POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Département de génie informatique et génie logiciel

INF3500

Conception et réalisation de systèmes numériques

Rapport de laboratoire #2

Circuits combinatoires

Critères	Points
Module UART	/ 8
Banc d'essai	/ 8
Questions	/ 2
Rapport : Présentation et qualité de la langue	/ 2
Total	/20

Soumis par :

Alexandre Morinvil, #1897222

Nicolas Valenchon, #2032097

Date :

15 avril 2020

Table des matières

1 Objectifs

L'objectifs de ce laboratoire est de mettre en pratique ses connaissances en circuit séquentielle en précédant à la description, la simulation et la synthétisation sur FPGA d'un circuit combinatoire décrit en VHDL. Le système conçu dans ce laboratoire a pour but d'implémenter le protocole de communication UART.

2 Description du système

Le système conçu dans le cadre de ce laboratoire est un module de communication sérielle composé d'un émetteur et d'un récepteur implémentant le protocole de communication UART.

Le protocole UART (Universal Asynchronous Receiver-Transmitter) est un protocole permet de procéder à la conversion lien parallèle vers lien sériel afin de procéder à la transmission. Lors de la transmission, la plus petite unité d'information est la trame. Le schéma d'une trame du protocole UART est affiché dans la figure suivante (Wikipedia).

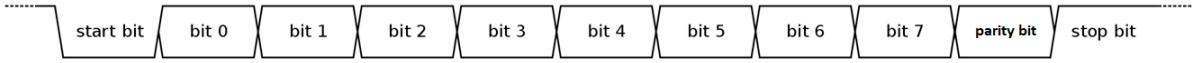

Figure 2-1 : Schema de la structure d'une trame du protocole UART

Ainsi, une trame transmise par le transmetteur consiste en un bit de départ dont la valeur est de 0 (le start bit), un ensemble de bits de données, un bit de parité et d'un bit d'arrêt (stop bit) dont la valeur est de 1. Ainsi, dans le cas du protocole implémenté dans ce laboratoire, les mots transmis contiennent 8 bits.

Enfin, la réception lors de la communication UART est un processus asynchrone qui procède à la conversion lien sériel vers lien parallèle réalisé par un module récepteur. laboratoire.

L'ensemble du code réalisé VHDL pour ce laboratoire est contenu dans le fichier `uart.vhdl` a donc consisté à compléter, d'une part, la logique de la transmission impliquant la conversion parallèle vers sériel des mots, l'encapsulation dans une trame et, d'autre part, à compléter la logique de réception asynchrone impliquant la désencapsulassions et la reconversion sérielle parallèle des mots.

La figure suivante affiche le diagramme d'état du transmetteur.

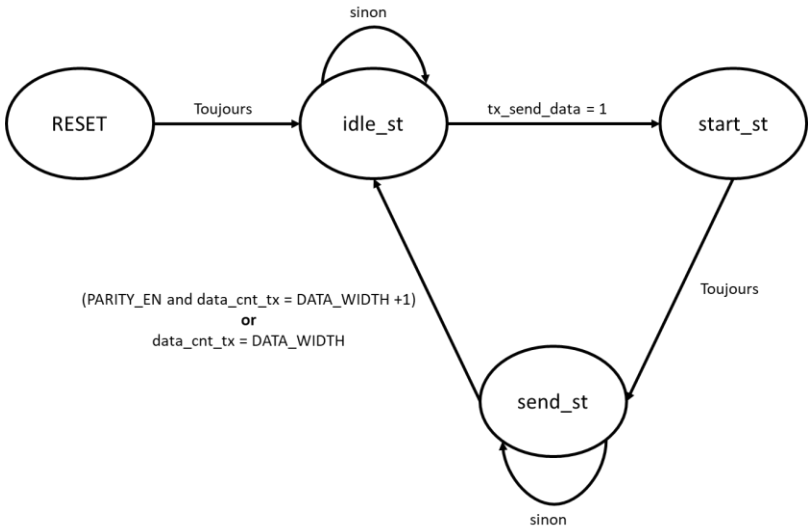

Figure 2-2 : Diagramme d'états du transmetteur UART

La figure suivante affiche le diagramme d'état du récepteur.

