Sistemas Digitais I
LESI :: 2º ano

Questões Práticas de Sistemas Combinacionais

António Joaquim Esteves
João Miguel Fernandes

www.di.uminho.pt/~aje

Bibliografia: secções 5.3 a 5.11, DDPP, Wakerly

DEP. DE INFORMÁTICA
ESCOLA DE ENGENHARIA
UNIVERSIDADE DO MINHO

5. Prática com Sistemas Combinacionais

- Sumário -

- PLDs *(dispositivos de lógica programável)*
- Descodificadores
- Descodificadores de 7-segmentos
- Codificadores
- *Buffers Tri-state*
- Multiplexadores
- Portas XOR e circuitos detectores de paridade
- Comparadores
- Somadores, subtractores e ALUs
- Multiplicadores

5. Prática com Sistemas Combinacionais

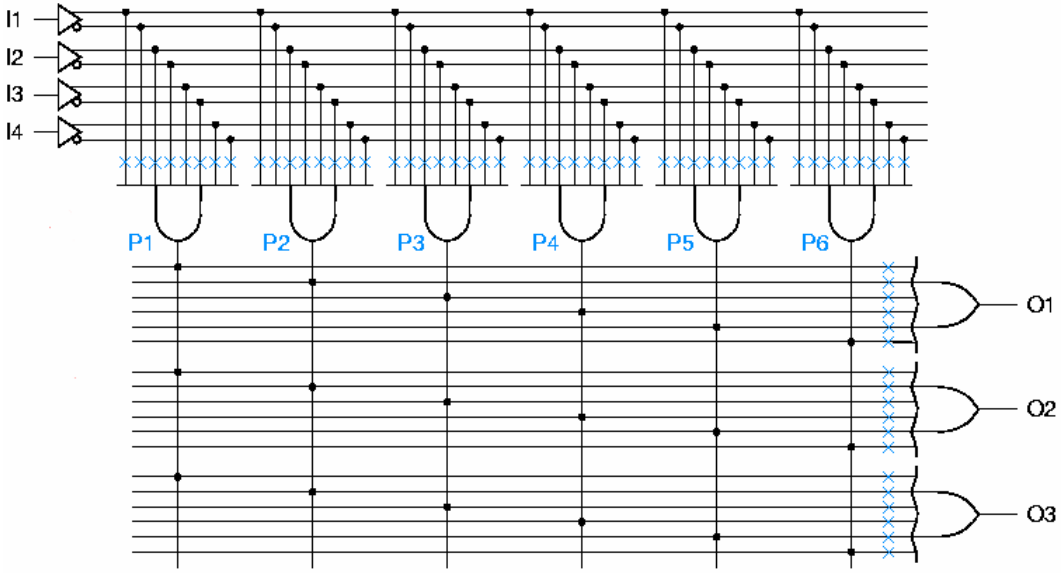
- PLDs (1) -

- Os primeiros PLDs a surgir foram os Programmable Logic Arrays (PLAs).
- Uma PLA é um dispositivo combinacional, com estrutura a 2-níveis AND-OR, que pode ser programado para efectuar qualquer expressão lógica do tipo soma-de-produtos.
- O tipo de expressão implementável numa PLA está limitada pelo:
  - número de entradas da PLA (n)
  - número de saídas da PLA (m)
  - número de termos de produto da PLA (p)
- O dispositivo é definido como "uma PLA n x m com p termos de produto". Normalmente, $p << 2^n$.
- Uma PLA n x m com p termos de produto contém p ANDs de 2n-entradas e m ORs de p-entradas.

5. Prática com Sistemas Combinacionais

- PLDs (2) -

- Cada entrada In está ligada a um *buffer* que gera uma cópia desse sinal (In) e o seu complemento ($\overline{In}$).
- As ligações possíveis de estabelecer estão assinaladas com Xs.
- O dispositivo é programado estabelecendo as ligações necessárias.
- As ligações são feitas através de fusíveis (células memória).

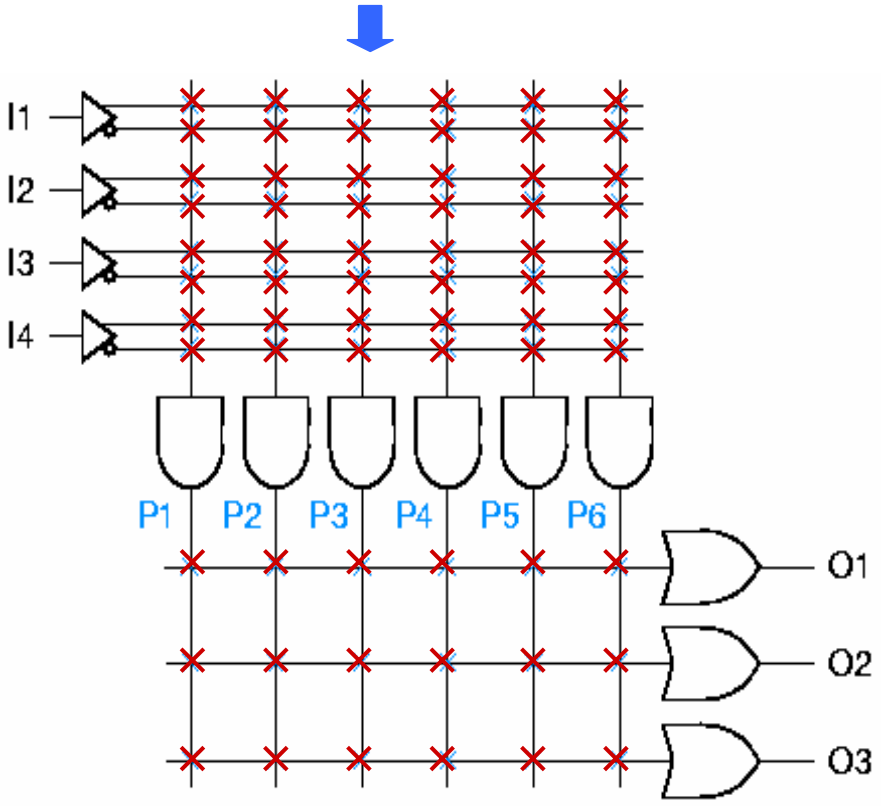


Uma PLA 4x3 com 6 termos de produto.

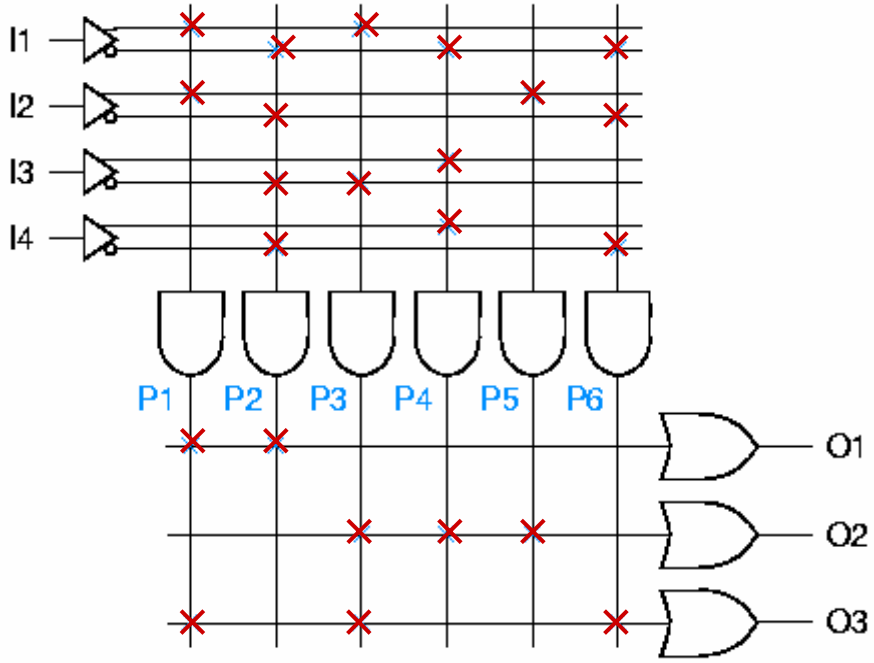
5. Prática com Sistemas Combinacionais

- PLDs (3) -

- Representação mais compacta da PLA 4x3 com 6 termos de produto.



- Exemplo



O1 = P1+P2 = I1·I2 + I1'·I2'·I3'·I4'

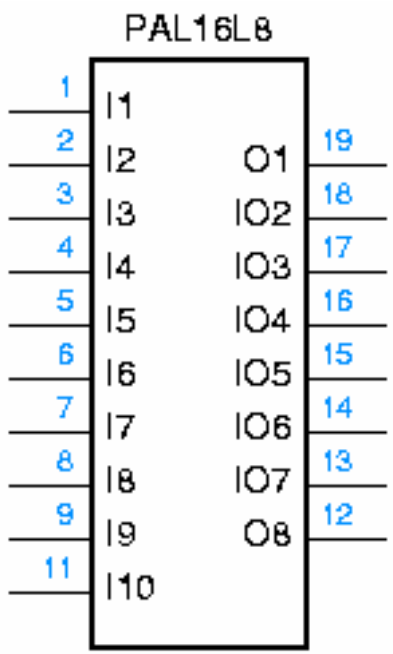
O2 = P3+P4+P5 = I1·I3' + I1'·I3·I4 + I2

O3 = P1+P3+P6 = I1·I2 + I1·I3' + I1'·I2'·I4'

5. Prática com Sistemas Combinacionais

- PLDs (4) -

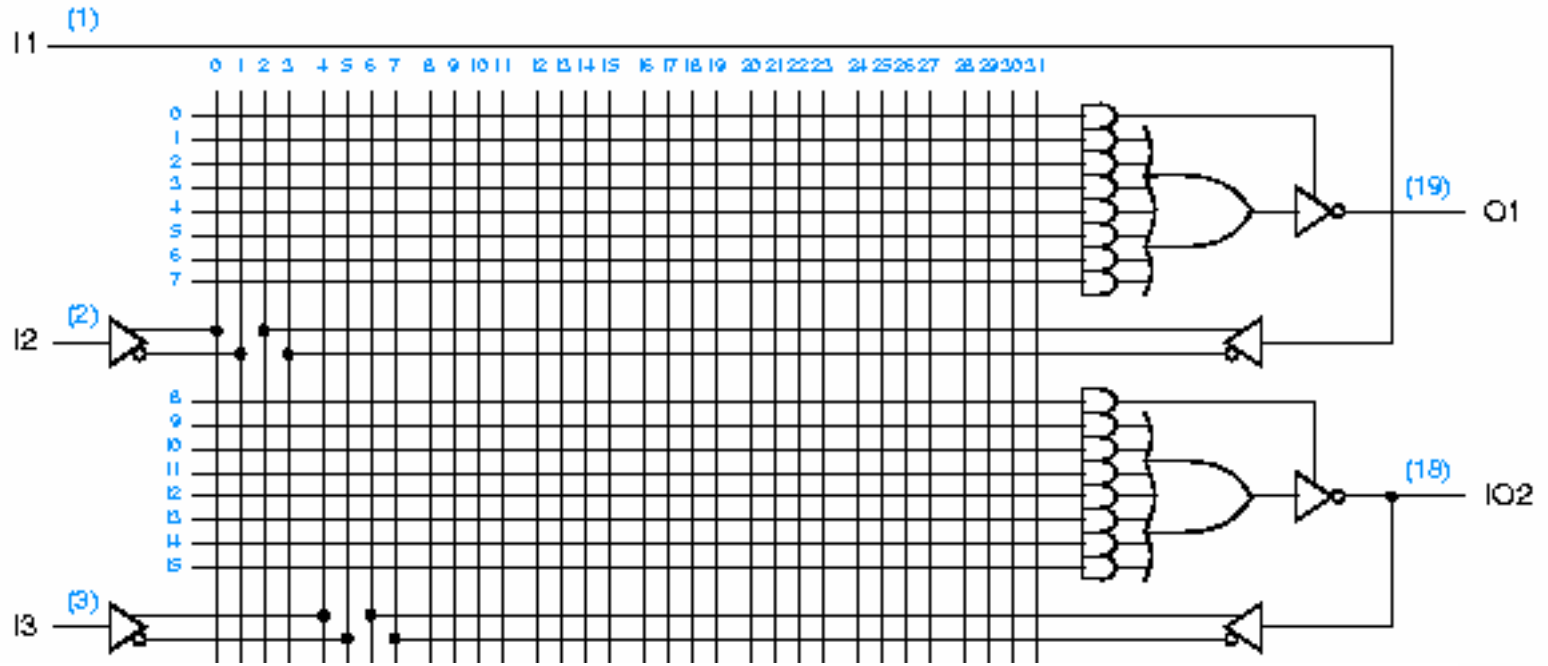
- Outro tipo de PLD, que mais não é do que um caso particular de PLA, é o Programmable Array Logic (PAL).
- Um dispositivo PAL tem um *array* de ORs fixo.
- Numa PAL, os termos de produto não são partilhados por várias saídas.
- Cada saída dispõe dum conjunto único e fixo de termos de produto que pode usar.
- Uma PAL é normalmente mais rápida do que uma PLA equivalente.



