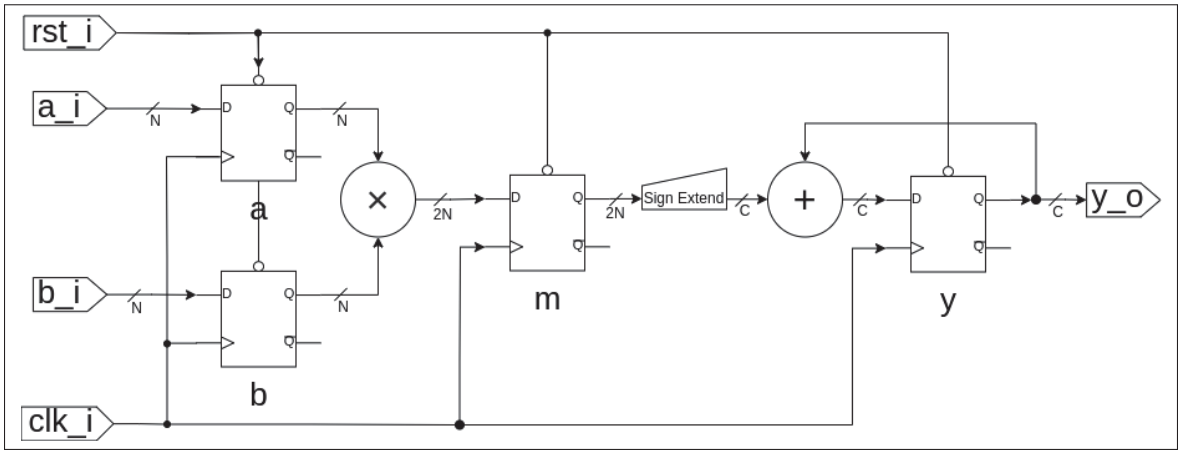


Figure 2.17 Schéma du MAC synchrone

Tous les registres sont synchronisés avec le même signal de contrôle, soit une horloge globale. Le circuit est sous sa forme optimisée par le fait que l'accès à la structure interne des opérations arithmétiques est réservé à l'outil de synthèse.

Une légère modification au circuit du MAC synchrone de la figure 2.17 était nécessaire pour le MAC synchrone avec recadencement. Comme observé dans la figure 2.18, cette modification consiste à permettre l'instanciation d'un nombre variable de registres intermédiaires entre le multiplicateur et l'accumulateur.

Puisque l'outil de synthèse est en mesure de déplacer ces registres internes à l'intérieur des structures arithmétiques afin d'égaliser le délai combinatoire, une quantité supérieure de registres internes peut être instanciée afin d'atteindre des performances similaires aux autres stratégies de synchronisation.

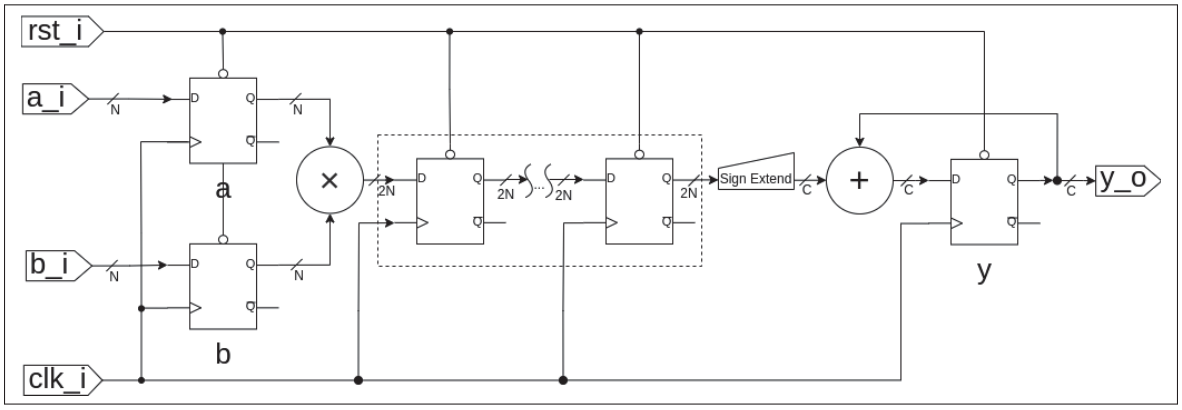


Figure 2.18 Schéma du MAC synchrone avec recadencement

2.3.2 Stratégie asynchrone

La figure 2.19 présente l'implémentation du circuit asynchrone BD avec, comme contrôleurs, des CE. Elle utilise la même structure que la stratégie synchrone pour diviser l'étape de multiplication et d'addition, car la période équivalente dépend, entre autres, du plus long chemin combinatoire.

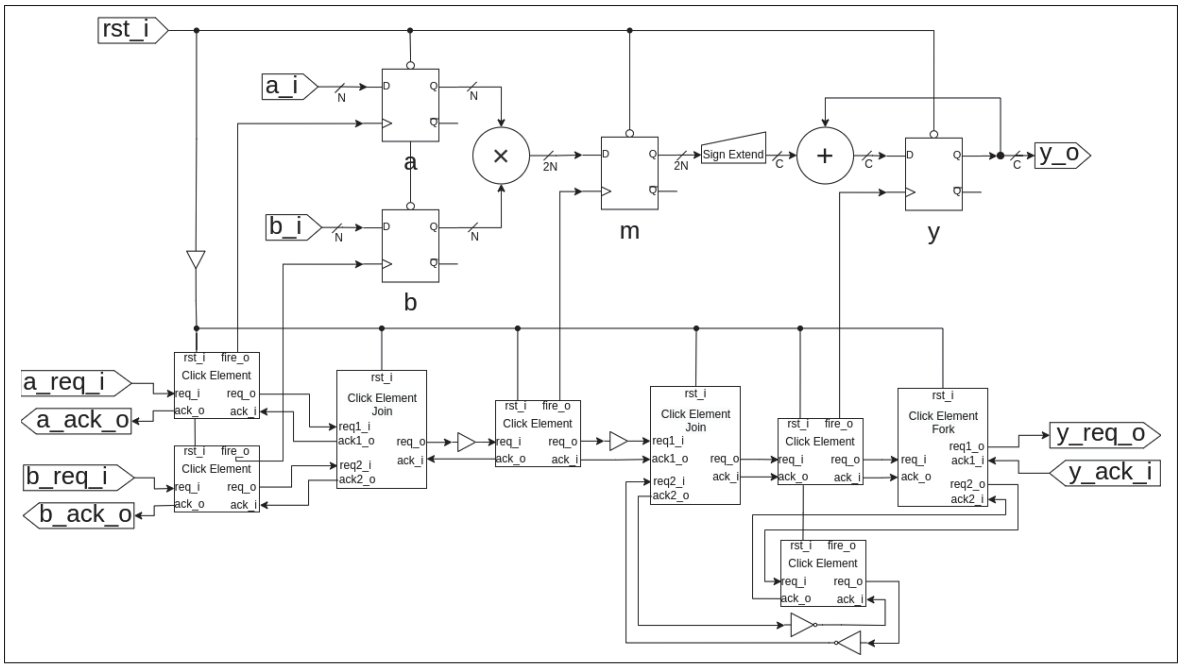


Figure 2.19 Schéma du MAC asynchrone BD avec CE

Pour être conforme aux spécifications de la stratégie, chaque registre du chemin de données est synchronisé avec un contrôleur CE. Le flux de données est ensuite imité dans le flux de contrôle par l'ajout d'opérateurs de fourche et d'union.

Une particularité du circuit se trouve au niveau de la structure de contrôle de la boucle de rétroaction. Dans sa thèse de doctorat, Gimenez (2021) explique la nécessité d'implémenter une boucle en utilisant l'opération de fourche, un contrôleur factice et une opération d'union pour garantir la vivacité du circuit. Cela explique donc la nécessité de la structure de contrôle implémentée pour le circuit de l'accumulateur.

2.3.3 Stratégie du *Wave Pipelining*

La figure 2.20 présente l'implémentation de la stratégie du *Wave Pipelining* pour le MAC. Elle ne prévoit aucun registre intermédiaire, conformément à la définition initiale de la stratégie par Cotten (1969). Afin de permettre une comparaison similaire à la stratégie PA avec emprunt

d'horloge, le paramètre K de l'équation 1.11 est posé comme étant égal à zéro, impliquant que l'entièreté du délai combinatoire est compensée par le délai asymétrique Δ_O posé sur la broche de contrôle du registre de l'étage de sortie.

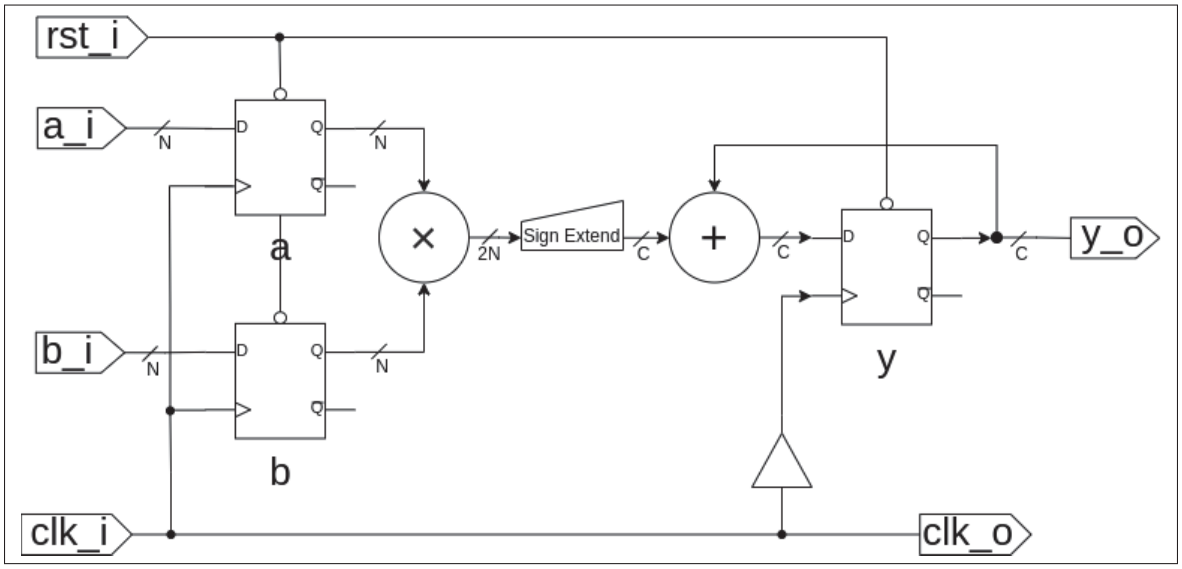


Figure 2.20 Schéma du MAC avec pipeline en vague

2.3.4 Stratégie pseudo-asynchrone avec emprunt d'horloge

La figure 2.21 décrit l'implémentation de la stratégie pseudo-asynchrone avec emprunt d'horloge. Cette dernière est basée sur la conversion du circuit synchrone de la figure 2.17 en une structure PA afin de pouvoir manuellement spécifier la compensation en fonction du délai des étages du pipeline.

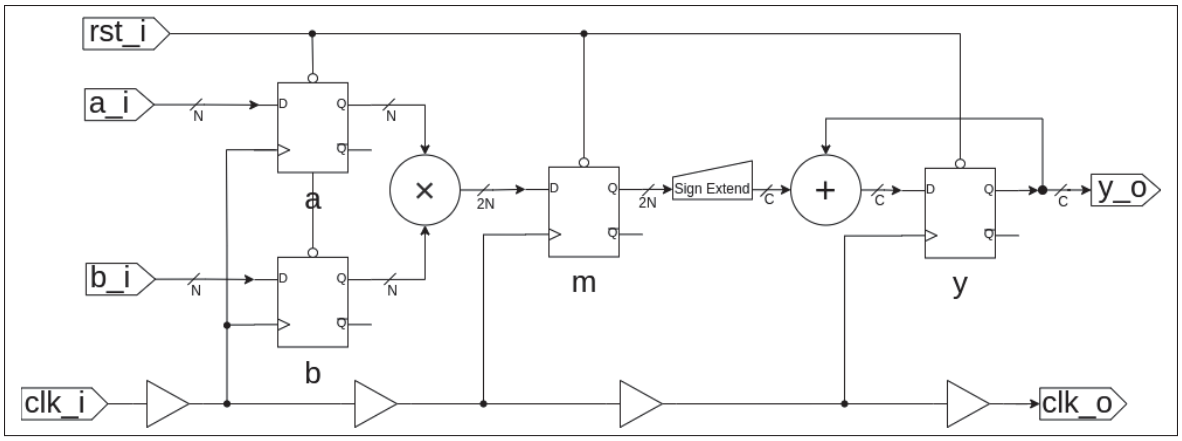


Figure 2.21 Schéma du MAC pseudo-asynchrone avec emprunt d'horloge

2.4 Sommaire

Le présent chapitre explicitait les informations relatives à la méthode développée pour la génération des résultats, aux contraintes utilisées pour implémenter les différentes stratégies de synchronisation et à la structure des différents circuits implémentés pour la comparaison. Dans le prochain chapitre, les résultats obtenus sont décrits et analysés afin de faire ressortir les caractéristiques de chaque stratégie quant à leurs forces et à leurs faiblesses.

CHAPITRE 3

RÉSULTATS ET ANALYSE

Ce chapitre présente les résultats obtenus suite à l'implémentation des différentes stratégies de synchronisation décrites dans le chapitre 2 sur la méthodologie. Les analyses utilisent la stratégie synchrone comme point de référence étant donné son utilisation *de facto* dans l'industrie.

Les variables utilisées pour dicter l'analyse sont la surface totale post-PNR, la période moyenne entre deux résultats successifs, la latence de propagation des vecteurs du chemin de données ainsi que les puissances moyennes et crêtes pour des vecteurs pseudo-aléatoires injectés à un débit maximal.

