Utilisation de la mémoire EEPROM AT93C46D

Introduction

Ce document a pour but de décrire l'utilisation et la validation des fonctions permettant d'activer, de lire et d'écrire dans cette mémoire.

La mémoire choisie pour ce projet est une mémoire **Atmel AT93C46D** communiquant avec le **FPGA** via une liaison **SPI** (Serial Peripheral Interface). Nous avons choisi de configurer la mémoire de manière à échanger des mots de 16 bits (la broche **ORG** est reliée à **VCC**).

1. Activation de la mémoire

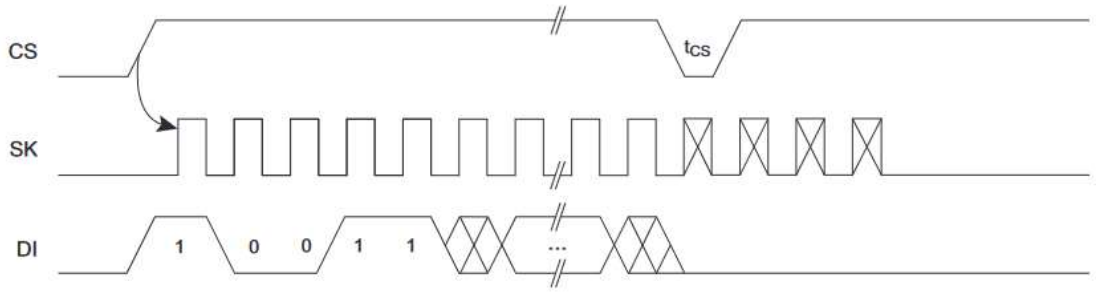
1.1. Présentation

Lors de toute première alimentation, il faut activer la mémoire via une instruction spécifique (**EWEN**). Cette instruction permet d'activer l'EEPROM et ainsi de pouvoir commencer à écrire ou lire dans l'EEPROM.

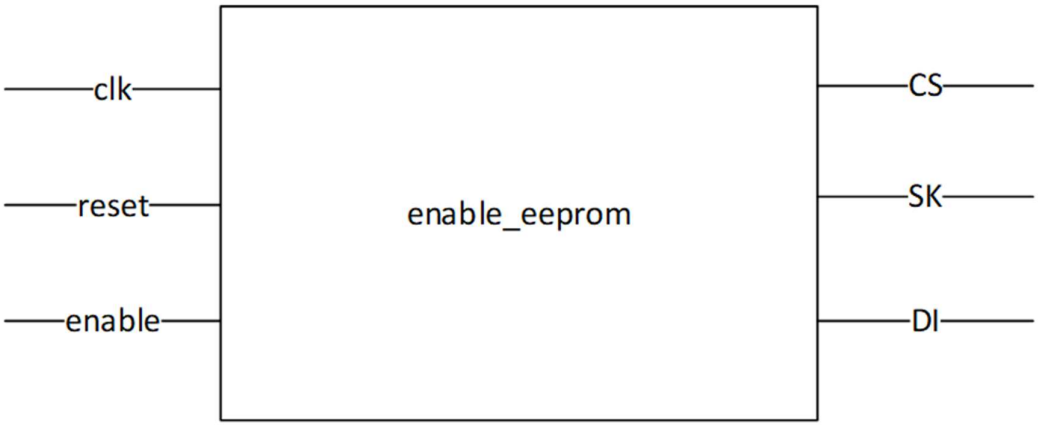
1.2. Symbole de enable_eeprom

Pour activer l'EEPROM, il faut envoyer sur la broche **DI**, l'instruction suivante en respectant le chronogramme suivant :

Figure 6-3. EWEN Timing



Par conséquent, le symbole de **enable_eeprom** est le suivant :



En entrée, nous avons :

- **`Clk`** : horloge 100 MHz
- **`Reset`** : actif sur front montant
- **`Enable`** : pour activer le lancement de l'instruction.

En sortie, nous avons :

- **`CS`** : signal d'activation de l'EEPROM
- **`SK`** : horloge de l'EEPROM
- **`DI`** : entrée série de l'EEPROM.

