Auteur : Spinelli Isaia et

Bacso Gaëtan

Prof : [Etienne](https://cyberlearn.hes-so.ch/user/view.php?id=104149&course=1) Messerli

Ing : Sébastien Masle

Date : 26.01.2020

Salle : A09

Classe : CSN

Liaison série synchrone avec application

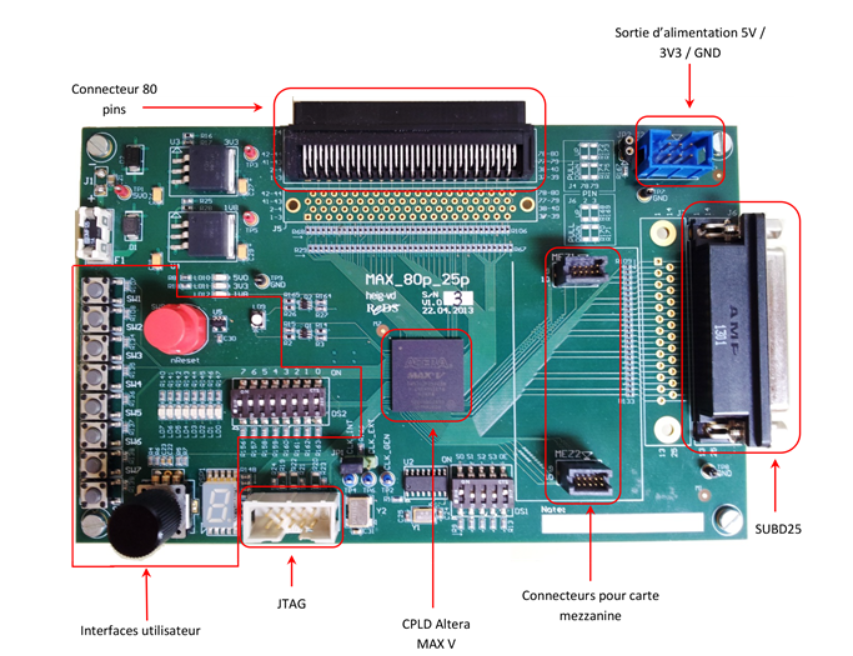
[CONCEPTION DE SYSTÈMES NUMÉRIQUES (CSN)](https://cyberlearn.hes-so.ch/course/view.php?id=14116#section-1) 

Table des matières

[Objectifs - 2 -](#_Toc30695114)

[Première partie - 2 -](#_Toc30695115)

[Analyse des différentes étapes - 3 -](#_Toc30695116)

[Réalisation de l’organigramme grossier - 4 -](#_Toc30695117)

[Réalisation de l’organigramme évolué - 5 -](#_Toc30695118)

[Réalisation du schéma UT - 7 -](#_Toc30695119)

[Réalisation du schéma UC (machine d’état) - 8 -](#_Toc30695120)

[Description VHDL des blocs - 9 -](#_Toc30695121)

[Regroupement des blocs - 9 -](#_Toc30695122)

[Synthèse et quantité logique - 9 -](#_Toc30695123)

[Vérification du fonctionnement - 10 -](#_Toc30695124)

[Test pratique de la communication - 11 -](#_Toc30695125)

[Deuxième partie : Application avec l'affichage des secondes - 12 -](#_Toc30695126)

[Analyse du système de fonctionnement - 13 -](#_Toc30695127)

[Conclusion - 14 -](#_Toc30695128)

[Difficultés rencontrées - 14 -](#_Toc30695129)

[Compétences acquises - 14 -](#_Toc30695130)

