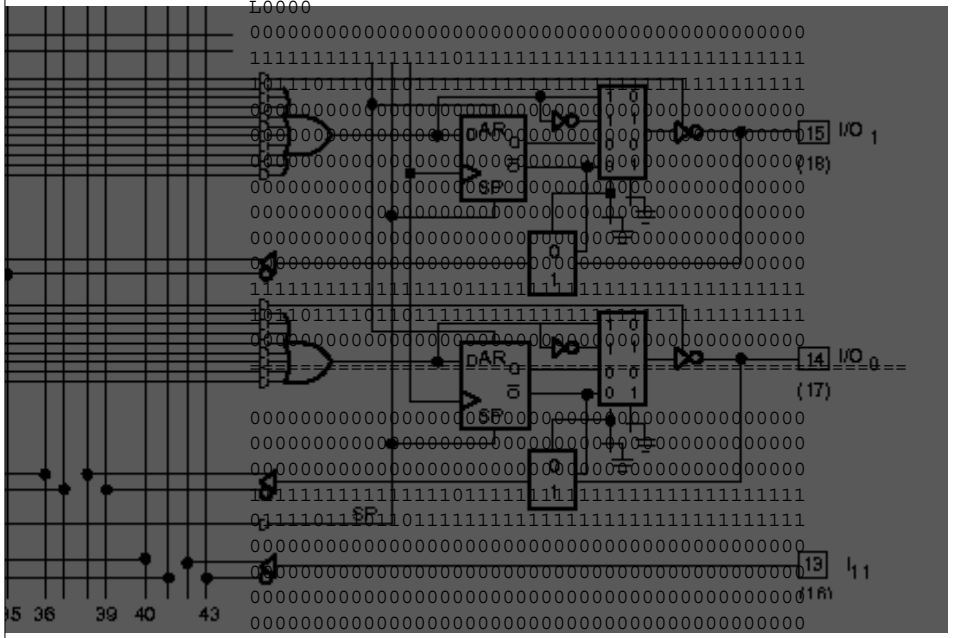
Composants logiques programmables

PAL, GAL, CPLD et FPGA.

ABEL, VHDL et JEDEC

Composants logiques programmables

```
module debut
title 'essai intelliflow'

debut device;
y7, y6, y5, y4, y3, y2, y1, y0 pin;
s0, s1, s2, data, enb pin;

h, l, z = 1, 2, .z.;
select = [s2,s1, s0];
output = [y7, y6, y5, y4, y3, y2, y1, y0];


equations
!y0 = (select == 0)& data;
!y1 = (select == 1)& data;
!y2 = (select == 2)& data;
!y3 = (select == 3)& data;
!y4 = (select == 4)& data;
!y5 = (select == 5)& data;
!y6 = (select == 6)& data;
!y7 = (select == 7)& data;

output.oe=enb;

end
```




```
ABEL 6.20 Data I/O Corp.
JEDEC file for: P22V10 V9.0
Created on: Tue Feb 12 18:52:02 2002

essai intelliflow
*
QP24* QF5828* QV0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS y7:17 y6:20 y5:16 y4:21 y3:15 y2:22 y1:14 y0:23 s0:1 s1:2 s2:3*
NOTE PINS data:4 enb:5*
L0000
00000000000000000000000000000000000000000000
11111111111111110111111111111111111111111111
10111101110110111111111111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
11111111111111110111111111111111111111111111
10111101110110111111111111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
11111111111111110111111111111111111111111111
01111011101101111111111111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000*
L5808
01010101000001010101*
C5742*
A079
```

Sommaire

1. Intérêt de ces composants
2. Différentes familles disponibles
3. Développement d’un circuit PAL/GAL
4. Présentation matérielle des GALs
5. Description graphique
6. Le langage ABEL
7. Le langage VHDL
8. Le fichier JEDEC
9. Exemple de fichiers .ABL .JED associés
10. Exercices

1. Intérêt de ces composants

- Réaliser des fonctions non prévues par les fabricants de circuits.
- Augmenter et faciliter l’intégration de composants logiques dans un boîtier. Cela permet de réduire la taille des cartes, même pour de petites séries (jusqu’à 25%).
- Réduire le temps de développement des applications par l’emploi de la simulation.
- Développer la flexibilité des applications et donc faciliter leur maintenance.
- Réduire le nombre de composants à garder en stock.

2. Différentes familles disponibles

- Les premiers de ces circuits ont été produits en 1978 par Monolitics Memories. Ce sont des PALs (Programmable Array Logic).
- PLD est le terme général pour les petits composants programmables par l'utilisateur (Programmable Logic Device).
- Les PLD standards sont constitués d'un réseau de ET, suivi d'un réseau de OU. L'un ou l'autre, ou les deux de ces réseaux sont programmables. Les entrées sont reliées au réseau de ET, pour fournir les termes de produit. Chacun de ces termes est relié aux portes OU pour fournir la fonction désirée.

2. Différentes familles disponibles

- PAL.

  Dans les PALs (Programmable Array Logic), le réseau de ET est programmable, et le réseau de OU est fixe.

  Il peut être quelques fois suivi de bascules.

  Ils sont réalisés en technologie bipolaire (TTL) et ne peuvent être programmés qu’une seule fois.

2. Différentes familles disponibles

- GAL.

  Les GALs (General Array Logic) sont comme les PALs, mais avec une cellule de sortie (OLMC) dont on peut programmer la nature combinatoire ou séquentielle.

  Chaque OLMC intègre une partie du réseau de « OU ».

  On peut les effacer et les reprogrammer. Ils sont réalisés en technologie CMOS.

  Attention, leur tension d’alimentation Vcc est de 5V.

  Comme les PALs, ils comptent environ 30 broches.

2. Différentes familles disponibles

- CPLD.

  Les CPLDs (Complex Programmable Logic Device) sont des PLDs (donc des GALs) complexes .

  Ils sont constitués de plusieurs GALs et ces GALS sont reliés entre eux à l ’aide d ’une matrice d ’interconnexion.

  L ’avantage de cette architecture (vis à vis des FPGAs) réside dans la matrice d ’interconnexion qui permet d ’avoir un temps de propagation fixe et connu à l ’avance.

  On peut les effacer et les reprogrammer. Ils sont réalisés en technologie CMOS.

2. Différentes familles disponibles

- FPGA (Field Programmable Gate Array)
  - C’est un ensemble de cellules régulièrement disposées (dans une matrice XY) appelées macrocellules.
  - Chaque macrocellule contient un élément de logique combinatoire et séquentielle programmable.
  - Un FPGA contient un très grand nombre de macrocellules permettant une très grand nombre de combinaisons.
  - La logique combinatoire programmable se fait avec une LUT (Lock Up Table) : basé sur un multiplexeur.
  - Le routage entre macrocellule se fait à l’aide d’une grande et complexe matrice. Le routage n’assure pas un temps de propagation constant (à priori).

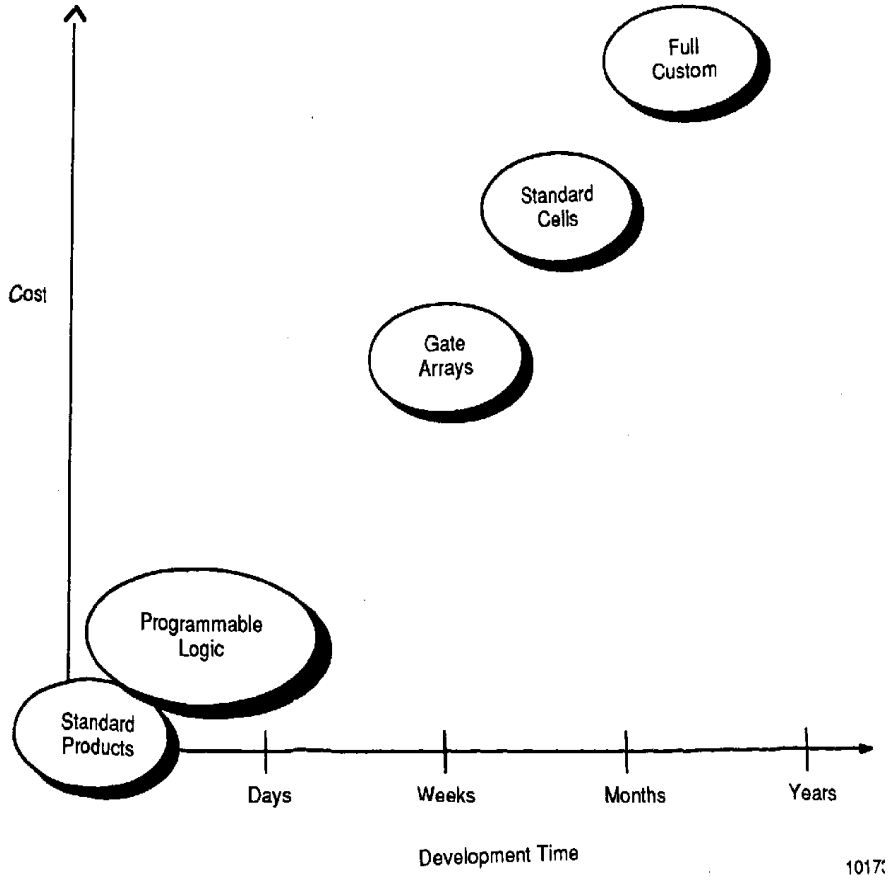
2. Différentes familles disponibles

- ASIC (Application Specific Integrated Circuits)
    - C’est un ensemble de cellules régulièrement disposées (dans une matrice XY) sur un substrat de silicium.Chaque cellule contient des portes logiques (le plus souvent des NAND), des transistors et des résistances.
    - Les métallisations de connexion dans et entre les cellules dépendent de la fonction à réaliser. Quand les résultats de simulation de la fonction sont corrects, le fondeur fabrique le circuit. Celui-la livre un produit directement utilisable. Un seul circuit peut compter jusqu’à 100.000 portes logiques.