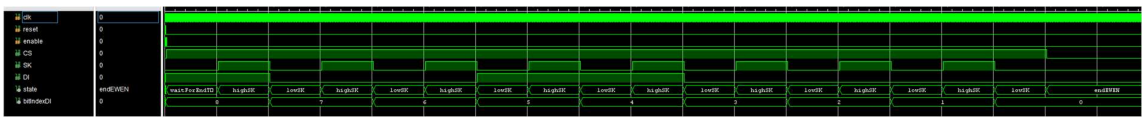
1.3. Architecture de enable_eeprom

L'architecture de ce composant est composée de 6 processus :

- Une machine à états finie
- Un registre à décalage pour envoyer les données en série
- Un compteur permettant de cadencer l'horloge de l'EEPROM
- 3 pour la gestion des sorties.

1.4. Simulation de enable_eeprom

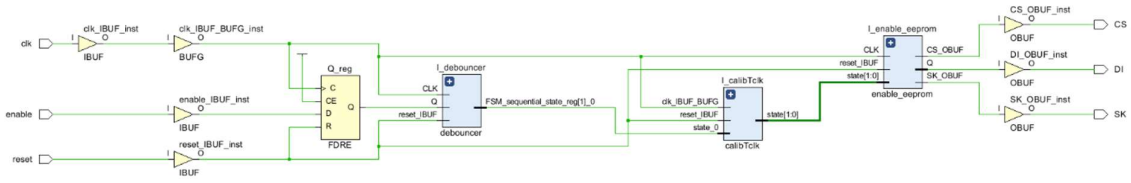
Pour faciliter l'architecture de **`enable_eeprom`**, il a été choisi d'avoir le même temps entre l'activation de l'horloge et une demi-période. Cela est possible grâce au fonctionnement de l'EEPROM. Une fois la simulation effectuée, nous obtenons les chronogrammes suivants :



Nous pouvons donc constater que la simulation correspond aux chronogrammes fournis dans la datasheet. Il ne nous reste plus qu'à implanter cette architecture dans la carte de développement Basys3.

1.5. Implémentation de enable_eeprom dans la carte Basys3

Pour assurer le fonctionnement sur la carte, nous créons le composant **`enable_eeprom_Basys3`** dont l'architecture est la suivante :



Nous utilisons les composants **`calibTclk`**, **`debouncer`** et une bascule D car le signal **`enable`** est implémenté via le bouton central de la carte. Cela nous permet d'éviter les effets de rebonds, est d'être sûr que l'appui sur le bouton se face correctement. L'appui sur le bouton du haut permet de réaliser le **`reset`** de chaque processus. Sur le port **`JA`**, sont connectées les broches permettant d'alimenter l'EEPROM avec les broches **`3.3V`** et **`GND`** pour l'alimentation. Les broches 1, 2, 3 et 4 sont respectivement connectées aux broches **`CS`**, **`SK`**, **`DI`** et **`DO`** et ceux pour n'importe quel module.

2. Ecriture dans la mémoire

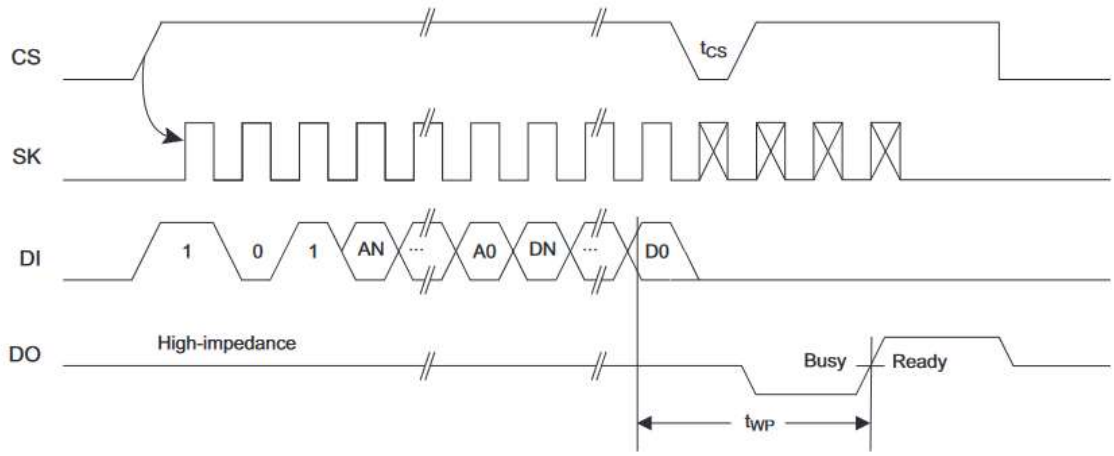
2.1. Présentation

Une fois notre EEPROM activée, nous pouvons écrire dans une de ses adresses.

