Coding for memory systems

Ankit S. Rawat, O. Ozan Koyluoglu, and Sriram Vishwanath

WNCG, UT Austin

May 9, 2013

Coding for memory systems

- Idea: Add redundancy to the data stored in memory bank in order to reduce latency, and increase memory access speed.
- Basic mechanism: Erasure codes allow to add redundancy such that when a memory bank is serving to a processor, another processor can be served simultaneously by re-constructing the data of the busy memory bank (i.e., considering the busy bank as erasure).

Simplest example

Scenario: A processor is accessing to one of the two memory banks. We want to double the capacity, and add another processor. But, this may create bank conflicts. Algorithmic Memory (by Memoir Systems) uses coding to resolve conflicts.
Note: Algorithmic Memory (by Memoir Systems) is using erasure codes (like Reed-Solomon codes). Different codes exist allowing efficiency for different tradeoffs.

Regenerating codes

Regenerating codes allow for trading-off a) the complexity of the circuitry (or bandwidth) used for the recovery of busy bank and b) data bus bandwidth connecting banks to processors.

- Encoding k memory banks, d_1 to d_k, into n banks by adding $n-k$ parity banks p_1 to p_{n-k}

Regenerating codes

- α: Data bus bandwidth connecting banks to each processor
- d: Number of banks connected by recovery circuitry in the event that a busy memory bank needs to be connected to a processor
- β: Amount of data downloaded to the recovery circuitry

Code design for the worst case

- Consider all the processors are interested in the first memory bank, d_1.
- Serve $\mathcal{P}_1$ from bank storing d_1. Use recovery circuitry for the rest of the processors. Total nodes need to satisfy $n \geq 1 + (p-1)d$. As we seek to maximize *processors served per memory bank* p/n, we set $n = 1 + (p-1)d$ for this analysis.
- Given $n = 1 + (p-1)d$, the code trades-off the followings
  - Maximize *capacity of the system* k/n.
  - *Total recovery bandwidth per processor:* $\max\{\alpha, d\beta\}$.
  - *Data bus bandwidth from recovery unit to each processor:* α.

Extensions

- Allow for delay instead of worst case code design.
- Memory write operations, and update efficiency.

Codes for 2 Reads

- We consider the following scenario:

Codes for 2 Reads

- We consider the following scenario:
  - We have to store two blocks of data $(\mathbf{a}, \mathbf{b})$.

Codes for 2 Reads

- We consider the following scenario:
  - We have to store two blocks of data $(\mathbf{a}, \mathbf{b})$.
  - $\mathrm{size}(\mathbf{a}) = \mathrm{size}(\mathbf{b}) =$ size of one memory bank $= M$.

Codes for 2 Reads

- We consider the following scenario:
  - We have to store two blocks of data $(\mathbf{a}, \mathbf{b})$.
  - $\mathrm{size}(\mathbf{a}) = \mathrm{size}(\mathbf{b}) =$ size of one memory bank $= M$.
  - Let $(a(1), \ldots, a(M))$ denote M elements of block (a).

