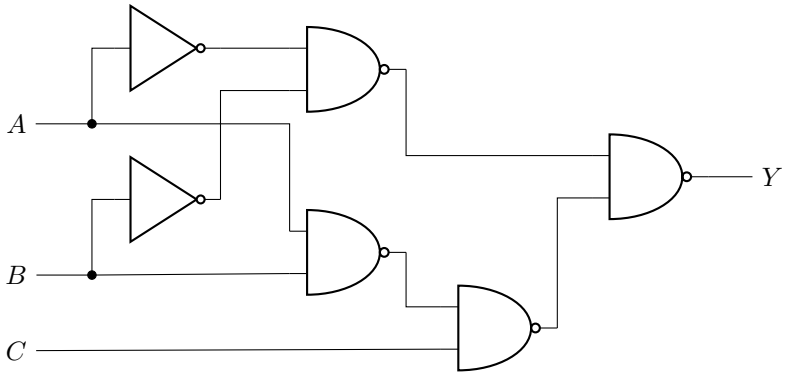
Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2020-10-30
Due: 2020-11-06

ICS 2020 Problem Sheet #8

Problem 8.1: *combinational digital circuit* (1+1 = 2 points)

You are given the following digital circuit.



a) What is the Boolean expression implemented by the digital circuit?

b) Derive algebraically step-by-step the disjunction (sum) of minterms from the Boolean expression implemented by the digital circuit.

Problem 8.2: *full adder using different kinds of gates* (1+1+1+1 = 4 points)

A full adder digital circuit was introduced in class. It is defined by the following two boolean functions:

$$\begin{aligned} S &= A \dot{\vee} B \dot{\vee} C_{in} \\ C_{out} &= (A \wedge B) \vee (C_{in} \wedge (A \dot{\vee} B)) \end{aligned}$$

a) Write both functions as a disjunction of product terms.

b) Write both functions as a conjunction of sum terms.

c) Write both functions using only not ($\neg$) and not-and ($\uparrow$) operations.

d) In a digital circuit, we can easily reuse common terms. Draw a small digital circuit implementing S and C_{out} using NAND gates only.

Problem 8.3: *haskell fizzbuzz* (1+1+2 = 4 points)

The "fizzbuzz" game is a common interview question for programming jobs: Print the natural numbers starting with 1 in increasing order. If the number is divisible by three, print `"fizz"` instead of the number. If the number is divisible by five, print `"buzz"` instead of the number. If the number is both divisible by three and by five, print `"fizzbuzz"` instead of the number.

During interviews, the discussion usually goes into the direction how to write the code in such a way that it can be easily extended, which then often leads to discussions about trade-offs between different possible solutions. We are not going into these details here.

a) Write a Haskell function `fizzbuzz :: Integer -> String` that takes a positive integer and returns the number rendered as a string or one of the strings `"fizz"`, `"buzz"`, or `"fizzbuzz"`, following the rules defined above.

b) Using `foldr`, write a *simple* expression that returns the fizzbuzz sequence as a list of strings for the numbers in the range 1 to 16. Do not use list comprehensions or other higher order functions or lambda functions.

c) Using `foldl`, write a *simple* expression that returns the fizzbuzz sequence as a list of strings for the numbers in the range 1 to 16. Do not use list comprehensions or other higher order functions (and ideally no lambda functions but you may use `flip`).