DIGITAL INSTRUCTION MODEL

Part 1: Static combinations

3rd Edition

Training in Technology

SystemTechnik

Lehr- + Lernmittel GmbH
Altdorfer Strasse 16
D-88276 Berg bei Ravensburg
Phone: +49 751/5 60 75-0
Telefax: +49 751/5 60 75 16

Order no.: V 0035

CONTENTS PART I

1. INTRODUCTION

The hps digital trainer type 3510 F is a universal experimenter for demonstrating, testing, designing and checking digital logic circuits and systems.

The digital trainer has sufficient standard logical building blocks for the construction of complex sequentially-combined circuits. These building blocks include gates, flipflops, memories, shift registers, adders, decoders, counters, dividers, monoflops, clock generators and input and output units (switches, pushbuttons, indicator lamps, relays and seven-segment display units).

All the laws of Boolean algebra can be demonstrated with ease with the aid of the gates provided (AND/NAND, OR/NOR, inverters). This makes the unit particularly suitable for such establishments as vocational colleges, company training centres, secondary schools and so on.

With the more complex, ready-wired digital circuits such as shift registers, flipflops, adders, dividers, decoders, memories etc, these digital systems can be demonstrated in a way that is visually informative, particularly because of the optical display of logical states by the indicator lamps.

These circuits can also be included in larger circuit arrangements, for sequential logic operations, for instance. Almost every type of circuit currently in industrial use can therefore be constructed, and this makes the trainer well suited for use in technical colleges, universities and development laboratories.

The digital circuit development engineer will find the trainer particularly useful, since it enables him quickly to check a circuit design for correct operation. This saves a great deal of time, since it replaces the far more difficult process of thinking a sequence through.

The slow operating speed is a great advantage when checking sequential circuits, since it makes it very easy to detect and correct mistakes that may have been made when setting up the circuit; this would be far more difficult at a high operating speed.

The trainer is designed so that only the most important logical connections have to be made; all others are already wired inside the trainer.

Unused circuits may normally be left unconnected. This means that the trainer is not tied to a particular digital circuit technique (e.g. LSL, DTL, TTL etc), so that circuits designed on paper can be transferred directly to the plugboard of the trainer without difficulty. For this reason the circuit examples are arranged in such a way that only the principle circuit, but not a complete plug-in plan is given.

The 200 examples given (part I and II) should serve as stimulation only. Although the manuals provide a representative cross-section of digital circuits, the scope for the construction of other digital circuits with the trainer is practically unlimited.

It takes only a short period of use to realise that the trainer provides a strong incentive for users to try out their own circuit designs, since it quickly shows whether the design is correct or not.

The learning process will be more successful if the basic principle only, rather than the complete circuit, is taken from the manuals, and the desired circuit is worked out by a combination of reasoned thought and the trial-and-error method.

All the symbols on the front panel conform to DIN 40700/14 or are drawn so that they are unmistakable.

If familiarisation is necessary, it is advisable to read the circuit descriptions of the various modules on pages 6 to 25, and to try out the circuits.

The trainer is designed in such a way that the modules cannot be damaged by incorrect connection.

Other recommended hps manuals on the subject of digital logic:

V 0005 Basic principles of digital engineering

V 0034 Introduction to digital engineering

2. Function groups in the digital trainer type 3510

Available components:

10 Input variable switches with Q- and $\bar{Q}$-outputs, optical indication of 1-state by light-emitting diodes (LEDs), two outputs are debounced

1 unwired toggle switch

1 unwired pushbutton switch

20 AND/NAND elements each with four inputs; two of the elements have Schmitt-trigger inputs

12 OR/NOR elements each with four inputs

6 power inverters (fan-out n = 30)

2 relays 1 A, 20 V dc, 60 V ac, 0.7 ms

5 JK-MS flipflops each with three J- and K-inputs, static set and reset inputs; optical display of the Q-state by red LEDs

6 JK-MS flipflops each with one J- and K-input, static set and reset inputs; optical display of the Q-state by red LEDs

11 indicator lamps (red LEDs)

1 50 Hz clock generator with Q- and $\bar{Q}$-output (mains frequency)

1 clock generator 1.00000 MHz (quartz stabilised)

1 clock generator, variable from 0.1 Hz to 100 kHz by six steps

1 monoflop with Q- and $\bar{Q}$-output, triggering by positive or negative edge (Schmitt-trigger input) as well as single triggering by pushbutton, pulse width variable from 1 µs to 10 s, adjustable in seven steps

3 monoflop with Q- and $\bar{Q}$-output, triggering by positive or negative edge (Schmitt-trigger input), pulse width variable from 1 µs to 10 s, adjustable in seven steps (monoflop = monostable flipflop)

Dividers

1 1:1000 divider with reset input

1 1:100 divider with reset input

1 1:10 divider with reset input

3 1:10 dividers with 2^0, 2^1, 2^2 and 2^3 outputs and reset input

1 1:6 divider with reset input

1 1:6 divider with 2^0, 2^1 and 2^2 outputs with reset input

1 1:5 divider with reset input

2 five-position shift registers with parallel inputs and outputs, reset input, optical display of Q-states (right-shifting)

1 BCD nines-complement code converter 8421

1 BCD/decimal decoder

2 four-bit full adder

2 four-bit memory (D-flipflops) with optical display of memory states

2 seven-segment display units with BCD/seven-segment decoder

3 optical displays of the decimal point

2 IC sockets with extractor, 16 pins (DIL)

12 sockets for logic 0 (L)

10 sockets for logic 1 (H)

Power supply:	220/110 V; 50-60 Hz
Power consumption:	60 VA
Mains fuse:	0,5 A, surge-resisting
Mains switch:	on/off, with indicator lamp
dc power supply: (internally wired, stabilised and short-circuit protected)	+5 V/3 A
Dimensions:	L x W x H = 650 x 500 x 225 mm
Weight:	20 kg approx

Note:

Where several hps digital units (3510 F or 3505 D) are used in parallel, it is only necessary to connect logic 0 $\hat{=}$ 0 V.

Modules, inputs, outputs etc can be temporarily identified with a felt pen. The markings can easily be removed with a damp cloth.

Attention:

When feeding voltage signals in from outside it is essential to ensure that the voltage level of the external signal is only between 0 and +5 V (not negative).
Failure to ensure this may result in damage to the systems.

3. Description of the function groups of hps digital trainer type 3510

The hps digital trainer type 3510 F is equipped with TTL integrated circuits (TTL = transistor-transistor logic) Positive logic with normal assignment of the logical states is used:

The logical state ONE (logic 1, H) corresponds to a typical voltage level of about +3 V; the logical state ZERO (logic 0, L) corresponds to a typical voltage level of about 0.2 V.

Unless specifically defined otherwise, in this manual an open, unconnected input is a logic 1. Typical overall operating times are in the range 10 to 50 ns; the typical maximum clock frequencies of the individual elements are 10 to 50 MHz.

3.1 Input variable switch

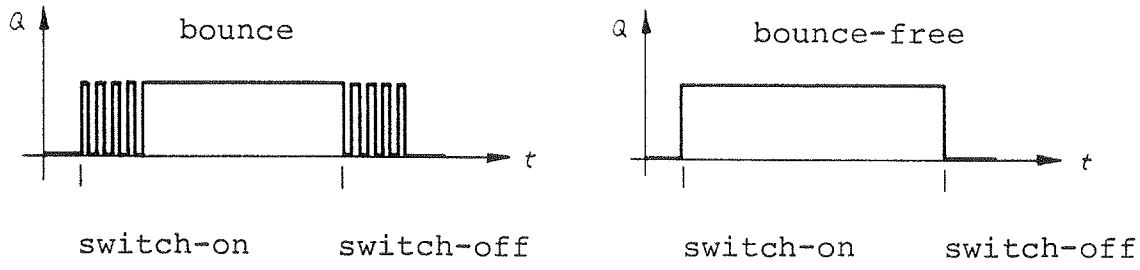
This switch is used to switch variable binary input quantities to digital devices (e.g. on coding, decoding, code conversion, logical operations etc). Each switch has a Q-output (up) and a $\bar{Q}$-output (down).

Toggle down: Q = 0, L $\bar{Q}$ = 1, H LED off

Toggle up: Q = 1, H $\bar{Q}$ = 0, L LED on

The switches are numbered from 0 to 9 (or from a to i).

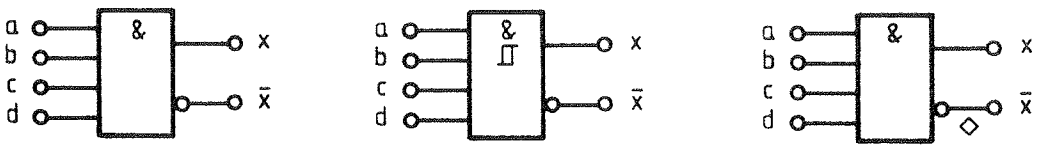
The output load factor (fan-out) is 10 for the Q- and $\bar{Q}$-outputs. The first and last switches, switches 0 and 9 (or a and i) are debounced.



3.2 Indicator lamps (LEDs)

The logical states can be displayed optically by means of these GaAsP (red) light-emitting diodes. If there is a logic 1 at the output (voltage present) the diode is lit; if there is a logic 0 (no voltage present) the diode is not lit. In this case an open output means logic 0, i.e. the lamp is not lit. The output load factor (fan-in) is less than 2.

3.3 AND/NAND gates

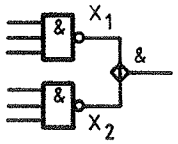


Function: $x = a \cdot b \cdot c \cdot d$; $(x = a \wedge b \wedge c \wedge d)$ AND

or: $\bar{x} = \overline{a \cdot b \cdot c \cdot d}$ NAND

Since an open (unconnected) input is treated as a logic 1 in this case (e.g. a = 1 or b = 1 or c = 1 or d = 1), these gates may be used directly without additional circuitry as a three-input AND gate (one input open) or as a two-input AND gate (two inputs open) or as an inverter (three inputs open). The two gates marked "S" (⊓) each have four Schmitt-trigger inputs, which are used to

make pulse edges steeper. The inverted outputs marked "R_C" (◊) are open-collector gates, which have a collector resistor connected internally to plus. These outputs are suitable for a wired AND operation.

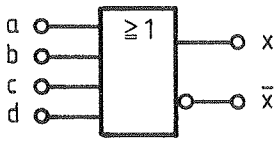


Function: $x = x_1 \cdot x_2$ AND

Fan-in per input: 1

Fan-out for x and $\bar{x}$: 10

3.4 OR/NOR gates



Function: $x = a + b + c + d$; $(x = a \vee b \vee c \vee d)$ OR

or: $\bar{x} = \overline{a + b + c + d}$ NOR

In these gates an open input means logic 0, so that unused inputs may be disregarded, (e.g. a = 0, b = 0, c = 0, d = 0). This gives, without additional circuitry, a gate with three, two or one inputs, depending on how many inputs are left open.

Fan-in per input: <2

Fan-out for x and $\bar{x}$: 10

3.5 Sockets for logic 0 (L) and logic 1 (H)

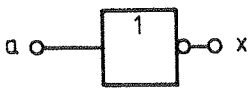
Since logic 0 and logic 1 are often needed in digital circuits, sockets with these two levels are distributed uniformly across the plugboard.

Coding:

Logic 0 socket 

Logic 1 socket 

3.6 Power inverter

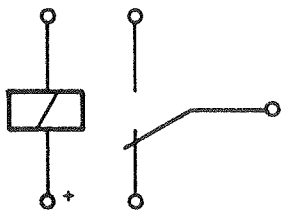


Function: $x = \bar{a}$

Fan-in: 1

Fan-out: 30

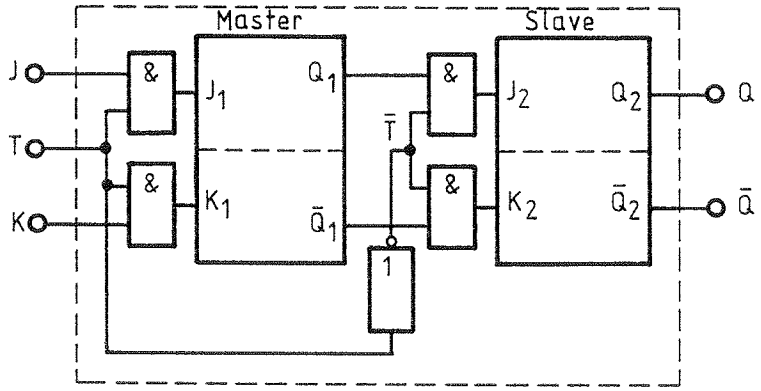
3.7 Relays



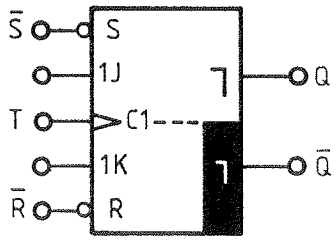
These relays can only be operated via a power inverter. The relay operates when the open end of the winding is connected to 0 V, or, if the relay is driven by an inverter, when a = 1.

Switching current	1 A
Switching voltage	20 V dc, 60 V ac
Response time	0.7 ms
Winding resistance R_{Cu}	$\approx$ 260 Ω

3.8 JK flipflop with one J- and one K-input



Arrangement of a JK master-slave flipflop made up of two JK flipflops with inverter



This is a clock-controlled JK master-slave flipflop in which any change at the output as a result of signals at the J- or K-input is only possible when there is a clock pulse at the T-input.

The logical behaviour is shown in the operation table

J	K	Q
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$
t_n		$t_n + 1$

Meanings of symbols:

t_n = time before the clock pulse or states before the clock pulse

$t_n + 1$ = time after the clock pulse or states after the clock pulse

Q_n = state before and after the clock pulse (0 or 1); no change by the clock pulse

$\overline{Q_n}$ = the state has changed after the clock pulse

Meanings of open J- and K-inputs: J = K = 1

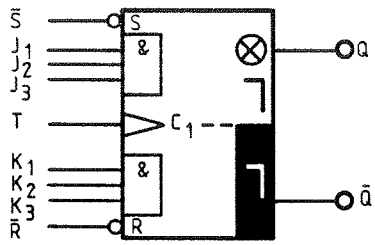
Since this is a master-slave flipflop the information at the J- and K-inputs is only accepted by the master flipflop on the positive edge of the clock pulse, and is only passed to the slave flipflop on the negative edge of the clock pulse, so that the information only appears at the outputs at this time. Information can only be transferred when the signals are applied to the J and K-inputs before the positive edge of the clock pulse. The clock pulse must be at least 20 ms long. The maximum clock frequency is 20 MHz.
The static set and reset inputs are unaffected by the clock.

Reset:	$\bar{R} = 0$	(Q = 0)	
Set:	$\bar{S} = 0$	(Q = 1)	
and $\bar{S} = 0$,	$\bar{R} = 0$	(Q = 1)	"dominant setting"

The Q-state is displayed optically with the aid of an LED.

Fan-in	J, K:	1
Fan-in	T, $\bar{R}$, $\bar{S}$:	2
Fan-out	Q, $\bar{Q}$:	10

3.9 JK flipflop with three J- and three K-inputs



This is basically the same input as in 3.8, but with a three-element AND circuit before the J-input and before the K-input.

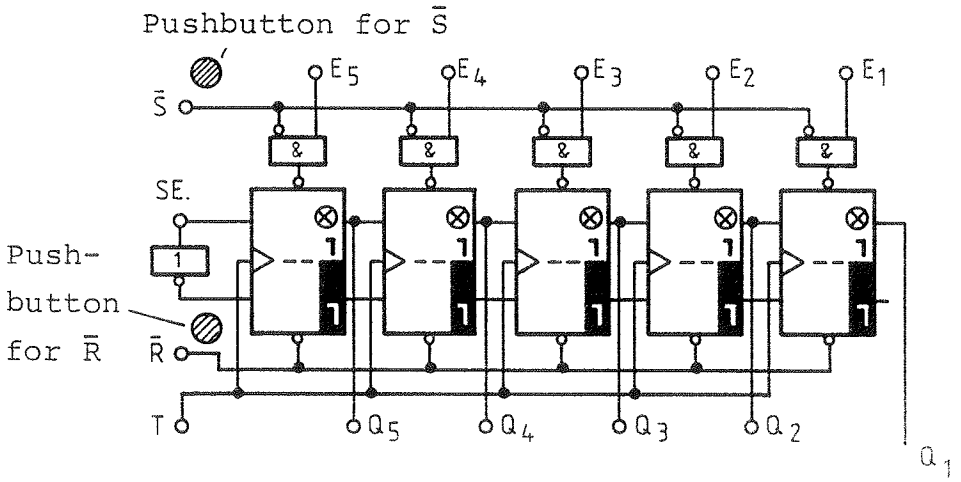
$$J = J_1 \cdot J_2 \cdot J_3$$

and

$$K = K_1 \cdot K_2 \cdot K_3$$

All other characteristics and data are as for the previous flipflop.

3.10 Shift registers



The two 5-bit shift registers (right-shifting) are made up of five JK master-slave flipflops (as described in 3.8). The serial input signal goes to serial input SE. For this, the information must be present at SE and at the reset input $\bar{R}$ = 1 (socket not connected) before the positive edge of clock pulse T.

At the positive edge of clock pulse T the information is taken from the previous flipflop or from SE, and is passed, at the negative edge of clock pulse T, to outputs Q_5 to Q_1.

When SE = $\{^1_0\}$, Q_5 = $\{^1_0\}$ after the negative edge of T.

Parallel information can be fed to inputs E_5 to E_1

regardless of the state of the clock and reset input $\bar{R}$. The information at E_5 to E_1 is passed simultaneously to the individual memory locations when $\bar{S} = 0$, or when the pushbutton above the $\bar{S}$ socket is pressed. Parallel output of the information is possible with the aid of outputs Q_5 to Q_1, so that this shift register may be used as a serial/parallel converter, parallel/series converter or memory (parallel and series operation).

All flipflops can be reset at the same time (Q_5 to $Q_1 = 0$) regardless of the clock signal, when $\bar{R} = 0$, or when the pushbutton above the $\bar{R}$ socket is pressed. The Q-states of the flipflops are displayed by LEDs.

If a 4-bit shift register only is required, for instance, the parallel information is fed to E_5 to E_2, and Q_2 is used as the serial output.

Open inputs E_1 to E_5 mean logic 0.

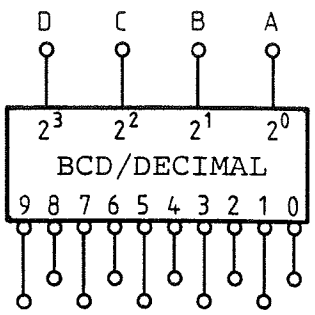
Fan-ins:	SE, $\bar{S}$, $\bar{R}$, T	= 1
	E_1 to E_5	< 2
Fan-outs:	Q_1 to Q_5	= 10

3.11 BCD→DECIMAL decoder

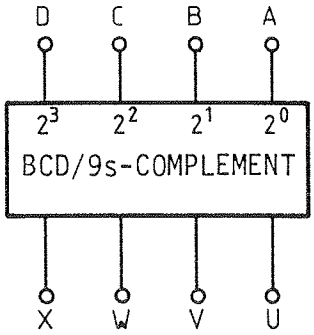
	D	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{8}$	$\overline{9}$
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
10	1	0	1	0	1	1	1	1	1	1	1	1	1	1
11	1	0	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	0	0	1	1	1	1	1	1	1	1	1	1
13	1	1	0	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	0	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Inputs				Outputs									

The function table clearly describes the operation of this decoder.
Open outputs correspond to logic 1.

Fan-out:
Input: 1
Output: 10



3.12 Nines-complement



Dec.	D	C	B	A	X	W	V	U	Dec.
0	0	0	0	0	1	0	0	1	9
1	0	0	0	1	1	0	0	0	8
2	0	0	1	0	0	1	1	1	7
3	0	0	1	1	0	1	1	0	6
4	0	1	0	0	0	1	0	1	5
5	0	1	0	1	0	1	0	0	4
6	0	1	1	0	0	0	1	1	3
7	0	1	1	1	0	0	1	0	2
8	1	0	0	0	0	0	0	1	1
9	1	0	0	1	0	0	0	0	0
	Inputs				Outputs				

The nines-complement of a one-digit decimal number is the amount that must be added to that number to make nine.
The nines-complement of the four-digit binary number D C B A is present in binary code at sockets X/W/V/U.

Fan-out:
Input: A, B, C, D: 1
Output: U, V, W, X: 10

3.13 4-bit full adder

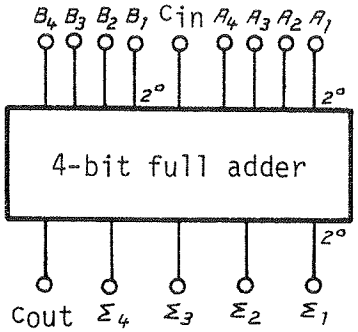
This full adder adds two 4-digit binary numbers $A_4/A_3/A_2/A_1$ and $B_4/B_3/B_2/B_1$, where a_1 and b_1 are the least significant digit (2^0).

c_{in} is the input carry of a previous addition. If there is no c_{in}, this socket must be connected to 0. $\Sigma_4/\Sigma_3/\Sigma_2/\Sigma_1$ is the sum of the two 4-digit binary numbers, and c_{out} is the output carry of this sum. Σ_1 is the least significant digit.

Signal:	13	$\hat{=}$	1	1	0	1	$\hat{=}$	A_4		A_3	A_2	A_1
	12	$\hat{=}$	1	1	0	0	$\hat{=}$	B_4		B_3	B_2	B_1
	+ 1	$\hat{=}$				1	$\hat{=}$					c_{in}
	26	$\hat{=}$	1	1	0	1	0 $\hat{=}$	c_{out}	Σ_4	Σ_3	Σ_2	Σ_1

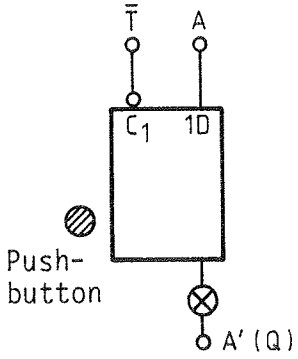
Fan-ins: A_2, A_4, B_2, B_4, c_{in} : 4, A_1, A_3, B_1, B_3 : 1
Fan-outs: Σ_1, Σ_2, Σ_3, Σ_4 : 10; c_{out} : 5

NB: Open inputs $\hat{=}$ logic 1 (H)



If a 1-bit full adder only is required, the following must be connected to 0: A_2, A_3, B_2, B_3, B_4. The sum is present at Σ_1, the output carrier at Σ_2. (In the experiment instructions in hps manuals V 0035 and V 0036 the letter S is used instead of the Greek letter Σ.)

3.14 Memory flipflops (D flipflops)



These flipflops have two stable states controlled by the clock signal ($\bar{T}$). As long as the clock pulse $\bar{T}$ = 0, the information at D is passed to output Q, and is held there even when $\bar{T}$ goes to 1 (e.g. socket open).

Logical function

t_n	t_{n+1}
A	A'
1	1
0	0

t_n = time before the clock pulse
t_{n+1} = time after the clock pulse

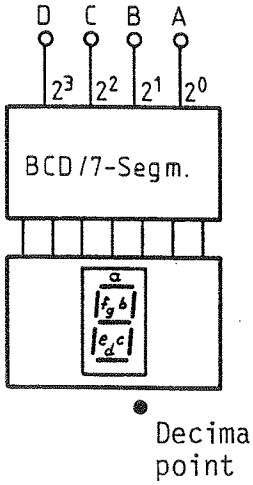
Information transfer by T can also be achieved by pressing the pushbutton. When Q = 1 a signal lamp lights up at the output. Open inputs A, B, C, D behave as logic 0.

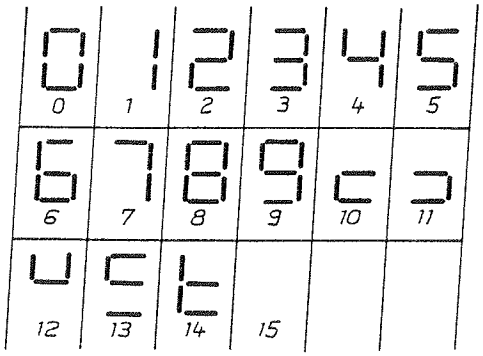
Fan-ins: $\bar{T}$: <2
A, B, C, D : 3
Q-outputs: A', B', C', D' : 10

3.15 Seven-segment displays

These units are made up of a decoder and a display. A decimal point can also be made to appear by means of the decimal point socket.
The decoder decodes binary-coded decimal numbers D/C/B/A (where the value of A is lowest) in binary code to seven segments a to g.

Dec.	D 2^3	C 2^2	B 2^1	A 2^0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	0	0	0	1	1	0	1
11	1	0	1	1	0	0	1	1	0	0	1
12	1	1	0	0	0	1	0	0	0	1	1
13	1	1	0	1	1	0	0	1	0	1	1
14	1	1	1	0	0	0	0	1	1	1	1
15	1	1	1	1	0	0	0	0	0	0	0



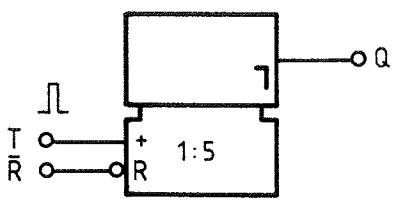


Fan-ins: A, B, C, D < 2
Decimal point: < 2

3.16 Dividers

All dividers are made up of negative-edge-triggered JK master-slave flipflops (information taken in on positive edge, passed on on negative edge. The dividers count with the binary code (BCD). All flipflops can be set to 0 (counter setting 0) with reset socket $\bar{R}$. The two 1:10 dividers on the right of the plugboard can also be manually reset with the pushbutton. Resetting takes place at $\bar{R} = 0$. All dividers are ready to count on $\bar{R} = 1$ (socket open). All the dividers work asynchronously.

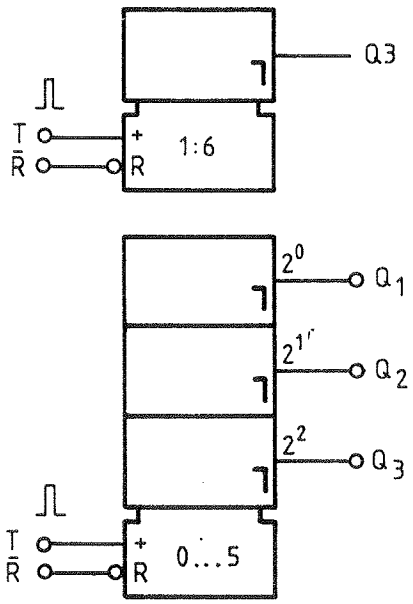
a) 1:5 divider



T	Q
0	0
1	0
2	0
3	0
4	1
0	0

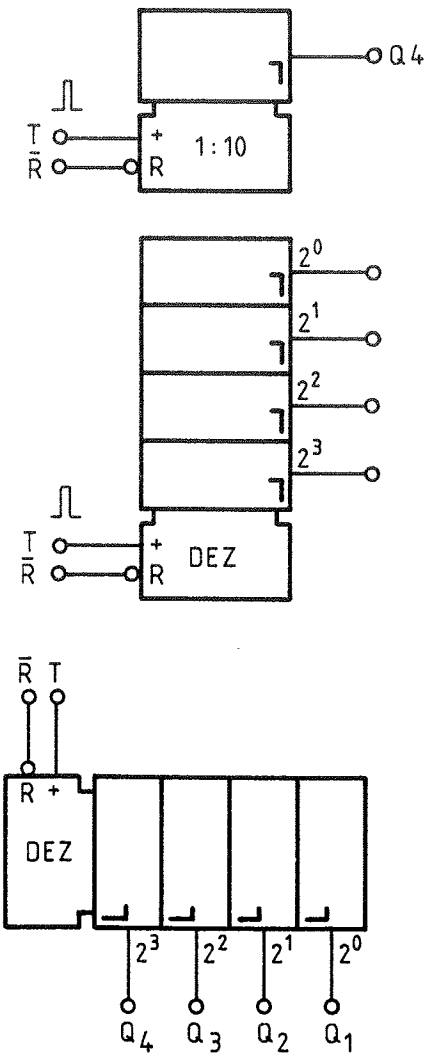
b) 1:6 divider

T	Q_3	Q_2	Q_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
0	0	0	0
	1:6	1:6	1:2

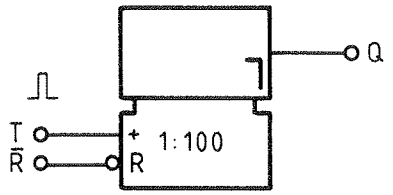


c) 1:10 divider

T	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0
	1:10	1:10		1:2

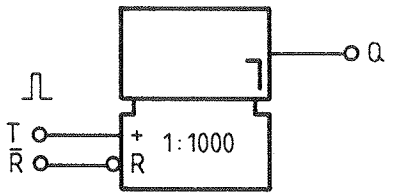


d) 1:100 divider



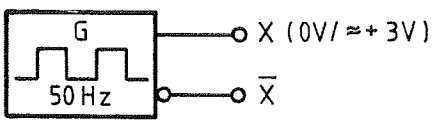
T	Q
0	0
1	0
.	.
.	.
.	.
80	1
.	.
.	.
.	.
99	1
0	0

e) 1:1000 divider



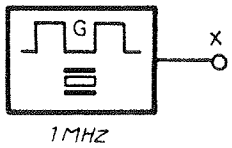
T	Q
0	0
1	0
.	.
.	.
.	.
800	1
.	.
.	.
.	.
999	1
0	0

3.17 50 Hz generator



Steep-edged square-wave pulses are available at output terminals x and $\bar{x}$ (amplitude +3 V approx.) The pulse length is shorter than the space between the pulses (at output x). The frequency is identical to the 50 Hz mains frequency. Fan-outs, x and $\bar{x}$: 10

3.18 1 MHz pulse generator



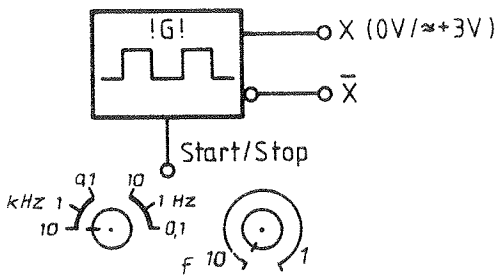
0 V/≈ +3 V

This generator produces square-wave pulses (amplitude +3 V) at the highly accurate frequency of 1.00000 MHz using a quartz crystal. The accuracy of the frequency is better than 10^{-5} (<0.01%). By using frequency dividers it is possible to produce highly accurate frequencies (for a digital clock, for instance).
Fan-out: x = 10

3.19 Variable-frequency clock generator

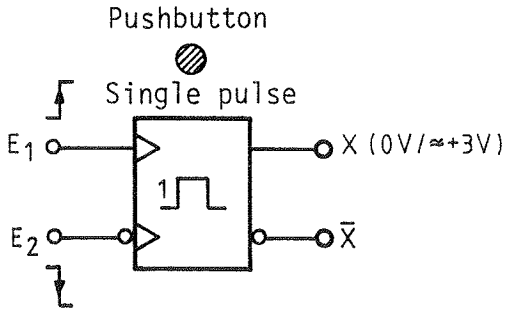
Outputs x and $\bar{x}$ supply steep-edged square wave pulses of variable frequency from 0.1 Hz to 100 kHz (amplitude 3 V approx).

Step 1	100 kHz	to	10 kHz
Step 2	10 kHz	to	1 kHz
Step 3	1 kHz	to	100 Hz
Step 4	100 Hz	to	10 Hz
Step 5	10 Hz	to	1 Hz
Step 6	1 Hz	to	0.1 Hz



The range switch is used for coarse setting, and a potentiometer (accuracy about 10%) is used for fine setting. There is an overlap of about 10% on the potentiometer at the beginning and end of adjacent steps. The duty factor of the output pulses varies slightly on fine adjustment. The generator can be disabled (stopped) by applying logic 0 to the start/stop socket, and can be enabled (started) by applying a logic 1 or leaving the socket disconnected, so that the generator does not give the first pulse until after the generator has been enabled. This is very useful for synchronised processes.
Fan-outs, x and $\bar{x}$: 10

3.20 Monoflop



The monoflops give steep-edged single pulses (amplitude 3 V approx) of adjustable pulse width τ at outputs x and $\bar{x}$. The pulse width τ can be varied from 1 µs to 10 s (accuracy about 10%).

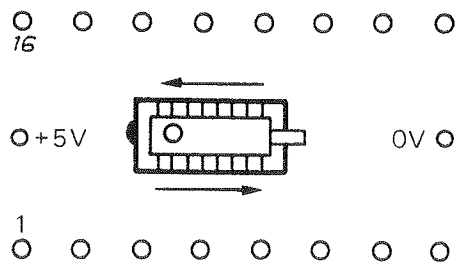
	Fixed	Variable
Step 1	1 µs	1 µs to 10 µs
Step 2	10 µs	10 µs to 100 µs
Step 3	100 µs	100 µs to 1 ms
Step 4	1 ms	1 ms to 10 ms
Step 5	10 ms	10 ms to 100 ms
Step 6	100 ms	100 ms to 1 s
Step 7	1 s	1 s to 10 s
Step 8	10 s	

The overlap of adjacent steps for the monoflop with variable pulse width setting is about 20%.
The single pulse is triggered at input E_1 by positive edges, and at input E_2 by negative edges.
The rise time or fall time of the trigger signals at E_1 and E_2 must be less than 1 V/s (Schmitt-trigger input)
Single pulses can also be generated by pressing the bounce-free pushbutton. A pulse is triggered when the pushbutton is pressed.

Triggering at E_1 is only possible if E_2 = 1 (socket open) and if the pushbutton is not pressed. Triggering at E_2 is only possible at E_1 = 0 (socket open) and if the pushbutton is not pressed.
It is only possible to give a single pulse by pressing the pushbutton if sockets E_1 and E_2 are open (E_1 = 0, E_2 = 1).

Fan-ins: E_1 : 2
E_2 : 1
Fan-outs: x and $\bar{x}$: 10

3.21 Empty socket



This socket enables digital ICs in dual-in-line cases with up to 16 pins to be studied or included in a circuit. The +5 V supply voltage required is available at the +5 V socket, and can be connected to the appropriate socket of the IC.

Caution:

Only use the +5 V to supply the IC, never as a logic 1 signal, otherwise internal components may be damaged.
The IC sockets are fitted with an extractor for easy removal of the ICs.

4. BASIC LOGIC CIRCUITS AND BOOLEAN ALGEBRA

4.1 General

The base of the binary signal processing are the Boolean operations AND-OR-NOT, which are named the logic functions. Each binary signal process can be derived from one of these three operations and from a time-dependent function, the memory. Components of electronic circuits are constructed such, that the output of the component satisfies the basic operation as a function of one, but usually from several inputs.

4.1.1 AND-function (conjunction, coincidence)

The AND-operations of several input variables (a, b, c...) cause a through-connection at the output x when all inputs are present at the same time, e.g. a, b and c. Its circuit corresponds to a series connection of relay contacts.

The two possible ranges of binary electric quantity are to be marked according to DIN 41700/14 with L (low) and H (high). The values of range L are closer to - ∞ and of range H closer to + ∞. At positive logic this is:

$$L \mathrel{\hat{=}} 0$$

$$H \mathrel{\hat{=}} 1$$

As nearly all technical literature uses 0.1, these easily understood symbols are used in these manuals henceforth.

Truth-table

c	b	a	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

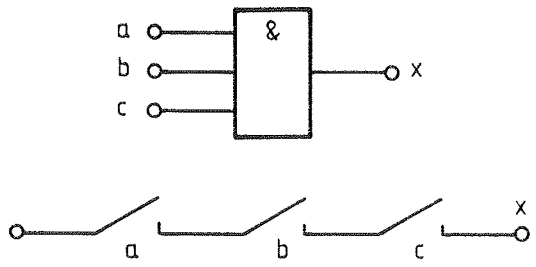
$x = a \cdot b \cdot c$

other form of notation:

$x = a \wedge b \wedge c$

$x = a \,\&\, b \,\&\, c$

Symbol:



4.1.2 OR-function (disjunction, mixture)

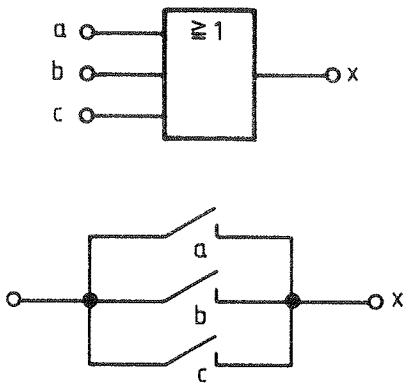
The OR-operation has a through-connection at the output, even if only one input is at "1". It corresponds to a parallel connection of contacts.

c	b	a	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$x = a + b + c$

$x = a \vee b \vee c$

Symbol:



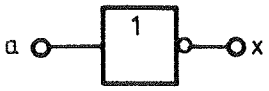
4.1.3 NOT-function

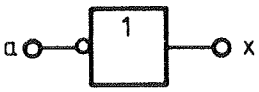
A NOT-operation means an inversion (negation) which is denoted by a line above the input and by a point (circle) in the symbol. The invertor is also named negator.

"0" → "1", $\bar{0} = 1$

"1" → "0", $\bar{1} = 0$

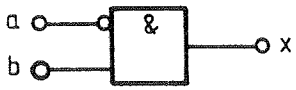
a	x
0	1
1	0



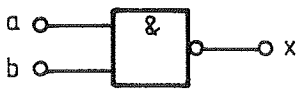


Examples:

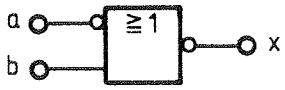
$x = \bar{a}b$



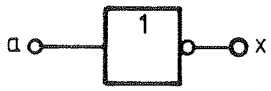
$x = \overline{ab}$



$x = \overline{\bar{a} + b}$



$x = \bar{a}$

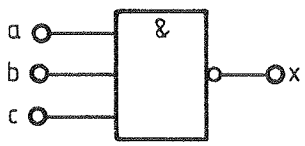


4.1.4 NAND-function

The NAND-operation is a negated AND-operation and has its name of NOT-AND. A NAND-operation causes at the output x a through-connection, if at least one input is at "0".

c	b	a	x
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Symbol:



$x = \overline{a \cdot b \cdot c}$ or

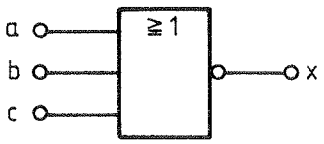
$x = \bar{a} + \bar{b} + \bar{c}$

4.1.5 NOR-function

The NOR-operation is a negated OR-operation and has its name from NOT-OR. A NOR-circuit causes a through-connection to the output, if none of the inputs are at "1".

c	b	a	x
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Symbol:



$x = \overline{a + b + c}$ or

$x = \bar{a} \cdot \bar{b} \cdot \bar{c}$

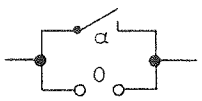
4.1.6 Theorems of switching algebra

OR: + or

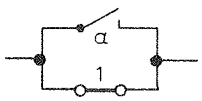
AND: · or

a, b, c = Boolean variables (0 or 1)

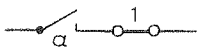
1 a) $a + 0 = a$



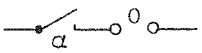
1 b) $a + 1 = 1$



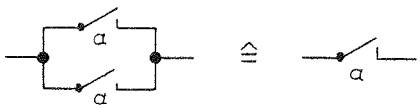
2 a) $a \cdot 1 = a$



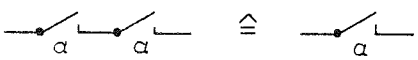
2 b) $a \cdot 0 = 0$



3 a) $a + a + a \ldots = n \cdot a = a$



3 b) $a \cdot a \cdot a \ldots = a^n = a$



4 $\overline{(\bar{a})} = \bar{\bar{a}} = a$

5 a) $a + b = b + a$

5 b) $a \cdot b = b \cdot a$

} Communitative law

6 a) $a + \bar{a} = 1$ 

6 b) $a \cdot \bar{a} = 0$ 

7 a) $a + b + c = (a + b) + c = a + (b + c) = b + (a + c)$

7 b) $a \cdot b \cdot c = (a \cdot b) \cdot c = a \cdot (b \cdot c) = b \cdot (a \cdot c)$

} *

*Associative law

8 a) $a \cdot b + a \cdot c = a \cdot (b + c)$

8 b) $(a + b) \cdot (a + c) = a + b \cdot c$

} distributive law

9 a) $a + a \cdot b = a$

9 b) $a \cdot (a + b) = a$

10 a) $a \cdot (\bar{a} + b) = a \cdot b$

10 b) $a + \bar{a} \cdot b = a + b$

} Reduction formulae

11 a) $a + \bar{a} \cdot \bar{b} = a + \bar{b}$

11 b) $\bar{a} + a \cdot b = \bar{a} + b$

12 $\bar{a} + a \cdot \bar{b} = \bar{a} + \bar{b}$

13 a) $(a + b) \cdot (\bar{a} + b) = b$

13 b) $(a \cdot b) + (\bar{a} \cdot b) = b$

14 a) $(a + b) \cdot (\bar{a} + c) = a \cdot c + \bar{a} \cdot b$

14 b) $(a + \bar{b}) \cdot (\bar{a} \cdot \bar{c}) = a \cdot \bar{c} + \bar{a} \cdot b$

15 a) $\overline{(a + b + c \ldots)} = \bar{a} \cdot \bar{b} \cdot \bar{c} \ldots$

15 b) $\overline{(a \cdot b \cdot c \ldots)} = \bar{a} + \bar{b} + \bar{c} \ldots$

} de Morgans theorem

16 a) $\overline{f\,(a \cdot b \cdot c \ldots)} = f\,(\bar{a} + \bar{b} + \bar{c} + \ldots)$

16 b) $\overline{f\,(a + b + c + \ldots)} = f\,(\bar{a} \cdot \bar{b} \cdot \bar{c} \ldots)$

} Shannons theorem

4.1.7 Set up of the output-function (switching function)

With the given truth table, the output functions may be found by 2 methods.

a) Disjunctive normal equation:
(disjunctive = connected to one another)

The formulation starts with all lines of the truth-table with "1" at the output x, a multiple serial connection (AND-operation) has to be set up; the corresponding row groups are to be connected parallel. In the Boolean Algebra this corresponds to the sum of several products, which are assigned to the individual lines with "1".

b) Conjunctive normal equation: (conjunctive = connecting)

A parallel connection (OR operation) is formulated from each row with a "0" at the output x. The rows are then connected by an AND operation and both inputs and outputs are inverted.

Example:

c	b	a	x
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

a, b, c = inputs (independent)

x = output (dependent)

Disjunctive normal equation:

$x = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + abc \qquad (abc \mathrel{\hat{=}} a \cdot b \cdot c)$

Conjunctive normal equation:

$x = (\bar{a} + b + c) \cdot (a + \bar{b} + c) \cdot (\bar{a} + b + \bar{c}) \cdot (a + \bar{b} + \bar{c})$

With the aid of the calculating rules both forms can be simplified so that the least expense for realization of the switching function is obtained.

4.1.8 Exclusive-OR (antivalence, X-OR)

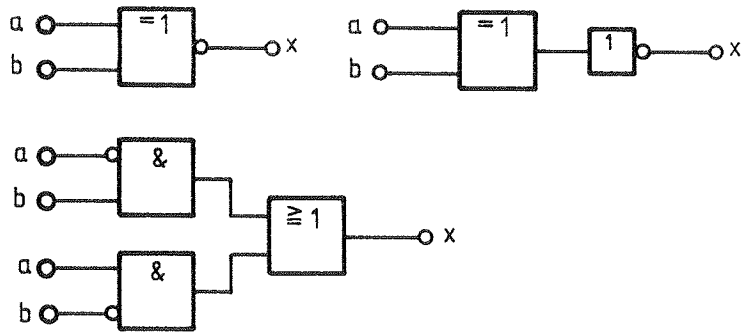
The exclusive-OR operation results at the output in a through-connection, if a input, but not several at the same time, show the value "1".

b	a	x
0	0	0
0	1	1
1	0	1
1	1	0

exclusive = excluding, i.e. the state a = b = 1

$x = \bar{a}b + a\bar{b}$

Symbol:



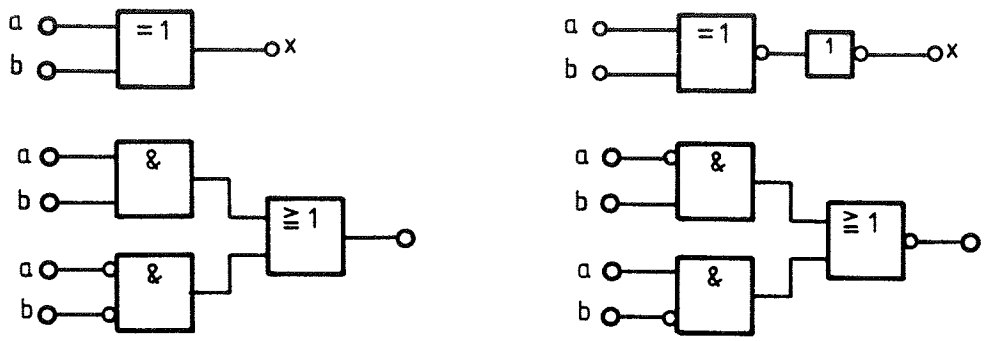
4.1.9 Exclusive-NOR (equivalence)

The exclusive-NOR operation results in a through-connection at the output, if all inputs are in the same state (0 or 1).

b	a	x
0	0	1
0	1	0
1	0	0
1	1	1

$x = ab + \bar{a}\bar{b}$ or $x = \overline{\bar{a}b + a\bar{b}}$

Symbol:



4.1.10 Negative logic

In negative logic the potential of the "0" and "1" are reversed.

a) AND-operation:

$x = a \cdot b \cdot c$

positive logic: normal "0" $\hat{=}$ 0 V
"1" $\hat{=}$ U
inverted "0" $\hat{=}$ U
"1" $\hat{=}$ 0 V

Negative logic AND-operation:

$x' = \bar{x} = \overline{a \cdot b \cdot c} = \bar{a} + \bar{b} + \bar{c}$ (corresponds to positive OR-operation)

b) OR-operation:

$x = a + b + c$

Negative logic OR-operation:

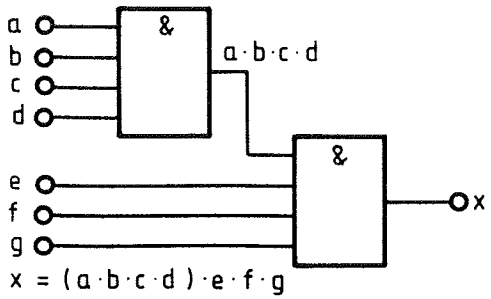
$x' = \bar{x} = \overline{a + b + c} = \bar{a} \cdot \bar{b} \cdot \bar{c}$ (corresponds to positive AND-operation)

It can be observed that in negative logic an AND element is the same as a positive OR element and vice versa. So in changing from positive to negative logic one needs only to exchange AND and OR elements.

