ASR1 – TD3 : Mémoires

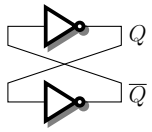
{ Andreea.Chis, Matthieu.Gallet, Bogdan.Pasca } @ens-lyon.fr

2 et 3 Octobre 2008

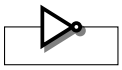
1 Construisons un registre

On se place ici en technologie CMOS, qui nous permet de construire uniquement des portes non-et (NAND) et des portes non-ou (NOR) avec un nombre quelconque d'entrées – la porte non (NOT) étant un cas particulier de ces portes.

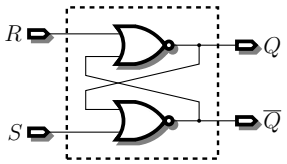
1. Quelle est la structure la plus simple permettant de "se souvenir" d'un état ? Il n'est pas demandé que l'on puisse initialiser l'état ; il suffit que la structure soit capable de se souvenir de l'importe quel état (0 ou 1).
Réponse :
C'est bien-sûr le bistable, consitué d'une boucle de deux inverseurs en série. Il stocke un état Q ainsi que son complémentaire $\overline{Q}$.



2. Tant que vous y êtes, construisez un oscillateur.
Réponse :
La même chose, avec un seul inverseur (ou tout du moins un nombre impair).

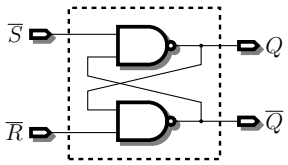


3. Sachant que l'on s'interdit de faire des court-circuits et autres cochonneries du même acabit, dessinez une bascule RS. Une bascule RS (ou *RS latch*) est une structure à deux entrées (R et S) et deux sorties (Q et $\overline{Q}$). La sortie Q garde l'état courant quand R et S sont à 0, passe à 0 lorsque R (*reset*) vaut 1, et passe à 1 lorsque S (*set*) vaut 1. (On suppose que R et S ne sont jamais simultanément à 1.)
Réponse :
On part du bistable et on transforme les inverseurs en portes NOR pour pouvoir forcer la valeur de Q à 0 lorsque R vaut 1, ou $\overline{Q}$ à 0 lorsque S vaut 1 :



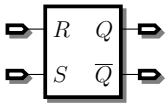
$$\begin{cases} Q = \overline{\overline{Q} \vee R}, \text{ et} \\ \overline{Q} = \overline{Q \vee S}. \end{cases}$$

De même, en utilisant plutôt des portes NAND et les entrées $\overline{R}$ et $\overline{S}$ plutôt que R et S, on a :

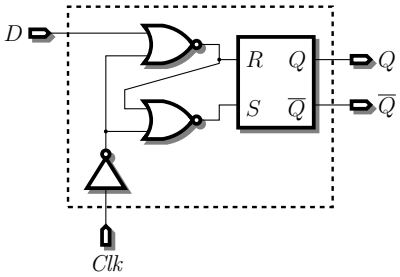


$$\begin{cases} Q = \overline{\overline{Q} \wedge \overline{S}}, \text{ et} \\ \overline{Q} = \overline{Q \wedge \overline{R}}. \end{cases}$$

On schématise la bascule RS de la manière suivante :

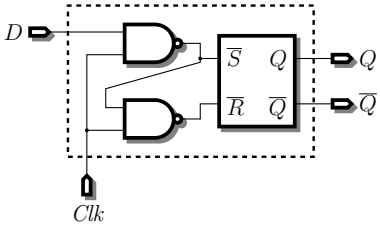


4. Réutilisez la bascule RS pour définir un registre sur état (aussi appelée bascule D, ou encore *D latch*) : une structure à deux entrées (D et Clk) et deux sorties (Q et $\overline{Q}$) qui recopie D (*data*) sur Q lorsque Clk (*clock*, l'horloge) vaut 1, et qui garde l'état courant lorsque Clk vaut 0.
Réponse :
L'entrée R sur la bascule RS doit valoir 1 lorsque la donnée D vaut 0 et que l'horloge Clk est à 1, et S doit valoir 1 lorsque D vaut 1 et que Clk est à 1. Par contre, R et S doivent être fixées à 0 lorsque Clk vaut 0 :



