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

Computer Architecture – WS14/15

Exercise Sheet 9 (due: 13.01.15, 32 points)

Exercise 1: (memory bus) (8 points)

Consider the sketch of the memory system from Figure 1. Using the lecture notes (or relevant sections of the script), list the lines that the memory bus b comprises and work out the following.

(a) For every line within bus b, state whether memory mm needs to put/take data on/from it. Add to memory mm the hardware necessary to support clean operation on every connected line.

(b) For every line within bus b, state whether cache $ca(i)$ needs to put/take data on/from it. Add to cache $ca(i)$ the hardware necessary to support clean operation on every connected line.

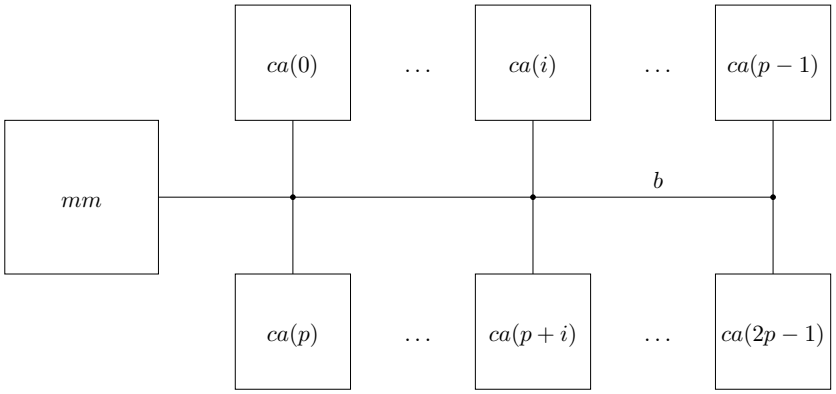

Figure 1: Memory system comprising main memory mm and $2p$ caches ca connected via bus b

Exercise 2: (bus arbiter) (8 points)

For the memory bus from the Exercise 1 we require an *arbiter*, a circuitry which in every hardware cycle "grants" the bus to exactly one cache. To be granted on the bus, cache i first raises its request signal $req[i]$. Not sooner than in the next cycle, cache i can be granted. In this exercise we implement the round-robin scheduling, of course with respect to the active requests.

Consider the arbiter depicted in Figure 2. Give a construction of the circuit which computes the $nextgrant \in \mathbb{B}^{2p}$ signal, s.t.

$$nextgrant[i] \leftrightarrow i = \begin{cases} \min\{j \mid req[j] \wedge j \geq k\} & grant[k] \wedge \exists j \geq k : req[j] \\ \min\{j \mid req[j]\} & \text{otherwise.} \end{cases}$$

Prove correctness of your implementation.

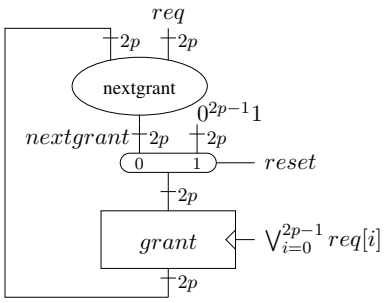

Figure 2: Simple bus arbiter

Exercise 3: (protocol states) (4 points)
List those states of the atomic MOESI protocol which can represent

(a) *clean* cache lines.

(b) *dirty* cache lines.

(c) *exclusive* cache lines.

(d) *shared* cache lines.

Exercise 4: (protocol tables) (12 points)
In the lecture we introduced master and slave tables together with the atomic cache coherence protocol. Explore the tables and explain what actions (including state transitions, bus signals, and data transfer) are performed by a master and slaves in case of a cache hit/miss for a

(a) read access.

(b) write access.

(c) CAS− access.

(d) CAS+ access.