Hardware Caches with Low Access Times and High Hit Ratios

Xiaodong Zhang

Ohio State University

Acknowledgement of Contributions:

Chenxi Zhang, Tongji University

Zhichun Zhu, University of Illinois, Chicago

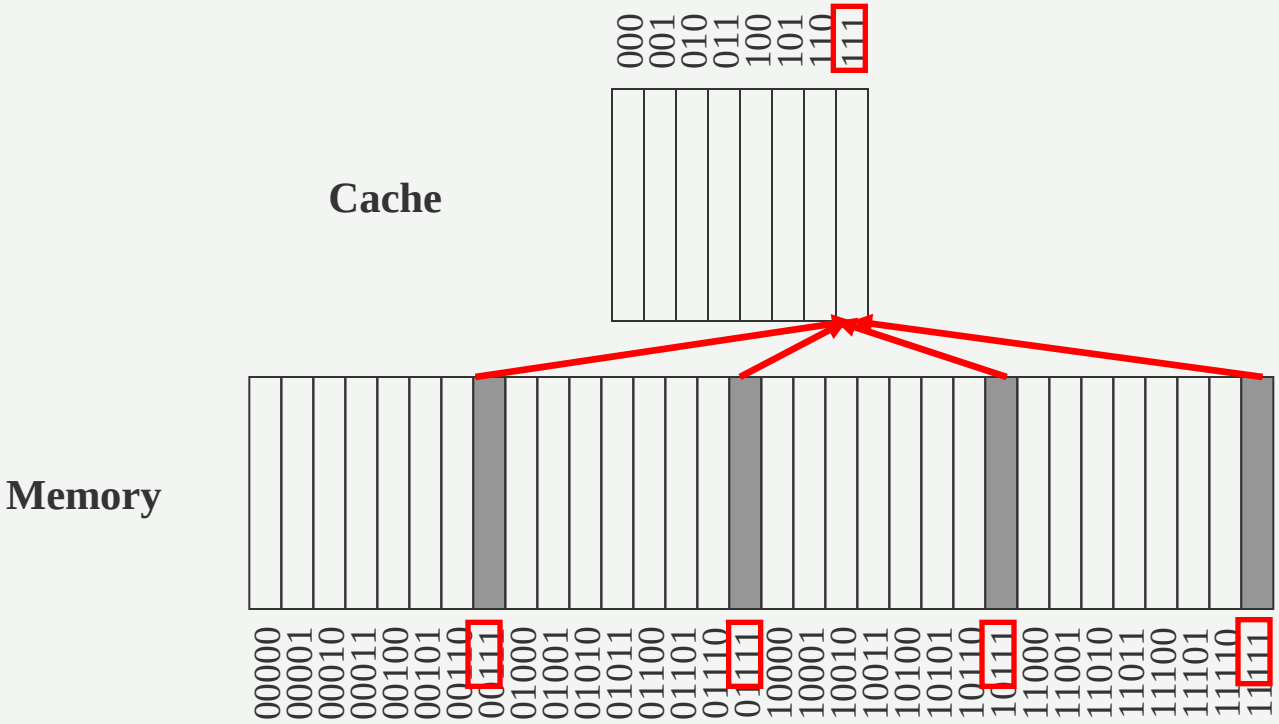
Basics of Hardware Caches

- A data item is referenced by its memory address.
- It is first searched in the cache.
- Three questions cover all the cache operations:
  - How do we know it is in the cache?
  - If it is (a hit), how do we find it?
  - If it is not (a miss), how to replace the data if the location is already occupied?

Direct-Mapped Cache

- The simplest, but efficient and commonly used.
  (Maurice Wilkes, IEEE TC 1965, a two-page paper)
- Each access is mapped to exactly one location.
- The mapping follows:
  - (memory address) mod (number of cache blocks)
- A standard block has 4 bytes (a word), but it is increasingly longer for spatial locality.

