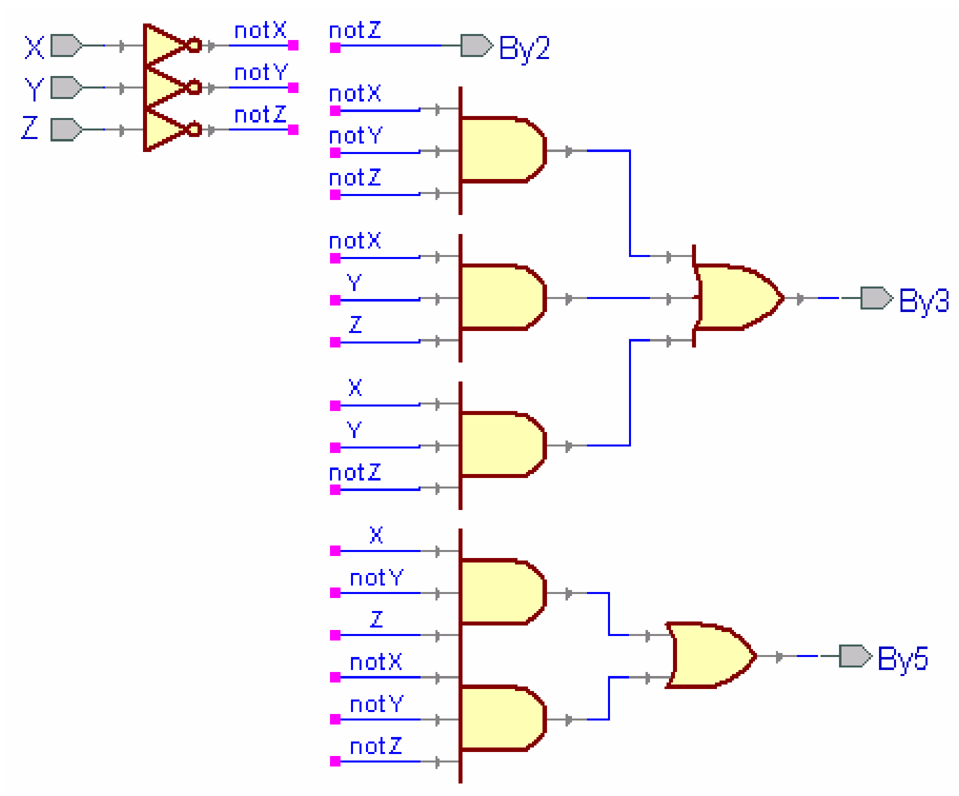
Exercise 2.1

The truth table for the functions looks as follows:

X	Y	Z	By2	By3	By5
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	0	0	0

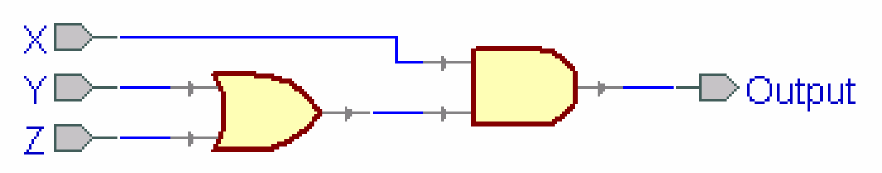
This translates into the following circuit. Please note that wires with the same name are considered to be connected.



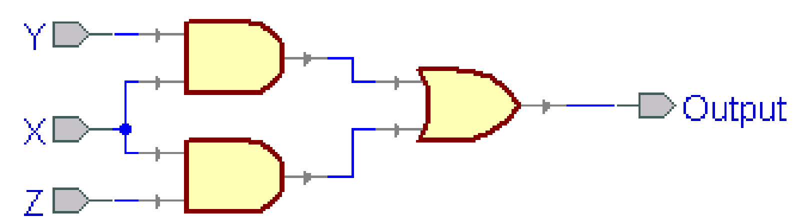
Exercise 2.2

Each of the formulas for parts a-e converts almost directly into logic gates.

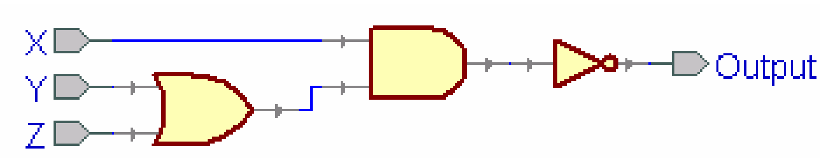
(a)



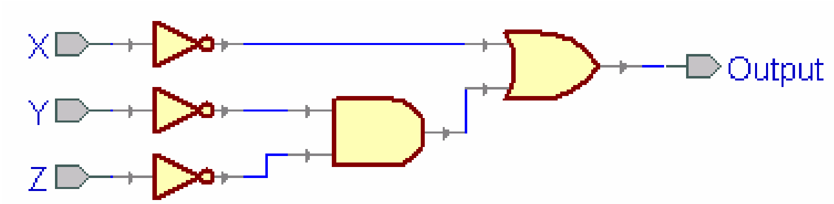
(b)



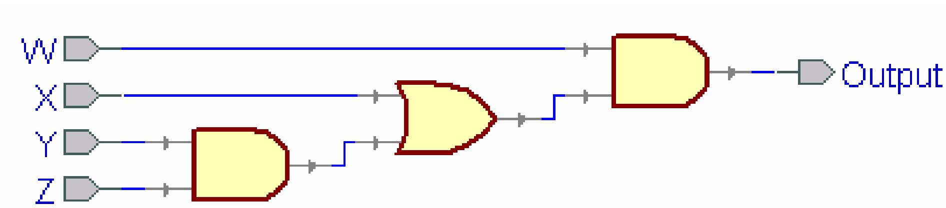
(c)



(d)



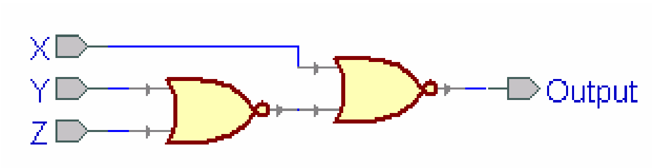
(e)



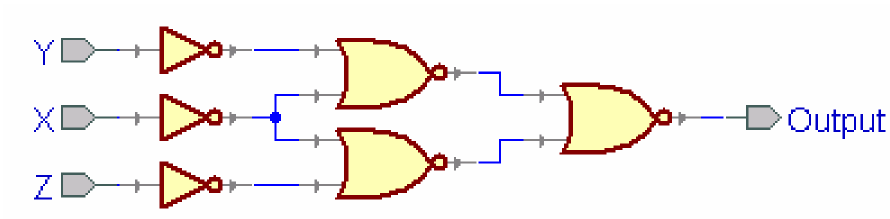
Exercise 2.3

Each of the formulas for both parts converts almost directly into logic gates.

(a)

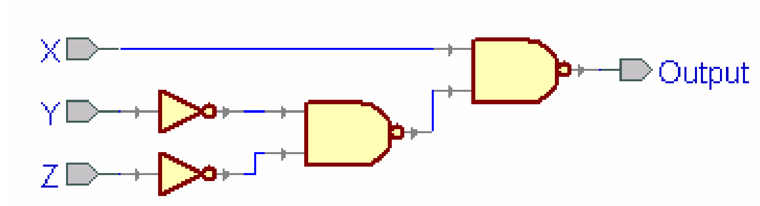


(b)

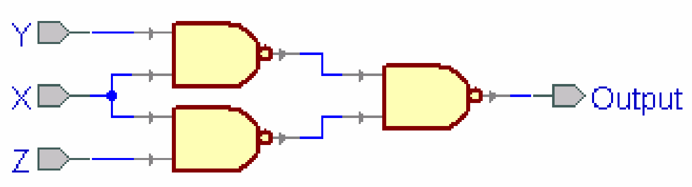


Exercise 2.4

(a) The formula converts easily to NAND and NOT gates.



(b) This solution uses DeMorgan's law to show that [[XY]' [XZ]'] ' = XY + XZ



Exercise 2.5

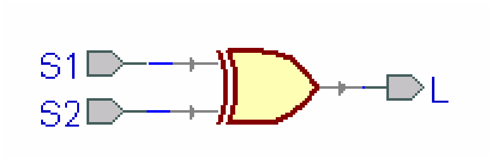
This question asks for two inputs and one output. Let the input S1 correspond to one switch, S2 correspond to the other switch, and let L correspond to the light.

Assume that when the light is installed both S1 and S2 are in the off position when L is in the off position.

What this gives is the following truth table for the circuit:

S1	S2	L
0	0	0
0	1	1
1	0	1
1	1	0

By inspection of the truth table, this function can be realized using an XOR gate as follows:



Exercise 2.6

Paragraphs that apply to the entire problem (or a single part problem) are not indented. This paragraph is “MainBody”, while lettered paragraphs below are “IndentedBody”. The title at the top is format “Heading”.

(a) $(X + Y)(X + Y') = X$

Using 8D: $X + (YY') = X$
Using 5D: $X + 0 = X$
Using 1D: $X = X$

(b) $X (X + Y) = X$

Using 8: $XX + XY = X$
Using 3D: $X + XY = X$
Using 8: $X (1 + Y) = X$
Using 2: $X (1) = X$
Using 1D: $X = X$

(c) $(X + Y') Y = XY$

Using 8: $XY + YY' = XY$
Using 5D: $XY + 0 = XY$
Using 1: $XY = XY$

(d) $(X + Y)(X' + Z) = XZ + X'Y$

Using 8: $(X + Y) X' + (X + Y) Z = XZ + X'Y$
Using 8: $XX' + YX' + XZ + YZ = XZ + X'Y$
Using 5D: $0 + YX' + XZ + YZ = XZ + X'Y$
Using 1: $X'Y + XZ + YZ = XZ + X'Y$
Using 3: $X'Y (1) + XZ (1) + YZ (1) = XZ + X'Y$
Using 5: $X'Y (Z + Z') + XZ (Y + Y') + YZ (X + X') = XZ + X'Y$
Using 8: $X'YZ + X'YZ' + XYZ + XY'Z + XYZ + X'YZ = XZ + X'Y$
Using 3: $X'YZ + X'YZ' + XYZ + XY'Z = XZ + X'Y$
Using 8: $X'Y (Z + Z') + XZ (Y + Y') = XZ + X'Y$
Using 5 and 1D: $X'Y + XZ = XZ + X'Y$

Exercise 2.7

(a) By definition: $(X + Y)^D = XY$

By definition: $(XY)^D = X + Y$

(b) Using 12: $[(X + Y)']^D = [X'Y']^D$

Using part (a): $[X'Y']^D = X' + Y'$

Using 12D: $X' + Y' = (XY)'$

This that NOR and NAND are duals of each other, since each function has one dual by definition, showing the direction from NOR to NAND is sufficient.

(c) Using part (a): $[XY' + X'Y]^D = (X + Y')(X' + Y)$

Using 8: $(X + Y')(X' + Y) = X'(X + Y') + Y (X + Y')$
Using 8: $X'(X + Y') + Y (X + Y') = XX' + X'Y' + XY + YY'$
Using 5D: $XX' + X'Y' + XY + YY' = X'Y' + XY$

The first step may appear kind of confusing since it uses the fact that AND and OR are duals three times. Again since each function has only one dual, it is sufficient to show XNOR is the dual of XOR to get XOR is the dual of XNOR.

(d) $(XY' + X'Y)' = (X'Y' + XY)$

Using 12: $(XY')' (X'Y)' = (X'Y' + XY)$
Using 12D: $(X' + Y) (X + Y') = X'Y' + XY$
Using 8: $(X' + Y) X + (X' + Y) Y' = X'Y' + XY$
Using 8: $X'X + XY + X'Y' + YY' = X'Y' + XY$
Using 5D: $X'Y' + XY = X'Y' + XY$

Exercise 2.8

The objective for proving using the truth tables is to show that when you apply the functions to every possible input, they agree on all of the inputs. This is the case in all of the parts below.

(a)

X	Y	Z	XY + YZ + XZ’	YZ + XZ’
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

(b)

A	B	(A + B’) B	AB
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

(c)

A	B	C	(A + B) (A’ + C)	AC + A’B
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(d)

A	B	C	ABC + A’BC + A’B’C + A’BC’ + A’B’C’	BC + A’B’ + A’C’
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Exercise 2.9

	$BC + A'B' + A'C' = ABC + A'$
Using 5:	$(A + A') BC + A'B' + A'C' = ABC + A'$
Using 8:	$ABC + A'BC + A'B' + A'C' = ABC + A'$
Using 8:	$ABC + A' (B' + C' + BC) = ABC + A'$
Using 5:	$ABC + A' (B'(C + C') + C' (B + B') + BC) = ABC + A'$
Using 8:	$ABC + A' (B'C + B'C' + BC' + B'C' + BC) = ABC + A'$
Using 8:	$ABC + A' [B (C + C') + B' (C + C')] = ABC + A'$
Using 5:	$ABC + A' [B + B'] = ABC + A'$
Using 5:	$ABC + A' = ABC + A'$

Exercise 2.10

(a) $[A (B + CD)]'$
$= A' + (B + CD)'$
$= A' + [B' (CD)']$
$= A' + [B' (C' + D')]$

(b) $[ABC + B (C' + D')] '$
$= (ABC)' [B (C' + D')]'$
$= (A' + B' + C') [B' + (C' + D')']$
$= (A' + B' + C') [B' + CD]$

(c) $(X' + Y')' = XY$

(d) $(X + YZ')'$
$= X' (YZ')'$
$= X' (Y' + Z)$

(e) $[(X + Y) Z]'$
$= (X + Y)' + Z'$
$= X'Y' + Z'$

(f) $[X + (YZ)']' = X'YZ$

(g) $[X (Y + ZW' + V'S)]'$
$= X' + (Y + ZW' + V'S')'$
$= X' + Y' (ZW')' (V'S)'$
$= X' + Y' (Z' + W) (V + S')$

Exercise 2.11

Note these solutions only take the complements, and do not simplify any further than using DeMorgan's laws.

(a) $f (A, B, C, D) = [A + (BCD)'][(AD)' + B (C' + A)]$

$$f ' (A, B, C, D) = [[A + (BCD)'][(AD)' + B (C' + A)]]'$$

$$= [A + (BCD)']' + [(AD)' + B (C' + A)]'$$

$$= A'BCD + AD [B (C' + A)]'$$

$$= A'BCD + AD [B' + (C' + A)']$$

$$= A'BCD + AD [B' + A'C]$$

(b) $f (A, B, C, D) = A'BC + (A' + B + D)(ABD' + B')$

$$f ' (A, B, C, D) = [A'BC + (A' + B + D)(ABD' + B')]'$$

$$= (A'BC)' [(A' + B + D)(ABD' + B')]'$$

$$= (A + B' + C')[(A' + B + D)' + (ABD' + B')']$$

$$= (A + B' + C')[AB'D' + B (ABD')']$$

$$= (A + B' + C')[AB'D' + B (A' + B' + D)']$$

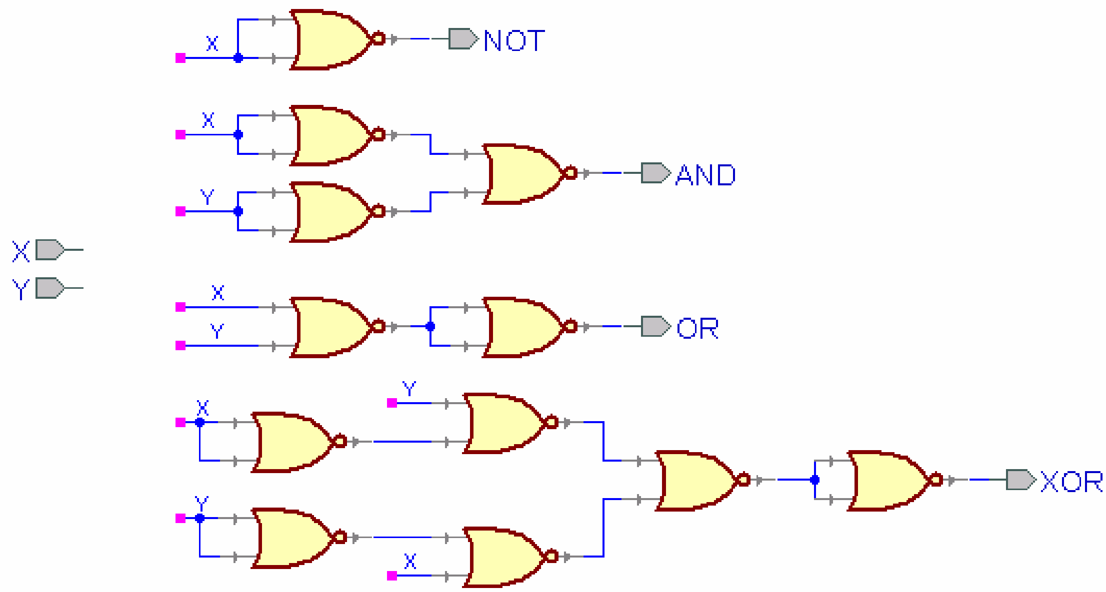
Exercise 2.12

[[X (XY)’]’ [Y (XY)’]’]’ = XY’ + X’Y

Using 12: [X (XY)’] + [Y (XY)’] = XY’ + X’Y
Using 12D: [X (X’ + Y’)] + [Y (X’ + Y’)] = XY’ + X’Y
Using 11: XY’ + X’Y = XY’ + X’Y

Exercise 2.13

The following figure demonstrates the implementation of the NOT, AND, OR, and XOR gates using the two-input NOR gate. The basis for many of these functions uses a combination of DeMorgan's laws and other Boolean Algebra simplifications.



Similarly NAND is a universal logic element since NAND and NOR are duals of each other, any function that NOR implements, NAND can implement the dual of that function. Since XOR and XNOR are complements of each other, combining NOT and XNOR can be implemented with NAND gates.

XOR is not a universal logic element, since it cannot implement NOT just using the inputs X and Y without leaving an input disconnected.

Exercise 2.14

Here define H_{1S} to be the sum output and H_{1C} to be the carryout of the first half adder that takes inputs A and B. H_{2S} is the sum output and H_{2C} is the carryout of the second half adder that takes H_{2S} and C_{in} as inputs. The output of H_{2S} is equivalent to S for the full adder, and OR is equivalent to C_{out} of the full adder.

A	B	C_{in}	H_{1S}	H_{1C}	H_{2S}	H_{2C}	OR	S	C_{out}
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0
0	1	0	1	0	1	0	0	1	0
0	1	1	1	0	0	1	1	1	1
1	0	0	1	0	1	0	0	1	0
1	0	1	1	0	0	1	1	0	1
1	1	0	0	1	0	0	1	0	1
1	1	1	0	1	1	0	1	1	1

Exercise 2.15

The waveform behavior will be different because in the direct implementation from Boolean equations the equations can be simplified to contain at most two gate delays assuming that there are no limitations on the fanout of each logic gate. In the case of the full adder, it takes two gate delays for each half-adder, plus an additional gate delay for the OR-gate. Thus the direct equations will lead to a much faster circuit than the hierarchical form.

Exercise 2.16

Let the variables A_0, B_0 represent the least significant bit of the two bit inputs and A and B, and let A_1, B_1 be the most significant bits respectively.

$S_0 = A_0B_0' + A_0'B_0$

$C_{out0} = A_0B_0$

$S_1 = A_1B_1C_{out0} + A_1B_1'C_{out0}' + A_1'B_1C_{out0}' + A_1'B_1'C_{out0}$

$= A_1B_1\, A_0B_0 + A_1B_1'\,(A_0B_0)' + A_1'B_1\,(A_0B_0)' + A_1'B_1'A_0B_0$

$= A_1B_1\, A_0B_0 + A_1B_1'\,(A_0'+B_0') + A_1'B_1(A_0' + B_0')' + A_1'B_1'A_0B_0$

$= A_0B_0A_1B_1 + A_0'A_1B_1' + B_0'A_1B_1' + A_0'A_1'B_1 + B_0'A_1'B_1 + A_0B_0A_1'B_1'$

$C_{out1} = B_1C_{out0} + A_1C_{out0} + A_1B_1$

$= B_1A_0B_0 + A_1A_0B_0 + A_1B_1$

Exercise 2.17

(a) $f(X, Y) = XY + XY'$

Using 8: $XY + XY' = X(Y + Y')$
Using 5: $XY + XY' = X$

(b) $f(X, Y) = (X + Y)(X + Y')$

Using 8: $f(X, Y) = X + XY' + XY + YY'$
Using 5D: $f(X, Y) = X + XY' + XY$
Using 8: $f(X, Y) = X + X(Y' + Y)$
Using 5: $f(X, Y) = X + X$
Using 3: $f(X, Y) = X$

(c) $f(X, Y, Z) = Y'Z + X'YZ + XYZ$

Using 8: $f(X, Y, Z) = Z(Y' + X'Y + XY)$
Using 8: $f(X, Y, Z) = Z(Y' + Y(X' + X))$
Using 5: $f(X, Y, Z) = Z(Y' + Y)$
Using 5: $f(X, Y, Z) = Z$

(d) $f(X, Y, Z) = (X + Y)(X' + Y + Z)(X' + Y + Z')$

Using 8: $f(X, Y, Z) = (XX' + XY + XZ + X'Y + Y + YZ)(X' + Y + Z')$
Using 5D and 10: $f(X, Y, Z) = (XZ + Y)(X' + Y + Z')$
Using 8: $f(X, Y, Z) = XX'Z + XYZ + XZZ' + X'Y + Y + YZ'$
Using 5D and 2D: $f(X, Y, Z) = XYZ + X'Y + Y + YZ'$
Using 10: $f(X, Y, Z) = Y$

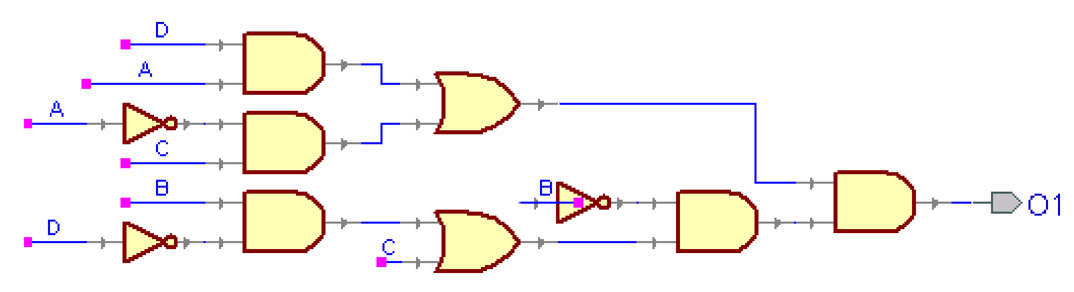
(e) $f(W, X, Y, Z) = X + XYZ + X'YZ + X'Y + WX + WX'$

Using 8: $f(W, X, Y, Z) = X(1 + W) + YZ(X + X') + X'Y + X'W$
Using 2: $f(W, X, Y, Z) = X + YZ(X + X') + X'Y + X'W$
Using 5: $f(W, X, Y, Z) = X + YZ + X'Y + X'W$

Exercise 2.18

Paragraphs that apply to the entire problem (or a single part problem) are not indented. This paragraph is "MainBody", while lettered paragraphs below are "IndentedBody". The title at the top is format "Heading".

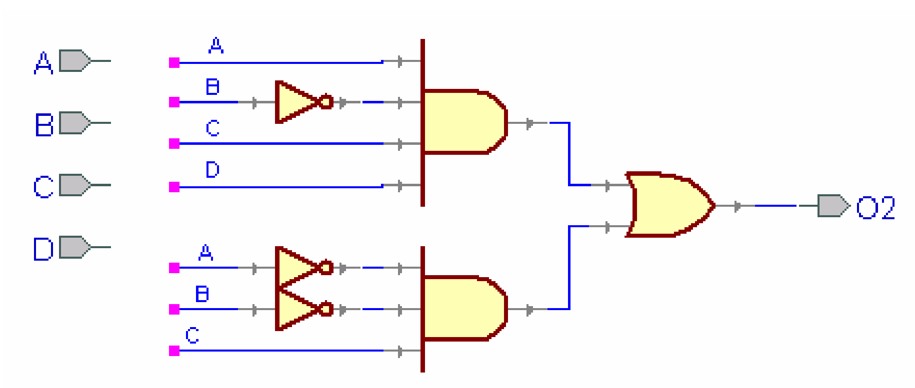
(a)



(b) Using 8: (AD + A'C)[B' (C + BD')] = (AD + A'C) [B'C + B'BD']

Using 3: (AD + A'C)[B'C + B'BD] = (AD + A'C) B'C

Using 8: (AD + A'C) B'C = AB'CD + A'B'C



<u>Exercise 2.19</u>

Each of the following solutions is put into their respected canonical form.

(a) A’B’C’D’ + A’B’C’D + A’B’CD’ + A’BCD + AB’C’D’ + AB’C’D + AB’CD’ + ABCD

(b) (A’ + B’ + C + D) (A’ + B + C’ + D’) (A’ + B + C’ + D) (A’ + B + C + D’) (A + B’ + C + D) (A + B + C’ + D’) (A + B + C’ + D) (A + B + C + D’)

(c) A’B’CD + A’BC’D’ + A’BC’D + A’BCD’ + AB’CD + ABC’D’+ ABC’D + ABCD’

(d) (A’ + B’ + C’ + D’) (A’ + B’ + C’ + D) (A’ + B’ + C + D’) (A’ + B + C + D) (A + B’ + C’ + D’) (A + B’ + C’ + D) (A + B’ + C + D’) (A + B + C + D)

Exercise 2.20

Each of the following solutions is put into their respected canonical form.

(a) A’B’C’D + A’B’CD’ + A’B’CD + A’BC’D + AB’C’D’ + ABC’D

(b) (A’ + B’ + C’ + D’) (A’ + B + C’ + D’) (A’ + B + C + D’) (A’ + B + C + D)
(A + B’ + C’ + D) (A + B’ + C + D’) (A + B’ + C + D) (A + B + C’ + D’)
(A + B + C + D’) (A + B + C + D)

(c) A’B’C’D’ + A’BC’D’ + A’BCD’ + A’BCD + AB’C’D + AB’CD’ + AB’CD +
ABC’D’ + ABCD’ + ABCD

(d) (A’ + B’ + C’ + D) (A’ + B’ + C + D’) (A’ + B’ + C + D) (A’ + B + C’ + D)
(A + B + C’ + D)

Exercise 2.21

(a) $f(A, B, C) = AB + B'C' + AC'$

$= ABC' + ABC + A'B'C' + AB'C' + AB'C' + ABC'$

$= A'B'C' + AB'C' + ABC' + ABC$

$= \sum m(0, 4, 6, 7)$

(b) $M(0, 4, 6, 7) = (A' + B' + C')(A + B' + C')(A + B + C')(A + B + C)$

Exercise 2.22

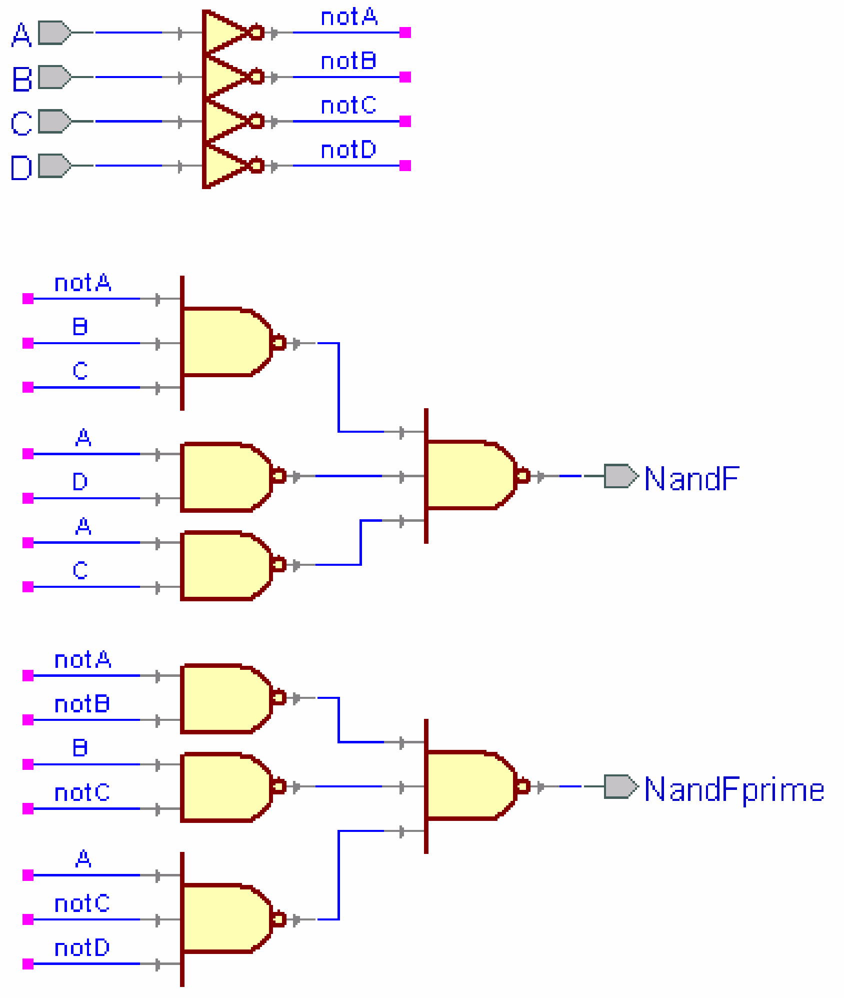
(a) $F(A, B, C, D) = \prod M(0, 1, 2, 3, 4, 5, 8, 12)$

(b) $F(A, B, C, D) = (A' + B' + C' + D')(A' + B' + C' + D)(A' + B' + C + D')$
$(A' + B' + C + D)(A' + B + C' + D')(A' + B + C' + D)$
$(A + B' + C' + D')(A + B + C' + D')$
$= (A'+B'+C')(A'+B'+C)(A'+B+C')(A+C'+D')$
$= (A'+B')(A'+B+C')(A+C'+D')$

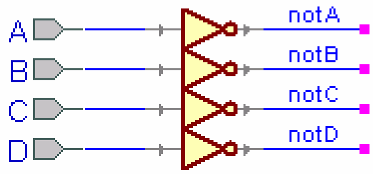
(c) $F'(A, B, C, D) = \prod M(6, 7, 9, 10, 11, 13, 14, 15)$
$= (A' + B + C + D')(A' + B + C + D)(A + B' + C' + D)$
$(A + B' + C + D')(A + B' + C + D)(A + B + C' + D)$
$(A + B + C + D')(A + B + C + D)$
$= (A' + B + C)(A + C' + D)(A + C)$

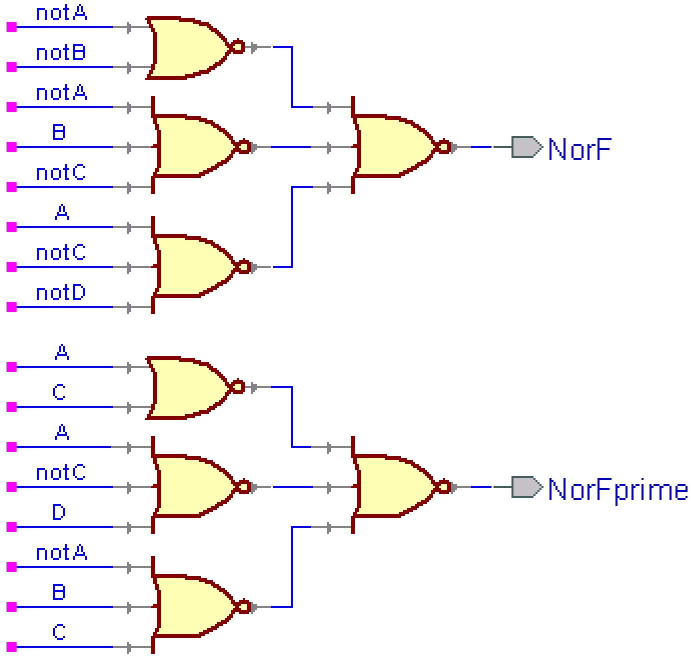
(d) $F'(A,B,C,D) = \sum M(0, 1, 2, 3, 4, 5, 8, 12)$
$= A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BC'D$
$+ AB'C'D' + ABC'D'$
$= A'B'C' + A'B'C + A'BC' + AC'D'$
$= A'B' + BC' + AC'D'$

(e) Note wires with the same name are considered to be connected to each other. Using the canonical sum-of-products formula and DeMorgen's laws, the transformation into NAND gates is fairly straightforward.



(f) Using the product-of-sums form, transformation to NOR gates is fairly straightforward.





Exercise 2.23

(a) Since the function is given in minimized product-of-sums form, it is easiest to convert it first to canonical product-of-sums form. By taking the complement of the canonical form, and converting the ∏M to ∑m the canonical sum-of-products is reached.

$$F(W, X, Y, Z) = (W + X' + Y') (W' + Z') (W + Y)$$

$$= (W' + X' + Y' + Z') (W' + X' + Y + Z') (W' + X + Y' + Z') (W' + X + Y + Z') (W + X' + Y' + Z') (W + X' + Y' + Z) (W + X' + Y + Z') (W + X' + Y + Z) (W + X + Y + Z') (W + X + Y + Z)$$

$$= \prod M (0, 2, 4, 6, 8, 9, 10, 11, 14, 15)$$

$$= \sum m (1, 3, 5, 7, 12, 13)$$

(b) $\sum m (1, 3, 5, 7, 12, 13) = W'X'Y'Z + W'X'YZ + W'XY'Z + W'XYZ + WXY'Z' + WXY'Z$

$$= W'X'Z + W'XZ + WXY'$$

$$= W'Z + WXY'$$

(c) $\sum m(0, 2, 4, 6, 8, 9, 10, 11, 14, 15) = W'X'Y'Z' + W'X'YZ' + W'XY'Z' + W'XYZ' + WX'Y'Z' + WX'Y'Z + WX'YZ' + WX'YZ + WXYZ' + WXYZ$

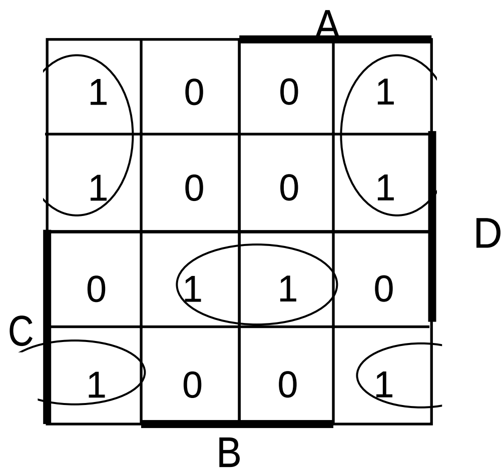
$$= W'X'Z' + W'XZ' + WX'Y' + WX'Y + WXY$$

$$= W'Z' + WX' + WY$$

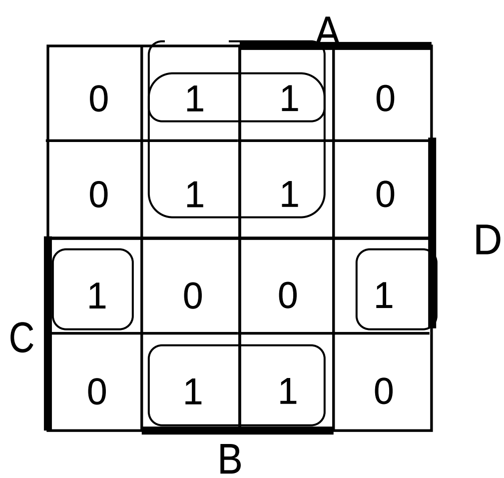
(d) $\prod M (1, 3, 5, 7, 12, 13) = (W' + X' + Y' + Z) (W' + X' + Y + Z) (W' + X + Y' + Z) (W' + X + Y + Z) (W+ X + Y' + Z') (W + X + Y' + Z)$

Exercise 2.24

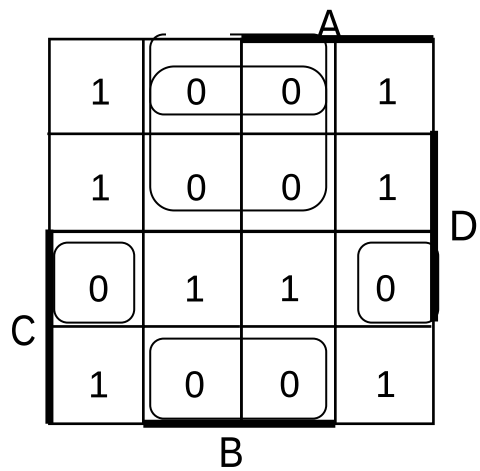
(a) F(A, B, C, D) = B'C' + BCD + B'CD'



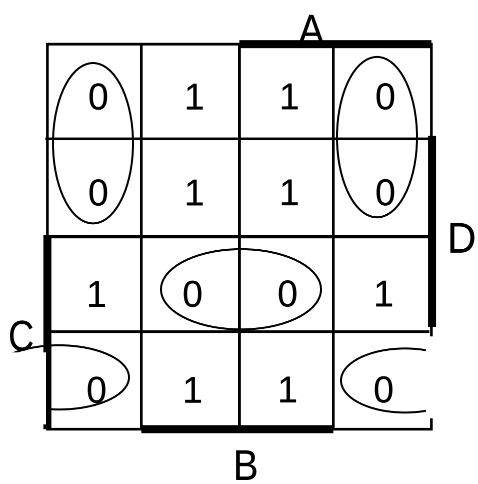
F'(A, B, C, D) = BD' + BC' + B'CD



(b) $F(A, B, C, D) = (B + D')(B + C')(B + C + D)$



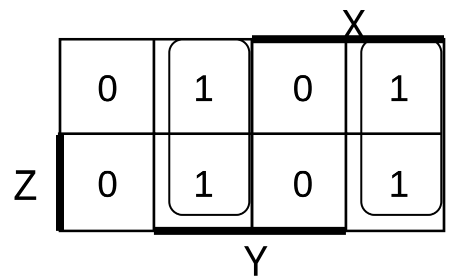
$F'(A, B, C, D) = (B' + C')(B + C + D)(B' + C + D')$



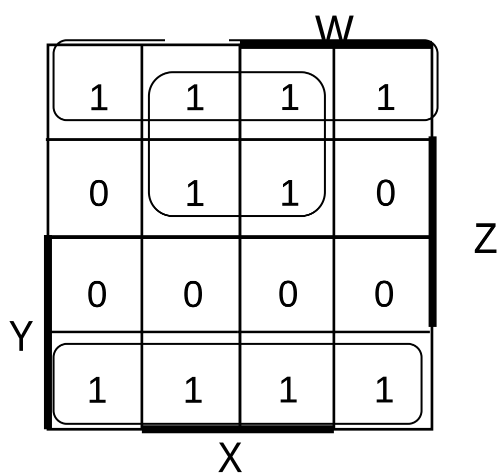
Exercise 2.25

Paragraphs that apply to the entire problem (or a single part problem) are not indented. This paragraph is “MainBody”, while lettered paragraphs below are “IndentedBody”. The title at the top is format “Heading”.

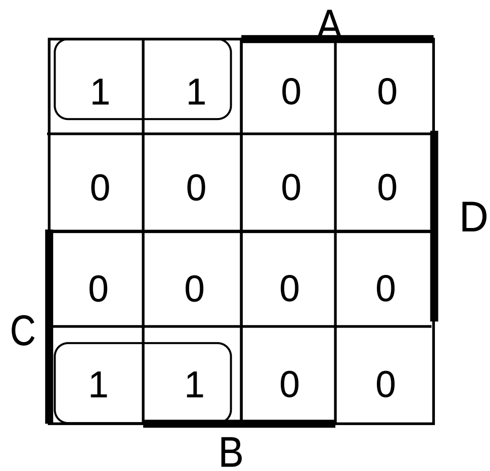
(a) f(X, Y, Z) = AB + AB’



(b) f(W, X, Y, Z) = Z’ + XY’

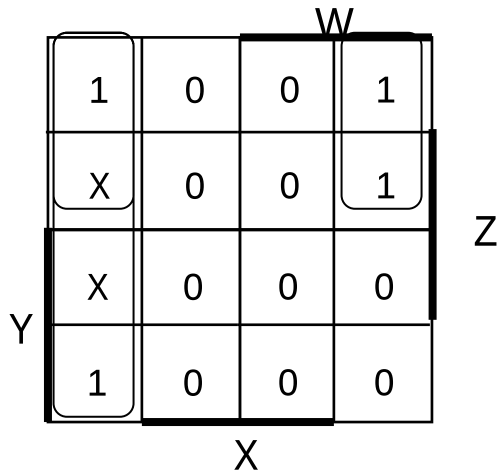


(c) $f(A, B, C, D) = A'D'$

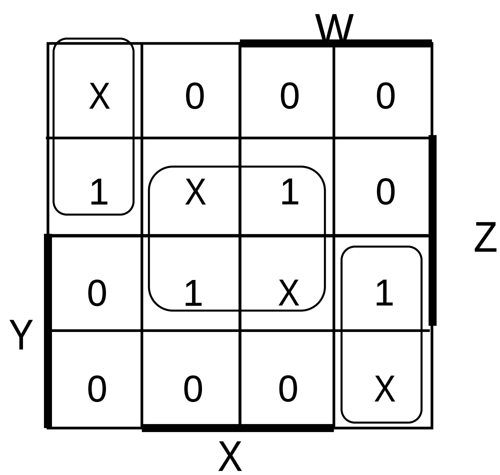


Exercise 2.26

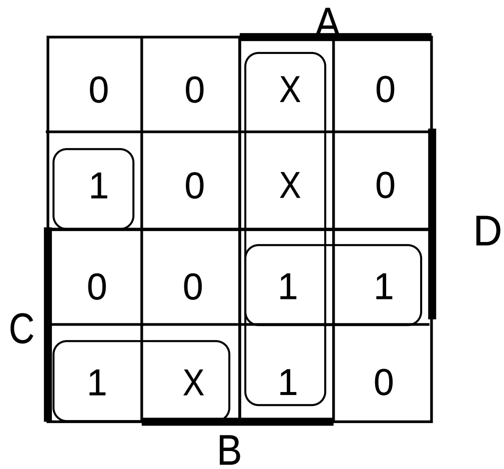
(a) f(W, X, Y, Z) = X’W’ + X’Y’



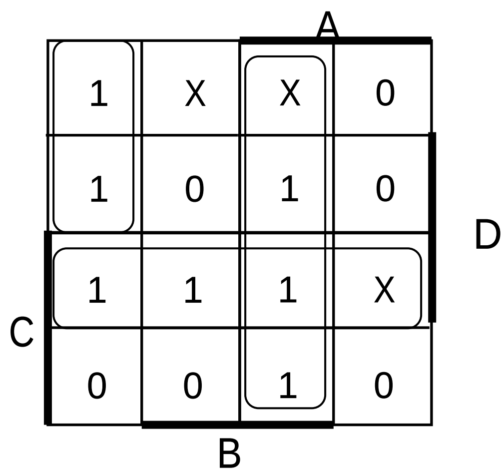
(b) f(W, X, Y, Z) = XZ + WX’Y + W’X’Y’



(c) f(A, B, C, D) = A’B’C’D + A’CD’ + ACD + AB

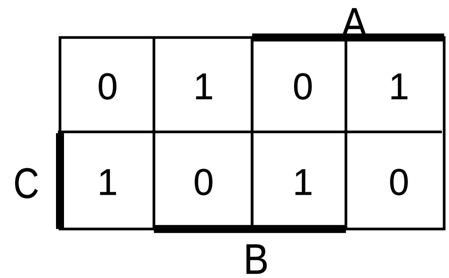


(d) f(A, B, C, D) = A’B’C’ + CD + AB

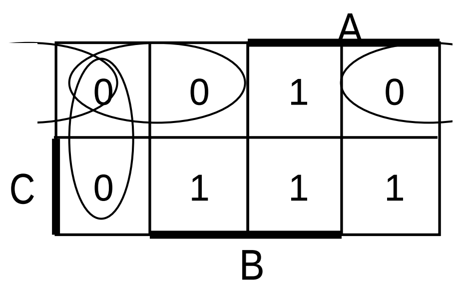


Exercise 2.27

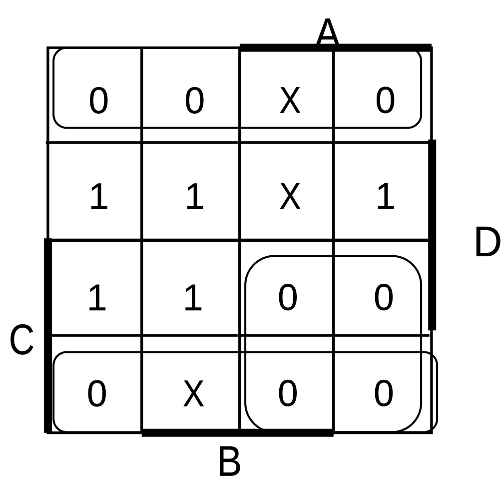
(a) f(A, B, C) = (A’ + B’ + C’) (A’ + B + C) (A + B + C) (A + B’ + C)



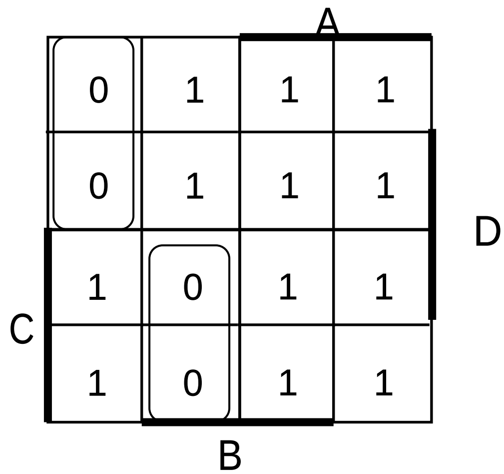
(b) f(A, B, C) = (A’ + B’) (A’ + C’) (A + B’)



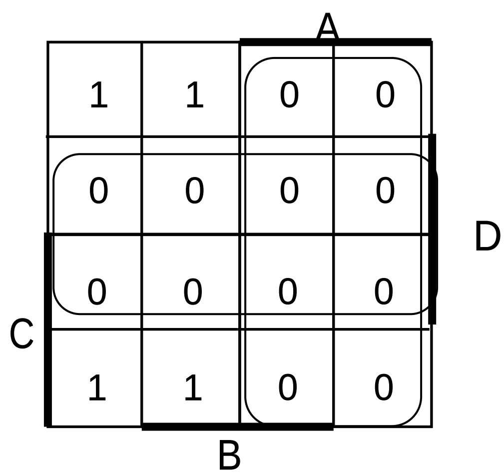
(c) f(A, B, C, D) = D’ (A + C)



(d) f(A, B, C, D) = (A’ + B + C) (A’ + B’ + C’)



(e) f(A, B, C, D) = AD



Exercise 2.28

(a) In this case S cannot be simplified any further.

$S(A, B, C) = A'B'C + A'BC' + AB'C' + ABC$

(b) $F(A, B, C) = A'B'C' + A'B'C + AB'C' + AB'C + ABC' + ABC$

Using 8: $= A'B'(C' + C) + AB' (C' + C) + AB (C' + C)$

Using 5: $= A'B' + AB' + AB$

Using 3: $= A'B' + AB' + AB' + AB$

Using 8: $= (A' + A) B' + A (B' + B)$

Using 5: $= B' + A$

(c) $G(A, B, C, D) = A'B'C'D' + A'B'CD' + AB'C'D' + AB'CD + ABC'D' + ABCD$

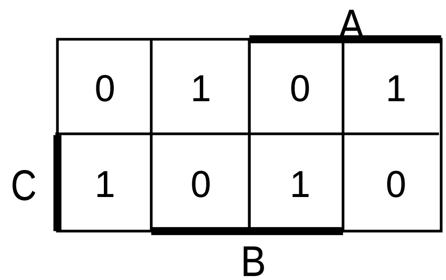
Using 8: $= A'B'(C' + C)D' + A(B + B')C'D' + A(B + B')CD$

Using 5: $= A'B'D' + AC'D' + ACD$

Exercise 2.29

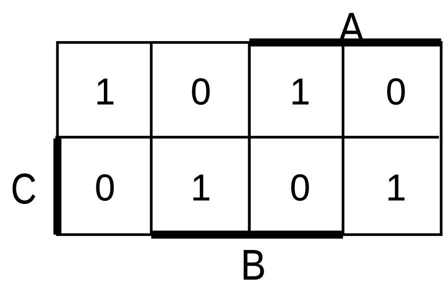
Solution for part 2.28 part (a). Note since neither the function nor its complement can be simplified in this case, the Karnaugh maps will be exactly the same for (a) and (b), and also for (c) and (d). Because of this, only the maps for part (a) and (c) are shown.

(a) S(A, B, C) = A’B’C + AB’C + ABC + AB’C’



(b) S(A, B, C) = (A’ + B’ + C’) (A’ + B + C) (A + B + C) (A + B’ + C)

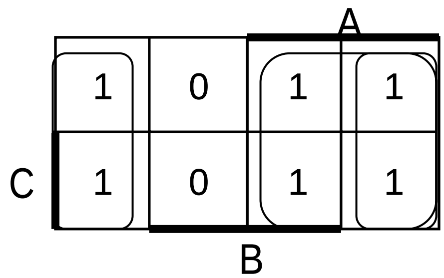
(c) S’(A, B, C) = A’B’C’ + A’BC + ABC’ + AB’C



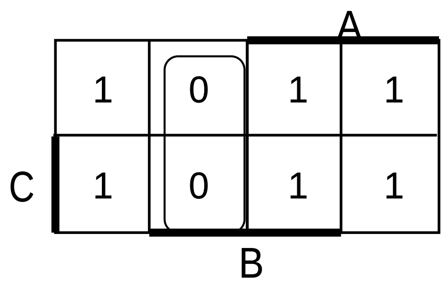
(d) S’(A, B, C) = (A’ + B’ + C) (A’ + B + C’) (A + B + C) (A + B’ + C’)

Solution for part 2.28 part (b):

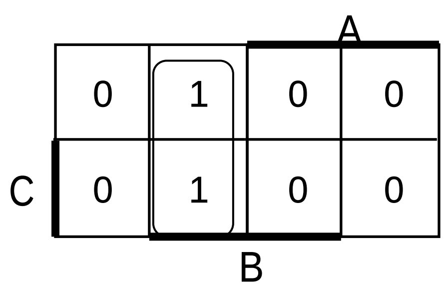
(a) F(A, B, C) = B’ + A



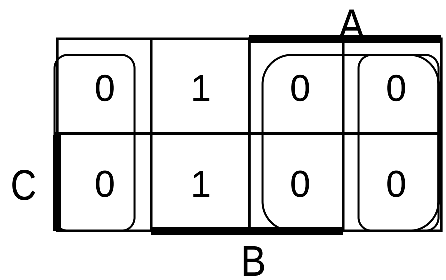
(b) F(A, B, C) = BA’



(c) F’(A, B, C) = B + A’

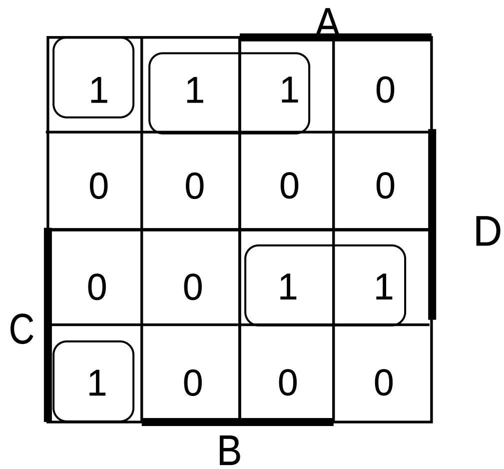


(d) F’(A, B, C) = AB’

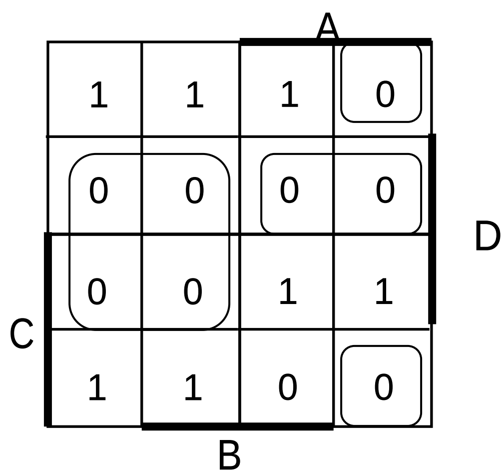


Solution for part 2.28 part (c):

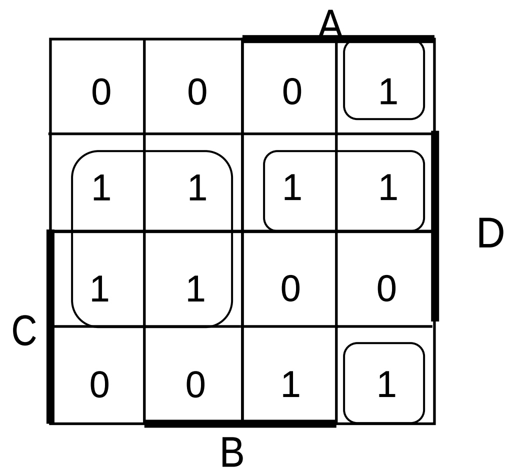
(a) G(A, B, C, D) = A’BD’ + BC’D’ + ACD



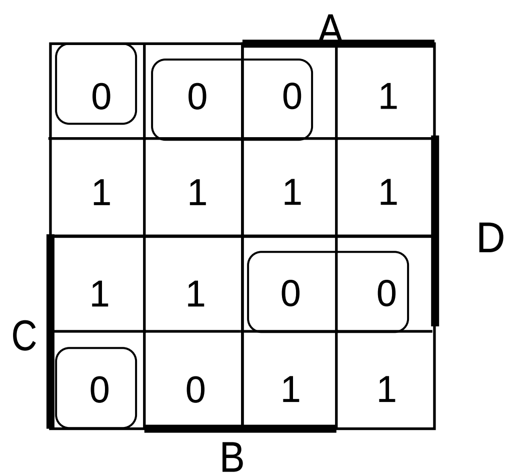
(b) G(A, B, C, D) = (A’ + D) (A + C’ + D) (A + B’ + D’)



(c) G’(A, B, C, D) = A’D + AC’D + AB’C’



(d) G’(A, B, C, D) = (A’ + B + D’) (B + C’ + D’) (A + C + D)



Exercise 2.30

(a) W(A, B, C) = A'BC' + A'BC + AB'C' + AB'C

Using 8: = A'B(C' + C) + AB' (C + C')
Using 5: = A'B + AB' = A $\otimes$ B

(b) X(A, B, C) = A'B'C' + A'BC + AB'C' + ABC

Using 8: = (A' + A) B'C' + (A' + A) BC
Using 5: = B'C' + BC

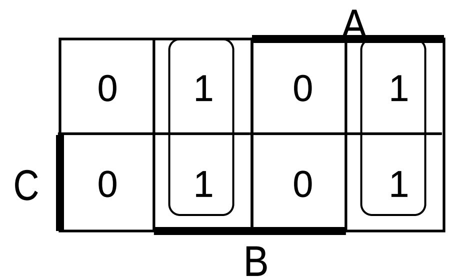
(c) Y(A, B, C, D) = A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + AB'C'D' + AB'CD'

Using 8: = A'B'C' (D + D') + AB' (C + C') D' + A'B'C (D + D')
Using 5: = A'B'C' + AB'D' + A'B'C
Using 8: = A'B'(C + C') + AB'D'
Using 5: = A'B' + AB'D'
Using 2: = A'B' (1 + D') + AB'D'
Using 8: = A'B' + A'B'D' + AB'D'
Using 8: = A'B' + (A' + A) B'D'
Using 5: = A'B' + B'D'

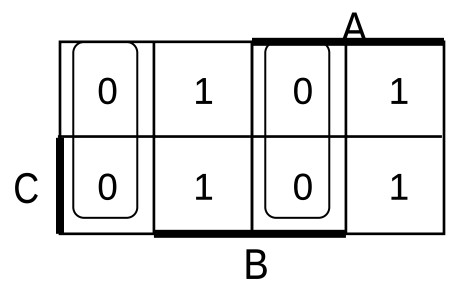
Exercise 2.31

Solution for part 2.30 part (a).

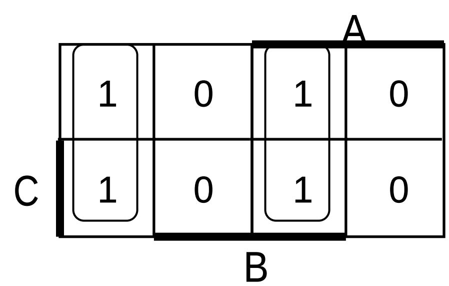
(a) W(A, B, C) = A'B + AB'



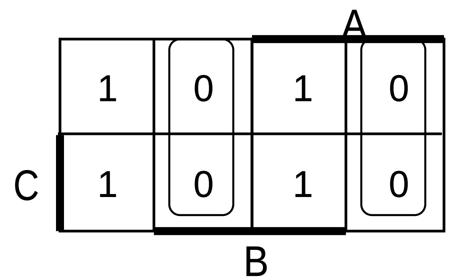
(b) W(A, B, C) = (A + B) (A' + B')



(c) W'(A, B, C) = A'B' + AB

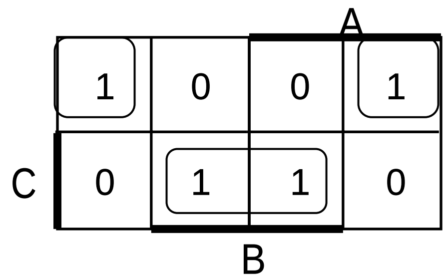


(d) W'(A, B, C) = (A' + B' + C) (A' + B + C') (A + B + C) (A + B' + C')

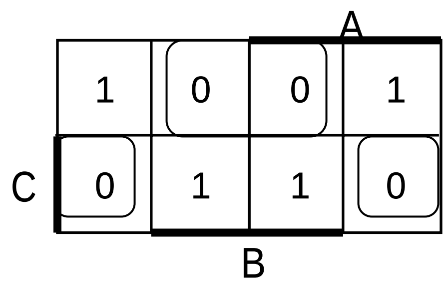


Solution for part 2.30 part (b).

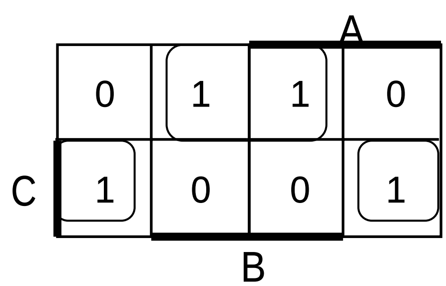
(a) X(A, B, C) = BC+B'C'



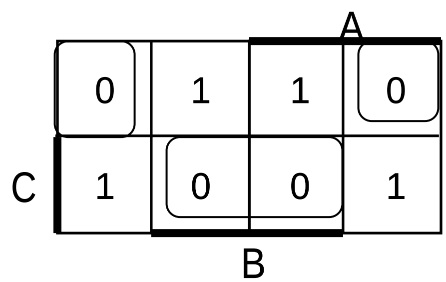
(b) X(A, B, C) = (B' + C) (B + C')



(c) X'(A, B, C) = B'C + BC'

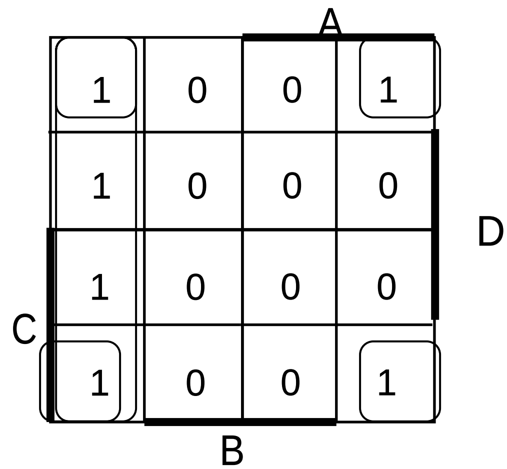


(d) X’(A, B, C) = (B + C) (B’ + C’)

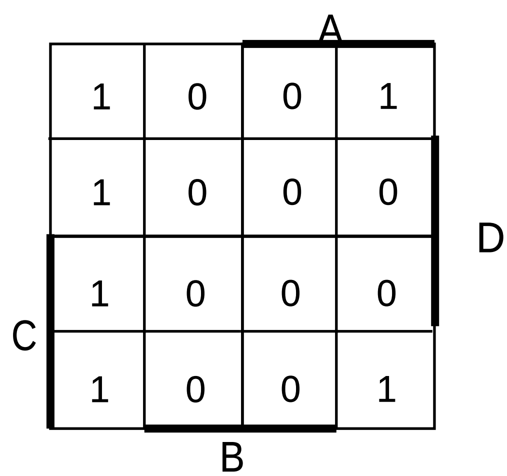


Solution for part 2.30 part (b).

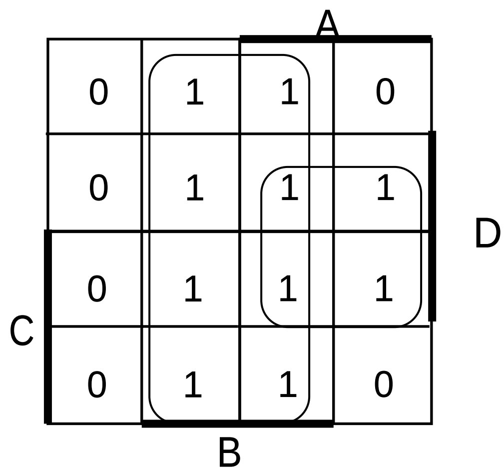
(a) Y(A, B, C, D) = A’B’ + B’D’



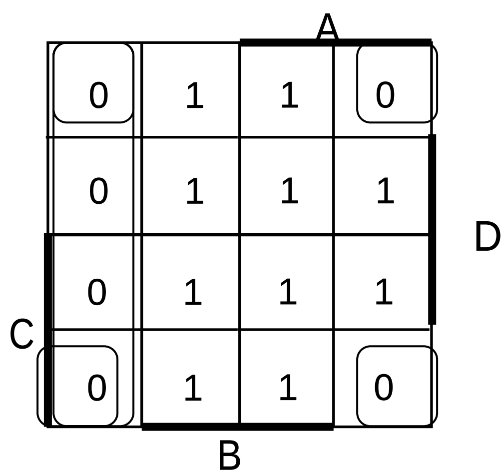
(b) Y(A, B, C, D) = B (A + D)



(c) Y’(A, B, C) = B + AD



(d) Y’(A, B, C) = (A’ + B’) (B’ + D’)



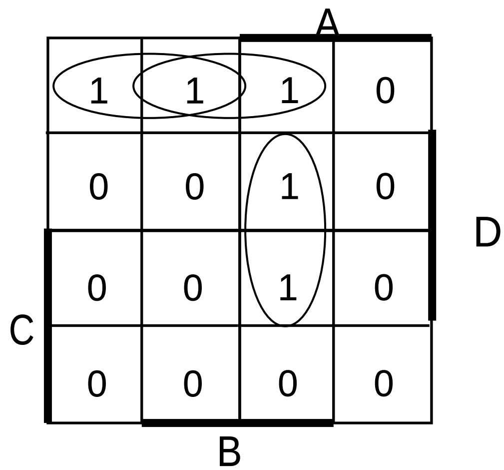
Exercise 2.32

The following two functions are equivalent as show by the K-maps below:

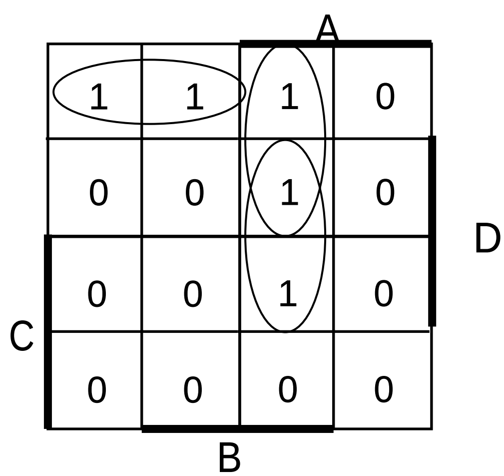
$$f_1(A, B, C, D) = ABD + BC'D' + A'C'D'$$

$$f_2(A, B, C, D) = ABD + ABC' + A'C'D'$$

f_1:

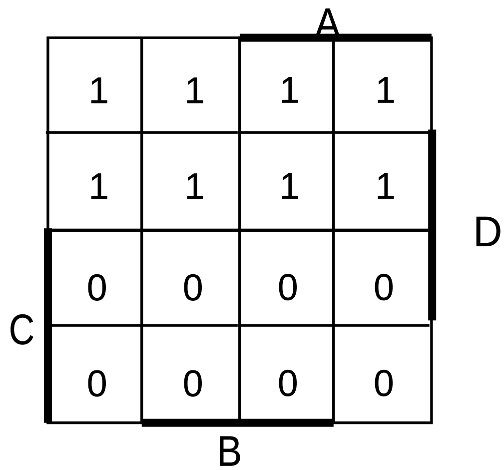


f_2:



Exercise 2.33

(a) The K-map below gives the sum-of-products form C’ and product-of-sums form C, each of which has one literal and one term.



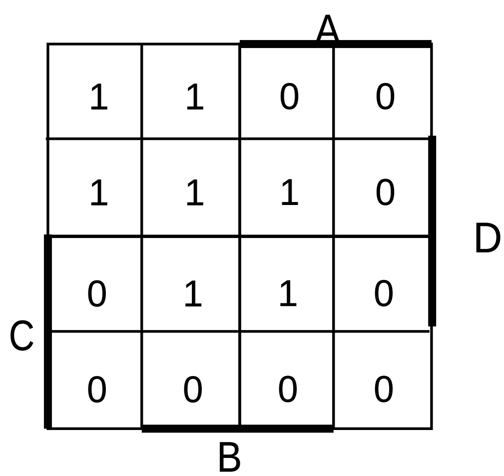
(b) The minimized sum-of-products for the following K-map is:

$$BD + A'C'$$

This equation has 4 literals and two terms. The minimized product-of-sums is:

$$(C + D')(A + B')(B' + C)(A + D')$$

This equation has eight literals and four terms.



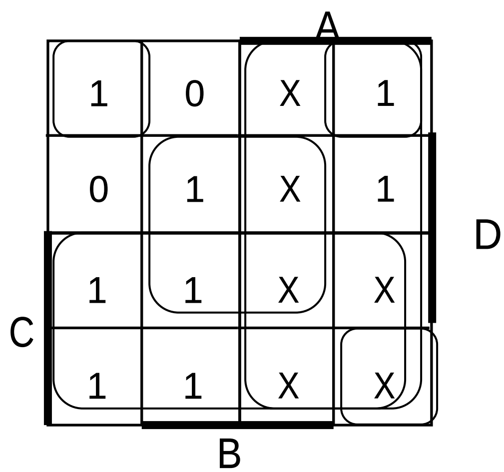
(c) By taking the complement of the previous K-map, the sum-of-products form now has eight literals and four terms, and the product-of-sums now has four literals and 2 terms:

Sum-of-products: $CD' + AB' + B'C + AD'$

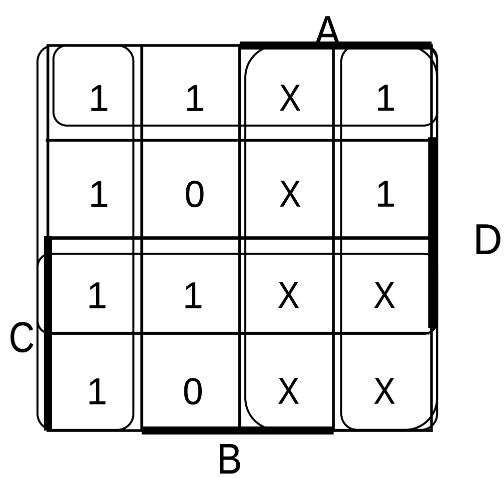
Product-of-sums: $(B + D)(A' + C')$

Exercise 2.34

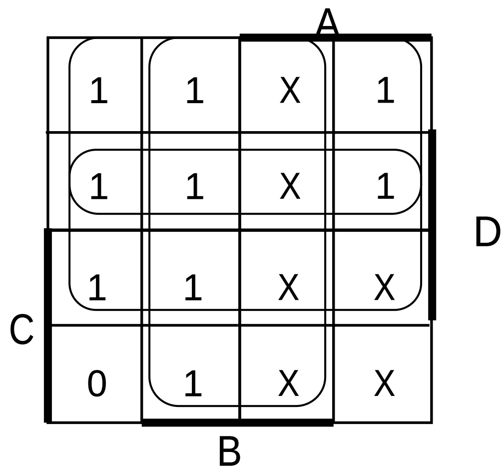
$C_0 = A + C + BD + B'D'$



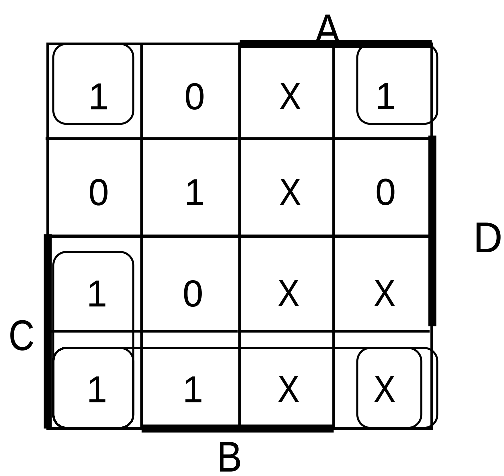
$C_1 = A + B' + C'D' + CD$



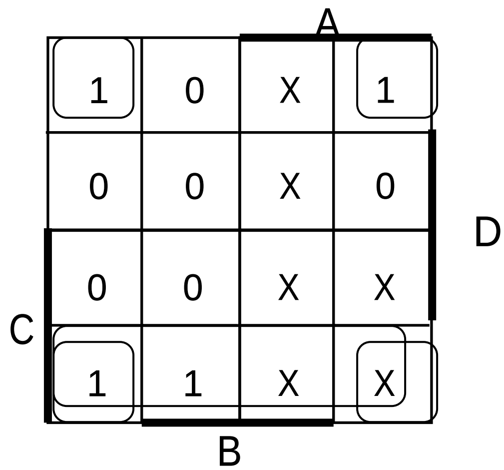
$C_2 = B + C' + D$



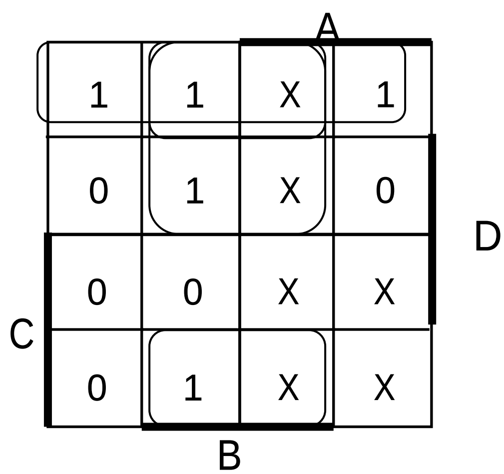
$C_3 = A'BC'D + B'D' + CD' + A'B'C$



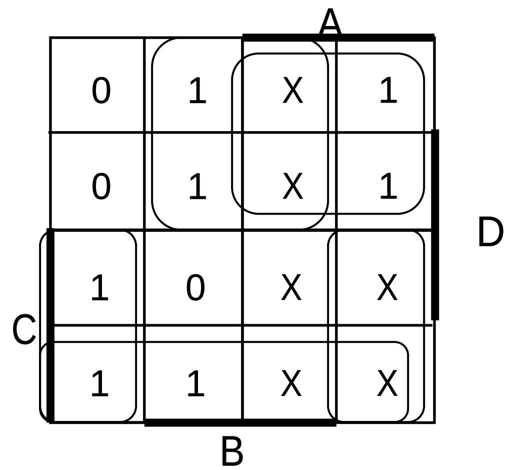
$C_4 = CD' + B'D'$



$C_5 = AC' + BD + BC'$



$C_6 = AC' + BC' + B'C + CD'$



Exercise 2.35

The truth table below provides an easy method for determining the canonical sum-of-products form:

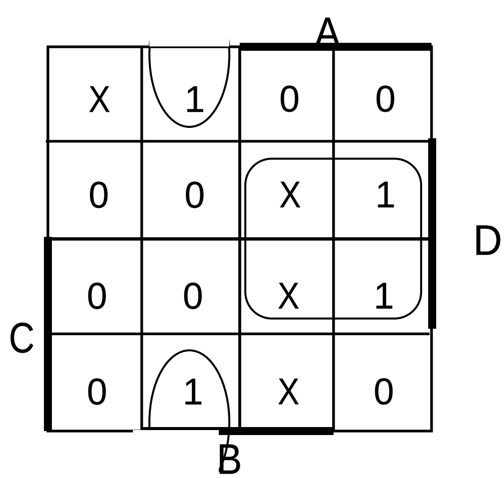
Month Code	d30	d31
0000	X	X
0001	0	1
0010	0	0
0011	0	1
0100	1	0
0101	0	1
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	0	1
1011	1	0
1100	0	1
1101	X	X
111X	X	X

$$d30 = \sum m(4, 6, 9, 11) + \sum d(0, 13, 14, 15)$$

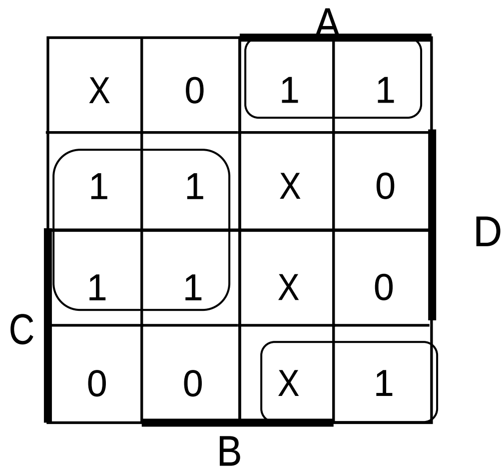
$$d31 = \sum m(1, 3, 5, 7, 8, 10, 12) + \sum d(0, 13, 14, 15)$$

These produce the following K-maps:

d30 = AD + A'BD'



d31 = A’D + AD’



Exercise 2.36

The truth table below provides an easy method for determining the canonical sum-of-products form (note the slight difference in encoding from the solution of Exercise 2.35):

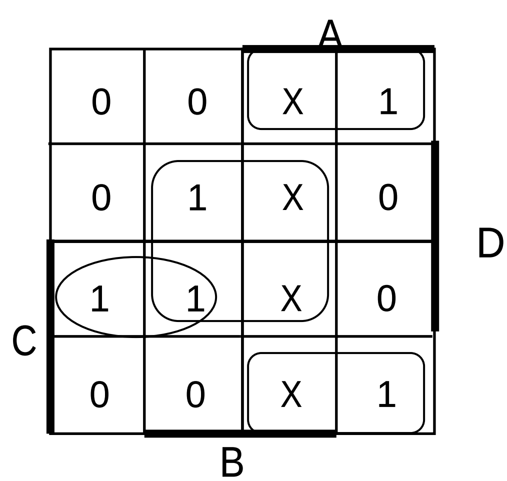
Month Code	d30	d31
0000	0	1
0001	0	0
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	0	1
0111	0	1
1000	1	0
1001	0	1
1010	1	0
1011	0	1
11XX	X	X

$$d30 = \sum m(3, 5, 8, 10) + \sum d(12, 13, 14, 15)$$

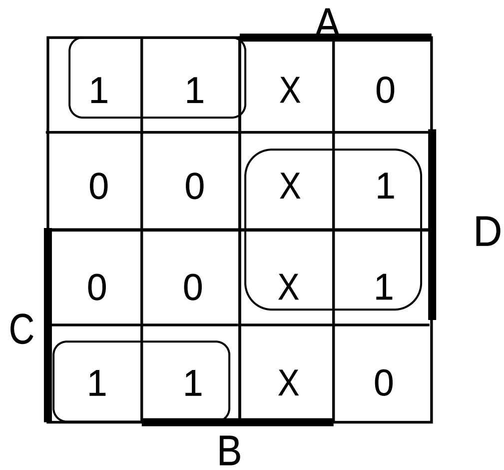
$$d31 = \sum m(0, 2, 4, 6, 7, 9, 11) + \sum d(12, 13, 14, 15)$$

These produce the following K-maps:

d30 = BD + AD' + A'CD



d31 = AD + A’D’



This minimized form turns out to be worse than the original encoding, since d30 now takes three terms instead of two, and two more literals. There isn’t much impact on d31 since it uses the same number of terms and literals with both encodings.

Exercise 2.37

Paragraphs that apply to the entire problem (or a single part problem) are not indented. This paragraph is "MainBody", while lettered paragraphs below are "IndentedBody". The title at the top is format "Heading".

(a) ABCD + ABDE = ABC (D + E)

(b) ACD + BC + ABE + BD = ACD + B (C + E + D)

(c) AC + ADE + BC + BDE = A (C + DE) + B (C + DE) = (A + B) (C + DE)

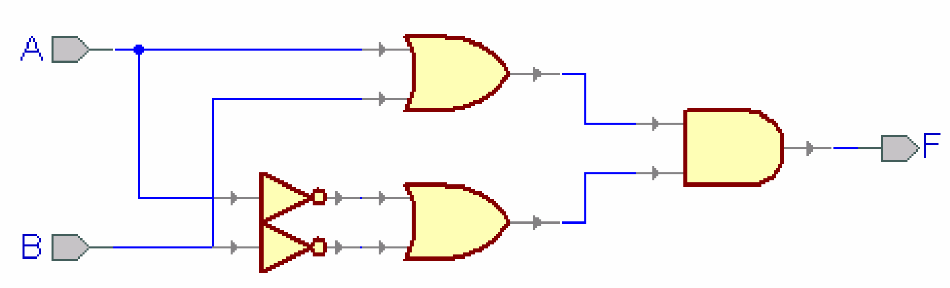
(d) AD + AE + BD + BE + CD + CE + AF = (A + B + C) E + (A + B + C) D + AF
= (A + B + C) (E + D) + AF

(e) ACE + ACF + ADE + ADF + BCE +BCF + BDF
= (AC + AD + BC) E + (AC + AD + BC) F + BDF
= (AC + AD + BC) (E + F) + BDF

Exercise 2.38

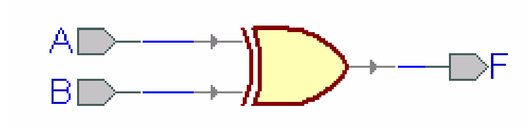
The following function implements the function in Figure Ex. 2.38 without using any NAND or NOR operations, as demonstrated by the figure.

$$F(A, B) = A' (A + B) + B' (A + B) = (A' + B') (A + B)$$



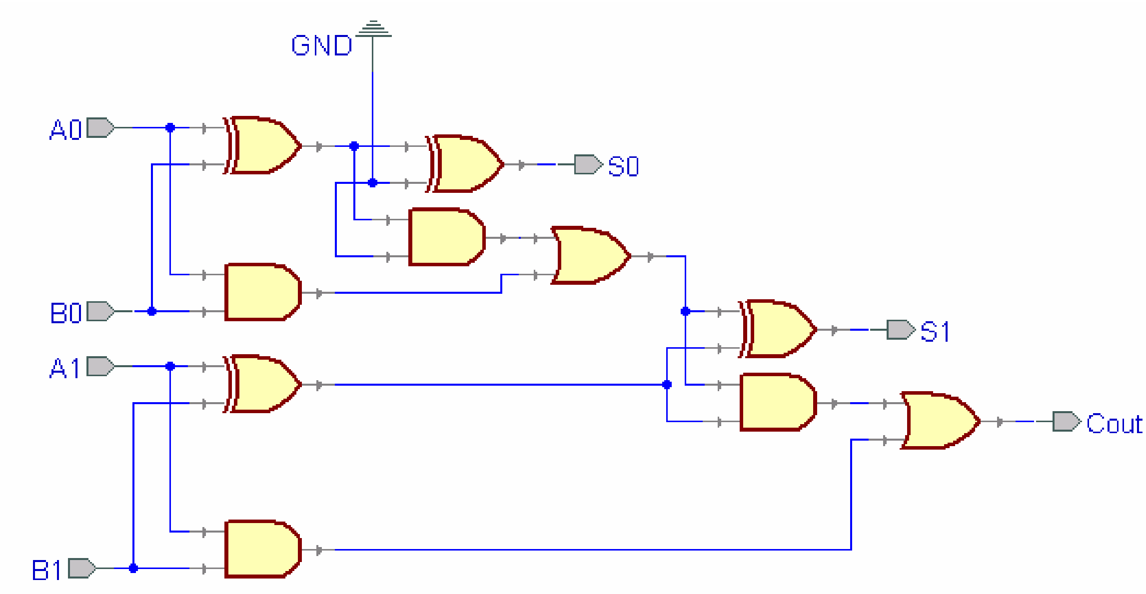
A minimized implementation using the fewest gates comes out to be the following function:

$$F(A, B) = (A' + B') (A + B) = (A'B + AB') = A \otimes B$$



Exercise 2.39

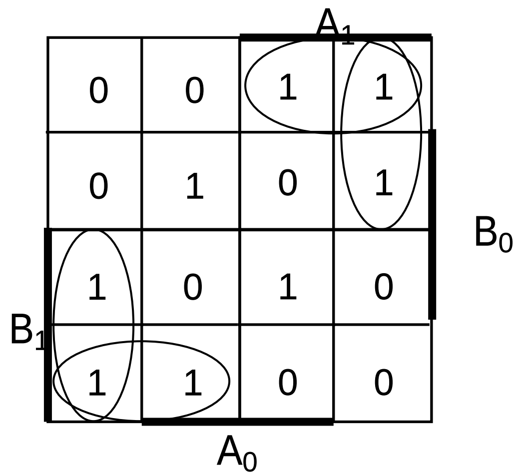
The method described in Figure 2.20 gives the following implementation of the full adder:



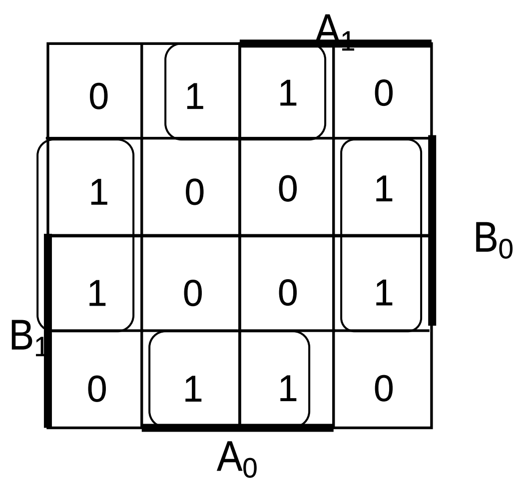
The truth table below is used to create K-maps, in order to create a minimized two level representation.

A1	A0	B1	B0	S1	S0	Cout
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	0	1

$$S_1 = A_1 A_0 B_1 B_0 + A_1' A_0 B_1' B_0 + A_1 B_1' B_0' + A_1 A_0' B_1'$$



$$S_0 = A_0 B_0' + A_0' B_0$$



$$C_{out} = A_1 B_1 + A_1 A_0 B_0 + A_0 B_1 B_0$$

Since the implementation of this will be fairly straightforward, a diagram is not being provided. The first implementation uses a variety of two-input gates including AND, OR, and XOR, whereas the second one uses a variety of AND and OR gates that can fanin up to 4 inputs. In the case of total number of gates the first implementation has ten gates, whereas the second implementation has eleven gates. The first implementation will have fewer wires, since factorization helps reduce the number of times the same solutions are computed. The second implementation will be faster though, since the worst case is through two gates with 4 inputs, whereas the first implementation has to travel through six gates along the worst path. The delays incurred from a larger fanin are not enough to cover the delays due to the 2 extra gates in this case.

Exercise 2.40

Paragraphs that apply to the entire problem (or a single part problem) are not indented. This paragraph is "MainBody", while lettered paragraphs below are "IndentedBody". The title at the top is format "Heading".

(a)

SHIFT	i_0	i_1	o_0	o_1
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	1
1	1	1	0	1

(b)

IN	SELECT	o_0	o_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

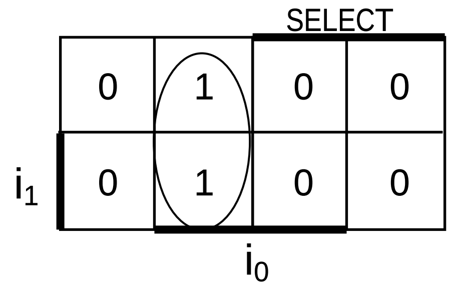
(c)

SELECT	i_0	i_1	OUT
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

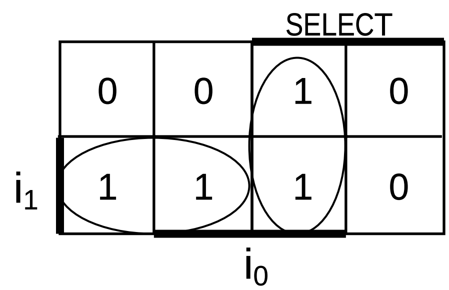
Exercise 2.41

Note: the solutions provided are dependent on the truth table in the answer to Exercise 2.40.

(a) $o_0 = \sum m(2, 3) = SELECT \bullet i_0$



$o_1 = \sum m(1, 3, 6, 7) = SELECT \bullet i_0 + SELECT' \bullet i_1$

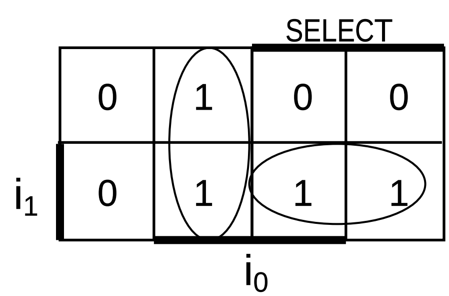


(b) $o_0 = \sum m(2) = SELECT' \bullet IN$

$o_1 = \sum m(3) = SELECT \bullet IN$

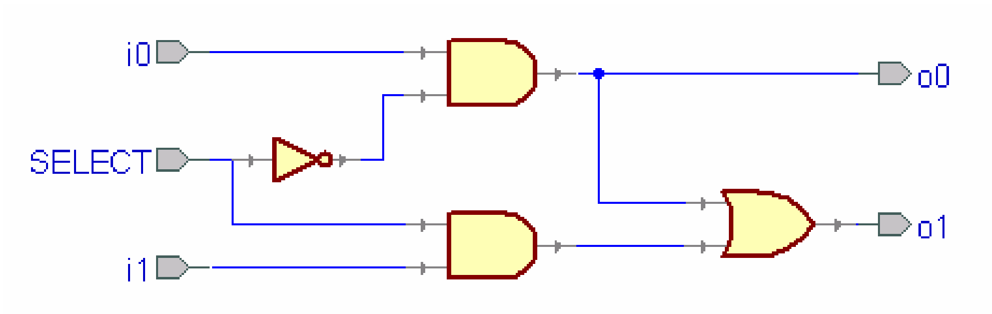
In this case it does not make sense to use a K-map for simplification since each function consists of only one term.

(c) $OUT = \sum m(2, 3, 5, 7) = SELECT' \bullet i_0 + SELECT \bullet i_1$

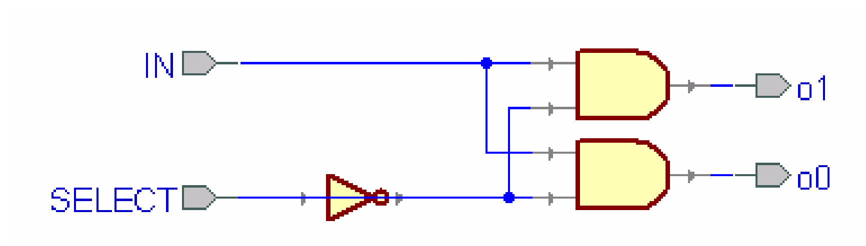


Exercise 2.42

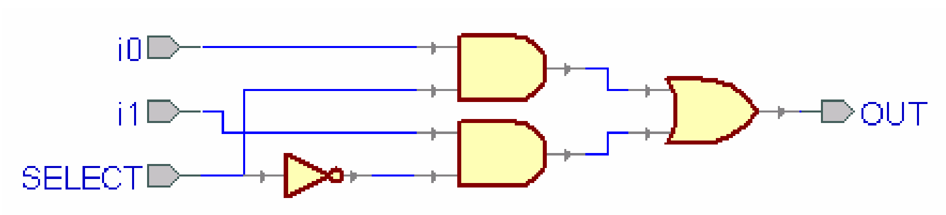
(a)



(b)



(c)



Exercise 2.43

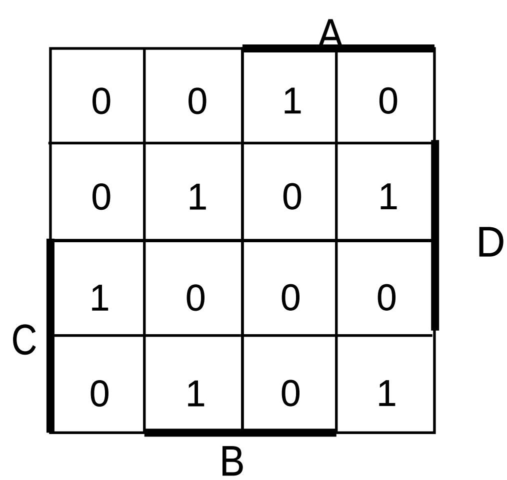
(a)

Input	Output
0000	0
0001	0
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	1
1011	0
1100	1
1101	0
1110	0
1111	0

(b) $\sum m(3, 5, 6, 9, 10, 12)$

(c) $\prod M(0, 1, 2, 4, 7, 8, 11, 13, 14, 15)$

(d) The sum-of-products K-map does not simplify since there are no adjacencies.



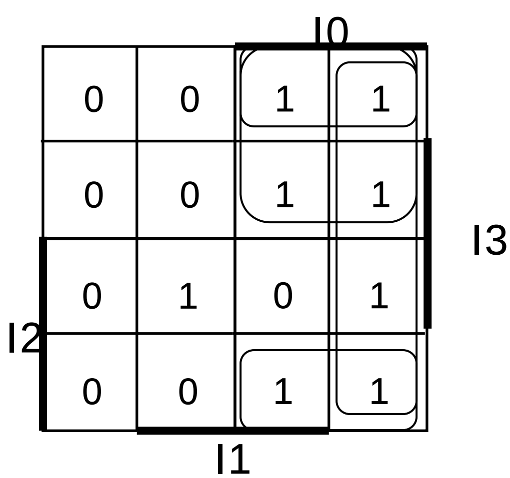
Exercise 2.44

(a)

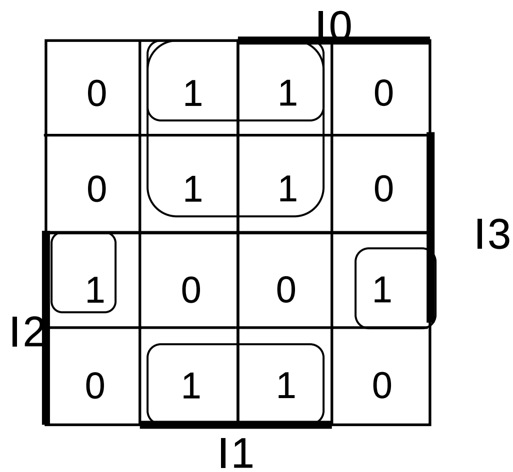
Input	Output
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	1010
1010	1011
1011	1100
1100	1101
1101	1110
1110	1111
1111	0000

(b) Note: I_0 and O_0 are the most significant bits in each of their corresponding bit streams.

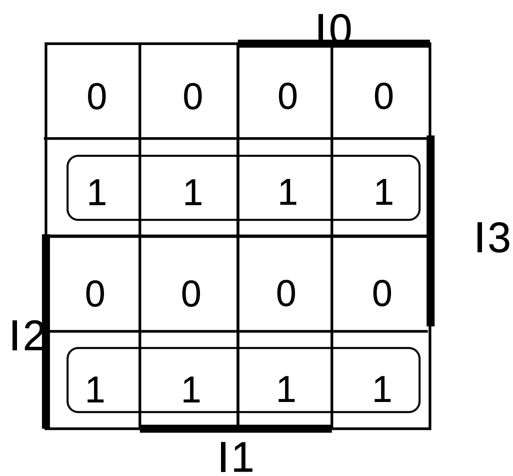
$$O_0 = I_0 I_2' + I_0 I_1' + I_0 I_3' + I_0' I_1 I_2 I_3$$



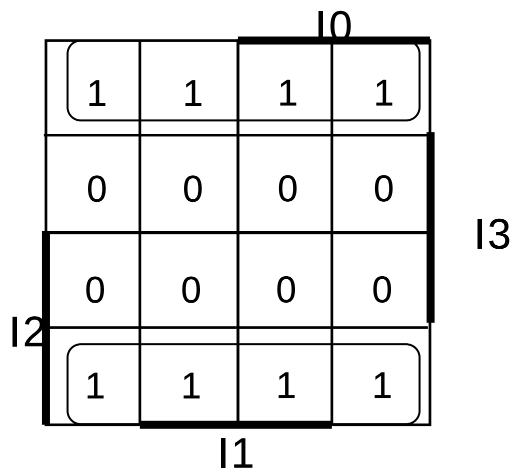
$O_1 = I_1 I_3' + I_1 I_2' + I_1' I_2 I_3$



$O_2 = I_2' I_3 + I_2 I_3'$

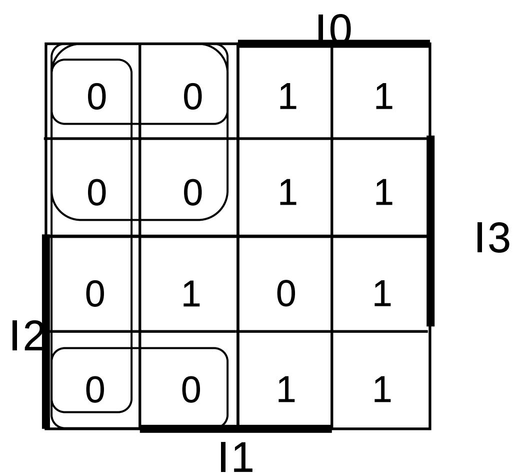


$O_3 = I_3'$

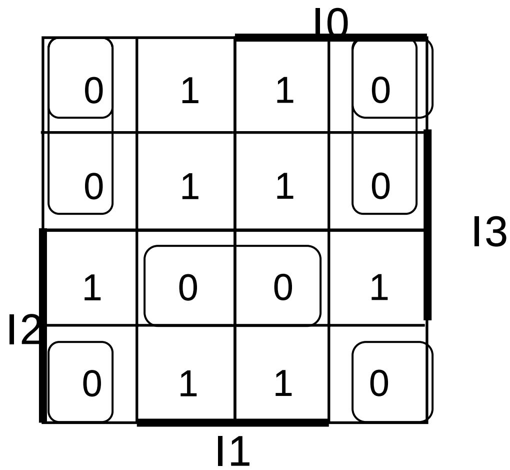


(c)

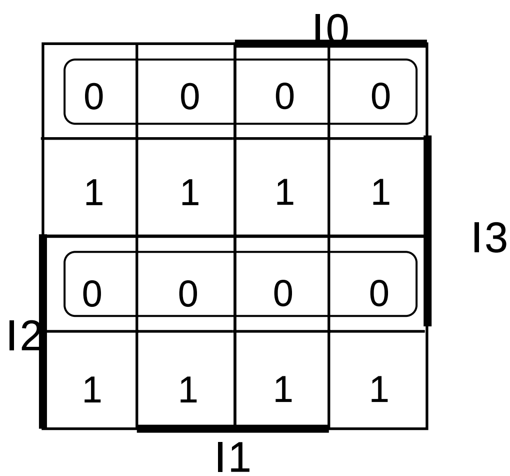
$O_0 = (I_0' + I_2') (I_0' + I_1') (I_0' + I_3') (I_0 + I_1 + I_2 + I_3)$



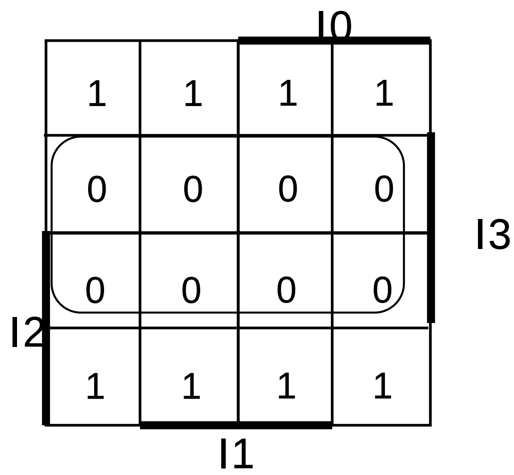
$$O_1 = (I_1' + I_3')(I_1' + I_2')(I_1 + I_2 + I_3)$$



$$O_2 = (I_2 + I_3)(I_2' + I_3')$$



$O_3 = I_3$



It turns out that both implementations are equivalent in the number of literals used for each of the Output bits.

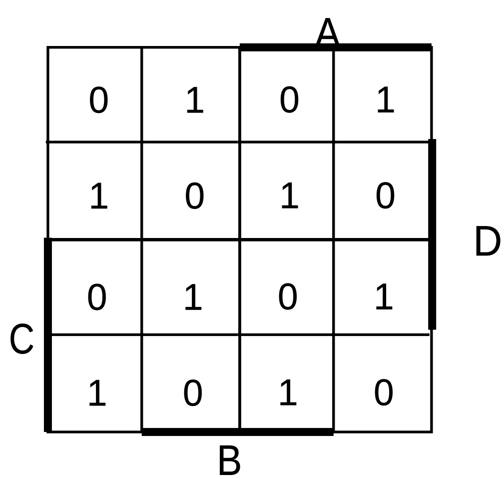
Exercise 2.45

(a)

Input	Output
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

(b) In the K-map, A represents the most significant bit of the input bit string. In this case the K-map method is not very useful in eliminating terms. The minimized form using the K-map is in fact the canonical sum-of-products form:

$$\sum m(1, 2, 4, 7, 8, 11, 13, 14)$$



(c) The truth table below shows that $A \oplus B \oplus C \oplus D$ is equivalent to the Output function. Logically this makes sense because if an even number of inputs are asserted, then the XOR of those inputs will always be 0. If an odd number of inputs are asserted, then *2n* asserted inputs before it produce a 0, which when XORed with the last asserted bit produces a 1.

Input	$A \oplus B \oplus C \oplus D$	Output
0000	0	0
0001	1	1
0010	1	1
0011	0	0
0100	1	1
0101	0	0
0110	0	0
0111	1	1
1000	1	1
1001	0	0
1010	0	0
1011	1	1
1100	0	0
1101	1	1
1110	1	1
1111	0	0

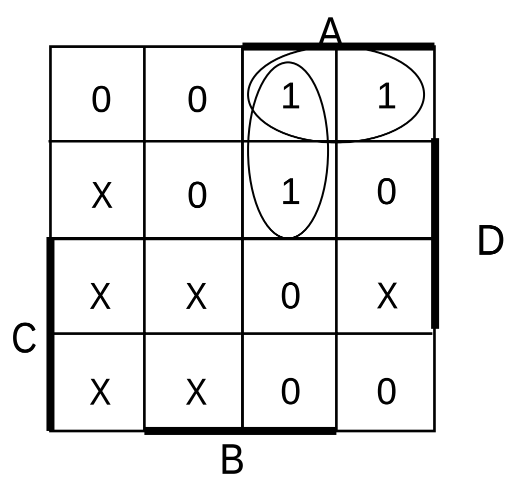
Exercise 2.46

(a)

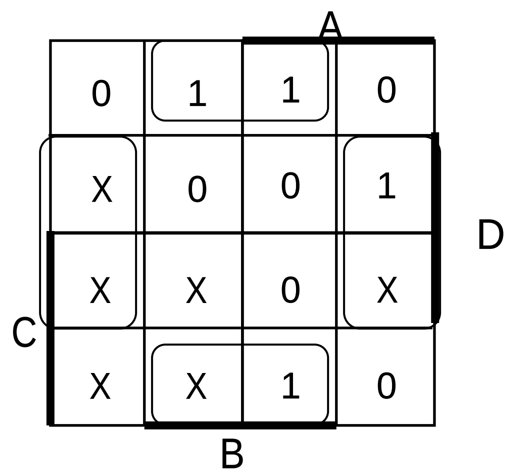
A	B	C	D	F	G
0	0	0	0	0	0
0	0	0	1	X	X
0	0	1	0	X	X
0	0	1	1	X	X
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	X	X
0	1	1	1	X	X
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	X	X
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	1	0	0

(b) In the case of F, the don't cares do not provide any help in simplifying the sum-of-products expression. They do help in simplifying G however.

F = ABC' + AC'D'

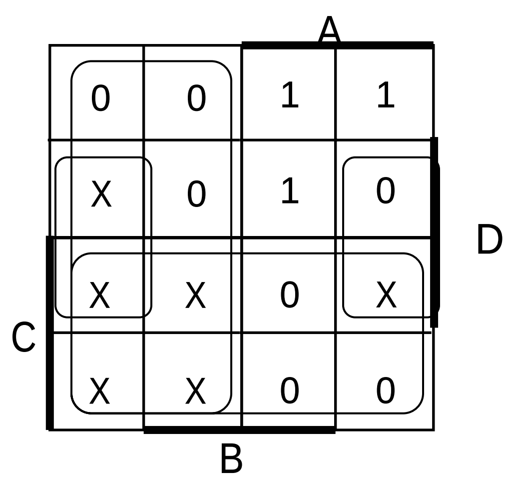


$$G = B'D + BD'$$



(c) The product-of-sums implementation turns out to be simpler than the sum-of-products implementation. This is because F now has only 4 literals instead of 6. Both implementations of G are about the same since they each have 4 literals and 2 terms.

$$F = (A')(C)(B' + D)$$



$$G = (B + D)(B' + D')$$

A

	0	1	1	0	
	X	0	0	1	D
C	X	X	0	X	
	X	X	1	0	

B