CEG 3555: Systèmes Numériques II
(Automne 2020)
Prof. Rami Abielmona
Projet: *Conception d'un UART*

15 Novembre, 2020

Contents

1 Objectif

L'objectif de ce projet est d'apprendre comment réalizer une conception d'un UART complet en VHDL.

Afin de completer ce projet, l'étudiant doit pouvoir:

- Concevoir, réaliser et examiner un module d'émetteur et un autre de récepteur;
- Concevoir, réaliser et examiner un module de générateur programmable de vitesse baud;
- Démontrer une compréhension complète pour la conception d'un UART.

2 Pré-Pro

Réalisez le test loopback décrit en figure 6, pour garantir que les signaux éléctriques de port série sont de bons voltages et de bons niveaux logiques. Étudiez la fiche technique du puce MAX232, et en particulier la figure 5 sur la page 16, puis construisez, en utilisant votre *breadboard* et les condensateurs fournis pour ce projet, le circuit qui aide à convertir les signaux CMOS aux signaux RS-232 d'une direction, et les signaux RS-232 aux signaux CMOS de l'autre direction. Un signal CMOS de voltage +5V et de niveau logique haut ('1') est converti en signal RS-232 de voltage -12V et de niveaux logique bas ('0'), tandis qu'un signal CMOS de voltage 0V et de niveau logique bas ('0') est converti en signal RS-232 de voltage +12V et de niveau logique haut ('1'). Cela aide les signaux à traverser des longs chemins éléctriques sans perdre de force. Le test *loopback* vous aidera à garantir que les niveaux logiques et voltages qui entrent et sortent du puce Cyclone sont de bonnes valeurs et ne détruiront pas le FPGA. Tout caractère appuyé au clavier du PC sera fait écho et affiché de nouveau au terminal du PC par le MAX232. **Du a COVID-19, nous connecterons directement le Cyclone UART au port série du PC, alros, ce pré-projet consisterai à tester la connexion sans breadboard.**

3 Introduction au UART

Un UART est un récepteur-émetteur asynchrone universaire, ou *Universal Asynchronous Receiver-Transmitter*, qui est utilisé pour la communication entre deux dispositifs. La plupart des ordinateurs et microcontrôleurs ont un ou plusieurs ports série d'information employés pour communiquer avec les dispositifs séries d'entrée-sortie tels que les claviers et les imprimeurs séries. On peut aussi communiquer entre deux ordinateurs (figure 1) en utilisant le UART de chaque ordinateur et un fil *crossover*, qui interconnecte l'émetteur (TxD) d'un UART au récepteur (RxD) de l'autre. Un GND commun est présent entre les deux ordinateurs pour avoir un voltage négatif commun de base.

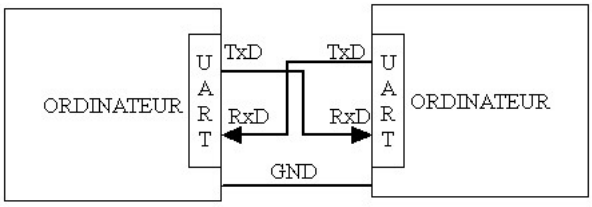

Figure 1: Communications entre deux ordinateurs

Un UART fournit le moyen d'envoyer des données avec un minimum de fils. Les données sont envoyées en bit-série et aucun signal d'horloge n'est envoyé avec. La fonction principale d'un UART est la conversion parallèle-à-série en émettant, et la conversion série-à-parallèle en recevant. Le fait qu'une horloge n'est pas transmise avec les données complique la conception d'un UART. Les deux systèmes (émetteur et récepteur) ont des signaux d'horloge séparés et non-synchronisés. Une partie de la fonction d'un UART est de *prélever* (sample) l'entrée série juste au bon moment pour capturer sûrement le train binaire. Ceci est fait en utilisant une horloge à grande vitesse pour prélever le train binaire plusieurs fois pour chaque bit d'information. Alors, *en émettant, le UART reçoit l'information en parallèle de l'application et le transmet en série sur le puce TxD, et en recevant, le UART reçoit l'information en série sur le puce RxD, et le fournit à l'application en parallèle.*

4 Encodage de l'Information

Afin que les deux systèmes se comprennent, nous avons besoin d'un encodage commun de l'information. Puisqu'il n'existe pas d'horloge, l'information (I) est envoyée asynchroniquement, un byte à la fois (figure 2). Quand aucune donnée n'est transmise, I demeure haut. Pour marquer le début d'une transmission, I devient bas pour un temps de bit, qui est désigné sous le nom du bit de départ (*start bit*). Puis, huit bits d'information sont envoyés, LSB en premier, en utilisant le code ASCII. Le dernier représente chaque caractère alphanumérique en 7-bits. Le huitième bit peut être utilisé comme bit de parité pour s'assurer de la connection. Dans la figure 2, la lettre U, encodé comme "1010101", est envoyé suivie par un 0 bit de parité, de sorte que tout le nombre de 1s soit un nombre pair (parité paire). Après la transmission des huits bits, I doit remonter en haut au moins pour un temps de bit, qui est désigné sous le nom du bit d'arrêt (*stop bit*). Alors la transmission d'un autre caractère peut commencer à tout moment. Le nombre de bits transmis par seconde est fréquemment désigné sous le nom de la vitesse de baud (*baud rate*).

