3 Combinatorial Logic

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

3. Combinatorical Logic

introductional example

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

The combinatorial logic shown in <impref pic1> enables to output distinct logic values for eacht 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 nibble, 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 histroy?

3.1 Combinatorical Circuit

Up to now, we looked onto simple logic circuits. Thes 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 combinatorical circuits. These are generally logic circuits with

- n inputs X_0, X_1, … X_{n-1}
- m outputs Y_0, Y_1, … 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 costumer 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

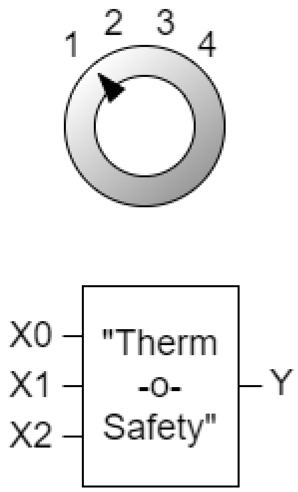


Fig. 2: Therm-o-Safety truth table

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 requierements 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 syntesize 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 re-arranged 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 conjuntion resuts 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

$\overline{X2}$ & X1 & X0

figure 3 shows the boolean expression for ths combination. In figure 4, this boolean expression is converted into a struction 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”. Similary 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 unter 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	**X2**	**X1**	**X0**	**Y**	**minterm**
	0	0	0	-	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\overline{X_2} \cdot X_1 \cdot X_0$
4	1	0	0	1	$X_2 \cdot \overline{X_1} \cdot \overline{X_0}$
	1	0	1	-	
OFF	1	1	0	0	
ON	1	1	1	1	$X_2 \cdot X_1 \cdot X_0$

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 1 when at least one of the minterms is 1. Therefore, the minterms have to be connected disjunctive:

\begin{align*} Y &= & Y' & \quad + & Y'' & \quad + & Y''' \\ Y &= & (X_0 \cdot X_1 \cdot \overline{X_2}) & \quad + & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & ({X_0} \cdot {X_1} \cdot {X_2}) \\ \end{align*}

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 snytesizing a logic circuit by sum of procucts, all 'don't care' terms outputing 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:

\begin{align*} Y &= & (X_0 \cdot X_1 \cdot \overline{X_2}) & \quad + & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & ({X_0} \cdot {X_1} \cdot {X_2}) & \quad | \text{associative law} \\ Y &= & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & (X_0 \cdot X_1 \cdot \overline{X_2}) & \quad + & ({X_0} \cdot {X_1} \cdot {X_2}) & \quad | \text{associative law } \\ Y &= & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & ((X_0 \cdot X_1) \cdot \overline{X_2}) & \quad + & (({X_0} \cdot {X_1}) \cdot {X_2}) & \quad | \text{distributive law } \\ Y &= & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & ((X_0 \cdot X_1) \cdot (\overline{X_2} + {X_2})) & & & \quad | \text{complementary element} \\ Y &= & (\overline{X_0} \cdot \overline{X_1} \cdot {X_2}) & \quad + & (X_0 \cdot X_1) \\ \end{align*}

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 hiere is the OR-operator.

Simmilarly, 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

Therm-o-Safety						
Input	**X2**	**X1**	**X0**	**Y**	**minterm**	**maxterm**
	0	0	0	-		
1	0	0	1	0		$X_2 + X_1 + \overline{X_0}$
2	0	1	0	0		$X_2 + \overline{X_1} + X_0$
3	0	1	1	1	$\overline{X_2} \cdot X_1 \cdot X_0$	
4	1	0	0	1	$X_2 \cdot \overline{X_1} \cdot \overline{X_0}$	
	1	0	1	-		
OFF	1	1	0	0		$\overline{X_2} + \overline{X_1} + X_0$
ON	1	1	1	1	$X_2 \cdot X_1 \cdot X_0$	

The formulas of figure 8 can again be transformed into gate circiuts (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{align*} Y &= & (\overline{X_0} + X_1 + X_2) & \quad \cdot & ({X_0} + \overline{X_1} + X_2) & \quad \cdot & (\overline{X_0} + \overline{X_1} + X_2) \\ &= & ... \\ Y &= & (\overline{X_0} + X_1 + X_2) & \quad \cdot & (\overline{X_0} + \overline{X_1}) \\ \end{align*}

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 resuls 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 snytesizing a logic circuit by sum of procucts, all 'don't care' terms outputing 1
- The products of sum is the DeMorgan dual of the sum of products **if** there are no don't care terms. Otherwise the resuls cannot be transformed into each other with the means of boolean rules.

3.2 Karnaugh Map

3.2.1 Introduction with example

For our therm-o-safety 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 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 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`

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

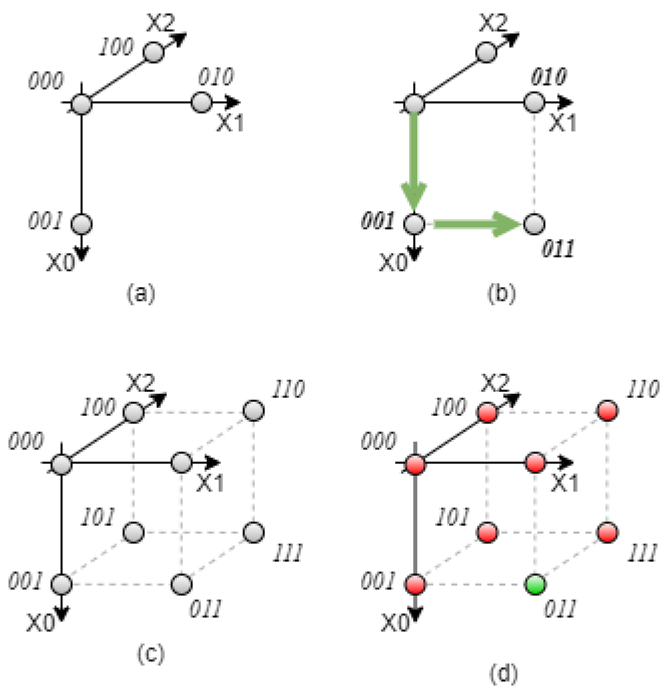


Fig. 12: 3 dimensional cube represetation

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

- The value 1 (independent from the inputs) lead to all positions are 1
- A single input equal 1 (independent from all other inputs, i.e. all others are `don't care`) lead in our example to the edges of a side surface of the cube.
  In out example $\color{green}{X_1=1}$ lead to the situation shown in figure 13 (a). When investigating the shown green dots `0`**1**`0`, `0`**1**`1`, `1`**1**`0`, `1`**1**`1` 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 $\color{green}{X_1=1}$ ($=X_0\cdot X_1$) result in the two edges on a corner of the cube (figure 13 (b)). For this coordinates (`0`**11**, `1`**11**) 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 $\color{green}{X_1=1}$ and $\color{brown}{X_2=0}$ ($=X_0\cdot X_1$) result in the two edges on a corner of the cube (figure 13 (b))

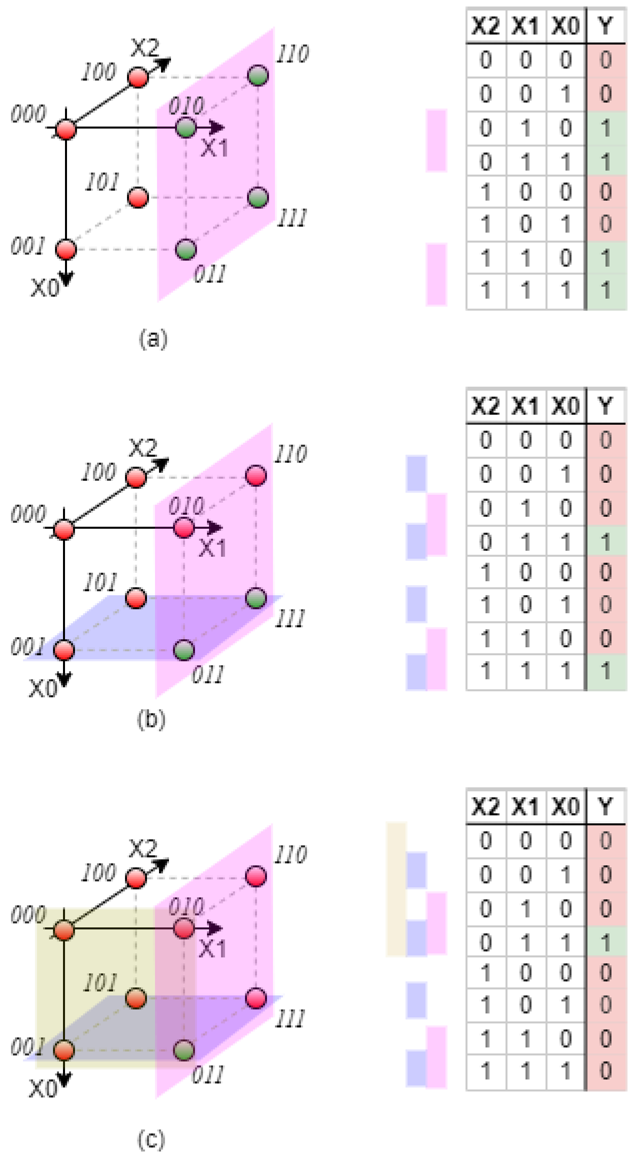


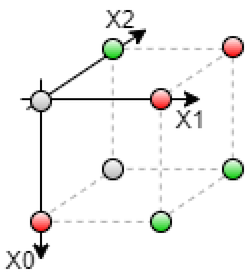
Fig. 13: examples in 3 dimensional cube represetation

With this representation in mind, we can simpilify other representations much more simplier. \\# 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 dimansional 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

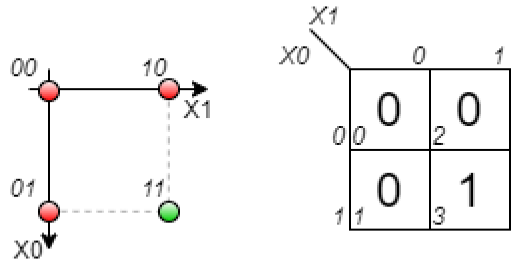


Therefore, we try to find a better way to sketch the coordinates, before we simplify our Therm-o-Safety. We take one step back and look onto a two dimensional and simple example: $Y = X_0 \cdot X_1$. This can be shown as the edges of a square (figure 11 (a)). We will in the future we will write this as in figure 11 (b). 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. In each cell the result is shown as larger number. The small number is the decimal representation of the number given by X_1 and X_0. This number is often not explicitely shown in the karnaugh map, but simplifies the fill-in of the map.

$X_0 \cdot X_1$

dec	X1	X0	Y
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

(a)



(b) (c)

Fig. 11: two dimensional karnaugh map

For the three dimensional karnaugh map it is a good idea “unwrap” the cube. This can be done as shown in figure 15.

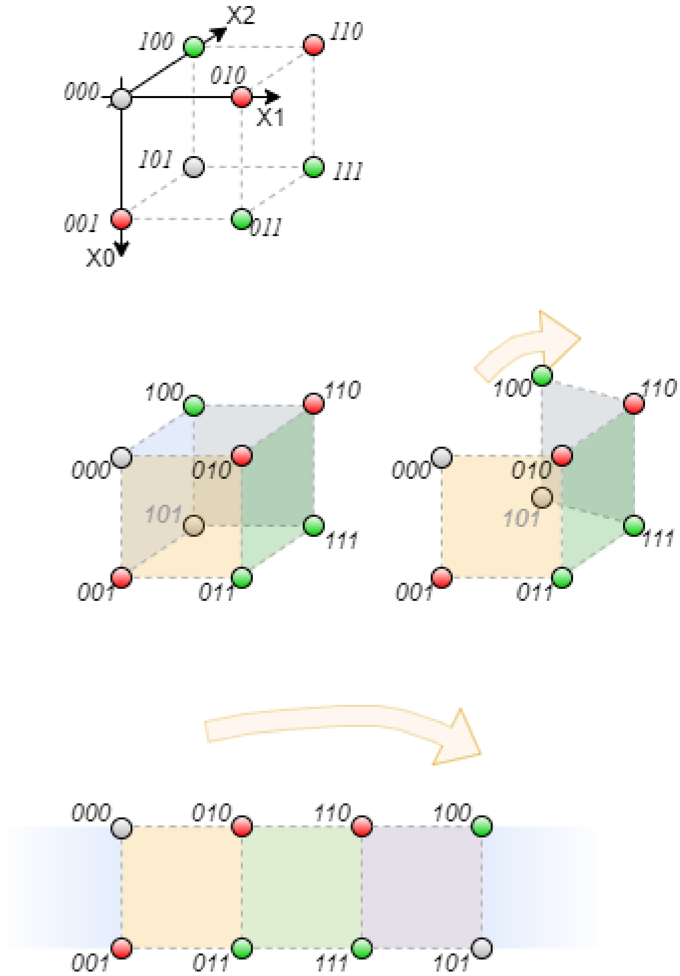
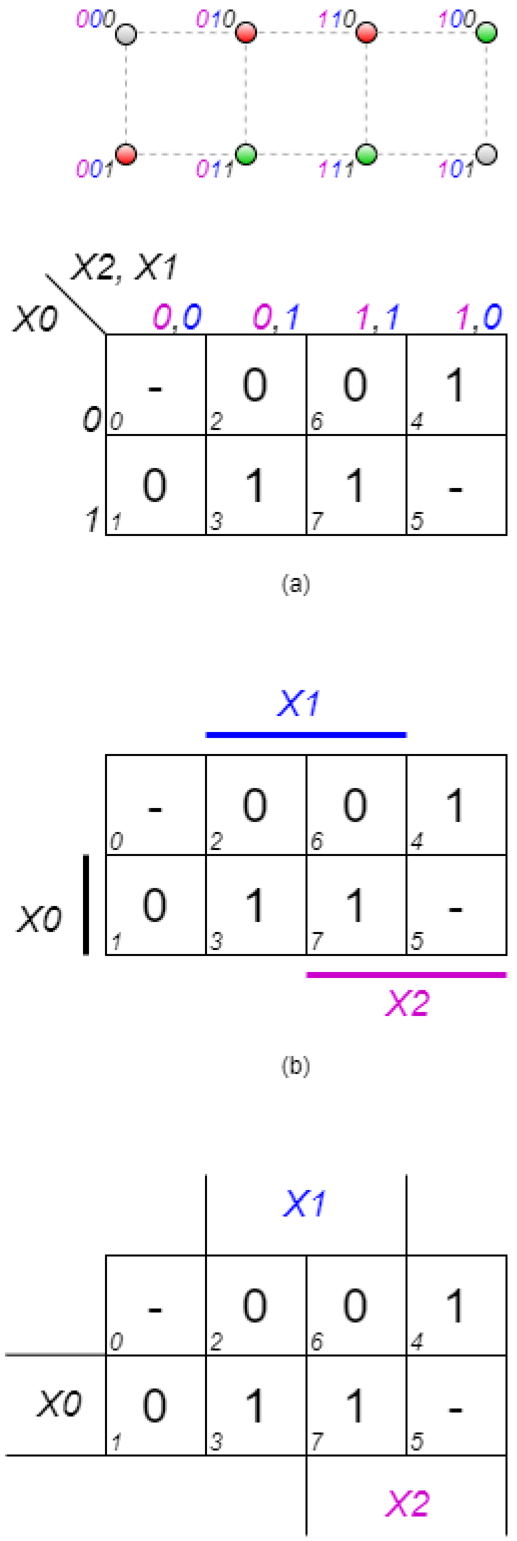


Fig. 15: three dimensional karnaugh map

Out of the flattened cube we can derive the three dimensional karnaugh map (see figure 16). One way is to show the variable names of the dimensions in the corner. Another way is to mark the columns / rows related to the dimension with a line. In figure 16 the dimensions are additionally marked with colors.



(c)

Fig. 16: three dimensional karnaugh map

interactive example

Exercise 3.1.x Further Querstions

1. compare the results with the output given here (the output y can be changed by clicking onto it)

From:
https://mexle.te.hs-heilbronn.de/ - **MEXLE Wiki**

Permanent link:
https://mexle.te.hs-heilbronn.de/introduction_to_digital_systems/combinatorial_logic?rev=1633826502

Last update: **2021/10/10 02:41**