2. Différentes familles disponibles

Diagramme

coût/développement



3. Développement d’un circuit PAL/GAL/CPLD

C’est à l’utilisateur (électronicien) de programmer chacun des circuits qu’il veut utiliser, selon la fonction qu’il veut réaliser.

On décompose le travail en 5 étapes:

3. Développement d’un circuit PAL/GAL/CPLD

a) Etablir la description de la fonction à implémenter dans le circuit. Cela peut se faire par description graphique, ou par un langage adapté : ABEL™ ou VHDL.

ABEL et VHDL sont des standards logiciels de description de fonctions logiques (et donc de composants).

Cette description est placée dans un fichier texte FILE.ABL ou FILE.VHDL. Pour la description graphique, elle dépend du logiciel utilisé

3. Développement d’un circuit PAL/GAL/CPLD

b) Simuler la fonction décrite.

c) Déterminer le boîtier à utiliser, en fonction de différentes contraintes :

- Vitesse
- Consommation
- Préférence pour un fabricant
- Gamme de température
- Technologie, etc…

3. Développement d’un circuit PAL/GAL/CPLD

d) Compiler le fichier de description pour obtenir le fichier texte FILE.JED.

JEDEC est un standard matériel de description de fonction.

e) Transférer, physiquement, les informations du fichier jedec dans le composant. Il faut pour cela utiliser un appareil appelé programmateur de PAL.

Il est raccordé à un PC, qui contient le fichier file.jed.

Sinon, le composant peut être programmé “IN SITU”, c-a-d directement sur la carte applicative.

3. Développement d’un circuit PAL/GAL/CPLD

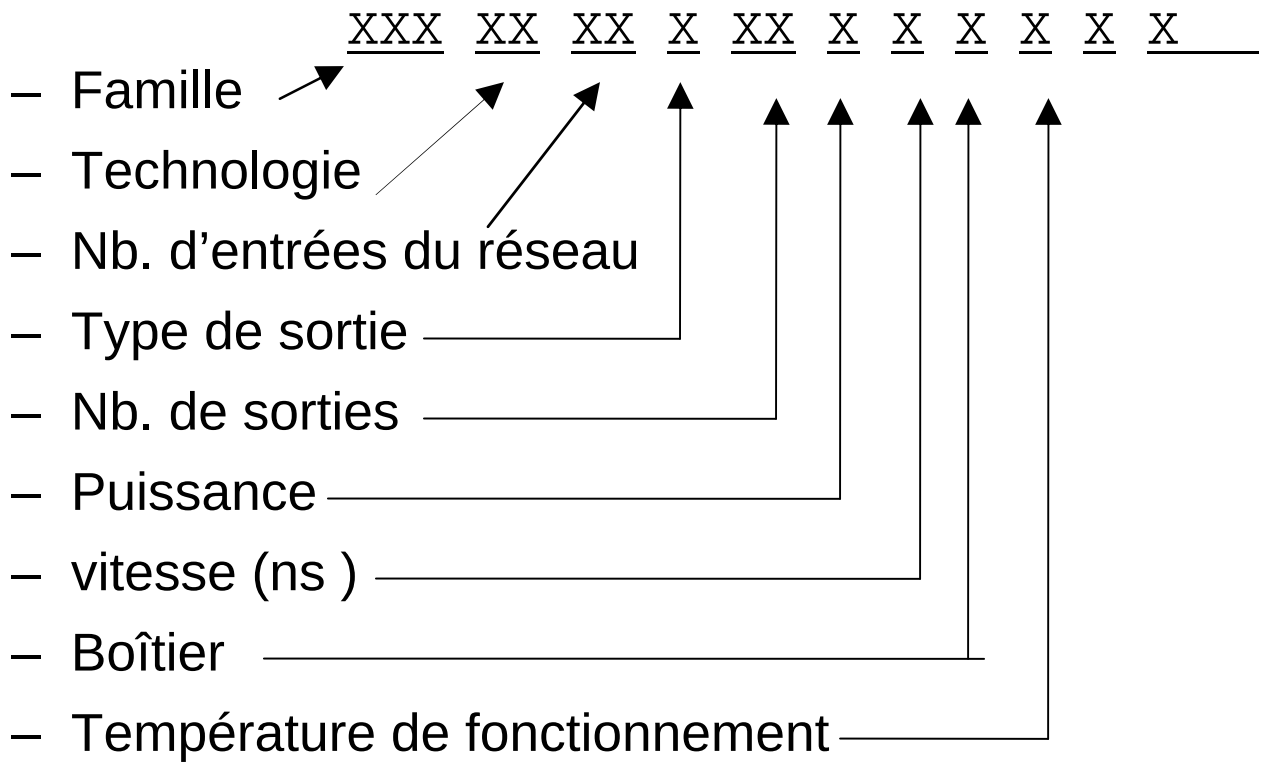
Ces étapes se font plus ou moins automatiquement avec des logiciels spécialisés

Nous utiliserons ispLEVER et ispVM, logiciel distribués gratuitement par Lattice.

Un exemple de simulation sera fait sous OrCAD.

4. Présentation matérielle des GALs

4.1 Références



4. Présentation matérielle des GALs

4.1 Références

Ex: PAL CE 22 V10 H -25 P C

- ***PAL*** ***C***mos***E***lectrically erasable,
- ***22*** broches de signal,
- des sorties ***V***ersatiles (combinatoires ou séquentielles),
- ***10*** broches de sortie (éventuellement configurables en entrée),
- ***H***alf power (Icc<150mA), Tpd=***25*** ns,
- boîtier ***P***lastique DIP 24 broches,
- gamme de température ***C***ommerciale 0°C à +75°C.

22V10 = 22 entrées dont 10 sont configurables en sortie

4. Présentation matérielle des GALs

- 4.2 Organisation d’un GAL22V10 en mode lecture

Ce circuit comprend :

Une matrice ligne-colonne dont certaines intersections deviendront des « ET » par programmation.

Des OLMC (Output Logic Macro Cell) programmables pour partie en bloc et pour partie individuellement. On détermine ainsi la nature combinatoire ou séquentielle du circuit. Il y a autant d’OLMC que de sorties.

4. Présentation matérielle des GALs

- 4.2 Organisation d’un GAL22V10 en mode lecture

On montre en Algèbre que n’importe quelle fonction logique de 4 (ou n) variables a, b, c, d, peut se mettre sous la forme F = /a.b.c./d + a./b.c.d + etc.
On appelle “terme de produit“ chacune des expressions /a.b.c./d, a./b.c.d

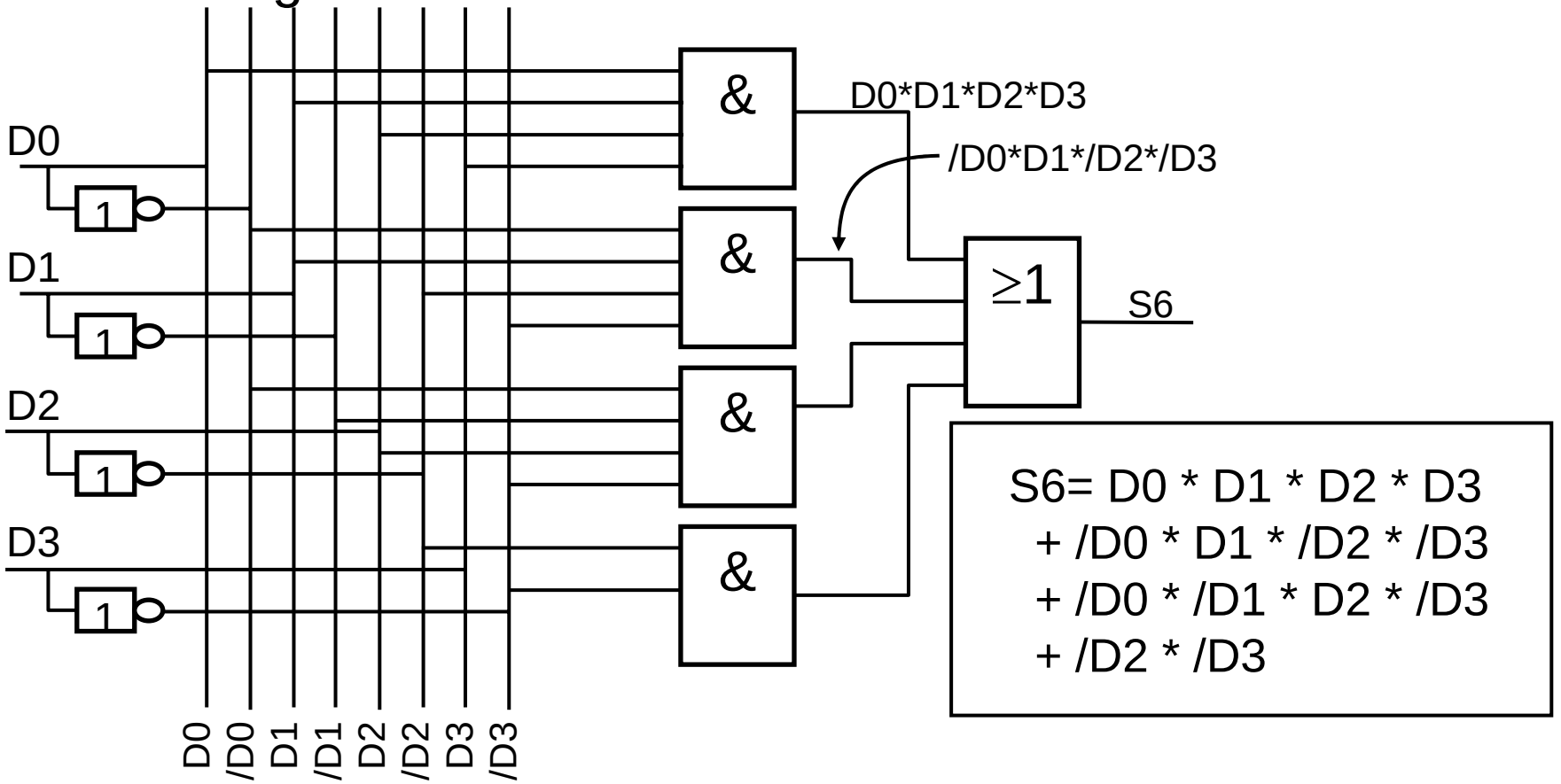
Ainsi, avec 4 variables d’entrée, on peut faire la “somme” de 16 (2^4) termes de produit au maximum