Codes for 2 Reads

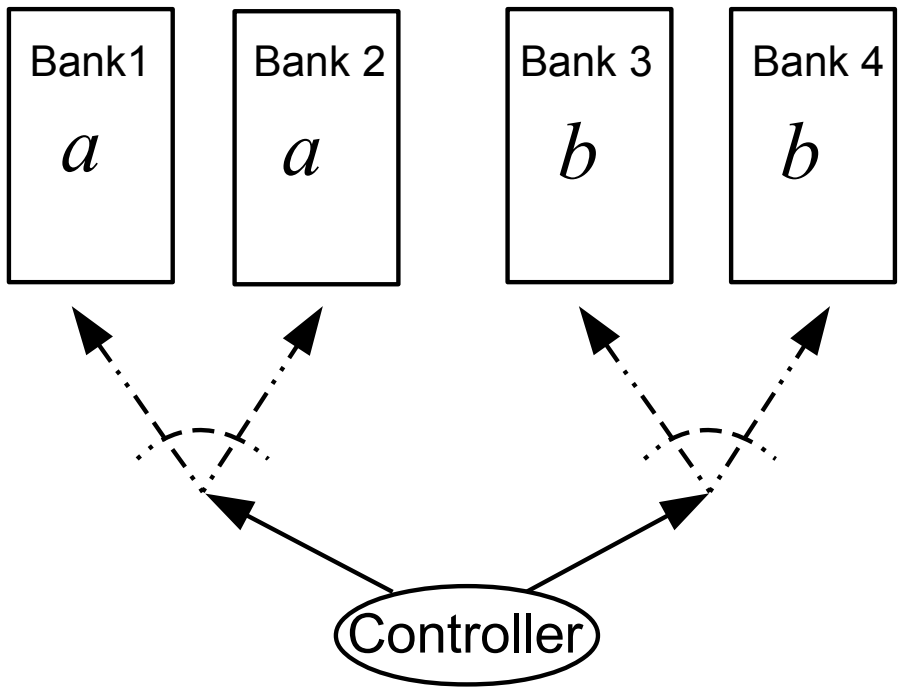
- We consider the following scenario:
  - We have to store two blocks of data $(\mathbf{a}, \mathbf{b})$.
  - $\text{size}(\mathbf{a}) = \text{size}(\mathbf{b}) =$ size of one memory bank $= M$.
  - Let $(a(1), \ldots, a(M))$ denote M elements of block (a).
  - Similarly, $\mathbf{b} = (b(1), \ldots, b(M))$.

Codes for 2 Reads

- We consider the following scenario:
  - We have to store two blocks of data $(\mathbf{a}, \mathbf{b})$.
  - $\text{size}(\mathbf{a}) = \text{size}(\mathbf{b}) =$ size of one memory bank $= M$.
  - Let $(a(1), \ldots, a(M))$ denote M elements of block (a).
  - Similarly, $\mathbf{b} = (b(1), \ldots, b(M))$.
  - Need to support any 2 reads: $(a(i), a(j)), (b(i), b(j))$, or $(a(i), b(j))$.

Codes for 2 Reads

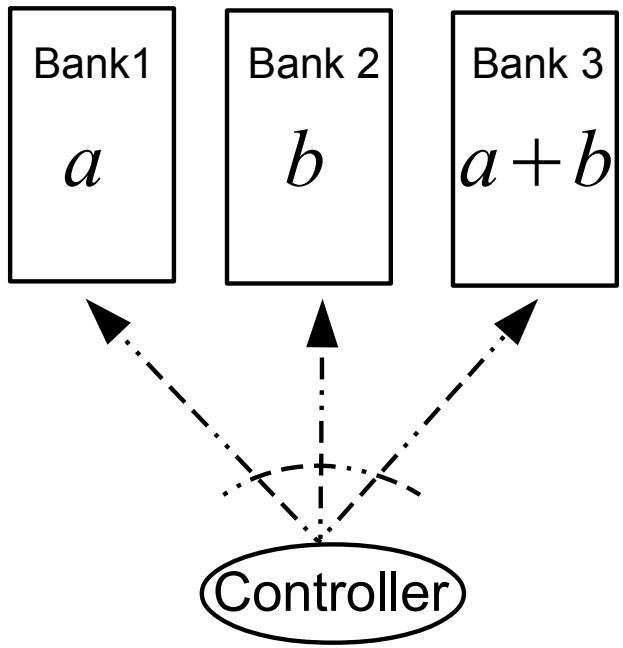
- Solution 1: 2-Repetition



- 4 memory banks to store 2 data blocks $\Rightarrow$ Rate 1/2.
- Total area 4 unit.

