Circuitos Lógicos

Profa. Grace S. Deaecto

Faculdade de Engenharia Mecânica / UNICAMP
13083-860, Campinas, SP, Brasil.
grace@fem.unicamp.br

Segundo Semestre de 2013

NOTA AO LEITOR

Estas notas de aula foram inteiramente baseadas nas seguintes referências :

- T. Floyd, "*Digital Fundamentals*", 10th Edition, Prentice Hall, 2009.
- R. J. Tocci, N. S. Widmer, G. L. Moss, "*Sistemas Digitais : Princípios e Aplicações*", Prentice-Hall, 2007.
- I. V. Iodeta, F. G. Capuano, "*Elementos de Eletrônica Digital*", Editora Érica, 2006.
- V. A. Pedroni, "*Circuit Design and Simulation with VHDL*", 2^{nd} Edition, MIT, 2010.

1. Circuitos Combinacionais
   - Funções e variáveis lógicas
   - Operações e portas lógicas
   - Tabela verdade e expressão lógica
   - Álgebra de Boole
   - Minimização
   - Aplicação prática : Elevador
   - Síntese de circuitos combinacionais

Funções e variáveis lógicas

- No nosso dia-a-dia estamos repletos de circunstâncias em que somente dois estados são possíveis : luz apagada ou acesa, pessoa morta ou viva, porta fechada ou aberta, etc.
- Em 1854 o matemático George Boole descreveu um conjunto de regras capaz de relacionar estas circunstâncias (entradas) de maneira a permitir a tomada de decisões (saídas).
- Este conjunto de regras foi denominado de álgebra booleana.
- A ideia deste capítulo é estudar a álgebra booleana nos aspectos de análise, síntese e simplificação de expressões lógicas.

Funções e variáveis lógicas

Seguem algumas definições importantes :

- Variável booleana é uma quantidade que pode ser, em diferentes momentos, igual a 0 ou 1.
- Função booleana associa a cada n variáveis de entrada uma única saída.
- Podemos descrever uma função booleana utilizando
  - tabela verdade
  - portas lógicas
  - equações
  - formas de onda
- Diferente da álgebra comum, a álgebra booleana possui somente três operações básicas : OR, AND e NOT, conhecidas como operações lógicas.

Tabela verdade

- Seja uma função $f(A_1, \cdots, A_n)$ com n entradas. A tabela verdade expressa o estado da saída para todas as combinações possíveis dos estados de entrada $\{A_1, \cdots, A_n\}$. Segue um exemplo para duas entradas.

A_1	A_2	$f(A_1, A_2)$
0	0	1
0	1	1
1	0	1
1	1	0

- Além de 0s e 1s a função $f(\cdot)$ pode ser igual ao caracter x, chamado de don't care. Este caracter serve para indicar que para uma dada combinação de entradas, x pode ser tanto 0 como 1. Como veremos, o don't care é estratégico no processo de simplificação booleana.

Operações e portas lógicas

- Operação NOT : Para qualquer entrada A, ela é definida como

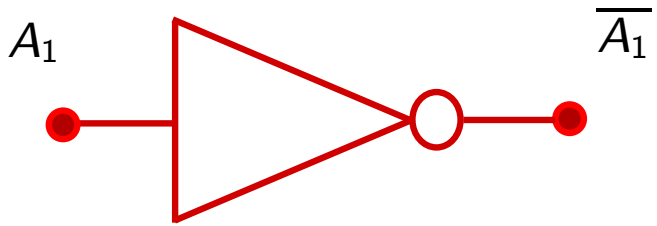
$$f(A) = \overline{A}$$

ou seja, é a entrada negada (barrada). Para uma entrada A_1, por exemplo, temos

Tabela verdade

A_1	$f(A_1)$
0	1
1	0

Porta lógica



Operações e portas lógicas

- Operação OR : Para entradas $\{A_1, \cdots, A_n\}$, ela é definida como

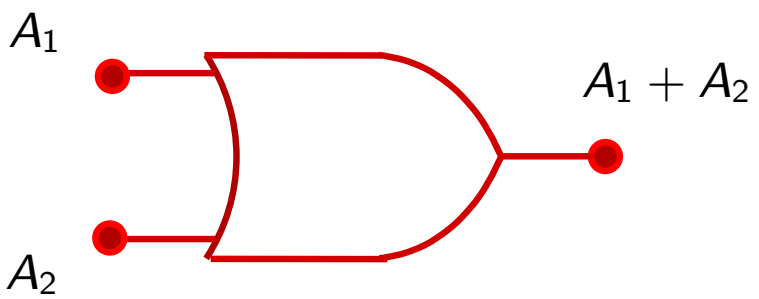
$$f(A_1, \cdots, A_n) = \sum_{i=1}^{n} A_i$$

e vale 1 se qualquer uma das entradas for igual a 1. Para duas entradas temos :

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Porta lógica



Operações e portas lógicas

- Operação AND : Para entradas $\{A_1, \cdots, A_n\}$, ela é definida como

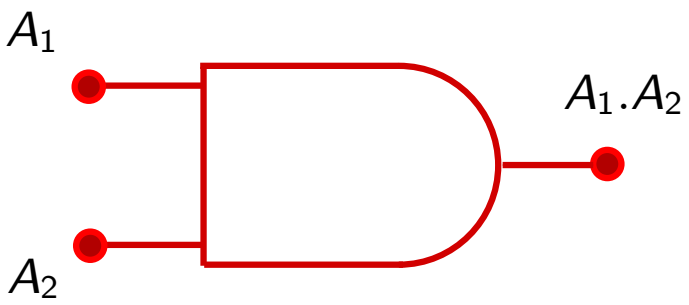
$$f(A_1, \cdots, A_n) = \prod_{i=1}^{n} A_i$$

e vale 1 apenas se todas as entradas forem iguais a 1. Para duas entradas temos :

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Porta lógica



Operações e portas lógicas

- Operação NOR : É a operação OR negada. Para duas entradas $\{A_1, A_2\}$, por exemplo, ela é definida como

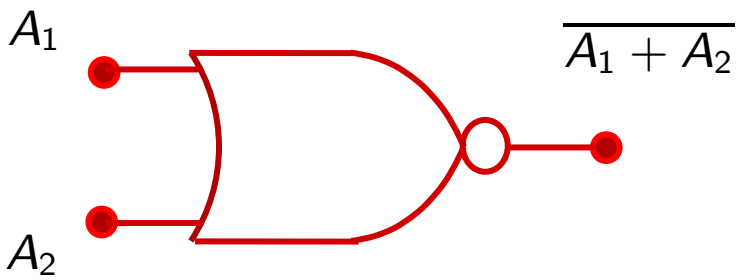
$$f(A_1, A_2) = \overline{A_1 + A_2}$$

e vale 1 apenas se todas as entradas forem iguais a 0.

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	1
0	1	0
1	0	0
1	1	0

Porta lógica



Operações e portas lógicas

Sobre a porta NOR podemos fazer os seguintes comentários :

- Utilizando a tabela verdade, podemos verificar que

$$\overline{A_1 + A_2} = \overline{A_1} \; . \; \overline{A_2}$$

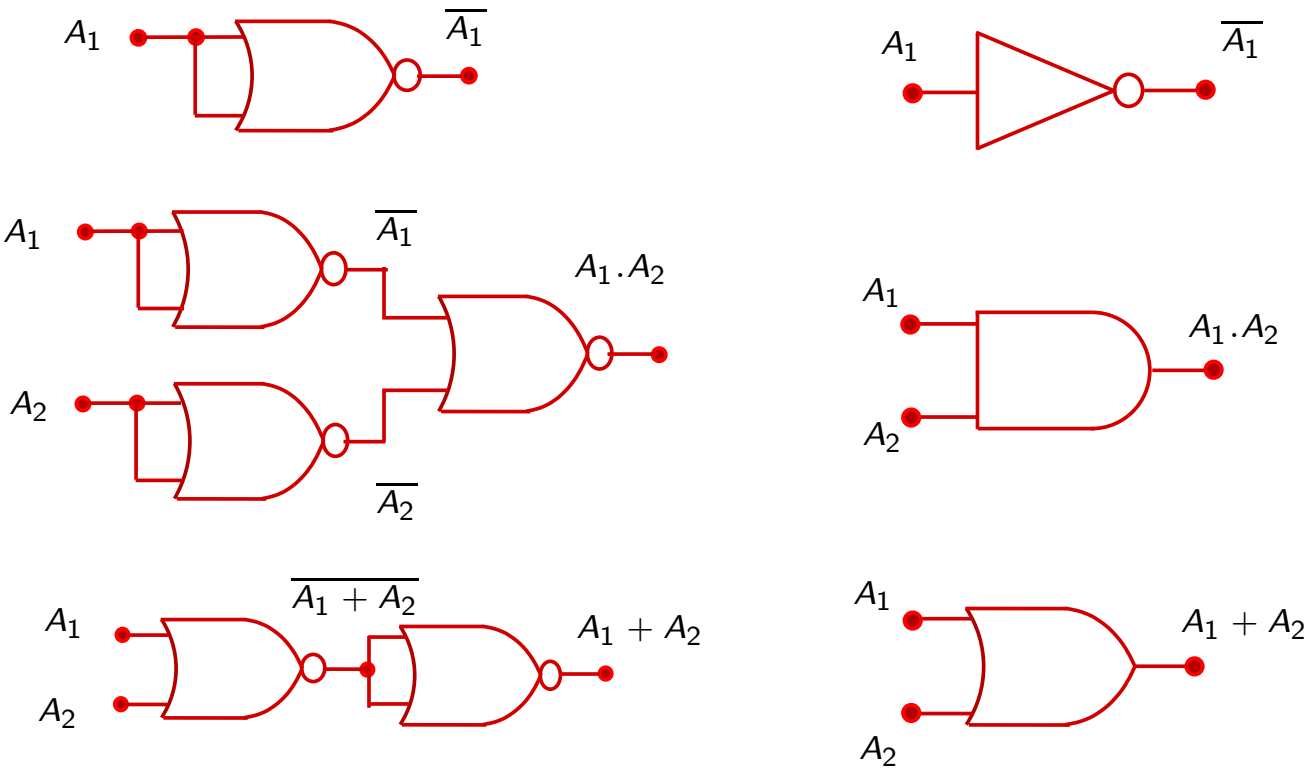
que é um dos resultados do Teorema de De Morgan que veremos posteriormente.

Tabela verdade

A_1	A_2	$\overline{A_1 + A_2}$	$\overline{A_1} \; . \; \overline{A_2}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Operações e portas lógicas

- Utilizando somente portas NOR podemos obter as três portas básicas :



- O que indica a vantagem tecnológica desta porta.

Operações e portas lógicas

- Operação NAND : É a operação AND negada. Para duas entradas $\{A_1, A_2\}$, por exemplo, ela é definida como

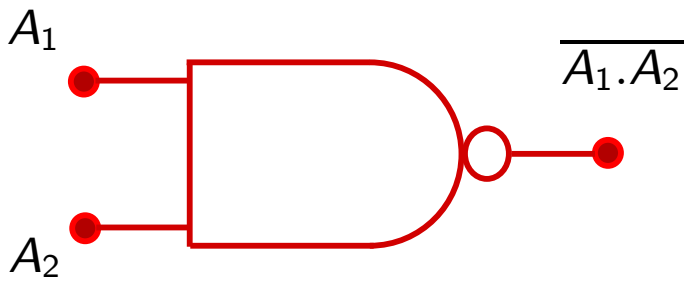
$$f(A_1, A_2) = \overline{A_1.A_2}$$

e vale 0 apenas se todas as entradas forem iguais a 1.