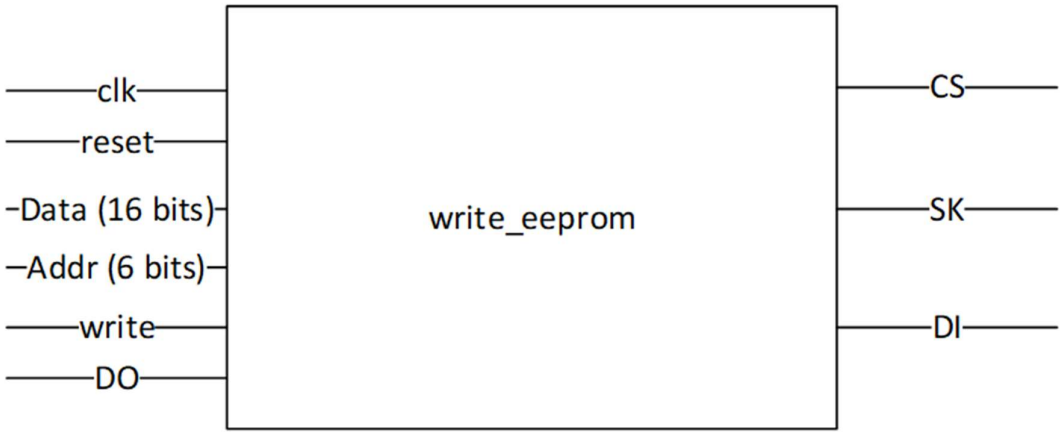
2.2. Symbole de write_eeprom

Pour écrire dans l'EEPROM, il faut envoyer sur la broche **`DI`**, l'instruction suivante en respectant le chronogramme suivant :

Figure 6-5. WRITE Timing



Par conséquent, le symbole de **`write_eeprom`** est le suivant :



En entrée, nous avons :

- **`Clk`** : horloge 100 MHz
- **`Reset`** : actif sur front montant
- **`Write`** : pour activer le lancement de l'instruction
- **`Addr`** : l'adresse sur 6 bits où stocker les données à écrire
- **`Data`** : les données à écrire sur 16 bits
- **`DO`** : signal de sortie de l'EEPROM qui permet d'indiquer la fin de l'écriture.

En sortie, nous avons :

- **`CS`** : signal d'activation de l'EEPROM
- **`SK`** : horloge de l'EEPROM

- **`DI`** : entrée série de l'EEPROM.

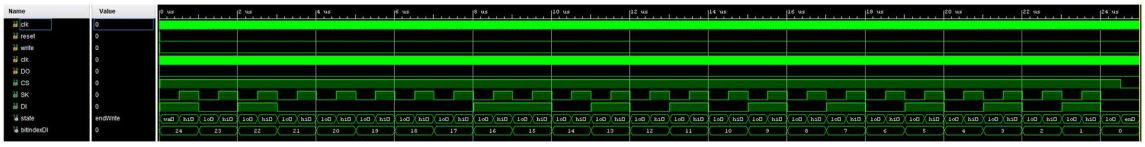
2.3. Architecture de write_eeprom

L'architecture de **`write_eeprom`** est très similaire à celle de **`enable_eeprom`**, elle comprend donc 6 processus :

- Une machine à états finie similaire à **`enable_eeprom`**
- Un registre à décalage pour envoyer les données en série
- Un compteur permettant de cadencer l'horloge de l'EEPROM, identique à **`enable_eeprom`**
- 3 pour la gestion des sorties.

