SITE

Université d'Ottawa

CEG 3555
Systèmes numériques II

Rapport du projet
Conception d'un UART

Par :

Aya Chatiou 300203768
Yassine Moumine 300140139

Date: le 7/12/2022

Introduction:

Notre projet de fin de session consiste à concevoir un contrôleur de feu de circulation, adapté sur les mêmes caractéristiques que celui qu'on a conçu au laboratoire 4, quatre états, sauf que cette fois-ci notre contrôleur produit des messages de débogage. Pour réaliser ceci nous allons réaliser un UART en VHDL en se basant sur le principe récepteur-émetteur, ensuite nous réalisons le contrôleur UART, puis nous connectons les deux composantes aux capteurs et déclencheurs réelles.

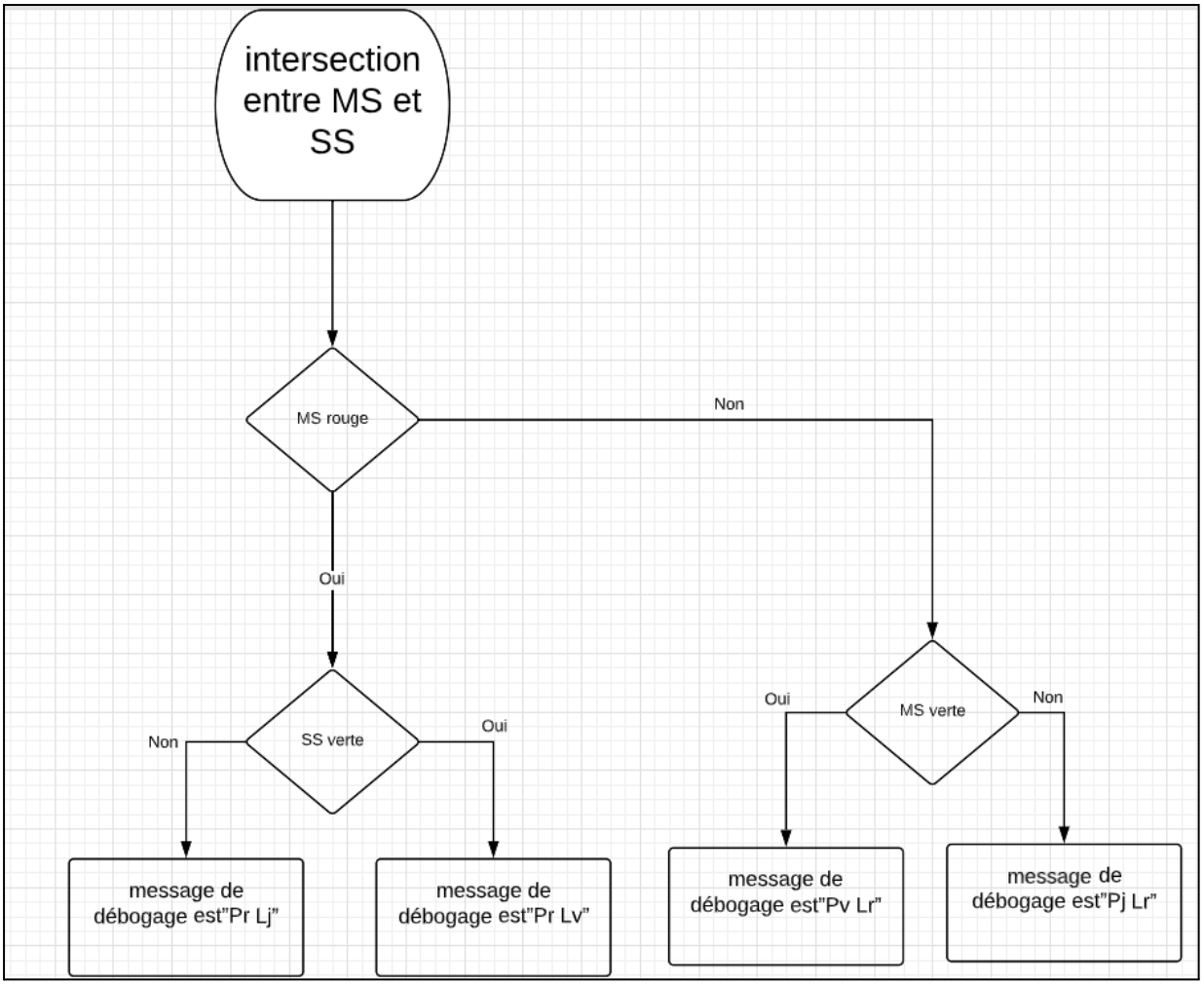
Objectifs:

- Apprendre à réaliser une conception d'un UART complet en VHDL.
- Concevoir, réaliser et examiner un module récepteur.
- Concevoir, réaliser et examiner un module émetteur.
- Concevoir, réaliser et examiner un module de générateur programmable de vitesse baud.
- Comprendre la conception d'un UART.

Problème:

Nous avons quatres états principaux:

1) rue **p**rincipale **verte**/rue **l**atérale **rouge** >> message de débogage est **"Pv Lr"**
2) rue **p**rincipale **jaune**/rue **l**atérale **rouge** >> message de débogage est **"Pj Lr"**
3) rue **p**rincipale **rouge**/rue **l**atérale **verte** >> message de débogage est **"Pr Lv"**
4) rue **p**rincipale **rouge**/rue **l**atérale **jaune** >> message de débogage est **"Pr Lj"**



Solution du problème:

Pour réaliser notre projet, nous utilisons principalement notre conception du labo 3 branchée avec un UART qui est un émetteur-récepteur asynchrone universel, il se compose évidemment d'un récepteur, un émetteur, un générateur de vitesse Baud et des registres UART, employé dans notre cas pour envoyer et recevoir des caractères ASCII des fils, pour faire fonctionner le UART correctement nous avons besoin d'ajouter un UART fsm qui simule la présence d'un microcontrôleur connectés par un bus de données et un autre d'adresse pour permettre au CPU de lire et écrire aux registres UART et permet d'envoyer les messages de débogages ainsi de les contrôler, pour rendre ces messages accessibles et lisibles nous utilisons un diviseur d'horloge qui fait en sorte de rendre l'horloge globale plus lente de façon que nous pouvons recevoir ces messages, ainsi nous ajoutons des décodeurs BCD employés pour démontrer visuellement la valeur courante d'un compteur sur un affichage duel-chiffre de BCD. De cette façon nous réussissons un système capable de résoudre le problème du projet contrôleur de feu de circulation fonctionne comme voulu.

Représentation d'organigramme de la solution du problème:

