Lógica combinacional modular

Última revisão em 01 de abril de 2016

Este documento é parte das notas de aula da disciplina MAC0329 (Álgebra Booleana e Circuitos Digitais). Nesta parte abordaremos a lógica combinacional modular.

Da mesma forma que funções em programas computacionais são reutilizadas diversas vezes, subcircuitos em circuitos lógicos podem ser reutilizados várias vezes. Do ponto de vista de quem os usa, em ambos os casos, os aspectos dos módulos que realmente interessam são a sua funcionalidade (o que ele faz), as suas entradas e saídas.

No caso de circuitos lógicos, alguns subcircuitos, tais como decodificadores, codificadores, multiplexadores e demultiplexadores, podem ser bastante úteis no desenho de sistemas digitais.

1 Multiplexadores (Seletores de dados ou MUX)

Multiplexadores são circuitos combinacionais com n entradas e uma saída. Apenas uma entrada é selecionada para ser enviada à saída. A entrada selecionada é justamente aquela cujo índice corresponde ao valor especificado pelos seletores, que consistem de k entradas binárias, com $k \geq \log_2 n$. A figura 1 mostra um multiplexador 4×1, isto é, um multiplexador de 4 entradas.

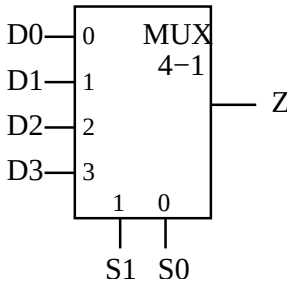


Figura 1: Multiplexador de 4 entradas.

Se os seletores forem $S_1S_0 = 00$, então teremos $Z = D_0$; se forem $S_1S_0 = 01$, então teremos $Z = D_1$; $S_1S_0 = 10$, então teremos $Z = D_2$; $S_1S_0 = 11$, então teremos $Z = D_3$.

Podemos observar que Z é uma função das variáveis S_0, S_1 e D_0, D_1, D_2, D_4. Assim, podemos escrever Z como

$$Z(D_0, D_1, D_2, D_4, S_1, S_0) = D_0\,\overline{S}_0\,\overline{S}_1 + D_1\,\overline{S}_0\,S_1 + D_2\,S_0\,\overline{S}_1 + D_3\,S_0\,S_1$$

e portanto multiplexadores podem ser realizados com circuitos dois-níveis, consistindo de n portas E no primeiro nível e uma porta OU no segundo nível, conforme mostrado na figura 2. Note que para cada possível combinação de valores de S_1S_0, apenas um dos produtos toma valor 1 na equação acima.

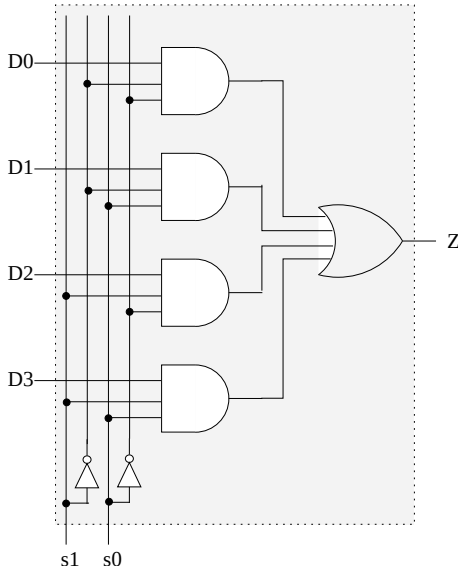


Figura 2: Circuito de um multiplexador de 4 entradas.

1.1 Realização de funções com MUX

Funções podem ser realizadas utilizando-se MUX. Para tanto, deve-se escolher as variáveis que funcionarão como seletores e, em seguida, as n entradas que poderão ou não depender das demais variáveis.

Exemplo: Realizar a função $f(a, b, c) = \overline{a}\,\overline{b} + a\,c$ usando um MUX 4×1, com as variáveis a e b como seletores.

Uma possível forma para fazer isso é expandir a expressão da função de forma que os literais correspondentes às variáveis a e b apareçam em todos os produtos da expressão resultante. No caso da função dada, fazemos

$$f(a, b, c) \;=\; \overline{a}\,\overline{b} + a\,c$$

$$\begin{aligned} &= \overline{a}\,\overline{b} + a\,(b + \overline{b})\,c \\ &= \overline{a}\,\overline{b} + a\,b\,c + a\,\overline{b}\,c \end{aligned}$$

Assim, no MUX 4×1 basta fazermos $s_1 = a$, $s_0 = b$, $D_0 = 1$, $D_1 = 0$, $D_2 = c$ e $D_3 = c$.

