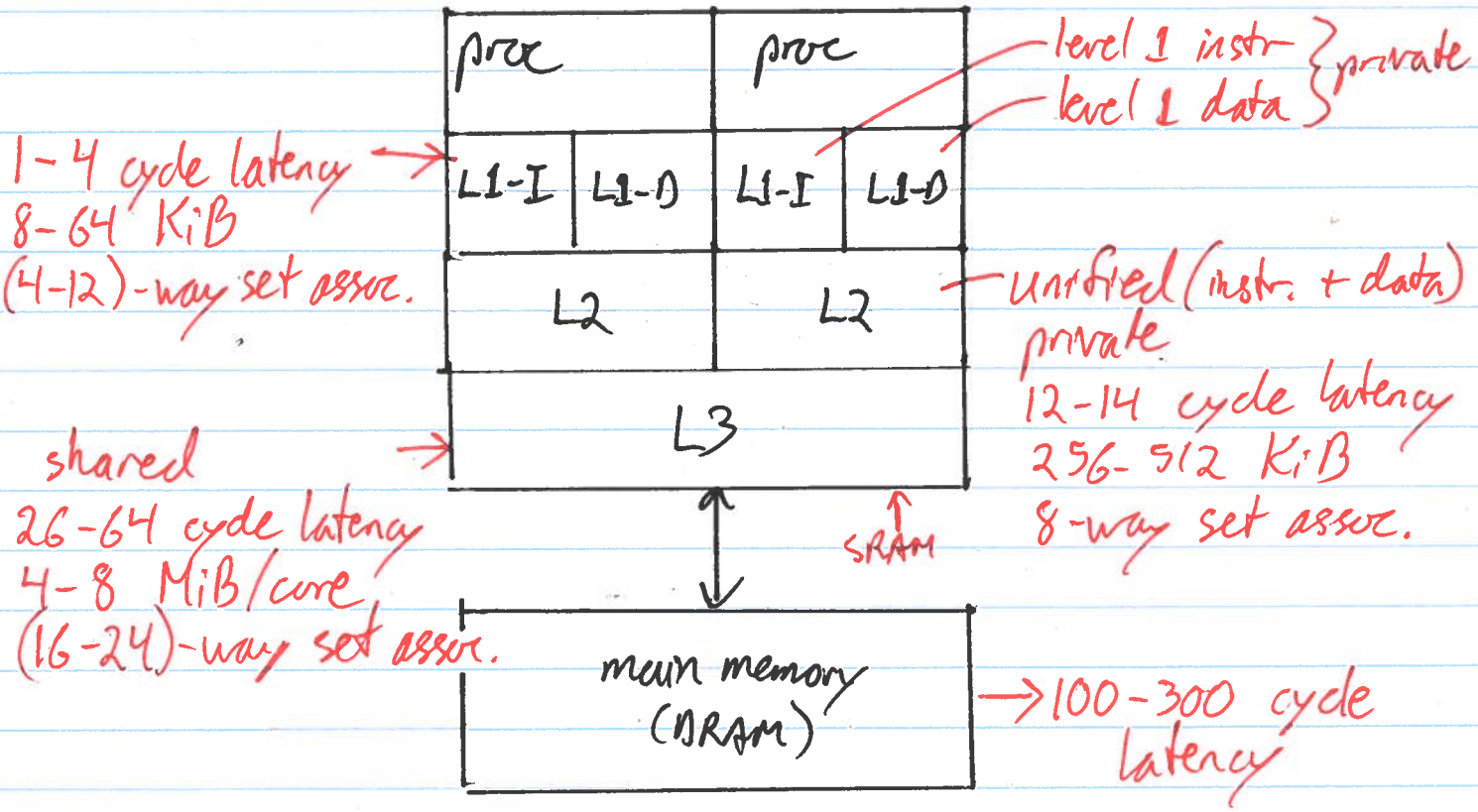
Memory Organization

Homogeneous Multicores



$t_{avg} = t_{hit} + \%_{miss} \times t_{miss}$

L1: $t_{miss} = t_{avg}(L2)$ ← typically small ⇒ optimize L1 for hit time

L2: $t_{miss} = t_{avg}(L3)$

L3: $t_{miss} = t_{DRAM}$ ← large ⇒ optimize L3 for low $\%_{miss}$

(doubling cache size typically reduces $\%_{miss}$ by $\sqrt{2}$)

