Titre: Title:	Conception d'un module de synchronisation pour l'intégration à l'échelle de la tranche de routeurs de communication
Auteur: Author:	Huu The Phiet Nguyen
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Nguyen, H. T. P. (2005). Conception d'un module de synchronisation pour l'intégration à l'échelle de la tranche de routeurs de communication [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7654/

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UN MODULE DE SYNCHRONISATION POUR L'INTÉGRATION À L'ÉCHELLE DE LA TRANCHE DE ROUTEURS DE COMMUNICATION

HUU THE PHIET NGUYEN
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2005

Your file Votre référence
ISBN: 978-0-494-16823-3
Our file Notre référence
ISBN: 978-0-494-16823-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONCEPTION D'UN MODULE DE SYNCHRONISATION POUR L'INTÉGRATION À L'ÉCHELLE DE LA TRANCHE DE ROUTEURS DE COMMUNICATION

Présenté par NGUYEN Huu The Phiet

En vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

A été dûment accepté par le jury d'examen constitué de:

M. François-Raymond Boyer, Ph.D., Président

M. Yvon Savaria, Ph.D., Membre et Directeur de Recherche

M. Claude Thibeault, Ph.D., Membre

Remerciements

Tout au long de ma maîtrise, j'ai eu l'assistance, le support moral et le support matériel de bien des amis, collègues de travail, de mon directeur de projet ainsi que la société HyperChip et la Société Canadienne de Microélectronique (CMC).

En premier lieu, j'aimerais remercier M. Yvon Savaria pour avoir accepté de me diriger dans les travaux, et pour tous les conseils et suggestions pratiques qu'il ait pu me donner durant nos nombreuses réunions. J'aimerais remercier la société HyperChip qui m'a offert l'occasion de travailler sur un projet de recherche intéressant.

Je me dois aussi remercier la Société Canadienne de Microélectronique pour son support matériel et technique menant à la réalisation de la puce d'évaluation.

J'aimerais remercier particulièrement Bernard Antaki, qui m'a offert des conseils précieux en rédaction. J'aimerais aussi souligner le travail de Martin Paré, technicien au Groupe de Recherche en Microélectronique de l'École Polytechnique de Montréal, pour son dévouement dans la réalisation de la carte adaptateur de la puce d'évaluation.

Résumé

Dans un système de communication intégré à l'échelle de la tranche, des données sont envoyées d'un transmetteur à un récepteur, sur de très longues lignes de transmission. Ces données peuvent prendre plus d'un cycle d'horloge pour arriver au récepteur. Divers problèmes tels que des gradients de température, des variations de la tension d'alimentation des répéteurs, des variations des paramètres du procédé de fabrication, ainsi que les interférences électromagnétiques, font en sorte que les signaux transmis de l'émetteur vers le récepteur présentent de grand biais de synchronisation qui peuvent compromettre l'intégrité des données reçues. De plus, dans le système envisagé, bien que le récepteur et l'émetteur aient chacun leurs propres horloges locales de même fréquence, la relation de phase entre ces horloges est arbitraire. Ceci complique la réception des données et requiert une nouvelle méthode de synchronisation robuste aux dérives et aux gigues sur les données, tout en permettant de maintenir une latence fixe pendant la séquence de transmission.

Pour résoudre ce genre de problème, la compagnie HyperChip a proposé une méthode de synchronisation pour recevoir des données du transmetteur et pour synchroniser les données à l'horloge locale du récepteur. L'idée repose sur le fait que les données sont envoyées au récepteur, accompagnées de l'horloge locale du transmetteur.

Le travail consiste à valider la méthode proposée par HyperChip, par calcul théorique et par simulation comportementale. Constatant que la solution proposée comporte des limitations, notamment une latence variable ou des pertes de données sous certaines conditions, et considérant que ces limitations sont inacceptables pour l'application, des méthodes ont été proposées pour

rendre le système plus robuste et plus fiable. Parmi les méthodes proposées, la solution la plus prometteuse a été retenue après des analyses poussées. Cette méthode permet de tolérer les variations de délai dans les interconnexions, et les biais de synchronisation entre les données, ainsi que ceux observés entre les données et l'horloge. La tolérance obtenue atteint nominalement T/4, et peut aller jusqu'à T/2 sous certaines conditions, tout en gardant une latence minimale et fixe. Finalement, le module de synchronisation a été implanté dans une puce d'évaluation. Cette puce permet de valider le fonctionnement et l'efficacité de la solution proposée.

La nouvelle architecture du module de synchronisation est présentement en instance de brevet pour la compagnie HyperChip.

Abstract

In a wafer scaled integrated communication system, the data, sent from the emitter along very long transmission line, can take more than one clock cycle to arrive at the receiver. Various problems such as temperature gradients, voltage drops, process corners, noise and crosstalk can skew signals traveling such a long distance, and thus, cause signal integrity problem at the receiver end. In our case, even though both emitter and receivers have the same clock frequency, the phase shift between these clocks are random and not known in advance. This adds even more complexity to the data recovery scheme and research for a new synchronization method is needed. The new synchronization scheme is required to be very tolerant on short term and long term jitter that exists on the transited data while sustaining a fixed latency during transmission.

HyperChip proposed a new synchronization scheme using a different way of sending emitter's clock along with data and a new way of synchronizing data on receiver's local clock.

At first, our work is to validate the synchronization scheme proposed by HyperChip, by mathematical equations and then by behavioral simulations. As the proposed solution exhibits some limitations such as variable latency or data loss under some particular operating conditions, and considering that those limitations are not acceptable for our application, other methods are then proposed and examined in order to enhance the system and make it more robust and efficient. From the newly proposed synchronization schemes, the most promising one is selected after further analysis. The proposed method offers better tolerance to delay variations due to longs interconnects. It also supports bigger skews on signals of the same data bus as well as bigger clock-data skew. The nominal supported skew is then T/4, and it could be up to T/2

under some conditions, while sustaining a small and fixed latency. Finally, the selected synchronization scheme is implemented in an evaluation chip. This integrated circuit will be used to validate the concept and to measure the performance of the new proposed solution.

Currently, the proposed architecture is being submitted as a patent pending for HyperChip.

Table des matières

Liste des figures

Liste des tableaux

Liste des sigles et abréviations

CDR	Clock and Data Recovery
CMC	Canadian Microelectronics Corporation
DRC	Design Rule Check
DSM	Deep Sub Micron
FSM	Finite State Machine
GALS	Globally Asynchronous Locally Synchronous
GDS	Fichier de format binaire, utilisé par les fonderies pour représenter les dessins de masque
GRM	Groupe de Recherche en Microélectronique
LVS	Layout Versus Schematic
MCM	Multi-Chip Module
PLL	Phase Locked Loop
SOC	System On Chip
T	Période ou durée d'un cycle d'horloge
T/2	Demi-période d'horloge
T_{hold}	Temps de maintien des bascules D
T_{su}	Temps de pré-positionnement des bascules D
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WSI	Wafer Scale Integration

Liste des annexes

Introduction

Dans le cadre des recherches sur les systèmes sur puce pour l'intégration à l'échelle de la tranche, la société HyperChip expérimente une nouvelle méthode de synchronisation pour résoudre les problèmes reliés à la transmission sur de long fils d'interconnexions.

Dans une application spécifique, les données sont envoyées sur de grandes distances par de très long bus de données qui sont communs à plusieurs récepteurs, comme le montre le modèle de communication ci-dessous.

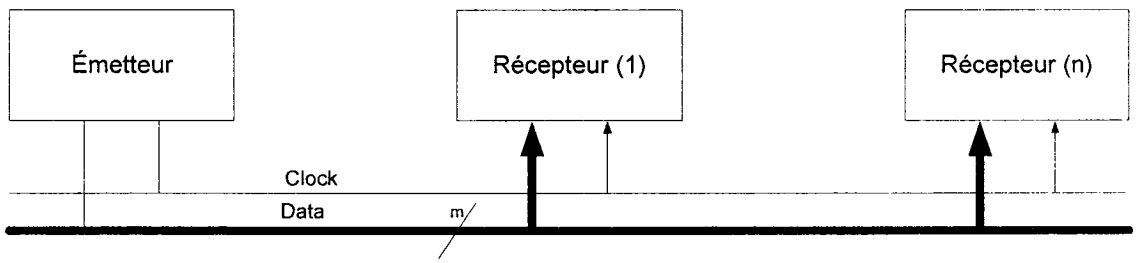

Figure 1. Modèle de communication globale

Dans cette configuration, l'émetteur ainsi que chacun des récepteurs possèdent leurs propres horloges locales. Les déphasages entre l'horloge locale de l'émetteur et chacune des horloges locales des récepteurs sont différents et ne sont pas connus d'avance. De plus, les variations dans les délais de propagation des longues interconnexions entre l'émetteur et les récepteurs introduisent des biais de synchronisations entre les données.

La société HyperChip a initialement proposé une méthode de communication qui, contrairement à d'autres techniques de design synchrones ou asynchrones connus, n'utilise pas des fréquences élevées pour le sur-échantillonnage. La solution proposée n'ajoute pas d'interconnexions additionnelles pour l'encodage ou des signaux pour le support d'un protocole de communication (hand-

shaking). De plus, il est important de minimiser les grandes latences vues dans les systèmes asynchrones avec protocoles de liaison.

Notre travail commence par la détermination des contraintes, des besoins et des problèmes spécifiques à la communication dans les systèmes sur puce et à l'intégration à l'échelle de la tranche. Notre premier objectif sera donc d'analyser et de valider un design proposé par HyperChip.

Dans un premier temps, nous étudions la proposition de la société HyperChip tout en examinant ce que la littérature nous propose.

Le système proposé par HyperChip est d'abord modélisé mathématiquement et ensuite analysé dans le deuxième chapitre. Parallèlement, nous faisons part des résultats des simulations fonctionnelles qui sont basées sur le modèle en VHDL du système proposé par HyperChip. Ces simulations fonctionnelles ont pour but de confirmer sa fonctionnalité et de mesurer ses performances ainsi que ses limites. À la lumière de ces analyses et de ces résultats de simulation, nous présentons les optimisations possibles avec le système original ainsi que les conséquences qui en découlent.

Constatant les limitations de la solution proposée par la société HyperChip, d'autres propositions ou variantes sont suggérées pour rendre le système plus robuste et fiable.

Ainsi, au chapitre suivant, une des propositions en apparence prometteuse mais non retenue est présentée. Une brève description de son principe de fonctionnement et une analyse sommaire sont présentées. Cette analyse montre la raison pour laquelle la solution n'a pas été retenue et elle met en évidence l'avantage de la proposition finale adoptée.

La solution la plus prometteuse est retenue et présentée dans le quatrième chapitre. Une description de son principe de fonctionnement est présentée, accompagnée de schémas blocs du système. Nous introduisons également de nouvelles spécifications et restrictions. La nouvelle solution est ensuite modélisée mathématiquement et elle fait l'objet d'une analyse théorique pour démontrer l'atteinte de la performance recherchée.

Ensuite, le cinquième chapitre présente le design détaillé de l'architecture choisie telle que nous l'avons implantée dans la puce d'évaluation. Nous y décrivons les circuits logiques et nous énonçons les limites d'opération qui découlent des choix faits lors de l'implantation de certaines parties.

Finalement, au dernier chapitre, les différents aspects de la validation sont présentés. D'abord, la méthodologie de simulation et de validation fonctionnelle est décrite. Ensuite, le schéma bloc et quelques circuits importants de la puce d'évaluation sont présentés. Il est aussi question de la réalisation d'une carte adaptateur pour pouvoir tester la puce d'évaluation sur les équipements de test disponibles. À la fin du chapitre, nous présentons les résultats des tests effectués.

Chapitre I.
Revue de littérature

La communication intermodules et la synchronisation dans les circuits intégrés sont des sujets qu'on étudie de façon approfondie depuis plusieurs années. Dans les designs de systèmes sur puce (SOC), les problèmes liés à la synchronisation vont en s'accroissant au fur et à mesure des progrès dans la technologie d'intégration, lesquels continuent à réduire les dimensions géométriques bien en dessous d'un micron. Les problèmes de synchronisation sont causés en grande partie par les délais de propagation et les difficultés associées aux très longs fils d'interconnexions. La synchronisation paraît plus problématique lorsqu'on parle de l'intégration directe sur tranche (Wafer scale integration - WSI), car les interconnexions, dont la longueur cause déjà des problèmes dans les grands circuits intégrés, sont encore plus problématiques sur les circuits WSI.

Avant l'ère submicronique, les transistors étaient relativement lents, de sorte que les délais dans les interconnexions étaient beaucoup moins significatifs. Étant donné les progrès dans les technologies d'intégration, les transistors sont maintenant rapides au point que les délais dans les fils sont devenus un facteur déterminant dans la limitation des performances des circuits intégrés [9], [10].

Pour ce qui est des longs fils d'interconnexion, les délais sont très sensibles aux variations des procédés de fabrication et à celles causées par l'environnement, comme les interférences électromagnétiques, la température et les fluctuations dans les tensions de l'alimentation.

Ces variations environnementales affectent de façon arbitraire et inégale, en tout temps, les délais de propagation dans les fils d'interconnexion. Les signaux

envoyés dans ces longs fils possèdent donc des dérives ou gigues. Ces dérives ou gigues sont reconnues pour être responsables de bien des pannes dans les circuits intégrés. Ces pannes sont intermittentes et sont particulièrement difficiles à détecter ou à analyser [9]. Lorsque les dérives ou gigues modifient les signaux importants qui servent à la synchronisation telle que l'horloge, on parle alors de problèmes de synchronisation. Les problèmes de synchronisation, causés par les longs fils, peuvent donc exister dans un même circuit intégré et s'aggravent à plus grande échelle, c'est-à-dire entre les différents circuits intégrés d'un module multipuce (Multichip module – MCM) ou entre les circuits qui composent un WSI.

Les problèmes de synchronisation se divisent en deux catégories: la synchronisation de l'horloge et la synchronisation des données.

La synchronisation de l'horloge est un élément crucial dans le design d'un circuit intégré et il est d'autant plus important lorsque les circuits numériques VLSI (Very large scale integration) ou WSI deviennent de plus en plus complexes et fonctionnent à des fréquences de plus en plus élevées. L'intégration directe sur tranche constitue un grand défi en ce qui concerne le design de l'arbre de distribution de l'horloge [31], [32]. Comme Keezer et Jain le montrent dans leur étude [19], [20], les deux principales difficultés dans la synchronisation de l'horloge sont les suivantes: (1) le signal de l'horloge doit être disponible à un nombre grandissant d'éléments et (2) les éléments qui doivent être synchronisés ensemble se trouvent assez éloignés les uns des autres. La deuxième difficulté provient des problèmes liés aux fils d'interconnexion que nous avons mentionnés précédemment.

On trouve donc dans les ouvrages et dans la pratique plusieurs méthodes de synchronisation de l'horloge qui visent à résoudre l'une ou l'autre, ou les deux difficultés susmentionnées.

Pour remédier au premier problème, la méthode la plus répandue est l'arbre de distribution (buffer tree), y compris l'arbre dit en H [4], [5]. Nigam et Keezer [26] ont comparé, dans une étude, les performances des différentes variations de l'arbre de distribution. Cette étude démontre que la performance des différents modèles d'arbre de distribution dépend en grande partie des applications. Cependant, dans la plupart des cas, un simple arbre en H offrirait un meilleur compromis en ce qui concerne le délai d'insertion, le biais de synchronisation et la puissance consommée par l'arbre de distribution.

Néanmoins, l'arbre en H, lorsque déployé à grande échelle dans le contexte d'intégration directe sur tranche, est assez sensible aux variations dans les procédés de fabrication et à la température, d'où l'écart de phase entre les éléments de même niveau se trouvant dans différentes régions. Cet écart de phase pourrait en fait être grandement touché. Embabi, Brueske [6], [15] et Aguiar, Santos [2] proposent différentes façons d'asservir l'arbre en H pour améliorer davantage le biais de synchronisation dans les WSI. Ils suggèrent ainsi de diviser le système en de plus petites régions isochrones. Il est possible de comparer et de mesurer le biais de synchronisation entre les différents éléments de même niveau, notamment et surtout les derniers niveaux de l'arbre en H qui se trouvent dans différentes régions. Des éléments d'asservissement vont changer de façon dynamique le délai à travers des éléments à délai variable et contrôlé pour assurer un meilleur écart de phase. L'avantage de cette méthode par rapport à l'utilisation étendue des boucles à verrouillage de phase (phase-locked loop – PLL) se situe évidemment au niveau de la surface et de la puissance consommées.

Parallèlement, pour protéger l'intégrité des signaux envoyés dans des fils d'interconnexion, surtout les signaux critiques tels que les horloges, la littérature propose plusieurs techniques de synthèse et de planification pour ces longs fils. Les techniques les plus connues vont de l'optimisation topologique des fils à l'insertion optimale des répéteurs au choix de la largeur des fils et de la grandeur des répéteurs [3], [8], [10], [11], [17], [27]. Ces solutions sont efficaces afin de protéger l'intégrité du signal et d'éviter les variations dans le délai causées par les interférences électromagnétiques. Cependant, l'utilisation de répéteurs augmente la possibilité de produire des variations dans les délais de propagation en raison des variations dans les procédés de fabrication, des variations de température et des fluctuations de la tension de l'alimentation. Ainsi, lorsque les fils d'interconnexion sont extrêmement longs (par rapport à la dimension géométrique des transistors et à la largeur du fil), ces solutions ne sont pas vraiment pratiques ou efficaces lorsqu'on cherche à minimiser les biais de synchronisation entre les terminaisons de l'arbre. À ce sujet, l'asservissement dynamique de l'arbre de distribution de l'horloge tel que proposé par Embabi et Brueske peut s'avérer efficace pour minimiser le biais de synchronisation. De leur côté, Sato, Onozawaa et Matsuda [29] proposent de combiner l'avantage de l'arbre équilibré avec la distribution en maille (mesh) pour réduire cet écart.

En diminuant le biais de synchronisation de l'arbre de distribution de l'horloge, on améliore par le fait même la performance du système. Cependant, la puissance instantanée exigée par le réseau de l'horloge augmente très rapidement lorsque le biais de synchronisation diminue.

On trouve aussi dans la pratique l'utilisation de fils de blindage (shielding) qui s'avèrent efficaces pour protéger l'intégrité des signaux critiques au détriment

de la puissance consommée par le réseau de distribution de l'horloge, car l'utilisation de fils de blindage ajoute considérablement de capacitance.

Par ailleurs, la puissance consommée par le réseau de l'horloge d'un circuit intégré crée de nouvelles inquiétudes: le réseau de distribution de l'horloge est déjà reconnu pour dissiper plus que le quart de la puissance totale [7]. De plus, au fur et à mesure que la technologie progresse en vue de réduire les dimensions géométriques, la puissance consommée par l'arbre de distribution de l'horloge peut atteindre jusqu'à 40 ou parfois 50 % de la puissance totale d'un circuit intégré, malgré l'importance grandissante d'une autre composante de la puissance associée aux courants de fuite [14]. Dans un grand système synchrone, les éléments faisant partie de l'arbre de distribution de l'horloge, comptés par milliers, commutent simultanément et produisent des demandes instantanées de courant importantes dans l'alimentation locale. Ces demandes sont suffisamment importantes, pour affecter les délais de propagation des circuits logiques ou des répéteurs qui se trouvent à proximité. Cette puissance instantanée, dans la plupart des cas, est de 3 à 5 fois la puissance moyenne du circuit intégré.

Le recours à la transmission différentielle de l'horloge ou de signaux critiques gagne donc en popularité dans les systèmes à très haute performance car la transmission différentielle a l'avantage de réduire considérablement les problèmes de bruits induits par couplages capacitives ou couplages inductives.

Étant donné les désavantages associés au modèle synchrone, le design asynchrone suscite beaucoup d'intérêt [12]. Depuis plusieurs années, les recherches sur les systèmes asynchrones se sont multipliées.

Dans une analyse exhaustive, Afghahi et Svensson [1] ont mis en évidence les inconvénients et les avantages du système asynchrone par rapport au système synchrone.

Les techniques de design asynchrone sont très connues et sont déjà utilisées pour transmettre les données sur de longues distances ou entre différents domaines d'horloge. Les techniques de design asynchrone peuvent être divisées en deux catégories générales: (1) les systèmes totalement asynchrones ou «self-timed» et (2) les systèmes GALS (globally asynchronous, locally synchronous).

Dans les systèmes GALS, les modules ont leur propre horloge locale et la communication intermodules se fait de façon asynchrone. Les signaux externes et asynchrones sont récupérés par les synchroniseurs ou circuits de recouvrement de l'horloge (CDR) (clock and data recovery). Les recherches sur les systèmes GALS présentent plusieurs propositions [13], [25].

Le modèle classique doté de deux bascules sensibles aux niveaux (latchs) (à polarité opposée) ou bascules sensibles aux transitions reste la méthode la plus connue et elle est largement utilisée dans la pratique courante. Il existe aussi des protocoles de communication asynchrones basés sur le modèle appelé pipeline de synchronisation [30].

Une autre technique parmi les plus connues est la synchronisation obtenue par l'établissement d'une liaison ou hand- shaking basée sur les différents modèles de tampon de synchronisation (fifo ou premier entré, premier sorti) et des signaux de liaison [18], [21], [22], [23]. Dans cette catégorie, les modèles les plus populaires sont à 2 ou 4 phases. Suivant cette méthode, des signaux de contrôle reliant l'émetteur et le récepteur permettent un dialogue entre les deux

parties: le récepteur doit avertir l'émetteur s'il est prêt à recevoir ou s'il n'est plus en mesure de recevoir des données, ou encore si les données sont bien reçues.

Cette méthode de synchronisation obtenue par l'établissement d'une liaison, bien connue pour synchroniser les données entre les différents domaines de l'horloge, ne fonctionne que lorsque le déphasage entre les domaines de l'horloge est connu d'avance et ne varie pas dans le temps. Par ailleurs, les communications asynchrones basées sur l'établissement d'une liaison, en plus de présenter une latence importante, ne sont plus efficaces lorsque la distance qui sépare les deux parties est grande. En effet, les protocoles de liaison posent un problème aussitôt que les signaux de liaison prennent plus d'un cycle pour qu'un signal se rende au récepteur, ce qui est facilement le cas des WSI de haute performance.

Une autre pratique courante est l'utilisation de fréquences d'horloge locale, très élevées par rapport à la fréquence d'opération du module. Ces hautes fréquences peuvent être utilisées pour échantillonner les données. Les fréquences de sur-échantillonnage entrent ainsi rapidement dans l'espace RF et l'utilisation de telles fréquences n'est pas sans problème [28]. C'est le cas de la transmission sérielle de données, qui est aussi une des méthodes de synchronisation les plus connues [33] [35].

Il existe aussi l'encodage comme technique de synchronisation de données. Ces techniques protègent bien l'intégrité des signaux lorsque ces derniers sont transmis dans un environnement bruyant, car ils offrent la possibilité de faire des corrections d'erreur de transmission. Le défaut d'une telle technique est l'augmentation du nombre d'interconnexions.

Par ailleurs, Yun et ses collègues suggèrent un autre modèle avancé de GALS [36], [37]. Ils proposent d'étirer ou alors d'arrêter et de repartir l'horloge locale pour éviter tout problème de métastabilité. Dans leur implantation, Yun et ses collègues utilisent un élément mutuellement exclusif (un élément mutuellement exclusif est un circuit logique qui laisse passer une seule requête parmi celles reçues et ce, sur une base de premier arrivé, premier servi) comme moyen d'intervention pour faire la séparation temporelle entre les fronts de l'horloge et les transitions des signaux globaux reçus. L'élément mutuellement exclusif (ME) forme avec une chaîne d'inverseur la boucle d'oscillation pour activer l'horloge locale. L'horloge locale est alors obtenue à la sortie de l'élément mutuellement exclusif et se trouve donc étirée lorsque ME arrête et repart la chaîne à délai au besoin, pour que les transitions de l'horloge et celles des données n'entrent pas dans les conditions de métastabilité.

Cependant, à cause des variations dans les procédés de fabrication et celles causées par l'environnement, l'utilisation de la chaîne à délai ne permet pas toujours d'obtenir l'horloge locale avec la fréquence désirée. Ainsi, Moore, Taylor, Cunningham et Mullins [25] sont allés plus loin et propose que la chaîne à délai utilisée par Yun soit autocalibrée par rapport à une horloge de précision que génère un oscillateur cristal. Le calibrage est possible grâce à l'utilisation d'éléments de délai contrôlables dans la chaîne formant la boucle d'oscillation qui active l'horloge. Après l'autocalibrage initial, l'horloge provenant de la chaîne à délai peut être arrêtée et repartie au besoin tout en restant calibrée à la fréquence désirée.

De son côté, Kessels, Peeters, Wielage et Kim [23] suggèrent une autre méthode de synchronisation GALS qui, contrairement au modèle GALS proposée par Yun, ne recourt pas à la chaîne d'oscillation en boucle fermée. La chaîne à délai est donc ouverte pour faire apparaître, à l'interface du module,

une horloge entrante et une horloge sortante. Ces deux horloges sont alors considérées comme les signaux de liaison et elles peuvent être utilisées par les circuits de synchronisation suivant un protocole de liaison classique. Dans cette configuration, la synchronisation de l'horloge est alors simplifiée par l'utilisation de simples éléments C. Aucune autre forme d'intervention ne sera nécessaire, sauf dans les cas où les ressources telles que les bus de données sont partagés.

Dans ce chapitre, nous avons survolé différentes méthodes de synchronisation qu'on trouve soit dans les ouvrages ou la pratique industrielle. Elles représentent l'aboutissement de plusieurs années de recherches afin de résoudre les différents problèmes liés à la synchronisation.

Au prochain chapitre, nous allons présenter et analyser la solution proposée par HyperChip en vue de résoudre le problème de biais de synchronisation entre les données et l'horloge ainsi que ceux dus aux longs fils d'interconnexion.

Chapitre II.
Solution proposée par la société HyperChip

2.1. Introduction

Dans ce chapitre, nous exposons d'abord le principe de fonctionnement de la méthode de synchronisation proposée par la société HyperChip. Cette présentation est appuyée par une architecture détaillée et exprimée sous la forme de diagrammes bloc du système.

