A Predictable and Command-Level Priority-Based DRAM Controller for Mixed-Criticality Systems

Hokeun Kim, David Broman, Edward A. Lee,
Michael Zimmer, Aviral Shrivastava and Junkwang Oh

Presented by Hokeun Kim, Dept. of EECS, UC Berkeley
RTAS 2015, April 13-16, 2015, Seattle, WA

Introduction

- **Mixed-Criticality Systems**
  - Tasks with different criticality
  - Sharing the same hardware
  - To save costs (space, weight, energy, etc.)
- **Competing Requirements in Mixed-Criticality**
  - Critical tasks – time predictability (hard real-time)
  - Non-critical tasks – high performance

Introduction

- **DRAMs in Mixed-Criticality Systems**
  - Larger and cheaper than SRAMs
  - Good for saving costs
- **Variable Latency of DRAMs**
  - Translation into different DRAM commands
  - Memory request scheduling
  - DRAM refreshes

Contributions

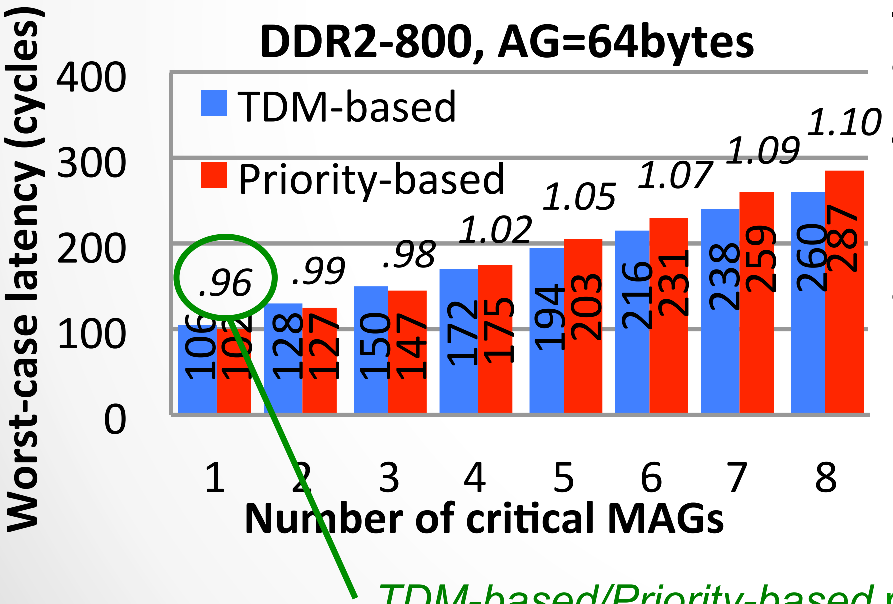
- **In This Paper, We Propose…**
  - A DRAM controller for mixed-criticality
  - With tight worst-case latency bounds for *critical tasks*
  - While providing significantly higher performance for *non-critical tasks*
  - Compared to a recent advanced approach based on time-division multiplexing (TDM) with command patterns
    - S. Goossens et al., “A reconfigurable real-time SDRAM controller for mixed time-criticality systems”, CODES+ISSS 2013
- **We also propose…**
  - Algorithms to compute worst-case latencies for the proposed DRAM controller

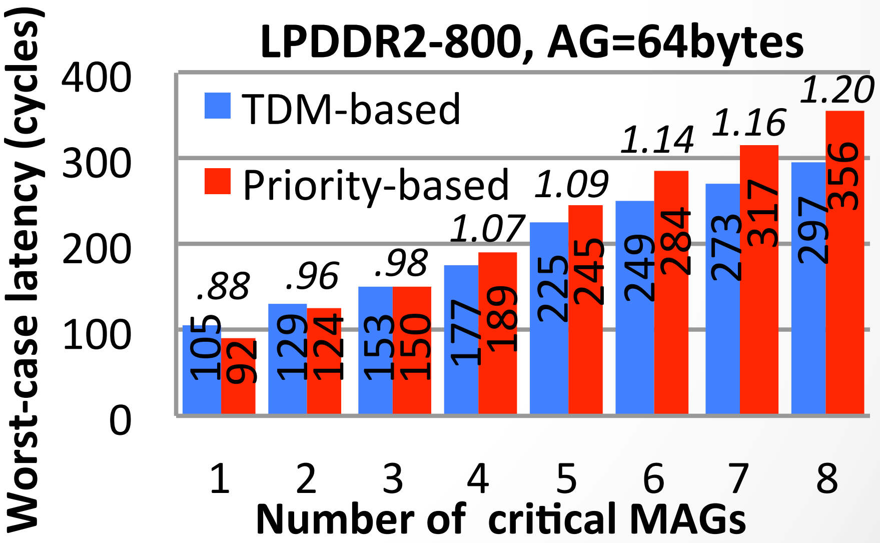
Contributions

- **Comparable Worst-case Latency Bounds**

Without any special care for critical tasks?

Could be unpredictable and drastically higher!
(depending on scheduling and refresh)