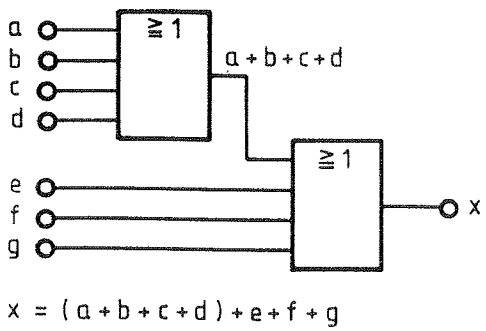
4.1.11 Extension of the inputs of AND and OR

In a circuit it may become necessary to put more inputs than are available on a gate for the extension of the inputs; the following circuit may be used:

AND

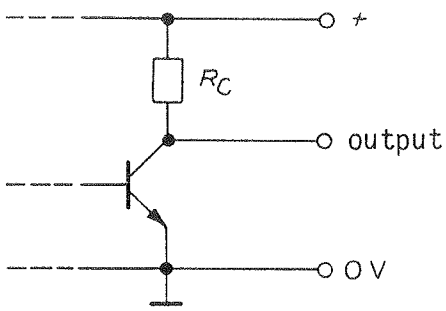


OR

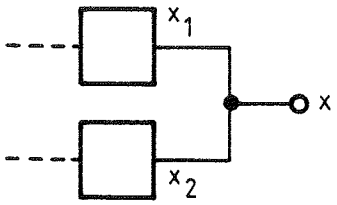


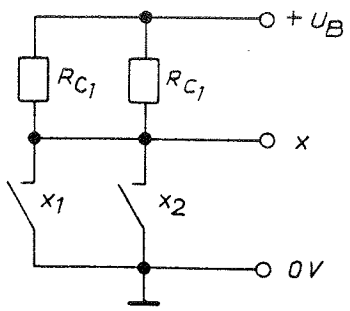
4.1.12 Wired-AND-operation (Phantom-AND-operation)

If the outputs x_1 and x_2 of two switching circuits are wired parallel, a new operation logic for the mutual output x is achieved. However, beforehand it has to be investigated, if such a parallel connection of the outputs is admissible and no break down occurs. The output of a number of switching circuit systems shows the following:



As there are 2 states only, the transistor is to be understood as a switch. How this appears with 2 parallel connected outputs is shown below:





Let positive logic be used:

$0 \mathrel{\hat{=}} \approx 0$ V (low)

$1 \mathrel{\hat{=}} \approx +U_B$ (high)

Function table

x_1	x_2	x
0	0	0
1	0	0
0	1	0
1	1	1

For the single switch:

x_1 or x_2 = 1 ⇒ switch open

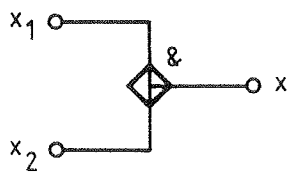
x_1 or x_2 = 0 ⇒ switch closed
(signal 0 dominates)

or:

$$x = x_1 \cdot x_2$$

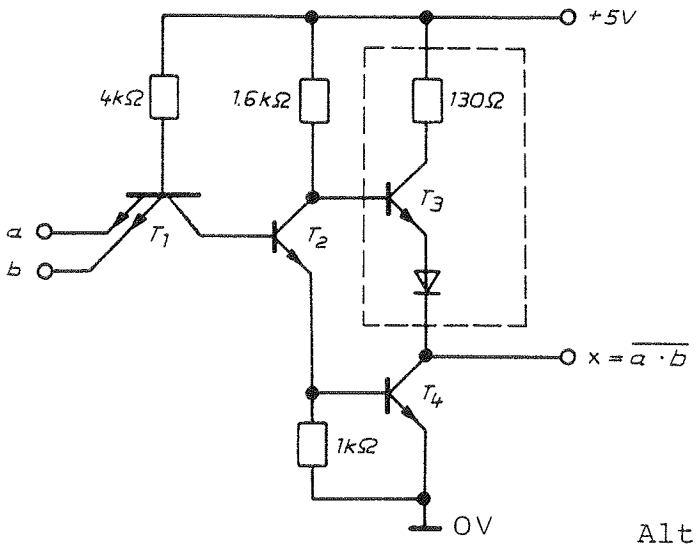
"Wired-AND-operation"

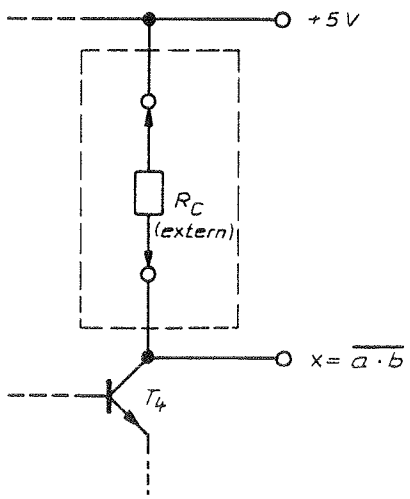
Symbol:



With TTL-switching circuits, the connection without any complication is only possible with gates with an open collector. With these an external mutual resistor R_C has to be provided.

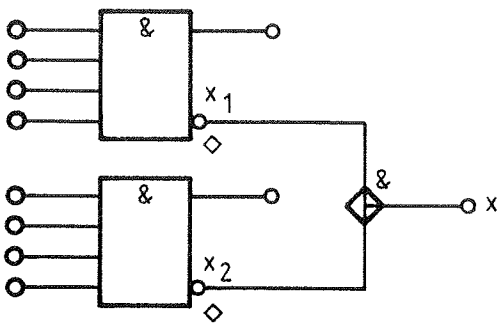
A TTL-NAND-element has the following structure:





Alteration to gates with "open collector"

The output circuit consists of an amplifier (T_3, T_4) where T_3, T_4 are controlled by T_2 such that T_3 conducts for "1" output and T_4 conducts for "0" output. So that the output for "0" and "1" becomes low-resistant and not suitable for wired-AND. This kind of operation nevertheless can be demonstrated with the device, because 2 gates with external collector resistor R_C are available (marked with R_C)



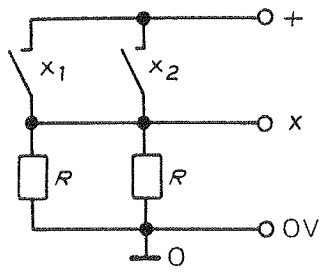
It is: $x = x_1 \cdot x_2$

$$x = \overline{a_1 \cdot b_1 \cdot c_1 \cdot d_1} \cdot \overline{a_2 \cdot b_2 \cdot c_2 \cdot d_2}$$

$$x = \overline{a_1 \cdot b_1 \cdot c_1 \cdot d_1 + a_2 \cdot b_2 \cdot c_2 \cdot d_2}$$

4.1.13 WIRED-OR-operation (Phantom-OR-operation)

If the switching transistor is located in the upper branch then a "wired-OR-operation" at positive logic is received.



x_1	x_2	x
0	0	0
1	0	1
0	1	1
1	1	1

$x_1, x_2 = 0 \rightarrow$ switch open

$x_1, x_2 = 1 \rightarrow$ switch closed
(signal 1 dominates)

$$x = x_1 + x_2$$ WIRED-OR

4.1.14 Switching functions

Input variable a	Input variable b	output – functions (Q) 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		constant	NOR	inhibition	negation (b)	inhibition	negation ($\bar{a}$)	anti-valence	NAND	AND	equi-valence	identity (a)	impli-cation	identity (b)	impli-cation	OR	constant
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
		+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V	+ 10V
operation symbol			$Q = \overline{a \vee b}$		$Q = \bar{b}$		$Q = \bar{a}$	$Q = a \neq b$	$Q = \overline{a \,\&\, b}$	$Q = a \,\&\, b$	$Q = a \equiv b$		$Q = a \leftarrow b$		$Q = a \rightarrow b$	$Q = a \vee b$	
gate symbol																	
operation with elementary switching function	$Q = 0$		$Q = \overline{a \vee b} = \bar{a} \,\&\, \bar{b}$	$Q = a \,\&\, \bar{b}$	$Q = \bar{b}$	$Q = \bar{a} \,\&\, b$	$Q = \bar{a}$	$Q = (a \vee b) \,\&\, (\bar{a} \vee \bar{b})$	$Q = \overline{a \vee b} = \bar{a} \,\&\, \bar{b}$	$Q = a \,\&\, b$	$Q = (a \,\&\, b) \vee (\bar{a} \,\&\, \bar{b})$	$Q = a$	$Q = a \vee \bar{b}$	$Q = b$	$Q = \bar{a} \vee b$	$Q = a \vee b$	$Q = 1$
other notations and symbols used in literature		Peir-cenian arrow $a \downarrow b$		inversion $\neg b$			inversion $\neg a$	disvalence exclusive OR $a \oplus b$	shetterian line $a \bar{\wedge} b$ $a \mid b$	conjunc-tion, log. multiplic. $a \wedge b$ $a \cdot b$ (ab)						disjunction log. addition inclusive OR $a + b$	

0 ≙ 0V
1 ≙ +10V

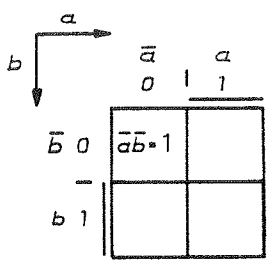
4.1.15 The Karnaugh-diagram

We have seen that with the aid of the disjunctive and conjunctive normal equation the output function can be set up from the truth-table. By use of the theorems of the Boolean Algebra the function can be reduced to a minimum. Essentially, simplification exists at the generally more favourable disjunctive normal equation by means of factoring out of terms of form $x + \bar{x} = 1$. This factoring out-method can be accomplished with the Karnaugh-diagram.

A rectangle is to be drawn, with as many fields as combinations of the input i.e. at n-input variations 2^n fields. The assignement of the fields to the corresponding combinations of the inputs has to be made in such a way that one field only is uniquely decided by one possible combination and at the transfer from one field to a neighbouring one, only one variable is altered.

Examples:

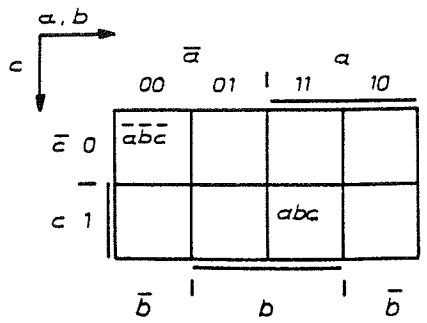
2 variables: a, b



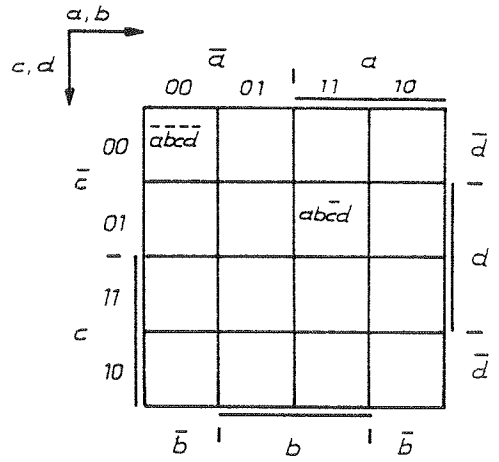
$\bar{a} \cdot \bar{b} = 1$ is:

a	b	x
0	0	1
0	1	0
1	0	0
1	1	0

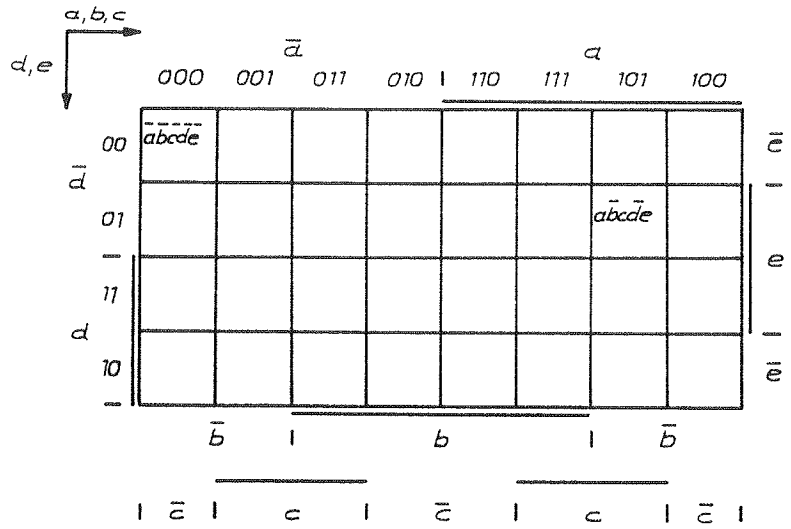
3 variables: a, b, c



4 variables: a, b, c, d



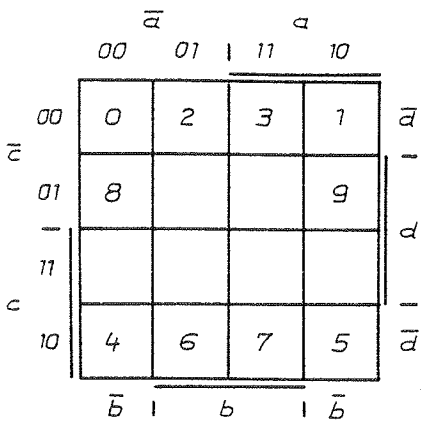
5 variables: a, b, c, d, e

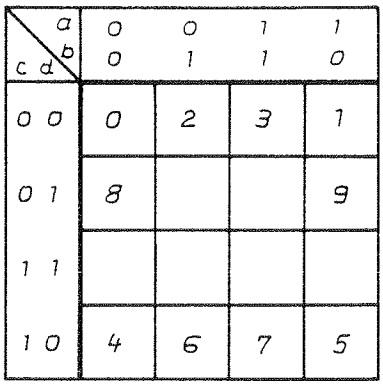


Each field is marked with the variables which stand at the border. The marking can be done in such a way that the value of the variables (1, 0) can be written in the field from left to right in alphabetical succession.
It is easier to write the variations of the vertical and horizontal directions like coordinates along the edges (attention - alphabetical succession!).

The marking of the fields is done as seen in an example for 4 variables.

a	b	c	d	
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9





Obviously, both kind of representation are identical.

Each AND-operation of the disjunctive normal equation (or each row of the truth-table for which the output function is "1") takes up exactly one field of the diagram. These fields are marked e.g. with a "1" (or x etc.). Neighbouring fields with a "1" are combined to form a block (block of two, block of four, block of eight).
The simplification consists of the fact that the variables which are common to all field of a block remain and those ones which change within the block disappear ($a \rightarrow \bar{a}$, $b \rightarrow \bar{b}$, ...). Therefore the blocks are chosen in a way, that only as few as possible of variables can be marked. Imagine the Karnaugh-diagram is shaped as a cylinder (horizontal or vertical direction), then these fields are neighbouring which are above or below of the same column resprectively at the beginning and end of the same row.
Fields used for one block are again useful for the set-up of another block if this indicates a simplification.

Example of application:

a	b	c	d	T
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
0	0	1	1	0
0	1	1	0	0
1	1	0	0	0
0	1	1	1	0
1	0	1	0	1
1	0	0	1	-
0	1	0	1	-
1	1	1	0	-
1	1	0	1	-
1	0	1	1	-
1	1	1	1	-

figure 1

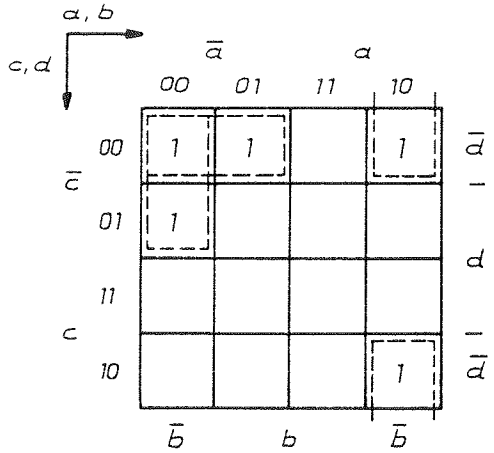
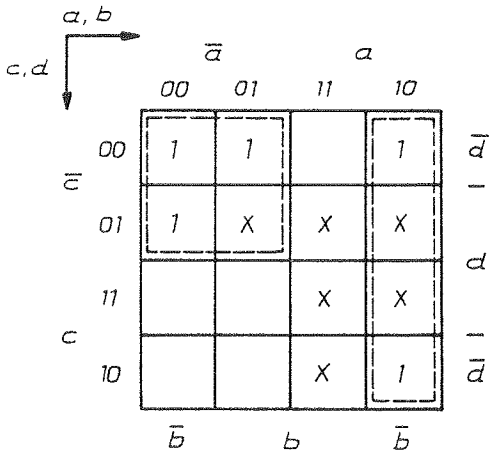


figure 2

not possible
("don't care")



The last 6 combinations are not possible on account of the circuit used.

From figure 1 we see that: $T = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{c}\bar{d} + a\bar{b}\bar{d}$

An additional simplification is received if the combinations which are not possible by means of the circuit is interpreted with "1" and included in the diagram, not being afraid to make a mistake. These don't care fields are marked by crosses (figure 2) and are to be used for blockformation, if this shows a simplification.

We receive with that: $T = \bar{a}\bar{c} + a\bar{b}$

4.2 Exercises

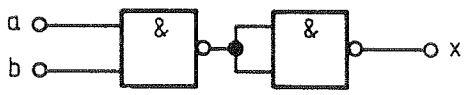
Exercise 4 - 1:

To realize the logic functions AND, OR, NOR, NOR and EXCLUSIVE OR in NAND-logic.

Solution 4 - 1:

a) AND-function

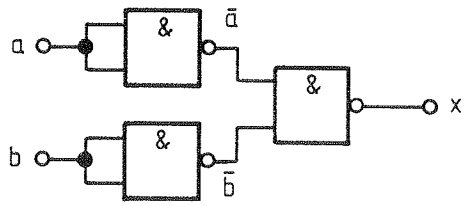
$$x = a \cdot b = \overline{\overline{a \cdot b}}$$



b) OR-function

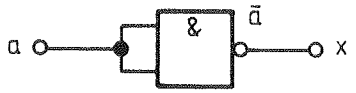
$$x = a + b = \overline{\overline{a + b}} = \overline{\bar{a} \cdot \bar{b}}$$

(De-Morgan's formula)



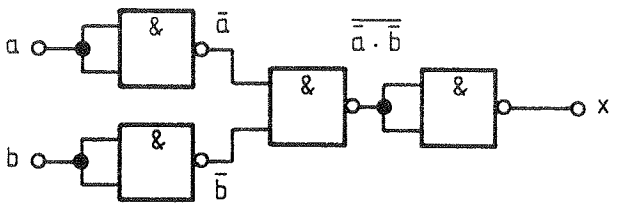
c) NOT-function

$$x = \bar{a}$$



d) NOR-function

$$x = \overline{a + b} = \bar{a} \cdot \bar{b} = \overline{\overline{\bar{a} \cdot \bar{b}}}$$

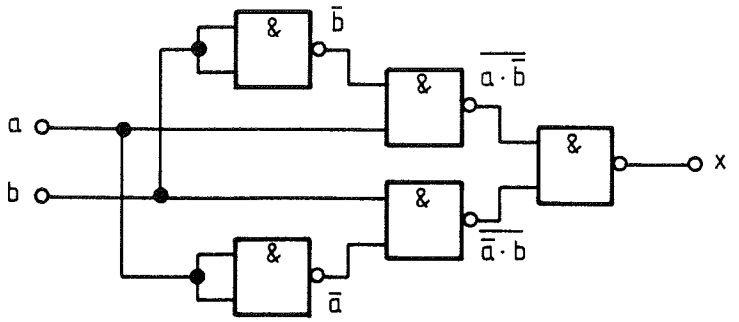


e) EXCLUSIVE-OR (antivalence)

$$x = \bar{a} \cdot b + a \cdot \bar{b}$$

$$= \overline{\overline{\bar{a} \cdot b + a \cdot \bar{b}}}$$

$$= \overline{\overline{(\bar{a} \cdot b)} \cdot \overline{(a \cdot \bar{b})}}$$



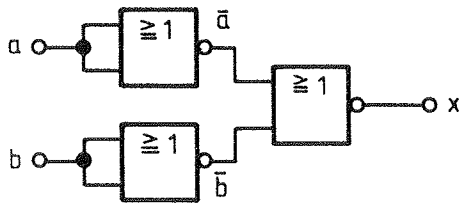
Exercise 4 - 2:

To realize the logic functions AND, OR, NOT, NAND and EXCLUSIVE OR in NOR-logic.

Solution 4 - 2:

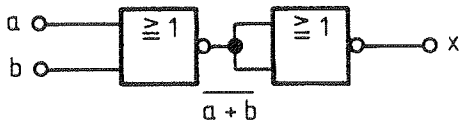
a) AND-function

$$x = a \cdot b = \overline{\overline{a \cdot b}} = \overline{\bar{a} + \bar{b}}$$



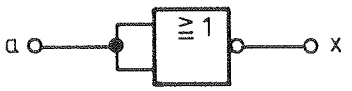
b) OR-function

$$x = a + b = \overline{\overline{a + b}}$$



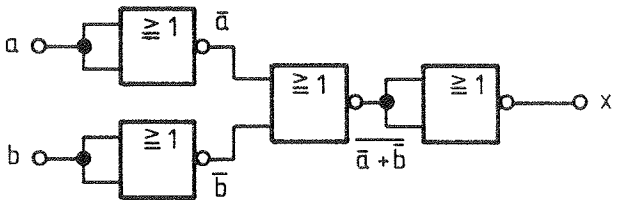
c) NOT-function

$$x = \bar{a}$$



d) NAND-function

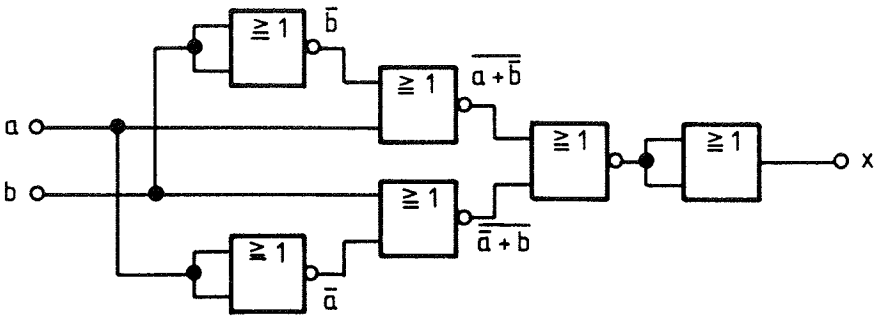
$$x = \overline{a \cdot b} = \bar{a} + \bar{b} = \overline{\overline{\bar{a} + \bar{b}}}$$



e) EXCLUSIVE-OR

$$x = \overline{\overline{\bar{a} \cdot b + a \cdot \bar{b}}} = \overline{\overline{\bar{a} \cdot b}} + \overline{\overline{a \cdot \bar{b}}} = \overline{a + \bar{b}} + \overline{\bar{a} + b}$$

$$x = a + \bar{b} + \bar{a} + b$$



Exercise 4 - 3:

To set up, from a given function-table, the disjunctive normal equation and to simplify the circuit by means of theorems of switching algebra.

function table

	a	b	c	x
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Solution 4 - 3:

Disjunctive normal equation

$x_0 = \bar{a} \cdot \bar{b} \cdot \bar{c}$

$x_3 = \bar{a} \cdot b \cdot c$

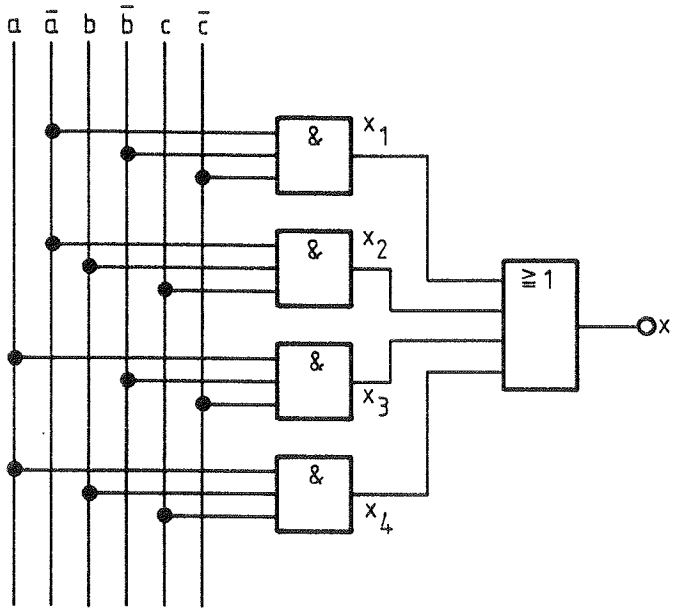
$x_4 = a \cdot \bar{b} \cdot \bar{c}$

$x_7 = a \cdot b \cdot c$

$x = x_0 + x_3 + x_4 + x_7$

$x = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot c$ (without simplification)

Circuit:

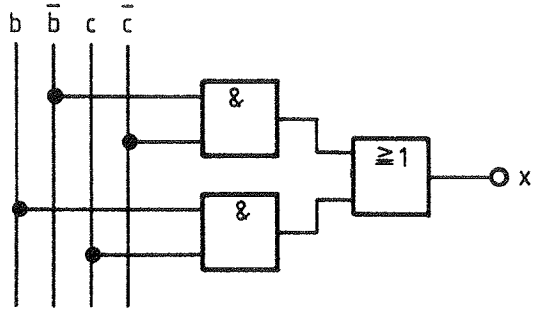


Simplification: (factoring out of common variables)

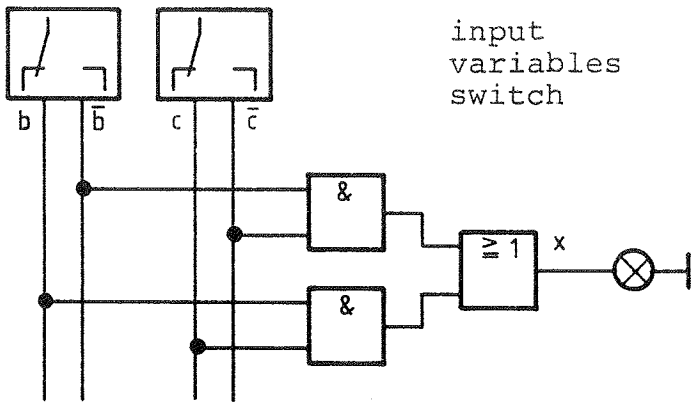
$$x = \bar{b} \cdot \bar{c}\ \underbrace{(\bar{a} + a)}_{1} + b \cdot c\ \underbrace{(\bar{a} + a)}_{1}$$

$$x = \bar{b} \cdot \bar{c} + b \cdot c$$

Principle circuit:



Realization by means of a circuit:



Exercise 4 - 4:

To set up from a given function table the conjunctive normal equation and to simplify the circuit by means of theorems of switching algebra.

	a	b	c	x
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Solution 4 - 4:

Conjunctive normal equation

$x_1 = a + b + \bar{c}$

$x_2 = a + \bar{b} + c$

$x_5 = \bar{a} + b + \bar{c}$

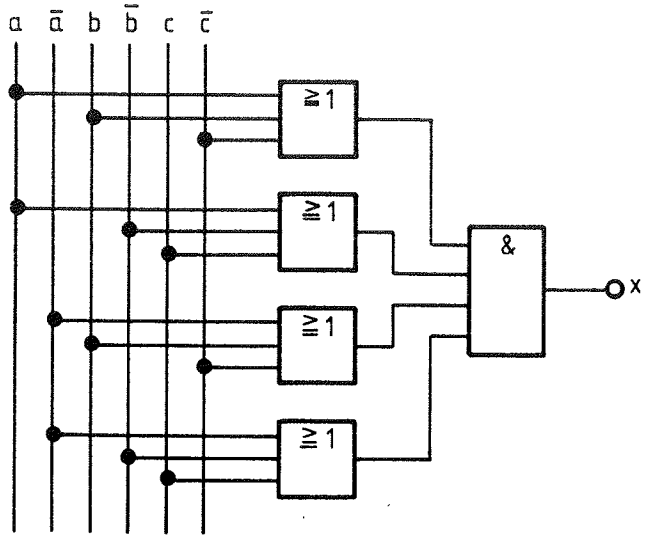
$x_6 = \bar{a} + \bar{b} + c$

$x = x_1 \cdot x_2 \cdot x_5 \cdot x_6$

$x = (a + b + \bar{c}) \cdot (a + \bar{b} + c) \cdot (\bar{a} + b + \bar{c}) \cdot (\bar{a} + \bar{b} + c)$

(without simplification)

Circuit



Comprehending:

1. Factorizing

$$x = (a + b + \bar{c}) \cdot (a + \bar{b} + c) \cdot (\bar{a} \cdot b + \bar{c}) \cdot (\bar{a} + \bar{b} + c)$$

$$x = (aa+a\bar{b}+ac+ab+\cancel{b\bar{b}}+bc+a\bar{c}+\bar{b}\bar{c}+\cancel{c\bar{c}}) \cdot (\bar{a}\bar{a}+\bar{a}\bar{b}+\bar{a}c+\bar{a}b+\cancel{b\bar{b}}+bc+\bar{a}\bar{c}+\bar{b}\bar{c}+\cancel{c\bar{c}})$$

2. Summarizing

$$x = [a + a\underbrace{(b + \bar{b})}_{1} + a\underbrace{(c + \bar{c})}_{1} + bc + \bar{b}\bar{c}] \cdot [\bar{a} + \bar{a}\underbrace{(b + \bar{b})}_{1} + \bar{a}\underbrace{(c + \bar{c})}_{1} + bc + \bar{b}\bar{c}]$$

$$x = (a + bc + \bar{b}\bar{c}) \cdot (\bar{a} + bc + \bar{b}\bar{c});$$

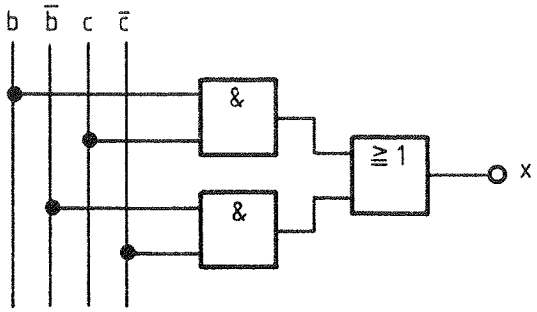
$$x = (a + B) \cdot (\bar{a} + B) \text{ with } B = bc + \bar{b}\bar{c}$$

auxiliary intermediate variable

With the aid of calculating rule 13 a) from page 38:

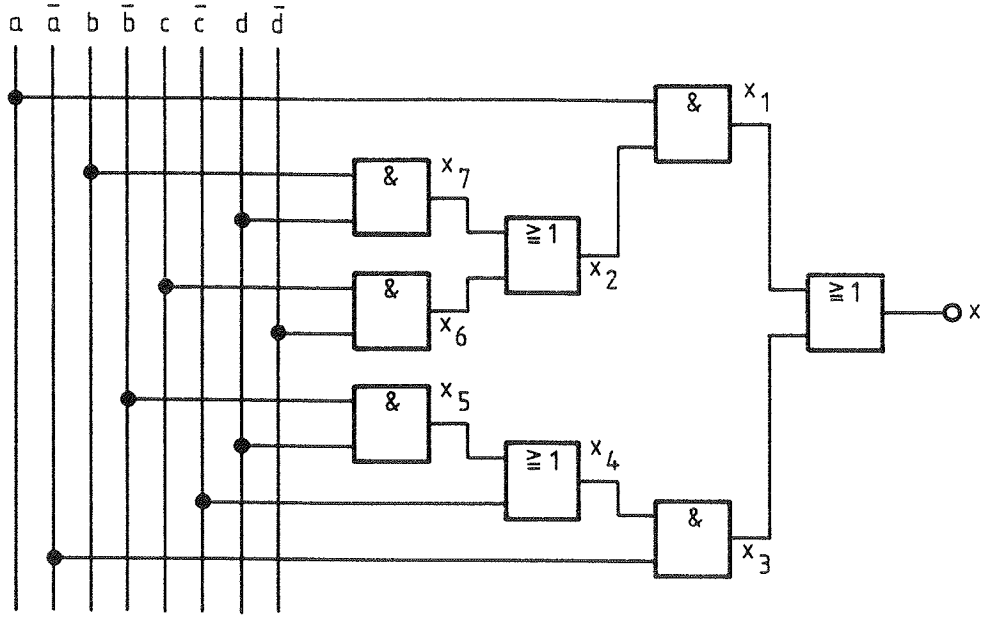
$$x = bc + \bar{b}\bar{c}$$

Circuit:



Exercise 4 - 5:

To develop the function table of the circuit below:



Solution 4 - 5:

$x = x_1 + x_3$

$x_1 = a \cdot x_2$

$x_2 = x_6 + x_7$

$x_3 = \bar{a} \cdot x_4$

$x_4 = x_5 + \bar{c}$

$x_5 = \bar{b} \cdot d$

$x_6 = c \cdot \bar{d}$

$x_7 = b \cdot d$

substituting:

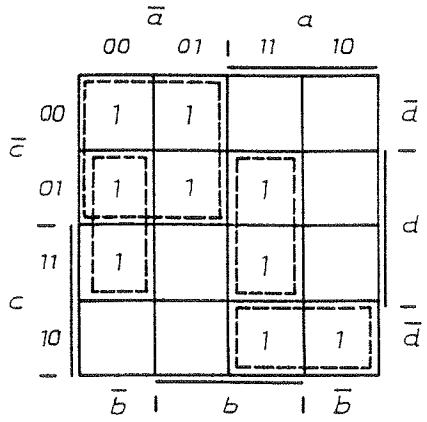
$x = a \cdot x_2 + \bar{a} \cdot x_4$

$= a \cdot (x_6 + x_7 + \bar{a} \cdot (x_5 + \bar{c})$

$= a \cdot (c \cdot \bar{d} + b \cdot d) + \bar{a} \cdot (\bar{b} \cdot d + \bar{c})$

factorize:

$x = ac\bar{d} + abd + \bar{a}\bar{b}d + \bar{a}\bar{c}$



From the Karnaugh-diagram follows the function table:

a	b	c	d	x
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

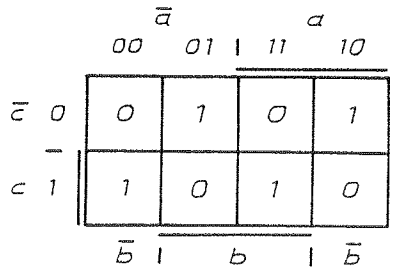
Exercise 4 - 6:

Let the output variable S of a logic circuit be dependent on the inputs a, b, c as follows: From each given combination of values of the inputs S should change its value, if only one of the three inputs or all three of them change their value, but if 2 inputs change their value, then S is to maintain its

present value. As a starting point take the condition:
If all three inputs have the value "0", S should also have the value "0". Develop the function table and design the circuit.

Solution 4 - 6:

Aid: The Karnaugh-diagram is built up such a way that the passing from one field into another neighbouring one, only one input changes its value. When writing in a Karnaugh-diagram in neighbouring fields starting at "000" alternatively "0" and "1", one has entered the function S.



Equation:

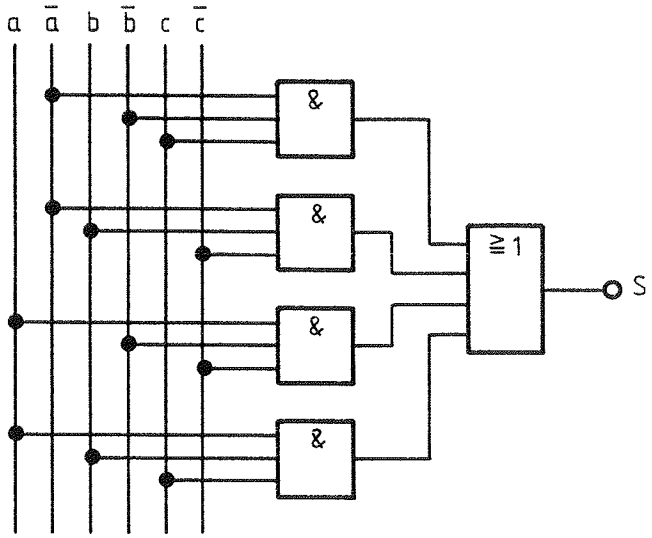
$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$ (1) a and $\bar{a}$ factorized:

or:

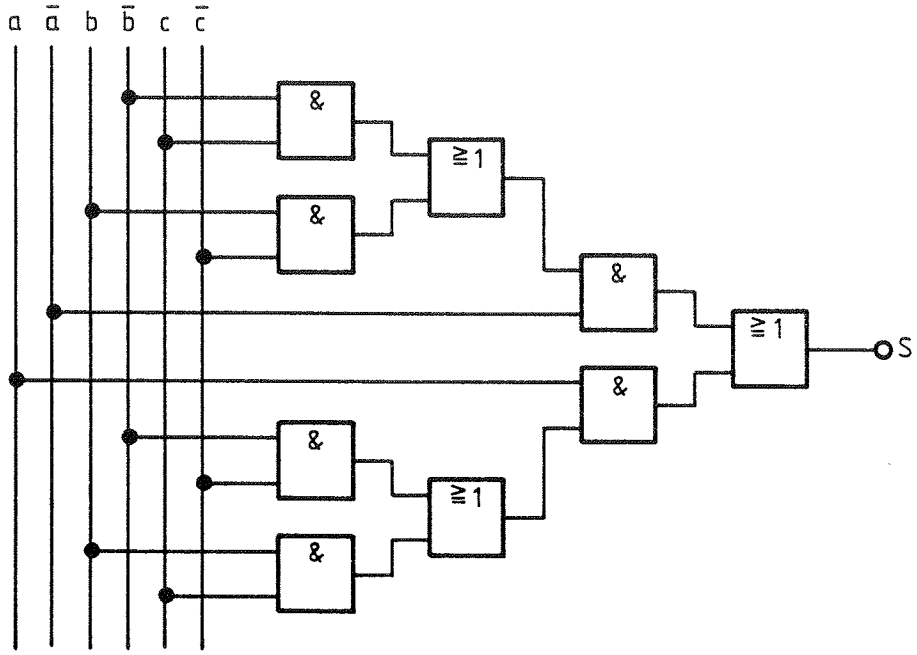
$S = \bar{a}\underbrace{(\bar{b}c + b\bar{c})}_{\text{antivalence}} + a\underbrace{(\bar{b}\bar{c} + bc)}_{\text{equivalence}}$ (2)

Circuit

1)



2)



Exercise 4 - 7:

Given the function t (a, b, c, d) in minimized disjunctive form:

$$t = \bar{a}c + bcd + a\bar{c}\bar{d}$$

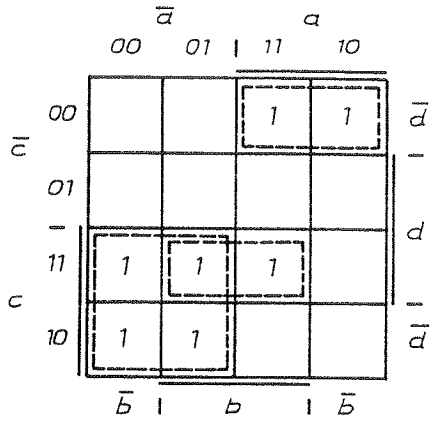
Alter this form simply into a conjunctive form. Write t in a Karnaugh-diagram, but minimize $\bar{t}$ and get out the minimized form of t by applying the Shannon theorems on $\bar{t}$.

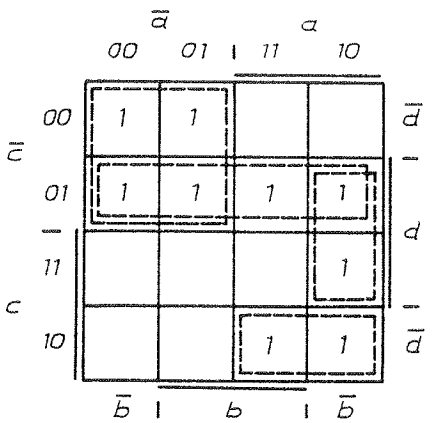
Solution 4 - 7:

t:

$\bar{t}$:

$\bar{t}$ form the missing fields of t in the Karnaugh-diagram





$$\bar{t} = \bar{a}\bar{c} + \bar{c}d + a\bar{b}d + ac\bar{d}$$

$$t = \bar{\bar{t}} = \overline{\bar{a}\bar{c} + \bar{c}d + a\bar{b}d + ac\bar{d}}$$

$$= (a + c) \cdot (c + \bar{d}) \cdot (\bar{a} + b + \bar{d}) \cdot (\bar{a} + \bar{c} + d)$$

Exercise 4 - 8:

Let a special priority circuit have the following inputs x_1, x_2, x_3 and outputs y_0, y_1, y_2, y_3 respectively. If all inputs have value "0" only the output y_0 should have value "1". If one or several inputs x_k take up value "1" and the highest index of these input variables is k, then only the output variable y_k should have value "1" and all other output variables should be zero. Evaluate the minimal circuit.

Solution 4 - 8:

Find the function table for this exercise:

input			output			
x_1	x_2	x_3	y_0	y_1	y_2	y_3
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	0	0	0	1	0	0
1	0	1	0	0	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$y_0 = \bar{x}_1 \bar{x}_2 \bar{x}_3$$

$$y_1 = x_1 \bar{x}_2 \bar{x}_3$$

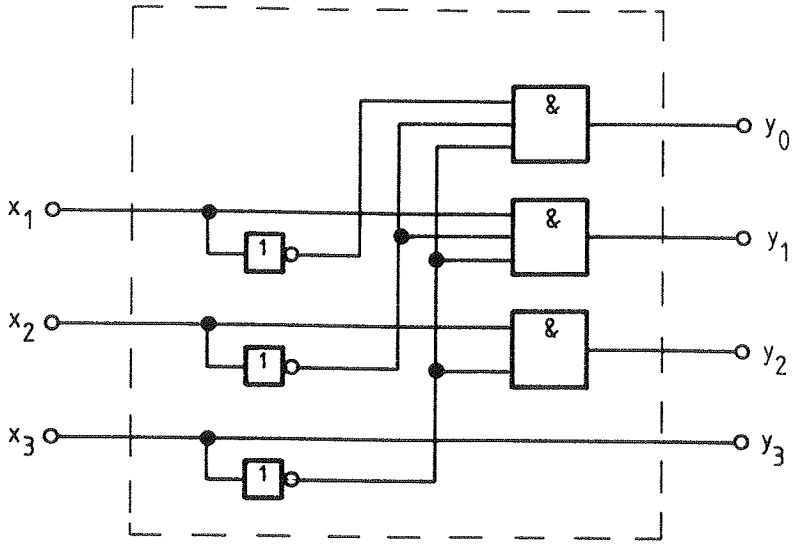
$$y_2 = \bar{x}_1 x_2 \bar{x}_3 + x_1 x_2 \bar{x}_3 = x_2 \cdot \bar{x}_3 \underbrace{(\bar{x}_1 + x_1)}_{=0}$$

$$y_3 = x_2 \bar{x}_3$$

From the table the following is read immediately

$$y_3 = x_3$$

Circuit:



5. CODING

5.1 General

Physical quantities can be represented and processed by analogous and digitalized means. An example of the analogue display is the temperature measurement done with a quick-silver thermometer.
The digital representation of a quantity is done by a number with a certain "resolution" e.g. by a counting device, which on its last figure can jump by one unit only. Intermediate values are not possible (e.g. km-counter).
The representation of digital quantities in engineering is simply done in the so-called "Binary system", which only knows two states, namely "0" and "1" (technically "on" and "off").
With n elements (position) - all of them can take two different states - an "alphabet" from $Z = 2^n$ different signs can be represented. Each sign can be assigned to another sign, which need not be based on the binary system. This assignement is named code. A code word consists of a number of binary signs "0" and "1".

