An Analyzable Memory Controller for Hard Real-Time CMPs

Marco Paolieri, Eduardo Quiñones, Francisco J. Cazorla, and Mateo Valero

***Abstract*—Multicore processors (CMPs) represent a good solution to provide the performance required by current and future hard real-time systems. However, it is difficult to compute a tight WCET estimation for CMPs due to interferences that tasks suffer when accessing shared hardware resources. We propose an analyzable JEDEC-compliant DDRx SDRAM memory controller (AMC) for hard real-time CMPs, that reduces the impact of memory interferences caused by other tasks on WCET estimation, providing a predictable memory access time and allowing the computation of tight WCET estimations.**

***Index Terms*—CMP, DDRx SDRAM, hard real-time, memory controller, worst case execution time (WCET).**

I. Introduction

NOWADAYS hard real-time embedded systems require more performance than what is currently provided by embedded processors [1]. Multicore processors (CMPs) have shown to be an effective solution due to their low cost and good performance-per-watt ratio. Moreover, in CMPs, the core design is maintained relatively simple preventing timing anomalies [8]. However, it is harder to provide a tight worst case execution time (WCET) for CMPs than for single-core processors due to *inter-task interferences* accessing hardware shared resources[1]. As a consequence, the execution time of a task may increase depending on the other tasks running simultaneously inside the processor: it becomes extremely difficult or even impossible to perform a tight WCET analysis for a given task.

Previous solutions [9], [10] focus on the effect of interferences caused by on-chip shared resources, like caches and buses, on the WCET estimation. In [9], we showed that on-chip interferences caused by CMPs with shared L2 cache increase the WCET estimation up to $1.29\times$ with respect to the WCET estimation computed in isolation, i.e., assuming no intertask conflicts accessing shared resources. However, in a CMP the interferences caused by off-chip shared memory system have the most significant impact on the execution time of the running tasks [5] and on the WCET estimation.

In our preliminary experiments, we ran in a four-core CMP, a *hard real-time task* (HRT), coscheduled with three instances of a very high memory-intensive synthetic benchmark which constantly access memory. Our results show an increment up to $2.20\times$ with respect to the WCET estimation computed in isolation. Therefore, if the interferences between tasks in the memory controller (*memory interferences*) are not taken into account, the execution time of a HRT when running in a CMP workload can go beyond its WCET estimation in isolation.

This letter focuses on JEDEC-compliant [7] DDRx SDRAM high speed memories[2] [6] (DDR, DDR2 and DDR3). These types of memories are commonly used in high performance CMPs and the trend is to use them also in embedded systems. Even though a common DRAM memory controller is analyzable for the worst-case response, few works have focused on computing and reducing the actual impact of memory interferences on the WCET estimation (see Section V).

In this letter, we present a JEDEC-compliant DDRx SDRAM *analyzable memory controller* (AMC) for multicore architectures, which reduces the impact that a memory request can suffer due to the memory interferences introduced by other tasks, allowing the computation of tight WCET estimations in a multicore environment. In other words, our AMC has been designed from a WCET point of view, allowing quantifying the impact of any JEDEC-compliant DRAM device on the WCET estimation. The main benefits of AMC are as follows.

1) The WCET estimation of a task is independent of the memory behavior of the other corunning HRTs and *non hard real-time tasks* (NHRTs). Without this independence, changes on one task in the taskset may have unexpected effects on others, making integration and maintenance extremely costly if not impossible in a multicore environment.
2) Since our analysis is based on generic timing constraints, AMC can be used with any JEDEC-compliant DDRx SDRAM devices. This allows quantifying the effect of using different DRAM devices on WCET estimations, and so the designer can choose the best device from a WCET estimation point of view, and not, based on the performance of the average case as it is usually the case.

Manuscript received October 23, 2009; revised November 24, 2009. Current version published February 05, 2010. This manuscript was recommended for publication by A. Raghunathan. This work has been supported by the Ministry of Science and Technology of Spain under contract TIN-2007-60625, by the HiPEAC European Network of Excellence, and by the MERASA STREP-FP7 European Project under the Grant 216415. Marco Paolieri is supported by the Catalan Ministry for Innovation, Universities and Enterprise of the Catalan Government and European Social Funds.

M. Paolieri and M. Valero are with the Barcelona Supercomputing Center, Spain, and the DAC, Universitat Politècnica de Catalunya, Spain (e-mail: marco.paolieri@bsc.es; mateo@ac.upc.edu).

