LES CIRCUITS

LOGIQUES

PROGRAMMABLES

M. Philippe LETENNEUR –Lycée Julliot de la Morandière
GRANVILLE

SOMMAIRE

I. Présentation de la Logique Programmée.

Actuellement les Objets Techniques (O.T.) utilisent de plus en plus la logique programmée (µP, Mémoires, µC, ...). Ces structures ont besoin de s'interfacer entre elles. Elles utilisent généralement pour réaliser ces interfaces des fonctions à base de fonctions logiques élémentaires, compteurs, registres , ... Le nombre de circuits nécessaires pour remplir ces fonctions peut devenir très vite important.

Pour diminuer les coûts de fabrication, de développement et de maintenance, les fabricants de circuits intégrés ont donné naissance aux Circuits Logique Programmable ou encore *P.L.D.* (*Programmable Logic Device*).

Ces circuits sont capables pour un O.T. de réaliser plusieurs fonctions logiques dans un seul circuit. Si ces fonctions étaient réalisées à base circuits de logique classique, il en faudrait plusieurs circuits.

Un autre avantage, l'évolution des fonctions d'un l'O.T. s'effectue par programmation comparée à une solution classique où il faut refaire un circuit imprimé si on veut modifier le fonctionnement.

II La classification des P.L.D.

II.1 Liste de toutes les familles de *P.L.D.*

Les constructeurs de *P.L.D.* se livrent entre eux à une guerre commerciale et sur les appellations, ce qui explique la difficulté à établir la classification des *P.L.D.*

La plus ancienne et la plus connue est certainement la famille des *P.A.L.* Le nom a été donné par la société M.M.I (Fusion depuis avec A.M.D.), c'est une appellation déposée, comme Walkman pour Sony ou Réfrigérateur pour Frigidaire.

*** *P.A.L.* signifie *Programmable Array Logic*, c'est à dire réseau logique programmable. La programmation de ces circuits s'effectue par destruction de fusibles. Une fois programmés on ne peut plus les effacer. On distingue deux sous familles:**

- **Les *P.A.L.* combinatoires ou *P.A.L.* simples. Ils sont constitués de fonctions de logique combinatoire.**
- **Les *P.A.L.* à registres ou *F.P.L.S. Field Programmable Logic Séquencer* pour séquenceur logique programmable. Ils sont constitués de logique combinatoire et séquentielle (Registre).**

*** Les *P.A.L.* effaçables: *E.P.L.D.***

Les *E.P.L.D.* Ce qui signifie *Erasable Programmable Logic Device*, c'est à dire circuit logique programmable et effasçable et qui sont aux *P.A.L.* ce que sont les *U.V.P.R.OM.* aux *P.R.O.M.* Les *E.P.L.D.* peuvent être effacés par U.V. ou électriquement. Ils sont encore appelés *P.A.L. CMOS*.

*** Les *G.A.L.***

Les *G.A.L.* Ce qui signifie *Generic Array Logic* ou encore réseau logique générique ce qui veut dire pas grand chose mais qui sont aux *P.A.L.* ce que sont les *E.E.P.R.O.M* aux *P.R.O.M.* Le nom de *G.A.L.* a été déposé par LATTICE SEMICONDUCTOR. Leur fonctionnement est identique aux *P.A.L. CMOS*, ils sont programmables et effaçables électriquement.

*** Les *C.P.L.D.***

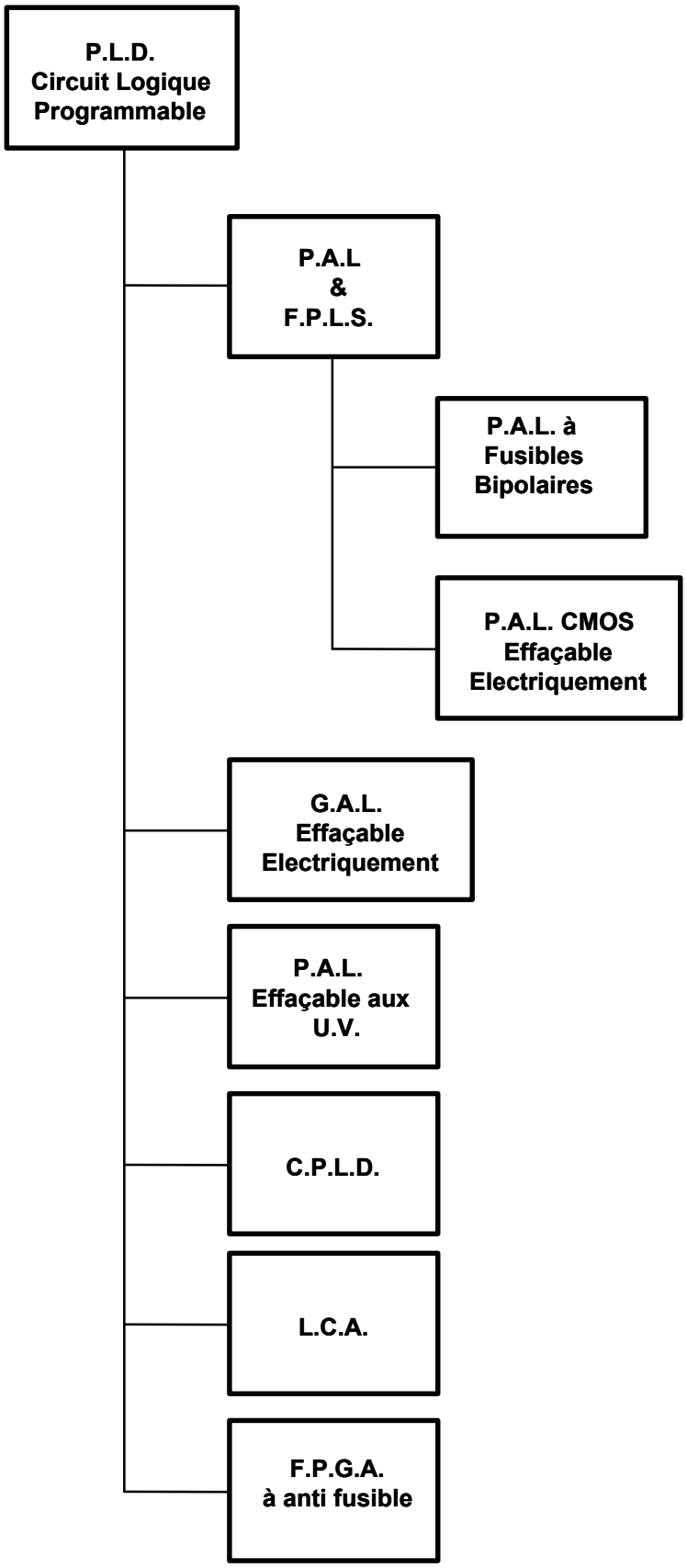
Les *C.P.L.D.* Ce qui signifie *Complex Programmable Logic Device*. Ces circuits sont composés de plusieurs P.A.L.s élémentaires (Par exemple l'équivalent de P.A.L.s 22V10) reliés entre-eux par une zone d'interconnexion. Grâce à cette architecture, ils permettent d'atteindre des vitesses de fonctionnement élevées (plusieurs centaine de Mhz).

*** Les *L.C.A. & F.P.G.A.* à anti-fusible.**

- Les *L.C.A.* Ce qui signifie *Logic Cell Array* ou encore réseau de cellules logiques. Ces circuits sont composés de blocs logiques élémentaires de 2000 à 10000 portes que l'utilisateur peut interconnecter.

- Les *F.P.G.A.* à anti fusibles sont identiques aux *L.C.A* sauf qu'ils permettent une plus grande intégration de portes et ils ne sont pas effaçables électriquement. Le nom anti-fusible vient de la programmation des connexions qui s'effectue par fermeture de circuits, comparé aux fusibles où l'on ouvre les circuits.

