Unité de transmission/réception série asynchrone version avec bit de parité

Avertissement

Ce document est le document de référence. Certaines notations et schémas peuvent différer dans les transparents de présentation.

1 Introduction

Une UART, Universal Asynchronous Receiver/Transmitter, est un contrôleur d'entrées/sorties qui gère les ports série des ordinateurs. Il en existe différents modèles : les 8250 (8 bits), les 16450 (16 bits) et la famille des 16550 (16 bits + FIFO). L'UART, présente aussi sur les modem, permet la communication entre le modem et le PC. L'objectif est l'implantation d'une UART 8 bits simplifiée avec **bit de parité**.

2 Description de l'UART (à développer)

2.1 Interface

L'UART que nous allons développer possède l'interface présentée en Figure 1.

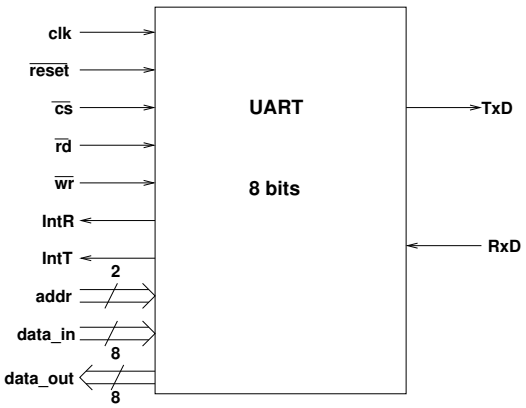

Figure 1 – Interface de l'UART

```
entity UARTunit is
  port ( clk, reset : in  std_logic;
         cs, rd, wr : in  std_logic;
         RxD        : in  std_logic;
         TxD        : out std_logic;
         IntR, IntT : out std_logic;
         addr       : in  std_logic_vector(1 downto 0);
         data_in    : in  std_logic_vector(7 downto 0);
         data_out   : out std_logic_vector(7 downto 0) );
end UARTunit;
```

2.2 Spécifications

L'UART est composée d'un registre de contrôle `ctrlReg` et de 4 unités fonctionnelles :

- `clkUnit`, cette unité décompose `clk`, l'horloge de base du circuit (celle du processeur), en deux horloges nommées `enableTX` et `enableRX` qui contrôlent respectivement l'émetteur et le récepteur ;
- `ctrlUnit`, cette unité contrôle l'état de l'UART ;
- `TXUnit` est l'unité d'émission ;
- `RXUnit` est l'unité de réception.

La FIGURE 2 montre la spécification interne du circuit UART (attention : toutes les connexions ne sont pas représentées).

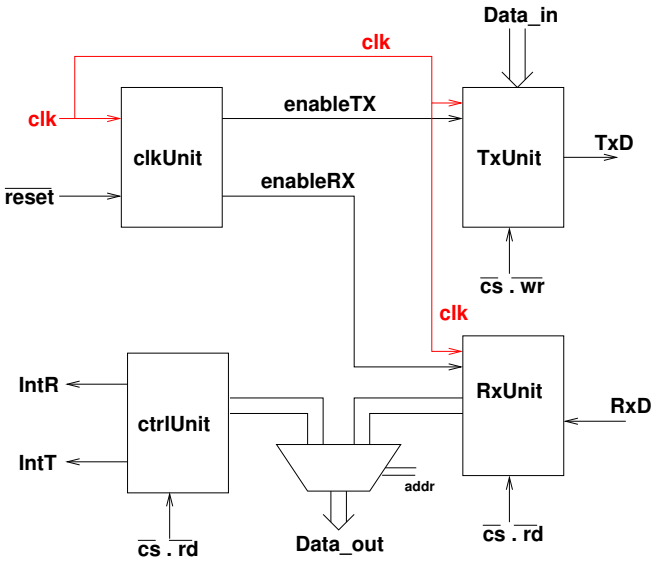

FIGURE 2 – Spécification interne de l'UART

Fonctionnement de l'UART :

- Lorsque le processeur (CPU) reçoit l'interruption `IntT=0`, l'UART est prête à émettre. Le processeur positionnera `cs=0` et `wr=0` pour émettre le caractère (octet) donné à l'entrée `data_in` ;
- Lorsque le CPU reçoit l'interruption `IntR=0`, un caractère a totalement été reçu par l'UART. Pour le récupérer, le CPU positionne `cs=0` et `rd=0` ainsi que `addr=00` afin de lire le registre de réception de l'unité `RxUnit`. Le caractère reçu est placé sur `data_out`.
- À tout moment, le CPU peut connaître l'état de l'UART en lisant son registre de contrôle : `addr=01`, `cs=0` et `rd=0`. La valeur du registre de contrôle sera placée sur `data_out`.