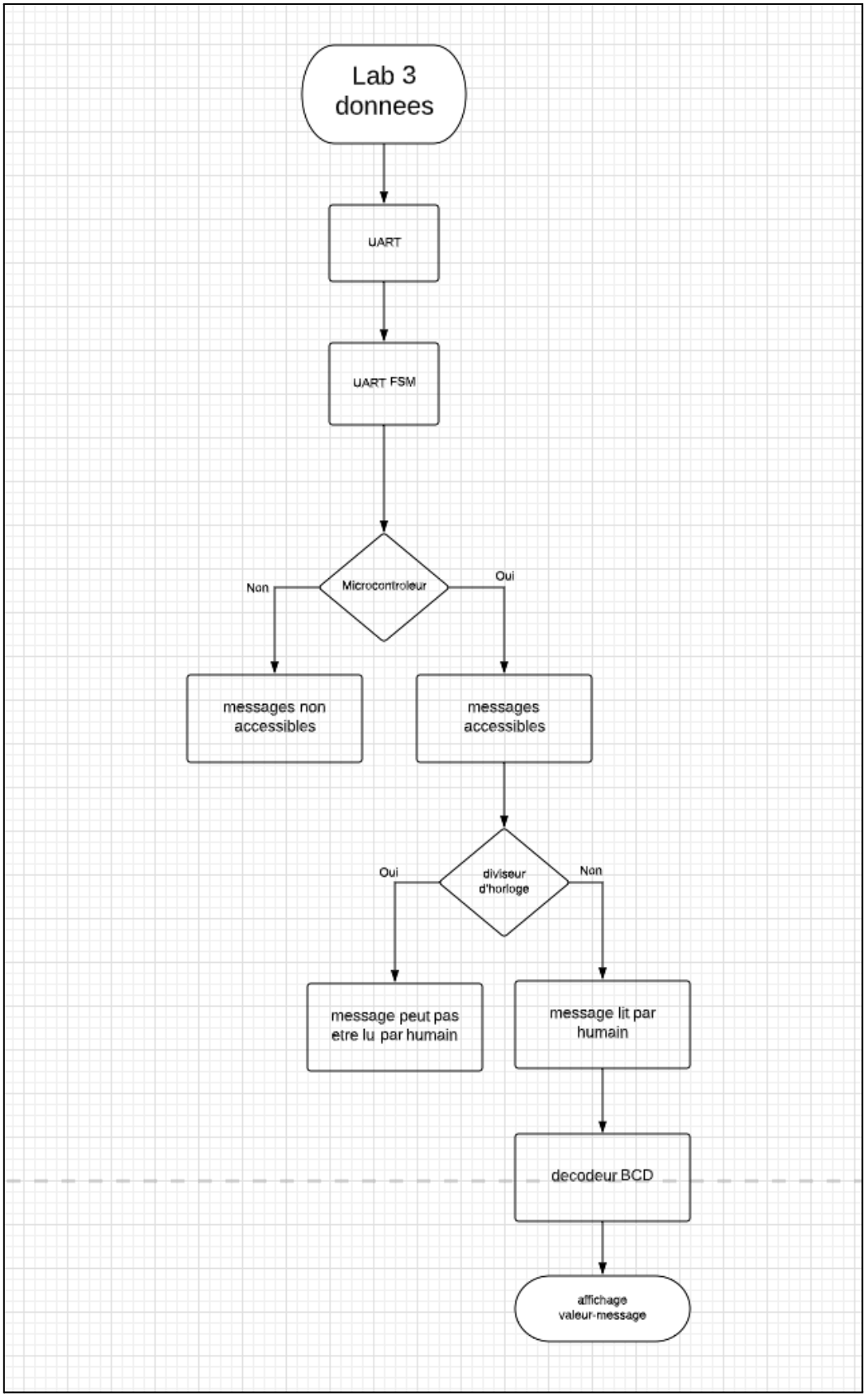
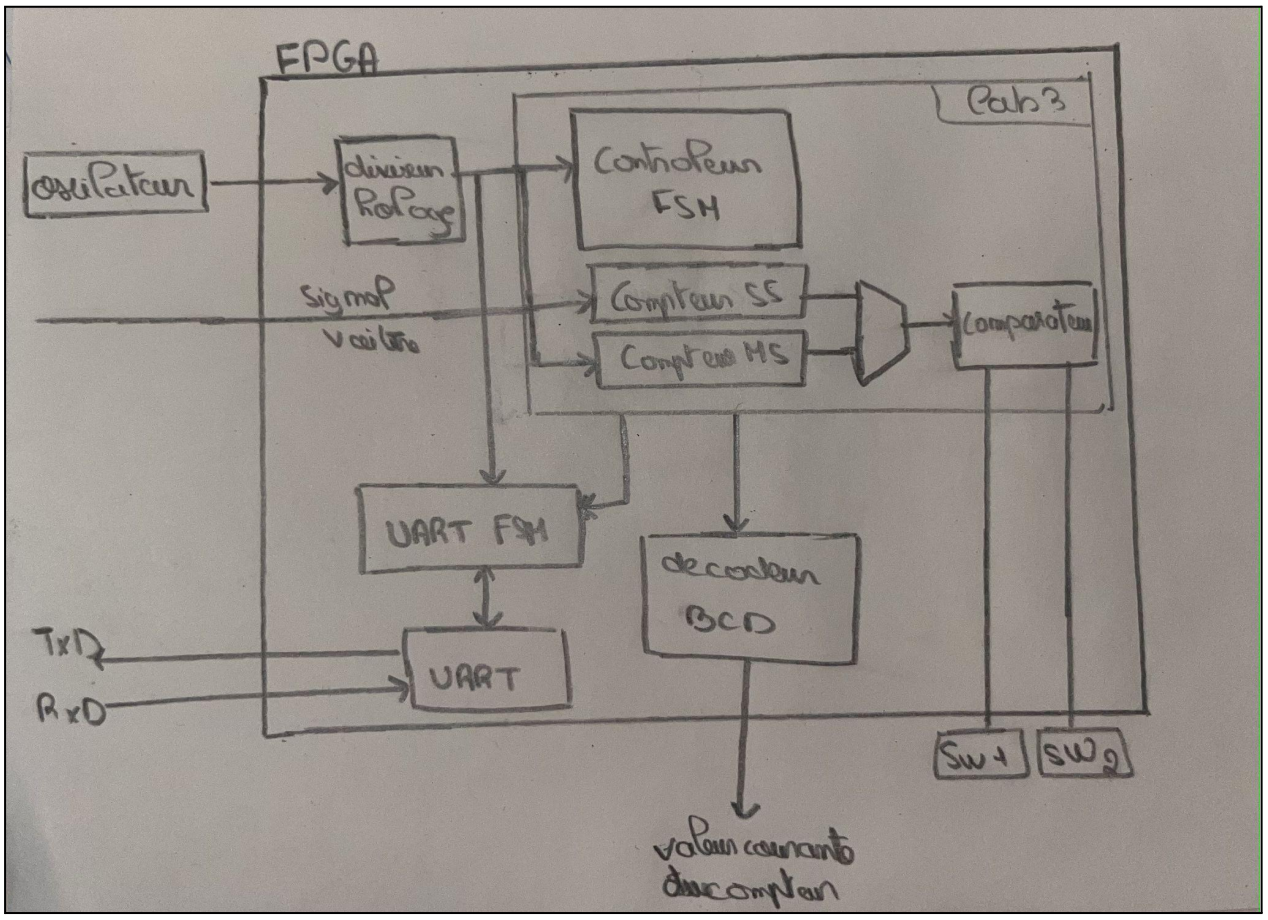


