Agradecimentos

Quero exprimir um especial agradecimento ao Professor Pimenta Rodrigues pelos ensinamentos que me transmitiu, bem como pelo estímulo que me deu para a publicação deste trabalho.

Agradeço também aos elementos do Centro de Cálculo do ISEL pelo seu contributo, quer através de críticas e sugestões, quer na correcção do próprio texto.

PREFÁCIO

O objectivo deste texto é providenciar informação que permita entender os fundamentos e o desenho dos actuais sistemas digitais com especial ênfase para os processadores. Este trabalho é baseado nos fascículos de Sistemas Digitais do Professor Pimenta Rodrigues, seguindo os mesmos critérios pedagógicos no que diz respeito ao carácter pragmático no tratamento e reflectindo no a experiência de vários anos de leccionação destes temas no ISEL com a inevitável adaptação às novas tendências tecnológicas e dando especial atenção a formas de implementação e arquitecturas de computadores.

Enquanto que os fundamentos da lógica digital pouco ou nada se têm modificado ao longo dos últimos anos, o mesmo não é verdade no que diz respeito à aplicação destes fundamentos. A a evolução tecnológica nesta área tem modificado a tónica a colocar em cada assunto, tanto na área do hardware, pelo surgimento de estruturas complexas programáveis e linguagens de descrição hardware, como no desenho das arquitecturas dos computadores devido ao surgimento de novos paradigmas na área das arquitecturas de processamento.

O aumento da capacidade de integração de silício observado nos últimos anos e as solicitações do mercado de computadores, aliados ao desenvolvimento tecnológico na área das arquitecturas de microprocessadores, têm permitido desenvolver processadores cada vez mais sofisticados, eficientes e, consequentemente, mais complexos. Esta complexidade tem criado sérias dificuldades no ensino básico de arquitecturas de computadores, dificuldades que se prendem essencialmente com: a arquitectura interna, normalmente paralela (*pipe line*); o conjunto das instruções extenso; bus de controlo com *timing* complexo; e uma largura do bus de dados e endereços que dificulta a realização de trabalhos laboratoriais. Este facto tem levado a que nos últimos anos o ensino de arquitecturas de computadores, na maioria das escolas de engenharia, se tenha refugiado na utilização de simuladores. Esta utilização, embora apresente muitos pontos positivos, não permite o contacto do aluno com estruturas de hardware mais elaboradas e, consequentemente, inibe-o de incursões futuras nesta área. Por outro lado, o simulador não podendo ser muito complexo nesta fase da aprendizagem, esconde pontos essenciais para o desenho de sistemas baseados em microprocessadores, como sejam os sinais de sincronização e a temporização do bus.

Perante este facto era necessária uma abordagem que permitisse continuar a transmitir os conceitos básicos ligados às novas arquitecturas sem que tivéssemos de lidar com grandes níveis de complexidade. Esta abordagem passava por criar uma plataforma onde fosse possível: especificar e implementar um CPU com uma arquitectura simples; inserir esse CPU num sistema obedecendo aos vários sinais aí disponíveis; escrever programas em linguagem máquina com o conjunto das instruções proposto; carregar em memória e testar esses mesmos programas; estudar os dispositivos mais vulgares que constituem uma arquitectura baseada em microprocessadores, especificando, implementando e inserindo-os na arquitectura.

Com este objectivo, foi criada uma plataforma para o ensino de arquitecturas de computadores envolvendo os seguintes elementos:

- Processador;
- Compilador (assembler);
- Sistema didáctico para desenvolvimento e teste de programas e periféricos;
- Ferramenta de teste (*debugger*);

O desenho da plataforma teve como principais preocupações:

- Possibilitar ao aluno especificar e implementar um processador;
- Dar suporte ao carregamento de programas em linguagem máquina através de um DMA, utilizando um teclado hexadecimal e um mostrador de LEDs;
- Efectuar teste de programas observando a execução das instruções ciclo a ciclo;
- Observar em osciloscópio a evolução temporal dos sinais que constituem o bus;
- Desenvolver e inserir periféricos na arquitectura existente.

Para dar suporte numa fase mais avançada de desenvolvimento, a plataforma permite que o carregamento e teste dos programas se façam através de um canal USB, a partir do computador pessoal, utilizando uma aplicação específica. No que diz respeito ao processador, foi definida uma interface, envolvendo *clock*, *reset*, sinais de protocolo e bus a 16 bits, com endereço multiplexado com dados. Esta estrutura poderá ser utilizada por qualquer processador desde que este obedeça às especificações estabelecidas, permitindo a implementação de CPUs com diferentes arquitecturas e usando diferentes técnicas, ou seja, recorrendo a emulação através de um microcontrolador ou *hardwired* utilizando um CPLD (*Complex Programmable Logic Device*).
Como actualmente a maioria dos computadores são sistemas electrónicos digitais, compreender e utilizar uma estrutura de computação implica o domínio das várias componentes de análise e síntese deste tipo de sistemas, razão pela qual começarmos com o estudo dos sistemas electrónicos digitais.
Quanto à forma de exposição, optamos por introduzir cada novo elemento, pela necessidade criada pelos desafios que vamos propondo. Este método, que é referido como aprendizagem baseada em problemas, tende, no nosso entender, a tornar a exposição mais aliciante e objectiva.
O tema **Fundamentos** abrange os primeiros 9 capítulos. O capítulo 1 é dedicado ao estudo dos fundamentos da lógica binária, o capítulo 2 é dedicado ao estudo dos circuitos integrados, o capítulo 3 trata o número e as operações aritméticas, os capítulos 4 e 5 abordam os módulos funcionais, e os capítulos 6 a 9 descrevem a organização de circuitos sequenciais a partir de células de memória unitárias, introduzem o conceito de modularidade na análise e síntese de sistemas, assim como métodos de projecto e grafismos para representação de algoritmos e de máquinas algorítmicas.
O tema **Microprocessadores** abrange os capítulos 10 a 16.
O tema **Periféricos** abrange os capítulos 17 a 19. No capítulo 17, estudam-se os portos de entrada e saída, o capítulo 18 trata os temporizadores e contadores e no capítulo 19 estuda-se o mecanismo de interrupções do CPU e os periféricos de gestão de pedidos de interrupção.

SISTEMA ELECTRÓNICO DIGITAL

A visão que o humano tem da natureza é a de um espaço contínuo, razão pela qual ao longo dos anos se tenham utilizado modelos contínuos para reprodução de fenómenos físicos como por exemplo o som nos discos de vinil. Foi nas primeiras décadas do século XX que teve início a descrição e quantificação sob a forma de um número de grandezas físicas sendo o MP3 um exemplo disso. Os sistemas que assim passaram a operar designaram-se por sistemas numéricos e por associação entre número e dígito, designaram-se por sistemas digitais que é a designação universalmente adoptada.

Ao falarmos de número falamos obrigatoriamente de base de numeração. Se por exemplo a base for 2 (binária) permite que qualquer grandeza seja representada utilizando apenas dois algarismos (dígitos). É nesta lógica que chegamos aos sistemas electrónicos digitais actuais.

