Université Sidi Mohammed Ben Abdellah
Faculté Des Sciences et Techniques Fès
Département de Génie Electrique

PROJET DE FIN D'ETUDE

Pour l'obtention du diplôme
Master en Sciences et Techniques
Spécialité
« Systèmes Microélectroniques, de Télécommunications et de l'Informatique Industrielle »

Préparé à ST-ERICSSON RABAT
Présenté et soutenue publiquement
Par
Abdelilah KADI
Le
30 juin 2011

Méthodologie pour la vérification du Clock Domain Crossing

Jury

Mlle. Farah ARRON	Encadrant (ST-ERICSSON)
Mr. Ali AHAITOUF	Encadrant (FST-FES)
Mme. Fatima ERRAHIMI	Encadrant (FST-FES)
Mr. El Hossain ABARKAN	Examinateur (FST-FES)
Mr. Abdelaziz AHAITOUF	Examinateur (FST-FES)

Avant-propos

Nom et prénom de l'étudiant stagiaire de la FST de FES:

Abdelilah KADI.

Intitulé du travail:

Méthodologie pour la vérification du Clock Domain Crossing.

Etablissement d'accueil:

ST-Ericsson, Division DSE IP, Avenue Mohamed Jazouli, Madinat Al Irfane 10000 RABAT, Maroc.

Nom et prénom de l'encadrant dans l'établissement d'accueil ST-Ericsson :

Mlle. Farah ARRON.

Mme. Rachida BAH.

Nom et prénom de l'encadrant à la Faculté des sciences et Techniques de Fès :

Pr. Ali Ahaitouf.

Pr. Fatima Errahimi.

Nom et prénom du responsable du Master SMTII à FST de FES:

Pr. El Hossain ABARKAN.

Date de début et de fin du stage :

Du 26 janvier au 20 juin 2011.

Soutien financier :

Stage rémunéré.

Dédicaces

A mes très chers parents,

Nulle dédicace ne saurait exprimer la profondeur de mes sentiments d'amour et de reconnaissance que j'éprouve à vos égards. Ma réussite est le fruit de vos efforts.

Vous avez su me donner des conseils, me montrer le chemin qui doit me conduire à ma liberté. Vos bénédictions et vos soutiens multiples ne m'ont jamais fait défaut. Ce jour est le vôtre.

A mes très cher frères et sœurs

Votre amour et soutien continu ont eu le plus grand effet sur mon parcours, je vous

souhaite tout le bonheur du monde.

A tous les membres de ma famille,

Pour leurs aides et leurs encouragements.

A mes amis,

Pour tous les bons moments qu'on a partagés ensemble.

Je dédie ce modeste travail

Abdelilah KADI

Remerciements

Au terme de ce modeste travail, je tiens tout d'abord à présenter mon énorme gratitude envers l'entreprise d'accueil ST-ERICSSON de m'avoir offerte l'opportunité d'effectuer mon projet de fin d'étude au sein de son organisme et d'avoir mis à ma disposition toutes les ressources possibles afin d'aboutir à mes objectifs.

Je présente ma sincère reconnaissance et mes remerciements les plus vifs à mes deux encadrantes Mlle Farah ARRON et Mme Rachida BAH, pour leurs Assistance, leurs conseils et leurs orientations judicieuses.

Je remercie Mr. Yassir LAAZIRI manager de l'équipe *IP Design* pour la confiance qu'il m'a accordé de travailler sur un sujet d'actualité plein d'innovation et de m'avoir permis d'acquérir de nouvelles connaissances et une expérience professionnelle enrichissante, je tiens à lui exprimer ma sincère reconnaissance.

Je tiens à remercier également les professeurs de la faculté des sciences et techniques de FES pour les précieuses connaissances que j'ai acquis sous leurs bienveillances. Et plus particulièrement à mes deux encadrant Mr. Ali AHAITOUF et Mme. Fatima ERRAHIMI pour avoir accepté de m'encadrer, et pour leurs efforts fournis tout au long de ma formation ainsi que pour leurs conseils.

Mes remerciements vont également aux membres du jury, pour avoir accepté de me faire profiter de leurs compétences pour évaluer ce travail.

Je suis particulièrement redevable envers tous les ingénieurs de l'équipe *IPDesign* pour l'intérêt qu'ils m'ont manifesté envers moi, en mettant à ma disposition tout ce dont j'avais besoin pour réussir mon travail.

Par crainte d'omettre toute personne ayant contribué au parachèvement de ce projet, je témoigne par le présent travail ma vive reconnaissance à tous ceux qui m'ont aidé de prés ou de loin.

Sommaire

Introduction générale

Un des problèmes majeurs rencontrés dans les systèmes sur puce (SOC), est que les différents blocs fonctionnent sur des horloges indépendantes. L'intégration de ces blocs se fait via la topologie des bus et d'autres interfaces de communication, cela peut être gênant à cause des comportements imprévisibles qui peuvent se produire lorsque ces interfaces asynchrones ne sont pas correctement synchronisées.

La vérification et la correction de la synchronisation manuellement dans le design, fait retarder le flot de conception, et souvent de nombreux problème dans le passage d'un domaine d'horloge à un autre (CDC : Clock Domain Crossing) sont négligées, ce qui entraine un nombre important de problèmes CDC dans la conception.

Un problème CDC détecté au niveau du silicium nécessite de revenir au stade du code RTL (Register Transfer Level) corriger le problème et refaire tout le flot de conception, ce qui alourdi et ralenti le process.

Il y a une façon de rendre ce processus moins lourd. Elle exige une discipline claire pour la synchronisation et une méthodologie d'analyse des CDC qui prenne en compte les modifications techniques dans la conception. Aussi, l'outil de vérification CDC doit prendre en charge les méthodes qui permettent de masquer les faux problèmes, et que ces faux problèmes seront pris en compte par l'outil a chaque vérification.

Le sujet de Mon projet de fin d'étude a pour premier objectif de définir une méthodologie pour la vérification du Clock Domain Crossing (CDC) fiable et efficace qui assure que le circuit est correctement conçu et qu'il est robuste devant tous les problèmes CDC.

Une deuxième partie de mon projet est l'automatisation de la vérification du Clock Domain Crossing, cette automatisation est transparente pour le designer et a pour objectif de faciliter et optimiser la vérification.

Chapitre I

Contexte du Stage

J'ai effectué mon stage de fin d'étude à ST ERICSSON au site de Rabat précisément, je commencerai ce chapitre par une présentation générale de l'entreprise d'accueil, puis on passera à l'équipe du design IP où j'ai effectué mon stage de fin d'étude, ensuite je le finirai en développant le cahier des charges en présentant l'outil utilisé pour la vérification du CDC.

I.1. Présentation de l'organisme d'accueil

Crée en 2009, ST-Ericsson est une co-entreprise à 50/50 réunissant la division semi-conducteurs pour des applications destinées aux téléphones cellulaires de STMicroelectronics (ST-NXP Wireless) et celle des plateformes mobiles d'Ericsson (Ericsson Mobile Platform) Installée en Suisse, son siège social se trouve à Genève. A la fin 2009, la société employait environ 7 700 personnes dans le monde, dont plus de 85 % sont dédiées à la Recherche et au développement (R&D).

ST-Ericsson est une entreprise de semi-conducteurs sans usine et assure la production de ses plaquettes via une combinaison des unités de production Front end de STMicroelectronics et de fonderies externes.

Les activités de ST-Ericsson sont étendues à travers le monde entier, avec des centres principaux en Europe (essentiellement en France et en Suède), et également solidement implantée dans le reste de l'Asie.

I.2. Le centre de design Rabat:

Le centre de design de ST-Ericsson à Rabat est formé de plusieurs équipes dont chacune à une mission particulière mais en complémentarité avec les autres groupes, comme le montre l'organigramme ci-dessous (Figure I.1) :

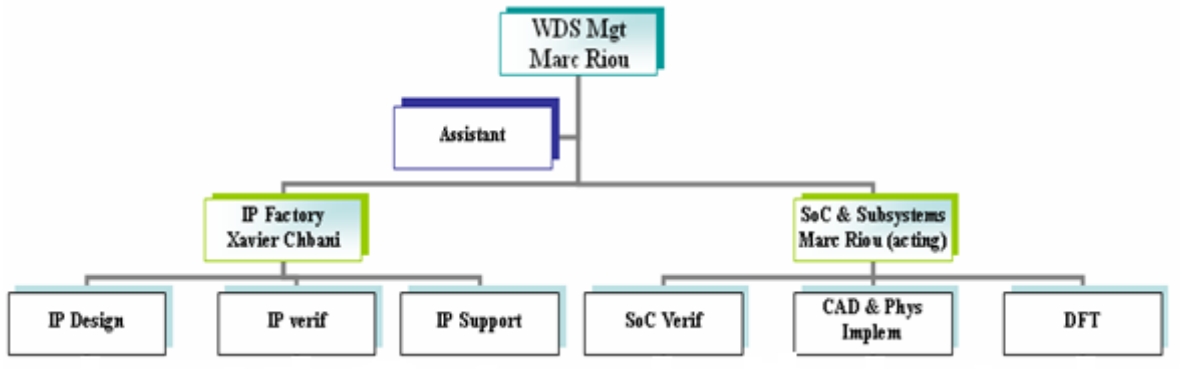


Figure I.1 : Organisation de ST-Ericsson Rabat

I.3. Equipe IP Design:

Le rôle de l'équipe Design IP est de traduire les spécifications du cahier des charges en un langage de haut niveau HDL (Hardware Description Language), afin d'obtenir un code RTL (Register Transfer Level). Ce code doit être validé, puis converti en un fichier appelé netlist.

I.4 Vérification fonctionnelle :

Le concepteur vérifie si le design qu'il a mis en place ne contient pas d'erreurs et qu'il répond aux fonctionnalités exigées par le cahier des charges. Pour ce faire, il compile, élabore puis simule le design par un « testbench » en appliquant des stimuli sur les entrées du circuit et visualiser les résultats sur les sorties. Les étapes de cette vérification sont :

-Créer les deux fichiers de configuration (cds.lib et hdl.var) :

- **cds.lib** : Définit les librairies Verilog et VHDL et le chemin des fichiers binaires.
- **hdl.var** : Définit quelques variables utilisées pour la configuration de NCSim.

-Compiler les fichiers :

Cette étape consiste à vérifier le code RTL et corriger les erreurs de syntaxe, on utilise l'outil de compilation NCVHDL de CADENCE, si le code ne contient aucune erreur de syntaxe alors on passe à la phase d'élaboration.

-Élaborer le design :

L'élaboration du code HDL a pour but de lier les différents modules et les interconnecter entre eux. On obtient finalement un seul bloc défini par ses entrées et sorties ainsi qu'un fichier « SNAPSHOT », résultat de la compilation et de l'élaboration.

-Simuler le design :

Lors de la simulation, l'outil de simulation reçoit le résultat de l'élaboration (« SNAPSHOT ») et génère les formes d'ondes, on peut alors vérifier le fonctionnement du circuit en suivant dans le temps les sorties en fonction des entrées. Ces étapes sont illustrées sur la figure suivante (Figure I.2).

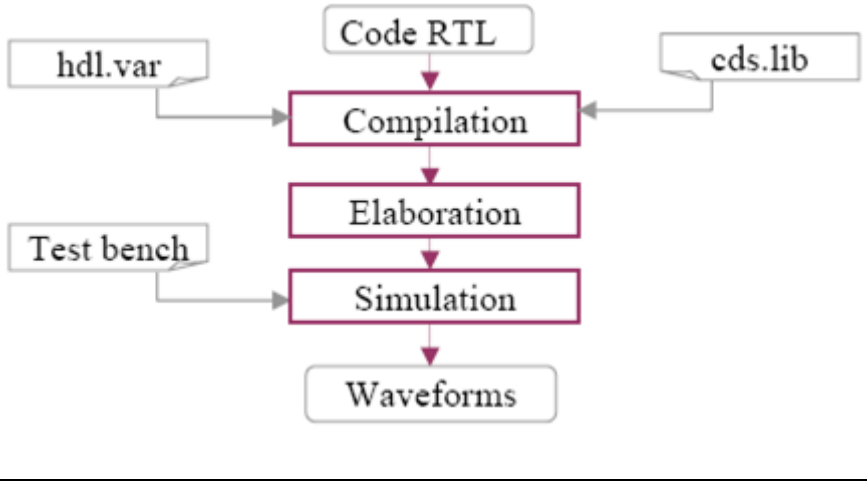


Figure I.2 : Vérification fonctionnelle

I.5 Vérification du RTL :

Cette étape consiste à évaluer la qualité du code RTL syntaxiquement et logiquement. En effet, elle permet de s'assurer si le code a été écrit conformément aux règles de la société ST-Ericsson. Il permet aussi de prévoir d'éventuels problèmes lors de la phase de l'implémentation physique, tel que le Clock Domain Crossing, la synthétisabilité du code, sa testabilité, ...

ST-ERICSSON utilise l'outil Spyglass pour valider la vérification du RTL, cet outil est un vérificateur du code, il permet la saisie la vérification, l'optimisation et l'exploration de la conception au début du flux de conception au stade RTL (Register Transfer Language),

quand on peut corriger les problèmes et explorer les alternatives avec plus de rapidité et de facilité.

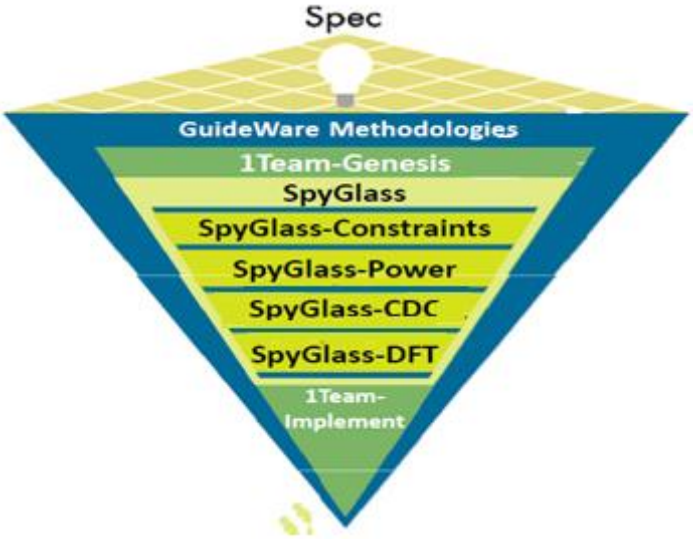


Figure I.3 : Vérification du RTL par l'outil spyglass

I.6 Vérification du Clock Domain Crossing

C'est à ce stade ou se situe mon projet de fin d'étude le but de cette vérification est d'assurer qu'aucun problème CDC n'existe lors du passage de la donnée entre deux domaines d'horloge asynchrones.

La solution SpyGlass-CDC d'Atrenta analyse les conceptions pour assurer l'exactitude des mécanismes de synchronisation utilisée dans l'IP (intellectual property). Les bogues au niveau des synchronisations de domaines d'horloge défectueuses entre les blocs IP sont difficiles à identifier avec des outils de vérification fonctionnelle.

I.7 Spécification et cahier des charges

Mon projet de fin d'étude consiste à définir une méthodologie pour la vérification du CDC, l'élaboration d'une nouvelle méthodologie nécessite une bonne maitrise des différents enjeux lié au Clock Domain Crossing, le projet a été organisé comme suit (Tableau I.1: spécification et cahier des charges) :

Documentation	
Etape	***Description***
1	Etudes des différents problèmes CDC
2	Explorer toutes les contraintes CDC définies par atrenta
3	Lire et comprendre quelques fichiers de contraintes qui existent dans la base de donné
Test	
Etape	***Description***
1	Comprendre le principe de fonctionnement de l'outil Spyglass.
2	Appliquer les contraintes définie sur les IPs (PCM_I2S ;I2CHS)
3	Comparaison des résultats (avant et après l'application de la méthodologie).
Automatisation:	
Etape	***Description***
1	Création d'un tableur permettant à l'utilisateur de saisir toutes les informations nécessaires pour la vérification du CDC d'une IP.
2	Création d'un script permettant à l'utilisateur de générer le fichier SGDC a partir de ce tableur
3	Test et validation du script sur les IP (Pcm_i2s, I2CHS)

Tableau I.1 Spécification et cahier des charges

Chapitre II

Généralité sur la synchronisation

II.1. Méthodes de synchronisation

Il existe plusieurs méthodes pour synchroniser un système :

– L'approche traditionnelle est celle du système **synchrone**, qui consiste à utiliser une horloge globale sur tout le système avec un décalage de phase pouvant être négligé.

