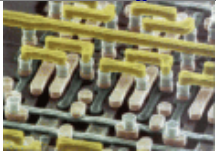




# IMPLEMENTACIÓ DE SISTEMES AMB HARDWARE I SOFTWARE

---

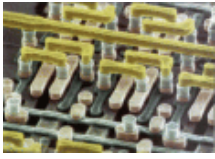
- **Resultat de la implementació de sistemes**
  - **Software:** Flexibilitat
  - **Hardware:** Rapidesa
- **NECESSITAT: Hardware més flexible**
  - Programació / Reprogramació
  - Adaptació i/o Millora reutilitzant
  - Correcció d'errors (*BUGs*)
- **SOLUCIÓ**
  - PLDs (**P**rogrammable **L**ogic **D**evice)



## PLD – CLASSIFICACIÓ

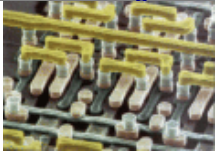
---

- Una **possible** classificació dels **PDLs** pot ser:
  - PLD (**P**rogrammable **L**ogic **D**evice)
    - SPLD (**S**imple **PLD**)
      - PROM, EPROM, E<sup>2</sup>PROM, PLA, PAL, GAL, ...
    - CPLD (**C**omplex **PLD**)
      - FPGA

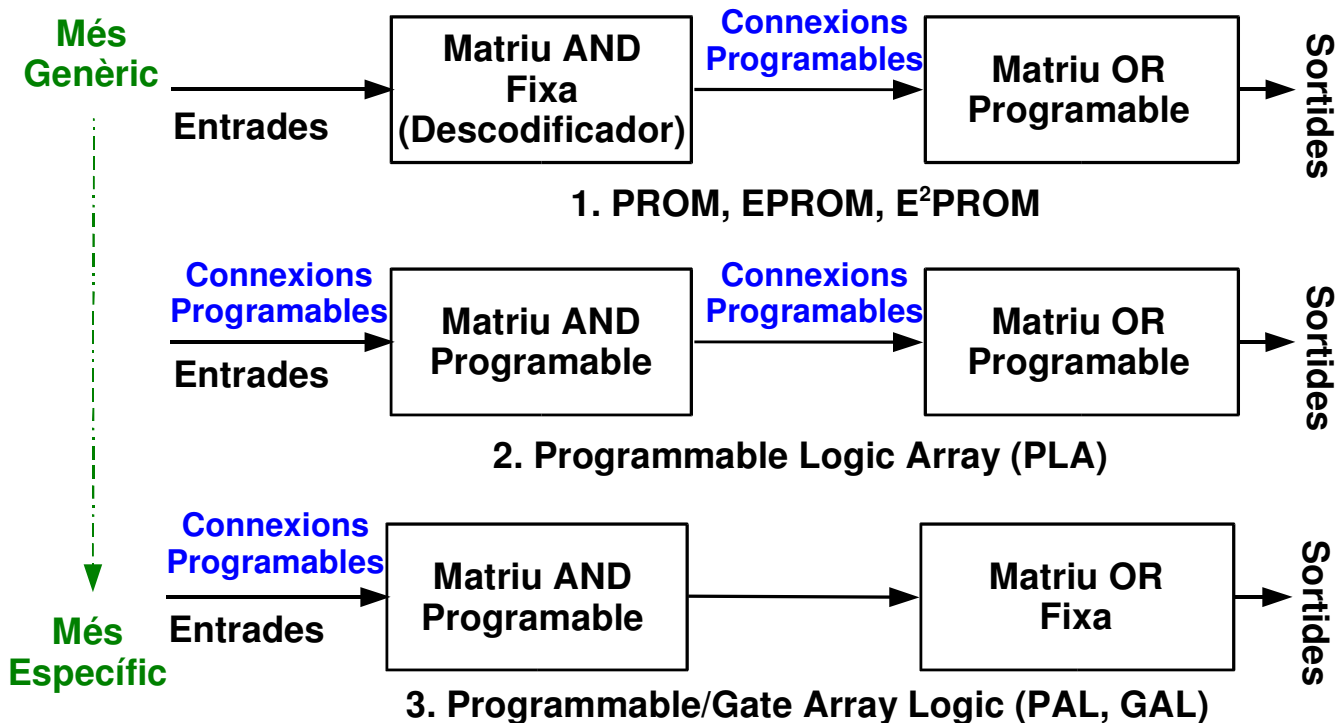


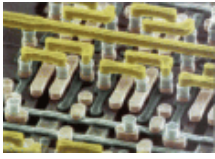
# PLD – TIPUS

- SPLD (**S**imple **PLD**)
  - ROM (**R**ead **O**nly **M**emory).
    - MROM (**M**ask **R**OM). Preprogramades de fàbrica.
    - PROM (**P**rogrammable **R**OM). Programables 1 sol cop.
    - EPROM, E<sup>2</sup>PROM. Reprogramables **n** vegades (finites).
  - PLA (**P**rogrammable **L**ogic **A**rray).
  - PAL (**P**rogrammable **A**rray **L**ogic). Cas particular de **PLA**.
  - GAL (**G**ate **A**rray **L**ogic). **PAL** esborrable elèctricament.
- CPLD (**C**omplex **PLD**)
  - FPGA (**F**ield **P**rogrammable **G**ate **A**rray). Basades en **SRAM**.



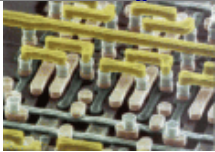
# SPLD – TIPUS



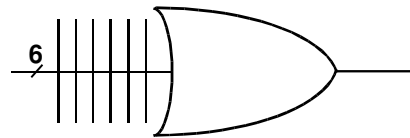
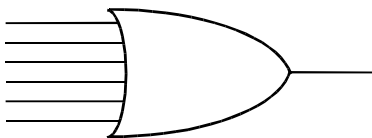


# CPLD - TIPUS

- CPLD (**C**omplex **PLD**)
  - FPGA (**F**ield **P**rogrammable **G**ate **A**rray)
    - Basades en **SRAM** (**S**tatic **R**andom **A**cces **M**emory)
    - Es programen utilitzant un computador o una **EPRM**
    - Es poden implementar circuits equivalents a dissenys de més de 10.000.000 portes
    - Poden tenir més de 1200 **PINs** d'Entrada/**S**ortida
- Exemples:

