# LIÇÃO 5

## COMBINANDO FUNÇÕES LÓGICAS

Nas duas lições anteriores estudamos as famílias lógicas CMOS e TTL, analisando suas características elétricas principais e a maneira como os componentes são fabricados através de alguns circuitos típicos.

Nesta lição continuaremos a estudar as funções lógicas, agora de uma forma mais completa. Analisaremos o que ocorre quando juntamos diversas funções lógicas, prevendo o que acontece com suas saídas. Os circuitos complexos, como os usados nos computadores, por exemplo, se aproveitam das operações complicadas que muitas portas lógicas podem realizar em conjunto. Assim, é de fundamental importância para nosso estudo saber analisar estas funções.

### 5.1 - As tabelas verdade

Os diversos sinais de entrada aplicados a uma função lógica, com todas as suas combinações possíveis, e a saída correspondente podem ser colocados numa tabela.

Nas colunas de entradas colocamos todas as combinações possíveis de níveis lógicos que as entradas podem assumir. Na coluna correspondente à saída colocamos os valores que esta saída assume em função dos níveis lógicos correspondentes na entrada.

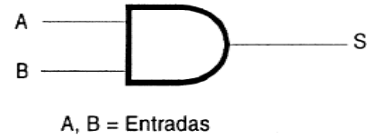


**Figura 1** - Funções ou porta AND (E) de duas entradas.

Vimos, desta forma, que a tabela verdade para uma função AND de duas entradas, como a representada na **figura 1**, pode ser dada por:

| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Veja que nas colunas de entrada (A e B) para termos todas as combinações possíveis, fazemos o equivalente à numeração binária de 0 a 3, já que:

| | | | |
|---|---|---|---|
| 0 | 0 | = | 0 |
| 0 | 1 | = | 1 |
| 1 | 0 | = | 2 |
| 1 | 1 | = | 3 |

Para uma tabela verdade feita para uma porta AND de 3 entradas teremos:

| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Neste caso, as combinações de níveis lógicos na entrada correspondem à numeração binária de 0 a 7 já que:

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | = | 0 |
| 0 | 0 | 1 | = | 1 |
| 0 | 1 | 0 | = | 2 |
| 0 | 1 | 1 | = | 3 |
| 1 | 0 | 0 | = | 4 |
| 1 | 0 | 1 | = | 5 |
| 1 | 1 | 0 | = | 6 |
| 1 | 1 | 1 | = | 7 |

O conhecimento da contagem binária facilita bastante a elaboração de tabelas verdades, quando todas as combinações possíveis de níveis lógicos em 2, 3 ou 4 entradas devam ser estudadas.

Assim, uma vez que o leitor conheça o comportamento das principais funções, sabendo o que ocorre na saída de cada uma quando temos determinadas entradas e sabendo elaborar tabelas verdades, fica fácil combinar funções e saber o que acontece em suas saídas.

### 5.2 - Lógica Combinacional

Vamos partir de um exemplo simples de lógica combinacional usando tabelas verdades para saber o que ocorre na sua saída, com o circuito da **figura 2**.

Este circuito faz uso de uma porta AND, um inversor e uma porta OR. O resultado desta configuração é uma função combinacional com três entradas e uma saída.

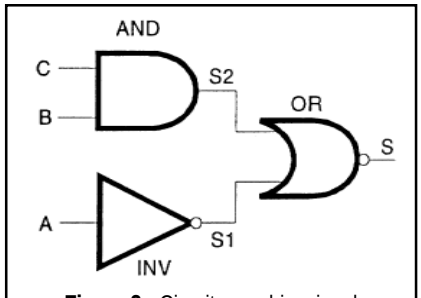


**Figura 2** - Circuito combinacional simples com três entradas.

Para elaborar a tabela verdade para este circuito e assim determinarmos todas as saídas possíveis em função das entradas, devemos levar em conta que ele é formado por duas etapas.

Na primeira etapa temos a porta AND e o inversor, enquanto que na segunda etapa temos a porta OR. Isso significa que as saídas dos circuitos da primeira etapa, que chamaremos de $S_1$ e $S_2$ são a entrada da segunda etapa.

Temos então de levar em conta estas saídas na elaboração da tabela verdade que terá no seu topo as seguintes variáveis:

| A | B | C | $S_1$ | $S_2$ | S |
|---|---|---|---|---|---|

A,B e C são as entradas dos circuitos. $S_1$ e $S_2$ são pontos intermediários do circuito que precisam ser analisados para a obtenção de S, que é a saída final do circuito.