– La seconde approche est la réalisation d'un système totalement **asynchrone**. Les communications se font par des mécanismes spéciaux. La différence entre la fréquence et la phase est aléatoire, Le système asynchrone a pour avantage de réduire la consommation dynamique.

– Une autre approche consiste en la distribution d'horloges **mésochrones**. Une horloge commune est distribuée sur tout le système. Les différentes parties du système communiquent d'une manière synchrone, mais avec une différence de phase constante. Les communications entre ces parties nécessitent une adaptation pour s'accommoder du décalage de phase.

Classe du circuit	$\Delta\phi$	Δf	Synchronisation
Synchrone	0	0	Non nécessaire
Mésochrone	ϕ_c	0	Compensation de phase
Asynchrone			Synchroniseurs à base de double flip-flop ou autre

Tableau II.1 : classes des circuits [2]

II.2. Domaine d'horloge

Des horloges synchrones appartiennent aux mêmes domaines d'horloges, alors que deux horloges asynchrones appartiennent à des domaines d'horloges différents.

Si une horloge dérivée d'une autre horloge par un diviseur de fréquence dans ce cas les deux horloges ont une phase constante, elles sont synchrones et appartiennent à un même domaine de d'horloge (figure II.1).

Deux horloges qui sont générés par la même PLL de fréquence 50Mhz et 33Mhz, ont une phase qui change avec le temps donc, les deux horloges appartiennent à deux domaines différents (figure II.2).

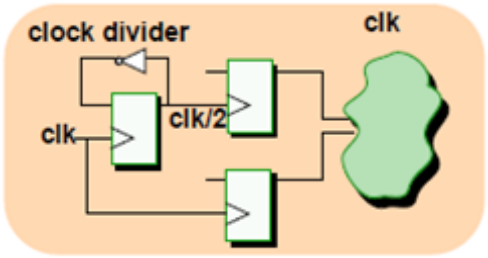


Figure II.1 : single domaine d'horloge [1]

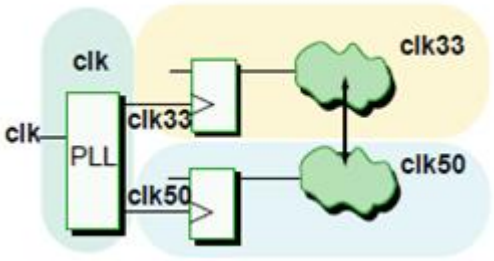


Figure II.2: Multiples Domaine d'horloge [1]

Si les entrés primaires à un circuit incluent de multiple horloges (source différentes), alors ces horloges asynchrones déterminent des domaines d'horloge distincts (figure II.3).

De même si les entrées d'un circuit sont asynchrones à ce circuit lui-même, alors ces entrées seront asynchrones appartenant à un domaine d'horloge différent (figure II.4).

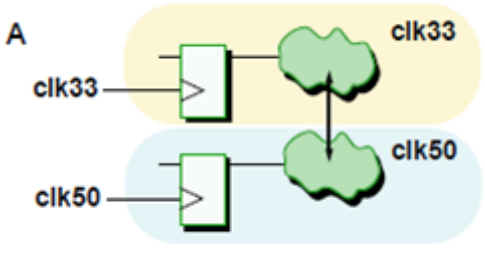


Figure II.3: Horloges asynchrones [1]

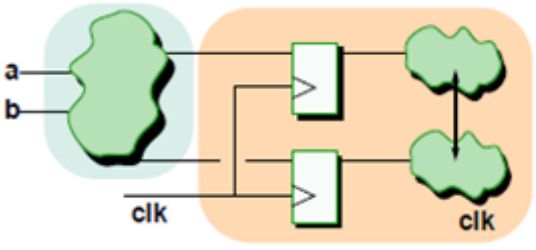


Figure II.4: Entrés asynchrones [1]

Toutes les horloges qui appartiennent à un même domaine d'horloge constituent un **groupe d'horloge**. Par conséquent les groupes d'horloge identifient les domaines d'horloges.

II.3 Reset synchrone et asynchrone

Reset synchrone signifie que le reset ne sera pris en compte que lors d'un front d'horloge. Dans le cas d'un reset synchrone, seule l'horloge est utile dans la liste de sensibilité, car il n y a qu'un front d'horloge qui entraînera une action.

Le Reset asynchrone est pris en compte a n'importe quel moment dans ce cas là, le signal de reset doit impérativement se trouver dans la liste de sensibilité.

```
Process(clk)
begin
If clk'event and clk='1' then
        If rst = '1' then
Count <= "0000" ;
        Elsif (count >= 9) then
        Count <= "0000" ;
        Else
        Count <= count + 1 ;
        End if ;
End if ;
End process ;
```

Figure II.5 : Reset synchrone

```
Process(clk, rst)
Begin
If rst = '1' then
Count <= "0000" ;
Elsif (clk'event and clk='1')
then
        if (count >= 9) then
        Count <= "0000" ;
        Else
        Count <= count + 1 ;
        End if ;
End if ;
End process ;
```

Figure II.6 : Reset asynchrone

Description VHDL d'un Reset synchrone (Figure II.5) et asynchrone (Figure II.6).

II.4 Réseau sur puce [3]:

Les communications dans les systèmes sur puce (SoC : System-on-Chip) sont réalisées a partir d'une topologie de bus, toutes les unités de traitement devant communiquer entre elles sont reliées au même medium de communication (« bus » ou « bus hiérarchique »), et un élément central (« arbitre de bus ») contrôle l'accès au medium pour les unités afin d'éviter les conflits (Figure II.7).

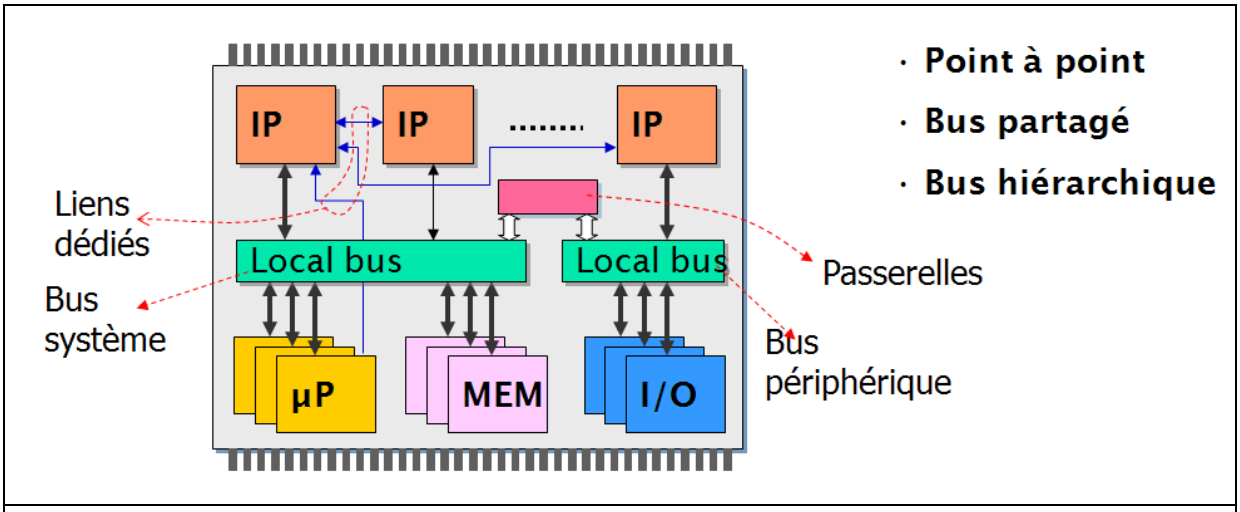


Figure II.7 : Communication par bus

Ces architectures présentent de nombreuses limitations : limitation du débit des communications (bande passante du bus), difficulté d'adaptation des débits des échanges aux besoins des applications, latences importantes, impossibilité de bien gérer le parallélisme dans le traitement des données compte tenu des contraintes d'acheminement des informations et des éléments de contrôle.

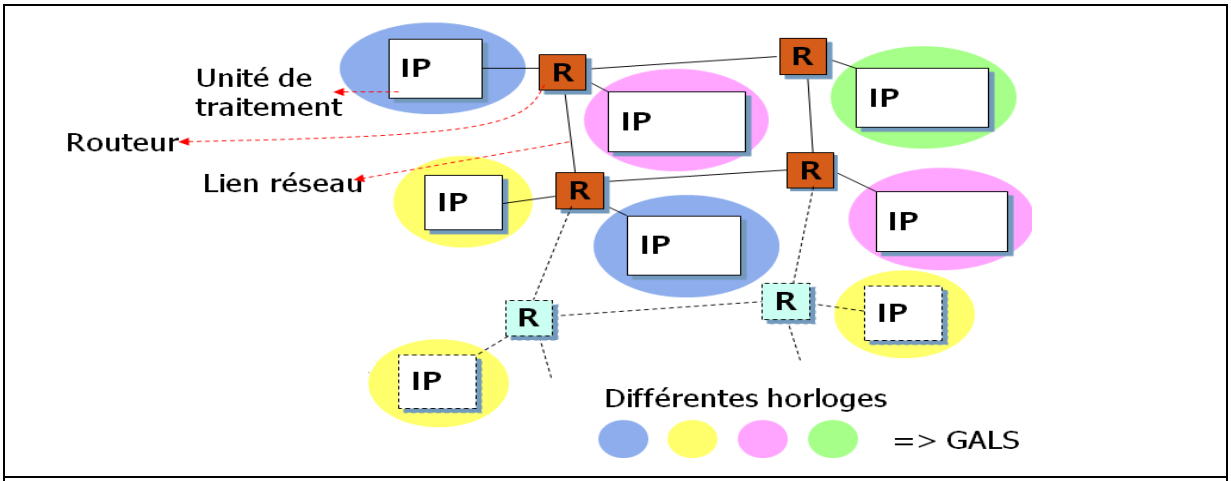


Figure II.8 : Réseau sur puce

La complexification croissante des applications (multimédia, télécoms…), et les besoins accrus de performances conduisent les concepteurs à réaliser des SoC implémentant un nombre toujours croissant d'unités de traitement. La quantité de données échangées entre ces unités est également en constante augmentation. Les architectures de communication sur puce deviennent donc un élément important qu'il est déterminant de maîtriser lors de la conception

d'un SoC complexe. Ceci a conduit les concepteurs à proposer de nouvelles architectures de type « réseau sur puce » (en anglais « Network-on-Chip (NoC) ») (Figure II.8). Cette architecture consiste en une adaptation des principes des réseaux informatiques pour réaliser des structures de communication flexibles et distribuées. Cette approche présente de nombreux avantages qui font pressentir les NoC comme les successeurs des bus : augmentation des débits d‘échanges et des performances globales des circuits numériques en mettant a propos les capacités d'interconnexion liées aux nouvelles générations technologiques, réduction des temps de latence entre traitements, extensibilité et flexibilité accrue de l'architecture du fait de sa régularité.

Chapitre III

Les Problèmes du Clock Domain Crossing

Ce chapitre décrit les différents problèmes qu'on trouve lors du passage d'un domaine d'horloge à un domaine d'horloge différent commençant par le problème basique la métastabilité, puis la perte de donnée ensuite l'incohérence de données, et le problème de la logique combinatoire.

III.1. Clock Domain Crossing

Le passage d'un domaine d'horloge à un autre se produit chaque fois que les données sont transférées à partir d'une flip-flop entraînée par une horloge à une flip-flop commandé par une autre horloge.

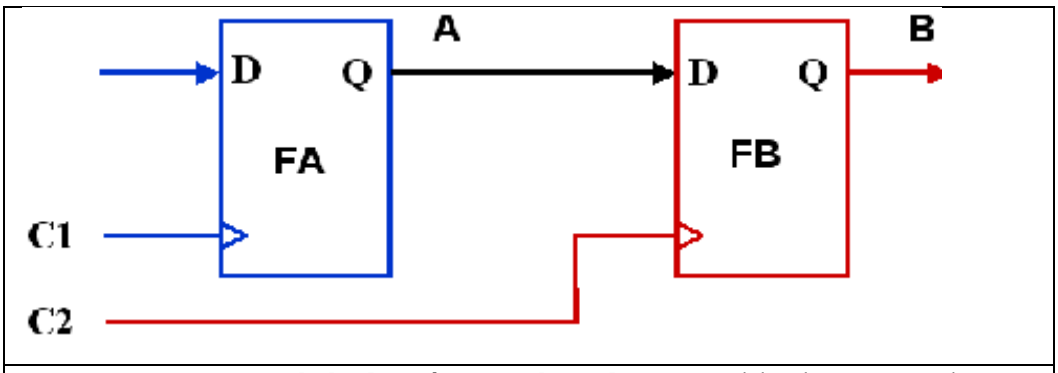


Figure III.1: passage de la donnés entre deux domaines d'horloges asynchrones

Dans la (Figure III.1), le signal A vient du domaine d'horloge C1 et doit être capturé correctement par le domaine d'horloge C2. En fonction de la relation entre les deux domaines d'horloges, il pourrait y avoir différents types de problèmes de transfert des données de la source d'horloge à l'horloge de destination.

III.2. La métastabilité

La métastabilité caractérise un État de non-équilibre de période longue, théoriquement illimitée, En électronique, ce phénomène peut apparaître sur les flip-flops, car ils sont conçus pour avoir deux états logiques (1 et 0). Cependant, entre ces deux états stables, il est possible d'identifier un état métastable (Figure III.2).

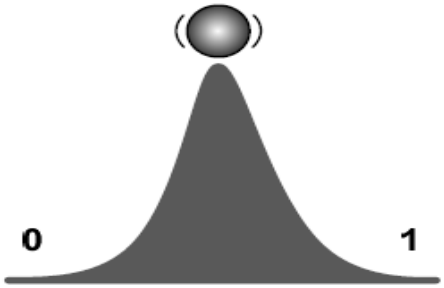


Figure III.2 : Etat Métastable

Il y a différent paramètres de la flip-flop qui entre dans le phénomène de métastabilité.

$\mathbf{T_{SU}}$: Temps d'établissement, c'est le délai avant le front d'horloge à partir duquel le signal d'entrée doit être stable (setup time) (Figure III.3).

$\mathbf{T_H}$: Temps de maintien, c'est le délai après le front d'horloge qui doit être garanti avant que la valeur du signal d'entrée puisse varier à nouveau (Figure III.3).

$\mathbf{T_P}$: est le temps de propagation dans la cellule, ou temps de réponse, avant d'obtenir l'établissement d'un signal de sortie stable (*setting time)* (Figure III.3).

Une flip-flop peut entrer dans un état métastable quand un signal est échantillonné à une valeur intermédiaire qui ne permet pas de déterminer son niveau logique.