5. Prática com Sistemas Combinacionais

- PLDs (5) -

- Vista parcial do diagrama lógico da PAL 16L8.



- 10 entradas *I1..I10*, 2 saídas *O1 e O8*, 6 entradas/saídas *IO2..IO7*, *64*32* fusíveis, cada OR recebe 7 termos de produto.

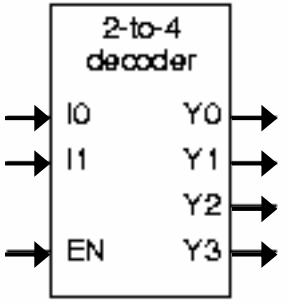
5. Prática com Sistemas Combinacionais

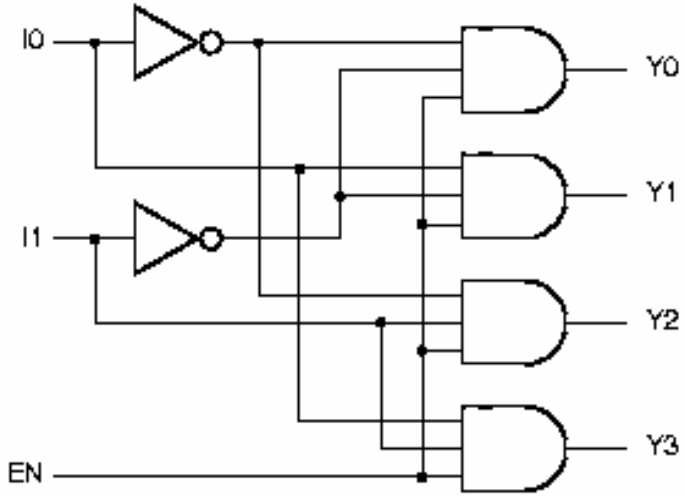
- Descodificadores (1) -

- Um **descodificador** é um circuito que converte uma entrada codificada numa saída codificada, sendo os códigos de entrada e saída diferentes.
- Habitualmente, o código de entrada tem menos bits que o de saída.
- O descodificador mais comum é o descodificador n-para-2^n ou binário.
- Usa-se um descodificador binário quando é preciso activar uma saída de entre 2^n possíveis, com base no valor duma entrada de n-bits.

- Descodificador 2-para-4

Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0





Por exemplo: Y0=EN./I1./I0

5. Prática com Sistemas Combinacionais

- Descodificadores (2) -

- Um CI 74x139 contém dois descodificadores 2-para-4 independentes.

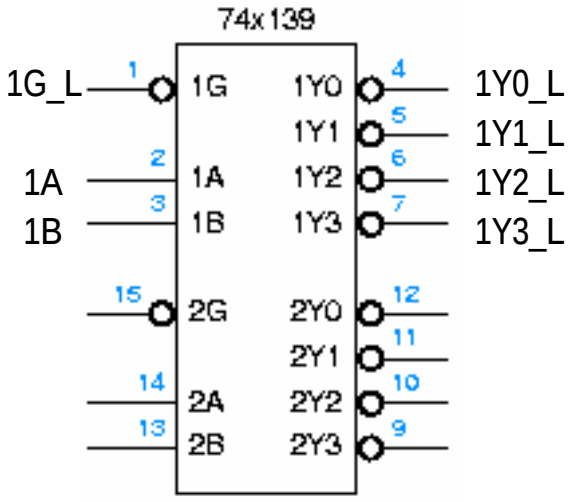


Tabela de verdade para um dos descodificadores

Inputs			Outputs			
G_L	B	A	Y3_L	Y2_L	Y1_L	Y0_L
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Os sinais com bolha são activos a zero
(anexa-se _L ao nome para indicar esse facto)

5. Prática com Sistemas Combinacionais

- Descodificadores (3) -

- UM CI 74x138 contém um descodificador 3-para-8.

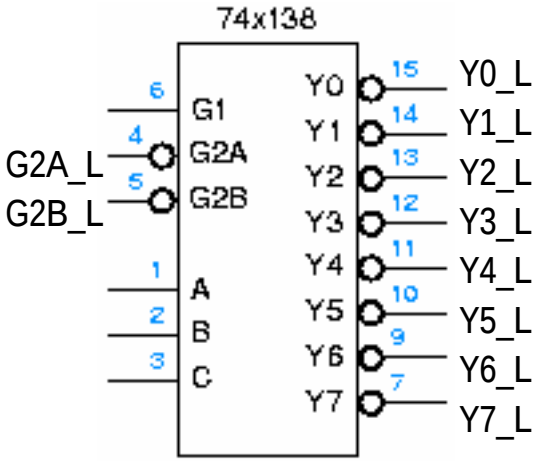


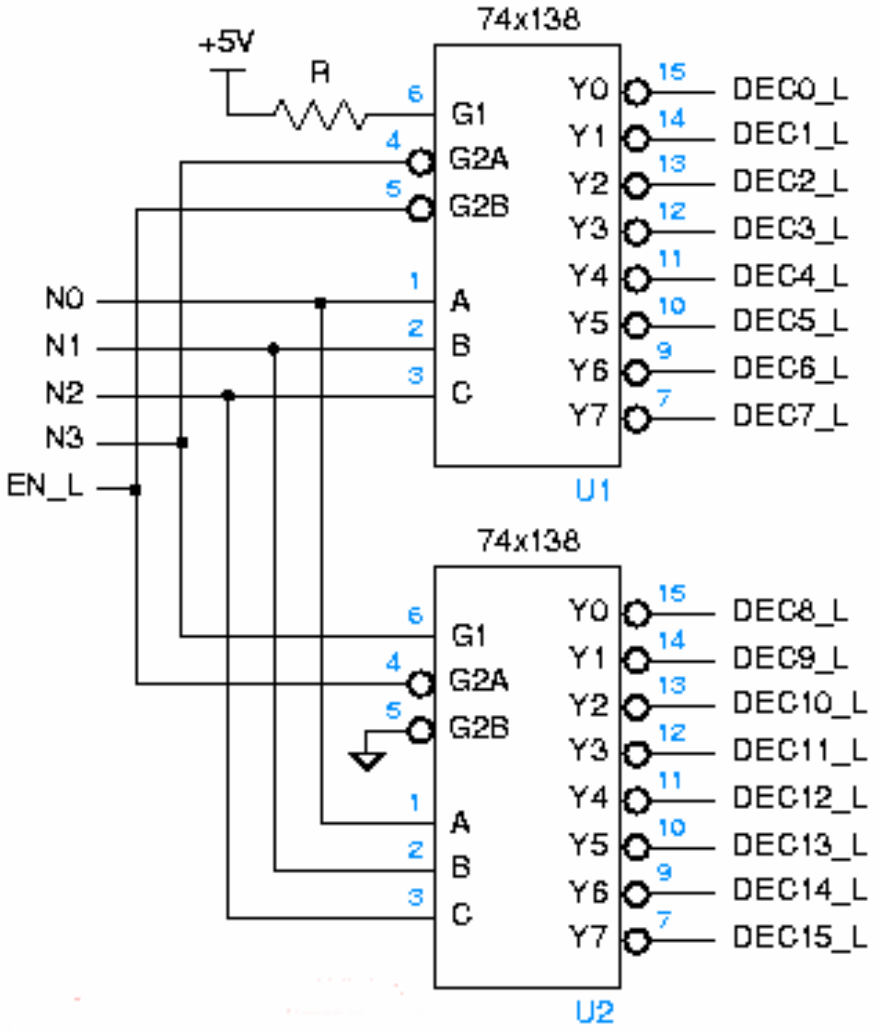
Tabela de verdade do descodificador

Inputs						*Outputs*							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

5. Prática com Sistemas Combinacionais

- Descodificadores (4) -

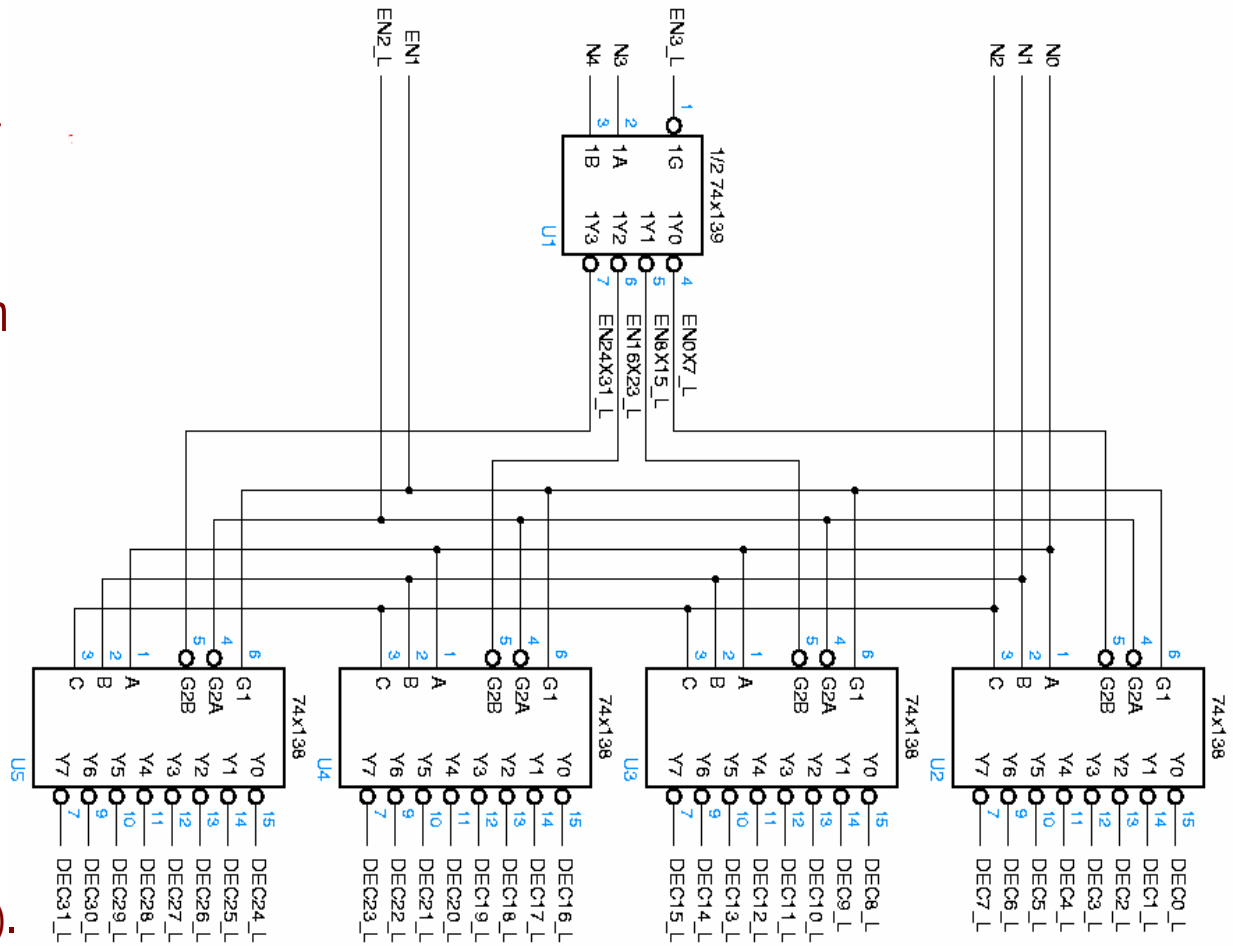
- Utilizando vários descodificadores é possível descodificar entradas com mais bits.
- Exemplo: utilizar 2 descodificadores 3-para-8 para construir um descodificador 4-para-16
- O descodificador de cima (U1) só está habilitado a funcionar quando N3=0 e o descodificador de baixo (U2) quando N3=1. EN_L=0 habilita ambos.



