



FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 5

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

Aula de hoje: circuitos um pouco mais complexos

5.1. Decodificadores

5.2. Codificadores

5.3. Demultiplexadores

5.4. Multiplexadores

5.5. Métodos Sistemáticos de Projeto

Nesta aula veremos como fazer circuitos um pouco mais complexos dos que já vimos até aqui.

Na seção 5.1 veremos como funcionam os decodificadores.

Na seção 5.2 os codificadores.

Na seção 5.3 os demultiplexadores.

Na seção 5.4 os multiplexadores.

Na seção 5.5 veremos alguns métodos sistemáticos de projeto.

5.1. Decodificadores

- 5.2. Codificadores
- 5.3. Demultiplexadores
- 5.4. Multiplexadores
- 5.5. Métodos Sistemáticos de Projeto

Primeiro, vejamos os decodificadores.

5.1. Decodificadores

- **Decodificador:** Circuito lógico que recebe um conjunto de entradas que representa um número binário e ativa apenas a saída correspondente a tal número

Basicamente, um decodificador é um circuito lógico que recebe um conjunto de entradas que representa o número binário e ativa apenas a saída correspondente a tal número.

5.1. Decodificadores

- **Decodificador:** Circuito lógico que recebe um conjunto de entradas que representa um número binário e ativa apenas a saída correspondente a tal número
 - Exemplo: usando códigos **00, 01, 10 e 11** para identificar 4 lâmpadas (decodificador de 2 bits):



• Tabela verdade:

B	A	S ₃	S ₂	S ₁	S ₀
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

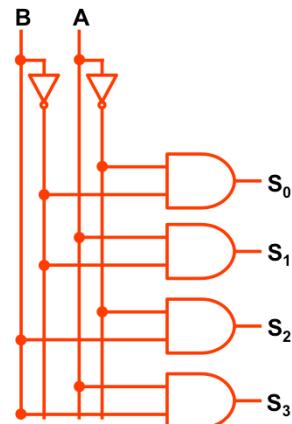
Na prática, imagine que o circuito tenha uma sequência de códigos para identificar quatro lâmpadas. Com o uso de um decodificador é possível traduzir o código inserido no circuito para que seja acendida apenas uma determinada lâmpada.

Veja a tabela verdade produzida por um decodificador.

5.1. Decodificadores

- Decodificador 2/4:

B	A	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



O circuito que produz tais saídas pode ser visto neste slide.

5.1. Decodificadores

- Decodificador 2/4 com **saídas ativas baixas**:

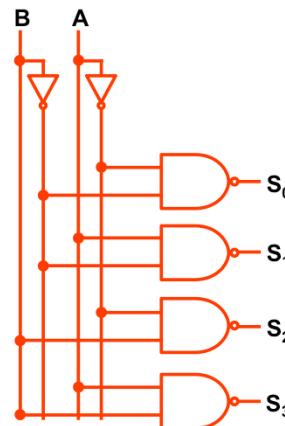
B	A	S ₃	S ₂	S ₁	S ₀
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Existe também uma versão dos decodificadores que produz saídas ativas baixas, ou seja, a saída é considerada ativa não em '1', mas em '0'.

5.1. Decodificadores

- Decodificador 2/4 com **saídas ativas baixas**:

B	A	S ₃	S ₂	S ₁	S ₀
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

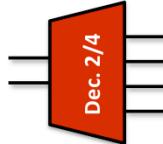
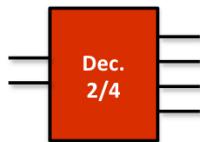


O circuito produzido por tal versão é o seguinte (basta trocar as portas "and" por portas "nand").

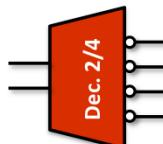
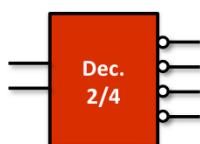
5.1. Decodificadores

- Decodificador - representação:

- Com saídas **ativas altas**:



- Com saídas **ativas baixas**:

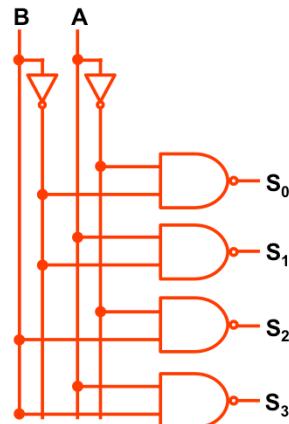
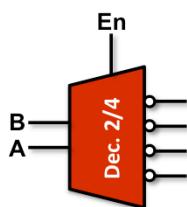


Um decodificador pode ser representado das seguintes formas.

5.1. Decodificadores

- Acrescentando um pino de **habilitação (enable)**

En	B	A	S ₃	S ₂	S ₁	S ₀
0	X	X	1	1	1	1
1	0	0	1	1	1	0
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1

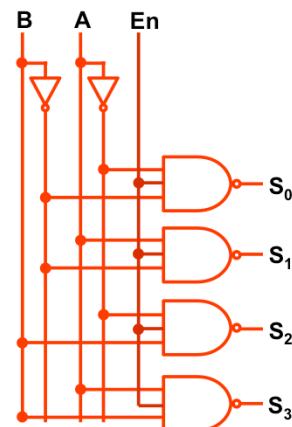
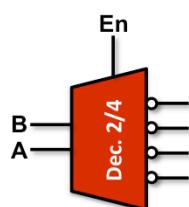


Pode-se ainda acrescentar um pino de habilitação, ou "enable", que faz com que o decodificador funcione apenas se tal pino estiver ativo.