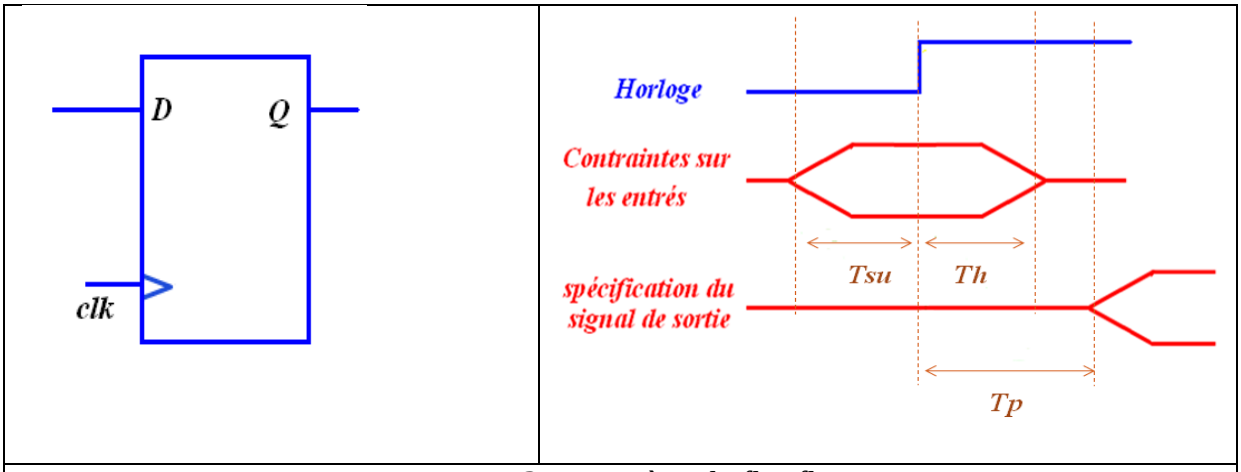


Figure III.3 : Paramètre du flip-flop

III.2.1 Erreurs de synchronisation due au non respect des contraintes temporelles :

Le phénomène de métastabilité va provenir de l'utilisation de la bascule D comme synchroniseur pour échantillonner sur front d'horloge un signal asynchrone qui ne respecte pas les contraintes temporelles. La métastabilité apparaît chaque fois qu'un bloc synchrone échantillonne au moyen de son signal d'horloge un signal asynchrone. Si la transition de ce signal asynchrone se produit trop près de la transition de l'horloge, on aboutit aux erreurs suivantes :

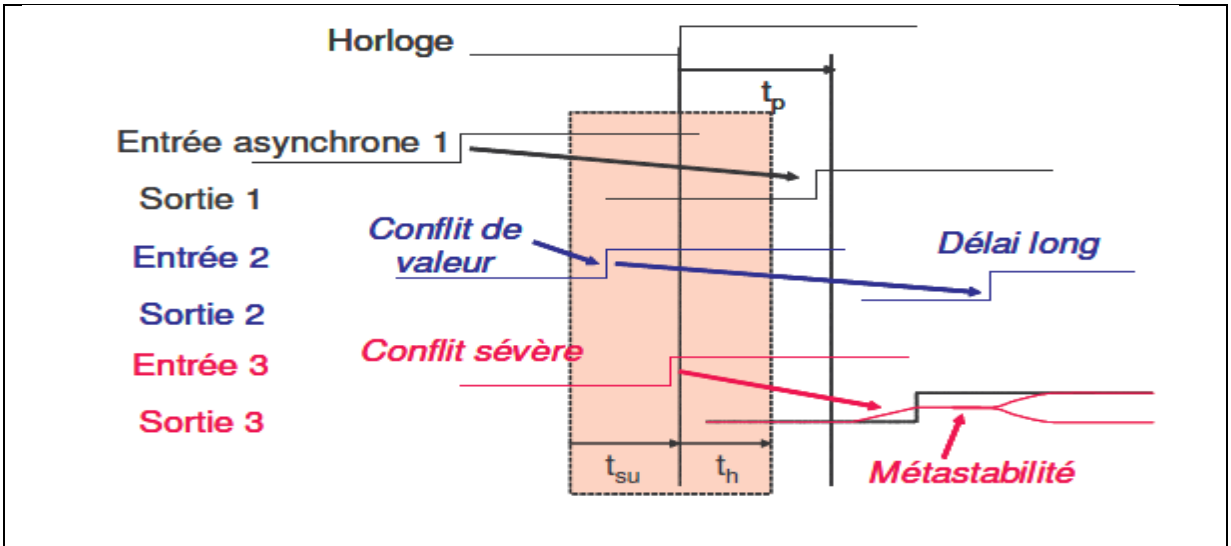


Figure III.4: Erreur de synchronisation due au non respect des contraintes temporelles [2]

Le conflit de valeur sur l'entrée 2 entraîne un retard à l'établissement de la sortie 2, ce qui peut entraîner la violation des contraintes temporelles sur une éventuelle bascule suivante.

Cependant la valeur fournie en sortie sera exacte en regard de l'entrée. Le conflit sévère sur la valeur 3 entraîne la métastabilité du circuit : le temps de stabilisation et la valeur de sortie sera arbitraires.

III.2.2 Effet de métastabilité :

La métastabilité peut avoir plusieurs conséquences à partir d'un point de vue design:

- Si des données instables envoyées à plusieurs endroits dans la conception, elles peuvent conduire à un haut débit, et même la détérioration de la puce dans le pire des cas.
- Il se peut que des signaux prennent des valeurs aléatoires, et cela peut entraîner le design dans un état inconnu, conduisant à des problèmes fonctionnels dans le design.

III.3. Perte de la donnée :

Tant que chaque transition sur le signal source est captée dans le domaine de destination, les données ne sont pas perdues.
Pour éviter les pertes de données, les données de base devraient rester stables pendant un certain temps minimum, de sorte que le temps d'établissement et de maintien soient respectés en ce qui concerne le front actif de l'horloge de destination.

Exemple1:

Supposant que la source d'horloge Ck1 est 2 fois plus rapide que Ck2 et il n y a pas de différence de phase entre les deux horloges, supposant que la séquence de donnés d'entrée générée sur le front l'horloge Ck1 est « 00110011 » et que la séquence de donné de sortie capturée sur le front d'horloge Ck2 doit « 0101 », tant que tout les transitions du signal sont capturées la donnée n'est pas perdue.

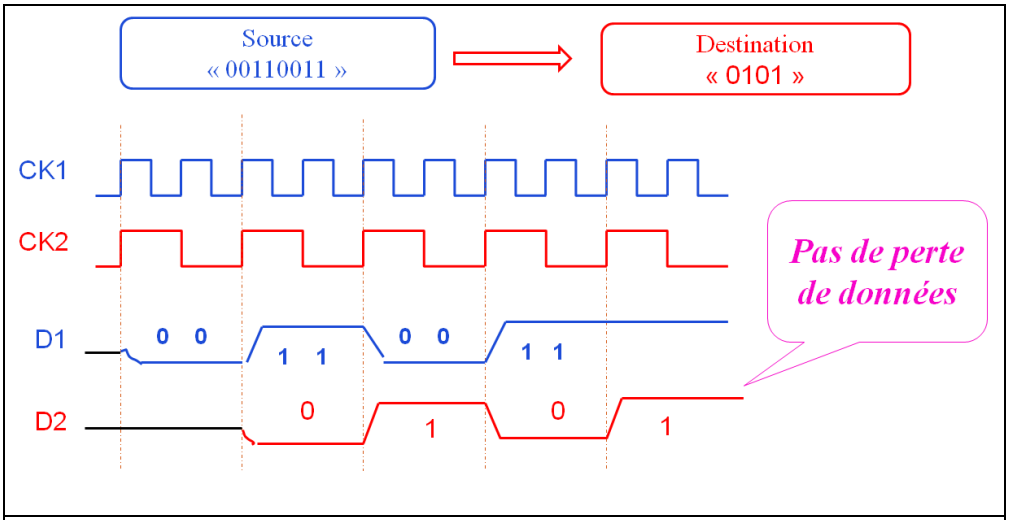


Figure III.5: pas de perte de données

Exemple 2:

Cependant, si la séquence de donnée d'entré est « 00100011 » et que la sortie doit être « 0101 », dans l'exemple ci-dessous le troisième bit n'est pas capturé pour la raison qu'il ne reste pas suffisamment stable, et la donné échantillonnée est « 0001», dans ce cas on a une perte de données.

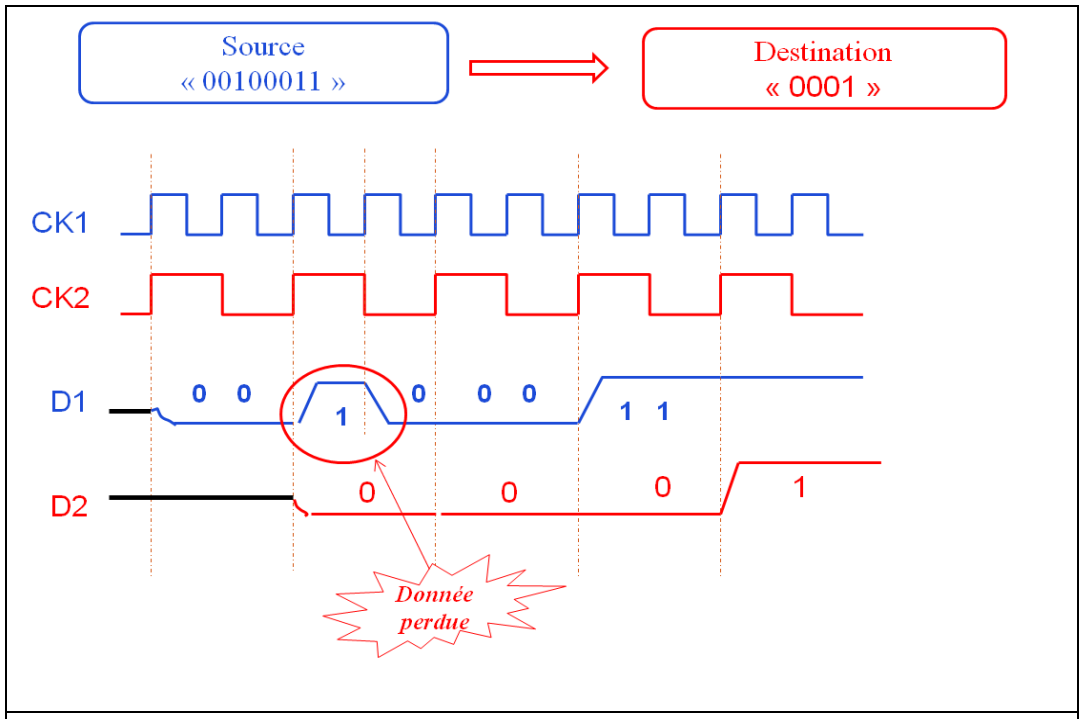


Figure III.6: perte d'une donnée

III.4. Donnée capturé dans le second front d'horloge de destination :

Chaque fois qu'une nouvelle source de données est générée, elle peut ne pas être capturée par le domaine de destination dans le premier cycle d'horloge de destination en raison de la métastabilité.

Si les fronts actifs d'horloge de C1 et C2 arrivent rapprochés, on risque de ne pas percevoir la transition de signal A (contrôlé par l'horloge C1) qui vient d'avoir lieu. Cette transition du signal A ne sera capturé qu'au second front d'horloge de destination (C2) (Figure III.7).

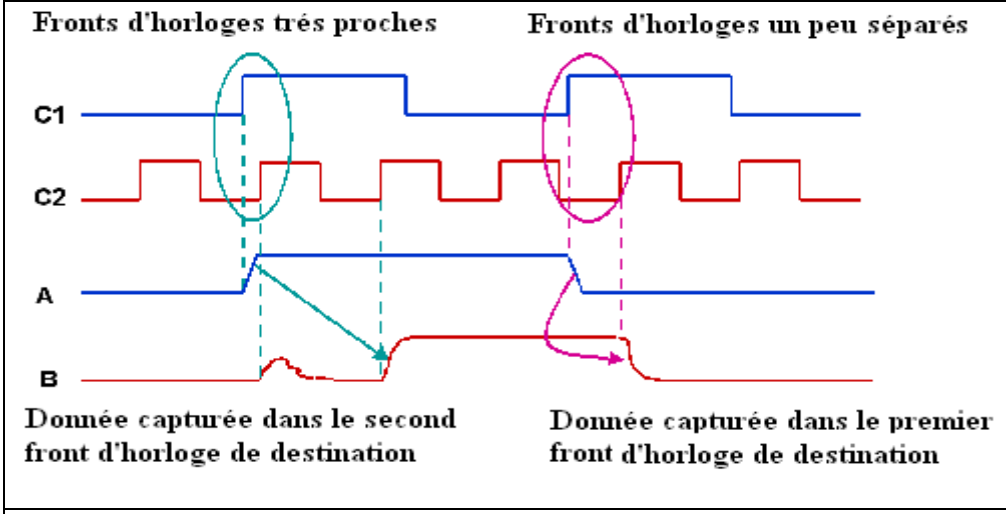


Figure III.7 : Donnée capturé dans le second front d'horloge de destination

Par conséquent, il se peut qu'il n'y a pas de correspondance de cycle entre la source et les données du domaine de destination.

III.4.1. Incohérence de la donnée:

Chaque fois que de nouvelles données sont générées dans le domaine d'horloge source, elles peuvent prendre un ou plusieurs cycles d'horloge de destination pour qu'elles soient capturées, en fonction du moment d'arrivée des fronts d'horloge active de destination.

Prenons le cas où de multiples signaux sont transférés d'un domaine d'horloge à un autre et chaque signal est synchronisé séparément à l'aide d'un synchroniseur multi-flop. Si tous les signaux évoluent en même temps et les fronts d'horloges source et destination arrivent rapprochées,

Certains des signaux peuvent être capturés dans le domaine de destination dans le premier cycle d'horloge tandis que d'autres peuvent être capturés dans le second cycle d'horloge, a cause de la métastabilité.

Il peut en résulter une combinaison non valide de valeurs sur les signaux sur le côté de destination. La cohérence des données est perdue dans un tel cas.

Si ces signaux contrôlent certaines fonctions de la conception, alors cet état non valide peut conduire à des erreurs fonctionnelles.

Exemple:

Supposant qu'on a un passage d'un domaine d'horloge de Ck1 source de fréquence plus rapide a un domaine d'horloge Ck2 de destination, les données source « 00 » et « 11 » sont générées par le signal source X[0 :1] synchronisé par l'horloge Ck1.

Initialement on a une transition de 1->0 pour les deux bits de X. les deux transitions sont capturées dans le domaine de destination dans le premier cycle d'horloge de destination, et la sortie Y[0 :1] de destination reçoit « 00 ».

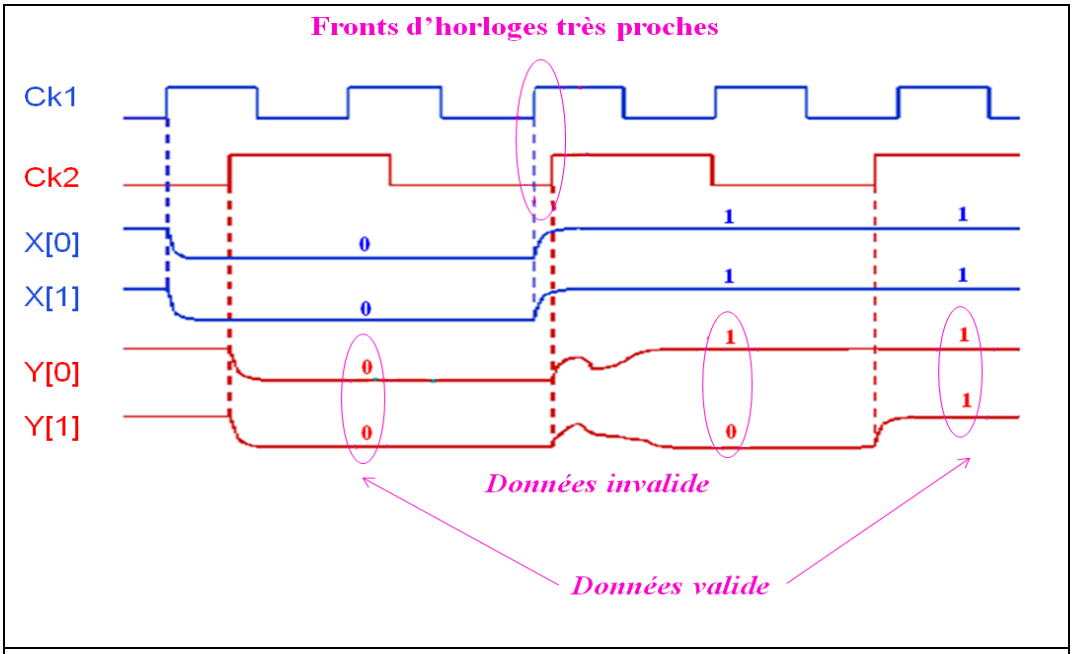


Figure III.8 : Incohérence des données

Ensuite, il y a une transition de 0 -> 1 à la fois sur les bits du signal X. Les fronts montant de la source d'horloge Ck1 et de destination Ck2 se rapproche de la transition sur le signal X. La transition sur X [0] est capturé dans le premier cycle d'horloge, la transition sur X [1] est capturée en second cycle d'horloge de Ck2. Il en résulte une valeur de «10» sur Y [0:1] qui est un état non valide. La cohérence des données est perdue dans ce cas.

III.5. Problème de la logique combinatoire :

Dans les circuits intégrés la logique combinatoire (AND, XOR …) peut générer des petites pulses indésirable qui peuvent produire des erreurs fonctionnelles dans le circuit (Glitch).

