CHAPITRE 4
CIRCUITS PROGRAMMABLES POUR LES DESIGNS COMBINATOIRES

- Jusqu'à présent, notre effort s'est porté sur la minimisation des designs et le recours aux portes simples. 74LSXX surtout NAND.

- Nous allons maintenant explorer d'autres circuits intégrés plus complexes.

  - Multiplexeurs MUX (circuits aiguilleurs 2^N entrées – 1 sortie peut-être de plus de 1 bit de long
  - DÉMUX (décodeurs)
  - ROM (sélection d'un mot complet)

ROM et MUX peuvent réaliser n'importe quelle table de vérité à N entrées souvent une simple reprogrammation suffit pour adapter le design à une nouvelle spécification.

On étudiera aussi les "programmable logic devices" PLD qui sont très efficaces pour réaliser des circuits SOPs.

Ex : PALs, FPGA

Qui nécessitent des outils CADS

4.1 Synthèse avec MSI

MSI = Medium Scale Integration

- multiplexeurs : 74150, 74151, 74153, 74153, 74157…
- décodeurs : 7442, 74138, 74154, 74, 155
- encodeurs de priorité : 74148
- générateurs de parité : 74180
- additionneurs : 7483
- multiplicateurs : 74284

- comparateurs : 7485
- compteurs, registres, 74374, 74373
- ACU : 74181

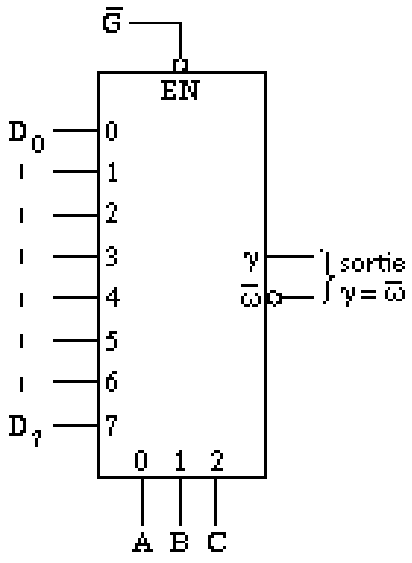
Avantages : Une fonction complète est définie dans un circuit intégré (c.i.) ou une librairie
⇒ Moins de câblage, moins d'espace occupé.

La méthode de synthèse par SSI vue précédemment est une méthode de dernier recours mais utile pour la "glue logic".

4.2 Synthèse avec multiplexeur

Circuit combinatoire, "aiguillage" à 2^n entrées d'information, n entrées d'adresse et une seule sortie.
Sortie = entrée de rang égal à l'adresse présente

Ex. MUX 74151 (8 à 1)



G = bit d'inhibition (ENABLE)

Si G = 1 ⇒ Y = 0
G = 0 ⇒ mode normal

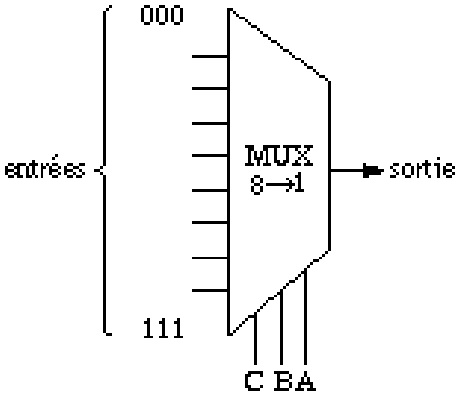


Table de vérité (tdv)

C	B	A	G	Y
X	X	X	1	0
0	0	0	0	D_0
0	0	1	0	D_1
0	1	0	0	D_2
0	1	1	0	D_3
1	0	0	0	D_4
1	0	1	0	D_5
1	1	0	0	D_6
1	1	1	0	D_7

Expression générale :

$Y=m_0D_0+m_1D_1+\ldots+m_{2n-1}D_{2n-1}$ m_j = adresse

D_j = entrée

La famille TT offre plusieurs modèles de MUX :

74150 – 16 à 1
74151 – 8 à 1
74153 – 2MUX 4 à 7
74157 – 4MUX 2 à 1

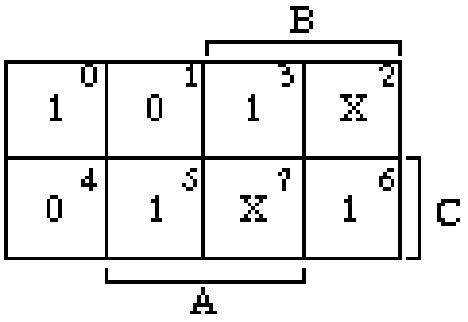
Applications des MUX :

- Sélection d'un bit
- Conversion parallèle – série
- Génération de fonctions combinatoires (l'application qui nous intéresse ici)

Exemple : Réaliser la table de vérité suivante avec et sans MUX

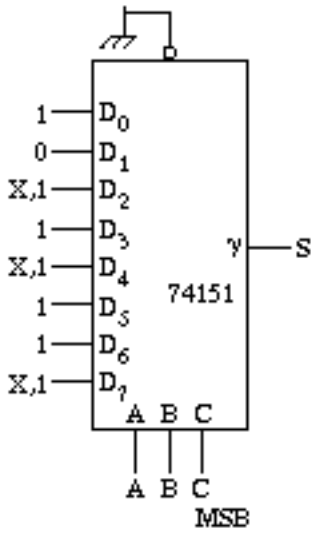
m(0,3,5,6)+X(2,4,7)

	C	B	A	S
0-	0	0	0	1
1-	0	0	1	0
2-	0	1	0	X
3-	0	1	1	1
4-	1	0	0	X
5-	1	0	1	1
6-	1	1	0	1
7	1	1	1	X

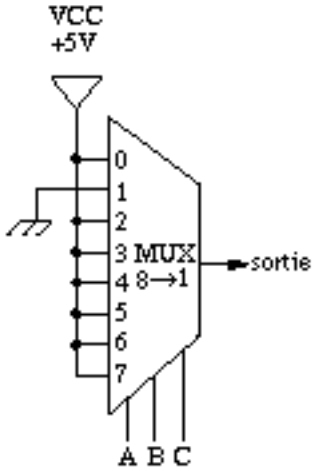


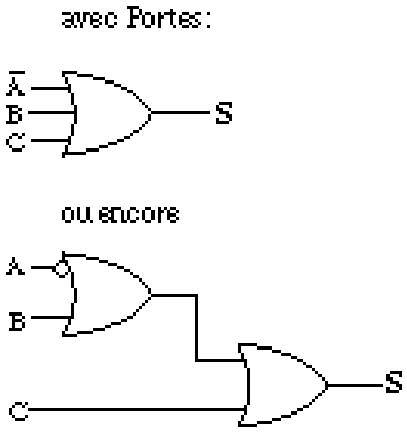
Solution SOP : S = B+ $\overline{A}$ +C

<u>Avec MUX</u> A,B,C servent d'adresse, MUX 8 à 1



Donc :