Le système sera ensuite modélisé mathématiquement et nous allons en identifier les chemins critiques. Chacun des chemins critiques ainsi identifiés sera analysé pour démontrer la fonctionnalité et pour déterminer les limites théoriques. Des propositions pour diverses optimisations seront aussi analysées pour trouver les nouvelles limites théoriques du système.

Notons que dans toutes les analyses qui suivent, un rapport cyclique de 50% est assumé pour simplifier les équations. Des analyses avec des rapports cycliques différents de 50% seraient plus réalistes et elles affecteraient les performances ou limites. Cependant elles alourdiraient considérablement les modèles mathématiques et rendraient difficile la compréhension et les analyses. De plus, sans mesurer sur un vrai système ou un prototype durant une vraie opération, il est difficile de prédire avec exactitude le rapport cyclique à utiliser pour les analyses ou pour les simulations. D'autre part, il existe des techniques simples pour produire des horloges dont le rapport cyclique est de 50%.

2.2. Description et principe de fonctionnement

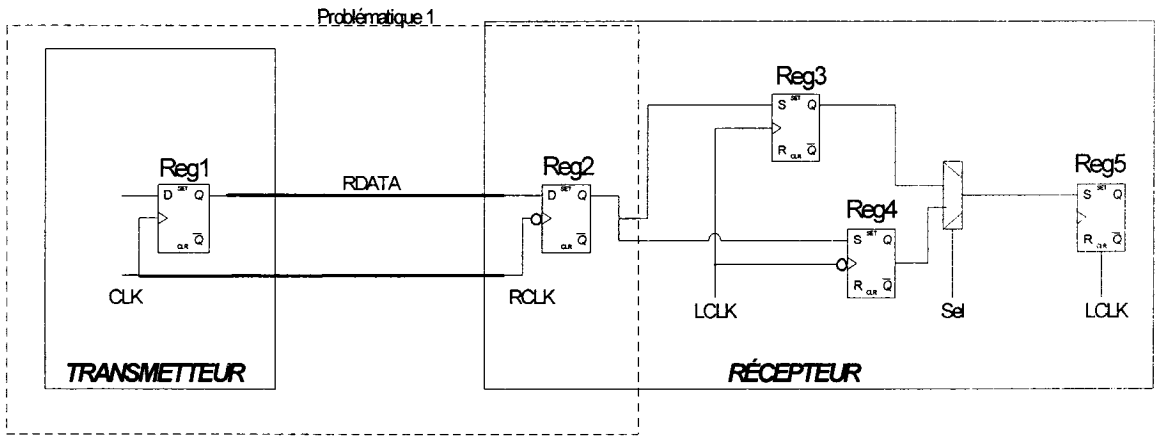

Figure 2.1. Schéma du circuit proposé par HyperChip

Le système proposé par la société HyperChip est présenté à la figure 2.1. Dans ce système, le transmetteur fonctionne sur une horloge locale, appelée CLK. Le récepteur possède sa propre horloge locale LCLK. Ces deux horloges ont la même fréquence, mais leur déphasage est supposé inconnu.

Au transmetteur, les données sont générées avec le front montant de CLK avant d'être envoyées au récepteur. L'horloge CLK est aussi envoyée au récepteur.

Après un certain délai de propagation, le signal CLK envoyé par le transmetteur est reçu au récepteur comme étant RCLK. Un contrôleur (non montré) va établir la relation de phase entre RCLK et l'horloge locale LCLK.

Arrivées au récepteur, les données sont d'abord resynchronisées sur le front descendant du signal RCLK. Ces données resynchronisées sont ensuite échantillonnées sur les deux fronts de l'horloge LCLK. Selon le déphasage déterminé, le module de contrôle sélectionne l'échantillon qui respecte le mieux

les contraintes de pré-positionnement et les conditions de maintien pour éviter les problèmes de métastabilité. Finalement, les données sont resynchronisées sur l'horloge locale LCLK.

Nous avons identifié deux chemins critiques. Le premier, identifié comme étant la problématique 1 sur la figure 2.1, est associé au chemin qui relie le registre de sortie du transmetteur, synchronisé sur CLK, au registre d'entrée de l'émetteur, synchronisé sur le front descendant de RCLK. Le second chemin critique se trouve entièrement dans le récepteur, entre le premier registre d'entrée et les deux registres de rééchantillonnage qui fonctionnent sur les deux fronts de l'horloge locale LCLK.

Ci-dessous, en utilisant des modèles mathématiques, nous allons procéder à l'analyse de la tolérance au biais de synchronisation des deux chemins critiques mentionnés.

2.3. Tolérance au biais de synchronisation Transmetteur - Récepteur

2.3.1. Analyse

Le premier chemin critique, identifié ci-dessus comme la problématique 1 dans le système proposé par la société HyperChip est reproduit à la figure 2.2

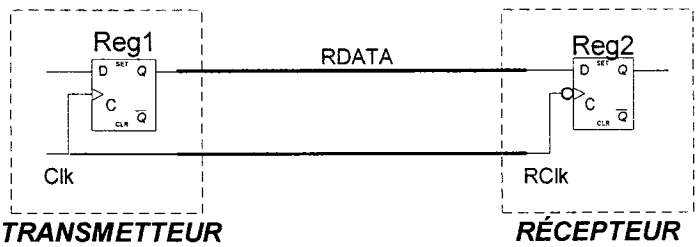

Figure 2.2. Chemin critique 1 - Connexion transmetteur - récepteur

Soit,

$T_{Reg2/D}$, temps d'arrivée des données au port D du Reg2 au récepteur,

$T_{Reg2/C}$, temps d'arrivée du front montant de l'horloge au port C du Reg2 au récepteur;

D_{CQ} , le délai de propagation de C à Q d'une bascule au transmetteur,

D_L , délai associé à la ligne de transmission dans les conditions idéales (sans bruit, sans diaphonie, pas d'effet gradient de température, pas d'effet de gradient de tension d'alimentation...);

T_{Clk} , temps d'arrivée du front montant de l'horloge distante au transmetteur, utilisé comme temps de référence;

T_{ClkN} , temps d'arrivée du front descendant de l'horloge distante au transmetteur tel que $T_{ClkN} = T_{Clk} + T/2$.

Dans le développement qui suit, nous supposons que l'horloge Clk est parfaite, c'est-à-dire sans gigue.

Dans un environnement réel, le délai de transmission de l'horloge distante et celui des données peuvent subir de façon indépendante des déviations par rapport au délai idéal D_L.

Pour modéliser ce phénomène, nous allons donc appeler:

G_{Clk} , fluctuations (gigue) sur le temps d'arrivée du signal RClk,

G_D , fluctuations (gigue) sur le temps d'arrivée au récepteur des données,

$|G_{Clk}|$, module de G_{Clk},

$|G_D|$, , module de G_D,

Commençons par définir les fluctuations (gigue) sur le temps d'arrivée au récepteur des données G_D tolérable par ce système.

Nous savons qu'en réalité, sous les conditions non-idéales, l'horloge transmise va prendre un temps différent de D_L, le délai de transmission idéal, à cause de la gigue, soit:

$$D_{Clk} = D_L \pm |G_{Clk}|$$

pour arriver au récepteur.

De façon similaire, les données vont prendre un temps de

$$D_D = D_L \pm |G_D|$$

pour arriver au récepteur.

Partant du front montant du signal Clk au transmetteur, les données arrivent au port D du Reg2 du récepteur après un temps

$$T_{Reg2/D} = T_{Clk} + D_{CQ} + D_D$$

De façon similaire, le temps d'arrivée de l'horloge au port C du Reg2 au récepteur est

$$T_{Reg2/C} = T_{ClkN} + D_{Clk} = T_{Clk} + T/2 + D_{Clk}$$

Le chronogramme à la figure 2.3 montre la relation temporelle entre les quantités ci-dessus.

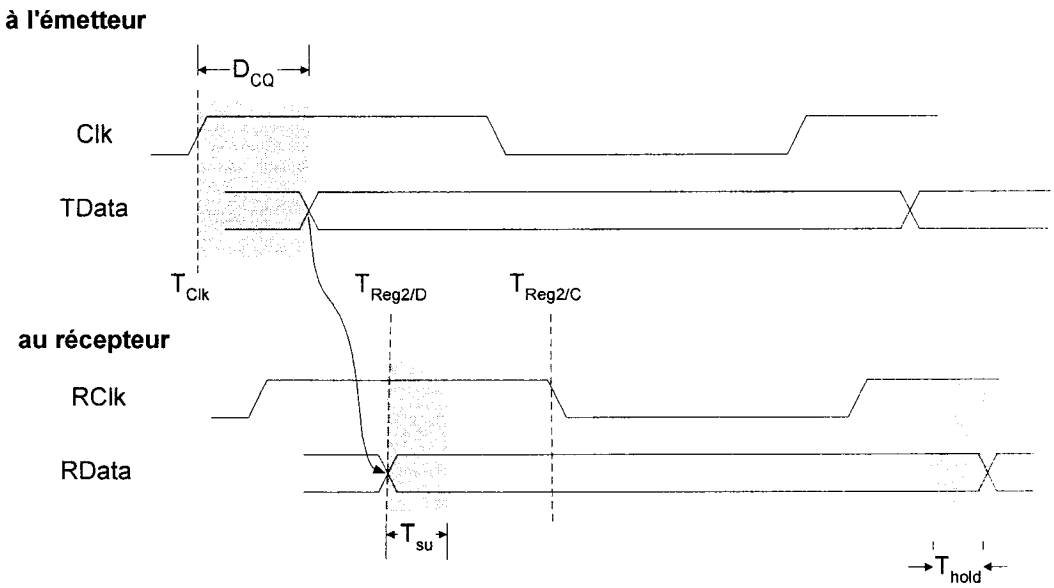

Figure 2.3. Chronogramme

2.3.1.1. Condition de pré-positionnement (setup)

Pour un bon fonctionnement au registre Reg2 et pour éviter les problèmes de métastabilité, il faut que la condition de prépositionnement soit respectée. En tenant compte explicitement du temps de pré-positionnement T_{su} des bascules D, la condition de prépositionnement du registre Reg2 est définie comme suit:

$$T_{Reg2/D} < T_{Reg2/C} - T_{su}$$

$$(T_{Clk} + D_{CQ} + D_D) < (T_{Clk} + T/2 + D_{Clk}) - T_{su} \quad (1)$$

Dans les pires conditions, à cause des fluctuations dans les délais de propagation, lorsque les données arrivent en retard et que l'horloge arrive plus tôt que leurs temps d'arrivée nominaux, on a:

$$D_D = D_L + |G_D|$$

$$D_{Clk} = D_L - |G_{Clk}|$$

en remplaçant les valeurs de D_D et D_{Clk}, l'inéquation (1) devient

$$T_{Clk} + D_{CQ} + (D_L + |G_D|) < T_{Clk} + T/2 + (D_L - |G_{Clk}|) - T_{su}$$

$$D_{CQ} + |G_D| < T/2 - |G_{Clk}| - T_{su}$$
$$|G_D| < T/2 - |G_{Clk}| - T_{su} - D_{CQ}$$

Ainsi, les grandeurs de gigue que le système peut tolérer tout en respectant les conditions de prépositionnement sont reliées par:

$$|G_{Clk}| + |G_D| < T/2 - (T_{su} + D_{CQ}) \quad (2)$$

2.3.1.2. Condition de maintien ("hold")

D'autre part, il faut aussi respecter les conditions de maintien pour ne pas être en situation de métastabilité. Dans notre cas, les conditions de maintien du registre Reg2 peuvent être exprimées comme suit:

$$T_{Reg2/D} + T > T_{Reg2/C} + T_{hold}$$
$$T_{Clk} + D_{CQ} + D_D + T > T_{Clk} + T/2 + D_{Clk} + T_{hold} \quad (3)$$

Dans les pires conditions, dues aux fluctuations dans les délais de propagation, lorsque les données arrivent plus tôt et que l'horloge arrive en retard,

$$D_D = D_L - |G_D|$$
$$D_{Clk} = D_L + |G_{Clk}|$$

en remplaçant les valeurs de D_D et D_{Clk}, l'inéquation (3) devient

$$T_{Clk} + D_{CQ} + (D_L - |G_D|) + T > T_{Clk} + T/2 + (D_L + |G_{Clk}|) + T_{hold}$$
$$D_{CQ} - |G_D| + T/2 > |G_{Clk}| + T_{hold}$$

Finalement, l'expression qui exprime le lien entre les grandeurs de gigue que le système peut tolérer tout en respectant les conditions de maintien est:

$$|G_{Clk}| + |G_D| < T/2 + D_{CQ} - T_{hold} \quad (4)$$

Pour ne pas être en situation de métastabilité, il faut donc respecter les conditions de prépositionnement et les conditions de maintien simultanément.

$$|G_{Clk}| + |G_D| < T/2 - T_{su} - D_{CQ} \quad (2)$$

$$|G_{Clk}| + |G_D| < T/2 + D_{CQ} - T_{hold} \quad (4)$$

Pour être rigoureux, nous pouvons généraliser que

$$T_{su} = [T_{su^-}, T_{su^+}]$$

$$T_{hold} = [T_{hold^-}, T_{hold^+}]$$

Constatant que les inéquations (2) et (4) sont plus contraignantes lorsque $T_{su} = T_{su^+}$ et $T_{hold} = T_{hold^+}$, les inéquations (2) et (4) peuvent être réécrites comme suit:

$$|G_{Clk}| + |G_D| < T/2 - T_{su^+} - D_{CQ}$$

$$|G_{Clk}| + |G_D| < T/2 + (-T_{su^+} - D_{CQ}) \quad (2a)$$

et

$$|G_{Clk}| + |G_D| < T/2 + D_{CQ} - T_{hold^+}$$

$$|G_{Clk}| + |G_D| < T/2 + (D_{CQ} - T_{hold^+}) \quad (4a)$$

Pour que (4a) soit plus contraignante que (2a), il faut que

$$-T_{su^+} - D_{CQ} > D_{CQ} - T_{hold^+}$$

$$T_{hold^+} > 2D_{CQ} + T_{su^+}$$

Or, en général et en pratique, T_{hold^+} est plus petit que T_{su^+} de sorte que l'inéquation précédente n'est pas valide. Donc, (2a) est plus contraignante. Il s'ensuit que si l'inéquation 2a est respectée, l'inéquation 4a va être aussi satisfaite.

En conclusion, les grandeurs de gigue que le système peut tolérer sont reliées par l'expression suivante:

$$|G_{Clk}| + |G_D| < T/2 - (T_{su+} + D_{CQ})$$

2.3.2. Optimisation

Nous venons de trouver l'expression mathématique qui exprime la limite des grandeurs de gigue que le système peut tolérer.

Cependant, il existe des optimisations théoriques possibles pour augmenter la valeur de $|G_D|+|G_{Clk}|$ tolérable. Une des solutions envisageables est de recentrer l'horloge afin de distribuer des pertes temporelles, dues à T_{su} et D_{CQ} et T_{hold}.

En effet, nous avons vu ci-dessus que les tolérances aux gigues sont limitées à
$T/2 - T_{su} - D_{CQ}$
et à
$T/2 + D_{CQ} - T_{hold}$
à gauche et à droite respectivement. En supposant que la condition de temps de pré-positionnement est plus contraignante que la condition de temps de maintien (comme c'est généralement le cas), la tolérance aux gigues est plus petite à gauche.

La tolérance optimale sera obtenue lorsque nous retardons l'horloge d'une quantité X telle que la tolérance soit égale des 2 côtés. En effet, en ajoutant un délai sur l'horloge, nous relaxons la condition de prépositionnement (setup) et nous contraignons plus la condition de maintien (hold).

Ainsi, les équations 2a et 4a deviennent:
$|G_{Clk}| + |G_D| < T/2 + (- T_{su}+ - D_{CQ}) + X$
$|G_{Clk}| + |G_D| < T/2 + (D_{CQ} - T_{hold}+) - X$

Lorsqu'elles sont égales, nous obtenons
$X = D_{CQ} + (T_{su} - T_{hold})/2$

Dans ce cas, la tolérance optimale sera

$|G_{Clk}| + |G_D| < T/2 - (T_{su} + T_{hold}) / 2$

Cette optimisation reste très théorique car en pratique, il est difficile voire impossible de prédire les quantités D_{CQ}, T_{su}, T_{hold}, car ces paramètres sont aussi affectés par les variations des procédés de fabrication et celles causées par l'environnement, comme la température et les fluctuations dans les tensions d'alimentation. De plus, à moins d'avoir des cellules à délai contrôlé, nous ne pouvons pas obtenir des pas assez précis pour retarder l'horloge exactement de la quantité calculée ci-dessus afin d'obtenir la tolérance optimale.

Dans l'inéquation ci-dessus, les quantités D_{CQ}, T_{su}, T_{hold} sont fixées ou dictées par la technologie. Il reste donc qu'une des optimisations possibles, pour encore augmenter la tolérance aux fluctuations dans le délai de propagation des données G_D, est de réduire la gigue sur l'horloge, G_{Clk}. Il existe plusieurs façons de réduire considérablement les fluctuations sur le délai de propagation de l'horloge jusqu'à ce que $|G_{Clk}|$ puisse être considéré comme négligeable à toutes fins pratiques.

Cependant, nous n'allons pas discuter des moyens pour y arriver car cela dépasse le cadre de cet ouvrage.

2.4. Tolérance au biais de synchronisation au récepteur

Le deuxième chemin critique est repris à la figure 2.4. L'analyse qui suit de cette structure suppose qu'une unité de contrôle choisit entre Reg3 et Reg4 sur la base d'un déphasage mesuré à un moment de synchronisation choisi. Une fois le chemin choisi, il est inchangé jusqu'à la prochaine resynchronisation:

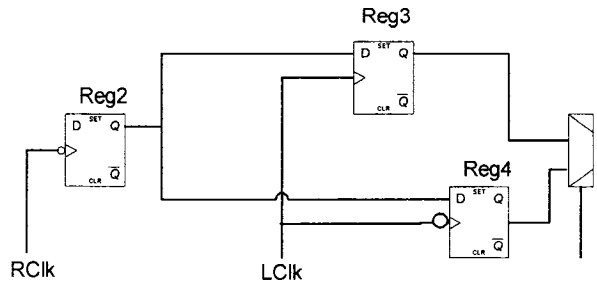

Figure 2.4. Chemin critique 2

2.4.1. Analyse

Définissons,

LClk , l'horloge locale au récepteur;

RClk , l'horloge distante reçue au récepteur;

T_{LClk} , temps d'arrivée du front montant de l'horloge locale, transmis dans un environnement idéal, (c'est-à-dire sans bruit et diaphonie);

T_{RClk} , temps d'arrivée du front montant de l'horloge distante, si transmis dans un environnement idéal, (sans bruit, sans diaphonie, sans effet de gradient de température, sans effet de gradient de tension d'alimentation, etc...);

$T_{RClk}(m)$, temps d'arrivée du front montant de l'horloge distante, transmis dans un environnement idéal, au moment m;

$T_{LClk}(k)$, temps d'arrivée du front montant de l'horloge locale, transmis dans un environnement idéal, au moment k.

Une transmission dans un environnement réel est sujette à des perturbations, dues aux bruits, aux interférences électromagnétiques, à l'effet de gradient de température et aux fluctuations dans les tensions d'alimentation. Cette perturbation par rapport au même signal transmis dans un environnement idéal peut être décomposée en 2 composantes: une perturbation de nature lente, appelée dérive et une perturbation de nature rapide, appelée gigue.

Pour séparer la composante lente de la composante rapide du signal, on peut penser à passer le signal dans un filtre passe bas et passe haut de même fréquence de coupure, de sorte que l'énergie totale soit conservée. Le choix de la fréquence de coupure est décidé en fonction de l'application et de l'implantation.

Ainsi, nous appelons

- $G1_{RCIk}$, dérive que RCIk puisse avoir lorsque ce dernier est transmis dans un environnement réel;
- $G1_{LCIk}$, dérive que LCIk puisse avoir lorsque ce dernier est transmis dans un environnement réel;
- $G2_{RCIk}$, gigue de l'horloge distante;
- $G2_{LCIk}$, gigue de l'horloge locale;

Supposons que $G2_{RCIk}$ et $G2_{LCIk}$ sont des variables aléatoires de moyenne nulle qui peuvent prendre des valeurs positives ou négatives. Ces quantités seront positives si elles retardent les horloges ou négatives si elles les accélèrent.

- $T_{RCIk}'(m)$, temps d'arrivée du front montant de l'horloge distante, transmis dans un environnement réel, au moment m;
- $T_{LCIk}'(k)$, temps d'arrivée du front montant de l'horloge locale, transmis dans un environnement réel, au moment k.

À l'instant n+1, dans un environnement réel

$$T_{RCIk}'(n+1) = T_{RCIk}(n+1) + G1_{RCIk} + G2_{RCIk}$$

Au moment k+1, dans un environnement idéal,

$$T_{LCIk}(k+1) = T_{LCIk}(k) + T$$

et dans un environnement réel

$T_{LClk}'(k+1) = T_{LClk}(k) + G1_{LClk(k+1)} + G2_{LClk(k+1)} + T$

Prenons l'instant k du LClk comme référence temporelle, tel que le moment n+1 de RClk se trouve entre deux fronts montants consécutifs k et k+1 de LClk. Dans ce cas, on a:

$T_{LClk}'(k) < T_{RClk}'(n+1) < T_{LClk}'(k+1)$ (1)

Soit,

ϕ , ayant comme unité temps, la différence de phase (déphasage) entre LClk(k) et RClk(n+1),

tel que

$T_{RClk}(n+1) = T_{LClk}(k) + \phi$

où $0 \leq \phi \leq T$.

Pour avoir une latence fixe, il faut que l'expression (1) soit satisfaite, sinon, un mot transmis arriverait à une période d'horloge autre que celle dans laquelle il est attendu. On perd alors une donnée si on entre dans une période précédente et on la dédouble si on entre dans une période d'horloge suivante de façon imprévue et incontrôlée.

$T_{LClk}'(k) < T_{RClk}'(n+1) < T_{LClk}'(k+1)$

En tenant compte des gigues et dérives sur les horloges,

$T_{LClk}(k) + G1_{LClk(k)} + G2_{LClk(k)} < T_{RClk}'(n+1) < T_{LClk}(k) + T + G1_{LClk(k+1)} + G2_{LClk(k+1)}$

$T_{LClk}(k) + G1_{LClk(k)} + G2_{LClk(k)} < T_{RClk}(n+1) + G1_{RClk(n+1)} + G2_{RClk(n+1)} < T_{LClk}(k) + T + G1_{LClk(k+1)} + G2_{LClk(k+1)}$

$$T_{LClk}(k) + G1_{LClk(k)} + G2_{LClk(k)} < T_{LClk}(k) + \phi + G1_{RClk(n+1)} + G2_{RClk(n+1)} < T_{LClk}(k) + T + G1_{LClk(k+1)} + G2_{LClk(k+1)}$$

$$G1_{LClk(k)} + G2_{LClk(k)} < \phi + G1_{RClk(n+1)} + G2_{RClk(n+1)} < T + G1_{LClk(k+1)} + G2_{LClk(k+1)} \quad (1b)$$

Si nous considérerons les changements de dérive de phase d'un cycle à l'autre comme négligeables, nous pouvons alors écrire:

$$G1_{LClk(k)} = G1_{LClk(k+1)}$$

L'équation (1b) devient

$$G2_{LClk(k)} < \phi + G1_{RClk(n+1)} + G2_{RClk(n+1)} - G1_{LClk(k)} < T + G2_{LClk(k+1)} \quad (1c)$$

Définissons ϕ_0, le déphasage initial entre $T_{RClk(n+1)}$ et $T_{LClk(k)}$. Le système envisagé est sujet à des resynchronisations périodiques qui redéfinissent à chaque fois la valeur de ϕ_0. En excluant la gigue présente au moment de la resynchronisation on a:

$$\phi_0 = \phi + G1_{RClk(n+1)} - G1_{LClk(k)}$$

Remplaçons cette valeur de ϕ_0 dans l'équation (1c), nous obtenons

$$G2_{LClk(k)} < \phi_0 + G2_{RClk(n+1)} < T + G2_{LClk(k+1)} \quad (1d)$$

d'où

$$\phi_0 + G2_{RClk(n+1)} < T + G2_{LClk(k+1)}$$

$$\phi_0 < T + G2_{LClk(k+1)} - G2_{RClk(n+1)} \quad (2)$$

et

$$G2_{LClk(k)} < \phi_0 + G2_{RClk(n+1)}$$

$$G2_{LClk(k)} - G2_{RClk(n+1)} < \phi_0$$

$$\phi_0 > G2_{LClk(k)} - G2_{RClk(n+1)} \quad (3)$$

Définissons,

$|G1_{RClk}|$, module de $G1_{RClk}$;

$|G1_{LClk}|$, module de $G1_{LClk}$;

$|G2_{RClk}|$,module de $G2_{RClk}$;

$|G2_{LClk}|$, module de $G2_{LClk}$;

Pour l'inéquation 2, nous avons un pire cas lorsque $G2_{RClk(n+1)}$ retarde le front n+1 de RClk, et $G2_{LClk(k+1)}$ accélère le front k+1 de LClk.

Dans ce contexte, l'inéquation (2) devient

$$\phi_0 < T + G2_{LClk(k+1)} - G2_{RClk(n+1)}$$

$$\phi_0 < T + (- |G2_{LClk(k+1)}|) - (+ |G2_{RClk(n+1)}|) \quad (4)$$

Pour l'inéquation 3, le pire cas arrive lorsque $G2_{RClk(n+1)}$ accélère le front n+1 de RClk et $G2_{LClk(k+1)}$ retarde le front k+1 de LClk. Dans cette condition, l'inéquation (3) devient donc

$$\phi_0 > |G2_{LClk(k)}| - (- |G2_{RClk(n+1)}|)$$

$$\phi_0 > |G2_{LClk(k)}| + |G2_{RClk(n+1)}| \quad (5)$$

Soit P, la perturbation sur RClk que le système peut tolérer telle que:

$$P = |G2_{RClk(n+1)}|$$

de (4) on peut écrire

$$\phi_0 < (T - |G2_{LClk(k+1)}|) - P$$

$$P < (T - |G2_{LClk(k+1)}|) - \phi_0 \quad (6)$$

et de (5), il s'ensuit que

$$P < \phi_0 - |G2_{LClk(k)}| \qquad (7)$$

À la lumière des deux dernières inéquations (6) et (7), on constate que P est dépendant, en grande partie de ϕ_o, le déphasage initial entre RClk et LClk au moment d'une resynchronisation.