Exercício 1: Escreva a realização da função $f(a, b, c) = \overline{a}\,\overline{b} + a\,c$ usando um MUX 8×1, com as variáveis a, b e c como seletores.

Exercício 2: Escreva a realização da função $f(a, b, c, d) = \sum m(0, 1, 3, 6, 7, 8, 11, 12, 14)$ usando um MUX 8×1, com as variáveis a, b e c como seletores.

Exercício 3: Escreva a realização da função $f(a, b, c, d) = \sum m(0, 1, 3, 6, 7, 8, 11, 12, 14)$ usando um MUX 4×1, com as variáveis a e b (neste caso, as entradas possivelmente dependerão das variáveis c e d e serão necessárias portas adicionais para a realização de f).

1.2 Realização de MUX como composição de MUX

Um MUX pode ser realizado como composição de MUXes com um menor número de entradas. Por exemplo, um MUX 4×1 pode ser realizado através de 3 MUX 2×1, conforme ilustrado na figura 3.

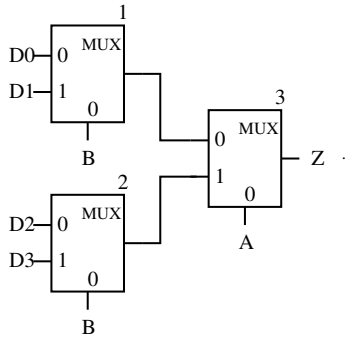


Figura 3: Realização de um MUX 4×1 como composição de três MUX 2×1.

Note que se $AB = 00$, então como $B = 0$ a saída do MUX 1 é D_0 e a do MUX 2 é D_2 e, como $A = 0$, a saída do MUX 3 é D_0. Se $AB = 01$ então as saídas são, respectivamente, D_1, D_3 e D_1. Se $AB = 10$, então as saídas são, respectivamente, D_0, D_2 e D_2. Finalmente, se $AB = 11$, então as saídas são, respectivamente, D_1, D_2 e D_3. Em

todos os casos, a terceira saída é a mesma de um MUX 4×1 com AB como entrada para os seletores.

Pergunta: E se trocarmos as entradas para os seletores na figura acima? Se colocarmos A no seletor dos MUX 1 e 2 e B na do MUX 3, ainda é possível realizar um MUX 4×1 com AB como entrada para seletores ?

1.3 Realização multi-níveis de funções com MUX

Uma função pode ser realizada com múltiplos níveis de multiplexadores. Para cada nível deve-se definir as variáveis que alimentarão os seletores. Em função disso ficam definidas as variáveis que alimentarão as entradas dos multiplexadores no primeiro nível.

Considere a função $f(a,b,c,d) = \sum m(2,5,8,9,11,12,14,15)$. Como visto acima, se for utilizado um MUX 8×1, então serão necessários três variáveis para alimentar os seletores. A quarta variável pode ser diretamente alimentada nas entradas, conforme mostrado na figura 4.

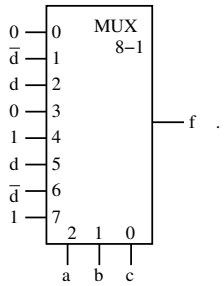


Figura 4: Realização de $f(a,b,c,d) = \sum m(2,5,8,9,11,12,14,15)$ com um MUX 8×1.

Se pensarmos em utilizar MUX 4×1 e MUX 2×1 na realização de f, duas possíveis soluções, mostradas na figura 5, são:

1. dois MUX 4×1 no primeiro nível, alimentando um MUX 2×1 no segundo nível
2. 4 MUX 2×1 no primeiro nível e 1 MUX 4×1 no segundo nível.

Estas estruturas podem ser obtidas a partir da análise dos mintermos arranjados em forma tabular, conforme mostrado a seguir. A tabela da esquerda considera o uso das

