Apresentação em PDF:
- selecione o modo de "ecran completo"
- navegue com Pg_Dn

Dispositivos Lógicos Programáveis

Princípio de funcionamento

Ambiente de desenvolvimento

Dispositivos Lógicos Programáveis

- Introdução
- Estruturas *hardware* programáveis
  - PROM
  - PLA
  - PAL (GAL)
- CUPL – linguagem de programação
  - Ambiente de desenvolvimento

PROM – Programmable Read Only Memory

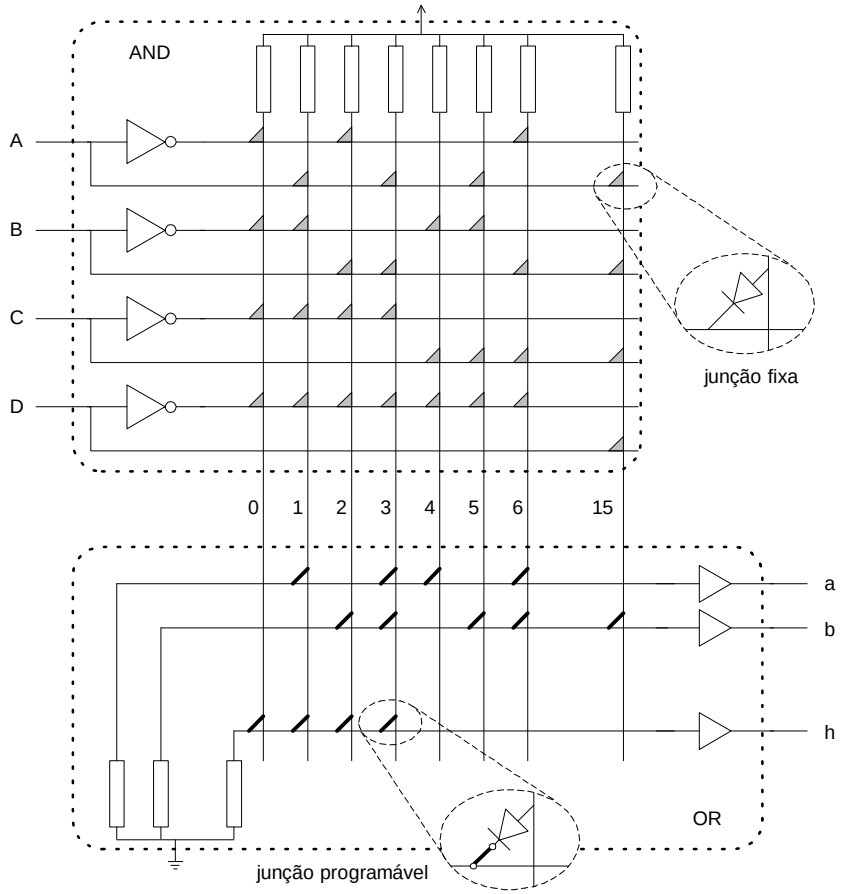
Nas memórias do tipo PROM apenas a malha OR é programável

A malha AND (inalterável) realiza **todos os termos mínimos** em função das variáveis de entrada

As funções de saída exprimem-se na forma **canónica** AND-OR

A programação corresponde a "escolher" os termos mínimos que formam cada uma das saídas

É uma estrutura genérica onde ocorre desperdício por não se gerarem apenas os termos AND necessários



