Multiplication binaire

Trop difficile pour les circuits combinatoires

Il faut utiliser un circuit séquentiel

Pour chaque front d'horloge, une étape du calcul est effectuée

On utlise l'algorithme habituel, mais avec accumulateur:

```
      1 1 0 1
        1 0 1
-------------
      1 1 0 1
    0 0 0 0
  1 1 0 1
-------------
1 0 0 0 0 0 1
```

```
      1 1 0 1
        1 0 1
-------------
      1 1 0 1  <---  accumulateur
    0 0 0 0
-------------
      1 1 0 1  <---  accumulateur
  1 1 0 1
-------------
1 0 0 0 0 0 1  <---  accumulateur
```

Multiplication binaire (suite)

On note les facteurs x et y, et le resultat r

Si x et y sont de n bits, alors r peut contenir 2n bits

On écrit y de la façon suivante:

$$y = y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \ ... \ + y_0 \ 2^0$$

Après étape numéro i, l'accumulateur contient:

$$x * (y_{i-1}2^{n-1} + y_{i-2}2^{n-2} + \ ... \ + y_0 \ 2^{n-i})$$

Donc, quand i = n, alors, l'accumulateur contient le resultat, r

Multiplication binaire (suite)

Pour passer d'une étape à l'étape suivante, on fait la chose suivante:

$$r_{i+1} = r_i / 2 + x * y_i 2^{n-1}$$

Car:

$$r_i / 2 + x * y_i 2^{n-1} =$$

$$x * (y_{i-1} 2^{n-1} + y_{i-2} 2^{n-2} + \ldots + y_0 2^{n-i}) / 2 + x * y_i 2^{n-1} =$$

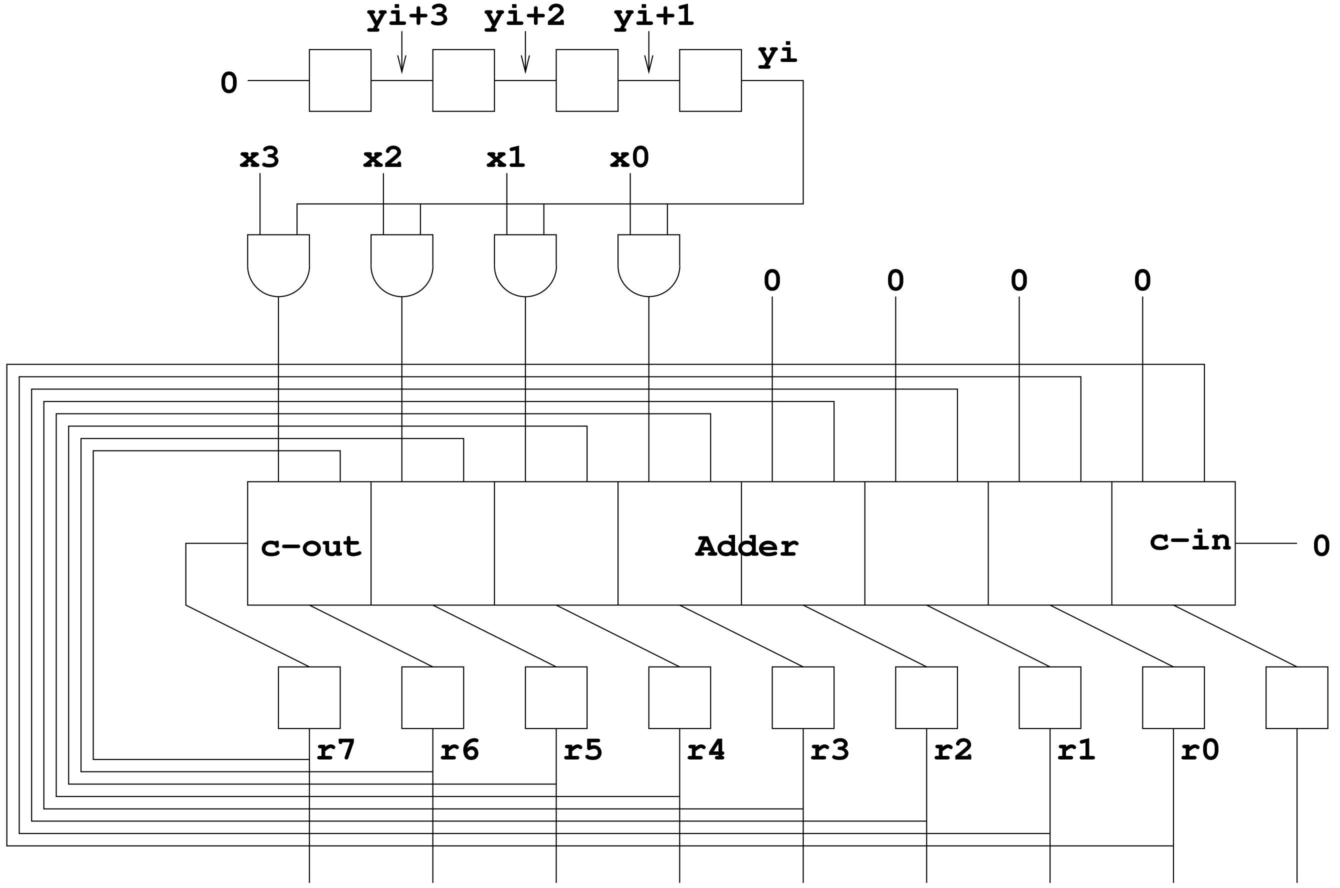
$$x * (y_{i-1} 2^{n-2} + y_{i-2} 2^{n-3} + \ldots + y_0 2^{n-(i+1)}) + x * y_i 2^{n-1} =$$

