5. MÓDULOS FUNCIONAIS DE MÉDIA COMPLEXIDADE

Como já referido anteriormente, quando pretendemos resolver problemas de maior complexidade é necessário elevar o nível de abordagem. Para tal o projectista deverá conhecer os módulos funcionais disponíveis no mercado, tal como acontece no conhecimento que é necessário ter sobre as bibliotecas das linguagens de alto nível, quando se pretende realizar projectos de alguma complexidade.

5.1 Multiplexer

A multiplexagem consiste em seleccionar (multiplexar) para a saída, em exclusão, uma de **n** entradas, podendo as entradas ser constituídas por um sinal lógico ou um vector lógico. Para determinar a cada momento qual das **n** entradas é canalizada para a saída, existirá uma palavra de selecção constituída por $\log_2(\mathbf{n})$ bits, que codifica o número da entrada seleccionada.
Na Figura 5-1 é apresentado o diagrama de um multiplexer.

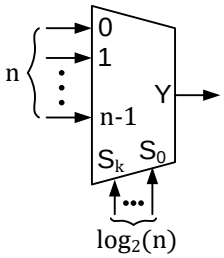


Figura 5-1

A implementação, baseia-se em poder interceptar individualmente cada uma das entradas e, posteriormente juntá-las (unir) numa única saída como mostra a Figura 5-2. Para podermos determinar qual a entrada que vai ser canalizada, ou seja, não interceptada, é necessário descodificar o código de selecção.

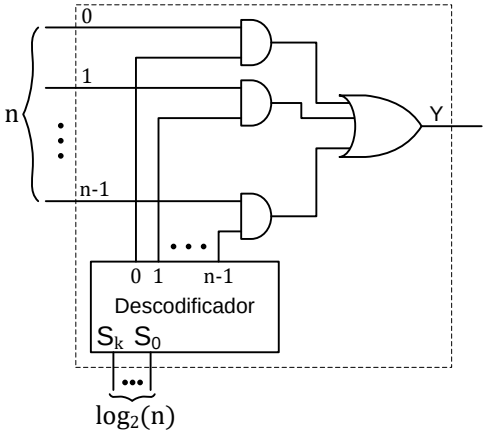


Figura 5-2

5.2 Descodificador

Num sistema digital, descodificar, é converter n bits de entrada em m bits de saída com n<=m<=2^n, tal que cada código válido de entrada produza um único código de saída. O componente que realiza esta função é denominado por descodificador. O descodificador de BCD/7 segmentos estudado anteriormente é um exemplo desta função, mas o bloco funcional que se denomina normalmente por descodificador, e que é de extrema importância nos sistemas digitais, é constituído por n bits de entrada de selecção e 2^n bits de saídas, fazendo corresponder o número da saída activa ao número binário presente na entrada de selecção. Poderemos também dizer que o descodificador gera todos os termos mínimos dos bits de selecção.

A Figura 5-3 mostra um descodificador de 2x4.

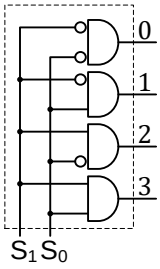
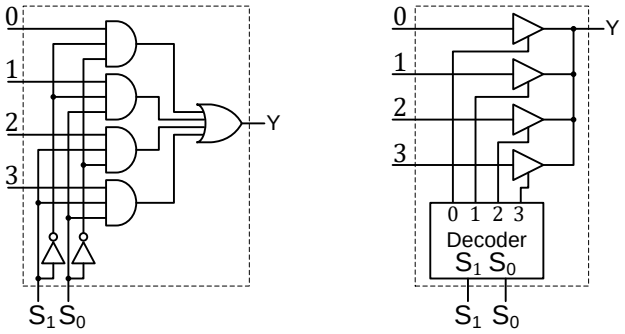