Um sistema electrónico digital é um circuito electrónico que baseia o seu funcionamento na lógica binária e em que a informação é representada por apenas dois níveis discretos de tensão eléctrica. Os sistemas electrónicos digitais são essencialmente constituídos por malhas de interruptores electrónicos (díodos e transístores) que interactuam segundo uma determinada lógica. Estes sistemas têm dois tipos distintos de comportamento: comportamento combinatório que se caracteriza por apresentar sempre o mesmo comportamento quando sujeitos a uma mesma combinação das entradas; e o comportamento sequencial, que devido ao facto de conter elementos de memória, apresenta diferentes comportamentos para uma mesma combinação das entradas.

1. CIRCUITOS LÓGICOS COMBINATÓRIOS

Diz-se que um circuito é combinatório quando apresenta para cada uma das combinações de entrada um mesmo comportamento, ou seja, não contém elementos de memória que lhe permitam reagir em função de factos anteriormente ocorridos.

1.1 Álgebra de Comutação

Com a vulgarização dos circuitos eléctricos rapidamente se atingiram níveis de complexidade que exigiam um tratamento que não fosse o da simples perspicácia de alguns especialistas sem qualquer garantia de que o circuito obtido teria a menor complexidade para o objectivo em vista. A solução para este problema, que se agudizou com o aparecimento das centrais telefónicas, teve em 1937 um contributo importante do cientista Claude Shannon, que na sua tese de Mestrado, adaptou os estudos do matemático George Boole às malhas de comutação (circuitos constituídos por elementos interruptores/comutadores). Na Figura 1-1 são mostrados exemplos de elementos interruptores e comutadores (mecânicos e electromecânicos) que constituem as malhas de comutação. Na posição de repouso do actuador, os contactos podem estar fechados (*Normally Closed*, NC) ou abertos (*Normally Open,* NO).

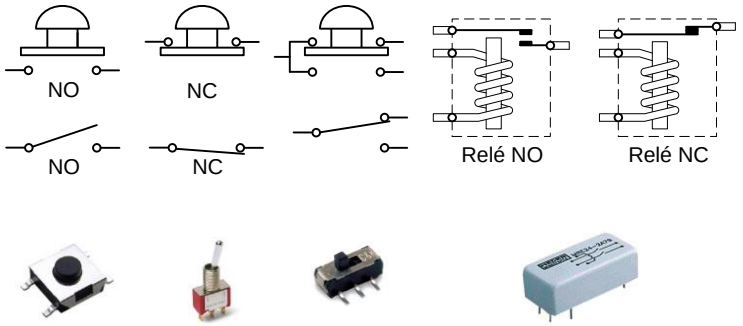


Figura 1-1

A razão de iniciarmos o estudo de sistemas digitais utilizando malhas de comutação baseado em interruptores mecânicos, deve-se ao facto de estes serem dispositivos binários, cujo princípio de funcionamento é de fácil compreensão. Nestas malhas poderemos estudar todas as propriedades e princípios dos operadores binários, os quais são directamente aplicáveis aos circuitos digitais electrónicos.

Como o interruptor só tem dois estados (activado ou desactivado), a utilização da álgebra de Boole no projecto de sistemas de comutação ou sistemas digitais encontra uma adaptação directa por se tratar de uma álgebra binária, permitindo utilizar toda a axiomática desenvolvida por Boole, na simplificação e validação deste tipo de sistemas.

A cada interruptor é atribuída uma designação, uma letra ou um nome que esteja de preferência relacionado com a função que este desenvolve no sistema.

Ao interligar os contactos de vários interruptores constitui-se o que se designa por malha de comutação.

Um conceito mais lato que o de interruptor, é o de actuador, que tem associado um ou mais contactos (interruptores), em que alguns deles estão normalmente fechados e outros normalmente abertos. Ou seja, quando um determinado actuador não estiver a ser actuado alguns dos seus contactos podem estar fechados e outros abertos.

Quanto ao relé, que também é um actuador, é um elemento de comutação electromecânico constituído por um ou mais interruptores de duas lâminas, (sendo uma das lâminas fixa e outra

flexível), uma bobina que quando percorrida por corrente eléctrica produz um campo magnético, que atrai as lâminas flexíveis abrindo ou fechando os vários interruptores que lhe estão associado.
Na figura 1-2, é mostrado um exemplo de uma malha de comutação constituída por três actuadores **A**, **B** e **C** que controlam o acender e apagar da lâmpada **L**.
Como podemos ver na Figura 1-2, o actuador **A** tem associado dois interruptores, um normalmente aberto e outro normalmente fechado.

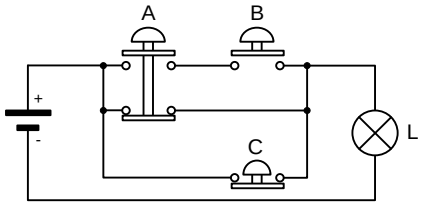


Figura 1-2

Determinar o comportamento do circuito é determinar para que combinações de A, B e C se estabelece um ou mais caminhos que leve a corrente a percorrer a lâmpada e assim acendê-la.
Podemos observar que a lâmpada L acende se actuarmos A e B em simultâneo, ou não actuarmos C ou não actuarmos A. Como se pode observar na Figura 1-2 existem outras combinações que levam a lâmpada a acender.

1.2 Formas de representação de uma função booleana

Dado que os actuadores só podem tomar um de dois estados (actuado ou não actuado), poderemos associar um actuador a uma variável booleana, fazendo corresponder o valor lógico verdadeiro da variável ao estado actuado e o valor lógico falso da variável a estado não actuado. Existem várias formas de representar o comportamento de um sistema, sendo cada uma dessas formas mais ou menos esclarecedora do seu comportamento global. Nos sistemas digitais binários, onde as entradas só podem tomar um de dois valores, quando o número de variáveis de entrada do sistema não é excessivo, podemos representar o comportamento do sistema sob a forma de uma tabela, onde enumeramos todas as possíveis combinações das variáveis de entrada e identificamos o valor que a saída toma para cada uma das combinações. Esta forma de representação permite a observação exaustiva do comportamento do sistema para todas as possíveis combinações das entradas. Neste caso dizemos que a função desempenhada pelo sistema está representada sob a forma de uma tabela de verdade.

1.2.1 Tabela de verdade

A tabela de verdade Tabela 1-1 representa o comportamento da malha da Figura 1-2, utilizando o digito 0 para designar variável falsa e o digito 1 para designar variável verdadeira.

A	B	C	L
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Tabela 1-1

1.2.2 Expressão Booleana

Sendo A, B e C variáveis independentes e L uma variável dependente, o comportamento da malha de comutação pode ser descrito através de uma expressão booleana $L = A.B + \overline{A} + \overline{C}$, que exprime para que condições de A, B e C a lâmpada acende, ou seja, em que L fica verdadeira (considerando L verdadeira quando a lâmpada acende). Na expressão booleana de L, o operador **OU** é representado pelo sinal (+), o operador **AND** pelo sinal (.) e o operador de complementação por uma barra sobre a variável. São estes os três operadores básicos definidos por Boole (AND, OR e NOT) e que permitem descrever o comportamento de qualquer circuito lógico.