Começamos por colocar em A, B e C todas as suas condições possíveis, ou todas as combinações de níveis lógicos que podem ser aplicadas ao circuito:

| A | B | C | $S_1$ | $S_2$ | S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | | | |
| 0 | 1 | 1 | | | |
| 1 | 0 | 0 | | | |
| 1 | 0 | 1 | | | |
| 1 | 1 | 0 | | | |
| 1 | 1 | 1 | | | |

O passo seguinte é colocar os valores possíveis de $S_1$, que corresponde à saída do inversor.

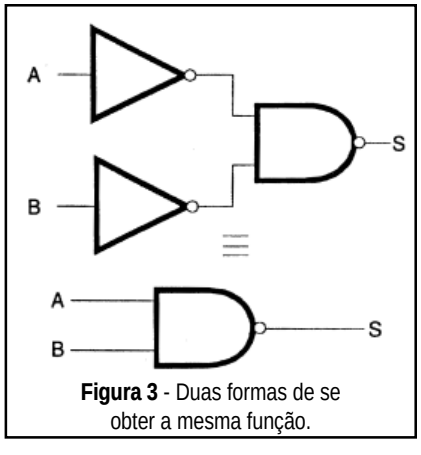


**Figura 3** - Duas formas de se obter a mesma função.

Sabemos que a tabela verdade para o inversor é:

| A | S |
|---|---|
| 0 | 1 |
| 1 | 0 |

Ora, como em nosso caso A é a entrada do inversor e $S_1$ é sua saída, podemos partir para a determinação de toda a coluna $S_1$ simplesmente invertendo os valores de A, da seguinte forma:

| A | B | C | $S_1$ | $S_2$ | S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | | |
| 0 | 0 | 1 | 1 | | |
| 0 | 1 | 0 | 1 | | |
| 0 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | | |
| 1 | 0 | 1 | 0 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 1 | 0 | | |

Para encontrar os valores da coluna $S_2$ devemos observar que ela corresponde à tabela verdade da função AND onde as entradas são B e C e a saída é $S_2$.

| B | C | $S_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Temos então:

| A | B | C | $S_1$ | $S_2$ | S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 1 | |

Finalmente, levando em conta que $S_1$ e $S_2$ são entradas de uma porta OR de duas entradas cuja saída é S, podemos elaborar a coluna final de saídas (S)

| $S_1$ | $S_2$ | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Resultando na seguinte tabela:

| A | B | C | $S_1$ | $S_2$ | S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

Trata-se de uma função bastante interessante que pode ser definida como “a que fornece uma saída alta somente quando a entrada A estiver no nível baixo, não importando as demais entradas ou ainda quando as três entradas estiverem no nível alto”.

**5.3 - Como Projetar Um Circuito Combinacional**

O problema de saber o que acontece com a saída de um circuito formado por muitas funções lógicas quando suas entradas recebem diversas combinações de sinais não é o mais importante para o projetista de equipamentos digitais. Na verdade, muito mais importante que este procedimento é justamente fazer o contrário, ou seja, projetar um circuito que, em função de determinados sinais de entrada, forneça exatamente na saída o que se deseja.

O projeto de um circuito que tenha uma determinada função envolve um procedimento de síntese em algumas etapas.

Na primeira etapa deve ser definido o problema, estabelecendo-se exatamente qual a função a ser executada, ou seja, quais as entradas e quais as saídas.

Numa segunda etapa, coloca-se o problema numa tabela verdade ou ainda na forma de equações lógicas.

O procedimento que abordaremos neste curso será basicamente o da obtenção das funções a partir das tabelas verdade e das equações lógicas.

Finalmente, numa terceira etapa, obtemos o circuito que exercerá as funções desejadas.

Na terceira etapa, um ponto importante consiste na minimização do circuito, já que na maioria dos casos

pode-se implementar a mesma função de muitas formas diferentes como atesta o circuito simples apresentado na **figura 3**.

Veja que podemos ter o mesmo circuito com quantidades de portas diferentes, na prática devemos sempre levar este fato em conta. Não é apenas o número de portas que determinará a configuração final, mas sim, seu custo e a eventual utilização em outras partes do circuito.

Por exemplo, se o circuito já estiver usando dois inversores dos seis disponíveis num circuito integrado e a nossa função tiver uma solução um pouco maior, mas que use estes inversores, será interessante adotá-la para aproveitar os inversores ociosos.

A seguir daremos um exemplo de como obter os circuitos a partir de uma tabela verdade.

