ISCTE IUL
Instituto Universitário de Lisboa
Departamento de Ciências e Tecnologias da Informação

Arquitectura de Computadores (I)

Textos de apoio

– Projecto de circuitos combinatórios –

Draft – Versão 0.02a – Out./2009 – João Baptista e Tomás Brandão.

Índice

Projecto de circuitos combinatórios

1. Domínio e operações lógicas

1.1. Domínio

O projecto de circuitos combinatórios fundamenta-se numa álgebra cujas operações são feitas num *domínio* com apenas dois valores. Em Arquitectura de Computadores designam-se esses dois valores por '0' e '1' [(1)].

Neste domínio definem-se três operações elementares: ***AND***, ***OR*** e ***NOT***. Por exemplo, a operação AND opera dois valores e dá os seguintes resultados:

- *o* AND *de '0' com '0' dá '0'*;
- *o* AND *de '0' com '1' dá '0'*;
- *o* AND *de '1' com '0' dá '0'*;
- *o* AND *de '1' com '1' dá '1'*;

Com estes quatro casos fica completamente definida a operação AND. Como o domínio tem apenas dois valores, só existem quatro formas diferentes de combiná-los: *'0' com '0'*; *'0' com '1'*; *'1' com '0'* e *'1' com '1'*. Uma vez que foram apresentados os resultados para todas as operações que se podem realizar, nada mais há a acrescentar sobre a operação AND. No entanto, existem formas mais compactas de representar estes conceitos.

1.2. Tabela de verdade

Um das formas de representação é através de uma ***tabela de verdade***. Uma tabela de verdade indica, de uma forma sintética, *o resultado da operação para todas as combinações possíveis dos valores dos operandos*.

A tabela de verdade do AND é representada no quadro anexo. Na primeira linha vemos que o AND *de '0' com '0' dá '0'*; na segunda linha vemos que o AND *de '0' com '1' dá '0'*; e assim sucessivamente.

X	Y	AND
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

A tabela tem quatro linhas, que correspondem a todas as combinações possíveis dos valores de X e Y no domínio {0, 1}: '00', '01', '10', e '11'.

A ordem de apresentação é arbitrária. A tabela ao lado, onde as mesmas combinações aparecem por outra ordem, também descreve a operação AND, uma vez que dá os mesmos resultados para as mesmas combinações dos operandos.

X	Y	AND
0	0	**0**
1	1	**1**
1	0	**0**
0	1	**0**

[1] Os valores lógicos '0'/'1' são muitas vezes designados por '*low*'/'*high*', ou '*falso*'/'*verdadeiro*', respectivamente.

No entanto, a ordem da primeira tabela é mais natural porque, embora não seja o que está em causa, pode ser lida por ordem numérica! De facto, a sequência '00', '01', '10', '11' pode ler-se como a representação em base 2 dos números 0, 1, 2 e 3. Não havendo um critério melhor, será sempre usada esta ordenação das possíveis combinações dos bits.

1.3. Notação algébrica

Voltando ao princípio: o AND opera sobre dois valores, dando como resultado um terceiro valor, todos no domínio {0, 1}.

Dito de outro modo: o AND é uma função de duas variáveis do domínio {0,1} dando um resultado dentro do mesmo domínio. Usando os mecanismos de notação vulgares, designando por X, Y duas variáveis independentes e por Z o resultado da função, pode-se escrever:

$Z = F(X,Y)$ sendo que $Y, X, Z \in \{0, 1\}$ e $F(0,0)=0$, $F(0,1)=0$, $F(1,0)=0$ e $F(1,1)=1$

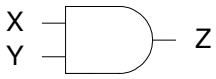
Algebricamente, representa-se a operação AND pelo símbolo '·' (ponto). Usando esta notação a função AND pode ser escrita como:

$Z=X \cdot Y$ com $0 \cdot 0 = 0$, $0 \cdot 1=0$, $1 \cdot 0=0$ e $1 \cdot 1=1$

1.4. Notação esquemática

Em última análise, o objectivo do estudo destas operações é o projecto ou a análise de circuitos. Para esse efeito, a representação esquemática é particularmente útil.

Em notação esquemática, o AND representa-se pelo seguinte símbolo:



O símbolo denota que, dados dois valores X e Y, o resultado Z comporta-se segundo a definição do AND. Tal como na expressão $Z = F(X, Y) = X \cdot Y$, o símbolo é claramente direccionado: à esquerda aparecem as variáveis independentes, X e Y; à direita tem-se o resultado (a variável dependente) Z.

No contexto da notação esquemática, é habitual designar por *entradas* as "variáveis independentes" e por *saídas* as variáveis dependentes.

Assim, diz-se que o símbolo anterior tem duas *entradas* X, Y e uma *saída* Z que comporta da seguinte maneira: se ambas as entradas forem 1, a saída será também 1. Se uma ou ambas as entradas forem 0, a saída será 0.

Exemplo: se X=0 e Y=1 tem-se que Z=0; com X=1 e Y=1 tem-se que Z=1.

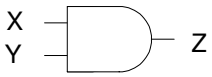
1.5. Operações elementares: AND, OR e NOT

As operações básicas que vão ser utilizadas definem-se do modo indicado na seguinte tabela, usando as várias notações introduzidas (tabela de verdade, notação algébrica e notação esquemática):

AND

$Z = F(X, Y) = X \cdot Y$

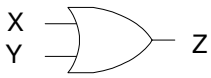
X	Y	AND
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**



OR

$Z = F(X, Y) = X + Y$

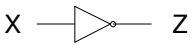
X	Y	OR
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**



NOT

$Z = F(X) = \overline{X}$

X	NOT
0	**1**
1	**0**



Tanto o OR como o AND operam sobre dois valores. Tal como já foi visto, o AND só dá '1' quando ambos os operandos são '1', dando '0' nos restantes casos. Já o OR só dá '0' quando ambos os operadores são '0', dando '1' quando pelo menos um dos operando é '1'.

O NOT (negação) é unário: opera sobre um valor e dá como resultado um outro valor; se o operando for '0' o NOT dá '1' e se o operando for '1' o NOT dá '0'.

1.6. Comutatividade e associatividade do AND e do OR

É fácil verificar que o AND é uma operação ***comutativa***, ou seja, a condição

$$X \cdot Y = Y \cdot X,$$

verifica-se para todos os valores possíveis de X e Y. Neste caso basta ver que as operações dão o mesmo resultado para as combinações '01' e '10', ou seja, que se tem $0 \cdot 1 = 1 \cdot 0$.

X	Y	Z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Da mesma forma, é fácil de verificar que a operação OR também é comutativa, ou seja,

$$X+Y = Y+X.$$

Também não é difícil verificar que a operação AND é ***associativa***, ou seja,

$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z).$$

Nesta expressão os parênteses têm o significado habitual: indicam que a operação que está entre parêntesis se deve fazer primeiro. Tomando como exemplo X=1, Y=1 e Z=0, tem-se:

- pela expressão da esquerda: $(1 \cdot 1) \cdot 0 = 1 \cdot 0 = 0$;
- pela expressão da esquerda: $1 \cdot (1 \cdot 0) = 1 \cdot 0 = 0$;

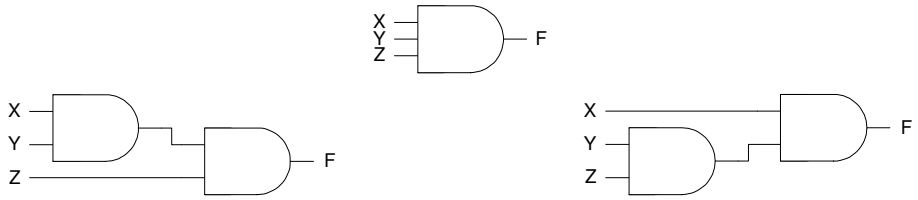
dando o mesmo resultado em ambos os casos.

Pode-se *demonstrar* esta propriedade fazendo os cálculos para todos os valores possíveis de X, Y e Z – neste caso para as 8 combinações binárias indicadas na tabela.

Da mesma forma, é fácil verificar que a operação OR também é uma operação associativa, ou seja,

$$(X + Y) + Z = X + (Y + Z).$$

A propriedade associativa permite estender as operações AND e OR para *n* variáveis. Assim, a expressão X·Y·Z é um "AND de três variáveis" que corresponde, indistintamente a (X·Y) ·Z ou X·(Y·Z). Em termos esquemáticos representamos um AND com três entradas, que substitui indistintamente um dos dois arranjos possíveis com ANDs de 2 entradas:



1.7. Elementos neutro e absorvente do AND e do OR

Outra característica importante das operações é a existência de ***elemento neutro*** ou de ***elemento absorvente***.

É fácil verificar que o '0' é elemento absorvente da operação AND: todas as combinações de operandos que envolvem pelo menos um '0' – '0·0', '0·1' e '1·0' – dão como resultado '0'. Por outras palavras, se um dos operandos da operação AND for '0' então o resultado da operação é '0'. Pode-se dizer que:

$X \cdot 0 = 0$, seja qual for o valor de X.

No caso do OR, o '1' é o elemento absorvente: todas as combinações em que está presente pelo menos um '1' – '0+1', '1+0' e '1+1' – dão como resultado '1'. Dito de outro modo, na operação OR, basta um dos operandos ser '1' para o resultado ser '1'. Pode-se dizer que:

$X+1 = 1$, seja qual for o valor de X.

