SOLUTIONS TO EXERCISES

for

INTRODUCTION TO DIGITAL SYSTEMS
Miloš D. Ercegovac, Tomás Lang and Jaime H. Moreno
Wiley & Sons, New York, 1999

prepared by

Alexandre F. Tenca, Tomás Lang, Miloš D. Ercegovac, and J.H. Moreno

with the help of

L. Alkalai, E. Gouriou, Y. He, M. Huguet, H. Liu, R. McIlhenny, M. Moraes de Azevedo, A. Nananarelli, V. Oklobdzija, A. Panangadan, M.D.F. Schlag, T. Torii, M. Tremblay, P. Vranes, and A. Wong.

Los Angeles, California
1999

Chapter 2

Exercise 2.1

Input: $x \in \{0, 1, ..., 9\}$ Output: $z \in \{0, 2, 4, 6, 8, 25, 36, 49, 64, 81\}$

$$z = \begin{cases} x^2 & \text{if } x > 4 \\ 2x & \text{otherwise} \end{cases}$$

x	0	1	2	3	4	5	6	7	8	9
$z = f(x)$	0	2	4	6	8	25	36	49	64	81

Exercise 2.2

Inputs: $\underline{x} = (x_{15}, x_{14}, ..., x_1, x_0)$ and $\underline{y} = (y_1, y_0)$, with $x_i, y_i \in \{0, 1, 2\}$.
Output: $\underline{z} = (z_{13}, z_{12}, ..., z_0)$, with $z_i \in \{0, 1, 2\}$.
Function:

$$\underline{z} = \begin{cases} (x_{15}, ..., x_{i+2}, x_{i-1}, ..., x_0) & \text{if } (x_{i+1}, x_i) = \underline{y} \text{ and } (x_{j+1}, x_j) \neq \underline{y} \text{ for } j > i \\ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) & \text{otherwise} \end{cases}$$

A tabular description would have as many rows as the number of combinations of $\underline{x}$ and $\underline{y}$, which is 3^{18}.

Exercise 2.3

Input: $\underline{x} = (x_{n-1}, \ldots, x_0) \in \{0, 1\}^n$
Output: an integer $1 \leq z \leq n \Leftrightarrow 1$.
Function:

$$z = \begin{cases} |i \Leftrightarrow j| & \text{if } (x_i = x_j = 1) \text{ and } (i \neq j) \text{ and} \\ & (\forall k \in \{0, \ldots, n \Leftrightarrow 1\}(i \neq k \neq j \Rightarrow x_k = 0)) \\ \text{Undefined} & \text{otherwise} \end{cases}$$

Exercise 2.4

Input: X is a 4×4 matrix of elements $x_{ij} \in \{a, b, c, d\}$:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

Output: $z \in \{0, 1\}$.
Function: The output is equal to 1 if one of x_{22}, x_{23}, x_{32}, or x_{33} is b and it is surrounded by eight a's. The conditional expression is

$$z = \begin{cases} 1 \textbf{ if } x_{i,j} = b \textbf{ and } (x_{k,m} = a \text{ for } k \in \{i \Leftrightarrow 1, i, i+1\} \text{ and } m \in \{j \Leftrightarrow 1, j, j+1\}) \\ \qquad \text{with } i, j \in \{2, 3\} \textit{ and } (k, m) \neq (i, j) \\ 0 \textbf{ otherwise} \end{cases}$$

A tabular representation would not be practical because there are $4 \times 4 = 16$ variables, each of which can have 4 values, resulting in a table of 4^{16} rows.

Exercise 2.5

Inputs: Integer $0 \leq x < 2^{16}$, binary control variable $d \in \{0, 1\}$

Output: Integer $0 \leq z < 2^{16}$.

$$z = \begin{cases} (x+1) \bmod 2^{16} & \text{if } d = 1 \\ (x \Leftrightarrow 1) \bmod 2^{16} & \text{if } d = 0 \end{cases}$$

Tabular representation: Requires 2^{16} rows for $d = 1$ and 2^{16} rows for $d = 0$.

Total rows $= 2 \times 2^{16} = 2^{17}$. A tabular representation is out of question.

Exercise 2.6

a	b	$f_1(a,b)$	$f_2(b, f_1(a,b))$
0	0	2	0
0	1	0	0
0	2	2	2
1	0	0	2
1	1	1	2
1	2	1	0
2	0	2	0
2	1	1	2
2	2	0	0

Exercise 2.7

Range from 10 to 25 $\Rightarrow$ 16 values.

Minimum number of binary variables: $\lceil \log_2 16 \rceil = 4$ variables.

Input: integer $10 \leq x \leq 25$ Output: integer $0 \leq z \leq 15$

$z = (x \Leftrightarrow 10)$

In the next table the output z is represented by a vector $\underline{z} = (z_3, z_2, z_1, z_0)$, with $z_i \in \{0, 1\}$.

x	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\underline{z}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Exercise 2.8

(a) Month: $\lceil \log_2 12 \rceil = 4$ bits

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011

(b) Day: $\lceil \log_2 31 \rceil = 5$ bits

Year (assuming 0-2500): $\lceil \log_2 2501 \rceil = 12$ bits

Month: 4 bits

A total of 21 bits would be needed.

(c) Each decimal digit needs 4 bits. Two digits are necessary to represent the day, and 4 digits to represent the year. The number of bits to represent these fields would be $6 \times 4 = 24$ bits. Adding

the bits used in the month field we get 28 bits total.
(d) If only one component is used, we need to specify a code for each particular day. This code may be calculated as:

$$(year \Leftrightarrow 1) * 12 * 31 + (month \Leftrightarrow 1) * 31 + day \Leftrightarrow 1$$

where *month*, *year* and *day* are the usual numbers associated to each field (Jan = 1, Feb = 2,...). Maximum date is 12/31/2500, which corresponds to the value:
$2499 * 372 + 11 * 31 + 30 = 929999$
So $\lceil log_2 929999 \rceil = 20$ bits are needed to represent all possible codes. The disadvantage of this method is that many calculations must be done to convert the code to the usual representation of the date, and vice-versa.

Exercise 2.9

decimal digit	2-4-2-1
0	0000
1	0001
2	0010 1000
3	0011 1001
4	0100 1010
5	0101 1011
6	0110 1100
7	0111 1101
8	1110
9	1111

Since there are six values that have two possible encodings, there are 2^6 different codes.

Exercise 2.10
Consider that the vector of weights is $\underline{w} = (w_0, w_1, w_2, w_3)$. A number x is represented as a vector $\underline{x} = (x_0, x_1, x_2, x_3)$ and its value in decimal is calculated as:

$$x = \sum_{i=0}^{3} x_i \cdot w_i, \qquad x_i \in \{0, 1\} \text{ and } w_i > 0$$

Without loss of generality we consider that the weights are sorted in ascending order, such that $w_0 \leq w_1 \leq w_2 \leq w_3$. The conditions that the weights have to satisfy in order to enable the representation of $x \in \{0, 1, 2, ..., 9\}$, assuming $w_i > 0$, are:

1. $\sum_{i-0}^{3} w_i \geq 9$. This condition is important to enable the generation of 9, and thus assure containment;

2. $w_0 = 1$, to generate $x = 1$;

3. $w_i \leq 9$, for all i;

4. Continuity condition: for $0 < i \leq 3$, w_i is chosen such that w_i is the sum of a subset (including the empty set) of weights w_j plus one, $j < i$.

   With this condition:

   $$w_1 \in \{w_0, w_0 + 1\} = \{1, 2\}$$

   thus, if we choose $w_1 = 1$, the set of values for w_2 is

   $$w_2 \in \{w_0, w_0 + 1, w_1 + 1, w_0 + w_1 + 1\} = \{1, 2, 3\}$$

   and for the choice $w_1 = 2$ we should have w_2 in the set $\{w_0, w_0 + 1, w_1 + 1, w_0 + w_1 + 1\} = \{1, 2, 3, 4\}$, but since we are assuming $w_0 \leq w_1 \leq w_2 \leq w_3$, we get:

   $$w_2 \in \{2, 3, 4\}$$

   Notice that when $w_2 \geq 5$ the value $x = 4$ cannot be represented since $w_0 + w_1 = 3$, and the other weights are greater than 4. The same reasoning is applied to the fourth weight.