**a) Passo 1 - Determinação das equações lógicas**

Lembramos que para as funções estudadas temos as seguintes representações:

Função E (AND)

$Y=A.B$

Função Não E (NAND)

$Y=\overline{A.B}$

Função OU (OR)

$Y=A+B$

Função Não OU (NOR)

$Y=\overline{A+B}$

Função Não (NOT) ou inversor

$Y=\overline{A}$

Função ou exclusivo
(Exclusive OR)

$Y=A(+)B$

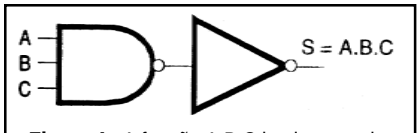


**Figura 4** - A função A.B.C implementada.

Vamos tomar como exemplo a tabela verdade abaixo para determinar a função lógica correspondente:

| A | B | C | Y | linha |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 1 | 3 |
| 1 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 1 | 5 |
| 1 | 0 | 1 | 0 | 6 |
| 0 | 1 | 1 | 0 | 7 |
| 1 | 1 | 1 | 1 | 8 |

Indicamos a linha na última coluna de modo a facilitar as explicações seguintes.

Observamos que temos saídas no nível 0 para as linhas 0, 3, 5 e 6, enquanto para as linhas 1, 2, 4 e 7 temos saídas 1.

Isso quer dizer que teremos a função OU para as linhas cuja saída é 1 que podem ser encaradas como operações OR com tabelas que teriam 1 na saída apenas nas linhas 1, 2, 4 e 7, conforme mostrado a seguir:

| A B C Y | | A B C S1 | | A B C S2 | | A B C S3 | | A B C S4 |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | 0 0 0 0 | | 0 0 0 0 | | 0 0 0 0 | | 0 0 0 0 |
| 0 0 1 1 | | 0 0 0 1 | | 0 0 1 0 | | 0 0 1 0 | | 0 0 1 0 |
| 0 1 0 1 | | 0 1 0 0 | | 0 1 0 1 | | 0 1 0 0 | | 0 1 0 0 |
| 0 1 1 0 | = | 0 1 1 0 | + | 0 1 1 0 | + | 0 1 1 0 | + | 0 1 1 0 |
| 1 0 0 1 | | 1 0 0 0 | | 1 0 0 0 | | 1 0 0 1 | | 1 0 0 0 |
| 1 0 1 0 | | 1 0 1 0 | | 1 0 1 0 | | 1 0 1 0 | | 1 0 1 0 |
| 1 1 0 0 | | 1 1 0 0 | | 1 1 0 0 | | 1 1 0 0 | | 1 1 0 0 |
| 1 1 1 1 | | 1 1 1 0 | | 1 1 1 0 | | 1 1 1 0 | | 1 1 1 1 |

Isso nos permite escrever as equações lógicas para cada uma das quatro tabelas da seguinte forma:

$S_1 = \overline{A} . \overline{B} . C$ que
corresponde a A=0, B=0 e C=1

$S_2 = \overline{A} . B . C$ que
corresponde a A=0, B=1 e A=0

$S_3 = A . \overline{B} . \overline{C}$ que
corresponde a A=1, B=0 e C=0

$S_4 = A . B . C$ que
corresponde a A=1, B=1 e C=1

Como a saída S é a combinação das quatro funções temos:

$S = S_1 + S_2 + S_3 + S_4$

Substituindo pelos valores encontrados teremos:

$S = \overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.\overline{C} + A.B.C$

Esta é então a função lógica que representa a tabela verdade que propusemos como parte inicial do problema e para a qual devemos encontrar um circuito equivalente.

**Passo 2 - Implementação dos Circuitos Combinacionais**

Conforme estudamos em lições anteriores, é possível usar as portas NAND e NOR como blocos lógicos universais a partir dos quais podemos elaborar qualquer outra função ou mesmo funções mais complexas.

Para exemplificar vamos analisar uma função um pouco mais simples do que a obtida no passo anterior. Tomemos a expressão:

$S = A . B . \overline{C} + \overline{A} . \overline{B} . C$

Podemos tentar implementá-la usando portas NAND e eventualmente inversores, já que a barra sobre cada letra indica sua negativa, conforme estudamos.

A operação (.) pode ser realizada utilizando-se uma porta NAND que ligada a um inversor nos fornece uma porta AND.

Assim, conforme a **figura 4**, podemos implementar A.B.C usando uma porta NAND de 3 entradas e um inversor.

Veja na **figura 5** como a operação A.B.C pode ser implementada.

A soma (+) pode ser implementada com uma porta OR ligada a dois inversores, **figura 6**.