Deux tailles de MAC sont implémentées pour chaque stratégie de synchronisation. Le premier possède des vecteurs en entrée de taille $N = 8$ bits et un accumulateur de taille $C = 19$ bits, nommé MAC8, alors que le deuxième possède des vecteurs en entrée de taille $N = 16$ bits et un accumulateur de taille $C = 38$ bits, nommé MAC16.

Le MAC8 a été développé à cause de sa petite taille, ce qui a permis de réduire le temps de compilation et de faciliter le développement des contraintes présentées dans le chapitre de la méthodologie. Quant au MAC16, il a été développé afin d'observer l'impact de l'augmentation de la taille du circuit sur les résultats, d'où le dédoublement des variables N et C.

Le chapitre débute par une exploration des résultats obtenus suite aux améliorations apportées à la stratégie LCS pour le MAC asynchrone BD avec CE. Ces résultats sont ensuite comparés aux autres stratégies implémentées sur trois étages de pipeline, soit le MAC synchrone et le PA avec emprunt d'horloge.

Par la suite, des observations spécifiques face aux résultats de la stratégie PA par emprunt d'horloge sont décrites et des optimisations sont proposées, afin d'améliorer les QOR. Cela donne naissance au PA optimisé, qui est utilisé dans les comparaisons subséquentes.

Enfin, les résultats des stratégies hautes performances, soit le MAC synchrone sur 4 étages, le PA optimisé et le WP sont comparés en fonction des performances maximales atteintes de la stratégie du PA optimisé.

Le tout est conclu avec une discussion récapitulative des constatations faites pour chaque stratégie de synchronisation en fonction des résultats présentés.

3.1 Amélioration de la méthode LCS

Comme décrit dans la méthodologie, les écrits sur la méthode LCS proposent une façon de contraindre le délai du chemin combinatoire en spécifiant le délai combinatoire voulu en utilisant la période des horloges racines et en retirant les exceptions sur les chemins multicycles.

En pratique cependant, cette méthode ne fonctionne pas. Cette situation force le concepteur à utiliser la méthode de base qui pousse l'outil de synthèse à compiler jusqu'à l'abandon. Cette limitation entraîne un grand temps de compilation et néglige des optimisations au niveau de la surface et de la puissance du circuit.

Les tableaux 3.1 et 3.2 présentent les résultats pour le MAC8 et le MAC16 respectivement, où une comparaison est faite entre la méthode LCS de base (c.-à-d. après la mise à jour pour ICC2) et celle qui contient les deux améliorations proposées.

Tableau 3.1 Résultats de l'amélioration de la méthode LCS pour le MAC8

Paramètres	**LCS de base**	**LCS améliorée**	**Variation (%)**
Surface totale (μm^2)	13217.30	11548.95	- 12.6
Période équivalente (ns)	4.42	4.66	+ 5.4
Latence moyenne (ns)	9.83	10.21	+ 3.9
Puissance moyenne (mW)	6.14	5.17	- 15.8
Puissance crête (mW)	27.4	26.5	- 3.3
Temps de synthèse (s)	295.51	168.44	- 43.0

Pour ce qui est du MAC8, on constate effectivement une réduction de 12.6% de la surface totale ainsi qu'une réduction de 15.8% de la puissance moyenne du circuit au coût d'une augmentation de la période équivalente. Cela s'explique par le fait qu'étant donné que l'outil de synthèse est équipé avec des contraintes spécifiques sur les délais attendus de chaque étage, il est en mesure de muter son centre d'attention de l'optimisation de la vitesse vers des optimisations en surface et en puissance lors de la convergence du STA.

Curieusement, on observe une augmentation de 5.4% de la période équivalente en réponse à l'application des contraintes d'optimisation, quand, dans les faits, on s'attendrait à obtenir un résultat similaire. À première vue, on pourrait croire que l'utilisation de cellules plus petites augmenterait les délais de propagations, ralentissant le flux d'exécution. Cependant, l'hypothèse la plus probable est que, étant donné la petite taille du circuit, l'outil de synthèse a plus de facilité à pousser les optimisations en vitesse en l'absence de contraintes. Cette hypothèse implique qu'avec l'augmentation de la taille circuit, la période équivalente entre les deux méthodes aura tendance à se rapprocher, ce qui est observé dans les résultats du MAC16.

Enfin, on observe une convergence beaucoup plus rapide vers un circuit par la réduction de 43.0% du temps nécessaire pour exécuter l'étape de synthèse, ce qui démontre que l'outil de synthèse n'est pas obligé d'exécuter des optimisations jusqu'à l'abandon.

Tableau 3.2 Résultats de l'amélioration de la méthode LCS pour le MAC16

Paramètres	**LCS de base**	**LCS améliorée**	**Variation (%)**
Surface totale (μm^2)	37693.78	34679.77	- 8.0
Période équivalente (ns)	5.65	5.68	+ 0.5
Latence moyenne (ns)	13.29	12.65	- 4.8
Puissance moyenne (mW)	13.1	11.8	- 9.9
Puissance crête (mW)	90.5	114.1	+ 26.1
Temps de synthèse (s)	801.32	437.96	- 45.3

Le tableau 3.2 montre que les tendances du MAC8 se retrouvent tout autant dans celles du MAC16. En effet, on observe une réduction de 8.0 % de la surface totale et une réduction de 9.9% de la puissance moyenne.

On observe aussi qu'avec l'augmentation de la taille des vecteurs, la période équivalente tend à être similaire étant donné une faible variation de 0.5%. Cela confirme l'hypothèse que l'augmentation de la taille du circuit a comme effet de rendre la tâche d'optimisation plus difficile pour l'outil de synthèse lorsqu'il n'est doté d'aucune contrainte.

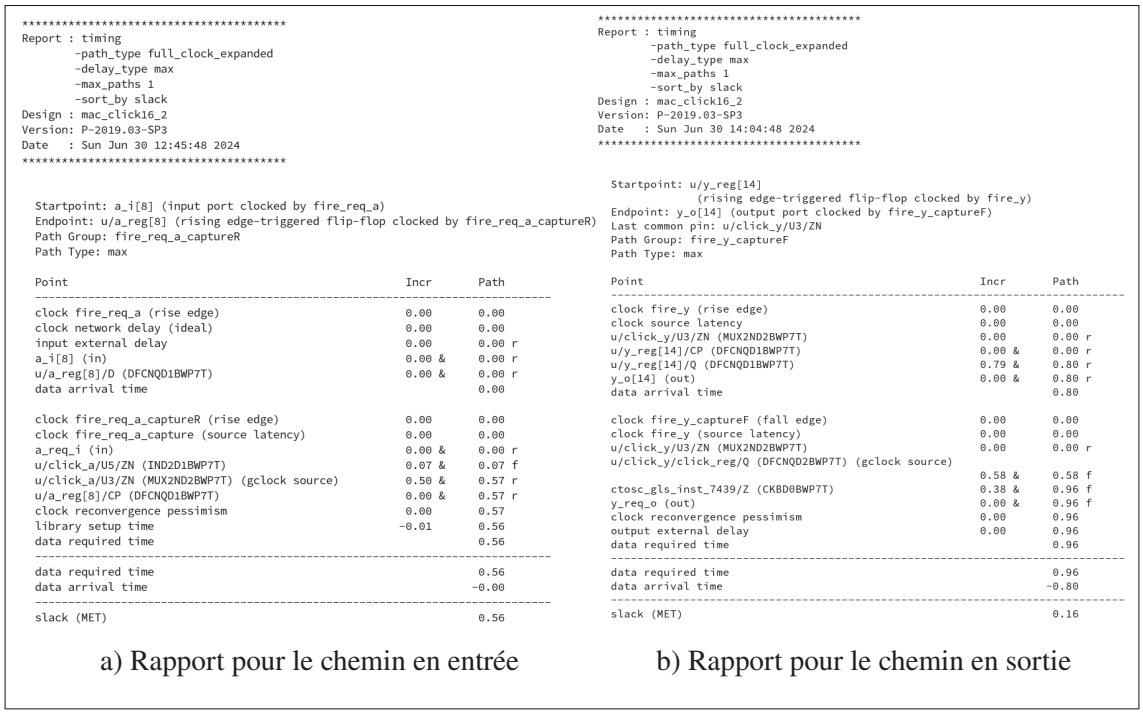
Étonnamment, on observe une grande augmentation de 26.1% de la puissance crête pour le circuit qui utilise la méthode LCS améliorée pour le pire coin d'analyse, ce qui semble contre-intuitif compte tenu de la réduction de la puissance moyenne. Il est cependant à noter que la tendance ne se maintient pas pour les autres coins d'analyse −ce qui s'observe dans le tableau I-3 placé en annexe−impliquant qu'en moyenne, pour tous les coins, la puissance crête tend à être similaire.

Finalement, tout comme pour le MAC8, l'ajout des contraintes permet d'atteindre une meilleure qualité des résultats beaucoup plus rapidement, soit par une réduction du temps de synthèse de 45.3%.

La section sur les améliorations de la méthode LCS se termine par la figure 3.1 qui présente les rapports de l'analyse statique des entrées et des sorties en réponse aux contraintes développées dans la section de la méthodologie. Ces rapports permettent de valider le fonctionnement aux interfaces par l'observation des marges (*slack*) positives de 0.56 ns et 0.16 ns respectivement.

L'aspect intéressant de l'amélioration est qu'elle permet d'effectuer l'analyse statique entre le chemin de données et un port de requête ou d'acquiescement, chose qui n'avait pas été développée par les auteurs de la méthode.

À noter que, comme les améliorations proposées génèrent des résultats supérieurs en termes de surface et de puissance, les configurations qui utilisent la méthode améliorée sont utilisées dans les prochaines comparaisons de ce chapitre.



```
****************************************
Report : timing
        -path_type full_clock_expanded
        -delay_type max
        -max_paths 1
        -sort_by slack
Design : mac_click16_2
Version: P-2019.03-SP3
Date   : Sun Jun 30 12:45:48 2024
****************************************


  Startpoint: a_i[8] (input port clocked by fire_req_a)
  Endpoint: u/a_reg[8] (rising edge-triggered flip-flop clocked by fire_req_a_captureR)
  Path Group: fire_req_a_captureR
  Path Type: max

  Point                                                   Incr       Path
  -----------------------------------------------------------------------
  clock fire_req_a (rise edge)                            0.00       0.00
  clock network delay (ideal)                             0.00       0.00
  input external delay                                    0.00       0.00 r
  a_i[8] (in)                                             0.00 &     0.00 r
  u/a_reg[8]/D (DFCNQD1BWP7T)                             0.00 &     0.00 r
  data arrival time                                                  0.00

  clock fire_req_a_captureR (rise edge)                   0.00       0.00
  clock fire_req_a_capture (source latency)               0.00       0.00
  a_req_i (in)                                            0.00 &     0.00 r
  u/click_a/U5/ZN (IND2D1BWP7T)                           0.07 &     0.07 f
  u/click_a/U3/ZN (MUX2ND2BWP7T) (gclock source)          0.50 &     0.57 r
  u/a_reg[8]/CP (DFCNQD1BWP7T)                            0.00 &     0.57 r
  clock reconvergence pessimism                           0.00       0.57
  library setup time                                     -0.01       0.56
  data required time                                                 0.56
  -----------------------------------------------------------------------
  data required time                                                 0.56
  data arrival time                                                 -0.00
  -----------------------------------------------------------------------
  slack (MET)                                                        0.56
```

a) Rapport pour le chemin en entrée

```
****************************************
Report : timing
        -path_type full_clock_expanded
        -delay_type max
        -max_paths 1
        -sort_by slack
Design : mac_click16_2
Version: P-2019.03-SP3
Date   : Sun Jun 30 14:04:48 2024
****************************************


  Startpoint: u/y_reg[14]
              (rising edge-triggered flip-flop clocked by fire_y)
  Endpoint: y_o[14] (output port clocked by fire_y_captureF)
  Last common pin: u/click_y/U3/ZN
  Path Group: fire_y_captureF
  Path Type: max

  Point                                                   Incr       Path
  -----------------------------------------------------------------------
  clock fire_y (rise edge)                                0.00       0.00
  clock source latency                                    0.00       0.00
  u/click_y/U3/ZN (MUX2ND2BWP7T)                          0.00       0.00 r
  u/y_reg[14]/CP (DFCNQD1BWP7T)                           0.00 &     0.00 r
  u/y_reg[14]/Q (DFCNQD1BWP7T)                            0.79 &     0.80 r
  y_o[14] (out)                                           0.00 &     0.80 r
  data arrival time                                                  0.80

  clock fire_y_captureF (fall edge)                       0.00       0.00
  clock fire_y (source latency)                           0.00       0.00
  u/click_y/U3/ZN (MUX2ND2BWP7T)                          0.00       0.00 r
  u/click_y/click_reg/Q (DFCNQD2BWP7T) (gclock source)
                                                          0.58 &     0.58 f
  ctosc_gls_inst_7439/Z (CKBD0BWP7T)                      0.38 &     0.96 f
  y_req_o (out)                                           0.00 &     0.96 f
  clock reconvergence pessimism                           0.00       0.96
  output external delay                                   0.00       0.96
  data required time                                                 0.96
  -----------------------------------------------------------------------
  data required time                                                 0.96
  data arrival time                                                 -0.80
  -----------------------------------------------------------------------
  slack (MET)                                                        0.16
```

b) Rapport pour le chemin en sortie

Figure 3.1 Analyse statique des chemins d'entrée (a) et de sortie (b) pour la méthode LCS

3.2 Stratégie sur trois étages de registres

Dans la présente section, on traite des stratégies implémentées avec trois étages de registres, soit un registre en entrée, un registre entre la multiplication et l'accumulation et un registre en sortie. On y retrouve les stratégies synchrones avec recadencement, PA par emprunt d'horloge et asynchrone BD avec CE (c.-à-d. LCS améliorée).

