

3 Combinatorial Logic

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

- 3. Combinatorial Logic** 2
 - introductional example 2
 - 3.1 Combinatorial Circuit** 2
 - 3.1.1 Example 2
 - 3.1.2 Sum of Products 3
 - Note! 4
 - Note! 5
 - 3.1.3 Product of Sums 6
 - Note! 6
 - Note! 7
 - 3.2 Karnaugh Map** 7
 - 3.2.1 Introduction with the two dimensional Map 7
 - 3.2.2 The three dimensional Karnaugh Map 8
 - Note! 15
 - 3.2.3 Four Dimensional Karnaugh Map 15
 - Disjunctive Form 18
 - Conjunctive Form 21
 - 3.2.4 Rules for the Karnaugh map 23
 - Allowed Groups 23
 - Necessary and Important Groups 24
 - Note! 24
 - higher dimensional Karnaugh Maps 27
 - Exercises 27
 - Exercise 3.1.1 CNF, DNF, Optimization 28
 - Exercise 3.1.2 Implementation of a BCD-to-7-Segment Decoder 28
 - Exercise 3.1.3 Further Questions 29

3. Combinatorial Logic

introductory example

Fig. 1: Simulation of a 7-segment encoder and display

The combinatorial logic shown in [figure 1](#) enables to output distinct logic values for each logic input. When you change the input nibble you can see that the the correct number appears on the 7-segment-display. By clicking onto the bits of the input, you can change the number.

Tasks:

1. Which output Y_0 ... Y_6 is generated from the input nibble 1000? Which from 1001?
2. Is the output only depending on the input? Is there a dependance on the history?

3.1 Combinatorial Circuit

Up to now, we looked onto simple logic circuits. These are relatively easy to analyze and synthesize (=develop). The main question in this chapter is: how can we set up and optimize logic circuits?

In the following we have a look onto combinatorial circuits. These are generally logic circuits with

- n inputs X_0, X_1, \dots, X_{n-1}
- m outputs Y_0, Y_1, \dots, Y_{m-1}
- no "memory", that is: a given set of input bits results in a distinct output

They can be description by

- truth table
- boolean formula
- hardware description language

The ladder one is not in the focus of this course.

The applications range:

- (simple) half/full adder
- [digital comparators](#) (logic circuit to compare 2 values)
- Multiplexer / demultiplexer
- Arithmetic logic units in microcontrollers and processors
- much more

3.1.1 Example

In order to understand the synthesis of a combinatoric logic we will follow a step-by-step example for this chapter.

Imagine you are working for a company called “mechatronics and robotics”. One customer wants to have an intelligent switch as input device connected to a microcontroller for controlling an oven. For this project “Therm-o-Safety” he needs a combinatoric logic:

- The intelligent switch has 4 user selectable positions: \$1\$, \$2\$, \$3\$, \$4\$
- Additionally there are 2 non-selectable positions for the case of failure.
- The output \$Y=1\$ will activate a temperature monitoring.
- The temperature monitoring has to be active for \$3\$ and \$4\$ and in case of a major failure. A major failure is for example, when the switch position is unclear. In this case the input of the combinatorial circuit is “ON”.
- There are no other cases of inputs.

This requirements are put into a truth table:

Therm-o-Safety				
Input	X2	X1	X0	Y
	0	0	0	-
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
	1	0	1	-
OFF	1	1	0	0
ON	1	1	1	1



Fig. 2: Therm-o-Safety truth table

Text is not SVG - cannot display

figure 2 shows one implementation of this requirements. The inputs 001 ... 011 represent the inputs \$1\$...\$4\$. The cases of failure are coded with 110 and 111.

The output \$Y\$ is activated as requested. For the two combinations 000 and 101 there is no output value defined. Depending on the requirements for a project these shall either better be 0 or 1 or the output of these does not matter. We had this “does not matter” before: the technical term is “I don't care”, and it is written as a - or a x.

By this, we have done the first step in order to synthesize the requested logic.

3.1.2 Sum of Products

Now, we want to investigate some of the input combinations (= lines in the truth table). At first, we have a look onto the input combination 011, where the output has to be \$Y=1\$.

If this input combination would be the only one for the output of \$Y=1\$, the following could be stated: “\$Y=1\$ (only) when the \$X_0\$ is \$1\$ AND \$X_1\$ is \$1\$ AND \$X_2\$ is \$0\$”. It can also be rearranged to:

“\$Y=1\$ (only) when the \$X_0\$ is \$1\$ AND \$X_1\$ is \$1\$ AND \$X_2\$ is not \$1\$”.

This statement is similar to $X_0 \cdot X_1 \cdot \overline{X_2}$. The used conjunction results only in

\$1\$, when all inputs are \$1\$. The negation of \$X_2\$ takes account of the fact, that \$X_2\$ has to be \$0\$.

Fig. 3: Therm-o-Safety truth table - first analysis

Therm-o-Safety				
Input	X2	X1	X0	Y
	0	0	0	-
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
	1	0	1	-
OFF	1	1	0	0
ON	1	1	1	1

$\overline{X_2} \& X_1 \& X_0$

figure 3 shows the boolean expression for this combination. In figure 4, this boolean expression is converted into a structure with logic gates.

Fig. 4: logic circuit for the combination '011'

With the same idea in mind, we can have a look for the other combinations resulting in \$Y=1\$. These are the combinations 100 and 111:

- For 100 The statement would be: “\$Y=1\$ (only) when the \$X_0\$ is \$0\$ AND \$X_1\$ is \$0\$ AND \$X_2\$ is \$1\$”. Similar to the combination above this leads to: $\overline{X_0} \cdot \overline{X_1} \cdot X_2$.
- For 111, the boolean expression is $X_0 \cdot X_1 \cdot X_2$.

Note!

- Each row in a truth table (=one distinct combination) can be represented by a **minterm** or **maxterm**
- A **minterm** is the conjunction (AND'ing) of all inputs, where under certain instances a negation have to be used
- In a minterm an input variable with 0 has to be negated, in order to use it as an input for the AND.
e.g. $X_0 = 0$ AND $X_1 = 1 \quad \rightarrow \quad \overline{X_0} \cdot X_1$
- A minterm results in a output of 1

Fig. 5: Therm-o-Safety truth table - sum of products

Therm-o-Safety			
Input (X ₂ X ₁ X ₀)	Y	Y'	Minterm
000	0	1	$\overline{X_2} \overline{X_1} \overline{X_0}$
001	0	1	$\overline{X_2} \overline{X_1} X_0$
010	1	0	$\overline{X_2} X_1 \overline{X_0}$
011	1	0	$\overline{X_2} X_1 X_0$
100	1	0	$X_2 \overline{X_1} \overline{X_0}$
101	0	1	$X_2 \overline{X_1} X_0$
110	0	1	$X_2 X_1 \overline{X_0}$
111	0	1	$X_2 X_1 X_0$

In figure 5 all minterms for $Y=1$ are shown. The figure 6 depicts all the logic circuits for the three minterms. These lead to the outputs Y' , Y'' , and Y''' .

Fig. 6: logic circuit for the combinations '100', '110', '111'

For the final step we have to combine the single results for the minterms. The output has to be $Y=1$ when at least one of the minterms is $Y=1$. Therefore, the minterms have to be connected disjunctive:

$$Y = \overline{X_2} X_1 \overline{X_0} + \overline{X_2} X_1 X_0 + X_2 \overline{X_1} \overline{X_0} + X_2 \overline{X_1} X_0 + X_2 X_1 \overline{X_0} + X_2 X_1 X_0$$

This leads to the logic circuit shown in figure 7. Here, you can input the different combinations by clicking onto the bits of the input nibble.

Fig. 7: logic circuit for therm-o-safety

Note!

- The disjunction of the minterms is called **sum of products, SoP, disjunctive normal form or DNF.**
- When all inputs are used in each of the minterms the normal form is called **full disjunctive normal form**
- When synthesizing a logic circuit by sum of products, all 'don't care' terms outputting 0 .

We have seen, that the sum of products is one tool to derive a logic circuit based on a truth table. Alternatively it is also possible to insert an intermediate step, where the logic formula is simplified.

In the following one possible optimization is shown:

$$Y = \overline{X_2} X_1 \overline{X_0} + \overline{X_2} X_1 X_0 + X_2 \overline{X_1} \overline{X_0} + X_2 \overline{X_1} X_0 + X_2 X_1 \overline{X_0} + X_2 X_1 X_0$$

$$= \overline{X_2} X_1 (\overline{X_0} + X_0) + X_2 \overline{X_1} (\overline{X_0} + X_0) + X_2 X_1 (\overline{X_0} + X_0)$$

$$= \overline{X_2} X_1 + X_2 \overline{X_1} + X_2 X_1$$

$$= \overline{X_2} X_1 + X_2 (\overline{X_1} + X_1)$$

$$= \overline{X_2} X_1 + X_2$$

3.1.3 Product of Sums

In the sub-chapter before we had a look onto the combinations which generates an output of $Y=1$ by means of the AND operator. Now we are investigating the combinations with $Y=0$. Therefore, we need an operator, which results in 0 for only on distinct combination.

The first combination to look for is 001 . If this input combination would be the only one for the output of $Y=0$, the following could be stated:

" $Y=0$ (only) when the X_0 is 1 AND X_1 is 0 AND X_2 is 0 "

With having the duality in mind (see cpt. [The Set of Rules](#)) the opposite is also true:

" $Y=1$ when X_0 is 0 OR X_1 is 1 OR X_2 is 1 "

This is the same like: $\overline{X_0} + X_1 + X_2$

The booleand operator we need here is the OR-operator.

Similarly, the combinations 010 und 110 can be transformed. Keep in mind, that this time we are looking for a formula with results in 0 only for the given one distinct combination.

Note!

- A **maxterm** is the disjunction (OR'ing) of all inputs, where unter certain instances a negation have to be used.
- In a maxterm an input variable with 1 has to be negated, in order to use it as an input for the OR.
- A maxterm results in a output of 0

The [figure 8](#) shows all the maxterms for the Therm-o-Safety example.

Fig. 8: Therm-o-Safety truth table

...	
t...	
	$\overline{X_0}$
t...	

Text is not SVG - cannot display

The formulas of [figure 8](#) can again be transformed into gate circuits ([figure 9](#)). Here, only for the inputs '001', '010', '110' one of the outputs Y' , Y'' or Y''' is 0 .

Fig. 9: logic circuit for the combinations '001', '010', '110'

When these intermediate outputs Y' , Y'' , Y''' are used as an input for an AND-gate the resultin output will get 0 when at least one of the intermediate outputs are 0 . This results in another way to synthesize the Therm-o-Safety (see [figure 10](#))

Fig. 10: logic circuit for therm-o-safety

Also the products of sum can be simplified:

$$\begin{aligned} Y &= (\overline{X_0} + X_1 + X_2) \cdot (\overline{X_1} + X_2) \cdot (\overline{X_0} + \overline{X_1} + X_2) \\ &\quad \cdot (\overline{X_0} + \overline{X_1} + X_2) \cdot (\overline{X_0} + \overline{X_1} + X_2) \cdot (\overline{X_0} + \overline{X_1} + X_2) \end{aligned}$$

This result Y by the sum of products is different compared to the result in product of sums:

- product of sums: $Y = (\overline{X_0} \cdot \overline{X_1} \cdot X_2) + (X_0 \cdot X_1)$
- sum of products: $Y = (\overline{X_0} + X_1 + X_2) \cdot (\overline{X_0} + \overline{X_1})$

In this case these results cannot be transformed into each other with the means of boolean rules.

Note!

- The disjunction of the maxterms is called **products of sum, PoS, conjunctive normal form** or **CNF**.
- When all inputs are used in each of the minterms the normal form is called **full conjunctive normal form**
- When synthesizing a logic circuit by sum of products, all 'don't care' terms outputting 1
- The products of sum is the DeMorgan dual of the sum of products **if** there are no don't care terms. Otherwise the results cannot be transformed into each other with the means of boolean rules.

3.2 Karnaugh Map

3.2.1 Introduction with the two dimensional Map

For a simple introduction we take one step back and look onto a simple example. The formula $Y(X_1, X_0) = X_0 \cdot X_1$ combines two variables. Therefore, it has two dimensions. In [figure 11](#) (a) the truth table of this is shown. The most left column shows the decimal interpretation of the binary numeral given by X_1 , X_0 (e.g. $(X_1=1, X_0=0) \rightarrow 10_2 = 2_{10}$).

The given logic expression can also be interpreted in a coordinate system, with the following conditions:

- There are only two coordinates on each axis possible: 0 and 1 .
- There are as much axis as variables given in the logic: For the example we have two variables X_0 and X_1 . These are spanning a two dimensional system.
- On the possible positions, the results Y have to be shown.

In the following pictures of this representation the values are shown as:

- green dot, when the result is 1
- red dot, when the result is 0
- grey dot, when the result is don't care

For the given example the coordinate system shows four possible positions: This are the edges of a

square (figure 11 (b)).

We will in the future write this as in figure 11 (c). This diagram is also called **karnaugh map** (often called k-map or KV map). In the shown Karnaugh map the coordinate X_0 is shown vertically and X_1 horizontally. Similar to the coordinate system the upper left cell is for $X_1=0$ and $X_0=0$. The upper right cell is for $X_1=1$ and $X_0=0$, the lower right one for $X_1=1$ and $X_0=1$. In each cell the result $Y(X_1, X_0)$ is shown as large number - similar to the color code in the coordinate system. The small number is the decimal representation of the number given by X_1 and X_0 . This index is often not explicitly shown in the Karnaugh map, but simplifies the fill-in of the map and helps for the start.

Fig. 11: two dimensional Karnaugh map

sxx_0_1_		y	
dec	x1	x0	y
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	0

The karnaugh map will help us in the following to find simplifications of more complex logic expressions.

3.2.2 The three dimensional Karnaugh Map

In this subchapter we will have a look onto our example of the therm-o-safety. For this example we found two possible gate logics which can produce the required output. We have also seen, that optimizing the terms (i.e. the min- or maxterms) is often possible. But we do not know how we can find the optimum implementation.

For this, we try to interpret the inputs of our example again as dimensions in a multidimensional space. The three input variables X_0 , X_1 , X_2 span a 3-dimensional space. The point 000 is the origin of this space. The three combinations 001, 010, 100 are onto the X_0 -, X_1 -, and X_2 -axis, respectively (see figure 12 (a)). The other combinations can be reached by adding these axis values together (see figure 12 (b)+(c)). This is similar to the situation of a two dimensional or three dimensional vector. Three inputs result in this representation in the edges of a cube.

In the figure 12 (d) the situation $X_0=1$, $X_1=1$, $X_2=0$ is shown.



Fig. 12: 3 dimensional cube representation

There is also an alternative way to look onto this representation:

- The formula $Y=1$ (independent from inputs $X_0...X_{n-1}$) lead to all positions are 1
- A single input equal 1 ($Y:\, X_0=1$, independent from all other inputs, i.e. all others are don't care) lead in the three dimensional example to the edges of a side surface of the cube. In our example $\color{violet}\{X_1=1\}$ lead to the situation shown in figure 13 (a). When investigating the shown green dots $\mathbf{010}, \mathbf{011}, \mathbf{110}, \mathbf{111}$ it is visible that the middle value (= the value for X_1) is the same. Generally: A single input equal 1 (independent from all other inputs) lead to a structure one dimension smaller than the number of inputs (In our example: 3 inputs \rightarrow two dimensional structure = surface).
- Multiple given inputs equal 1 lead to smaller structures correspondingly. In our example: $\color{blue}\{X_0=1\}$ and $\color{violet}\{X_1=1\}$ ($=\color{blue}\{X_0\}\cdot\color{violet}\{X_1\}$) result in the two edges on a corner of the cube (figure 13 (b)). For this coordinates ($\mathbf{011}, \mathbf{111}$) the last two values are the same.
- A minterm (=1 as an output) in our example is given by the intersection of all surfaces for the individual dimensions. In our example: $\color{blue}\{X_0=1\}$ and $\color{violet}\{X_1=1\}$ and $\color{brown}\{\overline{X_2}=0\}$ ($=\color{blue}\{X_0\}\cdot\color{violet}\{X_1\}\cdot\color{brown}\{\overline{X_2}\}$) result in the two edges on a corner of the cube (figure 13 (c))

On the right side of figure 13 also the truth table is shown. There, the combinations for each side surfaces of the cube is marked with the corresponding color.

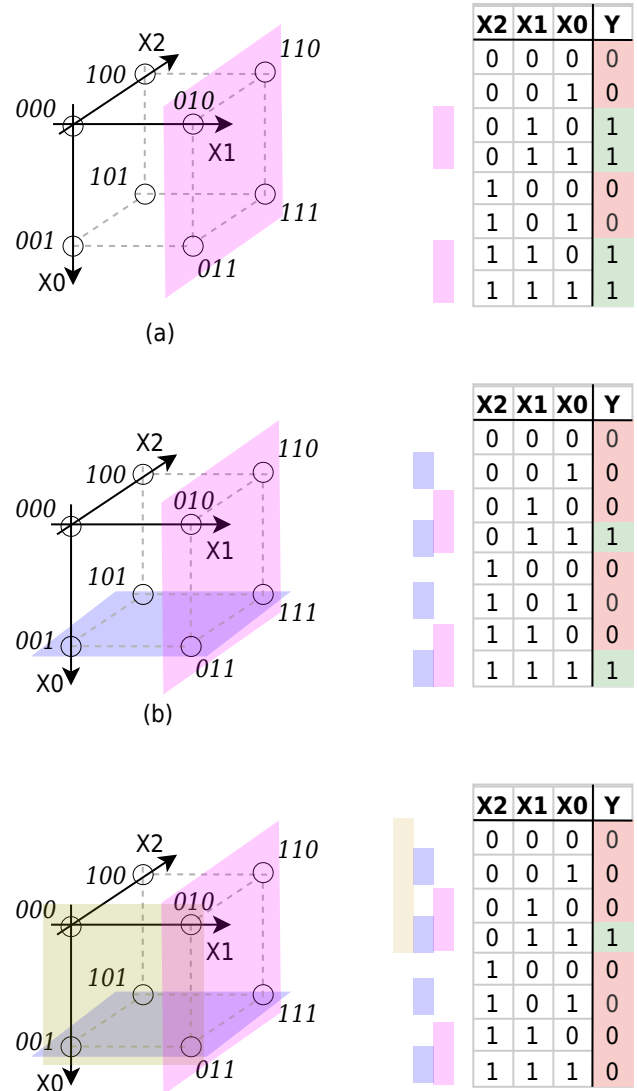


Fig. 13: examples in 3 dimensional cube representation

(c) Text is not SVG - cannot display

With this representation in mind, we can simplify other representations much more simpler. One example for this would be to represent the formula: $Y = X_0 \cdot X_1 \cdot \overline{X_2} + X_2 \cdot X_1 + X_0 \cdot X_1$. By drawing this into the cube one will see that it represents only a side surface of the cube. It can be simplified into $Y = X_1$.

We can also try to interpret our Therm-o-Safety truth table. The figure 14 shows the corresponding cube. The problem here is, that it is a bit unhandy to reduce a three dimensional cube onto a flat monitor or a paper. It will also get more stressfull for higher dimensions.

Fig. 14: Therm-o-Safety in multi-dimensional space

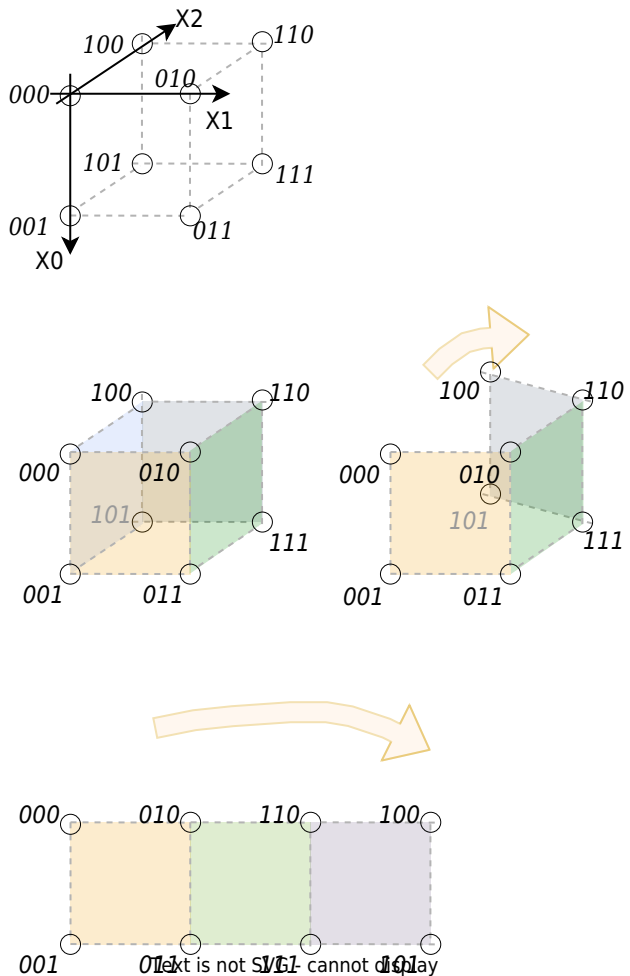
Therm-o-Safety				
Input	X2	X1	X0	Y
	0	0	0	-
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
	1	0	1	-
OFF	1	1	0	0
ON	1	1	1	1



Text is not SVG - cannot display

Therefore, we try to find a better way to sketch the coordinates, before we simplify our Therm-o-Safety. For the three dimensional Karnaugh map it is a good idea “unwrap” the cube. This can be done as shown in figure 15.

Fig. 15: flattening the 3 dimensional cube representation



Text is not SVG - cannot display

Out of the flattened cube we can derive the three dimensional Karnaugh map (see figure 16).

Generally, there are different ways to show the Karnaugh map.

1. One way is to show the variable names of the dimensions in the corner. Be aware, that the order of the numbers in horizontal direction is 0, 0, 0, 1, 1, 1, 1, 0 and not in ascending order!
2. In other ways the columns / rows related to the dimension are marked with lines. In this representation only the TRUE (=1) position of the coordinate is highlighted.

In [figure 16](#) the dimensions are additionally marked with colors.

In the following chapters mainly the visualisation shown in [figure 16](#) (b) is used.

Fig. 16: different representation of the three dimensional Karnaugh map I



		X_2, X_1			
		$0,0$	$0,1$	$1,1$	$1,0$
X_0	0	-	0	0	1
	1	0	1	1	-

(a)

		X_1			
		0	2	6	4
X_0	1	0	1	1	-
	3	7	5		

(b)

		X_1			
		0	2	6	4
X_0	1	0	1	1	-
	3	7	5		

(c)

Text is not SVG - cannot display

With this representation we can now try to read out the logic terms for the Therm-o-Safety from its Karnaugh map. [figure 17](#) shows the neighbouring combinations:

- In light brown the group (011 + 111) resp. position 3 and position 7 is shown. This can be simplified into $X_0 \cdot X_1$
- In light blue the group (101 + 111) resp. position 5 and position 7 is shown. This can be simplified into $X_0 \cdot X_2$. This group seems not to be needed, since 111 is already in the light brown group.

- In light violet the group (000 + 100) resp. position 0 and position 4 is shown. This can be simplified into $\overline{X_0} \cdot \overline{X_1}$

The first two groups are also neighbouring cells in the Karnaugh map. For the last one we have to keep in mind, that we had to cut the surface of the cube in order to flatten it. Therefore, this cells are also neighbouring. These leads to the conclusion, that the borders of the Karnaugh map are connected to the opposite borders!

For our example the result would be: (light brown the group) + (light violet the group) $= X_0 \cdot X_1 + \overline{X_0} \cdot \overline{X_1}$

Fig. 17: simplification of the three dimensional Karnaugh map



We also have to remember, that there are multiple permutations to show exactly the same logic assignment. This can be interpreted as other ways to unwrap the cube. The figure 18 shows the variant from before at (a). In image (b) the coordinates are mixed ($X_0 \rightarrow X_1$, $X_1 \rightarrow X_2$, $X_2 \rightarrow X_0$). In the image (c) the position of the origin is not on the upper left corner anymore.

Independent from the permutation, the grouped cells are always neighbouring each other.

Fig. 18: different representation of the three dimensional Karnaugh map II

Therm-o-Safety				
Input	X2	X1	X0	Y
	0	0	0	-
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
	1	0	1	-
OFF	1	1	0	0
ON	1	1	1	1



Text is not SVG - cannot display

Note!

- The Karnaugh map is an alternative way to represent a logic relation.
- There are possible grouping in order to simplify the logic.
- We need to take care of whether a group is needed or not. (will be done in chapter 3.3)

3.2.3 Four Dimensional Karnaugh Map

For the four dimensional Karnaugh map the situation becomes in the classical coordinate system more complicated. The respective object would be a four-dimensional hypercube (see figure 19). This is hard to print in a two dimensional layout like on a website or on a page. Here, a four dimensional Karnaugh map might be a good representation.



Fig. 19: four dimensional hypercube

Text is not SVG - cannot display

In order to create a four dimensional Karnaugh map, we look at first how the two and three dimensional Karnaugh map can be derived. The figure 20 (a) shows, that the two dimensional

Karnaugh map can be created from a one dimensional one by folding the table on the x-axis and adding X_1 to all values. The additional line is marked with the new dimension X_1 .

The three dimensional one is created by folding the table on the y-axis and adding X_2 to all values. The additional line is marked with the new dimension X_2 (figure 20 (b)). Be aware, that the X_0 marking now has to be extended - it is also folded to the right.

The four dimensional one is created by folding the table again on the x-axis and adding X_3 to all values. The additional line is marked with the new dimension X_3 (figure 20 (c)).

By this we can visually derive the four dimensional Karnaugh map.

Fig. 20: four dimensional Karnaugh map



Again, there are alternative ways to show the Karnaugh map. To get the index of each cell one can easily add up the values of the dimension $X_0 \dots X_3$. This is shown on an example in figure 21.

Fig. 21: different representations of a four dimensional Karnaugh map



For getting used to the four dimensional Karnaugh map, we expand our Therm-o-Safety: Instead of four user selectable levels for oven, the version 2.0 will have seven. The temperature monitoring (\$Y=1\$) has to be active starting with level \$4\$. Additionally, the Therm-o-Safety 2.0 has three non-selectable positions for the case of failure, where the last one needs active temperature monitoring. Some of the combinations (0000, 1000, 1001, 1010, 1011, 1100) are not needed.

The truth table of the new Therm-o-Safety 2.0 is shown in [figure 19](#)

The group $\$I\$$ in [figure 24](#) can be expanded, since there are don't care states nearby:

Fig. 25: optimized disjunctive solution of Therm-o-Safety 2.0

This can be transformed back into a formula. We can derive the boolean terms from the Karnaugh map with the following steps (see [figure 26](#)):

1. Investigate each single group individually.
2. For the sum of products each group has to be the product (AND-combination) of the inputs / coordinates.
3. Then the formula can be derived as the sum of products...

Fig. 26: Interpretation of the formula

The given groups are created as a product as following:

- group I: all of this minterms are in the columns of X_3 . They are also all in the columns of X_1 and in the rows of X_0 . This leads to $X_0 \cdot X_3 \cdot X_1$
- group II: all of this minterms are in the columns of $\overline{X_3}$. They are also all in the rows of X_2 . This leads to $\overline{X_3} \cdot X_2$

Out of this groups we can get the full formula by disjunctive combination:
$$Y = X_0 \cdot X_3 \cdot X_1 + \overline{X_3} \cdot X_2$$

Conjunctive Form

Similarly we look onto the conjunctive form. Here we have to find groups of 0s. One way of grouping the maxterms is shown here:

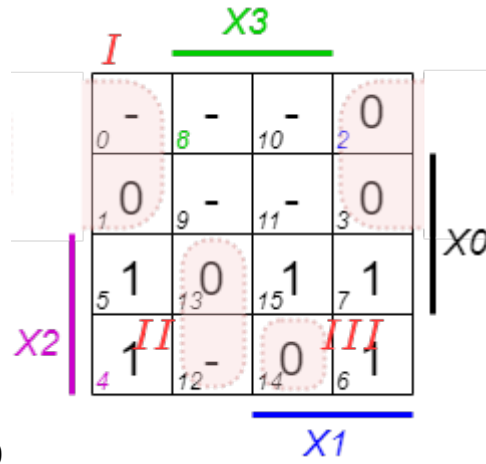


Fig. 27: Conjunctive solution of Therm-o-Safety 2.0

The optimization would be:



Fig. 28: optimized conjunctive solution of Therm-o-Safety 2.0

Also here, we can derive the boolean terms from the Karnaugh map with the following steps (see figure 29):

1. Investigate each single group individually.
2. For the products of sum each group has to be the sum (OR-combination) of the inputs / coordinates.
Be aware, that for the OR-combination we get the groups of 0s as "rest" of not marked cells!
3. In the end the formula can again be derived, now as the product of sums.

Fig. 29: Interpretation of the formula

```
\begin{align*} Y &= ( \color{magenta}\{X_2\} ) \cdot ( \color{green}\{\overline{X_3}\} + \color{blue}\{X_1\} ) \cdot ( \color{green}\{\overline{X_3}\} + \{X_0\} ) \end{align*}
```

Beyond the maxterms this formula can be optimized to

```
\begin{align*} Y &= \color{magenta}\{X_2\} \cdot ( \color{green}\{\overline{X_3}\} + ( \color{blue}\{X_1\} \cdot \{X_0\} ) ) \end{align*}
```

When comparing the disjunctive solution ($X_3 \cdot X_1 \cdot X_0 + \overline{X_3} \cdot X_2$) with the conjunctive one ($X_2 \cdot X_1 \cdot X_0 + \overline{X_3} \cdot X_2$) we see, that these are definitely different. The given truth table had don't care states. This can be taken as 0s or 1s - and therefore combined in groups of 0s or groups of 1s.

3.2.4 Rules for the Karnaugh map

We saw, that with the Karnaugh map we can analyze logic combinations much better. But to use this tool right we have first to look onto some definitions.

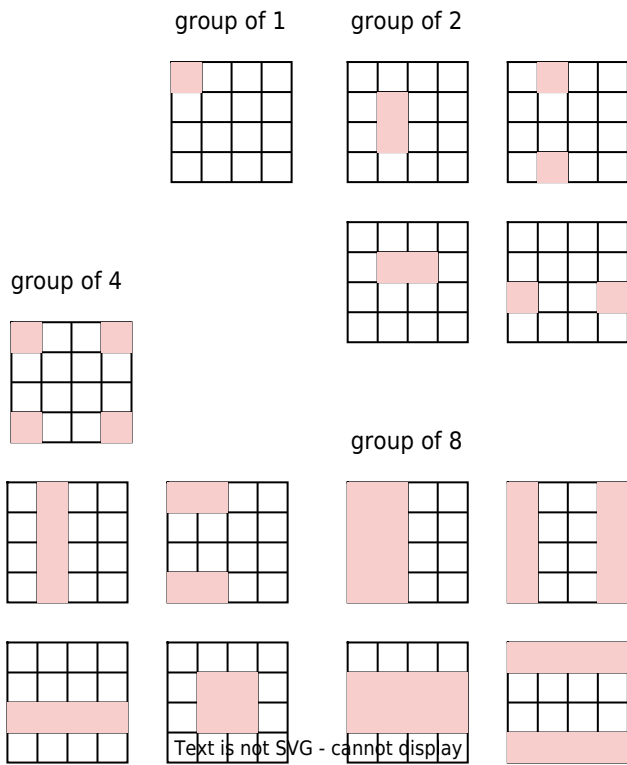
Allowed Groups

As we have seen, the groups in the Karnaugh map are created by the combination of the inputs. By this only distinct groups are allowed. These can only have 2^{n-1} cells for n inputs (= dimensions).

For a four dimensional Karnaugh map only the following groups are possible:

- groups of 1: Derived from 4 inputs, e.g. $X_3 \cdot \overline{X_2} \cdot \overline{X_1} \cdot \overline{X_0}$
- groups of 2: Derived from 3 inputs, e.g. $\overline{X_3} \cdot X_1 \cdot \overline{X_0}$
- groups of 4: Derived from 2 inputs, e.g. $X_2 \cdot X_0$
- groups of 8: Derived from 1 inputs, e.g. $\overline{X_1}$

Fig. 30: allowed groups in a four dimensional Karnaugh map



Keep in mind, that not all groups of 2, 4 or 8 cells are allowed. In [figure 31](#) some not allowed groups are shown



Fig. 31: examples for not allowed groups

Necessary and Important Groups

In the sub-chapter before the creation of the groups was done rather intuitively. This shall be explained more structured here. His needs some more definitions.

Note!

- **implicant** (I) is what we called “group” up to now. An implicant can be any allowed combination of minterms ($\$1\s) or maxterms ($\$0\s).
- **prime implicant** (PI) is an implicant, which is not completely included in any other possible implicant.
- **core prime implicant** (CPI) is a prime implicant, containing at least one term, which is not covered with any other prime implicant.

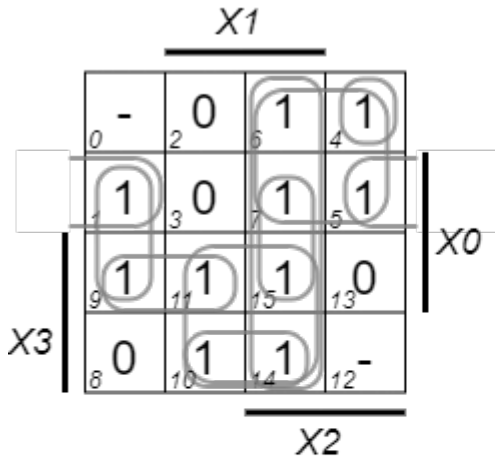
The (non core) prime implicant additionally are separated into:

- **Redundant prime implicants:** These are prime implicants, which are fully contained in a disjunction of multiple core prime implicants.
- **Selective prime implicants:** These are prime implicants, which are neither core prime implicants nor redundant prime implicants.

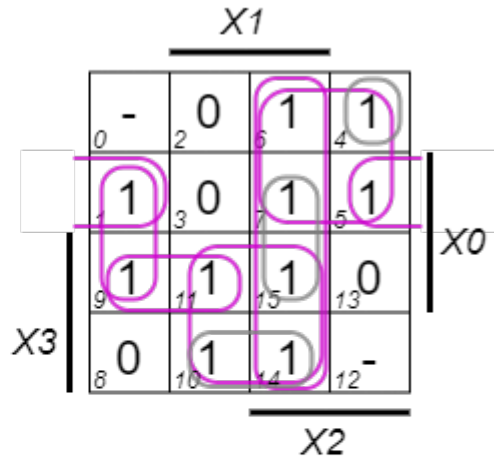
[figure 32](#) depicts the different implicants on one example.

Fig. 32: different implicants

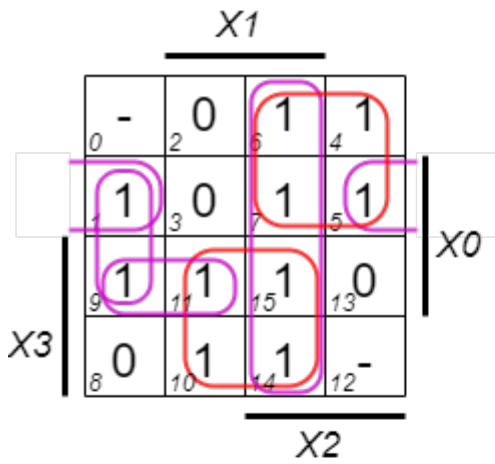
Implicants



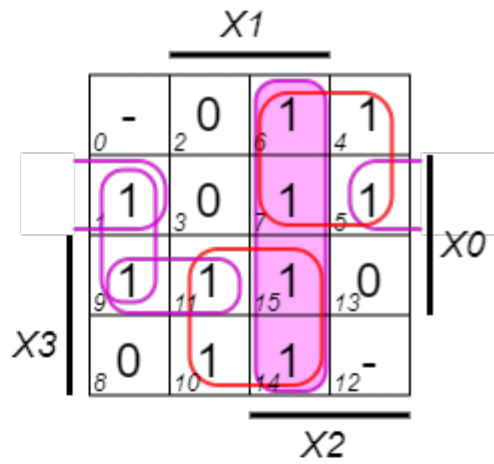
Prime Implicants



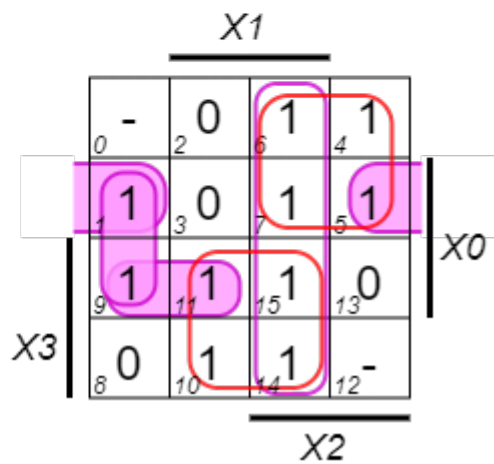
Core Prime Implicants



Redundant Prime Implicants



Selective Prime Implicants



In order to get all the necessary implicants the following has to be considered:

- **All core prime implicant** are needed. They contain terms, which are not covered anywhere else.
- **No redundant prime implicant** is needed. They are covered by core prime implicants
- **Selective prime implicant** need to be investigated. Some of them are necessary, depending on the solution (see [figure 33](#))

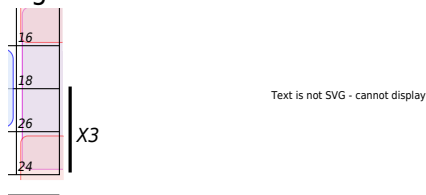
Fig. 33: selective prime implicants



higher dimensional Karnaugh Maps

For high dimensional Karnaugh maps the implicants look more am more unintuitive (see [figure 34](#)). The Karnaugh map is in our course used to understand the different types of implicants (there will be only three and four dimensional maps in the exam).

Fig. 34: five dimensional Karnaugh map



When solving higher dimensional boolean problems the [Quine-McCluskey algorithm](#) or a [heuristic approach, like the ESPRESSO algorithm](#) can be used. Languages like the Hardware Description Language (HDL) often come with implemented optimizer in the development environment.

Exercises

Tip: You can check your answer yourself.

- Online: [Normalforms](#) and [K-map](#)

- in the [Tool digital](#): via
 - Analysis » Synthesis
 - filling the truth table
 - change the variable names by right click onto the name
 - Kmap » Kmap

Exercise 3.1.1 CNF, DNF, Optimization

The following truth table is given.

1. Write down the DNF for the function table
2. Minimize the DNF step by step, stating the used boolean rules
3. Write down the CNF for the function table
4. Minimize the CNF step by step, stating the used boolean rules
5. Show, that the minimized DNF can be converted into the CNF (duality principle)
6. Create a Karnaugh map and mark the smallest number of largest prime implicants. Derive the minimized DNF and CNF from the map.

X2	X1	X0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Fig. 35: Exercise 3.1.1 CNF, DNF, Optimization

Text is not SVG - cannot display

Exercise 3.1.2 Implementation of a BCD-to-7-Segment Decoder

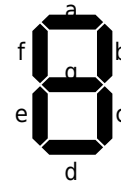
A full BCD-to-7-Segment Decoder with positive logic shall be developed. As a base the following truth table shall be used.

1. Write down the DNF and CNF for the function table and the outputs a...g. Use the don't care states wisely.
2. Optimize the functions by the use of 7 Karnaugh maps.

3. Show, that the minimized DNF can be converted into the CNF (duality principle)
4. Create a Karnaugh map and mark the smallest number of largest prime implicants.
Derive the minimized DNF and CNF from the map. Mark the smallest number of largest prime implicants.
Write down the derived, smallest formula based on CNF and DNF.
5. Check the expressions with the programm "Digital".
 1. Use at first only the DNF and CNF.
 2. After this, use the minimized DNF and the minimized DNF
 3. Check whether a 7-Segment-Display works fine with the created logic

Fig. 36: Exercise 3.1.2 Implementation of a BCD-to-7-Segment Decoder

Vaule		Input (binary)										
DEC	HEX	D3	D2	D1	D0	a	b	c	d	e	f	g
0	0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	1	0	1	1	0	0	0	0
2	2	0	0	1	0	1	1	0	1	1	0	1
3	3	0	0	1	1	1	1	1	1	0	0	1
4	4	0	1	0	0	0	1	1	0	0	1	1
5	5	0	1	0	1	1	0	1	1	0	1	1
6	6	0	1	1	0	1	0	1	1	1	1	1
7	7	0	1	1	1	1	1	1	0	0	0	0
8	8	1	0	0	0	1	1	1	1	1	1	1
9	9	1	0	0	1	1	1	1	1	0	1	1
10	A	1	0	1	0	-	-	-	-	-	-	-
11	B	1	0	1	1	-	-	-	-	-	-	-
12	C	1	1	0	0	-	-	-	-	-	-	-
13	D	1	1	0	1	-	-	-	-	-	-	-
14	E	1	1	1	0	-	-	-	-	-	-	-
15	F	1	1	1	1	-	-	-	-	-	-	-



Exercise 3.1.3 Further Questions

- Write a random allocation of 0s and 1s to the variations of inputs. Write the min- and maxterms
- Compare the results with the output given [here](#) (the output \$y\$ can be changed by clicking onto the grey cells)
- Use the [interactive example](#) and activate "Hide result" plus the push of the "Random example" button in order to get an individual test for a minterm optimization via a Karnaugh map.

From:

<https://wiki.mexle.org/> - **MEXLE Wiki**

Permanent link:

https://wiki.mexle.org/introduction_to_digital_systems/combinatorial_logic?rev=1667590558

Last update: **2022/11/04 20:35**