5. Prática com Sistemas Combinacionais

- Descodificadores (5) -

- Para descodificar entradas ainda com mais bits, pode usar-se uma hierarquia de descodificadores.
- Exemplo: pode construir-se um descodificador 5-para-32 com um descodificador 2-para-4 e quatro 3-para-8.
- O descodificador 2-para-4 descodifica os 2 bits mais significativos, gerando 4 *enables* para os descodificadores 3:8.
- Os 4 descodificadores 3-para-8 descodificam os 3 bits menos significativos (0:7, 8:15, 16:23 e 24:31).



5. Prática com Sistemas Combinacionais

- Descodificadores (6) -

- Um descodificador pode ser descrito em VHDL de várias formas.
- A forma mais primitiva *(e pouco legível)* consistiria em usar uma descrição **estrutural** equivalente ao circuito lógico do slide 8.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity V2to4dec is
  port (I0, I1, EN: in STD_LOGIC;
        Y0, Y1, Y2, Y3: out STD_LOGIC );
end V2to4dec;
architecture V2to4dec_a of V2to4dec is
  signal NOTI0, NOTI1: STD_LOGIC;
  component inv port (I: in STD_LOGIC; O: out STD_LOGIC ); end component;
  component and3 port (I0, I1, I2: in STD_LOGIC; O: out STD_LOGIC ); end component;
begin
  U1: inv port map (I0,NOTI0);
  U2: inv port map (I1,NOTI1);
  U3: and3 port map (NOTI0,NOTI1,EN,Y0);
  U4: and3 port map (   I0,NOTI1,EN,Y1);
  U5: and3 port map (NOTI0,   I1,EN,Y2);
  U6: and3 port map (   I0,   I1,EN,Y3);
end V2to4dec_a;
```

5. Prática com Sistemas Combinacionais

- Descodificadores (7) -

- A segunda alternativa (*mais legível*) consistiria em usar uma descrição **fluxo de dados.**
- Exemplo: descodificador 3:8 com enable

```
entity decoder3to8 is
    port (
     A   : in   std_logic_vector(2 downto 0);
     EN : in   std_logic;
     Y   : out std_logic_vector (7 downto 0)  );
end entity decoder3to8;
```

```
architecture dataFlow of decoder3to8 is
    signal Y_i : std_logic_vector (7 downto 0) ;
begin
    with A select Y_i <=
                "00000001"  when  "000",
                "00000010"  when  "001",
                "00000100"  when  "010",
                "00001000"  when  "011",
                "00010000"  when  "100",
                "00100000"  when  "101",
                "01000000"  when  "110",
                "10000000"  when  "111",
                "00000000"  when  others;
    Y <= Y_i when EN='1' else "00000000";
end architecture dataFlow;
```

← sinal auxiliar

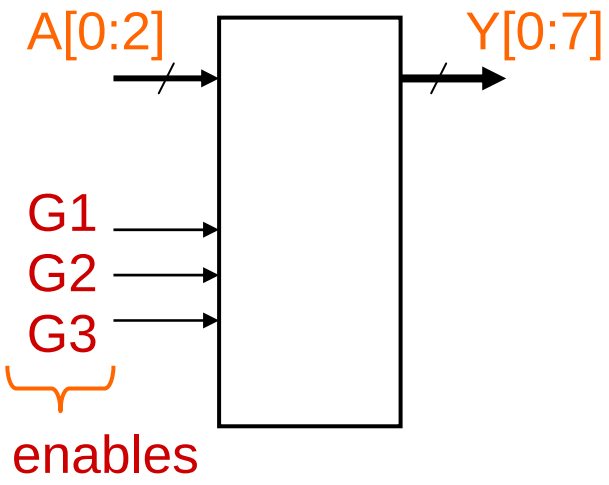
atribuição selectiva

atribuição condicional

5. Prática com Sistemas Combinacionais

- Descodificadores (8) -

- Outra alternativa para o descodificador 3:8 seria uma descrição **comportamental**. Neste exemplo, a entrada *A*, os 3 enables e a saída *Y* são todos activos a 1.




```
architecture V3to8dec_c of V3to8dec is
begin
process (G1, G2, G3, A)
  variable i: INTEGER range 0 to 7;
  begin
    Y <= "00000000";
    if (G1 and G2 and G3) = '1' then
      for i in 0 to 7 loop
        if i=CONV_INTEGER(A) then Y(i) <= '1'; end if;
      end loop;
    end if;
  end process;
end V3to8dec_c;
```

5. Prática com Sistemas Combinacionais

- Descodificadores de 7-segmentos (1) -

- Os **visores de 7-segmentos** usam-se em relógios, calculadoras e outros instrumentos que precisem mostrar informação digital.
- Um dígito é desenhado iluminando alguns dos 7 segmentos A a G.

- Um **descodificador de 7-segmentos** recebe como entrada um dígito BCD com 4-bits e tem como saídas os 7 sinais que fazem iluminar cada um dos segmentos.

5. Prática com Sistemas Combinacionais

- Descodificadores de 7-segmentos (2) -

Inputs					Outputs						
BI_L	D	C	B	A	a	b	c	d	e	f	g
0	x	x	x	x	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
1	0	0	1	0	1	1	0	1	1	0	1
1	0	0	1	1	1	1	1	1	0	0	1
1	0	1	0	0	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1	1	0	1	1
1	0	1	1	0	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0

- Descodificador 7-segmentos do CI 74x49

- TPC 1:
  Obter as expressões minimizadas para as saídas do descodificador de 7-segmentos ao lado usando mapas de Karnaugh *(sem BI_L)*

- TPC 2:
  Escrever em VHDL uma descrição fluxo de dados para o descodificador de 7-segmentos.

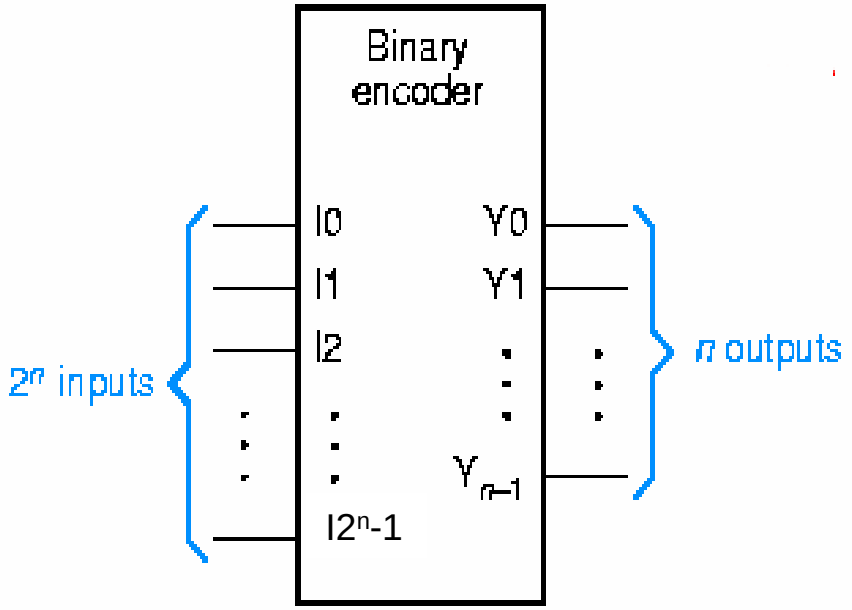
BI_L → permite que os segmentos iluminem (BI_L=1) ou não (BI_L=0)

5. Prática com Sistemas Combinacionais

- Codificadores (1) -

- Um **codificador** é um circuito que codifica uma entrada numa saída com um número de bits normalmente inferior ao da entrada.
- O codificador mais simples de construir é o codificador **2^n-para-n** ou **binário**. A sua funcionalidade é oposta à do descodificador binário.

- Codificador 2^n-para-n



5. Prática com Sistemas Combinacionais

- Codificadores (2) -

- Assume-se que apenas 1 entrada está activa em cada instante.
- Equações do codificador 8-para-3:

  Y0 = I1 + I3 + I5 + I7
  Y1 = I2 + I3 + I6 + I7
  Y2 = I4 + I5 + I6 + I7

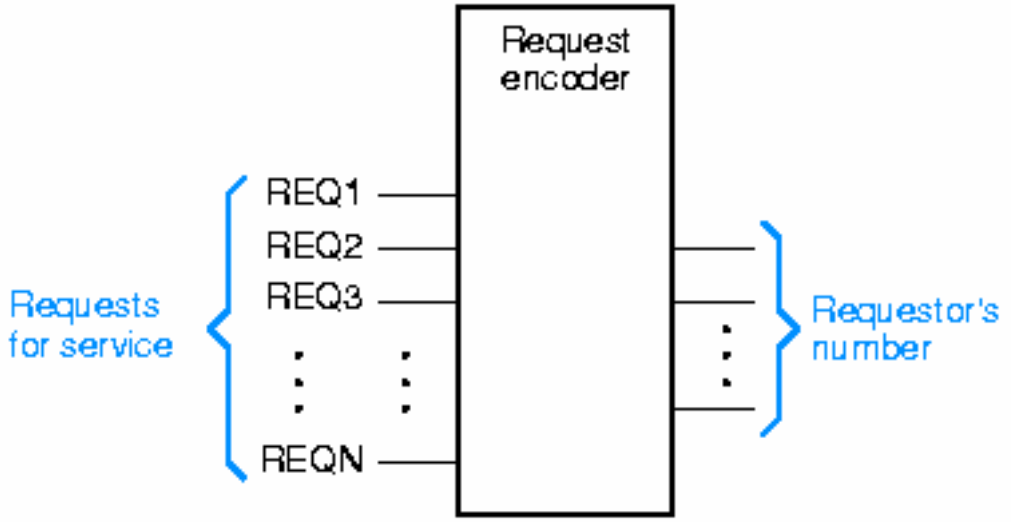
- O que acontece se 2 entradas estiverem activas (por ex. I2 e I4)? Porquê?

I7 I6 I5 I4 I3 I2 I1 I0	Y2 Y1 Y0
0 0 0 0 0 0 0 1	0 0 0
0 0 0 0 0 0 1 0	0 0 1
0 0 0 0 0 1 0 0	0 1 0
0 0 0 0 1 0 0 0	0 1 1
0 0 0 1 0 0 0 0	1 0 0
0 0 1 0 0 0 0 0	1 0 1
0 1 0 0 0 0 0 0	1 1 0
1 0 0 0 0 0 0 0	1 1 1

5. Prática com Sistemas Combinacionais

- Codificadores (3) -

- Para implementar um **codificador de pedidos** (*request encoder*), o codificador binário não funciona porque assume que apenas 1 entrada está activa.

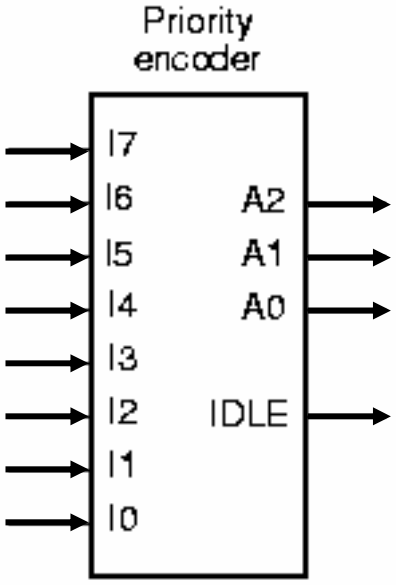


- Se for possível efectuar multiplos pedidos em simultâneo, deve atribuir-se uma prioridade a cada sinal de entrada REQ_i .
- Quando se efectuam multiplos pedidos, o dispositivo (**codificador de prioridade**) coloca na saída o número do pedido mais prioritário.

5. Prática com Sistemas Combinacionais

- Codificadores (4) -

- A entrada *I7* é a que possui maior prioridade.
- Nas saídas *A2:A0* é colocado o número da entrada mais prioritária activa, se houver alguma activa.
- A saída *IDLE* indica se há (*IDLE=0*) ou não alguma entrada activa (*IDLE=1*).
- Para obter as expressões das saídas, usam-se as variáveis intermédias *H0:H7*.
- *Hi* é 1, se *Ii* fôr a entrada a 1 mais prioritária:
  H7 = I7 H6 = I6·I7'
  H5 = I5·I6'·I7' H4 = I4·I5'·I6'·I7'
  ... (equações idênticas para H3:H0)

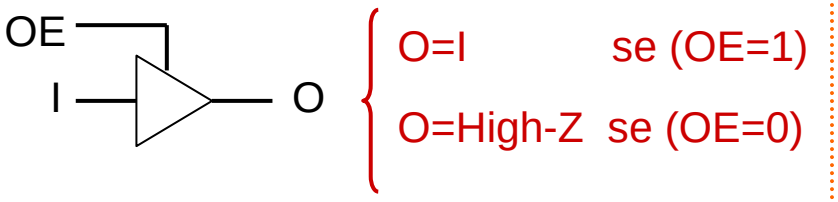


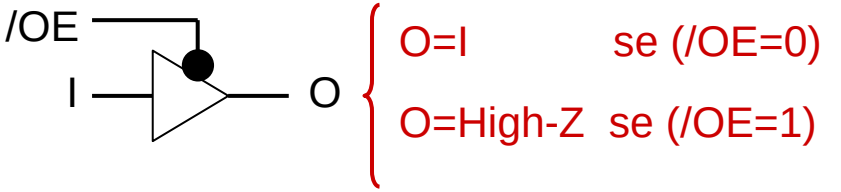
- A0 = H1 + H3 + H5 + H7
  A1 = H2 + H3 + H6 + H7
  A2 = H4 + H5 + H6 + H7
- IDLE= I0'·I1'·I2'·I3'·I4'·I5'·I6'·I7'

5. Prática com Sistemas Combinacionais

- Buffers Tri-state (1) -

- Algumas saídas de dispositivos podem assumir um 3º estado eléctrico, que não é 0 nem 1, mas sim alta impedância (high-Z) ou estado flutuante.
- Este estado equivale a ter a saída desligada do resto do circuito.
- Os dispositivos que têm uma saída passível de estar em alta impedância possuem uma entrada de controlo (*output enable*) que define se a saída está em High-Z ou não.
- Um ***buffer tri-state*** é um circuito que coloca na saída a entrada ou então coloca a saída em alta impedância.





5. Prática com Sistemas Combinacionais

- Buffers Tri-state (2) -

- Os *buffers tri-state* permitem ligar várias saídas a um mesmo ponto dum circuito, desde que apenas um OE esteja activo em cada instante.
- Esta situação é necessária quando vários dispositivos ligam a um barramento comum.
- Descrição dum *buffer tri-state* em VHDL:

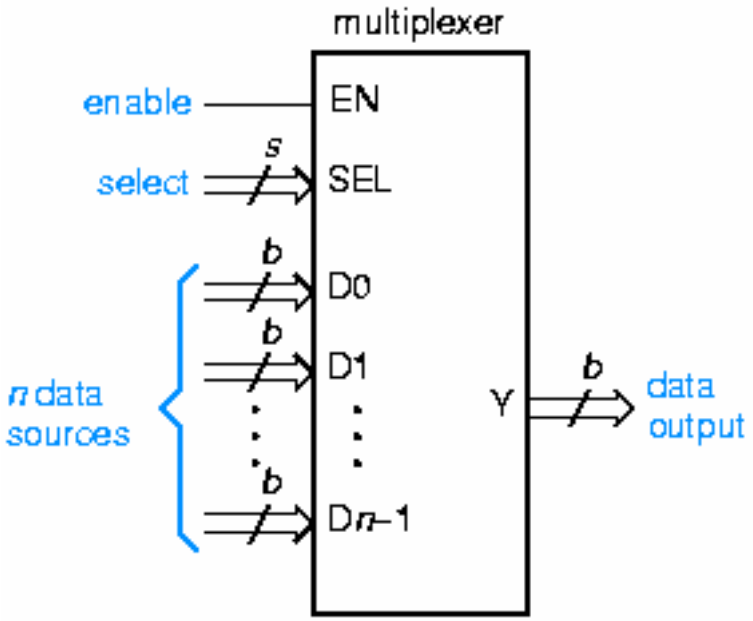
Usa-se o nível 'Z' do tipo STD_LOGIC para representar o estado High-Z.

