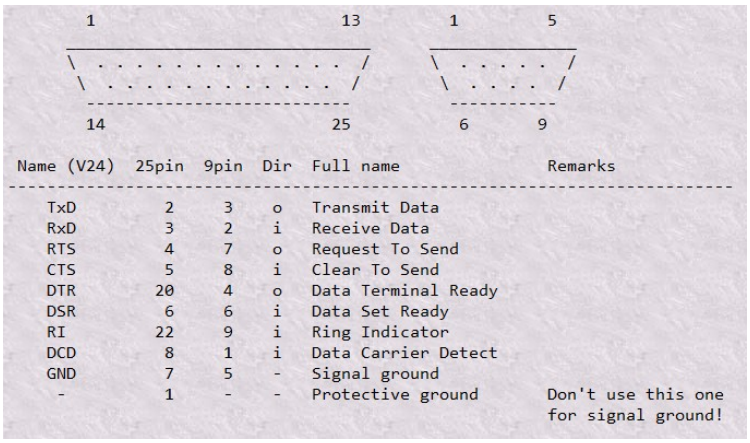
Fiche TD/TP 1

L'objectif est synthétiser l'IP UART (Universal Asynchronous Receiver/Transmitter). Le contrôleur de communication UART est utilisé pour envoyer une donnée parallèle bit par bit sur un bus. Deux modules sont nécessaires pour implémenter le protocole : le module émission TX et le module réception RX.

La figure ci-après donne la description des connecteurs RS232 souvent utilisés avec le contrôleur UART.



Name (V24)	25pin	9pin	Dir	Full name	Remarks
TxD	2	3	o	Transmit Data	
RxD	3	2	i	Receive Data	
RTS	4	7	o	Request To Send	
CTS	5	8	i	Clear To Send	
DTR	20	4	o	Data Terminal Ready	
DSR	6	6	i	Data Set Ready	
RI	22	9	i	Ring Indicator	
DCD	8	1	i	Data Carrier Detect	
GND	7	5	-	Signal ground	
-	1	-	-	Protective ground	Don't use this one for signal ground!

I- Module d'envoi

Le module TX permet de transmettre une donnée sur 8 bits sur la liaison série bit par bit selon le protocole imposé. Une opération de sérialisation est donc effectuée afin de pouvoir transmettre la donnée bit par bit. Une trame série est composée d'un bit de start, un bit de stop et 8 bits de données.

La vitesse de transmission est de 9600 bits par secondes pour une horloge du FPGA de 50 Mhz.

Chaque bit prend donc 104 µs pour être transmis. La fréquence de l'horloge étant de 50 Mhz, la période de l'horloge est de 20 ns, ce qui implique qu'il faut 5200 tops d'horloge pour transmettre un bit. On utilisera pour cela deux compteurs : un compteur modulo 16 et un compteur modulo 325. On obtient ainsi un compteur global de 16*325 = 5200.

1) Donner le code VHDL d'un compteur qui permet d'indiquer la fin de la transmission d'un bit à travers sa sortie fin_bit et la fin de transmission d'une trame à travers sa sortie fin_trame. L'entrée reset est une entrée asynchrone d'initialisation du comptage tandis que reset_counter est entrée d'initialisation synchrone. Le compteur peut être réalisé en faisant

le montage de trois compteurs en cascade : un compteur module 325, un compteur modulo 16 et un compteur modulo 10.

```
entity counter is port (
        reset_counter,reset,clk: in std_logic;
        fin_bit,fin_trame : out std_logic
);
end entity;
```

2) Donner la machine à état modélisant le comportement du module d'envoie d'une trame UART.
Le module d'envoie est supposé posséder la partie entity suivante :

```
entity send is port (
        start,reset,clk: in std_logic;
        TX : out std_logic
);
end entity;
```

L'entrée synchrone start permet d'initier l'envoie d'un octet.

3) Donner la description VHDL du module d'envoi d'une trame UART. Puis, réaliser sa simulation à l'aide de ModelSim sans écrire un fichier de banc de test.

II- Module de réception

La réception d'une trame UART s'effectue bit par bit selon la même vitesse de transmission. Ainsi, le module de réception est censé identifier l'arrivée de bit start de la trame pour commencer la réception effective de la donnée. Une opération de dé-sérialisation est effectuée afin de construire l'octet à partir des bits données reçus.

1) Donner la machine à états du module de réception et sa description VHDL. Pour simplifier, on pourra utiliser le compteur décrit pour le module d'envoi sans sa sortie fin_trame. Vérifier le fonctionnement de ce module en réalisant sa simulation.

2) Réaliser un module qui permet d'afficher la valeur d'un octet sur deux afficheurs 7 segments de la carte CYCLONE II.

3) Modifier la réalisation pour afficher l'octet reçu sur les afficheurs 7 segments.

4) Tester la réalisation en effectuant une communication entre deux cartes CYCLONE II.