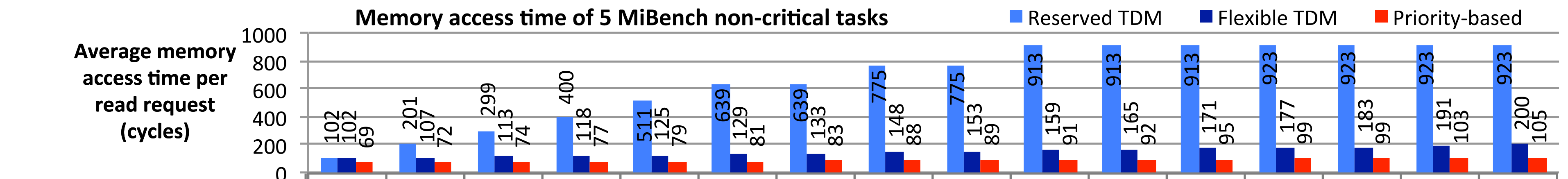




TDM-based/Priority-based ratio

Contributions

- **Significantly Higher Performance (= Less Memory Access Time)**



Number of critical MAGs	0	1	2	3	4	5	5	6	6	7	7	7	8	8	8	8
Number of critical tasks (safety critical, mission critical)	0 (0, 0)	1 (1, 0)	2 (2, 0)	3 (3, 0)	4 (4, 0)	5 (4, 1)	6 (4, 2)	7 (4, 3)	8 (4, 4)	9 (4, 5)	10 (4, 6)	11 (4, 7)	12 (4, 8)	13 (4, 9)	14 (4, 10)	15 (4, 11)
Priority-based / Reserved TDM	0.67	0.36	0.25	0.19	0.16	0.13	0.13	0.11	0.12	0.10	0.10	0.10	0.11	0.11	0.11	0.11
Priority-based / Flexible TDM	0.67	0.67	0.66	0.65	0.63	0.63	0.62	0.59	0.58	0.57	0.56	0.56	0.56	0.54	0.54	0.52

- 33%~89% less memory access time, depending on the number of critical tasks

Background - DRAM Basics

- **DRAM Bank**
  - A group of DRAM arrays that are accessed independently
- **DRAM Array**
  - Consists of rows, and columns within each row
- **DRAM Row Buffer**
  - Stores a DRAM row after row activation
- **Row Buffer Management Policies**
  - Open-page policy
    - Keep rows activated after access, better for exploiting locality
  - Close-page policy
    - Keep rows precharged after access, better for random accesses

Background - DRAM Basics

- **Important DRAM Commands**
  - PRECHARGE, ACTIVATE, READ, WRITE, REFRESH
- **DRAM Request Scheduling (Reordering)**
  - FRFCFS – Exploit bank parallelism
  - OpenRow – Exploit locality
- **Timing constraints between commands**
  - Minimum time delays between commands
  - Must be satisfied for correct DRAM operations
- **Types of timing constraints**
  - *Intra-bank* (for commands to the same bank)
  - *Inter-bank* (for commands to different banks)

Related Work

- **Software-based Approaches**
  - SW-based bank privatization & priority scheduling
    - H. Kim et al., “Bounding memory interference delay in COTS-based multi- core systems”, RTAS 2014
  - SW-based bank privatization (by allocating virtual pages to private banks)
    - H. Yun et al. “PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms”, RTAS 2014

Related Work

- **Hardware-based Approaches**
  - Bank privatization + Fixed TDM (Time Division Multiplexing) slots
    - J. Reineke et al., "PRET DRAM controller: Bank privatization for predictability and temporal isolation", CODES+ISSS 2011
  - Command pattern + Fixed TDM slots
    - B. Akesson and K. Goossens, "Architectures and modeling of predictable memory controllers for improved system integration", DATE 2011
  - Command pattern + Static priority scheduling
    - B. Akesson et al., "Real-time scheduling using credit-controlled static-priority arbitration", RTCSA 2008
  - Request-level scheduling + Close page + Priority
    - M. Paolieri et al., "Timing effects of DDR memory systems in hard real-time multicore architectures: Issues and solutions", ACM TECS 2013
  - Command pattern + Dynamically assigned TDM slots
    - S. Goossens et al., "A reconfigurable real-time SDRAM controller for mixed time-criticality systems", CODES+ISSS 2013

Technical Approach

(1) Bank-Aware Physical Address Space Allocation

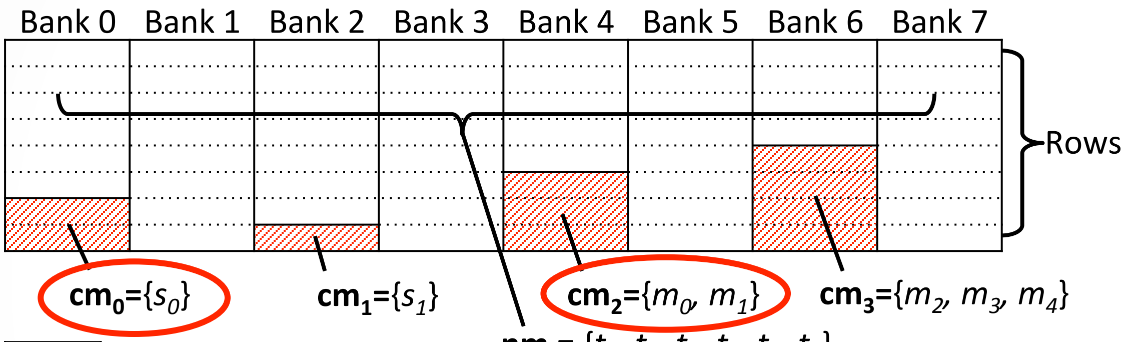
- **For Proposed DRAM Controller, We Define…**
  - Two types of physical memory space
    - Critical space - Reserved for critical requests and prioritizing them
      - At most one critical space per bank, to limit inter-bank interference
    - Non-critical space
  - Memory Access Groups (MAGs)
    - Critical MAG – A set of critical tasks, mapped to one critical space
    - Non-critical MAG – A set of non-critical tasks
  - Categories of criticality for *tasks*
    - Critical – Latency upper bound is guaranteed
      - Safety critical - One task per critical MAG
      - Mission critical – ≥ one task per critical MAG
    - Non-critical – Processed by schedulers for high performance

Technical Approach

(1) Bank-Aware Physical Address Space Allocation

- **Critical Space Allocation & Task Mapping Example**

Physical Address Space of a DRAM



Bank 0
Bank 1
Bank 2
Bank 3
Bank 4
Bank 5
Bank 6
Bank 7
Rows
$\mathbf{cm_0}=\{s_0\}$
$\mathbf{cm_1}=\{s_1\}$
$\mathbf{cm_2}=\{m_0, m_1\}$
$\mathbf{cm_3}=\{m_2, m_3, m_4\}$
nm = $\{t_0, t_1, t_2, t_3, t_4, t_5\}$