Exercise 2.11
(a)

$$\begin{aligned} 34\ 567 &= (0011\ 0100\ 0101\ 0110\ 0111)_{BCD} \\ &= (0110\ 0111\ 1000\ 1001\ 1010\ )_{Excess-3} \end{aligned}$$

(b) BCD does not have the complementary property, so an actual subtraction is needed:

$$99\ 999 \Leftrightarrow 34\ 567 = 65\ 432 = (0110\ 0101\ 0100\ 0011\ 0010)_{BCD}$$

2421 code has the complementary property, such that the subtraction is done by complementing each bit:

$$\begin{aligned} 34\ 567 &= (0011\ 0100\ 1011\ 1100\ 1101)_{2421\text{-code}} \\ 99\ 999 \Leftrightarrow 34\ 567 &= (1100\ 1011\ 0100\ 0011\ 0010)_{2421\text{-code}} \end{aligned}$$

Exercise 2.12
a) All possible 5-bit vectors $\underline{x} = (x_4 x_3 x_2 x_1 x_0)$ and the valid 2-out-of-5 combinations (marked with a tick mark ($\surd$)) are shown in Table 2.1.
b)
Input= $\underline{x} = (x_4 x_3 x_2 x_1 x_0)$

Output= $z \in \{0, 1\}$

Function:

$$z = \begin{cases} 1 \textbf{ if } (x_i = x_j = 1 \ and \ x_k = 0) \ for \\ \quad i \neq j \neq k \ and \ i, j, k \in \{0, 1, 2, 3, 4\} \\ 0 \textbf{ otherwise} \end{cases}$$

$\underline{x} = (x_4 x_3 x_2 x_1 x_0)$	valid 2-out-of-5 combinations
00000	
00001	
00010	
00011	$\surd$
00100	
00101	$\surd$
00110	$\surd$
00111	
01000	
01001	$\surd$
01010	$\surd$
01011	
01100	$\surd$
01101	
01110	
01111	
10000	
10001	$\surd$
10010	$\surd$
10011	
10100	$\surd$
10101	
10110	
10111	
11000	$\surd$
11001	
11010	
11011	
11100	
11101	
11110	
11111	

Table 2.1: Valid 2-out-of-5 combinations – Exercise 2.12(a)

Exercise 2.13

(a) $(1001010100011110)_2 = (1001\ 0101\ 0001\ 1110\ )_2 = (951E)_{16}$

(b) $(3456)_8 = (011\ 100\ 101\ 110)_2 = (011100101110)_2$

(c) To convert from radix-2 to radix-2^k we consider groups of k bits. The digits in radix-2^k are obtained converting each group (binary representation of the digit) into a single value in the new radix.

To convert from radix-2^k to radix-2, the digits in radix-2^k are converted to binary. The final vector, that corresponds to the concatenation of all digit representations in binary, is the radix-2 representation of the number.

Exercise 2.14

We could directly build the tables of the functions and show that the identity holds by comparing the tables. However, this would be time consuming since the table would have 32 rows. Therefore, we look at different assignments as follows:

- i) if $a = 0$ and $b = 0$ then both sides of the identity have value 0, since $M(x, y, z) = 0$ when two of its arguments are 0.
- ii) if $a = 1$ and $b = 1$ then both sides have value 1, since $M(x, y, z) = 1$ when two arguments are 1.
- iii) the two cases $a \neq b$ are equivalent due to the symmetry of the majority function and the fact that these variables appear together in all functions in the expression. Consequently, we check only the case $a = 0$ and $b = 1$. For this assignment,

$$M[0, 1, M(c, d, e)] = M(c, d, e)$$

and

$$M[M(0, 1, c), d, M(0, 1, e)] = M[c, d, e]$$

The identity is true for all assignments of a and b and is therefore true.

Exercise 2.15

(a) Prove that $f_{\text{XOR}}(f_{\text{AND}}(x_1, x_0), f_{\text{AND}}(x_1, x_0)) = f_{\text{EQUIVALENCE}}(x_1, x_0)$

x_1	x_0	$f_{\text{AND}}(x_1, x_0)$	$f_{\text{XOR}}(f_{\text{AND}}(x_1, x_0), f_{\text{AND}}(x_1, x_0))$	$f_{\text{EQUIVALENCE}}(x_1, x_0)$
0	0	0	0	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

The conclusion is:

$$f_{\text{XOR}}(f_{\text{AND}}(x_1, x_0), f_{\text{AND}}(x_1, x_0)) \neq f_{\text{EQUIVALENCE}}(x_1, x_0)$$

(b) Prove that $f_{\text{NAND}}(f_{\text{NAND}}(x_1, x_0), f_{\text{NAND}}(x_1, x_0)) = f_{\text{AND}}(x_1, x_0)$

x_1	x_0	$f_{\text{NAND}}(x_1, x_0)$	$f_{\text{NAND}}(f_{\text{NAND}}(x_1, x_0), f_{\text{NAND}}(x_1, x_0))$	$f_{\text{AND}}(x_1, x_0)$
0	0	1	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

The conclusion is:

$$f_{\text{NAND}}(f_{\text{NAND}}(x_1, x_0), f_{\text{NAND}}(x_1, x_0)) = f_{\text{AND}}(x_1, x_0)$$

Exercise 2.16

Each variable can have 2 values (0 or 1).
Total number of n-variable inputs: $\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{n} = 2^n$

For each input, the output function can have 2 values.
Total number of functions: $\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{2^n} = 2^{2^n}$

Exercise 2.17

a) Let f be a symmetric switching function of three variables, x, y, and z. Since the function is symmetric, $f(0,0,1) = f(0,1,0) = f(1,0,0)$ and $f(0,1,1) = f(1,0,1) = f(1,1,0)$ so that we have the following table:

x	y	z	f
0	0	0	a
0	0	1	b
0	1	0	b
0	1	1	c
1	0	0	b
1	0	1	c
1	1	0	c
1	1	1	d

where a, b, c, and d are binary variables. From this description any particular example can be generated by assigning values to a, b, c, and d.

b) Since a, b, c, and d can each take two values (0 or 1), the number of symmetric functions of three variables is $2^4 = 16$.

c) A symmetric switching function has the same value for all argument values that have the same number of 1's, since all these values can be obtained by permuting the arguments. Consequently, the values of the function of n arguments are decomposed into $n+1$ classes defined by the number of 1's in the argument vector (four classes in part a). Therefore the set A completely defines the function.

d) The function has value 1 whenever $0, 2$ or 3 arguments have value 1. The table is

w	x	y	z	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

e) In part (c) we saw that the argument values of symmetric switching function of n variables are divided into $n+1$ classes. For each of these classes the function can have value 1 or 0. Consequently, the number of symmetric switching functions of n variables is 2^{n+1}.

f) No, the composition of symmetric switching functions is not necessarily a symmetric function. Consider as counterexample

$$f(x, y, z) = f_{AND}(f_{OR}(x, y), z)$$

and interchange the variables x and z.

g) The table is

a	b	c	f_1	f_2	f
0	0	0	1	1	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	0	1	0

h) Let U be the set $\{0, 1, \ldots, n\}$ where n is the number of variables of the symmetric function, and let A be the set describing the function f. Since the complement of f is 0 when f is 1 and viceversa, it is represented by the set A_c such that

$$A_c = U \Leftrightarrow A$$

For example, considering a 4-variable symmetrical function with $A = \{0, 1\}$, we have $A_c = \{2, 3, 4\}$.

