PCS 3115 (PCS2215)

Sistemas Digitais I

Circuitos Combinatórios

Comparadores

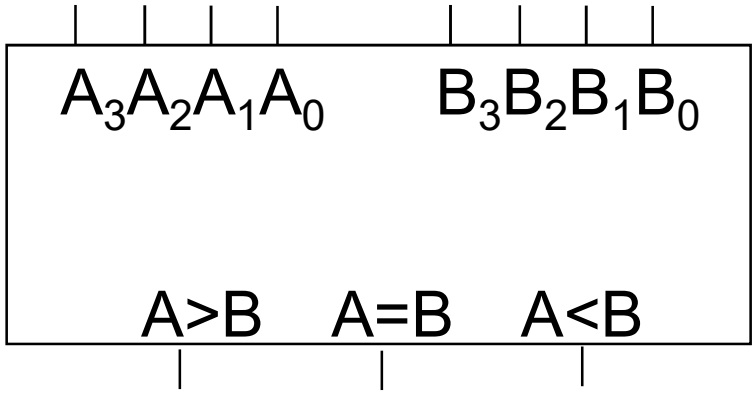
Prof. Dr. Marcos A. Simplicio Jr.

Atualização: Prof. Dr. Marco Túlio Carvalho de Andrade

versão: 5.0 (Abril/2020)

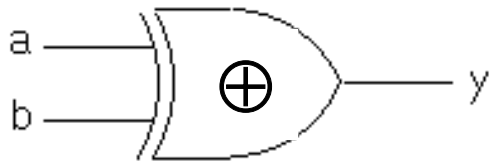
Comparadores

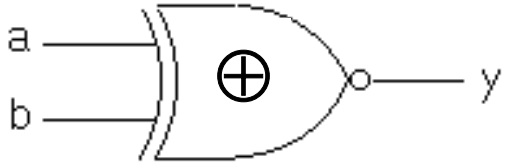
- Comparação entre palavras binárias é uma operação comum em sistemas digitais.
- Comparadores realizam essa função e podem indicar igualdade (= , ≠), e em alguns casos, relação aritmética (> , <).



Comparadores

- OU-exclusivo, NOU-exclusivo
  - São comparadores de 1 bit





- OU-exclusivo (XOR), NOU-exclusivo (XNOR)

- Tabela-verdade

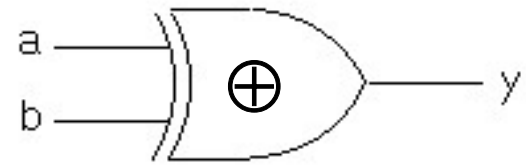
X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

- Tabela-verdade

X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

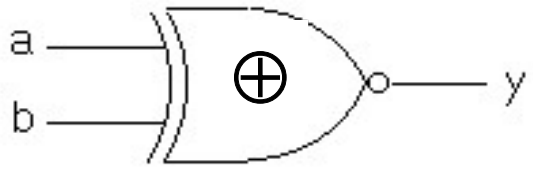
Comparadores

- (Não) OU-exclusivo – X(N)OU
  - São comparadores de 1 bit



Y= a ⊕ b = Dif

Dif = 1, se entradas são diferentes



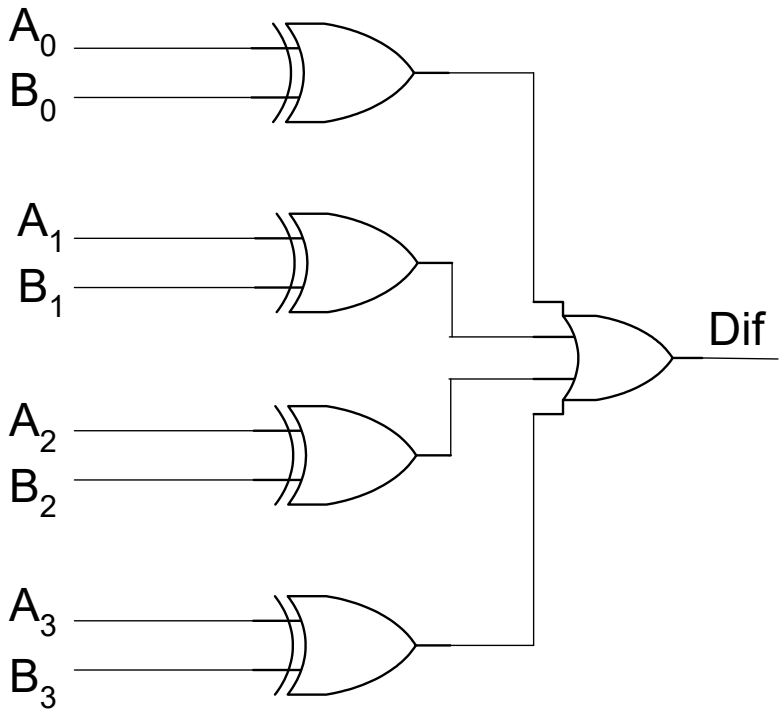
Y= (a ⊕ b)' = Eq

Eq = 1, se entradas são iguais

  - **Pergunta**: como fazer um comparador de n bits...?

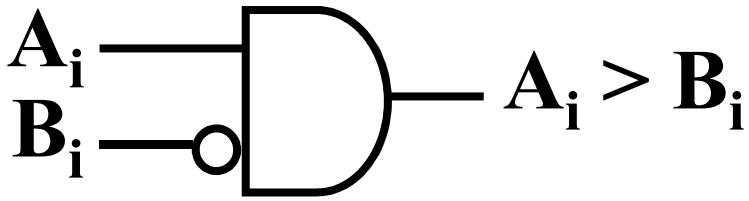
Comparadores

- Comparador (paralelo) de *n* bits
  - Compara bit a bit, duas palavras de *n* bits,
  - Sumariza o resultado



Comparador de magnitude

- Indica relação aritmética entre as palavras de dados comparadas
- **Pergunta** (comparação de 1 bit): Quando Ai > Bi?
  - Resposta: quando Ai = 1 e Bi = 0
- **Pergunta** (comparação de 1 bit): Como detectar que Ai > Bi com portas lógicas?
  - Resposta: Calculando Ai AND NOT(Bi).



Comparador de magnitude

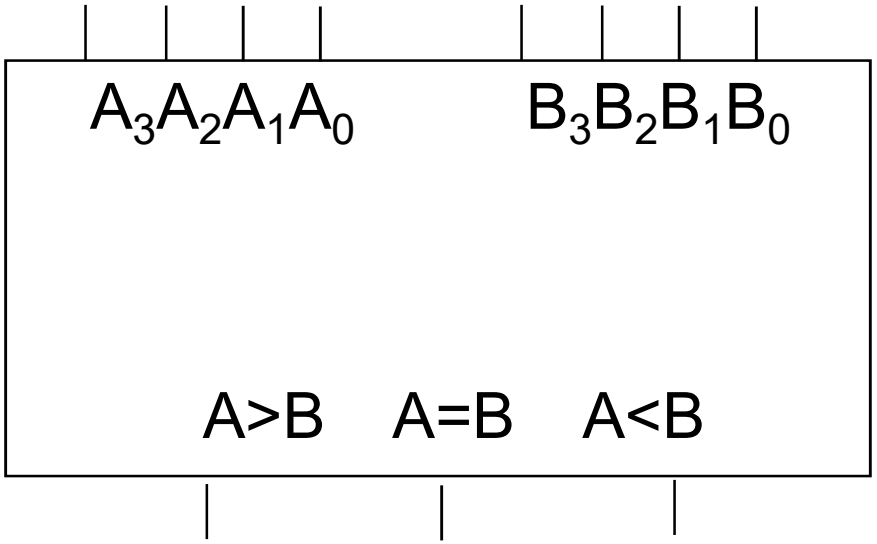
- **Pergunta** (comparação de n bits, sem sinal): como um humano faria essa comparação?
  - **Resposta**: analisar desigualdades dos bits, começando no mais significativo ➔ A > B se
    - A3>B3 (i.e. A3=1 e B3=0)
    - Ou se A3=B3 e A2>B2
    - Ou se A3=B3 e A2=B2 e A1>B1
    - Ou se A3=B3 e A2=B2 e A1=B1 e A0>B0

Comparador de magnitude

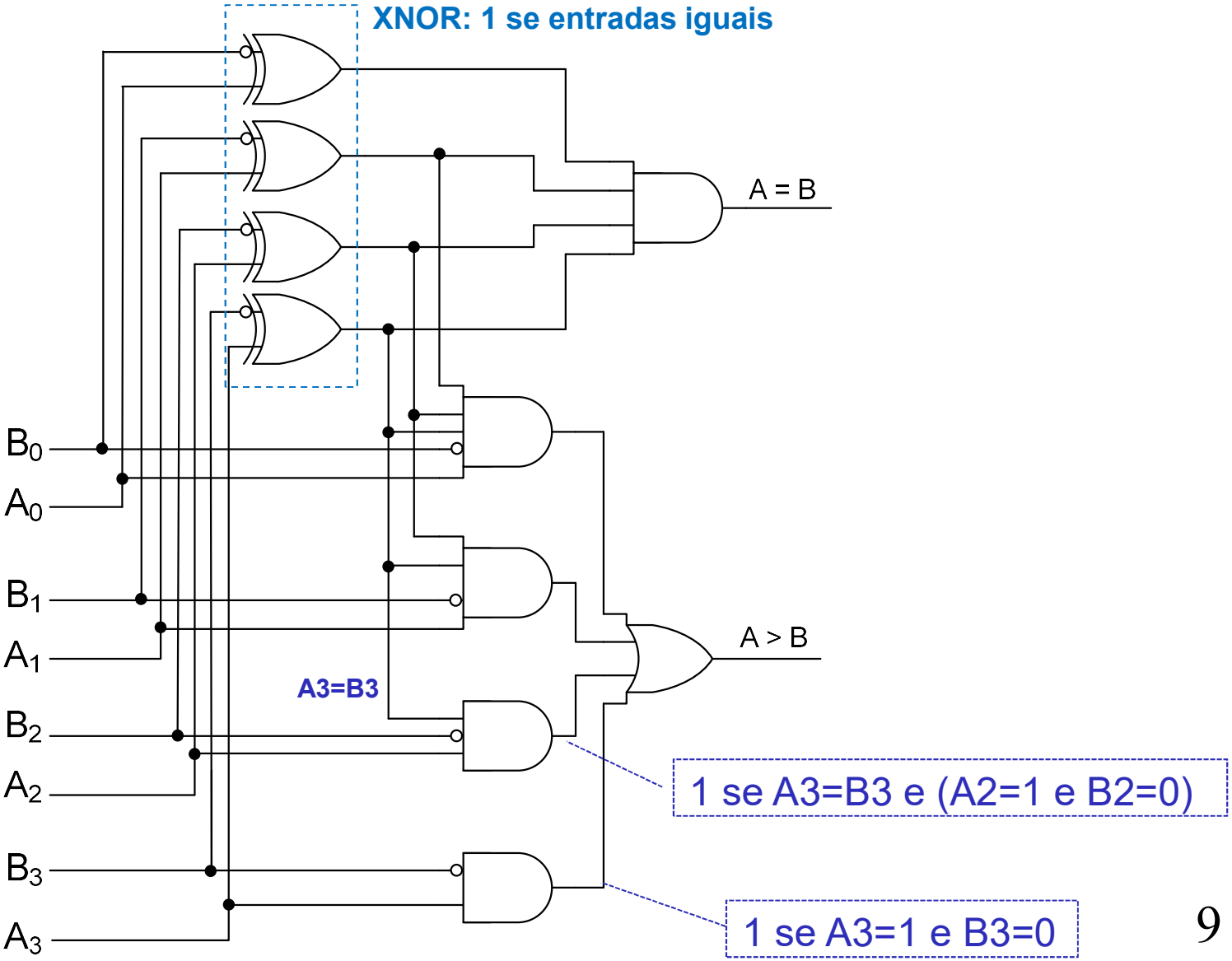
- A=B se
  - A3=B3 e
  - A2=B2 e
  - A1=B1 e
  - A0=B0

Ai > Bi se
Ai=1 e Bi=0

- A>B se
  - **A3>B3**
  - Ou se A3=B3 e **A2>B2**
  - Ou se A3=B3 e A2=B2 e **A1>B1**
  - Ou se A3=B3 e A2=B2 e A1=B1 e **A0>B0**

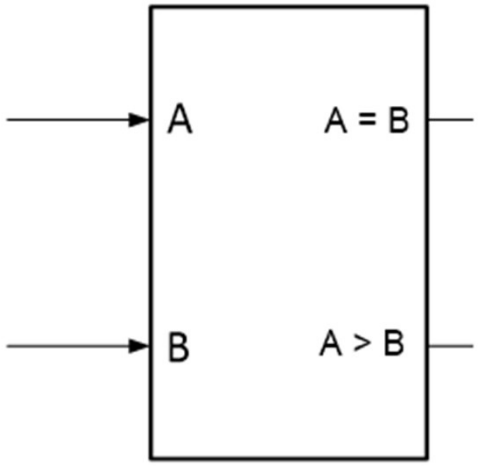


Comparador de magnitude



Comparador de magnitude

- Pergunta: Como obter as demais relações?



