ALLARD Elsa

ROCHE Hugo

16/12/2025

Compte rendu Projet Contrôleur de mémoire SRAM ZBT MT55L512Y36F

5A SE

2025-2026

POLYTECH
DIJON

Table des matières

Introduction

L'objectif du projet est d'analyser le fonctionnement d'une mémoire SRAM ZBT (référence MT55L512Y36F) à partir de sa datasheet ainsi que d'exemples de code fournis. À partir de cette étude, il s'agira de décrire un contrôleur VHDL de cette mémoire en utilisant les outils de développement Xilinx (Vivado).

L'utilisation de ce type de mémoire n'est pas triviale, notamment parce que les opérations de lecture et d'écriture s'effectuent sur un même bus de données bidirectionnel. Un problème de synchronisation d'horloge peut également apparaître lorsqu'un chargement de données coïncide avec un changement d'opération. De plus, la mémoire ne prévoit aucun cycle d'attente entre une lecture et une écriture (voir figure 8, page 19 de la datasheet).

Le contrôleur à coder devra donc permettre :

- de séparer les deux canaux de données de 36 bits (lecture et écriture) ;
- de synchroniser le chargement des données et le déclenchement des opérations sur une horloge unique, afin d'assurer un fonctionnement fiable entre le contrôleur et la SRAM.

La gestion des adresses (sur 19 bits) sera quant à elle réalisée en dehors du contrôleur.

Création de composant

La première étape de conception a consisté à définir le composant VHDL principal représentant le contrôleur de la SRAM. Ce composant, nommé **SRAM_ctrl_module**, constitue une entité autonome chargée de l'interface entre le système utilisateur et la mémoire.

Les choix de conception effectués lors de cette étape sont les suivants :

- définition d'interfaces claires pour les commandes de lecture et d'écriture ;
- séparation explicite entre les signaux utilisateur (adresses, données, commandes) et les signaux spécifiques à la SRAM ;
- paramétrage du contrôleur afin de permettre son évolution future (notamment pour la gestion du burst).

Cette structuration permet une bonne lisibilité du code VHDL et facilite les phases de validation et de débogage.

Composante SRAM ZBT MT55L512Y36F

La mémoire MT55L512Y36F est une SRAM synchrone d'une capacité de 512K mots de 36 bits chacun. Elle offre un bus bidirectionnel unique pour les données et permet des cycles de lecture et d'écriture consécutifs sans latence. La mémoire possède plusieurs signaux de contrôle critiques, tels que RW# pour déterminer le sens des opérations, OE# pour activer la sortie, CE#, CE2# et CE2 pour la sélection de la puce, BWx# pour le masquage des octets, et ZZ# pour le mode snooze. Les cycles de lecture et d'écriture sont déclenchés sur le front descendant de l'horloge, et le passage direct entre lecture et écriture est autorisé conformément au mode ZBT. Ces caractéristiques impliquent que le contrôleur doit gérer de manière précise les transitions et s'assurer que le bus de données n'est jamais soumis à un conflit.

Buffer entrée/sortie IOBuff

La gestion de la séparation des données entre le bus INOUT de la SRAM et les deux bus indépendants du contrôleur se fera à l'aide de composants présents dans la librairie UNISIM : des buffers entrée/sortie 3 états IOBUF.

Chaque composant IOBUF comme dans la figure 1 possède trois ports de données (un IN, un OUT, un INOUT) et un bit de trigger qui permet de contrôler l'orientation des données du INOUT (si TRIG=0, alors c'est l'entrée qui est transmise vers l'INOUT et donc du contrôleur vers la SRAM, et si TRIG=1 alors c'est la sortie qui est transmise depuis l'INOUT donc depuis la SRAM vers le contrôleur).

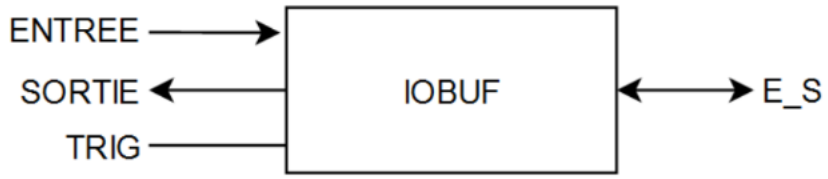

Figure 1:Schéma descriptif d'un composant IOBUF

Conception du contrôleur de SRAM

Le contrôleur SRAM_ctrl_module intègre l'ensemble des signaux nécessaires au pilotage de la mémoire. L'un des aspects centraux de son architecture est la gestion du bus bidirectionnel, rendue possible grâce à l'utilisation des IOBUF.

