Liaison série synchrone avec application

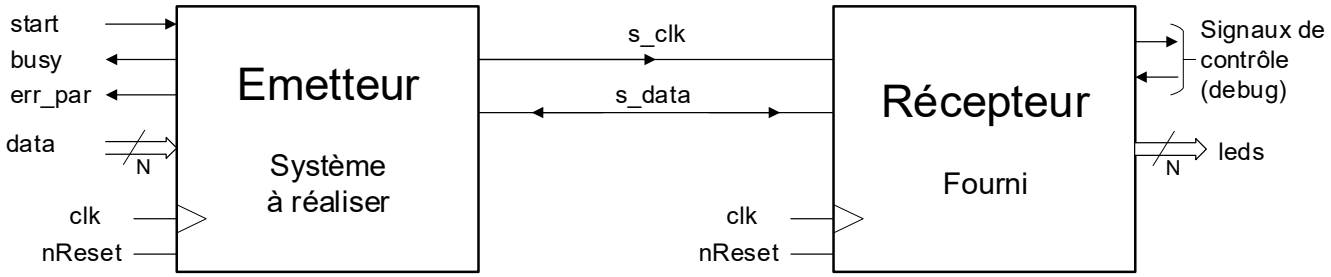
Mandat

Le but de ce projet est de concevoir, réaliser et tester un émetteur série. L'objectif est de pouvoir disposer d'un système qui permet de transmettre une donnée de N bits via une transmission série. Celle-ci sera basée sur le principe d'une ligne série synchrone avec 2 fils (voir ci-après). La fiabilité de la transmission est assurée par un bit de parité. Le nombre de bits, de la donnée, transmis doit être configurable dans la description VHDL. Le système sera conçu selon la méthodologie des Machines Séquentielles Synchrones complexes (MSS complexe).

Dans une seconde partie, une application sera réalisée pour l'affichage des secondes d'une horloge.

1ère Partie : Schéma bloc de l'ensemble

Voici le schéma bloc de l'ensemble avec l'émetteur et le récepteur.



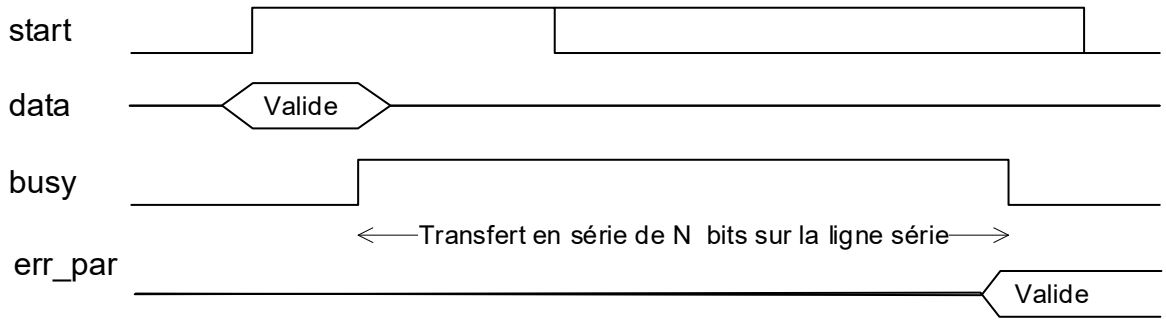
Le circuit émetteur transmet en série la donnée avec un bit de parité. Celle-ci est vérifiée par le récepteur qui transmet alors une confirmation. Cette réponse est vérifiée par l'émetteur, qui met à jour l'état des leds seulement si la vérification de la parité n'indique aucune erreur.

Les éventuels signaux de contrôle du récepteur ne sont pas indiqués. Ceux-ci seraient éventuellement utile pour la phase de mise au point du système.

