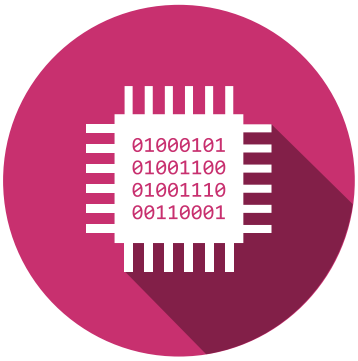


Mémoire mortes (ROM)

Cours Systèmes numérique

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientation : Informatique et systèmes de communication (ISC)
Cours : Systèmes numérique (DiD)
Auteur : Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Date : 25 août 2022
Version : v2.1

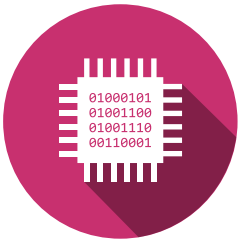
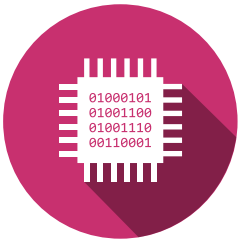


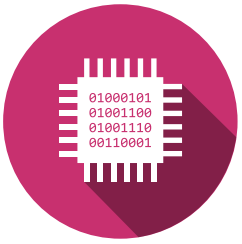
Table des matières



1 Introduction

Les mémoires mortes peuvent servir à réaliser une fonction logique combinatoire quelconque, par programmation de fusibles. Elles s'utilisent principalement pour contenir le programme d'un microprocesseur. A ce titre, elles sont connectées au microprocesseur de la même façon que les mémoires vives. Le développement des technologies permet de réaliser des mémoires réinscriptibles ; cependant cette opération reste plus lente que l'écriture dans une mémoire vive.

Ce chapitre présente la structure des mémoires mortes, les méthodes de connexion pour en augmenter la capacité, différents types de mémoires mortes et certains circuits typiques.



2 Fonction logique universelle

2.1 Structure de la ROM

Un démultiplexeur avec l'entrée de donnée à '1' décode une entrée de commande à n bits et génère 2^n sorties correspondant à toutes les combinaisons possibles des entrées. La table 1 présente la table de vérité d'un démultiplexeur à 2 entrées de commande et montre comment réaliser différentes fonction en combinant ces sorties à l'aide de portes OU.

a_1	a_0	w_0	w_1	w_2	w_3	d_3	d_2	d_1	d_0
0	0	**1**	0	0	0	1	0	0	1
0	1	0	**1**	0	0	0	1	1	1
1	0	0	0	**1**	0	1	1	1	1
1	1	0	0	0	**1**	0	1	0	0

TABLE 1 – Réalisation de fonctions à l'aide d'un démultiplexeur

Les fonctions présentées dans la table 1 sont données par les équations 2.1.

$$\begin{cases} d_3 = \overline{a_0} \\ d_2 = a_0 + a_1 \\ d_1 = a_0 \oplus a_1 \\ d_0 = \overline{a_0 \cdot a_1} \end{cases} \tag{1}$$

Le schéma correspondant est présenté à la figure 1.

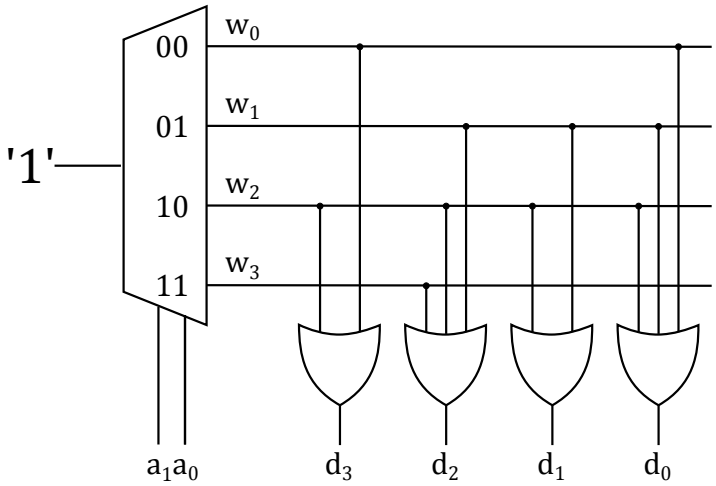
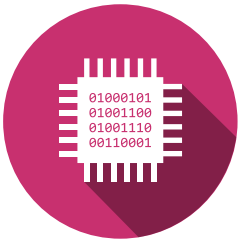