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	1
0	1	1
1	0	1
1	1	0

Porta lógica



Operações e portas lógicas

Sobre a porta NAND podemos fazer os seguintes comentários :

- Utilizando a tabela verdade, podemos verificar que

$$\overline{A_1.A_2} = \overline{A_1} + \overline{A_2}$$

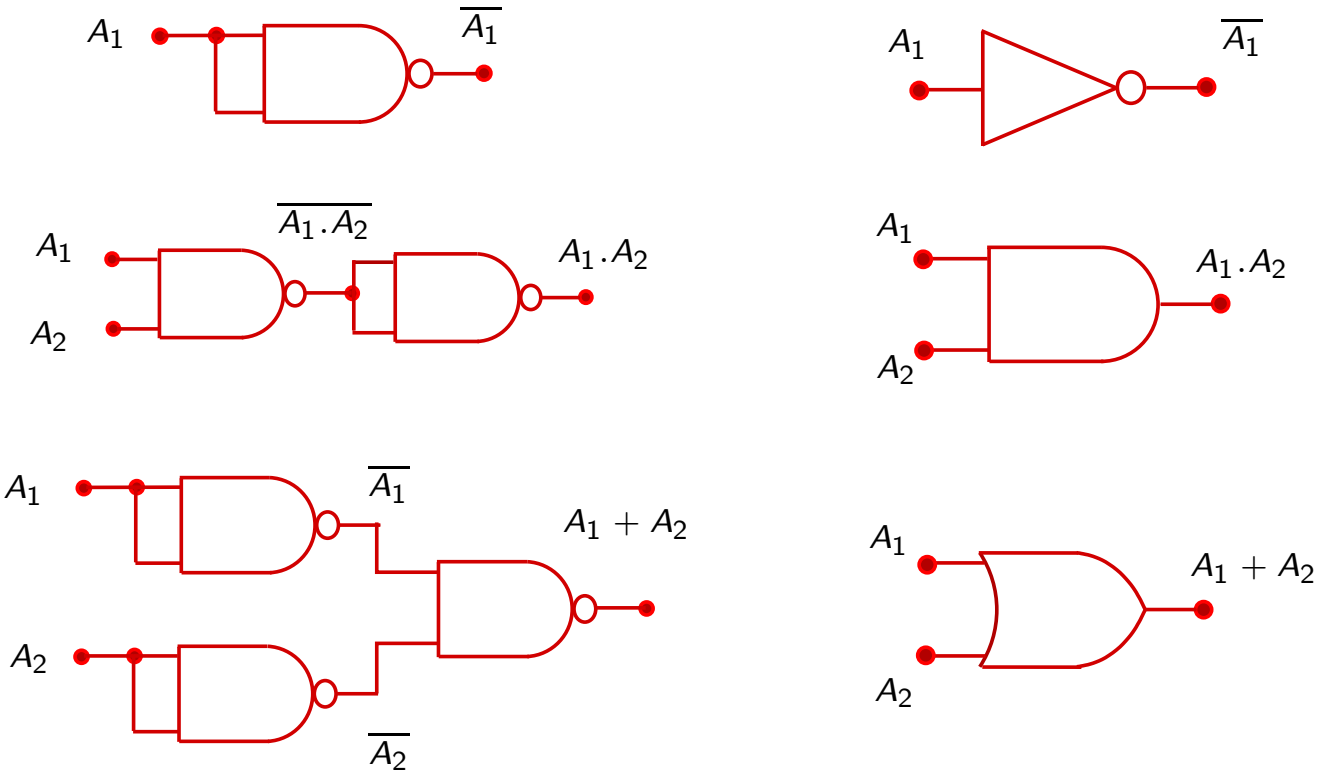
que é um dos resultados do Teorema de De Morgan.

Tabela verdade

A_1	A_2	$\overline{A_1.A_2}$	$\overline{A_1} + \overline{A_2}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Operações e portas lógicas

- Utilizando somente portas NAND podemos obter as três portas básicas :



- O que indica a vantagem tecnológica desta porta.

Operações e portas lógicas

- Operação XOR (ou exclusivo) : Definida apenas para duas entradas $\{A_1, A_2\}$ como sendo

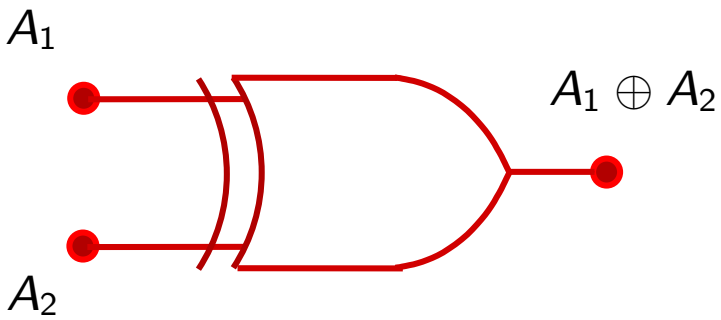
$$\begin{aligned} f(A_1, A_2) &= \overline{A_1} \,.\, A_2 + A_1 \,.\, \overline{A_2} \\ &= A_1 \oplus A_2 \end{aligned}$$

e vale 1 apenas se as entradas forem diferentes.

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Porta lógica



Operações e portas lógicas

- Operação XNOR (coincidência) : Definida apenas para duas entradas $\{A_1, A_2\}$ como sendo

$$\begin{aligned} f(A_1, A_2) &= \overline{A_1} \,.\, \overline{A_2} + A_1 \,.\, A_2 \\ &= A_1 \odot A_2 \end{aligned}$$

e vale 1 apenas se as entradas forem iguais.

Tabela verdade

A_1	A_2	$f(A_1, A_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Porta lógica

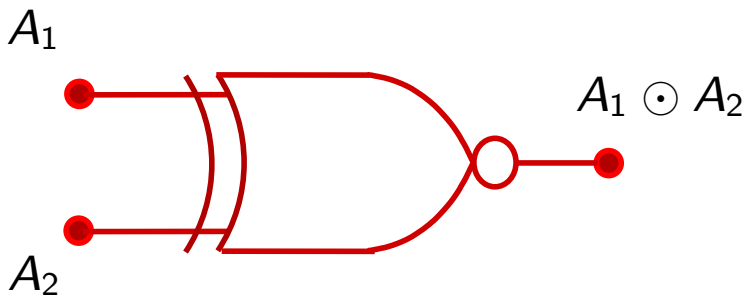


Tabela verdade e expressão lógica

- Dado um circuito lógico, sua expressão booleana é obtida facilmente a partir da simples leitura das portas lógicas. Da mesma maneira, dada uma expressão lógica, a obtenção do circuito é feita termo a termo utilizando as portas correspondentes.
- Vamos discutir agora como obter a expressão lógica a partir da sua tabela verdade.
- Existem duas maneiras : através de somas de produtos (mintermos) ou através de produtos de somas (maxtermos). No curso será adotado a primeira maneira, tendo em vista o método de simplificação que adotaremos e que será apresentado posteriormente.

Tabela verdade e expressão lógica

Para exemplificar as duas formas, vamos considerar a tabela verdade da função NAND.

A	B	$f(\cdot)$	mintermos	maxtermos
0	0	1	$\bar{A}\bar{B}$	$A+B$
0	1	1	$\bar{A}B$	$A+\bar{B}$
1	0	1	$A\bar{B}$	$\bar{A}+B$
1	1	0	AB	$\bar{A}+\bar{B}$

- Somas de produtos (mintermos) :

$$\begin{aligned} f(\cdot) &= 1.(\bar{A}\bar{B}) + 1.(\bar{A}B) + 1.(A\bar{B}) + 0.(AB) \\ &= \bar{A}\bar{B} + \bar{A}B + A\bar{B} \end{aligned}$$

- Produtos de somas (maxtermos) :

$$\begin{aligned} f(\cdot) &= \{1+(A+B)\}.\{1+(A+\bar{B})\}.\{1+(\bar{A}+B)\}.\{0+(\bar{A}+\bar{B})\} \\ &= \bar{A}+\bar{B} \end{aligned}$$

Tabela verdade e expressão lógica

- Podemos observar que ambas as expressões

$$\begin{aligned} \text{Utilizando mintermos} &: \bar{A}\bar{B} + \bar{A}B + A\bar{B} \\ \text{Utilizando maxtermos} &: \bar{A} + \bar{B} = \overline{AB} \end{aligned}$$

  são equivalentes, pois designam a porta lógica NAND. Entretanto, a primeira é escrita utilizando 3 portas lógicas com duas conexões cada e a segunda utiliza uma única porta lógica, a NAND de duas conexões.
- Logo, uma mesma tabela pode representar circuitos equivalentes. Estamos interessados naquele com expressão mínima.
- A seguir, estudaremos duas técnicas de minimização
  - Álgebra booleana.
  - Mapa de Karnaugh.

Álgebra de Boole

- As regras operacionais de minimização utilizando a álgebra de Boole decorrem dos postulados e propriedades a seguir :
  - Postulados da complementação

$$\overline{\overline{A}} = A$$

  - Postulados da adição

$$A + 0 = A,\ A + 1 = 1,\ A + A = A,\ A + \overline{A} = 1$$

  - Postulados da multiplicação

$$A \,.\, 0 = 0,\ A \,.\, 1 = A,\ A \,.\, A = A,\ A \,.\, \overline{A} = 0$$

  - Propriedades : Comutativa, associativa e distributiva são válidas para a adição e a multiplicação.

Teorema de De Morgan

- O seguinte teorema é importante pois permite simplificar expressões booleanas $\Longrightarrow$ minimização.

Teorema de De Morgan

As seguintes igualdades são verdadeiras :

- $\overline{A \,.\, B \,.\, C \,.\, \cdots \,.\, N} = \overline{A} + \overline{B} + \cdots + \overline{N}$
- $\overline{A + B + C + \cdots + N} = \overline{A} \,.\, \overline{B} \,.\, \cdots \,.\, \overline{N}$

- Exemplo 1 : Minimize a expressão sem utilizar o teorema.

$$\begin{aligned} \bar{A}\bar{B} + \bar{A}B + A\bar{B} &= \bar{A}(B + \bar{B}) + A\bar{B} \\ &= \bar{A}(1 + \bar{B}) + A\bar{B} \\ &= \bar{A} + (A + \bar{A})\bar{B} \\ &= \bar{A} + \bar{B} \end{aligned}$$

Teorema de De Morgan

- Exemplo 2 : Minimize a mesma expressão utilizando o teorema de De Morgan.