PLA – Programmable Logic Array

Nas PLA ambas as malhas (AND e OR) são programáveis

As saídas exprimem-se por formas AND-OR que não têm de ser canónicas

A versatilidade proposta é pouco aproveitada na malha OR

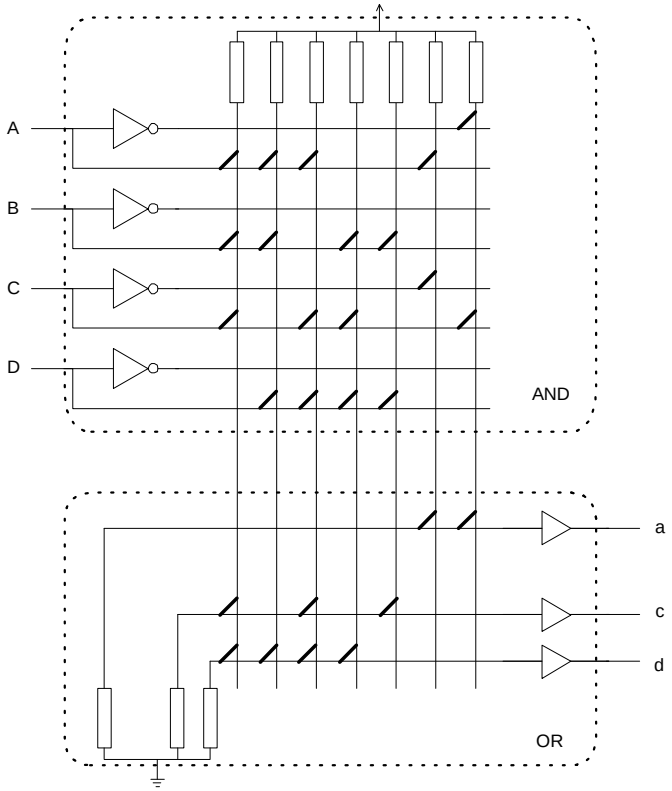
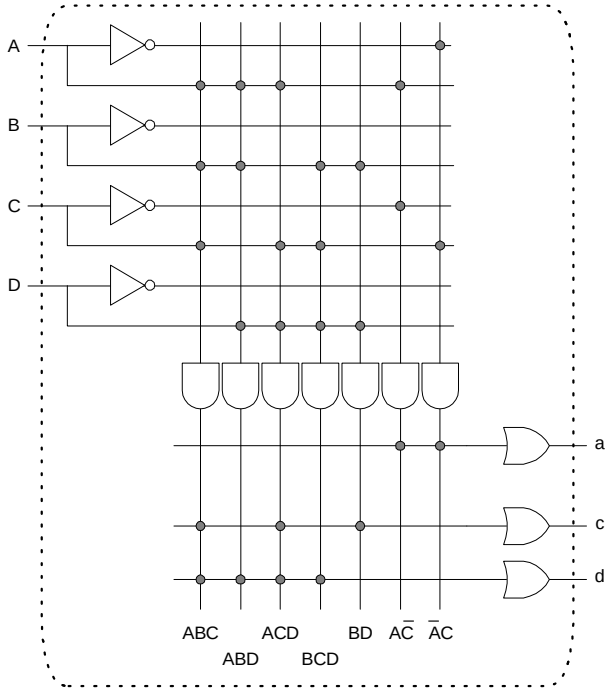


Diagrama Lógico Equivalente



PAL – Programmable Array Logic

Nas PAL apenas a malha AND é programável

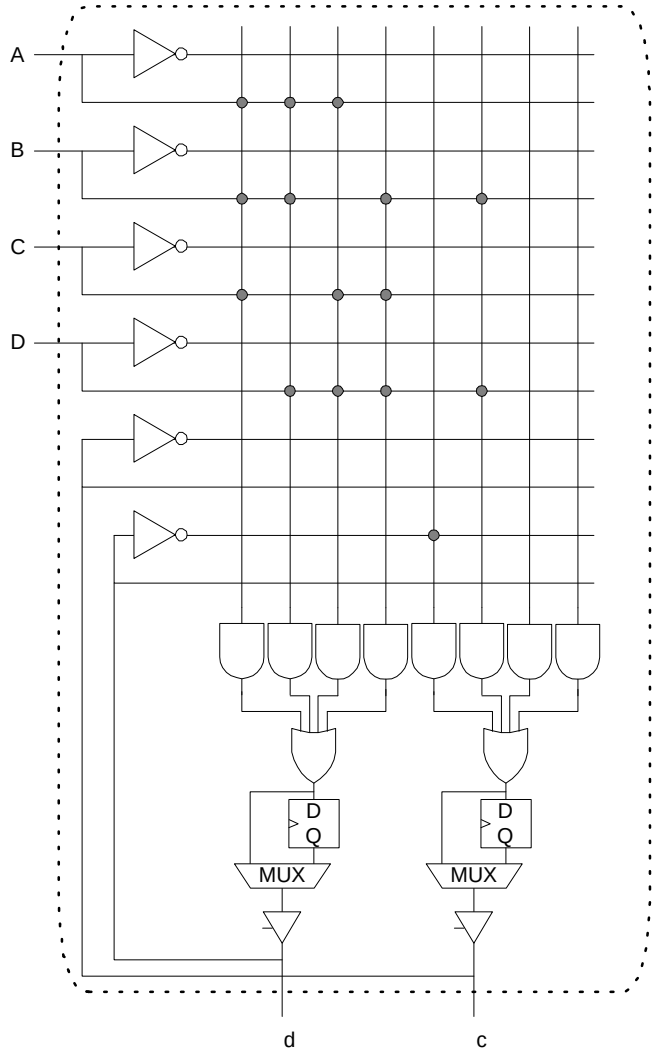
A malha OR é afectada pelos **termos AND** que estão ligados em cada nó