FIGURE 1 – Structure d'une ROM

Il est évident qu'avec cette technique combinant un démultiplexeur et une porte OU il est possible de réaliser n'importe quelle fonction logique combinatoire. De tels circuits existent sous forme de circuit intégré et sont appelés mémoires à lecture uniquement (**R**ead **O**nly **M**emory (ROM)) ou encore mémoires mortes. Le terme de mémoire provient de leur utilisation principale qui est la mémorisation de programmes d'ordinateur.

Utilisée comme fonction logique combinatoire, une ROM correspond à une table de vérité :



- les lignes de sortie du démultiplexeur correspondent aux lignes de la table,
- il y a 2^n lignes pour n entrées,
- une connexion à une porte OU correspond à un '1' dans la table, et l'abscence de connexion à un '0'.

2.2 Réalisation de la fonction OU programmable

Pour réduire la taille du circuit et pour en augmenter la régularité, la fonction OU se réalise de manière câblée. La figure 2 présente une réalisation possible de cette fonction OU sous forme de fonction NOR suivie d'un inverseur. Dans ce schéma, le NOR-câblé se réalise à l'aide de transistors MOS qui fonctionnent comme des interrupteurs : ils conduisent lorsque l'entrée de commande est à '1' et court-circuitent alors au '0' logique, V_{SS}, la ligne à laquelle ils sont connectés. Si aucun transistor ne force la ligne à '0', une résistance tire celle-ci au '1' logique, V_{DD}.

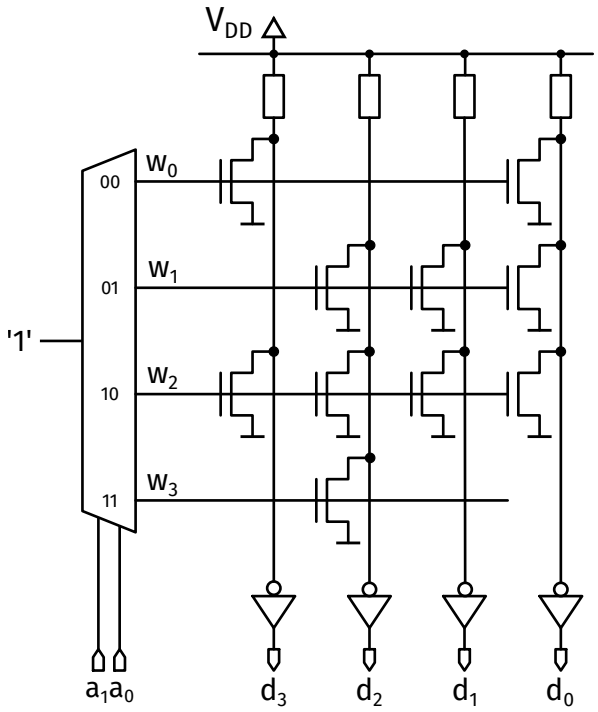
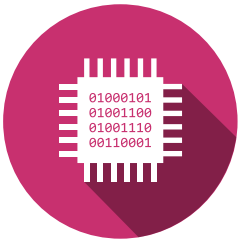


FIGURE 2 – Mémoire morte de type NOR

Un '0' dans la table de vérité correspond à un transistor absent et un '1' à un transistor présent.