5.1. Decodificadores

- Acrescentando um pino de **habilitação (enable)**

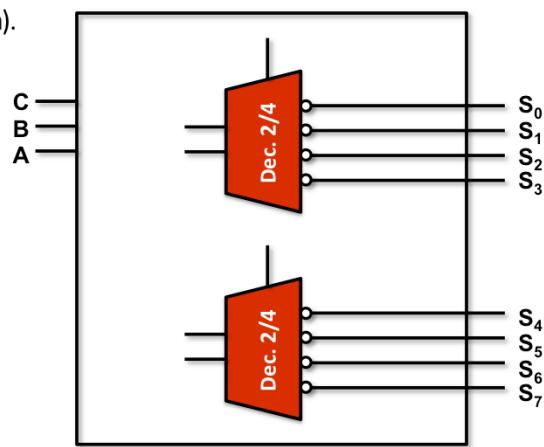
En	B	A	S ₃	S ₂	S ₁	S ₀
0	X	X	1	1	1	1
1	0	0	1	1	1	0
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1



A inserção deste no circuito do decodificador é simples, bastando acrescentar um "fio" em cada entrada de porta "and" (ou "nand", caso o decodificador seja de saída ativa baixa, como no slide).

5.1. Decodificadores

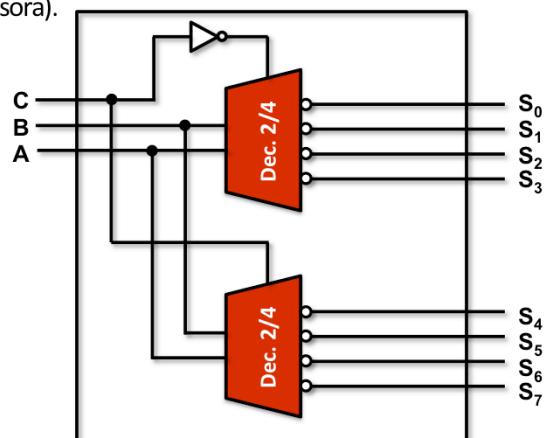
- Com pino de **habilitação (enable)** é possível associar decodificadores para montar um decodificador maior.
 - **Exemplo:** Montar um decodificador 3/8 usando dois 2/4 com enable. (Dica: use também uma porta inversora).



Com esse simples acréscimo é possível associar decodificadores para montar um decodificador maior, como no exemplo deste slide.

5.1. Decodificadores

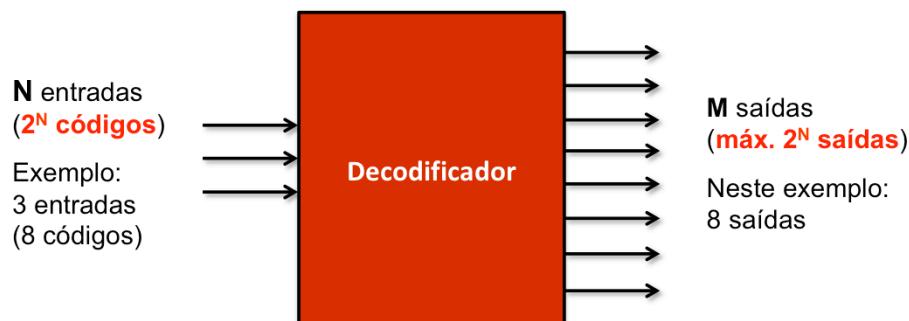
- Com pino de **habilitação (enable)** é possível associar decodificadores para montar um decodificador maior.
 - **Exercício 1:** Montar um decodificador 3/8 usando dois 2/4 com enable. (Dica: use também uma porta inversora).



Para realizar tal tarefa basta fazer com que o enable (que, neste caso, foi escolhido como a entrada C) em um dos decodificadores seja tivado em '1' e o outro em '0'. Para isso coloca-se uma porta "not" em uma das entradas de enable. As demais entradas (A e B) são ligadas normalmente em ambos os decodificadores.

5.1. Decodificadores

- De maneira geral:

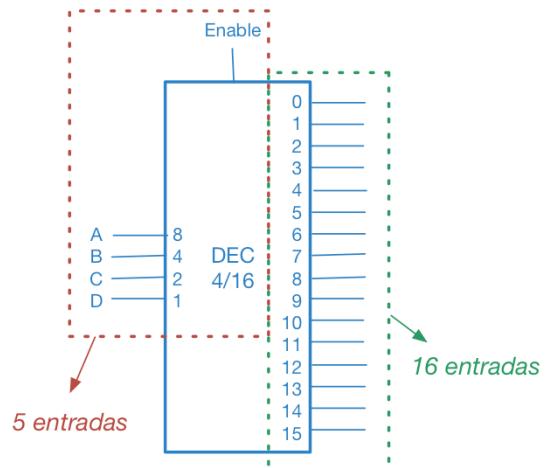


De maneira geral (sem enable), entram N entradas, o que gera 2^N (dois elevado a N) códigos possíveis. Enquanto isso, são geradas M saídas, em que M é menor ou igual a 2^N (dois elevado a N).

PROBLEMAS

Problema 5.1. Quantas entradas irá possuir um decodificador com 16 saídas e entrada de *enable*?