Figura 5-3

5.3 Implementações alternativas de multiplexer

Na Figura 5-4 são apresentadas duas implementações alternativas de um multiplexer de 4x1. A implementação a) corresponde à implementação já anteriormente apresentada agora com a integração da descodificação na intercepção. A alternativa b) utiliza portas com saída THREE-STATE, e é a normalmente utilizada em estruturas onde um elemento constituinte do sistema utiliza um único bus para comunicar com vários elementos espacialmente distribuídos, e onde o elemento de união será constituído pelas próprias linhas do bus.



a) b)

Figura 5-4

5.4 THREE-STATE

Uma saída diz-se ter característica THREE-STATE (três estados) se puder apresentar para além dos dois valores lógicos 0 e 1, um terceiro estado que coloque a saída em alta impedância, ou seja, quando se encontra neste estado a saída apresenta-se como se estivesse desligada da porta, não entrando por esta razão em conflito com uma outra qualquer saída que queira impor sobre ela o valor lógico 0 ou 1. Este comportamento é controlado por uma entrada designada por controlo de *three-state* que enquanto activa, leva a que a porta desempenhe a função que lhe está atribuída, e enquanto desactiva coloca a porta em *three-state*. Na implementação do multiplexer que utiliza porta identidade (buffer) com controlo de *three-state* o descodificador assegura que os vários buffers se encontram activos em exclusão.

Na Figura 5-5 está representado um esquema possível para um buffer com saída *three-state*. Quando se coloca a entrada C ao valor lógico 0, os dois transístores ficam OFF (circuito aberto) razão pela qual existe alta impedância (resistência elevada) entre S e qualquer um dos pólos +5V e 0V.

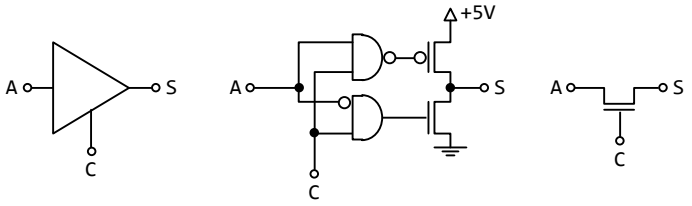


Figura 5-5

5.5 Expansão de multiplexers e descodificadores

Uma necessidade recorrente na utilização de ICs contendo descodificadores e multiplexers é sua expansão, ou seja, a necessidade de aumentar o número de entradas multiplexadas ou o número de saídas descodificadas. Essa expansão pode-se fazer juntando em cadeia vários desses elementos, no entanto é necessário tomar em conta o tempo de propagação da nova estrutura assim obtida. Na Figura 5-6 é apresentada a implementação de um multiplexer 8x1 e outro de 2x3. Para a implementação do multiplexer de 8x1 são utilizados dois multiplexers de 4x1 e um multiplexer de 2x1. O multiplexer 2x3, é constituído por duas entradas de dados, em que cada entrada tem três bits. Para a sua implementação, são utilizados três mutiplexers de 2x1.

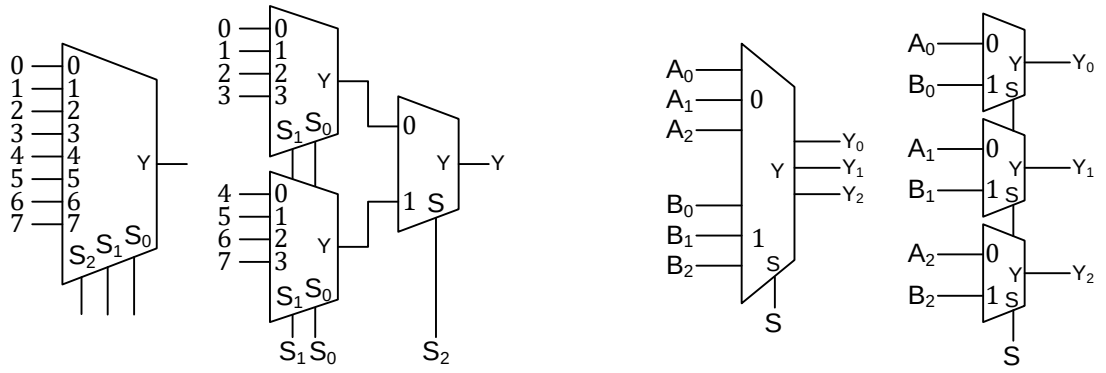


Figura 5-6

No caso do descodificador, a solução é mais simples se cada um dos descodificadores dispuser da funcionalidade de inibição e desinibição (*enabling*).

