1. Objectif

Le but de ce premier TP est de vous familiariser avec le langage Verilog en modélisant un contrôleur mémoire RS232. Dans le cadre du cours de Vérification ce contrôleur sera ensuite vérifié grâce à un test bench écrit en System Verilog.

Au cours de ce TP vous allez :

- étudier la spécification du contrôleur mémoire RS232
- concevoir la machine à états faisant l'interface entre la mémoire et la connexion RS232
- mettre en œuvre les bases du Verilog
- concevoir un test bench Verilog pour tester le fonctionnement de base du contrôleur

2. Architecture du contrôleur mémoire

Il s'agit ici de modéliser un contrôleur mémoire basé sur le protocole série RS232.

Le contrôleur RS232 utilisé sera configuré pour gérer des trames de 8 bits de données avec 1 bit de STOP, 1 bit de START et 1 bit de parité. Les transmissions seront de type half-duplex, c'est à dire que les émissions et réceptions seront effectuées séquentiellement.

L'interaction avec la mémoire s'effectue en envoyant les commandes 8 bits décrites ci-dessous :

Fonction	Mot 1	Mot 2	Mot 3	Mot 4 (status)
écriture	0xC0 + ~~adresse[13:8]~~	adresse[7:0]	donnée[7:0]	0x55 : OK 0xAA : ERREUR
lecture	0x80 + ~~adresse[13:8]~~	adresse[7:0]	donnée[7:0]	0x55 : OK 0xAA : ERREUR
effacement	0x40	0x5A	donnée[7:0]	0x55 : OK 0xAA : ERREUR
protection	0x05	0x12 : lecture seule 0x34 : écriture ON	0x56 : lecture seule 0x78 : écriture ON	0x01 : lecture seule 0x02 : écriture ON 0xAA : ERREUR

Pour **écrire**, la procédure consiste à envoyer en 3 mots la commande d'écriture, l'adresse et la donnée. En retour le contrôleur envoie soit OK si l'écriture s'est bien passée, soit ERREUR si la mémoire est en lecture seule ou occupée.

Pour **lire**, la procédure consiste à envoyer en 2 mots la commande de lecture et l'adresse. En retour le contrôleur envoie la valeur lue puis OK si la valeur est valide ou ERREUR si la valeur est invalide (contrôleur occupé).

Pour **effacer** la mémoire, la procédure consiste à envoyer 2 mots de commande puis la valeur qui

sera inscrite. En retour le contrôleur envoie soit OK si l'effacement est possible soit ERREUR si la mémoire est en lecture seule ou occupée. Le contrôleur envoie une interruption de fin d'effacement sur un port dédié une fois la mémoire entièrement effacée (voir la liste des ports ci-dessous)

Pour modifier la **protection** de la mémoire, la procédure consiste à envoyer 1 mot de commande puis 2 mots de contrôle. En retour le contrôleur envoie la nouvelle protection ou un code d'erreur si la commande est invalide. Par défaut la mémoire est en lecture seule. Un changement de protection pendant un effacement est possible. La nouvelle protection est mise à jour pour les prochaines opérations.

Les commandes de lecture, d'écriture et de modification de la protection sont effectuées instantanément. En revanche la commande d'effacement est prise en compte instantanément mais nécessite 2^{14}=~~16384~~ cycles pour être effective. Pendant l'opération d'effacement, toute nouvelle commande sera ignorée et le code d'erreur devra être retourné.