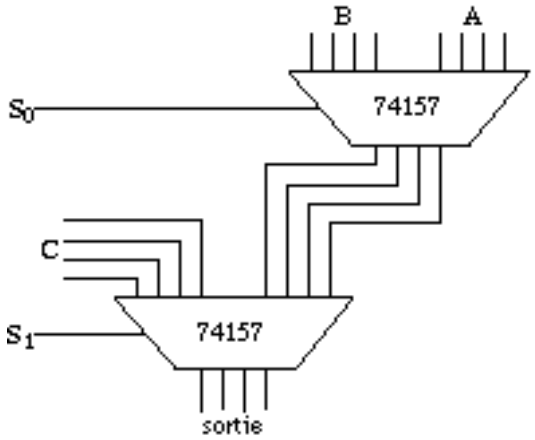
Coût 1

Coût $\frac{1}{2}+\frac{1}{6}$

<u>Cascades de MUX</u>

On peut cascader plusieurs MUX ensemble.

Ex : Réaliser un MUX 3 à 1 avec un mot de 4 bits de large. On prend 2 MUX 74LS157 (2 à 1, 4 bits de large) qui sont combinés comme suit :



S1	S0	Sortie
0	0	A
0	1	B
1	0	C
1	1	C

À cause de l'ordre de la cascade

4.3 MUX et théorème d'expansion de Shannon

Le théorème de shannon stipule que n'importe quelle expression booléenne F(X_1, X_2, …X_N) de N variables peut-être étendue.

$$F(X_1,\dots X_N) = X_1 \cdot F(1,X_2,\dots X_N) + \overline{X}_1 \cdot F(0,X_2,\dots X_N)$$

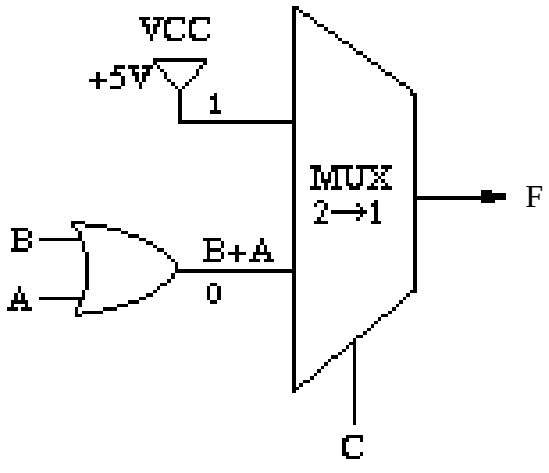
Exemple : si $F = [C = B = A]$

On peut dire que $F = C \cdot \left[\frac{1+B+A}{1}\right] + \overline{C}\,[0+B+A]$

$$F = C \cdot 1 + \overline{C} \cdot [B+A]$$

	C	B	A	A+B+C	B+A	$\overline{C}\,[B+A]$	$C+\overline{C}[B+A]$
1	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	1
0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1
0	1	1	0	1	1	0	1
0	1	1	1	1	1	0	1

idem



Le théorème de Shannon peut-être réappliqué une 2ième fois.

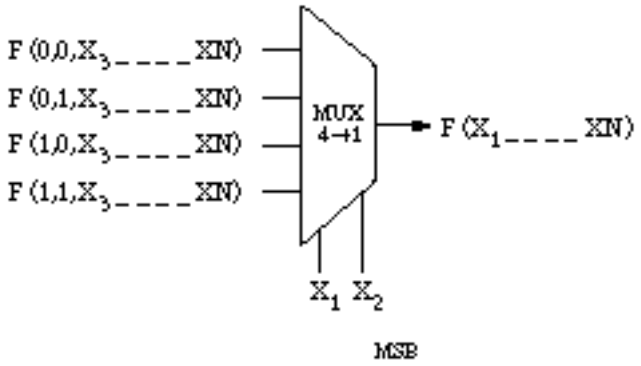
$$F(X_1 \dots X_N) = X_1 \cdot F(1, X_2 \dots X_N) + \overline{X} \cdot F(0, X_2 \dots Y_N)$$

$$= X_1 X_2 F(1,1, X_3 \dots X_N) + X_1 \overline{X}_2 \cdot F(1,0, X_3 \dots X_N) +$$

$$+ \overline{X}_1 X_2 \cdot F(0,1, X_3 \dots Y_N) +$$

$$+ \overline{X}_1 \overline{X}_2 \cdot F(0,0, X_3 \dots Y_N)$$

Ce qui se prête bien à la réalisation par MUX 4→1 :

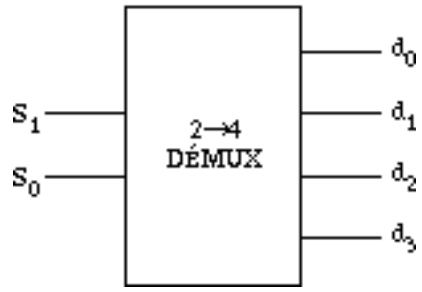


4.4 Démultiplexeur ou décodeur

Définition : circuit qui "décode" le poids logique d'une entrée. On a N entrées et 2^N sorties, aussi appelé "décodeur d'adresse".