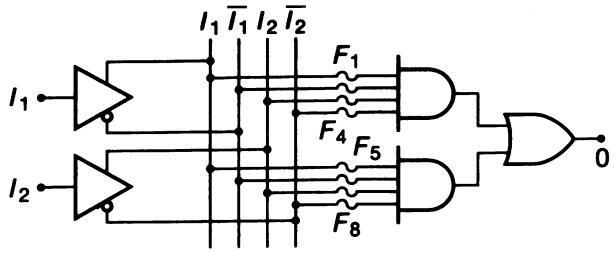
II.2 Résumé graphique des familles de *P.L.D.*



III Les *P.A.L.* (Programmable Array Logic)

L'invention des *P.A.L.* date d'une vingtaine d'années, ce sont les ingénieurs de chez M.M.I qui ont eu l'idée d'utiliser la technologie des fusibles.

La programmation s'effectue par destruction de fusible (un fusible détruit équivaut à un circuit ouvert), voir schéma ci-dessous.



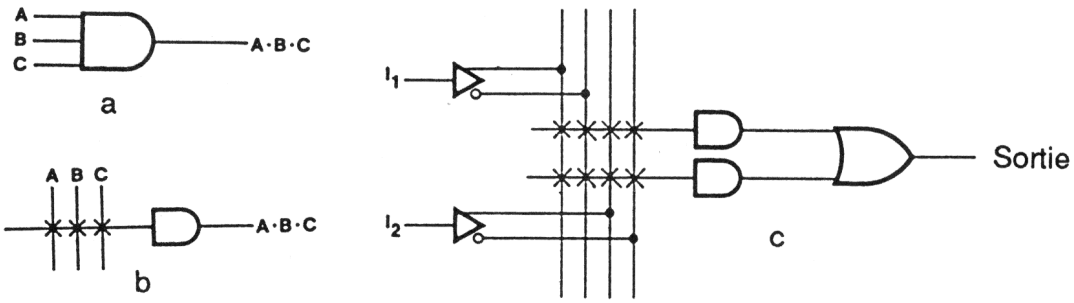
Structure simplifiée d'un PAL
(2 entrées et 1 sortie)

***I1* et *I2* représentent des entrées (I:Input).**
***O* représente une sortie (O:Output).**

A partir de cette structure de base il va être possible de réaliser de nombreuses fonctions logiques. La programmation va constituer à détruire les fusibles pour obtenir les fonctions désirées, en sachant que lors de l'achat d'un *P.A.L.* tous les fusibles sont vierges ou pas détruits.

III.1 La symbolisation et la représentation.

La représentation schématique de la précédente structure demande beaucoup d'espace pour représenter un *P.A.L.* en entier. Les industriels ont adopté une autre représentation voir ci-dessous.

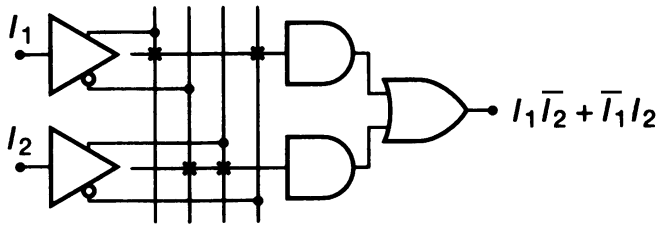


***a*: Porte ET à 3 entrées.**
***b*: Porte ET à 3 entrées représentation *P.A.L.* les croix représentent les fusibles intacts.**
***c*: Représentation de la structure interne d'un *P.A.L.*.**

Exemple: **Représentation d'un Ou Exclusif.**

$I1 \oplus I2 = \overline{I1} \cdot I2 + I1 \cdot \overline{I2}$

Cette équation se représente de la façon suivante:



III.2 Les différentes structures.

III.2.1 Structure générale.

Tout *P.A.L.* est constitué :

- D'entrées (Input): I1 à In avec 8<n<20.

- De sorties (Output) Ou d'entrées / sorties (I/O) de type Totem Pôle ou Trois Etats : O1 à On ou IO1 à IOn (2<n<15).

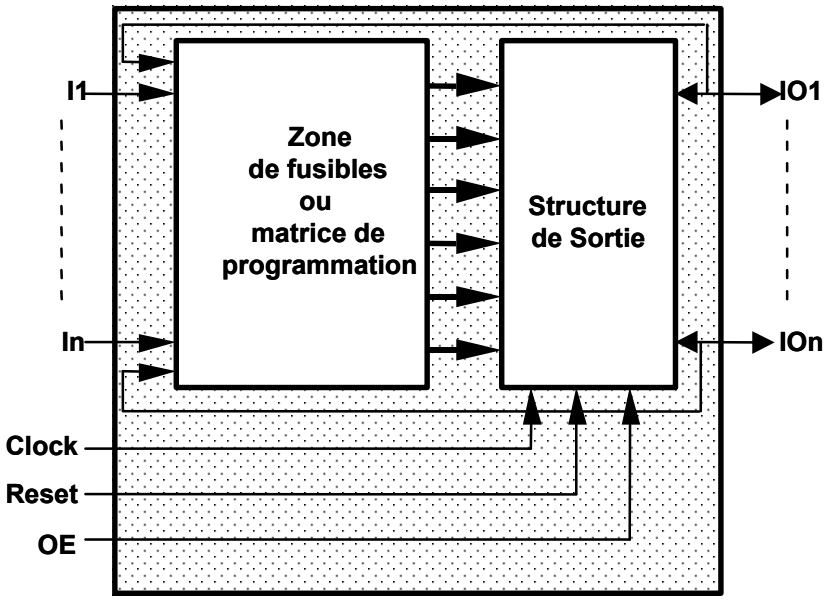
On peut trouver aussi:

- Une entrée d'horloge (Clock): Clk ou Clock.

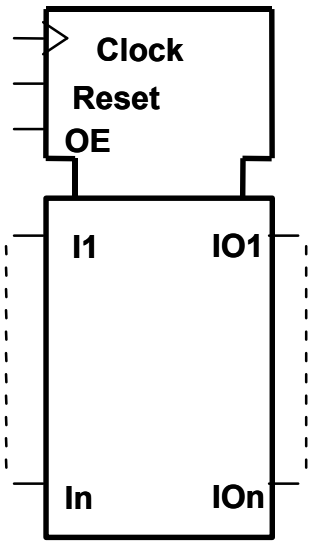
- Une entrée de validation des sorties trois états: OE (Output Enable) ou Enable.

- Une entrée de remise à zéro des registres: RESET.

D'un point de vue fonctionnel un *P.A.L.* est constitué d'une *zone d'entrée de fusibles ou matrice de programmation* et une *structure de sortie non programmable déterminant le type de circuit* voir schéma ci-dessous.



Symbolisation normalisée :



<u>Remarque:</u> Sur un schéma comportant un *P.A.L.*, on doit écrire les équations qui relient les entrées aux sorties ou le nom du document contenant les équations du *P.A.L.*

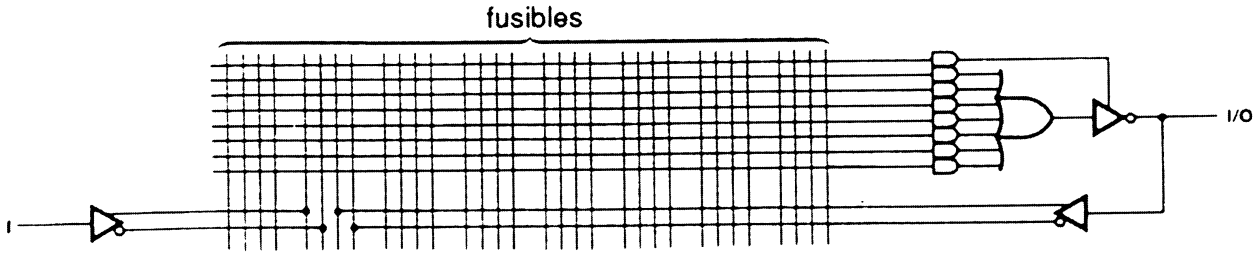
Dans l'exemple au paragraphe III.1, la programmation du Ou Exclusif était facilement réalisable, ce n'est pas toujours le cas. C'est pourquoi il existe un grand nombre de *P.A.L.* utilisant des structures de sorties différentes. On peut distinguer trois types de structures de base:

- **Combinatoire.**
- **Séquentielle.**
- **Versatile.**

III.2.2 Combinatoire.

