CSE1400 - Computer Organisation

Self-Study: Week 2
Logic Circuits

Delft University of Technology

2021/2022 Q1

Special thanks to Sára Juhošová, Ana Băltăreţu, Oskar Lorek, Kiril Vasilev and Damla Ortac for helping with the compilation of this set of questions.

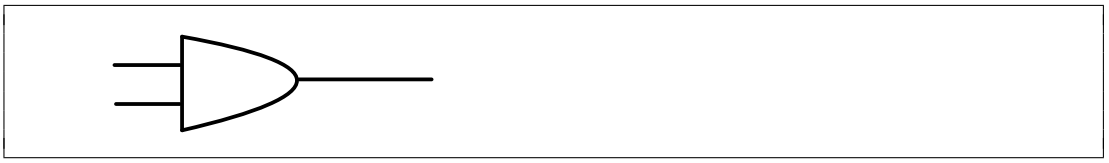
Important information:

1. If any question is unclear please consult Stack Overflow. For more specific questions, you can use the Queue during lab hours.
2. The average time for solving this self study is **3** hours, and **1** hour is allocated to giving feedback. Timings are included for each exercise to give you a more clear overview of how much time you should be spending on them.
3. The maximum amount of points for this self study is 200 points. To get the points you should submit a serious attempt on Peer and **properly review** your peers' submissions (100 points per full cycle, including review evaluation).
4. Answers will be provided during the weekly tutorial sessions.

1. (2 mins) What is the name of the inventor of the algebraic system that is used in digital computers and in what year did he invent it?

George Boole, 1854

2. (5 mins) Draw the following logic gates:
   (a) AND

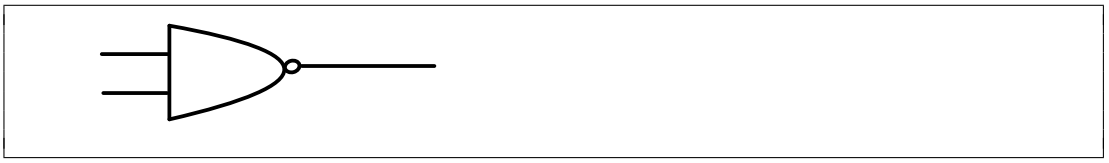


   (b) NOR

   (c) NOT

   (d) OR

   (e) NAND



3. (3 mins) Simplify the following equation using DeMorgan's laws: $\overline{((x \cdot \overline{y}) \cdot (\overline{y} + z))}$.

$\overline{((x.\bar{y}).(\bar{y}+z))} = (x.\bar{y}) + (\overline{\bar{y}+z}) = (x.\bar{y}) + (y.\bar{z}) = x\bar{y} + y\bar{z}$

4. (30 mins) Consider the following logic circuit:

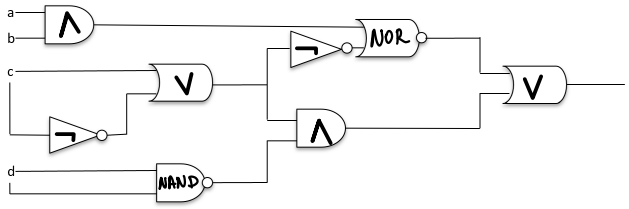


Figure 1: Logic Circuit

(a) Convert the logic circuit to the corresponding (non-simplified) boolean expression.

$\neg(((a \wedge b) \vee (\neg(c \vee \neg c))) \vee ((c \vee \neg c) \wedge \neg(d \wedge d)))$

(b) Draw a Karnaugh map from the expression you derived.

cd \ ab	00	01	11	10
00	1	1	1	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

(c) Find the minimal sum of products for the expression using either the Karnaugh map or boolean logic.