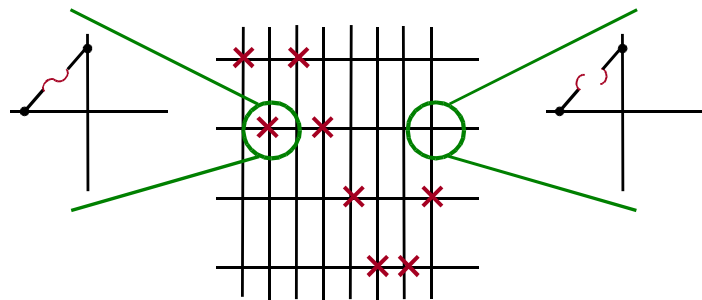
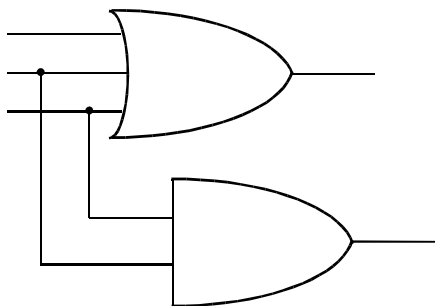


# PLD – NOTACIÓ (I)



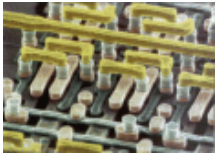
1. Simbologia Convencional

2. Simbologia en Lògica Programable



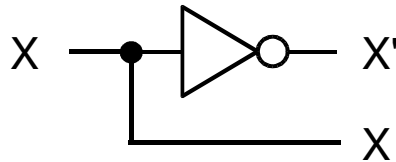
+ **No** hi ha connexió  
 • **Hi** ha connexió

+ **No** Hi ha connexió (Fusible fos)  
 × **Hi** ha connexió (Fusible intacte)

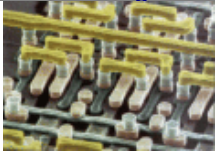
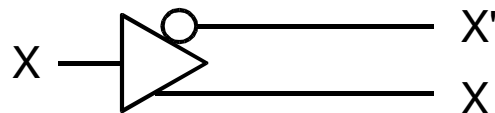


# PLD – NOTACIÓ (II)

## 1. Simbologia Convencional

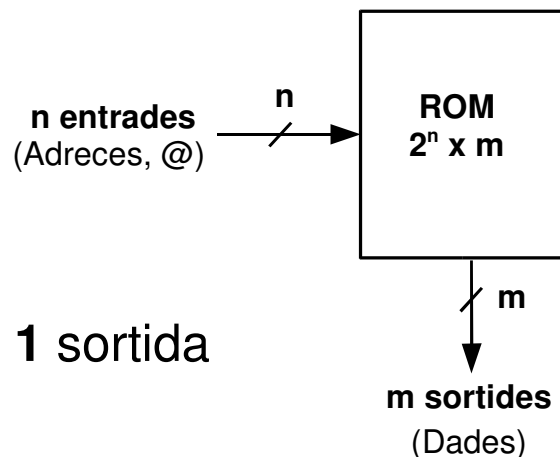


## 2. Simbologia en Lògica Programable



# SPLD – ROM (I) BLOC COMBINACIONAL

- **Read Only Memory** (Memòria de només lectura)
- **n** línies d'adreces
- **2<sup>n</sup>** cel·les de memòria
- **m** bits a cada cel·la
- Sistema combinacional
  - **m** funcions d'**n** entrades i **1** sortida



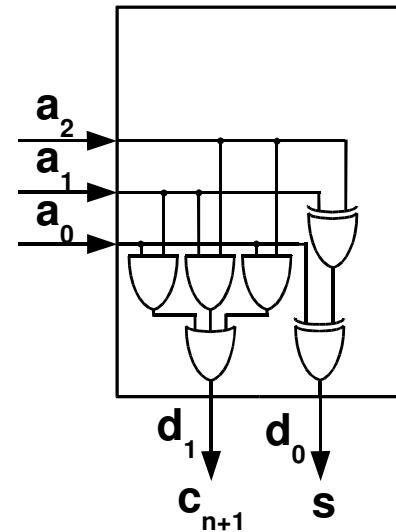
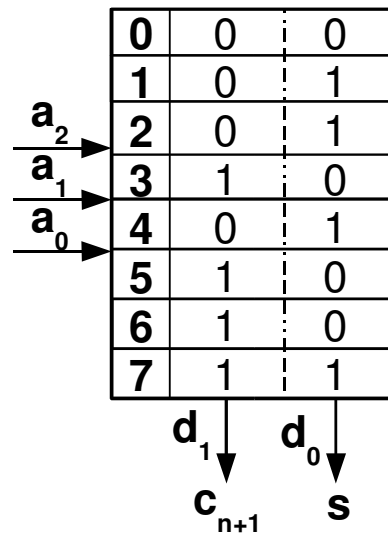


# SPLD – ROM (II)

## EXEMPLE D'IMPLEMENTACIÓ

- Sumador complert d'un bit

	$c_n$	$b$	$a$	$c_{n+1}$	$s$
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1



## SPLD – MASK ROM

- És la tecnologia més antiga
- El fabricant necessita crear unes *màscares* (la mateixa idea que els negatius en fotografia)
  - És **molt car** generar aquestes màscares
  - Surt a compte en la fabricació d'un gran número de dispositius idèntics
- Molt bo pels llocs on es produeixen molt pocs canvis
  - Cada canvi en el contingut de la **ROM** implica haver de generar noves màscares

