

# Lógica e Sistemas Digitais - 3

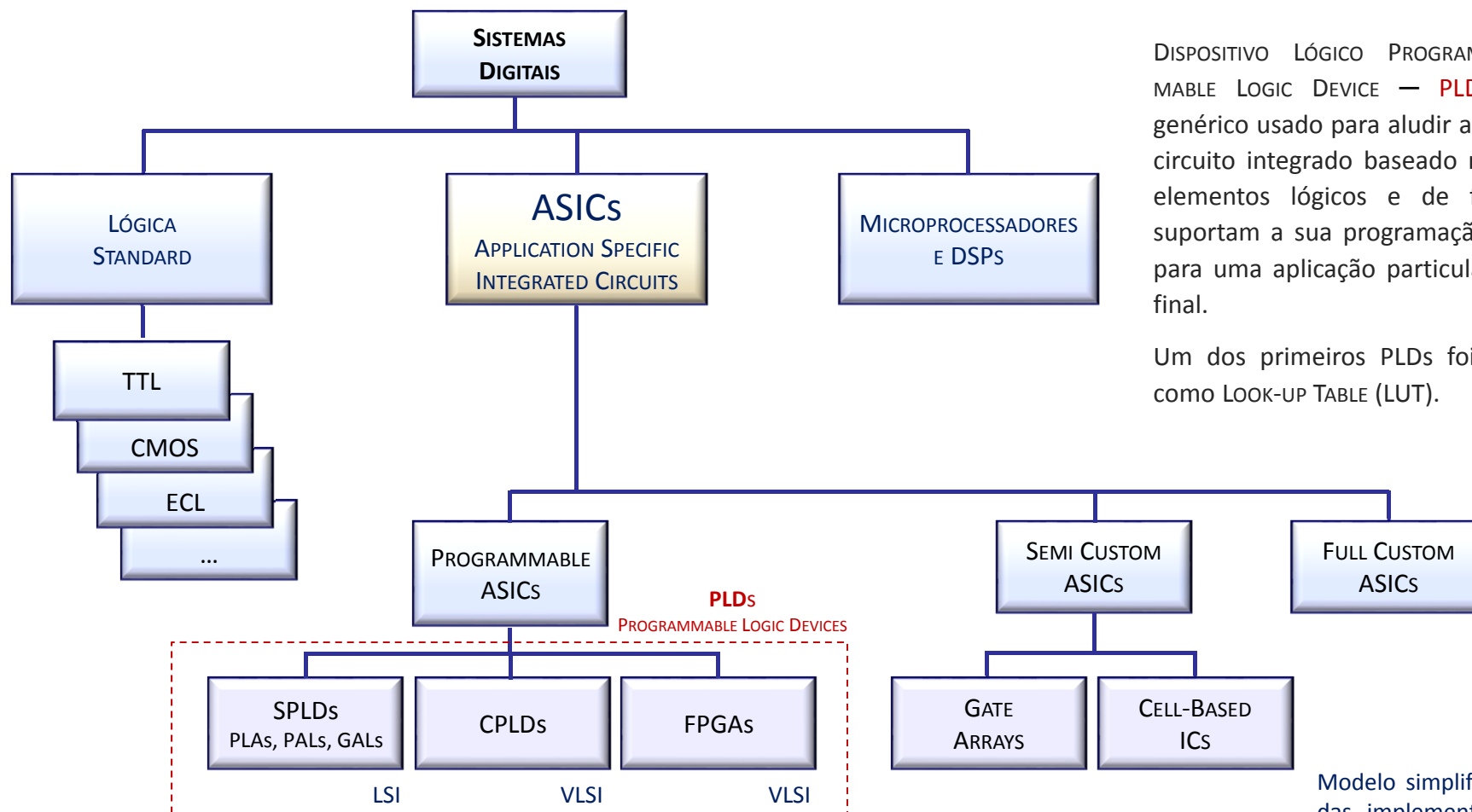
## ROM - PAL - PLA

ISEL

Departamento de Engenharia de Electrónica  
e Telecomunicações e de Computadores  
Lisboa

Mário Araújo

2016-1



DISPOSITIVO LÓGICO PROGRAMÁVEL (PROGRAMMABLE LOGIC DEVICE — **PLD**) é um termo genérico usado para aludir a qualquer tipo de circuito integrado baseado num conjunto de elementos lógicos e de ferramentas que suportam a sua programação e configuração para uma aplicação particular pelo utilizador final.

Um dos primeiros PLDs foi a PROM usada como LOOK-UP TABLE (LUT).

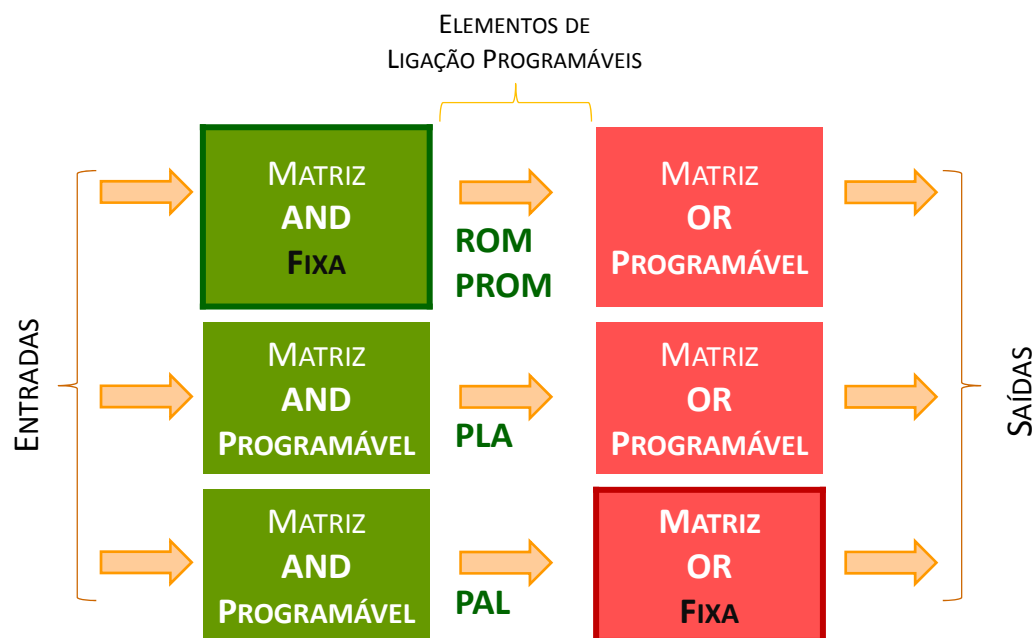
Modelo simplificado da hierarquia das implementações lógicas com destaque para as classes de DISPOSITIVOS LÓGICOS PROGRAMÁVEIS (PLDs).

PLD PROGRAMMABLE LOGIC DEVICE  
 PLA PROGRAMMABLE LOGIC ARRAY  
 PAL PROGRAMMABLE ARRAY LOGIC  
 GAL GENERIC ARRAY LOGIC  
 PROM PROGRAMMABLE READ ONLY MEMORY

SPLD SIMPLE PROGRAMMABLE LOGICAL DEVICES  
 CPLD COMPLEX PROGRAMMABLE LOGICAL DEVICES  
 FPGA FIELD PROGRAMMABLE GATE ARRAY  
 FPLA FIELD PROGRAMMABLE LOGIC ARRAY

Vimos nos capítulos anteriores que uma função lógica pode ser implementada como uma soma de termos produto (SOP).

Referem-se agora os dispositivos lógicos programáveis (PROGRAMMABLE LOGIC DEVICES – PLDs) que permitem a implementação de várias funções lógicas combinatórias por configuração das suas ligações internas programáveis. Geram **termos produto** através da programação de portas AND e realizam a sua soma através de portas OR. Há 3 tipos de PLDs que diferem na organização e capacidade de programação das matrizes AND e OR que possuem, como indicado na Fig.:



Tipos de PLDs e organização das matrizes AND e OR internas.

- A **ROM** (READ ONLY MEMORY) e **PROM** (PROGRAMMABLE READ ONLY MEMORY) implementam cada uma das saídas que possuem na forma canónica gerando internamente todos os mintermos das variáveis mesmo que não venham a ser utilizados. A programação corresponde à queima de fusíveis. As funções geradas ficam inalteráveis. A PROM difere da ROM pelo facto da programação ser exequível pelo programador.
- O **PLA** (PROGRAMMABLE LOGIC ARRAY) é mais flexível pois a matriz AND é programável e pode gerar apenas os termos produtos necessários (quer sejam mintermos ou não), a ser usados (mas não partilhados) pelas portas OR para geração de cada saída na forma Soma de Produtos.
- Na **PAL** (PROGRAMMABLE ARRAY LOGIC) apenas a malha AND é programável, sendo a matriz OR fixa. A programação corresponde a gerar os termos AND necessários por combinação das variáveis de entrada. Os diversos termos AND não são partilhados pelas saídas. As funções de saída exprimem-se em formas AND-OR simplificadas. É uma estrutura genérica onde se evita o desperdício característico das PROM. A PAL é normalmente mais rápida que uma PLA. As saídas podem ser 'registadas', facilitando a realização de circuitos sequenciais (como se verá no Capítulo-7).
- Os PLAs e as PALs são caracterizadas por uma estrutura em bloco, única, em encapsulamentos de 20 a 44 pinos, densidades médias entre as 100 e as poucas centenas de portas lógicas, elevadas velocidades de propagação e baixo custo. São dispositivos normalmente designados por SPLDs (SIMPLE PROGRAMMABLE LOGIC DEVICES).

## ROM E PROM

3-4

As memórias ROM (READ ONLY MEMORY) e PROM (PROGRAMMABLE READ ONLY MEMORY) em tecnologia bipolar, só de leitura e programáveis uma única vez (ONE-TIME PROGRAMMABLE – OTP), foram os primeiros dispositivos PLD que permitiram ao utilizador a implementação de funções lógicas por transposição directa das tabelas de verdade respectivas. Uma memória ROM é um circuito combinatório com  $n$  entradas e  $m$  saídas que pode ser vista como um dispositivo que guarda a tabela de verdade com  $2^n$  linhas duma função lógica combinatória com  $n$  entradas e  $m$  saídas (no exemplo ao lado  $n=3$  e  $m=4$ ).

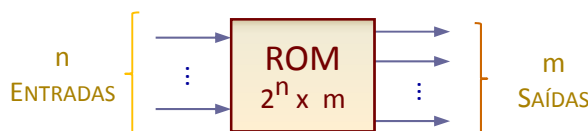
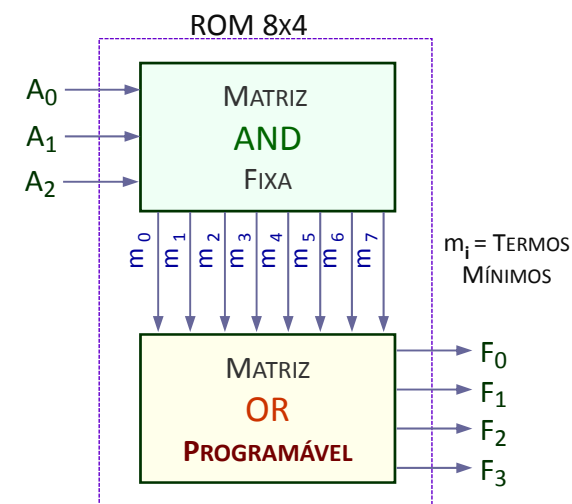


Diagrama de blocos de uma ROM com  $n$  entradas,  $m$  saídas e  $2^n$  termos produto.

Nas memórias ROM (e PROM) apenas a malha OR é programável. A malha AND fixa (inalterável) funciona como circuito de descodificação: para  $n$  entradas realiza todos os  $2^n$  termos mínimos em função das  $n$  variáveis de entrada. As funções de saída exprimem-se na forma canónica AND-OR. A programação corresponde a seleccionar os termos mínimos que formam cada uma das saídas. Como as funções lógicas raramente requerem mais do que alguns termos produto, ocorre desperdício por não se gerarem apenas os termos mínimos necessários o que torna as ROM e PROM ineficientes na realização de circuitos lógicos.

Para mais, a simples adição de uma entrada provoca a duplicação das portas AND da matriz fixa e também do número de fusíveis necessários na matriz OR programável.



Estrutura interna da ROM 8x4 assinalando a matriz AND fixa e a matriz OR programável evidenciando-se os 8 termos mínimos gerados.

