

UNIVERSITY OF MASSACHUSETTS  
Dept. of Electrical & Computer Engineering

Computer Architecture  
ECE 568

Part 14

Improving Memory Performance: Interleaving

Israel Koren  
Fall 2011

ECE568/Koren Part.14 .1

Adapted from UCB and other sources

Copyright UCB & Morgan Kaufmann

## Main Memory Background

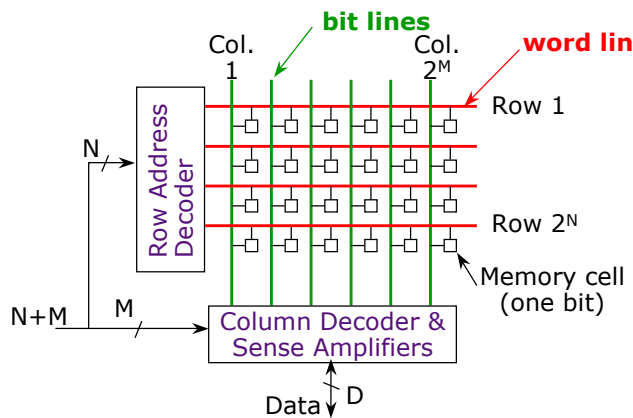
- ◆ Performance of Main Memory:
  - **Latency** -> Cache Miss Penalty
    - » **Access Time**: time between request and word arriving
    - » **Cycle Time**: time between requests > access time
  - **Bandwidth** -> I/O & Large Block Miss Penalty (L2)
- ◆ Cache uses **SRAM**: Static Random Access Memory
  - 6 transistors/bit vs. 1 transistor for DRAM cell
- ◆ 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**

ECE568/Koren Part.14 .2

Adapted from UCB and other sources

Copyright UCB & Morgan Kaufmann

## DRAM Architecture



- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel
- 4-8 logical banks/chip; each physically implemented as several smaller arrays

ECE568/Koren Part.14.3

Adapted from UCB and other sources

Copyright UCB & Morgan Kaufmann

## DRAM Operation

Two steps in read/write access to a given bank

- ◆ Row access (RAS)
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
- ◆ Column access (CAS)
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change latches which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (**burst mode**)

Each step has a latency of around 15-20ns in modern DRAMs  
 Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture

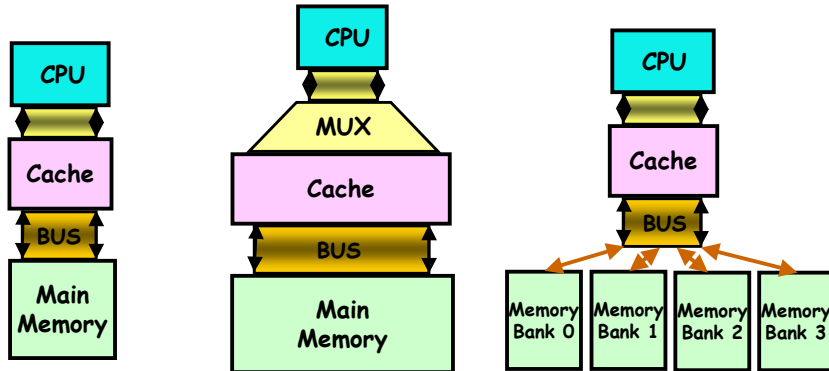
ECE568/Koren Part.14.4

Adapted from UCB and other sources

Copyright UCB & Morgan Kaufmann

## 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 UCB and other sources

Copyright 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
  - » Multiprocessor/multi-core
  - » I/O
  - » CPU with Non-blocking Cache

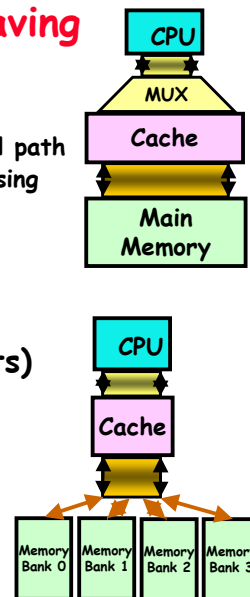
### ◆ Simple timing model (word size is 32 bits)

- 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** =  $(1+6+1) + 3 \times 1 = 11$

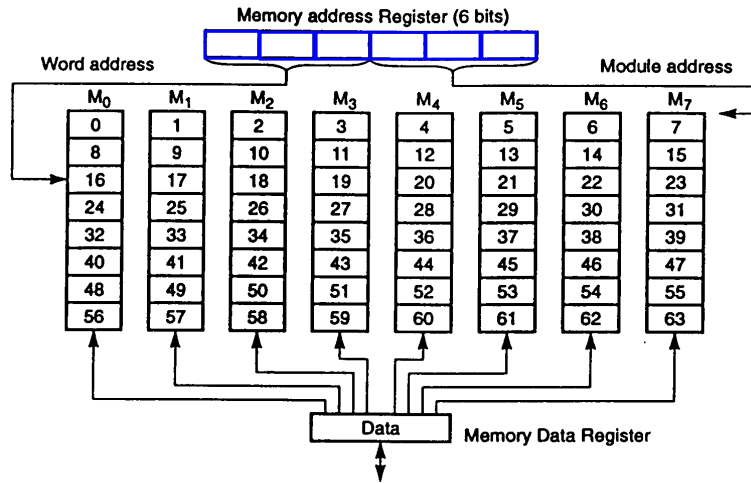


ECE568/Koren Part.14 .6

Adapted from UCB and other sources

Copyright UCB & Morgan Kaufmann

## 8-way Memory Interleaving



ECE568/Koren Part.14.7

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

## Independent Memory Banks

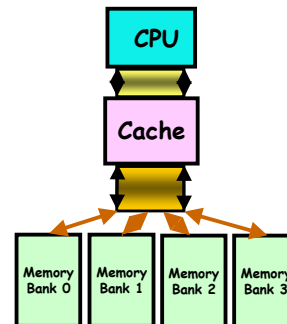
### ◆ How many banks?

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 UCB and other sources

Copyright UCB & Morgan Kaufmann

## Bandwidth

- ◆  $BW_{bank} = 1 / cycle\_time$
- ◆  $BW \text{ of } m \text{ banks} = m \times BW_{bank}$  (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  
 $\lambda$  = 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 $\lambda$

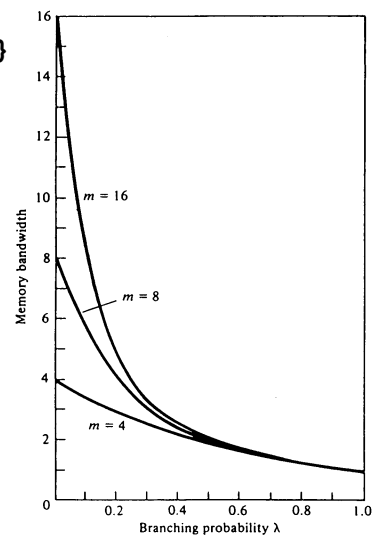
$$BW = \lambda + 2(1-\lambda)\lambda + 3(1-\lambda)^2 \lambda + \dots + m(1-\lambda)^{m-1} \lambda$$

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

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

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

$$\lambda \rightarrow 0$$



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
  - Difficult to implement