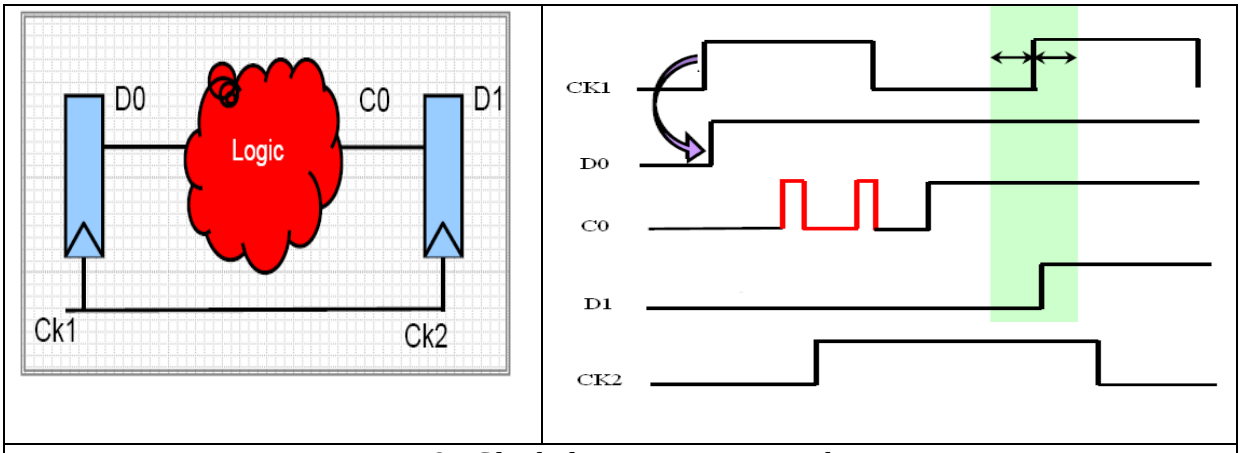


Figure III.9 : Glitch due a une sortie combinatoire

Dans la (Figure III.9) ci-dessus le signal D0 est généré par l'horloge Ck1 va passer dans une logique combinatoire qui génère le signal de sortie C0, la logique combinatoire transite des Glitch avant que le signal C0 se stabilise sur la valeur 1.

Si la deuxième flip-flop est commandée par l'horloge CK1, on échantillonne un signal D1 stable et correcte, alors si la deuxième flip-flop est contrôlée par une horloge asynchrone CK2, il se peut que le signal de sortie D1 prend une valeur incorrecte.

III.5.1 Logique combinatoire entre deux domaines d'horloge différents :

Dans la (Figure III.10) en dessous le signal D0 est contrôlé par l'horloge Ck1 et passe dans une logique combinatoire générant le signal C0, C0 est un signal qui doit avoir la valeur 0, a cause de la logique combinatoire ce signal comporte un Glitch, supposant que ce Glitch est capturée dans le domaine de destination, le signal D1 prend une valeur incorrecte.

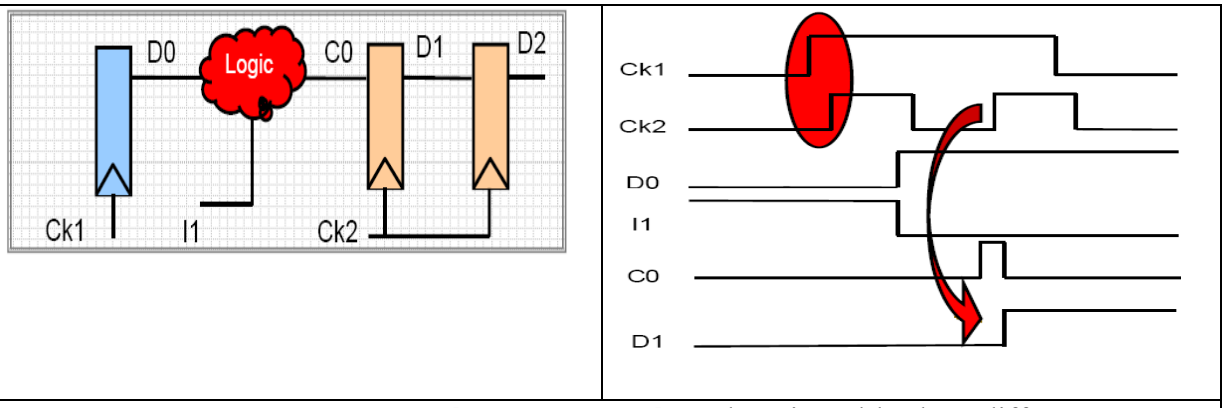


Figure III.10: Logique combinatoire entre deux domaines d'horloge différents

Pour éviter ce genre de problèmes un signal qui passe d'un domaine d'horloge source à un autre domaine de destination doit sortir d'une Flip-Flop contrôlée par l'horloge source (Figure III.11).

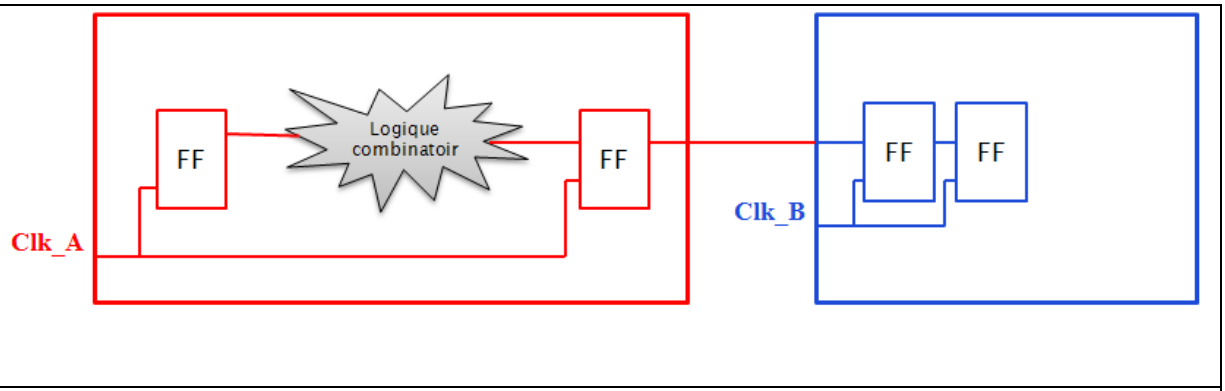


Figure III.11 : solution au problème de la logique combinatoire

Chapitre IV

Les mécanismes de synchronisations

A coté de la diversité des problèmes de passage de la donnée d'un domaine d'horloge à un autre, il existe différentes solutions à ces problèmes à savoir le (Multi-flop, le Multiplexeur de recirculation, la Fifo asynchrone, le handshake, le code Gray…) dans ce chapitre on définit ces différents mécanismes de synchronisation et leurs principes de fonctionnement.

IV.1. Le synchroniseur :

IV.1.1 Fonctionnalité :

La communication entre deux systèmes asynchrones, caractérisées par un mécanisme de synchronisation. L'un de ces mécanismes permettant d'échantillonner un signal asynchrone est le synchroniseur, La fonction du synchroniseur est de résoudre l'un de ces deux problèmes équivalents :

1. étant donné une transition sur le signal asynchrone à échantillonner et une transition sur le signal échantillonneur **(horloge)**, déterminé qui s'est produit en premier.

Ou

2. étant donné une valeur de tension sur le signal asynchrone à échantillonner, déterminer à un instant donné si cette valeur est au-dessus ou en dessous des valeurs de seuil respectivement d'état logique haut ou d'état logique bas du signal.

La complexité d'un tel circuit va dépendre à la fois du degré de fiabilité exigé et du niveau de performances attendu. Le paragraphe suivant présente le synchroniseur traditionnellement utilisé. C'est le circuit le plus simple et basique assurant une excellente fiabilité mais au prix d'une latence élevée.

IV.2 Le Multi-flop:

Le synchroniseur est généralement constitué de deux ou plusieurs bascules cadencées par l'horloge du domaine de destination. C'est un schéma de base pour la synchronisation qui est utilisé dans les différents mécanismes de synchronisation.

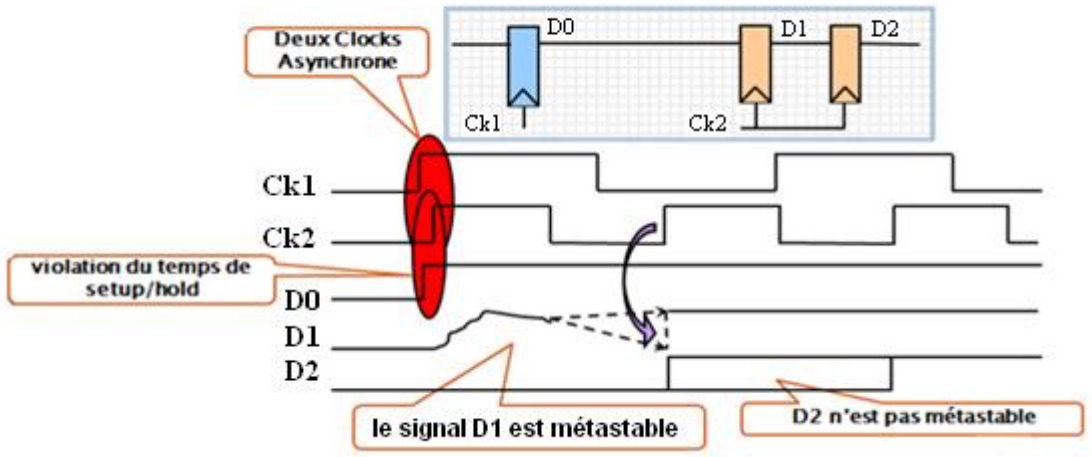


Figure IV.1 : le Multi-flop

Dans la (Figure IV.1 : le synchroniseur 2FF) le signal D0 passe de CK1 vers le domaine CK2, la sortie D1 entre dans un état métastable à cause d'une violation du temps d'établissement de CK2, pour isolé ce problème de métastabilité une deuxième flip-flop est inséré, qui a permit que le temps de stabilisation (*settling time*) autorisé pour le signal de sortie devient alors une entière période d'horloge.

En général le nombre de flip flop de synchronisation est **2**, ce nombre est calculé à partir de la formule suivante :

$$MTBF = \frac{e^{(T1/tau)}}{2 \times Fclk \times fdata \times T0}$$

MTBF : Main Time Before failure

Tau: temps de résolution (mesuré expérimentalement)

2*fdata: fréquence de la donnée d'entrée.

Fclk: fréquence de l'horloge d'entrée.

T0: fenêtre de métastabilité.

T1: temps de résolution T1=N×Tclock - (N-1)×(Tpd+Tsu)

Tclock: période de l'horloge=1/fclock

Tpd: délais de propagation clk to Q

Tsu: délais d'établissement

IV.3. Multiplexeur de recirculation :

Le signal de control (EN: Enable), généré dans le domaine source est synchronisé dans le domaine de destination en utilisant le synchroniseur multi-flop, le signal de control EN_sync conduit les entrés de sélection des Multiplexeurs, contrôlant ainsi le transfert des données pour tous les bits du bus de A. De cette façon, les bits du bus de données ne sont pas synchronisés séparément, et pas de problème d'incohérence des données.

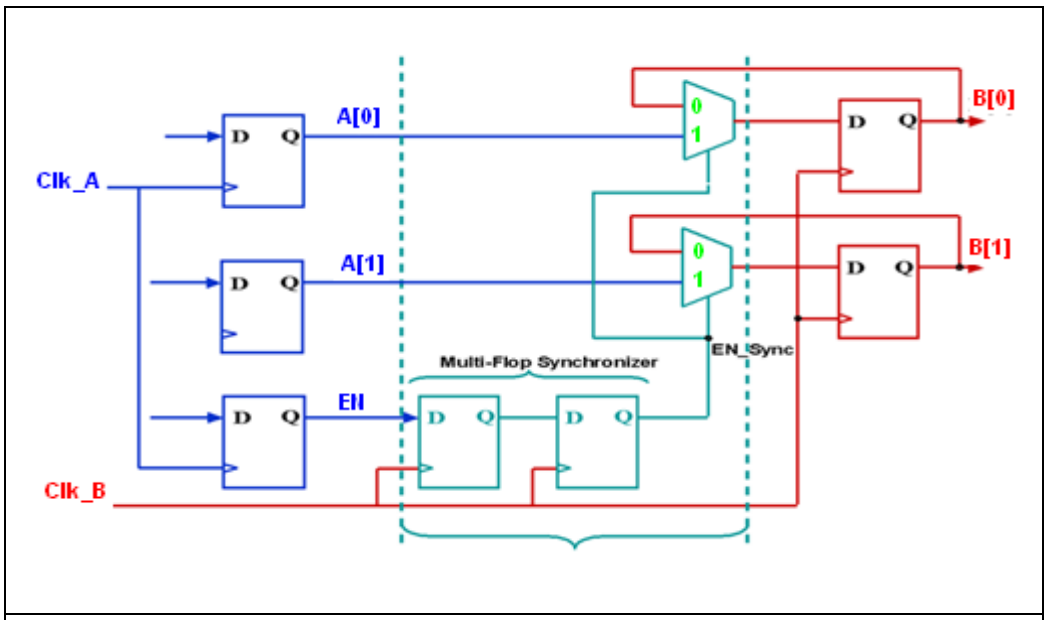


Figure IV.2 : Mux de recirculation

IV.4 Le code Gray

Le code de Gray, également appelé binaire réfléchi, est un type de codage binaire permettant de ne modifier qu'un seul bit à la fois quand un nombre est augmenté d'une unité.

	Binary count	Gray count
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Figure IV.3 : Code Gray

Cet avantage du code Gray permet de synchroniser un bus de données entre deux domaines d'horloges par le synchroniseur Multi-flop sans problème d'incohérence de données.

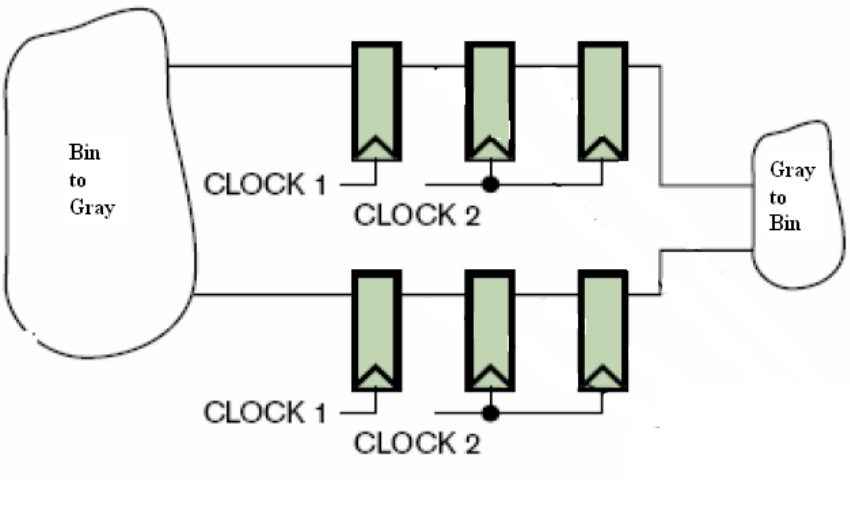


Figure IV.4 : synchronisation par le code Gray

IV.5 Synchroniseur des ports Resets :

Si le circuit est à reset synchrone, et que le signal reset à l'entré du circuit est asynchrone par rapport à l'horloge, alors ce signal reset à l'entré ne respecte pas les contraintes temporelles du flip flop (temps d'établissement, temps de maintien,…), à n'importe quel moment, il peut entrainer le circuit dans un état métastable.

Pour éviter ces problèmes on utilise le synchroniseur reset qui se compose en générale de 2 flip-flop en cascade, le nombre de flip flop dépend des paramètres de la flip-flop et de la fréquence utilisée.

