FALL 2006
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 18, 2006

GENERAL INSTRUCTIONS:

1. Answer each question in a separate book.
2. Indicate on the cover of *each* book the area of the exam, your code number, and the question answered in that book. On *one* of your books list the numbers of *all* the questions answered. *Do not write your name on any answer book.*
3. Return all answer books in the folder provided. Additional answer books are available if needed.

SPECIFIC INSTRUCTIONS:

Answer all of the following **SIX** questions. The questions are quite specific. If, however, some confusion should arise, be sure to state all your assumptions explicitly.

POLICY ON MISPRINTS AND AMBIGUITIES:

The Exam Committee tries to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and ambiguities. If you are convinced a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the *first hour* of the exam. In any case, you should indicate your interpretation of the problem in your written answer. Your interpretation should be such that the problem is non-trivial.

1. Serial Decimal Digit Detector

Let a serial *Decimal Digit Detector* (DDD) receive one bit each cycle, wait for four bits, assert its output if the four bits are a valid binary-coded decimal (BCD) digit (0000, 0001, .., 1001), and then repeat for the next new set of four bits. The output should be asserted at most once every four bits, because only aligned digits are considered.

INPUT: A new single-bit **IN** is available each cycle from a D-flip-flop clocked by **CLK**. Assume that the most-significant bit (MSB) arrives first.

OUTPUT: Serial output **OUT** should be 1 when an BCD digit is found and 0 otherwise (not BCD or, as yet, incomplete).

(a) Present a finite-state machine diagram (FSM) for a serial DDD. Indicate what state that your FSM should start in (since we are not asking for a reset signal in this problem).

(b) "Implement" your design by assigning state values and giving equations for combinational logic. You will be graded for design correctness first and clarity second. Grading will not consider speed or gate count.

2. Trends in Microarchitectural Techniques

Computer architects are constantly debating the value of different microarchitectural techniques, and techniques that were considered very important in one generation are sometimes discarded in the next generation. Two techniques, trace caches and simultaneous multithreading (or hyperthreading) that were considered to be very important for a previous-generation microarchitecture (e.g., the Intel Pentium IV) were not considered desirable in a more recent microarchitecture (e.g, the Intel Core 2).

Put yourselves in the shoes of the chief architect of a contemporary microarchitecture.

(a) Argue why a trace cache is a good choice.

(b) Argue why a trace cache is a poor choice.

(c) Argue why simultaneous multithreading is a good choice.

(d) Argue why simultaneous multithreading is a poor choice.

3. Memory Consistency Models

Consider a single Chip Multiprocessor (CMP) consisting of a eight cores that connect to a private write-back L1 caches kept coherent with an MOESI directory protocol implemented at the four banks of an on-chip multi-bank shared write-back L2 cache. Assume each L2 bank misses to a corresponding bank of external DRAM memory.

Assume that each core is a single-issue in-order processor that speculates on conditional branches. Let a core retire an instruction after the instruction completes execution and all previous instructions have retired. Let stores retire by writing their value to the tail of a first-in-first-out (FIFO) write buffer. An entry is removed from the head of the write buffer when sufficient coherence permission has been obtained that the head's value can be written into the L1 cache.

(a) What mechanisms are needed to ensure that this CMP implements a processor consistent memory consistency model (e.g., SPARC Total Store Order (TSO))?

(b) Give example code and execution where your answer to part (a) realizes a processor consistent execution that is *not* sequentially consistent (SC).

(c) What mechanisms are needed to ensure that this CMP implements sequential consistency?

4. Branch in Large Instruction Windows

A number of recent microarchitecture proposals strive to hide main memory latency by creating effective execution window sizes of a thousand instructions or more. But since typically one in six instructions is a branch, this naively implies accurately predicting over 150 branches to fill the window without incurring a misprediction. Even with a 99% accurate branch predictor, there is only a 22% chance of correctly predicting 150 consecutive branches.

(a) Discuss the hardware and software techniques that can reduce the number of mispredicted branches in a large instruction window microarchitecture.

(b) Discuss ways to mitigate the impact of mispredicted branches in a large instruction window microarchitecture.

5. Reliability and CMPs

As semiconductor technology continues to scale to smaller feature sizes, microprocessor systems become much more susceptible to both transient (soft) and permanent (hard) faults. For example, the small capacitances of static RAM cells make them much more susceptible to transient errors due to sub-atomic particle strikes. And small wire dimensions and increasing process variation greatly increase the likelihood of metal migration that results in open circuits (i.e., a blown fuse).

Many architects believe that even volume microprocessor systems must soon take significant steps to maintain current levels of reliability. Yet classical fault-tolerance techniques, such as triple modular redundancy (i.e., voting logic), is widely considered too expensive for volume applications. Worse, the high levels of integration expected in future chip-multiprocessor (CMP) systems means that much or all of the processing logic will reside on a single silicon chip.

(a) Describe how you might apply a main-frame style pair-and-spare redundancy scheme (e.g., IBM G5) to a single chip CMP system. What types of faults could this handle? What types of faults would prove a challenge?

(b) Describe how you might apply temporal redundancy schemes (e.g., Diva) to a single chip CMP system. What types of faults could this handle? What types of faults would prove a challenge?

(c) Compare and contrast the two approaches. Which has more promise for this design point? Would a synthesis of these ideas provide a sweet spot solution or unnecessary complexity?

6. Synchronization Mechanisms

As the number of cores on a CMP increases, computer architects may have to design novel on-chip synchronization mechanisms to facilitate efficient synchronization of the multiple CMP cores. To do so computer architects are likely to see how synchronization was done in older parallel machines, and if and how those techniques might be adapted to the CMP context. The parameters that are likely to be considered include: the problem domain, the bandwidth required, and the latency of the synchronization operation.

There have been a variety of parallel machines that have used different synchronization techniques. These include the Stanford DASH, the Thinking Machines CM-5, the Cray T3E, and others.

Suppose you are to design a synchronization mechanism for a CMP with 32-64 cores. What lessons can you learn from the synchronization mechanisms of older machines, such as those above, and how might you adapt them for the CMP context?