Il existe trois types:

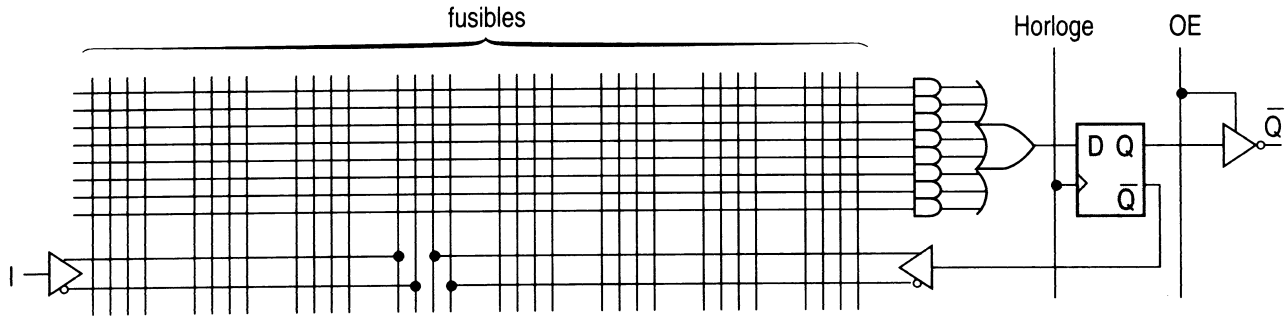
- **H -> (High) Porte ET suivit d'une Porte OU. Sortie active à l'état haut.**
- **L -> (Low) Porte ET suivit d'une Porte NON OU. Sortie active à l'état bas.**
- **C -> (Combinée) programmable en type H ou L.**



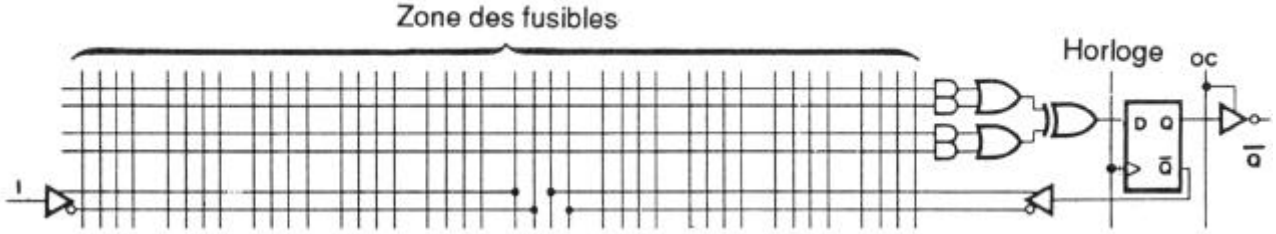
III.2.3 Séquentielle.

Il existe trois types:

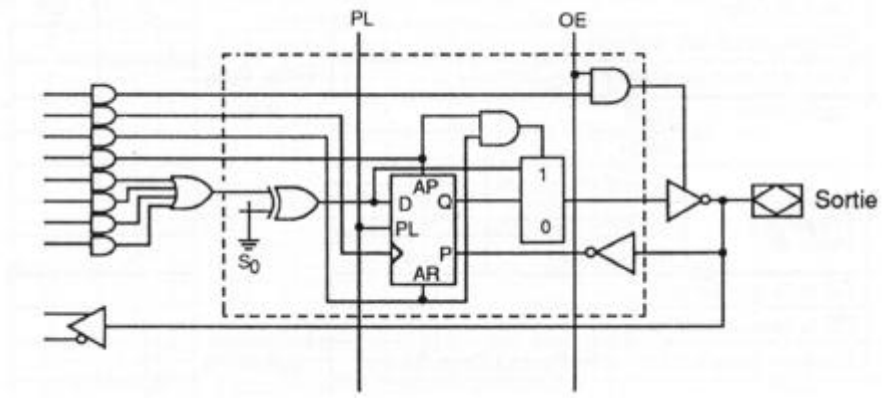
- R -> (Register): Registre. Ces circuits sont composés de bascule D. Les sorties des bascules sont de type trois états contrôlées par un signal de validation Enable ou OE, et une horloge est commune à toutes les bascules (clock).



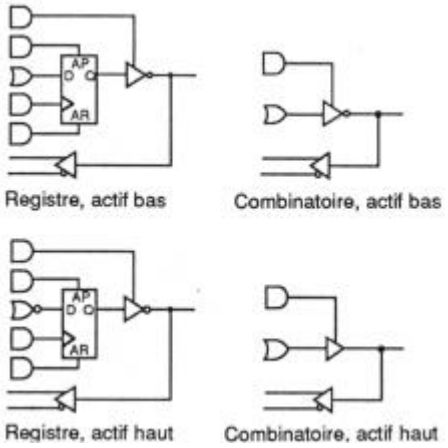
- X -> (Register Or Exclusif): Ou Exclusif et Registre.



- RA -> (Register Asynchron): Registre asynchrone.

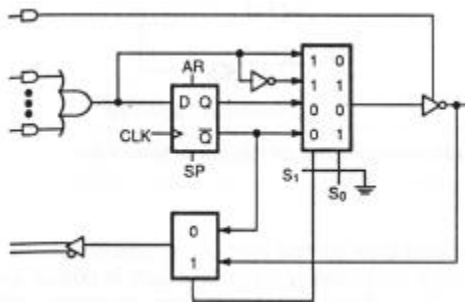


Les structures de sorties sont beaucoup plus évoluées par rapport aux autres *P.A.L.*, elles se rapprochent des *P.A.L.* de type versatile.
Elles peuvent prendre quatre configurations suivant les valeurs de AP et AR.



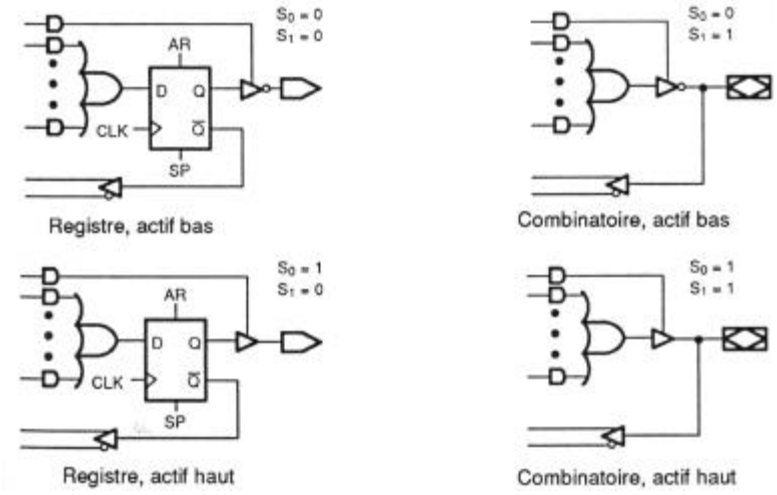
III.2.4 Les versatiles.

Ce type de structure représente les *P.A.L.* les plus évoluées, car les structures de sorties dite versatile proposent quatre configurations possibles. suivant les valeurs de S0 et S1.



