3 Circuitos Combinacionais

Os circuitos lógicos dos sistemas digitais podem ser de dois tipos: circuitos combinacionais ou circuitos seqüenciais.

Um circuito combinacional é constituído por um conjunto de portas lógicas as quais determinam os valores das saídas diretamente a partir dos valores atuais das entradas. Pode-se dizer que um circuito combinacional realiza uma operação de processamento de informação a qual pode ser especificada por meio de um conjunto de equações Booleanas. No caso, cada combinação de valores de entrada pode ser vista como uma informação diferente e cada conjunto de valores de saída representa o resultado da operação. A figura 3.1 mostra o diagrama de blocos genérico de um circuito combinacional.

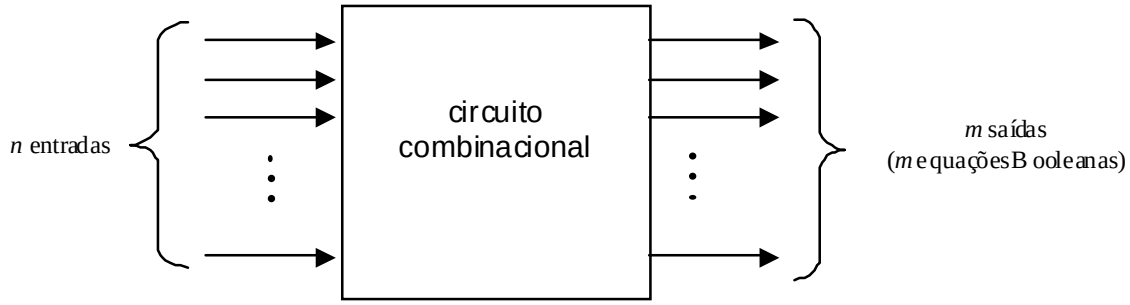


Figura 3.1 - Diagrama genérico de um circuito combinacional.

Um circuito seqüencial, por sua vez, emprega elementos de armazenamento denominados latches e flip-flops, além de portas lógicas. Os valores das saídas do circuito dependem dos valores das entradas e dos estados dos latches ou flip-flops utilizados. Como os estados dos latches e flip-flops é função dos valores anteriores das entradas, diz-se que as saídas de um circuito seqüencial dependem dos valores das entradas e do histórico do próprio circuito. Logo, o comportamento de um circuito seqüencial é especificado pela seqüência temporal das entradas e de seus estados internos. A figura 3.2 esboça um diagrama de blocos genérico para circuitos seqüenciais conhecido como modelo de Mealy. Circuitos seqüenciais serão objeto de estudo do próximo capítulo.

3.1 Procedimento para a Análise de um Circuito Combinacional

O objetivo da análise de um circuito combinacional é determinar seu comportamento. Então, dado o diagrama de um circuito, deseja-se encontrar as equações que descrevem suas saídas. Uma vez encontradas tais equações, pode-se obter a tabela verdade, caso esta seja necessária. É importante certificar-se que o circuito é combinacional e não seqüencial. Um modo prático é verificar se existe algum caminho (ou ligação) entre saída e entrada do circuito. Caso não exista, o circuito é combinacional.

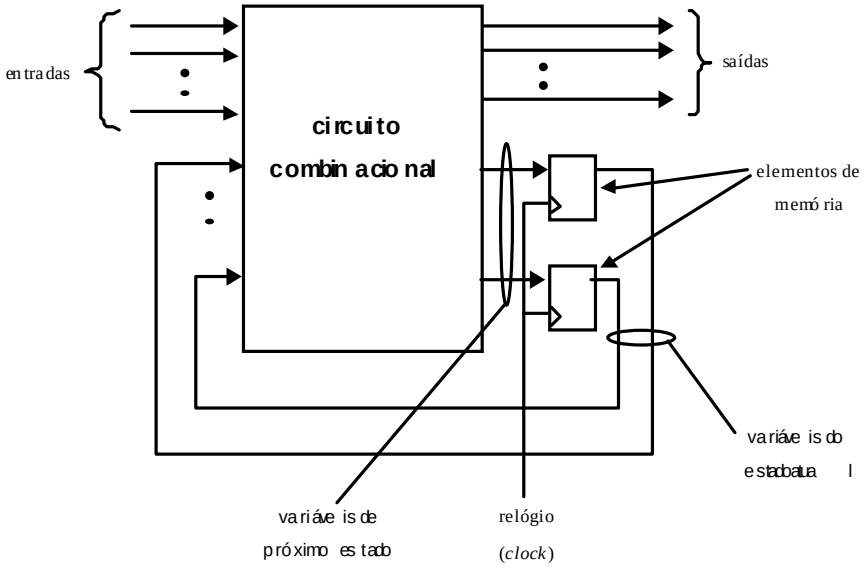


Figura 3.2 - Diagrama genérico de um circuito seqüencial segundo o modelo de Mealy.

O procedimento básico para se determinarem as equações que descrevem as saídas de um circuito combinacional é o seguinte:

1. dar um nome para as variáveis associadas a cada saída de cada porta do circuito, exceto aquelas saídas que já possuem nome (como por exemplo, as saídas do circuito);
2. a partir da esquerda, e seguindo a ordem de precedência determinada pelas ligações, determinar as equações associadas a cada variável, até que as equações de todas as saídas tenham sido encontradas.

Uma vez determinadas as equações das saídas, a montagem da tabela verdade será direta, havendo uma coluna para cada saída.

Exemplo 3.1: determinar as equações das saídas F1 e F2 do circuito que segue.