	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>
m <sub>0</sub>	0	0	0	1	0	1	1
m <sub>1</sub>	0	0	1	0	0	0	0
m <sub>2</sub>	0	1	0	1	1	0	1
m <sub>3</sub>	0	1	1	0	0	0	0
m <sub>4</sub>	1	0	0	1	1	1	0
m <sub>5</sub>	1	0	1	0	1	0	1
m <sub>6</sub>	1	1	0	0	0	0	0
m <sub>7</sub>	1	1	1	0	0	0	0

Tabela de verdade das 4 funções lógicas F<sub>0</sub>, F<sub>1</sub>, F<sub>2</sub> e F<sub>3</sub> de 3 variáveis A<sub>0</sub>, A<sub>1</sub>, e A<sub>2</sub> a implementar na ROM 8x4 evidenciando-se os 8 termos mínimos gerados.

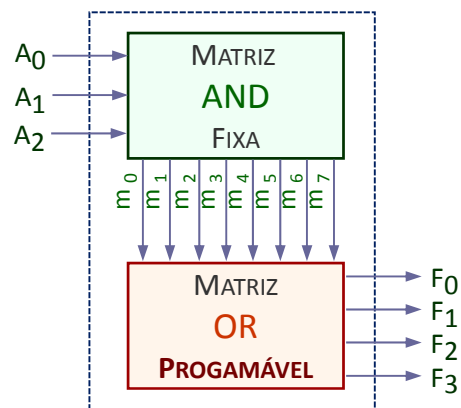


# ROM – IMPLEMENTAÇÃO DE 4 FUNÇÕES DE 3 VARIÁVEIS NUMA ROM

3-5



Diagrama de blocos de uma ROM 8x4.



Estrutura interna da ROM assinalando a matriz AND e a matriz OR.

$$F_0 = m_0 + m_2 + m_4$$

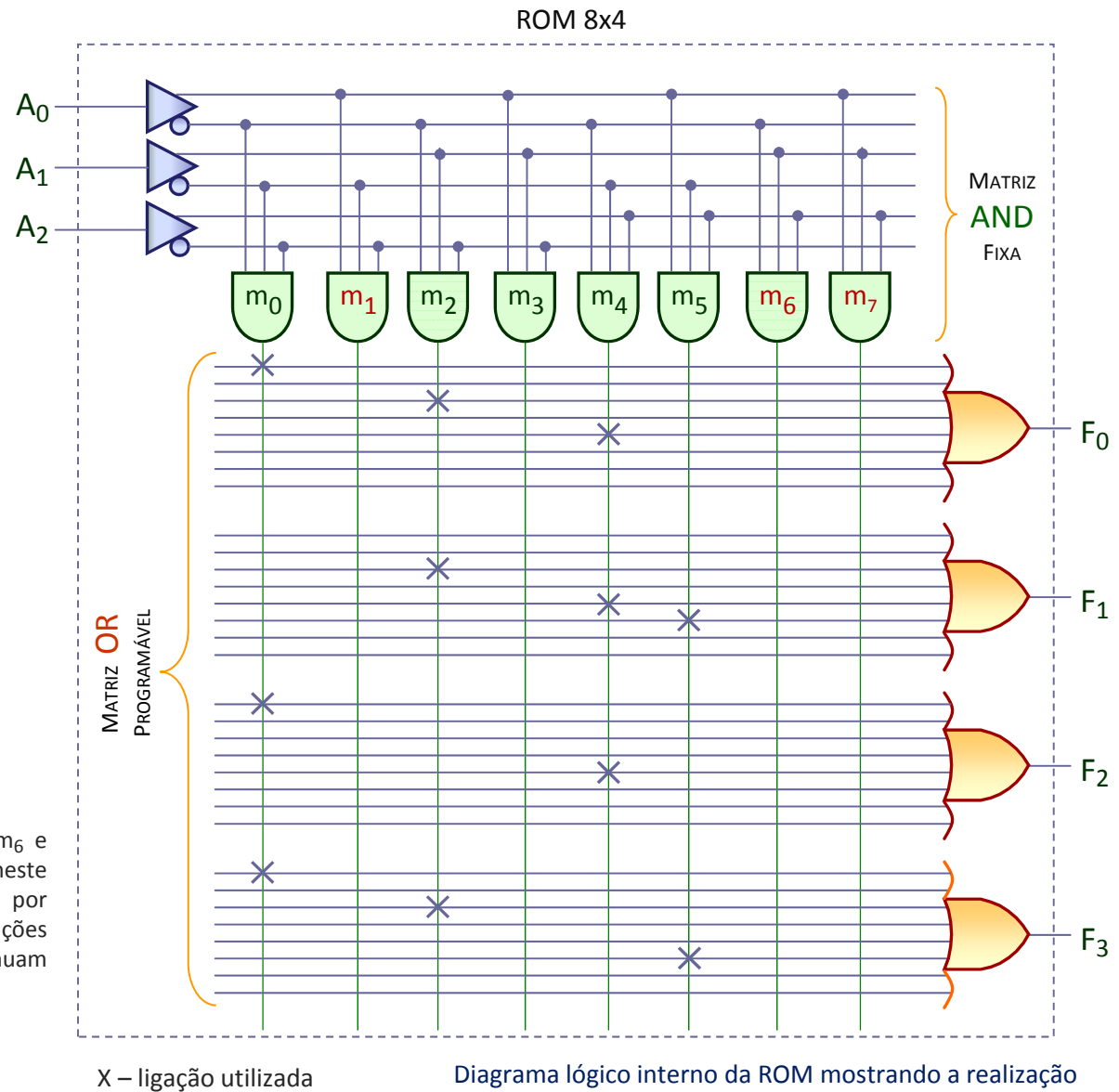
$$F_1 = m_2 + m_4 + m_5$$

$$F_2 = m_0 + m_4$$

$$F_3 = m_0 + m_2 + m_5$$

Os mintermos  $m_1$ ,  $m_6$  e  $m_7$  não são neste exemplo utilizados por nenhuma das 4 funções de saída mas continuam a ser gerados.

Expressões não simplificadas das 4 funções  $F_0$  a  $F_3$  a realizar em ROM.



X – ligação utilizada

Diagrama lógico interno da ROM mostrando a realização da matriz AND e da matriz OR com portas lógicas.

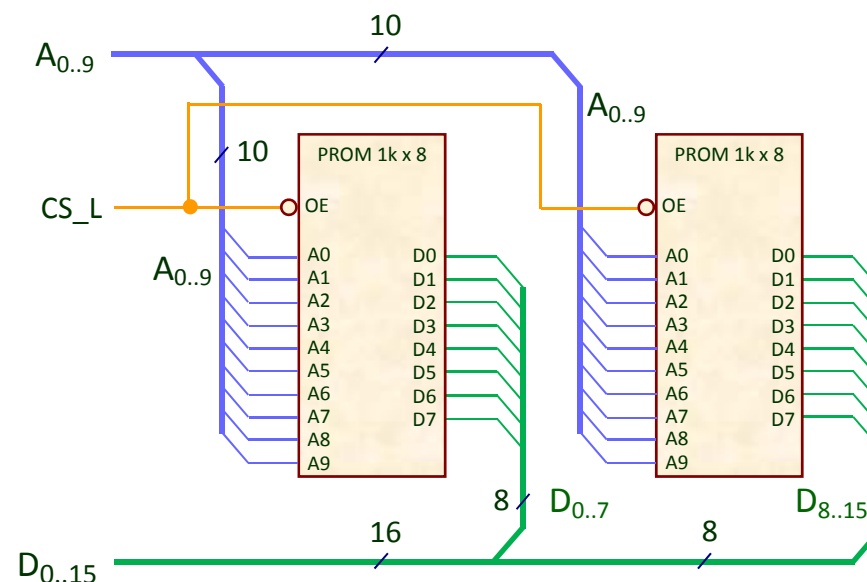


Viu-se que os módulos de memória ROM e PROM podem ser programados para implementar funções combinatórias. Ver-se-á agora como pode ser feita a sua utilização como blocos de memória.

Cada um dos módulos da Fig. ao lado possui como entradas um conjunto de 10 linhas de endereço (ADDRESS LINES A0..9), 8 linhas de dados (DATA LINES D0..D7) e 1 linha de controlo. Isto permite a cada um armazenar 1K palavras (WORDS) de 8 bits.

São necessários dois destes módulos de memória ligados em paralelo para perfazerem um bloco de memória PROM 1K x 16. Este bloco armazena 1K palavras de 16 bits (bits D0..D15), sendo que cada um dos módulos que o constitui fica com metade do número de bits (8) de cada palavra – o módulo da esquerda fica com os bits D0..D7 e o módulo da direita com os bits D8-D15 como indicado. Utilizando vários destes módulos é possível implementar blocos de memória com uma dimensão de palavra múltipla da do módulo original.

A linha de controlo OE – OUTPUT ENABLE – consiste num sinal de activação das saídas de dados. O sinal CS\_L (CHIP SELECT do tipo ACTIVE Low) ligado à entrada OE permite quando a 1 desactivar o acesso à memória colocando as saídas de dados (as DATA LINES D0..D7 de cada módulo) em alta impedância. Desta forma, será possível ligar vários blocos a um barramento único, tendo de se garantir que apenas um dos blocos está activo de cada vez.



Ligação das entradas e saídas de módulos de memória PROM 1Kx8 para implementação de um bloco PROM 1Kx16 com o dobro do comprimento das palavras.

Um PLA (PROGRAMMABLE LOGIC ARRAY) é um dispositivo que possui circuitos combinatórios AND-OR de 2 níveis e pode ser programado para realizar uma qualquer expressão lógica em termos de Soma de Produtos (SOP).

Um PLA é limitado:

- pelo número de entradas **n**
- pelo número de saídas **m**
- pelo número de termos produto **p**.

Um PLA referido como **n x m** com **p termos produto**, ou **n x m x p** (geralmente com  $p \ll 2^n$ ) possui **p** portas AND de **2n** entradas e **m** portas OR de **p** entradas.

Cada entrada está ligada a um buffer que produz duas saídas : uma complementada, outra não-complementada.

O PLA possibilita a geração de uma qualquer função cuja implementação não exceda o número de termos produto disponível. Se depois da minimização o número de termos produtos distintos ainda exceder **p** então um PLA de capacidade superior será requerido.

Sendo a matriz AND programável, a adição de uma entrada efectua-se sem que ocorra a duplicação do tamanho de qualquer uma das matrizes. A programação consiste em estabelecer as ligações necessárias, através de fusíveis. As saídas exprimem-se por formas AND-OR que não têm de ser canónicas.

Estes dispositivos tiveram um sucesso limitado pelo facto de serem lentos por apresentarem duas matrizes programáveis em série – num PLA ambas as malhas AND e OR são programáveis.

Os PLA são programados na fonte. Se forem programados pelo utilizador passam a designar-se por FPLA (FIELD PROGRAMMABLE LOGIC ARRAY).

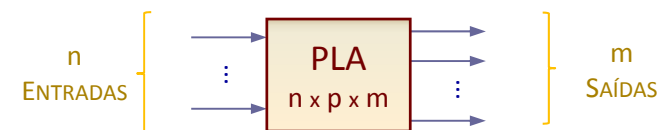
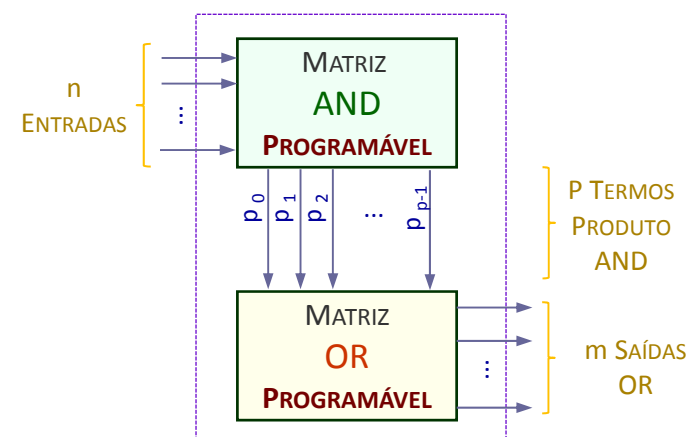


Diagrama de blocos de um PLA com **n** entradas, **m** saídas e **p** termos produto.



Estrutura interna de um PLA com **n** entradas, **m** saídas e **p** termos produto.