□ Non-critical space

▨ Critical space

cm$_i$: Critical memory access groups

nm: Non-critical memory access group

s_i: Safety critical tasks

m_i: Mission critical tasks

t_i : Non-critical tasks

- **Representing Critical Space**
  - Representation with a 32-bit register for a 8-bank DRAM

	Bank 0	Bank 1	Bank 2	Bank 3	Bank 4	Bank 5	Bank 6	Bank 7
[31:24] 8 bits	[23:21]	[20:18]	[17:15]	[14:12]	[11:9]	[8:6]	[5:3]	[2:0]

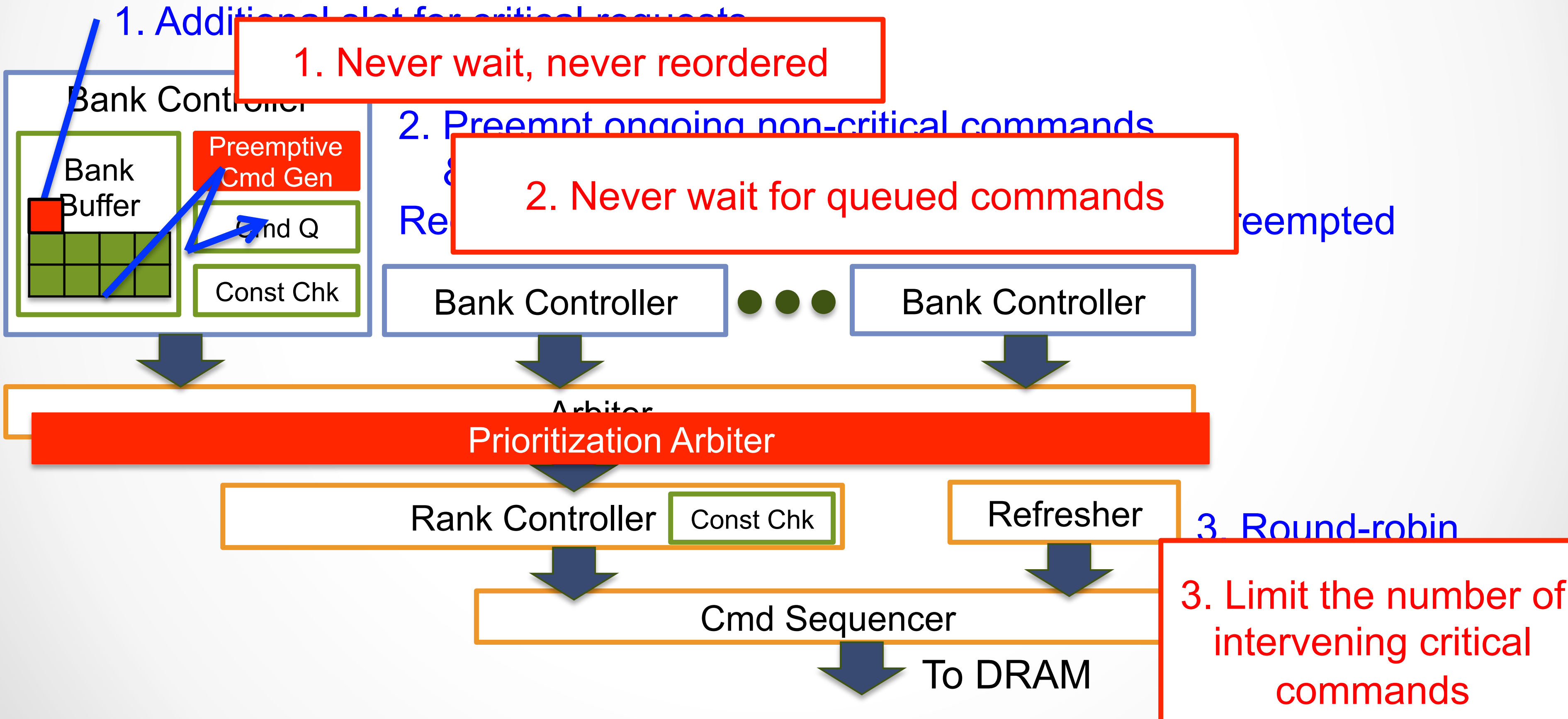
1 bit for each bank

Number of rows reserved for each bank

Technical Approach

(2) Command-Level Prioritization of Critical Requests

- **Modifications In *Proposed* DRAM Controller**
  - ***How worst-case latency is bounded?***