La figure en dessous représente le schéma d'un synchroniseur des ports Reset :

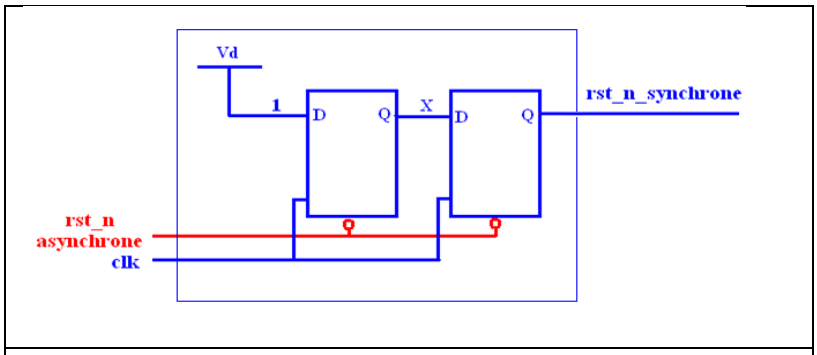


Figure IV.5 : Synchroniseur Reset

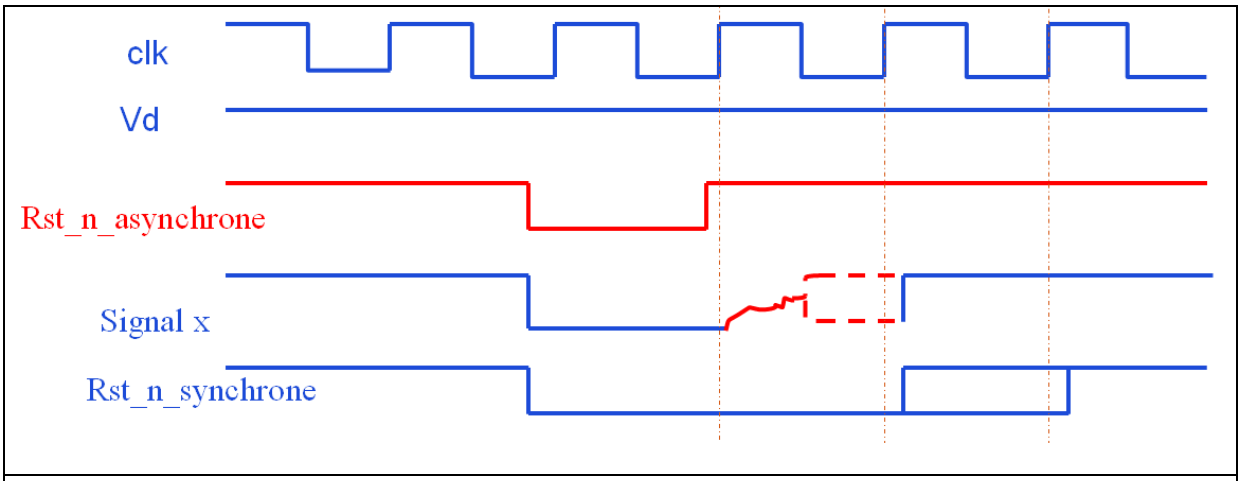


Figure IV.6 : Chronogramme du synchroniseur Reset

La chronogramme en dessus (Figure IV.6) explique le principe de fonctionnement du synchroniseur Reset, supposant que reset_n change l'état au moment du front montant d'horloge ce qui entraine le premier flip flop dans un état métastable (signal x), pour isoler cette métastabilité et rendre le signal en sortie stable, une deuxième flip-flop est ajoutée, on obtient un signal stable et résolut (Rst_n_synchrone).

IV.6 Le handshake

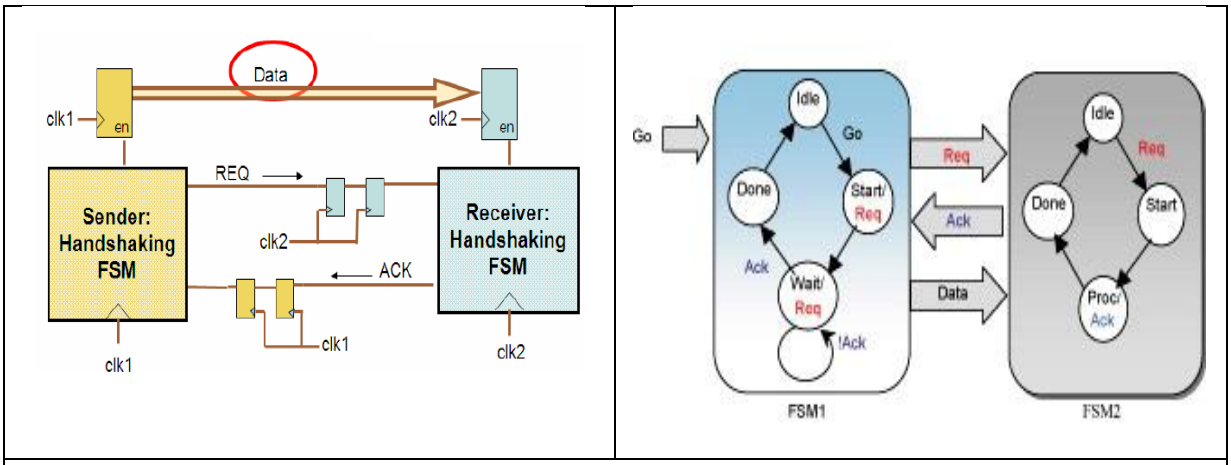


Figure IV.7 : Le synchroniseur Handshake

La méthode la plus courante et robuste pour la synchronisation de plusieurs signaux de données est la technique du handshake comme le montre la (Figure IV.7). Cette technique est la plus populaire vue qu'elle peut facilement gérer les changements des fréquences d'horloge.

La logique du handshake est beaucoup plus complexe que les structures de synchronisation standard, il se compose de deux machine à états (Sender, Receiver) qui contrôlent le passage de la donnée a l'aide du signal enable des (flip-flop source et destination) et deux signaux (Request) et (Acknowledge) synchronisés par le synchroniseur 2 flip-flop.

IV.7 Fifo asynchrone

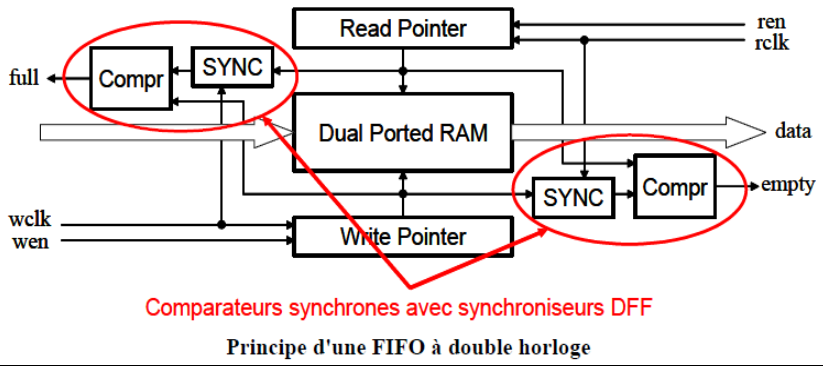


Figure IV.8 : Fifo asynchrone

La FIFO Asynchrone est un bloc responsable de la conversion (en fréquence) des signaux à partir de l'horloge source vers l'horloge de destination. Elle comporte deux parties: une partie

d’écriture et une partie de lecture. L'écriture est commandée par l’'horloge source "wclk" or la lecture est commandée par l’horloge de destination "rclk".

Les données sont écrites dans la FIFO à chaque «wen», a condition que la FIFO n'est pas pleine (n_full est affirmé). Quand un mot est écrit, le pointeur d'écriture est incrémenté.

A chaque demande de lecture « ren », tant que le signal empty est à 0 le mot indexé par le pointeur de lecture est mis dans la data de sortie a chaque front montant d’horloge ensuite le pointeur de lecture est incrémenté,

Si toutes les données écrites sur le FIFO sont lu, le signal empty est à 1. Ou si le nombre de données écrites sur le FIFO atteint la profondeur et aucune donnée n'est lue, le signal full est à 1.

IV.8 Flip flop magic

Au lieu d’utiliser deux flip-flop pour la synchronisation on peut utiliser une seule flip-flop spécifique nommé « Flip-Flop magic », cette flip-flop magic a des caractéristiques spécifiques qui permettent de résoudre le problème de Mètastabilité comme on peut le faire avec le synchroniseur normale 2 flip-flop, le problème majeur de la flip-flop magique est qu’elle occupe beaucoup de surface dans la conception.

Chapitre V :

Méthodologie pour la vérification du Clock Domaine Crossing

Ce chapitre est divisé en trois parties, la première partie définit la méthodologie que je propose pour la vérification du Clock domain Crossing, la deuxième partie comporte une étude des mécanismes de synchronisation, et la troisième partie décrit l'automatisation de la vérification du Clock Domain Crossing .

V.1 Problématique :

Les systèmes Complexes intègrent différents applications audio vidéo, Wireless ce qui implique plusieurs domaines d'horloges asynchrones.

Les horloges qui sont asynchrone les un par rapport aux autres peuvent atteindre des flops différents à des moments légèrement différents à chaque cycle pendant le fonctionnement normal du design. Cette incertitude en temps peut provoquer des problèmes CDC.

Les problèmes CDC ne peuvent pas être complètement identifiés à l'aide des méthodes de vérification traditionnelle, comme la simulation fonctionnelle et l'analyse temporelle statique (STA).

Un problème CDC détecté au niveau du silicium coute très cher car l'entreprise doit revenir au RTL chercher le problème corriger et refaire le flot de conception et de la vérification.

ST-Ericsson dispose d'un outil Spyglass d'Atrenta qui permet de faire la vérification du CDC et aussi des mécanismes de synchronisation (proposé par Atrenta) qui représentent la solution aux problèmes CDC.

Le problème est qu'il n'y a pas une méthodologie bien spécifique pour faire la vérification du CDC, et qui assure qu'aucun problème CDC n'existe dans le circuit.

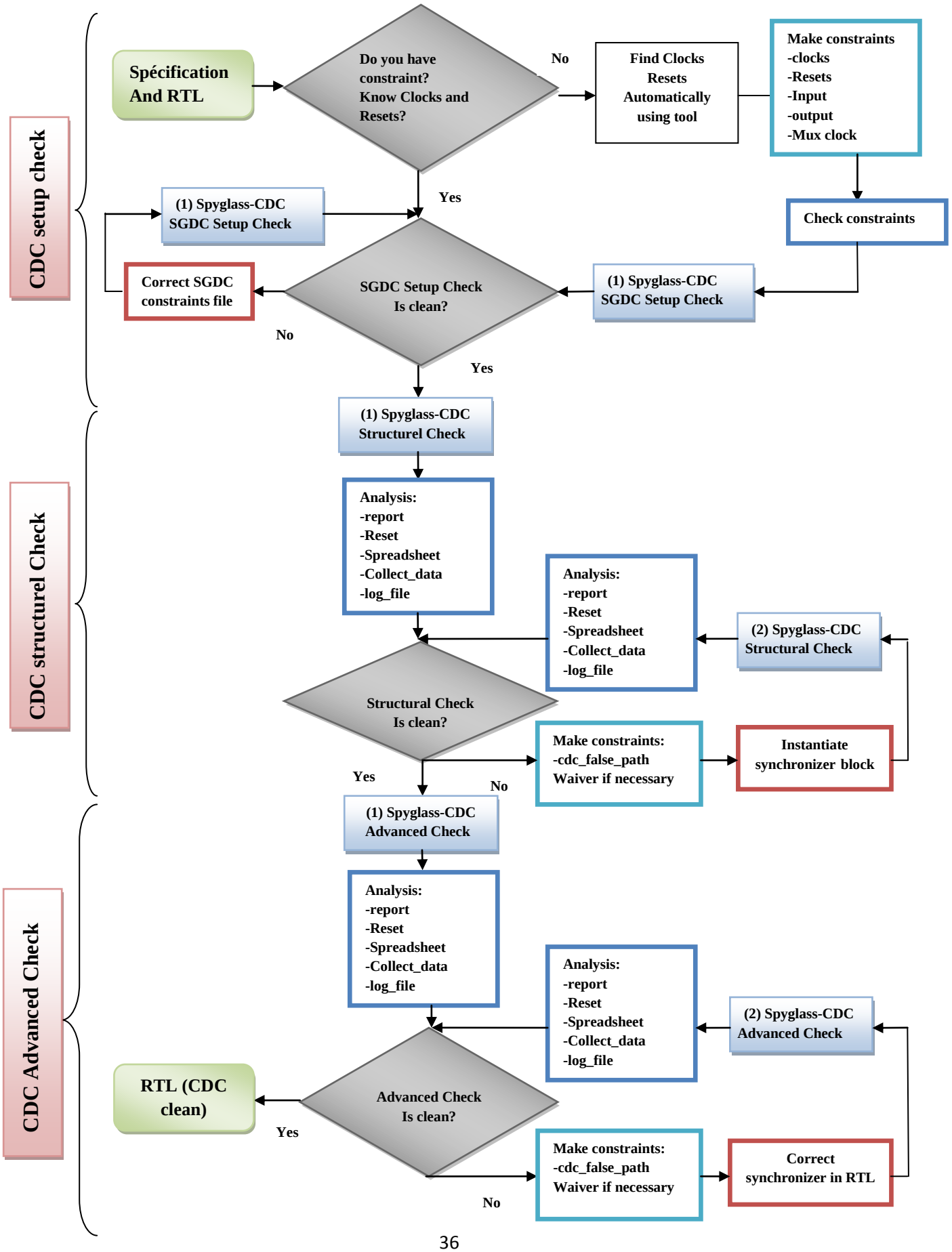
Une méthodologie doit répondre à ces trois questions :

- Quel est le chemin le plus optimal qu'on doit suivre pour réussir une vérification CDC rapide et efficace ?

- Quel sont les contraintes qu'on doit fournir à l'outil pour qu'il maitrise touts les passages CDC dans l'IP ?
- Quand et comment on peut utiliser les mécanismes de synchronisation proposés par Atrenta ?

V.2 Méthodologie pour la vérification CDC:

L'algorithme ci dessous représente la méthodologie que je propose pour la vérification du CDC en utilisant l'outil Spyglass-CDC :


CDC setup check
Spécification And RTL
Do you have constraint? Know Clocks and Resets?
No
Find Clocks Resets Automatically using tool
Make constraints -clocks -Resets -Input -output -Mux clock
Check constraints
Yes
(1) Spyglass-CDC SGDC Setup Check
Correct SGDC constraints file
SGDC Setup Check Is clean?
(1) Spyglass-CDC SGDC Setup Check
No
Yes
CDC structurel Check
(1) Spyglass-CDC Structurel Check
Analysis: -report -Reset -Spreadsheet -Collect_data -log_file
Analysis: -report -Reset -Spreadsheet -Collect_data -log_file
(2) Spyglass-CDC Structural Check
Structural Check Is clean?
Make constraints: -cdc_false_path Waiver if necessary
Instantiate synchronizer block
Yes
No
CDC Advanced Check
(1) Spyglass-CDC Advanced Check
Analysis: -report -Reset -Spreadsheet -Collect_data -log_file
Analysis: -report -Reset -Spreadsheet -Collect_data -log_file
(2) Spyglass-CDC Advanced Check
RTL (CDC clean)
Advanced Check Is clean?
Yes
Make constraints: -cdc_false_path Waiver if necessary
Correct synchronizer in RTL
No

V.2.1 Description :

L'algorithme ci dessus décrit la méthodologie que je propose pour la vérification du Clock Domain Crossing, qui fera en sorte que le circuit soit bien conçu pour gérer correctement les problèmes du passage de la donnée d'un domaine d'horloge à un autre.

La vérification se fait à l'aide d'outil Spyglass CDC d'Atrenta, il permet d'identifier rapidement les problèmes réels au sein des réseaux d'horloges et cela au stade le plus précoce possible du flot de conception.

SpyGlass-CDC permet d'appliquer une gamme de techniques d'analyse structurelle et fonctionnelle qui nous rassurent qu'aucun problème CDC n'existe.

Avant de passer à la vérification structurelle et fonctionnelle on doit valider la vérification du setup qui assure que les contraintes fournies à l'outil (input, output, horloge, reset, Mux d'horloge, quasi-static) sont correctes, si cette partie est non valide en risque d'avoir de fausses erreurs dans la vérification structurelle et fonctionnelle.

Une fois que la vérification du setup est valide on passe à la vérification structurelle, à ce stade l'outil identifie les passages CDC et vérifie l'existence des mécanismes de synchronisations, s'il y a des erreurs d'outil qui ne sont pas de vrais erreurs et qui sont justifiées par le concepteur on peut utiliser la contrainte (cdc_false_path) ou l'option « Waivers » pour les masquer, s'il y a des chemins qui nécessitent un mécanisme de synchronisation on l'instancie dans le RTL.