5.5.1 *Enabling*

A acção de inibição e desinibição é concretizada através de uma entrada denominada de *Enable* (permitir) que quando desactiva impede o dispositivo de desempenhar a sua normal função, ou seja, independentemente do valor lógico das restantes entradas, exibe sempre o mesmo valor de saída. No caso de o dispositivo possuir mais que uma entrada de *Enable*, o dispositivo fica desinibido quando simultaneamente as várias entradas de *Enable* ficarem activas. O valor lógico das saídas quando o dispositivo está inibido, é preestabelecido pelo fabricante e corresponde ao valor lógico zero caso a saída seja *active-high* ou ao valor lógico um, caso a saída seja *active-low*.

Na Figura 5-7 é apresentado um multiplexer de 4x1 e um descodificador 2x4, ambos com entrada de **E**.

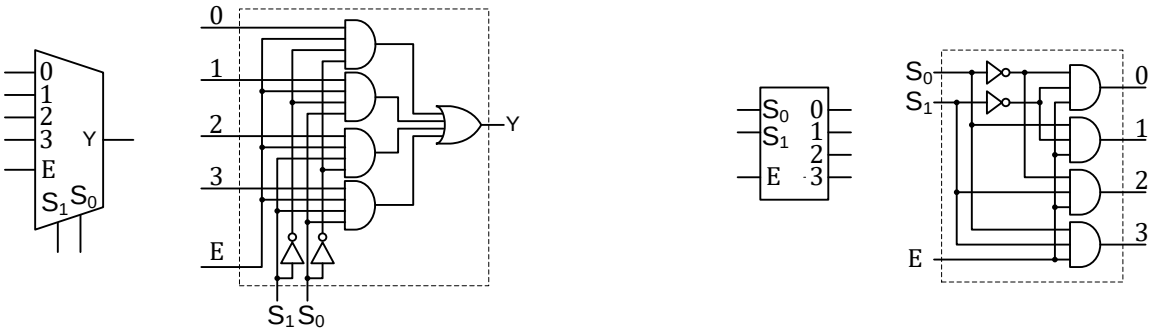


Figura 5-7

Na Figura 5-8 podemos observar expansão de um multiplexer e de um descodificador com utilização da entrada de *enable*.

