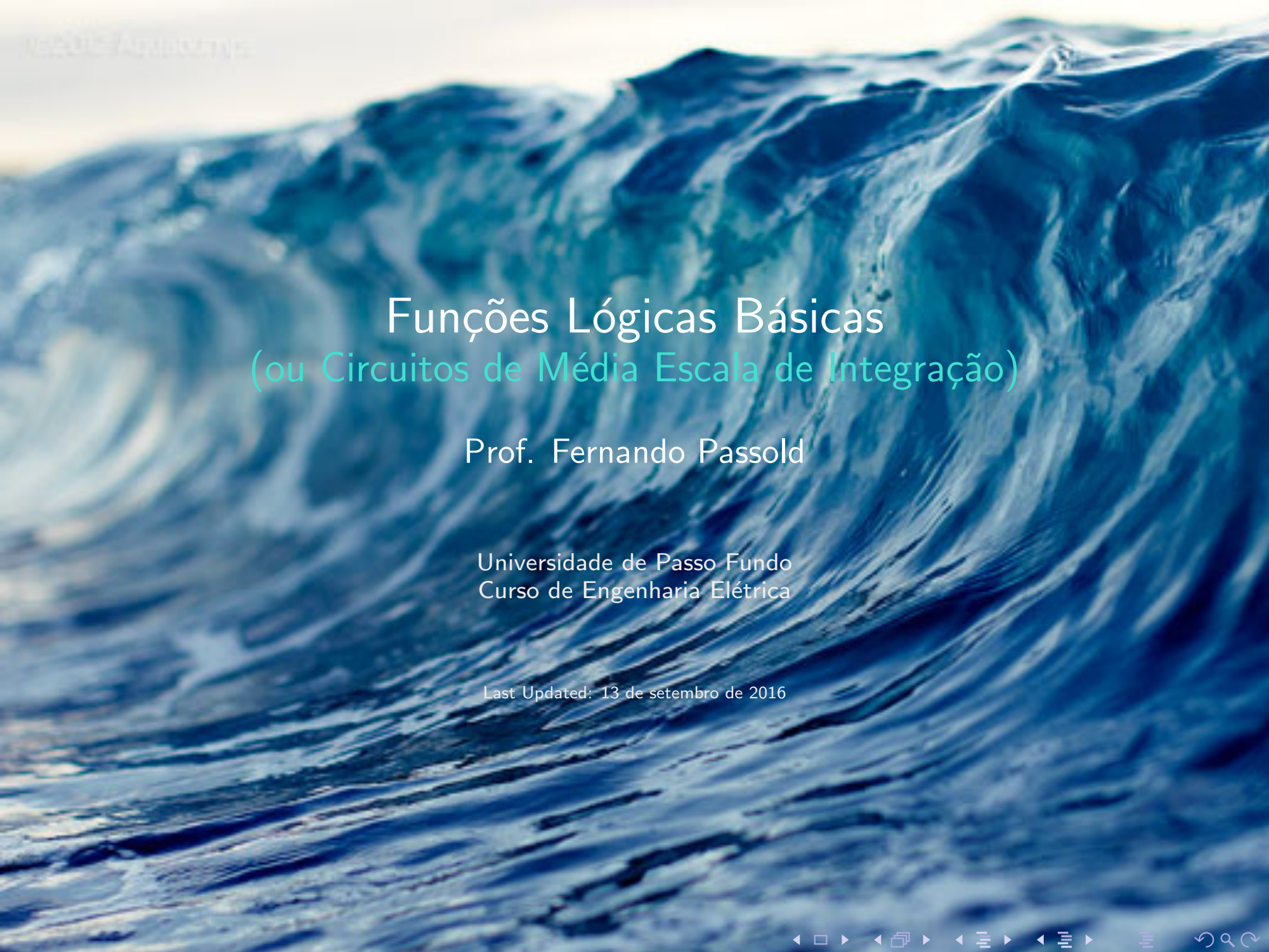


Funções Lógicas Básicas
(ou Circuitos de Média Escala de Integração)

Prof. Fernando Passold

Universidade de Passo Fundo
Curso de Engenharia Elétrica

Last Updated: 13 de setembro de 2016

Índice

Objetivos

1. Reconhecer as diferentes funções lógicas básicas;
2. Saber reconstituir a tabela verdade de uma função lógica básica;
3. Saber expressar algebricamente a(s) equação(ões) que descreve(m) uma função lógica básica;
4. Saber usar pastilhas comerciais MSI relacionadas com funções lógicas básicas.

Circuitos MSI

1. Decodificadores:
   - 2× DEC 2/4 = CI 74139;
   - DEC 3/8 = CI 74138;
   - DEC 4/10 = CI 7442;
   - DEC 4/16 = CI 74154;
2. Codificadores:
   - Codificador 10/4 = CI 74147;
   - Codificador 16/4 = CI 74148;
3. Multiplexadores:
   - 4× MUX-2 = CI 74157, 74158;
   - 2× MUX-4 = CI 74153;
   - MUX-8 = CI 74151;
   - MUX-16 = CI 74150;
4. Demultiplexadores
5. Comparadores de Magnitude: 4 bits = CI 7485;
6. Aplicações

Circuitos MSI

De acordo com a classificação dos CI's quanto ao nível de integração, diretamente relacionado com o número de portas lógicas, são considerados circuitos MSI e LSI os:

- Multiplexadores/seletores de dados;
- Decodificadores/Demultiplexadores;
- Codificadores de prioridade;
- Comparadores, e;
- Circuitos aritméticos.

Escala de integração de circuitos integrados

Abrev.	Denominação	Complexidade (números de transístores)		
		Interpretação comum	Tanenbaum[7]	Texas Instruments[8]
SSI	Small Scale Integration	10	1–10	em baixo de 12
MSI	Medium Scale Integration	100	10–100	12–99
LSI	Large Scale Integration	1.000	100–100.000	100–999
VLSI	Very Large Scale Integration	10.000–100.000	a partir de 100.000	ab 1.000
ULSI	Ultra Large Scale Integration	100.000–1.000.000	—	—
SLSI	Super Large Scale Integration	1.000.000–10.000.000	—	—

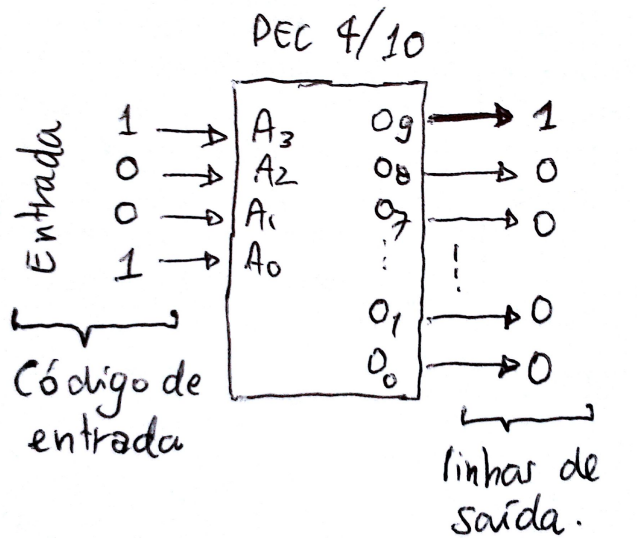
Índice

DECodificadores

Decodificadores

- **DEC**: recebe no seu conjunto de entradas, um número binário e ativa apenas a saída que corresponde ao número recebido.
- Possui n linhas de entrada (correspondente ao código binário de entrada) e ativa somente uma linha de suas 2^n linhas de saída (a que corresponde ao número binário presente na sua entrada).
- Suponha que: num DEC se apresente na suas entradas o código "$1001_{(2)} = 9_{(10)}$": neste caso, somente a linha 0_9 que correspondente à este código será ativada (as outras linhas de saída ficam desativadas) – ver figura ao lado.

Figura exemplo:



n-entradas $\longrightarrow 2^n$ linhas de saída.

Decodificadores

- DEC 2/4 (hipotético):

Suponha o caso de un decodificador de 2 para 4 linhas de saída:
Simbolo:

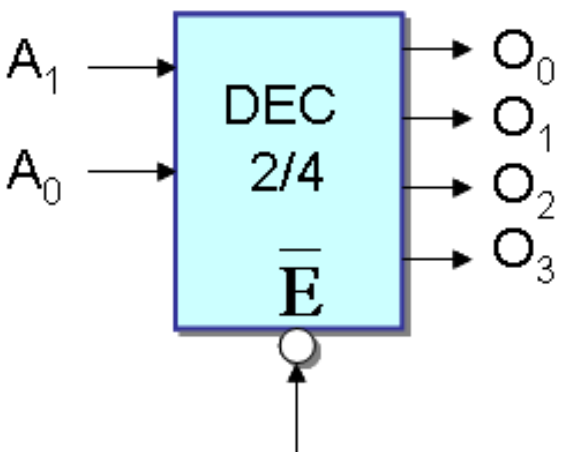


Tabela Verdade (ou funcional):

Entradas			Saídas			
$\overline{E}$	A_1	A_0	O_0	O_1	O_2	O_3
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	X	X	0	0	0	0

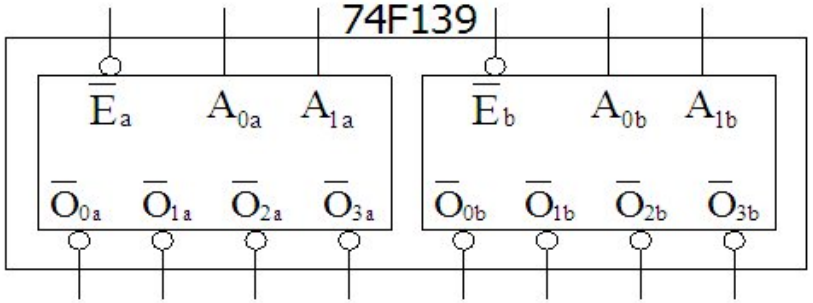
Equações:

$$O_0 = \overline{\overline{E}} \cdot \overline{A_1} \cdot \overline{A_0} = E \cdot \overline{A_1} \cdot \overline{A_0}$$
$$O_1 = \overline{\overline{E}} \cdot \overline{A_1} \cdot A_0$$
$$O_2 = \overline{\overline{E}} \cdot A_1 \cdot \overline{A_0}$$
$$O_3 = \overline{\overline{E}} \cdot A_1 \cdot A_0$$

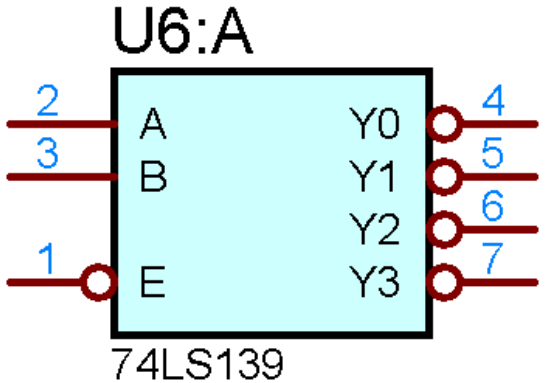
Decodificadores

- DEC (Comercial) Duplo de 2 para 4 linhas de saída:
  CI 74**139**:

Simbolo:



Símbolo no Proteus:



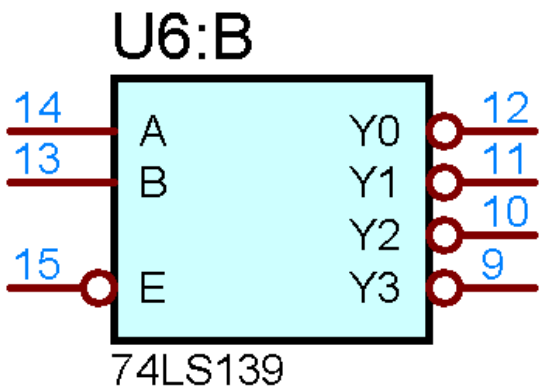


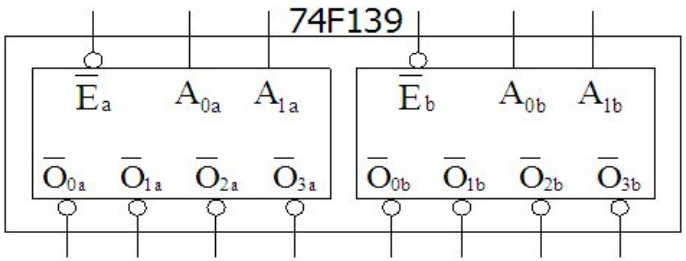
Tabela Verdade:

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

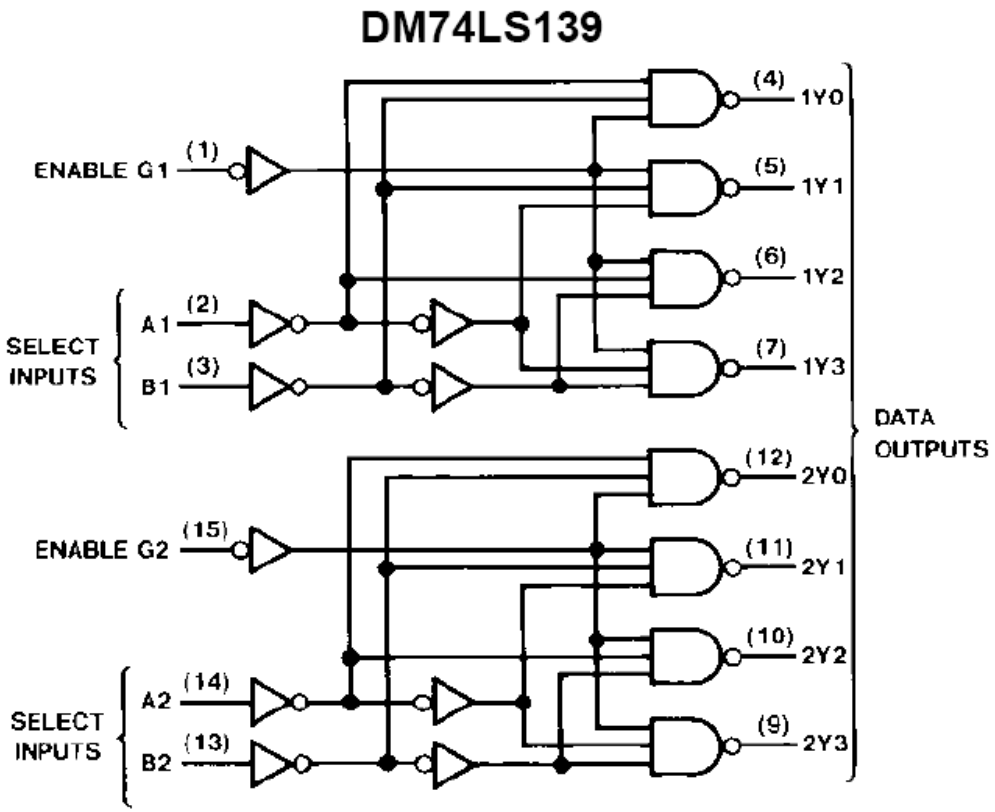
Decodificadores

- DEC (Comercial) Duplo de 2 para 4 linhas de saída: CI 74**139**:

Simbolo:



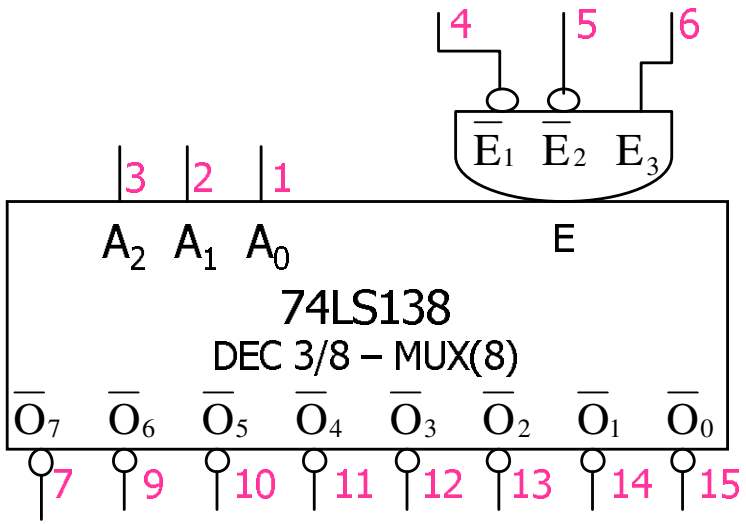
Circuito interno



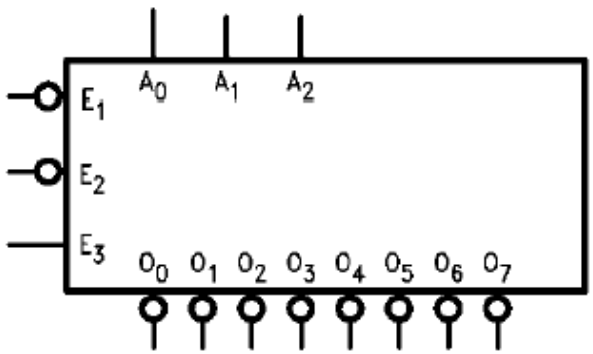
Decodificadores

- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

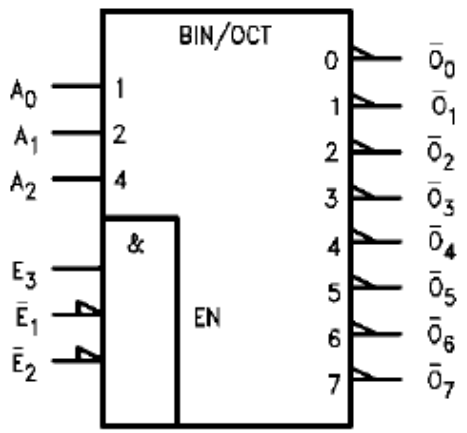
Símbolos:



Vcc=Pin 16
GND=Pin 8



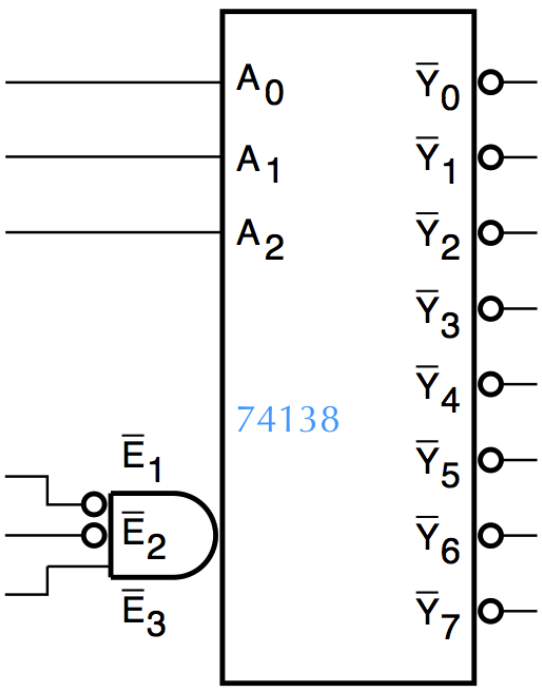
IEEE/IEC



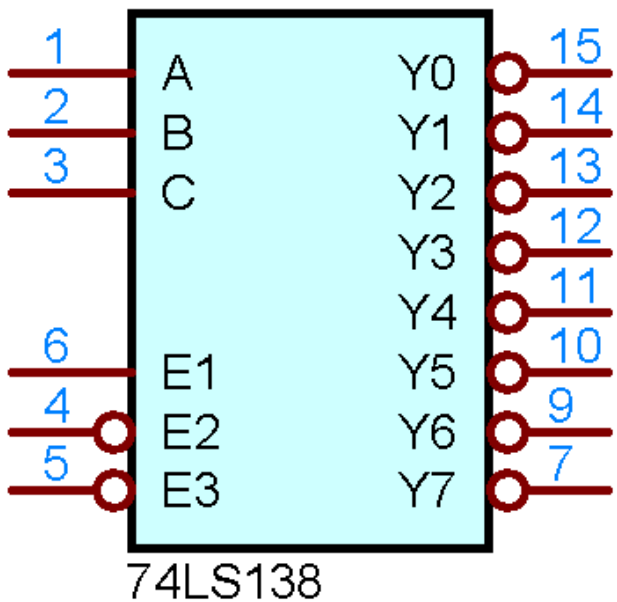
Decodificadores

- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

Símbolo (Philips):



Símbolo Proteus:



Decodificadores

- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

Símbolo (Philips):

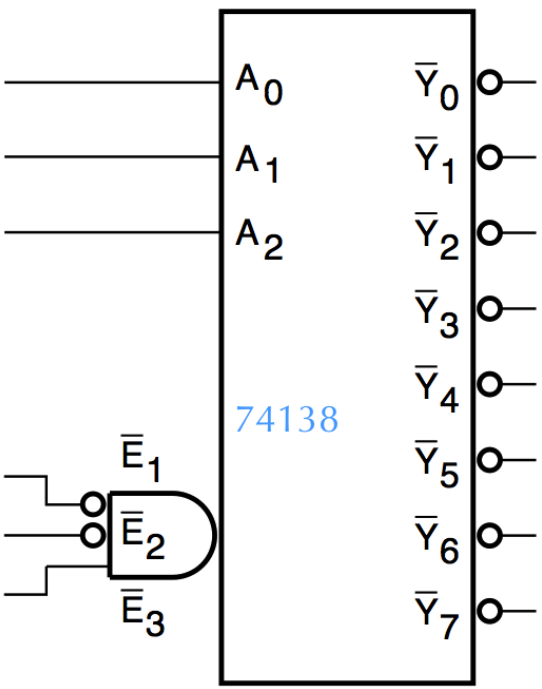


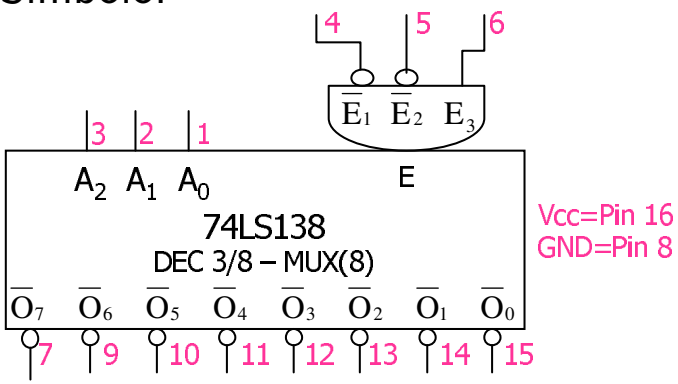
Tabela Verdade:

Inputs						Outputs							
$\bar{E}_1$	$\bar{E}_2$	E_3	A_0	A_1	A_2	$\bar{O}_0$	$\bar{O}_1$	$\bar{O}_2$	$\bar{O}_3$	$\bar{O}_4$	$\bar{O}_5$	$\bar{O}_6$	$\bar{O}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

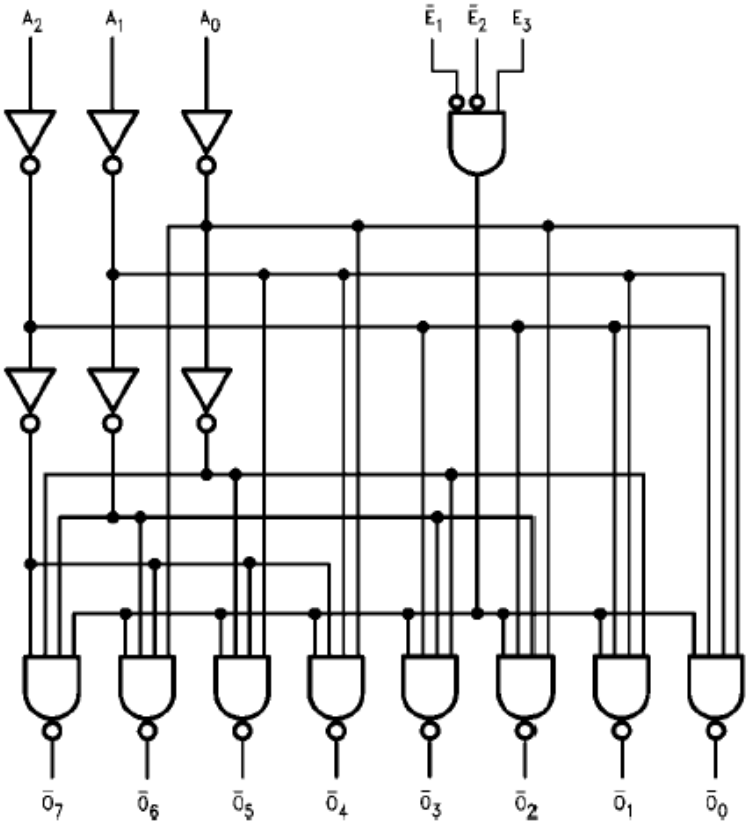
Decodificadores

- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

Simbolo:

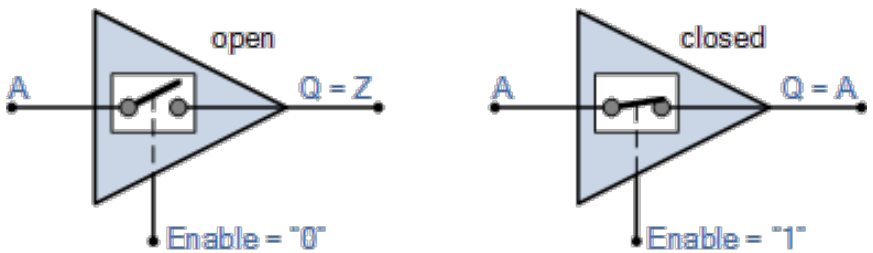


Circuito interno:



Saídas Tri-state

Existe um "terceiro estado lógico" que corresponde à saída que pode ser "desconectada eletrônicamente" do circuito onde sua saída esteja conectada. Corresponde ao que se denomina de "Estado de Alta Impedância" ou "Hi-Z". A figura à seguir mostra o caso de um buffer tri-state:

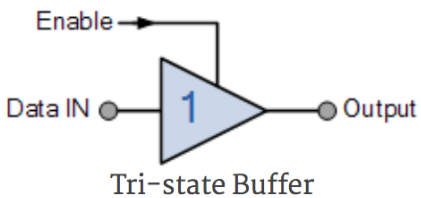


Repare que sua saída pode ser "desconectada" do circuito ao qual está conectado, quando sua entrada de *Enable* estiver desabilitada.

$$\text{No caso da figura: } \begin{cases} \text{quando } E = 0, & Q = Z \text{ ou } Q = Hi - Z \\ \text{quando } E = 1, & Q = A \end{cases}$$

Saídas Tri-state

A figura abaixo mostra o caso de um buffer não-inversor tri-state acompanhado de sua tabela verdade.

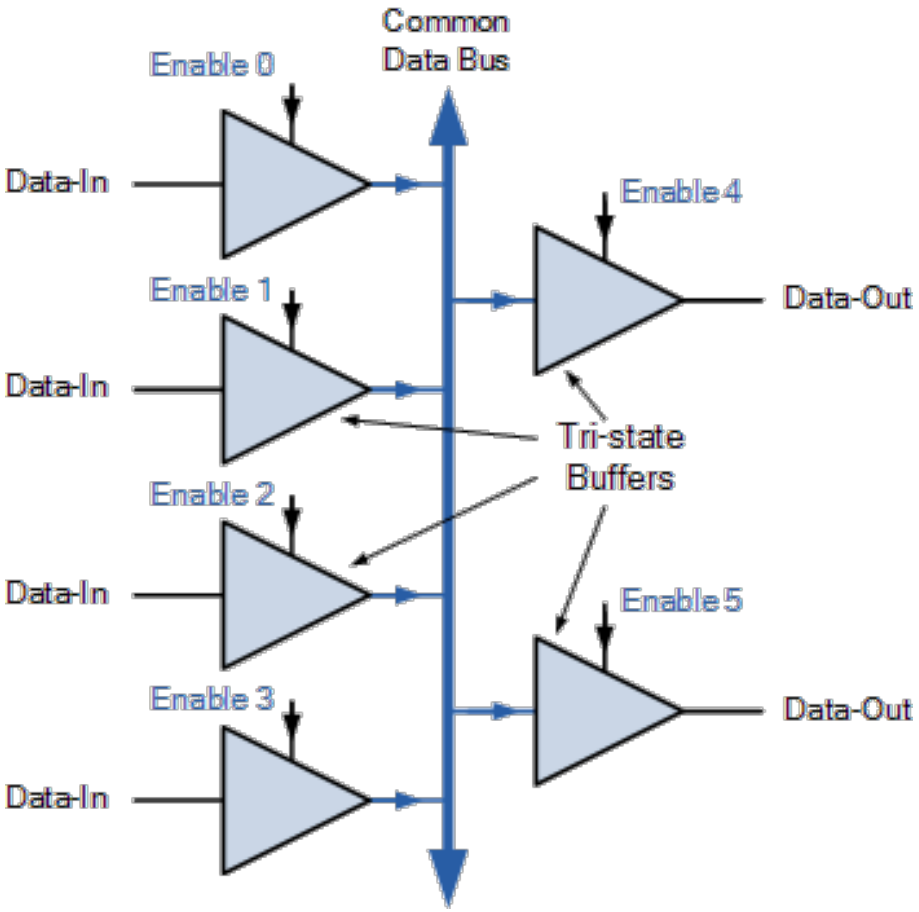
Symbol	Truth Table		
 Tri-state Buffer	Enable	IN	OUT
	0	0	Hi-Z
	0	1	Hi-Z
	1	0	0
	1	1	1
Read as Output = Input if Enable is equal to "1"			

No caso da figura, sua entrada *Enable* é do tipo *ativo "ALTO"*.
Note que o buffer não-inversor funciona tal qual uma porta inversora com exceção de que não inverte o nível lógico presente na sua entrada. O objetivo aqui seria apenas o de elevar o nível de corrente disponível na saída deste circuito ou isolar partes de um circuito colocando este componente no estado de alta-impedância (Hi-Z).

Obs.: Exitem buffers inversores (que neste caso se comportam da mesma forma que uma tradicional porta inversora).

Saídas Tri-state

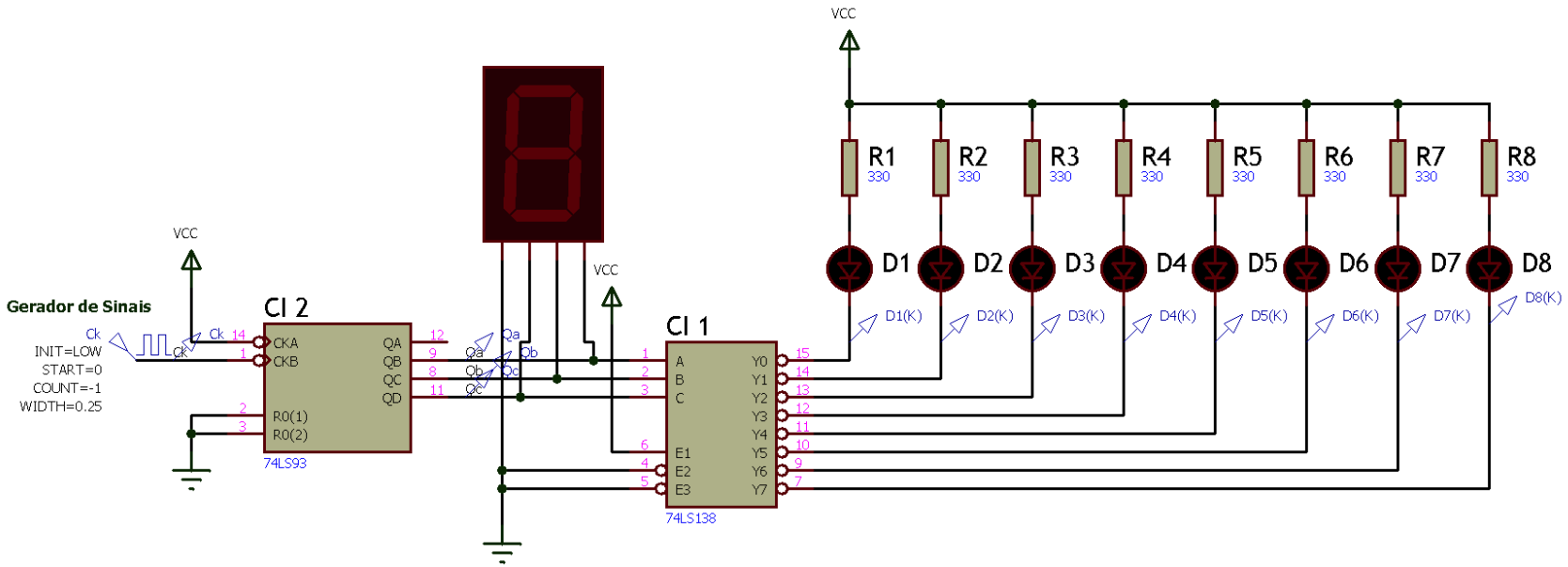
Exemplo de uso:



Note que neste caso, a linha *"Common Data Bus"* é compartilhada por 4 buffers. Para evitar uma indefinição de nível lógico se todas estas 4 portas resolvessem enviar ao mesmo tempo um nível lógico de saída neste único barramento, se chaveia uma porta de cada vez, que porta têm acesso ao barramento comum de dados, através da entrada *Enable* (de cada porta). Note que as saídas de um DEC podem ser usadas para gerar este sinal de controle (ativar ou não os *Enables* de forma conveniente).

- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

Circuito exemplo:



Vídeo no Youtube:

exemplo_dec_3_para_8_1_hz.mov, 507 KB

`http://youtu.be/ZnH7KCXbZhc` (Abr/2013)

DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**:

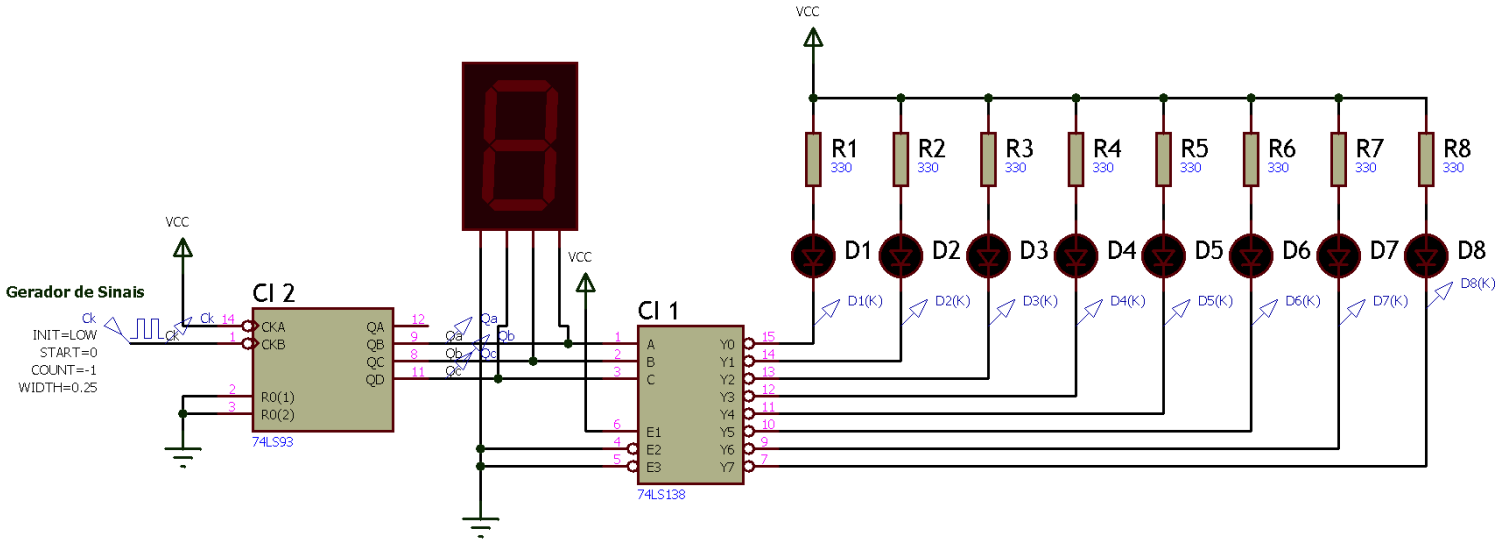
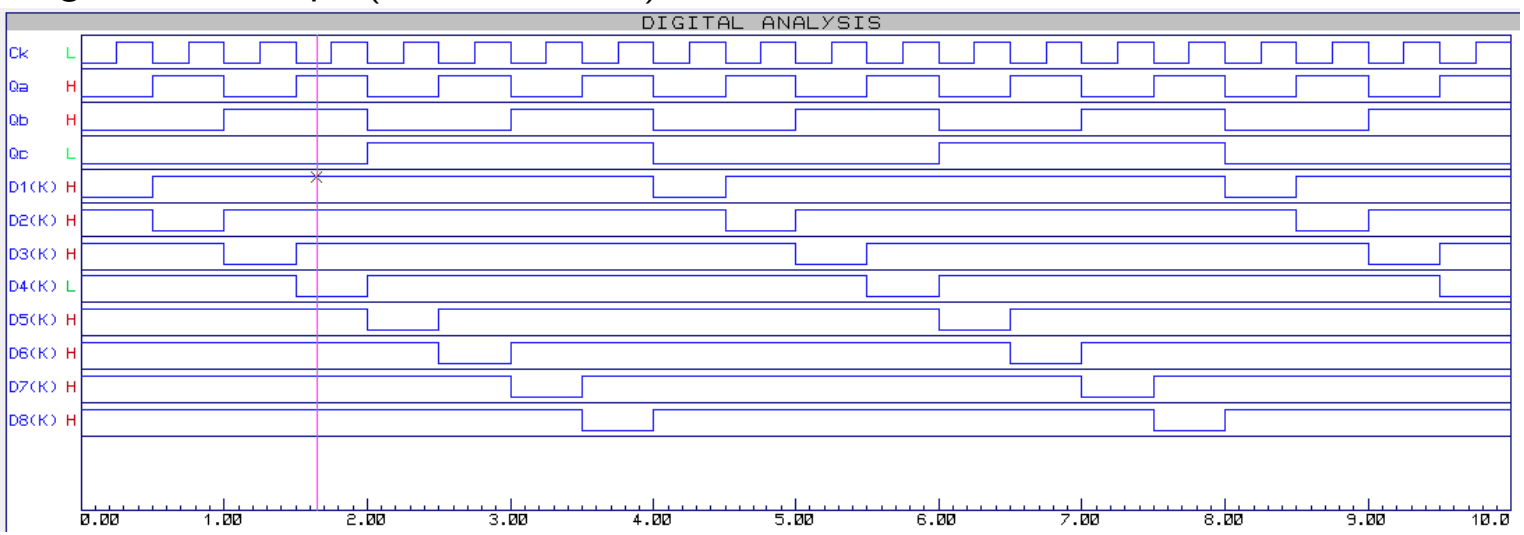
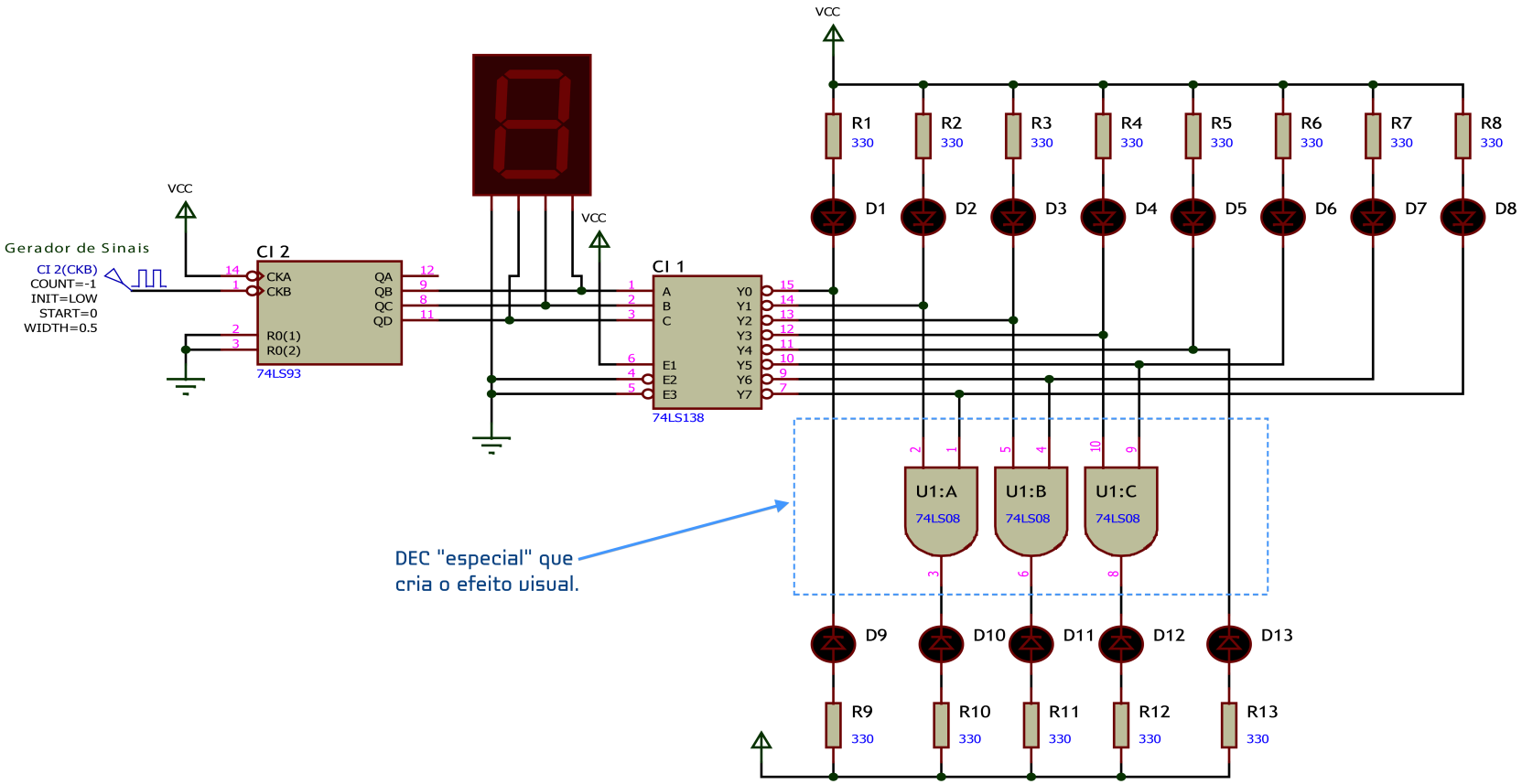


Diagrama no Tempo (formas de onda):



- DEC (Comercial) de 3 para 8 linhas de saída: CI 74**138**

Circuito exemplo de Efeito Visual:



Vídeo no Youtube:

exe_app_dec_3_para_8_full.mov, 1,9 MB

`http://youtu.be/YF7aPcTkwSQ` (Abr/2013)

Decodificadores

- DEC (Comercial) de 4 para 10 linhas de saída: CI 74**42**

Simbolo IEEE:

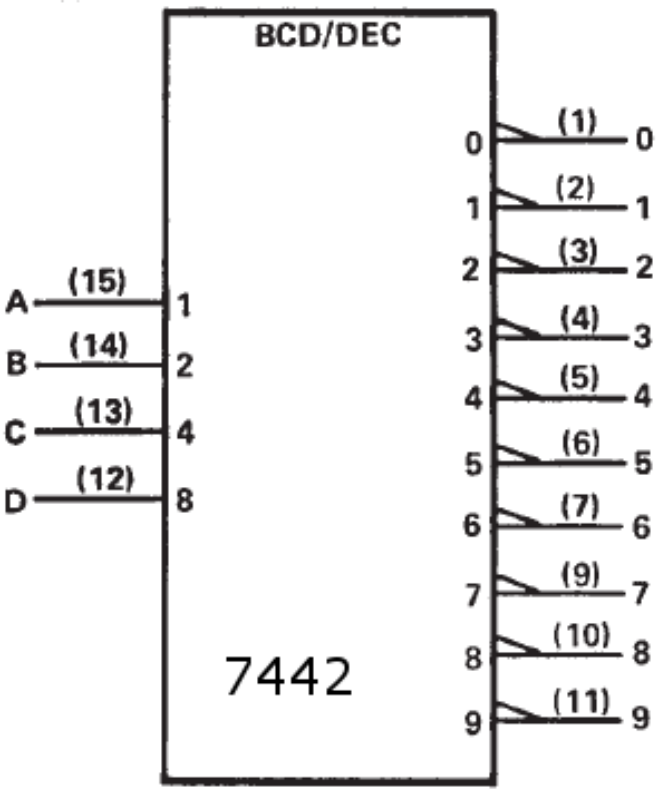
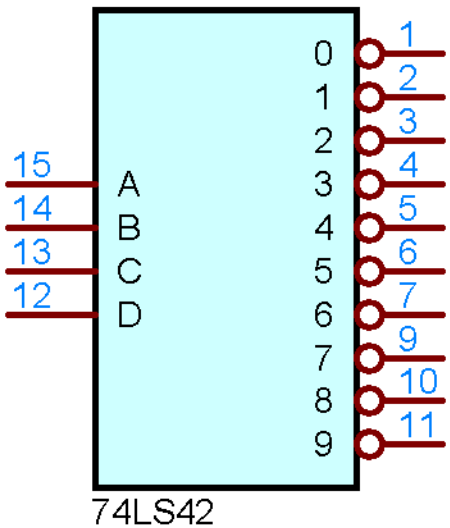


Tabela Verdade:

NO.	BCD INPUT				DECIMAL OUTPUT									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

DEC – Exemplo de Operação

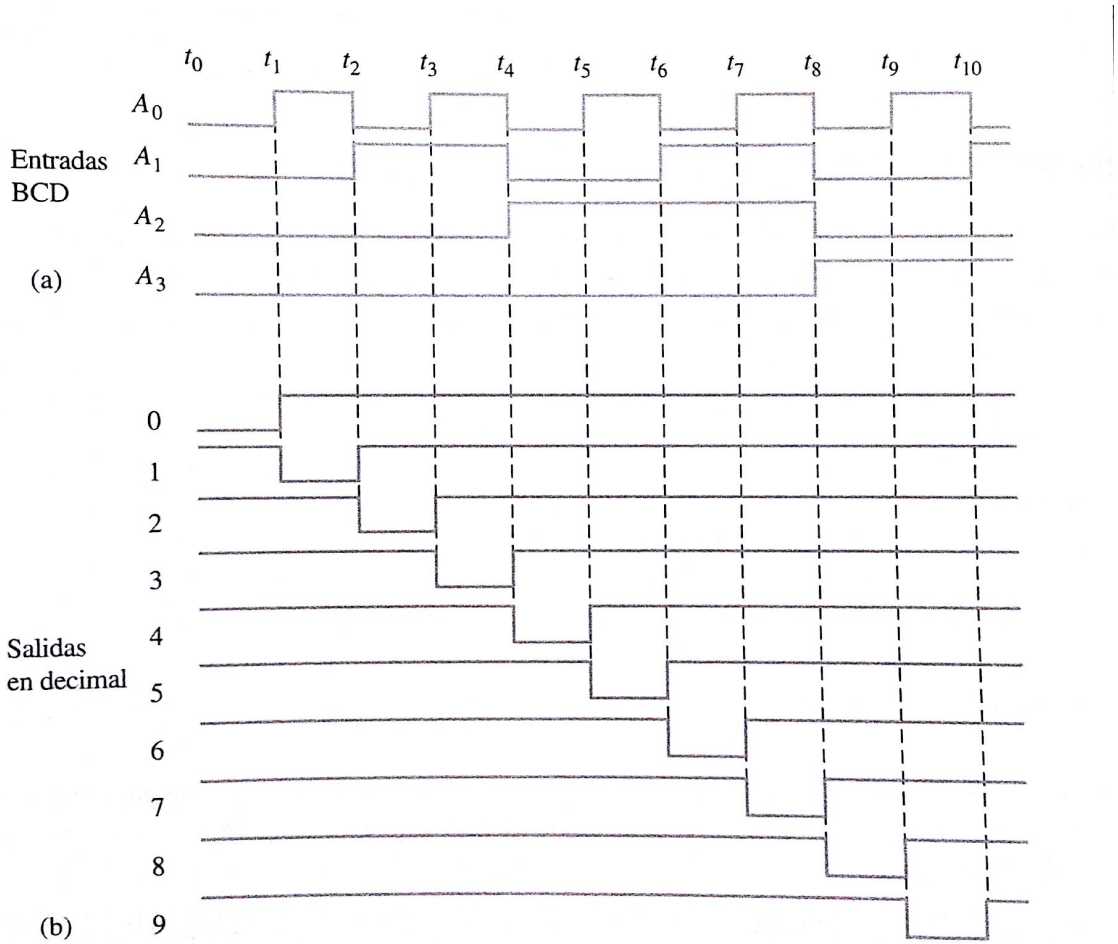
DEC 4/10: CI 7442:



Obs.: Símbolo extraído do PROTEUS.

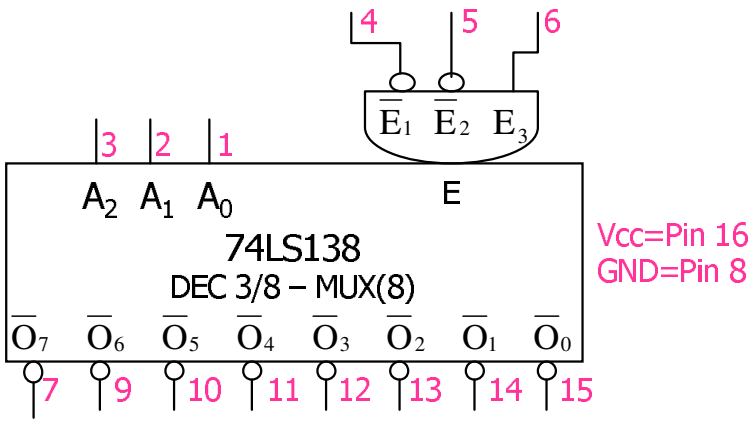
(1) Notar que a entrada A corresponde à 2^0 (bit LSB) de entrada.

(2) As saídas são ATIVO BAIXO.



DEC – Exercício

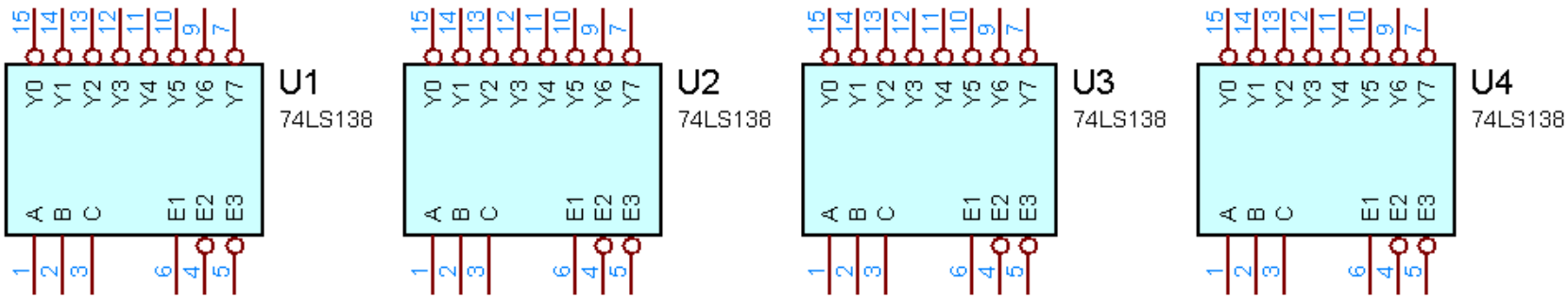
- Projete o circuito de um Decodificador de 5 para 32 linhas baseado no CI 74LS138 (DEC 3/8).
  Dado:



DEC – Exercício

- Projete o circuito de um Decodificador de 5 para 32 linhas baseado no CI 74LS138 (DEC 3/8).

Resolvendo..

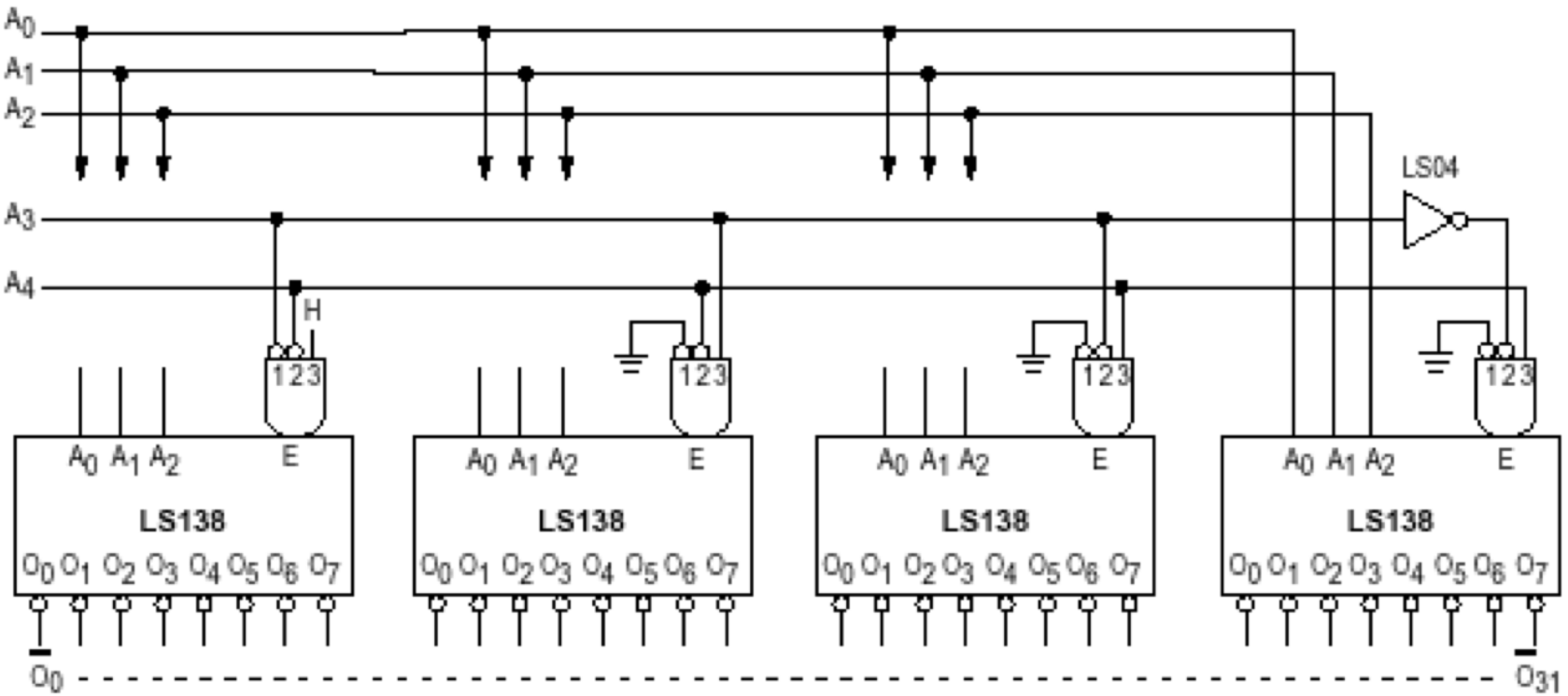


.