# PLA – IMPLEMENTAÇÃO DE 4 FUNÇÕES DE 3 VARIÁVEIS NUM PLA

3-8

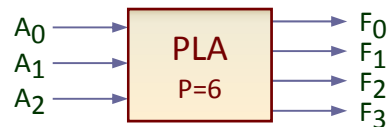
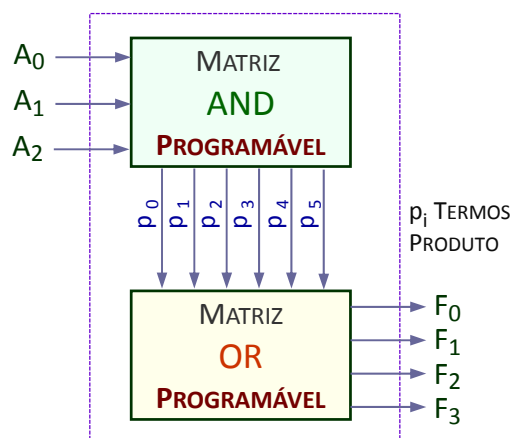


Diagrama de blocos de um PLA com 3 entradas, 4 saídas e 6 termos produto.



Estrutura interna do PLA assinalando a matriz AND e a matriz OR.

$$\begin{aligned} F_0 &= m_0 + m_2 + m_4 = p_1 + p_2 \\ F_1 &= m_2 + m_4 + m_5 = p_3 + p_4 \\ F_2 &= m_0 + m_4 = p_2 \\ F_3 &= m_0 + m_2 + m_5 = p_1 + p_5 \end{aligned}$$

Expressões não simplificadas (à esquerda) e simplificadas (à direita) das 4 funções F0 a F3 a realizar em PLA.

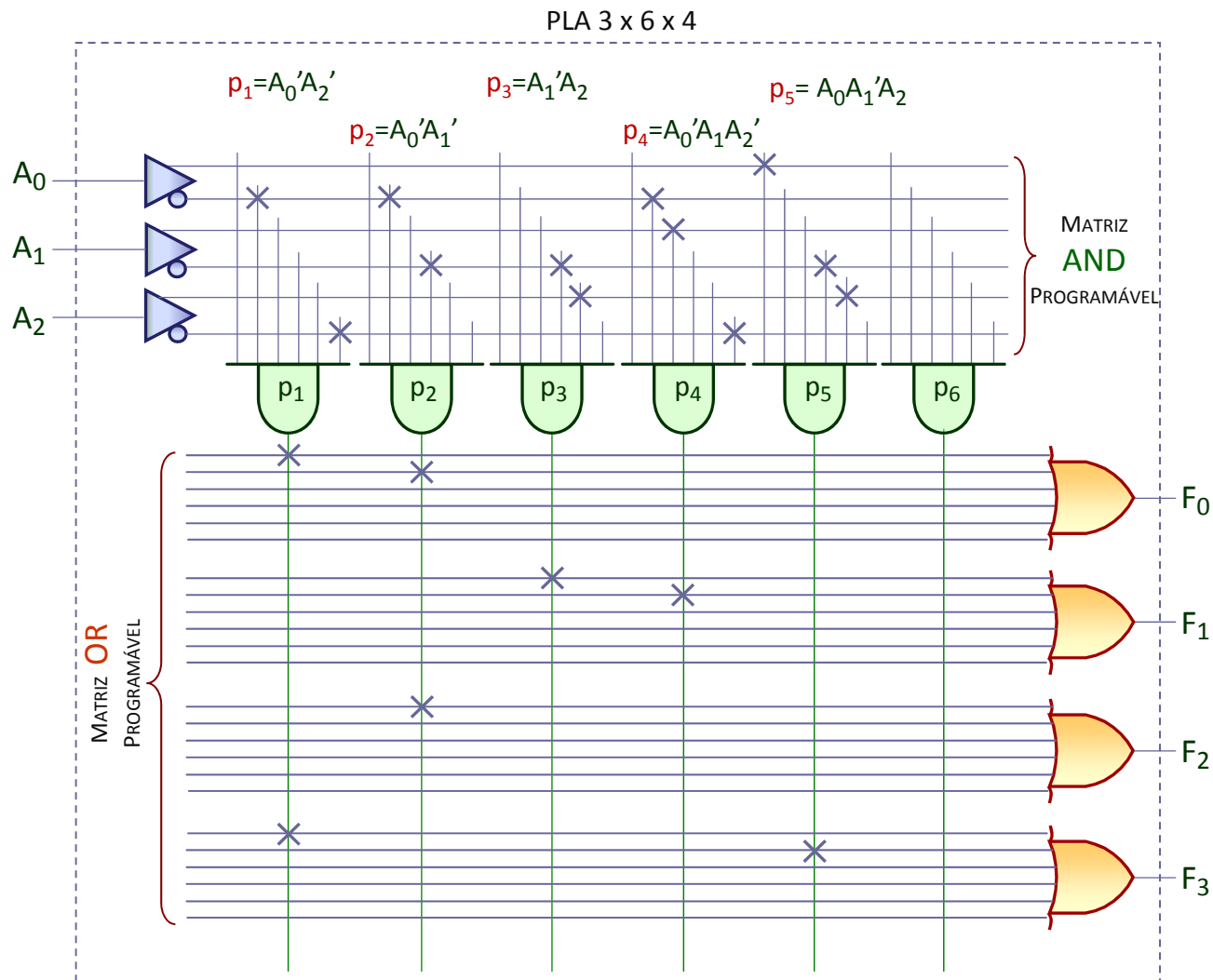


Diagrama lógico interno do PLA mostrando a realização da matriz AND e da matriz OR com portas lógicas.

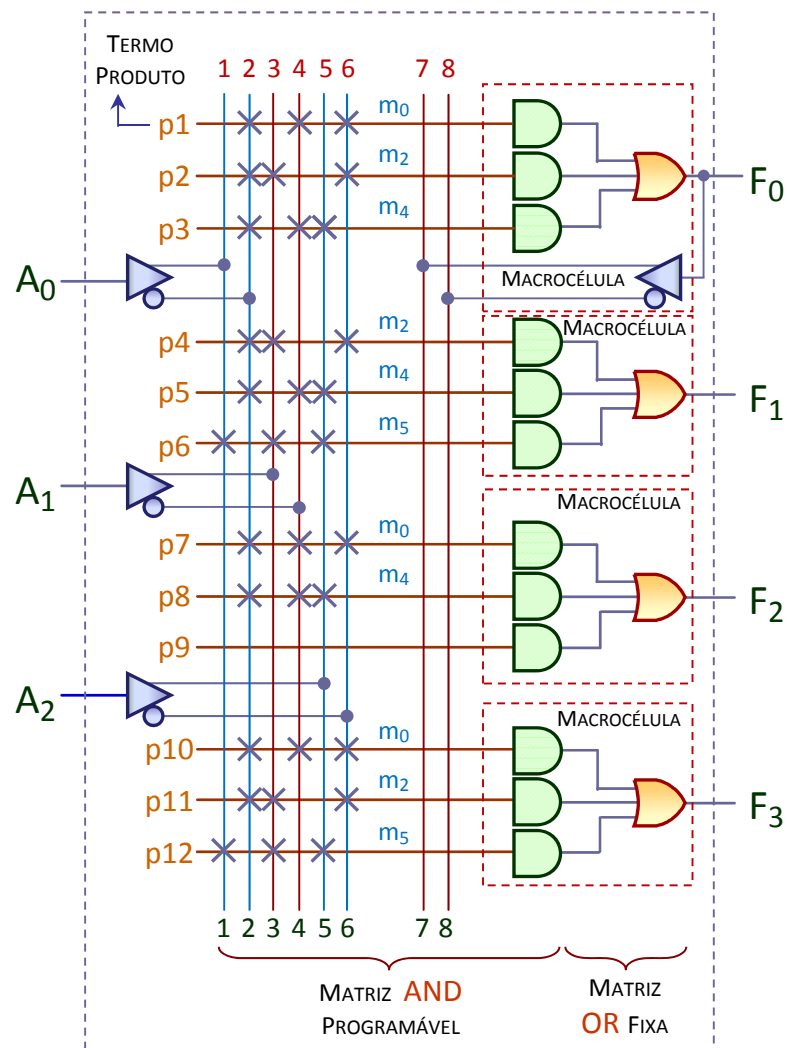
O termo p6 disponível neste PLA não é utilizado nesta aplicação. Também não foram utilizadas mais do que 2 das 6 entradas disponíveis em cada uma das 4 portas OR.





## PAL – IMPLEMENTAÇÃO DE 4 FUNÇÕES DE 3 VARIÁVEIS NUMA PAL (1)

3-9



Arquitetura interna da PAL mostrando a matriz programável AND e a matriz fixa OR.

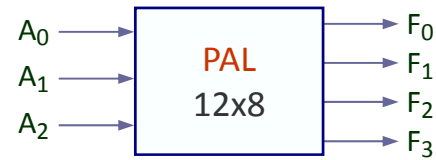


Diagrama de blocos de uma PAL com 3 entradas, 4 saídas e 12 termos produto.

$$\begin{aligned} F_0 &= m_0 + m_2 + m_4 \\ F_1 &= m_2 + m_4 + m_5 \\ F_2 &= m_0 + m_4 \\ F_3 &= m_0 + m_2 + m_5 \end{aligned}$$

Expressões não simplificadas das 4 funções F0 a F3 a realizar em PAL.

Para ilustrar a arquitectura típica de uma PAL considerou-se uma PAL hipotética com uma configuração de 3 entradas e 4 saídas, semelhante à da ROM e da PLA anteriores, com 12 termos produto, implementando as mesmas 4 funções já descritas.

A matriz de saída de qualquer PAL tem uma estrutura OR fixa não programável, o que torna a PAL mais fácil de fabricar e programar apesar da menor flexibilidade em termos de programação.

Na PAL representada cada uma das saídas é gerada por uma estrutura AND-OR que é idêntica para 3 das saídas F1, F2 e F3, e formada por três portas AND com entradas programáveis. Existe uma única saída F0 com realimentação na matriz de entrada através de um buffer (na forma complementada e não complementada). Nesta aplicação não foi feito uso desta realimentação.

Cada uma das portas AND possui assim 8 ligações de entrada programáveis das quais 6 provêm das 3 entradas A0, A1 e A2, e 2 dos sinais de realimentação da saída F0, tanto na forma direta como complementada.

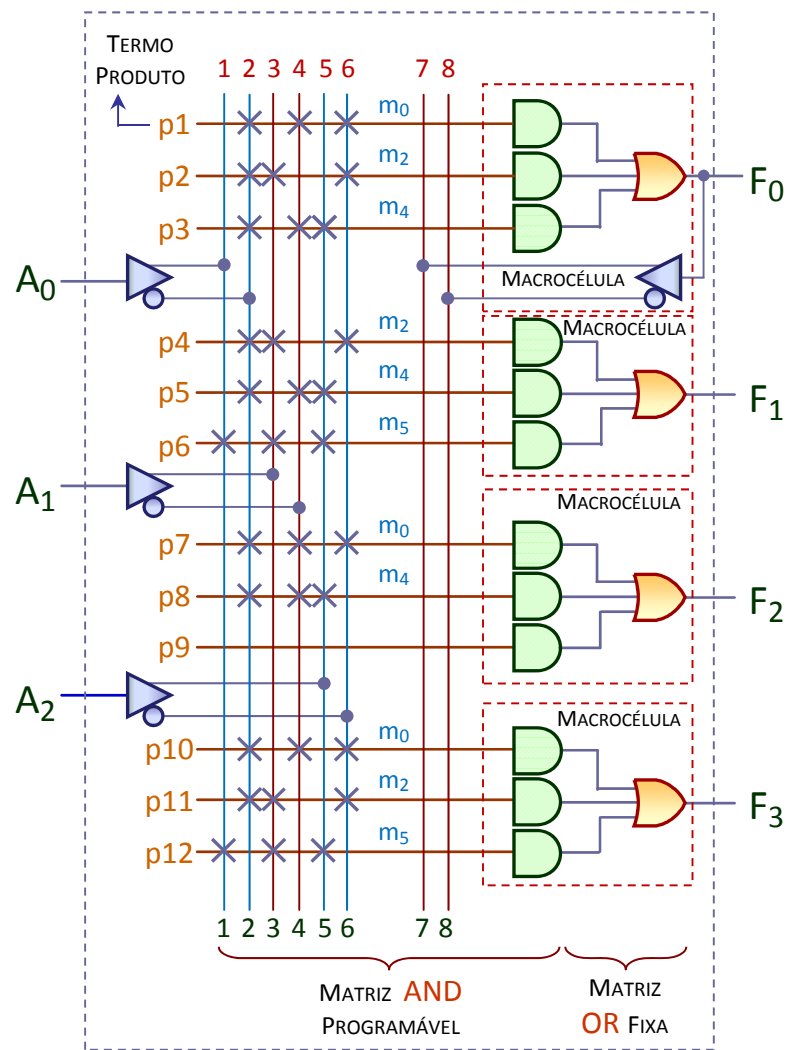
A lógica associada a cada uma das saídas – neste exemplo a estrutura AND-OR com três portas AND, ou esta estrutura adicionada do buffer de realimentação para o caso único da saída F0 – designa-se de CÉLULA de SAÍDA ou MACROCÉLULA (MACROCELL).