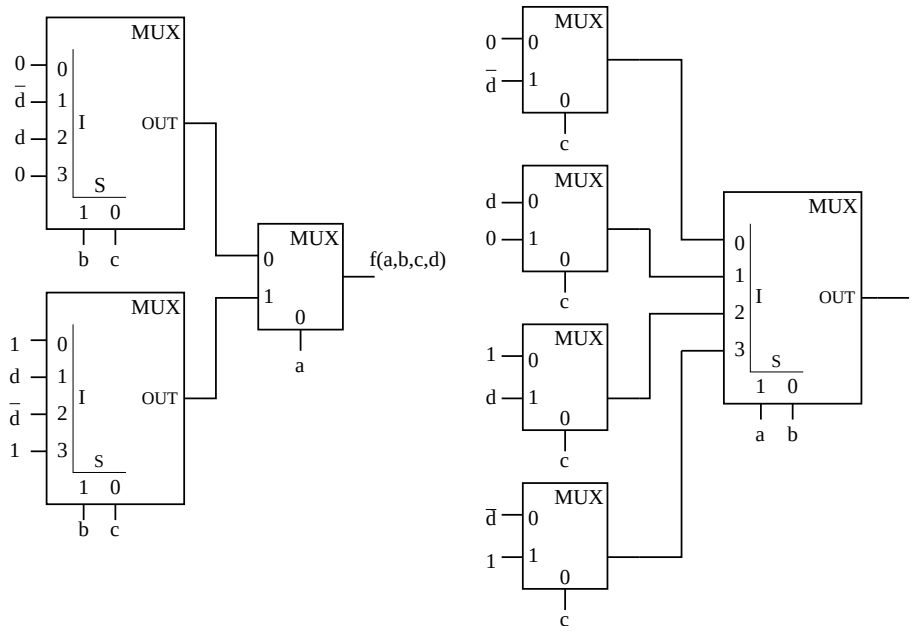


Figura 5: Realização de $f(a,b,c,d) = \sum m(2,5,8,9,11,12,14,15)$ com (a) dois MUX 4×1 no primeiro nível e um MUX 2×1 no segundo nível e (b) 4 MUX 2×1 no primeiro nível e 1 MUX 4×1 no segundo nível.

entradas b e c como seletores dos MUXes 4×1 no primeiro nível e o uso da varíável a como seletor do MUX 2×1 do segundo nível. A tabela da direita faz o análogo para a implementação com 4 MUX 2×1 no primeiro nível e 1 MUX 4×1 no segundo nível.

$abcd$	a	bc	d	input
0010	0	01	0	$\overline{d}$
0101	0	10	1	d
1000	1	00	0	$\overline{d}+d=1$
1001	1	00	1	
1100	1	10	0	$\overline{d}$
1110	1	11	0	$\overline{d}+d=1$
1111	1	11	1	

$abcd$	ab	c	d	input
0010	00	1	0	$\overline{d}$
0101	01	0	1	d
1000	10	0	0	$\overline{d}+d=1$
1001	10	0	1	
1100	11	0	0	$\overline{d}$
1110	11	1	0	$\overline{d}+d=1$
1111	11	1	1	

Outra abordagem para determinar a estrutura hierárquica dos MUXes na realizações de funções com múltiplos níveis de MUXes é baseada na aplicação sucessiva da expansão de Shannon. Dependendo da sequência de variáveis em torno das quais a expansão é aplicada, pode-se chegar a diferentes estruturas.

O teorema de **Expansão de Shannon** afirma que qualquer função f de n variáveis

pode ser escrita em termos de funções de $n-1$ variáveis da seguinte forma:

$$f(x_1,\ldots,x_k,\ldots,x_n) = \overline{x}_i f(x_1,\ldots,0,\ldots,x_n) + x_i f(x_1,\ldots,1,\ldots,x_n)$$

As funções $f(x_1,\ldots,0,\ldots,x_n)$ e $f(x_1,\ldots,1,\ldots,x_n)$ são funções de $n-1$ variáveis. O teorema pode ser aplicado recursivamente nessas duas funções.

Veremos esse teorema novamente quando formos tratar de aspectos algébricos das funções lógicas. No exemplo abaixo são aplicadas algumas manipulações algébricas.

Exemplo: Consideremos novamente a função $f(a,b,c,d) = \sum m(2,5,8,9,11,12,14,15)$. Sua realização com um MUX 8×1 está mostrada na figura 4 acima. Vamos mostrar agora que aplicando-se sucessivamente a expansão de Shannon é possível obter diferentes estruturas de realizações de f usando MUX.

$$\begin{aligned}
f &= \overline{a}\,\overline{b}\,c\,\overline{d} + \overline{a}\,b\,\overline{c}\,d + a\,\overline{b}\,\overline{c}\,\overline{d} + a\,\overline{b}\,\overline{c}\,d + a\,\overline{b}\,c\,d + a\,b\,\overline{c}\,\overline{d} + a\,b\,c\,\overline{d} + a\,b\,c\,d \\
&= \overline{a}\,\overline{b}\,c\,\overline{d} + \overline{a}\,b\,\overline{c}\,d + a\,\overline{b}\,\overline{c}\,\overline{d} + a\,b\,\overline{c}\,\overline{d} + a\,\overline{b}\,\overline{c}\,d + a\,\overline{b}\,c\,d + a\,b\,c\,\overline{d} + a\,b\,c\,d \quad \text{(rearranjo)} \\
&= \overline{a}\,\overline{b}\,c\,\overline{d} + \overline{a}\,b\,\overline{c}\,d + a\,\overline{c}\,\overline{d} + a\,\overline{b}\,d + a\,b\,c \quad \text{(algumas simplificações)} \\
&= \overline{a}\,(\overline{b}\,c\,\overline{d} + b\,\overline{c}\,d) + a(\overline{c}\,\overline{d} + \overline{b}\,d + b\,c) \quad \text{(expansão em torno de } a) \\
&= \overline{a}\,(\overline{b}(c\,\overline{d}) + b(\overline{c}\,d)) + a\,(\overline{b}(\overline{c}\,\overline{d} + d) + b\,(\overline{c}\,\overline{d} + c)) \quad \text{(expansão em torno de } b) \\
&= \overline{a}\,\overline{b}(c\,\overline{d}) + \overline{a}\,b(\overline{c}\,d) + a\,\overline{b}(\overline{c}\,\overline{d} + d) + a\,b\,(\overline{c}\,\overline{d} + c) \quad \text{(distribuição com respeito a } a) \\
&= \overline{a}\,\overline{b}(\overline{c}(0) + c\,(\overline{d})) + \overline{a}\,b(\overline{c}\,(d) + c(0)) + a\,\overline{b}(\overline{c}\,(1) + c(d)) + a\,b\,(\overline{c}\,(\overline{d}) + c(1)) \quad \text{(expansão em torno de } c)
\end{aligned}$$