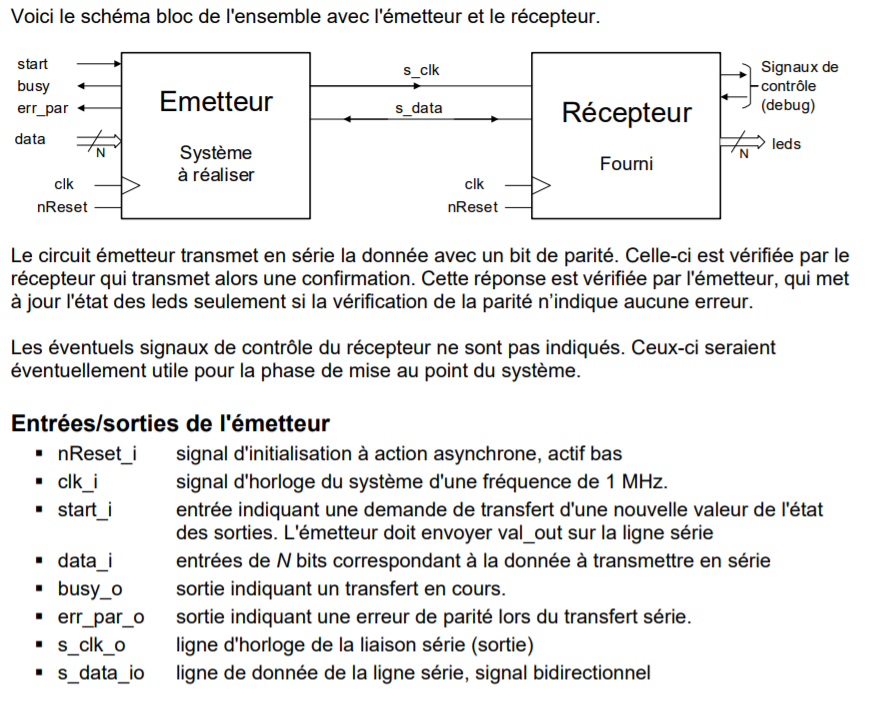
[Améliorations - 14 -](#_Toc30695131)

[Résultats obtenus - 14 -](#_Toc30695132)

## Objectifs

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. 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.

## Première partie



Pour avoir plus d’informations sur le système, les caractéristiques du protocole de transmission ou les caractéristiques électrique du bus série synchrone, veuillez vous référer à la donné du laboratoire 8 de CSN.

## Analyse des différentes étapes

Afin de s’exercer, il nous ait demandé de commencer par établir la liste des étapes du projet en nous basant sur la méthodologie de conception des machines séquentielles synchrones complexes (MSS complexes) pour la conception de l'émetteur série ainsi que pour la réalisation de l'application de l'horloge.

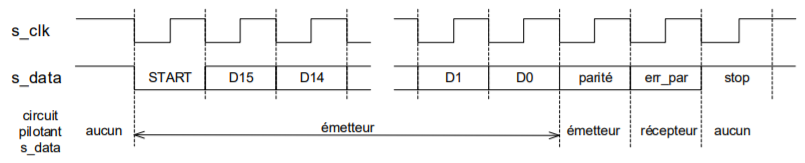


Grâce à ce planning en peut constater que nous avons mal estimé les temps d’analyse et de préparation qui effectivement prennent du temps. Cependant, une fois que tout a été réfléchis, la partie conception est bien plus rapide qui prévu, nous sommes particulièrement contents de ne pas avoir eu beaucoup de problème lors des tests.

## Réalisation de l’organigramme grossier

En fonction chronogramme du transfert d'une donnée de 16 bits fourni ci-dessous, nous avons établi un organigramme grossier réalisé par l’outil en ligne « planttext » :