E. Quiñones is with the Barcelona Supercomputing Center, Spain (e-mail: eduardo.quinones@bsc.es).

F. J. Cazorla is with the Barcelona Supercomputing Center, Spain, and the Artificial Intelligence Research Institute, Spanish National Research Council (CSIC), Spain (e-mail: francisco.cazorla@bsc.es).

Digital Object Identifier 10.1109/LES.2010.2041634

[1]In this letter by default, the term *resources* refers to *hardware resources*

[2]Within this letter by default, the term DRAM memory refers to JEDEC-compliant DDRx SDRAM devices

3) AMC reduces the impact of memory interferences on the WCET estimation with respect to other normal memory controllers by 45%.

The rest of the letter is organized as follows. Section II introduces basic concepts about DDRx SDRAM memories, Section III describes the multicore architecture used along the letter and covers in details our proposal. Experimental setup and results are presented in Section IV. Related work is described in Section V. Finally, conclusions are shown in Section VI.

II. DDRx SDRAM Fundamentals

A DDRx SDRAM device [6] is organized as a set of independent memory *banks*, each of them consisting of a *row-buffer* and a two-dimensional array of memory cells organized into *rows* and *columns*. A row is a group of storage cells that are loaded into the row-buffer using a row activate command (ACT). Once a row is open, any number of read (CAS) and write (CWD) commands can be issued to transfer data in and out of the row-buffer. The minimum amount of data that can be transferred in a single read/write command is called *burst*, and its size is called *burst length* (BL). Data is transferred on both the rising and falling edges of the memory clock, thus requiring $\mathrm{BL}/2$ memory cycles. Finally, a precharge command (PRE) *closes* the row-buffer, storing the data back into the memory cells. The *row-buffer* can be managed using two different policies: *close page* and *open page*. Moreover, data values must be periodically read out and restored to the full voltage level, otherwise the memory would be unable to read the values again. This operation is called *refresh* and it is performed through a refresh (REF) periodic command.

III. An Analyzable Memory Controller

In [9] we proposed a CMP architecture, that guarantees by design, that the maximum delay a HRT can suffer because of on-chip interferences caused by internal bus and L2 shared cache is upper bounded. We assumed an idealized memory system with a private memory controller per core, however, in a real memory system, memory interferences may arise in the memory controller that arbitrates, among the requests from different tasks, which one is the next to access the DRAM device. The reason is that chip bandwidth (pins) is one of the most costly resource that must be minimized, so in real chips HRTs and NHRTs have to share those pins originating intertask memory interferences. Therefore, the memory request scheduling policy used is a key factor to guarantee not only the performance, but also the predictability of the memory controller.

To this end, the AMC design is the result of an exhaustive analysis of the *upper bound delay* (UBD) introduced by memory interferences, considering the generic timing constraints defined in the JEDEC standard [6]. AMC implements a *round robin* policy among HRTs so intertask interferences are upper bounded based on the maximum number of tasks that can access simultaneously the memory, which is equivalent to the number of cores in the chip. Regarding NHRTs, the scheduler prioritizes HRTs over NHRTs, so the effect of NHRTs on the WCET estimation is reduced. Moreover, in order to isolate intratask interferences (originated by requests of the same task) from intertask interferences (originated by requests of different tasks), our memory controller uses one request queue per task. By doing this, AMC prevents interaction between the requests of different tasks. Therefore, the maximum delay that a request can suffer because of other requests depends only on the number of queues (cores), i.e., $N_{\mathrm{HRT}} - 1$.

AMC takes advantage of bank interleaving by making every memory request access to all banks (this letter considers a four-banks DRAM device), so DRAM commands can be effectively pipelined. AMC uses the same memory access granularity proposed in [2], interleaving the data along all banks and fixing the granularity equal to $BL \cdot N_{\mathrm{banks}} \cdot W_{\mathrm{BUS}}$ bytes, where N_{banks} corresponds to the number of banks and W_{BUS} to the bus width in bytes. AMC reduces the impact of interferences by implementing a *close-page* policy with all read and write commands issued with *auto-precharge*.

Other techniques (e.g., request bundling) have been proposed to improve the performance of memory controllers. However, when moving to real-time environments, it is required to consider WCET estimation over raw performance. Any additional technique to improve performance would require to be time analyzable. This is out of the scope of this letter, though it is part of our future work.

