ELECTRICAL & COMPUTER ENGINEERING

Lab 3: BOOLEAN ALGEBRA AND K-MAPS

The University of Washington | The Remote Hub Lab | Last Revised: March 2022

Summary

This lab introduces the concept of Karnaugh Maps (K-Maps), a graphic organizer method which converts Truth Tables into Boolean algebra equations or expressions. In this lab you will learn about Boolean algebra, K-Maps, and practice with an application on LabsLand.

Table of Contents

What is Boolean Algebra?

Boolean Algebra Background

Operators are the building blocks of digital logic because they relate input signals to the appropriate output signal(s). Recall from Lab 1 that examples of these operators or hardware are AND, OR, NOR, XOR, NAND, and MUX. While the “word” form of these operators is helpful for understanding their meaning, there is a simpler way to articulate the relationship between signals. To express all the information in a Truth Table in a concise way, Boolean algebra uses regular arithmetic symbols like ‘+’ to represent OR and ‘•’ to represent AND

Lab 1 provided a summary of operators and gate diagrams, depicted here by Figure 1.

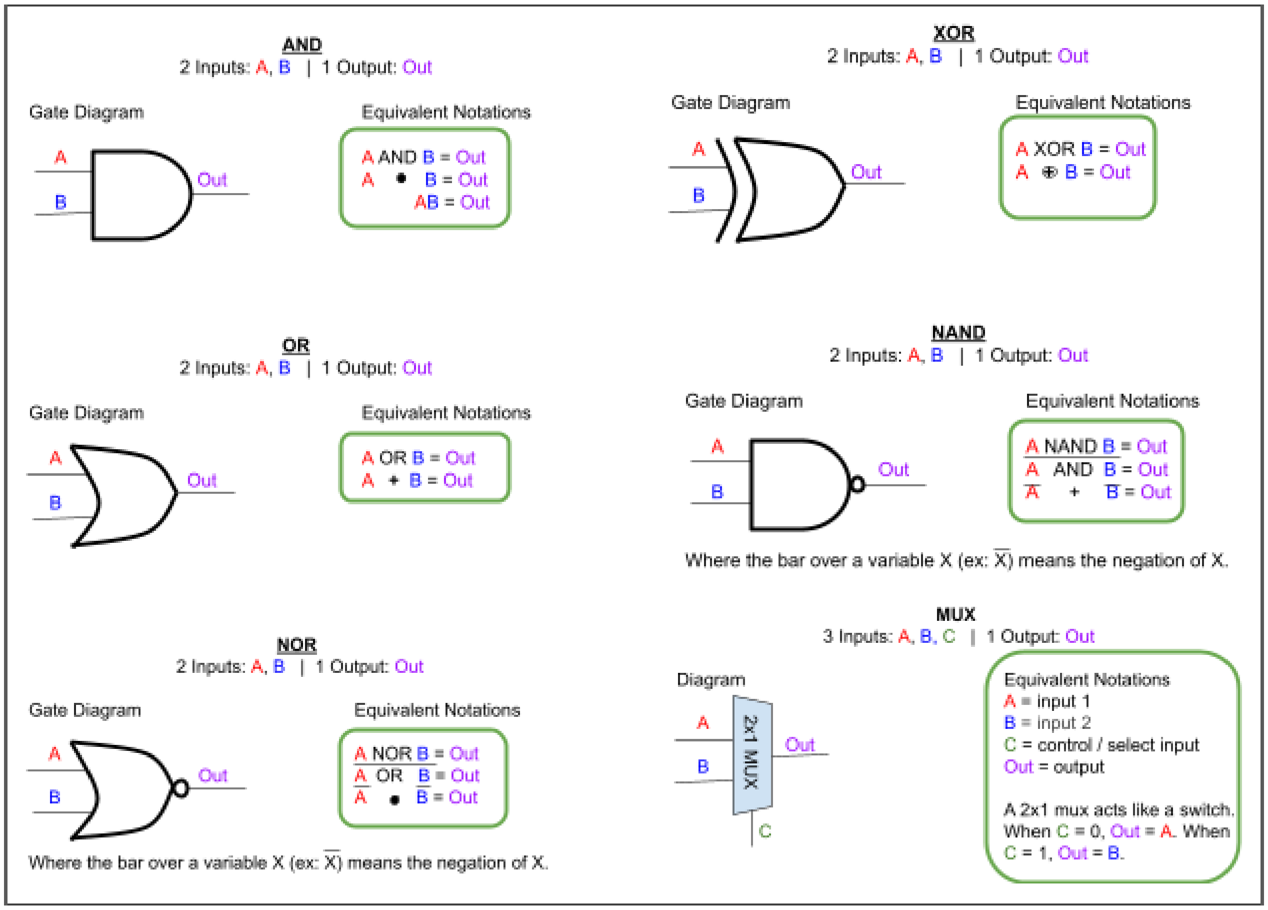

AND
2 Inputs: A, B | 1 Output: Out
Gate Diagram
Equivalent Notations
A AND B = Out
A • B = Out
AB = Out
XOR
2 Inputs: A, B | 1 Output: Out
A XOR B = Out
A ⊕ B = Out
OR
2 Inputs: A, B | 1 Output: Out
A OR B = Out
A + B = Out
NAND
2 Inputs: A, B | 1 Output: Out
$\overline{A \text{ NAND } B}$ = Out
$\overline{A}$ AND $\overline{B}$ = Out
$\overline{A}$ + $\overline{B}$ = Out
Where the bar over a variable X (ex: $\overline{X}$) means the negation of X.
MUX
3 Inputs: A, B, C | 1 Output: Out
Diagram
2x1 MUX
Equivalent Notations
A = input 1
B = input 2
C = control / select input
Out = output
A 2x1 mux acts like a switch. When C = 0, Out = A. When C = 1, Out = B.
NOR
2 Inputs: A, B | 1 Output: Out
$\overline{A \text{ NOR } B}$ = Out
$\overline{A}$ OR $\overline{B}$ = Out
$\overline{A}$ • $\overline{B}$ = Out
Where the bar over a variable X (ex: $\overline{X}$) means the negation of X.

Figure 1: Operators and Gate Diagrams

The right-hand columns labeled “Equivalent Notations” and boxed in **green** are examples of Boolean Algebra expressions. Recall that these gates are physical components in electronic circuits, so it is ideal that we minimize the complexity of gates in our operations. In particular, Boolean Algebra can help us reduce the number of inputs to the gates and the number of gates.

Rules to Remember

Figures 2-3 list some important Boolean algebra identities and laws, along with an explanation for how to think about the relationship. Recall that a variable with a bar over it means the "negation" or the "opposite" of the variable. For example, $\overline{X}$ means 'not X', so if X is true, $\overline{X}$ is false. Conversely, if X is false, $\overline{X}$ is true. Key identities that may be new to you are highlighted in green. Notice that many identities and laws are similar to the rules of regular algebra.

"OR" (+) Identities

⟶ *When is (X OR 0) true?* **Ans**: 0 can never be true. Answer is only when **X** is true.

$$X + 0 = X$$

⟶ *When is (X OR True) true?* **Ans**: 1 is always true no matter X. Answer is **1** (all the time).

$$X + 1 = 1$$

⟶ *When is (X OR X) true?* **Ans**: We have the same input signal. Answer is only when **X** is true.

$$X + X = X$$

⟶ *When is (X OR $\overline{X}$) true?* **Ans**: X and not X are the only possibilities, so at least one of them is true all the time, so the answer is **1**.

$$X + \overline{X} = 1$$

"AND" (•) Identities

⟶ *When is (X AND True) true?* **Ans**: 1 is always true, so the answer depends only on **X**.

$$X \bullet 1 = X$$

⟶ *When is (X AND False) true?* **Ans**: **0** is always false, no matter X.

$$X \bullet 0 = 0$$

⟶ *When is (X AND X) true?* **Ans**: Same input signal, so answer is only when **X** is true.

$$X \bullet X = XX = X$$

⟶ *When is (X OR $\overline{X}$) true?* **Ans**: X and not X are the only possibilities of X, so both cannot be true at the same time. The answer is **0**.

$$X \bullet \overline{X} = X\overline{X} = 0$$

Figure 2: Boolean Algebra Identities

Laws

Commutative Laws:

→ *Comparing when X OR Y is true is the same as comparing when Y OR X is true.*

$$X + Y = Y + X$$

→ *Comparing when X AND Y is true is the same as comparing when Y AND X is true.*

$$XY = YX$$

Associative Laws:

→ *Set A = (Y OR Z) = (Y+Z). Set B = (X OR Y) = (X+Y). (X OR A) is the same as (B OR Z).*

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

→ *Set A = YZ. Set B = XY. (X AND A) is the same as (B AND Z).*

$$X(YZ) = (XY)Z$$

DeMorgan's Laws:

→ *Negating an entire operation = negating each signal and changing to the opposite operator.*

$$\overline{(X + Y)} = \overline{X} \bullet \overline{Y} \qquad \overline{(X \bullet Y)} = \overline{X} + \overline{Y}$$

Absorption Laws:

- $X + XY = X(1 + Y) = X(1) = X$

$$X + XY = X$$

- $XY + X\overline{Y} = X(Y + \overline{Y}) = X(1) = X$

$$XY + X\overline{Y} = X$$

- $X + \overline{X}Y = X + XY + \overline{X}Y = X + Y(X + \overline{X}) = X + Y(1) = X + Y$

$$X + \overline{X}Y = X + Y$$

- $X(X + Y) = XX + XY = X + XY = X(1 + Y) = X(1) = X.$

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

- $(X + Y)(X + \overline{Y}) = XX + X\overline{Y} + XY + Y\overline{Y} = X + X\overline{Y} + XY + 0 = X + XY = X$

$$(X + Y)(X + \overline{Y}) = X$$

- $(X)(\overline{X} + Y) = X\overline{X} + XY = 0 + XY = XY$

$$X(\overline{X} + Y) = XY$$

Distributive Laws:

→ *The AND operator distributes over OR'ed quantities.*

$$X(Y + Z) = XY + XZ$$

→ *(X+Y)(X+Z) = XX + XZ + YX + YZ = X + XZ + XY + YZ = X(1+Z+Y) + YZ = X + YZ*

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

Figure 3: Boolean Algebra Laws

What is the simplest form of the expression $(A + C)(AD + A\overline{D}) + AC$? Use the identities and laws in Figures 2-3. (To see examples, refer to the “Absorption Laws” in Figure 3.)

What is a K-Map?

Karnaugh Map Background

Karnaugh Maps, or K-Maps, are another way of simplifying Boolean expressions. Using a grid-type chart, K-Maps allow you to arrive at a simplified Boolean expression by eliminating redundant variables. The size of the chart depends on the number of inputs, as shown in Figure 4. To understand how K-Maps work, look at the 2-input, 3-input, and 4-input cases in Figure 4.

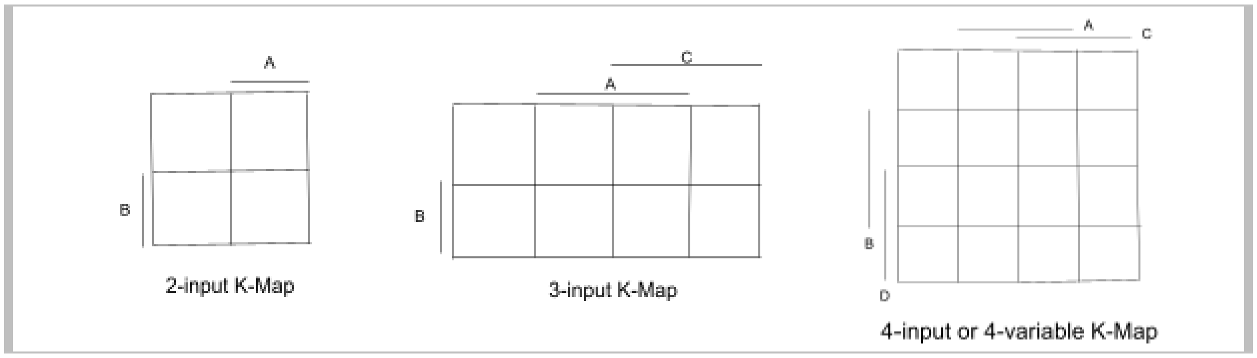


Figure 4: Blank K-Maps

All these K-Map types involve smaller grid squares. Each square represents the output value for an input combination (and thus a row on a Truth Table). For example, imagine a 2-input K-Map. It has labels for its variables written on the perimeter of the chart and marked with a bar for the relevant columns and rows. These labels designate when the inputs are true. Take the K-Map with sample data in Figure 5, assuming the inputs are called ‘A’ and ‘B’.

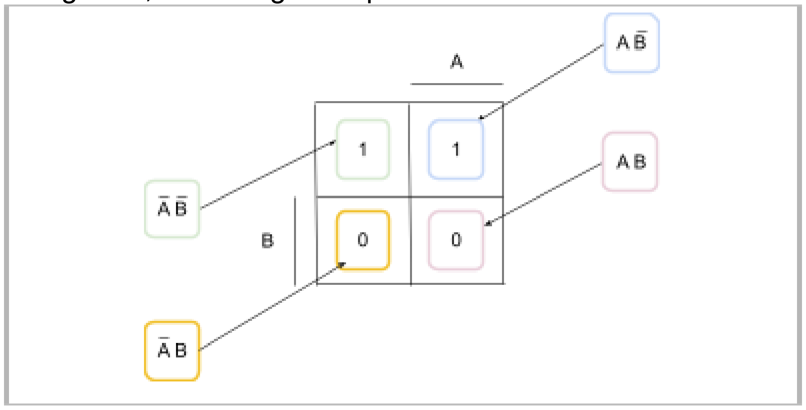


Figure 5: K-Map with Sample Data

Notice the labels for two inputs, A and B, mark a column and a row respectively. This means the squares under the column labeled A denote when A is true. The left-hand column signifies when A is false. Moreover, the squares in the row aligned with the label B designate when B is true, and the top row designates when B is false. If the output is ‘true’ when inputs A and B are on, then the corresponding box for ‘AB’ will contain a 1. In this K-Map, AB is false, so the pink square in the bottom right corner has a 0.

The 3-input and 4-input cases follow the same reasoning. Because each square in a K-Map represents the output of an input combination, we can derive a Truth Table from a K-Map and vice versa. For accuracy, you need to “fill in” a K-Map in a certain order; that’s where binary numbers come in. Observe the numbers in the bottom left of the grid squares in each K-Map in Figure 6; they correspond to specific rows (and thus input combinations) on a Truth Table.

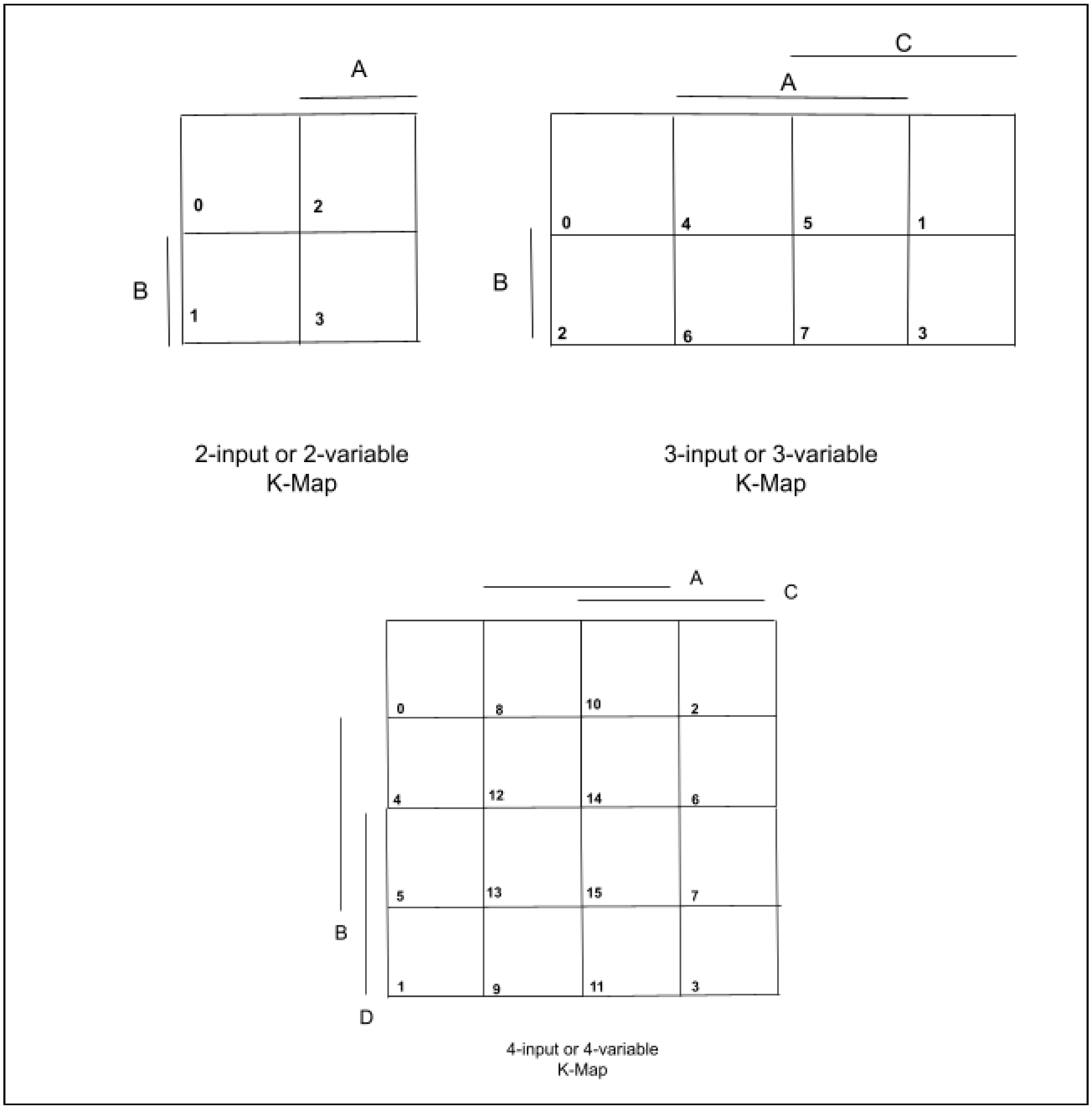
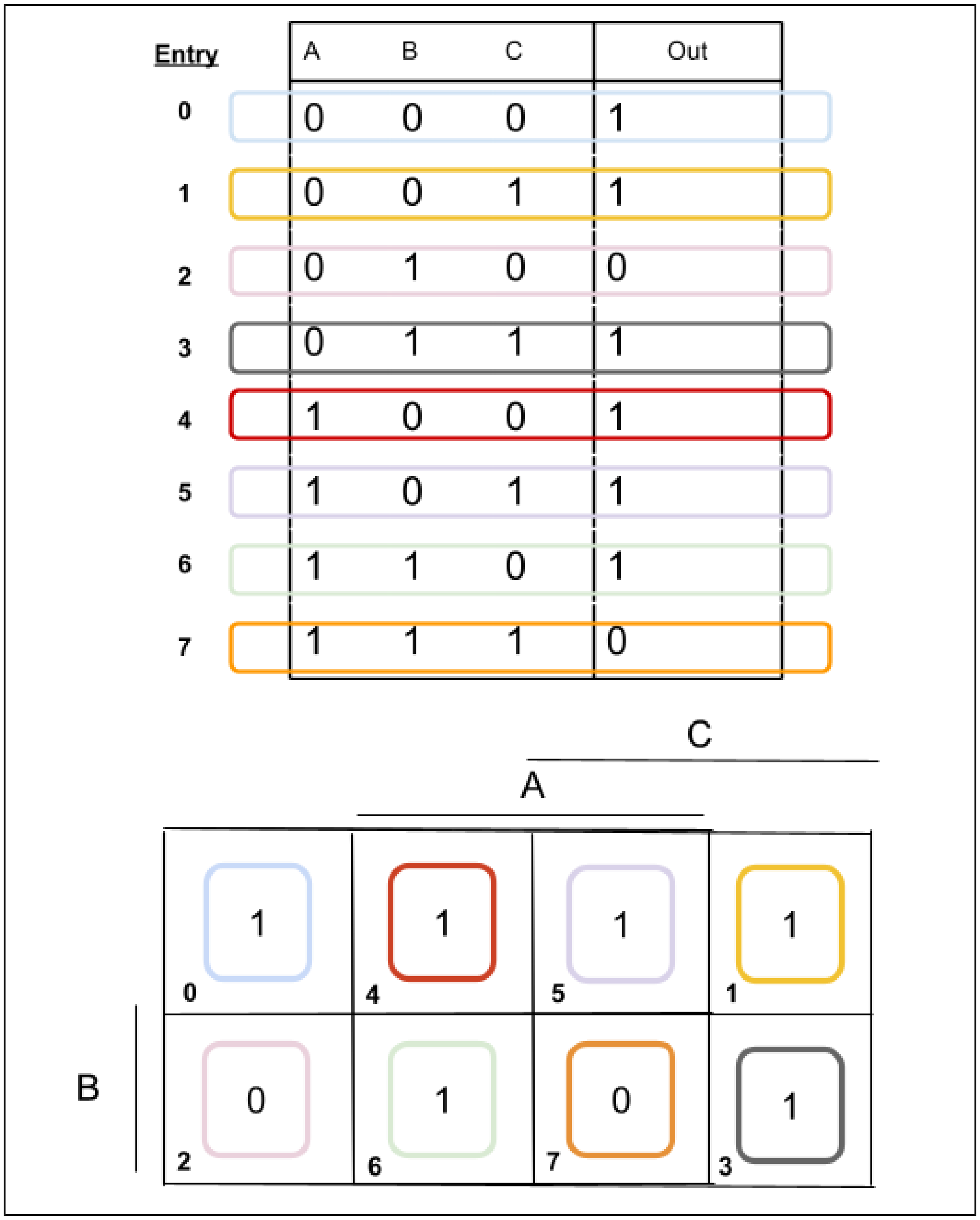

A
B
0
2
1
3
2-input or 2-variable K-Map
C
A
B
0
4
5
1
2
6
7
3
3-input or 3-variable K-Map
A
C
B
D
0
8
10
2
4
12
14
6
5
13
15
7
1
9
11
3
4-input or 4-variable K-Map

Figure 6: Input Combinations Mapped to K-Map Grid Squares

As an example, let’s look at the Truth Table and corresponding K-Map in Figure 7.



Entry	A	B	C	Out
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

	1 0	1 4	1 5	1 1
B	0 2	1 6	0 7	1 3

Figure 7: Truth Table to K-Map Conversion

Notice how the 3-inputs map to labels around the perimeter of the K-Map. Moreover, notice how the nth entry of the Truth Table corresponds to the grid square label in the bottom left corner. This illustrates how, when translating a Truth Table to a K-Map, you should carefully transfer the outputs to the appropriate grid square. This process also applies to translating a K-Map to a Truth Table.

Simplifying K-Maps

Now that we know how to understand and write K-Maps from Truth Tables and vice versa, how can we simplify the Boolean algebra? With 2-input, 3-input, and 4-input K-Maps, the trick is to look for groups of adjacent 1s in quantities of one, two, four, eight, or sixteen and group them together. The goal is to maximize the number of 1s in a group, but minimize the number of groups we make. Only the squares encompassed by a group will be included in the expression.

A 2-input example is shown in Figure 8. Notice that the output contains the expression that best describes the group of 1s. In this case, we need an expression for the top row. We notice that the blue group is the region not included by B. As a result, the output of this system is $\overline{B}$ according to the K-Map.

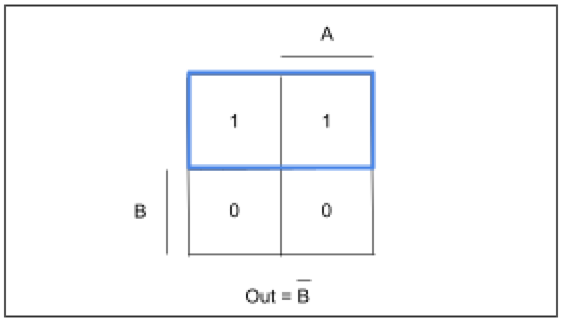


Figure 8: Forming Groups on K-Maps (2-Input Example)

Figure 9 depicts another example with a 3-input K-Map. Note that the simplest expressions may have a group of four which can "go off the grid." This can happen from right-to-left (as shown in Figure 9) or top-to-bottom. Because the bottom left corner 1 and top right corner 1 were already covered by the group of four, we only need to cover the two 1s highlighted in pink. But, we notice that there are adjacent 1s to these lone, uncovered pink 1s. It is important to note that our goal is to simplify, so we would rather have 2-variable terms rather than 3-variable terms. As a result, we allow overlap so lone, uncovered 1s can join a group of two, four, eight, or sixteen.

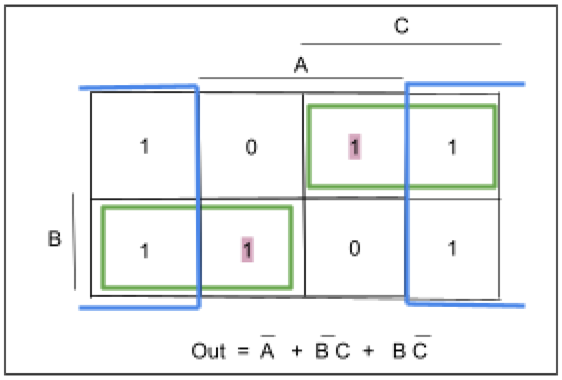
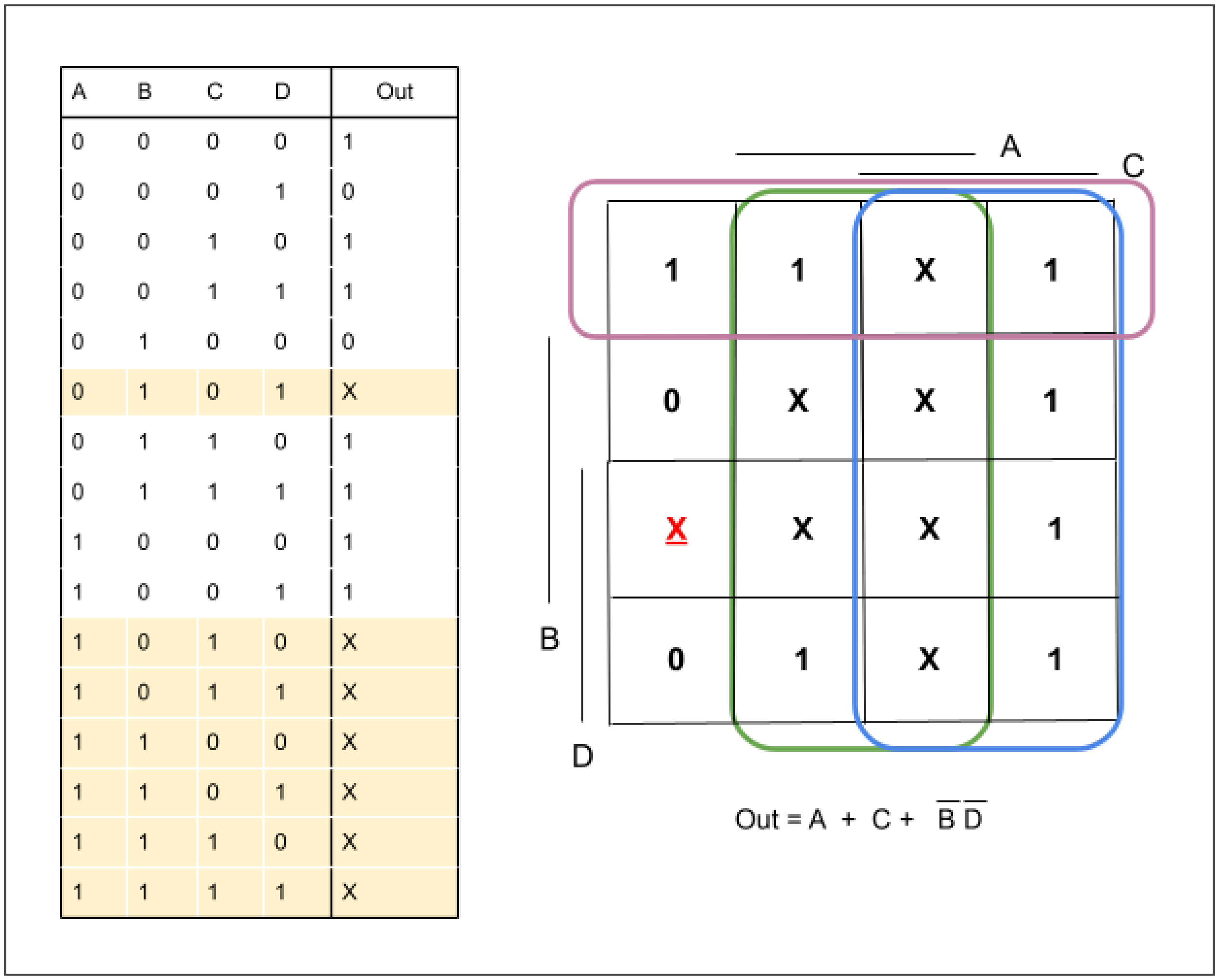


Figure 9: Forming Groups on K-Maps (3-Input Example)

The result is the sum of all possible groups. In this case, we found three groups so there are three terms added together - the "OR"-ing of all possibilities.

Sometimes, there are Truth Tables (and by extension, K-Maps) with input combinations which are not relevant to the system. For example, assume we have a 4-input Truth Table which has defined behavior for all input combinations except for 5, 10, 11, 12, 13, 14, and 15. In this case, we should put X's in the output column(s) of these input combinations which are irrelevant to the system. One of these X's is called a "Don't Care." This Truth Table example and how to simplify its K-Map is shown in Figure 10.



A	B	C	D	Out
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	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Out = A + C + $\overline{B}\,\overline{D}$

Figure 10: Truth Table with Don't Cares and K-Map Simplification

In Figure 10's Truth Table, notice that we "don't care" about the input combinations which are highlighted in yellow, so the output column is marked with X's. These Don't Cares can be treated as '1s' or '0s' if it helps us maximize the number of 1s in a group, but minimize the number of groups we make. This simplification is seen in Figure 10's K-Map. All Don't Cares in the middle two columns can help create non-redundant groups of eight 1s. Consequently, all Don't Cares are assumed to be '1' except for the Don't Care in red underlined.

STEP 1: Draw the K-Map.

1. *Draw a 2x2 square grid (2-input system), a 4x2 square grid (3-input system), or a 4x4 square grid (4-input system).*
2. *Fill in the smaller squares with the outputs from the Truth Table.*

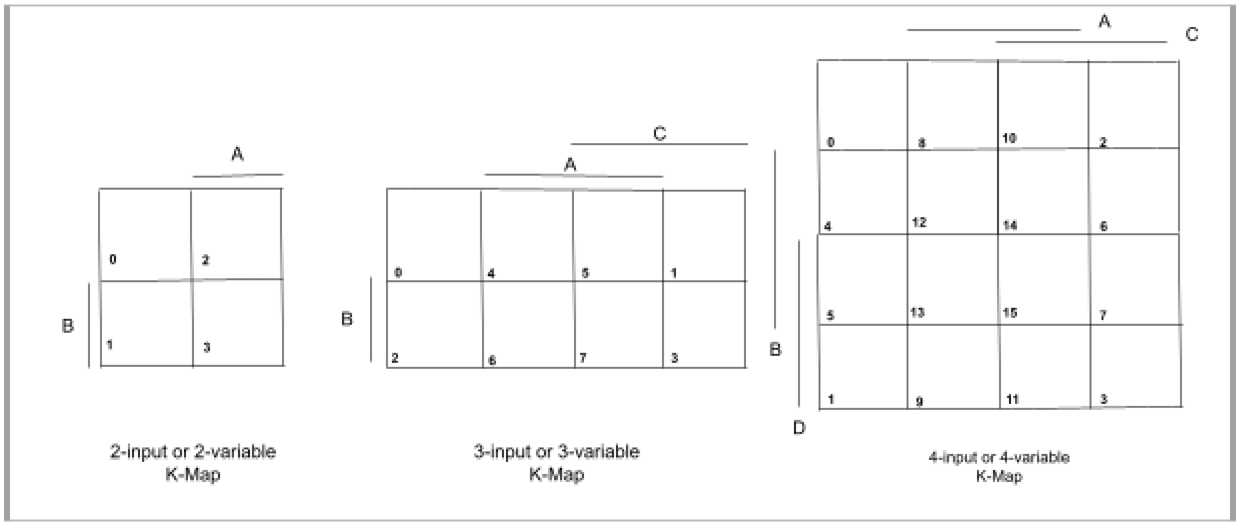

A
B
0
2
1
3
2-input or 2-variable K-Map
C
A
B
0
4
5
1
2
6
7
3
3-input or 3-variable K-Map
A
C
B
D
0
8
10
2
4
12
14
6
5
13
15
7
1
9
11
3
4-input or 4-variable K-Map

Figure 11: Input Combinations Mapped to K-Map Grid Squares

STEP 2: Form groups according to the following rules:

1. *The goal is to form groups of adjacent 1s.*
2. *Groups must take a square or rectangular shape (no diagonals).*
3. *Groups of 1s can be a power of two in size: one, two, four, eight, or sixteen.*
4. *Maximize the number of 1s in each group, and minimize the number of groups you make. (This may involve Don't Care's.)*

A group is "maximized" when it includes all possible adjacent 1s. The example in Figure 12 shows a maximized group of four vs. two non-maximized pairs. The one on the left is more correct because it maximizes the number of 1s in a group (there are four 1s) and minimizes the number of groups (there is only one group instead of two). Note that going "off the grid" or off the sides of the grid is legal!

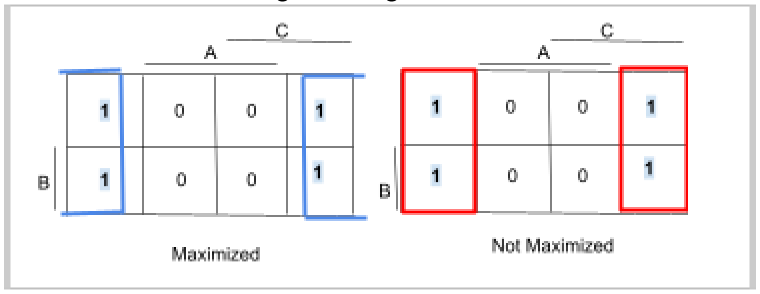


Figure 12: Maximized vs. Non-Maximized K-Maps

5. *Groups can overlap, but groups cannot be fully redundant; only make groups if there is a 1 that is uncovered/ungrouped.*

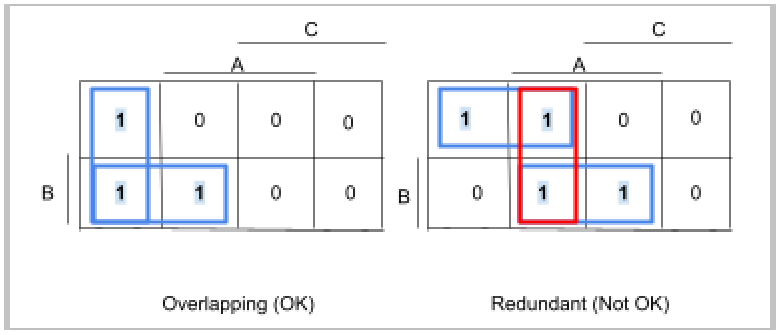


Figure 13: Overlapping vs. Redundant K-Map Groups

6. *You are "done" finding groups when every 1 is part of a maximized group*.

   In Figure 13, notice that, after finding a pair of ones in the left K-Map (say the pair in the bottom row), there is still a 1 above without a group. Remember that we want to maximize the number of ones in a group and minimize the number of groups, so we would rather have two groups of two 1s rather than one pair and a single 1. As a result, we group the remaining 1 (in the $\overline{A}\,\overline{B}\,\overline{C}$ position) with the 1 below it. So, we get two groups which overlap.

STEP 3: Translate into a Boolean expression.

1. *Name the groups you found using the labels on the K-Map perimeter.*

   In the 2-input K-Map, notice that the label 'A' is aligned with the right column and the label 'B' is aligned with the bottom row. Similarly, in the 3-input K-Map, notice that the label 'A' is aligned with the middle four-square region, 'C' is aligned with the right four-square region, and 'B' is aligned with the bottom row.

   When naming a group, locate which region(s) contain it and which region(s) do not contain it. For example, look at the pair of ones in the bottom row of the left K-Map in Figure 13. Notice that this group is not contained by C but is contained by B. However, one element of this group is contained by A but the other element is not contained by A. Therefore, we only need B and C to describe the group, and we can name it $B\overline{C}$ because it is NOT in C and it is in B. Following similar reasoning, the vertical pair in the leftmost column is not contained by A, is not contained by C, and is split over B. Therefore, this group is called $\overline{A}\,\overline{C}$.

2. *Add the terms together to arrive at the simplified result.*

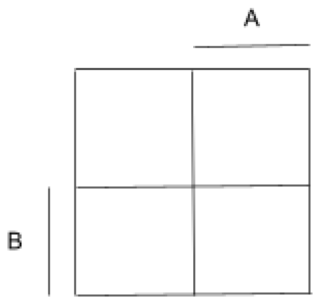
   This works because you are essentially "OR-ing" each possible group together. The simplified result for the left (correct) K-Map in Figure 13 is $Out\ =\ B\overline{C}\ +\overline{A}\,\overline{C}$.

Practice: Truth Table to K-Map to Boolean Expression

Using the given Truth Tables, draw the corresponding K-Maps and determine the Boolean expression.

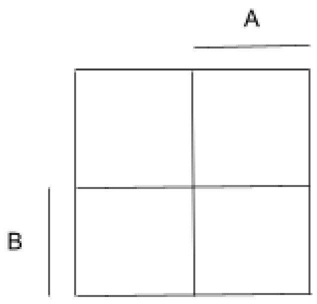
2-input K-Maps

A	B	Out
0	0	0
0	1	0
1	0	1
1	1	0



Out = ____________________

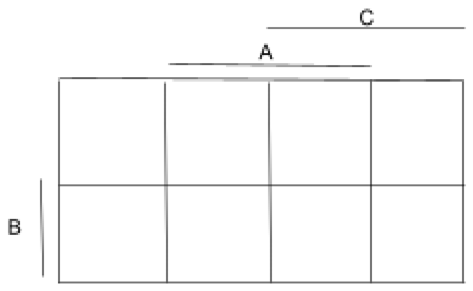
A	B	C
0	0	1
0	1	1
1	0	0
1	1	0



Out = ____________________

3-input K-Maps

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Out = ____________________

K-Maps with the Digital Trainer

For each challenge corresponding to the Digital Trainer Activity, do the following:

1. Fill in the Truth Table. You can use your answers from Lab 1 or Lab 2.
2. Use the Truth Table to make a K-Map.
3. Use the K-Map to arrive at a Boolean Expression/Equation.
4. Check that the Boolean Equation you arrive at matches the correct operator you found in Lab 1. Remember that there are multiple equivalent representations of the same Boolean relationship. For example, $\overline{(X + Y)} = \overline{X} \bullet \overline{Y}$ and $\overline{(X \bullet Y)} = \overline{X} + \overline{Y}$.

Table 1: Digital Trainer Challenge 1

SW1	SW0	LEDout

Boolean Equation: ____________________________

Table 2: Digital Trainer Challenge 2

SW1	SW0	LEDout

Boolean Equation: ____________________________

Table 3: Digital Trainer Challenge 3

SW1	SW0	LEDout

Boolean Equation: ____________________________

Table 4: Digital Trainer Challenge 4

SW1	SW0	LEDout

Boolean Equation: ______________________

Table 5: Digital Trainer Challenge 5

SW1	SW0	LEDout

Boolean Equation: ______________________

Table 6: Digital Trainer Challenge 6

SW2	SW1	SW0	LEDout

Boolean Equation: ______________________

Table 7: Digital Trainer Challenge 7

SW2 SW1 SW0	LEDout

Boolean Equation: ____________________________

Table 8: Digital Trainer Challenge 8

SW2 SW1 SW0	LEDout

Boolean Equation: ____________________________

Table 9: Digital Trainer Challenge 9

SW2	SW1	SW0	LEDout

Boolean Equation: ____________________________

Table 10: Digital Trainer Challenge 10

SW2	SW1	SW0	LEDout

Boolean Equation: ____________________________

Exercise on LabsLand

Use the Boole-Web feature on LabsLand to see the whole I/O → Truth Table → K-Map process, and verify your answers to Challenges 1-10 from the Digital Trainer Activity. Follow the process detailed in this section to access Boole-Web, and repeat steps 4-10 for each of the challenges.

1. On your LabsLand dashboard, navigate to the “Boole” feature, and click on “Access this lab.”

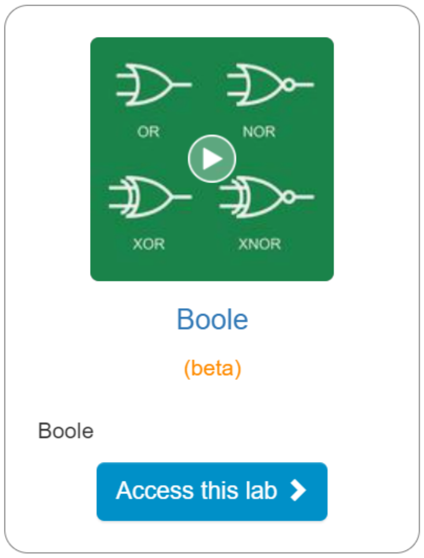


Figure 14: Boole Feature

2. Click on “Access” under “Boole IDE.”

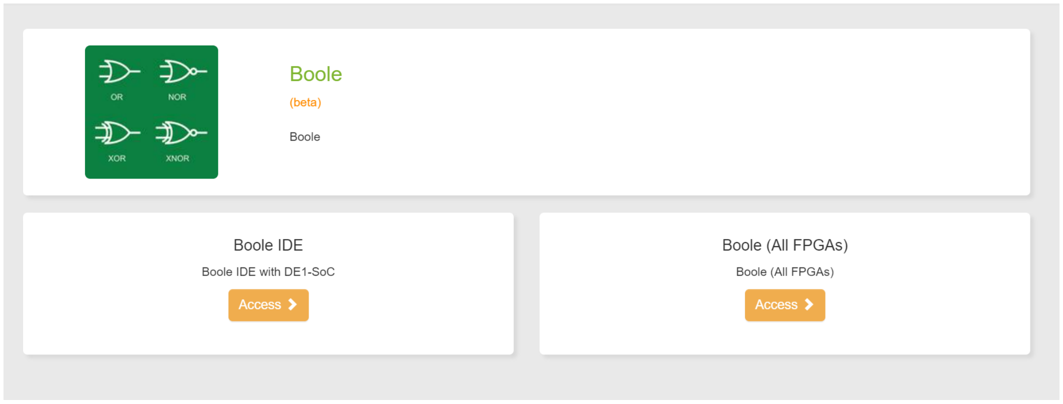
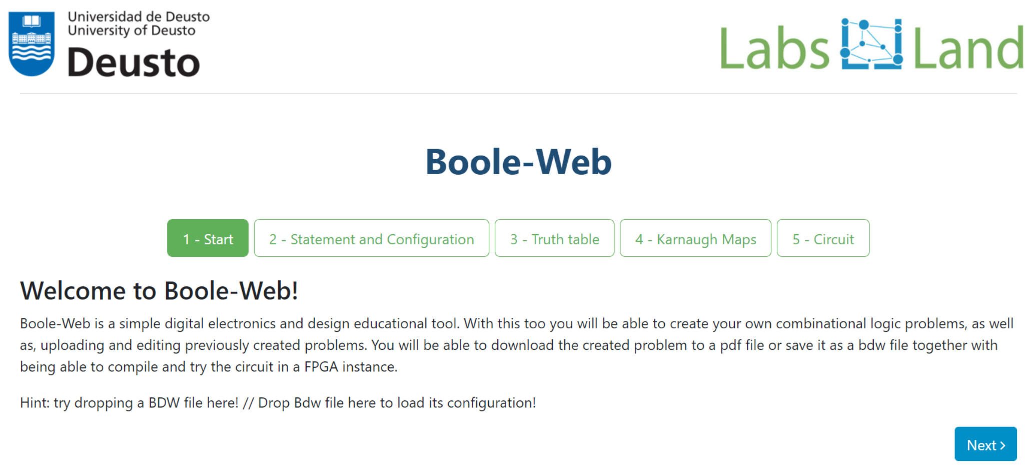


Figure 15: Boole IDE

3. You will see the Boole-Web welcome page as shown in Figure 16. Read the intro, and press “Next.” You should now see the screen shown in Figure 17.


Universidad de Deusto
University of Deusto
Deusto

LabsLand

Boole-Web

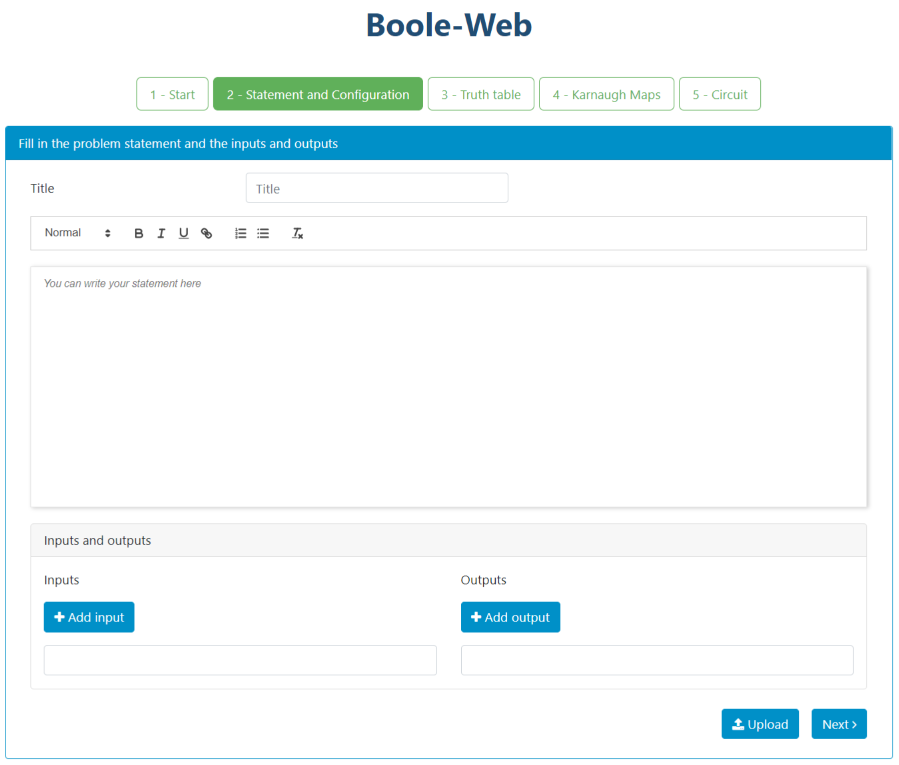
1 - Start | 2 - Statement and Configuration | 3 - Truth table | 4 - Karnaugh Maps | 5 - Circuit

Welcome to Boole-Web!

Boole-Web is a simple digital electronics and design educational tool. With this too you will be able to create your own combinational logic problems, as well as, uploading and editing previously created problems. You will be able to download the created problem to a pdf file or save it as a bdw file together with being able to compile and try the circuit in a FPGA instance.

Hint: try dropping a BDW file here! // Drop Bdw file here to load its configuration!

Figure 16: Boole-Web Welcome Page


Boole-Web

1 - Start | 2 - Statement and Configuration | 3 - Truth table | 4 - Karnaugh Maps | 5 - Circuit

Fill in the problem statement and the inputs and outputs

Title: Title

You can write your statement here

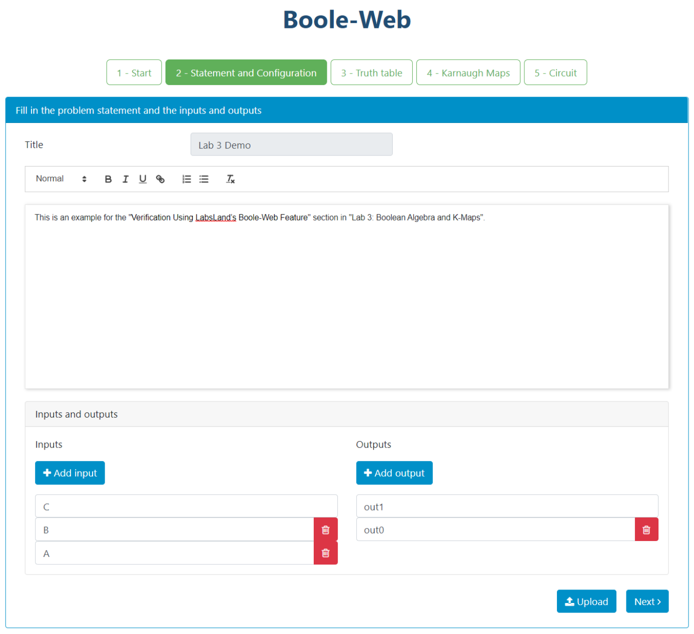
Inputs and outputs

Inputs | Outputs

Figure 17: Blank Statement and Configuration Page

4. Add a project title and description in the Statement and Configuration Page (Figure 17). This is for documentation purposes.

5. Add the inputs and outputs of your system. For example, given a 3-input system with inputs ‘A’, ‘B’, and ‘C’ and 2 outputs ‘Out1’ and ‘Out0’, click ‘Add input’ twice and ‘Add output’ once. Enter the appropriate input and output names in each blank. Ensure that the topmost name is the rightmost variable in your TruthTable; in this case, that is input C and output out1. Press ‘Next’ when finished, as depicted in Figure 18.



Title: Lab 3 Demo

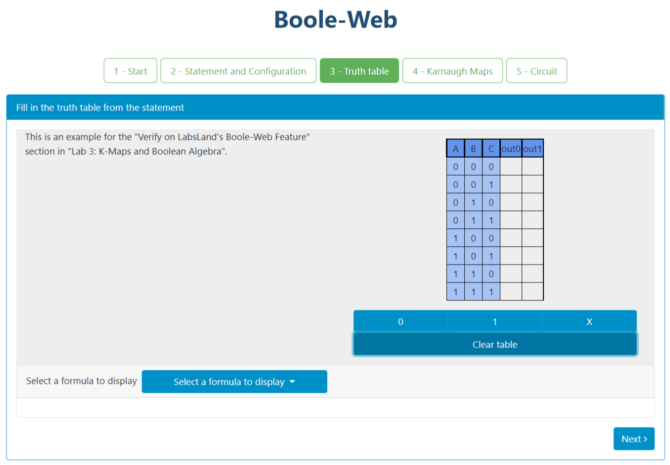
This is an example for the "Verification Using LabsLand's Boole-Web Feature" section in "Lab 3: Boolean Algebra and K-Maps".

Inputs: C, B, A

Outputs: out1, out0

Figure 18: Statement and Configuration Page with Data

6. You should now see your project description in the top left with a K-Map missing its outputs in the top right. Note how the rows of the K-Map have its inputs auto-filled.


Boole-Web
1 - Start
2 - Statement and Configuration
3 - Truth table
4 - Karnaugh Maps
5 - Circuit
Fill in the truth table from the statement
0
1
X
Clear table
Select a formula to display
Select a formula to display
Next ›

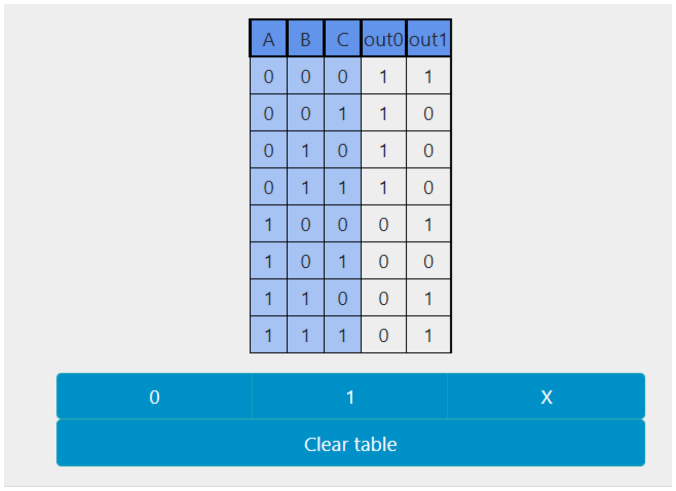
This is an example for the "Verify on LabsLand's Boole-Web Feature" section in "Lab 3: K-Maps and Boolean Algebra".

A	B	C	out0	out1
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Figure 19: Statement and Configuration Page with Data

7. Fill in out0 and out1 according to your observed I/O behaviors. notice the shortcuts below the Truth Table: ‘Clear Table’, ‘0’ (fill all remaining output rows with 0) , ‘1’ (fill all remaining output rows with 1) , ‘X’ (fill all remaining output rows with X which means “Don’t Care” - something we’ll learn about in Lab 5).

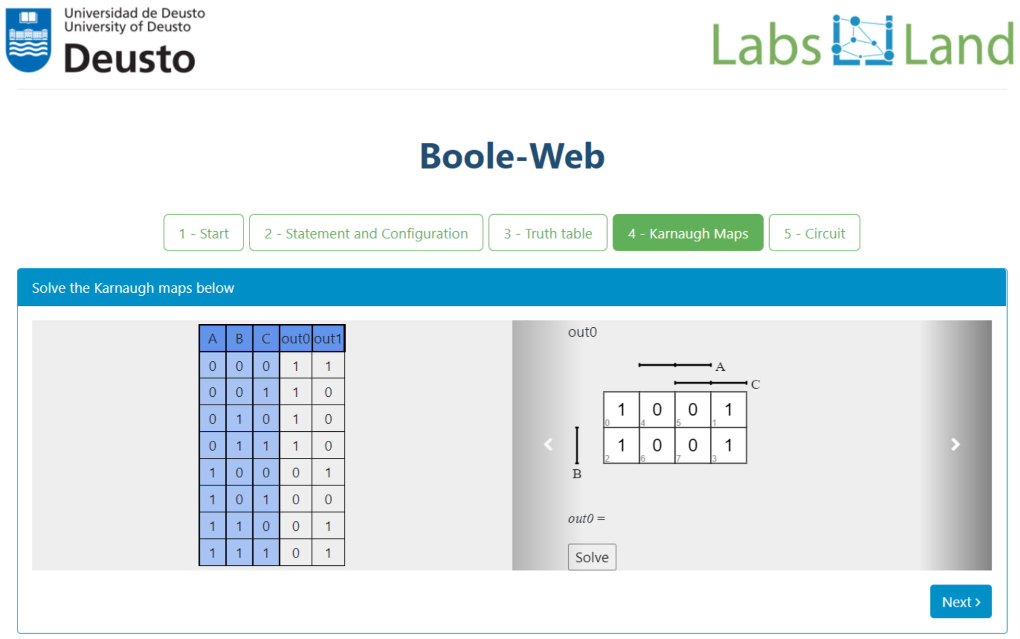
   A Truth Table may look like the example in Figure 20. Press ‘Next’ when finished.



A	B	C	out0	out1
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	1
1	1	1	0	1

Figure 20: Boole-Web Truth Table with Data

8. You should see a page as shown in Figure 21 with the K-Maps drawn for each of your outputs. Press the left and right arrows highlighted in gray to scroll through the K-Maps.

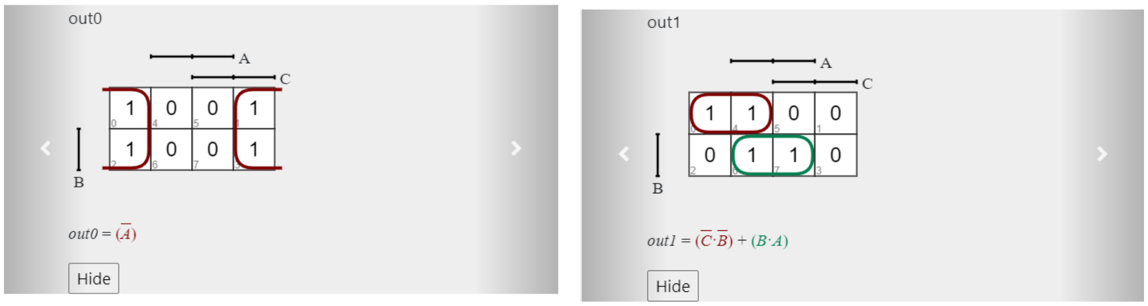


A	B	C	out0	out1
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	1
1	1	1	0	1

1	0	0	1
1	0	0	1

Figure 21: Boole-Web K-Map Page

9. Press 'Solve' to see the resulting Boolean Expressions from each K-Map. Boole-Web solves the grouping and simplification for you!

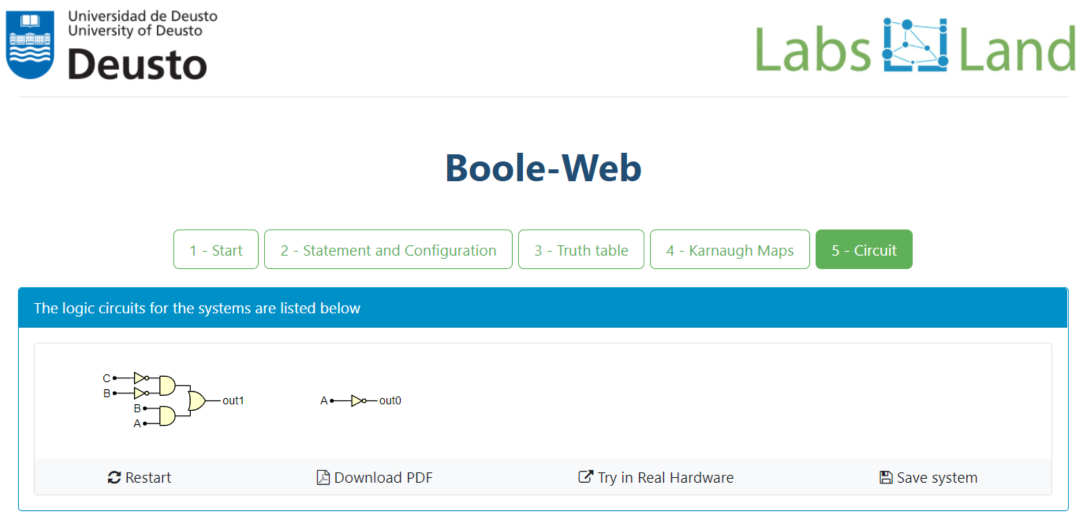


$out0 = (\overline{A})$

$out1 = (\overline{C}\cdot\overline{B}) + (B\cdot A)$

Figure 22: Solved K-Maps in Boole-Web

10. To see the schematic of gates which are built according to the Boolean expressions found by Boole-Web, press "Next." You should see the screen as shown in Figure 22 with a combination of logic gates for your system.



The logic circuits for the systems are listed below

Figure 23: Circuit Schematics in Boole-Web

11. Repeat steps 4-10 for each of the Digital Trainer challenges from Labs 1 and 2. Verify that the resulting K-Maps and Boolean equations you found in Tables 1-10 are equivalent to those computed by Boole-Web. Use the "Recognizing Patterns" section in Lab 3 to write down your observations.

12. OPTIONAL: Note that Boole-Web has many features which are not covered by this Lab but you are free to explore. If you're curious, under tab "3-Truth table, look at the options under "Select a formula to display." What do you notice for each option? In tab "5-Circuit', check out "Try in Real Hardware." Can you connect signals to I/O on an FPGA to observe real-world behavior? Also, take a look at the "Download PDF" and "Save system" features for easy access outside of LabsLand.

Recognizing Patterns

Compare your K-Maps and Boolean equations in Tables 1-10 to the K-Maps and Boolean Equations computed by Boole-Web. If your guess matches, compare your initial reasoning to Boole-Web's process. If the K-Maps or equations do not match, revisit the LabsLand Digital Trainer and double check the system behavior. Describe what you observe.

Table 11: K-Map Reasonings

Challenge	Operator Guess	Did your guess match Boole-Web? Why or why not? (1-2 sentences)
1		
2		
3		
4		
5		

6		
7		
8		
9		
10		

Looking Ahead

This lab wraps up the fundamental topics covered in this lab series for understanding digital logic. Now that you know about Boolean Algebra and K-Maps, we can look into applying it in real systems. The following lab will shift to introducing computer tools which will help us visualize the digital logic concepts we've been discussing. Future labs will call on all the foundational knowledge (Boolean Algebra, K-Maps, Truth Tables, Binary Numbers, Gates, and Operators) we've built so far.

Reflection and Observations

Lab 3 introduced you to K-Maps and Boolean algebra. Reflect on the new topics and the things you found interesting and/or challenging in the space below.

What questions do you still have, if any?