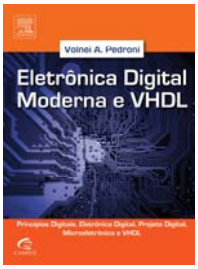


Eletrônica Digital Moderna e VHDL
Volnei A. Pedroni, Elsevier, 2010

Tradução (com revisão, atualização e ampliação) de
Digital Electronics and Design with VHDL
Elsevier / Morgan Kaufmann, USA, 2008

Soluções dos Exercícios Ímpares dos Capítulos 1-5

Capítulo 2: Representações Binárias

Exercício 2.1. Número de palavras de código

a) $C_8^3 = 8!/(3!5!) = 56$

b) $C_8^0 = 1$, $C_8^1 = 8$, $C_8^2 = 28$, $C_8^3 = 56$, $C_8^4 = 70$, $C_8^5 = 56$, $C_8^6 = 28$, $C_8^7 = 8$, $C_8^8 = 1$

O valor é máximo quando metade ($N/2$) dos bits são altos e metade são baixos. Se N for ímpar, o valor máximo será o mesmo com $N/2$ arredondado para cima ou para baixo.

Exercício 2.3. Conversão de binário para decimal #2

a) $1000\ 1001_b = 2^7 + 2^3 + 2^0 = 137_d$

b) $1000\ 1111\ 0000_b = 2^{11} + 2^7 + 2^6 + 2^5 + 2^4 = 2288_d$

c) $0010\ 1000\ 0000\ 0001_b = 2^{13} + 2^{11} + 2^0 = 10241_d$

Exercício 2.5. Conversão de binário para hexadecimal #2

a) $100\ 1110_b = 4E_h$

b) $0011\ 1111\ 0010_b = 3F2_h$

c) $1\ 1111\ 1010\ 0001\ 1001_b = 1FA19_h$

Exercício 2.7. Conversão de decimal para binário #2

a) 63 → 6, 64 → 7, 512 → 10, 2007 → 11, 99999 → 17

b) $63_d = 11\ 1111_b$, $64_d = 100\ 0000_b$, $512_d = 10\ 0000\ 0000_b$, $2007_d = 111\ 1101\ 0111_b$, $99.999_d = 1\ 1000\ 0110\ 1001\ 1111_b$

Exercício 2.9. Conversão de decimal para hexadecimal #2

$255_d = FF_h$, $256_d = 100_h$, $4096_d = 1000_h$, $12345_d = 3039_h$

Exercício 2.11. Conversão de hexadecimal para binário #2

$D_h = 1101_b$, $29C_h = 10\ 1001\ 1100_b$, $F000_h = 1111\ 0000\ 0000\ 0000_b$, $13FF_h = 1\ 0011\ 1111\ 1111_b$

Exercício 2.13. Conversão de hexadecimal para decimal #2

$AA_h = 170_d$, $990_h = 2448_d$, $7001_h = 28673_d$, $FF007_h = 1044487_d$

Exercício 2.15. Conversão de decimal para octal

$3_d = 3_{oct}$, $77_d = 115_{oct}$, $111_d = 157_{oct}$, $2222_d = 4256_{oct}$

Exercício 2.17. Conversão de decimal para BCD #2

$03_d = 0000\ 0011_{BCD}$

$65_d = 0110\ 0101_{BCD}$

$900_d = 1001\ 0000\ 0000_{BCD}$

$7890_d = 0111\ 1000\ 1001\ 0000_{BCD}$

Exercício 2.19. Código gray #1

11111 → 11110 → 11100 → 11101 → 11001 → 11000 → 11010 → 11011 → 10011 → 10010 → 10000 → 10001 → 10101 → 10100 → 10110 → 10111 → 00111 → 00110 → 00100 → 00101 → 00001 → 00000 → 00010 → 00011 → 01011 → 01010 → 01000 → 01001 → 01101 → 01100 → 01110 → 01111

Exercício 2.21. Código one-hot

a) 000001, 000010, 000100, 001000, 010000, 100000

b) A distância de Hamming entre quaisquer duas palavras é constante e igual a 2.

Exercício 2.23. Faixa decimal #1

Bits	Sem sinal	Com sinal
6	0 a $2^6 - 1$ = 0 a 63	-2^5 a $2^5 - 1$ = –32 a 31
1	0 a $2^{12} - 1$ = 0 a 4_095	-2^{11} a $2^{11} - 1$ = –2_048 a 2_047
24	0 a $2^{24} - 1$ = 0 a 16_777_215	-2^{23} a $2^{23} - 1$ = –8_388_608 to 8_388_607

Exercício 2.25. Conversão de decimal para sinal-magnitude

Decimal	Sinal-magnitude
+3	000 0011
–3	100 0011
+31	001 1111
–31	101 1111
+48	011 0000
–48	111 0000

Exercício 2.27. Conversão de decimal para complemento de um

Decimal	Complemento de um
+3	000 0011
–3	111 1100
+31	001 1111
–31	110 0000
+48	011 0000
–48	100 1111

Exercício 2.29. Conversão de decimal para complemento de dois #1

Decimal	Complemento de dois
+3	000 0011
–3	111 1101
+31	001 1111
–31	110 0001
+48	011 0000
–48	101 0000

Exercício 2.31. Conversão de complemento de dois para decimal #1

a) 010101 → 21

b) 101010 → –22

c) 0000 0001 → 1

d) 1000 0001 → –127

Exercício 2.33. Representação por ponto flutuante

a) $0{,}75 = 1/4 + 1/2 = 3/4 = 11_b \cdot 2^{-2} = 1{,}1_b \cdot 2^{-1}$. Logo:

$S = 0$, $F = 1000...00$, $E = e + 127 = 126 = 01111110$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...	2^{-22}	2^{-23}
0	0	1	1	1	1	1	1	0	1	0	0	0	…	0	0

b) $-2{,}625 = -(2 + 1/2 + 1/8) = -21/8 = -10101_b \cdot 2^{-3} = -1{,}0101_b \cdot 2^1$. Logo:

$S = 1$, $F = 010100...00$, $E = e + 127 = 128 = 10000000$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...	2^{-22}	2^{-23}
1	1	0	0	0	0	0	0	0	0	1	0	1	0 …	0	0

c) $37/32 = 37 \cdot 2^{-5} = 100101_2 \cdot 2^{-5} = 1{,}00101_2 \cdot 2^0$. Logo:

$S = 0$, $F = 001010...0$, $E = e + 127 = 127 = 01111111$

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...	2^{-22}	2^{-23}
0	0	1	1	1	1	1	1	1	0	0	1	0	1 0 …	0	0

Exercício 2.35. Conversão de inteiro para ponto flutuante #1

a) $0{,}1875 = 3/16 = 11_b \cdot 2^{-4} = 1{,}1_b \cdot 2^{-3}$. Logo:

$S = 0$, $F = 100…0$, $E = e + 127 = 124 = 01111100$

b) $-0{,}185$. Resolução idem acima, trocando apenas o valor de S por 1 devido ao sinal negativo. Portanto:

$S=1$, $F = 100...0$, $E = 01111100$

c) $1 = 1_b \cdot 2^0$. Logo:

$S = 0$, $F = 000...0$, $E =127 = 01111111$

d) $4{,}75 = 19/4 = 10011_b \cdot 2^{-2} = 1{,}0011_b \cdot 2^2$. Logo:

$S = 0$, $F = 00110...0$, $E = e + 127 = 129 = 10000001$

Exercício 2.37. Conversão de ponto flutuante para decimal #1

Valor decimal = $(-1)^S(1 + F)^{E-127}$

a) $(-1)^1(1 + 2^{-23}) \cdot 2^{31-127} = -(1+2^{-23}) \cdot 2^{-96}$ $(\approx -2^{-96})$

b) $(-1)^0(1 + 2^{-1} + 2^{-2}) \cdot 2^{143-127} = 114.688$

Exercício 2.39. Código ASCII #1

Hi! = 1001000 1101001 0100001

Exercício 2.41. Código ASCII #3

1010110 1001000 1000100 1001100 = VHDL

Exercício 2.43. Codificação Unicode UTF-8 #2

U_+0031, U_+0020, U_+1000, U_+0020, $U_+020000 = 1 + 1 + 3 + 1 + 4 = 10$

Exercício 2.45. Codificação Unicode UTF-16 #1

U_+00A1:

$00A1_h = 161_d$, o qual requer dois bytes para sua representação: byte1 = $00_h = 00000000_b$, byte2 = $A1_h = 10100001_b$.

U_+050C:

$050C_h = 1292_d$, o qual requer dois bytes para sua representação: byte1 = $05_h = 00000101_b$, byte2 = $0C_h = 00001100_b$.

U_+F333:

$F333_h = 62259_d$, , o qual também requer dois bytes: byte1 = $F3_h = 11110011_b$, byte2 = $33_h = 00110011_b$.

U_+1234A_h:

$1234A_h = 74570_d$. Como esse ponto está na faixa de 65536 a 1M, ele requer quatro bytes. Observando que $01234A_h$ = $0000\ 0001\ 0010\ 0011\ 0100\ 1010_{BCD}$, obtém-se *bbbbb* = 00001 e *aaaa aaaa aaaa aaaa* = 0010 0011 0100 1010. Então *c*=*b*–1 (truncado à esquerda) é *c*=0000. Portanto, byte1= 1101 1000 = $D8_h$, byte2 = 0000 1000 = 08_h, byte3 = 1101 1111 = DF_h, byte4 = 0100 1010 = $4A_h$.

Exercício 2.47. Codificação Unicode UTF-16 #3

U_+002F: byte1 = 00_h, byte2 = 00_h, byte3 = 00_h, byte4 = $2F_h$

U_+01FF: byte1 = 00_h, byte2 = 00_h, byte3 = 01_h, byte4 = FF_h

U_+11FF: byte1 = 00_h, byte2 = 00_h, byte3 = 11_h, byte4 = FF_h

U_+1111F: byte1 = $D8_h$, byte2 = 04_h, byte3 = $9D_h$, byte4 = $1F_h$

Exercício 2.49. Codificação Unicode UTF-32 #2

Cada um tem 4 bytes; portanto, 20 bytes (160 bits).

Capítulo 3: Aritmética Binária

Exercício 3.1. Adição sem sinal #1

a) $0 \le$ entrada $\le 2^5-1$ (ou seja, de 0 a 31)

$0 \le$ saída $\le 2^5-1$ (também de 0 a 31)

b) 16 + 15 = 10000 + 01111 = 11111 = 31 (correto)

c) 16 + 16 = 10000 + 10000 = 00000 = 0 (incorreto)

Exercício 3.3. Adição sem sinal #3

a) 111101 + 000001 = 111110 (61 + 1 = 62 → correto)

b) 111101 + 100001 = 011110 (61 + 33 = 30 → incorreto)

c) 000001 + 100001 = 100010 (1 + 33 = 34 → correto)