Schéma fonctionne de la solution adaptée:



Partie 1 : FSM du contrôleur de lab #3

Notre problème concerne le trafic à l'intersection entre MS et SS. la rue principale a une priorité élevée que la latérale, et reste ainsi verte jusqu'à ce qu'au moins une voiture sur la rue latérale arrive à l'intersection. Un capteur de voiture SSCS est présent sur la rue latérale pour un tel but. Le capteur produit un '1' a la présence d'une voiture sur la rue latérale et un '0' autrement. En addition de notre implémentation précédente, nous ajoutons le message de débogage correspondant à chaque étape. C'est alors que les spécifications peuvent être parfaitement résumé dans le pseudo-code suivant :

```
while(TRUE) {
        MSTL = VERT;
        Msg = "Pv_Lr";  // SSTL = RED, ÉTAPE A
                if(SSCS and MST)
                 {
                        MSTL = JAUNE AFTER MSC;
                        Msg = "Pj_Lr";      // ÉTAPE B
                        MSTL = RED AFTER MST;
                        SSTL = VERT;
```

```
                    Msg = “Pr_Lv”;      // ÉTAPE C
                    SSTL = JAUNE AFTER SSC;
                    Msg = “Pr_Lj”;      // ÉTAPE D
                    SSTL = RED AFTER SST;
                   // Lr
            }
}
```