3 Assemblage de circuits de mémoire

3.1 Capacité d'une mémoire

La capacité d'une mémoire est donnée par le nombre de bits de la table de vérité qui lui correspond. Cette capacité est donnée par le produit du nombre de lignes de sortie du multiplexeur, n_w, par le nombre de lignes de sortie ou de données, n_d de la mémoire (équation 3.1).

$$C = n_w \cdot n_d = 2^{n_a} \cdot n_d \tag{2}$$

Rappelons que le nombre de lignes de sortie du multiplexeur, n_w, vaut 2 à la puissance du nombre de lignes d'entrée ou d'adresse de la mémoire, n_a (équation 3.1).

$$n_w = 2^{n_a} \tag{3}$$

3.1.1 Exemples

La mémoire à 2 lignes d'entrée et 4 lignes de sortie de la figure 2 a une capacité de 4 x 4 bits.

Une mémoire à 10 lignes d'entrée et 8 lignes de sortie est une mémoire de 1024 x 8 bits. En considérant que 1024 = 1k et que 8 bits sont un *octet (byte, B)*, cette mémoire a une capacité de 1kB.

Une mémoire à 16 lignes d'entrée et 8 lignes de sortie est une mémoire de 65536 x 8 bits, soit 64kB.

3.2 Extension de mémoire

Le développement des microprocesseurs et la demande toujours accrue de mémoires de taille plus importante pour des programmes toujours plus complexes incite les utilisateurs à étendre la mémoire de leur système. L'accroissement de la capacité d'un mémoire se fait soit en augmentant le nombre d'adresses, et ainsi le nombre de lignes de la table de vérité, soit en augmentant le nombre de lignes de données.

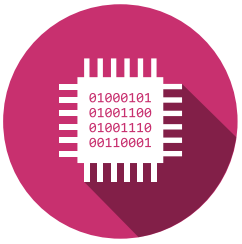
3.3 Mise en série

Pour augmenter la capacité d'une mémoire en gardant constant le nombre de lignes de données, il est possible de mettre des circuits en série.

Pour cela, les sorties des circuits sont connectées ensembles et un circuit de décodage sélectionne l'un des circuits qui est appelé à transférer son information sur les lignes de données. Un seul circuit est sélectionné à la fois et les autres circuits ont leurs sorties en haute impédance pour ne pas interférer. Ceci est présenté à la figure 3.

3.4 Décodage

Pour des circuits de mémoire de capacité identique, le décodage se fait par un démultiplexeur. Il est contrôlé par les bits de poids fort de l'adresse et les bits de poids faible sont transmis



directement aux mémoires pour leur démultiplexeur interne.

Pour des circuits de mémoire de capacité différente, le décodage nécessite un circuit plus complexe, correspondant en général à un arbre de démultiplexeurs adapté à la taille des démultiplexeurs des mémoires.

FIGURE 3 – ROMs mises en série

3.5 Carte de la mémoire

Lors de la réalisation de systèmes à processeur avec différents circuits de mémoire, il s'impose de faire une *carte de la mémoire (memory map)* indiquant quel circuit répond à quelle adresse.

Cette carte indique le fonctionnement du circuit de décodage qui sélectionne un circuit de mémoire en fonction des adresses présentées par le microprocesseur.

3.5.1 Exemple

La figure 4 présente un exemple d'occupation de la mémoire d'un microprocesseur à 16 lignes d'adresses.

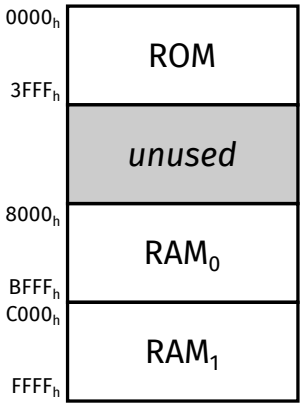


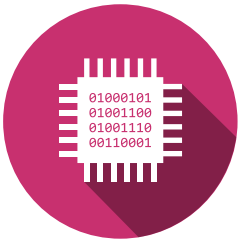
FIGURE 4 – Exemple d'occupation de la mémoire d'un microprocesseur

Avec cette occupation, la ROM répond tant que l'adresse se trouve entre 0000_h et $3FFF_h$. Si l'adresse est entre 8000_h et $BFFF_h$, la RAM_0 est sélectionnée. Avec une adresse entre 4000_h et $7FFF_h$, aucun circuit ne réagit et les lignes de données restent en haute impédance.