1.3 Propriedades e teoremas dos operadores Lógicos

O conhecimento das propriedades dos vários operadores lógicos (AND, OR e NOT), permitem a simplificação algébrica de uma malha de comutação.

AND

O operador AND, também conhecido por produto lógico ou conjunção, é um operador binário, ou seja, definido entre duas variáveis. A operação AND entre as variáveis A e B tem a expressão booleana $F = A.B$ à qual corresponde a malha da Figura 1-3. Tal como acontece na escrita de expressões algébricas pode-se omitir o ponto entre os identificadores das variáveis obtendo-se: $F = AB$

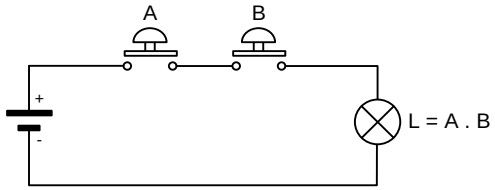


Figura 1-3

O comportamento do operador AND pode ser lido de forma positiva, dizendo-se que a função só é verdadeira quando A e B forem simultaneamente verdadeiras, ou de forma negativa dizendo que basta que uma das variáveis seja falsa para que a função seja falsa.
Da Figura 1-3 podemos deduzir as seguintes propriedades:

$A.1 = A$; 1 é elemento neutro.

$A.0 = 0$; 0 é elemento absorvente.

$A.A = A$; teorema da idempotência

$A.\overline{A} = 0$; teorema da complementação

$A.B = B.A$; propriedade comutativa

OR

O operador OR, também conhecido por disjunção, é binário, ou seja, é definido entre duas variáveis. A operação OR entre as variáveis A e B é dada pela expressão $F = A + B$ à qual corresponde a malha da Figura 1-4. O seu comportamento lógico pode ser lido de forma positiva, dizendo-se que a função é verdadeira quando A ou B forem verdadeiras. De forma negativa podemos dizer que a função só é falsa se ambas as variáveis forem falsas.

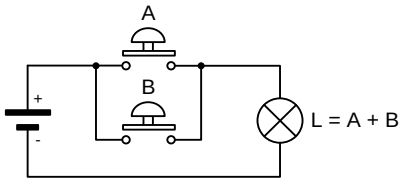


Figura 1-4

A função lógica OR definida na álgebra de Boole é inclusiva, ou seja, também é verdade quando A e B são verdadeiras.
Da Figura 1-4 podemos deduzir as seguintes propriedades

$A + 0 = A$; 0 é elemento neutro

$A + 1 = 1$; 1 é elemento absorvente

$A + A = A$; teorema da idempotência

$A + \overline{A} = 1$; teorema da complementação

$A + B = B + A$; propriedade comutativa

1.3.1 Propriedades envolvendo a operação AND e a operação OR

$A + B + C = (A + B) + C = A + (B + C)$; propriedade associativa

$A.B.C = (A.B).C = A.(B.C)$; propriedade associativa

$A(B + C) = AB + AC$; propriedade distributiva do AND em relação ao OR

$A + BC = (A + B)(A + C)$; propriedade distributiva do OR em relação ao AND

NOT

O operador NOT, também conhecido por complementação ou negação, é unário, ou seja, definido para uma variável. A operação NOT sobre a variável A tem a expressão booleana $F = \overline{A}$ à qual correspondem as malhas da Figura 1-5. O seu comportamento lógico pode ser lido como: a função F é verdade quando A é falso, ou quando A é verdade a função é falsa.

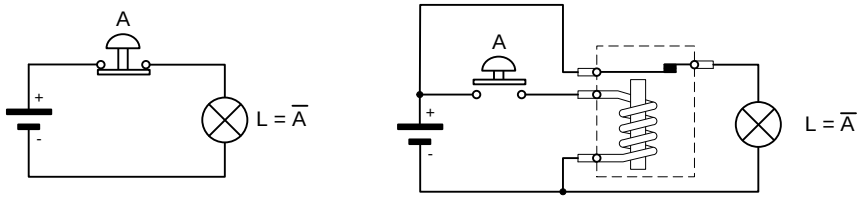


Figura 1-5

$A = \overline{\overline{A}}$; teorema da involução

Exemplo:

Admita que quer construir um circuito para acender e apagar a lâmpada de um quarto, mas que tenha também como objectivo poupar energia.
Assim sendo, a lâmpada só deve acender se existir alguém dentro do quarto e não existir iluminação vinda do exterior. Existindo Sol, a lâmpada acende se as cortinas estiverem corridas.
Identificam-se três variáveis: um sensor P de pessoas dentro do quarto, um sensor de luminosidade que indica a existência de Sol e um sensor de que o estore está corrido.
Tratando-se de apenas três variáveis, podemos representar o comportamento do sistema através de uma tabela de verdade Tabela 1-2.

P	S	E	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Tabela 1-2

Podemos então dizer que a lâmpada acende em três intercepções de P, S e E, o que permite escrever a seguinte expressão:

$$L = P\overline{S}\,\overline{E} + P\overline{S}E + PSE$$

por aplicação das propriedades anteriormente observadas, poderemos dizer que:

$L = P(\boxed{\overline{S}\,\overline{E}} + \boxed{\overline{S}E} + SE)$ distributiva do AND em relação ao OR

$L = P(\overline{S}(\overline{E} + E) + SE)$ como $(\overline{E} + E) = 1$

$L = P(\overline{S}.1 + SE)$ como $\overline{S}.1 = \overline{S}$

$L = P(\overline{S} + SE)$ que corresponde ao circuito da Figura 1-6.

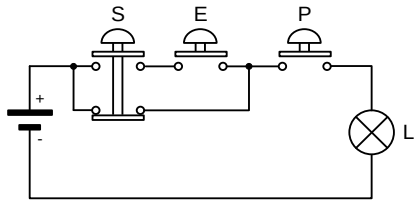


Figura 1-6

1.3.2 Teorema da absorção

Dadas as propriedades que até agora estudámos seríamos levados a considerar que a expressão encontrada determinava o circuito mais simples para L, mas assim não é. Observemos o seguinte:

$A + AB = A(1 + B) = A.1 = A$ donde se conclui que:

$\boxed{A + AB = A}$ (teorema da absorção)

Se tivermos a função $F = A + \overline{A}B$, e utilizando o teorema anterior, podemos dizer que:

$F = \boxed{A + AB} + \overline{A}B$ =>

$F = A + AB + \overline{A}B = A + B(A + \overline{A}) = A + B.1 = A + B$

Donde se conclui que:

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

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

Podemos agora então obter uma expressão mais simples para L

Sendo $L = P(\overline{S} + SE)$ => $L = P(\overline{S} + E)$
que corresponde ao circuito da Figura 1-7 e em que o actuador S, só tem uma lâmina.

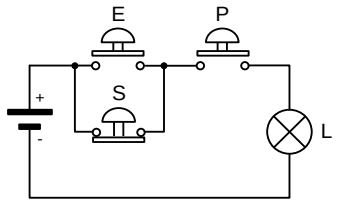


Figura 1-7

O mesmo é dizer que a lâmpada só acende se existir Pessoa e não existir Sol, ou se existir Pessoa e Sol, só acende se o Estore estiver corrido.