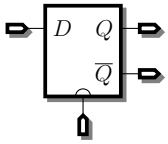
$$\begin{cases} R = \overline{D} \wedge Clk, \text{ et} \\ S = D \wedge Clk. \end{cases}$$

On a la même chose avec des portes NAND :



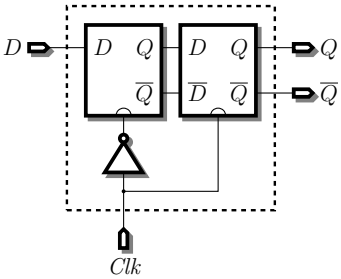
$$\begin{cases} \overline{R} = \overline{\overline{D} \wedge Clk}, \text{ et} \\ \overline{S} = \overline{D \wedge Clk}. \end{cases}$$

La bascule D se schématise de la manière suivante (notez l'arc de cercle symbolisant le fil d'horloge, avec transition sur état) :

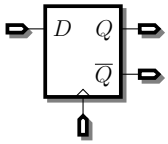


5. Réutilisez la bascule D pour définir un registre sur front (aussi appelé *D flip-flop*) : une structure à deux entrées (D et Clk) et deux sorties (Q et $\overline{Q}$) qui recopie D sur Q lorsque Clk passe de 0 à 1 (*front montant*), et qui garde l'état courant le reste du temps.
Réponse :
Le registre se construit à l'aide de deux bascules D placées en série : la première est passante

sur l'horloge basse, la seconde sur l'horloge haute. Ainsi, la valeur de D est préchargée dans la première bascule lorsque l'horloge vaut 0, puis transférée dans la seconde bascule quand l'horloge passe à 1. L'opposition de phase des deux bascules permet ensuite d'isoler les sorties Q et $\overline{Q}$ jusqu'au prochain front montant.

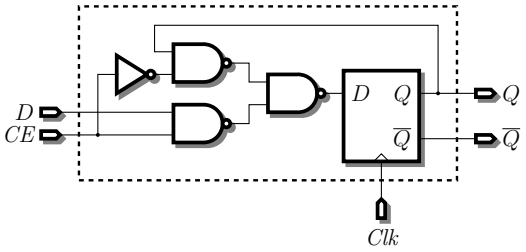


Il est schématisé par le symbole suivant (notez l'angle sur le fil d'horloge, qui dénote la sensibilité sur front) :

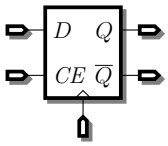


6. Philosophez sur l'intérêt fondamental du registre sur front.
Réponse : Il est plus facile de synchroniser sur un événement discret et ponctuel que sur toute la durée d'un état.

7. Dessinez un registre "débrayable" : rajoutez une entrée CE telle que la recopie de D sur Q ne se fait que si CE (*clock enable*) est à 1 lors du front montant de Clk.
Réponse :
On rajoute un multiplexeur sur un registre D, de sorte à ce que l'entrée D de ce registre soit l'entrée D globale lorsque CE est à 1, et la sortie Q sinon.

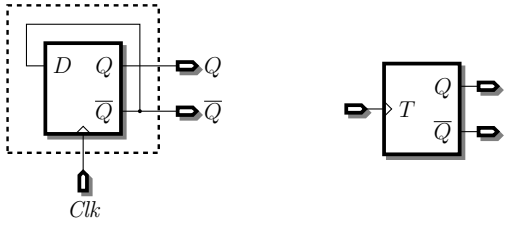


Son schéma est comme suit :

