Conception de circuits numériques et architecture des ordinateurs

Équipe pédagogique :
Marie Badaroux (TD), Julie Dumas (TD+TP), Florence Maraninchi (TD+TP),
Olivier Muller (TD+TP), **Frédéric Pétrot** (CM+TD), Pierre Ravenel (TD+TP)
Eduardo Tomasi Ribeiro (TP), Lionel Rieg (TD+TP),
Sébastien Viardot (TD+TP Apprentis)

Année universitaire 2024-2025

C1	Codage des nombres en base 2, logique booléenne, portes logiques, circuits combinatoires
C2	Circuits séquentiels
C3	Construction de circuits complexes
C4	Micro-architecture et fonctionnement des mémoires
C5	Machines à état
C6	Synthèse de circuits PC/PO
C7	Optimisation de circuits PC/PO
C8	Interprétation d'instructions - 1
C9	Interprétation d'instructions - 2
C10	Interprétation d'instructions - 3
C11	Introduction aux caches

Intérêt I

Les équipements informatiques utilisent différents types de « mémoire » avec différentes caractéristiques
Pourquoi est-ce ainsi ?
Stockage peut-être :

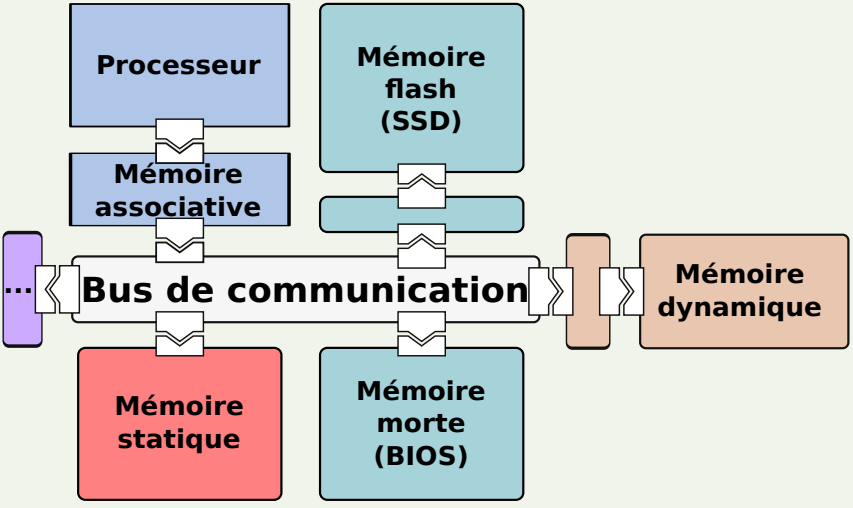
- temporaire ou pérenne
- du code et des données
- en petite ou grande quantité
- plus ou moins rapidement
- à coût faible ou élevé

Dépend de la technologie de fabrication !

fast/reliable/cheap : pick 2 !

Intérêt II

Exemple de système numérique

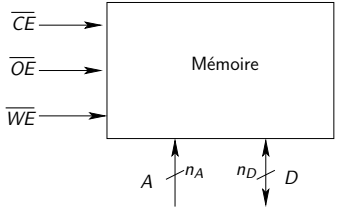


Plan détaillé du cours d'aujourd'hui

Plan

Introduction

Mémoires : regroupement massif d'éléments mémorisants



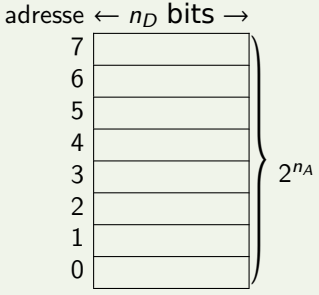
Signaux d'interface usuels : combinatoire

A adresses, sur n_A bits pour 2^{n_A} éléments (« cases mémoire »)
D données, sur n_D bits, généralement 1 ou un multiple de 8
$\overline{CE}$ *Chip Enable*, activation de la mémoire
$\overline{OE}$ *Output Enable*, génère sur D le contenu de la case d'adresse A
$\overline{WE}$ *Write Enable*, échantillonne dans case d'adresse A donnée sur D
D accessible en lecture ou en écriture à un instant donné

Vision « logicielle » de la mémoire)