1.3.3 Teorema de DeMorgan

Admita que pretende realizar uma malha de comutação com dois interruptores A e B e uma lâmpada L tal que a lâmpada só apague quando A e B forem actuados simultaneamente. O mesmo é dizer que se pretende negar a função $L = A.B$ obtendo-se assim:

$\overline{L} = A.B$ => negando ambos os membros da função a igualdade não se altera

$\overline{\overline{L}} = \overline{A.B}$ => $L = \overline{A.B}$

Como se trata da negação da função, ou seja do valor de L, poderíamos realizá-la à custa de um relé como mostra a Figura 1-8, o que implica a utilização de um elemento electromecânico.

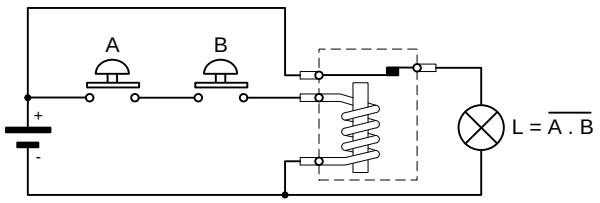


Figura 1-8

No entanto se descrevermos a função L por uma tabela de verdade podemos concluir o seguinte:

A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

Da tabela de verdade obtemos a função $L = \overline{A}\,\overline{B} + \overline{A}\,B + A\,\overline{B}$

$L = \overline{A}(\overline{B} + B) + A\overline{B} \Rightarrow L = \overline{A} + A\overline{B} \Rightarrow L = \overline{A} + \overline{B}$

ou seja

$\overline{A.B} = \overline{A} + \overline{B}$ Teorema de DeMorgan

Da mesma forma

$\overline{A + B} = \overline{A}.\overline{B}$ Teorema de DeMorgan

A estas expressões correspondem os circuitos da Figura 1-9

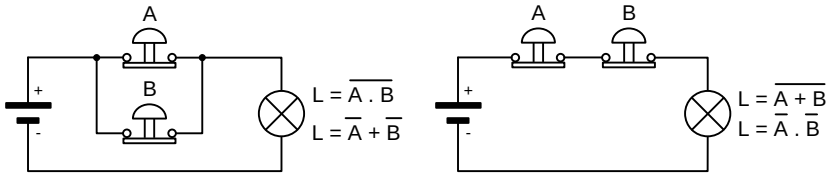


Figura 1-9

Podemos generalizar o Teorema de DeMorgan, obtendo-se as seguintes expressões:

$\overline{X1.X2\ldots Xn} = \overline{X1} + \overline{X2} + \ldots + \overline{Xn}$

$\overline{X1 + X2 + \ldots + Xn} = \overline{X1}.\overline{X2}\ldots\overline{Xn}$

1.3.4 Formas standard de representação algébrica de uma função

Existe uma variedade de formas algébricas de representar uma função booleana. No entanto existem duas formas consideradas standard para representar uma função booleana. Estas formas são a AND-OR e a OR-AND.

Consideremos a função lógica dada pela tabela de verdade Tabela 1-3.

C	B	A	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Tabela 1-3

Da tabela, e sem realizarmos nenhuma simplificação, podem extrair-se duas expressões para F:
A forma AND-OR por extracção dos 1s da função

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

A forma OR-AND obtida por extracção dos 0s da função

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

A primeira expressão é denominada por forma canónica AND-OR, também denominada por união de intercepções ou união de termos produto. Na forma canónica em cada *termo-produto* figuram todas as variáveis da função. Aos *termos-produto* onde intervêm todos as variáveis da função complementadas ou não, denominam-se *termos mínimos*. A razão do nome termo mínimo ou mintermo deve-se ao facto de este corresponder a uma única entrada na tabela de verdade.
A segunda expressão é denominada por forma canónica OR-AND, também denominada por intercepção de uniões ou intercepção de *termos-soma*. Em cada *termo-soma* figuram todas as variáveis da função. Os *termos-soma* em que intervêm todas as variáveis da função denominam-se *termos máximos*. A razão do nome termo máximo ou maxtermo, deve-se ao facto de este corresponder a uma função que tem um único zero na tabela de verdade (exemplo a função A+B+C só tem um zero na entrada $\overline{A}.\overline{B}.\overline{C}$). Poderemos obter uma expressão na forma OR-AND a partir da forma AND-OR utilizando o teorema de De Morgan da seguinte forma:

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

Aplicando a distribuição do AND em relação ao OR, obtemos:

$$Z = \overline{(A+\overline{B}+\overline{C}).(\overline{A}+\overline{A}C+\overline{A}B+BC)} = \overline{(A+\overline{B}+\overline{C}).(\overline{A}.(1+C+B)+BC)}$$
$$Z = \overline{(A+\overline{B}+\overline{C}).(\overline{A}+BC)}$$
$$Z = \overline{ABC + \overline{A}\,\overline{B} + \overline{A}\,\overline{C}} = (\overline{ABC}).(\overline{\overline{A}\,\overline{B}}).(\overline{\overline{A}\,\overline{C}}) = (\overline{A}+\overline{B}+\overline{C}).(A+B).(A+C)$$

1.3.5 Manipulação algébrica para simplificação de funções

Imaginemos que se pretendia implementar o circuito com o comportamento dado pela expressão:

$$F = \overline{A}BC + \overline{A}B\overline{C} + AC = \overline{A}B\left(C+\overline{C}\right) + AC = \overline{A}B + AC$$

A implementação desta função implicava três ramos paralelos tendo o actuador A e C três lâminas e o actuador B duas. Manipulando a expressão por aplicação das várias propriedades já estudadas podemos chegar a uma expressão mais simples, que produz uma malha com menos ramos e menos lâminas.

$$F = \overline{A}B + AC$$

Exemplos de simplificação algébrica de funções:

$$F1 = \overline{B} + AB\overline{C} = \overline{B} + A\overline{C}$$
$$F2 = (A+AB).(C+\overline{C}D) + \overline{A}D = A(C+D) + \overline{A}D = AC + AD + \overline{A}D = AC + D$$
$$F3 = (A+B+C).(A+\overline{B}+C) = \boxed{A + A\overline{B} + AC + AB} + \boxed{BC + AC + \overline{B}C + C} \Rightarrow$$
$$F3 = A\underbrace{(1+B+C)}_{1} + C\underbrace{(B+A+1)}_{1} = A + C$$

$F4 = \overline{A}B + AC + BC$

$F5 = \overline{A}\,\overline{C} + \overline{A}D + \overline{A}B + A\overline{D} + \overline{B}\,\overline{D}$

$F6 = \overline{A}D + CD + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$

Na função F4, utilizando as propriedades e teoremas até agora estudados (distributiva e eliminação de termos complementares) poderíamos ser levados a pensar que a função não é simplificável, no entanto podemos verificar pela demonstração que se segue que o termo BC é redundante, podendo ser eliminado.