Le module est nommé **RS232_memory** et a pour interface les ports suivants :

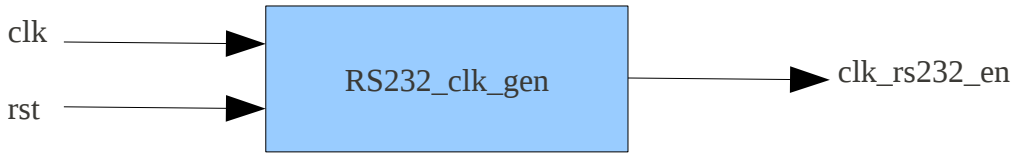
nom	taille (en bit)	direction	description
clk	1	input	horloge interne du contrôleur
rst	1	input	reset
rx	1	input	RS232 – ligne de réception
tx	1	output	RS232 – ligne de transmission
end_of_erase	1	output	Interruption de fin d'effacement
PARITY	1	paramètre	0: parité paire 1: parité impaire
RS232_RATIO	~~8~~	paramètre	Diviseur de l'horloge interne pour générer l'horloge de l'interface RS232

Le contrôleur indique la fin d'effacement en montant pendant 1 cycle de clk le signal end_of_erase.

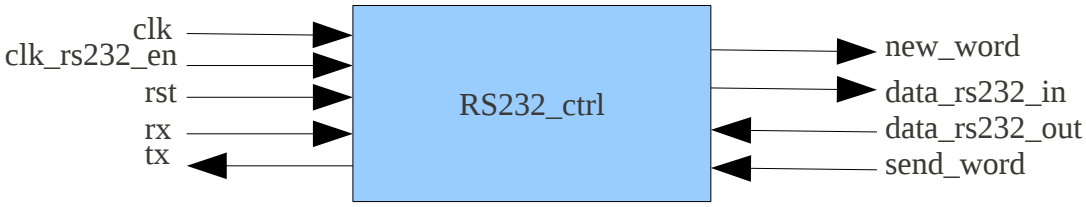
3. Implémentation

L'architecture du module RS232_memory sera composée des sous modules suivants :

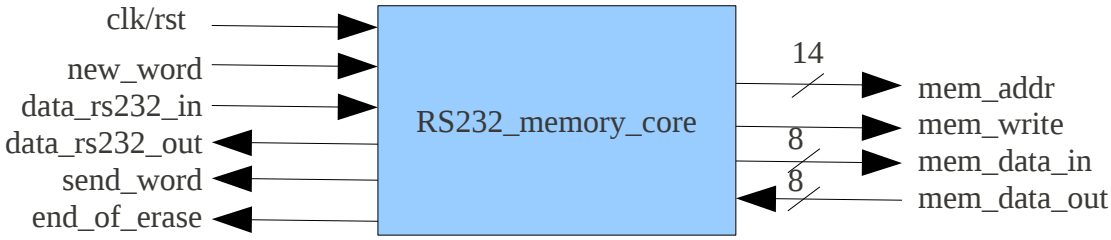
- RS232_clk_gen : génère un signal « clk_rs232_en » servant à diviser l'horloge du coeur « clk » suivant la valeur de RS232_RATIO



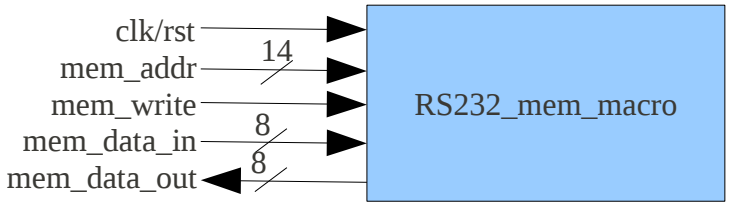
- RS232_ctrl : sert d'interface entre l'interface RS232 et le contrôleur de mémoire



- RS232_memory_core : décode les commandes transmises par RS232_ctrl puis effectue les actions vers la mémoire.



- RS232_mem_macro : contient le modèle de mémoire.



Dans le cadre d'un projet coopératif, une telle architecture permet aux designers de se répartir le travail. Elle a également l'avantage de permettre le test individuel des sous blocs.

4. Travail demandé

1. Implémenter chaque sous-bloc en Verilog « synthétisable ». Un soin particulier devra être apporté aux commentaires, à l'indentation et à l'utilisation matérielle.
2. Tester la fonctionnalité de chaque module séparément avec un test bench Verilog.
3. Écrire le module RS232_memory instanciant tous les sous-blocs.
4. Tester la fonctionnalité du composant RS232_memory avec un test bench Verilog.