As funções de saída exprimem-se em formas AND-OR simplificadas

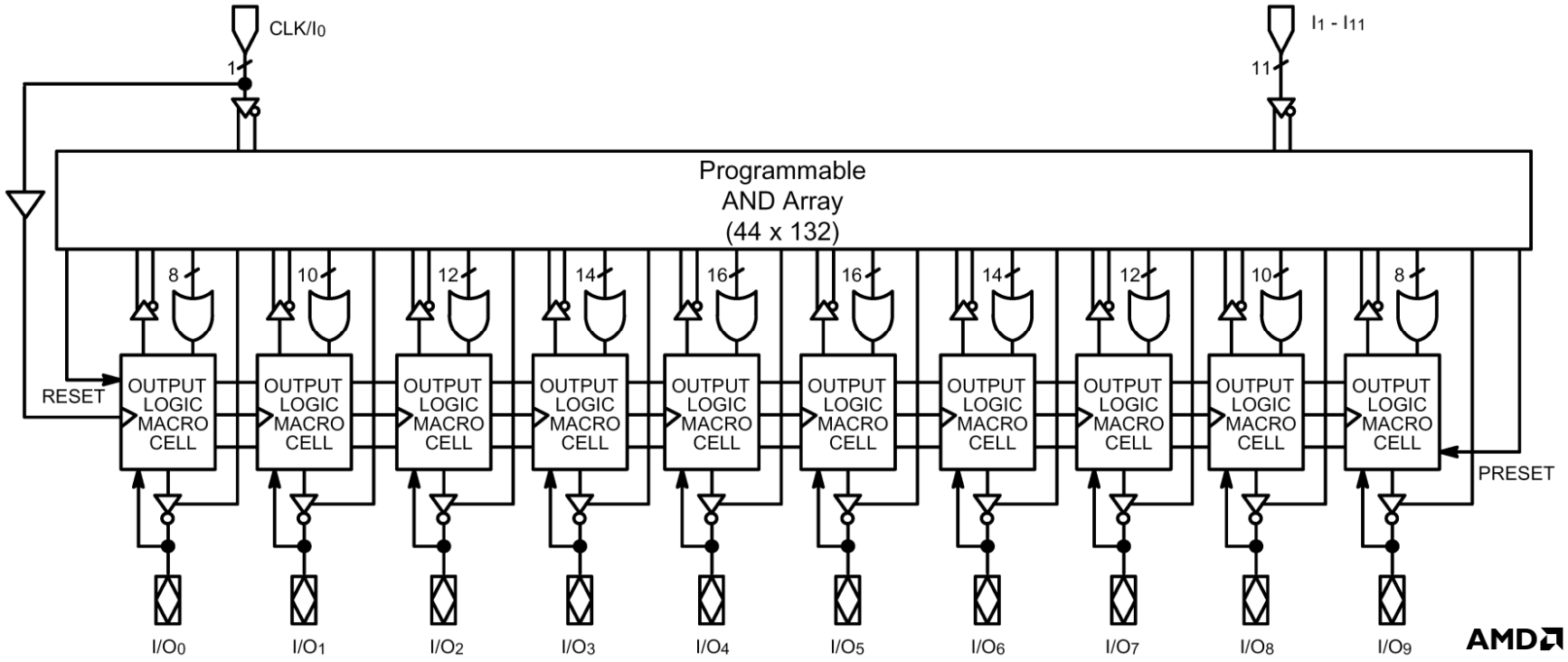
As saídas podem ser registadas, facilitando a realização de **circuitos sequenciais**

A programação corresponde a "realizar" os termos AND, por combinação das entradas

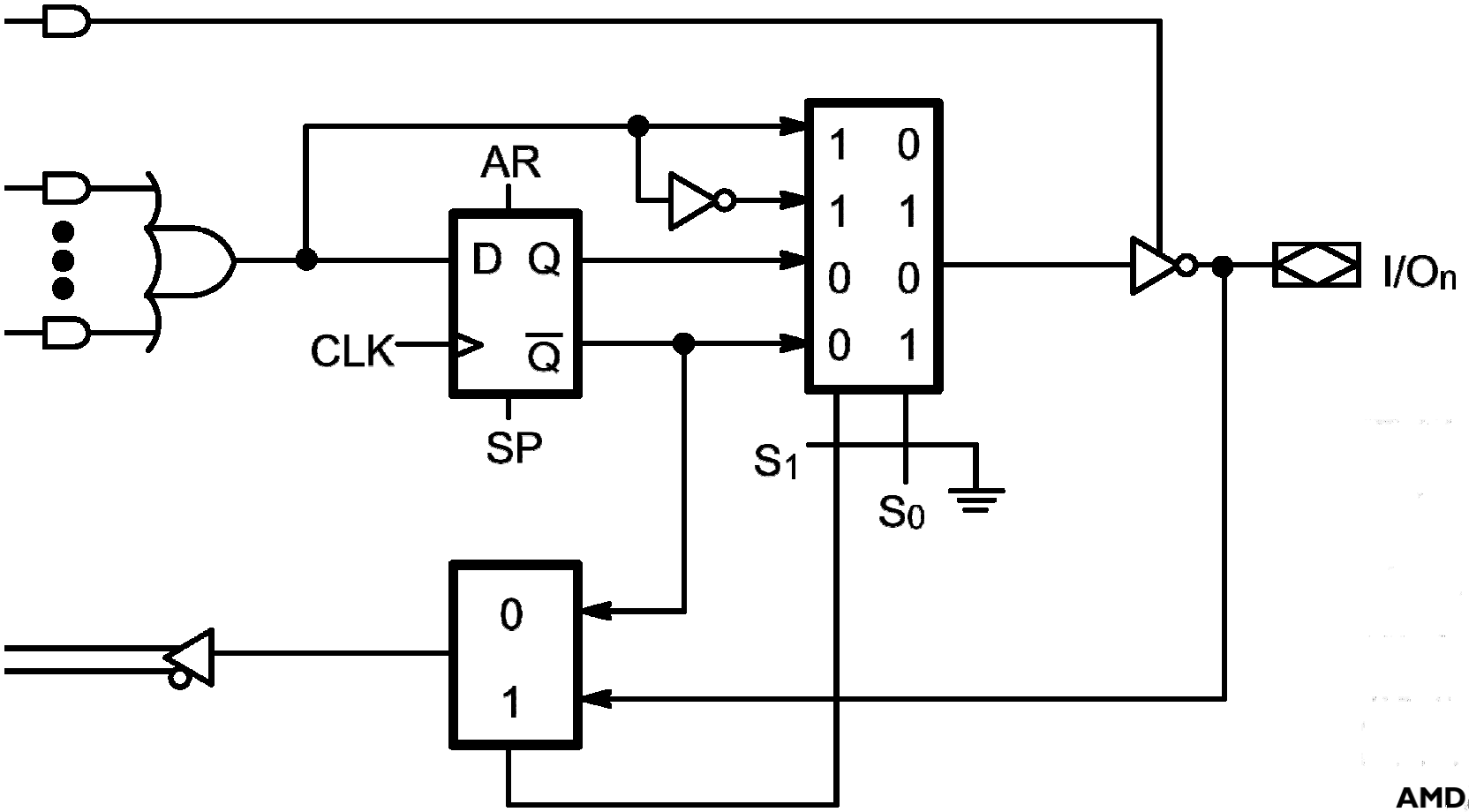
É uma estrutura genérica onde se evita o desperdício, característico das PROM