Lorsque le signal de contrôle est positionné à zéro, les données issues du FPGA sont envoyées vers la SRAM. À l'inverse, lorsqu'il est positionné à un, les données provenant de la SRAM sont capturées par le contrôleur. Cette stratégie garantit qu'aucun conflit n'apparaît sur le bus et que les échanges de données sont réalisés dans le bon sens en fonction de l'opération demandée.

Machine à état pour la commande Read/Write et contrôle des bits

L'objectif du contrôleur est d'implémenter deux signaux, **READ** et **WRITE**, permettant de déclencher respectivement une opération de lecture ou d'écriture dans la SRAM.

Les modes BURST ne seront pas pris en compte pour l'instant.

En se référant à la figure 7 page 11 de la datasheet, on obtient le diagramme des états internes de la SRAM. À partir de celui-ci, la machine à états du contrôleur sera conçue selon le principe des machines à sorties décodées (vu en 4e année), avec deux bits de commande : **R** et **W**.

- L'état par défaut est **Deselect (DS)**.
- L'état **READ** est prioritaire : si R = 1, l'état devient READ, quelle que soit la valeur de **W**.

La machine à états pourra donc être définie comme suit :

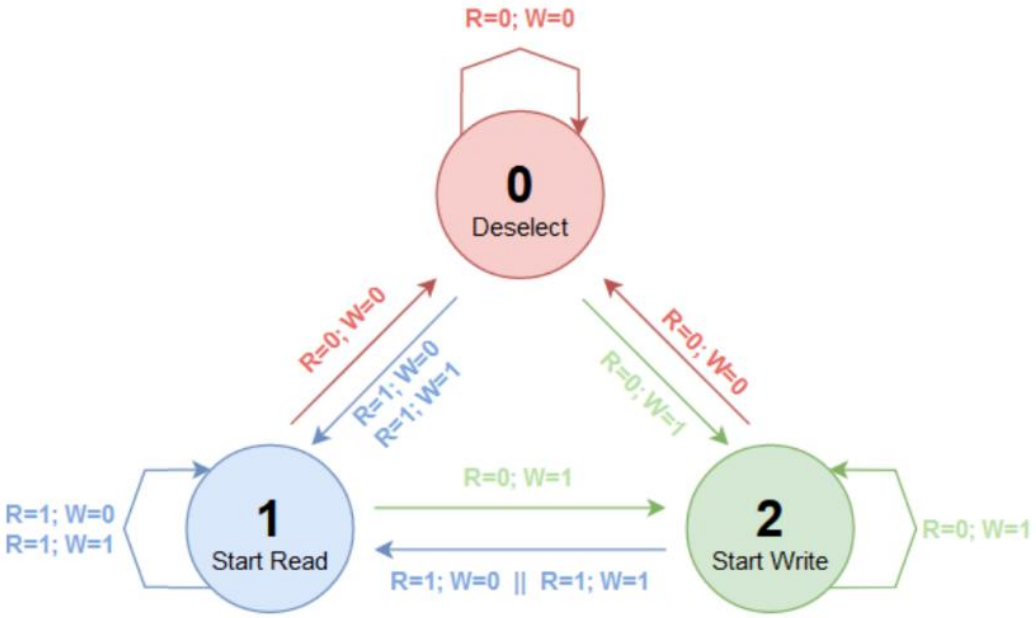

Figure 2: Machine à état code VHDL, sans les modes burst

La machine à états vu dans la figure 2 comporte quatre états. Le premier, **INIT**, n'est actif que lorsque le contrôleur est en reset. Il initialise les signaux de la SRAM de manière à désactiver toute opération sur la mémoire (ce/ce2 = 1).

Ensuite, l'état **IDLE** réactive la SRAM (ce/ce2 = 0) et attend une commande de lecture ou d'écriture. Afin d'éviter toute écriture involontaire, les signaux RW et tristate sont positionnés à '1', empêchant ainsi l'envoi de données vers la SRAM.

L'état **WRITE** est activé lorsqu'une commande d'écriture est reçue : le contrôleur prend la main sur le bus de données (tristate = '0') et envoie les données à la mémoire (RW = '0').

L'état **READ** est activé lors d'une commande de lecture : le contrôleur autorise la réception des données sur le bus (tristate = '1') et lit la mémoire (RW = '1').

La transition entre READ et WRITE peut se faire directement, sans repasser par IDLE, et donc sans perte de cycle d'horloge.

Enfin, pour protéger les données contenues dans la mémoire, une commande simultanée de lecture et d'écriture est interprétée comme une lecture. La lecture est donc prioritaire, ce qui évite une écriture accidentelle susceptible d'écraser des données.