Si la vérification structurelle est valide on peut passer à la vérification fonctionnelle, à cette étape l'outil vérifie les problèmes fonctionnels (perte de donnée) au niveau des mécanismes de synchronisations utilisés, si les erreurs affichées par l'outil ne sont pas de vraies erreurs on peut utiliser la contrainte cdc_false_path ou les « Waivers » pour les masquer, s'il y a de vraies erreurs on fait des corrections dans le RTL.

Une fois que les trois vérifications sont valides, on peut être sur que le circuit est robuste devant les problèmes CDC.

V.2.2 Concept (SGDC : SpyGlass Design Constraints)

Directives utilisées pour fournir des informations qui n'apparaissent pas dans le RTL. Ces contraintes représentent une description de l'IP permettant à l'outil de reconnaitre les différents chemins CDC, les contraintes sont écrites dans un fichier texte d'extension .sgdc.

La vérification du CDC par l'outil nécessite de tourner les 3 Template (SGDC_Setup_Check, Structural_Check et Advanced_check …)

V.2.3 SGDC_Setup_Check :

Dans cette partie de vérification du CDC l'outil Spyglass analyse la syntaxe du fichier de contrainte ainsi les contraintes qui sont nécessaires pour la vérification du CDC et qui ne sont pas écrites dans le fichier de contraintes, cette vérification donne en résultats de sortie différents règles

Règle selon l'ordre de priorité SGDC Setup Check	Description
Clock_info03a	Des horloges ne sont pas définies.
Reset_info09a	Des ports reset ne sont pas définis.
Clock_info18	Des ports input et output ne sont pas définis.
Clock_info05a	Il y a des Mux d'horloges qui nécessitent des contraintes

Tableau V.1 Règle du SGDC Setup Check selon l'ordre de priorité

.

V.2.4 Structural_Check :

Dans cette partie de vérification structurelle du CDC l'outil analyse les passages CDC dans le RTL et vérifie l'existence des différentes structures de synchronisation (Multi-flop, Mux de recirculation, Fifo asynchrone …) instancié dans le RTL, cette vérification donne en résultat de sortie différents règles.

Règle selon l'ordre de priorité CDC Structurel Check	**Description**
Clock_sync01	Il y a un passage CDC nécessite un synchroniseur
Reset_sync01	Des Reset asynchrones, qui doivent être synchronisé par le synchroniseur des ports reset
Clock_sync08	Bus de données synchronisé par le synchroniseur Multi-flop
Clock_sync02	Liste les différents mécanismes instanciés dans l'IP

Tableau V.2 Règles de la vérification structurel selon l'ordre de priorité

V.2.5 Advanced Check :

Dans cette partie de vérification Fonctionnelle du CDC l'outil analyse le fonctionnement des mécanismes de synchronisation instanciés dans le RTL. Cette vérification résulte en sortie différents règles.

Règle selon l'ordre de priorité : CDC Advanced Check.	**Description**
Ac_cdc01a	Perte de la donnée lors du passage d'un domaine de fréquence rapide à un domaine de fréquence long
Ac_fifo01	Statuts (plain/vide) de la Fifo
Ac_handshake	Perte des données au niveau du Handshake

Tableau V.3 Règle du CDC Advanced Check selon l'ordre de priorité

V.2.6 Contraintes CDC :

a. Horloges :

Les horloges permettent à l'outil d'identifié les différents passages CDC dans l'IP.

- La définition des horloges du design a une grande importance pour une vérification correcte du CDC. Si les hypothèses périodes/edges sont erronées ou non déclarées, peuvent provoquer de fausses violations.

- Si une horloge n'est pas déclarée dans le fichier de contrainte elle ne sera pas vérifiée par l'outil.

```
clock    -name pcm_i2s_clk
         -domain master_domain
         -period 26.0416666
         -edge {0 13.0208333}
```

Dans La vérification CDC setup on doit s'assurer que la règle SGDC_setup_check/**Clock_info03a**, pour les horloges non déclarées dans le fichier de contraintes, n'apparait pas dans les messages d'erreurs.

b. Resets

Si le circuit comporte une reset synchrone et que le signal reset en entrée est asynchrone au circuit on risque d'entrainer le circuit dans un état métastable si on ne met pas un synchroniseur, La contrainte reset permet de spécifier les ports « Reset » du circuit et leur nature synchrone ou asynchrone.

```
reset -name reset_n
      -value 1
      -async
```

On doit vérifier la règle SGDC_Setup_Check/**Reset_info09a** qui indique les ports Resets non spécifié dans le fichier de contrainte.

c. Inputs

On doit assurer que les inputs écrits dans le fichier de contrainte sont synchronisées par l'horloge convenable. Si un input est non déclarée dans le fichier de contrainte elle ne sera pas vérifiée par l'outil.

Si les inputs sont synchrones au circuit on met l'horloge convenable à cet input.

```
input -name A_in  -clock clk_A.
```

Si les inputs sont asynchrones au circuit on leur accorde des horloges virtuelles.

```
input -name A_in  -clock virtuel_clk
```

- des inputs appartenant au même domaine d'horloge ont la même horloge virtuelle.

Vérifier la règle SGDC_Setup_Check/ **Clock_info18 a**, qui assure que tous les ports sont déclarés dans le fichier de contraintes.

d. Les outputs

La contrainte output est utilisée pour spécifier le domaine d'horloge dans les ports de sorties de l'IP.

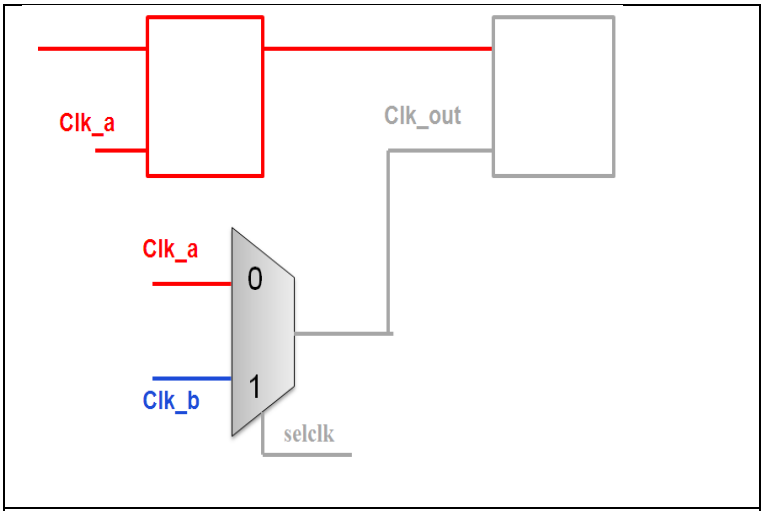
```
Output -name out_p -clock p_clk
```

On doit s'assurer que la règle SGDC_Setup_Check/ **Clock_info18 a**, qui assure que tous les ports sont déclarés dans le fichier de contrainte.

e. Mux d'horloges

Les arbres d'horloges utilisent souvent des Multiplexeur combinant différentes horloges pour différents modes de fonctionnement dans la conception. Dans de tels cas, il y a deux solutions, soit on définit le mode de fonctionnement en limitant la sélection du multiplexeur en utilisant la contrainte « **set_case_analysis** » dans le fichier des contraintes, ou on définit l'horloge à la sortie du multiplexeur.

Lors de l'analyse des résultats on doit s'assurer que la règle SGDC_Setup_Check/**Clock_info05a**, n'apparait pas comme message d'erreur.



Figue V.1: Mux d'horloges

```
set_case_analysis –name selclk  -value 1
```

f. Fifo asynchrone

On utilise La Fifo asynchrone comme mécanisme de synchronisation, normalement la Fifo asynchrone est reconnue par l'outil, elle doit être détectée lors de la vérification structurelle dans la règle Clock_sync02.

- Si la Fifo est non reconnue par l'outil on peut la spécifier dans le fichier de contrainte.

```
fifo [-memory <memory_name>]
     [-rd_data <read_data>]
     [-wr_data <write_data>]
     [-rd_ptr <read_pointer>]
     [-wr_ptr <write_pointer>]
```

- Etre sure que les statues **overflows/underflow** qui signifient qu'on a dépassé les limites de la mémoire de la Fifo et qu'on risque de perdre des données n'apparait pas dans la vérification fonctionnelle du CDC **Adv_checks** (Ac_fifo01).

g. Les faux chemins CDC

La contrainte **cdc_false_path** permet d'annuler des chemins qui ne doivent pas être vérifié, pour des raisons du concepteur qui doivent être justifiés. On peut spécifier les faux chemins CDC par la contrainte.

```
cdc_false_path   [-from <obj1_name>]
                 [-to <obj2_name>]
                 [-through <obj3-name>]
```

V.3.6.h Quasi-statique

Désigne des flops qui prennent une valeur constante pendant le fonctionnement régulier d'un design (ils peuvent changer la valeur que durant l'installation ou l'initialisation du design, on les appelle aussi les registres de configuration) Souvent, les flops quasi-statiques ne nécessitent pas de synchroniseurs, même si elles sont impliquées dans les CDC.

Il existe une contrainte Spyglass « quasi_static» permettant de spécifier les signaux quasi statiques dans la conception.

```
Quasi_static –name <sig-name>
```

Les contraintes CDC, soit on les écrits dans le fichier des contraintes, ou on les spécifie à l'aide de l'interface graphique de l'outil (Figure V.2)

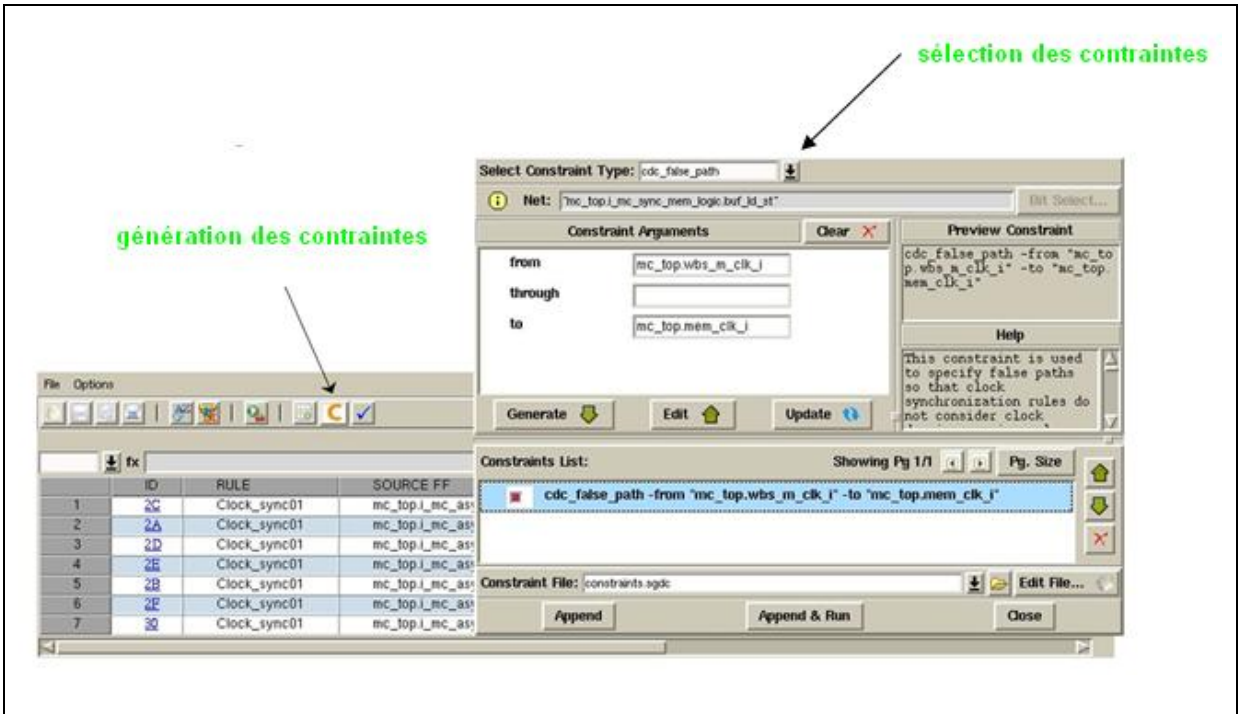


Figure V.2 : Fenêtre Spyglass pour la génération des contraintes

V.2.7 Waivers

Parfois quand on analyse les messages d'erreurs affichés par l'outil, il y a des erreurs qui ne sont pas de vraies erreurs, l'outil permet l'option waivers pour masquer ces messages avec justificatif (pourquoi ce message d'erreur n'est pas une vraie erreur) écrit par le concepteur.

Quand on refait le test avec l'outil Spyglass, les messages masqués par l'option Waivers n'apparait plus dans les messages d'erreurs.

Cette option permet d'optimiser la vérification CDC.

V.3 Etude des mécanismes de synchronisation

V.3.1 Problème

Atrenta propose différents mécanismes de synchronisation (décrit dans le chapitre précédent), le problème est que on doit connaitre lequel de ces mécanismes de synchronisation est convenable a un tel problème CDC, a savoir que si on ajoute une seule flip flop dans l'IP, peut changer tout le fonctionnement de cette IP, une mauvaise utilisation de ces mécanismes peut conduire aux problèmes de perte de données ou d'incohérence de données, le but de cette étude est de définir quand on peut utiliser ces mécanismes selon les ratios de fréquences pour ne pas avoir des problèmes CDC.

La méthode adopté pour étudier ces mécanismes de synchronisations consiste à mettre le synchroniseur entre deux domaines d'horloges asynchrones domaine source A et domaine de destination B.

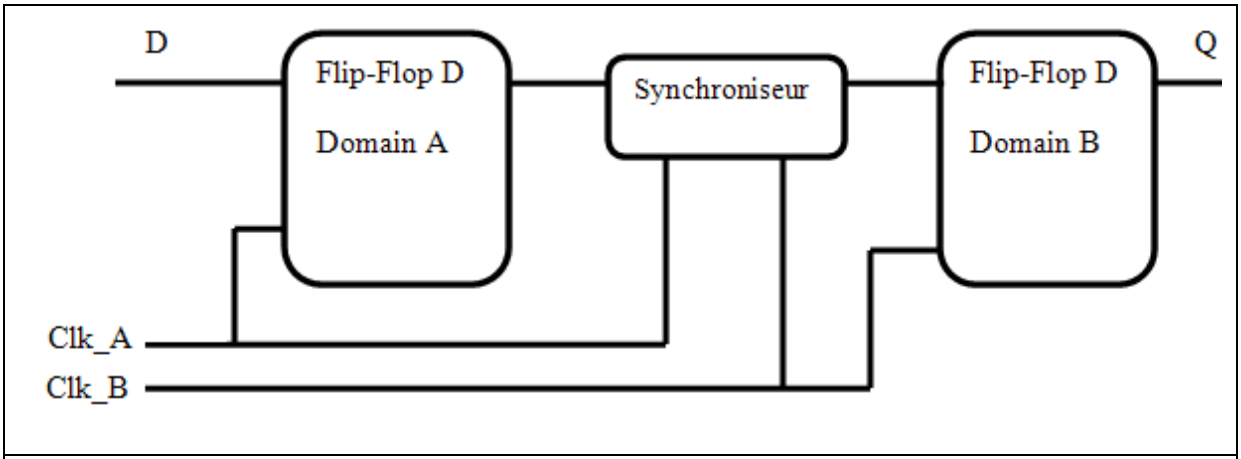


Figure V.3 : Schéma de test des mécanismes de synchronisation

V.3.2 Etapes d'études.

Le tableau ci dessous décrit les étapes suit pour l'élaboration de cette étude.

Etapes	**Description**
1	Exploration des différents mécanismes de synchronisation proposée par Atrenta.
2	Conception.
3	Simulation fonctionnel de ces mécanismes (Ncsim) selon les ratios de fréquences (Test Bench).
4	Test Spyglass des mécanismes de synchronisation selon les ratios de fréquence (fichier de contraintes).

Tableau V.4 : étapes d'étude des mécanismes de synchronisation

V.3.3 Résultats d'études

Le tableau ci dessous représente les résultats de cette étude. Dans le tableau les mécanismes de synchronisation sont classés selon l'ordre de complexité.