Concept de base

- Vecteur de 2^{n_A} éléments, chaque élément contenant n_D bits



- `x := MEMOIRE[4]`
  met dans `x` le contenu de la case d'adresse 4
- `MEMOIRE[6] := y`
  change le contenu de la case d'adresse 6 avec la valeur de `y`

Plan

ROM, RAM

Mémoire « morte » : conserve son état même non alimentée

ROM : *Read Only Memory*

- Mémoire en lecture seulement
- Programmées à la fabrication
- Lecture : $A = a/\overline{CE} = 0/\overline{OE} = 0 \Rightarrow D = Mem(a)$

Mémoire « vive » :

RAM : *Random Access Memory*

- Nommée par opposition aux bandes et autres cartes perforées de la « préhistoire »
- Temps de lecture et d'écriture sensiblement identiques
- Généralement asynchrone, parfois synchrone (CLK)
- Statique ou dynamique (précision dans la suite du cours)
- Lecture : $A = a/\overline{CE} = 0/\overline{OE} = 0 \Rightarrow D = Mem(a)$
- Écriture : $A = a/\overline{CE} = 0/\overline{WE} = 0 \Rightarrow Mem(a) = D$

RAM

Statique (SRAM) : ≈ 6 transistors par bit

- Rapide ($\times$2-$\times$10 plus lent qu'un accès registre)
- Prédictible

Dynamique (DDR, DRAM) : ≈ 1 transistor par bit

- Coût imbattable
- Circuiterie complexe et technologie spécifique
- Latence d'accès élevée ($\times$10 plus lent que SRAM)
- Durée de rétention de l'information limitée
  Standard JEDEC : $T_r = 64$ ms à 70°
  Rafraîchir chaque point en une période T_r
- Lecture destructrice
  réécrire la valeur lue

NVRAM : *Non-volatile RAM*

Flash

- Reprogrammable, par « pages », et par des chemins détournés : interface spéciale, tension élevée, etc
- Dissymétrie des temps de lecture ($\times 1$) et d'écriture ($\times 10$-$\times 100$)
- Lecture par « pages » de typiquement 2048+64 octets (NAND, carte SD, disques SSD) ou par mot (NOR, firmware, ROM de configuration), écriture par page
- Nombre de programmation limité : de 1 (OTP) à 100K
- Durée de rétention de 10 à 20 ans
- Lecture : $A = a / \overline{CE} = 0 / \overline{OE} = 0 \Rightarrow D = Mem(a)$

Avancées technologiques : MRAM, STT-RAM, PCRAM, ...

Nouvelles technologies prometteuses

- Densité élevée (entre SRAM et DRAM)
- Baisse des temps d'accès lecture/écriture
- Accès mot ou ligne
- Consommation maîtrisée
- Temps d'accès encore supérieur à SRAM
- Endurance parfois problématique

À suivre dans les années qui viennent !

Types de mémoire

CAM : *Content-Addressable Memory*

- Mémoire statique « associative »
  constituée un point mémoire à 6 transistors et 1 comparateur à 3 transistors
- Usages spécialisés pour la recherche de la présence d'une donnée dans un ensemble
  prend une donnée et produit l'adresse à laquelle elle se trouve
- Peu répandue en volume, mais incontournable (caches processeur, routeurs réseau, ...)

Plan

En pratique

Vision logicielle ≈ Image d'Épinal

Ça se complique, ...

- Taille minimum des accès inférieure à n_D
  accès à 8, 16, 24 ou 32 bits dans une mémoire avec $n_D = 32$
- Taille maximum des accès supérieure à n_D
  accès 64 bits dans une mémoire avec $n_D = 32$

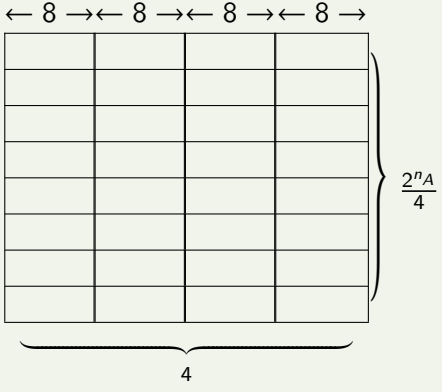
On ne peut pas vivre avec un vecteur !

- Questions soulevées :
  - Quels accès sont aisément réalisables en matériel ?
  - Quelles sont les *adresses* des éléments en fonction de leur taille ?

