Computer Science 418 Midterm

2004 October 28

Name (1 point):____

Student ID:____

This exam is **open-book, open-notes**. You may **not** use a calculator or any other electronic device during the exam. The exam ends at 12:20pm. **Stop writing immediately when time is called.** Good luck!

(Leave this table blank.)

Name (1 pt)	
Short Questions (20 pts)	
Long Questions (10 pts)	
VLSI (5 pts)	
Snooping (15 pts)	
Victims (10 pts)	
Token Coherence (40 pts)	
Total (out of 100 pts)	

1 Short Questions (20 pts)

Give short answers in the space provided (about one sentence of answer per question mark).

(a) What is the term to describe having different numbers of sectors on different tracks/cylinders of a disk? Why do they do this?

(b) What is the term to describe having extra tracks or sectors on a disk, that the controller substitutes in place of bad tracks or sectors? Why do they do this, instead of shipping perfect disks?

(c) In the Shah, Giaccone, and Prabhakar, "Efficient Randomized Algorithms..." paper, they use what they call a "diagonal traffic load". A truly diagonal load would have each input i wanting to talk only to output i. Under that scenario, what is the probability that algorithm Random I finds the optimum schedule? Random II? Roughly how well would Random III fare after n iterations (for an $n \times n$ switch)? Roughly how well would Random IV fare after n iterations, with $\eta = (n-1)/n$?

(d) Ethernet allows variable-length messages. Why is there a minimum message size (messages must be longer than some number of bits)? Why is there a maximum message size?

(e) What fraction of peak bandwidth is lost, on average, in a cross-bar switch if we allow variable-length packets?

(f) What are the 4 C's of cache misses?

(e) What is the single word that describes why we can have a lower-level cache snooping the bus without worrying about what is contained in higher-level caches?

(extra credit) Which paper that we read mentions UBC? Where exactly is UBC mentioned?

2 Long Questions (Short Answers) (10 pts)

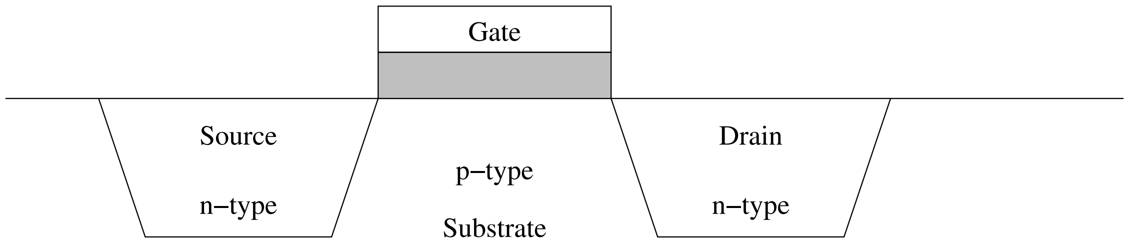
(a) In campus restaurants, I sometimes get stuck waiting in line behind someone ordering a fancy coffee drink, while everyone waits for a fancy coffee maker to complete some elaborate brewing process. Meanwhile, I just want to buy a can of pop and leave, but I have to wait. What paper we read described a similar performance bottleneck, what was it called, and how do you fix it?

(b) On the front page of the newspaper *USA Today* (2004-October-11), there was an infographic, claiming that "Based on past average per capita GDP growth rates for the past 20 years, world income would increase nearly 300% by 2050." The current world income is US\$35 trillion (35×10^12), and in the year 2050, it would be over US\$135 trillion. Show how you would compute the annual growth rate based on these figures. How does the growth rate compare to Moore's Law 2?

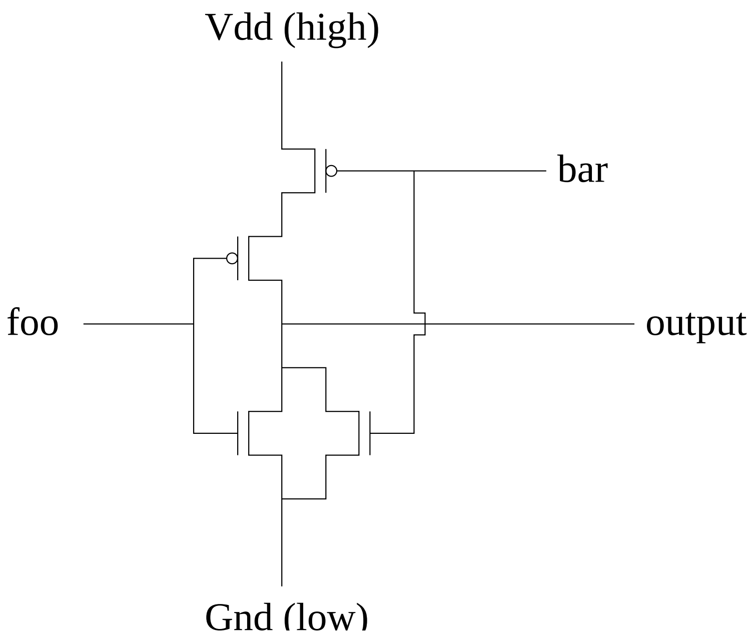
(c) In 1980, IBM started development of what became the IBM PC. At the time, IBM was still the dominant computer company in the world, but people felt they had fallen behind, with the success of personal computers from Apple, Tandy, and Commodore. IBM traditionally developed their own processors and operating systems in-house, but for the IBM PC, they purchased these from Intel and Microsoft, which at the time were small companies, and which became major companies thanks to IBM's decision. Which paper did we read that described similar business circumstances and a similar decision? What is the reason that is common to both cases for the company not to develop everything themselves?