Résolvons simultanément (6) et (7), nous obtenons

$$P = \min (T - |G2_{LClk(k+1)}|) - \phi_0 \ , \phi_0 - |G2_{LClk(k)}|)$$

2.4.2. Étude des cas limites

Dans la relation ci-dessus qui donne la perturbation maximale que le système peut tolérer, nous constatons que la valeur $|G2_{LClk(k+1)}|$ est une constante. Cette dernière est caractéristique et elle est due à la topologie du système. Nous allons maintenant déterminer les valeurs du déphasage initial ϕ_0 qui correspondent à des limites inférieures et supérieures de P.

2.4.2.1. Tolérance maximale

En examinant la relation ci-dessus, nous constatons que la situation qui conduit à une tolérance maximale aux variations est lorsque

$$\phi_o = T/2$$

Dans ce cas, P atteint $T/2 - \lambda$ où λ est petit

2.4.2.2. Tolérance minimale

D'autre part, la situation qui conduit à une tolérance minimale aux variations est lorsque

$$\phi_o = T - |G2_{LClk(k+1)}|$$

ou lorsque

$$\phi_o = |G2_{LClk(k)}|$$

Dans ces 2 cas, P est réduit à λ où λ est petit

2.4.3. Optimisation

Il est possible d'augmenter encore la tolérance au biais de synchronisation avec le même circuit. En effet, dans le cas où on peut choisir dynamiquement le front montant ou le front descendant de LClk, on a:

$$T_{LClk}'(k) + T/2 < T_{RClk}'(n+1) < T_{LClk}'(k+1) + T/2$$

ou

$$T_{LClk}'(k) - T/2 < T_{RClk}'(n+1) < T_{LClk}'(k+1) - T/2$$

Dans l'analyse qui suit, nous allons utiliser une forme condensée dans laquelle les signes $\pm$ sont corrélés pour représenter les 2 cas comme suit:

$$T_{LClk}'(k) \pm T/2 < T_{RClk}'(n+1) < T_{LClk}'(k+1) \pm T/2$$

Par un raisonnement similaire à l'analyse ci-dessus, on obtiendra, au lieu de (1d)

$$\pm |G2_{LClk(k)}| \pm T/2 < \phi_o \pm |G2_{RClk(n+1)}| < T \pm |G2_{LClk(k+1)}| \pm T/2$$

d'où

$$\phi_o \pm |G2_{RClk(n+1)}| < T \pm |G2_{LClk(k1+)}| \pm T/2$$

$$\phi_o < T \pm |G2_{LClk(k+1)}| - (\pm |G2_{RClk(n+1)}|) \pm T/2 \quad (8)$$

et

$$\phi_o \pm |G2_{RClk(n+1)}| > \pm |G2_{LClk(k)}| \pm T/2$$

$$\phi_o > \pm |G2_{LClk(k)}| \pm T/2 - (\pm |G2_{RClk(n+1)}|) \quad (9)$$

En utilisant le même raisonnement que précédemment, le pire cas de (8) est lorsque $G2_{RClk(n+1)}$ retarde le front n+1 de RClk, alors que $G2_{LClk(k+1)}$ accélère le front k+1 de LClk. Dans ce cas l'équation (8) devient

$$\phi_o < T + (-|G2_{LClk(k+1)}|) - (|G2_{RClk(n+1)}|) \pm T/2 \quad (10)$$

Le pire cas de (9) arrive lorsque $G2_{RClk(n+1)}$ accélère le front n+1 de RClk, alors que $G2_{LClk(k+1)}$ retarde le front de LClk. Dans ce cas, l'équation (9) devient

$$\phi_o > |G2_{LClk(k)}| \pm T/2 - (-|G2_{RClk(n+1)}|) \quad (11)$$

Soit P', la nouvelle perturbation sur RClk que le système peut tolérer, tel que

$$P' = |G2_{RClk(n+1)}|$$

Remplaçons $|G2_{RClk(n+1)}|$ par P' dans (10)

$$\phi_o < T - P' - (|G2_{LClk(k+1)}|) \pm T/2$$

ou

$$P' < T - |G2_{LClk(k+1)}| - \phi_o \pm T/2 \quad (12)$$

et dans (11)

$$\phi_o > |G2_{LClk(k)}| \pm T/2 + P'$$

ou

$$P' < \phi_o - (\pm T/2) - |G2_{LClk(k)}| \qquad (13)$$

Par définition, $0<\phi_o<T$, (12) et (13) montrent qu'on ne peut ajouter T/2 que lorsque $\phi_o<T/2$ et enlever de T/2 que lorsque $\phi_o > T/2$.

Ainsi, lorsque

$\phi_o > T/2$,

(12) devient

$$P' < T - |G2_{LClk(k+1)}| - \phi_o + T/2$$

$$P' < T - |G2_{LClk(k+1)}| - (\phi_o - T/2)$$

et (13) devient

$$P' < \phi_o - T/2 - |G2_{LClk(k)}|$$

En résolvant simultanément les 2 inéquations précédentes, nous avons finalement

$$P' = \min (T - |G2_{LClk(k+1)}| - (\phi_o - T/2), (\phi_o - T/2) - |G2_{LClk(k)}|)$$

$\phi_o < T/2$,

(12) devient

$$P' < T - |G2_{LClk(k+1)}| - \phi_o - T/2$$

$$P' < T - |G2_{LClk(k+1)}| - (\phi_o + T/2)$$

et (13) devient

$$P' < \phi_o + T/2 - | G2_{LClk(k)}|$$

Résolvons simultanément les 2 inéquations précédentes, nous avons finalement

$$P' = \min (T - (\phi_o + T/2) - |G2_{LClk(k+1)}|, (\phi_o + T/2) - | G2_{LClk(k)}|)$$

L'équation ci-dessus représente la forme améliorée de la tolérance au biais de synchronisation du deuxième chemin critique. Cependant, elle dépend encore de la phase initiale ϕ_o.

2.5. Conclusion

Nous venons de faire l'analyse théorique des deux chemins critiques de la solution originalement proposée par la société HyperChip.

Le premier chemin, exploitant un schème couramment utilisé en communication asynchrone entre le transmetteur et le récepteur, permet de tolérer un biais de synchronisation allant jusqu'à $T/2 - \varepsilon$. Avec certaines précautions, on peut optimiser pour rapetisser ε encore plus et pousser la tolérance encore plus proche de T/2.

Le deuxième chemin critique, au récepteur et entre deux domaines d'horloge différents, possède certaines propriétés particulières. Théoriquement, la tolérance maximale au biais de synchronisation peut atteindre $T/2-\varepsilon$. Cependant, dans certains cas, le système doit dynamiquement changer la latence durant la réception pour pouvoir maintenir cette tolérance maximale. Ainsi, dans le cas d'un système à latence fixe, la tolérance au biais de synchronisation peut être réduite à une valeur aussi petite que ε. En effet, nous avons démontré que la tolérance peut varier de ε à $T/2-\varepsilon$ selon la phase initiale

entre l'horloge distante (l'horloge émise par l'émetteur et reçue au récepteur) et celle du récepteur.

Par ailleurs, ce comportement est aussi confirmé par des simulations comportementales en utilisant des modèles VHDL du système.

Dans la pratique, un système dont la latence varie d'un cycle à un autre peut être aussi acceptable, selon l'application. Cependant, dans les applications de communication telles qu'envisagées par la société HyperChip, une latence variable d'un cycle à un autre n'est pas désirable, car ceci se traduit en des duplications ou des pertes de données. Ceci nous amène à chercher d'autres moyens pour améliorer la proposition originale de la société HyperChip. Plusieurs autres solutions ont été proposées.

Dans le prochain chapitre, nous allons présenter l'une des idées parmi celles qui nous ont été suggérées et qui ont été examinées. Cependant, cette dernière n'a pas été retenue compte tenu de ses limitations.

Chapitre III.
Autres cas étudiés

3.1. Introduction

Pendant la recherche de solutions de rechange pour le système original promu par la société HyperChip et avant d'arriver à la solution finale présentée dans le chapitre suivant, nous avons examiné plusieurs idées qui ont été suggérées. Dans ce chapitre, nous allons présenter une de ces propositions. Cette idée est par ailleurs suggérée par la société HyperChip comme une dérivation la plus proche de l'idée originale et qui se rapproche de la solution retenue.

Comme cette proposition n'est pas retenue, nous n'allons faire qu'un survol du principe de fonctionnement. Nous allons aussi présenter une analyse de son modèle mathématique pour évaluer ses performances et pour identifier ses limites.

3.2. Description et principe de fonctionnement

Au moment de resynchronisation, RClkN, le front descendant de RClk, doit être à α et β des deux fronts montants consécutifs n et n+1 de LClk, comme montre la figure ci-dessous.

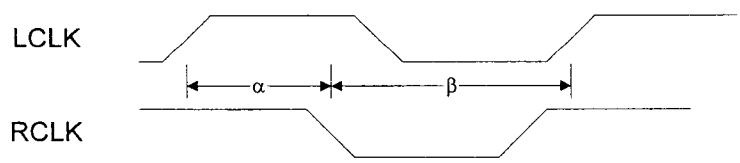

Figure 3.1. Relation LClk – RClkN

Le système proposé établira aussi une horloge intermédiaire, appelée aussi ICIk, tel que ICIk va être au milieu de l'intervalle la plus grande entre α et β. L'idée est de générer l'horloge intermédiaire ICIk de telle sorte que ICIk se trouve au milieu du plus grand des 2 intervalles (α et β) qui séparent ICIk des deux fronts montants n et n+1 de LCIk. La sélection de ICIk peut se faire périodiquement comme par exemple pendant un temps mort entre 2 messages.

Mathématiquement, la règle de sélection de ICIk peut être exprimée comme suit:

$T_{ICIk} = T_{LCIk(n)} + \alpha + \beta/2$ si $\beta > \alpha$

ou

$T_{ICIk} = T_{LCIk(n)} + \alpha/2$ si $\alpha > \beta$

où,

$T_{LCIk(n)}$ le temps d'arrivée du n ième front montant de LCIk.

$T_{LCIk(n+1)}$ le temps d'arrivée du n+1 ième front montant de LCIk.

T_{RCIkN} le temps d'arrivée du front descendant de RCIk.

T_{ICIk} le temps d'arrivée du front montant de ICIk.

et que

$T_{RCIkN} - \alpha = T_{LCIk(n)}$

$T_{RCIkN} + \beta = T_{LCIk(n+1)}$

$\alpha + \beta = T$

De cette manière, les données synchronisées par ICIk sont prêtes à être resynchronisées aussitôt avec le premier front montant de LCIk qui suit RCIkN.

L'avantage de cette proposition est que la latence reste minimale (0 cycle lorsque $\alpha > \beta$ ou 1 cycle dans le cas contraire) tout en garantissant les temps de pré-positionnement t_{su} et temps de maintien t_{hold} .

3.3. Analyse

Comme $\alpha + \beta = T$, si $\alpha > \beta$, alors $\alpha > T/2$

Si $\beta > \alpha$, alors $\beta > T/2$

En appliquant ces relations dans les équations définissant T_{ICIk} ci-dessus, nous obtenons que T_{ICIk} est toujours d'au moins T/4 de chaque front montant de LCIk. La tolérance τ sera donc théoriquement plus grande que T/4.

Cependant, si au moment de synchronisation, RCIkN a déjà subi une fluctuation de nature rapide γ, la tolérance sera

$\tau' > T/4 - \gamma$

Les cas limites arrivent lorsque γ se rapproche de T/4. Dans ce cas, τ sera considérablement réduit et ne pourra plus garantir les contraintes de pré-positionnement.

3.4. Conclusion

Dans ce chapitre, nous venons de présenter une des propositions parmi celles qui ont été suggérées visant à améliorer ou remplacer la proposition initiale de la société HyperChip.

Cependant, cette idée n'a pas été retenue en raison de ses limitations qui ont été démontrées par l'analyse de son modèle mathématique.

Dans le prochain chapitre, nous allons présenter la solution la plus prometteuse parmi celles qui ont été suggérées comme amélioration ou remplacement de l'idée originale de la société HyperChip. Nous allons décrire son principe de fonctionnement et nous y présenterons les analyses basées sur le modèle mathématique de la solution retenue pour montrer que cette dernière possède les performances recherchées.

Chapitre IV.
Solution retenue

4.1. Introduction

La limitation du système originalement proposé par la société HyperChip nous amène à chercher des améliorations ou encore une solution de rechange. Nous cherchons donc une solution qui, par construction, ne doit plus avoir de dépendance à la phase initiale entre l'horloge locale et l'horloge distante. Ainsi, la solution doit présenter une plus grande tolérance au biais de synchronisation, tout en ayant une latence fixe, d'un cycle à un autre et ce, pour toute la durée de réception de données.

Plusieurs idées nous sont présentées. Nous avons examiné l'une d'elles dans le chapitre précédent. Finalement, nous arrêtons notre choix sur la solution la plus prometteuse, qui sera présentée dans ce chapitre.

Dans ce chapitre, nous expliquerons d'abord les principes de fonctionnement du système proposé, avec un diagramme temporel à l'appui. Nous identifierons les chemins critiques. Ensuite, nous ferons les analyses théoriques de ces chemins critiques pour démontrer les performances ou limites théoriques du nouveau système. Cependant, pour éviter les répétitions, l'analyse théorique ne sera refaite que sur les changements par rapport à la solution originale proposée par HyperChip, c'est-à-dire sur la partie récepteur de la figure 2.1.

4.2. Principe de fonctionnement

Le schéma bloc de la solution retenue est montré à la figure 4.1.

Dans la première partie, comme la solution originale de la société HyperChip, la solution retenue utilise le schème de transfert asynchrone avec l'horloge distante accompagnant les données arrivées au récepteur. Ensuite, nous introduisons une nouvelle horloge intermédiaire, dérivée de l'horloge locale. De plus nous proposons de faire des resynchronisations périodiques pour maintenir cette horloge intermédiaire en phase avec l'horloge distante. Par exemple, cette resynchronisation peut se faire entre 2 messages. De cette façon, nous pouvons tolérer les gigues de nature rapide sur l'horloge distante.

En effet, à cause de l'environnement, l'horloge distante possède des fluctuations, ce qui fait que la phase entre celle-ci et l'horloge locale varie continuellement. Comme nous avons vu dans le chapitre 2, les variations peuvent être de nature lente ou rapide. Ainsi, une resynchronisation périodique, dont la fréquence dépend de l'application, aura pour effet de minimiser, sinon d'éliminer, les fluctuations de nature lente. Par la suite, grâce à la stabilité de l'horloge intermédiaire nous pouvons donc tolérer des fluctuations rapides sur l'horloge distante.

4.2.1. Schéma bloc du récepteur

La figure 4.1 montre le schéma bloc du récepteur de la solution retenue. Nous omettons la partie de l'émetteur ainsi que les connexions qui relient ces deux parties pour les raisons mentionnées ci-dessus.

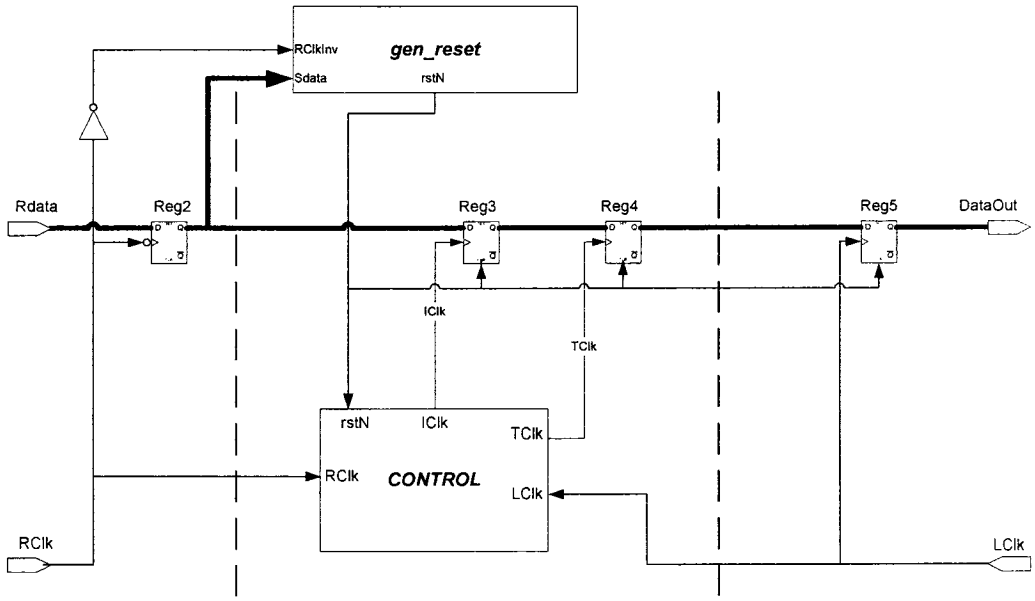

Figure 4.1. Schéma bloc du récepteur de la solution retenue

4.2.2. Description et principe de fonctionnement

Les descriptions ci-dessous s'appliquent à la partie du récepteur, tel que montré à la figure 4.1.

En première partie, comme dans la solution originale proposée par la société HyperChip, le récepteur reçoit, en plus des données, l'horloge distante RClk. Le récepteur possède aussi sa propre horloge locale LClk.

Le schéma bloc simplifié du bloc CONTROL, suffisant pour en comprendre le principe de fonctionnement, est montré à la figure 4.2. Une version plus complète de ce circuit est décrite en détail au chapitre suivant. Notons ici que ce bloc génère une troisième horloge, appelée l'horloge intermédiaire, IClk, dérivée de LClk. En effet, à l'intérieur du bloc CONTROL, LClk passe à travers une chaîne à délai pour générer différentes phases, appelées LClkd(i). Le module CONTROL sélectionne une des phases LClkd(i) pour en faire IClk. Ensuite,

connaissant la sélection de ICIk et donc la relation de phase entre ICIk et LCIk, le bloc CONTROL génère une quatrième horloge, appelée l'horloge de transition, TCIk. Ces différentes horloges sont utilisées pour faire passer les données de façon sécuritaire, comme montré à la figure 4.1.

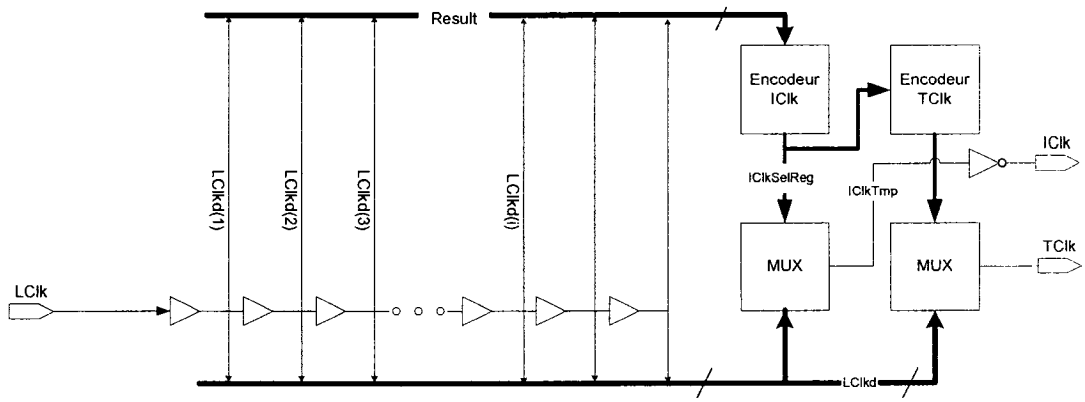

Figure 4.2. Schéma bloc simplifié du module CONTROL

De façon globale, le système fonctionne en deux modes comme suit:

a. Mode auto-calibrage (self-reset)

Le système doit s'auto-synchroniser (self-reset) à une certaine fréquence. L'intervalle des auto-synchronisations ainsi que le protocole à utiliser seront déterminés en fonction de l'application et des contraintes de performance qui en découlent. Dans le design pour montrer le concept, nous assumons que le système s'auto-synchronise après avoir reçu une séquence « 0F, F0, 0F ». Ceci n'est qu'un protocole simple qui a été suggéré parmi tant d'autres protocoles plus complexes envisageables. La requête de resynchronisation est détectée par le bloc gen_reset qui, en plus de remettre à zéro le pipeline de données, avertit le module CONTROL qu'une resynchronisation est demandée.

Pendant cette phase d'auto-calibrage,

(i) d'abord, le système sélectionne l'une des phases LCIkd(i), dérivée de LCIk, pour en faire ICIkTmp, un signal interne au bloc CONTROL. On choisit

comme signal IClkTmp la phase LClkd(i) dont le front montant est le plus proche du front descendant de l'horloge distante RClk reçue, c'est-à-dire RClkN. La sélection est alors registrée et reste inchangée jusqu'au prochain auto-calibrage. IClkTmp sera inversé pour finalement devenir IClk.

Dans le design élaboré pour montrer le concept, nous avons choisi une implantation qui prendra une phase LClkd(i) dont le front montant est le plus proche ***après*** le front descendant de RClk pour en faire IClkTmp. Selon l'application, d'autres implantations pour améliorer la précision de sélection pourraient être envisagées.

(ii) connaissant le déphasage de IClk par rapport à LClk, le système va déterminer si c'est le front montant ou descendant de LClk qui sera utilisé comme TClk. TClk est alors utilisé au registre *reg4* pour assurer la transition entre IClk et LClk. La règle de sélection est la suivante

si le front montant de IClk se trouve à au moins T/2 + α du front montant de LClk où α est une marge de sécurité, le front descendant de LClk va être utilisé comme TClk. Autrement, le front montant de LClk va être utilisé comme TClk

Ce mode d'auto-calibrage débutera immédiatement dans le cycle d'horloge qui suit une remise à zéro (*reset*) synchrone, sinon, pendant le cycle qui suit une remise à zéro asynchrone. Après l'auto-calibrage, qui peut prendre plus d'un cycle d'horloge pour s'accomplir selon l'implantation choisie, le système passe directement au mode réception de données.

b. Mode réception de données

Quand les horloges sont déterminées:

- Les données sont d'abord rééchantillonnées par *Reg2*, sur le front descendant de l'horloge distante reçue.
- Les données sont ensuite resynchronisées sur *IClk,* au *Reg3*.
- Le *Reg4*, fonctionnant sur TClk, assure la transition entre les deux domaines d'horloge LClk et IClk.
- Finalement, *Reg5* enregistre les données sur LClk.

La figure 4.3 montre un diagramme temporel du système en mode réception de données pour un déphasage donné entre LClk et RClk.

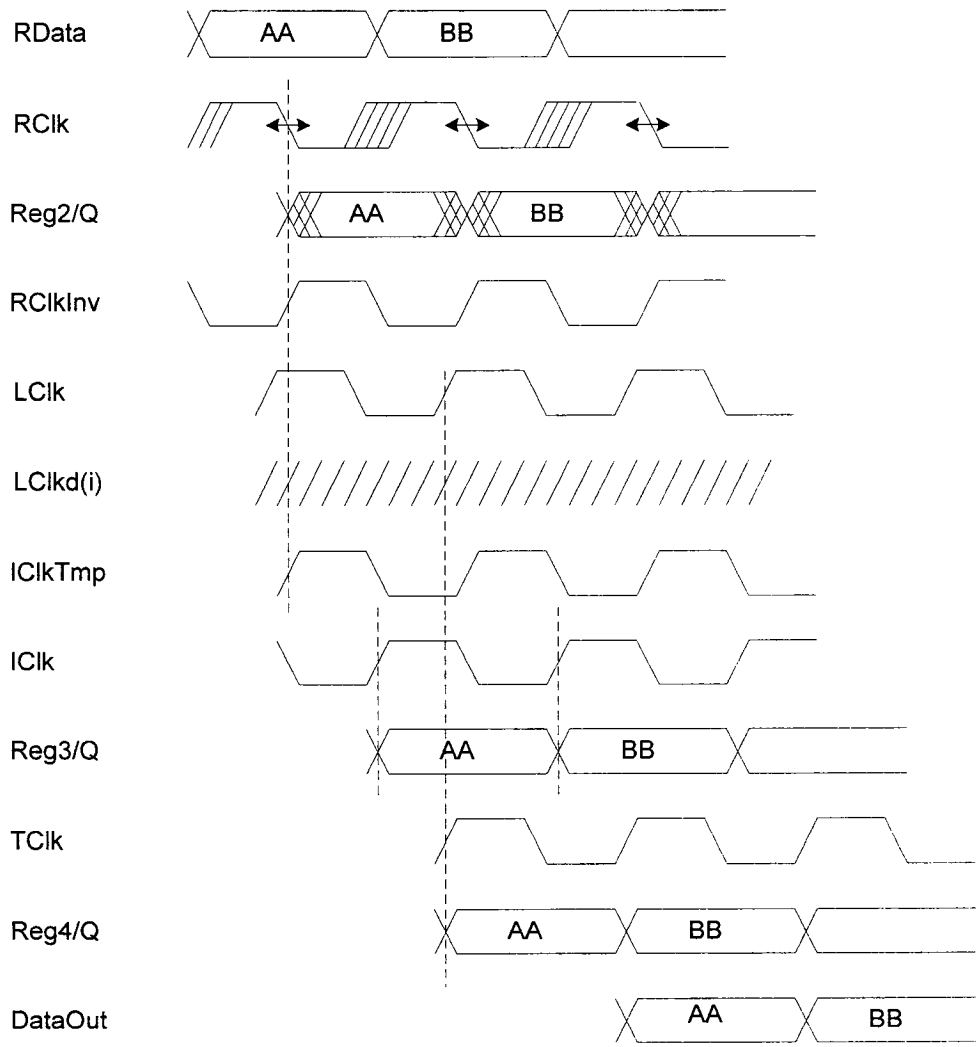

Figure 4.3. Diagramme temporel du système proposé en mode réception

4.3. Analyse du chemin critique

Une inspection du diagramme bloc permet de constater que les chemins critiques se trouvent dans les changements de domaine d'horloge, de RClkN à IClk et de IClk à TClk. Cependant, par construction, nous notons que le chemin qui relie RClkN à IClk est le plus à risque. Pour analyser ce dernier, il est donc nécessaire de connaître la relation de phase entre les horloges RClkN et IClk.