Example for a binary code:

a = 110; b = 011; c = 101 etc.

With these three numbers an alphabet of

$Z = 2^3 = 8$ signs can be coded

(Code words are to be used for letters, operations, decimal numbers, etc.). Analogous to the decimal system which is arranged according to falling powers of 10:

$5673 = 5 \cdot 10^3 + 6 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$,

the binary system is arranged according to falling powers of 2:

$101101 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
(decimal 45) = 32 + 8 + 4 + 1

To represent a number in the binary system, two current methods are used:

a) closed binary

e.g. the representation of number 105 as a binary number in the Binary code:

$$105_{10} = 1101001$$

Besides the Binary code there are other binary codes e.g. the Gray-codes. From them two successive values differ by one bit-place only, to make them better suitable than the Binary code for photoelectrical and mechanical counters. From the many possible Gray-codes the specified one is known as the Gray-code. (page 66)

b) Binary Coded Decimal representation (BCD)

Hereby each decimal place (0...9) is represented by a code. If as a Binary code the first 10 figures of the binary system are used, then the decimal numbers 753 reads:

753_{10} = 0111 0101 0011 (8421-Code)
binary "7" "5" "3"

The assignment of tetrades (words consisting of 4 bits) to decimal figures is arbitrary, binary is only one possibility another possibility is the Gray-code already known. The following tables show some of the derived possibilities from the binary system.

Often used BCD-Codes

	Decimal numbers	Gray-code	Two out of 5 codes $\binom{5}{2}$	Biquinary
number of places		4	5	7
place value		-	74210	50 43210
	0	0000	11000	01 00001
	1	0001	00011	01 00010
	2	0011	00101	01 00100
	3	0010	00110	01 01000
	4	0110	01001	01 10000
	5	0111	01010	10 00001
	6	0101	01100	10 00010
	7	0100	10001	10 00100
	8	1100	10010	10 01000
	9	1101	10100	10 10000
	10	1111		
	11	1110		
	12	1010		
	13	1011		
	14	1001		
	15	1000		
advantage		immune to reading mistake	capable of being tested	simple arithmetic rule, capable of being tested
of no advantage for		calculate	calculate	store
arithmetic rule		not general applicable	not general applicable	simple place shifting

	Decimal numbers	Binary numbers	Binary-Code	Stibitz- or Excess-3-Code	Aiken-Code
number of places			4	4	4
place value			8421	$84\bar{2}\bar{1}$	2421
	0	0000	0 0000	-	0 0000
	1	0001	1 0001	-	1 0001
	2	0010	2 0010	-	2 0010
	3	0011	3 0011	0 0011	3 0011
	4	0100	4 0100	1 0100	4 0100
	5	0101	5 0101	2 0101	-
	6	0110	6 0110	3 0110	-
	7	0111	7 0111	4 0111	-
	8	1000	8 1000	5 1000	-
	9	1001	9 1001	6 1001	-
	10 ≙ A	1010	-	7 1010	-
	11 ≙ B	1011	-	8 1011	5 1011
	12 ≙ C	1100	-	9 1100	6 1100
	13 ≙ D	1101	-	-	7 1101
	14 ≙ E	1110	-	-	8 1110
	15 ≙ F	1111	-	-	9 1111
advantage			calculate, store	calculate, store, forming the complement	
of no advantage for			not capable of being tested		
arithmetic rules			If a pseudo-tetrade or a carry over appears binary 6 has to be added. This correction may have consequences for the carry over place. A further correction addition is not allowed to follow.	If a carry over appears a binary 3 has to be added. In other cases binary 3 has to be subtracted	A pseude-tetrade appears: If no carry over appears binary 6 has to be added. In other cases binary 3 has to be subtracted.
				The carry over place will not be changed by a correction	

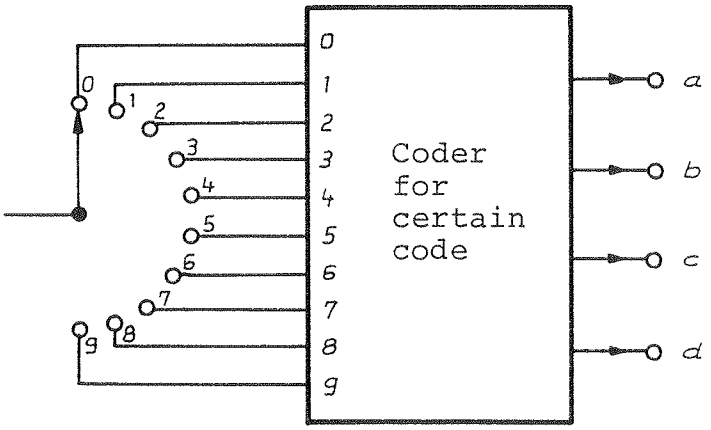
Further BCD-Codes

Decimal-Code	Petherick-Code	White-Code	Glixon-Code	Walking-Code	Telegraph-Code	Hamming-Code
0	0101	0000	0000	00011	01101	0000000
1	0001	0001	0001	00101	11101	0000111
2	0011	0011	0011	00110	11001	0011001
3	0010	0101	0010	01010	10000	0011110
4	0110	0111	0110	01100	01010	0101010
5	1110	1000	0111	10100	00001	0101101
6	1010	1001	0101	11000	10101	0110011
7	1011	1011	0100	01001	11100	0110100
8	1001	1101	1100	10001	01100	1001011
9	1101	1111	1000	10010	00011	1001100
10						1010010
11						1010101
12						1100000
13						1100110
14						1111000
15						1111111

Each one of the decimal-codes has its advantage with respect to addition, subtraction and forming the complement or detection of faults (here not treated in details). The current tetradic codes (of 4 bit length) have 6 pseudo-tetrades, because from the 16 possible combinations 6 are not used. This not-using of all possible combinations is called redundance.

5.2 Coder

From the many possible types of information only two are in practical use, namely the so-called alphanumeric signs and numbers. All other types of information will be translated into alphanumeric via a previously agreed code table. Alphanumeric signs include big and small letters, the numbers from 0 to 9 and also a number of special signs such as space, line feed, full stop, comma etc. For these signs there exist some international codes e.g. the telegraph code or ASC II Code. A coder takes over the function to convert these information sorts into a certain code.



Coder for digit-by-digit translation of the decimal numbers into a tetradic-decadic code.

The subsequent table shows the assignement of the decimal numbers 0 to 9 for the binary code. Basically the numbers also correspond to a code assignement, namely "1" to "10" Code, whereby the redundance is extremely large ($2^{10} = 1024$, but only 10 combinations are used).

Table

9	8	7	6	5	4	3	2	1	0	2^3 d	2^2 c	2^1 b	2^0 a
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

5.3 Exercises

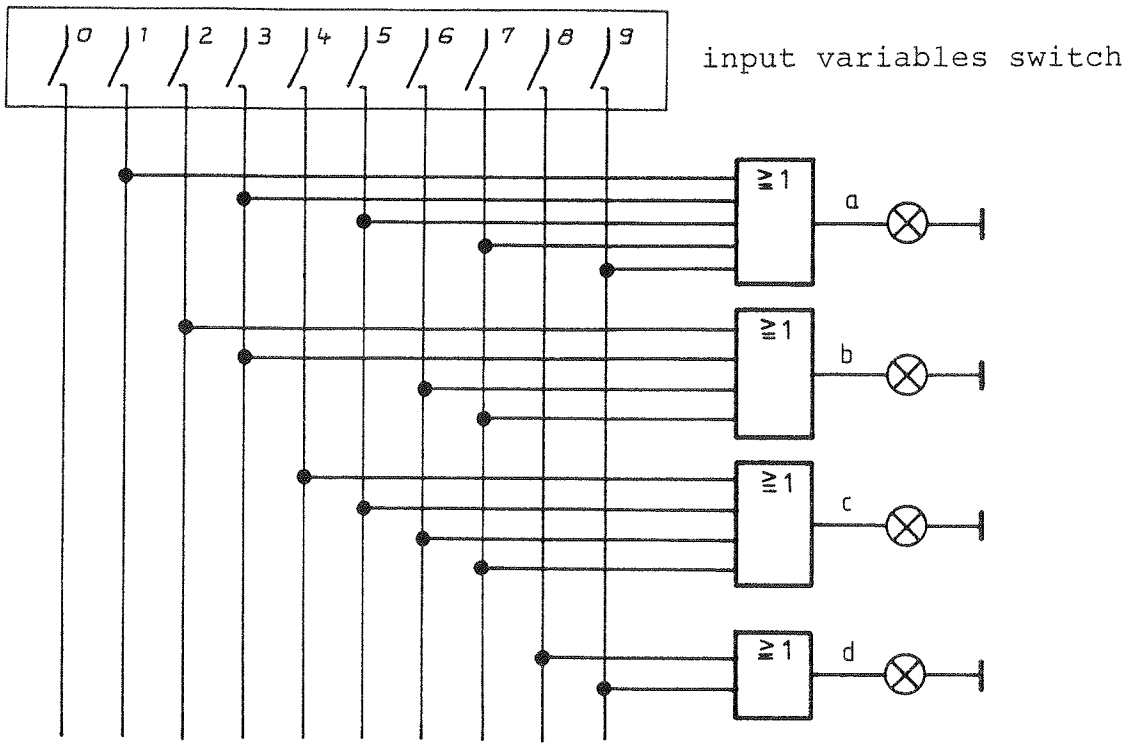
Exercise 5 - 1:

Design a coder, that converts the decimal numbers from 0 to 9 in the corresponding binary numbers.

Solution 5 - 1:

Dec. System	2^3 d	2^2 c	2^1 b	2^0 a
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

From the table it can be seen, when the outputs a, b, c, d must switch through. The operation is done by means of the disjunctive normal equation, i.e. the inputs 0...9 are to be connected with each other by means of the OR-gate. Of course the inputs 0...9 are to occur individually (1 out of 10).



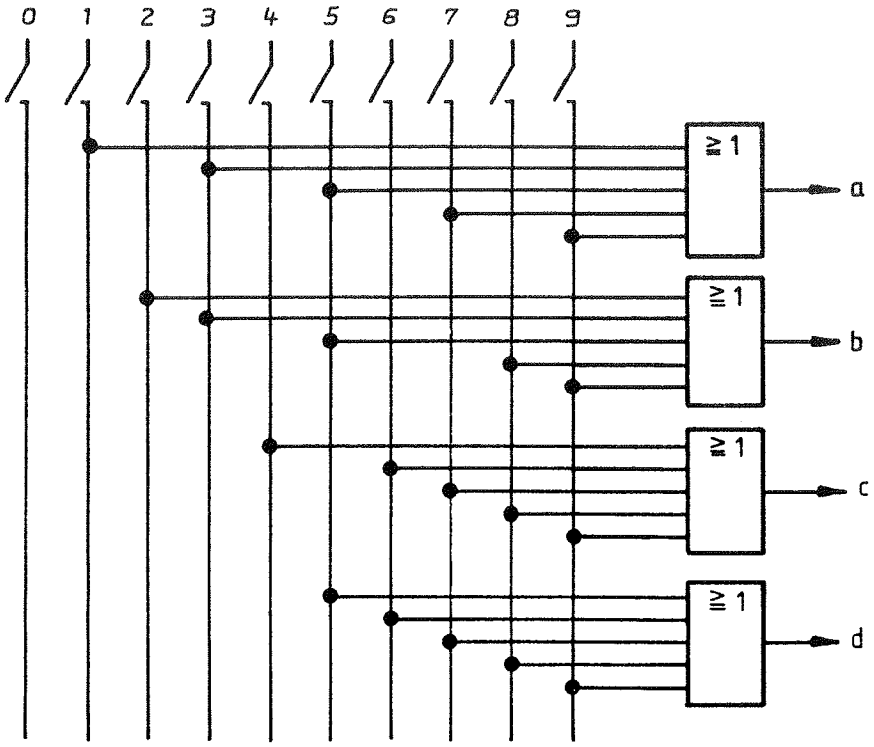
Exercise 5 - 2:

Design a code-circuit for converting Decimal → Aiken code (2421-code).

Solution 5 - 2:

Dec. System	d	c	b	a
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

The set up of equation is according to the preceding coding.

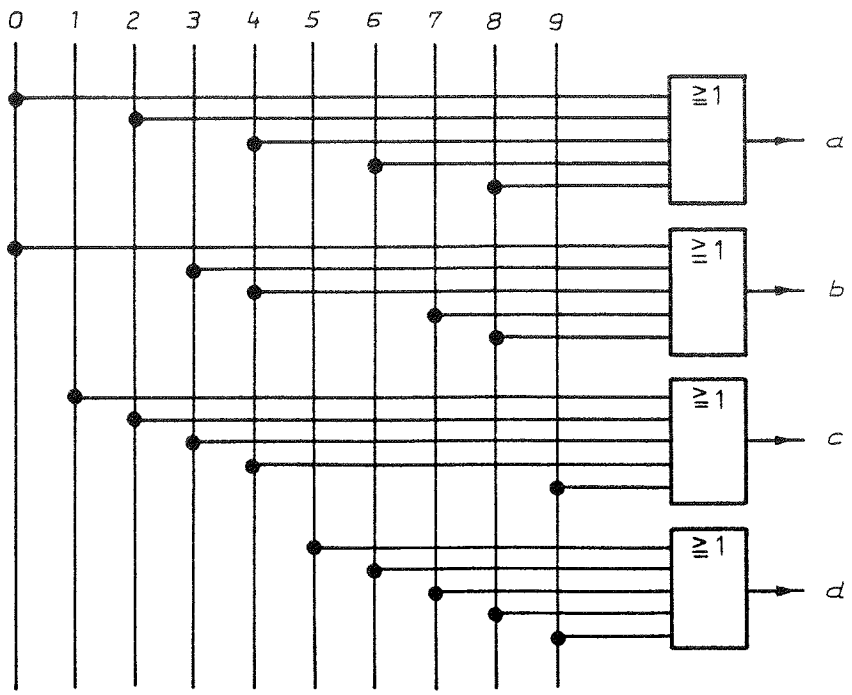


Exercise 5 - 3:

Design a code-circuit of converting Decimal → Excess-3-Code.

Solution 5 - 3:

Dec. System	d	c	b	a
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

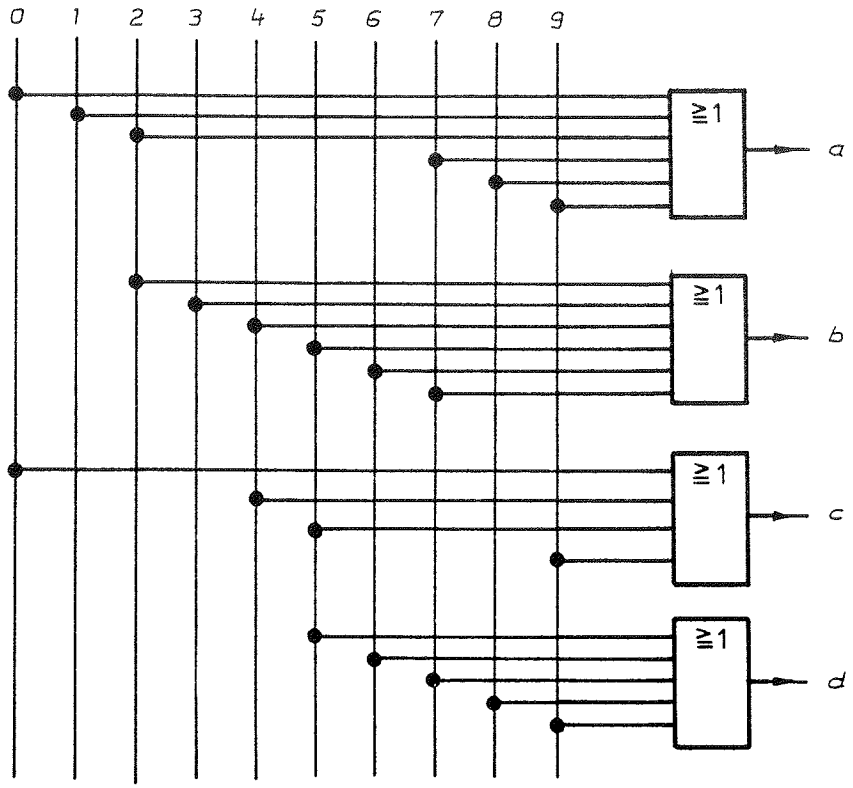


Exercise 5 - 4:

Design a code-circuit of converting Decimal → Petherick-Code.

Solution 5 - 4:

Dec. System	d	c	b	a
0	0	1	0	1
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	1	1	1	0
6	1	0	1	0
7	1	0	1	1
8	1	0	0	1
9	1	1	0	1

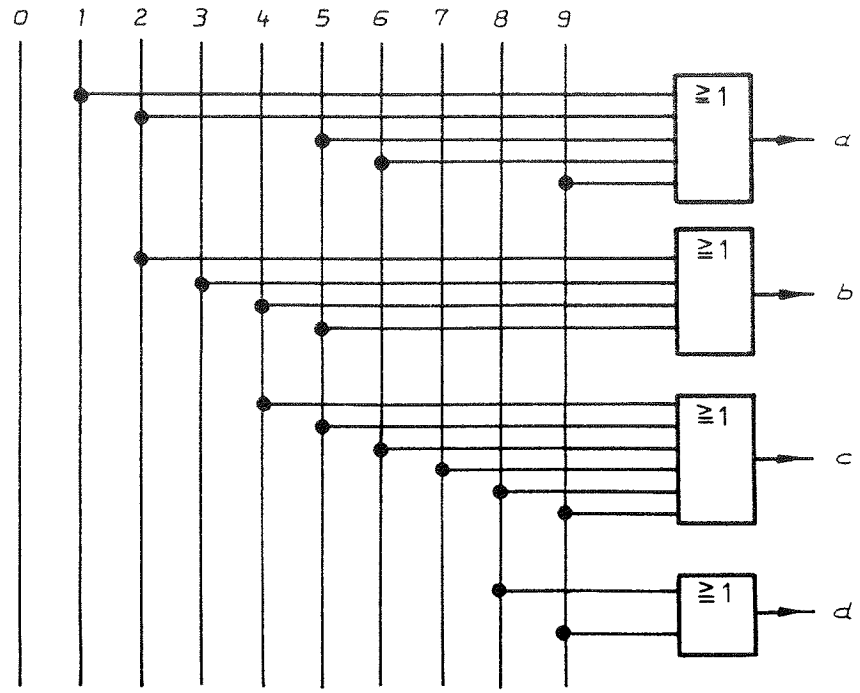


Exercise 5 - 5:

Design a code-circuit of converting Decimal → Gray-Code.

Solution 5 - 5:

Dec. System	d	c	b	a
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1

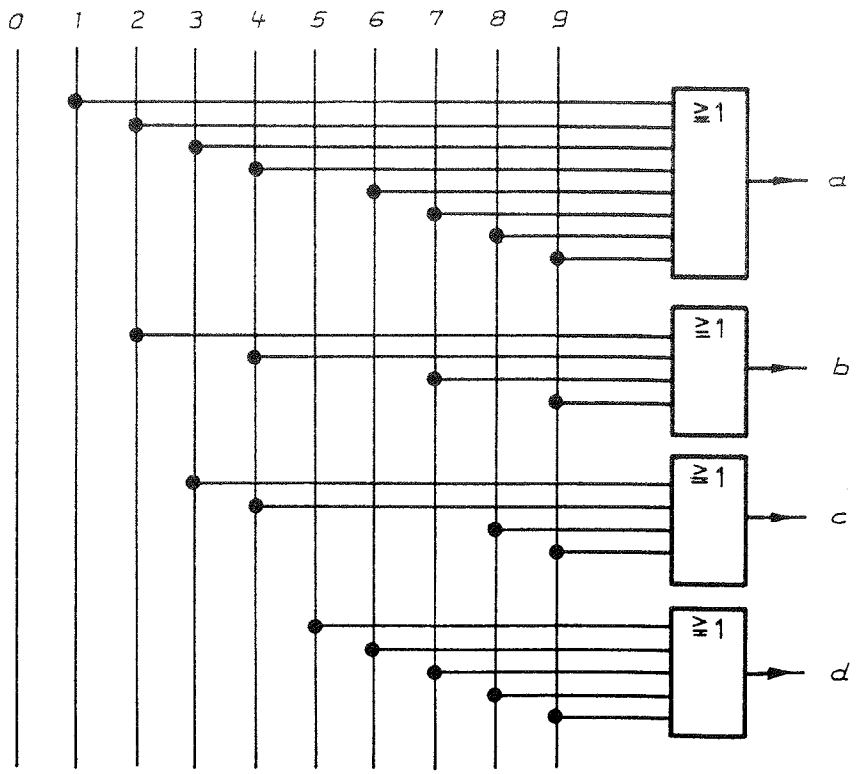


Exercise 5 - 6:

Design a code-circuit of converting Decimal → White-Code.

Solution 5 - 6:

Dec. System	d	c	b	a
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	1
8	1	1	0	1
9	1	1	1	1

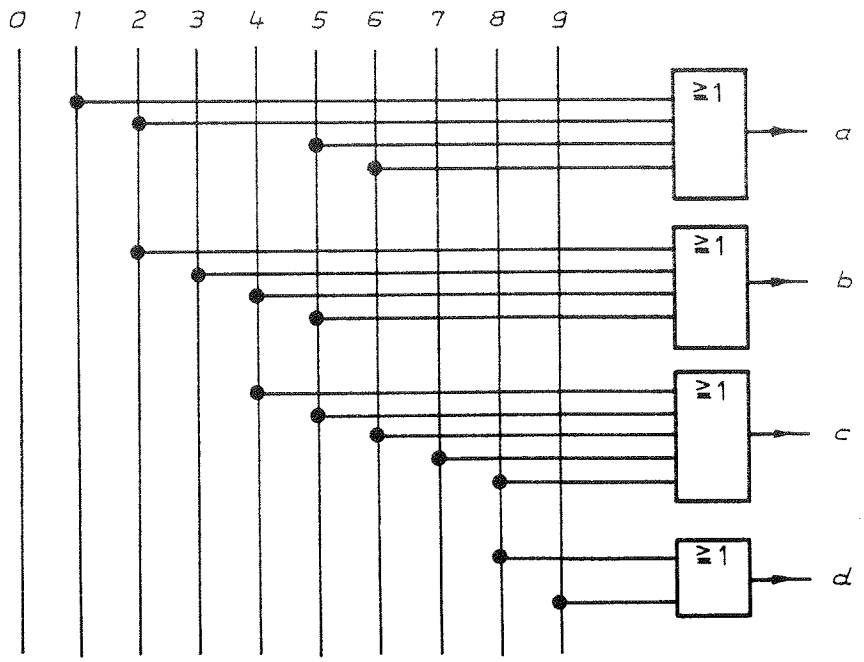


Exercise 5 - 7:

Design a code-circuit of converting Decimal → Glixon-Code.

Solution 5 - 7:

Dec. System	d	c	b	a
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	0	0	0

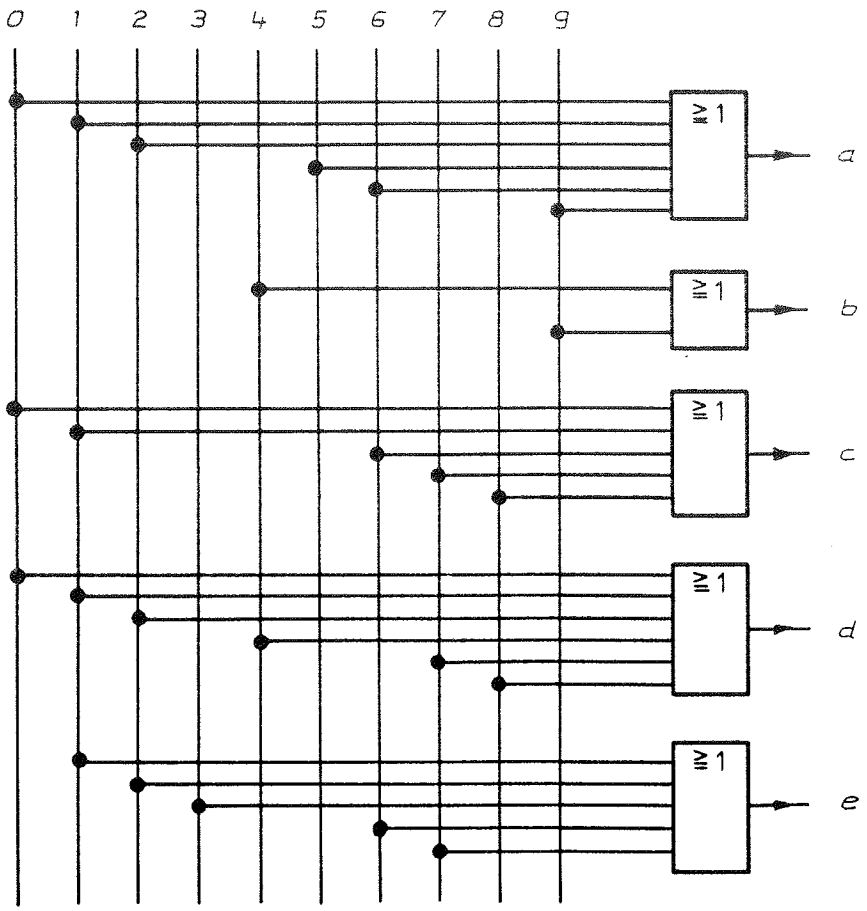


Exercise 5 - 8:

Design a code-circuit of converting Decimal → Telegraph-Code (only figures).

Solution 5 - 8:

Dec. System	e	d	c	b	a
0	0	1	1	0	1
1	1	1	1	0	1
2	1	1	0	0	1
3	1	0	0	0	0
4	0	1	0	1	0
5	0	0	0	0	1
6	1	0	1	0	1
7	1	1	1	0	0
8	0	1	1	0	0
9	0	0	0	1	1



Exercise 5 - 9:

Design a code-circuit of converting Decimal → Hamming-Code.

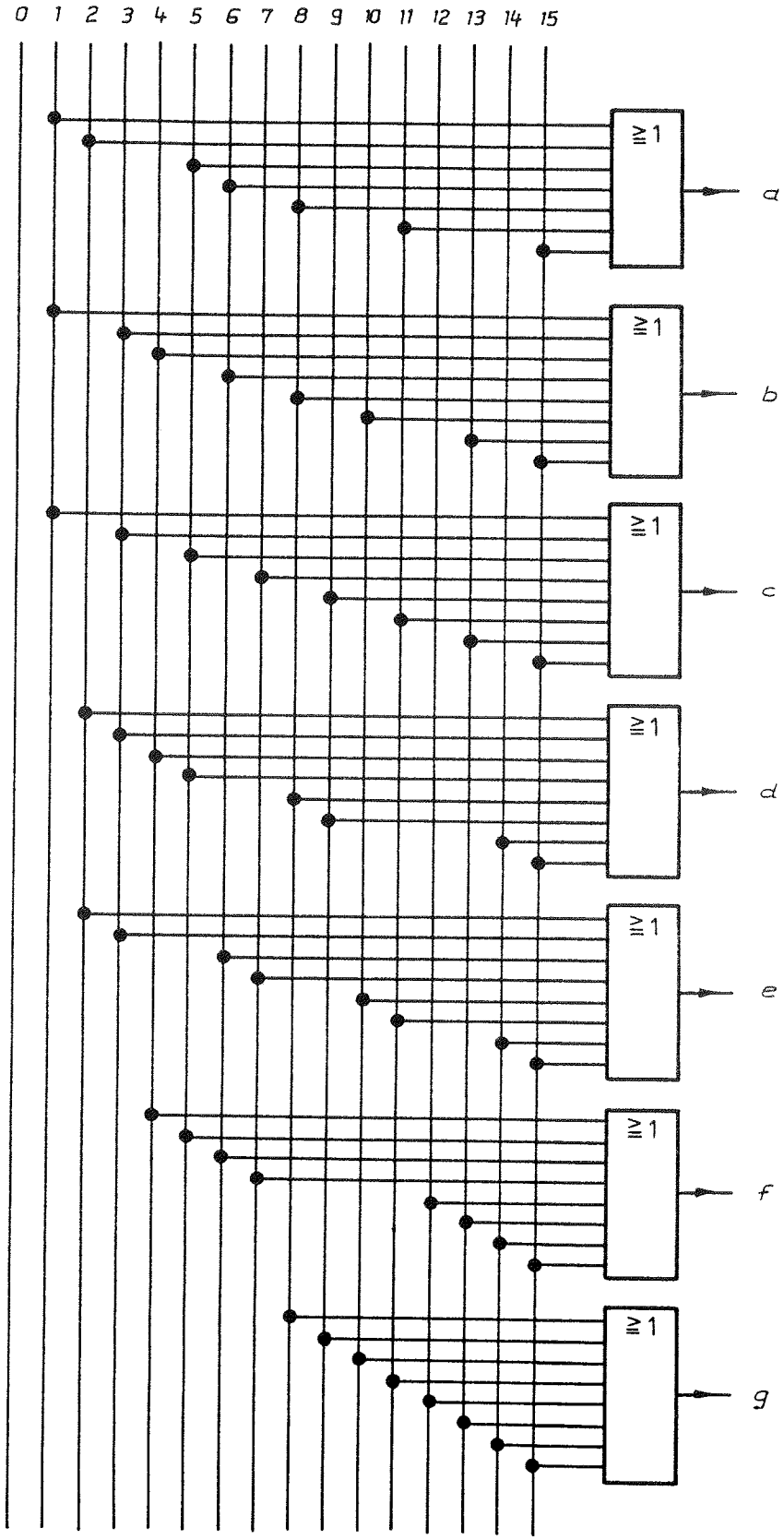
Solution 5 - 9:

Dec. System	g	f	e	d	c	b	a
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	0	1	0	1
12	1	1	0	0	0	0	0
13	1	1	0	0	1	1	0
14	1	1	1	1	0	0	0
15	1	1	1	1	1	1	1

If faulty transmission of a code word occurs using this extensive code, two errors can be detected and one corrected. The so-called Hamming-Distance is d = 3. With n = 7 places it would be possible to construct z = 128 combinations, nevertheless only z' = 16 possibilities are used.

$$z = 2^n = 2^7 = 128$$

$$z' = 2^{n-d} = 2^{7-3} = 16$$


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
≧1
a
≧1
b
≧1
c
≧1
d
≧1
e
≧1
f
≧1
g

6. DECODER

6.1 General

Decoders have the task to convert a number code of n code words into a 1-out-of-n-code. Therefore the decoder has n outputs. Since 1-out-of-n-codes usually are not suitable for subsequent digital treatment, normally decoder circuits are installed at the output of the digital part of an installation. The 1-out-of-10-code is frequently used for the display of a counting or calculation result (for direct decimal display) but also the so-called 7 segment code is now frequently used.

Example: 8-4-2-1-Code → decimal system (1-out-of-10-code)

d	c	b	a	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

6.2 Exercises

Exercise 6 - 1:

Develop a decoder circuit which converts the 8-4-2-1-code into the decimal system and minimize this circuit by means of a Karnaugh-Diagram.

Solution 6 - 1:

The most used code in digital technics is the binary coded decimal code with the valence 8-4-2-1 derived from the binary system. The conversion of this code into the familiar decimal numbers is done using decoders.

a 2^0	b 2^1	c 2^2	d 2^3	Dec. System
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	1	7
0	0	0	1	8
1	0	0	1	9

Equations:

$x_0 = \bar{a}\bar{b}\bar{c}\bar{d}$ $\quad$ $x_5 = a\bar{b}c\bar{d}$

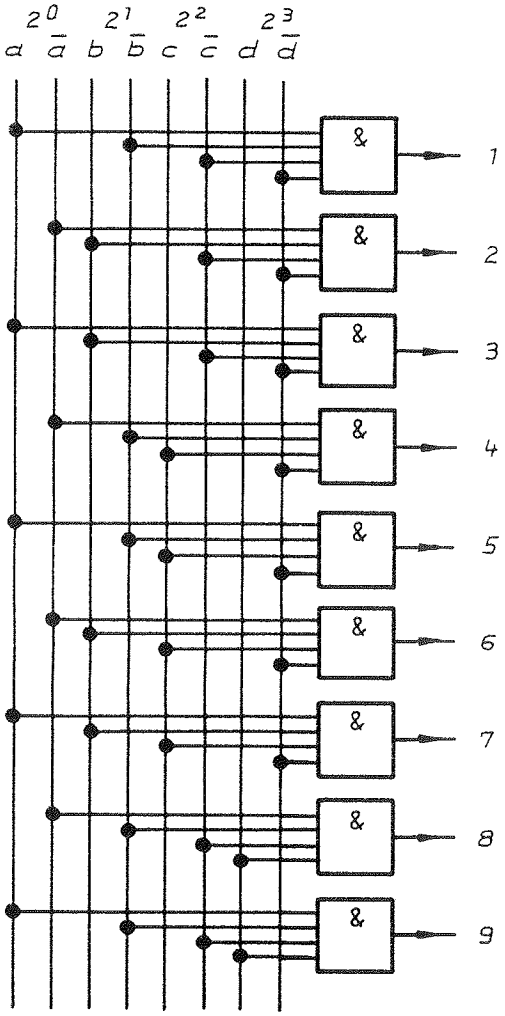
$x_1 = a\bar{b}\bar{c}\bar{d}$ $\quad$ $x_6 = \bar{a}bc\bar{d}$

$x_2 = \bar{a}b\bar{c}\bar{d}$ $\quad$ $x_7 = abc\bar{d}$

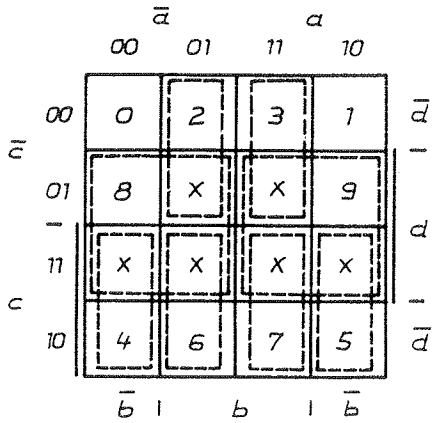
$x_3 = ab\bar{c}\bar{d}$ $\quad$ $x_8 = \bar{a}\bar{b}\bar{c}d$

$x_4 = \bar{a}\bar{b}c\bar{d}$ $\quad$ $x_9 = a\bar{b}\bar{c}d$

Principal circuit without simplification

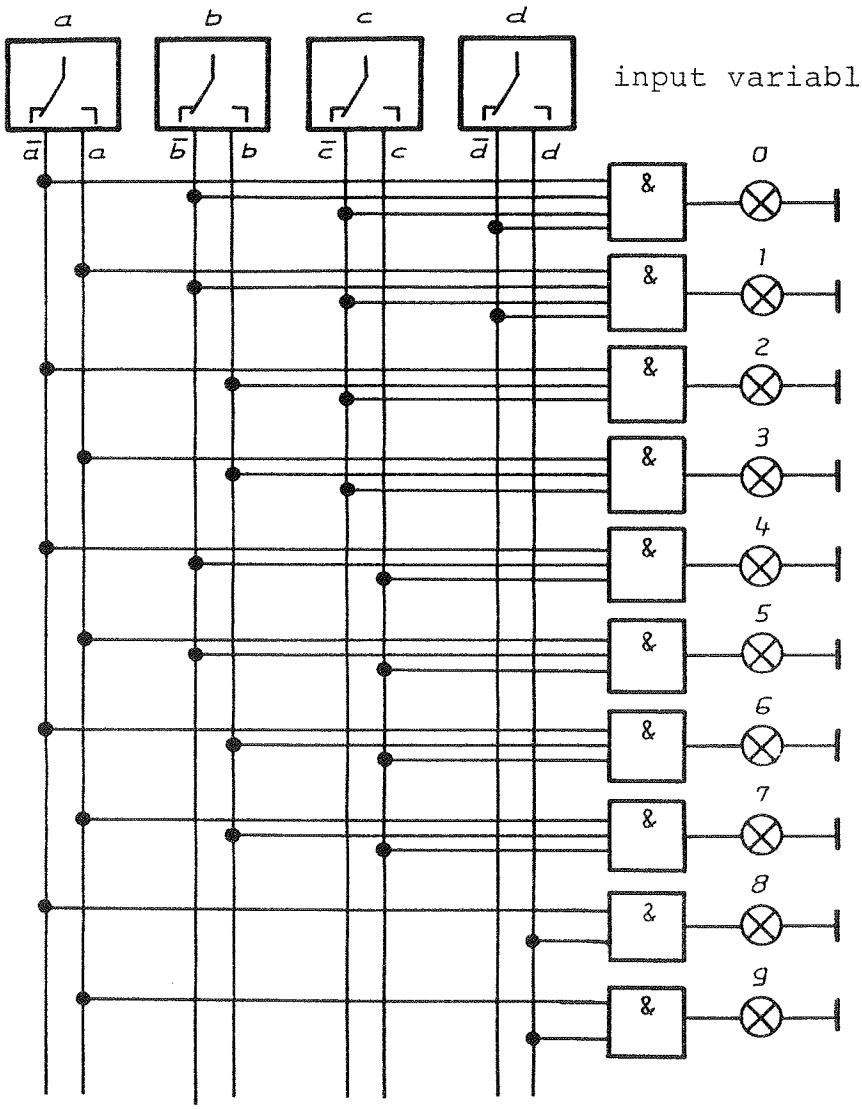


Since with the 8-4-2-1-code from 16 possible combinations only these from 0 to 9 appear, the remaining 6 combinations from 10 to 15 (the so-called pseudo tetrades) may be used (for simplification purposes) as "don't care fields" in the Karnaugh-Diagram (marked with X).



$x_0 = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$
$x_1 = a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$
$x_2 = \bar{a} \cdot b \cdot \bar{c}$
$x_3 = a \cdot b \cdot \bar{c}$
$x_4 = \bar{a} \cdot \bar{b} \cdot c$
$x_5 = a \cdot \bar{b} \cdot c$
$x_6 = \bar{a} \cdot b \cdot c$
$x_7 = a \cdot b \cdot c$
$x_8 = \bar{a} \cdot d$
$x_9 = a \cdot d$

Circuit (minimalized):



Exercise 6 - 2:

Develop an error detection circuit of pseudo tetrads for the 8-4-2-1/decimal-decoder.

Solution 6 - 2:

The "Don't care conditions" are simultaneously signals for the error detection f.

The completed table is as follows:

d	c	b	a	Dec.
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	(10) f
1	0	1	1	(11) f
1	1	0	0	(12) f
1	1	0	1	(13) f
1	1	1	0	(14) f
1	1	1	1	(15) f

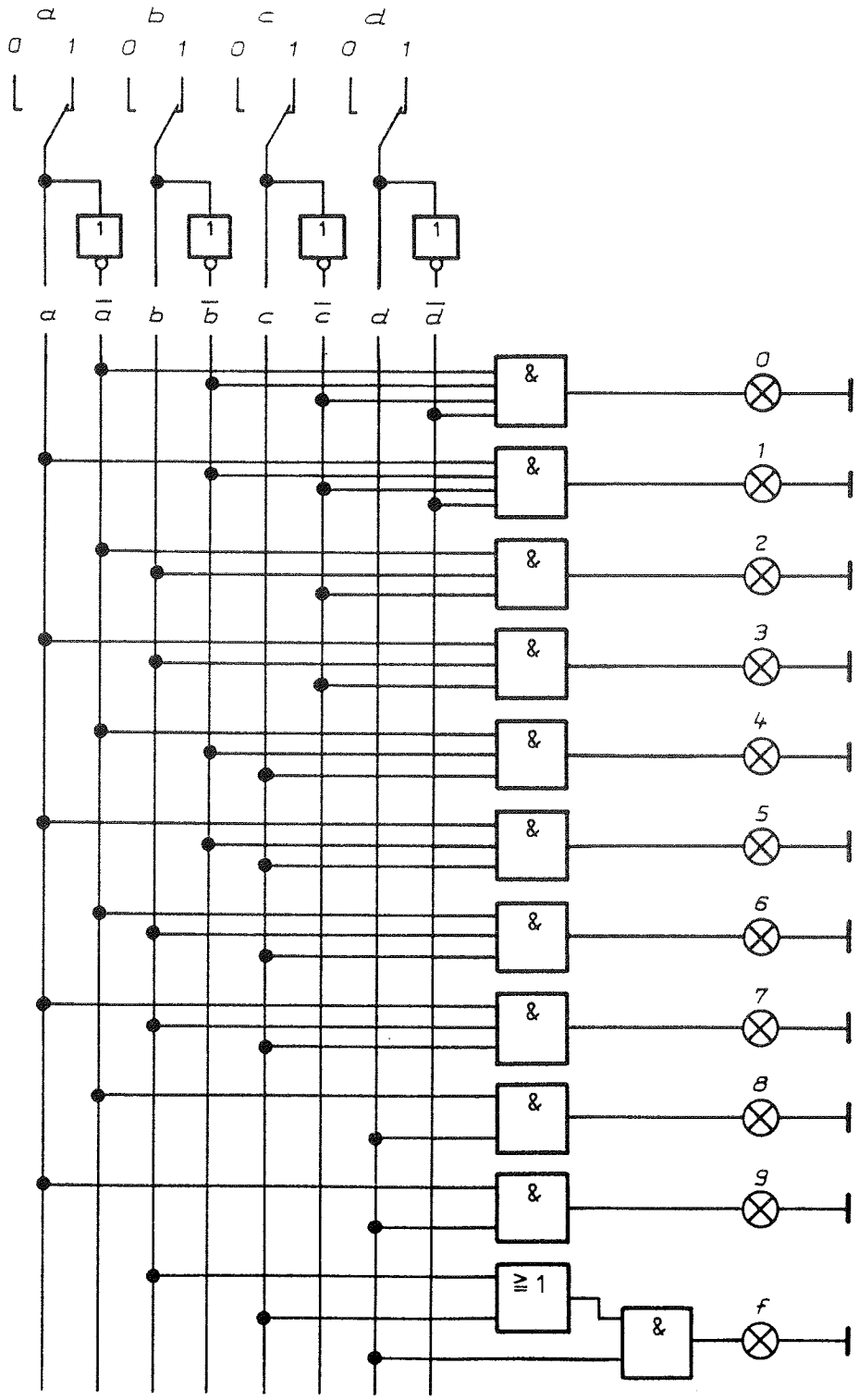
Error detection f by pseudo tetrads

$f = c \cdot d + b \cdot d$

$f = d\,(b + c)$

b	a \ d c	0 0	0 1	1 1	1 0
0	0	0	4	f	8
1	0	2	6	f	f
1	1	3	7	f	f
0	1	1	5	f	9

Circuit:



Exercise 6 - 3:

Develop the BCD/decimal-decoder with pseudo-tetrads detection by means of NOR-gates with the aid of the De Morgans Theorems:

$$a \cdot b = \overline{\bar{a} + \bar{b}} \quad \text{and} \quad \bar{\bar{a}} = a$$

Solution 6 - 3:

Realization of x_0 up to x_9 and f by NOR-gates:

$$x_0 = \overline{\bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}} = \overline{a + b + c + d}$$

$$x_1 = \overline{a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}} = \overline{\bar{a} + b + c + d}$$

$$x_2 = \overline{\bar{a} \cdot b \cdot \bar{c}} = \overline{a + \bar{b} + c}$$

$$x_3 = \overline{a \cdot b \cdot \bar{c}} = \overline{\bar{a} + \bar{b} + c}$$

$$x_4 = \overline{\bar{a} \cdot \bar{b} \cdot c} = \overline{a + b + \bar{c}}$$

$$x_5 = \overline{a \cdot \bar{b} \cdot c} = \overline{\bar{a} + b + \bar{c}}$$

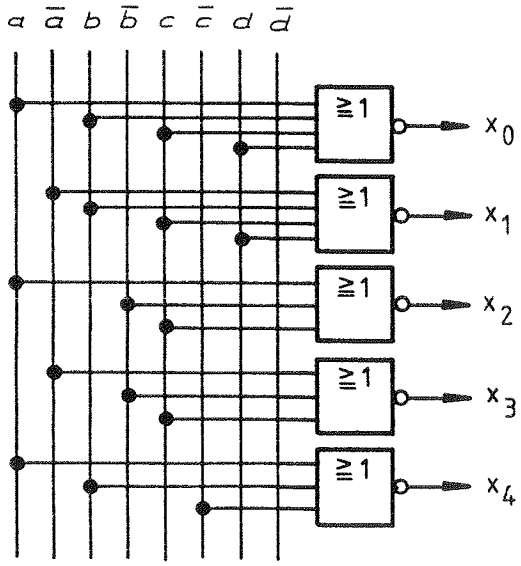
$$x_6 = \overline{\bar{a} \cdot b \cdot c} = \overline{a + \bar{b} + \bar{c}}$$

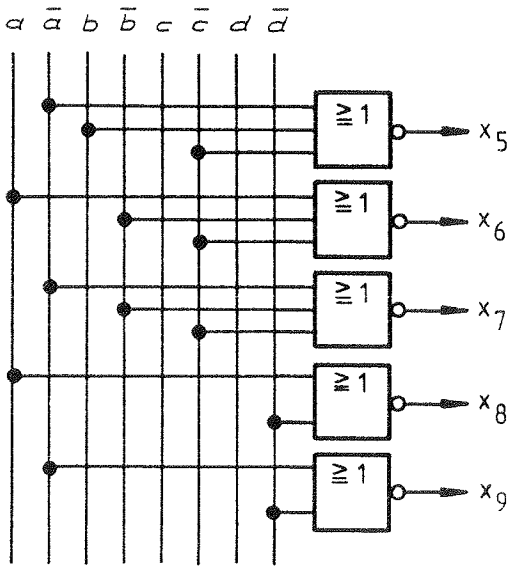
$$x_7 = \overline{a \cdot b \cdot c} = \overline{\bar{a} + \bar{b} + \bar{c}}$$

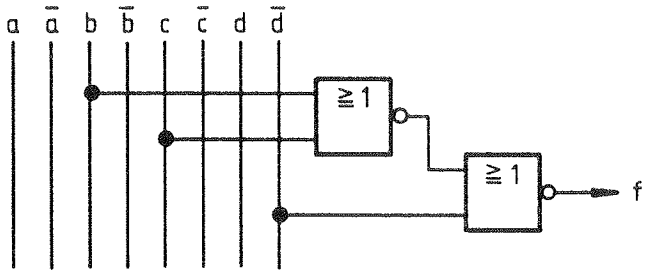
$$x_8 = \overline{\bar{a} \cdot d} = \overline{a + \bar{d}}$$

$$x_9 = \overline{a \cdot d} = \overline{\bar{a} + \bar{d}}$$

$$f = \overline{d\ (b + c)} = \overline{\bar{d} + \overline{(b + c)}}$$







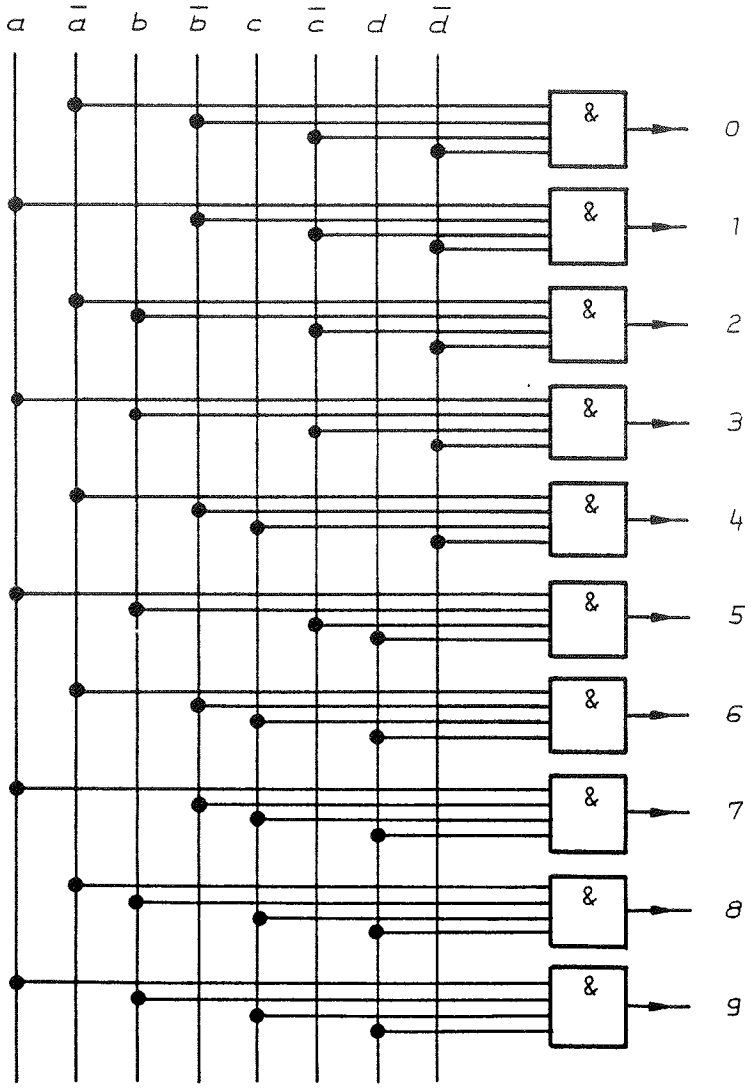
With the following decoders the circuits are shown simplified according to the logical functions.