1. Additional slot for critical requests

1. Never wait, never reordered

2. Preempt ongoing non-critical commands

&

Re

reempted

2. Never wait for queued commands

3. Round-robin

3. Limit the number of intervening critical commands

Bank Controller

Bank Buffer

Preemptive Cmd Gen

Cmd Q

Const Chk

Bank Controller

Bank Controller

Arbiter

Prioritization Arbiter

Rank Controller

Const Chk

Refresher

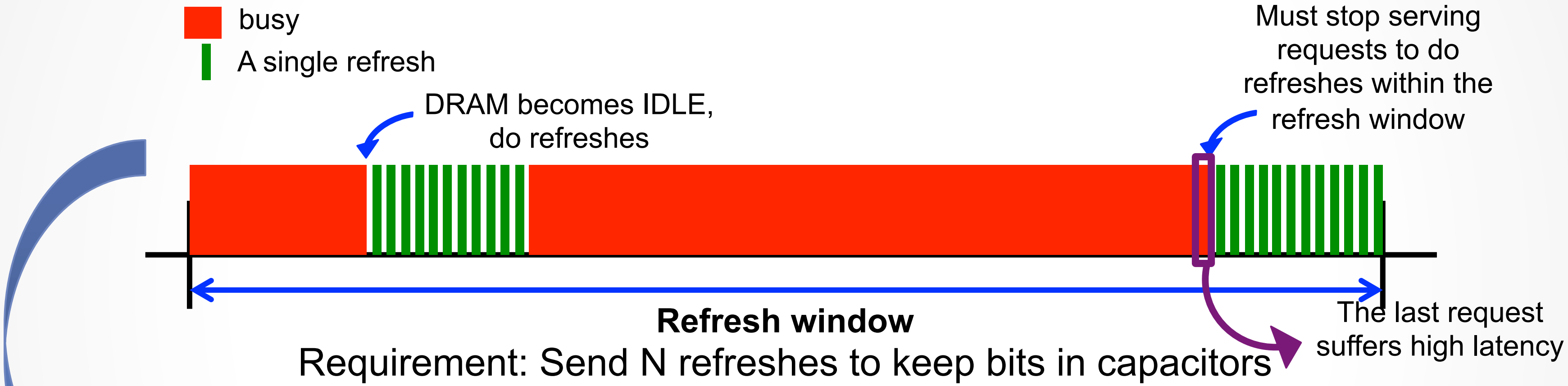
Cmd Sequencer

To DRAM

Technical Approach

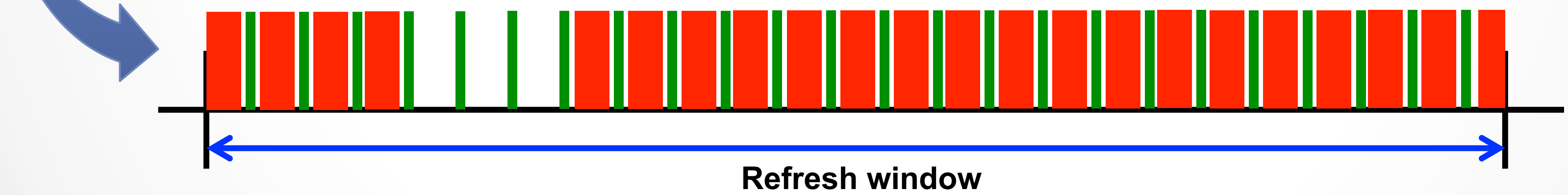
(3) Making DRAM Refresh Predictable

- **Refresh Scheduling for High Throughput**


🟥 busy
🟩 A single refresh
DRAM becomes IDLE, do refreshes
Must stop serving requests to do refreshes within the refresh window
Refresh window
The last request suffers high latency

Requirement: Send N refreshes to keep bits in capacitors

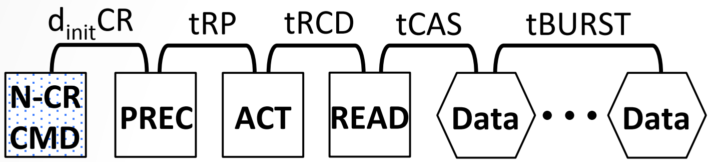
- **Distributing Refresh uniformly**



  - Bound effect of refresh on latency
  - At a cost of slightly higher average latency

Worst-case Bound Analysis

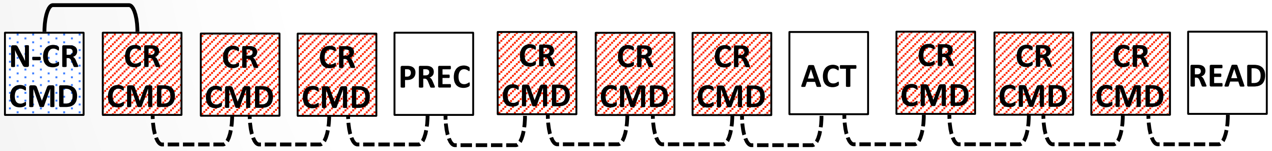
- **Finding Worst-case Latency**
  - Worst-case DRAM Command Sequence



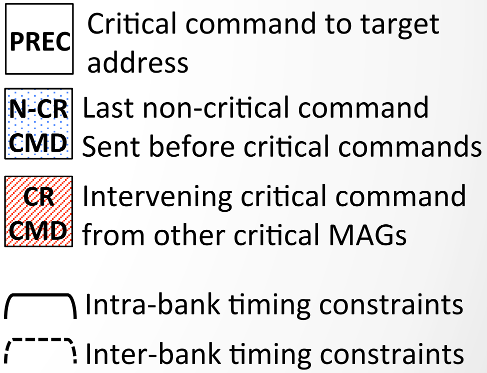
(a)

  - Maximum Number of Interventing Critical Commands

“# Critical MAG – 1” for each command



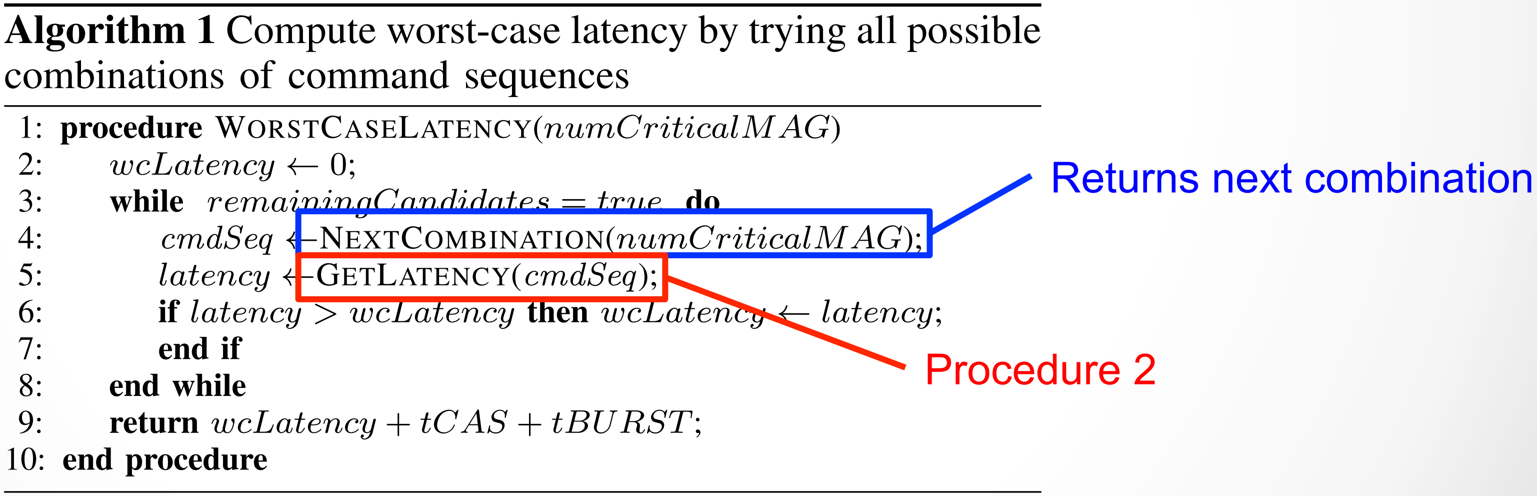
(c)