Vamos chamar as variáveis associadas às saídas das portas de T1, T2, T3 etc. Há somente duas portas cujas saídas já tem nome, que são justamente as saídas do circuito: F1 e F2. Listando as equações para essas variáveis, segue:

T1=

T2=

T3=

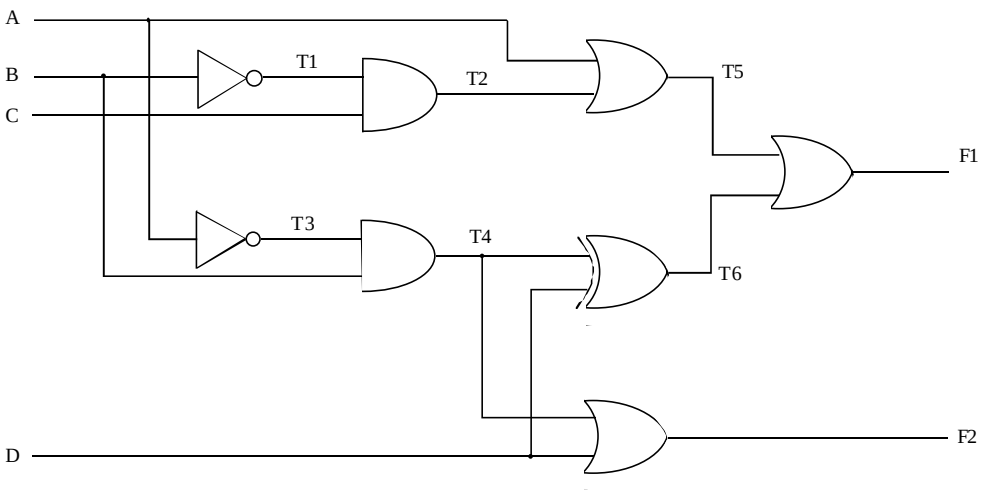
T4=

F2=

T5=

T6=

F1=



3.2 Procedimento para o Projeto de um Circuito Combinacional

O projeto de um circuito combinacional inicia na especificação do problema e culmina no diagrama do circuito (ou no conjunto de equações que o descrevem). Um procedimento genérico para o projeto envolve os seguintes passos:

1. escolher um símbolo para cada variável de entrada e para cada variável de saída;
2. a partir da especificação do problema, determinar a tabela verdade (caso ela já não faça parte da especificação do problema);
3. obter as equações simplificadas;
4. mapear o circuito para a biblioteca de portas disponível (se for o caso);
5. desenhar o circuito final

Exemplo 3.1: projetar um circuito que recebe um inteiro binário de 3 bits e determina se este número é menor ou igual a 3. Usar somente portas NAND de duas entradas e inversores.

Chamemos a função que representa a saída do circuito de "menor". Denominando de $A_2A_1A_0$ aos 3 bits que compõem o número, podemos montar o mapa de Karnaugh para essa função. Os valores para os "menor" deve valer 1 são $A_2A_1A_0$={(0,0,0);(0,0,1);(0,1,0);(0,1,1)}. Para os demais combinações de entradas, "menor" deverá ser igual a 0.

menor

A_2 \ A_1A_0	00	01	11	10
0				
1				

Equação em soma de produtos, simplificada.

menor=

Circuito para a equação em soma de produtos, simplificada:

Circuito mapeado para portas NAND de duas entradas e inversores.

3.3 Circuitos Combinacionais de Interconexão

Os circuitos combinacionais são os responsáveis pelas operações lógicas e aritméticas dentro dentro de um sistema digital (vale lembrar que um computador é um sistema digital). Além das operações lógicas e aritméticas como adição, subtração complementação, existem ainda outras funções necessárias para a realização de conexões entre os diversos operadores. Dentre essas funções estão a multiplexação e a decodificação. Os elementos que realizam essas últimas operações são denominados multiplexadores e decodificadores, respectivamente, e são também circuitos combinacionais. A seguir, veremos como tais circuitos são constituídos.

3.3.1 Decodificadores

Um decodificador é um circuito combinacional usado para **ativar** ou **habilitar** um (e somente um) dentre m componentes. É assumido que cada componente possui um índice entre 0 e m-1, representado por um endereço em binário.

Um decodificador n : m (lê-se n por m) possui n entradas e m saídas, com $m \leq 2^n$.

No caso de um decodificador 3:8, serão 8 saídas, onde cada saída pode ser encarada como um endereço diferente. Para ativar uma dentre 8 saídas são necessárias 3 variáveis de entrada (daí 3:8). Cada combinação das variáveis de entrada seleciona um e somente uma dentre as 8 saídas, de modo que cada saída somente será selecionada por uma das 8 combinações. Desta forma, é natural que se associe a cada saída um índice decimal que represente a combinação de entradas responsável pela sua ativação. Assumindo-se ativação em lógica direta, isto é, que uma saída está ativada se ela vale 1, então a tabela verdade para um decodificador 3:8 será:

endereço	Entradas (sinais de controle)			saídas							
	A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Note que cada saída só vale 1 para uma determinada combinação das variáveis de entrada. Além disso, cada combinação de entrada só ativa uma dentre todas as 8 saídas.

O circuito de um decodificador 3:8 terá, portanto, 8 saídas, sendo cada saída um dentre os 8 mintermos possíveis para uma função Booleana de 3 variáveis. A figura 3.3a mostra o

símbolo para o decodificador 3:8, enquanto a figura 3.3b mostra um circuito possível para o mesmo decodificador, utilizando portas E de 3 entradas e inversores.

