Appendices

A Changes in Version 1.1. – March 21st 2012

To switch from version 1.0 to version 1.1, download the new version and copy a previously written scheduler.c and scheduler.h to the src/ directory in the new version. USIMM Version 1.1 incorporates the following changes (multiple new features and one bug fix):

- The trace format has changed in two ways. Instead of representing each non-memory instruction with an "N" on a new line (version 1.0), each memory instruction line starts with a number that represents the number of preceding non-memory instructions (version 1.1). Also, each memory read instruction line ends with the PC of the instruction that initiated the memory read.
- The instruction PC for a memory read is recorded in the ROB data structure and the request queue data structure. The simulator does nothing with this PC, but a scheduler might potentially find it useful.
- The input/ directory includes billion instruction traces for single-threaded executions for five PARSEC v2.0 pre-compiled binaries. The traces represent the start of the region of interest in each program. It takes tens of minutes to simulate about a billion cycles on modern machines. For short tests, users can simulate a subset of the entire trace. The "runsim" file has been updated to do an example simulation with these new traces.
- The main.c file has been updated to also accept traces for multi-threaded applications (a few multi-threaded applications will be included in the final competition workload). The individual traces of a multi-threaded application must follow a specific naming convention (starting with "MT0..", "MT1..", and so on). Typically, the addresses from each trace are given a unique prefix that matches their core ID. When a multi-threaded application is detected, the addresses from each of those trace files are given a prefix that matches the core ID for thread 0. In other words, addresses from each trace are given different prefixes, except when they belong to the same multi-threaded application.
- The scheduler is now allowed to issue an auto-precharge command in the same cycle as a column-read or column-write. This allows the row to be closed without consuming an additional command bus cycle. The scheduler does this through the is_autoprecharge_allowed( ) and issue_autoprecharge( ) commands.
- The scheduler is allowed to activate an arbitrary row even if there is no pending request in the queues for that row. This is done via the issue_activate_command( ) and is_activate_allowed( ) commands. The number of such speculative activations is tracked by stats_num_activate_spec. By default, we assume that all of these activations are done for future column-reads. This count is therefore used to influence the row buffer hit rates for memory reads.
- Bug fix: The function issue_refresh_command has been updated to correctly set the next_cmd timing constraints based on current DRAM state. In the earlier code, the dram_state was being set to REFRESHING before the state-dependent next_cmd times were being calculated. This change has a negligible impact on performance.