PREC
Critical command to target address
N-CR CMD
Last non-critical command Sent before critical commands
CR CMD
Intervening critical command from other critical MAGs
Intra-bank timing constraints
Inter-bank timing constraints

  - Worst-case Combination
    - Each intervening command can be either PRECHARGE, ACTIVATE, READ, or WRITE
    - We propose *mechanical procedures* for this!

Worst-case Bound Analysis

- **Procedures to to Compute Worst-case Latency**
  - Procedure 1: Iterate through all combinations to find the worst-case

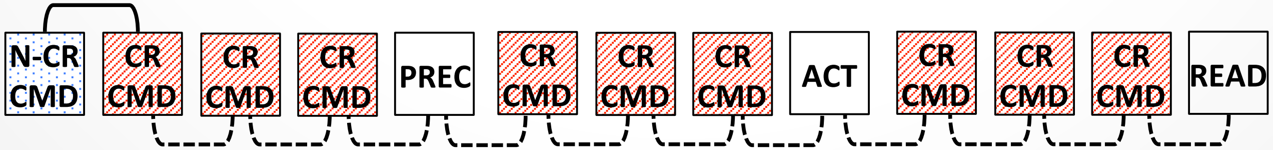


Algorithm 1 Compute worst-case latency by trying all possible combinations of command sequences

```
procedure WorstCaseLatency($numCriticalMAG$)
    $wcLatency \leftarrow 0$;
    while $remainingCandidates = true$ do
        $cmdSeq \leftarrow$ NextCombination($numCriticalMAG$);
        $latency \leftarrow$ GetLatency($cmdSeq$);
        if $latency > wcLatency$ then $wcLatency \leftarrow latency$;
        end if
    end while
    return $wcLatency + tCAS + tBURST$;
end procedure
```

Returns next combination

Procedure 2



(c)

Worst-case Bound Analysis

- Procedure 2: Compute latency of a given combination

Algorithm 2 Get latency to send all commands in *cmdSeq*

```
procedure GetLatency($cmdSeq$)
    int d[len($cmdSeq$)];
    d$\leftarrow$ 0;                                ▷ initialize array elements to zero
    for $i = 1$ to len($cmdSeq$)$-1$ do
        for $j = i - 1$ down to 0 do
            $(cmd_{from}, bank_{from}) \leftarrow cmdSeq[j]$;
            $(cmd_{to}, bank_{to}) \leftarrow cmdSeq[i]$;
            if $bank_{from} = bank_{to}$ then
                $t \leftarrow$ d[$j$]$+$$intraDelay(cmd_{from}, cmd_{to})$;
            else
                $t \leftarrow$ d[$j$]$+$$interDelay(cmd_{from}, cmd_{to})$;
            end if
            if $t >$ d[$i$] then d[$i$]$\leftarrow$t;
            end if
            if (d[$i$]$-$d[$j$]) $\geq maxDelay$ then break;
            end if
        end for
    end for
    return d[len($cmdSeq$)$-1$];
end procedure
```

Matrices for timing constraints for each command pair

- **Timing Constraints Example (LPDDR2-800MHz)**
  - Intra-bank timing constraints (cycles)

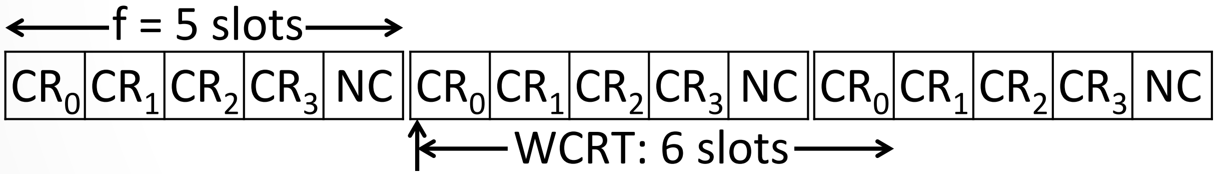
From \ To	READ	WRITE	PRECHARGE	ACTIVATE
READ	8	15	9	N/A
WRITE	16	8	18	N/A
PRECHARGE	N/A	N/A	N/A	6
ACTIVATE	6	6	17	N/A

  - Inter-bank timing constraints (cycles)

