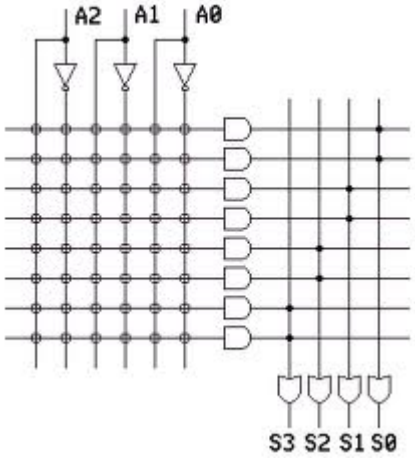
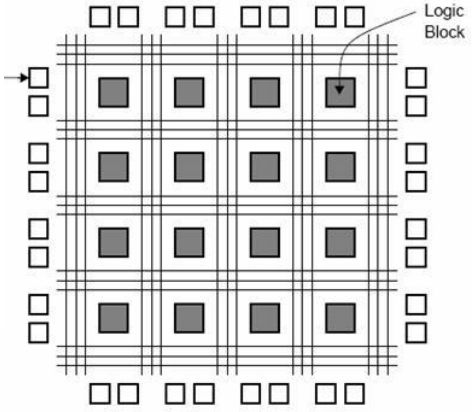
TÉCNICO LISBOA

Sistemas Digitais (SD)

Lógica Programável





TÉCNICO LISBOA

Aula Anterior

- **Na aula anterior:**
  - Circuitos de controlo, transferência e processamento de dados
  - Exemplo de uma arquitectura simples de um processador

TÉCNICO LISBOA

Planeamento

SEMANA	TEÓRICA 1	TEÓRICA 2	PROBLEMAS/LABORATÓRIO
17/Fev a 21/Fev	Introdução	Sistemas de Numeração	
24/Fev a 28/Fev	CARNAVAL	Álgebra de Boole	P0
02/Mar a 06/Mar	Elementos de Tecnologia	Funções Lógicas	VHDL
9/Mar a 13/Mar	Minimização de Funções	Minimização de Funções	L0
16/Mar a 20/Mar	Def. Circuito Combinatório; Análise Temporal	Circuitos Combinatórios	P1
23/Mar a 27/Mar	Circuitos Combinatórios	Circuitos Combinatórios	**L1**
30/Mar a 03/Abr	Circuitos Sequenciais: Latches	Circuitos Sequenciais: Flip-Flops	P2
06/Abr a 10/Abr	FÉRIAS DA PÁSCOA	FÉRIAS DA PÁSCOA	FÉRIAS DA PÁSCOA
13/Abr a 17/Abr	Caracterização Temporal	Registos	L2
20/Abr a 24/Abr	Contadores	Circuitos Sequenciais Síncronos	P3
27/Abr a 01/Mai	Síntese de Circuitos Sequenciais Síncronos	Síntese de Circuitos Sequenciais Síncronos	L3
04/Mai a 08/Mai	Exercícios	Memórias	P4
11/Mai a 15/Mai	Máq. Estado Microprogramadas: Circuito de Dados e Circuito de Controlo	Máq. Estado Microprogramadas: Microprograma	L4
18/Mai a 22/Mai	Circuitos de Controlo, Transferência e Processamento de Dados de um Processador	Lógica Programável	P5
25/Mai a 29/Mai	P6	P6	L5

Teste 1

TÉCNICO LISBOA

Sumário

- **Tema da aula de hoje:**
  - Lógica programável:
    - ROM
    - PLA
    - PAL
    - FPGA
  - Linguagens de descrição de hardware
    - VHDL

- **Bibliografia:**
  - **G. Arroz, C. Sêrro, "Sistemas Digitais: Apontamentos das Aulas Teóricas", IST, 2005**: Capítulo 18 (disponível no **Fenix**)

Programmable Logic Device

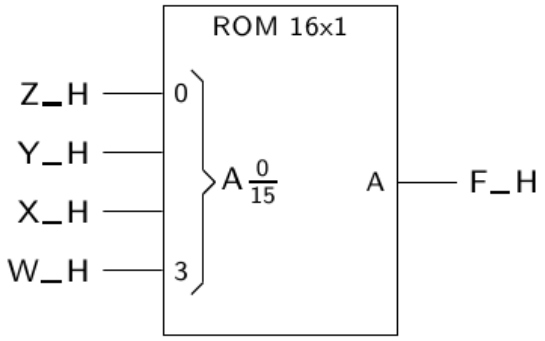
- **PLD: Programmable Logic Device**
  - Vários dispositivos disponíveis com a possibilidade de programação da função lógica implementada:
    - ROM: Read-Only Memory (ROM, PROM, EPROM,EEPROM, etc…)
    - PLA: Programmable Logic Array
    - PAL: Programmable Array Logic
    - FPGA: Field Programmable Gate Array
  - **Função:** implementação, num só circuito integrado, de circuitos com lógica combinatória (e/ou sequencial) de média complexidade, que de outra forma seriam implementados com vários circuitos integrados.

Read Only Memory (ROM)

- **ROM: Read-Only Memory**
  - Diferentes famílias disponíveis:
    - ROM - mask programmable ROM
    - PROM – field Programmable ROM
    - EPROM - Erasable Programmable ROM
    - EEPROM - Electrically Erasable Programmable ROM

Read Only Memory (ROM)

- **ROM: Read-Only Memory**
  - Exemplo:



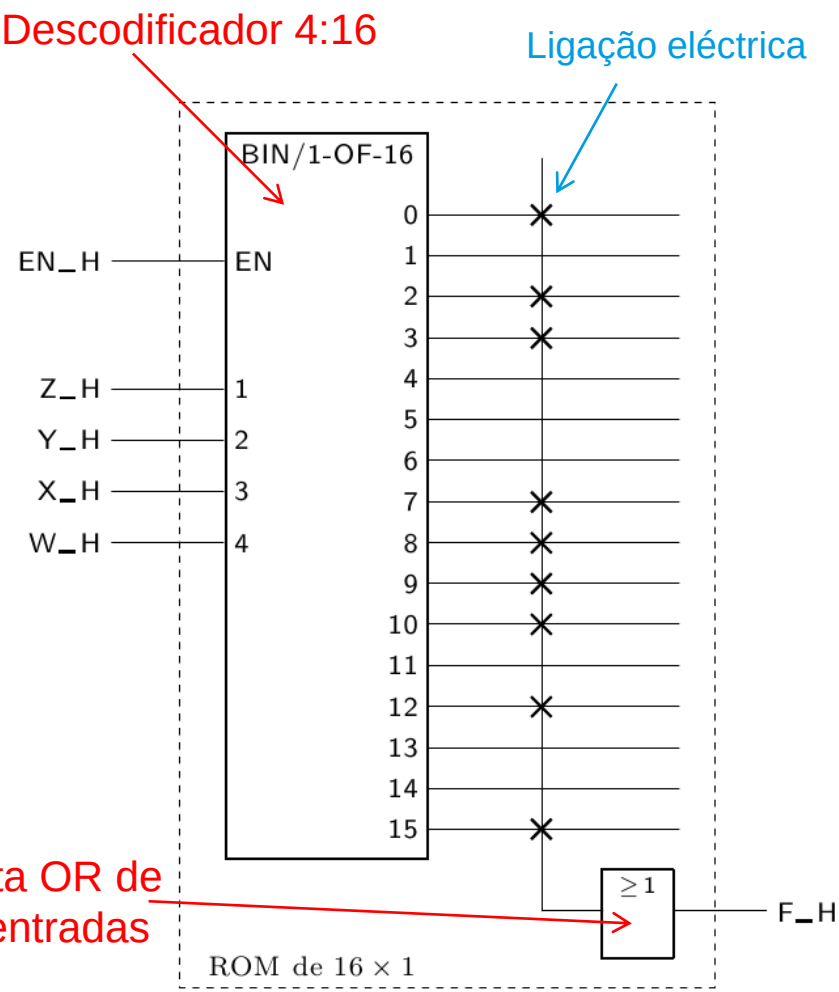
Como implementar uma função booleana F(W,X,Y,X) definida pela tabela de verdade?