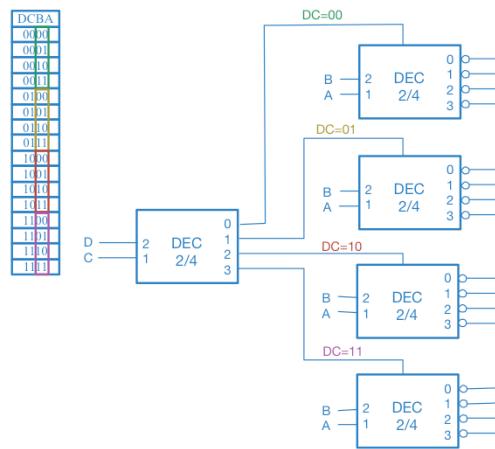
Solução Problema 5.1:



PROBLEMAS

Problema 5.2. Montar um decodificador 4/16 com saídas ativas baixas usando um decodificador 2/4 com saídas ativas altas e quatro decodificadores 2/4 com saídas ativas baixas e *enable*.

Solução Problema 5.2:



5.1. Decodificadores

5.2. Codificadores

5.3. Demultiplexadores

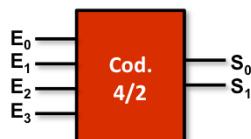
5.4. Multiplexadores

5.5. Métodos Sistemáticos de Projeto

Agora falaremos sobre os codificadores.

5.2. Codificadores

- De maneira simplificada: **codificadores** fazem a função oposta à dos **decodificadores**.
- Assim, **codificadores** em geral possuem 2^N entradas e N saídas



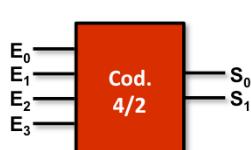
E ₃	E ₂	E ₁	E ₀	S ₁	S ₀
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Basicamente, os codificadores fazem a função oposta à dos decodificadores. Logo, os codificadores em geral (sem enable) possuem 2^N (dois elevado a N) entradas e N saídas.

Observe esta tabela verdade dos codificadores e perceba que é exatamente o contrário da de um decodificador com saída ativa alta.

5.2. Codificadores

- De maneira simplificada: **codificadores** fazem a função oposta à dos **decodificadores**.
- Assim, **codificadores** em geral possuem 2^N entradas e N saídas
- Podem ser ativos baixos e ativos altos
- Para evitar problemas quando duas entradas são ativadas, existem codificadores com prioridade
 - Exemplo: codificador com prioridade e entradas ativas altas



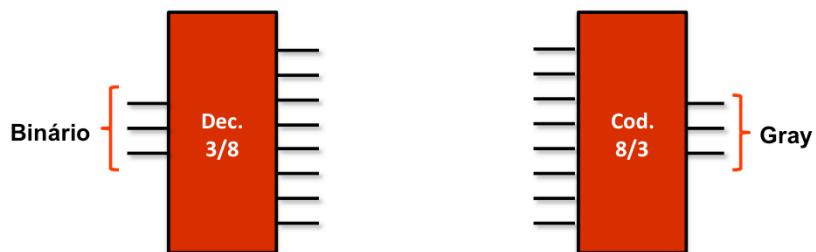
E_3	E_2	E_1	E_0	S_1	S_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

Assim como os decodificadores, os codificadores também podem ser tanto ativos baixos como altos.

Para evitar problemas quando duas entradas estãoativas ao mesmo tempo, existem os codificadores com prioridade. Na sua tabela verdade, há X no lugar dos 0's após o primeiro '1' da entrada.

PROBLEMAS

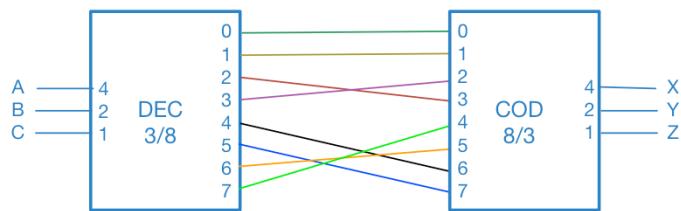
Problema 5.4. Faça o projeto de um conversor binário para *Gray* de 3 bits usando um decodificador (3:8) e um codificador (8:3).



PROBLEMAS

Problema 5.4. Faça o projeto de um conversor binário para *Gray* de 3 bits usando um decodificador (3:8) e um codificador (8:3).

Solução Problema 5.4:



Binário	Gray
ABC	XYZ
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

5.1. Decodificadores

5.2. Codificadores

5.3. Demultiplexadores

5.4. Multiplexadores

5.5. Métodos Sistemáticos de Projeto

Sigamos para os demultiplexadores.

5.3. Demultiplexadores

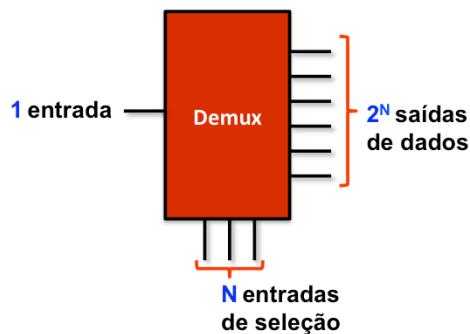
- Também conhecidos como **distribuidores de dados**.
- Permitem que a **entrada** seja desviada para **uma das saídas**.
- Aplicações:
 - Distribuição de dados em geral
 - Roteamento
 - Conversão série/paralelo

Estes componentes são também conhecidos como distribuidores de dados, isso porque permitem que a entrada seja desviada para uma das saídas.

