Chapter 5

Exercise 5.1

		z	z	
	0	1	0	1
x	0	1	0	0
			y	y

f_1

		z	z	
	-	0	1	1
x	-	0	0	1
			y	y

f_0

Exercise 5.3

(a) $E(w,x,y,z) = \prod M(1,3,4,7,10,13,14,15) = \sum m(0,2,5,6,8,9,11,12)$

z

1	0	0	1
0	1	0	1
1	0	0	0
1	1	1	0

x

w

y

(b)$E(w,x,y,z) = \sum m(0,4,5,9,11,14,15), dc(w,x,y,z) = \sum m(2,8)$

z

1	0	0	-
1	1	0	0
0	0	1	1
-	1	1	0

x

w

y

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

z

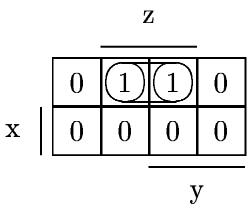
1	1	0	0
1	0	0	1

x

y

Exercise 5.5

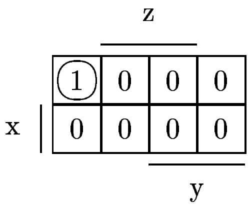
(a) More implicants than minterms



0	1	1	0
0	0	0	0

implicants: $x'yz$, $x'y'z$, and $x'z$
minterms: $x'yz$ and $x'y'z$

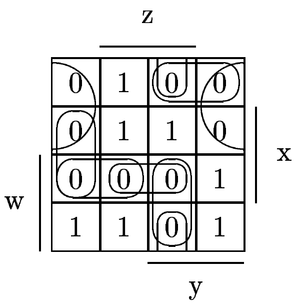
(b) Equal number of implicants and minterms



1	0	0	0
0	0	0	0

implicant: $x'y'z'$
minterm: $x'y'z'$

Exercise 5.7
$f(w, x, y, z) = one\_set(1, 5, 7, 8, 9, 10, 14)$



(a) prime implicates are:
$(w + z), (w + x + y'), (x + y' + z'), (w' + y' + z'), (w' + x' + z'), (w' + x' + y), (x' + y + z)$
(b) essential prime implicate is: $(w + z)$
(c) a minimal product of sums expression that implements $f(w, x, y, z)$ is:

$$E(w, x, y, z) = (w + z)(x + y' + z')(w' + x' + z')(x' + y + z)$$

the solution is not unique because there are other ways to cover the 0-cells (not covered by the essential prime implicate) with the same number of terms.

Exercise 5.9

To repeat the Exercise 5.8 using the Quine-McCluskey minimization method we make use of two tables. The first table is used to obtain the prime implicants and the second to select the minimum cover.

The one_set for this function is one_set= $\{2, 3, 5, 7, 11, 13\}$

Minterms	3-literal Prods	2-literal Prods	1-literal Prods
0010 N	001-		
0011 N	-101		
0101 N	-011		
	0-11		
0111 N	01-1		
1011 N			
1101 N			

The second table consider the prime-implicants obtained in the previous table:

Prime Implicants	2	3	5	7	11	13	Essential
001-	x	x					•
-101			x			x	•
-011		x			x		•
0-11		x		x			
01-1			x	x			

Based on the table, the function is represented by 3 essential terms and the minterm 7 must be covered with either $0-11$ or $01-1$. The minimal SP expressions are:

$$y = a'b'c + bc'd + b'cd + a'cd$$

or

$$y = a'b'c + bc'd + b'cd + a'bd$$

Exercise 5.11 A high-level specification for the error detector is:
Input: x is a 2-out-of-5 code represented as $\underline{x} = (a, b, c, d, e)$, where $a, b, c, d, e \in \{0, 1\}$.
Output: $f \in \{0, 1\}$.
Function:

$$f = \begin{cases} 0 & \text{if the number of 1s in the input is 2} \\ 1 & \text{otherwise} \end{cases}$$

The synthesis of this function using Quine-McCluskey minimization method is shown in Table 5.1. The obtained 3-literal products generate a Prime-implicant chart similar to the one shown in Exercise 5.10, where it was concluded that all products are essential. The minimal sum of products is:

$$\begin{aligned} f &= a'b'c'd' + abc + abd + acd + bcd + abe + ace + ade + bce + bde + cde \\ &\quad +a'b'c'e' + a'b'd'e' + a'c'd'e' + b'c'd'e' \end{aligned}$$

The gate network that implements the function f is shown in Figure 5.1. Note that the OR gate has 15 inputs, which might make a two-level implementation impractical.