O elemento absorvente é o valor que faz com que o resultado seja ele próprio. O elemento neutro é o valor que não condiciona o resultado.

No caso do AND, o elemento neutro é o '1'. Isso quer dizer que para todas as combinações onde um dos valores é '1' o resultado é igual ao outro valor: $0 \cdot 1 = 0$ e $1 \cdot 1 = 1$. Tem-se então que:

$X \cdot 1 = X$, independentemente do valor de X.

No caso do OR, o elemento neutro é o '0'. Isso quer dizer que, em todas as combinações onde um dos valores seja '0', o resultado é o outro valor: $0+0 = 0$ e $1+0 = 1$. Ou seja, tem-se:

$X+0 = X$.

O seguinte quadro sintetiza estas propriedades:

	AND	OR
Elemento neutro	X·**1** = X X, 1 → X	X+**0** = X X, 0 → X
Elemento absorvente	X·**0** = 0 X, 0 → 0	X+**1** = 1 X, 1 → 1

2. Circuitos lógicos

2.1. Expressão algébrica e tabela de verdade

Tomando por exemplo a expressão algébrica

$$F = XY + X\overline{Z},$$

pode-se verificar que esta conjuga várias das operações elementares definidas no ponto anterior. A uma expressão algébrica corresponde também um circuito e a uma tabela de verdade.

A tabela de verdade pode ser obtida calculando o valor da expressão para todas as combinações das variáveis envolvidas. Neste exemplo, temos 3 variáveis (X, Y e Z) pelo que as combinações possíveis são 2^3=8, como se ilustra na tabela. Para preencher a tabela, calcula-se o valor lógico de F para cada uma das combinações. Por exemplo:

X	Y	Z	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

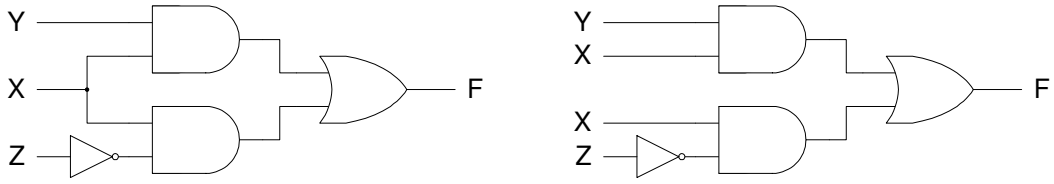
- para ‘000’ obtém-se F = 0·0 + 0·1 = **0**
- para ‘010’ obtém-se F = 0·1 + 0·1 = **0**
- para ‘100’ obtém-se F = 1·0 + 1·1 = **1**

Repare que na construção da tabela de verdade se usou o critério definido anteriormente: as combinações binárias aparecem pela ordem numérica dos números que representam. A primeira – ‘000’ – corresponde ao número 0, a segunda – ‘001’ – corresponde ao número 1 e assim sucessivamente até se chegar à combinação ‘111’, que corresponde ao número 7 (o maior número inteiro representável com 3 bits).

Esta ordenação é fácil de se obter usando um pequeno truque empírico: primeiro preenche-se a coluna da variável mais à direita, neste caso o Z, com a sequência “01010101”, alternando ‘0’s com ‘1’s; depois preenche-se a coluna do Y, alternando dois ‘0’s com dois ‘1’s, ou seja, “00110011”; finalmente o X, alternando de quatro em quatro, o que dá “00001111”.

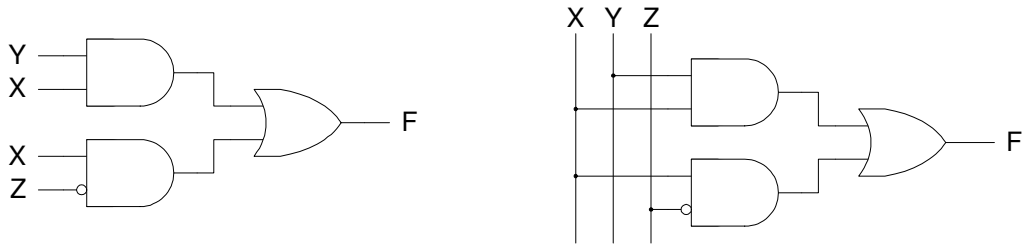
2.2. Circuitos

Utilizando os símbolos definidos na secção 1.5, o esquema de um circuito que corresponda à expressão algébrica anterior pode ser qualquer um dos seguintes:



As duas figuras são inteiramente equivalentes. Na primeira, a entrada *X* divide-se em dois ramos; na segunda menciona-se duas vezes a mesma variável *X* como entrada.

É também comum representar as negações (NOTs) com pequenas "bolinhas" que aparecem nas entradas das restantes portas lógicas. Isto permite obter representações como as que se seguem, onde a negação da variável *Z* é representada desse modo. As representações dos circuitos são obviamente equivalentes às anteriores.



Repare que na última representação os pontos de ligação são explicitamente marcados com um "ponto negro". Os cruzamentos entre linhas que não estão marcados com pontos não representam qualquer ligação.

2.3. Agrupamentos e parêntesis

Na expressão anterior os parênteses foram omitidos. A expressão completa seria:

$$F = (XY) + (X(\overline{Z})),$$

onde os parênteses têm o significado habitual: a expressão dentro dos parênteses tem que ser calculada primeiro, sendo substituída pelo respectivo resultado para continuar o cálculo da expressão. Quando os parênteses são omitidos, existe uma ordem de precedência dos operadores: primeiro vem o NOT, depois o AND e finalmente o OR.

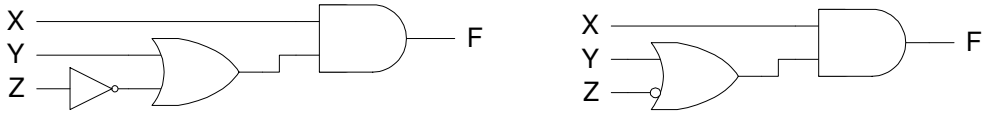
Considere agora a expressão

$$F = X(Y + \overline{Z}),$$

que é equivalente à anterior. Neste caso os parênteses são necessários e explicitam que primeiro deve ser feito o OR entre Y e a negação de Z e só depois se faz o AND entre esse resultado e X. Pode-se verificar que o resultado é equivalente ao anterior, construindo uma nova tabela de verdade. Por exemplo:

- para a sequência '000' tem-se F = 0·(0+1) = 0·1 = 0
- para a sequência '100' tem-se F = 1·(0+1) = 1·1 = 1

Uma representação do circuito correspondente poderia ser qualquer um dos seguintes esquemas:



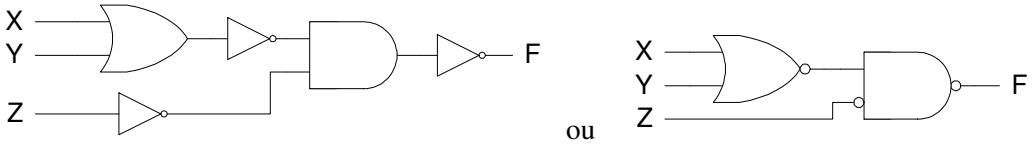
Há uma forma particular de agrupamento implícito que é o agrupamento através de uma barra de negação. A barra indica a negação da expressão contida por baixo dela, denotando implicitamente um par de parênteses que englobam essa expressão. Sendo assim, a expressão:

$$F = \overline{\overline{X + Y}.\overline{Z}}$$

tem o mesmo significado que:

$$F = \overline{\left(\overline{(X + Y)} \cdot (\overline{Z})\right)}.$$

O circuito correspondente é:



3. Síntese de circuitos

3.1. Termos mínimos

Considere agora o problema da síntese de um circuito, ou seja, o problema de, dada uma tabela de verdade, obter a expressão algébrica e o circuito que funciona de acordo com essa tabela de verdade.

Tomando a tabela de verdade anexa como exemplo, não é difícil chegar à conclusão de que uma expressão algébrica para a função lógica correspondente é dada por:

$$F = \overline{X}Y\overline{Z}$$

Para chegar a essa conclusão, repare que a tabela de verdade exibe apenas um '1', para a combinação de entrada '010', isto é, para os valores de X=0, Y=1 e Z=0. Para as restantes combinações o resultado dá sempre '0'.

X	Y	Z	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Uma expressão do género $\overline{X}Y\overline{Z}$ só dá resultado '1' para uma única combinação da tabela de verdade. Efectivamente, a expressão só dá '1' quando os valores das variáveis conduzirem a que todos os termos do AND sejam '1'. E isso só acontece para uma combinação em particular – neste caso só acontece quando os valores das variáveis conduzem à expressão

$$\overline{0}\cdot 1\cdot\overline{0} = 1\cdot 1\cdot 1 = 1$$

Assim, o termo $\overline{X}Y\overline{Z}$ só dá '1' quando X=0, Y=1 e Z=0. Para qualquer outro conjunto de valores a expressão dá '0'. Por exemplo:

para X=0, Y=0, Z=0 daria $\overline{0}\cdot 0\cdot\overline{0} = 1\cdot 0\cdot 1 = 0$;

para X=1, Y=1, Z=0 daria $\overline{1}\cdot 1\cdot\overline{0} = 0\cdot 1\cdot 1 = 0$; etc...