Ce décodage peut se faire avec un démultiplexeur de 1 à 4, commandé par les 2 bits de poids fort des adresses.

3.5.2 Commentaire

Comme les microprocesseurs sont prévus pour communiquer avec les mémoires par un bus d'adresses et de données, les fabricants proposent des circuits d'entrée/sortie, comme par exemple des ports série ou parallèle, lesquels utilisent le même mode de communication. Le microprocesseur voit ces circuits comme de petites mémoires, et une lecture des données d'un tel circuit permet au processeur d'accéder aux données arrivées par ce port. On parle dans ce cas *d'entrées/sorties situées en mémoire (memory-mapped I/O)*.



3.6 Mise en parallèle

Pour augmenter la capacité d'une mémoire en gardant constant le nombre de lignes d'adresse, il est possible de mettre des circuits en parallèle (figure 5).

Ce montage est typique aux microprocesseurs travaillant sur 16, 32 ou 64 bits de données alors que les mémoires courantes fournissent 1, 4 ou 8 bits.

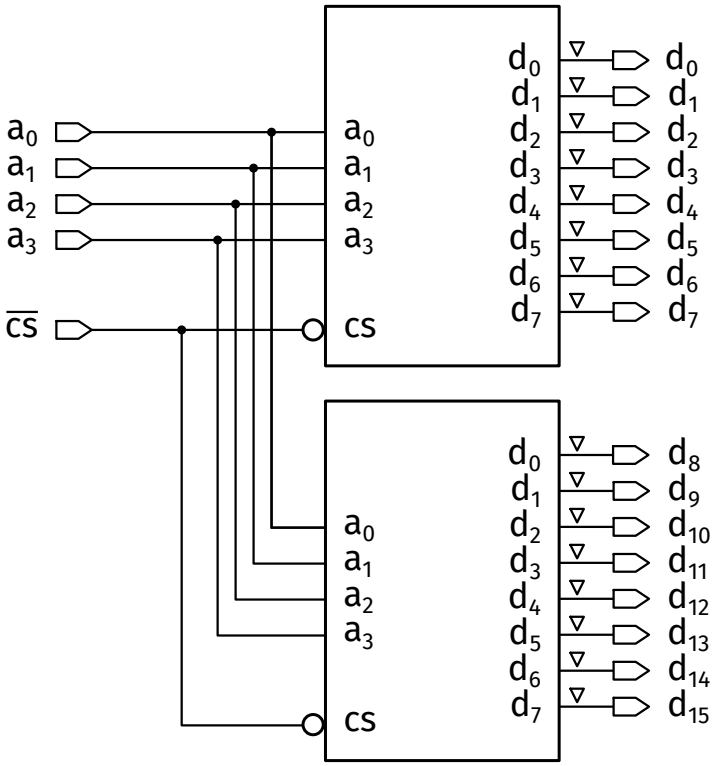
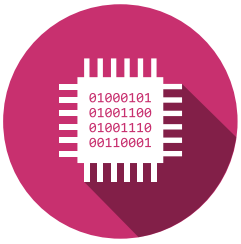


FIGURE 5 – ROMs mises en parallèle



4 Types de mémoires mortes

4.1 ROM

La fonction d'une ROM est donnée par l'existence ou non de transistors. Le placement des transistors se fait en usine, à la fabrication du circuit.

La réalisation d'une ROM nécessite un investissement de quelques milliers de francs et un délai de quelques semaines. Le choix d'une ROM n'est avantageux que pour de grandes séries, de l'ordre de plusieurs milliers de pièces.