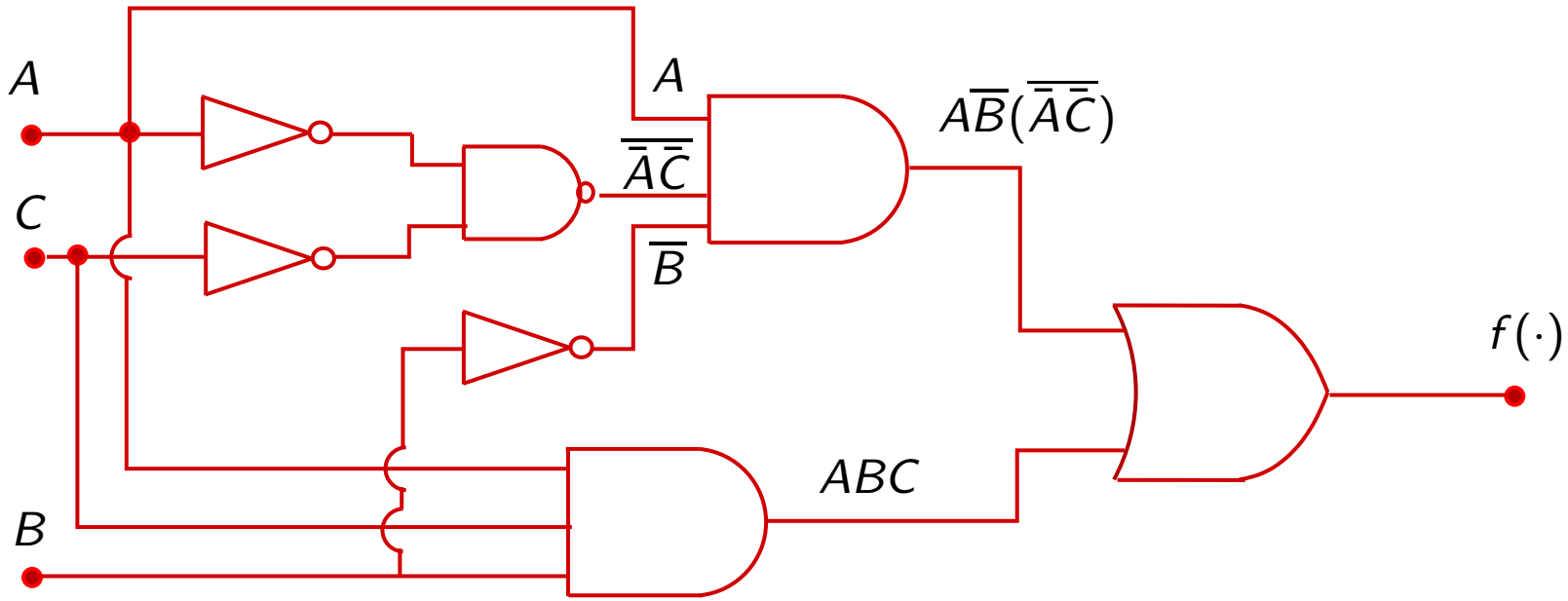
$$\begin{aligned}
\bar{A}\bar{B} + \bar{A}B + A\bar{B} &= \bar{A}(B + \bar{B}) + A\bar{B} \\
&= \overline{\overline{\bar{A} + A\bar{B}}} \\
&= \overline{A.(\bar{A} + B)} \\
&= \overline{AB} \\
&= \overline{A} + \overline{B}
\end{aligned}$$

- Exemplo 3 : Minimize a seguinte expressão

$$\begin{aligned}
ABC + A\overline{B} + A\overline{C} &= A(BC + \overline{B} + \overline{C}) \\
&= A(BC + \overline{\overline{\overline{B} + \overline{C}}}) \\
&= A(BC + \overline{BC}) \\
&= A
\end{aligned}$$

Exercício

- Descreva a expressão lógica que representa o circuito a seguir



- A expressão lógica é dada por

$$f(A,B,C) = A \,.\, B \,.\, C + A \,.\, \overline{B} \,.\, (\overline{\overline{A} \,.\, \overline{C}})$$

Simplificação algébrica

Minimizar a expressão de um circuito lógico, significa obter uma outra equivalente com menos termos e operações. Isto implica em menos portas lógicas e conexões.

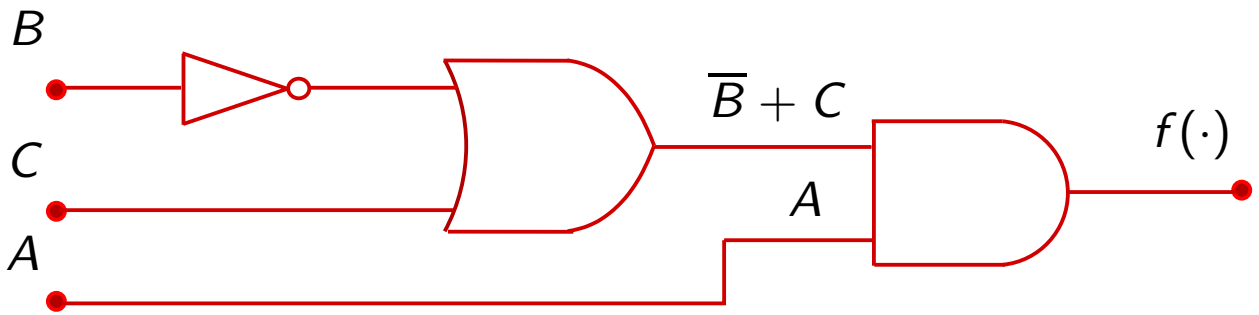
- Como vimos, podemos usar a álgebra de Boole para realizar a minimização.
- Neste caso, a simplificação nem sempre é óbvia.
- Geralmente, podemos seguir dois passos essenciais :
  - colocar a expressão na forma de soma de produtos
  - identificar fatores comuns e realizar a fatoração
- Algumas vezes devemos contar com habilidade e experiência para obter uma boa simplificação.

Simplificação algébrica

- Utilizando a álgebra de Boole, podemos minimizar a expressão da função do exercício anterior

$$\begin{aligned} f(A,B,C) &= A \,.\, B \,.\, C + A \,.\, \overline{B} \,.\, (\overline{\overline{A} \,.\, \overline{C}}) \\ &= A \,.\, B \,.\, C + A \,.\, \overline{B} \,.\, (A + C) \\ &= A \,.\, C \,.\, (B + \overline{B}) + A \,.\, \overline{B} \\ &= A \,.\, (C + \overline{B}) \end{aligned}$$

- O circuito lógico simplificado é dado por.



A quantidade de portas lógicas foi reduzida de 7 para 3!!!

Exercício

- A partir do circuito apresentado anteriormente, obtenha a sua tabela verdade e, a partir dela, obtenha a expressão lógica.

A	B	C	$f(A,B,C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

- Utilizando a tabela, sua expressão lógica é dada por

$$f(A,B,C) = A \,.\, \overline{B} \,.\, \overline{C} \;+\; A \,.\, \overline{B} \,.\, C \;+\; A \,.\, B \,.\, C$$

embora seja equivalente à função obtida através do circuito, ela possui um número maior de termos.

Simplificação algébrica

- Minimizando a expressão, temos

$$\begin{aligned} f(A,B,C) &= A \,.\, \overline{B} \,.\, \overline{C} \;+\; A \,.\, \overline{B} \,.\, C \;+\; A \,.\, B \,.\, C \\ &= A \,.\, \overline{B} \,.\, (\overline{C} + C) \;+\; A \,.\, B \,.\, C \\ &= A \,.\, (\overline{B} \;+\; B \,.\, C) \end{aligned}$$

que, como sabemos, não é a expressão mínima. De fato, fazendo um passo adicional, nem sempre óbvio, obtemos

$$\begin{aligned} f(A,B,C) &= A \,.\, (\overline{B} \;+\; B \,.\, C) \\ &= A \,.\, (\overline{B} \,.\, (1 \;+\; C) \;+\; B \,.\, C) \\ &= A \,.\, (\overline{B} \;+\; C \,.\, (\overline{B} + B)) \\ &= \underbrace{A \,.\, (\overline{B} \;+\; C)}_{\text{expressão mínima}} \end{aligned}$$

Simplificação via mapa de Karnaugh

- O mapa de Karnaugh é um método gráfico sistemático para simplificar expressões booleanas.
- Por ser um método procedimental ele não depende da habilidade do projetista com as regras da álgebra booleana.
- Ele converte a tabela verdade na função booleana minimizada.
- Sua utilidade prática está restrita a problemas com até cinco variáveis.
- Para mais de cinco variáveis utiliza-se o método de Quine-McCluscky que funcionalmente é similar ao de Karnaugh, mas é tabular e, portanto, mais eficiente para ser processado pelo computador.
- A seguir estudaremos a simplificação via mapa de Karnaugh de duas a cinco variáveis de entrada.

Simplificação via mapa de Karnaugh

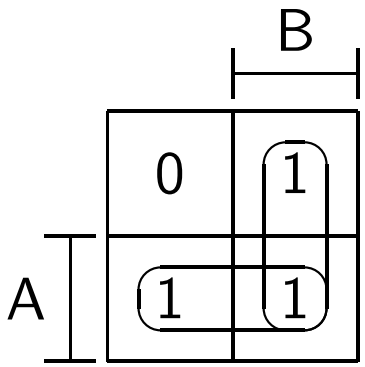
- Para n variáveis, o mapa apresenta 2^n posições, cada uma representando uma situação da entrada. Ademais, o valor de cada posição "0" ou "1" representa a saída correspondente.

- Como o mapa utiliza a forma de soma de produtos na simplificação, geralmente, expressa-se somente as posições em que $f(\cdot) = 1$.

Simplificação via mapa de Karnaugh

- Para duas variáveis :

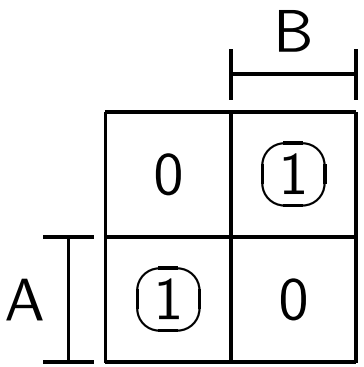
A	B	f(A,B)
0	0	0
0	1	1
1	0	1
1	1	1



$$f(A,B) = \overline{A} \,.\, B + A \,.\, \overline{B} + A \,.\, B$$
$$= A + B$$

Simplificação via mapa de Karnaugh

- Na simplificação agrupamos as regiões $f(A, B) = 1$ para obter o menor número de agrupamentos.
- Os agrupamentos devem formar quadrados ou retângulos maiores possíveis.
- O número de elementos em cada agrupamento deve ser uma potência de 2.
- O mapa de Karnaugh



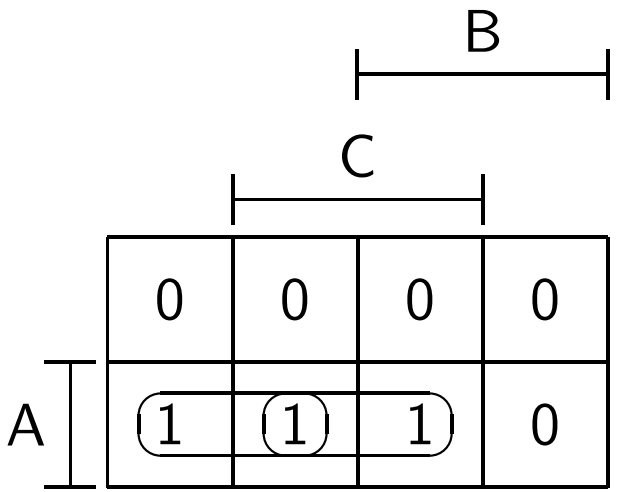
$$f(A, B) = \underbrace{\overline{A} \,.\, B + A \,.\, \overline{B}}_{XOR}$$

não admite simplificação ! !