Exercise 2.18

(a) The threshold switching function of three variables with $w_1 = 1$, $w_2 = 2$, $w_3 = \Leftrightarrow 1$, and $T = 2$ is:

x	$\underline{x} = \{x_3, x_2, x_1\}$	$z = f(x)$
0	000	0
1	001	0
2	010	0
3	011	1
4	100	0
5	101	0
6	110	0
7	111	0

one-set = {3}
zero-set = {0,1,2,3,4,5,6,7}

(b) The system of inequalities for the function $one \Leftrightarrow set(3,5,7)$ is:

$$\begin{aligned} 0 &\leq T && (2.1)\\ w_0 &\leq T && (2.2)\\ w_1 &\leq T && (2.3)\\ w_2 &\leq T && (2.4)\\ w_0 + w_1 &> T && (2.5)\\ w_0 + w_2 &> T && (2.6)\\ w_1 + w_2 &\leq T && (2.7)\\ w_2 + w_1 + w_0 &> T && (2.8) \end{aligned}$$

A solution for this system is:

$$w_0 = 2 \quad w_1 = 1 \quad w_2 = 1 \quad T = 2$$

(c) We have the following eight equations:

$$\begin{aligned} 0 &> T && (2.9)\\ w_0 &\leq T && (2.10)\\ w_1 &\leq T && (2.11)\\ w_1 + w_0 &> T && (2.12) \end{aligned} \qquad \begin{aligned} w_2 &\leq T && (2.13)\\ w_2 + w_0 &> T && (2.14)\\ w_2 + w_1 &> T && (2.15)\\ w_2 + w_1 + w_0 &\leq T && (2.16) \end{aligned}$$

From these equations, we get:

$w_0 + w_1 \leq 2T$ [(10)+(11)]
$T < w_0 + w_1 \leq 2T$ [(12) and (10)+(11)]

Therefore:

$$T < 2T \qquad \text{but (1) gives} \qquad T < 0$$

These conditions are incompatible, we can't find such a threshold function.
(d) The majority function is expressed as a threshold function with the weights $w0 = w_1 = w_2 = w_3 = 1$, and $T = 2$. Since $x_0 + x_1 + x_2 + x_3 > 2 \Leftrightarrow$ three or four bits have the value 1

Exercise 2.19 The input of the system is $\underline{x} = (x_3, x_2, x_1, x_0)$, and the output is the bit-vector $\underline{z} = (z_1, z_0)$. The output encoding and the respective function table are:

z	$z_1 z_0$
1	0 1
2	1 0
3	1 1

	$x_3x_2x_1x_0$	z_1	z_0
0	0000	-	-
1	0001	-	-
2	0010	-	-
3	0011	0	1
4	0100	-	-
5	0101	1	0
6	0110	0	1
7	0111	-	-
8	1000	-	-
9	1001	1	1
10	1010	1	0
11	1011	-	-
12	1100	0	1
13	1101	-	-
14	1110	-	-
15	1111	-	-

z_1 : one-set(5,9,10), zero-set(3, 6, 12), dc-set(0,1,2,4,7,8,11,13,14,15)
z_0 : one-set(3,6,9,12), zero-set(5, 10), dc-set(0,1,2,4,7,8,11,13,14,15)

Exercise 2.20 Using BCD code, the function specified in Exercise 2.1 is described as:

x	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0
3	0	0	0	0	0	1	1	0
4	0	0	0	0	1	0	0	0
5	0	0	1	0	0	1	0	1
6	0	0	1	1	0	1	1	0
7	0	1	0	0	1	0	0	1
8	0	1	1	0	0	1	0	0
9	1	0	0	0	0	0	0	1
10	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-

Using a two-dimensional table:

x_3x_2 \ x_1x_0	00	01	10	11
00	00000000	00000010	00000100	00000110
01	00001000	00100101	00110110	01001001
10	01100100	10000001	-	-
11	-	-	-	-

Using one-set and d.c.-set representation:

- $z_7 = one\_set(9), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_6 = one\_set(7, 8), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_5 = one\_set(5, 6, 8), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_4 = one\_set(6), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_3 = one\_set(4, 7), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_2 = one\_set(2, 3, 5, 6, 8), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_1 = one\_set(1, 3, 6), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_0 = one\_set(5, 7, 9), dc\_set(10, 11, 12, 13, 14, 15)$

Exercise 2.21: Using EXCESS-3 code, the function specified in Exercise 2.1 is described as:

x	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0
0	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	0	0	1	1	0	0	1	1
4	0	0	1	1	0	1	0	1
5	0	0	1	1	0	1	1	1
6	0	0	1	1	1	0	0	1
7	0	0	1	1	1	0	1	1
8	0	1	0	1	1	0	0	0
9	0	1	1	0	1	0	0	1
10	0	1	1	1	1	1	0	0
11	1	0	0	1	0	1	1	1
12	1	0	1	1	0	1	0	0
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-

Using a two-dimensional table:

x_3x_2 \ x_1x_0	00	01	10	11
00	-	-	-	00110011
01	00110101	00110111	00111001	00111011
10	01011000	01101001	01111100	10010111
11	10110100	-	-	-

Using one-set and dc-set representation:

- $z_7 = one\_set(11, 12), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_6 = one\_set(8, 9, 10), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_5 = one\_set(3, 4, 5, 6, 7, 9, 10, 12), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_4 = one\_set(3, 4, 5, 6, 7, 8, 10, 11, 12), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_3 = one\_set(6, 7, 8, 9, 10), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_2 = one\_set(4, 5, 10, 11, 12), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_1 = one\_set(3, 5, 7, 11), dc\_set(10, 11, 12, 13, 14, 15)$
- $z_0 = one\_set(3, 4, 5, 6, 7, 9, 11), dc\_set(10, 11, 12, 13, 14, 15)$

Exercise 2.22 Using 2-out-of-5 code, the function specified in Exercise 2.1 is described by the following table. As the table has 32 entries, only the 10 entries that are useful are shown, the others have don't care values:

x	z_9	z_8	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_0
3	0	0	0	1	1	0	0	0	1	1
5	0	1	0	0	1	1	1	0	0	0
6	0	1	1	0	0	0	0	1	1	0
9	0	0	1	1	0	1	0	0	1	0
10	1	0	1	0	0	0	1	0	1	0
12	0	0	0	1	1	0	0	1	1	0
17	1	0	0	1	0	0	0	1	0	1
18	0	0	0	1	1	1	0	0	0	1
20	0	0	0	1	1	1	0	0	1	0
24	0	0	0	1	1	1	0	1	0	0

Using a two-dimensional table:

	x_1x_0			
$x_4x_3x_2$	00	01	10	11
000	-	-	-	0001100011
001	-	0100111000	0110000110	-
010	-	0011010010	1010001010	-
011	0001100110	-	-	-
100	-	1001000101	0001101001	-
101	0001110010	-	-	-
110	0001110100	-	-	-
111	-	-	-	-

Using one-set and d.c.-set representation:

- $z_9 = one\_set(10, 17)$,
  $dc\_set(0, 1, 2, 4, 7, 8, 11, 13, 14, 15, 16, 19, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31)$

- $z_8 = one\_set(5,6)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_7 = one\_set(6,7,8)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_6 = one\_set(3,9,12,17,18,20,24)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_5 = one\_set(3,5,12,18,20,24)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_4 = one\_set(5,9,18,20,24)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_3 = one\_set(5,10)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_2 = one\_set(6,12,17,24)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_1 = one\_set(3,6,9,10,12,20)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$
- $z_0 = one\_set(3,17,18)$,
  $dc\_set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)$

Exercise 2.23
The number of values of n binary variables is 2^n. For each of these, the function can take three possible values ($0, 1$ or dc) resulting in 3^{2^n} functions. Out of these 2^{2^n} are completely specified functions.

Exercise 2.24
(a)

$$\begin{aligned} a'b' + ab + a'b &= a'(b + b') + ab \\ &= a'1 + ab \\ &= a' + b \end{aligned}$$

(b)

$$\begin{aligned} a' + a(a'b + b'c)' &= a' + (a'b + b'c)' \\ &= a' + (a + b')(b + c') \\ &= a' + ab + ac' + bb' + b'c' \\ &= a' + b + c' + b'c' \\ &= a' + b + c' \end{aligned}$$

(c)

$$\begin{aligned}(a'b'+c)(a+b)(b'+ac)' &= (a'b'+c)(a+b)b(a'+c')\\ &= (a'b'+c)b(a'+c')\\ &= (ba'b'+bc)(a'+c')\\ &= bc(a'+c')\\ &= a'bc\end{aligned}$$

(d)

$$\begin{aligned}ab'+b'c'+a'c' &= ab'+(a+a')b'c'+a'c'\\ &= ab'+ab'c'+a'b'c'+a'c'\\ &= ab'(1+c')+a'(1+b')c'\\ &= ab'+a'c'\end{aligned}$$

(e)

$$\begin{aligned}wxy+w'x(yz+yz')+x'(zw+zy')+z(x'w'+y'x) &= wxy+w'xy+z(x'(w+y')+x'w'+y'x)\\ &= (w+w')xy+z(x'(w+y'+w')+y'x)\\ &= xy+z(x'+xy')\\ &= xy+z(x'+y')\\ &= xy+z(xy)'\\ &= xy+z\end{aligned}$$