La période cible de cette comparaison est de 2.35 ns pour le MAC8 et de 3.00 ns pour le MAC16. Elle est dictée par la période la plus basse pouvant être atteinte par la stratégie synchrone avec recadencement pour le pire coin d'analyse.

Cette période cible est exemptée pour la stratégie asynchrone BD, car, par sa nature autorégulée dépendant des délais combinatoires, il est impossible de contrôler la période équivalente du circuit, ce qui explique la différence marquée dans les résultats.

Il est à noter que la stratégie synchrone sans recadencement est omise des résultats étudiés, car elle n'était pas en mesure d'atteindre les performances minimales requises pour la comparaison. En effet, cette stratégie était en mesure d'atteindre une période minimale de 2.86 ns pour le MAC8 et une période de 3.65 ns pour le MAC16 pour le pire coin d'analyse. Cependant, comme le PA par emprunt d'horloge peut être grossièrement comparé à un recadencement manuel, il est plus cohérent d'observer les différences entre les résultats que l'outil de synthèse est en mesure d'offrir et les résultats de la nouvelle stratégie.

Les tableaux 3.3 et 3.5 présentent les résultats des circuits du MAC8 et du MAC16 respectivement. Ils sont accompagnés de la distribution de la consommation de la puissance moyenne avec les tableaux 3.4 et 3.6 pour les circuits du MAC8 et du MAC16 respectivement.

Tableau 3.3 Comparaison des stratégies synchrone avec recadencement, pseudo-asynchrone et asynchrone pour le MAC8

Paramètres	**Synchrone,** $L = 3$	**PA**	**Asynchrone**
Surface totale (μm^2)	11737.73	10943.07	11548.95
Période (ns)	2.32	2.24	4.66
Latence (ns)	5.71	6.18	10.26
Puissance crête (mW)	47.2	29.1	26.5
Puissance moyenne totale (mW)	10.3	14.2	5.20
mW/MHz moyen	0.0238	0.0318	0.0242

Tableau 3.4 Distribution des puissances moyennes du MAC8 pour la comparaison des stratégies synchrone, pseudo-asynchrone et asynchrone

Puissance moyenne	**Synchrone,** $L = 3$	**Pseudo-asynchrone**	**Asynchrone**
Réseau de l'horloge (mW)	4.33	9.38	2.15
Registres (mW)	0.89	0.81	0.52
Combinatoire (mW)	5.09	3.98	2.50

Pour un circuit développé à contraintes égales, on observe, dans le tableau 3.3, que la stratégie PA par emprunt d'horloge a généré un circuit 6.8% plus petit au niveau de la surface totale et 3.4% plus rapide que son équivalent synchrone. Ces améliorations viennent cependant au

coût d'une augmentation de 38% de la puissance moyenne totale, tel qu'observé dans le tableau 3.4, où 9.38 mW (66%) de cette puissance est utilisé par le réseau de l'horloge pour le PA, contrairement à 4.33 mW (42%) pour la synchronisation synchrone.

Cette augmentation de 117% de la puissance moyenne associée au réseau de l'horloge s'explique par la forte utilisation de tampons dans le réseau de l'horloge afin de compenser le délai combinatoire. L'aspect unique de ces tampons est qu'ils ont la caractéristique d'être conçus pour que le délai de la section *p-channel metal-oxide-semiconductor* (PMOS) soit égal au délai de la section *n-channel metal-oxide-semiconductor* (NMOS) de la cellule normalisée afin d'éviter une distorsion de l'impulsion de l'horloge.

Comme la section PMOS du circuit est normalement plus lente que la portion NMOS, une augmentation de la surface PMOS est effectuée pour diminuer le délai de propagation, ce qui a comme effet d'augmenter la puissance consommée par ces tampons, justifiant ainsi l'augmentation de la consommation.

On retrouve cependant une diminution de 38% de la puissance crête par rapport à la synchronisation synchrone à cause de l'asymétrie proposée par la stratégie. Cette asymétrie est fortement exploitée par l'outil PNR autant globalement que localement, ce qui a comme effet d'étaler le pic de puissance.

Au niveau du circuit asynchrone implémenté à l'aide de la méthode LCS améliorée, on observe que sa période équivalente est le double de la période ciblée pour la même taille que son équivalent synchrone, et ce, malgré les efforts d'optimisations. Cela est dû à deux facteurs qui proviennent de la logique de contrôle : le signal d'acquiescement et le contrôleur factice dans la boucle de rétroaction.

Pour ce qui est du signal d'acquiescement, suivant la réception d'une requête ainsi qu'une impulsion sur le signal *fire*, le signal d'acquiescement est généré par la bascule D du CE après 0.69 ns, puis il se propage à travers la logique du contrôleur de l'étage précédent en 0.52 ns, totalisant un délai de 1.21 ns. Ce délai est problématique, car, contrairement à la stratégie PA qui

ne possède pas ce chemin d'acquiescement, la prochaine donnée peut être envoyée seulement après $T_{REQ} + T_{ACK}$, réduisant ainsi le débit de l'étage de pipeline.

Pour ce qui est de la boucle de rétroaction, comme mentionné dans la description du circuit, un contrôleur factice doit être ajouté avec une opération de fourche et d'union avec le contrôleur du registre du chemin de donnée afin de garantir la vivacité du circuit (Gimenez, 2021). Cependant, lorsqu'on regarde les délais dans l'analyse statique, on observe une marge de 1.14 ns en comparant le délai de la boucle de rétroaction du chemin de contrôle par rapport au délai de la boucle de rétroaction du chemin de données. En analysant de plus près le rapport de STA, on observe que 82% de cette marge, soit 0.94 ns, sont attribués au délai du contrôleur factice. Le retrait de ce contrôleur factice serait donc une optimisation permettant d'augmenter la période moyenne du circuit.

Finalement, au niveau de la puissance moyenne, on observe que le circuit asynchrone (LCS amélioré) consomme la moitié de celle du circuit synchrone, une caractéristique directement liée à sa période moyenne plus faible.

Tableau 3.5 Comparaison des stratégies synchrone, pseudo-asynchrone et asynchrone pour le MAC16

Paramètres	**Synchrone,** $L = 3$	**Pseudo-asynchrone**	**Asynchrone**
Surface combinatoire (μm^2)	28287.35	21102.46	29297.14
Surface non combinatoire (μm^2)	6366.08	5007.25	5382.63
Surface totale (μm^2)	34653.43	26109.71	34679.77
Période (ns)	2.98	2.92	5.67
Latence maximale (ns)	7.03	8.24	12.73
Puissance moyenne totale (mW)	22.2	18.5	11.8
Puissance crête (mW)	110.5	67.5	114.1
mW/MHz moyen	0.0662	0.0540	0.0669

Des constatations intéressantes sont obtenues lorsque la taille du MAC est doublée, telle qu'illustrée par les résultats du tableau 3.5. On observe, dans un premier temps, que la synchronisation du PA obtient de meilleures performances pour tous les paramètres évalués, à l'exception de la latence par rapport à la stratégie synchrone.

La différence la plus évidente est celle de la surface totale du PA, qui affiche une réduction moyenne totale de 24.7% par rapport aux deux autres stratégies. En regardant plus précisément la distribution de la surface, on observe que la stratégie synchrone utilise 28287.35 μm^2 pour la surface combinatoire et 6366.08 μm^2 pour la surface non combinatoire, contrairement au PA qui utilise 21102.46 μm^2 pour la surface combinatoire et 5007.25 μm^2 pour la surface non combinatoire. Ces résultats constituent respectivement une réduction de 25.4% de la surface combinatoire ainsi qu'une réduction de 21.3% de la surface non combinatoire pour la stratégie du PA.

Pour ce qui est de la surface combinatoire, ce résultat s'explique par le fait que la stratégie PA semble avoir un avantage lors de l'étape de synthèse parce que les contraintes ne poussent pas le circuit jusqu'à son potentiel maximal, contrairement à la stratégie synchrone. Cela implique que la grande partie des cellules utilisées sont plus petites étant donné les contraintes moins strictes, engendrant une augmentation de la latence de 17.2% par rapport à la stratégie synchrone.

Pour ce qui est de la surface non combinatoire, l'augmentation observée du côté synchrone s'explique par le fait que l'algorithme de recadencement a tenté de minimiser le nombre de registres dupliqués en fonction des nœuds du circuit, mais qu'il n'a pas réussi à atteindre une quantité finale égale à celle de départ. Cela s'exprime par une augmentation de 19.7% du nombre de registres, ces derniers passant de 102 à 127.

Tableau 3.6 Distribution des puissances moyennes du MAC16 pour la comparaison des stratégies synchrone, pseudo-asynchrone et asynchrone

Puissance moyenne	**Synchrone,** $L = 3$	**Pseudo-asynchrone**	**Asynchrone**
Réseau de l'horloge (mW)	5.71	7.44	3.00
Registres (mW)	1.66	1.38	0.91
Combinatoire (mW)	14.8	9.73	7.86

Dans le cas de la puissance moyenne, on observe une réduction de 16.7% entre la stratégie synchrone et la stratégie PA, ce qui ne semble pas suivre les résultats observés au niveau du

MAC8. Cette tendance se reflète également sur le ratio de la puissance par unité de fréquence, démontrant une réduction 18.4%.

En décomposant les sources de la consommation de puissance, comme montré dans le tableau 3.6, on remarque que la consommation liée au réseau de l'horloge est de 8.35 mW (45.15%) pour la stratégie du PA et de 5.71 mW (25.74%) pour la stratégie synchrone, soit une augmentation de 46.2% de la consommation du côté PA confirmant ainsi la tendance de surconsommation à ce niveau.

Cette puissance moyenne inférieure s'explique principalement par le fait au fait que la surface du PA est largement inférieure à la surface synchrone. De ce fait, elle utilise des cellules combinatoires plus petites et donc moins énergivores, ce qui s'observe par la réduction de 34.2% entre les deux stratégies pour cette catégorie.

Pour la puissance crête, on observe tout comme le MAC8 une grande réduction 39.9% par rapport à la synchronisation synchrone, ce qui indique que la tendance reste donc similaire pour ce paramètre.

Au niveau de la stratégie asynchrone, on observe quelques tendances semblables par rapport au MAC8 : une surface équivalente au circuit synchrone pour une période environ doublée et une puissance moyenne divisée en deux.

Bien que la puissance crête soit approximativement égale à celle du circuit synchrone, la tendance n'est pas constante pour les autres coins d'analyse, comme observé dans le tableau I-4 en annexe, ce qui implique la possibilité que ce ne soit qu'une question de délais. Étant donné ses piètres performances, la stratégie est omise des prochaines sections.

3.3 Optimisations du pseudo-asynchrone avec emprunt d'horloge

Suite à l'implémentation de la stratégie PA par emprunt d'horloge, une analyse du réseau de l'horloge a été effectuée avec l'outil d'analyse statique de ICC2. Le rapport d'analyse statique présenté dans la figure 3.2 montre que l'outil de PNR a inséré une ligne à retard entre le port

d'entrée de l'horloge *clk_i* et le port de sortie de l'horloge *clk_o* lors de l'implémentation de l'arbre de l'horloge. Dans cette figure, on remarque que les cellules instanciées sont des tampons d'horloges (*CKBD*) ayant un le nom *ctosc_gls_inst_####*, où l'acronyme CTO représente *Clock Tree Optimisation*. Le rapport démontre le fonctionnement du lot de contraintes développé pour l'horloge virtuelle *clk_latency*, cette dernière ayant une latence de 6.13 ns, soit la latence maximale du chemin combinatoire.

Une première constatation qui découle de ce rapport d'analyse statique est que la marge de temps est de 0.81 ns, ce qui indique que l'on a un surplus de tampons sur la ligne à retard. Ces tampons, plus spécifiquement les CKB12BWP7T se qui se trouvent à la fin de la chaîne, ont comme caractéristique d'augmenter injustement la taille ainsi que la puissance moyenne du circuit parce qu'ils sont insérés à cause des limitations de l'outil PNR.

Une analyse plus détaillée de la structure du réseau de l'horloge permet d'observer que des tampons excédentaires sont notés tout au long de la ligne à retard. La commande ICC2 *remove_buffers* est utilisée pour retirer ces tampons excédentaires et ainsi faire tendre la marge de temps vers zéro, tel qu'illustré dans la figure 3.3.

Une deuxième observation révèle que l'outil de PNR n'infère pas toujours des tampons d'horloge de taille assez élevée pour compenser la forte charge capacitive associée au nombre de bascules connecté au dernier tampon, telle qu'illustrée par la partie gauche de la figure 3.4. Cela a comme impact de déformer le signal de l'horloge en l'augmentant de la constante de temps *RC* du circuit, entraînant ainsi des violations de la taille d'impulsion minimale de l'horloge.

La commande ICC2 *size_cell* est donc utilisée sur le tampon qui s'attaque au nœud afin d'augmenter sa taille, de diminuer sa résistance de source et d'augmenter sa capacité à fournir du courant.