Ex :

Nentrées 2^N sorties



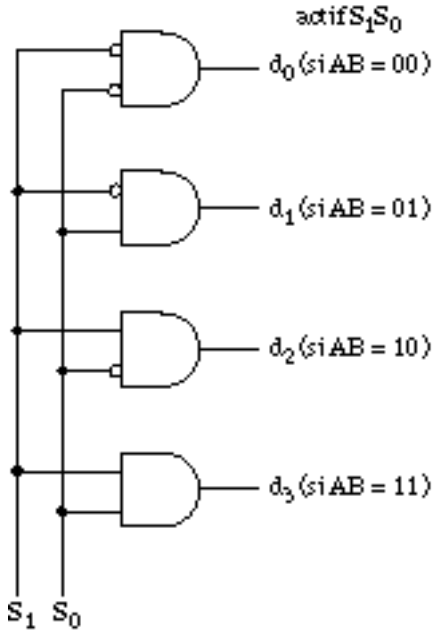
On active la ligne d_i qui correspond au poids binaire des entrées S_1, S_0

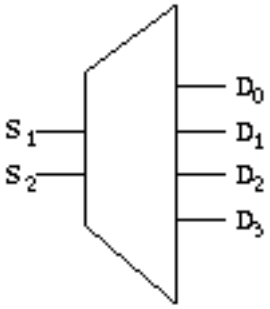
⇒ 2 entrées et 4 sorties

(N) (2^N)

Réalisation :

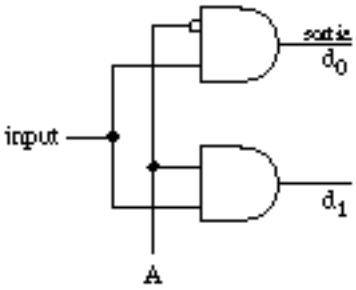
Si S_1S_0=11 $\Rightarrow d_3$ Actif
Si S_1S_0=10 $\Rightarrow d_2$ Actif
Si S_1S_0=01 $\Rightarrow d_1$ Actif
Si S_1S_0=00 $\Rightarrow d_0$ Actif





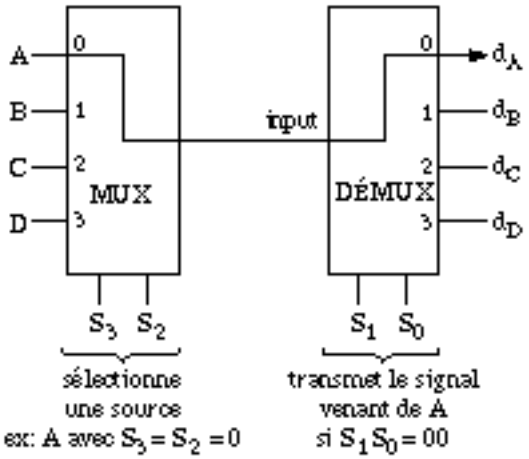
SYMBOLE

Il y a aussi des DEMUX avec une entrée qui est une ligne commune à toutes les entrées.



Si input=0 $\Rightarrow$ $\forall$sorties=0
Si input=1 $\Rightarrow$ d_0=1 si A=0
d_1=1 si A=1

On peut alors réaliser un "Routeur" en connectant ensemble MUX et DEMUX :

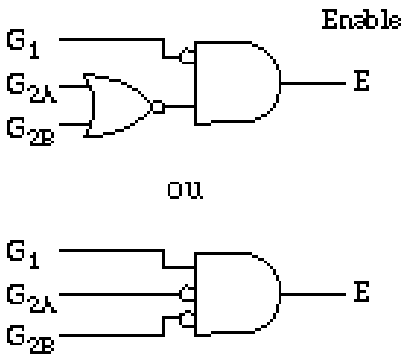


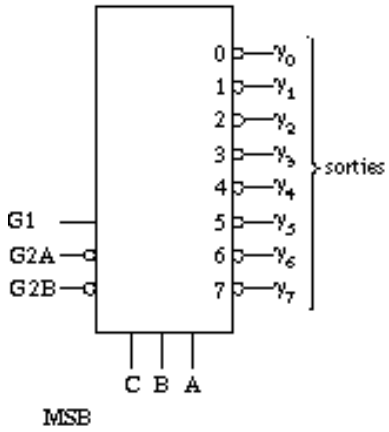
Transmet un des 4 signaux sur n'importe laquelle sortie d_A, d_B d_C, d_D

4.5 Synthèse par décodeurs (DMUX) ou démultiplexeurs

- Circuit combinatoire à N entrées et 2^N sorties, seule une sortie est activée à la fois.
- Le rang de la sortie activée correspond à la valeur binaire de l'adresse présente.

Ex : décodeurs DMUX (74138 3 à 8)





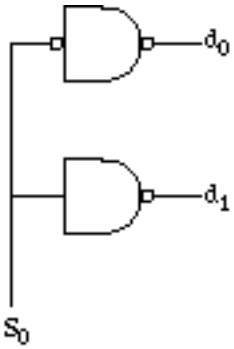
$si\left[G_1 \cdot \overline{G2A} \cdot \overline{G2B}\right] = 1$
$\Rightarrow$ mode normal
sinon $Y_i = 1$

$G_1 \cdot (G_{2A} + G_{2B}) = 1$

G_1	(G2A+G2B)	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

Extrait du Fast TTL Logic Series, Philips Semiconductors, 1992, p. 168

En c.i., les DEMUX sont généralement "Actif low" avec des "NAND" gates



S_0	d_0	d_1
0	0	1
1	1	0

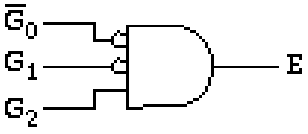
Actif

Chips communs :	74139	2X DEMUX	$2 \rightarrow 4$ (16 PINS)
	74138	1X DEMUX	$3 \rightarrow 8$ (16 PINS)
	74154	1X DEMUX	$4 \rightarrow 16$ (24 PINS)

Ces chips ont un "ENABLE" qui permet de désactiver les sorties : si E = High $\Rightarrow$ toutes les sorties à High.

Le ENABLE est parfois composé de plusieurs entrées :

Ex : 74138



$E = \overline{G}_0 \overline{G}_1 G_2$

, si E = L $\Rightarrow \forall$ sorties à H

Actif si E = H

i.e si $\overline{G}_0 = 0, \overline{G}_1 = 0$ *et* $G_2 = 1 \Rightarrow E = H$

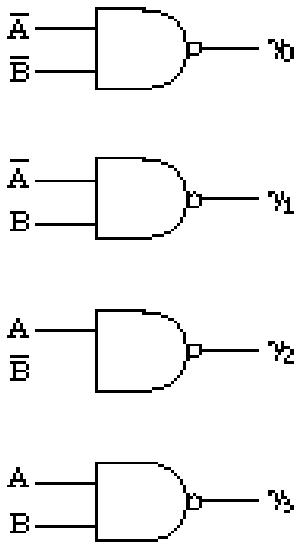
nb	ET
00	0
01	0
10	0
11	1

Les entrées "ENABLE" permettent de cascader les devices

Décodeur 2 à 4 désignes : - 2 lignes entrées
- 4 lignes de sortie

2 à 4 DE-MUX : 1 seule des 4 sorties est activée à la fois

Réalisation :