Palavra #	W_H A3	X_H A2	Y_H A1	Z_H A0	Dados
0	L	L	L	L	H
1	L	L	L	H	L
2	L	L	H	L	H
3	L	L	H	H	H
4	L	H	L	L	L
5	L	H	L	H	L
6	L	H	H	L	L
7	L	H	H	H	H
8	H	L	L	L	H
9	H	L	L	H	H
10	H	L	H	L	H
11	H	L	H	H	L
12	H	H	L	L	H
13	H	H	L	H	L
14	H	H	H	L	L
15	H	H	H	H	H

Read Only Memory (ROM)

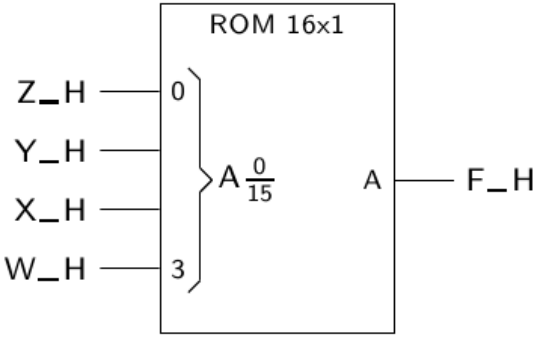
- **ROM: Read-Only Memory**

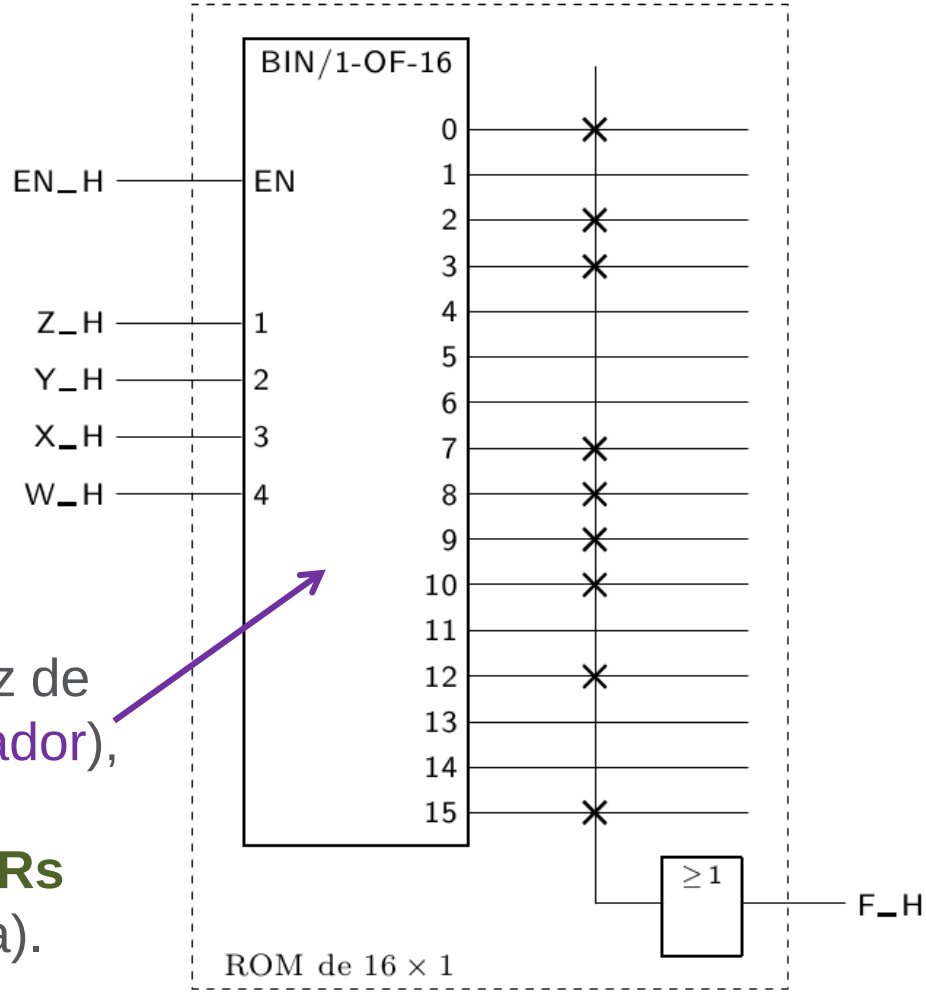
Palavra #	W_H A3	X_H A2	Y_H A1	Z_H A0	Dados
0	L	L	L	L	H
1	L	L	L	H	L
2	L	L	H	L	H
3	L	L	H	H	H
4	L	H	L	L	L
5	L	H	L	H	L
6	L	H	H	L	L
7	L	H	H	H	H
8	H	L	L	L	H
9	H	L	L	H	H
10	H	L	H	L	H
11	H	L	H	H	L
12	H	H	L	L	H
13	H	H	L	H	L
14	H	H	H	L	L
15	H	H	H	H	H



Read Only Memory (ROM)

- **ROM: Read-Only Memory**
  - Exemplo:

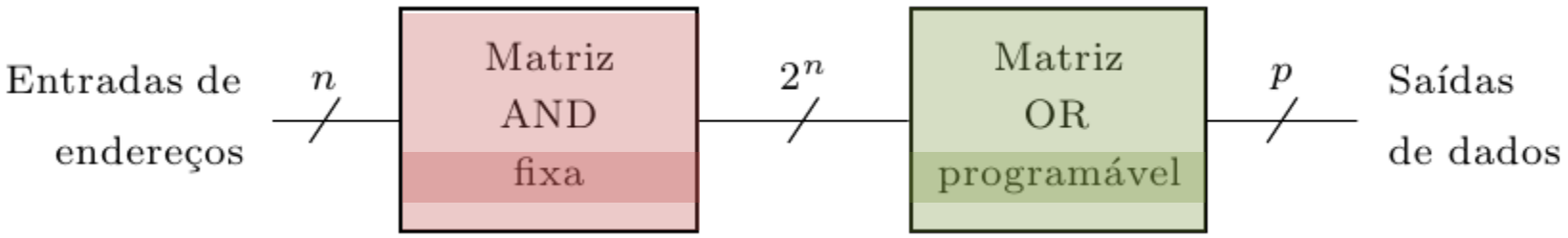




A **ROM** pode ser vista como uma matriz de **ANDs** não programável (do descodificador), correspondente a todos os mintermos possíveis, seguida de uma matriz de **ORs** programável (uma porta por cada saída).

Read Only Memory (ROM)

- **ROM: Read-Only Memory**



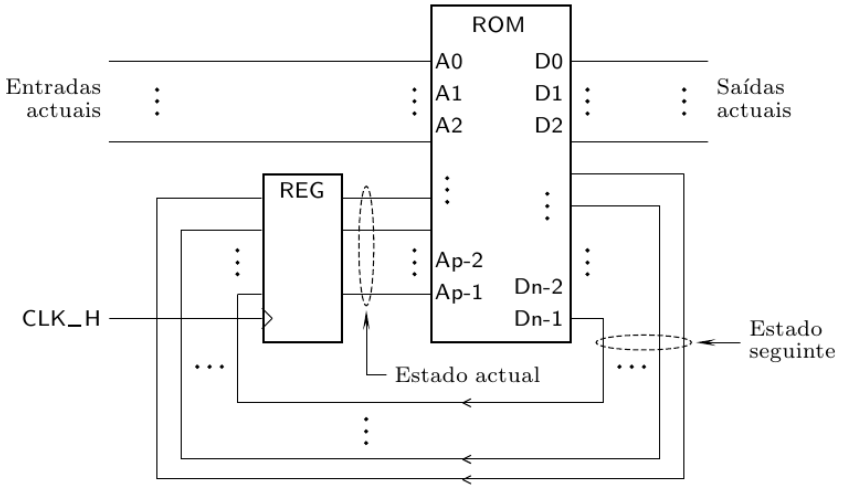
Geração de todos os mintermos

Agrupamento dos mintermos

- Ao contrário de outros dispositivos (ver a seguir), a ROM não impõe restrições no número de mintermos gerados (2^n) e agrupados.
- Exemplo:
  - uma ROM de 8k x 8 bits pode implementar, no máximo, 8 funções booleanas simples (uma por cada saída) de 13 variáveis booleanas (porque $8k = 2^{13}$).

Read Only Memory (ROM)

- **ROM: Read-Only Memory**
  - Exemplos de aplicação:
    - Implementação de funções booleanas combinatórias (genéricas);
    - Implementação de sistemas sequenciais micro-programados;
    - Armazenamento, em memória não volátil, de programas executados por processadores;
      - **Exemplo:** configuração do sistema de interface de entradas e saídas (BIOS) de um computador.



TÉCNICO LISBOA

Read Only Memory (ROM)

- **ROM: Read-Only Memory**
  - **Vantagens:**
    - Facilidade e rapidez de definição do seu conteúdo a partir da tabela de verdade da função;
    - Existe software para programação automática;
    - Pouco dispendiosas.
  - **Desvantagens:**
    - Uma vez que gera todos os mintermos para o conjunto de variáveis de entrada, conduz a desperdício de recursos, caso esses mintermos não sejam utilizados pela função;
    - Quando o número de entradas é muito elevado, pode tornar-se impraticável a utilização de ROMs, devido à limitação do número de entradas;
    - Mais lenta e consome mais potência do que circuitos dedicados.

