How Not to Configure Your DRAM System

Bruce Jacob

Electrical & Computer Engineering
University of Maryland, College Park
http://www.ece.umd.edu/~blj/

OUTLINE:

- DRAM Primer
- Yesterday’s Results
- Today’s Experiments & Results
- Conclusions
Sources


“DDR2 and Low Latency Variants,” *Memory Wall Workshop*, in conjunction w/ ISCA ’00. B. Davis, T. Mudge, V. Cuppu, and B. Jacob.

Recent experiments by Vinodh Cuppu, Ph.D. student at University of Maryland
Goal

PRELIMINARY DRAM STUDY:

• Bus Transmission
• Row Access
• Column Access
• Data Transfer
• Bus Wait/Synch Time
• Stalls Due to Refresh
• The OVERLAP of These Components (with each other) (with CPU execution)

MODEL EXISTING TECHNOLOGY
DRAM Primer

BUS TRANSMISSION

- CPU
- MEMORY CONTROLLER
- BUS
- Data In/Out Buffers
- Column Decoder
- Sense Amps
- ... Bit Lines...
- Memory Array
- Row Decoder
DRAM Primer

ROW ACCESS
DRAM Primer

COLUMN ACCESS

CPU

MEMORY CONTROLLER

DRAM

Data In/Out Buffers

Column Decoder

Sense Amps

... Bit Lines...

Row Decoder

Memory Array
DRAM Primer

DATA TRANSFER

note: page mode enables overlap with COL
DRAM Primer

BUS TRANSMISSION

note: overlapped component not shown
DRAM Primer

Read Timing for Conventional DRAM

- Data Transfer
- Transfer Overlap
- Column Access
- Row Access

Diagram showing the timing of RAS, CAS, address, column, row, and dataout signals.
DRAM Primer

Read Timing for Fast Page Mode DRAM

- Data Transfer
- Transfer Overlap
- Column Access
- Row Access

RAS  CAS  Address  Column Address  Column Address  Column Address
Row Address  DQ  Valid Dataout  Valid Dataout  Valid Dataout
DRAM Primer

Read Timing for Extended Data Out DRAM
DRAM Primer

Read Timing for Synchronous DRAM

- Clock
- RAS
- CAS
- Address
- Row Address
- Column Address
- DQ
- Valid Dataout
- Valid Dataout
- Valid Dataout

Legend:
- Data Transfer
- Transfer Overlap
- Column Access
- Row Access
DRAM Primer

Read Timing for Rambus DRAM
DRAM Primer

Read Timing for Direct Rambus DRAM

Row [2:0]  Col [4:0]  Data [17:0]

ACT (this)  Col Addr  Col Addr  Col Addr  Col Addr

PRE (next)  Valid Dataout  Valid Dataout  Valid Dataout

ACT (next)

4 cycles

Data Transfer
Transfer Overlap
Column Access
Row Access
Simulator Overview

CPU: SimpleScalar v3.0a
- 8-way out-of-order
- L1 cache: split 64K/64K, lockup free x32
- L2 cache: unified 1MB, lockup free x1
- L2 blocksize: 128 bytes

Main Memory: 8 64Mb DRAMs
- 100MHz/128-bit memory bus
- Optimistic open-page policy
  (close-immediately can be calculated)

Represents a “typical” workstation
How Not to Configure Your DRAM System

Bruce Jacob

University of Maryland

Memory & CPU — PERL

Stalls due to Memory Bandwidth
Stalls due to Memory Latency
Overlap between Execution & Memory
Processor Execution

Newer DRAMs

Yesterday's CPU
Tomorrow's CPU
Today's CPU

Cycles Per Instruction (CPI)

FPM  EDO  SDRAM  ESDRAM  DRDRAM

DRAM Configuration
Conclusions

100MHz/128-bit Bus is Current Bottleneck

- Solution: Fast Bus/es & MC on CPU (e.g. Alpha 21364, Sony Emotion, ...)

Current DRAMs Solving Bandwidth Problem (but not Latency Problem)

- Solution: New cores with on-chip SRAM (e.g. ESDRAM, VCDRAM, ...)
- Solution: New cores with smaller banks (e.g. MoSys “SRAM”, FCRAM, ...)

