**Universität des Saarlandes**
**FR 6.2 - Informatik**
Prof. Dr. W.J. Paul
Dipl.-Ing. Christoph Baumann

**Exercise Sheet 3**
**Computer Architecture II**
(Due: Nov 15th, 2011)

---

**Exercise 1: (Leading Zero Counter)** **(5+7+8 Points)**

Let $n \in \mathbb{N}^+$, $x \in \mathbb{B}^n$ and $y \in \mathbb{B}^m$ with $m = \lceil \log_2(n+1) \rceil$. For a bit string $x$, we denote the number of leading zeros of $x$ by $lz(x)$. An $n$-bit leading zero counter ($n$-LZC) is a circuit with input $x$ and output $y$ satisfying:

$$\langle y \rangle = lz(x)$$

a) Give a recursive definition for $lz(x)$!

b) Construct an $n$-LZC for an arbitrary n!

c) Prove the correctness of your construction!

**Exercise 2: (Alignment Shift Limitation)** **(10 Points)**

When adding two IEEE-normal floating-point numbers $(s_a, e_a, f_a)$ and $(s_b, e_b, f_b)$, it is necessary to align the significands by multiplying the operand having the smaller exponent with $2^{-\delta}$, where we assume $\delta = e_a - e_b \geq 0$ wlog. This is called an *Alignment Shift*. We have shown in the lecture that it is enough to use the $p+1$-representative $f' = [2^{-\delta} \cdot f_b]_{p+1}$ instead of $2^{-\delta} \cdot f_b$ in the computation. In particular we had:

$$S = 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot 2^{-\delta} \cdot f_b) =_{p-\hat{e}} 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot f')$$

Now imagine we used only the usual $p$-representative $f'' = [2^{-\delta} \cdot f_b]_p$ in the computation of the sum $S''$:

$$S'' = 2^{e_a} \cdot ((-1)^{s_a} \cdot f_a + (-1)^{s_b} \cdot f'')$$

In order to show that this does not suffice, find a counter-example and give values for $s_a$, $s_b$, $f_a$, $f_b$ and $\delta$ such that

$$S =_{p-\hat{e}} S''$$

does *not* hold!

**Hint:** Compute the normalized representations $\hat{\eta}(S) = (s, \hat{e}, \hat{f})$ and $\hat{\eta}(S'') = (s, \hat{e}, \hat{f}'')$ to obtain $\hat{e}$!