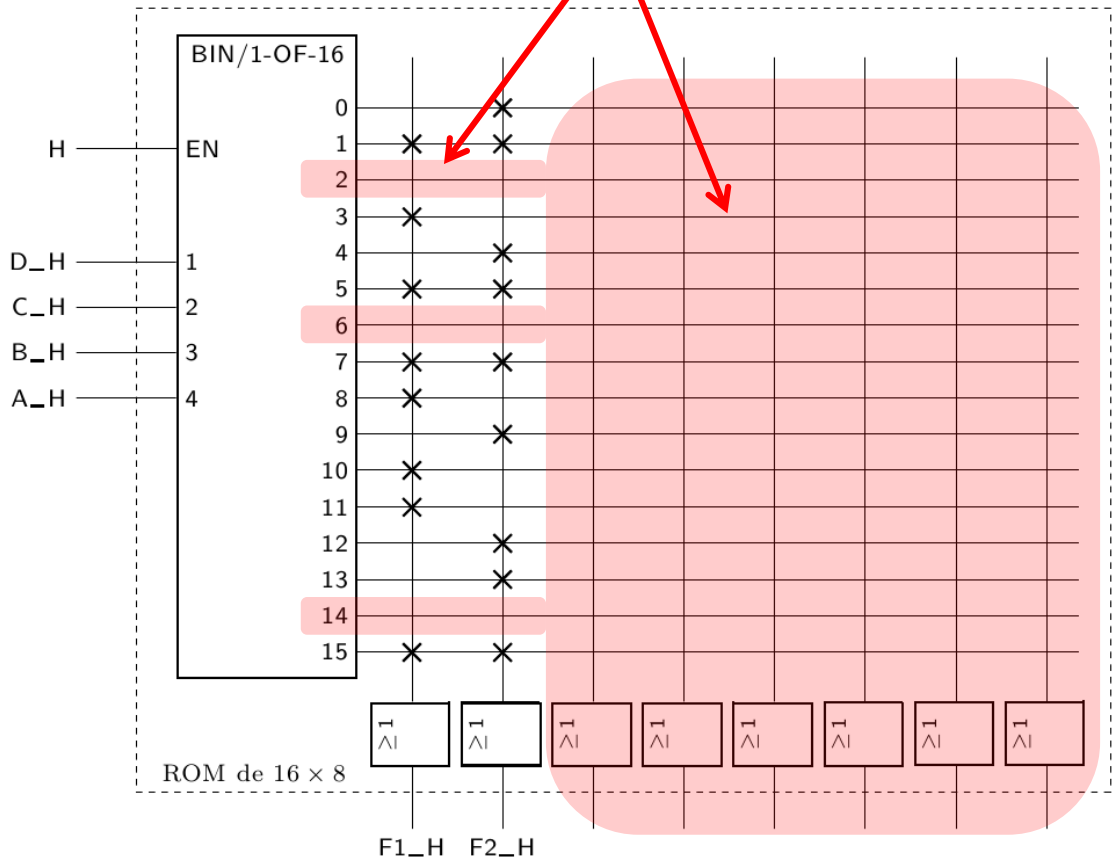
Read Only Memory (ROM)

- **ROM: Read-Only Memory - Exemplo**

Desperdício:
- mintermos não utilizados
- saídas não utilizadas

Tabela de Verdade

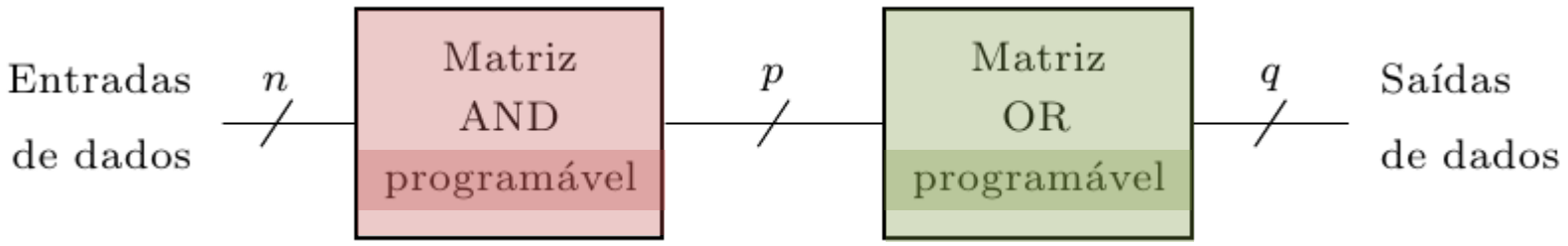
A_H	B_H	C_H	D_H	F1_H	F2_H
L	L	L	L	L	H
L	L	L	H	H	H
L	L	H	L	L	L
L	L	H	H	H	L
L	H	L	L	L	H
L	H	L	H	H	H
L	H	H	L	L	L
L	H	H	H	H	H
H	L	L	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	L	H	H	H	L
H	H	L	L	L	H
H	H	L	H	L	H
H	H	H	L	L	L
H	H	H	H	H	H



Programmable Logic Array (PLA)

PLA: Programmable Logic Array

- Para ultrapassar os inconvenientes da utilização de ROMs, os fabricantes de circuitos integrados conceberam dispositivos programáveis (PLDs), com restrições ao nível de:
  - Nº de entradas (n)
  - Nº de portas AND (p)
  - Nº de portas OR (q)



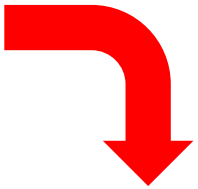
Geração de mintermos (máx. p)

Agrupamento de mintermos (máx. q)

Programmable Logic Array (PLA)

PLA: Programmable Logic Array

- Para ultrapassar os inconvenientes da utilização de ROMs, os fabricantes de circuitos integrados conceberam dispositivos programáveis (PLDs), com restrições ao nível de:
  - Nº de entradas (n)
  - Nº de portas AND (p)
  - Nº de portas OR (q)



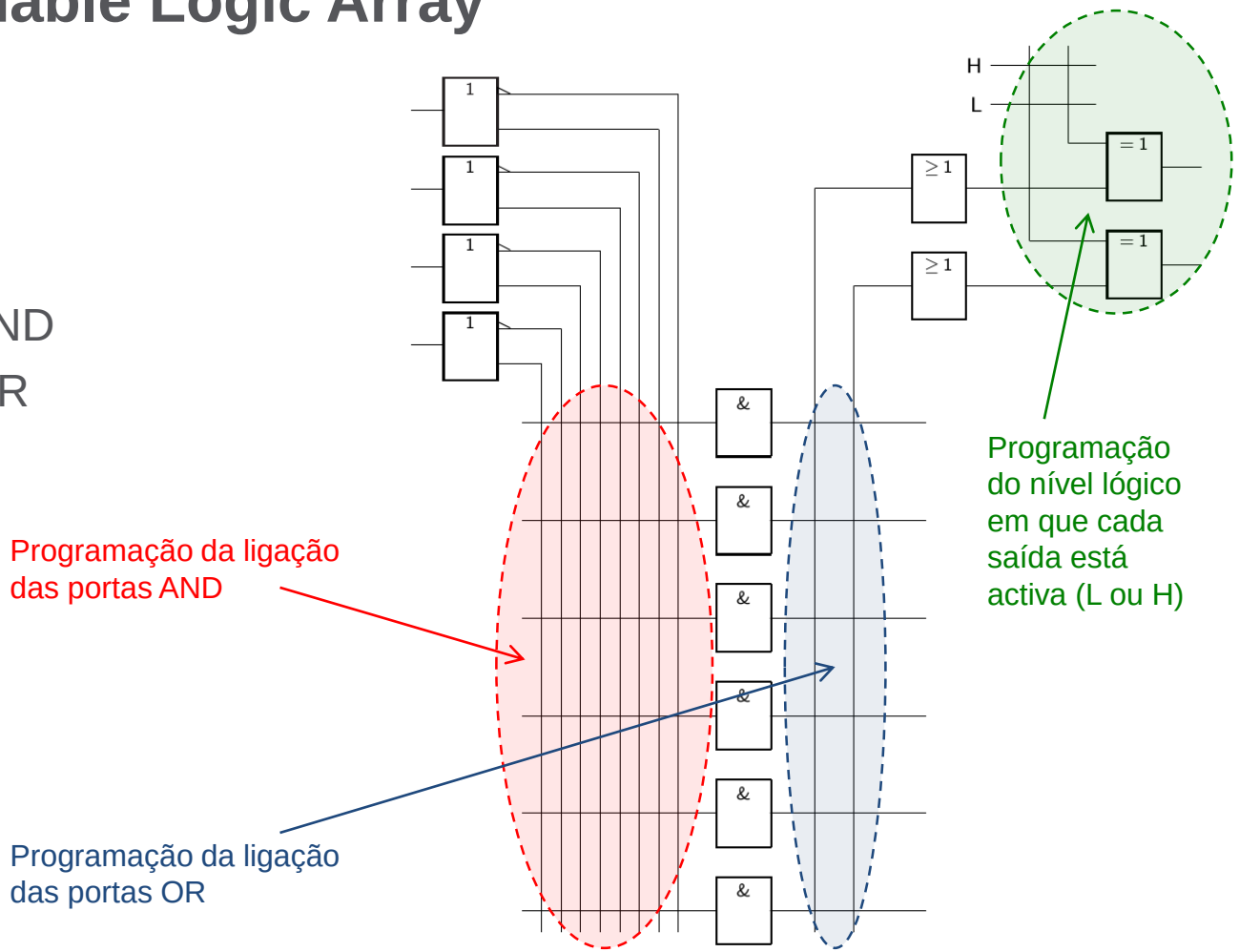
- **Consequências:**
  - Cada uma das q funções tem de ser expressa numa soma de produtos;
  - O número total de implicantes disponíveis não pode ultrapassar p.

- Estas restrições não existem nas ROMs, pois todos os mintermos estão disponíveis nas saídas do descodificador interno da ROM.