Organisation matérielle

Organisation physique matricielle

Matrice de 2^{n_A} éléments de taille 8, avec $\frac{2^{n_A}}{4}$ lignes et 4 colonnes



Lecture ou écriture systématique des 4 colonnes lors d'un accès

Accès aisément réalisables

Rappel : $n_D = 32$

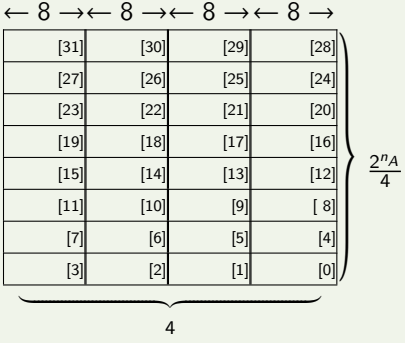
Ceux *à l'intérieur d'une ligne*

- Faisables en un seul accès si taille 8, 16, 24 et 32
  8 bits : n'importe où
  autres : ne peuvent franchir une frontière entre deux lignes
- Contrainte supplémentaire : absence de recouvrement de deux mots de taille identique
  16 bits : 2 premières cases ou 2 dernières cases de la ligne
  24 bits : sans intérêt, car ça prend forcément 32 bits !
  32 bits : les 4 cases d'une ligne
- Cas 64 bits : deux accès 32 bits à deux lignes successives

Adresses

Utilisation d'adresses octet : identifie l'élément minimal à rechercher

- Les contraintes précédentes imposent :
  Adresse élément de 8 bits multiple de 1
  Adresse élément de 16 bits multiple de 2 (A[0]=0)
  Adresse élément de 32 bits multiple de 4 (A[1 :0]=00)
  Adresse élément de 64 bits multiple de 8 (A[2 :0]=000)
  Contraintes dites « d'alignement »

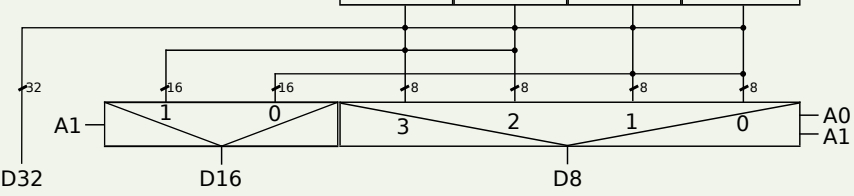


A quoi sont dues ces contraintes ?

Faire du matériel (conceptuellement relativement) simple !

Exemple de la lecture

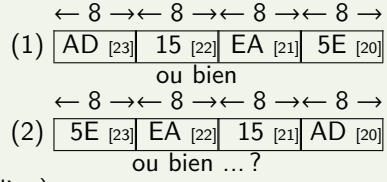
[31]	[30]	[29]	[28]
[27]	[26]	[25]	[24]
[23]	[22]	[21]	[20]
[19]	[18]	[17]	[16]
[15]	[14]	[13]	[12]
[11]	[10]	[9]	[8]
[7]	[6]	[5]	[4]
[3]	[2]	[1]	[0]



Boutisme (endianness)

Ordre des octets dans un mot de (par ex.) 32 bits

Question : comment stocker les octets d'un mot de 32 bits en mémoire ?
Réponse : ça dépend !
valeur (en hexa) $\underbrace{\texttt{AD}}_{\substack{\text{poids}\\\text{fort}}}\texttt{15EA}\underbrace{\texttt{5E}}_{\substack{\text{poids}\\\text{faible}}}$ [a] à stocker à l'adresse 20



(1) Petit boutisme (little endian)
adresse du mot = adresse de l'octet de poids faible

(2) Grand boutisme (big endian)
adresse du mot = adresse de l'octet de poids fort

a. *Endianness is AD15EA5E.*

Boutisme (endianness) I

MSB : *Most Significant Byte*, octet de poids fort
LSB : *Least Significant Byte*, octet de poids faible

Adresses mot de 32 bits toujours alignées sur un multiple de 4

Little endian

MSB[15]	[14]	[13]	LSB[12]
MSB[11]	[10]	[9]	LSB [8]
MSB [7]	[6]	[5]	LSB [4]
MSB [3]	[2]	[1]	LSB [0]