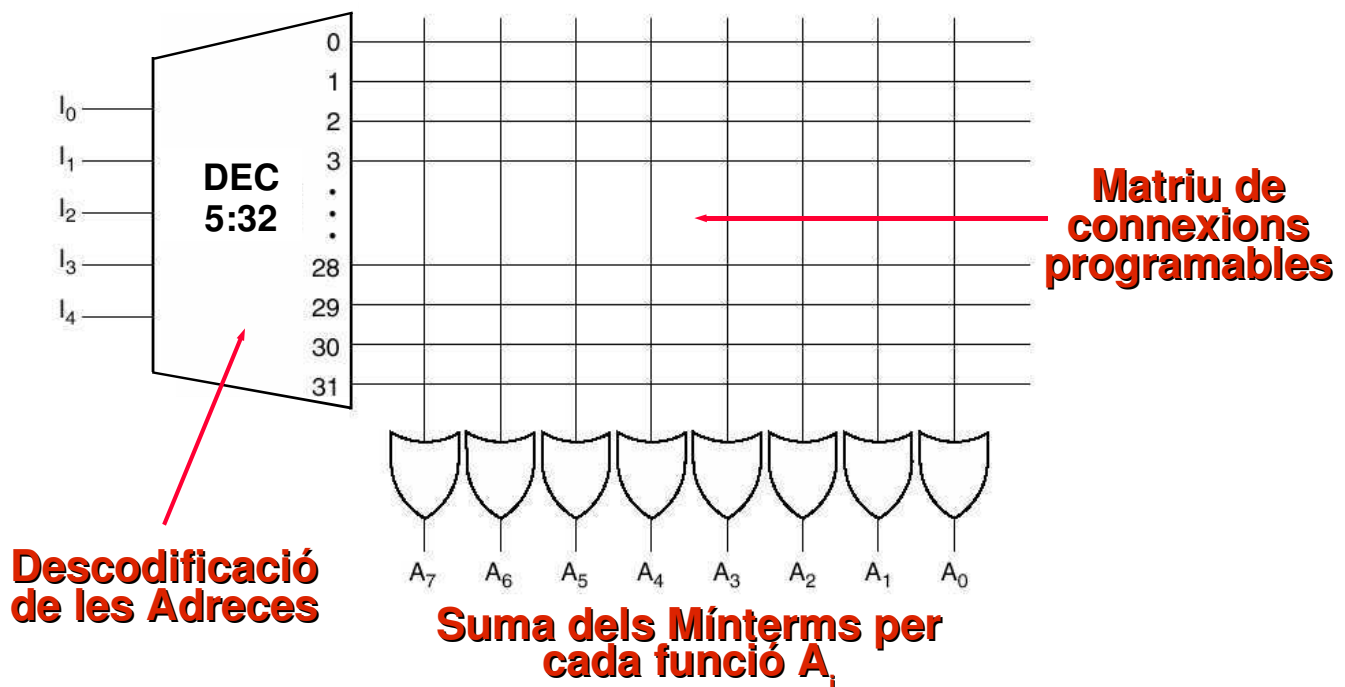


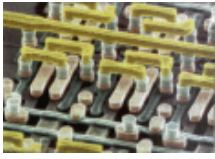
# SPLD – PROM (I) CARACTERÍSTIQUES

- Programmable **R**ead **O**nly **M**emory
- Programació
  - Basada en “*fondre fusibles*” o deixar-los intactes
  - Programable una sola vegada
  - Se sol programar utilitzant un dispositiu especialitzat
  - Utilització de voltatges més alts que el de treball
  - Vida molt llarga
- Explotació
  - Eliminar les que han quedat mal grabades



# SPLD – PROM (II) ESTRUCTURA INTERNA





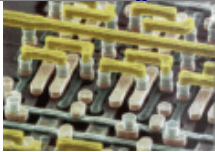
# SPLD – PROM (III) IMPLEMENTACIÓ (I)

- Taula de veritat (parcial) de 8 funcions de 5 entrades

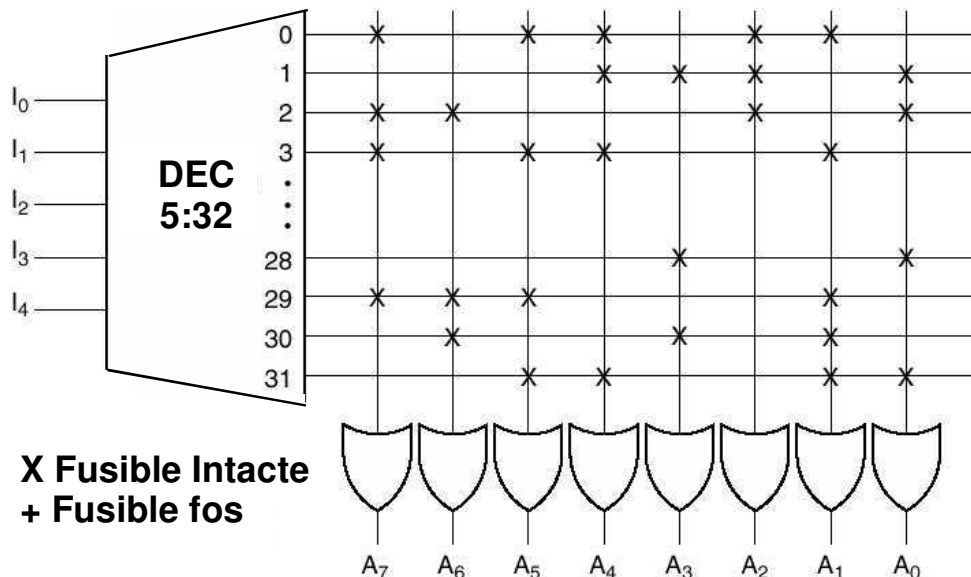
5 Entrades					8 Sortides							
Seran les Adreces					Seran les dades o Instruccions							
$I_4$	$I_3$	$I_2$	$I_1$	$I_0$	$A_7$	$A_6$	$A_5$	$A_4$	$A_3$	$A_2$	$A_1$	$A_0$
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

**Exemple:**  
Posició 3 de memòria

**Contingut:**  
Pot ser la dada 178<sub>d</sub> o la instrucció MOV DL, 0 en un 8086



# SPLD – PROM (IV) IMPLEMENTACIÓ (II)

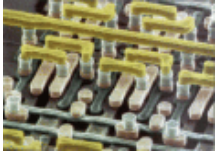


$$A_0 = \sum m(1, 2, \dots, 28, 31)$$

$$A_2 = \sum m(0, 1, 2, \dots)$$

$$A_1 = \sum m(0, 3, \dots, 29, 30, 31)$$

$$A_7 = \sum m(0, 2, 3, \dots, 29)$$



# SPLD – EPROM

## CARACTERÍSTIQUES

---

- Memòria reprogramable
  - EPROM (**E**rasable **P**PROM) (**UV** EPROM)
    - Amb llum UV (Ultravioleta) es pot “*esborrar*” l'**EPROM**
      - L'esborrat és lent (aproximadament 10 minuts)
    - Molta més flexibilitat que una **PROM**
      - Canvis de programa / dades
      - Correcció de BUGs
      - Recuperació de dispositius “*mal grabats*”
    - Pot mantenir el valor un temps limitat (10 anys)
    - Se sol programar utilitzant un dispositiu especialitzat



# SPLD - E<sup>2</sup>PROM

## CARACTERÍSTIQUES

---

- Memòria reprogramable
  - E<sup>2</sup>PROM (EEPROM, **E**lectrically **E**rasable **P**PROM)
    - Tecnologia similar a la EPROM
    - Es pot esborrar per blocs utilitzant un voltatge elevat
    - Lenta de programar
    - Normalment es programa directament al sistema final
    - Número finit de *reprogramacions*
    - És l'anomenada **Flash ROM**
      - Càmeres digitals
      - **BIOS** modernes dels **PCs**
      - Flash-Disc



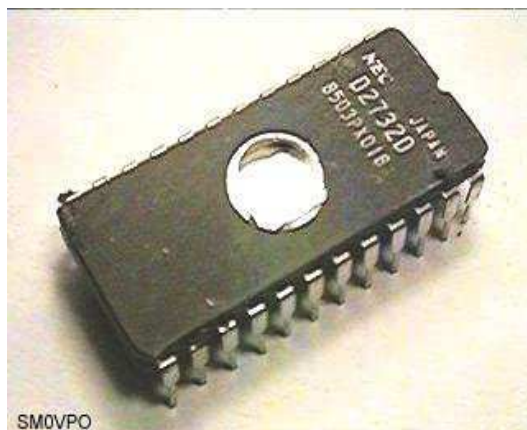
# SPLD – ROMS UTILITZACIÓ

- Per emmagatzemar DADES o PROGRAMES
  - Exemple: **BIOS** (**B**asic **I**nput **O**utput **S**ystem) del PC
    - Antigament era una PROM preprogramada de fàbrica
    - Actualment és de tecnologia E<sup>2</sup>PROM per poder actualitzar-la
- Per implementar funcions combinacionals
  - ROM 2<sup>n</sup> x m bits
    - S'hi poden implementar **m** funcions d'**n** entrades