DEC – Exercício

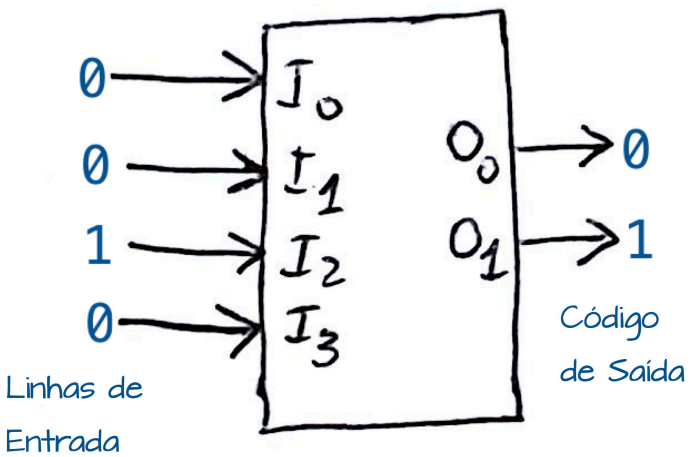
- Projete o circuito de um Decodificador de 5 para 32 linhas baseado no CI 74LS138 (DEC 3/8).

Solução:



CODECs ou Codificadores ou Encoders

Codificadores



Definição:

- Realiza a função "inversa" do DEC;
- Ou seja, transforma em código binário, a linha que está ativa na sua entrada;
- São: 2^n linhas de entrada $\longrightarrow$ n-linhas de saída.

Codificadores

- Codificador "simples":

Simbolo:

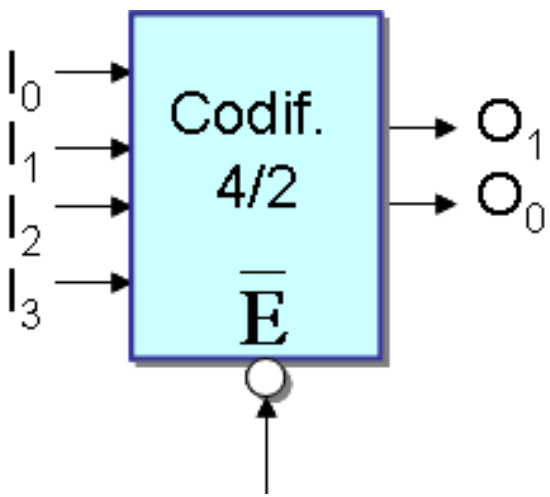


Tabela Verdade:

Entradas				Saídas	
I_0	I_1	I_2	I_3	O_1	O_0
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Equações:

$$\begin{aligned} O_0 &= I_1 + I_3 \\ O_1 &= I_2 + I_3 \end{aligned}$$

Repare que a entrada I_0 não é utilizada!

Codificadores

- Codificador "simples":

Simbolo:

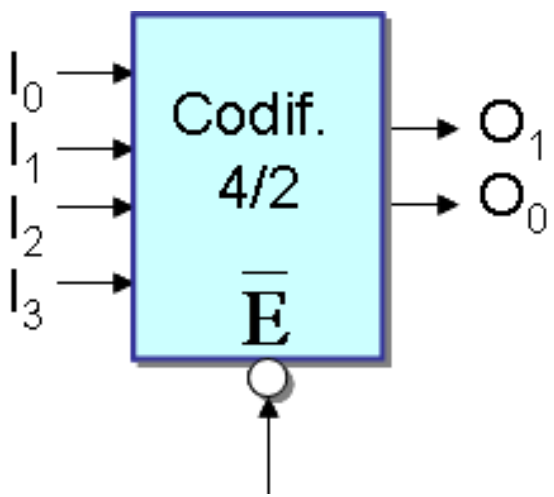


Tabela Verdade:

Entradas				Saídas	
I_0	I_1	I_2	I_3	O_1	O_0
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Equações:
$$\begin{aligned} O_0 &= I_1 + I_3 \\ O_1 &= I_2 + I_3 \end{aligned}$$
$\leftarrow$ Problema $\Rightarrow$ ERRO!

Onde está o erro?
– O que acontece quando mais de 2 entradas forem ativadas?

Codificadores

- Codificador "simples":

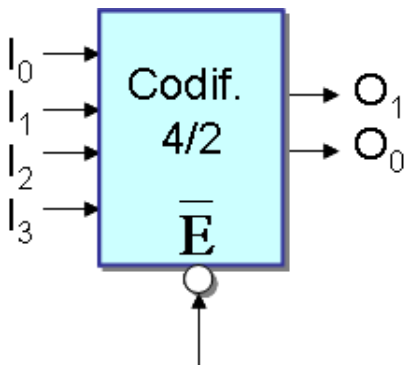
Equações: $\begin{array}{lcl} O_0 & = & I_1 + I_3 \\ O_1 & = & I_2 + I_3 \end{array}$ ← Problema ⇒ ERRO!

O que acontece quando mais de 2 entradas foram ativadas?

Verificando...

Suponha 2 casos:

Simbolo:



1) Linhas 1 e 3 ativas →

$$\begin{array}{lcllcll} O_0 & = & I_1 + I_3 & = & 1 + 1 & = & 1 \\ O_1 & = & I_2 + I_3 & = & 0 + 1 & = & 1 \end{array} \Rightarrow 3_{(10)} \quad \checkmark$$

Corresponde ao código 3 de saída!

2) Linhas 1 e 2 ativas →

$$\begin{array}{lcllcll} O_0 & = & I_1 + I_3 & = & 1 + 0 & = & 1 \\ O_1 & = & I_2 + I_3 & = & 1 + 0 & = & 1 \end{array} \Rightarrow 3_{(10)} \quad \frown$$

⇒ Também corresponde ao código 3 de saída!

(Note que a linha 3 nem está ativa!)

Codificadores

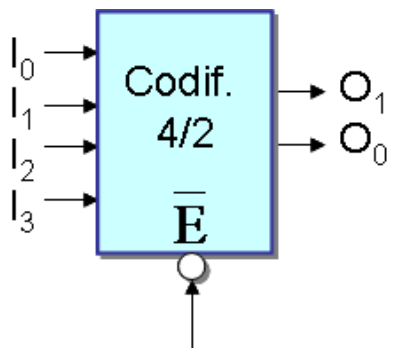
- Codificador **com Prioridade**:

Equações:
$$\begin{array}{lcl} O_0 & = & I_1 + I_3 \\ O_1 & = & I_2 + I_3 \end{array} \leftarrow \text{Problema!}$$

Correção:

Prever o caso de mais de uma entrada ativa em cada instante de tempo. Neste caso, o código de saída corresponde a maior linha de entrada ativa naquele instante de tempo. A isto se dá o nome de "prioridade". Note que a equação acima muda completamente. Necessitamos de uma nova tabela verdade: Tabela verdade "corrigida":

Simbolo:



Entradas				Saídas	
I_0	I_1	I_2	I_3	O_1	O_0
0	0	0	0	0	0
1	0	0	0	0	0
X	1	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1

Codificador com Prioridade

Codificador SEM prioridade:

Entradas				Saídas	
I_0	I_1	I_2	I_3	O_1	O_0
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

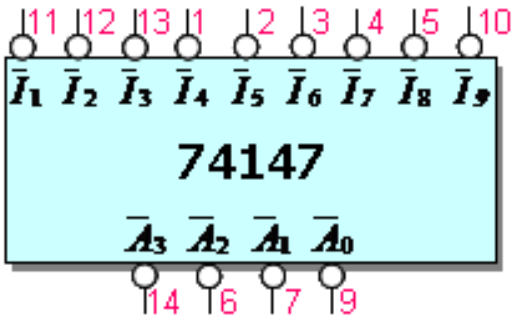
Codificador COM prioridade:

Entradas				Saídas	
I_0	I_1	I_2	I_3	O_1	O_0
0	0	0	0	0	0
1	0	0	0	0	0
X	1	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1

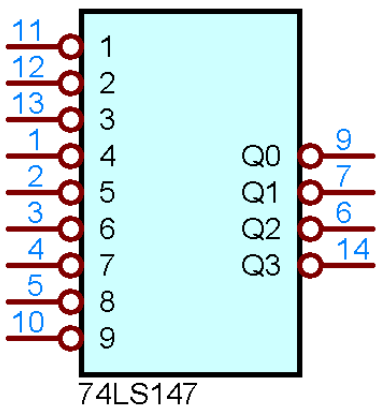
Codificadores

- Codificador com Prioridade (Comercial): CI 74147 (*10 to 4 line priority encoder*):

Simbolo:



Símbolo Proteus:



5(10)=0101(2)

1010(2)

Entradas									Saídas			
$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	$\bar{I}_8$	$\bar{I}_9$	$\bar{A}_3$	$\bar{A}_2$	$\bar{A}_1$	$\bar{A}_0$
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	**L**	L	H	H	L
X	X	X	X	X	X	X	**L**	H	L	H	H	H
X	X	X	X	X	X	**L**	H	H	H	L	L	L
X	X	X	X	X	**L**	H	H	H	H	L	L	H
X	X	X	X	**L**	H	H	H	H	H	L	H	L
X	X	X	**L**	H	H	H	H	H	H	L	H	H
X	X	**L**	H	H	H	H	H	H	H	H	L	L
X	**L**	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

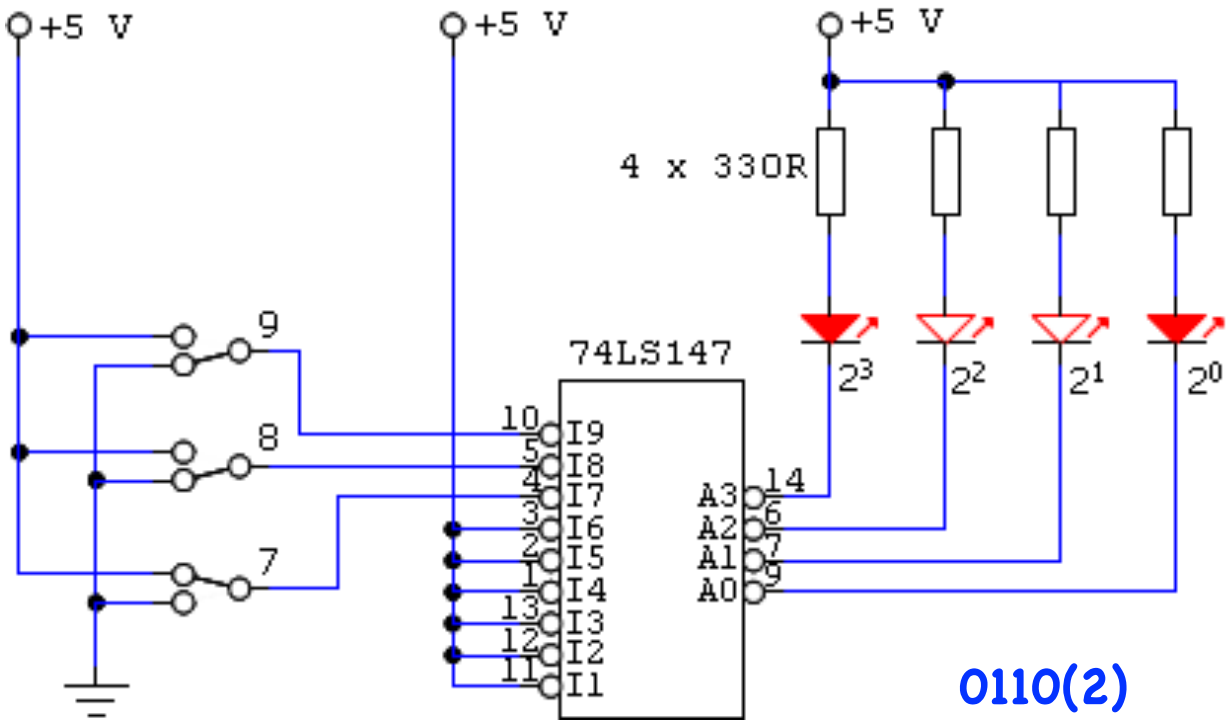
Tabela verdade do 74147.

Codificadores

- Codificador com Prioridade (Comercial): CI 74147 (*10 to 4 line priority encoder*):

9(10)=1001(2)

Exemplo de aplicação:

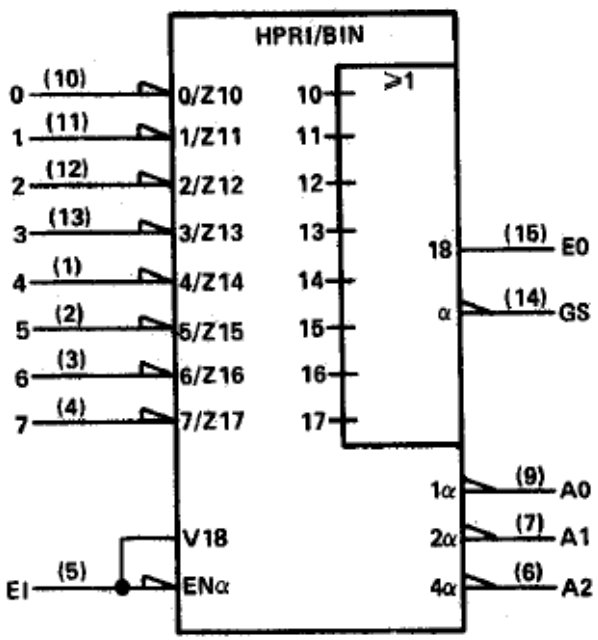


0110(2)

Codificadores

- Codificador com Prioridade (Comercial): IC 74**148** (*8-line to 3-line priority encoder*):

Símbolo:



Padrão IEEE 91-1984.

2(10)=010(2)
101(2)

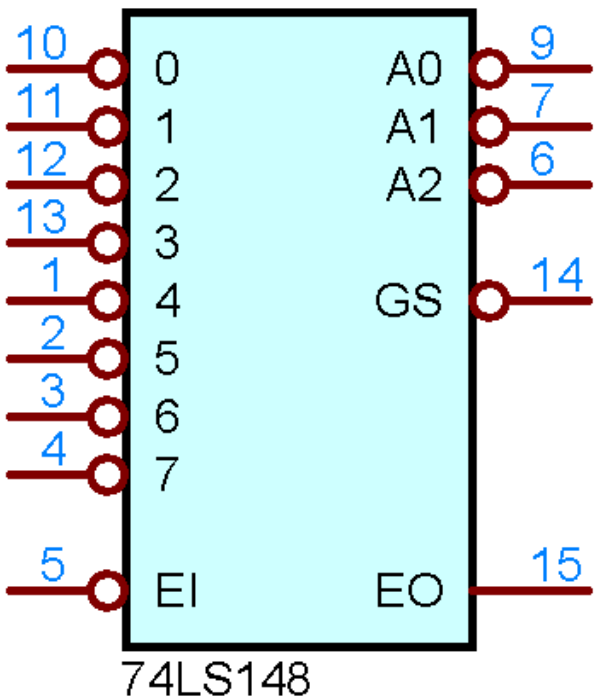
Entradas									Saídas				
$\overline{EI}$	$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	$\bar{A}_2$	$\bar{A}_1$	$\bar{A}_0$	$\overline{GS}$	E0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Tabela verdade do 74148.

Codificadores

- Codificador com Prioridade (Comercial): IC 74**148** (*8-line to 3-line priority encoder*):

Símbolo:



Símbolo no Proteus.

Entradas									Saídas				
$\overline{EI}$	$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	$\bar{A}_2$	$\bar{A}_1$	$\bar{A}_0$	$\overline{GS}$	E0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	**L**	L	L	L	L	H
L	X	X	X	X	X	X	**L**	H	L	L	H	L	H
L	X	X	X	X	X	**L**	H	H	L	H	L	L	H
L	X	X	X	X	**L**	H	H	H	L	H	H	L	H
L	X	X	X	**L**	H	H	H	H	H	L	L	L	H
L	X	X	**L**	H	H	H	H	H	H	L	H	L	H
L	X	**L**	H	H	H	H	H	H	H	H	L	L	H
L	**L**	H	H	H	H	H	H	H	H	H	H	L	H

Tabela verdade do 74148.

Codificadores

- Codificador com Prioridade (Comercial): IC 74**148** (*8-line to 3-line priority encoder*):

Entradas									Saídas				
$\overline{EI}$	$\bar{I}_0$	$\bar{I}_1$	$\bar{I}_2$	$\bar{I}_3$	$\bar{I}_4$	$\bar{I}_5$	$\bar{I}_6$	$\bar{I}_7$	$\bar{A}_2$	$\bar{A}_1$	$\bar{A}_0$	$\overline{GS}$	E0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	**L**	L	L	L	L	H
L	X	X	X	X	X	X	**L**	H	L	L	H	L	H
L	X	X	X	X	X	**L**	H	H	L	H	L	L	H
L	X	X	X	X	**L**	H	H	H	L	H	H	L	H
L	X	X	X	**L**	H	H	H	H	H	L	L	L	H
L	X	X	**L**	H	H	H	H	H	H	L	H	L	H
L	X	**L**	H	H	H	H	H	H	H	H	L	L	H
L	**L**	H	H	H	H	H	H	H	H	H	H	L	H

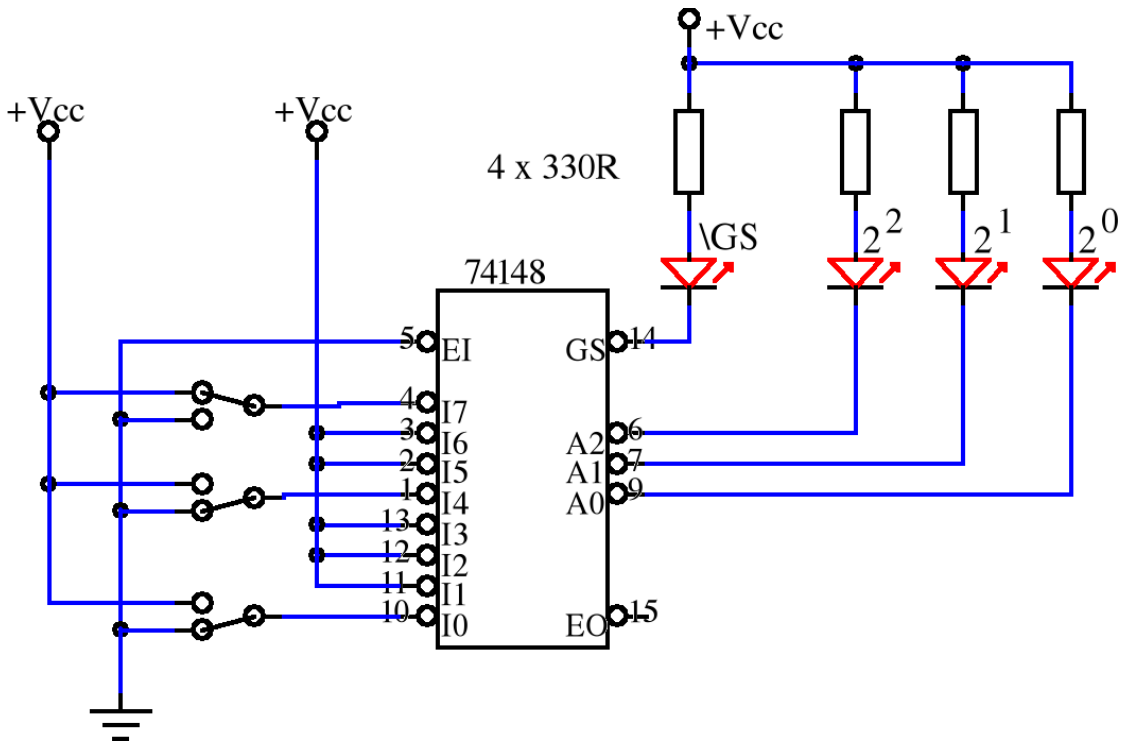
Tabela verdade do 74148.