Uma expressão do género da anterior – um AND entre todas as variáveis da função, em que cada uma delas aparece negada ou não – designa-se por ***termo mínimo***. São termos mínimos, por exemplo:

$$XYZ \qquad \overline{X}Y\overline{Z} \qquad \overline{X}\,\overline{Y}\overline{Z} \qquad \overline{X}YZ$$

Não são termos mínimos, por exemplo:

- $\overline{X}\,\overline{Z}$, porque não envolve todas as variáveis;
- $\overline{X} + Y + \overline{Z}$, pois nem sequer é um AND;
- $\overline{X}Y\overline{\overline{Z}}$, porque não contém apenas variáveis isoladamente negadas ou não.

3.2. Notação sintética

Considerando de novo uma tabela com três variáveis, é fácil de verificar que neste caso existem 8 termos mínimos, um por cada combinação.

O processo de obter o termo mínimo correspondente a uma dada combinação das variáveis de entrada é fácil: onde a combinação tem '0', nega-se a variável respectiva; onde a combinação tem '1' aparece a variável sem estar negada.

Por exemplo, para a combinação '011' temos primeiro X negado, depois Y, e finalmente Z sem estar negado. O termo mínimo correspondente a essa combinação é portanto $\overline{X}YZ$. Substituindo os valores das variáveis pelos valores da combinação daria $\overline{0}\cdot 1\cdot 1=1$.

Por vezes é conveniente interpretar as combinações binárias da tabela de verdade como se fossem números (apesar de muitas vezes não ser isso que elas representam). Já foi mencionado que isso permite definir um critério para ordenar a apresentação da tabela de verdade onde, por exemplo, a combinação '011' (que representa também o número binário 3) aparece depois de '010' (número 2) e antes da '100' (número 4).

Pelo mesmo critério, os termos mínimos designam-se por números – designa-se por m_i o termo mínimo associado à combinação binária que representa o número *i*. Assim,

$\boldsymbol{m_0}$ é o mesmo que $\overline{X}\,\overline{Y}\,\overline{Z}$ (000)
$\boldsymbol{m_1}$ é o mesmo que $\overline{X}\,\overline{Y}\,Z$ (001)
…
$\boldsymbol{m_6}$ é o mesmo que $X\,Y\,\overline{Z}$ (110)
$\boldsymbol{m_7}$ é o mesmo que $X\,Y\,Z$ (111)

3.3. Soma de termos mínimos

Considere agora a função representada na tabela de verdade ao lado. É fácil verificar que a função pode ser escrita de acordo com a seguinte expressão:

$$F=\overline{X}\,\overline{Y}\,Z+X\,\overline{Y}\,Z$$

A função tem dois '1's: um para a combinação '001', outro para a combinação '101'. Para se chegar à expressão, juntaram-se através de um OR os termos mínimos correspondentes a cada um desses '1's.

X	Y	Z	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

De seguida apresenta-se com mais detalhe o raciocínio que conduziu a esse resultado.

Olhando para uma função descrita por uma tabela de verdade, podem-se definir dois grupos de combinações: aquelas para as quais a função dá '1' e aquelas para as quais a função dá '0'. O primeiro grupo é composto pelos chamados "*uns da função*" e o segundo grupo é composto pelos "*zeros da função*".

Considere agora uma expressão que seja a soma lógica (OR) dos termos mínimos correspondentes aos *1's da função*. Vejamos o resultado de uma expressão deste tipo para cada um dos dois grupos mencionados.

Tomando uma combinação que esteja no grupo dos *1's da função* temos a seguinte situação:

- o termo mínimo correspondente está na expressão, logo a expressão vai dar '1' para essa combinação;
- como a expressão global é um OR, é suficiente que um dos termos mínimos seja '1' para que o resultado final seja também '1', tal como se pretende.

Tomando agora uma combinação que esteja no grupo dos *0's da função*. Nesse caso o termo mínimo correspondente não está na expressão. Os termos mínimos presentes na expressão darão sempre '0' para essa combinação e logo o resultado final será também '0', como se pretende.

Exemplo: considere de novo a expressão anterior

$$F = \overline{X}\,\overline{Y}\,Z + X\,\overline{Y}\,Z$$

Seja uma combinação do grupo de *1's da função*, por exemplo '001'. Neste caso o termo mínimo está presente na função e o resultado será '1':

$$F = \overline{0} \cdot \overline{0} \cdot 1 + ... = 1 + ... = 1$$

Considerando agora uma combinação pertencente ao grupo dos *0's da função*, por exemplo '111'. Neste caso o termo mínimo correspondente (que seria o XYZ) não está presente. Como todos os termos mínimos presentes dão '0' para esta combinação, o resultado final será a soma de dois '0's.

$$F = \overline{1} \cdot \overline{1} \cdot 1 + 1 \cdot \overline{1} \cdot 1 = 0 \cdot 0 \cdot 1 + 1 \cdot 0 \cdot 1 = 0 + 0 = 0$$

Os exemplos anteriores ilustram sempre casos com 3 variáveis. É claro que os conceitos se aplicam, na mesma forma para qualquer número de variáveis.

Por exemplo,

- numa situação com 2 variáveis os termos mínimos são 4;
- numa situação com 4 variáveis os termos mínimos são 16.

4. Simplificação através de mapas de *Karnaugh*

4.1. Simplificação

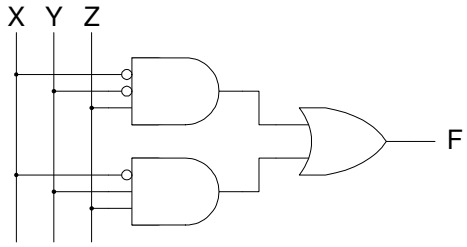
Considere a tabela de verdade anexa.

X	Y	Z	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

A função representada tem dois '1's. A expressão algébrica pode ser a que obtém somando os dois termos mínimos correspondentes:

$$F = \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,Z$$

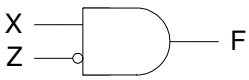
O circuito correspondente pode ser o seguinte:



Acontece que, como se verá mais à frente, esta expressão pode ser simplificada para:

$$\begin{aligned} F &= \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,Z \\ &= \overline{X}Z\left(Y + \overline{Y}\right) \\ &= \overline{X}Z \cdot 1 \\ &= \overline{X}Z \end{aligned}$$

Quer isto dizer que a expressão $\overline{X}\,\overline{Y}\,Z + \overline{X}YZ$ dá sempre o mesmo resultado que $\overline{X}Z$. Sendo assim, em vez do circuito anterior pode-se ter para o mesmo efeito o seguinte circuito (que é bem mais simples):



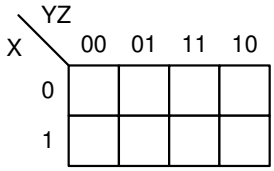
No primeiro circuito usavam-se dois ANDs (de 3 entradas), um OR e dois NOTs. No segundo circuito, foram usados apenas um AND e um NOT para implementar a mesma função.

Anteriormente foi vista uma forma automática de deduzir uma expressão algébrica (e consequentemente um circuito) para implementar uma função definida por uma tabela de verdade. Esta secção descreve um processo de simplificar essa expressão de modo a obter um circuito mais simples para implementar a mesma tabela de verdade.

4.2. Representação num mapa de *Karnaugh*

Um ***mapa de Karnaugh*** é uma forma de representação de uma função lógica, através de uma *tabela de duas entradas*, com tem algumas particularidades.

Considere-se de novo uma função lógica de 3 variáveis, designadas por X, Y e Z. Para este caso (3 variáveis) um mapa de Karnaugh tem a seguinte estrutura:



O mapa de Karnaugh tem neste caso 8 casas, que correspondem às 8 combinações possíveis dos valores de três variáveis. No canto superior esquerdo encontra-se a casa para o valor correspondente à combinação '000', ao lado encontra-se lugar para a combinação '001', depois para a combinação '011' e finalmente, no canto superior direito, está a casa para a combinação '010'. Na linha de baixo temos, da esquerda para a direita, as casas correspondentes às combinações '100', '101', '111' e '110'.

Sinteticamente, o quadro seguinte indica a posição correspondente a cada um dos 8 termos mínimos definidos por uma função de 3 variáveis:

X \ YZ	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Exemplo: uma tabela de verdade e a respectiva representação num mapa de Karnaugh:

X	Y	Z	F
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

X \ YZ	00	01	11	10
0	1	0	0	1
1	1	1	1	0

4.3. Adjacências

Contrariamente ao que foi indicado na secção 1 como prática a adoptar, repare que num mapa de Karnaugh a combinação correspondente ao par YZ não está representa por ordem "numérica" – está representada na ordem '00', '01', '11', '10', que corresponde à sequência dos números 0, 1, 3, 2.

Neste caso é assim mesmo que se deve proceder e este é um ponto fulcral: esta ordem faz com que *entre casas adjacentes do mapa, apenas varie um bit da combinação binária correspondente*.

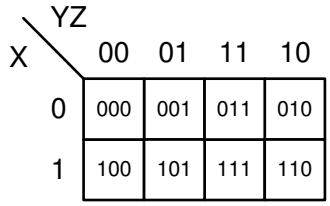
A partir daqui diz-se que duas combinações binárias são adjacentes se diferem apenas num bit. Assim, por exemplo:

- '000' e '001' são adjacentes
- '110' e '100' são adjacentes
- '101' e '110' não são adjacentes.

