A Requests Bundling DRAM Controller for Mixed-Criticality System

by: Danlu Guo, Rodolfo Pellizzoni

April 23, 2017
RTAS 2017
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Introduction

- Multicore architecture
  - Shared DRAM main memory
  - Inter-core memory interference

- Real-Time system
  - Hard Real-Time (HRT) applications
  - Soft Real-Time (SRT) applications

- What do we want from DRAM
  - Tighter upper bound latency for HRT request
  - Better lower bound bandwidth for SRT request

- Solution:
  - Innovative predictable DRAM controllers
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
DRAM Background

- **Organization**
  - Channel: Independent DRAM controller
  - Rank: Share Command/Data Bus
  - Bank: Access in Parallel
  - Row, Column, Row Buffer: data cells
DRAM Background

- **Operation**
  - Activate (ACT): retrieve data
  - Column-Access-Strobe (RD/WR): access data
  - Precharge (PRE): restore data
  - Timing Constraints (DDR Specifications)

- **RD [0,0,1]**

\[
A \quad t_{RCD} \quad R \quad t_{RL} \quad \text{Data} \quad t_{RTP} \quad P
\]
DRAM Background

- **Page Policy**
  - Close-Page: Precharge (PRE) after access (CAS)
    - Close (Miss)
  - Open-Page: Precharge (PRE) when required
    - Open (Hit)
DRAM Background

- Data Allocation
  - Shared Banks
    - Allows data sharing among cores
    - Contention on the same bank
  - Private Bank
    - Allows isolation between cores/banks
    - Limits data sharing
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Predictable DRAM Controllers Evaluation

- Shared bank + Close-Page

- Private Bank + Open-Page
Predictable DRAM Controllers Evaluation

- Private Bank + Open-Page

- Private Bank and Open-Page + CAS reordering [L.Ecco & R. Ernst, RTSS’15]

Ex: DDR3-1600H
RD-RD: 4
RD-WR: 7
WR-RD: 18
Predictable DRAM Controllers Evaluation

- Current Analytical Model

- Pipeline System

Not the actual command arrival time

HRT Latency Objective
Predictable DRAM Controllers Evaluation

- Mixed Criticality System
  - Co-existing of HRT and SRT applications on different cores
  - Fixed priority can guarantee the HRT latency but limit SRT bandwidth

Bank0
Bank1
Bank2
Bank3

Bank4
SRT Request

Request
Request
Request
Request
Request
Request
Request

SRT Bandwidth
Objective

Starvation
Objective Summary

- HRT Latency:
  - Apply **Pipelining** can cover the overlap interference.
  - Apply **Reordering** can avoid the repetitive CAS switching.

- SRT Bandwidth:
  - Apply **Co-schedule** of SRT and HRT requests can avoid the starvation.

---

Requests Bundling DRAM Controller

Reordering CAS breaks the execution sequence
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Classification
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Requests Bundling (REQBundle) DRAM Controller

HRT Latency

- Isolation
  - Private bank

- Pipelining and Reordering
  - Close-Page
    => Fixed command sequence
  - Reordering on the request level
    => Avoid multiple switching
    => Fixed request sequence

SRT Bandwidth

- Fast Access
  - Shared bank + Open-page

- Co-schedule SRT and HRT requests
  - Fixed SRT execution slots before HRT
Command Scheduler

- HRT Banks
- SRT Banks

InRound Scheduler
OutRound Scheduler

Schedule HRT & SRT Commands
Bundle same type of requests
Switch access type between round