Minterms	4-literal prods	3-literal prods
00000 N	0000-	--111
	000-0	-1-11
00001 N	00-00	-11-1
00010 N	0-000	-111-
00100 N	-0000	1--11
01000 N		1-1-1
10000 N	0-111 N	1-11-
	-0111 N	11--1
00111 N	01-11 N	11-1-
01011 N	-1011 N	111--
01101 N	011-1 N	
01110 N	-1101 N	
10011 N	0111- N	
10101 N	-1110 N	
10110 N	1-011 N	
11001 N	10-11 N	
11010 N	1-101 N	
11100 N	101-1 N	
	1-110 N	
01111 N	1011- N	
10111 N	11-01 N	
11011 N	110-1 N	
11101 N	11-10 N	
11110 N	1101- N	
	111-0 N	
11111 N	1110- N	
	1111- N	
	111-1 N	
	11-11 N	
	1-111 N	
	-1111 N	

Table 5.1: Quine-McCluskey table for Exercise 5.11

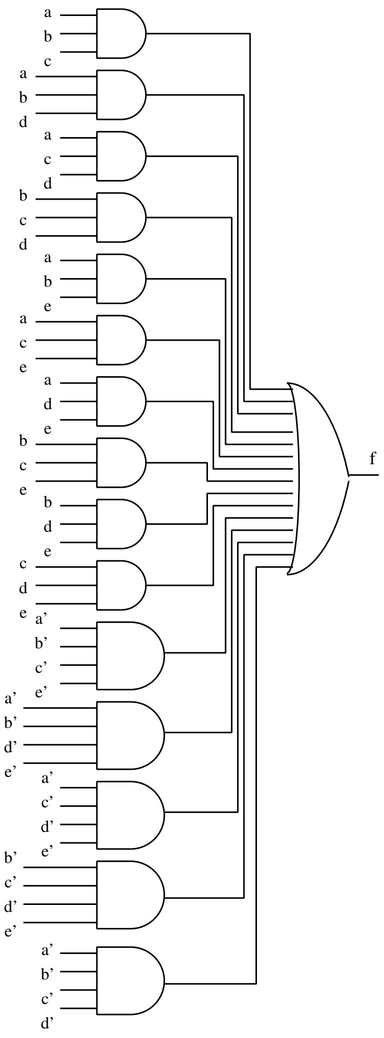


Figure 5.1: Network for Exercise 5.11

Exercise 5.13

Input: x in the range $[0, 15]$

Output: y in the range $[0, 7]$

Function: $y = x \bmod 7$

The switching functions are shown in the table

$x = (x_3x_2x_1x_0)$	$y = (y_2y_1y_0)$
0000	000
0001	001
0010	010
0011	011
0100	100
0101	101
0110	110
0111	000
1000	001
1001	010
1010	011
1011	100
1100	101
1101	110
1110	000
1111	001

From K-maps (not shown) we obtain the following minimal switching expressions

$$\begin{aligned}
y_2 &= (x_3 + x_2)(x_2' + x_1' + x_0')(x_3' + x_1' + x_0)(x_2 + x_1)\\
&= x_2x_1' + x_3x_2'x_1x_0 + x_3'x_2x_0'\\
y_1 &= (x_3 + x_1)(x_1 + x_0)(x_3 + x_2' + x_0')(x_3' + x_2' + x_1')(x_3' + x_1' + x_0')\\
&= x_3x_1'x_0 + x_3'x_2'x_1 + x_3'x_1x_0' + x_2'x_1x_0'\\
y_0 &= (x_3 + x_0)(x_3' + x_2 + x_0')(x_3' + x_1 + x_0')(x_2' + x_1' + x_0)(x_3 + x_2' + x_1')\\
&= x_3x_1'x_0' + x_3x_2'x_0' + x_3x_2x_1x_0 + x_3'x_1'x_0 + x_3'x_2'x_0
\end{aligned}$$

Using the lowest cost version of the network that produces each output, we obtain the gate network shown in Figure 5.2.