3 VLSI (5 pts)

Is the following a diagram of an n-type or a p-type MOS transistor? (Hint: n-type conducts when the input is high; p-type conducts when the input is low.)



What logic function does the following CMOS circuit implement? (foo and bar are the inputs.)



4 Extra RAID (15 pts)

Imagine a scheme like RAID 1, except we have mirroring over 3 disks (or $G = 1$ and $C = 2$ in the paper's notation). Any data written to the "data disk" is also written to the two "check disks", so they always hold the same information.

(a) Let m_d be the MTTF for each disk, and let m_r be the MTTR for either one or two disks (i.e., you can replace two disks at once). Derive a formula for the MTTF for the group using this RAID scheme. You may make the same simplifying assumptions used in the RAID paper (e.g., independent, exponential distributions for failure times, that m_d is much larger than m_r, approximating $(1 - e^{-x})$ by x for $0 < x << 1$, etc.).

(b) Does this scheme improve reliability over RAID 1? (yes/no)

(c) Does this scheme improve availability over RAID 1? (yes/no)

(d) Compute the performance characteristics for this scheme, using the same notation and analysis as in the paper. I'll answer the first one to get you started:

Total Number of Disks	$3D$
Overhead Cost	
Usable Storage Capacity	

Events/Sec vs. Single Disk

	Full RAID	Efficiency Per Disk
Large Reads		
Large Writes		
Large R-M-W		
Small Reads		
Small Writes		
Small R-M-W		

5 Victim Caches (10 pts)

Assume a machine with 32-bit addresses, byte-addressable, with 16-byte cache lines. The L1 cache is 128KB, and is 2-way set associative. The L1 is backed up by an 8-entry, fully associative victim cache.

(a) Show how an address is divided into the different fields needed to access the cache (byte offset, index, tags). How many bits are in each field? What order are they in?

(b) What happens when the processor attempts to read address 0x00001234 (That's a number in hexadecimal – let me know if you don't know what this means.) and this misses in the L1 but hits in the victim cache? State exactly which lines in the L1 and victim caches change, and what their new values are. (You may refer to the original values in the L1 using strings like L1S2-0000, for the original value in the L1, set 2, line number 0000 (please use hexadecimal). Similarly, you may refer to the original data values in the victim cache using strings like V-0000, where 0000 is the tag of the value in the victim cache, and original memory values with strings like mem0000, where 0000 is the address. Note that the number of digits I've shown for the various address and tag sizes may not be the correct number needed.) Assume set 1 is the one evicted.

(c) What happens when the processor attempts to read address 0x00002234, and this misses in both the L1 and the victim cache? Again, state exactly which lines in the L1 and victim caches change, and what their new values are.

6 Token Coherence (40 pts)

This problem relates to the paper "Token Coherence: Decoupling Performance and Correctness" by Martin et al.

(a) What is the issue being addressed in this paper?

(b) What is the proposed solution? (Describe the solution. Don't just say, "Token Coherence"!)

(c) How do the authors substantiate that their solution works? Summarize their main findings.

(d) How does Token Coherence gain a performance advantage over a directory protocol like DASH?

(e) When might a directory protocol have better performance?

(f) Both snooping and TokenB assume a broadcast communication network. What is the key different assumption that TokenB makes about the network, which means traditional snooping won't work?

(g) What cache state in Goodman's paper (or MESI terminology, if you prefer) corresponds to having one token in Token Coherence? Zero tokens? Half of the tokens?

(h) Rank snooping, DASH, and TokenB in terms of scalability (least scalable to most scalable). Justify your ordering.

(i) The paper states "Since many commercial workloads exhibit abundant thread-level parallelism,...." Given an example of such a commercial workload that we have studied in class.

(j) Briefly explain the difference between a persistent request and a transient request. Which should occur more often in practice?

(k) The paper points out that snooping orders transactions (determines which request goes first) based on their order at a bus, whereas directory protocols order transactions based on their order at the directory. Where does the ordering of transactions occur for TokenB?

(l) What can go wrong if a processor is allowed to write a block when it has $T - 1$ tokens? Give a specific scenario.

(m) With the "Optimized token counting" scheme in Section 3.2, would the protocol still operate correctly (no coherence violations) if there were two owner tokens in the system? (Assuming storage for both owner tokens, etc.) Why or why not?

(n) In TokenB, how should a processor with some tokens respond to a request for exclusive access to that block?

(o) A common data access pattern is the producer-consumer model, where one processor writes data to a memory location and another processor reads that location. Describe how TokenB would behave in this model, with and without the migratory sharing optimization.

(p) In the performance comparison of TokenB to Directory and Hammer (Section 6, Question 3), the authors repeatedly mention avoiding the "third interconnect traversal" for cache-to-cache misses. Describe the 3 interconnect traversals needed for Directory or Hammer. How many interconnect traversals does TokenB require (in the common case)?

(q) In Section 7, the authors note that they could reduce the bandwidth requirements for Token Coherence by using a destination-set predictor to guess which processors need to be multicast a request message. (E.g., a predictor could guess which processors have a given cache block cached, and we would send the request to tokens only to those processors, instead of broadcasting to everyone.) Why is this idea much easier to implement in the Token Coherence framework, rather than in the DASH framework (when they run out of directory pointers and rely on broadcast)?