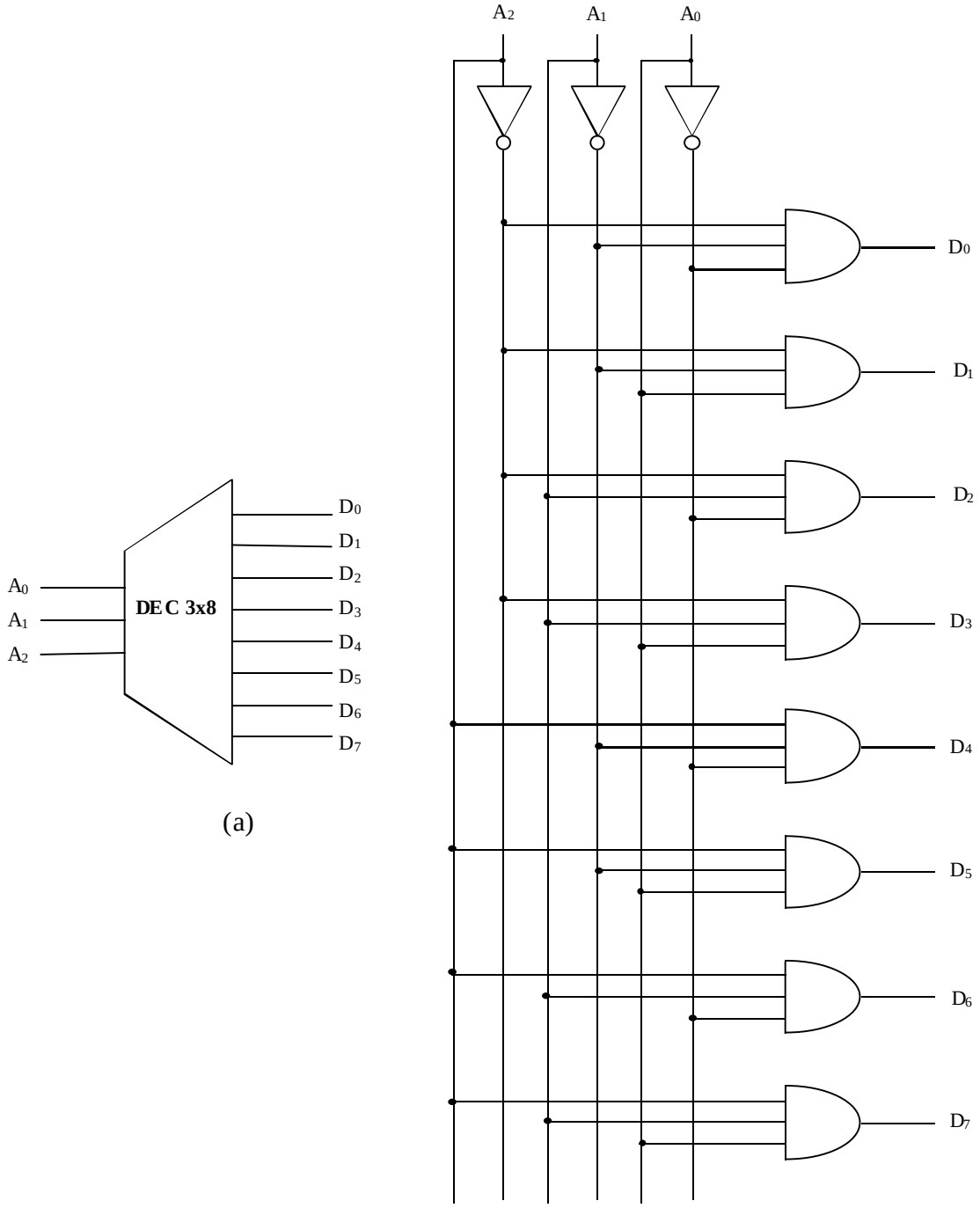


Figura 3.3- Símbolo (a) e diagrama (b) de um decodificador 3:8.

Um decodificador pode possuir uma entrada de habilitação (*enable*, em inglês). Esta entrada tem a função de habilitar ou desabilitar seu funcionamento. Assim, se esta entrada valer 0, nenhuma saída estará ativada, independente dos valores das demais entradas. Por outro lado, se a entrada de habilitação valer 1, o decodificador estará ativando uma das saídas.

Neste exemplo, foi considerado que a habilitação do decodificador se dá com lógica direta, isto é, quando a entrada de habilitação valer 1. A lógica de habilitação poderia ser negada, ou seja, habilita se a entrada de habilitação valer 0 e não habilita, caso contrário.

A tabela verdade de um decodificador 2:4 com ativação e habilitação em lógica direta é a seguinte:

	endereço	entradas (sinais de controle)			saídas			
		E	A_1	A_0	D_0	D_1	D_2	D_3
saídas desabilitadas	-	0	0	0	0	0	0	0
	-	0	0	1	0	0	0	0
	-	0	1	0	0	0	0	0
	-	0	1	1	0	0	0	0
	0	1	0	0	1	0	0	0
	1	1	0	1	0	1	0	0
	2	1	1	0	0	0	1	0
	3	1	1	1	0	0	0	1

Como pode-se verificar, nas primeiras 4 linhas o sinal de habilitação (E) vale zero, o que desativa as saídas, independentemente dos valores das demais entradas (A_1 e A_0). Desta forma, podemos re-escrever esta tabela de maneira mais compacta, indicando numa única linha que, quando E=0, os valores das entradas A_1 e A_0 não interessam (=*don't cares* de entrada):

	endereço	entradas (sinais de controle)			saídas			
		E	A_1	A_0	D_0	D_1	D_2	D_3
saídas desabilitadas →	-	0	X	X	0	0	0	0
	0	1	0	0	1	0	0	0
	1	1	0	1	0	1	0	0
	2	1	1	0	0	0	1	0
	3	1	1	1	0	0	0	1

A figura 3.4a mostra o símbolo para esse decodificador e a figura 3.4b mostra uma possível implementação (circuito lógico).