Algumas aplicações dos demultiplexadores são em distribuição de dados em geral, roteamento e conversão série/paralelo.

5.3. Demultiplexadores

- Também conhecidos como **distribuidores de dados**.
- Permitem que a **entrada** seja desviada para **uma das saídas**.
- Estrutura:



De maneira geral, cada entrada de dados pode ter **múltiplos bits** e, portanto, a saída pode ter **múltiplos bits**.

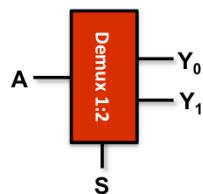
A sua estrutura básica possui apenas uma entrada, 2^N saídas de dados para onde a entrada pode ser enviada e N entradas de seleção.

Entretanto, de maneira geral, cada entrada de dados pode ter múltiplos bits e, portanto, a saída pode ter múltiplos bits.

5.3. Demultiplexadores

- Demultiplexador 1:2

S	Y_1	Y_0
0	0	A
1	A	0



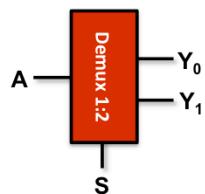
Aqui temos um demultiplexador simples, cuja entrada A possui apenas um bit, assim como saída de seleção S. Assim haverá apenas duas saídas de dados (Y_1 E Y_0) de um bit cada.

Neste slide também é apresentada a sua tabela verdade compactada.

5.3. Demultiplexadores

- Demultiplexador 1:2

S		Y_1	Y_0
0	0	0	A
1	A	A	0



S	A	Y_1	Y_0
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

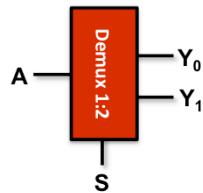
A sua tabela verdade completa seria esta.

Veja que, assim como na tabela anterior, se S é '0' a saída de Y_0 depende de A, enquanto quando S é '1' a saída Y_1 depende de A.

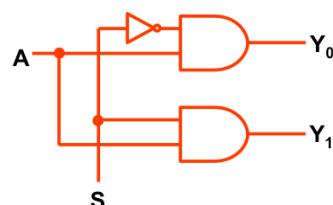
5.3. Demultiplexadores

- Demultiplexador 1:2

S		Y_1	Y_0
0		0	A
1		A	0



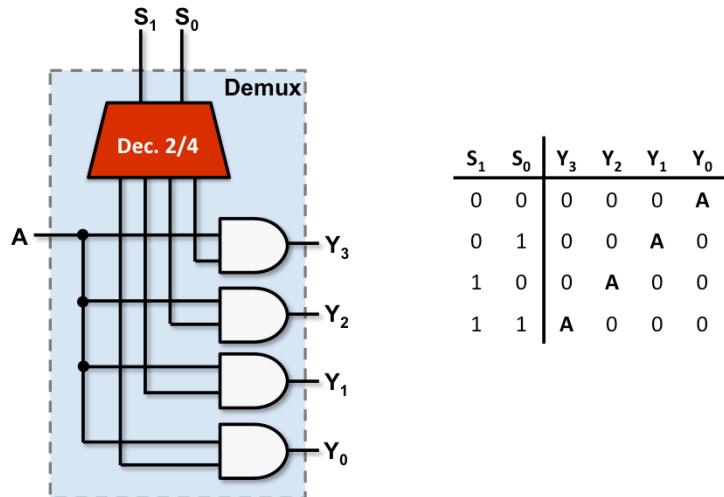
S	A	Y_1	Y_0
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0



E, a partir dela, obtemos o seguinte circuito.

5.3. Demultiplexadores

- Um demultiplexador com 4 saídas:



Para construir um demultiplexador com 4 ou mais saídas de dados é eficiente utilizarmos um demultiplexador no circuito, como ilustrado neste slide.

5.1. Decodificadores

5.2. Codificadores

5.3. Demultiplexadores

5.4. Multiplexadores

5.5. Métodos Sistemáticos de Projeto

Vejamos agora os multiplexadores.

5.4. Multiplexadores

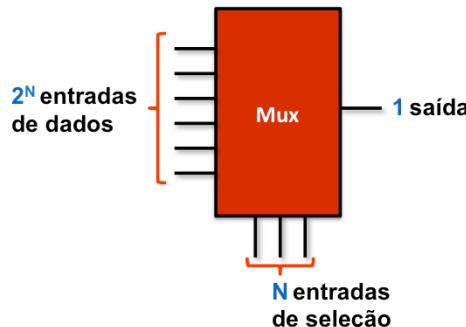
- Também conhecidos como **seletores de dados**
- Permitem que apenas uma entrada de cada vez fique disponível na saída
- Aplicações:
 - Seleção de dados
 - Roteamento
 - Conversão paralelo/série
 - *Barrel-shifter*
 - Implementação sistemática de funções lógicas
 - *Lookup table*

Os multiplexadores são conhecidos também como seletores de dados, isso porque permitem que apenas uma entrada de cada vez fique disponível na saída.

Algumas das suas aplicações são em seleção de dados, roteamento, conversão paralelo/série, como "barrel-shifter", em implementações sistemáticas de funções lógicas e em "lookup tables".