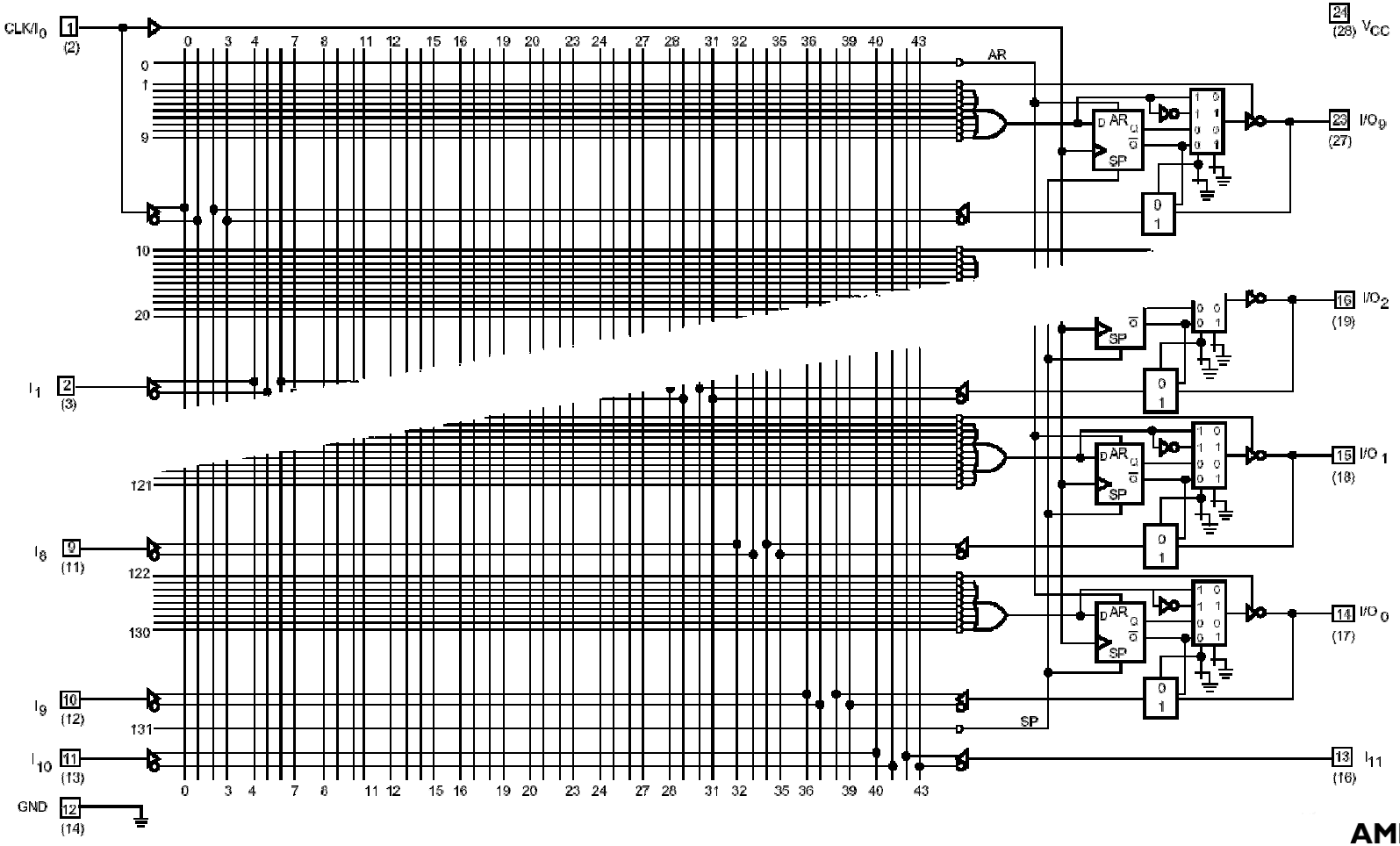
PALCE22V10 – Estrutura geral



PALCE22V10 – Macro-célula



PALCE22V10 – Malha de ligações (corte)