**Figura 5** - Implementação da função A.B.C

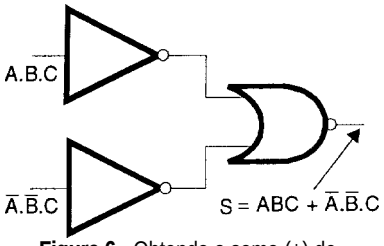


**Figura 6** - Obtendo a soma (+) de duas expressão lógicas.

Combinando os três circuitos podemos chegar à configuração final desejada, **figura 7**.

Veja que a inversão da inversão usada no circuito anterior nos leva ao circuito original. Isso significa que podemos simplificar a configuração eliminando as duplas inversões em série. Isso nos leva à configuração final do circuito mostrada na **figura 8**.

Logo, quando temos uma expressão formada pela soma de produtos, podemos usar portas NAND sem a necessidade de inversores, bastando apenas lembrar duas propriedades:

As combinações de entrada podem ser aplicadas a portas NAND.

As saídas das portas NAND podem ser aplicadas à entrada de uma segunda porta NAND obtendo-se na saída a função desejada.

Vamos agora fazer uma tentativa de implementar uma função usando portas NOR, o que será escolhido quando tivermos um produto de somas.

Tomemos como exemplo a função:

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

As somas podem ser obtidas facilmente a partir de portas NOR com

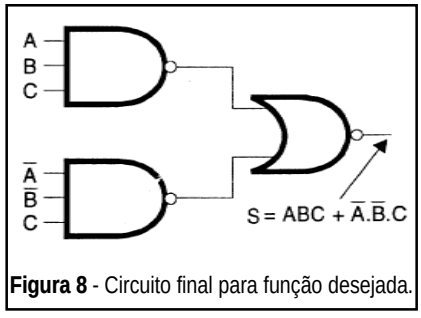


**Figura 8** - Circuito final para função desejada.

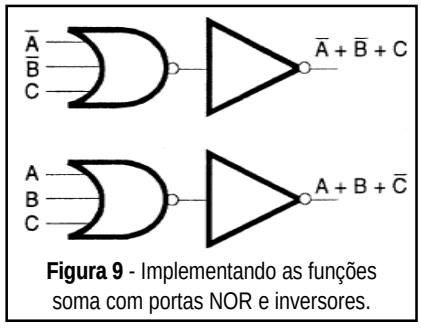


**Figura 9** - Implementando as funções soma com portas NOR e inversores.

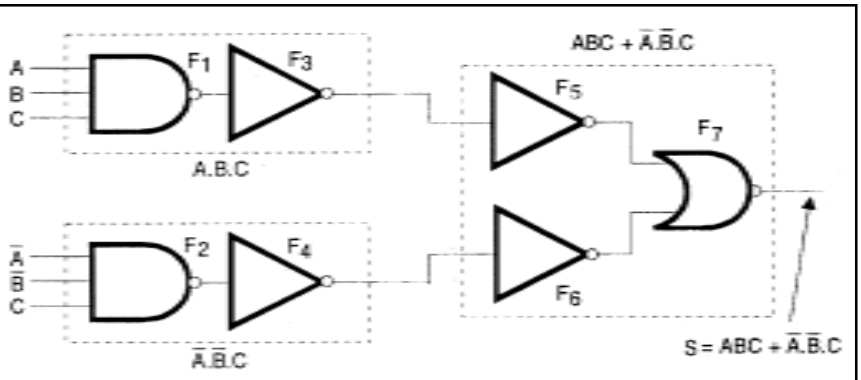


**Figura 7** - A função S=$\overline{A}\overline{B}$C+ABC de forma que ainda pode ser minimizada.

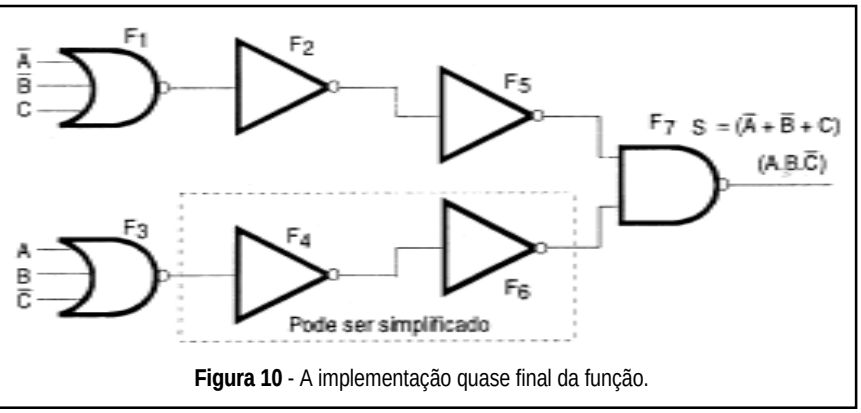


**Figura 10** - A implementação quase final da função.

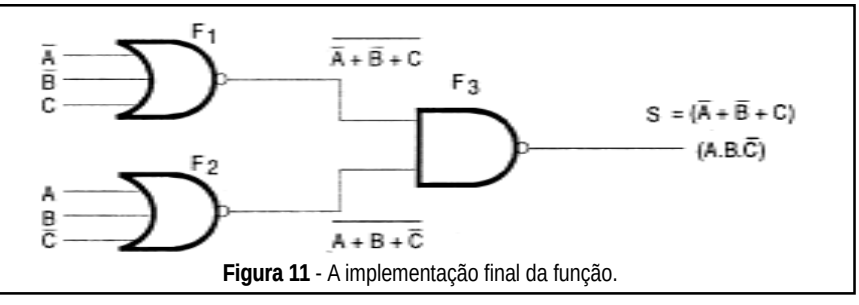


**Figura 11** - A implementação final da função.

as saída aplicadas a um inversor. A negação de NOR é OR. O circuito equivalente para três entradas é mostrado na **figura 9**.

O produto das duas somas é obtido com dois inversores aplicando os sinais a uma outra porta OR, ou seja, a uma outra configuração NOR.

Como nas duas linhas de sinais temos inversores em série, e o inversor do inverso de um nível lógico é ele mesmo, podemos simplificar o circuito eliminando todos os inversores.

Isso nos permite chegar à configuração final que é mostrada na **..**

Assim, se quisermos implementar uma função que consiste num produto de somas, basta seguir dois procedimentos básicos:

Aplicar as entradas correspondentes a cada soma a uma porta OR que pode ser obtida associando-se uma porta NOR a uma inversor.