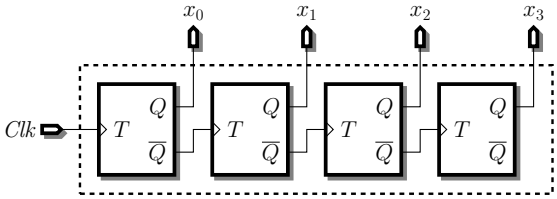


2 Compteurs

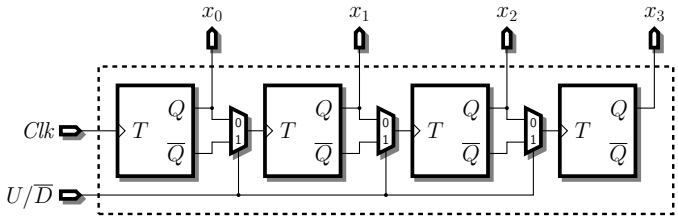
1. Dessinez un circuit qui compte modulo 2^n.
Réponse :
On définit le registre T, qui change d'état sur chaque front montant de son horloge Clk. On prend pour cela un registre D dont la sortie inversée $\overline{Q}$ reboucle sur l'entrée D :



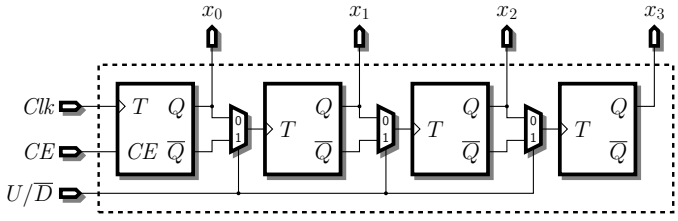
Il suffit alors de mettre plusieurs registres T en série, la sortie $\overline{Q}$ de l'un étant connectée à l'horloge du suivant. On obtient ainsi un compteur binaire, comme représenté ci-dessous pour 4 bits (les poids faibles sont à gauche) :



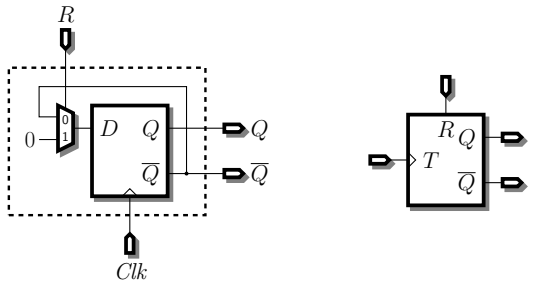
2. Étendez-le en un compteur/décompteur. (Rajoutez une entrée qui dit si on compte ou si on décompte.)
Réponse :
Le circuit pour décompter est identique au précédent, à la seule différence que c'est la sortie Q et non $\overline{Q}$ d'un registre qui va sur l'entrée T du suivant. Il suffit alors de multiplexer Q ou $\overline{Q}$ pour l'entrée T, selon la valeur du signal $U/\overline{D}$ (pour Up/$\overline{\text{Down}}$*, valant 1 si l'on compte vers le haut, et 0 vers le bas).*



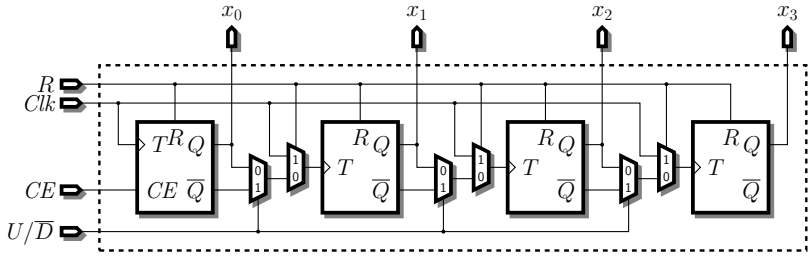
3. Rajoutez une entrée qui désactive le comptage/décomptage.
Réponse :
Il suffit de rajouter un clock enable *sur le premier registre T.*



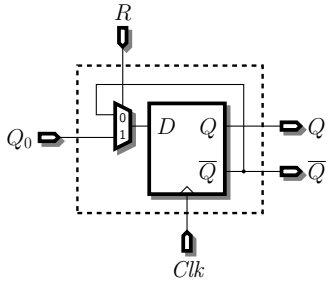
4. Rajoutez une entrée de remise à zéro (reset) synchrone.
Réponse :
On modifie nos registres T de sorte à leur rajouter une entrée reset R *synchrone :*



Il faut ensuite s'assurer que lorsque R est à 1, tous les registres passent à 0 sur le front montant de l'horloge Clk. Il faut donc multiplexer l'horloge des registres selon R.