Exercício 3.4. Adição com sinal #1

a) –16 ≤ entrada ou saída ≤ 15

b) –8 – 8 = "11000" + "11000" = "10000" (= –16 → correto)

c) –8 – 9 = "11000" + "10111" = "01111" (= 14 → incorreto)

d) 8 – 9 = "01000" + "10111" = "11111" (= –1 → correto)

e) 8 + 8 = "01000" + "01000" = "00000" (= 0 → incorreto)

Exercício 3.5. Adição com sinal #2

a) $-2^4 \le$ entrada $\le 2^4-1$ (ou seja, de –16 a 15)

$-2^5 \le$ saída $\le 2^5-1$ (ou seja, de –32 a 31)

b) 11000 + 11000 = 110000 (–8 + –8 = –16 → correto)

c) 11000 + 10111 = 101111 (–8 + –9 = –17 → correto)

d) 01000 + 10111 = 111111 (8 + –9 = –1 → correto)

e) 01000 + 01000 = 010000 (8 + 8 = 16 → correto)

Exercício 3.7. Shift lógico #1

a) 00110

b) 11000

c) 01000

d) 11000

Exercício 3.9. Shift aritmético #1

a) 11110

b) 11111

c) 01000

d) 11111

Exercício 3.11. Shift circular #1

a) 10101

b) 01101

c) 10100

d) 01101

Exercício 3.13. Shift versus multiplicação sem sinal

a) 001111 (=15) → 011110 (=30, correto)

b) 010001 (=17) → 100010 (=34, correto)

c) 110011 (=51) → 100110 (=38, incorreto)

Há restrição quando o bit mais significativo é 1.

Exercício 3.15. Shift versus divisão sem sinal

a) 001100 (=12) → 000011 (=3, correto)

b) 000110 (=6) → 000001 (=1, correto, com arredondado para baixo)

c) 111101 (=61) → 001111 (=15, correto, com arredondado para baixo)

Não há restrições; apenas o resultado é arredondado para baixo.

Exercício 3.17. Multiplicação sem sinal #1

a) Resultado (com 10 bits): 00110 11001 (=217)

b) Resultado (com 10 bits): 00111 00000 (=224)

Exercício 3.19. Multiplicação com sinal #1

a) Resultado (com 10 bits): 11101 01100 (= –84)

b) Resultado (com 10 bits): 11100 00000 (= –128)

Exercício 3.21. Divisão sem signal

a) 31/7

Dividendo→ (=31)	0 0 0 0 1 1 1 1 1	0 0 1 1 1	←Divisor (=7)
	0 0 1 1 1	0 0 1 0 0	←Quociente (=4)
	0 0 0 0 0 1 1		←Resto (=3; usa-se 5 bits)

b) 16/4 → Quociente = 00100, Resto = 00000

Exercício 3.23. Adição/subtração com ponto flutuante #1

a) Para 1: $1 = 1_b \cdot 2^0$. Logo, $S = 0$, $F = 0...0$, $E = 127 = 01111111$.

Para 0,875: $0,875 = 7/8 = 111_b \cdot 2^{-3} = 1,11_b \cdot 2^{-1}$. Logo, $S = 0$, $F = 1100...00$, $E = e + 127 = 126 = 01111110$.

b) $1 + 0,875 = 1_b \cdot 2^0 + 1,11_b \cdot 2^{-1} = 1_b \cdot 2^0 + 0,111_b \cdot 2^0 = 1,111_b \cdot 2^0$. Logo, $S = 0$, $F = 1110...0$, $E = e + 127 = 127 = 01111111$.

c) $-1 + 0,875 = -1_b.2^0 + 1,11_b.2^{-1} = -1_b.2^0 + 0,111_b.2^0$

Como um dos valores é negativo, ele deve sofrer complementação de dois, isto é, $-1 \rightarrow 1$. Da soma então resulta $1_b \cdot 2^0 + 0,111_b \cdot 2^0 = 11,111_b \cdot 2^0$ (inverte-se o bit de carry para obter o MSB da soma – ver caso 2 na seção 3.2). Como o resultado é negativo, complementação de dois é novamente necessária, $11,111_b \rightarrow 00,001_b$, levando a $-00,001 \cdot 2^0 = -1 \cdot 2^{-3}$. Logo, $S = 1$, $F = 000...000$, $E = e + 127 = 124$.

Exercício 3.25. Adição/subtração com ponto flutuante #3

a) Para 8,125: 8,125 = 130/16 = 10000010/2^4 = 10000010·2^{-4} = 1,000001·2^3

Para –8: –8 = complemento de dois de 8 = 1000 = 1·2^3

Logo, 8,125 + (–8) = 1,000001·2^3 + 1·2^3 = 00,000001·2^3 = 1·2^{-3} (inverte-se o bit de carry para obter o MSB da soma – ver caso 2 na seção 3.2). Logo:

S = 0, *F* = 000...00, *E* = *e* + 127 = 124.

b) Para –19: –19 = complemento de dois de 19 = 101101 = 1,01101·2^5

Para –32,0625: 32,0625 = 513/16 = 1000000001/2^4 = 1000000001·2^{-4} = 1,000000001·2^5

Mas –32,0625 = complemento de dois de 32,0625 = 0,111111111·2^5

Logo, (–19) + (–32,0625) = (1,01101 + 0,111111111)·2^5 = 10,011001111·2^5 (aqui, não inverte-se o bit de carry para obter o MSB da soma – ver seção 3.2). Como o resultado é negativo, deve-se obter seu complemento de dois, 10,011001111·2^5 → 01,100110001·2^5. Portanto:

S = 1, *F* = 100110001, *E* = *e* + 127 = 132 (10000100)

Exercício 3.27. Multiplicação com ponto flutuante #2

a) 8,125 × (−8) = 1,000001·2^3 × (−1·2^3) = −1,000001·2^6. Truncando com 3 bits de fração, obtém-se −1,000·2^6. Logo, *S* = 1, *F* = 000 e *E* = *e* + 127 = 133 (10000101).

b) (−19) × (−12,5) = (−1,0011·2^4) × (−1,1001·2^3) = 1,11011011·2^7. Após truncamento, resulta 1,111·2^7. Logo, *S* = 0, *F* = 111 e *E* = e + 127 = 134 (10000110).