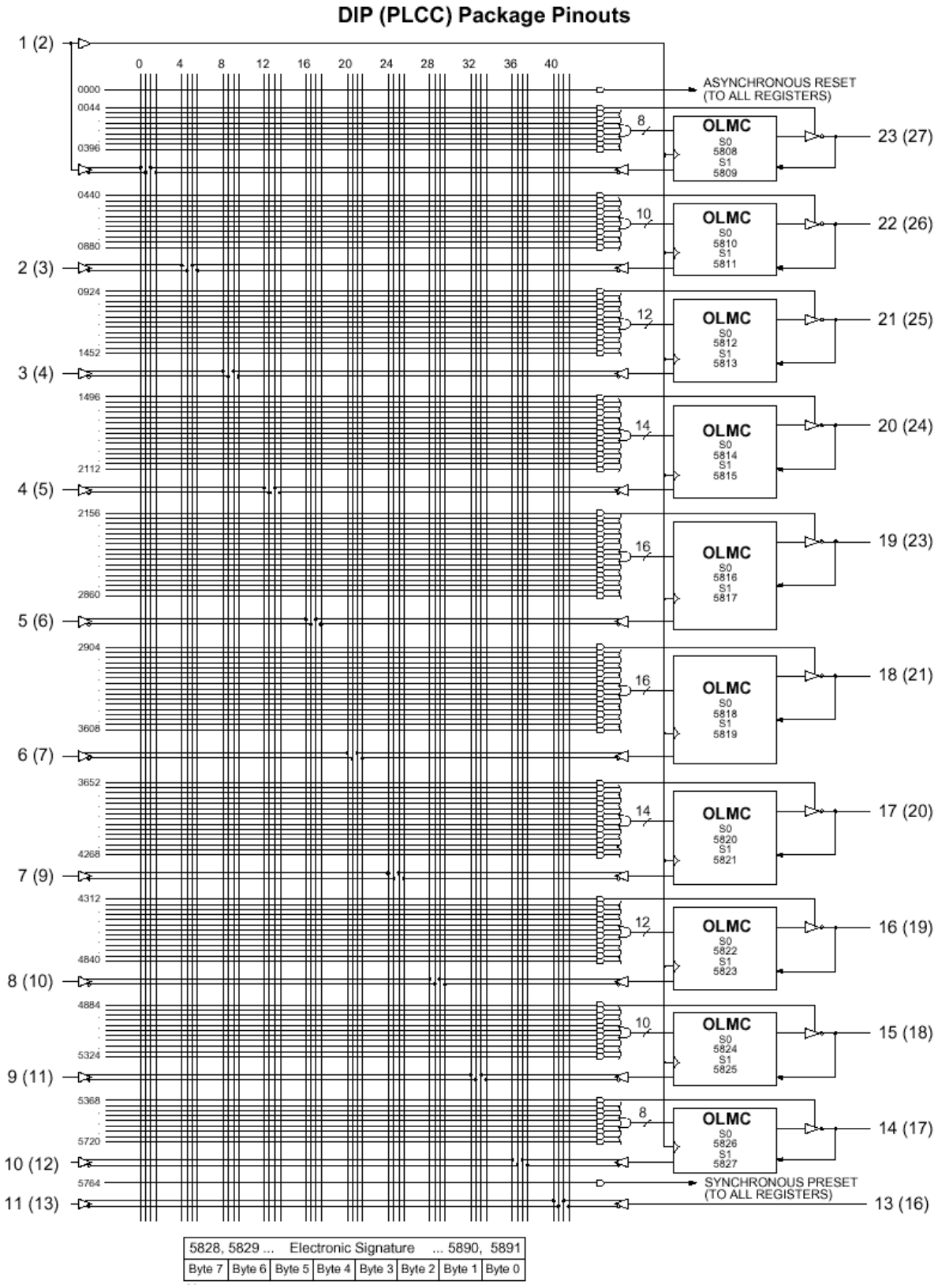
S_1	S_0	Configuration des sorties
0	0	Registre, actif bas
0	1	Registre, actif haut
1	0	Combinatoire, actif bas
1	1	Combinatoire, actif haut

Ce qui donne:



Le plus célèbre des circuits versatile est certainement le *P.A.L.* 22V10 de chez AMD. Il permet d'émuler pratiquement tout les autres types de *P.A.L.* et dispose d'un circuit de remise à zéro des registres à la mise sous tension du circuit.

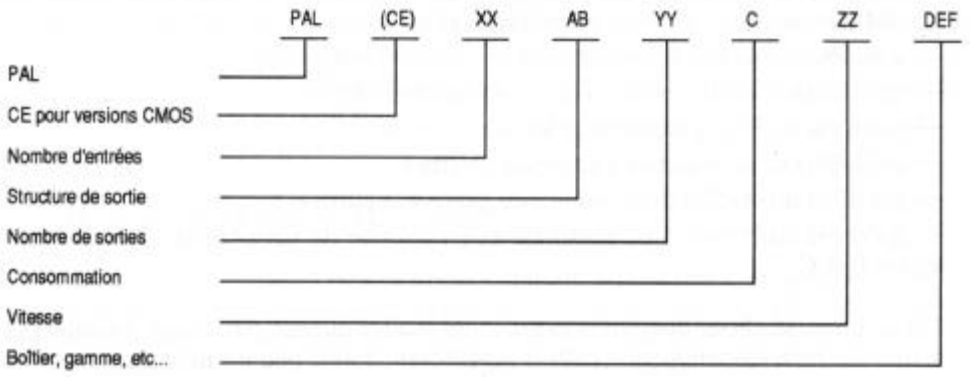
Schéma interne d'un PAL 22V10 : (GAL22V10 Lattice Semiconductor).



III.3 Les références des *P.A.L.*

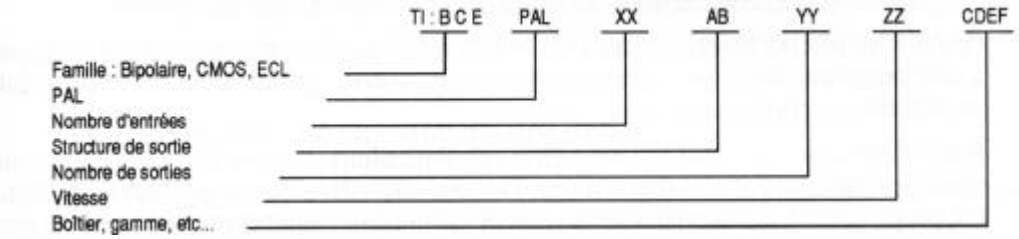
Les constructeurs de *P.A.L.* ont standardisé le marquage des *P.A.L.*

Chez AMD:



Lettre(s) code(s)	Structure de sortie
L	Combinatoire active bas
H	Combinatoire active haut
R	Registre
RA	Registre asynchrone
X	Registre et OU exclusif
V	Versatile

Chez TEXAS INSTRUMENTS:



Exemples:

Un PAL 16L8 : 16 entrées maximum, 8 sorties maximum et il dispose d'une structure de sortie active sur niveau bas.

PAL	Entrées max Input (I)	Sorties max Output (O)	Entrées / Sorties (I/O)	Registres	Fonction
10H8	10	8			AND OR
12H6	12	6			AND OR
14H4	14	4			AND OR
16H2	16	2			AND OR
10L8	10	8			AND OR INVERT
12L6	12	6			AND OR INVERT
14L4	14	4			AND OR INVERT
16L2	16	2			AND OR INVERT
16C1	16	1			AND-OR /AND OR-INVERT
16L8	10	8	6		AND OR INVERT
16R8	8	8		8	AND OR INVERT-REGISTER
16R6	8	8	2	6	AND OR INVERT-REGISTER
16R4	8	8	4	4	AND OR INVERT-REGISTER
16X4	8	8	4	4	AND OR INVERT-XOR-REGISTER
16A4	8	8	4	4	AND CARRY-OR-XOR-INVERT-REGISTER