$F4 = \overline{A}B + AC + BC = \overline{A}B + AC + BC(A + \overline{A})$

$F4 = \overline{A}B + AC + ABC + \overline{A}BC$

$F4 = \overline{A}(B + BC) + AC(1 + B) \quad ; (B + BC) = B\ e\ (1 + B) = 1$

$F4 = \overline{A}B + AC$

$\overline{A}B + AC + BC = \overline{A}B + AC$ (Teorema do consenso)

Os exemplos que se seguem, demonstram como se pode tornar difícil a simplificação de uma função pelo método algébrico, sendo necessário para tal alguma perspicácia.

$F5 = \overline{A}\,\overline{C} + \overline{A}D + \overline{A}B + A\overline{D} + \overline{B}\,\overline{D}$

A aplicação das várias propriedades estudadas, no sentido de obter termos simplificáveis, não surte qualquer efeito. Este facto poderia levar-nos a pensar que a função não é simplificável, no entanto a função é simplificável como demonstraremos em seguida.

O método que iremos utilizar para simplificar a função, consiste em injectar um termo redundante que produza efeito *dominó*. Pelo teorema do consenso podemos injectar mais uma intercepção que possa produzir simplificação com outros elementos. Por exemplo, na união dos termos $\overline{A}D + \overline{B}\,\overline{D}$ podemos injectar o termo redundante $\overline{A}\,\overline{B}$, porque pelo teorema do consenso $\overline{A}D + \overline{B}\,\overline{D} + \overline{A}\,\overline{B} = \overline{A}D + \overline{B}\,\overline{D}$. Sendo assim obteremos para F5 a seguinte expressão:

$F5 = \boxed{\overline{A}\,\overline{C}} + \boxed{\overline{A}D} + \boxed{\overline{A}B} + A\overline{D} + \overline{B}\,\overline{D} + \boxed{\overline{A}\,\overline{B}}$

$F5 = \overline{A}(\underbrace{\overline{C} + D + \boxed{B + \overline{B}}}_{1}) + A\overline{D} + \overline{B}\,\overline{D} = \boxed{\overline{A} + A\overline{D}} + \overline{B}\,\overline{D} = \overline{A} + \overline{D} + \overline{B}\,\overline{D} = \overline{A} + \overline{D}(1 + \overline{B}) =>$

$F5 = \overline{A} + \overline{D}$

Outra forma seria associar ao termo $\overline{B}\,\overline{D}$ o termo $(A + \overline{A})$ produzindo,

$\overline{B}\,\overline{D}(A + \overline{A}) = \overline{B}\,\overline{D}\,A + \overline{B}\,\overline{D}\,\overline{A}$ obtendo-se para F5 a seguinte expressão:

$F5 = \overline{A}(\underbrace{\overline{C} + D + B + \overline{B}\,\overline{D}}_{1}) + A\overline{D} + \overline{B}\,\overline{D}A = \overline{A} + A\overline{D} + \overline{B}\,\overline{D}A = \overline{A} + \overline{D}$

$$F6 = \overline{A}D + CD + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

Pelo teorema da complementação $X + \overline{X} = 1$, podemos produzir

$$F6 = \overline{A}D + CD(\overline{A} + A) + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \underbrace{\overline{A}D + \overline{A}CD}_{\overline{A}D} + ACD + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \overline{A}D + ACD + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \overline{A}D + ACD(B + \overline{B}) + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \overline{A}D + \underbrace{ABCD + BD}_{BD} + \underbrace{A\overline{B}CD + A\overline{B}C}_{A\overline{B}C} + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \overline{A}D + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

Outra forma seria utilizando o elemento nulo da intercepção $X.1 = \mathrm{X}$

$$F6 = \overline{A}D + CD(\underbrace{\overline{A}\,\overline{B} + \overline{A}B + A\overline{B} + AB}_{1}) + BD + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$F6 = \underbrace{\overline{A}D + \overline{A}\,\overline{B}CD + \overline{A}BCD}_{\overline{A}D} + \underbrace{CDAB + BD}_{BD} + \underbrace{A\overline{B}CD + A\overline{B}C}_{A\overline{B}C} + \overline{A}\,\overline{B}\,\overline{C}$$

Em alternativa ao termo CD podemos eliminar o termo $\overline{A}D$, que fica como desafio para o aluno.

Como podemos constatar no exercício anterior, a simplificação de um termo produto pode implicar um número elevado de operações (inserção de termos) as quais têm que ser "cirurgicamente" escolhidas. Por outro lado, não existia nada à partida que nos levasse a pensar que o termo CD ou em alternativa o termo $\overline{A}D$ fossem redundantes.

Para colmatar esta dificuldade na simplificação de funções booleanas, desenvolveram-se várias técnicas tabulares, sendo o mapa de *Karnaugh* e *Quine McCluskey* as mais conhecidas. O método *Quine McCluskey* garante maior simplificação pois recorre a exploração exaustiva de várias combinações, sendo por isso mais adequada à simplificação algébrica apoiada em computador.

1.4 Simplificação Tabular de funções

Representemos a função $F4 = \overline{A}B + AC + BC$ num diagrama de Venn como mostra Figura 1-10.

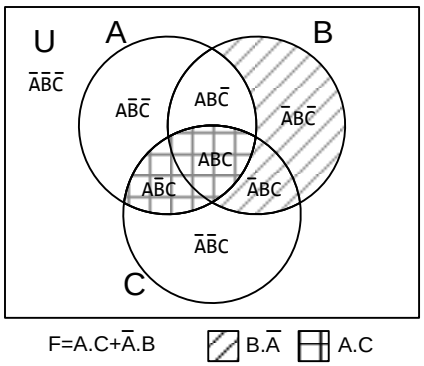


Figura 1-10

Facilmente se verifica que a função pode ser dada pela união das áreas (intercepção) $B.\overline{A}$ com $A.C$, estando a área $B.C$ já incluída nesta união. A representação no diagrama de *Venn* permite identificar com grande facilidade quais as áreas que unidas nos permitem representar todo espaço onde a função é verdadeira. Estes diagramas têm o inconveniente de serem constituídos pela intercepção de formas circulares e logo difíceis de desenhar, principalmente com o aumento do número de variáveis.

Podemos representar uma função de forma tabular com as mesmas propriedades que o diagrama de *Venn*. Esta forma de representar, é referida como mapas de *Karnaugh* ou *Kmaps*, em homenagem ao autor, e não são mais que uma forma rectilínea de representar o diagrama de *Venn*.

1.4.1 Mapas de Karnaugh

No exemplo da Figura 1-11 é mostrado um mapa de *Karnaugh* equivalente ao diagrama de *Venn* anteriormente representado. O mapa de *Karnaugh*, tal como o diagrama de *Venn*, é uma forma de representação de uma função booleana, em tudo equivalente a uma tabela de verdade, com a grande vantagem de os termos-mínimos simplificáveis estarem em quadrículas adjacentes, ou seja, cada termo mínimo tem como vizinho todos os termos mínimos que diferem dele de uma variável complementada. Observemos por exemplo o termo $A\,\overline{B}\,\overline{C}$ que tem como termos adjacentes $\overline{A}\,\overline{B}\,\overline{C}, A\,\overline{B}\,C, A.B.\overline{C}$ ou o termo $\overline{A}\,\overline{B}\,\overline{C}$ que tem adjacente os termos $\overline{A}\,B\,\overline{C}, \overline{A}\,\overline{B}\,C, A.\overline{B}.\overline{C}$, podendo dizer-se que o mapa está inscrito num tubo como é mostrado na Figura 1-11.