$$x * (y_i 2^{n-1} + y_{i-1} 2^{n-2} + y_{i-2} 2^{n-3} + \ldots + y_0 2^{n-(i+1)}) = r_{i+1}$$

Multiplication binaire (suite)

Mais: $r_{i+1} = r_i / 2 + x * y_i 2^{n-1} = (r_i + x * y_i 2^n) / 2$

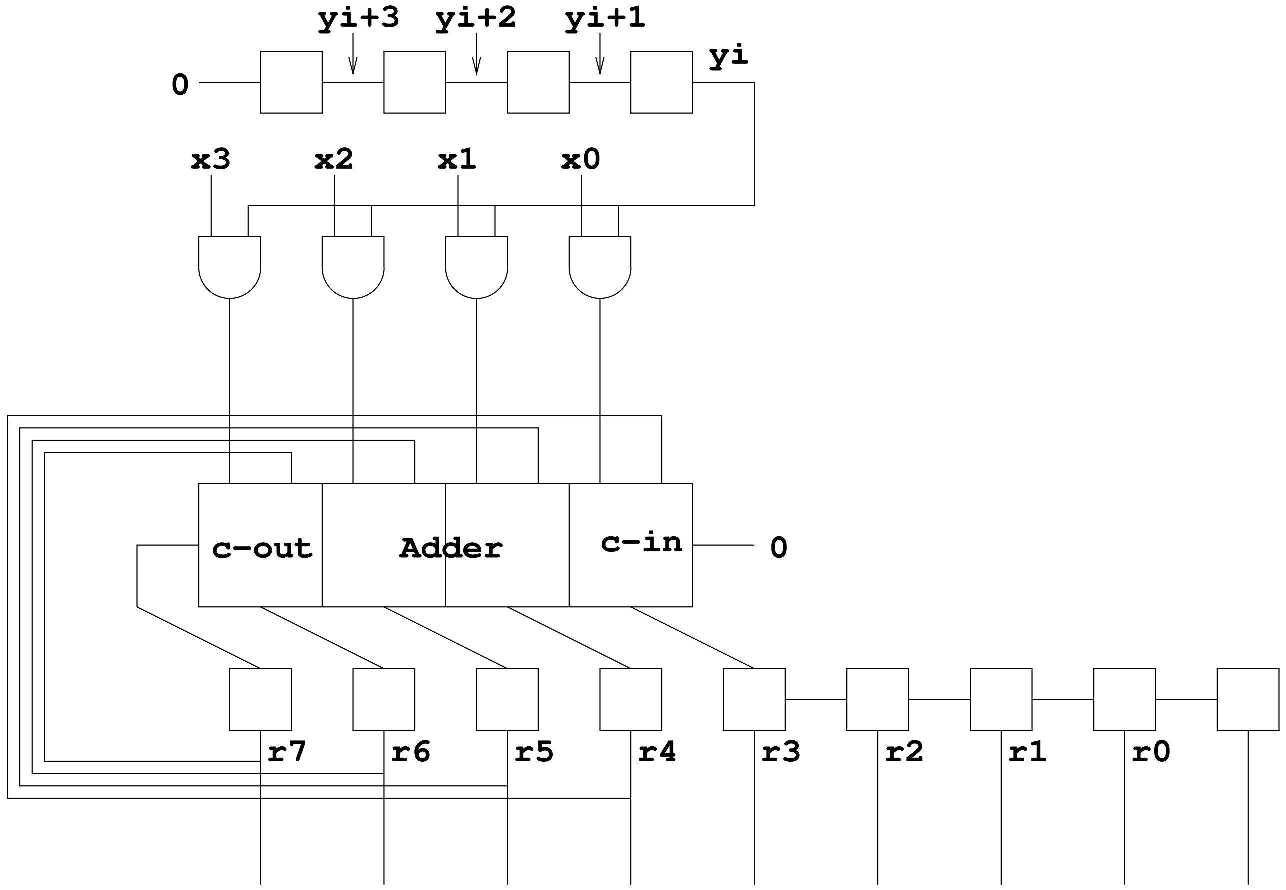
Voici un circuit séquentiel pour effectuer ce calcul:



Multiplication binaire (suite)

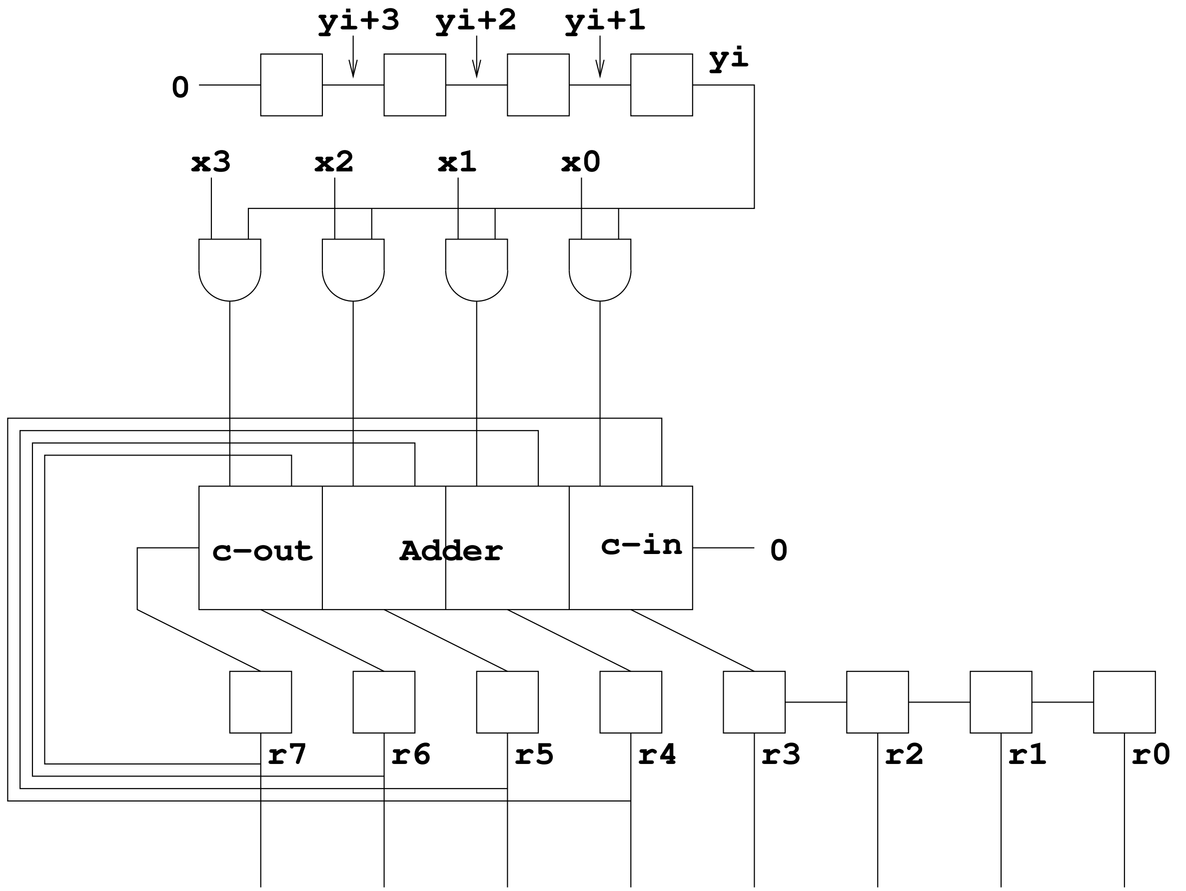
Le circuit précédent marche, mais contient des parties inutiles