Big endian

LSB[15]	[14]	[13]	MSB[12]
LSB[11]	[10]	[9]	MSB [8]
LSB [7]	[6]	[5]	MSB [4]
LSB [3]	[2]	[1]	MSB [0]

Adresses mot de 16 bits toujours alignées sur un multiple de 2

Little endian

MSB1[15]	LSB1[14]	MSB0[13]	LSB0[12]
MSB1[11]	LSB1[10]	MSB0 [9]	LSB0 [8]
MSB1 [7]	LSB1 [6]	MSB0 [5]	LSB0 [4]
MSB1 [3]	LSB1 [2]	MSB0 [1]	LSB0 [0]

Big endian

LSB1[15]	MSB1[14]	LSB0[13]	MSB0[12]
LSB1[11]	MSB1[10]	LSB0 [9]	MSB0 [8]
LSB1 [7]	MSB1 [6]	LSB0 [5]	MSB0 [4]
LSB1 [3]	MSB1 [2]	LSB0 [1]	MSB0 [0]

Boutisme (endianness) II

Adresses mot de 64 bits toujours alignées sur un multiple de 8

Little endian

MSB1[15]	[14]	[13]	[12]
[11]	[10]	[9]	LSB1[8]
MSB0 [7]	[6]	[5]	[4]
[3]	[2]	[1]	LSB0[0]

Big endian

LSB1[15]	[14]	[13]	[12]
[11]	[10]	[9]	MSB1[8]
LSB0 [7]	[6]	[5]	[4]
[3]	[2]	[1]	MSB0[0]

Tous les goûts sont dans la nature

- Processeurs Little et/ou Big, dépend du constructeur, du modèle, ...
- Source de problèmes
  processeurs récents configurables en Little ou Big

Plan

Micro-architecture des mémoires

Attention !

Mémoire ≠ Banc de registres

Pourquoi ?

Problème de gros sous

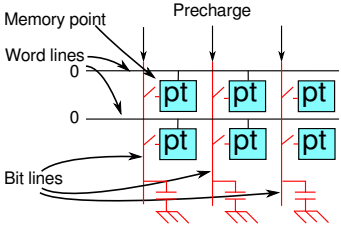
- Registre ≈ 12 transistors/bit + beaucoup de connexions
- SRAM ≈ 6 transistors/bit + partage connexions
- ROM, DRAM, FLASH ≈ 1 transistor/bit + partage connexions

Prix à payer

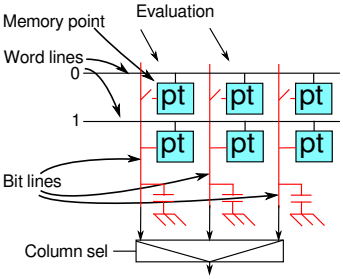
- Complexité des circuits (c.f. pour aller plus loin)
- Temps d'accès mémoire > (voire >>) temps d'accès registre
- Granularité d'accès : ensemble de n mots
- PAS D'ACCÈS SIMULTANÉ EN LECTURE ET EN ÉCRITURE [a]

a. C'est clair ?

Principe de fonctionnement : lecture



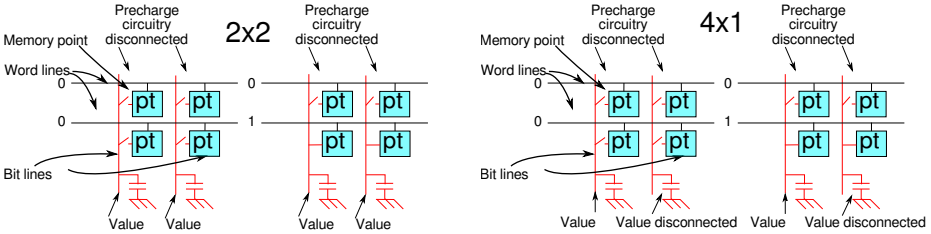
- Phase de précharge :
  toutes WL = 0 puis mise à 1 des BL
  (charge de capacités)
- Phase d'evaluation :
  Une WL et une seule = 1,
  si PT à 0, décharge des capacités
  si PT à 1, rien ne se passe