Partie 2 : FSM du UART

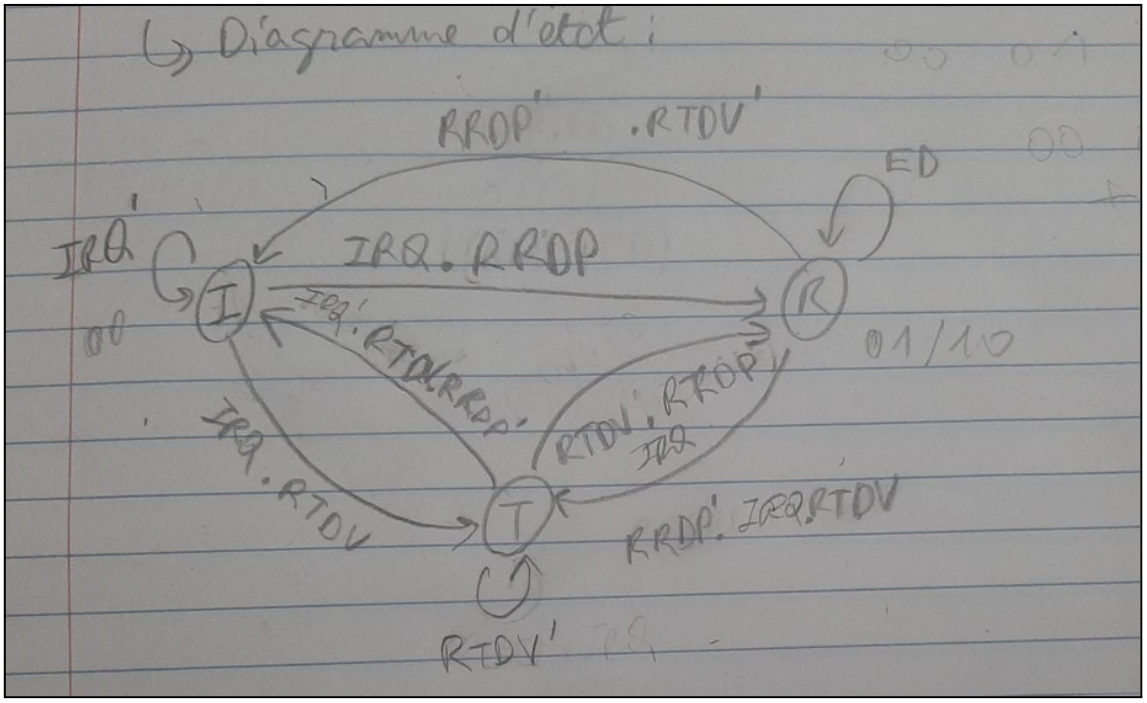
Ce composant simule un microcontrôleur et reçoit l’information d'état et interface au UART pour envoyer les messages de débogage à chaque changement d'état. Ce composant reçoit aussi les caractères du UART et décide d’inverser ou non les messages de débogage. D'où notre implémentation VHDL suivante :

```
   process(Bclk)
begin

   if infoEtat = "00" then
      Msg <= "Pv_Lr";     --ETAPE A
   elsif infoEtat = "01" then
      Msg <="Pj_Lr";      --ETAPE B
   elsif infoEtat = "10" then
      Msg <="Pr_Lv";      --ETAPE C
   elsif infoEtat = "11" then
      Msg <="Pv_Lj";      --ETAPE D
   end if;

end process;
```

Le UART est employé pour envoyer ou recevoir des caractères ASCII sur les fils TxD et RxD. Pour un fonctionnement parfait le UART, récepteur et émetteur exige une communication ininterrompable. Ceci dit des interactions enchevêtrées et asynchrones ou l’un ne perturbe pas l’autre. D'où nous se retrouvons avec différents cas de communication :

I > R > T > I
OU
I > T > R > I
OU
I > R > I
OU
I > T > I
I > T > I ET I > T > a

On déduit que:



Cependant pour le recepteur et emetteur nous avons le suivant :