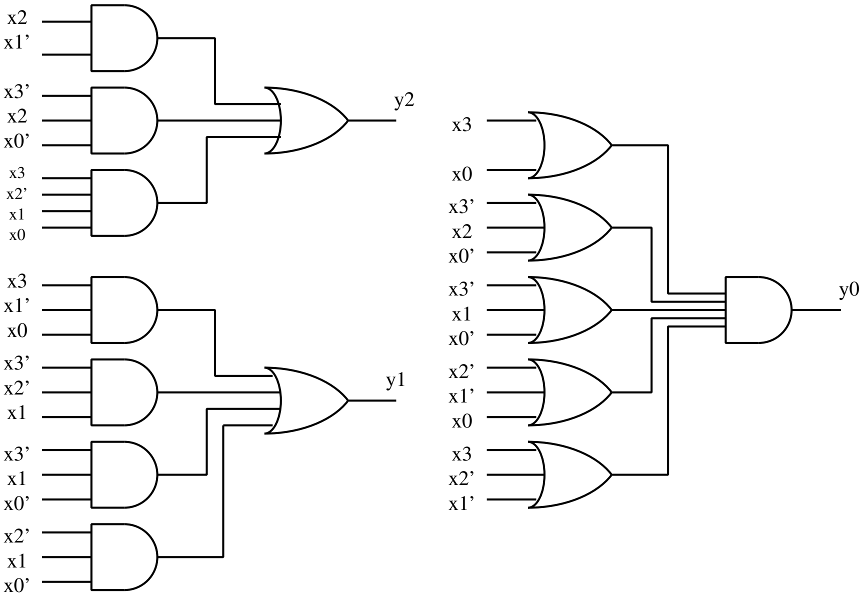


Figure 5.2: Exercise 5.13

Exercise 5.15

Input: binary code represented as $\underline{b} = (b_3b_2b_1b_0)$, where $b_i \in \{0,1\}$

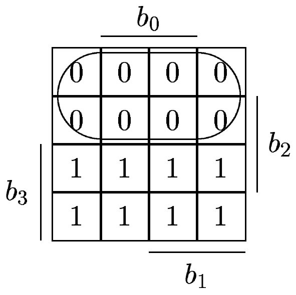
Output: Gray code represented as $\underline{g} = (g_3g_2g_1g_0)$, where $g_i \in \{0,1\}$

Function: g is the Gray code that corresponds to b. The correspondence between binary and Gray codes is shown in the following table:

Binary	Gray
$b_3b_2b_1b_0$	$g_3g_2g_1g_0$
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

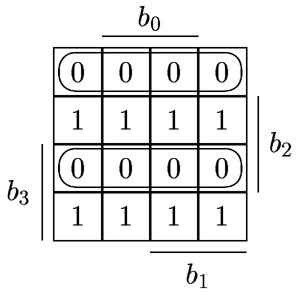
The correspoding Kmaps are as follows:

g_3



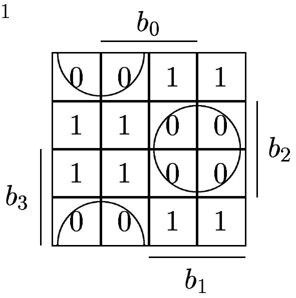
0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

g_2



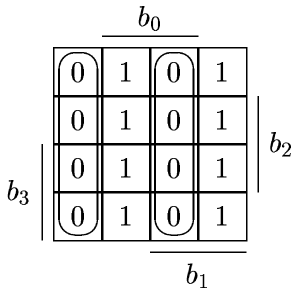
0	0	0	0
1	1	1	1
0	0	0	0
1	1	1	1

g_1



0	0	1	1
1	1	0	0
1	1	0	0
0	0	1	1

g_0



0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

To obtain a NOR-NOR network we produce the minimal product of sums:

$$\begin{aligned} g_3 &= b_3 \\ g_2 &= (b_2 + b_3)(b_2' + b_3') \\ g_1 &= (b_1 + b_2)(b_1' + b_2') \\ g_0 &= (b_0 + b_1)(b_0' + b_1') \end{aligned}$$

The gate network is presented in Figure 5.3.