Inclusivity Options

1. inclusive — $L_i \subset L_{i+1}$
2. exclusive — $L_i \cap L_{i+1} = \emptyset$
3. non-inclusive — $L_i \cap L_{i+1} \neq \emptyset$

Inclusive Caches

- make cache coherency easier
  - if the outer level (e.g. L3) is inclusive of inner caches (e.g. L1 + L2) then only outer cache need participate in the coherency protocol (such MESI)
  - the L3 would still communicate invalidates to the inner caches
- inclusive caches reduce total storage due to duplication

Intel: L2 non-inclusive, L3 inclusive
AMD: L2 inclusive, L3 non-inclusive

Lockup-Free

- when a request misses, it is added to a Miss Status Holding Register (MSHR) and the request is forwarded to the next level
- the # MSHR entries determines max outstanding requests
- lockup-free behaviour is important for out-of-order processors and for shared caches

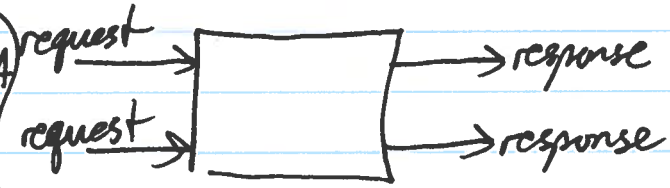
Increasing Cache Bandwidth

(1) pipelined operations: TLB check, tag read, tag compare, data read
- latency may be 4+ cycles but throughput is 1 request/cycle

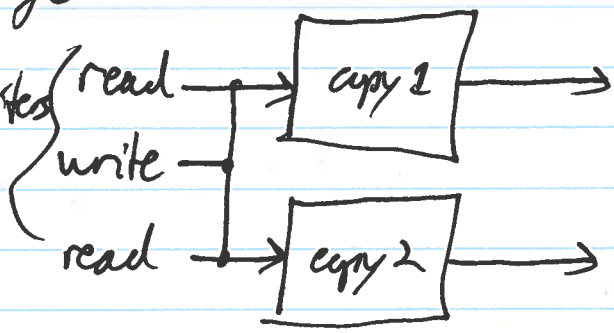
(2) clock cache faster than core - only for slow cores

(3) for instructions load multiple bytes/instr. per access
e.g. x86 fetches 16-32 B of instr. per access (instructions are 1-15 B)

(4) multiported cache
- dual connections to each bit cell
- cells increases ~~for~~ from 6T/cell to 8T/cell
- latency also increases
- latency is okay for an FPGA implementation since the clock rate is lower than in ASICs

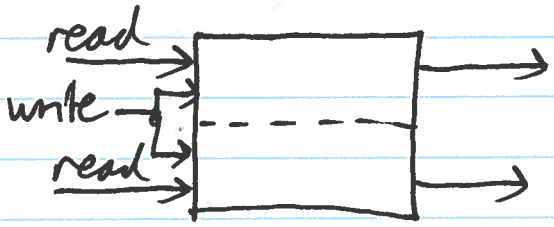


(5) duplicated cache storage
- multiple readers or single writes
- increases size but not latency
- only good for small caches

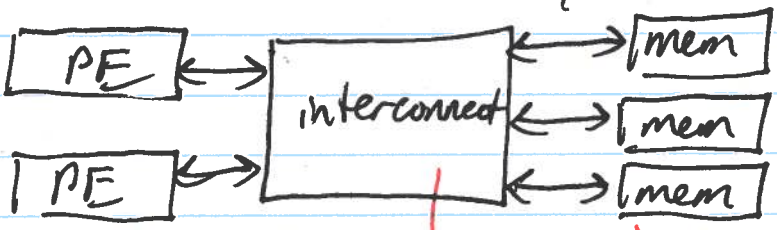


(6) multibanked cache

- two reads per cycle or one write per cycle
- conflicts are serialized



Multibanked Cache/Memory



crossbar or NoC (network on chip)

Ports vs Banks

- consider C cores and N banks or ports
- assume round-robin access to memories and heavy traffic

mem access from core 4

latency = C

- ~~but~~ consecutive accesses to same bank

- multiported (multiple accesses to same memory)

$N \geq C$, latency = 1

$N < C$, latency = C/N

- multibanked
  - also assume requests are uniformly and independently distributed*
  - probability of a conflict for a pair of requests = $1/N$
  - avg latency = $1 + (C-1)/N$