Ces termes de produit ne sont autres que des opérateurs “ET”

Circuit d'exemple

4. Présentation matérielle des GALs

- 4.2 Organisation d'un GAL22V10 en mode lecture

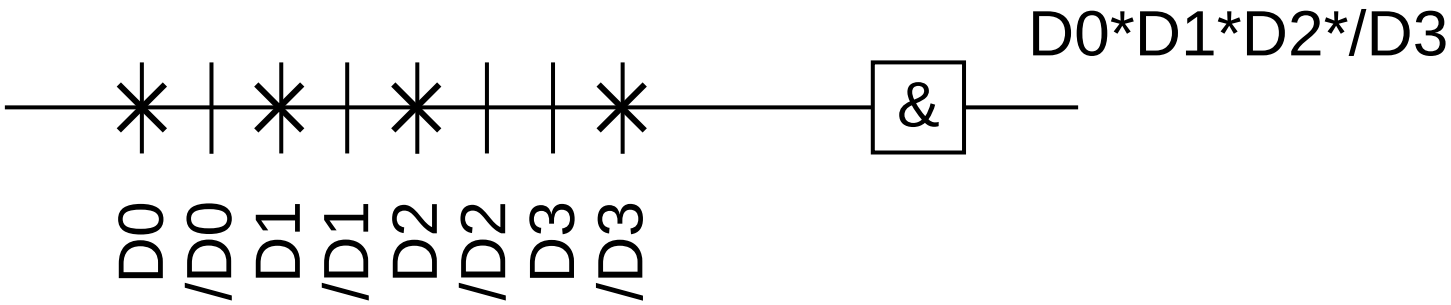


S6= D0 * D1 * D2 * D3
+ /D0 * D1 * /D2 * /D3
+ /D0 * /D1 * D2 * /D3
+ /D2 * /D3

Convention

4. Présentation matérielle des GALs

- 4.2 Organisation d'un GAL22V10 en mode lecture

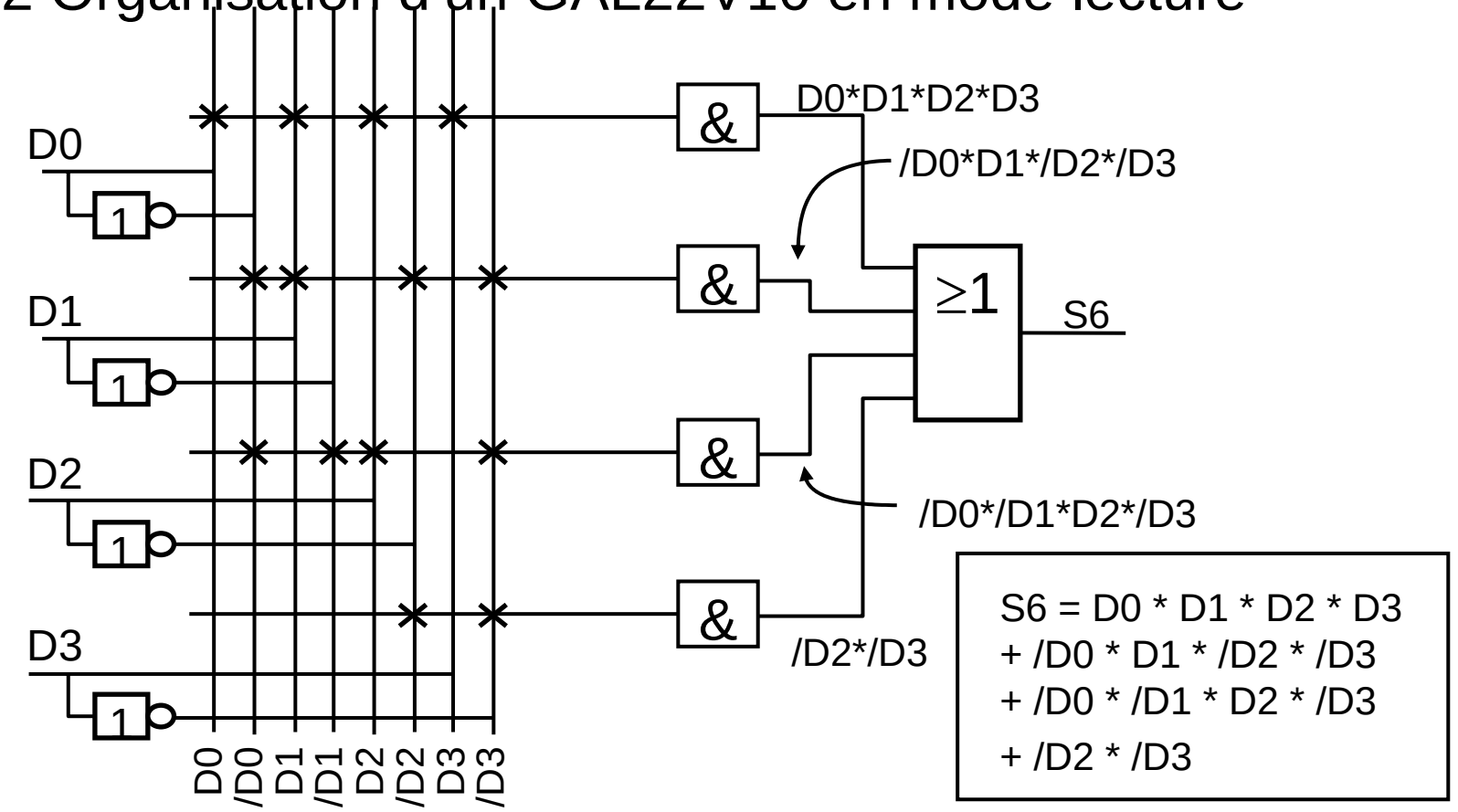


× : Participe au terme ET

Transformation de la représentation du circuit

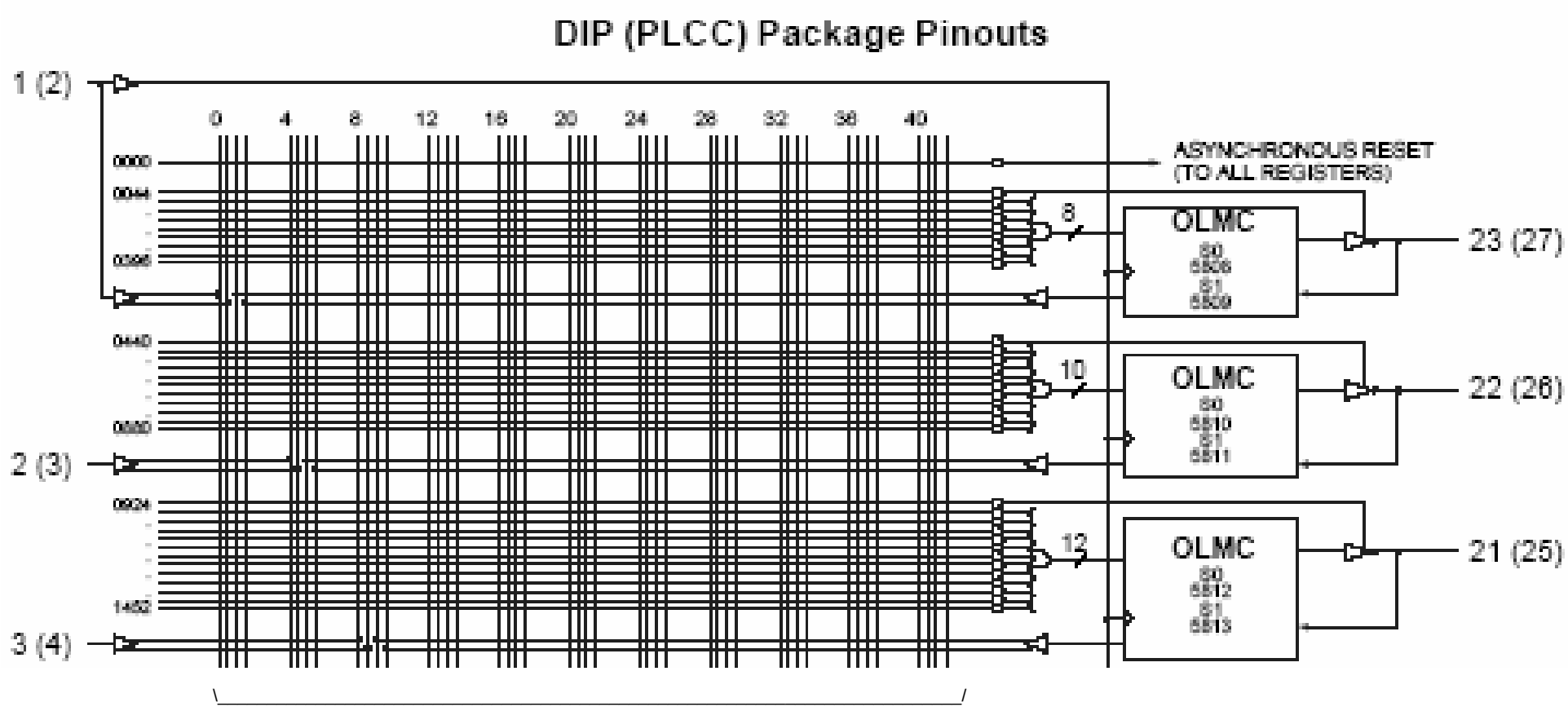
4. Présentation matérielle des GALs

- 4.2 Organisation d'un GAL22V10 en mode lecture



S6 = D0 * D1 * D2 * D3
+ /D0 * D1 * /D2 * /D3
+ /D0 * /D1 * D2 * /D3
+ /D2 * /D3

4. Présentation matérielle des GALs



matrice « ET »

4. Présentation matérielle des GALs

- La matrice ET.