En envoyant, le UART prend huit bits d'information parallèle et converti les données à un train binaire série qui consiste d'un bit de départ, de huits bits d'information, et d'un ou plus de bits d'arrêt. En recevant, le UART détecte

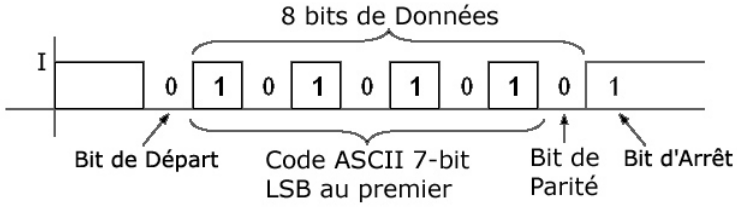

Figure 2: Encodage standard de l'information en série

le bit de départ, reçoit les huits bits de données, et converti l'information série en information parallèle quand il détecte le bit d'arrêt. Puisqu'aucune horloge n'est transmise, le UART doit synchroniser le train binaire entrant avec l'horloge locale.

5 Composants d'un UART

Un UART est composé de quatre composants principaux: l'émetteur, le récepteur, le générateur de la vitesse de baud et les registres du UART. On suppose que le UART est connecté à un microcontrôleur par un bus de donnée et un autre d'adresse, pour permettre au CPU de lire et écrire aux registres du UART. Référez-vous à la figure 3 pour les discussions des composants suivantes.

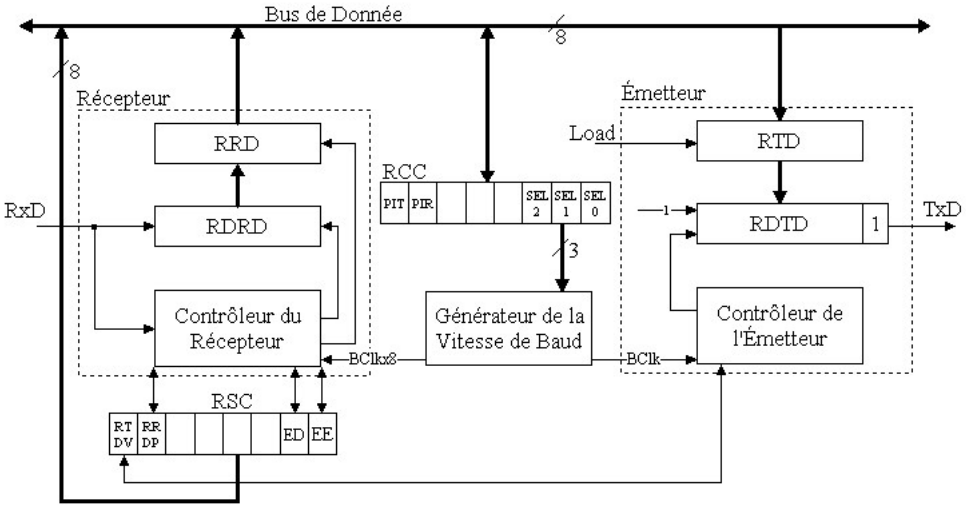

Figure 3: Les composants du UART

5.1 Registres du UART

Il existe, habituellement, quatre registres pour chaque UART: le registre de réception de donnée (RRD), le registre de transmission de donnée (RTD), le registre de statut de communication (RSC) et le registre de contrôle de communication (RCC).

RRD Ce registre à 8-bits reçoit les données d'un registre à décalage à 8-bits (RDRD) et contient un byte reçu en série par le UART;

RTD Ce registre à 8-bits transmet les données de l'application à un registre à décalage à 8-bits (RDTD) et contient un byte qui doit être transmise en série;

RSC Ce registre à 8-bits contient les signaux de statut du UART: RTDV, RRDP, ED et EE; et

RCC Ce registre à 8-bits contient les signaux de contrôle du UART: PIT, PIR, et SEL[2:0].

Le RDTD (Registre à Décalage de Transmission de Donnée) accepte la sortie du RTD, et le RDRD ((Registre à Décalage de Réception de Donnée) fournit l'entrée du RRD. Les signaux de statut et de contrôle sont définit comme suit dans la table 1.