A dif. B

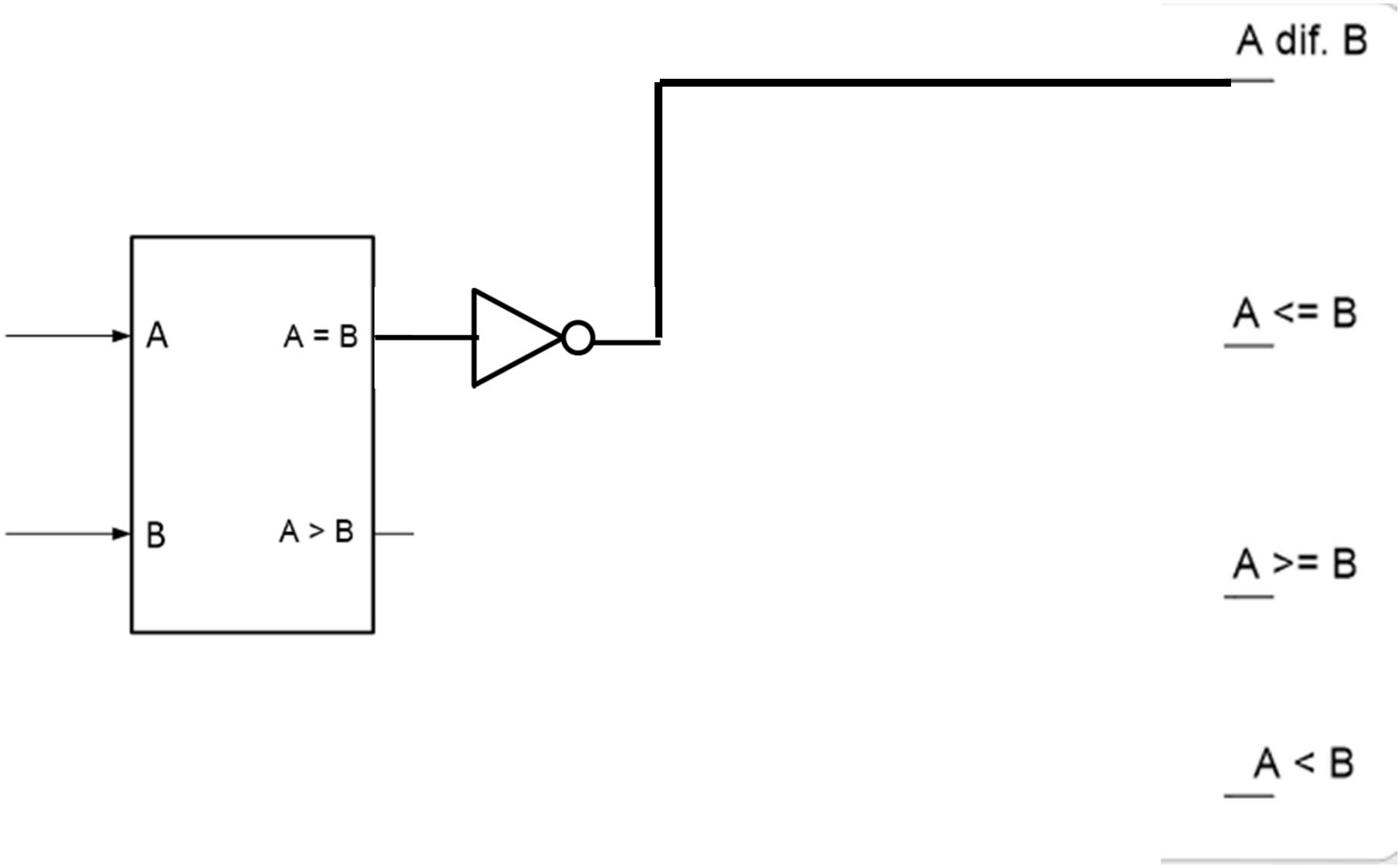
A <= B

A >= B

A < B

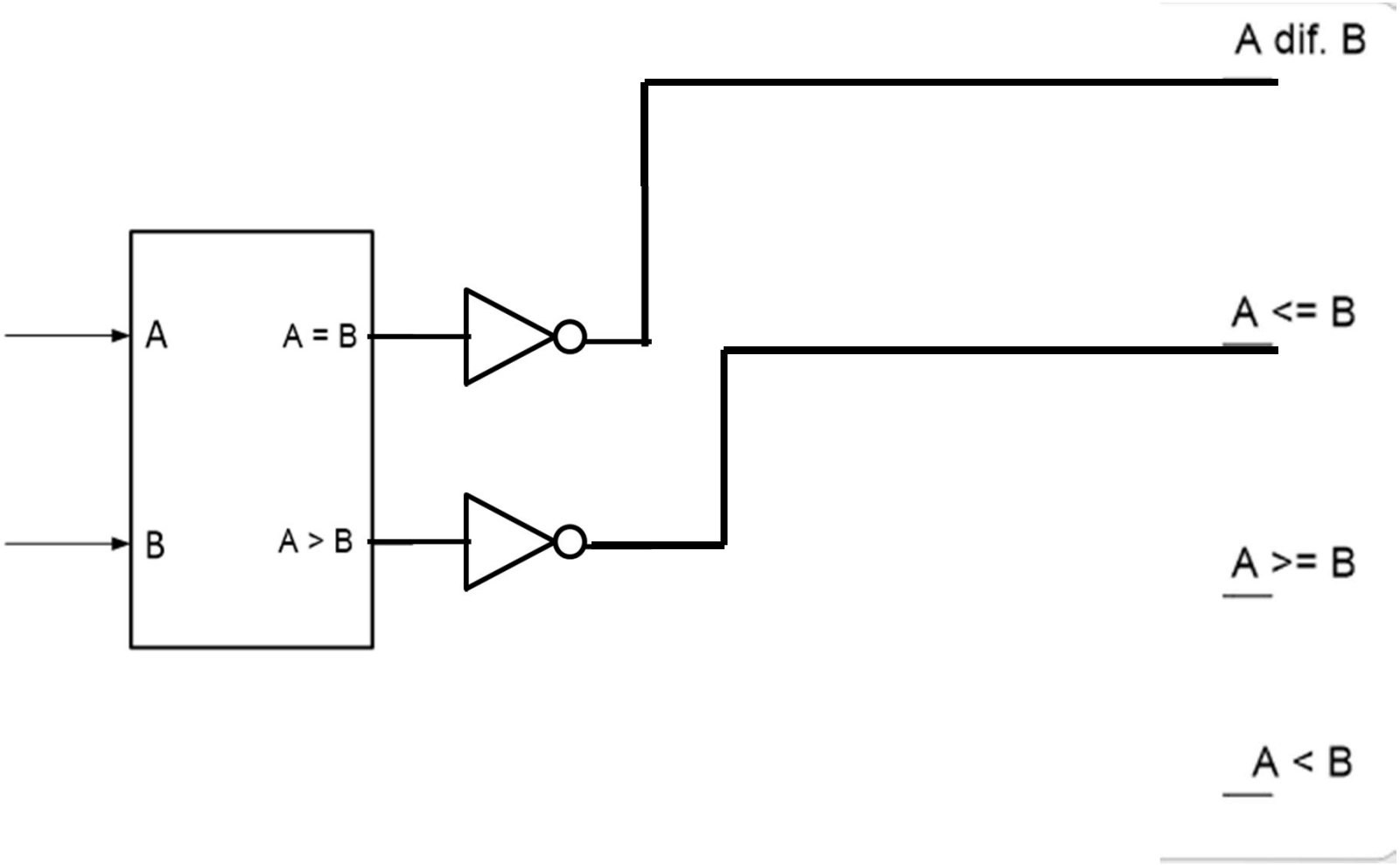
Comparador de magnitude

- A mais direta é “A dif. B”, pois basta ser “Não Igual”!



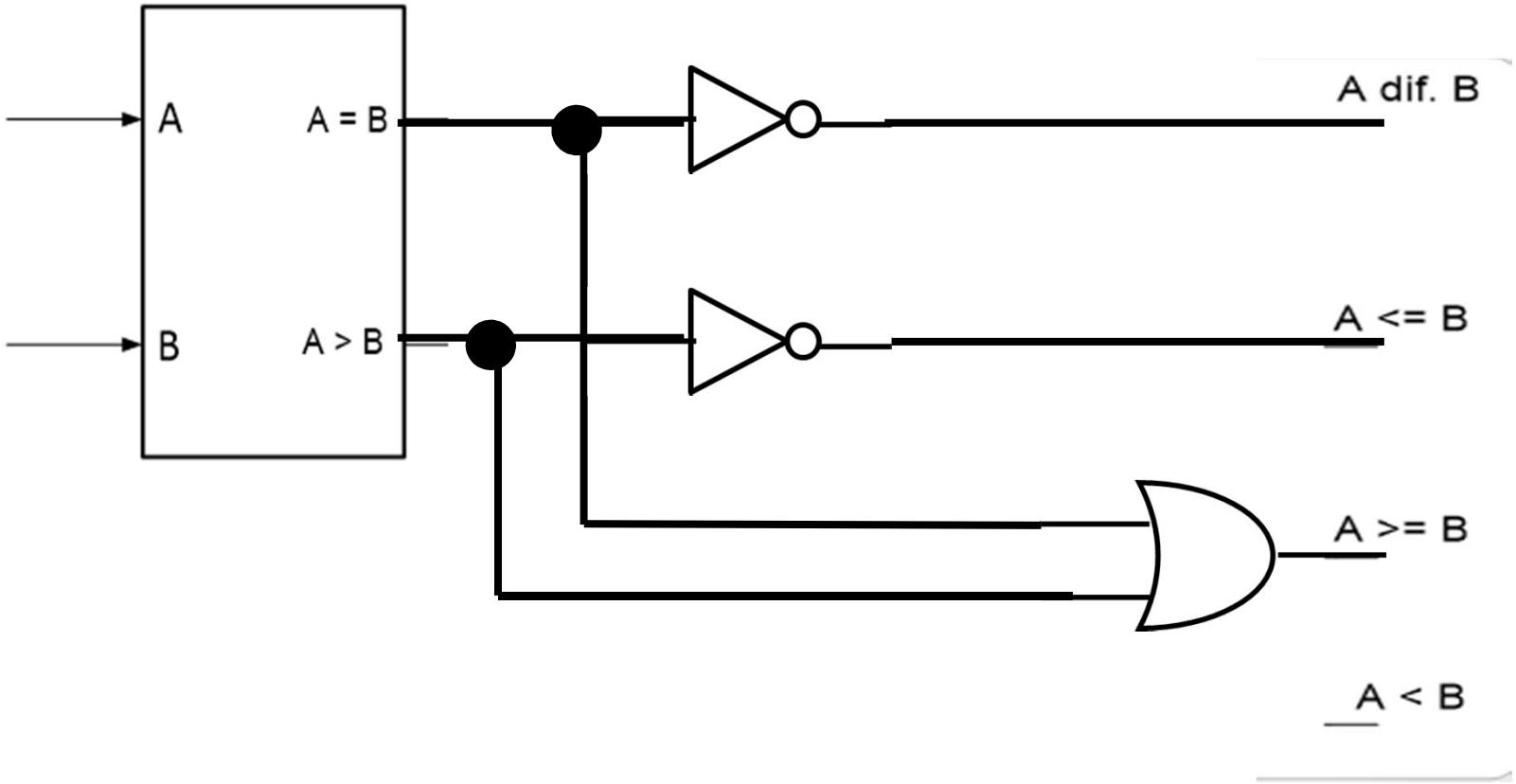
Comparador de magnitude

- A próxima é “A<=B”, pois basta ser “Não Maior”!



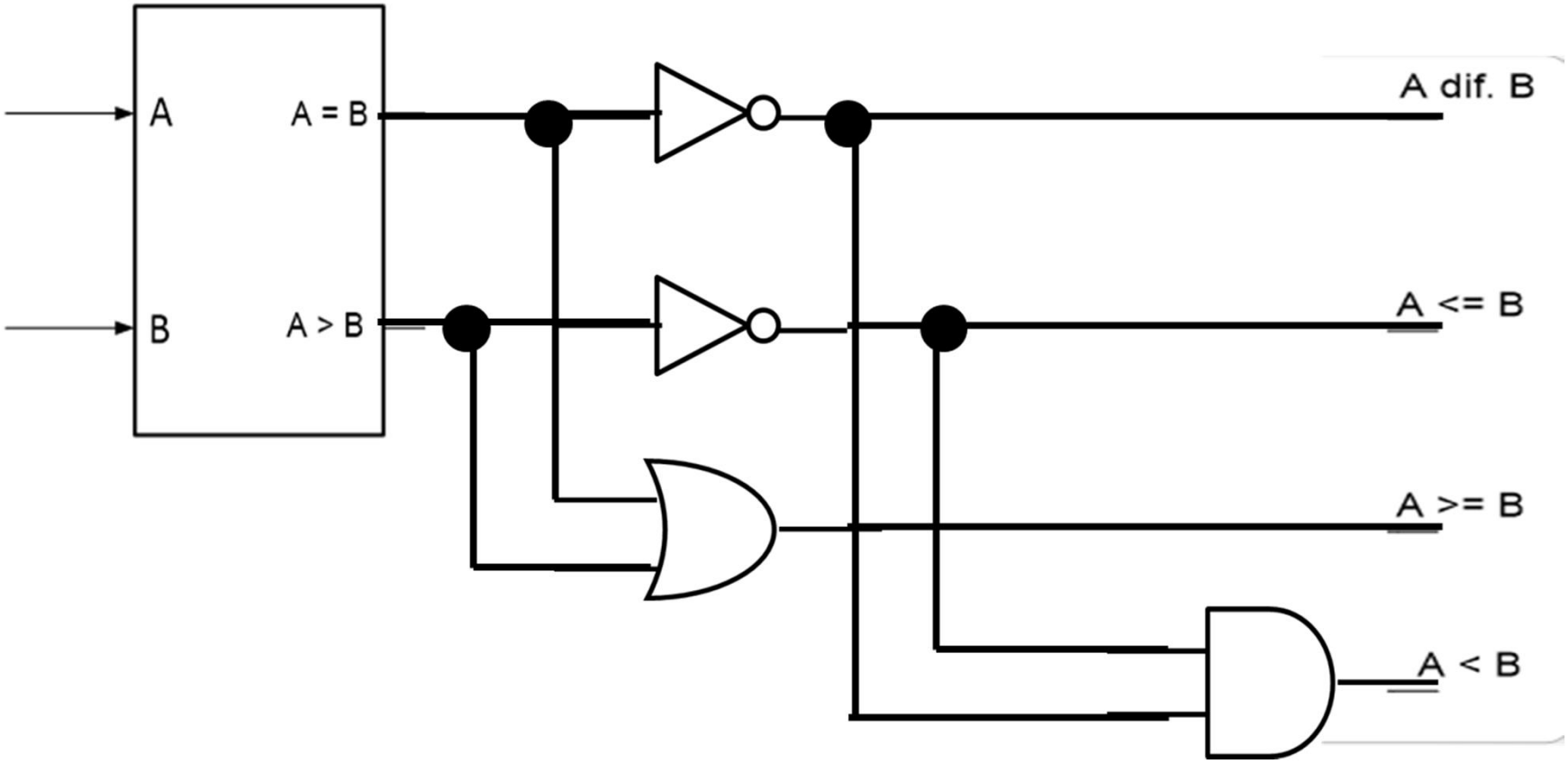
Comparador de magnitude

- Mas sabe-se que se “A=B” ou “A>B” então “A>=B”!



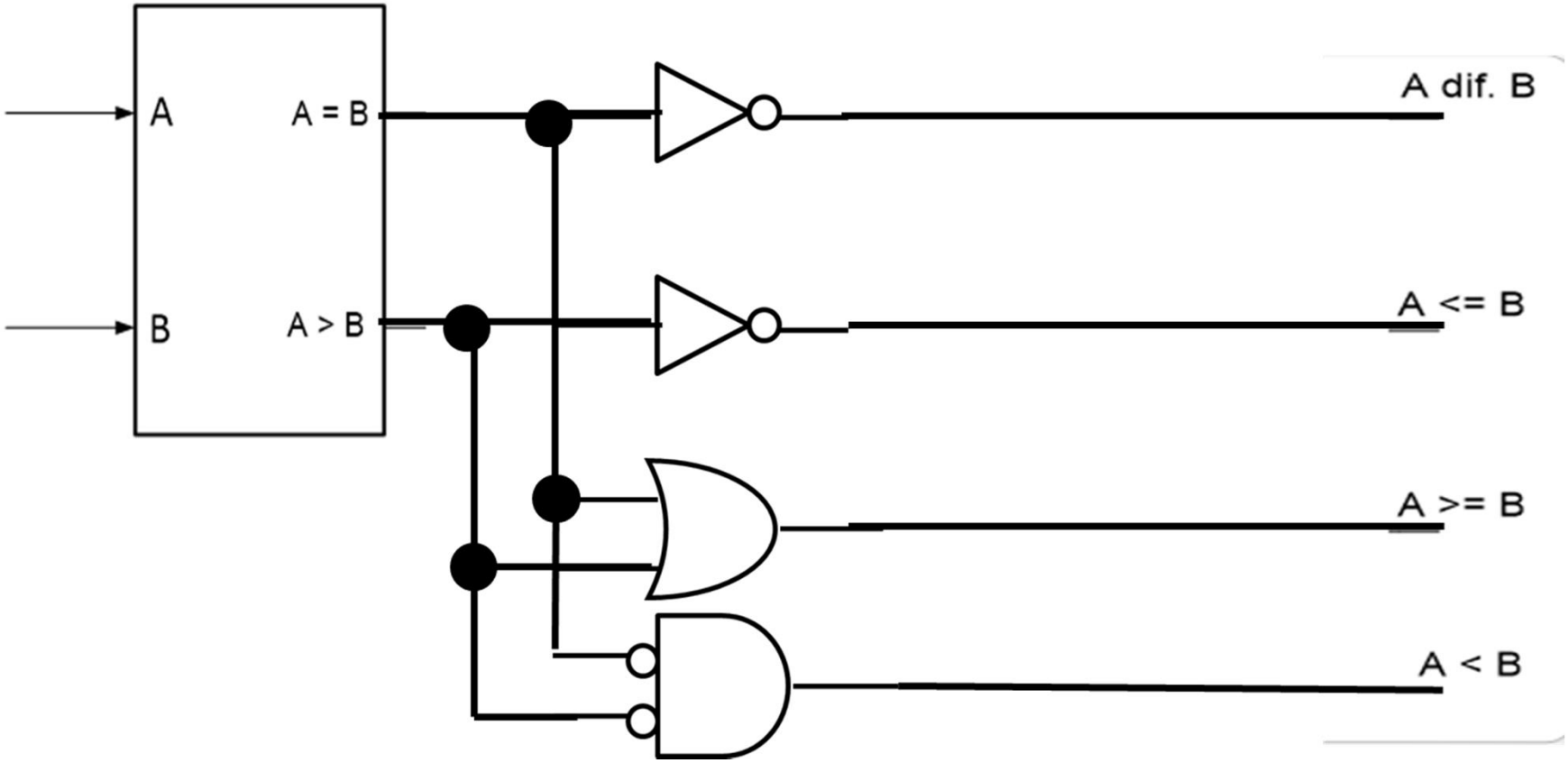
Comparador de magnitude

- Sabe-se que se A é diferente de B, e também ao mesmo tempo, A é menor ou igual a B, então “A<B”!



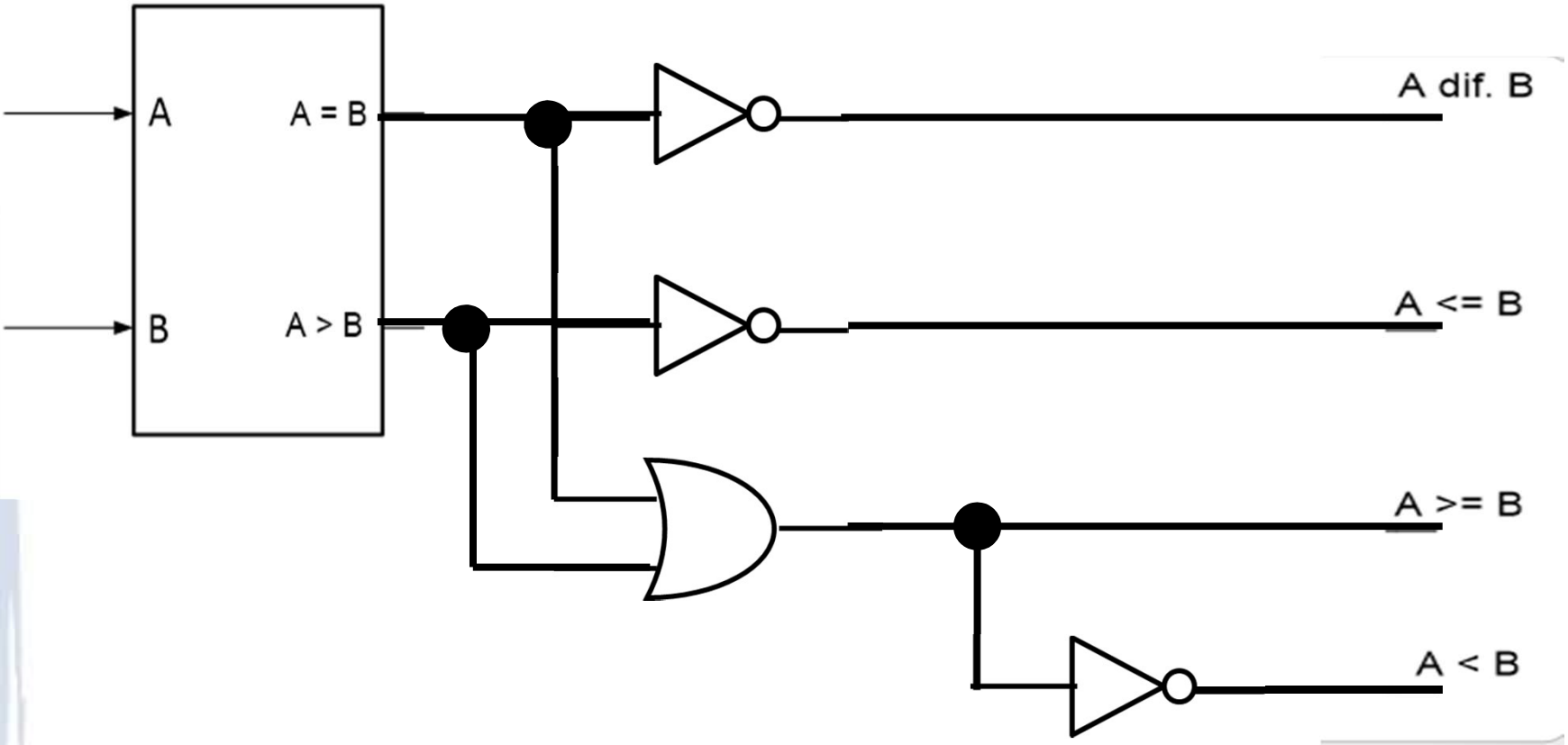
Comparador de magnitude

- Sabe-se que se A é diferente de B, e também ao mesmo tempo, A é menor ou igual a B, então “A<B”! Rearranjando o circuito:



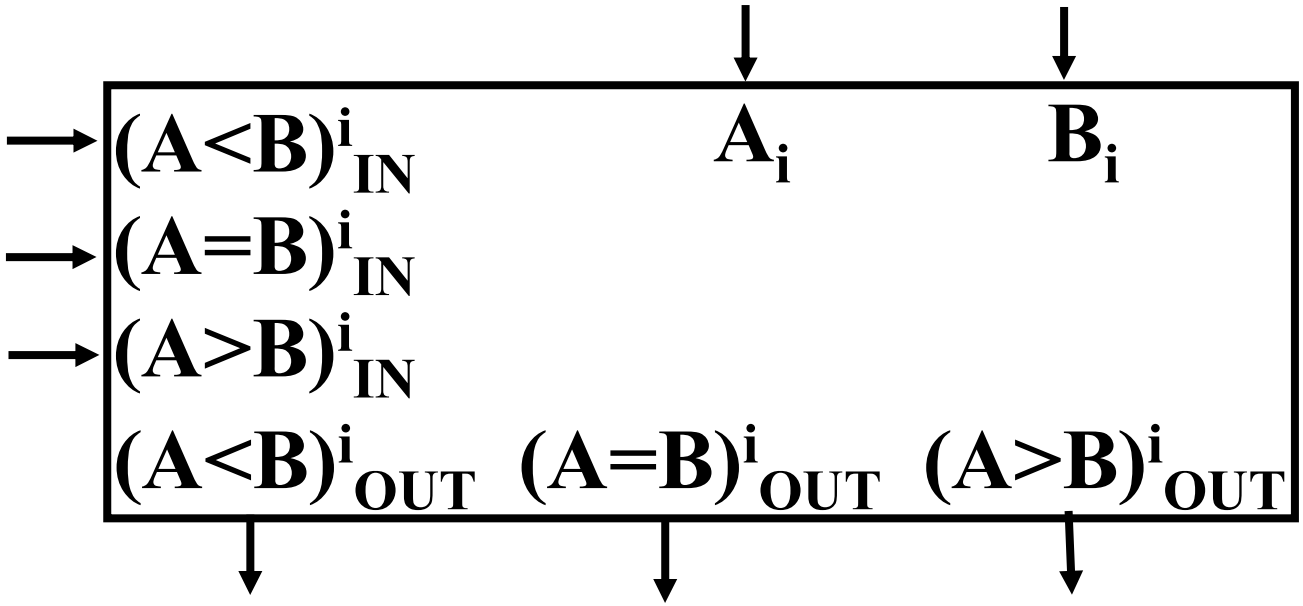
Comparador de magnitude

- Pelo Teorema de De Morgan tem-se que “not(X) and not(Y)” é igual a “not(X or Y)”, e já temos a porta “X or Y”.



Circuitos Iterativos

Comparadores, concebidos utilizando-se a técnica de Circuitos Iterativos



Comparadores “iterativos”

- Circuitos combinatórios “iterativos”:
  - Compostos de *n* módulos idênticos cascateados.
- Cada módulo possui “entradas e saídas”:
  - Primárias (in e out);
  - E para **associação em cascata** (X).
- Adequados para problemas que podem ser resolvidos com algoritmos iterativos.
- Mais lentos que circuitos paralelos equivalentes.

Comparadores "iterativos"

- A inspiração desta técnica em *hardware* pode ter vindo de seu uso em *software*.

- *Software* iterativo:

for (X_0 = valor inicial, i=0 ; i < n ; i++) {

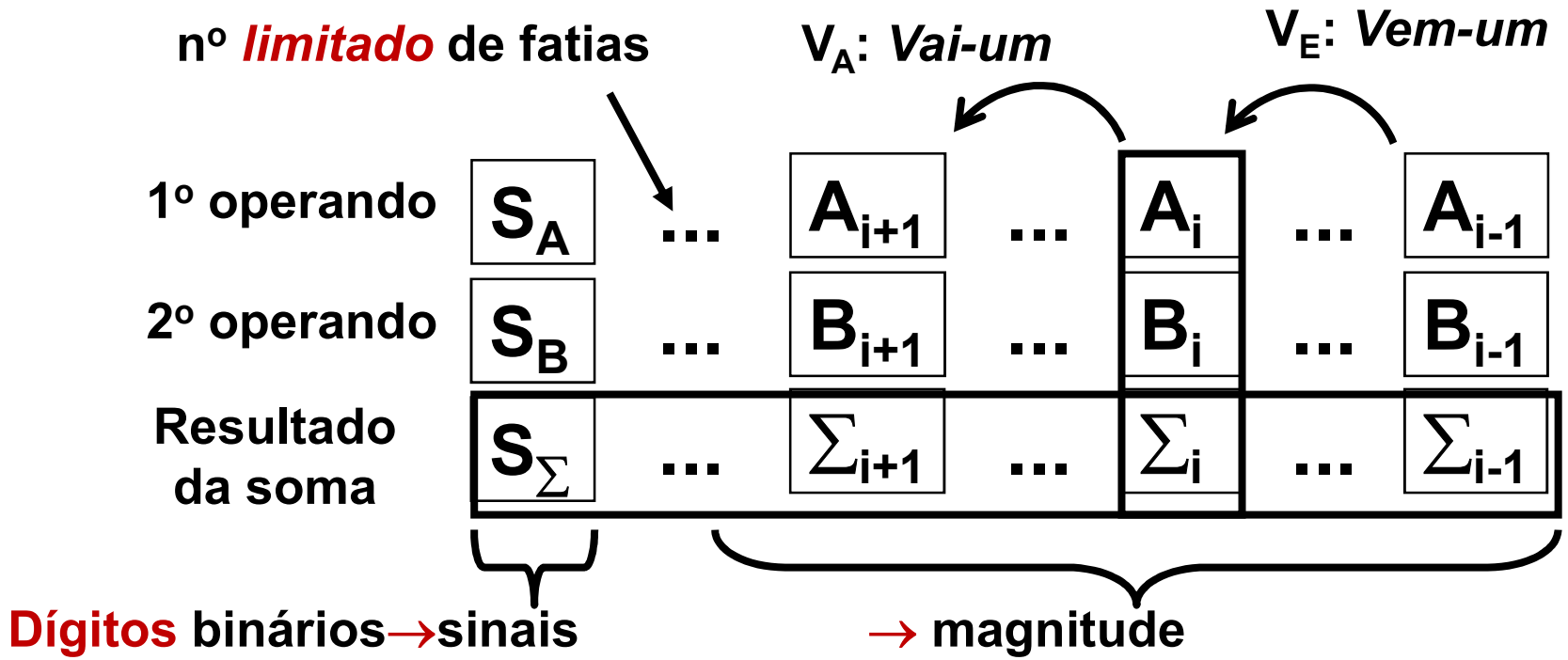
$out_i = f(X_i , in_i)$;

$X_{i+1} = g(X_i , in_i)$;

}

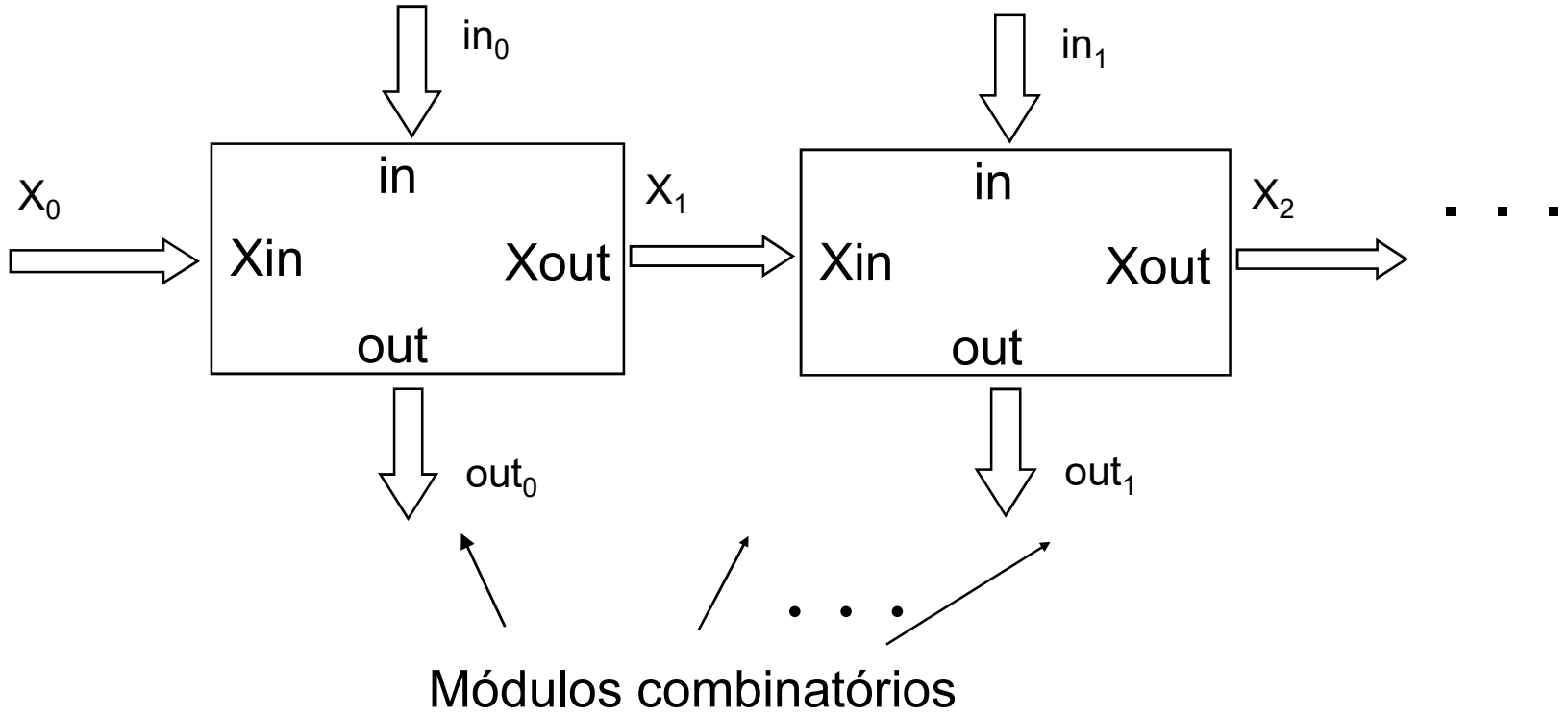
Lembrando: Adição de Números Binários

- **Lembrando: Operações com números binários:** O resultado da adição pode ser decidido (projetado) calculando-se os resultados parciais em ***fatias*** individuais, **iterativamente** ..... Depois realiza-se o cascateamento dos módulos ....



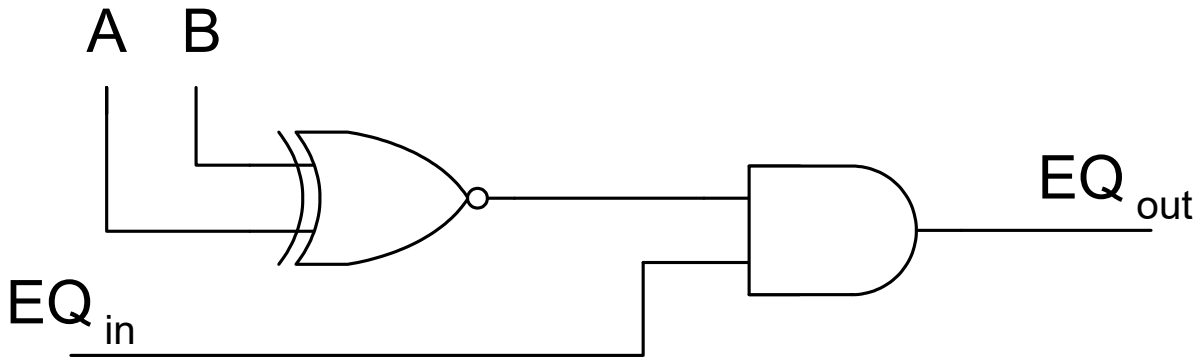
Comparadores “iterativos”

- Cada módulo possui:
  - Entradas e saídas primárias (in e out)
  - Entradas e saídas para **associação em cascata** (X)

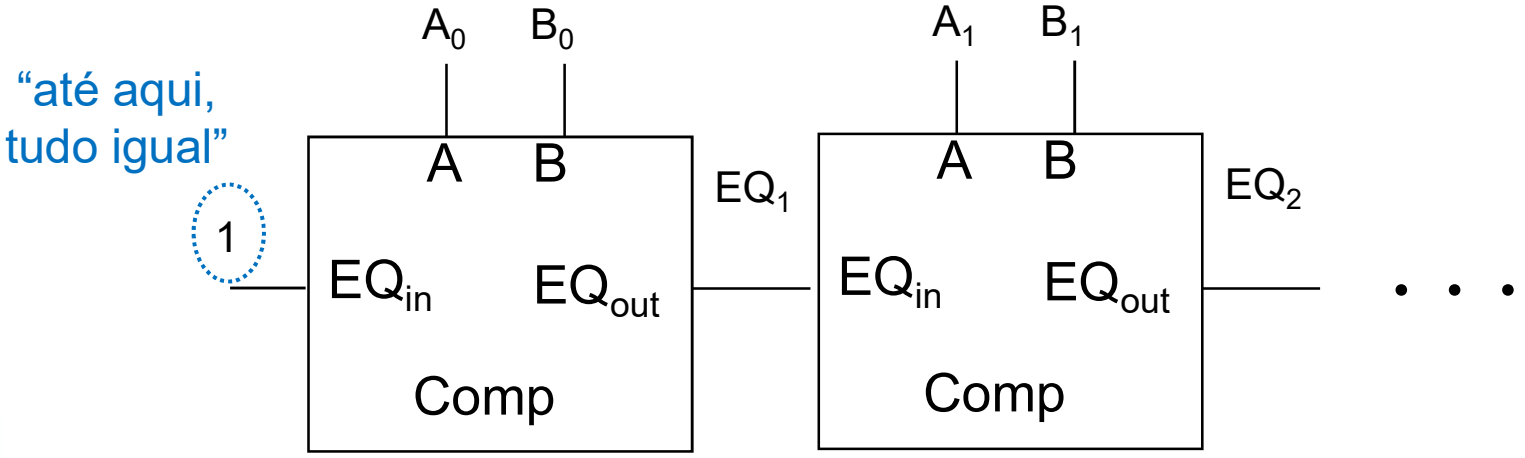


Comparadores “iterativos”

- Módulo combinatório básico do comparador

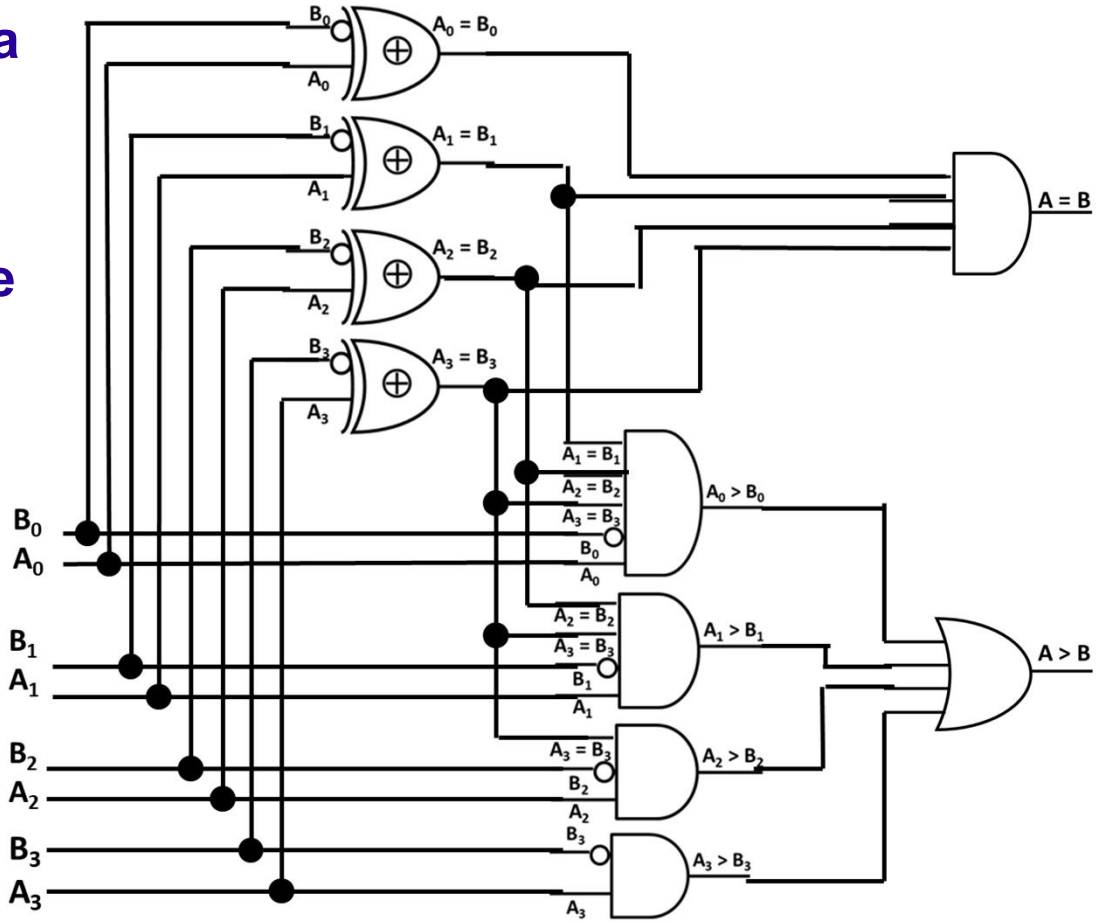


- Módulos do comparador associados em cascata



Comparador de Magnitude 4 bits: Circuito completo

Se formos usar a técnica de circuitos iterativos para reprojetar o circuito do comparador de magnitude de palavras de 4 bits que fizemos, o que deve ser alterado e/ou modificado no projeto original ???