Mécanisme de synchronisation	Test en fréquence Domaine de fréquence long -> Domaine de fréquence rapide	Test en fréquence Domaine de fréquence rapide -> Domaine de fréquence long	Passage de la donnée à travers un bit	Passage de la donnée à travers un bus de donnée
Multi-flop	Ok	Not Ok (perte de donnée)	Ok	Not OK (incohérence de données)
Multiplexeur de Recirculation	Ok	Not Ok (perte de donnée)	Ok	Not OK (incohérence de données)
Code Gray	Ok	Not Ok (perte de donnée)		Ok (bus incrémental)
Handshake	Ok	Ok	Ok	Ok
Fifo asynchrone	Ok	Ok	Ok	Ok

Tableau V.5 : Résultats d'études de synchronisation

a. Multi-flop :

Commençant par le synchroniseur basique le **Multi-flop**, ce synchroniseur n'est robuste aux problèmes CDC que si on a un passage d'un domaine d'horloge source (de fréquence lente) à un domaine d'horloge de destination (de fréquence rapide), si on inverse on risque de perdre les données (Figure V.5)

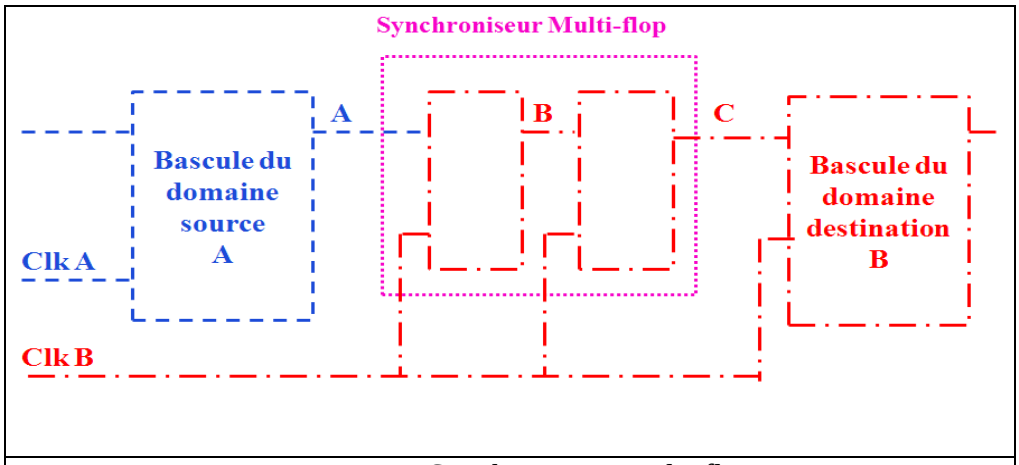


Figure V.4 : Synchroniseur Multi-flop

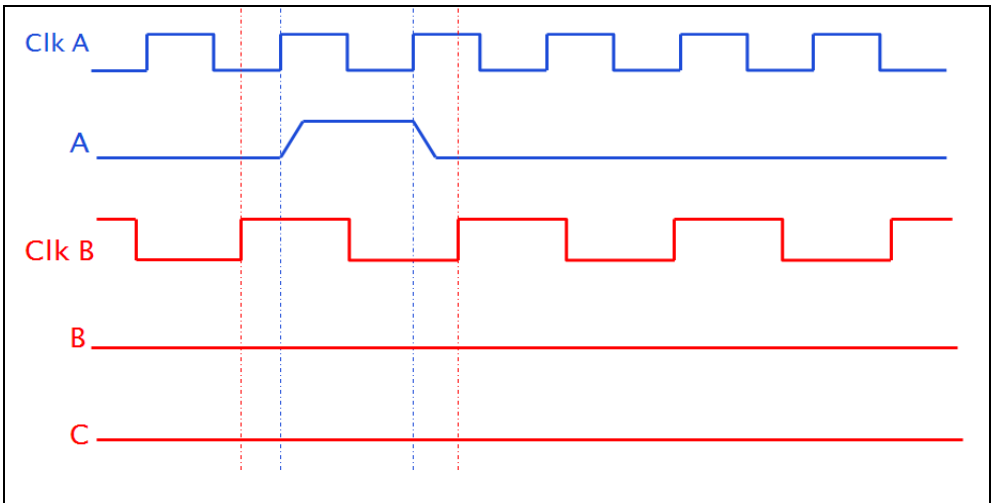


Figure V.5 : Perte de la donnée au niveau du synchroniseur Multi-flop

b. Synchroniser un bus en utilisant le multi-flop :

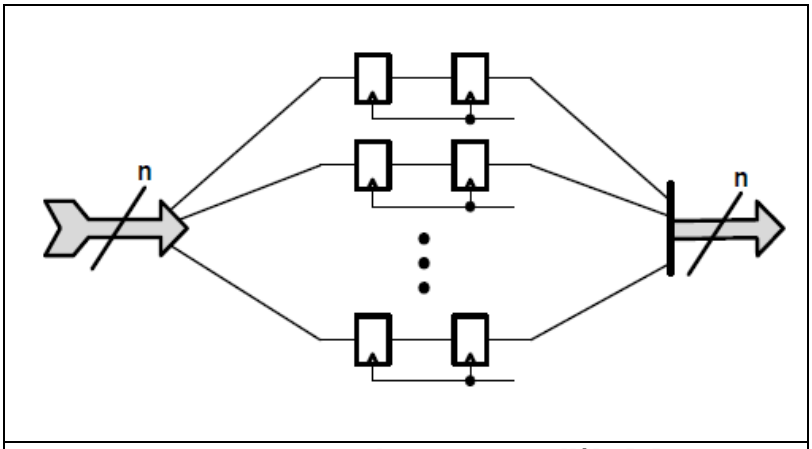


Figure V.6 : Synchroniseur parallèle [4]

La Figure ci-dessus représente une architecture utilisée dans le but de synchroniser les données d'un mot multi-bit en utilisant le synchroniseur Multi-flop par bit. Ce schéma représente un synchroniseur correct, les bits de données sont synchronisées et pas de logique combinatoire insérés entre les flip-flops. Cependant, les données synchronisées ne seront pas toujours correct. Si les échanges des données en entrée sont près de la fenêtre métastable, chaque synchroniseur multi-flop peut finir par faire quelque chose différents: certains peuvent prendre les données avant le front montant, d'autres peuvent prendre les nouvelles données, et d'autres peuvent entrer dans un état métastable. Il n'y a aucun moyen de garantir la cohérence des données multi-bits en sortie. Un bus de données cohérentes à plusieurs bits ne peut pas être synchronisé à l'aide d'un synchroniseur parallèle.

c. Mux de recirculation :

Le deuxième synchroniseur **Mux de recirculation**, ce synchroniseur n'est robuste que si on a un passage d'un domaine d'horloge source (de fréquence lente) à un domaine d'horloge de destination de fréquence rapide, ce synchroniseur est utilisé spécifiquement pour les bus de données, il permet d'assurer qu'il n y a aucun de problème d'incohérence de données au niveau du bus (Figure V.7).

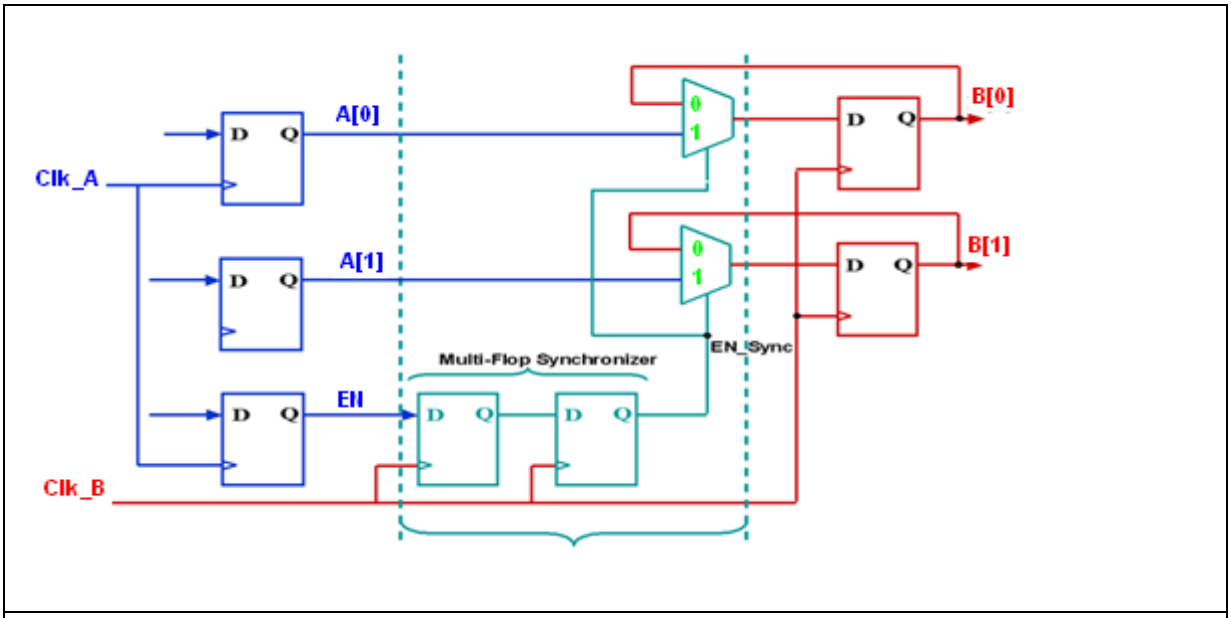


Figure V.7: Schéma du test du mécanisme Mux de Recirculation

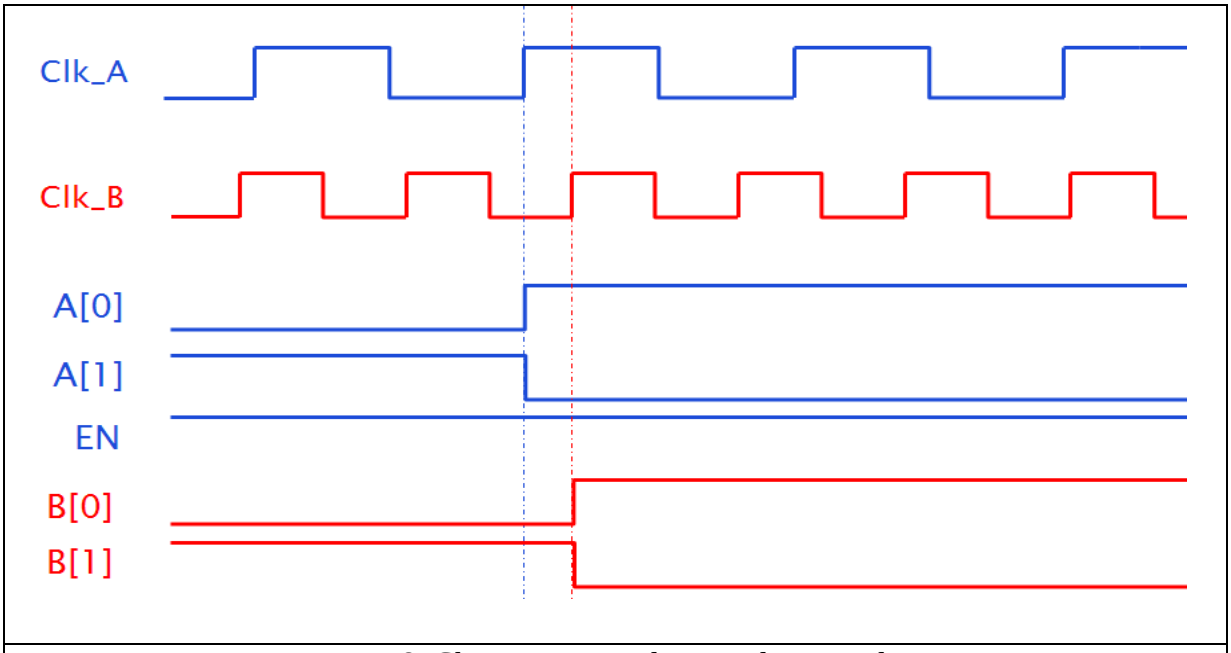


Figure V.8: Chronogramme du Mux de recirculation

d. Code Gray

La troisième technique de synchronisation consiste à synchroniser les bus des données codé en gray en utilisant le synchroniseur Multi-flop (Figure V.8), on peut utiliser ce mécanisme pour transférer les données d'un domaine d'horloge source (de fréquence lente) à un domaine d'horloge de destination de (fréquence rapide), pour utiliser ce mécanisme on doit être sur que le bus s'incrémente d'un état à un autre par 1.

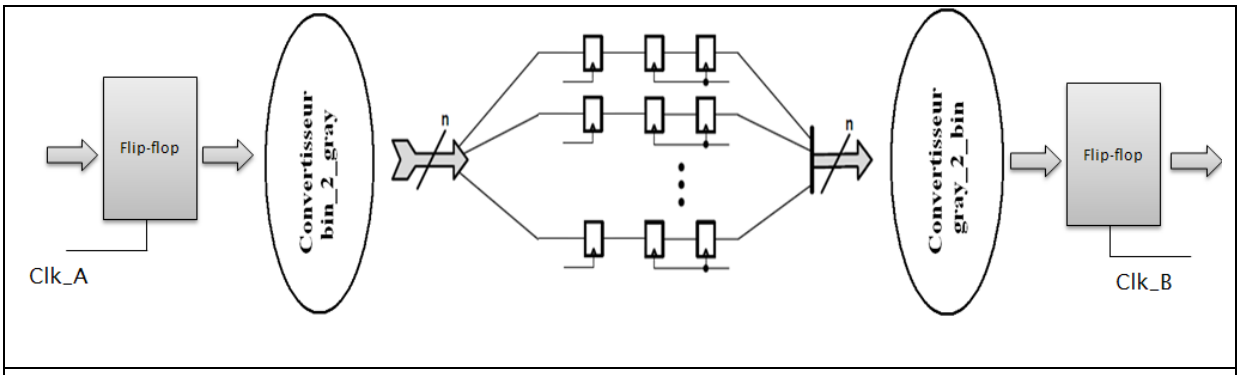


Figure V.9: Synchronisation des bus de données codé en gray par le synchroniseur Multi-flop

e. Handshake :

Le quatrième synchroniseur **Handshake**, il se compose de deux machines à états et les deux pointeurs (req, ack), cette structure complexe lui permet de contrôler le passage des données source sans perte, il permet de transférer la donnée quelque soit la fréquence des deux domaines d'horloges.

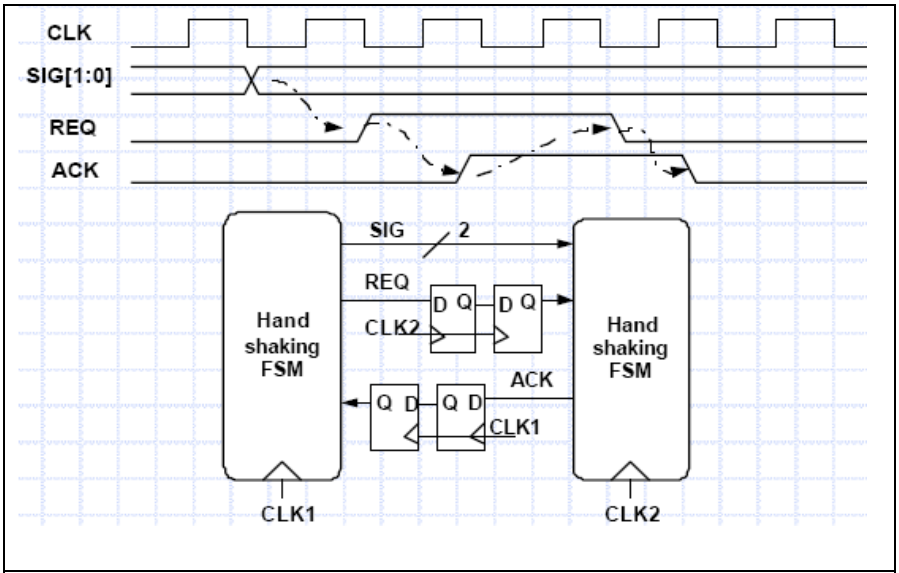


Figure V.10 : Le Handshake

f. Fifo asynchrone:

Le **Fifo asynchrone** est le synchroniseur le plus complexe par rapports aux autres mécanismes de synchronisation, il se caractérise par une structure complexe (pointeur (Read, Write), mémoire) lui permet de contrôler le passage de la donnée entre les deux domaines d'horloge, la Fifo asynchrone assure que la donnée est stable en sortie et qu'il n'y a pas de problème de perte de donnée quelque soit les fréquences des deux domaines.

V.4 Automatisation de la vérification :

V.4.1 Problème :

Lors des tests effectués pour l'élaboration de cette méthodologie, j'ai rencontré plusieurs problèmes.

- Besoin d'informations CDC sur l'IP avant de passer le test.