A. Analyzing the Execution Time of Memory Requests

We define *issue latency*[3] (t_{IL}) as the maximum delay that a request can suffer due to a previous one.

AMC pipelines the DRAM commands of memory requests by accessing to all memory banks (e.g., from bank B0 to bank B3). However, banks cannot be simultaneously activated because of the data bus serialization. A bank is activated every $\mathrm{t_{BURST}}$ cycles such that the data transmission from one bank starts as soon as the transmission from the previous bank has finished. In addition to that, it is required to guarantee that, before activating a bank the previous request has released it. Therefore, $\mathrm{t_{IL}}$ is determined by: (1) the minimum interval time between two consecutive row activations of the same bank, *time issue bank* $(\mathrm{t_{IB}})$; and (2) the data bus serialization that determines the time required to transfer a request, that is $t_{\mathrm{BURST}} \cdot N_{\mathrm{banks}}$ cycles.

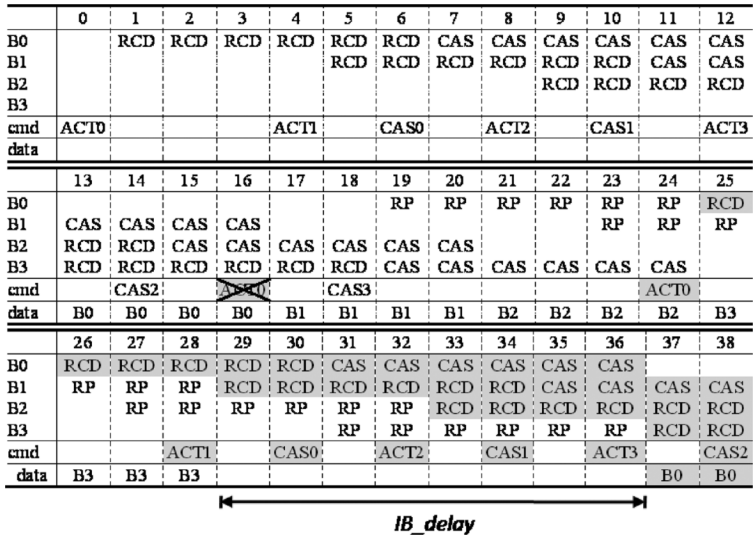
On one hand, the minimum interval time between two activations to the same bank $\mathrm{t_{IB}}$, is at least equal to $\mathrm{t_{RC}}$ cycles. However, depending on the type of the previous request [6], i.e., either a read or a write, the $\mathrm{t_{IB}}$ may increase because the previous request has not finished, resulting into two different expressions:

- $\mathrm{t_{IBR}} = \max\{(\mathrm{t_{RCD}} + \max(\mathrm{t_{BURST}}, \mathrm{t_{RTP}}) + \mathrm{t_{RP}}), \mathrm{t_{RC}}\}$, when the previous request is a read;
- $\mathrm{t_{IBW}} = \max\{(\mathrm{t_{RCD}} + \mathrm{t_{CWD}} + \mathrm{t_{BURST}} + \mathrm{t_{WR}} + \mathrm{t_{RP}}), \mathrm{t_{RC}}\}$, when the previous request is a write.

On the other hand, the data bus serialization also determines the minimum time interval at which every memory request can be issued and it equals $t_{\mathrm{BURST}} \cdot N_{\mathrm{banks}}$. However, if the two consecutive operations are not of the same type, like a *read after write* or a *write after read*, there are additional factors that impact the data bus serialization.

- In case of *write after read*, the time increases because of $(\mathrm{t_{WTR}})$ [6] timing constraint, i.e., when a write request

[3]Notice that this time is different from the *memory latency*, which is the interval between the first bank activation and the last data transfer.