Il est à noter qu'une alternative possible face à la violation de l'impulsion minimale serait d'ajouter une contrainte pour limiter le nombre d'éléments connecté au même nœud lors de l'étape de la construction de l'arbre de l'horloge.

```
icc2_shell> report_timing -delay_type min -to clk_latency -corner WC_corner
...
  Point                                                   Incr      Path
  ------------------------------------------------------------------------------
  clock clk (rise edge)                                   0.00      0.00
  clock network delay (propagated)                        0.00      0.00

  clk_i (in)                                              0.00      0.00 r
  ctosc_gls_inst_1645/Z (CKBD12BWP7T)                     0.10      0.10 r
  ctosc_gls_inst_1644/Z (CKBD2BWP7T)                      0.14      0.24 r
  ctosc_gls_inst_1643/Z (CKBD3BWP7T)                      0.13      0.37 r
  ctosc_gls_inst_1642/Z (CKBD4BWP7T)                      0.13      0.50 r
  ctosc_gls_inst_1641/Z (CKBD6BWP7T)                      0.13      0.63 r
  ctosc_gls_inst_1640/Z (CKBD6BWP7T)                      0.13      0.76 r
  ctosc_gls_inst_1639/Z (CKBD6BWP7T)                      0.13      0.88 r
  ctosc_gls_inst_1638/Z (CKBD6BWP7T)                      0.13      1.01 r
  ctosc_gls_inst_1664/Z (CKBD4BWP7T)                      0.15      1.16 r
  ctosc_gls_inst_1656/Z (CKBD3BWP7T)                      0.13      1.29 r
  ccd_setup_inst_2682/Z (CKBD2BWP7T)                      0.16      1.44 r
  ctosc_gls_inst_1649/Z (CKBD3BWP7T)                      0.17      1.61 r
  ctosc_gls_inst_1677/Z (CKBD4BWP7T)                      0.16      1.77 r
  ctosc_gls_inst_1682/Z (CKBD6BWP7T)                      0.14      1.91 r
  ctosc_gls_inst_1701/Z (CKBD6BWP7T)                      0.12      2.03 r
  ctosc_gls_inst_1702/Z (CKBD6BWP7T)                      0.12      2.15 r
  ctosc_gls_inst_1703/Z (CKBD6BWP7T)                      0.12      2.27 r
  ctosc_gls_inst_1705/Z (CKBD4BWP7T)                      0.12      2.39 r
  ctosc_gls_inst_1707/Z (CKBD3BWP7T)                      0.13      2.52 r
  ctosc_gls_inst_1711/Z (CKBD2BWP7T)                      0.15      2.67 r
  ctosc_gls_inst_1718/Z (CKBD3BWP7T)                      0.14      2.82 r
  ctosc_gls_inst_1722/Z (CKBD3BWP7T)                      0.13      2.94 r
  ctosc_gls_inst_1726/Z (CKBD3BWP7T)                      0.13      3.08 r
  ctosc_gls_inst_1730/Z (CKBD3BWP7T)                      0.14      3.22 r
  ctosc_gls_inst_1734/Z (CKBD3BWP7T)                      0.13      3.35 r
  ctosc_gls_inst_1739/Z (CKBD1BWP7T)                      0.14      3.49 r
  ctosc_gls_inst_1744/Z (CKBD2BWP7T)                      0.15      3.63 r
  ctosc_gls_inst_1748/Z (CKBD3BWP7T)                      0.14      3.77 r
  ctosc_gls_inst_1752/Z (CKBD3BWP7T)                      0.13      3.90 r
  ctosc_gls_inst_1756/Z (CKBD2BWP7T)                      0.14      4.04 r
  ctosc_gls_inst_1780/Z (CKBD3BWP7T)                      0.14      4.18 r
  ctosc_gls_inst_1798/Z (CKBD3BWP7T)                      0.13      4.31 r
  ctosc_gls_inst_1812/Z (CKBD3BWP7T)                      0.13      4.44 r
  ctosc_gls_inst_1846/Z (CKBD3BWP7T)                      0.12      4.56 r
  ctosc_gls_inst_1861/Z (CKBD2BWP7T)                      0.16      4.72 r
  ctosc_gls_inst_1870/Z (CKBD4BWP7T)                      0.13      4.85 r
  ctosc_gls_inst_1854/Z (CKBD4BWP7T)                      0.14      4.99 r
  ctosc_gls_inst_1836/Z (CKBD12BWP7T)                     0.12      5.11 r
  ctosc_gls_inst_1831/Z (CKBD12BWP7T)                     0.11      5.23 r
  ctosc_gls_inst_1826/Z (CKBD12BWP7T)                     0.11      5.34 r
  ctosc_gls_inst_1820/Z (CKBD12BWP7T)                     0.11      5.45 r
  ctosc_gls_inst_1806/Z (CKBD12BWP7T)                     0.11      5.57 r
  ctosc_gls_inst_1803/Z (CKBD12BWP7T)                     0.11      5.68 r
  ctosc_gls_inst_1800/Z (CKBD12BWP7T)                     0.11      5.79 r
  ctosc_gls_inst_1789/Z (CKBD12BWP7T)                     0.11      5.91 r
  ctosc_gls_inst_1786/Z (CKBD12BWP7T)                     0.11      6.02 r
  ctosc_gls_inst_1784/Z (CKBD12BWP7T)                     0.11      6.13 r
  ctosc_gls_inst_1700/Z (CKBD1BWP7T)                      0.22      6.34 r
  ctosc_gls_inst_1697/Z (CKBD12BWP7T)                     0.15      6.50 r
  ctosc_gls_inst_1694/Z (CKBD12BWP7T)                     0.10      6.60 r
  ctosc_gls_inst_1692/Z (CKBD2BWP7T)                      0.20      6.80 r
  APS_CLK_ISO_0/Z (CKBD12BWP7T)                           0.14      6.94 r
  clk_o (out)                                             0.00      6.94 r
  data arrival time                                                 6.94

  clock clk_latency (rise edge)                           0.00      0.00
  clock network delay (ideal)                             6.13      6.13
  output external delay                                   0.00      6.13
  data required time                                                6.13
  ------------------------------------------------------------------------------
  data required time                                                6.13
  data arrival time                                                -6.94
  ------------------------------------------------------------------------------
  slack (MET)                                                       0.81
```

Figure 3.2 Rapport d'analyse statique généré lors de l'analyse de l'horloge *clk_latency* pour la stratégie PA

La combinaison des commandes ICC2 *remove_buffer* et *size_cell* peuvent être utilisées afin d'optimiser l'arbre de l'horloge et de réduire la taille des tampons d'horloges instanciées par l'outil de PNR. En effet, dans certains cas des tampons de fortes tailles sont instanciés à la place

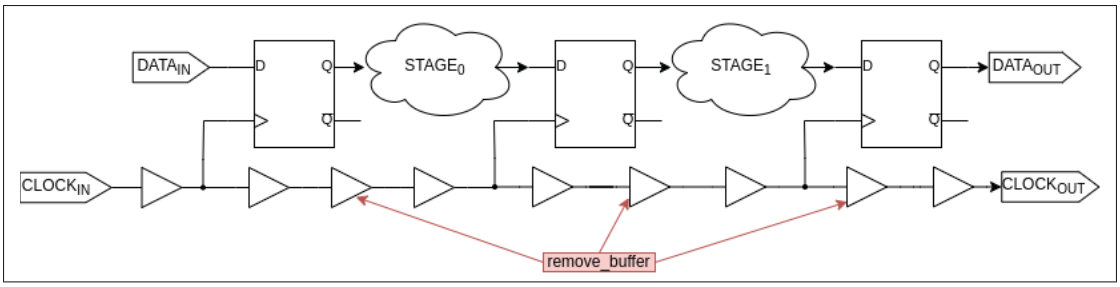


Figure 3.3 Ajustement du nombre de tampons sur la ligne à retard pour la stratégie PA

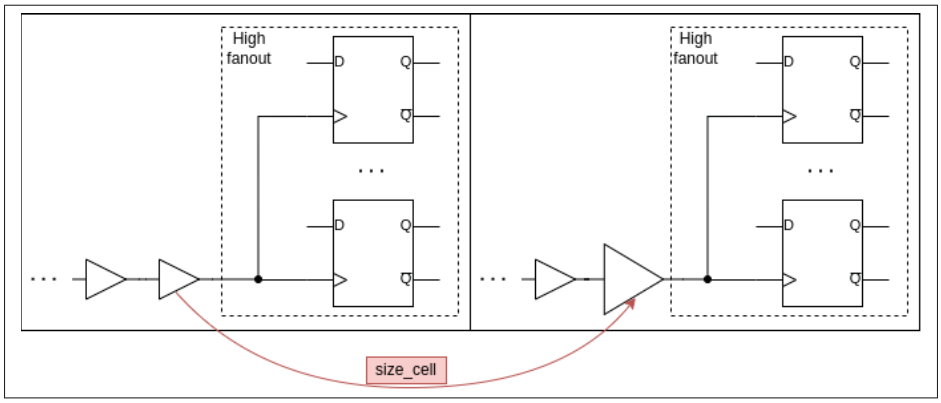


Figure 3.4 Ajustement de la taille du tampon connecté aux registres

de petits tampons d'horloges. Ces gros tampons ont la caractéristique de fournir des délais de propagation très courts, de prendre plus d'espace et d'être plus énergivores. Il y a donc avantage à les supprimer ou à les redimensionner en des tampons plus petits.

Ces deux commandes sont généralement suivies des commandes ICC2 *route_opt* et *compute_clock_latency* afin de corriger les erreurs de *Design Rule Check* (DRC), de réorganiser la logique combinatoire et de recalculer le délai des horloges virtuelles utilisé pour contraindre les entrées et les sorties.

Le tableau 3.7 présente les résultats initiaux du MAC8 PA suivi des résultats pour le même circuit optimisé manuellement. Le circuit est synthétisé avec une période cible de 2.2 ns.

Tableau 3.7 Résultats de l'optimisation manuelle du PA pour le MAC8

Paramètres	PA de base @ 2.2 ns	PA optimisé @ 2.2 ns	Variation (%)
Surface totale (μm^2)	11818.96	11309.67	- 4.3
Période (ns)	2.15	2.18	+ 1.4
Latence maximale (ns)	6.1	6.58	+ 7.7
Puissance moyenne totale (mW)	16.4	12.2	- 25.6
Puissance crête (mW)	40.9	36.0	- 12.0
mW/MHz moyen	0.0353	0.0266	- 24.6

Tableau 3.8 Différence de distribution des puissances moyennes du MAC8 pour l'optimisation manuelle du PA

Puissance moyenne	PA de base @ 2.2 ns	PA optimisé @ 2.2 ns	Variation (%)
Réseau de l'horloge (mW)	10.57	6.08	- 42.4
Registres (mW)	0.89	0.79	- 11.2
Combinatoire (mW)	4.92	5.34	+ 8.5

En réponse à l'optimisation manuelle, on observe une diminution de 509.25 μm^2 de la surface totale utilisée liée à au retrait et à l'ajustement de la taille des tampons, ce nombre passant de 143 à 124.

On retrouve aussi un réajustement de la taille des cellules du chemin combinatoire suivant la réorganisation des tampons avec la commande *route_opt*, ce qui a pour effet d'augmenter de 7.7% la latence du circuit. Cette augmentation pourrait être théoriquement évitée dans le cas où le concepteur optimiserait parfaitement la quantité et la taille tampons tout en évitant les violations du STA. Cependant, cette tâche peut s'avérer coûteuse au niveau du temps d'implémentation étant donné la complexité du réseau qui peut être généré.

Malgré cela, le plus grand bénéfice observé est une réduction globale de la puissance moyenne consommée de 25.6% pour la proportion consommée par le réseau de l'horloge. En effet, la puissance du réseau diminue de 42.4%, passant de 10.57 mW (64.46%) à 6.07 mW (49.78%).

Le dernier élément observé est l'augmentation de la période de l'horloge. En effet, étant donné que la période de l'horloge dépend de l'équation 1.14, tout comme pour le WP, l'ajustement des tampons offre la possibilité de modifier la période de l'horloge en fonction de la plus petite marge du chemin combinatoire le moins long.

Cette constatation a suscité une hypothèse intéressante, soit de relaxer les contraintes lors de la synthèse afin de générer un circuit plus petit et d'utiliser les méthodes d'optimisations manuelles décrites plus haut pour réduire la période de l'horloge.

Le tableau 3.9 présente la comparaison de deux circuits PA, un de base synthétisée avec une période de 2.75 ns sans interventions manuelles et un synthétisé avec une période de 2.80 ns et ajusté manuellement pour retirer les tampons excédentaires et tester l'hypothèse pour réduire la période de l'horloge. Grâce au relâchement des contraintes, l'outil de synthèse est en mesure

Tableau 3.9 Résultats de l'optimisation manuelle du PA pour le MAC16

Paramètres	**PA de base @ 2.75**	**PA optimisé @ 2.80**	**Variation (%)**
Surface totale (μm^2)	31775.52	26834.12	- 15.6
Période (ns)	2.73	2.7	- 1.1
Latence maximale (ns)	7.18	8.27	+ 15.2
Puissance moyenne (mW)	26.7	20.9	- 21.7
Puissance crête (mW)	76.8	64.7	- 12.1
mW/MHz moyen	0.0729	0.0564	-22.6

de générer un circuit 15.6% plus petit en surface pour la même période au prix d'une latence 15.2% plus élevé. Cela est dû à cause, d'une part, de la plus petite taille des cellules du chemin combinatoire et, d'autre part, des imperfections de l'optimisation manuelle.

Au niveau de la puissance moyenne, la tendance se maintient avec une réduction globale de 21.7% de la consommation. On observe aussi une diminution de 22.6% de la puissance moyenne consommée par unité de fréquence, ce qui implique que l'optimisation donne un circuit beaucoup plus efficient. Pour comprendre la source de ces diminutions, les résultats de la distribution de puissance moyenne sont présentés dans le tableau 3.10.

Tableau 3.10 Différence de distribution des puissances moyennes du MAC16 pour l'optimisation manuelle du PA

Puissance moyenne	**PA de base @ 2.75 ns**	**PA optimisé @ 2.80 ns**	**Variation (%)**
Réseau de l'horloge (mW)	11.08	7.94	- 28.3
Registres (mW)	1.56	1.51	- 3.2
Combinatoire (mW)	14.06	11.46	- 18.0

Le tableau 3.10 montre que la puissance du réseau de l'horloge diminue de 11.08 mW à 7.94 mW, soit une réduction de 28.3%, et que la puissance moyenne associée à la logique combinatoire diminue de 14.06 mW à 11.46 mW, soit une réduction de 18%. Ces résultats confirment que l'optimisation des tampons permet de diminuer la puissance consommée par le réseau de l'horloge, et, qu'en plus le relâchement des contraintes permet de réduire la puissance consommée du chemin combinatoire.