(f)

$$\begin{aligned}abc'+bc'd+a'bd &= abc'+(a+a')bc'd+a'bd\\ &= abc'+abc'd+a'bc'd+a'bd\\ &= abc'(1+d)+a'bd(1+c')\\ &= abc'+a'bd\end{aligned}$$

Exercise 2.25

$$\begin{aligned}xz'+x'z = &\\ &= x(xy'+x'y)'+x'(xy'+x'y)\\ &= x(xy')'(x'y)'+x'xy'+x'x'y\\ &= x(x'+y)(x+y')+x'y\\ &= (xx'+xy)(x+y')+x'y\\ &= (xyx+xyy')+x'y\\ &= xy+x'y\\ &= (x+x')y\\ &= 1\cdot y\end{aligned}$$

Exercise 2.26

$$\begin{aligned} a + a'b + a'b'c + a'b'c'd + a'b'c'd'e &= \\ &= a + a'b + a'b'c + a'b'c'(d + d'e) \\ &= a + a'b + a'b'(c + c'(d + e)) \\ &= a + a'b + a'b'(c + d + e) \\ &= a + a'(b + b'(c + d + e)) \\ &= a + a'(b + c + d + e) \\ &= a + b + c + d + e \end{aligned}$$

Exercise 2.27 (a) With the assumption that $c = a * b$ we obtain

$$\begin{aligned} b * c &= b * (a * b) \\ &= b * (ab + a'b') \\ &= b(ab + a'b') + b'(ab + a'b')' \\ &= ab + a'bb' + b'(ab')'(a'b')' \\ &= ab + b'(a' + b')(a + b) \\ &= ab + (a'b' + b')(a + b) \\ &= ab + a'ab' + a'bb' + ab' + b'b \\ &= ab + ab' \\ &= a(b + b') \\ &= a \end{aligned}$$

Therefore, $a = b * c$
(b)

$$\begin{aligned} a * bc &= a * b(a * b) \\ &= a * b(ab + a'b') \\ &= a * (ab + ba'b') \\ &= a * ab \\ &= aab + a'(ab)' \\ &= ab + a'(a' + b') \\ &= ab + a' + a'b' \\ &= ab + a' \\ &= a' + b \end{aligned}$$

Therefore, $a * bc \neq 1$

Exercise 2.28 As a counterexample consider the case $a = 0, b = 0$, and $c = 1$. For this set of values $ab = ac = 0$ and $b \neq c$.

Exercise 2.29 To be a boolean algebra the system must satisfy the postulates of *commutativity, distributivity, complement and additive (multiplicative) identity.* Lets consider each one of these cases:

abc	$a\#(b\&c)$	$(a\#b)\&(a\#c)$
000	$0\#0=0$	$0\&0=0$
001	$0\#1=0$	$0\&0=0$
002	$0\#2=0$	$0\&0=0$
010	$0\#1=0$	$0\&0=0$
011	$0\#1=0$	$0\&0=0$
012	$0\#2=0$	$0\&0=0$
020	$0\#2=0$	$0\&0=0$
021	$0\#2=0$	$0\&0=0$
022	$0\#2=0$	$0\&0=0$
100	$1\#0=0$	$0\&0=0$
101	$1\#1=1$	$0\&1=1$
102	$1\#2=1$	$0\&1=1$
110	$1\#1=1$	$1\&0=1$
111	$1\#1=1$	$1\&1=1$
112	$1\#2=1$	$1\&1=1$
120	$1\#2=1$	$1\&0=1$
121	$1\#2=1$	$1\&1=1$
122	$1\#2=1$	$1\&1=1$
200	$2\#0=0$	$0\&0=0$
201	$2\#1=1$	$0\&1=1$
202	$2\#2=2$	$0\&2=2$
210	$2\#1=1$	$1\&0=1$
211	$2\#1=1$	$1\&1=1$
212	$2\#2=2$	$1\&2=2$
220	$2\#2=2$	$2\&0=2$
221	$2\#2=2$	$2\&1=2$
222	$2\#2=2$	$2\&2=2$

Table 2.2: Proof of distributivity for system in Exercise 2.29

- (a) $(\#), (\&)$ are commutative because the table is symmetric about the main diagonal, so postulate 1 $(P1)$ is satisfied.
- (b) For distributivity, we must show that $a\#(b\&c) = (a\#b)\&(a\#c)$. Let us prove that this postulate is true for the system by perfect induction, as shown in Table 2.2.
- (c) The additive identity element is 0 since $a\&0 = 0\&a = a$. The multiplicative identity element is 2 since $a\#2 = 2\#a = a$.
- (d) For 1 to have a complement $1'$ we need $1\&1' = 2$ and $1\#1' = 0$. Consequently, from the table we see that 1 has no complement.

Exercise 2.30

Let us call the four elements $0, x, y$, and 1 with 0 the additive identity, 1 the multiplicative identity and let x be the complement of y. The corresponding operation tables are

+	0	x	y	1
0	0	x	y	1
x	x	x	1	1
y	y	1	y	1
1	1	1	1	1

*	0	x	y	1
0	0	0	0	0
x	0	x	0	x
y	0	0	y	y
1	0	x	y	1

Exercise 2.31

To show that the NAND operator is not associative, that means, $((a.b)'.c)' \neq (a.(b.c)')'$ we just need to find a case when the expressions are different. Take $a = 0$, $b = 0$ and $c = 1$. For these values we have:

$((a.b)'.c)' = ((0.0)'.1)' = (1.1)' = 0$

$(a.(b.c)')' = (0.(0.1)')' = 1$

So, the NAND operator is not associative.

To show that the NOR operator is not associative, that means, $((a + b)' + c)' \neq (a + (b + c)')'$ we take $a = 0$, $b = 0$ and $c = 1$. For these values we have:

$((a + b)' + c)' = ((0 + 0)' + 1)' = 0$

$(a + (b + c)')' = (0 + (0 + 1)')' = (0 + 0)' = 1$

So, the NOR operator is not associative.

Exercise 2.32

Part (a)

(i) show that XOR operator is commutative:

a	b	$a \oplus b$	$b \oplus a$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

(ii) show that XOR operator is associative: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

abc	$a \oplus b$	$b \oplus c$	$(a \oplus b) \oplus c$	$a \oplus (b \oplus c)$
000	0	0	0	0
001	0	1	1	1
010	1	1	1	1
011	1	0	0	0
100	1	0	1	1
101	1	1	0	0
110	0	1	0	0
111	0	0	1	1

(iii) show that XOR operator is distributive with respect to AND operator, that means:
$a.(b \oplus c) = ab \oplus ac$

abc	$a.b$	$a.c$	$(b \oplus c)$	$a.(b \oplus c)$	$ab \oplus ac$
000	0	0	0	0	0
001	0	0	1	0	0
010	0	0	1	0	0
011	0	0	0	0	0
100	0	0	0	0	0
101	0	1	1	1	1
110	1	0	1	1	1
111	1	1	0	0	0

Part (b)
(a) prove that if $x \oplus y = 0$ then $x = y$.
$x \oplus y = xy' + x'y = 0 \rightarrow (x = 1$ **and** $y = 1)$ **or** $(x = 0$ **and** $y = 0)$
This implies $x = y$.
(b) $x' \oplus y = (x \oplus y)'$

x	y	$x' \oplus y$	$(x \oplus y)'$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

(c) $x \oplus y = x' \oplus y'$

x	y	$x \oplus y$	$x' \oplus y'$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

(d) prove that if $x \oplus y = z \oplus y$ then $x = z$.
$x \oplus y = z \oplus y$
$(x \oplus y) \oplus y = (z \oplus y) \oplus y$
$x \oplus (y \oplus y) = z \oplus (y \oplus y)$ (Associativity)
$x \oplus 0 = z \oplus 0$ $(y \oplus y = 0)$
$x = z$
Using the function table we come to the same conclusion. The lines where $x \oplus y = z \oplus y$ are marked with an *.

x	y	z	$x \oplus y$	$z \oplus y$	
0	0	0	0	0	*
0	0	1	0	1	
0	1	0	1	1	*
0	1	1	1	0	
0	0	0	0	0	*
0	0	1	0	1	
0	1	0	1	1	*
0	1	1	1	0	
1	0	0	1	0	
1	0	1	1	1	*
1	1	0	0	1	
1	1	1	0	0	*
1	0	0	1	0	
1	0	1	1	1	*
1	1	0	0	1	
1	1	1	0	0	*

(e) For an even number of x's the expression

$$x \oplus x \oplus x \oplus x \oplus ... \oplus x$$

can be rewritten as

$$(x \oplus x) \oplus (x \oplus x) \oplus ...(x \oplus x)$$

since $x \oplus x = 0$ the expression becomes:

$$0 \oplus 0 \oplus 0... \oplus 0$$

and since $0 \oplus 0 = 0$, the resulting value is 0.
For an odd number of x's the same grouping of x's can be done and the expression is transformed into:

$$0 \oplus 0 \oplus 0...0 \oplus x = 0 \oplus x = 0'x + 0.x' = x$$

Exercise 2.33
(a) *Yes.* (b) *Yes.* (c) *No*, Unmatched parenthesis.