O número de entradas e de saídas, bem como a estrutura da MACROCÉLULA, são os principais parâmetros lógicos diferenciadores duma PAL.



## PAL – IMPLEMENTAÇÃO DE 4 FUNÇÕES DE 3 VARIÁVEIS NUMA PAL (2)

3-10



$$F_0 = m_0 + m_2 + m_4$$

$$F_1 = m_2 + m_4 + m_5$$

$$F_2 = m_0 + m_4$$

$$F_3 = m_0 + m_2 + m_5$$

Expressões não simplificadas das 4 funções F0 a F3 a implementar na PAL à esquerda.

$$F_0 = p_1 + p_2$$

$$F_1 = p_3 + p_4$$

$$F_2 = p_2$$

$$F_3 = p_1 + p_5$$

Expressões simplificadas das 4 funções F0 a F3 a implementar na PAL à direita.

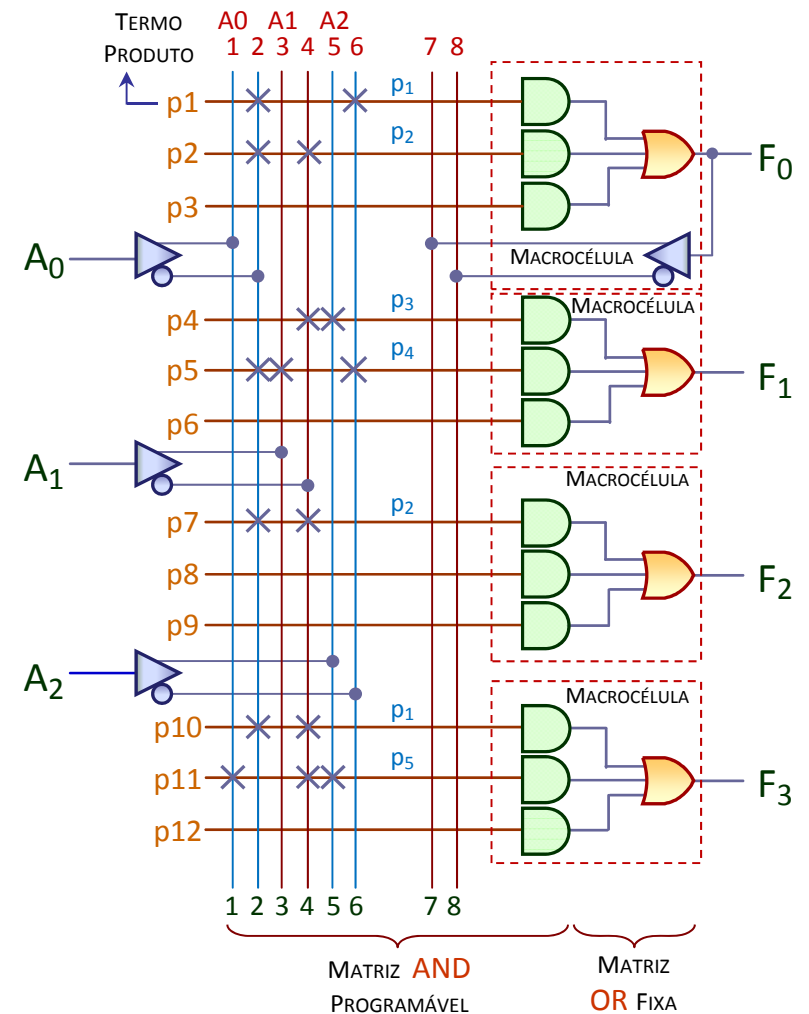
$$p_1 = A_0'A_2'$$

$$p_2 = A_0'A_1'$$

$$p_3 = A_1'A_2$$

$$p_4 = A_0'A_1A_2'$$

$$p_5 = A_0A_1'A_2$$

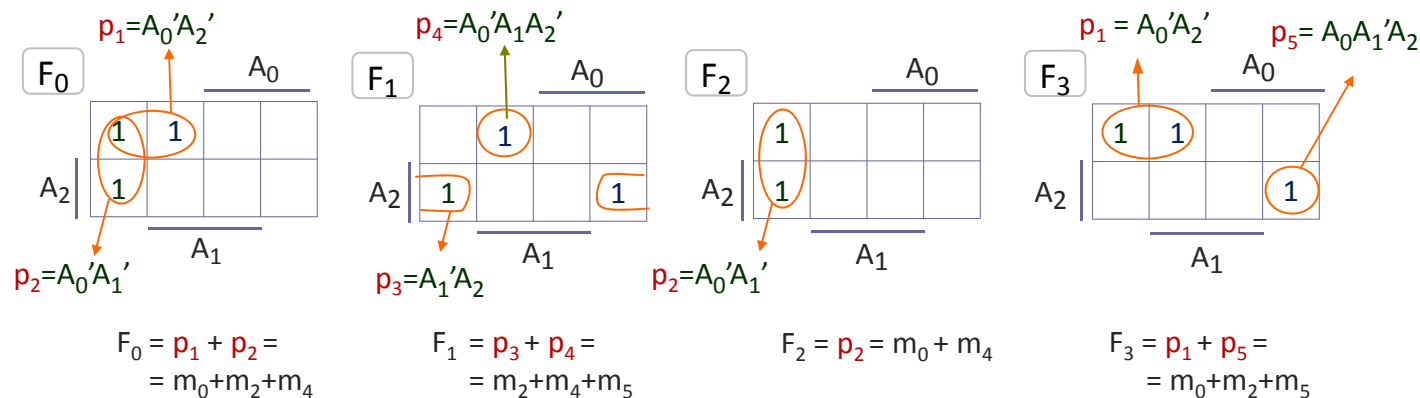


# IMPLEMENTAÇÃO DE 4 FUNÇÕES DE 3 VARIÁVEIS EM ROM, PLA E PAL

3-11

	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>
m <sub>0</sub>	0	0	0	1	0	1	1
m <sub>1</sub>	0	0	1	0	0	0	0
m <sub>2</sub>	0	1	0	1	1	0	1
m <sub>3</sub>	0	1	1	0	0	0	0
m <sub>4</sub>	1	0	0	1	1	1	0
m <sub>5</sub>	1	0	1	0	1	0	1
m <sub>6</sub>	1	1	0	0	0	0	0
m <sub>7</sub>	1	1	1	0	0	0	0

Tabela de verdade das 4 funções a implementar.



Mapas de Karnaugh para simplificação das 4 funções a implementar.

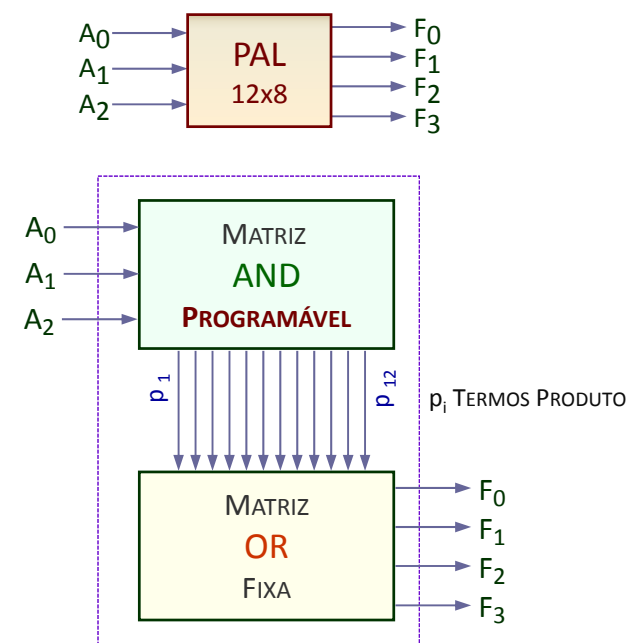
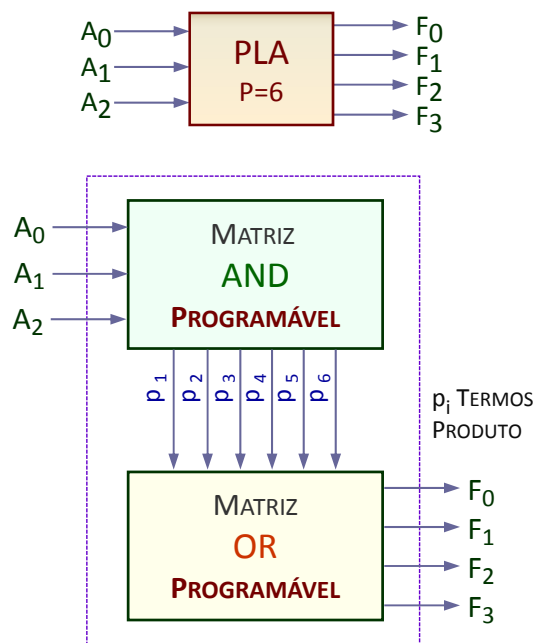
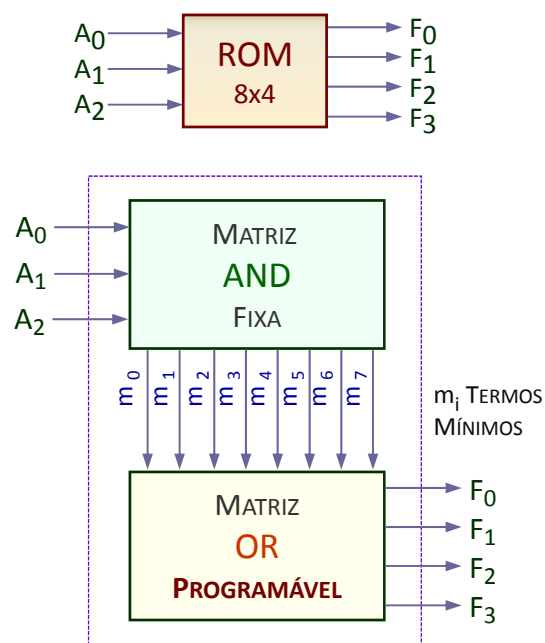
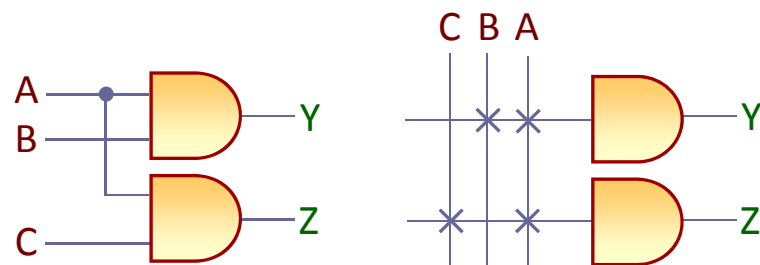
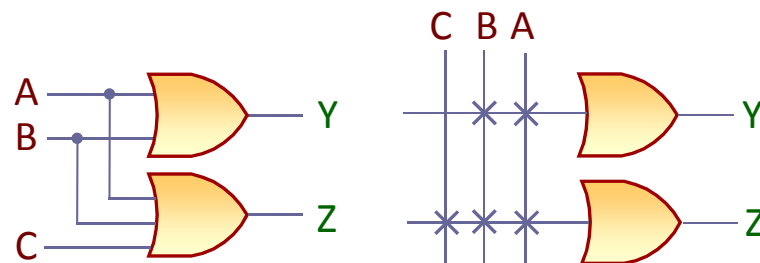


Diagrama de blocos (em cima), e estrutura interna (em baixo) da ROM, PLA e PAL assinalando o tipo de matriz AND e de matriz OR de cada uma.





Portas AND.

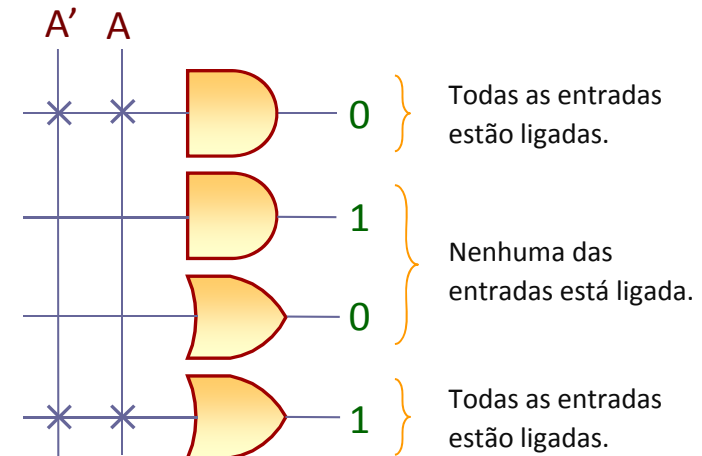


Portas OR.