Ces résultats se contrastent avec les résultats du MAC8 présenté plus haut par une diminution moins importante de la consommation de puissance du réseau de l'horloge. Cette situation s'explique simplement par le fait que l'optimisation manuelle varie plus facilement d'une implémentation à l'autre contrairement à une implémentation par ordinateur.

Ces améliorations sont possibles pour la stratégie PA car, contrairement à la synchronisation synchrone et au WP, la structure du réseau de l'horloge est plus tolérante aux variations parasitiques. Cela est principalement dû à ses propriétés asynchrones BD, soit que la donnée est jumelée au contrôle du circuit, ce qui implique que le seul objectif du réseau de l'horloge est de compenser le délai combinatoire de la logique.

Il est à noter que la stratégie PA avec emprunt d'horloge optimisée manuellement est donc utilisée pour les comparaisons dans la prochaine section parce qu'elle génère les meilleures performances pour cette stratégie. Son appellation est abrégée comme étant la stratégie PA optimisé.

3.4 Résultats avec performances maximales du PA optimisé

Une des caractéristiques observées de la synchronisation PA optimisé est qu'une période minimale de 2.11 ns et de 2.70 ns était atteinte pour le pire coin d'analyse des circuits du MAC8 et du MAC16 respectivement, et ce avec seulement trois étages de registres.

En comparaison, cette période était impossible à atteindre par son équivalent synchrone sur trois étages qui utilisait la fonction de recadencement, ce qui a forcé l'ajout d'un étage de registres de pipeline pour atteindre la période ciblée ($L = 4$).

De plus, les résultats de la stratégie WP ayant un nombre de cycles d'horloge $K = 0$ et une latence d'entrée $\Delta_I = 0$ sont présents, car le WP atteignait facilement la période ciblée par la stratégie PA avec emprunt d'horloge tout en proposant une structure similaire.

Comme mentionné, la stratégie asynchrone BD avec CE utilisant la méthode LCS améliorée est omise étant donné l'absence de contrôle de la période dû à sa structure autorégulée, ce qui la rend peu compétitive face aux autres stratégies.

Le tableau 3.11 présente les résultats de l'implémentation du MAC8 pour la stratégie synchrone sur quatre étages, pour la stratégie PA optimisé manuellement et pour la stratégie du WP. Les circuits sont conçus pour atteindre la période minimale du PA avec emprunt d'horloge, soit de 2.11 ns pour le pire coin d'analyse.

Tableau 3.11 Performances maximales pseudo-asynchrone du MAC8

Paramètres	**Synchrone,** $L = 4$	**PA optimisé**	**WP,** $K = 0$
Surface totale (μm^2)	10049.63	10822.34	10374.52
Période (ns)	2.11	2.11	2.09
Latence maximale (ns)	7.18	6.11	5.82
Puissance moyenne (mW)	10.3	14.5	10.0
Puissance crête (mW)	44.0	42.2	116.1
mW/MHz moyen	0.0217	0.0306	0.0209

Dans un premier temps, le tableau 3.11 montre que la surface utilisée par chaque stratégie de synchronisation tend vers des valeurs similaires à la stratégie synchrone ayant la plus petite taille. Cela s'explique par le fait que pour quatre étages de pipelines, la stratégie synchrone a une plus grande marge de manœuvre lors de la synthèse. Ce fait s'observe dans les résultats du tableau I-5 placé en annexe sur les performances maximales atteignables par cette stratégie, soit une période de 1.90 ns.

Pour ce qui est du WP, on observe que, bien qu'il ne possède aucun registre intermédiaire, il demeure 3.1% plus gros que le MAC8 synchrone. Dans la mesure où la tendance se maintient pour le MAC16, on peut imaginer que l'absence de registres intermédiaires ne serait pas bénéfique quant à la surface totale du circuit.

Au niveau de la latence, on observe un décrescendo entre de la stratégie synchrone et la stratégie du WP. Cette descente est intuitivement logique, considérant le fait que la quantité de registres diminue entre chaque stratégie. Notons que, bien qu'impactant cette latence, le nombre de registres représente tout de même un facteur parmi d'autres. En effet, le délai de propagation $T_{CQ} \approx 0.50ns$, pour le pire coin d'analyse, ne correspond pas nécessairement à la différence entre chaque stratégie. Il est possible d'affirmer que le surplus serait dû aux différences liées à l'effort de synchronisation, tels le temps de préparation T_{SU}, le temps de maintien T_H ainsi que l'existence de marges non nulles dans les rapports de STA.

Tableau 3.12 Distributions des puissances pour le MAC8 avec performances maximales pour le PA

Paramètres	**Synchrone,** $L = 4$	**PA optimisé**	**WP,** $K = 0$
Réseau de l'horloge (mW)	5.83	9.22	3.65
Registres (mW)	1.11	0.90	0.61
Logique combinatoire (mW)	3.36	4.37	5.79

Au niveau de la puissance, on constate que la tendance se maintient pour ce qui est du PA avec emprunt d'horloge, du fait que, pour une taille équivalente, la puissance moyenne est plus élevée à cause de la consommation du réseau de l'horloge. En effet, malgré les optimisations manuelles,

le tableau 3.12 permet de constater une augmentation de 58.4% de la puissance moyenne dédiée au réseau de l'horloge, soit de 9.23 mW (63.65%) pour la stratégie PA avec emprunt d'horloge contre 5.76 mW (54.58%) pour la stratégie synchrone.

Le tableau démontre également que, malgré leurs similarités, le PA consomme en moyenne 45% plus d'énergie que le WP. Cette constatation semble s'expliquer du fait que sa structure étant similaire à une synchronisation synchrone, l'implémentation de l'arbre de l'horloge par l'outil de PNR est plus optimisée pour le WP. On observe cela en comparant le nombre de tampons d'horloges instanciés par l'outil PNR, soit 104 pour la stratégie PA et 35 pour la stratégie du WP.

Une observation également intéressante est l'inversion de la distribution de puissance entre la stratégie du WP et la stratégie synchrone. En effet, la puissance consommée par le réseau de l'horloge affiche 3.64 mW (36.4%) pour le WP contre 5.83 mW (56.58%) pour le synchrone, tandis que la consommation de la logique combinatoire affiche 5.76 mW (57.6%) pour le WP contre 3.36 mW (32.66%) pour le synchrone.

Cette inversion peut s'expliquer par la différence d'objectif entre les deux stratégies : la stratégie du WP tente de transmettre plusieurs données simultanément, ce qui implique plus de transitions dans le chemin de données, alors que la stratégie synchrone stocke les données dans des registres intermédiaires, ce qui implique une grosse charge capacitive induite sur le réseau de l'horloge.

Il est à noter que la stratégie du WP atteint une puissance crête environ 2.7 fois supérieure par rapport aux autres stratégies, ce qui serait dû au grand niveau de commutation de la logique combinatoire. Finalement, comme les trois stratégies atteignent essentiellement la même fréquence moyenne, le ratio de la puissance moyenne sur fréquence moyenne suit la tendance de la puissance moyenne.

Le tableau 3.13 présente les résultats des stratégies synchrones sur quatre étages, du PA avec emprunt d'horloge et du WP pour le circuit du MAC16 conçu avec une période de référence de 2.7 ns.

Tableau 3.13 Performances maximales pseudo-asynchrone pour le MAC16

Paramètres	**Synchrone,** $L = 4$	**PA optimisé**	**WP,** $K = 0$
Surface totale post-synthèse (μm^2)	26546.55	27501.46	28992.01
Surface des tampons post-PNR (μm^2)	1323.71	2045.93	4735.05
Surface totale post-PNR (μm^2)	27846.11	26834.71	29371.78
Période (ns)	2.70	2.70	2.71
Latence (ns)	9.35	8.27	7.49
Puissance moyenne (mW)	21.4	20.9	22.3
Puissance crête (mW)	94.4	64.7	69.2
mW/MHz moyen	0.0578	0.0564	0.0604

La première observation est que, en réponse à l'augmentation de la taille, la tendance de la stratégie PA avec emprunt d'horloge à générer la plus petite surface totale, démontré par les résultats préoptimisés du tableau 3.5, semble se maintenir.

Cependant, contrairement aux résultats du tableau 3.5, cette diminution de taille ne provient pas du fait que l'outil de synthèse a eu plus de facilité à converger vers un résultat, mais plutôt du fait que l'outil de PNR a été en mesure d'optimiser la surface de la stratégie. En effet, la surface totale de la stratégie synchrone, qui était était de 26546.55 μm^2 suite à la synthèse, a augmenté à 27501.46 μm^2 suite au PNR.

En comparaison, pour le circuit résultant de la synchronisation PA, on a observé une réduction 2.42%, passant de 27501.46 μm^2 post-synthèse à 26834.71 μm^2 post-PNR. Ces résultats impliquent donc que l'outil de PNR est en mesure de redimensionner les cellules plus aisément avec la stratégie PA afin de réduire la surface et la puissance consommée.

Pour ce qui est de la stratégie du WP, sa surface a légèrement augmenté après le PNR, tout comme pour la stratégie synchrone, passant de 28992.00 μm^2 à 29371.78 μm^2. Cependant, l'aspect intéressant du résultat est surtout qu'il confirme la tendance expliquée dans les résultats du MAC8 pour cette même analyse, soit que l'absence de registres intermédiaires implique une plus grande surface totale.

Ce résultat pointe vers le fait que l'absence de registres intermédiaires force une surutilisation de tampons dans le chemin combinatoire pour renforcer l'intégrité du signal, ce qui ne semble pas être la meilleure méthode avec les outils EDA synchrones. Cela découle du fait que la surface utilisée par les tampons dans la stratégie WP est de 4735.05 μm^2, comparativement à la stratégie PA avec emprunt d'horloge où la surface dédiée aux tampons est seulement de 2045.93 μm^2.

Pour ce qui est de la latence, elle suit la même tendance que pour le circuit du MAC8, à savoir que la stratégie synchrone est la plus lente et la stratégie du WP la plus rapide. La stratégie synchrone demeure désavantagée par rapport aux autres stratégies à cause de son besoin d'attendre L fronts montant de l'horloge, peu importe le délai combinatoire des étages logiques, avant que la donnée soit capturée par le registre en sortie. Cela limite sa performance contrairement à la latence des stratégies PA et du WP avec $K = 0$, où ces dernières peuvent être ajustées en fonction du délai combinatoire de chaque étage.

Contrairement au MAC8, l'augmentation de la taille du circuit ne semble pas générer une grande différence au niveau de la puissance moyenne totale consommée pour les trois stratégies, ce qui se reflète aussi dans la puissance par unité de fréquence moyen. L'histoire se complique toutefois lorsqu'on analyse les résultats relatif à la distribution de la consommation de la puissance, présentés dans le tableau 3.14.

Tableau 3.14 Distributions des puissances pour le MAC16 avec performances maximales pour le PA

Paramètres	**Synchrone,** $L = 4$	**PA optimisé**	**WP,** $K = 0$
Réseau de l'horloge	10.63	7.94	5.74
Registres	2.27	1.51	1.18
Logique combinatoire	8.48	11.5	15.73

Dans le tableau 3.14, il est intéressant de noter la façon dont la distribution de puissance tend à se déplacer de façon incrémentale du réseau de l'horloge et des registres vers la logique combinatoire lorsqu'on passe de la stratégie synchrone à la stratégie PA puis à la stratégie du

WP. Tout comme pour la latence, cela s'explique par le fait que le nombre d'étages de registres diminue en décrément entre chaque stratégie.

De plus, l'optimisation effectuée pour le circuit de la stratégie PA du MAC16 s'est révélé plus importante que dans le cas du MAC8, ce qui s'observe dans la différence de tampons d'horloge utilisés, soit 84 pour la stratégie PA contre 64 pour la stratégie du WP. L'augmentation du nombre de tampons pour le WP est principalement due à l'augmentation de la taille du circuit, ce qui a permis de diminuer de plus facilement le nombre de tampons de la stratégie PA vers une quantité équivalente.

Pour ce qui est de la puissance crête, contrairement aux résultats indiqués par le MAC8 à performance maximale, le WP montre une puissance crête similaire à celle du PA par emprunt d'horloge. Cela peut être dû à l'augmentation du nombre de tampons d'horloge, qui cause une plus grande asymétrie au niveau du contrôle, étalant ainsi le pic de courant sur un temps plus long.

Les figures II-1, II-2 et II-3, que l'on trouve en annexe, illustrent les implémentations physiques du MAC16 synchrone, du MAC16 PA optimisé et du MAC16 WP respectivement afin de visualiser les implémentations de l'arbre de l'horloge en fonction de la stratégie de synchronisation.

Enfin, il est important de mentionner un point dont ces comparaisons ne traitent pas : la performance maximale atteinte par la stratégie synchrone sur quatre étages et le WP. En effet, dû à l'équation 1.20, il était impossible de réduire la période du PA en dessous de 2.11 ns pour le circuit du MAC8 et de 2.7 ns pour le circuit du MAC16.

Cela était problématique compte tenu du fait que la période minimale qui pouvait être atteinte par la stratégie synchrone sur quatre étages et la stratégie du WP était de 1.90 ns et 1.77 ns respectivement pour le circuit du MAC8 ainsi que 2.44 ns et 2.25 ns respectivement pour le circuit du MAC16.

Afin d'obtenir des résultats équivalents, il aurait été nécessaire d'ajouter des registres de pipeline supplémentaires pour la stratégie du PA, une tâche qui s'est avérée impossible à cause des

limitations imposées par la nature de haut niveau du code RTL développé. Les résultats pour les performances maximales du MAC8 et du MAC16 sont indiqués en annexe dans les tableaux I-5 et I-6 respectivement.