Codes for 2 Reads

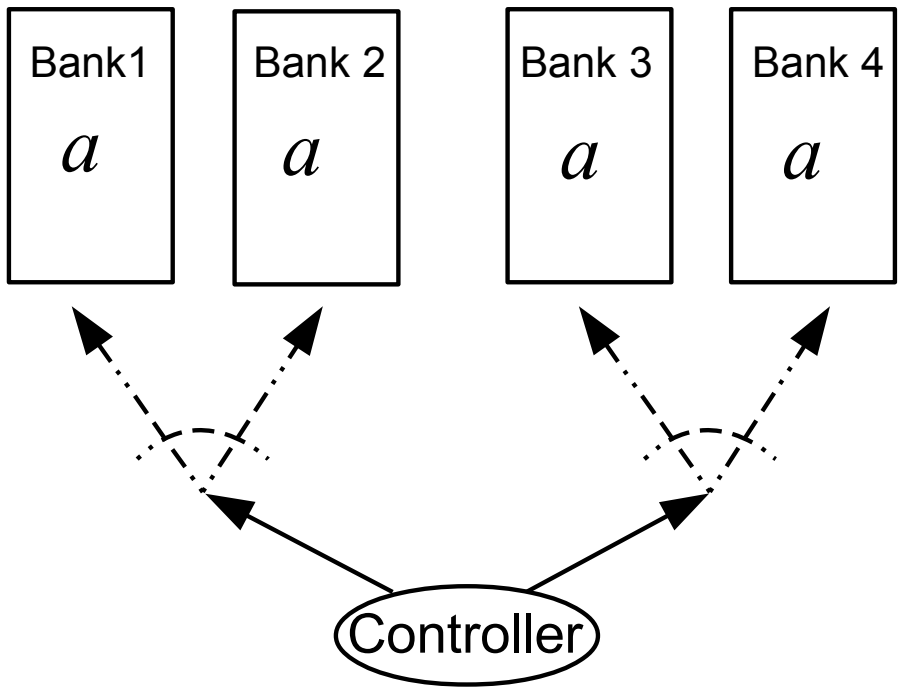
- Solution 2: XOR code



- $a + b = (a(1) + b(1), \ldots, a(M) + b(M))$.
- 3 memory banks to store 2 data blocks $\Rightarrow$ Rate 2/3.
- Total area 3 unit + controller circuitry (includes decoder).