Intégration du Burst

Une extension du contrôleur a consisté à intégrer la gestion du mode burst, permettant d'effectuer des lectures ou écritures successives sur plusieurs mots consécutifs.

Le mode burst repose sur les mécanismes suivants :

- Utilisation du signal ADV/LD pour incrémenter automatiquement l'adresse interne de la SRAM.
- Introduction d'un compteur interne permettant de suivre le nombre de mots restant à transférer.
- Ajout d'états spécifiques dans la machine à états (WRITE_BURST et READ_BURST).

Lors d'un accès burst :

- Le premier cycle charge l'adresse initiale.
- Les cycles suivants incrémentent automatiquement l'adresse sans rechargement.
- Le signal burst_cmd_done est activé lorsque le transfert est terminé.

Les simulations montrent que les données sont correctement transférées lors des accès burst, aussi bien en lecture qu'en écriture, sans violation des contraintes temporelles.

Testbench et validation fonctionnelle

Afin de valider le bon fonctionnement du contrôleur, un testbench a été développé. Celui-ci permet d'assembler le contrôleur avec un modèle comportemental de la SRAM et de vérifier les échanges de données.

Le testbench inclut :

- la génération de l'horloge et du reset ;
- l'instanciation du contrôleur SRAM ;
- un modèle comportemental de la mémoire MT55L512Y36F ;
- des scénarios de test en lecture et en écriture.

Afin de simplifier la validation fonctionnelle du contrôleur, la mémoire SRAM a été modélisée dans le testbench par un modèle comportemental. Ce choix permet de valider la logique de contrôle, la gestion du bus bidirectionnel et les commandes de lecture/écriture, sans introduire la complexité des délais temporels du modèle constructeur.

De plus, l'instanciation du contrôleur SRAM est réalisée à l'aide de la syntaxe entity work.SRAM_ctrl_module. La bibliothèque work correspond à la bibliothèque de travail par défaut de Vivado, dans laquelle sont compilées toutes les entités VHDL définies localement dans le

projet. L'utilisation explicite de work permet de lever toute ambiguïté sur l'origine du composant instancié et constitue une bonne pratique de conception, notamment dans des projets contenant plusieurs bibliothèques ou différentes versions d'un même module. Cette approche améliore la lisibilité et la robustesse du code sans modifier le comportement fonctionnel du système.

Assemblage du contrôleur et de la SRAM + test bench

On va finalement chercher à utiliser le contrôleur avec la SRAM, il faut pour cela créer un nouveau fichier (ici SRAM_crtl_module.vhd) afin de créer un nouveau composant.

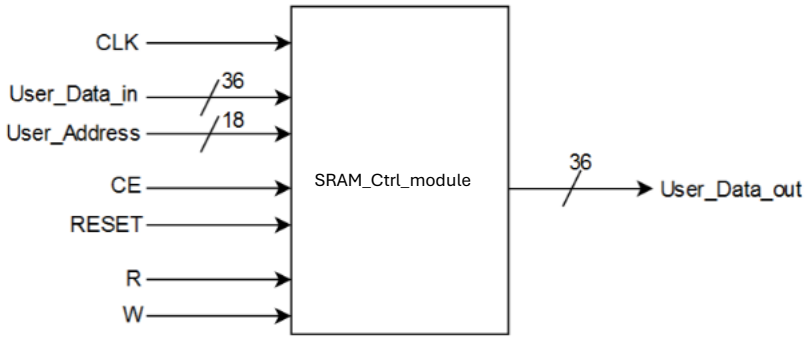

Figure 3: Schéma descriptif du composant

On va instancier les deux composants : la SRAM (mt55l512y36f) et le composant CTRL_SRAM_bloc_wrapper créé plus tôt à partir de notre contrôleur en block design, puis on va créer tous les signaux nécessaires afin de faire le lien entre les deux (les sorties du contrôleur seront ainsi liées aux entrées de la SRAM et les ports assignés seront ceux vus dans le schéma).

On mappe ensuite les deux composants (CTRL_SRAM1 et SRAM1).