5. Rajoutez la possibilité de précharger le compteur/décompteur à une valeur donnée en entrée.
Réponse :
On fait comme pour la mise à zéro, sauf qu'au lieu de sélectionner entre $\overline{Q}$ et 0 pour l'entrée de chaque registre, R peut sélectionner une valeur Q_0 donnée par un autre fil :



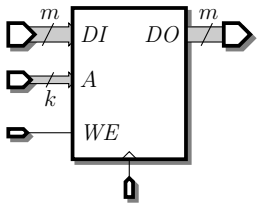
3 Mémoire adressable

On dit qu'une mémoire adressable est de type $2^k \times m$ si elle contient 2^k mots de m bits, adressés par une adresse sur k bits.

1. Construisez, en utilisant des registres et des portes logiques de base, une mémoire $2^1 \times 1$, une mémoire $2^2 \times 1$, une mémoire $2^k \times 1$, une mémoire $2^k \times m$.
Réponse :
L'interface d'une mémoire $2^k \times m$ comprend :
– *un signal d'adresse A sur k bits,*
– *un signal de données entrantes DI sur m bits,*

– *un signal de données sortantes DO sur m bits,*
– *un signal d'horloge Clk, et*
– *un signal* write enable *WE.*

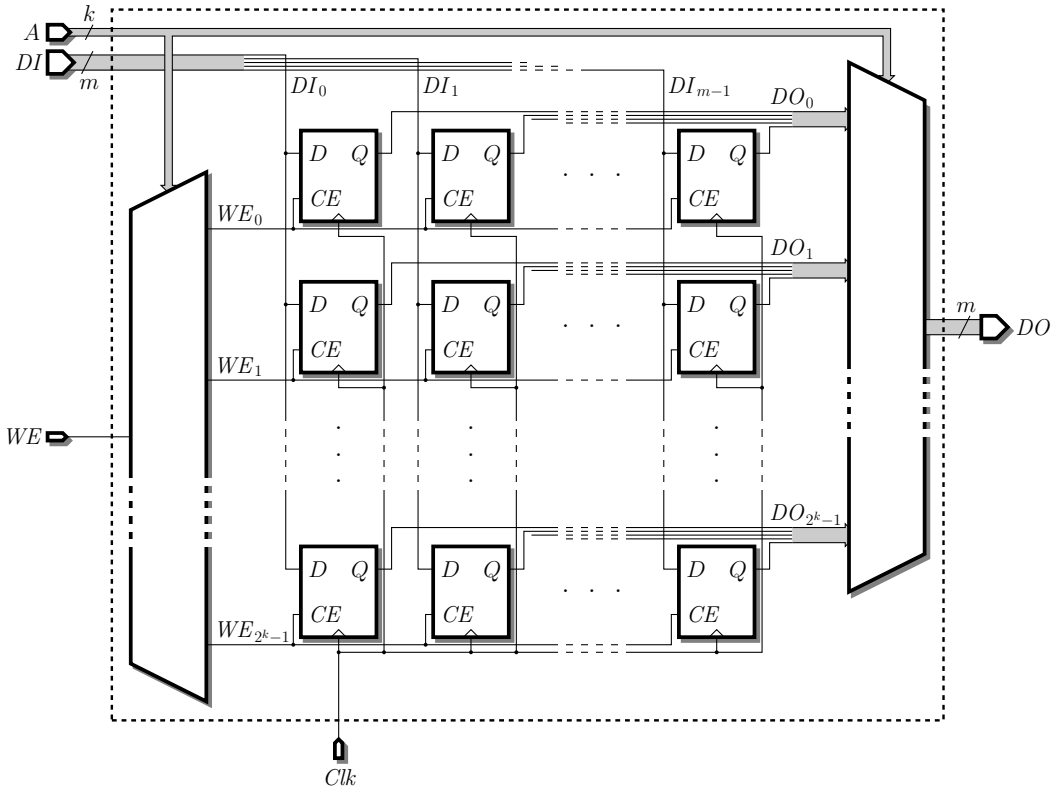
On la schématise de la manière suivante :