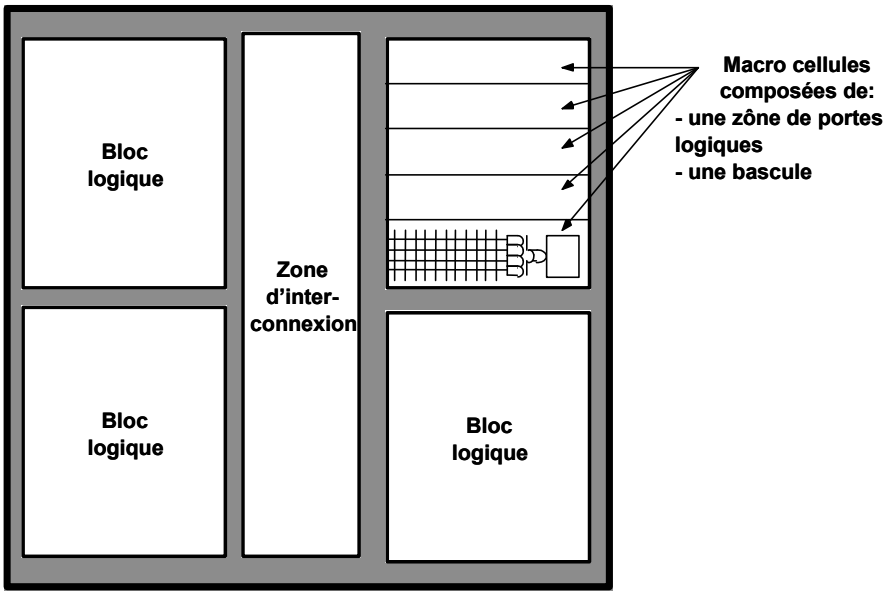
III.4 La duplication de *P.A.L.* et le bit de sécurité.

Un circuit *P.A.L.* peut être dupliqué comme une EPROM, pour le protéger les constructeurs ont ajouté un bit dit de sécurité. Si ce bit est programmé alors le circuit ne peut plus être relu.

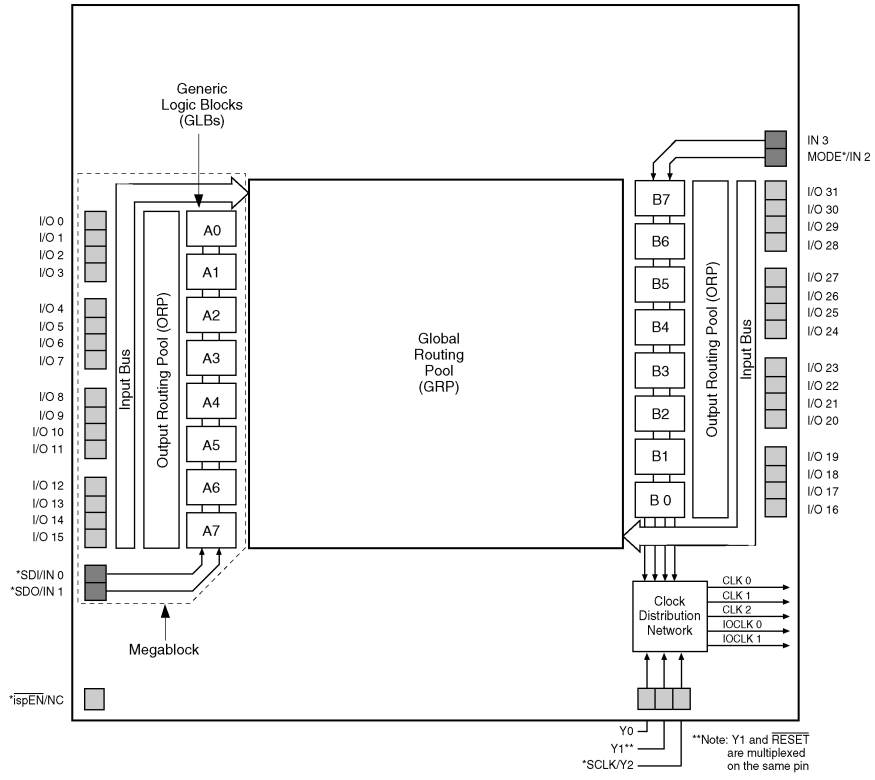
IV LES C.P.L.D.s (Complex Programmable Logic Device).

Ces circuits ont une capacité en nombre de portes et en possibilités de configuration très supérieure à celle des PALs. Leurs architectures sont basées sur celles des PALS. Un CPLD c'est l'équivalent de plusieurs PALs mis dans le même circuit associé à une zone d'interconnexion. Le nombres de portes peut varier entre 100 et 100 000 portes logiques et entre 16 et 1000 bascules voir plus.

Structure générale d'un CPLD.



Exemple de CPLD le circuit lsp1016 de LATTICE :

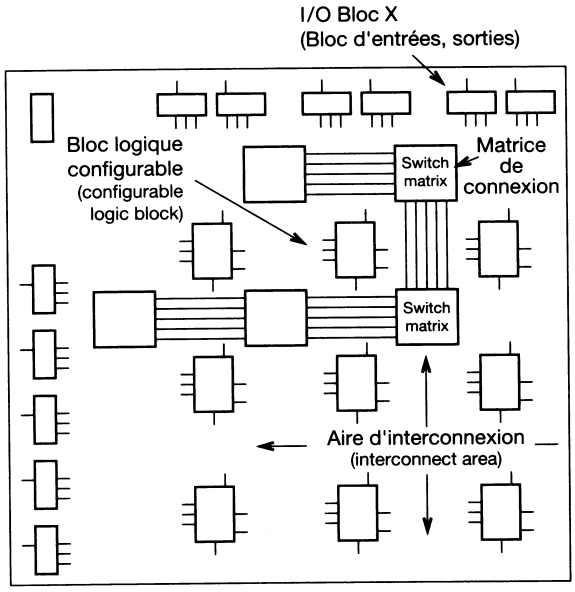


V LES F.P.G.A.s (Fields Programmable Gate Array).

les FPGAs à la différence des CPLDs sont assimilables à des A.S.I.C. (Application Specific Integrated Circuit) programmables par l'utilisateur. La puissance de ces circuits est telle qu'ils peuvent être composés de plusieurs milliers voire millions de portes logiques et de bascules. Les dernières générations de FPGA intègrent même de la mémoire vive (RAM). Les deux plus grands constructeurs de FPGA sont XILINX et ALTERA.

Ils sont composés de blocs logiques élémentaires (plusieurs milliers de portes) qui peuvent être interconnectés.

STRUCTURE INTERNE D'UN FPGA (TYPE XILINX)



De plus en plus les capacités des CPLDs et des FPGAs se rapprochent. Le principal critère de choix entre les deux familles est la vitesse de fonctionnement. En effet les CPLDs acceptent des fréquences de fonctionnement beaucoup plus élevées que les FPGAs.

VI Les outils de développements.