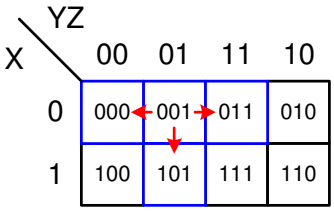
O mapa de Karnaugh é construído de forma que as combinações adjacentes ocupem posições também adjacentes na tabela. Pode verificar que para cada combinação de 3 bits há três outras combinações que lhe são adjacentes – justamente todas as que se obtêm trocando um dos bits. Por exemplo, as combinações adjacentes a '010' são:

- '110' – trocando o bit da esquerda;
- '000' – trocando o bit do meio;
- '011' – trocando o bit da direita.

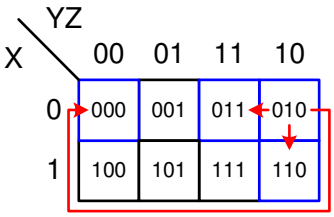
Observando de novo o mapa de Karnaugh e assinalando as combinações binárias correspondentes a cada casa do mapa, pode-se verificar que, para todos os casos, as três combinações adjacentes são as que se encontram nas três casas vizinhas.



O quadro seguinte representa, como exemplo, as adjacências da combinação '001':



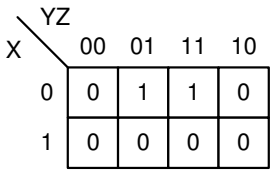
Quanto às casas ocupadas pelos cantos, considere por exemplo a combinação '010'. Para esse caso, o mapa deve ser visto como um cilindro, de modo a que as casas da coluna mais à esquerda sejam vizinhas das da coluna mais à direita. As adjacências de '010' são portanto as indicadas no seguinte mapa:



4.4. Simplificação através do mapa de Karnaugh

Considere novamente a seguinte tabela de verdade, que se traduz pelo mapa de Karnaugh indicado.

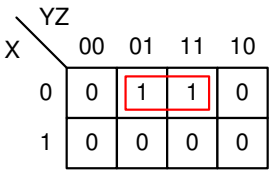
X	Y	Z	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**



Quer através da tabela de verdade, quer através do mapa de Karnaugh, pode-se ver que a função tem apenas dois '1's, um correspondente à combinação '001' e outro à combinação '011'. Uma expressão algébrica para F poderia portanto ser:

$$F = \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,Z$$

Acontece que usando o mapa de Karnaugh se consegue obter facilmente uma expressão simplificada. Para isso juntam-se os dois '1's adjacentes, tal como o indicado na figura.



Considerando o agrupamento indicado deduz-se a expressão algébrica correspondente aplicando o seguinte raciocínio:

- Como o valor de X para ambas as casas do agrupamento é '0', fica $\overline{X}$;
- O agrupamento engloba dois valores de Y, por isso Y não vai entrar na expressão;
- O agrupamento apanha apenas um valor de Z. Como esse valor é '1' fica Z;

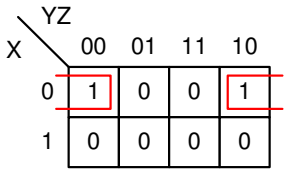
A expressão final vai ficar portanto $F = \overline{X}Z$

Basicamente, um mapa de Karnaugh permite fazer as simplificações algébricas de uma forma gráfica. O procedimento para esse efeito consiste em construir agrupamentos de '1's pertencentes a casas adjacentes. Mas atenção que o número de '1's dentro de cada agrupamento tem que ser uma potência natural de 2. Isto quer dizer que se podem fazer agrupamentos com 2, 4, 8, 16, … '1's adjacentes.

Para cada agrupamento deduz-se uma expressão algébrica aplicando a seguinte regra prática: para cada variável, vêm-se os valores que ela toma nas posições do grupo considerado: se variar (*i.e.* se tomar valores '0' e '1') a variável é descartada; se tomar apenas o valor '1' a variável fica; se tomar apenas o valor '0' a variável fica, mas aparece negada. A expressão algébrica correspondente é a que se obtém fazendo AND entre as variáveis que ficam.

Exemplo: no mapa de Karnaugh indicado representa-se um agrupamento que junta dois ‘1’s adjacentes colocados em dois cantos adjacentes do quadro.

O raciocínio é idêntico – para o grupo assinalado no quadro:

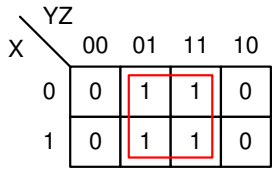


- X tem sempre o valor ‘0’, logo fica $\overline{X}$;
- Y varia, e portanto não aparece na expressão;
- verifica-se que Z tem também o valor ’0’, logo fica $\overline{Z}$.

  A expressão será então $F = \overline{X}\,\overline{Z}$

Neste outro exemplo pretende-se ilustrar o que acontece quando se têm quatro ‘1’s adjacentes.

Aplicando o mesmo raciocínio:



- X varia, logo sai;
- Y varia e também sai;
- Z tem o valor ‘1’, logo fica Z.

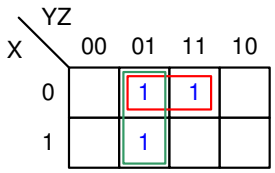
  A expressão final é $F = Z$

4.5. Obter uma expressão simplificada

Os exemplos anteriores mostram como se obtém a expressão algébrica para um grupo isolado. No entanto, é comum aparecerem vários grupos dentro do mesmo mapa de Karnaugh. De uma forma geral, quando se quer obter uma expressão algébrica simplificada a partir de um mapa de Karnaugh, aplica-se o seguinte procedimento:

1. obter todos os grupos possíveis, juntando em cada grupo o maior número possível de ‘1’s adjacentes (em grupos de 2, 4, 8, 16, ...);
2. seleccionar um conjunto de grupos de forma a apanhar todos os ‘1’s;
3. obter a expressão final juntado num OR as expressões obtidas para cada um dos grupos.

Exemplo: Considere o seguinte mapa (apenas foram representados os ‘1’s da função, já que é isso que interessa; as casa vazias representam os ‘0’s da função).



A função tem três ‘1’s que se podem agrupar segundo os dois grupos indicados. O grupo organizado ao alto corresponde à expressão $\overline{Y}Z$; o outro (organizado na horizontal) corresponde à expressão $\overline{X}Z$.

A expressão resultante é a soma (OR) desses dois termos, ou seja:

$$F = \overline{X}Z + \overline{Y}Z$$

A estratégia a seguir é formar agrupamentos de ‘1’s com a maior dimensão possível, mesmo que isso conduza a sobreposições.

O resultado final é evidente:

- se calcular a expressão para uma combinação que não faça parte de qualquer grupo todos os termos darão ‘0’, logo o resultado final será ‘0’.

  Exemplo: ‘000’ => F = 1.0 + 1.0 = 0 + 0 = 0

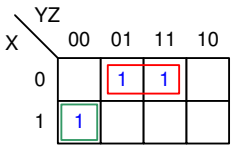
- se calcular a expressão para uma combinação incluída apenas num único grupo, um dos termos dará ‘1’ e os restantes ‘0’, sendo o resultado final ‘1’.

  Exemplo: ‘011’ => F = 0.1 + 1.1 = 0+1 = 1

- Se calcular a expressão para uma combinação incluída em vários grupos, os termos correspondentes a esses grupos darão ‘1’, sendo o resultado final também ‘1’.

  Exemplo: ‘010’ => F = 1.1 + 1.1 = 1+1 = 1

Exemplo:



Neste mapa existem três ‘1’s: dois deles estão em casas adjacentes permitindo formar um grupo que corresponde a $\overline{X}Z$; o outro está isolado, formando um grupo que contém apenas um ‘1’ e que corresponde ao termo mínimo $X\,\overline{Y}\,\overline{Z}$.

A expressão final é:

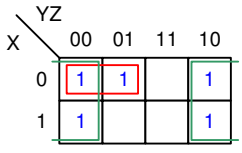
$$F = \overline{X}Z + X\,\overline{Y}\,\overline{Z}$$

Em suma, num mapa de Karnaugh com 3 variáveis, podem ser feitos os seguintes agrupamentos:

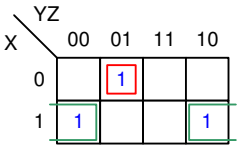
- um ‘1’ isolado, obtendo-se um termo com 3 variáveis (termo mínimo);
- dois ‘1’s adjacentes, obtendo um termo com 2 variáveis;
- quatro ‘1’s adjacentes, obtendo um termo com 1 variável;
- oito ‘1’s adjacentes, obtendo um termo sem variáveis (o valor constante ‘1’).

O número de ‘1’s que formam um grupo tem que ser uma potência de 2, e os agrupamentos feitos têm que ter a forma de rectângulos ou de quadrados.

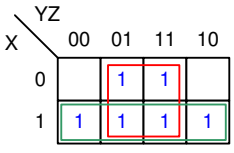
Exemplos:



$F = \overline{X}\,\overline{Y} + \overline{Z}$



$F = \overline{X}\,\overline{Z} + \overline{X}\,\overline{Y}\,Z$



$F = X + Z$

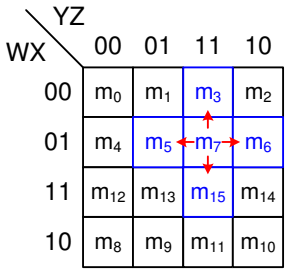
4.6. Mapas de Karnaugh com 4 variáveis

A estrutura de um mapa de Karnaugh com 4 variáveis é a indicada no quadro seguinte, que mostra os termos mínimos e as posições correspondentes. A ordem de preenchimento, partindo de uma tabela de verdade, é a indicada pela sequência de termos mínimos.

WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Importa reparar que a troca na ordem da sequência se verifica agora também para o par WX, tal como já acontecia num mapa de 3 variáveis para o par YZ – ambos seguem a sequência '00', '01', '11', '01'.

Cada combinação tem agora 4 casas adjacentes. O quadro anexo mostra as adjacências do termo m_7 – $\overline{W}\,X\,Y\,Z$ (0111), que são:



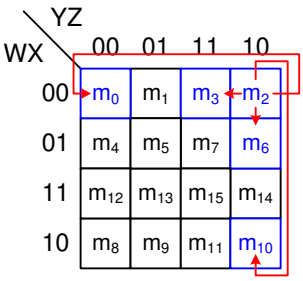
m_3 – $\overline{W}\,\overline{X}\,Y\,Z$ (0011)

m_5 – $\overline{W}\,X\,\overline{Y}\,Z$ (0101)

m_6 – $\overline{W}\,X\,Y\,\overline{Z}$ (0110)

m_{15} – $W\,X\,Y\,Z$ (1111)

Tal como já foi salientado, um mapa de Karnaugh deve ser entendido de uma forma circular, quer no sentido do par YZ (como foi visto nos mapas de 3 variáveis) quer no sentido do par WX. O quadro seguinte ilustra, por exemplo, as adjacências da combinação m_2 – $\overline{W}\,\overline{X}\,Y\,\overline{Z}$ (0010), que são:



m_1 – $\overline{W}\,\overline{X}\,Y\,Z$ (0011)

m_6 – $\overline{W}\,X\,Y\,\overline{Z}$ (0110)

m_{10} – $W\,\overline{X}\,Y\,\overline{Z}$ (1010)

m_0 – $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ (0000)

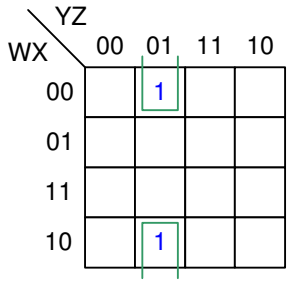
Num mapa de Karnaugh com 4 variáveis podem fazer-se os seguintes agrupamentos:

- um '1' isolado, obtendo um termo com 4 variáveis (termo mínimo);
- dois '1's adjacentes, obtendo um termo com 3 variáveis;
- quatro '1's adjacentes, obtendo um termo com 2 variáveis;
- oito '1's adjacentes, obtendo um termo com 1 variável;
- 16 '1's adjacentes, obtendo um termo com 0 variáveis (o valor constante '1');

Para obter a expressão correspondente a um grupo aplica-se a mesma regra prática que já foi vista a propósito dos mapas de 3 variáveis. Observe agora os seguintes exemplos:

Exemplo 1:

Neste caso temos um grupo de dois ‘1’s adjacentes. Considerando as duas posições deste grupo temos que:

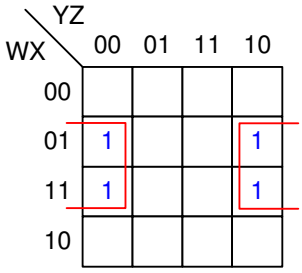


- W varia (tem dois valores para o grupo), logo sai;
- X tem o valor 0, logo fica $\overline{X}$;
- Y tem o valor 0, logo fica $\overline{Y}$;
- Z tem o valor 1, logo fica Z.

A expressão é $F = \overline{X}\,\overline{Y}\,Z$

Exemplo 2:

Neste caso temos um grupo com quatro ‘1’s adjacentes. Considerando as quatro posições deste grupo temos que:



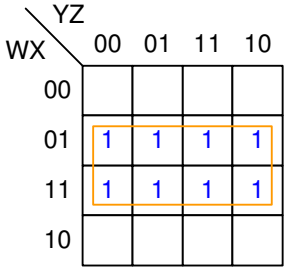
- W varia, logo sai;
- X tem o valor 1, logo fica X;
- Y varia, logo sai;
- Z tem valor 0 logo, fica $\overline{Z}$;

A expressão é $F = X\overline{Z}$

Exemplo 3:

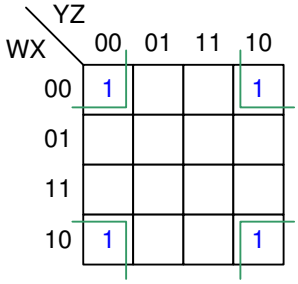
Neste exemplo temos um grupo com oito ‘1’s adjacentes. Considerando as oito posições do grupo é fácil de verificar que X é a única variável que permanece com valor lógico constante e igual a 1. Sendo assim:

A expressão é F = X



Exemplo 4:

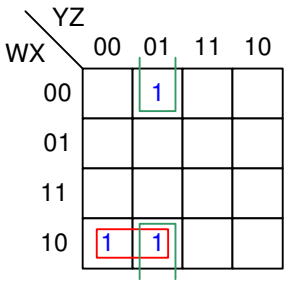
Um grupo muito característico é o que aparece pela adjacência dos quatro cantos do mapa. Neste caso, considerando as quatro posições do grupo, temos que:



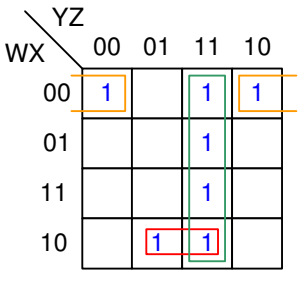
- Y varia, logo sai;
- Z tem valor, 0 logo fica $\overline{Z}$;
- W varia, logo sai;
- X tem valor 0, logo fica $\overline{X}$;

A expressão é $F = \overline{X}\,\overline{Z}$

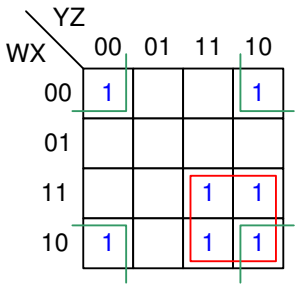
Nos seguintes exemplos estão presentes diversos agrupamentos de 1's dentro do mesmo mapa de Karnaugh:



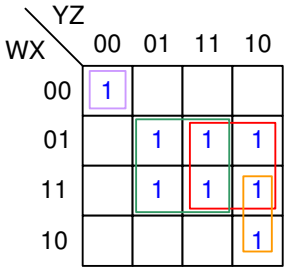
$F = W\overline{X}\overline{Y} + \overline{X}YZ$



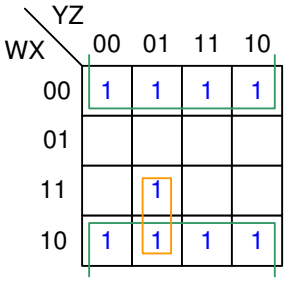
$F = YZ + W\overline{X}Z + \overline{W}\,\overline{X}\,\overline{Z}$



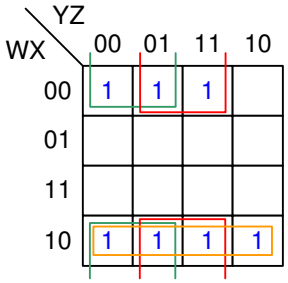
$F = \overline{X}\overline{Z} + WY$



$F = \overline{W}\overline{X}\overline{Y}\overline{Z} + XZ + XY + WY\overline{Z}$



$F = \overline{X} + W\overline{Y}Z$



$F = W\overline{X} + \overline{X}\overline{Y} + \overline{X}Z$

5. Álgebra de Boole

5.1. Propriedades elementares

Os fundamentos do projecto de circuitos lógicos combinatórios assentam na *álgebra de Boole*. Anteriormente já foi visto que esta álgebra funciona num domínio de 2 valores, designados por '0' e '1'. As operações básicas definidas neste domínio são o AND, o OR e o NOT, tal como vimos na secção 1.

Face as estas definições são válidas as seguintes identidades:

	AND	OR
(1) Elemento neutro	$X \cdot 1 = X$	$X + 0 = X$
(2) Elemento absorvente	$X \cdot 0 = 0$	$X + 1 = 1$
(3) Identidade	$X \cdot X = X$	$X + X = X$
(4) Contrário	$X \cdot \overline{X} = 0$	$X + \overline{X} = 1$
(5) Prop. Comutativa	$X \cdot Y = Y \cdot X$	$X + Y = Y + X$
(6) Prop. Associativa	$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$	$X + (Y + Z) = (X + Y) + Z$
(7) Prop. Distributiva	$X \cdot (Y + Z) = XY + XZ$	$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
(8) Regras de DeMorgan	$\overline{X \cdot Y} = \overline{Y} + \overline{X}$	$\overline{X + Y} = \overline{Y} \cdot \overline{X}$

As expressões em (1) dizem que '1' é o elemento neutro do AND e que '0' é o elemento neutro do OR. Estas identidades podem ser facilmente demonstradas por exaustão dos valores possíveis das variáveis envolvidas, que são apenas dois: o '0' e o '1'. Por exemplo, podemos demonstrar a expressão (1) para o OR fazendo as duas únicas operações possíveis para a expressão:

- para X=0, que dá 0+0 = 0 ou seja o valor de X;
- para X=1, que dá 1+0 = 1 ou seja, também, o valor de X;

As expressões em (2) indicam que o AND de qualquer coisa com '0' dá sempre '0', ou seja, que '0' é o elemento absorvente do AND e que, da mesma forma, o '1' é o elemento absorvente do OR. As expressões em (2) podem ser generalizadas a outras situações, por exemplo:

$$\overline{X}YZ + \overline{Z} + 1 = 1 \text{ ou } X \cdot Y \cdot Z \cdot 0 = 0$$

As expressões em (3) indicam que perante um AND ou um OR entre dois termos semelhantes se pode cortar um deles. Por exemplo:

$$X\overline{Y}Z + X\overline{Y}Z = X\overline{Y}Z \text{ ou } \overline{(X + Y) \cdot (X + Y)} = \overline{X + Y}$$

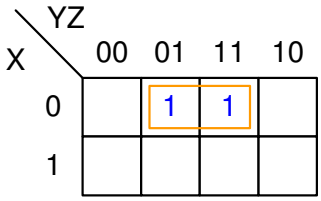
As expressões em (4) mostram que um AND ou um OR entre um termo e a sua negação dão '0' e '1', respectivamente. Por exemplo:

$$AB + \overline{AB} = 1 \text{ ou } (X + Y + Z) \cdot \overline{X + Y + Z} = 0$$

As expressões em (5) e (6) indicam que as operações AND e OR são comutativas e associativas, como já tinha sido referido na secção 1.