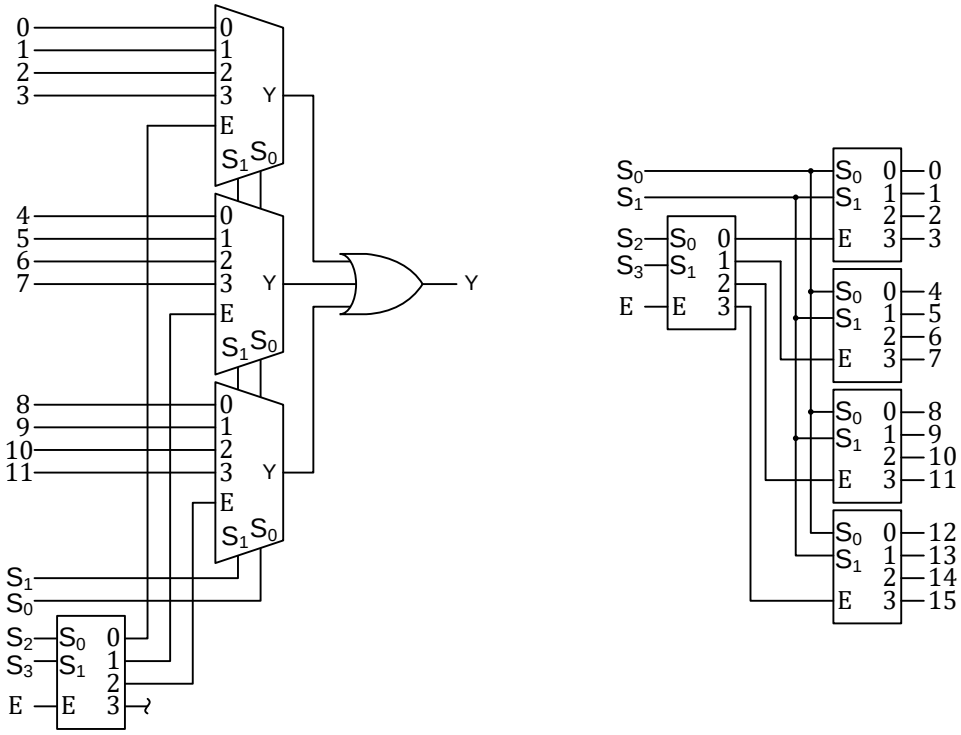


Figura 5-8

5.6 Implementação de funções utilizando multiplexers e descodificadores

O descodificador, como já vimos anteriormente, gera todos os termos mínimos das variáveis de selecção, que por sua vez aparecem em união na saída do multiplexer. Uma forma óbvia de implementar uma função com um multiplexer, é a de colocar ao valor lógico 1 as entradas correspondentes aos termos mínimos que conduzem a função a ser verdadeira.

5.6.1 Implementação com multiplexer

Tomemos como exemplo a implementação da saída C_{n+1} de um somador completo que tendo como entrada An, Bn e Cn tem como elemento de síntese um multiplexer de 8x1. Como se pode ver na Figura 5-9 também é possível a sua implementação com um multiplexer de 4x1, se tomarmos para entrada do multiplexer uma das variáveis.

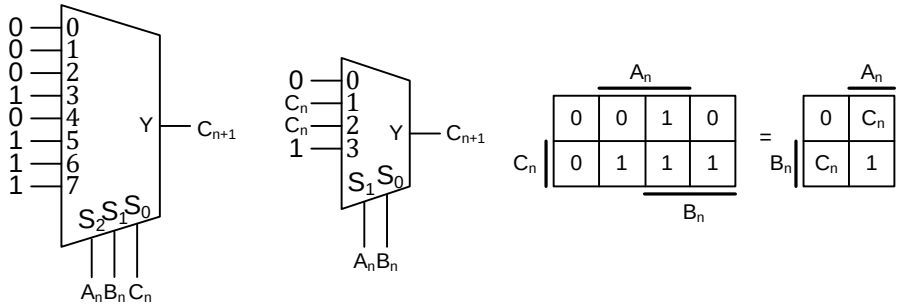


Figura 5-9

A função C_{n+1} apresenta uma simetria que permite obter o mesmo resultado independentemente das variáveis que escolhamos para selecção. No entanto para outras funções a escolha das variáveis a colocar nas entradas de selecção do multiplexer pode tornar a implementação mais ou menos complexa.

Tomemos como exemplo a função $F(A,B,C,D) = A\,\overline{C}\,\overline{D} + C\,D + B\,D$, cuja implementação natural seria um multiplexer de 16x1, a Figura 5-10 mostra uma implementação apenas com um multiplexer de 4x1. Podemos observar na Figura 5-10, que a escolha das variáveis que fazem a selecção do multiplexer influencia a complexidade da solução. Por exemplo a escolha de AB, AC ou AD para variáveis de selecção implicavam para além do multiplexer o uso de portas lógicas ou mais mutiplexers. Se as variáveis de selecção forem CD é necessário apenas um multiplexer. Como exercício poderá realizar as restantes combinações BC e BD e verificar se existe outra igualmente simples.