Première simplification (éviter d'additioner avec 0):



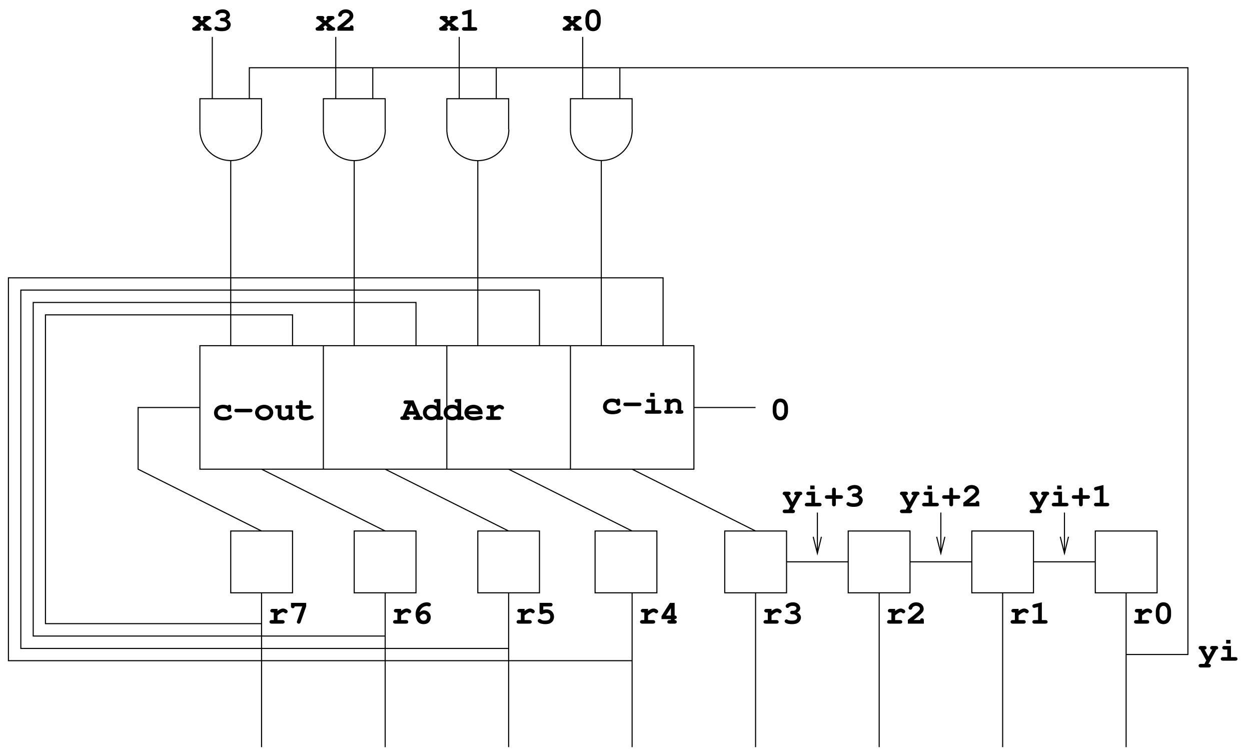
Multiplication binaire (suite)

Deuxième simplification (le dernier bit contient toujours 0):



Multiplication binaire (suite)

Troisième simplification (stocker y dans r):

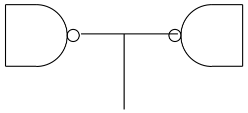


Logique à trois états

Normalement, un circuit peut avoir l'une des deux valeurs 0 ou 1

Avec la logique à trois état, on introduit une troisème possibilité: non définie

Si l'on branche deux sorties de deux circuits différents ensemble, alors on risque de détruire au moins l'un des deux:



Avec des circuits à trois états, on peut le faire, à condition qu'au plus un des circuits branchés ait une valeur définie (0 ou 1)

