

# COMSM1302 Lab Sheet 1

This lab sheet covers content from week 1 lectures, focusing on logic gates and NAND implementation. On successful completion of the lab, students should be able to:

1. Simplify a given Boolean expression, which contains a selection of  $\wedge$ ,  $\vee$ , and  $\neg$ .
2. Assemble and validate a circuit on Logisim, corresponding to a given Boolean expression.
3. Implement a given Boolean expression on Logisim, using only NAND gates.
4. Create a Karnaugh map, with up to 5 variables, corresponding to a given truth table.
5. Produce a simplified Boolean expression, with up to 5 variables, corresponding to a given Karnaugh map.

You'll need the Logisim software to implement gates and circuits, and may need to install Java first if running Logisim on your own computer. Details on how to download and run Logisim can be found on the [unit page](#) and if you need help with using Logisim, try referring to the [Logisim user guide](#).

You will also need to borrow a NAND board to try physical implementation, which you can find details of on the [unit page](#). If you get stuck using NAND boards, try watching the [NAND board explanation](#) or have a look through the [GitHub documentation](#).

## Simplification

Simplify the following Boolean expressions, where each simplified solution will be either 0, 1, or  $A$ .

- |                |              |                     |                   |
|----------------|--------------|---------------------|-------------------|
| • $0 \wedge 0$ | • $0 \vee 0$ | • $0 \wedge A$      | • $0 \vee A$      |
| • $0 \wedge 1$ | • $0 \vee 1$ | • $1 \wedge A$      | • $1 \vee A$      |
| • $1 \wedge 0$ | • $1 \vee 0$ | • $A \wedge A$      | • $A \vee A$      |
| • $1 \wedge 1$ | • $1 \vee 1$ | • $\neg A \wedge A$ | • $\neg A \vee A$ |

## Circuit assembly

Assemble a logically equivalent circuit on Logisim for each of the following Boolean expressions, using logic gates with two or fewer inputs<sup>1</sup>. Validate your solution by checking every possible combination of input values produces the output value you'd expect.

**Hint:** You'll need to draw out each expression's truth table to validate your circuits.

- |  |  |
|--|--|
| • $((A \wedge C) \vee (B \wedge C)) \wedge C$  | • $(A \wedge B) \vee \neg(C \vee \neg A)$    |
| • $\neg(A \vee \neg(B \wedge C))$              | • $A \vee (B \wedge (C \vee A))$             |
| • $((A \vee B) \wedge (A \vee \neg B)) \vee C$ | • $\neg(\neg(A \wedge B \wedge C) \wedge C)$ |

**Challenge:** For each circuit, applying Boolean algebra simplification techniques, what are the least number of gates (with two or fewer inputs) you can implement it with?

<sup>1</sup>NOT gates only have one input, but you should use two input versions of any other logic gates.

## Propositional logic

Rewrite each of the following statements as Boolean expressions by representing each true/false condition with a propositional variable (e.g. A, B, C, ...) and combining them using  $\wedge$ ,  $\vee$ , and  $\neg$ .

**Hint:** Try thinking in terms of the inputs (e.g. “the Wi-Fi is on”, “knows the password”) and the output (e.g. “can connect”).

- Employees qualify for an end-of-year bonus if they have worked the whole year, unless they are on probation. Employees can also get the bonus if they have an ‘excellent’ performance rating.
- Parking is free at weekends, whereas on weekdays it is only free if you display a permit. However, you must always pay if your vehicle is a lorry.
- Anyone with a ticket may enter a concert, except children under 12 unless they are accompanied by an adult. Performers can always enter, even without a ticket.

## Karnaugh maps

For each of the truth tables below, draw out a K-map and then find a corresponding minimal Boolean expression.

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

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

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

Validate each expression by assembling a logically equivalent circuit on Logisim, using logic gates with two or fewer inputs, and checking every possible combination of input values produces the output value on the truth table.

## NAND gates

Using principles covered in the NAND lecture, implement the following Boolean expressions on Logisim, only using 2-input NAND gates.

**Hint:** For A XOR B, start from the formula  $(\neg A \wedge B) \vee (A \wedge \neg B)$ .

- NOT A
- A AND B
- A OR B
- A NOR B
- A XOR B

Validate your solutions by assembling each of your designs on a NAND board, being careful to sensibly plan where you will connect wires first.

**Challenge:** Can you optimise each of your circuit designs to use the least number of 2-input NAND gates?

**Hint:** For A XOR B, try starting from the formula  $(\neg A \vee \neg B) \wedge (A \vee B)$ !

## 5-variable logic

Find a minimal Boolean expression for the given 5-variable truth table. You should follow a similar process to the end of Lecture 1.2 and first create two 4-variable Karnaugh maps that represent:

- All values of BC\DE when A is 0.
- All values of BC\DE when A is 1.

Next, assemble a logically equivalent circuit on Logisim using your Boolean expression. Your circuit should use logic gates with 2 or fewer inputs, and can be validated by checking every possible combination of input values produces the output value on the truth table.

A	B	C	D	E	Out	A	B	C	D	E	Out
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	0
0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	0	0	1
0	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0	1	1	0	0
0	0	1	1	1	1	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	1	1	1	0	1	0	1
0	1	0	1	1	1	1	1	0	1	1	1
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	0	1	0	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0

**Challenge:** Create a 5-variable Karnaugh map that represents the given truth table. What do you notice if you draw the same groups of 1s from you two 4-variable Karnaugh maps?

**Hint:** You'll need to calculate the sequence of 3 bit binary-reflected Gray code to list ABC inputs.

## NAND implementation

Re-implement your 5 variable circuit from the previous exercise on Logisim only using 2-input NAND gates.

**Hint:** Try replacing each of the gates on your original design with their NAND implementations. This will likely be harder to optimise than algebraic methods, but is satisfactory for the purpose of this question.

**Challenge:** Implement your 5-variable NAND circuit on a NAND board! **Hint:** In order to attempt this, you'll need a design that uses 16 NAND gates or fewer, a clear system for transferring your design, as well as consideration for how you'll represent the 5 input variables with only 4 push buttons.