```
architecture funcional1 of bufTriState is
begin
   O <= I when OE='1' else 'Z';
end funcional1;
```

5. Prática com Sistemas Combinacionais

- Multiplexadores (1) -

- Um **multiplexador** (ou apenas **mux**) é um comutador digital que encaminha a informação de uma das **n** entradas para a única saída. →
- A entrada SEL, com **s** bits, permite seleccionar uma de entre **n** entradas, logo $s = \lceil \log_2 n \rceil$.
- Quando o *enable* está inactivo (EN=0), Y=0. Quando está activo (EN=1), o mux funciona.

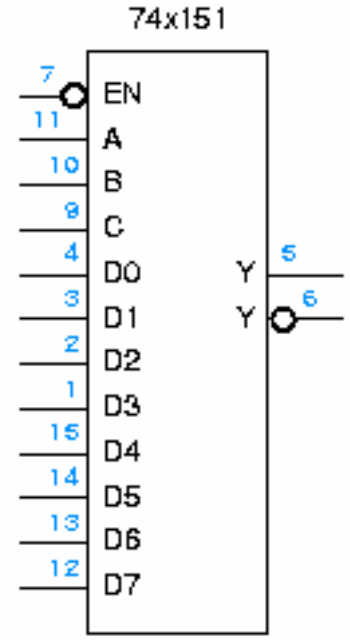


- Os multiplexadores são usados nos computadores, entre os registos do processador e a ALU, para seleccionar os registos que ligam às entradas da ALU.

5. Prática com Sistemas Combinacionais

- Multiplexadores (2) -

- O CI 74x151 implementa um multiplexador 8:1 (8 entradas de 1-bit: D0..D7).
- As entradas selectoras são A,B e C, em que C é o bit mais significativo (msb).
- A entrada de *enable* EN_L é activa a zero.
- O CI disponibiliza duas versões da saída: uma activa a zero (Y) e outra activa a 1 (Y_L).

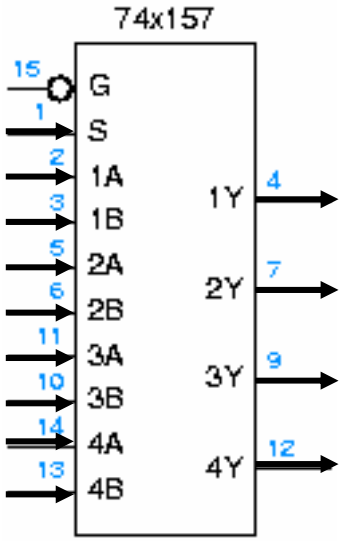


Inputs				Outputs	
EN_L	C	B	A	Y	Y_L
1	X	X	X	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′

5. Prática com Sistemas Combinacionais

- Multiplexadores (3) -

- Exemplo: multiplexador 2:1 de 4-bits.
- A entrada selectora é S.
- A entrada de *enable* G_L é activa a zero.
- A notação da tabela de verdade foi estendida para que as entradas apareçam nas colunas das saídas, tornando a tabela mais clara e mais compacta.

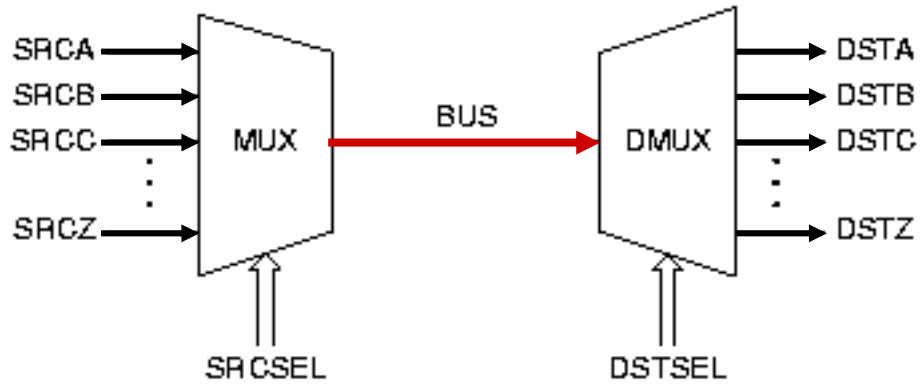


Inputs		Outputs			
G_L	S	1Y	2Y	3Y	4Y
1	X	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

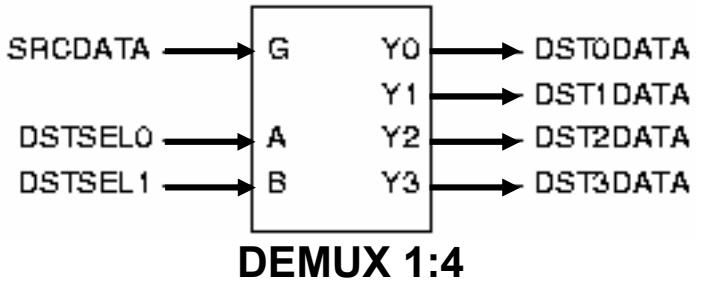
5. Prática com Sistemas Combinacionais

- Multiplexadores (4) -

- Pode usar-se um multiplexador para seleccionar uma de **n** origens dos dados a enviar para um barramento.
- Por outro lado, pode usar-se um **desmultiplexador** para encaminhar os dados que fluem num barramento para um de **m** destinos. →



- A funcionalidade dum desmultiplexador é inversa daquela que apresenta um multiplexador.
- Um desmultiplexador de 1-bit e n-saídas, possui uma entrada de dados e s entradas selectoras para escolher para qual das $n=2^s$ saídas encaminha a única entrada de dados. →



DEMUX 1:4

5. Prática com Sistemas Combinacionais

- Multiplexadores (5) -

- Exemplo em VHDL: MUX 4:1 com entradas de 8 bits (A, B, C e D).
- Estilo fluxo de dados com uma atribuição selectiva.

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

entity mux4in8b is
    port (
        S: in STD_LOGIC_VECTOR (1 downto 0);      -- Select inputs, 0-3 ==> A-D
        A, B, C, D: in STD_LOGIC_VECTOR (1 to 8); -- Data bus input
        Y: out STD_LOGIC_VECTOR (1 to 8)          -- Data bus output
    );
end mux4in8b;

architecture mux4in8b of mux4in8b is
begin
    with S select Y <=
      A when "00",
      B when "01",
      C when "10",
      D when "11",
      (others => 'U') when others; -- this creates an 8-bit vector of 'U'
end mux4in8b;
```

5. Prática com Sistemas Combinacionais

- Multiplexadores (6) -

- Exemplo em VHDL: MUX 4:1 com entradas de 8 bits (A, B, C e D).
- Estilo comportamental com uma instrução CASE.

```
architecture mux4in8p of mux4in8b is
begin
process(S, A, B, C, D)
  begin
    case S is
      when "00" => Y <= A;
      when "01" => Y <= B;
      when "10" => Y <= C;
      when "11" => Y <= D;
      when others => Y <= (others => 'U');  -- 8-bit vector of 'U'
    end case;
  end process;
end mux4in8p;
```

- Em VHDL é fácil descrever um multiplexador que apresente uma estratégia de selecção da entrada a colocar na saída bem particular e rebuscada→ usando as potencialidades do CASE / SELECT e da condição por defeito [exemplo: *(others => 'U')*].