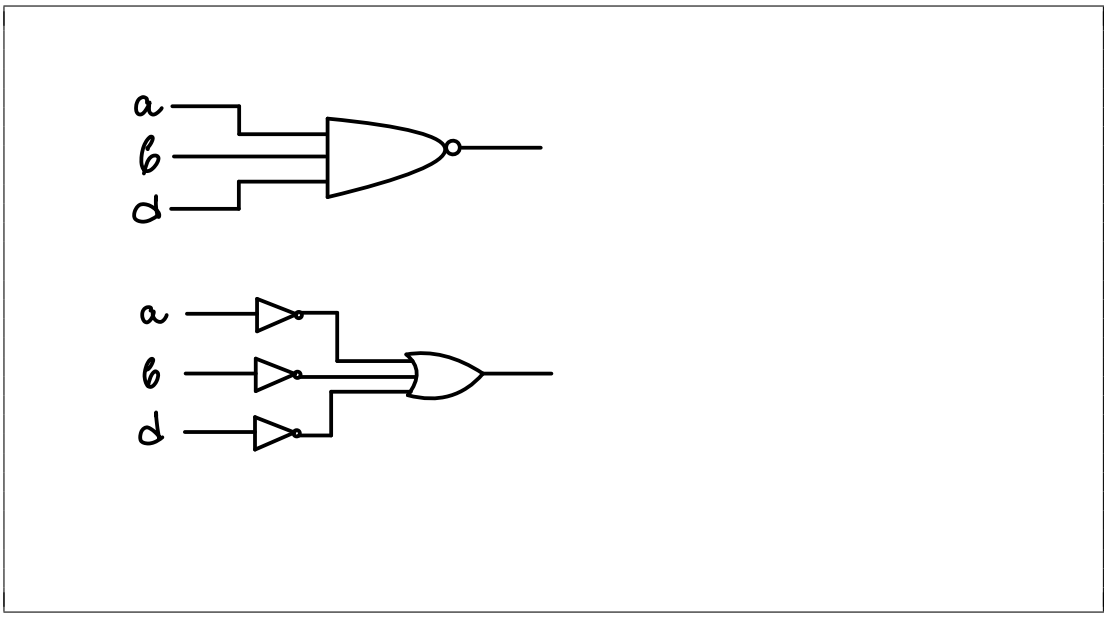
$\bar{a} + \bar{b} + \bar{d}$

$\neg(((a \wedge b) \vee (\neg(c \vee \neg c))) \vee ((c \vee \neg c) \wedge \neg(d \wedge d))) \equiv$

$\equiv (((\neg a \vee \neg b) \wedge (T)) \vee (T \wedge \neg d) \equiv \neg a \vee \neg b \vee \neg d$

$\equiv \neg(a \wedge b \wedge d) \mid \overline{abd}$

(d) Draw a logic circuit corresponding to the minimal sum of products expression.



(e) Does the logic circuit that expresses a minimal sum of products always use the least possible amount of logic gates? Is it possible to redesign the logic circuit from the previous exercise to use fewer logic gates?

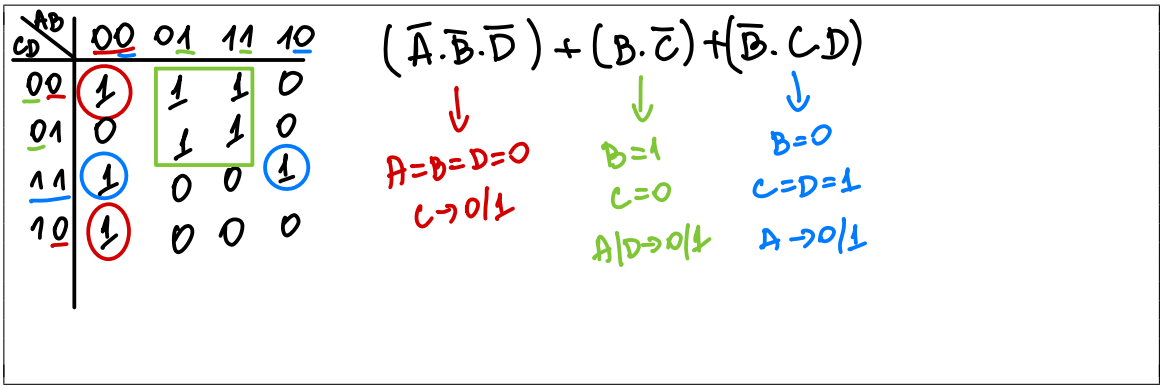
No, the logic circuit that expresses a minimal sum of products doesn't always use the least amount of logic gates. For example, using Karnaugh map and/or boolean logic in the previous exercise simplifies the logic circuit to the expression $\bar{a}+\bar{b}+\bar{d}$ which can be constructed with 4 logic gates (3 NOT + 1 OR). However, this can be expressed as $\overline{abd}$, which can be constructed using 1 NAND gate.

5. (10 mins) Otto, Thomas and Stefan were fighting for the honorary title of "Candy King", and Thomas was slowly taking the lead with his weekly candy deliveries to the TAs. Otto realised that in order to make a comeback, he had no choice but to bring out his secret weapon: Eugenia[1] biscuits. Unfortunately, while he was dropping off a package of goodies, the bottom of it broke and covered one of Otto's Karnaugh maps with Eugenias.