3 L'unité d'horloge (à développer)

3.1 Interface

```
entity clkUnit is
 port ( clk, reset : in  std_logic;
        enableTX   : out std_logic;
        enableRX   : out std_logic );
end clkUnit;
```

3.2 Spécifications

Cette unité a pour objectif de découper la fréquence de l'horloge de base `clk` (celle du processeur) en deux horloges :

- une pour l'émission, `enableTX`, de fréquence 9.6 kHz ;
- une pour la réception, `enableRX`, de fréquence 155 kHz.

Par souci de simplification, on fait l'hypothèse que l'horloge en entrée `clk` est de fréquence 155 kHz, même fréquence que l'horloge de réception.

L'horloge de réception, `enableRX`, est 16 fois plus rapide que l'horloge d'émission `enableTX`.

De cette manière, il sera possible à l'unité de réception de contrôler l'asynchronisme de la communication [1]. Ce comportement est donné par la Figure 3.

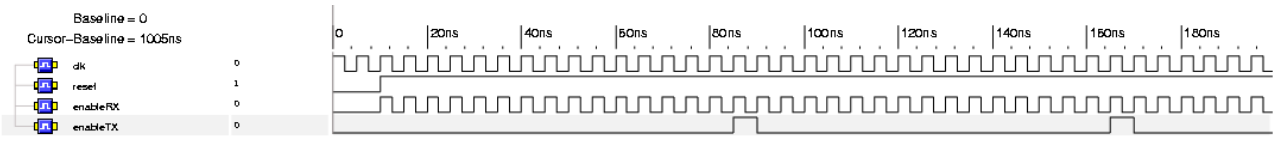

Figure 3 – Chronogramme de l'unité d'horloge

1. voir présentation

4 L'unité d'émission (à développer)

4.1 Interface

```
entity TXUnit is
  port ( clk, reset : in std_logic;
         enable     : in std_logic;
         ld         : in std_logic;
         txd        : out std_logic;
         regE       : out std_logic;
         bufE       : out std_logic;
         data       : in std_logic_vector(7 downto 0) );
end TXUnit;
```

4.2 Spécifications

Cette unité transmet la donnée positionnée sur `data` bit à bit sur le support de communication via la sortie `txd`. Le format utilisé est de type RS232 avec 1 bit de stop et **avec** parité.

L'unité d'émission utilise 2 registres : un registre tampon et un registre d'émission (respectivement notés `BufferT` et `RegisterT`, T pour *Transmission*). L'état de ces registres est donné par les sorties `bufE` et `regE` (E pour *Enable*) :

- `bufE=1` : le tampon est vide ; `bufE=0` : le tampon est occupé ;
- `regE=1` : le registre d'émission est vide ; `regE=0` : le registre est occupé.

Initialement, `txd=1`, i.e. la ligne au repos est à l'état haut. L'émission est rythmée par le signal `enable` et se déroule en 6 étapes :

1. Attente d'une demande d'émission : on sort de cette attente lorsque l'entrée `ld` est activée (`=1`) ; la donnée positionnée sur l'entrée `data` est alors chargée dans le tampon ;
2. On charge le contenu du tampon dans le registre d'émission ;
3. On lance l'émission par l'envoi du bit de start : `txd=0` ;
4. On émet les 8 bits de données contenues dans le registre d'émission du poids fort au poids faible ;
5. On émet le bit de parité (xor entre les 8 bits de données) ;
6. L'émission se termine par le bit de stop : `txd=1`.

Ces étapes ne sont pas rythmées par la même horloge [2].

Lors d'une transmission, si le tampon est vide, le "processeur" peut demander à émettre une seconde donnée. Ainsi, dès que la transmission en cours est terminée, le TxUnit recommence à partir du point 2 pour transmettre cette seconde donnée.

La Figure 4 donne le contenu de l'unité d'émission ainsi que le lien entre les différents éléments. Et la Figure 5 montre un exemple d'émission d'un caractère.

