2. PORTAS LÓGICAS

Com o aparecimento dos semicondutores, os sistemas digitais tornaram-se o grande veículo para o desenvolvimento que hoje assistimos em quase todas as áreas tecnológicas, tendo até a quase totalidade dos sistemas electrónicos analógicos sido convertidos em sistemas digitais. Dada a complexidade cada vez maior dos sistemas digitais, o desenvolvimento destes, exigiu a criação de níveis de abordagem que não fosse a dos elementos interruptores. O primeiro nível de abordagem estruturante a introduzir é o da porta lógica (*gate*), elemento com uma ou mais entradas que produz um sinal de saída, função dos valores presentes na entrada como mostra a Figura 2-1. Como veremos adiante, estão disponíveis no mercado componentes electrónicos denominados por *chips*, contendo várias portas lógicas.

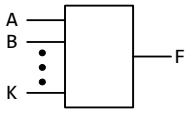


Figura 2-1

Uma característica importante das portas lógicas é que o sinal de saída e o sinal de entrada têm a mesma característica eléctrica, ou seja, permitem interligar entradas e saídas de várias portas lógicas e assim constituir-se um circuito lógico digital capaz de implementar uma qualquer função lógica.

O símbolo esquemático das portas lógicas pode ser desenhado utilizando diferentes nomenclaturas. Embora exista uma nomenclatura estabelecida pela IEEE, a que é mais vulgar e que iremos adoptar, foi estabelecida pelos fabricantes de componentes digitais, no início da era digital. Embora exista uma enorme variedade de portas lógicas disponíveis no mercado, as mais comuns são o AND, OR, NAND, NOR, XOR, NOT e IDENT(buffer), e que têm o símbolo esquemático mostrado na Figura 2-2.

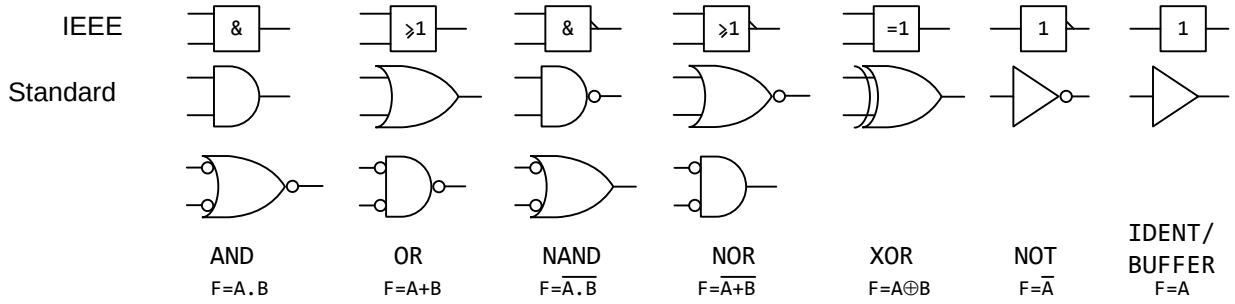


Figura 2-2

A Figura 2-3 mostra o diagrama de um circuito lógico, vulgarmente referido como desenho esquemático. Como se pode observar, a representação de uma função na forma AND-OR ou na forma OR-AND, pode alterar o número de portas lógicas necessárias à implementação da função $F = X\overline{Y} + XZ = X(\overline{Y} + Z)$.

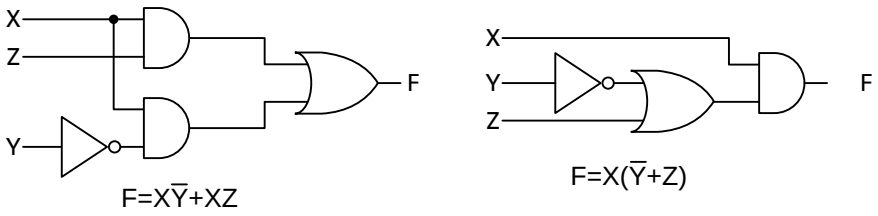


Figura 2-3

Notas sobre desenho esquemático:
Em desenho esquemático as portas lógicas deverão ser preferencialmente orientadas da esquerda para a direita. Quando não for conveniente poderão ser orientadas para baixo. Quanto às entradas estão sempre à esquerda ou na parte superior do componente e as saídas sempre do lado direito ou na parte inferior do componente. As linhas são sempre horizontais ou verticais e nunca estabelecem uma ligação quando se cruzam. As ligações são sempre realizadas na intercepção de duas linhas como mostra a Figura 2-4 acompanhadas de ponto para evidenciar a ligação. Estas regras deverão ser sempre observadas, pois aumentam a clareza da leitura, evitando-se assim confusão entre entrada e saída e entre ligação e cruzamento.

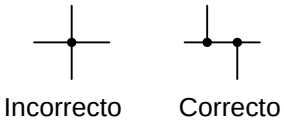


Figura 2-4

Exemplo:
Consideremos que se pretende realizar um circuito denominado descodificador (circuito de n entradas que produz k saídas com k>n) de 7 segmentos como mostra a Figura 2-5, utilizando portas lógicas:

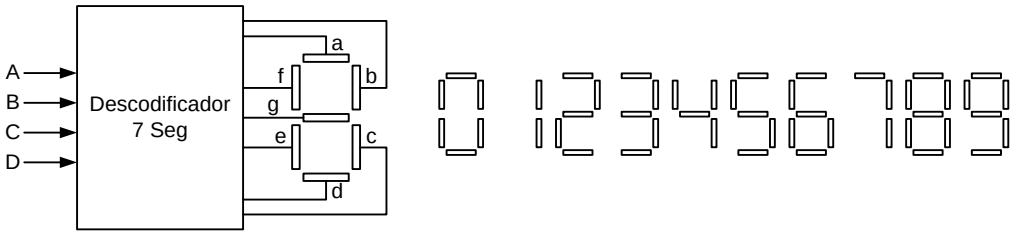


Figura 2-5

O circuito descodificador tem quatro variáveis de entrada e produz sete variáveis de saídas com o comportamento que a seguir se descreve. A cada uma das saídas do descodificador está associada um segmento de um mostrador no qual é possível desenhar todos algarismos entre 0 e 9. Admitamos que as variáveis de entrada são geradas por um sistema electromecânico, que só gera 10 combinações, ou seja, o descodificador não é sujeito às 16 (2^4) possíveis combinações das quatro variáveis. Esta especificação vai trazer um novo conceito na simplificação das funções de saída, pois leva a que o projectista não tenha que contemplar no projecto, qual a reacção do descodificador

quando sujeito às restantes combinações. Estas combinações são designadas por *don't care* (não importa) e são representadas no mapa de Karnaugh com o símbolo (**–**) ou(**x**). Como o termo mínimo assim assinalado corresponde a um valor lógico por definir, então ele poderá ser associado aos 1's no sentido de criar um termo mais simples, caso contrário será tomado como zero lógico.
Consideremos a correspondência entre as combinações de entrada e de saída apresentadas na Tabela 2-1.
Como foi anteriormente referido, admite-se que as combinações de entrada são geradas por um sistema electromecânico que se pressupõe transitar entre algarismos na sequência decimal. Para evitar que o sistema gerador de configurações ao passar de um dígito para outro na zona de transição possa gerar uma configuração errónea, a codificação proposta assegura que entre combinações adjacentes se altere um único bit. Esta codificação é denominada por codificação *Gray*.

	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	1	1	1	0	1	1	0	1
3	0	0	1	0	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1	0	0	1	1
5	0	1	1	1	1	0	1	1	0	1	1
6	0	1	0	1	1	0	1	1	1	1	1
7	0	1	0	0	1	1	1	0	0	0	0
8	1	1	0	0	1	1	1	1	1	1	1
9	1	0	0	0	1	1	1	1	0	1	1

Tabela 2-1

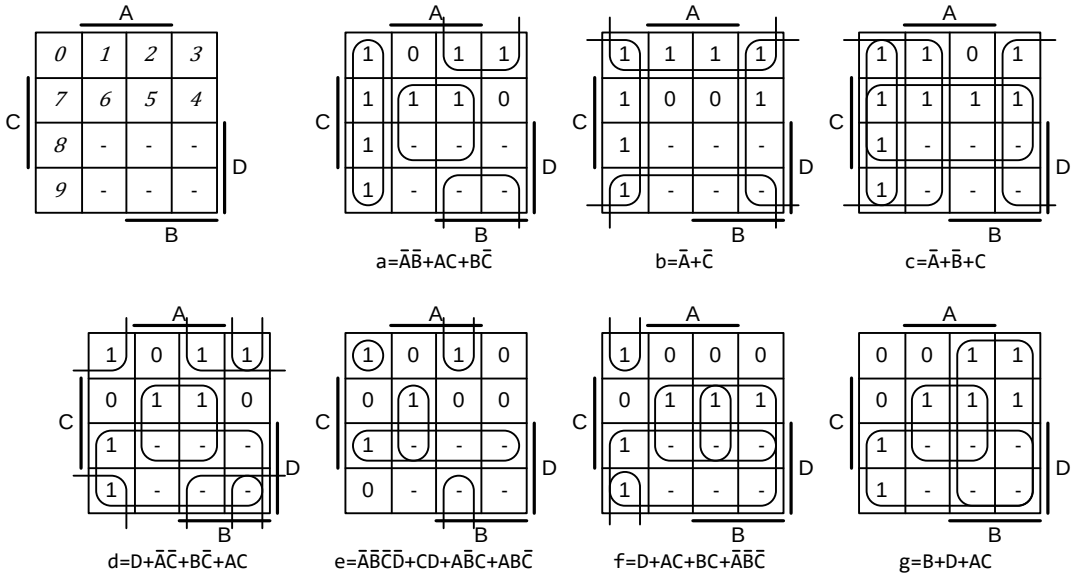


Figura 2-6

Num sistema com várias saídas dependentes das mesmas variáveis de entrada, poderemos utilizar como forma de minimizar o número de portas, um método denominado por implementação em multi-nível ou *bridging*, que consiste em determinar um qualquer padrão comum a várias funções e

utilizá-lo na implementação de cada uma das funções de saída do sistema. No presente exercício, podemos ver nos mapas de Karnaugh da Figura 2-6, que o termo $(D + AC)$, é comum a quase todos os segmentos, pelo que podemos gerar uma única vez este termo, designá-lo por X e que entrará em união em todos aqueles segmentos que tornem a expressão mais simples. A aplicação deste método levaria à obtenção das seguintes expressões:

$$a = X + \overline{A}\,\overline{B} + B\overline{C}$$
$$d = X + \overline{A}\,\overline{C} + B\overline{C}$$
$$f = X + BC + \overline{A}\,\overline{B}\,\overline{C}$$
$$g = X + B$$

A implementação do sistema descodificador de7 segmentos utilizando portas lógicas traduzir-se-ia na implementação da Figura 2-7.

