Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2017-11-07
Due: 2017-11-14

ICS 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 (short) disjunction of product terms.

b) Write both functions as a (short) 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. What is 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. 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 Booleans. 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 Booleans 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 `fa_c` and `fa_s` that receive the two input boolean values and a carry boolean value and calculate the carry (`fa_c`) and the sum (`fa_s`) of the full adder digital circuit. Use these two functions to implement `rc_add`, a ripple carry adder. For simplicity, `rc_add` is not returning the carry bit.

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

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

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

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

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