Aplicar as saídas obtidas nas funções que devem ser multiplicadas a inversores que são ligados às entradas de uma porta OR final, também obtida com a associação de um inversor a uma porta NOR.

Como os inversores em série se anulam, eles podem ser eliminados e o circuito implementado utilizando-se apenas portas NOR.

É possível resolver o problema de implementar circuitos combinacionais reduzindo as funções a produtos de somas ou ainda a soma de produtos,

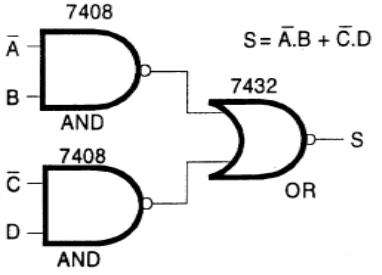


**Figura 12** - Dois tipos diferentes de portas são usados neste circuito.

casos em que podemos trabalhar com funções NAND ou NOR.

Como as duas soluções levam aos mesmos resultados, num projeto prático é interessante analisar as configurações obtidas para um problema nos dois casos. Adota-se então a solução que utilizar menos circuitos ou que for mais conveniente, por exemplo, aproveitando portas ociosas de um circuito integrado já utilizado no mesmo projeto com outras finalidades.

### 5.4 - SIMPLIFICANDO E MINIMIZANDO

Uma consequência da possibilidade de construir funções complexas a partir de portas básicas como OR e AND (OU e E) é a otimização de um projeto aproveitando poucos tipos de circuitos integrados básicos.

Assim, se tivermos uma função que seja obtida utilizando-se portas AND e OR como a mostrada na figura 12, ela terá o inconveniente de precisar de dois tipos diferentes de circuitos integrados.

Se quisermos esta função com circuitos TTL, por exemplo, aproveitaremos três das três portas de três entradas de um circuito 7411 e também precisaremos aproveitar uma das quatro portas OR de duas entradas de um circuito integrado 7432.

Evidentemente, estaremos usando dois circuitos integrados, desperdiçando 1/3 de um e 3/4 do outro.

Podemos simplificar consideravelmente este circuito se usarmos apenas portas NAND com a configuração equivalente mostrada na figura 13.

Este circuito, que apresenta a mesma função do anterior, usa as três portas de um circuito integrado 7410. Utilizamos apenas um circuito integrado que é totalmente aproveitado, sem nenhuma parte ociosa.

### 5.5 - DIAGRAMAS DE KARNAUGH

Um processo bastante interessante para representar uma tabela verdade e a partir dela obter uma simplificação dos circuitos utilizados para sua implementação é o que faz uso dos chamados diagramas ou mapas de Karnaugh.