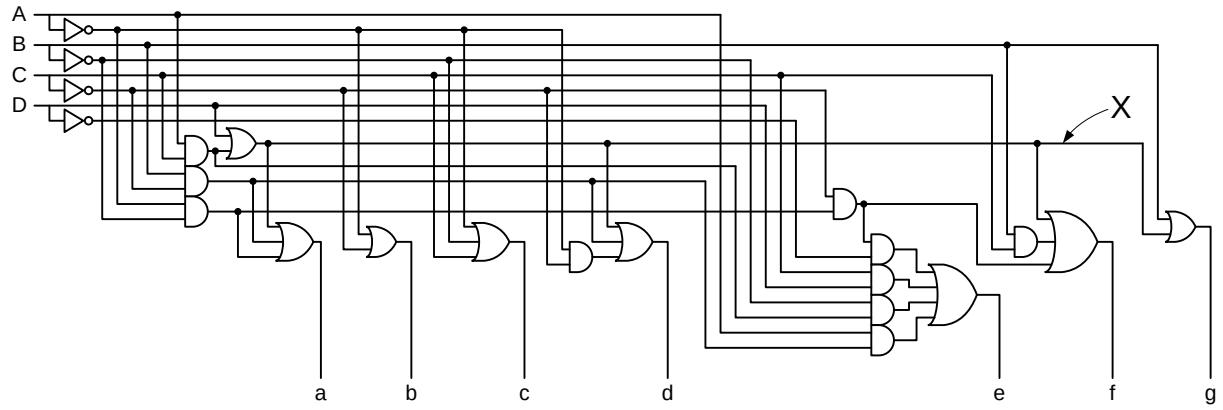


Figura 2-7

Como exercício podemos determinar, a partir da simplificação feita nos mapas de *Karnaugh* quais as configurações que seriam exibidas no mostrador de 7 segmentos se o sistema fosse sujeito às restantes 6 configurações. Complete a Tabela 2-2.

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

Tabela 2-2

2.1 Porta Lógica electrónica

Vejamos então como são constituídas as portas lógicas digitais electrónicas disponíveis no mercado. Dado que existem várias tecnologias de fabrico de dispositivos electrónicos de comutação, e que com cada uma delas, podemos conceber diferentes arquitecturas, é natural que existam disponíveis no mercado várias famílias de componentes lógicas, organizados e empacotados de diferentes formas, sendo a mais popular a família 7400. A arquitectura interna desta família foi sofrendo algumas alterações ao longo do tempo com o desenvolvimento tecnológico dos semicondutores. Começou por utilizar tecnologia bipolar standard, depois passou a utilizar tecnologia bipolar *schottky* e por último tecnologia CMOS. Quanto à tensão de funcionamento, a família 7400 necessita de 5V ou 3,3V, função da tecnologia de implementação. Como veremos mais adiante, também poderemos dispor destes elementos de forma organizada em estruturas complexas programáveis. Os vários circuitos que iremos estudar nesta primeira fase pertencem à família 7400. Esta família faz corresponder o valor lógico 1 ao valor de tensão +5V e o valor lógico 0 ao valor de tensão 0V.

2.1.1 Díodo como dispositivo binário.

O elemento mais simples de comutação electrónica é o díodo. O díodo é composto por elementos semicondutores polarizados, um positivamente e outro negativamente, de tal forma que quando juntos ficam separados por um pequeno espaço inter-molecular que não permite a total anulação de um dos pólos a favor do outro. Esta composição, confere-lhe como característica comportamental, só se deixarem percorrer por corrente eléctrica num único sentido. Esta composição produz um efeito semelhante ao de um monte de pedras que foi produzido por termos despejado sobre uma superfície plana uma caixa contendo pedras. No momento em que despejamos a caixa, estas rolam uma sobre as outras até que a energia potencial de cada pedra não seja suficiente para ultrapassar o atrito de rolamento entre elas (o que não aconteceria se fosse água). Para que pedras continuem a rolar, basta ir colocando pedras no topo do monte até que o potencial de atrito ao rolamento seja ultrapassado. Simultaneamente é necessário retirar pedras na base do monte para que o processo tenha continuidade. É óbvio, que se colocarmos pedras na base do monte estas não rolam para o topo. Também no díodo, para que tenhamos corrente através deste, é necessário ultrapassar a tensão de contenção do espaço inter-molecular colocando uma tensão aos terminais do díodo, tal que, as cargas negativas (corrente electrões) continuem a atravessar o espaço inter-molecular no sentido do pólo negativo para o pólo positivo.

O díodo tem o símbolo esquemático apresentado na Figura 2-8 a) correspondendo o ânodo ao pólo positivo e cátodo ao pólo negativo. O sentido da corrente eléctrica (convencional) é do ânodo para o cátodo. Embora o díodo tenha uma característica tensão corrente não linear, pode ser linearizada quando utilizado em circuitos lógicos obtendo-se o gráfico da Figura 2-8 b) e que traduz o seguinte comportamento: quando a tensão entre o ânodo e o cátodo atinge a tensão limiar de condução (V_{LC}) o díodo apresenta resistência nula à passagem de corrente (circuito fechado), caso contrário apresenta resistência infinita (circuito aberto).

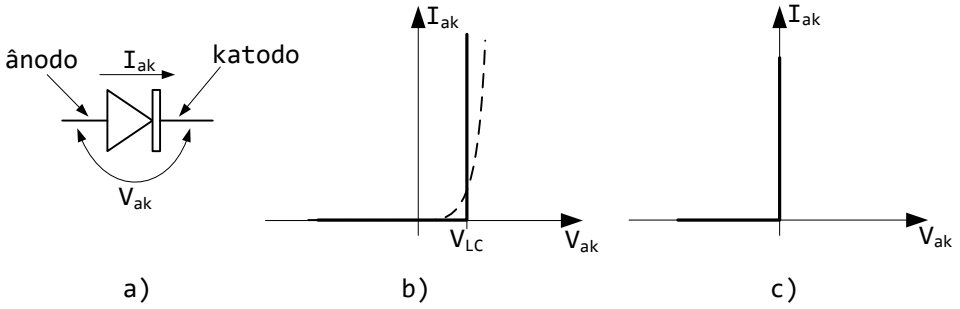


Figura 2-8

Para efeitos de análise de uma malha de comutação, o comportamento dos díodos pode definir-se da seguinte maneira:

- Um díodo só permite ser atravessado por corrente, se a tensão entre o ânodo e cátodo atingir o limiar de condução V_{LC}, cujo valor depende do material semicondutor utilizado e da própria construção (silício 0,7V, germânio 0,3V, *shottky* 0,2V etc..). Nesta situação, diz-se que o díodo está em condução (interruptor fechado),

- Quando em condução a tensão entre o ânodo e cátodo é sempre a tensão de limiar de condução.