2. voir explications + traduction en VHDL

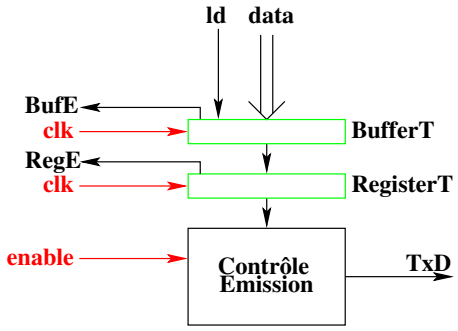

Figure 4 – Schéma de l'émetteur

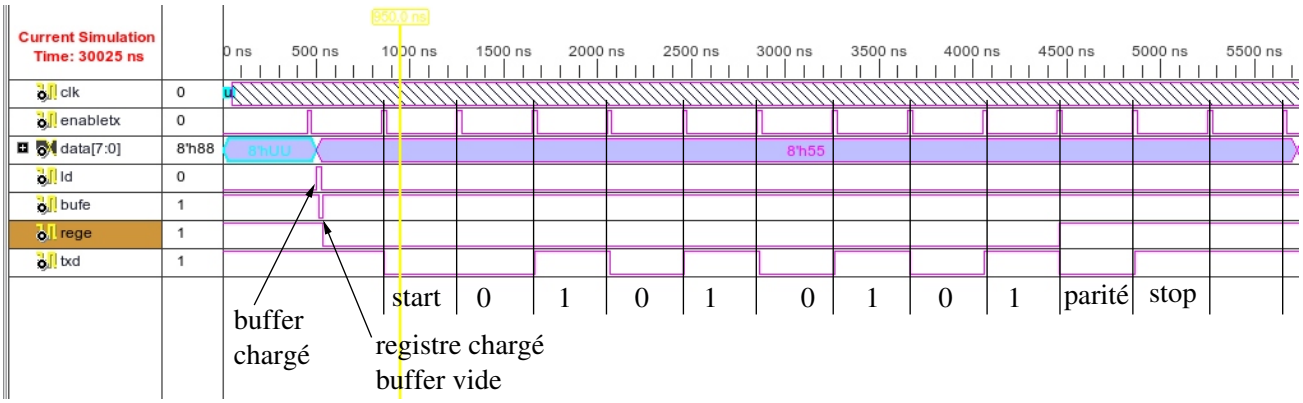

Figure 5 – Transmission du caractère 'U' (code ASCII : 85 (dec), Ox55 (hex))

5 L'unité de réception (à développer)

5.1 Interface

```
entity RxUnit is
  port ( clk, reset        : in  std_logic;
         enable            : in  std_logic;
         read              : in  std_logic;
         rxd               : in  std_logic;
         data              : out std_logic_vector(7 downto 0);
         FErr, OErr, DRdy  : out std_logic );
end RxUnit;
```

5.2 Spécifications

Cette unité réceptionne la donnée arrivant bit à bit sur l'entrée `rxd` et la transmet au processeur via la sortie `data` lorsque la réception est terminée et que le processeur positionne le signal `read`. Cette unité fournit aussi 3 signaux :

- `DRdy=1` lorsqu'une donnée est reçue.
- `FErr=1` si la trame reçue est erronée : le bit de parité ou(et) le bit de stop est(sont) erroné(s) ;
- `OErr=1` si lorsqu'une donnée est prête, elle n'est pas lue à temps par le processeur.

Le fonctionnement de l'unité de réception est rythmé par les signaux `enable` pour la réception et `clk` pour la communication avec le processeur. Il est constitué de 2 parties :

- un élément qui gère les instants de réception ;
- un élément qui gère les états de l'unité de réception.

Le premier élément est cadencé par `enable` (horloge de réception) et attend le bit de start. Quand celui-ci est détecté, cet élément va commencer à compter afin de générer l'horloge `tmpClk` qui aura la fréquence de l'horloge d'émission et dont les fronts montants seront "au milieu" des bits reçus.

Le principe, pour générer `tmpClk`, est donc de compter 8 tops de `enable` dès que l'on détecte le bit de start puis de compter 16, autant de fois qu'il le faut, jusqu'à la réception complète de la trame.

Le rôle du deuxième élément est de construire la donnée et de vérifier sa bonne réception en récupérant un bit à chaque front de `tmpClk` :

- le bit de start ;
- les 8 bits de données, du poids fort au poids faible ;
- le bit de parité ;
- le bit de stop.