  Les 22 signaux E et S sont disponibles sous leur forme directe ou complémentée. La matrice comprend 2*22 = 44 colonnes repérées 0-43.

4. Présentation matérielle des GALs

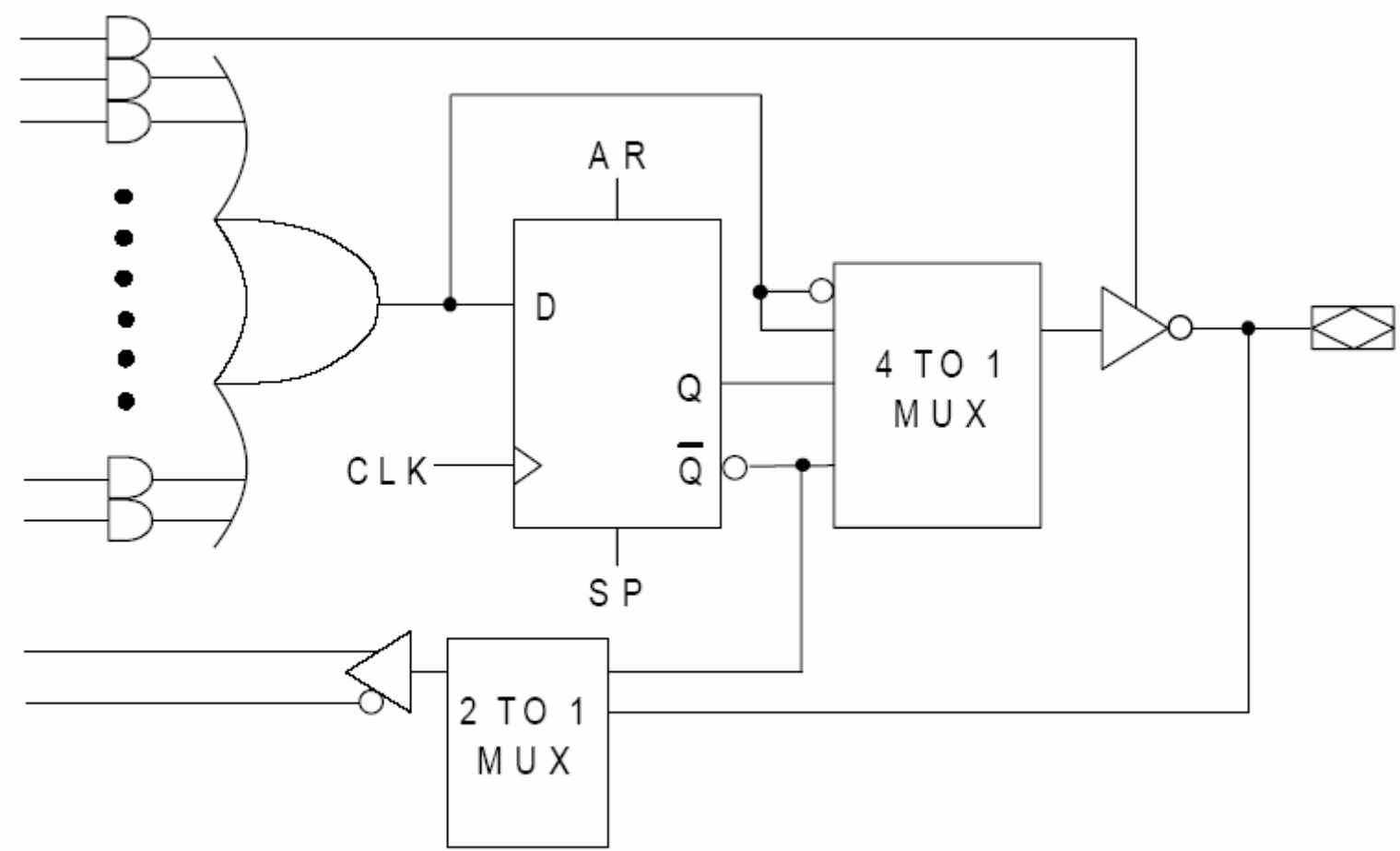
- Les OLMC

Une OLMC est un système binaire dont on peut programmer la nature combinatoire ou séquentielle.

La programmation se fait avec 4 bits (22V10 Lattice).

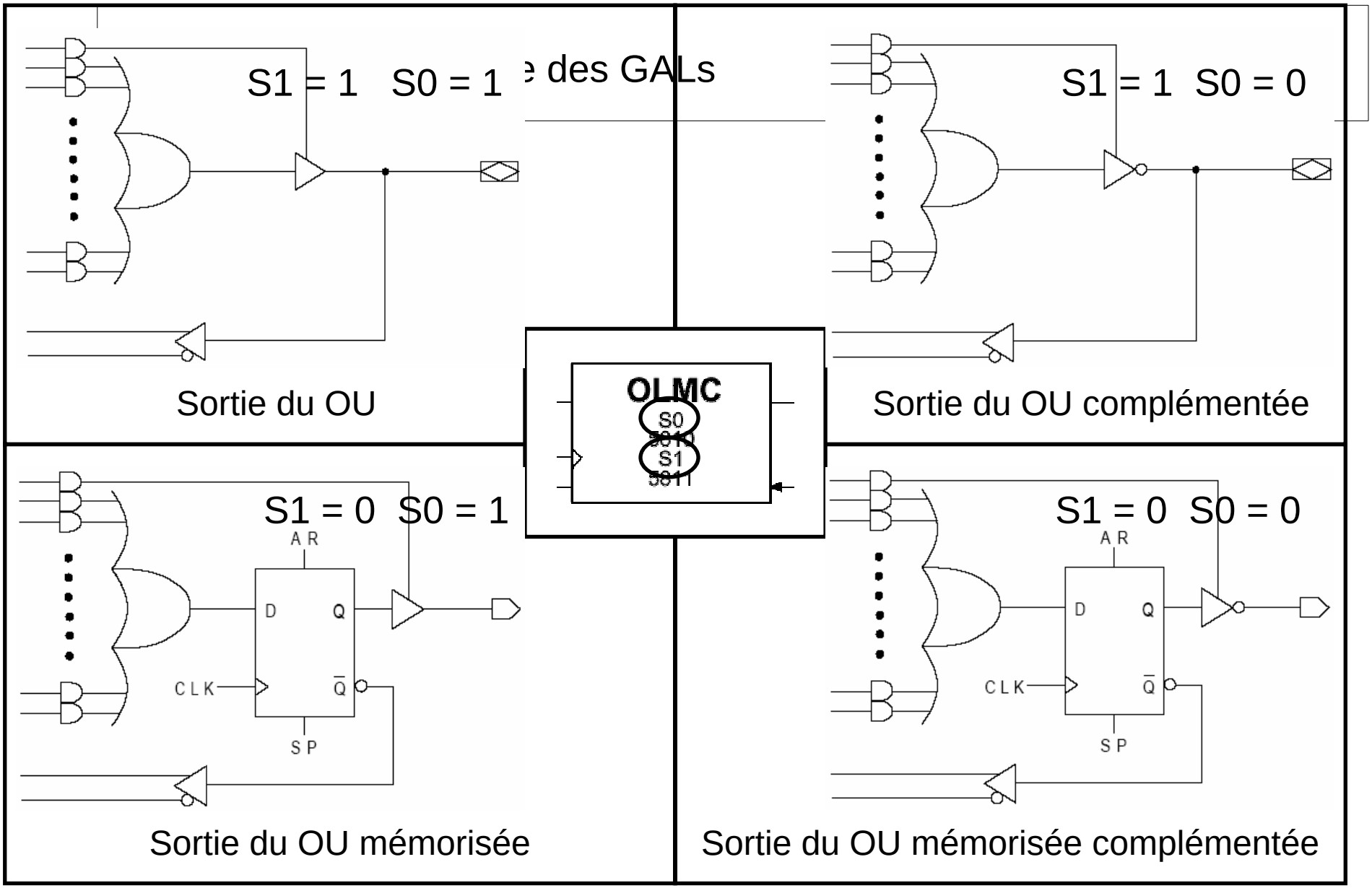
L’OLMC à la forme suivante :

4. Présentation matérielle des GALs



GAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

Cellule de sortie programmable : OLMC



4. Présentation matérielle des GALs

Les adresses

Chaque information programmable a son adresse.
Les connexions, qui formeront les termes de produit, sont numérotées. L'adresse du fusible correspondant est :
adr. = (N° ligne * nbre de colonnes) + N° colonne

Remarques :