Signal	*S/C*	*Description*
RTDV	Statut	Registre de Transmission de Donnée Vide
RRDP	Statut	Registre de Réception de Donnée Plein
ED	Statut	Erreur de Dépassement (Overrun)
EE	Statut	Erreur d'Encadrement (Framing)
PIT	Contrôle	Permit d'Interruption de Transmission
PIR	Contrôle	Permit d'Interruption de Réception
SEL[2:0]	Contrôle	Selection de la Vitesse de Baud

Table 1: Signaux de statut et de contrôle du UART

5.2 Émetteur

L'émetteur du UART consiste des registres RTD et $RDTD$ et d'un contrôleur d'émetteur. Le bit de statut $RTDV$ dans le RSC est mis (à 1) par le contrôleur quand RTD est vide. Quand le microcontrôleur est prêt à envoyer de l'information, le suivant se produit:

1. Le microcontrôleur attend jusqu'à ce que $RTDV =' 1'$ et puis enregistre un byte de donnée dans RTD. $RTDV$ est ensuite remis à 0.
2. Le UART transferts le byte du RTD au $RDTD$ et mets $RTDV =' 1'$.

3. Le UART sort un bit de départ ('0') pour un temps de bit, et puis décale $RDTD$ à la droite pour transmettre les huit bits de donnée suivi par un bit d'arrêt ('1').

5.3 Récepteur

L récepteur du UART consiste des registres RRD et RDRD et d'un contrôleur de récepteur. Le bit de statut $RRDP$ dans le RSC est mis (à 1) par le contrôleur quand RRD est plein. Quand le UART détecte un bit de départ, le suivant se produit:

1. Le UART lit et décale les huit autres bits en série dans $RDRD$.
2. Quand les huits bits et le bit d'arrêt ont été reçus, la valeur de $RDRD$ est enregistré dans RRD, et le signal $RRDP$ est mis à 1.
3. Le microcontrôleur vérifie le signal de statut, et s'il est mis, le RDR est lu et $RRDP$ est remis à 0.

Le train binaire entrant le puce RxD n'est pas synchronisé avec l'horloge locale ($BClk$). Si on essayait de lire RxD sur chaque impulsion positive de $BClk$, on aurait un problème si RxD change près de l'impulsion. On pourrait avoir d'autres problèmes si la vitesse de bit du train binaire entrant différait même un peu de $BClk$, puisqu'on pourrait lire certains bits au mauvais temps. Pour éviter ces problèmes, on va prélever RxD huit fois pendant chaque temps de bit (certains systèmes prélève 16 fois chaque bit). On va prélever aux impulsions positives de $BClkx8$, qui sont indiquées par les flèches de la figure 4. Idéalement, on doit lire la valeur du bit au milieu de chaque temps de bit pour la fiabilité maximum. Alors, quand RxD descend à 0, on va attendre quatre périodes de $BClkx8$, pour être près du milieu du bit de départ. Puis, on continue à lire à chaque huit périodes de $BClkx8$ pour enregistrer chaque bit de donnée, jusqu'à ce que nous ayons lu le bit d'arrêt.

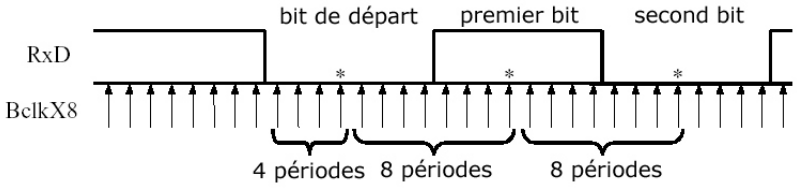

Figure 4: Le prélèvement de RxD avec $BClkx8$

5.4 Générateur de la Vitesse de Baud

Le générateur de la vitesse de baud est programmable en utilisant les trois bits de contrôle ($SEL[2:0]$) dans RCC. Puisqu'on utilise trois bits, on a le choix de 8

vitesses de baud. Supposons que l'horloge du système est 25.175 MHz, et qu'on veut des vitesses de baud de 300, 600, 1200, 2400, 4800, 9600, 19200 et 38400. La fréquence maximum pour $BClkx8$ nécessaire est $38400x8 = 307200$. Pour obtenir cette fréquence, on divise 25.175 MHz par 81.95. Puisque nous pouvons diviser seulement par un nombres entier, nous devons accepter une petite erreur dans la vitesse baud ou ajuster la fréquence de l'horloge de système à 25.1904 MHz pour compenser. On peut voir de la figure 5 que le premier diviseur divise l'horloge par 41, et puis envoie cet horloge à un compteur binaire à 8-bits. Chaque bascule de ce dernier correspond à une division par 2, alors la première correspond à une division par 2, la deuxième correspond à une division par 4, jusqu'à la huitième et dernière qui correspond à une division par 256. Une de ces huits horloges est choisie en utilisant un multiplexeur et les signaux de commande $SEL[2:0]$.