5. Prática com Sistemas Combinacionais

- Portas XOR e circuitos detectores de paridade (1) -

- Uma porta **OR-exclusivo** (XOR) é uma porta de 2-entradas cuja saída é 1, se exactamente uma das entradas for 1.
- Uma porta XOR gera um 1 na saída se as entradas forem diferentes.

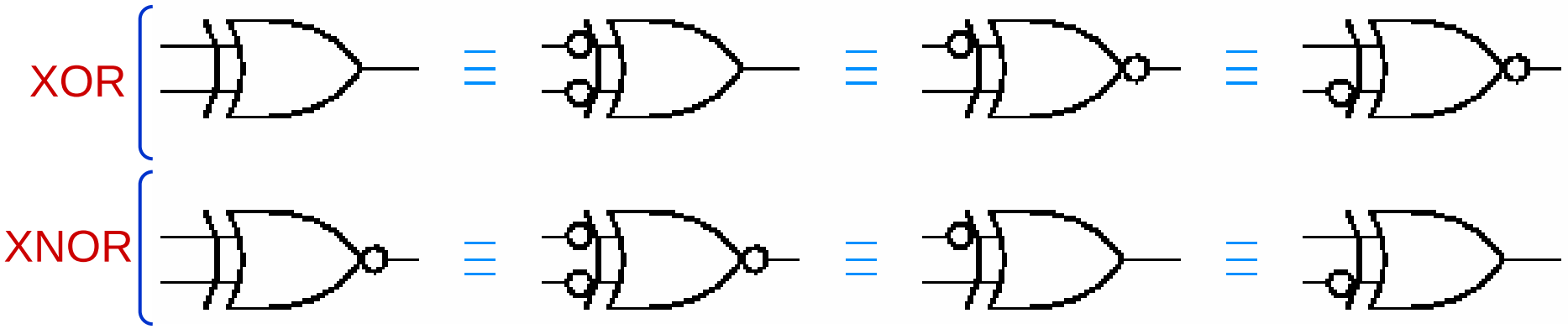
X	Y	X ⊕ Y (XOR)	$\overline{(X \oplus Y)}$ (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

- Uma porta **NOR-exclusivo** (XNOR) é precisamente o oposto: gera um 1 na saída se as entradas forem iguais.
- A operação XOR é por vezes representada pelo símbolo ⊕.
- $X \oplus Y = X' \cdot Y + X \cdot Y'$
- $(X \oplus Y)' = X' \cdot Y' + X \cdot Y$

5. Prática com Sistemas Combinacionais

- Portas XOR e circuitos detectores de paridade (2) -

- Pode usar-se um de 4 símbolos para representar uma porta XOR e XNOR:



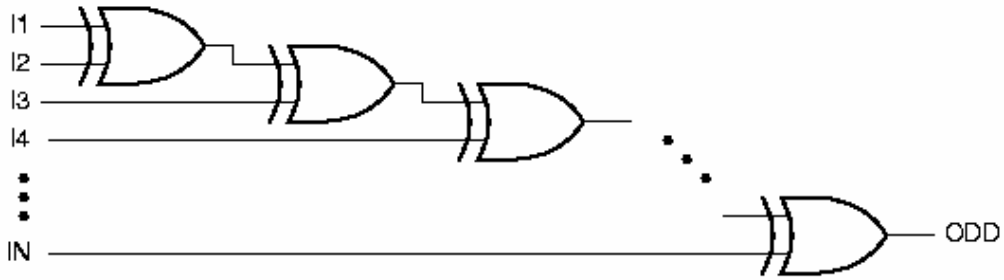
- Estas alternativas resultam da aplicação da seguinte regra:
  - Quaisquer 2 sinais (entradas ou saída) duma porta XOR ou XNOR podem ser complementados sem que a função lógica implementada seja alterada.
- Escolhe-se o símbolo que é mais representativo da função lógica implementada.

5. Prática com Sistemas Combinacionais

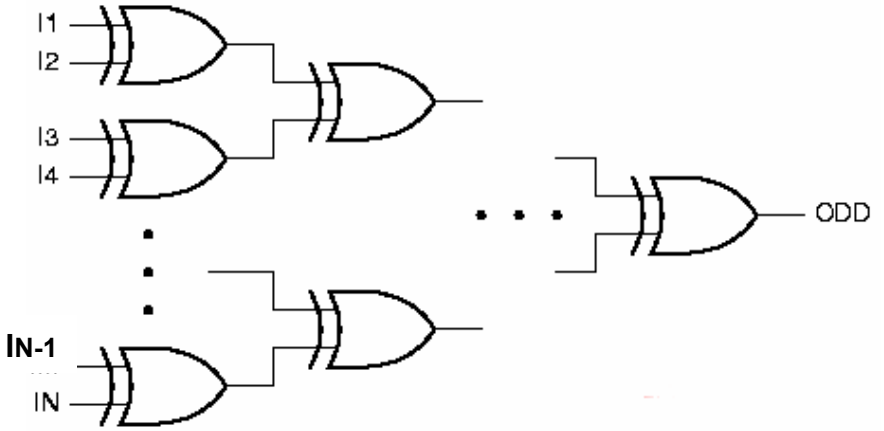
- Portas XOR e circuitos detectores de paridade (3) -

- Ao colocar N-1 portas XOR em cascata constroi-se um circuito com N entradas e uma única saída que funciona como um **detector de paridade impar**, que gera um 1 na saída quando tiver um número impar de entradas a 1.
- Se negarmos a saída de qualquer dos 2 circuitos, obtém-se um **detector de paridade par**, em que a saída é 1 quando o circuito tiver um número par de entradas a 1.

estrutura *daisy-chain*



estrutura tipo árvore (+ rápida)



5. Prática com Sistemas Combinacionais

- Portas XOR e circuitos detectores de paridade (4) -

- O VHDL dispõe de operadores **`xor`** e **`xnor`** nativos.
- Uma porta XOR de 3-entradas pode ser especificada pelo seguinte código VHDL em estilo fluxo de dados.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity vxor3 is
  port (A, B, C: in STD_LOGIC;
        Y: out STD_LOGIC);
end vxor3;

architecture vxor3 of vxor3 is
begin
  Y <= A xor B xor C;
end vxor3;
```

5. Prática com Sistemas Combinacionais

- Portas XOR e circuitos detectores de paridade (5) -

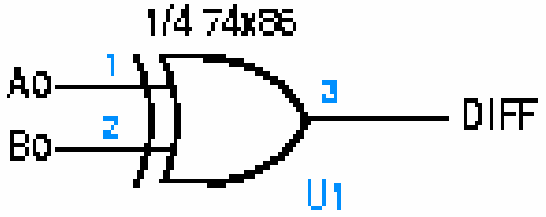
- Exemplo em VHDL comportamental: um detector de paridade com uma entrada de 9-bits.
- Pode acontecer que esta descrição comportamental não seja eficientemente sintetizada pela ferramenta de síntese, o que pode implicar ter que recorrer a uma descrição estrutural.

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

entity parity9 is
  port (I: in STD_LOGIC_VECTOR (1 to 9);
        EVEN, ODD: out STD_LOGIC);
end parity9;

architecture parity9p of parity9 is
begin
process (I)
  variable p : STD_LOGIC;
  begin
    p := I(1);
    for j in 2 to 9 loop
      if I(j) = '1' then p := not p; end if;
    end loop;
    ODD <= p;
    EVEN <= not p;
  end process;
end parity9p;
```

5. Prática com Sistemas Combinacionais

- Comparadores (1) -

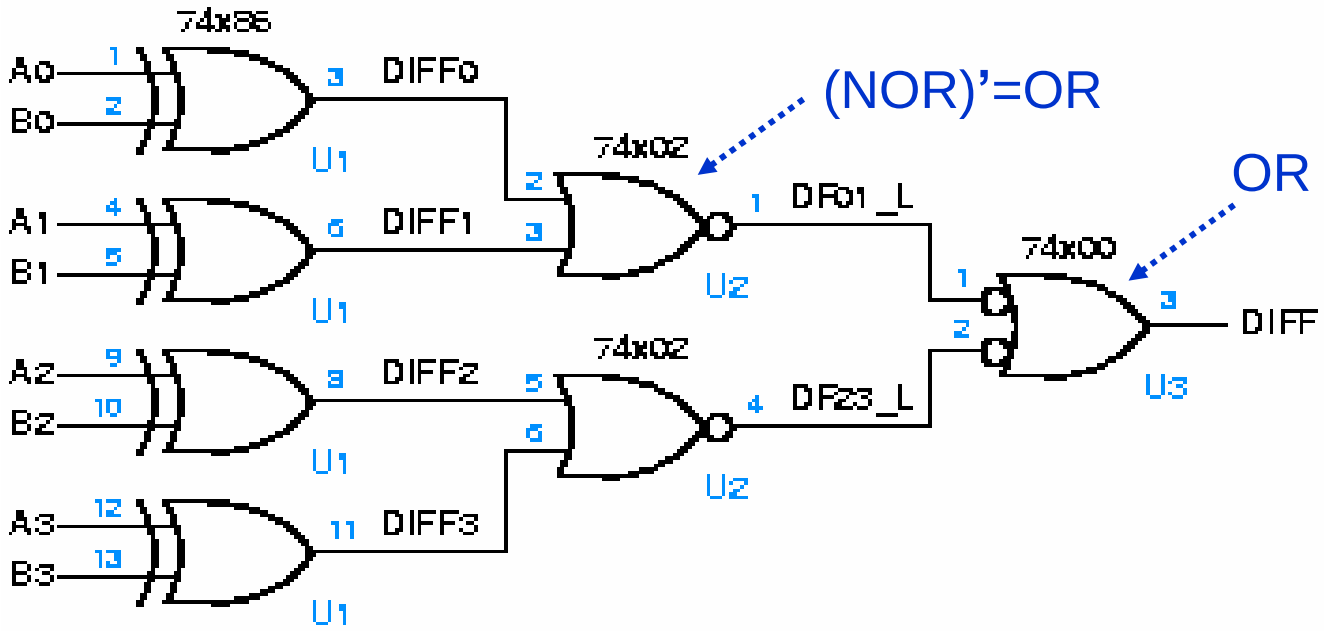
- Comparar 2 palavras binárias é uma operação comum num computador.
- Um circuito que compara 2 palavras binárias e indica se elas são iguais ou diferentes é um **comparador**.
- Alguns comparadores (i) distinguem entre entradas que representam números com e sem sinal e (ii) indicam uma relação aritmética entre as entradas (maior que ou menor que).
- Estes circuitos são geralmente denominados por **comparadores de amplitude**.
- Um porta XOR e XNOR pode ser vista como sendo 1 comparador de 1-bit.
- A saída DIFF é activada quando as entradas forem diferentes.



5. Prática com Sistemas Combinacionais

- Comparadores (2) -

- Exemplo: um comparador de 4-bits efectuando o OR entre as saídas de 4 portas XOR.

