Conservative Open-Page Policy for Mixed Time-Criticality Memory Controllers

Sven Goossens¹, Benny Akesson ², and Kees Goossens ¹

¹ Eindhoven University of Technology
² CISTER (Porto, Portugal)
Mixed-Time Criticality

• Embedded multi-core systems are getting more complex:
  – Integrating more applications
  – Applications get more complex
  – Functionality / Energy demand increases
• Driven by power, area and cost constraints

• Results in a mix of applications of different time-criticalities sharing hardware resources
  – Firm real-time + Soft real-time = Mixed real-time

⇒ The hardware can no longer be tailored for a specific time-criticality class
SDRAM Controllers

- DRAM: Most commonly used off-chip memory resource
  - Shared across FRT and SRT
- Performance metrics: **bandwidth** (throughput) and **latency** (response time)
- Difficult to bound performance:
  - One reason: **locality dependent**

**Firm Real-Time Controllers**
- Maximize **worst-case** performance
- Simple / analyzable command scheduler
- No attention for average-case performance
- Do not exploit locality

**Soft Real-Time Controllers**
- Maximize **average-case** performance
- Complex high performance command scheduler
- Guaranteeable performance is usually low
- Exploit locality as much as possible

**Mixed Real-Time Controllers: requirements**
- For FRT: guarantee **enough** worst-case performance to satisfy requirements
- For SRT: **maximizing** the average-case performance

**How can locality be exploited by a MRT controller?**
Outline

Introduction

SDRAM

Conservative Open-Page Policy

Experiments / Results

Conclusions
SDRAM Commands

- SDRAM consists of banks, rows and columns
  - Banks share their command, data, and address bus
- A row has to be opened or **activated** before it is accessible
- To open a different row, the old one has to be closed by **precharging**
  - Either using **explicit PRE** command or with an **auto precharge-flag** on a RD/WR
- Timing constraints enforce a minimum distance between the commands

6 commands:
- activate (ACT)
- precharge (PRE)
- read (RD)
- write (WR)
- refresh (REF)
- NOP
Memory accesses

- It is hard to reason about individual commands due to the many timing constraints
- One approach from the FRT-controller domain is to group commands into patterns, and use those to derive the real-time properties of the memory controller.

- The required granularity is often larger than 1 burst, which enables bank-parallelism
- The properties of a pattern are influenced by:
  - The number of banks a request is interleaved over (Banks Interleaved, BI)
  - The number of bursts per bank (Burst Count, BC)

<table>
<thead>
<tr>
<th>Cmd bus:</th>
<th>The number of Banks Interleaved (BI) in the pattern (2 in this example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT0 4x NOP</td>
<td>RD0 3x NOP</td>
</tr>
</tbody>
</table>

Burst Count (BC)
Page Policies

• Close-page policy
  – Precharge active row as soon as possible after a request, using auto-precharge
  – Used in FRT memory controllers
  – Minimizes the execution time of requests that target a different row in the same bank
  – **Side effect:** maximizes the execution time of requests targeting the same row in the same bank!

• Open-page policy
  – Keep active row open until address for next request is known
  – Used in SRT memory controllers
  – Minimizes the execution time of requests that target the same row in the same bank
  – If an open row is targeted sufficiently often, the policy outperforms the close-page policy
  – Worst-case is worse than that of an close-page policy
• Color indicates locality (and request origin)
• For the blue requestor the open-page policy:
  – Increases the worst-case execution time
  – Reduces the average-case execution time
Introduction

SDRAM

Conservative Open-Page Policy

Experiments / Results

Conclusions
Conservative Open-Page policy

- Key idea:
  - Do not precharge if next request is known to target the open row
  - Precharge if next address is not known in time, or in case of a miss

Request arrivals:

1 2 3 4
What does “in time” mean?

- We do not want to reduce the guarantees given by the close-page policy
  - The cycle at which the next row can be activated in the conservative open-page policy may not be later than that of the close-page policy
  - Assume a miss if the next address is not known before the cycle where a close-page policy would precharge

Example:

- Hit window (14 cc)
- ACT-to-ACT constraint = 38 cycles
- Orange cycles contain auto-precharge flags

- If a request arrives within the hit window, we can omit the extra NOP’s at the end of the current schedule, and the initial tRCD cycles of the next schedule
  - Can we do even better?
• Use **explicit precharges instead of auto-precharge flags**
• Postpone the precharge as long as possible

---

**ACT-to-ACT constraint = 38 cycles**

**Hit window (14 cc)**

**Hit window (28 cc)**

**PRE-to-ACT = 10**

---

• In the paper we provide a heuristic that determines the maximum PRE-cycle for a known close-page schedule at design time
• A run-time command scheduler would have to use its constraint checker
The hit window size depends on:
- The type of access (read or write)
- The controller configuration (BI, BC)
- Whether the previous access was a hit or a miss:

The paper contains the obtained hit-window sizes for a range of controller configurations.
Outline

Introduction
SDRAM
Conservative Open-Page Policy
Experiments / Results
Conclusions
Benchmark set analysis

<table>
<thead>
<tr>
<th>Trace</th>
<th>adpcm</th>
<th>aes</th>
<th>bf</th>
<th>gsm</th>
<th>jpeg</th>
<th>mips</th>
<th>motion</th>
<th>sha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. bandwidth MB/s</td>
<td>846</td>
<td>878</td>
<td>253</td>
<td>1910</td>
<td>100</td>
<td>1577</td>
<td>2426</td>
<td>236</td>
</tr>
<tr>
<td>#requests</td>
<td>645</td>
<td>742</td>
<td>873</td>
<td>644</td>
<td>1685</td>
<td>541</td>
<td>617</td>
<td>791</td>
</tr>
</tbody>
</table>

- Spatial locality per trace for 3 controller configurations, interleaving over 1, 2 and 4 banks respectively.
Experimental setup

- Traces recorded using SimpleScalar
- Trace player allows at most 4 outstanding requests, runs at 1400 MHz
- Memory: DDR3-1600x16 module, running at 800 MHz
- Pattern based memory controller (Predator)
Results (single application)

- First (striped) bar: percentage of potential locality that is exploited
- Second bar: conservative open-page execution time reduction
Results (single application)

- 70% of potential locality captured on average
- 17% average execution time reduction
  - Max: 33% (motion)
  - Min: 1% (jpeg)
- Depends on memory load of the application, effectiveness scales with how memory intensive an application is

<table>
<thead>
<tr>
<th>Trace</th>
<th>jpeg</th>
<th>mips</th>
<th>motion</th>
<th>sha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Bw MB/s</td>
<td>100</td>
<td>1577</td>
<td>2426</td>
<td>236</td>
</tr>
<tr>
<td>#requests</td>
<td>1685</td>
<td>541</td>
<td>617</td>
<td>791</td>
</tr>
</tbody>
</table>
Results (multi-application)

- 4 applications, running simultaneously (mips, motion, jpeg, bf)
- multi-tdm-1: work-conserving TDM arbiter, 4 slots, 1 slot per application
- multi-tdm-2: work-conserving TDM arbiter, 8 slots, 2 consecutive slots per application
Results (multi-application)

- Fine grained interleaving destroys locality in the tdm-1 experiment.
  - 25% of locality captured
  - Negligible (total) execution time reduction
- 2 consecutive slots in the table per application → more locality exploitation:
  - 54% of locality captured
  - 7% Total execution time reduction
  - Max: 27% (mips)
  - Min: 2.6% (jpeg)
- Note that changing the arbiter in this way trades-off worst-case latency for average-case latency!

⇒ The policy can be successfully applied in multi-application use cases, if the arbiter allows some requests of the same source to be scheduled consecutively
Controller configuration influence

- Single application runs, 64-byte access granularity configurations are tested
  - Higher BI $\rightarrow$ Higher worst-case bandwidth
  - Higher BI $\rightarrow$ Higher amount of **potential** spatial locality
  - Higher BI $\rightarrow$ Smaller hit-window size

- The absolute difference with the execution time in the worst configuration is only 0.3%

$\rightarrow$ The differences are so small, that a configuration can be selected based on its worst-case performance, without hurting the average case.
Conclusions

Firm Real-Time Controllers
- Maximize worst-case performance
- Simple / analyzable command scheduler
- No attention for average-case performance
- Do not exploit locality
- Close-page policy

Soft Real-Time Controllers
- Maximize average-case performance
- Complex high performance command scheduler
- Guaranteeable performance is usually low
- Exploit locality as much as possible
- Open-page policy

Mixed Real-Time Controllers: requirements
For FRT: guarantee **enough** worst-case performance to satisfy requirements
For SRT: **maximizing** the average-case performance
Exploit locality as long as it does not hurt worst-case performance using a **conservative open-page policy**
Conclusions

• **Conservative Open-Page policy can be used in a MRT controller:**
  – Worst-case guarantees are equal to a close-page policy
  – Average-case performance is better, leading to lower execution times
  – The execution time reduction depends on the memory load of the application

• The policy can be successfully applied in multi-application use cases
  – Assuming that the arbiter allows some requests of the same application to be scheduled consecutively
  – **Changing the arbiter in this way trades off worst-case request latency for average-case request latency**

• The controller configuration (Banks Interleaved, Burst Count) has little influence on the exploited locality
  – A **configuration can be selected based on its worst-case performance, without hurting the average case, so the right choice can be made at design time**