2.4. Simulation de write_eeprom

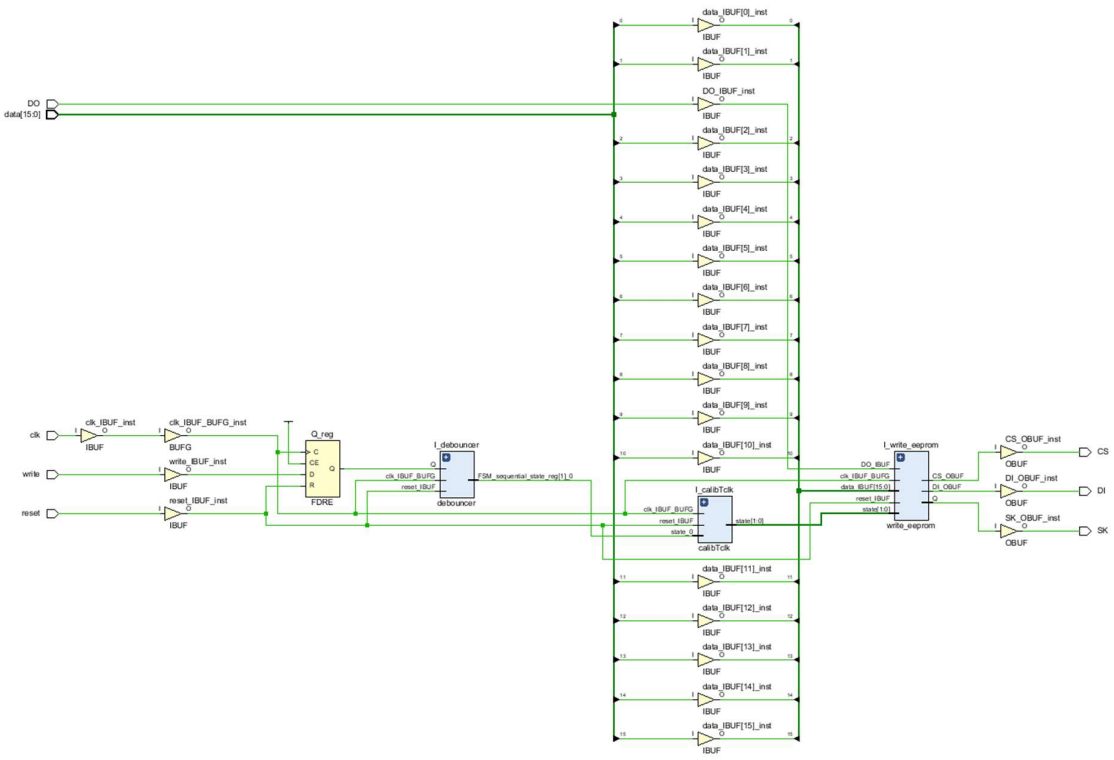
Une fois la simulation effectuée, nous obtenons les chronogrammes suivants :



Nous pouvons donc constater que la simulation correspond aux chronogrammes fournis dans la datasheet. Il ne nous reste plus qu'avec implanter cette architecture dans la carte de développement Basys3.

2.5. Implémentation de write_eeprom dans la carte Basys3

Pour assurer le fonctionnement sur la carte, nous créons le composant **`write_eeprom_Basys3`** dont l'architecture est la suivante :



Nous utilisons les composants **`calibTclk`**, **`debouncer`** et une bascule D car le signal **`write`** est implémenté via le bouton central de la carte. Cela nous permet d'éviter les effets de rebonds, est d'être sûr que l'appui sur le bouton se face correctement. L'appui sur le bouton du haut permet de réaliser le reset de chaque processus. Pour simplifier le fonctionnement, l'adresse où écrire est définie directement dans l'architecture. Les données à écrire sont recopiées des interrupteurs de la carte et l'appui sur **`write`** permet de lancer l'écriture. A la fin de l'écriture le signal **DO** passe à 1 et la led0 s'allume (attention durée très courte donc difficilement perceptible). Nous pouvons ainsi vérifier grâce au module **`read_eeprom`** si à cette adresse les données ont été écrites.