Programmable Logic Array (PLA)

PLA: Programmable Logic Array

- Exemplo:
  - n = 4 entradas
  - p = 6 portas AND
  - q = 2 portas OR



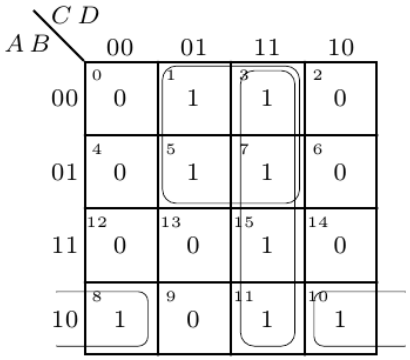
Programmable Logic Array (PLA)

PLA: Programmable Logic Array – Exemplo

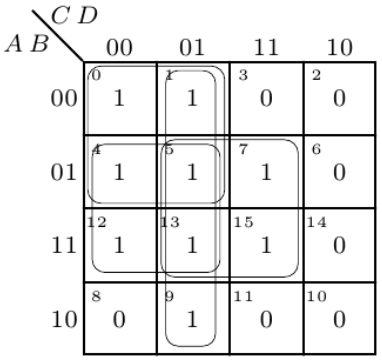
- Exemplo:
  - n = 4 entradas
  - p = 6 portas AND
  - q = 2 portas OR

Tabela de Verdade

A_H	B_H	C_H	D_H	F1_H	F2_H
L	L	L	L	L	H
L	L	L	H	H	H
L	L	H	L	L	L
L	L	H	H	H	L
L	H	L	L	L	H
L	H	L	H	H	H
L	H	H	L	L	L
L	H	H	H	H	H
H	L	L	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	L	H	H	H	L
H	H	L	L	L	H
H	H	L	H	L	H
H	H	H	L	L	L
H	H	H	H	H	H



$F1 = \overline{A}\,D + C\,D + A\,\overline{B}\,\overline{D}$



$F2 = \overline{A}\,\overline{C} + B\,\overline{C} + \overline{C}\,D + B\,D$

- 3 portas AND
- 1 porta OR
- Saída não negada (porta XOR)

- 4 portas AND
- 1 porta OR
- Saída não negada (porta XOR)

3 + 4 = 7 portas AND !!!

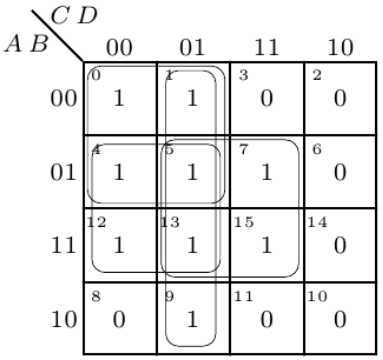
Programmable Logic Array (PLA)

PLA: Programmable Logic

- **Observação:**
  - Se agruparmos os maxtermos, em vez dos mintermos, obteremos uma expressão mais simples

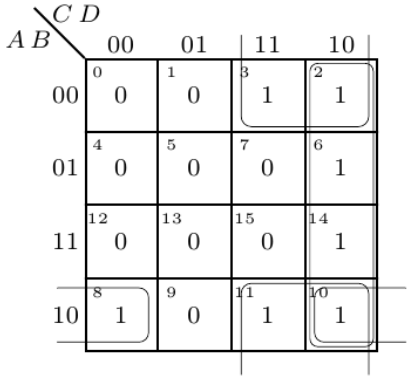
Problema:

  - A PLA não tem estrutura que facilite o uso de produtos de somas



$$F2 = \overline{A}\,\overline{C} + B\,\overline{C} + \overline{C}\,D + B\,D$$

- Alternativa:
  - Obter a expressão na negação de F2: $\overline{F2}$
  - Depois nega-se esta negação: $F2 = \overline{\overline{F2}}$



$$\overline{F2} = \overline{B}\,C + C\,\overline{D} + A\,\overline{B}\,\overline{D}$$

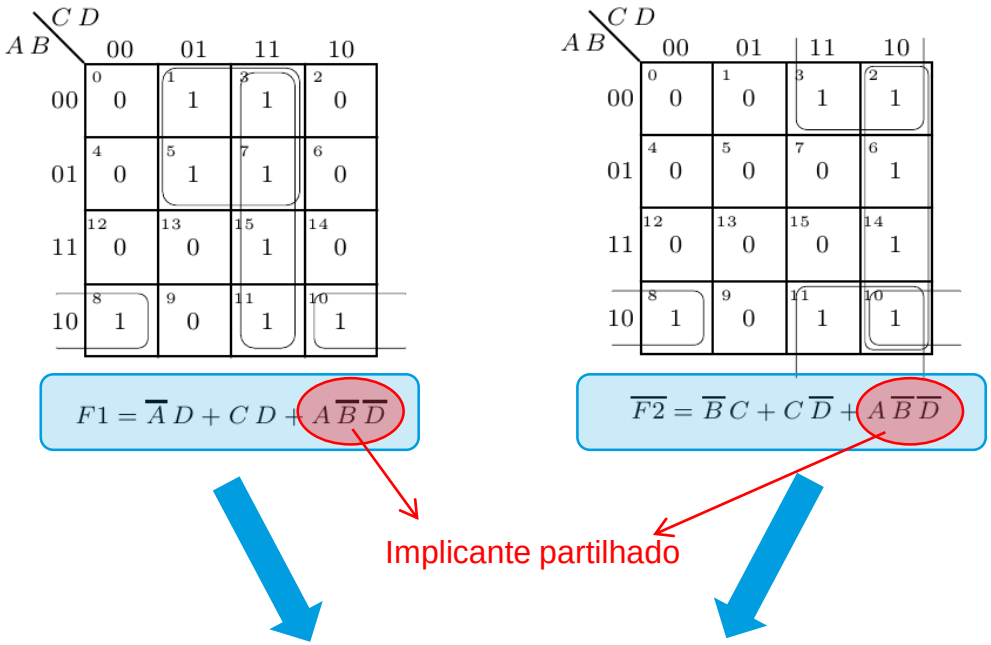
Programmable Logic Array (PLA)

PLA: Programmable Logic Array – Exemplo

- Exemplo:
  - n = 4 entradas
  - p = 6 portas AND
  - q = 2 portas OR

Tabela de Verdade

A_H	B_H	C_H	D_H	F1_H	F2_H
L	L	L	L	L	H
L	L	L	H	H	H
L	L	H	L	L	L
L	L	H	H	H	L
L	H	L	L	L	H
L	H	L	H	H	H
L	H	H	L	L	L
L	H	H	H	H	H
H	L	L	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	L	H	H	H	L
H	H	L	L	L	H
H	H	L	H	L	H
H	H	H	L	L	L
H	H	H	H	H	H



- 5 portas AND
- 2 porta OR
- 1 saída não negada (F1)
- 1 saída negada (F2)

OK!

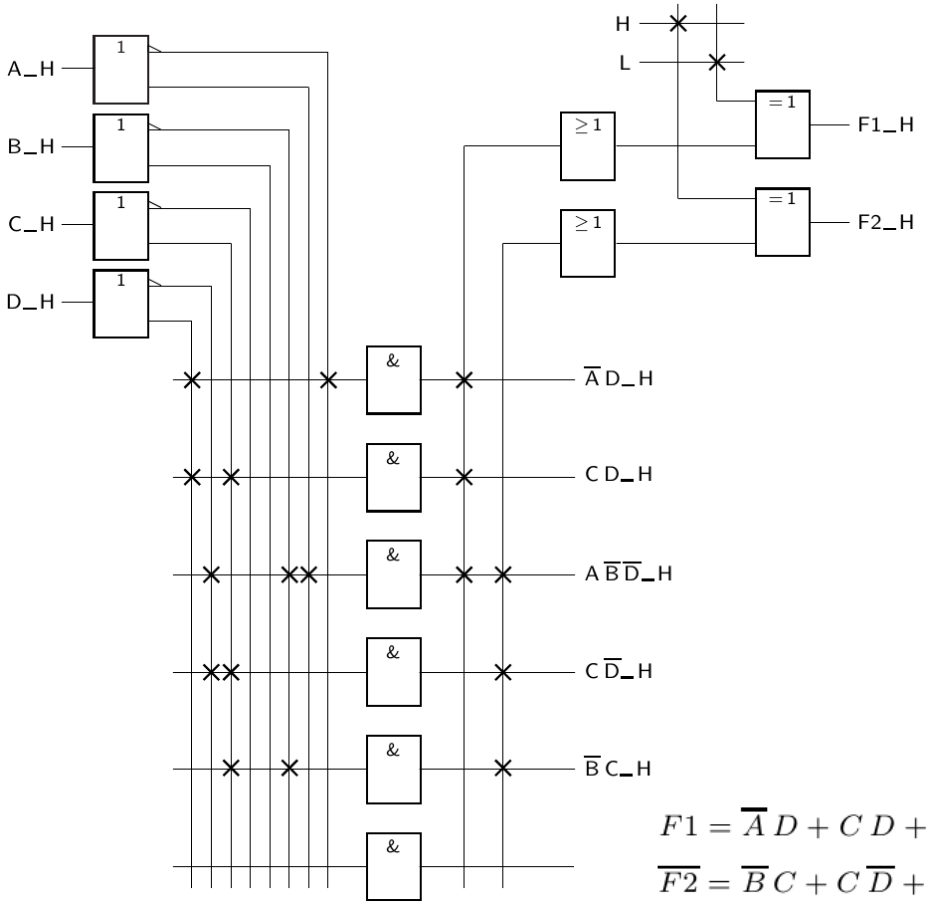
Programmable Logic Array (PLA)