Les circuits de ce type ont une entrée supplémentaire que l'on appel "enable"

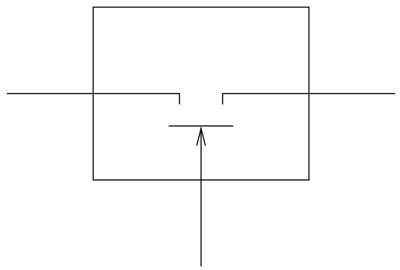
Si cette entrée est 1, alors le circuit se comporte comme un circuit normal

Si cette entrée est 0, alors la sortie du circuit est non définie

Logique à trois états (suite)

Il est toujours possible de convertir un circuit normal en un circuit à trois états avec un circuit que l'on appel "bus driver" ou "pilote de bus".

Voici comment on peut imaginer son fonctionnement:

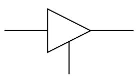


enable

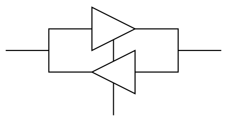
Quand "enable" est 1, alors il y a libre passage entre les deux autres ports

Quand "enable" est 0, alors il n'y a pas de connexion entre les deux autres ports

Symboles:



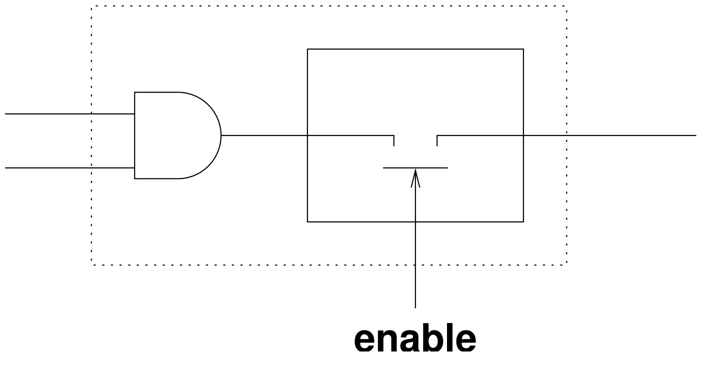
unidirectionnel



bidirectionnel

Logique à trois états (suite)

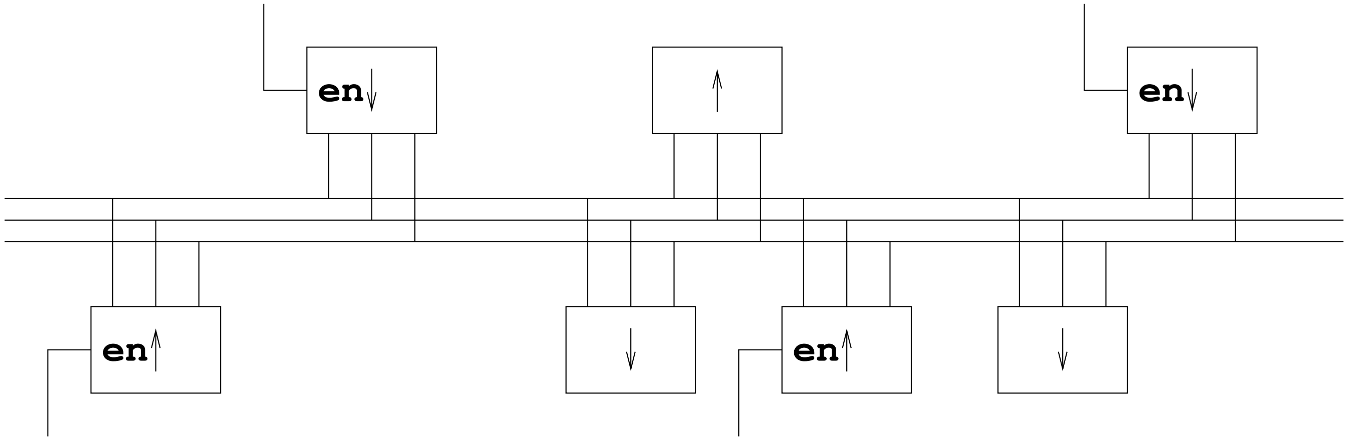
Pour convertir un circuit normal (par exemple une porte "et") en circuit à trois états, il suffit de faire passer la sortie par un "bus driver":