# SPLDS - ROMS EXEMPLES

EPROM 2732



E<sup>2</sup>PROM 29EE011





# SPLD – PLA (I)

- PLA (**P**rogrammable **L**ogic **A**rray)
- IDEA:
  - Funcions d'**n** variables en forma de **Suma de Productes**
    - Cada terme (producte) pot tenir literals d'un conjunt de **2n**
    - Com a màxim es podrien sumar **2<sup>n</sup>** termes

- Exemples: **n = 3** variables

$$f_1(a,b,c) = \Sigma_3 (0, 3, 5, 6) = a'b'c' + a'bc + ab'c + abc'$$

$$f_2(a,b,c) = \Sigma_3 (1, 2, 4, 7) = a'b'c + a'bc' + ab'c' + abc$$

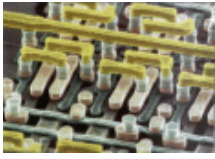
- 6 ( $2 \cdot 3$ ) literals possibles: **{a, a', b, b', c, c'}**
- Màxim 8 ( $2^3$ ) termes diferents ( $f_i = \Sigma_3 t_j$ )

	a	b	c	f <sub>1</sub>	f <sub>2</sub>
0	0	0	0	1	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	0	1



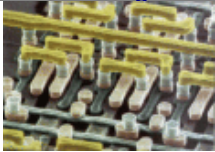
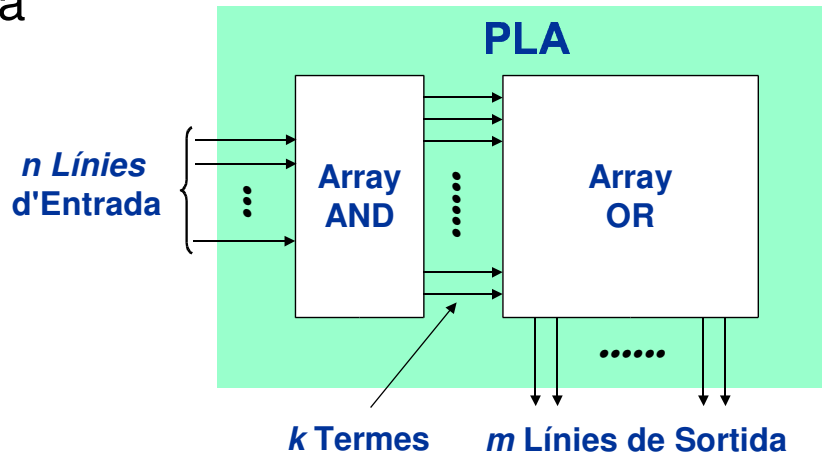
# SPLD – PLA (II)

- SOLUCIÓ: PLA **n**x**m** i **k** termes
  - **n** entrades (I)
    - Array de portes AND amb **2n** entrades **possibles** ( $I_j$  i  $I_j'$ ) (programables) cada una
  - **m** sortides (O)
    - Cada AND es pot connectar (programable) a cada OR de l'array d'**m** d'ORs de la sortida
  - **k** portes AND
    - k és molt més petit que les 2<sup>n</sup> possibles AND diferents que poden aparèixer en una funció



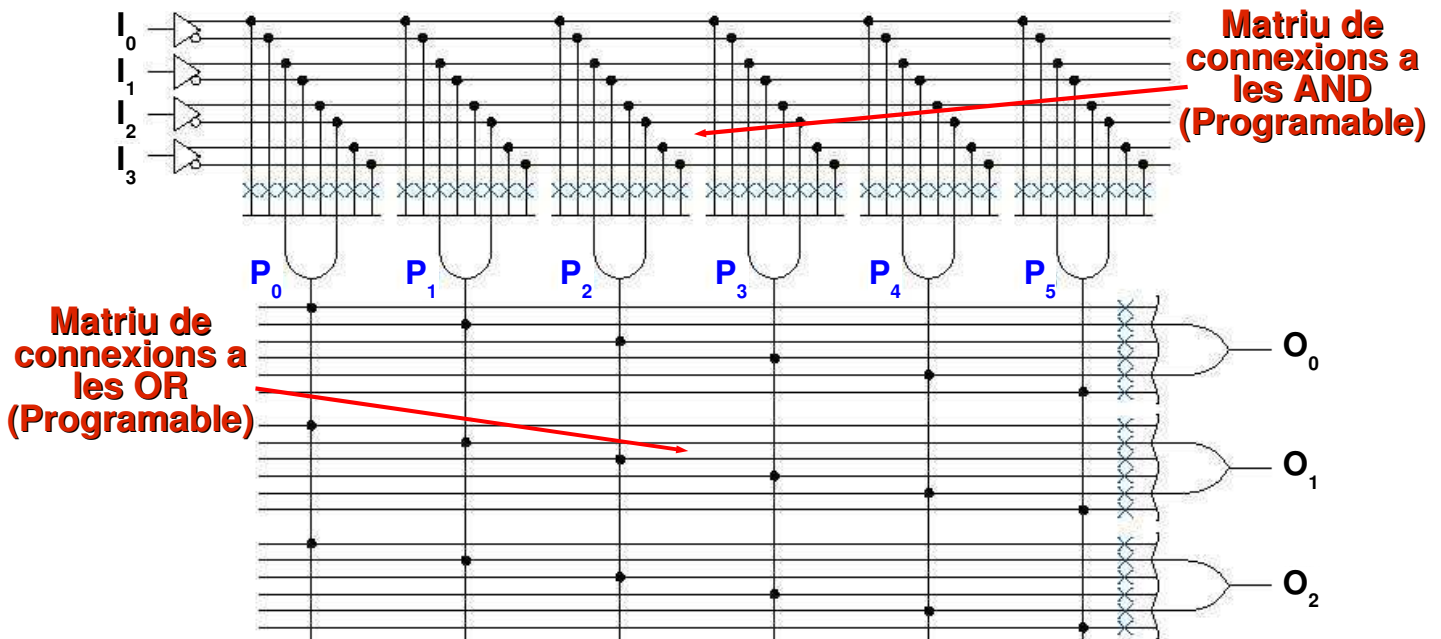
# SPLD – PLA (III)

- PLA  $n \times m$  i  $k$  termes
  - Permet implementar  **$m$  funcions** en forma de **Suma de Productes** d' **$n$  variables** i de fins a  **$k$  termes** cada una