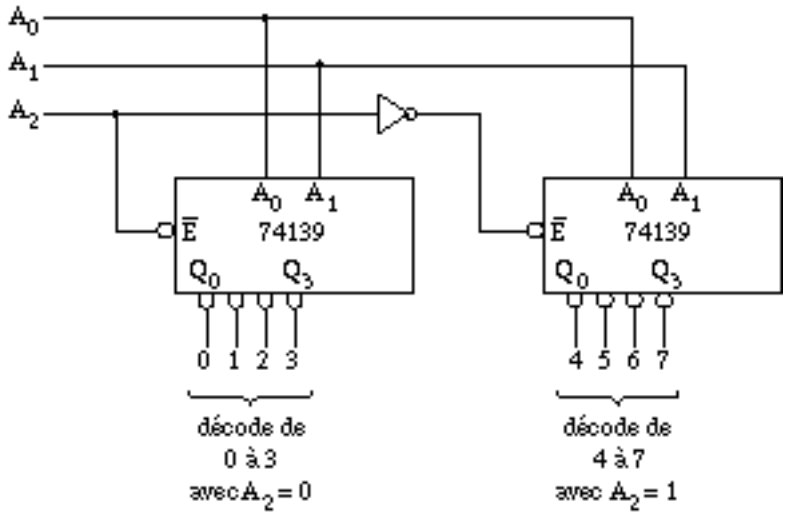


Note : Parfois le décodage n'est pas binaire → décimal

Ex : 7442
BCD → décimal

Famille	7442	-	BCD →	décimal
	7443	-	Excess 3 →	décimal
	74154	-	DEMUX	4 à 16
	74138	-	DEMUX	3 à 8
	74155	-	2 DEMUX	2 à 4

Par exemple pour réaliser un DEMUX 3 à 8 avec 2 DEMUX 2 à 4
DEMUX $\Rightarrow$ 2 à 4 en cascade permet d'avoir un décodeur 3 à 8



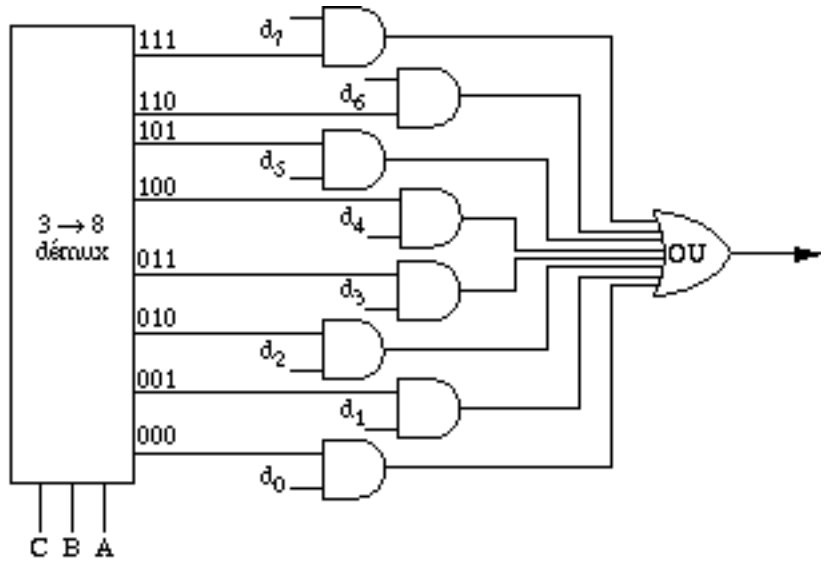
A_2	A_1	A_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

À faire à la maison décodeur 5 à 32
Solution employer 4 X 74138 + 1 X 74139
74138 = décodeur 7 à 8

Décodeur et Démux sont des circuits très utiles pour l'interface aux microprocesseurs, microcontrôleurs, mémoires, registres…

4.6 Mémoire "Read Only"(lecture seulement) ROM

On peut voir la structure d'un ROM à partir d'un démultiplexeur. Exemple : mémoire à 8 bits



Sortie

Un ROM ou une mémoire c'est en fait un circuit SOP à contenu programmable.

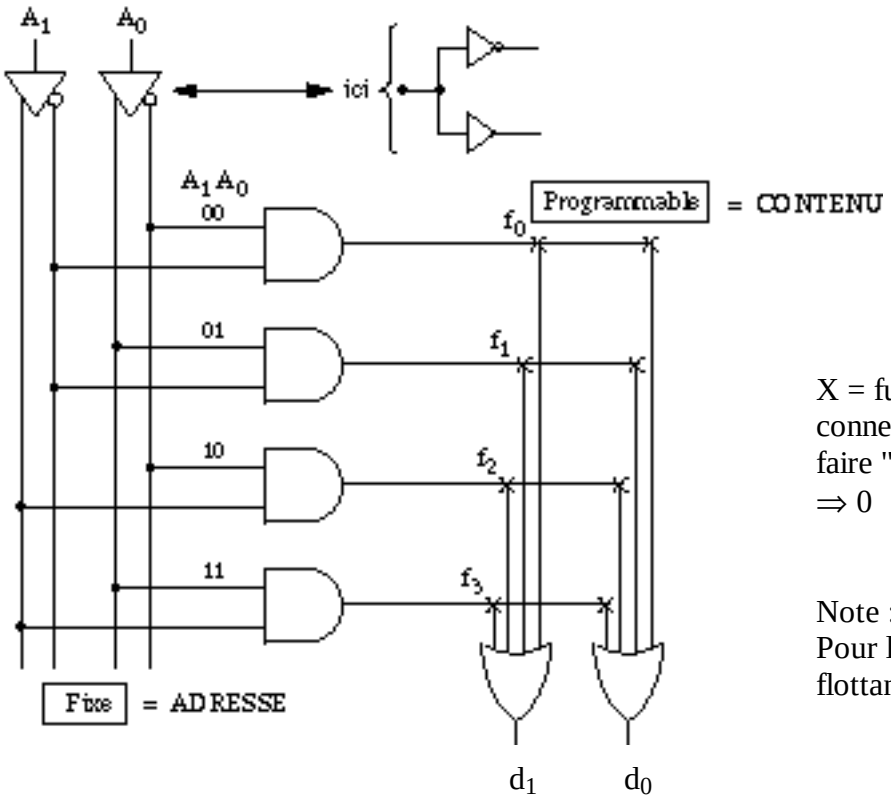
$$sortie = d_7 CBA + d_6 CB\overline{A} + ... + d_0 \overline{C}\,\overline{B}\,\overline{A}$$

Évidemment ce schéma n'est pas pratique pour de grandes mémoires, plutôt que d'employer des portes ET pour les bits de data, on emploie des "interrupteurs internes" qui peuvent être mis à 1 ou à 0 selon le contenu de la mémoire.