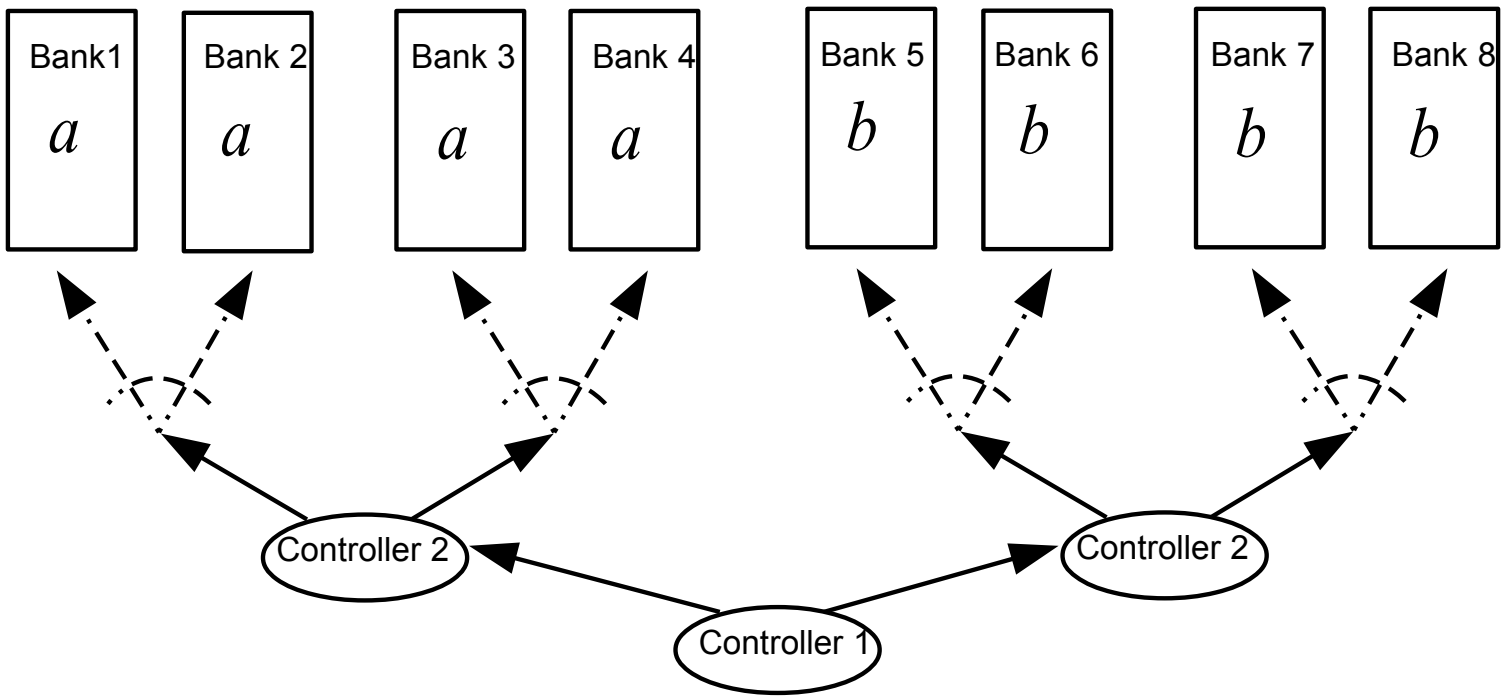
Codes for 2 Reads and 1 Write

- Solution 1: 4-Repetition



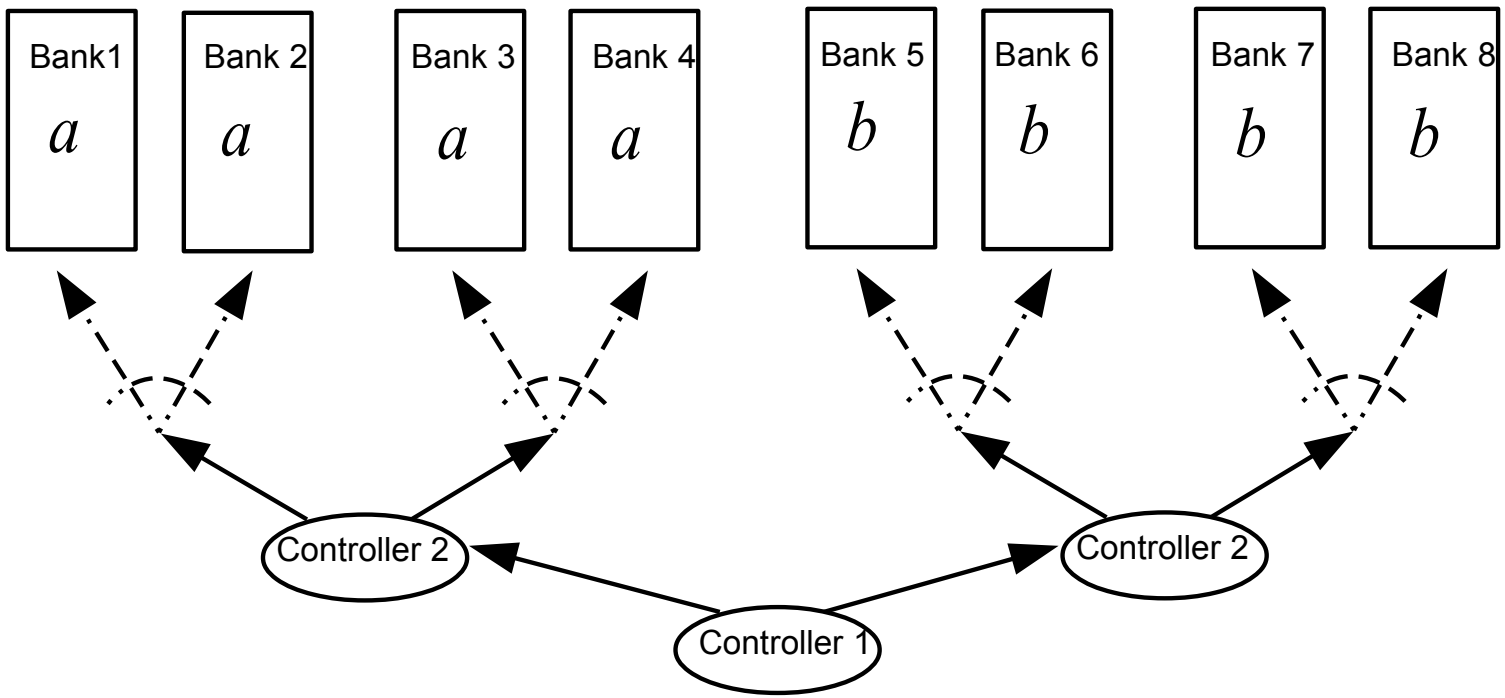
Codes for 2 Reads and 1 Write

- Solution 1: 4-Repetition