- Quando polarizado com tensão ânodo cátodo inferior à tensão de limiar ou polarizado em sentido inverso (tensão no cátodo maior que a tensão no ânodo) o díodo não conduz corrente, comportando-se como um interruptor aberto (díodo ao corte).

As malhas constituídas por díodos que em seguida vamos estudar, serão só analisadas do ponto de vista do seu comportamento lógico. Para tal admitiremos que V_{LC} é aproximadamente zero (díodo ideal) como mostra a Figura 2-8 c).
Estabeleçamos a seguinte relação entre valores lógicos e valores de tensão: 0V $\rightarrow$ 0 lógico e 5V $\rightarrow$ 1 lógico.
Observemos o comportamento da malha da Figura 2-9 constituía por díodos.

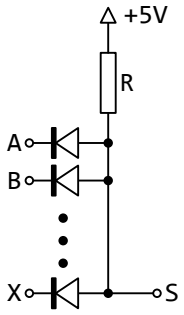


Figura 2-9

Se qualquer das entradas (A..X) estiver ao valor lógico zero, o que implica colocar o cátodo ao valor de tensão 0V, o díodo correspondente por ficar com o ânodo mais positivo que o cátodo conduz (fecha/curto-circuita) obrigando a saída S a ficar ao valor de tensão zero ($V_{LC} \cong 0$). Como consequência, todos os díodos cujo cátodo esteja ao valor de tensão 5v estarão sem conduzir (abertos) pois têm o ânodo a 0V imposto pelo(s) que conduz(em). A saída só toma o valor lógico 1 quando todas as entradas estiverem a 5V, implicando que todos os díodos estejam sem conduzir (tensão no ânodo igual à do cátodo). Esta situação leva a que não exista corrente em R o que implica que a queda de tensão nela seja igual a zero, ficando S com o valor de tensão +5v.
Pelo comportamento descrito, diremos que a malha implementa a função **AND** de n entradas, pois basta que uma das entradas esteja a zero para que a saída seja zero, só sendo 1 quando todas as entradas estiverem a 1.
Observemos o comportamento da malha da Figura 2-10 constituída por díodos.

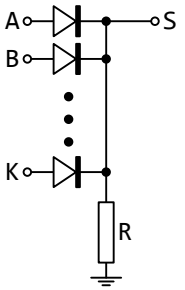


Figura 2-10

Se qualquer das entradas estiver ao valor lógico 1, ou seja, à tensão de +5V, o díodo que lhe corresponde fica com o ânodo mais positivo que o cátodo conduzindo (fecha/curto-circuita) e obrigando a saída S a ficar ao valor de tensão 5V. Isto implica que todos os díodos cujo ânodo (entrada) esteja ao valor de tensão 0V estarão sem conduzir (abertos) pois têm o cátodo mais positivo que o ânodo, imposto pelo(s) que conduz(em). A saída só toma o valor lógico 0 quando todas as entradas estiverem a 0V, implicando que todos os díodos estejam sem conduzir (tensão no ânodo igual à do cátodo). Esta condição leva a que não haja corrente em R e por conseguinte a tensão aos seus terminais ser igual a zero.
Pelo comportamento descrito, diremos que a malha implementa a função **OR** de n entradas, pois basta que uma das entradas esteja ao valor lógico 1 para que a saída tenha o valor lógico 1, só sendo 0 quando todas estiverem ao valor lógico 0.

2.1.2 Transístor

Nos actuais sistemas digitais, o elemento de comutação utilizado é o transístor, pois é mais versátil que o díodo e a sua integração é idêntica à do díodo. Embora existam duas tecnologias de transístores (bipolares e MOS), actualmente o mais utilizado é o transístor CMOS (Complementar Metal Oxide Semicondutor) por apresentar um consumo inferior ao bipolar, e por ser actualmente tão rápida quanto a tecnologia bipolar.
Por ser o mais utilizado e por ser o de mais fácil compreensão, só iremos utilizar o transístor CMOS, na análise das várias malhas que constituem as portas lógicas com transístores. O transístor tem o símbolo esquemático apresentado na Figura 2-11.

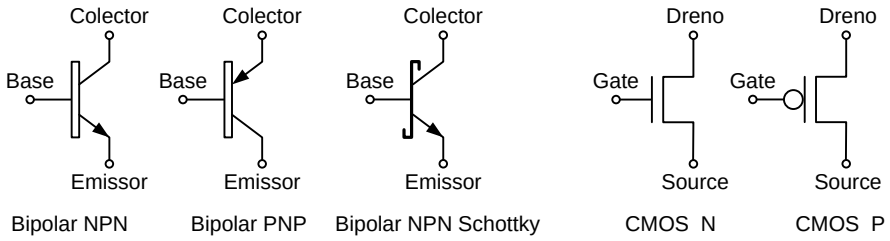


Figura 2-11

Este tipo de transístor embora tenha uma característica não linear, a sua utilização em circuitos lógicos pode ser linearizada, sendo o seu comportamento análogo ao de um interruptor, em que acção de abertura e fecho é controlada através da entrada *Gate*.
Para efeitos de análise de uma malha de comutação, o comportamento do transístor pode definir-se da seguinte maneira:

- Um transístor só estará em condução (interruptor fechado) se existir uma tensão +V na *Gate* caso seja do tipo N, ou tensão zero caso seja do tipo P.
- Quando em condução estabelece uma resistência próxima de zero entre os terminais *Drain* e *Source*, comportando-se desta forma como um interruptor fechado.
- Quando a tensão na gate for zero o transístor tipo N não conduz, estabelecendo uma resistência de centenas de mega Ohm entre o *Drain* e a *source* comportando-se como um circuito aberto. Para a mesma tensão na *Gate* o transístor tipo P conduz comportando-se como um interruptor fechado.
- Quando a tensão na *Gate* for +V o transístor tipo N fecha e o tipo P abre.