Porta NOT.

Representação de 3 diagramas lógicos na forma convencional (à esquerda) e na forma simplificada (à direita) .



Convenções utilizadas na notação simplificada.

Sendo os PLDs dispositivos com muitos componentes e ligações a sua representação na forma tradicional seria complexa.

Surge a necessidade da utilização de uma notação simplificada em que se utilizam:

- linhas únicas para mostrar as entradas de cada porta AND;
- buffers de entrada e saída representados com duas saídas;
- cruzeiros (X) nas interseções que significam que as entradas respectivas são utilizadas (a ausência de cruzeiros significam que não são).

Estas convenções estão representadas na Fig. em cima.

A PAL16L8 possui:

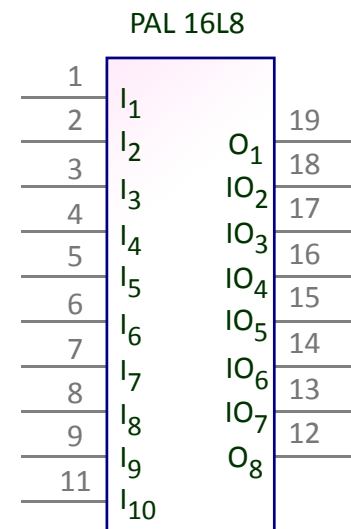
- 10 entradas primárias (pinos 1-9 e 11);
- 2 saídas dedicadas do tipo ACTIVE-LOW (pinos 12 e 19);
- 6 pinos bidirecionais (pinos 13-18) que podem ser programados como entrada ou como saída.

A PAL pode assim ter 16 entradas e 2 saídas, ou 10 entradas e 8 saídas, ou qualquer outra combinação entre estes limites.

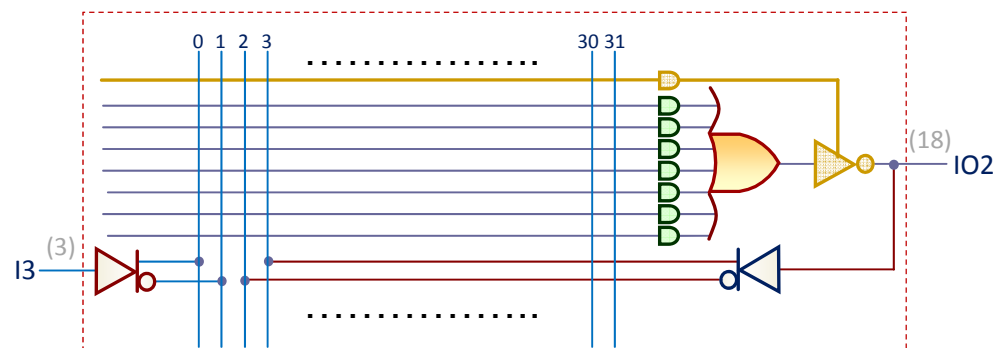
O encapsulamento do circuito tem apenas 20 pinos, incluindo os dois pinos de alimentação (VCC no pino 20 e GND no pino 10). A PAL16L8 possui uma matriz programável com 64 linhas (8 saídas OR de 8 portas AND cada) e 32 colunas (correspondentes ao conjunto das 16 entradas complementadas e não complementadas), perfazendo um total de 2048 (64x32) ligações programáveis.

Cada célula de saída é formada por:

- uma estrutura AND-OR com 7 portas AND de 32 entradas cada uma – as entradas não conectadas são interpretadas por cada porta AND como estando o valor lógico 1;
- uma porta do tipo BUFFER TRI-STATE inversor à saída de cada porta OR controlada por um termo produto – esta é a oitava porta AND à entrada de cada OR, também com 32 entradas – para que o BUFFER TRI-STATE inversor esteja sempre activo, basta não ligar nenhum sinal à sua porta AND de controlo;
- um buffer de realimentação da saída respectiva para a matriz de entrada, apenas existente para os 6 pinos bidirecionais (pinos 13-18).



Símbolo lógico tradicional da PAL 16L8 assinalando os pinos de entrada e saída.



Circuito lógico parcial da PAL16L8 mostrando uma entrada e uma das 6 saídas bidirecionais programável como entrada ou saída consoante o controle OUTPUT ENABLE do BUFFER TRI-STATE associado.

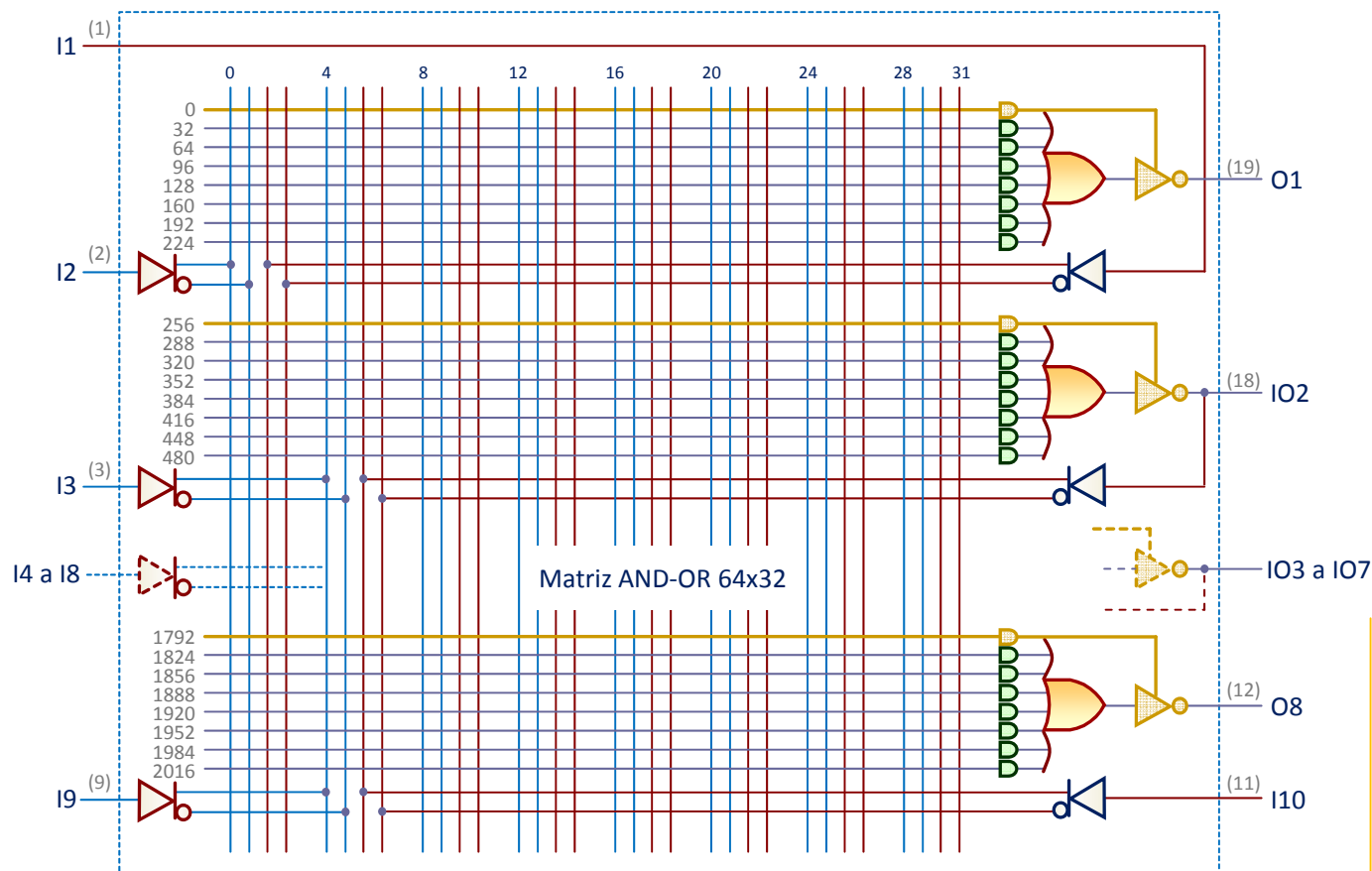


Diagrama lógico da PAL 16L8.

PAL 16L8

1	I <sub>1</sub>		19
2	I <sub>2</sub>	O <sub>1</sub>	18
3	I <sub>3</sub>	IO <sub>2</sub>	17
4	I <sub>4</sub>	IO <sub>3</sub>	16
5	I <sub>5</sub>	IO <sub>4</sub>	15
6	I <sub>6</sub>	IO <sub>5</sub>	14
7	I <sub>7</sub>	IO <sub>6</sub>	13
8	I <sub>8</sub>	IO <sub>7</sub>	12
9	I <sub>9</sub>	O <sub>8</sub>	
11	I <sub>10</sub>		

Símbolo lógico tradicional da PAL 16L8.

- 20 pinos
- matriz AND 64x32
- 7 Termos Produto (AND) por saída OR
- 10 entradas primárias (I1 a I10)
- 32 entradas por cada porta AND (16 entradas e os seus complementos)
- 2 saídas combinatórias dedicadas do tipo ACTIVE-LOW (O1 e O8)
- 6 saídas combinatórias bidirecionais (IO2 a IO7) disponíveis também como entradas
- 1 Termo Produto para controlo do BUFFER TRI-STATE inversor de cada saída.

Nas Linguagens de Descrição de Hardware **HDL** (HARDWARE DESCRIPTION LANGUAGES) o projetista cria um ficheiro de texto seguindo um conjunto de regras (sintaxe da linguagem), e usa um compilador para gerar os dados de programação do dispositivo PLD. As HDLs possuem alguma semelhança com as linguagens de programação, mas são especificamente orientadas para a descrição da estrutura e do comportamento do hardware. Podem representar diretamente equações booleanas, tabelas verdade e operações aritméticas.

No final da década de 1970 o uso de linguagens HDL era limitado e dedicado essencialmente à descrição de equações lógicas a serem implementadas em PLDs.

A partir de então a utilização das linguagens HDL acelerou-se com o crescimento e embaratecimento progressivo de PLDs, CPLDs e FPGAs. Nos dias de hoje elas são de longe o recurso mais comum na descrição de alto nível do projeto com PLDs – os diagramas esquemáticos são apenas utilizados na descrição das ligações ao nível das placas de circuitos.

A primeira linguagem HDL de uso comercial foi o:

- **PALASM** (de PAL Assembler)

desenvolvida no início da década de 1980 pelo fabricante Monolithic Memories Inc. (MMI) que foi a firma inventora da PAL (mais tarde absorvida pela Advanced Micro Devices – AMD). Seguiram-se as linguagens:

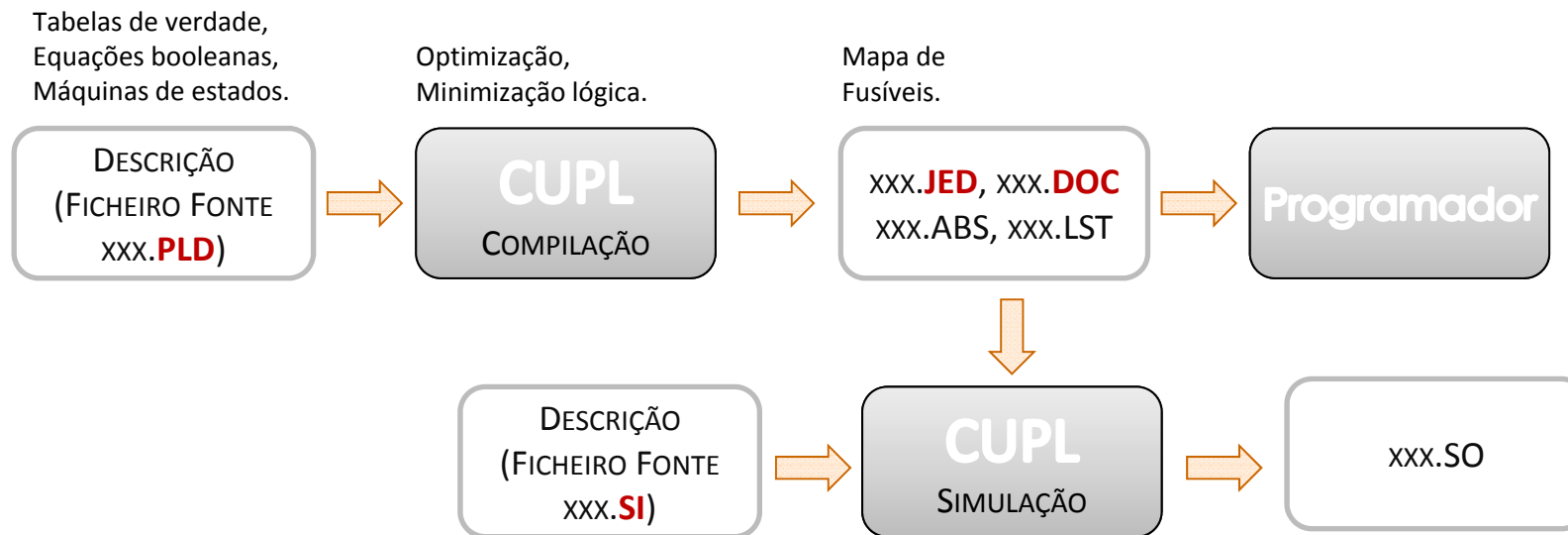
- **CUPL** (COMPILER UNIVERSAL FOR PROGRAMMABLE LOGIC), colocada no mercado em 1983 por uma empresa que mudou várias vezes de propriedade. A firma Atmel disponibiliza uma licença gratuita e permanente para utilização do compilador WinCupl para plataformas Windows.
- **ABEL** (ADVANCED BOOLEAN EXPRESSION LANGUAGE), da Data I/O.

