1 Introduction

Dans ce mini-projet vous devez concevoir un système à base de μ-processeur en réutilisant des blocs de composants de base. Les objectifs de ces TP sont multiples. L'objectif premier est de réactualiser vos connaissances sur le VHDL, et de les renforcer en introduisant les concepts de réutilisation et de généricité. Enfin une introduction à Verilog est prévue.

Dans un premier temps vous travaillerez sur le code d'une RAM simple décrite en VHDL, vous la rendrez générique, puis vous créerez un bloc contenant plusieurs instances de cette RAM. Enfin, l'objectif sera d'intégrer ce bloc à un système comprenant également un processeur dont le code est écrit en Verilog.

Enfin il faudra simuler l'ensemble afin de démontrer la bonne intégration des différents éléments dans le système.

2 Récupération d'un modèle de RAM

Le point de départ du travail est un modèle simple de mémoire. Ce modèle fonctionnel n'est pas synthétisable mais permet d'émuler une mémoire simple.

Dans un premier temps récupérez le modèle ramchip.vhd et son testbench sur le site de la société Doulos :
http ://www.doulos.com/knowhow/vhdl_designers_guide/models/simple_ram_model/

1. Analyser le système récupéré ainsi que son testbench.
2. Utilisant le type *std_logic_vector()* ce modèle ne suit plus la norme VHDL et il convient donc de les remplacer par un type *unsigned()*.
3. Simuler le système pour vérifier que la modification n'a pas changé son fonctionnement.
4. On souhaite rendre ce bloc générique sur la largeur des données et sur le nombre de données mémorisables. Proposez une solution et implémentez la.

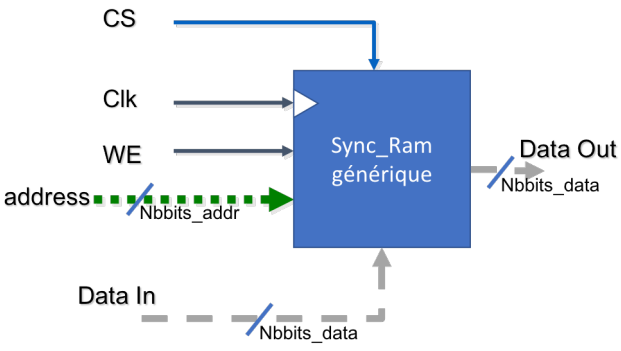

FIGURE 1 – Bloc mémoire générique attendu.

5. Lors de la suite du projet nous réutiliserons ce bloc de base pour créer des mémoires plus grosses en connectant entre eux plusieurs instances de Sync_Ram. Il faut alors ajouter un signal *CS* asynchrone permettant de mettre à l'état '*Z*' les sorties de la mémoire.

Vous devez, après avoir vérifié par simulation l'ensemble des fonctionnalités, obtenir le bloc Sync_Ram générique comme présenté sur la Figure 1.

3 Réutilisation du bloc Sync_Ram

L'objectif est maintenant de créer un composant mémoire de plus grande capacité contenant N=4 instances de la RAM décrite précédemment. Les signaux d'entrée /sortie sont communs aux quatre blocs, sauf le signal '*CS*' qui doit maintenant être généré par un décodeur prenant en entrée les bits de poids fort du bus d'adresse, qui ont été rajoutés à cet effet. Le système attendu pour N=4 est représenté sur la Figure 2.