Obs.: Repare que tanto suas linhas de entrada quanto suas saídas são ATIVO BAIXO. Uma saída nula ($\bar{A}_2\bar{A}_1\bar{A}_0 = 111_{(2)}$) aparece quando todas as suas linhas de entrada estão em nivel lógico ALTO (desativadas) e neste caso, a saída $\overline{GS}$ (*Group Signal*) permanece em nivel lógico ALTO, o que significa que nenhuma linha de entrada foi ativada. Se alguma linha de entrada foi ativada, então a saída $\overline{GS}$ muda para nivel lógico BAIXO indicando precisamente a ativação de alguma linha de entrada (mesmo que seja a linha $\bar{I}_0$).

Codificadores: Exemplo de Uso

- Codificador com Prioridade (Comercial): IC 74**148** (*8-line to 3-line priority encoder*):

Exemplo de aplicação:



Encoder Absoluto: de Gray $\longrightarrow$ Binário

- Código Gray: Usado em sensores de posição angular, para gerar posição angular de um eixo;
- Faz parte de sensores do tipo "absoluto": conhecidos também como "encoders absolutos";
- Fornece a indicação da posição angular em qualquer instante de tempo (não exige inicialização como no caso dos encoders relativos).
- Note que entre uma "raia" e outra (ou entre uma sequencia do Gray e vizinhos), somente 1 bit é modificado! $\Rightarrow$ Minimiza erros de leitura (hardware).

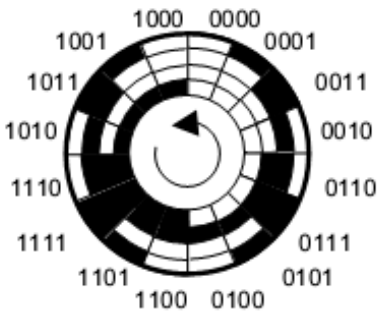
Disco Gray de 10-bits.

Resolução de: $= \frac{360^o}{2^{10}} = \frac{360^o}{1024} = 0,3515625^o$

Encoder Absoluto:: de Gray $\longrightarrow$ Binário

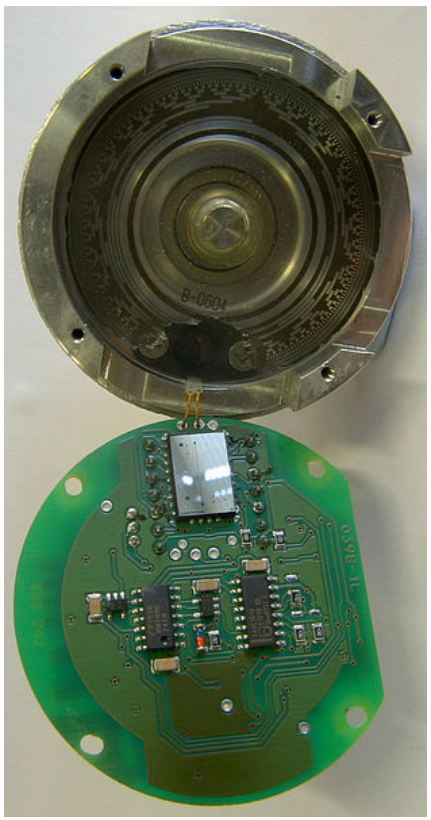
- Código Gray: Usado em sensores de posição angular, para gerar posição angular de um eixo;
- Faz parte de sensores do tipo "absoluto": conhecidos também como "encoders absolutos";
- Fornece a indicação da posição angular em qualquer instante de tempo (não exige inicialização como no caso dos encoders relativos).
- Note que entre uma "raia" e outra (ou entre uma sequencia do Gray e vizinhos), somente 1 bit é modificado! $\Rightarrow$ Minimiza erros de leitura (hardware).

Exemplo de Encoder absoluto de 3-bits.



Exemplo de Encoder de 4-bits.
Observe a sequencia das raias!

Encoder Absoluto:: de Gray $\longrightarrow$ Binário

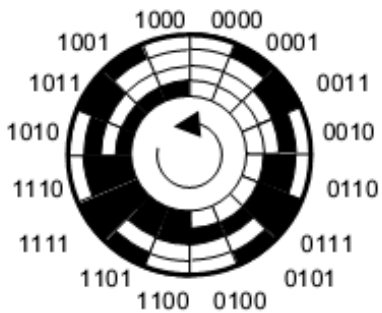


Encoder com código Gray de 13-bits.

Resolução de: $= \frac{360^o}{2^{13}} = \frac{360^o}{8192} = 0,0439453125^o$

3-bit rotary encoder

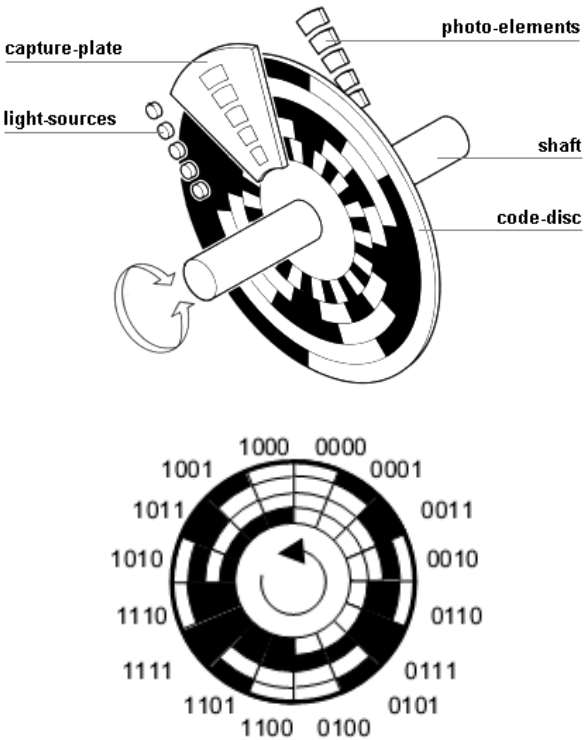
Exemplo de Encoder absoluto de 3-bits.



Gray code

Exemplo de Encoder de 4-bits.
Observe a sequencia das raias!

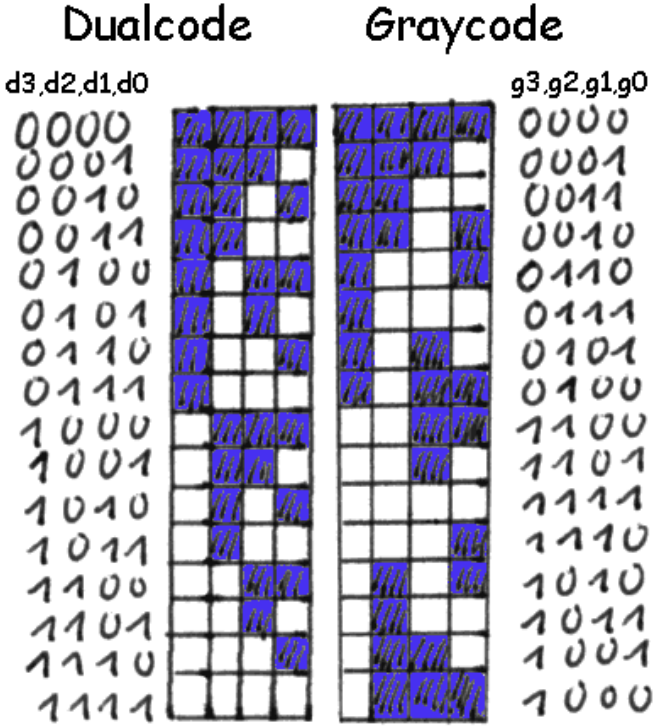
Encoder Absoluto:: de Gray $\longrightarrow$ Binário



Gray code

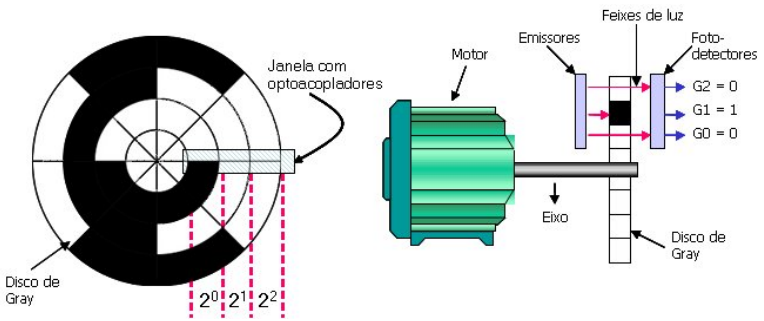
Exemplo de Encoder de 4-bits.
Observe a sequencia das raias!

Tabela Verdade com Sequencias:



Encoder Absoluto:: de Gray $\longrightarrow$ Binário

– Projete um circuito para converter de gray de 3-bit para binário.



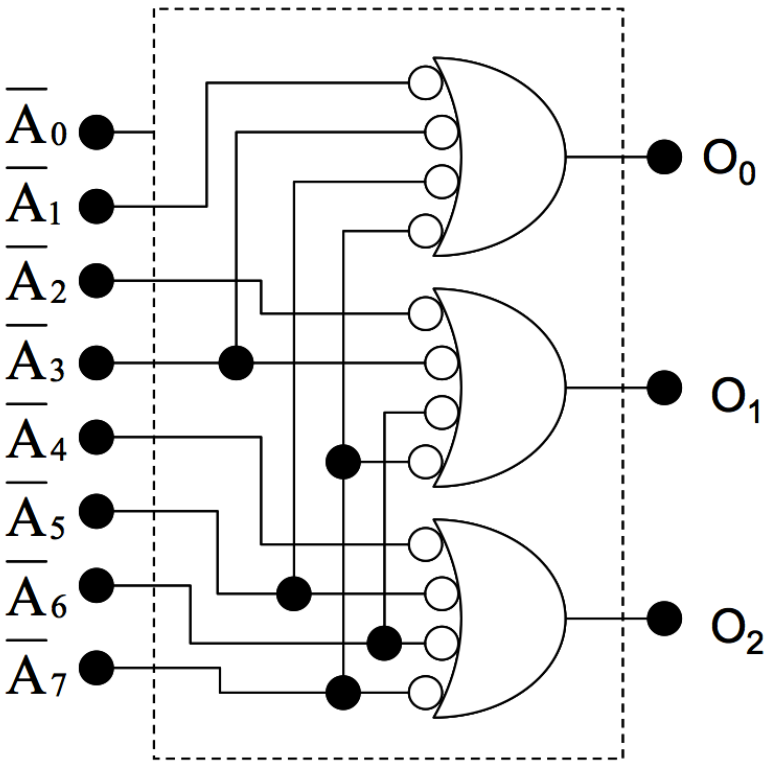
Exemplo de Encoder de 3-bits.

Tabela Verdade:

Setor	Código Gray			Ângulo	Código Binário
	G_2	G_1	G_0		
1	0	0	0	0^o to 45^o	000
2	0	0	1	45^o à 90^o	001
3	0	1	1	90^o à 135^o	010
4	0	1	0	135^o à 180^o	011
5	1	1	0	180^o à 225^o	100
6	1	1	1	225^o à 270^o	101
7	1	0	1	270^o à 315^o	110
8	1	0	0	315^o à 360^o	111

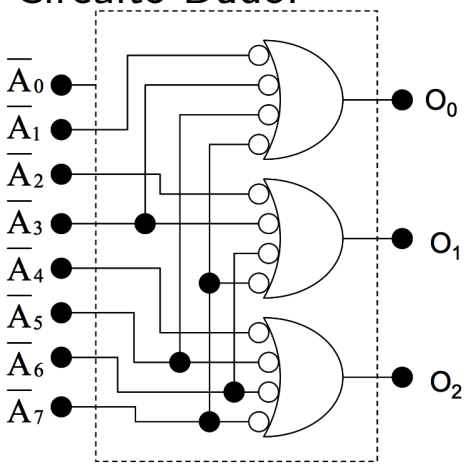
Codificadores – Exercício

Exemplo: Deduza a função desempenhada pelo circuito abaixo:



Codificadores – Exercício

Circuito Dado:



Solução → Equações:

$$O_2 = \overline{\overline{A_4} + \overline{A_5} + \overline{A_6} + \overline{A_7}} = A_4 + A_5 + A_6 + A_7$$

$$O_1 = \overline{\overline{A_2} + \overline{A_3} + \overline{A_5} + \overline{A_6}} = A_2 + A_3 + A_6 + A_7$$

$$O_0 = \overline{\overline{A_1} + \overline{A_3} + \overline{A_5} + \overline{A_7}} = A_1 + A_3 + A_5 + A_7$$

O que resulta na seguinte tabela verdade...

Codificadores – Exercício

Equações:

$$O_2 = \overline{\overline{A_4} + \overline{A_5} + \overline{A_6} + \overline{A_7}} = A_4 + A_5 + A_6 + A_7 \quad =1$$

$$O_1 = \overline{\overline{A_2} + \overline{A_3} + \overline{A_5} + \overline{A_6}} = A_2 + A_3 + A_6 + A_7 \quad =1 \Rightarrow 7_{(10)}$$

$$O_0 = \overline{\overline{A_1} + \overline{A_3} + \overline{A_5} + \overline{A_7}} = A_1 + A_3 + A_5 + A_7 \quad =1$$

Circuito Dado:

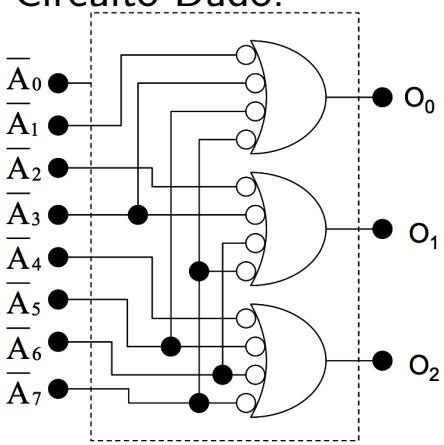


Tabela verdade resultante:

Entradas								Saídas		
$\overline{A_0}$	$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

Note: saídas adequadas APENAS se uma entrada estiver ativa de cada vez!

Exercício: O que acontece se as entradas $\overline{A_3}$ e $\overline{A_5}$ estiverem ativas ao mesmo tempo ?

Resp.: Forma o código de saída: $O_2 O_1 O_0 = 111_{(2)} = 7_{(10)}$!.

Codificadores – Problema

- Deduza o circuito interno de um codificador de prioridade para 4 linhas de entrada.

Tabela Verdade:

Entradas				Saídas		
I_0	I_1	I_2	I_3	G_S	O_1	O_0
0	0	0	0	0	0	0
1	0	0	0	1	0	0
X	1	0	0	1	0	1
X	X	1	0	1	1	0
X	X	X	1	1	1	1

Note que existe uma saída extra: G_S (*Group Signal*) que sinaliza quando alguma linha de entrada foi ativada.

MUXes ou Multiplexadores

Multiplexador

- Multiplexador: $\rightarrow$ Atua como uma chave seletora de datos de entrada

Suponha o caso de um MUX-2 (para 2 entradas de dados):

Símbolo:

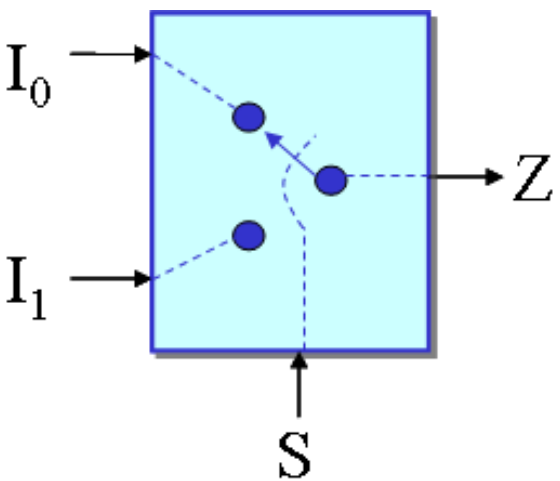


Tabela Funcional:

Entradas S	Saídas Z
0	I_0
1	I_1

Equação:

$$Z = I_0 \cdot \overline{S} + I_1 \cdot S$$

Multiplexador

- Multiplexador: $\rightarrow$ Atua como seletor de dados de entrada.