BL évaluées simultanément
Valeur de la BL est valeur du point mémoire WL/BL

Organisations :
2 mots de 3 bits BL
6 mots de 1 bit mux 3 → 1 sur BL

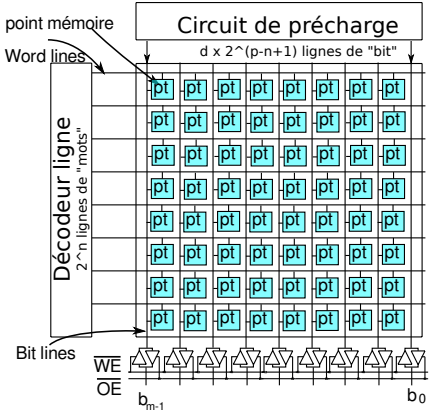
Principe de fonctionnement : écriture



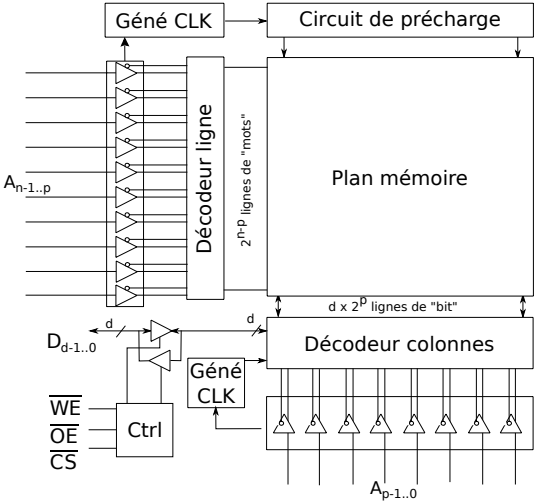
- Phase de positionnement :
  toutes WL = 0 puis mise des BL à leurs valeurs
  BL des « autres » mots d'une WL disconnectées (cas 4×1 ici)
- Phase de mémorisation :
  Une WL et une seule = 1, PT prends la valeur de la BL

Toutes les BL sont positionnées en même temps
Valeur du point mémoire WL/BL est celle de BL

Plan mémoire



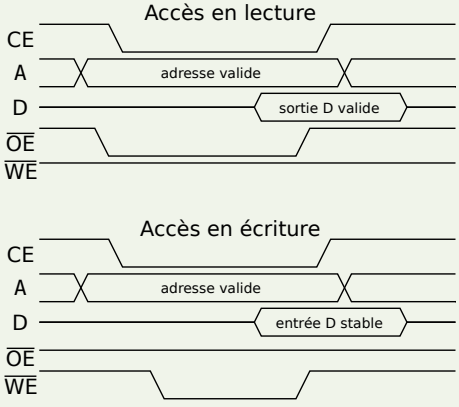
Micro-architecture des mémoires



Plan

Protocole d'accès

Spécifications temporelles sous forme de chronogrammes



Respect de délais spécifiques (et précis) entre les différents signaux pour un fonctionnement correct

Plan

Wrap-up

Mémoire coté logiciel

- tableau indexé avec des adresses d'**octets**
- **contraintes** sur les index qui dépendent du **type** de la donnée
- placement des octets dans les types parfois utile à connaître

Mémoire coté matériel

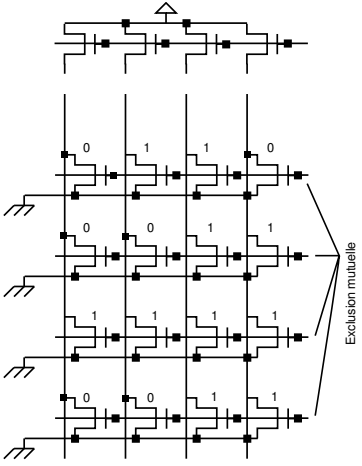
- différentes technologies pour différents usages
- basées sur le même principe :
  - décoder poids fort adresses pour sélectionner une ligne
  - décoder poids faible adresses pour sélectionner un sous ensemble de colonnes
  - lecture **ou** écriture à un instant t
  - protocole spécifique pour l'accès