La notion de bus

Un bus est une collections de fils sur lesquels on peut connecter la sortie de plusieurs circuits à trois états, ainsi que l'entrée de circuits arbitraires

Un seul des circuits à trois états peut avoir ses sorties à 0 ou 1. Les autres doivent avoir la valeur non définie



Mémoires

Une mémoire est un circuit similaire à une bascule

**Ce n'est ni un circuit séquentiel (car sans horloge),
ni un circuit combinatoire (car son état dépend du passé)**

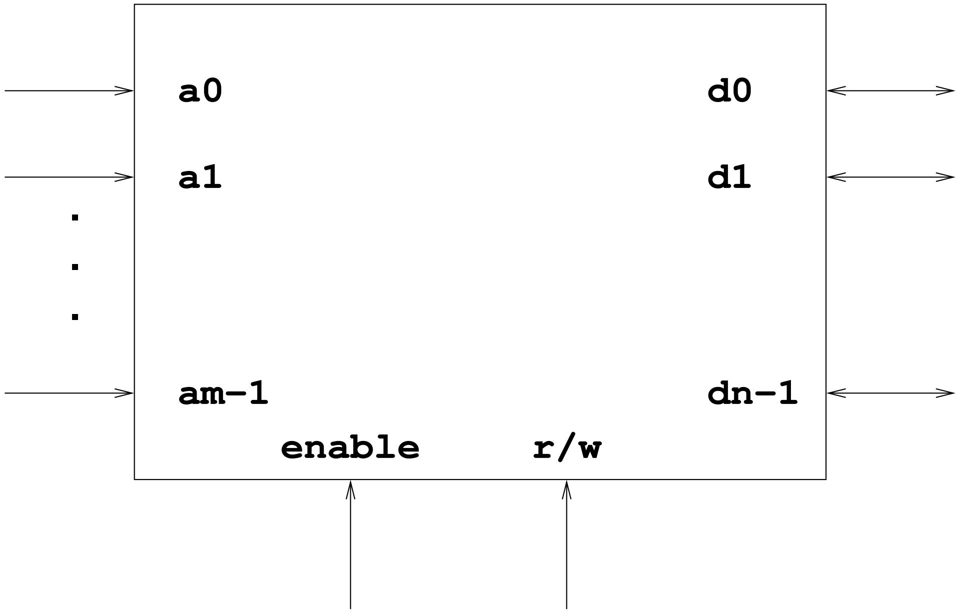
**De plus, pour diminuer le nombre de ports, les entrées et
les sorties sont connectées sur les même fils**

Une mémoire peut stocker un certain nombre de bits d'information

**Les bits sont organisés en un certain nombre de mots de taille fixe
(par exemple 8, 16, 32, 64, 128, ou 256 bits), mais pas forcement
puissance de 2**

Le nombre de mots est souvent une puissance de 2

Mémoires (suite)



Les entrées a0 ... am−1 sont utilisées pour sélectionner un mot

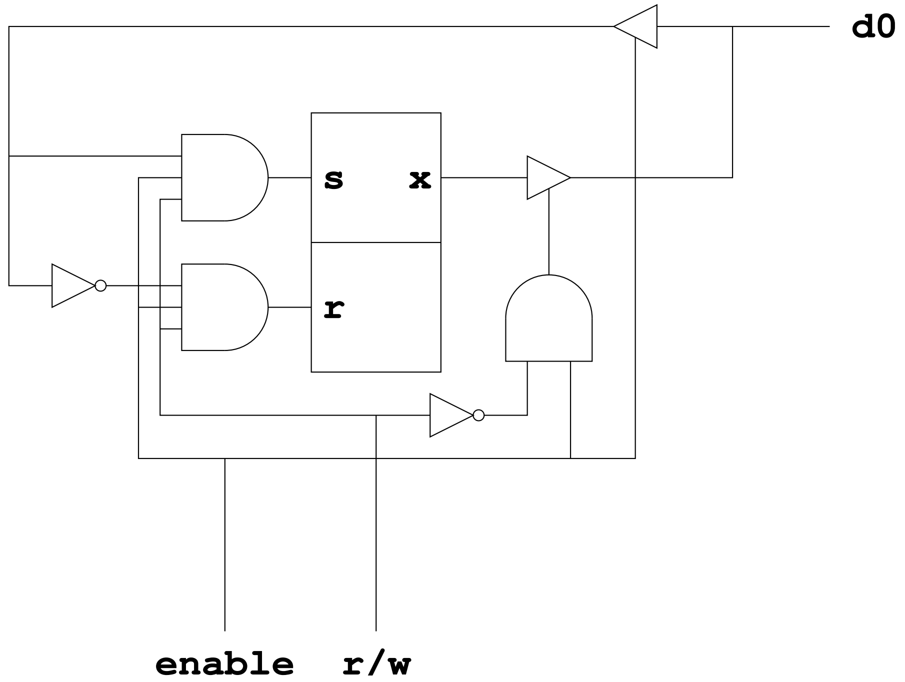
L'entrée enable indique si les sorties d0 ... dn−1 sont dans un état défini