AMD

PALCE22V10 – Explicação da referência

PAL CE 22 V 10 H -5 P C /5

FAMILY TYPE
PAL = Programmable Array Logic

TECHNOLOGY
CE = CMOS Electrically Erasable

NUMBER OF ARRAY INPUTS

OUTPUT TYPE
V = Versatile

NUMBER OF OUTPUTS

POWER
Q = Quarter Power (55 mA Icc)
H = Half Power (90–140 mA Icc)

SPEED
-5 = 5 ns t_{PD}
-7 = 7.5 ns t_{PD}
-10 = 10 ns t_{PD}
-15 = 15 ns t_{PD}
-20 = 20 ns t_{PD}
-25 = 25 ns t_{PD}

PACKAGE TYPE
P = 24-Pin 300 mil Plastic SKINNYDIP (PD3024)
J = 28-Pin Plastic Leaded Chip Carrier (PL 028)
S = 24-Pin Plastic Gull-Wing Small Outline Package (SO 024)

OPERATING CONDITIONS
C = Commercial (0°C to +75°C)
I = Industrial (–40°C to +85°C)

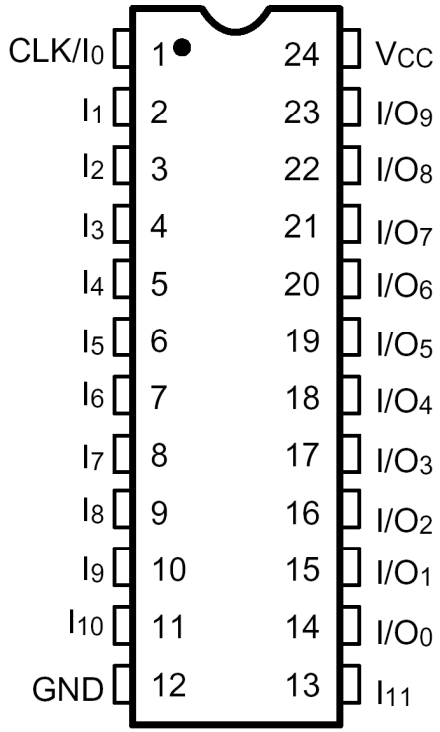
PROGRAMMING DESIGNATOR
Blank = Initial Algorithm
/4 = First Revision
/5 = Second Revision (Same Algorithm as /4)

OPTIONAL PROCESSING
Blank = Standard Processing

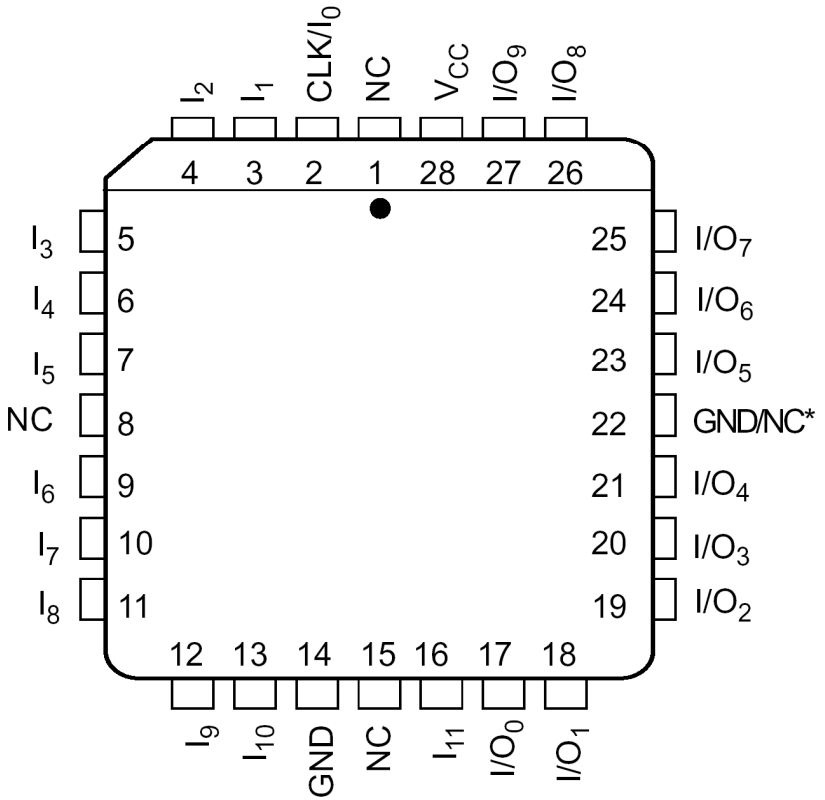
AMD

PALCE22V10 – Encapsulamento

SKINNYDIP/SOIC/FLATPACK



PLCC/LCC



AMD

Linguagens Descritivas de *Hardware* (HDL)