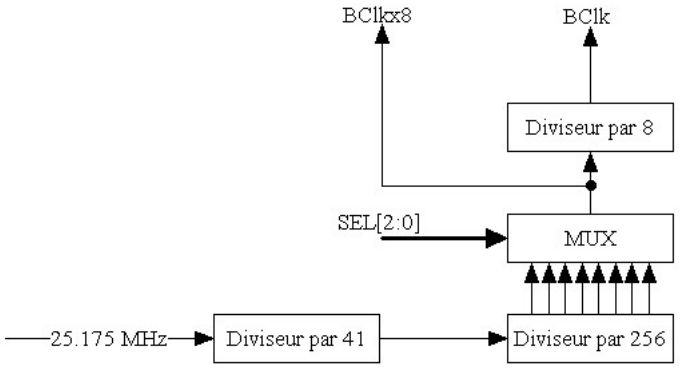

Figure 5: Le générateur programmable de la vitesse baud

En supposant une horloge de système de 25.175 MHz, on obtient les fréquences spécifiées dans la table 2.

SEL[2:0]	*Vitesse baud (BClk)*
000	38377
001	19188
010	9594
011	4797
100	2398
101	1199
110	600
111	300

Table 2: Fréquences de $BClk$

5.5 Générateur d'Interruption

La génération d'une interruption IRQ qui interrompe le CPU quand l'émetteur ou le récepteur du UART exige l'attention. Quand PIR (permit d'interruption de réception) est mis à 1 dans RCC, IRQ est générée quand $RRDP$ ou ED est '1'. Par contre, quand PIT (permit d'interruption de transmission) est mis à 1 dans RCC, IRQ est généré quand $RTDV$ est '1'.

5.6 Décodeur d'Adresse

Le UART est connecté à un ordinateur ou microcontrôleur de sorte que nous puissions lire et écrire des caractères de/à l'extérieur. Nous aurons besoin d'un décodeur d'adresse pour choisir quelles actions nous voulons exécuter, et par conséquent quels registres nous voulons lire et écrire. Premièrement, on a besoin d'un permis pour le UART qu'on nomme $UART_Select$. Quand ce dernier est '1', alors nous employons le UART, autrement nous n'avons pas besoin de l'accès au UART. Deuxièmement, on a besoin d'une table de décodeur d'adresse, comme suit dans la table 3.

ADDR[1:0]	$R/\overline{W}$	*Action*
00	1	BUS_DONNÉE ← RRD
00	0	RTD ← BUS_DONNÉE
01	1	BUS_DONNÉE ← RSC
01	0	BUS_DONNÉE ← Z
1d	1	BUS_DONNÉE ← RCC
1d	0	RSC ← BUS_DONNÉE

Table 3: Table de décodeur d'adresse

6 Projet

Dans ce projet, vous allez, en utilisant la théorie d'un récepteur-émetteur synchrone universaire présentée dans les sections précédentes, procéder à réaliser un **UART**, en concevant les composants du UART (section 5), et en réalisant le circuit de contrôleur de UART et finalement en connectant le tous aux capteurs et aux déclencheurs réelles. À partir des caractéristiques données de chaque composant, votre but est de concevoir, réaliser et examiner les circuits séquentiels synchrones, en utilisant le logiciel de Quartus II et la carte DE-2 d'Altera (pour plus de détails, voir la section 7) qui réalisent ensemble un UART incorporé avec le module de contrôleur de feu de circulation, le dernier réalisé en laboratoire #3, pour produire un système réelle débogable!

