REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DE TECHNOLOGIE
DEPARTEMENT D'ELECTRONIQUE
N° :……..

DOMAINE : SCIENCE ET TECHNOLOGIE
FILIERE : ELECTRONIQUE
OPTION : SYSTEMES EMBARQUES

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: Taibi Souad

Intitulé

ANALYSE ET SYNTHÈSE D'UN UART EN VHDL

Soutenu devant le jury composé de:

Bentoumi Miloud	Université Mohamed Boudiaf M'sila	Président
Ballouti Adel	Université Mohamed Boudiaf M'sila	Rapporteur
Brik Mourad	Université Mohamed Boudiaf M'sila	Co-Rapporteur
Bakhti Elhaddi	Université Mohamed Boudiaf M'sila	Examinateur

Année universitaire : 2022 / 2023

RESUME

Le VHDL est un langage de description matériel utilisé pour développer des circuits numériques en décrivant leur fonctionnement et leur architecture de manière rigoureuse. Il est utilisé pour simplifier la création de circuits, qu'ils soient simples ou complexes. Dans le contexte de la transmission série asynchrone, un circuit intéressant est présenté. Avant d'explorer ce circuit, il est important de définir la transmission série et ses différentes classes, ainsi que la norme RS-232. Ensuite, la structure interne de l'UART est expliquée, et les concepts fondamentaux nécessaires au VHDL sont rappelés. Enfin, les parties essentielles de l'UART, telles que l'émetteur, le récepteur et le générateur de vitesse, sont décrites en utilisant le VHDL.

ملخص

VHDL هي لغة وصف اٱلجهزة المستخدمة لتطوير الدوائر الرقمية عن طريق وصف عملياتها وبنيتها بدقة. يتم استخدامه لتبسيط إنشاء الدوائر سواء كانت بسيطة أو معقدة. في سياق الإرسال التسلسلي غير المتزامن ، يتم تقديم دائرة مثيرة للاهتمام. قبل استكشاف هذه الدائرة ، من المهم تحديد الإرسال التسلسلي وفئاته المختلفة ، بالإضافة إلى معيار.RS-232 بعد ذلك ، يتم شرح الهيكل الداخلي لـUART ، ويتم استدعاء المفاهيم اٱلساسية الا زمة لـVHDL .أخيرًا ، يتم وصف الأجزاء الأساسية منUART ، مثل المرسل والمستقبل ومولد السرعة ، باستخدام.VHDL

Dédicaces

A tous ceux qui m'ont appris une lettrel...

A celui qui a bu la coupe vide pour me donner une goutte d'amour et d'espoir A celui qui a récolté les épines de mon chemin pour paver le chemin de la connaissance "Le paradis de ma mère et le parfum de mon père et ses habitants sont mes frères et tous mes les proches "

Le voyage n'était pas court, et il ne devrait pas l'être. Le rêve n'était pas proche, et ce n'était pas facile, et ce n'est pas non plus difficile. Aujourd'hui, je t'écris, ô toi qui étais mon armée, qui m'as soutenu dans toutes mes marche et m'a consolé dans toutes mes blessures. Puisses-tu toujours être mon soutien et ma fierté.

REMERCIEMENTS

Louange à Dieu, assez, et prières soient sur l'élu bien-aimé et sa famille et ceux qui ont accompli:

Je ne peux pas dire merci, car on ne le dit qu'à la fin des événements, et je me vois toujours au début.

Cependant, pour être honnête, cette lettre porte mon nom, mais je n'aurais pas pu le faire sans le merveilleux soutien et le soutien constant de l'équipe médicale qui a surveillé chaque détail de moi. Je voudrais également exprimer ma gratitude aux excellents professeurs, M. Ballouti Adel et M. Mourad Brik, pour leur patience et leur soutien continus à mon égard. Mes remerciements particuliers à M. Mustafa Tabakh, chef du département électronique, pour son soutien moral.

Je m'excuse pour chaque lacune que j'ai commise, par Dieu, ce n'était qu'un oubli, et tous ceux dont le nom a été omis de mes lettres, mais c'est fermement établi dans mon mémorandum, vous tous, et vous ne répondez pas tous à mes remerciements, En effet, ce fut votre récompense, et votre effort est louable.

Sommaire

Sommaire

Liste des figures

Introduction Générale

La dernière décennie a vu une évolution technologique inéluctable dans tous les domaines, notamment dans le monde des systèmes de transmission numérique qui acheminent des informations entre une source et un destinataire à l'aide d'un support physique tel que le câble ou la fibre optique. Pour cela on peut considérer deux modes de transmission parallèle et série : en mode parallèle toutes les données sont envoyées en même temps sur autant de lignes de transmission, par contre en mode série les données sont envoyées les unes après les autres sur une seule ligne de transmission. Dans ce dernier mode, il existe deux grands types de liaison série : synchrone ou asynchrone d'où l'utilisation d'un seul fil pour l'émission et un autre fil pour la réception.

Un UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) est un circuit utilisé pour la communication série asynchrone entre un microcontrôleur ou un processeur et d'autres appareils.

Le travail présenté dans cette mémoire consiste à modéliser et de concevoir un circuit numérique à plusieurs niveaux d'abstraction qui permettra la transmission et la réception des données série de manière asynchrone (UART) en utilisant le langage VHDL, le logiciel Xilinx Vivado 2018.1 et la carte de développement Nexys Artix A7.

Problématique

Notre objectif est de concevoir la partie émettrice de l'UART, dont le but est de transmettre les données issu d'un capteur connecté à la carte FPGA vers un micro-ordinateur via le port série.

Chapitre I

La liaison série et la norme RS232

I.1 Introduction

La liaison série et la norme RS232 sont des concepts fondamentaux dans le domaine des communications électroniques. La liaison série est une méthode de transmission de données qui envoie les bits les uns après les autres sur un seul fil de données, tandis que la norme RS232 définit les spécifications électriques et les protocoles de communication pour les liaisons série.

La norme RS232 utilise généralement des niveaux de tension positifs et négatifs pour représenter les bits de données, où un niveau positif peut représenter un "1" logique et un niveau négatif peut représenter un "0" logique. Les signaux de contrôle RS232 comprennent des broches pour la transmission de données (Tx), la réception de données (Rx), les signaux de contrôle du flux de données (RTS et CTS), les signaux de contrôle de la transmission (DTR et DSR), etc.

La liaison série et la norme RS232, sont des concepts importants dans le domaine des communications électroniques. La liaison série permet la transmission de données bit par bit sur un fil unique, tandis que la norme RS232 définit les spécifications électriques et les protocoles de communication pour les liaisons série.

I.2 Port

Un port RS232 est généralement constitué d'un connecteur physique (tel qu'un connecteur DB9 ou DB25) et des broches correspondantes qui permettent la connexion d'un câble série. Ce câble est utilisé pour transférer les signaux de données, les signaux de contrôle et les signaux d'alimentation conformément à la norme RS232.

En utilisant le port RS232, les données peuvent être transmises en série entre les appareils connectés. Les signaux de données sont généralement transmis sur les broches Tx (transmission) et Rx (réception), tandis que les signaux de contrôle, tels que les signaux de contrôle du flux de données (RTS - Request to Send et CTS - Clear to Send) et les signaux de contrôle de la transmission (DTR - Data Terminal Ready et DSR - Data Set Ready), sont également transmis sur des broches spécifiques du port RS232. on parle communément du port RS232 ou du port parallèle. [1]

I.3 Types de transmission

On distingue deux types de transmission principale, la transmission parallèle et la transmission série.

I.3.1 Transmission parallèle

La transmission parallèle, comme son nom l'indique, est utilisée pour transférer des données entre différents composants électroniques tels que des processeurs, des mémoires et des périphériques de manière parallèle.

Dans ce type de transmission parallèle, chaque bit de données est transmis simultanément sur des lignes distinctes. Par exemple, dans une liaison parallèle à 8 bits, 8 lignes sont utilisées pour transférer les données, chaque ligne représente un bit (0 ou 1). Cette approche permet un transfert de données rapide, car tous les bits sont transmis en parallèle à chaque cycle d'horloge (figure I .1). [1]

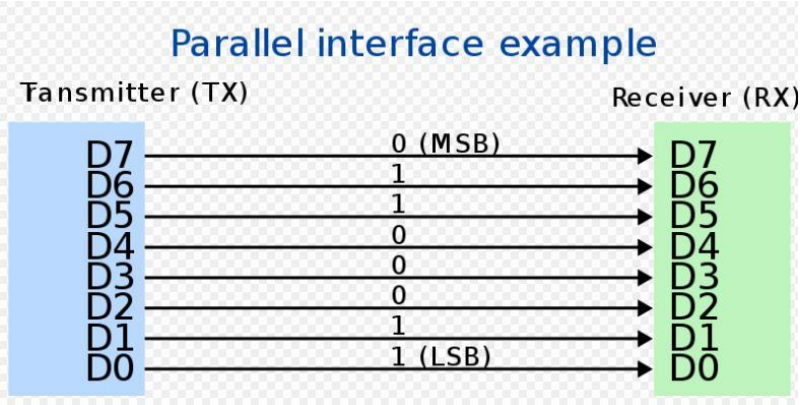

Figure.I.1. Exemple d'une liaison parallèle

I.3.2 La Transmission série

La transmission série est un mode de communication utilisé pour transférer des données de manière série entre un émetteur (transmetteur) et un récepteur. Dans la transmission série, les données sont envoyées bit par bit d'un ordinateur à l'autre en bi-direction. Chaque bit a son taux d'impulsions d'horloge. 8 bits sont transférés à la fois avec un bit de départ et d'arrêt (habituellement connu sous le nom de bit de parité), c'est-à-dire 0 et 1 respectivement. La transmission série est utilisée pour des transferts à longue distance, offrant une simplicité, une fiabilité et une flexibilité accrues par rapport à la liaison parallèle.