Entrées/sorties de l'émetteur

- nReset_i signal d'initialisation à action asynchrone, actif bas
- clk_i signal d'horloge du système d'une fréquence de 1 MHz.
- start_i entrée indiquant une demande de transfert d'une nouvelle valeur de l'état des sorties. L'émetteur doit envoyer val_out sur la ligne série
- data_i entrées de *N* bits correspondant à la donnée à transmettre en série
- busy_o sortie indiquant un transfert en cours.
- err_par_o sortie indiquant une erreur de parité lors du transfert série.
- s_clk_o ligne d'horloge de la liaison série (sortie)
- s_data_io ligne de donnée de la ligne série, signal bidirectionnel

Spécification de l'émetteur



Protocole de la liaison série synchrone

La connexion entre le maître (émetteur) et l'esclave (récepteur) est réalisée via une ligne série synchrone avec 2 signaux. Le terme synchrone signifie qu'un signal d'horloge est transmis avec les données pour synchroniser le transfert.

Voici la spécification de ces 2 signaux :

- s_clk, signal d'horloge qui permet de synchroniser les informations transmises sur la ligne s_data.
- s_data signal sur lequel l'information (data) est transmise en série à raison d'un bit à chaque période du signal s_clk. Cette ligne est bidirectionnelle. Elle doit être pilotée par une sortie à collecteur ouvert

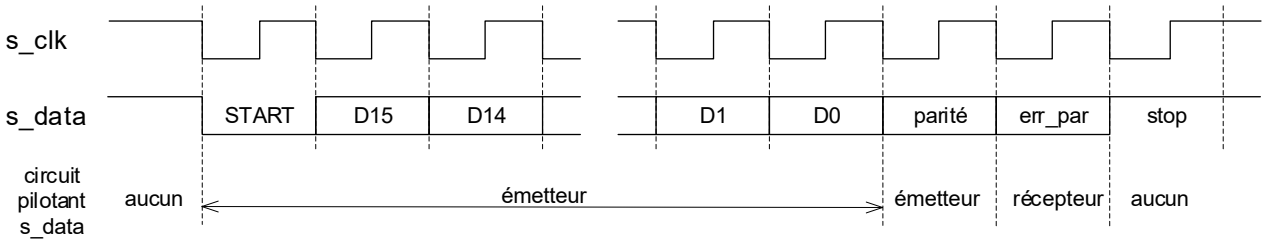
Lorsqu'il n'y a aucune transmission, les signaux s_clk et s_data sont à l'état haut ('1').

Lors d'une transmission, l'émetteur fourni l'horloge synchrone s_clk. Il place une information (un bit) sur s_data sur le flanc descendant de s_clk. Tandis que le récepteur échantillonne les données de la ligne s_data sur le flanc montant de s_clk. Le transfert d'un mot complet nécessite *N+4* périodes consécutives de s_clk.

Voici les étapes d'un transfert :

- La transmission débute par un « bit de Start » : le signal s_data est mis à '0' durant une période.
- Après le bit de Start, l'émetteur place successivement sur s_data les *N* bits de l'information, soit data, en commençant par les bits de poids fort.
- Dans la période d'horloge suivante, l'émetteur met le « bit de parité » sur s_data. Le bit de parité est à '1' si l'information transmise sur la ligne série comporte un nombre pair de '1'. Nous parlons de parité paire (*even parity* en anglais).
- Après avoir envoyé le bit de parité, l'émetteur libère la ligne pendant une période (état haute impédance). Le récepteur peut alors transmettre le résultat de la vérification de la parité. S'il n'y a pas d'erreur il met un '0' sur s_data, sinon il met un '1'. Cette information est lue par l'émetteur qui mettra alors à jour l'information reçue et indiquera, si nécessaire, une erreur en activant le signal err_par.
- Finalement, la transmission est terminée par un bit STOP. Il y aura ainsi au moins une période d'horloge de s_clk entre deux transmissions.

Nous donnons ci-après le chronogramme du transfert d'une donnée de 16 bits.



Le transfert sur la ligne s_data est bidirectionnel. C'est l'émetteur qui active un transfert. Lors de la phase de validation de la parité (Err_Par), c'est le récepteur qui fournit l'information.