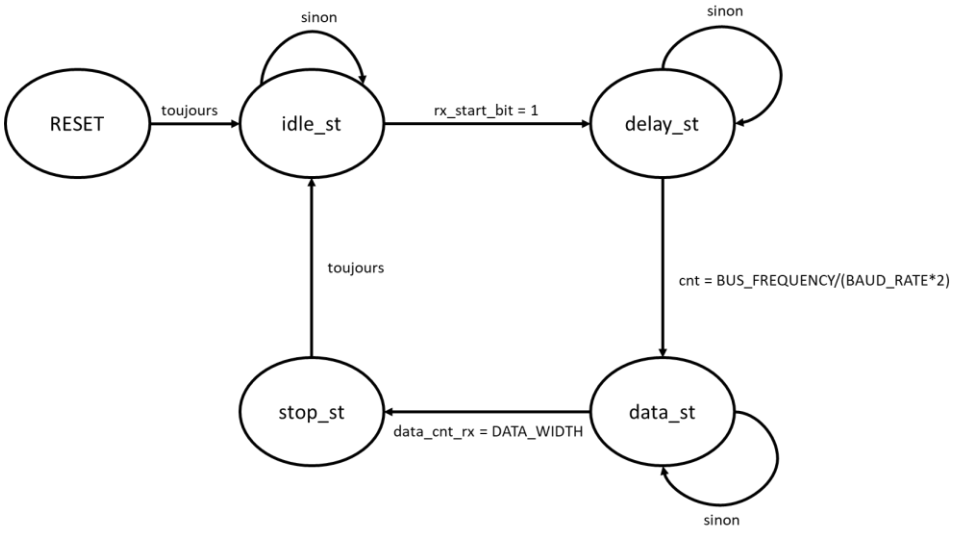

Figure 2-3 : Diagramme d'états du récepteur UART

Les détails spécifiques du traitement effectué dans chacun des états est décrit le fichier `uart.vhdl`. Les sous sections suivantes décrivent les principales fonctionnalités de chacun des états.

2.1 État RESET

L'état RESET, est l'état dans lequel le circuit se place suite lorsque l'on appuie sur le bouton de « reset » du système. Cet état est donc l'état initial du circuit. Cet état permet également de réinitialiser les données dans le tampon de tranmission.

2.2 États de l'émetteur

2.2.1 Idle_st

L'état `idle_st`, ou « état immobile », est l'état dans lequel le système demeure en attente d'un signal lui indiquant qu'une nouvelle transmission est prête. Le signal `rx_start_bit` est le signal permettant de déclencher le processus de transmission. L'état suivant le signal `rx_start_bit` est l'état `start_st`.

2.2.2 Start_st

L'état start_st, ou « état de départ », est l'état dans lequel le système transmet le bit start dont la valeur est de 0. Après le passage à cet état, le système passe systématique à l'état `send_st`.

2.2.3 Send_st

L'état `send_st`, ou « état d'envois », est l'état dans lequel le système tansmet l'ensemble des éléments de la trame (à l'exception du bit start qui est transmis lors de l'état `start_st`). Ainsi, cet état a la charge de procéder à l'envois de 8 bits de données, de calculer la parité et envoyer le bit de parité si le signal `PARITY_EN` est indique que la trame doit contenir la parité, ainsi que d'envoyer le bit de d'arrêt dont la valeur est de 1. Après avoir procéder à l'envois complet de la trame, le système retourne à l'état `idle_st` dans l'attente d'un nouveau message à transmettre.

2.3 États du récepteur

2.3.1 Idle_st

L'état `idle_st`, ou « état immobile », est l'état dans lequel le système demeure en attente d'un signal lui indiquant qu'une nouvelle réception s'amorce. Le signal `rx_start_bit` est le signal permettant de déclencher le processus de réception. L'état suivant le signal `rx_start_bit` est l'état `delay_st`.