Exercise 2.34
(a)

$$\begin{aligned} E(w,x,y,z) &= w + x' + (x+y)(w+z)' \\ E(0,1,1,0) &= 0 + 1' + (1+1)(0+0)' = 0 + 0 + (1)(0)' = 0 + 0 + 1 \cdot 1 \\ &= 1 \\ E(1,1,1,0) &= 1 + 1' + (1+1)(1+0)' = 1 + 0 + (1)(1)' = 1 + 1 \cdot 0 \\ &= 1 \end{aligned}$$

(b)

$$\begin{aligned} E(x_3, x_2, x_1, x_0) &= x_2x_1 + x_3 \\ E(0, 1, 1, 0) &= 1 \cdot 1 + 0 = 1 + 0 = 1 \\ E(1, 1, 1, 0) &= 1 \cdot 1 + 1 = 1 + 1 = 1 \end{aligned}$$

Exercise 2.35 Since the only term containing w is the same in both expressions (yw), to simplify the tabular description (only 3 variables) we transform the proof as follows:

$$xyz + yw + x'z' + xy' = y'z' + yw + xz + x'yz' \text{ if } xyz + x'z' + xy' = y'z' + xz + x'yz'$$

Let us call $E_1 = xyz + x'z' + xy$ and $E_2 = y'z' + xz + x'yz'$. We show that $E_1 = E_2$ in the following table.

x	y	z	E_1	E_2
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Note that if E_1 is not equivalent to E_2 it is still possible that the original expressions are equivalent. An example of this type of situation is $ab' + b = a + b$, in which ab' and a are not equivalent.

Exercise 2.36
(a) $E(x, y) = xy + xy'$

xy	$E(x, y)$
0 0	0
0 1	0
1 0	1
1 1	1

(b) $E(x, y, z) = xyz + x'y + xyz'$

x	y	z	$E(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Exercise 2.37

			Expr a	Expr b	Expr c	Expr d	Expr e
x	y	z	$x'y' + xz + xz'$	$xy + x'y' + yz'$	$xyz + x'y'z + x'z' + xyz'$	$y'z + x'z' + xyz$	$x'y' + x'z' + xyz$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	0	1	0
1	1	0	0	1	1	0	0
1	1	1	1	1	1	1	1

Equivalent expressions: Expr a and Expr d, Expr b and Expr c.

Exercise 2.38

(a) $E(a,b,c,d) = abc'd + ab'c + bc'd + ab'c' + acd + a'bcd$

$$\begin{aligned} E(a,b,c,d) &= c'd(ab+b) + ab'(c+c') + (a+a'b)cd \\ &= bc'd + ab' + (a+b)cd \\ &= bd(c+c') + ab' + acd \\ &= bd + ab' + a(b+b')cd \\ &= bd(1+ac) + ab'(1+cd) \\ &= ab' + bd \end{aligned}$$

(b) $E(a,b,c,d) = acb + ac'd + bc'd + a'b'c' + ab'c'd' + bc'd$

$$\begin{aligned} E(a,b,c,d) &= bc'(d+d') + b'c'(a'+ad') + acb + ac'd \\ &= bc' + b'c'a' + b'c'd' + acb + ac'd \\ &= c'(b + b'a' + b'd' + ad) + acb \\ &= c'(b + a' + d + d') + acb \\ &= c' + acb \\ &= ab + c' \end{aligned}$$

Exercise 2.39

$f = a'b'c + a'bc' + a'bc + ab'c = m_1 + m_2 + m_3 + m_5 = \sum m(1,2,3,5)$

$f = (a+b+c)(a'+b+c)(a'+b'+c)(a'+b'+c') = M_0 + M_4 + M_6 + M_7 = \prod M(0,4,6,7)$

Exercise 2.40

(a)

$$\begin{aligned} a'b + ac + bc &= a'b(c+c') + a(b+b')c + (a+a')bc \\ &= a'bc + a'bc' + abc + ab'c + abc + a'bc \\ &= m_3 + m_2 + m_7 + m_5 + m_7 + m_3 \\ &= \sum m(2,3,5,7) = \prod M(0,1,4,6) \end{aligned}$$

(b) $g(a,b,c,d,e,f) = (ab+c)(d'e+f) = g_1(a,b,c)g_2(d,e,f)$
Since the g_1 and g_2 are disjoint, we may obtain their minterms separately and combine them later to get the minterms for $g(a,b,c,d,e,f)$.

$$g_1(a,b,c) = ab + c = abc + abc' + abc + a'bc + ab'c + a'b'c = \sum m(1,3,5,6,7) = \prod M(0,2,4)$$

$$g_2(d,e,f) = d'e + f = d'ef + d'ef' + def + d'ef + de'f + d'e'f = \sum m(1,2,3,5,7) = \prod M(0,4,6)$$

If g_1 has a minterm $m_i(a,b,c)$ and g_2 has a minterm $m_j(d,e,f)$, since the expression for g is the AND of both expressions, g has the minterms $m_{8i+j}(a,b,c,d,e,f) = m_i(a,b,c).m_j(d,e,f)$. Consequently, the minterms and maxterms of g are:

CSP: $\sum m(\ 9,10,11,13,15,25,26,27,29,31,41,42,43,45,47,49,50,51,53,55,57,58,59,61,63)$
CPS: $\prod M(\ 0,1,2,3,4,5,6,7,8,12,14,16,17,18,19,20,21,22,23,24,28,30,32,33,34,35,36,37,$
$38,39,40,44,46,48,52,54,56,60,62)$

(c)

$$\begin{aligned} a'b(ab+c)(b+c'd) &= (a'bab + a'bc)(b+c'd) = a'bc(b+c'd) \\ &= a'bcb + a'bcc'd = a'bc(d+d') \\ &= a'bcd + a'bcd' \end{aligned}$$

CSP: $\sum m(6,7)$
CPS: $\prod M(0,1,2,3,4,5,8,9,10,11,12,13,14,15)$

Exercise 2.41
CSP equivalent:

$$\begin{aligned} E(x,y,z) &= x' + x(x'y + y'z)' \\ &= x'(y+y')(z+z') + x((x'y)'.(y'z)') \\ &= x'yz + x'yz' + x'y'z + x'y'z' + x(x+y').(y+z') \\ &= x'yz + x'yz' + x'y'z + x'y'z' + x(xy + xz' + y'y + y'z') \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xy + xz' + xy'z' \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xy(z+z') + xz'(y+y') + xy'z' \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xyz + xyz' + xyz' + xy'z' + xy'z' \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xyz + xyz' + xy'z' \\ &= \sum m(0,1,2,3,4,6,7) \end{aligned}$$

CPS equivalent:

$$\begin{aligned} E(x,y,z) &= x' + x(x'y + y'z)' \\ &= (x'+x).(x' + (x'y + y'z)') \\ &= x' + (x'y)'(y'z)' \\ &= (x' + (x'y)')(x' + (y'z)') \\ &= (x' + x + y')(x' + y + z') \\ &= 1.(x' + y + z') \\ &= \prod M(5) \end{aligned}$$

Exercise 2.42

a) We first convert to a sum of products

$$\begin{aligned}[((a+b+a'c')c+d)'+ab]' = & \\ &= (((a+b+a'c')c+d)')'(ab)' \\ &= ((a+b+a'c')c+d)(a'+b') \\ &= (ac+bc+d)(a'+b') \\ &= a'bc+a'd+ab'c+b'd\end{aligned}$$

