Laboratoire #3 : Circuits séquentiels

INF3500 - Conception et réalisation de systèmes numériques
Hiver 2019

Objectifs

Ce laboratoire a trois objectifs :

— vous familiariser avec les outils utilisés lors des laboratoires du cours INF3500 ;
— vous faire implémenter un circuit séquentiel ; et
— vous faire utiliser le concept de machine à état dans un circuit séquentiel.

Préparation au laboratoire

Avant d'arriver au laboratoire, suivre les étapes suivantes :

1. Revoir la matière des cours des semaines 6 et 7 ; et,
2. Lire le guide d'utilisation de Vivado.
3. Lire la FAQ du laboratoire

Familiarisation avec les outils

Suivre les instructions du guide pratique d'utilisation de Vivado. Ce tutoriel est indispensable pour faire la simulation, la synthèse, l'implémentation et la programmation du FPGA.

Transmission série

En informatique, de très nombreux protocoles de communication (USB, Sata, PCI Express, Ethernet, UART, etc.) utilisent un lien de transmission série entre un émetteur et un récepteur. Lorsqu'un transmission série est utilisée, une unité d'information, dans ce laboratoire on considère un bit, est envoyé par cycle d'horloge vers le récepteur. Le récepteur ré-assemble ensuite l'information reçue en un mot, pour l'utiliser sur un bus de donnée ayant une largeur de plusieurs bits. Ainsi ces protocoles nécessitent un module émetteur faisant la conversion lien parallèle (bus de plusieurs bits) vers un lien série, et un récepteur faisant l'opération inverse (conversion lien série vers lien parallèle).

Dans le cadre dans ce laboratoire vous allez concevoir un **récepteur UART**.

Le protocol UART

L'UART (en anlgais Universal Asynchronous Receiver-Transmitter) est une interface de communication à basse vitesse utilisée dans les systèmes embarqués.

L'UART est un protocole de communication bidirectionnel, **asynchrone**,permettant d'échanger de l'information entre un récepteur et un émetteur. L'échange d'information est réalisée au travers d'un lien de **transmission série**. L'information échangée entre l'émetteur et le récepteur est transmise à raison d'un bit par cycle d'horloge. La période d'horloge est l'inverse de la vitesse du lien de connexion UART entre l'émetteur et du récepteur, qui est exprimé en Baud/seconde. Afin que le récepteur puisse détecter qu'une information lui est tranmise, le protocole UART spécifie un format sous lequel l'information doit être envoyée.

Consultez l'article Wikipedia afin de vous familiariser avec le protocole UART. Sans une bonne compréhension du protocol UART, il n'est pas possible de réaliser ce laboratoire. Par conséquent prenez le temps de bien lire et analyser le document décrivant le protocol UART. A la suite de la lecture de ce document vous devez être capable de répondre aux questions suivantes :

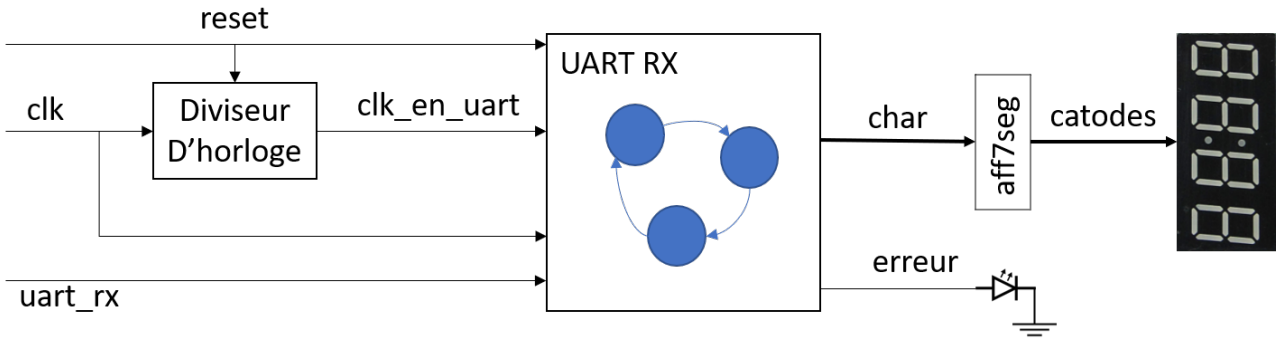
1. Qu'est-ce qu'une trame UART ?
2. Ou le mot transmis par l'émetteur est-il inséré au sein de la trame UART ?
3. Comment le récepteur UART peut-il détecter que l'émetteur est en train de lui envoyer un nouveau mot ? Quel est la valeur du premier bit transmis par l'émetteur lors de l'envoie d'un message ?
4. Comment le récepteur UART peut-il détecter que l'envoie d'un mot par le récepteur est terminé ? Est-ce que la longeur du mot envoyé est fixe ?

Remarquez bien dans le document présenté que l'horloge du récepteur et de l'émetteur est absente de la figure présentant la constitution d'une trame UART. C'est normal, l'UART est un protocol **asynchrone**. Ainsi, l'émetteur et le récepteur UART n'ont pas une horloge commune synchronisée. Par conséquent, l'horloge interne du récepteur peut être déphasée par rapport à l'horloge interne du récepteur. Ainsi, lorsque le récepteur UART reçoit un mot de l'émetteur, le récepteur doit synchroniser l'horloge utilisée pour l'échantillonnage sur l'horloge utilisée par l'émetteur. Cette particularité sera expliquée durant le laboratoire.

Dans ce TP, vous devez implémenter un **récepteur** UART récepteur, c'est-à-dire qui reçoit un mot envoyé par un émetteur.