PLA: Programmable Logic Array – Exemplo

- Exemplo:
  - n = 4 entradas
  - p = 6 portas AND
  - q = 2 portas OR

Tabela de Verdade

A_H	B_H	C_H	D_H	F1_H	F2_H
L	L	L	L	L	H
L	L	L	H	H	H
L	L	H	L	L	L
L	L	H	H	H	L
L	H	L	L	L	H
L	H	L	H	H	H
L	H	H	L	L	L
L	H	H	H	H	H
H	L	L	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	L	H	H	H	L
H	H	L	L	L	H
H	H	L	H	L	H
H	H	H	L	L	L
H	H	H	H	H	H

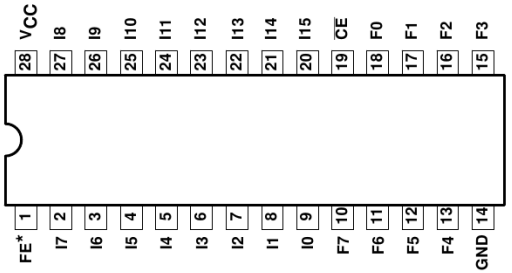


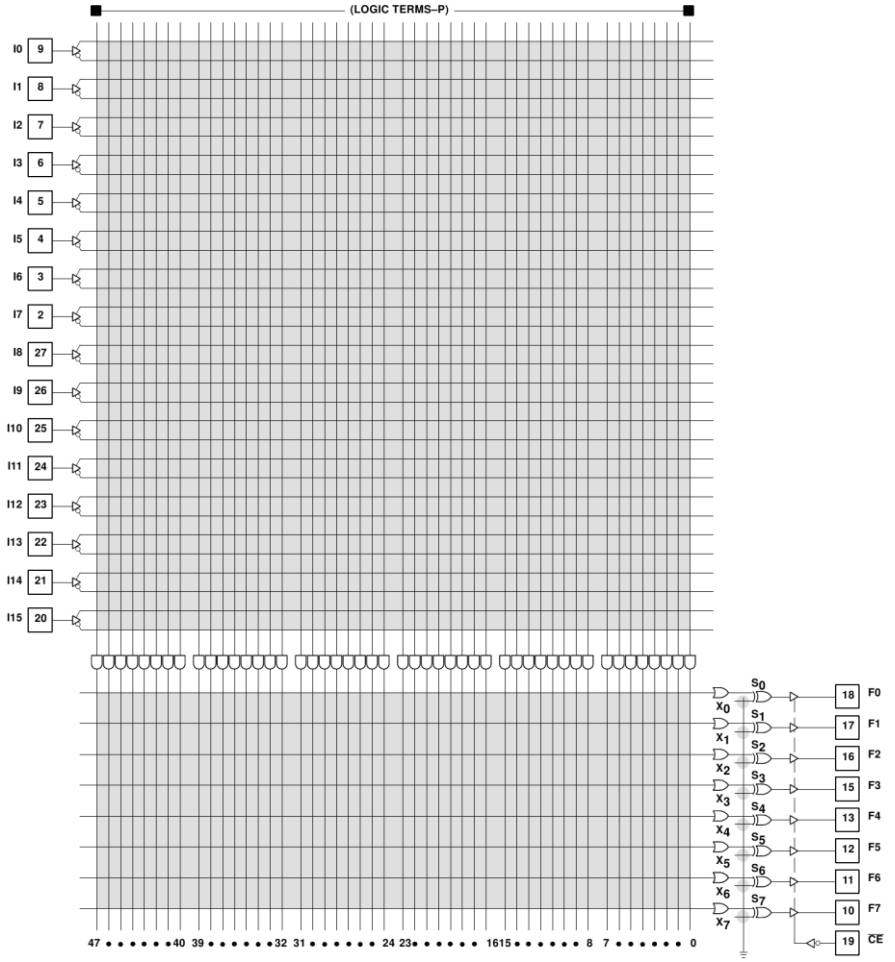
$$F1 = \overline{A}\,D + C\,D + A\,\overline{B}\,\overline{D}$$

$$\overline{F2} = \overline{B}\,C + C\,\overline{D} + A\,\overline{B}\,\overline{D}$$

Programmable Logic Array (PLA)

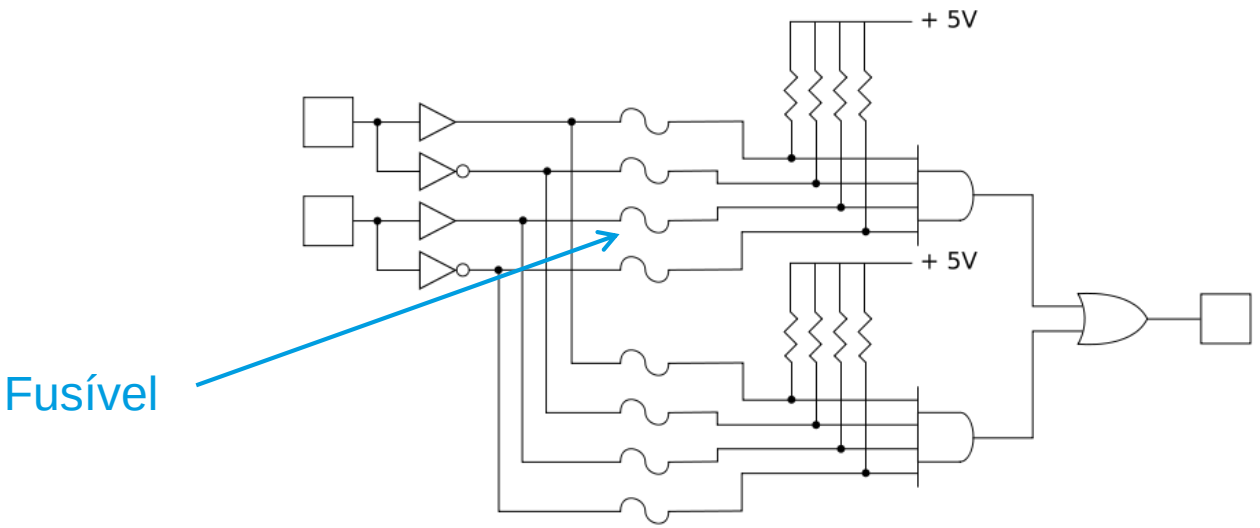
- **PLA: Programmable Logic Array**
  - Exemplo: PLS100 (Philips)
    - 16 entradas
    - p = 48 portas AND
    - q = 8 portas OR





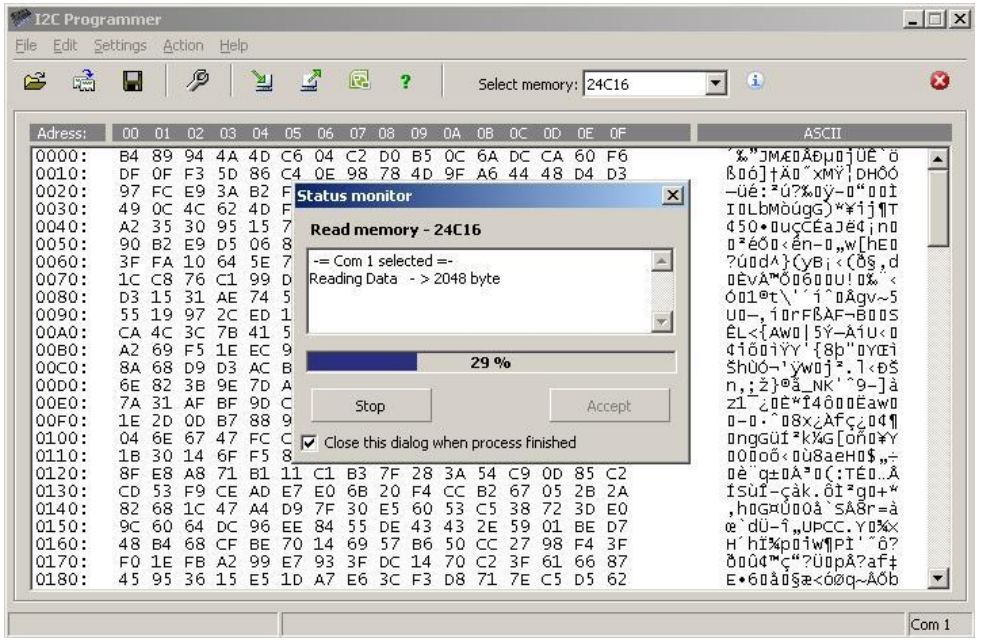
Programmable Logic Array (PLA)

- **Programação**
  - ***One-Time-Programming* (OTP)** - podem ser programados apenas uma única vez
    - Aquando da programação, existem fusíveis que são “queimados” e que irão definir os operandos de cada **mintermo**.



