Classification and Analysis of Predictable Memory Patterns

Benny Akesson, Williston Hayes Jr., and Kees Goossens
Eindhoven University of Technology
MPSoC design gets increasingly complex
- Number of applications in a device is increasing
- More resources enable increased application-level parallelism
  - More processors, hardware accelerators, and memories
  - Many applications execute concurrently
- Some applications have (hard) real-time requirements
  - Missing a deadline results in significant quality degradation
Applications **share resources** in the system to reduce cost
- Resource sharing results in **interference** between applications

Verification is typically done by system-level simulation
- **All use-cases** must be verified instead of all applications
- Verification must be **repeated** if applications are added or modified
- Slow process with **poor coverage**

Verification is **costly** and effort is expected to **increase** in future!
Formal verification is alternative to simulation
- Provides **analytical bounds** on latency or throughput
- Covers **all combinations** of concurrently running applications

Approach requires **predictable systems**
- Needs performance models of both applications and hardware
- We model applications and hardware as data-flow graphs
- We have proposed a predictable hardware platform
  - Processor tile with MicroBlaze processor
  - Aethereal network-on-chip
  - Memory tiles with SRAM controller or Predator SDRAM controller
SDRAM bandwidth is **scarce** and must be **efficiently** utilized
- Off-chip pins are expensive in terms of area and power

Predator **guarantees bandwidth and latency** to requestors
- Dynamically schedules predictable **memory patterns**

Controller only supports a **limited set** of memory patterns
- Increasingly **inefficient** with faster memories, such as DDR3 SDRAM

The problem in this paper is to **enable efficient formal verification** in systems with DDR2/DDR3 SDRAM.
The **four contributions** of this paper are:

1. Introduces **burst count** as a memory pattern parameter
   - Increases efficiency with faster memories

2. Presents a **classification** of memory patterns into four classes
   - Based on what triggers worst-case latency and bandwidth

3. Derives **bounds on bandwidth and latency**
   - Cover any combination of burst counts and pattern classes
   - Earlier work covered a single case

4. Shows **memory efficiency trends** for DDR2/DDR3 memories
An SDRAM is organized in **banks, rows and columns**
- A row buffer stores a currently active (open) row

Interface has a **command bus, address bus, and a data bus**
- Buses shared between banks to reduce the number of off-chip pins
Memory map decodes address to bank, row, and column

Row is **activated** and copied into the row buffer of the bank

Read bursts and/or write **bursts are issued to the active row**
  - Programmed **burst length** (BL) of 4 or 8 words

Row is **precharged** and stored back into the memory array
Memory Efficiency

Memory efficiency
- The fraction of clock cycles when requested data is transferred
- The exchange rate between peak bandwidth and net bandwidth

Five categories of memory efficiency for SDRAM:
- Refresh efficiency
- Read/write efficiency
- Bank efficiency
- Command efficiency
- Data efficiency

Memory efficiency is the product of these five categories
Timing behavior **hardly changes** between SDRAM generations
- Timings of memory core in nanoseconds are almost the same

Newer memories are clocked at higher frequencies
- Timings of memory core in clock cycles increase

This results in **reducing memory efficiency** for newer memories
- Still takes one clock cycle to transfer two data elements
- Overhead cycles scale with frequency
Introduction

SDRAM overview

Predictable SDRAM controller

Extensions

Experiments

Conclusions
→ Predictability through precomputed **memory access patterns**
  - Patterns are precomputed sub-schedules of SDRAM commands

→ There are **five types** of memory access patterns
  - Read, write, r/w switch, w/r switch, and refresh patterns

→ Pattern to request mapping:
  - Read request → read pattern (potentially first w/r switch)
  - Write request → write pattern (potentially first r/w switch)
  - Refresh pattern issued when required
Patterns enable scheduling at higher level than commands
- Less state and fewer constraints, making them easier to analyze

Read/write patterns issue one burst to each bank in sequence
- Results in high worst-case efficiency
- Requires large requests (64 bytes for 16-bit memory with 4 banks)

Patterns are automatically generated by a tool

Read pattern for DDR2-400
Controller and analysis supports any predictable arbiter
  - Example: Round-Robin, TDM, or CCSP
  - Latency computed in number of interfering requests
  - Latency bound in clock cycles is easily derived since:
    - **Request to pattern mapping is known** (scheduling rules)
    - **Pattern to cycle mapping is known** (length of patterns)

Design provides **bounds** on latency and bandwidth
  - For **any combination** of DDR2/DDR3 memory and supported arbiter
Presentation Outline

Introduction
SDRAM overview
Predictable SDRAM controller

Extensions
Experiments
Conclusions
Faster memories have **tighter timing constraints** in clock cycles
- E.g. first bank not ready when previous burst to last bank finishes

Addressed by issuing **multiple bursts** to each bank
- The number of bursts is a pattern parameter called **burst count**
- **Improves bank efficiency** by amortizing bank conflict overhead
- Requires larger requests, which **may reduce data efficiency**
- Larger requests also **increase memory latency**
Bounding bandwidth and latency requires knowledge about the **worst-case combination of patterns**

Four cases identified based on patterns lengths:
1. Read-dominant pattern sets
2. Write-dominant pattern sets
3. Mix-read-dominant pattern sets
4. Mix-write-dominant pattern sets
Earlier bandwidth and latency analysis is limited to

- Burst count = 1
  - Preventing efficient use of SDRAM with large requests

- Mix-read-dominant pattern sets
  - Most common type, but not always most efficient

Paper presents new general bounds for all combinations of burst counts and pattern types.
Introduction
SDRAM overview
Predictable SDRAM controller
Extensions
Experiments
Conclusions
Experiments consider a range of DDR2/DDR3 memories
- DDR2-400, DDR2-800, DDR3-800, DDR3-1600
- From the slowest DDR2 device to the fastest DDR3 device

All memories have
- a capacity of 512 Mb and a 16-bit interface
- a programmed burst length of 8 words

All DDR2 memories have 4 banks and DDR3 memories 8
This experiment assumes large requests
- Size = banks x burst count x burst length x word size
  - 64 B x burst count for DDR2 and 128 B x burst count for DDR3
  - Data efficiency is 100%

Memory efficiency
- increases monotonically with burst count
- decreases for faster memories, although less for higher burst counts
Net bandwidth increases for faster memories, despite reducing efficiency.

Most patterns mix-read dominant, which is the common case.
- DDR2-800 with BC=2 and BC=4 are write dominant and hence not supported by earlier work.
This experiment studies the impact of using small requests.

Bandwidth does not increase with burst count for small requests.

Fast memories fundamentally require large requests to be efficient (>80%)
- 64B sufficient for DDR2-400
- 256B required by DDR3-1600
There are two additional experiments in the paper:

1. Evaluation of tightness of bound on bandwidth
   - Simulation with worst-case stimuli shows deviation of only 0.2%

2. Show bandwidth/latency trade-off
   - Demonstrates that the new concepts help satisfy a larger set of synthetic use-cases
Presentation Outline

Introduction
SDRAM overview
Predictable SDRAM controller
Extensions
Experiments
Conclusions
This work addresses efficient formal verification of real-time requirements in systems with DDR2/DDR3 SDRAM.

Extends the Predator predictable SDRAM controller design
- Predictable by dynamically scheduling memory patterns
- Supported patterns increasingly inefficient for faster memories

The four contributions of this paper are:
1. A burst count parameter that increases efficiency of patterns
2. A classification of memory patterns into four categories
3. A bandwidth and latency analysis covering all burst counts and pattern classes
4. A demonstration of efficiency trends for DDR2/DDR3 memories