2.3.2 Delay_st

L'état `delay_st`, ou « état délais », est l'état dans lequel le système ajuste le positionnement de son échantillonage afin d'échantilloner les bits qu'il recevra au milieu de ceux-ci. Pour ce faire, le récepteur dont la fréquence est supérieure à celle de la réception, calcul un certain nombre de cycles de son horloge qu'il laisse s'écouler afin que chacune des réception de bit subséquente s'exécute dans le milieu de la transmission des bits. Ceci permet d'obtenir une meilleur précision lors de l'échantillonnage par rapport à un échantillonage qui s'exécuterait au tout début ou à la toute fin d'un bit. Suite au délais, le système passe à l'état `data_st`.

2.3.3 Data_st

L'état `data_st`, ou « état données », est l'état dans lequel le système procède à la réception de l'ensemble des éléments de la trame (à l'exception du bit start qui est utilisé par l'état `delay_st` afin de faire l'ajustement du temps d'échantillonnage du récepteur). Ainsi, cet état a la charge de procéder à la réception de 8 bits de données, de procéder à la réception du bit de parité et de le comparer à la valeur de parité calculée si le signal `PARITY_EN` est indique que la trame doit contenir la parité. Dans le cas où la parité calculé et la parité reçue ne correspondraient pas, un signal d'erreur `rx_frame_err` est déclenché. Après avoir procéder à réception complète de la trame, le système passe à l'état `stop_st` dans l'attente d'un nouveau message à transmettre.

2.3.4 Stop_st

L'état `stop_st`, ou « état d'arrêt », est l'état dans lequel le système procède à la réception du bit d'arrêt dont la valeur de 1. Après le passage à cet état, le système passe systématique à l'état `idle_st`.

3 Vérification par simulation

Pour vérifier les différentes fonctionnalités de notre module et ainsi les tester, nous avons réalisé un banc d'essai générant plusieurs tests consécutifs.

D'une part, nous testons le module de réception de message (RX) en envoyant sur l'entrée `rx_sdata` plusieurs types de trames.

Chacun des tests commence par un reset, afin de les rendre indépendant entre eux. Le signal d'entrée `rx_sdata` est initialisé à 1 (c'est sa position de repos) et une modification sur ce dernier permet de commencer l'envoie d'un message. Pour envoyer chacun des bits composants la trame, il faut modifier la valeur de `rx_sdata` en fonction du message avec une attente de 1/Baudrate seconde entre chaque bit. C'est pour cela qu'on retrouve un wait entre chaque modification de rx_sdata. Une fois qu'une trame est envoyée, il faut attendre qu'elle soit décodée, c'est-à-dire que rx_pdata_valid soit passée à 1. On peut ensuite vérifier si le message décodé est correct et si des erreurs ont été détectées ou non.

Le fonctionnement est le même pour tous les tests concernant le récepteur, sauf pour le deuxième qui doit envoyer un message de 16 bits. En réalité ce test est très proche de l'envoie d'un message normal, à la différence qu'on réalise deux fois le test sans effectuer de reset entre les deux (ce qui a lieu entre deux tests différents).

D'autre part, nous testons le module d'émission de message (TX) en donnant un message à envoyer sur l'entrée tx_pdata et en amorçant l'envoie avec tx_send_data. Il faut ensuite vérifier chacun des bits envoyés sur la sortie tx_sdata en mesurant sa valeur tous les 1/Baudrate seconde. Un message d'erreur est envoyé à chaque bit n'ayant pas la bonne valeur.

La figure suivant affiche le chronogramme résultant de la simulation ainsi que la sortie de la console qui confirment le fonctionnement du système.