Exercício 3.29. Divisão com ponto flutuante #2

a) –4,75 / –0,1875 = –1,0011·2^2/–1,1·2^{-3}. Subtrai-se os expoentes e os significandos são divididos, resultando 0,11001010...·2^5 = 1,1001010...·2^4. Esse valor deve ser truncado (e arredondado, se necessário), resultando 1,101·2^4. Logo, *S* =0, *F* = 101, *E* = *e* + 127 = 131.

b) –19/–12,5 = –1,0011·2^4/–1,1001·2^3 = 0,1100001001...·2^1 = 1,100001001...·2^0. Após truncamento (e arredondado, se necessário), resulta 1,100·2^0. Logo, *S* =0 , *F* = 100, *E* = *e* + 127 = 127.

Capítulo 4: Introdução aos Circuitos Digitais

Exercício 4.1. Consumo de potência estática #1

a) $P_{static} = V_{DD}.I_D$ = 5 x 4.9/10k = 2.45 mW

b) $P_{static} = V_{DD}.I_D$ = 3.3 x 3.2/10k = 1.06 mW

Exercício 4.3. Consumo de potência estática #3

a) $P_{static} = V_{DD}.I_D$ = 0

b) $P_{static} = V_{DD}.I_D$ = 3.3V x 1pA = 3.3 pW

Exercício 4.5. Transistores ideais versus transistores não-ideais

Somente a constante de tempo será afetada, pois depois de certo tempo não mais fluirá corrente através do capacitor, não alterando assim a tensão final de *y*.

Exercício 4.6. Margens de ruído

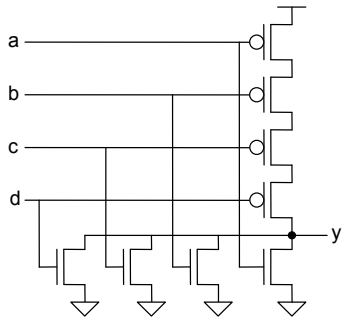
a) 0.6V, 1.1V
b) 0.18V, 0.18V
c) 0.12V, 0.12V

Exercício 4.7. Detectores de zeros e de uns

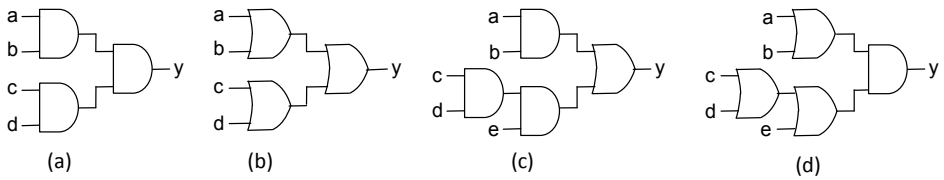
a) NOR (ver tabela da figura 1.7)

b) AND (ver tabela da figura 1.7)

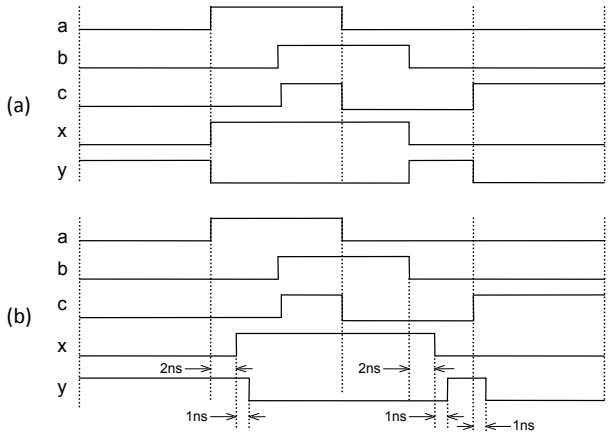
Exercício 4.9. Porta NOR CMOS de quatro entradas



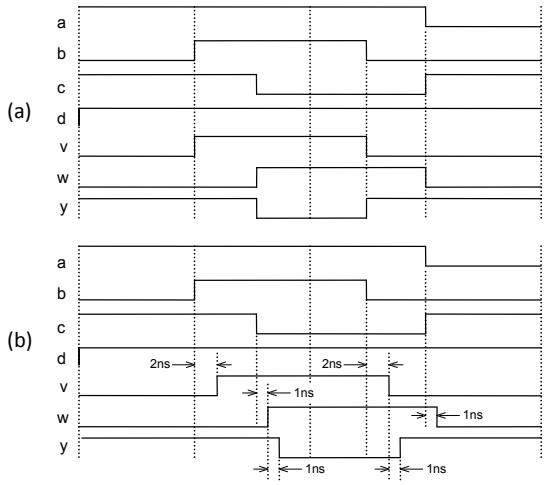
Exercício 4.11. Circuito AND/OR #2



Exercício 4.13. Análise temporal de um par OR-NOR



Exercício 4.15. Análise temporal de um circuito de NANDs



Exercício 4.17. Propriedades da porta XOR #2

a) 0⊕0⊕0 ...⊕0 = 0

Sabendo que 0⊕0 = 0 (equação 4.13), e aplicando isso repetitivamente, verifica-se que a propriedade é verdadeira.