Estas linguagens já eram mais evoluídas por incluírem minimização lógica e utilizarem descritores (STATEMENT CONSTRUCTS) de alto nível, com a possibilidade de serem derivadas equações lógicas a partir deles. Seguiram-se as linguagens:

- **VHDL** (VERY HIGH SPEED INTEGRATED CIRCUIT (**VHSIC**) HARDWARE DESCRIPTION LANGUAGE), patrocinada pelo USA DoD (DEPARTMENT OF DEFENSE) e mais tarde adotada como standard pelo IEEE (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS) em várias versões que lhe foram estendendo a funcionalidade.
- **VERILOG**, criada em 1984 pela Gateway Design Automation e standardizada pelo IEEE em várias versões, a primeira datada de 1995 e a última de 2001 (apesar de ter surgido uma alteração de expressão diminuta em 2005).

Ambas possuem suporte de codificação hierárquica e modular (como em C), e, se bem que tivessem nascido como linguagens de simulação, são hoje as linguagens HDL dominantes que suportam o desenvolvimento, a verificação, a síntese e o ensaio de projetos de hardware.





O WinCUPL é um ambiente de projecto de sistemas digitais de pequena dimensão. É possível lançar janelas de opções tanto para o compilador, como para o simulador, como para os dispositivos a programar e ainda para o ambiente WinCUPL.

O ficheiro de extensão PLD engloba a descrição funcional do circuito a implementar. A edição pode ser executada directamente em WinCUPL ou noutro editor de texto.

O ficheiros mais importantes gerados na fase de compilação são os ficheiros de extensão JED e DOC. O ficheiro JED é o ficheiro usado na programação do dispositivo e usa o formato ASCII.

A fase de compilação inclui os passos de optimização e de minimização lógica. A optimização tem o objectivo de ajustar melhor as expressões aos recursos do dispositivo alvo. Há vários métodos de optimização entre os quais Best for Polarity, Demorgan, Keep Xor Equations e P-Term Sharing. Para a minimização há 4 opções: 'Quick', 'Quine-McCluskey', 'Presto' e 'Expresso', existindo ainda a opção de não ser ativado nenhum método de minimização.

A simulação do circuito é um passo opcional. O simulador recebe um ficheiro com a extensão SI criado pelo utilizador, que contém os vectores de teste, e gera um ficheiro de saída com extensão SO com os resultados da simulação. Este ficheiro é usado para gerar os resultados em forma gráfica.



<b>PLD</b>	<p>CRIADO PELO <b>UTILIZADOR</b>.</p> <p>Contém todas as instruções lógicas necessárias à definição do circuito pretendido. É o programa propriamente dito, em CUPL.</p>
<b>DOC</b>	<p>GERADO PELO <b>COMPILADOR</b> DE CUPL.</p> <p>Contém todas equações lógicas expandidas, uma tabela de símbolos com as variáveis usadas, a utilização dos termos produto e informação sobre as ligações programáveis ('fusemap'). Reporta os erros encontrados durante a compilação, mesmo em caso de ocorrência de erro, com indicação do local aonde ocorreram.</p> <p>Descreve a correspondência entre o circuito pretendido e o dispositivo seleccionado.</p>
<b>ABS</b>	<p>GERADO PELO <b>COMPILADOR</b> DE CUPL.</p> <p>Ficheiro 'absoluto' necessário ao processo de simulação e destinado a alimentar o programa CSIM (módulo de simulação do CUPL).</p>
<b>LST</b>	<p>GERADO PELO <b>COMPILADOR</b> DE CUPL.</p> <p>Contém as linhas do programa original com a respectiva numeração.</p> <p>Os erros encontrados são indicados (com o símbolo ^ ) nas linhas onde ocorreram.</p>
<b>JED</b>	<p>GERADO PELO <b>COMPILADOR</b> DE CUPL.</p> <p>Ficheiro ASCII em formato JEDEC usado na programação do dispositivo PLD para seleccionar as ligações a queimar.</p>
<b>SI</b>	<p>CRIADO PELO <b>UTILIZADOR</b>.</p> <p>É o ficheiro de entradas para o simulador.</p> <p>Contém a lista dos <b>vectores de teste</b>.</p>
<b>SO</b>	<p>GERADO PELO <b>SIMULADOR</b> DE CUPL.</p> <p>Contém os resultados da simulação, incluindo eventuais erros.</p> <p>Utilizado para visualização gráfica dos resultados da simulação.</p>

Nomes das extensões (sufixos) e descrição dos ficheiros criados pelo utilizador do WinCupl ou resultantes da sua compilação.

## Table of Contents

### Section 1

Introduction to Programmable Logic .....	1-1
1.1 What is Programmable Logic? .....	1-1
1.2 Device Technologies and Packaging .....	1-6
1.3 Programming Logic Devices .....	1-7
1.4 Functionally Testing Logic Devices .....	1-7

### Section 2

Designing with the CUPL™ Language .....	2-1
2.1 Declaration of Language Elements .....	2-1
2.2 Usage of the Language Syntax .....	2-2
2.3 Advanced Language Syntax .....	2-14

### Section 3

Using the CUPL Compiler .....	3-1
3.1 About The Compiler .....	3-1
3.2 Output File Format Descriptions .....	3-6

### Section 4

CUPL Tutorial .....	4-1
4.1 Tutorial for Gates .....	4-1
4.2 Tutorial for COUNT10 .....	4-3
4.3 Tutorial for SQUARE.PLD .....	4-5

### Section 5

CUPL Software Features .....	5-1
5.1 CUPL - PALexpert .....	5-1
5.2 CUPL - PLDmaster .....	5-1
5.3 CUPL - Total Designer .....	5-1
5.4 CUPL - Total Designer VHDL .....	5-1
5.5 ONCUPL .....	5-2
5.6 Liaison .....	5-2
5.7 PLPartition .....	5-2
5.8 How to Contact Logical Devices.....	5-2



Para uma informação mais completa sugere-se a consulta do WINCUPL USER'S MANUAL do fabricante ATMEL do qual se apresenta o índice em cima .



APPEND	ASSEMBLY	ASSY	COMPANY
CONDITION	DATE	DEFAULT	DESIGNER
DEVICE	ELSE	FIELD	FLD
FORMAT	FUNCTION	FUSE	GROUP
IF	JUMP	LOC	LOCATION
MACRO	MIN	NAME	NODE
OUT	PARTNO	PIN	PINNODE
PRESENT	REV	REVISION	SEQUENCE
SEQUENCED	SEQUENCEJK	SEQUENCERS	SEQUENCET
TABLE			

Palavras reservadas do CUPL.

OPERADOR ARITMÉTICO	DESCRIÇÃO	EXEMPLO	PRECEDÊNCIA
**	Exponenciação	2**3	1
*	Multiplicação	2*i	2
/	Divisão	4/2	2
%	Módulo	9%8	2
+	Adição	2+4	3
-	Subtração	4-i	3

Operadores Aritméticos com exemplificação do seu uso e regras de precedência.

NOME DA BASE DE NUMERAÇÃO	BASE	PREFIXO	EXEMPLO	VALOR DECIMAL
Binária	2	'b' ou 'B'	'b'1101	13
			'b'1X11	com dc
Octal	8	'o' ou 'O'	'O'663	435
			'O'0X6	com dc
Octal (range)			'o'[300..477]	192..314
Decimal	10	'd' ou 'D'	'd'189	189
Hexadecimal	16	'h' ou 'H'	'h'BA	186
			'H'[3FXX..7FFF]	com dc

Bases de numeração utilizadas com exemplificação do prefixo, conversões e do uso de don't cares (dc).

&	#	(	)	-
	+	[	]	/
:	.	..	/*	*/
;	,	!	'	=
@	\$	^		

Símbolos reservados do CUPL.

Por omissão, os números são assumidos em base 16, mas quando se indicam índices a base é 10.

OPERADOR LOGICO	DESCRIÇÃO	EXEMPLO	PRECEDÊNCIA
!	NOT	!A	1
&	AND	A&B	2
#	OR	A#B	3
\$	XOR	A\$B	4

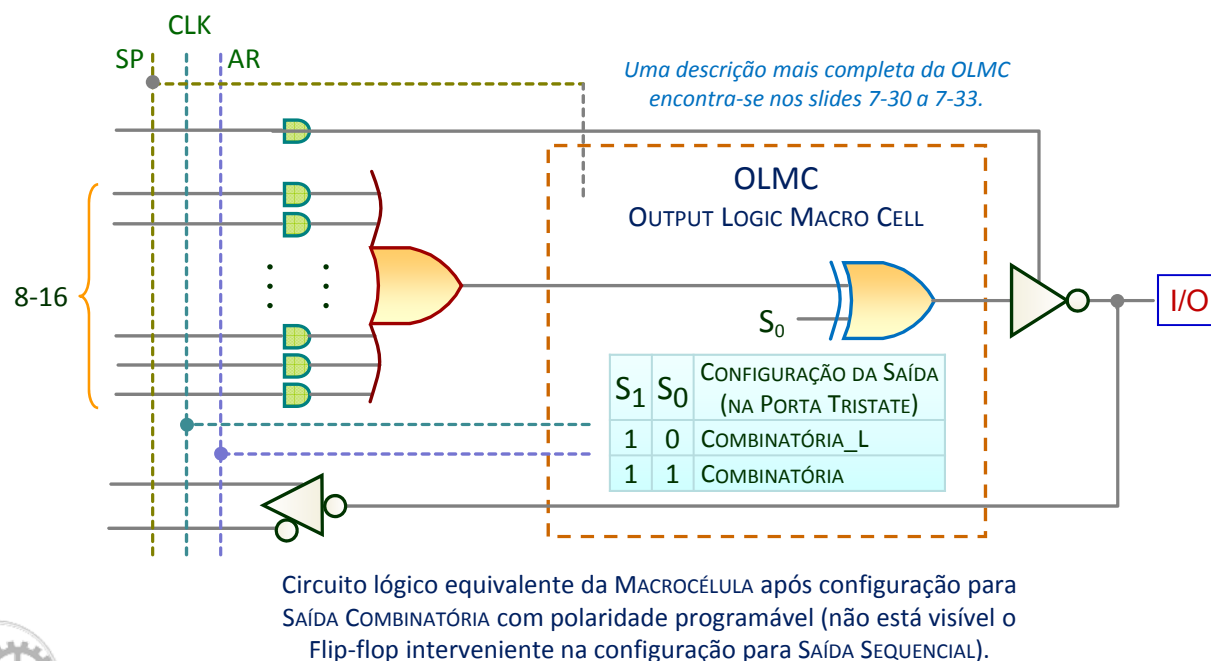
Operadores Lógicos com exemplificação do seu uso e regras de precedência.

A arquitectura da PAL 22V10 tornou-se numa norma de facto para o desenvolvimento posterior de outras PALs. A principal diferença entre a arquitectura da PAL 22V10 e das PAL suas antecessoras reside no conjunto de recursos disponíveis em cada uma das saídas, que constitui um bloco designado pelo termo de MACROCÉLULA (do original OLMC—OUTPUT LOGIC MACRO CELL).