↑ for 1 core's request ↖ conflicts with other C-1 cores' requests

$C/N < 1 + \frac{C-1}{N}$

multiported — multibanked

better performance from multiporting except that it adds latency to every requests

DRAM

- main memory - dynamic RAM - large and slow
- has been multi-banked since 1st gen. DDR
- Handouts > dram handout
  256 Mib : 8192 rows × 8192 b/row (1024 8-bit col/row)
- Operation: (within a bank)
  Activate: opens a row
  Read/write: columns in the open row
  Precharge: closes the open row and pre-charges the bit lines to Vdd/2 to prepare for next Activate
- DDR3: precharge + activate takes 30-40 cycles, column read/writes take ~4 cycles
- performance/throughput is affected by the # of open/close operations

DRAM Configurations

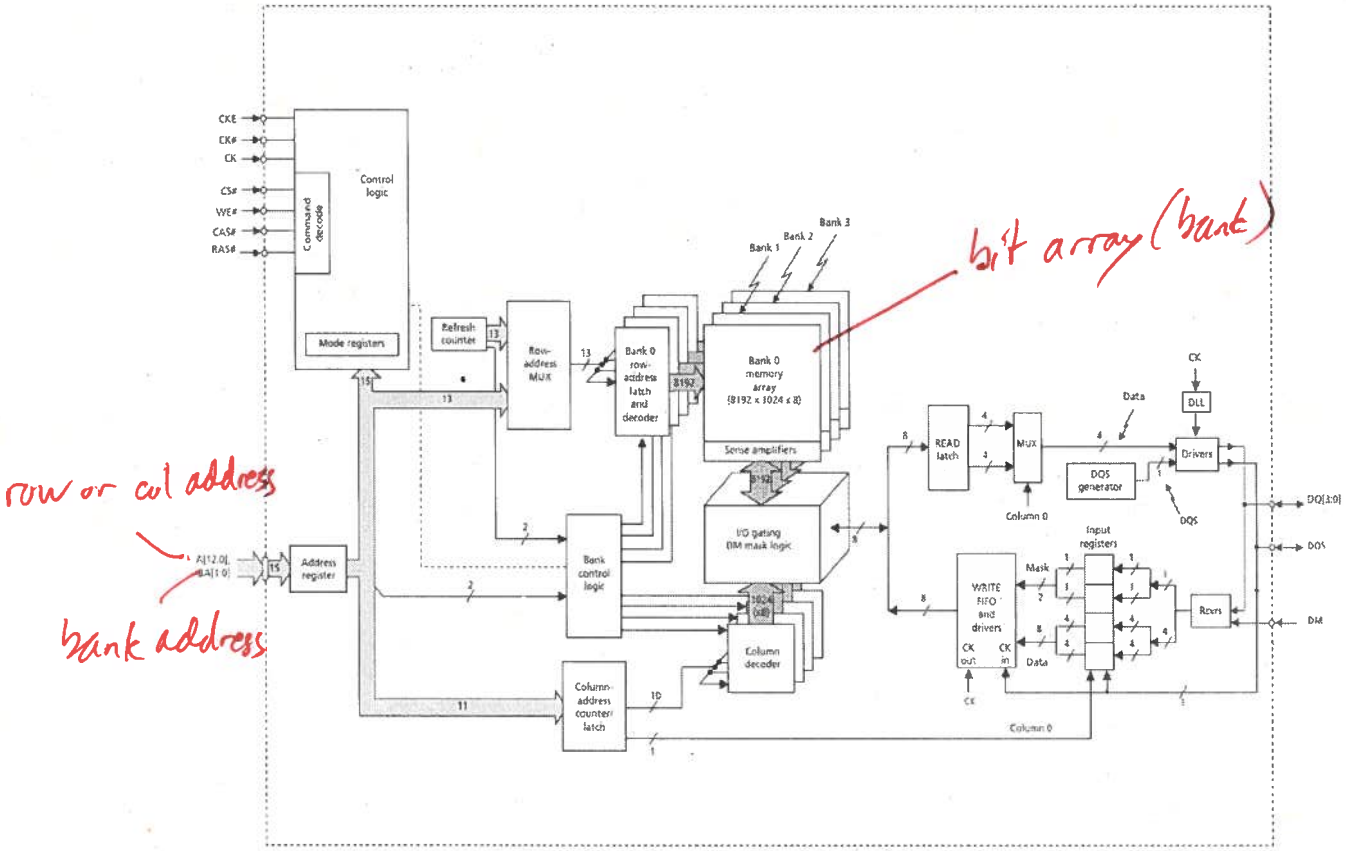
① Interleaved: each PE spreads its data across all banks

row	Bank 1	Bank 2	Bank 3	Bank 4
0	0-1fff	2000-3ffff	4000-5ffff	6000-7fff
1	8000-9ffff			
2				
...				