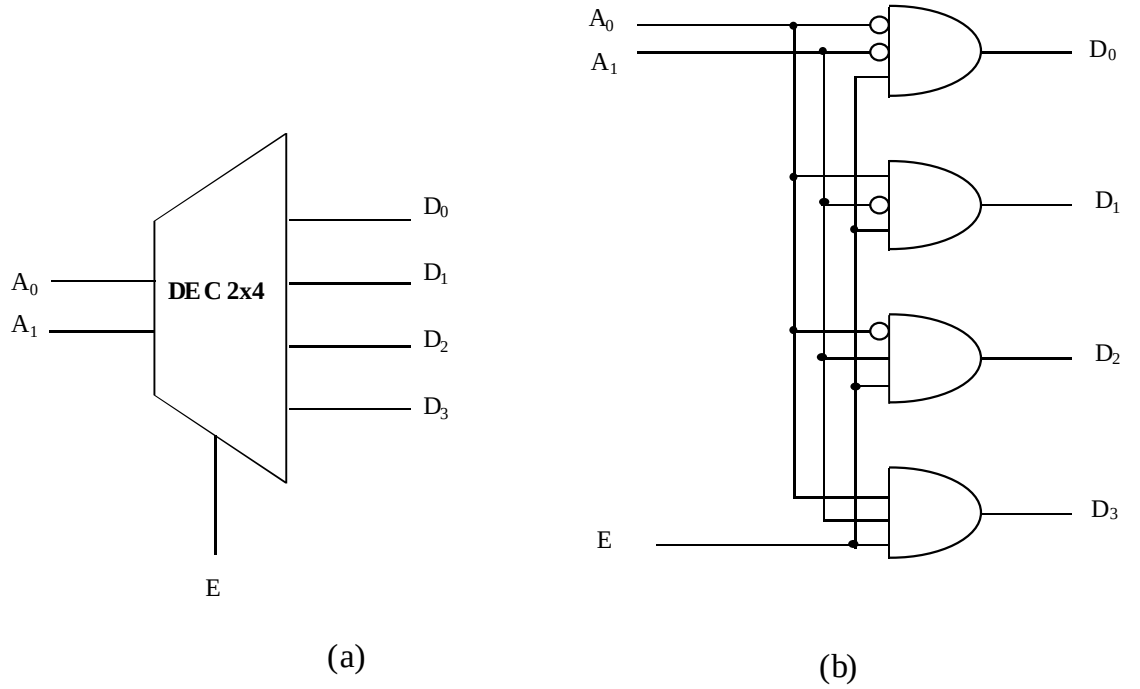


Figura 3.4: símbolo (a) e diagrama (b) de um decodificador 2x4 com entrada de habilitação.

3.3.2 Seletores

Um seletor (também conhecido como multiplexador) é um circuito combinacional usado para **selecionar** uma dentre um conjunto de m fontes de informação disponíveis. Um seletor que possui n entradas para realizar a seleção é capaz de selecionar uma dentre 2^n entradas. Logo, m deve ser menor ou igual a 2^n.

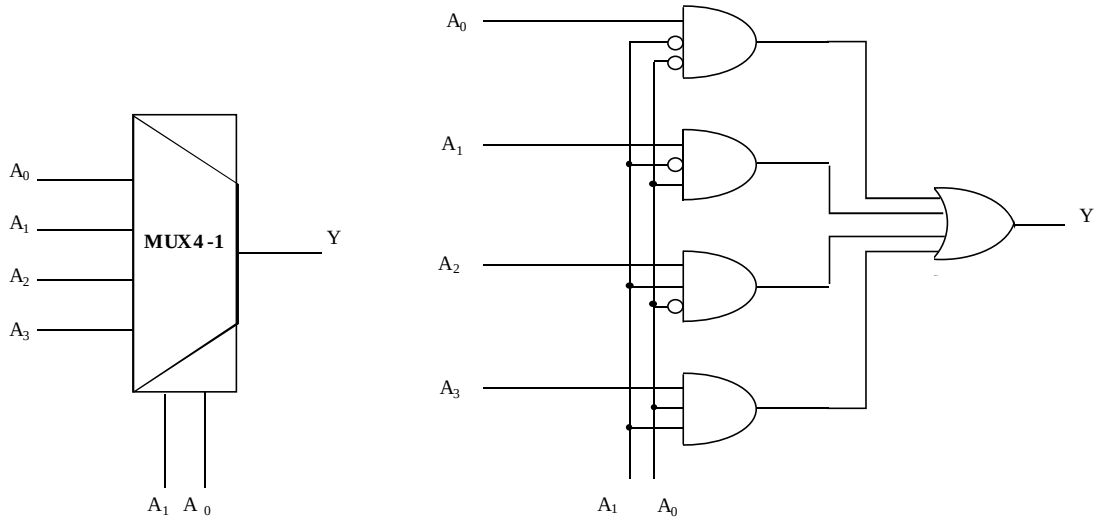
Dado o conjunto de entradas A_0, A_1, A_2 e A_3, e as variáveis de seleção S_0 e S_1, a tabela verdade para um seletor 4-1 será:

endereço	variáveis de seleção		saída
	S_1	S_0	Y
0	0	0	A_0
1	0	1	A_1
2	1	0	A_2
3	1	1	A_3

Pela tabela verdade acima percebe-se que a saída Y pode ser implementada por um circuito em soma de produtos, onde em cada produto estarão presentes as variáveis S_0 e S_1 e uma dentre as variáveis de entrada A_0, A_1, A_2 e A_3:

$$Y = \overline{S_0} \cdot \overline{S_1} \cdot A_0 + \overline{S_0} \cdot S_1 \cdot A_1 + S_0 \cdot \overline{S_1} \cdot A_2 + S_0 \cdot S_1 \cdot A_3$$

A figura 3.5a mostra o símbolo para tal seletor e a figura 3.5b mostra um possível circuito em soma de produtos.