An Example of an Direct Mapped-cache



7_{10} $(00111)_2$ Mod 8_{10} = 111_2

15_{10} $(01111)_2$ Mod 8_{10} = 111_2

23_{10} $(10111)_2$ Mod 8_{10} = 111_2

31_{10} $(11111)_2$ Mod 8_{10} = 111_2

Nature of Direct-mapped Caches

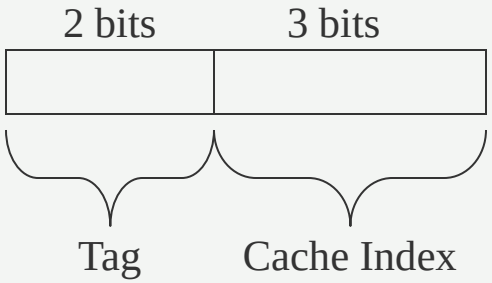
- If the number of cache blocks is a power of 2, the mapping is exactly the low-order $\log_2$ (cache size in blocks) bits of the address.

- If cache = 2^3 = 8 blocks, the 3 low-order bits are directly mapped addresses.

- The lower-order bits are also called cache index.

Tags in Direct-Mapped Caches

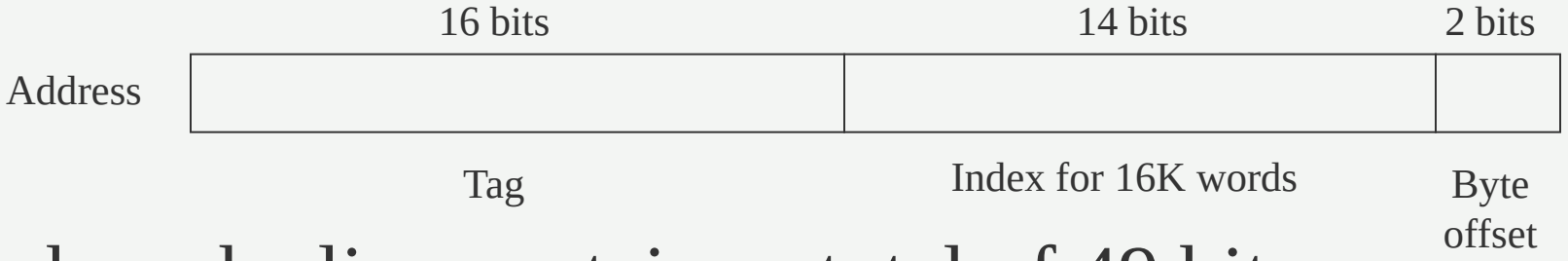
- A cache index for a cache block is not enough because multiple addresses will be mapped to the same block.
- A ``tag'' is used to make this distinction. The upper portion of the address forms the tag:



- If both the tag and index are matched between memory and cache, a ``hit'' happens.
- A valid bit is also attached for each cache block.

Allocation of Tag, Index, and Offset Bits

- Cache size: 64 Kbytes.
- Block size: 4 Bytes (2 bits for offset)
- 64 Kbytes = 16 K blocks = 2^{14} (14 bits for index)
- For a 32-bit memory address: 16 bits left for tag.