- Mécanisme de synchronisation utilisé (Multi-flop handshake, Fifo …).
- Flip-flop Magic.
- L'existence des PLL ou des générateurs d'horloge.
- Sorties venant d'une logique combinatoire qui sont justifié.
- Les signaux quasi statiques.
- Les Mux d'horloge de test.

- Ecrire un bon fichier de contraintes demande une bonne maitrise du fonctionnement de l'IP.

V.4.2 Information CDC nécessaire sur l'IP avant de commencer la vérification

Avant de commencer la vérification du Clock Domain Crossing on doit avoir les informations suivantes sur l'IP.

V.4.2.1 Mécanisme de synchroniseur utilisé

On a besoin des informations sur les mécanismes de synchronisation utilisés dans l'IP:

a. Multi-flop

Pour ce synchroniseur basique on doit savoir combien y a-t-il de flip-flop de synchronisations utilisées et combien de foi il est instancié dans l'IP, pour que l'outil reconnaisse ce mécanisme, et aussi s'il y a d'autres spécifications à propos de ce mécanisme.

b. Multiplexeur de recirculation

On doit savoir combien de fois ce synchroniseur est instancié dans l'IP, on ne peut pas faire confiance à l'outil, parfois l'outil détecte des Mux comme Mux de recirculation qui ne sont pas utilisé pour la synchronisation.

c. Flip flop Magic :

S'il y a des flip-flop Magic dans l'IP on doit les spécifier à l'outil pour qu'il ne les considère pas comme des flip-flops simples.

d. Handshake :

On doit savoir si un handshake existe dans l'IP, parfois l'outil ne détecte pas ce mécanisme, on risque d'avoir des erreurs qui ne sont pas de vrais erreurs.

e. Fifo asynchrone :

On doit savoir s'il y a une Fifo asynchrone instanciée dans l'IP et aussi les paramètres de cette Fifo (Read_data, Write_data, mémory, Read_pointer, Write_pointer).

V.4.2.2 L'existence des PLLs :

Si une PLL existe dans l'IP, on doit spécifier à l'outil les horloges qui sort de cette PLL pour que l'outil les prenne en considération et aussi pour éviter le risque d'avoir un problème CDC.

V.4.2.3 Sortie venant d'une logique combinatoire :

Parfois il y a des sorties venant d'une logique combinatoire, le designer de l'IP assure que dans ce chemins la donnée est stable, il y'aura pas de problème CDC, on doit connaitre ces chemin pour les spécifier à l'outil.

V.4.2.4 Multiplexeurs d'horloges :

L'outil n'est pas capable d'identifier les Multiplexeurs d'horloges qui existent dans la conception, on doit connaitre ces Multiplexeurs pour les spécifier à l'outil à l'aide de la commande « set_case_analysis » pour ne pas avoir des erreurs qui ne sont pas de vraies erreurs.

V.4.3 Tableur CDC:

La solution proposée à ce problème est de créer un tableur CDC qui comporte toutes les informations CDC nécessaires pour faire la vérification, et qu'il doit être toujours accompagné par l'IP.

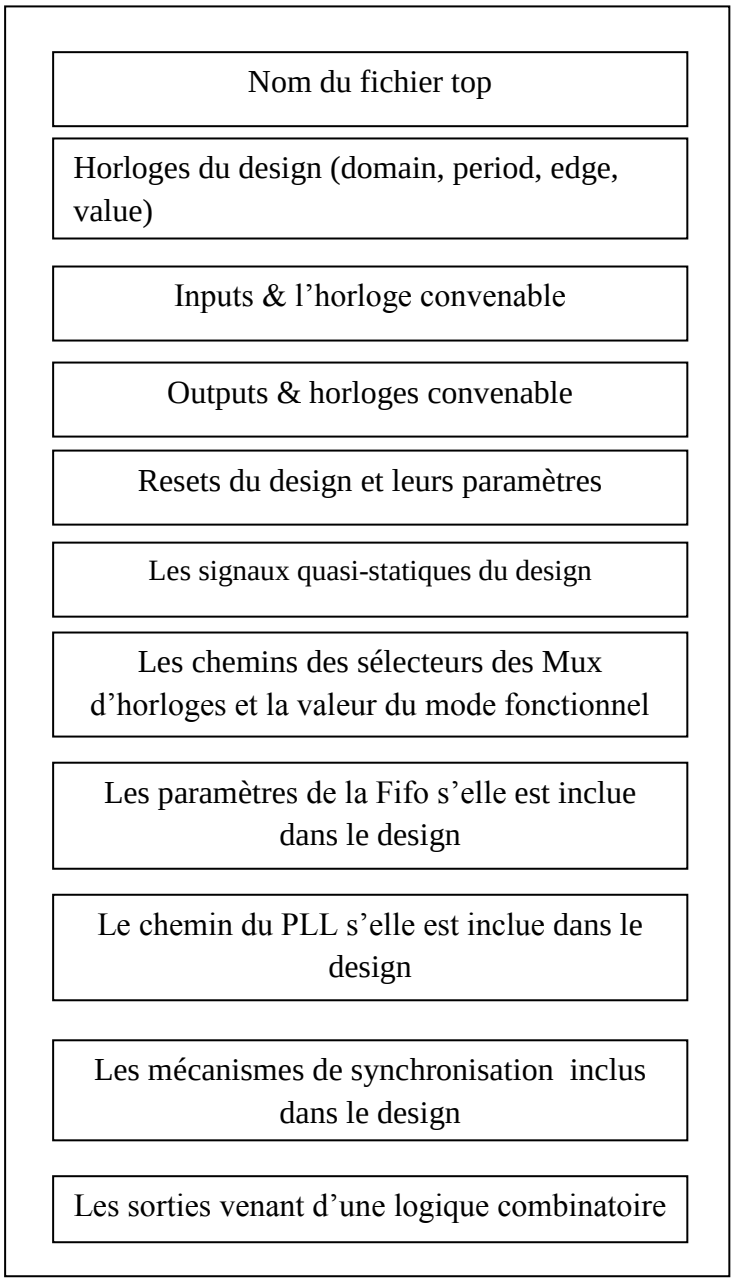


Figure V.11 : Tableur CDC

V.4.4 Automatisation :

Le concept d'automatisation des taches d'un processus, est un sujet au cœur de l'actualité industrielle. Une automatisation peut être totale dans les domaines qui ne nécessitent pas une grande intervention humaine, tandis que dans certain cas, elle est seulement partielle, puisque les taches du processus utilisé ne peuvent être dissociées du savoir-faire humain.

Pour notre cas on ne peut pas automatiser toute la vérification du Clocks Domaine Crossing, on ne peut pas faire confiance à l'outil, l'intervention du concepteur est nécessaire pour conduire la vérification (spécifier les faux chemin CDC, la fréquence des horloges, instancier les mécanismes de synchronisation…).

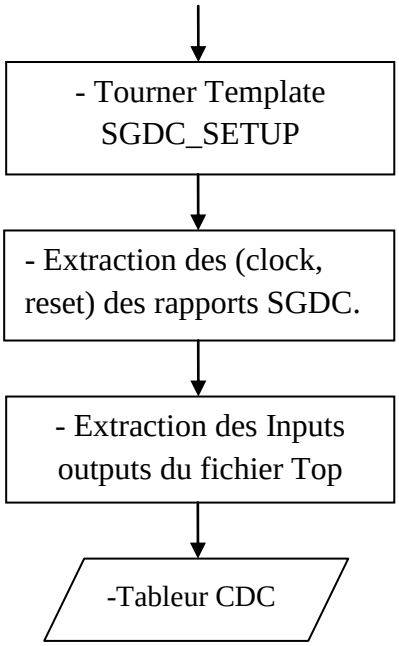
Dans le but de faciliter la vérification CDC on propose un script en perl qui permet la création du tableur CDC avec le minimum de contraintes et la génération du fichier de contraintes d'extension (.sgdc) à partir du tableur CDC avec une manière transparente au concepteur.

V.4.4.1 Langage Perl :

Perl est un langage de programmation créé par Larry Wall en 1987 et reprenant des fonctionnalités du langage C et des langages de scripts sed, awk et shell (sh). C'est un langage interprété, polyvalent, et particulièrement adapté au traitement et à la manipulation de fichiers texte, notamment du fait de l'intégration des expressions régulières dans la syntaxe même du langage.

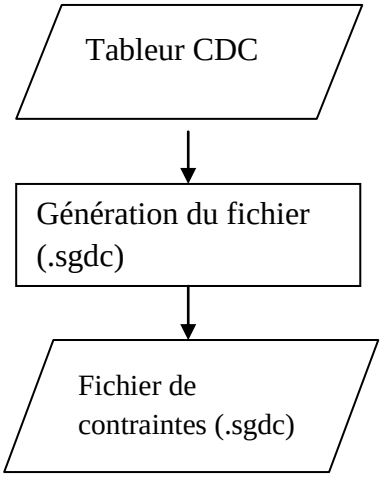
V.4.4.2 Création du tableur CDC:

L'outil spyglass comporte une Template (SGDC_Setup) , permet de générer des rapports contenant des informations sur les horloges et les ports reset du design, l'idée est d'exploiter ces rapports (Clock, Reset) spécifié par l'outil et créer un tableur CDC avec le minimum de contraintes (inputs, output, clock, reset), cette taches est faite à l'aide d'un script en perl.



V.4.4.3 Génération du fichier de contraintes (.sgdc) :

La deuxième fonctionnalité du script est qu'il permet la génération du fichier d'extension (.sgdc) remplis des contraintes en respectant la syntaxe à partir du tableur CDC.



Conclusion générale

Lors de mon projet réalisé à ST-ERICSSON, j'ai pu répondre au cahier des charges en mettant une méthodologie qui assure que l'IP est correctement conçu et qu'aucun problème CDC n'existe. Aussi dans le but de faciliter la vérification CDC de l'IP j'ai réussi à automatisé des parties qui ne demande pas beaucoup du savoir faire humain.

Ce travaille était bénéfique et enrichissant en termes d'expérience professionnelle et personnelle. Il m'a permis de comprendre le flot de conception des circuits intégrés propre à ST-Ericsson.

Aussi en terme de connaissance, il m'a permis d'explorer les différents problèmes CDC et les mécanismes de synchronisation ainsi d'avoir une vision profonde sur la synchronisation en générale.

En plus au cours de ce stage j'étais amené à maitriser plusieurs environnements de travail ce qui demande des capacités d'analyse, de patience et de structuration.

En outre, ce stage a été une occasion pour me familiariser avec le langage perl et amélioré mes connaissance du coté programmation et algorithmique.

Enfin, ce stage a représenté une vraie opportunité pour développer le sens des relations humaines et pour réaliser un esprit d'équipes et d'initiatives, de l'organisation, de l'aptitude à la communication et à la recherche d'informations. C'était un défi qui m'a incité à surmonter toutes les difficultés pour atteindre les objectifs visés.

Liste des Figure

Liste des Tableaux

Bibliographie

[ref 1]- 0-In® CDC Analyzer User Guide Auteur: Mentor graphic

[ref 2]- CONCEPTION DE RESEAUX DE COMMUNICATION SUR PUCE ASYNCHRONES : APPLICATION AUX ARCHITECTURES GALS Auteur : Jérôme QUARTANA en année 2004

[ref 3]- Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC Auteur: Xuan-Tu TRAN en année 2008

[ref 4]- CONCEPTION ET IMPLANTATION D'UN MICRO-RESEAU SUR PUCE AVEC GARANTIE De SERVICE Auteur : Ivan MIRO PANADES en année 2008

- SpyGlass® Clock-Reset Rules Reference Auteur: Atrenta

- FORMALLY VERIFYING CLOCK DOMAIN CROSSING JITTER USING ASSERTION-BASED VERIFICATION Auteur: Tai Ly; Neil Hand; Chris Ka-kei Kwok

- Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs

Auteur: SNUG-2001 San Jose, CA

- Understanding clock domain crossing issues

Auteur: By Saurabh Verma Engineering Manager Atrenta & Ashima S. Dabare Consulting Applications Engineer Atrenta

Annexe

```
#############################################################################
#
#                                  current_design
#
#############################################################################
#
current_design     pcm_i2s_top
#############################################################################
#
#                                       input
#
#############################################################################
#
input         -name psel                              -clock p_vclk
input         -name penable                           -clock p_vclk
input         -name pwrite                            -clock p_vclk
input         -name paddr                             -clock p_vclk
input         -name pwdata                            -clock p_vclk
input         -name pcm_i2s_tx_dma_clr                -clock vclk
input         -name pcm_i2s_rx_dma_clr                -clock vclk
input         -name pcm_i2s_ws_in                     -clock vclk_in
input         -name pcm_i2s_tx_data_in                -clock vclk_in
input         -name pcm_i2s_rx_data_in                -clock vclk_in
#############################################################################
#
#                                       output
#
#############################################################################
#
output        -name pcm_i2s_irq_n                     -clock pclk
output        -name pcm_i2s_frame_irq_n               -clock pclk
output        -name prdata                            -clock pclk
output        -name pcm_i2s_tx_dma_breq               -clock pclk
output        -name pcm_i2s_rx_dma_breq               -clock pclk
output        -name pcm_i2s_out_sclk                  -clock pcm_i2s_clk
output        -name pcm_i2s_out_sclk_en_n             -clock pcm_i2s_clk
output        -name pcm_i2s_ws_out                    -clock pcm_i2s_clk
output        -name pcm_i2s_ws_out_en_n               -clock pcm_i2s_clk
output        -name pcm_i2s_tx_data_out               -clock pcm_i2s_clk
output        -name pcm_i2s_tx_data_out_en_n          -clock pcm_i2s_clk
output        -name pcm_i2s_rx_data_out               -clock pcm_i2s_clk
output        -name pcm_i2s_rx_data_out_en_n          -clock pcm_i2s_clk
output        -name clock_name                        -clock pclk
#############################################################################
###
#                                        clock
#
#############################################################################
###
clock         -name pcm_i2s_clk              -domain master_domain    -period
26.0416666    -edge {0 13.0208333}
clock         -name pcm_i2s_in_s_clk         -domain apb_domain       -period
5                   -edge {0 2.5}
```

```
clock        -name p_clk                    -domain serial_domain   -period
52.0833332   -edge {0 26.0416666}
##############################################################################
###
#                                        reset
#
##############################################################################
###
reset -name reset_n      -value 1    -async
reset -name preset_n     -value 1    -async
##############################################################################
###
#                           voici les quasi_static
#
##############################################################################
###
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.tx_enable
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.rx_enable
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.mode
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.sclkpol
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.datasize
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.nb_channel
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.tx_ch_enable
quasi_static       -name pcm_i2s_top.a_pcm_i2s_s_trx.rx_ch_enable
##############################################################################
###
#                                  set_case_analysis
#
##############################################################################
###
set_case_analysis        -name tcr_noinvert_clk        -value 1
##############################################################################
###
#                                          fifo
#
##############################################################################
###
curent_design pcm_i2s_fifo
fifo   -memory ram_data        -rd_data data_out       -wr_data di       -
rd_ptr read_ptr    -wr_ptr write_ptr
```

Fichier de contraintes .sgdc(PCM_I2S)

Résumé

Ce rapport porte sur l'étude des différents problèmes du passage de la donnée d'un domaine d'horloge à un autre (CDC : Clock Domain Crossing), et les mécanismes de synchronisation qui représentent une solution à ces problèmes.

Sur la base de cette étude et des tests effectué sur des IP par l'outil Spyglass CDC, nous développons une nouvelle méthodologie pour la vérification du CDC qui permet de corriger les problèmes au niveau du RTL la ou on peut facilement corriger les problèmes, cette méthodologie assure que le circuit est correctement conçu et capable de traverser tous les domaines d'horloge sans aucun problème CDC,

Ainsi dans le but de faciliter et optimiser la vérification, un script a été proposé testé et validé, permettant d'automatiser des parties dans la vérification avec une manière transparente au concepteur.

Abstract

This report focuses on the study of various problems switching from one clock domain to another (CDC), and synchronization mechanisms that represent a solution to these problems.

Based on this study and testing of IP by the CDC spyglass tool we develop a new methodology for the verification of the CDC that allows correct problems at the RTL that can easily correct the problems, this methodology ensure that circuit that has been designed properly to handle clock domain crossing issues,

Thus in order to facilitate and optimize the audit, a script has been proposed tested and validated, to automate parties of check CDC with transparent manner to the designer.