RTMemController 1.0

Yonghui Li[1], Benny Akesson[2], Kees Goossens[1]
[1]Eindhoven University of Technology, The Netherlands
[2]Czech Technical University in Prague, Czech Republic

May 8, 2014

This software manual is used to provide detailed information about RTMemController, such that users can modify this tool according to their needs.

1 The inputs of RTMemController

RTMemController requires as inputs memory traces and memory specifications, which should be specified by the user. Example input files are included in the folders "memtraces/" and "memspecs/", respectively.

memtraces/: This folder contains 12 Mediabench [1] application traces, each of which has a transaction size of 32 bytes, 64 bytes or 128 bytes. A trace presents every transaction on a separate line, including the time interval (in cycles) between successive transactions (RequestInterval), the transaction type (AccessType: read or write), logical address in decimal and the transaction size in bytes. There are more fields in the input files, while they are not relevant to this tool. Note that this tool uses these traces based on the assumption that the SDRAM and the processors running the applications have the same frequency. If the frequencies are different, users have to convert the RequestInterval of their own traces into the SDRAM clock cycles.

Users can specify which memory traces are used. By running traces.py, these specified traces representing different requestors are combined with a first-come first-serve (FCFS) mechanism. As a result, a combined trace is generated and given by combinedTrace.dat. For example, users may find the following lines in combinedTrace.dat. Each line provides the information of a transaction in terms of time stamp, type, logical address and size.

⋮
108,read,268455040,128
109,read,4251744,32
⋮

Alternatively, users can also use their own memory traces instead of the combinedTrace.dat for running the command scheduler (see Section 2).

memspecs/: This folder includes 3 different DDR3 memory specifications, each of them provides the necessary architecture parameters and timing constraints of an SDRAM as given by JEDEC DDR3 standard [2]. These memory specifications are identified by their names, which indicate the capacity, type, frequency, and the width of a column. For example, the memory specification JEDEC_2Gb_DDR3-800D_16bit shows that the JEDEC specified DDR3 SDRAM has a capacity of 2 Gb and its frequency is 800 MHz while the speed bin is 800D. Moreover, its column width is 16 bits. In addition to the name, the page size is assumed to be 2 KB. The following lines are included in the JEDEC_2Gb_DDR3-800D_16bit. The format of a line is the name of the parameter and its value.

```
  ⋮
nbrOfRows, 16384
width, 16
AL, 0
CCD, 4
  ⋮
```

Users can add more DDR3 memory specifications in addition to the 3 memory specifications in this folder. DRAMPower [3] has provided many memory specification data sheets, from which users can make their own for using RTMemController.

2 The command scheduler of RTMemController

With the input memory specification and combined memory trace, RTMemController executes every transaction by dynamically scheduling commands according to the algorithm proposed in our paper [4].

DynamicCmdScheduler.cc is the main interface file for users, where the memory specification file and the combined trace file can be specified by command line options. It reads every transaction from the trace file and triggers the AnalyticalScheduler to execute the transaction.

src/Trans.h is a class to identify a transaction with the id, type (read/write), time stamp, logical address and size. Moreover, it contains the class of physical address in terms of bank, row, and column.

src/MemCommand.h describes the format of a command according to the command type, issued time and physical address, etc.

src/AnalyticalScheduler.cc is used to execute each individual transaction by dynamically scheduling commands. Moreover, it calls the WC-scheduler to provide both scheduled and analytical worst-case execution time (WCET) per transaction size. Finally, the actual execution time of

each transaction is calculated, where the maximum one is collected as the measured WCET.

src/WCmdscheduler.cc provides the scheduled WCET based on the worst-case initial bank states according to ALAP scheduling, and the analytical WCET on the basis of the theorems in [4]. These WCET results per transaction size are independent from the input traces.

src/Parameter.cc describes the format of a parameter consisting of id, type and value. It is used to read every parameter record in the memory specification file.

src/MemorySpecification.cc reads the parameters from the specified memory specification file, and configures the parameters in **src/MemArchitectureSpec.cc** and **src/MemTimingSpec.cc**. The former has the memory architecture parameters e.g., burst length, number of banks, data rate, while the latter includes the memory timing parameters.

src/MemCtrlConfig.cc builds a lookup table which consists of the best bank interleaving number (BI) and burst count (BC) per transaction size to achieve lowest execution time.

src/AddressDecoder.cc translates the logical address of a transaction into physical address in terms of bank, row, and column.

3 The outputs of RTMemController

results/ folder collects the command scheduling results of each transaction, and the worst-case and average-case execution time results.

The command scheduling results are presented in **results/cmdSchedulingResults.dat**. For example, a 128 bytes read transaction is executed by scheduling commands to 4 banks, each of which receives an activate commands and 2 read commands. The command scheduling results are given below. Every line consists of the transaction id (2), the command, the bank, the scheduling time and the relative scheduling time. Typically, four commands are used including Activate (ACT), Read (RD), Write (WR) and Refresh (REF). Since an auto-precharge policy is assumed, an auto-precharge flag is attached to the last RD or WR command to each bank. Therefore, these commands are represented by RDA or WRA. The relative scheduling time is calculated based on the scheduling time of the first ACT of a transaction.

```
2 ACT 0 72 0
2 ACT 1 78 6
2 RD 0 80 8
2 RDA 0 84 12
2 ACT 2 85 13
2 RD 1 88 16
```

2 ACT 3 91 19
2 RDA 1 92 20
2 RD 2 96 24
2 RDA 2 100 28
2 RD 3 104 32
2 RDA 3 108 36

results/StatisticsResult.dat collects the WCET and ACET as well as some other relevant information. For example, the below information is obtained for variable transaction sizes with 32 bytes, 64 bytes and 128 bytes. The numbers of the first column are the transaction size. The second column has the transaction type (Read or Write) that experiences the measured WCET. The following two columns are the BI and BC. The 5^{th} column presents the ACET per transaction size while the overall ACET is given by the 9^{th} column. The measured, scheduled and analytical WCET results per transaction size are provided by the 6^{th}, 7^{th} and 8^{th} columns, respectively. The 10^{th} column shows the number of transactions according to their sizes, while the last column gives the total number of transactions.

32 Read 2 1 33.6556 42 46 47 35.0253 19734 208450
64 Write 4 1 31.8695 49 58 61 35.0253 96984 208450
128 Read 4 2 38.6564 58 68 68 35.0253 91732 208450

References

[1] C. Lee et al., *MediaBench: a tool for evaluating and synthesizing multimedia and communications systems.* in Proc. MICRON, 1997.

[2] *DDR3 SDRAM Specification*, Jesd79-3e ed., JEDEC Solid State Technology Association, 2010.

[3] Karthik Chandrasekar, Christian Weis, Yonghui Li, Benny Akesson, Norbert Wehn, and Kees Goossens. *DRAMPower: Open-source DRAM power and energy estimation tool.* URL: http://www.drampower.info.

[4] Yonghui, Li and Benny, Akesson and Kees, Goossens. *Dynamic Command Scheduling for Real-Time Memory Controllers.* In Proc. Euromicro Conference on Real-Time Systems (ECRTS), 2014.