À la fin de la réception de la trame, plusieurs cas peuvent se présenter :

1. Si le bit de stop est erroné, i.e. égal à zéro, ou le bit de parité est incorrect, on avertit le processeur en positionnant `FErr=1` ;
2. Sinon, on positionne `DRdy=1` et la donnée en sortie. Au front montant suivant de `clk`, on rabaisse `DRdy` et
   (a) Si `read` est à zéro, il y a une erreur de transmission : `OErr=1`, le "processeur" n'a pas lu la données reçue,
   (b) Si `read` passe à un, tout s'est bien passé, le processeur a lu la donnée reçue.

La Figure 6 représente les composants de la partie récepteur.

La Figure 7 montre la réception normale du caractère 'U' (Ox55), la Figure 8 montre une erreur de transmission, la Figure 9 montre une erreur de parité (les deux situations activent le signal `FErr`) et enfin, la Figure 10 un non-positionnement du signal de lecture et l'émission du signal `OErr`.

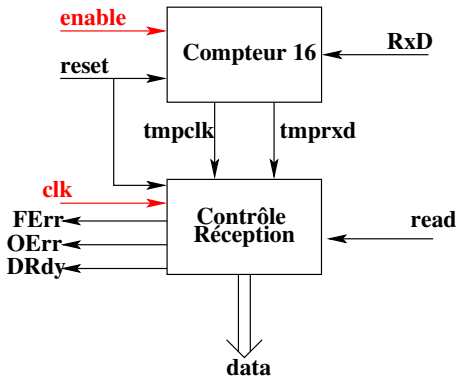

Figure 6 – Schéma du récepteur

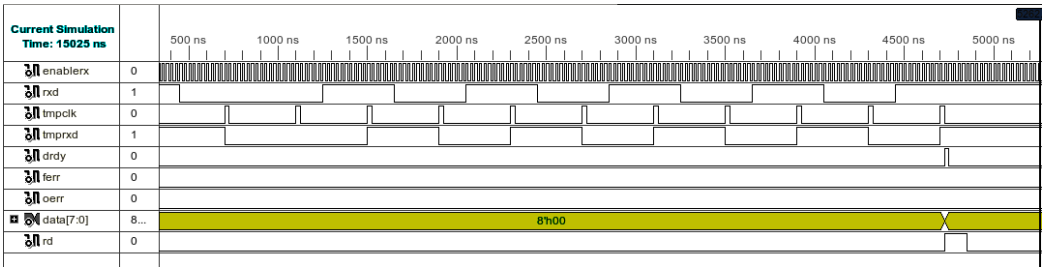

Figure 7 – Exemple de la réception normale

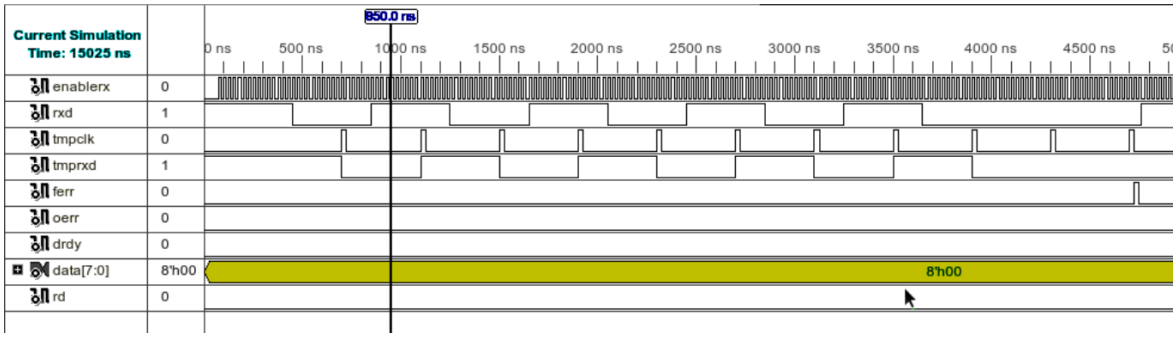

Figure 8 – Exemple de réception erronée : bit de stop faux

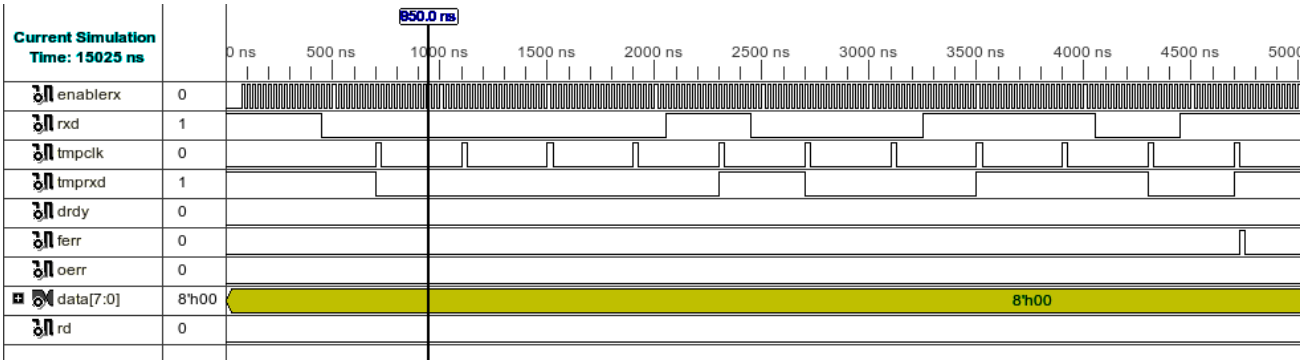

Figure 9 – Exemple de réception erronée : bit de parité faux

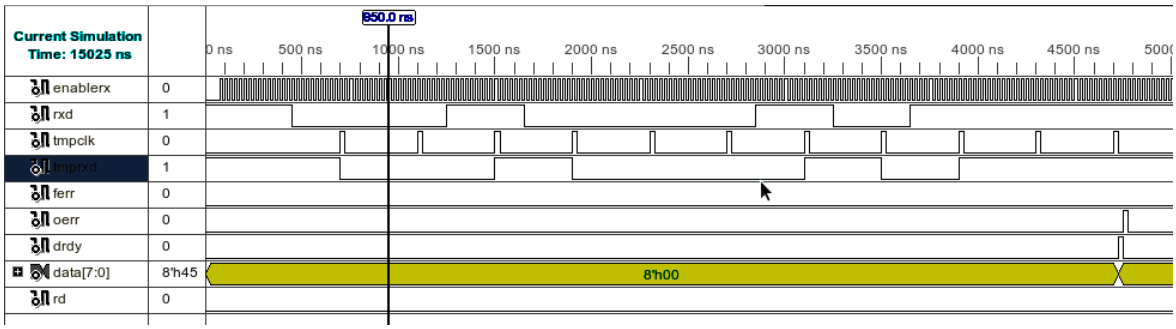

Figure 10 – Exemple de non lecture par le processeur : il ne positionne pas read

6 L'unité de contrôle (fournie)

6.1 Interface