Suponha o caso de um MUX-4 (de 4 canais de entradas de dados):

Tabela Funcional:

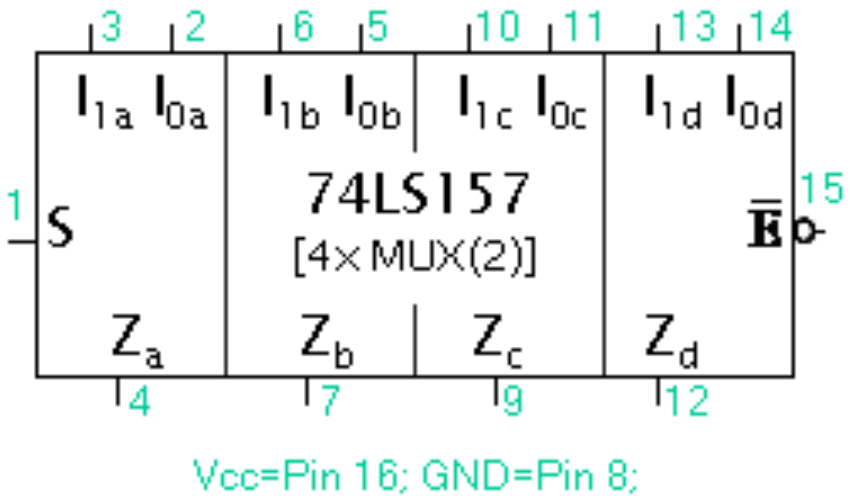
Entradas		Saídas
S_1	S_0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Equação:

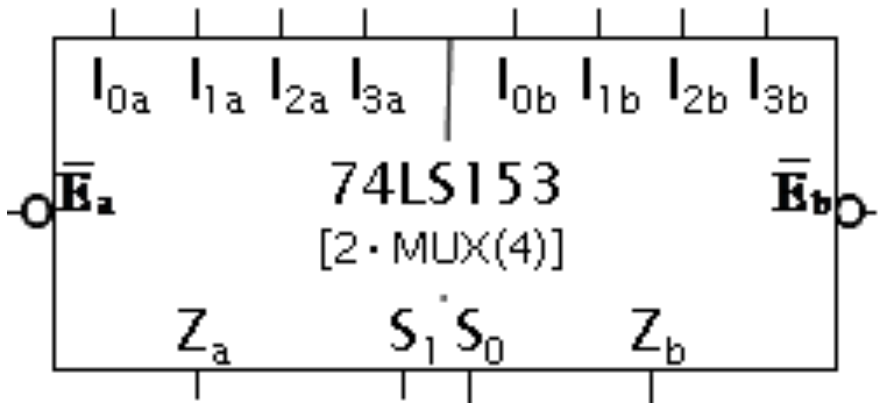
$$Z = I_0 \cdot \overline{S_1} \cdot \overline{S_0} + I_1 \cdot \overline{S_1} \cdot S_0 + I_2 \cdot S_1 \cdot \overline{S_0} + I_3 \cdot S_1 \cdot S_0$$

Multiplexadores comerciais

MUX-2: IC 74**157**

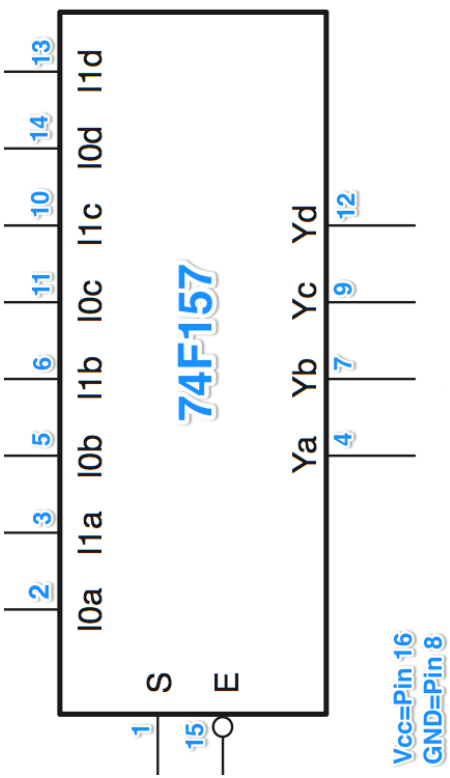


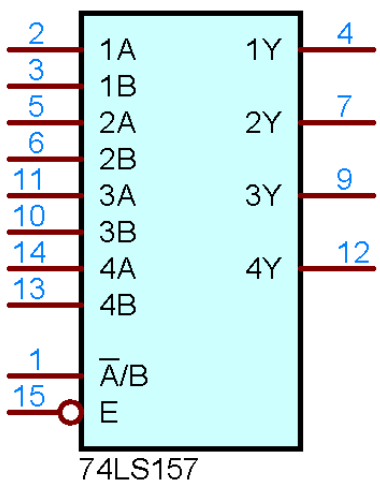
MUX-4: IC 74**153**



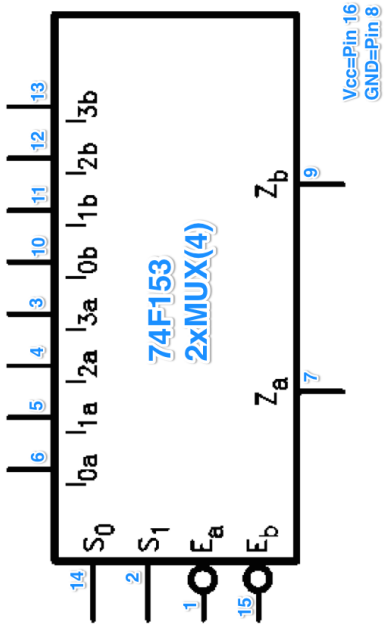
Multiplexadores comerciais

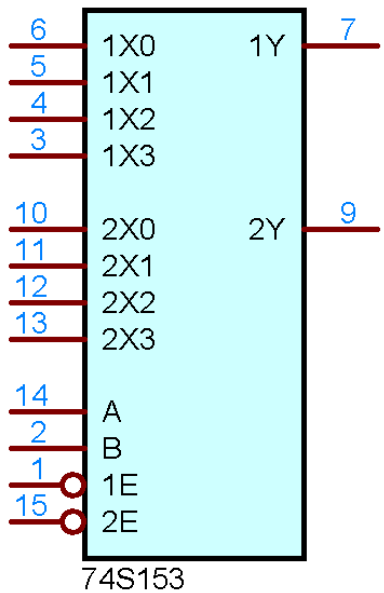
MUX-2: IC 74**157**





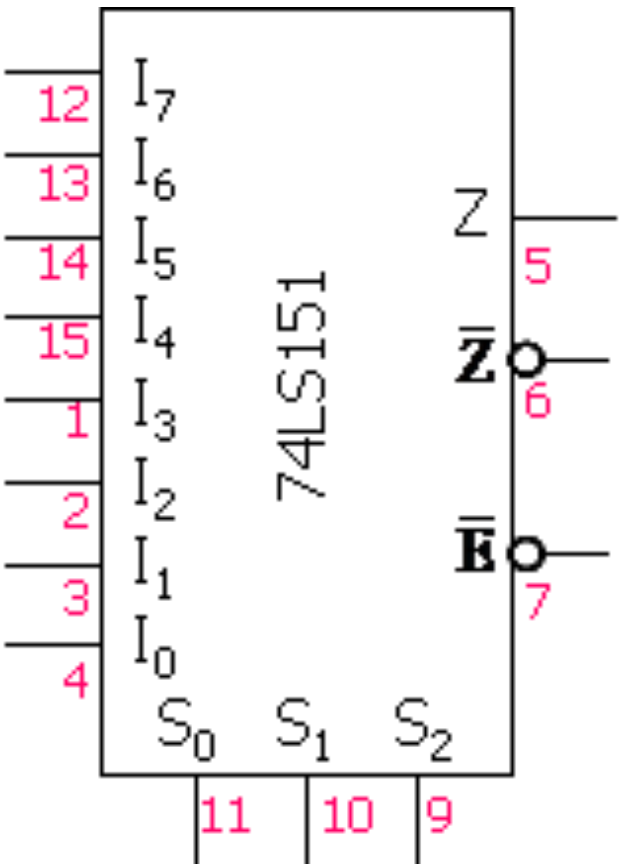
MUX-4: IC 74**153**

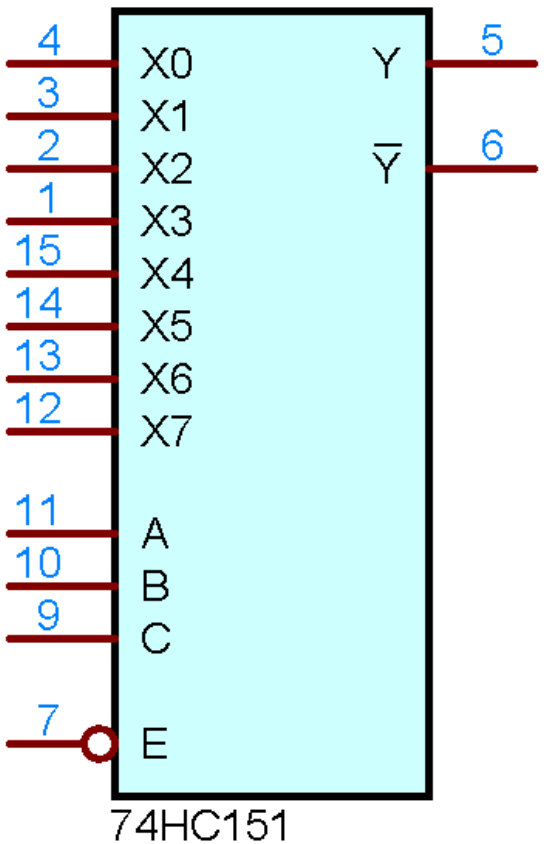




Multiplexadores comerciais

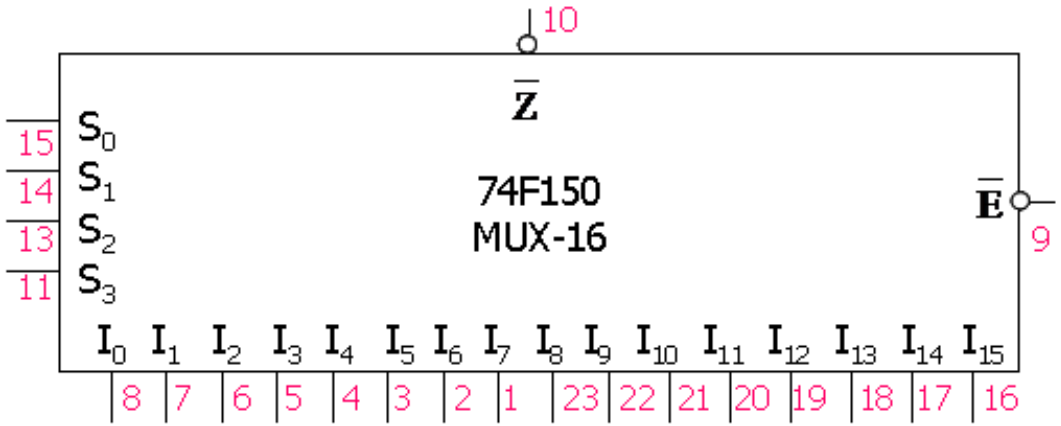
MUX(8) – IC 74**151**:

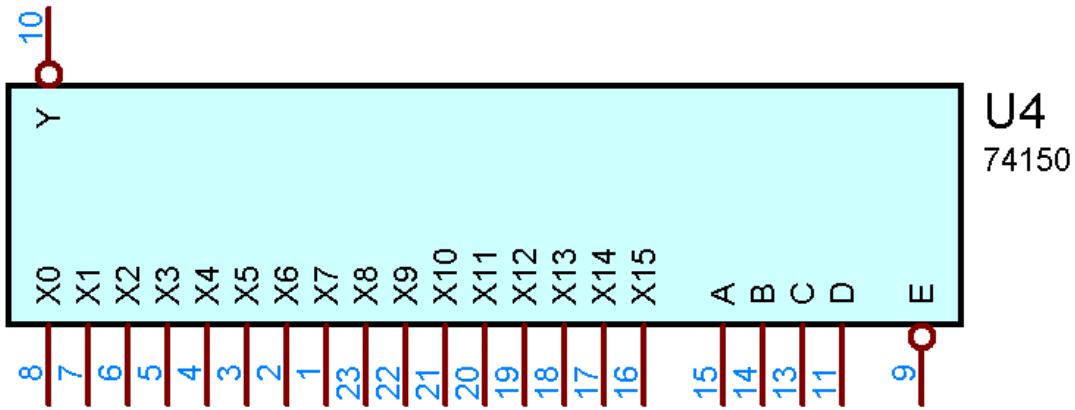




Multiplexadores comerciais

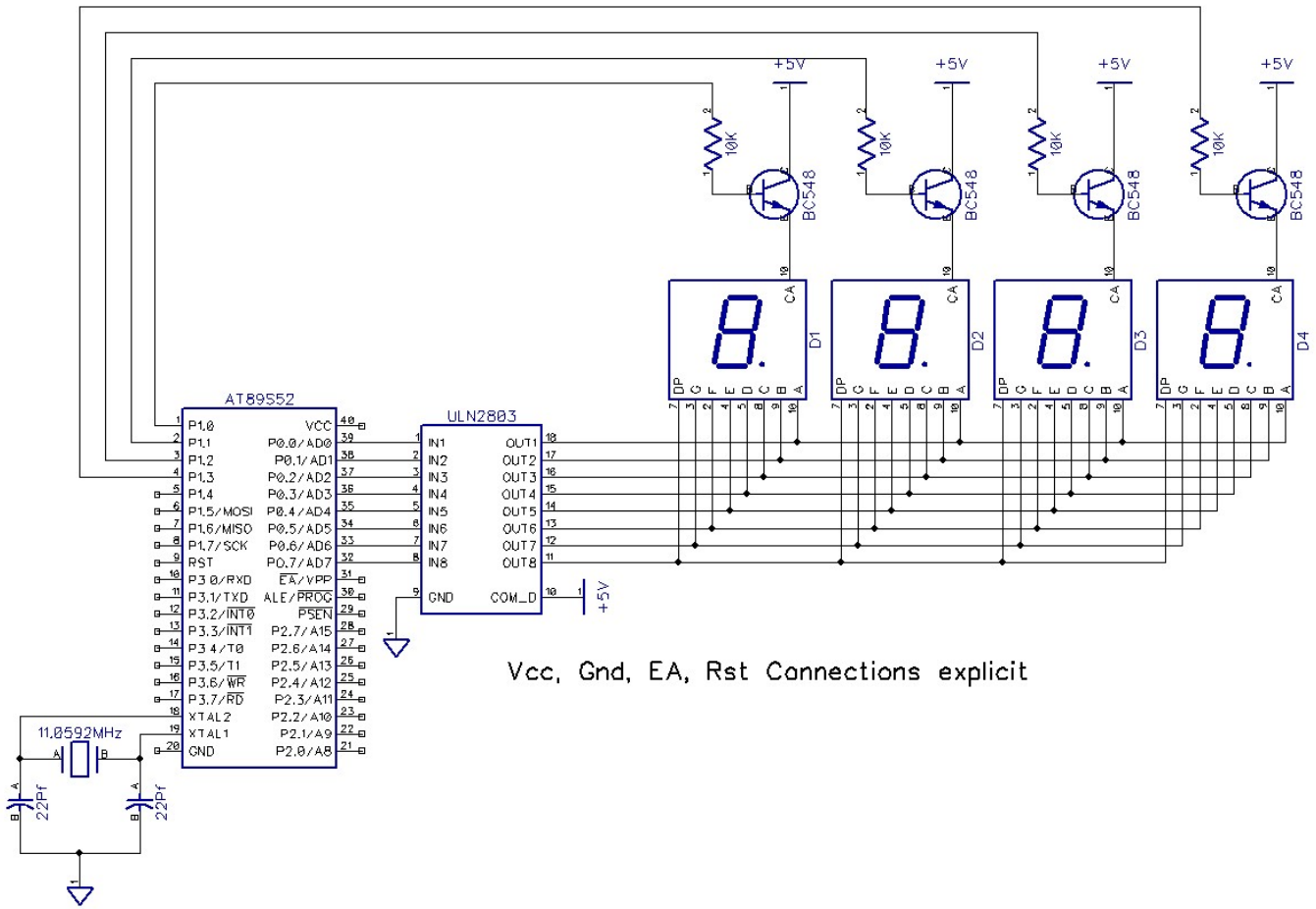
MUX(16) – IC 74**150**:





Filosofia da Multiplexação de Dados

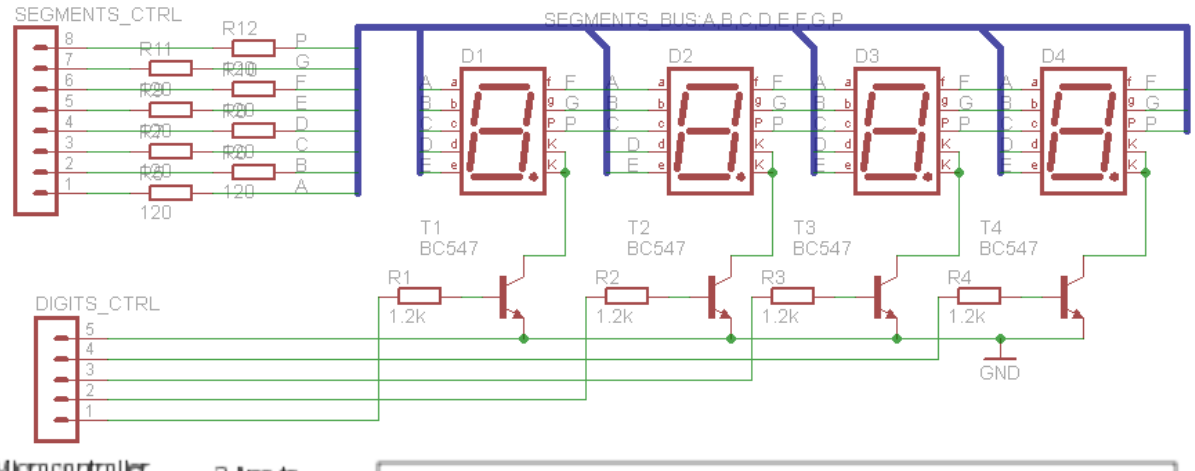
Exemplo de Multiplexação de 4 × Displays de 7-Segmentos:

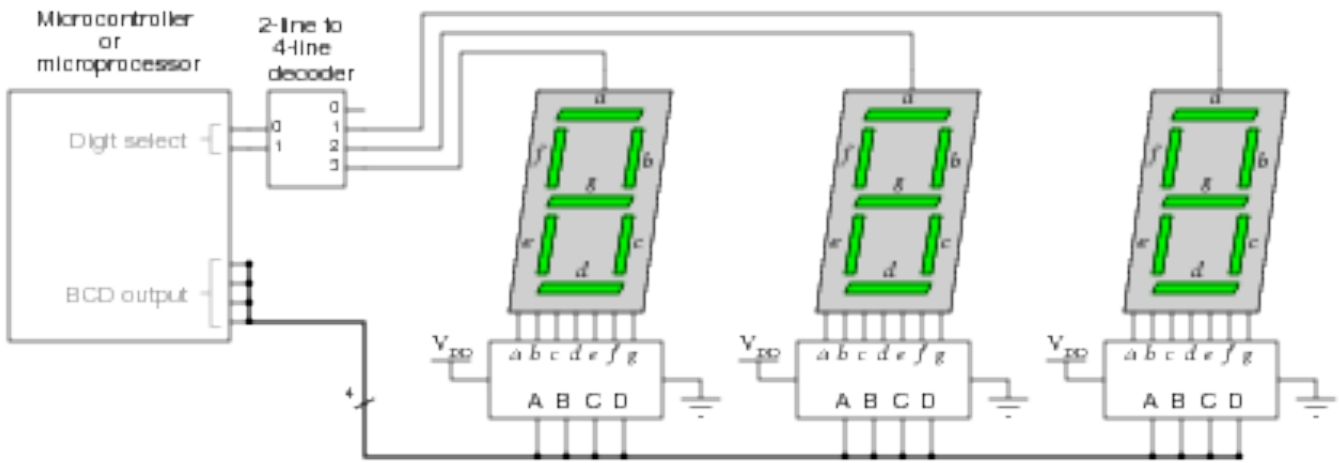


Vcc, Gnd, EA, Rst Connections explicit

Filosofia da Multiplexação de Dados

Detalhe de circuitos Multiplexando de 4 $\times$ Displays de 7-Segmentos:

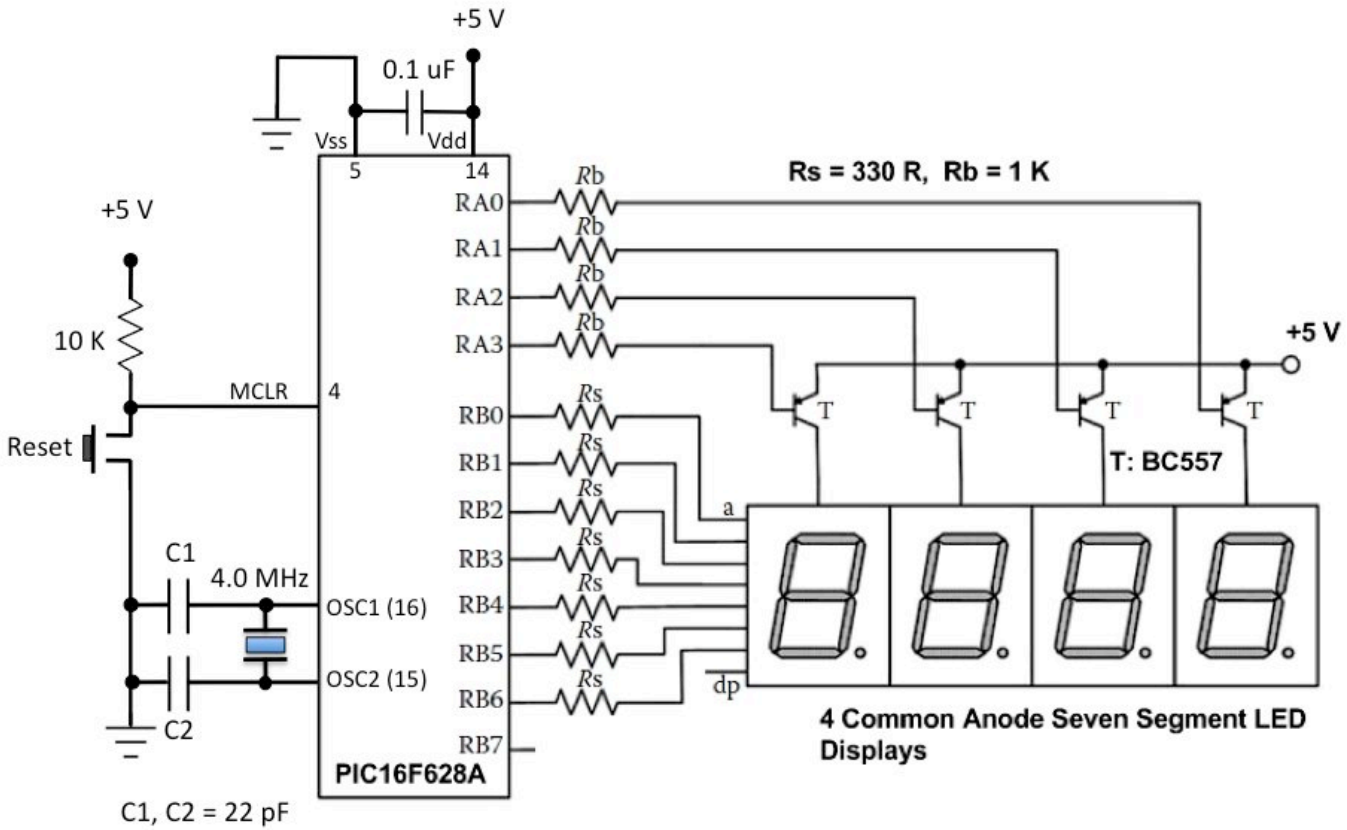




ou

Filosofia da Multiplexação de Dados

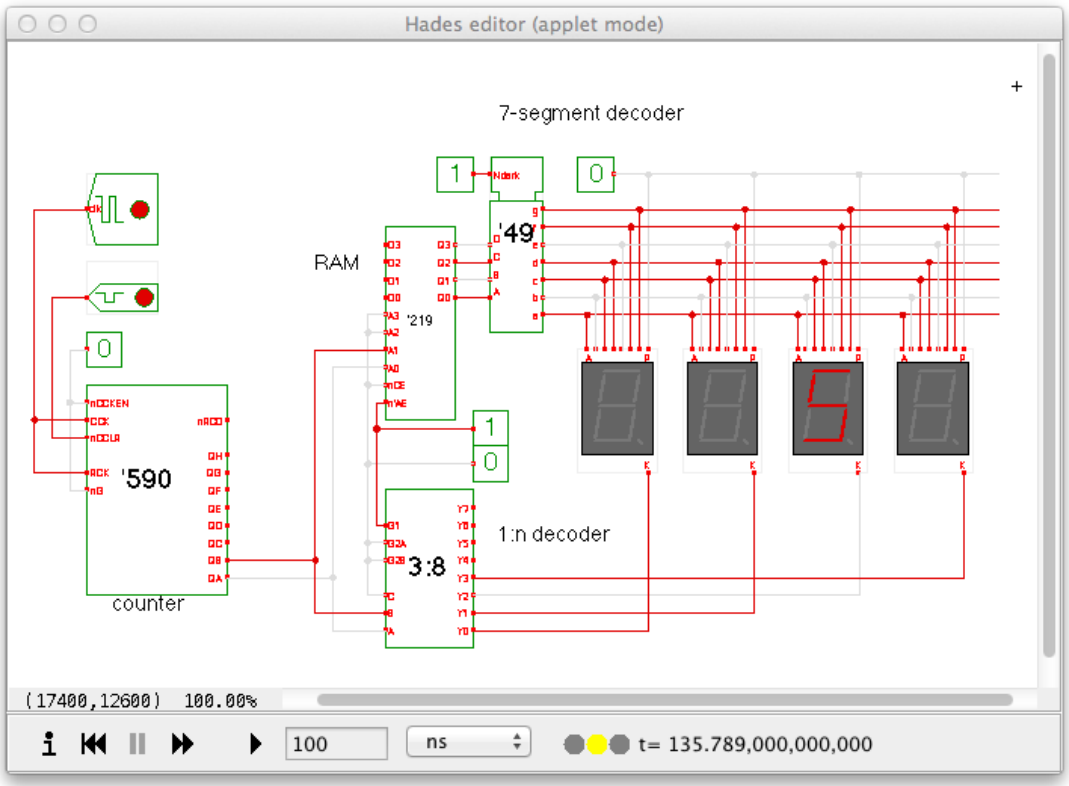
Outro exemplo de Multiplexação de Displays de 7-Segmentos usando outro μC:



Ref: "Lab 11: Multiplexing seven segment LED displays":
`http://embedded-lab.com/blog/?p=2086`
(Publicado em Mar 28th, 2011, Disponível em Abr 2013)

Filosofia da Multiplexação de Dados

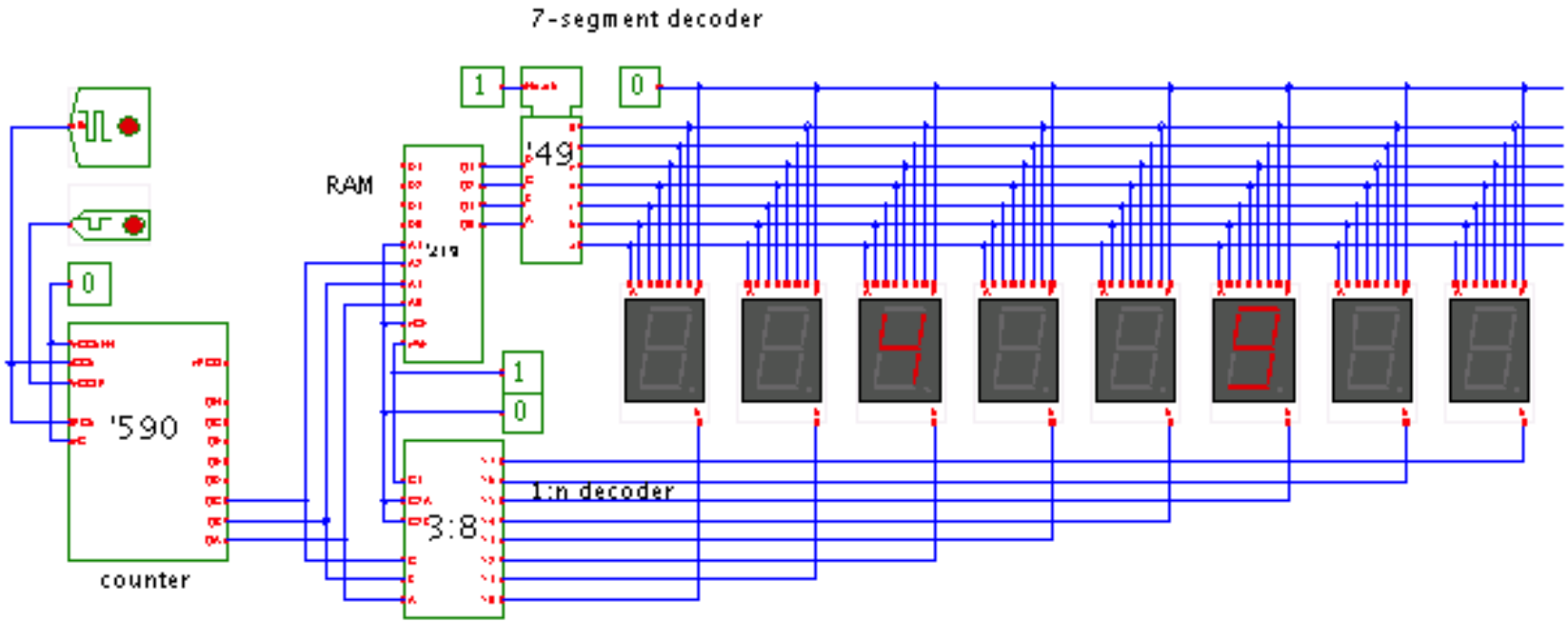
Exemplo de Multiplexação de 4 × Displays de 7-Segmentos:



Applet Java disponível em: `http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/45-misc/50-displays/multiplexed-display.html` (Abr 2013)

Multiplexadores – Exemplos de Uso

Exemplo de Multiplexação de 8 × Displays de 7-Segmentos:



Applet Java disponível em: `http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/45-misc/50-displays/multiplexed-display8.html` (Abr 2013)

Demultiplexadores

- Realiza a função inversa do multiplexador.
- Toma dados de uma linha de entrada e os distribui a uma determinada linha de saída.
- É conhecido também como "Distribuidor de dados"

Exemplo: circuito interno de um demultiplexador de 1 para 4 linhas:
Tabela funcional:

	Entradas		Saídas			
Ref	S_1	S_0	D_0	D_1	D_2	D_3
0	0	0	I	0	0	0
1	0	1	0	I	0	0
2	1	0	0	0	I	0
3	1	1	0	0	0	I

Equações:

$$\begin{aligned} D_0 &= I\,\overline{S_1}\,\overline{S_0} \\ D_1 &= I\,\overline{S_1}\,S_0 \\ D_2 &= I\,S_1\,\overline{S_0} \\ D_3 &= I\,S_1\,S_0 \end{aligned}$$

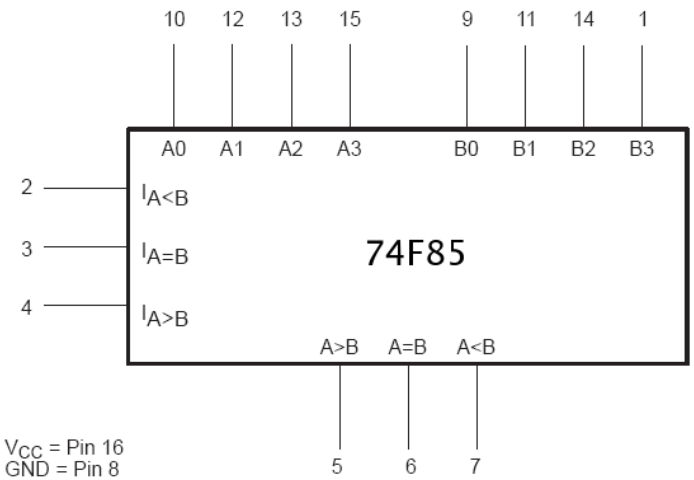
Circuito interno:

<Desenhar>

Comparadores de Magnitude

Comparador de Magnitude:

CI 74F85: Comparador de Magnitude de 4 bits:

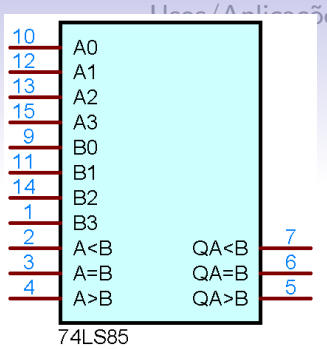


Simbolo.

COMPARING INPUTS				EXPANSION INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

H = High voltage level
L = Low voltage level
X = Don't care

Tabela funcional.



A ≠ B

Comparador de Magnitude:

IC 74F85: Comparador de Magnitude de 4 bits:

COMPARING INPUTS				EXPANSION INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

H = High voltage level
L = Low voltage level
X = Don't care

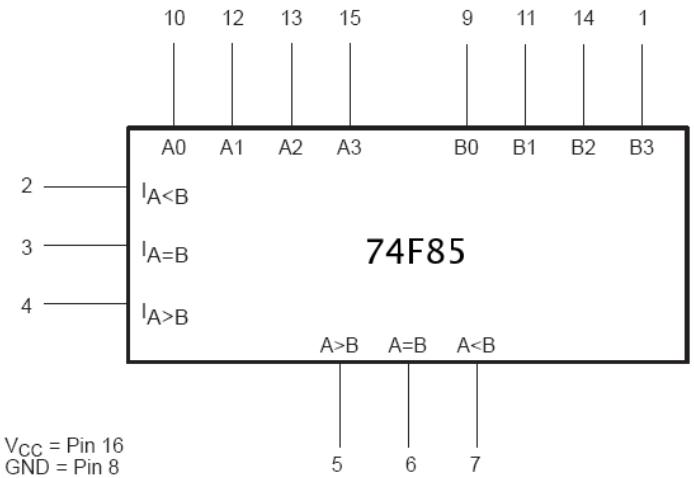
A == B

Tabela funcional.

Comparador de Magnitude – Exercício

Projete o circuito de um comparador de magnitude para palavras de 8 bits usando 2 × CIs 7485.

Dado:



Simbolo.

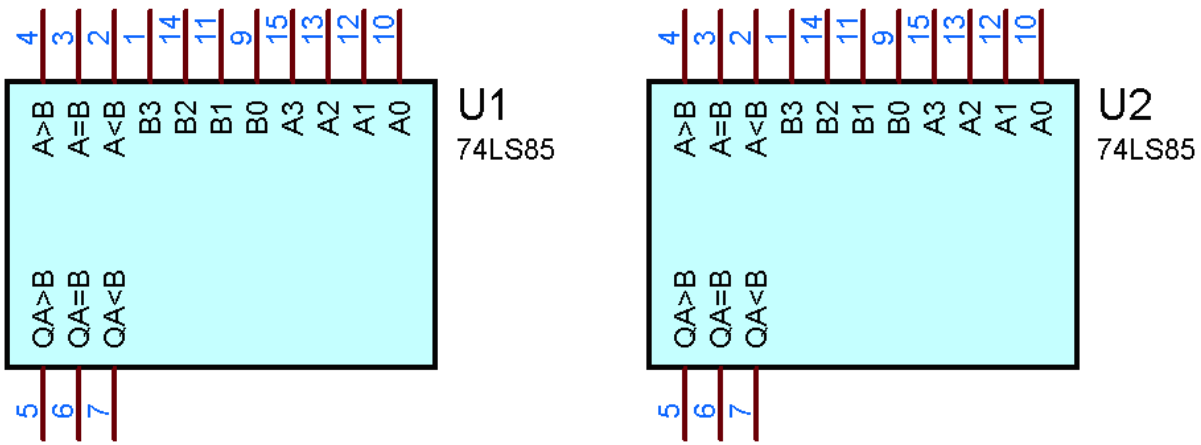
COMPARING INPUTS				EXPANSION INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

H = High voltage level
L = Low voltage level
X = Don't care

Tabela funcional.

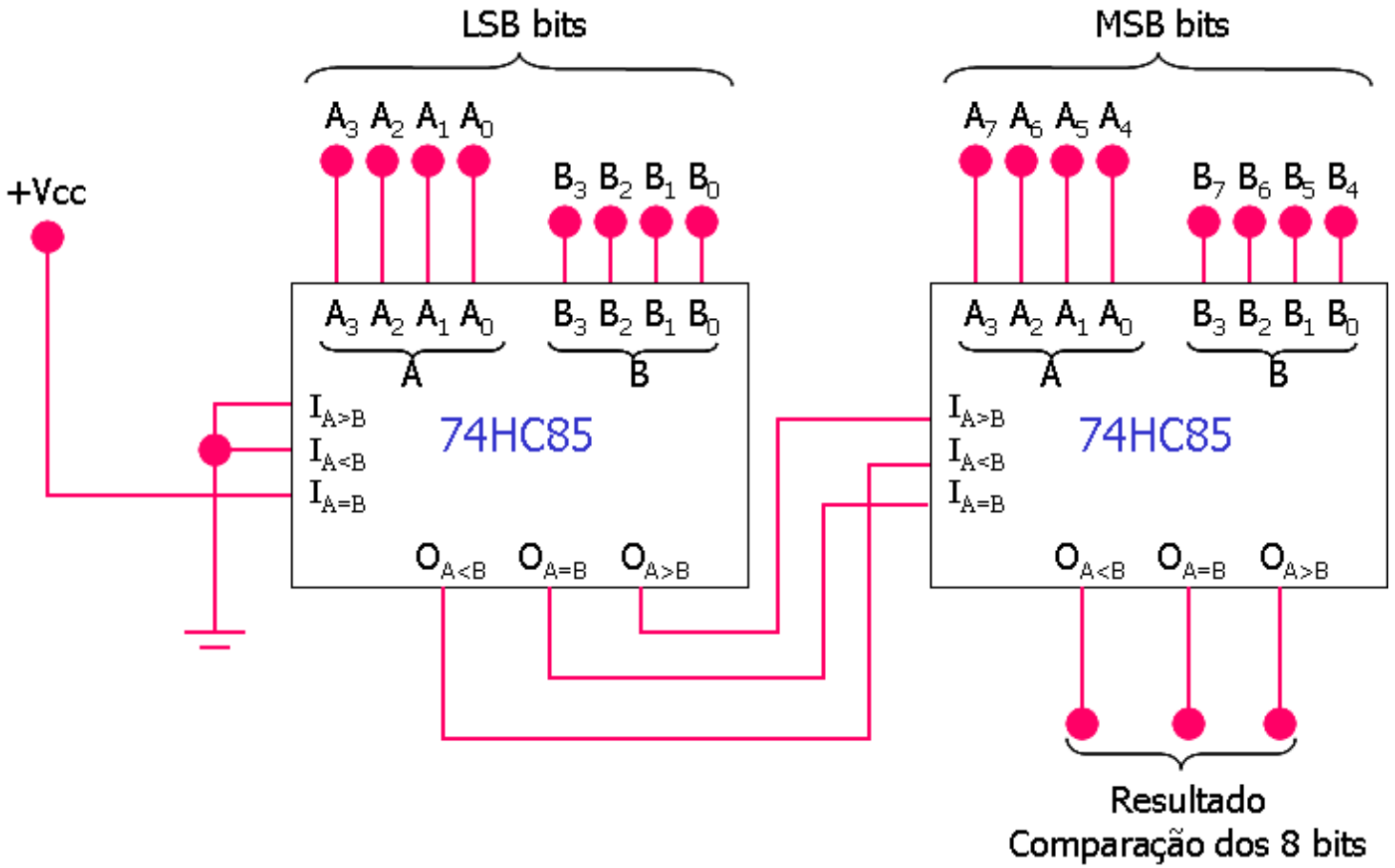
Comparador de Magnitud – Exercício

Projete o circuito de um comparador de magnitude para palavras de 8 bits usando 2 $\times$ CIs 7485.