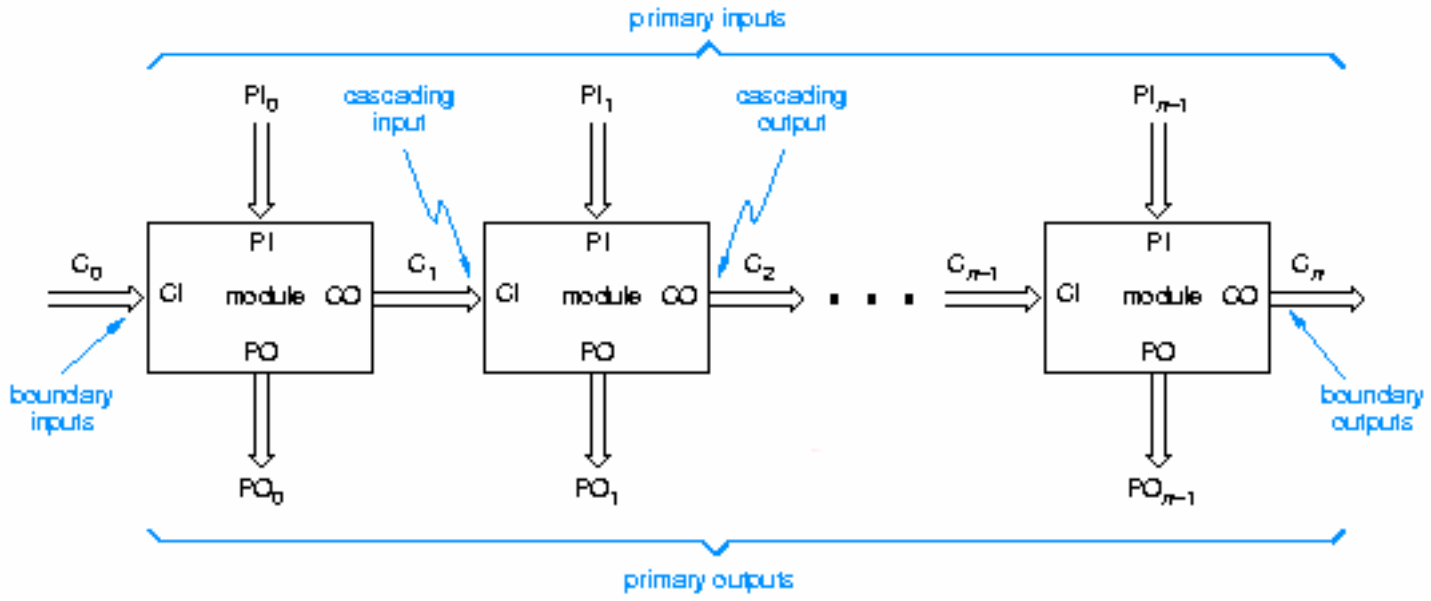


- A saída DIFF é activada quando algum par de bits da entrada (Ai , Bi) for diferente.
- O circuito pode ser facilmente adaptado para funcionar com palavras de qualquer nº de bits.

5. Prática com Sistemas Combinacionais

- Comparadores (3) -

- Um **circuito iterativo** genérico é um circuito combinacional com a seguinte estrutura.



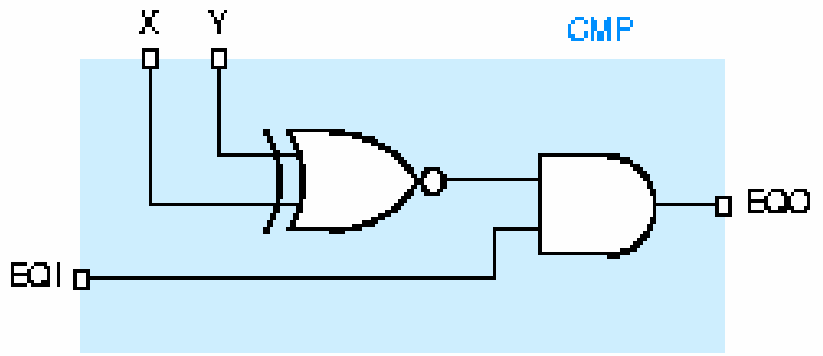
- O circuito é composto por **n** módulos idênticos, cada um com <u>entradas e saídas primárias</u> e ainda com <u>entradas e saídas em cascata</u>.
- As entradas em cascata mais à esquerda (*boundary inputs*) estão normalmente ligadas a valores fixos.

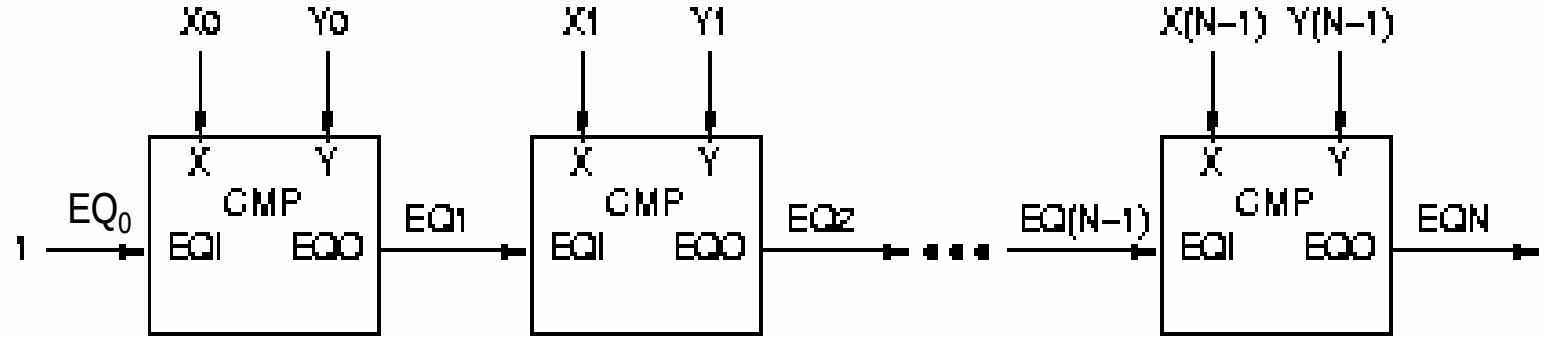
5. Prática com Sistemas Combinacionais

- Comparadores (4) -

- Pode comparar-se dois valores X e Y de n-bits efectuando a comparação bit-a-bit, usando em cada etapa apenas um bit EQ_i que indica se todos os pares de bits comparados até ai são iguais ou não:

- 1. Colocar EQ_0 a 1 e *i* a 0.
  2. Se EQ_i é 1 e $X_i=Y_i$, colocar EQ_{i+1} a 1. Senão colocar EQ_{i+1} a 0.
  3. Incrementar *i*.
  4. Se $i < n$ regressar à etapa 2.



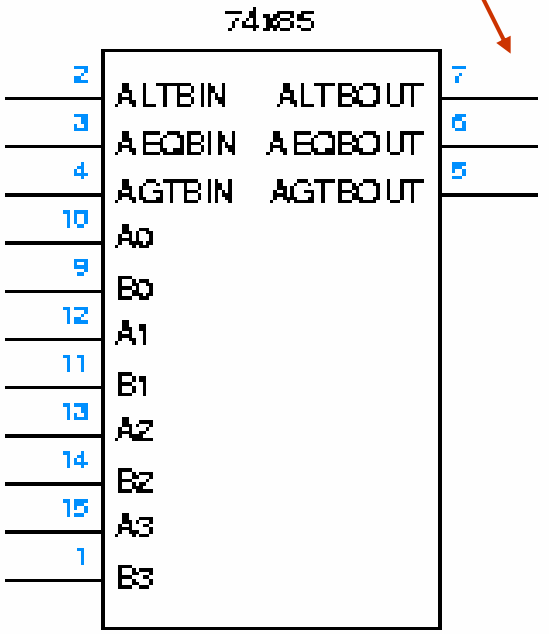


5. Prática com Sistemas Combinacionais

- Comparadores (5) -

apenas uma saída está a 1 em cada instante

- Estão disponíveis no mercado vários comparadores MSI.
- O CI 74x85 implementa um comparador de 4-bits.
- Este CI dispõe de saídas com indicação de "*maior que*", "*menor que*" e "*igual a*".
- O CI 74x85 possui ainda entradas em cascata para combinar multiplos circuitos de modo a construir comparadores com mais do que 4 bits.
- AGTBOUT = (A>B) + (A=B) · AGTBIN
  AEQBOUT = (A=B) · AEQBIN
  ALTBOUT = (A<B) + (A=B) · ALTBIN



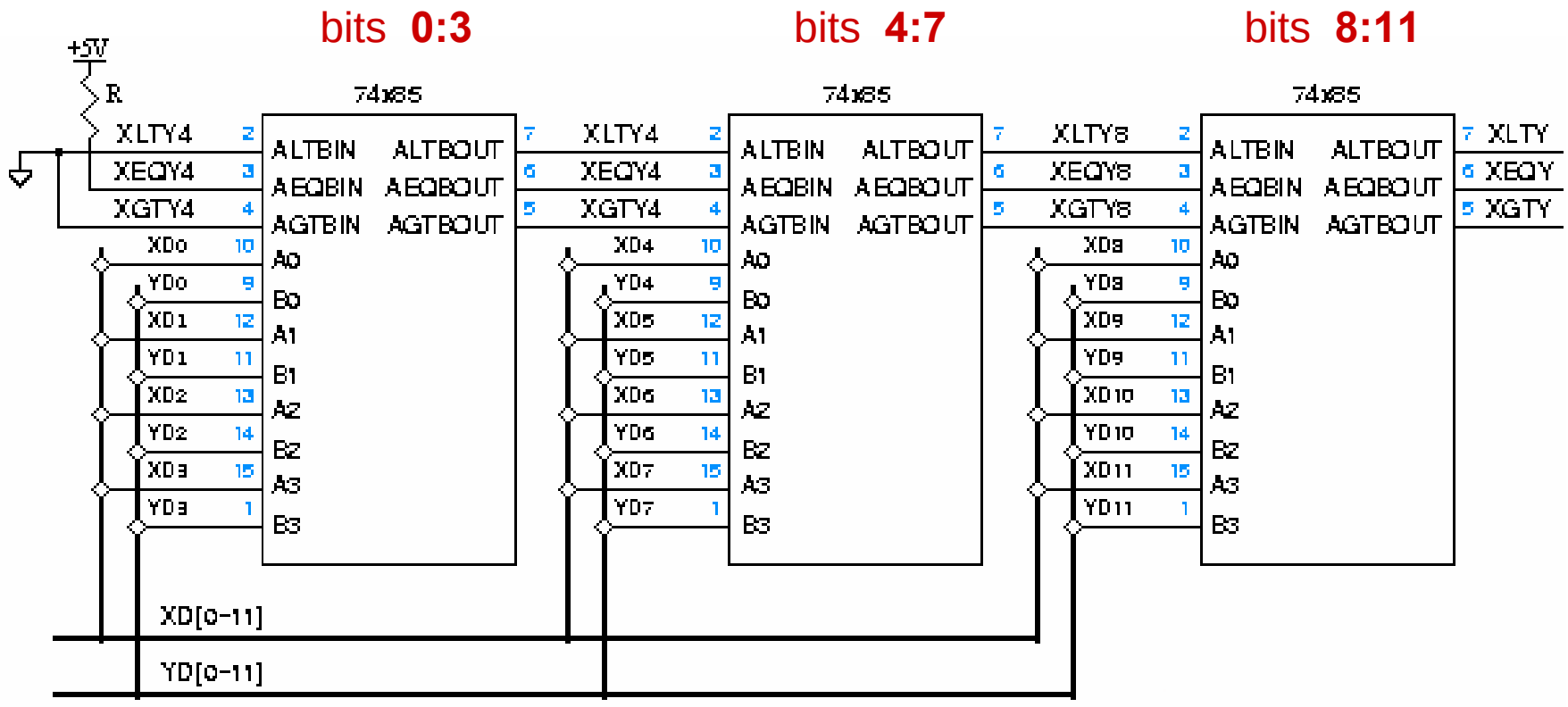
Por exemplo, (A>B) é dado por:

$A3 \cdot B3' + (A3 \oplus B3)' \cdot A2 \cdot B2' + (A3 \oplus B3)' \cdot (A2 \oplus B2)' \cdot A1 \cdot B1' + (A3 \oplus B3)' \cdot (A2 \oplus B2)' \cdot (A1 \oplus B1)' \cdot A0 \cdot B0'$

5. Prática com Sistemas Combinacionais

- Comparadores (6) -

- Com 3 circuitos 74x85, pode construir-se um comparador de 12-bits.



5. Prática com Sistemas Combinacionais

- Comparadores (7) -

- O VHDL dispõe de operadores para comparar qualquer dos tipos de dados nativos.
- Os operadores igualdade (**=**) e desigualdade (**/=**) aplicam-se a todos os tipos.
- Para comparar *arrays* e *records*, os operandos devem ter a mesma dimensão e estrutura, e os operandos são comparados elemento-a-elemento.
- Os outros operadores do VHDL (**>**, **<**, **>=**, **<=**) aplicam-se apenas a inteiros, tipos enumerados e *arrays* unidimensionais de inteiros ou de tipos enumerados.