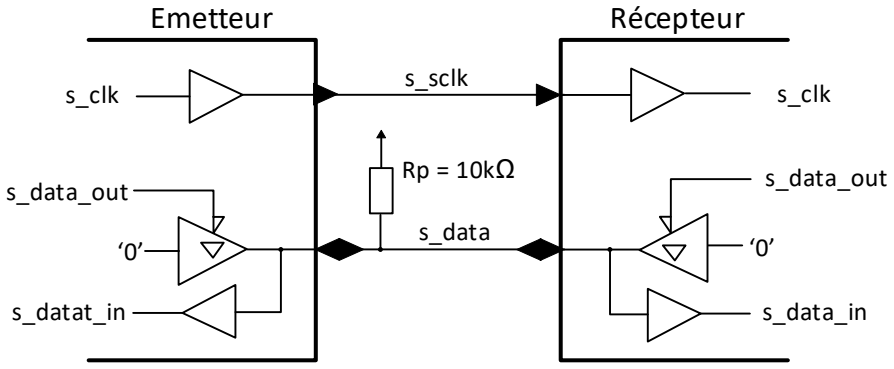
Caractéristique particulières :

- Le signal s_clk est actif seulement durant la transmission. Entre deux transferts, ce signal reste à l'état '1'.
- La ligne s_data est actionnée via une sortie à collecteur ouvert. Il est ainsi possible d'avoir 2 sources connectées ensemble.
- Au repos, la ligne s_data est au niveau '1'. Cet état est obtenu par une résistance de *pull-up*.
- La fréquence de s_clk sera de 50 KHz. Le facteur de division, pour obtenir la fréquence de s_clk, sera défini dans un paquetage. Si d'autres tailles sont nécessaires au projet, elles seront aussi définies dans ce paquetage.
- Le taille de la donnée transmise sur la ligne série sera définie par une constante dans un paquetage. Le nombre de bits varie de 2 à 32. Il sera défini pas défaut à 8 bits.
- Lorsqu'il y a une erreur de parité : err_par = '1', sinon il est à '0'.
- La fréquence de l'horloge du système est de 1MHz.

Caractéristiques électriques du bus série synchrone

La ligne s_clk est toujours pilotée par l'émetteur (maître). La ligne s_data est pilotée soit par l'émetteur (Start, D*N-1* à D0, parité) ou soit par le récepteur (err_par). Il est nécessaire d'utiliser des portes à collecteur ouvert (état '0' ou 'Z') pour générer le signal s_data. Ce type de portes n'existe pas dans les circuits logiques programmables (CPLD ou FPGA). Nous allons réaliser la porte à collecteur ouvert avec une porte 3 états comme cela est indiqué dans le schéma ci-dessous.

Voici le schéma des portes qui pilotent les deux signaux s_clk et s_data.



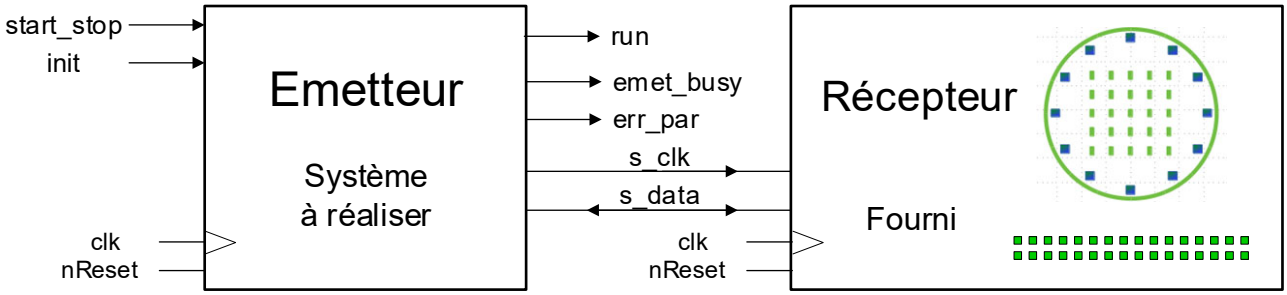
La ligne s_clk est unidirectionnelle. L'information circule toujours dans le même sens de l'émetteur vers le récepteur. Il s'agit d'une sortie pour l'émetteur et d'une entrée pour le récepteur.

La ligne s_data est bidirectionnelle. L'information circule dans les deux sens entre l'émetteur et le récepteur. Il est nécessaire de disposer d'une connexion bidirectionnelle (entrée & sortie) pour cette ligne. L'état '1' est obtenu par un état 'Z' et une résistance de polarisation vers le VCC (5 [V]). L'état '0' est obtenu par l'activation de l'état '0' par la porte 3 états.