Simplificação via mapa de Karnaugh

- Para três variáveis :

A	B	C	f(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

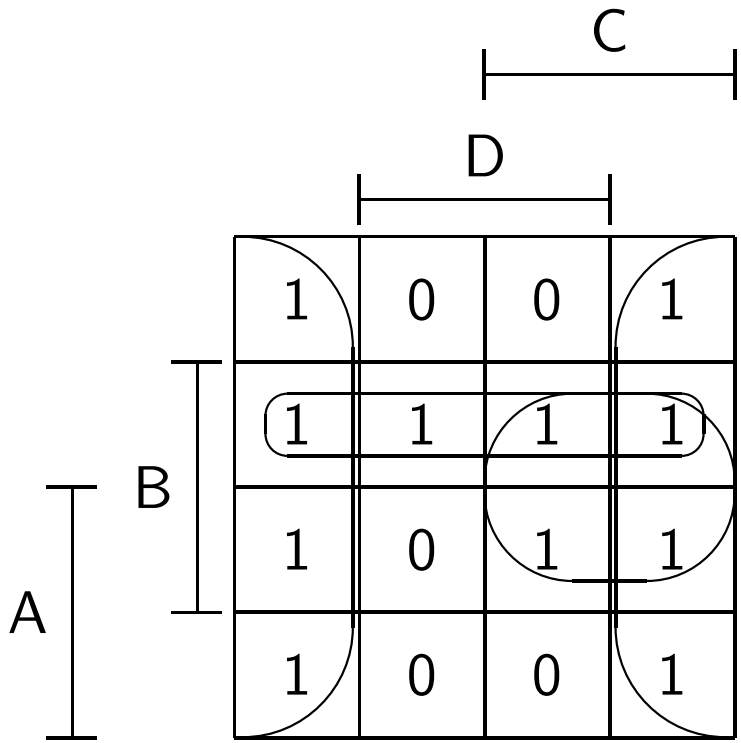


$$f(A,B,C) = A \,.\, \overline{B} + A \,.\, C$$
$$= A \,.\, (\overline{B} + C)$$

Simplificação via mapa de Karnaugh

- Para quatro variáveis :

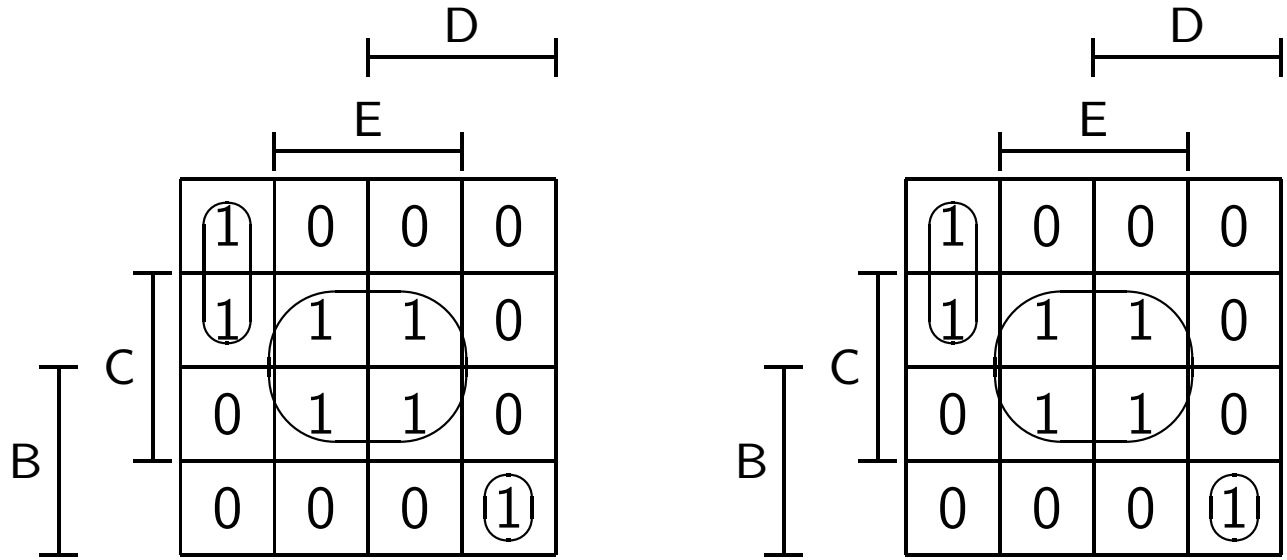
A	B	C	D	f(A,B,C,D)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



$$f(A,B,C,D) = \overline{D} + B \;.\; C + \overline{A} \;.\; B$$

Simplificação via mapa de Karnaugh

- Para cinco variáveis :



$\overline{A}$ A

$$f(A,B,C,D,E) = C\ .E + \overline{B}\ .\ \overline{D}\ .\ \overline{E} + B\ .\ \overline{C}\ .\ D\ .\ \overline{E}$$

Aplicação prática : Controle da porta de um elevador

- Em um prédio de três andares deseja-se projetar um circuito lógico para controlar a abertura da porta de um elevador. As variáveis de entrada são A, B, C, D em que :
  - A indica que o elevador está em movimento quando igual a 1.
  - B, C, D indicam que o elevador está posicionado nos andares $1, 2, 3$ quando iguais a 1, respectivamente.
- Projete a saída Ab que indica, quando em nível alto, que o elevador deve abrir a porta. Para isto :
  - Determine a tabela verdade do problema.
  - Faça simplificações utilizando um dos métodos estudados.
  - Desenhe o circuito lógico correspondente.

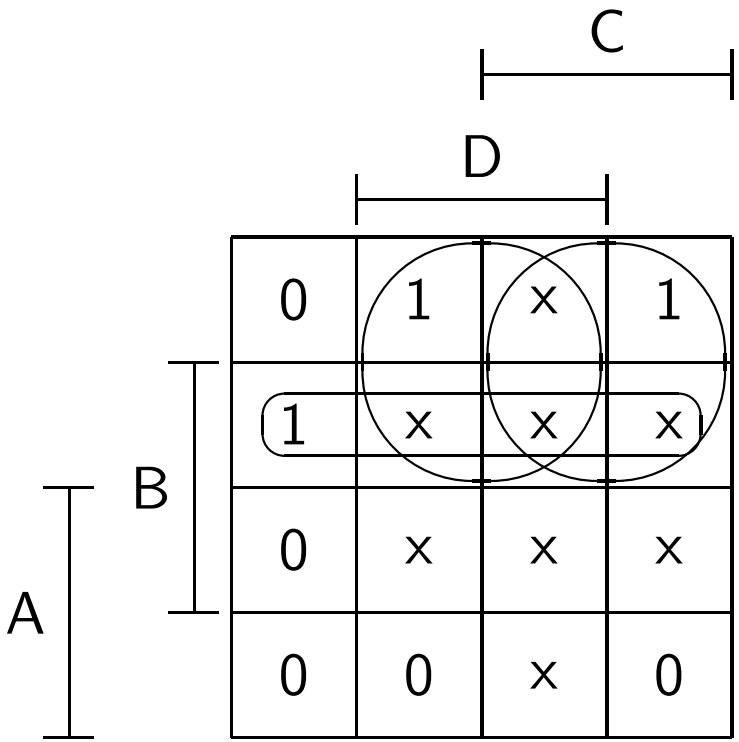
Aplicação prática : Controle da porta de um elevador

Tabela verdade :

A	B	C	D	Ab
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Aplicação prática : Controle da porta de um elevador

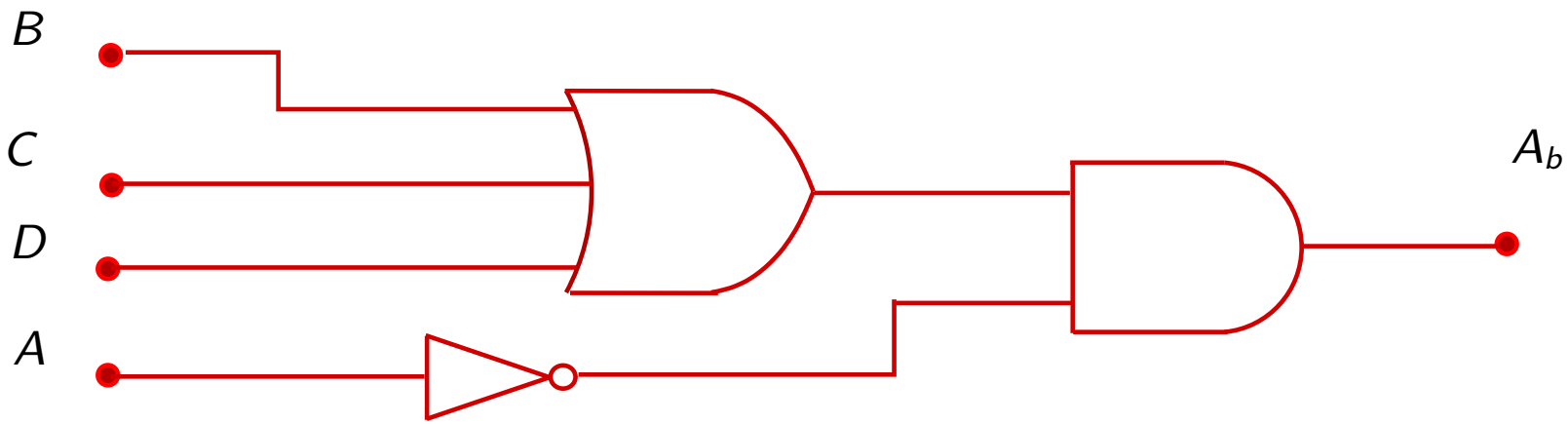
Minimização via mapa de Karnaugh :



$$Ab = \overline{A} \,.\, (B \;+\; C \;+\; D)$$

Aplicação prática : Controle da porta de um elevador

Representação do circuito :



Síntese de circuitos combinacionais

- Vamos agora utilizar os conceitos iniciais apresentados para realizar a síntese de alguns circuitos combinacionais importantes :
  - meio somadores e somadores completos
  - comparadores
  - codificadores e decodificadores
  - multiplexadores e demultiplexadores

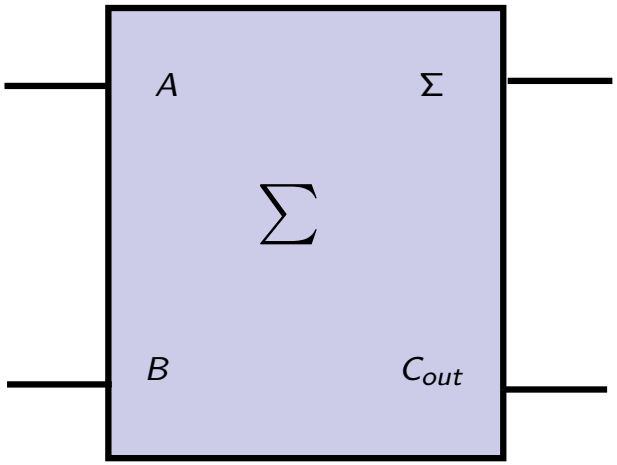
Meio somador

- O meio somador aceita duas variáveis de entrada A e B e possui como saídas a soma Σ e o carry out C_{out}.