Envoie du bit de stop

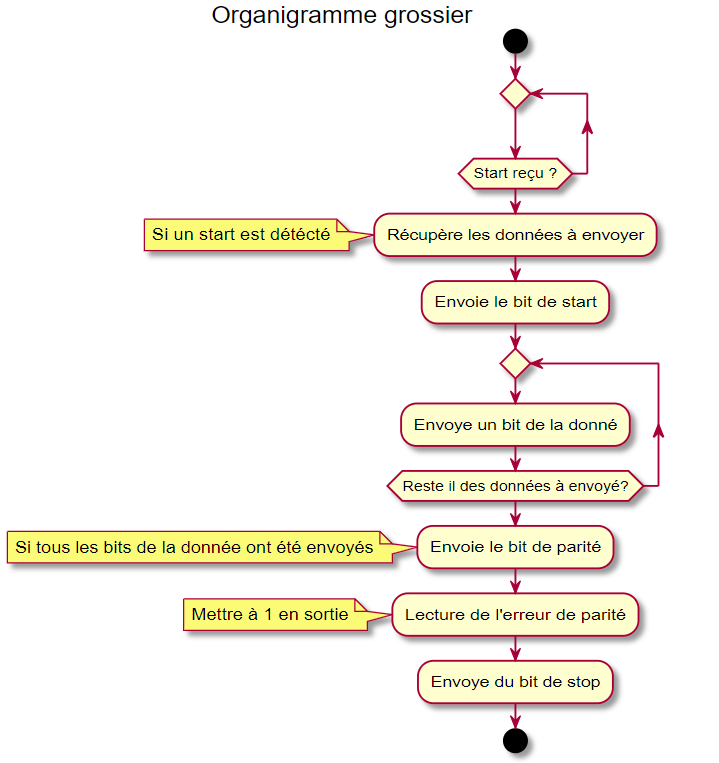


Mise à 1 de s\_clk

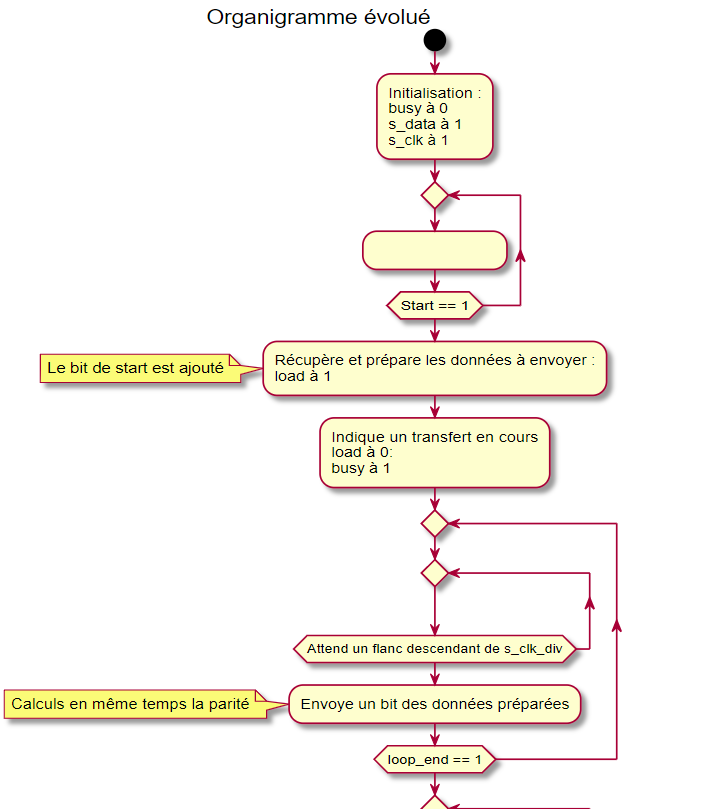
Lecture de l’erreur de parité

Envoie des data

Réception du start



## Réalisation de l’organigramme évolué

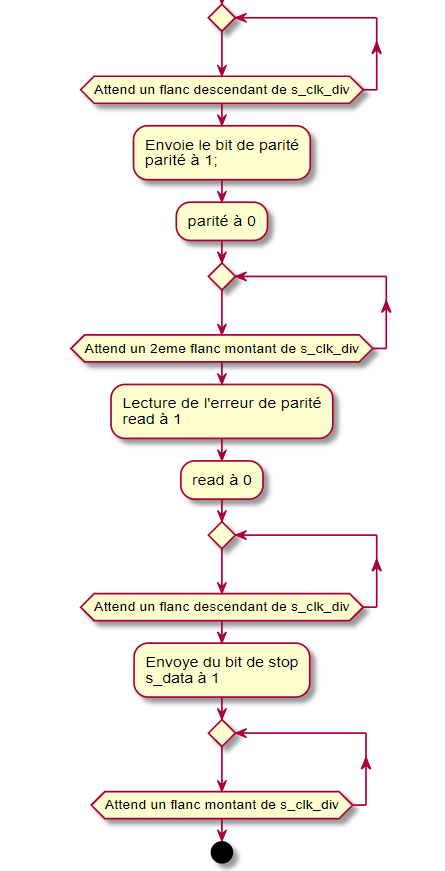


oui

non

oui

non



oui

Grâce à l’organigramme évolué, nous pouvons définir plusieurs blocs qui seront utile dans notre schéma d’unité de traitement dont :

* Un registre à décalage (load des données et envoie en série)