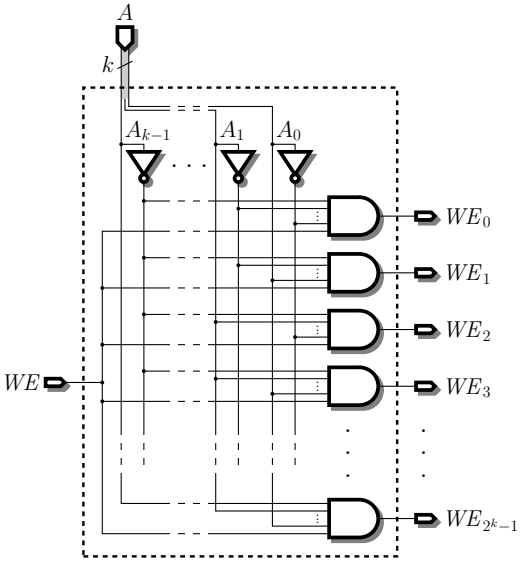
Elle repose sur une grille de 2^k lignes de m registres D munis d'un clock enable*. Chacun des m bits de DI est distribué aux 2^k registres correspondants, le choix du registre dans lequel la donnée sera écrite étant réalisé grâce aux CE.*

En effet, en démultiplexant le signal WE selon l'adresse A, on obtient 2^k signaux WE_i, chacun distribué aux registres de la ligne correspondante.

Enfin, les 2^k sorties DO_i des lignes de registres sont multiplexée selon A pour sélectionner la valeur de la ligne demandée.



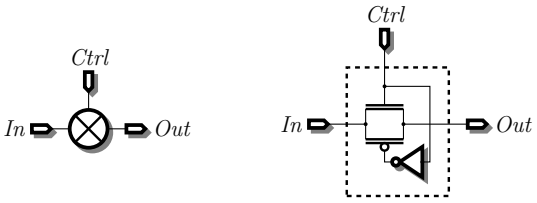
En détail ici, la construction du démultiplexeur pour WE :



2. En plus des portes logiques, on se donne la porte trois-états du cours. Mêmes questions. Discutez des avantages et des inconvénients.

Réponse :

La porte trois-états (aussi appelée porte de transmission) agit comme un interrupteur. Ci-dessous son schéma et sa réalisation à l'aide d'un inverseur et de deux transistors :