3. Lecture dans la mémoire

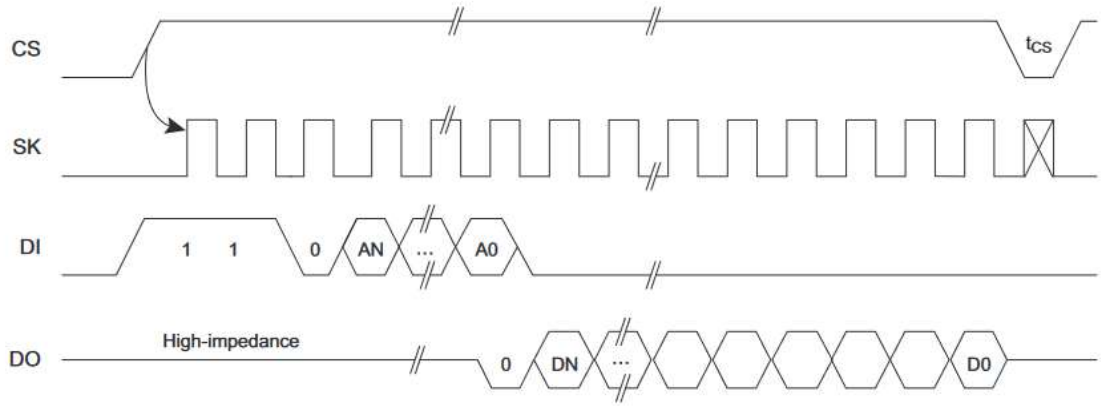
3.1. Présentation

Après avoir écrit dans l'EEPROM, nous allons vérifier si les données ont été correctement envoyées. Pour cela, nous utilisons le module **`read_eeprom`**.

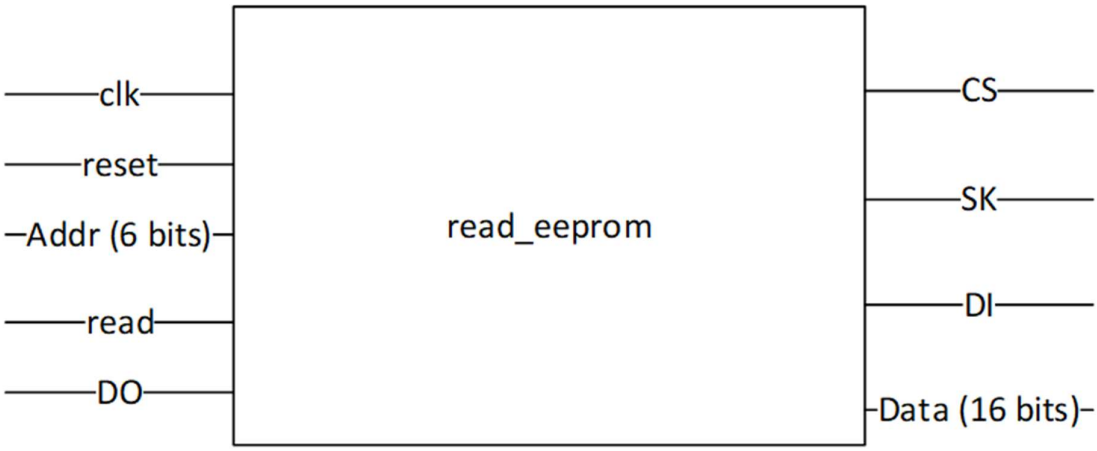
3.2. Symbole de read_eeprom

Pour écrire dans l'EEPROM, il faut envoyer sur la broche **`DI`**, l'instruction suivante en respectant le chronogramme suivant, on récupère sur la broche **DO** les données stockées à cette adresse :

Figure 6-2. READ Timing



Par conséquent, le symbole de **`read_eeprom`** est le suivant :



En entrée, nous avons :

- **`Clk`** : horloge 100 MHz
- **`Reset`** : actif sur front montant
- **`Read`** : pour activer le lancement de l'instruction
- **`Addr`** : l'adresse sur 6 bits où stocker les données à écrire
- **`DO`** : signal de sortie de l'EEPROM qui permet d'envoyer les données de l'adresse en série.

En sortie, nous avons :

- **`CS`** : signal d'activation de l'EEPROM
- **`SK`** : horloge de l'EEPROM
- **`DI`** : entrée série de l'EEPROM
- **`Data`** : les données écrites dans l'adresse sur 16 bits.

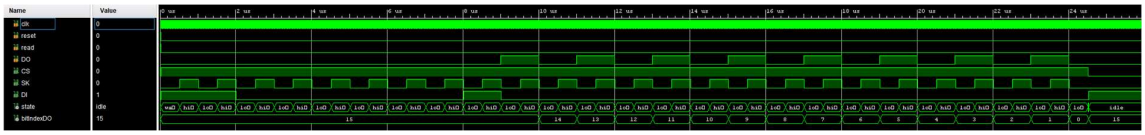
3.3. Architecture de read_eeprom

L'architecture de **`read_eeprom`** est très similaire à celle de **`enable_eeprom`**, elle comprend donc 8 processus :

- Une machine à états finie similaire à **`enable_eeprom`**
- Un registre à décalage pour envoyer les données en série
- Un compteur permettant de cadencer l'horloge de l'EEPROM, identique à **`enable_eeprom`**
- Un registre à décalage pour stocker les informations en provenance de l'EEPROM
- 4 pour la gestion des sorties.