Programmable Logic Array (PLA)

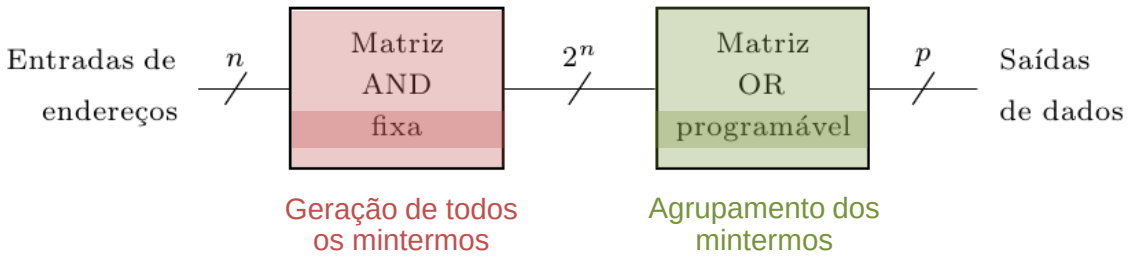
- **Programação**
  - O programador está ligado a um computador (PC), que lê um ficheiro com a tabela de verdade pretendida para o circuito



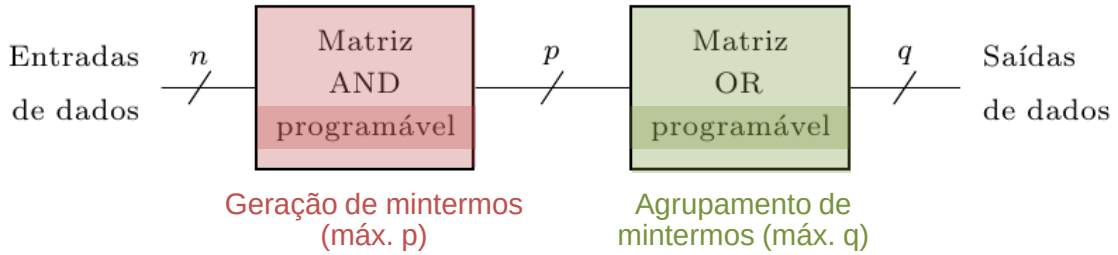
ROMs vs PLAs

ROMs vs PLAs

- No caso das ROMs, as ligações das portas AND estão fixas e é possível programar as ligações das portas OR:



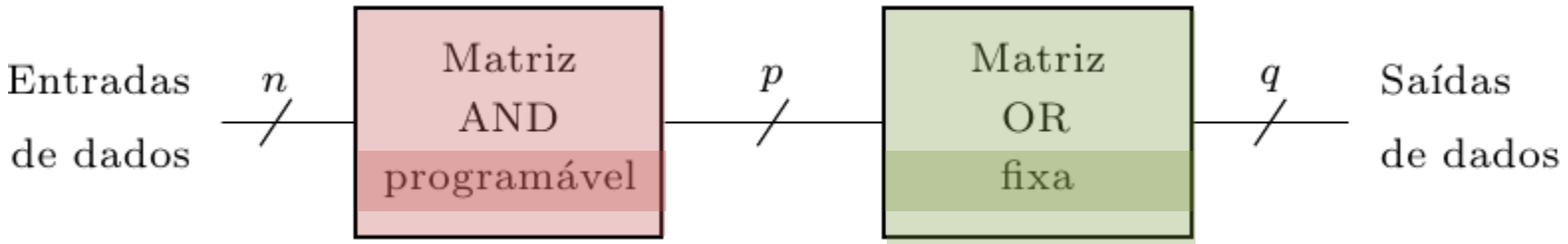
- No caso das PLAs, tanto as ligações das portas AND como as ligações das portas OR são programáveis:



Programmable Array Logic (PAL)

PAL: Programmable Array Logic

- No caso das PALs, as ligações entre as portas AND e as portas OR estão fixas, e apenas é possível programar as ligações das portas AND às entradas:



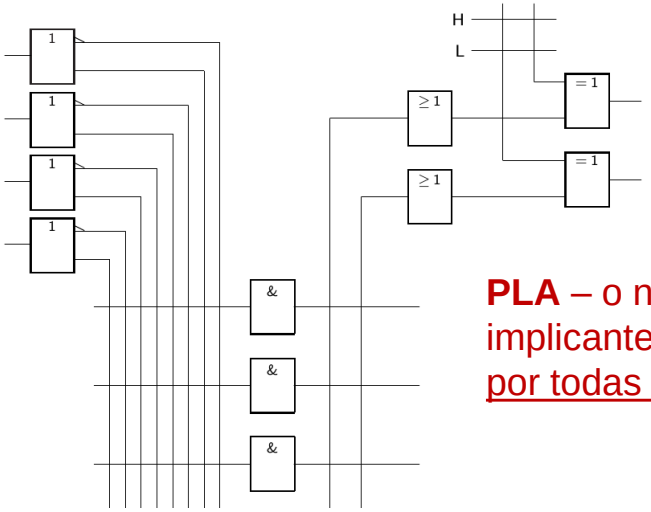
Geração de mintermos (máx. p)

Agrupamento de mintermos (máx. q)

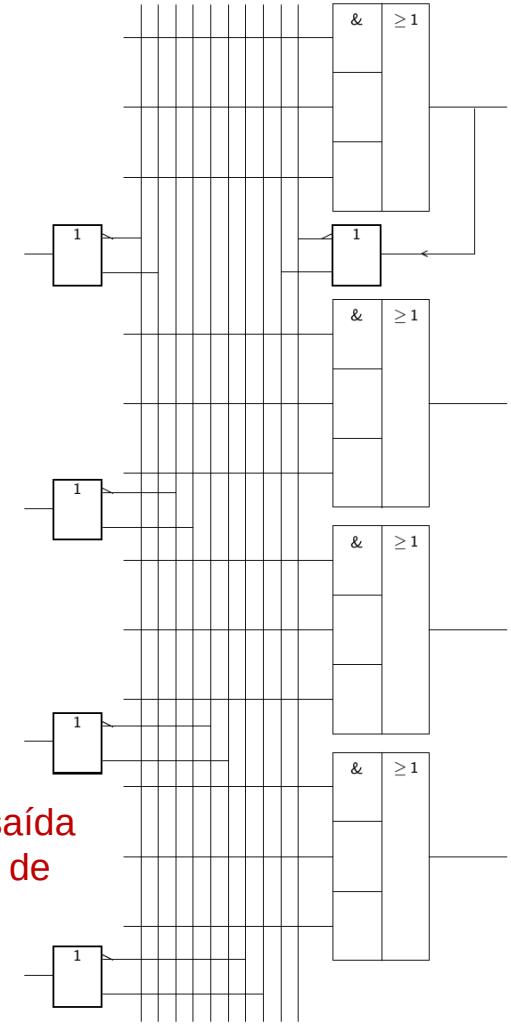
- Restrições:
  - Cada uma das q funções tem de ter a forma de uma soma de produtos;
  - O número de implicantes da soma não pode exceder p por função (numa PLA, o número de implicantes (p) é partilhado por todas as funções).

PALs vs PLAs

- **PALs vs PLAs:**



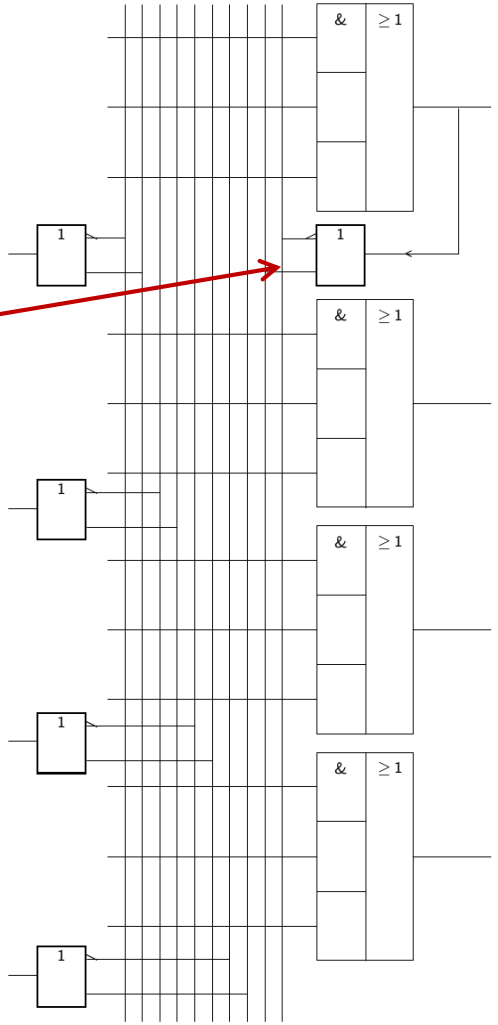
PLA – o número de implicantes (p) é partilhado por todas as funções.



PAL – cada função de saída pode usar p implicantes de forma independente.

Programmable Array Logic (PAL)

- **PAL: Programmable Array Logic**
  - Uma das linhas de saída pode ser realimentada para o interior da PAL, para permitir construir funções que necessitem de um maior número de portas AND.
  - Algumas PALs incluem também **flip-flops** nas saídas, de modo a permitir realizar circuitos sequenciais.