PALASM

ABEL

CUPL

Verilog

VHDL

CUPL – Ficheiros envolvidos no desenvolvimento e na simulação

PLD	Criado pelo utilizador Contém todas as instruções lógicas necessárias à definição do circuito pretendido É o programa propriamente dito, em CUPL
DOC	Gerado pelo compilador de CUPL Contém todas equações lógicas resultantes da descrição (programa em CUPL) Reporta os erros encontrados durante a compilação, com indicação do local Descreve a correspondência entre o circuito pretendido e o dispositivo seleccionado
ABS	Gerado pelo compilador de CUPL Ficheiro necessário ao processo de simulação
LST	Gerado pelo compilador de CUPL Contém as linhas do programa original com a respectiva numeração Os erros são indicados (com o símbolo ^) nas linhas onde ocorreram
JED	Gerado pelo compilador de CUPL Ficheiro utilizado pelo programador para seleccionar as ligações a queimar O nome do ficheiro (em formato MS-DOS) é dado pelo campo *Name* do ficheiro .PLD
SI	Criado pelo utilizador É o ficheiro de entradas para o simulador Contém a lista dos vectores de teste
SO	Gerado pelo simulador de CUPL Contém os resultados da simulação, incluindo eventuais erros Utilizado para visualização gráfica dos resultados da simulação

CUPL – Estrutura do ficheiro .PLD

- Cabeçalho
  - O padrão é proposto pelo compilador (**WinCupl** da firma **Atmel**)
  - Todos os campos têm de constar, mas o preenchimento é facultativo
  - O campo ***Name*** **tem de ser preenchido** (em formato MS-DOS) e determina o nome do ficheiro .JED
  - Recomenda-se que no campo ***Device*** conste **`p22v10`**
- Declarações
  - Identificam-se aqui as variáveis (de entradas e de saída) do programa
- Corpo do programa
  - Contém todas as expressões lógicas da realização pretendida

CUPL – Operadores e notações numéricas

Operador	Exemplo	Descrição
!	!A	NOT
&	A&B	AND
#	A#B	OR
$	A$B	XOR

Número	Base	Valor Decimal
'b'0	binário	0
'b'1101	binário	13
'o'663	octal	435
'D'92	decimal	92
'h'BA	hexadecimal	186
'b'[001..100]	binário	intervalo de 1 a 4

Por omissão, os números são assumidos em base dezasseis, excepto quando indicam índices, onde se usa base dez

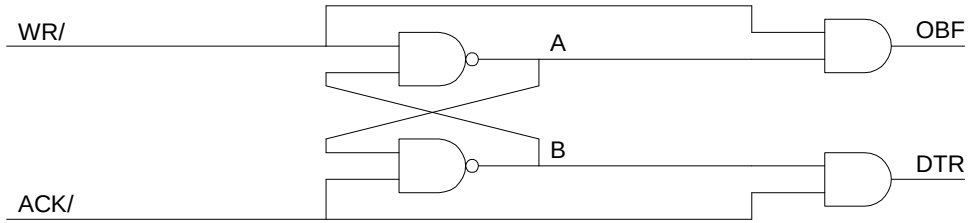
As variáveis diferem entre maiúsculas e minúsculas e não podem conter espaços nem podem coincidir com palavras-chave

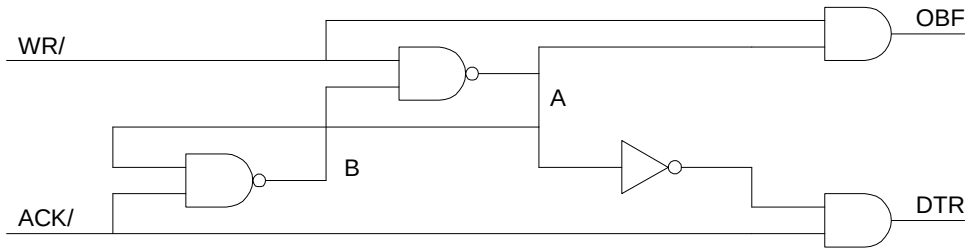
CUPL – Realização de uma função à custa dos termos AND

```
Name     xor ;
PartNo   00 ;
Date     2002.01.13 ;
Revision 01 ;
Designer HM ;
Company  ISEL ;
Assembly None ;
Location  ;
Device   g22v10 ;

/* ***** INPUT PINS ******/
PIN 1 = a ;
PIN 2 = b ;

/* ***** OUTPUT PINS *****/
PIN 14 = term1 ;
PIN 15 = term2 ;
PIN 16 = xor ;

/* ******** BODY *********/
term1 = !a & b ;
term2 = a & !b ;
xor = term1 # term2 ;
```

CUPL – Realização de uma máquina de estados assíncrona com saídas função de estado e entrada