	0	1	2	3	4	5	6	7	8	9	10	11	12
B0		RCD	RCD	RCD	RCD	RCD	RCD	CAS	CAS	CAS	CAS	CAS	CAS
B1						RCD	RCD	RCD	RCD	RCD	RCD	CAS	CAS
B2										RCD	RCD	RCD	RCD
B3													
cmd	ACT0				ACT1		CAS0		ACT2		CAS1		ACT3
data													

	13	14	15	16	17	18	19	20	21	22	23	24	25
B0							RP	RP	RP	RP	RP	RP	RCD
B1	CAS	CAS	CAS	CAS							RP	RP	RP
B2	RCD	RCD	CAS	CAS	CAS	CAS	CAS	CAS					
B3	RCD	RCD	RCD	RCD	RCD	RCD	CAS	CAS	CAS	CAS	CAS	CAS	
cmd		CAS2		~~ACT0~~		CAS3						ACT0	
data	B0	B0	B0	B0	B1	B1	B1	B1	B2	B2	B2	B2	B3

	26	27	28	29	30	31	32	33	34	35	36	37	38
B0	RCD	RCD	RCD	RCD	RCD	CAS	CAS	CAS	CAS	CAS	CAS		
B1	RP	RP	RP	RCD	RCD	RCD	RCD	RCD	RCD	CAS	CAS	CAS	CAS
B2		RP	RP	RP	RP	RP	RP	RCD	RCD	RCD	RCD	RCD	RCD
B3						RP	RP	RP	RP	RP	RP	RCD	RCD
cmd			ACT1		CAS0		ACT2		CAS1		ACT3		CAS2
data	B3	B3	B3									B0	B0

Fig. 1. Time-line of two consecutive read operations (one in white and one in gray) using a four-bank JEDEC-compliant 256 Mb × 16 DDR2-800E SDRAM device [7].

is followed by a read request. t_{WTR} accounts for the time that DRAM requires to allow I/O gating to overdrive the sense amplifiers before the read command can start, switching the direction of the bus. Moreover, an additional constraint described in JEDEC standard [7] should be taken into account: the minimum time interval between an issue of a CWD and a CAS command. For these reasons, a *write after read* involves an additional delay of $t_{WTR} + t_{CAS}$.

- In case of *read after write*, the time increases by one cycle, because t_{CWD} latency is always defined as $t_{CAS} - 1$ cycles [7] and this generates a shift on the data bus by 1 cycle.

In conclusion, t_{IL} requires to consider both the previous and the current memory requests. This results in four different t_{IL} expressions.

- If the previous request is a read and the current too, the *read-to-read issue latency* (t_{ILRR}) is $t_{ILRR} = \max\{t_{BURST} \cdot N_{banks}, t_{IBR}\}$ cycles.
- If the previous request is a read and the current is a write, the *read-to-write issue latency* (t_{ILRW}) is $t_{ILRW} = \max\{t_{BURST} \cdot N_{banks} + 1, t_{IBR}\}$ cycles.
- If the previous is a write and the current too, the *write-to-write issue latency* (t_{ILWW}) is defined as $t_{ILWW} = \max\{t_{BURST} \cdot N_{banks}, t_{IBW}\}$ cycles.
- If the previous is a write and the current is a read, the *write-to-read issue latency* (t_{ILWR}) is defined as $t_{ILWR} = \max\{(t_{BURST} \cdot N_{banks}) + t_{WTR} + t_{CAS}, t_{IBW}\}$ cycles.

An example is shown in Fig. 1. In particular Fig. 1 shows the commands bus (cmd), data bus (*data*), and bank status (B0–B3) of two consecutive read memory requests, one in white and one in grey, on a four-bank JEDEC 256 Mb × 16 DDR2-800E SDRAM device [7]. Even though the experiments shown in Section IV are carried out using a DDR2-400B SDRAM device, we provide here the example using a DDR2-800E memory device to show that our solution can be applied to a different JEDEC-compliant memory system.

The use of a multibank system allows to transfer data from B0 (cycle 13), while other banks are being simultaneously accessed, effectively pipelining the DRAM commands. Thus, each bank is activated every t_{BURST} cycles (at cycles 0, 4, 8, and 12), so the data is transferred in consecutive cycles (from cycle 13 to cycle 28). However, if a new request is ready to be served (in grey), it cannot be issued t_{BURST} cycles after activating B3 (cycle 16, crossed cell in Fig. 1) because t_{IBR} of B0, which in this case is equal to t_{RC} cycles, would be violated. Instead, the new request must wait until cycle 24.

We call this extra delay IB_delay, that can be expressed as $|t_{ILxx} - (t_{BURST} \cdot N_{banks})|$. Note that IB_delay also affects the data bus efficiency, reducing it down to $(t_{BURST} \cdot N_{banks})/t_{ILxx}\%$.