Observemos o comportamento das malhas da Figura 2-12 constituída por transístores, admitindo que a resistência entre o *Drain* e *Source* para a condução e corte são respectivamente zero e infinito (transístor ideal).

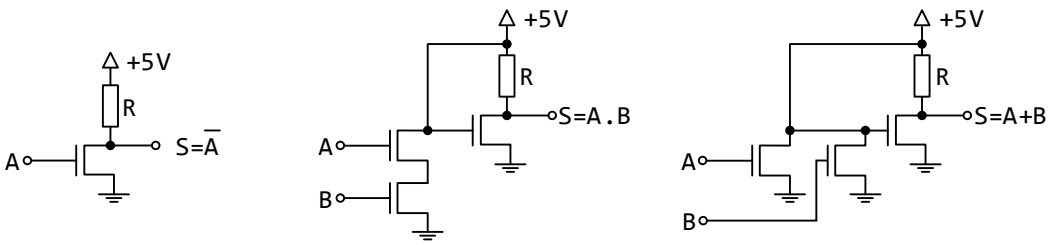


Figura 2-12

No caso da malha da esquerda, quando a tensão em A é zero, o transístor não conduz (corte) pelo que a corrente em R é zero e por conseguinte não promove queda de tensão em R, ficando a saída a +5V. Quando a tensão em A é +5V, o transístor conduz estabelecendo 0V entre *Drain* e *Source*, colocando assim a saída S a 0V. Pelo comportamento anteriormente descrito podemos concluir que o circuito desempenha a função NOT ($S = \overline{A}$).

Um dos defeitos da arquitectura mostrada na Figura 2-12, é não apresentar na saída a mesma impedância quando exibe o valor lógico 0 e 1, o que se traduz em tempos de comutação diferentes, ou seja, quando transita de 1 para 0 é mais rápido de que quando transita de 0 para 1. Esta arquitectura apresenta outros inconvenientes como sejam o maior consumo e incapacidade de atacar muitas entradas em simultâneo. Na Figura 2-13 são apresentadas as arquitecturas que constituem as actuais portas lógicas da família 7400 HCT(*High speed Complementary* MOS **T**ransístor *logic compatible*). O termo família, denota um conjunto de circuitos integrados que guardam entre si compatibilidade de interligação e semelhanças nas características e na arquitectura.

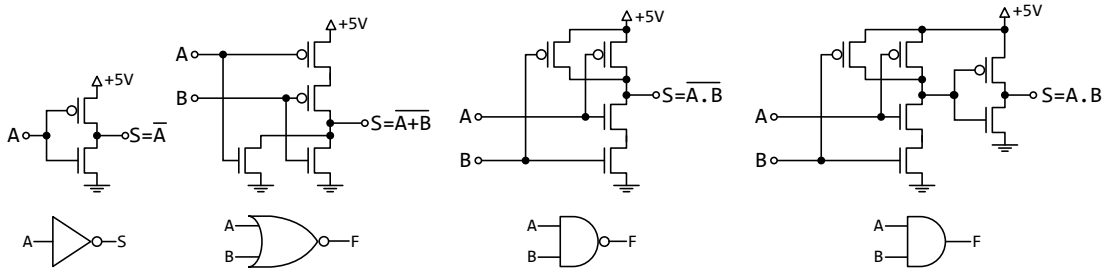


Figura 2-13

2.2 Implementação de um sistema digital utilizando Circuitos Integrados

2.2.1 Circuito Integrado Digital

A tecnologia associada ao fabrico dos circuitos integrados (IC) digitais tem vindo a desenvolver-se desde o princípio da década de 60, possibilitando a implantação de um número cada vez maior de componentes activos (transístor) numa única pastilha de silício. Os fabricantes de componentes digitais põem disponível no mercado, e sobe diversas formas de empacotamento, uma enorme variedade de ICs desempenhando as mais variadas funções. Daí o facto, de que o projecto e implementação de um sistema digital, tenha que obedecer a parâmetros muito complexos e de vária ordem no que diz respeito à escolha da tecnologia a adoptar.
Definem-se actualmente quatro escalas de integração:

SSI: (*Small Scale Integration*) Integração em pequena escala envolvendo dezenas de transístores podendo envolver uma dezena de portas lógicas.

MSI: (*Medium Scale Integration*) Integração em média escala podendo integrar uma a duas centenas de portas lógicas.

LSI: (*Large Scale Integration*) Integração em larga escala podendo integrar milhares de transístores permitindo construir um sistema digital complexo por exemplo um microprocessador.

VLSI: (*Very Large Scale Integration*) podendo integrar milhões de transístores.

A minimização do número de ICs que compõem o sistema poderá ser um critério a seguir. Admitindo que o projecto que pretendemos realizar está desenhado com recurso a portas lógicas, vejamos então como poderemos diminuir o número circuitos integrados para a sua implementação.

No mercado estão disponíveis circuitos integrados contendo uma grande variedade de portas lógicas e diversos empacotamentos. Na Figura 2-14 estão alguns exemplos de empacotamento. Embora estejam disponíveis ICs contendo portas com mais de duas entradas, iremos utilizar na implementação dos vários circuitos apenas portas lógicas de duas entradas por serem as mais vulgares.