From \ To	READ	WRITE	PRECHARGE	ACTIVATE
READ	8	8	1	1
WRITE	16	8	1	1
PRECHARGE	1	1	1	1
ACTIVATE	1	1	1	4

Worst-case Bound Analysis

- **Modeling Competing Approach for Comparison**
  - TDM slot assignment for memory accesses
    - One TDM slot for each critical MAG
    - One TDM slot for non-critical MAG (to minimize worst-case bounds while supporting non-critical tasks)
  - Example with 4 critical MAGs (f: frame size)



f = 5 slots

CR_0	CR_1	CR_2	CR_3	NC	CR_0	CR_1	CR_2	CR_3	NC	CR_0	CR_1	CR_2	CR_3	NC

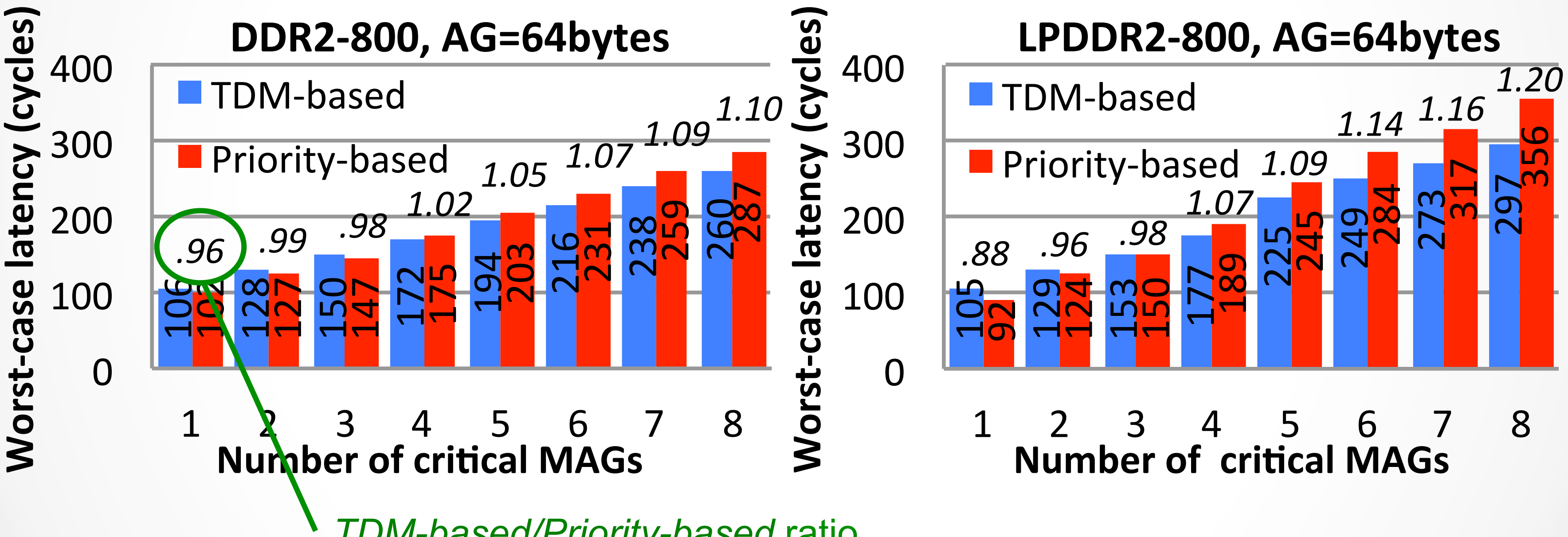
WCRT: 6 slots

Worst-case arrival time for a critical request from CR_0

  - Worst-case latency bound estimation
    - (f + 1) x slot size (cycles)
    - Slot sizes are estimated based on papers on the competing approach