* Un compteur et une bascule T (Générer du s\_clk)
* Une porte XOR et une bascule (calcul de la parité)
* Bascule RS (Enregistrement l’erreur de parité lue)
* Des multiplexeurs (Gestion en fonction des signaux reçus de l’UC)
* Des comparateurs (Générer des résultats pour l’UC)

## Réalisation du schéma UT

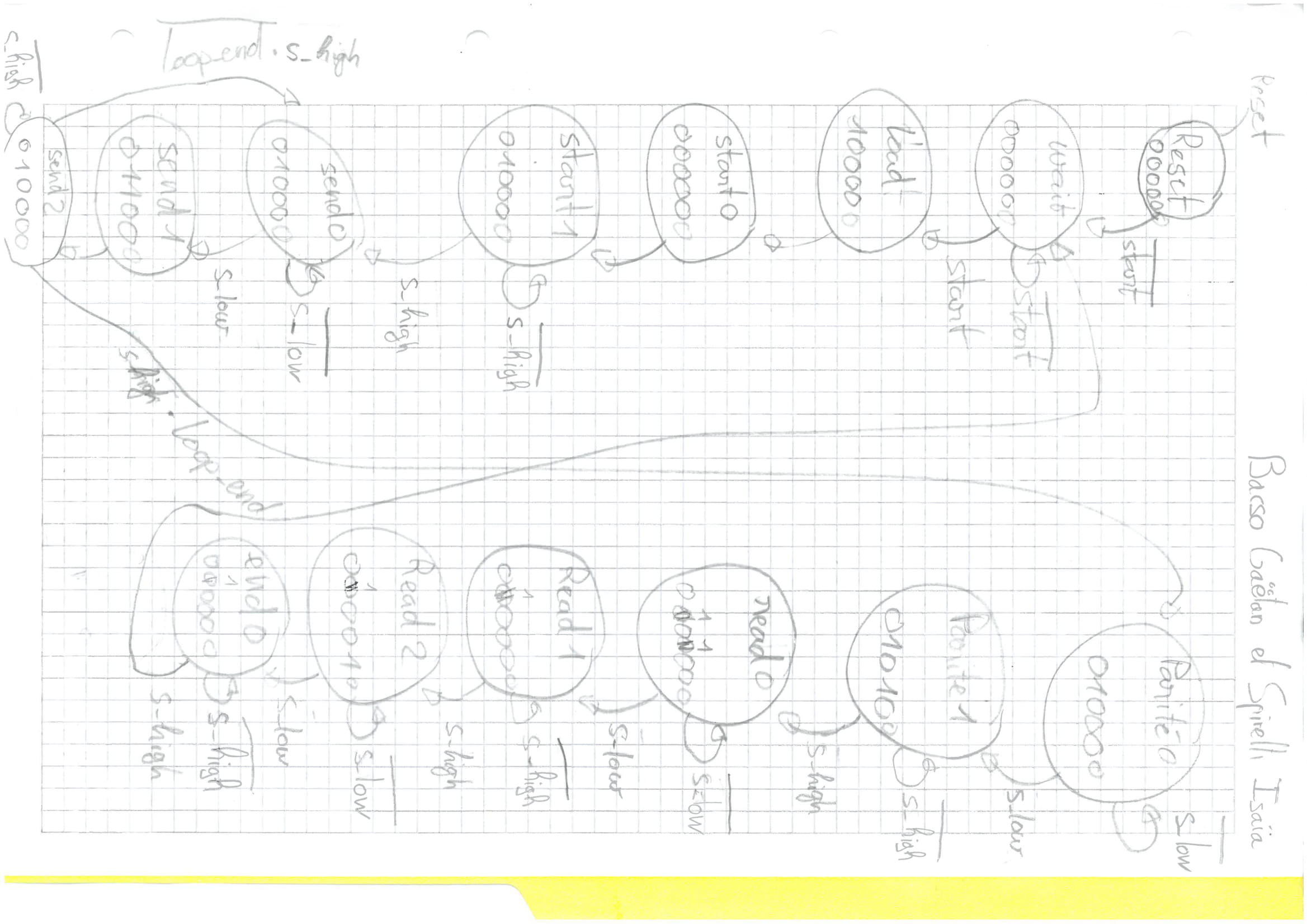
Maintenant que nous avons une meilleure idée de la conception de notre système, nous pouvons dessiner le schéma de l’unité de traitement :