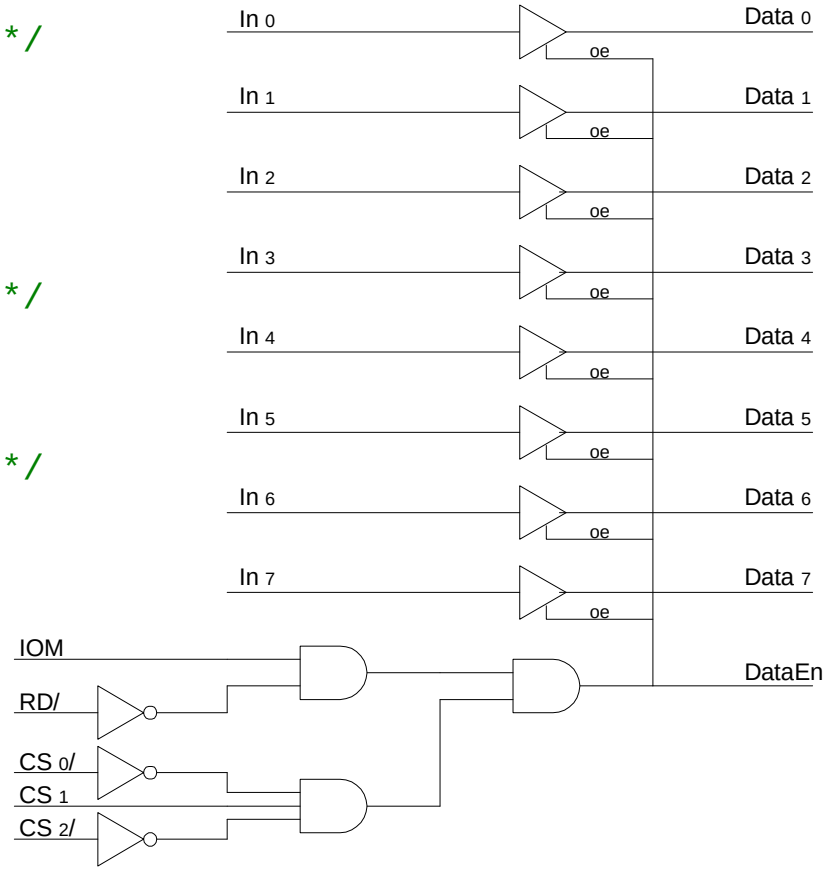
```
/* ****** INPUT PINS *******/
PIN 1 = !WR ;
PIN 2 = !ACK ;

/* ****** OUTPUT PINS ******/
PIN 14 = OBF ;
PIN 15 = DTR ;
PIN 16 = A ;
/*PIN 17 = B ; */

/* ********* BODY **********/
/*
A   = !(!WR  & B) ;
B   = !(!ACK & A) ;

OBF = !WR  & A ;
DTR = !ACK & B ;
*/

A = !(!WR & !(A & !ACK)) ;
OBF = !WR & A ;
DTR = !ACK & !A ;
```





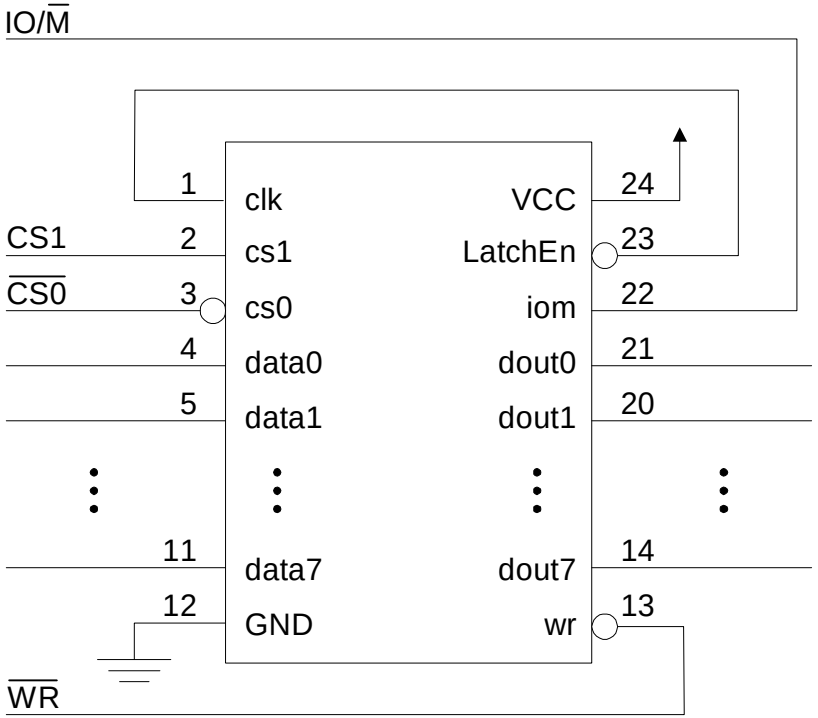
CUPL – *Octal buffer tri-state (input port)*

```
/* ********** INPUT PINS *************/
pin [1..3] = [!cs2,cs1,!cs0] ;
pin 13 = !rd ;
pin 22 = iom ;
pin [4..11] = [in0..7] ;

/* ********** OUTPUT PINS ************/
pin [14..21] = [data7..0] ;
pin 23 = DataEn ;

/* ************ BODY ***************/
DataEn = [cs0, cs1, cs2, iom, rd]:& ;

[data0..7] = [in0..7] ;

$REPEAT i=[0..7]
          data{i}.oe = DataEn ;
$REPEND
```