4.2 PROM

Pour de plus petites séries, on peut utiliser des mémoires mortes programmables (PROM). Les liaisons entre transistors et lignes se font à travers de fusibles. La programmation de la ROM se fait en brûlant les fusibles connectés aux transistors indésirables. La figure 6 présente une telle PROM avec les fusibles non grillés.

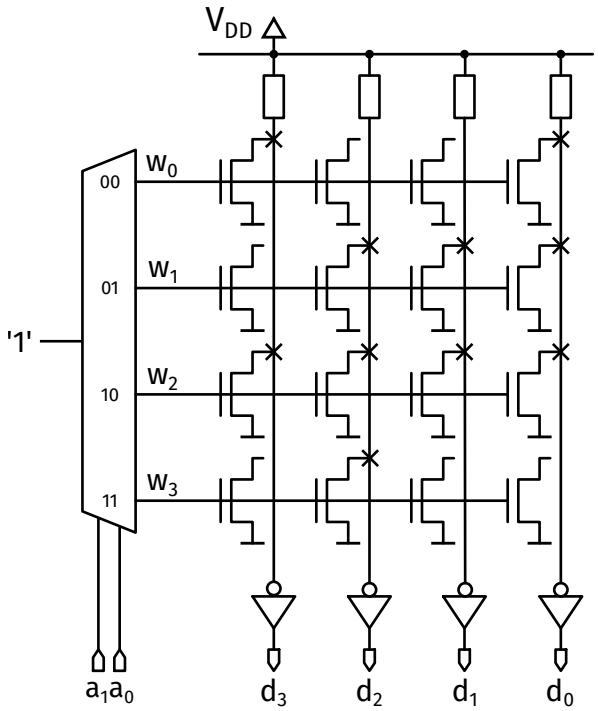
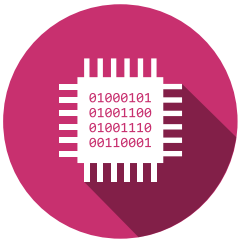


FIGURE 6 – PROM

La programmation d'une PROM nécessite l'utilisation d'un appareil ad hoc : un programmateur de mémoires. Avec des circuits vierges et un programmateur à disposition, le délai de production est presque nul.

4.2.1 Commentaire

La programmation du OU câblé se fait en brûlant des fusibles, et un '0' dans la table de vérité correspond à un transistor absent et donc un fusible grillé. Si donc le contenu d'une PROM n'est pas complètement utilisé, il est conseillé de laisser à '1' tous les bits de la partie libre, question de réduire le temps de programmation et de conserver la possibilité d'étendre la fonctionnalité du système.



4.3 Fichiers de programmation

Il existe plusieurs formats des fichiers de programmation de PROMs. Ces formats ont principalement été développés par des producteurs de microprocesseurs.

Le listing 4.3 donne le contenu d'une table de sinus en format Intel Hex.

```
:020000020000FC
:10000000000D1925313C47515B636A71767A7E7F1A
:100010007F7F7E7A76716A635B51473C3125190D8B
:1000200000F3E7DBCFC4B9AFA59D968F8A868281A6
:10003000808182868A8F969DA5AFB9C4CFDBE7F316
:00000001FF
```

Listing 1 – Intel HEX RAM Data

Les lignes y sont construites par le caractère " :" suivi d'un code donnant la longueur de l'information a venir, de l'adresse de départ, d'un code spécifiant le type d'information, des données elles-même et d'un code de vérification (checksum).

Le listing 4.3 donne le contenu d'une table de sinus en format Motorola EXORciser.

```
S00B000044415441204 92f4fF3
S1130000000D1925313C47515B636A71767A7E7F1A
S11300107F7F7E7A76716A635B51473C3125190D8B
S113002000F3E7DBCFC4B9AFA59D968F8A868281A6
S11300308081828 68A8F969DA5AFB9C4CFDBE7F316
S9030000FC
```

Listing 2 – Motorola EXORciser RAM Data

Les lignes y sont construites par le caractère "S" suivi d'un code donnant le type d'information a venir, d'un code donnant la longueur, de l'adresse de départ, des données elles-même et d'un code de vérification.