Codes for 2 Reads and 1 Write

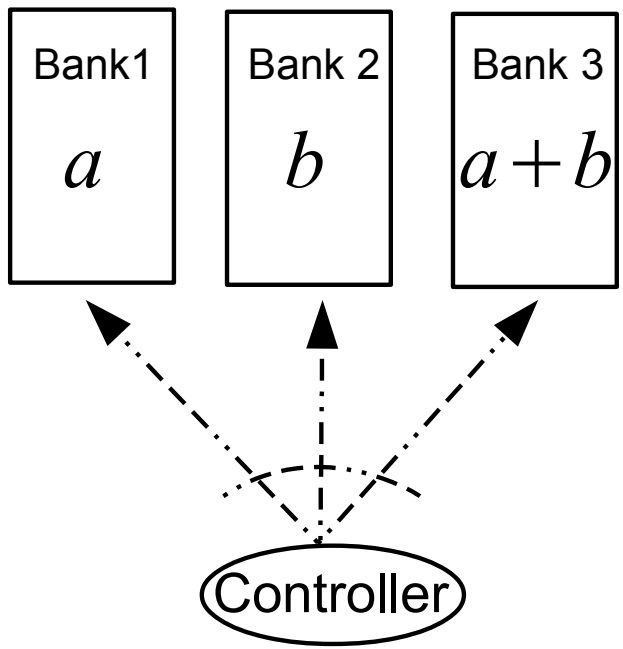
- Solution 1: 4-Repetition



- 8 memory banks to store 2 data blocks $\Rightarrow$ Rate 1/4.
- Total area 8 unit + controller circuitry.