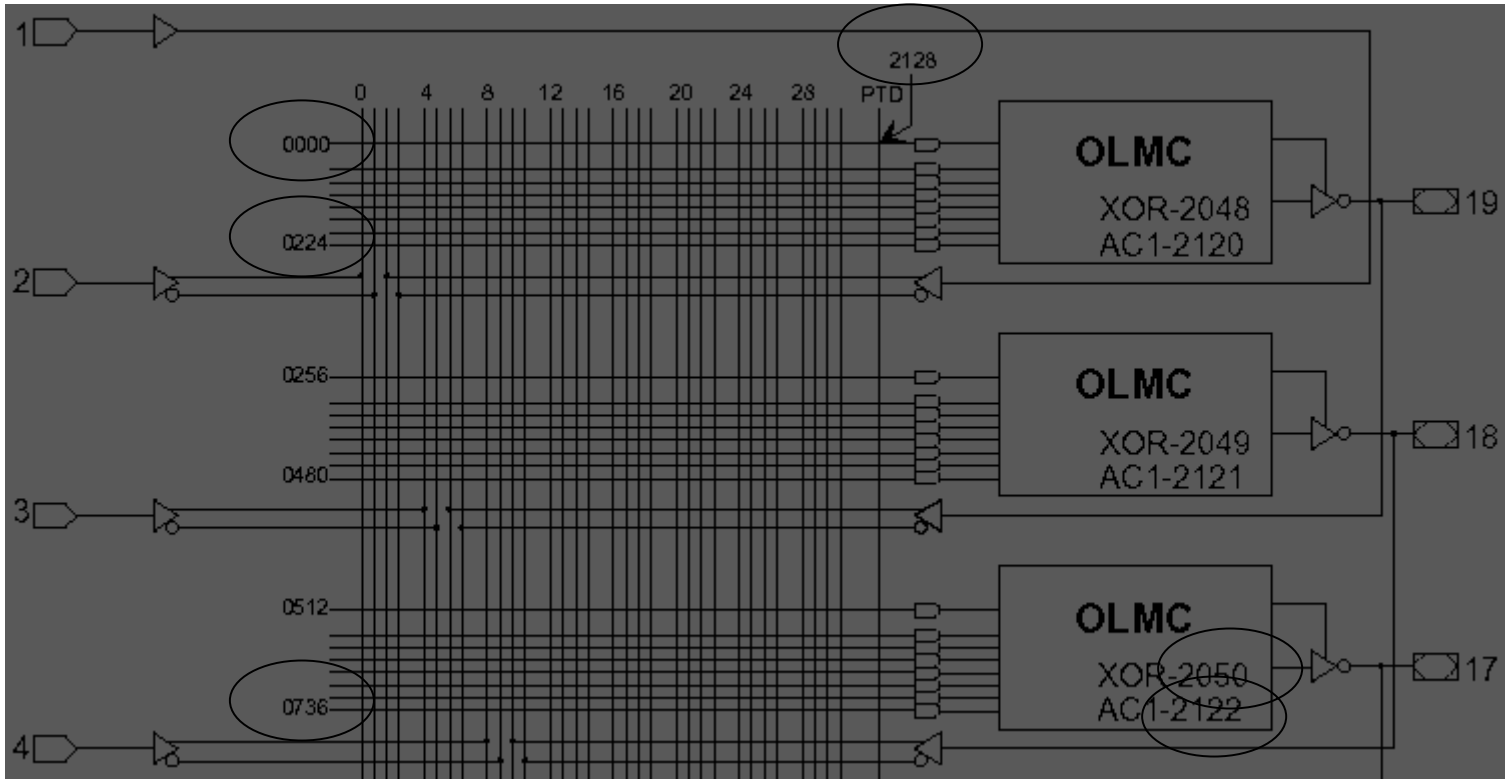
La validation du terme de produit a lieu à la programmation, pas pendant l'usage normal du composant.

L'UES occupe 64 bits, soit 8 octets.

Cartographie du 16V8 Lattice :

0000	2047	: matrice ET
2048	2055	: XOR(n) (SL1n)
2056	2119	: User Electron. Sign.
2120	2127	: AC1(n) (SL0n)
2128	2191	: validation des termes de produit
	2192	: bit SYN
	2193	: bit AC0 (SG1)