# SPLD – PLA DIAGRAMA LÒGIC (I)

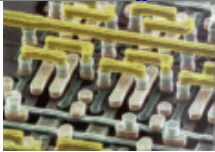
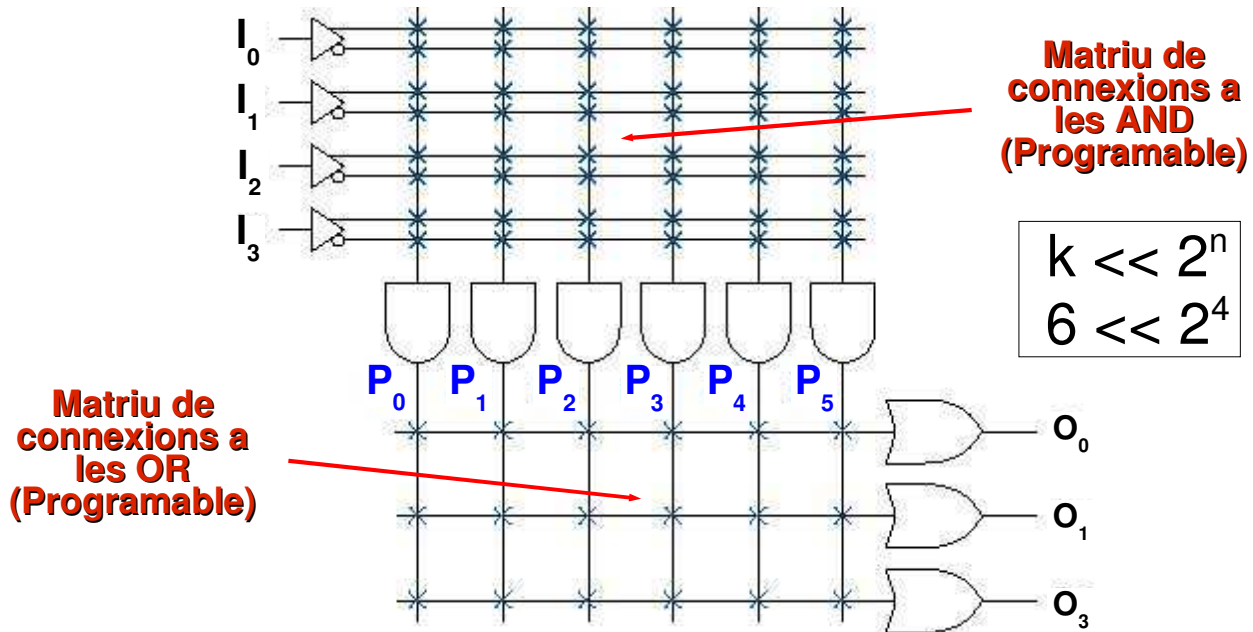
- PLA  $4 \times 3$  amb 6 termes ( $n = 4$ ,  $m = 3$  i  $k = 6$ )





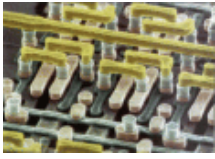
# SPLD – PLA DIAGRAMA LÒGIC (II)

- PLA 4x3 amb  $k = 6$ . Representació compacta:



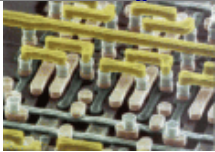
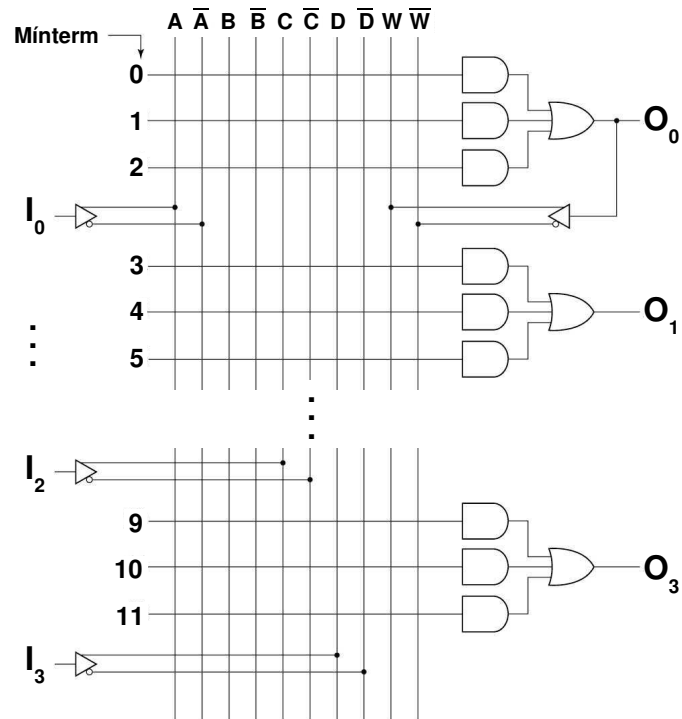
# SPLD - PAL

- PAL (**P**rogrammable **A**rray **L**ogic)
- IDEA:
  - Cas particular d'una **PLA**
  - L'Array **OR** és fixe
- **Resultat**
  - Pot resoldre els mateixos problemes que la **PLA** escollint correctament els paràmetres **n**, **m** i **k** per cada problema determinat
  - És més simple de programar: només cal programar les connexions AND



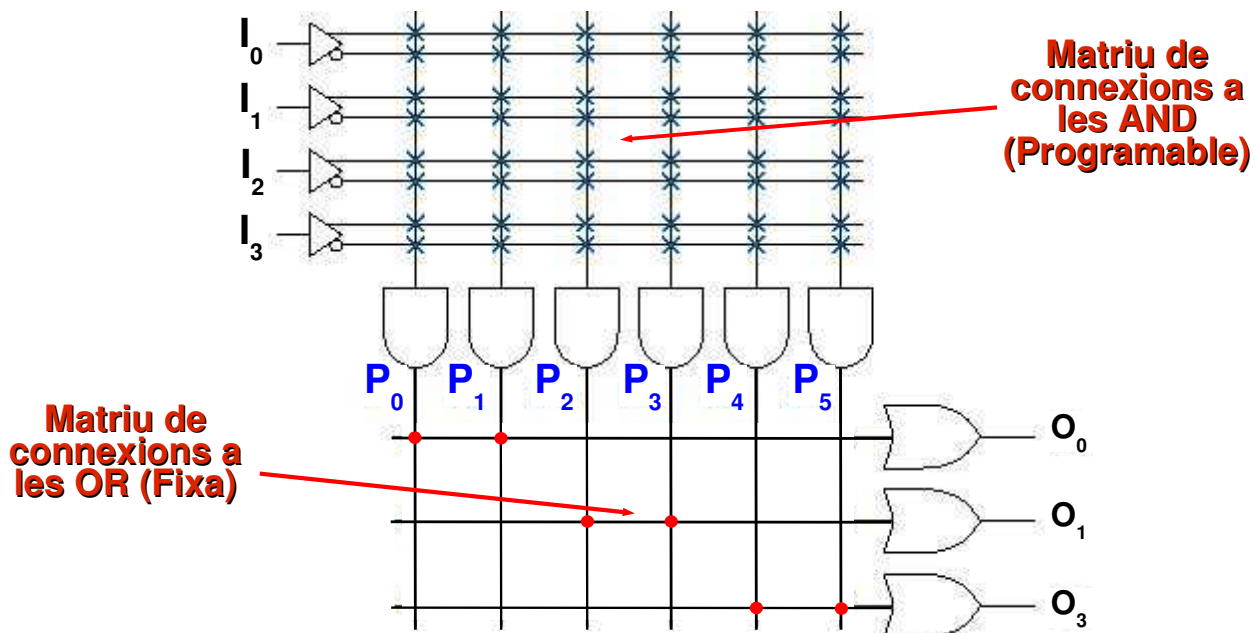
# SPLD – PAL DIAGRAMA LÒGIC (I)