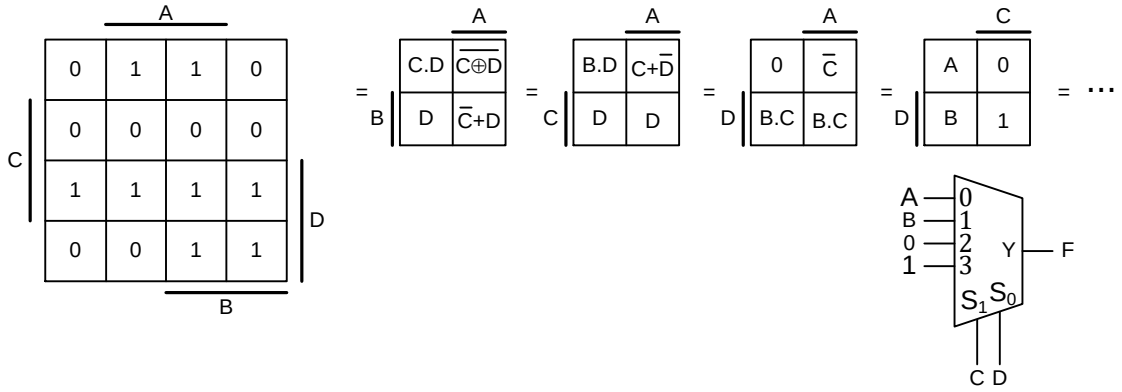


Figura 5-10

5.6.2 Implementação com descodificador

Na Figura 5-11 é apresentada a implementação de um somador completo utilizando um descodificador de 8x3. A implementação com descodificador segue a mesma lógica que a implementação com multiplexers, utiliza os termos produto gerados pelo descodificador, sendo a solução aconselhada quando geramos várias funções dependentes do mesmo conjunto de variáveis.

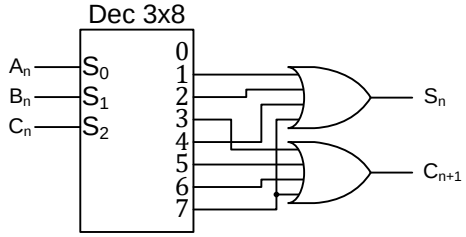


Figura 5-11

5.7 Demultiplexer

Em certas circunstâncias a informação que foi multiplexada, necessita posteriormente de ser demultiplexada (distribuída por n saídas), que corresponde ao inverso da multiplexagem. A informação recebida numa linha é posta disponível numa das 2^n possíveis saídas. O circuito que realiza esta acção é denominado por demultiplexer. A saída específica para onde o sinal de entrada é transmitido, é controlada pela combinação binária presente nas n linhas de selecção. Para realizar esta acção não é necessário criar um novo componente, pois podemos utilizar um descodificador com entrada de *enable* e utilizar a entrada de *enable* como a entrada do sinal a demultiplexar como é mostrado na Figura 5-12.

Se $Data_{in}$ tomar o valor lógico um, activa a entrada EN, levando a saída seleccionada pelos bits $A_{0\text{-}2}$ a tomar o valor lógico um. Se $Data_{in}$ estiver ao valor lógico zero desactiva a entrada EN e leva a saída seleccionada ao valor lógico zero.

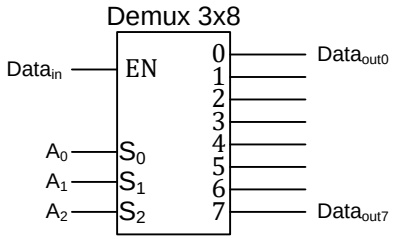


Figura 5-12

5.8 Codificador

O conceito de codificação e descodificação estão associados à ideia de compressão e descompressão. Como vimos anteriormente no caso do descodificador de 7 segmentos, tínhamos na entrada quatro bits codificados em código *Gray* que eram descodificados para 7 saídas que controlavam os 7 segmentos que desenhavam os 10 algarismos decimais.