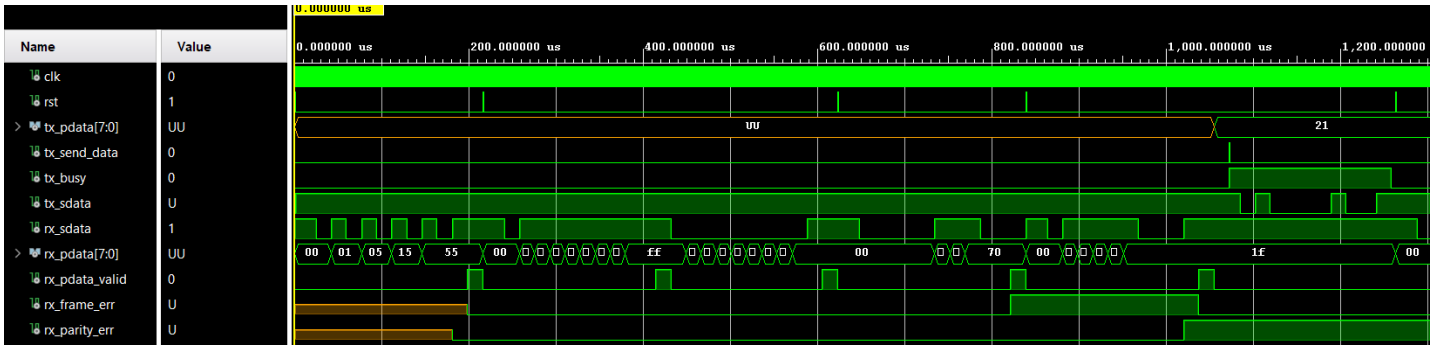

Figure 3-1 : Chronogramme de la simulation

```
run all
Failure: Simulation ended
Time: 2527318 ns  Iteration: 0  Process: /uart_tb/line__93  File: D:/Users/Nicolas/Documents/INF3500/TP5/tp5/tp5.srcs/sim_1/new/uart-tb.vhd
$finish called at time : 2527318 ns : File "D:/Users/Nicolas/Documents/INF3500/TP5/tp5/tp5.srcs/sim_1/new/uart-tb.vhd" Line 341
```

Figure 3-2 : Sortie console résultant de la simulation

Le code du banc de test est disponible dans le fichier `uart-tb.vhd`.

4 Ressources utilisées et performance

Le système n'ayant pas été implémenté, les statistiques d'utilisation ne sont pas considérées dans ce laboratoire.

5 Réponses aux questions

5.1 Question 1

Dans quel cas le calcul de la parité d'une trame ne marche pas ? i.e. il n'y a pas d'erreur dans la parité, mais il y a une erreur dans la trame.

Si deux bits (ou n'importe quel multiple de 2) sont inversés, alors la parité est erronée : le calcul de parité est réalisé avec un modulo 2, donc s'il y a un multiple de 2 d'erreurs dans les bits, alors la parité ne changera pas. En effet, 10000000 et 11100000 ont les mêmes parités, donc si le premier message est celui envoyé et le deuxième celui reçu, le récepteur ne pourra savoir qu'il est erroné.

5.2 Question 2

Pour quelle raison le récepteur UART doit faire un délai après la réception de la condition START ?

Le récepteur doit faire un délai après la réception de la condition START pour que la mesure du signal d'entrée ne se fasse pas sur un front montant ou descendant mais à un moment où le signal est stable. Cela permet d'éviter une perte d'information si le signal reçu à un quelconque retard.

6 Discussion

À l'issue de ce laboratoire, module transmetteur et récepteur UART ont été décrite en VHDL puis simulé. Un banc d'essais a été conçu. Lors de l'utilisation du banc de teste fourni avec, il a été possible de confirmer le bon fonctionnement du système. Ainsi, les résultats obtenus expérimentalement correspondent aux résultats attendus théoriquement, il n'y a donc pas eux de phénomènes inattendus lors de la réalisation du système.

Pour ce qui est de l'optimalité du système conçu, l'utilisation des ressources et la performance n'ont pas été considérés puisque le système n'a pas été implémenté.

De plus, un autre des défis de ce laboratoire réside dans le fait que ce laboratoire pour une première fois a compris d'une part la conception d'un système numérique et d'autre par la conception du banc de test permettant de valider son fonctionnement. Néanmoins, ce défi supplémentaire est formateur en si qui à traits à la familiarisation avec la conception d'un système numérique complet et les différents aspects qui entoure ce processus.

Il est dommage qu'il n'ait pas été possible de procéder à l'implémentation du système.

7 Références

Dion, O. (2020). *Labo 4 - Circuits séquentiels.* Montréal.

Wikipedia