Bruce Jacob
University of Maryland
Recent Work

Detailed Study of DDR2 Proposals in Concurrent Environment, Including Comparison with DRDRAM

Highly Concurrent System Organizations (Multiple Channels, Queueing Mechanisms, Priority Schemes, Optimal Burst Sizes)
How Not to Configure Your DRAM System

Bruce Jacob
University of Maryland

DDR2 Study Results

Architectural Comparison

Normalized Execution Time (DDR2)

Benchmark
HOW NOT TO CONFIGURE YOUR DRAM SYSTEM

Bruce Jacob
University of Maryland

DDR2 Study Results

Perl Runtime

Execution Time (Sec.)

Processor Frequency

- pc100
- ddr133
- ddr
- ddr2
- ddr2ems
- ddr2vc
DDR2 Study Results

Architectural Comparison for Trace

Normalized Execution Time (DDR2)

Trace

0 0.2 0.4 0.6 0.8 1 1.2 1.4

oltp1w  oltp8w  xm_access  xm_cpu−mark  xm_gcc  xm_quake

ddr2  ddr2ems  ddr2vc
Concurrency Study: Timing

READ REQUEST TIMING:

\[ t_0 \]

ADDRESS BUS

DRAM BANK

DATA BUS

<ROW> <COL> <PRE>

<DB0><DB1><DB2><DB3>
Read/Write Request Shapes

**READ REQUESTS:**

- ADDRESS BUS: 1.25ns
- DRAM BANK: 90ns
- DATA BUS: 61.25ns
- TIME: 10ns

**WRITE REQUESTS:**

- ADDRESS BUS: 1.25ns
- DRAM BANK: 90ns
- DATA BUS: 61.25ns
- TIME: 10ns
Pipelined/Split Transactions

(a) Legal if R/R to different banks:

(b) Legal if turnaround ≤ 8.75ns and R/W to different banks:
(note: write can start up to 7.5ns later if turnaround = 1.25ns)

(c) Back-to-back R/W pair that cannot be nested:
Channels & Banks

1, 2, 4  800 MHz Channels
8, 16, 32, 64  Data Bits per Channel
1, 2, 4, 8  Banks per Channel (Indep.)
32, 64, 128  Bytes per Burst
Burst Scheduling
(Back-to-Back Read Requests)

- Critical-burst-first
- Non-critical bursts are promoted
- Writes have lowest priority
  (tend back up in request queue ...)
- Tension between large & small bursts: amortization vs. faster time to data
The Bottom Line

Cycles per Instruction (CPI)

- Worst Organization
- Average Organization
- Best Organization

Benchmarks: bzip, gcc, mcf, parser, perl, vpr, average
It’s Not Queue Size ...

Black = infinite request queue,
Red = 32-entry request queue
... It’s Also Not Turnaround ... 

Benchmark = BZIP (SPEC 2000), 32-byte burst, 16-bit bus
... It’s Related to Concurrency

Benchmark = BZIP (SPEC 2000), 32-byte burst, 16-bit bus
New Bar-Chart Definition

- **t\textsubscript{PROC}** — CPU with 1-cycle L2 miss
- **t\textsubscript{REAL}** — realistic CPU/DRAM config
- **t\textsubscript{SYS}** — CPU with 1-cycle DRAM latency
- **t\textsubscript{DRAM}** — time seen by DRAM system
System Overhead

System overhead 10-40% over perfect memory

Cycles per Instruction (CPI)

System Bandwidth
(GB/s = 1 Channel * Width * 800Mhz)
8MSHRs, 32-Byte Burst

Benchmark = BZIP (SPEC 2000)
Bandwidth vs. Burst Width

Cycles per Instruction (CPI)

System Bandwidth
(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
Bandwidth vs. Burst Width

System Bandwidth
(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
Bandwidth vs. Burst Width

System Bandwidth
(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
Bandwidth vs. Burst Width

GOLDILOCKS PRINCIPLE

Wide channels (32/64-bit) want large bursts

System Bandwidth
(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
How not to configure your DRAM system

Bandwidth vs. Burst Width

GOLDILOCKS PRINCIPLE

Narrow channels (8-bit) want small bursts

System Bandwidth

(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
Bandwidth vs. Burst Width

GOLDILOCKS PRINCIPLE

Medium channels (16-bit) want medium bursts

System Bandwidth
(GB/s = Channels * Width * 800MHz)