3.4. Simulation de read_eeprom

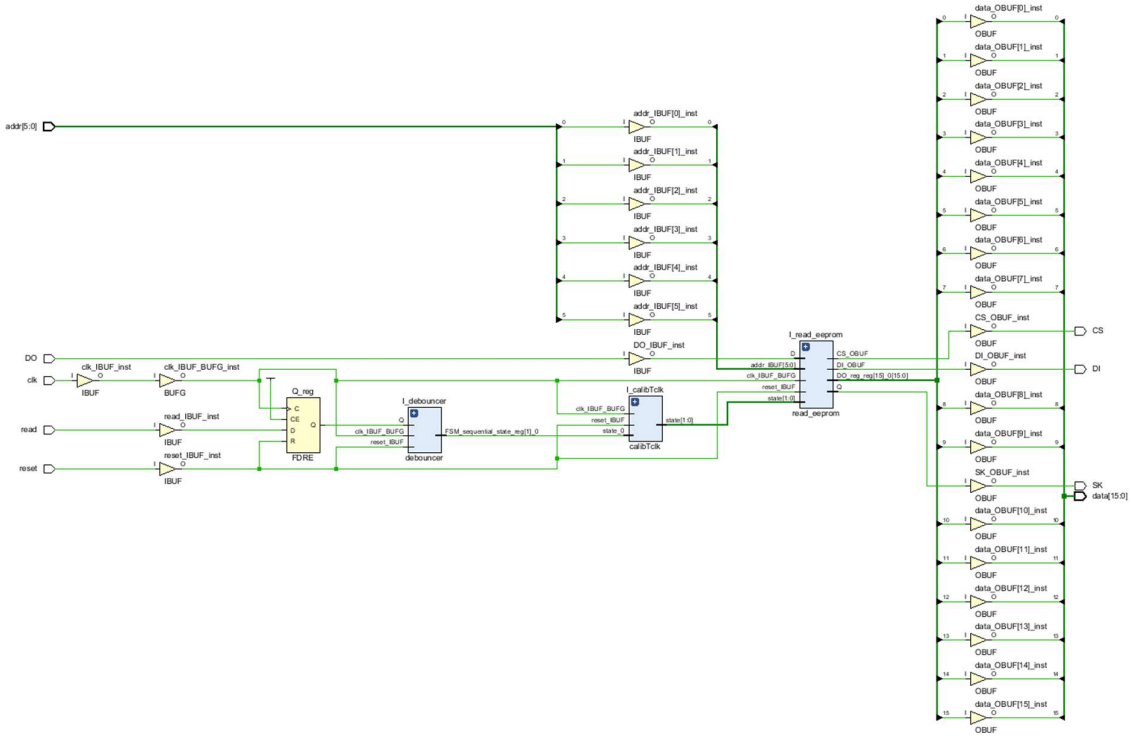
Une fois la simulation effectuée, nous obtenons les chronogrammes suivants :



Nous pouvons donc constater que la simulation correspond aux chronogrammes fournis dans la datasheet. Il ne nous reste plus qu'avec implanter cette architecture dans la carte de développement Basys3.

3.5. Implémentation de read_eeprom dans la carte Basys3

Pour assurer le fonctionnement sur la carte, nous créons le composant **`read_eeprom_Basys3`** dont l'architecture est la suivante :



Nous utilisons les composants **calibTclk**, **debouncer** et une bascule D car le signal **read** est implémenté via le bouton central de la carte. Cela nous permet d'éviter les effets de rebonds, est d'être sûr que l'appui sur le bouton se face correctement. L'appui sur le bouton du haut permet de réaliser le reset de chaque processus. Les interrupteurs SW0 à SW5 permettent de définir l'adresse qui doit être lue et l'appui sur **read** lance le fonctionnement. Les leds recopient donc l'état de data et indiquent donc les données stockées à l'adresse demandée.

