Conception de circuits numériques et architecture des ordinateurs

Frédéric Pétrot

GRENOBLE INP UGA Ensimag

Année universitaire 2022-2023

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

Comment construire des opérateurs complexes à partir des portes élémentaires ? Question à laquelle il faut répondre pour qu'un ordinateur puisse faire des calculs !

Plan détaillé du cours d'aujourd'hui

1. Portes complexes et opérateurs arithmétiques
2. Algorithmique
   - Vrac
   - Itération
   - Récursion
3. Assemblage ad-hoc

Plan

Opérateurs complexes

But :

Construire des opérateurs complexes en assemblant des portes
sans passer par la simplification des équations booléennes
en utilisant la structure des opérateurs

- plus de bits (arité), même fonctions : opérateurs booléens, registres, ...
- opérateurs arithmétiques : addition, décalage, ...
- fonctionnalités nouvelles : multiplexeur, compteur, registres à décalage, ...

Basés sur des approches d'assemblage suivantes :

- algorithmique
  - vrac
  - itératif
  - récursif
- *ad-hoc*

Plan

Opérateur de choix

Multiplexeur (*mux*) : opérateur d'affectation conditionnelle

- i_j, signaux d'entrées
- o, signal de sortie
- s, signal de sélection

```
case S is
   when '0' => O := I0;
   when '1' => O := I1;
end case;
```

```
with S select
   O <= I0 when '0',
        I1 when '1';
```

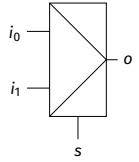
Plus généralement :

$$o = \begin{cases} i_0 & \text{si } s = 0, \\ i_1 & \text{si } s = 1, \\ i_2 & \text{si } s = 2, \\ i_{\ldots} & \text{si } s = \ldots, \\ i_{n-1} & \text{si } s = n-1. \end{cases}$$

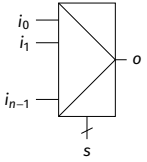
```
with S select
   O <=   I0 when "0...00",
          I1 when "0...01",
          ...
          In-2 when "1...10",
          In-1 when others;
```

Opérateur de choix

multiplexeur 2 vers 1



multiplexeur n vers 1



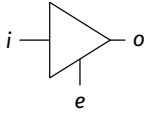
De 2 types :

- décodés : n entrées $\implies$ n signaux de selection
  $\exists! k$ tel que $s_k = 1, 0 \leq k < n$
  $o = i_k$ tel que s_k=1
- encodés : n entrées $\implies$ $\lceil \log_2 n \rceil$ signaux de selection
  $o = i_{\sum_{k=0}^{\lceil \log_2 n \rceil} s_k \times 2^k}$

Opérateur de choix : implantation distribuée

Repose sur des portes dites « trois états »

- binaire : $\{0, 1\}$, valeurs logiques
- trois-états : $\{0, 1, hz\}$, ou hz représente la déconnexion ou *haute impédance* (*high zee*)



e	i	o
0	0	hz
0	1	hz
1	0	0
1	1	1

Comportement d'un interrupteur commandé par e
Connexion des o_i possible ssi $\exists$ au plus un j tel que $e_j = 1$
Si $e = 0$, on peut imposer une valeur sur o de l'extérieur

Opérateurs arithmétiques

$x + y$, $x * y$, $\frac{x}{y}$, $\sqrt{x}$, ln, exp, ...