Benchmark = GCC (SPEC 2000), 2 banks/channel
Focus on 3.2 GB/s — MCF

Cycles per Instruction (CPI)

Benchmark = MCF (SPEC 2000)
Focus on 3.2 GB/s — MCF

Benchmark = MCF (SPEC 2000)
Focus on 3.2 GB/s — MCF

Benchmark = MCF (SPEC 2000)
Focus on 3.2 GB/s — MCF

Benchmark = MCF (SPEC 2000)

Multiple channels not always a good idea
Focus on 3.2 GB/s — MCF

Benchmark = MCF (SPEC 2000)
Focus on 3.2 GB/s — BZIP

Benchmark = BZIP (SPEC 2000)
Focus on 3.2 GB/s — BZIP

Benchmark = BZIP (SPEC 2000)
Conclusions

CAREFUL TUNING YIELDS 30–40% GAIN

MORE CONCURRENCY == BETTER

- Via Channels → NOT w/ LARGE BURSTS
- Via Banks → ALWAYS SAFE
- Via Bursts → DOESN’T PAY OFF
- Via MSHRs → NECESSARY

WIDER == BETTER (Thank you, Pontiac)

- Gang Multiple RAMBUS Channels

BURSTS AMORTIZE COST OF PRECHARGE

- Typical Systems: 32 bytes (even DDR2) → THIS IS NOT ENOUGH
CONTACT INFO:

Prof. Bruce Jacob

Electrical & Computer Engineering
University of Maryland, College Park
http://www.ece.umd.edu/~blj/
blj@eng.umd.edu
Dilemma: THIS ...

STATUS QUO in MEMOY-SYSTEM RESEARCH:

... 

if (INSTR.loadstore) {
    if (L1_cache_miss(INSTR.daddr)) {
        if (L2_cache_miss(INSTR.daddr)) {
            cycles += DRAM_LATENCY;
        }
    }
}

...
... or THIS ...

STATUS QUO in MEMORY-SYSTEM RESEARCH:

...  

if ( INSTR.loadstore ) {
    if (L1_cache_miss( INSTR.daddr )) {
        if (L2_cache_miss( INSTR.daddr )) {

            INSTR.ready = now() + DRAM_LATENCY;

        }
    }
}

...
... or THIS

Fast Page Mode Read Cycle
Motivation

HERE’S WHAT YOU MISS:

CPU  bus  MC  DRAM  DRAM  ...  DRAM

CPU  bus  MC  ...  DRAM

DRAM LATENCY:

DATA TRANSFER

OVERLAP

COLUMN ACCESS

ROW ACCESS

BUS TRANSMISSION
Motivation

HERE’S WHAT YOU MISS:

DRAM LATENCY:

DATA TRANSFER?

OVERLAP?

COLUMN ACCESS?

ROW ACCESS?

BUS TRANSMISSION?
Definitions (var. on Burger, et al)

- $t_{\text{PROC}}$ — processor with perfect memory
- $t_{\text{REAL}}$ — realistic configuration
- $t_{\text{BW}}$ — CPU with wide memory paths
- $t_{\text{DRAM}}$ — time seen by DRAM system
DRAM Configurations

FPM, EDO, SDRAM, ESDRAM:

CPU and caches

128-bit 100MHz bus

Memory Controller

Rambus, Direct Rambus, SLDRAM:

CPU and caches

128-bit 100MHz bus

Memory Controller

Note: TRANSFER WIDTH of Direct Rambus Channel

- equals that of ganged FPM, EDO, etc.
- is 2x that of Rambus & SLDRAM
DRAM Configurations

Strawman: Rambus, etc.
Overhead: Memory vs. CPU

Variable: speed of processor & caches
Average Latency of DRAMs

note: SLDRAM & RDRAM 2x data transfers
Average Latency of DRAMs

Bus Wait Time
Refresh Time
Data Transfer Time
Data Transfer Time Overlap
Column Access Time
Row Access Time
Bus Transmission Time

note: SLDRAM & RDRAM 2x data transfers
Cost-Performance

FPM, EDO, SDRAM, ESDRAM:
• Lower Latency => Wide/Fast Bus
• Increase Capacity => Decrease Latency
• Low System Cost

Rambus, Direct Rambus, SLDRAM:
• Lower Latency => Multiple Channels
• Increase Capacity => Increase Capacity
• High System Cost

However, 1 DRDRAM = Multiple SDRAM