Tabela verdade

A	B	Σ	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Meio somador



Meio somador

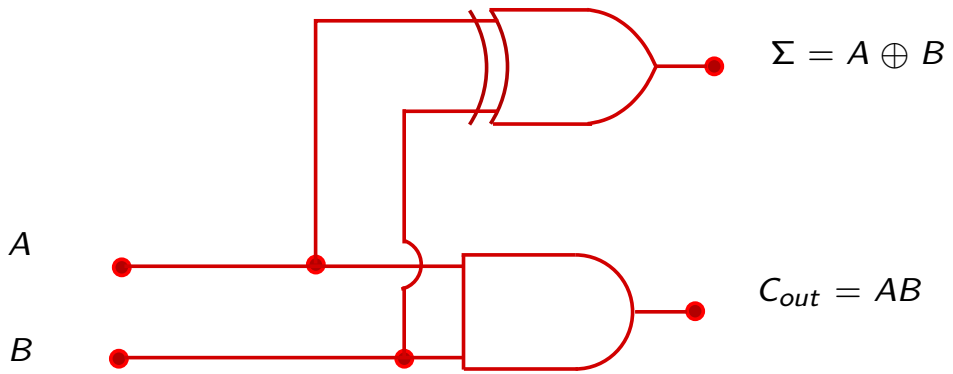
- Não é difícil verificar que

$$\begin{aligned} \Sigma &= \bar{A}B + A\bar{B} \\ &= A \oplus B \end{aligned}$$

e que

$$C_{out} = AB$$

- Seu circuito é dado por



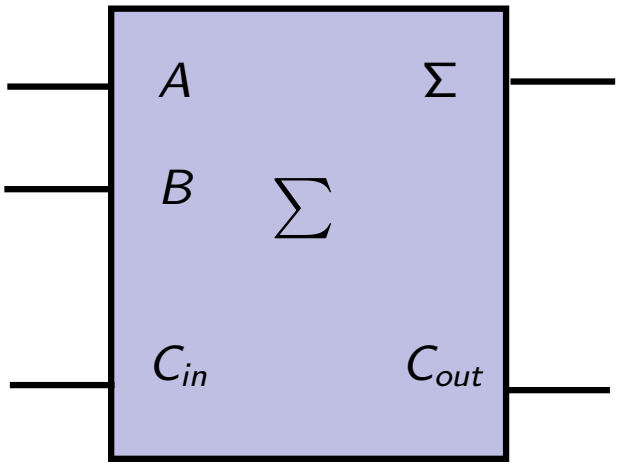
Somador completo

- O somador completo possui como variáveis de entrada A, B e o carry in C_{in} e como variáveis de saída a soma Σ e o carry out C_{out}.

Tabela verdade

A	B	C_{in}	Σ	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Somador completo



Somador completo

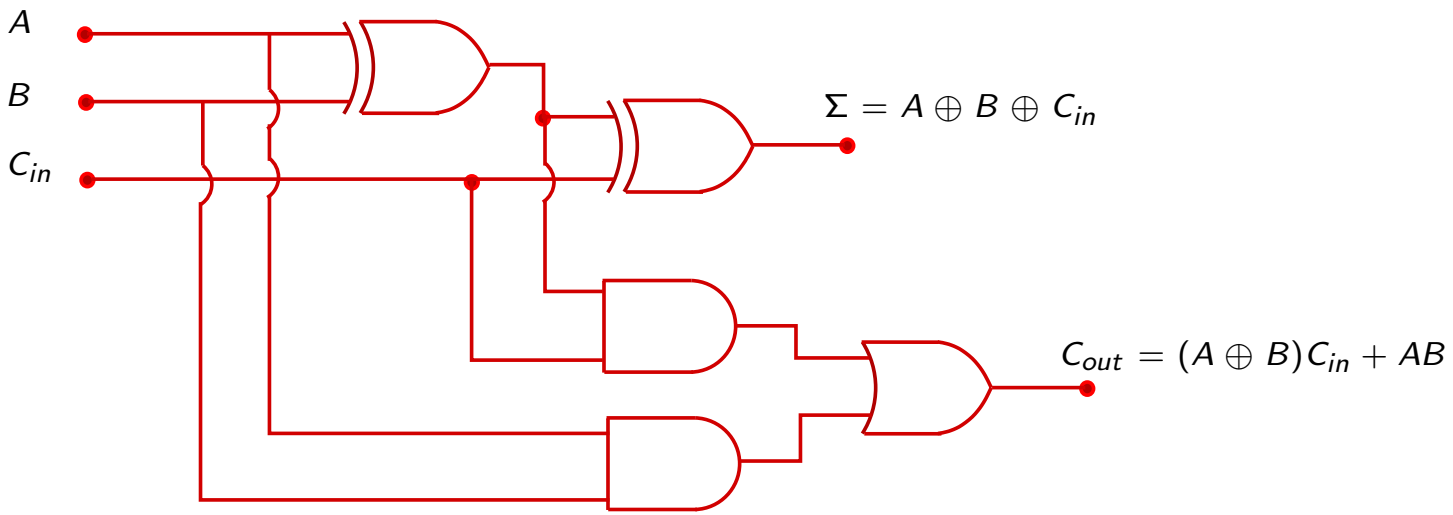
- Podemos verificar que

$$\Sigma = A \oplus B \oplus C_{in}$$

e que

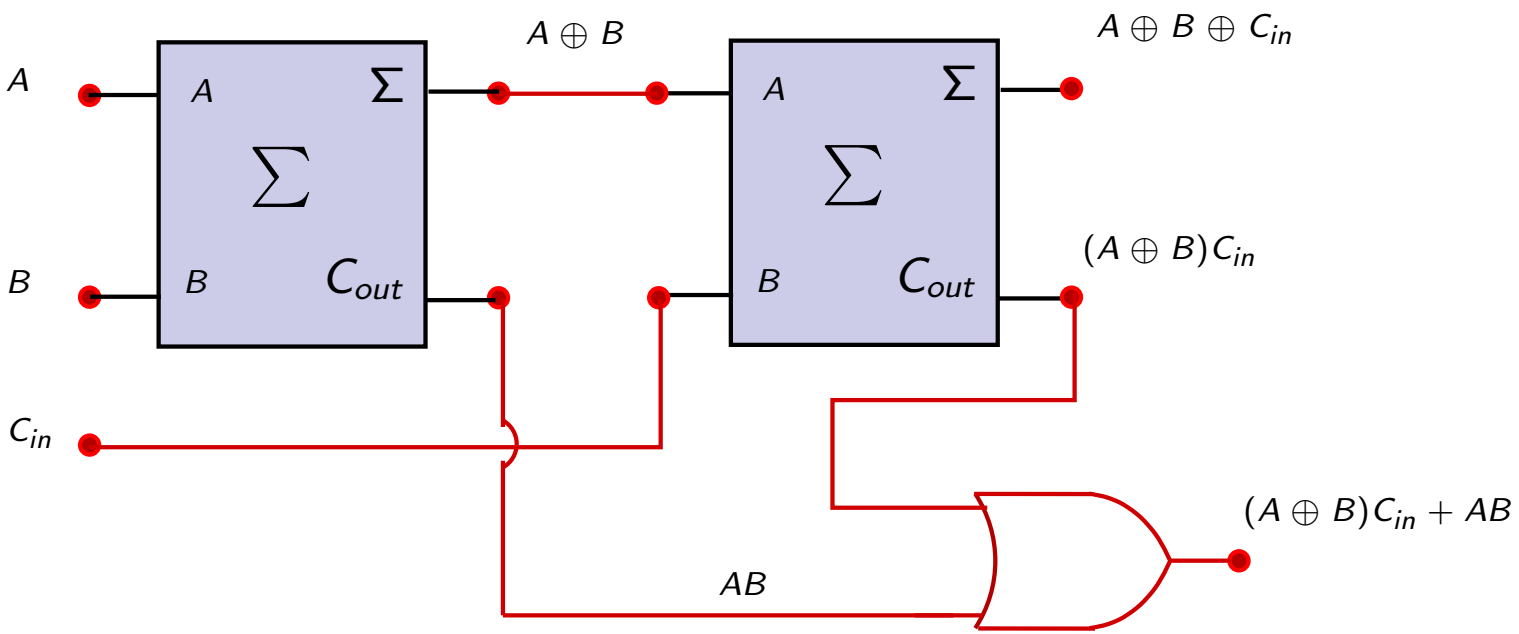
$$\begin{aligned} C_{out} &= \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} \\ &= (A \oplus B)C_{in} + AB \end{aligned}$$

- Seu circuito é dado por



Somador completo

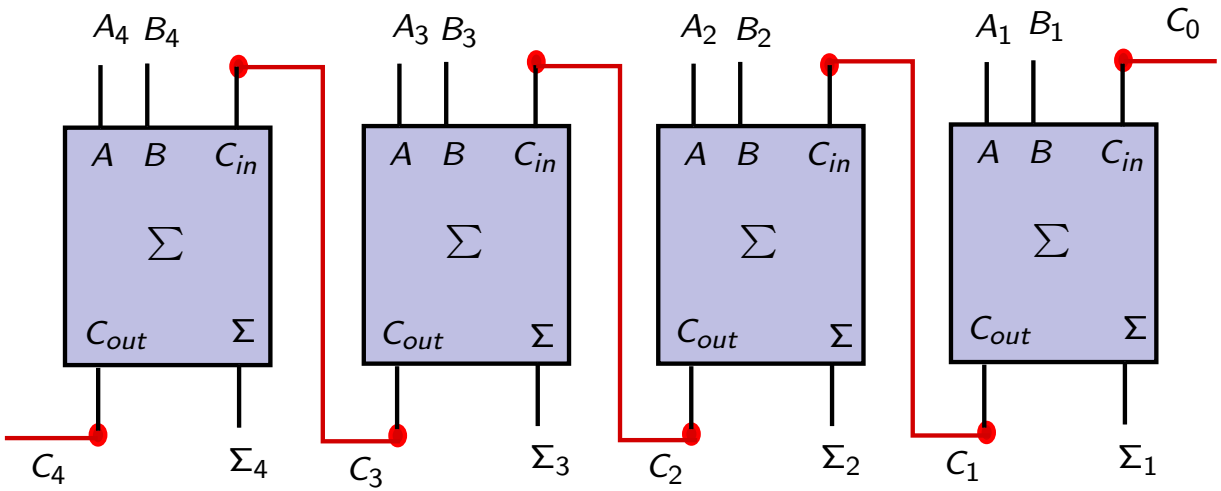
- O somador completo pode ser construído a partir de dois meio somadores.

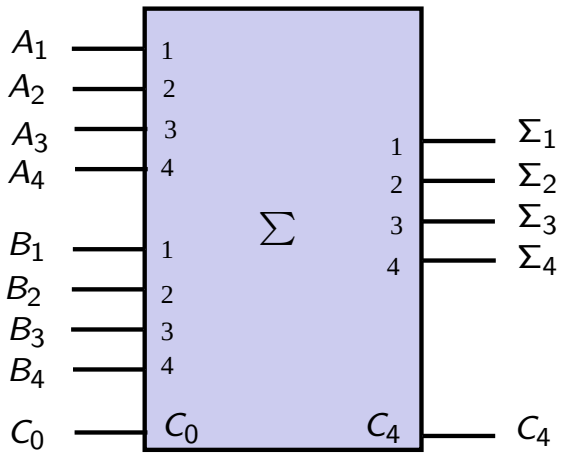


Somador completo

- Para números de 4 bits, um somador paralelo básico está apresentado a seguir.

$$\begin{array}{r} A_4\ A_3\ A_2\ A_1 \\ +B_4\ B_3\ B_2\ B_1 \\ \hline C_4\ S_4\ S_3\ S_2\ S_1 \end{array}$$





- Podemos cascatear os somadores de maneira a considerar palavras maiores.

