6. Simplification of logical functions

Logical functions may consist of many variables and may be quite complex. We may try to simplify functions on the basis of the postulates etc, but this is in many cases not an easy task. For this reason there exist systematic approaches, even CAD (computer aided design) programs. We will discuss only one method here. Arranging min (max) terms on a square lattice in a proper way, we obtain a so-called **Karnaugh diagram**:

	$\bar{c}$	$\Downarrow$	c		
	$\bar{a}\bar{b}\bar{c}\bar{d}$ 0	$\bar{a}\bar{b}\bar{c}d$ 1	$\bar{a}\bar{b}cd$ 3	$\bar{a}\bar{b}c\bar{d}$ 2	$\bar{b}$
$\bar{a}$	$\bar{a}b\bar{c}\bar{d}$ 4	$\bar{a}b\bar{c}d$ 5	$\bar{a}bcd$ 7	$\bar{a}bc\bar{d}$ 6	$\Leftarrow$
$\Rightarrow$	$ab\bar{c}\bar{d}$ 12	$ab\bar{c}d$ 13	$abcd$ 15	$abc\bar{d}$ 14	b
a	$a\bar{b}\bar{c}\bar{d}$ 8	$a\bar{b}\bar{c}d$ 9	$a\bar{b}cd$ 11	$a\bar{b}c\bar{d}$ 10	$\Leftarrow$ $\bar{b}$
	$\bar{d}$ $\Uparrow$	d	$\Uparrow$	$\bar{d}$	

We make the following observations:

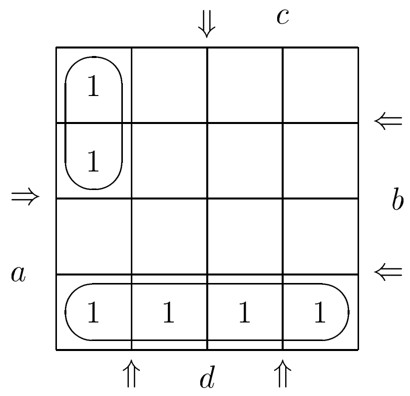
- Going up, down, left, or right, only a single variable changes.
- Horizontally, the variables cd follow the values 00, 01, 11, 10.
- Vertically, the variables ab follow the values 00, 01, 11, 10.
- The single–variable change is also *cyclic* around the edges. $\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}$
- We can consider the variables $abcd$, with a being the most significant bit, to represent a 16–counter $0, 1, 2, \cdots, 15$, therefore representing all the possible min terms.
- A function is represented by placing for each min term a 1 in the corresponding box.
- When two min terms occur in horizontally or vertically neighbouring boxes, a variable can be eliminated.

The Karnaugh diagram allows an easy simplification by visual inspection. By taking the **largest** blocks of 1's, most variables can be eliminated.

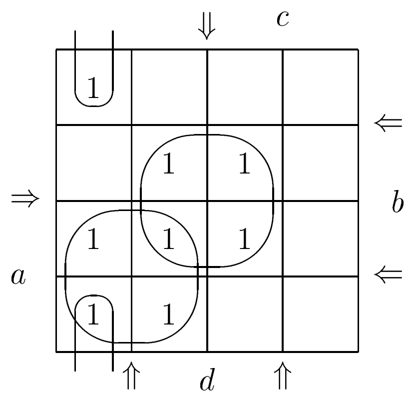
Example:
For the two blocks 5 and 7 in the centre we have $\bar{a}b\bar{c}d + \bar{a}bcd$. Only the c term differs, so $m_5 + m_7 = \bar{a}bd$. The term c has been eliminated.

Example:
$\sum 5, 7, 13, 15 = \bar{a}b\bar{c}d + \bar{a}bcd + ab\bar{c}d + abcd = bd$. The variables a and c are eliminated. Check this algebraically and in the Karnaugh diagram!

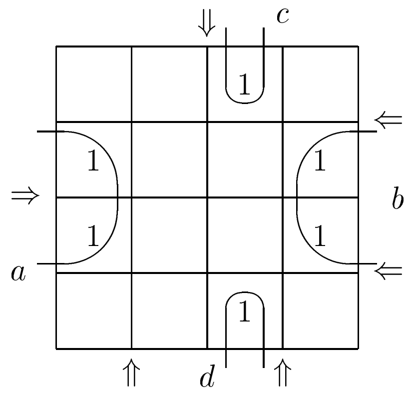
More examples:



$\sum 0, 4, 8, 9, 10, 11 = a\bar{b} + \bar{a}\bar{c}\bar{d}$.
A block of four 1's: 2 variables eliminated.
A block of two 1's: 1 variable eliminated.

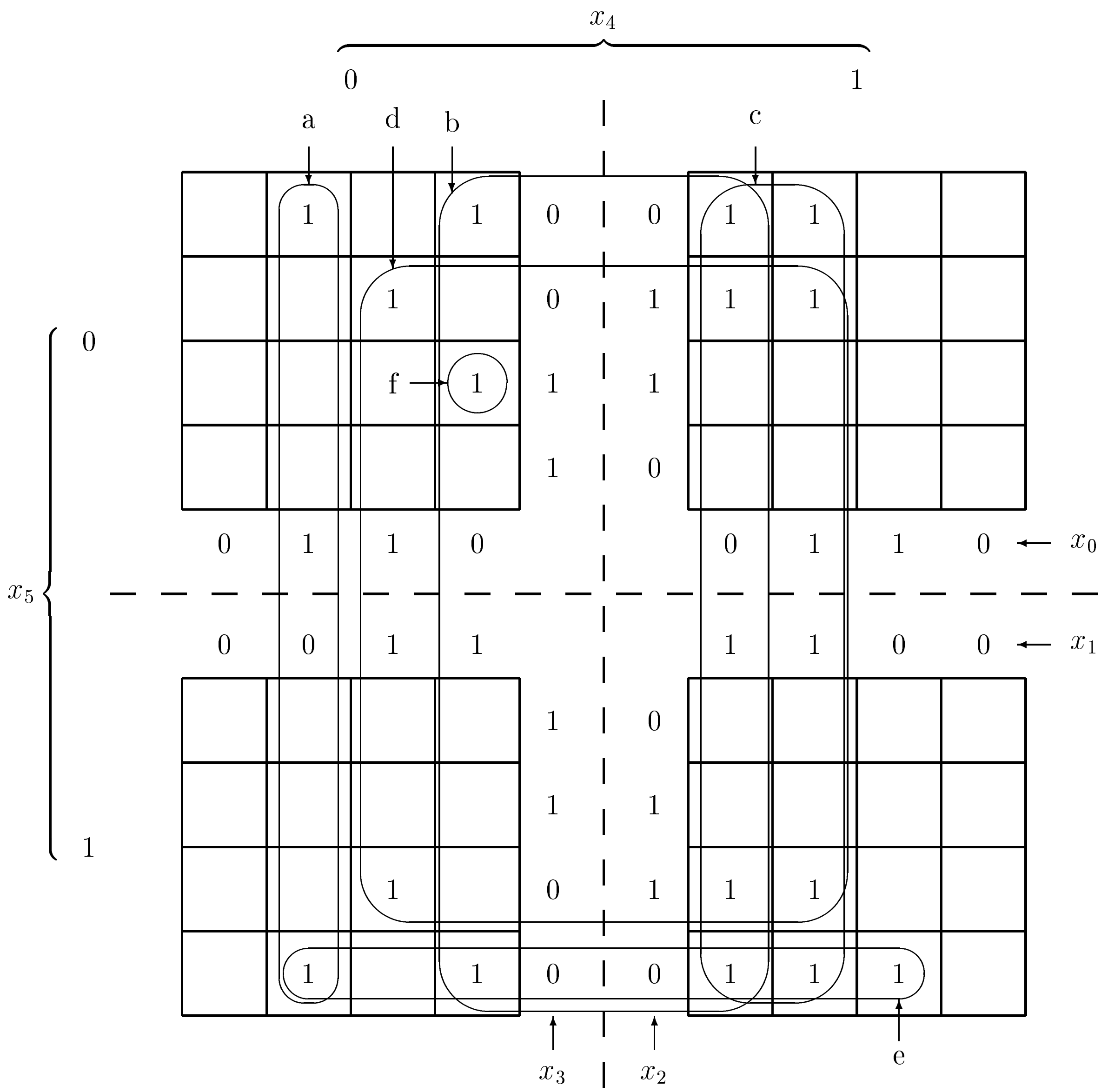


$\sum 0, 5, 7, 8, 9, 12, 13, 15 = a\bar{c} + bd + \bar{c}\bar{b}\bar{d}$.



$\sum 3, 4, 6, 11, 12, 14 = b\bar{d} + \bar{b}cd$.

As an example, a complex Karnaugh diagram with 6 variables is shown here.



After simplification, we get

$$a = \bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1x_0$$

$$b = \bar{x}_3\bar{x}_2x_1\bar{x}_0$$

$$c = x_4\bar{x}_3x_1$$

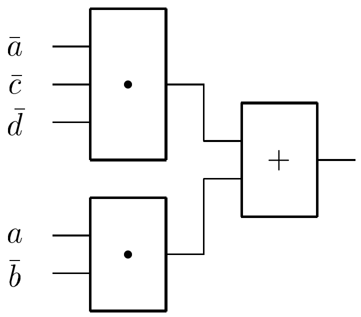
$$d = \bar{x}_3x_2x_1x_0$$

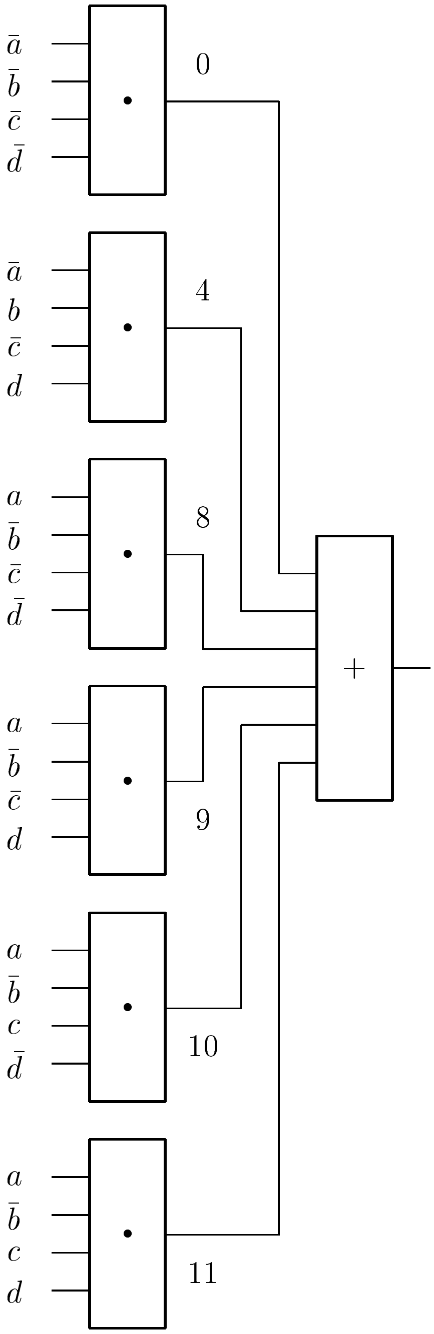
$$e = x_5\bar{x}_3\bar{x}_2\bar{x}_1x_0$$

$$f = \bar{x}_5\bar{x}_4x_3x_2x_1\bar{x}_0$$

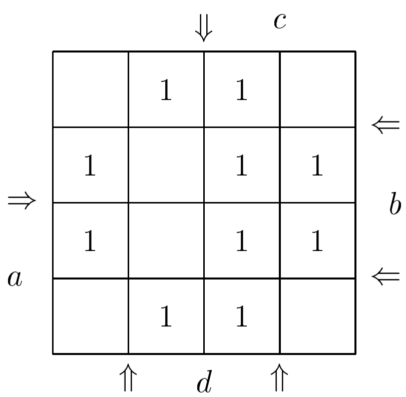
6.1 Technical realisations

The first example ($\sum 0, 4, 8, 9, 10, 11$) requires many elements, namely 6 ANDs with 4 inputs each, i.e. 24 inputs, plus one OR with 6 inputs. Total: 7 gates and 30 inputs. The equivalent function $a\bar{b} + \bar{a}\bar{c}\bar{d}$ requires only three gates and 7 inputs. These circuits are shown to the right and below.



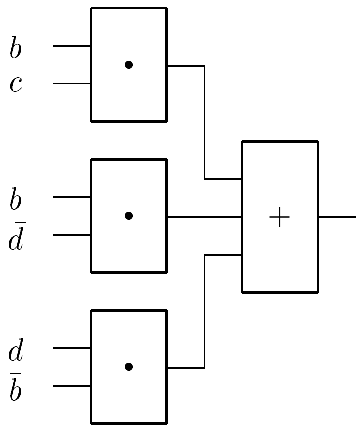


Let us use another example: $\sum(1, 3, 4, 6, 7, 9, 11, 12, 14, 15)$. Using the Karnaugh diagram

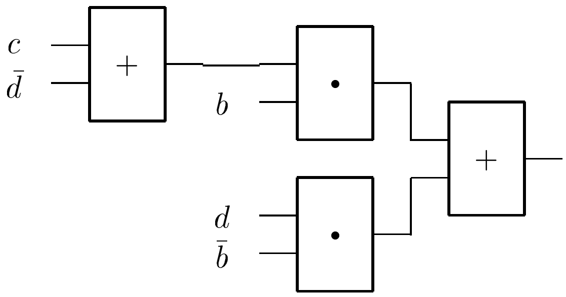


we can simplify this function to $bc + b\bar{d} + d\bar{b}$. This function can be realised in different ways.

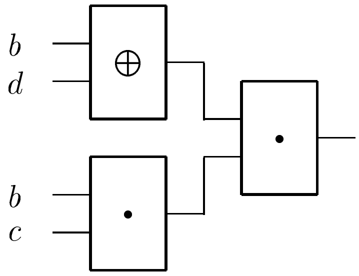
1) Straightforward, using two stages with 4 gates and 9 inputs:



2) By means of $b(c+\bar{d}) + d\bar{b}$, which requires also 4 gates but only 8 inputs. However, this is a three-stage circuit. Because each stage has a certain delay, for example 10ns, it is a little bit slower than the first solution. We speak of *ripple-through:* a change of c first changes the OR output, then the AND, and finally the output OR. This means 30ns in total, which may be too much in time-critical applications.



3) We can write $bc + b\bar{d} + d\bar{b} = bc + (b \oplus d)$. This solution requires only 3 gates with 6 inputs. However, now we need an EXCL-OR, which costs an extra chip if there is not one left over from a chip for some other circuit. One of the principles of hardware design is the optimisation of the use of available space on a printed circuit board by minimising the number of chips!



• An exclusive-OR combination is evident because of the structure in the Karnaugh diagram:

	1	1	
1			1

In general, an exclusive-OR can de recognised by checkerboard patterns:

		c	c	
	1		1	
1		1		b
	d	d		

$b \oplus c \oplus d$

		c	c	
1		1		
	1		1	b
	d	d		

$$\begin{aligned}(b \oplus c \oplus d)' &= b \oplus c \oplus \bar{d} \\ &= \bar{b} \oplus c \oplus d \\ &= b \oplus \bar{c} \oplus d\end{aligned}$$

If we need to use an exclusive-OR gate with 4 inputs, for instance because it is available on a printed circuit board, the last example becomes $b \oplus c \oplus d \oplus 1$. Verify this!

6.2 Don’t cares

With the definition of a logical function, all combinations of the input variables are assigned the two values *true* or *false* (1 or 0). However, it is possible that certain combinations can never occur, depending on other information or conditions. So one could assume arbitrarely a 0 or a 1 for these. This can lead to further simplifications! We speak of **don’t cares**.

Example: $T = \sum(1, 3, 4, 6, 7, 9, 11, 12, 14, 15, \underline{5}, \underline{13})$.
This is the last example of the previous section, but extended with the don’t cares 5 and 13 (we normally underline the minterms and represent the don’t cares in the Karnaugh diagram with a $\times$ symbol:

			c		
		1	1		
	1	$\times$	1	1	b
a	1	$\times$	1	1	
		1	1		
		d			

If we assume that the don’t cares are both 1,
we can take two blocks of 8 one’s.
Hence $T = b + d$.

For the development of conjunctive forms, one can make use of $\bar{T}$ and invert the result. This implies collecting all 0’s in the Karnaugh diagram: $\bar{T} = \bar{b}\bar{d} \rightsquigarrow T = b + d$. This example is not very illustrative. Let us take another

Example: $T = \sum(0, 1, \underline{2}, \underline{3}, 4, \underline{6}, 8, \underline{10}, 13, \underline{14})$.

			c		
	1	1	$\times$	$\times$	
	1	0	0	$\times$	b
a	0	1	0	$\times$	
	1	0	0	$\times$	
		d			

$\bar{T} = c + da\bar{b} + ab\bar{d} + \bar{a}db$
$T_{\text{conj}} = \bar{c}.(\bar{d} + \bar{a} + b).(\bar{a} + \bar{b} + d).(a + \bar{d} + \bar{b})$
$T_{\text{disj}} = \bar{b}\bar{a} + \bar{b}\bar{d} + \bar{d}\bar{a} + ab\bar{c}d$

The conjunctive form requires 3 ORs and 1 AND with 13 inputs, whereas the disjunctive form requires 1 OR and 4 ANDs with 14 inputs. However, the disjunctive form is a little bit simler because of three ANDs having only two inputs. This function can be further simplified by increasing the depth to three levels and employing one exclusive-OR:

$$T_{\text{conj}} = \bar{c}.[\bar{a} + (\bar{b} + d).(b + \bar{d})].(a + \bar{d} + \bar{b}).$$

This form requires only 4 elements with 9 inputs. *Check this!*

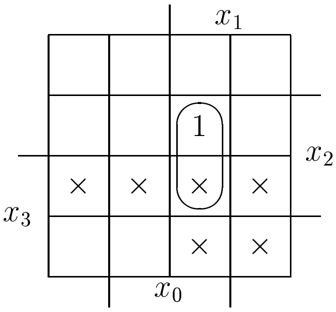
6.3 Application: code conversion

We will illustrate the application of don't cares by looking at the conversion between different codes.

6.3.1 BCD to DECIMAL

In this case we have 4 input lines x_3, x_2, x_1 and x_0, but 10 output lines labeled 0, 1, 2, ..., 9. The BDC codes go from 0000, 0001, 0010, to 1001, but the following binary combinations cannot occur. So we can use these as don't cares in the Karnaugh diagram:

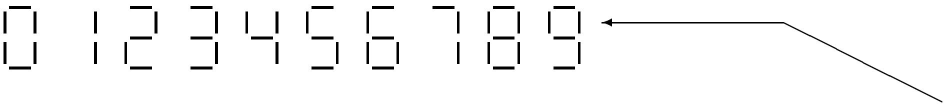
x_3	x_2	x_1	x_0	DEC
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	×
·	·	·	·	×
1	1	1	1	×



Example: $T_7 = x_2 x_1 x_0$.
It may be a good idea to try one other output yourself.

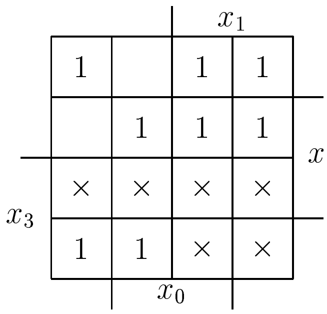
6.3.2 BCD to 7-SEGMENT

Some displays use seven line segments to show the numbers 0 to 9:



We must now make a table for each segment. We'll take the upper horizontal one:

x_3	x_2	x_1	x_0	−
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	×
·	·	·	·	×
1	1	1	1	×



We can recognise two blocks of 8 ones (x_3 and x_1), plus two blocks of 4 ones (in the center and the four corners).

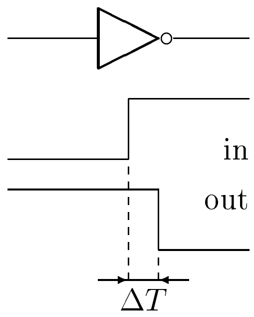
Hence:

$$\text{top} - = \bar{b}\bar{d} + a + c + bd = (b \oplus d)' + a + c$$

This can be done for all segments. *Try one!*

6.4 Hazards in combinatorial circuits

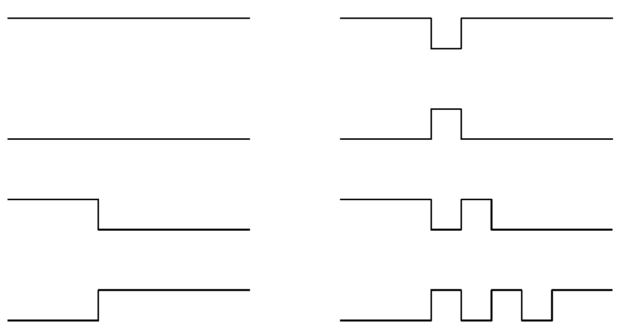
A **hazard** is a momentary error (a spurious signal or glitch) that may occur in an output of a combinatorial circuit, i.e. a circuit that consists only of normal gates without feedback loops. The circuit works correctly according it's specification, but due to different delay times some different output signals may occur before all outputs reach a steady state:



Taking an invertor (but the same holds for all different gates), if the input changes from 0 to 1, the output will go from 1 to 0, but with a delay of ΔT, normally around xx ns.

The delay ΔT may be about equal for different gates, but not exactly equal. Moreover, there may exist different paths with a different number of gates, in which case timing differences of $n\Delta T$ may occur. If a combinatorial circuit drives a sequential circuit which consists of flip-flops with edge- or transition-sensitive inputs, hazards may cause serious problems.

There are different hazard types. At the left we see stable signals or transitions without hazards and at the right with hazards:



static 1 hazard

static 0 hazard

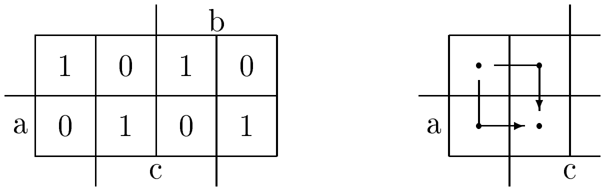
dynamic 1-to-0 hazard

dynamic 0-to-1 hazard

In the case of a static 1 hazard, the signal must remain 1 but shows a small 0 glitch. In the case of a dynamic 1-to-0 hazard, the signal should go from 1 to zero, but it shows one logic 1 glitch. In the last case the signal should go from 0 to 1, but shows two logic 0 glitches. If a circuit is more complicated, the number of glitches can be bigger.

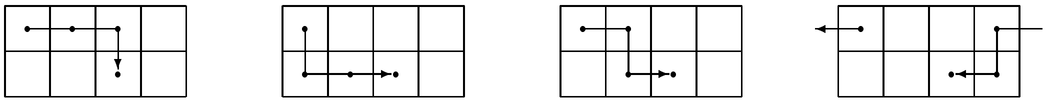
Now, how can hazards be detected and eliminated? They can be traced in a Karnaugh diagram when two or more input signals change and the function change cannot be covered by function combinations in the diagram. These are called **function hazards**.

Example:



If $b = 0$ and ac changes from 00 to 11, then depending on what input changes first, a or c, two different paths are followed which both include an output value 0, so the output goes from 1 via 0 to 1: This is a static 1 function hazard. *Check* $b = 0$ and $ac = 10 \Rightarrow ac = 01$.

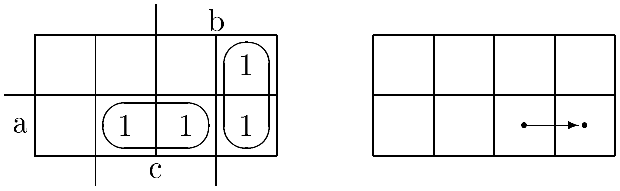
Taking the same Karnaugh diagram, but going from $abc = 000$ to $abc = 111$, we get in all cases This is a dynamic 1-to-0 function hazard. Four different paths are shown in the diagrams below. *Check* $abc = 100 \Rightarrow 011$.



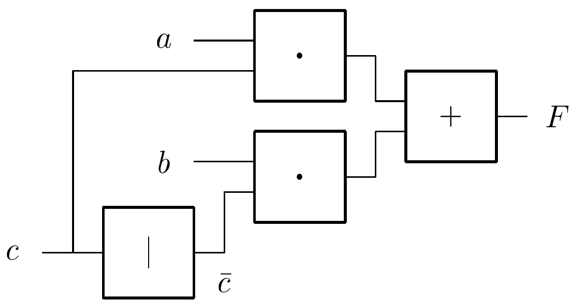
Function hazards in a combinatorial circuit cannot be controlled or avoided, unless exact timings of input signals and gate delays are respected. However, this is in practice impossible. In such cases we need to apply synchonous sequential circuits and flip-flops, which will be treated later on in Chapters 7 and 8.

A **logic hazard** is due to different delays when only one input signal changes. Logic hazards can be eliminated!

Example:



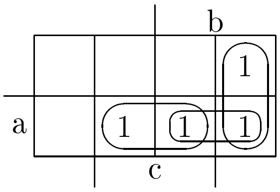
In this case we get the function $F = a.c + b.\bar{c}$. If $ab = 11$ and c changes from 1 to 0, as indicated by the arrow, we do not expect any problem because at both places we have a 1 without passing a 0. But in practice we can obtain , which is a static 1 function hazard. In order to explain this, we need to look at the circuit:



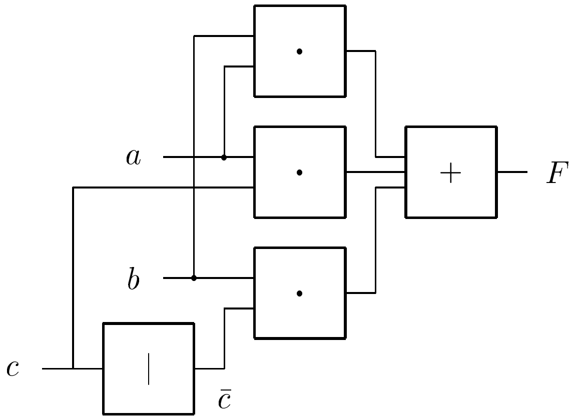
The delay of the invertor (the gate with the $|$ symbol) causes different input combinations at the inputs of the two AND gates:

c	$\bar{c}$	
1	0	steady situation, $\bar{c} = \neg c$
0	0	input changed but output still 0, $\bar{c} \neq \neg c$
0	1	steady situation again, $\bar{c} = \neg c$

Solution: include an extra min-term that <u>connects</u> the subsets in the Karnaugh diagram:



We obtain the function $F = a.c + b.\bar{c} + ab$.
Since a and b do not change, only c changes, the term ab keeps F at the level 1 when c changes. The drawback is that more gates are required, see the circuit below.



Below are two other examples. Do you know how many and which redundand terms are required to avoid hazards?