(a) (b)

Figura 3.5 - Símbolo (a) e diagrama (b) de um seletor 4-1.

3.4 Circuitos Aritméticos

Um circuito combinacional aritmético implementa operações aritméticas como adição, subtração, multiplicação e divisão com números binários. A operação aritmética mais simples é a adição de dois dígitos binários, que consiste de quatro possíveis operações elementares: 0+0=0, 0+1=1, 1+0=1 e 1+1=10. As três primeiras operações produzem um dígito de soma. Entretanto, quando ambos os operandos são iguais a 1, são necessários dois dígitos para expressar seu resultado. Neste caso, o transporte (vai-um ou *carry*, em inglês) é somado ao próximo par mais significativo de bits. Um circuito combinacional que implementa a adição de dois bits é chamado **meio-somador** (***half adder,*** em inglês). Um circuito que implementa a adição de três bits (dois bits significativos e um carry) é chamado de **somador completo** (***full adder,*** em inglês). Estes nomes decorrem do fato de que com dois meio-somadores pode-se implementar um somador completo. O somador completo é um circuito aritmético básico a partir do qual todos os outros circuitos aritméticos são construídos.

3.4.1 Meio somador (*half adder*) e somador completo (*full adder*)

A operação aritmética mais simples é a adição de dois dígitos binários (bits), a qual pode ser vista como a adição de dois números binários de um bit cada. Considerando-se todas as 4 combinações de valores que podem ocorrer, os resultados possíveis dessa adição são:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Repare que no último caso acima, o resultado da adição é o valor 2, que em binário necessita de dois dígitos para ser representado (10). Ora, um circuito aritmético para realizar a adição de dois bits deve operar corretamente para qualquer combinação de valores de entrada. Isso significa que o circuito para a adição de dois bits deve possuir duas entradas e duas saídas, conforme ilustrado na figura 3.6.

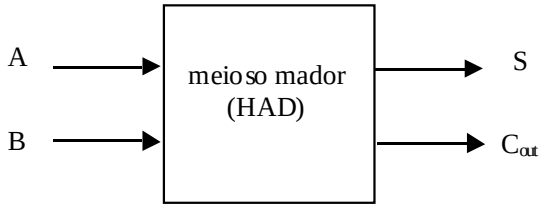


Figura 3.6 - Esquema das entradas e saídas de um meio somador (*half adder* ou HAD).

Denomina-se **meia-soma** a operação de adição de dois bits. O circuito mostrado na figura 3.6 é denominado **meio somador** (*half adder*, em inglês). As duas entradas, A e B, representam os dois bits a serem adicionados. A saída S representa o dígito menos significativo do resultado, enquanto que a saída C_{out} representa o dígito mais significativo do resultado, o qual também é conhecido por **transporte de saída** (*carry out*, em inglês), uma vez que ele assume valor 1 somente quando o resultado da soma de A e B não pode ser representado num único dígito.

A fim de se projetar o circuito do meio somador, devemos montar uma tabela verdade para as saídas S e Cout utilizando-se os valores que resultam da adição de dois dígitos binários, como segue:

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Note que a saída S nada mais é do que o **XOR** entre A e B ($S = \overline{A}.B + A.\overline{B} = A \oplus B$). Já a saída Cout é o **E** entre A e B ($C_{out} = A \cdot B$). Então, um circuito para o meio somador usa apenas uma porta XOR de duas entradas e uma porta E de duas entradas, conforme mostrado na figura 3.7.

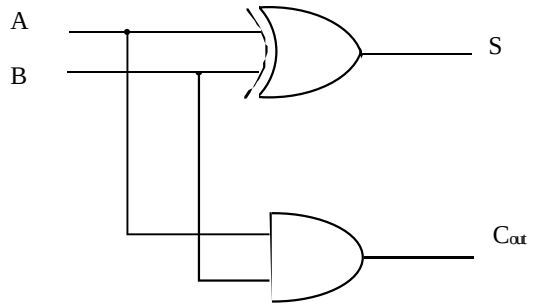


Figura 3.7 - Circuito para o meio somador (*half adder* ou HAD).

Entretanto, quando ao somarmos dois números binários que possuem mais de um dígito cada ocorrer transporte diferente de zero para a soma de um par de dígitos intermediários, a soma do par seguinte deverá considerar esse transporte proveniente do par anterior, conforme ilustra o exemplo a seguir (figura 3.8).

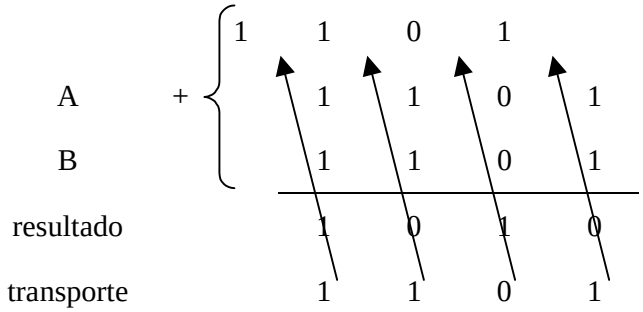


Figura 3.8 - Exemplo de adição de dois números binários com mais de um dígito.

O exemplo mostrado na figura 3.8 ilustra bem o fato de, para cada posição exceto a menos significativa, o resultado é obtido mediante a adição de três bits: um pertencente ao número A, um pertencente ao número B e um terceiro que é o transporte proveniente do resultado da adição entre os bits da posição anterior.