Résultats de simulation

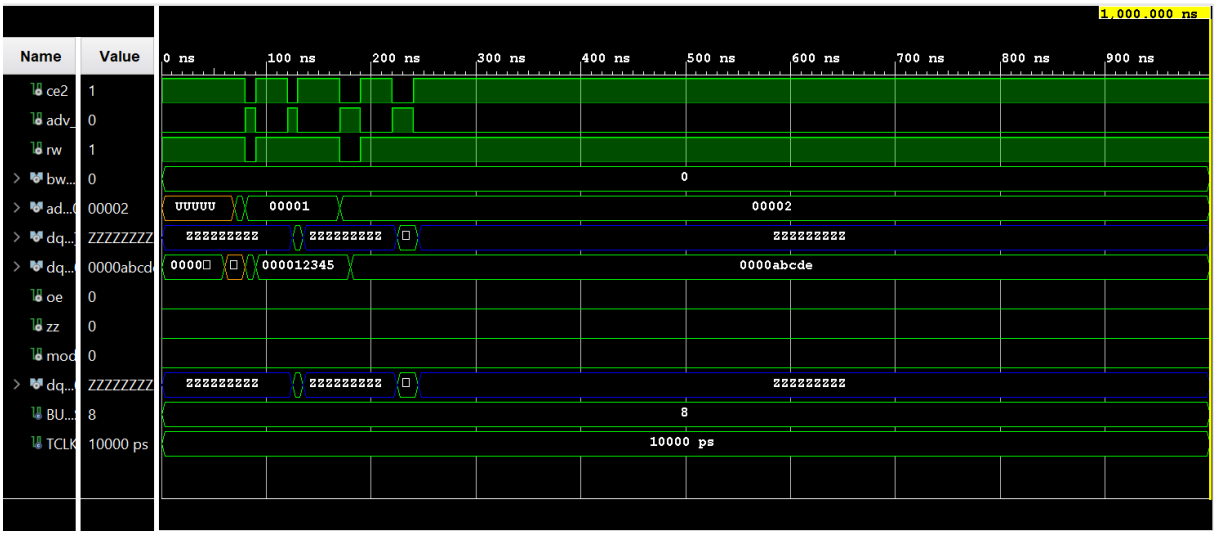

Figure 4: Résultat de la simulation

Analyse d'une opération d'écriture

Lors d'une opération d'écriture, on observe les séquences suivantes :

- L'adresse (addr) est positionnée et reste stable pendant toute la durée de l'accès mémoire.
- Le signal rw est positionné à 0, indiquant une opération d'écriture.
- Le signal tristate passe à 0, ce qui signifie que le contrôleur prend le contrôle du bus de données.
- Les données à écrire apparaissent sur dq_out et sont transmises vers la SRAM via le bus bidirectionnel.

Cette séquence confirme que les données sont présentes et stables au moment du front actif de l'horloge, conformément aux spécifications de la SRAM.

Analyse d'une opération de lecture

Lors d'une opération de lecture, le comportement observé est le suivant :

- L'adresse est maintenue stable.
- Le signal rw passe à 1, sélectionnant une lecture.
- Le signal tristate est positionné à 1, libérant le bus de données.

- La SRAM place alors les données correspondantes sur le bus, visibles sur dq_in.
- Les données lues sont ensuite transférées vers user_data_out.

On constate que les données apparaissent sur le bus après le front actif de l'horloge, ce qui justifie l'utilisation d'un registre de synchronisation afin de garantir leur stabilité et leur exploitation correcte par la FSM.

Absence de conflit sur le bus

Les chronogrammes montrent clairement que le bus de données n'est jamais piloté simultanément par le contrôleur et la SRAM. Le signal tristate assure une séparation stricte des responsabilités, évitant tout conflit électrique et toute corruption de données.

Les résultats de simulation montrent que les données écrites sont correctement stockées et relues, que le bus bidirectionnel est correctement maîtrisé et qu'aucun conflit n'apparaît lors des transitions entre opérations.

Conclusion

Ce projet a permis de concevoir, simuler et valider un contrôleur SRAM ZBT fonctionnel en VHDL. Les principaux défis rencontrés concernaient la gestion du bus bidirectionnel et la synchronisation des données en lecture.

Un problème majeur identifié était le suivant : les données lues depuis la SRAM étaient initialement utilisées directement, alors qu'elles arrivent sur le bus après le front actif de l'horloge. Cette désynchronisation entraînait des lectures incorrectes ou instables.

La solution apportée a consisté à ajouter un registre de synchronisation sur le signal dq_in, permettant d'aligner correctement les données lues avec la FSM du contrôleur. Cette correction a permis de fiabiliser totalement les opérations de lecture.

Plusieurs améliorations sont envisageables :

- une gestion plus fine du signal oe afin d'optimiser la consommation ;
- l'optimisation du pipeline pour augmenter le débit mémoire ;
- une validation matérielle sur FPGA avec une SRAM réelle.
- Une simulation plus réaliste pourrait être obtenue en instanciant le modèle constructeur MT55L512Y36F, ce qui permettrait de valider le respect strict des contraintes temporelles.