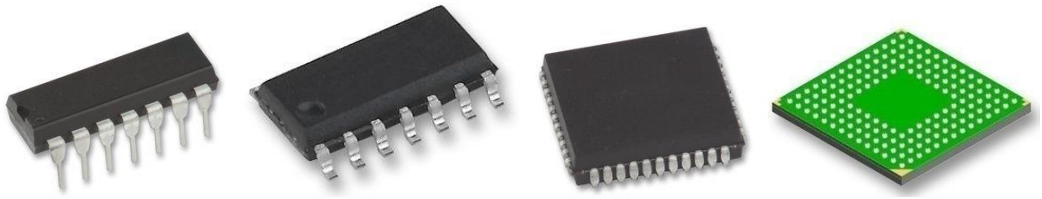


Figura 2-14

Exemplo:

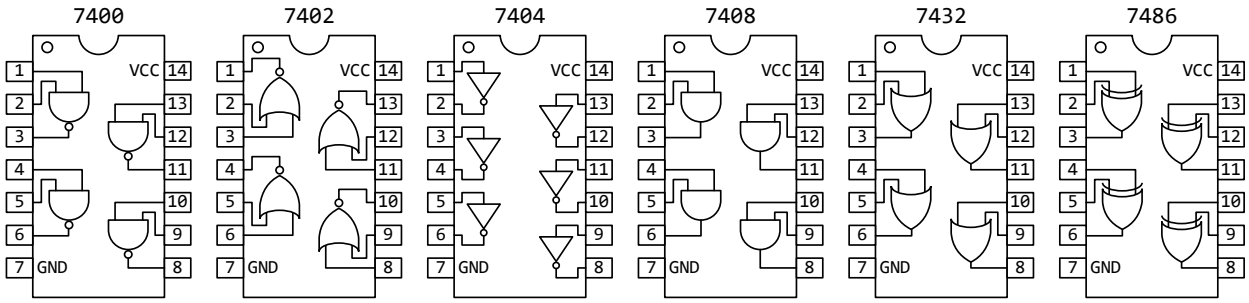


Figura 2-15

A título de exemplo, implementar a função $F = A\overline{B} + \overline{C}$ usando os circuitos integrados referidos na Figura 2-15.

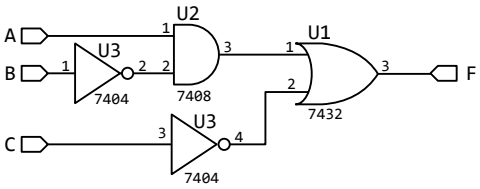


Figura 2-16

A implementação da Figura 2-16 mostra que seriam necessários três circuitos integrados, um de ANDs (U2) outro de ORs (U1) e ainda um de NOTs (U3).

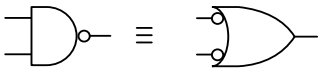
2.2.2 Operação NAND, NOR e XOR

A operação NAND tal como o NOR são funcionalmente completas, ou seja, com elas podemos sintetizar qualquer função booleana recorrendo exclusivamente a esta operação. Por esta razão os fabricantes de circuitos integrados põem disponíveis no mercado ICs constituídos exclusivamente por portas NAND ou por portas NOR.

2.2.2.1 Operação NAND

Definição: Operação sobre n variáveis que toma o valor lógico 1 quando uma ou mais entradas tomarem o valor lógico 0, ou de outra forma, só toma o valor lógico 0 quando todas as entradas tomarem o valor lógico 1.

Símbolo esquemático do operador NAND:



A função NAND apresenta as seguintes propriedades:

$\overline{\mathrm{A}.1} = \overline{\mathrm{A}}$

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

$\overline{\mathrm{A}.\mathrm{A}} = \overline{\mathrm{A}}$

$\overline{\mathrm{A}.\overline{\mathrm{A}}} = 1$

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

$\overline{\mathrm{A}.\mathrm{B}} = \overline{\mathrm{A}} + \overline{\mathrm{B}}$; teorema de DeMorgan

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

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

Da ultima propriedade podemos conclui que a operação NAND não é associativa.

Na Figura 2-17 podemos constatar que a partir de NANDs é possível sintetizar as operações NOT AND e OR.

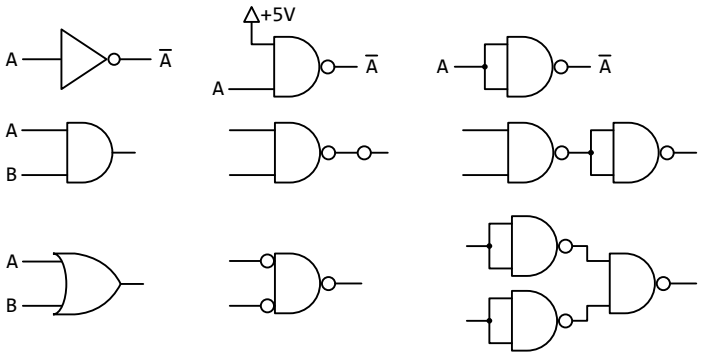
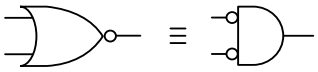


Figura 2-17

2.2.2.2 Operação NOR

Definição: Operação sobre n variáveis que toma o valor lógico 1 quando todas as entradas tomarem o valor lógico 0, ou de outra forma, toma o valor lógico 0 quando uma ou mais entradas tomarem o valor lógico 1.

Símbolo esquemático:



A função NOR apresenta as seguintes propriedades:

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

$\overline{A + 1} = 0$

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

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

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

$\overline{A + B} = \overline{A} \,.\, \overline{B}$; teorema de DeMorgan

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

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

Da última propriedade pode-se conclui que a operação NOR não é associativa.