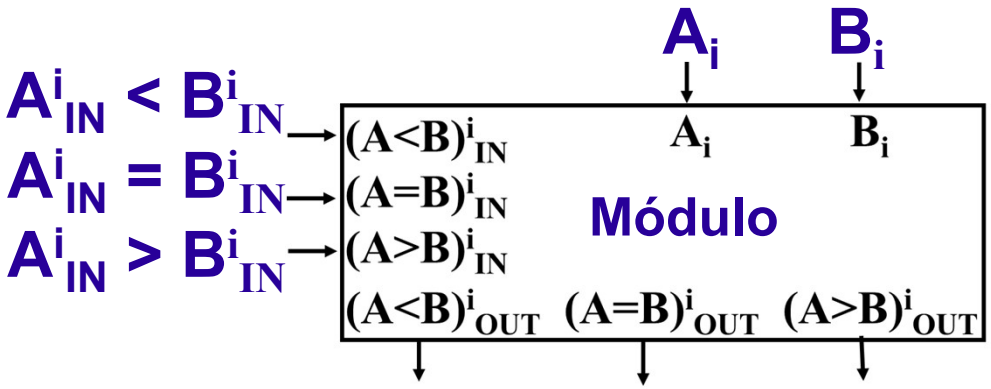


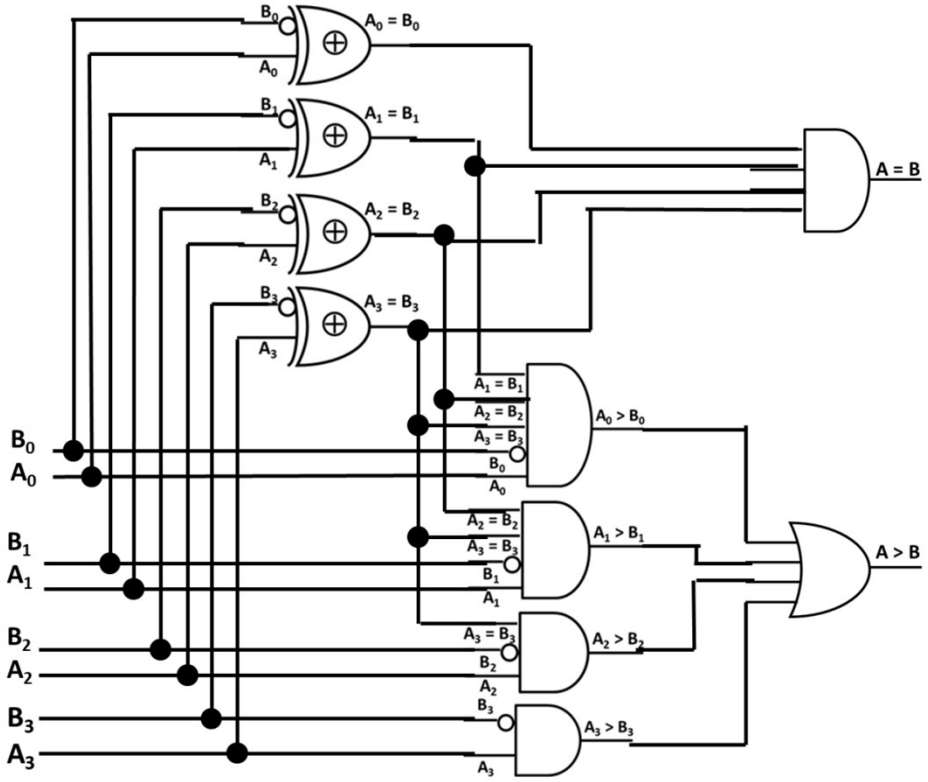
Resposta:

1-) Definição da unidade primitiva (Módulo);

2-) Redução da comparação de palavras de 4 bits (em paralelo) a comparação de apenas um par de bits (A_i;B_i);

3-) Introdução de entradas com o resultado da comparação das fatias menos significativas.



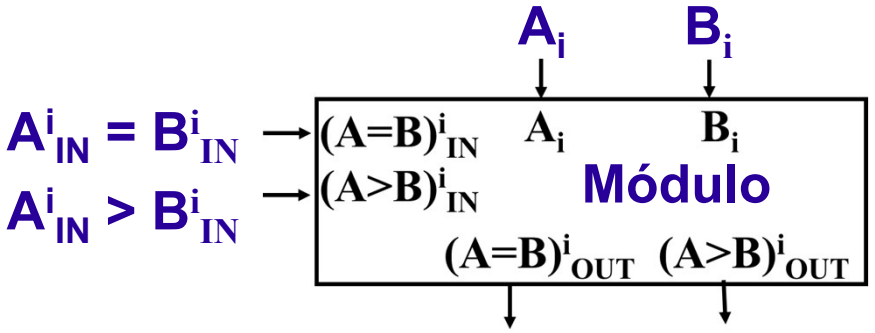


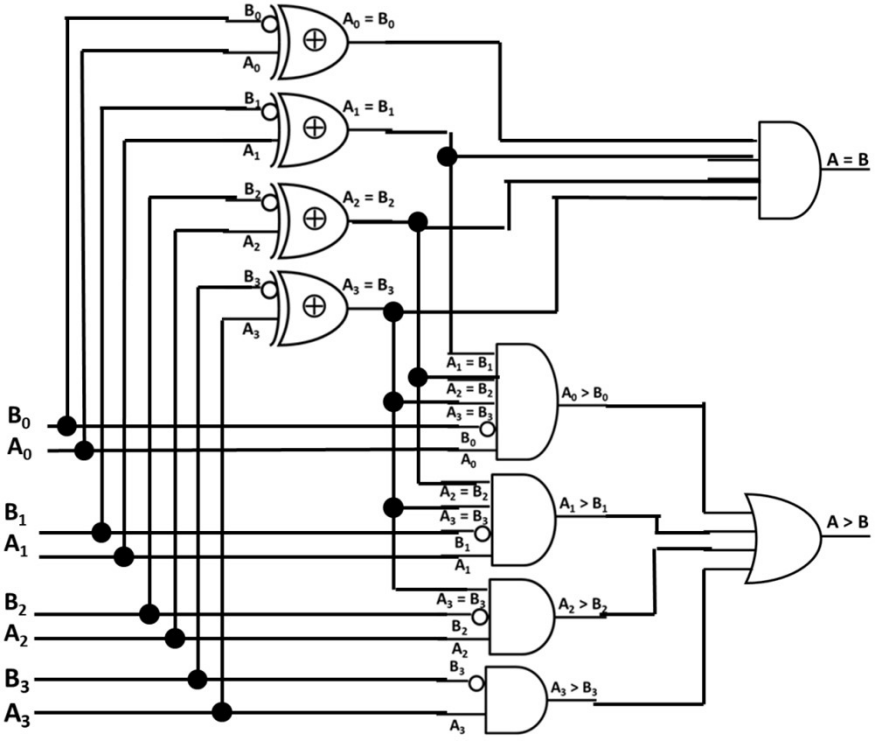
Exercício interessante:

1-) A partir do comparador paralelo (4bits) projetado;

2-) Realizar um "desmanche" em fatias de pares de bits (A_i;B_i), e em paralelo, fazer a comparação de um par (A_i;B_i) em cada fatia;

3-) Introduzir entradas com o resultado da comparação das fatias menos significativas.





Dicas - Exercício:

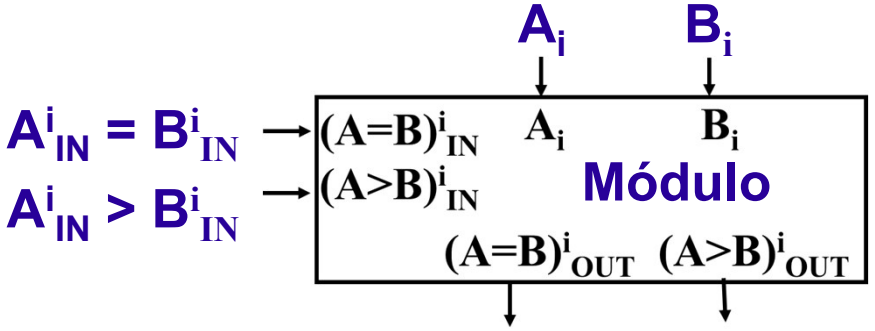
1-) "Desmanchar" $A_3>B_3$ junto com a porta OR (com 2 entradas apenas);

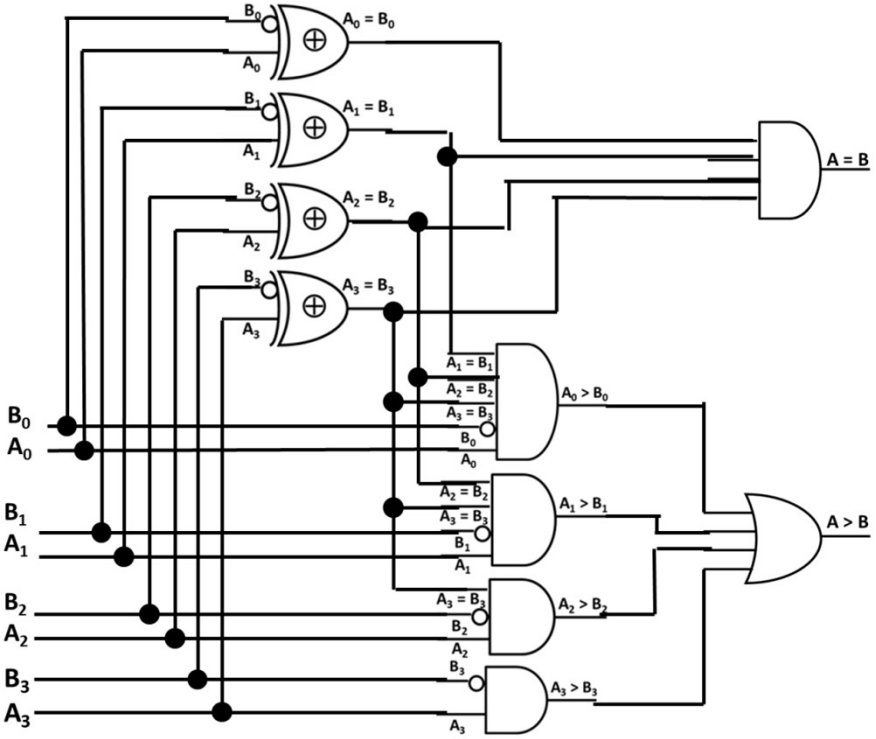
2-) "Desmanchar" $A_3=B_3$ junto com a porta AND (com 2 entradas apenas);

3-) Introduzir duas portas AND (2-*input*) para obter o resultado da comparação da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de $A_3=B_3$.

Comparador de Magnitude 4 bits: modelo com circuito iterativo





Comparador de Magnitude 4 bits: modelo com circuito iterativo

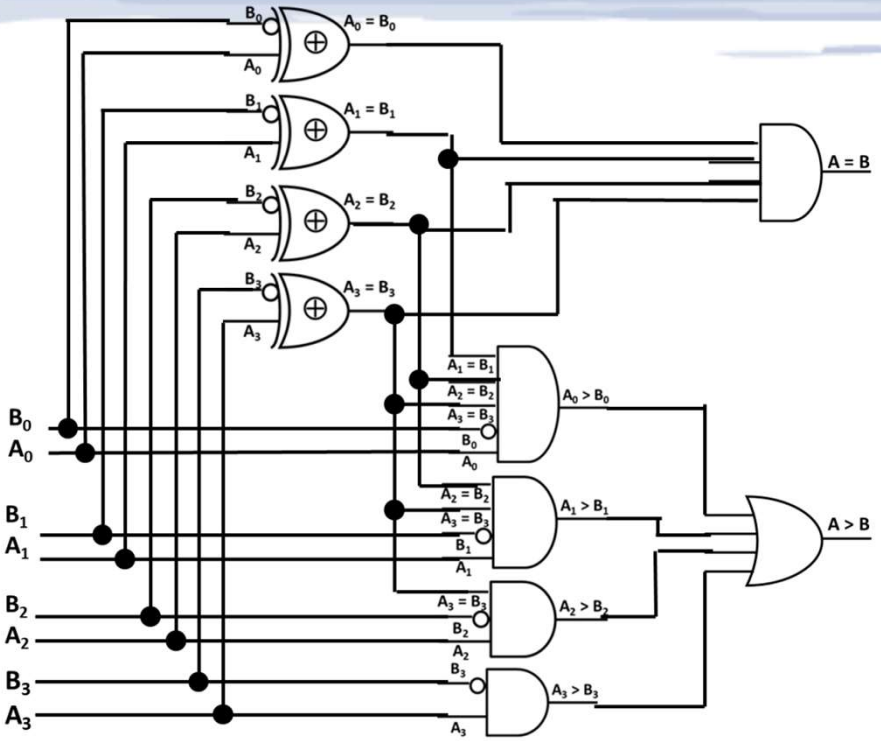
Dicas - Exercício:

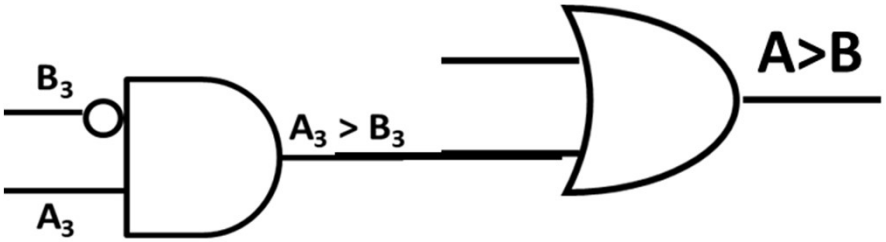
1-) "Desmanchar" $A_3>B_3$ junto com a porta OR (com 2 entradas apenas);

2-) "Desmanchar" $A_3=B_3$ junto com a porta AND (com 2 entradas apenas);

3-) Introduzir duas portas AND (2-*input*) para obter o resultado da com-paração da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de $A_3=B_3$.





Comparador de Magnitude 4 bits: modelo com circuito iterativo

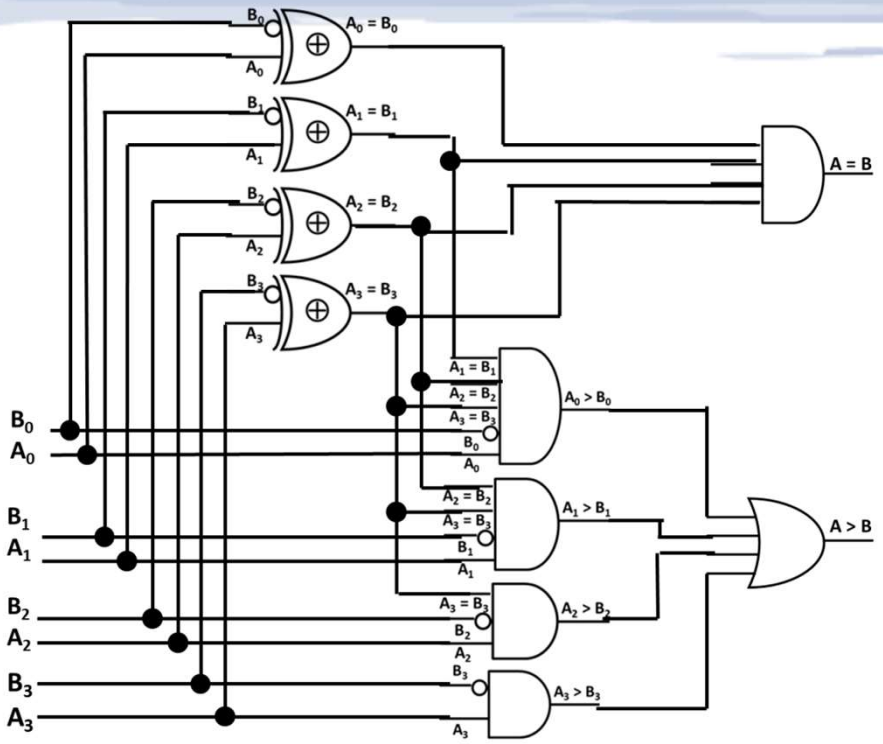
Dicas - Exercício:

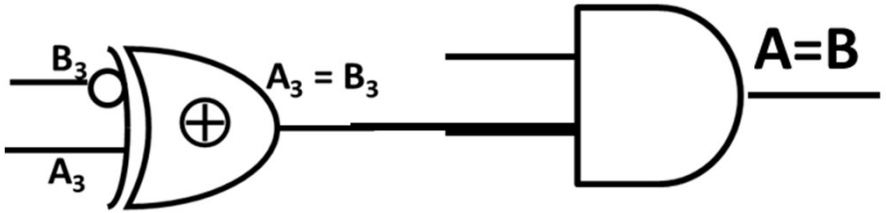
1-) “Desmanchar” A3>B3 junto com a porta OR (com 2 entradas apenas);

2-) “Desmanchar” $A_3=B_3$ junto com a porta AND (com 2 entradas apenas);

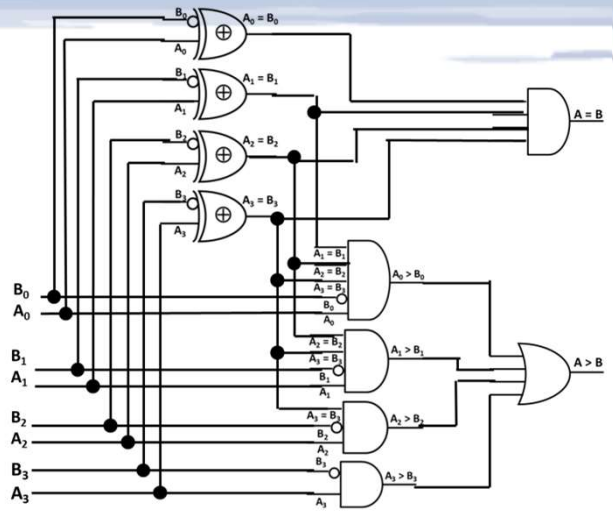
3-) Introduzir duas portas AND (2-*input*) para obter o resultado da com-paração da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de $A_3=B_3$.