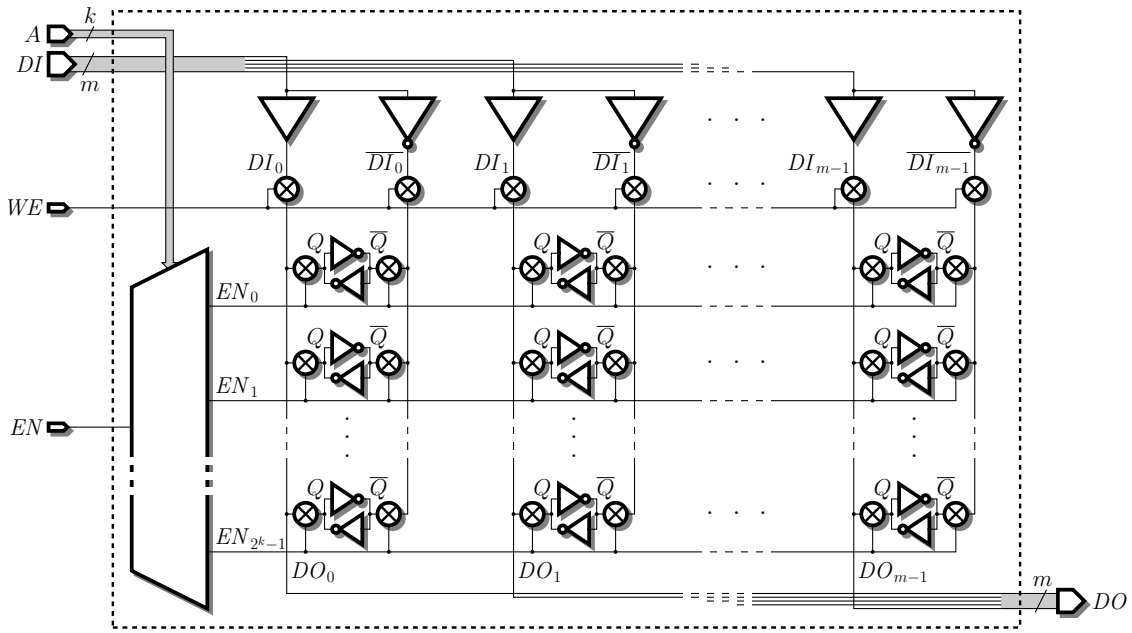


On se donne ce coup-ci une interface sensiblement différente pour la mémoire. On supprime le signal d'horloge Clk et on rajoute un signal enable *noté EN qui contrôle le moment où les données sont lues ou écrites.*

C'est ce signal EN qui va être démultiplexé selon l'adresse A en $2^k - 1$ lignes EN_i. De même que pour la version précédente, le signal EN_i de chaque ligne va être distribué aux m cellules mémoire composant cette ligne.

Pour éviter d'utiliser un multiplexeur pour sélectionner la bonne valeur de DO, on fait ici appel aux portes trois-états : on se donne m fils verticaux DO_j, correspondant chacun à un bit de DO. Grâce aux portes trois-états, on peut isoler ce fil DO_j de la sortie Q de chacun des éléments de mémorisation correspondant au bit de rang j, sauf pour la ligne i sélectionnée par le signal EN_i.

Pour pouvoir écrire dans un élément de mémorisation, on peut réutiliser ces fils verticaux DO_j et imprimer la valeur DI_j dessus, avec suffisamment de puissance pour écraser l'ancienne valeur Q contenue dans les cellules sélectionnées. Comme les éléments de mémorisation choisis ici sont de simples bistables, il faut aussi changer la valeur de $\overline{Q}$ en même temps que celle de Q. On dédouble donc les fils verticaux pour former les paires DI_j et $\overline{DI_j}$. L'amplification de ces deux signaux est réalisée avec de gros buffers / inverseurs. Enfin, le signal WE vient contrôler des portes trois-états sur ces fils pour les déconnecter des colonnes de mémorisation en cas de lecture.



3. Quelle est la différence entre espace d'adressage et taille mémoire ?
Réponse :
Une mémoire $2^k \times m$ a un espace d'adressage de 2^k mots (c'est-à-dire le nombre d'adresses possibles) mais une taille mémoire de $2^k \times m$ bits.

4. On dispose de boîtiers $32\text{K} \times 8$, combien y a-t-il de broches d'adresse et de broches de données ?
Réponse :
Il nous faut 15 broches d'adresse, 8 broches de données entrantes et 8 broches de données sortantes. Plus bien-sûr tous les signaux de contrôle (EN, WE, ...).

5. On veut fabriquer une mémoire 1M mots avec des mots de 32 bits avec ces boîtiers, comment fait-on ?
Réponse :
Pour une mémoire de 1M mots, il nous faut 20 bits d'adresse : il nous faut donc 32 banques de boîtiers. Les 5 bits de poids fort de l'adresse sélectionnent la banque, et les 15 bits de poids faible adressent les boîtiers de chaque banque. Enfin, comme on veut des mots de 32 bits, cela nous fait 4 boîtiers par banque.