Cada uma das 10 MACROCÉLULAS permite:

- a programação da polaridade do sinal de saída;
- a colocação da saída no estado de alta ou baixa impedância e
- a inclusão de um flip-flop no percurso de saída (não representado na Fig. em baixo).

A saída pode ainda ser realimentada, reaparecendo como entrada da matriz AND. A realimentação combinatória é feita a partir da saída do BUFFER TRI-STATE inversor, e só é possível se o pino I/O correspondente for configurado como saída.



PAL 22V10

1	CLK/I <sub>1</sub>	
2	I <sub>2</sub>	IO <sub>1</sub> 23
3	I <sub>3</sub>	IO <sub>2</sub> 22
4	I <sub>4</sub>	IO <sub>3</sub> 21
5	I <sub>5</sub>	IO <sub>4</sub> 20
6	I <sub>6</sub>	IO <sub>5</sub> 19
7	I <sub>7</sub>	IO <sub>6</sub> 18
8	I <sub>8</sub>	IO <sub>7</sub> 17
9	I <sub>9</sub>	IO <sub>8</sub> 16
10	I <sub>10</sub>	IO <sub>9</sub> 15
11	I <sub>11</sub>	IO <sub>10</sub> 14
13	I <sub>12</sub>	

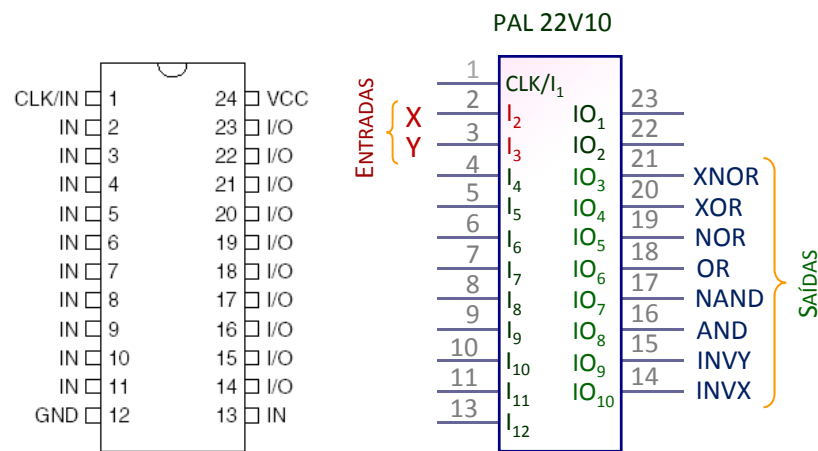
Símbolo lógico tradicional da PAL 22V10 assinalando os 12 pinos de entrada e saída.

O número de termos produto por cada uma das 10 Macro células da 22V10 (de 24 pinos) é variável de 8 a 16 como indicado:

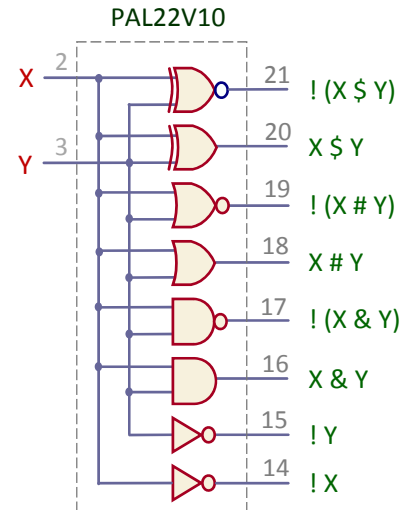
- Pino 1 – Clock e Entrada
- Pinos 2 a 11 e 13 – Entradas (12 no total com o Clock)
- Pinos 14, 23 – IO Programável com 8 Termos Produto
- Pinos 15, 22 – IO Programável com 10 Termos Produto
- Pinos 16, 21 – IO Programável com 12 Termos Produto
- Pinos 17, 20 – IO Programável com 14 Termos Produto
- Pinos 18, 19 – IO Programável com 16 Termos Produto.

## REALIZAÇÃO DE PORTAS SIMPLES NUMA PAL 22V10

3-20



Pinout e símbolo lógico da PAL assinalando os pinos de entrada e saída utilizados.



Exemplo de implementação de portas simples de 2 entradas numa PAL combinatória 22V10: funcionalidade interna da PAL.

O quadro ao lado mostra parcialmente a estrutura do ficheiro PLD a ser editado, do qual constam várias partes:

- o cabeçalho (omisso neste desenho);
- as declarações respeitantes à identificação das variáveis de entrada e de saída;
- o corpo de programa com a descrição lógica feita pelas equações.

Uma variável é uma letra seguida de letra ou dígitos. As variáveis diferem entre minúsculas e maiúsculas e não podem nem conter espaços nem coincidir com palavras-chave. Uma variável indexada é terminada entre 0 e 31.

*/\* Inputs: definição de entradas \*/*

PIN 2 = x;

PIN 3 = y;

*/\* Outputs: definição das saídas \*/*

PIN 14 = invx ;

PIN 15 = invy ;

PIN 16 = and ;

PIN 17 = nand ;

PIN 18 = or ;

PIN 19 = nor ;

PIN 20 = xor ;

PIN 21 = xnor ;

*/\* Expressão de portas lógicas em CUPL\*/*

invx = !x; */\* inverters \*/*

invy = !y;

and = x & y; */\* and gate \*/*

nand = ! (x & y); */\* nand gate \*/*

or = x # y; */\* or gate \*/*

nor = ! (x # y); */\* nor gate \*/*

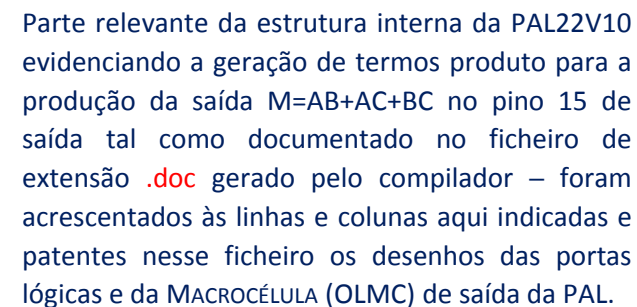
xor = x \$ y; */\* exclusive or gate \*/*

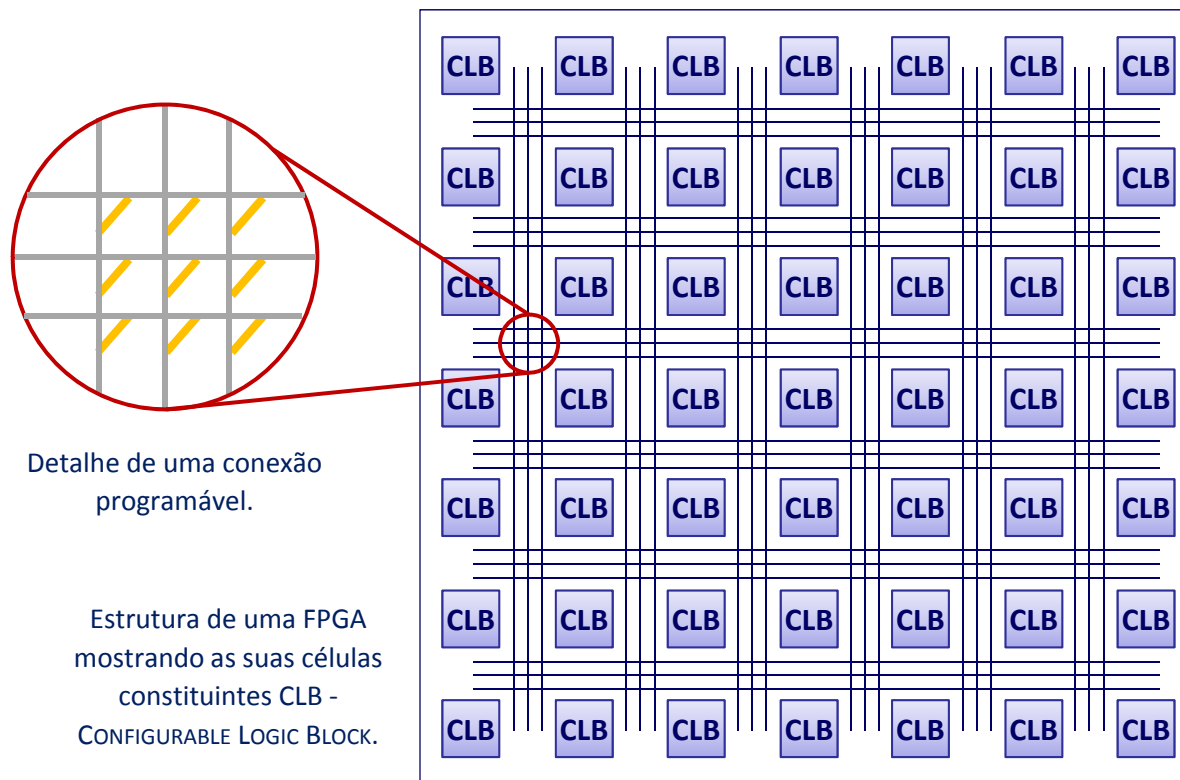
xnor = ! (x \$ y); */\* exclusive nor gate \*/*

Troço do código CUPL da PAL combinatória.









Os dispositivos SPLDs e CPLDs permitem a implementação de uma grande variedade de circuitos lógicos de pequena dimensão que podem ser acomodados num único chip – normalmente até 32 entradas e saídas para as PLAs e PALs, e de valor superior para os CPLDs. Para se implementarem circuitos lógicos de maior dimensão utiliza-se de um tipo diferente de chip designado de FPGA (FIELD-PROGRAMMABLE GATE ARRAY).

Enquanto que os CPLDs possuem planos de portas AND-OR semelhantes aos PLAs e um conjunto grande de interligações contínuas e pré-definidas, nos FPGAs a lógica é implementada empregando múltiplos planos e portas com reduzido número de

entradas, o que leva ao aumento da sua densidade lógica. As FPGAs são indicadas para circuitos com mais de 1 milhão de transistores.

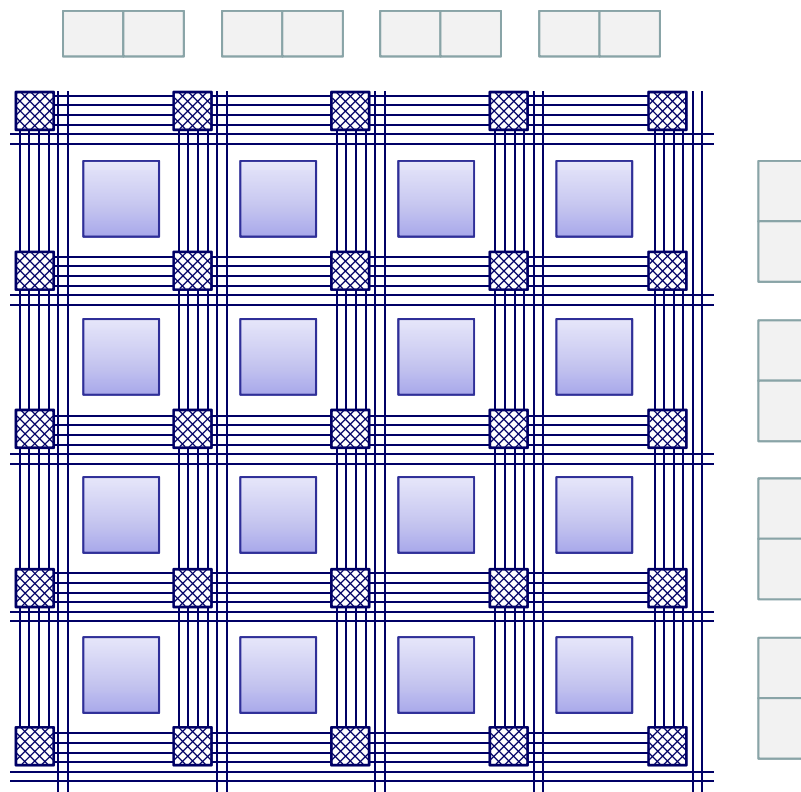
Conceptualmente um FPGA pode ser visto como uma matriz de Blocos Lógicos Configuráveis (CONFIGURABLE LOGIC BLOCK – CLB) independentes, rodeados na periferia por BLOCOS DE ENTRADA/SAÍDA (I/O Blocks), interligáveis por meio de um conjunto de recursos de encaminhamento, cuja configuração é controlada por um conjunto de células de memória.