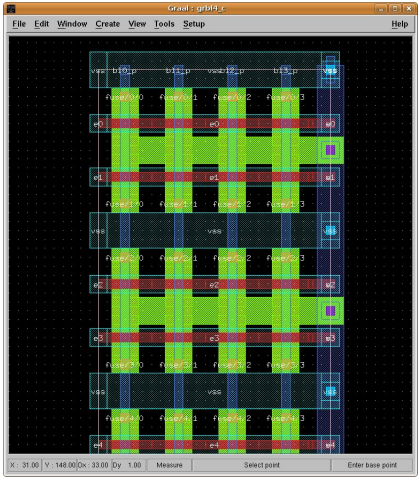
Plan

Pour aller plus loin

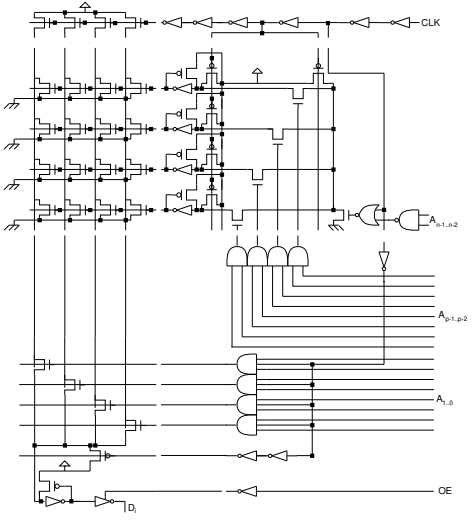
À ne lire que si vous voulez en savoir plus !

Point mémoire ROM





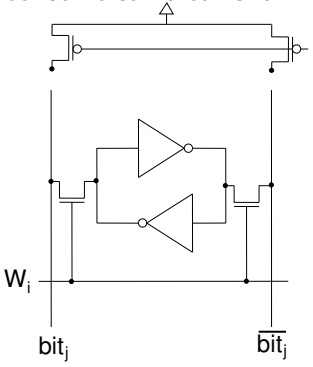
Micro-architecture d'une ROM

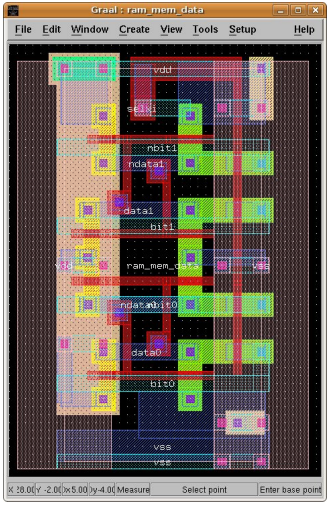


- W_i sélectionne une ligne de mots
- Principe :
  - $CLK = 1$: Précharge à 1 des lignes de bits
  - $CLK = 0$: sélection W_i par les $A_{3..2}$, mise à la masse des lignes à travers les TN connectés,
    sélection d'1 bit parmi 4 par $A_{1..0}$
  - Note : les mêmes bits de 4 mots successifs sont voisins dans cette architecture

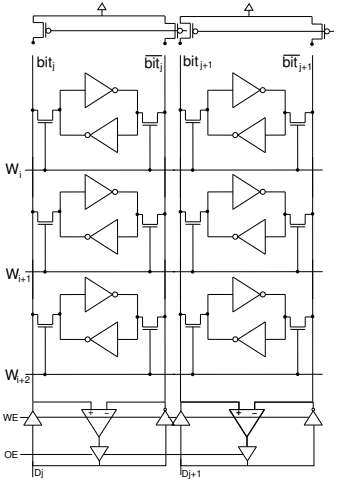
Point mémoire SRAM

Point mémoire statique :
conserve sa valeur si alimenté





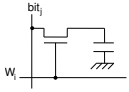
Micro-architecture d'une SRAM

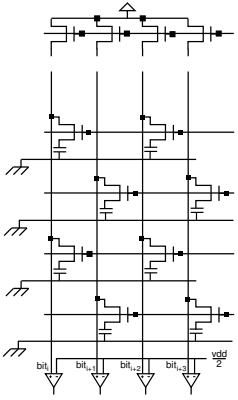


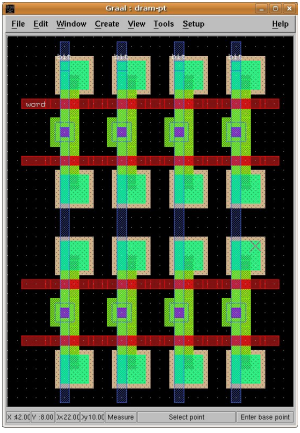
- Mux possible en sortie des *sense amplifiers*
- Lecture :
  - Précharge à 1 des lignes *bit* et $\overline{bit}$
  - Évaluation par mise à 1 du bon W_i
  - Sense accélère le choix dès que $\Delta(V_{bit}, V_{\overline{bit}}) \approx 10$ mV
- Écriture :
  - Force $bit = D_j$ et $\overline{bit} = \overline{D_j}$
  - Mise à 1 du bon W_i
  - Conflit électrique
    gros *drivers* sur lignes de bits