O circuito capaz de realizar a soma de três bits (A, B e C_{in}), gerando o resultado em dois bits (S e C_{out}) é denominado **somador completo** (*full adder*, em inglês). Apesar da entrada C_{in} normalmente receber o transporte proveniente da soma imediantamente anterior (*carry in*, em inglês), a rigor as três entradas são absolutamente equivalentes sob o ponto de vista funcional. A tabela verdade para a soma completa é mostrada a seguir, juntamente com o mapa de Karnaugh e as equações mínimas resultantes para S e C_{out}. A figura 3.9 mostra um circuito para o somador completo.

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S

A \ BC_{in}	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

Conforme pode-se ver pelo mapa de Karnaugh acima, a expressão mínima em soma de produtos para S contém todos os mintermos da função:

$$S = \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C_{in}} + A \cdot \overline{B} \cdot \overline{C_{in}} + A \cdot B \cdot C_{in}$$

O circuito que implementa a saída S do somador completo pode ser derivado a partir da equação em soma de produtos acima. No entanto, pode-se ainda manipular tal equação conforme segue:

$$S = \overline{A} \cdot (\overline{B} \cdot C_{in} + B \cdot \overline{C_{in}}) + A \cdot (\overline{B} \cdot \overline{C_{in}} + B \cdot C_{in}) = \overline{A} \cdot (B \oplus C_{in}) + A \cdot (\overline{B \oplus C_{in}}) = A \oplus B \oplus C_{in}$$

Logo, o circuito para a saída S do somador completo pode também ser representado com duas portas XOR, conforme mostra a figura 3.9.

C_{out}

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

A saída C_{out} tem como expressão mínima em soma de produtos:

$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$$

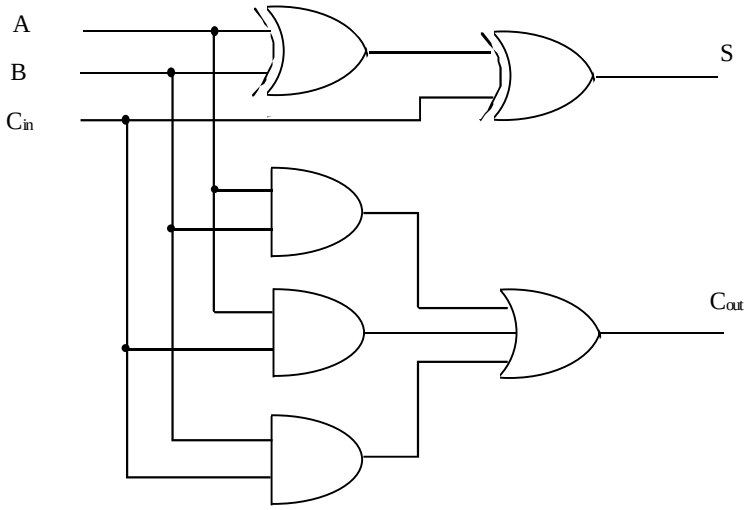


Figura 3.9 - Circuito para o somador completo (*full adder* ou FAD).

3.4.2 O somador paralelo tipo *ripple carry*

Utilizando-se n somadores completos, pode-se realizar um somador capaz de operar dois números binários de n bits. Particularmente, o dígito de ordem i do resultado, S_i, será obtido pela adição de A_i, B_i e C_i, onde C_i é o transporte proveniente do dígito anterior. O somador de índice i recebe como entradas A_i, B_i e C_i, gerando a soma S_i e o valor de transporte C_{i+1}, o qual será entrada para o somador completo do dígito seguinte (i+1). A figura 3.10 mostra uma representação de bloco possível para o somador completo da figura 3.9. A figura 3.11 mostra um circuito somador paralelo para números binários com 4 bits.

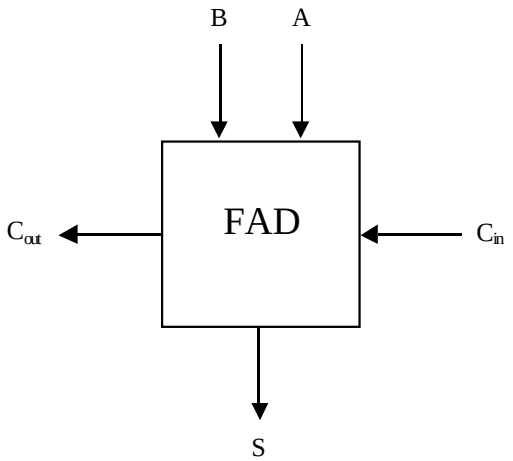


Figura 3.10 - Representação de bloco para o somador completo (*full adder* ou FAD).

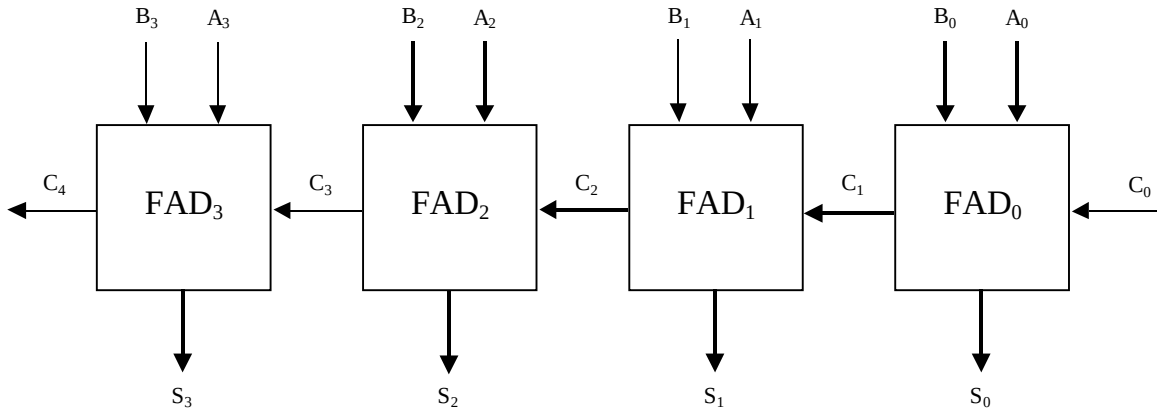


Figura 3.11- Somador paralelo de 4 bits.