Comparador de Magnitude 4 bits: modelo com circuito iterativo



Dicas - Exercício:

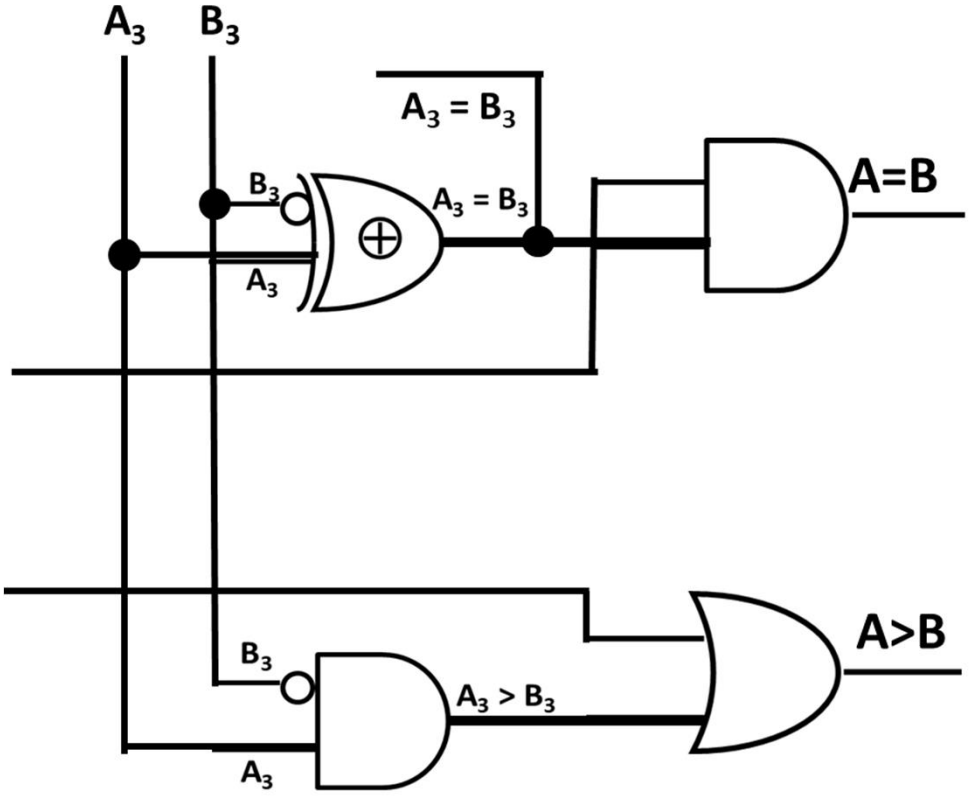
1-) "Desmanchar" A3>B3 junto com a porta OR (com 2 entradas apenas);

Reorganizar 1-) e 2-)

2-) "Desmanchar" A_3=B_3 junto com a porta AND (com 2 entradas apenas);

3-) Introduzir duas portas AND (2-*input*) para obter o resultado da comparação da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de A_3=B_3 .



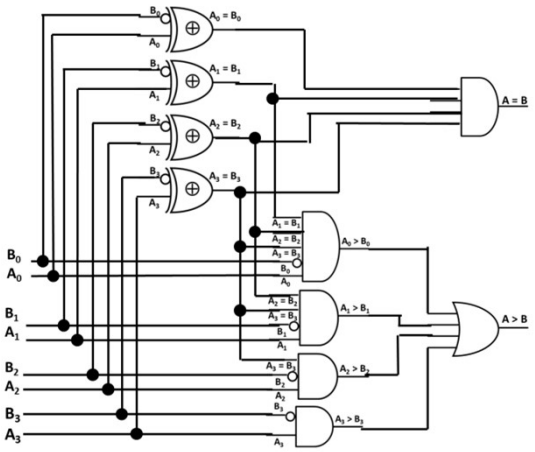
Dicas - Exercício:

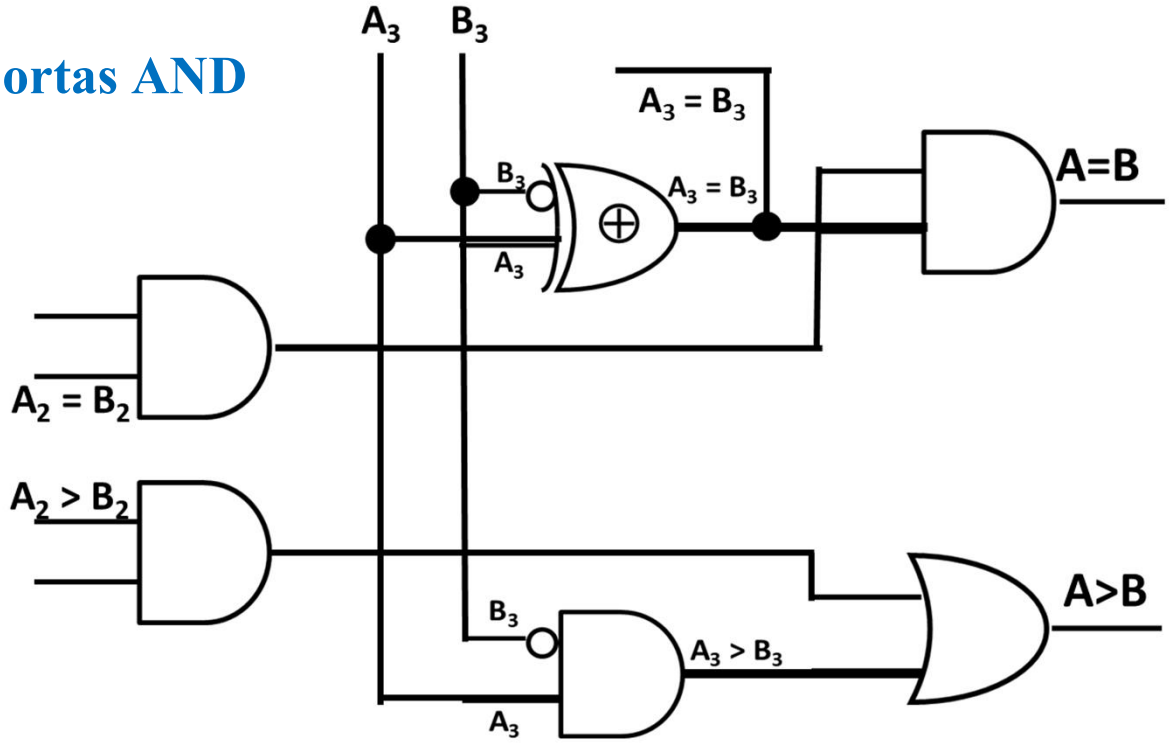
1-) "Desmanchar" A3>B3 junto com a porta OR (com 2 entradas apenas);
2-) "Desmanchar" A3=B3 junto com a porta AND (com 2 entradas apenas);

3-) Introduzir duas portas AND (2- *input*) para obter o resultado da comparação da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de A_3=B_3 .

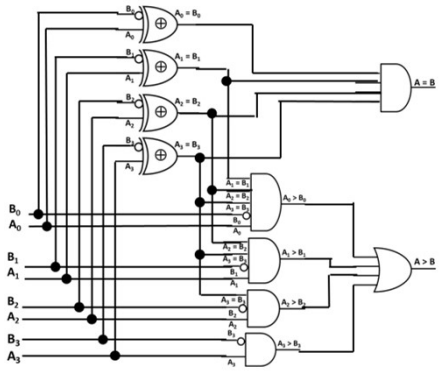
Comparador de Magnitude 4 bits: modelo com circuito iterativo





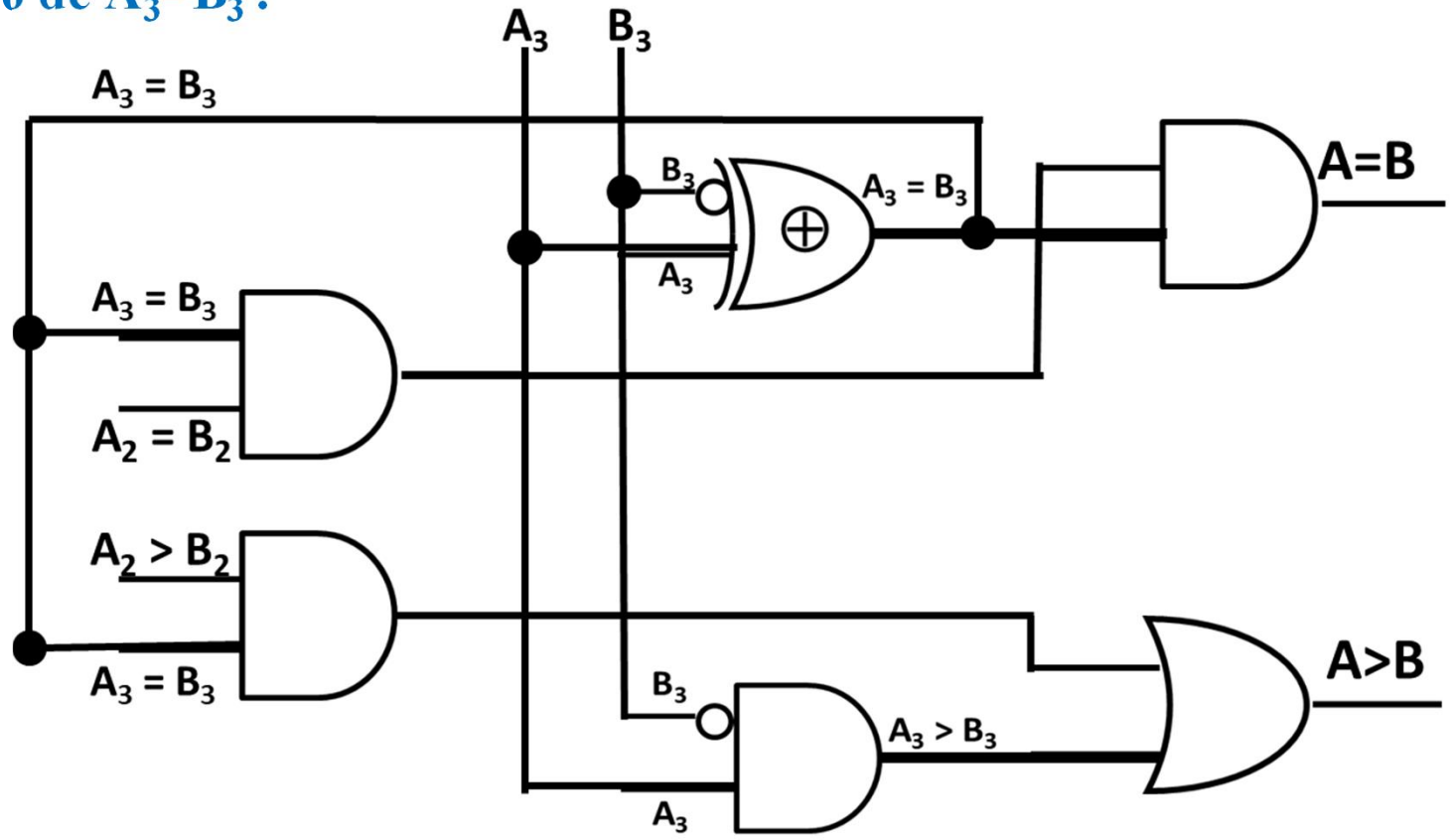
Dicas - Exercício:

Comparador de Magnitude 4 bits: modelo com circuito iterativo



1-) “Desmanchar” A3>B3 junto com a porta OR (com 2 entradas apenas);
2-) “Desmanchar” A3=B3 junto com a porta AND (com 2 entradas apenas);
3-) Introduzir duas portas AND (2- input) para obter o resultado da comparação da fatia menos significativa imediatamente anterior.

4-) Habilitar a entrada das portas AND com o resultado de $A_3=B_3$.

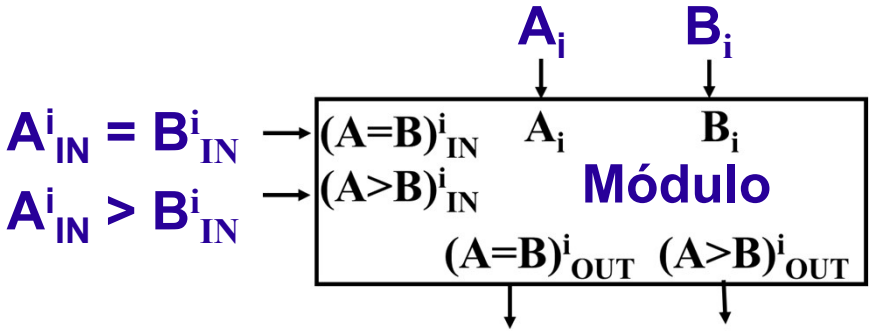


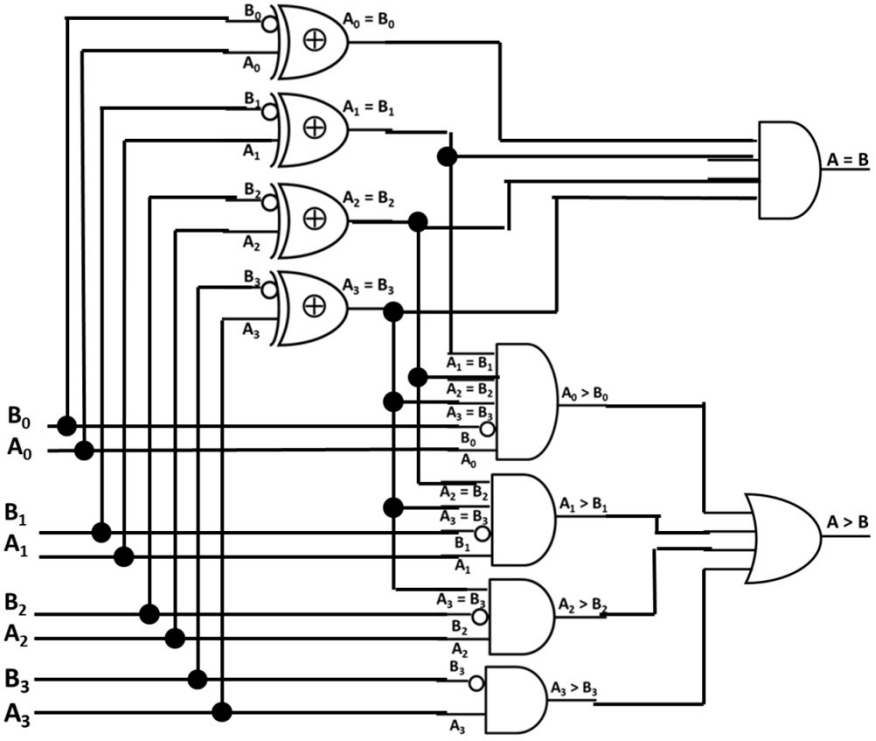
Exercício - desdobramentos:

1-) Projetar os demais módulos;

2-) Realizar o cascateamento entre os módulos;

3-) Construir comparadores de maior tamanho de palavra.