Somador completo

Como podemos verificar, a saída C_{out} de cada somador completo está conectada à entrada C_{in} do somador seguinte. Desta forma, a soma em cada estágio só pode ser efetuada após o recebimento do carry C_{out} do estágio anterior provocando um atraso de propagação no processo de adição. Para evitar este atraso pode-se projetar um circuito antecipador de carry. De fato, fazendo a minimização, para o somador completo de 1 bit temos

$$\begin{aligned} C_{out} &= (A \oplus B)C_{in} + AB \\ &= (A + B)C_{in} + AB \end{aligned}$$

Somador Completo

Para um somador de 2 bits, definindo $Cgi = A_i B_i$ e $C_{pi} = A_i + B_i$ para o estágio i, temos :

- Primeiro estágio :

$$C_{out1} = C_{g1} + C_{p1} C_{in1}$$

- Segundo estágio :

$$\begin{aligned} C_{in2} &= C_{out1} \\ C_{out2} &= C_{g2} + C_{p2}(C_{g1} + C_{p1} C_{in1}) \end{aligned}$$

e, desta forma, não há atraso de propagação pois todos os carries são calculados no mesmo instante uma vez que todos dependem apenas do primeiro C_{in1}.

Comparador

- A função do comparador é comparar a magnitude de números binários.
- Para comparar a igualdade de dois bits, basta utilizar a porta lógica XNOR, que fornecerá nível lógico alto apenas na igualdade.
- Desta maneira, para comparar se dois números binários, por exemplo, $A = A_3A_2A_1A_0$ e $B = B_3B_2B_1B_0$ são iguais basta agrupar os bits dois a dois da forma $\{A_3, B_3\}$, $\{A_2, B_2\}$, $\{A_1, B_1\}$ e $\{A_0, B_0\}$ e conectá-los, respectivamente, à quatro portas XNORs. As saídas destas portas são conectadas à uma porta AND de quatro entradas. A saída da porta AND terá nível alto somente se os números forem iguais.

Comparador

- Para comparar se dois números são diferentes e detectar qual deles é o maior, basta analisá-los começando com o bit mais significativo. Por exemplo, para dois números binários $A = A_3A_2A_1A_0$ e $B = B_3B_2B_1B_0$, o procedimento a seguir é realizado.
  - Se $A_3 = 1$ e $B_3 = 0$ então $A > B$.
  - Se $A_3 = 0$ e $B_3 = 1$ então $A < B$.
  - Se $A_3 = B_3$ realizam-se as verificações anteriores para o bit consecutivo menos significativo.

Codificador e decodificador

- Os circuitos codificadores e decodificadores são aqueles que efetuam a passagem de um código para outro.
- O circuito codificador torna possível a passagem de um código conhecido para um desconhecido. Exemplo : o circuito inicial de uma calculadora que transforma decimal (nossa linguagem) para binário (linguagem da máquina).
- O circuito decodificador faz o inverso, ou seja, transforma um código desconhecido em outro conhecido.
- É claro, que o termo codificador ou decodificador depende do referencial que estamos considerando. Se estivermos considerando a máquina como referencial o raciocínio é inverso.

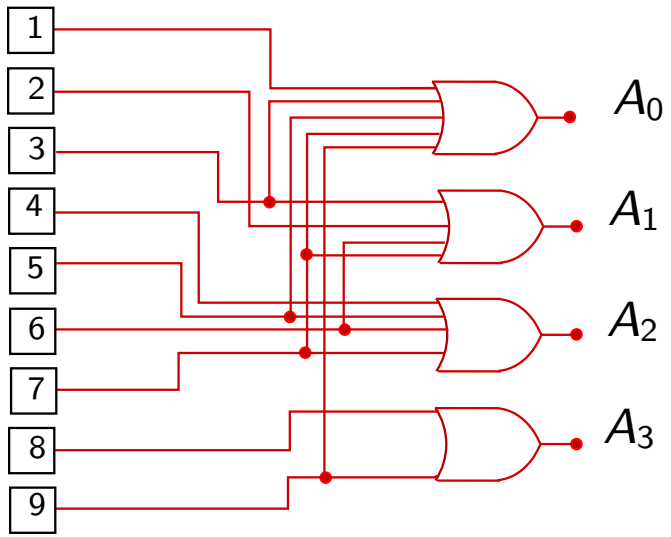
Codificador

- Um exemplo de circuito codificador é aquele que passa de decimal para BCD. Neste caso, temos 10 entradas e 4 saídas.

Tabela verdade

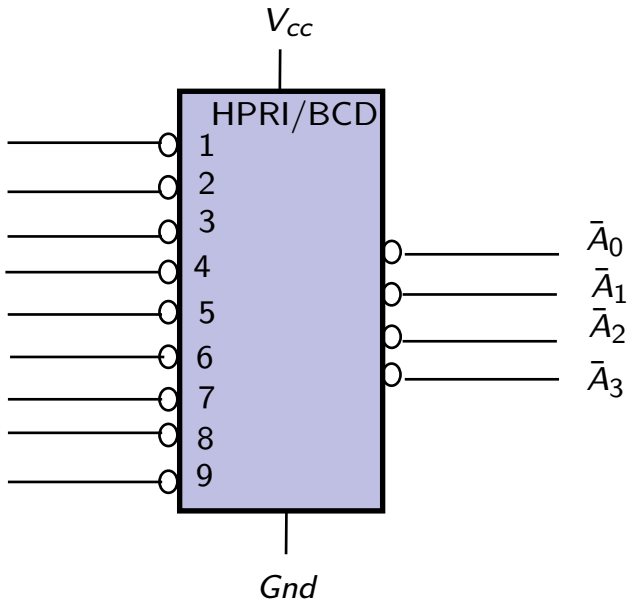
Decimal	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Circuito decimal/BCD



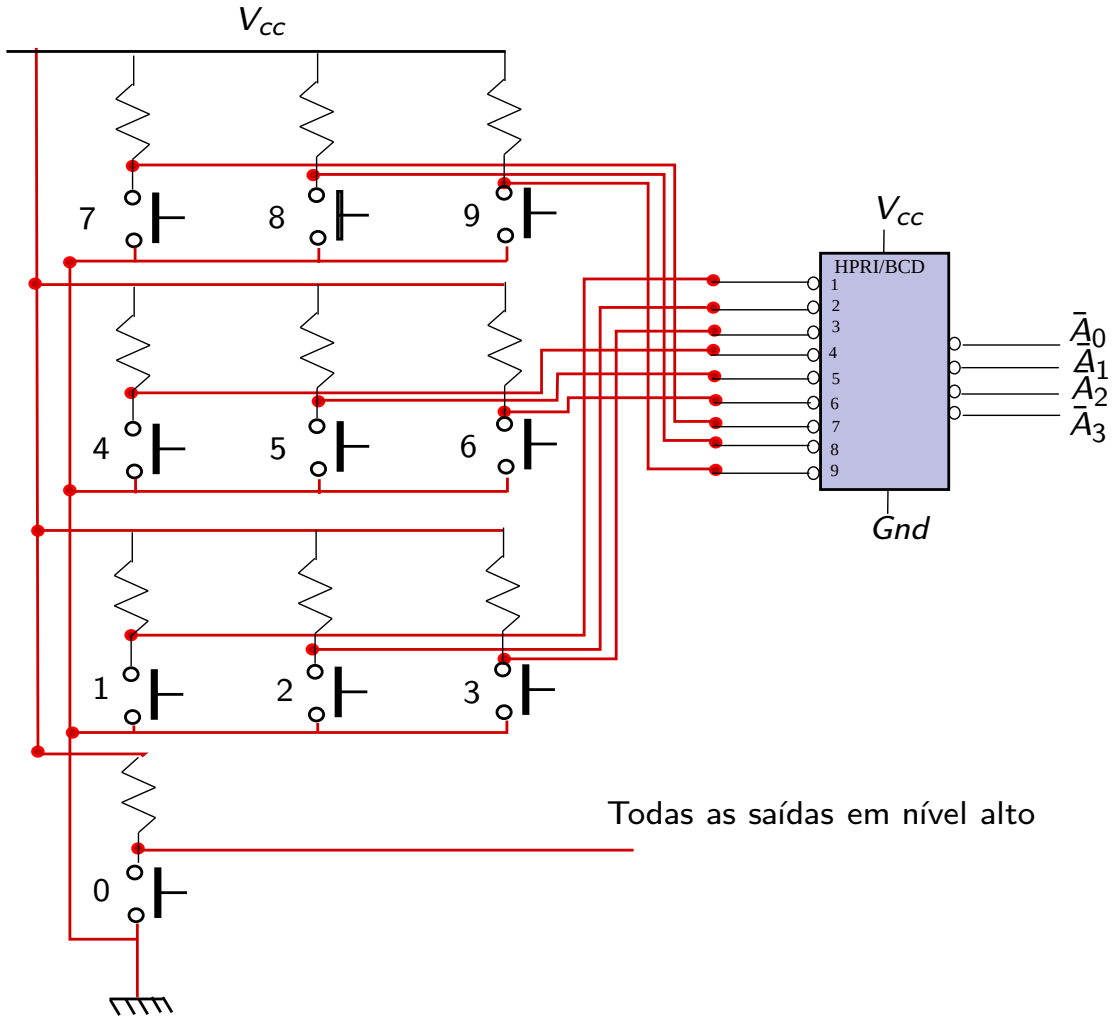
Codificador

- Existe ainda o codificador decimal para BCD com prioridade. Neste caso, se dois números decimais forem acionados, o codificador fornecerá o código BCD do maior deles.
- A figura a seguir apresenta o chip 74HC147 que é um codificador decimal para BCD com prioridade. Neste chip as entradas e as saídas são ativas em nível baixo.



Codificador

- Exemplo de um teclado. Entradas e saídas ativas em nível baixo.



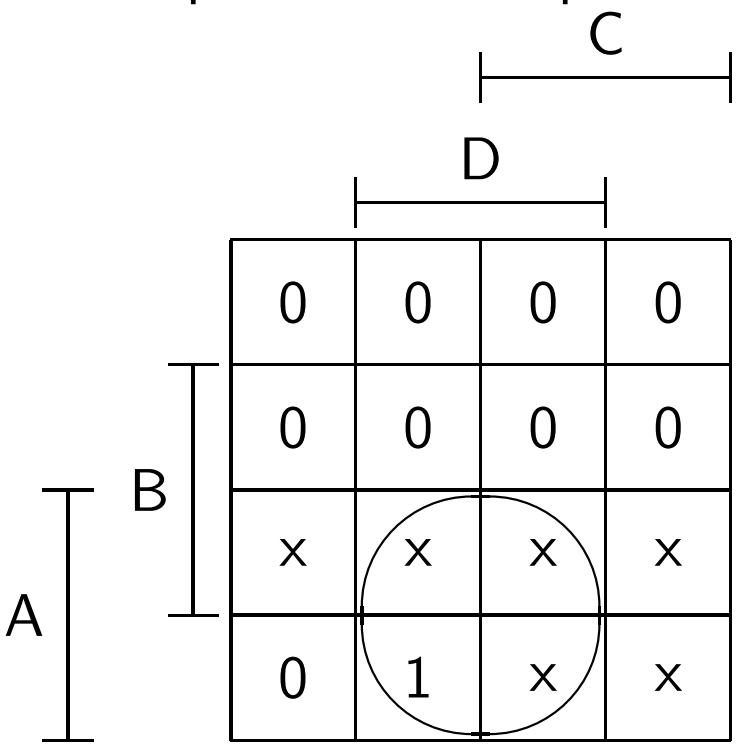
Decodificador

- Segue um exemplo de decodificador BCD para decimal. Ele possui 4 variáveis de entrada e 10 variáveis de saída relacionadas como na tabela a seguir.