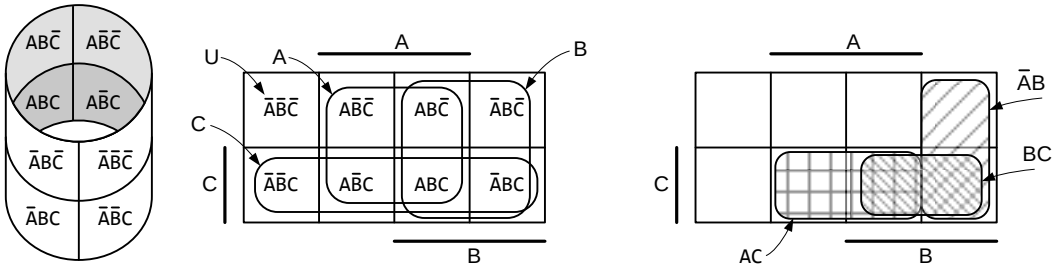


Figura 1-11

Na Figura 1-12 estão representadas exemplos de mapas de *Karnaugh* com diferente número de variáveis. Mapas de *Karnaugh* com mais que quatro variáveis não são tão simples de utilizar, razão pela qual serem pouco usuais.

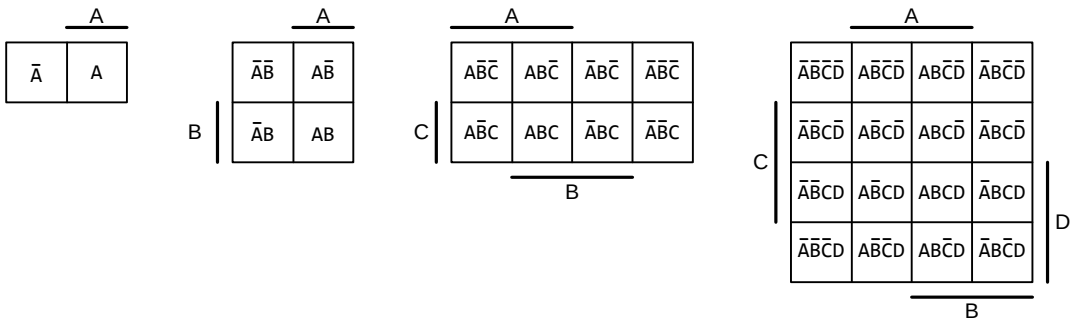


Figura 1-12

1.4.2 Representação de uma função no mapa de *Karnaugh*

A representação de uma função num mapa de *Karnaugh*, faz-se colocando o valor 1 nas quadrículas correspondentes às intercepções verdadeiras que constituem a união.

Como se trata de representar a união das intercepções, e sabendo a propriedade do **OR** que 1+1=1, sempre que ao inserir uma nova intercepção a quadrícula correspondente já se encontre a 1, manteremos apenas esse 1 pela propriedade anteriormente referida. Quando um termo produto, não contém todas as variáveis, é óbvio que a sua inserção no mapa implica vários 1s.

Tomemos como exemplo a Figura 1-13, onde estão representação em mapa de *Karnaugh* as seguintes funções:

$$F1(A,B) = B + \overline{A}\,\overline{B}$$
$$F2(A,B,C) = \overline{A}B + \overline{A}\,\overline{B}C + A\,\overline{B}\,\overline{C}$$
$$F3(A,B,C) = AB + BC + AC + \overline{A}B\overline{C}$$
$$F4(A,B,C,D) = A\overline{B} + \overline{A}\,\overline{B}\,\overline{D} + \overline{B}\,D + B\overline{C}D + \overline{A}BC\overline{D} + AB\overline{C}$$

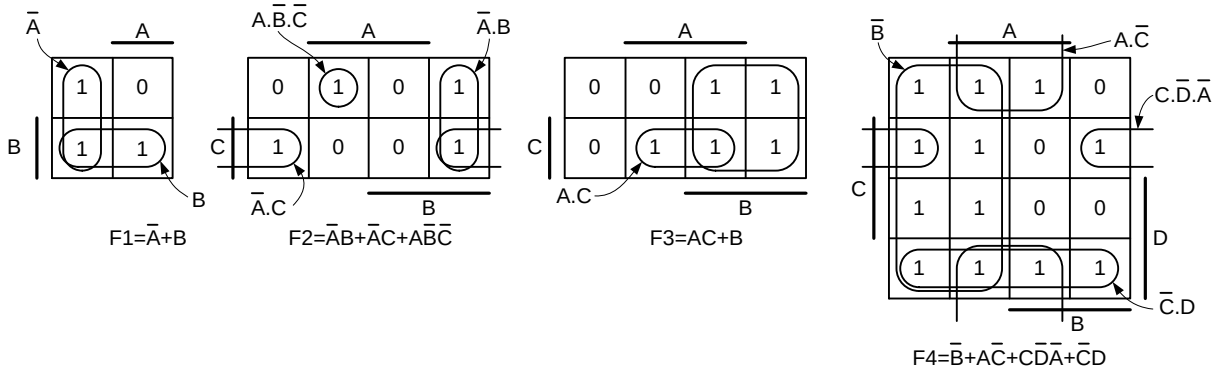


Figura 1-13

1.4.3 Simplificação de funções no mapa de *Karnaugh*

Para obter a função simplificada no mapa de *Karnaugh*, agrupam-se todos os 1s que tenham a mesma característica e na maior quantidade possível. Embora se possa agrupar qualquer número de 1s, para que do agrupamento resulte uma intercepção é necessário que o agrupamento seja constituído por 1s adjacentes, numa quantidade potência inteira de 2. A expressão simplificada da função, será obtida pela união dos grupos (intercepções) efectuados, e que representa toda a "área" onde a função é verdadeira. Para constituir um novo grupo de 1s podem utilizar-se 1s que já tenham servido para constituir outros grupos. Como se pode ver na Figura 1-13, as expressões assim obtidas estão na forma AND-OR.

O agrupamento de 0s permite obter, uma expressão na forma OR-AND como mostra a Figura 1-14.

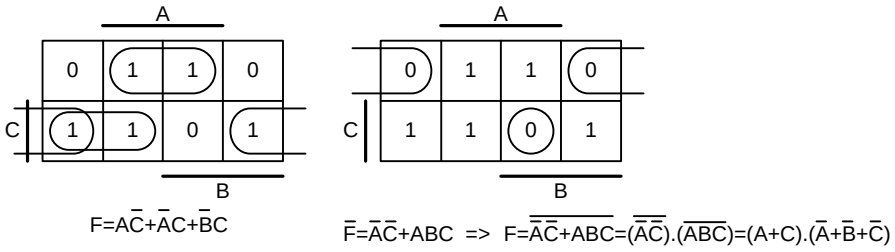


Figura 1-14

Algumas funções poderão ter várias expressões e todas elas igualmente simples como mostra a Figura 1-15.

