COMSM1302 Lab Sheet 1 (Solutions)

Simplification

- $0 \wedge 0 \equiv 0$
- $0 \wedge 1 \equiv 0$
- $1 \wedge 0 \equiv 0$
- $1 \wedge 1 \equiv 1$
- $0 \vee 0 \equiv 0$
- $0 \vee 1 \equiv 1$
- $1 \vee 0 \equiv 1$
- $1 \vee 1 \equiv 1$
- $0 \wedge A \equiv 0$
- $1 \wedge A \equiv A$
- $A \wedge A \equiv A$
- $\neg A \wedge A \equiv 0$
- $0 \vee A \equiv A$
- $1 \vee A \equiv 1$
- $A \vee A \equiv A$
- $\neg A \vee A \equiv 1$

The first column is just the values in the AND truth table, while the second column is the values in the OR truth table. You can prove the other simplifications by drawing out the truth tables e.g:

A	$0 \wedge A$
0	0
1	0

As $0 \wedge A$ outputs 0, no matter the value of A, it is equivalent to 0 i.e. always false.

A	$A \vee A$
0	0
1	1

As $A \vee A$ outputs the same value as A, it is equivalent to A.

Circuit assembly

Direct circuit implementations are shown in the Logisim solutions (subcircuit names detailed below), as well as the simplified "challenge" version. Note that these solutions use the least number of NOT, AND, and OR gates; there may be solutions that use fewer gates and use other 2-input gates such as NAND.

Simplification	Rationale
$((A \wedge C) \vee (B \wedge C)) \wedge C$ $((A \vee B) \wedge C) \wedge C$ $(A \vee B) \wedge C$	See subcircuit *circuit1* $\Rightarrow$ Distributivity, taking out the common variable "$\wedge\ C$" $\Rightarrow$ $((A \vee B) \wedge C) \wedge C \equiv (A \vee B) \wedge (C \wedge C)$ and $C \wedge C \equiv C$
$\neg(A \vee \neg(B \wedge C))$ $\neg A \wedge (B \wedge C)$	See subcircuit *circuit2* $\Rightarrow$ De Morgan's
$((A \vee B) \wedge (A \vee \neg B)) \vee C$ $(A \vee (B \wedge \neg B)) \vee C$ $A \vee C$	See subcircuit *circuit3* $\Rightarrow$ Distributivity, taking out the common variable "$A \vee$" $\Rightarrow$ $B \vee \neg B \equiv 0$ and $A \vee 0 \equiv A$
$(A \wedge B) \vee \neg(C \vee \neg A)$ $(A \wedge B) \vee (\neg C \wedge A)$ $A \wedge (B \vee \neg C)$	See subcircuit *circuit4* $\Rightarrow$ De Morgan's $\Rightarrow$ Distributivity, taking out the common variable "$A \wedge$"
$A \vee (B \wedge (C \vee A))$ $(A \vee B) \wedge (A \vee (C \vee A))$ $(A \vee B) \wedge (A \vee C)$ $A \vee (B \wedge C)$	See subcircuit *circuit5* $\Rightarrow$ Distributivity, adding in the common variable "$A \vee$" $\Rightarrow$ $A \vee (C \vee A) \equiv (A \vee A) \vee C$ and $A \vee A \equiv A$ $\Rightarrow$ Distributivity, taking out the common variable "$A \vee$"
$\neg(\neg(A \wedge B \wedge C) \wedge C)$ $(A \wedge B \wedge C) \vee \neg C$ $(A \vee \neg C) \wedge (B \vee \neg C) \wedge (C \vee \neg C)$ $(A \vee \neg C) \wedge (B \vee \neg C)$ $(A \wedge B) \vee \neg C$	See subcircuit *circuit6* $\Rightarrow$ De Morgan's $\Rightarrow$ Distributivity, adding in the common variable "$\vee\ \neg C$" $\Rightarrow$ $C \vee \neg C \equiv 1$ and $(A \vee \neg C) \wedge (B \vee \neg C) \wedge 1 \equiv (A \vee \neg C) \wedge (B \vee \neg C)$ $\Rightarrow$ Distributivity, taking out the common variable "$\vee\ \neg C$"

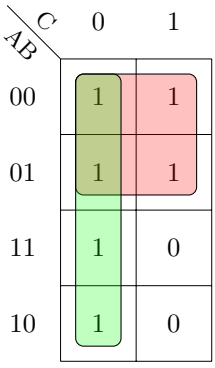
Propositional logic

Each input and output variable should be defined as a proposition: something that can clearly be described in terms of true or false. Remember a useful way to check is using the phrase "It is true that...".