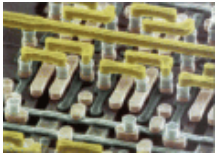
- Exemple de PAL
  - n = 4 Entrades
  - m = 4 Sortides
  - k = 3 Termes
- Reutilització de Funcions (W)



# SPLD – PAL DIAGRAMA LÒGIC (II)

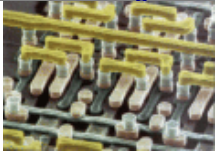
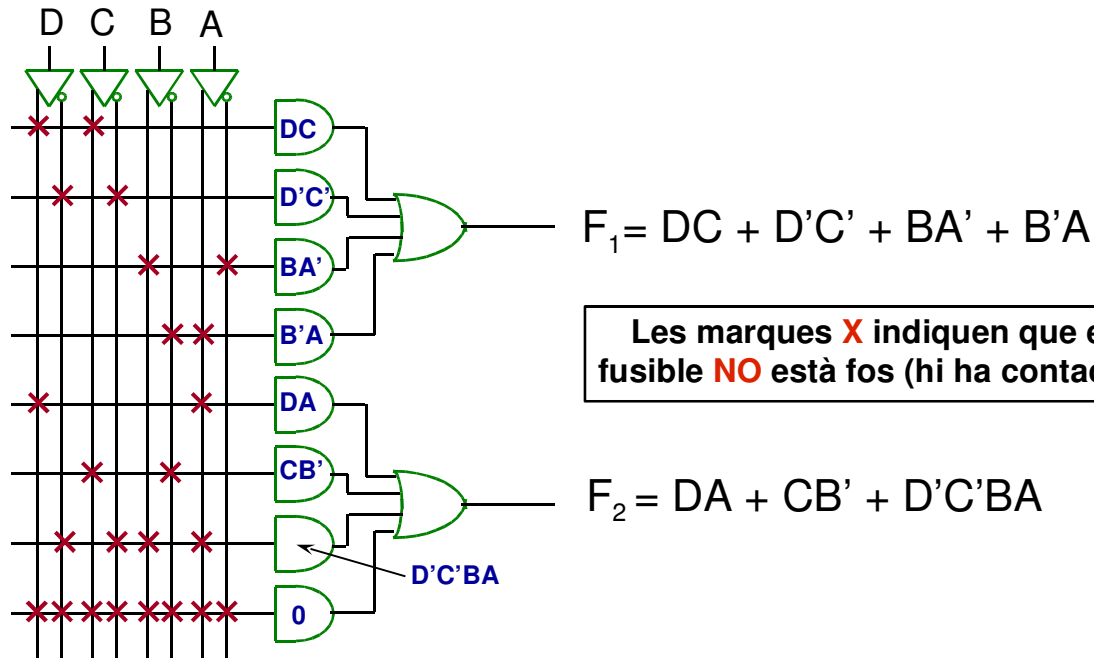
- PAL 4x3 amb k = 2 termes per funció





# SPLD – PAL EXEMPLE

- **PAL 4x2** amb  $k = 4$  termes per funció



# SPLD - GAL

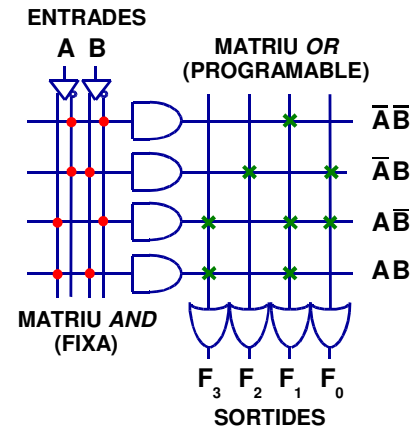
- **GAL (Gate Array Logic)**
  - Bàsicament és una PAL
    - Molt densa (molts termes: **ANDs**)
    - Connexions reprogramables
      - La matriu AND és d'alguna tecnologia reprogramable ( $E^2$ )
    - A la sortida de les **OR** hi ha cel·les programables (sequencials / combinacionals)
      - OLMC (**O**utput **L**ogic **M**acro**C**ell)