As expressões em (7) indicam que a operação AND é distributiva relativamente ao OR e que a operação OR é também distributiva relativamente ao AND.

As expressões em (4) e (7) relacionam-se muito directamente com as simplificações feitas através do mapa de Karnaugh. Considere, por exemplo, o agrupamento descrito no mapa anexo. Os dois '1's representados correspondem aos termos mínimos:



$$F = \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,Z$$

O agrupamento no mapa de Karnaugh corresponde à simplificação algébrica

$$\begin{aligned} F &= \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,Z \\ &= \overline{X}Z\left(Y + \overline{Y}\right) \\ &= \overline{X}Z \cdot 1 \\ &= \overline{X}Z \end{aligned}$$

Relembre que uma adjacência no mapa de Karnaugh corresponde a duas combinações que diferem apenas num bit. Nessas condições, a duas combinações correspondem também a dois termos mínimos que diferem apenas numa variável (num deles aparece negada e no outro não). Ao fazermos desaparecer a variável que varia no grupo está-se na realidade a fazer a simplificação algébrica do mesmo modo que o ilustrado no exemplo.

As expressões em (8) são as chamadas *leis de DeMorgan*. A primeira diz que a negação da soma (OR) é o produto (AND) das negações. A segunda diz que a negação do produto (AND) é a soma (OR) das negações.

5.2. Alguns exemplos

Exemplo 1 – Manipulações simples:

a) $X + XY = X(1 + Y) = X$

b) $XY + X\overline{Y} = X\left(Y + \overline{Y}\right) = X$

c) $X + \overline{X}Y = (X + \overline{X})(X + Y) = X + Y$

d) $X(X + Y) = X + XY = X(1 + Y) = X$

e) $(X + Y)(X + \overline{Y}) = XX + X\overline{Y} + XY + Y\overline{Y} = X$

f) $X(\overline{X} + Y) = XY$

Exemplo 2 – Complicar para depois simplificar

$\overline{X} + X\overline{Y} = (X + \overline{X})(\overline{X} + \overline{Y}) = \overline{X} + \overline{Y}$ ou $\overline{X}(1 + \overline{Y}) + X\overline{Y} = \overline{X} + \overline{X}\overline{Y} + X\overline{Y} = X + \overline{Y}(X + \overline{X}) = X + \overline{Y}$

Exemplo 3 – Teorema do Consenso

Pretende-se mostrar que

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

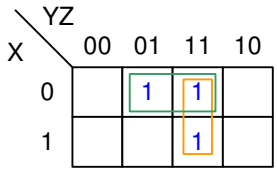
assim:

$$\begin{aligned} XY + \overline{X}Z + YZ &= XY + \overline{X}Z + YZ(X + \overline{X}) \\ &= XY + \overline{X}Z + XYZ + \overline{X}YZ \\ &= XY(1 + Z) + \overline{X}Z(1 + Y) \\ &= XY + \overline{X}Z \end{aligned}$$

Exemplo 4 – Mapa de Karnaugh

A função representada no MK corresponde a

$$F = \overline{X}\,\overline{Y}Z + \overline{X}YZ + XYZ,$$



e simplifica-se para:

$$\begin{aligned} F &= \overline{X}\,\overline{Y}Z + \overline{X}YZ + XYZ \\ &= \overline{X}\,\overline{Y}Z + \overline{X}YZ + \overline{X}YZ + XYZ \\ &= XZ(Y + \overline{Y}) + YZ(X + \overline{X}) \\ &= \overline{X}Z \cdot 1 + YZ \cdot 1 \\ &= \overline{X}Z + YZ \end{aligned}$$

6. Dualidade OR–AND

6.1. Princípio

As propriedades da álgebra de Boole evidenciam um princípio que é o chamado *princípio da dualidade*: a partir de uma expressão válida pode-se obter uma outra expressão válida trocando os '0's por '1's e os ANDs por ORs. Esta troca permite obter propriedades relacionadas com o OR a partir das propriedades correspondentes relacionadas com o AND, e vice-versa.

Por exemplo, partindo da propriedade "elemento absorvente do OR":

$X + 1 = 1$,

se trocarmos o OR por AND e os '1's por '0's obtém-se a propriedade "elemento absorvente do AND":

$X \cdot 0 = 0$.

6.2. Termos máximos

A forma de obter um circuito combinatório a partir de uma tabela de verdade normalmente é feita com base nos '1's da função (os termos mínimos). Com o princípio da dualidade, pode-se reformular este processo, fazendo-o neste caso com base nos '0's da função.

Considere-se por exemplo a tabela de verdade anexa. Pode-se obter o circuito correspondente partindo dos termos mínimos, e neste caso chega-se à função:

$$F = \overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}\,\overline{Y}\,Z + X\,\overline{Y}\,\overline{Z} + X\,\overline{Y}\,Z + X\,Y\,\overline{Z}$$

X	Y	Z	F
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

Tal como já foi visto, um termo mínimo é uma expressão que consiste num AND entre todas as variáveis (negadas ou não). Por consequência, só dá '1' para uma das combinações de valores possíveis das variáveis. Por exemplo, à combinação '001' corresponde o termo mínimo $\overline{X}\,\overline{Y}\,Z$ que só dá '1' quando todos os termos do produto forem '1' (a combinação '001').

O *dual* de um *termo mínimo* designa-se por *termo máximo*.

Um termo máximo opera sobre todas as variáveis (negadas ou não) através de um OR e, com base no princípio da dualidade, só dá '0' para uma das combinações de valores possíveis das variáveis. Por exemplo, à combinação '011' corresponde o termo máximo $X + \overline{Y} + \overline{Z}$, que só dá zero quando todos os termos da soma forem '0'.

Dada uma combinação binária, a regra prática para obter o termo mínimo é negar as variáveis correspondentes aos '0's da combinação; para o termo máximo, a regra prática será então negar as variáveis correspondentes aos '1's da combinação, como é feito nos seguintes exemplos:

Combinação	Termo mínimo	Termo máximo
000	$\overline{X}\,\overline{Y}\,\overline{Z}$	$X+Y+Z$
001	$\overline{X}\,\overline{Y}\,Z$	$X+Y+\overline{Z}$
111	$X\,Y\,Z$	$\overline{X}+\overline{Y}+\overline{Z}$

De modo análogo ao que acontecia com os termos mínimos, pode-se também obter uma expressão para qualquer função lógica na forma *produto de termos máximos*. Para tal, basta ver na tabela de verdade quais os termos máximos correspondentes aos '0's da função e juntá-los num produto lógico.

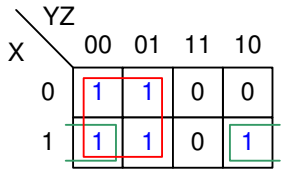
Por exemplo, na tabela de verdade anterior, o produto de termos máximos correspondente é o seguinte:

$$F=(X+\overline{Y}+Z)\cdot(X+\overline{Y}+\overline{Z})\cdot(\overline{X}+\overline{Y}+\overline{Z})$$

6.3. Simplificação pelos '0's

Anteriormente foi visto que, usando um mapa de Karnaugh obtém-se uma expressão simplificada formando grupos de '1's adjacentes. Usando o princípio da dualidade, também se pode obter uma expressão simplificada com base nos '0's adjacentes.

Considere o mapa de Karnaugh anexo. Em cada grupo excluíram-se as variáveis para as quais há variação do valor dentro do grupo. As restantes formam um termo produto (AND) em que as variáveis que têm valor constante '0' aparecem negadas. A expressão final é a soma (OR) dos termos assim obtidos.

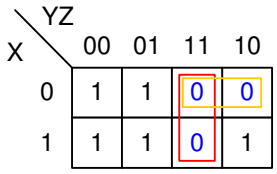


Para este caso obtém-se: $F=\overline{Y}+X\overline{Z}$

Aplicando o princípio da dualidade, em cada grupo excluem-se as variáveis para as quais há variação do valor dentro do grupo. As restantes formam um termo soma (OR) em que as variáveis que têm valor constante '1' aparecem negadas. A expressão final é o produto (AND) dos termos assim obtidos.