- Écrit une fois pas effaçable ROM
- Écrit et effaçable UV EPROM
- Écrit et effaçable électrique EE PROM
- Écrivable-programmable, pas effaçable PROM

Définition de mémoire : Unité qui emmagasine des valeurs ordonnées selon leurs adresses respectives.

En réalité, on emploie des "Fusibles" internes et on a le schéma suivant :

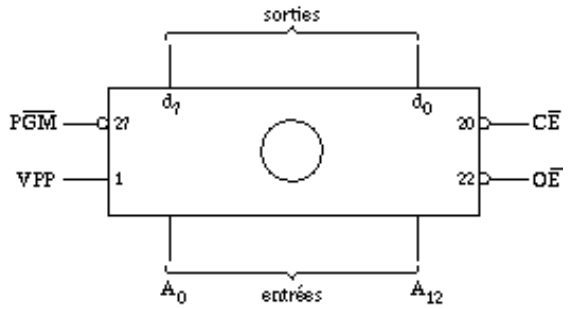


X = fusible interne ou départ tout est connecté (fusible intact) qu'on peut faire "brûler" sélectivement si brûlé $\Rightarrow 0$

Note : entrée flottante est à Ø.
Pour les TTL c'est l'inverse entrée flottante = 1

$$d_0 = \overline{A}_0 \overline{A}_1 f_0 + \overline{A}_0 A_1 f_1 + A_0 \overline{A}_1 f_2 + A_0 A_1 f_3$$

Exemple chip 2764 stocke à 28 PINS
2^{16} = 65536 = 64k bits
avec 13 lignes d'adresse (2^{13} = 8192), donc
8192 mots de 8 bits (65536 = 8 X 8192)



Mode lecture $O\overline{E}$ = $C\overline{E}$ = 0
Pin 27 $PG\overline{M} = 1$ *et* $VPP = 1$

$\overline{OE}$ et $\overline{CE}$: utiles pour sélectionner la mémoire lorsque reliée à un BUS commun ou à plusieurs devices. (cf chap.8)

Le 2764 peut-être effacé par de la lumière UV (très énergétique), le device au complet retourne alors à l'état 11111 11111 (F F).

Pour écrire, il faut VPP + 25 volts et lorsque $\overline{PGM}$ est "pulsed low" pour 50ms, les valeurs de d_i sont écrites ("brûlées") à l'adresse présente en A_i (ceci se fait avec un programmeur d'EPROM. Ça prend ≈ 3 à 4 minutes à programmer.

L'usage de ROMs est très répandu :

- PC - pour le, BIOS (Basic Input Output System) généralement 32 K x 8 bits permet le lancement du OS (système d'opération)
- Contrôle des polices de caractère dans les imprimantes
- Programmes pour les micro-contrôleurs (processeurs autonomes et complets)

Ex : – Four à micro-onde
– vidéo
– lecteur CD

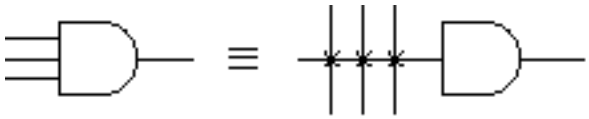
4.7 PLD : Programmable Logic Devices

- Un ROM est un type de PLD
- Programmable signifie que l'usager peut le configurer avec une installation modeste (ex : programmeur d'EPROM)
- PLD : généralement circuits de type SOP configurable à la demande

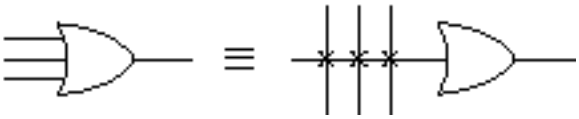
Il y a d'autres types de PLA ex : PAL

4.8 PAL – Programed Array Logic

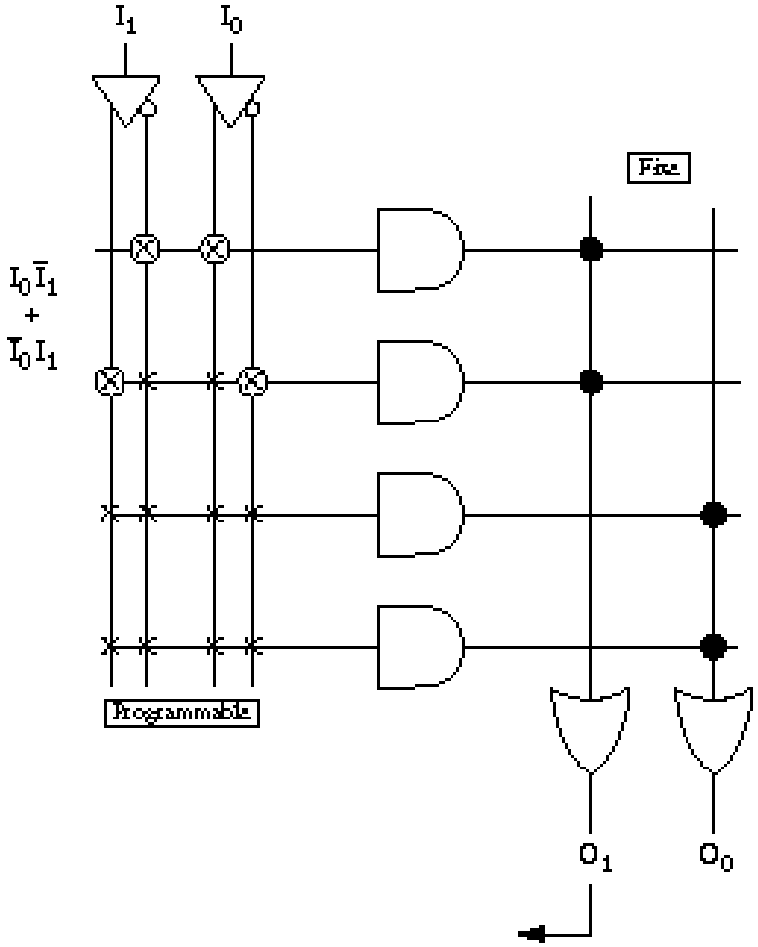
- Marque de commerce de Monolithique Memories (MM)
- Ici, le réseau ET est programmable, le réseau OU est fixe. Utile, plus efficace (plus versatile car au départ tous les produits contiennent <u>toutes</u> les variables et sont identiques).
- convention