b) 1⊕0⊕0⊕0...⊕0 = 1

Dado que 1⊕0 = 1 (equação 4.13), e aplicando este resultado aos demais zeros, concluímos que a propriedade é verdadeira.

c) 1⊕1⊕0⊕0...⊕0 = 0

Sabendo que 1⊕1 = 0 (equação 4.13), e usando a propriedade do item (a) acima, demonstra-se que esta propriedade também é verdadeira.

d) 1⊕1⊕1⊕0⊕0... ⊕0 = 1

Dado que 1⊕1 = 0 e 1⊕0 = 1, e repetindo este último resultado, verificamos que a propriedade é verdadeira.

e) a⊕a'⊕b⊕b' = 0

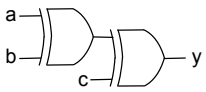
Levando em conta que a⊕a' = 1 (equação 4.14e), 1⊕b = b' (equação 4.14b) e b'⊕b' = 0 (equação 4.14c), temos que a propriedade é verdadeira.

Exercício 4.19. Porta XOR de 3 entradas

a b c	y
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

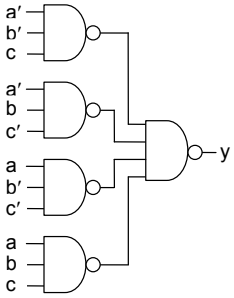
(a)

(b) $y = a'.b'.c + a'.b.c' + a.b'.c' + a.b.c$



(c)

(d)



Exercício 4.21. Tabela-verdade #1

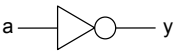
(a)

a	y
0	1
1	0

(b)

a	y
0	1
1	0

(c)



Exercício 4.23. Combinacional versus seqüencial

Aquele do exercício 4.22 (é um latch SR – veja figura 13.2b), pois no outro a saída não depende de entradas anteriores.

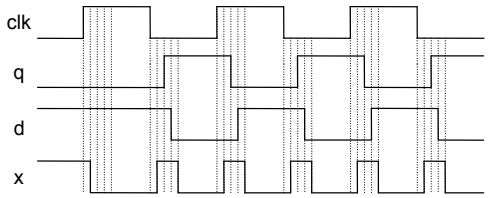
Exercício 4.25. Buffer de dreno aberto

a) A tensão de saída só assume valor alto quando todas as entradas estão altas, portanto é uma porta AND.

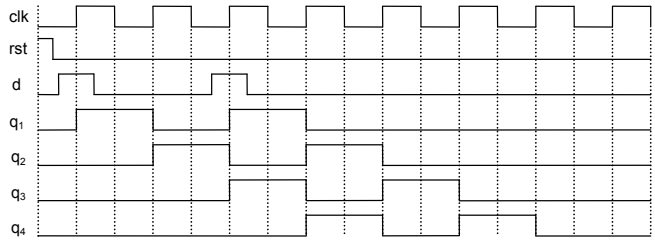
b) A potência é $P = V_{DD}^2/R = 5^2/10k = 2.5mW$, tanto para um transistor ligado como para todos ligados.

Exercício 4.27. Análise de tempo de um flip-flop #2

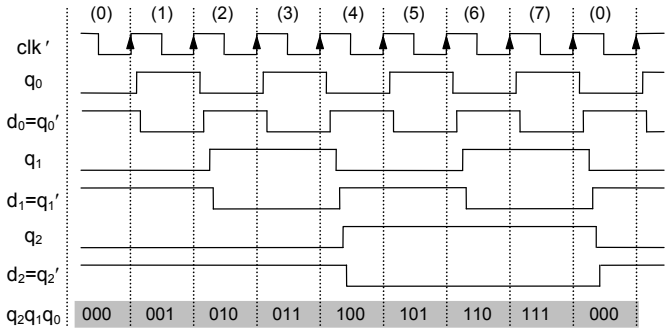
Note no diagrama abaixo que a frequência na saída é o dobro da frequência na entrada. Delays foram incluídos para facilitar a visualização das formas de onda.



Exercício 4.29. Análise temporal de registrador de deslocamento



Exercício 4.31. Análise temporal de um contador assíncrono



Capítulo 5: Álgebra Booleana

Exercício 5.1. Teorema do consenso

Equação 5.4a: Podemos escrever $b.c = a.b.c + a'.b.c$. Portanto:

$a.b + b.c + a'.c = a.b + a.b.c + a'.b.c + a'.c = a.b(1 + c) + a'.c(b + 1) = a.b + a'.c$

Equação 5.4b: Podemos escrever $(b + c) = (a + b + c).(a' + b + c)$. Logo:

$(a + b).(a + b + c).(a' + b + c).(a' + c)$.

Usando a equação 5.3a (duas vezes), obtém-se:

$(a + b).(a + b + c) = (a + b)$ e $(a' + b + c).(a' + c) = (a' + c)$. Logo:

$(a + b).(b + c).(a' + c) = (a + b).(a' + c)$

Exercício 5.3. Teorema do termo comum

Aplicando o princípio da dualidade: $y = a'b_1' + a'.b_2' + ... + a'b_n'$

Colocando a' em evidência: $y = a'(b_1' + b_2' + ... + b_n')$

Reaplicando o princípio da dualidade: $y = a + b_1b_2...b_n$

Exercício 5.5. Teoremas da absorção e do consenso

a) Podemos escrever $a.b' + a.b.c = a(b' + b.c)$. Aplicando o teorema da absorção à parte entre parênteses, obtem-se $a(b' + b.c) = a(b' + c) = a.b' + a.c$.

b) Esta é uma aplicação direta do teorema do consenso. Apenas substitua *b* com *a*, *c* com *b*, e *a* com *c*, e a equação dada se transformará na equação 5.4a.