4.4 EPROM

Pour le développement de circuits à base de ROMs, l'utilisation de EPROMs permet de réaliser rapidement des prototypes. Cependant, chaque nouveau prototype nécessiterait de brûler une nouvelle EPROM. Pour ce type d'application, les producteurs proposent des mémoires mortes programmables effacables (**E**rasable **P**rogrammable **R**ead **O**nly **M**emory (EPROM)).

Dans ces circuits, les paires transistor-fusible sont remplacés par un transistor à grille flottante, comme présenté à la figure 7. La programmation consiste à amener une charge sur la grille flottante, laquelle ajoute une tension constante à celle de commande du transistor. Lorsque la grille est correctement chargée, le transistor ne conduit jamais.

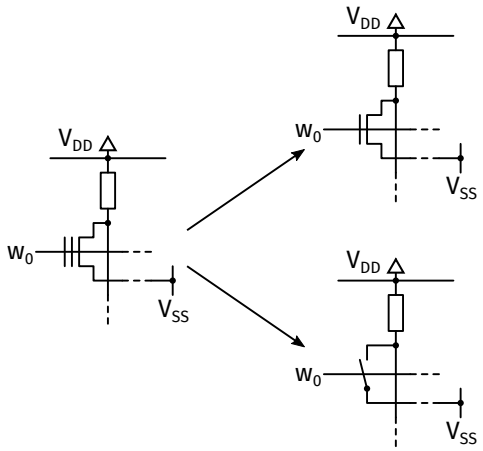


FIGURE 7 – Transistor à grille flottante

Une EPROM programmée peut être effacée en exposant le circuit intégré à des rayons ultraviolets. Ceci se fait à travers une fenêtre transparente aménagée dans le boîtier. Cette illumination donne suffisamment d'énergie aux charges piégées sur la grille flottante pour en échapper.

4.4.1 Remarque

Comme la lumière ambiante contient aussi des rayons UV, les EPROMs ont tendance à s'effacer petit à petit. Pour éviter ce phénomène, il est conseillé, après programmation, de recouvrir la fenêtre par une étiquette autocollante.

4.5 OTP-ROM

Les boîtiers à fenêtre étant plus chers que les boîtiers normaux, les fabriquants proposent aussi des EPROMs en boîtier normal, et donc non reprogrammables. Ces PROMs sont appelées mémoires mortes programmables une seule fois (**O**ne **T**ime **P**rogrammable **R**ead **O**nly **M**emory (OTP-ROM)).

Avec ces boîtiers plus économiques, les circuits intégrés d'EPROM, fabriqués en grande quantité, peuvent être vendus à plus bas prix.