O diagrama de Karnaugh consiste numa tabela retangular com número de quadros que corresponde a 2 elevado ao expoente N, onde N é o número de variáveis do circuito.

Cada variável lógica ocupa no gráfico metade da sua extensão e seu complemento ocupa a outra metade.

Na **figura 13** temos o modo como são elaborados os diagramas de Karnaugh para 1, 2 e 3 variáveis, com as expressões lógicas correspondentes a cada caso.

Estas expressões são obtidas de uma forma muito semelhante à usada no conhecido joguinho de "batalha naval" onde a posição de cada "tiro" é dada por duas coordenadas, uma correspondente às linhas e outra às colunas.

Na **figura 15** mostramos, como exemplo, de que modo um diagrama de Karnaugh de 4 variáveis pode ser obtido com a inclusão dentro de cada quadro da expressão correspondente. No diagrama (b) da **figura 14** os quadros foram preenchidos com os valores 0 e 1 correspondentes às entradas. Este diagrama é chamado também de diagrama de Veitch. Uma observação importante em relação a esta representação por 0 e 1 é que

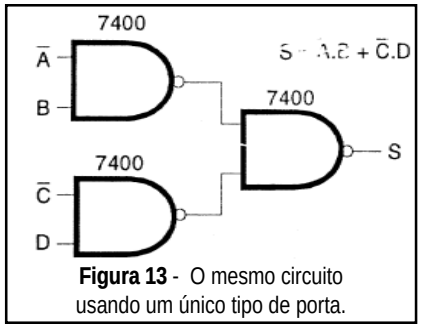


**Figura 13** - O mesmo circuito usando um único tipo de porta.

cada quadro difere do adjacente em apenas um dígito.

Dizemos que são adjacentes os termos que estão à direita e à esquerda de cada quadro e também os que estão acima e abaixo. Também são adjacentes os que estiverem na mesma fila, mas um na primeira coluna e outro na última.

Na **figura 16** temos um mapa com a identificação das adjacências.

Assim, o que fazemos é plotar a tabela verdade da função que desejamos implementar num mapa de Karnaugh com o que será possível identificar melhor as adjacências e assim fazer as simplificações.

Para que o leitor entenda como "funciona" o mapa de Karnaugh numa simplificação de uma função, vamos tomar como exemplo a função que é dada pela seguinte tabela verdade:

| A | B | S |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Desejamos expressar esta tabela como a soma de produtos, o que significa que os valores adjacentes que

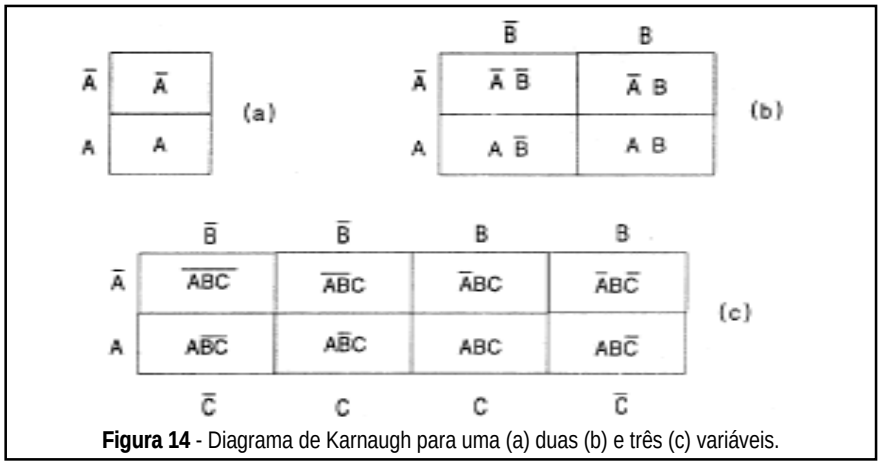


**Figura 14** - Diagrama de Karnaugh para uma (a) duas (b) e três (c) variáveis.

devemos procurar na tabela são os "1". Se fôssemos expressar esta função como o produto de uma soma, os valores considerados seriam os "0" e o procedimento final seria o mesmo.

Construímos então o Diagrama de Karnaugh para esta tabela conforme a **figura 17**.

A partir deste diagrama nosso próximo passo consiste em tentar fazer simplificações que possam levar a circuitos mais simples na implementação.

A idéia é agrupar os termos adjacentes iguais, havendo para isso diversas possibilidades que são apresentadas na **figura 18**.