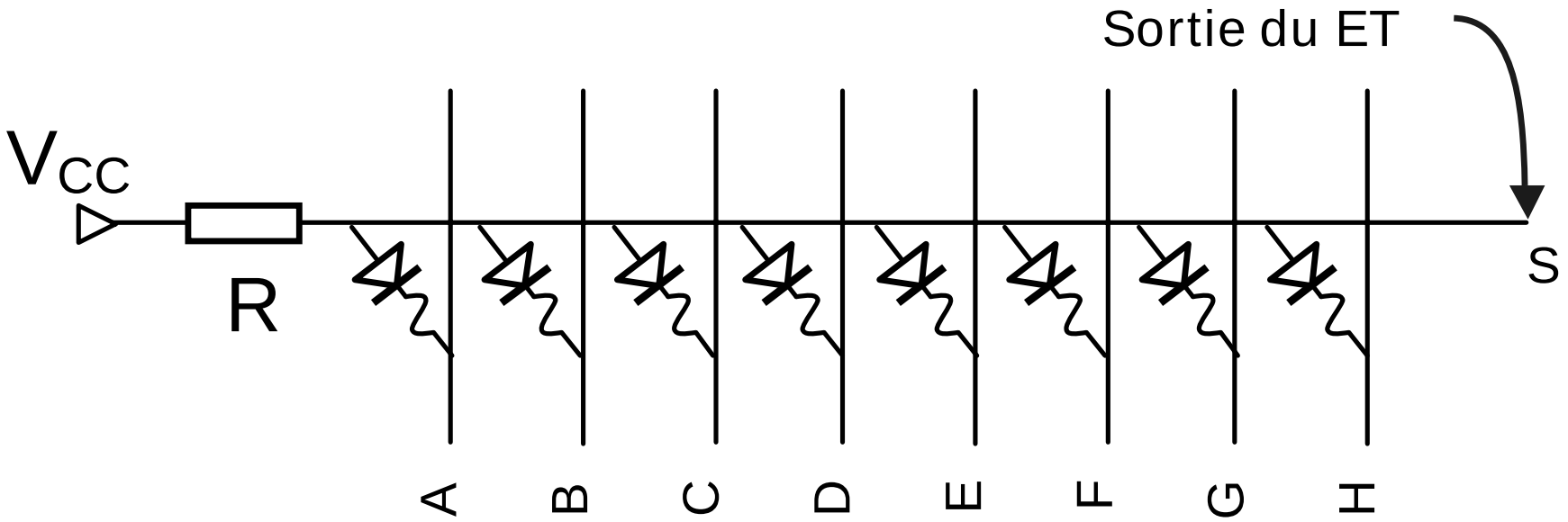
4. Présentation matérielle des GALs



adresses

Constitution du ET programmable

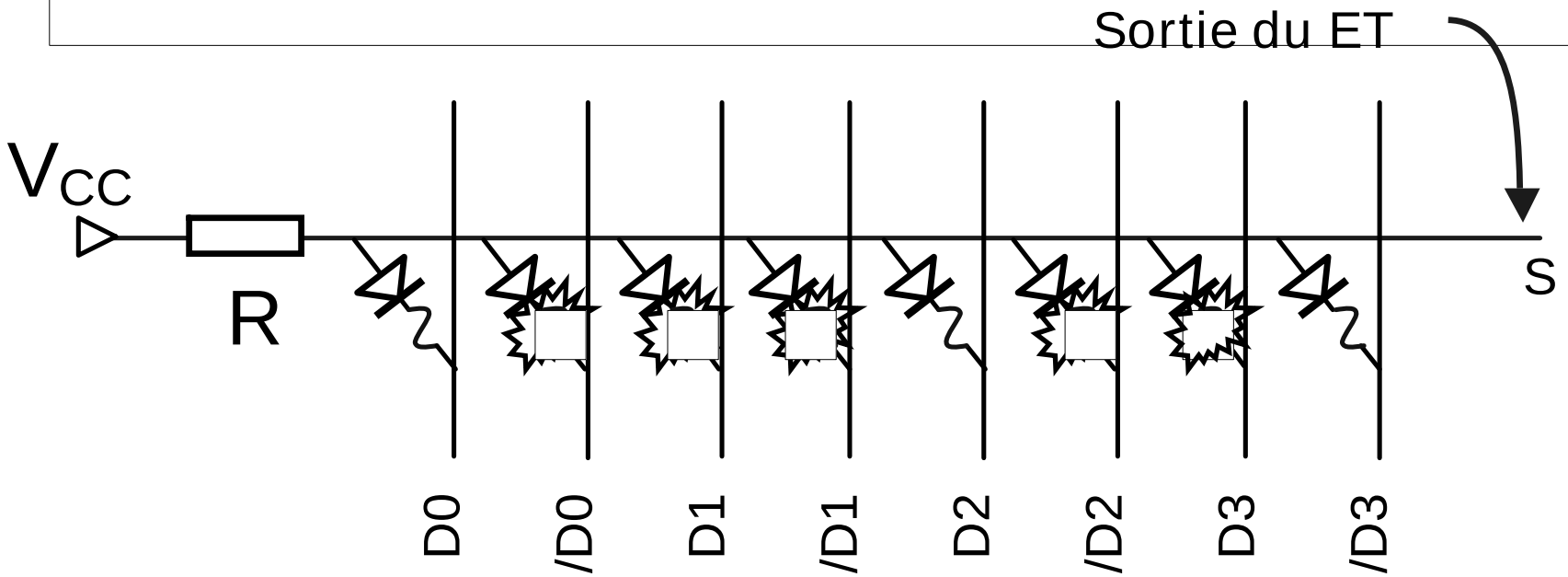
4. Présentation matérielle des GALs



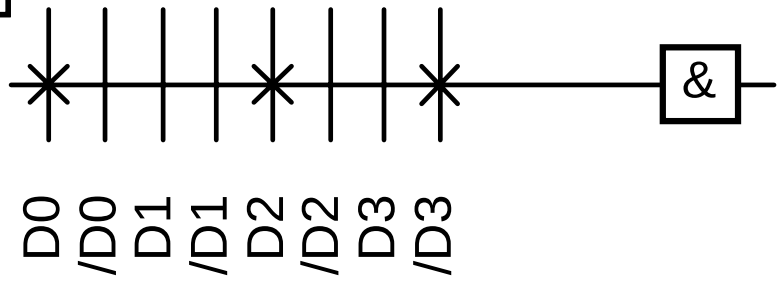
S=A*B*C*D*E*F*G*H

Constitution du ET programmable

4. Présentation matérielle des GALs

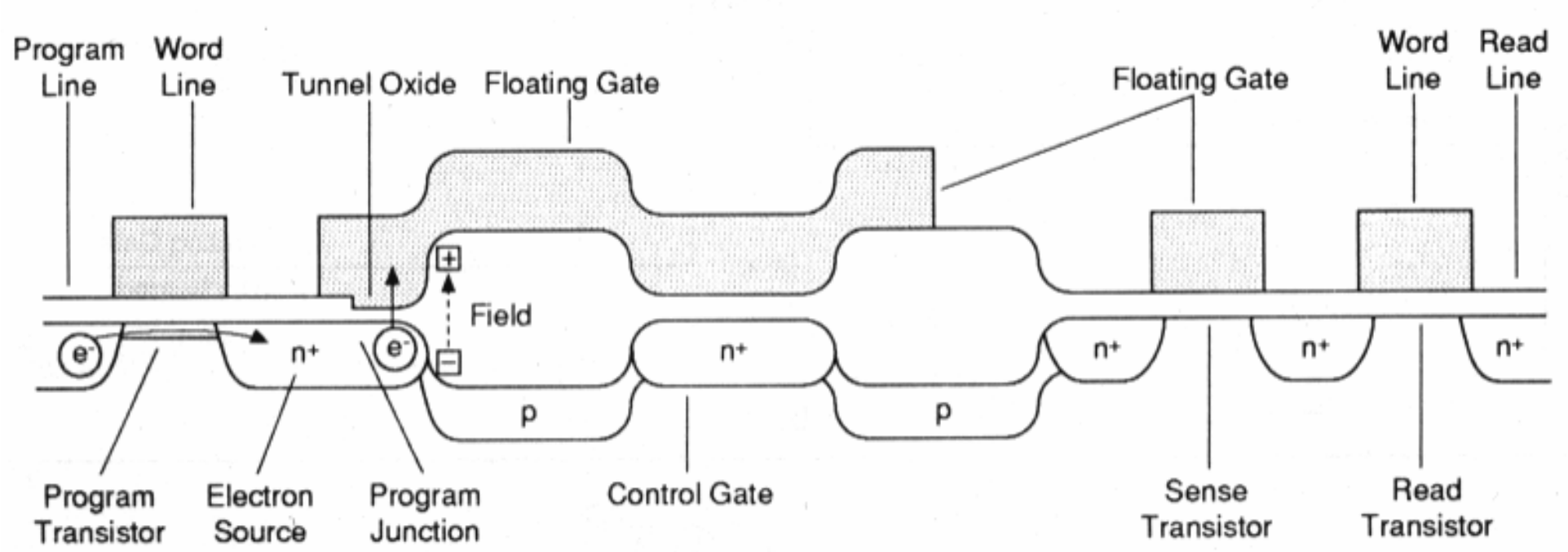


S=D0*D2*/D3



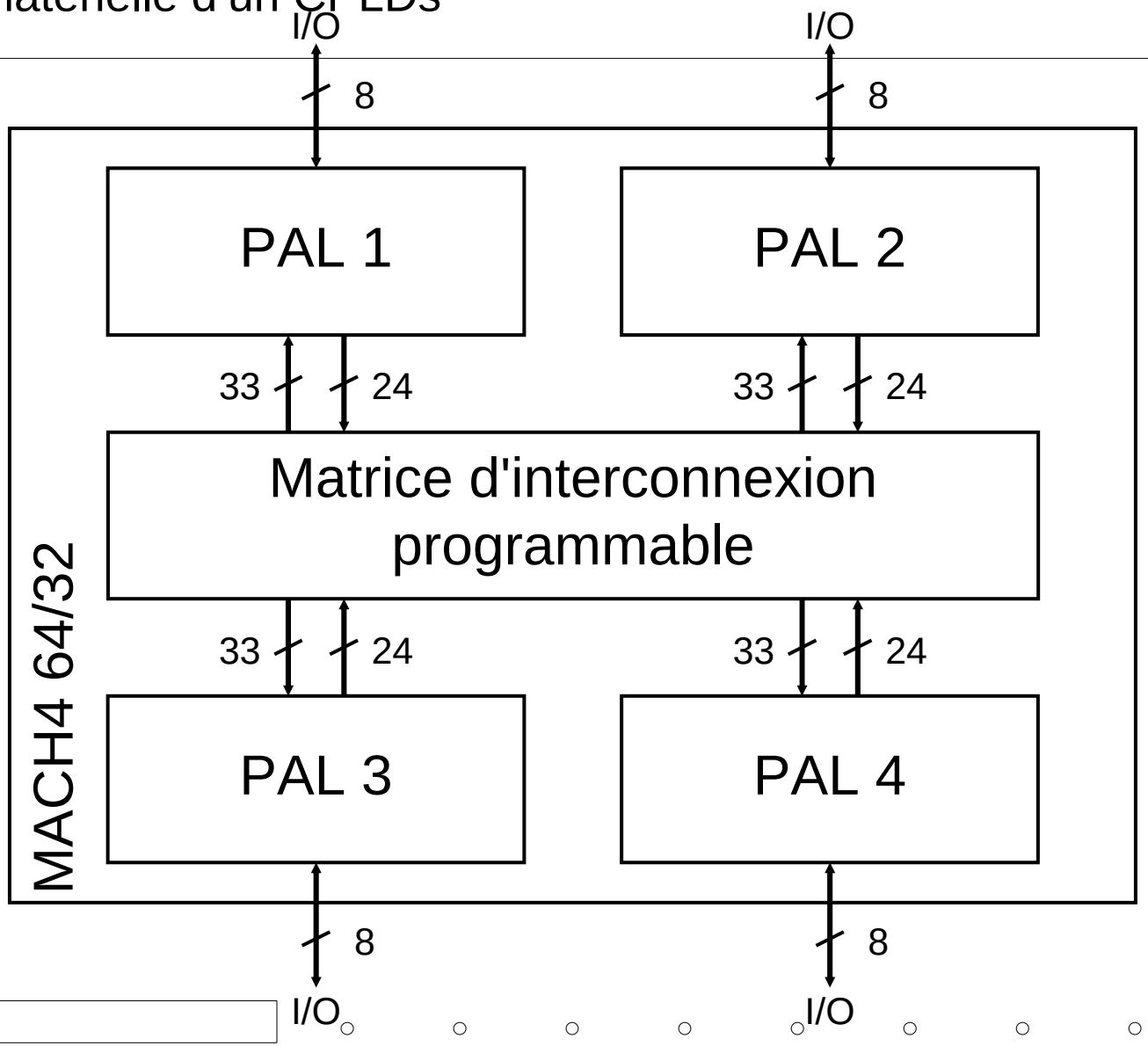
Technologie effaçable

4. Présentation matérielle des GALs



Structure d'un MACH4

4. Présentation matérielle d'un CPLDs



5. La description graphique

- La description graphique

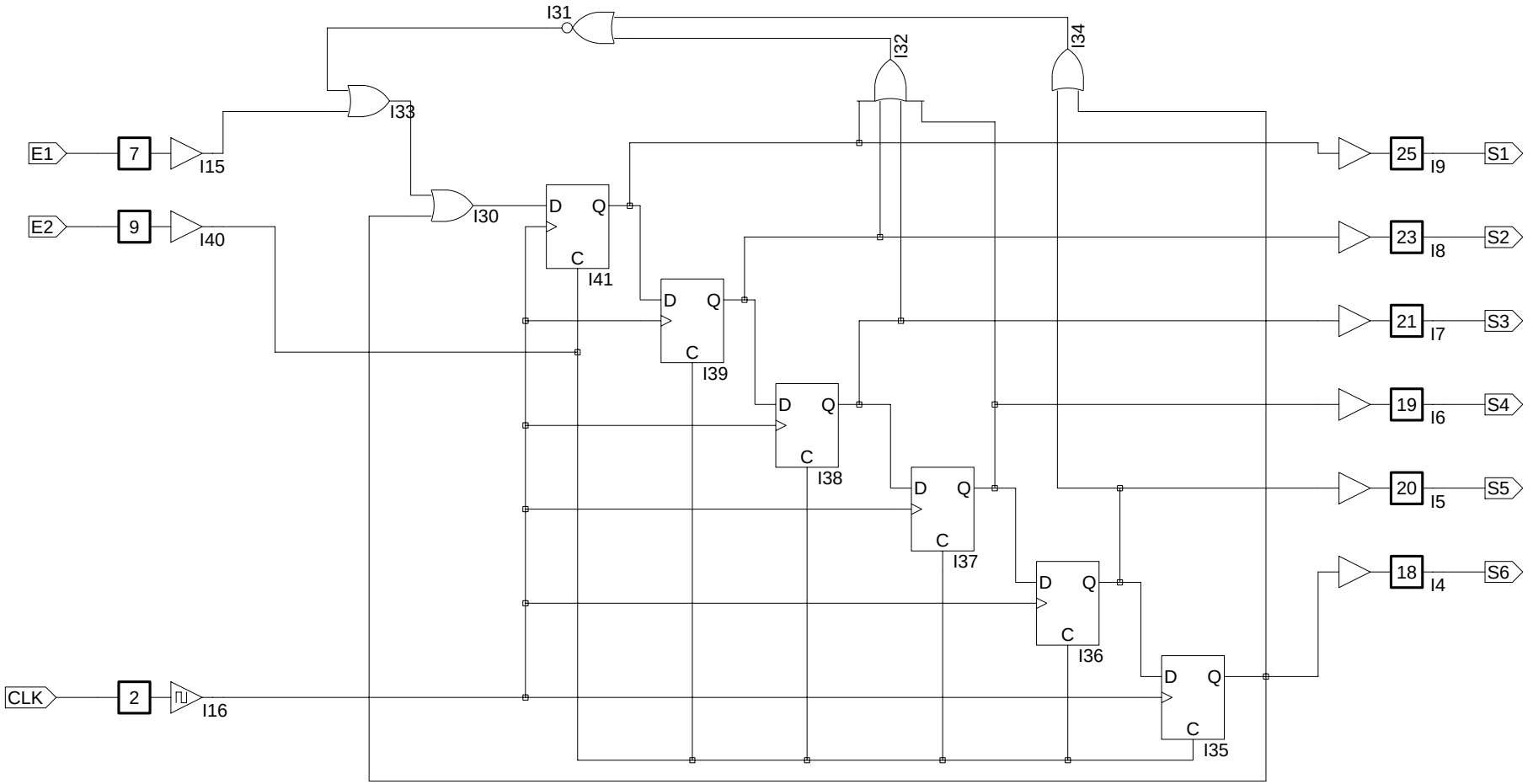
Cette méthode de programmation est similaire à la conception d'un schéma logique en composants discrets.