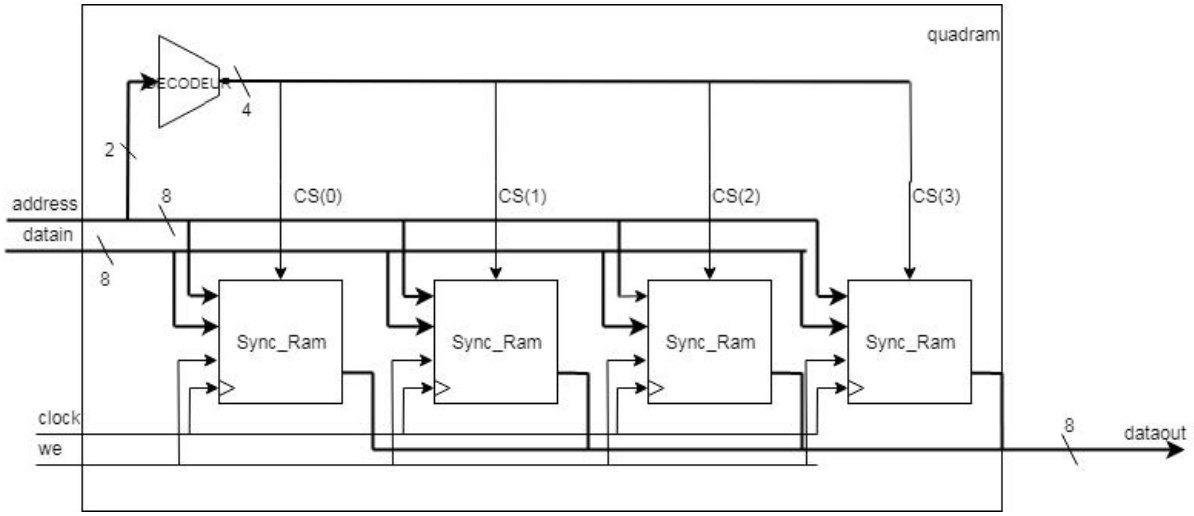

FIGURE 2 – Bloc mémoire final instanciant N=4 blocs Sync_Ram. On remarque que puisque N=4 le décodeur d'adresse utilise 2 bits en entrée.

1. La première étape consiste à créer un décodeur d'adresses. Simuler ce bloc avant son utilisation.
2. Créer le "*top entity*" *quadram* qui :
   (a) inclue 4 blocs Sync_Ram et votre décodeur d'adresses,
   (b) Valider cet ensemble.
3. Rendre le bloc *quadram* générique sur le nombre N de Sync_Ram en utilisant la structure *for ... generate* de VHDL et en rendant votre décodeur d'adresses générique.

4 Prise en main d'un modèle de processeur

Une fois que le bloc mémoire est fonctionnel on souhaite l'utiliser avec un processeur simple récupéré sur le réseau. Ce processeur simple (SimpleProc.v) décrit en Verilog est fournit sur Madoc.

dans sa version initiale le processeur fonctionne avec une mémoire interne qui sert au stockage du programme et des données. Il ne possède de fait aucun signal d'entrée/sortie, c'est donc un modèle purement simulable (pas de synthèse possible). On souhaite modifier cela et utiliser notre bloc *quadram* pour le stockage des données (i.e. le programme reste dans la mémoire du processeur). Le système final souhaité est représenté sur la Fig. 3.