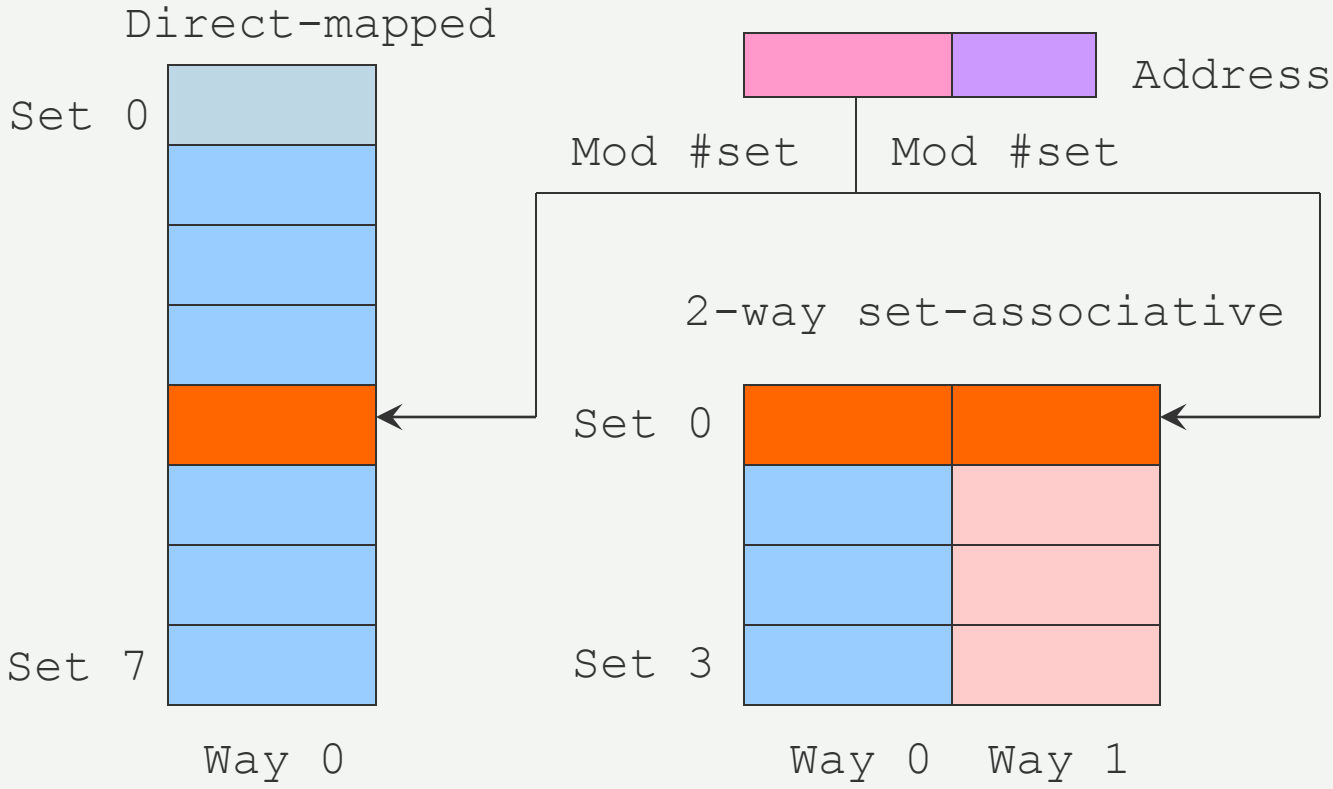


- Each cache line contains a total of 49 bits:
  - 32 bits (data of 4 Bytes)
  - 16 bits (tag)
  - 1 bit for valid bit.

Set-associative Caches

- Direct-mapping one location causes high miss rate.
- How about increasing the number of possible locations for each mapping?
- The number of locations is called ``associativity''. A set contains more than one location.
- associativity = 1 for direct-mapped cache
- Set-associative cache mapping follows:
  - (memory address) mod (number of sets)
- Associativity = number of blocks for fully associative cache.