et pour le ou



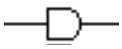
- Schéma d'un PAL



Exemple : pour implanter le OU – exclusif

Convention

si ∀ fusibles intacts ⇒ produit = 0

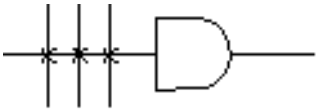


si ∀ fusibles brûlés ⇒ produit = 1

$O_1 = I_0 \bar{I}_0 I_1 \bar{I}_1 + I_0 \bar{I}_0 I_1 \bar{I}_1$

Au départ, comme un ROM,
∀ les fusibles à 1 du côté ET

Si tous les fusibles sont intacts :



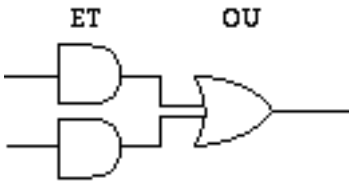
Et après configuration (ou exclusif) :

$$O_1 = I_0 \bar{I}_1 + \bar{I}_0 I_1$$

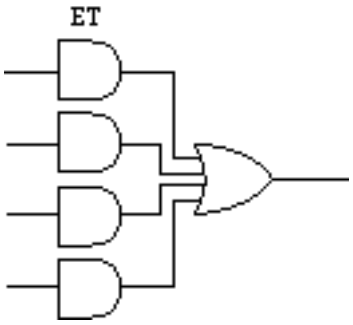
Limitation des PALs :

- PAL est de type OTP (One Time Programming)
  1 erreur et le chip est inutilisable, il existe cependant des versions effaçables UV.
- Limite du nombre de portes ET par portes OU ce qui va limiter le type de fonctions qu'on peut y incorporer.

Exemple :



Ici somme de 2 termes



Ici somme de 4 termes

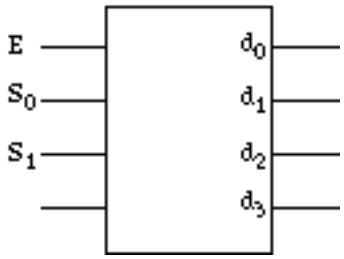
Avantages

- Délais de propagation plus petit que dans les ROM
- Prend moins de place qu'un ROM / TTL (car cellule + petites)
  Ex : 20PIN 16L8
  Équivalent de 72 ET @ 8 entrées
  et OU
- Coûte moins cher
- Peut-être programmé en plusieurs étapes

Note : Les PALs se programment généralement à l'aide de langages spécialisés.

Ex. : ABEL, VHLD, Verilog
Vus dans d'autres cours

Réaliser le DÉMUX suivant avec un PAL



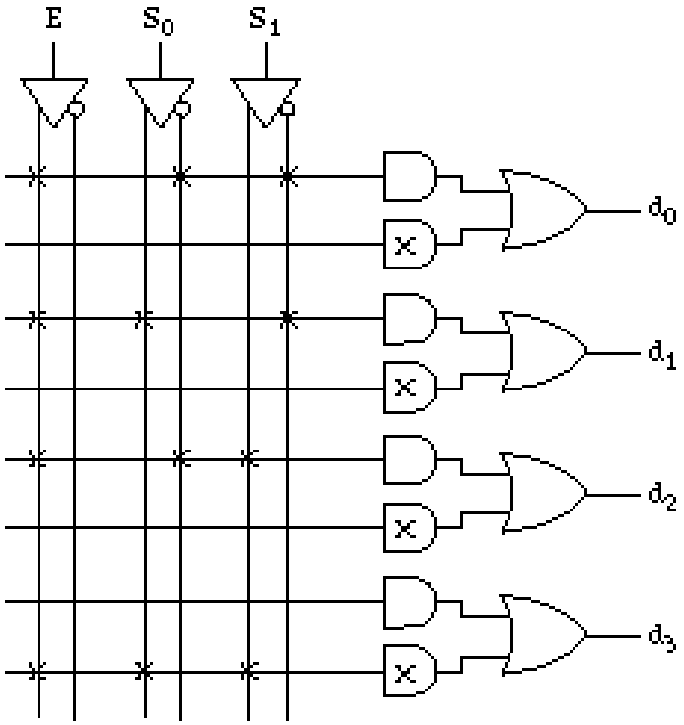
On a de suite que :

$$d_0 = \overline{S}_0 \overline{S}_1 E$$
$$d_1 = S_0 \overline{S}_1 E$$
$$d_2 = \overline{S}_0 S_1 E$$
$$d_3 = S_0 S_1 E$$

E	S_1	S_0	d_0	d_1	d_2	d_3
L	X	X	L	L	L	L
H	0	0	H	L	L	L
H	0	1	L	H	L	L
H	1	0	L	L	H	L
H	1	1	L	L	L	H

Exemple :

Intact $\Rightarrow 0$
$S_0 \overline{S}_0 \overline{E} \overline{\overline{E}} \overline{S}_1 \overline{S}_1 = 0$



On peut évidemment employer les notions de table de Karnaugh pour simplifier la tdv avant de l'implanter dans le PAL. Cela parfois essentiel sinon ça ne "rentre pas" (pas assez de produits par somme).

Exemple : Réaliser un PAL qui indique si 2 mots de 3 bits sont différents.

$B_2B_1B_0 \neq A_2A_1A_0$, A=B

Q1 – Combien de termes de produits nécessaires ?
Q2 – Tracer circuit PAL ?

Analysons les cas où il y a identité ie [A=B]

Un mot de 3 bits $\Rightarrow 2^3 = 8$ possibilités donc il y aura 8 cas sur les $2^6 = 64$ possibles où [A=B] et donc 64-8 = 56 (cas où A≠B).

Note : On aurait pu faire la tdv, mais long car 6 variables $\Rightarrow 2^6 = 64$ lignes.