A última expressão acima pode ser realizada com 4 MUX 2×1 no primeiro nível, com c como entrada para os seletores, mais um MUX 4×1 no segundo nível, com a e b como entrada para os seletores.

Nas equações acima, logo após a expansão em torno de b, poderíamos ter prosseguido da seguinte forma:

$$\begin{aligned}
f &= \overline{a}\,(\overline{b}(c\,\overline{d}) + b(\overline{c}\,d)) + a\,(\overline{b}(\overline{c}\,\overline{d} + d) + b\,(\overline{c}\,\overline{d} + c)) \quad \text{(expansão em torno de } b) \\
&= \overline{a}\,(\overline{b}\,c\,(\overline{d}) + \overline{b}\,\overline{c}(0) + b\,\overline{c}\,(d) + b\,c\,(0)) + a\,(\overline{b}\overline{c}\,(1) + \overline{b}\,c(d) + b\,\overline{c}\,(\overline{d}) + b\,c\,(1)))
\end{aligned}$$

Esta última expressão pode ser realizada com 2 MUX 4×1 no primeiro nível, com b e c como entrada para os seletores, mais um MUX 2×1 no segundo nível, com a como entrada para os seletores.

2 Demultiplexadores (DMUX ou Distribuidores de dados)

Demultiplexadores são circuitos combinacionais inversos aos multiplexadores, isto é, possuem apenas uma entrada que é transmitida a apenas uma das n saídas. A saída selecionada é aquela cujo índice corresponde ao valor correspondente aos seletores. Se o demultiplexador possui n saídas, então são necessários k seletores, com $2^k \geq n$.

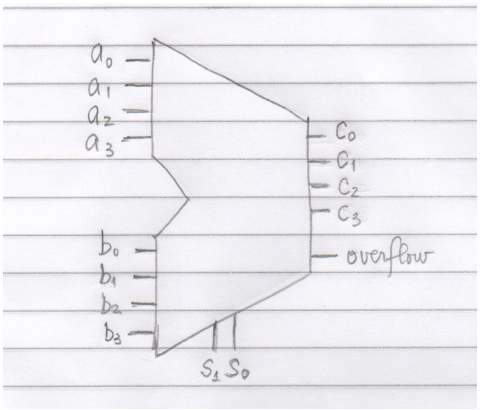
Uma implementação simples de DMUX consiste em n portas OU. Cada saída z_i, $i = 1, 2, \ldots, n$ corresponde à função dada por

$$z_i(d, s_{k-1}, \ldots, s_1, s_o) = d\, m_i$$

onde d é a variável de entrada e m_i é o mintermo correspondente a i. Por exemplo, se $k = 2$ então $z_1 = d\overline{s}_1\, s_o$ ($m_1 = \overline{s}_1\, s_o$.

3 Exemplos de utilização de MUX e DMUX

ULA A ULA (unidade lógico-aritmética) é a parte do processador responsável pelas operações lógicas (comparações basicamente) e pelas operações aritméticas. Um esquema da ULA, para palavras de 4 bits, é mostrado na figura a seguir



As entradas são oito bits ($A = a_3\, a_2\, a_1\, a_0$ e $B = b_3\, b_2\, b_1\, b_0$) correspondendo aos dois números de entrada, e mais os bits seletores $S = s_1\, s_0$ que indicam a operação a ser executada.

Uma ULA tipicamente contém subcircuitos para as várias operações que ela é capaz de executar. Para que a operação indicada nos seletores possa ser executada, pode-se usar um DMUX para transmitir os dados de entrada para o subcircuito adequado e um MUX para transmitir a saída desse subcircuito para a saída da ULA. Desta forma, garante-se que a saída da ULA será o resultado da operação especificada.