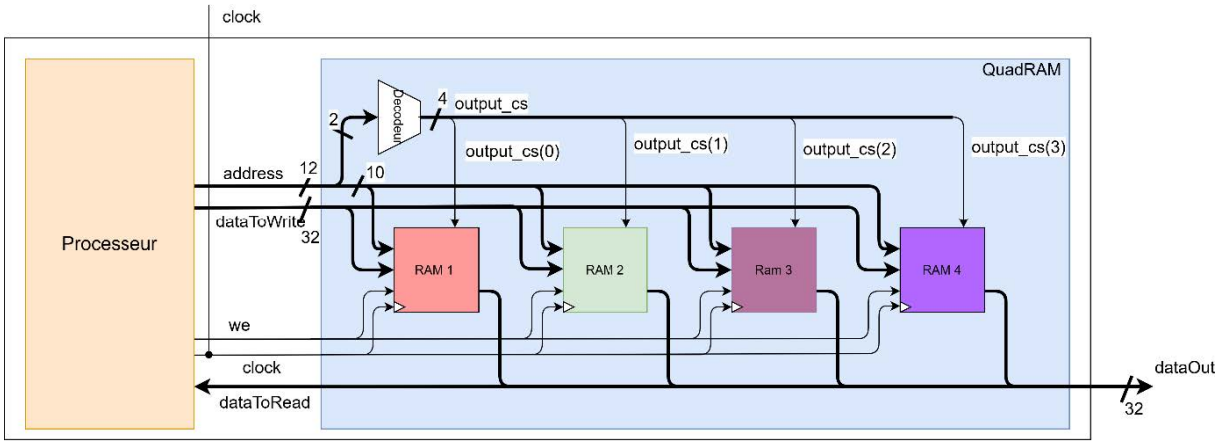

FIGURE 3 – Interfacage du processeur simple avec le bloc quadram précédent.

Il faut donc modifier le processeur afin de pouvoir utiliser le bloc mémoire externe pour le stockage des données. Ces modifications sont de deux ordres :

1. Modification de l'architecture matérielle du processeur afin de pouvoir l'interfacer avec le bloc *quadram* :
   - Ajout des bus et des signaux de contrôle, permettant la connection avec le bloc quadram. Le processeur initial n'a pas d'horloge, et on pourra prévoir en option un signal de reset.
   - Les données étant maintenant stockés dans une mémoire externe les opérations arithmétique ne peuvent plus y accéder directement. Il convient donc de rajouter des registres tampon (A et B par example) pour la gestion de la mémoire.
   - D'autres modifications du matériel peuvent être nécessaires en fonction de votre stratégie d'implémentation du processeur.
2. Les modifications architecturales ci dessus nécessitent alors la modification de l'ISA [1] du processeur.
   - Toutes les opérations arithmétique qui accédaient aux données dans la mémoire interne doivent être modifiés afin de travailler avec les registres tampon intégrés précédemment
   - L'instruction *STR* doit être adapter pour accéder à la mémoire externe. Attention au temps de synchronisation requis par le bloc quadram.
   - Enfin des instructions doivent être ajoutées comme une instruction *LDA* pour charger une donnée dans un registre A servant aux opérations arithmétiques. De même pour une instruction *LDB* si il y a lieu (en fonction encore une fois de vos propres développements). On ne gérera ici que le mode d'adressage direct. On pourra envisager des instructions en adressage immédiat pour l'initialisation de la mémoire.

1. Instruction Set Architecture

Le code Verilog du processeur définit des instructions, leurs valeurs binaires, ainsi que la description de chaque instruction selon l'architecture ci-dessous.

4 bits	4 bits	12 bits	12 bits
OpCode	CCode	Operand AA	Operand BB

Le premier champ, OPCODE, correspond à l'instruction à effectuer, par exemple une addition ou un load. Le second champ permet de renseigner des conditions. Les deux derniers champs sont sur 12 bits et représente les opérandes de l'instruction. Ces deux opérandes vont permettre par exemple d'ajouter des données complémentaires utiles pour l'instruction souhaitée.

5 Création du système

Le système complet final doit intégrer le processeur modifié et le bloc générique *quadram*. Avant de pouvoir simuler le comportement du processeur, il faut créer un bloc qui liera le processeur avec le bloc de RAM générique. Il est rédigé en VHDL, et instancie le processeur Verilog en tant que "*component*" et crée également une instance de la RAM générique. Il est alors nécessaire de définir des signaux internes au bloc afin de relier ces deux composants.

Afin de valider complètement le système il faut faire exécuter un petit programme au processeur. Il est possible d'initialiser la mémoire interne du processeur avec un fichier qui contiendra les codes héxadécimaux du programme. Ce fichier, d'extension .h, doit être placé dans le répertoire du projet et chargé dans la mémoire du processeur grâce à une commande ModelSim ($readmemh).

Faire **valider** l'ensemble du projet par un enseignant.

6 Question pour un champion (optionnelle)

Rendre le modèle du système complet synthétisable sur une cible FPGA Xilinx. Pour cela il conviendra de modifier le modèle de mémoire afin d'instancier les mémoires physiques du FPGA, et modifier le code du processeur afin de le rendre réellement synchrone sur le signal d'horloge.