5.4. Multiplexadores

- Também conhecidos como **seletores de dados**
- Permitem que apenas uma entrada de cada vez fique disponível na saída
- Estrutura:



- De maneira geral, cada entrada de dados pode ter **múltiplos bits** e, portanto, a saída pode ter **múltiplos bits**

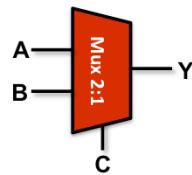
A sua estrutura básica é exatamente oposta à dos demultiplexadores, com 2^N entradas de dados, N entrada de seleção e N entradas de seleção.

Da mesma forma que os demultiplexadores, cada entrada de dados pode ter múltiplos bits, fazendo com que a saída, portanto, também possa ter múltiplos bits.

5.4. Multiplexadores

- Multiplexador 2:1

C	Y
0	A
1	B



C	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

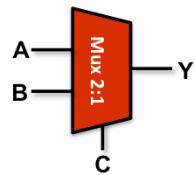
Vejamos este exemplo: temos duas entradas de dados de um bit cada, A e B, portanto uma saída de dados de um bit Y e uma entrada de seleção C de um bit.

Não deixe de observar as tabelas verdade deste multiplexador.

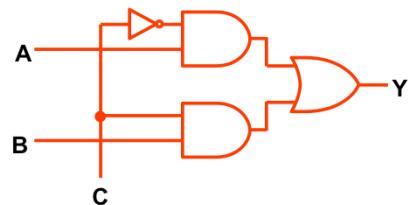
5.4. Multiplexadores

- Multiplexador 2:1

C	Y
0	A
1	B



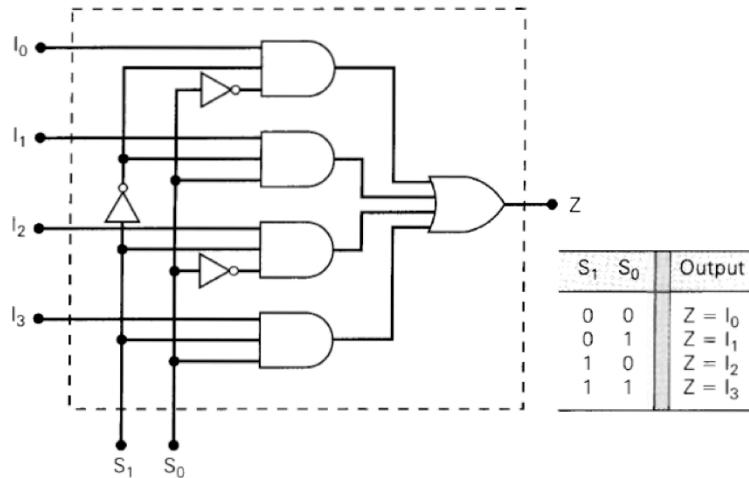
C	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Elas produzem este circuito.

5.4. Multiplexadores

- Multiplexador de 4 entradas



Para construir multiplexadores de 4 ou mais entradas de dados a lógica de obtenção do circuito é a mesma.

- 5.1. Decodificadores
- 5.2. Codificadores
- 5.3. Demultiplexadores
- 5.4. Multiplexadores

5.5. Métodos Sistemáticos de Projeto

Por último, vejamos os métodos sistemáticos de projeto.

5.5. Métodos Sistemáticos de Projeto

- Uma desvantagem do projeto por **soma de minitermos** ou **produto de maxitermos**: mesmo sabendo o número de **entradas** e **saídas** do circuito a ser projetado, não é possível saber de antemão:
 - Quantas portas lógicas serão necessárias
 - Como tais portas estarão posicionadas e conectadas
- Ideia nos **métodos sistemáticos**: fornecer uma estrutura genérica básica para implementação de sistemas digitais que possa ser configurada sem muito esforço
 - **Vantagens**: facilidade e flexibilidade no projeto
 - **Desvantagem**: circuitos maiores e com maior custo/desperdício

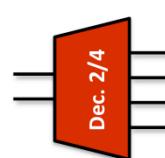
Uma desvantagem do projeto por soma de minitermos ou produto de maxitermos é que, mesmo sabendo o número de entradas e saídas do circuito a ser projetado, não é possível saber de antemão quantas portas lógicas serão necessárias e nem como tais portas estarão posicionadas e conectadas.

A ideia nos métodos sistemáticos é fornecer uma estrutura genérica básica para implementação de sistemas digitais que possa ser configurada sem muito esforço. As grandes vantagens disso são a facilidade e a flexibilidade que se pode ter no projeto, entretanto o custo disso são circuitos maiores e com maior desperdício.

5.5. Métodos Sistemáticos de Projeto

- Decodificadores podem ser usados para implementar **funções lógicas genéricas** de forma sistemática
 - Qual função lógica descreve cada saída de um decodificador ativo alto?
 - Cada saída é dada por um **minitermo** diferente
 - Fazendo uma **soma de minitermos** é possível implementar qualquer função lógica usando um decodificador

B	A	S ₃	S ₂	S ₁	S ₀
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

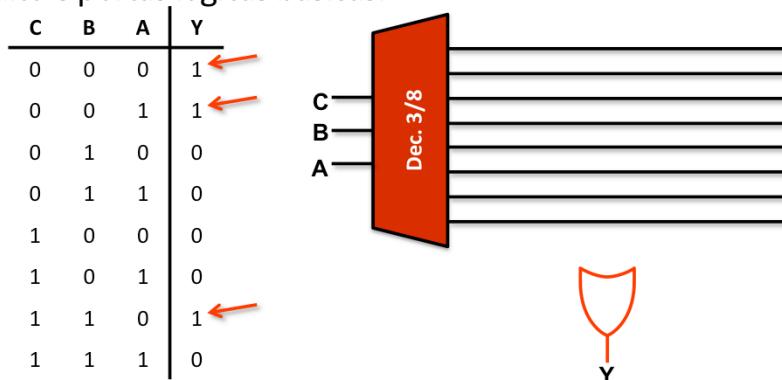


Um dos métodos sistemáticos de projeto utiliza decodificadores para implementar funções lógicas genéricas.