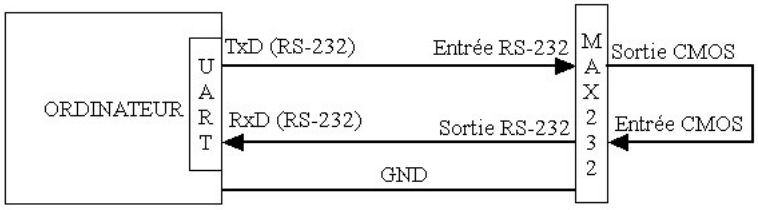

Figure 6: Test loopback du pré-projet

6.1 Spécifications de Problème

Votre équipe a été louée pour concevoir un contrôleur de feu de circulation, qui montre les mêmes caractéristiques que celle précédemment conçues dans le laboratoire #3, sauf que le contrôleur produit des messages de débogage. Les messages correspondent à un état spécifique dans la machine à états finis du feu de circulation. On suppose la présence de quatre états principaux: (1) rue principale verte/rue latérale rouge, (2) rue principale jaune/rue latérale rouge, (3) rue principale rouge/rue latérale verte, et (4) rue principale rouge/rue latérale jaune. Pendant le premier état, le message de débogage doit être "Pv_Lr", et pendant le deuxième état, le message de débogage doit être "Pj_Lr", et pendant le troisième état, le message de débogage doit être "Pr_Lv", et finalement pendant le quatrième état, le message de débogage doit être "Pr_Lj". Tous les messages de débogage doivent terminer avec un retour chariot pour commander l'ordinateur de retourner au début de la ligne de texte en cours. Chaque message de débogage contient 6 octets, incluant le carriage return. Les derniers doivent être transmis en utilisant le UART du Cyclone, au UART de l'ordinateur et terminant par afficher les caractères sur l'écran de l'ordinateur, en utilisant un programme de lecteur de port série (p.ex. HyperTerminal ou MiniCom). Votre tâche est de developper un contrôleur de feu de circulation qui affiche des messages de débogage sur l'écran d'un ordinateur. Les spécifications de rendement du contrôleur sont designées par le tableau 4. **Dans ce projet, la solution**

exigée n'utilise pas un clavier connecté à la carte DE-2 d'Altera et n'inclus pas le MAX232!

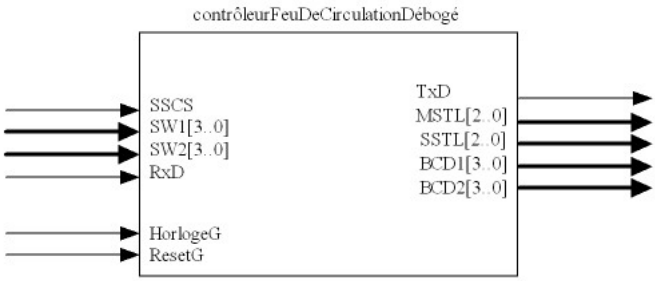

Figure 7: Contrôleur de feu de circulation

Type	*Nom*	*Description*
Donnée	HorlogeG	Horloge globale nécessaire pour synchroniser les circuits
Donnée	ResetG	Reset globale necéssaire pour apporter les circuits aux états connus
Donnée	SSCS	Capteur de voiture de la rue latérale
Donnée	SW1[3..0]	Commutateur DIP à 4-bit de la rue principale
Donnée	SW2[3..0]	Commutateur DIP à 4-bit de la rue latérale
Donnée	RxD	Fil de réception de port série
Sortie	TxD	Fil de transmission de port série
Sortie	MSTL[2..0]	Sortie du feu de circulation de la rue principale en utilisant le codage one-hot
Sortie	SSTL[2..0]	Sortie du feu de circulation de la rue latérale en utilisant le codage one-hot
Sortie	BCD1[3..0]	Valeur BCD à 4-bit de compteur courant pour le chiffre gauche
Sortie	BCD2[3..0]	Valeur BCD à 4-bit de compteur courant pour le chiffre droit

Table 4: Spécifications de données et sorties

6.2 Conseils de Projet

La section suivante est censée pour présenter une solution possible au problème, tout en discutant les majeures parties de cette solution.

Présenté en figure 8 est le diagramme de réseau du projet (**noter que du a COVID-19, TxD et RxD seront directement connectés au UART de l'ordinateur**). Incluses dans cette figure sont les entrées et les sorties principales du système.

La figure 9 démontre une solution potentielle du projet en incluant les blocs du laboratoire #3. Notez clairement l'utilisation de quatre différentes FSMs