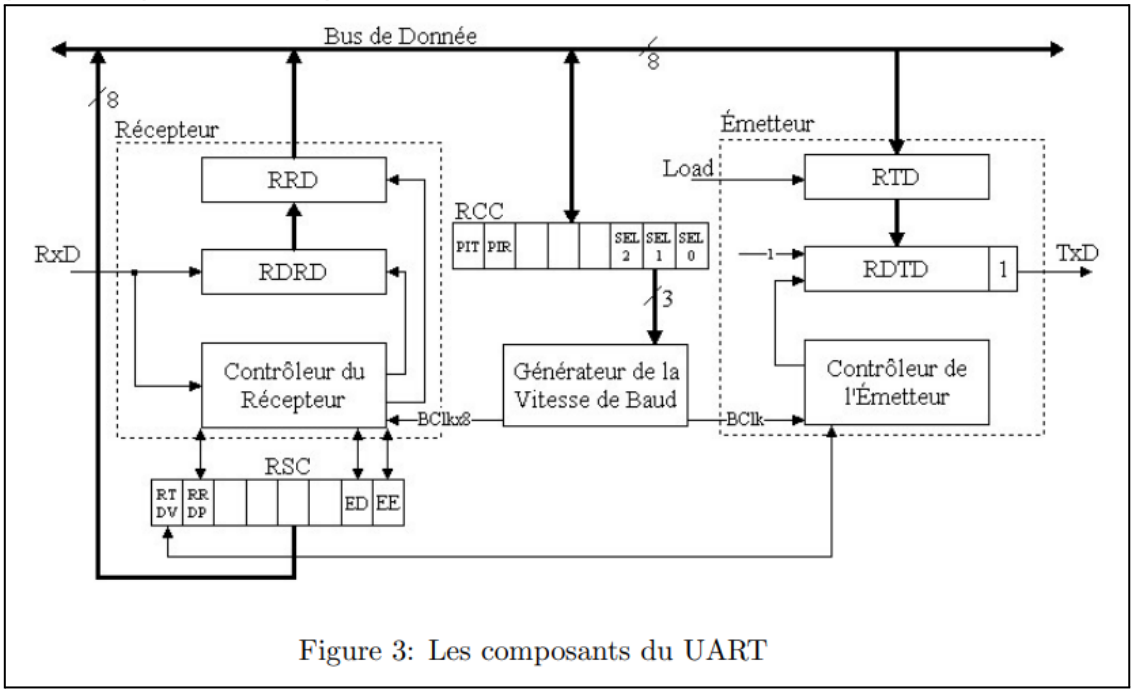


Figure 3: Les composants du UART

Partie 3 : FSM de l'émetteur

Lors d'une opération que ca soit l'émission ou la réception un signal **IRQ** est généré afin d'interrompre le CPU et signaler que l'un des deux opérations exige de l'attention. Comment différencier entre les deux opérations sera au niveau de FSM UART, comme vu à l'étape précédente.

Au niveau de la transmission, le signal **IRQ** est généré quand **PIT** et **RTDV** sont mis à 1. Donc :

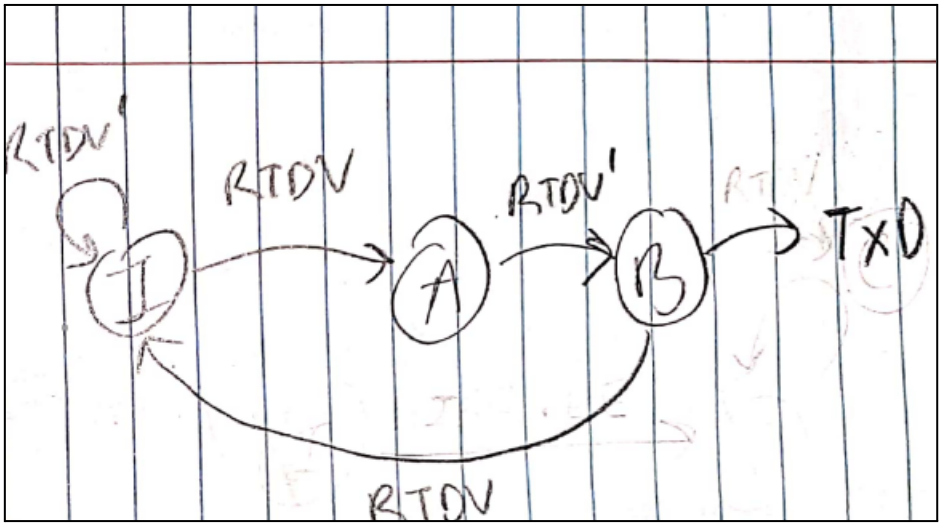
if(PIT == 1 AND RTDV) IRQ=1

L'opération d'émission peut se représenter en trois étapes principaux **(I,A et B)** :

I. Le microcontrôleur attend jusqu'à ce que **RTDV** = 1 et puis enregistre un byte de donnée dans **RTD**. **RTDV** est ensuite remis à 0.