Repare que o somador completo de índice zero, FAD_0, também possui uma entrada C_{in}, aqui denominada C_0. Como a priori não existe um valor de transporte a ser somado aos dígitos menos significativos, A_0 e B_0, esta entrada deverá estar constantemente ligada a zero. Já a saída de transporte C_{out} do dígito mais significativo, C_4 no caso do somador de 4 bits, serve para indicar se o resultado da adição entre A e B pode ser representado em 4 bits. Caso o resultado não pode ser representado em 4 bits, C_4 irá exibir o valor 1.

Repare também que, uma vez que um novo par de valores A e B é fornecido ao circuito somador, as últimas 2 saídas que se estabilizam são S_3 e C_4 , uma vez que estas dependem de C3, que por sua vez depende da estabilização de C2 e assim por diante. Desta forma, pode-se aproximar o atraso deste somador como sendo proporcional ao número de estágios (=número de somadores completos cascateados). Com efeito, a propagação do transporte ou *carry* ao longo da cadeia de somadores é o ponto fraco deste tipo de somador. Existem outros tipos de somadores capazes de operar mais rapidamente, mas que por razões de tempo não serão estudados nesta disciplina.

A construção de um somador para operar dois números binários de n bits requer o uso de n somadores completos, conectados segundo a mesma topologia mostrada na figura 3.11.

É importante ressaltar que tal somador pode operar dois números inteiros quaisquer, positivos ou negativos, desde que ambos estejam representados em complemento de 2.

3.4.3 O somador/subtrator

A subtração de dois números inteiros em binário pode ser feita utilizando-se a seguinte fórmula:

$$A - B = A + \overline{B} + 1$$

onde todas as operações são aritméticas, exceto $\overline{B}$, que representa a complementação de B, bit a bit.

A figura 3.12 mostra um circuito somador/subtrator de 4 bits. Esse circuito é originado do somador paralelo de 4 bits, porém com a adição de portas xor nas entradas associadas a B, de modo a permitir a negação individual de cada bit de B. A tabela que segue mostra o funcionamento deste circuito, em função dos sinais de controle sel_1 e sel_2. Note que sel_1 coincide com C_0.

A exemplo do que ocorre com o somador paralelo apresentado na seção anterior, também o somador/subtrator pode operar dois números inteiros quaisquer, positivos ou negativos, desde que tais números estejam representados em complemento de dois. Caso os dois números a serem operados estivessem representados em sinal-magnitude, por exemplo, seria necessário existir um circuito para testar o sinal de cada número e comparar as magnitudes, para só então realizar a soma ou a subtração. Como isso representaria a necessidade de um hardware mais complexo, e possivelmente mais caro e mais lento, a representação em complemento de dois é dominantemente utilizada nos computadores atuais.

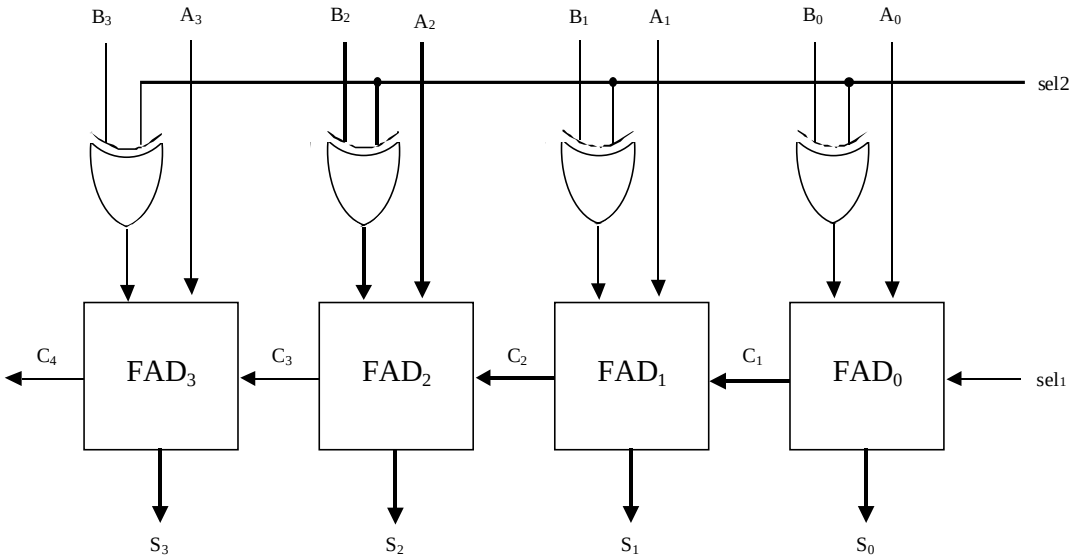


Figura 3.12 - Somador/subtrator de 4 bits.