Transmissão serial Suponha que temos dois subsistemas A e B, e que A possui n saídas que geram sinais que devem ser transmitidos para os respectivos n receptores de B. A transmissão dos n sinais seria direta se houvesse um canal de comunicação entre cada saída de A para a respectiva entrada de B. No entanto, em algumas situações pode haver apenas um canal de transmissão entre A e B. Para transmitir os n sinais, pode-se transmiti-los sequencialmente, um por vez. Para tanto, pode-se utilizar um multiplexador na saída de A e um demultiplexador na entrada de B, com sincronização dos seletores. Se os valores dos seletores do MUX e do DMUX forem os mesmos, digamos i, o MUX seleciona sua entrada i (portanto, a saída i de A) para ser trasnmitido, e na outra ponta, o DMUX irá direcionar o sinal recebido para a saída i (portanto, para a entrada i do sistema B). Ao se variar sucessivamente o valor dos seletores de 0 a $n-1$, pode-se realizar a transmissão sucessiva dos n sinais.

4 Decodificadores

Decodificadores são circuitos combinacionais com n entradas e 2^n saídas. As n entradas são interpretadas como um número em binário e portanto elas podem codificar os números de 0 a $2^n - 1$. Apenas uma única saída é ativada; justamente aquela cujo índice corresponde ao valor codificado na entrada. A figura 6 mostra um esquema genérico de um decodificador $n : 2^n$. As entradas $x_{n-1} \ldots x_1\, x_0$ são interpretadas como um número binário entre 0 e $2^n - 1$ e tem-se $z_i = 1 \iff \sum_{k=0}^{n-1} x_k\, 2^k = i$.

Exemplo: Seja o decodificador $2 : 4$ e suponha que as entradas são $B\,A$ (i.e., $x_0 = A$ e $x_1 = B$). Um circuito correpondente ao decodificador é ilustrado na figura 7. No caso temos $z_0 = \overline{B}\,\overline{A}$, $z_1 = \overline{B}\,A$, $z_2 = B\,\overline{A}$ e $z_3 = BA$.

Conceitualmente, o circuito acima poderia ser estendido para realizar decodificadores $n : 2^n$, para um valor de n arbitrariamente grande. No entanto, na prática existem limitações tecnológicas (físicas) conhecidas como *fan-in* (número máximo de entradas

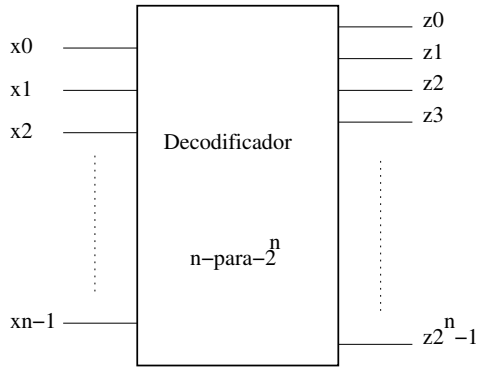


Figura 6: Esquema de um decodificador $n : 2^n$.

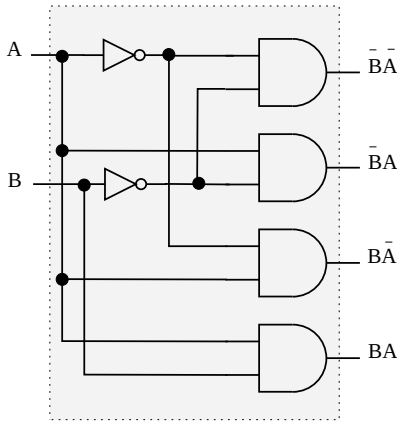


Figura 7: Circuito de um decodificador $2 : 4$.

possíveis em uma porta lógica) que restringem este valor n. Para valores grandes de n, decodificadores são realizados por circuitos multi-níveis, conforme veremos mais adiante.

4.1 Memórias ROM

Um exemplo de uso de decodificadores são as memórias do tipo ROM (*Read-Only Memory*). Suponha por exemplo uma memória com 4 posições. O endereço destas posições pode ser codificado em dois bits $x_1\, x_0$. Decodificadores podem ser utilizados para se endereçar uma certa posição na memória, gerando-se um sinal na linha de saída que corresponde à posição a ser endereçada.