3.5 Discussion

Plusieurs constatations peuvent être tirées des résultats présentés dans les sections précédentes. Ils sont décrits dans cette section pour chaque stratégie évaluée.

3.5.1 Stratégie asynchrone BD

En tout premier lieu, il est assez clair que la nature autosynchronisée de la stratégie asynchrone BD implémentée à l'aide de la méthode LCS améliorée a été un frein quant à sa performance face aux autres stratégies. En effet, l'absence de contrôle de ce type de synchronisation envers le débit final a eu pour effet de la rendre beaucoup moins attrayante, notamment due au fait qu'elle a seulement fourni la moitié de la performance d'un circuit synchrone pour une surface équivalente.

Aussi, il est clair que la boucle de rétroaction a eu comme effet de limiter les performances du circuit via l'utilisation d'un contrôleur factice. Cependant, étant donné qu'une grande majorité des circuits dépendent de rétroactions, il est irréaliste de croire que l'on devrait uniquement se limiter à un pipeline linéaire lors de l'implémentation d'un design.

De plus, prenant en compte la quantité de travail nécessaire pour implémenter la stratégie utilisant la méthode LCS, il est difficile de justifier son utilisation sachant que la stratégie PA est en mesure de générer des résultats très favorables tout en exploitant les mêmes avantages qu'offre la synchronisation asynchrone BD.

Pour terminer, les avantages liés à la puissance consommée n'ont pas été aussi importants contrairement à ce qui était présenté dans la littérature scientifique. En effet, pour une période deux fois plus grande, on retrouvait une consommation moyenne deux fois plus petite par rapport

à la synchronisation synchrone, ce qui ne pointe vers aucune économie en puissance même si la période était équivalente. Il semble donc qu'en réalité, pour ce type de synchronisation, le plus grand facteur de consommation dépend de la taille des cellules du chemin combinatoire.

3.5.2 Stratégie synchrone

Pour ce qui est de la stratégie synchrone, il est indéniable que son support intrinsèque par les outils EDA rend son expérience de conception beaucoup plus agréable comparativement aux autres stratégies.

Les résultats du STA sont aisément corrélés aux résultats de la GLS post-PNR, contrairement aux autres stratégies. En effet, d'une part, la stratégie asynchrone BD nécessite la mesure de la période moyenne en simulation étant donné l'impossibilité d'utiliser les rapports d'analyse statique pour la calculer. D'une autre part, les stratégies du WP et du PA avec emprunt d'horloge nécessitent une simulation pour déterminer la période minimale compte tenu de leur dépendance aux délais combinatoires minimaux (D_{MIN}) et aux boucles de rétroactions.

Malgré ses avantages liés à sa simplicité, la stratégie synchrone pénalise les performances du circuit au niveau de surface totale, de la puissance moyenne et de la puissance crête lorsque leur complexité augmente, contrairement à la stratégie du PA qui est en mesure d'extraire une meilleure QOR avec les gros circuits.

Notons qu'il n'est pas surprenant que la puissance de crête fût amortie par rapport à ce qui serait attendu normalement pour un circuit synchrone. Cela serait causé par trois facteurs. Le premier est causé par l'utilisation d'une contrainte pour limiter le *fanout* lors de la synthèse (une contrainte constante pour toutes les stratégies). Le deuxième est causé par un délai asymétrique induit par l'algorithme de recadencement lors de la synthèse. Le troisième et dernier est causé par des optimisations du délai asymétrique par l'outil de PNR lors de l'étape de l'optimisation de l'arbre de l'horloge.

Cependant il reste que les autres stratégies ont une meilleure gestion des pics de consommation par leur structure basée sur l'utilisation de délais dans le réseau de l'horloge.

3.5.3 Stratégie du *Wave Pipelining*

Au niveau de la synchronisation WP, le fait de poser le nombre de cycles $K = 0$ donne naissance à une structure intéressante à cause de sa similarité avec la stratégie PA avec emprunt d'horloge, et ce en dépit de son utilisation quasi inexistante dans la littérature scientifique.

En effet, l'utilisation de tampons pour compenser le chemin combinatoire lui donne l'avantage de réduire sa dépendance aux variations PVT en retirant la borne de la période minimale indiquée par la contrainte 1.16. De plus, ses similarités avec la stratégie synchrone lui permettent d'obtenir une consommation moyenne de puissance totale réduite par une meilleure implémentation de l'arbre de l'horloge que la stratégie PA. Finalement, sa capacité à ne pas être limitée par l'équation 1.20 de la stratégie PA avec emprunt d'horloge semble être un avantage pour l'atteinte d'une période et une latence de fonctionnement minimale.

L'absence de registres internes constitue un grand avantage par rapport à la latence du circuit, cependant elle vient aux prix d'une augmentation de la surface totale due à l'utilisation d'une quantité accrue de tampons dans la logique combinatoire afin de rehausser le signal transmis. On peut donc conjecturer qu'une utilisation de registres intermédiaires aurait comme impact de réduire la surface de la logique combinatoire de la même façon que pour la stratégie du PA.

Toutefois, étant donné la nature parallèle des lignes à retard dans cette stratégie, on se retrouverait, d'une part, avec une explosion de tampons pour prendre en compte le délai combinatoire des étages précédents et d'une autre part, avec une difficulté beaucoup plus prononcée quant à la synchronisation des différents étages.

Il est à noter qu'une nouvelle source de problème s'est présentée pour la stratégie du WP avec $K = 0$ en simulation lorsque la fréquence variait. Effectivement, étant donné que la capture de la donnée en sortie se fait après un nombre de cycles défini par l'équation 2.4, une augmentation

de la période avait comme effet de diminuer le nombre de cycles avant que la donnée soit prête à être capturée.

Cette caractéristique est problématique lorsqu'on prend en considération les techniques modernes de variation dynamique de la fréquence, car une modification des attentes au niveau du nombre de cycles affecte toute logique de contrôle déterminant la validité du résultat. Pour éviter ce problème, il faudrait limiter la valeur de $\Delta \ll T_{CLK}$, ce qui est possible uniquement lorsque $K \geq 2$ et non recommandé à cause de sa difficulté d'implémentation (Weste & Harris, 2011).

Finalement, un aspect négligé dans les résultats, mais pouvant grandement réduire les performances se trouve dans les variations des délais supplémentaires associés au séquencement. En effet, étant donné que les circuits sont considérés comme étant idéaux, les variations affectant l'horloge ont été négligées. Cependant, elles réduisent la période maximale de l'horloge dans le cas de la stratégie du WP comme démontré par l'équation 3.1 tiré de l'article de Burleson *et al.* (1998).

$$T_{CLK} \geq (D_{MAX} - D_{MIN}) + (T_{SU} + T_H + 2\Delta_{CLK}) \tag{3.1}$$

Cette négligence a pour effet de rapporter une période théorique supérieure à celle qui serait obtenue en pratique. En effet, utilisant l'estimation que l'asymétrie de l'horloge est égale au dixième de sa période (Mustafa, 2011), il faudrait plutôt utiliser les résultats de la stratégie du MAC16 compilé avec la période minimale de $2.25ns$ indiqué dans le tableau I-6 pour la comparaison du tableau 3.13.

En effet, pour le cas d'une période de 2.25 ns, le délai de synchronisation ajouté à la période de l'horloge serait de $2\Delta_{CLK} = 2 \cdot \frac{1}{10}2.25 = 0.45$ ns, ce qui permettrais d'atteindre une période pratique de $2.25 + 0.45 = 2.70ns$, soit la période cible des résultats présentés pour la stratégie PA avec performance maximale.

Ces raisons expliquent pourquoi on retrouve très peu l'utilisation de la stratégie en industrie malgré son apparence théorique attrayante.

3.5.4 Stratégie du pseudo asynchrone avec emprunt d'horloge

Finalement, en ce qui concerne la stratégie PA avec emprunt d'horloge, les résultats montrent qu'en général, cette stratégie permet de générer un circuit qui a une plus petite surface grâce aux des contraintes assouplies de la stratégie, et ce pour des performances similaires à une stratégie synchrone cadencée. Cet assouplissement permet au concepteur d'effectuer une optimisation en surface ou en latence en fonction des contraintes.

De plus, les circuits qui résultent de l'implémentation de cette stratégie ont tendance à consommer plus de puissance en moyenne lorsqu'ils ne sont pas assez gros, contrairement aux autres stratégies. Cela est causé par un surplus de tampons d'horloges utilisé pour la synchronisation. Rappelons que ces tampons ont la caractéristique de consommer plus de puissance à cause de leur structure cherchant à égaliser le délai des transmissions.

L'utilisation d'optimisation manuelle devient donc une étape critique pour assurer qu'une quantité exacte de tampons soient utilisés afin d'éviter d'affecter négativement les QOR. Toutefois, le fait de devoir manuellement intervenir dans l'implémentation de l'arbre de l'horloge avec le redimensionnement et le retrait des tampons d'horloge est un désavantage à cause de l'expertise devant être acquise par un concepteur. Néanmoins, ces modifications requises sont relativement simples à implémenter, surtout lorsqu'on les compare à la courbe d'apprentissage de la méthode LCS. Il est aussi envisageable de développer une méthode automatisée pour guider l'étape d'implémentation de l'arbre de l'horloge et ainsi obtenir des meilleurs QOR.

Il convie d'expliciter une caractéristique affectant tout autant la stratégie du WP et du PA avec emprunt d'horloge est que les stratégies dépendant de tampons d'horloges pour le contrôle sont assujetties à des problèmes d'électro-migration, comme mentionné dans le guide d'utilisateur de ICC2 (Synopsys, 2023b). Il est recommandé d'utiliser une contrainte d'espacement applicable avec la commande ICC2 *set_clock_cell_spacing*, pour éviter les amas de tampons et ainsi diminuer la densité de courant.

Enfin, grâce au fait que la stratégie est comparable à une synchronisation asynchrone BD, elle hérite de plusieurs caractéristiques positives tout en résolvant certaines caractéristiques négatives. Bien que non évalué dans ce travail, par analyse théorique, on observe une capacité intrinsèque au *clock gating* par le traitement en fonction de la demande, une meilleure modularité par la simplicité de synchronisation ainsi qu'une meilleure résistance aux effets PVT par le jumelage de la donnée au contrôle.

Cette stratégie évite aussi les caractéristiques négatives de la stratégie asynchrone, telles l'absence de contrôle sur le débit ainsi que l'absence de la nécessité d'utiliser des contrôleurs pour synchroniser les éléments séquentiels.

Il est à noter que la stratégie a été évaluée à un niveau modulaire. Cependant, des modules *First In First Out* (FIFO) asynchrones devraient être instanciés aux interfaces afin de gérer automatiquement la synchronisation du module lors d'une implémentation a un niveau systémique.

3.6 Sommaire

Dans ce chapitre, les résultats obtenus suite à l'implémentation de la stratégie synchrone, de la stratégie asynchrone BD, de la stratégie PA avec emprunt d'horloge et de la stratégie du WP ont été comparés et analysés afin d'observer leur impact sur l'implémentation d'un MAC avec des entrées de taille $N = 8$ bits et un accumulateur de taille $C = 19$ bits ainsi que d'un MAC avec des entrées de taille $N = 16$ bits et un accumulateur de taille $C = 38$ bits.

La stratégie PA par emprunt d'horloge a obtenu de meilleurs résultats quant à la surface et la puissance consommée lors de l'augmentation de la taille du circuit par rapport à la stratégie synchrone avec recadencement et du WP ayant un nombre de cycles d'horloge $K = 0$. Elle n'a cependant pas été en mesure d'atteindre les performances minimales obtenues par la stratégie du WP à cause de son budget implicitement défini par l'équation 1.20.

En dernier lieu, la stratégie asynchrone BD s'est démarquée négativement de la masse en proposant des résultats largement inférieurs aux autres stratégies à cause des délais induits par la logique de contrôle.

Le prochain chapitre conclut ce mémoire en effectuant un retour sur le projet, en décrivant les limitations dans la méthodologie et les résultats, puis propose une ouverture pour les recherches futures sur le sujet.

CONCLUSION ET RECOMMANDATIONS

Pour conclure, ce projet de recherche consistait à évaluer différentes stratégies de synchronisation par rapport à la stratégie pseudo-asynchrone avec emprunt d'horloge proposée par Michel Kafrouni (2022).

Pour s'y faire, différentes contributions ont été effectuées afin de générer des résultats. On retrouve une mise à jour et des améliorations à la méthode LCS pour l'implémentation des circuits asynchrones BD, on retrouve le développement de contraintes SDC pour l'implémentation de la stratégie du PA avec les outils de Synopsys et on retrouve le développement de contraintes SDC pour l'implémentation de la stratégie du WP avec les outils de Synopsys.

Quoique les principaux efforts du projet ont été affectés au développement de contraintes pour le circuit du MAC, par son importance dans l'IA, les contributions proposées sont assez génériques pour être appliquées à tout autre circuit cherchant à implémenter une des stratégies documentées dans ce travail de recherche.

Cela est entre autres dû au fait que le circuit ciblé du MAC comporte une structure de traitement directe, par l'entremise du circuit de multiplication, ainsi qu'une structure de traitement par rétroaction, par l'entremise du circuit de l'accumulateur. Ces deux structures permettent l'implémentation des stratégies de synchronisation autant pour les pipelines combinatoires que pour les circuits séquentiels.

Enfin, ce qui a été observé, pour la stratégie PA avec emprunt d'horloge stratégie, est qu'elle permettait d'obtenir des circuits plus petits, plus performants et moins énergivores contrairement aux trois autres stratégies évaluées pour des circuits de grandes tailles.