5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (1) -

- A adição (ou soma) é a operação aritmética mais frequente nos sistemas digitais.
- Um **somador** combina dois operandos aritméticos aplicando as regras da adição.
- Nos números sem sinal e nos números com sinal em complemento para 2 usam-se as mesmas regras da adição, logo os mesmos somadores.
- Um somador pode efectuar a subtracção através da soma do minuendo com o subtraendo complementado.
- Também se pode construir um circuito **subtractor** que efectua de forma directa a subtracção.
- Uma **ALU** (Unidade Aritmética e Lógica) efectua a soma, a subtracção e outras operações lógicas.

5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (2) -

- O somador mais simples, designado por **semi-somador**, soma dois operandos X e Y de 1-bit, produzindo um resultado com 2-bits.
- O resultado da soma varia de 0 a 2, logo é necessário 2 bits para o expressar.
- O bit menos significativo do resultado é designado por HS (*half sum*).
- O bit mais significativo do resultado é designado por CO (*carry out*).
- As equações que definem o resultado da soma são:
  $HS = X \oplus Y = X \cdot Y' + X' \cdot Y$
  $CO = X \cdot Y$
- Para somar operandos com mais do que um bit, é preciso considerar o transporte de uma soma (ao nível do bit) para a seguinte.

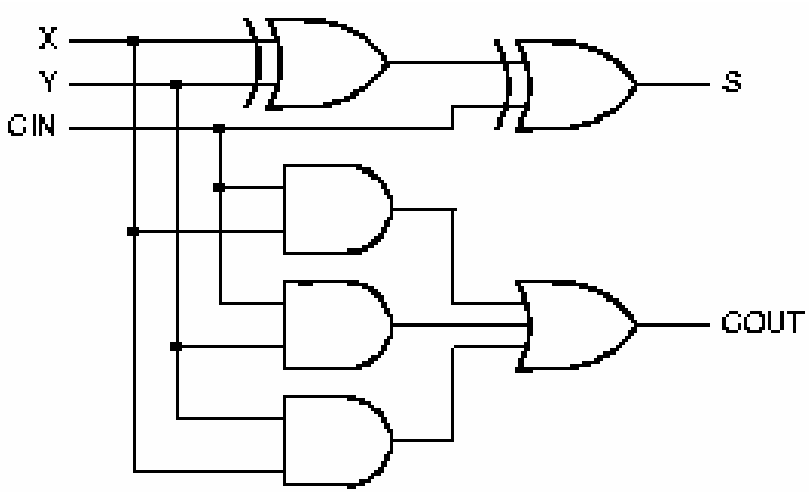
5. Prática com Sistemas Combinacionais

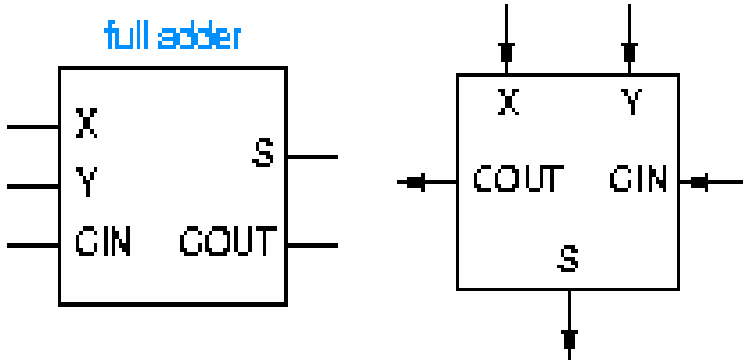
- Somadores, subtractores e ALUs (3) -

- O bloco elementar com que se constroi um circuito que executa esta operação é um **somador completo**.
- Além das entradas X e Y a adicionar, um somador completo possui um bit de transporte como entrada: CIN.
- O resultado da soma dos 3 bits varia de 0 a 3 e pode ser expresso por duas saídas de 1 bit: S e COUT.
- As equações que definem o resultado da soma são:

  $S = X \oplus Y \oplus CIN$

  $COUT = X \cdot Y + X \cdot CIN + Y \cdot CIN$

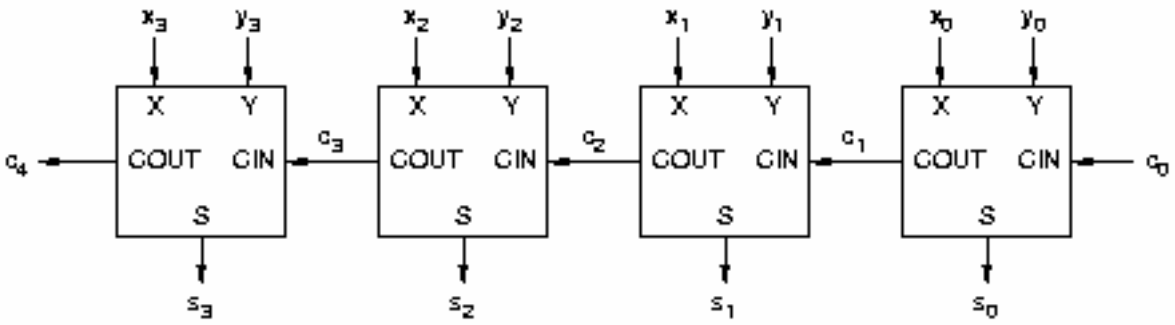




5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (4) -

- Duas palavras em binário, cada uma com **n** bits, podem ser somadas usando um somador de *ripple*.
- Um **somador de *ripple*** é composto por uma cadeia de **n** somadores completos, onde cada somador processa um bit.



É lento porque o transporte é propagado do somador 0 (LS) até ao 3 (MS)

- A entrada de transporte do somador do bit menos significativo (c_0) é colocada normalmente a 0.
- A saída de transporte de cada somador completo é ligada à entrada de transporte do somador completo seguinte e que é mais significativo do que ele.

5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (5) -

- A operação de subtracção binária é análoga à adição binária.
- Um **subtractor completo** possui como entradas X (minuendo), Y (subtraendo) e BIN (*borrow in*) e como saídas D (diferença) e BOUT (*borrow out*).
- As equações que definem o resultado da subtracção são:

  $D = X \oplus Y \oplus BIN$

  $BOUT = X' \cdot Y + X' \cdot BIN + Y \cdot BIN$

  manipulando estas equações obtém-se:

  $D = X \oplus Y' \oplus BIN'$ ← usando $Y \oplus BIN = Y' \oplus BIN'$

  $BOUT' = X \cdot Y' + X \cdot BIN' + Y' \cdot BIN'$ ← T. DeMorgan e distributividade da '+'
- Um subtractor completo pode ser construído com um somador completo:

  $X-Y = X+ (-Y) = X+Y'+1$

  BIN' substitui CIN, logo CIN[0]=0 passa a BIN[0]=1

  Y' é o complemento para 1 de Y

5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (6) -

- Comparando as equações da soma

  $S = X \oplus Y \oplus CIN$

  $COUT = X \cdot Y + X \cdot CIN + Y \cdot CIN$

  com as equações da subtracção

  $D = X \oplus Y' \oplus BIN'$

  $BOUT' = X \cdot Y' + X \cdot BIN' + Y' \cdot BIN'$

- Constata-se que um somador pode funcionar de subtractor se:

  $S \rightarrow D$

  $X \rightarrow X$

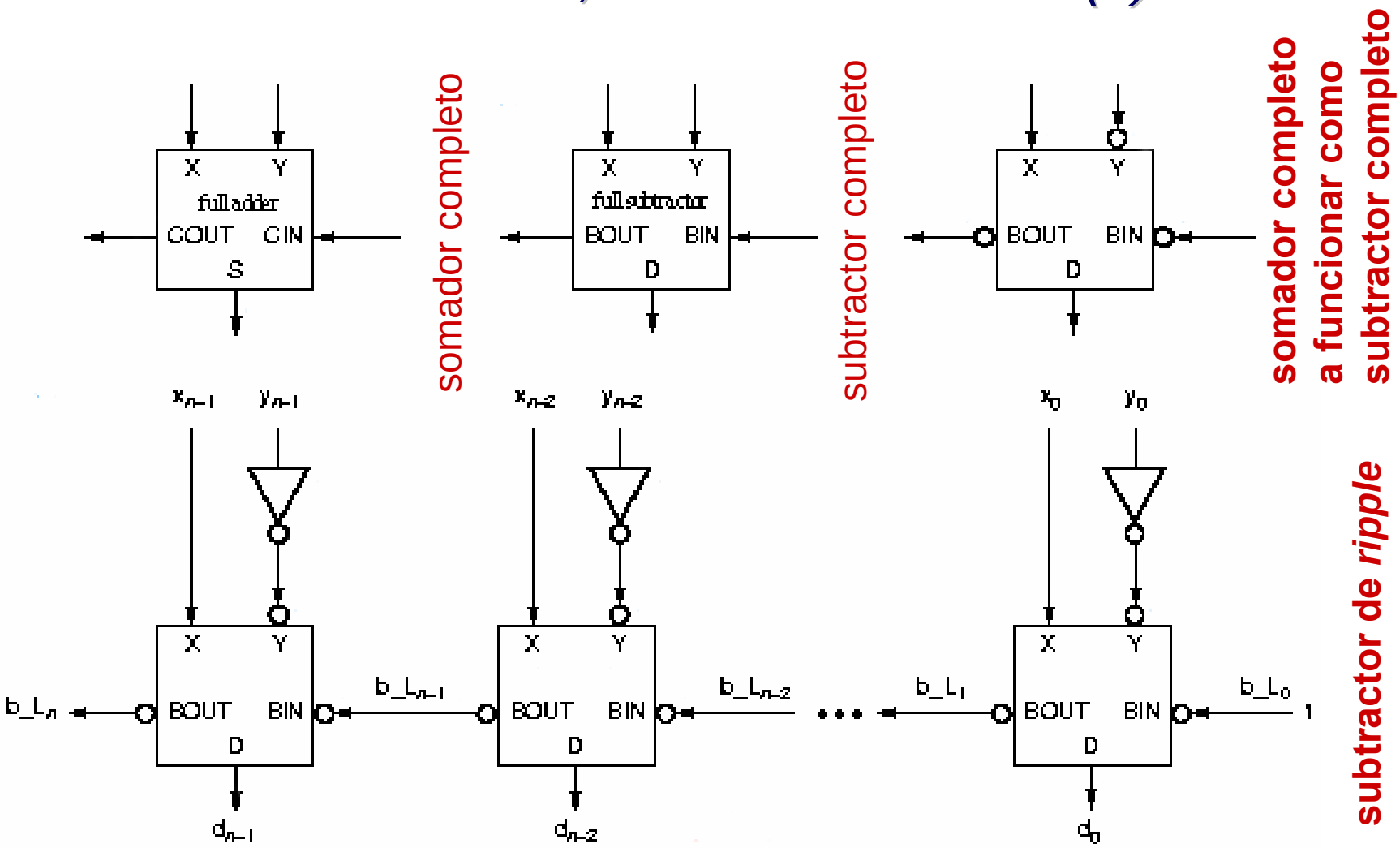
  $Y \rightarrow Y'$

  $CIN \rightarrow BIN'$

  $COUT \rightarrow BOUT'$

5. Prática com Sistemas Combinacionais

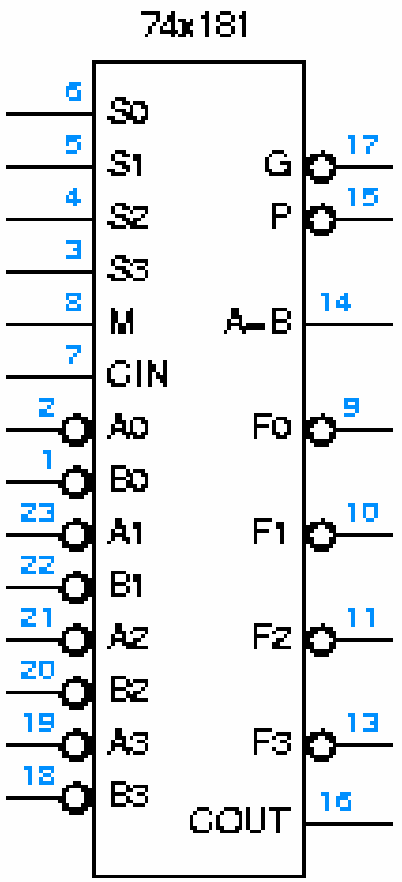
- Somadores, subtractores e ALUs (7) -



5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (8) -

- Uma ALU é um circuito combinacional que pode efectuar diversas operações aritméticas e lógicas com um par de operandos de b-bits.
- A operação a efectuar é especificada por um conjunto de entradas selectoras.
- Tipicamente, uma ALU implementada num CI do tipo MSI possui 2 operandos de 4-bits e 3 a 5 entradas selectoras da operação a efectuar, permitindo escolher uma operação de entre 32 possíveis.
- Exemplo: o CI 74x181 implementa uma ALU de 4-bits.
- A operação a efectuar pela ALU do 74x181 é seleccionada pelas entradas M e S3-S0.