Différentes implémentations avec différents coûts :

- surface
- vitesse
- puissance

Nécessitent une analyse fine des équations
N'utilisent pas les techniques de minimisation booléennes
Non abordés dans ce cours.
Néanmoins fondamental pour faire une addition sur 32 bits en moins de 1 *ns*

Cf. livre de Jean-Michel Muller (Ensimag'83) qui date mais les principes y sont

`https://hal-ens-lyon.archives-ouvertes.fr/ensl-00086707`

et le cours d'Alain Guyot (enseignant à l'Ensimag jusqu'en 2005)

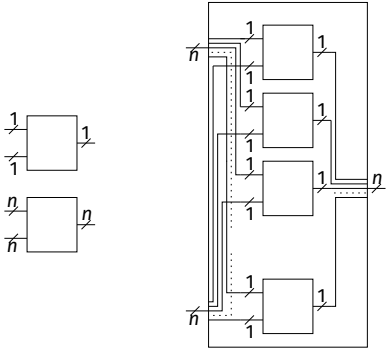
`http://users-tima.imag.fr/cdsi/guyot/Cours/`

Principe de la construction itérative

Principe :

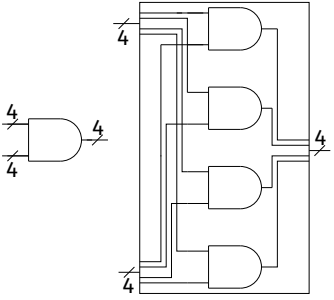
- choisir une porte logique
- faire l'opération sur n bits simultanément en connectant les portes correctement

$$\forall 0 \leq j \leq n-1, o_j = i_{0_j} \text{ op } i_{1_j}$$

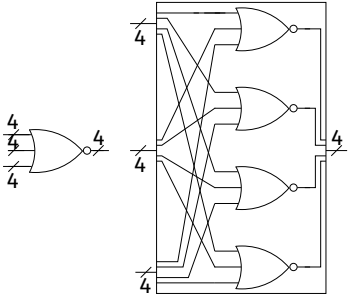


Exemples de construction itérative

and avec 2 entrées de 4 bits



nor avec 3 entrées de 4 bits

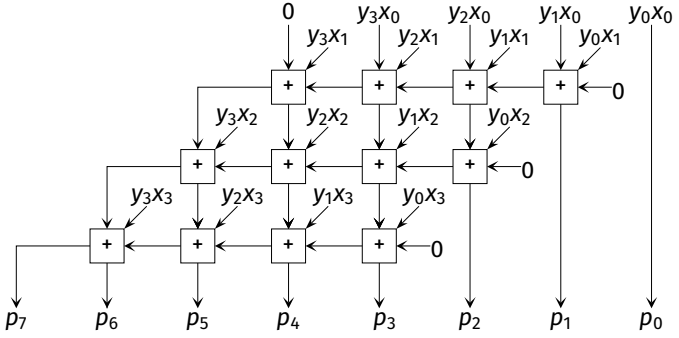


Autres exemples au tableau :

- additionneur n bits à propagation de retenue
- registre n bits (n petit et n grand)

Exemples de construction itérative en 2 dimensions

Étant donnés $y = y_3y_2y_1y_0$ et $x = x_3x_2x_1x_0$, calculer $p = y \times x$
Solution :



Principe de la construction récursive

Principe :

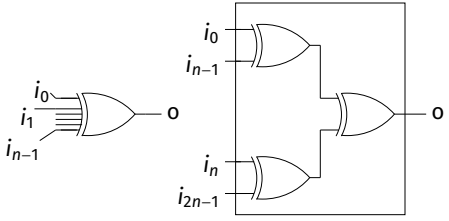
- étant données des portes d'arité n, $n-1$, $n-2$, ...
- construire la porte d'arité $m > n$ avec la même fonctionnalité

Typiquement :

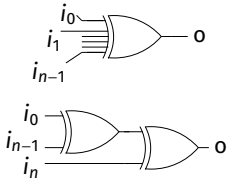
- construire la porte à $2 \times n$ bits des portes à n et 2 bits