Switch
Starts
ends/Start

Bank0
Bank1
Bank2
Bank3
SRT Bank

Write

OutRound

InRound

Schedule SRT Commands only

Write

Write
InRound Scheduler

- Execution Time of an InRound
  - $t_{\text{Snapshot}}$: time to determine the number of HRT requests (N)
  - $t_{\text{CAS}}^{\text{SRT}}$: time to issue the last SRT CAS
  - $t_{\text{ACT}}^{\text{issue}}$: time to issue the last HRT ACT
  - Execution time $R(N) = \max(t_{\text{switch}}^{\text{SRT}} + (N-1) \times t_{\text{CCD}}^{\text{SRT}}, t_{\text{ACT}}^{\text{issue}} + t_{\text{RCD}}^{\text{SRT}})$
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Case 0: Arrives before snapshot of same type of round

\[ L_{Req} = R(No) + t_{RL} + t_{Bus} \]
Case 1: Arrives before/after snapshot of different type of round

\[ L_{\text{Req}} = R(\text{No}) + R(\text{N1}) + t_{RL} + t_{Bus} \]
Case 2: Arrives after snapshot in the same type of round

\[ L_{\text{Req}} = R(\text{No}) + R(N1) + R(N2) + t_{RL} + t_{Bus} \quad \text{(Worst Case)} \]
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Evaluation

- Implemented in a general DRAM controller simulation framework in C++
  - [DRAMController Demo RTSS’16]
- EEMBC benchmark memory traces generated from MACsim
  - CPU 1GHz
  - Private L1/2 Cache
  - Shared L3 Cache
- Evaluate against Command Bundling (CMDBundle) DRAM Controller
  - [L.Ecco and R.Ernst, RTSS’15]
  - Burst Mode
  - Non-Burst Mode
Benchmark Worst Case Execution Time (8 HRTs)

- HRT0 runs benchmark trace and other 7 HRTs run memory intensive traces
- Normalized on CMDBundle (non-burst)
Worst Case HRT Request Latency (8 HRTs)

- RD Request
- WR Request

<table>
<thead>
<tr>
<th>DDR3 Device</th>
<th>800D</th>
<th>1066E</th>
<th>1333G</th>
<th>1600H</th>
<th>1866K</th>
<th>2133L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read HR(Burst)</td>
<td>0.64</td>
<td>0.61</td>
<td>0.54</td>
<td>0.48</td>
<td>0.37</td>
<td>0.26</td>
</tr>
<tr>
<td>Write HR(Burst)</td>
<td>0.76</td>
<td>0.75</td>
<td>0.67</td>
<td>0.58</td>
<td>0.4</td>
<td>0.31</td>
</tr>
<tr>
<td>Read HR(NBurst)</td>
<td>0.83</td>
<td>0.83</td>
<td>0.8</td>
<td>0.74</td>
<td>0.65</td>
<td>0.55</td>
</tr>
<tr>
<td>Write HR(NBurst)</td>
<td>0.93</td>
<td>0.93</td>
<td>0.88</td>
<td>0.8</td>
<td>0.67</td>
<td>0.61</td>
</tr>
</tbody>
</table>
Worst Case SRT Requests Bandwidth (8 HRTs)

- **RD Bandwidth**

- **WR Bandwidth**
Mixed-Criticality System (8 HRTs, 8 SRTs)

- **HRT Latency**

  ![HRT Latency Diagram]

- **SRT Bandwidth**

  ![SRT Bandwidth Diagram]

- Implement virtual HRT requestor mechanism for CMDBundle
  - Considered as a HRT cores in the system
  - All SRT requests share the virtual requestors
Outline

- Introduction
- DRAM Background
- Predictable DRAM Controller Evaluation
- Requests Bundling DRAM Controller
- Worst Case Latency Analysis
- Evaluation
- Conclusion
Conclusion

- Employing request bundling with pipelining can improve the worst case request latency.

- Considering the command timing constraints gaps can provide a good trade-off between the SRT bandwidth and HRT latency.

- Compared with a state-of-the-art real-time memory controller and show the balance point based on the row-hit ratio of a task.
  - Measurement row hit ratio is lower than 50%. A guaranteed row hit ratio requires static analysis and is lower than measured ratio.
THANK YOU