A	B	C	D	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

Decodificador

- Como o código BCD não possui valores maiores do que 9, para fins de simplificação, o "don't care" é utilizado nas possibilidades excedentes.
- Para cada dígito fazemos o mapa de Karnaugh e simplificamos a expressão. Para o dígito 9 (D_9), temos



$$D_9 = A \,.\, D$$

Decodificadores

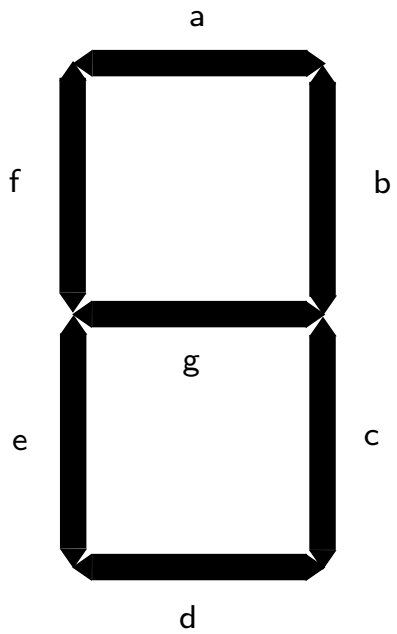
- Procedendo com a simplificação para os demais dígitos, obtemos o seguinte resultado.

$$
\begin{aligned}
D_8 &= A \,.\, \bar{D} \\
D_7 &= B \,.\, C \,.\, D \\
D_6 &= B \,.\, C \,.\, \bar{D} \\
D_5 &= B \,.\, \bar{C} \,.\, D \\
D_4 &= B \,.\, \bar{C} \,.\, \bar{D} \\
D_3 &= \bar{B} \,.\, C \,.\, D \\
D_2 &= \bar{B} \,.\, C \,.\, \bar{D} \\
D_1 &= \bar{A} \,.\, \bar{B} \,.\, \bar{C} \,.\, D \\
D_0 &= \bar{A} \,.\, \bar{B} \,.\, \bar{C} \,.\, \bar{D}
\end{aligned}
$$

Verifique !

Decodificador

- O display de 7 segmentos nos permite escrever números de 0 a 9 e algumas letras ou sinais. A figura a seguir apresenta uma unidade genérica do display com sua nomenclatura de identificação.



Decodificador

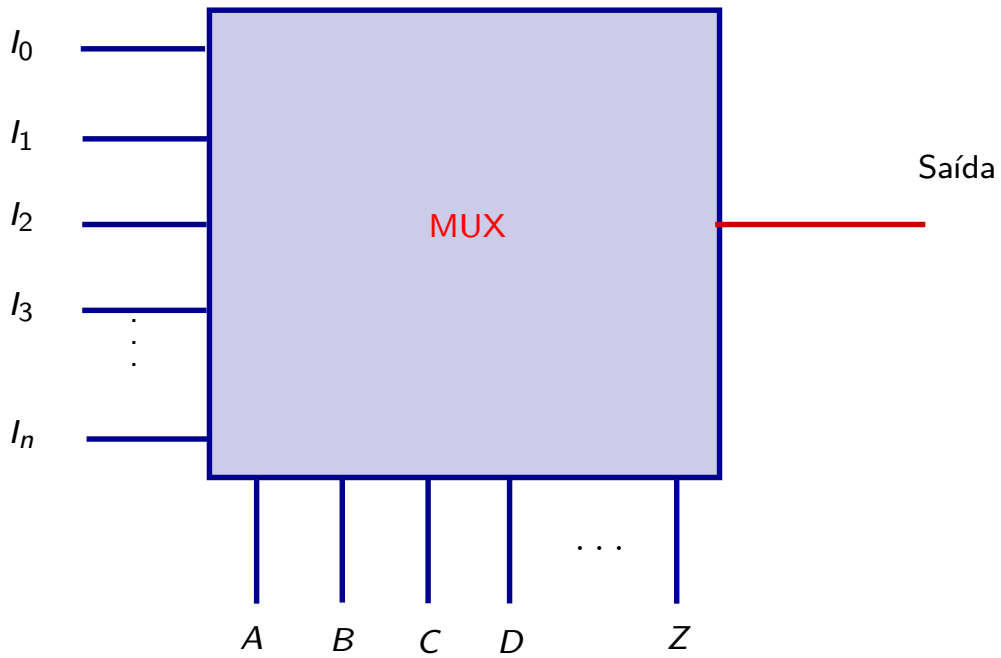
- Elaboração de um decodificador de código BCD para display de 7 segmentos.

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Simplique e apresente o circuito correspondente !

Multiplexadores

- Os multiplexadores são circuitos que permitem passar uma informação digital proveniente de diversos canais em um só canal. Eles também são chamados de selecionadores de dados.
- A Figura a seguir apresenta o esquema de um multiplexador.



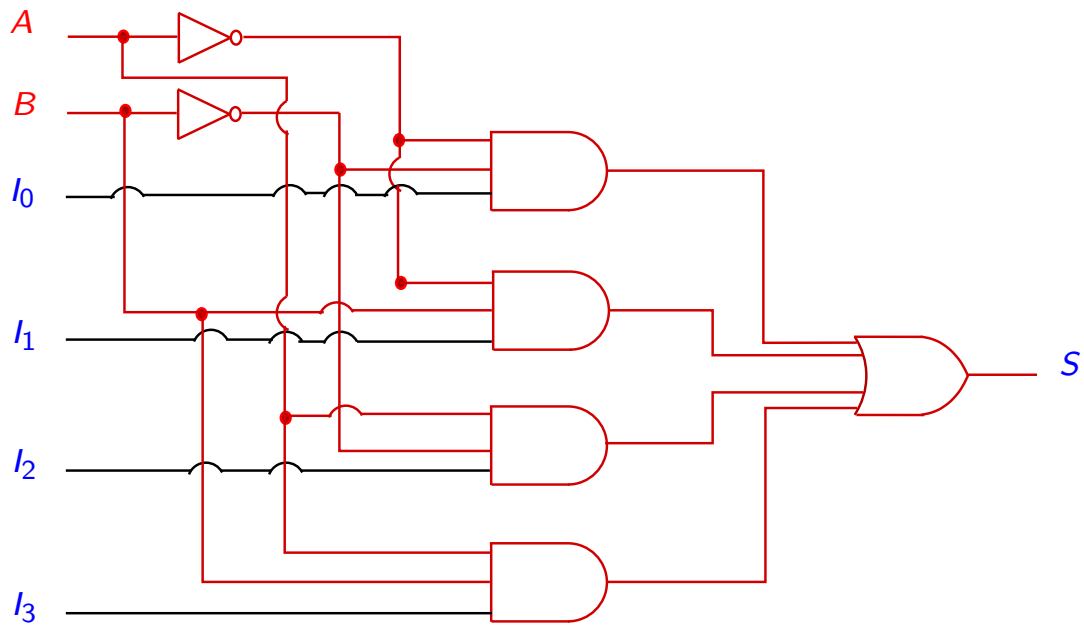
Multiplexadores

- Vamos supor que temos 4 linhas de informações e apenas uma linha de transmissão. Neste caso o selecionador possui 2 bits e seu circuito está apresentado a seguir.

Tabela verdade

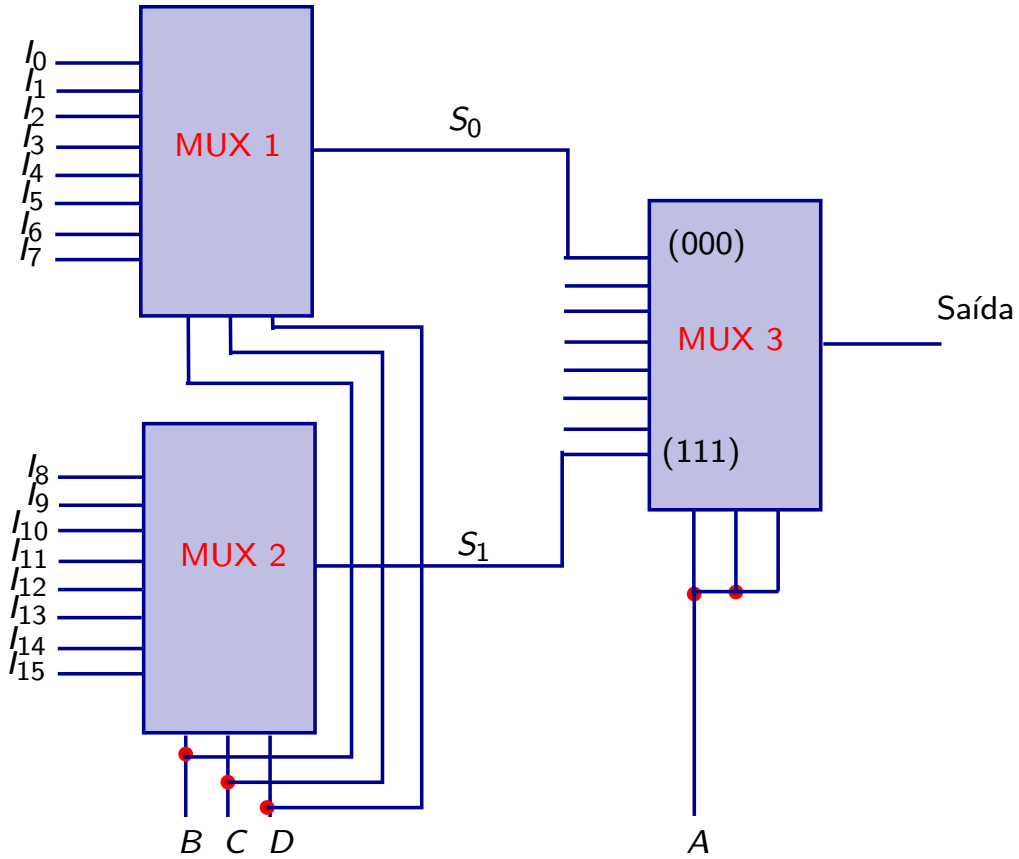
A	B	S
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Multiplexador de 4 entradas



Multiplexadores

- Para ampliarmos a capacidade de um multiplexador podemos cascatear outros de menor capacidade. Exemplo : Multiplex de 16 canais a partir de multiplex de 8 canais.