Exercise 6 - 4:

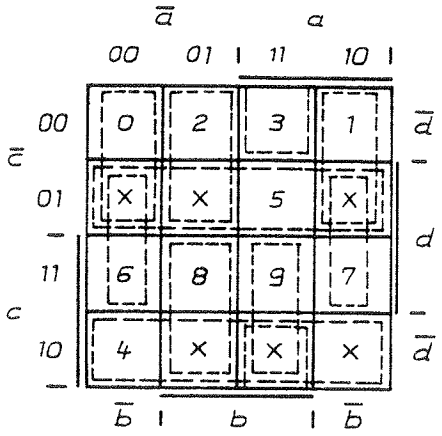
Develop a decoder circuit for the conversion of the Aiken-Code (2421-Code) into the decimal system and minimize the circuit.

d	c	b	a	Dec. System
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
1	0	1	1	5
1	1	0	0	6
1	1	0	1	7
1	1	1	0	8
1	1	1	1	9

Solution 6 - 4: (without simplification)



Karnaugh-Diagram for minimization



$x_0 = \bar{a} \cdot \bar{b} \cdot \bar{c}$

$x_1 = a \cdot \bar{b} \cdot \bar{c}$ or $a\bar{b}\bar{d}$

$x_2 = \bar{a} \cdot b \cdot \bar{c}$ or $\bar{a}b\bar{d}$

$x_3 = a \cdot b \cdot \bar{d}$

$x_4 = c \cdot \bar{d}$

$x_5 = \bar{c} \cdot d$

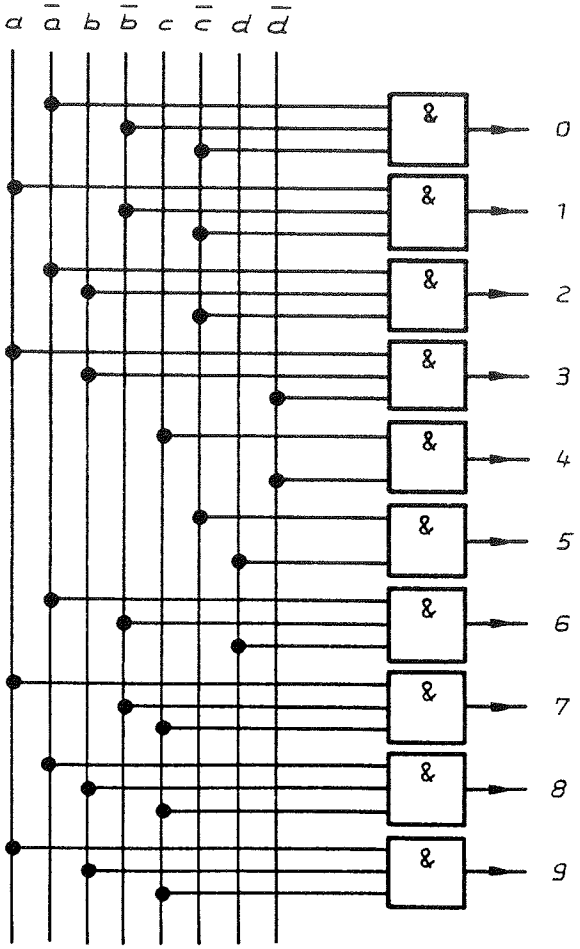
$x_6 = \bar{a} \cdot \bar{b} \cdot d$

$x_7 = a \cdot \bar{b} \cdot d$ or $a \cdot \bar{b} \cdot c$

$x_8 = \bar{a} \cdot b \cdot c$ or $\bar{a} \cdot b \cdot d$

$x_9 = a \cdot b \cdot c$

Minimized circuit:

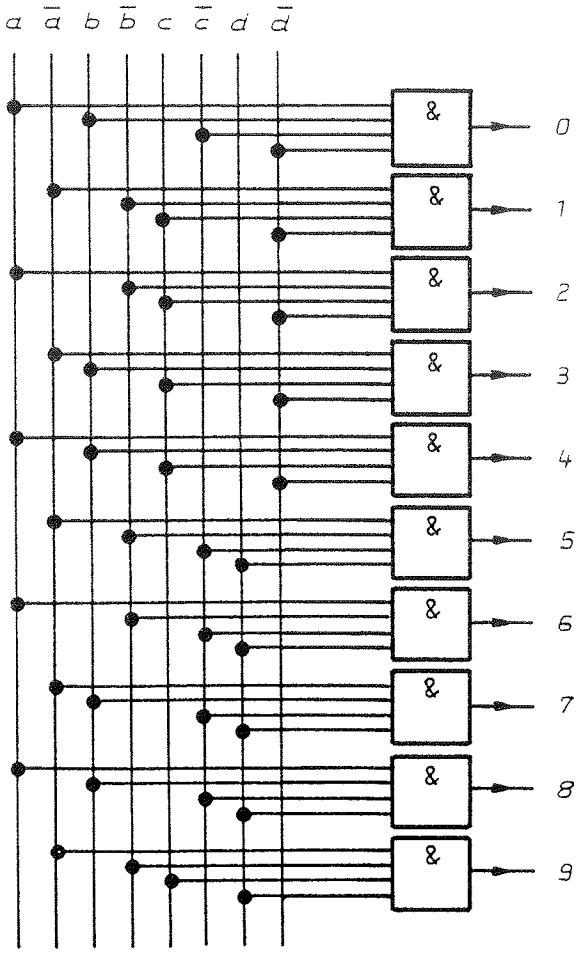


Exercise 6 - 5:

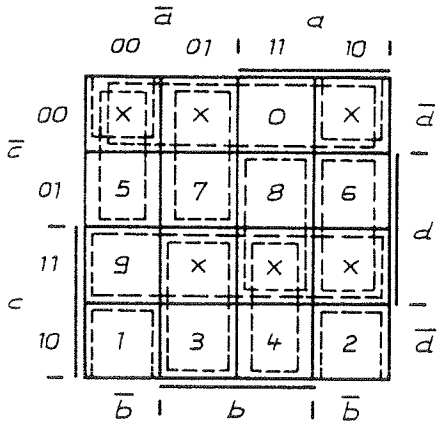
Develop a decoder circuit for conversion of the 3-excess-code into the decimal system and minimize the circuit.

d	c	b	a	Dec. System
0	0	1	1	0
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	4
1	0	0	0	5
1	0	0	1	6
1	0	1	0	7
1	0	1	1	8
1	1	0	0	9

Solution 6 - 5: (without simplification)



Karnaugh-diagram for the simplification of the circuit:



$x_0 = \bar{c} \cdot \bar{d}$

$x_1 = \bar{a} \cdot \bar{b} \cdot \bar{d}$

$x_2 = a \cdot \bar{b} \cdot \bar{d}$ (or $a\bar{b}c$)

$x_3 = \bar{a} \cdot b \cdot c$ (or $\bar{a}b\bar{d}$)

$x_4 = a \cdot b \cdot c$

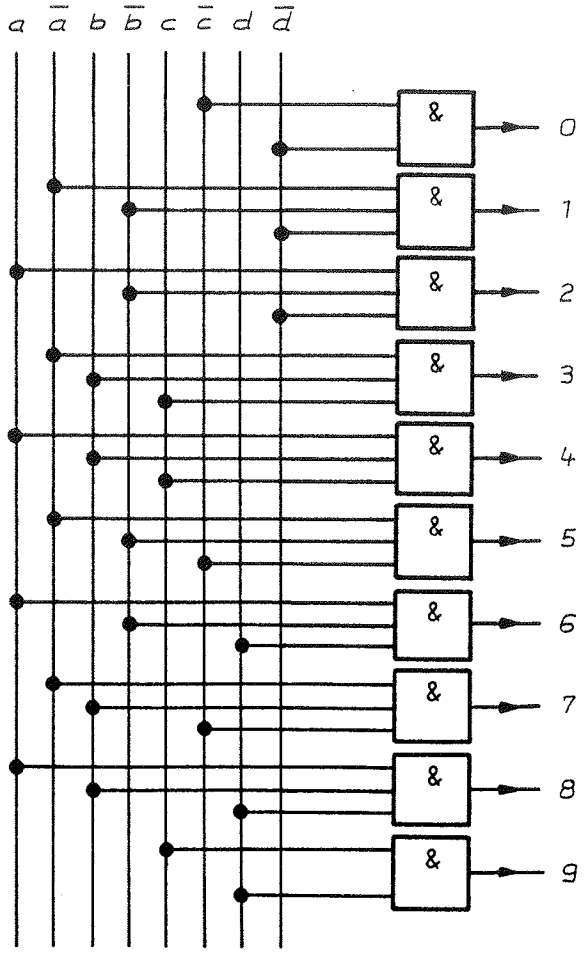
$x_5 = \bar{a} \cdot \bar{b} \cdot \bar{c}$

$x_6 = a \cdot \bar{b} \cdot d$ (or $a\bar{b}\bar{c}$)

$x_7 = \bar{a} \cdot b \cdot \bar{c}$ (or $\bar{a}bd$)

$x_8 = a \cdot b \cdot d$

$x_9 = c \cdot d$

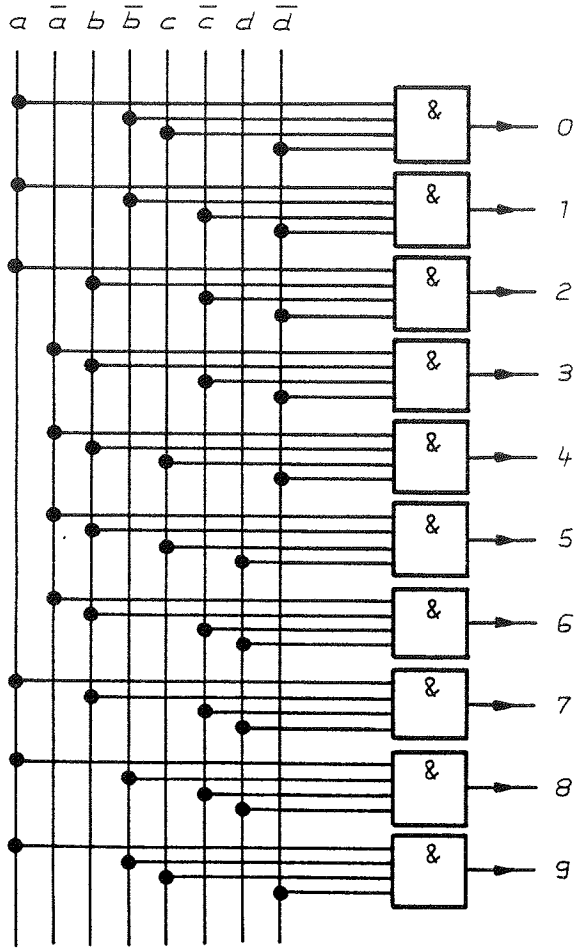


Exercise 6 - 6:

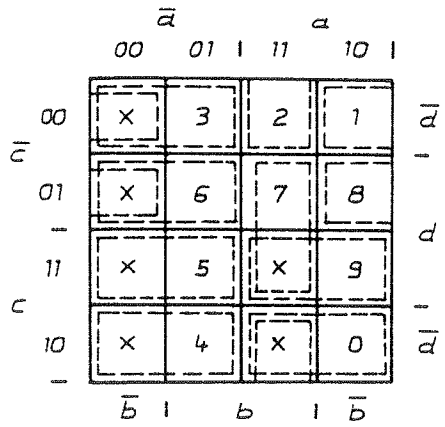
Construct a decoder circuit which converts the Petherick-Code into the decimal system and minimize the circuit.

d	c	b	a	Dec. system
0	1	0	1	0
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
1	1	1	0	5
1	0	1	0	6
1	0	1	1	7
1	0	0	1	8
1	1	0	1	9

Solution 6 - 6: (not simplified)

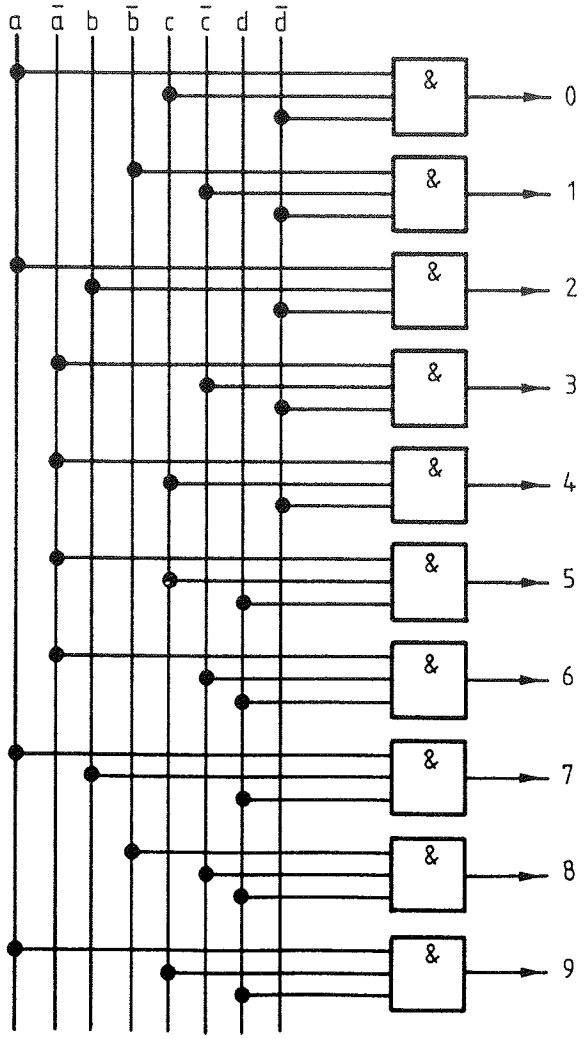


Simplification:



$x_0 = a \cdot c \cdot \bar{d}$
$x_1 = \bar{b} \cdot \bar{c} \cdot \bar{d}$
$x_2 = a \cdot b \cdot \bar{d}$
$x_3 = \bar{a} \cdot \bar{c} \cdot \bar{d}$
$x_4 = \bar{a} \cdot c \cdot \bar{d}$
$x_5 = \bar{a} \cdot c \cdot d$
$x_6 = \bar{a} \cdot \bar{c} \cdot d$
$x_7 = a \cdot b \cdot d$
$x_8 = \bar{b} \cdot \bar{c} \cdot d$
$x_9 = a \cdot c \cdot d$

Minimized circuit:

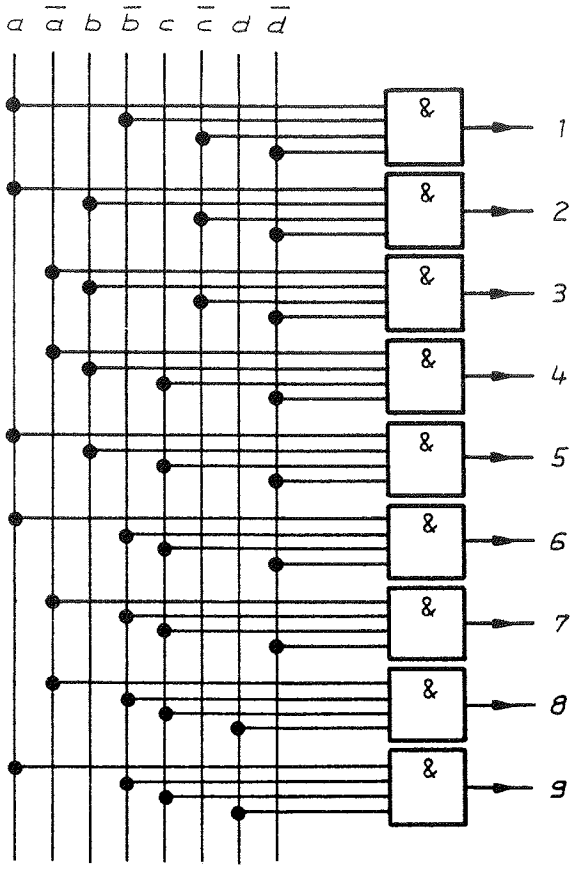


Exercise 6 - 7:

Develop a decoder circuit for the conversion of the Gray-Code into the decimal system.

d	c	b	a	Dec. system
0	0	0	0	0
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
0	1	1	1	5
0	1	0	1	6
0	1	0	0	7
1	1	0	0	8
1	1	0	1	9
1	1	1	1	10

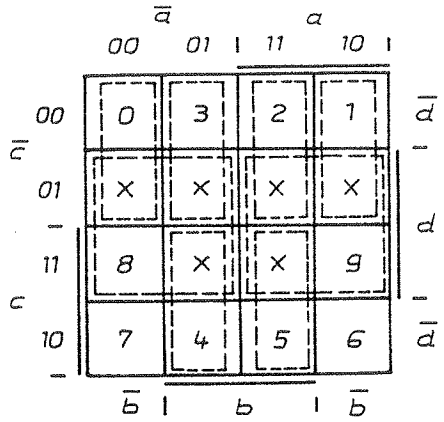
Solution 6 - 7: (without simplification)



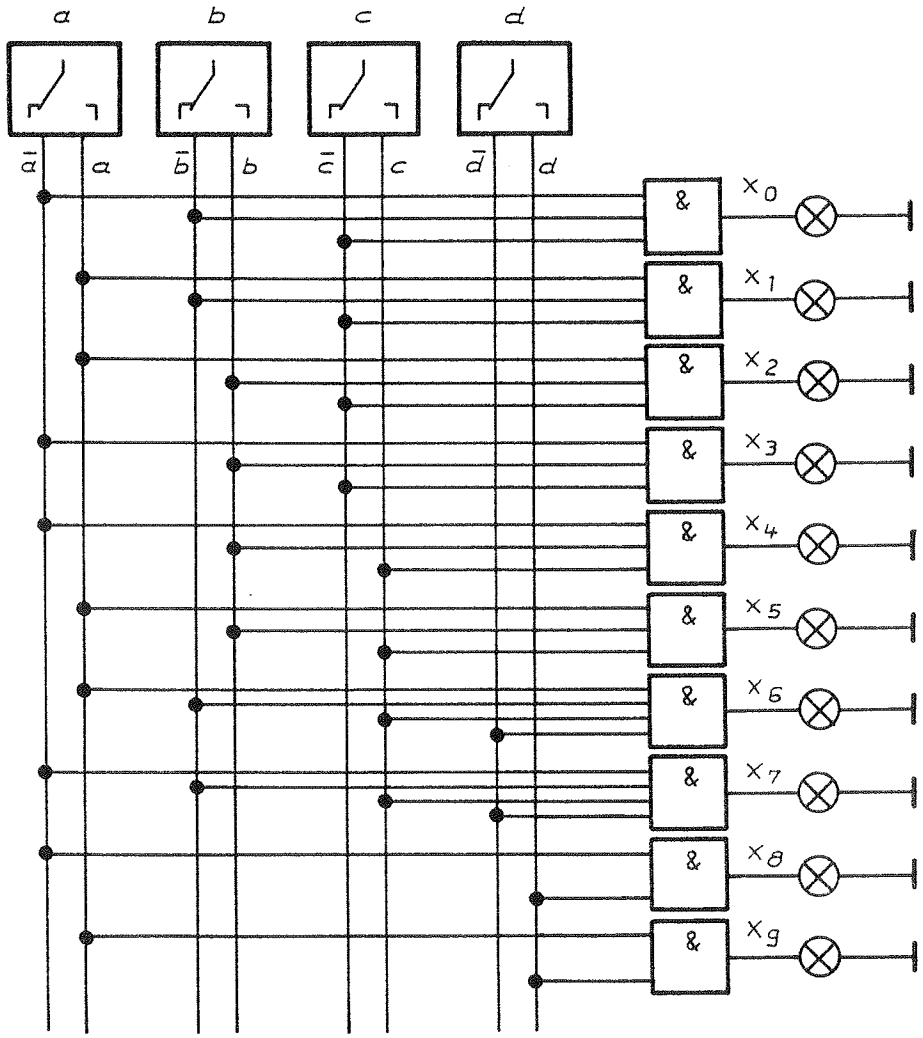
Simplification:

$$
\begin{aligned}
x_0 &= \bar{a} \cdot \bar{b} \cdot \bar{c} \\
x_1 &= a \cdot \bar{b} \cdot \bar{c} \\
x_2 &= a \cdot b \cdot \bar{c} \\
x_3 &= \bar{a} \cdot b \cdot \bar{c} \\
x_4 &= \bar{a} \cdot b \cdot c \\
x_5 &= a \cdot b \cdot c \\
x_6 &= a \cdot \bar{b} \cdot c \cdot \bar{d} \\
x_7 &= \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d} \\
x_8 &= \bar{a} \cdot d \\
x_9 &= a \cdot d
\end{aligned}
$$

Karnaugh-Diagram:



Simplified circuit:

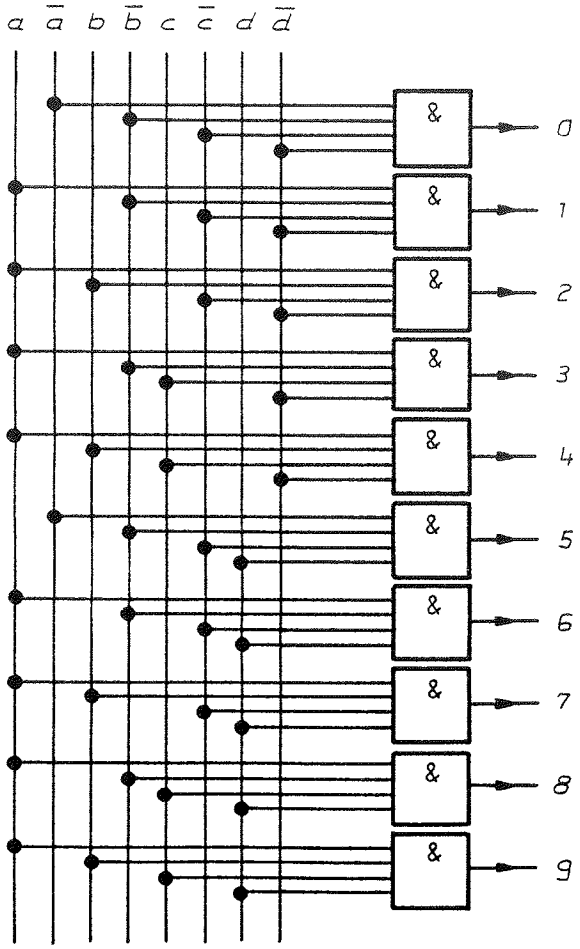


Exercise 6 - 8:

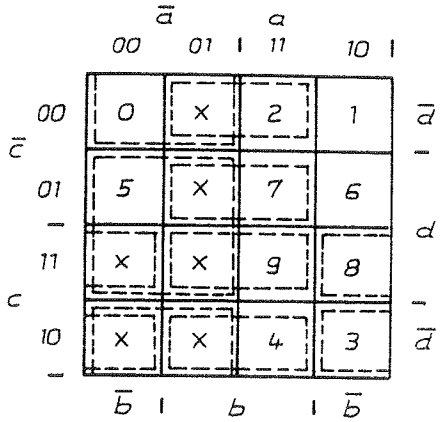
Develop a decoder circuit for the conversion of the White-Code into the decimal system and simplify the circuit.

d	c	b	a	Dec. system
0	0	0	0	0
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	0	5
1	0	0	1	6
1	0	1	1	7
1	1	0	1	8
1	1	1	1	9

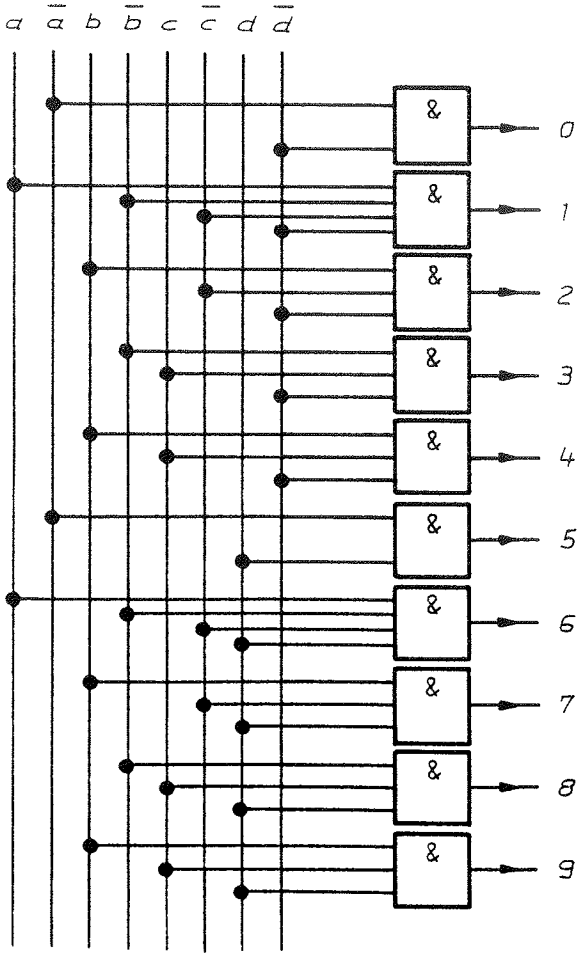
Solution 6 - 8: (not simplified):



Simplification with the aid of the Karnaugh-diagram and "don't care fields":



$x_0 = \bar{a} \cdot \bar{d}$
$x_1 = a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$
$x_2 = b \cdot \bar{c} \cdot d$
$x_3 = \bar{b} \cdot c \cdot \bar{d}$
$x_4 = b \cdot c \cdot \bar{d}$
$x_5 = \bar{a} \cdot d$
$x_6 = a \cdot \bar{b} \cdot \bar{c} \cdot d$
$x_7 = b \cdot \bar{c} \cdot d$
$x_8 = \bar{b} \cdot c \cdot d$
$x_9 = b \cdot c \cdot d$

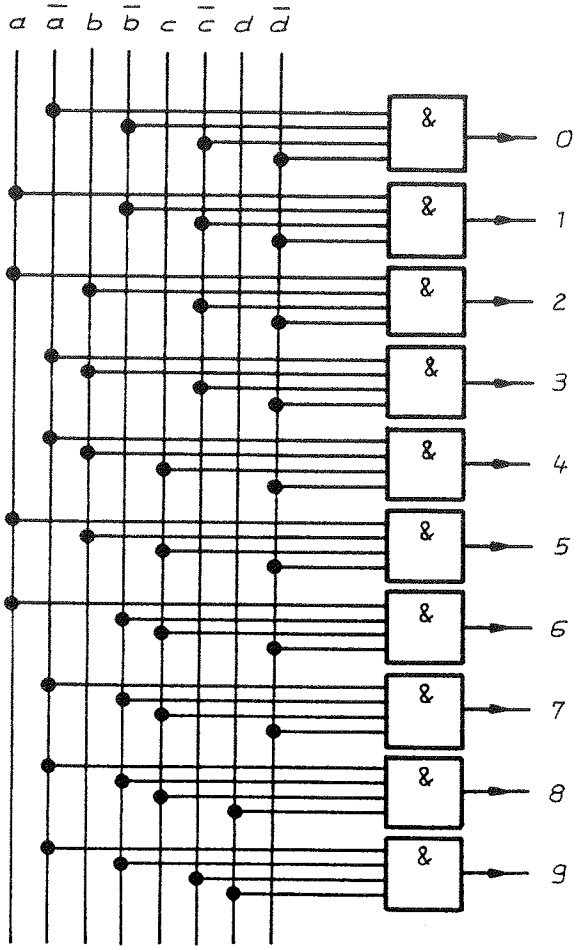


Exercise 6 - 9:

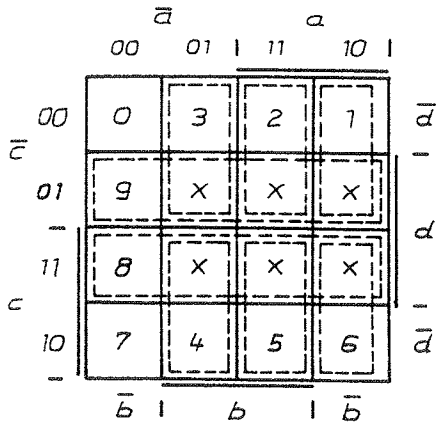
Develop a decoder circuit for the conversion of the Glixon-Code into the decimal system and minimize the circuit.

d	c	b	a	Dec. system
0	0	0	0	0
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
0	1	1	1	5
0	1	0	1	6
0	1	0	0	7
1	1	0	0	8
1	0	0	0	9

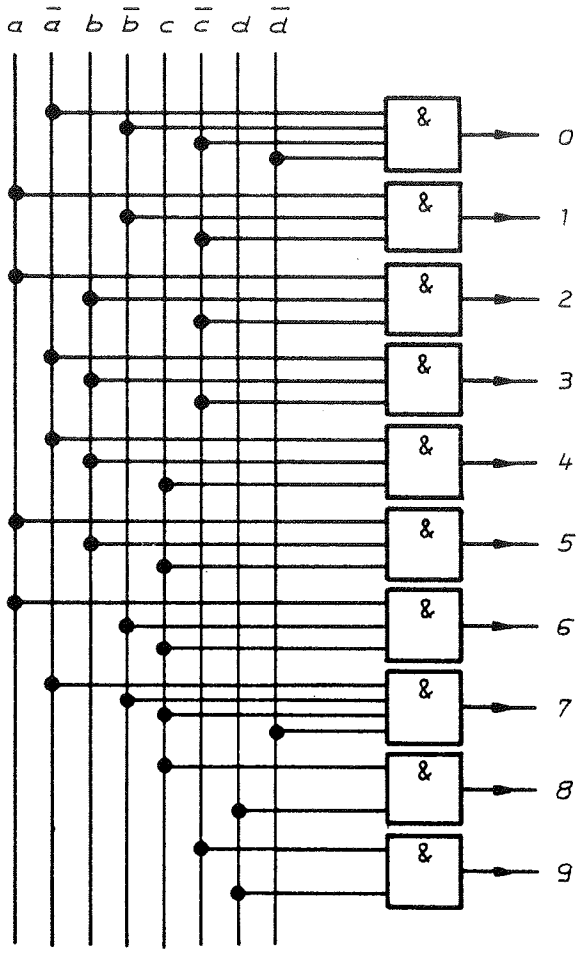
Solution 6 - 9: (not minimized)



Minimized solution:



$x_0 = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$
$x_1 = a \cdot \bar{b} \cdot \bar{c}$
$x_2 = a \cdot b \cdot \bar{c}$
$x_3 = \bar{a} \cdot b \cdot \bar{c}$
$x_4 = \bar{a} \cdot b \cdot c$
$x_5 = a \cdot b \cdot c$
$x_6 = a \cdot \bar{b} \cdot c$
$x_7 = \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d}$
$x_8 = c \cdot d$
$x_9 = \bar{c} \cdot d$



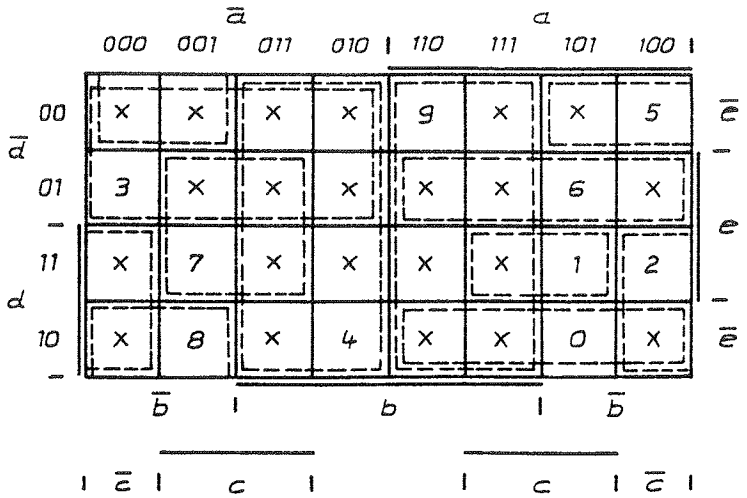
Exercise 6 - 10:

Construct a decoder circuit which converts the Telegraph-Code into the decimal system (digits only).

e	d	c	b	a	Dec. system
0	1	1	0	1	0
1	1	1	0	1	1
1	1	0	0	1	2
1	0	0	0	0	3
0	1	0	1	0	4
0	0	0	0	1	5
1	0	1	0	1	6
1	1	1	0	0	7
0	1	1	0	0	8
0	0	0	1	1	9

Solution 6 - 10:

Minimization with the aid of the Karnaugh-Diagram:



$x_0 = a \cdot d \cdot \bar{e}$

$x_1 = a \cdot c \cdot d \cdot e$

$x_2 = \bar{b} \cdot \bar{c} \cdot d$

$x_3 = \bar{a} \cdot \bar{d}$

$x_4 = \bar{a} \cdot b$

$x_5 = \bar{b} \cdot \bar{d} \cdot \bar{e}$

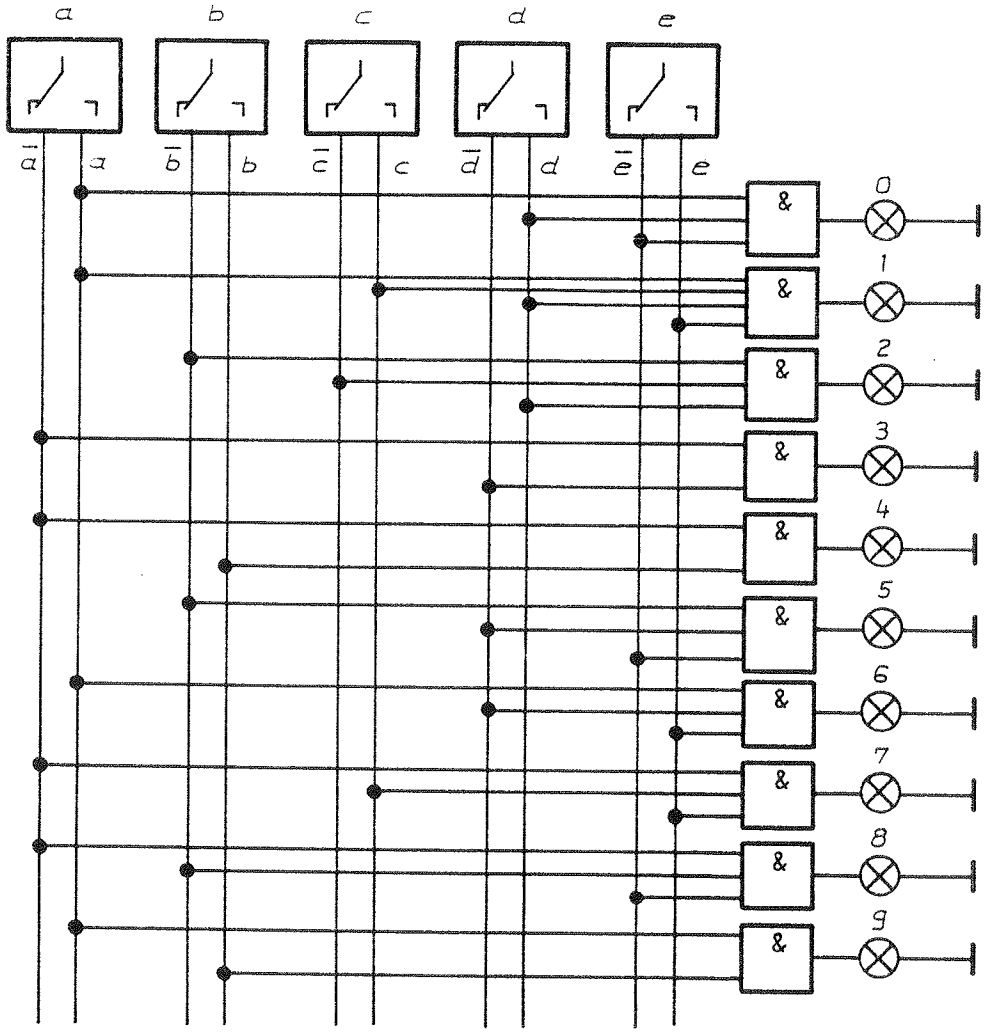
$x_6 = a \cdot \bar{d} \cdot e$

$x_7 = \bar{a} \cdot c \cdot e$

$x_8 = \bar{a} \cdot \bar{b} \cdot \bar{e}$

$x_9 = a \cdot b$

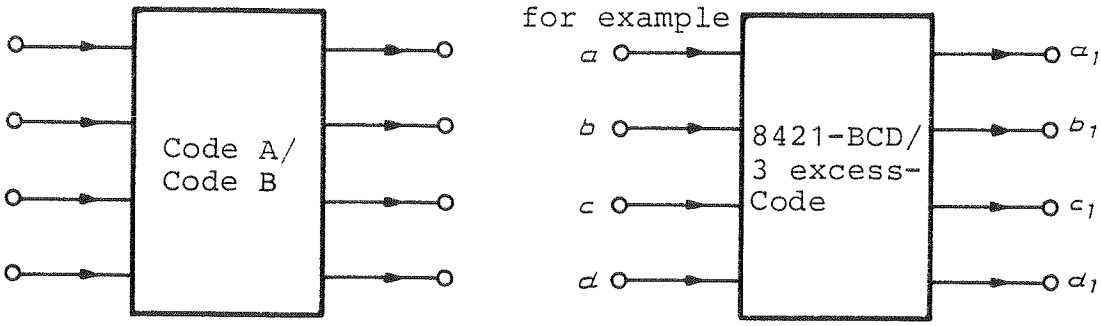
Circuit



7. CODE CONVERTER

7.1 General

Code converters are logical networks which convert one code into another.



Example:

The theoretical handling of such a code converter (8-4-2-1-BCD → 3 excess-Code) is shown below.

Table:

	8-4-2-1-BCD-Code				3 excess-Code			
Dec.	d	c	b	a	d_1	c_1	b_1	a_1
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

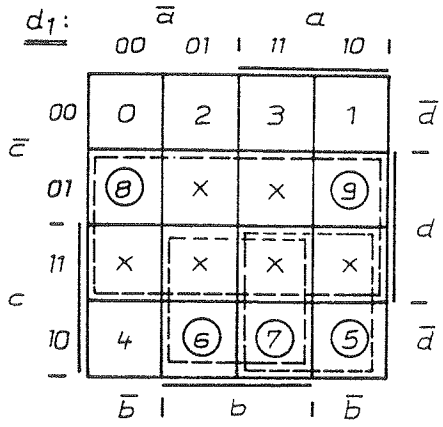
The six combinations from 10 up to 15 are redundant i.e. they may be used as "don't care fields".

$$d_1 = 5 + 6 + 7 + 8 + 9$$
$$c_1 = 1 + 2 + 3 + 4 + 9$$
$$b_1 = 0 + 3 + 4 + 7 + 8$$
$$a_1 = 0 + 2 + 4 + 6 + 8$$

(abbreviated notation, the exact notation would be to write down the corresponding combinations of the 8-4-2-1-BCD-Code).

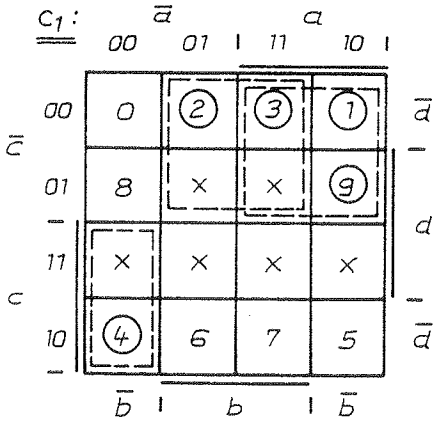
With the aid of the Karnaugh-diagram these equations have to be simplified.

d_1:



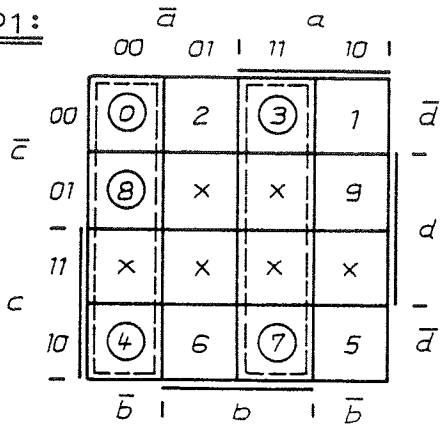
$$d_1 = ac + \bar{b}c + d$$

c_1:



$$c_1 = a\bar{c} + b\bar{c} + \bar{a}\bar{b}c$$

b_1:



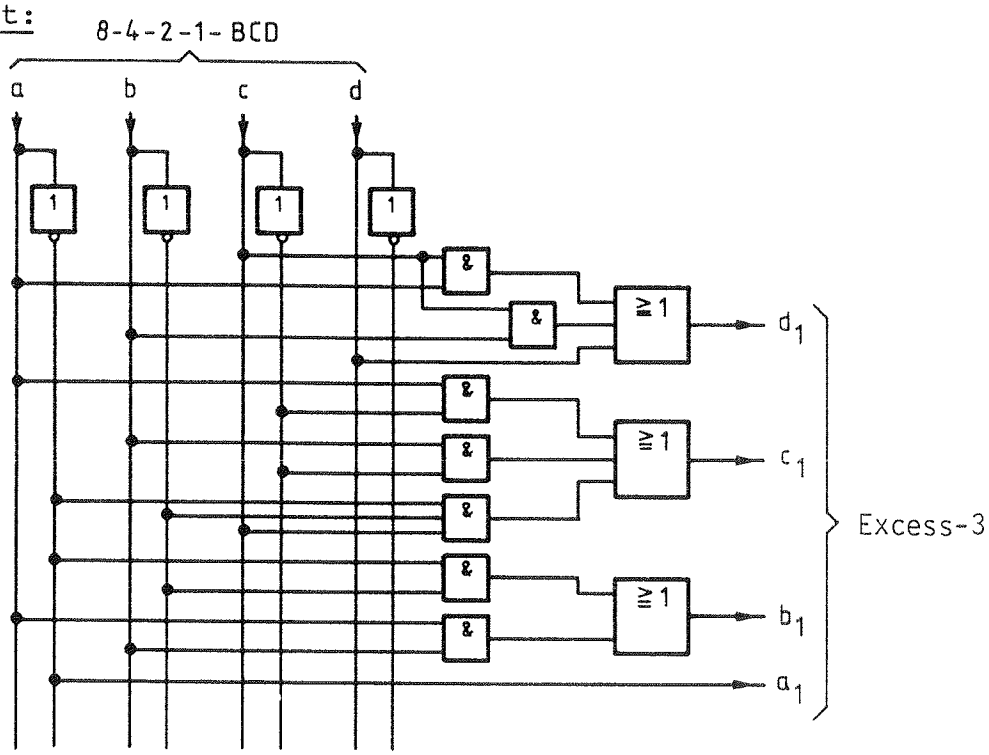
$$b_1 = \bar{a}\bar{b} + ab$$

a_1:

From the table it is obvious that

$$a_1 = \bar{a}$$

Circuit:



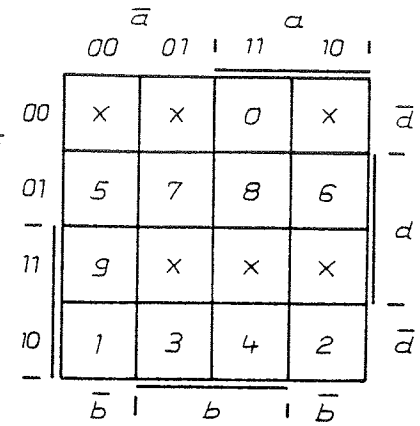
7.2 Exercises

Exercise 7 - 1:

Develop a code-converter for the conversion of the excess-three-code into the 8421-code.