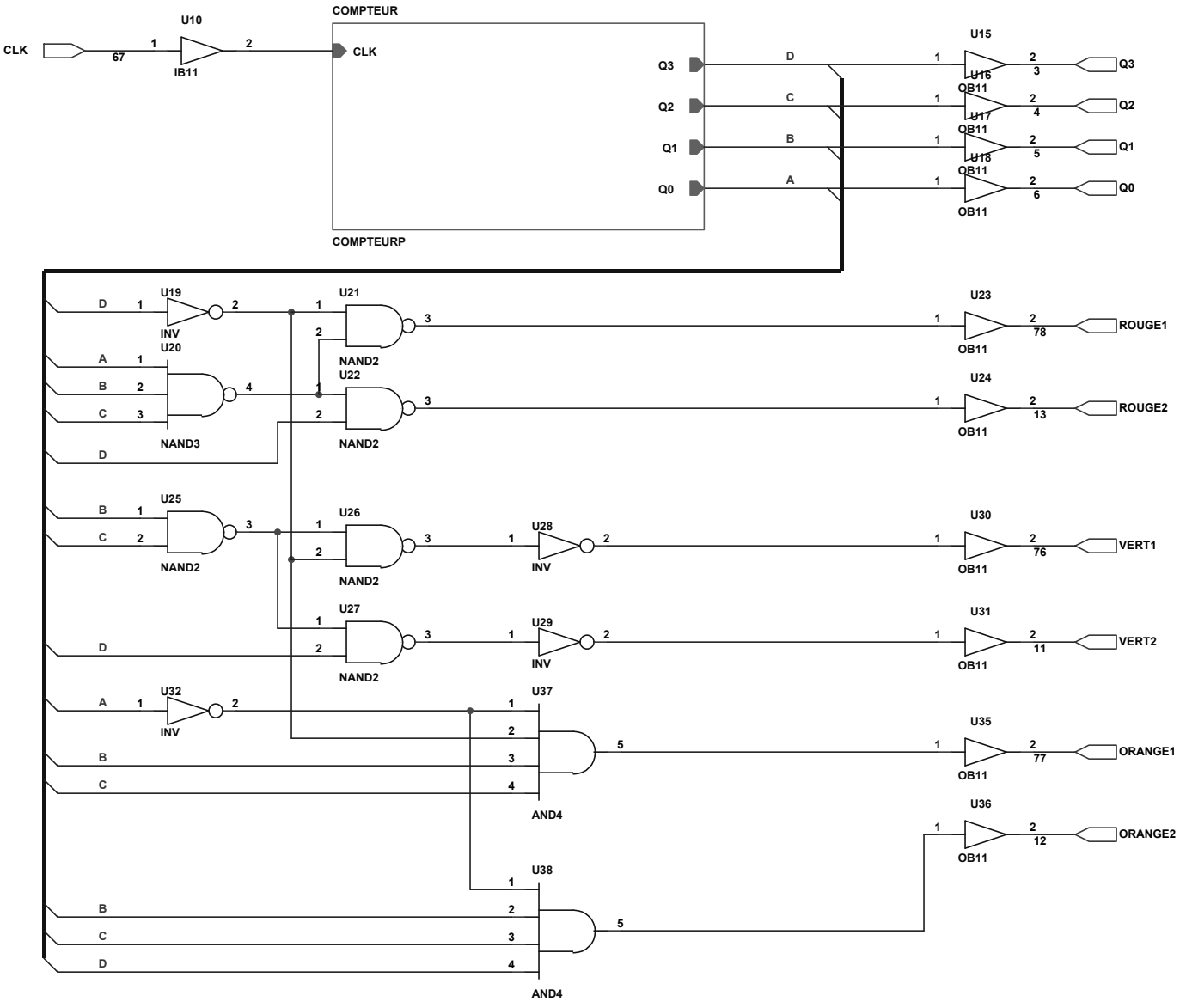
Ils sont composés en général de deux outils:
- Le système de développement.
- Le programmateur.

VI.1 Le système de développement.

Ces systèmes produisent une table représentant les fusibles à détruire en fonction des équations logiques, diagramme d'états et tables de vérités écrit dans le langage propre au système, c'est le rôle du compilateur ou synthétiseur.

La description du fonctionnement des circuits peut se faire de plusieurs façons, soit :

- **Par un schéma à base de fonctions logique élémentaires (Portes ET,OU,NON, … bascules, compteurs, registres à décalages).**



- En utilisant un langage de description comportementale *H.D.L.* (*H*ardware *D*escription *L*anguage). Les plus anciens sont *PALASM*, *ORCAD/PLD* et le plus connu et utilisé est sans conteste *ABEL* (utilisé par la plus part des systèmes de développements). Enfin les langages dit de haut niveau, *VHDL* (*V*ery high speed *H*ardware *D*escription *L*anguage) et VERILOG sont en général utilisés pour des circuits complexes. Le langage VHDL est très utilisé en Europe.

ORCAD/PLD :

```
|GAL16V8 in:(A15,A14,A13,A11,A10),
|        io:(RAM0,RAM1,RAM2,RAM3,ROM,INTER1,INTER2,INTER3)
|
| RAM0 = (A15' & A11' & A10')'
| RAM1 = (A15' & A11' & A10)'
| RAM2 = (A15' & A11 & A10')'
| RAM3 = (A15' & A11 & A10)'
| ROM = (A15 & A14 & A13)'
| INTER1 = (A15 & A14' & A13')'
| INTER2 = (A15 & A14 & A13)'
| INTER3 = (A15 & A14 & A13')'
```

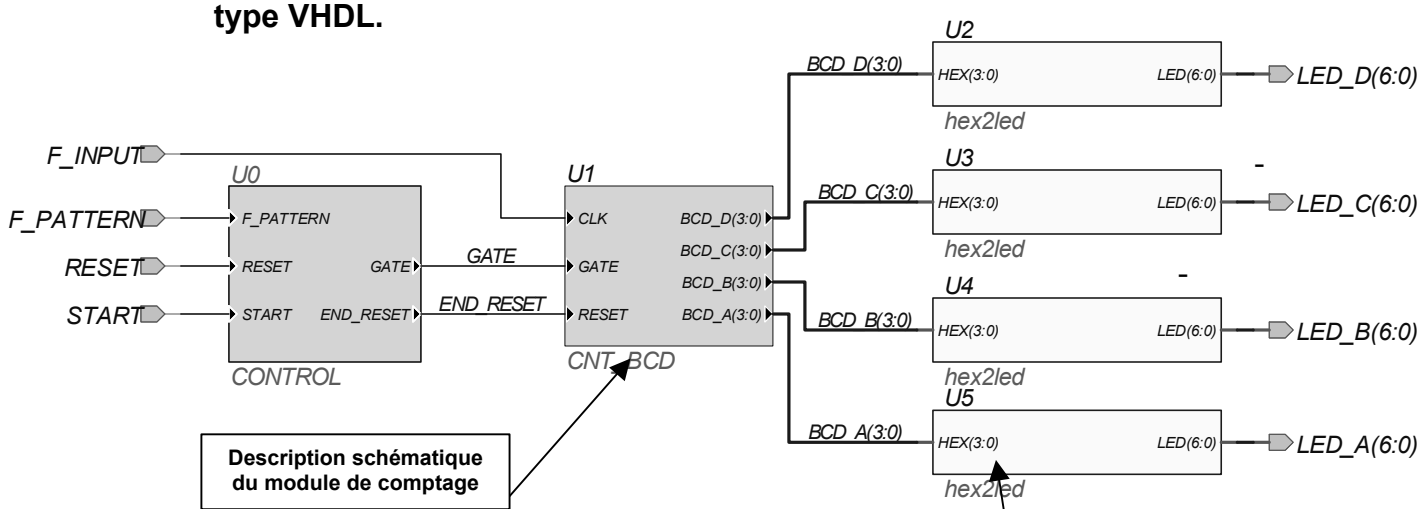
ABEL :

```
module tp1
title 'decodage d adresse'

declarations
        A15,A14,A13,A11,A10 pin;
        RAM0, RAM1, RAM2, RAM3, ROM, INTER1, INTER2, INTER3 pin istype 'com';
        x =.X.;
        adresse = [A15,A14,A13,x, A11,A10,x,x, x,x,x,x, x,x,x,x];

equations
        !RAM0   = (adresse >=^h0000) & (adresse <=^h03FF);
        !RAM1   = (adresse >=^h0400) & (adresse <=^h07FF);
        !RAM2   = (adresse >=^h0800) & (adresse <=^h0BFF);
        !RAM3   = (adresse >=^h0C00) & (adresse <=^h0FFF);
        !ROM    = (adresse >=^hE000) & (adresse <=^hFFFF);
        !INTER1 = (adresse >=^h8000) & (adresse <=^h8001);
        !INTER2 = (adresse >=^hA000) & (adresse <=^hA001);
        !INTER3 = (adresse >=^hC000) & (adresse <=^hC00F);
end;
```