4. Bilan

En résumé, pour faire fonctionner la mémoire EEPROM il faut implémenter successivement les modules suivants :

- **Enable_eeprom_basys3** : pour activer l'EEPROM (uniquement lors de sa première utilisation)
- **Write_eeprom_basys3** : pour écrire les données dans une adresse prédéfinie
- **Read_eeprom_basys3** : pour lire les données contenues dans une adresse.

Les procédures pour valider le fonctionnement sont les suivantes :

1. Implémenter le module **enable_eeprom_basys3** sur la carte avec le fichier **xdc** correspondant et appuyer sur le bouton central.

2. Implémenter le module **`write_eeprom_basys3`** avec le fichier **`xdc`** correspondant, définir les données à envoyer en utilisant les interrupteurs sur une adresse prédéfinie dans le programme
3. Implémenter le module **`read_eeprom_basys3`** avec le fichier **`xdc`** correspondant, sur les interrupteurs SW0 à SW5 définir l'adresse à lire, le résultats de la lecture sera affiché sur les leds.

Tous les modules fonctionnent et ont été testés sur la carte Basys3.

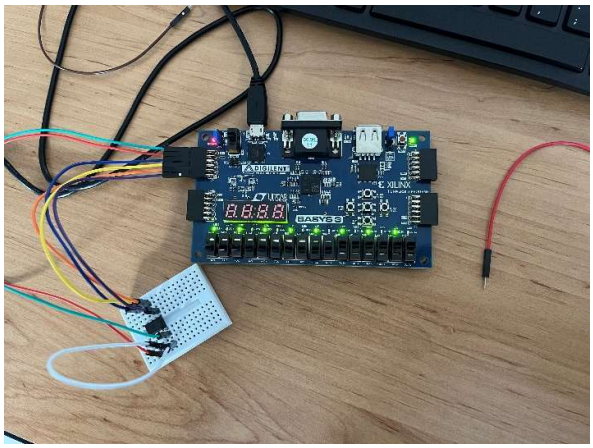


Figure 1 : Illustration du fonctionnement (data 1010101010101010 à l'adresse 111111)

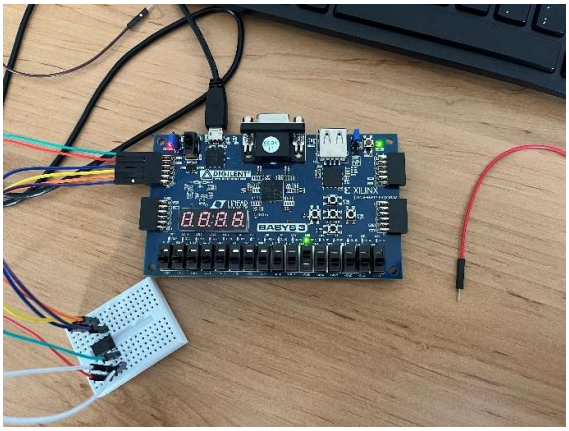


Figure 2 : Illustration du fonctionnement (data 0000000000100000 à l'adresse 100000)

Pour aller plus loin et aboutir ce projet, il faut réaliser un programme qui permettent de réaliser ces trois modules en un. Pour démontrer le fonctionnement il a été choisi d'en faire 3 pour bien montrer l'utilisation des périphériques de la cartes et ainsi valider étape après étape le fonctionnement des différents module.

Notes

L'ensemble des fichiers **vhd** et **xdc** nécessaires à l'utilisation de l'EEPROM est disponible sur le répertoire git.

Ou alors dans le dossier eeprom.zip joint avec ce document. De plus sont ajoutés les fichiers **vhd** permettant la simulation de chaque module.

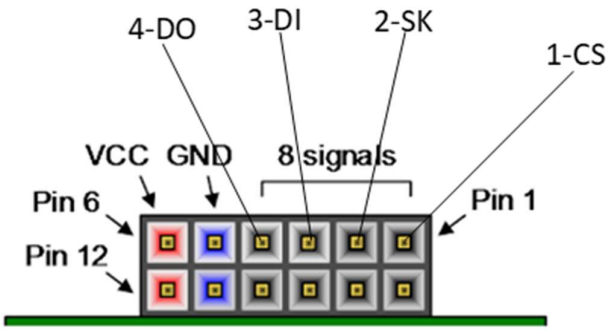


Figure 3 : Schéma de câblage du connecteur JA pour relier l'EEPROM à la Basys3