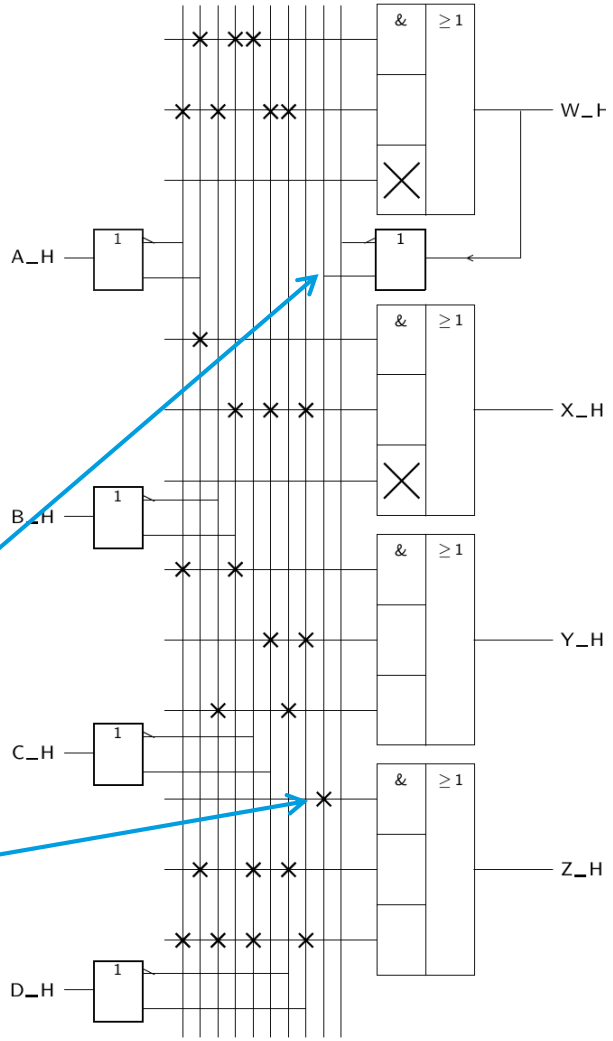
Programmable Array Logic (PAL)

PAL: Programmable Array Logic

- Exemplo:

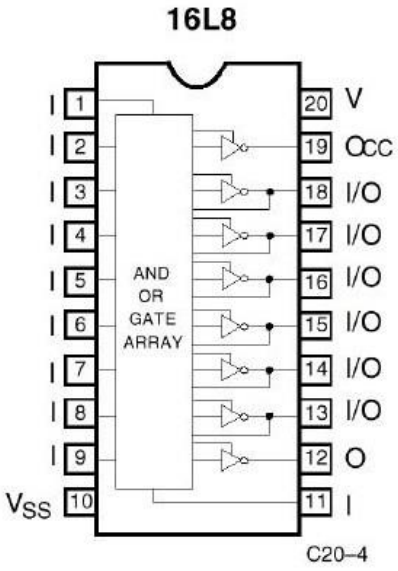
$$W = A\,B\,\overline{C} + \overline{A}\,\overline{B}\,C\,\overline{D}$$
$$X = A + B\,C\,D$$
$$Y = \overline{A}\,B + C\,D + \overline{B}\,\overline{D}$$
$$Z = A\,B\,\overline{C} + \overline{A}\,\overline{B}\,C\,\overline{D} + A\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D$$
$$= W + A\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D$$

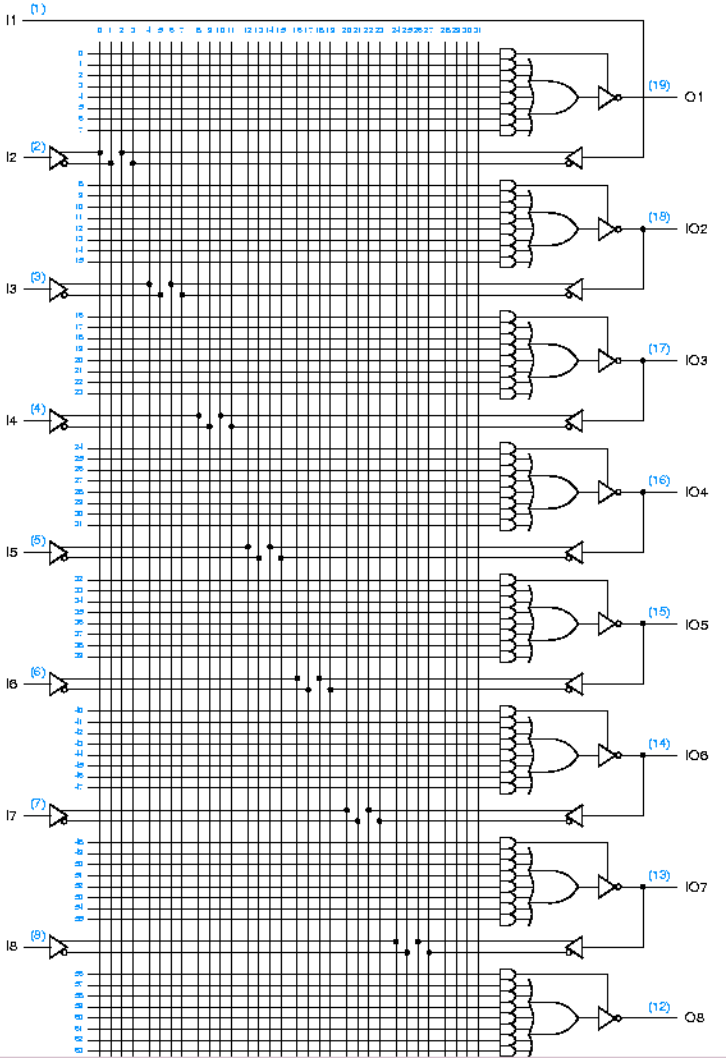
Realimentação da saída da função W (que corresponde, também, a mintermos da função Z), a fim de alargar o número de operandos da porta AND.



Programmable Array Logic (PAL)

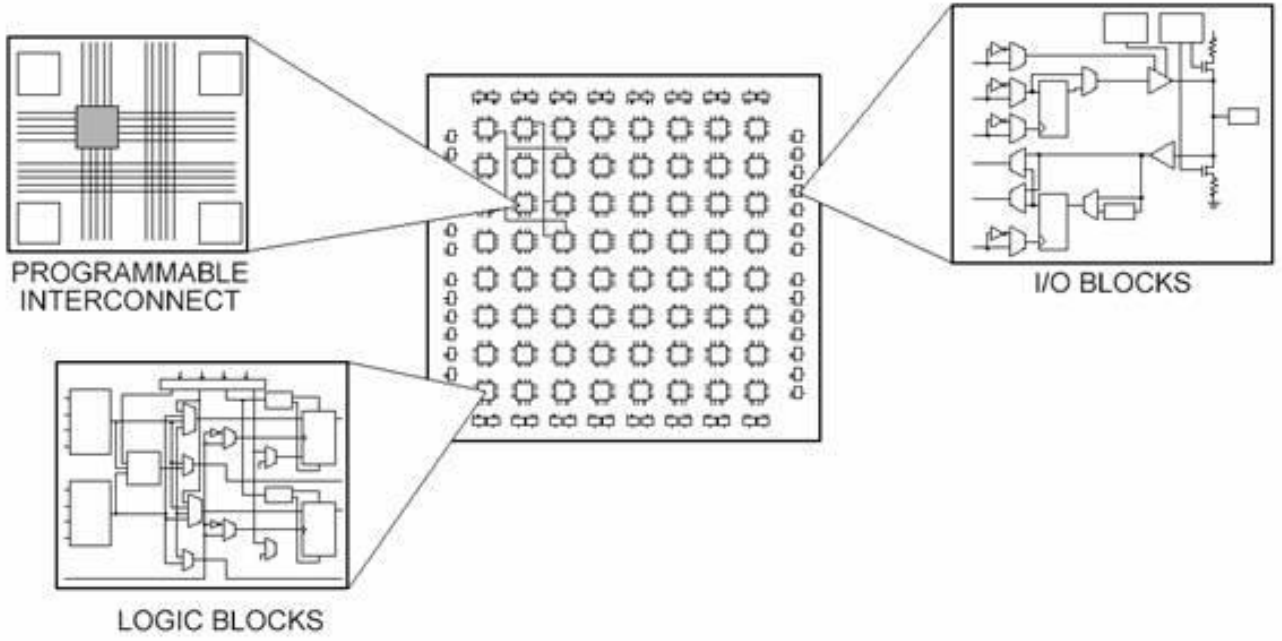
- **PAL: Programmable Array Logic**
  - Exemplo: PAL16L8





Field-Programmable Gate Array (FPGA)