Mémoire DRAM

Dynamique : perte de l'information au cours du temps

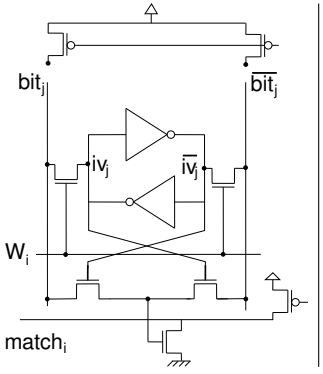




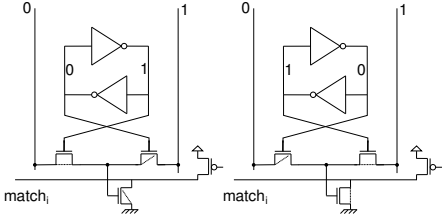


Capacités non-visibles : construites verticalement dans le substrat de silicium

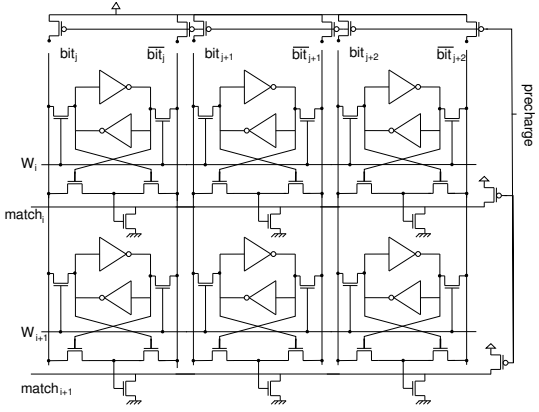
Mémoire CAM



Mémoire SRAM usuelle
Intègre un comparateur
Compare bit_j et iv_j
Précharge : $bit_j = \overline{bit_j} = hz$
Évaluation : bit_j et $\overline{bit_j}$ positionnés, $match_i$ calculé



Mémoire CAM



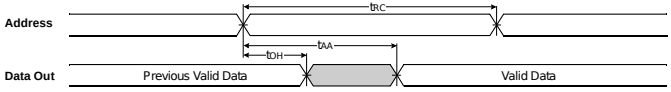
- *match line* préchargée à 1
- *match line* partagée par l'ensemble des bits d'un mot

match$_i$ = 0 si il existe au moins un bit du mot tel que

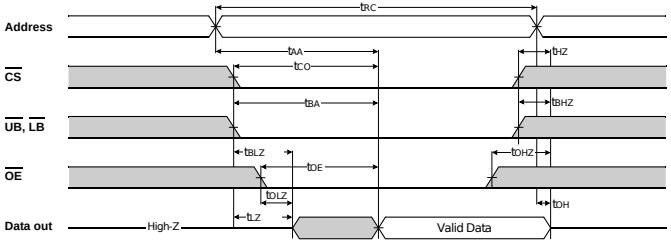
$\overline{bit_j} = 0$ et $iv_j = 1$
ou
$bit_j = 0$ et $\overline{iv_j} = 1$

Protocole d'accès à une SRAM : chronogramme lecture

TIMING WAVEFORM OF READ CYCLE(1) (Address Controlled, $\overline{CS}$=$\overline{OE}$=V_{IL}, $\overline{WE}$=V_{IH}, $\overline{UB}$, $\overline{LB}$=V_{IL})



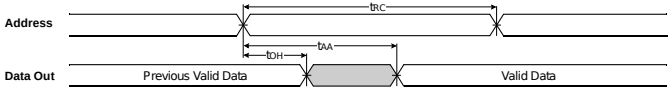
TIMING WAVEFORM OF READ CYCLE(2) ($\overline{WE}$=V_{IH})