# SPLD – IMPLEMENTACIÓ AMB ROMs, PLAs i PALs

**FUNCIONS:**  $F_0 = \bar{A}B + A\bar{B}$   $F_1 = A + \bar{B}$   $F_2 = \bar{A}B$   $F_3 = A$

## 1. PROM $n=2$ $m=4$



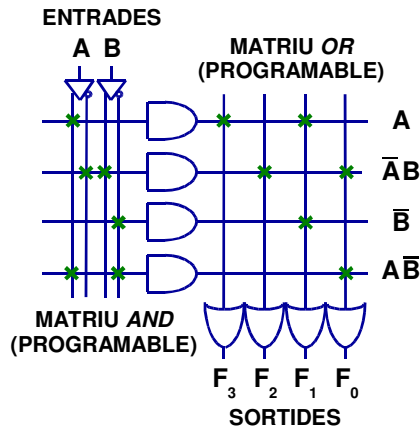
$$F_0 = \bar{A}B + A\bar{B}$$

$$F_1 = \bar{A}\bar{B} + A\bar{B} + AB = A + \bar{B}$$

$$F_2 = \bar{A}\bar{B}$$

$$F_3 = \bar{A}\bar{B} + AB = A(B + \bar{B}) = A \cdot 1 = A$$

## 2. PLA $n=2$ $m=4$ $k=4$



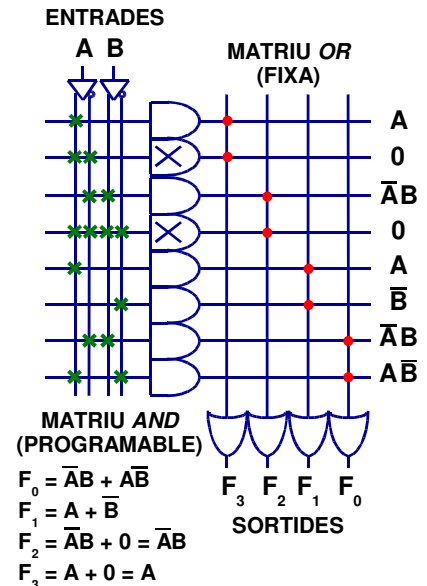
$$F_0 = \bar{A}B + A\bar{B}$$

$$F_1 = A + \bar{B}$$

$$F_2 = \bar{A}\bar{B}$$

$$F_3 = A$$

## 2. PAL $n=2$ $m=4$ $k=2$

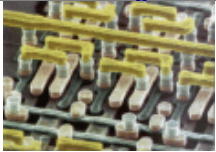


$$F_0 = \bar{A}B + A\bar{B}$$

$$F_1 = A + \bar{B}$$

$$F_2 = \bar{A}\bar{B} + 0 = \bar{A}\bar{B}$$

$$F_3 = A + 0 = A$$



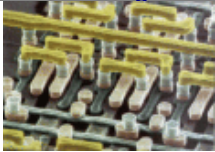
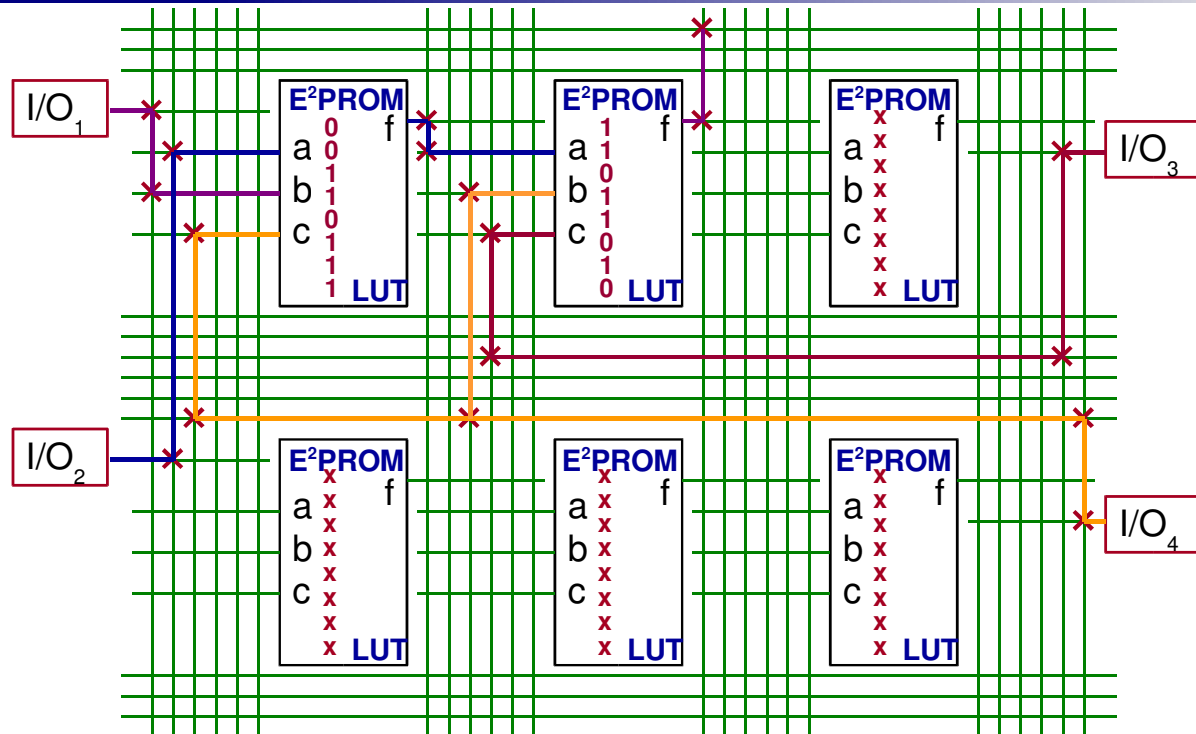
# SPLD – ROM vs PLA vs PAL

(MÉS GENÈRIC) ROM → PLA → PAL (MÉS ESPECÍFIC)

- **ROMs**
  - Per implementar funcions expressades en forma canònica
  - Per funcions incompletes es desprecia molta memòria
- **PLA**
  - Permet **reutilitzar minterms** a diferents funcions de sortida
- **PLA i PAL**
  - Permeten implementar funcions simplificades
  - Limitació en el **número de termes** dins la suma
  - Cal escullir la **PAL** o la **PLA** amb els paràmetres **n**, **m** i **k** que s'ajustin millor al problema que es vol resoldre



# CPLD - FPGA DIAGRAMA LÒGIC



# SISTEMES DE DESENVOLUPAMENT

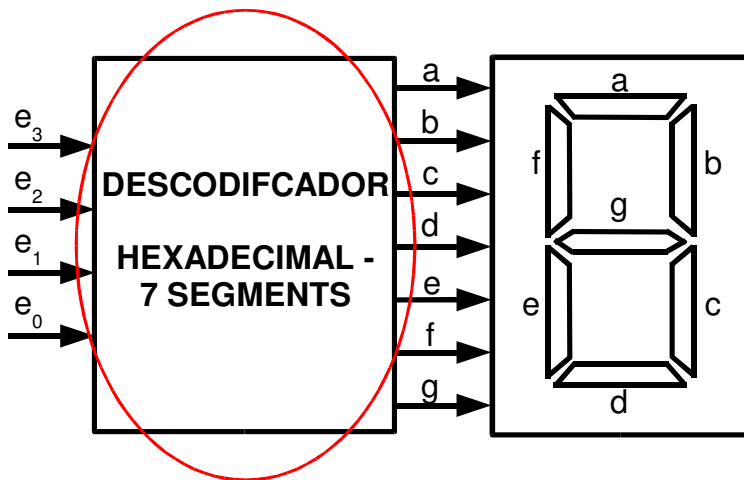
- Sistemes dissenyats per desenvolupar usant PLDs:
  - **Software**
    - Llenguatge d'Alt Nivell: HDL (**H**ardware **D**escription **L**anguage)
    - Sistema CAD (**C**omputer **A**ided **D**esign)
    - Editor de màquines d'estats
    - Compilador
    - Simulador temporal (retards de propagació)
  - **Hardware**
    - **Programador** de dispositius de lògica programable
      - PAL, PLA, PLD, FPGA
  - Exemples
    - ABEL (**A**dvanced **B**oolean **E**xpression **L**anguage)
    - VHDL (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit) HDL)



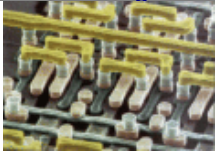
# PLD - ABEL

## EXEMPLE: HEX-7SEG (I)

- Implementació
  - Descodificador HEX-7Segments



	$e_3$	$e_2$	$e_1$	$e_0$	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	1	1	1	0	1	1	1
11	1	0	1	1	0	0	1	1	1	1	1
12	1	1	0	0	0	0	1	0	1	0	1
13	1	1	0	1	0	1	1	1	1	1	0
14	1	1	1	0	1	0	0	1	1	1	1
15	1	1	1	1	1	0	0	0	1	1	1