Tel que déterminé à la section précédente, l'horloge IClk est sélectionnée pendant la phase d'auto-synchronisation et cette sélection est enregistrée de sorte que la relation de phase ne change pas pendant la réception de données. Commençons donc par trouver la relation de IClk durant le mode d'auto-calibrage.

4.3.1. Relation IClk - RClkN au moment d'auto-synchronisation

Soit,

- T_{RClkN} Temps d'arrivée du front descendant de l'horloge distante RClk dans un environnement idéal;
- $T_{IClkTmp}$ Temps d'arrivée du front montant IClkTmp;
- T_{IClk} Temps d'arrivée du front montant IClk tel que;
  $T_{IClk} = T_{IClkTmp} + T/2$.

L'horloge locale LClk est considérée sans perturbation et de rapport cyclique 50%.

- $G1_{RClk}$, dérive que RClk puisse avoir lorsque ce dernier est transmis dans un environnement réel;
- $G1_{RClko}$, dérive que RClk puisse avoir lorsque ce dernier est transmis dans un environnement réel au moment de synchronisation;

$G2_{RClk}$, gigue de l'horloge distante;

$G2_{RClko}$, gigue de l'horloge distante au moment de synchronisation;

T_{RClkN}' , temps d'arrivée du front descendant de l'horloge distante RClk.

dans un environnement réel,

$$T_{RClkN}' = T_{RClkN} + G1_{RClk} + G2_{RClk}$$

Au moment de resynchronisation,

$$T_{RClkN}' = T_{RClkN} + G1_{RClko} + G2_{RClko}$$

Le principe de l'algorithme de sélection de IClk dans l'implantation de la puce de démonstration est de choisir la première phase LClkd(i) de LClk après le front montant de RClkN.

Soit $T_{LClkd(i)}$ et $T_{LClkd(i+1)}$, 2 temps d'arrivée successifs de LClk dans la chaîne à délai tel que

$$T_{LClkd(i)} < T_{RClkN}' < T_{LClkd(i+1)}$$

dans lequel

$$T_{LClkd(i+1)} = T_{LClkd(i)} + D_{(i)}$$

où $D_{(i)}$ représente le temps de propagation d'un étage dans la chaîne à délai.

Comme nous avons mentionné dans la description du fonctionnement, le système va d'abord choisir IClkTmp tel que

$$T_{RClkN}' < T_{IClkTmp} < T_{RClkN}' + D_{(i)}$$

Idéalement $T_{IClkTmp}$ doit être le plus proche possible de T_{RClkN}'. Dans cette optique, le pire cas de l'équation précédente arrive lorsque la différence entre T_{RClkN}' et T_{IClk} est la plus grande possible, i.e., lorsque

$$T_{IClkTmp} = T_{RClkN}' + D_{(i)}$$

En remplaçant T_{RClkN}' dans l'équation précédente, nous obtenons

$$T_{IClkTmp} = T_{RClkN} + G1_{RClko} + G2_{RClko} + D_{(i)}$$

Finalement, avec la relation entre IClkTmp et IClk calculée ci-dessus, nous avons

$$T_{IClk} = T_{RClkN} + G1_{RClko} + G2_{RClko} + D_{(i)} + T/2 \quad (1)$$

4.3.2. Pendant la réception

Le premier chemin le plus critique du système en mode de réception est la transition de donnée entre 2 domaines d'horloge RClk et IClk et il est illustré par la figure 4.4.

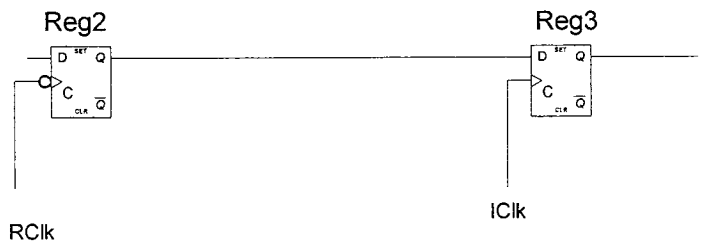

Figure 4.4. Relation RClkN et IClk

Soit,

$T_{Reg2/C}$, temps d'arrivée du front montant de l'horloge au port C du registre Reg2;

$T_{Reg3/C}$, temps d'arrivée du front montant de l'horloge au port C du registre Reg3;

$T_{Reg3/D}$, temps d'arrivée des données au port D du registre Reg3;

T_{RClkN} , temps d'arrivée du front descendant de l'horloge distante dans un environnement idéal;

T_{IClk} , temps d'arrivée du front montant de l'horloge intermédiaire dans un environnement idéal;

$G1_{RClk}$ dérive de l'horloge distante RClk lorsque ce dernier est transmis dans un environnement réel;

$G2_{RClk}$ gigue de l'horloge distante RClk lorsque ce dernier est transmis dans un environnement réel;

T_{RClkN} , temps d'arrivée du front descendant de l'horloge distante dans un environnement idéal;

T_{RClkN}' , temps d'arrivée du front descendant de l'horloge distante dans un environnement réel tel que

$T_{RClkN}' = T_{RClkN} + G1_{RClk} + G2_{RClk}$

D_{CQ} , le délai de propagation de C à Q d' une bascule.

Partant du port C du registre Reg2, arrivant au port D de Reg3

$T_{Reg3/D} = T_{Reg2/C} + D_{CQ}$ + délai sur la ligne

Assumons que le placement physique au niveau du dessin des masques des deux registres au récepteur soit bien contrôlé de sorte qu'on peut considérer pratiquement nul le délai sur la ligne, c'est-à-dire que

$T_{Reg3/D} = T_{Reg2/C} + D_{CQ}$

or

$T_{Reg2/C} = T_{RClkN}'$

ou

$$T_{Reg2/C} = T_{RClkN} + G1_{RClk} + G2_{RClk}$$

D'où

$$T_{Reg3/D} = T_{RClkN} + G1_{RClk} + G2_{RClk} + D_{CQ}$$

D'autre part, au port C de Reg3 au récepteur

$$T_{Reg3/C} = T_{IClk}$$

a. Condition de pré-positionnement (setup)

Pour un bon fonctionnement de Reg3, il faut que les données arrivent avant l'horloge, i.e.

$$T_{Reg3/D} < T_{Reg3/C}$$

En tenant compte du temps de pré-positionnement T_{su} des bascules D, on a

$$T_{Reg3/D} < T_{Reg3/C} - T_{su}$$

$$(T_{RClkN} + G1_{RClk} + G2_{RClk} + D_{CQ}) < (T_{IClk}) - T_{su} \quad (2)$$

En remplaçant (1) dans (2)

$$(T_{RClkN} + G1_{RClk} + G2_{RClk} + D_{CQ}) < (T_{RClkN} + G1_{RClko} + G2_{RClko} + D_{(i)} + T/2) - T_{su}$$

$$G1_{RClk} + G2_{RClk} + D_{CQ} < D_{(i)} + G1_{RClko} + G2_{RClko} + T/2 - T_{su}$$

Une resynchronisation périodique a pour but d'éliminer la dépendance à la dérive $G1_{RClk}$, c'est à dire

$$G1_{RClk} = G1_{Rclko}$$

Ainsi,

$$G2_{RClk} + D_{CQ} < D_{(i)} + G2_{RClko} + T/2 - T_{su}$$

Le pire cas arrive lorsque $G2_{RClk}$ est de direction opposée à $G2_{RClko}$, et que $G2_{RClko}$ est une quantité négative

$$|G2_{RClk}| < D_{(i)} - |G2_{RClko}| + T/2 - T_{su} - D_{CQ} \quad (3)$$

b. Condition de maintien (hold)

D'autre part, l'équation pour le temps de maintien des bascules D est

$$T_{Reg3/D} + T > T_{Reg3/C} + T_{hold}$$

$$T_{RClkN} + G1_{RClk} + G2_{RClk} + D_{CQ} + T > T_{IClk} + T_{hold} \quad (4)$$

En remplaçant (1) dans (4)

$$T_{RClkN} + G1_{RClk} + G2_{RClk} + D_{CQ} + T > (T_{RClkN} + G1_{RClko} + G2_{RClko} + D_{(i)} + T/2) + T_{hold}$$

$$G1_{RClk} + G2_{RClk} + D_{CQ} + T > G1_{RClko} + G2_{RClko} + D_{(i)} + T/2 + T_{hold}$$

La resynchronisation périodique a pour effet d'éliminer la dépendance à la dérive $G1_{RClk}$. Ainsi,

$$G2_{RClk} + D_{CQ} + T > G2_{RClko} + D_{(i)} + T/2 + T_{hold}$$

Le pire cas arrive lorsque $G2_{RClk}$ est de direction opposée à $G2_{RClko}$, et que $G2_{RClko}$ est une quantité positive

$$-|G2_{RClk}| + D_{CQ} + T > D_{(i)} + |G2_{RClko}| + T/2 + T_{hold}$$

$$|G2_{RClk}| < D_{CQ} + T/2 - D_{(i)} - |G2_{RClko}| - T_{hold} \quad (5)$$

Pour un bon fonctionnement, nous devons satisfaire les 2 équations (3) et (5) simultanément. Ainsi, la plus contraignante des 2 inéquations (3) et (5), reprise ci-dessous, dictera le performance du système

$$|G2_{RClk}| < T/2 + D_{(i)} - |G2_{RClo0}| - T_{su} - D_{CQ} \quad (3)$$

$$|G2_{RClk}| < T/2 + D_{CQ} - D(i) - |G2_{RClk0}| - T_{hold} \quad (5)$$

Il s'ensuit que la perturbation tolérable par ce système est la plus petite valeur de $|G2_{RClk}|$ dans les deux inéquations précédentes.

En pratique, nous pouvons considérer que D_{CQ}, $D_{(i)}$, T_{hold} et T_{su} sont petits par rapport à T, d'où le système peut tolérer une perturbation allant jusqu'à T/2–ε.

4.3.3. Étude des cas limites

Supposons que le front descendant de RClk, c'est-à-dire RClkN, a un gigue allant jusqu'à T/4, au moment de la resynchronisation. En d'autres termes, $G2_{RClko} = T/4$.

L'inéquation (3) devient

$$|G2_{RClk}| < T/2 + D_{(i)} - T/4 - T_{su} - D_{CQ}$$
$$|G2_{RClk}| < T/4 + D_{(i)} - T_{su} - D_{CQ}$$

et (5) devient

$$|G2_{RClk}| < T/2 + D_{CQ} - D_{(i)} - T/4 - T_{hold}$$
$$|G2_{RClk}| < T/4 + D_{CQ} - D_{(i)} - T_{hold}$$

d'où la perturbation tolérée se rapproche de T/4–ε.

L'absence des termes qui réfèrent au déphasage initial dans l'inéquation finale ci-dessus démontre que la solution retenue, quoiqu'elle dépende encore de la perturbation qui existe au moment de la synchronisation, ne dépend plus de la phase initiale entre l'émetteur et le récepteur. La tolérance, dans le pire des cas, peut aller jusqu'à T/4. Notons que la contrainte exprimée à la section 2.3.1.2 s'applique, mais elle est toujours moins sévère.

4.4. Conclusion

Nous venons de présenter le principe de fonctionnement de la solution qui a été retenue comme solution de rechange à celle proposée par HyperChip. Nous avons aussi analysé le modèle mathématique de son chemin le plus critique. L'analyse mathématique démontre que le système proposé peut, en général, tolérer des fluctuations sur l'horloge distante allant jusqu'à T/2–ε et jusqu'à T/4–ε dans les pires cas. L'étude de cas limite démontre aussi que la tolérance aux fluctuations sur l'horloge distante ne dépend plus du déphasage de celle-ci par rapport à l'horloge locale.

Au prochain chapitre, l'architecture des parties importantes de la solution retenue telles qu'elles ont été implantées dans la puce de démonstration sera présentée.

Chapitre V.
Conception du récepteur

Les équations mathématiques et les analyses théoriques présentées dans les chapitres précédents ont montré que la solution proposée répond aux objectifs recherchés. L'étape suivante est donc de prouver le concept expérimentalement, en implantant les algorithmes présentés dans une puce de démonstration.

Dans ce chapitre, nous allons d'abord présenter et décrire en détail les quelques parties importantes de la solution retenue telle qu'elles sont implantées sur la puce de démonstration.

Pendant la conception de cette puce, à plusieurs reprises, comme il existe plus d'une implantation possible pour un algorithme ou une architecture donnée, nous avons du faire des choix et des compromis. Ces choix et compromis ont évidemment des conséquences sur les performances de la puce de démonstration. La suite de ce chapitre soulignera donc les limites d'opération qui découlent de ces choix et compromis. De plus, nous allons mentionner quelques améliorations futures possibles auxquelles nous avons pu penser.

5.1. Interface

5.1.1. Symbole électrique

Le récepteur est conçu de façon modulaire pour permettre la réutilisation (re-use). L'interface du récepteur exprimée au niveau schématique est montrée à la figure 5.1

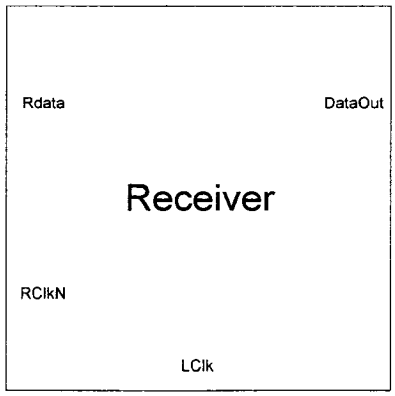

Figure 5.1. Interface du système

5.1.2. Description des entrées - sorties

Le tableau 5.1 liste et décrit brièvement les entrées et sorties du récepteur que nous avons conçu.

Tableau 5.1. Description des entrées et sorties du récepteur

Nom du signal	*Largeur*	*Direction*	*Description*
RClkN	1	Input	Entrée de l'horloge distante
Rdata	8	Input	Entrée des données
LClk	1	Input	Entrée de l'horloge locale
DataOut	8	Output	Sortie de donnée, synchronisée sur LClk

5.1.3. Schéma bloc du récepteur

Le schéma bloc du récepteur tel qu'implanté dans la puce de démonstration est montré à la figure 5.2.

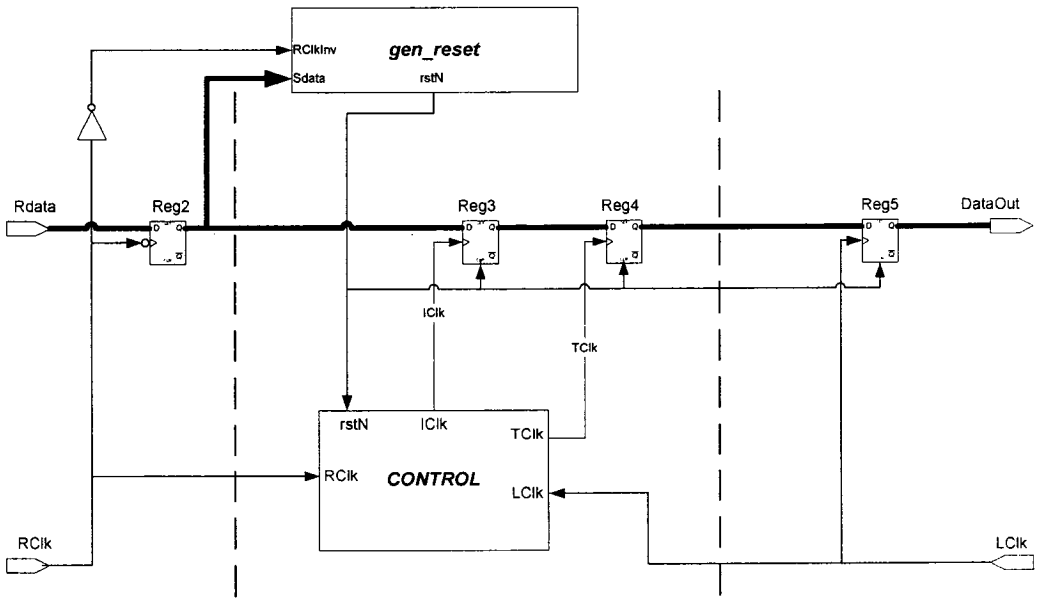

Figure 5.2. Schéma bloc du récepteur

Le système est constitué de deux blocs fonctionnels important, gen_reset et control, en plus des 4 étages de registres qui constituent le pipeline et qui sont synchronisés respectivement sur RClkN, IClk, TClk et LClk.

5.2. Bloc gen_reset

Ce module reçoit les données déjà synchronisées sur RClkN, c'est-à-dire le front descendant de l'horloge distante.

Lorsque la séquence prédéterminée (« 0x0F, 0xF0, 0x0F ») du protocole suggéré est entièrement reçue, le module génère un signal de remise à zéro actif bas, qui, en plus de ré-initialiser les registres du pipeline, demande au module Control de se resynchroniser, c'est-à-dire, d'ajuster l'horloge intermédiaire IClk.

5.2.1. Description

Le tableau 5.2 liste et décrit brièvement les entrées et sorties de ce module.

Tableau 5.2. Description des entrées et sorties du bloc gen_reset

Nom du signal	*Largeur*	*Direction*	*Description*
RClkInv	1	Input	Entrée de l'horloge distante
Sdata	8	Input	Entrée des données
RstN	1	Output	Signal demandant la resynchronisation (actif bas)

Le bloc gen_reset est constitué principalement d'une machine à états, fonctionnant sur RClkInv. Son organigramme est montré à la figure 5.3.

5.2.2. Description des états de la machine à états

Le tableau 5.3 ci-dessous décrit les états utilisés dans l'organigramme de la machine à états du bloc gen_reset.

Tableau 5.3. Description des états de la machine à états de gen_reset

Nom	*Description*
St0	État initial, en attente
St1	Premier état, ayant reçu 0F
St2	Deuxième état, ayant reçu la séquence 0F, F0
St3	Troisième état, ayant reçu la séquence 0F, F0, 0F, génération de demande de synchronisation, rstN='0'

5.2.3. Organigramme de la machine à états

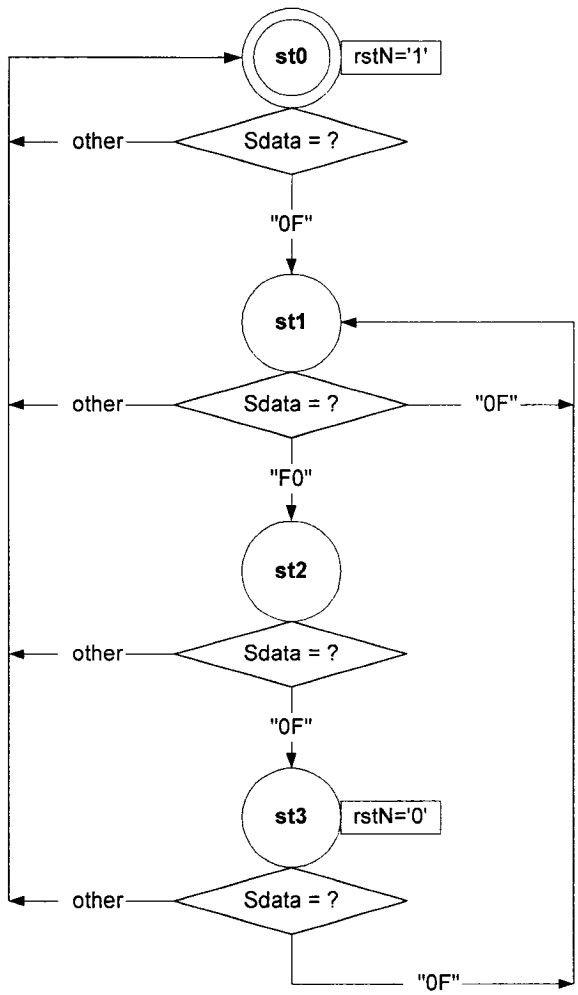

Figure 5.3. Machine à états de contrôle du bloc gen_reset

La machine à états reste en attente à st0 tant qu'elle ne détecte pas le début de la séquence "0F F0 0F" qui demande la resynchronisation. Le signal rstN est désactivé. Une valeur "0F" sur Sdata fait avancer la machine jusqu'à l'état st1 au prochain cycle d'horloge. Ici, si Sdata est de nouveau égal à "0F", nous

resterons à cet état car, quoique la séquence soit brisée, il est possible que la séquence ne soit recommencée qu'avec ce nouveau "0F". Nous retournons alors à st0 au prochain cycle d'horloge avec toute autre valeur de Sdata.

Ensuite, une valeur de "F0" indique que la séquence continue, nous allons à l'état st2.

À st2, seule une valeur de "0F" peut faire avancer à st3. Toute autre valeur de Sdata retourne la machine à l'état st0 car la séquence n'est pas suivie.

Arrivant à l'état st3, nous activons rstN pendant une période d'horloge car la séquence qui demande la resynchronisation est complétée. Ici, une valeur de "0F" sur Sdata indique que la séquence peut être recommencée, nous retournons à l'état st1 au prochain cycle d'horloge. Autrement, la machine à états retournera à st0, en attente d'autres séquences de resynchronisation.

5.3. Bloc *Control*

Dans cette section, nous allons d'abord présenter le fonctionnement du bloc Control. Un diagramme temporel illustre l'enchaînement des états suivi par le bloc Control. Ensuite, nous allons présenter individuellement chacun des blocs qui composent le module Control dont le schéma bloc est présenté à la figure 5.4. Les entrées et sorties du bloc Control sont décrites dans le tableau 5.4.

5.3.1. Schéma bloc

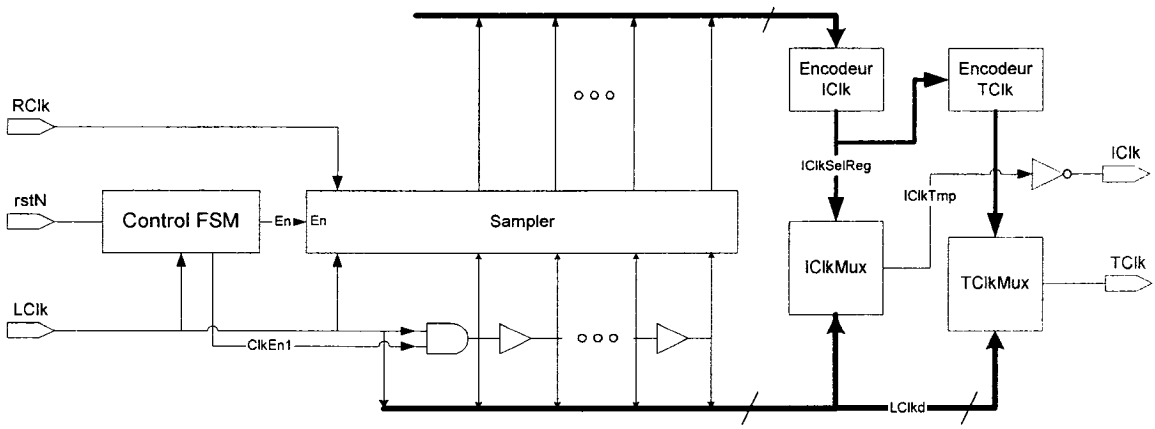

Figure 5.4. Schéma bloc du bloc Control

Tableau 5.4. Description des entrées et sorties du bloc Control

Nom du signal	*Largeur*	*Direction*	*Description*
RstN	1	Input	Signal demandant la resynchronisation
RClk	1	Input	L'horloge distante
LClk	1	Input	L'horloge locale
TClk	1	Output	L'horloge de transition
IClk	1	Output	L'horloge intermédiaire

5.3.2. Description de fonctionnement

Le module Control est constitué d'une machine à états de contrôle « control FSM », d'une chaîne à délai, d'un échantillonneur dans le module Sampler, d'un encodeur de IClk, d'un encodeur de TClk, d'un multiplexeur de sélection IClkMux, et d'un second multiplexeur de sélection TClkMux.

L'ensemble des sorties de la chaîne à délai forme le bus LClkd. Cette ligne à délai permet d'obtenir les différentes phases de LClk. Ces phases sont utilisées

ensuite comme une base de temps pour l'échantillonnage de RClk et aussi pour générer IClk. Les éléments de la ligne à délai sont choisis de manière à couvrir au moins T+ε malgré les variations des procédés de fabrication. Le nombre d'étages est dépendant de la bibliothèque de cellules et du dessin des masques et sera dé0terminé lors des essais de synthèse et de placement et routage.

L'horloge distante RClk est échantillonnée, avec les différentes phases de LClk obtenue avec la chaîne à délai, par l'échantillonneur du module Sampler. Le signal ClkEn1 permet de ne laisser passer qu'une seule propagation d'onde dans LClkd pendant l'échantillonnage pour permettre à l'échantillonneur de se stabiliser et de faire la correction d'erreur. Les résultats sont ensuite chargés dans un registre.

Le résultat d'échantillonnage Result passe ensuite à travers les blocs d'encodeur de IClk et TClk. Ces encodeurs, selon les critères ou algorithmes annoncés dans le chapitre précédent, activent les multiplexeurs de sélection et choisissent IClkTmp et TClk à partir des phases de LClk. IClkTmp est inversé pour finalement devenir IClk.

Ci-dessous, nous allons montrer chacune des parties qui composent le bloc Control.

5.3.2.1. Machine à états de contrôle Control FSM

La machine à états se synchronise sur LClk. Les états de la machine à états sont décrits dans le tableau 5.5 ci-dessous. L'organigramme de cette machine à états est montré à la figure 5.5.

Tableau 5.5. Description des états de la machine à états du Control FSM

État	*Description*
st_rst	État initial, remise à zéro
st_samp	Échantillonner RClk
st_calc	Stabiliser l'échantillonneur Faire la correction d'erreur d'échantillonnage en cas de métastabilité
st_select	Registrer les résultats d'échantillonnage Filtrer les erreurs d'échantillonnage
st_ready	Sélectionner et registrer la sélection de IClkTmp et TClk à partir du résultat d'échantillonnage
st_recep	Le système entre en mode de réception

La machine à états reçoit rstN comme une remise à zéro asynchrone. A toutes fins pratiques, on peut considérer cette machine à états comme un compteur simple à remise à zéro asynchrone, car elle avance automatiquement au prochain état à chaque coup d'horloge. Pour éviter des problèmes de métastabilité, le signal rstN est resynchronisé à l'aide d'un synchroniseur sur LClk.