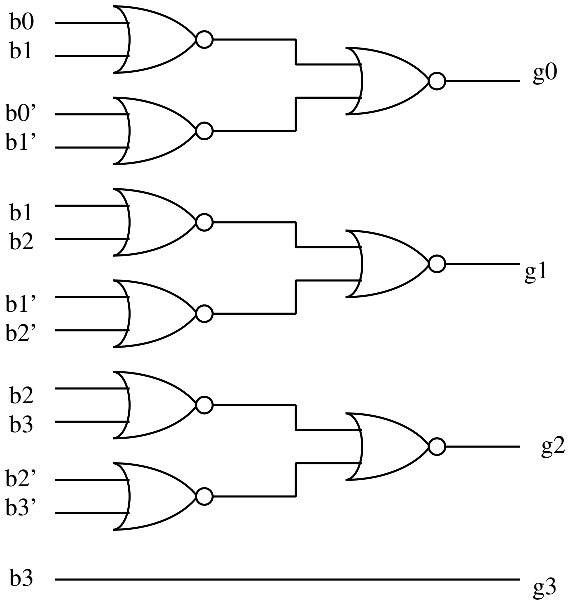


Figure 5.3: Exercise 5.15

Exercise 5.17
A high-level specification of the system is:
Input: x is an alphanumeric character coded in ASCII
Output: $z \in \{0, 1\}$
Function:

$$z = \begin{cases} 1 & \text{if } x \in \{A, B, C, D, E\} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

	x_6	x_5	x_4	x_3	x_2	x_1	x_0	z
A	1	0	0	0	0	0	1	1
B	1	0	0	0	0	1	0	1
C	1	0	0	0	0	1	1	1
D	1	0	0	0	1	0	0	1
E	1	0	0	0	1	0	1	1
otherwise								0

Since all combinations producing output 1 have $x_6x_5x_4x_3 = 1000$, we define $m = x_6x_5'x_4'x_3'$. The K-map for this function is:

$$\begin{array}{c|c|c|c|c|} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{2}{c}{x_0} & \multicolumn{1}{c}{} \\ \cline{2-5} & 0 & m & m & m \\ \cline{2-5} x_2 & m & m & 0 & 0 \\ \cline{2-5} \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{2}{c}{x_1} \end{array}$$

From the Kmap:

$$z = mx_2x_1' + mx_2'x_0 + mx_2'x_1$$

A two-level NAND network is obtained from this expression. It has 3 6-input NAND gates and one 3-input NAND gate. The description of it is:

$$NAND(NAND(x_6, x_5', x_4', x_3', x_2, x_1'), NAND(x_6, x_5', x_4', x_3', x_2', x_0), NAND(x_6, x_5', x_4', x_3', x_2', x_1))$$

For a NOR-NOR implementation we need to obtain an expression in PS form as follows:

$$z = m(x_2' + x_1')(x_2 + x_1 + x_0) = x_6 x_5' x_4' x_3'(x_2' + x_1')(x_2 + x_1 + x_0)$$

that corresponds to the following description:

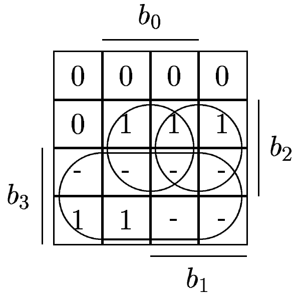
$$z = NOR(x_6', x_5, x_4, x_3, NOR(x_2', x_1'), NOR(x_2, x_1, x_0))$$

The gate networks are easily obtained from these expressions and descriptions.