2ère Partie : Application avec l'affichage des secondes

Dans cette seconde partie, nous allons réaliser une application utilisant l'émetteur série précédemment réalisé. L'objectif est de réaliser une horloge qui compte 60 secondes et qui allume les leds correspondantes de la carte Max-10_leds.

Voici une vue générale du système de l'horloge avec l'émetteur série synchrone et le bloc d'affichage comprenant un récepteur qui est fourni.



Entrées/sorties du système de l'horloge :

Port	Direction	Taille	Description
start_stop_i	entrée	1	Entrée permettant d'activer ou de désactiver l'horloge.
init_i	entrée	1	Signal d'initialisation de l'horloge : éteindre toutes les leds, remettre horloge à zéro et quittancer l'erreur.
run_o	sortie	1	Signal indiquant si l'horloge est activée (état '1').
emet_busy_o	sortie	1	Signal indiquant un transfert série en cours
err_par_o	sortie	1	Signal indiquant une erreur de parité lors d'un transfert en série.
s_clk_o *s_data_io*	sortie entrée/sortie	2	Signaux de la ligne série synchrone: • horloge de la ligne série. Fréquence 50 KHz • donnée série en bidirectionnel.
clk_i	entrée	1	Horloge du système synchrone, 1 MHz
nReset_i	entrée	1	Signal de remise à zéro asynchrone

L'horloge comprend 2 boutons de commande, soit :

- start/stop Une activation du bouton permet d'enclencher l'horloge. Une seconde activation déclenche l'horloge. L'état est affiché sur la sortie *run_o*.
- init Lorsque le bouton est actif, l'horloge est remise à zéro (tout est éteint) si l'horloge est en mode arrêt.

Voici la spécification pour l'affichage de l'heure sur l'horloge lorsqu'elle est enclenchée :

- À chaque seconde la led correspondante est allumée sur l'affichage.
- À t = 60 secondes, la led 60 est allumée et l'affichage reste ainsi pendant 1 seconde.
- À t = 1 seconde, toutes les leds doivent être éteintes et la led 1 seconde allumée.

Optionnel

- À t = 15, 30 et 45 seconde, la leds RGB des heures vis-à-vis est allumée avec la couleur bleu faible.
- À t = 60 seconde, la leds RGB de midi est allumée avec la couleur bleu faible. Les 3 autres leds RGB sont éteintes.

Nous allons utiliser la liaison série synchrone avec une donnée de 20 bits. L'affichage de la carte Max-10_leds utilise un mot de 20 bits permettant de piloter l'ensemble des leds disponibles sur celle-ci. L'annexe 2 vous donne comment est utilisé ce mot de 20 bits pour commander les différentes leds de la carte.

A rendre :

Ce laboratoire sera réalisé par groupe de 2 étudiants. Il comprendra une évaluation intermédiaire et un rendu final.

Evaluation orale intermédiaire : au début de la séance du 16 janvier 2020

Présenter l'organigramme évolué de votre émetteur (1ère partie) avec un schéma de l'unité de traitement (UT) montrant les différents blocs. Vous expliquerez vos choix de partition UC/UT.

Prévoir des copies papier pour professeur & assistant

Rendu final :

Vous devrez rendre à l'issu de ce travail de laboratoire:

- **UN SEUL fichier PDF** avec votre rapport de laboratoire comprenant l'ensemble de vos explications, étapes de conception, schémas, les preuves des vérifications réalisées, réponses aux différentes étapes et question demandées, ainsi que les descriptions VHDL synthétisable
- un fichier zip ou tar avec: scripts tcl, répertoires /src/… et /src_tb/…

Ces documents sont à déposer sur Cyberlearn, sur la page du cours.

Marche à suivre

Vous devez concevoir l'émetteur série permettant de transmettre une donnée de N bits via la ligne série synchrone. Ensuite vous réalisez le système, vous le simulez pour différentes valeurs du nombre de bits. Finalement, vous devez intégrer votre solution dans la carte Max-V 25-80 en utilisant le top *maxv_top_emet.vhd* pour un test de la transmission avec une donnée de 20 bits permettant d'allumer des leds sur la carte Max10_leds (voir annexe).