A primeira possibilidade mostrada em (a) nos leva a uma soma de três produtos, cada qual obtido pela intersecção da linha com a coluna em que está o "1" correspondente.

Assim, o primeiro está na coluna que intercepta A-0 com B-0. Ora, o valor zero na indexação indica inversão, portanto, isso significa que o primeiro fator de nosso produto será:

$$\overline{A}.\overline{B}$$

O segundo "1" a ser considerado está na coluna A=1 e B=0, portanto, temos A invertido e B sem inversão, o que nos leva ao segundo fator de nosso produto:

$$\overline{A}.B$$

Finalmente, o terceiro "1" a ser considerado está na linha A=1 e B=1, o que significa um fator com A multiplicado por B sem inversões ou:

$$A.B$$

Como devemos expressar a função na forma de uma soma de produtos fazemos:

$$S = \overline{A}.\overline{B} + A.\overline{B} + A.B$$

Para o segundo caso (b) temos uma simplificação maior, já que agrupamos os dois "1" da primeira linha de modo que podemos adotar para ele:

$$\overline{A}$$

Para o outro valor "1" que está na casa que corresponde à intersecção de A-1 com B-1 vale a soma (sem inversão):

$$A + B$$

A expressão final na forma de um produto de somas será então:

$$\overline{S} = A + B.A$$

Da mesma forma chegamos à simplificação (b) que permite a expressão mais simples, pois conseguimos juntar três casas adjacentes.

Raciocinando da mesma forma chegamos à expressão:

$$\overline{S} = A + B$$

O procedimento que vimos como exemplo envolveu uma função simples com apenas duas variáveis de entrada.