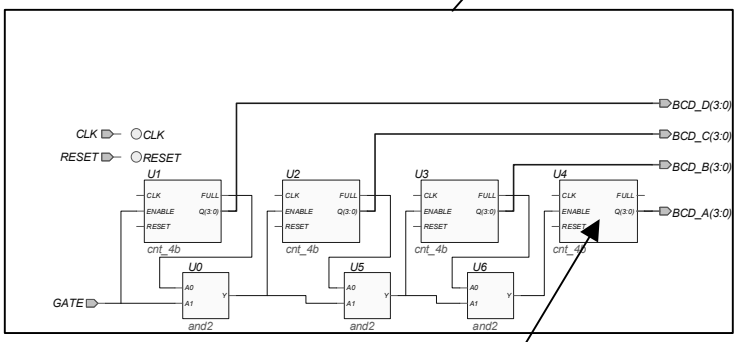
VHDL :

```
-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;

ENTITY DECODAGE is
        PORT (
                A15, A14, A13, A12, A11, A10      : IN STD_LOGIC;
                RAM0                              : OUT STD_LOGIC;
                RAM1                              : OUT STD_LOGIC;
                RAM2                              : OUT STD_LOGIC;
                RAM3                              : OUT STD_LOGIC;
                ROM                               : OUT STD_LOGIC;
                INTER1                            : OUT STD_LOGIC;
                INTER2                            : OUT STD_LOGIC;
                INTER3                            : OUT STD_LOGIC);
END DECODAGE;
ARCHITECTURE behavior OF DECODAGE IS
SIGNAL ADRESSE: STD_LOGIC_VECTOR(15 downto 0);
BEGIN
        ADRESSE <= A15 & A14 & A13 & A12 & A11 & A10 & "----------";
        ROM    <= '0' when (ADRESSE >= x"E000") and (ADRESSE <= x"FFFF") else '1';
        RAM0   <= '0' when (ADRESSE >= x"0000") and (ADRESSE <= x"03FF") else '1';
        RAM1   <= '0' when (ADRESSE >= x"0400") and (ADRESSE <= x"07FF") else '1';
        RAM2   <= '0' when (ADRESSE >= x"0800") and (ADRESSE <= x"0CFF") else '1';
        RAM3   <= '0' when (ADRESSE >= x"0D00") and (ADRESSE <= x"0FFF") else '1';
        INTER1 <= '0' when (ADRESSE >= x"8000") and (ADRESSE <= x"8001") else '1';
        INTER2 <= '0' when (ADRESSE >= x"A000") and (ADRESSE <= x"A001") else '1';
        INTER3 <= '0' when (ADRESSE >= x"C000") and (ADRESSE <= x"C00F") else '1';
END behavior;
```

- **En utilisant un schéma et des descriptions en langage de haut niveau de type VHDL.**



Description schématique du module de comptage



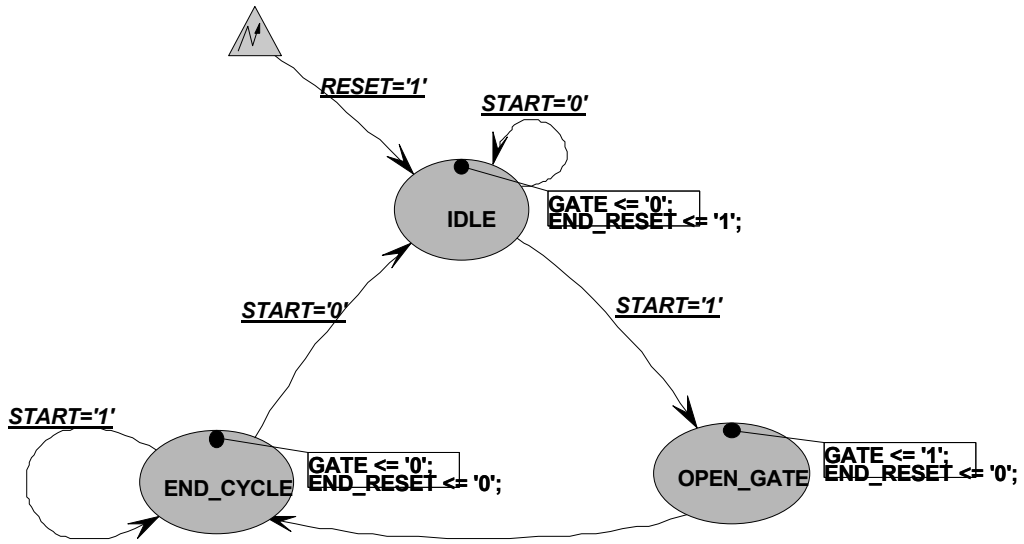
Description VHDL du module de comptage BCD

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity CNT_4B is
    port (
        CLK: in STD_LOGIC;
        RESET: in STD_LOGIC;
        ENABLE: in STD_LOGIC;
        FULL: out STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (3 downto 0)
    );
end CNT_4B;

architecture CNT_4B of CNT_4B is
        signal Qint: STD_LOGIC_VECTOR(3 downto 0);

begin
        process (CLK, RESET)
        begin
                if RESET = '1' then
                        Qint <= (others => '0');
                elsif CLK='1' and CLK'event then
                        if ENABLE = '1' then
                             if Qint = 9 then
                                Qint <= (others => '0');
                             else
                                Qint <= Qint + 1;
                             end if;
                        end if;
                end if;
        end process;
        Q <= Qint;
        FULL <= '1' when (Qint = 9) else '0';
end CNT_4B;
```

Description VHDL du module de décodage 7 segments

```
library IEEE;
use IEEE.std_logic_1164.all;

entity hex2led is
    port (
        HEX: in STD_LOGIC_VECTOR (3 downto 0);
        LED: out STD_LOGIC_VECTOR (6 downto 0)
    );
end hex2led;