   Otto quickly realised that this event created an excellent question and he took a picture of the Karnaugh map. Now he tasks you with filling in the spots that were covered by the Eugenias.

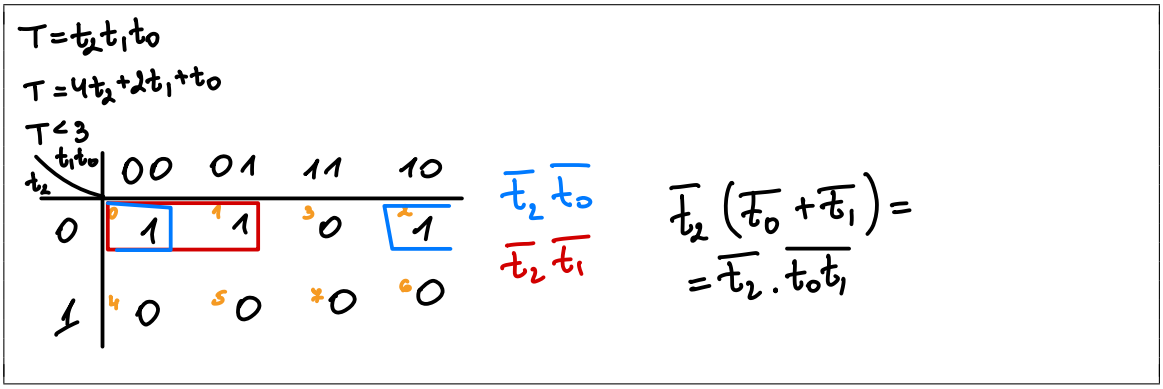
   You also know that the minimum **Sum of Products** for this Karnaugh map was:
   $(\overline{A} \cdot \overline{B} \cdot \overline{D}) + (B \cdot \overline{C}) + (\overline{B} \cdot C \cdot D)$.

   Draw the original Karnaugh map and show your work.



[1] Eugenias are sandwich biscuits filled with chocolate cream and rum flavour. These treats were famous with Romanian children because of the chocolate filling, and they were popular with the parents too, because they didn't cost too much.

6. (7 mins) The current temperature is represented as a 3 bit unsigned number T = $t_2t_1t_0$, where t_2 is the most significant bit. Any temperature of 7 degrees or above is stored as 111 and any temperature of 0 degrees or below is stored as 000. Find a minimal Sum-of-Products equation that is only true when the temperature is strictly below 3 degrees.



7. (5 mins) Consider the following CMOS circuit:

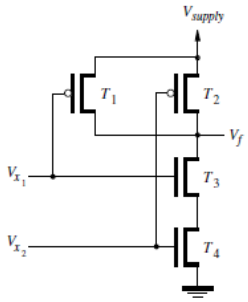
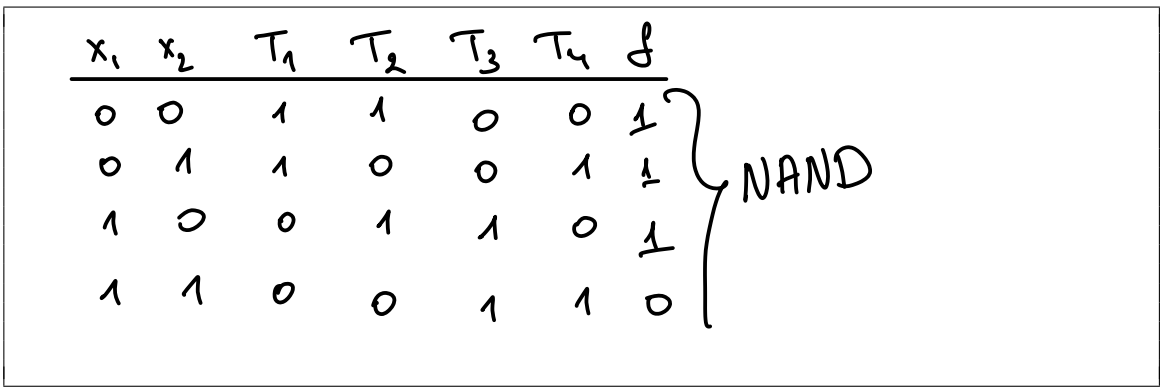


Figure 2: CMOS Circuit

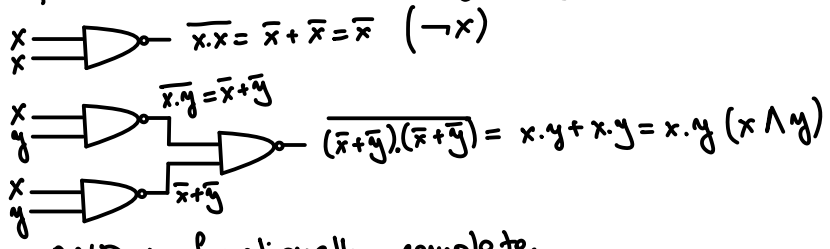
Draw the table of outputs for each possible input of x_1, x_2 and state the name of the gate.