Exercício 5.7. Propriedades XOR

a) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

$a \oplus (b'c + bc') = (a'b + ab') \oplus c$

$a'(b'c + bc') + a(b'c + bc')' = c'(a'b + ab') + c(a'b + ab')'$

$a'(b'c + bc') + a(b+c')(b'+c) = c'(a'b + ab') + c(a+b')(a'+b)$

$a'b'c + a'bc' + abc + ab'c' = a'bc' + ab'c' + abc + a'b'c$

b) $a(b \oplus c) = ab \oplus ac$

$a(b'c + bc') = (ab)'ac + ab(ac)'$

$ab'c + abc' = (a' + b')ac + ab(a' + c')$

$ab'c + abc' = ab'c + abc'$

Exercício 5.9. Lei de DeMorgan #1

a) $y = [a.(b.c)'.(d + (a'.d')')]' = [a.(b' + c').(d + a + d)]' = [a.b' + a.c']' = (a' + b).(a' + c) = a' + b.c.$

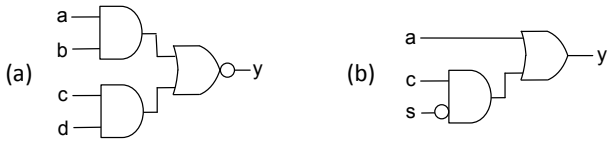
b) $y = a + b'.[\ c + d'.(a + b)'\ ]' = a + b'.[\ c + d'.a'.\ b'\ ]' = a + b'.\ c'.(a + b + d) = a + b'.\ c'.d$

c) $y = [((a + b)' + c).(a + (b + c)').(a + b + c)']' = [(\ a'.b' + c).(a + b'.c').a'.b'.c']' = [(a'.\ b'.c')]' = a + b + c$

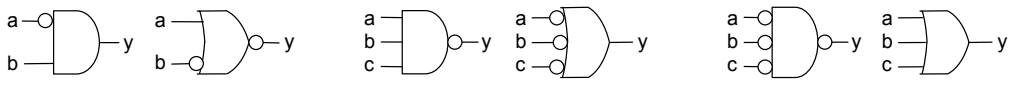
Exercício 5.11. Simplificação de circuito #1

a) A equação deste circuito (após simplificações) é $y = (a.b + c.d)'$, implementada, por exemplo, pelo circuito em (a).

a) A equação resultante é $y = a + c.s'$, implementada, por exemplo, pelo circuito em (b).



Exercício 5.13. Princípio da dualidade para portas AND



(a) (b) (c)

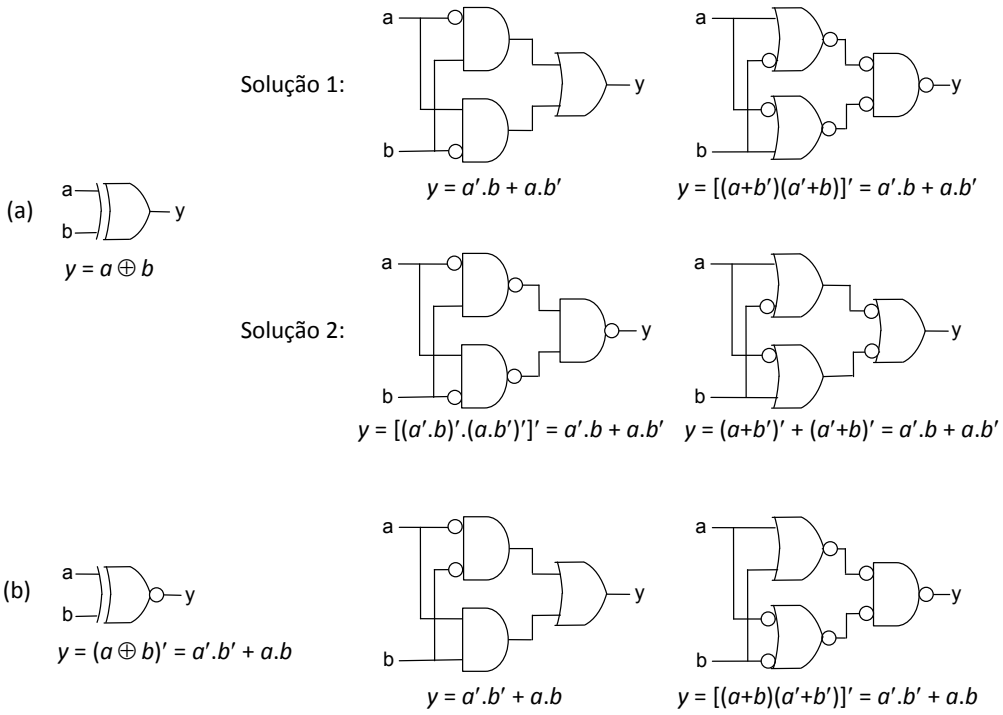
Antes: $y = a'.b = (a + b')'$
Depois: $y = (a + b')'$

Antes: $y = (a.b.c)' = a' + b' + c'$
Depois: $y = a' + b' + c'$

Antes: $y = (a'.b'.c')' = a + b + c$
Depois: $y = a + b + c$

Exercício 5.15. Princípio da dualidade para portas XOR

Lembre-se de que o princípio da dualidade lida com portas AND, OR e INV (não com portas XOR e XNOR, diretamente). Portanto, as decomposições mostradas nas figuras abaixo são necessárias. Para a parte (a), duas soluções são mostradas. Note que o circuito bem à direita é sempre o dual do circuito à sua esquerda.