B. Computing the UBD of a Memory Request

When computing a WCET estimation, it is required to take into account the longest possible t_{IL} for every memory request, defined as $t_{ILWORST} = \max\{t_{ILRR}, t_{ILRW}, t_{ILWW}, t_{ILWR}\}$.

By doing this, it is not required to consider the whole sequence of memory accesses of all the tasks that run simultaneously in the processor to know which is the impact of memory interferences on the WCET computation, because the worst-case scenario is always considered. Therefore, the UBD of a memory request only depends on $t_{ILWORST}$, and it depends on the interference caused by others HRTs and NHRTs running at the same time. Given that we prioritize HRT requests over NHRT requests and we apply a round-robin policy between the requests of HRTs, UBD is defined as follows.

- Regarding HRTs, the worst-case scenario occurs when all HRTs that are executed simultaneously in the multicore processor try to access the memory at the same time. In this case the maximum delay that a task can suffer is bounded by the total number of HRTs executed simultaneously $UBD_{HRT} = (N_{HRT} - 1) \cdot t_{ILWORST}$.
- Although NHRTs have lower priority than HRTs, it may happen that a request coming from a HRT arrives just

one cycle after a request from a NHRT was sent to main memory. In this case, the maximum delay is bounded by $\mathrm{t_{ILWORST}}$, that is $\mathrm{UBD_{NHRT}} = t_{\mathrm{ILWORST}} - 1$.

In conclusion, the maximum delay that a memory request from a HRT can suffer due to other tasks is the sum of both $\mathrm{UBD_{HRT}}$ and $\mathrm{UBD_{NHRT}}$ and it is equal

$$\begin{aligned}\mathrm{UBD} &= \mathrm{UBD_{NHRT}} + \mathrm{UBD_{HRT}} \\ &= N_{\mathrm{HRTs}} \cdot t_{\mathrm{ILWORST}} - 1\end{aligned}$$

C. Refresh Operation

The refresh operation is an important source of interferences on DRAM devices. A refresh operation is released every $\mathrm{t_{REFI}}$ cycles and no other command can be issued until it finishes. Thus, depending on the time the refresh occurs, the effect on the WCET of HRTs varies. It is commonly the case that DRAM refreshes lead to increase of the execution time, which must be properly accounted for in real-time systems [4]. Moreover, the exact time in which the refresh operation occurs cannot be statically determined because it depends on when the application starts.

To have a tight and safe WCET estimation, we propose synchronizing the start of a HRT with the occurrence of a refresh operation at analysis and execution time. So in both cases, the start of the task execution is delayed until the first refresh operation takes place. By doing so, the refresh commands will produce the same interferences during the analysis and the execution, as they will occur exactly at the same time with respect to the start of the task. In the worst-case, the task arrives one cycle after the memory has finished a refresh command, and so it must wait $(\mathrm{t_{REFI}} - 1)$. The overall WCET is defined as follows

$$\mathrm{WCET_{TOTAL}} = \mathrm{WCET} + t_{\mathrm{REFI}} - 1$$

where WCET is the WCET estimation obtained using the WCET computation mode.

D. Considering the UBD in the WCET Analysis

AMC implements the WCET *computation mode* [9] in order to consider the UBD into the WCET analysis. When analyzing HRTs, the processor is set in this execution mode and each HRT under study is executed in isolation. In this execution mode, the memory controller artificially delays every memory request by UBD cycles, which depends only on the number of HRTs the task under analysis is going to corun with. For instance, in a WCET computation mode 3, the UBD is computed assuming three HRTs and other NHRTs running at the same time.

Therefore, AMC allows the computation of a safe and tight WCET estimation for a HRT running in a mixed workload because the maximum delay that a request can suffer due to interferences accessing a shared resource, i.e., UBD, is always taken into account. It has been formally proved that even if instructions execute before their estimated time in the worst case, the computed WCET is safe [3], [10]. Moreover, AMC requires no changes to current WCET analysis tools. So the same tools and techniques that are used and are valid for single-core processors can be used in the analysis of multicore processors.

IV. Results

We model a four-bank JEDEC-compliant 256 Mb $\times$ 16 DDR2-400B SDRAM device connected to our CMP [9]. To do so, we have integrated DRAMsim [11] inside our simulation framework [9]. We assume a CPU-SDRAM clock ratio of four, i.e., the clock of the CPU (800 MHz) runs at four times the frequency of the memory (200 MHz). We use a real hard real-time application, a collision avoidance algorithm provided by Honeywell Corporation (Hon), that requires high-performance. It is based on an algorithm for 3D path planning used in autonomous driven vehicles. WCET estimations are computed using RapiTime tool without any modification.