# PLD - ABEL

## EXEMPLE: HEX-7SEG (II)

- Descripció utilitzant equacions

```

module hex7seg
  title 'Descodificador Hexadecimal - 7 Segments'
  hex7seg device 'p16v6';
  e3, e2, e1, e0 pin 2, 3, 4, 5;
  a, b, c, d, e, f, g pin 19, 18, 17, 16, 15, 14, 13 istype 'com';
  h, l = 1, 0;

  equations
    a = ( !e2 & !e0 # e2 & e1 # !e3 & e2 & e0 # !e3 & e1 & e0 #
          e3 & !e2 & !e1 );
    b = ( !e2 & !e0 # !e3 & !e2 # !e3 & !e1 & !e0 # !e3 & e1 & e0 #
          e3 & !e1 & e0 );
    c = ( !e1 # !e3 & !e2 # e3 & !e2 # !e3 & e1 );
    d = ( !e2 & !e1 & !e0 # e2 & !e1 & e0 # e3 & !e2 & e0 #
          !e3 & !e2 & e1 # e2 & e1 & !e0 );
    e = ( !e2 & !e0 # e3 & e2 # e1 & !e0 # e3 & e1 );
    f = ( e3 & !e2 # e3 & e1 # !e2 & !e1 & !e0 # !e3 & e2 & !e1 #
          e2 & e1 & !e0 );
    g = ( e3 # e1 & !e0 # e2 & !e1 # !e2 & e1 );
end hex7seg
  
```

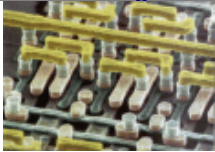


# PLD - ABEL

## EXAMPLE: HEX-7SEG (III)

- **Descripció utilitzant la taula de veritat**

```
module hex7seg
  title 'Descodificador Hexadecimal - 7 Segments'
  hex7seg device 'p16v6';
  e3, e2, e1, e0 pin 2, 3, 4, 5;
  a, b, c, d, e, f, g pin 19, 18, 17, 16, 15, 14, 13 istype 'com';
  truth_table ([e3, e2, e1, e0] -> [a, b, c, d, e, f, g])
    [0, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 0];
    [0, 0, 0, 1] -> [0, 1, 1, 0, 0, 0, 0];
    [0, 0, 1, 0] -> [1, 1, 0, 1, 1, 0, 1];
    [0, 0, 1, 1] -> [1, 1, 1, 1, 0, 0, 1];
    [0, 1, 0, 0] -> [0, 1, 1, 0, 0, 1, 1];
    [0, 1, 0, 1] -> [1, 0, 1, 1, 0, 1, 1];
    [0, 1, 1, 0] -> [1, 0, 1, 1, 1, 1, 1];
    [0, 1, 1, 1] -> [1, 1, 1, 0, 0, 0, 0];
    [1, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 1];
    [1, 0, 0, 1] -> [1, 1, 1, 1, 0, 1, 1];
    [1, 0, 1, 0] -> [1, 1, 1, 0, 1, 1, 1];
    [1, 0, 1, 1] -> [0, 0, 1, 1, 1, 1, 1];
    [1, 1, 0, 0] -> [0, 0, 1, 0, 1, 0, 1];
    [1, 1, 0, 1] -> [0, 1, 1, 1, 1, 0, 1];
    [1, 1, 1, 0] -> [1, 0, 0, 1, 1, 1, 1];
    [1, 1, 1, 1] -> [1, 0, 0, 0, 1, 1, 1];
end hex7seg
```

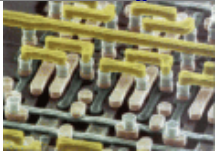
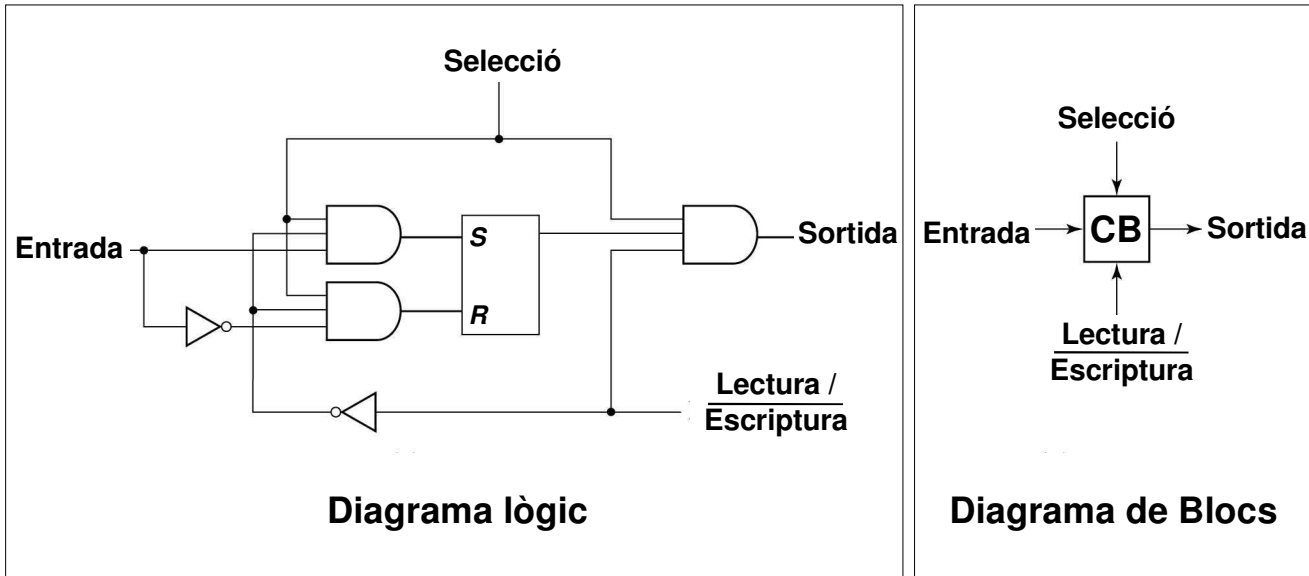


# MEMÒRIA RAM

- **RAM (Random Access Memory)**
  - Agrupació en cel·les bàsiques (d'1 bit)
  - Accés a les cel·les similar a una **ROM** (Descodificació)
  - Sistema seqüencial
- **TIPUS**
  - **SRAM (Static RAM)**
  - **DRAM (Dinamic RAM)**
    - Cal refrescar-la però necessita menys transistors
    - És almenys 4 vegades més densa
    - Molt més barata
    - Consumeix menys



# RAM – CEL·LA BÀSICA DE MEMÒRIA



# RAM – EXEMPLE DE 4x4 BITS