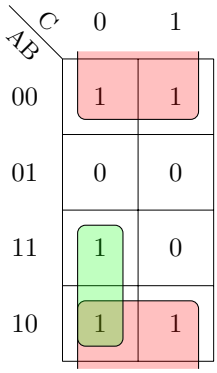
Inputs	Output
They have worked the whole year. (A) They are on probation. (B) They have an 'excellent' performance rating. (C)	Employee qualifies for end-of-year bonus. $(A \wedge \neg B) \vee C$
It is the weekend. (A) A permit is displayed. (B) The vehicle is a lorry. (C)	The parking is free. $(A \vee B) \wedge \neg C$
They have a ticket. (A) They are under 12. (B) They are accompanied by an adult. (C) They are a performer. (D)	Person may enter a concert. $(A \wedge (\neg B \vee C)) \vee D$

When you write your Boolean expression, there will be more than one correct way of expressing the logic. For example, $(A \wedge \neg(B \wedge \neg C)) \vee D$ is a logically equivalent formula for the last question.

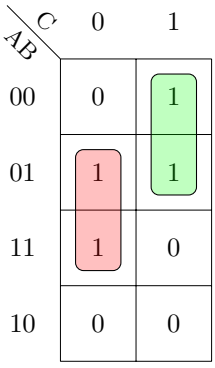
Karnaugh maps



$\neg A \vee \neg C$
See subcircuit *kmap1*



$\neg B \vee (A \wedge \neg C)$
See subcircuit *kmap2*



$(B \wedge \neg C) \vee (\neg A \wedge C)$
See subcircuit *kmap3*

NAND gates

When trying to "find NANDs" in Boolean expressions, you're looking to manipluate your expression into the form $\neg(X \wedge Y)$, where X and Y may contain further NAND expressions. The final result may also include extra $\neg$'s, like in the case of AND, as this can also be implemented using a singular NAND gate with copied inputs. A common trick to achieve this is to first add a double negation, which creates a logically equivalent expression that, if needed, can then have a $\neg$ pushed through (using De Morgan's) to remove any $\vee$'s from the expression.

Simplification	Rationale
$\neg A$	See subcircuit *not_nand*
$\neg(A \wedge A)$	$\Rightarrow A \equiv A \wedge A$
$A \wedge B$	See subcircuit *and_nand*
$\neg\neg(A \wedge B)$	$\Rightarrow$ Double negation
$A \vee B$	See subcircuit *or_nand*
$\neg\neg(A \vee B)$	$\Rightarrow$ Double negation
$\neg(\neg A \wedge \neg B)$	$\Rightarrow$ De Morgan's
$\neg(A \vee B)$	See subcircuit *or_nor*
$\neg\neg\neg(A \vee B)$	$\Rightarrow$ Double negation
$\neg\neg(\neg A \wedge \neg B)$	$\Rightarrow$ De Morgan's
$(\neg A \wedge B) \vee (A \wedge \neg B)$	See subcircuit *xor_nand5*
$\neg\neg((\neg A \wedge B) \vee (A \wedge \neg B))$	$\Rightarrow$ Double negation
$\neg(\neg(\neg A \wedge B) \wedge \neg(A \wedge \neg B))$	$\Rightarrow$ De Morgan's
$(A \vee B) \wedge (\neg A \vee \neg B)$	See subcircuit *xor_nand4*
$(A \vee B) \wedge \neg(A \wedge B)$	$\Rightarrow$ De Morgan's
$(A \wedge \neg(A \wedge B)) \vee (B \wedge \neg(A \wedge B))$	$\Rightarrow$ Distributivity, adding the common variable "$\wedge\ \neg(A \wedge B)$"
$\neg\neg((A \wedge \neg(A \wedge B)) \vee (B \wedge \neg(A \wedge B)))$	$\Rightarrow$ Double negation
$\neg(\neg(A \wedge \neg(A \wedge B)) \wedge \neg(B \wedge \neg(A \wedge B)))$	$\Rightarrow$ De Morgan's

5-variable logic

A = 0

DE \ BC	00	01	11	10
00	0	0	1	1
01	1	0	1	1
11	1	0	1	1
10	0	0	1	1

A = 1

DE \ BC	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	0	0	0
10	0	0	1	1

$(\neg A \wedge D) \vee (C \wedge \neg D \wedge \neg E) \vee (B \wedge \neg C \wedge D)$

See subcircuit *5vars*

DE \ ABC	000	001	011	010	110	111	101	100
00	0	1	1	0	0	1	1	0
01	0	0	0	0	0	0	0	0
11	1	1	1	1	1	0	0	0
10	1	1	1	1	1	0	0	0

The key takeaway from the 5-variable Karnaugh Map is the mirror line between 010 and 110 for boxes that cover 1s on both sides of it. In particular, the yellow box doesn't directly connect but can be grouped because of the symmetry and the green box cannot be extended as symmetry must be maintained. The red box does not require this symmetry as it cover 1s only on one side of the mirror line.

NAND implementation

A solution is valid as long as it produces the correct outputs and only uses 2-input NAND gates. One method is to replace each logic gate with its NAND implementation, with the possibility to simplify further as shown in slide 6 of the NAND lecture. Another method is to bubble push starting from the final output.

When attempting a physical implementation on NAND boards, you should be able to identify which NAND on your NAND board represents each NAND in your Logisim design. For the fifth variable, a wire can be connected to a constant pin for 1 or disconnected for 0.

For a NAND implementation only using 14 NAND gates, see subcircuit *5vars_nand*.