Exercise 5.19 The high-level specification for this system is:
Input: b is a decimal digit, represented in BCD
Output: e is a decimal digit, represented in Excess-3 code
Function: $e = b$

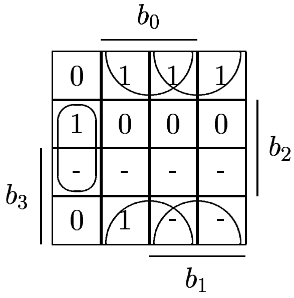
The conversion of a BCD code, represented as $\underline{b} = (b_3, b_2, b_1, b_0)$ to an Excess-3 code, represented by $\underline{e} = (e_3, e_2, e_1, e_0)$, is defined by the following K-maps:

e_3 :



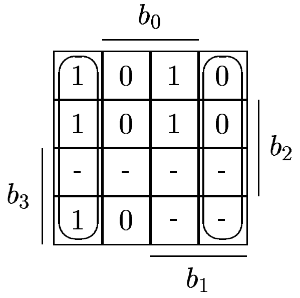
0	0	0	0
0	1	1	1
-	-	-	-
1	1	-	-

e_2 :



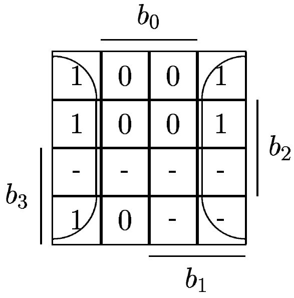
0	1	1	1
1	0	0	0
-	-	-	-
0	1	-	-

e_1 :



1	0	1	0
1	0	1	0
-	-	-	-
1	0	-	-

e_0 :



1	0	0	1
1	0	0	1
-	-	-	-
1	0	-	-

The minimal sum of products are:

$$\begin{aligned} e_3 &= b_1 b_2 + b_0 b_2 + b_3 \\ e_2 &= b_1 b_2' + b_0 b_2' + b_2 b_1' b_0' \\ e_1 &= b_1 b_0 + b_1' b_0' \\ e_0 &= b_0' \end{aligned}$$

The implementation of these expressions by a PLA is shown in figure 5.4.

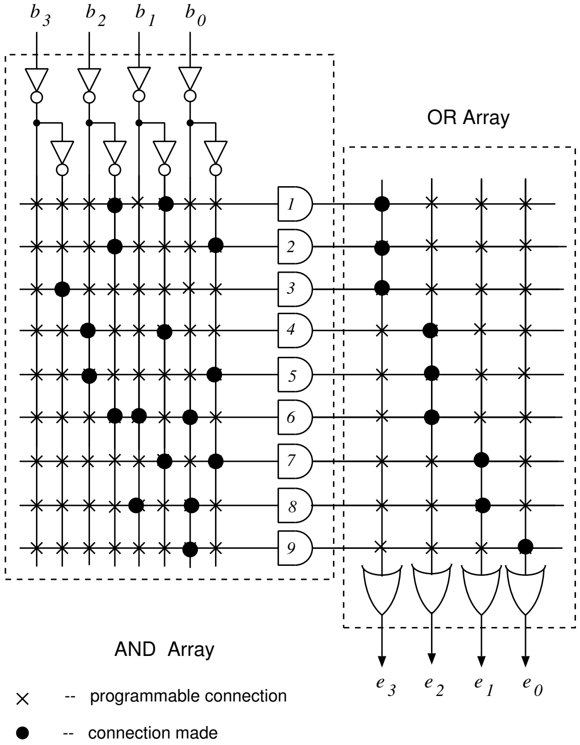


Figure 5.4: PLA implementation of a BCD to Excess-3 converter

Exercise 5.21 A high-level specification for this system is:
Input: x is a decimal digit represented in BCD
Output: $y = (y^{(1)}y^{(0)})$, where $y^{(1)}$ and $y^{(0)}$ are both BCD digits.
Function: $y = 3x$.

From this specification we define $y = (y_7, y_6, ..., y_1, y_0)$, $y_i \in \{0, 1\}$. The table for the switching functions is shown next:

$x = (x_3x_2x_1x_0)$	$y_7y_6y_5y_4$	$y_3y_2y_1y_0$
0000	0000	0000
0001	0000	0011
0010	0000	0110
0011	0000	1001
0100	0001	0010
0101	0001	0101
0110	0001	1000
0111	0010	0001
1000	0010	0100
1001	0010	0111

The implementation of this function using a PAL is shown in Figure 5.5. Output y_7 and y_6 were not mapped to the PAL since they are always zero. No minimization was performed in this solution. An *enable* input was included to activate the circuit outputs.

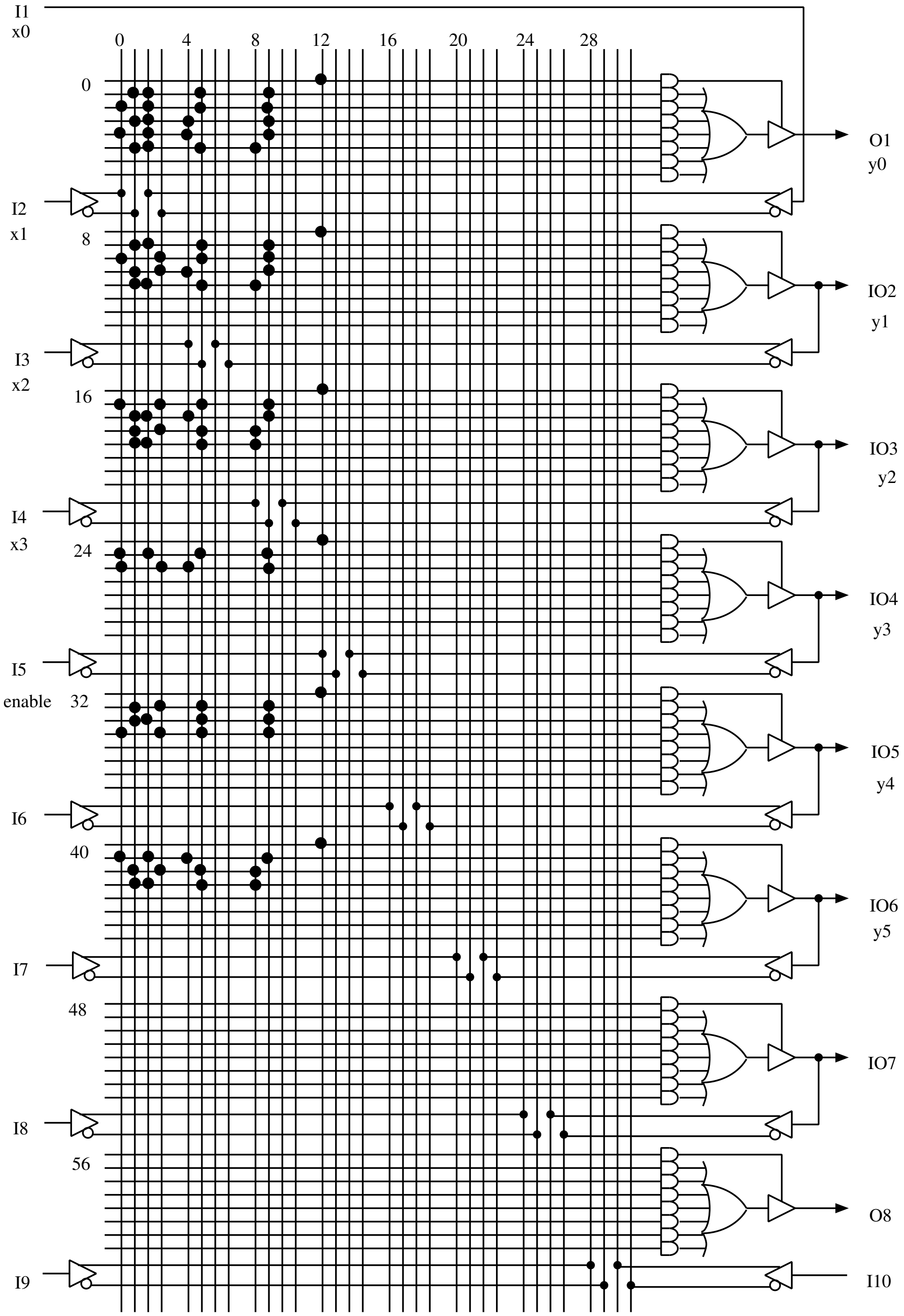


Figure 5.5: PAL implementation for Exercise 5.21