Fig. 2 shows the WCET estimations of RapiTime, for the Hon application as we vary the WCET computation mode from 1 to 4. Concretely, we compute a WCET estimation under different scenarios: (1) assuming a private DDR SDRAM memory controller for each task and having interferences only in on-chip shared resources [9] (labeled as PR_MC); and (2) on-chip and memory interferences are considered at the same time, using AMC shared among HRTs and controlling on-chip resources with the proposal [9] (labeled as *AMC*). In each scenario, for each WCET computation mode, we vary the cache size assigned to the Hon application (from 128 KB to 8 KB), and we show how AMC behaves as the pressure on the memory system increases.

Moreover, in order to evaluate whether the WCET estimations when using AMC are tight, we also measure the maximum observed execution time (labeled as *MOET*) of the HRT when running in a very high memory demanding workload composed by several instances of the *opponent* benchmark, a high memory demanding synthetic benchmark, in which each instruction is a store that systematically misses in the last level cache and hence, it always accesses the main memory.

Each WCET computation mode is compared to its corresponding workload: WCET computation mode 4 corresponds to a workload composed by the HRT under study (Hon application), and 3 instances of the *opponent* running as HRTs and no NHRT; WCET computation mode 3 corresponds to the HRT under study, 2 HRTs and 1 NHRT *opponents*, and so on.

As expected, memory interferences have a tremendous impact on the WCET estimation, significantly higher than on-chip interferences. Thus, in the highest possible memory demanding scenario, i.e., assigning 8 KB of L2 to the HRT and a WCET computation mode of 4 [Figs. 2(a)] the memory interferences increase the WCET estimation from $1.2 \times (\mathrm{PR\_MC})$ to $2.22 \times (\mathrm{AMC})$. This is a WCET increment of at least $1.02\times$. Obviously, as memory pressure decreases, i.e., the cache partition of L2 given to the HRT increases and/or WCET computation mode decreases [Figs. 2(b), (c), (d)], the impact of memory interferences also decreases, reaching the smallest impact when the WCET computation mode is 1.

Although such high impact of memory interferences, AMC allows computing a tight WCET estimation. When comparing the MOET of the Hon application in the highest memory demanding workload, i.e., assigning a cache partition of 8 KB and 3 HRT *opponents* [Figs. 2(a)] with its computed WCET estimation for the corresponding workload, we observe only a difference of $0.5\times$ (from $1.72\times$ to $2.22\times$).

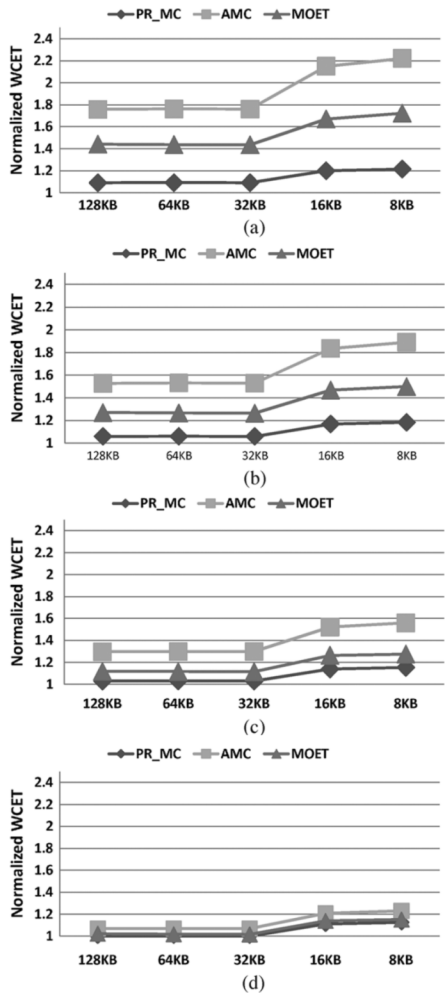


Fig. 2. Normalized WCET Estimation for the Hon application when using a JEDEC-compliant 256 Mb × 16 DDR2-800E SDRAM device. (a) WCET mode 4, (b) WCET mode 3, (c) WCET mode 2, (d) WCET mode 1.