A codificação, por oposição à descodificação, consiste em comprimir n entradas em k saídas com k<n.

5.8.1 *Priority Encoder*

Tomemos como exemplo a síntese de um circuito, que tendo como entrada um teclado constituído por dez actuadores de 0 a 9, ponha disponível em quatro bits a codificação em binário natural do actuador que está activado. Para codificar as dez teclas necessitamos de 4 bits (2^3=8, 2^4=16). Como a configuração de entrada é gerada por dez teclas independentes, então é possível que sejam premidas várias teclas em simultâneo. Por esta razão é necessário que a codificação se faça estabelecendo exclusão entre as entradas com um esquema de prioridades. Para que se reconheça (valide) que o código presente na saída corresponde a uma tecla premida, é necessário que o codificador tenha um dos seguintes comportamentos: ou exiba um código diferente dos restantes quando não exista nenhuma tecla premida, ou disponibilize um sinal de saída GS (*Group Select*) informando que existe uma ou mais entradas activas. A primeira solução nem sempre é aplicável, pois podemos, no problema em questão, necessitar das 16 configurações. Por esta razão a solução adoptada, é a de adicionar uma saída denominada GS que, quando activa indica que existe pelo menos uma entrada activa, obtendo-se assim o diagrama mostrado na Figura 5-13.

Se o sistema de actuadores pela sua natureza construtiva garantir que não é activada mais que uma entrada em simultâneo então o descodificador pode ter menor complexidade e será simplesmente denominado por *encoder*.

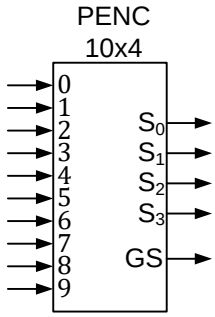


Figura 5-13

Tratando-se de um circuito combinatório de 10 entradas, obter as expressões de cada uma das quatro saídas de codificação, torna-se um problema relativamente complexo. Por esta razão iremos sintetizar o PENC de 10x4 à custa de PENCs de 4x2 concatenados como fizemos anteriormente na expansão de multiplexers e descodificadores. Na Figura 5-14 estão representadas duas soluções, uma utilizando concatenação com multiplexers e outra utilizando o método de *enabling*.

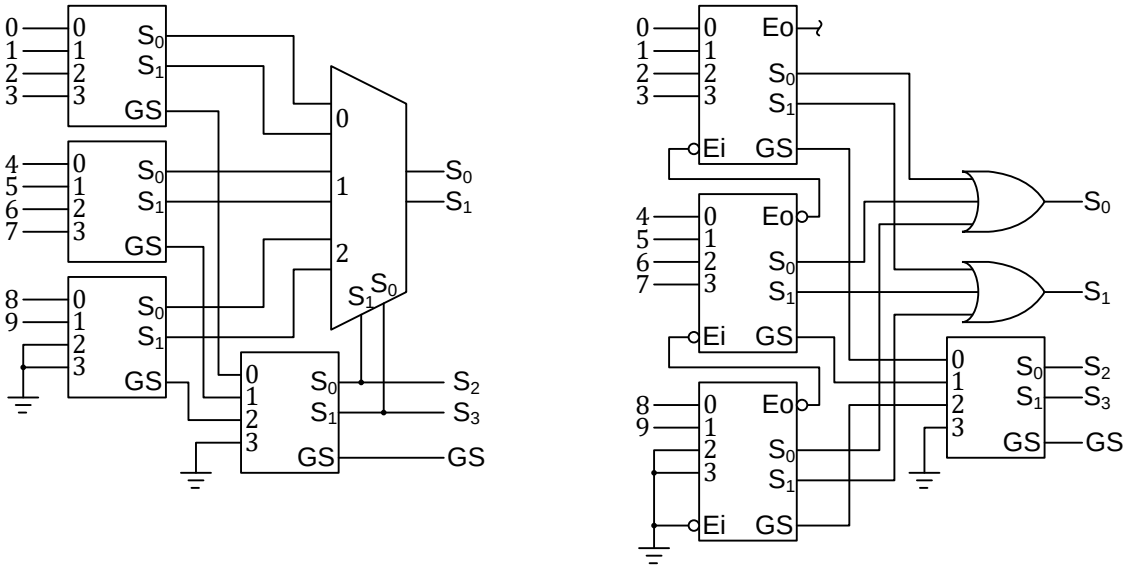
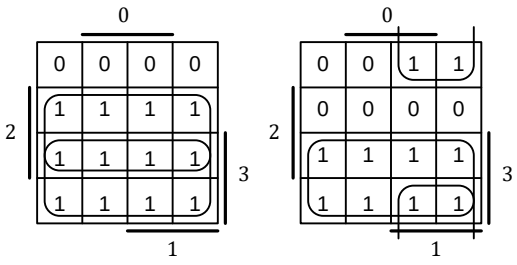