Conception d'un recepteur UART

Le schéma d'un récepteur UART est présenté dans la figure suivante.



Un récepteur UART est composé de deux modules, un diviseur d'horloge, ainsi qu'une module faisant la conversion transmission série vers transmission parallèle. Une description de chacun de ces modules est présentée ci-dessous.

1. **Diviseur d'horloge :** Le recepteur UART ne fonctionne pas à la même fréquence que l'horloge de référence du FPGA. Par conséquent, l'horloge de référence du FPGA est divisée afin d'obtenir une période égale à la période d'échantiollonage du récepteur UART.
2. **Machine à etats pour le controle de la conversion transmission série vers une transmission parallele** : l'UART est un protocole de communication **asynchrone** entre deux machines. Les bits sont transmis de maniere sériel. Le recepteur récupère chaque bit afin de former um mot (encodée sur plusieurs bits, transmis sur un bus parallèle). La machine à état doit identifier le début de la transmission d'un mot ("Start bit", ou bit de démarrage), faire la synchronisation entre l'horloge de transmission et l'horloge de réception, récuperer le mot transmis, verifier l'intégrité de la trame, et finalement valider la séquence de fin de réception de la trame (bit d'arrêt).

Partie A : Diviseur d'horloge

Vous devez instantier un diviseur d'horloge paramétrisable pour le récepteur UART. Le code pour le diviseur d'horloge est fournis ici. Le diviseur d'horloge a deux signaux de sortie : une horloge dont la fréquence est divisée par un facteur k par rapport à l'horloge d'entrée, ainsi qu'un signal d'activation synchronisé sur le front montant de l'horloge de sortie, et valide pendant un cycle de l'horloge d'entrée.

Livrable :Effectuez une démonstration du diviseur configuré pour 1Hz et utilisez le signal généré pour faire clignoter une DEL (à la fin du laboratoire). Pour cela, utilisez le module *clock_divider.vhd* fournis.

Partie B : Récepteur UART

Vous devez développer un récepteur UART pour décoder un mot envoyé par l'émetteur UART. Les requis pour le module de réception sont décrits ci-dessus :

- Dévelopez une machine à état pour implémenter le récepteur. A chacune des étapes présentée dans la section "Machine à etats pour le controle de la conversion transmission série vers une transmission parallele" doit être associé un état dans votre machine à état.
- La vitesse d'opération doit être de 57600 bauds. Utilisez le module *reg_clken.vhd* fournis.
- Votre module doit recevoir des trames UART contenant un mot encodé sur 8 bits, avec parité impaire et stop bit simple.
- Utilisez le signal d'activation généré par le diviseur d'horloge comme marqueur de temps pour votre module.
- Échantillonnez au milieu de chaque bit transmis pour assurer un échantillage correct.
- Affichez la donnée décodée par votre récepteur UART sur un afficheur 7 segments. Le code de base de l'afficheur est disponible ici, fichier *display.vhd*
- Une DEL doit s'allumer en cas d'erreur de parité. La DEL doit rester allumée jusqu'à ce qu'une trame correcte soit reçue.
- Une DEL doit s'allumer en cas d'erreur de trame (par exemple lorsque le bit de fin n'est pas détecté). La DEL doit rester allumée jusqu'à ce qu'une trame correcte soit reçue.

Livrable : Remettre le code de votre module, le diagramme des transitions d'états ainsi qu'un diagramme bloc de votre circuit.

Partie C : Synthèse et Implémentation

Synthétisez et implémentez votre module sur la carte Nexys4 DDR. Utilisez le fichier uart.xdc ci-dessous comme base pour votre implémentation.

```
## Clock signal
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];

##7 segment display

set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { CA }];
set_property -dict { PACKAGE_PIN R10 IOSTANDARD LVCMOS33 } [get_ports { CB }];
set_property -dict { PACKAGE_PIN K16 IOSTANDARD LVCMOS33 } [get_ports { CC }];
set_property -dict { PACKAGE_PIN K13 IOSTANDARD LVCMOS33 } [get_ports { CD }];
set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { CE }];
set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { CF }];
set_property -dict { PACKAGE_PIN L18 IOSTANDARD LVCMOS33 } [get_ports { CG }];

set_property -dict { PACKAGE_PIN H15 IOSTANDARD LVCMOS33 } [get_ports { DP }];

set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { AN[0] }];
set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { AN[1] }];
set_property -dict { PACKAGE_PIN T9 IOSTANDARD LVCMOS33 } [get_ports { AN[2] }];
set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { AN[3] }];
set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { AN[4] }];
set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { AN[5] }];
set_property -dict { PACKAGE_PIN K2 IOSTANDARD LVCMOS33 } [get_ports { AN[6] }];
set_property -dict { PACKAGE_PIN U13 IOSTANDARD LVCMOS33 } [get_ports { AN[7] }];

##USB-RS232 Interface

set_property -dict { PACKAGE_PIN C4 IOSTANDARD LVCMOS33 } [get_ports { UART_TXD_IN }];
set_property -dict { PACKAGE_PIN D4 IOSTANDARD LVCMOS33 } [get_ports { UART_RXD_OUT }];

## LEDs

#set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
#set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];

##Buttons

#set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { reset }];
```

Pour tester votre circuit, vous devez communiquer avec un ordinateur. Utilisez un terminal serial tel que PuTty,pour envoyer des commandes à votre récepteur UART.

Rapport

Rédigez votre rapport selon les directives demandées ici.

Barème

Critère	points
Partie A	2
Partie B	10
Partie C	6
Rapport : présentation et qualité du français	2
Total	20