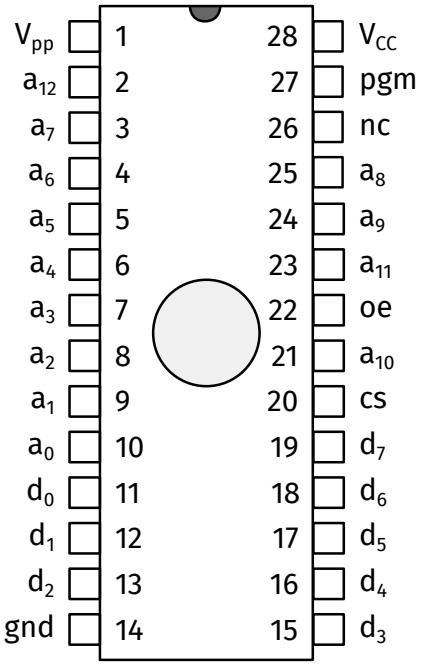


FIGURE 8 – EPROM boîtier avec fenêtre transparente

4.6 EEPROM

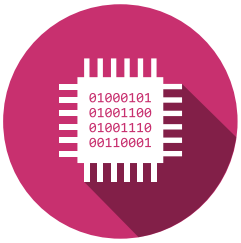
L'évolution des connaissances des phénomènes physiques liées à la charge et à la décharge des grilles flottantes permet aujourd'hui de réaliser des circuits où la décharge des grilles flottantes se réalise électriquement, sans illumination ultra-violette. Les circuits où l'effacage du contenu de la mémoire est électrique sont des mémoires mortes programmables effacables électriquement (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory (EEPROM)).

Bien qu'elles puissent être indépendamment écrites ou lues, les EEPROMs restent des mémoires utilisées principalement en lecture, car l'effacement et la programmation sont lents et le nombre de reprogrammations reste limité.

4.7 Mémoire flash

Pour réduire le matériel nécessaire à l'effacement des bits de l'EPROM, des fabricants proposent des circuits où l'effaçage et l'écriture subséquente se fait par blocs plutôt que mot par mot. Ces mémoires sont appelées des mémoires "flash".

Avec ce type de mémoire, l'écriture d'une nouvelle valeur dans un bit implique l'effacement de toute une page de la mémoire.



5 Circuits Typiques

5.1 Signaux supplémentaires

Outre les lignes d'adresse et de données, les ROMs possèdent des signaux de contrôle supplémentaires. Ces signaux peuvent se classer en 2 catégories :

- signaux de programmation : protection contre une programmation involontaire, tension spéciale de programmation, ...
- signaux de contrôle : sélection du circuit, sélection du mode d'accès, ...

5.1.1 Exemple : PROM 16 x 8

La figure 9 donne un exemple typique de signaux associés à une PROM).

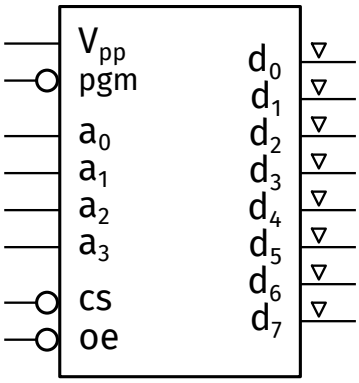


FIGURE 9 – Signaux de commande d'une PROM

Les signaux de programmation sont utilisés uniquement au moment de la programmation de la PROM. En mode de fonctionnement normal, ces entrées doivent être mises à un potentiel de l'alimentation comme spécifié par le fabricant du circuit.

La figure 10 donne une possibilité de séquencement des commandes nécessaires à la lecture de cette ROM. On y remarque que les 2 signaux $\overline{oe}$ et $\overline{cs}$ doivent être actifs pour que le composant active les lignes de données.

5.2 ROMs à accès série

Pour réduire le brochage et la complexité de raccordement de ROMs, certains fabricants proposent des mémoires à accès sériel : les données et les adresses sont transmises séquentiellement sur une même ligne et un signal d'horloge permet de coordonner le traffic des informations.

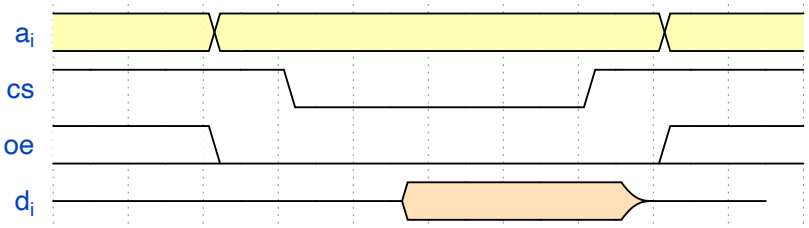
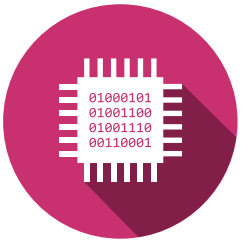