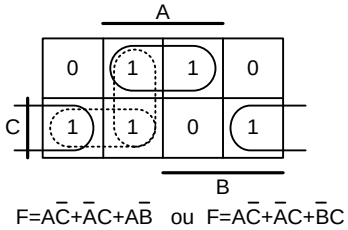


Figura 1-15

Para mostrar a grande importância dos mapas *Karnaugh*, simplifiquemos agora a função F5 e F6 do exercício anterior, que se verificou ser de difícil simplificação algébrica.

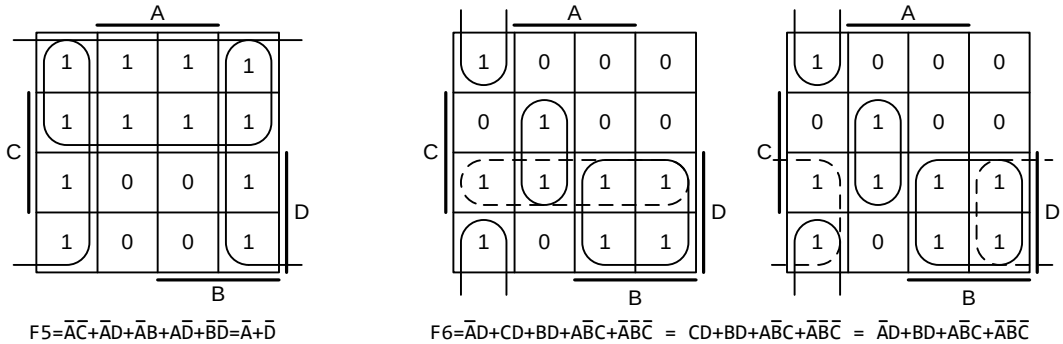


Figura 1-16

Como se pode observar na Figura 1-16, a expressão resultante é igual à obtida por simplificação algébrica e sem necessidade de tanta perspicácia. A redundância do termo CD ou do termo $\overline{A}D$ é agora evidente.

1.4.4 Simplificação de funções com mais de quatro variáveis no mapa de *Karnaugh*

A necessidade de simplificar funções com mais de quatro variáveis é pouco usual. Assim caso o sistema que estejamos a projectar tenha mais do que quatro entradas, é aconselhável a divisão do sistema em blocos funcionais, tal que cada um deles apresente uma complexidade facilmente abarcável pelo humano. Caso surja a necessidade de simplificar uma função com mais de quatro variáveis podemos utilizar como método a divisão em mapas de quatro variáveis.
Utilizemos este método na simplificação de uma função de quatro variáveis:

$$F(A,B,C,D) = \overline{A}.\overline{B}.\overline{D} + \overline{B}.D + B.C.\overline{D} + CD + A.\overline{C}.D + \overline{A}.B.\overline{C}.\overline{D}$$

Realizando a simplificação por divisão em dois mapas de *Karnaugh* de três variáveis, como mostra a Figura 1-17, obtemos a mesma expressão que obteríamos por simplificação realizada com um mapa de quatro variáveis.
Quando agrupamos o mesmo conjunto nos dois mapas a variável **D** cai, quando o agrupamento só surge num dos mapas, então o mintermo inclui o valor da variável **D** nesse mapa.

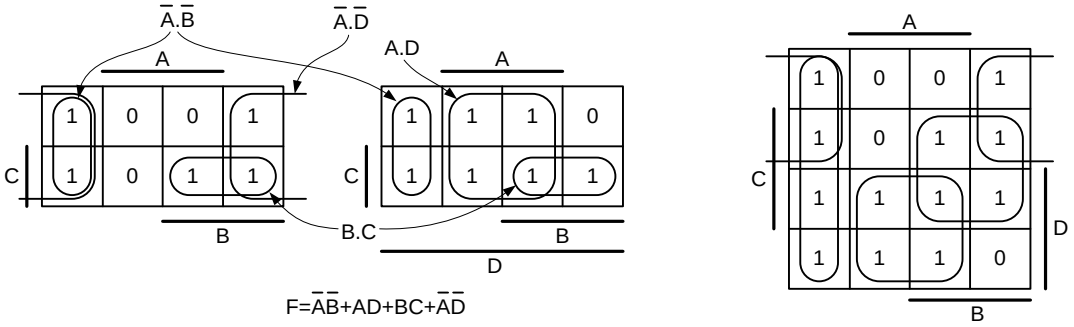


Figura 1-17

1.4.5 Simplificação de funções na forma OR-AND utilizando *Karnaugh*

Caso uma função se apresente na forma OR-AND, a sua transcrição para o mapa de *Karnaugh* não é imediata, sendo necessária a passagem à forma AND-OR por aplicação da distribuição entre os vários termos. Mas como a forma OR-AND é constituída por expressões na forma AND-OR, poderemos não realizar a distribuição e em vez disso realizar o AND entre os mapas de *Karnaugh* contendo a expressão dos termos AND-OR, como mostra o exemplo da Figura 1-18.

$$F(A,B,C) = \left(\overline{A}.\overline{B}.C + A.\overline{C} + \overline{A}.B\right).\left(A.\overline{C} + A.B + B.\overline{C}\right)$$

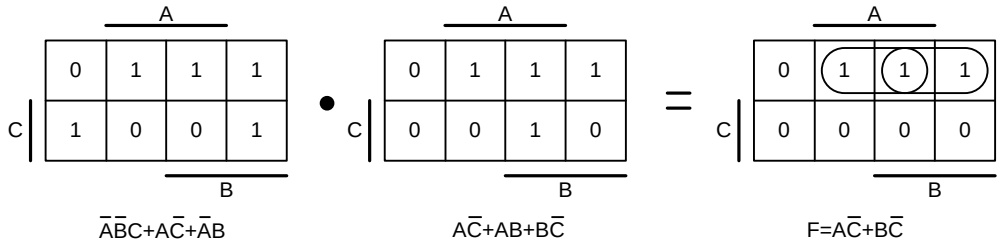


Figura 1-18

1.4.6 Exercício:

Pretende-se construir o sistema para controlo de uma lâmpada de iluminação de uma sala. A sala tem três portas, estando associado a cada uma das portas um actuador biestável (que após ser manipulado permanece estável na ultima condição, ou seja, activado ou desactivado). O sistema de controlo deve ter o seguinte comportamento: se todos os actuadores (A, B e C) estiverem desactivados (valor lógico zero) a lâmpada deve estar apagada; Independentemente do estado em que se encontre cada um dos actuadores, se for alterado o estado de um qualquer actuador, a lâmpada deve acender caso esteja apagada, ou apagar caso esteja acesa. Este comportamento permite entrar ou sair por qualquer porta, e ter sempre possibilidade de no actuador que lhe está associada, apagar ou acender a lâmpada.

Analisemos o comportamento do sistema numa tabela de verdade (Tabela 1-4), partindo da situação de A,B e C a zero que corresponde a lâmpada apagada.