La machine à états est remise à son état initial (st_rst) lorsque le rstN est à l'état logique 0. Après la levée du signal de reset, nous avançons automatiquement au prochain état st_samp. Le signal ClkEn1 est désactivé après une demi-période pour ne laisser qu'une seule propagation d'onde dans LClkd pendant l'échantillonnage. Le temps alloué à l'échantillonnage étant un cycle, la machine à états avance ensuite à l'état suivant (st_calc). L'échantillonneur Sampler a donc une période pour se stabiliser et pour faire la correction d'erreurs d'échantillonnage due à une éventuelle métastabilité. Pendant ce temps, le signal En est activé en préparation à l'enregistrement des résultats

d'échantillonnage. À l'état suivant, st_select, pendant que En est activé, les résultats d'échantillonnage sont enregistrés. Le système prend une période pour passer les résultats d'échantillonnage dans un filtre passe-bas numérique pour enlever les hautes fréquences. Après une demi-période, le signal ClkEn1 est réactivé. À l'état suivant (st_ready), le système décode le résultat filtré pour déterminer IClkTmp et TClk. IClkTmp est ensuite inversé pour devenir IClk. Les 2 nouvelles horloges IClk et TClk deviennent valides après une demi-période. La réception commence à la prochaine période (état st_recep).

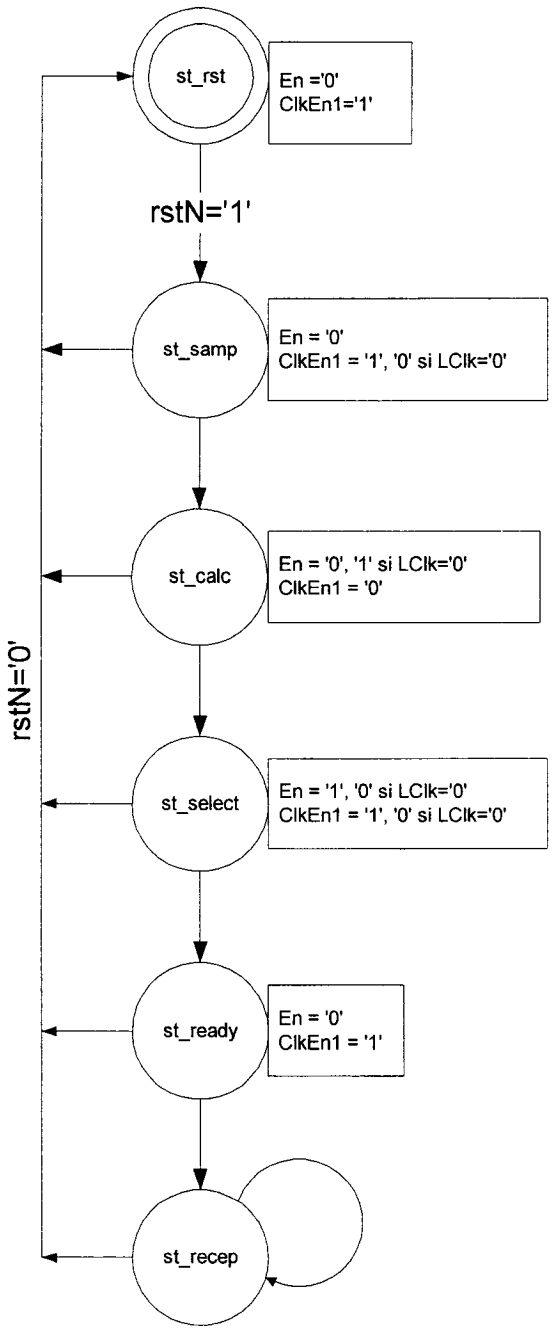

Figure 5.5. Machine à états du bloc Control FSM

Le diagramme temporel suivant illustre l'enchaînement des opérations de la machine à états, suite à une demande de resynchronisation.

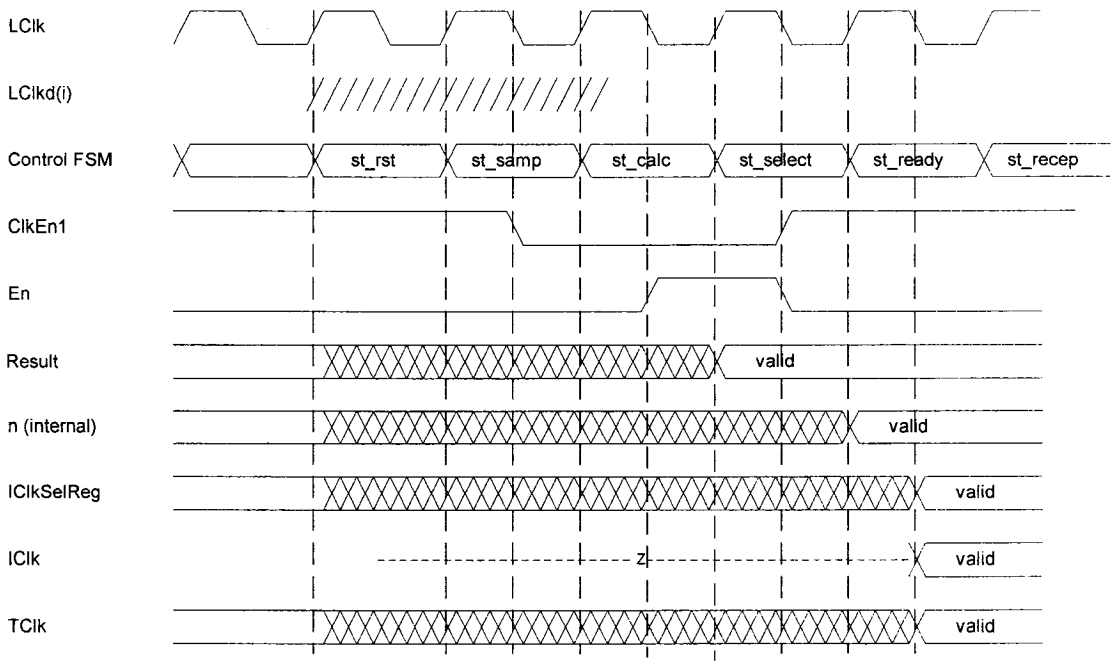

Figure 5.6. Diagramme temporel de la machine à états Control FSM

5.3.2.2. Échantillonneur Sampler

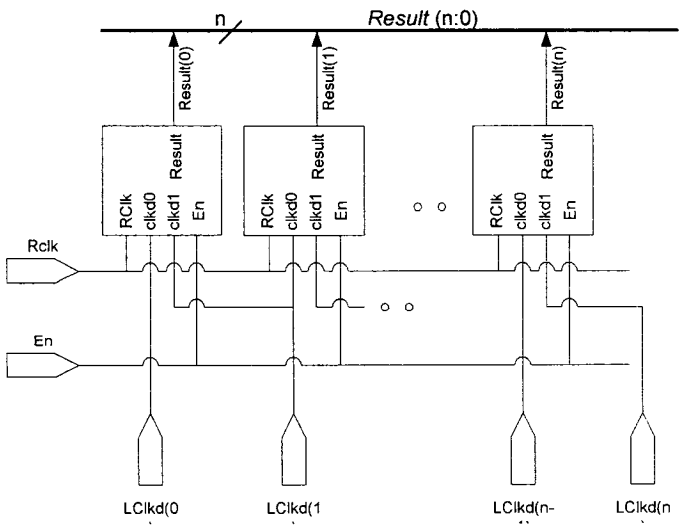

Figure 5.7. Schéma du bloc Sampler

Le bloc Sampler a pour fonction d'échantillonner l'horloge distante RClk avec les différentes phases de LClk. Le résultat d'échantillonnage servira à reconstruire l'image de RClk dans le domaine de LClk.

Le module Sampler montré à la figure 5.7 est un agencement de plusieurs blocs de base. Chaque bloc de base reçoit le signal En pour savoir quand il est permis de registrer les résultats. Chacun de ces blocs de base reçoit 2 phases consécutives de LClk, nommé LClkd(i) et LClkd(i+1) du bus LClkd. L'entrée RClk reçoit l'horloge distante RClk. Les résultats individuels sont ensuite combinés pour former le mot Result(n:0).

Le schéma de la figure 5.8 montre le circuit de base qui est utilisé dans le bloc *Sampler* pour échantillonner RClk.

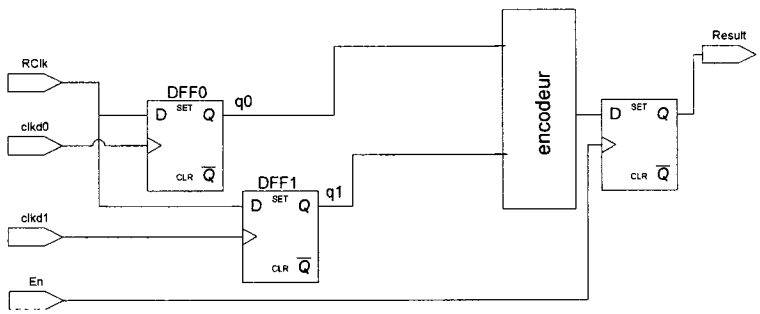

Figure 5.8. Schéma du circuit de base du bloc Sampler

Notons que le circuit présenté aux Figure 5.7. et 5.8. a été implémenté de façon modulaire, pour en faciliter la conception et la vérification. Il est cependant partiellement redondant, car toutes les phases, à l'exception de la première et de la dernière, sont échantillonnées deux fois.

Cette configuration ressemble à des architectures de sur-échantillonnage (oversampling) couramment utilisées dans les circuits de recouvrement

d'horloge aussi appelés CDR (clock and data recovery) [16], [24], [34]. Cependant, ces circuits de recouvrement d'horloge classiques n'utilisent que 3 phases localement autour du front actif de l'horloge pour sur-échantillonner les données et pour ensuite repositionner l'horloge. Dans notre design, nous faisons l'échantillonnage avec plusieurs dizaines de phases couvrant une période complète. Ainsi, nous pouvons reconstruire une image complète de l'horloge distante.

RClk est d'abord échantillonné au front montant de Clkd0 et à nouveau au front montant de Clkd1. Les résultats d'échantillonnage sont passés dans l'encodeur pour la correction d'erreur en cas de métastabilité.

Lorsque l'échantillonnage est fait loin des transitions de RClk, les sorties q0 et q1 des bascules DFF0 et DFF1 sont identiques et deviennent le résultat final qui va être chargé dans un registre. Par contre, lorsque l'échantillonnage est fait autour d'une transition de RClk, ou lorsqu'il y a de la métastabilité, q0 et q1 pourraient être différents et c'est l'encodeur qui détermine la valeur à prendre.

Le tableau 5.6 suivant montre les cas possibles de q0 et q1 ainsi que la décision de l'encodeur.

Tableau 5.6. Cas de figure de l'échantillonnage

Cas	*q0*	*q1*	*Sortie de l'encodeur*	*RClk*
1	0	0	0	Stable à 0
2	0	1	1	Transition de 0 à 1
3	1	0	1	Transition de 1 à 0
4	1	1	1	Stable à 1

Dans notre implantation, selon les considérations pratiques et l'analyse des cas possibles, l'encodeur va attribuer un '1' en cas de métastabilité. En fait, nous nous assurons que les bascules ont suffisamment de temps pour se stabiliser avant l'arrivée du signal En. Par ailleurs, par placement et routage manuels, nous nous assurons que la transition de RClk est assez nette et plus rapide que le délai entre 2 phases consécutives LClkd(i) et LClkd(i+1) de LClk.
Finalement, le résultat après l'encodage est chargé dans un registre sur le front montant du signal En et est disponible via le port Result.

Comme montré à la figure 5.7, ces blocs de base du module Sampler combinent ensuite leur sortie Result(i) pour former le bus de données Result(n:0).

5.3.2.3. Encodeur de IClk

L'enchaînement à l'intérieur de l'encodeur de IClk est montré à la figure 5.9 ci-dessous.

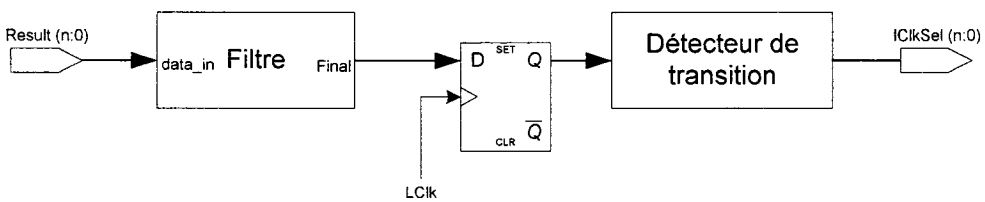

Figure 5.9. Schéma bloc de l'encodeur de IClk

Le bus de données Result(n:0) qui provient du bloc Sampler passe d'abord dans un filtre numérique pour enlever les incertitudes dues aux bruits sur RClk au moment d'échantillonnage car le processus de quantification est sujet à des erreurs produisant des 1 ou 0 isolés tel que discuté plus loin. Le résultat filtré est enregistré au début du cycle d'horloge suivant tel qu'illustré à la Figure 5.6 dans le diagramme temporel de la machine à états Control FSM. Ensuite, le détecteur de transition est simplement un encodeur de priorité qui parcourt

séquentiellement le résultat filtré et registré, et qui cherchera un premier changement de 1 à 0 dans la séquence. La position j d'un tel changement est convertie ensuite en un mot IClkSel(n:0) dans lequel seul le j-ième bit est égal à 0 (actif bas) tandis que les autres bits sont à 1. Ce mot IClkSel(n :0) sera utilisé par le multiplexeur de sélection pour choisir une des phases de LClk qui sera utilisée comme IClkTmp.

L'architecture du filtre est de type propagation en cascade (ripple) et elle est basée sur un algorithme de majorité. En effet, l'ensemble des échantillons représente l'évolution d'un signal (RClk) dans le temps car la fréquence d'échantillonnage est beaucoup plus élevée que celle du signal échantillonné. Par conséquent, un 1 ou un 0 isolé peut-être rencontré lorsque nous parcourons séquentiellement les échantillons enregistrés. Une telle valeur isolée représente donc un bruit à haute fréquence ou une erreur d'échantillonnage. Ainsi, un algorithme de filtrage basé sur la majorité permet d'enlever ces hautes fréquences non-désirées.

Algorithme de filtrage
Soit,

- *data_in*, l'entrée du filtre, l'information à filtrer;
- *Final*, la sortie du filtre, résultat final après filtrage;
- *N,* index du dernier bit du bus *data_in.*

Nous utilisons une variable temporaire FResult de même largeur que le bus data_in, pour contenir le résultat partiel. Nous avons choisi de mettre 0 dans le premier bit de la variable temporaire. La raison de ce choix sera expliquée plus loin. Ensuite, pour chaque bit du bus data_in de 0 jusqu'à N–1, nous formons un mot temporaire de 3 bits tmp(2:0) avec la valeur du FResult(j), data_in(j) et data_in(j+1). La valeur que contient la majorité des 3 bits du mot ainsi formé

deviendra la valeur du bit j+1 du bus FResult. Finalement, nous décalons Fresult d'un bit pour exclure son premier bit et nous transférons le reste du data contenu dans FResult dans Final. Nous avons aussi choisi de mettre un 1 au dernier bit de Final pour avoir la même longueur que les entrées data_in. La raison de ce choix sera aussi expliquée plus loin

Le tableau 5.7. suivant illustre le filtrage pour une longueur de 12 bits.

Tableau 5.7. Exemple du processus de filtrage

bits	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
data_in	0	0	1	0	0	1	1	0	1	1	1	1	
FResult	0												Initial
	0	0											j=0→tmp = 000
	0	0	0										j=1→tmp = 001
	0	0	0	0									j=2→tmp = 010
	0	0	0	0	0								j=3→tmp = 000
	0	0	0	0	0	0							j=4→tmp = 001
	0	0	0	0	0	0	1						j=5→tmp = 011
	0	0	0	0	0	0	1	1					j=6→tmp = 110
	0	0	0	0	0	0	1	1	1				j=7→tmp = 101
	0	0	0	0	0	0	1	1	1	1			j=8→tmp = 111
	0	0	0	0	0	0	1	1	1	1	1		j=9→tmp = 111
	0	0	0	0	0	0	1	1	1	1	1	1	j=10 tmp = 111
Final	0	0	0	0	0	1	1	1	1	1	1	1	Décaler Fresult d'un bit pour exclure son premier bit et mettre un 1 au dernier bit de Final

Dans l'algorithme de filtrage ci-dessus, nous avons choisi de mettre un 0 au début de la variable temporaire FResult et d'ajouter un 1 à la fin du résultat Final afin d'éviter de générer les fausses transitions de 1 à 0 pour le détecteur de transition.

En effet, comme nous ne cherchons que la première transition de 1 à 0, si FResult(0)=1 et la séquence commence avec des 0, il génèrera une fausse transition de 1 à 0 qui déjouera le détecteur. Alors que FResult(0)=0 avec la même séquence, aucune transition n'est générée. Et dans le cas FResult(0)=0 et une séquence qui commence avec des 1, il ne génèrera qu'une fausse transition de 0 à 1 qui va être ignorée par le détecteur. Le tableau 5.8 illustre tous les cas.

Tableau 5.8.Cas de figures avec le premier bit de Fresult

Cas	*FResult (0)*	*Début de la séquence à filtrer*	*premier tmp*	*Problème*
1	0	00	000	Aucun
2	0	11	011	Fausse transition de 0 à 1
3	1	00	100	Fausse transition de 1 à 0
4	1	11	111	Aucun

Ainsi, nous constatons que FResult(0) = 0 est le meilleur choix. Un raisonnement semblable permet de constater que l'insertion d'un 1 à la fin du bus des résultats Final est un choix approprié.

Dans ce qui suit, l'algorithme de filtrage utilisé est donné sous forme de pseudo-VHDL,

```
FResult(0) = '0' ;
(BOUCLE) Pour j allant de 0 à N–1
    tmp := FResult(j) & data_in(j) & data_in(j+1) ;
    cas de tmp
        "000"=> FResult(j+1) = '0' ;
        "001"=> FResult(j+1) = '0' ;
        "010"=> FResult(j+1) = '0' ;
        "011"=> FResult(j+1) = '1' ;
        "100"=> FResult(j+1) = '0' ;
        "101"=> FResult(j+1) = '1' ;
        "110"=> FResult(j+1) = '1' ;
        "111"=> FResult(j+1) = '1' ;
    fin de cas ;
fin de BOUCLE
Final(0 à N–1) <= FResult ( 1 à N ) ;
Final(N) <= '1' ;
```

Due à la longueur du train binaire (bit-stream), le filtrage réalisé avec l'algorithme ci-dessus, selon les premiers essais de synthèse en utilisant la technologie CMOS à 0,18µm de TSMC, nécessitera au moins 4 cycles d'horloge (à 8ns/cycle d'horloge) pour compléter. Afin de pouvoir compléter l'opération dans un seul cycle, nous utilisons une architecture mixte dans laquelle nous montons plusieurs filtres (de type propagation en cascade ci-dessus) selon une architecture de sélection de retenue communément appelée «carry-select». En fait, il nous est possible de faire ce traitement en parallèle même si le filtrage est temporel car les résultats d'échantillonnage sont déjà

emmagasinés dans des registres et sont tous disponibles au moment du filtrage. Évidemment, nous notons une pénalité en termes de surface de silicium avec l'implantation en parallèle. Cependant, les surfaces supplémentaires générées sont bien à l'intérieur de la surface allouée par la Société Canadienne de Microélectronique (CMC) pour la puce de démonstration.

Ainsi, l'algorithme de filtrage proposé ci-dessus est d'abord implanté, avec moins d'entrées, dans un sous-module appelé FILTRE. Ensuite, une unité de base, comme montrée à la figure 5.9, est alors construite avec 2 blocs FILTRE en configuration carry-select: chaque bloc FILTRE reçoit les mêmes données à l'entrée data_in mais une valeur initiale de FResult(0) différente. Le résultat final est sélectionné par l'entrée carry_in.

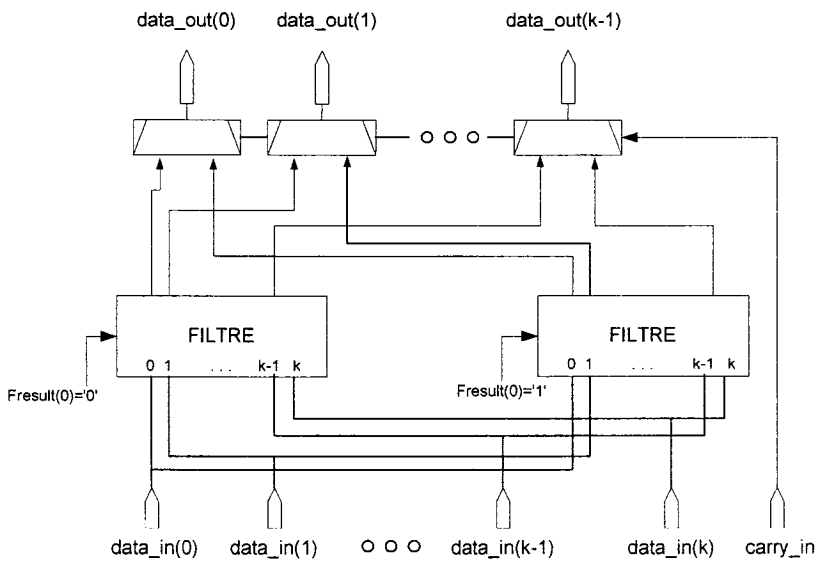

Figure 5.10. Bloc de base du filtre numérique de k bits

Finalement, le filtrage désiré de m bits est accompli en mettant plusieurs de ces blocs de base en cascade.

La figure 5.11 montre un exemple d'arrangement de n blocs de base de k bits pour former un filtre de n * k bits.

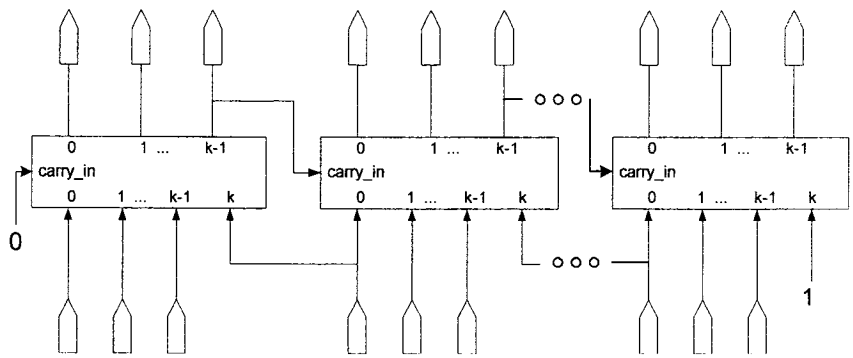

Figure 5.11. Arrangement de blocs de base pour filtrer n x k bits

L'entrée carry_in du premier bloc reçoit un 0, tandis que le dernier bit (k+1) du dernier bloc recevra 1.

Pour cette architecture, le temps requis pour filtrer n * k bits est égal à
Temps de filtrer k bits + n–1 fois le temps de propagation du MUX 2 à 1

Dans la version finale du filtre implanté dans la puce de démonstration, nous avons utilisé 7 blocs de 12 bits. Cette combinaison permet de compléter l'opération de filtrage dans un cycle d'horloge, et elle fait augmenter de 75% la surface du filtre. Cependant, tout le filtre n'occupe qu'environ 3600µm^2 qui est à l'intérieur de la surface allouée pour la puce de démonstration La synthèse confirme que le processus de filtrage de 84 bits se complète en dedans d'un cycle d'horloge de 8 ns.

Notons que le besoin d'introduire une logique de sélection de retenue, pour accélérer le calcul, découle du choix d'origine d'implémenter la logique de majorité suivant une structure en cascade. Cette logique gonfle cependant la complexité et il aurait possiblement été préférable d'implémenter des modules

parallèles indépendants, initialement plus complexes, mais pour lesquels le chemin critique reste court.

5.3.2.4. Encodeur de TClk

Ce module reçoit aussi le résultat de sélection IClkSel(n :0) venant de l'encodeur de IClk et détermine ainsi la phase de LClk choisie comme IClkTmp. Ce module détermine lequel des fronts (montant ou descendant) de LClk permettra d'éviter la métastabilité lors du transfert des données entre IClk et TClk. Finalement, le module indique son choix avec le signal TClkSel.

5.3.2.5. Multiplexeur de sélection IClk

Selon le mot IClkSel reçu, le multiplexeur sélectionne une des phases de LClk, via le bus LClkd. Pour éviter que les différences dans les chemins d'un multiplexeur conventionnel ayant un grand nombre d'entrées changent les caractéristiques temporelles des phases de LClk, une architecture de bus à haute impédance est utilisée, avec des amplificateurs à 3 états (tri-state). L'encodeur aura la responsabilité d'éviter les contentions. Nous avons donc choisi de fermer tous les amplificateurs 3-états une demi-période avant d'activer celui choisi pour éviter toute contention et pour ne pas nuire à la latence. Le circuit du multiplexeur de IClk est montré à la figure 5.12.