Na Figura 2-18 podemos constatar que utilizando apenas o operador NOR é possível sintetizar as operações NOT AND e OR.

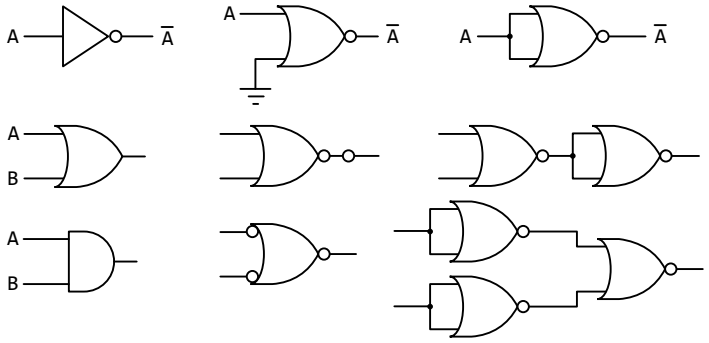


Figura 2-18

A Figura 2-19 mostra como utilizando as propriedades do operador NAND poderemos sintetizar o circuito da Figura 2-16, utilizando apenas portas NAND e assim implementar a função F com um único circuito integrado. Esta transformação é realizada segundo o princípio de que: $\overline{\overline{A}} = A$, ou seja, se negarmos um sinal (uma linha) duas vezes consecutivas, não alteramos o seu valor e desta forma tentamos gerar ORs com entradas negadas (por serem equivalentes a NANDs) e gerar ANDs de

entradas negadas (por serem equivalentes a NORs). Quando isto não é possível na totalidade realizamos o NOT utilizando o NOR ou NAND.

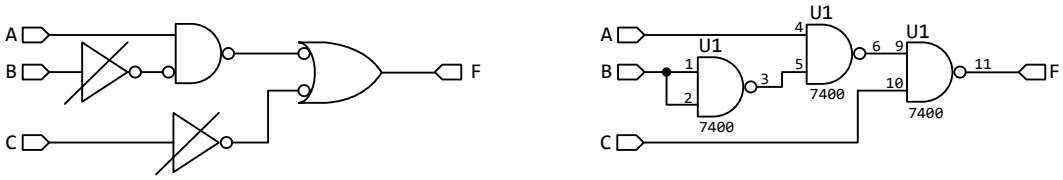


Figura 2-19

2.2.2.3 Operação XOR

A operação XOR, já anteriormente definida, apresenta algumas propriedades de grande importância no desenho de sistemas digitais. A propriedade $A \oplus 0 = A$ e $A \oplus 1 = \overline{A}$ permitem concluir como mostra a Figura 2-20 que a porta XOR pode ser encarada como programável, tomando uma das entradas como controlo.

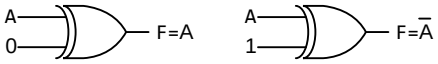


Figura 2-20

Outra propriedade importante do XOR e que é única, é o facto de uma negação poder transitar da entrada para a saída ou vice versa como mostra a Figura 2-21.

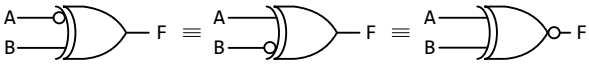


Figura 2-21

Exemplo:

Realizar uma porta lógica de duas entradas A e B que através de uma terceira entrada C possa ser programável da seguinte forma: Se a entrada C estiver ao valor lógico "0" realiza a operação NAND se estiver ao valor lógico "1" realiza a operação NOR. A porta a realizar tem o diagrama mostrado na Figura 2-22.

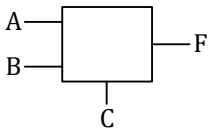


Figura 2-22

Dado que $\overline{A + B} = \overline{A}.\overline{B} = \overline{\overline{A}.\overline{B}}$, utilizando a porta NAND como base e a porta XOR para realizar o complemento ou a identidade da saída e das entradas, obtemos a solução da Figura 2-23.

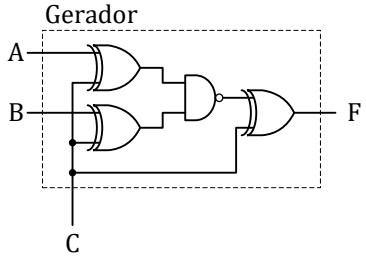
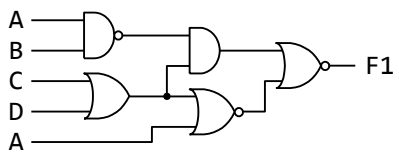


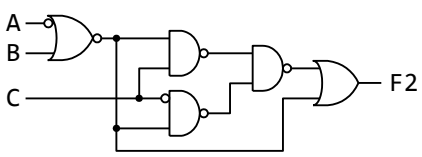
Figura 2-23

2.3 Exercícios do Capítulo 2

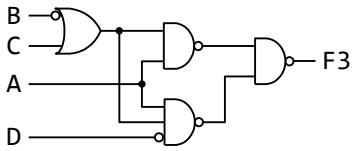
[1] Dado o esquema da figura, obtenha uma expressão simplificada de F1 na forma OR-AND e implemente a função, podendo utilizar exclusivamente portas NAND e NOR de duas entradas, não dispondo do complemento das variáveis.



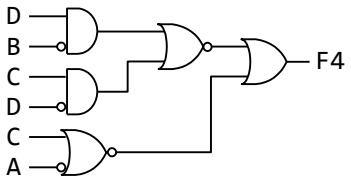
[2] Dado o esquema da figura, obtenha uma expressão simplificada de F2 na forma AND-OR e implemente a função F2, utilizando exclusivamente portas NOR de duas entradas, não dispondo do complemento das variáveis.



[3] Dado o esquema da figura, obtenha uma expressão simplificada de F3 na forma OR-AND e implemente a função F3, utilizando exclusivamente portas NOR, XOR e NAND de duas entradas.