Um FPGA possui três tipos principais de recursos: BLOCOS LÓGICOS CONFIGURÁVEIS (CLB), BLOCOS DE ENTRADA E SAÍDA (IOB), e INTERLIGAÇÕES PROGRAMÁVEIS entre blocos lógicos executada por uma matriz de interconexões.

Os blocos lógicos configuráveis CLB implementam funções lógicas (que podem ser combinatórias ou sequenciais) e formam um arranjo bidimensional, e as interligações programáveis são organizadas como canais de encaminhamento horizontal e vertical entre as linhas e colunas de blocos lógicos que os permitem conectar de maneira conveniente, em função das necessidades de cada projeto. Há interligações de tipo distinto para sinais de Clock, sinais Locais e sinais ditos de Caminho Longo.



Estrutura conceptual de um FPGA evidenciando os recursos disponíveis.



-  BLOCO LÓGICO CONFIGURÁVEL (CONFIGURABLE LOGIC BLOCK – CLB)
-  BLOCO de Entrada/Saída (INPUT/OUTPUT BLOCK – IOB)
-  COMUTADOR DE ENCAMINHAMENTO (SWITCH MATRIX)

Os blocos lógicos funcionais são normalmente compostos por uma ou mais tabelas de consulta (LOOK-UP TABLE – LUT).

A programação da lógica consiste no armazenamento na LUT da tabela de verdade da função lógica a realizar. As tabelas de consulta apresentam, na grande maioria das arquitecturas 4 entradas, e assemelham-se a memórias ROM (READ ONLY MEMORY) de 16 por 1, permitindo a implementação de lógica combinatória.

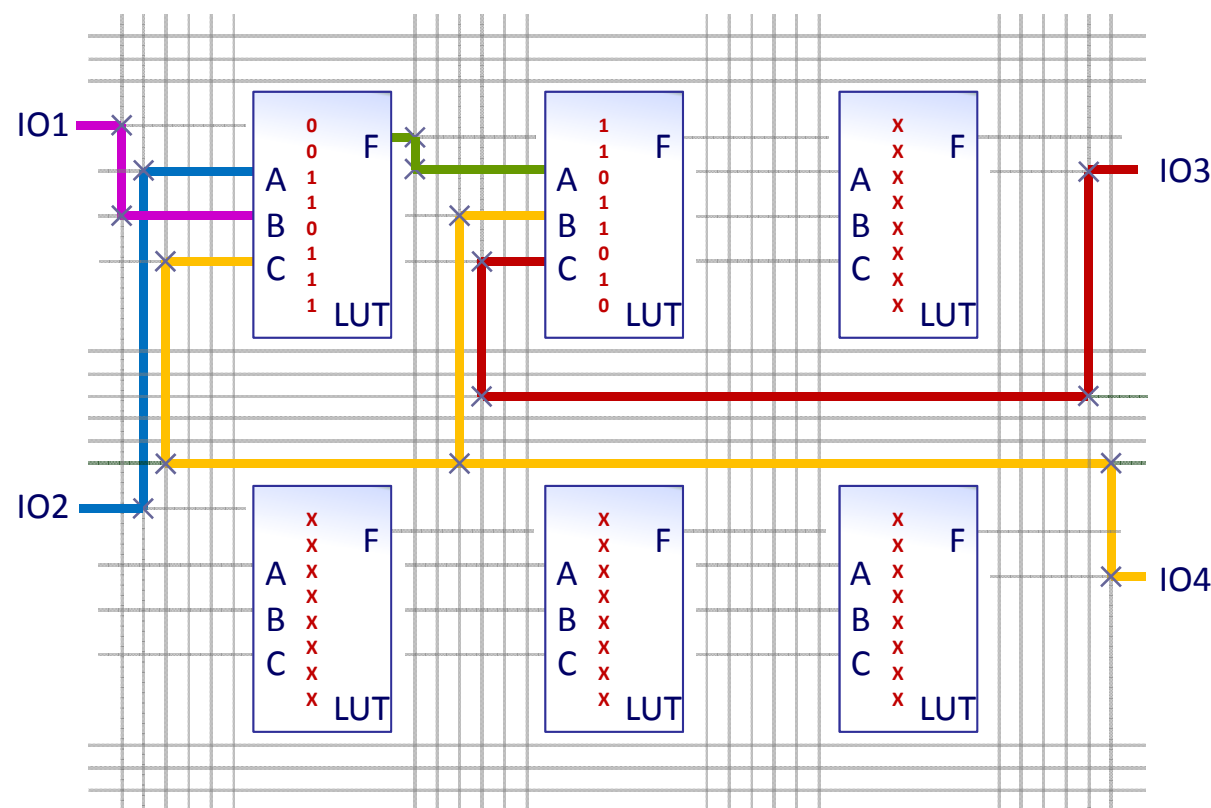
As arquitecturas de granulosidade fina são caracterizadas por blocos lógicos mais simples, com uma ou mais portas lógicas de duas entradas, um ou mais multiplexers usados na implementação das funções lógicas definidas pelo utilizador, e em alguns casos, um flip-flop.

O uso de tabelas de consulta ou de multiplexers na implementação de funções lógicas conduz a que os FPGAs sejam classificadas em dois grupos: FPGAs baseados em multiplexers e FPGAs baseados em tabelas de consulta LUT.

Existem também diferenças na tecnologia de programação: os FPGAs de maior densidade de integração têm por base a tecnologia das memórias estáticas SRAM (STATIC RANDOM ACCESS MEMORY), embora a tecnologia anti-fusível tenha também muita expressão no mercado. Existem igualmente FPGAs baseadas em tecnologia EEPROM (ELECTRICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY) e FLASH.

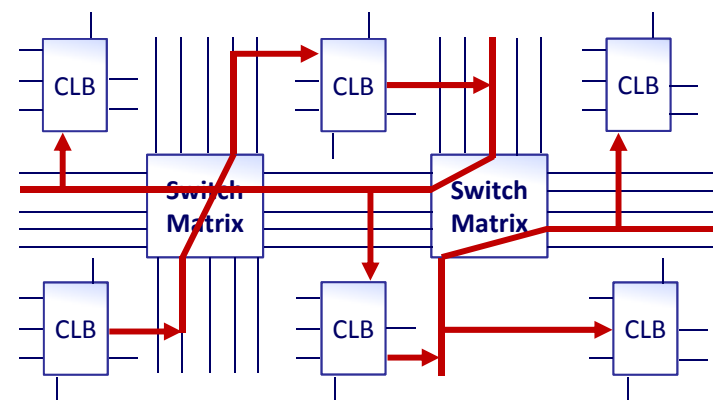
A arquitectura ao lado é baseada no FPGA XC4000 – da empresa Xilinx que domina cerca de 50% do mercado de FPGAs e introduziu a primeira FPGA pouco depois da sua fundação em 1984.





Exemplo de programação de LUTs num FPGA.

As LUT (LOOK UP TABLES) podem ser programadas para qualquer tipo de porta necessária ao projeto. As interligações entre as LUTs são programáveis como exemplificado pelas linhas coloridas. A configuração de uma LUT está exemplificada pela sequência de 1s e 0s no seu interior.



Exemplo de uma arquitectura de encaminhamento num FPGA.

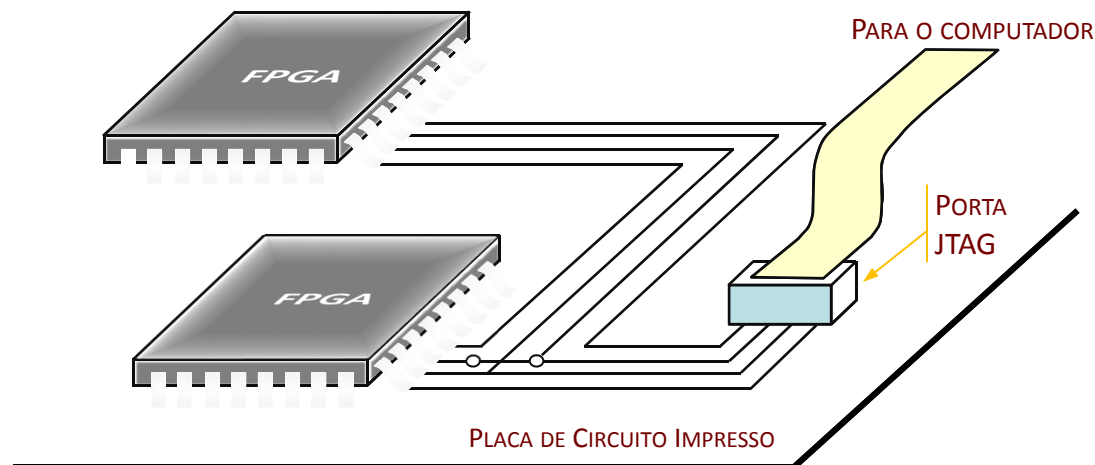
As entradas ou saídas de cada bloco CLB podem interligar-se. Uma matriz de interconexões formada por um conjunto de COMUTADORES DE ENCAMINHAMENTO (SWITCH MATRIX) programáveis situados em cada uma das intersecções entre os canais horizontais e verticais assegura o encaminhamento de sinais entre segmentos do mesmo canal ou do canal transversal.

Alguns sinais atravessam os COMUTADORES DE ENCAMINHAMENTO sem serem interrompidos, embora possam ter uma ou mais derivações intermédias. O tempo de propagação aumenta devido ao atraso introduzido por cada comutador atravessado.

Existem várias maneiras de programar CPLDs e FPGAs: sendo dispositivos delicados para serem manipulados, é comum serem programados quando já soldados na própria placa de circuito impresso (o que se designa por IN-SYSTEM PROGRAMMING – ISP) minimizando-se o manuseio e a necessidade de manter previamente um inventário de dispositivos já pré-programados.

Com o aumento da densidade de componentes nestas placas, e métodos sofisticados de montagem como o BALL GRID ARRAY, tem-se tornado cada vez mais popular o uso da porta de interface padronizada pelo standard IEEE 1149.1 designado TEST ACCESS PORT AND BOUNDARY-SCAN, comumente chamado JOINT TEST ACTION GROUP – JTAG.

O JTAG propôs uma arquitetura básica para testes (BOUNDARY-SCAN TEST), a ser incorporada nos circuitos complexos.



Placa de circuito de impresso exemplificando a porta padronizada para teste e reconfiguração parcial dinâmica no circuito de dispositivos CPLD e FPGA.

Dispositivos contendo a arquitetura JTAG podem receber dados e instruções através dos pinos de I/O, de forma a ser testada a sua funcionalidade e assegurada a integridade de componentes individuais e de ligações ao nível de placa de circuito impresso.

O hardware interno dedicado recebe dados fornecidos por 4 sinais:

- Clock do Teste (Test Clock–TCK), Seleção do Modo de Teste (Test Mode Select–TMS), Entrada de Dados de Teste (Test Data In–TDI) e Saída de Dados de Teste (Test Data Out–TDO).

O padrão JTAG vem sendo cada vez mais utilizado na reprogramação de CPLDs e FPGAs.

1. **LSD-3 – ROM – PAL – PLA**
2. Hierarquia da Implementação de Circuitos Lógicos
3. Circuitos de Lógica Programável
4. ROM e PROM
5. ROM – Implementação de 4 Funções de 3 Variáveis numa ROM
6. Blocos de Memória Construídos com Módulos PROM
7. PLA
8. PLA – Implementação de 4 Funções de 3 Variáveis num PLA
9. PAL – Implementação de 4 Funções de 3 Variáveis numa PAL (1)
10. PAL – Implementação de 4 Funções de 3 Variáveis numa PAL (2)
11. Implementação de 4 Funções de 3 Variáveis em ROM, PLA e PAL
12. Diagramas Lógicos de Notação Simplificada
13. PAL 16L8
14. Arquitectura Interna da PAL 16L8
15. Linguagens de Descrição de Hardware
16. WinCUPL – Ficheiros Implicados no Processo de Gravação da PAL
17. WinCUPL – Ficheiros Envolvidos no Desenvolvimento e na Programação
18. CUPL – Sintaxe e Elementos da Linguagem
19. PAL 22V10
20. Realização de Portas Simples numa PAL 22V10
21. Realização de uma Função Simples numa PAL 22V10
22. Análise Parcial do Fuse-plot do Ficheiro «.Doc» Gerado pelo CUPL
23. FPGA – Field-Programmable Gate Array
24. Arquitectura Típica de um FPGA
25. LUTs e Comutadores de Encaminhamento de um FPGA
26. Programação de CPLDs e FPGAs
27. LSD – 3 Índice 1