FIGURE 10 – Cycle de lecture



5.2.1 Exemple : EEPROM I2C

La figure 11 donne un exemple de signaux associés à une EEPROM série. L'accès à la mémoire se fait par les lignes **S**erial **Cl**ock (SCL) et **S**erial **Da**ta (SDA). Pour pouvoir brancher 8 composants identiques sur le même bus série, les 3 broches A2 .. A0 permettent de leur spécifier des adresses différentes. Enfin le signal de contrôle **W**rite **P**rotect (WP) permet d'empêcher l'écriture dans le circuit.

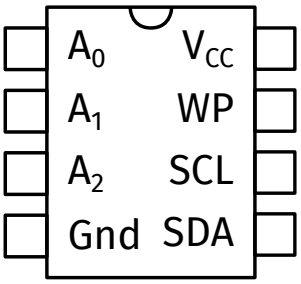


FIGURE 11 – Boîtier d'EEPROM à liaison série

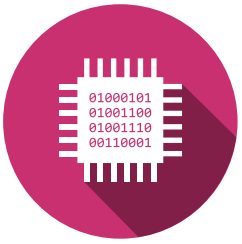
La figure 12 donne la séquence des signaux SCL et SDA pour un accès à un mot de mémoire.

FIGURE 12 – Cycle d'accèss série

5.3 Utilisation de ROMs

Les ROMs s'utilisent principalement pour mémoriser des program- mes pour des microprocesseurs : code de démarrage pour ordinateurs ou programme d'application de systèmes à microcontrôleur. A ces fins, les mémoires "flash" sont tout particulièrement désignées.

On utilise aussi des ROMs pour contenir des tables de fonctions complexes à évaluer (sinus, codes pour l'affichage de caractères...).



Références

[1] Suhail Almani. *Electronic Logic Systems*. second edition. New-Jersey : Prentice-Hall, 1989.

[2] Jean Michel Bernard et Jean Hugon. *Pratique Des Circuits Logiques*. quatrième édition. Paris : Eyrolles, 1987.

[3] Michael D. Ciletti et M. Morris Mano. *Digital Design*. second edition. New-Jersey : Prentice-Hall, 2007.

[4] Clive Maxfield. *Bebop to the Boolean Boogie*. Elsevier, 2009. ISBN : 978-1-85617-507-4. DOI : 10.1016/B978-1-85617-507-4.X0001-0. URL : https://linkinghub.elsevier.com/retrieve/pii/B9781856175074X00010 (visité le 27/05/2021).

[5] Betty Prince. *Semiconductor Memories : A Handbook of Design, Manufacture and Application*. Wiley, 1991. 830 p. ISBN : 978-0-471-92465-4. Google Books : VNsmAQAAMAAJ.

[6] Ronald J. Tocci et André Lebel. *Circuits Numériques : Théorie et Applications*. deuxième édition. Ottawa : Editions Reynald Goulet inc. / Dunod, 1996.

[7] John F. Wakerly. *Digital Design : Principles And Practices*. 3rd edition. Prentice-Hall, 2008. ISBN : 0-13-082599-9.

[8] John F. Wakerly. *Digital Design : Principles and Practices*. 3rd ed. Upper Saddle River, N.J : Prentice Hall, 2000. 949 p. ISBN : 978-0-13-769191-3.

Acronymes

EEPROM **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. 1, 11, 13

EPROM **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. 1, 9–11

OTP-ROM **O**ne **T**ime **P**rogrammable **R**ead **O**nly **M**emory. 1, 10

PROM **P**rogrammable **R**ead **O**nly **M**emory. 1, 8–10, 12

ROM **R**ead **O**nly **M**emory. 1, 3, 6–9, 12, 13

SCL **S**erial **Cl**ock. 13

SDA **S**erial **Da**ta. 13

UV **U**ltra–**V**iolet. 10

WP **W**rite **P**rotect. 13