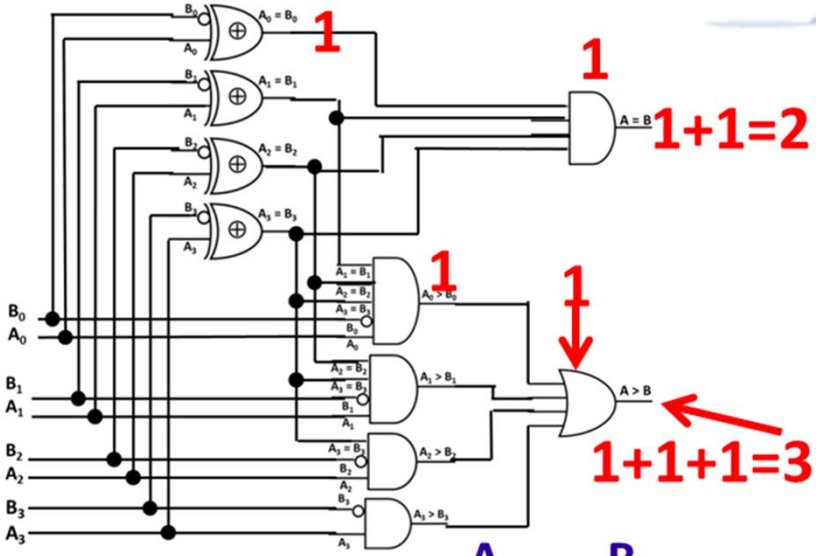


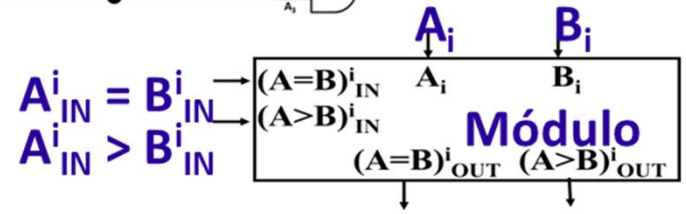
Exercício - desdobramentos:

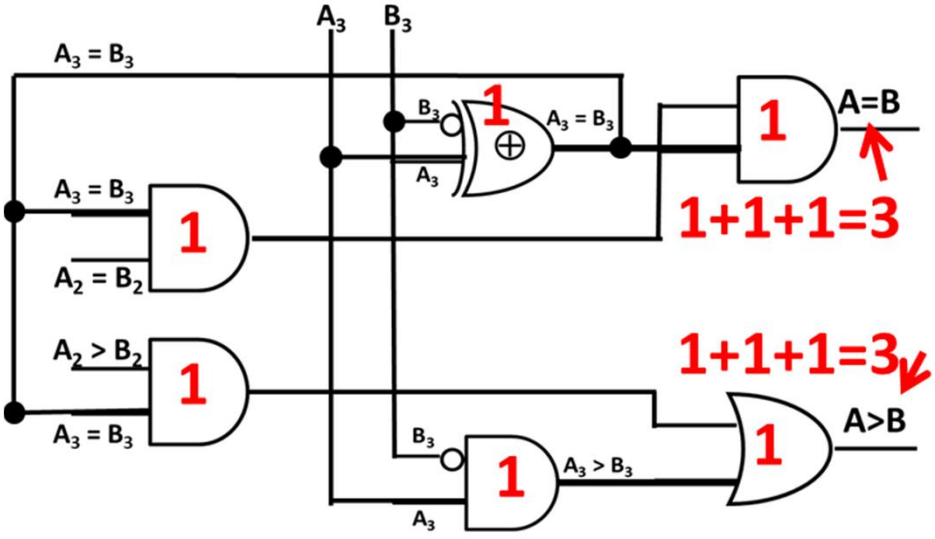
4-) A questão dos tempos de atraso na propagação de sinais;

4.a-) Vamos normalizar os atrasos de portas (T=1);
4.b-) Calcular o pior atraso para as duas soluções;
4.c-) Circuito iterativo - O atraso é de 1 só módulo;

5-) Qual seria o atraso (pior caso) na obtenção da resposta para um comparador de duas palavras de 8 bits implementado com módulos iterativos?





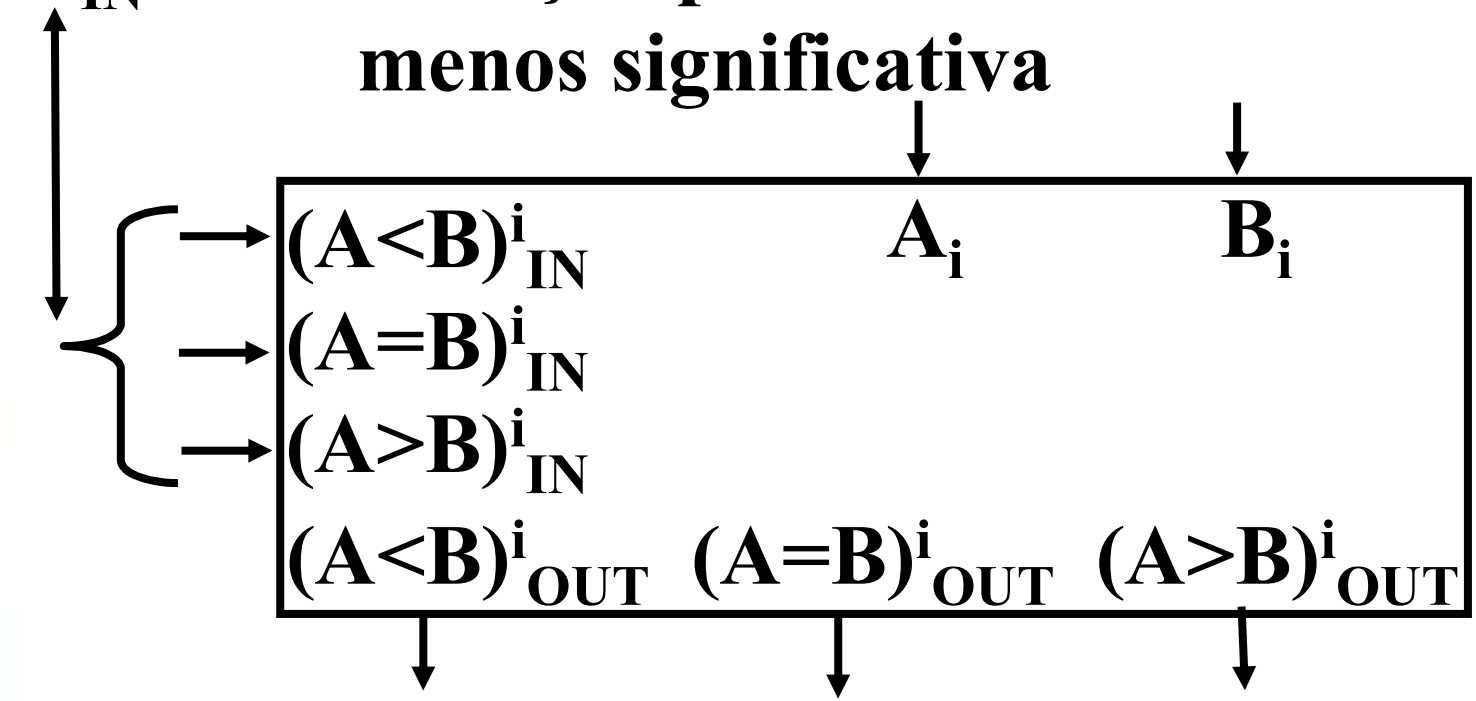


Comparadores “iterativos”

- Circuitos combinatórios “iterativos”: Também se pode utilizar os métodos tradicionais de síntese de circuitos combinatórios para obtê-los.
  - Descrição do comportamento funcional por meio de Tabela Verdade.
  - Síntese por meio de Mapas de *Karnaugh*.
  - Vamos nos aproveitar da oportunidade e incluir a comparação A<B, que não havia anteriormente;
  - Os slides seguintes mostram o resultado de síntese com esta técnica.
  - No Apêndice mais detalhes, passo a passo, são apresentados.

Projeto de um Comparador de Magnitude

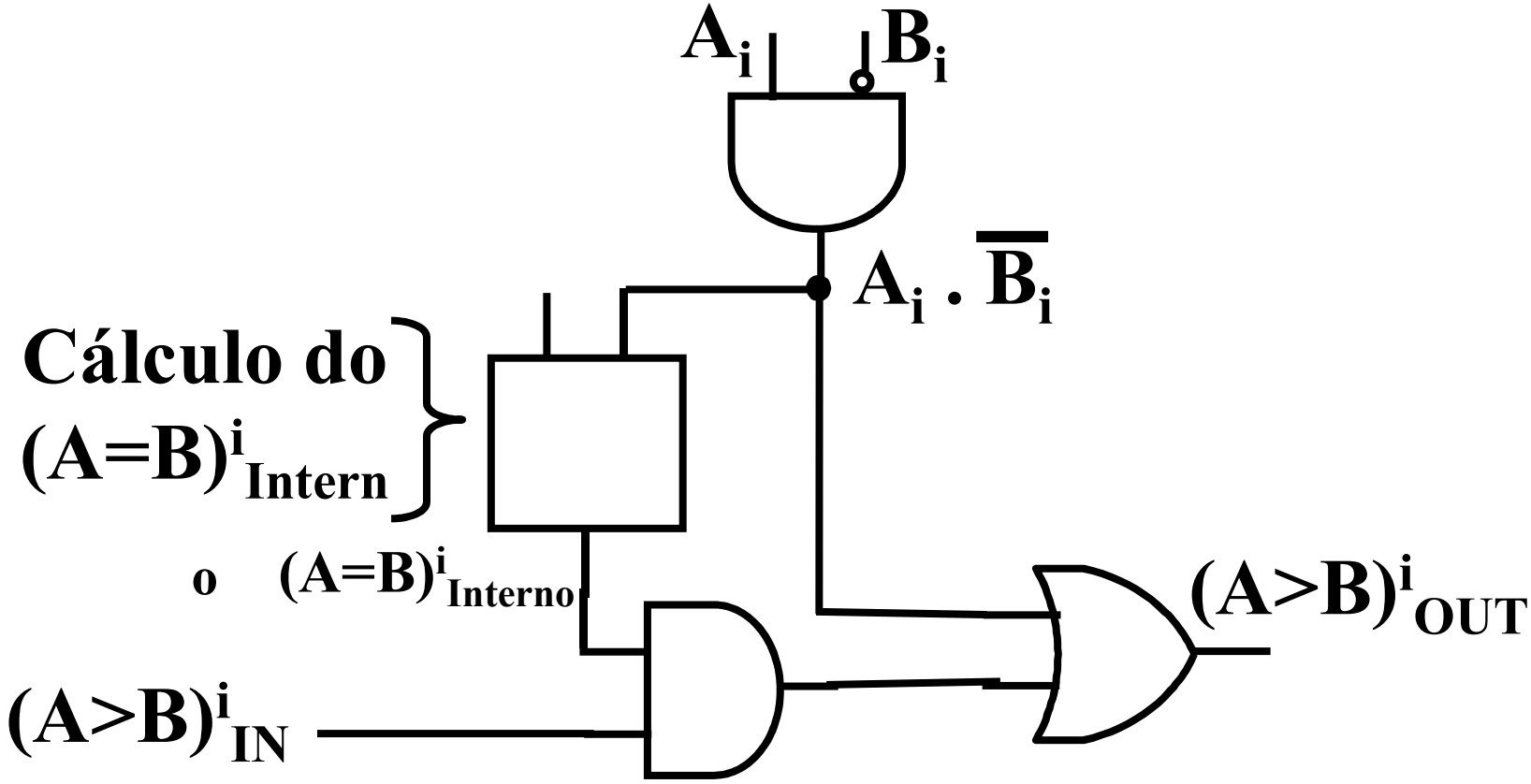
C^i_{IN} – Informação proveniente da fatia menos significativa



$(A<B)^i$	$(A=B)^i$	$(A>B)^i$
0	0	1
0	1	0
1	0	0

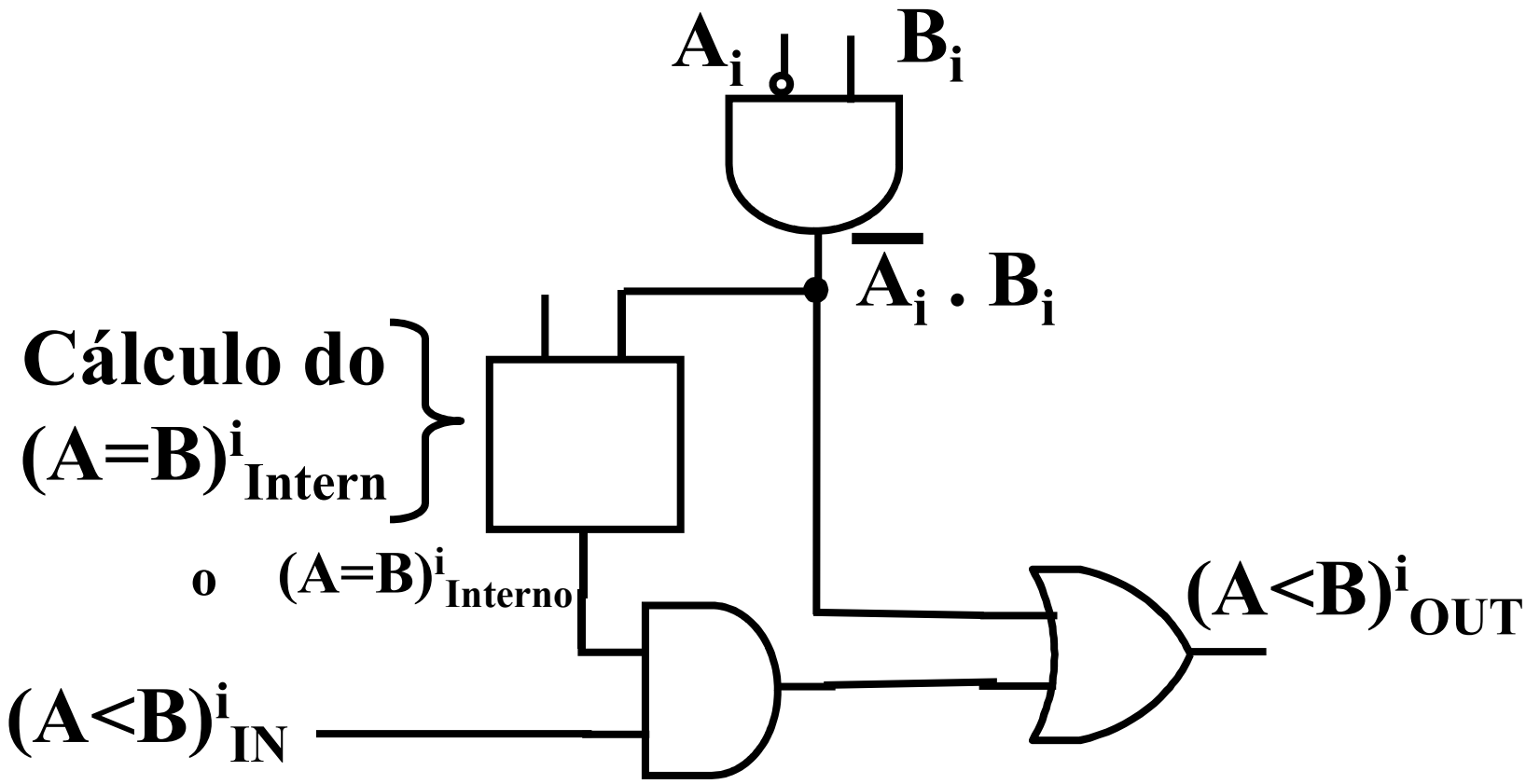
Projeto de um Comparador de Magnitude

$$(A>B)^i_{OUT} = A_i \cdot \overline{B_i} + (A>B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}]}_{(A=B)^i_{Interno}}$$

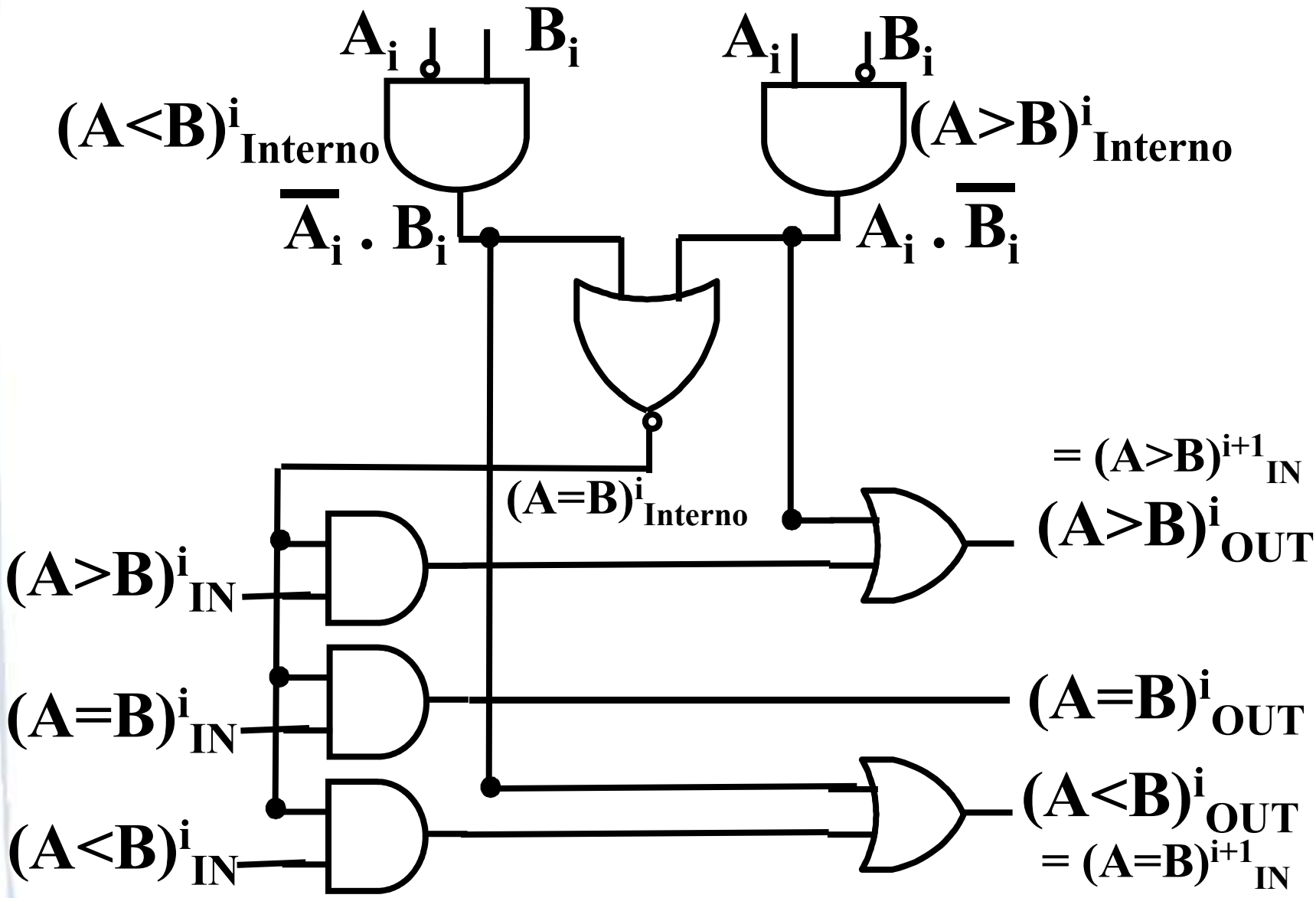


Projeto de um Comparador de Magnitude

$$(A<B)^i_{OUT} = \overline{A}_i \cdot B_i + (A<B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A}_i \cdot \overline{B}_i]}_{(A=B)^i_{Interno}}$$



Projeto de um Comparador de Magnitude



VHDL

Como se pode descrever um comparador em VHDL?

Qual seria uma proposta de realização de um *testbench* para um comparador?

Comparador de magnitude: VHDL

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

entity compmag is
   port (inpA,inpB : in std_logic_vector(3 downto 0);
         greater, equal, smaller  : out std_logic);
end compmag;

architecture compmag_arch of compmag is

begin
   -- std_logic e std_logic_vector suportam comparações
greater <= '1' when (inpA > inpB) else '0';
equal  <= '1' when (inpA = inpB) else '0';
   smaller <= '1' when (inpA < inpB) else '0';

end compmag_arch;
```

Desafio para a disciplina

- Projetar um **cadeado digital**
  - Assuma uma senha fixa, de fábrica: **149**

Combinatório

Sequencial

Cadeado Digital Combinatório

```
-- Simple Lock design
library IEEE;
use IEEE.std_logic_1164.all;

entity locker is
port(
  s1: in std_logic_vector(3 downto 0);
  s2: in std_logic_vector(3 downto 0);
  s3: in std_logic_vector(3 downto 0);
  q: out std_logic);
end locker;

--Senha: 149
architecture locker_arch of locker is
signal locks : std_logic_vector(1 to 3);
begin
    locks(1) <= '1' when (s1 = "0001") else '0';
    locks(2) <= '1' when (s2 = "0100") else '0';
    locks(3) <= '1' when (s3 = "1001") else '0';
    q <= locks(1) and locks(2) and locks(3);
end locker_arch;
```

Cadeado Digital - Testbench

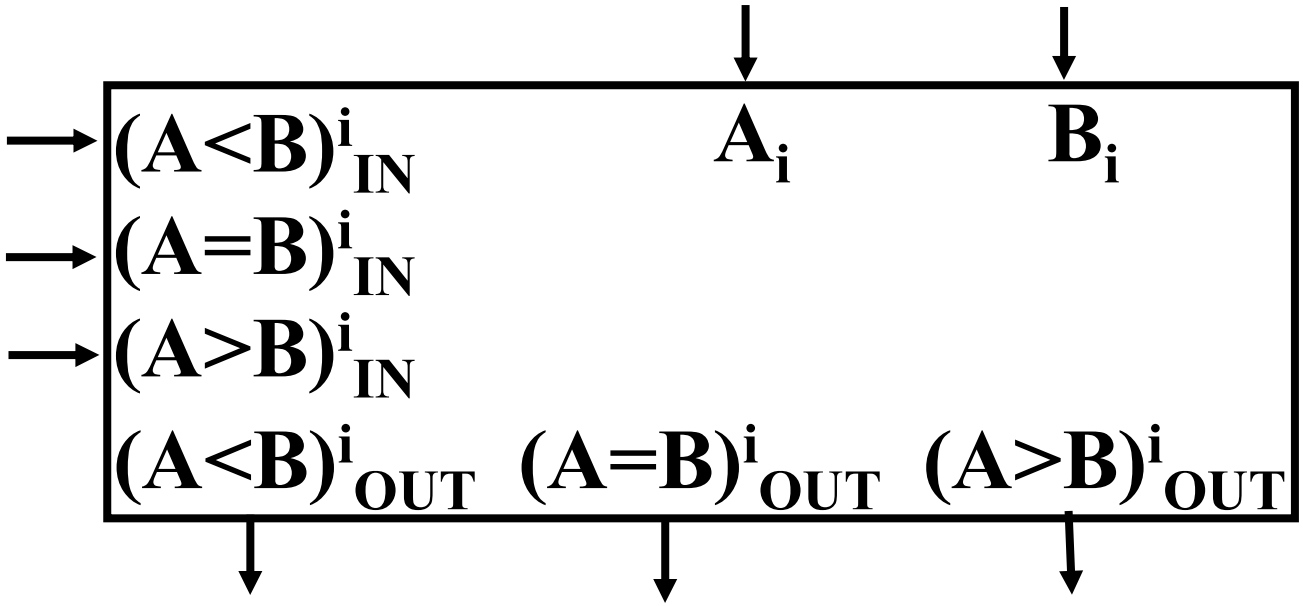
```
-- Testbench for cadeado combinatorio
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
-- empty
end testbench;

architecture tb of testbench is
   -- DUT (Device Under Test) component
   component locker is
       port(  s1: in std_logic_vector(3 downto 0);
              s2: in std_logic_vector(3 downto 0);
              s3: in std_logic_vector(3 downto 0);
              q: out std_logic);
   end component;

   -- Sinais de teste: entradas e saídas
   signal s1_in, s2_in, s3_in: std_logic_vector(3 downto 0);
   signal q_out: std_logic;

begin
```

Cadeado Digital – Testbench (cont.)

```
 -- Conecta DUT a entradas e saídas
DUT: locker port map(s1_in, s2_in, s3_in, q_out);

process
begin -- Conjunto de testes
   s1_in <= "0000"; s2_in <= "0000"; s3_in <= "0000";
   wait for 1 ns;
   assert(q_out='0') report "Fail 1/4" severity error;

   s1_in <= "0001"; s2_in <= "0000"; s3_in <= "0000";
   wait for 1 ns;
   assert(q_out='0') report "Fail 2/4" severity error;

   s1_in <= "0001"; s2_in <= "0100"; s3_in <= "0000";
   wait for 1 ns;
   assert(q_out='0') report "Fail 3/4" severity error;

   s1_in <= "0001"; s2_in <= "0100"; s3_in <= "1001";
   wait for 1 ns;
   assert(q_out='1') report "Fail 4/4" severity error;

   -- Clear inputs
   s1_in <= "0000"; s2_in <= "0000"; s3_in <= "0000";