Neste caso tem-se então:

- no grupo vertical sai X; ficam Y e Z ambos a '1': $\overline{Y}+\overline{Z}$
- no grupo horizontal sai Z; fica X a '0' e Y a '1': $X+\overline{Y}$



A expressão final é:

$$F=(\overline{Y}+\overline{Z})\cdot(X+\overline{Y})$$

Como seria de esperar, ambas as expressões obtidas para F são equivalentes:

$$F=(\overline{Y}+\overline{Z})\cdot(X+\overline{Y})=\overline{Y}+X\overline{Z}$$

Para demonstrar, basta aplicar a distributividade do OR em relação ao AND.

7. Operações e portas lógicas derivadas

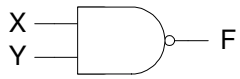
Além das portas lógicas NOT, AND e OR existem outras portas lógicas disponíveis para montar circuitos. Estas portas lógicas correspondem a operações lógicas derivadas, isto é, que se podem definir à custa das operações básicas NOT, AND e OR.

7.1. NAND e NOR

A função **NAND** é dada por $F = \overline{XY}$ e pode ser descrita pela tabela de verdade anexa. O NAND é uma operação que dá como resultado a negação do AND (daí o nome NAND – "NOT AND"). Sendo assim, só dá '0' quando ambos os operadores são '1' e dá '1' nos restantes casos.

X	Y	F
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

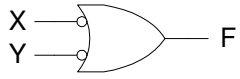
A representação esquemática de uma porta NAND é dada pelo seguinte símbolo:



Ou seja, uma porta AND seguida com uma "bolinha" que representa a negação. Em alternativa, e com base nas regras de DeMorgan, tem-se que:

$$\overline{XY} = \overline{X} + \overline{Y},$$

o que indica que uma porta NAND é o mesmo que uma porta OR com as entradas negadas. Sendo assim, outra possível representação de uma porta NAND é dada pelo símbolo:



O **NOR** é a operação *dual* do NAND. O resultado do NOR é exactamente a negação do OR: só dá '1' quando ambos os valores de entrada são 0, como é evidenciado na tabela anexa.

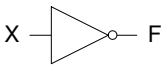
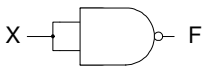
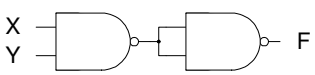
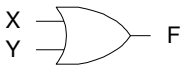
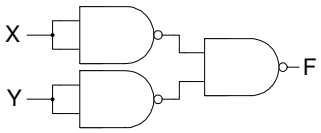
X	Y	F
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

A função lógica desempenhada por um NOR é dada por $F = \overline{X + Y}$ e para a sua representação esquemática pode-se usar um dos símbolos seguintes:

O primeiro é a forma directa, que representa o NOR como um "OR negado". O segundo justifica-se com base na regra de DeMorgan $\overline{X + Y} = \overline{X} \cdot \overline{Y}$, que basicamente diz que um NOR é mesmo que um AND com as entradas negadas.

7.2. Circuitos normalizados com NANDs

O NAND é uma porta lógica universal, pois é possível implementar cada uma das operações elementares NOT, AND e OR usando apenas portas lógicas NAND. Tal pode ser visto no seguinte quadro:

	Porta Lógica	Equivalente com NANDs
NOT	$F = \overline{X}$ 	$F = \overline{XX} = \overline{X}$ 
AND	$F = XY$ 	$F = \overline{\overline{XY}} = XY$ 
OR	$F=X+Y$ 	$F = \overline{\overline{X}\,\overline{Y}} = \overline{\overline{X}} + \overline{\overline{Y}} = X + Y$ 

O NOR também é uma porta lógica universal pois usando apenas NORs também é possível implementar todas as operações básicas NOT, AND e OR.

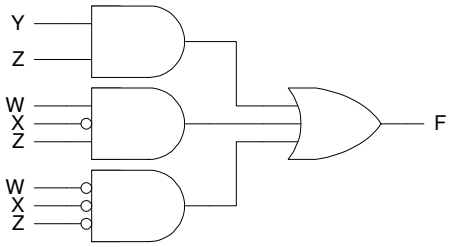
Uma vez que o NAND pode substituir qualquer das operações elementares, qualquer circuito pode ser construído utilizando apenas NANDs – basta substituir cada porta lógica AND, OR ou NOT pela versão correspondente com NANDs.

Ou seja: dado um circuito qualquer, para obter o mesmo circuito só com NANDs *substitui-se cada um dos AND, OR e NOT do circuito original pela versão correspondente com NANDs*, de acordo com a tabela indicada no ponto anterior.

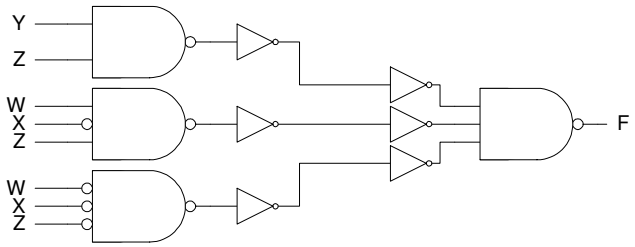
O método de síntese de circuitos usado normalmente – termos mínimos e simplificação através do agrupamento dos 1's no mapa de Karnaugh – conduz a circuitos que se apresentam na forma normalizada "soma de produtos". Em termos de implementação (e deixando de fora as negações) estes circuitos são compostos por dois "andares" de portas lógicas: um primeiro "andar" com várias portas AND, uma por cada termo produto, e um segundo andar composto por um OR entre o resultado desses mesmos ANDs.

Exemplo:

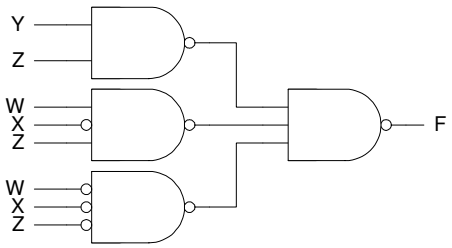
A expressão $F = YZ + W\overline{X}Z + \overline{W}\,\overline{X}\,\overline{Z}$ corresponde ao circuito



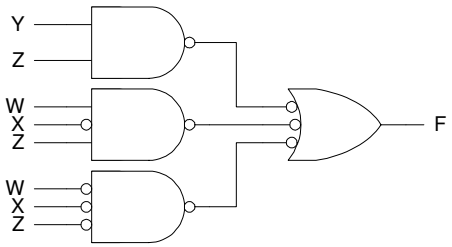
Substituindo os ANDs e OR por NANDs obtém-se o seguinte circuito:



Repare que existem agora situações em que aparecem pares de NOTs seguidos. Como $\overline{\overline{X}} = X$, então esses NOTs são redundantes e podem-se "cortar". Resulta assim um circuito que se obtém a partir do original usando a seguinte regra prática: *basta substituir os AND e OR por NANDs*. Mas atenção que esta regra empírica só se pode aplicar quando a expressão lógica está na forma normalizada "soma de produtos".

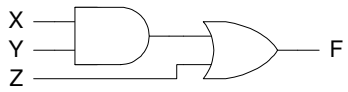


Esta substituição pode também ser feita graficamente considerando o símbolo usado para representar o NAND: partindo do circuito original, basta acrescentar "bolinhas" nas saídas dos ANDs e nas entradas do OR:

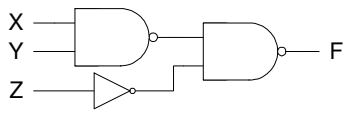


Relembre que um OR com as entradas negadas é o mesmo que um NAND.

Um outro exemplo. Considere o circuito correspondente a F = XY + Z.



Neste caso há um termo isolado, o Z, que necessita de ser negado para que a substituição do OR por NAND não afecte o resultado. Á parte da negação, o circuito transformado em NANDs será então:



Em suma: dado um circuito na forma soma de produtos, pode-se implementar o mesmo circuito só com NANDs

- substituindo os ANDs e OR por NANDs;
- negando os termos isolados (ou retirando a negação se o termo isolado já estiver, ele próprio, negado).

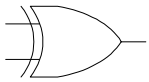
7.3. OU-exclusivo (XOR)

O **XOR** (eXclusive OR) representa uma operação lógica cujo resultado é '1' quando apenas um dos operandos é '1'. Pode também ser visto como uma operação que dá '1' se as duas variáveis de entrada tiverem valores diferentes.

Sendo assim, o XOR tem a seguinte tabela de verdade:

X	Y	F
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

O XOR representa-se algebricamente por $X \oplus Y$, e esquematicamente pelo seguinte símbolo:



Tendo em conta a sua tabela de verdade, o XOR pode ser também definido como:

$$X \oplus Y = \overline{X}Y + X\overline{Y}$$

O XOR é uma operação comutativa e associativa. Ou seja, verifica-se que:

- $X \oplus Y = Y \oplus X$ e
- $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$

Outras igualdades interessantes:

- $X \oplus 0 = X$ – '0' é elemento neutro do XOR;
- $X \oplus 1 = \overline{X}$ – o XOR com '1' funciona como uma negação.

O **XNOR** (eXclusive NOR) tem a tabela de verdade anexa. Representa uma operação lógica que dá '1' quando as duas variáveis têm o mesmo valor (chama-se *equivalência*). Representa-se algebricamente por $\overline{X \oplus Y}$ e tem-se que $\overline{X \oplus Y} = \overline{X}\,\overline{Y} + XY$.