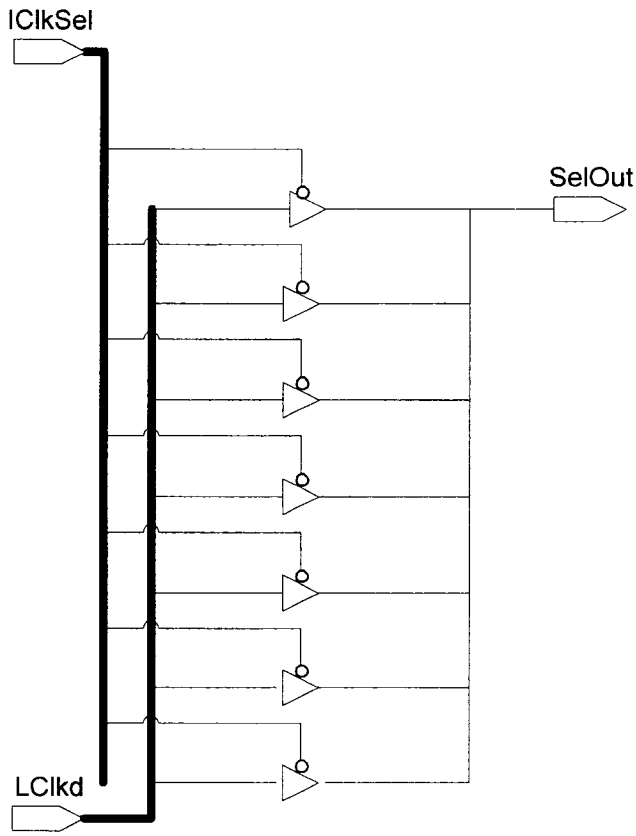

Figure 5.12. Schéma du multiplexeur de sélection de IClk

5.3.2.6. Multiplexeur de sélection de TClk

Ce multiplexeur de sélection de TClk reçoit TClkSel venant de l'encodeur de TClk et reçoit LClk. Selon TClkSel, ce module laisse passer LClk ou LClk inversée pour devenir TClk. Nous avons utilisé une simple porte OU-EXCLUSIF ayant comme entrées LClk et TClkSel car un simple 1 sur TClkSel permet de produire l'inverse du signal LClk sur l'autre entrée.

5.4. Améliorations futures

Étant donné que la réalisation de la puce d'évaluation se veut seulement une preuve de concept, certains choix ou compromis ont été faits lors de

l'implantation des différentes parties du module de synchronisation au récepteur. Ainsi, selon les besoins spécifiques et selon la performance recherchée dans les futures applications, il est possible de trouver d'autres implantations plus robustes, plus performantes pour chacune des parties du module de synchronisation, tout en suivant le même principe de fonctionnement. Nous présentons ci-dessous quelques améliorations possibles.

a. Le bloc Control FSM re-capture automatiquement ICIk

L'implantation choisie assume que le système réussit à échantillonner RCIk et à construire ICIk à chaque demande de resynchronisation. En réalité, quoique la probabilité soit extrêmement faible, il est possible que les signaux reçus soient tellement bruyants que les résultats d'échantillonnage, même après filtrage, contiennent quand même des erreurs d'échantillonnage. Dans cette condition, le système n'arrive pas à reconstruire correctement l'image de RCIk et ne peut pas décider laquelle parmi les phases de LCIk sera ICIk. Pour se protéger lors de telles éventualités et pour rendre le système plus robuste, il est suggéré que le bloc Control FSM puisse décider par lui-même de refaire l'échantillonnage si un tel phénomène se produit, jusqu'à ce qu' ICIk puissent être déterminé de façon fiable.

b. Possibilités de choix de ICIkTmp

L'implantation choisie oblige à avoir ICIkTmp juste après le front descendant de RCIk au moment de synchronisation. Nous pourrions en effet choisir plutôt ICIkTmp telle que la transition sur ce signal soit juste avant le front descendant de RCIk.

c. Utilisation d'une chaîne de délai asservie pour construire les différentes phases de LClk

La chaîne de délai est construite avec des composants d'une bibliothèque de cellules normalisées. D'une puce à une autre du même lot ou d'un lot à un autre, les variations de délai à travers de telles portes sont inconnues et peuvent être assez grandes. Ainsi, les pas de délai et donc le délai total à travers la chaîne ne sont pas connus d'avance. De plus, la longueur de la chaîne de délai est optimisée pour couvrir une période fixe de 8ns dans le pire cas. Par conséquent, l'utilisation du circuit avec d'autres fréquences, notamment des fréquences inférieures à 125 MHz, pourrait rendre le circuit instable. L'utilisation d'une boucle à verrouillage de phase phase-locked-loop (PLL) pour asservir la chaîne permettrait d'adapter le circuit à d'autres fréquences. De plus, il serait possible d'avoir des pas de délais connus en terme de fractions de la période d'opération.

d. Ajout d'un signal annonçant la validité des données resynchronisées

Le système tel qu'implanté prend plusieurs cycles pour l'auto-calibrage. Pendant ce temps, le pipeline, quoiqu'il soit déjà remis à zéro par précaution, peut contenir des données invalides. Une amélioration possible est de générer un signal de contrôle pour indiquer la validité des données présentes dans le pipeline.

e. Élimination des bascules redondantes de l'échantillonnage

Comme mentionné ci-dessus, à cause de la réplication du module de base, chaque phase LClkd(i), à l'exception de la première et la dernière, est échantillonnée 2 fois. Il est effectivement possible d'enlever ce niveau de hiérarchie et d'échantillonner ensuite chaque phase avec une seule bascule D.

f. Architecture du filtre

Sachant que le signal sera stable autour des transitions et que celles-ci seront nettes, grâce aux précautions prises lors du placement et routage, une autre architecture de filtre sans propagation a été proposée en utilisant des fenêtres plus larges (5 ou 7 bits) centrées autour de chaque bit de la séquence à filtrer. En effet, une fenêtre plus large autour de chaque bit nous fournit suffisamment d'information pour traiter le bit central de chaque fenêtre indépendamment des autres. Ainsi, nous n'avons plus de chemin critique dû aux propagations des résultats.

g. Élimination de la resynchronisation et la gigue rapide au moment de resynchronisation

Même si la resynchronisation peut se faire pendant un temps mort entre 2 messages, le flot de données est quand même arrêté. De plus, lorsqu'un message est trop long, la gigue lente sur l'horloge distante RClk ne sera plus négligeable et la tolérance au biais de synchronisation du système sera affectée. Il a été suggéré que le système calcule continuellement la moyenne des temps d'arrivée de RClk et ajuste automatiquement IClk sans arrêter le flot de données. De plus, étant une variable aléatoire de moyenne nulle comme nous l'avons définie précédemment, la gigue rapide sur l'horloge distante au moment de resynchronisation $G2_{RCLk_0}$ qui affecte la détermination de la phase et donc la tolérance du système pourrait être aussi éliminée.

5.5. Conclusion

Dans ce chapitre, nous avons décrit en profondeur quelques parties importantes du module de synchronisation, telle qu'elles sont implantées dans la puce de démonstration. Nous avons aussi discuté des améliorations possibles pour augmenter la robustesse de l'architecture choisie. Dans le prochain chapitre, il sera question de la réalisation de la puce de démonstration, ainsi que les différents travaux effectués pour valider le design.

Chapitre VI.
Réalisation, validation et résultats

6.1. Introduction

Dans le chapitre précédent, nous avons présenté l'architecture du récepteur telle qu'implantée dans la puce de démonstration.

Dans ce chapitre, nous présentons d'abord les étapes qui ont mené à la réalisation de la puce. Ensuite, nous enchaînons avec les divers travaux de validation. Dans la partie relative à la validation fonctionnelle du design en VHDL, nous expliquons aussi comment nous avons modélisé les perturbations lentes et rapides pour valider le design du récepteur. Par la suite, nous présentons les différents aspects de la validation expérimentale avec le prototype. Il y sera question des spécifications de tests ainsi que les résultats obtenus.

6.2. Les étapes de design

La réalisation de la puce de démonstration en soi passe par plusieurs étapes. D'abord, il y a le design fonctionnel pendant lequel l'architecture choisie est modélisée avec un langage de modélisation de matériel tel que le VHDL. Nous effectuons des simulations fonctionnelles pour confirmer la fonctionnalité du circuit. Ensuite, nous faisons la synthèse logique pour transformer la description matérielle en une description de circuit avec des portes logiques élémentaires. Nous faisons le placement et routage pour créer un dessin des masques à partir de cette description de circuit (netlist). Nous effectuons aussi les vérifications statiques de délai. Finalement, nous exécutons les vérifications finales (DRC –

LVS) avant d'envoyer le dessin des masques à la fabrication. Ci-dessous, nous allons présenter la méthodologie utilisée pour chacune des étapes énumérées ci-dessus.

6.2.1. *Design fonctionnel*

Le design fonctionnel a été fait suivant l'approche du bas vers le haut (bottom-up). Nous sub-divisons le design de façon à faire de la réutilisation (reuse) et en fonction du placement et routage hiérarchique prévu. À chaque niveau, la fonctionnalité de l'étage ou du module est vérifiée avec un banc d'essai avant d'être intégrée dans l'ensemble. En particulier, nous allons expliquer les détails de la validation fonctionnelle du récepteur plus loin dans ce chapitre.

Finalement, des simulations VHDL sont faites sur le design au niveau de la puce pour nous assurer qu'il rencontre toutes les spécifications.

6.2.2. *Synthèse logique*

Tel que mentionné dans la section 5.3.2.3 du chapitre précédent, les essais de synthèse, en plus de transformer les descriptions en langage de modélisation matérielle en des descriptions de circuit avec portes élémentaires, ont révélé l'existence d'un chemin critique trop long. Des modifications architecturales ont été faites pour pouvoir rencontrer les contraintes temporelles.

La synthèse logique a été faite aussi de façon hiérarchique pour construire et pour préserver les structures de base qui sont ensuite réutilisées. Ainsi, nous avons préservé une certaine hiérarchie de design pour permettre le placement et routage hiérarchique.

Nous avons aussi essayé d'introduire une chaîne de balayage (scan chain) à cette étape. Une telle chaîne est une structure de test classique. Cependant, l'utilisation de lignes à délai pour dériver les différentes phases d'horloge complexifie grandement l'insertion d'une chaîne de balayage, au point d'y renoncer tout en réalisant la perte d'observabilité qui en découle.

6.2.3. Placement et routage

Le placement et routage a été fait de façon hiérarchique selon l'approche du bas vers le haut. Le placement et routage hiérarchique a pour but de préserver les caractéristiques des blocs qui seront réutilisés, en plus de permettre de mieux gérer les changements mineurs par ECO (Engineering Change Order) plus tard.

En général, nous utilisons le placement et routage automatique. Cependant, outre les blocs hiérarchiques qui doivent être préservés, certaines parties critiques comme les chaînes à délai et les blocs du filtre numérique ont nécessité des interventions et des optimisations manuelles de placement à l'aide de l'outil Design Planner™. Pour éviter que l'outil automatique défasse les placements optimisés, plusieurs techniques avancées dans l'intégration hiérarchique sont utilisées: changement de l'attribut des cellules pour préserver les placements, placement par script, aplatissement sélectif (FLATTEN), création de groupe et des régions de placement, placement ECO.

D'autre part, comme le design est limité par les entrées-sorties (IO bound), nous nous sommes prémunis contre les problèmes d'interférence électromagnétique en utilisant des techniques de routage avancées comme l'espacement supplémentaire autour des horloges. De plus, nous avons aussi suivi la méthodologie de CMC pour la détection des interférences électromagnétiques (crosstalk).

6.2.4. *Analyses de délais statiques*

Après le placement routage, en vue de l'analyse statique des délais avec PrimeTime™, nous avons inclus les résistances et capacitances parasites extraites à partir du dessin des masques (layout). Nous avons utilisé la bibliothèque à délai maximum (worst case) pour vérifier les conditions de pré-positionnement tandis que les conditions de maintien ont été vérifiées avec la bibliothèque à délai minimum (best case).

6.2.5. *L'intégration*

L'intégration a été faite de façon hiérarchique. En effet, nous générons un GDS (fichier de format binaire standard pour représenter les designs de circuits intégrés) pour chacun des trois principaux blocs hiérarchiques: le récepteur, l'émetteur, et la puce. Avec l'outil de layout de Cadence, nous unissons les GDS est générons un seul GDS qui contient le démonstrateur.

Ensuite, nous faisons les vérifications des règles géométriques de design DRC (Design Rule Check) suivant la méthodologie de CMC. Cette étape a pris plusieurs itérations et a requis la modification du dessin de masques, car il y a des erreurs de modélisation dans la bibliothèque et ces erreurs ne peuvent pas être détectées plus tôt avec les vérifications DRC utilisant les modèles dites boîtes noires (black-box) de la bibliothèque. Le placement et routage hiérarchique se montre utile et efficace contre ces changements de dernières minutes.

Finalement, nous avons aussi exécuté les vérifications LVS (Layout versus Schematic) selon la méthodologie de design de CMC pour vérifier que le design résultant du dessin des masques correspond au design logique avant d'envoyer

le dessin des masques à la fabrication. Cette étape est très manuelle et elle a pris beaucoup de temps.

6.3. Simulation fonctionnelle du récepteur

Pour vérifier que le récepteur répond aux performances recherchées, c'est à dire qu'il est capable de tolérer une combinaison de gigues lente et rapide, un script en Perl (Annexe I) est créé pour générer le banc d'essai (test bench) de l'émetteur en langage de modélisation matérielle VHDL. À chaque cycle de simulation, nous calculons d'abord le temps d'arrivée de l'horloge soumise à une gigue lente. Ensuite nous y superposons une gigue rapide. Les 2 gigues sont calculées de façon indépendante.

La gigue lente a pour effet de faire dériver l'horloge de façon régulière et incrémentale d'un cycle à un autre, et ce, dans une même direction. Dans le script, la direction de la gigue lente peut être programmée pour faire avancer ou retarder l'horloge. À cause de l'accumulation de la gigue lente, l'horloge peut donc être retardée (ou avancée) de plus d'un cycle à la fin de la simulation. Le script peut être facilement adapté pour générer un banc d'essai plus long dans lequel la gigue lente fait dériver l'horloge de plusieurs cycles.

Quant à la gigue rapide, le générateur de nombre pseudo-aléatoire du langage Perl est utilisé pour créer et moduler la nature arbitraire des sauts rapides. En effet, à chaque cycle de simulation, la gigue rapide est modulée par des sauts rapides et aléatoires allant jusqu'à 1.5 ns avant ou après le point d'ancrage. Ainsi, la pire condition rencontrée lorsque la gigue rapide ajoute 1.5 ns à la gigue lente à un cycle donné et fait un saut qui enlève 1.5 ns de la gigue lente au cycle suivant. Dans ce cas, l'horloge fait donc un saut total instantané allant jusqu'à 3 ns.

Notons que le script génère aussi automatiquement la séquence de resynchronisation après un certain nombre (programmable) de cycles de simulation.

L'algorithme utilisé dans le générateur de banc d'essai est expliqué ci-dessous en détail.

Les variables suivantes sont importantes dans la génération des vecteurs de test:

n nombre de cycles de simulation

reset nombre de cycles entre deux séquences de resynchronisation.

Pour chaque cycle de simulation, le programme calcule:

Une dérive lente équivalente à 10 (ns) / n. (10 ns étant l'accumulation totale de la gigue lente à la fin de l'exécution du banc d'essai. Cette valeur est choisie arbitrairement et elle est facilement paramétrisable)

Une dérive rapide

- la fonction *rand()* génère un nombre aléatoire entre 0 et 1.
- Soustraire 0.5 pour obtenir un nombre aléatoire centré autour de 0, c'est à dire dans l'intervalle [–0.5, 0.5]
- Multiplier la valeur obtenue par 3 (ns) pour avoir le saut causé par la dérive rapide. Ainsi, la plage de tolérance de gigue rapide testée par le banc d'essai est de –1.5 ns à 1.5 ns autour du point d'ancrage.

Les 2 types de dérives sont ensuite combinés et appliqués sur l'horloge RClkN

La valeur des données est aussi générée par le générateur de nombre aléatoire de Perl, pour l'intervalle 0 – 255 (8 bits). Finalement, pour faciliter l'observation et l'analyse lors des tests, le programme évite que des valeurs identiques se trouvent dans deux cycles consécutifs.

Nous avons d'abord effectué les simulations avec les descriptions de circuits en VHDL. Ensuite, nous avons refait les simulations avec les descriptions de circuits (netlist) avec les portes logiques obtenues après la synthèse (gate-level). Les simulations montrent que les données resynchronisées à la sortie du récepteur sont intègres et dans le bon ordre.

6.4. Validation expérimentale

Un démonstrateur, sous forme de circuit intégré, réalisé avec la technologie 0,18µm de TSMC[1] a été soumis à la fabrication chez CMC pour valider expérimentalement le concept. Dans cette section, nous allons présenter l'architecture du démonstrateur ainsi que la carte adaptatrice conçue pour tester la puce.

6.4.1. Démonstrateur

Le schéma bloc du démonstrateur fabriqué chez CMC est montré à la figure 6.1. Les entrées et sorties du démonstrateur sont présentées au tableau 6.1.

Étant donné que les sauts causés par la gigue rapide sont difficiles à simuler de l'extérieur, un circuit émetteur est ajouté au démonstrateur. Le bloc d'émetteur sert donc à moduler les gigues sur les signaux envoyés au récepteur.

De plus, nous avons ajouté des multiplexeurs sur les chemins qui relient le récepteur à l'émetteur pour faciliter le déverminage et l'exécution des tests. En effet, comme la puce de démonstration est divisée en deux parties distinctes qui forment le chemin de test, les multiplexeurs permettent de court-circuiter le premier étage (émetteur) et d'exercer directement le récepteur dans le cas où l'émetteur ne fonctionnerait pas. Nous avons donc créé des chemins simples (par exemple avec LClkInIO comme entrée et LClkOutIO comme sortie) pour vérifier si l'alimentation est bien connectée et si la puce répond aux stimuli.

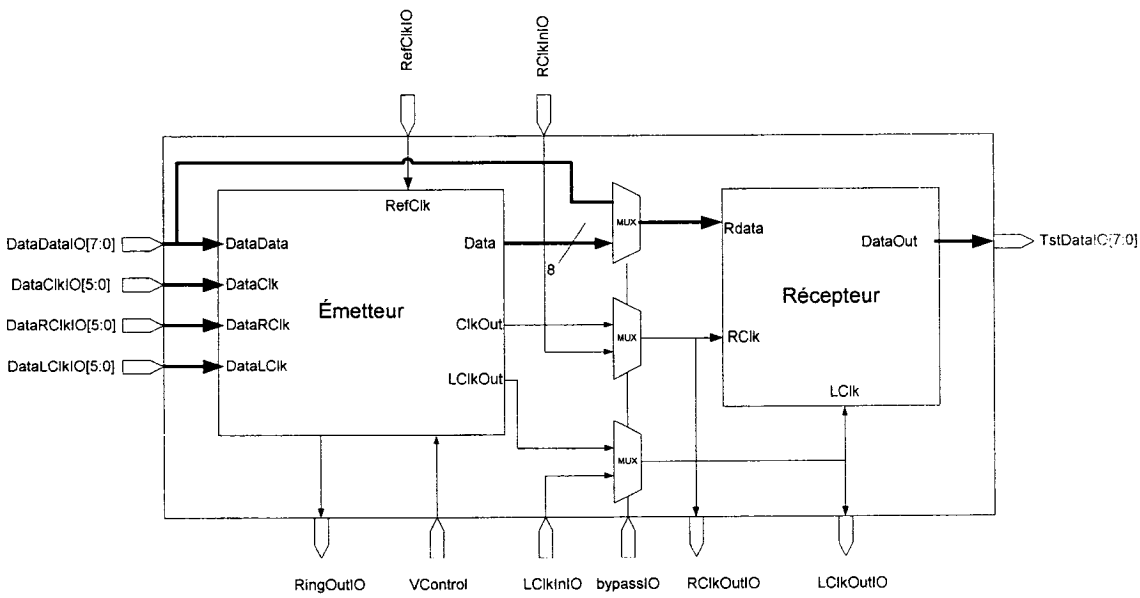

Figure 6.1. Schéma bloc du démonstrateur

Tableau 6.1. Description des entrées et sorties du démonstrateur

Nom du signal	*Largeur*	*Direction*	*Description*
DataDataIO	8	Input	Données à envoyer à l'émetteur
DataClkIO	6	Input	Valeur servant à moduler la gigue sur les 4 bits LSB de la donnée.
DataRClkIO	6	Input	Valeur servant à moduler la gigue sur RClk
DataLClkIO	6	Input	Valeur de déphasage entre RClk et LClk
Vcontrol	1	Input	Tension analogue pour asservir la chaîne à délai servant de base temporelle à l'intérieur de l'émetteur
BypassIO	1	Input	Sélectionne les signaux externes ou ceux générés par l'émetteur
LClkInIO	1	Input	Horloge LClk externe
RClkInIO	1	Input	Horloge RClk externe
RefClkIO	1	Input	Horloge de référence du système.
RClkOutIO	1	Output	RClk généré par l'émetteur observable
LClkOutIO	1	Output	LClk généré par l'émetteur observable
RingOutIO	1	Output	Sortie de la chaîne à délai observable
TstDataIO	8	Output	Données resynchronisées sur LClk par le récepteur

6.4.2. L'émetteur

Le schéma bloc de l'émetteur est montré à la figure 6.2.

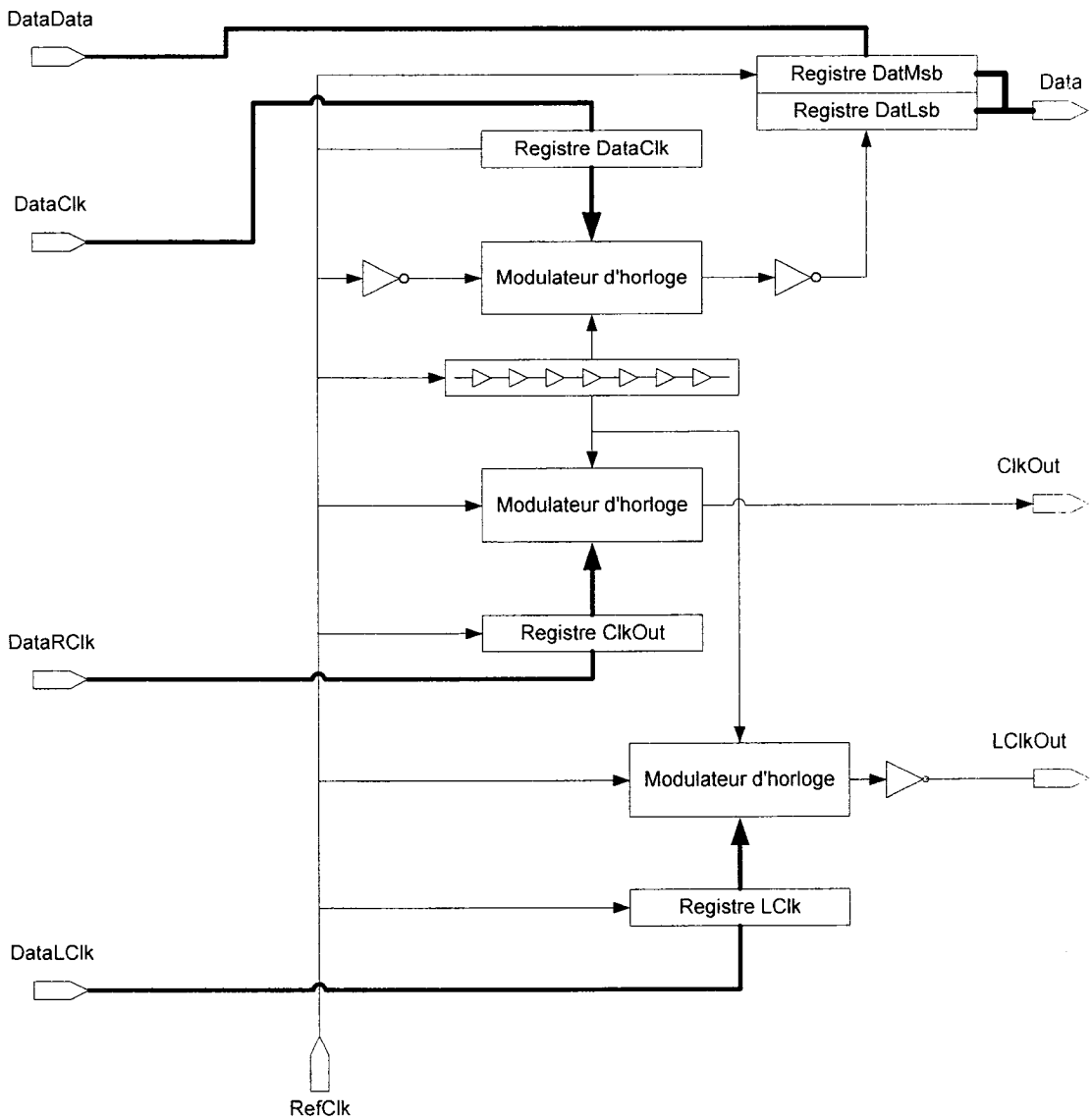

Figure 6.2. Schéma bloc de l'émetteur

6.4.2.1. Description de fonctionnement

Les entrées et sorties de l'émetteur sont présentées dans le tableau 6.2.

Tableau 6.2. Description des entrées et sorties de l'émetteur

Nom du signal	*Largeur*	*Direction*	*Description*
DataData	8	Input	Données à envoyer à l'émetteur
DataClk	6	Input	Valeur servant à moduler la gigue sur les 4 bits LSB de la donnée.
DataRClk	6	Input	Valeur servant à moduler la gigue sur RClk
DataLClk	6	Input	Valeur de déphasage entre RClk et LClk
RefClk	1	Input	Horloge de référence du système
Data	8	Output	Données modulées à envoyer à l'émetteur
ClkOut	1	Output	Horloge avec gigue qui deviendra RClk
LClkOut	1	Output	Horloge déphasée qui deviendra LClk

L'entrée RefClk est l'horloge de référence de l'émetteur. Toutes les données venant de l'extérieur sont enregistrées dans les registres respectifs. Le bloc Modulateur d'horloge est utilisé pour générer une onde ayant les fronts montants en phase avec RefClk et le temps d'arrivée du front descendant programmable à chaque cycle selon le mot à l'entrée. Ainsi, branchée à la sortie ClkOut du bloc émetteur au niveau de la puce, l'horloge distante RClk a donc les fronts descendants programmables. Aussi, nous inversons la sortie d'un autre bloc Modulateur d'horloge pour créer l'horloge ayant le front montant programmable qui deviendra LClk (ce signal étant branché à la sortie LClkOut au niveau de la puce). Quant aux données à envoyer au récepteur, les bits les plus significatifs sont envoyés avec le front montant de RefClk tandis que les

bits les moins significatifs sont soumis à un effet de gigue en modulant l'horloge qui les envoie au récepteur. En effet, les bits les moins significatifs sont transmis avec une horloge ayant les fronts descendants en phase avec RefClk et les fronts montants modulés. Pour ce faire, nous modulons d'abord l'horloge RefClk inversée et nous inversons la sortie du modulateur d'horloge.