```
entity ctrlUnit is
  port ( clk, reset        : in  std_logic;
         rd, cs            : in  std_logic;
         DRdy, FErr, OErr  : in  std_logic;
         BufE, RegE        : in  std_logic;
         IntR              : out std_logic;
         IntT              : out std_logic;
         ctrlReg           : out std_logic_vector(7 downto 0) );
end ctrlUnit;
```

6.2 Spécifications

L'unité de contrôle fournit les signaux d'interruption `IntR` et `IntT` indiquant respectivement si une donnée a été reçue ou non et s'il est possible de transmettre une donnée. Elle gère aussi les données contenues dans le registre ctrlReg donné FIGURE 11.

Si une donnée est disponible, `DRdy=1`, `IntR=0`. `IntR` sera remis à 1 lorsque la lecture de la donnée est effectuée.

Lorsque le buffer ou le registre d'émission est libre (`BufE=1` ou `RegE=1`), il est possible de transmettre une donnée : `IntT=0`. Sinon, l'écriture d'une donnée dans le buffer ou le registre d'émission désactive l'interruption : `IntT=1`.

Gestion du registre de contrôle ctrlReg :
- le champs TxRdy indique que l'émetteur est prêt à émettre une donnée ;
- le champs RxRdy indique que le récepteur a reçu une donnée ;
- les champs FErr et OErr correspondent aux erreurs de transmission obtenues par l'unité de réception ;
- les champs réservés, notés `Res`, sont toujours à 1.

Res	Res	Res	Res	TxRdy	RxRdy	FErr	OErr

FIGURE 11 – Le registre de contrôle regCtrl.