Exercício 5.17. Expansão minterm/maxterm #2

Expansão de minterms: $y = m_2 + m_3 + m_4 + m_5 + m_6$ (onde m_2=$a'.b.c'$, m_3=$a'.b.c$, …, m_6=$a.b.c'$)

Expansão de maxterms: $y = M_0.M_1.M_7$ (onde M_0=$a+b+c$, M_1=$a+b+c'$, M_7=$a'+b'+c'$)

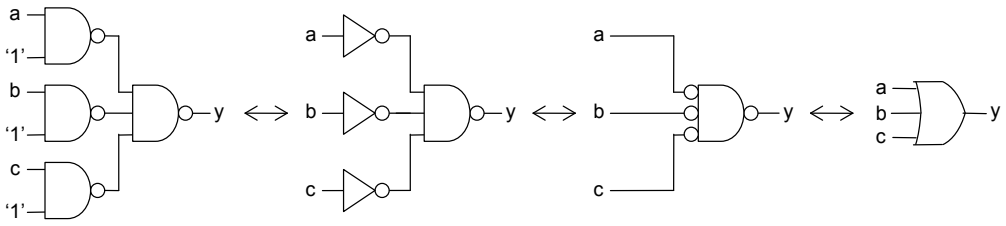
Exercício 5.19. Implicantes primos e essenciais

a) Disponível na seção 5.6.

b) Disponível na seção 5.6.

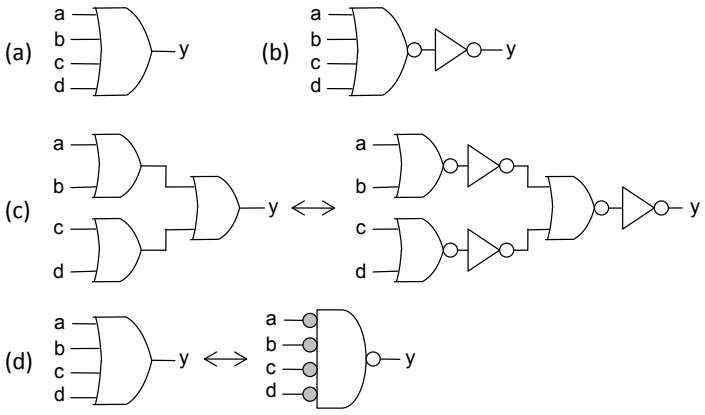
c) Máximo de 5 implicantes essenciais e mínimo de 5 implicantes primos.

Exercício 5.21. Circuito SOP padrão

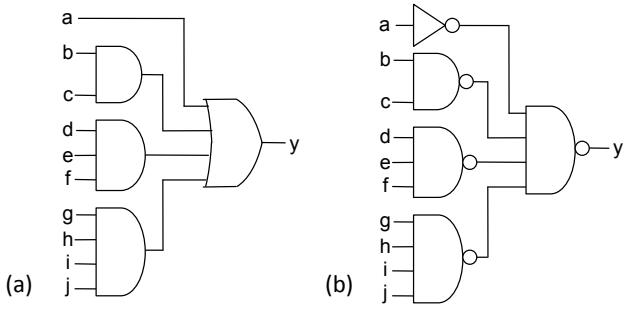


Exercício 5.23. Implementação de função #2

Os circuitos são mostrados abaixo. Os inversores foram representados explicitamente (usando o símbolo tradicional) ou através de bolhas (escuras). Em (b) e (c), o inversor pode ser construído com uma porta NOR com todas as entradas, exceto uma, conectadas a 0, enquanto que em (d) ele pode ser construído com uma porta NAND com todas as entradas, exceto uma, conectadas a 1 (como na figura 5.13c).



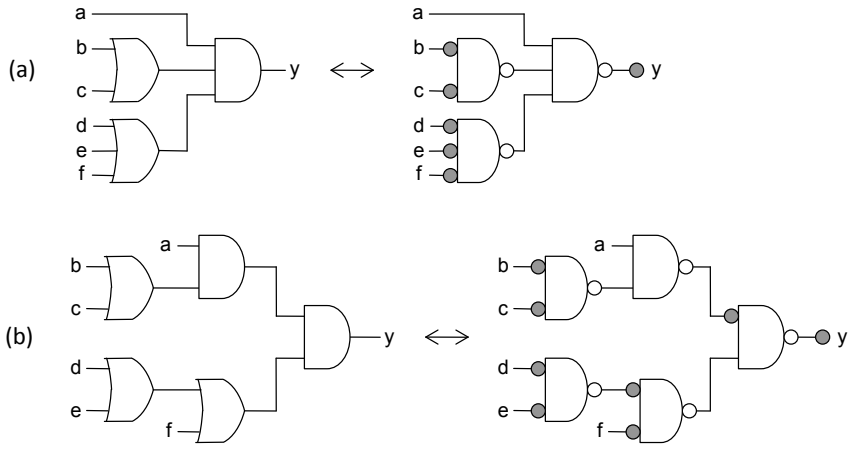
Exercício 5.25. Implementação de função #4



Para obter o circuito em (b), basta aplicar o princípio da dualidade à segunda camada do circuito em (a).

Exercício 5.27. Implementação de função #6

A solução abaixo foi obtida usando novamente o princípio da dualidade. Os inversores (bolhas escuras) podem ser construídos como explicado no exercício 5.23.



Exercício 5.29. Teorema do consenso

a) $y = a.b + b.c + a'.c$ é uma SOP, obtida por expansão de minterms, representada no mapa de Karnaugh da figura (a) abaixo. Como pode-se ver, o termo $b.c$ é de fato redundante.