6.4.3. Carte adaptateur pour le testeur IMS

Nous avons prévu d'utiliser le testeur IMS du groupe de recherche en microélectronique (GRM) de l'École Polytechnique de Montréal pour valider les stimulus de test. Il était prévu dans un deuxième temps d'effectuer des tests de performance aux fréquences désirées avec un autre testeur IMS compatible avec le premier, mais plus performant, localisé dans les locaux de la CMC à Kingston. En effet le testeur du GRM ne peut fonctionner qu'à une vitesse maximale de 100MHz alors que l'autre peut opérer à 200 MHz. Rappelons que le système visé devait opérer à 125 MHz

La puce de démonstration utilise un boîtier à haute performance CFP-80 spécialement conçu pour des tests à fréquences élevées et choisi selon les recommandations de CMC. Cependant, la carte adaptatrice de test pour ce boîtier, CFP80TF, disponible déjà chez CMC et utilisé jusqu'alors seulement pour tester les circuits analogiques, ne permet d'utiliser que quelques signaux utiles et ne répond donc pas à nos besoins en terme de nombre de signaux utiles. De plus, l'adaptateur existant impose une assignation spécifique de broches qui n'est pas convenable pour notre puce. Particulièrement, les broches d'alimentation ne sont pas bien réparties et ne sont pas disponibles de chaque côté de la puce.

Nous avons donc fait de l'ingénierie inverse (reverse engineering) et nous avons conçu une carte adaptatrice (Annexe II) qui sera vue par le testeur IMS comme une carte standard de type « Open XL IO ». Cette carte adaptatrice est hautement configurable et elle s'adapte à d'autres puces utilisant un boîtier CFP-80 car nous n'imposons pas de broches pré-assignées et que nous pouvons avoir jusqu'à 64 signaux utiles, c'est-à-dire le nombre maximal de canaux qui se trouve sur IMS. Par ailleurs, la carte adaptatrice a été utilisée par d'autres étudiants dès son apparition.

6.4.4. Assignation des broches

L'assignation des broches de la puce de démonstration ainsi que les canaux du testeur IMS correspondants utilisés pour valider la puce sont présentés dans l'Annexe III.

6.5. Les tests

Dans cette section, nous allons présenter en premier lieu les spécifications de tests, ensuite nous allons faire part des résultats obtenus.

6.5.1. Spécifications de tests

La puce de démonstration peut être divisée en trois sections importantes: émetteur, récepteur et section analogique. Chacune des sections peut être validée séparément. Nous présentons ci-dessous les étapes de validation expérimentale ainsi que leurs spécifications.

Comme nous utilisons la carte adaptatrice pour valider la puce, nous devons d'abord vérifier que la carte adaptatrice est utilisable.

a. Déverminage de la carte adaptatrice

Objectif: Vérifier la connectivité de la carte adaptatrice et vérifier la correspondance entre les canaux de IMS et les futures broches de la puce.

- Configurer la carte sans mettre la puce à tester;
- Configurer en entrée tous les canaux de IMS associés à un signal utile de la puce, c'est-à-dire utiliser le testeur IMS seulement comme générateur d'onde;
- Injecter différentes formes d'onde;
- Observer les signaux injectés sur les plots (*pad*) de soudures (où seront soudées les broches de la puce).

b. Déverminage de la puce

Objectif: Vérifier que la puce répond aux stimuli, en utilisant le circuit de court-circuit (bypass)

- BypassIO = '1';
- Injecter une horloge dans RClkInIO et/ou LClkInIO;
- Observer les signaux RClkOutIO et/ou LClkOutIO respectivement.

c. Section analogique

Objectif: Déterminer la valeur de Vcontrol pour obtenir une fréquence d'oscillation 60MHz et 125MHz. L'oscillation est observable à travers RingOutIO à 60MHz et 125MHz.

- Tracer la courbe Fréquence – valeur de *Vcontrol*

d. Section Émetteur

Objectif: observer et valider le bon fonctionnement des circuits de modulation dans l'émetteur.

- *BypassIO* = '0' ;
- Injecter une horloge sur *RefClkIO*;
- Injecter les données de simulation sur *DataRClkIO:*
  - une séquence linéaire et monotone, c'est à dire une séquence dans laquelle les valeurs incrémentent de 1 à chaque cycle d'horloge;
  - des séquences avec des sauts.
- Observer/mesurer *RClkOutIO*;
- Injecter les données de simulation sur *DataLClkIO*:
  - une séquence linéaire et monotone;
  - des séquences avec des sauts;
- Observer/mesurer *LClkOutIO*;
- Injecter les données de simulation sur *DataDataIO*;
- Injecter les données de simulation sur *DataClkIO*.

e. Récepteur

Objectif: Observer et valider le bon fonctionnement du circuit de resynchronisation, du chemin de réception sous conditions statiques, c'est-à-dire avec les horloges sans gigues.

- BypassIO = '1';
- Injecter une horloge sur RClkInIO;
- Observer RClkOutIO;
- Injecter une horloge sur LClkInIO, avec un certain déphasage par rapport à RClkInIO;
- Observer LClkOutIO;
- Injecter des données de simulation sur DataDataIO, ayant des séquences de resynchronisation;
- Observer TstDataIO;
- Changer le déphasage entre RClkInIO et LClkInIO et refaire le test.

f. Mesure de performance

Ces mesures de performance doivent être exécutées avec le IMS-ATS2 de CMC à Kingston, Ontario.

- Refaire toutes les étapes de déverminage précédentes à 125 MHz;
- Générer des séquences de données plus longues et collecter les statistiques de transmission (taux d'erreur).

6.5.2. Résultats de tests

6.5.2.1. Déverminage de la carte adaptatrice de l'IMS

Après avoir configuré la carte, nous vérifions à l'aide du multimètre ses connexions et sa distribution des alimentations.

Ensuite, nous vérifions la configuration de la carte adaptatrice avec un programme de test conçu spécifiquement pour le déverminage. Cet exercice nous a permis d'apprendre et de nous familiariser avec la programmation et le contrôle du IMS. Nous installons donc la carte sans la puce et nous programmons le IMS de sorte que cette dernière envoie des formes d'ondes différentes sur chaque broche de signaux utiles de la puce. À l'aide de l'oscilloscope, nous avons observé que les connexions sont faites correctement jusqu'où se trouveront les broches de la puce.

Conclusion: la carte adaptatrice est bien configurée.

Cependant, nous avons aussi découvert que le socle adaptateur fait par CMC présente un désalignement lorsque nous l'installons sur notre carte adaptatrice. Nous avons donc du procéder à des ajustements, et ce en collaboration avec CMC.

6.5.2.2. Déverminage de la puce

Après avoir mis la puce en place sur la carte adaptatrice et branché les alimentations, nous programmons le IMS avec la configuration du test régulier et nous procédons comme mentionné dans la spécification. Nous n'observons aucune activité aux sorties LClkOutIO et RClkOutIO, et ce avec plusieurs des échantillons reçus.

6.5.2.3. Test de section analogique

Nous varions la tension de contrôle Vcontrol dans la plage permise, c'est-à-dire entre 0 et 3,3V. Nous observons que la sortie RingOutIO est collée à 1, et ce avec plusieurs différents échantillons.

6.5.2.4. Autres tests

Étant donné les résultats mentionnés ci-dessus, nous procédons donc à d'autres tests de caractérisation.

D'abord nous mesurons les impédances entres les alimentations des différents échantillons. Sur les 6 puces reçues, 4 présentent une impédance qui varie de 1.0 à 8.0 Ohms entre VDD 1.8V est VSS. Les 2 autres présentent une impédance de l'ordre de 45 à 55 Ohms. L'impédance entre VDD 3,3V et VDD 1.8V ainsi que celle entre VDD 3,3V et VSS sont dans les attentes, c'est-à-dire de l'ordre de plusieurs centaines de kilo-ohms à des méga-ohms.

Nous ouvrons aussi les boîtiers et inspectons au microscope les fils de liaison (bonding wire). Nous observons que, sauf quelques exceptions, ces fils de liaison de tous les échantillons ont été faits correctement.

Nous avons aussi comparé l'orientation de la puce dans le boîtier pour nous assurer que la puce n'est pas mal orientée. Nous constatons que l'orientation de la puce est correcte. En effet, à l'aide du multimètre, nous avons pu vérifier que les alimentations sont à leur broche respective.

Ensuite, nous vérifions le fonctionnement des plots d'entrées et de sorties. Nous savons que les plots d'entrées et de sorties sont protégées contre les décharges électrostatiques (ESD) par des diodes en polarisation inverse et que les plots d'entrées et de sorties sont alimentées à 3,3V. Ainsi, nous appliquons très brièvement une tension de l'ordre de 4,2V à 4,5V (c'est à dire une tension plus grande que l'alimentation 3,3V plus un seuil de diode de 0,7V) à chaque entrée ou sortie et nous observons le courant. Si la diode de protection contre les décharges électrostatiques (ESD), branchée en polarisation directe entre le signal et l'alimentation, est présente et fonctionnelle, elle serait en conduction et un grand courant de court-circuit va circuler. Par la suite, nous vérifions la présence de la deuxième diode, branchée en polarisation directe entre VSS et le signal en appliquant une tension de –1,0V (c'est à dire une tension plus basse que le 0V moins un seuil de diode 0,7V) à chaque entrée ou sortie. S'il y a un courant de court-circuit qui y passe, la diode est présente et fonctionnelle.

Les courants de court-circuit sont de l'ordre de 45 à 70mA, un courant assez fort pour provoquer un échauffement important des fils de liaison. Pour éviter que l'échantillon ne surchauffe lors des tests, nous n'appliquons les tensions mentionnées ci-dessus que très brièvement. Nous nous assurons que les plots ou les fils de liaison ne seront pas détruits, même en refaisant les tests plusieurs fois. Nous avons répété ces tests et obtenu essentiellement les mêmes résultats.

Les résultats de ce test effectué sur un des 2 échantillons ayant 45 Ohms entre VDD 1.8V et VSS sont présentés dans le tableau 6.3. Nous indiquons notamment s'il y a ou non un courant en polarisation directe.

Tableau 6.3. Résultats des tests de continuité

# broche	Type	Courant en polarisation directe	
		V= –1,0V	V= 4,2V
2	Entrée	oui	non
3	Entrée	oui	non
5	Entrée	oui	non
6	Entrée	oui	non
8	Entrée	oui	non
9	Entrée	oui	non
13	Sortie	oui	oui
15	Entrée	non	non
24	Sortie	oui	oui
26	Sortie	oui	oui
28	Sortie	oui	oui
30	Sortie	oui	oui
32	Sortie	non	non
34	Sortie	oui	oui
38	Sortie	non	non
39	Entrée	non	non
40	Entrée	non	non
42	Entrée	oui	non
43	Entrée	oui	non
45	Entrée	oui	non

# broche	Type	Courant en polarisation directe	
		V= –1,0V	V= 4,2V
47	Entrée	oui	non
49	Entrée	oui	non
51	Entrée	oui	non
53	Entrée	oui	non
55	Entrée	oui	non
57	Entrée	oui	non
59	Entrée	oui	oui
61	Entrée	oui	oui
64	Entrée	oui	non
65	Entrée	oui	non
67	Entrée	oui	non
68	Entrée	oui	non
70	Entrée	oui	non
71	Entrée	oui	non
73	Entrée	oui	Non
74	Entrée	oui	non
76	Entrée	oui	non
77	Entrée	oui	non
79	Entrée	oui	non
80	Entrée	oui	non

Les résultats du tableau 6.3 montrent qu'il y a probablement un problème dans la conception des plots ou dans la fabrication. En effet, l'absence du courant en polarisation directe des structures ESD chez plusieurs entrées ou sorties indique que leurs diodes de protection sont absentes ou ne fonctionnent pas.

Ensuite, nous traçons les courbes I-V des plots qui réagissent favorablement au test précédent (V=4.2V et V=–1.0V). Les résultats sont montrés ci-dessus. Il est à noter que nous effectuons surtout des mesures autour des tensions d'alimentation plus ou moins un seuil de diode, c'est-à-dire plus ou moins 0.7V autour de 0V et 3,3V.

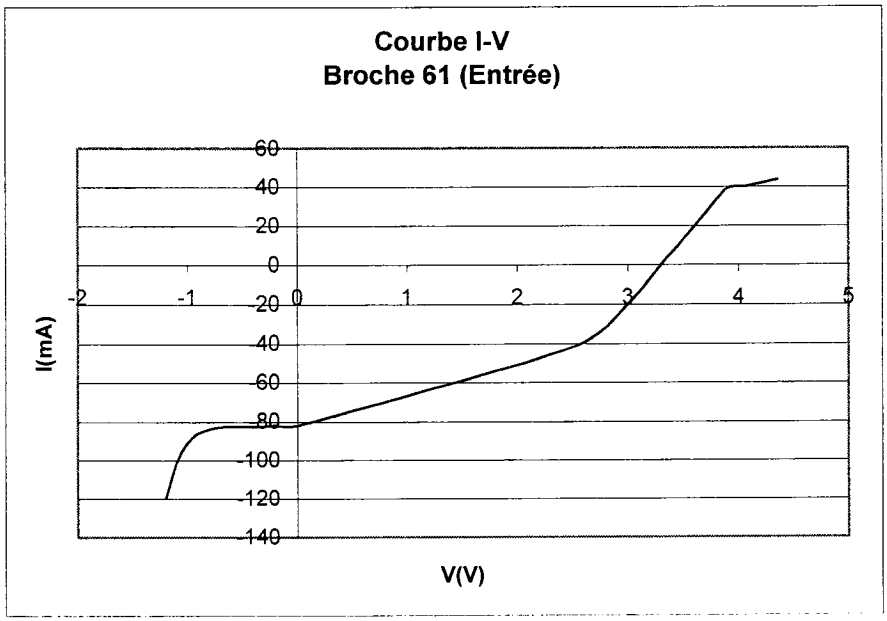

Figure 6.3. Courbe I-V de la broche 61

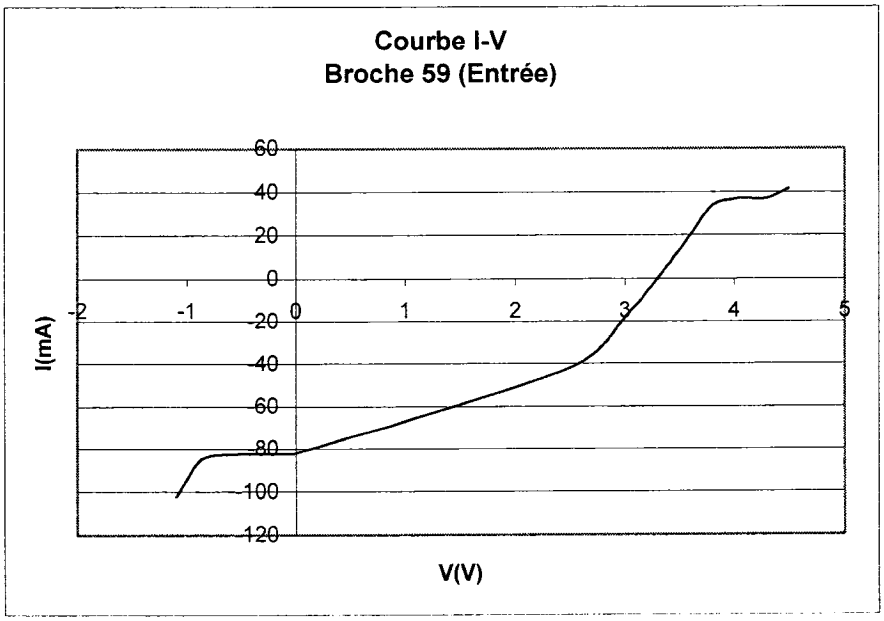

Figure 6.4. Courbe I-V de la broche 59

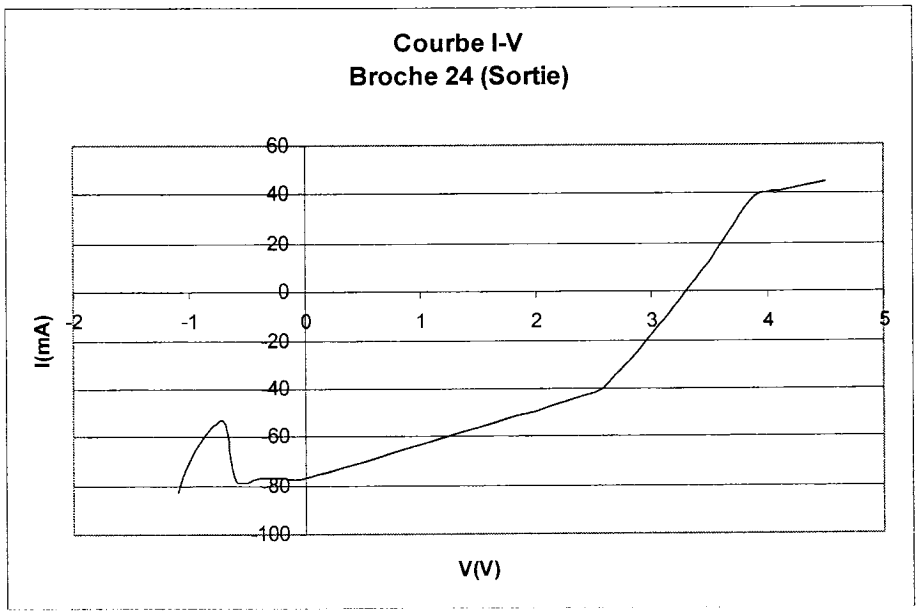

Figure 6.5. Courbe I-V de la broche 24

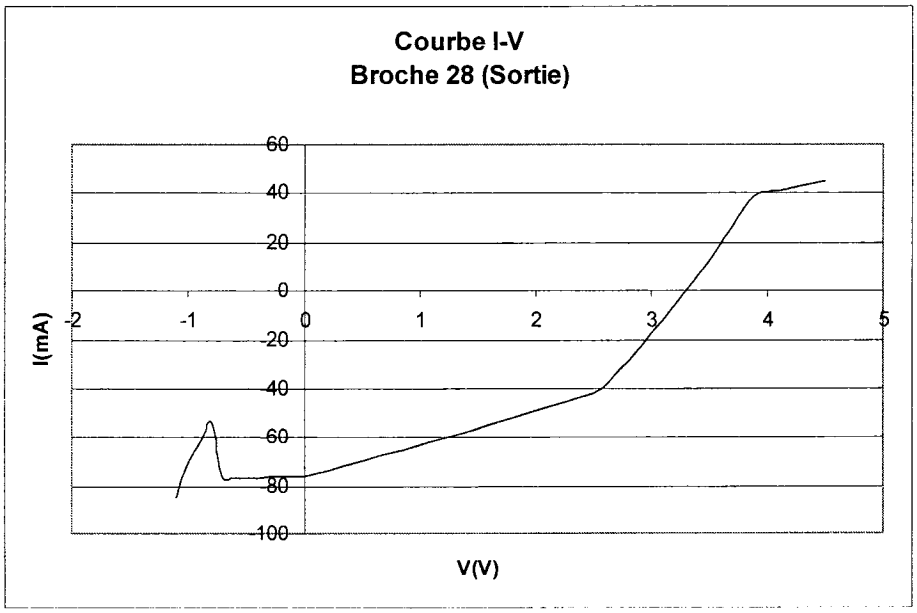

Figure 6.6. Courbe I-V de la broche 28

Nous constatons que la partie gauche (V=–1.0 à 0V) des figures 6.3 et 6.4 ressemble au fonctionnement de la diode de protection. Cependant, le croisement entre 0A et 3,3V observé dans toutes les 4 courbes ne ressemble pas à une réponse normale d'une diode de protection. Cette partie de la courbe indique la présence d'un élément résistif plutôt qu'une diode de protection contre les décharges électrostatiques. Ceci explique pourquoi nous avons observé un 1 collé lors des tests précédents.

Malgré des efforts considérables pour diagnostiquer le problème, nous n'avons pas réussi à trouver les causes du problème.

6.6. Conclusion

Dans ce chapitre, nous avons présenté les étapes menant à la réalisation du prototype. Nous avons aussi présenté la méthodologie et les travaux exécutés pour valider le design en VHDL et la puce de démonstration. Malheureusement, nous avons rencontré certains problèmes avec les entrées et sorties des prototypes. Ces problèmes nous ont empêchés de pousser plus loin les travaux de validation.

Conclusion

Comme il a été mentionné au début de ce mémoire, des gradients de température, des variations de la tension d'alimentation des répéteurs, des variations des paramètres du procédé de fabrication, ainsi que les interférences électromagnétiques font en sorte que les signaux qui traversent une longue distance dans un système sur puce soient soumis à de grand biais de synchronisation. Dans le cadre des recherches sur les systèmes sur puce pour l'intégration à l'échelle de la tranche, la société HyperChip a proposé une nouvelle méthode de synchronisation pour résoudre les problèmes reliés à la transmission sur de longs fils d'interconnexions, c'est-à-dire une méthode de communication qui tolère un plus grand biais de synchronisation. Dans ce mémoire, nous avons donc étudié la nouvelle technique proposée par la société HyperChip pour finalement y apporter des améliorations.

En premier lieu, nous avons présenté le schème de communication envisagé par la société HyperChip. Par la suite, le déphasage entre l'émetteur et le récepteur ainsi que les problèmes d'intégrité dus au biais de synchronisation des données, c'est-à-dire des problèmes connus reliés à la transmission de données sur de grandes distances dans ce type de circuit intégré, ont été étudiés et les causes de ces problèmes ont été identifiées. De plus, nous avons présenté une revue de littérature sur les diverses propositions et solutions existantes ou sur les pratiques courantes visant à résoudre totalement ou partiellement ces problèmes. Les mérites et les faiblesses de ces propositions ont été mis en évidence pour faire ressortir la particularité de la technique utilisée par la société HyperChip. En effet, la méthode de communication suggérée par HyperChip se distingue non seulement par l'absence de protocole de communication entre l'émetteur et le récepteur mais se distingue aussi par le peu de fils d'interconnexion ajoutés.

En deuxième lieu, nous avons modélisé mathématiquement la méthode de synchronisation proposée par la société HyperChip. Des équations décrivant les chemins critiques ont été établies pour évaluer la performance du système. Des techniques d'optimisations du design ont aussi été utilisées pour pousser encore les limites d'opération. Cependant, les analyses mathématiques des chemins critiques ont aussi permis de constater que dans les cas limites, le système étudié ne répond plus aux exigences de l'application. En effet, dépendamment de la phase initiale entre l'horloge locale et distante au départ, la tolérance aux gigues rapides du système pourrait être réduite de façon considérable et dans certains cas, l'intégrité des données sera compromise car le système risque de perdre ou redoubler une donnée sous l'effet d'une gigue rapide. Considérant que ceci n'est pas acceptable pour un système de communication, nous avons cherché à améliorer la proposition initiale.

En troisième lieu, nous avons présenté les principes de fonctionnement d'une des propositions en apparence prometteuse mais non retenue. Cette idée a été aussi suggérée par la société HyperChip comme une variante de l'idée originale. Par ailleurs, c'est cette idée non retenue qui a inspiré la solution finale. Comme l'idée n'est pas retenue, nous n'avons pas détaillé les équations mathématiques décrivant les chemins critiques. Par conséquent, nous avons modélisé cette nouvelle proposition en nous basant seulement sur son principe de fonctionnement. Cependant, les équations mathématiques ainsi développées sont suffisantes pour montrer que, bien que plus tolérant que la proposition initiale, la performance de cette nouvelle solution est encore dépendante de la phase initiale entre l'horloge distante et locale. Les performances recherchées ne sont donc pas garanties.

En quatrième lieu, nous avons présenté la solution qui a été retenue finalement comme remplacement de la proposition originale de la société HyperChip. La

particularité de cette nouvelle solution est qu'elle fait usage de 2 horloges intermédiaires et des synchronisations périodiques sont nécessaires pour enlever l'effet de la gigue lente ou pour enlever la dépendance à la phase initiale entre l'horloge distante et locale. Nous avons donc décrit son principe de fonctionnement et introduit de nouvelles spécifications et restrictions. Les nouveaux chemins critiques ont été identifiés et modélisés mathématiquement. Les analyses mathématiques ont démontré que même dans les cas limites, cette solution finale reste viable et permet d'atteindre les performances recherchées tout en possédant les caractéristiques souhaitées. En effet, l'intégration des données est préservée et la tolérance aux gigues de la solution finale ne dépend plus de la phase initiale de l'horloge locale et distante.

En cinquième lieu, nous avons détaillé l'architecture de la solution finale telle qu'implantée dans la puce de démonstration. Le design en VHDL étant sub-divisé en plusieurs modules hiérarchiques, nous avons donc décrit les interfaces et les circuits logiques utilisés dans chacun des modules. En particulier, nous avons dû modifier l'architecture et adapter l'algorithme de filtrage pour pouvoir rencontrer les contraintes temporelles suite à des résultats préliminaires des essais de synthèse. Les limites d'opération qui découlent ainsi des compromis faits lors de l'implantation de certaines parties ont été énoncées. Nous avons aussi suggéré des possibilités d'amélioration futures.