Moreover, AMC reduces the UBD with respect to other memory controllers. Thus, when using an *open-page* policy instead of close page and considering the same access granularity, the UBD of a read memory request is increased up to 42 cycles, which represents an increment of 45%. When setting the access granularity to a single bank instead of multiple banks the UBD increases up to 88 cycles as AMC does.

V. Related Work

Predator [2] is probably the most similar related work. It is a memory controller for multiprocessor system-on-chip that guarantees an user-defined bandwidth requirement to a given task and that requires the user to assign a fixed priority to each task. This solution fits well in streaming or multimedia real-time applications, in which a bandwidth requirement can be easily defined. Predator is targeted only for a specific DRAM device: a JEDEC-compliant 32 Mb × 16 DDR2-400B SDRAM.

Instead, the AMC approach requires neither knowing the bandwidth requirements, nor assigning a fixed priority to each task allowing AMC being applied to control based applications where the bandwidth requirements are not known. Moreover, we provide a generic solution that can be easily applied to any JEDEC compliant DDRx SDRAM device, with any set of timing parameters. That is, our solution defines the UBD of any DRAM device based on the number of HRTs running simultaneously in the processor and generic DRAM timing constraints.

TABLE I
Normalized WCET Estimation of Hon Application Using UBDs of Predator

	128KB	64KB	32KB	16KB	8KB
priority 0	1.36	1.36	1.67	1.67	1.98
priority 1	1.70	1.70	2.31	2.36	2.99
priority 2	2.37	2.39	3.59	3.82	5.02
priority 3	4.77	4.81	7.50	8.38	11.19

Table I shows the impact of Predator on the WCET estimation with respect to the WCET estimation in isolation, i.e., the same baseline of Section IV. In the highest memory demanding scenario, i.e., assigning a cache partition of 8 KB, Predator increases the WCET estimation of the highest priority HRT by $1.98\times$ and by $11.19\times$ for the lowest priority HRT. Instead, when using AMC for the same cache size WCET computation mode 4, the WCET estimation of Hon application increases by $2.22\times$. Therefore, although Predator reduces the WCET estimations of the HRTs with priority 0, it dramatically increases the WCET estimations of HRTs with priority 2 and 3 with respect to AMC.

VI. Conclusion

In this letter, we propose AMC, a JEDEC-compliant DDRx SDRAM analyzable memory controller for CMPs designed with the objective of minimizing the impact of memory intertask interferences on the WCET estimation. AMC can be applied to any JEDEC-complaint DRAM device allowing computing the UBD based on generic timing constraints.

References

[1] 2007 [Online]. Available: www.merasa.org, MERASA EU-FP7 Project:
[2] B. Akesson, K. Goossens, and M. Ringhofer, "Predator: A predictable SDRAM memory controller," in *CODES+ISSS*, New York, NY, USA, 2007.
[3] A. Andrei, P. Eles, Z. Peng, and J. Rosen, "Predictable implementation of real-time applications on multiprocessor systems-on-chip," in *Proc. Int. Conf. VLSI Design*, Washington, D.C., 2008.
[4] P. Atanassov and P. Puschner, "Impact of DRAM refresh on the execution time of real-time tasks," in *Proc. IWARCC*, 2001.
[5] D. Burger, J. R. Goodman, and A. Kägi, "Memory bandwidth limitations of future microprocessors," in *ISCA*, Philadelphia, Pennsylvania, United States, 1996.
[6] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk.*. New York: Kaufmann, 2008.
[7] 2008, JEDEC. DDR2 SDRAM SPECIFICATION JESD79-2E.
[8] T. Lundqvist and P. Stenstrom, "Timing anomalies in dynamically scheduled microprocessors," in *Real-Time Syst. Symp.*, Phoenix, AZ, 1999.
[9] M. Paolieri, E. Quinones, F. J. Cazorla, G. Bernat, and M. Valero, "Hardware support for WCET analysis of hard real-time multicore systems," in *Int. Symp. Comput. Arch.*, Austin, TX, Jun. 2009.
[10] J. Rosen, A. Andrei, P. Eles, and Z. Peng, "Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip," in *Real-Time Syst. Symp.*, Tucson, AZ, 2007.
[11] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "Dramsim: A memory system simulator," in *SIGARCH Comput. Archit. News*, 2005.