

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Computer Architecture
ECE 568

Part 14

Improving Memory Performance: Interleaving

Israel Koren
Fall 2009

ECE568/Koren Part.14 .1

Adapted from Patterson, Katz and Kubiatiowicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

Main Memory Background

- ◆ Performance of Main Memory:
 - **Latency** -> Cache Miss Penalty
 - » **Access Time**: time between request and word arriving
 - » **Cycle Time**: time between requests
 - **Bandwidth** -> I/O & Large Block Miss Penalty (L2)
- ◆ Main Memory is **DRAM**: Dynamic Random Access Memory
 - Dynamic since needs to be refreshed periodically (8 ms, 1% time)
 - Addresses divided into 2 halves (Memory as a 2D matrix):
 - » **RAS** or **Row Access Strobe**
 - » **CAS** or **Column Access Strobe**
- ◆ Cache uses **SRAM**: Static Random Access Memory
 - No refresh (6 transistors/bit vs. 1 transistor)
 - Size**: DRAM/SRAM - 4-8,
 - Cost/Cycle time**: SRAM/DRAM - 8-16

ECE568/Koren Part.14 .2

Adapted from Patterson, Katz and Kubiatiowicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

4 Key DRAM Timing Parameters

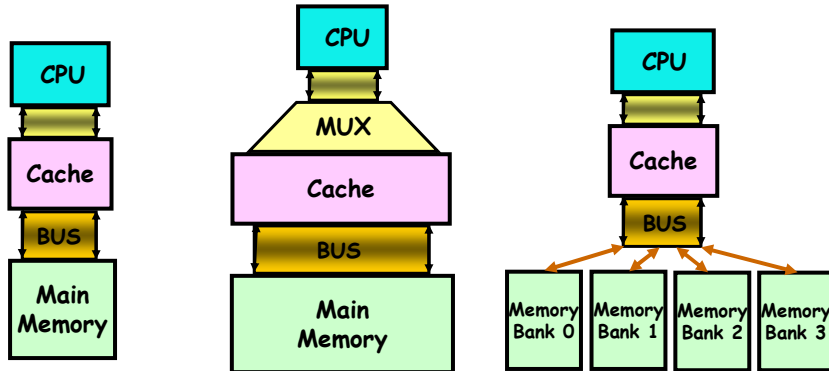
- ◆ t_{RAC} : minimum time from RAS line falling to valid data output (Quoted as the speed of a DRAM)- **access time**
 - A typical 4Mb DRAM $t_{RAC} = 60$ ns
- ◆ t_{RC} : minimum time from the start of one row access to the start of the next - **cycle time**
 - $t_{RC} = 110$ ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- ◆ t_{CAC} : minimum time from CAS line falling to valid data output
 - 15 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- ◆ t_{PC} : minimum time from the start of one column access to the start of the next
 - 35 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns

DRAM Performance

- ◆ A 60 ns (t_{RAC}) DRAM can
 - perform a row access only every 110 ns (t_{RC})
 - perform column access (t_{CAC}) in 15 ns, but time between column accesses is at least 35 ns (t_{PC})
 - » In practice, external address delays and turning around buses make it 40 to 50 ns
- ◆ Fast DRAMs
 - Multiple CAS accesses (same memory row) - **page mode**
 - **RAMBUS**: modified DRAM interface
 - » Each Chip a module vs. slice of memory
 - » Short bus between CPU and chips
 - » Variable amount of data returned
 - **Synchronous DRAM**: 2 banks on chip, transfer synchronous to system clock (66 - 150 MHz)
 - Both have higher bandwidth but also higher latency

Faster Memory Through Organization

- ◆ **Simple:** CPU, Cache, Bus, Memory same width (32 or 64 bits)
- ◆ **Wide:** CPU-to-Cache/Mux 1 word; Bus, Memory N words (Alpha: 64 bits & 256 bits; UltraSPARC: 512)
- ◆ **Interleaved:** CPU, Cache, Bus 1 word; Memory N Modules (4 Modules)



ECE568/Koren Part.14.5

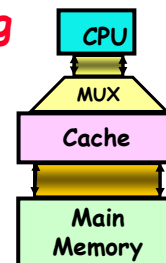
Adapted from Patterson, Katz and Kubiatiowicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

Wide memory vs. Interleaving

◆ Disadvantages of wide memory

- Wide system bus
- MUX between CPU & cache on processor's critical path
- Not useful for multiple units independently accessing memory



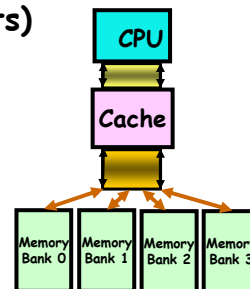
◆ Simple timing model (word size is 32 bits)

- 1 CPU cycle to send address
- 6 CPU cycles access time, 1 cycle to send data
- Cache Block is 4 words