-

C B A	F
0 0 0	0
0 0 1	1
0 1 0	1
1 0 0	1

C B A	F
0 0 1	1
0 1 1	0
1 0 1	0

C B A	F
0 1 0	1
0 1 1	0
1 1 0	0

C B A	F
1 0 0	1
1 1 0	0
1 0 1	0

C B A	F
1 0 1	0
1 1 0	0
0 1 1	0
1 1 1	1

C B A	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

Tabela 1-4

Se utilizarmos o mapa de Karnaugh para obter uma expressão simplificada da função como mostra a Figura 1-19, concluímos que esta não é simplificável, e que implicaria a união de quatro termos mínimos. A esta função que denominaremos por *exclusive* OR (XOR), estabelece um novo operador de grande importância no desenho de sistemas digitais.

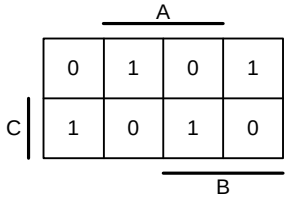


Figura 1-19

1.4.7 XOR (*exclusive* OR)

A função XOR, é definida da seguinte forma: é verdadeira quando existe um número impar de variáveis com o valor lógico 1. Para a configuração 000, a quantidade de varáveis com o valor lógico 1 é igual a zero. Porque zero, por definição é um número par, daí que a função valha 0 quando todas as variáveis têm o valor lógico zero. A função XOR entre duas variáveis tem a tabela de verdade expressa na Figura 1-20.

A B	$F = \overline{A}B+A\overline{B} = A \oplus B$
0 0	0
0 1	1
1 0	1
1 1	0

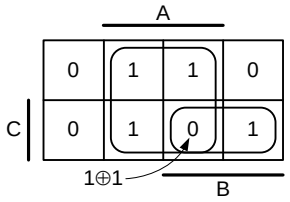
Figura 1-20

Propriedades do operador XOR:

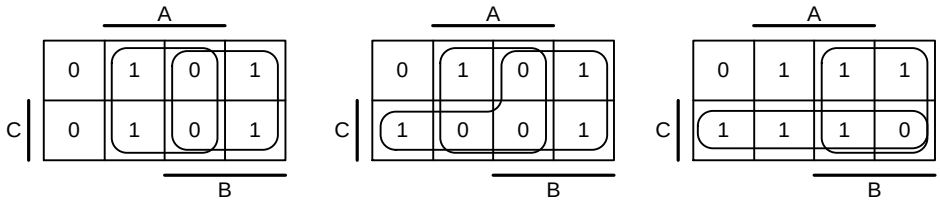
$A \oplus 0 = A$
$A \oplus 1 = \overline{A}$
$A \oplus A = 0$
$A \oplus \overline{A} = 1$
$A \oplus B = B \oplus A$
$A \oplus B \oplus C = (A \oplus B) \oplus C = A \oplus (B \oplus C)$
$A.B \oplus A.C = A(B \oplus C)$
$A \oplus \overline{B} = \overline{A} \oplus B = \overline{A \oplus B} = \overline{A}.\overline{B} + A.B$
Se $A \oplus B \oplus C = 0$ então $A \oplus B = C$, $A \oplus C = B$ e $B \oplus C = A$

As propriedades $1 \oplus 1 = 0$ e $1 \oplus 0 = 1$ têm na simplificação de funções através de mapas de Karnaugh algumas consequências que poderão ser exploradas no sentido de diminuir o número de operadores. Tomemos como exemplo a seguinte função:

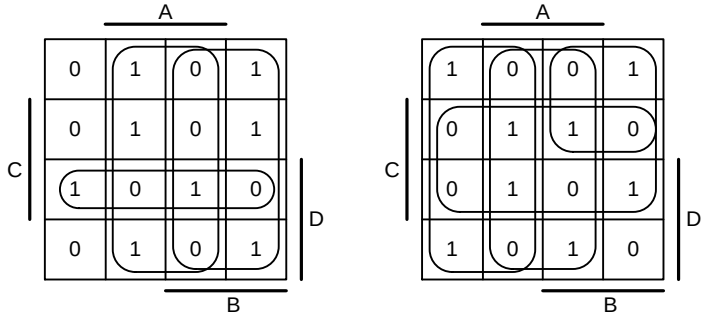
$$F = A \oplus (BC)$$



Esta função expressa na forma AND-OR seria $F = A\overline{B} + A\overline{C} + \overline{A}BC$ e na forma OR-AND seria $F = (A + B).(\overline{A} + \overline{B} + \overline{C}).(A + \overline{B} + C)$. Qualquer uma das formas implica, um número mais elevado de operadores, razão pela qual este método é utilizado quando se pretende diminuir o número de operadores quando a função em causa assim o permite. Como veremos adiante a utilização deste operador pode simplificar a implementação de sistemas lógicos. Na Figura 1-21 estão alguns exemplos da aplicação da função XOR na simplificação de funções.
Exemplos:



$F = A \oplus B$ $\quad F = A \oplus (B + C)$ $\quad F = A + (B \oplus C)$



$F = A \oplus B \oplus CD$ $\quad F = A \oplus \overline{B} \oplus C \oplus B\overline{D}$

Figura 1-21

1.5 Exercícios do capítulo 1

$E1(A,B,C,D) = \overline{A}.\overline{B}.\overline{C} + \overline{B}.C.\overline{D} + A.\overline{B}.\overline{C} + \overline{A}.B.C.\overline{D}$

$E2(A,B,C,D) = A.\overline{C}.\overline{D} + \overline{B}.C.(A+D) + \overline{\overline{A}+C} + B.C.D$

$E3(A,B,C,D) = \left(A+\overline{B}+C\right).\left(\overline{A}+\overline{B}+C\right).\left(B+C+\overline{D}\right)$

$E4(A,B,C) = \left(A.\overline{B}+\overline{A}.B+A.\overline{C}\right).\left(\overline{A}.\overline{B}+A.\overline{B}+A.\overline{C}\right)$

$E5(A,B,C) = \overline{A} + B + \overline{\overline{B} \oplus AC}.\overline{A} + A.\overline{B}.C + \overline{A}.\overline{C}$

$E6(A,B,C) = \overline{B.\overline{A.B} + \overline{A + B.\overline{C}}}$

E7 - Utilizando a transformação algébrica comprove a igualdade

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

Soluções:

$E1(A,B,C,D) = \overline{B}.\overline{D} + \overline{B}.\overline{C} + \overline{A}.C.\overline{D}$

$E2(A,B,C,D) = A.\overline{B} + C.D + \overline{C}.A$

$E3(A,B,C,D) = C + \overline{B}.\overline{D}$

$E4(A,B,C) = A.\overline{C} + A.\overline{B}$

$E5(A,B,C) = A + B + A.C + \overline{A}.\overline{C}$

$E6(A,B,C) = A$

E7

$A.B + \boxed{B.\overline{C}.\overline{D}} + \overline{A}.B.C + \boxed{\overline{C}.D}$ associando os dois termos sublinhados, pelo teorema do consenso, podemos escrever:

$A.B + \boxed{B.\overline{C}.\overline{D}} + \overline{A}.B.C + \boxed{\overline{C}.D} + \boxed{\overline{C}.B}$ que colocando B em evidência (distributiva) permite obter

$B.\underbrace{\left(A + \overline{C}.\overline{D} + \overline{A}.C + \overline{C}\right)}_{1} + \overline{C}.D = B.1 + \overline{C}.D = B + \overline{C}.D$