(assume 8 KiB/row)

- see Bank Configuration handout

Micron 256Mb DDR Datasheet



Bank Configuration

Interleaved

	Bank 1	Bank 2	Bank 3	Bank 4
PE #1 data				
PE #2 data				

Partitioned

Bank 1	Bank 2	Bank 3	Bank 4

PE #1 PE #2

- interleaved: PEs accessing different rows in the same banks will cause excessive open/close operations

(2) Partitioned

- see handout

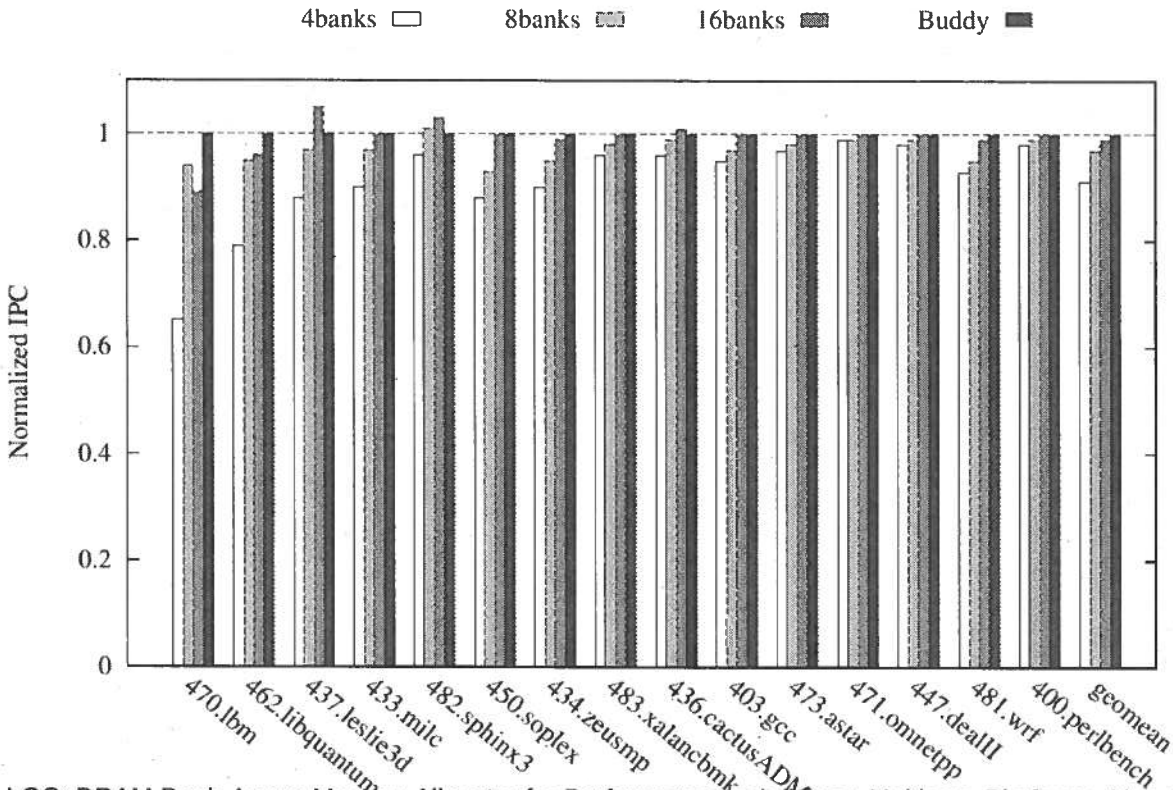
- each PE has fewer open rows (bad option for a single ~~core~~ PE) - limits parallelism ~~available~~ available to a single PE but reduces conflicts between PEs