## Réalisation du schéma UC (machine d’état)

Voici notre schéma d’unité de contrôle réalisé après le schéma d’unité de traitement :

Cond.

a = Load / b = busy / c = send / d = parité / e = read / f = end

*Remarque : Finalement, nous n’utilisons plus le signal end.*

*Remarque : Après avoir essayé de passer le test Bench, nous avons un peu modifié notre machine d’état afin d’envoyer la parité directement après avoir détecté « loop\_end ».*

## Description VHDL des blocs

Afin d’avoir une bonne structure du code, nous avons décidé de le répartir en plusieurs fichiers distincts :

* emet\_serie.vhd
* srgn.vhd
* cal\_parity.vhd
* divisor.vhd
* flipflop\_t.vhd
* dff\_en.vhd
* emet\_serie\_uc.vhd
* flipflop\_rs.vhd

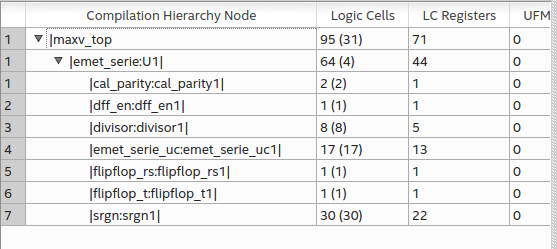
Toutes les descriptions VHDL sont en annexes.

## Regroupement des blocs

Une fois que toutes les descriptions VHDL ont été codées et assemblées, nous avons pu faire la synthèse afin de vérifier que ce composant est synthétisable ainsi qu’analyser la quantité logique obtenue.

### Synthèse et quantité logique

Nous avons fait la synthèse de notre système à l’aide du logiciel Quartus. Voici la quantité de logique obtenue pour une configuration de 20 bits d’envois :



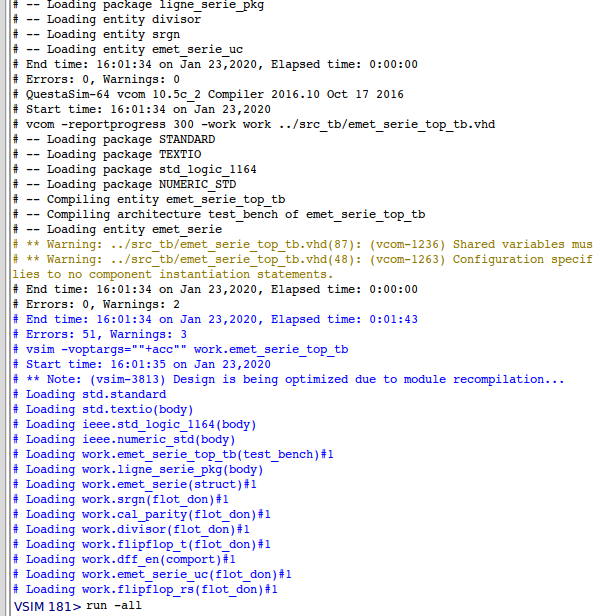
Ces valeurs correspondent bien aux quantités logiques auxquelles nous nous attendions.