Neste método é importante observar que, no final das contas, cada saída de um decodificador é dada por um minitermo diferente. A partir disso, pode-se fazer uma soma de minitermos para implementar qualquer função lógica usando um decodificador.

5.5. Métodos Sistemáticos de Projeto

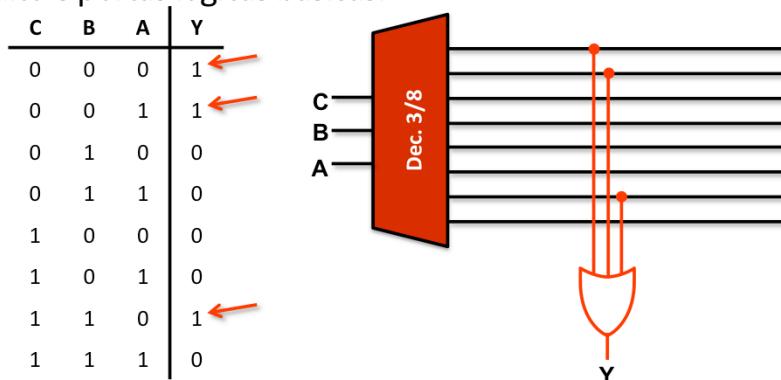
- **Exemplo:** Implementar o circuito correspondente à tabela verdade apresentada abaixo usando um decodificador ativo alto e portas lógicas básicas.



Tomemos este exemplo, notando os três minitermos necessários para produzir tal tabela verdade.

5.5. Métodos Sistemáticos de Projeto

- **Exemplo:** Implementar o circuito correspondente à tabela verdade apresentada abaixo usando um decodificador ativo alto e portas lógicas básicas.

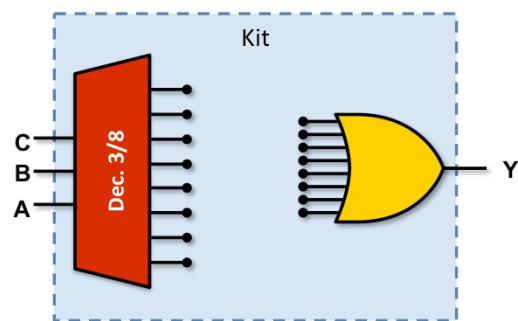


Para produzir esta tabela verdade através deste método sistemático basta ligar as saídas do decodificador correspondentes a tais minitermos em uma porta "or".

5.5. Métodos Sistemáticos de Projeto

- **Exemplo:** Implementar o circuito correspondente à tabela verdade apresentada abaixo usando um decodificador ativo alto e portas lógicas básicas.

C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

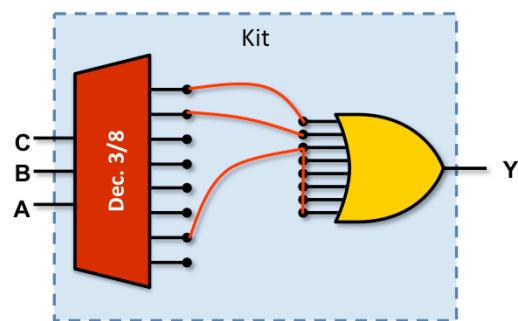


Uma forma mais padronizada de representar isso é a seguinte...

5.5. Métodos Sistemáticos de Projeto

- **Exemplo:** Implementar o circuito correspondente à tabela verdade apresentada abaixo usando um decodificador ativo alto e portas lógicas básicas.

C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

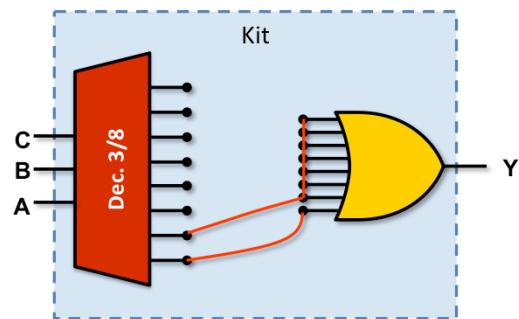


Note que quando ligamos uma mesma saída do decodificador em várias entradas da porta "or" é o mesmo que ligá-la em uma só.

5.5. Métodos Sistemáticos de Projeto

- Outro exemplo:

C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Aqui temos outro exemplo de implementação através deste método.

PROBLEMAS

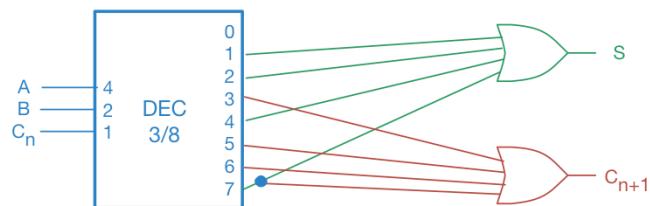
Problema 5.5. Faça o projeto do circuito que funciona de acordo com a tabela verdade de um somador completo apresentada usando um decodificador com saídas ativas altas e duas portas OR de 4 entradas.

