Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2018-11-07
Due: 2018-11-15

ICS 2018 Problem Sheet #8

Problem 8.1: *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 8.2: *ripple carry adder and carry lookahead adder (haskell)* (1+1+2+2 = 6 points)

You task is to implement a ripple carry adder and a carry lookahead adder. Binary numbers will be represented as a list of `Bool` values. We break things into small steps:

a) Implement a function `bin m n` that converts the non-negative integer number `n` into a list of Bools. The list returned list will have the length `m`.

```
ghci> bin 4 5
[False,True,False,True]
ghci> bin 8 42
[False,False,True,False,True,False,True,False]
```

b) Implement a function `dec x` that converts a list of Bool values into the corresponding non-negative integer number.

```
ghci> dec [False,True,False,True]
5
ghci> dec [False,False,True,False,True,False,True,False]
42
```

c) Implement the functions `faC` and `faS` that receive two input boolean values and a carry boolean value and calculate the carry (`faC`) and the sum (`faS`) of the full adder digital circuit. Use these two functions to implement `rcAdd`, a ripple carry adder. For simplicity, `rcAdd` is not returning the final carry bit.

```
ghci> rcAdd [False,True,False,True] [True,False,False,False]
[True,True,False,True]
```

Combining `rcAdd` with the other functions, you should be able to do computations like this:

```
ghci> dec (rcAdd (bin 4 5) (bin 4 8))
13
```

d) Implement the functions `haC` and `haS` that receive two input boolean values and calculate the carry (`haC`) and the sum (`haS`) of the half adder digital circuit. Use these two functions to implement `claAdd`, a carry lookahead adder. It is sufficient to implement the carry calculator as a recursive function. For simplicity, `claAdd` is not returning the carry bit.

```
ghci> claAdd [False,True,False,True] [True,False,False,False]
[True,True,False,True]
ghci> dec (claAdd (bin 4 5) (bin 4 8))
13
```

Submit your Haskell code plus an explanation (in Haskell comments) as a plain text file.