A expansão original é $y = (a.b.c' + a.b.c) + (a'.b.c + a.b.c) + (a'.b'.c + a'.b.c) = m_6 + m_7 + m_3 + m_7 + m_1 + m_3 = m_1 + m_3 + m_6 + m_7$. Portanto, y=1 ocorre para abc = {001, 011, 110, 111}.

b) $y = (a+b).(b+c).(a'+c)$ é uma POS, obtida por expansão de maxterms, representada no mapa de Karnaugh abaixo. Como pode-se ver, o termo $b+c$ é de fato redundante.

A expansão original é $y = [(a+b+c').(a+b+c)].[(a'+b+c).(a+b+c)].[(a'+b'+c).(a'+b+c)] = M_1.M_0.M_4.M_0.M_6.M_4 = M_0.M_1.M_4.M_6$. Portanto, y=0 ocorre para abc = {000, 001, 100, 110}. Equivalentemente, usando a equação 5.12, $y = m_2 + m_3 + m_5 + m_7$, logo y=1 ocorre para abc = {010, 011, 101, 111}.

(a)

c \ a b	00	01	11	10
0	0	0	1	0
1	1	1	1	0

$a.b$, $a'.c$, $b.c$

(b)

c \ a b	00	01	11	10
0	0	1	0	0
1	0	1	1	1

$b+c$, $a+b$, $a'+c$

Exercício 5.31. Simplificação de função com mapas de Karnaugh #1

c \ a b	00	01	11	10
0	0	1	1	0
1	0	0	1	1

(a) $y = a.b + a'.b.c' + a.b'.c$
$= b.c' + a.c$

c \ a b	00	01	11	10
0	0	1	1	0
1	1	1	0	0

(b) $y = a'.b + a.b.c' + a'.b'.c$
$= b.c' + a'.c$

c d \ a b	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	1
10	1	0	0	1

(c) $y = a.b'.c' + a.b'.c.d + b'.c.d' + b'.c'.d'$
$= a.b' + b'.d'$

Exercício 5.33. Simplificação de função com mapas de Karnaugh #3

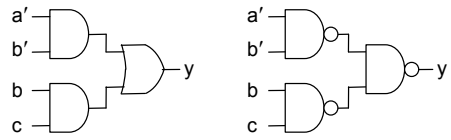
a) Conforme mostra o mapa de Karnaugh abaixo, há dois implicantes primos ($a'.b'$ e $b.c$). Ambos são tambem implicantes essenciais.

b) $y = a'.b' + b.c$

c-d) Veja circuitos abaixo.

c d \ a b	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	1	1	0
10	1	1	1	0

$a'.b'$, $b.c$



Exercício 5.35. Simplificação de função com mapas de Karnaugh #5

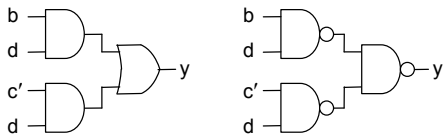
a) Conforme mostra o mapa de Karnaugh abaixo, há dois implicantes primos ($b.d$ e $c'.d$). Ambos são tambem implicantes essenciais.

b) $y = b.d + c'.d$

c-d) Veja circuitos abaixo.

c d \ a b	00	01	11	10
00	0	0	0	0
01	1	1	1	X
11	0	1	X	X
10	0	0	0	0

$c'.d$, $b.d$



Exercício 5.37. Mapa de Karnaugh extenso #1

a) Veja figura (a) abaixo.

b) Mostrados nas figuras (b)-(c).

c) $y = a'.y$ (para a=0) + $a.y$ (para a=1) = $a'.(d'.e + b.e) + a.(b'.c'.d) = a'.d'.e + a'.b.e + a.b'.c'.d$

minterm	a b c d e	y
m_1	0 0 0 0 1	1
m_5	0 0 1 0 1	1
m_9	0 1 0 0 1	1
m_{11}	0 1 0 1 1	1
m_{13}	0 1 1 0 1	1
m_{15}	0 1 1 1 1	1
m_{18}	1 0 0 1 0	1
m_{19}	1 0 0 1 1	1
others		0

(a)

(a=0)	b c			
d e	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

(b) $d'.e + b.e$

(a=1)	b c			
d e	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	0
10	1	0	0	0

(c) $b'.c'.d$

Exercício 5.39. Circuito combinacional com glitches #1

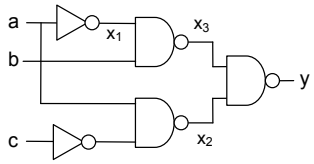
A solução é mostrada abaixo. Na parte superior consta a tabela verdade, com o mapa de Karnaugh, circuito e diagrama de tempo correspondentes. A função implementada é $y = a'.b + a.c'$, a qual apresenta um glitch na transição entre implicante primos (de a=1 para a=0, com b=1 e c=0). O delay de propagação foi considerado o mesmo (T) em todos os gates.

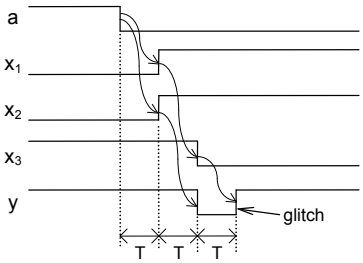
O circuito sem glitch consta na parte inferior da figura, contendo um implicante extra (primo, mas não essencial), responsável pela eliminação do glitch. A equação do novo circuito é $y = a'.b + a.c' + b.c'$.

a b c	y
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0

(a) Antes

	a b			
c	00	01	11	10
0	0	1	1	1
1	0	1	0	0





(b) Depois

	a b			
c	00	01	11	10
0	0	1	1	1
1	0	1	0	0