A	B	C _n	C _{n+1}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

PROBLEMAS

Problema 5.5. Faça o projeto do circuito que funciona de acordo com a tabela verdade de um somador completo apresentada usando um decodificador com saídas ativas altas e duas portas OR de 4 entradas.

Solução Problema 5.5:



5.5. Métodos Sistemáticos de Projeto

- Multiplexadores também podem ser usados para o projeto sistemático de projeto de circuitos lógicos, de forma análoga ao projeto com decodificadores
- De maneira geral, é possível construir qualquer função lógica com 1 saída usando um multiplexador apropriado
 - Várias saídas: vários multiplexadores...

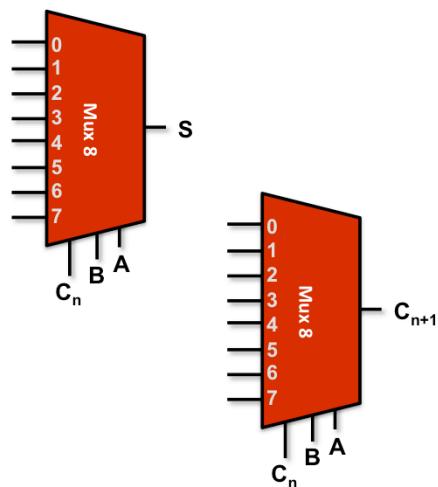
Outro método sistemático de projeto envolve a utilização de multiplexadores, e isso é feito de forma análoga ao projeto com decodificadores.

De maneira geral, é possível construir qualquer função lógica com uma saída usando um multiplexador apropriado. Se tivermos várias saídas, basta que se utilize vários multiplexadores.

5.5. Métodos Sistemáticos de Projeto

- **Exemplo:** Projetar um circuito que funciona de acordo com a tabela verdade dada abaixo usando dois multiplexadores de 8 entradas.

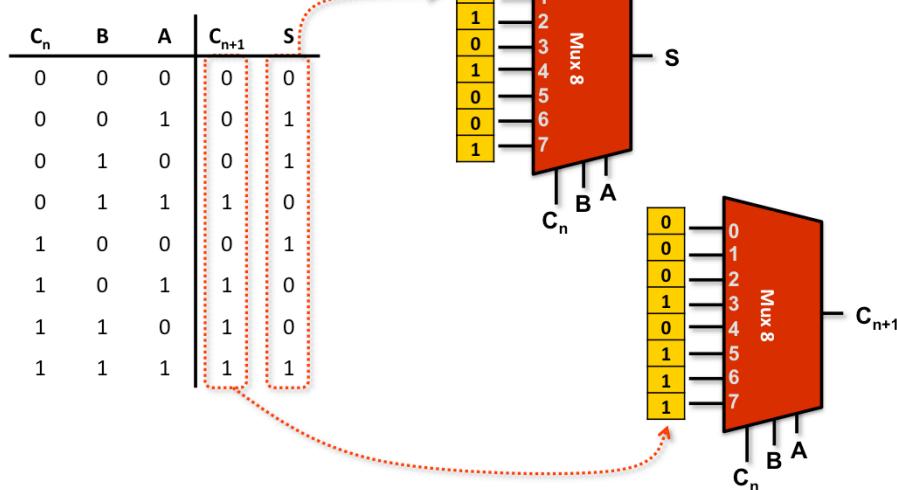
C_n	B	A	C_{n+1}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Vejamos o exemplo deste slide.

5.5. Métodos Sistemáticos de Projeto

- Exemplo:** Projetar um circuito que funciona de acordo com a tabela verdade dada abaixo usando dois multiplexadores de 8 entradas.

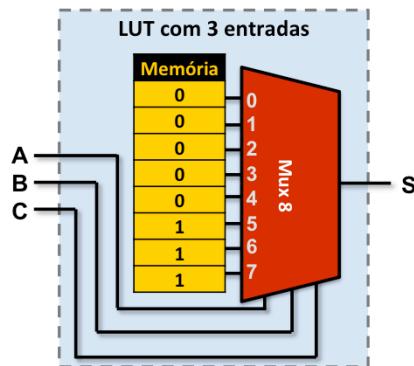


Para resolvê-lo basta ligarmos os números das saídas em suas entradas correspondentes no multiplexador. Assim, estas entradas de dados serão selecionadas de acordo com a entrada de seleção, que corresponde às entradas da tabela verdade em questão.

Vendo por outro ângulo, é como se houvesse uma tabela que é consultada para que o circuito diga qual deve ser a saída.