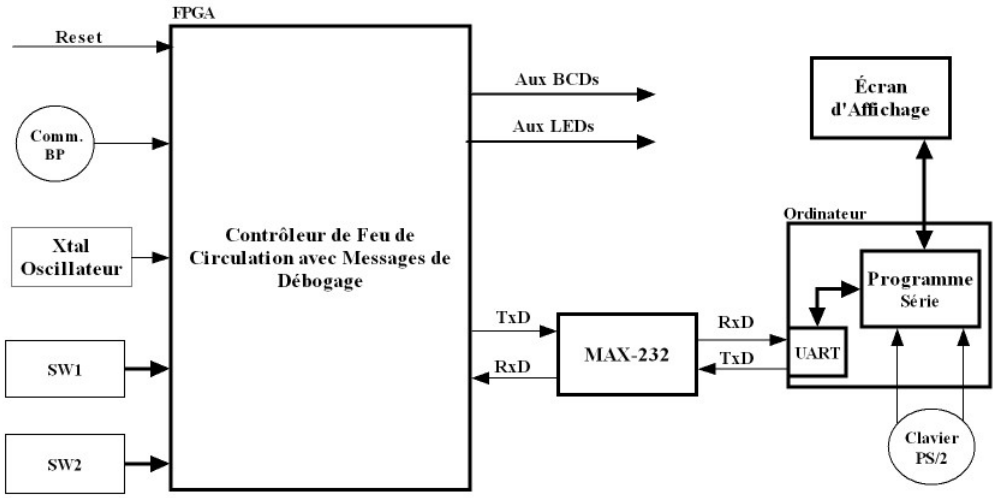

Figure 8: Diagramme de résau du projet

au même temps: (1) FSM du contrôleur de lab #3, (2) FSM du UART, (3) FSM de l'émetteur et (4) FSM du récepteur. Le FSM du UART est unique puisqu'il simule la présence d'un microcontrôleur qui interface normalement au UART. Le dernier accepte l'information d'état (2 bits pour représenter 4 états) et produit les signaux pour envoyer ou recevoir des bytes sur le UART. N'oubliez pas les signaux *Select* et IRQ. Le premier permet l'accès au UART, et le dernier exige l'attention du UART FSM quand le UART a envoyé ou a reçu un caractère. Aussi, n'oubliez pas le décodeur d'adresse (section 5.6) qui est présent dans le bloc du UART et qui décide quel registre commande le bus de données (cycle d'écriture) ou quel registre sauve la valeur courante présente sur le bus de données (cycle de lecture). Finalement, le bloc UART lui-même a été déjà démontré en figure 3, alors vérifiez cette figure pour votre réalisation finale.

Finalement, présenté en figure 10 est le diagramme du laboratoire #3 modifié pour ce projet-ci. Ce diagramme est seulement démontré pour revoire la solution potentielle et pour l'étudier comme incorporé dedans la figure 9.

6.2.1 Décomposition

Nous pouvons décomposer le diagramme de système (figure 9) en cinq composants principaux:

Lab #3 Ce composant est le contrôleur de feu de circulation déjà construit pour le laboratoire #3. Le FSM interne doit être re-conçu pour sortir l'information d'état;

UART FSM Ce composant simule un microcontrôleur et recoit l'information

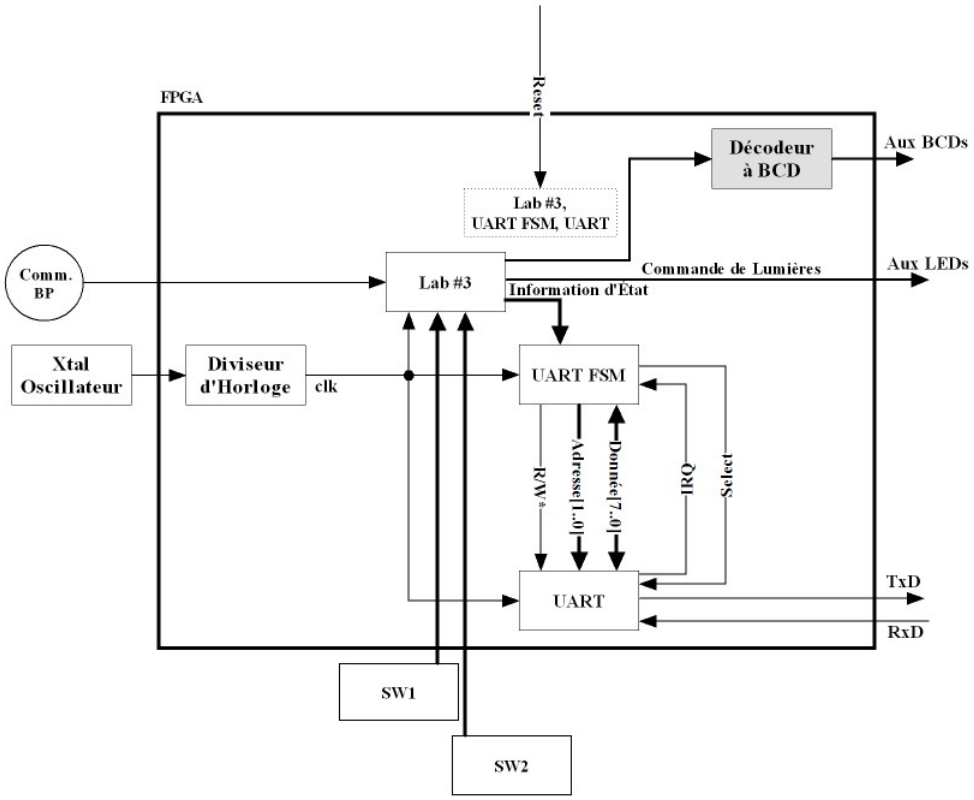

Figure 9: Un diagramme de système potentiel

d'état et interface au UART pour envoyer les messages de débogage à chaque changement d'état. Ce composant recoit aussi les caractères du UART et décide d'inverser ou non les messages de débogage;