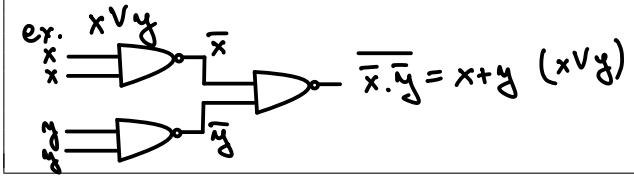
8. (10 mins) Alice wants to build a supercomputer that can perfectly predict the stock prices. Unfortunately, as a TUDelft student, she is low on budget. Thus, she wants to cut the price by reducing the number of different types of gates used in her supercomputer (she would have to build a separate production line for each logic gate).

   What is the minimum number of different types of gates that she could use in her supercomputer to express every possible logic circuit? **Explain why.**

She can build her supercomputer only using NAND logic gates.
From logic we know that $\{\wedge, \neg\}$ is functionally complete set, thus if we can perform those operations only using NAND, it is also functionally complete.



$\Rightarrow$ NAND is functionally complete.



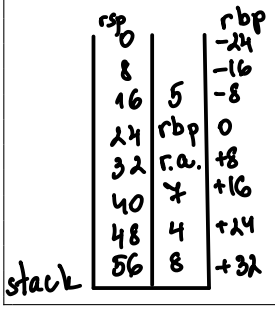
9. (7 mins) What value does the RAX register hold after it returns to line 6 (from calling the foo function)?

```
movq   $1, %rax
pushq  $8
pushq  $4
pushq  $7
movq   $5, %rdi
addq   $8, %rsp
call   foo

foo:
pushq  %rbp
movq   %rsp, %rbp

pushq  %rdi
mulq   -8(%rbp)
addq   16(%rbp), %rax

movq   %rbp, %rsp
popq   %rbp
ret
```



value of %rax = $1 \times 5 + 4 = 9$

10. (15 mins) You are given the following piece of Assembly x86-64 code:

```
1.   movq    $0, %rsi
2.   pushq   %rsi ->0
3.   call    foo
4.   popq    %rdi ->0
5.   decq    %rax ->41
6.   addq    %rdi, %rax ->41
7.   pushq   %rax ->41

8.   movq    $0, %rdi
9.   call    exit

10.  foo:
11.  pushq   %rbp
12.  movq    %rsp, %rbp

13.  addq    $42, %rsi
14.  pushq   %rsi ->42
15.  decq    %rsi ->41
16.  popq    %rax ->42
17.  pushq   %rsi ->41

18.  movq    %rbp, %rsp
19.  popq    %rbp
20.  ret
```

Before line 1 is executed, the 8-byte aligned stack looks like this:

Address	Contents	Pointers
400	42	
392	"Coati"	**RBP**
384	10	**RSP**
376	123	
368	"Owls"	
360	13	
352	2	
344	666	

(a) What will the stack look like after line 17 is executed? Explain how you got to the answer.

1÷2: RSP points to address 376, which has a value of 0.
3: RSP points to address 368, which stores return address.
11÷12: RSP points to address 360, and %RBP as well
13÷17: RSP points to address 352, which has a value of 41.

You can use the following table to fill in your answer:

Address	Contents	Pointers
400	42	
392	"Coati"	(RBP)
384	10	
376	0	
368	<ret>	
360	392 (RBP)	RBP
352	41	RSP
344	666	

(b) What will the stack look like after line 7 is executed? Explain how you got to the answer.

18-19: RSP points to address 368, after popping %RBP from 360
20: RSP points to address 376, which has a value of 0.
4: RSP points to address 384, which has a value of 10.
7: RSP points to address 376, which has a value of 41.

You can use the following table to fill in your answer:

Address	Contents	Pointers
400	42	
392	"Coati"	RBP
384	10	
376	41	RSP
368	<ret>	
360	392 (RBP)	
352	41	
344	666	