To get the CSP we expand and eliminate repeated minterms

$$\begin{aligned}&= a'bc(d+d')+a'd(b+b')(c+c')+ab'c(d+d')+b'd(a+a')(c+c') \\ &= a'bcd+a'bcd'+a'bc'd+a'b'cd+a'b'c'd+ab'cd+ab'cd'+ab'c'd\end{aligned}$$

In m-notation,

$$E(a,b,c,d) = \sum m(1,3,5,6,7,9,10,11)$$

b) We first get a sum of products

$$\begin{aligned}&[(w'+(xy+xyz'+(x+z)')')(z'+w')]' = \\ &= (w'+(xy+xyz'+x'z')')'+(z'+w')' \\ &= (w(xy+xyz'+x'z')+zw \\ &= wxy+wxyz'+wx'z'+wz\end{aligned}$$

Now we expand and eliminate repeated minterms

$$\begin{aligned}&= wxy(z+z')+wxyz'+wx'z'(y+y')wz(x+x')(y+y') \\ &= wxyz+wxyz'+wx'yz'+wx'y'z'+wxy'z+wx'yz+wx'y'z\end{aligned}$$

In m-notation,

$$E(w,x,y,z) = \sum m(8,9,10,11,13,14,15)$$

Exercise 2.43

(a) $E(w,x,y,z) = xyz+yw+x'z'+xy'$

$$\begin{aligned}E(w,x,y,z) &= x(yz+y')+yw+x'z' \\ &= x(y'+z)+yw+x'z' \\ &= (x(y'+z)+yw+x')(x(y'+z)+yw+z') \\ &= (y'+z+yw+x')(xy'+z'+xz+yw) \\ &= (w+x'+y'+z)(xy'+x+z'+yw) \\ &= (w+x'+y'+z)(x+z'+yw) \\ &= (w+x'+y'+z)(w+x+z')(x+y+z') \\ &= (x+x'+y'+z)(w+x+y+z')(w+x+y'+z')(w'+x+y+z')\end{aligned}$$

(b) $E(a,b,c,d) = (abc + ab')'(a'b + c')$

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

Exercise 2.44
(a) $\sum m(0,1,3,4,11,12,14,15) = \prod M(2,5,6,7,8,9,10,13)$

(b) $\prod M(0,1,3,5,6,9,10,13) = \sum m(2,4,7,8,11,12,14,15)$

Exercise 2.45

$$f(x,y,z) = \prod M(0,1,4,6,7)$$

$$g(x,y,z) = (f(x,y,z))'$$

$$g(x,y,z) = \sum m(0,1,4,6,7) = \prod M(2,3,5)$$

Exercise 2.46
(a)
Input: $A, B \in \{0,1,2,3\}$ and $C_{in} \in \{0,1\}$
Output: $Z \in \{0,1,2,3\}$ and $C_{out} \in \{0,1\}$

$$Z = (A + B + C_{in}) \bmod 4$$

$$C_{out} = \begin{cases} 1 & \text{if } (A + B + C_{in}) \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

(b)

C_{in}	A	B	C_{out}	Z
0	0	0	0	0
0	0	1	0	1
0	0	2	0	2
0	0	3	0	3
0	1	0	0	1
0	1	1	0	2
0	1	2	0	3
0	1	3	1	0
0	2	0	0	2
0	2	1	0	3
0	2	2	1	0
0	2	3	1	1
0	3	0	0	3
0	3	1	1	0
0	3	2	1	1
0	3	3	1	2
1	0	0	0	1
1	0	1	0	2
1	0	2	0	3
1	0	3	1	0
1	1	0	0	2
1	1	1	0	3
1	1	2	1	0
1	1	3	1	1
1	2	0	0	3
1	2	1	1	0
1	2	2	1	1
1	2	3	1	2
1	3	0	1	0
1	3	1	1	1
1	3	2	1	2
1	3	3	1	3

(c)

i	C_{in}	a_1	$a0$	b_1	b_0	C_{out}	z_1	z_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	1
2	0	0	0	1	0	0	1	0
3	0	0	0	1	1	0	1	1
4	0	0	1	0	0	0	0	1
5	0	0	1	0	1	0	1	0
6	0	0	1	1	0	0	1	1
7	0	0	1	1	1	1	0	0
8	0	1	0	0	0	0	1	0
9	0	1	0	0	1	0	1	1
10	0	1	0	1	0	1	0	0
11	0	1	0	1	1	1	0	1
12	0	1	1	0	0	0	1	1
13	0	1	1	0	1	1	0	0
14	0	1	1	1	0	1	0	1
15	0	1	1	1	1	1	1	0
16	1	0	0	0	0	0	0	1
17	1	0	0	0	1	0	1	0
18	1	0	0	1	0	0	1	1
19	1	0	0	1	1	1	0	0
20	1	0	1	0	0	0	1	0
21	1	0	1	0	1	0	1	1
22	1	0	1	1	0	1	0	0
23	1	0	1	1	1	1	0	1
24	1	1	0	0	0	0	1	1
25	1	1	0	0	1	1	0	0
26	1	1	0	1	0	1	0	1
27	1	1	0	1	1	1	1	0
28	1	1	1	0	0	1	0	0
29	1	1	1	0	1	1	0	1
30	1	1	1	1	0	1	1	0
31	1	1	1	1	1	1	1	1

$$\begin{aligned}\text{one-set}(C_{out}) &= \{7, 10, 11, 13, 14, 15, 19, 22, 23, 25, 26, 27, 28, 29, 30, 31\}\\ \text{one-set}(z_1) &= \{2, 3, 5, 6, 8, 9, 12, 15, 17, 18, 20, 21, 24, 27, 30, 31\}\\ \text{one-set}(z_0) &= \{1, 3, 4, 6, 9, 11, 12, 14, 16, 18, , 21, 23, 24, 26, 29, 31\}\end{aligned}$$

Exercise 2.47

(a)

Inputs: Integer $x \in \{0, \ldots, 15\}$, control variable Ctl $\in$ {Increment,Decrement}

Output: $z = \begin{cases} (x+1) \bmod 16 & \text{if Ctl=Increment} \\ (x \Leftrightarrow 1) \bmod 16 & \text{if Ctl=Decrement} \end{cases}$

(b) See Table 2.3 on page 26.

(c) We have to choose a code for x, Ctl and z. Coding x as $\underline{x} = (x_3, x_2, x_1, x_0)$ and z as $\underline{z} = (z_3, z_2, z_1, z_0)$, using the conventional binary representation. Coding Ctl as c in binary with 0 meaning Decrement and 1 meaning Increment. Table 2.4 on page 26 shows the tabular representation of the switching functions.

The zero sets are:

zero-set(z_3)={1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 31}

zero-set(z_2)={1, 2, 3, 4, 9, 10, 11, 12, 16, 17, 18, 23, 24, 25, 26, 31}

	z	
x	Ctl=Decrement	Ctl=Increment
0	15	1
1	0	2
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	8
8	7	9
9	8	10
10	9	11
11	10	12
12	11	13
13	12	14
14	13	15
15	14	0

Table 2.3: Exercise 2.47(b)

cx_3x_2	x_1x_0			
	00	01	10	11
000	1111	0000	0001	0010
001	0011	0100	0101	0110
010	0111	1000	1001	1010
011	1011	1100	1101	1110
100	0001	0010	0011	0100
101	0101	0110	0111	1000
110	1001	1010	1011	1100
111	1101	1110	1111	0000
	(z_3, z_2, z_1, z_0)			

Table 2.4: Exercise 2.47(c)

zero-set(z_1)=$\{1, 2, 5, 6, 9, 10, 13, 14, 16, 19, 20, 23, 24, 27, 28, 31\}$
zero-set(z_0)=$\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 19, 31\}$

Exercise 2.48
(a) A high-level description is

- Input: $\underline{x} = (x_3, x_2, x_1, x_0)$, and $x_i \in \{0, 1\}$
- Output: $z \in \{0, 1, 2, 3, 4\}$
- Function:

$$z = \sum_{i=0}^{3} x_i$$

(b) the table for arithmetic function and (c) the table for the switching functions considering binary code for z are shown next:

$x_3x_2x_1x_0$	z
0000	0
0001	1
0010	1
0011	2
0100	1
0101	2
0110	2
0111	3
1000	1
1001	2
1010	2
1011	3
1100	2
1101	3
1110	3
1111	4

$x_3x_2x_1x_0$	$z_2z_1z_0$
0000	000
0001	001
0010	001
0011	010
0100	001
0101	010
0110	010
0111	011
1000	001
1001	010
1010	010
1011	011
1100	010
1101	011
1110	011
1111	100

The one-sets that correspond to this table are:
one-set(z_2)=$\{15\}$
one-set(z_1)=$\{3, 5, 6, 7, 9, 10, 11, 12, 13, 14\}$
one-set(z_0)=$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Exercise 2.49
(a)

- Inputs: x, y where $x, y \in \{0, 1, 2, 3\}$
- Output: $z \in \{0, 1, 2, 3, 4, 6, 9\}$
- Function: $z = x \cdot y$

(b) The table for the arithmetic function is

x, y	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9

(c) considering binary representation for inputs and outputs we obtain the following table:

	x		y		z			
i	x_1	x_0	y_1	y_0	z_3	z_2	z_1	z_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	0	0
5	0	1	0	1	0	0	0	1
6	0	1	1	0	0	0	1	0
7	0	1	1	1	0	0	1	1
8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	1	0
10	1	0	1	0	0	1	0	0
11	1	0	1	1	0	1	1	0
12	1	1	0	0	0	0	0	0
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	1	1	0
15	1	1	1	1	1	0	0	1

one-set$(z_3) = \{15\}$
one-set$(z_2) = \{10, 11, 14\}$
one-set$(z_1) = \{6, 7, 9, 11, 13, 14\}$
one-set$(z_0) = \{5, 7, 13, 15\}$

Exercise 2.50 (a) A conditional expression is

$$L = \begin{cases} ON \textbf{ if } \text{ an odd number of switches are ON} \\ OFF \textbf{ otherwise} \end{cases}$$

The state of the light cannot be considered as an input because a feedback loop is formed and the circuit generated with this loop will not be stable (it will oscillate for some combinations of the switches).
(b) Calling w, x, y, z the four switches, the table for the switching functions is:

$wxyz$	L
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

(c) The minimal sum of products expression for this function is

$$L = w'x'y'z + w'x'yz' + w'xy'z' + w'xyz + wx'y'z' + wx'yz + wxy'z + wxyz'$$

Exercise 2.51

a)

- Input: x — a decimal digit
- Output: z — a decimal digit
- Function: $z = 9 \Leftrightarrow x$

b) The tables for the specified codes are shown in book. Inputs and outputs use the same code. The exercise asks for only one case, but for the sake of completeness we show the solution for all four cases.

b1) Tabular description of the function using Excess-6 code.

dec	$\underline{x} = (x_3, x_2, x_1, x_0)$	$\underline{z} = (z_3, z_2, z_1, z_0)$
0	0110	1111
1	0111	1110
2	1000	1101
3	1001	1100
4	1010	1011
5	1011	1010
6	1100	1001
7	1101	1000
8	1110	0111
9	1111	0110

z_3:one-set(6,7,8,9,10,11,12,13), dc-set(0,1,2,3,4,5)

z_2:one-set(6,7,8,9,14,15), dc-set(0,1,2,3,4,5)
z_1:one-set(6,7,10,11,14,15), dc-set(0,1,2,3,4,5)
z_0:one-set(6,8,10,12,14), dc-set(0,1,2,3,4,5)
c1) Switching expressions:

$$\begin{aligned} z_3 &= x_3x_2' + x_3x_2x_1' + x_3'x_2x_1 \\ z_2 &= x_3'x_2x_1 + x_3x_2'x_1' + x_3x_2x_1 \\ z_1 &= x_1 \\ z_0 &= x_0' \end{aligned}$$

b2) Tabular description of the function using 2-out-of-5 code.

dec	$\underline{x} = (x_4, x_3, x_2, x_1, x_0)$	$\underline{z} = (z_4, z_3, z_2, z_1, z_0)$
0	00011	00101
1	11000	01001
2	10100	10001
3	01100	00110
4	10010	01010
5	01010	10010
6	00110	01100
7	10001	10100
8	01001	11000
9	00101	00011

The dc-set is the same for all output functions and correspond to
dc-set(0,1,2,4,7,8,11,13,14,15,16,19,21,22,23,25,26,27,28,29,30,31)
z_4:one-set(20,10,17,9)
z_3:one-set(24,18,6,9)
z_2:one-set(3,12,6,17)
z_1:one-set(12,18,10,5)
z_0:one-set(3,24,20,5)
c2) Switching expressions:

$$\begin{aligned} z_4 &= x_4x_3'x_2x_1'x_0' + x_4'x_3x_2'x_1x_0' + x_4x_3'x_2'x_1x_0 + x_4'x_3x_2'x_1'x_0 \\ z_3 &= x_4x_3x_2'x_1'x_0' + x_4x_3'x_2'x_1x_0' + x_4'x_3'x_2x_1x_0' + x_4'x_3x_2'x_1x_0 \\ z_2 &= x_4'x_3'x_2'x_1x_0 + x_4'x_3x_2x_1'x_0' + x_4'x_3'x_2x_1x_0' + x_4x_3'x_2'x_1x_0 \\ z_1 &= x_4'x_3x_2x_1'x_0' + x_4x_3'x_2'x_1x_0' + x_4'x_3x_2'x_1x_0' + x_4'x_3'x_2x_1'x_0 \\ z_0 &= x_4'x_3'x_2'x_1x_0 + x_4x_3x_2'x_1'x_0' + x_4x_3'x_2x_1'x_0' + x_4'x_3'x_2x_1'x_0 \end{aligned}$$

b3) Tabular description of the function using 4,3,2,1 code. Since this code allows two representations for some decimal values (such as 3 or 4) we adopted the representation with least number of 1 bits.

dec	$\underline{x} = (x_3, x_2, x_1, x_0)$	$\underline{z} = (z_3, z_2, z_1, z_0)$
0	0000	1110
1	0001	1101
2	0010	1100
3	0100	1010
4	1000	0110
5	0110	1000
6	1010	0100
7	1100	0010
8	1101	0001
9	1110	0000

z_3:one-set(0,1,2,3,6), dc-set(3,5,7,9,11,15)
z_2:one-set(0,1,2,8,10), dc-set(3,5,7,9,11,15)
z_1:one-set(0,4,8,12), dc-set(3,5,7,9,11,15)
z_0:one-set(1,13), dc-set(3,5,7,9,11,15)
c3) Switching expressions:

$$\begin{aligned} z_3 &= x_3'x_2'x_1' + x_3'x_2'x_1x_0' + x_3'x_2x_1'x_0' + x_3x_2'x_1'x_0 \\ z_2 &= x_3'x_2'x_1' + x_3'x_2'x_1x_0' + x_3x_2'x_1'x_0' + x_3'x_2x_1x_0 \\ z_1 &= x_3'x_2'x_1'x_0' + x_3'x_2x_1'x_0' + x_3x_2'x_1'x_0 + x_3x_2x_1'x_0' \\ z_0 &= x_3'x_2'x_1'x_0 + x_3x_2x_1'x_0 \end{aligned}$$

b4) Tabular description of the function using 8,-2,2,-1 code.

dec	$\underline{x} = (x_3, x_2, x_1, x_0)$	$\underline{z} = (z_3, z_2, z_1, z_0)$
0	0000	1011
1	0011	1000
2	0010	1001
3	1101	1110
4	1100	1111
5	1111	1100
6	1110	1101
7	1001	0010
8	1000	0011
9	1011	0000

z_3:one-set(0,2,3,12,13,14,15), dc-set(1,4,5,6,7,10)
z_2:one-set(12,13,14,15), dc-set(1,4,5,6,7,10)
z_1:one-set(0,8,9,12,13), dc-set(1,4,5,6,7,10)
z_0:one-set(0,2,8,12,14), dc-set(1,4,5,6,7,10)
c4) Switching expressions:

$$\begin{aligned} z_3 &= x_3x_2 + x_3'x_2'x_1 + x_3'x_2'x_1'x_0' \\ z_2 &= x_2 \\ z_1 &= x_1' \\ z_0 &= x_0' \end{aligned}$$

Exercise 2.52

(a)

Input:

$\underline{x} = (x_3, x_2, x_1, x_0) \in \{0,1\}^4$

Output:

$\underline{z} = (z^1, z^0)$, where z^1 and z^0 are in the range $[0,9]$, and $z = 10z^1 + z^0$.