### Vérification du fonctionnement

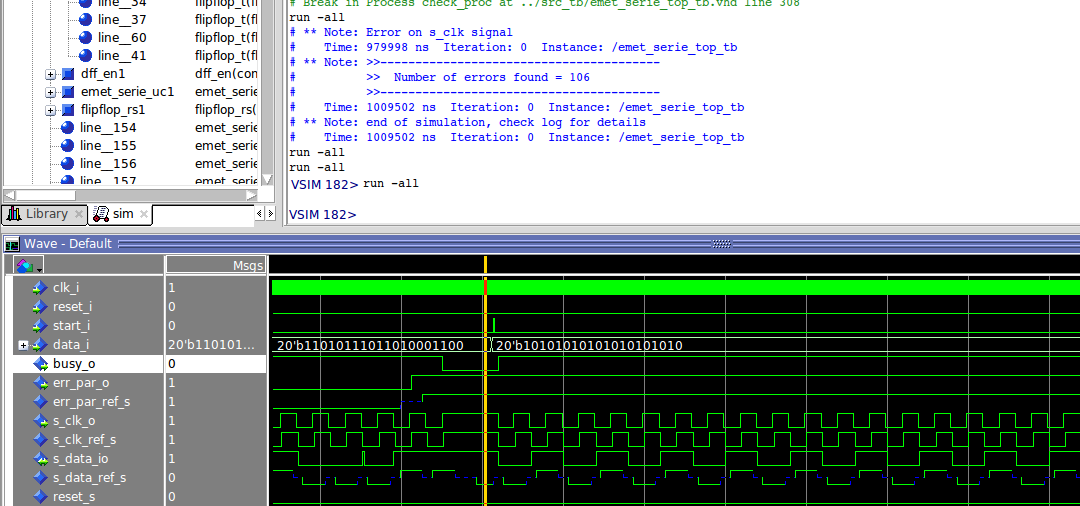
Maintenant, grâce au test Bench fourni, nous pouvons tester si notre émetteur est fonctionnel. Malheureusement le test Bench fourni nous a fait perdre énormément de temps car il comportait des erreurs. Nous remercions Etienne Messerli d’avoir passé beaucoup de temps à nous aider à corriger quelques erreurs.

Finalement, nous avons obtenue quelques erreurs car notre clock est décalée mais cela n’est rien d’important.

Voici les logs du lancement du test Bench.



Voici le résultat du test Bench :



Les erreurs déclarées sont dû à la clock qui est à peine décalée ainsi que la où le rectangle rouge ce trouve mais c’est le test Bench qui est dans l’erreur car il continue à générer un ‘0’ alors que la communication est finie.

## Test pratique de la communication

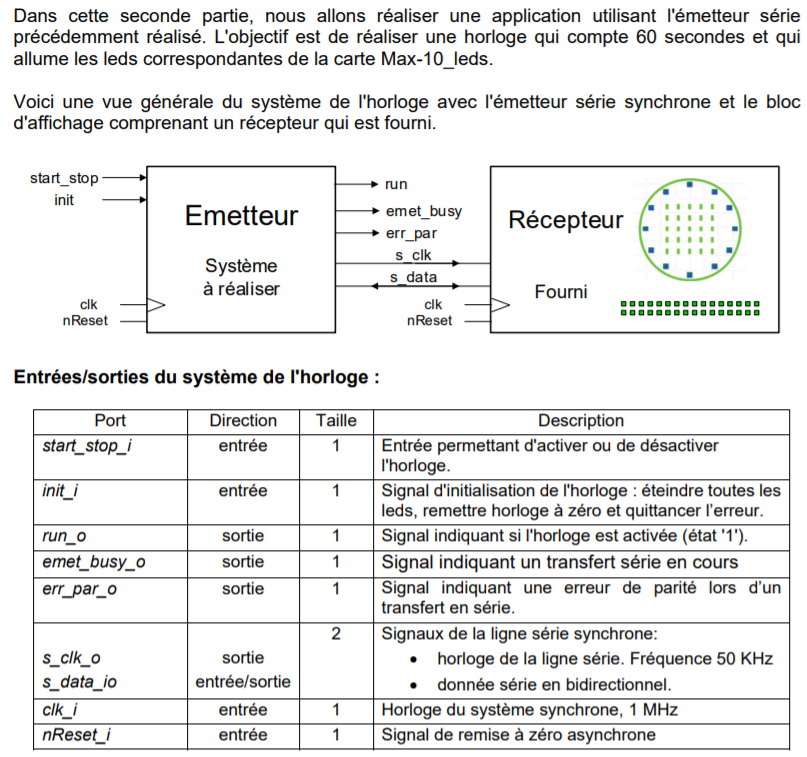
Une fois avoir fait l’intégration et le montage du système, nous avons pu tester directement sur le matériel notre communication :