Figura 5-14

Na Figura 5-15 está representada a implementação de um PENC de 4 entradas utilizando portas lógicas.

Ei 3	2	1	0	S_1	S_0	Eo	GS
0	-	-	-	0	0	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	-	0	1	0	1
1	1	-	-	1	0	0	1



$S_1=2+3$ $S_0=(1.\overline{2})+3$

$GS=Ei(0+1+2+3)$

$Eo=Ei.\overline{GS}$

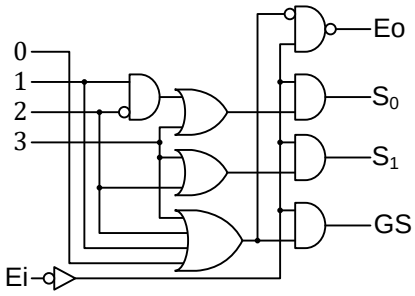


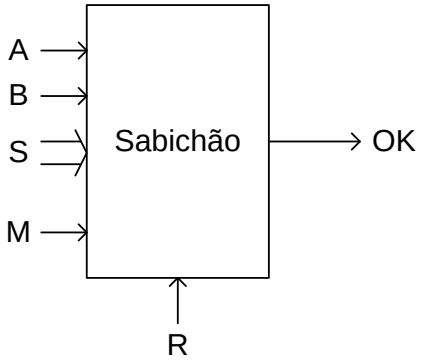
Figura 5-15

Exercício:

Pretende-se construir um sistema, que denominaremos por Sabichão, que tem por objectivo testar o conhecimento acerca do comportamento de portas lógicas de duas entradas. O sistema é operado por um examinador que estabelece qual a porta lógica sobre a qual o aluno vai ser examinado, e os valores lógicos presentes nas duas entradas. O examinando responde qual o valor que deve ficar presente na saída da porta, dado os valores presentes às entradas. Durante a preparação da pergunta por parte do examinador e a resposta por parte do examinando, o sistema deverá manter inibida a saída que indica o resultado do teste. A figura mostra o diagrama do sistema onde se podem identificar as várias entradas e saídas.
Modo de operação do sistema:

1º. O examinador selecciona, mediante um conjunto de sinais de selecção **S**, uma das seguintes portas: (AND, OR ou NAND) e estabelece quais os valores lógicos das entradas através dos sinais **A** e **B**;
2º. O examinando responde através da entrada **R** qual o valor lógico que a porta seleccionada deve exibir na saída mediante estas condições;
3º. O examinador activa a entrada **M**, para mostrar na saída **OK** se a resposta do examinando está correcta.

O sistema deve permitir a realização do exame por um examinador que desconheça o comportamento das várias portas que o sistema avalia.