[4] O projectista, ao testar o circuito da figura em laboratório, constatou que ao estabelecer a configuração A . B . C . D = 0 a saída F4 ficava verdadeira quando, afinal, ele pretendia que fosse falsa. Caso se trate de um erro de projecto, corrija e implemente, utilizando exclusivamente portas NAND ou NOR de duas entradas, não dispondo do complemento das variáveis.



[5] Dada a função $F5 = \left(\overline{B}.\overline{D} \oplus \overline{D}.(A+B)\right)\overline{(\overline{C}.(\overline{D}+A.B))}$, obtenha uma expressão simplificada de F5 na forma OR-AND e implemente F5 exclusivamente com portas NAND ou NOR de duas entradas sem dispor do complemento das variáveis.

[6] Dado a função $F6 = (\overline{A}+\overline{B}).\left(\overline{C}+\overline{D}\right)+A.B.\overline{(\overline{C}+\overline{D})}$, obtenha uma expressão simplificada de F6 na forma AND-OR e implemente a função F6, exclusivamente com portas NAND e XOR de duas entradas.

[7] Dada a função $F7 = \left((A \oplus D)+\left(\overline{A \oplus D} \oplus \overline{B}\right)\right).(C+D)$, obtenha uma expressão simplificada de F7 na forma OR-AND e implemente F7 exclusivamente com portas NAND e NOR de duas entradas sem dispor do complemento das variáveis. Considere que F7(D,C,B,A) tem indiferenças (*don't care*) nos termos $\sum(3, 4, 6, 15)$.

[8] Dada a função $F8 = \left(\left(C + A.\overline{B}\right) \oplus \overline{(\overline{A}.D + \overline{A}.B + A.\overline{B} + \overline{B}.\overline{D})}\right)(\overline{A}.B + A.\overline{C})$, obtenha uma expressão simplificada de F8 e implemente exclusivamente com portas NOR e NAND de duas entradas sem dispor do complemento das variáveis.

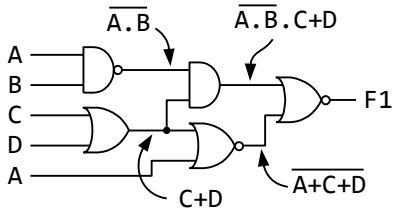
2.4 Soluções:

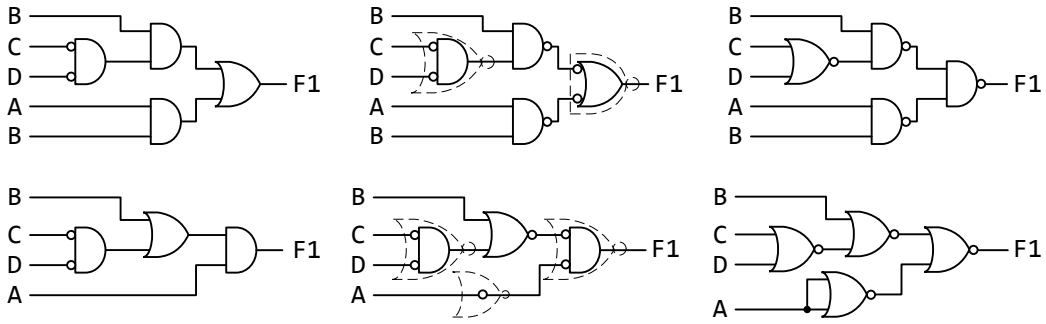
[1] Começa-se por estabelecer as expressões em cada um dos troços da esquerda para a direita obtendo-se:

$F1 = \overline{\overline{A.B}.(C+D) + \overline{A+C+D}}$

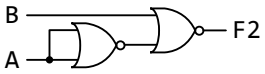
$F1 = \left(A.B + \overline{C+D}\right)(A+C+D)$

$F1 = AB + B\,\overline{C}\,\overline{D} = A.\left(B + \overline{C}.\overline{D}\right)$

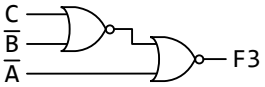




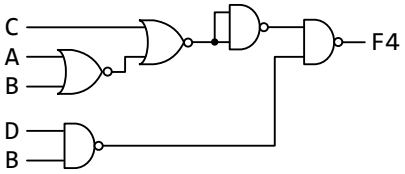
[2] $F2 = A.\overline{B}$



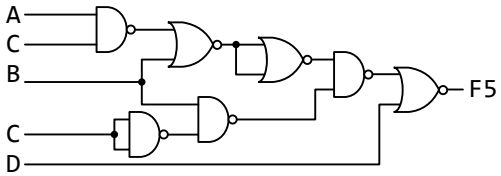
[3] $F3 = A.\left(\overline{B} + C\right)$



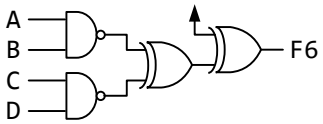
[4] Trata-se de um erro de projecto e a solução é:



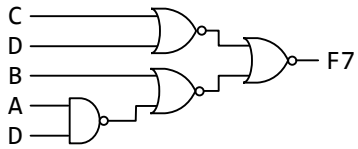
[5] $F5 = \overline{D}(\overline{A}\,\overline{B} + \overline{B}\,\overline{C} + BC) = \overline{D}(\overline{B} + C)(\overline{A} + B + \overline{C})$



[6] $F6 = A.B.C.D + \overline{B}.\overline{D} + \overline{A}.\overline{D} + \overline{B}.\overline{C} + \overline{A}.\overline{C} = \overline{A.B \oplus C.D}$



[7] $F7 = (C + D)(\overline{A} + B + \overline{D})$



[8] $F8 = A\overline{C} + \overline{A}BC$