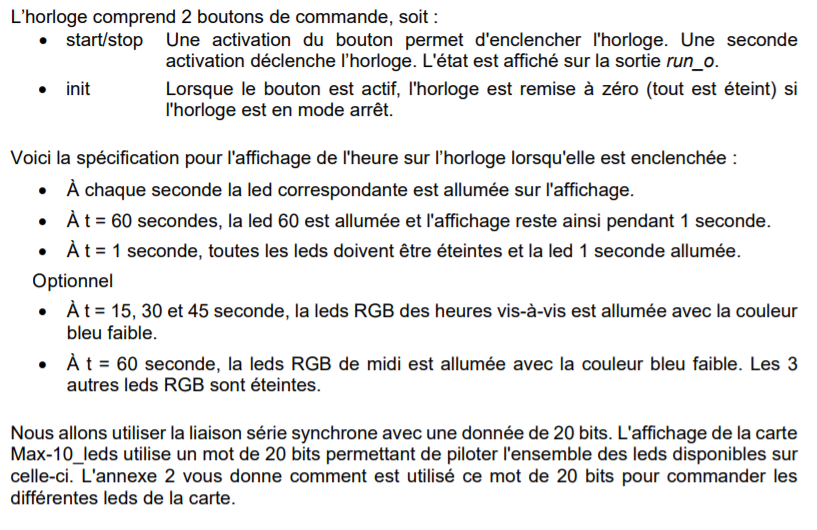
On peut constater que la communication fonctionne bien. Nous avons pu tester quasiment toutes les combinaisons possibles.

Au final, notre système a pu être validé par M.Masle le jeudi 23 janvier 2020.



# Deuxième partie : Application avec l'affichage des secondes





## Analyse du système de fonctionnement

## Conclusion

### Difficultés rencontrées

Au début nous avons eu de la difficulté pour déboguer notre système avec le test bench étant donné qu’il y avait quelques signaux. De plus, nous avons eu de la difficulté à faire une analyse correcte rapidement.

### Compétences acquises

Nous nous sommes familiarisés avec la méthodologie de travail ainsi qu’avec la conception de machine d’état complexe.

De plus, nous avons perfectionné nos compétences en analyse de système ainsi qu’en VHDL de manière générale.

### Améliorations

Nous avons séparé notre code en plusieurs fichier afin d’avoir une bonne structure. Cependant, il serait préférable d’implémenter un composant connu (exemple : registre à décalage) par fichier et de ne pas ajouter notre logique dans le même fichier mais en dehors.

### Résultats obtenus

Evidemment, tout n’a pas fonctionné du premier coup. Après quelques corrections, nous avons tout de même réussi à faire une communication correcte avec le récepteur et nous en sommes particulièrement fier.

Malheureusement, nous n’avons pas eu le temps de finir la partie 2 du laboratoire.

*Date : 26.01.20*

*Nom des étudiants : Bacso Gaëtan et Spinelli Isaïa*