Comparador de Magnitud – Exercício

Projete o circuito de um comparador de magnitude para palavras de 8 bits usando 2 × CIs 7485.



Usos/ Aplicações Exemplos

Índice

Multiplexador – Aplicação

- Implementação de funções lógicas:

Seja a tabela verdade abaixo que se deseja implementar:

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	
2	0	1	0	1	
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	
6	1	1	0	1	
7	1	1	1	0	

– Equação do circuito ?

Esta tabela verdade pode ser implementada usando CI Multiplexador...
– Como !?

Multiplexador – Aplicação

- Implementação de funções lógicas:

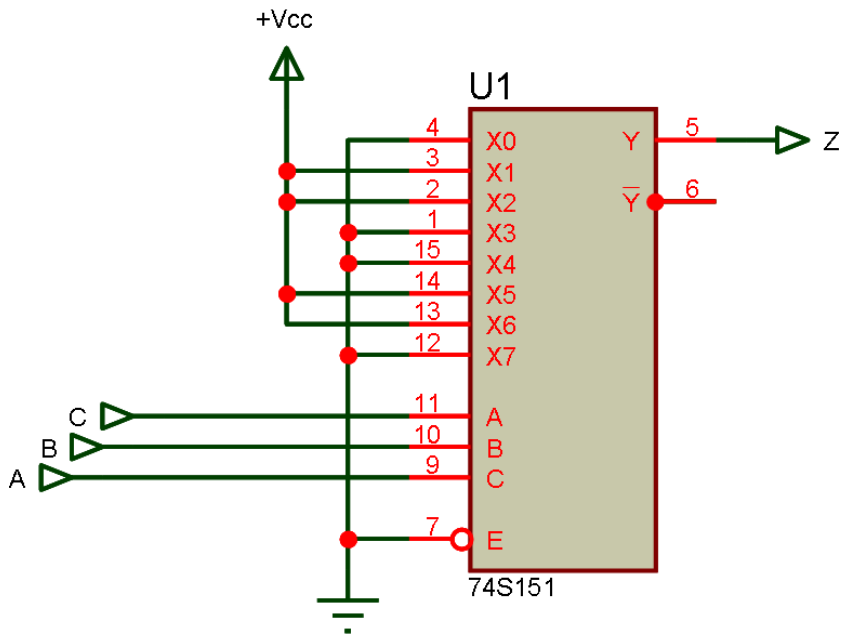
Seja a tabela verdade abaixo que se deseja implementar:

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	$= \overline{A}\,\overline{B}\,C$
2	0	1	0	1	$= \overline{A}\,B\,\overline{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	$= A\,\overline{B}\,C$
6	1	1	0	1	$= A\,B\,\overline{C}$
7	1	1	1	0	

Equação resultante:

$$Z = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C} \qquad (1)$$

A equação (1) pode ser realizada através de um MUX de 8 canais (CI 74151):

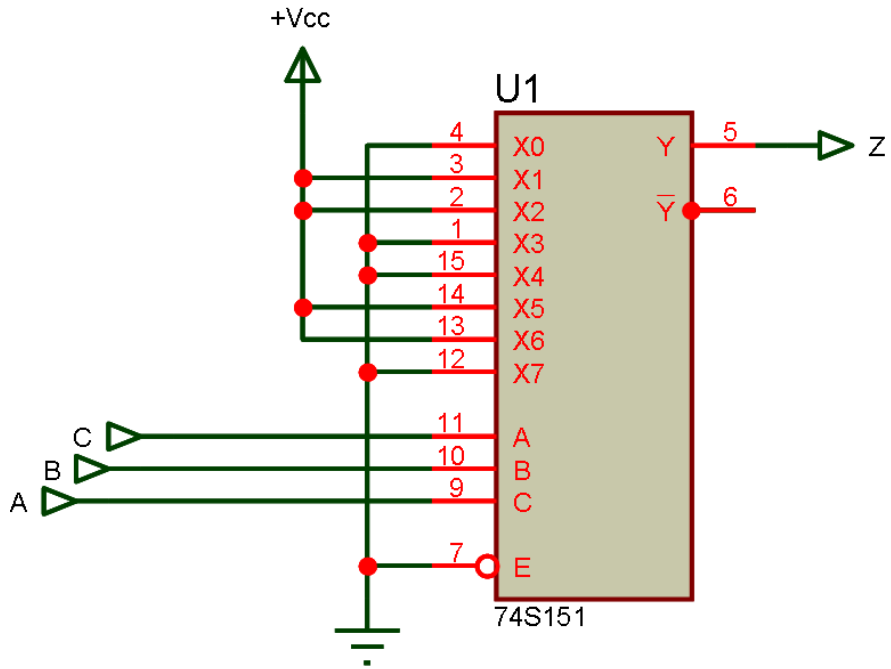


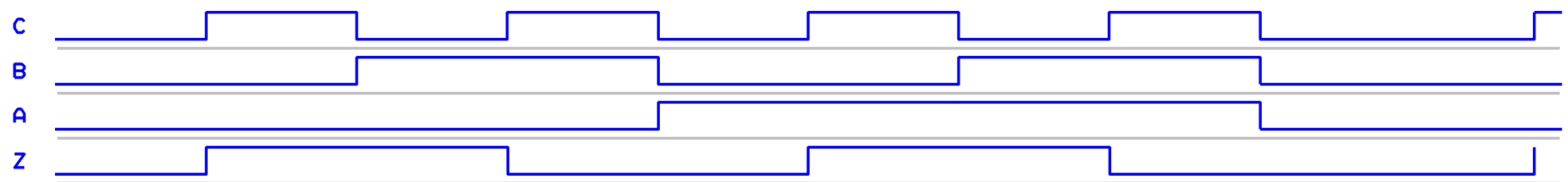
Multiplexador – Aplicação

- Implementação de funções lógicas (continuação):

Circuito resultante para a função:
$Z = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C}$
usando um MUX-8 (CI 74151):

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	$= \overline{A}\,\overline{B}\,C$
2	0	1	0	1	$= \overline{A}\,B\,\overline{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	$= A\,\overline{B}\,C$
6	1	1	0	1	$= A\,B\,\overline{C}$
7	1	1	1	0	





Formas de onda.

Decodificadores – Aplicação

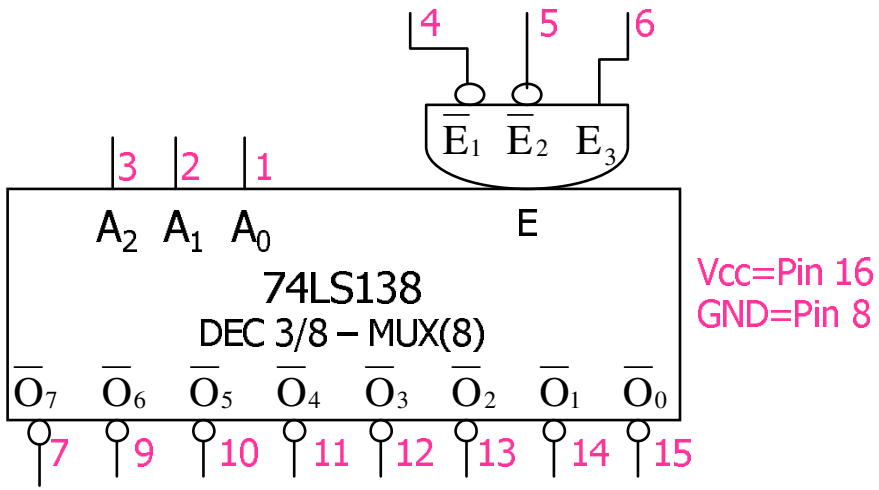
- Implementação de funções lógicas (**outra forma**):

O circuito anterior, eq. (1):
$Z = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C}$
também pode ser implementada atravéz de um DEC de 3 para 8 linhas (CI 74138):

Tabela verdade referente à eq. (1):

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	$= \overline{A}\,\overline{B}\,C$
2	0	1	0	1	$= \overline{A}\,B\,\overline{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	$= A\,\overline{B}\,C$
6	1	1	0	1	$= A\,B\,\overline{C}$
7	1	1	1	0	

Símbolo do DEC 3/8 (CI 74LS138):



Decodificadores – Aplicação

- Implementação de função lógica **usando DEC**:

Seja a função:
$Z = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C}$
Sua correspondente tabela verdade:

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	$= \overline{A}\,\overline{B}\,C$
2	0	1	0	1	$= \overline{A}\,B\,\overline{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	$= A\,\overline{B}\,C$
6	1	1	0	1	$= A\,B\,\overline{C}$
7	1	1	1	0	

Solução: usando DEC **não comercial** com saídas em **ATIVO ALTO**:

- $Z = \sum_m\{1, 2, 5, 6\}$
- poderia ser implementado simplesmente "coletando" as saídas O_1, O_2, O_5 e O_6 do DEC numa porta OR de 4 entradas:
- $Z = O_1 + O_2 + O_5 + O_6$;
- Significa que basta que uma das linhas de saída do DEC citadas (previstas) no item anterior se ative, a saída Z será ativada.

Mas... os DEC's comerciais trabalham com saídas em **ATIVO BAIXO**! – Como resolver !?

Decodificadores – Aplicação

- Implementação de função lógica **usando DEC** – saídas em ATIVO BAIXO:

Seja a função:
$Z = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C}$
Sua correspondente tabela verdade:

Ref	Entradas A	B	C	Saída Z	Observações
0	0	0	0	0	
1	0	0	1	1	$= \overline{A}\,\overline{B}\,C$
2	0	1	0	1	$= \overline{A}\,B\,\overline{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	1	$= A\,\overline{B}\,C$
6	1	1	0	1	$= A\,B\,\overline{C}$
7	1	1	1	0	

Solução: usando DEC comercial (saídas em ATIVO BAIXO):

- $Z = \sum_m \{1, 2, 5, 6\}$
- $Z = O_1 + O_2 + O_5 + O_6$;

Trabalhando a expressão anterior:

- $Z = \overline{\overline{O_1 + O_2 + O_5 + O_6}}$
- $Z = \overline{\overline{O_1} \cdot \overline{O_2} \cdot \overline{O_5} \cdot \overline{O_6}}$

Conclusão: uma porta NAND de 4 entradas resolve o problema.

Decodificadores – Aplicação

- Implementação de função lógica **usando DEC** – saídas em ATIVO BAIXO:

Solução: usando DEC comercial (saídas em ATIVO BAIXO):

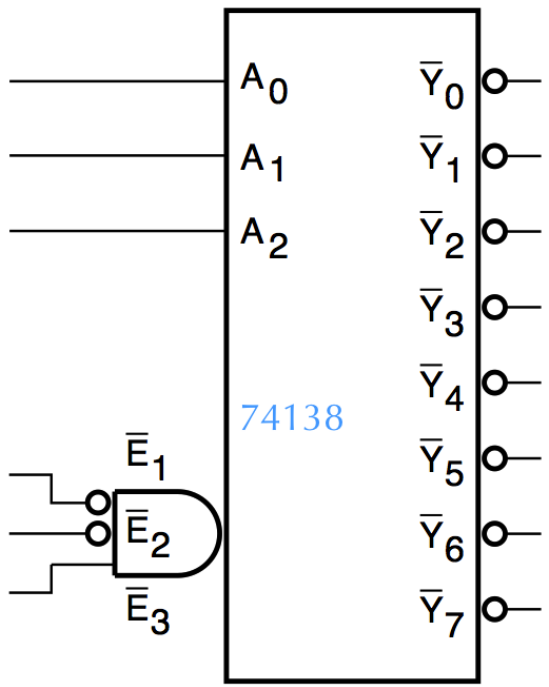
- $Z = \sum_m \{1, 2, 5, 6\}$
- $Z = O_1 + O_2 + O_5 + O_6$;

Trabalhando a expressão anterior:

- $Z = \overline{\overline{O_1 + O_2 + O_5 + O_6}}$
- $Z = \overline{\overline{O_1} \cdot \overline{O_2} \cdot \overline{O_5} \cdot \overline{O_6}}$

Conclusão: uma porta NAND de 4 entradas resolve o problema.

Obs.: Propositalmente falta completar o circuito abaixo:

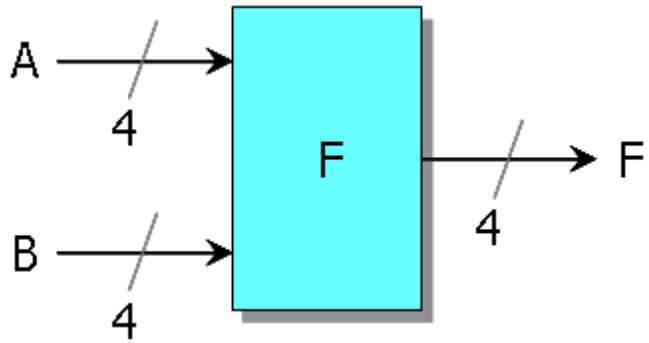


Comparador de Magnitude – Aplicação

Projete o circuito digital necessário para implementar a função lógica:

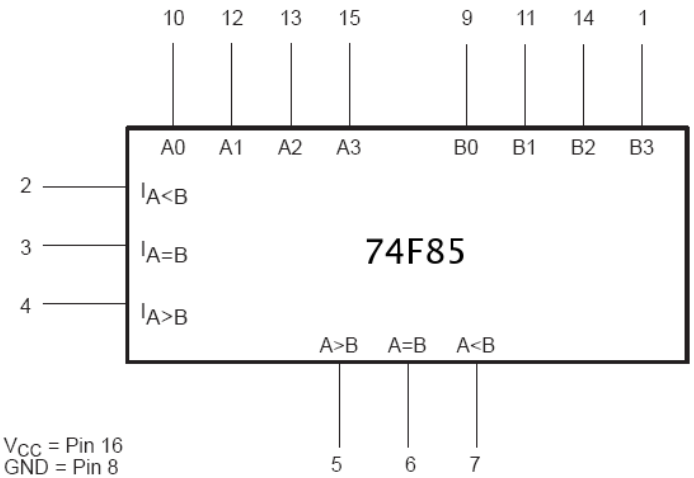
$F = Maior(A, B)$

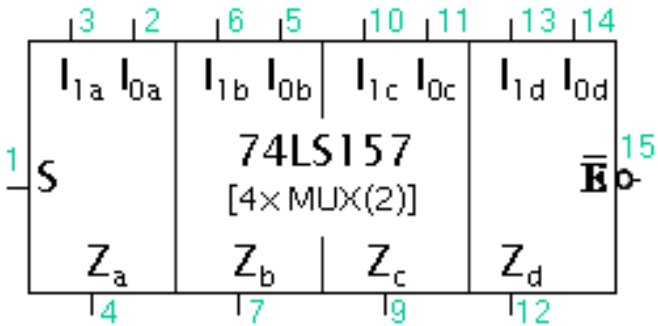
Guiar-se pela figura ao lado.



Dicas: É necessário usar:

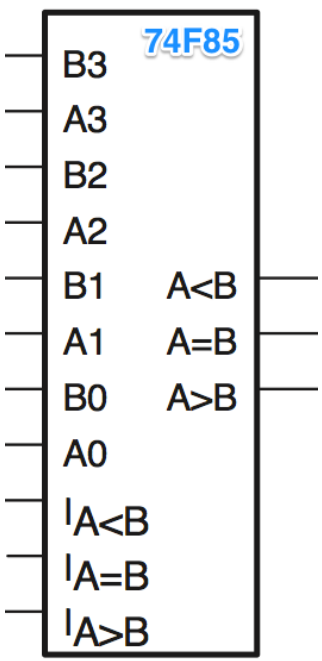
- 1 × Comparador de Magnitude (IC 7485);
- 4 × MUX-2 (IC 74157);

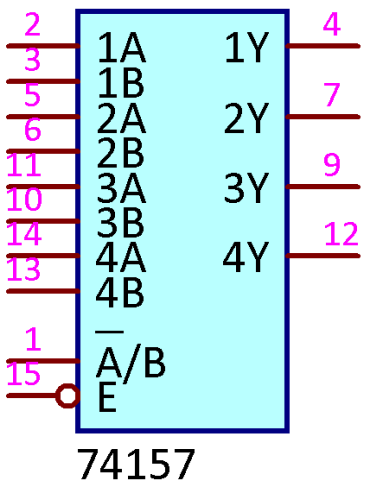




Comparador de Magnitude – Aplicação

- Projeto de: $F = Maior(A, B)$:





Circuitos de Média Escala de Integração

Fim

Notas Adicionais

Este material foi editado usando:

- Sistema de editoração LaTeX:
  Tutoriais disponíveis em:
  - Apostila da Eng. Telecomunicações da Universidade Federal Fluminense:
    `http://www.telecom.uff.br/pet/petws/downloads/apostilas/LaTeX.pdf`
  - Outra apostila LaTeX:

    `http://arquivoescolar.org/bitstream/arquivo-e/197/1/apostila_latex_marcio_nascimento_da_silva_uva_ce_brasil.pdf`
- Editor TexMaker → `http://www.xm1math.net/texmaker/`
- Pacote Beamer para criação dos slides:
  - Quick Start Guide de Ruben Rostamian (Last Updated: Dec 15, 2011)
    `http://www.math.umbc.edu/~rouben/beamer/`
  - Beamer class for LaTeXda Universidade de LeipZig:

    `https://courses.washington.edu/b572/public/beamer2.pdf`

Obs: Todos os programas citados são free e multiplataforma.