A cada endereço $(x_1\, x_0)$ fonecido como entrada do decodificador, apenas uma saída do decodificador ficará ativa. A palavra na posição de memória endereçada (isto é, o dado armazenado na linha de saída ativada) será transmitida para a saída $(z_3\, z_2\, z_1\, z_0)$ (note que o esquema da figura está simplificado; a rigor, cada porta OU possui exatamente

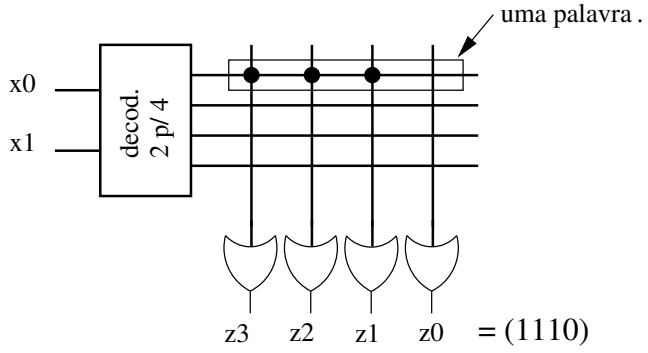


Figura 8: Esquema de uma memória ROM com 4 posições e palavras de 4 bits.

4 entradas que poderão ou não estar conectadas a cada uma das linhas de saída do decodificador).

Memórias ROM possuem uma estrutura semelhante aos PLAs (i.e., um plano de portas E e um plano de portas OU)PLA's serão descritos em um outro documento. As diferenças em relação a um PLA são o fato de que ROMs possuem necessariamente 2^n portas E (todos os produtos são canônicos) e apenas o plano OU é programável. Se o plano OU tem conexões fixas, trata-se de uma ROM. Se o plano OU pode ser programado, então trata-se de uma PROM (*Programmable ROM*), e se o plano OU pode ser reprogramado trata-se de uma EPROM (*Erasable Programmable ROM*).

4.2 Realização de funções com decodificadores

Uma vez que um decodificador $n : 2^n$ realiza todos os produtos canônicos de n variáveis, qualquer função com n variáveis pode ser realizada com um decodificador $n : 2^n$ e uma porta OU (com um número de entradas maior ou igual ao número de 1s da função) ou uma porta NÃO-OU (com um número de entradas maior ou igual ao número de 0s da função).

O custo da realização de uma função com decodifcadores é, em termos de portas lógicas, (muito provavelmente) maior que o da realização SOP minimal. No entanto, a simplicidade de projeto torna-o atraente. Além disso, quando múltiplas funções precisam ser realizadas, menor tende a ser a diferença dos custos entre a realização SOP minimal e a realização com decodificadores.

Exemplo: A função $f(a,b,c) = \sum m(0,1,4,6,7) = \prod M(2,3,5)$ pode ser realizada usando decodificadores conforme ilustrado na figura 9. No caso da realização com porta NÃO-OU, observe que $f(a,b,c) = \prod M(2,3,5) = \overline{\overline{M_2 \cdot M_3 \cdot M_5}} = \overline{\overline{M_2} + \overline{M_3} + \overline{M_5}} = \overline{m_2 + m_3 + m_5}$.

4.3 Realização multi-níveis de decodificadores*

Vimos que decodificadores possuem n entradas e 2^n saídas e que sua realização trivial utiliza 2^n portas E, com n entradas cada. Para contornar o problema de *fan-in* (número máximo de entradas possíveis em

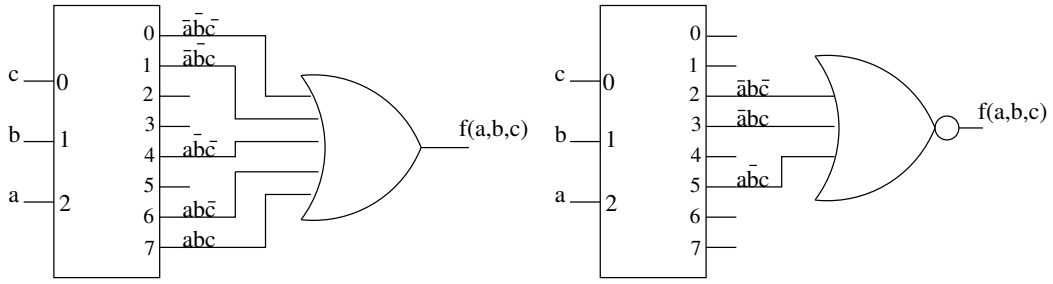


Figura 9: Realização de $f(a,b,c) = \sum m(0,1,4,6,7)$ com decodificador $3:8$ e uma porta OU (esquerda) ou uma porta NÃO-OU (direita).

uma porta lógica), decodificadores com grande número de entradas podem ser realizados por circuitos com múltiplos níveis. A figura 10 mostra como pode ser realizado um decodificador $12 : 2^{12}$ em três níveis.