Worst-case Bound Analysis

- **Results on Two Different DRAMs**

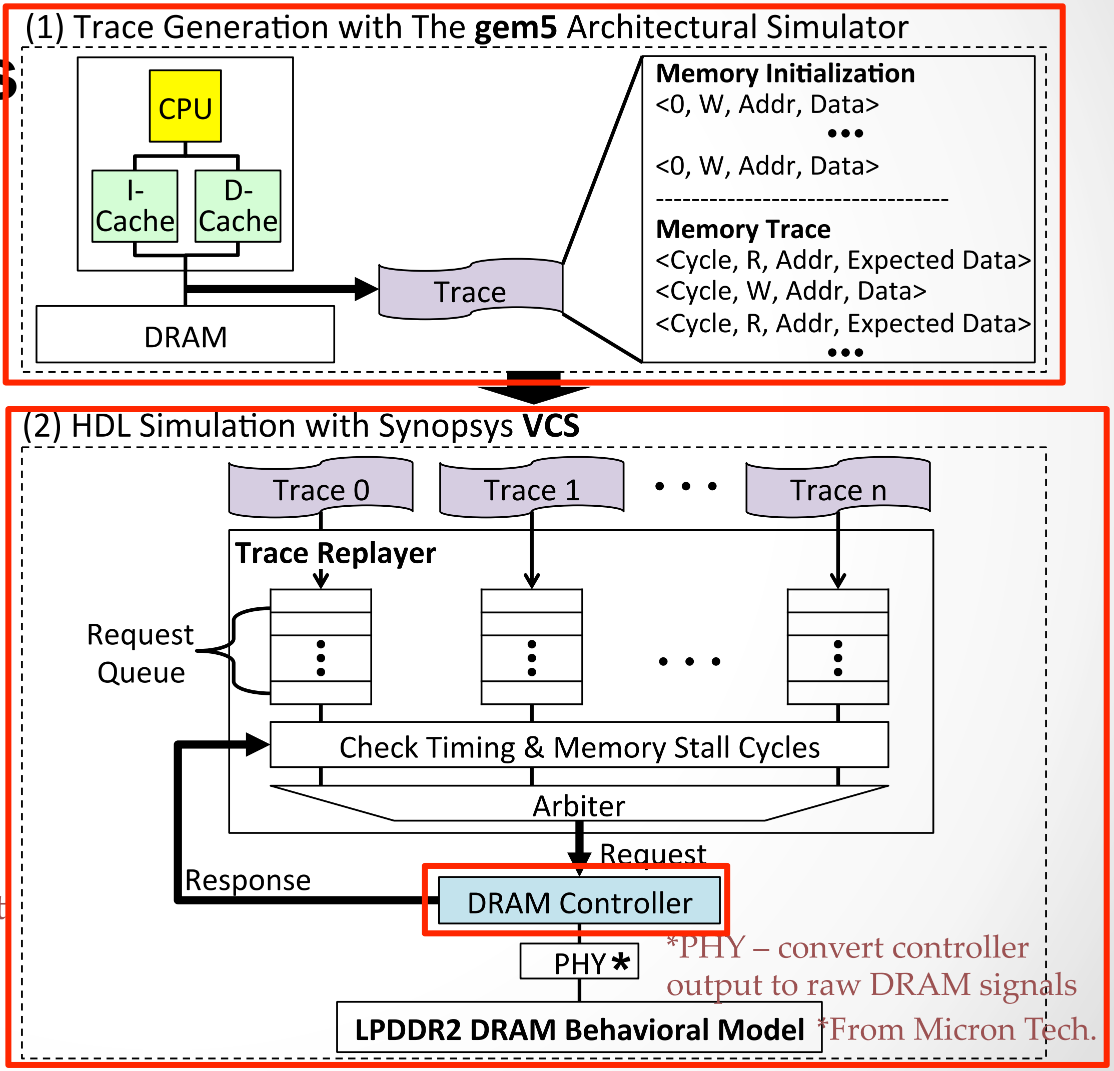


TDM-based/Priority-based ratio

Experiments and Results

- **Flow of experiments**
  - (1) Trace generation
  - (2) HDL simulation
- **DRAM controller implementation**
  - Proposed
    - Chisel* ➔ Verilog RTL
  - TDM-based approach
    - Verilog behavioral

*Chisel – a Scala embedded HDL developed at UC Berkeley, can generate Verilog RTL



(1) Trace Generation with The **gem5** Architectural Simulator
CPU
I-Cache
D-Cache
DRAM
Trace
Memory Initialization
<0, W, Addr, Data>
•••
<0, W, Addr, Data>
Memory Trace
<Cycle, R, Addr, Expected Data>
<Cycle, W, Addr, Data>
<Cycle, R, Addr, Expected Data>
•••
(2) HDL Simulation with Synopsys **VCS**
Trace 0
Trace 1
• • •
Trace n
Trace Replayer
Request Queue
Check Timing & Memory Stall Cycles
Arbiter
Request
Response
DRAM Controller
PHY*
LPDDR2 DRAM Behavioral Model

*PHY – convert controller output to raw DRAM signals

*From Micron Tech.

Experiments and Results

- **Benchmarks Used for Trace Generation**

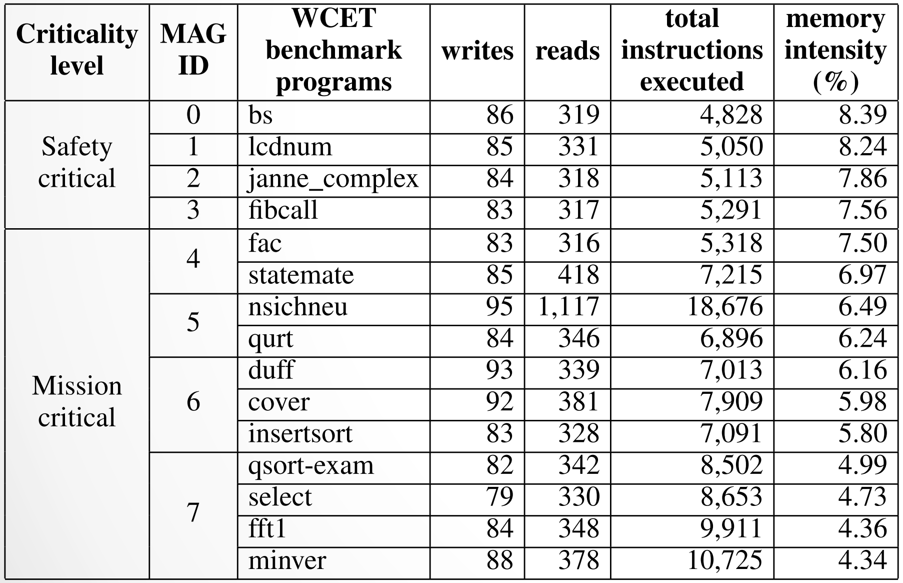
○ Mälardalen WCET benchmark

- For safety critical and mission critical tasks

○ MiBench

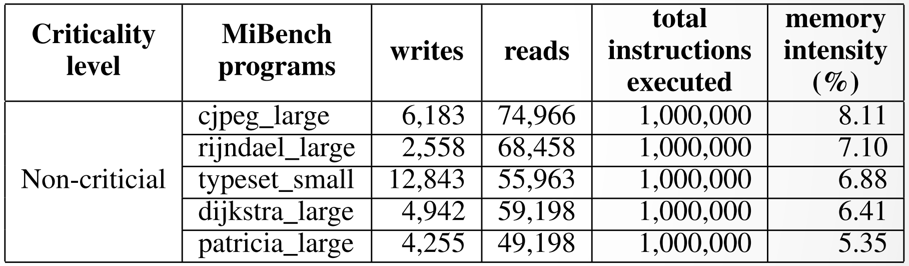
- For non-critical tasks

TABLE I. LIST OF BENCHMARKS USED AS CRITICAL TASKS



Criticality level	MAG ID	WCET benchmark programs	writes	reads	total instructions executed	memory intensity (%)
Safety critical	0	bs	86	319	4,828	8.39
	1	lcdnum	85	331	5,050	8.24
	2	janne_complex	84	318	5,113	7.86
	3	fibcall	83	317	5,291	7.56
Mission critical	4	fac	83	316	5,318	7.50
		statemate	85	418	7,215	6.97
	5	nsichneu	95	1,117	18,676	6.49
		qurt	84	346	6,896	6.24
	6	duff	93	339	7,013	6.16
		cover	92	381	7,909	5.98
		insertsort	83	328	7,091	5.80
	7	qsort-exam	82	342	8,502	4.99
		select	79	330	8,653	4.73
		fft1	84	348	9,911	4.36
		minver	88	378	10,725	4.34