L'entrée r/w (read/write) determine la direction (entrée ou sortie) des ports d0 ... dn−1

Mémoires (suite)

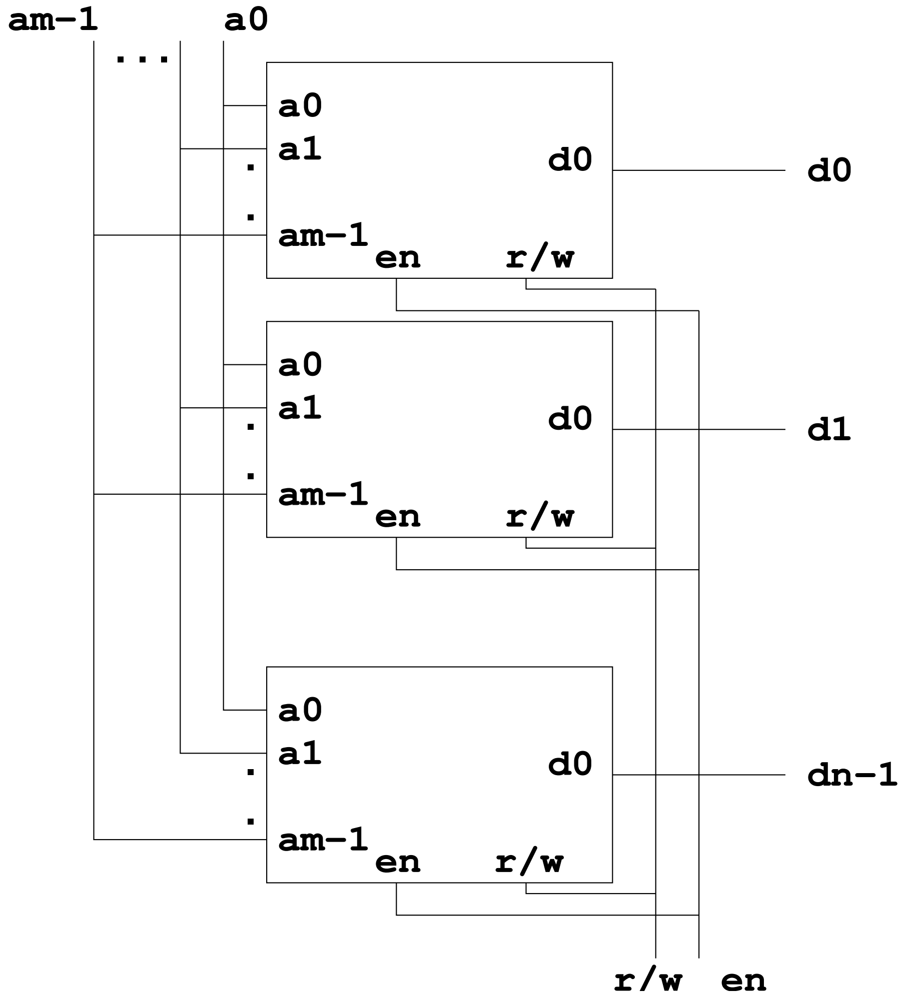
Nous allons montrer comment construire une mémoire à 2^m mots, chacun de n bits

Pour cela on commence avec une mémoire avec m=0 et n=1



Mémoires (suite)

Étant donné n mémoires avec 2^m mots de 1 bit, il est facile de construire une mémore avec 2^m mots de n bits:



Mémoires (suite)

De la même manière, étant donné deux mémoires de 2^m mots de n bits, il est facile de créer une mémoire de 2^(m+1) mots de n bits: