CIRCUITOS DIGITAIS
MAIS CIRCUITOS COMBINACIONAIS

Marco A. Zanata Alves

Slides baseados nos slides de Marcelo Andrade da Costa Vieira - EESC/USP (2016)
http://iris.sel.eesc.usp.br/sel414m/

COMPARADOR BINÁRIO

EXERCÍCIO COMPARADOR

Faça um comparador binário com 3 saídas, maior, menor e igual.

- 1 bit de entrada s/ sinal
- 2 bits de entrada s/ sinal
- 8 bits de entrada s/ sinal

EXERCÍCIO COMPARADOR 1-BIT

Faça um comparador binário com 3 saídas:

$X: a < b$

$Y: a = b$

$Z: a > b$

Entrada: 1 bit de entrada a e b sem sinal

EXERCÍCIO COMPARADOR 1-BIT

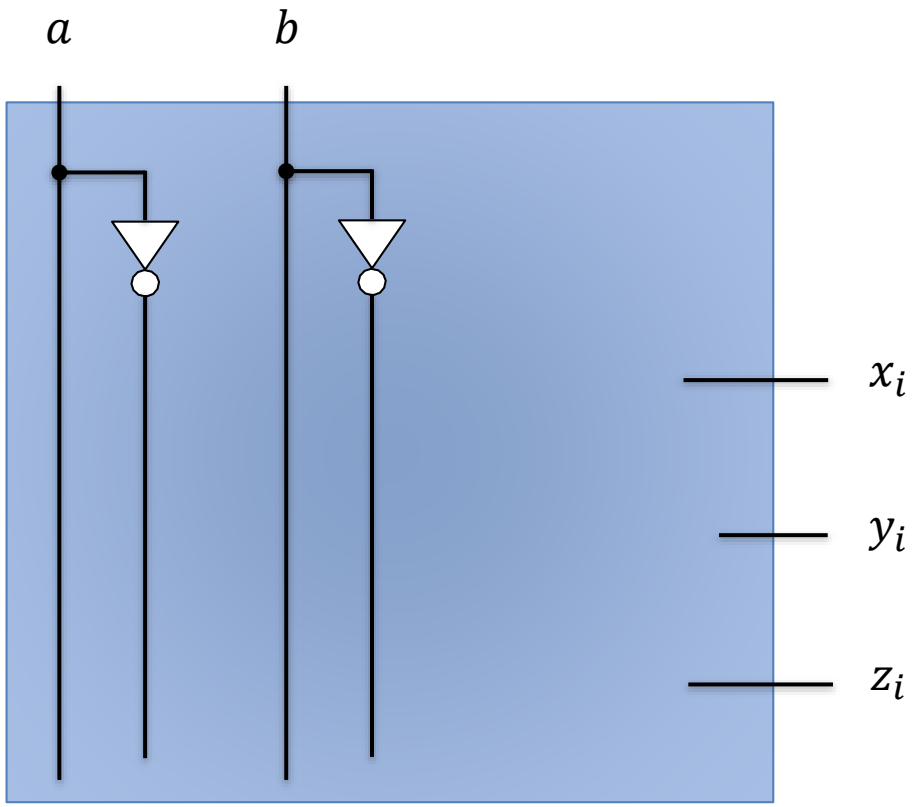
Faça um comparador binário com 3 saídas:

$X: a < b$

$Y: a = b$

$Z: a > b$

Entrada: 1 bit de entrada a e b sem sinal



EXERCÍCIO COMPARADOR 1-BIT

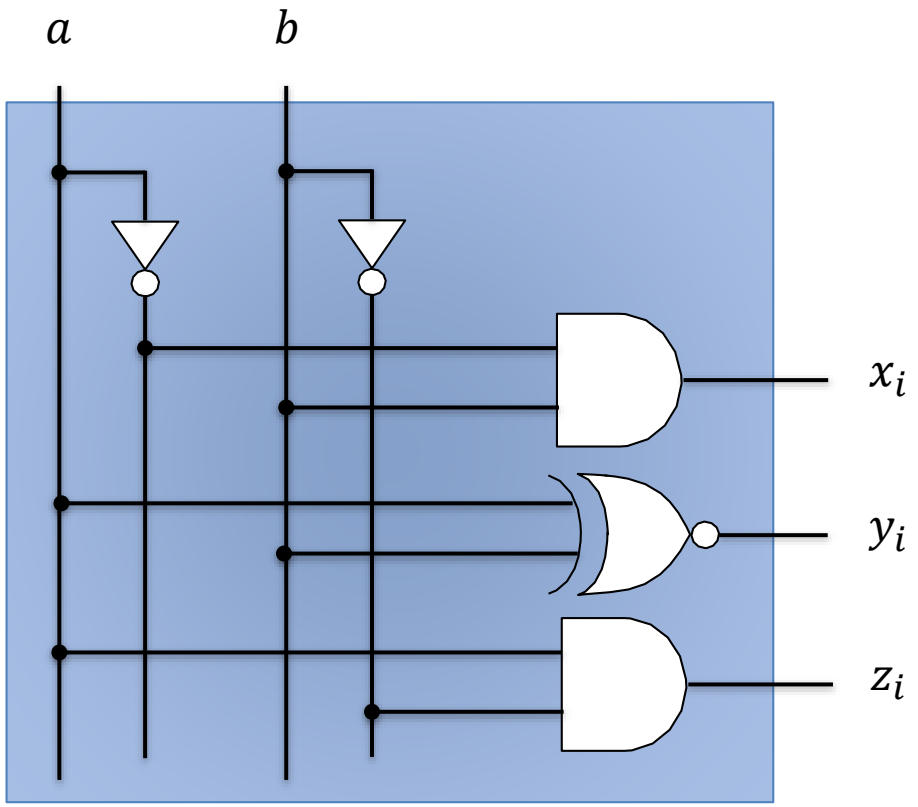
Faça um comparador binário com 3 saídas:

$X: a < b$

$Y: a = b$

$Z: a > b$

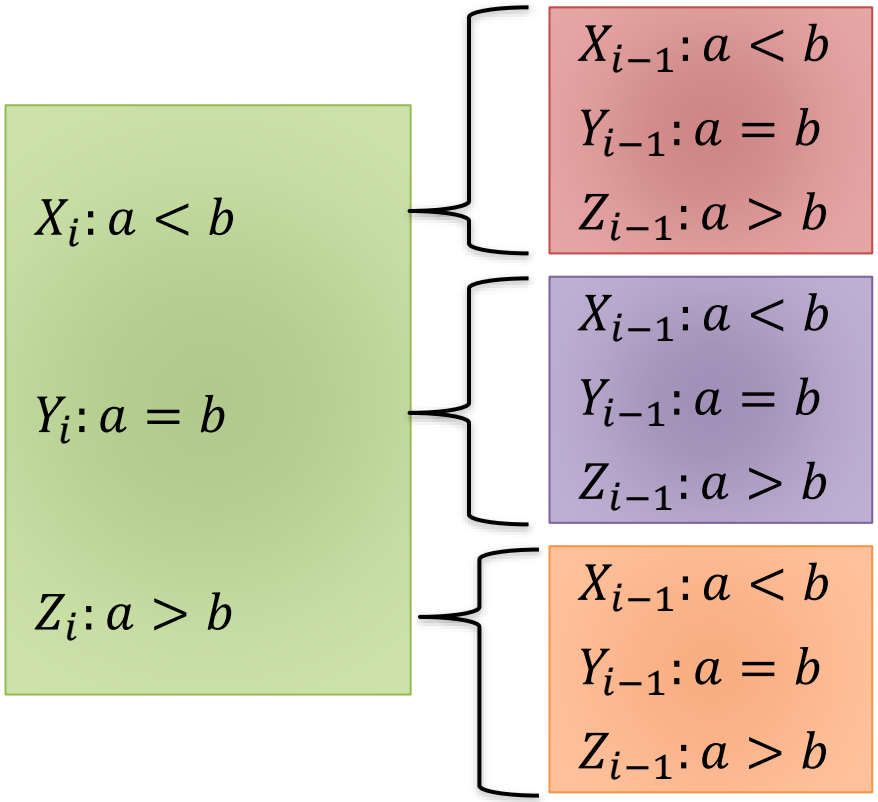
Entrada: 1 bit de entrada a e b sem sinal



EXERCÍCIO COMPARADOR 2-BITS

Faça um comparador binário com 3 saídas, maior, menor e igual.

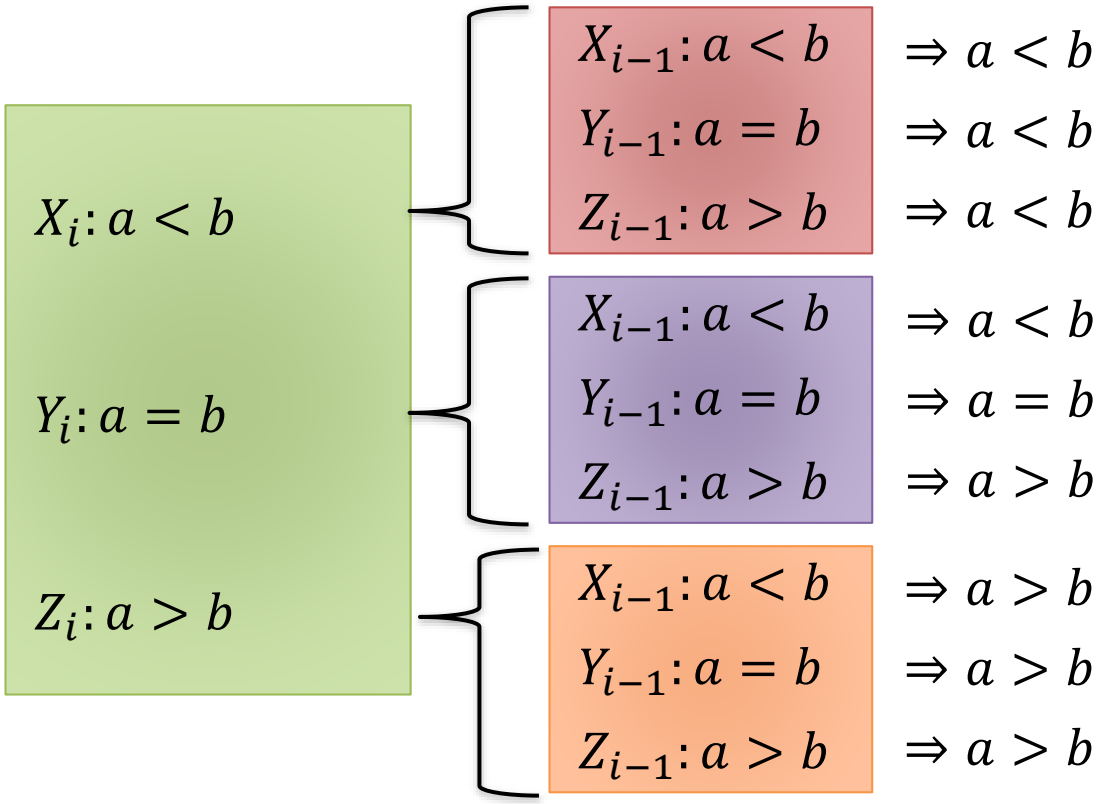
- 2 bits de entrada s/ sinal



EXERCÍCIO COMPARADOR 2-BITS

Faça um comparador binário com 3 saídas, maior, menor e igual.

- 2 bits de entrada s/ sinal



EXERCÍCIO COMPARADOR 2-BITS

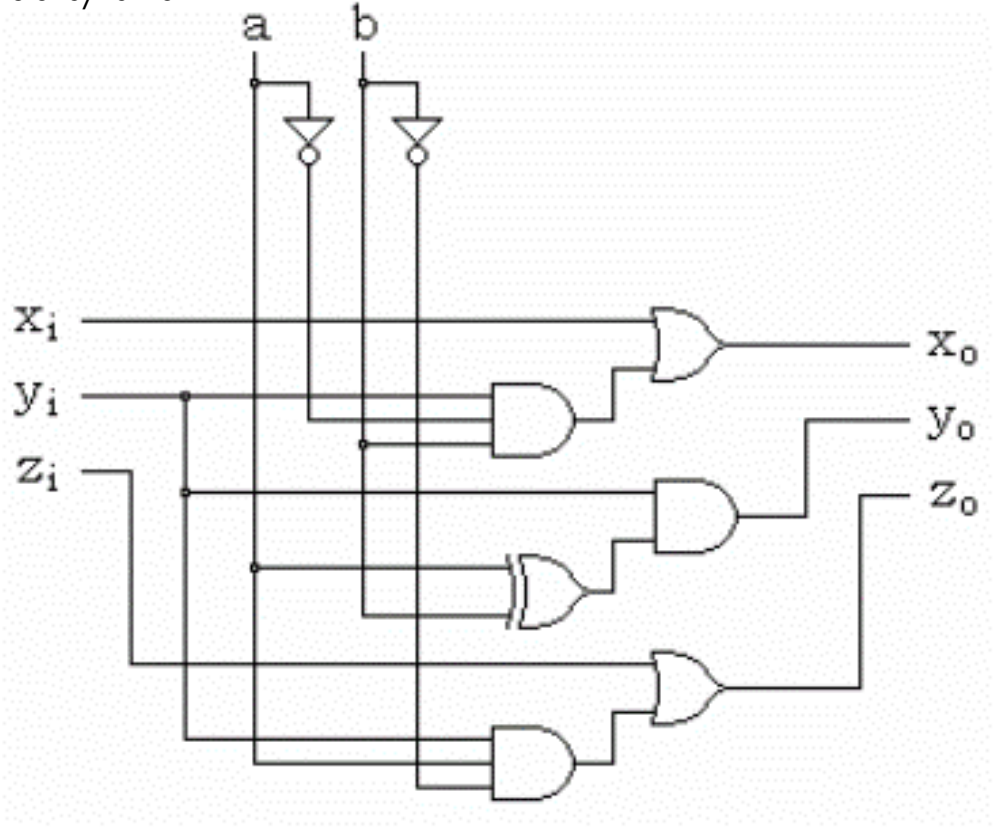
Faça um comparador binário com 3 saídas, maior, menor e igual.

- 2 bits de entrada s/ sinal

$X: a < b$

$Y: a = b$

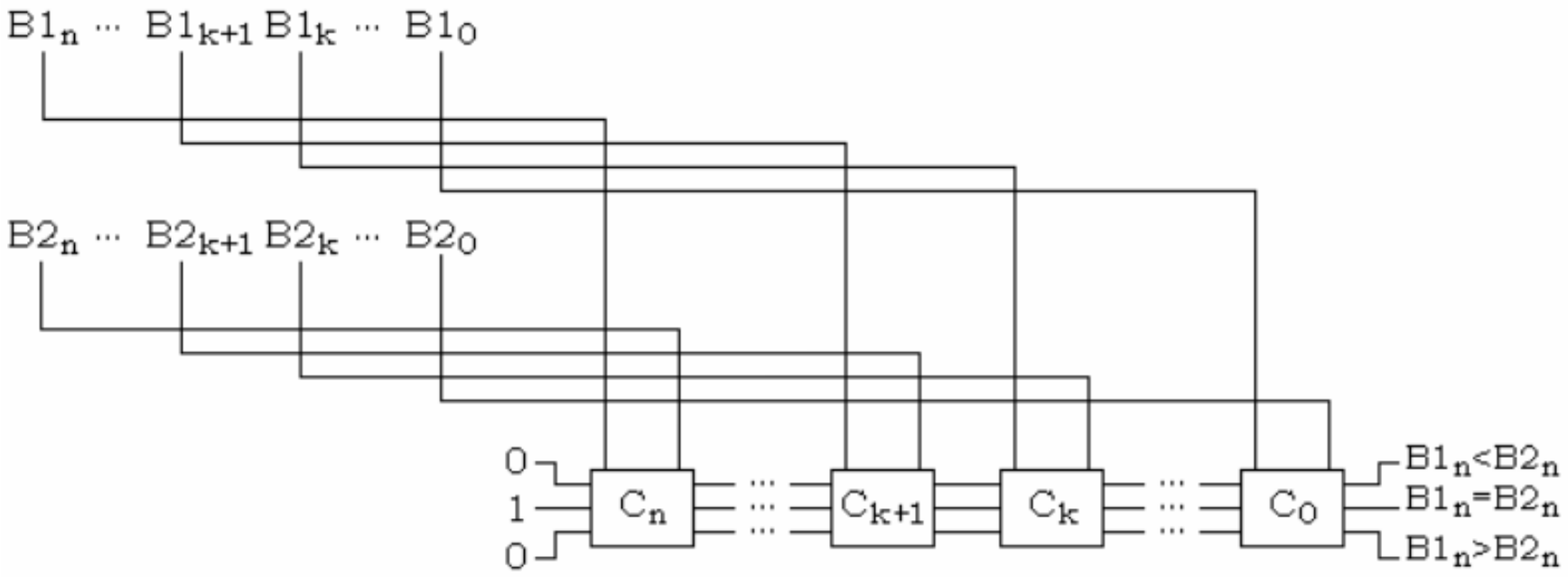
$Z: a > b$



EXERCÍCIO COMPARADOR N-BITS

Faça um comparador binário com 3 saídas, maior, menor e igual.

- 8 bits de entrada s/ sinal



EXERCÍCIO COMPARADOR DO SINAL MAGNITUDE

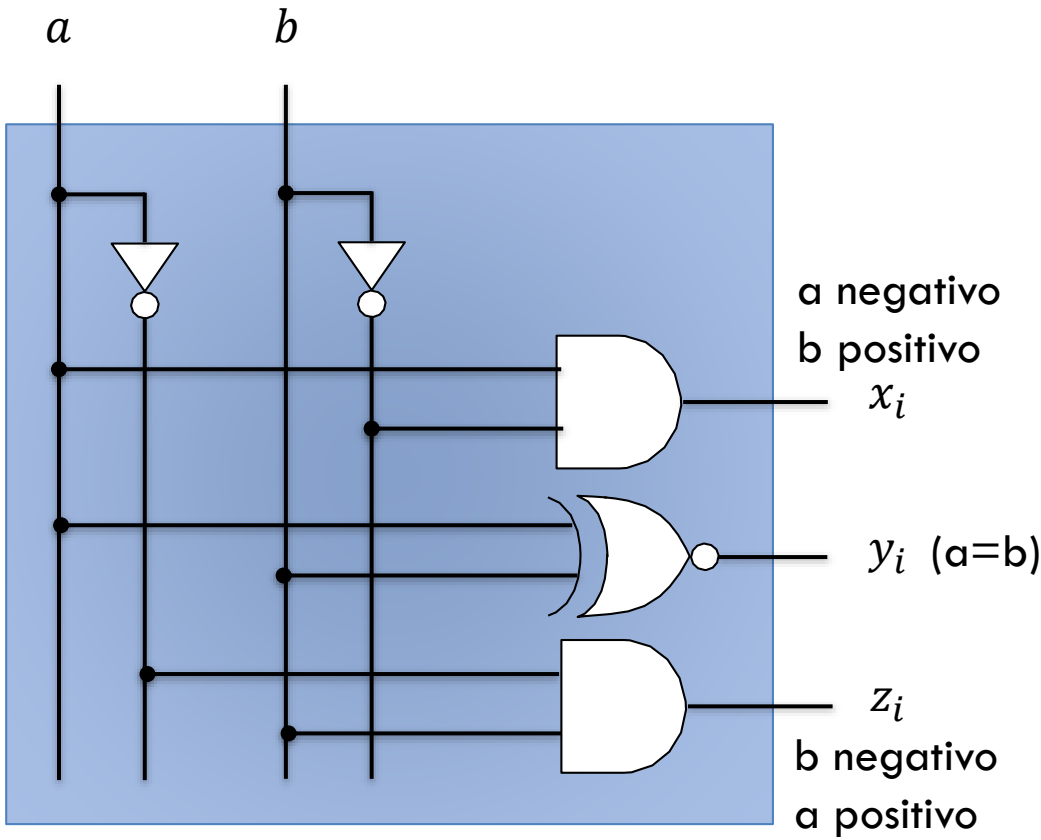
Faça um comparador binário com 3 saídas:

$X: a < b$

$Y: a = b$

$Z: a > b$

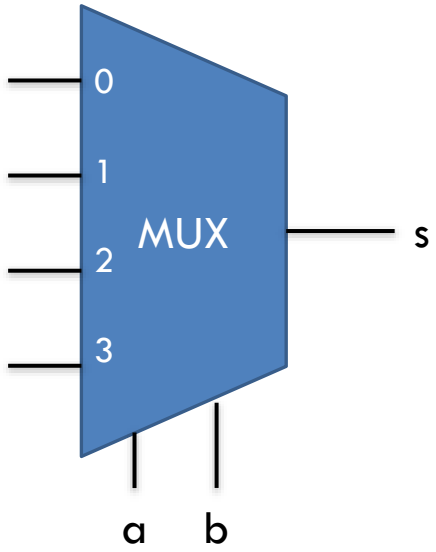
Entrada: 1 bit de entrada a e b representando o sinal magnitute



IMPLEMENTANDO FUNÇÕES COM MULTIPLEXADORES

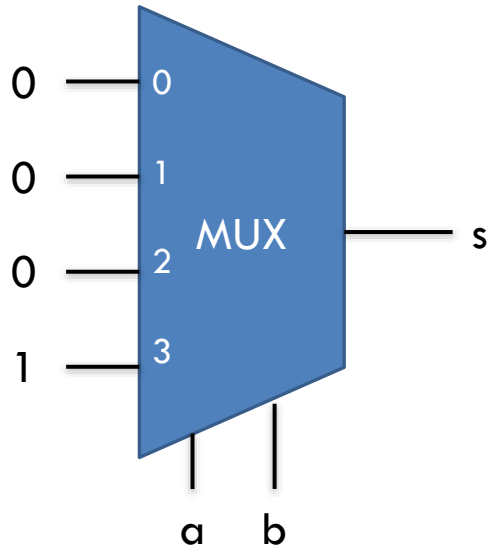
EXERCÍCIO FX-MUX

Implemente uma porta lógica AND de duas entradas com um MUX.



EXERCÍCIO FX-MUX

Implemente uma porta lógica AND de duas entradas com um MUX.



EXERCÍCIO FX-MUX

Implemente a porta lógica XOR de três entradas com um MUX 8:1.

EXERCÍCIO FX-MUX

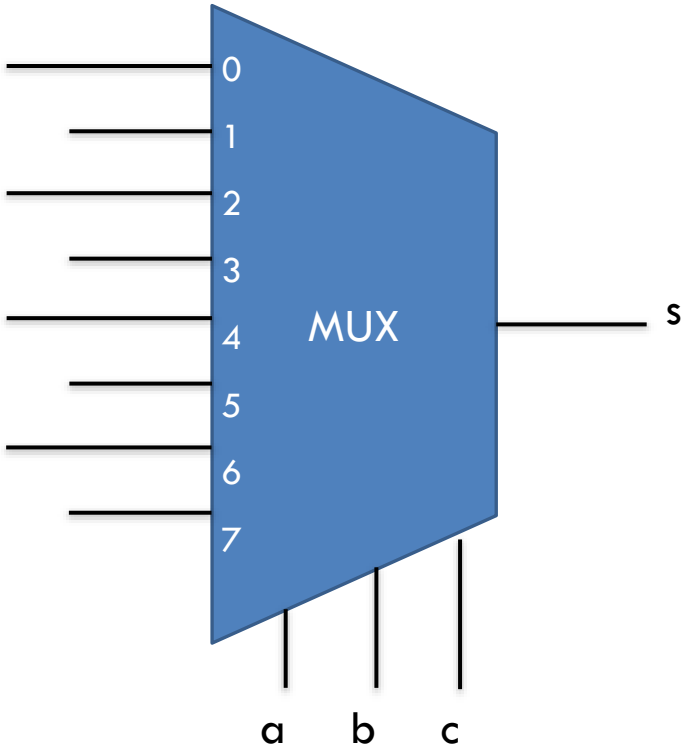
Implemente a porta lógica XOR de três entradas com um MUX 8:1.

A	B	C	Saída
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

EXERCÍCIO FX-MUX

Implemente a porta lógica XOR de três entradas com um MUX 8:1.

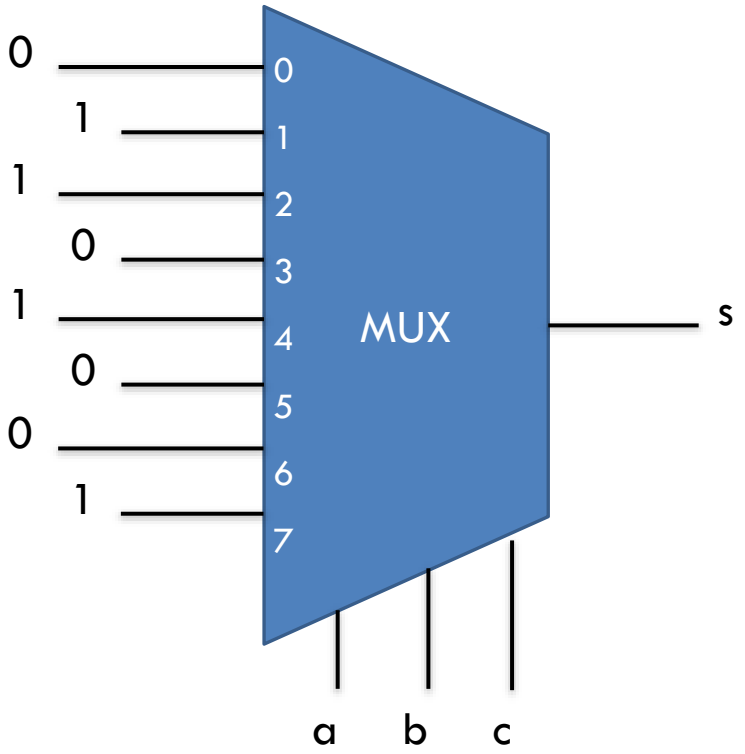
A	B	C	Saída
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



EXERCÍCIO FX-MUX

Implemente a porta lógica XOR de três entradas com um MUX 8:1.

A	B	C	Saída
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

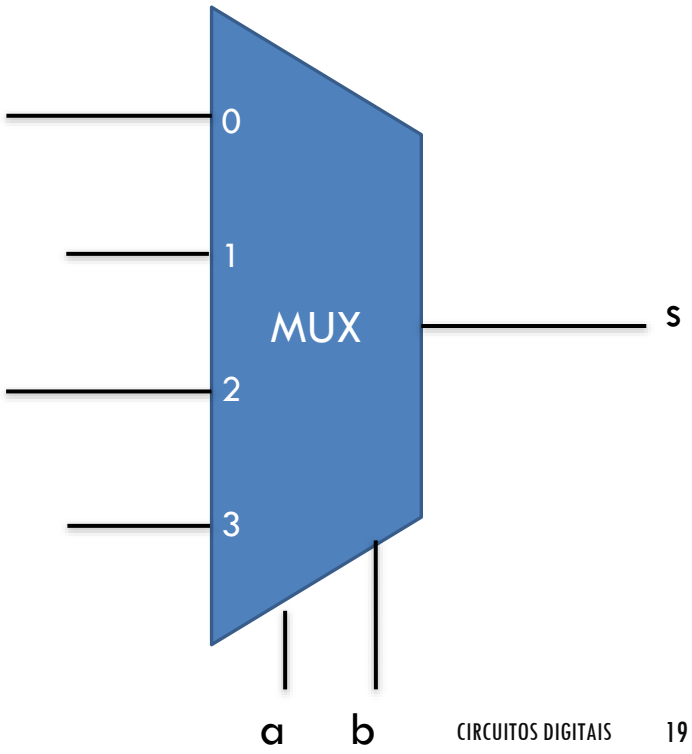


EXERCÍCIO FX-MUX

Implemente a porta lógica XOR de três entradas com um MUX 4:1.

Faça em função de C !

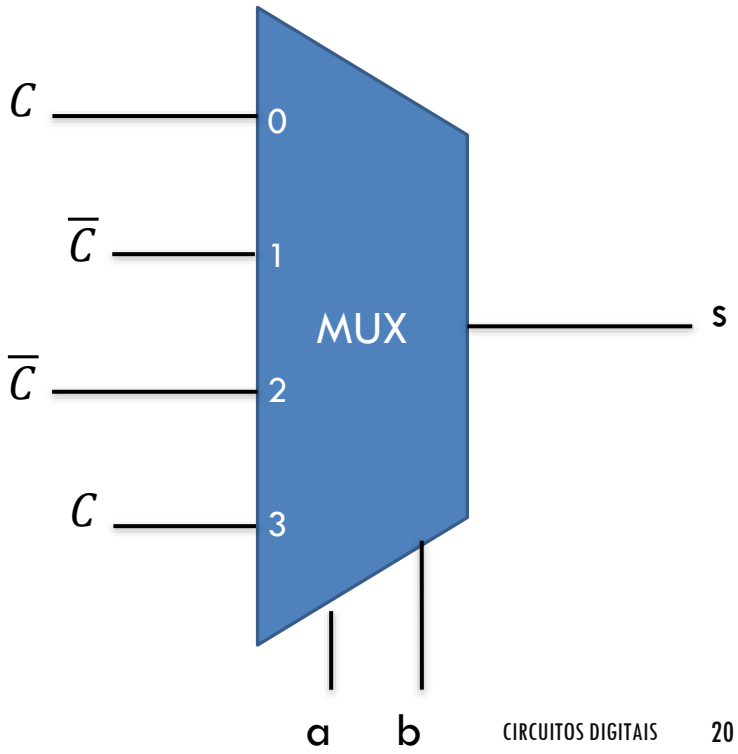
A	B	C	Saída
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



EXERCÍCIO FX-MUX

Implemente a porta lógica XOR de três entradas com um MUX 4:1.

Faça em função de C !

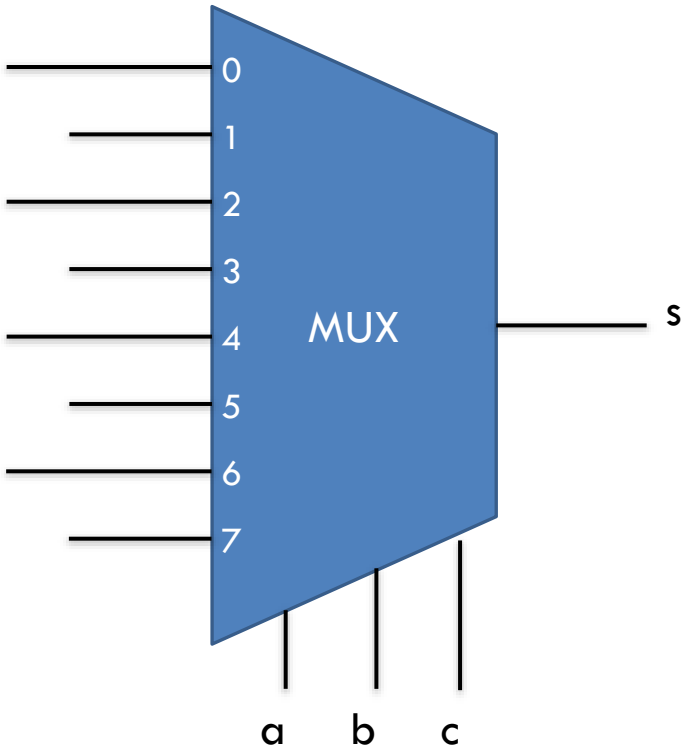


EXERCÍCIO FX-MUX

Implemente a seguinte função:

$$F = A'B'C'D' + A'B'CD + A'BC'D + A'BC'D' + AB'C'D + AB'CD' + ABC'D + ABC'D'$$

- Utilizando portas lógicas
- Utilizando um MUX 8:1

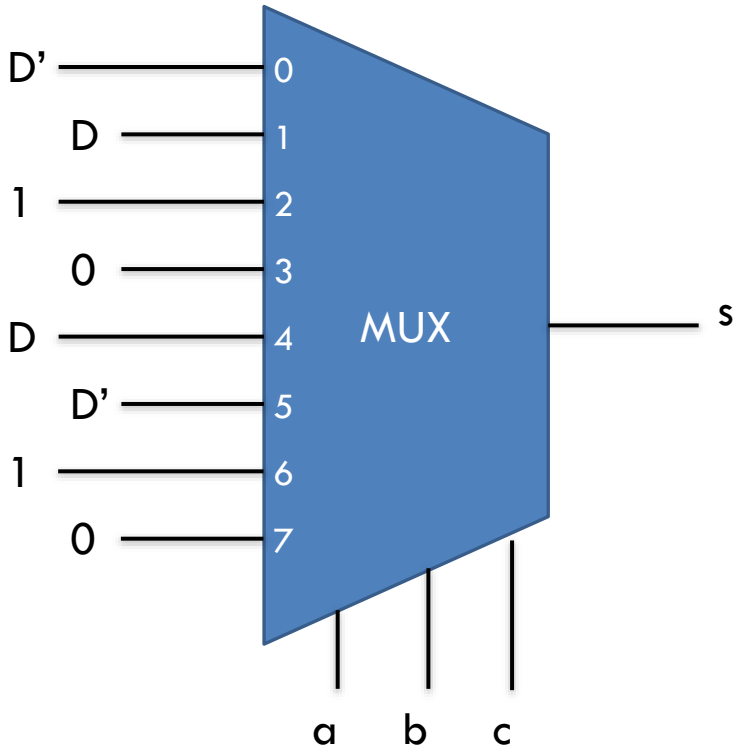


EXERCÍCIO FX-MUX

Implemente a seguinte função:

$$F = A'B'C'D' + A'B'CD + A'BC'D + A'BC'D' + AB'C'D + AB'CD' + ABC'D + ABC'D'$$

- Utilizando portas lógicas
- Utilizando um MUX 8:1



UTILIZANDO DON'T CARE

EXERCÍCIO DON'T CARE

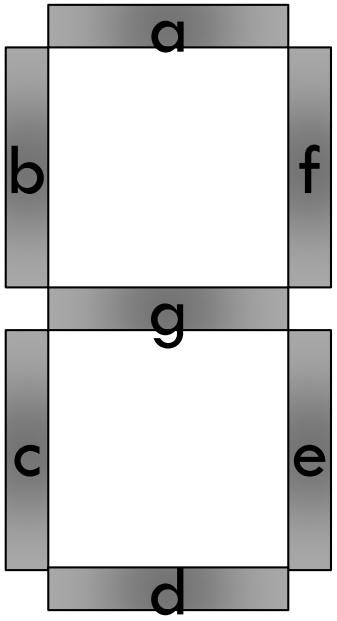
A codificação binária decimal, também conhecida como BCD (Binary-coded decimal), é um sistema de numeração muito utilizado na Informática e em sistemas digitais eletrônicos.

Estamos falando de um sistema de base dois e posicional.

O BCD codifica o sistema decimal em binário, do números (decimais) 0 a 9, onde cada número é representado pelo seu equivalente binário.

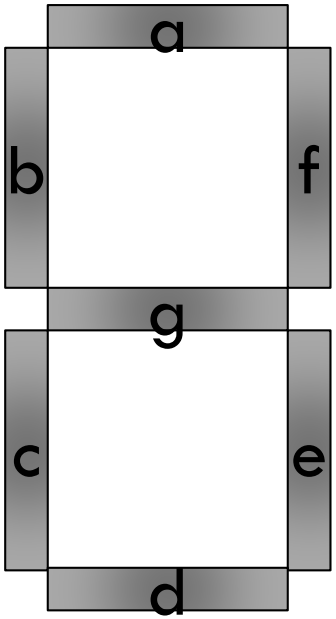
Assim 4 bits são utilizados separadamente para representar cada digito decimal.

Simplifique com mapas de Karnaugh e implemente a lógica para acender o **led a** utilizando a codificação BCD (binário – decimal):



EXERCÍCIO DON'T CARE

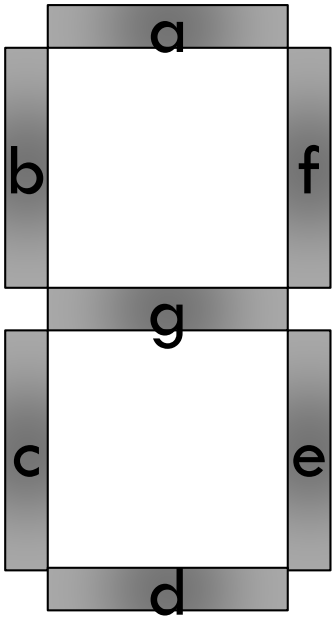
Simplifique com mapas de Karnaugh e implemente a lógica para acender o **led a** utilizando a codificação BCD (binário – decimal):



D	b_3	b_2	b_1	b_0	Led A
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
-	1	0	1	0	
-	1	0	1	1	
-	1	1	0	0	
-	1	1	0	1	
-	1	1	1	0	
-	1	1	1	1	

EXERCÍCIO DON'T CARE

Simplifique com mapas de Karnaugh e implemente a lógica para acender o **led a** utilizando a codificação BCD (binário – decimal):



D	b_3	b_2	b_1	b_0	Led A
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
-	1	0	1	0	X
-	1	0	1	1	X
-	1	1	0	0	X
-	1	1	0	1	X
-	1	1	1	0	X
-	1	1	1	1	X

EXERCÍCIO DON'T CARE

	$b_1\ b_0$	$b_1\ \overline{b_0}$	$\overline{b_1}\ \overline{b_0}$	$\overline{b_1} b_0$
$b_3\ b_2$				
$b_3\ \overline{b_2}$				
$\overline{b_3}\ \overline{b_2}$				
$\overline{b_3} b_2$				

D	b_3	b_2	b_1	b_0	Led A
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
-	1	0	1	0	X
-	1	0	1	1	X
-	1	1	0	0	X
-	1	1	0	1	X
-	1	1	1	0	X
-	1	1	1	1	X

EXERCÍCIO DON'T CARE

	$b_1\, b_0$	$b_1\, \overline{b_0}$	$\overline{b_1}\, \overline{b_0}$	$\overline{b_1} b_0$
$b_3\, b_2$	x	x	x	x
$b_3\, \overline{b_2}$	x	x	1	1
$\overline{b_3}\, \overline{b_2}$	1	1	1	
$\overline{b_3} b_2$	1			1

D	b_3	b_2	b_1	b_0	Led A
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
-	1	0	1	0	X
-	1	0	1	1	X
-	1	1	0	0	X
-	1	1	0	1	X
-	1	1	1	0	X
-	1	1	1	1	X

EXERCÍCIO DON'T CARE

D	b_3	b_2	b_1	b_0	Led A
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
-	1	0	1	0	X
-	1	0	1	1	X
-	1	1	0	0	X
-	1	1	0	1	X
-	1	1	1	0	X
-	1	1	1	1	X

	$b_1\,b_0$	$b_1\,\overline{b_0}$	$\overline{b_1}\,\overline{b_0}$	$\overline{b_1}\,b_0$
$b_3\,b_2$	x	x	x	x
$b_3\,\overline{b_2}$	x	x	1	1
$\overline{b_3}\,\overline{b_2}$	1	1	1	
$\overline{b_3}\,b_2$	1			1

$$A = b_3 + b_1 b_0 + \overline{b_2}\,\overline{b_0} + b_2 b_0$$