A. Le UART transfère le byte du **RTD** au **RDTD** et mets **RTDV** = 1.

B. 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.



Nous pouvons résumer le tout :

RTD <= BUS DE DONNÉES;
RTDV = 0;
RDTD <= RTD;
RTDV = 1;
TxD <= 0 & RDTD & 1;

Partie 4 : FSM du récepteur

Cependant, au niveau de la réception, le signal **IRQ** est généré quand **PIT** est mis à 1 mais aussi si l'un des deux signaux **RRDP** ou **ED** est mis à 1. D'où :

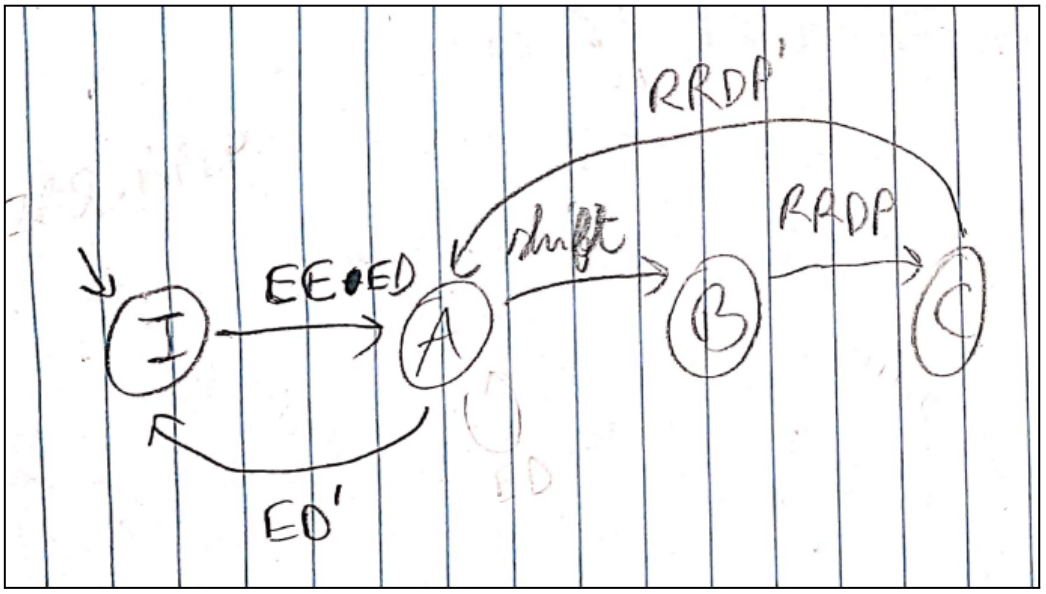
if(PIR == 1 AND (RRDP or ED)) IRQ=1.

L'opération de réception peut se représenter en quatres étapes principaux **(I,A,B et C)** :

I. Le UART lit, genre un signal **EE** au cas d'un débordement et décale les huit autres bits en série dans **RDRD**.

A/B. Quand les huits bits et le bit d'arrêt ont été reçus, la valeur de **RDRD** est enregistrée dans **RRD**, et le signal **RRDP** est mis à 1. Si le UART détecte une autre réception, il enregistre la nouvelle information dans **RDRD** et génère le signal **ED** (Overflow Error).

C. Le microcontrôleur vérifie le signal de statut, et s'il est mis, le **RDR** est lu et **RRDP** est remis à 0.



Ceci peut être résumé comme suit :

```
while(ED and EE)
{
        RDRD <= Décalage de RxD
        RRD <= 8BIT DE RDRD
        RRDP = 1;
        IF(RRDP) {
                BUS DE DONNÉES <= RRD;
                RRDP = 0;
        }
```

}

Les difficultés que nous avons eu se présentait au niveau de la présentation de tous les cas possibles et de présenter le parfait fonctionnement de chaque module, mais avec une bonne analyse du matériel de projet, du cours et des recherches additionnelles nous avons réussi ceci.

Conclusion:

À la fin de ce projet qui achève notre cours , nous avons conçu un module de circuit d'une machine séquentielle synchrones pour un contrôleur de feu de circulation basé sur le laboratoire présent, avec des modifications sur le fonctionnement et les tâches . Ceci a été possible et réalisable en suivant toutes les étapes de la méthodologie des machines à états finis (FSM). Non seulement cette expérience, mais celle qui la précède aussi, nous a appris à bien traduire un problème technique à une conception logique suivant différentes méthodologies, ainsi nous avons appris à réaliser la conception d'un UART sur VHDL en examinant aussi des nouveaux modules : récepteur, émetteur et générateur programmable de vitesse baud. En addition, nous avons appris l'importance d'un diagramme d'un système fonctionnelle afin de simplifier la difficulté des spécifications du problème et leurs compréhension mais aussi de faciliter la conception VHDL.