En dernier lieu, nous avons présenté les étapes qui ont mené à la réalisation finale de la puce de démonstration, fabriquée avec la technologie CMOS 0,18µm de TSMC. Pour pouvoir générer des stimuli pour le récepteur, un module émetteur équipé des générateurs de signaux a été inclus dans la puce de test. Le schéma bloc et une brève description de fonctionnement de ce module émetteur ont été fournis. Nous avons adopté la stratégie d'intégration hiérarchique qui s'est avérée efficace lorsque des erreurs de modélisation dans

la bibliothèque nous ont obligées à modifier les dessins de masques en dernière minute. Nous avons aussi suivi la méthodologie de CMC pour vérifier les problèmes d'interférences électromagnétiques et pour les étapes de DRC et LVS. Les vérifications DRC ont nécessité plusieurs itérations car nous n'avons accès qu'à des modèles dîts boîte noire. Nous avons également présenté l'ensemble des travaux de validation du design en VHDL et de la puce d'évaluation, sans oublier les spécifications de test. De plus, l'algorithme de création de banc d'essai pour le récepteur et la puce a été présenté. Nous avons aussi parlé de la réalisation d'une carte adaptatrice configurable qui permet de tester la puce d'évaluation ayant utilisé un boîtier spécial car les adaptateurs pour ce boîtier existants chez CMC ne répondent pas à nos besoins de signaux et d'alimentation. Enfin, nous avons décrit et discuté les résultats des tests avec les échantillons de la puce de démonstration. Malheureusement, les échantillons n'ont pas réagi à nos tests et plusieurs signaux sont collés à 1. Nous avons donc procédé à d'autres tests de déverminage et de caractérisation. Nous avons ainsi découvert une basse impédance entre l'alimentation 1.8V et VSS dans toutes les puces. De plus, plusieurs plots d'entrées et de sorties ne répondent pas aux tests servant à détecter la présence de leur diodes de protection. Les problèmes ont été rapportés à CMC.

Les problèmes avec les échantillons nous ont empêchés d'aller plus loin. Nous n'avons pas pu démontrer la viabilité de la nouvelle méthode de communication avec la puce d'évaluation.

Cependant, nous espérons que des efforts de recherche sur les méthodes de communication dans les puces continueront à être déployés. De fait, les recherches dans ce domaine ne seront pas seulement bénéfiques aux systèmes sur puce pour l'intégration à l'échelle de la tranche. Les futures recherches

pourront aussi servir aux systèmes intégrés à grande échelle dont le niveau de complexité ne cesse de croître avec la diminution de la taille des transistors.

Références

[1] AFGHAHI, M.; SVENSSON, C., (July 1992). Performance of synchronous and asynchronous schemes for VLSI systems, IEEE Transactions on Computers, Volume: 41, Issue: 7, pp. 858–872

[2] AGUIAR, R.L., SANTOS, D.M., (1998). Wide-area clock distribution using controlled delay lines, IEEE International Conference on Electronics, Circuits and Systems, Volume 2, pp. 63-66

[3] ALPERT, C.J.; THIAGARAJAN, M.; HASSOUN, S., (November 2002). Optimal buffered routing path constructions for single and multiple clock domain systems, 2002 International Conference on Computer-Aided Design (ICCAD '02), pp. 247-253

[4] BAKOGLU, H., (1990). Circuits Interconnections, and Packaging for VLSI. Addison-Wesley, Massachusetts

[5] BAKOGLU, H., WALKER, J., and MEINDL, J., (1986). A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock-skew in ULSI and WSI circuits. Proc. IEEE International Conf. on Computer Design, pp. 118-122

[6] BRUESKE, D.E., EMBABI, S.H.K., (Aug. 1994). A dynamic clock synchronization technique for large systems, IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, Volume: 17 Issue 3, pp. 350-361

[7] CHEN, R.Y., VIJAYKRISHNAN, N., IRWIN, M.J., (1999). Clock power issues in system-on-a-chip designs, VLSI '99. Proceedings. IEEE Computer Society Workshop On, pp. 48 –53

[8] CHOU, N.-C., CHENG, C.-K., (1993). Wire Length and Delay Minimization in General Clock Net Routing, Proc. IEEE Intl. Conf. on Computer-Aided Design, pp. 552-555

[9] CONG, J., (December 1997). Challenges and Opportunities for Design Innovations in Design for Deep Submicron ICs, Src Design Sciences Concept Paper

[10] CONG, J., HE, L., KHOO, K. Y., KOH, C. K., and PAN, Z., (November 1997). Interconnect Design for Deep Submicron ICs, Proc Intl. Conf. on Computer-Aided Design, pp. 478-585

[11] CONG, J., HE, L., KOH, C.-K., and PAN, Z., (November 1997). Global Interconnect Sizing and Spacing with Considerations of coupling Capacitance", Proc Intl. Conf. on Computer-Aided Design, pp. 628-633

[12] DAVIS al and NOWICK STEVEN M., (1995). Asynchronous circuit design: Motivation, background, and methods, Graham Birtwistle and Al Davis, editors, Asynchronous Digital Circuit Design, Workshops in Computing, Springer-Verlag, pp. 1-49

[13] DOBKIN, R.; GINOSAR R.; SOTIRIOU C.P, (April 2004). Data Synchronization Issues in GALS SoCs, 10th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'04), pp. 170-180

[14] DUARTE, D.; NARAYANAN, V.; IRWIN, M.J. (2002). Impact of technology scaling in the clock system power, VLSI on Annual Symposium, IEEE Computer Society ISVLSI 2002, pp. 59–64

[15] EMBABI, S.H.K.; BRUESKE, D.E., (1994). Clock synchronization for WSI systems, IEEE Proceedings of the Sixth Annual IEEE International Conference on Wafer Scale Integration, pp. 228 –234

[16] HOGGE; CHARLES R. Jr., (Dec 1985). A self correcting clock recovery circuit, IEEE Trans. on Electron. Devices, Volume Ed-32, No 12

[17] JACKSON, M.A.B.; SRINIVAN., A. and KUH, E.S., (1990). Clock Routing for high performance ics, Proc. Design Automation Conferences, pp. 573-579

[18] KEES, V.B., (1993). Handshake Circuits: an Asynchronous Architecture for VLSI Programming, International Series on Parallel Computation, Cambridge University Press, Volume 8

[19] KEEZER, D.C.; JAIN, V.K., (1991). Clock distribution strategies for WSI: a critical survey, IEEE Proceedings of the 3rd International Conference on Wafer Scale Integration, pp. 277-283

[20] KEEZER, D.C.; JAIN, V.K., (1992). Design and evaluation of wafer scale clock distribution, IEEE Proceedings of the 4th International Conference on Wafer Scale Integration, pp. 168-175

[21] KESSELS, J., (March 2005). Register Communication between Mutually Asynchronous Domains, 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05), pp. 66-75

[22] KESSELS, J.; KIM, S.J.; PEETERS, A.; WIELAGE, P., (April 2002). Clock Synchronization through Handshake Signalling, Eighth International Symposium on Asynchronus Circuits and Systems (ASYNC'02), pp. 59

[23] KESSELS, J.; PEETERS, A.; WIELAGE, P.; KIM, S.J., (2002). Clock synchronization through handshake signalling, Proceedings of Eighth International Symposium on Asynchronous Circuits and Systems, IEEE pp. 52-61

[24] LEE, S-H., HWANG, M.-S. et al., (Dec 2002). A 5Gb/s 0.25μm CMOS inter-tolerant variable-interval oversampling clock/data recovery circuit, IEEE J.Solid-State Circuits, , Volume 37, no. 12, , pp. 1822-1830

[25] MOORE, S.W.; TAYLOR, G.S.; CUNNINGHAM, P.A.; MULLINS, R.D. and ROBINSON, P., (2000). Self calibrating clocks for globally asynchronous locally synchronous systems, IEEE Proceedings of IEEE international Conference on Computer Design

[26] NIGAM, N.; KEEZER, D.C., (Jan 1993), A comparative study of clock distribution approaches for WSI, Proceedings of Fifth Annual IEEE International Conference on Wafer dsScale Integration, pp. 243-251

[27] PULLELA, S.; MENEZES, N.; OMAR, J.; PILLAGE, L., (1993), Skew and Delay Optimization for Reliable Buffered Clock Trees, Proc. IEEE Intl. Conf. on Computer-aided Design, pp. 556-562

[28] RAZAVI, B., (Aug. 2002). Challenges in the design high-speed clock and data recovery circuits, IEEE Communications Magazine, Volume: 40 Issue: 8, pp 94-101

[29] SATO, H.; ONOZAWA, A.; MATSUDA, H., (1996). A balanced-mesh clock routing technique using circuit partitioning, European Design and Test Conference, 1996. Proceedings, pp. 237-243

[30] SEIZOVIC, J.N., (Nov 1994). Pipeline synchronization, Proc. Internationnal Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 87-96

[31] SHERIDAN, N.G., HABIGER, C. M., LEA, R.M., (Jan 1993). WSI clock & signal distribution. A novel approach, IEEE Proceedings of the 5th Annual IEEE International Conference on Wafer Scale Integration, pp 252-261

[32] SRIDHAR, R., (January 2004). System-on-Chip (SoC): Clocking and Synchronization Issues, 17th International Conference on VLSI Design, pp. 520

[33] WALKER, R., (Feb, 2002). Clock and data recovery for serial digital communication, ISSCC Short course,

[34] WANG, H.; NOTTENBURG, R., (1999). A 1Gb/s CMOS clock and data recovery circuit. ISSCC

[35] YANG, C-K. K.; FARJAD-RAD, R.; HOROWITZ, M. A., (May 1998). A 0.5μm CMOS 4.0 Gbit/s serial link transceiver with data recovery using oversampling, IEEE J.Solid-State Circuits, Volume 3, no. 5, pp. 713-722

[36] YUN, K.Y.; DONOHUE, R. P., (1996). Pausible clocking: a first step toward heterogeneous systems, VLSI in Computers and Processors, 1996. ICCD '96. Proceedings, IEEE International Conference Computer Design, 7-9 Oct 1996. pp. 118 -123

[37] YUN, K.Y.; DOOPLY, A.E., (Dec 1999). Pausible clocking-based heterogeneous systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume: 7 Issue: 4, pp. 482 -488

Annexe I. Script en Perl pour générer le banc d'essai

Ce script en Perl est utilisé pour générer automatiquement les fichiers VHDL qui sont ensuite utilisés comme banc de test pour exercer et valider le design du récepteur.

```
#-----------------------------------------------------------------------------------------
# Ce programme genere les fichiers VHDL qui vont etre utilise comme test bench
# Auteur: Phiet Nguyen
#-----------------------------------------------------------------------------------------

#!/usr/local/bin/perl -w

#-----------------------------------------
# ARGV[0] est ouput testbench vhdl
#-----------------------------------------


$tampon_output="b_top.tmp" ;  # nom du fichier tampon de sortie
open (TMP, ">$toto" ) or die ;


$t=0 ;
$n=500 ;       # nombres de cycles de simulation

$reset = 22 ; # nombre de cycles entre 2 auto-synchronisation
$old_data=0;
```

```
for $i ( 1 .. $n ){
print "======= $i ======== \n ";

# calcul de derive lente
$slow = ($i*10)/$n ;

# calcul de derive rapide

$rval=rand() ;
$inc= ( $rval - 0.5 ) * 3 ;

# combine les 2 type de derives
$t= 4 + 8*$i + $inc + $slow ;

# generation aleatoire de valeurs de donnees
$data= int ($rval * 256) ;

# point d'ancrage ←→ l'horloge ne subit que la gigue lente
$ideal= 4 + 8*$i + 0 + $slow ;

if ( ($i % $reset)==2 )
                { $data= 15 ; }
elsif ( ($i % $reset)==3 )
        { $data = 240 ; }
elsif ( ($i % $reset ) == 4 )
                { $data = 15 }
```

détecte et évite que des valeurs identiques se trouvent sur 2 cycles consécutifs

```
if ($old_data==$data) {
                $data+=1 ;
}
printf TMP ("%02x %.2f %.2f \n", $data, $t, $ideal );
$old_data=$data ;
}

close (TMP);


open (FH, ">$ARGV[0]") or die ;
print "...VHDL generating to $ARGV[0] ...\n" ;
print FH "
library ieee ;
use ieee.std_logic_1164.all ;
use work.all ;

entity b_top is
end b_top ;

architecture bench of b_top is

component receiver
port (
      rclk    : in std_logic ;
      rdata   : in std_logic_vector ( 7 downto 0 ) ;
      lclk    : in std_logic ;
      dataout : out std_logic_vector ( 7 downto 0 )
);
```

```
end component ;

signal  rdata   : std_logic_vector ( 7 downto 0 ) ;
signal  rclk    : std_logic ;
signal  idealrclk : std_logic ;
signal  lclk    : std_logic ;
signal  dataout : std_logic_vector ( 7 downto 0 ) ;

begin -- architecture


TOP: receiver port map (
      rclk    => rclk ,
      rdata   => rdata ,
      lclk    => lclk ,
      dataout => dataout
) ;

--process
--begin
--      rclk <= '1' , '0' after 4 ns ;
--      wait for 8 ns ;
--end process ;

process
begin
      lclk <='0', '1' after 4 ns ;
      wait for 8 ns;
end process ;
```

```
process
begin


";
        print FH "rdata <= \"00000000\" ;\n" ;
        print FH "rclk <='1' ; \n";
        print FH "idealrclk <= '1' ;\n";

open (TMP, $tampon_output) or die ;
while (<TMP>){
        ($data, $time, $ideal)=$_ =~/(\S+)\s+(\S+)\s+(\S+)/ ;
        @dt=split '', $data ;


        if ( $dt[0] eq "0" )
                { $first = "0000" ; }
        elsif ( $dt[0] eq "1" )
                { $first = "0001" ; }
        elsif ( $dt[0] eq "2" )
                { $first = "0010" ; }
        elsif ( $dt[0] eq "3" )
                { $first = "0011" ; }
        elsif ( $dt[0] eq "4" )
                { $first = "0100" ; }
        elsif ( $dt[0] eq "5" )
                { $first = "0101" ; }
        elsif ( $dt[0] eq "6" )
```

```
        { $first = "0110" ; }
elsif ( $dt[0] eq "7" )
        { $first = "0111" ; }
elsif ( $dt[0] eq "8" )
        { $first = "1000" ; }
elsif ( $dt[0] eq "9" )
        { $first = "1001" ; }
elsif ( $dt[0] eq "a" )
        { $first = "1010" ; }
elsif ( $dt[0] eq "b" )
        { $first = "1011" ; }
elsif ( $dt[0] eq "c" )
        { $first = "1100" ; }
elsif ( $dt[0] eq "d" )
        { $first = "1101" ; }
elsif ( $dt[0] eq "e" )
        { $first = "1110" ; }
elsif ( $dt[0] eq "f" )
        { $first = "1111" ; }


if ( $dt[1] eq "0" )
        { $second = "0000" ; }
elsif ( $dt[1] eq "1" )
        { $second = "0001" ; }
elsif ( $dt[1] eq "2" )
        { $second = "0010" ; }
elsif ( $dt[1] eq "3" )
        { $second = "0011" ; }
```

```
elsif ( $dt[1] eq "4" )
      { $second = "0100" ; }
elsif ( $dt[1] eq "5" )
      { $second = "0101" ; }
elsif ( $dt[1] eq "6" )
      { $second = "0110" ; }
elsif ( $dt[1] eq "7" )
      { $second = "0111" ; }
elsif ( $dt[1] eq "8" )
      { $second = "1000" ; }
elsif ( $dt[1] eq "9" )
      { $second = "1001" ; }
elsif ( $dt[1] eq "a" )
      { $second = "1010" ; }
elsif ( $dt[1] eq "b" )
      { $second = "1011" ; }
elsif ( $dt[1] eq "c" )
      { $second = "1100" ; }
elsif ( $dt[1] eq "d" )
      { $second = "1101" ; }
elsif ( $dt[1] eq "e" )
      { $second = "1110" ; }
elsif ( $dt[1] eq "f" )
      { $second = "1111" ; }

print FH "------------------------- $data ------------------------------------\n" ;
$t_tmp=$time - 3 ;
print FH "rdata <= transport \"${first}${second}\" after $t_tmp ns ;\n" ;
print FH "rclk  <= transport '1' after $t_tmp ns , '0' after $time ns ; \n";
```

```
        $ideal_tmp= $ideal - 4 ;
        print FH "idealrclk  <= transport '1' after $ideal_tmp ns , '0' after $ideal ns ;
\n";

}
close (TMP);

print FH "
wait ;
end process ;
end bench ;
";

print "\nqvhcom $ARGV[0]\n" ;
# system ("qvhcom $ARGV[0]");
```

Annexe II. Carte adaptatrice de IMS

Une carte adaptatrice a été conçu pour adapter le boîtier CFP-80 (80 broches) au testeur IMS (64 canaux).

Les figures A.1 et A.2 montrent les 2 côtés de la carte adaptatrice.

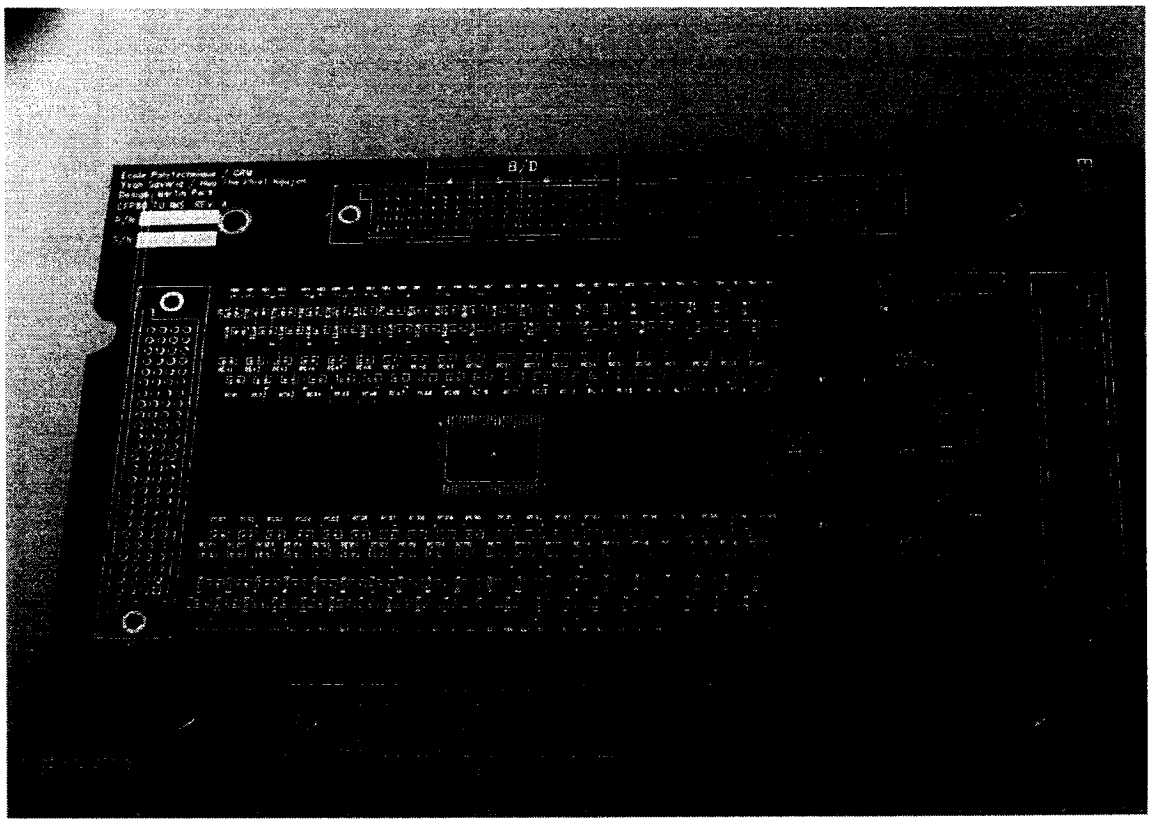

Figure A.1. Photographie du dessus de la carte adaptatrice

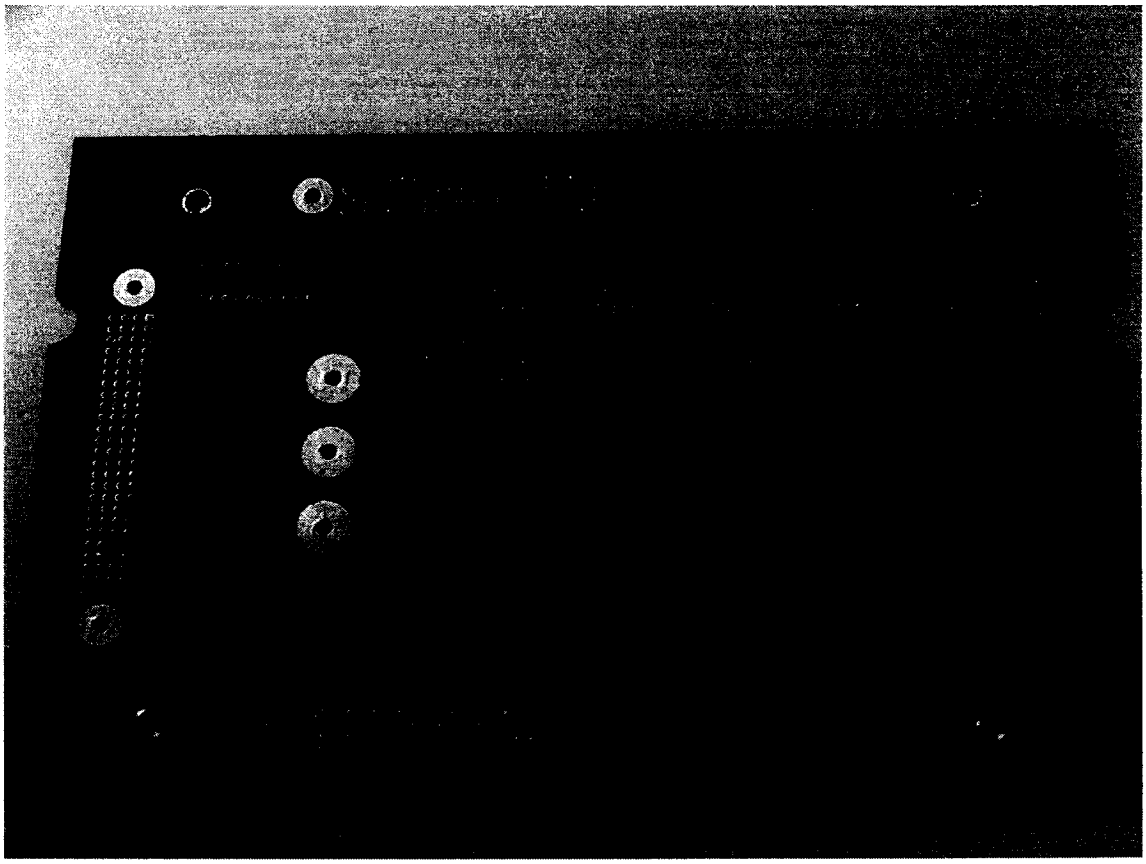

Figure A.2. Photographie du dessous de la carte adaptatrice

Sur la carte adaptatrice, chacune des 80 broches du boîtier CFP-80 peut être configurée selon les besoins avec des résistances ou des ponts de soudures en

- VDD1;
- VDD2;
- VSS;
- Pull up à VDD1 ou à VDD2;
- Pull down;
- Diviseur de tension entre VDD1 / VSS ou VDD2 / VSS ou VDD1 / VDD2.

Pour faciliter l'assignation de broches, il est fortement recommandé de répartir les signaux et surtout les alimentations uniformément tout autour du boîtier.

Nous suggérons d'avoir au moins une alimentation (VDD1 ou VDD2 ou VSS) pour chaque groupe de 4 signaux utiles, ou en d'autres termes, nous déconseillons d'avoir plus de 8 signaux utiles consécutifs.

Annexe III. Assignation de broches

Le tableau A.1. montre l'assignation de broches de la puce de démonstration ainsi que les canaux du testeur IMS correspondants utilisés pour valider la puce.

Dans ce tableau, les signaux nommés VDD33 réfèrent à l'alimentation 3,3V. L'alimentation 1.8V est indiquée par VDD.

Tableau A.1. Assignation de broche de puce

Nom du signal	*# Pin*	*Canal de IMS*
VDD33	1	
DataLClkIO[0]	2	4A0
DataLClkIO[1]	3	5A7
VSS	4	
DataLClkIO[2]	5	5A6
DataLClkIO[3]	6	5A4
VDD	7	
DataLClkIO[4]	8	5A3
DataLClkIO[5]	9	5A2
VDD33	10	
Vcontrol	11	5A1
VDD	12	
RingOutIO	13	6A7
VSS	14	
RefClkIO	15	6A6
VDD	16	
PLL_Vctl	17	6A5
PLL_RefClk	18	6A3

Nom du signal	*# Pin*	*Canal de IMS*
VDD	41	
RClkInIO	42	7B2
DataDataIO[7]	43	6B5
VSS	44	
DataDataIO[6]	45	6B4
VDD33	46	
DataDataIO[5]	47	6B6
VSS	48	
DataDataIO[4]	49	6B1
VDD	50	
DataDataIO[3]	51	6B3
VSS	52	
DataDataIO[2]	53	5B5
VDD	54	
DataDataIO[1]	55	5B4
VSS	56	
DataDataIO[0]	57	5B7
VDD33	58	

Nom du signal	*# Pin*	*Canal de IMS*
VSS	19	
PLL_pllout	20	6A2
VSS	21	
VDD	22	
VDD	23	
TstDataIO[7]	24	7A7
VSS	25	
TstDataIO[6]	26	7A4
VDD33	27	
TstDataIO[5]	28	7A3
VDD	29	
TstDataIO[4]	30	7A2
VSS	31	
TstDataIO[3]	32	7A0
VDD	33	
TstDataIO[2]	34	7B5
VDD33	35	
TstDataIO[1]	36	7B7
VSS	37	
TstDataIO[0]	38	7B1
bypassIO	39	7B0
LClkInIO	40	7B3

Nom du signal	*# Pin*	*Canal de IMS*
LClkOutIO	59	5B1
VSS	60	
RClkOutIO	61	5B3
VDD	62	
VSS	63	
DataClkIO[0]	64	4B7
DataClkIO[1]	65	4B6
VDD	66	
DataClkIO[2]	67	4B4
DataClkIO[3]	68	4B3
VDD33	69	
DataClkIO[4]	70	4B2
DataClkIO[5]	71	4B1
VSS	72	
DataRClkIO[0]	73	4A5
DataRClkIO[1]	74	4A4
VDD	75	
DataRClkIO[2]	76	4A6
DataRClkIO[3]	77	4A7
VSS	78	
DataRClkIO[4]	79	4A3
DataRClkIO[5]	80	4A2