TABLE II. LIST OF BENCHMARKS USED AS NON-CRITICAL TASKS



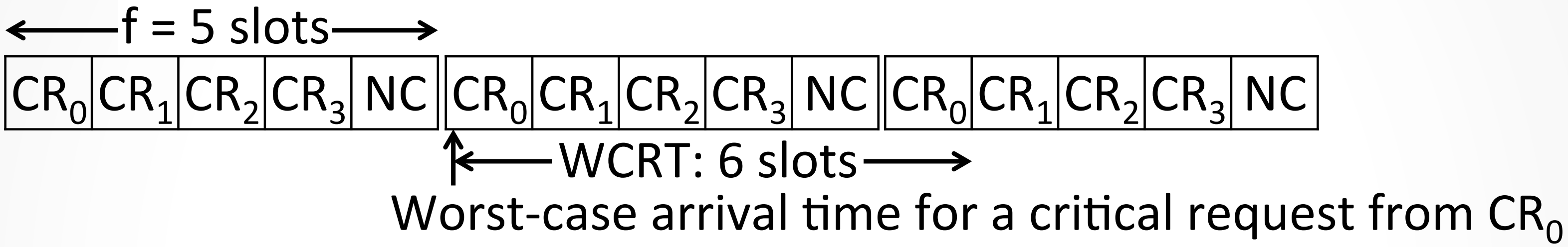
Criticality level	MiBench programs	writes	reads	total instructions executed	memory intensity (%)
Non-criticial	cjpeg_large	6,183	74,966	1,000,000	8.11
	rijndael_large	2,558	68,458	1,000,000	7.10
	typeset_small	12,843	55,963	1,000,000	6.88
	dijkstra_large	4,942	59,198	1,000,000	6.41
	patricia_large	4,255	49,198	1,000,000	5.35

Tasks with highest “memory access / instruction” are selected

* Critical tasks are repeated periodically,
safety critical every 250k cycles, mission critical every 500k cycles.

Experiments and Results

- **Competing TDM-Based Approach Modeling**



f = 5 slots

CR_0	CR_1	CR_2	CR_3	NC	CR_0	CR_1	CR_2	CR_3	NC	CR_0	CR_1	CR_2	CR_3	NC

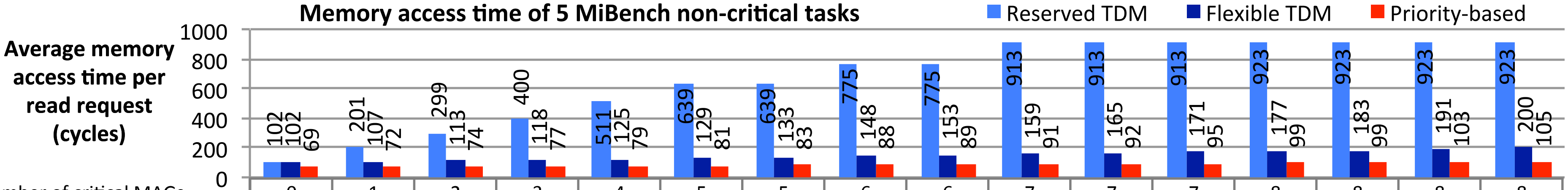
WCRT: 6 slots

Worst-case arrival time for a critical request from CR_0

  - ○ *Reserved TDM*
    - Each slot is only used by an assigned MAG
  - ○ *Flexible TDM*
    - Extension for our experiments
    - Idle slots for critical MAGs may be used by non-critical MAG

Experiments and Results

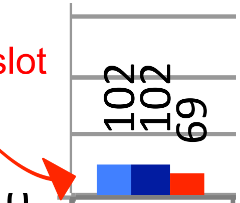
- Average memory access times of non-critical tasks



Number of critical MAGs	0	1	2	3	4	5	5	6	6	7	7	7	8	8	8	8
Number of critical tasks (safety critical, mission critical)	0 (0, 0)	1 (1, 0)	2 (2, 0)	3 (3, 0)	4 (4, 0)	5 (4, 1)	6 (4, 2)	7 (4, 3)	8 (4, 4)	9 (4, 5)	10 (4, 6)	11 (4, 7)	12 (4, 8)	13 (4, 9)	14 (4, 10)	15 (4, 11)
Priority-based / Reserved TDM	0.67	0.36	0.25	0.19	0.16	0.13	0.13	0.11	0.12	0.10	0.10	0.10	0.11	0.11	0.11	0.11
Priority-based / Flexible TDM	0.67	0.67	0.66	0.65	0.63	0.63	0.62	0.59	0.58	0.57	0.56	0.56	0.56	0.54	0.54	0.52

Increasing number of critical tasks (safety critical, mission critical)

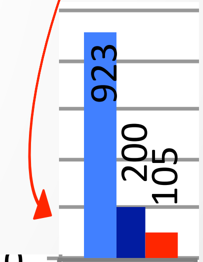
Even when there's only one slot for non-critical, still due to restrictions



Number of critical MAGs	0
Number of critical tasks (safety critical, mission critical)	0 (0, 0)
Priority-based / Reserved TDM	0.67
Priority-based / Flexible TDM	0.67

(1) Priority-based **vs** Reserved TDM
- TDM slots are wasted even when there's no critical request

(2) Priority-based **vs** Flexible TDM
- Due to restrictions (command patterns, close-page)



Number of critical MAGs	8
Number of critical tasks (safety critical, mission critical)	15 (4, 11)
Priority-based / Reserved TDM	0.11
Priority-based / Flexible TDM	0.52

Conclusion

- **Advantages of Proposed DRAM Controller**
  - Guarantee worst-case bounds that are comparable to a recent advanced technique, can help WCET analysis
  - Higher performance for non-critical tasks than the competing approach
- **How Can Our Proposed DRAM Controller Outperform for Non-Critical Tasks?**
  - Almost no overhead (e.g. certain page management policies, fixed command patterns) for guaranteeing worst-case latency bounds for critical tasks
  - Benefits from scheduling techniques for achieving high performance

Q & A

- **Thank you for your attention!**