Operações possíveis para o somador/subtrator da figura 3.12.

sel_2	**sel_1**	**operação**	**descrição**
0	0	$S = A + B + 0$	adiciona A e B ($S = A + B$)
0	1	$S = A + B + 1$	adiciona A e B incrementado ($S = A + B + 1$)
1	0	$S = A + \overline{B} + 0$	subtrai B decrementado de A ($S = A - B - 1$)
1	1	$S = A + \overline{B} + 1$	subtrai B de A ($S = A - B$)

3.4.4 O multiplicador

A multiplicação de números binários é realizada da mesma maneira como a de números decimais. O multiplicando é multiplicado por cada bit do multiplicador, começando do bit menos significativo. Cada uma destas multiplicações formam um produto parcial. Os sucessivos produtos parciais são deslocados uma posição para a esquerda. O produto final é obtido a partir da soma dos produtos parciais.

Para entender como um multiplicador binário pode ser implementado com um circuito combinacional, considere a multiplicação de dois números de dois bits mostrada na figura abaixo:

			B_1	B_0	← multiplicando
×			A_1	A_0	← multiplicador
	C_3	C_2			← transportes
+			$A_0\,B_1$	$A_0\,B_0$	
		$A_1\,B_1$	$A_1\,B_0$	-	
	M_3	M_2	M_1	M_0	← resultado

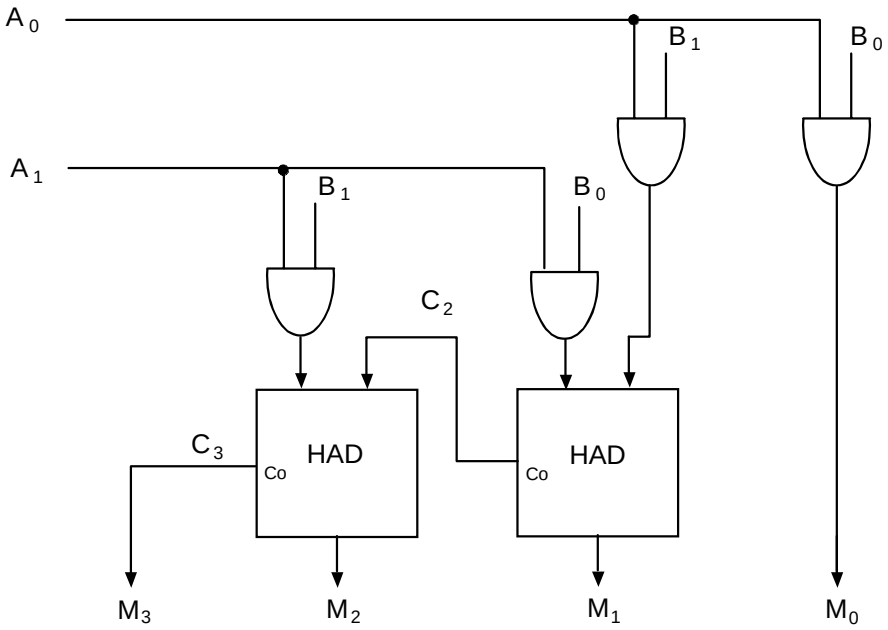


Figura 3.13 - Multiplicador de 2 bits.

Os bits do multiplicando são B_1 e B_0, os bits do multiplicador são A_1 e A_0 e o produto é $M_3M_2M_1M_0$. O primeiro produto parcial é formado pela multiplicação de B_1B_0 por A_0. A multiplicação de dois bits, tais como A_0 e B_0, produz um 1 se ambos os bits são 1, do contrário ela produz um 0. Isto é idêntico à operação **E**. Assim, o produto parcial pode ser implementado com **portas E** como mostrado no circuito da figura 3.13. O segundo produto parcial é formado pela multiplicação de B_1B_0 por A_1 e é deslocado uma posição para a esquerda. Os dois produtos parciais são somados com dois circuitos meio-somadores. Usualmente tem-se mais bits nos produtos parciais, fazendo-se necessário o uso de somadores completos para produzir a soma dos produtos parciais.

Um circuito multiplicador binário combinacional com mais bits pode ser construído de maneira semelhante. Um bit do multiplicador é operado por um **E** com cada bit do multiplicando em tantos níveis quanto existam bits no multiplicador. A saída binária em cada nível de portas **E** é somada em paralelo com o produto parcial do nível anterior para formar um novo produto parcial. O último nível produz o resultado. Para *j* bits no multiplicador e *k*

bits no multiplicando, serão necessários $j \times k$ portas **E** e (j-1) somadores de k bits para gerar um produto de $j+k$ bits.

Exercícios

Exercício 3.1 - Projetar um decodificador 3:8 com ativação em lógica negada (isto é, para cada saída Di, se Di=0, Di está ativada, se Di=1, a Di está desativada).

Exercício 3.2 - Projetar um decodificador 2:4 com entrada de habilitação. Tanto a habilitação como a ativação das saídas deve se dar em lógica negada.

Exercício 3.3 - Reprojete o decodificador do exercício anterior utilizando somente portas NAND de 2 entradas.

Exercício 3.4 - Projete um decodificador 3:8 utilizando um inversor e 2 decodificadores 2:4 com entrada de habilitação.

Exercício 3.5 - Projetar um seletor 8-1 a partir de seletor 4-1.

Exercício 3.6 - Projetar um seletor 4-1, onde cada entrada é composta por um conjunto de 2 bits. Usar o símbolo para representar cada seletor 4-1, ao invés de desenhar o circuito detalhado.

Exercício 3.7 - Desenhe o circuito lógico de um multiplicador de quatro bits.

Bibliografia Suplementar

[1] GAJSKI, Daniel D. **Principles of Digital Design**, New Jersey: Prentice Hall, 1997 (ISBN 0-13-301144-5)

[2] MANO, M. Morris; **Computer Engineering: Hardware Design.** New Jersey: Prentice Hall, 1988 (ISBN 0-13-162926-3)