--}} End of automatically maintained section

architecture hex2led of hex2led is
-- segment encoding
--      0
--     ---
--  5 |   | 1
--     ---   <- 6
--  4 |   | 2
--     ---
--      3
begin
        with HEX select
        LED     <=      "1111001" when "0001",    --1
                        "0100100" when "0010",    --2
                        "0110000" when "0011",    --3
                        "0011001" when "0100",    --4
                        "0010010" when "0101",    --5
                        "0000010" when "0110",    --6
                        "1111000" when "0111",    --7
                        "0000000" when "1000",    --8
                        "0010000" when "1001",    --9
                        "0001000" when "1010",    --A
                        "0000011" when "1011",    --b
                        "1000110" when "1100",    --C
                        "0100001" when "1101",    --d
                        "0000110" when "1110",    --E
                        "0001110" when "1111",    --F
                        "1000000" when others;    --0
end hex2led;
```

Par l'utilisation de graphes d'états. (FSM : Flow States Machines)



Quelques systèmes de développements connus :

1) Synario de LATTICE SEMI CONDUCTOR (Schéma, ABEL et VHDL)
WEB : http://www.latticesemi.com

2) Express d'ORCAD (Schémas, VHDL)
WEB : http://www.orcad.com/ pour la France WEB : http://www.alsdesign.fr/

3) ViewPLD de VIEW LOGIC (schémas, ABEL et VHDL)
WEB : http://www.viewlogic.com

4) StateCad de STATECAD (Graphes d'états).
WEB : http://www.statecad.com/

5) Active VHDL de ALDEC (Schémas, VHDL et Graphes d'états).
WEB : http://www.aldec.com/

6) Warp de CYPRESS (VHDL et graphes d'états).
WEB : http://www.cypress.com/

7) MaxPlus d'ALTERA (Schémas, VHDL et Graphes d'états).
WEB : http://www.altera.com/

8) Foundation d'XILINX (Schémas, VHDL et Graphes d'états).
WEB : http://www.xilinx.com/

VI.2 Description générale de la chaîne des outils utilisés pour mettre au point des circuits logiques programmables.

VI.2.1 Pour les PALs.

Ces outils peuvent posséder d'un simulateur logique pour vérifier la programmation du circuit.

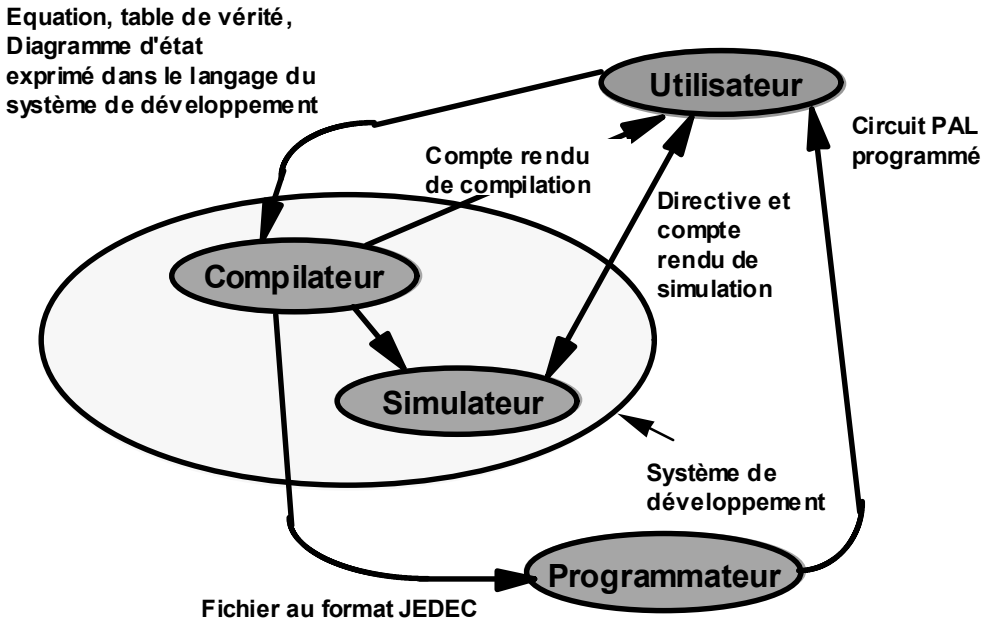
Le format de la table des fusibles à détruire a été normalisé par les constructeurs de circuits pour qu'il puisse être accepté par le *programmateur* de *P.A.L.*, c'est le format JEDEC (Format de fichier de programmation des circuits logiques : image des fusibles à griller).

VI.2.1.1 Le programmateur.

Il permet de:
- **Vérifier la virginité du circuit.**
- **Lire le fichier au format JEDEC.**
- **Programmer le circuit.**
- **Vérifier la programmation.**

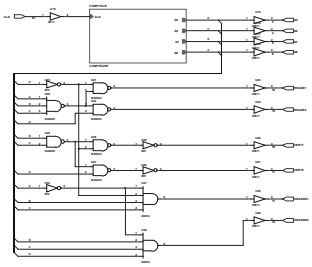
Il se présente souvent par une carte à insérer dans le PC reliée à un support de programmation et d'un logiciel permettant le dialogue entre lui et l'utilisateur.

VI.2.1.2 Schéma fonctionnel d'un outil de développement de PAL.

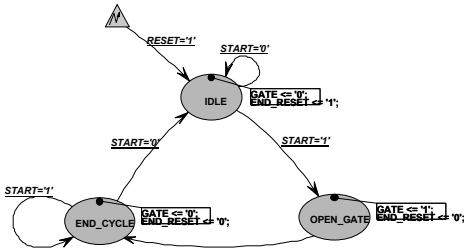


VI.2.2 Pour les CPLDs et FPGAs.

VI.2.2.1 Schéma fonctionnel d'un outil de développement de FPGA /CPLD.



```
library IEEE;
use IEEE.std_logic_1164.all;
use
IEEE.STD_LOGIC_UNSIGNED.all;

entity CNT_4B is
    port (
        CLK: in STD_LOGIC;
        RESET: in STD_LOGIC;
        ENABLE: in STD_LOGIC;

        FULL: out STD_LOGIC;
        Q: out STD_LOGIC_VECTOR
(3 downto 0)
```



Entrée schématique | **Entrée syntaxique langages: - VHDL** | **Diagramme d'états**

VHDL *VHDL* *VHDL*

Vérification des erreurs: de schéma et de syntaxe

STILMULI ou TESTBENCH ↔ **Simulation comportementale**

VHDL

SYNTHETISEUR

Il convertit en fonction du ciruit cible (CPLD ou FPGA: suivant le modèle choisit) le projet en portes logiques et bascules de bases.
De plus on peut lui spécifier des contraites de technologies, par exemple la vitesse de fonctionnement

VITAL, VHDL

STILMULI ou TESTBENCH ↔ **Simulation après synthèse**

SYNTHETISEURS CONNUS:

-FPGA Expres (Synosys).
-EXEMPLAR (Exemplar).
-SYNPLICITY (Synplicity).

EDIF, XNF

Outils de placement et routage interne au circuit. Cet outil est propre à chaque fabriquant

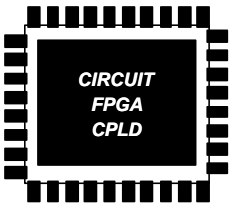
Optimisation Placement / routage dans le circuit choisit

VITAL, VHDL, SDF

STILMULI ou TESTBENCH ↔ **Simulation temporelle**

JEDEC

La programmation du circuit peut se faire soit:
- En utilisant un programmateur.
- En téléchragement le code JEDEC directement sur l'objet technique (JTAG ou ISP)



VII) Lexique

ASIC (Application Specific Integrated Circuit) : Circuit non programmable configuré lors de sa fabrication pour une application spécifique.

CPLD (Complex Programmable Logic Device) : Désigne des PLD ayant un haut niveau d'intégration.

EEPROM ou E2PROM (Electrical Erasable Programmable Read-Only Memory) : Mémoire programmable à lecture seule, effaçable électriquement.

EPLD (Erasable Programmable Logic Device) : Circuits logiques reprogrammables.

EPROM (Erasable Programmable Read-Only Memory) : Mémoire programmable à lecture seule, effaçable par ultraviolets.

FPGA (Forecasting Programmable Gate Array) : Réseau de portes programmables à la demande. Technologie qui utilise des circuits encapsulés comportant des réseaux de portes logiques non reliées : l'utilisateur réalise les interconnexions nécessaires par programmation.

FPLS (Field Programmable Logic Sequencer) : Ancien nom donné aux PAL à registres.

GAL (Generic Array Logic) : Circuits logiques PAL reprogrammables à technologie CMOS.

ISP (In System Programmable) : Circuit que l'on peut programmer (et donc effacer) même lorsqu'il est en place sur l'application.

JEDEC : Format de fichier de programmation des circuits logiques (image des fusibles à griller).

LSI (Large Scale Integration) : Intégration à grande échelle : circuits regroupant quelques centaines à quelques milliers de portes logiques (CI de télécommande, décodeur de code à barre, etc …).

MSI (Medium Scale Integration) : Intégration à échelle moyenne : circuits regroupant quelques dizaines de portes logiques (décodeurs, multiplexeurs, bascules …).

PAL (Programmable Array Logic) : Circuits logiques programmables dans lesquels seules les fonctions ET sont programmables, les fonctions OU ne le sont pas.

PAL CMOS ou PAL EECMOS : *c.f. GAL.*

PLD (Programmable Logic Device) : Famille des circuits programmables qui comprend les PAL, GAL, EPLD et FPGA.

SSI (Small Scale Integration) : Intégration à petite échelle : circuit ne regroupant que quelques portes logiques (fonctions de base des séries 74 ou 4000).

VHDL : Langage de programmation utilisé pour programmer les PLD.

VLSI (Very Large Scale Integration) : Intégration à très grande échelle : circuits regroupant quelques dizaines de milliers de portes logiques (microprocesseurs …).

VIII) BIBLIOGRAPHIE

- **Circuits logiques programmables (*Christian Tavernier - DUNOD, Paris, 1996*)**
- **Guide du technicien en électronique (*C. Cimelli, R. Bourgeron - HACHETTE, Paris, 1995*)**
- **Electronique Radio-Plans (*n° 567*).**
- **Elektor (*n° 197, novembre 1994*).**
- **Lexique électronique (*P. Roussel - NATHAN, 1996*).**

IX) ILLUSTRATIONS .

- **Circuits logiques programmables (*Christian Tavernier - DUNOD, Paris, 1996*)**
- **Guide du technicien en électronique (*C. Cimelli, R. Bourgeron - HACHETTE, Paris, 1995*)**
- **Document réalisés par L.P. AMPERE - 13010 MARSEILLE**