Vous devez commencer par établir la liste des étapes du projet. Vous vous baserez sur la méthodologie de conception des machines séquentielles synchrones complexes (MSS complexes) pour la conception de l'émetteur série. Vous devez établir un planning pour chaque étape y compris la réalisation de l'application de l'horloge. Ce planning doit être rendu au début de la séance du 9 janvier 2020.

Lors du déroulement du laboratoire, vous noterez les durées effectives. Vous ferez une comparaison entre les durées effectives et le planning à la fin du travail.

Votre conception doit pouvoir s'adapter aux valeurs définie dans le paquetage.

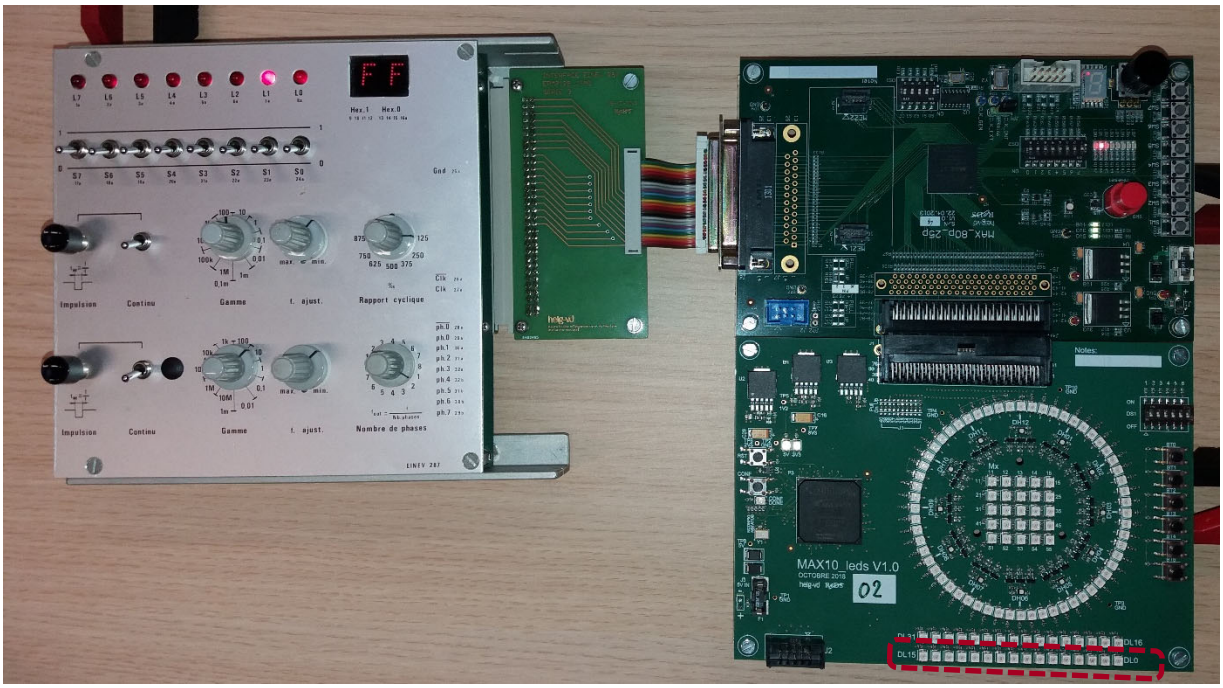
Une validation orale aura lieu au début de la séance du 16 janvier 2020 (voir: A rendre).

Vous devrez faire valider votre montage final par le professeur ou l’assistant.