L'utilisateur dispose de librairies contenants des opérateurs logiques combinatoires et séquentielles simples et complexes. Il les place et réalise les équipotentielles.

Il faut ensuite spécifier sur quelles broches physiques les entrées et sorties seront programmées.

Le fichier JEDEC sera ensuite créé à partir de cette description

5. La description graphique



6. Le language ABEL

ABEL est un standard logiciel de description de fonctions logiques (et donc de composants).

Un exemple de fichier .ABL permet de voir quelques éléments de ce langage:

Structure d'un fichier ABEL

```
MODULE Logique

TITLE 'Programmation didactique'

// Pour pouvoir exposer ...

DECLARATIONS

// Variables d'entrée
A,B,C,D,E pin  14,15,16,17,18;

// Variables de sortie
del1,del8 pin  31,24 istype 'com';

EQUATIONS
del1=!(A & B & C & D & E);
del8=!(A # B # C # D # E);

TEST_VECTORS
  ([A,B,C,D,E] -> [del1, del8]);
   [0,0,0,0,0] -> [1,1];  // Tout ét
   [0,0,0,1,0] -> [1,0];  // del 8 a

END Logique
```

① Section d'entête

② Section de déclarations

③ Section de description

④ Section de test

⑤ Déclaration de fin de fichier

Sections d’entête et de fin

```
MODULE Logique

TITLE 'Programmation didactique'

// Pour pouvoir exposer ...
.
.
.
.
.
.
.

END Logique
```

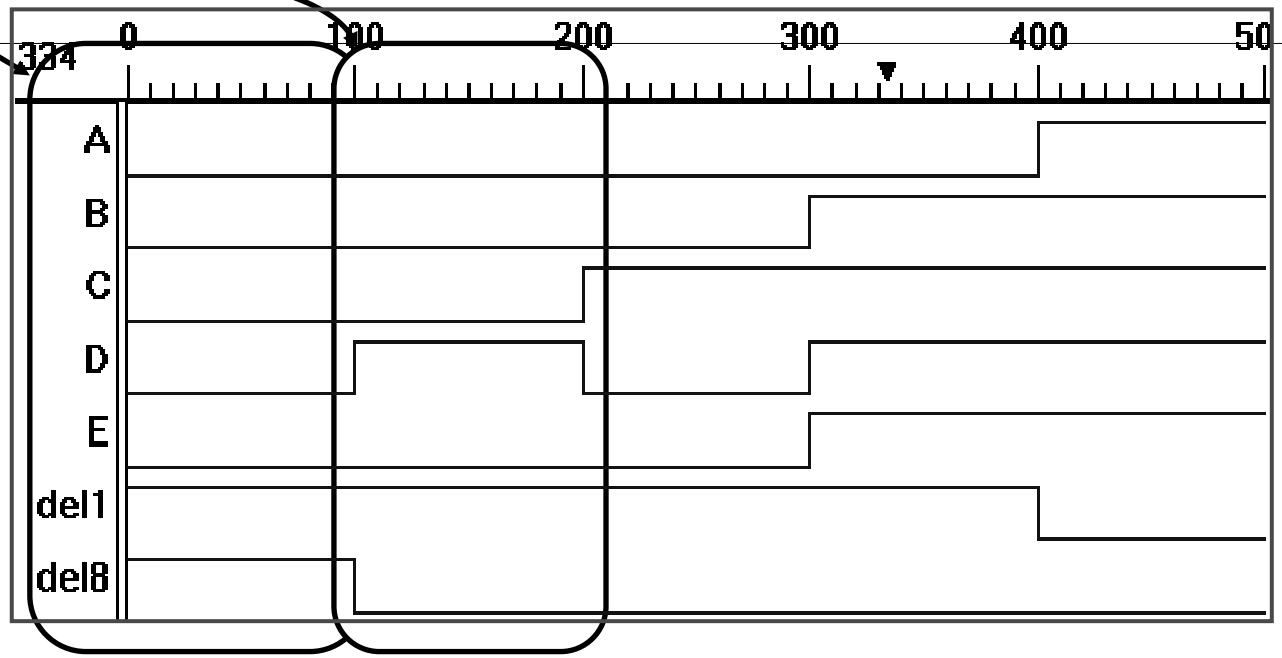
Section des déclarations

Spécifications matérielles de compléments sur les sorties :
Impose l'utilisation du complément

```
.
.
DECLARATIONS

A,B,C,D,E pin  14,15,16,17,18;

del1,del8 pin  31,24      istype 'com';

Y    pin  12 istype 'com, invert';
.
```

Section de test

```
TEST_VECTORS
  ([A,B,C,D,E] -> [del1, del8]);
   [0,0,0,0,0] -> [1,1];  // Tout éteint
   [0,0,0,1,0] -> [1,0];  // del 8 allumée
```



Section de description logique

EQUATIONS

..

TRUTH_TABLE

..

STATE_DIAGRAM

..

FUSES

..

Description logique par équations

```
EQUATIONS

      del1=!(A & B & C & D & E);

      del8=!(A # B # C # D # E);
```

Groupe de variables (set) ou BUS

```
N = [A3,A2,A1,A0];  //msb:A3 lsb:A0
```

```
S = [B7..B0];       //msb:B7 lsb:B0
```

6. Le language ABEL

Comme tous les langages, ABEL a un vocabulaire, une grammaire et une syntaxe.

Opérateurs logiques

non	!
et	&
ou	#
XOR.	$
XNOR	!$

Opérateurs relationnels

différent	!=
égal	==
inférieur	<
inférieur ou égal	<=
supérieur	>
supérieur ou égal	>=

Description logique

broche	.pin
horloge	.clk
sortie de rebouclage	.bf
validation des sorties	.oe
.D	Entrée D de bascule D.
.J	Entrée J de bascule JK.
.K	Entrée K de bascule IK.
.R	Entrée R de bascule RS.
.S	Entrée S de bascule RS .
.T	Entrée T de bascule Toggle.
.Q	Sortie.
.PR	Preset.
.AP	Asynchrone preset.
.AR	Asynchrone reset.
.SP	Synchrone preset
.SR	Synchrone reset
.LE	Entrée de validation du verrou
.LH	Entrée de val. (high) du verrou
.LD	Registre de préchargement

Attributs des signaux

signal combinatoire	‘com’
sortie active à 0	‘invert’
sortie active à 1	‘buffer’

Opérateurs combinatoires

Exemples

Écriture classique	ABEL
S = /A	**S=!A**
S = A + B	**S=A#B**
S = A * B	**S=A&B**
S = /(A + B)	**S=!(A#B)**
S = /(A * B)	**S=!(A&B)**
S = /(A ⊕ B)	**S=A!$B**
S = A ⊕ B	**S=A$B**

Description logique par table de vérité

```
N = [A3,A2,A1,A0];   // Definition de N

TRUTH_TABLE  // Definition par la TDV

(N -> [SA, SB, SC, SD, SE, SF, SG] );

 0 -> [1,1,1,1,1,1,0];  // Affichage 0
 1 -> [0,1,1,0,0,0,0];  // Affichage 1
 2 -> [1,1,0,1,1,0,1];  // Affichage 2
 3 -> [1,1,1,1,0,0,1];  // Affichage 3
```

Table de vérité

N	SA	SB	SC	SD	SE	SF	SG
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1

Description séquentielle / combinatoire

Pour définir des sorties combinatoires :

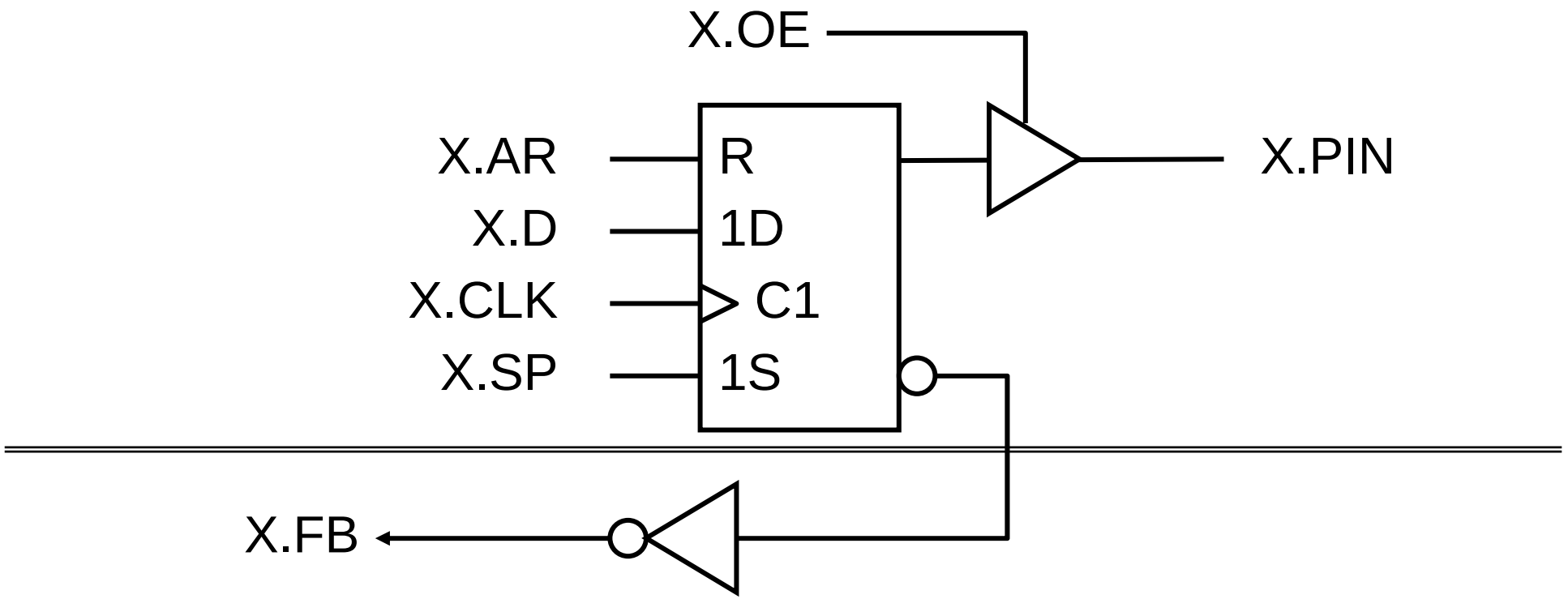
csrom,csram pin 13,12 istype 'com';