Solution 7 - 1:

	Excess-3-Code				8421-Code			
Dec.	d	c	b	a	d_1	c_1	b_1	a_1
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	0
3	0	1	1	0	0	0	1	1
4	0	1	1	1	0	1	0	0
5	1	0	0	0	0	1	0	1
6	1	0	0	1	0	1	1	0
7	1	0	1	0	0	1	1	1
8	1	0	1	1	1	0	0	0
9	1	1	0	0	1	0	0	1
	1	1	0	1	1	0	1	0
	1	1	1	0	1	0	1	1
	1	1	1	1	1	1	0	0
	0	0	0	0	1	1	0	1
	0	0	0	1	1	1	1	0
	0	0	1	0	1	1	1	1



The lower six rows are redundant.

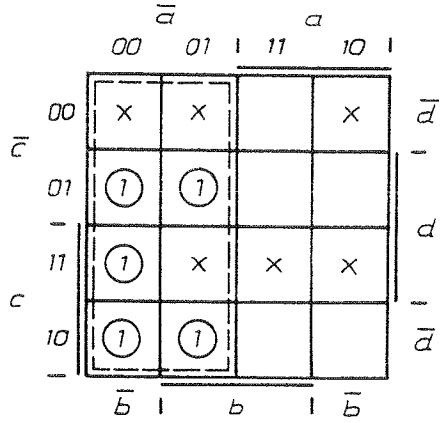
$d_1 = 8 + 9$

$b_1 = 2 + 3 + 6 + 7$

$c_1 = 4 + 5 + 6 + 7$

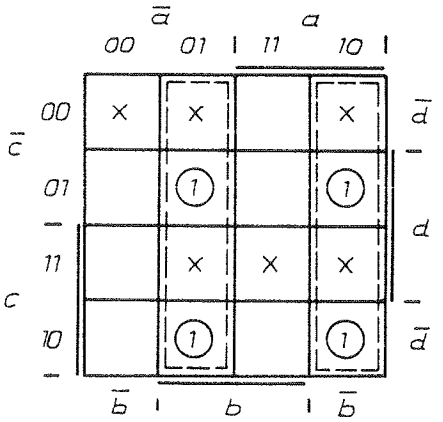
$a_1 = 1 + 3 + 5 + 7 + 9$

a_1:



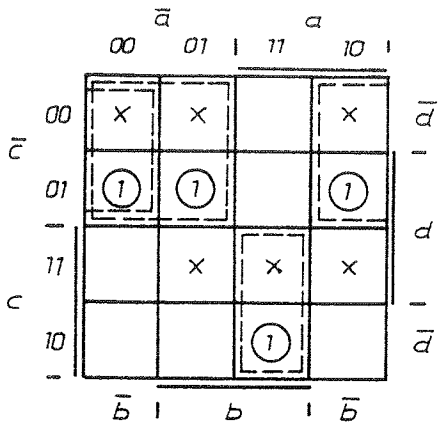
$a_1 = \bar{a}$

b_1:



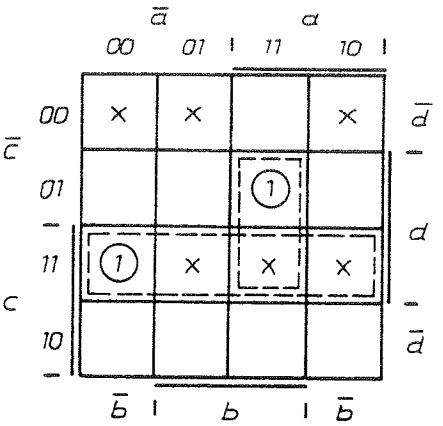
$b_1 = a\bar{b} + \bar{a}b$ (antivalence)

c_1:

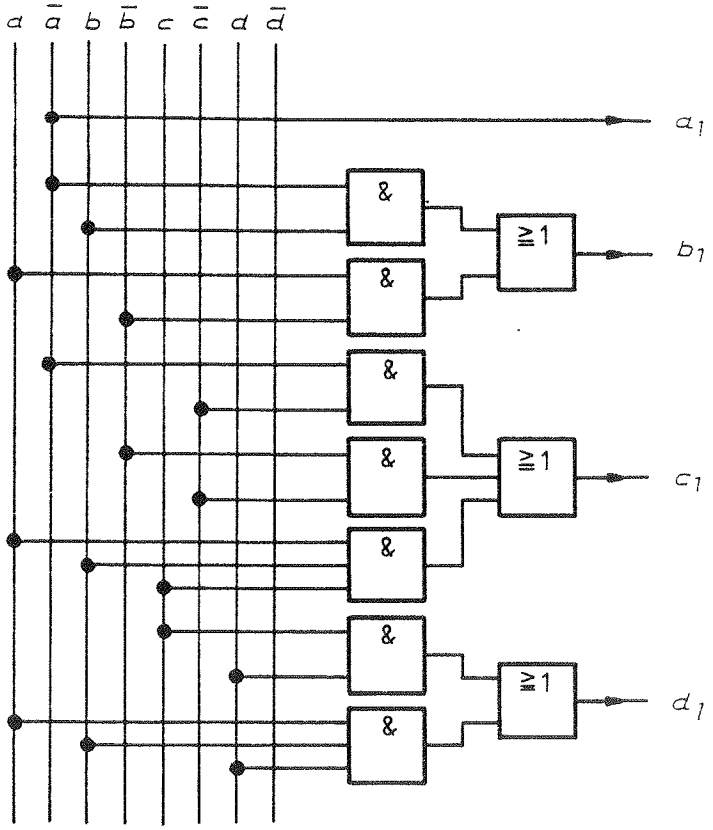


$c_1 = \bar{a}\bar{c} + abc + \bar{b}\bar{c}$

d_1:



$d_1 = cd + abd$

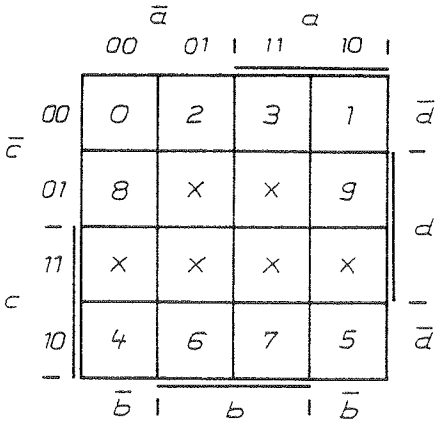

a
ā
b
b̄
c
c̄
d
d̄
&
≧1
a1
b1
c1
d1

Exercise 7 - 2:

Develop a code-converter circuit which converts the 8421-code into the Gray-Code.

Solution 7 - 2:

	8421-code				Gray-Code			
Dec.	d	c	b	a	d_1	c_1	b_1	a_1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
	1	0	1	0	1	1	1	1
	1	0	1	1	1	1	1	0
	1	1	0	0	1	0	1	0
	1	1	0	1	1	0	1	1
	1	1	1	0	1	0	0	1
	1	1	1	1	1	0	0	0



The last six rows are redundant.

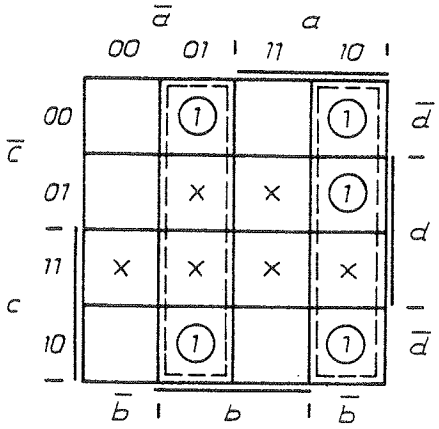
$d_1 = 8 + 9$

$b_1 = 2 + 3 + 4 + 5$

$c_1 = 4 + 5 + 6 + 7 + 8 + 9$

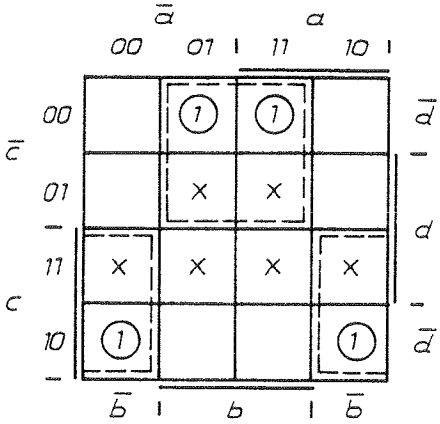
$a_1 = 1 + 2 + 5 + 6 + 9$

a_1:



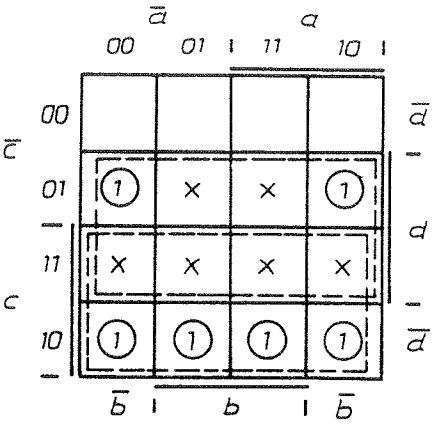
$a_1 = \bar{a}b + a\bar{b}$

b_1:



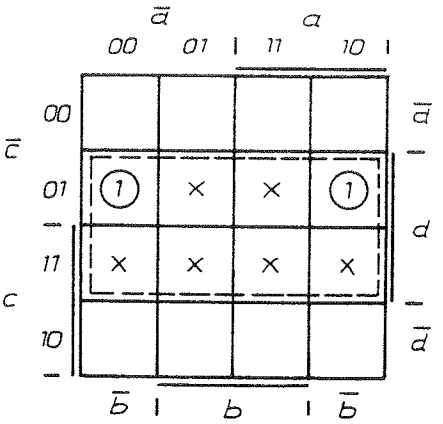
$b_1 = b\bar{c} + \bar{b}c$

c_1:

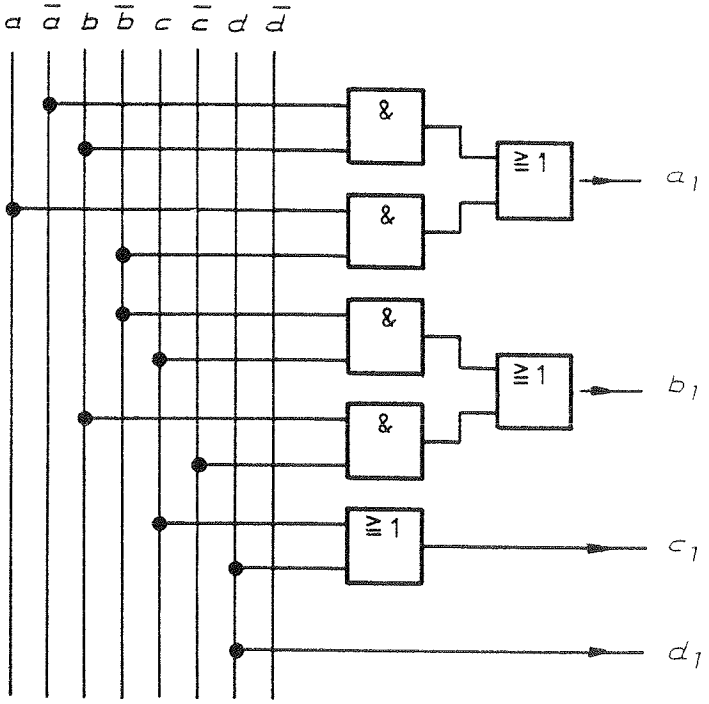


$c_1 = c + d$

d_1:



$d_1 = d$

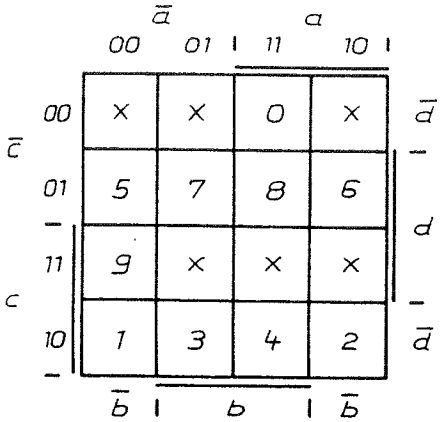


Exercise 7 - 3:

Develop a code-converter circuit which converts the excess-three-code into the Petherick-code.

Solution 7 - 3:

	Excess-3-code				Petherick-code			
Dec.	d	c	b	a	d_1	c_1	b_1	a_1
0	0	0	1	1	0	1	0	1
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	1
3	0	1	1	0	0	0	1	0
4	0	1	1	1	0	1	1	0
5	1	0	0	0	1	1	1	0
6	1	0	0	1	1	0	1	0
7	1	0	1	0	1	0	1	1
8	1	0	1	1	1	0	0	1
9	1	1	0	0	1	1	0	1
	1	1	0	1				
	1	1	1	0				
	1	1	1	1				
	0	0	0	0				
	0	0	0	1				
	0	0	1	0				



The last six rows are redundant.

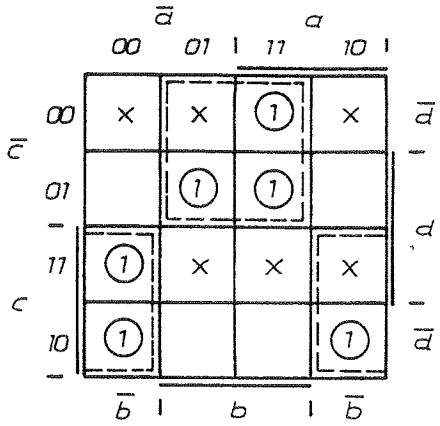
$d_1 = 5 + 6 + 7 + 8 + 9$

$b_1 = 2 + 3 + 4 + 5 + 6 + 7$

$c_1 = 0 + 4 + 5 + 9$

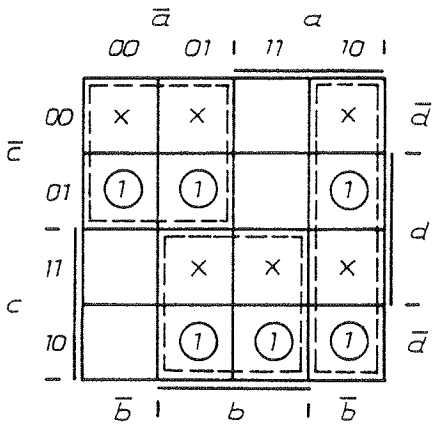
$a_1 = 0 + 1 + 2 + 7 + 8 + 9$

a_1:



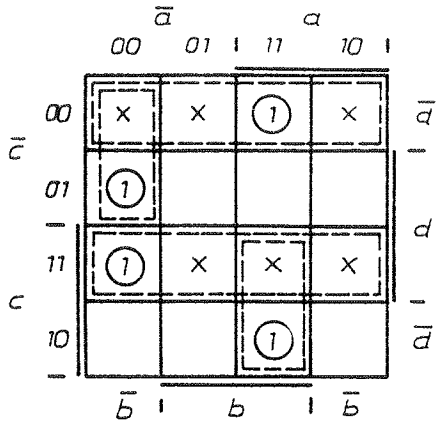
$a_1 = b\bar{c} + \bar{b}c$

b_1:



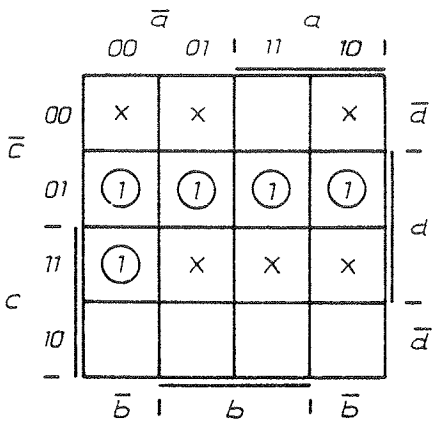
$b_1 = \bar{a}\bar{c} + a\bar{b} + bc$

c_1:

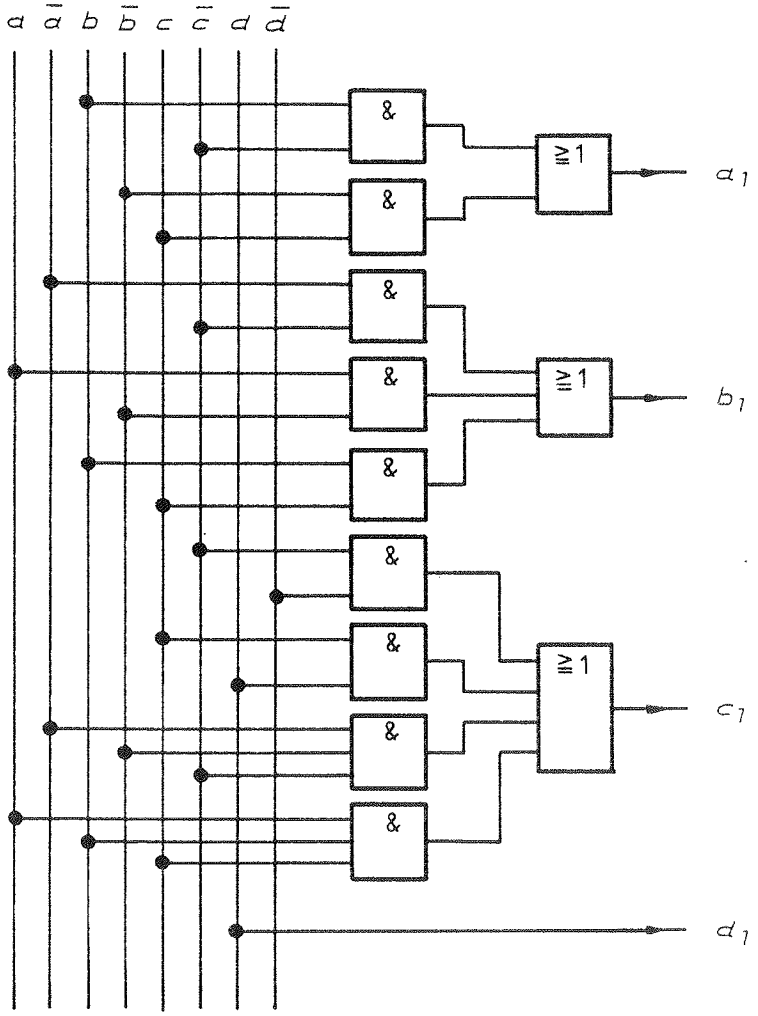


$c_1 = \bar{c}\bar{d} + cd + \bar{a}\bar{b}\bar{c} + abc$

d_1:



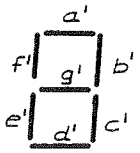
$d_1 = d$

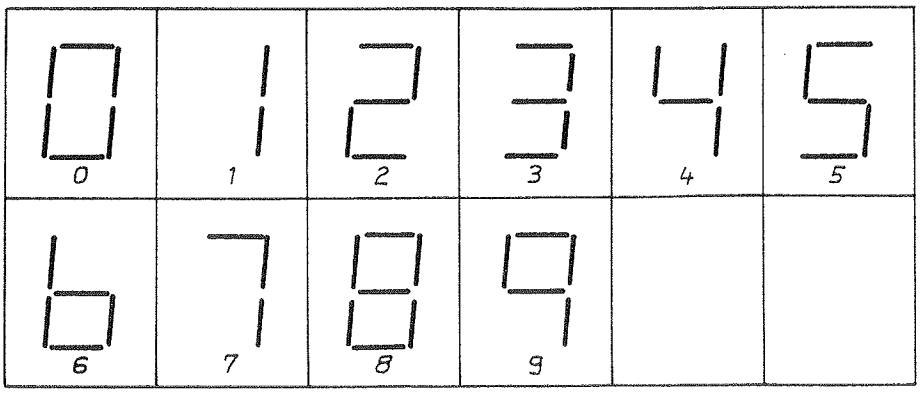

a ā b b̄ c c̄ d d̄
&
≧1
a1
b1
c1
d1

Exercise 7 - 4:

Develop a decoder for the 7-segment-display of the 8421-BCD-Code.

Display of digits by means of 7 segments:





Solution 7 - 4:

Dec.	a 2^0	b 2^1	c 2^2	d 2^3	a'	b'	c'	d'	e'	f'	g'
0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	0	0	1	1	1	1	0	0	1
4	0	0	1	0	0	1	1	0	0	1	1
5	1	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

Functions:

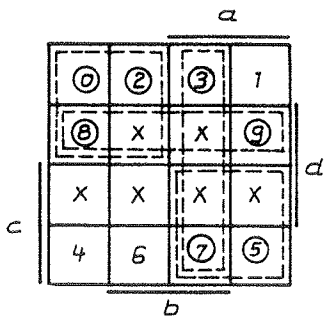
a' = 0+2+3+5+7+8+9
b' = 0+1+2+3+4+7+8+9
c' = 0+1+3+4+5+6+7+8+9
d' = 0+2+3+5+6+8
e' = 0+2+6+8
f' = 0+4+5+6+8+9
g' = 2+3+4+5+6+8+9

(simplified notation, to be exact, one has to write down the combinations according to the 8421-code, which are assigned to the corresponding decimal numbers.)

Karnaugh-diagram

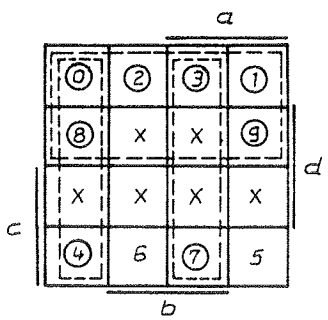
c	d \ a b	0 0	0 1	1 1	1 0
0	0	0	2	3	1
0	1	8	X	X	9
1	1	X	X	X	X
1	0	4	6	7	5

Segment a':



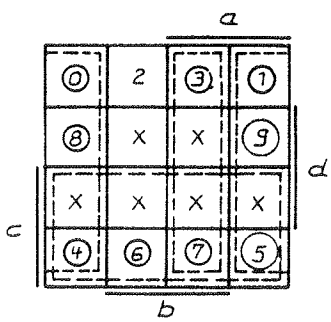
$a' = \bar{a}\bar{c} + ab + \bar{c}d + ac$

Segment b':



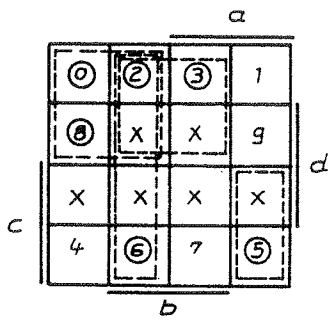
$b' = \bar{c} + \bar{a}\bar{b} + ab$

Segment c':



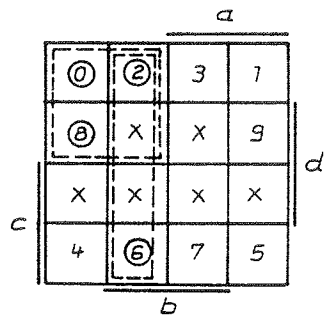
$c' = c + a + \bar{b}$

Segment d':



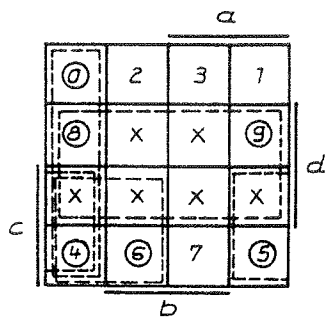
$d' = \bar{a}\bar{c} + \bar{a}b + b\bar{c} + a\bar{b}c$

Segment e':



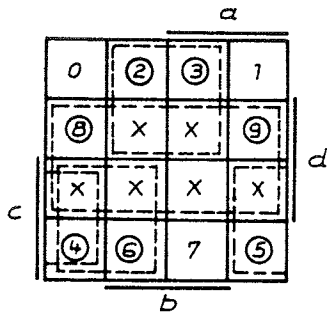
$e' = \bar{a}\bar{c} + \bar{a}b$

Segment f':

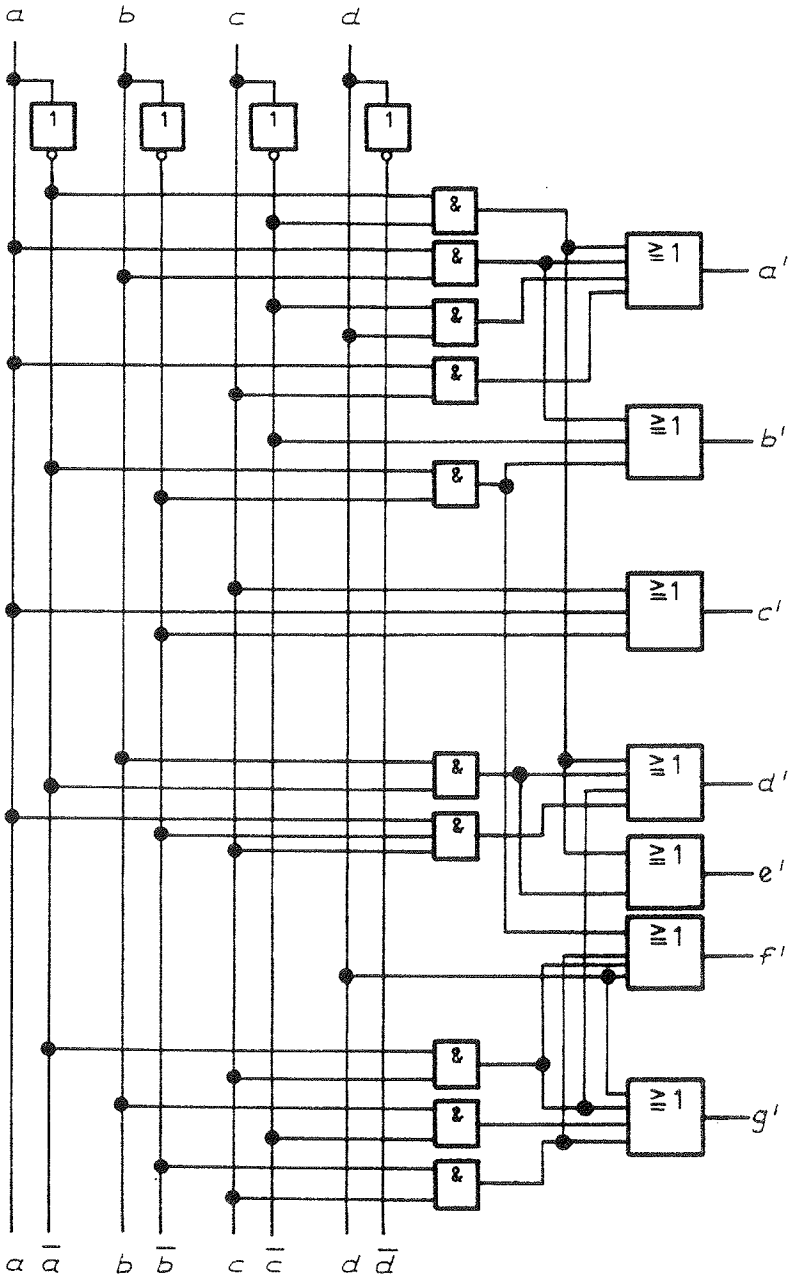


$f' = d + \bar{a}\bar{b} + \bar{b}c + \bar{a}c$

Segment g':



$g' = d + \bar{a}c + b\bar{c} + c\bar{b}$


a
b
c
d
1
&
≧1
a'
b'
c'
d'
e'
f'
g'
a ā b b̄ c c̄ d d̄

8. ARITHMETIC CIRCUITS

8.1 Exercises

Exercise 8 - 1:

Develop a circuit for a half-adder, that can add two one-placed binary numbers.

Solution 8 - 1:

A half-adder adds two binary bits a + b, e.g. the half-adder is used to add the lowest place of two n-placed binary numbers where there is no carry from a preceding place.

a = 1. term of the sum
b = 2. term of the sum
s = sum of a + b
c_s = carry of the sum

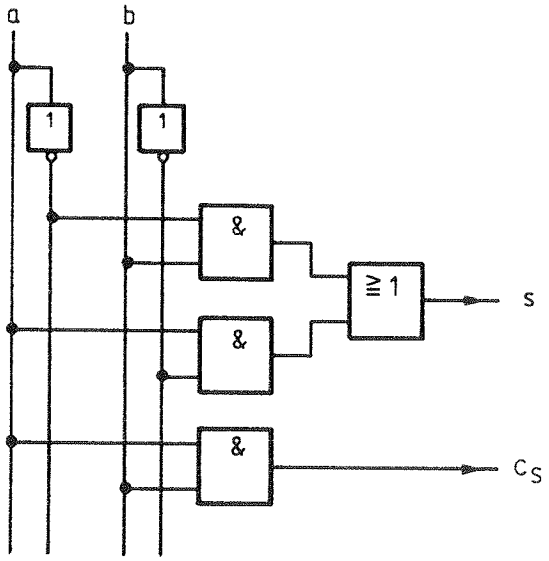
The possible cases are as follows:

0 + 0 = 0, no carry
0 + 1 = 1, no carry
1 + 0 = 1, no carry
1 + 1 = 0, carry 1

The table display:

a	b	s	c_s
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

i.e.: $c_s = ab$
and: $s = \bar{a}b + a\bar{b}$
(antivalence or exclusive-OR)



Exercise 8 - 2:

Develop a half-subtracter with the inputs a + b.

Solution 8 - 2:

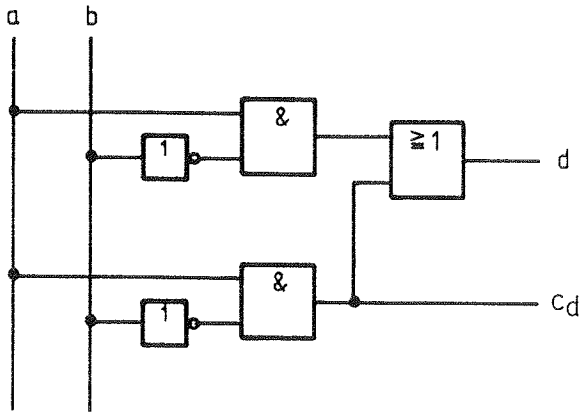
A half-subtracter should form the difference d and the carry c_d (resulting from a borrow) from two binary numbers a and b, the binary numbers zero and one are represented by the binary values 0 and 1 respectively.
The 4 possibilities there are:

0 - 0 = 0, no carry
1 - 0 = 1, no carry
0 - 1 = 1, carry
1 - 1 = 0, no carry

Display in the function table:

a	b	d	c_d
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$c_d = \bar{a}b$
$d = a\bar{b} + \bar{a}b$
(antivalence)



Exercise 8 - 3:

Design a full-adder for a binary place (1 bit).

Solution 8 - 3:

A half-adder does not master a full adding operation, because the carry of the preceding place has not been considered. By the full-adder not only the total of both binary numbers a_n and b_n is made up but also is this carry c_{n-1} to be considered for the summation.

Function table with all possible cases:

a_n	b_n	c_{n-1}	s_n	c_n	comb.
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	2
1	1	0	0	1	3
0	0	1	1	0	4
0	1	1	0	1	5
1	0	1	0	1	6
1	1	1	1	1	7

a_n = 1. term of the sum of the n^{th} binary place

b_n = 2. term of the sum of the n^{th} binary place

c_{n-1} = carry of the preceeding $(n-1)^{th}$ place

s_n = sum of the n^{th} place

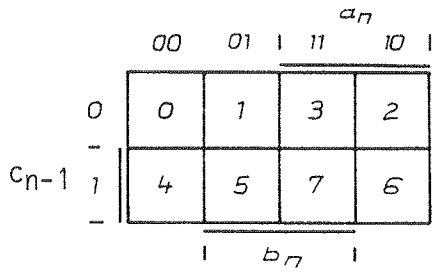
c_n = carry of the n^{th} place

With the aid of the disjunctive standard form the following is noted:

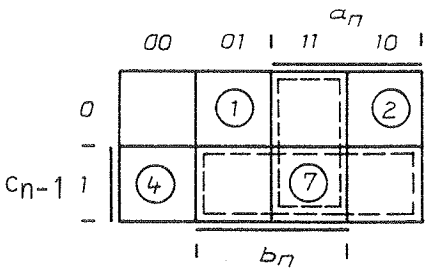
$$s_n = 1 + 2 + 4 + 7$$
$$c_n = 3 + 5 + 6 + 7$$

(the numbers are a substitution for the corresponding combinations).

Simplification in the Karnaugh-Diagram



Sum:

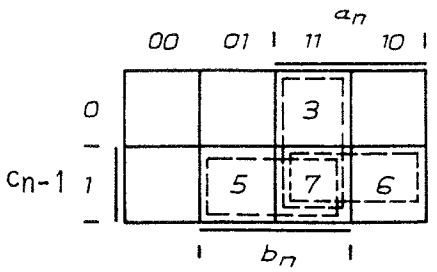


Obviously for the sum no simplification is possible

Also:

$$s_n = \overline{a_n} \cdot b_n \cdot \overline{c_{n-1}} + a_n \cdot \overline{b_n} \cdot \overline{c_{n-1}} + \overline{a_n} \cdot \overline{b_n} \cdot c_{n-1} + a_n \cdot b_n \cdot c_{n-1}$$

Carry:

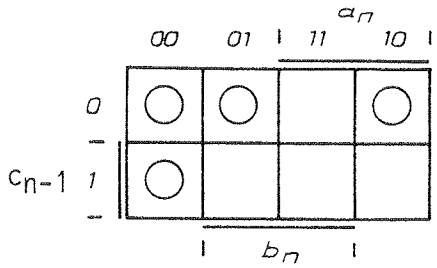


$$c_n = a_n \cdot b_n + b_n \cdot c_{n-1} + a_n \cdot c_{n-1}$$

By means of clever factoring out the sum term can be simplified:

$$s_n = (a_n + b_n + c_{n-1}) \cdot \underbrace{(\overline{b_n} \cdot \overline{c_{n-1}} + \overline{a_n} \cdot \overline{c_{n-1}} + \overline{a_n} \cdot \overline{b_n})}_{c} + a_n \cdot b_n \cdot c_{n-1}$$

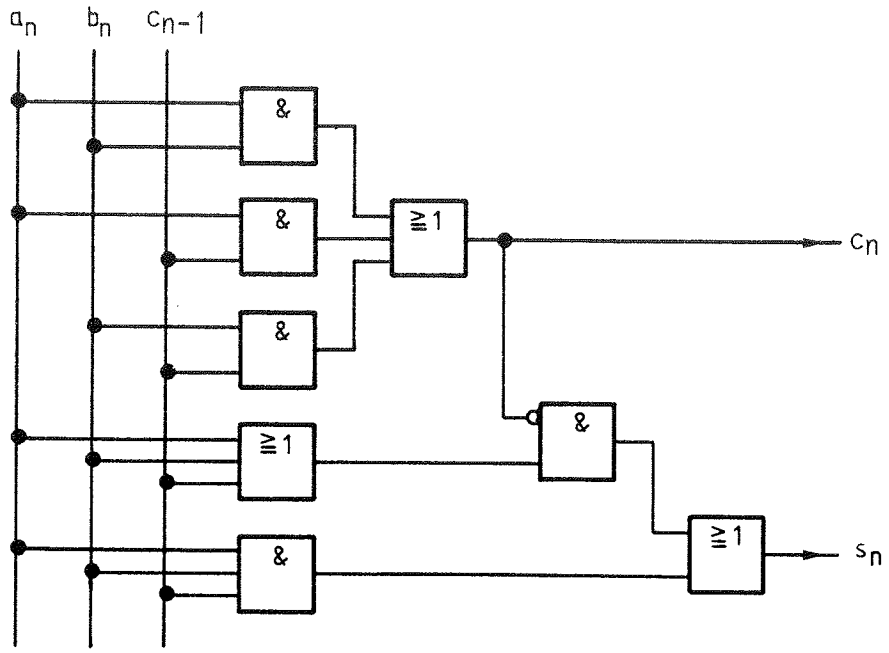
Put the bracket term c in the Karnaugh-Diagram so that the marked fields can supplement themselves with the carry fields, i.e.: $c = \overline{c_n}$



Received:

$$s_n = (a_n + b_n + c_{n-1}) \cdot \overline{c_n} + a_n \cdot b_n \cdot c_{n-1}$$

Circuit:



Exercise 8 - 4:

Build a full-adder from two half-adders.

Solution 8 - 4:

a_n	b_n	c_{n-1}	s_n	c_2
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

For the half-adder only the functions with $c_{n-1} = 0$ are to be used.

Half-adder:

$$\boxed{s_1 = \bar{a}_n b_n + a_n \bar{b}_n}$$

$$= \overline{\bar{a}_n b_n + a_n \bar{b}_n} = \overline{(\bar{a}_n \cdot \bar{b}_n)(a_n \cdot \bar{b}_n)} = \overline{(a_n + \bar{b}_n)(\bar{a}_n + b_n)} =$$

$$= \overline{\underbrace{a_n \cdot \bar{a}_n}_{0} + a_n \cdot b_n + \bar{b}_n \cdot \bar{a}_n + \underbrace{\bar{b}_n \cdot b_n}_{0}}$$

$$\boxed{s_1 = \overline{a_n \cdot b_n + \bar{a}_n \cdot \bar{b}_n} = \overline{a_n \cdot b_n}\ (\bar{a}_n \cdot \bar{b}_n) = \overline{a_n \cdot b_n}\ (a_n + b_n)}$$

$$\boxed{c_1 = a_n \cdot b_n}$$

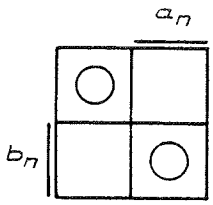
Full-adder:

Sum: $s_n = \bar{a}_n b_n \bar{c}_{n-1} + a_n \bar{b}_n \bar{c}_{n-1} + \bar{a}_n \bar{b}_n c_{n-1} + a_n b_n c_{n-1}$

$$= \bar{c}_{n-1}\ \underbrace{(\bar{a}_n b_n + a_n \bar{b}_n)}_{s_1} + c_{n-1}\ \underbrace{(\overline{a_n \bar{b}_n} + a_n b_n)}_{\bar{s}_1}$$

with $\overline{a_n \bar{b}_n} + a_n b_n = \overline{a_n \cdot \bar{b}_n + \bar{a}_n \cdot b_n} = \bar{s}_1$

following $s_n = \bar{c}_{n-1} \cdot s_1 + c_{n-1} \cdot \bar{s}_1$



with s_1 as sum function of a half-adder.

From the build-up the expression for s_n corresponds to the expression for s_1 also the above made conversion can be done.

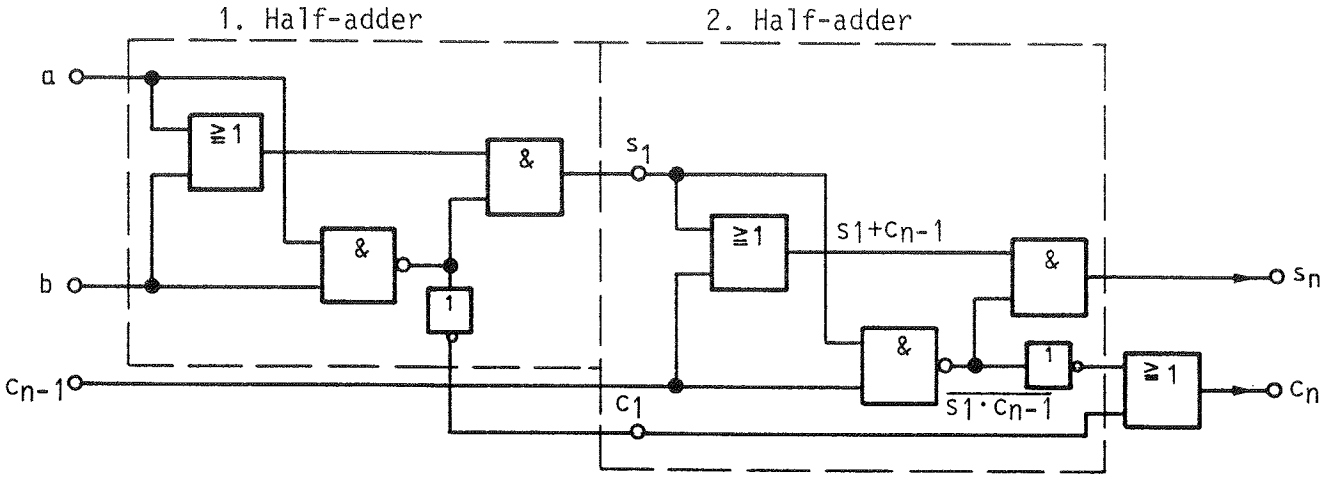
Therefore: $\boxed{s_n = \overline{s_1 \cdot c_{n-1}}\ (s_1 + c_{n-1})}$

Carry:
$$c_n = a_n b_n \overline{c_{n-1}} + \overline{a_n} b_n c_{n-1} + a_n \overline{b_n} c_{n-1} + a_n b_n c_{n-1}$$
$$= a_n b_n \underbrace{(\overline{c_{n-1}} + c_{n-1})}_{1} + c_{n-1} \underbrace{(\overline{a_n} b_n + a_n \overline{b_n})}_{s_1}$$

Therefore: $c_n = \underbrace{a_n b_n}_{c_1} + c_{n-1} \cdot s_1$

$$\boxed{c_n = c_1 + c_{n-1} \cdot s_1}$$

Circuit:



Exercise 8 - 5:

Design a logic operation circuit, which is working with the aid of a control function z alternatively as a half-adder and a half-subtracter (combined half-adder◄►half-subtracter).

Solution 8 - 5:

Half-adder: z = 1; a + b

Half-subtracter: z = 0; a - b

Function table:

a	b	s	c_s	d	c_d
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0

s = sum
c_s = carry sum
d = difference
c_d = carry difference (borrow)

Immediately s = d is recognized.

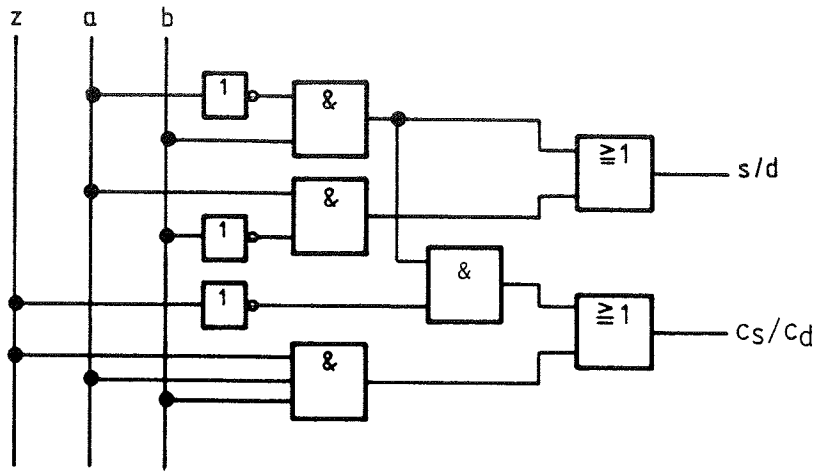
$$s = \bar{a}b + a\bar{b}$$
$$d = \bar{a}b + a\bar{b}$$

$c_s = z\ (ab)$; $c_d = \bar{z}\ (\bar{a}b)$

Mutual output for the carries:

$$c_s/c_d = z \cdot ab + \bar{z} \cdot \bar{a}b$$

Circuit:



Exercise 8 - 6:

Design a full-subtracter for a binary number (1 bit), $(a_n - b_n - c_{n-1})$.

Solution 8 - 6:

A full-subtracter subtracts two binary numbers a_n and b_n and considers like a full-adder the carry (borrower) c_{n-1} of the preceding place.