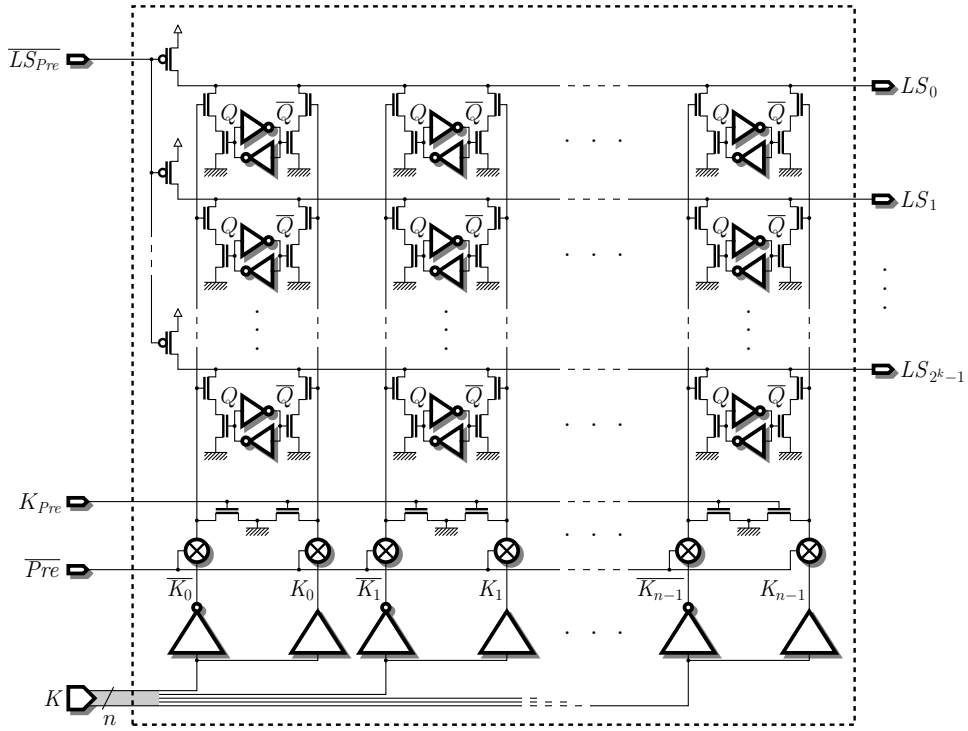
6. Dessinez une mémoire associative à 4 positions, à clefs sur 4 bits et à données sur 4 bits.
Réponse :
Une mémoire associative (ou CAM, *pour* Content-Addressable Memory*) fonctionne exactement comme une mémoire classique sur des mots de $n+m$ bits, mais où les n bits de poids fort de ces mots jouent un rôle de clef. Il est en effet toujours possible d'accéder (en lecture et en écriture) au contenu d'une ligne suivant son adresse A, mais un mécanisme supplémentaire de sélection de ligne de données permet aussi d'accéder à une ligne suivant la valeur de sa clef.*

En plus des ports classiques, la mémoire associative reçoit en entrée une clef recherchée K, sur n bits, ainsi qu'un signal de contrôle sélectionnant l'accès par adresse ou bien par clef. Dans le cas de l'accès par clef, la clef K est recherchée parmi toutes les clefs des mots stockés dans la mémoire, et la ligne correspondante (si elle existe) est alors sélectionnée.

Les mémoires associatives sont utilisées pour réaliser des caches, des TLB et des tables de routage, entre autres.

Le principe pour comparer une clef sotckée sur la ligne i à la clef recherchée est de tirer un fil horizontal LS_i (pour line select*) que l'on précharge à 1 à l'aide d'un transistor P et d'un signal de contrôle $\overline{LS_{Pre}}$.*

*Ce fil parcourt ensuite tous les n bits de la clef stockée en les comparant un à un aux bits K_j de la clef recherchée. À la moindre différence détectée, un mécanisme (*pull-down*) à base 4 transistors N décharge la ligne LS_i. Ainsi, seule une ligne correspondant à la bonne clef sera sélectionnée, car son signal LS_i se sera bien maintenu à 1.*



Notez ici que la précharge à 1 des lignes LS_i demande de déconnecter d'abord ces lignes des éléments de mémorisation, ce qui se fait en préchargeant les fils K_j et $\overline{K_j}$ à 0, grâce au signal K_{Pre}. Cette précharge requiert à son tour l'isolation des fils K_j et $\overline{K_j}$, grâce à des portes trois-états contrôlées par le signal $\overline{Pre}$.