Solução:

O projecto pode ser abordado como um único problema, ou decomposto em outros de menor complexidade. A classificação das variáveis em grupos característicos indicia estratégias de abordagem por blocos funcionais.
Se o projecto for abordado como um único problema, trata-se da observação exaustiva das 64 combinações possíveis das seis entradas, o que torna o exercício complexo e de difícil simplificação e teste da solução encontrada.
Vamos decompor o problema estabelecendo um diagrama de blocos como mostra a Figura 5-16, com as várias funcionalidades pretendidas.
Poderemos pensar inicialmente em três blocos: um gerador que mediante a porta seleccionada pelo examinador e os valores presentes nas entradas **A** e **B** produza na saída (**VC**) o valor lógico correcto; outro bloco que compare a resposta **R** do examinando com o valor **VC** do gerador e um terceiro bloco que iniba o valor **C** resultado da comparação.

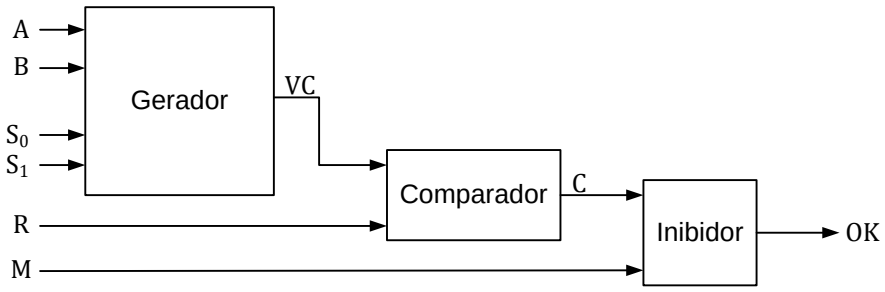


Figura 5-16

Quanto à solução para o gerador poderemos recorrer a uma de três hipóteses.

1ª. Estabelecemos um mapa de Karnaugh de quatro variáveis e extraímos uma expressão simplificada.

2ª. Utilizamos um multiplexer de 4x1 e colocamos nas entradas as portas AND, OR e NAND como mostra a Figura 5-17.

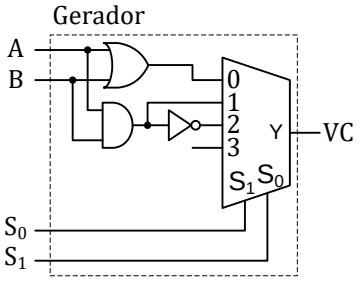


Figura 5-17

3ª. Baseado na lei de De Morgan em que poderemos a partir de uma qualquer porta lógica obter qualquer outra por negação das entradas e saídas e sabendo que a porta lógica XOR pode comportar-se como uma porta entidade ou inversora por controlo de uma das entradas, poderemos obter a solução mostrada na Figura 5-18.

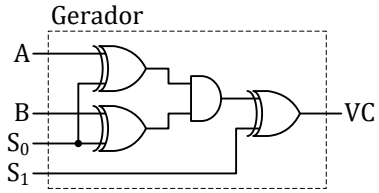


Figura 5-18

Quanto ao módulo comparador, facilmente se deduz que se trata de um XNOR.
O módulo Inibidor corresponde a uma intercepção da saída C pela entrada M.
Obtendo-se assim o esquema da Figura 5-19.

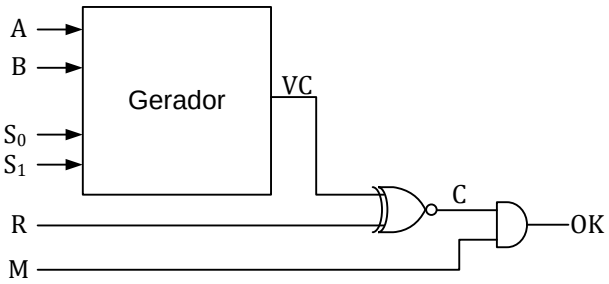


Figura 5-19

Como já foi referido anteriormente a melhor solução depende da tecnologia de implementação que estivermos a utilizar.