Annexe 1

Montage pour le test de la 1ère partie : Emetteur série synchrone

Vue du montage pour le test de l'émetteur série synchrone avec la carte Max-10_leds :



Pour ce test, nous allons utiliser un transfert d'une donnée de 20 bits pour commander la ligne de 16 leds (DL15 à DL0) situées en bas de la carte Max-10_leds. Voici la description du codage des 20 bits transmis pour contrôler l'affichage sur les leds :

Bits	Fonction	Description
19 … 16	Code	Valeur "0100" fixé en dur dans maxv_top.vhd Ce code "0100" permet de piloter les leds DL15..0
15 … 0	Data	Donnée envoyée sur les leds sélectionnées par le code, soit dans notre cas les leds : DL15…0

Voici la liste des connections des entrées/sorties pour le test:

Signal	Carte Maxv_80p_25p	Description
clk_i	Oscillateur local, f = 1 MHz	Horloge du système
nReset_i	Bouton rouge	Remise à zéro asynchrone
start_i	Bouton SW1	Démarrage d'un transfert
data_i	"0100" & Switch & DipSwitch	Switch : boite de commande DipSwitch : carte MaxV
busy_o	Led(0)	Emetteur occupé
err_par_o	Led(1)	Erreur de parité

La connexion entre les deux cartes utilise le connecteur 80 pôles. Voici la liste des connections :

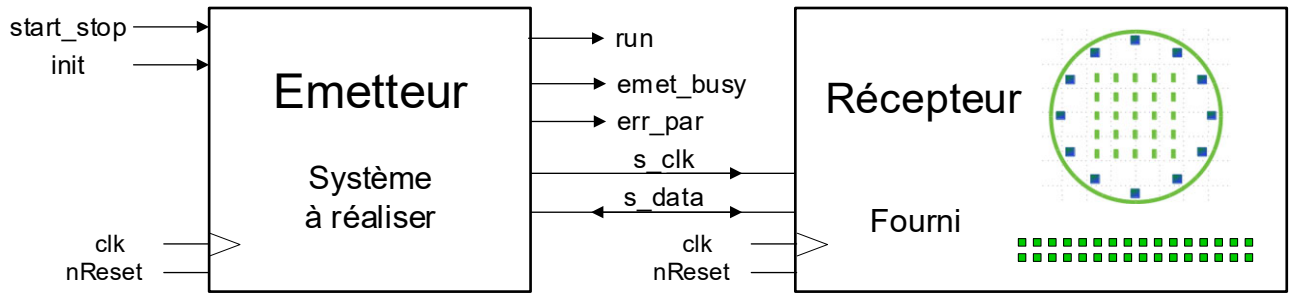
Signal	80p	Schéma	Maxv_top	Description
s_clk	30	R_Con_80p_30	Con_80p_io (30)	Horloge (output)
s_data	31	R_Con_80p_31	Con_80p_io (31)	Data (bidirectionnel) avec Rpull-up

La résistance de pull-up est montée sur la carte du récepteur, soit la carte Max-10_leds.

Annexe 2

Montage pour le test de la 2ème partie : Application pour une horloge

Schéma du montage pour le test de l'application pour une horloge avec la carte Max-10_leds :



Pour cette application, nous allons utiliser un transfert d'une donnée de 20 bits pour piloter l'affichage de la carte Max-10_leds. Voici la description du codage des 20 bits transmis pour contrôler l'affichage sur les leds :

Bits	Fonction	Description
19 … 16	Code	Sélection des leds contrôlées.
15 … 0	Data	Donnée envoyée sur les leds sélectionnées par le code

Ce codage de 20 bits permet de commander l'ensemble des leds de la carte Max-10_leds en utilisant différentes valeurs pour le code de contrôle. La table ci-dessous vous donne le codage utilisé :

Code	Spécification des bits de donnée D[15..0]
0000	D[15] not used, D[14..0] Leds secondes DS15...1
0001	D[15] not used, D[14..0] Leds secondes DS30...16
0010	D[15] not used, D[14..0] Leds secondes DS45...31
0011	D[15] not used, D[14..0] Leds secondes DS60...46
0100	D[15..0] Ligne leds DL15...0
0101	D[15..0] Ligne leds DL31...16
0110	D[15] not used, D[14..0] Carré de leds DM11-15/21-25/31-35
0111	D[15..10] not used, Carré de leds DM41-45/51-55
1000	D[15..10] Leds heures DH04-01 Int(3..0), 4 bits d'intensité par led
1001	D[15..10] Leds heures DH08-05 Int(3..0), 4 bits d'intensité par led
1010	D[15..10] Leds heures DH12-09 Int(3..0), 4 bits d'intensité par led
1011	Reserved
…	…
1111	Reserved