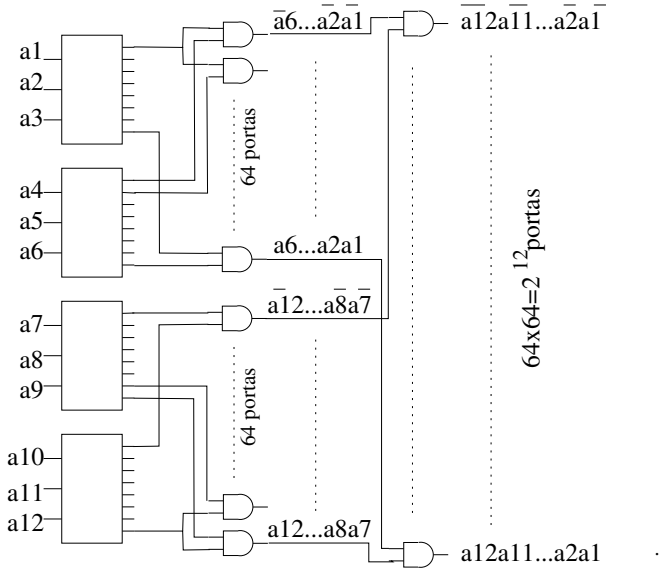


Figura 10: Realização três-níveis de um decodificador $12 : 2^{12}$.

No primeiro nível são usados 4 decodificadores $3:8$. No segundo nível, 64 portas E de duas entradas são usadas para combinar cada uma das 8 saídas do primeiro decodificador com cada uma das 8 saídas do segundo decodificador. A mesma coisa para as saídas do terceiro com as do quarto decodificador. Cada uma das saídas das primeiras 64 portas E do segundo nível são combinadas com cada uma das saídas das últimas 64 portas E no mesmo nível, resultando em um total de $64 \times 64 = 2^{12}$ portas E no terceiro nível. As saídas dessas 2^{12} portas E correspondem aos produtos canônicos de 12 variáveis.

A solução acima utiliza portas E com três entradas no primeiro nível e portas E com duas entradas nos demais níveis. Se o circuito fosse realizado em apenas um nível, as portas E teriam 12 entradas.

Em uma outra possível realização, poderíamos substituir as 128 portas E de duas entradas no

segundo nível acima por 2^{12} portas E de quatro entradas e eliminar as portas do terceiro nível. Isto aparentemente reduziria o número total de portas, mas uma vez que 2^{12} domina de longe 128 e uma vez que as portas agora teriam quatro entradas em vez de duas, não se pode dizer que há economia no custo total.

Um outro problema devido às limitações tecnológicas é o conhecido por *fan-out* (número máximo de portas que podem ser alimentadas por uma saída de uma porta lógica). No caso da realização três-níveis do decodificador $12 : 2^{12}$ visto acima, as saídas das portas no segundo nível alimentam 64 portas no terceiro nível.

Para contornar o *fan-out*, uma possível solução são as realizações em estruturas de árvore. A figura 11 mostra a realização de um decodificador $3 : 8$ em uma estrutura de árvore. Em vez de termos todas as variáveis alimentando portas no primeiro nível, temos variáveis que alimentam portas nos outros níveis. Usando este esquema, pode-se reduzir o número de portas no próximo nível que precisam ser alimentadas pela saída de uma porta no nível anterior. Mesmo assim, o problema de *fan-out* não é totalmente eliminado pois as variáveis introduzidas nos níveis posteriores do circuito precisam alimentar muitas portas. No entanto, é mais fácil controlar o sinal de algumas poucas entradas (variáveis) para que eles sejam capazes de alimentar um maior número de portas do que fazer o mesmo com as saídas das portas lógicas. Esta solução possui um maior número de níveis e um maior número de portas lógicas do que o esquema mostrado na figura 10, mas para decodificadores de muitas entradas pode ser a única solução.

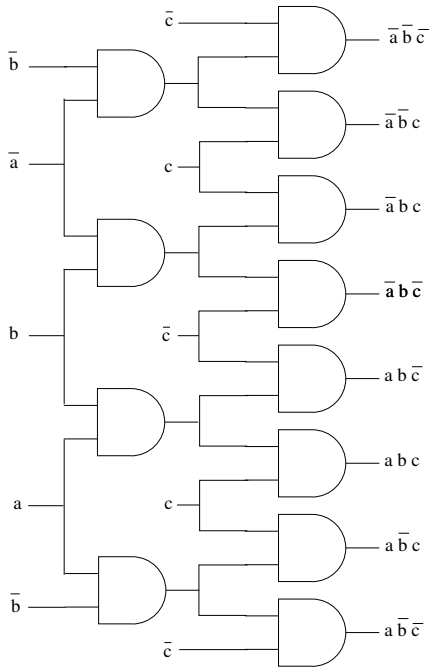


Figura 11: Decodificador em estrutura de árvore.