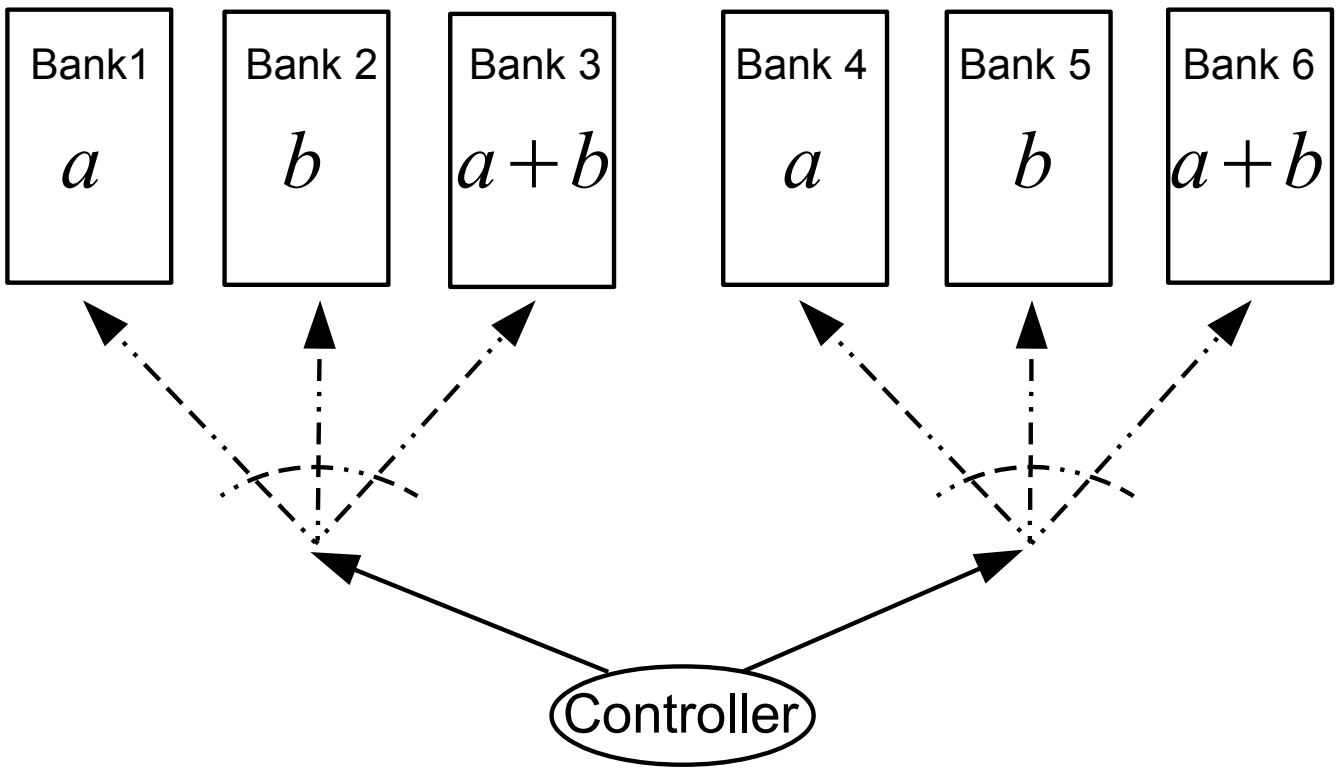
Codes for 2 Reads and 1 Write

- Solution 2: 2-Repetition of XOR code



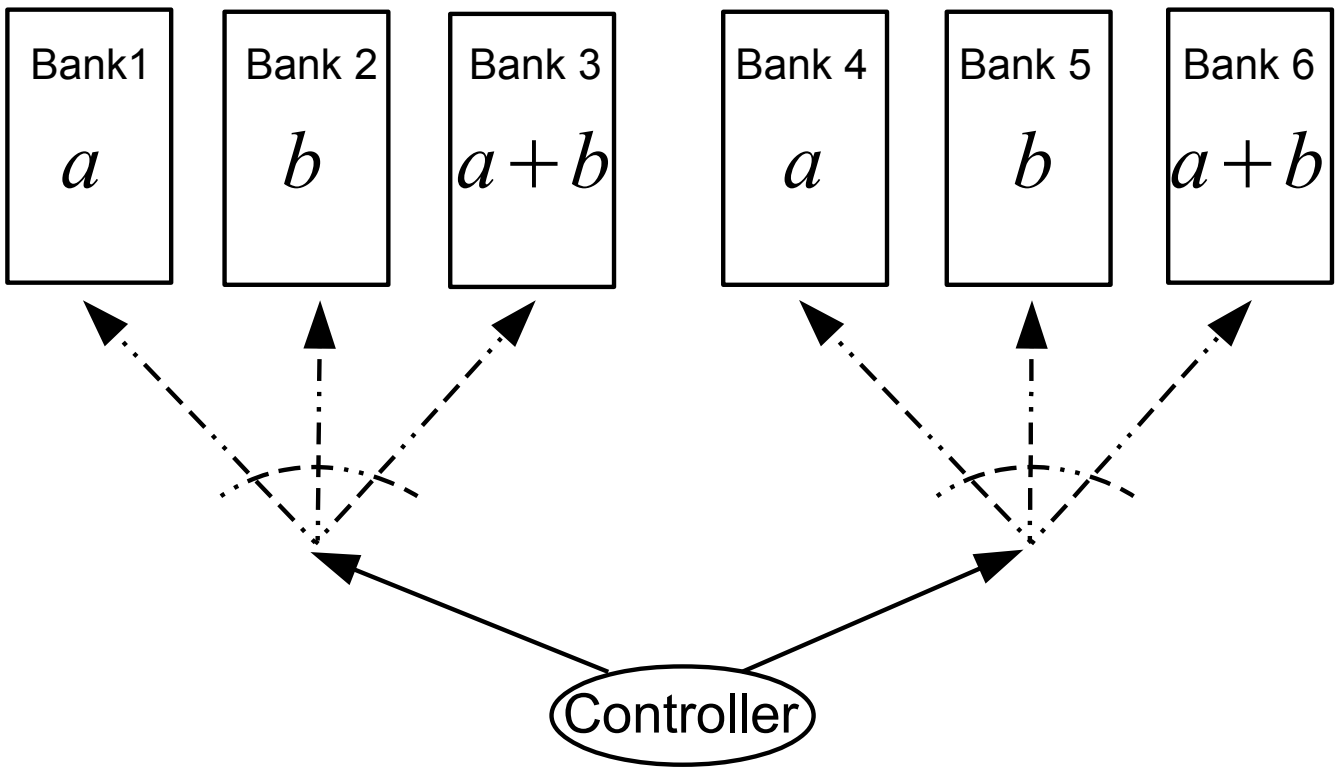
Codes for 2 Reads and 1 Write

- Solution 2: 2-Repetition of XOR code



Codes for 2 Reads and 1 Write

- Solution 2: 2-Repetition of XOR code



- 6 memory banks to store 2 data blocks $\Rightarrow$ Rate 1/3.
- Total area 6 unit + controller circuitry (includes decoder).

Analysis of Traces from Huawei (I)

- Total length: **999471** cycles (processor clock).
- Fraction of time memory is accessed (read/write): **0.3045**.
- Fraction of memory access for read: **0.3874**.
- Fraction of memory access for write: **0.6932**.
- At most **two** fetch instructions in one clock cycles $\Rightarrow$ At most **two** memory access in one clock cycle.
- Fraction of cycles when memory is accessed and two memory operations are performed: **0.0746**.

Analysis of Traces from Huawei (II)

At most $\underline{2}$ memory access in one clock cycle.

Analysis of Traces from Huawei (II)

At most 2 memory access in one clock cycle.

- Possible memory operations in one clock cycle:
  1. 1 read.
  2. 1 write.
  3. 1 read 1 write.
  4. 2 reads.
  5. 2 writes.

Analysis of Traces from Huawei (II)

At most 2 memory access in one clock cycle.

- Possible memory operations in one clock cycle:
  1. 1 read.
  2. 1 write.
  3. 1 read 1 write.
  4. 2 reads.
  5. 2 writes.

Both solutions for **2 reads and 1 write** work.
('4-repetition' and '2-repetition of XOR code'.)

Analysis of Traces from Huawei (III)

Fraction of time memory is accessed (read/write): 0.3045

Analysis of Traces from Huawei (III)

Fraction of time memory is accessed (read/write): 0.3045

Fraction of cycles when memory is accessed and two memory operations are performed: 0.0746.

Analysis of Traces from Huawei (III)

Fraction of time memory is accessed (read/write): 0.3045

Fraction of cycles when memory is accessed and two memory operations are performed: 0.0746.

- Most of the instructions do not access memory.
- Bursty access pattern though.
- Very few clock cycles have two simultaneous access to memory.

Analysis of Traces from Huawei (III)

Fraction of time memory is accessed (read/write): 0.3045

Fraction of cycles when memory is accessed and two memory operations are performed: 0.0746.

- Most of the instructions do not access memory.
- Bursty access pattern though.
- Very few clock cycles have two simultaneous access to memory.