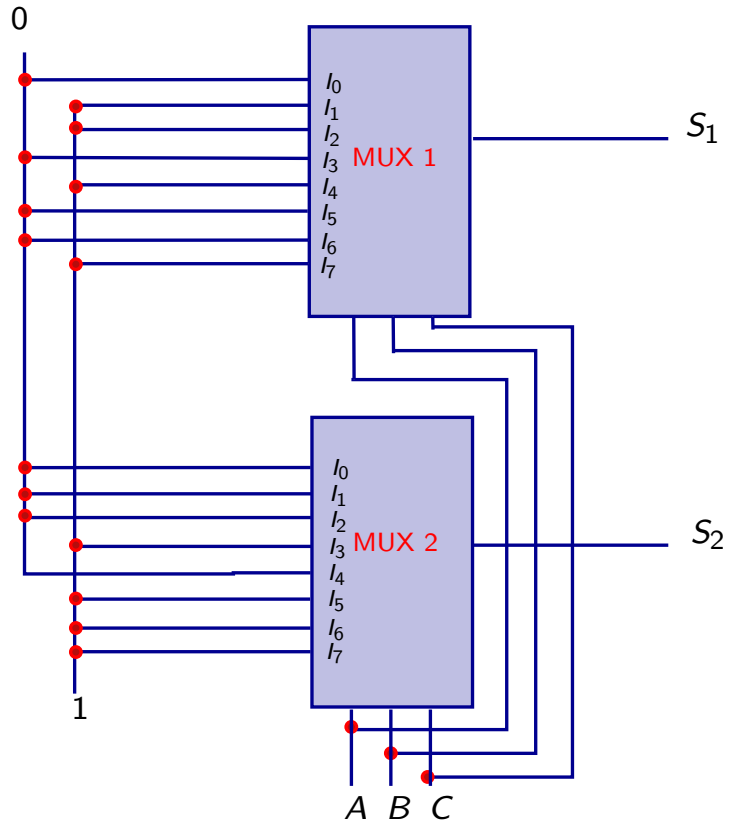
Multiplexadores

- Podemos também utilizar o multiplexador para construir um circuito combinacional qualquer como apresentado a seguir.

Tabela verdade

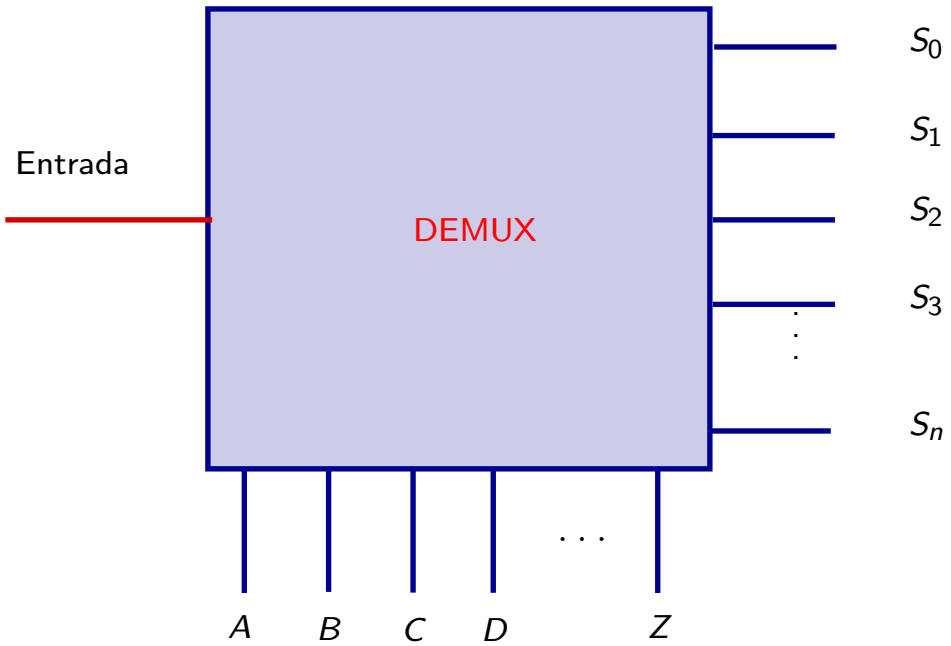
Variáveis de seleção			MUX 1	MUX 2
A	B	C	S_1	S_2
0	0	0	$I_0 = 0$	$I_0 = 0$
0	0	1	$I_1 = 1$	$I_1 = 0$
0	1	0	$I_2 = 1$	$I_2 = 0$
0	1	1	$I_3 = 0$	$I_3 = 1$
1	0	0	$I_4 = 1$	$I_4 = 0$
1	0	1	$I_5 = 0$	$I_5 = 1$
1	1	0	$I_6 = 0$	$I_6 = 1$
1	1	1	$I_7 = 1$	$I_7 = 1$

Circuito



Demultiplexadores

- Os demultiplexadores são circuitos capazes de enviar informações contidas em um único canal de entrada à vários canais de saída.
- A Figura a seguir apresenta o esquema de um demultiplexador.



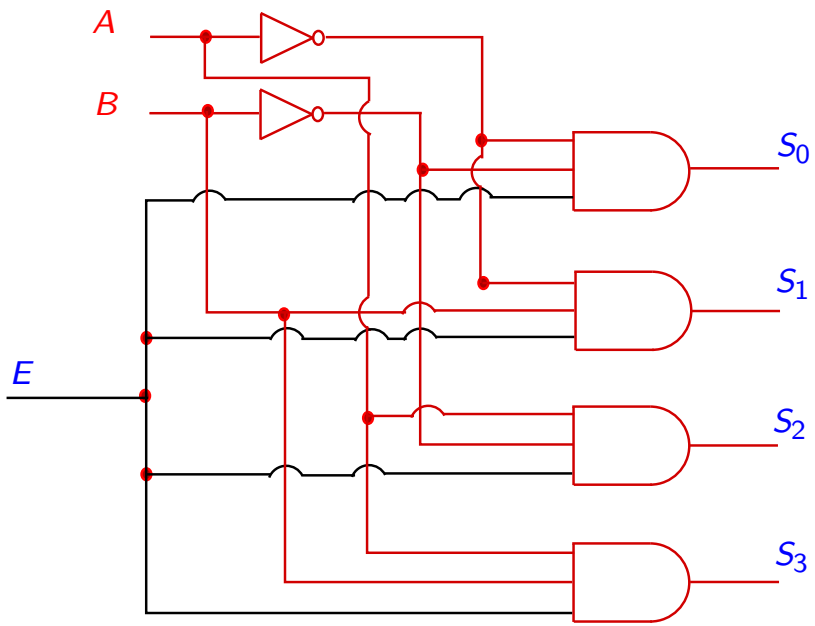
Demultiplexadores

- Vamos supor que temos 1 linha de informação e 4 linhas de transmissão. Neste caso o selecionador possui 2 bits e seu circuito está apresentado a seguir.

Tabela verdade

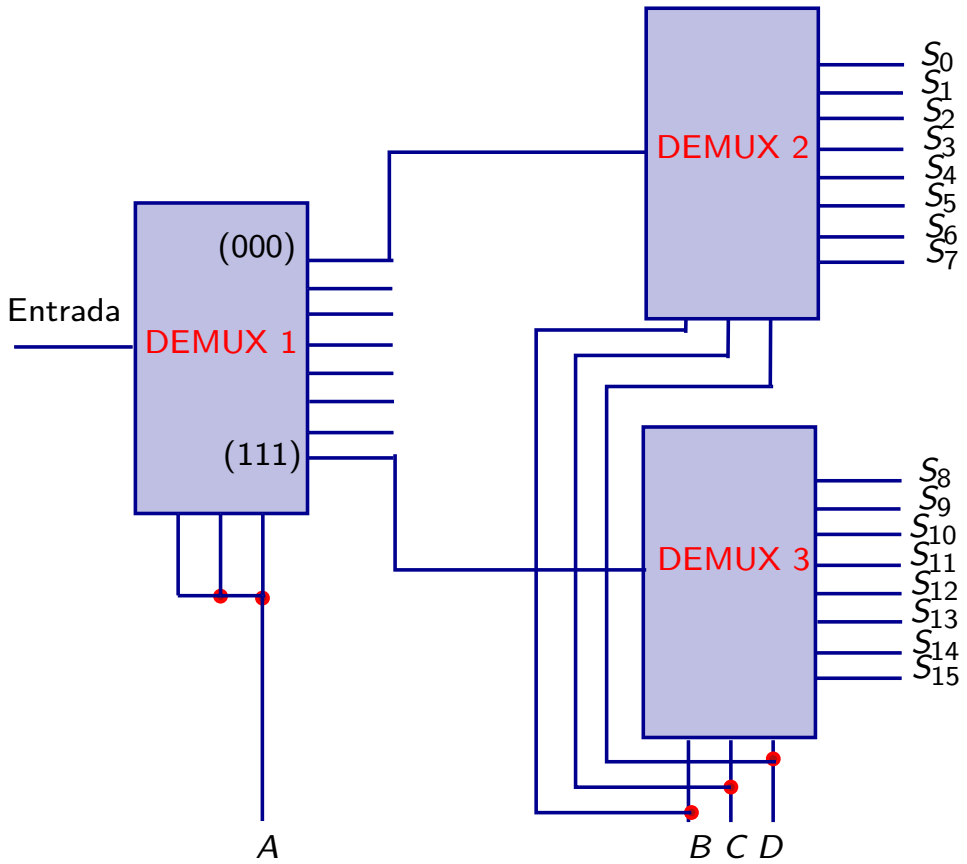
A	B	S_0	S_1	S_2	S_3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

Demultiplexador de 4 entradas



Demultiplexadores

- Para ampliarmos a capacidade de um demultiplexador podemos cascatear outros de menor capacidade. Exemplo : Demultiplex de 16 canais a partir de demultiplex de 8 canais.



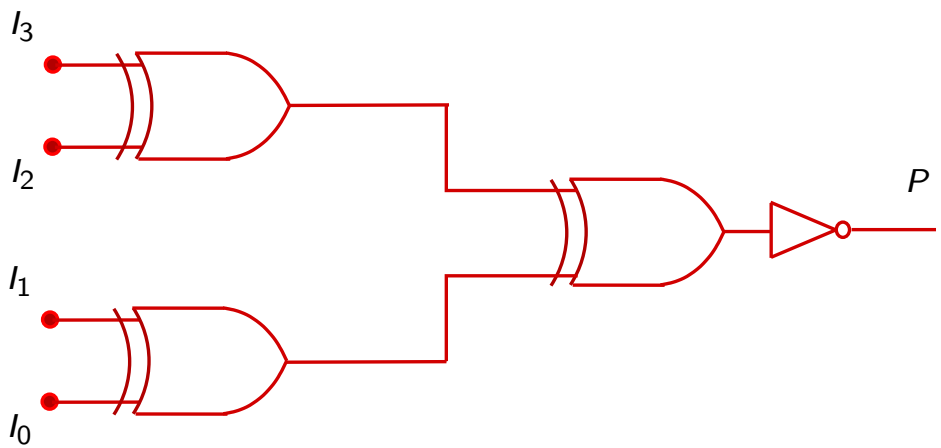
Gerador de paridade

- Considerando paridade ímpar podemos construir um circuito gerador de paridade.
- Para uma transmissão de 4 bits, a tabela verdade representa a saída do bit de paridade.

Tabela verdade

I_3	I_2	I_1	I_0	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Circuito



$$P = \overline{I_0 \oplus I_1 \oplus I_2 \oplus I_3}$$