5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (9) -

Inputs				Function	
S3	S2	S1	S0	M = 0 (arithmetic)	M = 1 (logic)
0	0	0	0	F = A minus 1 plus CIN	F = A′
0	0	0	1	F = A · B minus 1 plus CIN	F = A′ + B′
0	0	1	0	F = A · B′ minus 1 plus CIN	F = A′ + B
0	0	1	1	F = 1111 plus CIN	F = 1111
0	1	0	0	F = A plus (A + B′) plus CIN	F = A′ · B′
0	1	0	1	F = A · B plus (A + B′) plus CIN	F = B′
0	1	1	0	F = A minus B minus 1 plus CIN	F = A ⊕ B′
0	1	1	1	F = A + B′ plus CIN	F = A + B′
1	0	0	0	F = A plus (A + B) plus CIN	F = A′ · B
1	0	0	1	F = A plus B plus CIN	F = A ⊕ B
1	0	1	0	F = A · B′ plus (A + B) plus CIN	F = B
1	0	1	1	F = A + B plus CIN	F = A + B
1	1	0	0	F = A plus A plus CIN	F = 0000
1	1	0	1	F = A · B plus A plus CIN	F = A · B′
1	1	1	0	F = A · B′ plus A plus CIN	F = A · B
1	1	1	1	F = A plus CIN	F = A

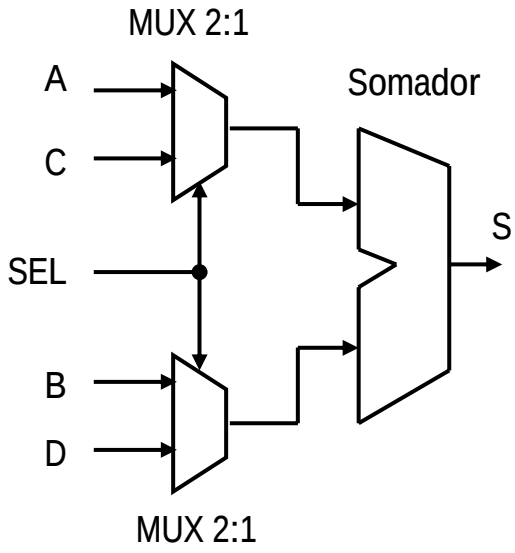
M=1 → operações lógicas que não usam transporte

5. Prática com Sistemas Combinacionais

- Somadores, subtractores e ALUs (10) -

- Circuito descrito em VHDL de modo a que ao sintetizar se partilhe um somador entre duas somas: A+B e C+D
- Poupa-se lógica porque em vez de 2 somadores usa-se 1 somador mais 2 MUX2:1 para seleccionar as entradas (A ou C; B ou D)

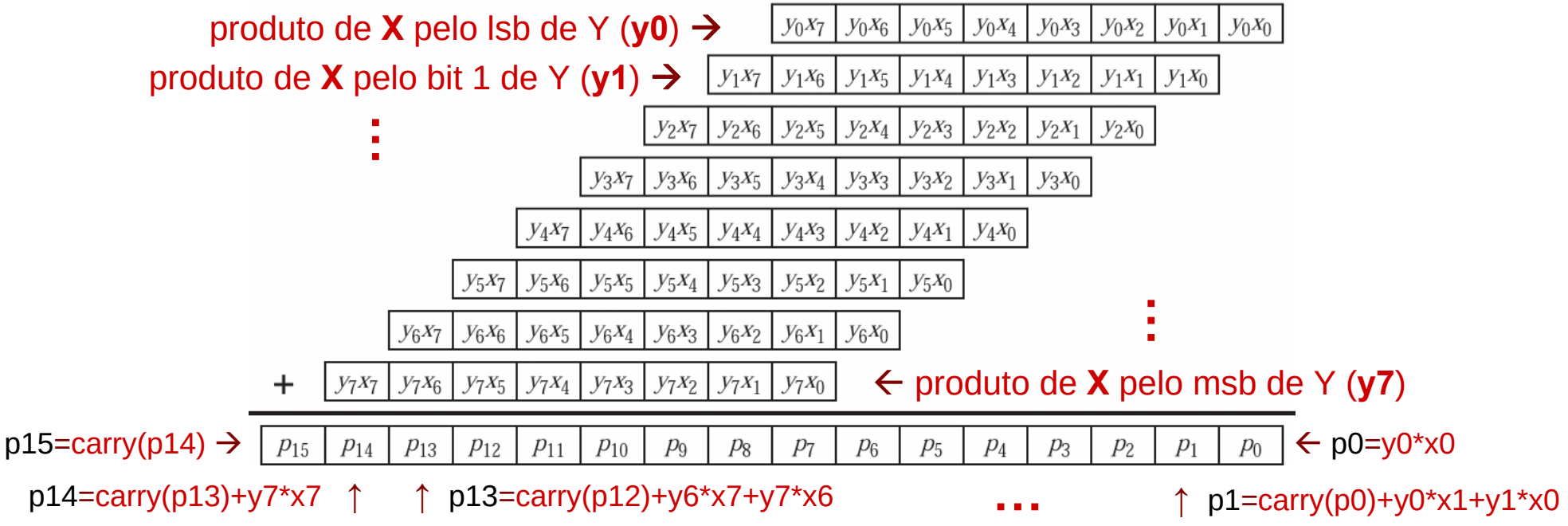
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity vaddshr is
    port (
        A, B, C, D: in SIGNED (7 downto 0);
        SEL: in STD_LOGIC;
        S: out SIGNED (7 downto 0)
    );
end vaddshr;

architecture vaddshr_arch of vaddshr is
begin
    S <= A + B when SEL = '1' else C + D;
end vaddshr_arch;
```



5. Prática com Sistemas Combinacionais

- Multiplicadores (1) -

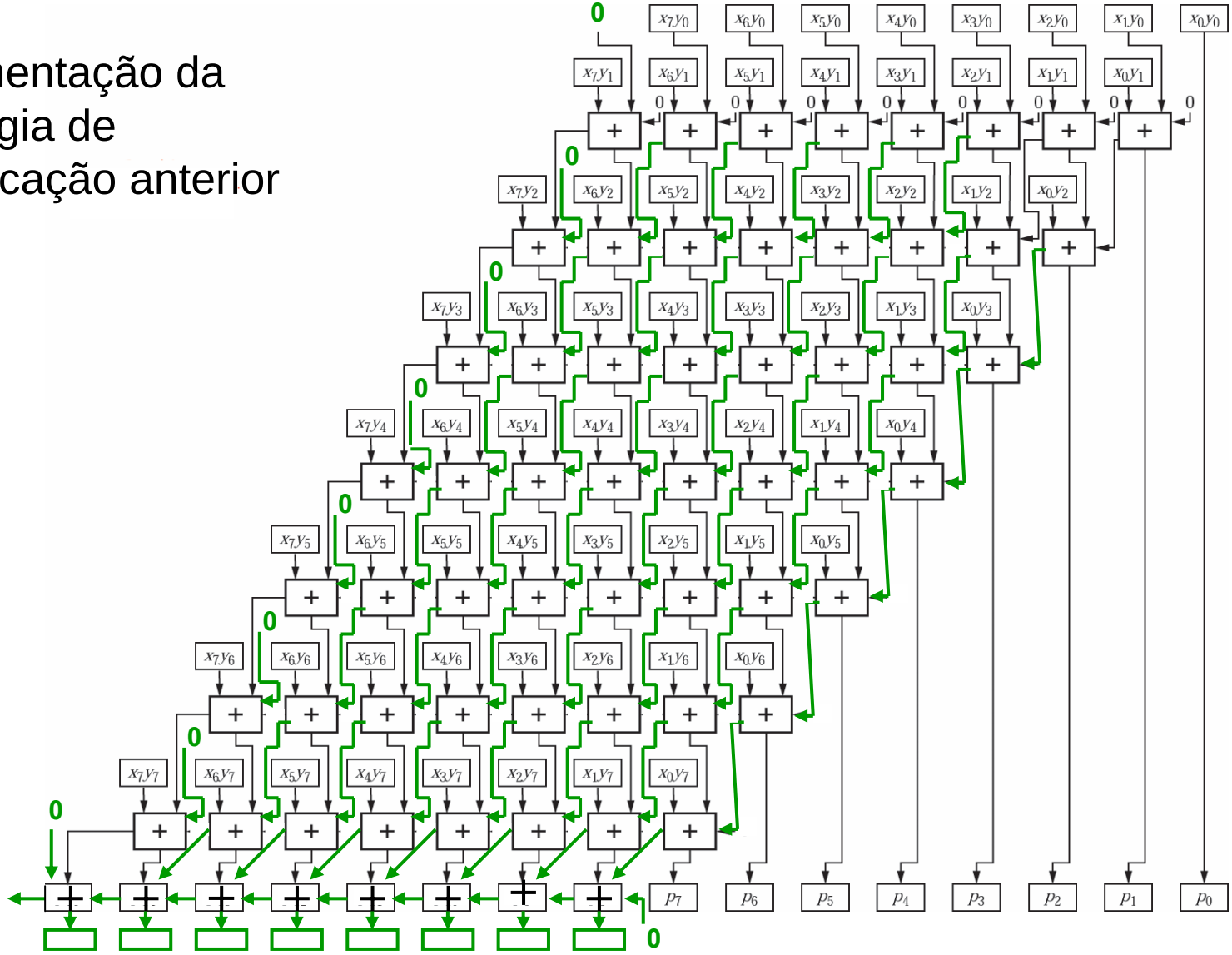
- O algoritmo tradicionalmente usado para multiplicar números binários emprega deslocamentos e somas na obtenção do resultado.
- Contudo, não é essa a única forma possível de implementar a multiplicação.
- Dadas duas entradas de n-bits (X, Y), pode escrever-se a tabela de verdade que expressa o produto P=X*Y de 2n-bits através duma função combinacional de X e Y.
- A maior parte das estratégias de implementação de multiplicadores combinacionais baseiam-se no algoritmo tradicional com deslocamentos e somas.



5. Prática com Sistemas Combinacionais

- Multiplicadores (2) -

Implementação da estratégia de multiplicação anterior



somador completo de 1 bit

O pior atraso é **15** * atraso dum somador completo

5. Prática com Sistemas Combinacionais

- Multiplicadores (3) -

- Pode descrever-se em VHDL comportamental a estratégia de multiplicação anterior, usando uma descrição que imita essa estrutura. Para representar as ligações entre somadores de 1 bit usam-se arrays bi-dimensionais, como se ilustra a seguir:

```
type ARRAY8x8 is array (7 downto 0) of std_logic_vector(7 downto 0);
variable CARRY : ARRAY8x8;
```

5. Prática com Sistemas Combinacionais

- Multiplicadores (4) -

- Contudo a biblioteca IEEE **std_logic_arith** fornece o operador "*****" para operandos com sinal (SIGNED) e sem sinal (UNSIGNED).
- Este operador é descrito usando um algoritmo idêntico ao anterior e permite escrever um programa que multiplica 2 valores sem sinal com uma única linha de código:

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

entity vmul8x8i is
    port (
        X: in UNSIGNED (7 downto 0);
        Y: in UNSIGNED (7 downto 0);
        P: out UNSIGNED (15 downto 0)
    );
end vmul8x8i;

architecture vmul8x8i_arch of vmul8x8i is
begin
  P <= X * Y;
end vmul8x8i_arch;
```