Rép. Donc il y aura 56 cas, 56 produits pour lesquels A≠B

Il semble plus facile de synthétiser :

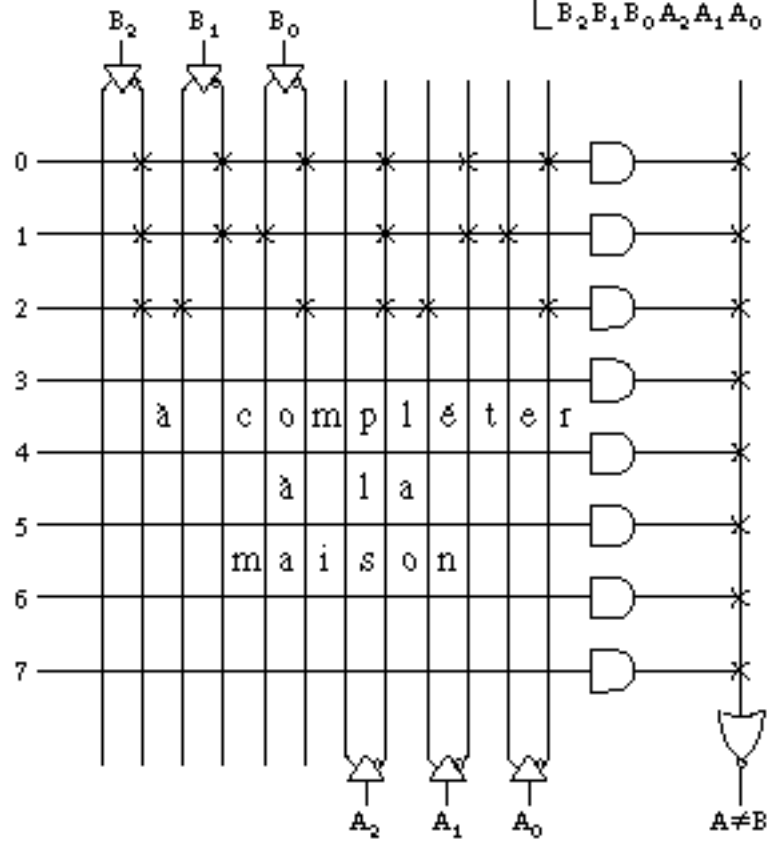
(8 termes au lieu de 56 termes)

$$S_{[A \neq B]} = S_{[\overline{A=B}]}$$

les 8 cas sont : $\begin{bmatrix} 000 & --- & 001 & --- & 111 \\ 000 & & 001 & & 111 \end{bmatrix}$

on peut tracer le PAL directement

$\begin{bmatrix} \bar{B}_2\bar{B}_1\bar{B}_0\bar{A}_2\bar{A}_1\bar{A}_0 \\ \text{à} \\ B_2B_1B_0A_2A_1A_0 \end{bmatrix}$



C'est un PAL complémenté

4.9 Spécification des PALs

Il y en a de plusieurs types.

La numérotation est généralement la suivante :

EE T SS

EE : Nb d'entrées du réseau

T : TYPE

SS : Nb de sorties du réseau

Ex : 12 H 6

12 entrées, 6 sorties

4.10 Retournement, Partition (Folding)

Parfois les sorties d'un PAL peuvent être employées comme entrées d'un second PAL pour accommoder des fonctions plus complexes.

Par exemple :

- Pour cascader différents circuits dans le PAL
- Scinder une expression Booléenne qui a trop d'entrées pour 1 PAL

Exemple : Soit un PAL 5 entrées, 4 sorties

Implanter simultanément avec 2 PALS la fonction

a) $P = \overline{E}F\left[\overline{D} + \overline{C}\right] + B$

b) $Q = \underbrace{\overline{D} + C} + \overline{E} \cdot F \cdot B$

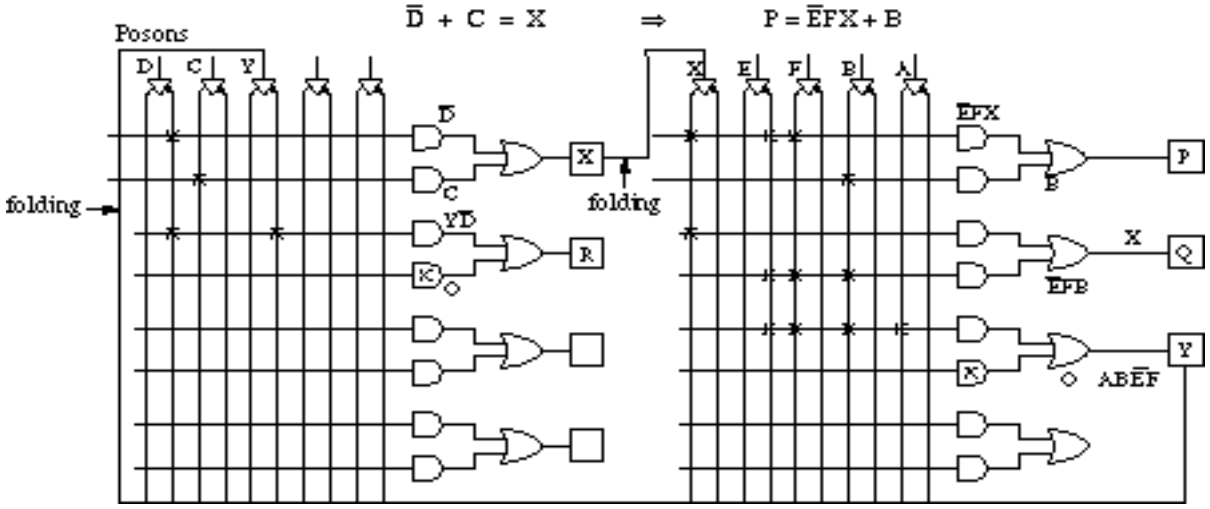
$Q = X + \overline{E}FB$

c) $R = AB\,\overline{D}\overline{E}F = A\overline{D}B\overline{E}F$

Il manque une entrée ! $\Rightarrow$ on doit réécrire l'expression ("D" est dans le PAL #1)

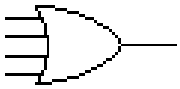
$R = (\underbrace{AB\overline{E}F}_{\text{dans PAL 2}})\,\overline{D} = Y\overline{D}$ ↖Ramené au PAL 1 par Folding

Note : On ne peut l'implanter directement car il y a deux sommes en cascade. Il faut aussi employer 2 PALS car on a pas assez d'entrées car on a 6 variables D,C,B,E,F et X

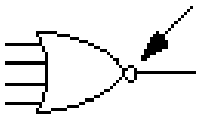


Types de PAL disponibles

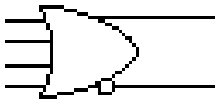
H= Sorties active haut H



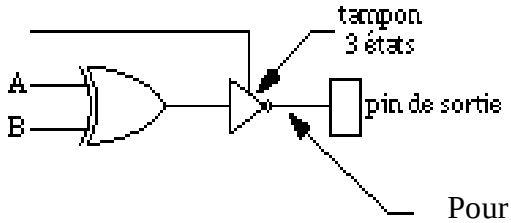
L = Sorties active Bas L



C= Sorties complémentées H et L



X= Sorties avec porte XOR



Pour connexion de devices multiples entre eux (exemple sur les bus de données).