Exemples de récursion

porte xor à n entrées ($\bigoplus_{j=0}^{n-1} i_j$) vers $2n$ entrées

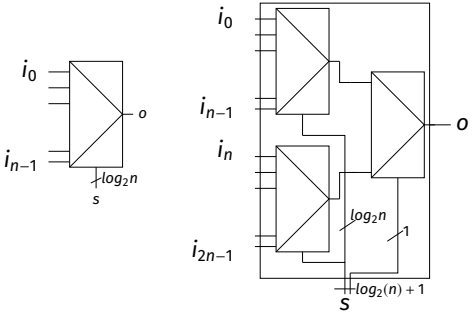


porte xor à n entrées vers $n+1$ entrées



Exemples de récursion

Mux n entrées vers mux $2 \times n$ entrées



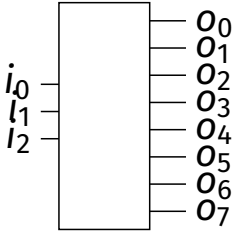
Exemples de récursion

Décodeur : n entrées vers 2^n sorties

$$o_k = \begin{cases} 1 & \text{si } k = \sum_{j=0}^{n-1} i_j \times 2^j, \\ 0 & \text{sinon.} \end{cases}$$

Unique sortie à 1 dont le numéro est encodé par l'entrée

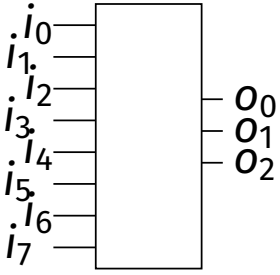
Exercice : décodeur n entrées vers $n+1$ entrées



Exemples de récursion

Encodeur n vers $\lceil \log_2 n \rceil$

Hypothèse :
$\exists! i_j = 1, 0 \leq j < n$
$(o_{\lceil \log_2 n \rceil - 1}, o_{\lceil \log_2 n \rceil - 2}, \ldots, o_0) = k$
for $i_k = 1, i_j = 0 \forall j \neq k$
En d'autres termes : la sortie encode le numéro de l'entrée qui est à 1



Plan

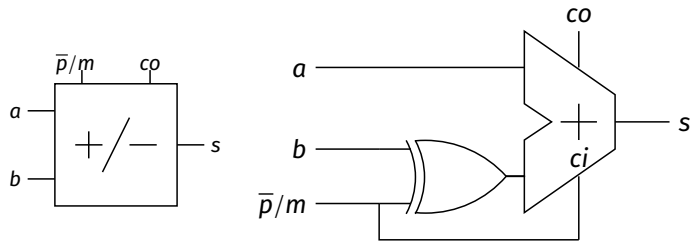
Principe de l'assemblage ad-hoc

Principe :

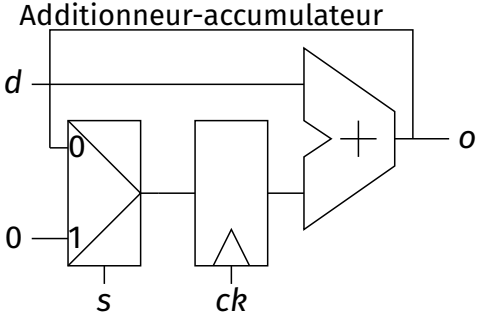
- Etant données des cellules complexes
- construire une cellule implantant une nouvelle fonction en assemblant des portes et des cellules complexes

Exemple :Additionneur/soustracteur n bits

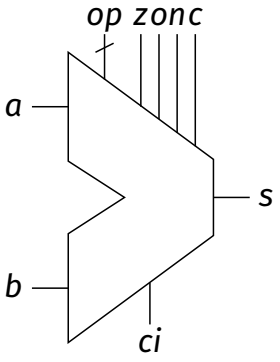
$$s = \begin{cases} a + b & \text{si } \bar{p}/m = 0, \\ a + \bar{b} + 1 = a - b & \text{si } \bar{p}/m = 1. \end{cases}$$



Autres exemples



Unité arithmétique et logique (*alu*)



Autres exemples d'assemblage

Exercices :

- banc de registres de 4 mots de 2 bits chacun
- registre à décalage
- compteur/décompteur