On distingue deux types de transmission : les transmissions asynchrones et les transmissions synchrones, suivant le mode de synchronisation de l'horloge du récepteur sur celle de l'émetteur.

I.3.2.1 Transmission synchrone

Dans ce type de transmission, l'émetteur et le récepteur doivent être synchronisés sur une horloge commune. L'horloge génère un signal régulier qui est utilisé pour diviser les données en trames ou en octets, et pour déterminer le moment précis de l'échantillonnage des bits. Lors de la transmission de données synchrones, l'émetteur envoie les données en respectant le rythme de l'horloge. Chaque bit est transmis pendant une période de temps définie, généralement à des intervalles réguliers. Le récepteur utilise la même horloge pour détecter et récupérer les bits à l'autre extrémité de la liaison. La synchronisation précise de l'horloge est essentielle pour garantir une communication correcte. Les écarts de temps ou les décalages entre l'horloge de l'émetteur et celle du récepteur peuvent entraîner des erreurs de transmission. Des mécanismes tels que des bits de synchronisation ou des signaux de synchronisation spéciaux peuvent être utilisés pour maintenir la synchronisation entre les deux extrémités de la liaison.

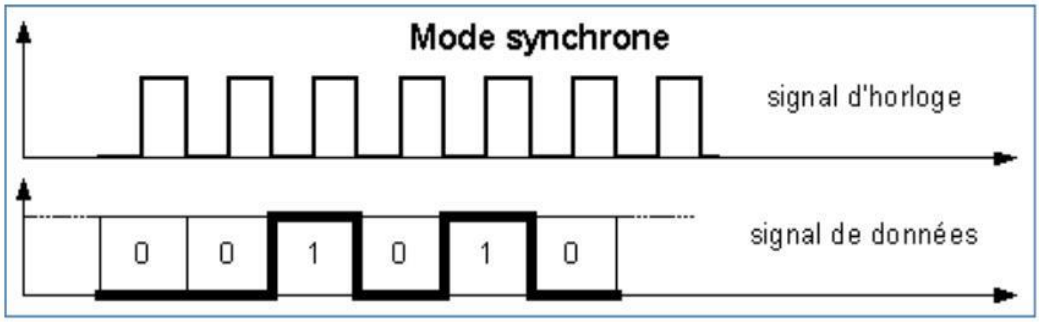

Figure.I.2 Exemple d'une liaison série synchrone.

I.3.2.2 Transmission asynchrones

Contrairement aux protocoles synchrones, la transmission asynchrone ne nécessite pas de synchronisation stricte entre l'émetteur et le récepteur. Les données sont transmises de manière indépendante avec des bits de départ et de fin pour la délimitation, offrant une flexibilité en termes de débit et de transmission. L'UART est un exemple courant de la transmission asynchrone.

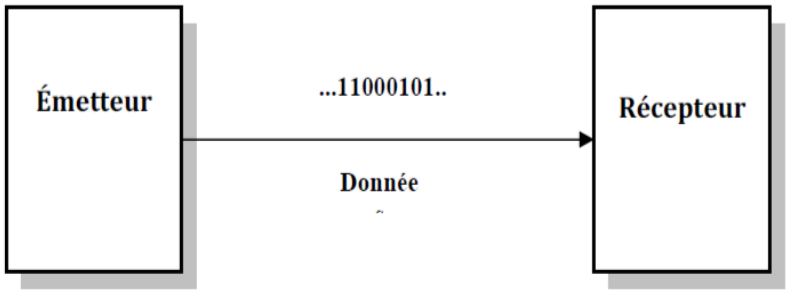

Figure.I.3. Exemple d'une liaison série asynchrone

Il est important de noter que la norme RS232 et l'UART sont souvent utilisés ensemble, où l'UART est utilisé pour la conversion des signaux parallèles en signaux série, tandis que la norme RS232 définit les spécifications électriques et les connexions physiques pour la communication série. Chaque mot « donnée » en transmission asynchrone, étant séparé, a besoin, pour être identifié par le récepteur, de contenir sa propre information de synchronisation. Ceci se fait en cadrant les bits d'information utile entre un bit dit de départ (START bit) et un ou deux bits d'arret (STOP bit).

Dans le domaine des micro-ordinateurs, le format d'une unité de données transmise est montré dans la figure I.4.

- Un bit START unique, de valeur 0 (norme universelle) ;
- Un mot d'information comportant 5,6 ou le plus souvent 7 bits (code ASCII) ;
- Un bit (facultatif) de parité, paire ou impaire ;
- Un ou deux bits STOP

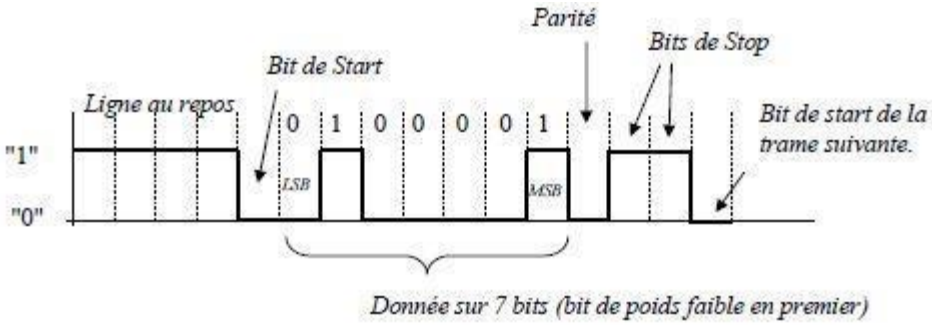

Figure.I.4. Transmission série asynchrone

I.3.3 La Parité

La parité est une technique utilisée dans les communications série pour la détection d'erreurs de transmission. Elle consiste à ajouter un bit supplémentaire, appelé bit de parité, à chaque octet de données transmis.

Le bit de parité est déterminé en fonction du nombre de bits de valeur 1 présents dans l'octet de données. Il existe deux types de parité :

1. Parité paire : Dans ce cas, le bit de parité est fixé de manière à ce que le nombre total de bits de valeur 1, y compris le bit de parité, soit un nombre pair. Par exemple, si l'octet de données contient un nombre impair de bits de valeur 1, le bit de parité sera réglé sur 1 pour obtenir un nombre pair total de bits de valeur 1.

2. Parité impaire : Ici, le bit de parité est fixé de manière à ce que le nombre total de bits de valeur 1, y compris le bit de parité, soit un nombre impair. Si l'octet de données contient un nombre pair de bits de valeur 1, le bit de parité sera réglé sur 1 pour obtenir un nombre impair total de bits de valeur 1.

La parité est une méthode simple de détection d'erreurs, mais elle ne permet pas de corriger les erreurs. Elle ne peut identifier que la présence d'erreurs, mais ne fournit pas d'informations sur les bits spécifiques qui sont erronés. Si une erreur de parité est détectée, il est généralement nécessaire de renvoyer les données pour une nouvelle transmission.

I.4 Bus ou liaison

L'une des deux natures d'interconnexions est la liaison, qui peut être réalisée selon deux types d'architectures.

- Le premier type correspond à une interconnexion entre deux équipements, indépendamment de leur nature (voir la figure I.5). Il peut s'agir d'une liaison entre émetteurs, récepteurs ou émetteurs/récepteurs. [1]

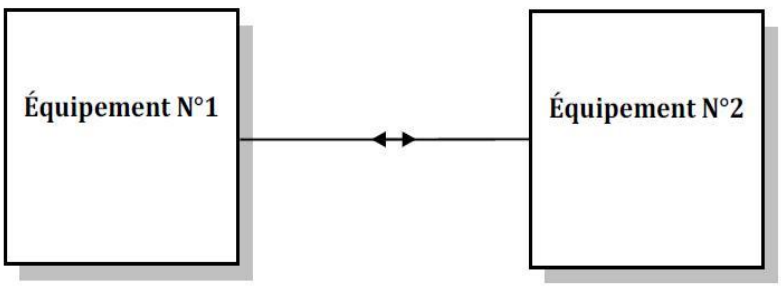

Figure.I.5. Une liaison

- Le deuxième type d'architecture est représenté par le bus illustré dans la figure I.6, car sa structure matérielle permet l'interconnexion de plusieurs équipements sur un même support physique. Cette approche architecturale est significativement plus complexe que la simple capacité d'interface bus des équipements à interconnecter. En effet, elle nécessite la mise en place d'un protocole à la fois au niveau matériel et logiciel pour gérer les communications entre les différents équipements.

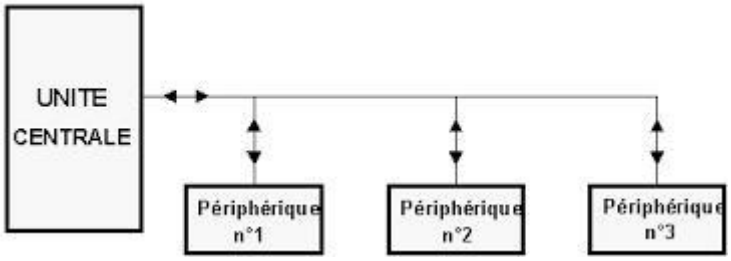

Figure.I.6. Exemple d'un bus.

I.5 Types ou modes de liaison

En télécommunications, un canal de communication peut être :

Simplex : l'information est transportée dans un seul sens (unidirectionnel)

Duplex : l'information est transportée dans les deux sens (bidirectionnel) Selon que l'information peut être transportée simultanément dans les deux sens ou non. On parle respectivement de canal **full-duplex** ou **half-duplex**.

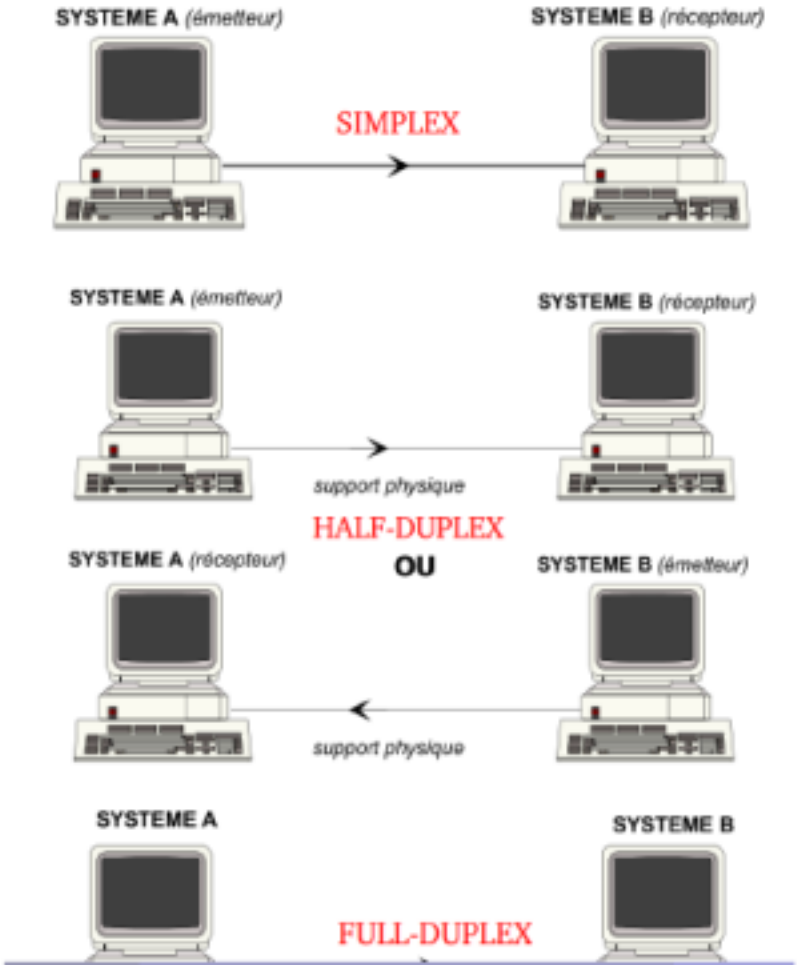

Figure. I.7. Différents modes de transmission.

I.6 Descriptions techniques de la norme RS232

I.6.1 Description électrique des signaux

Nous allons nous pencher sur le niveau de tension présent sur la liaison afin de mieux comprendre son fonctionnement [1].

La norme RS232, est un protocole de communication série largement utilisé pour la transmission de données entre des dispositifs électroniques. Elle définit les caractéristiques électriques des signaux utilisés dans la communication série.

Voici les caractéristiques électriques typiques des signaux RS232 :

- **Tension logique :** Dans la norme RS232, les niveaux de tension logique sont définis comme étant négatifs (marqués comme "1") et positifs (marqués comme "0"). Une tension positive (généralement entre +5V et +15V) représente un "0 logique", tandis qu'une tension négative (généralement entre -5V et -15V) représente un "1 logique".

- **Signaux de données :** Les signaux de données RS232 comprennent la ligne de transmission de données (Txd pour l'émetteur et Rxd pour le récepteur) qui permet le transfert des données binaires d'un dispositif à un autre.
- **Bits de données :** Les données sont transmises en utilisant un format série asynchrone où chaque caractère est constitué de bits de données, où le nombre de bits de données peut être 5, 6, 7 ou 8 bits.
- **Bits de contrôle :** En plus des bits de données, RS232 peut également inclure des bits de contrôle tels que le bit de parité (utilisé pour la vérification d'erreur) et les bits de démarrage/arrêt (pour synchroniser la transmission).
- **Connexions :** RS232 utilise généralement des connecteurs DB-9 ou DB-25 pour les connexions physiques entre les dispositifs. Les broches du connecteur sont attribuées à différents signaux, y compris les signaux de données, les signaux de contrôle, les signaux de terre, etc.

Il est important de noter que les niveaux de tension spécifiques, les configurations des signaux et les connecteurs peuvent varier en fonction de la mise en œuvre spécifique de RS232. Par conséquent, il est essentiel de se référer aux spécifications précises des appareils et des câbles RS232 utilisés pour assurer la compatibilité et le bon fonctionnement de la communication. La norme RS232 est fréquemment utilisée pour la communication série entre les ordinateurs et les périphériques tels que les imprimantes, les modems, les scanners de codes-barres, etc. Elle offre une méthode simple et fiable de transmission de données série sur de courtes distances.

I.6.2 La trame RS232

La trame RS232 fait référence à la structure des données transmises dans une communication série conformément à la norme RS232. Elle est composée de plusieurs éléments qui assurent l'organisation et la transmission fiable des données entre les appareils. Les principaux éléments d'une trame RS232 typique sont les suivants :

Bits de démarrage : La trame RS232 débute généralement par un ou plusieurs bits de démarrage. Ces bits signalent le début de la trame et permettent au récepteur de synchroniser la réception des données. Le nombre de bits de démarrage est habituellement d'un ou deux.

Bits de données : Les bits de données contiennent les informations à transmettre. Le nombre de bits de données dans une trame RS232 peut varier (généralement entre 5 et

8 bits) en fonction de la configuration spécifique. Chaque bit de données représente un élément d'information binaire.

Bit de parité : Le bit de parité est un élément facultatif de la trame RS232 utilisé pour la vérification d'erreur. Il permet de détecter les erreurs de transmission en ajoutant un bit de parité supplémentaire à la trame. Le bit de parité peut être pair, impair ou inexistant (pas de bit de parité).

Bits d'arrêt : Les bits d'arrêt indiquent la fin de la trame RS232. Ils signalent au récepteur que la transmission des données est terminée. Le nombre de bits d'arrêt est généralement d'un ou deux.

La structure d'une trame RS232 peut varier en fonction de la configuration spécifique et des exigences de la communication. Par exemple, des bits de contrôle supplémentaires peuvent être inclus dans une trame RS232 pour des fonctionnalités spécifiques, tels que les signaux de contrôle de flux utilisés pour réguler le flux de données entre les appareils.

Il est crucial de noter que la structure de la trame RS232 doit être identique à la fois du côté de l'émetteur et du récepteur afin d'assurer une communication correcte. Les appareils et les logiciels utilisés pour la communication RS232 doivent être configurés en conséquence, en respectant la structure de la trame et les paramètres de transmission appropriés.

La figure I.8 illustre la structure de la trame RS232 définie par la norme.

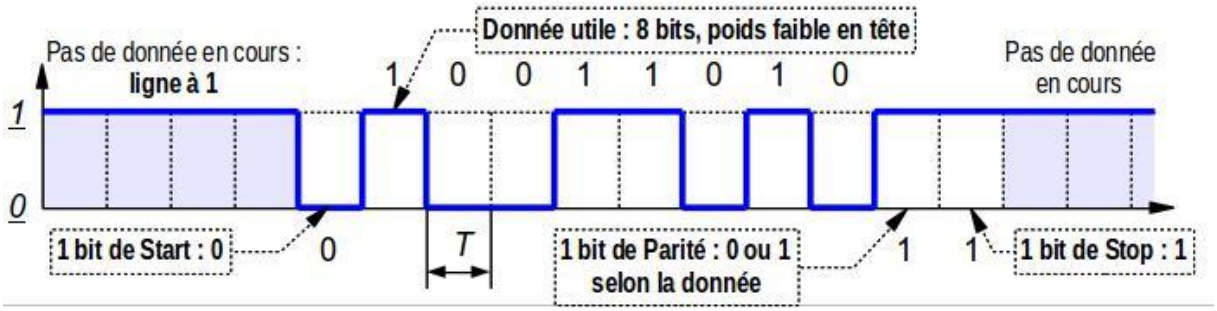

Figure. I.8. Chronogramme d'une trame de données

I.6.3 Fonction Des Signaux

Les signaux utilisés dans une communication RS232 remplissent des fonctions spécifiques qui déterminent leur rôle dans la transmission des données. Voici les fonctions courantes des signaux RS232 :

- ✓ **Signal de transmission de données (TXD) :** Ce signal est utilisé par l'émetteur pour transmettre les données binaires vers le récepteur. Il envoie des signaux électriques correspondant aux bits de données de la trame RS232.
- ✓ **Signal de réception de données (RXD) :** Ce signal est utilisé par le récepteur pour recevoir les données binaires provenant de l'émetteur. Il analyse les signaux électriques reçus pour reconstituer les bits de données de la trame RS232.
- ✓ **Signal de contrôle de flux (RTS/CTS) :** Les signaux de contrôle de flux, tels que Ready To Send (RTS) et Clear To Send (CTS), servent à réguler le flux de données entre l'émetteur et le récepteur. L'émetteur utilise le signal RTS pour indiquer qu'il est prêt à envoyer des données, tandis que le récepteur utilise le signal CTS pour autoriser ou bloquer la transmission des données.
- ✓ **Signal de demande de ligne de données (DSR) :** Le signal de demande de ligne de données (DSR) indique que le récepteur est prêt à recevoir des données. Lorsque le récepteur est prêt, il envoie un signal DSR à l'émetteur pour l'informer qu'il est en état de recevoir des données.
- ✓ **Signal de signalisation de port série (DTR/DSR) :** Les signaux de signalisation de port série, tels que Data Terminal Ready (DTR) et Data Set Ready (DSR), servent à indiquer l'état de disponibilité des équipements de communication. L'émetteur utilise le signal DTR pour signaler que le dispositif est prêt à établir une communication, tandis que le récepteur utilise le signal DSR pour indiquer qu'il est prêt à recevoir des données.
- ✓ **Signaux de commande et de configuration (RI, DCD) :** Les signaux Ring Indicator (RI) et Data Carrier Detect (DCD) servent à indiquer différents événements ou conditions liés à la communication. Le signal RI peut être utilisé pour indiquer un appel entrant ou une notification spéciale, tandis que le signal DCD peut être utilisé pour indiquer que la connexion avec l'autre appareil est établie.

Ces signaux jouent un rôle essentiel dans la communication RS232 en garantissant la transmission correcte des données et en fournissant des mécanismes de contrôle et de régulation du flux. La gestion adéquate de ces signaux est cruciale pour une communication fiable et efficace entre les appareils utilisant la norme RS232.

I.7 Conclusion

Dans le contexte de la liaison asynchrone, la communication est réalisée en utilisant un circuit UART qui assure la conversion entre les données série et parallèles. Le circuit UART gère la synchronisation des bits de données en utilisant des mécanismes d'horloge appropriés et implémente également des protocoles de communication asynchrones pour permettre une transmission fiable des données.

Il convient de noter que la liaison asynchrone et l'utilisation du circuit UART en VHDL sont des techniques couramment utilisées dans les applications de communication série, offrant une solution fiable et flexible pour la transmission de données sur de longues distances.

Chapitre II

UART (Universal Asynchronous Receiver Transmitter)

II.1 Introduction

L'UART (Universal Asynchronous Receiver Transmitter) est une interface matérielle de communication largement utilisée qui facilite la transmission série de données entre un microcontrôleur ou un microprocesseur et d'autres dispositifs. Il offre une méthode fiable et simple pour l'envoi et la réception de données de manière asynchrone, c'est-à-dire sans recours à un signal d'horloge synchronisé.

La communication via UART est caractérisée par l'utilisation de bits de démarrage et d'arrêt en combinaison avec les bits de données. Le bit de démarrage est un signal de niveau bas (0 logique) qui marque le début d'une trame de données, tandis que le bit d'arrêt est un signal de niveau haut (1 logique) qui indique la fin de la trame. Les bits de données peuvent varier en longueur et sont généralement composés de 7 ou 8 bits, représentant les données réelles transmises.

II.2 Le principe d'une liaison sérié

Pour effectuer la transmission série, une conversion préalable des données parallèles en données sérielles est nécessaire. Cette conversion est généralement réalisée à l'aide d'un registre à décalage, tel que montré dans la Figure II.1. Ce registre à décalage permet de convertir les données parallèles en une séquence de bits sériels. À l'autre extrémité de la liaison, un autre registre à décalage effectue la conversion inverse, convertissant les données sérielles en mots parallèles. Ce registre à décalage joue le rôle de conversion série-parallèle.

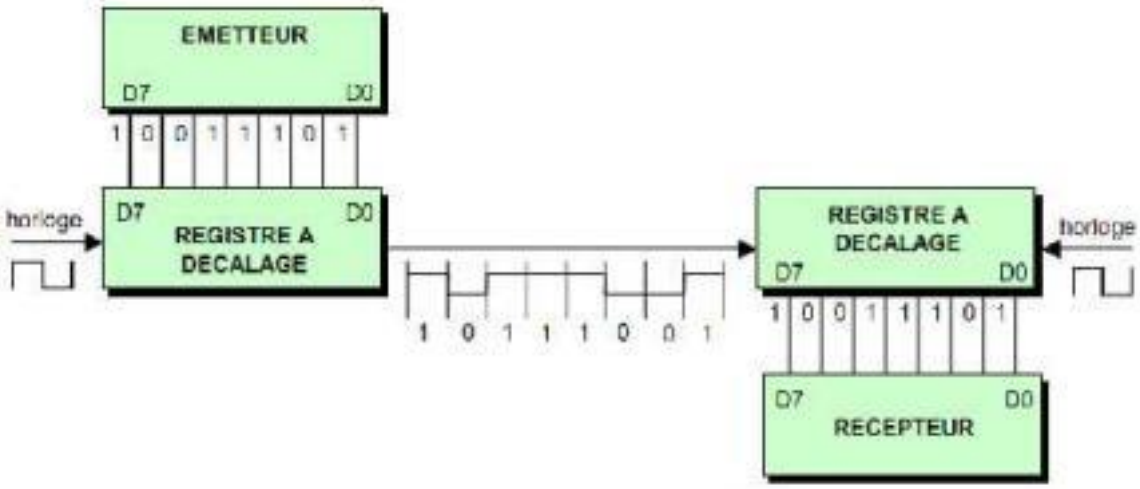

Figure.II.1. Principe d'une liaison série

II.3 L'interface série

Cette interface se compose généralement de trois composants essentiels : l'émetteur, le récepteur et le support de communication. L'émetteur joue le rôle de convertir les données en une séquence de bits sériels et de les transmettre sur la ligne de communication. De son côté, le récepteur reçoit les bits sériels et les convertit en données compréhensibles par le système. Le support de communication peut prendre différentes formes telles qu'un câble, une fibre optique ou tout autre moyen physique permettant le transfert des signaux.

De plus, cette interface permet d'atteindre des vitesses de transmission élevées, en minimisant les risques d'interférences électromagnétiques sur de longues distances. En raison de ces avantages, l'interface série est couramment choisie pour de nombreuses applications de communication.

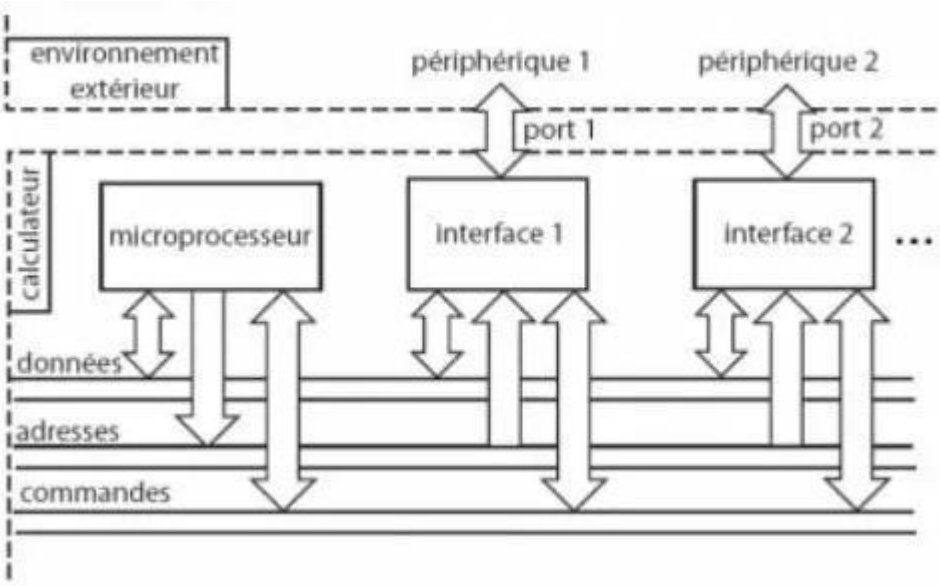

Figure.II.2. Principe de l'interface série.

II.4 Structure de l'UART

Le schéma-bloc de l'UART est représenté à la Figure II.3. Il comprend généralement plusieurs composants essentiels qui sont les suivants :

1. **Générateur d'horloge :** Il s'agit d'une composante qui génère un signal d'horloge utilisé pour synchroniser la transmission et la réception des données. Ce signal est crucial pour assurer la cohérence et la précision des opérations de l'UART.
2. **Canal d'émission (TRANSMITTER) :** Cette partie de l'UART est responsable de la conversion des données en une séquence de bits sériels à transmettre. Elle prépare les données pour qu'elles soient envoyées sur la ligne de communication dans le format approprié.
3. **Canal de réception (RECEIVER) :** Cette partie de l'UART reçoit les bits sériels provenant de la ligne de communication et les convertit en données reconnaissables par le système. Elle assure la détection et la correction d'éventuelles erreurs de transmission.
4. **Autres sous-fonctions d'UART :** En plus des composants mentionnés ci-dessus, l'UART peut également comporter d'autres sous-fonctions spécifiques en fonction des exigences du système. Cela peut inclure des fonctions telles que le contrôle de flux, la gestion des erreurs, la synchronisation des données, etc.

Ces composants fondamentaux de l'UART travaillent ensemble pour faciliter la transmission et la réception des données série de manière fiable et efficace. Ils sont essentiels pour permettre la communication entre un microcontrôleur ou un microprocesseur et d'autres périphériques ou systèmes.

II.4.1 Un générateur d'horloge

La fréquence de l'horloge de l'UART est généralement déterminée par les besoins spécifiques de communication du système. Elle est communément exprimée en bauds (bits par seconde) et doit être compatible avec la vitesse de transmission des données configurée pour l'UART. Le générateur d'horloge de l'UART est chargé de produire le signal d'horloge qui synchronise précisément les opérations de transmission et de réception des données. Son implémentation peut varier en fonction des spécifications et des exigences du système, incluant l'utilisation d'un oscillateur interne, d'une horloge externe ou d'un oscillateur à quartz. La configuration appropriée du générateur d'horloge revêt une importance cruciale pour garantir une communication fiable via l'UART, en assurant la synchronisation précise des transmissions de données.

$V_{Baud} = f_{UART}(Hz) / 16* X$

V_{Baud} : représente le débit en bauds.

$f_{UART}(Hz)$: représente la fréquence UART.

X : le nombre d'échantillons par bit.

II.4.2 Un canal d'émission (TRANSMITTER)

Le canal d'émission, également appelé émetteur, joue un rôle crucial dans la communication série en assurant la conversion, l'encodage, le formatage et la transmission des données sur la liaison série. Il est composé de plusieurs éléments qui travaillent de concert pour garantir une transmission efficace des données. [3]

Au cœur du canal d'émission se trouve le registre de conversion parallèle-série (P-S), qui se charge de convertir les données parallèles provenant du système ou du microprocesseur en une séquence binaire sérielle. Ce registre joue un rôle essentiel en transformant les données en bits sériels pour une transmission appropriée sur la ligne de communication.

Un autre composant clé est la logique de mise en forme, responsable de l'insertion de bits de cadrage tels que la parité, le bit de départ (START) et le bit de fin (STOP). Ces bits de cadrage sont essentiels pour structurer et délimiter les données transmises, permettant ainsi au récepteur de les synchroniser et de les interpréter correctement. [3]

Le canal d'émission est connecté en entrée au bus interne, ce qui le rend compatible avec le microprocesseur ou le microcontrôleur. En sortie, il est relié à la ligne de transmission série TxD (Transmit Data), permettant la transmission des données vers le récepteur.
La fréquence de transmission est contrôlée par l'horloge de transmission TxC (Transmitter Clock), dont la transition descendante sert de référence pour synchroniser lesopérations de transmission. La logique de contrôle de transmission joue un rôle essentiel en activant le signal TxEN (send enable) pour autoriser les opérations d'émission. De plus, l'activation du signal CTS (clear to send) est nécessaire pour initier un transfert.

La logique de contrôle gère également deux signaux d'état : TxE (transmetteur vide, Null Transmitter) et TxRDY (Transmitter ReaDY), qui fournissent des informations sur l'état du canal d'émission. Par exemple, la broche TxRDY indique si le tampon de sortie est vide, et cela est déterminé par l'expression logique TxRDY = (Tampon vide) ET (CTS=0) ET (TxEN=1).

II.4.3 Un canal de réception (RECEIVER)

Le canal de réception, également connu sous le nom de récepteur, est un composant essentiel de la communication série. Il est composé de plusieurs éléments qui travaillent en harmonie pour effectuer la conversion, le décodage, la mise en forme et la réception des données sur la liaison série. [3] Ces éléments incluent un registre de conversion série-parallèle (S-P), chargé de convertir les données binaires sérielles en données parallèles, ainsi qu'une logique de mise en forme de caractère qui supprime les bits de cadrage tels que la parité, le bit de départ (START) et le bit de fin (STOP) pour extraire le caractère réel transmis.

Le canal de réception est contrôlé par une logique de commande qui exécute plusieurs fonctions essentielles. Tout d'abord, elle gère un commutateur d'autorisation de réception RxEN (Receive Enable), qui est accessible via un programme et permet d'autoriser la réception des données. Lorsque RxEN=1, la réception est autorisée.

Ensuite, la logique de commande gère une bascule d'état qui interprète le niveau bas de la ligne RxD. Ce niveau bas peut correspondre soit à une ligne inutilisée, soit à une condition de rupture (BREAK) entre les caractères transmis. [3] La bascule est associée à la détection de rupture (BRKDET) et passe à l'état "1" lorsque la ligne RxD reste basse pendant deux cycles de caractères complets. Une fois que RxD revient à l'état haut, BRKDET est réinitialisé à "0".

Le canal de réception comprend également un déclencheur qui indique si un état "1" valide a été détecté sur la ligne d'entrée RxD après une réinitialisation générale. Tant que cette condition n'est pas remplie, aucune donnée ne sera acceptée de RxD. Une seule occurrence de cet état valide après la réinitialisation est suffisante pour autoriser la réception de données.

La logique de commande du canal de réception effectue également la vérification du bit de parité en calculant sa valeur lors de la réception des bits de données et en la comparant au bit de parité reçu. En cas de divergence, un bit d'erreur de parité (parity error) est défini à "1". De plus, elle vérifie la présence d'un bit de fin (STOP) à la fin du caractère reçu. Si ce bit de fin est absent, un bit d'erreur de trame (frame error) est défini à "1". Il est important de noterque le canal de réception vérifie uniquement un seul bit de fin, même si l'UART a été conçu pour prendre en charge 1,5 ou 2 bits de fin.

Enfin, la logique de commande émet une bascule d'état RxRDY (Receiver Ready), qui indique que tout un caractère est sur le point d'être reçu. Ce signal peut être utilisé pour envoyer une demande d'interruption au processeur afin de traiter le caractère reçu. De plus, elle détecte si un nouveau caractère a été reçu avant que le précédent n'ait été lu par le microprocesseur en déclenchant un symbole d'erreur de dépassement (overrun

II.5.Autres sous-fonction de l'UART

En complément des canaux d'émission et de réception (TRANSMITTER et RECEIVER), l'UART (Universal Asynchronous Receiver Transmitter) peut inclure diverses fonctionnalités pour garantir un fonctionnement complet et efficace de la communication série. Parmi ces fonctionnalités figurent notamment :

- **Générateur d'horloge :** L'UART peut intégrer un générateur d'horloge interne qui fournit le signal d'horloge essentiel pour synchroniser la transmission et la réception des données. Ce générateur garantit un échantillonnage et un traitement précis des bits de données. [3]

- **Mémoire tampon de données :** Une mémoire tampon de données peut être présente dans l'UART afin de stocker temporairement les données reçues ou celles à transmettre. Cette mémoire tampon régule le flux de données entre l'émetteur et le récepteur, évitant ainsi toute perte de données ou saturation.

- **Contrôle de flux matériel :** Certaines implémentations d'UART prennent en charge le contrôle de flux matériel. Cette fonctionnalité repose sur l'utilisation de signaux de contrôle supplémentaires tels que RTS (Ready To Send) et CTS (Clear To Send) pour réguler le flux de données entre les périphériques connectés. Le contrôle de flux matériel prévient les pertes de données lorsque la vitesse de transmission diffère entre l'émetteur et le récepteur. [3]

- **Interruptions :** L'UART peut être configuré pour générer des interruptions lors de certains événements spécifiques tels que la réception de données ou la transmission de données complète. Ces interruptions permettent au processeur de traiter efficacement les événements en temps réel sans avoir à surveiller en permanence l'état de l'UART.

- **Configuration des paramètres de communication :** L'UART offre souvent des mécanismes de configuration permettant de définir les paramètres de communication tels que le débit de transmission, le nombre de bits de données, la parité, le nombre de
- **Configuration des paramètres de communication :** L'UART offre souvent des mécanismes de configuration permettant de définir les paramètres de communication tels que le débit de transmission, le nombre de bits de données, la parité, le nombre de bits d'arrêt, etc. Ces paramètres peuvent être ajustés en fonction des besoins spécifiques du système et de la communication.
- **Mécanismes de contrôle d'erreur :** Certains modèles d'UART prennent en charge des mécanismes de détection et de correction d'erreurs tels que le contrôle de redondance cyclique (CRC) ou d'autres méthodes d'intégrité des données. Ces mécanismes permettent de vérifier et garantir l'intégrité des données transmises.

l'UART peut intégrer différentes fonctionnalités telles que le générateur d'horloge, la mémoire tampon de données, le contrôle de flux matériel, les interruptions, la configuration des paramètres de communication et les mécanismes de contrôle d'erreur. Ces fonctionnalités contribuent à assurer une communication série fiable et efficace entre les périphériques.

II.6 Conclusion

Après avoir effectué une analyse détaillée de l'architecture interne de l'UART et décrit le fonctionnement de chaque composant, ainsi que les différents registres utilisés, nous allons maintenant nous concentrer sur l'explication du langage VHDL utilisé pour décrire chaque composant individuel. L'objectif est de combiner ces descriptions de composants pour créer notre propre UART, en assurant une conception cohérente et fiable. Le langage VHDL, largement utilisé pour la conception de circuits numériques, nous permettra de décrire le comportement, la structure et les interconnexions des différents blocs de l'UART, en fournissant une base solide pour la mise en œuvre pratique de notre propre système de communication série.

Chapitre III

Langage VHDL et circuits FPGAs

III.1 Introduction

Le langage VHDL (VHSIC Hardware Description Language) est largement utilisé dans le domaine de l'électronique pour la preuve formelle de l'équivalence du circuit, la simulation et la spécification du fonctionnement des systèmes matériels. Au fil du temps, il a également trouvé une application majeure dans le domaine de la synthèse automatique des circuits. L'acronyme VHDL signifie VHSIC Hardware Description Language, où VHSIC fait référence Very High Speed Integrated Circuit, soulignant ainsi sa pertinence pour la description matérielle des circuits intégrés à très haute vitesse. Le VHDL offre une approche structurée et formelle pour décrire le comportement et la structure des systèmes matériels, permettant une conception plus précise, une vérification approfondie et une optimisation efficace des circuits. En utilisant le VHDL, les concepteurs peuvent modéliser et simuler le fonctionnement des circuits électroniques, faciliter la preuve de leur équivalence, ainsi que générer automatiquement des circuits équivalents à partir de descriptions de haut niveau. Ainsi, le VHDL joue un rôle essentiel dans le développement et l'implémentation de systèmes électroniques complexes et de haute performance. [4]

III.2 Qu’est ce que le VHDL

III.2.1 Historique

Le langage VHDL (VHSIC Hardware Description Language) a été conçu dans les années 1980 sous l'impulsion du projet VHSIC (Very High-Speed Integrated Circuit), lancé par le Département de la Défense des États-Unis. L'objectif principal de VHDL était de fournir une méthode normalisée et rigoureuse pour la description et la simulation des circuits numériques à haute vitesse. [5]

En 1987, le standard IEEE 1076 a été publié, établissant la première version officielle du langage VHDL. Depuis lors, plusieurs révisions et améliorations ont été apportées, donnant lieu

à des versions ultérieures telles que VHDL-93, VHDL-2002, VHDL-2008 et VHDL-2019, qui ont enrichi les fonctionnalités et la flexibilité du langage.

Le VHDL a connu une adoption rapide dans l'industrie des semi-conducteurs, devenant le langage de choix pour la conception de circuits numériques complexes. Il permet de décrire de manière abstraite le comportement et la structure d'un circuit, indépendamment des technologies matérielles sous-jacentes. Ainsi, il est utilisé pour la conception de divers systèmes numériques tels que des processeurs, des ASIC (Application-Specific Integrated Circuit) et des FPGA (Field-Programmable Gate Array). [5]

En plus de son utilisation industrielle, le VHDL est largement utilisé dans le milieu académique pour l'enseignement et la recherche en conception de circuits intégrés. Son haut niveau d'abstraction et sa capacité à effectuer des vérifications formelles font de VHDL un outil puissant pour la conception et la simulation de systèmes numériques.

Au fil des années, VHDL a évolué pour répondre aux besoins croissants des concepteurs et pour prendre en charge de nouvelles fonctionnalités. Il reste l'un des langages de description matériel les plus répandus et est étroitement intégré aux outils de conception électronique utilisés aujourd'hui.

Le langage VHDL a été développé dans le cadre du projet VHSIC et est devenu un élément essentiel de la description et de la conception des circuits numériques. Son histoire riche et son adoption généralisée en font un langage de référence dans le domaine de la conception électronique.

III.2.2 Pourquoi utiliser le VHDL

Le langage VHDL (VHSIC Hardware Description Language) est largement adopté dans l'industrie de la conception de circuits numériques en raison de plusieurs avantages significatifs :

- ✓ **Description formelle du matériel :** VHDL permet une description précise et rigoureuse du fonctionnement et de l'architecture des circuits numériques. Il fournit un langage structuré qui permet de spécifier de manière détaillée les comportements, les interconnexions et les contraintes de synchronisation des composants matériels.
- ✓ **Abstraction des niveaux de conception :** VHDL permet de décrire un circuit à différents niveaux d'abstraction, allant du niveau comportemental au niveau structurel. Cette approche facilite la conception hiérarchique et modulaire des systèmes numériques, permettant aux concepteurs de gérer efficacement la complexité et de réutiliser des blocs de conception.
- ✓ **Simulation et vérification :** VHDL est utilisé pour simuler les circuits numériques avant leur implémentation physique. Les concepteurs peuvent créer des bancs d'essai (testbenches) pour vérifier le fonctionnement du circuit et effectuer des tests fonctionnels et de performance. La simulation permet de détecter et de corriger les erreurs de conception avant la fabrication physique.
- ✓ **Synthèse logique :** VHDL est utilisé pour générer automatiquement la description logique d'un circuit à partir d'une description de haut niveau. Cette synthèse logique

permet de convertir le code VHDL en une netlist, qui peut ensuite être utilisée pour la conception et la programmation de circuits programmables, tels que les FPGA (Field-Programmable Gate Array).

- ✓ **Vérification formelle :** VHDL offre la possibilité d'effectuer des vérifications formelles sur la description du matériel. Cela permet de garantir que les propriétés spécifiées sont respectées, ce qui contribue à assurer une fiabilité accrue du circuit et à réduire les risques d'erreurs.

- ✓ **Portabilité :** VHDL est un langage normalisé par l'IEEE, ce qui signifie qu'il est indépendant des technologies matérielles spécifiques. Les conceptions matérielles écrites en VHDL peuvent être facilement portées sur différents types de circuits intégrés et de plates-formes de développement, offrant ainsi une plus grande flexibilité et une plus grande compatibilité.

- ✓ **Support des outils de conception :** Le langage VHDL bénéficie d'un large éventail d'outils de conception électronique, tels que les simulateurs, les synthétiseurs, les vérificateurs de propriétés et les outils de vérification formelle. Cette prise en charge étendue facilite l'intégration du code VHDL dans les flux de conception existants et contribue à améliorer l'efficacité et la productivité des concepteurs.

L'utilisation du langage VHDL présente de nombreux avantages pour la conception de circuits numériques, notamment une description formelle et précise, une modélisation hiérarchique, une simulation précise, une vérification formelle, une synthèse logique et une portabilité. Il demeure l'un des langages les plus répandus et les plus utilisés dans l'industrie pour la conception de systèmes numériques complexes.

III.3 La Syntaxe Du Langage VHDL:

Pour décrire un circuit en VHDL, il est nécessaire de spécifier trois parties essentielles : la librairie, l'entité et l'architecture.

La déclaration des librairies regroupe les fonctions et les types nécessaires pour former le modèle du circuit. Cette section permet d'inclure des bibliothèques externes ou des modules spécifiques qui seront utilisés dans la description du circuit. [6]

La déclaration de l'entité représente l'interface du modèle. Elle définit les entrées et les sorties du circuit, ainsi que d'autres composants qui peuvent être inclus. Chaque composant est défini de la même manière que l'entité principale du circuit, avec ses propres entrées, sorties et signaux internes. [6]

L'architecture de l'entité décrit le comportement du circuit. Elle spécifie comment les entrées sont traitées et comment les sorties sont générées en fonction des signaux internes et des opérations effectuées. L'architecture peut être conçue de différentes manières, telles que des descriptions comportementales ou structurelles, en utilisant des processus, des équations logiques, des opérations arithmétiques, etc.

En respectant ces trois parties, la description VHDL d'un circuit peut être réalisée de manière comportementale ou structurelle ou les deux manières en même temps. Cela facilite la compréhension, la maintenance et la réutilisation du code, ainsi que la simulation et la vérification du circuit.

III.3.1 Les Librairie

Les librairies VHDL offrent la possibilité d'utiliser des types, des fonctions et des constantes définis dans des packages spécifiques sans avoir à spécifier la librairie à chaque utilisation. Cela facilite l'utilisation des éléments prédéfinis et améliore la lisibilité du code en évitant des répétitions inutiles. [7]

Pour utiliser une librairie, vous devez d'abord la déclarer dans votre code VHDL en utilisant la directive `library`. Par exemple : [6]

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
```

Figure. III.1.exemple de déclaration de Library

III.3.2 La Déclaration D'entité

En VHDL, la déclaration d'entité revêt une importance capitale dans la conception de composants matériels, car elle permet de définir l'interface de ces derniers. L'entité spécifie les ports d'entrée et de sortie du composant, ainsi que les signaux internes qui seront utilisés dans les différentes architectures du composant.

La syntaxe générale de la déclaration d'entité est la suivante :

```
Entity Nom_Entite is
port (  -- Déclaration des ports d'entrée et de sortie
            Nom_Port_1 : mode type;
            Nom_Port_2 : mode type;
        -- ... Nom_Port_n : mode type  );
End entity Nom_Entite;
```

Figure. III.2. Exemple de déclaration d'entité

III.3.3 Déclaration de l'architecture

La déclaration de l'architecture en VHDL est utilisée pour décrire le comportement interne d'une entité. Elle spécifie comment les signaux et les composants sont interconnectés et comment les opérations sont effectuées à l'intérieur de cette entité.

L'architecture est définie en utilisant le mot-clé "Architecture", suivi du nom de l'architecture. [6]