UART Ce composant est l'ensemble des registres de UART, de l'émetteur, du récepteur et du générateur de vitesse baud qui est employé pour envoyer ou recevoir des caractères ASCII sur les fils TxD et RxD;

Diviseur d'Horloge Ce composant divise l'horloge globale en une plus lente pour contrôler les différents composants. Le but est de fournir les messages de débogage en temps lisible par un humain; et

Décodeurs à BCD Ces composants sont employés pour démontrer visuellement la valeur courante d'un compteur sur un affichage duel-chiffre de BCD.

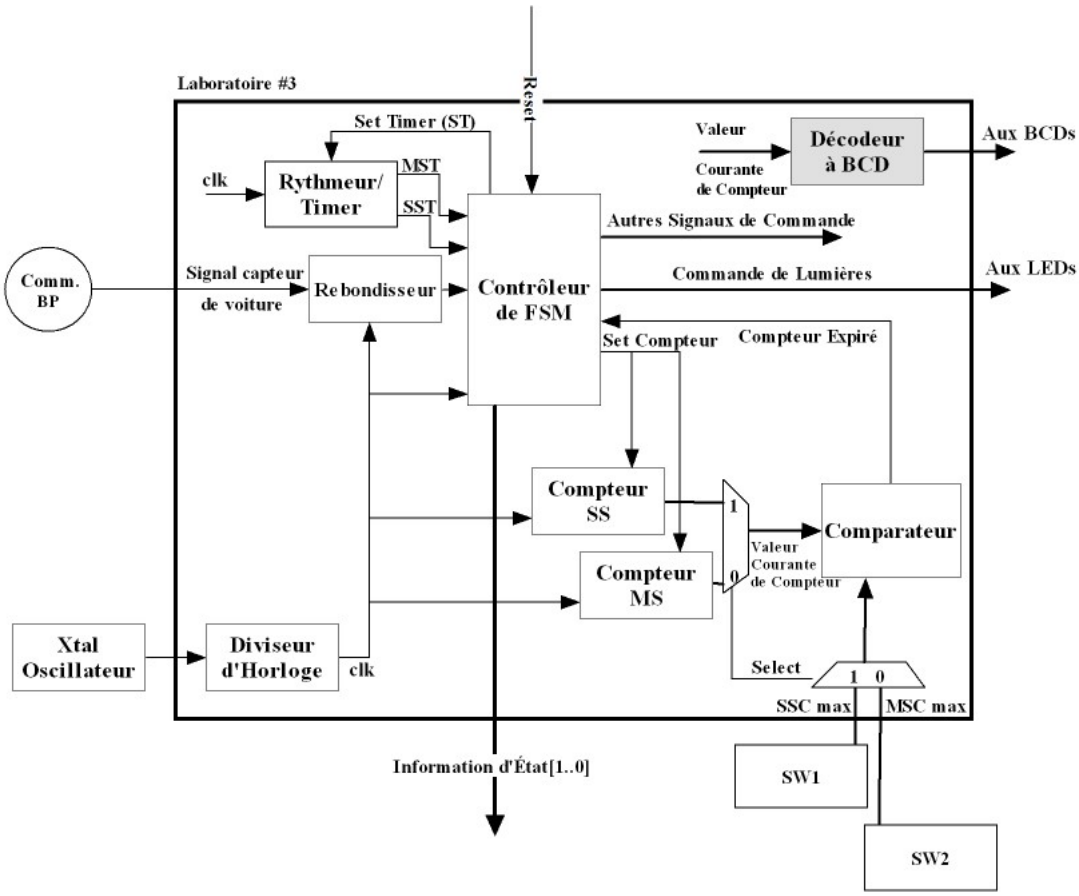

Figure 10: Un diagramme de système du lab #3 modifié pour le projet

6.2.2 Conseils au Niveau de Système

Il est digne pour noter qu'une **remise globale** est fournie à tout contrôleur de FSM, qui remet à zéro efficacement tous les registres d'état chez le FSM, aussi bien que toutes les sorties du FSM (sortant du système ou à d'autres modules dans le système). Ceci causera les autres blocs d'entrer dans leur état de remise, et ainsi par exemple, remettre le rythmeur à sa valeur de compte initiale ou le registre à décalage à 0. Cette *distribution de reset* n'est pas montrée dans la figures, mais on le suppose que les concepteurs prendront soin de ce réseau.