Na prática, as realizações de decodificadores para um número grande de entradas é baseada em uma combinação das estruturas da figura 10 e da figura 11.

5 Codificadores

Codificadores são circuitos combinacionais que são o inverso de decodificadores. Um codificador de n entradas deve possuir s saídas satisfazendo

$$2^s \geq n \quad \text{ou } s \geq \log_2 n$$

Usualmente deve-se ter apenas uma entrada ativa e a saída será o código binário correspondente à entrada. Isto é, se a i-ésima entrada estiver ativa, a saída será o código binário de i. A figura 12 mostra o esquema de um codificador de n entradas.

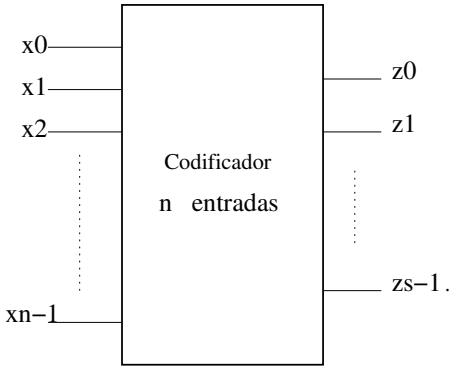


Figura 12: Esquema de um codificador.

Embora usualmente os codificadores sejam definidos como um circuito no qual apenas uma entrada encontra-se ativa, é possível termos codificadores com propósitos específicos que, por exemplo, para certos tipos de combinação de entradas gera um dado código de saída e para outras combinações de entradas gera outro código de saída.

5.1 Teclado

Decodificadores e codificadores podem ser usados, por exemplo, em teclados. Suponha por exemplo que um teclado simplificado possui 70 teclas. Em vez de se ter 70 fios conectando cada uma das teclas a um gerador de código ASCII, podemos ter um esquema como o ilustrado na figura 13.

O cruzamento das saídas do decodificador 3-8 com as entradas do codificador 16×4 corresponde a uma tecla. Quando houver sinal na linha de saída correspondente à tecla pressionada, o sinal entrará no codificador. A saída do codificador indica em qual das 14 colunas está a tecla pressionada, enquanto as linhas que ligam a saída do decodificador ao gerador de código ASCII indicam em qual das 5 linhas a tecla está. Com essas informações, o gerador de código ASCII sabe qual tecla foi pressionada e pode gerar o código correspondente à tecla. O contador à esquerda da figura alimenta as entradas do decodificador (varia ciclicamente de 0 a 7), tendo o efeito de gerar saída em uma das 5 linhas, ciclicamente. Obviamente há várias questões que precisam ser consideradas tais como garantir que o contador realize um ciclo completo durante o período de tempo em que uma tecla está pressionada (para "não perder" a tecla

pressionada), mas também não mais que um ciclo (para não produzir o efeito de duas pressões), ou então tratar as combinações de teclas que usualmente são pressionadas simultaneamente (como SHIFT+outra, CTRL+outra). Essas questões não são consideradas no modelo simplificado do teclado.

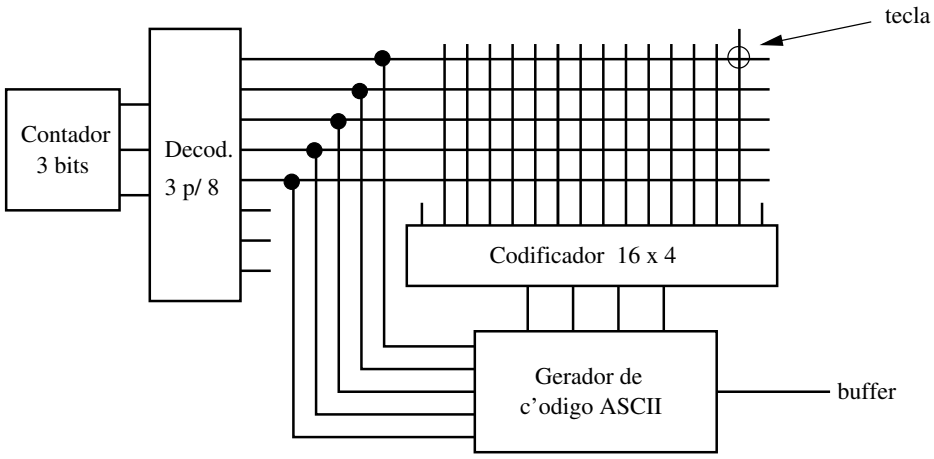


Figura 13: Esquema de um teclado. O decodificador identifica a linha e o codificador a coluna da tecla pressionada. É esperado que a cada tecla pressionada, o contador na entrada do decodificador execute um ciclo e, portanto, ative cada uma das 8 saídas do decodificador uma única vez.