Pour définir des sorties séquentielles :

sortie istype 'buffer,reg_d';

Ces options déterminent la configuration de l'OLMC

Description séquentielle



Description séquentielle

Les extensions caractérisent le fonctionnement des étages de sorties.

.AR	Reset asynchrone
.CLK	Horloge des registres de sortie
.D	entrée D d'une bascule D
.Jentrée	J d'une bascule JK
.Kentrée	K d'une bascule JK
.OE	Sortie trois états

Exemple:

```
sortie istype 'buffer,reg_d';
        equations
                sortie.clk=h;
                sortie.oe=oe;
                sortie.ar=r;
```

Description séquentielle

Exemple d'un compteur **4 bits avec mise à 0 synchrone**

4 sorties regroupées dans un vecteur appelé **out**;

r est l'entrée de mise à 0 synchrone active à 1

et H l'entrée d'horloge.

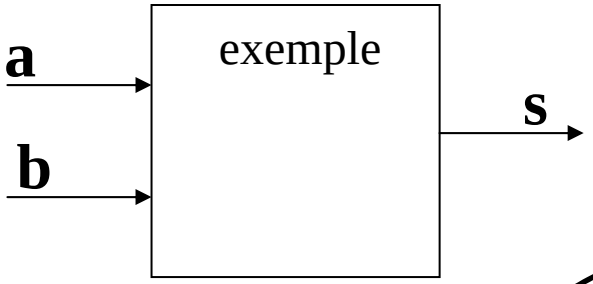
```
out=[q3..q0];
equations
        out.clk = H;
        when (r==1) then out : = 0; else out := out+1;
```

7. Le langage VHDL

Le langage VHDL (**V**ery high speed integrated circuit, **H**ardware **D**escription **L**anguage) a été créé pour le développement de circuits intégrés logiques complexes.

VHDL a des domaines d'utilisation plus vastes que la synthèse de circuits logiques en ABEL.

1 – L'entité : L'entité décrit l'interface entre le circuit et le monde extérieur.



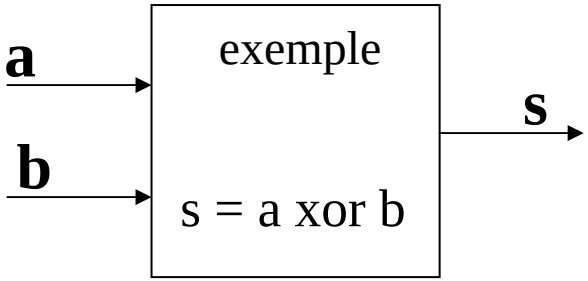
Nom du circuit

```
entity exemple is
port (
        a,b: in bit;
        s: out bit
        );
end exemple;
```

Ports d'accès et mode de fonctionnement

Nom du port d'accès : **a**,**b** *ou* **s**
Mode de fonctionnement: **in** *ou* **out**

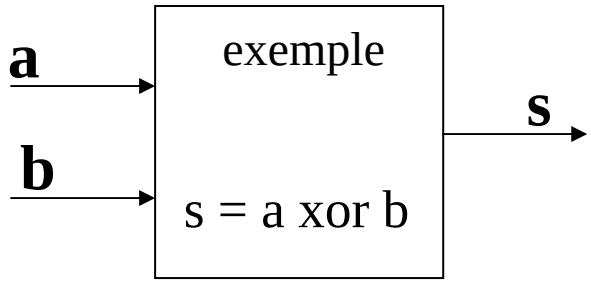
2 – L'architecture : L'architecture décrit l'intérieur de la boîte noire.




```
architecture simple of exemple is
begin
s <= a xor b;
end simple;
```

2 – L'architecture

Autres descriptions possibles :



- **architecture** simple **of** exemple **is**
  begin
  S <= '0' **when** a = b **else** '1';
  end simple;

- **architecture** simple **of** exemple **is**
  begin
  with (a & b) select
  S <= '1' **when** ''01'' | ''10'',
  '0' **when others**;
  end simple;

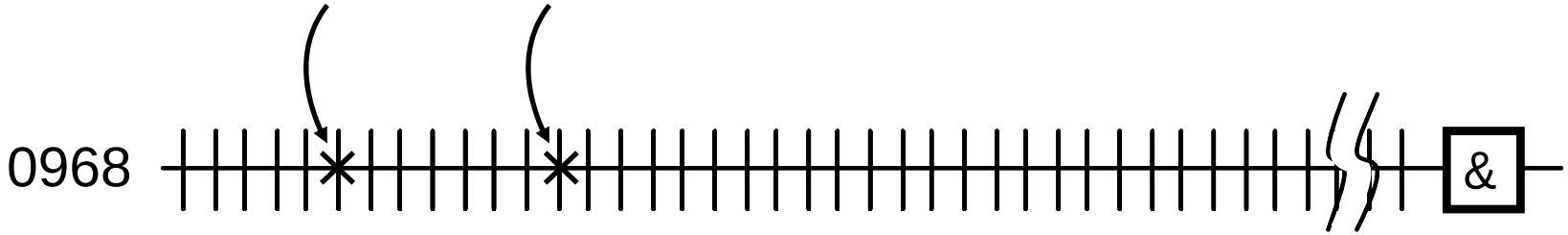
8. Le fichier JEDEC

- JEDEC : Joint Electronic Device Engineering Council
- Le fichier file.JED est celui à télécharger dans le composant physique.
- Le transfert s’effectue avec un programmateur de PAL, par exemple ALL11. Cet appareil est à relier au port série d’un PC et nécessite une application spécifique (Access.exe).

- La programmation peut se faire « In Situ », c-a-d que le composant n’a pas besoin d ’être programmé sur un programmateur avant d’être soudé sur le circuit imprimé. C’est ce que nous ferons avec le logiciel ispVM.

8. Le fichier JEDEC

```
L0440 11111111111111111111111111111111111111111111*
L0484 11111011111111111111111111111111111111111111*
L0924 11111111111111111111111111111111111111111111*
L0968 11111011111110111111111111111111111111111111*
```



1 : le fusible est détruit

0 : le fusible est conservé

9. Exemple de fichiers .ABL et .JED associés

Fichier ABEL :

```
module
fsimV2

        FSIM device 'P16V8R';
        TOUT0,WAY,TMROUT0,nstp_on PIN
1,2,3,4;
        NOCON,RAZ,ENB PIN 5,9,11;
        DECON,CKOUT,PHA,PHB PIN
15,16,17,18;
        PHA,PHB istype 'reg_d';
        PHA,PHB istype 'invert';
EQUATIONS
DECON=!(NOCON);
CKOUT=(TMROUT0 & !nstp_on);
PHA.D=((WAY & !PHB.Q # !WAY & PHB.Q) #
RAZ);
PHA.C=TOUT0;
!PHA.OE=ENB;
PHB.D=((WAY & PHA.Q # !WAY & !PHA.Q) #
RAZ);
PHB.C=TOUT0;
!PHB.OE=ENB;
END
```

Fichier JEDEC :

```
EZ-ABEL 4.32 Data I/O Corp. JEDEC file for:
P16V8R V9.0
Created on: Sun Apr 14 20:25:58 1996
*
QP20* QF2194* QV0* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS TOUT0:1 WAY:2 TMROUT0:3 nstp_on:4
NOCON:5 RAZ:9 ENB:11 DECON:15*
NOTE PINS CKOUT:16 PHA:17 PHB:18*
L0256 10111111111011111111111111111011*
L0288 01111111110111111111111111111011*
L0512 01111110111111111111111111111011*
L0544 10111101111111111111111111111011*
L0768 11111111111111111111111111111111*
L0800 11110111101111111111111111111111*
L1024 11111111111111111111111111111111*
L1056 11111111111101111111111111111111*
L2048 01110000*
L2120 00011000*
L2128
11111111111111111111111111111111111111111111
11111111111111111111*
L2193 1*
C268C*
9B01
```

10. Exercices - Exercice 1

PALCE 16V8 Lattice, c.f. photocopie:

On établit des jonctions aux adresses suivantes:
256, 260, 264, 288, 300, 321, 324, 336, 353, 357 et 372. Quelle est la fonction réalisée ?

En déduire AC0, AC1(?) et XOR(?).

10. Exercices - Exercice 2

(PALCE 16V8 Lattice, c.f. photocopie)

Quelles sont les adresses des fusibles qui permettent de programmer /(17) = (15) $\oplus$ (6).

Quelles sont les valeurs de AC0, AC1(?) et XOR(?) ?