Il est à prendre en considération que la présente recherche est limitée dans la portée de son analyse en vertu du délai fixé pour effectuer le travail. Ces principales limitations sont ainsi exposées pour comprendre leur impact sur les résultats.

Dans un premier temps, les simulations post-PNR effectuées pour déterminer la puissance moyenne ont été conçues avec la prémisse que le système est stimulé à 100% de ses capacités. Cela ne permet donc pas d'observer les avantages de la stratégie asynchrone et la stratégie PA avec emprunt d'horloge lors des temps morts, par leur disposition intrinsèque à implémenter le *clock gating*.

Ensuite, dans un deuxième temps, la méthode développée pour l'implémentation la stratégie PA avec emprunt d'horloge comporte un aspect ne pouvant pas être automatisé. Lors de l'insertion de la ligne à retard par l'outil de PNR ICC2, une grande quantité de tampons supplémentaires sont ajoutés dans le circuit, ce qui a pour effet d'augmenter la consommation de puissance moyenne du système.

Ces tampons doivent donc être manuellement supprimés par le concepteur pour éviter afin d'obtenir de meilleures performances. Possiblement qu'un lot de contraintes et de commandes peuvent être utilisées à cet escient, cependant il n'a pas pu être découvert à temps.

Aussi, dans un troisième temps, les simulations effectuées pour mesurer la puissance moyenne ne prennent pas en considération la dégradation de l'alimentation due aux grands pics de consommation. En effet, comme les simulations ne sont pas analogiques, il est possible d'affirmer que les stratégies présentant de grandes valeurs pour la puissance crête soient avantagées par rapport à celle la limitant.

En dernier, la plus grande limitation du projet est qu'il idéalise la modélisation des circuits. Cela implique qu'aucune marge de précautions n'a été mise en œuvre dans les contraintes du STA, et qu'il est donc difficile de déterminer, hors une analyse théorique, la robustesse des stratégies par rapport aux variations PVT.

Malgré cela, les résultats obtenus permettent d'ouvrir quelques pistes pour les recherches futures. En premier, la réduction de la technologie de transistor utilisée pourrait être un grand avantage

dans le cadre des stratégies asynchrones, du PA et du WP. En effet, étant donné que le délai des interconnexions est largement supérieur au délai combinatoire dans les technologies modernes, il serait possible de réduire la quantité de tampons d'horloges nécessaires pour ralentir le signal de l'horloge.

Ensuite, comme le projet était spécifiquement dédié à la comparaison des stratégies et non à l'optimisation de ces dernières, il serait envisageable de tenter d'améliorer l'implémentation de l'arbre de l'horloge de la stratégie du PA lors du PNR et d'améliorer la méthode LCS pour permettre le retrait du contrôleur factice. Une suggestion pour la stratégie PA serait d'instancier une chaîne de tampons avec un script via la commande ICC2 *insert_buffer* avant l'étape du placement, ce qui pourrait guider l'outil à construire un réseau de l'horloge optimal.

Finalement, il serait intéressant d'étudier une stratégie hybride entre le PA avec emprunt d'horloge et le WP, étant donné leurs propriétés similaires, afin d'obtenir le meilleur des deux stratégies.

ANNEXE I

TABLEAUX

Tableau-A I-1 Liste des outils utilisés pour le flux de travail

Nom de l'outil	Fonction	Version
Design Compiler	Synthèse logique	2022.12
IC Compiler 2	Placement et Routage	2022.12
PrimeTime	Analyse Statique des Délais	2019.03-SP3
Synopsys VCS	Simulation Post-Placement et Routage	2023.03-SP1
PrimePower	Analyse dynamique de la puissance	2019.12

Tableau-A I-2 Coins d'analyse pour la technologie TSMC 180nm

Coin	Nom	Tension (V)	Temp ($°C$)	Procédé
WCL	Worst, Low Temperature	1.62	-40	SS
WC	Worst	1.62	125	SS
TC	Typical	1.80	25	TT
ML	Maximum Leakage	1.98	125	FF
LT	Low Temperature	1.98	-40	FF
BC	Best	1.98	0	FF

Tableau-A I-3 Comparaison des puissances crêtes du MAC16 asynchrone pour tous les coins d'analyse

Configuration Coin	WCL	WC	TC	ML	LT	BC
LCS de base (mW)	91.1	90.5	135.9	238.7	310.9	236.7
LCS amélioré (mW)	62.0	114.1	143.7	249.6	182.1	237.6

Tableau-A I-4 Comparaison des puissances crêtes du MAC16 synchrone, pseudo-asynchrone et asynchrone pour tous les coins d'analyse

Configuration Coin	**WCL**	**WC**	**TC**	**ML**	**LT**	**BC**
Synchrone, $L = 3$ (mW)	84.9	110.5	192.2	242.6	252.9	262.8
Pseudo-asynchrone (mW)	66.8	67.5	95.2	173.4	160.6	166.2
Asynchrone LCS amélioré (mW)	62.0	114.1	143.7	249.6	182.1	237.6

Tableau-A I-5 Performances maximales pour le MAC8

Paramètres	**Synchrone,** $L = 4$	***Wave pipelining*****,** $K = 0$
Surface totale (μm^2)	12708.01	10534.76
Période (ns)	1.9	1.77
Latence (ns)	6.74	5.00
Puissance moyenne (mW)	13.9	11.9
Puissance crête (mW)	60.8	144.5

Tableau-A I-6 Performances maximales pour le MAC16

Paramètres	**Synchrone,** $L = 4$	***Wave pipelining*****,** $K = 0$
Surface totale (μm^2)	34060.72	34196.83
Période (ns)	2.44	2.25
Latence (ns)	8.93	6.88
Puissance moyenne (mW)	28.2	28.9
Puissance crête (mW)	86.8	124.5

ANNEXE II

FIGURES

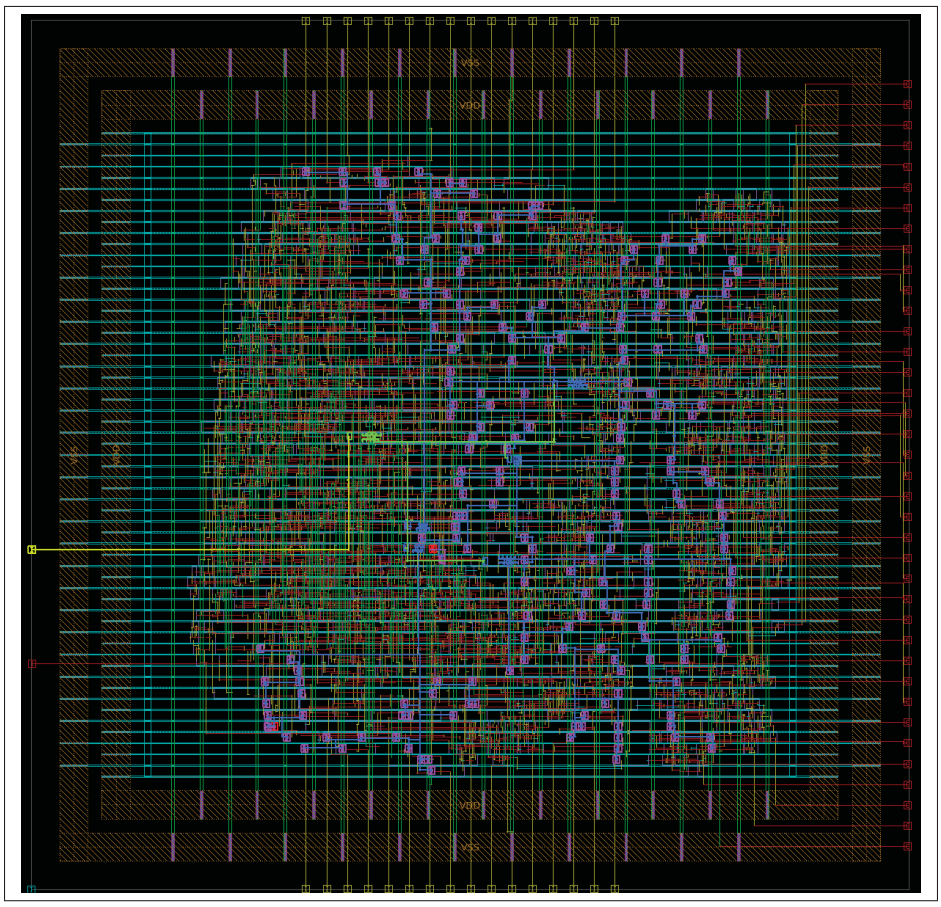


Figure-A II-1 Résultat de l'implémentation physique du MAC16 synchrone recadancé avec $L = 4$ avec l'arbre de l'horloge en surbrillance

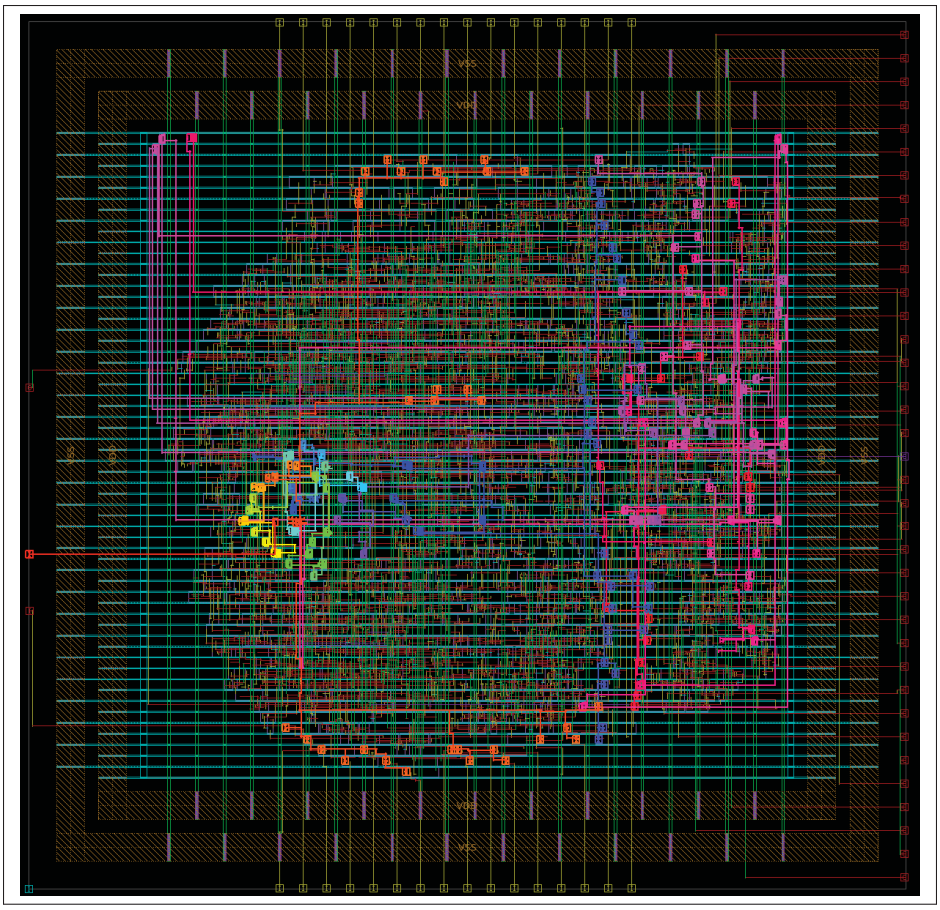


Figure-A II-2 Résultats de l'implémentation du MAC16 PA optimisé avec l'arbre de l'horloge en surbrillance

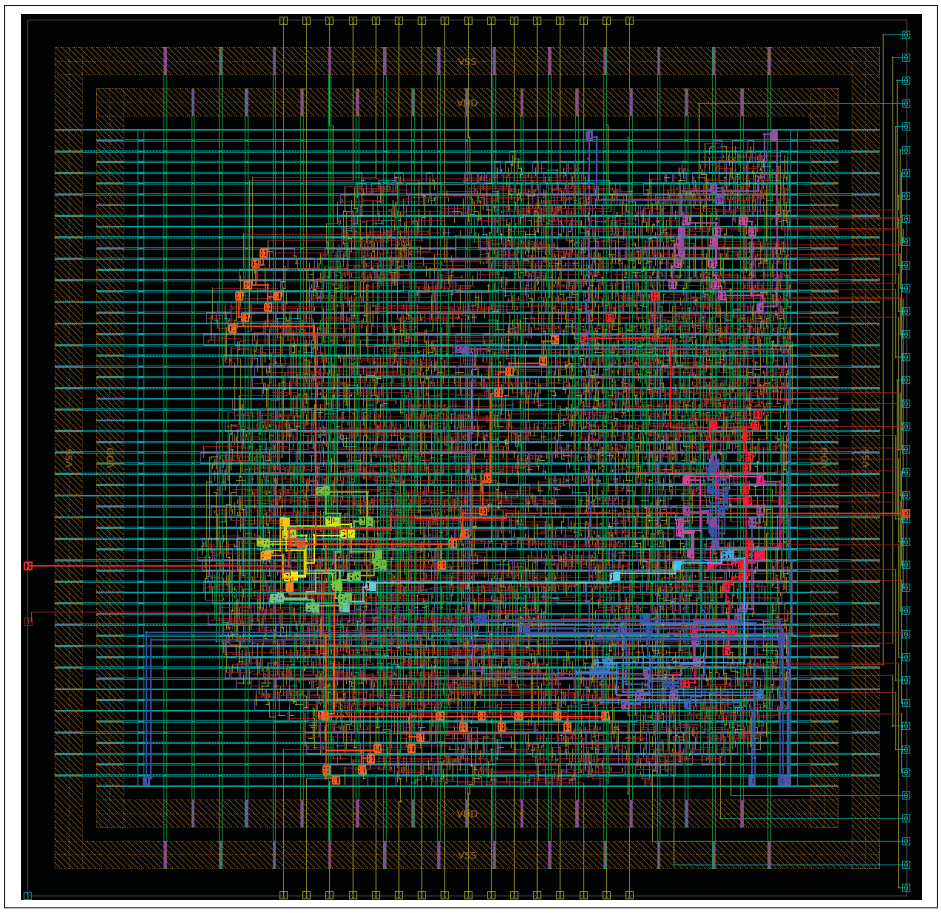


Figure-A II-3 Résultat de l'implémentation du MAC16 WP avec l'arbre de l'horloge en surbrillance

BIBLIOGRAPHIE

A., A. M., K., C. B., B., N., B.Pallavi, P., S., R. & K., M. P. (2019). Survey on Artificial Intelligence. *International Journal of Computer Sciences and Engineering*, 7, 1778-1790. doi : https://doi.org/10.26438/ijcse/v7i5.17781790.