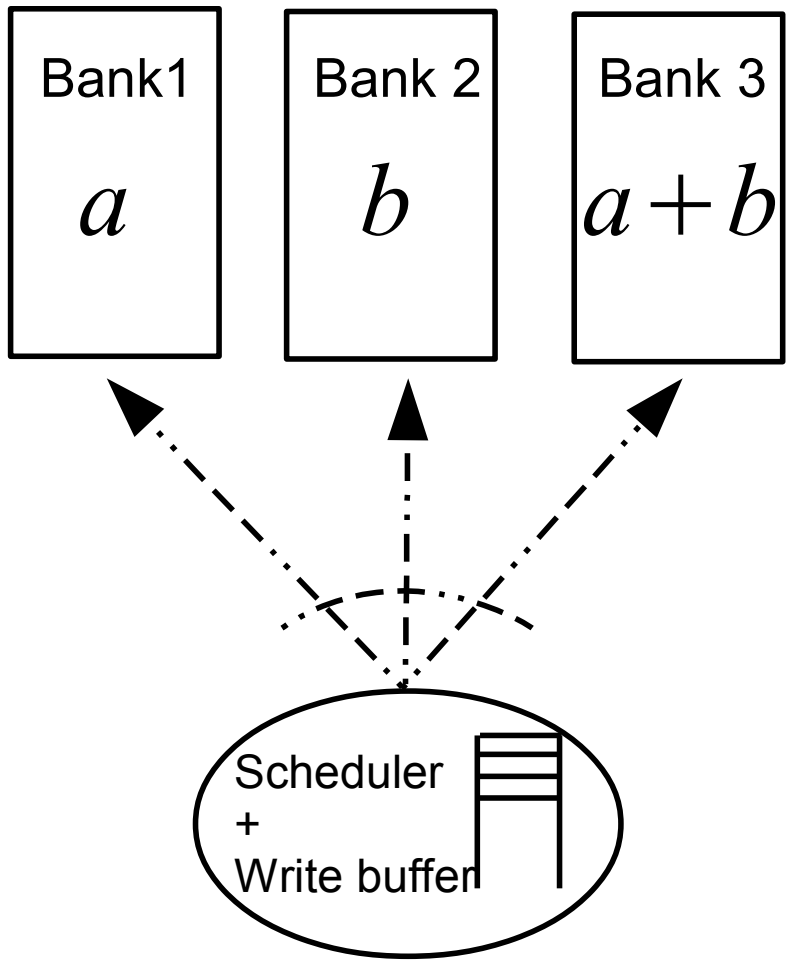
- Should we design for worst case 'read/write' access pattern?
- Backlogged write requests can be cleared when
(i) no memory access
(ii) single memory operation.

Write Buffers to Improve Write Efficiency

Potential Solution: XOR code with Write Buffer

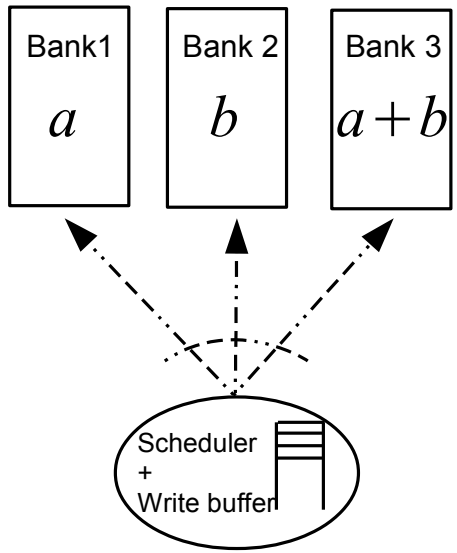
Write Buffers to Improve Write Efficiency

Potential Solution: XOR code with Write Buffer



Write Buffers to Improve Write Efficiency

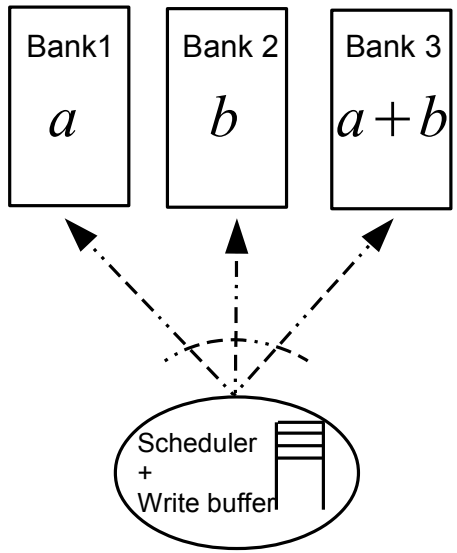
Potential Solution: XOR code with Write Buffer



- This scheme can server 2 simultaneous read or 1 write.
- Buffer some write requests and clear them when opportunity arise.
- Can a read request for data in write buffer be directly served from write buffer?

Write Buffers to Improve Write Efficiency

Potential Solution: XOR code with Write Buffer



- This scheme can server 2 simultaneous read or 1 write.
- Buffer some write requests and clear them when opportunity arise.
- Can a read request for data in write buffer be directly served from write buffer?
- Similar ideas in RAID (with non-volatile cache) literature [**?**, **?**].