Study Guide for the EDA284 V20 Examination

Miquel Pericàs, EDA284 Examiner

March 10, 2020

This study guide highlights the topics of each lecture that may appear in the exam. The written examination will consist of conceptual questions and practical exercises. The practical exercises will be of the style you have seen in the practice sessions. In the conceptual questions you will be asked to concisely describe some aspects from the lecture taken from the following list:

- **Lecture 2**
  - What metrics are used to characterize performance and how do they differ? Why is it important to use benchmark sets when comparing computers?
  - Amdahl's Law and Gustafson's Law: Definition. How are they related? What motivates Gustafson's Law?
  - Vector Architecture and Memory System. Difference between SIMD and Vectors. Why use vector/SIMD architectures instead of Out-of-Order?
  - *Out of scope: ARM SVE*
- **Lecture 3**
  - Only the parts on multi-level caches that are specific to multiprocessors: inclusive, non-inclusive & exclusive caches, non-blocking caches, and write policies (in L4)
  - *This lecture was mostly review of DAT105 so most of its contents are outside the exam's scope*
- **Lecture 4**
  - Snoop-based cache coherence: Basic idea. Protocols: Valid-Invalid, MSI, MESI, MOESI. What is the meaning of each state?
  - Advanced topics: update-based protocols, impact of inclusion/exclusion and write policy on multi-level caches
- **Lecture 5**
  - True/false sharing. Essential vs non-essential misses.
  - Scalable cache coherence (directory protocols): Why bus does not scale? Basic concept of scalable protocols. Presence-flag vector scheme. ccNUMA protocols. Optimizations to reduce memory requirements: Limited pointer protocol, Coarse Vector Scheme, Directory Cache.
- **Lecture 6**
  - Explain the purpose of the roofline model. What is the difference between the simplified (DRAM) roofline model and the hierarchical roofline model.
- **Lecture 7**

  - Core Multithreading: Why is it necessary? Why is software multithreading not enough to solve the problem?
  - Coarse-grain multithreading vs Fine-grained multithreading vs Barrel processors
  - Simultaneous multithreading. How does it differ from fine-grained MT. Why does it match better with out-of-order processors?

- **Lecture 8**
  - Heterogeneous CMPs. Why is heterogeneity good for? Temporal vs spatial heterogeneity + examples of these techniques
  - CMP Cache considerations: Shared Cache Architecture, Private/shared cache configuration. Cooperative sharing vs destructive interference, Memory Bandwidth.
  - CMP Topologies: Bus-based vs Ring-based vs Mesh-based vs Crossbar. Understand how coherence is implemented in these topologies
  - *Out of scope: Amdahl's law in the multicore era*

- **Lecture 9**
  - *Lecture 9 was a guest lecture and is thus outside the scope of the examination*

- **Lecture 10**
  - GPGPU Programming Model, SIMT and the CUDA Threading Model. Relation between SIMT, Warps and SIMD. Handling Branches.
  - GPU Microarchitecture: SIMT Front-end and SIMD Back-end. Multithreading? What is the problem with the register File, and how to solve it?

- **Lecture 11**
  - Synchronous vs asynchronous message passing: What are the pros/cons?
  - HW support for message passing protocols: DMA, user level messages, dedicated message processors
  - *Out of scope: Infiniband*

- **Lecture 12**
  - Mutual exclusion, Dekker's algorithm and Barrier Synchronization.
  - Basic synchronization theory: acquire, release, and waiting algorithms.
  - Synchronization using Atomics: Why are atomics necessary? Test-and-set and its implications on coherence, T&S with back-off, Test-and-test-and-set. Implementation of T&S.
  - *Out of scope: LL-SC, C++ support for atomics*

- **Lecture 13**
  - Definitions of coherence: strict coherence and plain coherence. How do they differ? Why Forwarding Store Buffers do not break coherence

- **Lecture 14**
  - Why Coherence is not sufficient? What is memory consistency? How does it differ from coherence?
  - Sequential Consistency: Sufficient Condition. Why SC is too strict for HW implementation? Implementation of SC in in-order (5 stage) pipeline.

- Relaxed Memory Consistency Models. Difference between SC, TSO and RMO. Enforcing orders with memory barriers and synchronizing operations. Implementation of these models in in-order (5 stage) pipelines
- *Out of scope: C++ atomics and consistency model*

The lecture slides contain several examples of modern hardware implementing various techniques covered in EDA284. You do not need to memorize any of these examples.