```
Architecture Nom_de_l'architecture OF nom de l'entité IS
        Zone de déclaration (facultative ou optionnelle)
Begin
        Description de la structure logique.
End Nom_de_l'architecture;
```

Figure. III .3 Exemple Déclaration de l'architecture.

III.4 Les circuits FPGAs

Les FPGAs (Field Programmable Gate Arrays) sont des dispositifs à semi-conducteurs basés sur une matrice de blocs logiques configurables (CLB) connectés via des interconnexions programmables. Les FPGA peuvent être reprogrammés selon les exigences d'application ou de fonctionnalité souhaitées après la fabrication. Cette caractéristique distingue les FPGA des circuits intégrés spécifiques à l'application (ASIC), qui sont fabriqués sur mesure pour des tâches de conception spécifiques. Bien que des FPGA programmables une seule fois (OTP) soient disponibles, les types dominants sont basés sur la SRAM qui peut être reprogrammée à mesure que la conception évolue.

III.4.1. Architecture Des FPGAs

Les circuits FPGA sont constitués d'une matrice de blocs logiques (CLB : Configurable Logic Block) programmables entourés de blocs d'entrée sortie (IOB) programmable. L'ensemble est relié par un réseau d'interconnexions programmable.

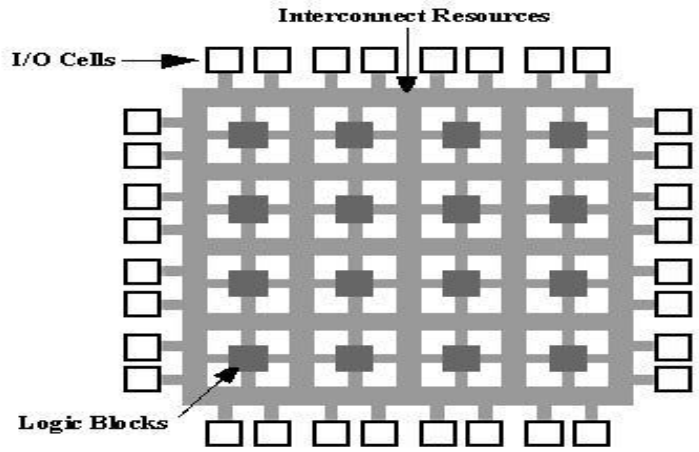

Figure. III.4. Schéma block d'un FPGA

Les FPGA sont bien distincts des autres familles de circuits programmables tout en offrant le plus haut niveau d'intégration logique.

L'utilisateur peut programmer la fonction réalisée par chaque cellule (appelée CLB par Xilinx: Configurable Logic Block):

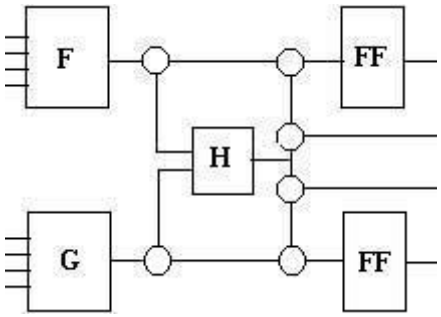

Figure. III.5. Schéma bloc d'une cellule

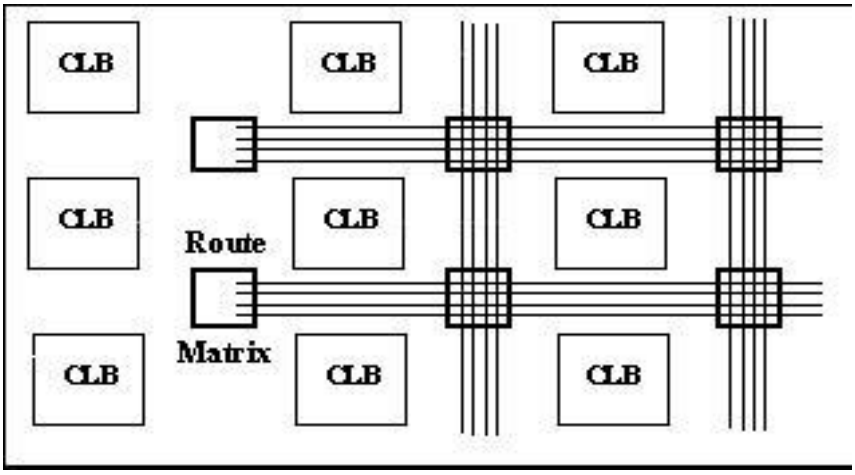

Figure. III.6. Les interconnexions entre les cellules d'un FPGA

On programme aussi les interconnexions entre les cellules, ainsi que les entrées et sorties du circuit. L'avantage des FPGA est de pouvoir être configuré sur place, sans envoi du circuit chez le fabricant, ce qui permet de les utiliser quelques minutes après leur conceptions.

III.5 Conclusion

L'UART est un circuit essentiel dans la communication série, il offre la possibilité de transmettre et de recevoir des données de manière asynchrone, l'UART peut se retrouver sous différentes formes qu'il soit un circuit indépendant ou intégré à un autre système. Avec l'émergence de nouveaux langages de description matérielle, comme le VHDL, le domaine de la recherche et de la création de systèmes numériques ne cesse de croître. Dans ce cadre, il est prévu de développer des travaux visant à concevoir un UART (Universal Asynchronous Receiver Transmitter) utilisant le langage VHDL évoqué précédemment et des circuits récents comme les FPGA.

Dans le chapitre suivant, nous allons concevoir un circuit UART en utilisant VHDL et une carte FPGA, contribuant ainsi à son implémentation et son intégration efficaces dans les systèmes numériques.

Chapitre IV

Simulation et Conception

Le schéma block de l'architecture proposé issu du logiciel Vivado est donné par la figure IV.2

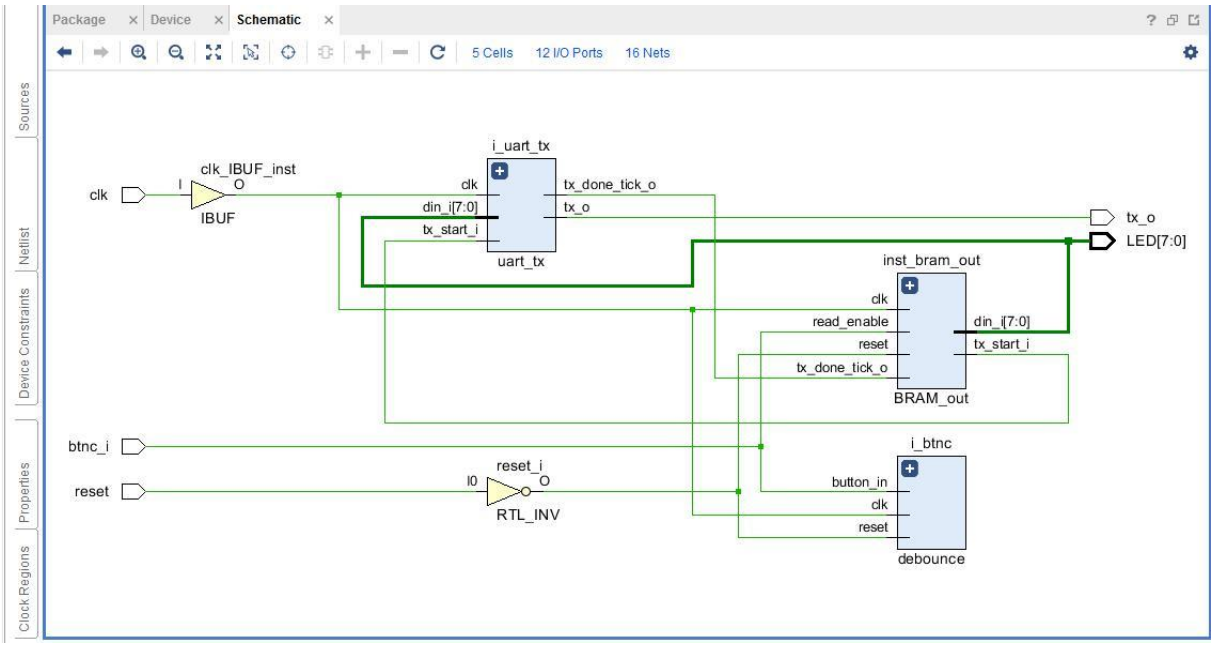

Figure. IV.1 Schéma block de l'architecture proposé.

IV.1 Machine d'état du modèle de la partie transmetteur de l'UART

Nous utiliserons la Machin d'état de Mealy synchrone pour implémenter le module de transmission.

Le diagramme d'état illustré ci-dessus à la figure IV.3 indique que le la machine d'état a quatre états ; IDLE, START, DATA et ARRÊT.

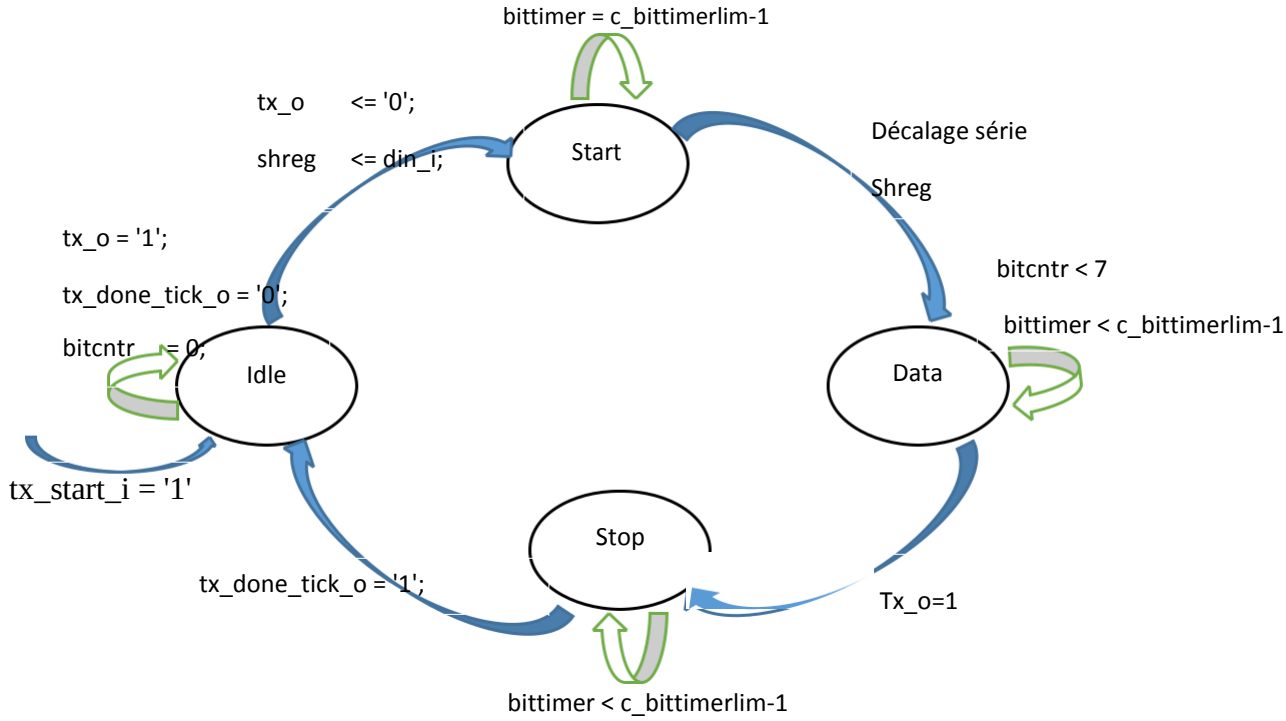

Figure .IV.2 Machine d'état de partie de transmission.

État IDLE : lorsque l'UART est réinitialisé, la machine d'état sera dans cet état. Si tx_start_i est '1' alors transfert de la machine d'état vers l'état START et sinon la machine d'état reste dans l'état IDLE.

État START : Dans cet état, lorsque shreg =7 et bittimer = c_bittimerlim-1, alors la machine d'état passera à l'état de données et sinon elle restera dans le même état.

État DATA : Dans cet état, lorsque s_tick n'est pas égal à "1" et bitcntr est égal à "7", les données se déplaceront une par une jusqu'à ce que tous les 8 bits sont transmis et stockés dans le registre b_next, c'est-à-dire b_next<= {tx, b_reg[7:1]}. Si tous les bits de données sont transmis et s_tick=1 alors la machine d'état passera à l'état STOP.

État STOP : dans cet état, si tx_done=1, alors la machine d'état repasse à l'état IDLE.

IV. 2 Simulation

La simulation du module de transmission UART_TX est donnée par la figure.IV.4

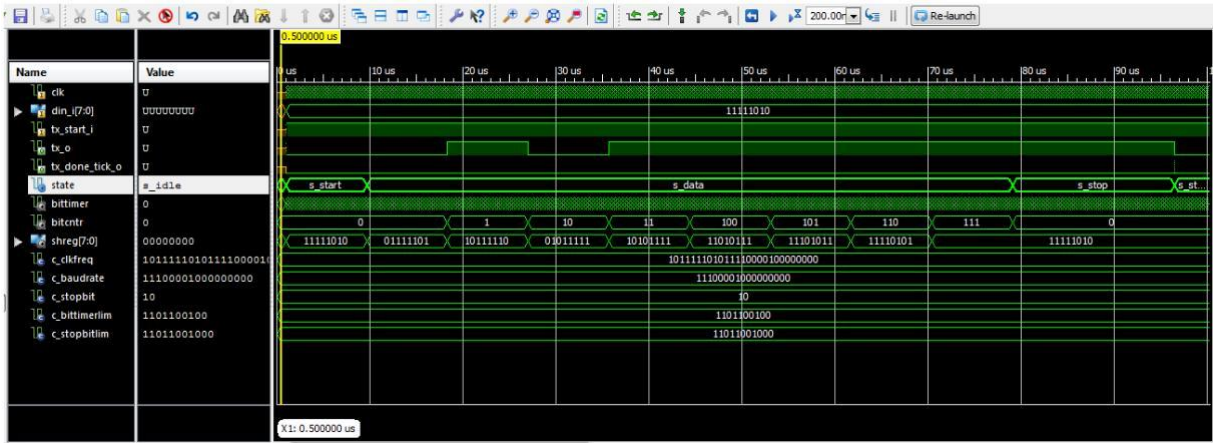

Figure .IV.3 Simulation du module de transmission UART_TX.

IV.3 Conclusion

Dans ce chapitre nous avons conçu le module de transmission UART, ce module est connecté à une RAM dont le rôle est de stocker les informations d'un capteur 16 bits. On note que le module d'émission peut transmettre 8 bits à chaque opération, ce qui nous a obligés à utiliser un registre à décalage parallèle/série.

Conclusion Générale

En conclusion, la conception de la partie émettrice de l'UART pour la transmission des données d'un capteur connecté à une carte FPGA vers un micro-ordinateur via le port série est un objectif clé à atteindre. En utilisant un registre à décalage parallèle-série, il est possible de convertir les données stockées dans une mémoire parallèle sur 16 bits en une forme séquentielle adaptée à la transmission série. Cette approche permet une communication fiable et efficace entre le capteur et le micro-ordinateur, facilitant ainsi l'échange de données et l'exploitation des informations collectées. La conception et l'implémentation réussies de cette partie émettrice de l'UART sont essentielles pour assurer une intégration harmonieuse du système global et une transmission fluide des données du capteur vers le micro-ordinateur.

GLOSSAIRE

UART : Universal Asynchronous Receiver/Transmitter.

VHSIC : Very High Speed Integrated Circuit.

VHDL : VHSIC Hardware Description Languag.

FPGA : Field-Programmable Gate Array.

IEEE: Institute of Electrical and Electronics Engineers.

PC : Program Cunter.

RS : The Recommended Standard.

SCSI : Small Computer System Interface.

SYNC : Synchronization.

DTE Data Terminal Equipement.

DCE Data Communication Equipement.

SAS : Serial attached SCSI).

EIA : The Electronic Industries Association.

FSM : Finite State Machine.

EDA : Electronic Design Automation.

CLB : Configurable Logic Bloc

BIBLIOGRAPHIE

[1] la liaison RS232 Description technique et mise œuvre 2ème édition PHILIPPE ANDRÉ, Dunod, Paris, 2002
http://fr.wikipedia.org/wiki/Duplex_%28canal_de_communication%29 , avril 2013.

[2] Léon CLEMENT « Microprocesseurs et circuits associés» 1987 2ème édition. madame céline guilleminot « étude et intégration numérique d'un système multicapteurs amrc de télécommunication basé sur un prototype virtuel utilisant le langage de haut niveau vhdl-ams » memoire de thèse doctorat de l'université de Toulouse II 01 décembre 2005.

[3] Haute Ecole d'Ingénierie et de Gestion du Canton de VHDL « Manuel VHDL synthèse et simulation » septembre 2007.

[4] KAHOUL NADHIR « COURS VHDL PLD » (introduction au langage de description VHDL avec les circuits logiques programmables et le logiciel de développement QuartusII). Université Med KHIDER BISKRA, Département de Génie Electrique, Filière Electronique, 2012.

[5] JERRY TEKOBEN « FPGA and Traffic Network Analysis » Ingénieur de conception en Génie Electrique et Télécommunication, Ecole Nationale Supérieure Polytechnique de Yaounde Cameroun 2007.

[6] Denis Rabasté,IUFM d'Aix-Marseille « programmation des CPLD et FPGA en VHDL avec Quartus http://genelaix.free.tf/IMG/pdf/aide_VHDL_quartus.pdf