Function table with all possible cases:

a_n	b_n	c_{n-1}	d_n	c_n	number of combination
0	0	0	0	0	0
1	0	0	1	0	1
0	1	0	1	1	2
1	1	0	0	0	3
0	0	1	1	1	4
1	0	1	0	0	5
0	1	1	0	1	6
1	1	1	1	1	7

a_n = Minuend of the n^{th} binary place

b_n = Subtrahend of the n^{th} binary place

c_{n-1} = Borrower (borrowing carry) of the $(n-1)^{th}$ binary place

d_n = difference of the n^{th} binary place

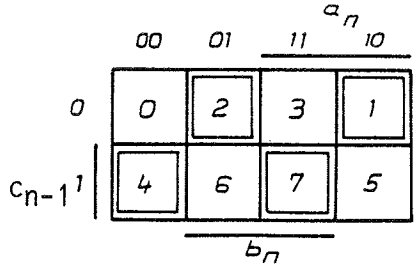
c_n = borrower of the n^{th} binary place

With the aid of the disjunctive standard form the following is received:

$d_n = 1 + 2 + 4 + 7$

$c_n = 2 + 4 + 6 + 7$

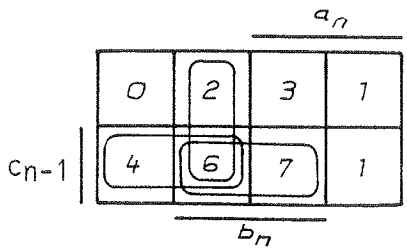
Simplification of the Karnaugh-Diagram:



For the difference no simplification is possible:

$$d_n = a_n\overline{b_n}\,\overline{c_{n-1}} + \overline{a_n}b_n\overline{c_{n-1}} + \overline{a_n}\,\overline{b_n}c_{n-1} + a_nb_nc_{n-1}$$

c_n:

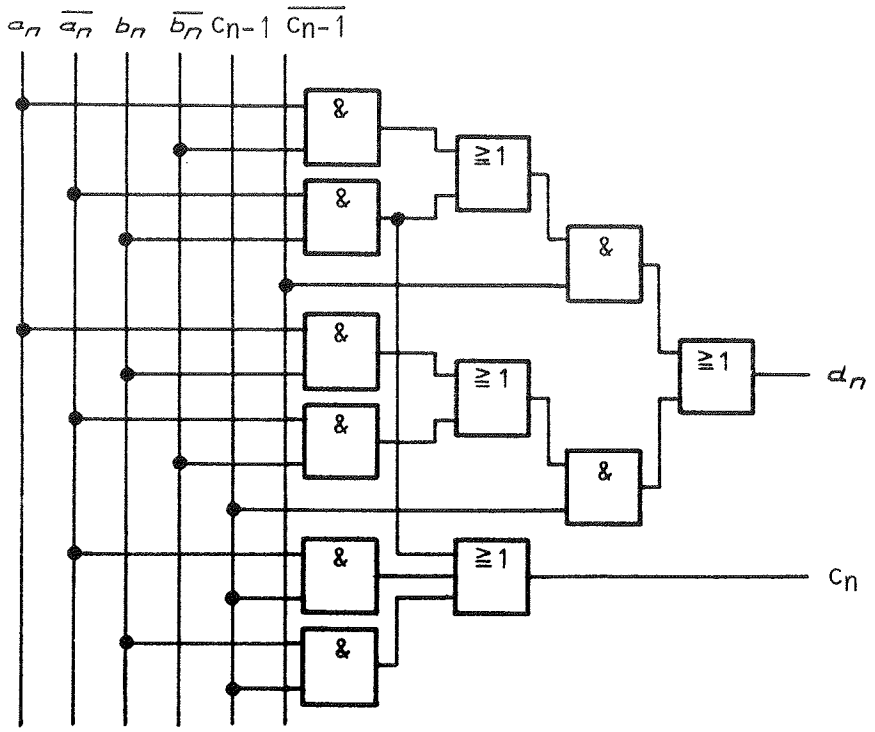


$$c_n = \overline{a_n} \cdot b_n + \overline{a_n} \cdot c_{n-1} + b_n \cdot c_{n-1}$$

The difference expression still can be simplified by means of factoring out:

$$d_n = \overline{c_{n-1}} \underbrace{(a_n \cdot \overline{b_n} + \overline{a_n} \cdot b_n)}_{\text{antivalence}} + c_{n-1} \underbrace{(a_n \cdot b_n + \overline{a_n} \cdot \overline{b_n})}_{\text{equivalence}}$$

Circuit:

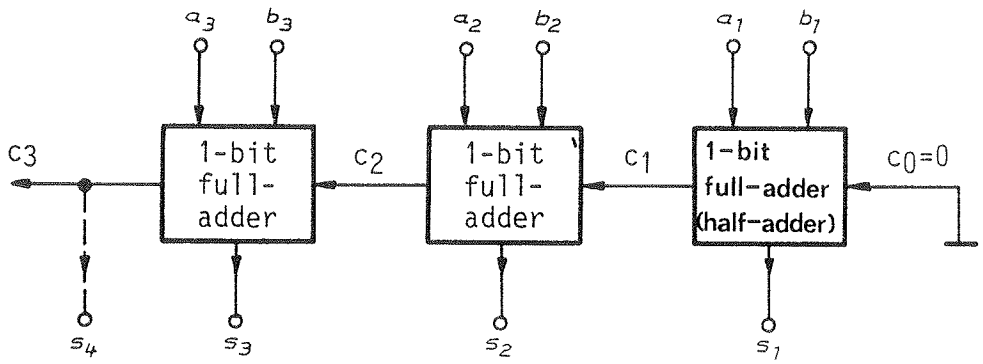


Exercise 8 - 7:

Specify a principle circuit for a parallel adder, which is built up from 1 bit full-adders.

Solution 8 - 7:

The simultaneous addition of two multi-bit binary numbers can be accomplished with a parallel adder, if both sum terms are available at the same time. For each binary number a full-adder is needed. The neighbouring adders have to be connected in such a way that the carry from one place can be passed on directly to the next highest binary number. As an example the addition of two 3-placed binary numbers is shown.



The sum does not appear immediately due to the delay time of adders. Only after the total delay time is the correct sum available at the output.
There are special circuits to reduce the delay time of the carry formation (jump-method, parallel carry generation).
The adder of the lowest binary place can be a half-adder, because there are no carry inputs.

8.2 Forming the complement

A subtraction can be reduced to an addition and complementation of the subtrahend, as the example of the decimal system shows:

a - b = c; for example: a = 57 b = 26 57 - 26 = 31

or:

```
  57
+ 74   Complement to 100 (100 - 26)
 ¹31
  └──► Carry is separated
```

At the decimal place the ten-complement can be used:

```
  7          7
- 4        + 6   ──► ten complement (10 - 4)
  3         ¹3
```

A further complement is the nine-complement. After the first addition the missing 1 (difference to 10) has to be added. In practice the carry, which always occurs in its highest number, is added.

Example:

```
  8          8
- 2        + 7   ──► nine-complement
  6         ¹5
        ┌─
        └► + 1
           6
```

With binary numbers the complementation is by the same principle. Because a binary number has 2 states only here the ten-complement corresponds with the two-complement.

Nine-complement for the 8421-Code

The circuit for forming the nine-complement in the 8421-Code is designed in such a way that the result appears in the function table, e.g. nine-complement of 7 is 2 (nine-complement of a number = difference between this number and nine).

Exercise 8 - 8:

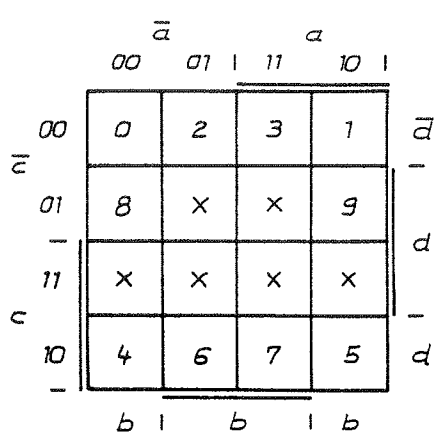
Develop a circuit to form the nine-complement of the 8421-Code.

Solution 8 - 8:

Realization as with the codes from chapter 7.

Dec. No.	d	c	b	a	x	w	v	u	9-complement
0	0	0	0	0	1	0	0	1	9 - 0 = 9
1	0	0	0	1	1	0	0	0	9 - 1 = 8
2	0	0	1	0	0	1	1	1	9 - 2 = 7
3	0	0	1	1	0	1	1	0	9 - 3 = 6
4	0	1	0	0	0	1	0	1	9 - 4 = 5
5	0	1	0	1	0	1	0	0	9 - 5 = 4
6	0	1	1	0	0	0	1	1	9 - 6 = 3
7	0	1	1	1	0	0	1	0	9 - 7 = 2
8	1	0	0	0	0	0	0	1	9 - 8 = 1
9	1	0	0	1	0	0	0	0	9 - 9 = 0
	8421-Code				nine-complement				

Simplification with Karnaugh-Diagram:



u = 0 + 2 + 4 + 6 + 8

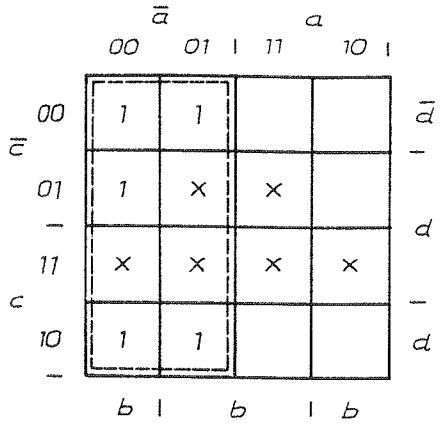
v = 2 + 3 + 6 + 7

w = 2 + 3 + 4 + 5

x = 0 + 1

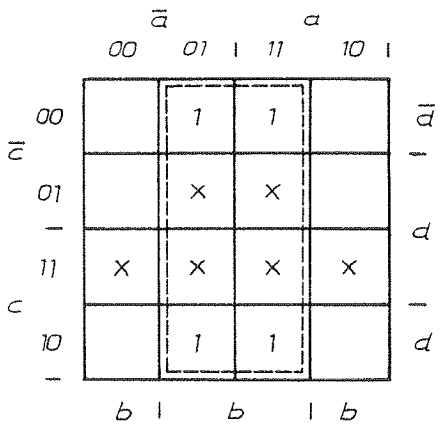
(numbers are representative for corresponding combinations)

u:



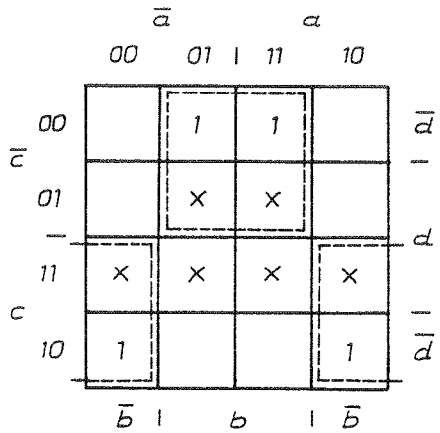
$u = \bar{a}$

v:



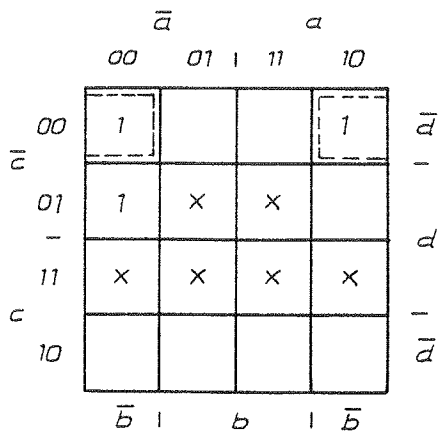
$v = b$

w:



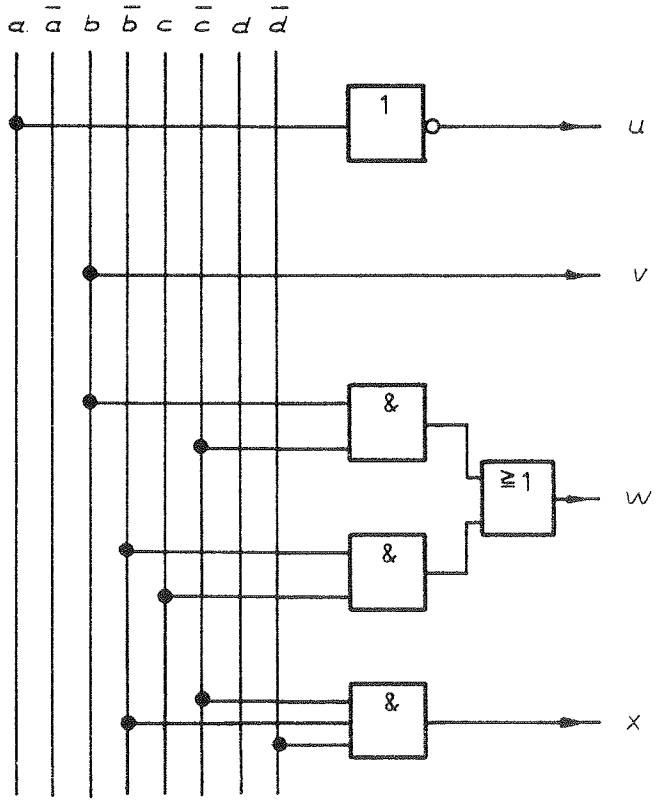
$w = b\bar{c} + \bar{b}c$

x:



$x = \bar{b}\bar{c}\bar{d}$

Circuit:



Exercise 8 - 9:

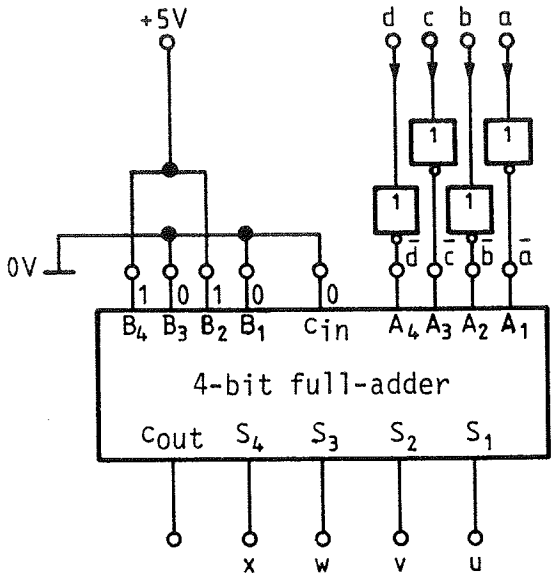
Realize the 9-complement of the 8421-Code by means of a 4-bit full-adder.

Solution 8 - 9:

With the aid of a 4-bit full-adder the nine-complement can be obtained easier.
For this the 4-placed binary number has to be inverted and the number 10 has to be added. The subsequent-table is explanatory.

Dec. number	d	c	b	a	$\bar{d}$	$\bar{c}$	$\bar{b}$	$\bar{a}$			+ 10	x	w	v	u
0	0	0	0	0	1	1	1	1	$\hat{=}$	15	25 = 1	1	0	0	1
1	0	0	0	1	1	1	1	0	$\hat{=}$	14	24 = 1	1	0	0	0
2	0	0	1	0	1	1	0	1	$\hat{=}$	13	23 = 1	0	1	1	1
3	0	0	1	1	1	1	0	0	$\hat{=}$	12	22 = 1	0	1	1	0
4	0	1	0	0	1	0	1	1	$\hat{=}$	11	21 = 1	0	1	0	1
5	0	1	0	1	1	0	1	0	$\hat{=}$	10	20 = 1	0	1	0	0
6	0	1	1	0	1	0	0	1	$\hat{=}$	9	19 = 1	0	0	1	1
7	0	1	1	1	1	0	0	0	$\hat{=}$	8	18 = 1	0	0	1	0
8	1	0	0	0	0	1	1	1	$\hat{=}$	7	17 = 1	0	0	0	1
9	1	0	0	1	0	1	1	0	$\hat{=}$	6	16 = 1	0	0	0	0
	binary number				inverted						will be split off	nine-complement			

Circuit:



Exercise 8 - 10:

State a circuit, which forms the two-complement of the binary code and convince yourself that the same configuration converts the two-complement in the original binary code.

Solution 8 - 10:

Subtraction from binary numbers can be reduced to an addition of the two-complement. As an example the conversion of the 4-placed binary code into the corresponding two-complement is shown. The two-complement is obtained by inverting the binary numbers and addition of number 1 to the lowest binary place.

Function table:

Dec.	d	c	b	a	$\bar{d}$	$\bar{c}$	$\bar{b}$	$\bar{a}$	d'	c'	b'	a'			
0	0	0	0	0	1	1	1	1	0	0	0	0	≙	16	16 + 0 = 16
1	0	0	0	1	1	1	1	0	1	1	1	1	≙	15	15 + 1 = 16
2	0	0	1	0	1	1	0	1	1	1	1	0	≙	14	14 + 2 = 16
3	0	0	1	1	1	1	0	0	1	1	0	1	≙	13	13 + 3 = 16
4	0	1	0	0	1	0	1	1	1	1	0	0	≙	12	12 + 4 = 16
5	0	1	0	1	1	0	1	0	1	0	1	1	≙	11	11 + 5 = 16
6	0	1	1	0	1	0	0	1	1	0	1	0	≙	10	10 + 6 = 16
7	0	1	1	1	1	0	0	0	1	0	0	1	≙	9	9 + 7 = 16
8	1	0	0	0	0	1	1	1	1	0	0	0	≙	8	8 + 8 = 16
9	1	0	0	1	0	1	1	0	0	1	1	1	≙	7	7 + 9 = 16
10	1	0	1	0	0	1	0	1	0	1	1	0	≙	6	6 + 10 = 16
11	1	0	1	1	0	1	0	0	0	1	0	1	≙	5	5 + 11 = 16
12	1	1	0	0	0	0	1	1	0	1	0	0	≙	4	4 + 12 = 16
13	1	1	0	1	0	0	1	0	0	0	1	1	≙	3	3 + 13 = 16
14	1	1	1	0	0	0	0	1	0	0	1	0	≙	2	2 + 14 = 16
15	1	1	1	1	0	0	0	0	0	0	0	1	≙	1	1 + 15 = 16
					inverted				two-complement						

From the function table it is recognizable that the two-complement of a binary number is the completion of this number to the maximal number of the n-placed binary number (here 15) plus 1 makes (16).

For the realization the same applies as with a code-converter.

a' = 1 + 3 + 5 + 7 + 9 + 11 + 13 + 15

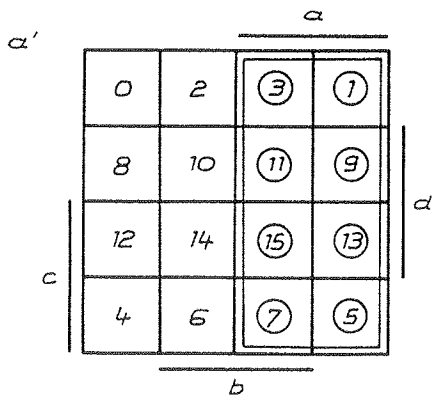
b' = 1 + 2 + 5 + 6 + 9 + 10 + 13 + 14

c' = 1 + 2 + 3 + 4 + 9 + 10 + 11 + 12

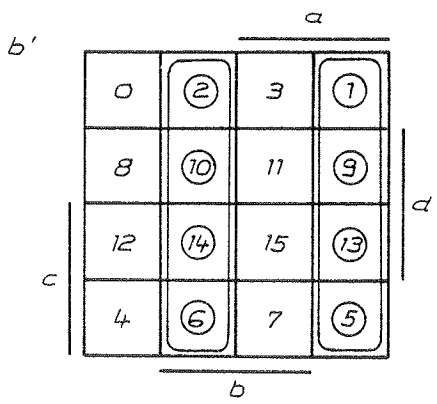
d' = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8

Here too the decimal numbers represent the corresponding combinations of the function table.

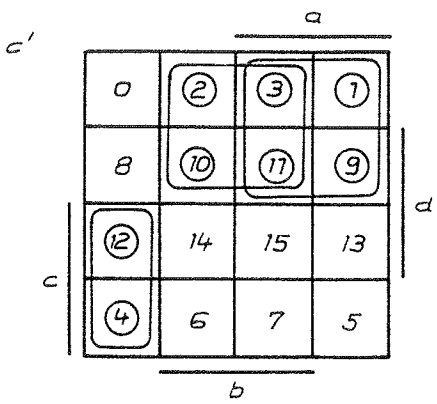
Simplification:



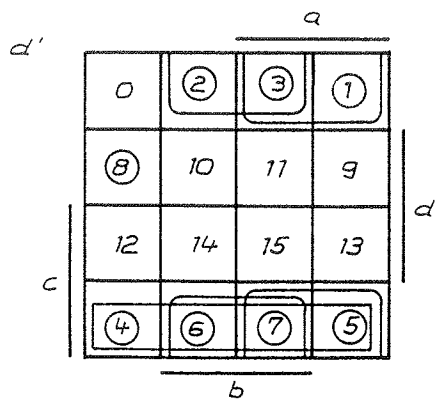
$a' = a$



$b' = \bar{a}b + a\bar{b}$



$c' = b\bar{c} + a\bar{c} + \bar{a}\bar{b}c$

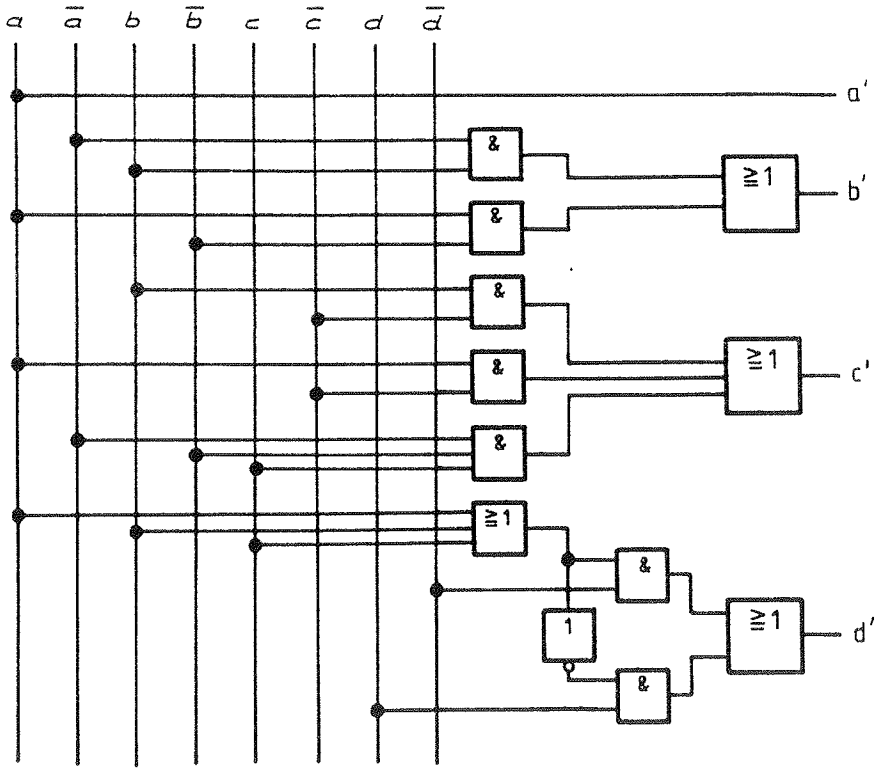


$d' = c\bar{d} + b\bar{d} + a\bar{d} + \bar{a}\bar{b}\bar{c}d$

$= \bar{d}\,(a + b + c) + \bar{a}\bar{b}\bar{c}d$

$= \bar{d}\,(a + b + c) + \overline{(a + b + c)} \cdot d$

Circuit (type 1):



Looking at the table it is recognized immediately that the same operation is applied for the conversion of the two-complement into a binary code.

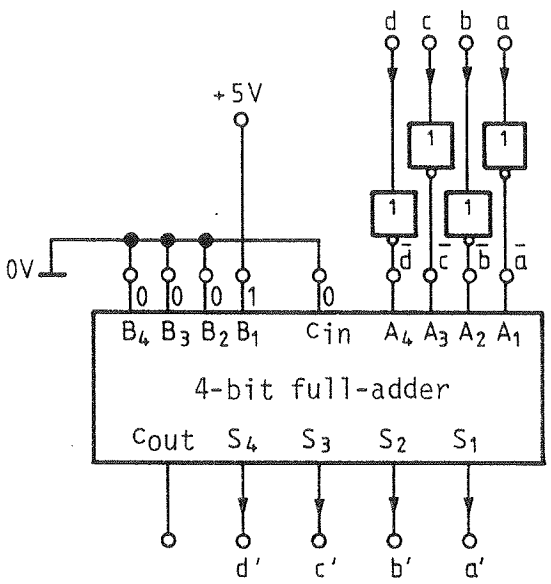
Exercise 8 - 11:

Evaluate the two-complement of the binary code by means of a 4-bit full-adder.

Solution 8 - 11:

As there is a 4-bit full-adder in the teaching model, the two-complement can be formed somewhat simplier, considering the fact that the two-complement is yielded by inverting the binary number plus addition of a one to the lowest place.

Circuit (type 2):



Exercise 8 - 12:

Develop a circuit which forms the nine-complement of the Aiken-code.

Solution 8 - 12:

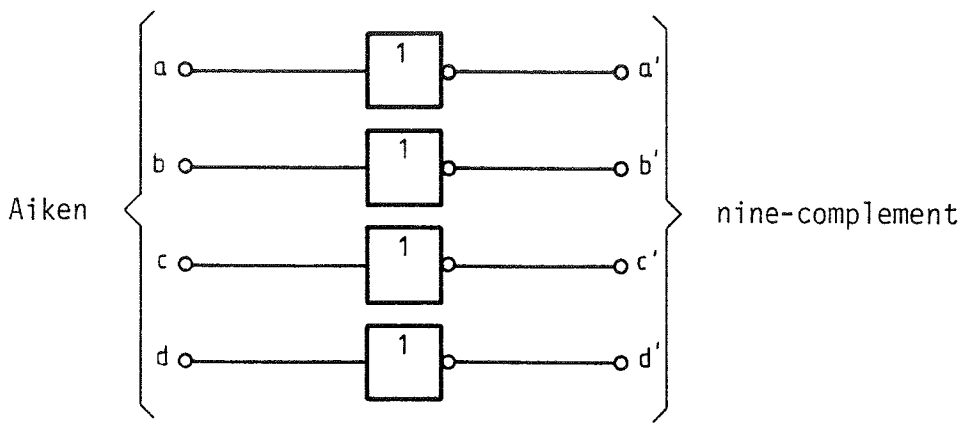
Also here the nine-complement of the code-word in the Aiken-code is the completion of this word to 9.

Function table:

Dec.	d	c	b	a	$\bar{d}$	$\bar{c}$	$\bar{b}$	$\bar{a}$		
0	0	0	0	0	1	1	1	1	$\hat{=}$	9
1	0	0	0	1	1	1	1	0	$\hat{=}$	8
2	0	0	1	0	1	1	0	1	$\hat{=}$	7
3	0	0	1	1	1	1	0	0	$\hat{=}$	6
4	0	1	0	0	1	0	1	1	$\hat{=}$	5
5	1	0	1	1	0	1	0	0	$\hat{=}$	4
6	1	1	0	0	0	0	1	1	$\hat{=}$	3
7	1	1	0	1	0	0	1	0	$\hat{=}$	2
8	1	1	1	0	0	0	0	1	$\hat{=}$	1
9	1	1	1	1	0	0	0	0	$\hat{=}$	0
	Aiken				nine-complement					

Obviously the nine-complement is the inversion of the code-word, therefore the Aiken-code is especially suitable for arithmetic operations.

Circuit:



Exercise 8 - 13:

Develop a circuit which forms the nine-complement of the Excess-3-code.

Solution 8 - 13:

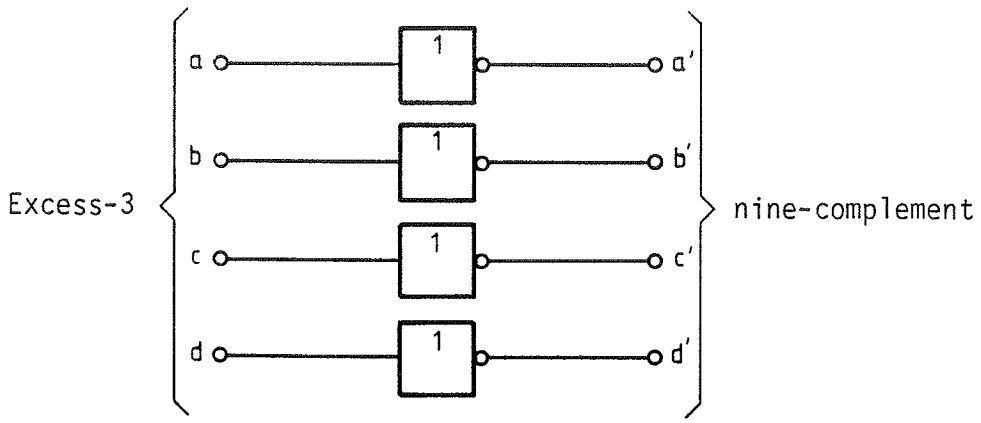
The solution corresponds to the solution of exercise 8 - 12.

Function table:

Dec.	d	c	b	a	d'	c'	b'	a'		
0	0	0	1	1	1	1	0	0	≙	9
1	0	1	0	0	1	0	1	1	≙	8
2	0	1	0	1	1	0	1	0	≙	7
3	0	1	1	0	1	0	0	1	≙	6
4	0	1	1	1	1	0	0	0	≙	5
5	1	0	0	0	0	1	1	1	≙	4
6	1	0	0	1	0	1	1	0	≙	3
7	1	0	1	0	0	1	0	1	≙	2
8	1	0	1	1	0	1	0	0	≙	1
9	1	1	0	0	0	0	1	1	≙	0
	Excess-3				nine-complement					

Also here the nine-complement is the inversion of the Excess-3-code.

Circuit:



Exercise 8 - 14:

Convert the two-complement of the BCD-binary code into decimal system.

Solution 8 - 14:

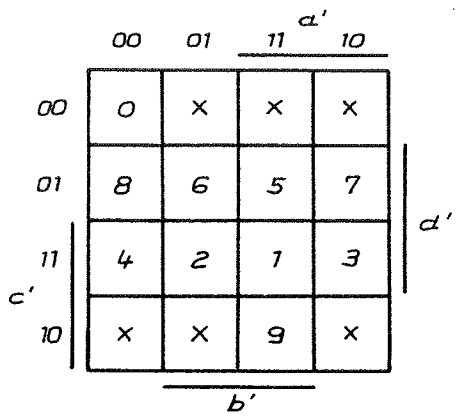
The conversion corresponds to a decoding.

Function table:

d'	c'	b'	a'	Dec.
0	0	0	0	0
1	1	1	1	1
1	1	1	0	2
1	1	0	1	3
1	1	0	0	4
1	0	1	1	5
1	0	1	0	6
1	0	0	1	7
1	0	0	0	8
0	1	1	1	9

The two-complement of number 10 to 15 should not appear.
(X ≙ don't-care-fields)

Karnaugh-diagram

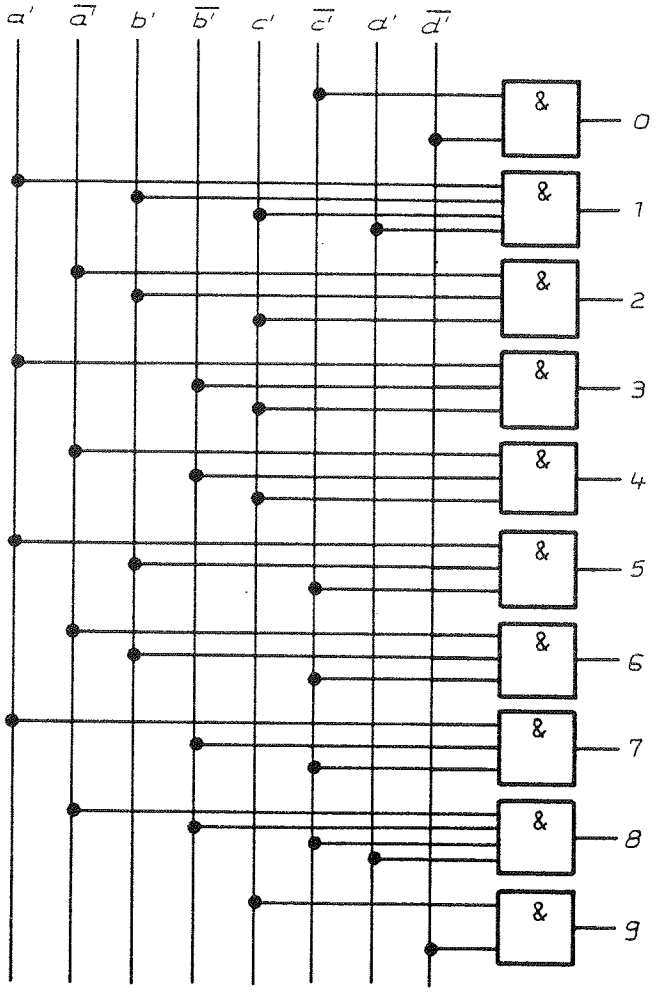


The solution succeeds as with a decoder, whereby this can be simplified with the Karnaugh-diagram and the don't-care-fields.

Equations:

0 = $\bar{c}'\bar{d}'$; 1 = a'b'c'd'; 2 = $\bar{a}'$b'c'; 3 = a'$\bar{b}'$c'
4 = $\bar{a}'\bar{b}'$c'; 5 = a'b'$\bar{c}'$; 6 = $\bar{a}'$b'$\bar{c}'$; 7 = a'$\bar{b}'\bar{c}'$
8 = $\bar{a}'\bar{b}'\bar{c}'$d'; 9 = c'$\bar{d}'$;

Circuit:



Exercise 8 - 15:

State a logic operation, that reconverts the nine-complement of the BCD-binary code in the decimal system.

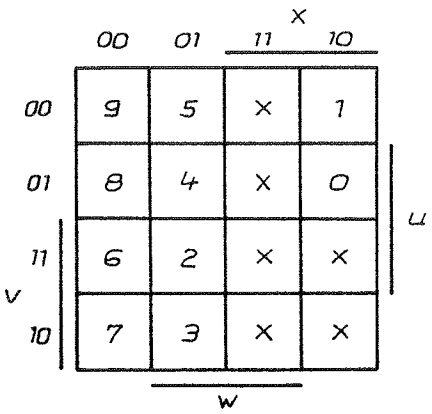
Solution 8 - 15:

(Deduction as with the two-complement → Dec.)

Function table

x	w	v	u	Dec.
1	0	0	1	0
1	0	0	0	1
0	1	1	1	2
0	1	1	0	3
0	1	0	1	4
0	1	0	0	5
0	0	1	1	6
0	0	1	0	7
0	0	0	1	8
0	0	0	0	9

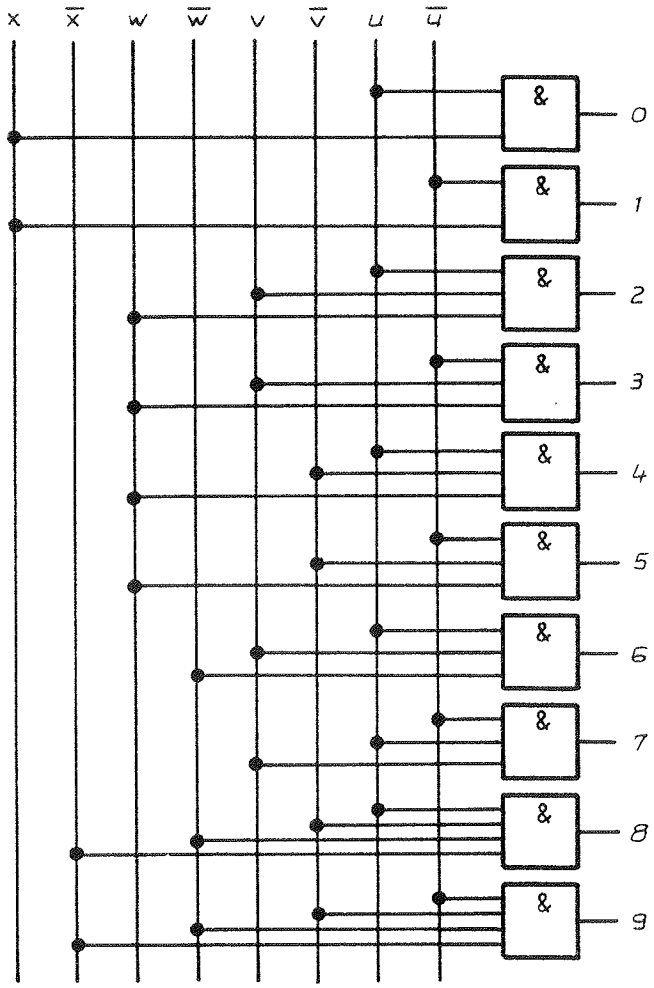
Karnaugh-diagram



$0 = ux$; $1 = \bar{u}x$; $2 = uvw$; $3 = \bar{u}vw$; $4 = u\bar{v}w$

$5 = \bar{u}\bar{v}w$; $6 = uv\bar{w}$; $7 = \bar{u}v\bar{w}$; $8 = u\bar{v}\bar{w}\bar{x}$; $9 = \bar{u}\bar{v}\bar{w}\bar{x}$

Circuit:


x
$\overline{x}$
w
$\overline{w}$
v
$\overline{v}$
u
$\overline{u}$
&
0
1
2
3
4
5
6
7
8
9

8.3 BCD-adder

Exercise 8 - 16:

Develop a correction circuit with AND- and OR-gates, which will convert the sum of a 4-bit full-adder coded in the binary code to BCD.

Solution 8 - 16:

When adding two BCD-binary numbers with a 4-bit full-adder, the sum can contain the binary numbers form 10 to 19 (9 + 9 + 1) which do not appear in BCD-form. When adding decimal numbers of several places they have to be presented as binary coded decimal numbers because of coding and decoding.
Therefore combinations from 10 to 18 have to be converted into the BCD-binary codes with decade-carry. A correction-circuit is necessary.

Function table

Dec. number	c_{out}	s_4	s_3	s_2	s_1	c_{out}	s_4'	s_3'	s_2'	s_1'	
0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	1	
2	0	0	0	1	0	0	0	0	1	0	
3	0	0	0	1	1	0	0	0	1	1	
4	0	0	1	0	0	0	0	1	0	0	
5	0	0	1	0	1	0	0	1	0	1	
6	0	0	1	1	0	0	0	1	1	0	
7	0	0	1	1	1	0	0	1	1	1	
8	0	1	0	0	0	0	1	0	0	0	
9	0	1	0	0	1	0	1	0	0	1	
10	0	1	0	1	0	1	0	0	0	0	(≙ 16)
11	0	1	0	1	1	1	0	0	0	1	(≙ 17)
12	0	1	1	0	0	1	0	0	1	0	(≙ 18)
13	0	1	1	0	1	1	0	0	1	1	(≙ 19)
14	0	1	1	1	0	1	0	1	0	0	(≙ 20)
15	0	1	1	1	1	1	0	1	0	1	(≙ 21)
16	1	0	0	0	0	1	0	1	1	0	(≙ 22)
17	1	0	0	0	1	1	0	1	1	1	(≙ 23)
18	1	0	0	1	0	1	1	0	0	0	(≙ 24)
19	1	0	0	1	1	1	1	0	0	1	(≙ 25)

sum in binary codes — sum in BCD-binary codes

The realization is done according to the same principle as with a code-converter.

From the function table it is read: $\boxed{s_1' = s_1}$

$s_2' = 2 + 3 + 6 + 7 + 12 + 13 + 16 + 17$

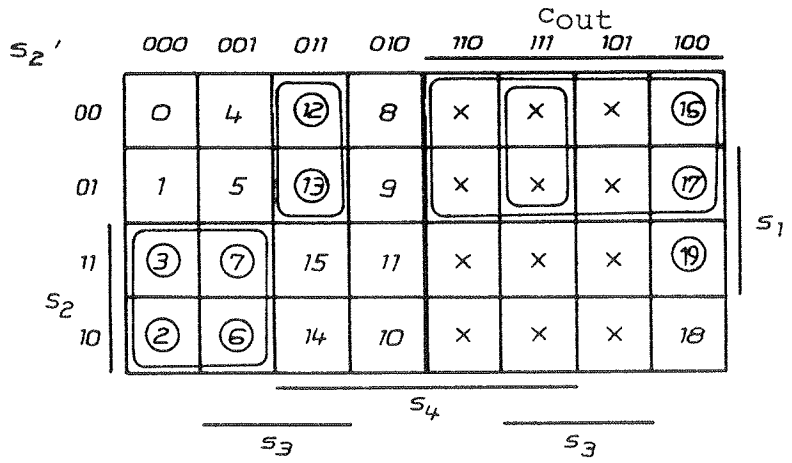
$s_3' = 4 + 5 + 6 + 7 + 14 + 15 + 16 + 17$

$s_4' = 8 + 9 + 18 + 19$

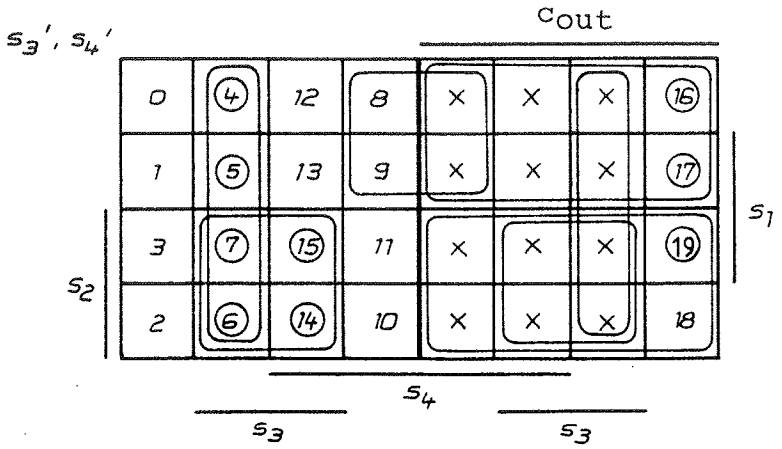
$c_{out}' = 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19$

Karnaugh-diagram (page 48)

X ≙ don't-care-fields (20 to 31)

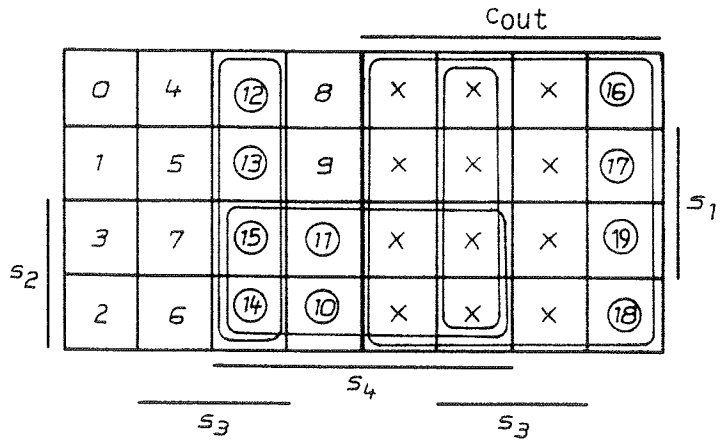


$$\boxed{s_2' = \overline{c_{out}} \cdot \overline{s_4} \cdot s_2 + s_4 \cdot s_3 \cdot \overline{s_2} + c_{out} \cdot \overline{s_2}}$$



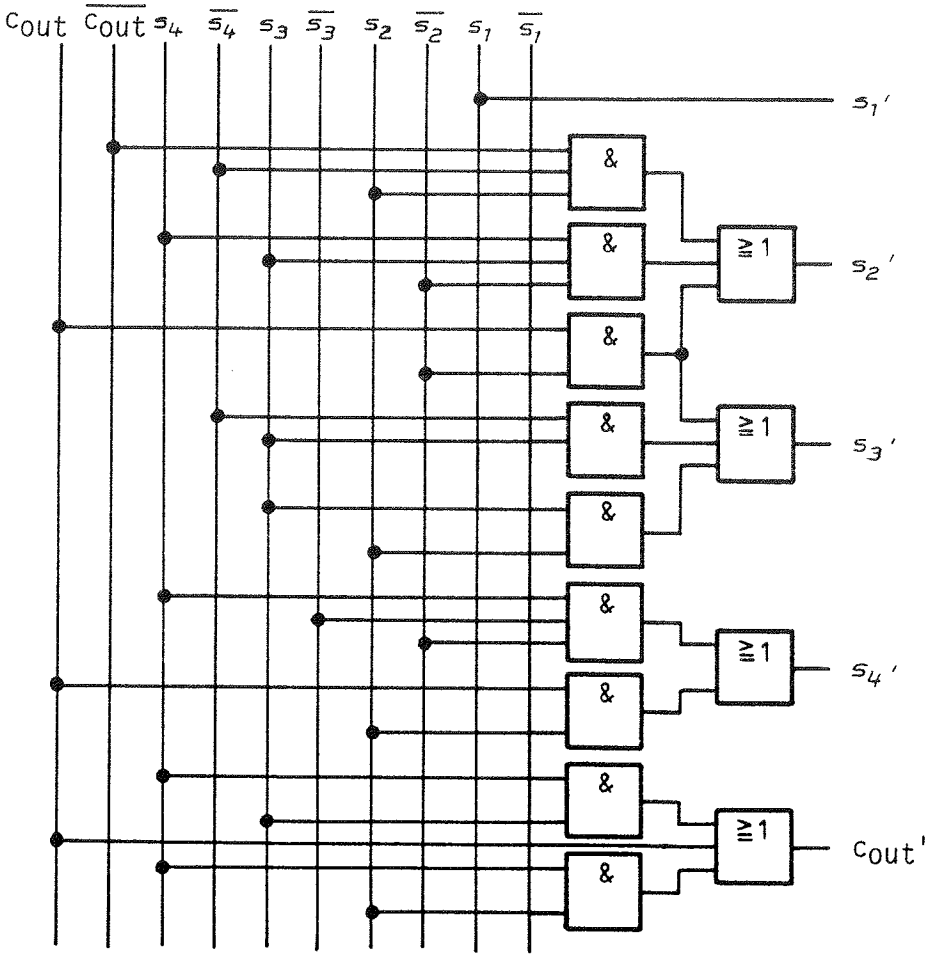
$$\boxed{s_3' = \overline{s_4} \cdot s_3 + s_3 \cdot s_2 + c_{out} \cdot \overline{s_2}}$$

$$\boxed{s_4' = s_4 \cdot \overline{s_3} \cdot \overline{s_2} + c_{out} \cdot s_2}$$



$$C_{out}' = s_4 \cdot s_3 + s_4 \cdot s_2 + C_{out}$$

Circuit (type 1)



Exercise 8 - 17:

Develop a BCD-correction-circuit (see exercise 8 - 16), but with an additional 4-bit full-adder.

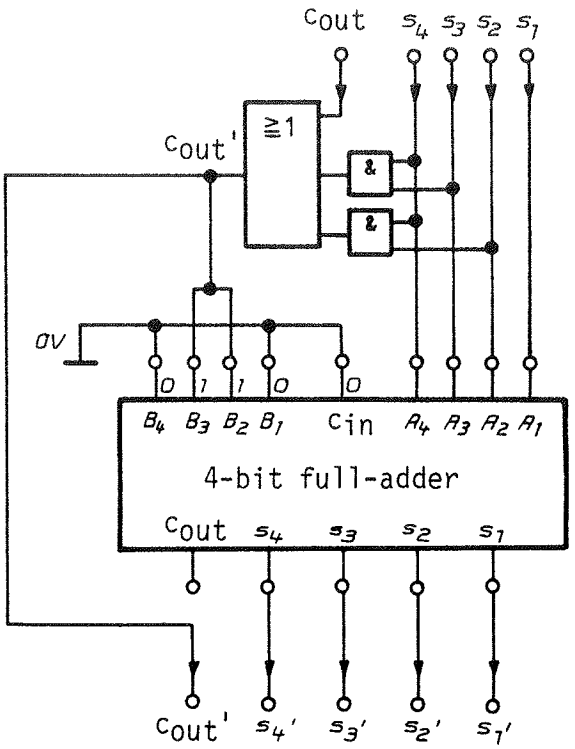
Solution 8 - 17:

The circuit of page 146 is very complicated. With the aid of a second 4-bit full-adder the correction-circuit can be made remarkably simpler.
From the function table on page 144 obviously the sum in BCD (s_4', s_3', s_2', s_1') is always bigger by six for the numbers from 10 to 18 compared to the sum in the binary code (s_4, s_3, s_2, s_1).
In case of $c_{out}' = 1$ for the sum $s_4\ s_3\ s_2\ s_1$ $6 \hat{=} 0\ 1\ 1\ 0$ has to be added. The corrected carry c_{out}' remains unchanged.

$$c_{out}' = s_4\ s_3 + s_4\ s_2 + c_{out}$$

$$c_{out}' = s_4 \cdot (s_2 + s_3) + c_{out}$$

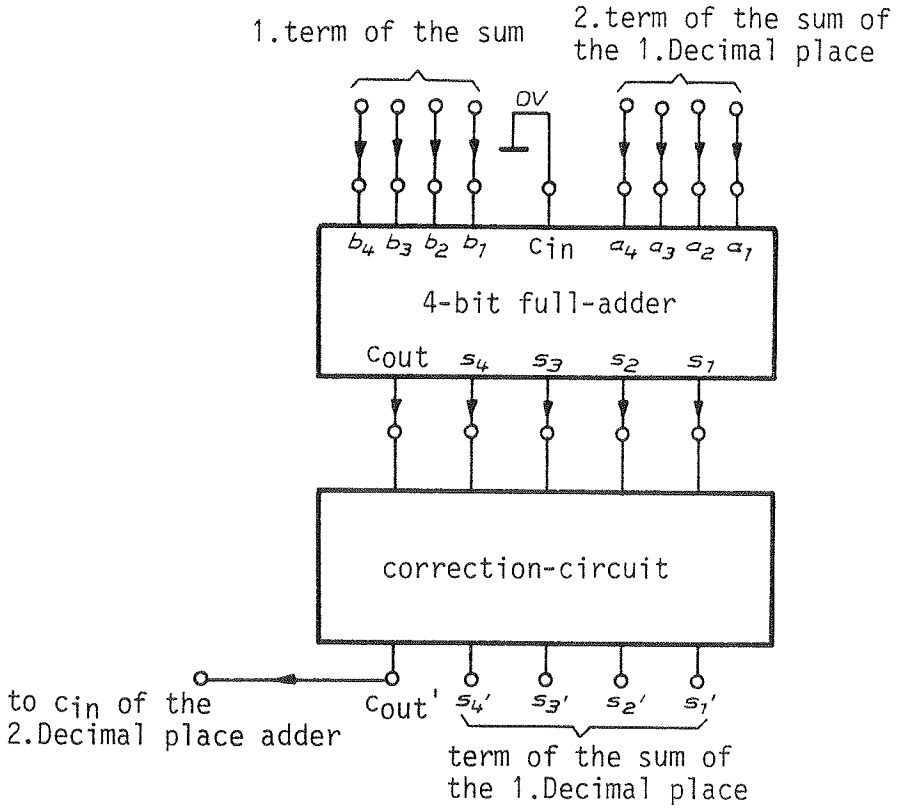
Circuit (type 2):



Exercise 8 - 18:

Design a complete BCD-adder circuit for the first Decimal-place.

Solution 8 - 18:



8.4 BCD-Subtracter

Exercise 8 - 19:

Design a BCD-subtracter circuit for one decimal-place, taking a 4-bit full-adder and a BCD-correction-circuit as a basis.

Solution 8 - 19:

With closed binary coded numbers a subtraction of two binary numbers can always be reduced to an addition with the two-complement.
For binary-coded decimal-numbers (BCD) this is only possible for such numbers where the minuend is bigger than the subtrahend. This is valid for each decimal place.

Example: 34 - 18 = 16; 18 ≙ 0001 1000

34 ≙ 0 0 1 1 0 1 0 0 ; Two-complement of 18:

1 1 1 1 1 0 0 0

	0	0	1	1	0	1	0	0
+	1	1	1	1	1	0	0	0
[1]	0	0	1	0	[0]	1	1	0
		2				12		

The solution is not correct (instead 6 → 12 or instead 1 → 2). A special logic circuit would have to be developed for correction if the carry is 0 (minuend < subtrahend). It is more convenient however, to use the same arithmetic-unit for a subtraction of BCD-numbers as for BCD-addition. For this instead of the two-complement the nine-complement of the minuend has to be used. The nine-complement of the minuend is the completion of the respective number to nine.

Mathematical operation:

Difference: $a - b = c$ with $a > b$

Nine-complement of b: $(9 - b)$

Addition of the nine-complement: $a + (9 - b) = \underbrace{a - b}_{c} + 9$

$$a + (9 - b) = c + 9$$

As the difference c is always bigger or equal to 1 (a > b) the result (c + 9) is always bigger or equal to 10, e.g. the BCD correction-circuit starts operating. Besides the adder's result (c + 9) is too big by the value of nine.

The BCD-correction-circuit reduces the numbers from 10 to 18 by the value of 10, i.e. by 1 too many, as shown on table:

number	c_{out}	s_4	s_3	s_2	s_1	c_{out}'	s_4'	s_3'	s_2'	s_1'	
.											
.											
.											
10	0	1	0	1	0	1	0	0	0	0	≙ 0 = 10 - 10
11	0	1	0	1	1	1	0	0	0	1	≙ 1 = 11 - 10
12	0	1	1	0	0	1	0	0	1	0	≙ 2 = 12 - 10
13	0	1	1	0	1	1	0	0	1	1	≙ 3 = 13 - 10
14	0	1	1	1	0	1	0	1	0	0	≙ 4 = 14 - 10
15	0	1	1	1	1	1	0	1	0	1	≙ 5 = 15 - 10
16	1	0	0	0	0	1	0	1	1	0	≙ 6 = 16 - 10
17	1	0	0	0	1	1	0	1	1	1	≙ 7 = 17 - 10
18	1	0	0	1	0	1	1	0	0	0	≙ 8 = 18 - 10
		binary					BCD				

This error is easy to correct if the carry $c_{out}' = 1$ is once again fed to the input c_{in} of the 4-bit full-adder (9 + 1 = 10).
The flow of a subtraction action is shown with two examples.

25 - 13 = 12

25 ≙ 0 0 1 0 | 0 1 0 1

13 ≙ 0 0 0 1 | 0 0 1 1

nine-complement of 13 ≙ 1 0 0 0 | 0 1 1 0
(8 | 6)

addition:

	0 0 1 0	0 1 0 1
+	1 0 0 0	0 1 1 0
	1 0 1 0	1 0 1 1
	10	11

BCD-correction:

1 | 0 0 0 0 | 1 | 0 0 0 1

+ 1 (carry)

+ 1

0 0 0 1	0 0 1 0
result: 1	2

23 - 15 = 8

23 ≙ 0 0 1 0 | 0 0 1 1

15 ≙ 0 0 0 1 | 0 1 0 1

nine-complement of 15 ≙ 1 0 0 0 | 0 1 0 0
(8 | 4)

addition:

	0 0 1 0	0 0 1 1
+	1 0 0 0	0 1 0 0
	1 0 1 0	0 1 1 1
	10	7

BCD-correction:

1 | 0 0 0 0 | 0 | 0 1 1 1

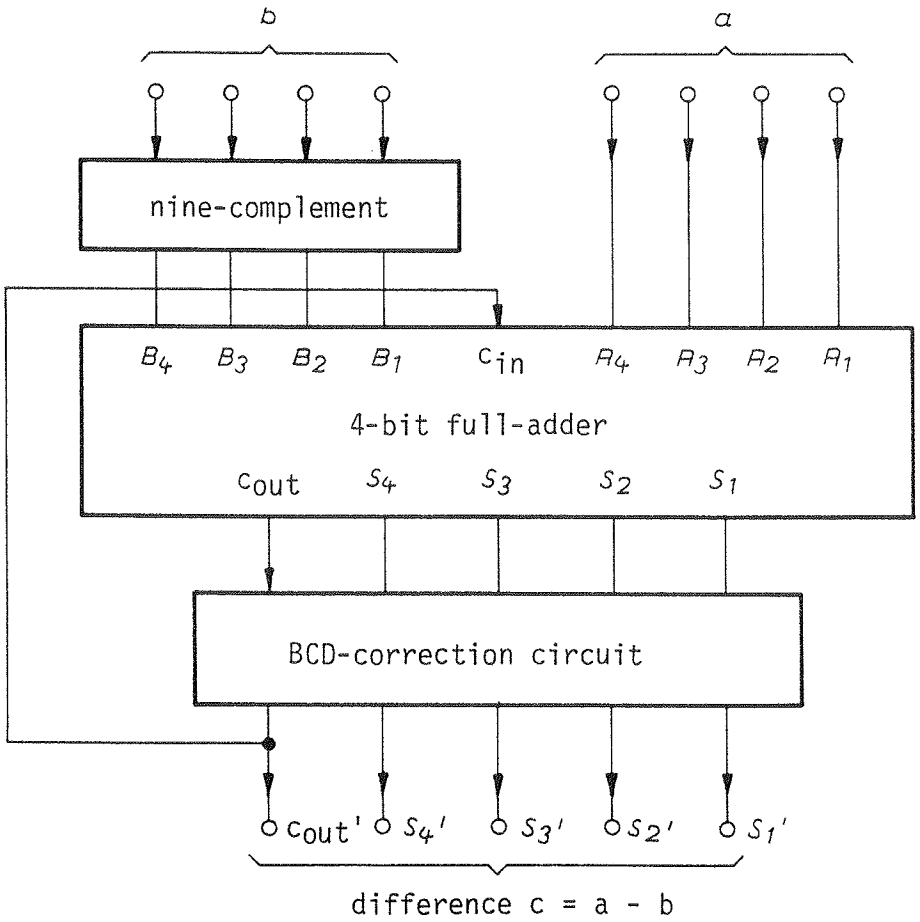
+ 0 (carry)

+ 1

0 0 0 0	1 0 0 0
result: 0	8

As seen from both examples with numbers with several decimal digits, the carry output of the highest place has to be added to the carry input of the 4-bit full-adder of the lowest place.

Circuit for one decimal-place:



8.5 Combined BCD-adder/subtracting-unit

Exercise 8 - 20:

Develop a combined BCD-adder/subtracting unit for one decimal place using solution 8 - 19 and 8 - 20.

Solution 8 - 20:

The nine-complement of number b = $B_4\ B_3\ B_2\ B_1$ (page 130 ... 132).

$U = \overline{B_1}$

$V = B_2$

$W = B_2 \cdot \overline{B_3} + \overline{B_2} \cdot B_3$

$X = \overline{B_1}\ \overline{B_2}\ \overline{B_3}$

Corresponds:

$B_1 \mathrel{\hat{=}} a$

$B_2 \mathrel{\hat{=}} b$

$B_3 \mathrel{\hat{=}} c$

$B_4 \mathrel{\hat{=}} d$

With a control order L the kind of calculating is decided.

Addition: $L = 1$

Subtraction: $L = 0$

At the B-inputs ($B_4'B_3'B_2'B_1'$) of the 4-bit full-adder the following has to be applied:

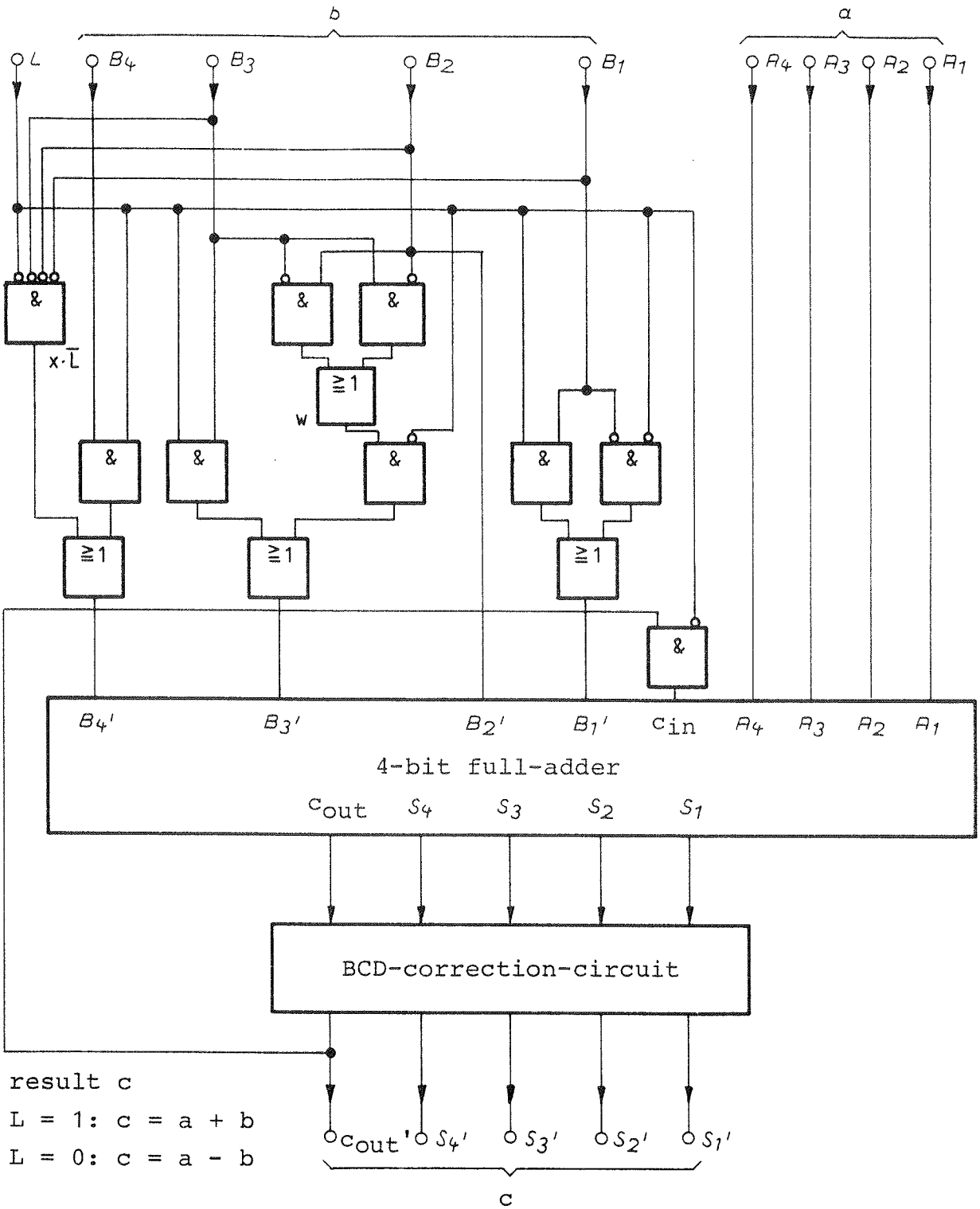
$B_1' = B_1 \cdot L + U \cdot \bar{L} = B_1 \cdot L + \overline{B_1} \cdot \bar{L}$

$B_2' = B_2$

$B_3' = B_3 \cdot L + W \cdot \bar{L} = B_3 \cdot L + (B_2 \cdot \overline{B_3} + \overline{B_2} \cdot B_3) \cdot \bar{L}$

$B_4' = B_4 \cdot L + X \cdot \bar{L} = B_4 \cdot L + \overline{B_1} \cdot \overline{B_2} \cdot \overline{B_3} \cdot \bar{L}$

Circuit:



At N-decimal places $c_{in\ 0} = 0$; $c_{out'n} = c_{in\ n+1}$; 0 = 1 Dec.place; n = 0...N has to be combined and in case of parallel-subtraction: $c_{in\ 0} = c_{out'N}$; $c_{out'n} = c_{in\ n+1}$.

8.6 Parallel multiplier

Exercise 8 - 21:

Develop a parallel multiplier for a two-placed binary code and work out the circuit in NAND-technique.

Solution 8 - 21:

a_2 a_1 = multiplier
b_2 b_1 = multiplicator

With 2 binary places the numbers 0, 1, 2, 3 can be represented, the largest product is 3 by 3 = 9. All possibilities have to be tested.

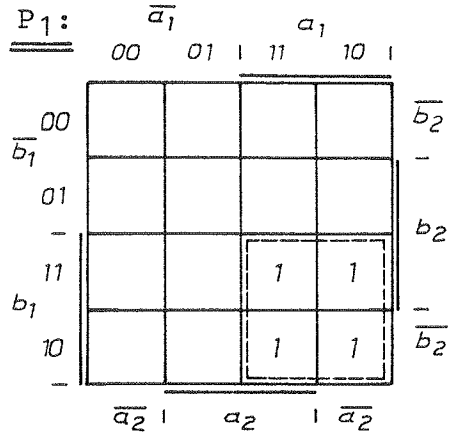
	a_2	a_1	b_2	b_1		P_4	P_3	P_2	P_1	= result
0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0		0	0	0	0	0
2	1	0	0	0		0	0	0	0	0
3	1	1	0	0		0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0
1	0	1	0	1		0	0	0	1	1
2	1	0	0	1		0	0	1	0	2
3	1	1	0	1		0	0	1	1	3
0	0	0	1	0	2	0	0	0	0	0
1	0	1	1	0		0	0	1	0	2
2	1	0	1	0		0	1	0	0	4
3	1	1	1	0		0	1	1	0	6
0	0	0	1	1	3	0	0	0	0	0
1	0	1	1	1		0	0	1	1	3
2	1	0	1	1		0	1	0	0	6
3	1	1	1	1		1	0	0	1	9

$P_1 = 1 + 3 + 9$
$P_2 = 2 + 3 + 6$
$P_3 = 4 + 6$
$P_4 = 9$

Layout

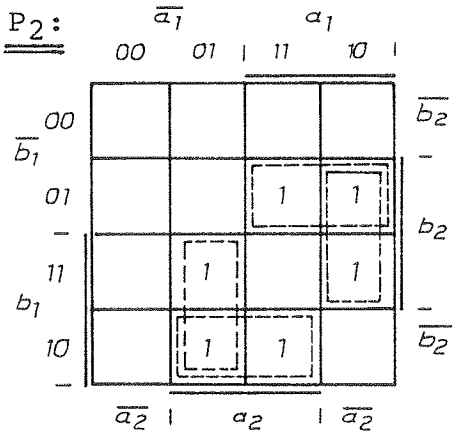
			0	2	3	1
		a_2	0	1	1	0
		a_1	0	0	1	1
	b_2	b_1				
0	0	0	0	0	0	0
2	1	0	0	4	6	2
3	1	1	0	6	9	3
1	0	1	0	2	3	1

P_1:



$P_1 = a_1\ b_1 = \overline{(\overline{a_1}\ \overline{b_1})}$

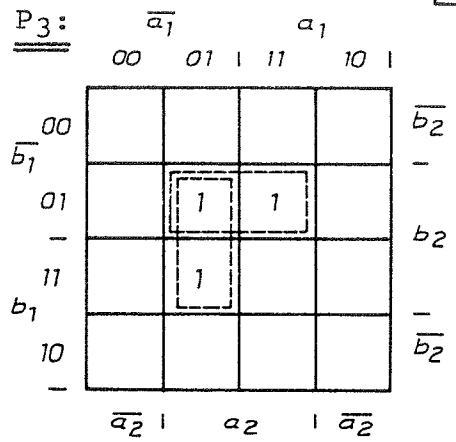
P_2:



$P_2 = a_1\ \overline{b_1}\ b_2 + a_1\ \overline{a_2}\ b_2 + \overline{a_1}\ a_2\ b_1 + a_2\ b_1\ \overline{b_2}$

$P_2 = \overline{a_1\ \overline{b_1}\ b_2} \cdot \overline{a_1\ \overline{a_2}\ b_2} \cdot \overline{\overline{a_1}\ a_2\ b_1} \cdot \overline{a_2\ b_1\ \overline{b_2}}$

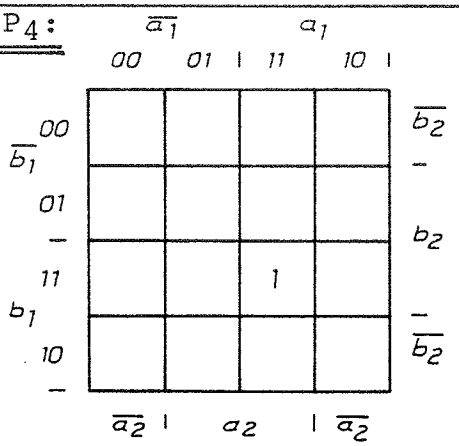
P_3:



$P_3 = a_2\ \overline{b_1}\ b_2 + \overline{a_1}\ a_2\ b_2$

$P_3 = \overline{a_2\ \overline{b_1}\ b_2} \cdot \overline{\overline{a_1}\ a_2\ b_2}$

P_4:

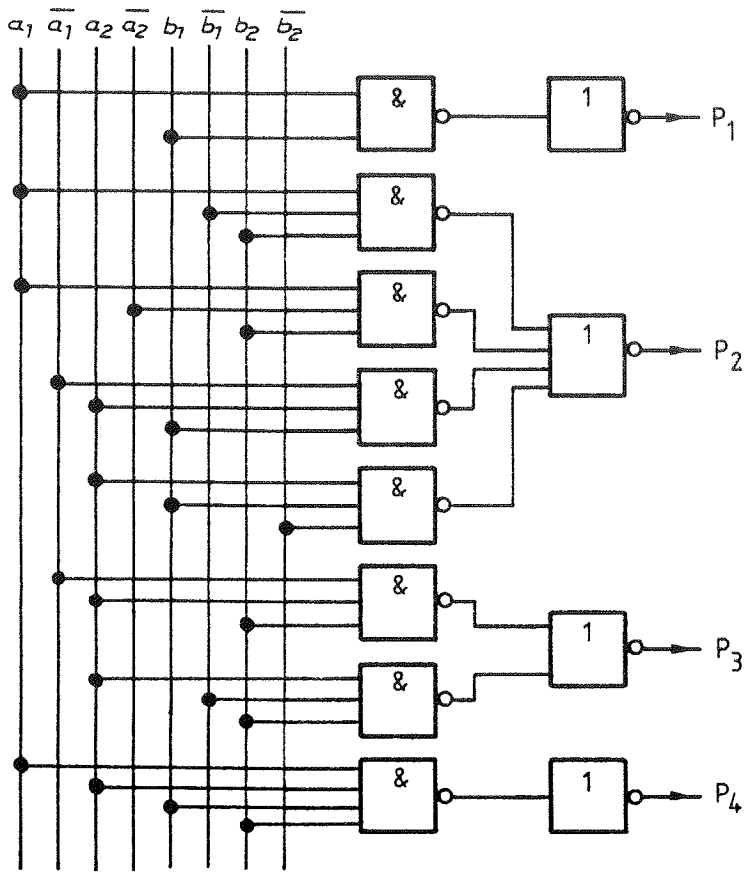


$P_4 = a_1\ a_2\ b_1\ b_2$

$P_4 = \overline{\overline{a_1}\ \overline{a_2}\ \overline{b_1}\ \overline{b_2}}$

Dec. system	P_4	P_3	P_2	P_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Circuit:



9. COMPARATOR-, THRESHOLD-, SELECTION-CIRCUITS

9.1 Comparator-circuits

Exercise 9 - 1:

Develop a comparator-circuit, that compares simultaneously the two 1-placed binary numbers a, b.

Solution 9 - 1:

a	b	$x \hat{=} a \equiv b$	$y \hat{=} a > b$	$z \hat{=} a < b$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

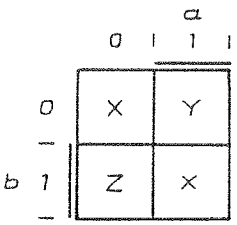
equivalence

$x = \bar{a}\bar{b} + ab$

$y = a\bar{b}$

$z = \bar{a}b$

The Karnaugh-diagram shows:

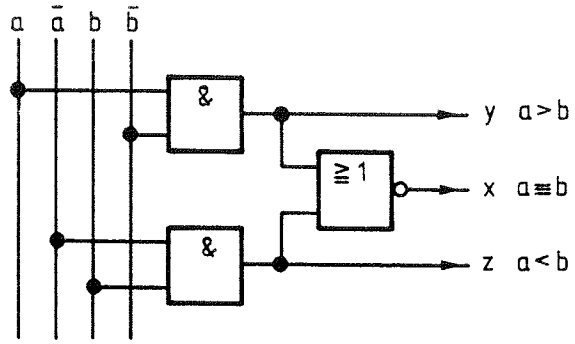


$\bar{x} = y + z$

or

$x = \overline{y + z}$ (equivalence circuit)

Circuit:



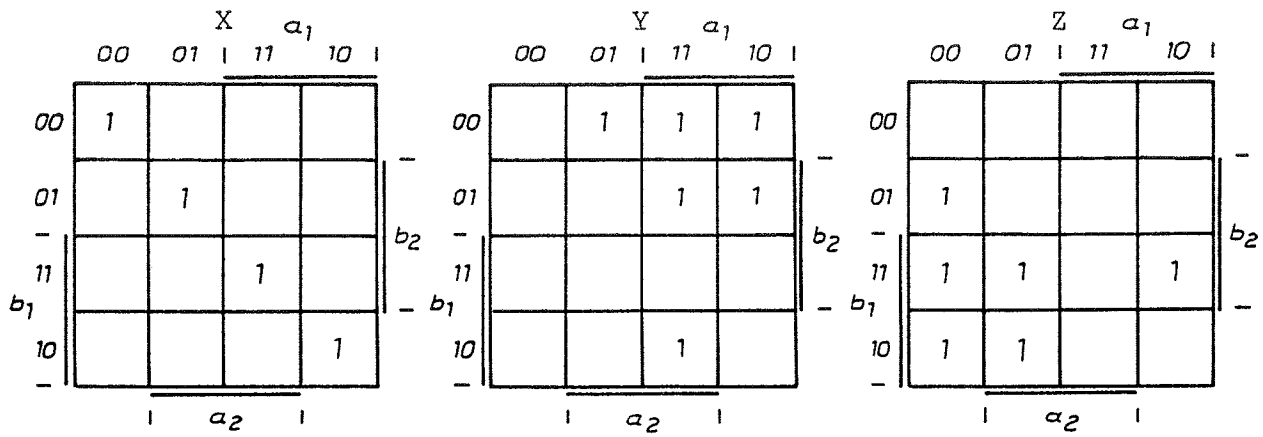
Exercise 9 - 2:

Develop a comparator-circuit, that compares simultaneously the two 2-placed binary numbers a, b.

Solution 9 - 2:

a		b				
a_1	a_2	b_1	b_2	$x \hat{=} a_1\, a_2 \equiv b_1\, b_2$	$y \hat{=} a_1\, a_2 > b_1\, b_2$	$z \hat{=} a_1\, a_2 < b_1\, b_2$
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0
↑	↑	↑	↑			
2^1	2^0	2^1	2^0			

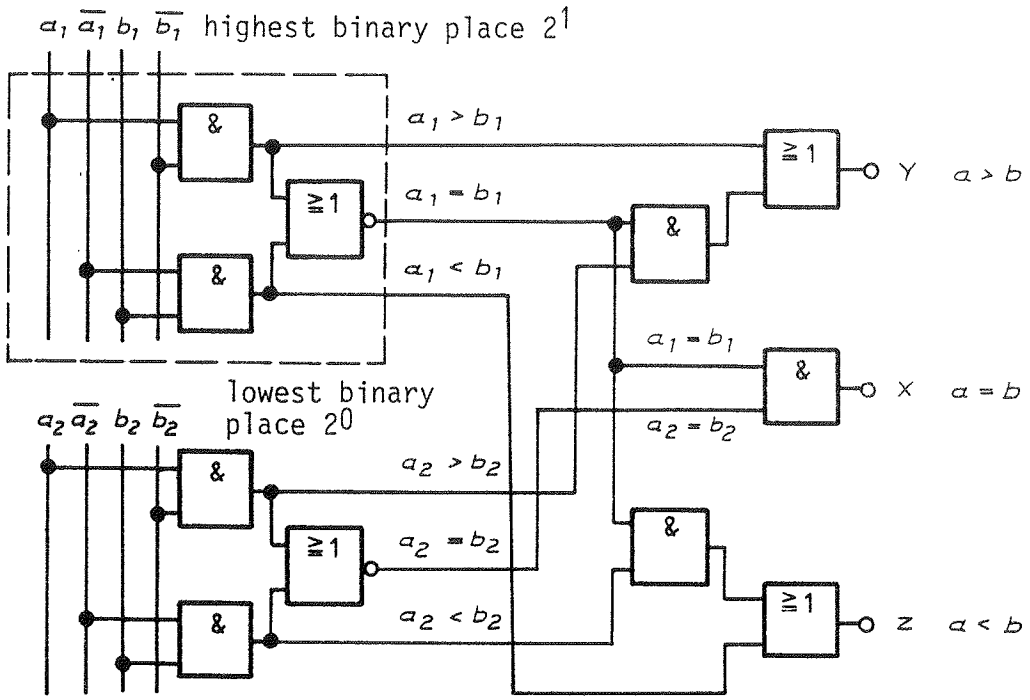
Layouts:



Each field of the Karnaugh-diagram is marked for one only output x, y or z, because only one output is possible. Now the terms can be simplified accordingly. If the problem is reduced to one 1-bit-comparator, then the solution becomes remarkably simpler and can easily be extended to binary numbers with more than 2 places. a_1 and b_1 are the highest places.

If $a_1 > b_1$ then the complete binary number is larger than b, or $a_1 < b_1 \rightarrow a < b$.

If the highest places are equal ($a_1 \equiv b_1$) then the lower places have to be compared: $a_2 = b_2$; $a_2 > b_2$; $a_2 < b_2$. This comparison was already carried out on the previous page.

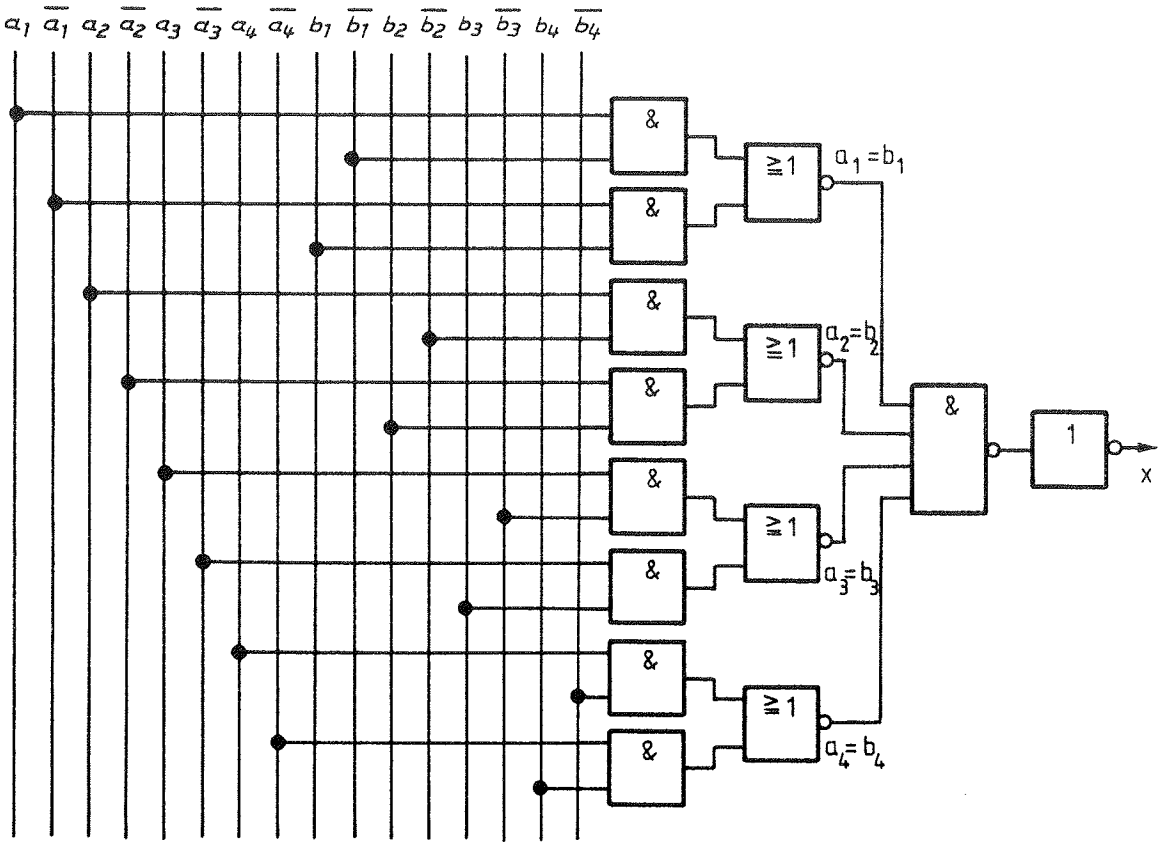


Exercise 9 - 3:

Build a 4-bit-comparator (equivalence circuit) for two inputs a and b.

Solution 9 - 3:

With the aid of equivalence circuits it can be observed whether or not two binary values a and b are equal. For an output it is necessary that the individual code places are equal. For the comparison of each code place the equivalence-gate is used. The circuit supplies a "1" at the output x if all code places are equal. For each of the code places an equivalence-gate as described on page 158 has to be present.



9.2 Threshold circuit

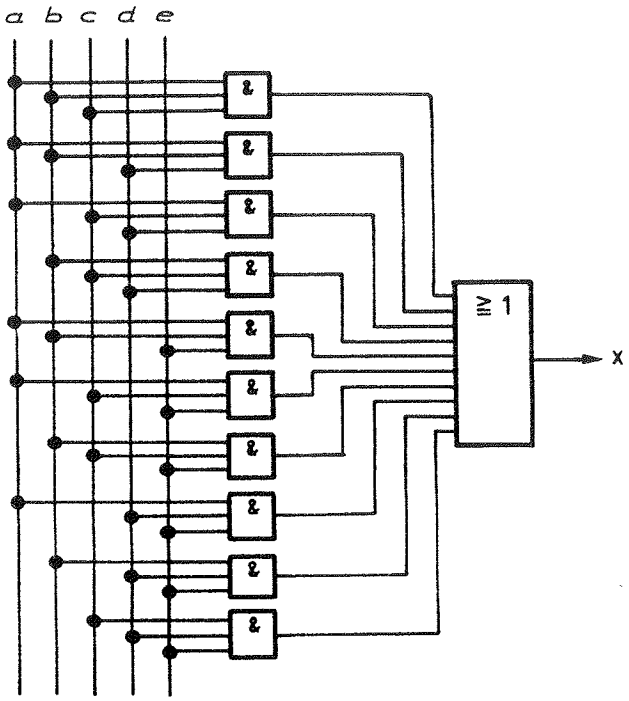
A threshold circuit connects through at the output (x = 1) when a "1" potential is applied to at least a certain number of inputs. This minimum number is called "threshold" of the circuit.

Exercise 9 - 4:

From the inputs a, b, c, d, e at least 3 must have the value "1", so that at the output at x a "1" appears.

Solution 9 - 4:

It is sufficient that all possible three-combinations of inputs are combined by AND-gates and leading to an OR-gate.



a	b	c	d	e	x
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

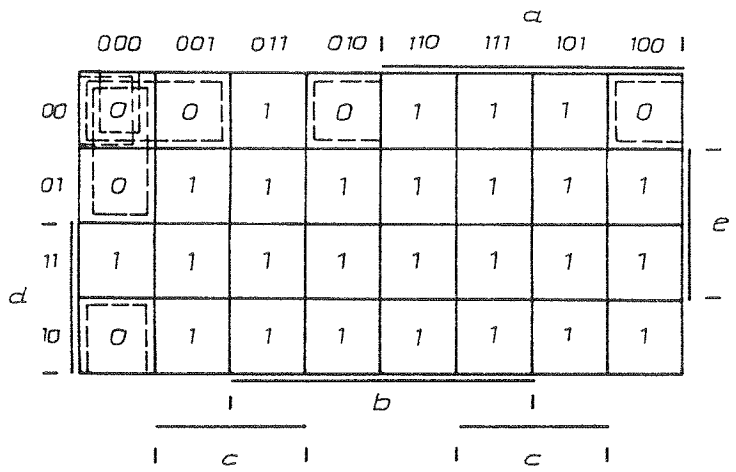
Exercise 9 - 5:

Design a threshold circuit in which the output x is a "1" if from the inputs a, b, c, d and e at least two have a "1"-potential.

Solution 9 - 5:

The circuit may only have a "0" at the output x if only one or no input is at "1". At threshold 2 the possible combinations to four (n - 1) inputs can be combined by OR-gates and put on an AND-gate, as shown on the diagram.

Karnaugh-diagram



a	b	c	d	e	x
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

As only 6 combinations yield to x = 0, but 28 x = 1, it is meaningful to make the set up with the negated form of the output variables.

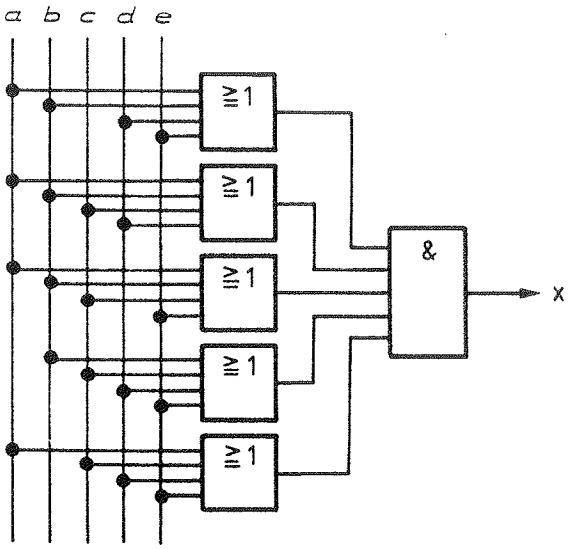
$$\bar{x} = \bar{a}\,\bar{b}\,\bar{d}\,\bar{e} + \bar{a}\,\bar{b}\,\bar{c}\,\bar{d} + \bar{a}\,\bar{b}\,\bar{c}\,\bar{e} + \bar{b}\,\bar{c}\,\bar{d}\,\bar{e} + \bar{a}\,\bar{c}\,\bar{d}\,\bar{e}$$

Conversion with the aid of the Morgan's rule:

$$x = \bar{\bar{x}} = \overline{\bar{a}\,\bar{b}\,\bar{d}\,\bar{e} + \bar{a}\,\bar{b}\,\bar{c}\,\bar{d} + \bar{a}\,\bar{b}\,\bar{c}\,\bar{e} + \bar{b}\,\bar{c}\,\bar{d}\,\bar{e} + \bar{a}\,\bar{c}\,\bar{d}\,\bar{e}}$$

$$\boxed{x = (a+b+d+e)\,(a+b+c+d)\,(a+b+c+e)\,(b+c+d+e)\,(a+c+d+e)}$$

Circuit:



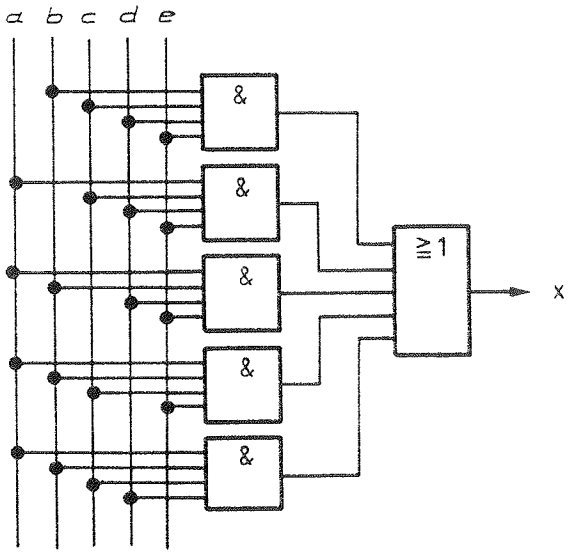
Exercise 9 - 6:

Design a circuit, reading at the output x value "1" if at least four of the five inputs a, b, c, d and e have the value "1".

Solution 9 - 6:

All four-combinations of inputs are operated by AND and combined by an OR-gate. The four-combinations are (refer table page 165):

x = b c d e + a c d e + a b d e + a b c e + a b c d



9.3 Selection circuit "2 out of n"

Exercise 9 - 7:

Design an operation-logic to find out if from n = 4 inputs

a) 2 inputs show state one
(recognition output x_1 = 1)

b) 2 or more inputs show state one
(recognition output x_2 = 1)

Solution 9 - 7:

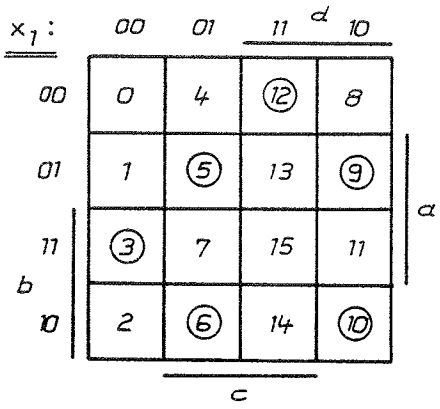
Table:

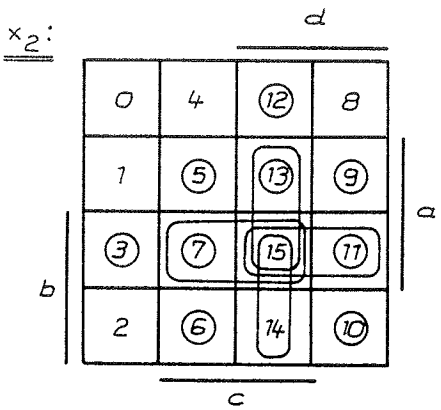
Dec.	d	c	b	a	x_1	x_2
0	0	0	0	0	0	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	1	1
4	0	1	0	0	0	0
5	0	1	0	1	1	1
6	0	1	1	0	1	1
7	0	1	1	1	0	1
8	1	0	0	0	0	0
9	1	0	0	1	1	1
10	1	0	1	0	1	1
11	1	0	1	1	0	1
12	1	1	0	0	1	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	0	1

$x_1 = 3 + 5 + 6 + 9 + 10 + 12$

$x_2 = \underbrace{3 + 5 + 6 + 9 + 10 + 12}_{x_1} + 7 + 11 + 13 + 14 + 15$

Karnaugh-diagram:

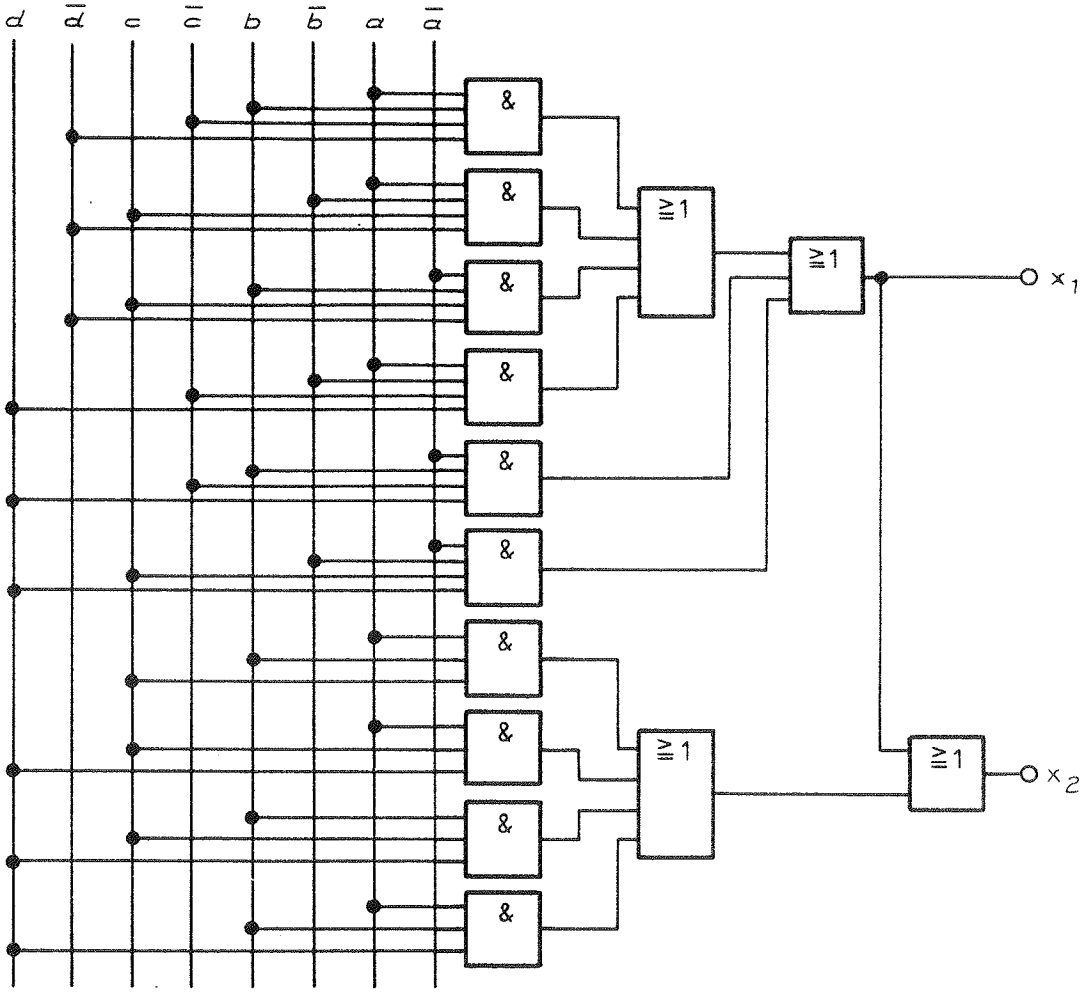




There is no simplification possible for x_1:

$x_1 = a\,b\,\bar{c}\,\bar{d} + a\,\bar{b}\,c\,\bar{d} + \bar{a}\,b\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d + \bar{a}\,b\,\bar{c}\,d + \bar{a}\,\bar{b}\,c\,d$
$x_2 = x_1 + a\,b\,c + a\,c\,d + b\,c\,d + a\,b\,d$

Circuit:



10. ERROR DETECTION AND ERROR CORRECTION

Exercise 10 - 1:

Develop a circuit for testing the (2 out of 5)-code.

Solution 10 - 1:

From this 5 bit code 2 are always used. From the possible 32 combinations ($2^5 = 32$) thus only 10 are used.

$\left[\binom{5}{2} = \frac{5 \cdot 4}{1 \cdot 2} = 10 \right]$ 22 combinations are redundant.

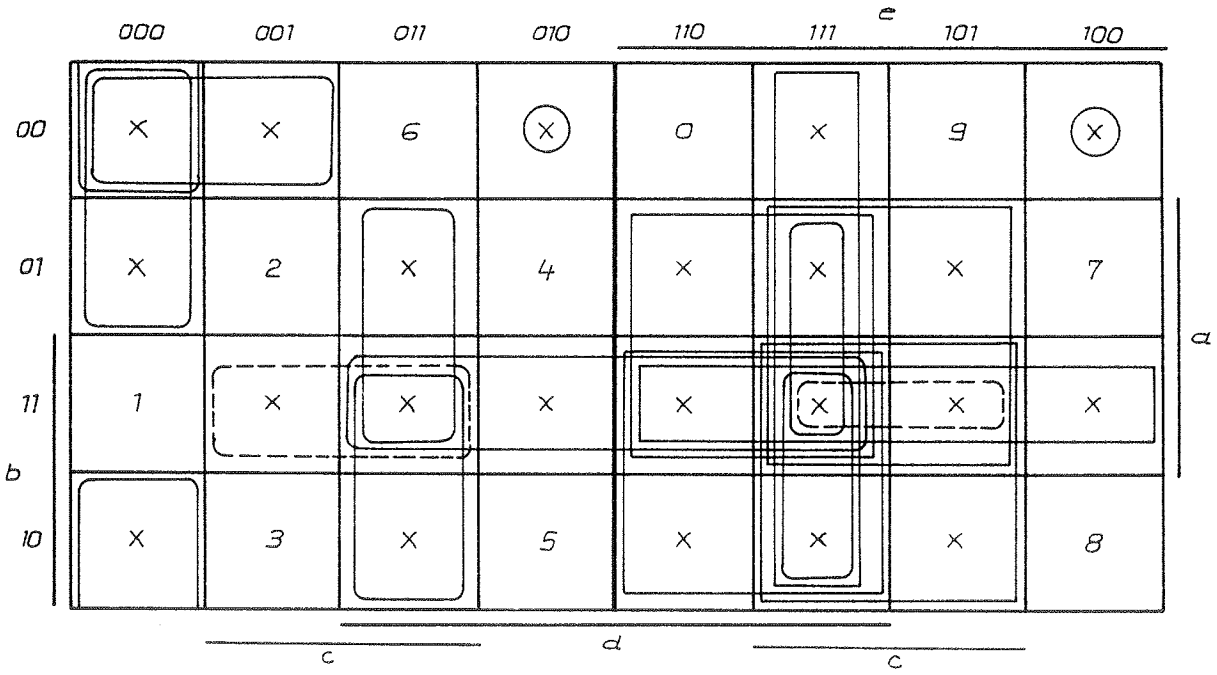
Code table:

Dec.	e	d	c	b	a
0	1	1	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0

The combinations from 10 to 31 are incorrect and should be indicated by the output $x_f = 1$.

Karnaugh-diagram:

Errors with $x_f = 1$ are marked with (X).



$$\overline{x_f} = edc + dba + eba + eda + eca + edb + ecb + cba + dcb + dca + \overline{edba} + \overline{edcb} + \overline{edca} + \overline{edcba} + \overline{edcba}$$

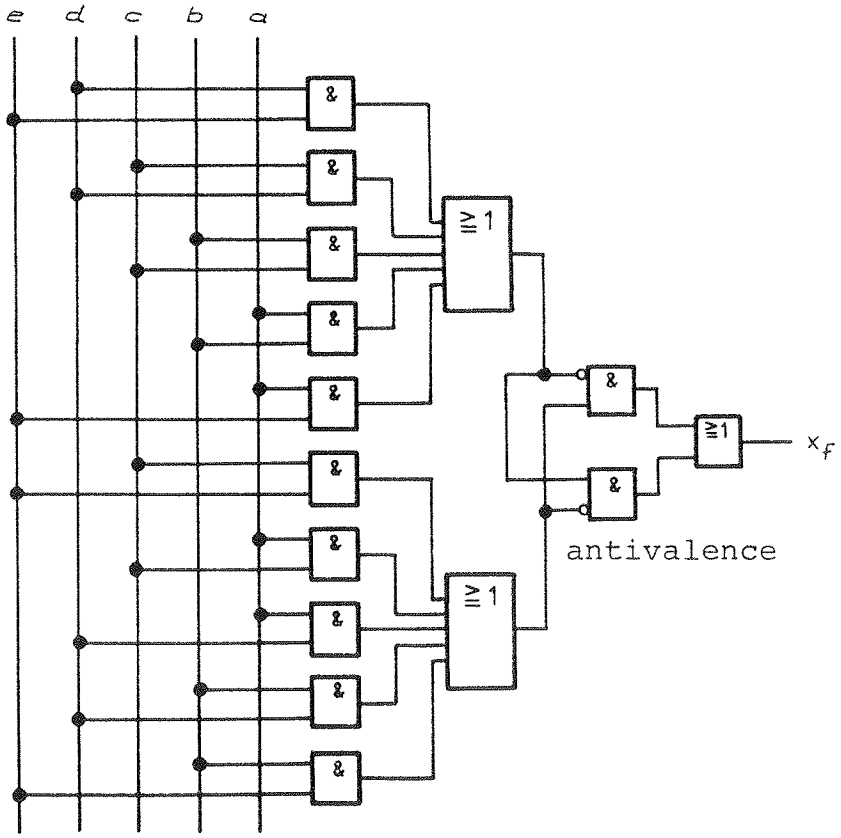
The expression is very extensive, but it can be realized by the digital teaching model.

After a longer translation the Boolean expression for x_f can be simplified as follows:

$$\overline{x_f} = (de + cd + bc + ab + ae) \ddagger (ce + ac + ad + bd + be)$$

($\ddagger$ antivalence)

Circuit:



Exercise 10 - 2:

Specify a recognition circuit of the pseudo tetrads of the 8421-BCD-Code.

Solution 10 - 2:

The 8421-BCD-Code uses only the combinations from 0 to 9 from the 16 possible combinations. The remaining 6 states from 10 to 15 do not have to appear and can be used for error detection.

Dec. system	d	c	b	a	x_f
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

The remaining not shown value combinations are "0" in the table.

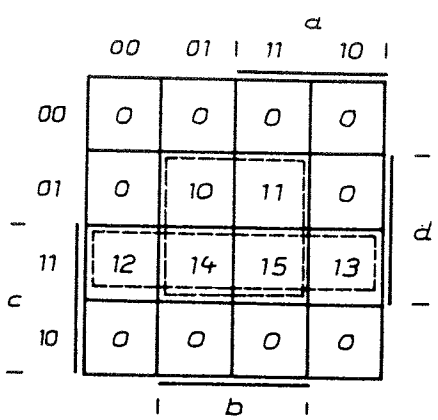
$x_f = \bar{a}b\bar{c}d + ab\bar{c}d + \bar{a}\bar{b}cd + a\bar{b}cd + \bar{a}bcd + abcd$

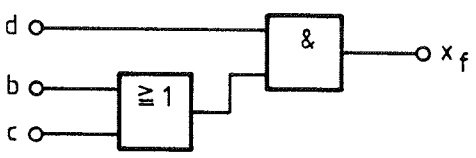
Simplification by factoring out:

$x_f = b\bar{c}d + \bar{b}cd + bcd$

$x_f = bd + cd = d\ (b + c)$

The same result is achieved with the aid of the Karnaugh-Diagram.

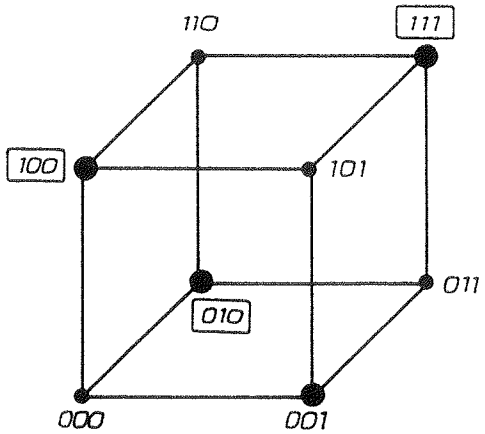




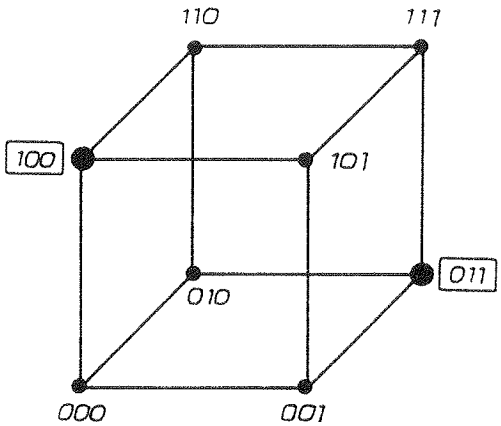
10.1 One error-detection- and correction circuit for code words of the Hamming-Code

When transferring binary data the capability of a received code word being tested and possibly corrected is indispensable. For this, special codes are needed which must have a redundancy i.e. not all possible combinations are used to form a code. They consist of effective words and pseudo words (test-words).

One of these codes is the "Systematic or Hamming-Code". With this code any two useful words (information carrying words) are different by a constant minimum number of bits d. d is also named Minimum-Hamming-Distance. The principle is made clear by a space diagram of a 3 bit code.



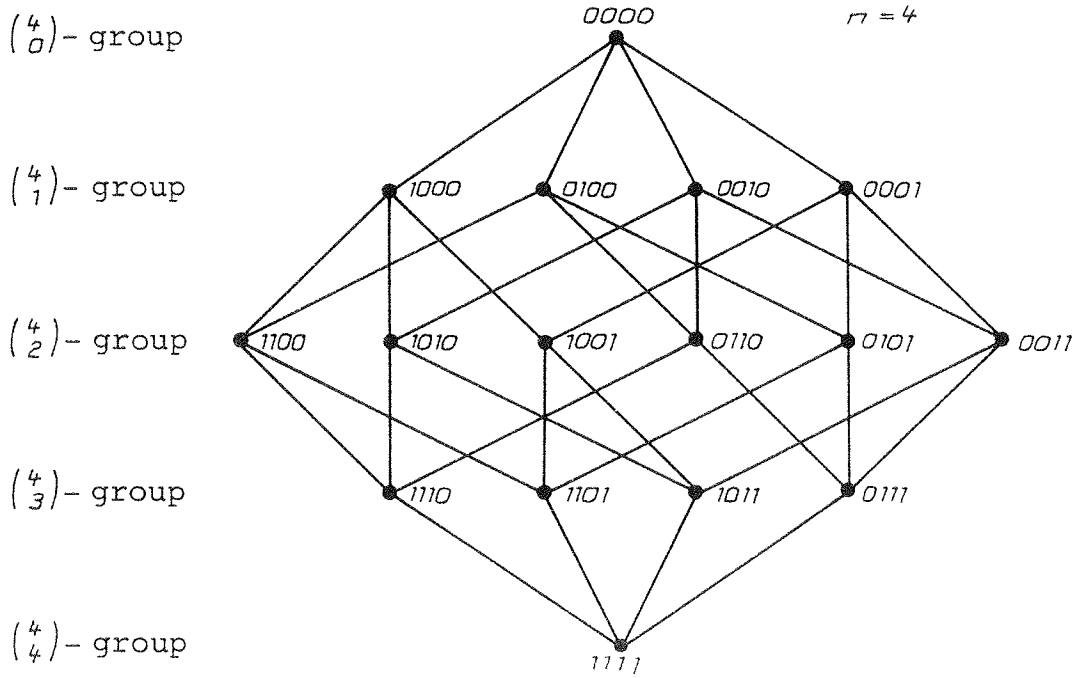
error detectable code
d = 2



error correctable code
d = 3

d also means the number of cube edges which are between one code word (= cube corner) and another code word. The codes used here are arbitrarily (010, 100, 111 and 011, 100). Of course, other combinations can be used.

If the number of bits increase, the geometric body for the code-space becomes more complicated. As an example the code-space for a 4-placed code is mentioned.



For the Hamming-Code the following is valid:

Number of recognizable errors:

$$N_e = d - 1$$

Number of correctable errors:
for an odd least-Hamming-distance d

$$N_k = \frac{d - 1}{2}$$

for an even least-Hamming distance d

$$N_k = \frac{d}{2} - 1$$

Exercise 10 - 3:

Develop a one-error-detection circuit for 3 binary places. The code of the net-words has to be built up in such a way that the cross-sum is always even (even parity).

Solution 10 - 3: (see also page 174)

Correct code words

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

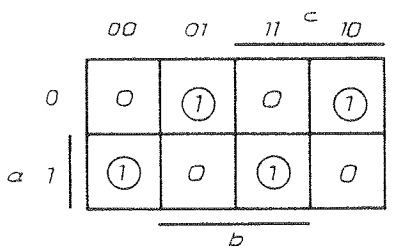
code words with one error

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

Table:

c	b	a	x_f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

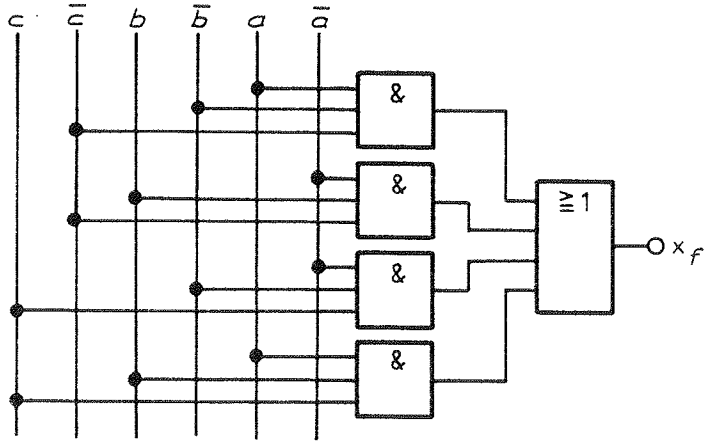
Karnaugh-diagram:



$x_f \hat{=}$ faulty code word

Equation (no simplification is possible):

$$x_f = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$



The Karnaugh-diagram shows, that no blocks can be constructed, because the neighbouring fields differ by one variable only, but for code words in the Hamming-Code with d = 2 the code words always differ by 2 variables.

Exercise 10 - 4:

Develop a one-error-correction circuit for 3 binary places. The effective words should be (refer to page 174): 0 1 1, 1 0 0

Solution 10 - 4:

With one error the following code words can appear:

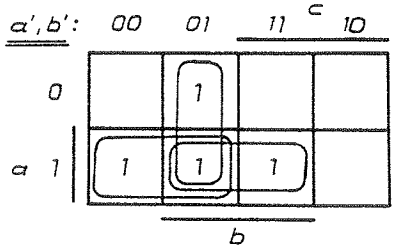
	c	b	a	c'	b'	a'	
correct	0	1	1	0	1	1	
incorrect	0	0	1	0	1	1	corrected
incorrect	0	1	0	0	1	1	corrected
incorrect	1	1	1	0	1	1	corrected
correct	1	0	0	1	0	0	
incorrect	0	0	0	1	0	0	corrected
incorrect	1	0	1	1	0	0	corrected
incorrect	1	1	0	1	0	0	corrected

Equations: $a' = \bar{c}ba + \bar{c}\bar{b}a + \bar{c}b\bar{a} + cba$

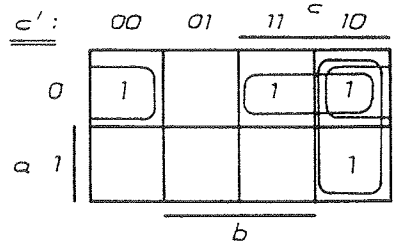
$b' = a'$

$c' = c\bar{b}\bar{a} + \bar{c}\bar{b}\bar{a} + c\bar{b}a + cb\bar{a}$

Karnaugh-diagram:



$$a' = b' = b\bar{c} + a\bar{c} + ab$$



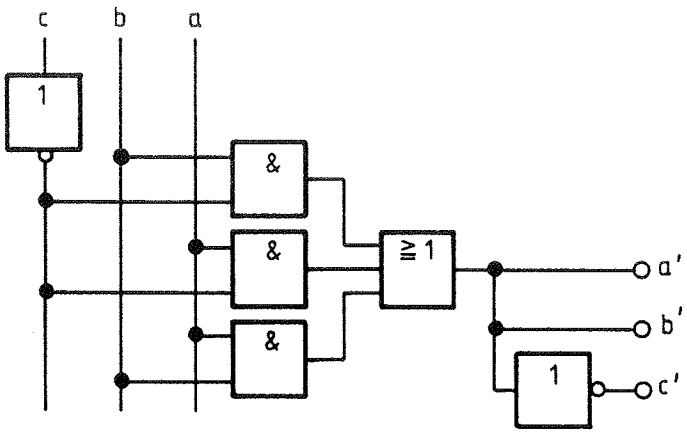
$$c' = \bar{a}c + \bar{a}\bar{b} + \bar{b}c$$

or

$$c' = \bar{a}' = \bar{b}'$$

(from table or Karnaugh-diagram)

Circuit:



11. MODULO N-CONVERTER

11.1 General

Mathematical: $\boxed{a = b \cdot \text{mod } n}$ (calculating rule) is valid

or expressed differently:

$$a = b + m \cdot n$$

whereby m is an integer number which is received when a is divided by n.

$$m = \left[\frac{a}{n}\right]$$

n is the highest appearing number value
a is the number from which the modulo is to be formed
b is the difference from a and the integral multiple of n

Example:

When calculating with trigometric functions for the determination of the function value of an angle from over 360 degree only the difference to 360 degree or 720 degree, 1080 degree is needed (m · 360 degree).

e.g. $\alpha = 1563°$ ($\hat{=}$ a)
$n = 360°$
$b = 1563° - 4 \cdot 360° = 1563° - 1440°$
$b = 123°$

therefore

$$1563° = 123° \text{ mod } 360°$$

With digital numbers a counter can only count to its top number. Then the counter starts again from state zero. If e.g. 37 pulses are given to a 10-counter, then the counter counts through 3 times and stops on value 7. Thus:

$$37 = 7 \text{ mod } 10 \quad (a = 37;\ b = 7;\ n = 10)$$

Assignment table for modulo n-converter for numbers in binary code (3 bit places)

	c	b	a	c'	b'	a'	c'	b'	a'	c'	b'	a'	c'	b'	a'
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
2	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0
3	0	1	1	0	0	1	0	0	0	0	1	1	0	1	1
4	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
5	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0
6	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1
7	1	1	1	0	0	1	0	0	1	0	1	1	0	1	0
0	0	0	0												
	⋮			⋮			⋮			⋮			⋮		
				mod 2			mod 3			mod 4			mod 5		

etc.

Exercise 11 - 1:

State the circuit of a modulo 2-converter for 3 binary places.

Solution 11 - 1:

Take from table:

$$a' = a\bar{b}\bar{c} + ab\bar{c} + a\bar{b}c + abc$$

$$a' = a\bar{b}\,\underbrace{(\bar{c} + c)}_{1} + ab\,\underbrace{(\bar{c} + c)}_{1}$$

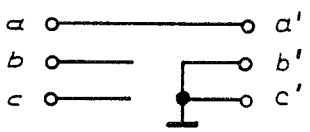
$$a' = a\bar{b} + ab = a\,(\bar{b} + b) = a$$

$$\boxed{a' = a}$$

This result can be read directly from the table; further:

$$\boxed{c = 0;\ b = 0}$$

Circuit:



Exercise 11 - 2:

Develop a converter modulo 3 for 3 binary coded inputs.

Solution 11 - 2:

From table page 180 follows:

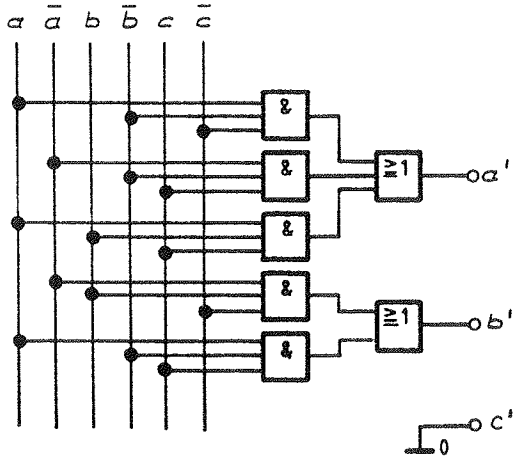
$a' = a\bar{b}\bar{c} + \bar{a}\bar{b}c + abc$ (cannot be simplified)

$b' = \bar{a}b\bar{c} + a\bar{b}c$ (cannot be simplified)

$c' = 0$

This solution is valid with 3 inputs a, b, c only i.e. for 8 possible combinations.

Circuit converter modulo 3:



Exercise 11 - 3:

A logic operation is wanted for a converter modulo 4 for 3 inputs in the binary code.

Solution 11 - 3: (refer page 180)

$a' = a\bar{b}\bar{c} + ab\bar{c} + a\bar{b}c + abc$

$a' = a$ (as with modulo 2)

$b' = \bar{a}b\bar{c} + ab\bar{c} + \bar{a}bc + abc$

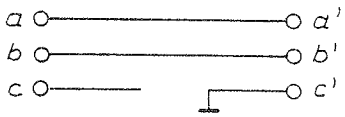
$b' = \bar{a}b\underbrace{(\bar{c} + c)}_{1} + ab\underbrace{(\bar{c} + c)}_{1}$

$b' = b\,(a + \bar{a})$

$b' = b$

$c' = 0$

Circuit:

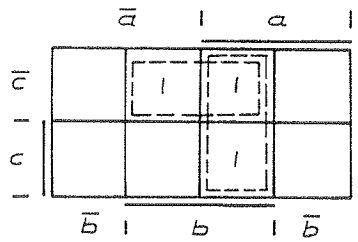


Exercise 11 - 4:

State a circuit of a converter modulo 5 for 3 bit places of the binary code.

Solution 11 - 4:

Table page 180 yields:



$a' = a\bar{b}\bar{c} + ab\bar{c} + \bar{a}bc$

$a' = a\bar{c}\ (\bar{b} + b) + \bar{a}bc$

$a' = a\bar{c} + \bar{a}bc$

$b' = \bar{a}b\bar{c} + ab\bar{c} + abc$

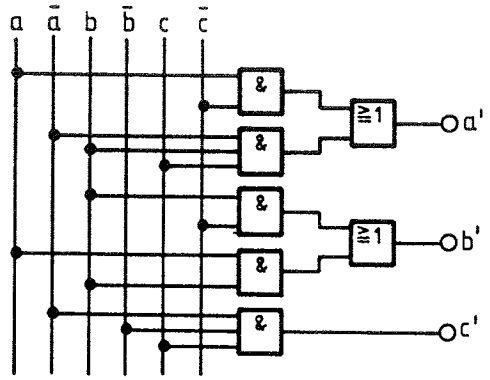
$b' = \bar{a}b\bar{c} + ab$

$b' = b\bar{c} + ab$

$c' = \bar{a}\bar{b}c$

This solution is valid likewise with 8 possible input combinations a, b, c only.

Circuit:



12. DIGITAL SWITCHES

Exercise 12 - 1:

Develop a circuit by which the illumination of a staircase from 3 places independently from each other can be switched on and off. (Digital circuit for staircase illumination)

Solution 12 - 1:

3 switches $\hat{=}$ 3 variables: a, b, c (per floor a switch and a lamp)

a $\hat{=}$ 1. floor
b $\hat{=}$ 2. floor
c $\hat{=}$ 3. floor

Assignment: switch closed $\hat{=}$ 1
switch open $\hat{=}$ 0

Switching function for the lamps: f

light on: f = 1
light off: f = 0

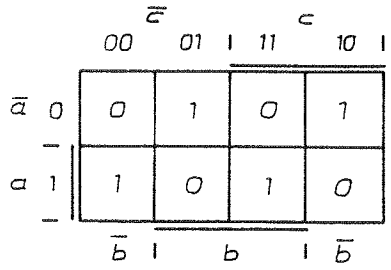
Function table:

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Equation

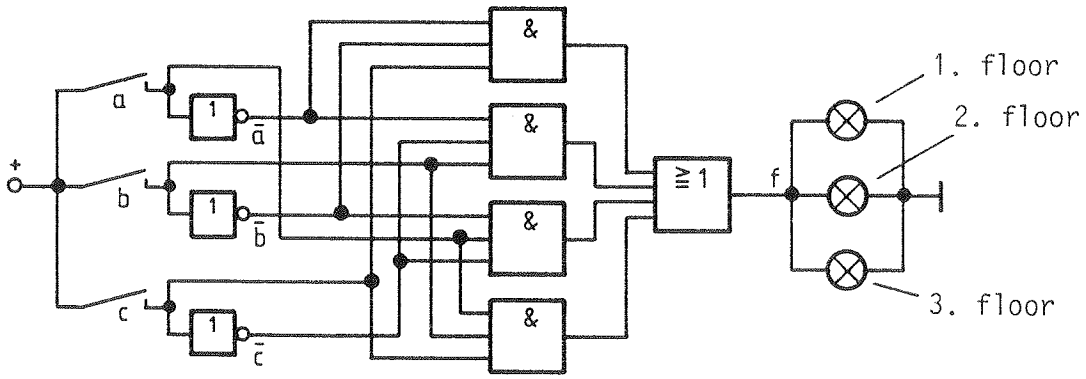
$$f = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

Karnaugh-diagram



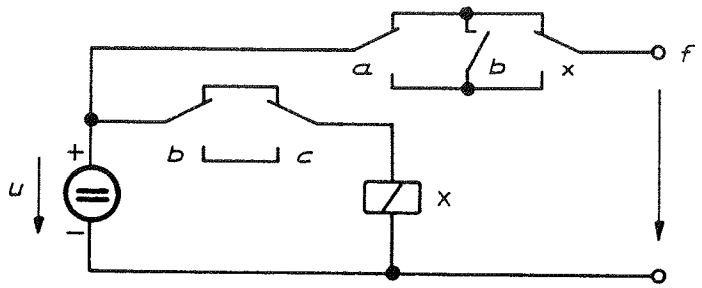
From the Karnaugh-diagram it is seen that further simplification is not possible.

Circuit:



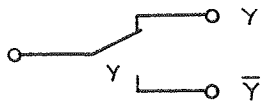
Exercise 12 - 2:

The circuit in relay-technique is to be realized by a semiconductor circuit consisting of NAND-gates.



Solution 12 - 2:

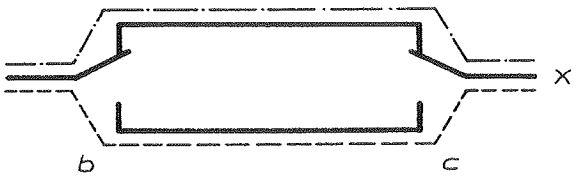
First a definition of contact positions must be made.
(y = general variables)



switch up: normal y
switch down: inverted y

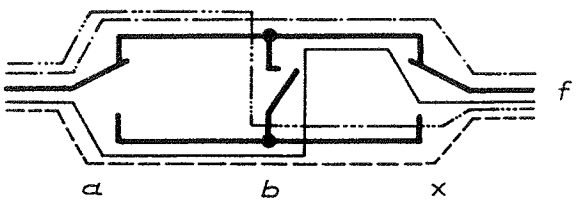
At a contact net work every way that leads to a through-switching has to be considered. Then all ways have to be combined with each other through an OR-operation.

Intermediate variable x:



$x = bc + \overline{b}\overline{c}$

Output variable f:



} possible through-switching ways

$f = ab\bar{x} + ax + \bar{a}\bar{x} + \bar{a}bx$

$= \bar{x}\,(ab + \bar{a}) + x\,(a + \bar{a}b)$

Now the variable x or $\bar{x}$ has to be used. Evaluation of $\bar{x}$:

$x = bc + \bar{b}\bar{c}$

$\bar{x} = \overline{bc + \bar{b}\bar{c}}$

$= \overline{bc} \cdot \overline{\bar{b}\bar{c}}$

$= (\bar{b} + \bar{c}) \cdot (b + c)$

$= \underbrace{\bar{b} \cdot b}_{0} + \bar{b} \cdot c + \bar{c} \cdot b + \underbrace{\bar{c} \cdot c}_{0}$

thus:

$\bar{x} = \bar{b}c + \bar{c}b$

$f = (\bar{b}c + \bar{c}b)\ \underbrace{(ab + \bar{a})}_{\bar{a} + b} + (bc + \bar{b}\bar{c})\ \underbrace{(a + \bar{a}b)}_{a + b}$ (see page 37 10 b and 11 b)

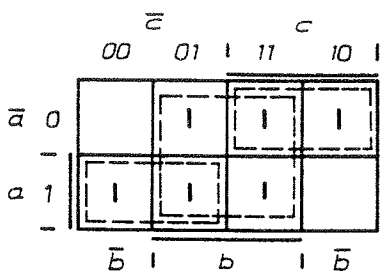
factoring out:

$f = \bar{b}c \cdot \bar{a} + \bar{b}cb + \bar{c}b\bar{a} + \bar{c}bb + bca + bcb + \bar{b}\bar{c}a + \bar{b}\bar{c}b$

with $b \cdot b = b$ and $\bar{b} \cdot b = 0$ follows:

$f = \bar{a}\bar{b}c + \bar{a}b\bar{c} + b\bar{c} + abc + bc + a\bar{b}\bar{c}$

This expression is to be simplified by means of the Karnaugh-diagram.



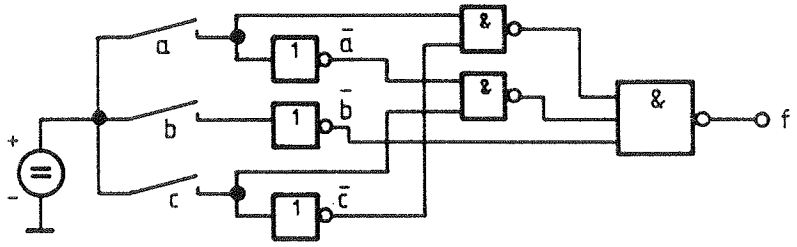
$f = b + a\bar{c} + \bar{a}c$

in NAND-techniques

$f = \overline{\overline{b} + \overline{a\bar{c}} + \overline{\bar{a}c}}$

$f = \overline{\bar{b} \cdot \overline{a\bar{c}} \cdot \overline{\bar{a}c}}$

Circuit:



Exercise 12 - 3:

Develop a simple one-pole digital on-off switch.

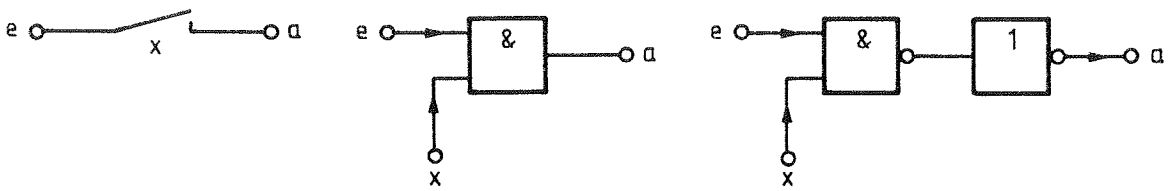
Solution 12 - 3:

The through-switching is done with the control signal x:

x = 0 switch open, output 0
x = 1 switch closed,
output variable a = input variable e

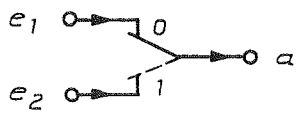
This realizes an AND-gate

$a = x \cdot e$ or in NAND-techniques $a = \overline{\overline{x \cdot e}}$



Exercise 12 - 4:

Develop a one-pole digital reverse contact or information switch.



Solution 12 - 4:

Control signal: x = 0 (rest position) a = e_1

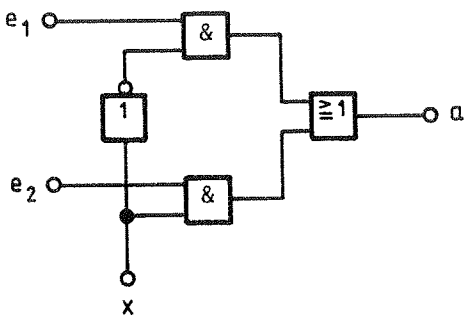
x = 1 a = e_2

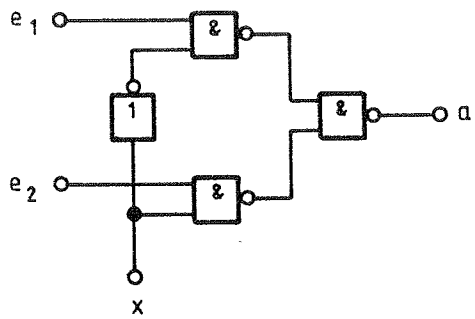
$$a = \bar{x} \cdot e_1 + x \cdot e_2$$

or in NAND-techniques

$$a = \overline{\overline{\bar{x} \cdot e_1} + \overline{x \cdot e_2}} = \overline{\overline{\bar{x} \cdot e_1} \cdot \overline{x \cdot e_2}}$$

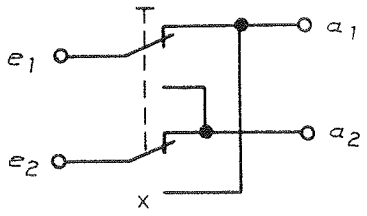
Circuit:





Exercise 12 - 5:

Develop a digital pole reversal switch.



Solution 12 - 5:

Here also for the switch-over of the digital switch the control signal x is used.

x = 0 : switch up

$a_1 = e_1$; $a_2 = e_2$

x = 1 : switch down

$a_1 = e_2$; $a_2 = e_1$

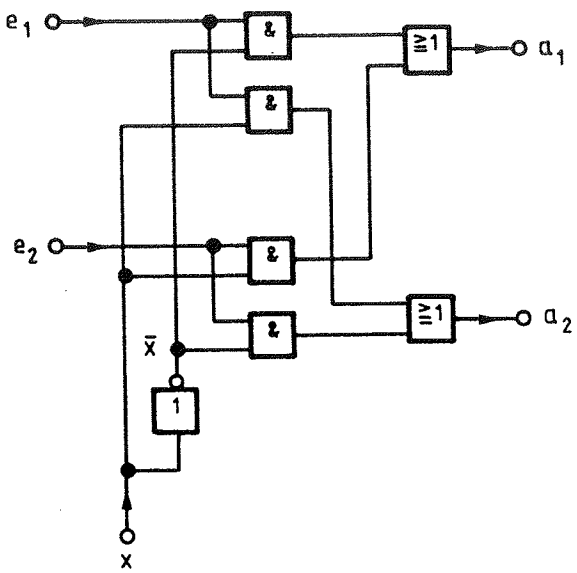
Equations:

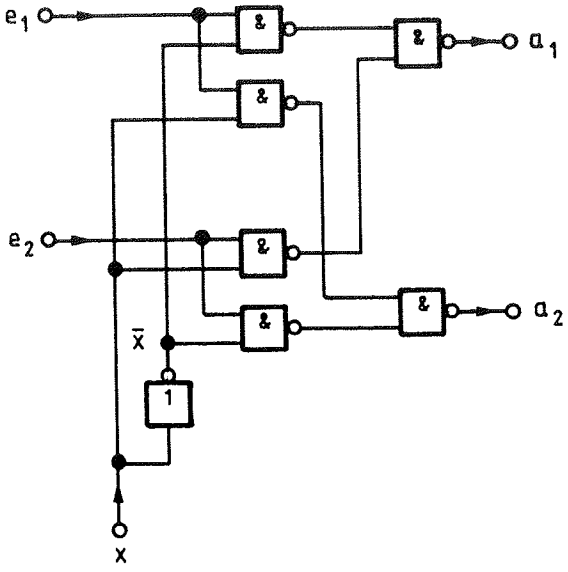
$$a_1 = \bar{x} \cdot e_1 + x \cdot e_2$$
$$a_2 = \bar{x} \cdot e_2 + x \cdot e_1$$

or

$$a_1 = \overline{(\overline{\bar{x} \cdot e_1}) \cdot (\overline{x \cdot e_2})}$$
$$a_2 = \overline{(\overline{\bar{x} \cdot e_2}) \cdot (\overline{x \cdot e_1})}$$

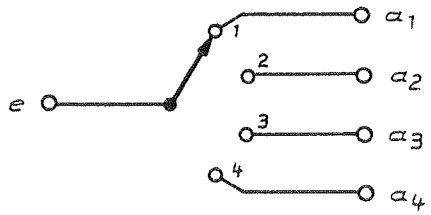
NAND-techniques





Exercise 12 - 6:

Develop a digital rotary-switch with 4 switch positions where the distribution of an information e, at the input, should be passed alternatively to 4 outputs a_1, a_2, a_3 a_4 (demultiplexer).



Solution 12 - 6:

The 4 switch positions have to be defined by two control signals x, y.

e.g.

x	y	position
0	0	1
1	0	2
0	1	3
1	1	4

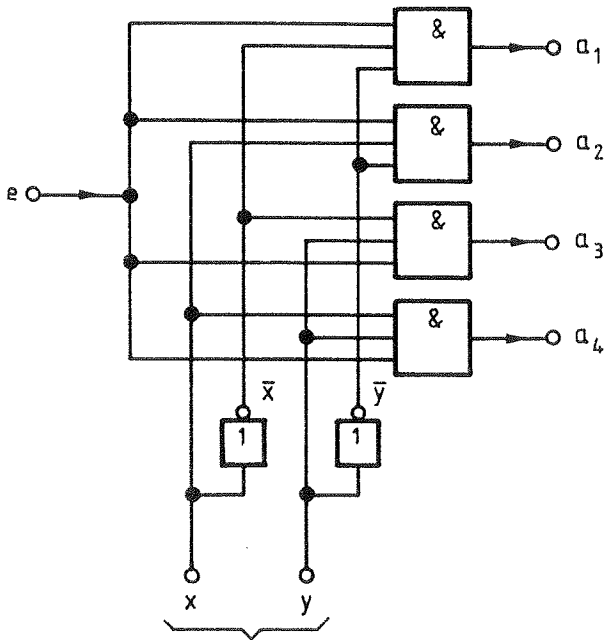
The logic equations are:

$$a_1 = \bar{x}\bar{y} \cdot e$$

$$a_2 = x\bar{y} \cdot e$$

$$a_3 = \bar{x}y \cdot e$$

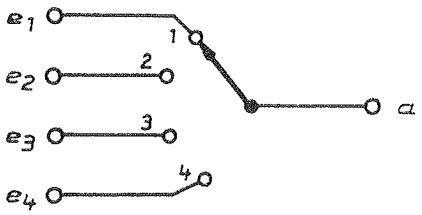
$$a_4 = xy \cdot e$$



Data selection inputs

Exercise 12 - 7:

Develop a digital rotary-switch with 4 switch positions where 4 information inputs e_1, e_2, e_3, e_4 are passed, as required, to one output, a. (Data selector, Multiplexer)

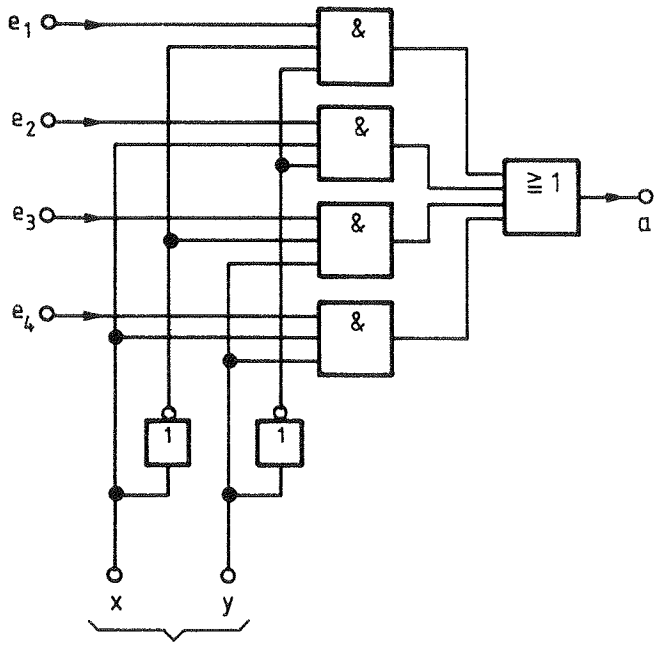


Solution 12 - 7:

Here also the 4 switch positions have to be defined by the control signals x and y (page 190).

The logic equation is:

$$a = \bar{x}\bar{y} \cdot e_1 + x\bar{y} \cdot e_2 + \bar{x}y \cdot e_3 + xy \cdot e_4$$

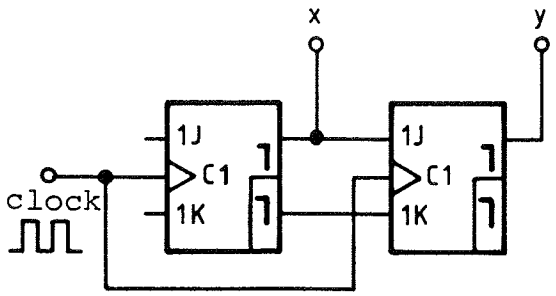


Data selection inputs

If the 4 switch positions are selected successively, this can be done with a counter which counts up to 4. As normally such digital rotary switches do not work independently from other switching instructions in a larger circuit, it is meaningful to use a clock controlled (synchronous) counter.

x	y	Clock
0	0	1
1	0	2
0	1	3
1	1	4
0	0	5

The counter, that runs cyclically through these states is called a modulo 4-counter.



Exercise 12 - 8:

Sketch the principle of a digital time-multiplex-system with the aid of a 4-bit-multiplexer and a 4-bit-demultiplexer.

Solution 12 - 8:

With both digital switches, a time-multiplex-system for transfer of 4 information sources through a channel, can be realized. In the sender as well as in the receiver are synchronous controlled modulo 4-counters, which are switched by a central clock.