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 2.3
Input: $\underline{x} = (x_{n-1}, \ldots, x_0),\ x_i \in \{0, 1\}$
Output: an integer $1 \leq z \leq n-1$.
Function:
$z = |i - j|$ if $(x_i = x_j = 1)$

Exercise 2.5

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

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

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

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 2.7

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 $0 \leq z \leq 15$

$z = (x - 10)$

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 2.9

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 2.11
(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:

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

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

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

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 2.15

(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 2.17

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

$$A_c = U - A$$

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

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 2.21: 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_1x_0			
x_3x_2	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 2.23

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 2.25

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

Exercise 2.27 (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, $a * bc \neq 1$

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 2.1: Proof of distributivity for system in Exercise 2.29

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:

- (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 **??**.
- (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 2.31

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 2.33

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

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 2.37

			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 2.39

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

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

Exercise 2.41

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 2.43

(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 2.45

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

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

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

	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 2.2: Exercise 2.47(b)

Exercise 2.47

(a)

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

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

(b) See Table **??** on page **??**.

(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 **??** on page **??** 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)=$\{1, 2, 3, 4, 9, 10, 11, 12, 16, 17, 18, 23, 24, 25, 26, 31\}$

zero-set(z_1)=$\{1, 2, 5, 6, 9, 10, 13, 14, 16, 19, 20, 23, 24, 27, 28, 31\}$

zero-set(z_0)=$\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 19, 31\}$

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 2.3: Exercise 2.47(c)

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 2.51

a)

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

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:

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

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 2.53

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 2.55

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.