Study: Impact of Bank Partitioning
Handouts > dram handout - second page
- study performed with SPEC2006 benchmark programs
- total banks = 16
- Buddy = Linux VM memory allocator
- IPC = instructions per cycle (performance)

- Solo Application
  - performance impact of partitioning was mostly small (~~normalized~~ speedup ~ 0.6 to 1)
    - depends on the clustering/spread of accesses

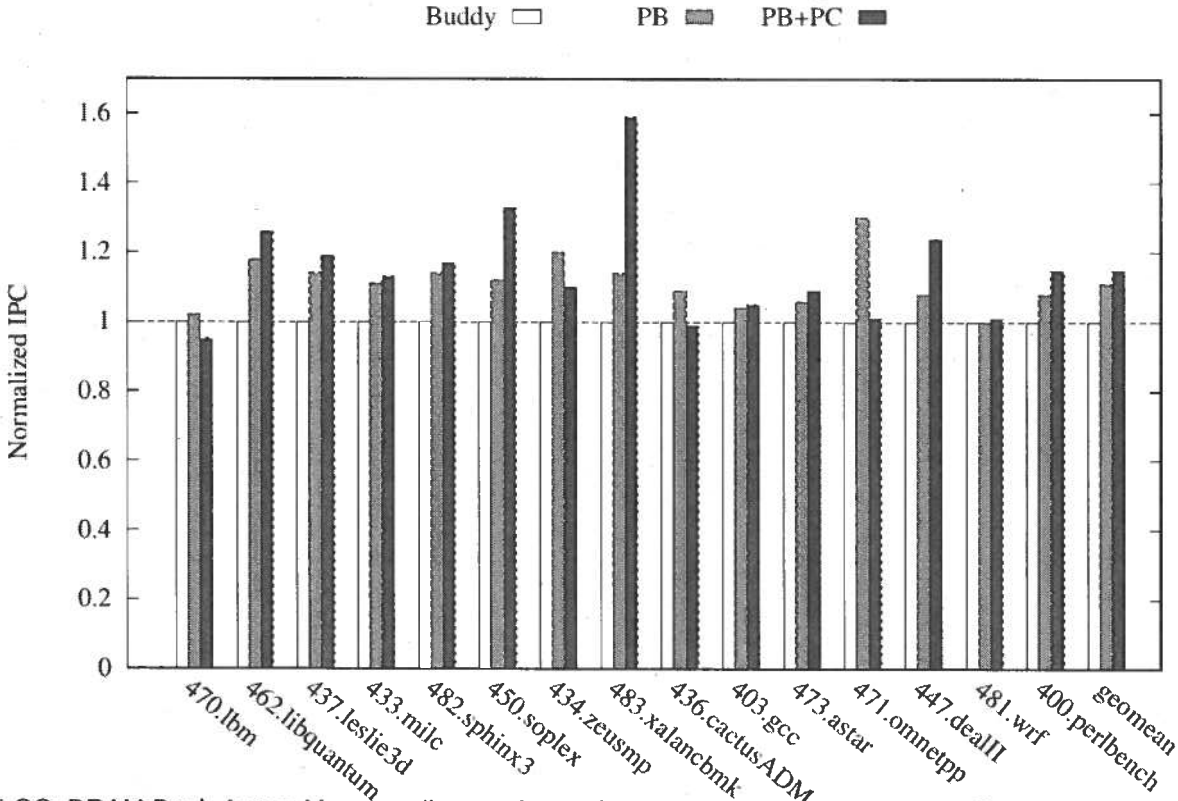
- Co-run Applications
  - PB + PC = partitioned mem. banks + partitioned cache
  - greater impact on performance (speedup ~ 1 - 1.6)

Impact of Bank Partitioning on a Solo Application on 1 Core of a 4-Core Xeon (other cores idle)



PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms, Yun et al, 2014.

Impact of Bank Partitioning on Co-run Applications with other 3 cores running 470.lbm



PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms, Yun et al,

Scratchpad Memory

- an alternative to cache memory
- explicitly managed local (on-chip) SRAM memory
  - mapped to a region of the address space
  - PEs DMA data to/from scratchpad memories
  - run a task using 1 scratchpad while DMAing data to prepare other scratchpad for next task

- pros:
  - predictable access times
  - energy efficient - no tag memory

- cons:
  - harder to use - exposed to the programmer
  - moving linked data structures to/from scratchpad is more work