Function:

$\underline{z}^1 = BCD(z^1) = (z_7, z_6, z_5, z_4)$ and $\underline{z}^0 = BCD(z^0) = (z_3, z_2, z_1, z_0)$, with $z_i \in \{0,1\}$

As the value of x is $x = \sum_{i=0}^{3} x_i 2^i$, we define the digit values z^1 and z^0 as:

$$z^1 = x \text{ div } 10$$

$$z^0 = x \bmod 10$$

(b)

Since $0 \leq x \leq 15$ then $z^1 \in \{0,1\}$, and therefore:

$$z_7 = z_6 = z_5 = 0$$

one-set$(z_4) = \{10, 11, 12, 13, 14, 15\}$
one-set$(z_3) = \{8, 9\}$
one-set$(z_2) = \{4, 5, 6, 7, 14, 15\}$
one-set$(z_1) = \{2, 3, 6, 7, 12, 13\}$
Since $z_0 = z^0 \bmod 2 = (x \bmod 10) \bmod 2 = x_0$ then $z_0 = x_0$.
Therefore :

$$\begin{aligned}
z_7 &= 0 \\
z_6 &= 0 \\
z_5 &= 0 \\
z_4 &= x_3x_2'x_1x_0' + x_3x_2'x_1x_0 + x_3x_2x_1'x_0' + x_3x_2x_1'x_0 + x_3x_2x_1x_0' + x_3x_2x_1x_0 = x_3(x_1 + x_2) \\
z_3 &= x_3x_2'x_1'x_0' + x_3x_2'x_1'x_0 = x_3x_2'x_1' \\
z_2 &= x_3'x_2x_1'x_0' + x_3'x_2x_1'x_0 + x_3'x_2x_1x_0' + x_3'x_2x_1x_0 + x_3x_2x_1x_0' + x_3x_2x_1x_0 = x_3'x_2 + x_3x_2x_1 \\
z_1 &= x_3'x_2'x_1x_0' + x_3'x_2'x_1x_0 + x_3'x_2x_1x_0' + x_3'x_2x_1x_0 + x_3x_2x_1'x_0' + x_3x_2x_1'x_0 = x_3'x_1 + x_3x_2x_1' \\
z_0 &= x_0
\end{aligned}$$

Exercise 2.53

Let the inputs be a and b and the output w and z, all hexadecimal integers. A high-level description is

$$w = \begin{cases} a \textbf{ if } a \geq b \\ b \textbf{ otherwise} \end{cases}$$

$$z = \begin{cases} a \textbf{ if } a < b \\ b \textbf{ otherwise} \end{cases}$$

b) In the radix-2 representation all bit-vectors have four bits.

$$\underline{a} = (a_3, a_2, a_1, a_0), \quad \underline{b} = (b_3, b_2, b_1, b_0)$$
$$\underline{z} = (z_3, z_2, z_1, z_0) \quad \underline{w} = (w_3, w_2, w_1, w_0)$$

For the binary description we use the intermediate variables g, e, and s obtained from a comparison of a and b, so that

$$\begin{aligned} z_i &= a_i(g+e) + b_i s \\ w_i &= b_i(g+e) + a_i s \end{aligned}$$

Switching expressions for g, e, and s are

$$\begin{aligned} g &= a_3 b_3' + e_3 a_2 b_2' + e_3 e_2 a_1 b_1' + e_3 e_2 e_1 a_0 b_0' \\ e &= e_3 e_2 e_1 e_0 \\ s &= g' e' \end{aligned}$$

where $e_i = a_i b_i + a_i' b_i'$

Exercise 2.54

a) A high level description of the priority encoder can be given as:

- Input: $\underline{x} = (x_7, x_6, ..., x_1, x_0)$, with $x_i \in \{0, 1\}$.
- Output: integer $0 \leq z \leq 7$
- Function:
$$z = k, \text{ **where** } x_k = 1 \text{ and } (x_j = 0 \text{ for every } j < k)$$

b) In order to obtain switching expressions more easily, we divide the priority encoder into two subsystems: a priority resolution system and an encoder. This is shown in Figure 2.16 in the textbook.

The priority resolution subsystem has the same number of outputs as inputs, and gives a 1 on the output that corresponds to the highest priority input that has value 1. So, the switching expressions for the priority resolutor are

$$\begin{aligned} w_0 &= x_0 \\ w_1 &= x_1 x_0' \\ w_2 &= x_2 x_1' x_0' \\ w_3 &= x_3 x_2' x_1' x_0' \\ w_4 &= x_4 x_3' x_2' x_1' x_0' \\ w_5 &= x_5 x_4' x_3' x_2' x_1' x_0' \\ w_6 &= x_6 x_5' x_4' x_3' x_2' x_1' x_0' \\ w_7 &= x_7 x_6' x_5' x_4' x_3' x_2' x_1' x_0' \end{aligned}$$

The second part is an encoder. If the output is represented in the binary code, we obtain the following table:

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	z_2	z_2	z_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Now we can write the output switching expressions:

$$\begin{aligned} z_0 &= w_1 + w_3 + w_5 + w_7 \\ z_1 &= w_2 + w_3 + w_6 + w_7 \\ z_2 &= w_4 + w_5 + w_6 + w_7 \end{aligned}$$

c) For the output in Gray code we obtain the table:

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	z_2	z_1	z_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	0	0	1	0	0

The output expressions are

$$\begin{aligned} z_0 &= w_1 + w_2 + w_5 + w_6 \\ z_1 &= w_2 + w_3 + w_4 + w_5 \\ z_2 &= w_4 + w_5 + w_6 + w_7 \end{aligned}$$

Exercise 2.55

The combinational comparator module compares hexadecimal digits a and b. The output z has three values called G, E, and S. A high-level specification is

$$z = \begin{cases} G & \textbf{if } a > b \\ E & \textbf{if } a = b \\ S & \textbf{if } a < b \end{cases}$$

A tabular representation of this function is given next:

	b															
a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	E	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
1	G	E	S	S	S	S	S	S	S	S	S	S	S	S	S	S
2	G	G	E	S	S	S	S	S	S	S	S	S	S	S	S	S
3	G	G	G	E	S	S	S	S	S	S	S	S	S	S	S	S
4	G	G	G	G	E	S	S	S	S	S	S	S	S	S	S	S
5	G	G	G	G	G	E	S	S	S	S	S	S	S	S	S	S
6	G	G	G	G	G	G	E	S	S	S	S	S	S	S	S	S
7	G	G	G	G	G	G	G	E	S	S	S	S	S	S	S	S
8	G	G	G	G	G	G	G	G	E	S	S	S	S	S	S	S
9	G	G	G	G	G	G	G	G	G	E	S	S	S	S	S	S
10	G	G	G	G	G	G	G	G	G	G	E	S	S	S	S	S
11	G	G	G	G	G	G	G	G	G	G	G	E	S	S	S	S
12	G	G	G	G	G	G	G	G	G	G	G	G	E	S	S	S
13	G	G	G	G	G	G	G	G	G	G	G	G	G	E	S	S
14	G	G	G	G	G	G	G	G	G	G	G	G	G	G	E	S
15	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	E

For the binary description, digit a is represented by the vector $\underline{a} = (a_3, a_2, a_1, a_0)$ and digit b is represented by $\underline{b} = (b_3, b_2, b_1, b_0)$. We encode the output variable z using 3 bits as follows:

z	z_2	z_1	z_0
G	1	0	0
E	0	1	0
S	0	0	1

We can derive switching expressions for the outputs considering the equality variable e_i (bits a_i and b_i are equal) and the conditional conditional expressions. The switching expressions are:

$$\begin{aligned} z_2 &= a_3 b_3' + e_3 a_2 b_2' + e_3 e_2 a_1 b_1' + e_3 e_2 e_1 a_0 b_0' \\ z_1 &= e_3 e_2 e_1 e_0 \\ z_0 &= b_3 a_3' + e_3 b_2 a_2' + e_3 e_2 b_1 a_1' + e_3 e_2 e_1 b_0 a_0' \end{aligned}$$

where $e_1 = a_i' b_i' + a_i b_i$, equality function.