Protocole d'accès à une SRAM : temps lecture

Samsung K6R4016V1D, 256Kx16 Bit CMOS Static RAM

TIMING WAVEFORM OF READ CYCLE(1) (Address Controlled, $\overline{CS}$=$\overline{OE}$=V_{IL}, $\overline{WE}$=V_{IH}, $\overline{UB}$, $\overline{LB}$=V_{IL})



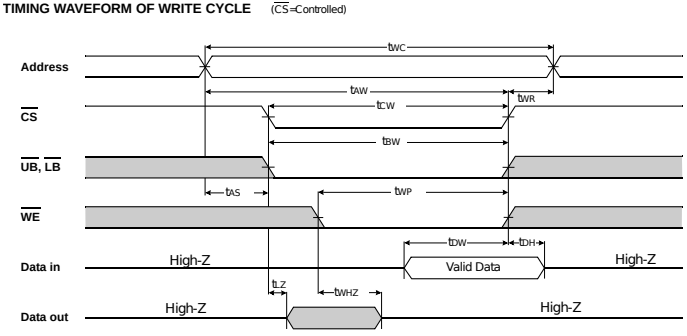
READ CYCLE*

Parameter	Symbol	K6R4016V1D-08		K6R4016V1D-10		Unit
		Min	Max	Min	Max	
Read Cycle Time	t_{RC}	8	-	10	-	ns
Address Access Time	t_{AA}	-	8	-	10	ns
Chip Select to Output	t_{CO}	-	8	-	10	ns
Output Enable to Valid Output	t_{OE}	-	4	-	5	ns
$\overline{UB}$, $\overline{LB}$ Access Time	t_{BA}	-	4	-	5	ns
Chip Enable to Low-Z Output	t_{LZ}	3	-	3	-	ns
Output Enable to Low-Z Output	t_{OLZ}	0	-	0	-	ns
$\overline{UB}$, $\overline{LB}$ Enable to Low-Z Output	t_{BLZ}	0	-	0	-	ns
Chip Disable to High-Z Output	t_{HZ}	0	4	0	5	ns
Output Disable to High-Z Output	t_{OHZ}	0	4	0	5	ns
$\overline{UB}$, $\overline{LB}$ Disable to High-Z Output	t_{BHZ}	0	4	0	5	ns
Output Hold from Address Change	t_{OH}	3	-	3	-	ns
Chip Selection to Power Up Time	t_{PU}	0	-	0	-	ns
Chip Selection to Power DownTime	t_{PD}	-	8	-	10	ns

* The above parameters are also guaranteed at industrial temperature range.

Protocole d'accès à une SRAM : chronogramme écriture

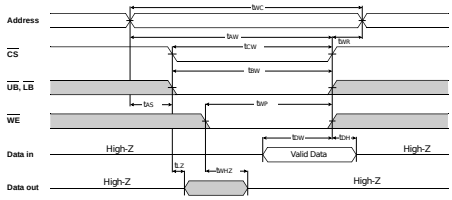
Samsung K6R4016V1D, 256Kx16 Bit CMOS Static RAM



Protocole d'accès à une SRAM : temps écriture

Samsung K6R4016V1D, 256Kx16 Bit CMOS Static RAM

TIMING WAVEFORM OF WRITE CYCLE ($\overline{CS}$-Controlled)



WRITE CYCLE*

Parameter	Symbol	K6R4016V1D-08		K6R4016V1D-10		Unit
		Min	Max	Min	Max	
Write Cycle Time	t_{WC}	8	-	10	-	ns
Chip Select to End of Write	t_{CW}	6	-	7	-	ns
Address Set-up Time	t_{AS}	0	-	0	-	ns
Address Valid to End of Write	t_{AW}	6	-	7	-	ns
Write Pulse Width($\overline{OE}$ High)	t_{WP}	6	-	7	-	ns
Write Pulse Width($\overline{OE}$ Low)	t_{WP1}	8	-	10	-	ns
$\overline{UB}$, $\overline{LB}$ Valid to End of Write	t_{BW}	6	-	7	-	ns
Write Recovery Time	t_{WR}	0	-	0	-	ns
Write to Output High-Z	t_{WHZ}	0	4	0	5	ns
Data to Write Time Overlap	t_{DW}	4	-	5	-	ns
Data Hold from Write Time	t_{DH}	0	-	0	-	ns
End of Write to Output Low-Z	t_{OW}	3	-	3	-	ns

* The above parameters are also guaranteed at industrial temperature range.