5.5. Métodos Sistemáticos de Projeto

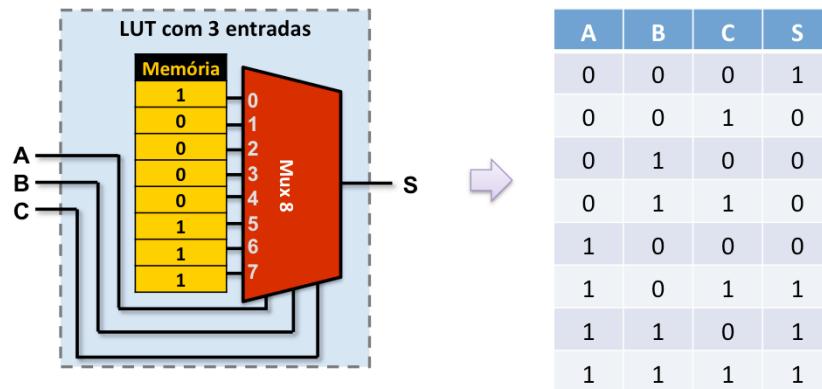
- **Look-up table (LUT):**
 - Pode ser usada para implementar qualquer função lógica apenas gravando os valores de sua memória
 - Mudando valores na **memória**, função é modificada



Assim, tal técnica ficou conhecida como "lookup table" (tabela de pesquisa). Ela pode ser utilizada para implementar qualquer função lógica apenas gravando os valores necessários em sua memória (veja o slide). Assim, mudando os valores na memória, a própria função é modificada.

5.5. Métodos Sistemáticos de Projeto

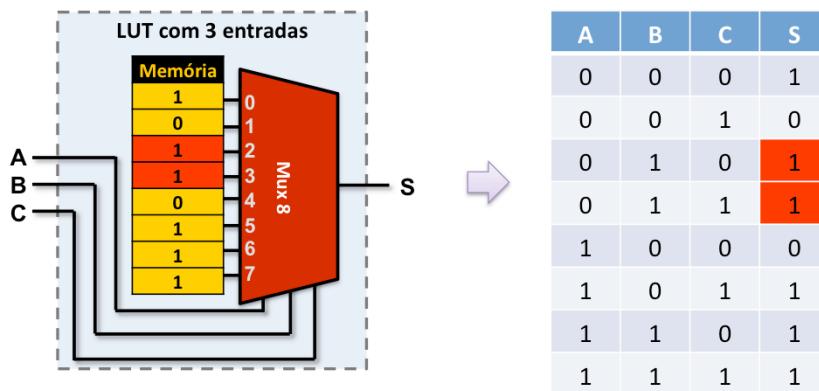
- **Look-up table (LUT):**
 - Pode ser usada para implementar qualquer função lógica apenas gravando os valores de sua memória
 - Mudando valores na memória, função é modificada



A "lookup table" em questão produz esta tabela verdade.

5.5. Métodos Sistemáticos de Projeto

- **Look-up table (LUT):**
 - Pode ser usada para implementar qualquer função lógica apenas gravando os valores de sua memória
 - **Mudando valores na memória, função é modificada**



Veja bem, novamente, que mudando os valores na memória, a própria função é modificada.

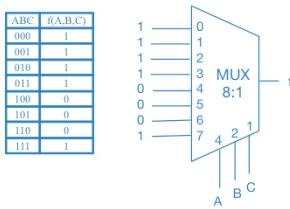
PROBLEMAS

Problema 5.6. Implemente a função booleana $f(A, B, C) = \bar{A}\bar{B} + \bar{A}\bar{C} + BC$ utilizando:

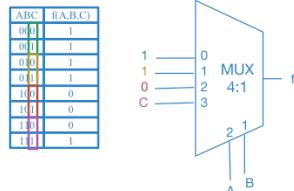
- Um multiplexador com 3 entradas de seleção MUX(8:1).
- Um multiplexador com 2 entradas de seleção MUX(4:1).
- Um multiplexador com 1 entrada de seleção MUX(2:1) e uma porta AND de 2 entradas.

Solução Problema 5.6:

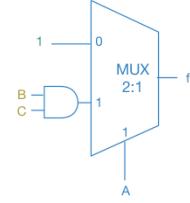
a)



b)



c)



PROBLEMAS

Problema 5.7. Faça o projeto do circuito com entrada de 3 bits, $X=\{x_2, x_1, x_0\}$, e saída de 2 bits, $Y=\{y_1, y_0\}$, que funciona de acordo com a tabela verdade apresentada usando apenas multiplexadores MUX(8:1), MUX(4:1), MUX(2:1) e portas lógicas NAND de duas entradas.

x_2	x_1	x_0	y_1	y_0
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

PROBLEMAS

Problema 5.7. Faça o projeto do circuito com entrada de 3 bits, $X=\{x_2, x_1, x_0\}$, e saída de 2 bits, $Y=\{y_1, y_0\}$, que funciona de acordo com a tabela verdade apresentada usando apenas multiplexadores MUX(8:1), MUX(4:1), MUX(2:1) e portas lógicas NAND de duas entradas.

Solução Problema 5.7:

a)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	1
100	0
101	1
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

b)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	1
100	0
101	1
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

c)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	1
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

d)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

e)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

f)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

g)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

h)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

i)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

j)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

k)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

l)

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—

$x_2 x_1 x_0$	y_1
000	1
001	1
010	1
011	0
100	0
101	0
110	—
111	—



FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 5

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

Exercícios

- **Exercícios deste conjunto de slides**
- Exercícios da 11^a edição do livro do Tocci:
 - **9.1 e 9.2**
 - **9.5** (resolver somente depois que estudarmos contadores)
 - **9.13** [apenas itens (a), (b) e (c)]
 - **9.37, 9.38 e 9.39**
- **A versão digital da 11^a edição do livro do Tocci está disponível no site da BU**
 - Mais especificamente em:
http://150.162.4.10/pergamum/biblioteca_s/php/login_pearson.php