Si vous ne pouvez pas réalisez le mode d'interruption du UART en utilisant PIT et PIR dans SCC, vous pouvez arranger pour leur contre-partie de *mode de polling*. Par conséquent, vous pouvez lire le registre de statut jusqu'à ce qu'un bit soit affirmé (soi le RRDP ou

le RTDV), puis lire/écrire de/à un registre (soi le RRD ou le RTD).

6.3 Boni: Incorporez les Décodeurs de BCD

Pour un **boni de 3 notes**, incluez le bloc de **décodeur de BCD** pour réaliser l'affichage des valeurs courantes d'un compteur sur les BCDs dans le format décimal, comme est décrit dans le laboratoire #3. Vous devez démontrer votre nouvelle conception à votre AE.

7 Restrictions de Conception

- Les implémentations de Verilog ne seront pas acceptées. Effectuez toutes les réalisations en code de VHDL seulement
- Le niveau comportemental (behavioral) de modeler ne sera pas accepté. La conception devrait être faite au niveau structurel de modeler
- L'utilisation de Logique de Transfert-Registre (RTL) en conception et codage est obligatoire
- Employez la conception graphique en Quartus II pour l'entité supérieure (top-level), et employez votre jugement pour tout autre bloque secondaire. Cependant, tous les modules atomiques doivent être réalisés en VHDL (i.e. bascule de type D, 1-bit additionneur, 1-bit comparateur, etc.)
- Aucune utilisation de coeur est permetté (i.e. LPMs d'Altera ou d'autre IP coeurs gratuits de l'Internet). Tous les modules doivent être conçus et réalisés par le groupe
- Le groupe peut employer n'importe lequel des coeurs fournis dans le cours dans leur conception (c.-à-d. rebondisseur, diviseur d'horloge, registre, registre à décalage, additionneur, compteur, comparateur et ainsi de suite). Rappelez-vous de mettre en référence tous les coeurs que le groupe eux-mêmes n'a pas développés
- Si vous utilisez le coeur de diviseur d'horloge en particulier, son niveau comportemental de modeler est accepté
- Le groupe peut utiliser une machine de Mealy ou de Moore en tant que leur contrôleur de n'importe quel FSM
- Le groupe peut réutiliser les modules qui été concus dans les laboratoires précédents
- L'entité du niveau supérieur (top-level) est donnée dans le format de spécifications d'entréee-sortie, mais les sous-modules sont laissés jusqu'au groupe

- La conception doit être synchrone et globalement réintialisable. Ceci signifie que l'horloge globale et le signal "Reset" sont exigés dans tous les blocs fonctionnels
- Simulez vos conceptions et vérifiez vos résultats de simulation avec vos résultats théoriques (par exemple: essayer d'appuyer sur le bouton-poussoir pour simuler une voiture sur la rue latérale)
- Téléchargez votre conception au Cyclone FPGA de la carte DE-2, et employez des contacts DIP pour entrer SW1[3..0] et SW2[3..0], un commutateur à bouton-poussoir pour entrer SSCS (le capteur de voiture), et deux ensembles de trois LEDs pour les feux de circulation démontrées dans la figure 7. Finalement, le signal TxD sort d'un trou de votre en-tête de prototypage et le signal RxD entre dans un autre trou de la même en-tête, tandis qu'un GND commun est pris du trou #12 de l'en-tête de prototypage, JP1 ou JP2, de la carte DE-2 et connecté au GND de port série de l'ordinateur (fourni par l'en-tête DB-9 à trois fils). Si réalisant la section de boni, deux décodeurs à BCD doivent être employés pour démontrer la fonctionnalité correcte du compteur (BCD1[3..0] et BCD2[3..0])
- Chaque groupe doit démontrer une version fonctionnante du projet au professeur, avant la date du rapport

8 Rappels de Rapport

- Incluez les simulations de synchronisation (timing) avec des explications, pour tous les fichiers source de VHDL
- Décrivez et commentez tous vos fichiers source de VHDL
- Incluez une représentation d'organigramme de votre solution au problème
- Incluez un schéma fonctionnel de votre solution au problème
- Si en utilisant la conception d'ASM, incluez tous les diagrammes appropriés et le datapath et controlpath
- Si en utilisant la conception de FSM, incluez tous les diagrammes, tables et réalisations de circuit appropriés
- Décrivez, dans votre propres mots, votre solution au problème
- Décrivez vos obstacles de conception et comment ils ont été surmontés
- Apposez tous les fichiers graphiques de conception (*.bdf) à votre rapport
- Soumettez tous les fichiers de VHDL et conception graphique avec votre rapport

9 Remerciements

Les figures 2 3, 4 et 5 et la description de l'opération d'un UART sont prises hors du manuel: *Digital Systems Design using VHDL* par Charles H. Roth, Jr.