   assert false report "Test done." severity note;
    wait;
 end process;
end tb;
```

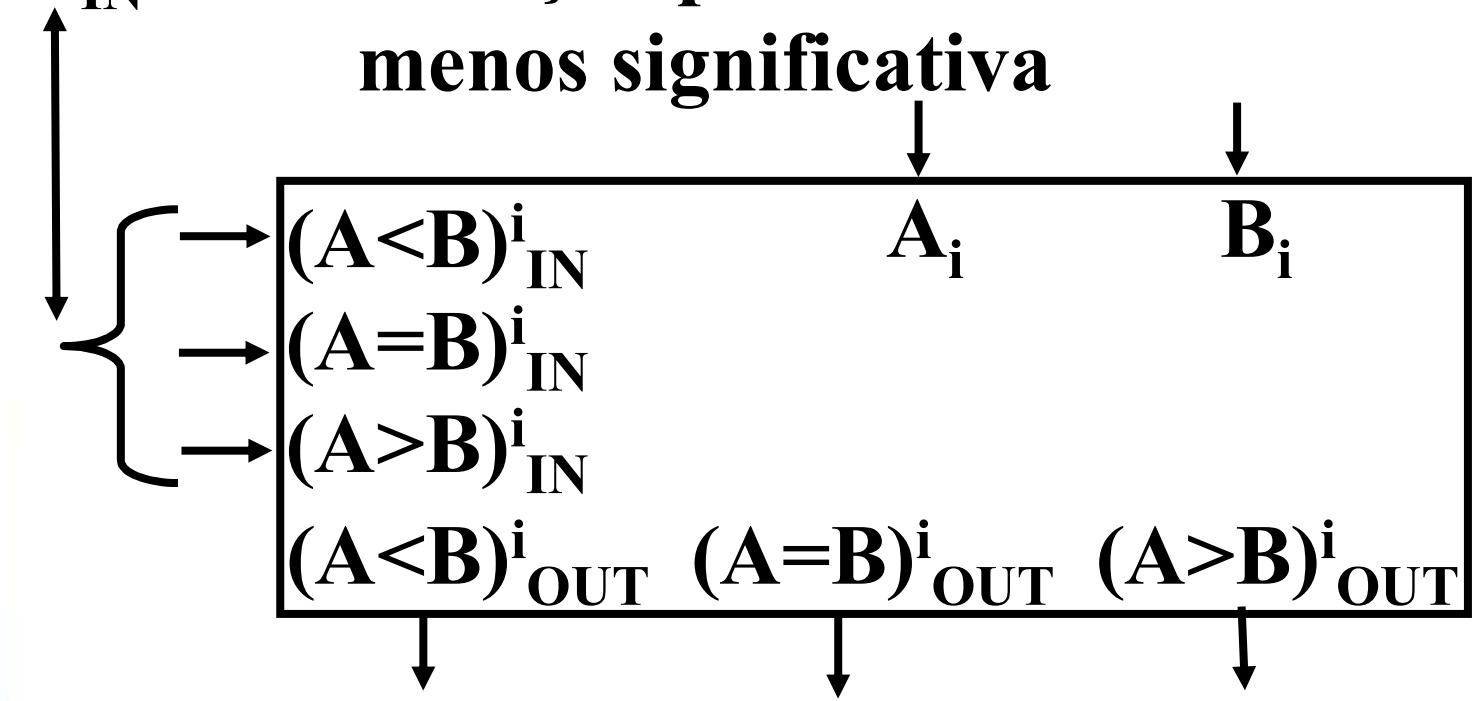
APÊNDICE

Comparadores, Circuitos Iterativos – Detalhes adicionais sobre o projeto:



Projeto de um Comparador de Magnitude

C^i_{IN} – Informação proveniente da fatia menos significativa

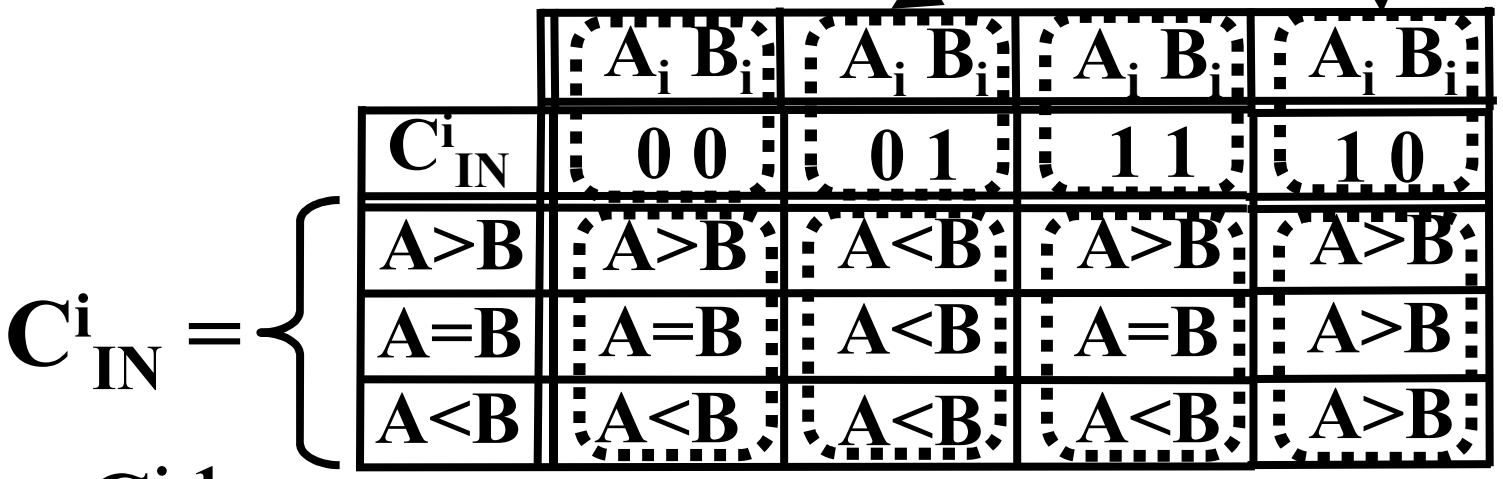


C^i_{IN}	$(A>B)^i$	$(A=B)^i$	$(A<B)^i$
A>B	1	0	0
A=B	0	1	0
A<B	0	0	1

Projeto de um Comparador de Magnitude

Comparação (C^i_{OUT}) decidida nesta fatia mais significativa.

Comparação (C^i_{OUT}) independe do valor da entrada (C^i_{IN}).



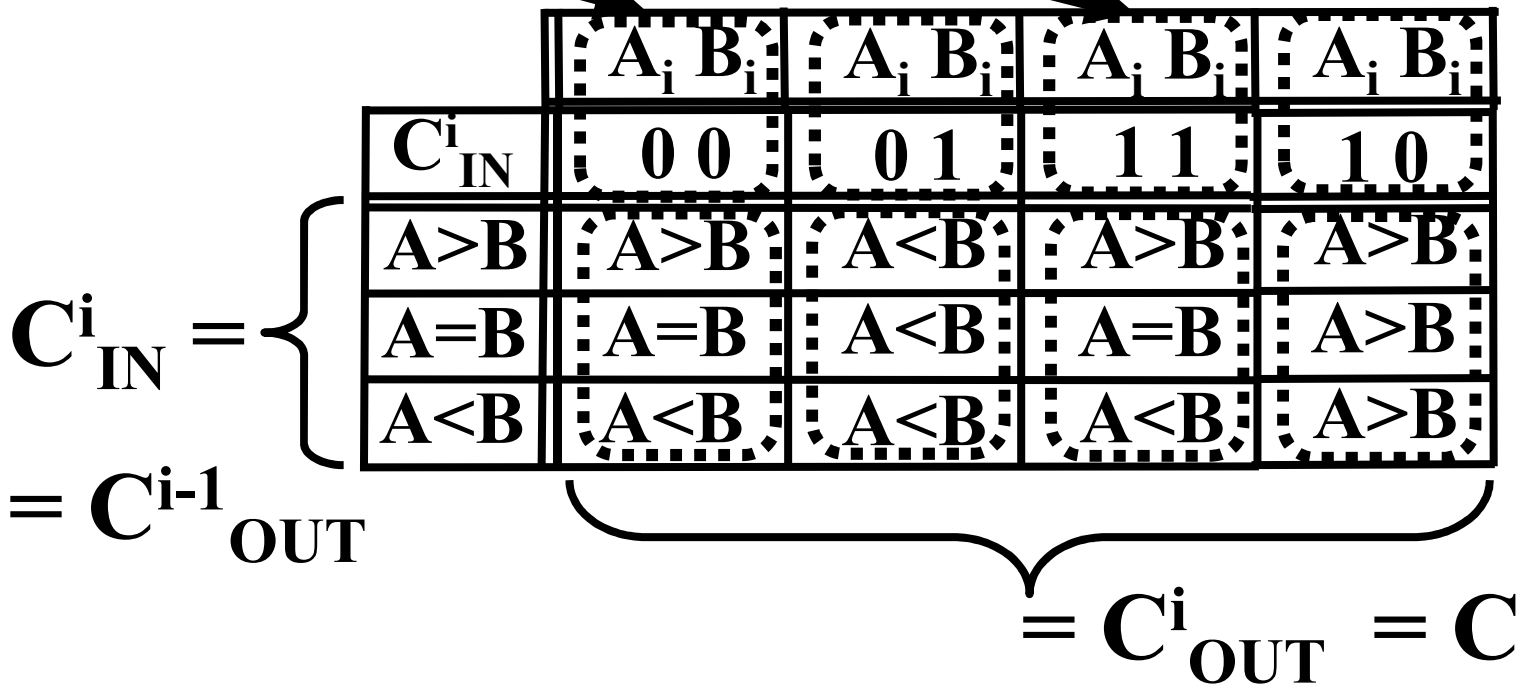
$$C^i_{IN} = \begin{cases} A>B \\ A=B \\ A<B \end{cases} = C^{i-1}_{OUT}$$

$$= C^i_{OUT} = C^{i+1}_{IN}$$

Projeto de um Comparador de Magnitude

Comparação (C^i_{OUT}) será decidida pelo valor de entrada (C^i_{IN}).

Comparação (C^i_{OUT}) depende do valor de entrada (C^i_{IN}), pois não é decidida nesta fatia.



Projeto de um Comparador de Magnitude

$$(A>B)^i_{OUT} = C^{i+1}_{IN}$$

$(A>B)^i_{IN}$ \ $A_i B_i$	0 0	0 1	1 1	1 0
0	0	0	0	1
1	1	0	1	1

$$(A>B)^i_{OUT} =$$

$$= A_i \cdot \overline{B_i} + (A>B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}]}_{= (A=B)^i_{Interno}}$$

Projeto de um Comparador de Magnitude

$$(A=B)^i_{OUT} = C^{i+1}_{IN}$$

$(A=B)^i_{IN}$ \ $A_i\,B_i$	0 0	0 1	1 1	1 0
0	0	0	0	0
1	**1**	0	**1**	0

$$(A=B)^i_{OUT} = (A=B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}]}_{= (A=B)^i_{Interno}}$$

Projeto de um Comparador de Magnitude

$$(A<B)^i_{OUT} = C^{i+1}_{IN}$$

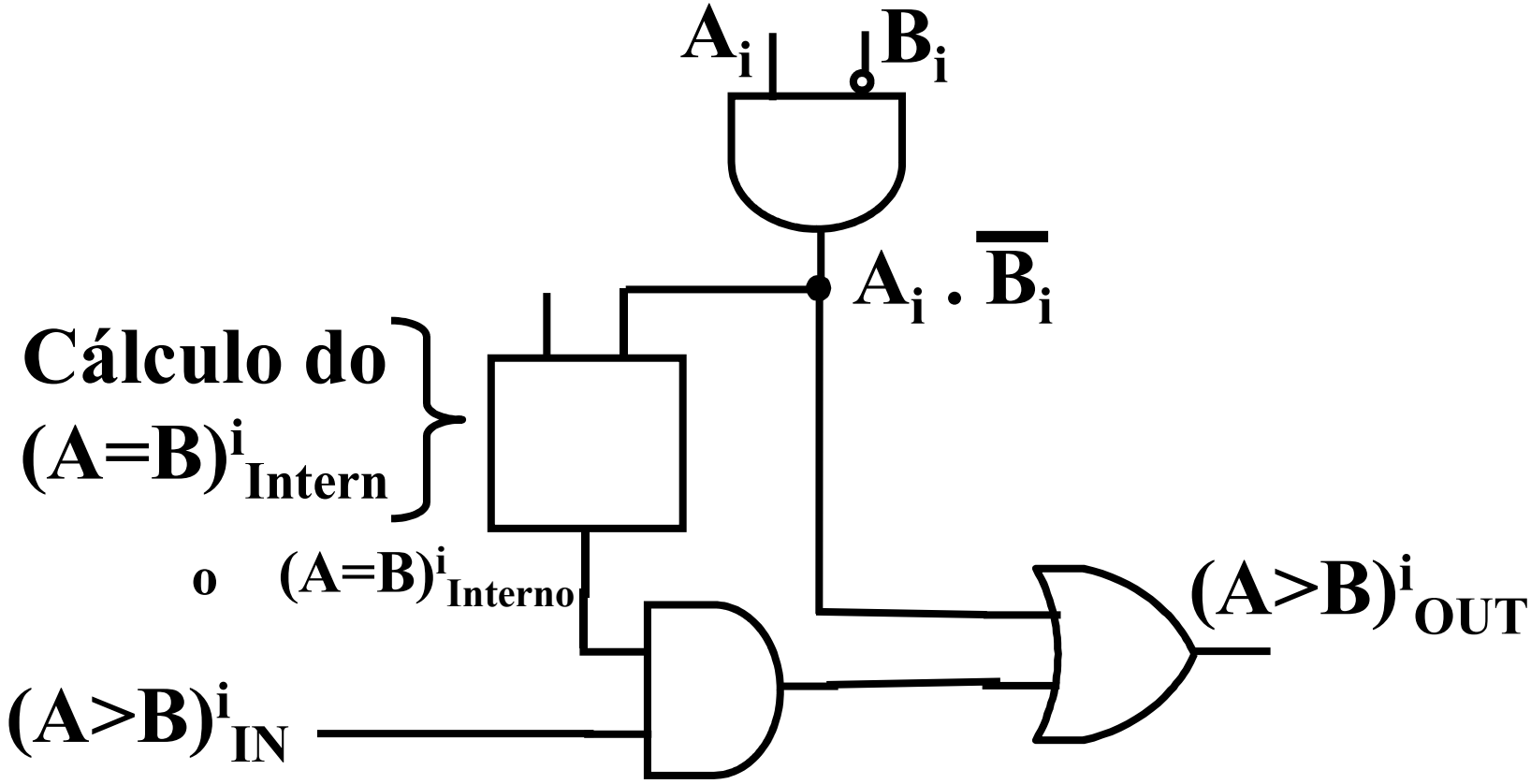
$(A<B)^i_{IN}$ \ $A_i B_i$	0 0	0 1	1 1	1 0
0	0	1	0	0
1	1	1	1	0

$$(A<B)^i_{OUT} =$$

$$= \overline{A}_i \cdot B_i + (A<B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A}_i \cdot \overline{B}_i]}_{= (A=B)^i_{Interno}}$$

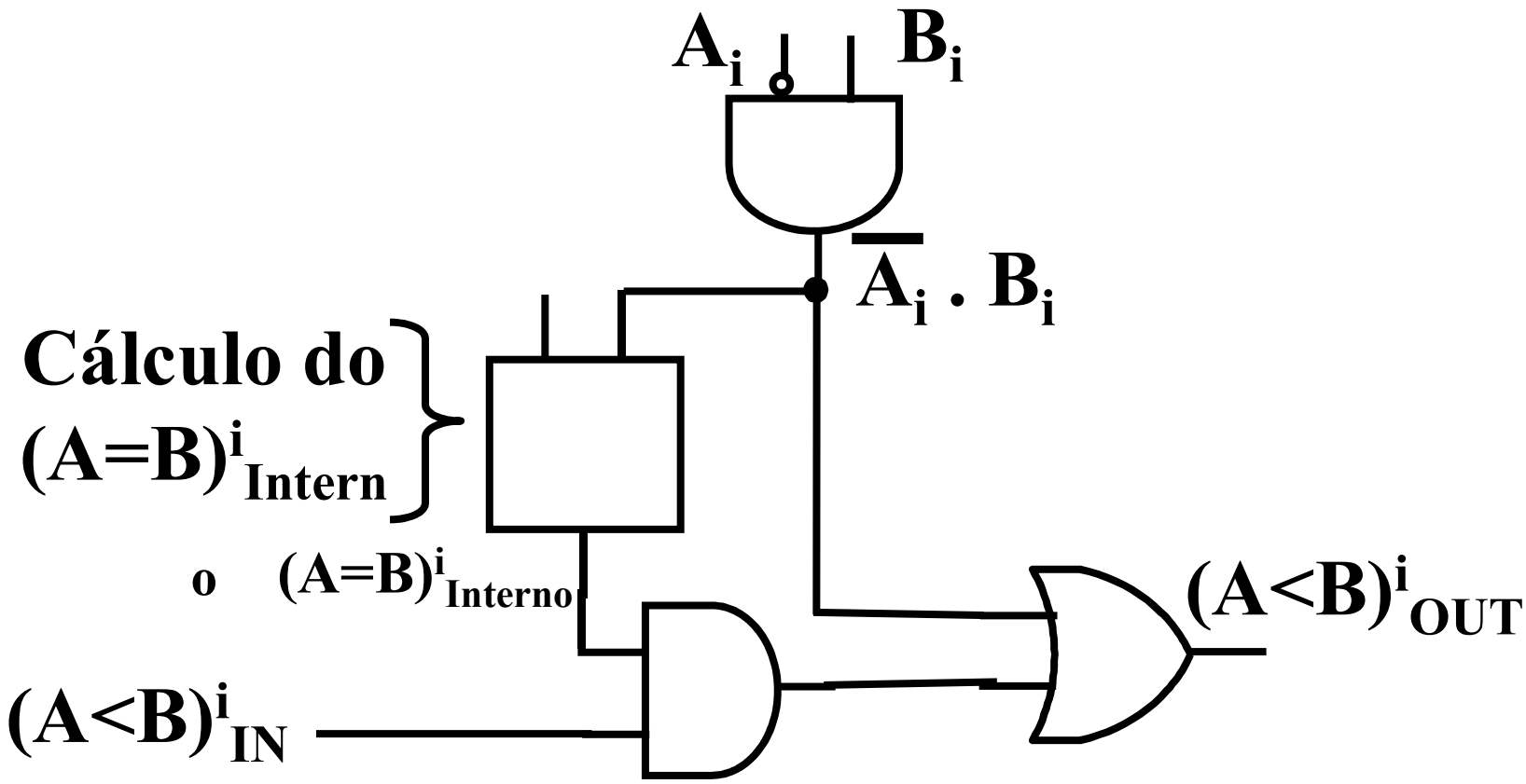
Projeto de um Comparador de Magnitude

$$(A>B)^i_{OUT} = A_i \cdot \overline{B_i} + (A>B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}]}_{(A=B)^i_{Interno}}$$



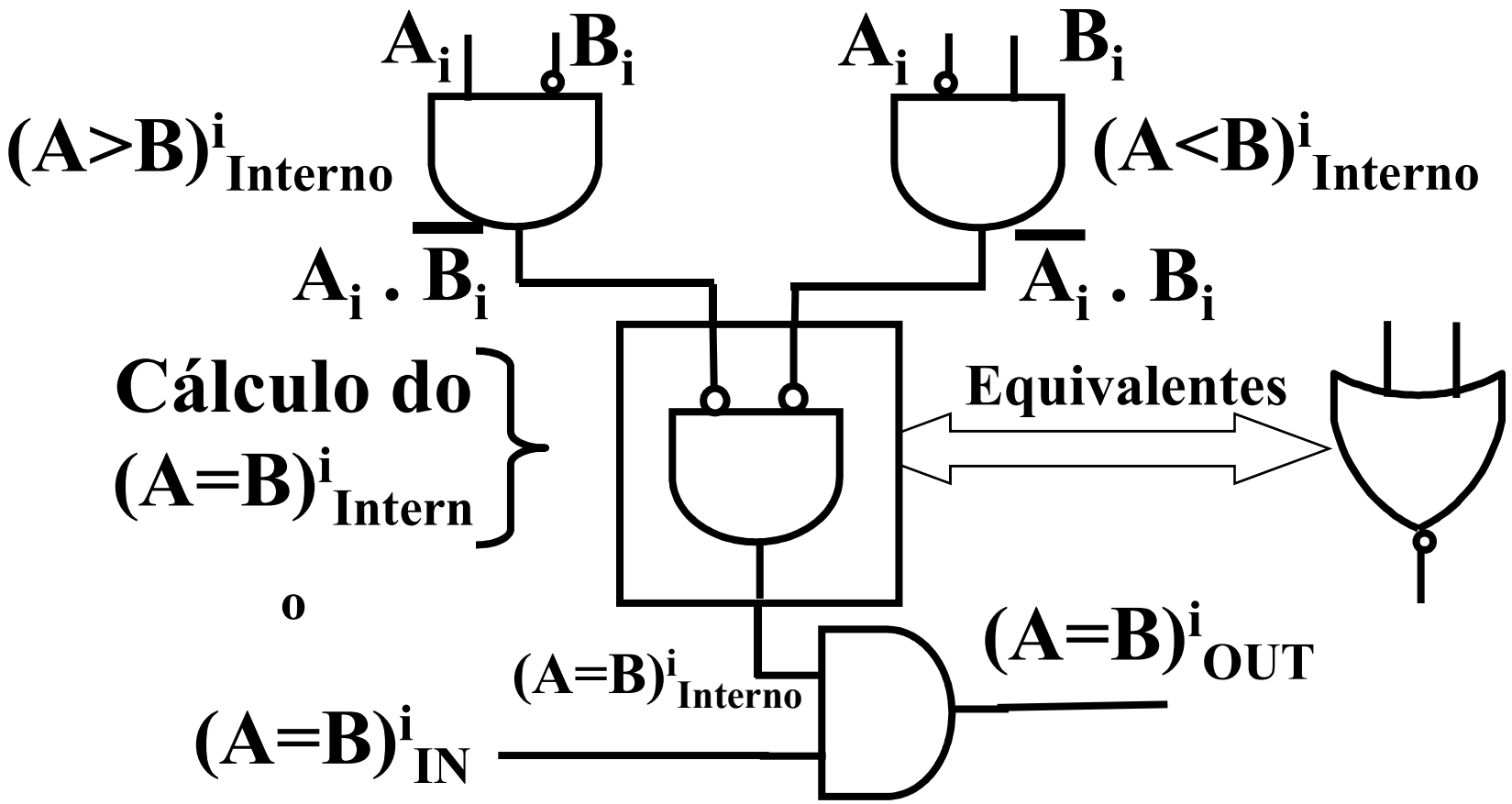
Projeto de um Comparador de Magnitude

$$(A<B)^i_{OUT} = \overline{A}_i \cdot B_i + (A<B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A}_i \cdot \overline{B}_i]}_{(A=B)^i_{Interno}}$$

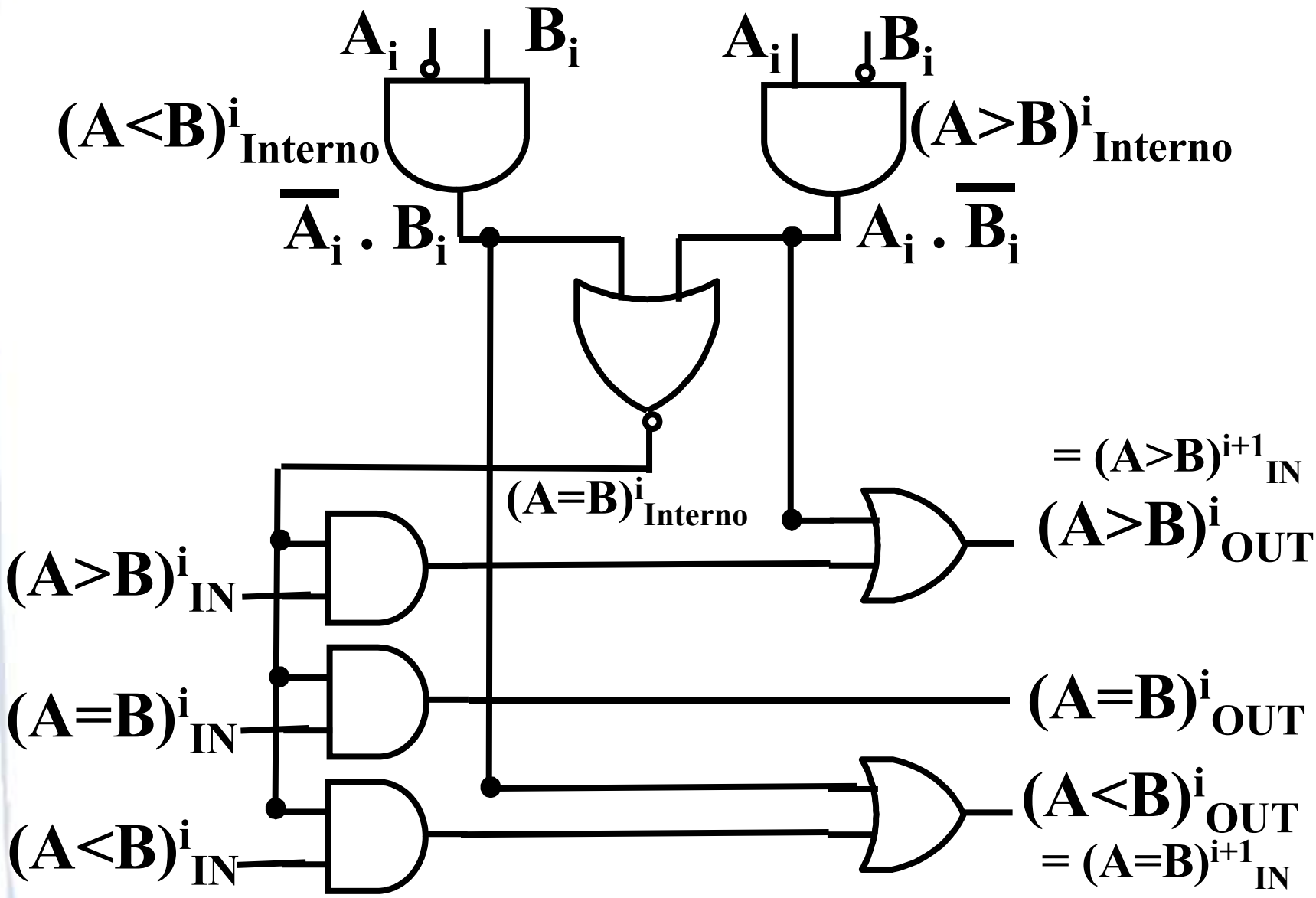


Projeto de um Comparador de Magnitude

$$(A=B)^i_{OUT} = (A>B)^i_{IN} \cdot \underbrace{[A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}]}_{= (A=B)^i_{Interno}}$$



Projeto de um Comparador de Magnitude



Lição de Casa

- Leitura Obrigatória:
  - Capítulo 6 do Livro Texto, ênfase em 6.9.
- Exercícios obrigatórios:
  - Capítulo 6 do Livro Texto.

Livro Texto

- Wakerly, J.F.; *Digital Design – Principles & Practices;* Fourth Edition, ISBN: 0-13-186389-4, Pearson & Prentice-Hall, Upper Saddle, River, New Jersey, 07458, 2006.

Bibliografia Adicional

- Fregni, Edson; Saraiva, Antonio Mauro *Engenharia do Projeto Lógico Digital*. Editora Edgard Blücher Ltda. São Paulo, SP, Brasil, 1.995;

- Tocci, Ronald J.; Widmer, Neal S.; Moss, Gregory L. *Sistemas Digitais – Princípios e Aplicações*. Pearson Prentice Hall, 10a edição, São Paulo, SP, Brasil, 2.007.