◆ **Simple M** = $4 \times (1+6+1) = 32$

◆ **Wide M** = $1 + 6 + 1 = 8$

◆ **Interleaved M** = $(1+6+1) + 3 \times 1 = 11$

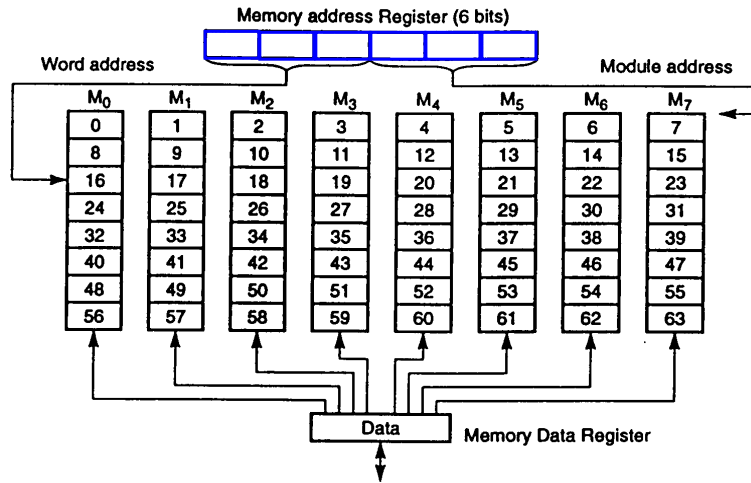


ECE568/Koren Part.14.6

Adapted from Patterson, Katz and Kubiatiowicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

8-way Memory Interleaving



ECE568/Koren Part.14.7

K. Hwang, "Advanced Computer Architecture," McGraw-Hill, 1993.

Independent Memory Banks

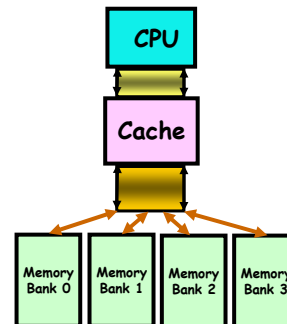
◆ How many banks?

Ideally: number banks \geq number clocks to access word in bank

- For sequential accesses, otherwise will return to original bank before it has next word ready

◆ Increasing DRAM capacity

- => fewer chips
- => harder to have multiple banks



ECE568/Koren Part.14.8

Adapted from Patterson, Katz and Kubiatiowicz © UCB

Copyright 2005 UCB & Morgan Kaufmann

Bandwidth

- ◆ $BW_{bank} = 1 / cycle_time$
- ◆ BW of m banks = $m \times BW_{bank}$ (Ideal case)
- ◆ If random memory requests (from different units):
 $BW \approx m^{.56} \times BW_{bank} \approx \sqrt{m} \times BW_{bank}$
- ◆ Example of bandwidth analysis (Burnett & Coffman, Hellerman)
 - CPU maintains a request queue A_1, A_2, \dots, A_q
 - Before each cycle a **request sequence** A_1, A_2, \dots, A_k ($k \leq m$ & $k \leq q$) is selected so that no two requests are to same bank
 - The closer k is to m , the better
 - $p(k)$ = probability density function of request sequence length
 - $BW = E\{k\} = \sum_{k=1}^m k p(k)$
 - Assume all requests are instruction addresses with λ = Prob. of a branch
 - $p(1) = \lambda$; $p(k) = (1 - \lambda)^{k-1} \lambda$ for $1 < k < m$; $p(m) = (1 - \lambda)^{m-1}$

ECE568/Koren Part.14.9

J. Hayes, "Computer Architecture & Organization," McGraw-Hill, 1998.

Bandwidth as a function of λ

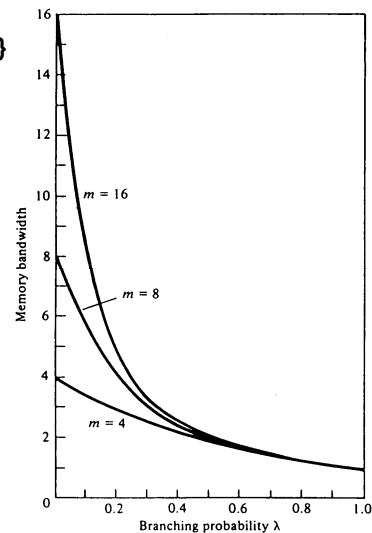
$$BW = \lambda + 2(1-\lambda)\lambda + 3(1-\lambda)^2 \lambda + \dots + m(1-\lambda)^{m-1} \lambda$$

$$BW = [1 - (1-\lambda)^m] / \lambda$$

$$BW = m \quad \text{when } \lambda = 1$$

$$\lim_{\lambda \rightarrow 0} BW = m$$

$$\lambda \rightarrow 0$$



J. Hayes, "Computer Architecture & Organization," McGraw-Hill, 1998.

ECE568/Koren Part.14.10

Avoiding Bank Conflicts

- ◆ Can occur even with many banks

```
int x[256][512];
  for (j = 0; j < 512; j = j+1)
    for (i = 0; i < 256; i = i+1)
      x[i][j] = 2 * x[i][j];
```

- ◆ Even with 128 banks, since 512 is multiple of 128, conflict on word accesses
- ◆ SW: loop interchange or declaring array not power of 2 ("array padding")
- ◆ HW: Prime number of banks
 - bank number = address mod number of banks
 - address within bank = address / number of words in bank
 - bank number? easy if 2^N words per bank

Main Memory Summary

- ◆ Wider Memory
- ◆ Interleaved Memory: for sequential or independent accesses
 - Multiprocessor
 - I/O
 - CPU with Non-blocking Cache
- ◆ Reducing bank conflicts: SW & HW
- ◆ DRAM specific optimizations: page mode & Specialty DRAM (e.g., Video RAM)