No entanto, o mesmo procedimen-

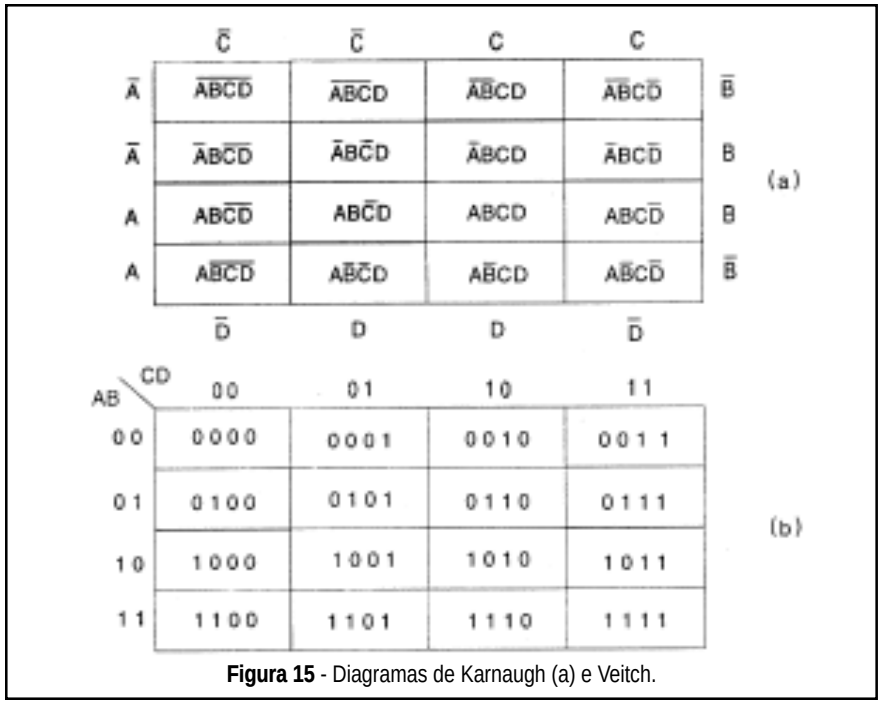


**Figura 15** - Diagramas de Karnaugh (a) e Veitch.

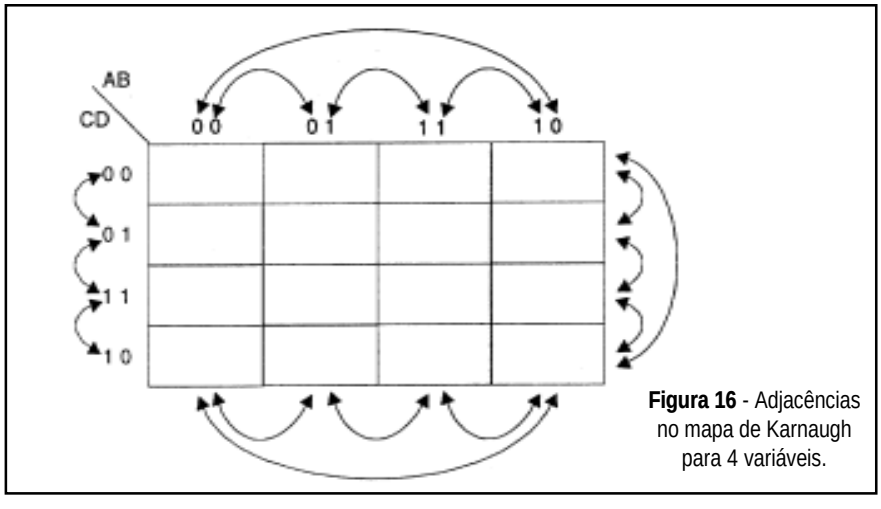


**Figura 16** - Adjacências no mapa de Karnaugh para 4 variáveis.

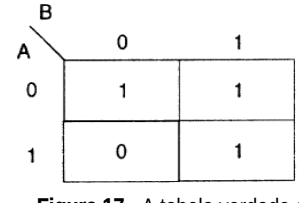


**Figura 17** - A tabela verdade é plotada no Mapa de Karnaugh.

to é válido para qualquer número de variáveis. Os leitores interessados em aprofundar-se neste estudo devem procurar treinar os procedimentos indicados, trabalhando com funções cada vez mais complexas.

## CONCLUSÃO

O espaço disponível para nosso curso não permite um aprofundamento maior neste assunto e um certo treino se faz necessário para o domínio das técnicas envolvidas. Assim, para os leitores interessados no tema, sugerimos a procura de literatura complementar. Mostramos os procedimentos lógicos que permitem trabalhar com as funções de modo a chegar aos circuitos.

Assim, uma tabela verdade que tenha qualquer combinação de entradas que nos leve a qualquer combinação de saída pode ser elaborada na prática com funções básicas (NOR e NAND) e isso não exige que se "quebre a cabeça".

Conhecendo os procedimentos para resumir tudo em produto de somas e soma de produtos e também o uso dos mapas de Karnaugh para simplificação, obteremos configurações simples que facilitam qualquer projeto.

## QUESTIONÁRIO

1. Os valores combinados de todas as entradas e a saída correspondente podem ser colocados numa tabela denominada:

a) Mapa de Karnaugh
b) Diagrama de Veitch
c) Tabela verdade
d) Produto de somas

2. A tabela verdade abaixo, corresponde à qual função:

A B S
0 0 1
0 1 1
1 0 1
1 1 0

a) AND (E)
b) NAND (Não-E)
c) OR (OU)
d) NOR (Não-OU)

3. Qualquer circuito lógico pode ser implementado utilizando-se que funções básicas?

a) NAND e inversores
b) NAND e NOR
c) OR e Inversores
d) AND e Inversores

4. Para implementar um circuito que corresponda a uma função dada por uma soma de produtos usamos quais funções lógicas?

a) Portas NAND
b) Inversores
c) Portas OR
d) Não é possível fazer isso

5. Se numa implementação lógica precisarmos usar inversores em série, o que podemos fazer com eles?

a) Ligá-los à portas AND
b) Colocá-los em paralelo
c) Inverter suas saídas
d) Eliminá-los ■

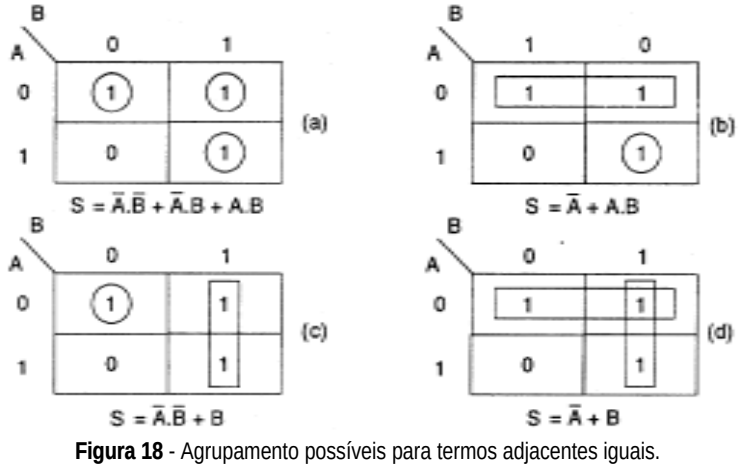


**Figura 18** - Agrupamento possíveis para termos adjacentes iguais.

Respostas: 1-C, 2-B, 3-B, 4-A, 5-D