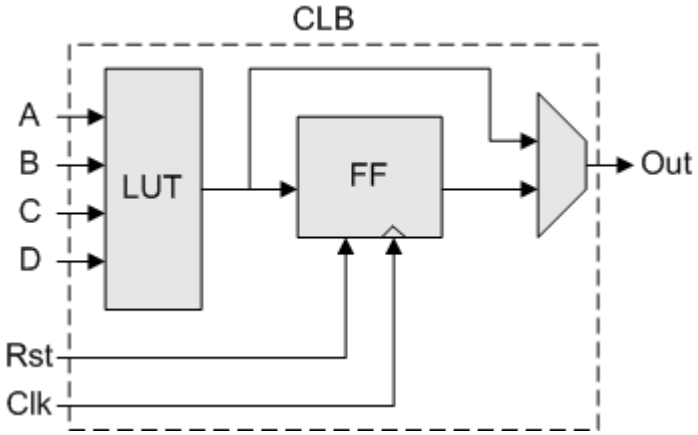
- **FPGA: Field-Programmable Gate Array**
  - Dispositivo constituído por uma grelha com milhares de blocos lógicos programáveis interligados entre si (**CLB: Configurable Logic Blocks**), em que cada bloco implementa uma função booleana simples:



Field-Programmable Gate Array (FPGA)

Configurable Logic Block (CLB)

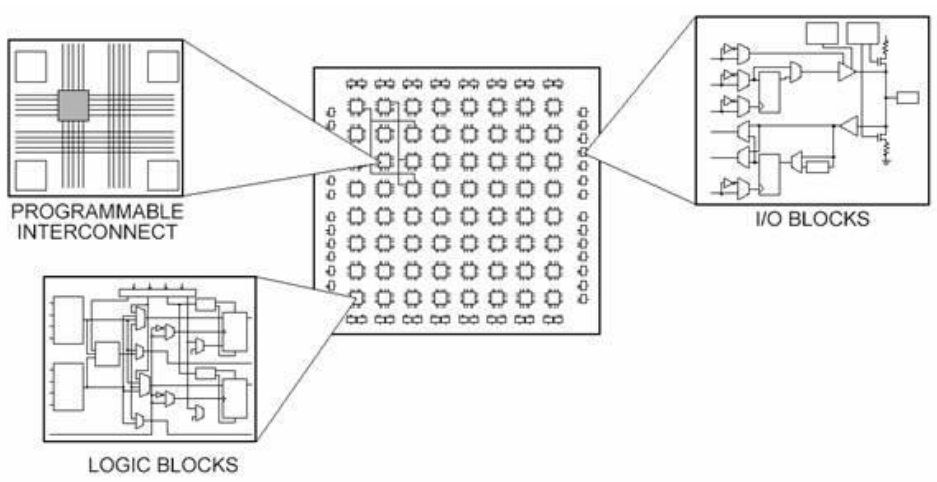
- Pode ser constituído por:
  - **Look-Up Table (LUT)**, semelhante a uma ROM, que permite definir uma qualquer **função combinatória** arbitrária de *n* entradas
  - Elemento de memória (ex: **Flip-Flop**), ligado à saída da LUT, que permite a realização de **circuitos sequenciais**.
- Exemplo (simples):



Field-Programmable Gate Array (FPGA)

FPGA: Field-Programmable Gate Array

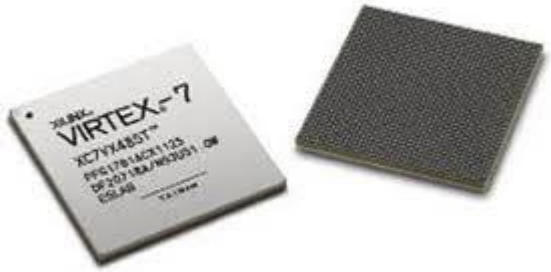
- A programação/configuração é feita aquando do ciclo de inicialização, em que a FPGA lê um ficheiro de configuração (.bit) a partir de uma ROM externa, a fim de configurar:
  - LUTs de todos os CLBs;
  - MUXs de saída de todos os CLBs;
  - Interligações entre CLBs;
  - Memórias internas;
  - Interface com o exterior (I/O).



- Pode ser configurada múltiplas vezes!

Field-Programmable Gate Array (FPGA)

- **FPGA: Field-Programmable Gate Array**
  - O grande número de CLBs (>10^6) actualmente disponibilizados por FPGAs de última geração permite a integração e implementação, num único chip, de:
    - Vários processadores (sistemas multi-core)
    - Processadores Digitais de Sinal (DSP)
    - Micro-controladores
    - Memórias, etc.

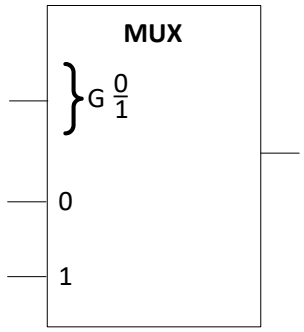


  - **Programação**:
    - Dada a elevada complexidade dos circuitos envolvidos, estes dispositivos são geralmente programados através de linguagens de descrição de circuitos (***Hardware Description Languages – HDL***):
      - VHDL
      - Verilog

VHDL

VHDL (VHSIC Hardware Description Language)

- **Exemplo 1**: multiplexer 2:1

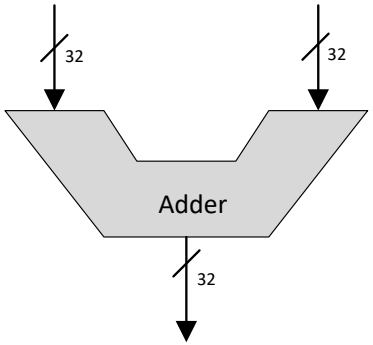



```
entity MUX is
  port (
    A   : in std_logic;
    B   : in std_logic;
    Sel : in std_logic;
    Out : out std_logic);
end entity MUX;

architecture RTL of MUX is
begin
    Out <= A when Sel = '1' else B;
end architecture RTL;
```

NOTA: esta informação é disponibilizada para efeitos meramente ilustrativos, não fazendo parte do programa de “Sistemas Digitais”.

- **VHDL (VHSIC Hardware Description Language)**
  - **Exemplo 2**: somador binário

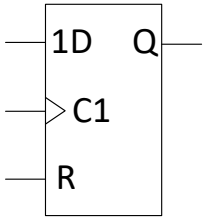



```
entity ADDER is
  generic (
    WIDTH : in natural := 32);
  port (
    OP1   : in std_logic_vector(WIDTH-1 downto 0);
    OP2   : in std_logic_vector(WIDTH-1 downto 0);
    SUM   : out std_logic_vector(WIDTH-1 downto 0));
end entity ADDER;

architecture RTL of ADDER is
begin
    SUM <= OP1 + OP2;
end architecture RTL;
```

NOTA: esta informação é disponibilizada para efeitos meramente ilustrativos, não fazendo parte do programa de “Sistemas Digitais”.

- **VHDL (VHSIC Hardware Description Language)**
  - **Exemplo 3**: flip-flop tipo D



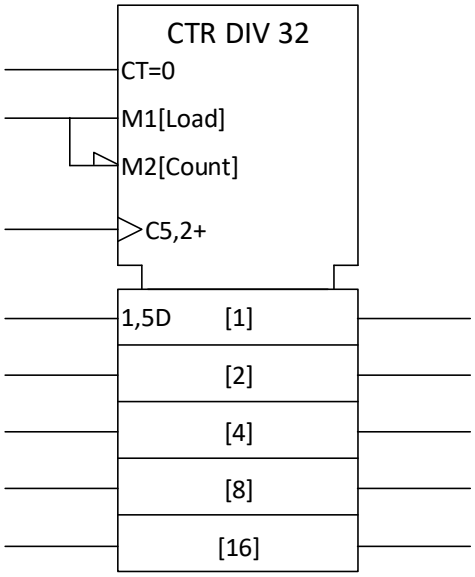

```
entity FLIP_FLOP is
  port (
    RST   : in std_logic;
    CLK   : in std_logic;
    D     : in std_logic;
    Q     : out std_logic);
end entity FLIP_FLOP;

architecture RTL of FLIP_FLOP is
begin
  process(RST, CLK)
  begin
    if RST = '1' then
      Q <= '0';
    elsif rising_edge(CLK) then
      Q <= D;
    end if;
  end process;
end architecture RTL;
```

NOTA: esta informação é disponibilizada para efeitos meramente ilustrativos, não fazendo parte do programa de “Sistemas Digitais”.

- **VHDL (VHSIC Hardware Description Language)**
  - **Exemplo 4**:

    Contador binário




```
entity COUNTER is
  generic (
    WIDTH : in natural := 5);
  port (
    RST   : in std_logic;
    CLK   : in std_logic;
    LOAD  : in std_logic;
    DATA  : in std_logic_vector(WIDTH-1 downto 0);
    Q     : out std_logic_vector(WIDTH-1 downto 0));
end entity COUNTER;

architecture RTL of COUNTER is
  signal CNT : unsigned(WIDTH-1 downto 0);
begin
  process(RST, CLK) is
  begin
    if RST = '1' then
      CNT <= (others => '0');
    elsif rising_edge(CLK) then
      if LOAD = '1' then
        CNT <= unsigned(DATA);
      else
        CNT <= CNT + 1;
      end if;
    end if;
  end process;

  Q <= std_logic_vector(CNT);
```

TÉCNICO LISBOA

Próxima Aula

- **Tema da Próxima Aula:**
  - Série de Problemas P6 – 1ª parte

Agradecimentos

Algumas páginas desta apresentação resultam da compilação de várias contribuições produzidas por:

- Nuno Roma
- Guilherme Arroz
- Horácio Neto
- Nuno Horta
- Pedro Tomás