CUPL – Registo de oito bits (*output port*)

```
/* ********* INPUT PINS *********/
pin 1 = clk ;
pin [2..3] = [cs1,!cs0] ;
pin 13 = !wr ;
pin 22 = iom ;
pin [4..11] = [data0..7] ;

/* ******** OUTPUT PINS *********/
pin [14..21] = [dout7..0] ;
pin 23 = !LatchEn ;

/* ********** BODY ************/
LatchEn = [cs0, cs1, iom, wr]:& ;

[dout0..7].ar = 'b'0 ;
[dout0..7].sp = 'b'0 ;
[dout0..7].d = [data0..7] ;
```



CUPL – Codificador hexadecimal → 7segmentos

```
/* ********************** INPUT PINS ****************************/
PIN [1..4] = [d0..3] ;                  /* 4-bit Hexadecimal input */
PIN 5 = dot ;
/* ********************** OUTPUT PINS ***************************/
PIN [14..21] = [a,b,c,d,e,f,g,h] ;
                                /* 7-segments     - a -            */
                                /*  +dot        /     /           */
                                /*             f     b           */
                                /*           /- g -/             */
                                /*          e     c              */
                                /*         /     /               */
                                /*         - d -     h           */
FIELD number = [d0..3] ;
FIELD segments = [a,b,c,d,e,f,g] ;
h = dot ;
/* ***************************** BODY ****************************/
/* Primeira variante:
   definem-se as 7 funcoes de saida ah custa da tabela de verdade */
TABLE number => segments {
0=>'b'1111110;  4=>'b'0110011;  8=>'b'1111111;  C=>'b'1001110;
1=>'b'0110000;  5=>'b'1011011;  9=>'b'1111011;  D=>'b'0111101;
2=>'b'1101101;  6=>'b'1011111;  A=>'b'1110111;  E=>'b'1001111;
3=>'b'1111001;  7=>'b'1110000;  B=>'b'0011111;  F=>'b'1000111;
}
```

CUPL

Codificador hexadecimal → 7segmentos
(2ª variante)

```
$DEFINE zero    number:0
$DEFINE one     number:1
$DEFINE two     number:2
$DEFINE three   number:3
$DEFINE four    number:4
$DEFINE five    number:5
$DEFINE six     number:6
$DEFINE seven   number:7
$DEFINE eight   number:8
$DEFINE nine    number:9
$DEFINE Ah      number:A
$DEFINE Bh      number:B
$DEFINE Ch      number:C
$DEFINE Dh      number:D
$DEFINE Eh      number:E
$DEFINE Fh      number:F

/* Segunda variante:
     soma de termos produto, correspondentes aos valores da entrada
     (entendida como numero codificado em binario natural) */

a = zero # two # three # five # six # seven # eight # nine # Ah # Ch # Dh # Eh # Fh ;
b = zero # one # two # three # four # seven # eight # nine # Ah # Dh ;
c = zero # one # three # four # five # six # seven # eight # nine # Ah # Bh # Dh ;
d = zero # two # three # five # six # eight # nine # Bh # Ch # Dh # Eh ;
e = zero # two # six # eight # Ah # Bh # Ch # Dh # Eh # Fh ;
f = zero # four # five # six # eight # nine # Ah # Bh # Ch # Eh # Fh ;
g = two # three # four # five # six # eight # nine # Ah # Bh # Dh # Eh # Fh ;
```

CUPL – Codificador hexadecimal → 7segmentos (3ª variante)

```
/* Terceira variante:
           identica ah segunda */

$DEFINE n  number

a = n:0 # n:2 # n:3 # n:5 # n:6 # n:7 # n:8 # n:9 # n:A # n:C # n:d # n:E # n:F ;
b = n:0 # n:1 # n:2 # n:3 # n:4 # n:7 # n:8 # n:9 # n:A # n:d ;
c = n:0 # n:1 # n:3 # n:4 # n:5 # n:6 # n:7 # n:8 # n:9 # n:A # n:b # n:d ;
d = n:0 # n:2 # n:3 # n:5 # n:6 # n:8 # n:9 # n:b # n:C # n:d # n:E ;
e = n:0 # n:2 # n:6 # n:8 # n:A # n:b # n:C # n:d # n:E # n:F ;
f = n:0 # n:4 # n:5 # n:6 # n:8 # n:9 # n:A # n:b # n:C # n:E # n:F ;
g = n:2 # n:3 # n:4 # n:5 # n:6 # n:8 # n:9 # n:A # n:b # n:d # n:E # n:F ;
```

CUPL – Contador configurável módulo 2, 3 ou 4

```
/* ********** BODY **********/
Q0.ar = 'b'0 ;
Q0.sp = 'b'0 ;
Q1.ar = 'b'0 ;
Q1.sp = 'b'0 ;

SEQUENCE [Q1,Q0] {
PRESENT 2
          IF d  OUT s ;
          NEXT 3 ;
PRESENT 3
          NEXT 1 ;
PRESENT 1
          OUT s ;
          NEXT 0 ;
PRESENT 0
          IF p  NEXT 0 ;
          IF !p & !g  NEXT 3 ;
          DEFAULT  NEXT 2 ;
}
```