X	Y	F
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

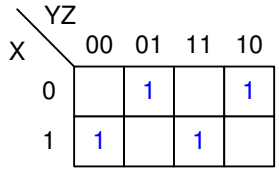
7.4. Função ímpar

A função $F = X \oplus Y \oplus Z$ correspondente ao XOR entre três variáveis e tem a tabela de verdade anexa. Repare que o resultado da função é '1' para as *combinações de entrada com um número ímpar de '1's*. Diz-se por isso que é a *função ímpar*.

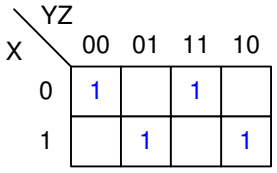
X	Y	Z	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Como se pode facilmente verificar, a sua negação – $F = \overline{X \oplus Y \oplus Z}$ – dá resultado '1' para as combinações que têm um número par '1's. A negação da *função impar* será portanto a *função par*.

O mapa de Karnaugh destas funções evidencia um outro aspecto importante: depois de serem colocados os '1's no mapa, verifica-se que não existem '1's localizados em casas adjacentes, logo não se podem fazer agrupamentos. A expressão algébrica resultante da soma de termos mínimos não é simplificável.

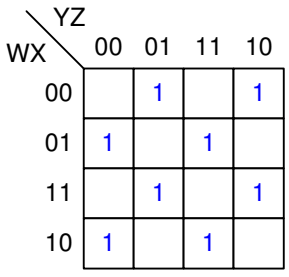


$F = X \oplus Y \oplus Z$

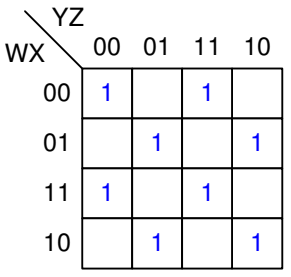


$F = \overline{X \oplus Y \oplus Z}$

O mesmo "padrão em xadrez" é exibido para no caso de serem 4 variáveis: a função ímpar caracteriza-se por dar '1' para as combinações que têm um número ímpar de 1's (ex: 0001, 0111) e a função par por dar '1' para as combinações que têm um número par de 1's (ex: 0000, 1100, 1111). Estas funções também não são simplificáveis.



$F = W \oplus X \oplus Y \oplus Z$



$F = \overline{W \oplus X \oplus Y \oplus Z}$

Nestes casos, a única forma de minimizar o número de portas lógicas utilizadas no circuito correspondente seria a utilização de portas lógicas XOR.

8. Projecto de Circuitos Combinatórios

8.1. Circuito na forma normalizada

Um circuito lógico combinatório é um circuito de N entradas e M saídas para o qual, em cada momento, *os valores lógicos nas saídas dependem apenas dos valores lógicos às entradas*.

O comportamento de um circuito combinatório pode ser descrito por uma tabela de verdade, que indica o valor lógico de cada uma das M saídas para cada uma nas 2^N combinações possíveis dos valores das variáveis de entrada.

O método de projecto de um circuito combinatório trata cada uma das variáveis de saída à vez. Por outras palavras, no caso de um circuito combinatório ter várias saídas, aplica-se o mesmo método de projecto para se obter uma expressão lógica para cada uma das saídas.

O método consiste nos seguintes passos:

1. Obter a tabela de verdade da saída;
2. Obter uma função simplificada usando um mapa de Karnaugh.

O resultado deste processo é um circuito em que cada saída aparece na forma normalizada "soma de produtos", podendo as variáveis em cada produto aparecer negadas ou não.

De uma forma geral, um circuito deste tipo implementa-se com:

- negações – para as variáveis que aparecem negadas nalguns dos termos;
- AND's – um para cada termo produto;
- OR – para "somar" os termos produto (*i.e.*, para combinar as saídas dos ANDs).

Em alternativa, tal como foi visto na secção 7 pode-se facilmente transformar este circuito de forma a usar apenas portas NAND ou NOR.

8.2. Minimizar o número de portas no circuito

No entanto, nem sempre a forma normalizada é a que permite implementar mais facilmente o circuito. O mapa de Karnaugh permite obter o circuito mais simples na forma "soma de produtos", mas que pode não ser necessariamente o circuito mais simples considerando outros aspectos.

Por exemplo, pode-se acrescentar um terceiro ponto ao método que é:

3. Minimizar o número de porta lógicas do circuito.

Para este ponto não será apresentada nenhuma metodologia específica. Fica ao critério de quem projecta o circuito investir mais ou menos tempo com vista a obter uma implementação mais simples, poupando no material e no tempo de montagem.

Uma possível técnica que pode ser usada é colocar termos comuns em evidência, criando uma topologia com mais níveis, mas que poderá levar a uma poupança de material. Por exemplo, a função:

$$F = \overline{W}YZ + \overline{W}X\overline{Y}\ ,$$

pode-se representar como:

$$F = \overline{W}\left(YZ + X\overline{Y}\right).$$

Assumindo que todas as portas lógicas disponíveis têm duas entradas, uma implementação directa da primeira opção corresponde a 7 portas lógicas, 4 ANDs, um OR e duas negações. A segunda opção poderia ser implementada com 6 portas lógicas – 3 ANDs, um OR e duas negações.

Se existissem também portas lógicas com três entradas, já poderia ser mais eficiente uma implementação da primeira opção: dois ANDs de três entradas, um OR e duas negações.

Outro exemplo:

$F = XY + X\overline{Z}$,

pode-se representar como

$F = X(Y + \overline{Z})$.

A primeira implementação requer três portas lógicas de duas entradas ao passo que a segunda apenas requer duas portas lógicas.

8.3. Utilização de XOR e XNOR

Ocasionalmente pode haver lugar à utilização de portas lógicas XOR ou XNOR. Por exemplo, a função

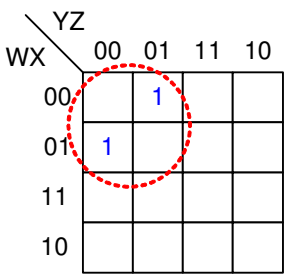
$$F = \overline{W}\,\overline{X}\,\overline{Y}\,Z + \overline{W}\,X\,\overline{Y}\,\overline{Z}\ ,$$

pode ser representar como:

$$F = \overline{W}\,\overline{Y}\left(\overline{X}Z + X\overline{Z}\right) = \overline{W}\,\overline{Y}(X \oplus Z)$$

Neste exemplo foi possível reduzir de uma forma substancial o número de portas lógicas necessárias.

Também nesta tarefa a representação visual pode ajudar. Veja, por exemplo, a representação do caso anterior num mapa de Karnaugh.



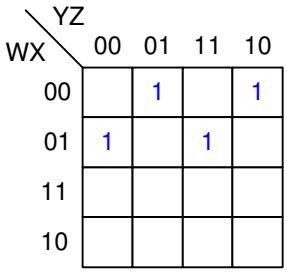
O grupo assinalado pelo tracejado a representa a tabela de verdade de um XOR entre as variáveis X e Z. Por outro lado, esse grupo, corresponde ao termo $\overline{W}\,\overline{Y}$; tem-se, portanto, um XOR entre X e Z quando $\overline{W}\,\overline{Y}$ dá '1', ou seja, $\overline{W}\,\overline{Y}(X \oplus Z)$.

Estes outros exemplos mostram situações triviais em que a inspecção visual ajuda a reconhecer a presença das funções par e ímpar.

Exemplo 1

Esta é uma situação em que a parte de cima (ou seja, a parte correspondente a W=0) é a função ímpar.

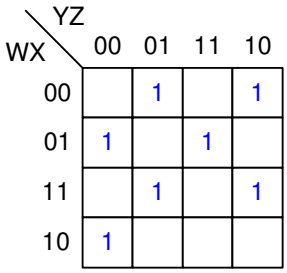
$$F = \overline{W}(X \oplus Y \oplus Z)$$



Exemplo 2

Esta é uma situação do mesmo género mas mais localizada: "quase" todos os '1's da função ímpar estão presentes; mas falta o que se situaria no grupo $W\overline{X}Z$. Logo, tem que se impedir a produção do '1' nessas circunstâncias:

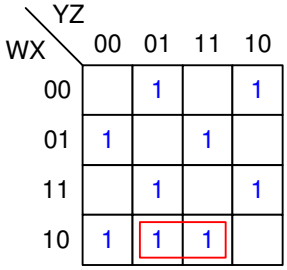
$$F = (W \oplus X \oplus Y \oplus Z)\overline{W\overline{X}Z}$$



Exemplo 3:

Finalmente, esta é uma situação em que estão presentes todos os '1's correspondentes à função ímpar e mais um termo que se pode agrupar.

$$F = (W \oplus X \oplus Y \oplus Z) + W\overline{X}Z$$



Lista de Revisões

Versão	Autor	Data	Comentários
001	JRG	Dez/2004	Versão draft; Publicação antecipada para o período de avaliações 2004/2005.
001a	TB	Jan/2005	Correcção de algumas gralhas.
002	TB	Set/2009	Revisão aprofundada do documento: correcção de gralhas, formatação e reformulação de texto, introdução de novas figuras.
002a	JPO, TB	Out/2009	Correcção de gralha na tabela de verdade da secção 3.3; Reformatação de duas equações (sec. 4.1 e sec. 7.2); Correcção de gralha ortográfica na sec. 4.4; Reformulação de texto num parágrafo da sec. 8.2.