Burleson, W., Ciesielski, M., Klass, F. & Liu, W. (1998). Wave-pipelining : a tutorial and research survey. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(3), 464-474. doi : 10.1109/92.711317.

Burleston, W., Cotton, L., Klaus, F. & Ciesielski, M. (1994). Wave-pipelining : is it practical ? *1994 IEEE International Symposium on Circuits and Systems (ISCAS)*, 4, 163-166 vol.4. doi : 10.1109/ISCAS.1994.409223.

Capra, M., Bussolino, B., Marchisio, A., Masera, G., Martina, M. & Shafique, M. (2020). Hardware and Software Optimizations for Accelerating Deep Neural Networks : Survey of Current Trends, Challenges, and the Road Ahead. *IEEE Access*, 8, 225134-225180. doi : 10.1109/ACCESS.2020.3039858.

Chu, T.-A. (1986). On the models for designing VLSI asynchronous digital systems. *Integration*, 4(2), 99-113. doi : https://doi.org/10.1016/S0167-9260(86)80002-5.

Cotten, L. W. (1969). Maximum-rate pipeline systems. *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, (AFIPS '69 (Spring)), 581–586. doi : 10.1145/1476793.1476883.

Dai, W. & Staepelaere, D. (2002). Useful-Skew Clock Synthesis Boosts Asic Performance. Dans Chinnery, D. & Keutzer, K. (Éds.), *Closing the Gap between ASIC & Custom* (pp. 209-221). United States of America : Kluwer Academic Publishers.

Device, I. R. F. & Systems. (2022). *International Roadmap For Devices and Systems* (Rapport n°2022 Update, More Moore). IEEE.

Emerson, K. (1997). Asynchronous design-an interesting alternative. *Proceedings Tenth International Conference on VLSI Design*, pp. 318-320. doi : 10.1109/ICVD.1997.568097.

Gibiluka, M., Moreira, M. T. & Vilar Calazans, N. L. (2015). A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework. *2015 Euromicro Conference on Digital System Design*, pp. 79-86. doi : 10.1109/DSD.2015.104.

Gimenez, G. (2021). *Concevoir des circuits sécurisés à très faible consommation : une alternative basée sur l'asynchrone*. (Thèse de doctorat, Univerité Genoble Alpes, Grenoble).

Gimenez, G. (2025). Local Clock Set Benchmark [Format]. Repéré à https://gitlab.com/gregoire.gimenez/LocalClockSetBenchmark.

Gimenez, G., Cherkaoui, A., Cogniard, G. & Fesquet, L. (2018). Static Timing Analysis of Asynchronous Bundled-Data Circuits. *2018 24th IEEE Internaional Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 110-118. doi : 10.1109/ASYNC.2018.00036.

Gimenez, G., Simatic, J. & Fesquet, L. (2019). From Signal Transition Graphs to Timing Closure : Application to Bundled-Data Circuits. *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 86-95. doi : 10.1109/ASYNC.2019.00020.

Gray, C., Liu, W. & Cavin, R. (1994). Timing constraints for wave-pipelined systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(8), 987-1004. doi : 10.1109/43.298035.

Huang, Y., Xiao, S., Li, Z. & Yu, Z. (2022). An Asynchronous Bundled-Data Template With Current Sensing Completion Detection Technique. *IEEE Transactions on Circuits and Systems II : Express Briefs*, 69(9), 3904-3908. doi : 10.1109/TCSII.2022.3169819.

Jiang, J.-H. R. & Brayton, R. K. (2006). Retiming and Resynthesis : A Complexity Perspective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(12), 2674-2686. doi : 10.1109/TCAD.2006.882520.

Kim, S., Do, S. & Kang, S. (2017). Fast predictive useful skew methodology for timing-driven placement optimization. *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6. doi : 10.1145/3061639.3062247.

Kim, W. & Kim, Y.-B. (2003). Automating wave-pipelined circuit design. *IEEE Design & Test of Computers*, 20(6), 51-58. doi : 10.1109/MDT.2003.1246164.

Kim, W. J. & Kim, Y.-B. (2005). Wave Pipelined Circuits Synthesis. *2005 IEEE Instrumentationand Measurement Technology Conference Proceedings*, 1, 32-36. doi : 10.1109/IMTC.2005.1604063.

Kra, Y., Noy, T. & Teman, A. (2020). WavePro : Clock-less Wave-Propagated Pipeline Compiler for Low-Power and High-Throughput Computation. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1291-1294. doi : 10.23919/DATE48585.2020.9116524.

Kuzmin, P. A. (2023). Clock Network Design Challenges. *2023 IEEE XVI International Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE)*, pp. 110-117. doi : 10.1109/APEIE59731.2023.10347656.

Laurence, M. (2012). Introduction to Octasic Asynchronous Processor Technology. *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*, pp. 113-117. doi : 10.1109/ASYNC.2012.28.

Lee, W., Sharma, T. & Stevens, K. S. (2016). Path Based Timing Validation for Timed Asynchronous Design. *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 511-516. doi : 10.1109/VLSID.2016.111.

Lee, Y.-L., Tsung, P.-K. & Wu, M. (2018). Techology trend of edge AI. *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1-2. doi : 10.1109/VLSI-DAT.2018.8373244.

Leiserson, C. E. & Saxe, J. B. (1981). Optimizing synchronous systems. *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pp. 23-36. doi : 10.1109/SFCS.1981.34.

Lin, C. & Zhou, H. (2006). An efficient retiming algorithm under setup and hold constraints. *2006 43rd ACM/IEEE Design Automation Conference*, pp. 945-950. doi : 10.1109/DAC.2006.229416.

Lin, S.-C., Zhang, Y., Hsu, C.-H., Skach, M., Haque, M. E., Tang, L. & Mars, J. (2018). The Architectural Implications of Autonomous Driving : Constraints and Acceleration. *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, (ASPLOS '18), 751–766. doi : 10.1145/3173162.3173191.

Lines, A. (2003). Nexus : an asynchronous crossbar interconnect for synchronous system-on-chip designs. *11th Symposium on High Performance Interconnects, 2003. Proceedings.*, pp. 2-9. doi : 10.1109/CONECT.2003.1231470.

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y. & Shi, W. (2019). Edge Computing for Autonomous Driving : Opportunities and Challenges. *Proceedings of the IEEE*, 107(8), 1697-1716. doi : 10.1109/JPROC.2019.2915983.

Manoranjan, J. V. & Stevens, K. S. (2016). Qualifying Relative Timing Constraints for Asynchronous Circuits. *2016 22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 91-98. doi : 10.1109/ASYNC.2016.23.

Meher, P. K. (2016). On Efficient Retiming of Fixed-Point Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4), 1257-1265. doi : 10.1109/TVLSI.2015.2453324.

Muller, D. E. (1955). *Theory of Asynchronous Circuits*. Document inédit, University of Illinois, Illinois.

Mustafa, K. (2011). *Defining Skew, Propagation-Delay, Phase Offset (Phase Error)* (Rapport n°SCAA055). Dallas, Texas 75265 : Texas Instruments.

Nowick, S. M. & Singh, M. (2015). Asynchronous Design—Part 1 : Overview and Recent Advances. *IEEE Design & Test*, 32(3), 5-18. doi : 10.1109/MDAT.2015.2413759.

Nowka, K. & Flynn, M. (1995). System design using wave-pipelining : a CMOS VLSI vector unit. *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, 3, 2301-2304 vol.3. doi : 10.1109/ISCAS.1995.523889.

Peeters, A., te Beest, F., de Wit, M. & Mallon, W. (2010). Click Elements : An Implementation Style for Data-Driven Compilation. *2010 IEEE Symposium on Asynchronous Circuits and Systems*, pp. 3-14. doi : 10.1109/ASYNC.2010.11.

Quinnell, E., Swartzlander, E. E. & Lemonds, C. (2007). Floating-Point Fused Multiply-Add Architectures. *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pp. 331-337. doi : 10.1109/ACSSC.2007.4487224.

Roy, S., Mattheakis, P. M., Masse-Navette, L. & Pan, D. Z. (2014). Evolving challenges and techniques for nanometer SoC clock network synthesis. *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1-4. doi : 10.1109/ICSICT.2014.7021158.

Sharma, T. & Stevens, K. S. (2020). Automatic Timing Closure for Relative Timed Designs. *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 82-87. doi : 10.1109/VLSI-SOC46417.2020.9344096.

Shenoy, N. (1997). Retiming : Theory and practice. *Integration*, 22(1), 1-21. doi : https://doi.org/10.1016/S0167-9260(97)00002-3.

Singh, M. & Nowick, S. M. (2007). MOUSETRAP : High-Speed Transition-Signaling Asynchronous Pipelines. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(6), 684-698. doi : 10.1109/TVLSI.2007.898732.

Singh, R. & Gill, S. S. (2023). Edge AI : A survey. *Internet of Things and Cyber-Physical Systems*, 3, 71-92. doi : https://doi.org/10.1016/j.iotcps.2023.02.004.

Sparsø, J. (2020). *Introduction to Asynchronous Circuit Design*. Technical University of Denmark : DTU Compute.

Stevens, K., Ginosar, R. & Rotem, S. (1999). Relative timing. *Proceedings. Fifth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 208-218. doi : 10.1109/ASYNC.1999.761535.

Stevens, K. S., Xu, Y. & Vij, V. (2009). Characterization of Asynchronous Templates for Integration into Clocked CAD Flows. *2009 15th IEEE Symposium on Asynchronous Circuits and Systems*, pp. 151-161. doi : 10.1109/ASYNC.2009.26.

Sutherland, I. E. (1989). Micropipelines. *Commun. ACM*, 32(6), 720–738. doi : 10.1145/63526.63532.

Synopsys. (2023a). *Design Compiler® User Guide*. Version U-2023.12 [Manuel d'utilisateur]. Lieu d'édition : Synopsys.

Synopsys. (2023b). *IC Compiler™ II Implementation User Guide*. Version V-2023.12 [Manuel d'utilisateur]. Lieu d'édition : Synopsys.

Synopsys. (2023c). *PrimeTime® User Guide*. Version V-2023.12 [Manuel d'utilisateur]. Lieu d'édition : Synopsys.

Synopsys. (2024). *PrimePower and PrimePower RTL User Guide*. Version V-2023.12-SP2 [Manuel d'utilisateur]. Lieu d'édition : Synopsys.

Vireen, V., Seetharaman, G. & Venkataramani, B. (2008). Synthesis techniques for implementation of wave-pipelined circuits in ASICs. *2008 International Conference on Electronic Design*, pp. 1-6. doi : 10.1109/ICED.2008.4786670.

Wang, J., Lin, J. & Wang, Z. (2018). Efficient Hardware Architectures for Deep Convolutional Neural Network. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 65(6), 1941-1953. doi : 10.1109/TCSI.2017.2767204.

Weste, H. N. & Harris, M. D. (2011). *CMOS VLSI Design a Circuits and Systems Perspective* (éd. 4). 501 Boylston Street, Suite 900, Boston, Massachusetts 02116 : Pearson Education Inc.

Williams, T. E. (1994). Performance of Iterative Computation in Self-Timed Rings. Dans Meng, T. H. & Malik, S. (Éds.), *Asynchronous Circuit Design for VLSI Signal Processing* (pp. 17–31). Boston, MA : Springer US. doi : 10.1007/978-1-4615-2794-7_3.

Wirth, N. (1995). *Digital Cicruit Design, An Introductory Textbook* (éd. 1). Berlin : Springer-Verlag Berlin Heidelberg. doi : 10.1007/978-3-642-57780-2.

Wong, D., De Micheli, G. & Flynn, M. (1993). Designing high-performance digital circuits using wave pipelining : algorithms and practical experiences. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(1), 25-46. doi : 10.1109/43.184841.

Wu, H., Chen, W., Su, Z., Wei, S., He, A. & Chen, H. (2019). A method to transform synchronous pipeline circuits to bundled-data asynchronous circuits using commercial EDA tools. *2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pp. 1-2. doi : 10.1109/EDSSC.2019.8754234.

Wu, H., Su, Z., Zhang, J., Wei, S., Wang, Z. & Chen, H. (2021). A Design Flow for Click-Based Asynchronous Circuits Design With Conventional EDA Tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(11), 2421-2425. doi : 10.1109/TCAD.2020.3038337.

Zhang, J., Wu, H., Chen, W., Wei, S. & Chen, H. (2021). Design and tool flow of a reconfigurable asynchronous neural network accelerator. *Tsinghua Science and Technology*, 26(5), 565-573. doi : 10.26599/TST.2020.9010048.

Zhang, X. & Sridhar, R. (1994). Synchronization of wave-pipelined circuits. *Proceedings 1994 IEEE International Conference on Computer Design : VLSI in Computers and Processors*, pp. 164-167. doi : 10.1109/ICCD.1994.331880.

Zhou, Y. (2022). Investigation of asynchronous pipeline circuits based on bundled-data encoding : Implementation styles, behavioral modeling, and timing analysis. *Tsinghua Science and Technology*, 27(3), 559-580. doi : 10.26599/TST.2021.9010089.