R= Avec registre interne (voir plus tard)

A= Arithmétique (voir plus tard)

4.11 Programmable Logic ARRAY

ROM - SOP, somme programmable

PAL - SOP, produit programmable

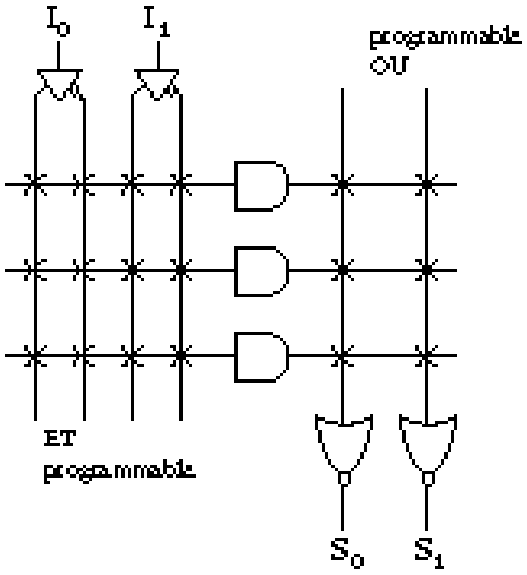
PLA - SOP, produit et somme programmable

Program Logic ARRAY

Plus coûteux que les PALs
(les 2 réseaux sont programmables)
– Plus lent
– Pas évident à comprendre par rapport aux PALs, ROM.

⇒Fabricant : Sigenetics et autres

→ Contient aussi souvent d'autres blocs e.g. : bascules (voir plus tard)



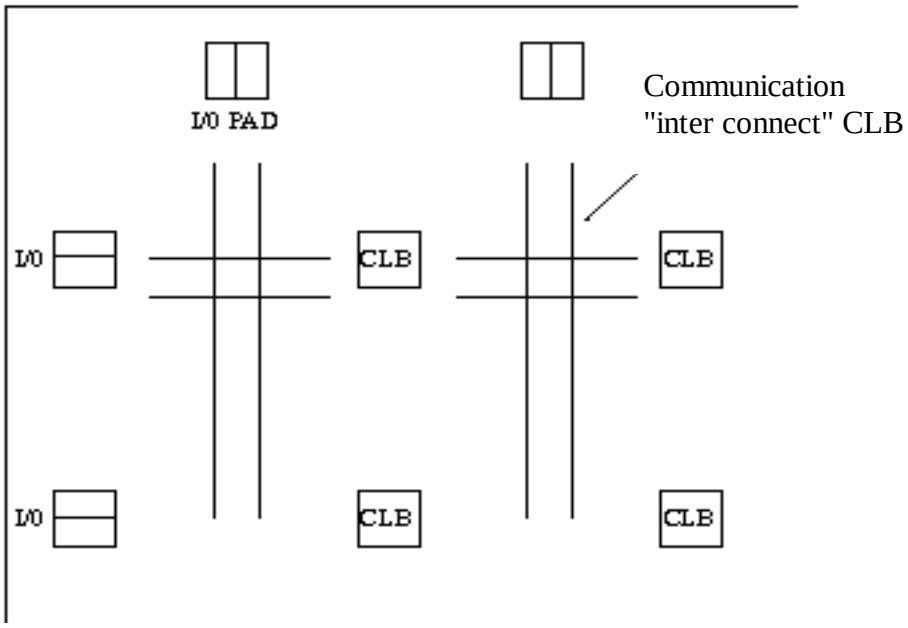
4.12 FPGA

– Field Programmable Gate Array (FPGA) : Approche plus évoluée que les simples PLD
CLB=Combinational Logic Block

– Contient des blocs logiques pouvant à loisir être configurés pour implanter différentes fonctions

Ex. : circuits combinatoires à 4 couches

– Contient aussi des circuits d'I/O entrées/sortie et lignes de communication interconnect



Fabricant, par exemple Xilinx

XC2064 ⇒ 1200 portes
64 CLB
58 I/O
122 Bascules

Le "Routage" des signaux n'est pas une tâche aisée.

– Fait par algorithmes spécialisés essais/erreurs
Ex : algo génétique

Un CLB peut contenir
- MUX
- Bascules
- Bloc de fonction (fonction UNIT programmable n'importe qu'elle fonction de 4 bits
Jusqu'à - 8 entrées
- 2 sorties

4.13 Logiciel de programmation

ROM – programmation directe : Il suffit de créer une liste d'adresses avec contenu et d'employer un programmeur d'EPROM, PROM.

PALs – ON doit optimiser la solution SOP : car pas assez de minterms disponibles dans le réseau pour les N entrées

⇒ Besoin de logiciel, exemple ABEL, Data I/o Corp, PALASM (gratuit de AMD), PAL-assembler, etc.

ABEL : langage qui permet une solution via 3 approches
- tdv
- Équations
- Diagramme d'état

PALASM : on doit fournir la forme SOP

Note : Parfois les logiciels de programmation avec programmeur d'EPROM ne trouvent pas toujours une solution optimale, le designer doit donc rester à l'affût !

Ex : Problème A > B

A : $[a_7 \ldots a_0]$
B : $[a_7 \ldots a_0]$

Solution – ABEL :256 minterm $\Rightarrow$ 10 PALS

- brillante en scindant le problème comparaisons bit à bit

1 bit > $a_0 > b_0$

$a_7 > b_7$

FPGA : Il faut vraiment employer un logiciel spécialisé fournit par le manufacturier (les FPGA sont trop différents d'un manufacturier à l'autre).

Exemple : Xilinx Foundation employé dans le cours.