Saarland University
Department 6.2 – Computer Science
Prof. Dr. W. J. Paul
M. Sc. Petro Lutsyk

Computer Architecture – WS14/15

Exercise Sheet 3 (due: 18.11.14, 34 points)

Exercise 1: (parallel prefix) (8 points)

For set M and associative (!) operation $\circ : M \times M \to M$ we define a *parallel-prefix* function

$$PP_\circ : M^n \to M^n,$$

s.t., for $i \in [0 : n-1]$ the output $y = PP_\circ(x)$ is defined as follows:

$$y[i] = \begin{cases} x[i] & i = 0 \\ x[i] \circ y[i-1] & i > 0. \end{cases}$$

Give a construction of the circuit implementing the $PP_\vee$, i.e., the parallel-prefix function for $M = \mathbb{B}$ and $\circ = \vee$, that has

- linear cost and linear delay. **(half points)**
- linear cost and logarithmic delay. **(full points)**

Prove that your implementation is correct (for simplicity you can assume that $n = 2^k$ for $k > 0$).

Exercise 2: (smart half-decoder) (6 points)

Give a construction of a "smart" half-decoder, i.e. a circuit which fulfills the following specification:

- input $x \in \mathbb{B}^n$
- output $y \in \mathbb{B}^{2^n}$, s.t.

$$y = 0^{2^n - \langle x \rangle} 1^{\langle x \rangle}$$

Prove that your construction

(a) mets the specification.

(b) has a delay logarithmic in n.

Exercise 3: (msb and lsb) (10 points)

Consider two new assembly instructions from table 1.

opcode	Mnemonic	Assembler-Syntax	Effect
Most and Least Significant Bits Set			
111 011	msb	msb *rt rs*	rt = (rs $\neq$ 0 ? 0 : 1) 0^{26} msb(rs)_5
111 111	lsb	lsb *rt rs*	rt = (rs $\neq$ 0 ? 0 : 1) 0^{26} lsb(rs)_5

Table 1: New I-type instructions

In this exercise we develop the hardware for computation of the most- and least-significant bits set (!) in the n-bit string. Here we assume that n is a power of two, i.e., $n = 2^k$ for $k \in \mathbb{N}$.

For $x \in \mathbb{B}^n$ we specify functions

$$msb, lsb : \mathbb{B}^n \rightharpoonup \mathbb{N}$$

as follows:

$$\begin{aligned} msb(x) &= \max\{i \in [0 : n-1] \mid x[i]\} \\ lsb(x) &= \min\{i \in [0 : n-1] \mid x[i]\}. \end{aligned}$$

(a) Give a construction of the circuit implementing the *msb* function, s.t., it has linear cost and logarithmic delay. Prove that your implementation is correct. Hint: in order to satisfy the requirements you may need to introduce a 'not zero' signal in your recursive construction.

Note that *bit numbering* is an attribute of the architecture. We number the bits from right to left starting with zero (LSB 0), i.e., for $x \in \mathbb{B}^n$ we have

$$x = x[n-1:0].$$

(b) For implementation of the *lsb* function we invoke an auxiliary circuitry computing the function

$$f1 : \mathbb{B}^n \to \mathbb{B}^n$$

(first one), s.t., for $i \in [0 : n-1]$

$$f1(x)[i] \leftrightarrow i = lsb(x).$$

Hint: use parallel prefix.

(c) Combine the results of (a) and (b) and give the construction of a component with the following

- inputs: $x \in \mathbb{B}^n$ and $m \in \mathbb{B}$, and
- outputs: $y \in \mathbb{B}^k$ and $z \in \mathbb{B}$, s.t.

$$\begin{array}{lcl} y & = & \begin{cases} bin_k(msb(x)) & m \\ bin_k(lsb(x)) & \text{otherwise} \end{cases} \\ z & \leftrightarrow & x = 0^n \end{array}$$

Exercise 4: (jump instructions) (4 points)

Study the instruction tables of the MIPS ISA for jump instructions.

Give the predicates which in the MIPS configuration c would indicate the following instructions:

- jump,
- jump-and-link,
- jump register,
- jump-and-link register.

For every instruciton above specify its semantics, i.e., the next state configuration

$$c' = \delta_M(c).$$

Exercise 5: (misnomer in ISA) (2 points)

Recall, during the lecture it was mentined that the MIPS instruction set has a "well known" *misnomer* in the instruction mnemonics. Explain (in words) which instructions this issue concerns and what in the actual implementation is different from the intuitive meaning given by the mnemonics.

Exercise 6: (BCE) (4 points)

Revise the lecture notes (or lookup in the tables) and implement the *Branch Condition Evaluation* component of the processor. For that

(a) repeat the control table of the BCE (signal $bf \in \mathbb{B}^4$),

(b) construct the BCE component and prove that your implementation is correct.

You can use zero and equality testers from the lecture as black-boxes and rely on their correctness.