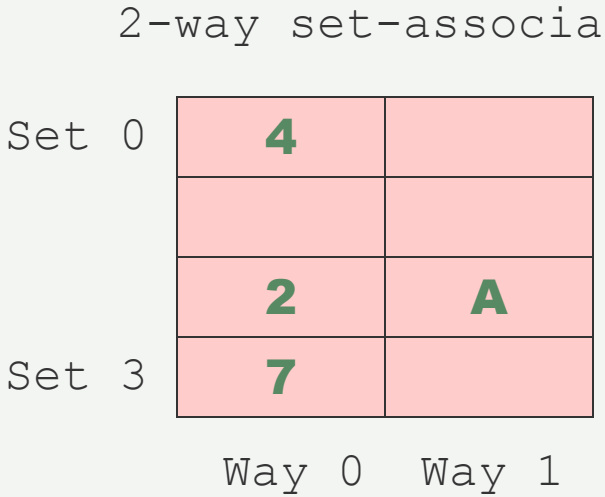
Direct-mapped vs. Set-associative



Cache Accesses

References

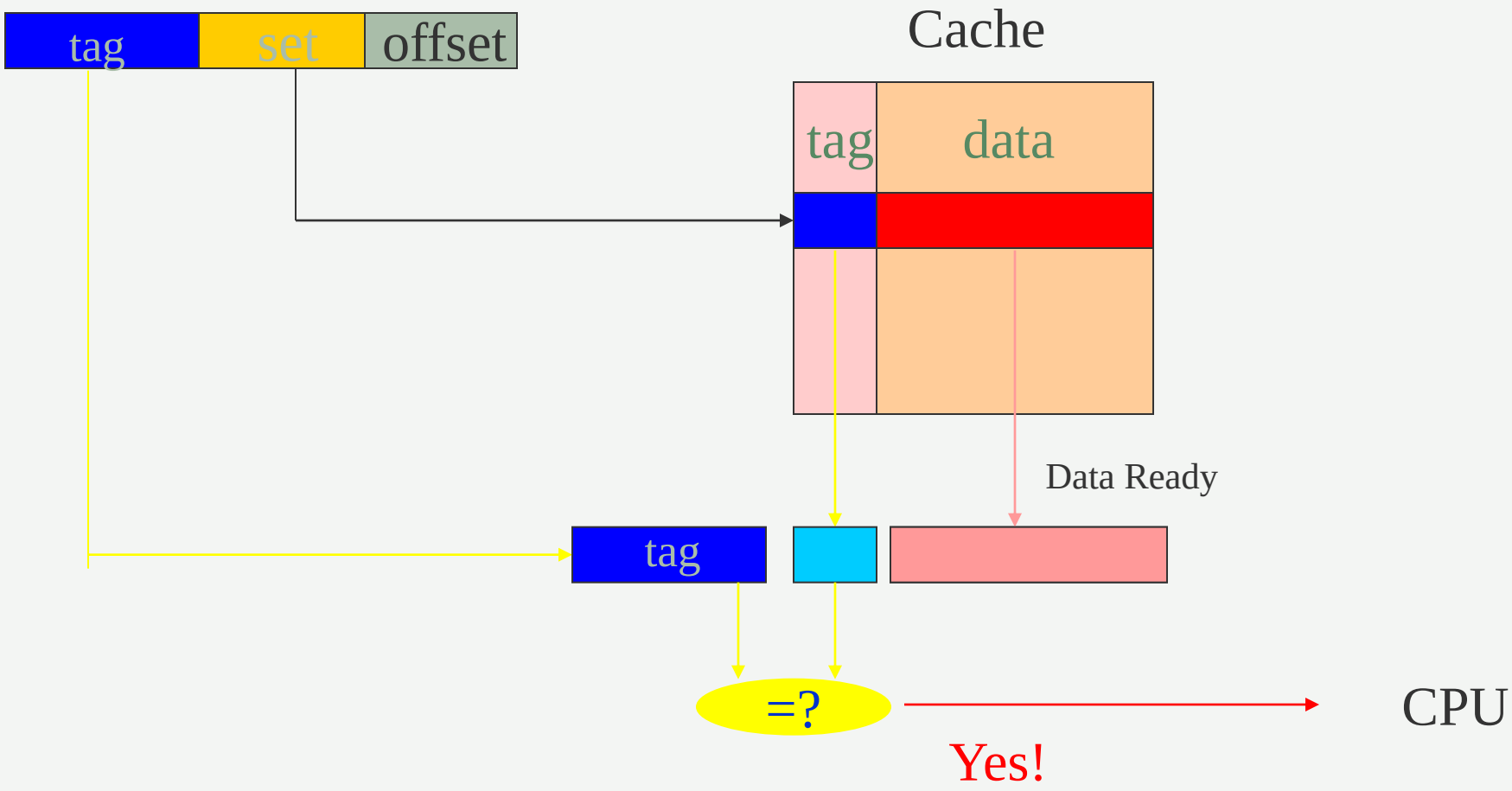
2 7 A 4 2 7 A 2



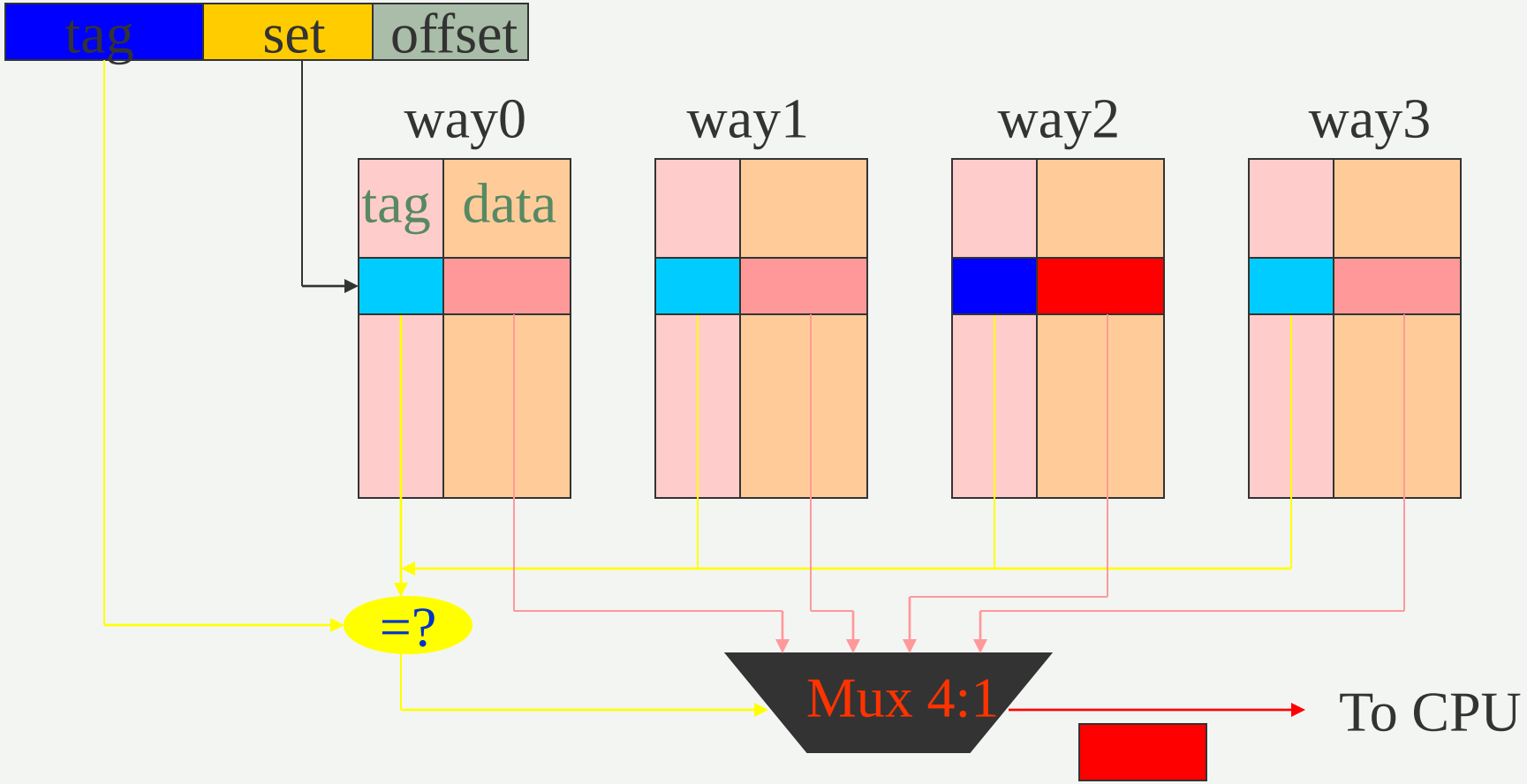
7 misses

4 misses

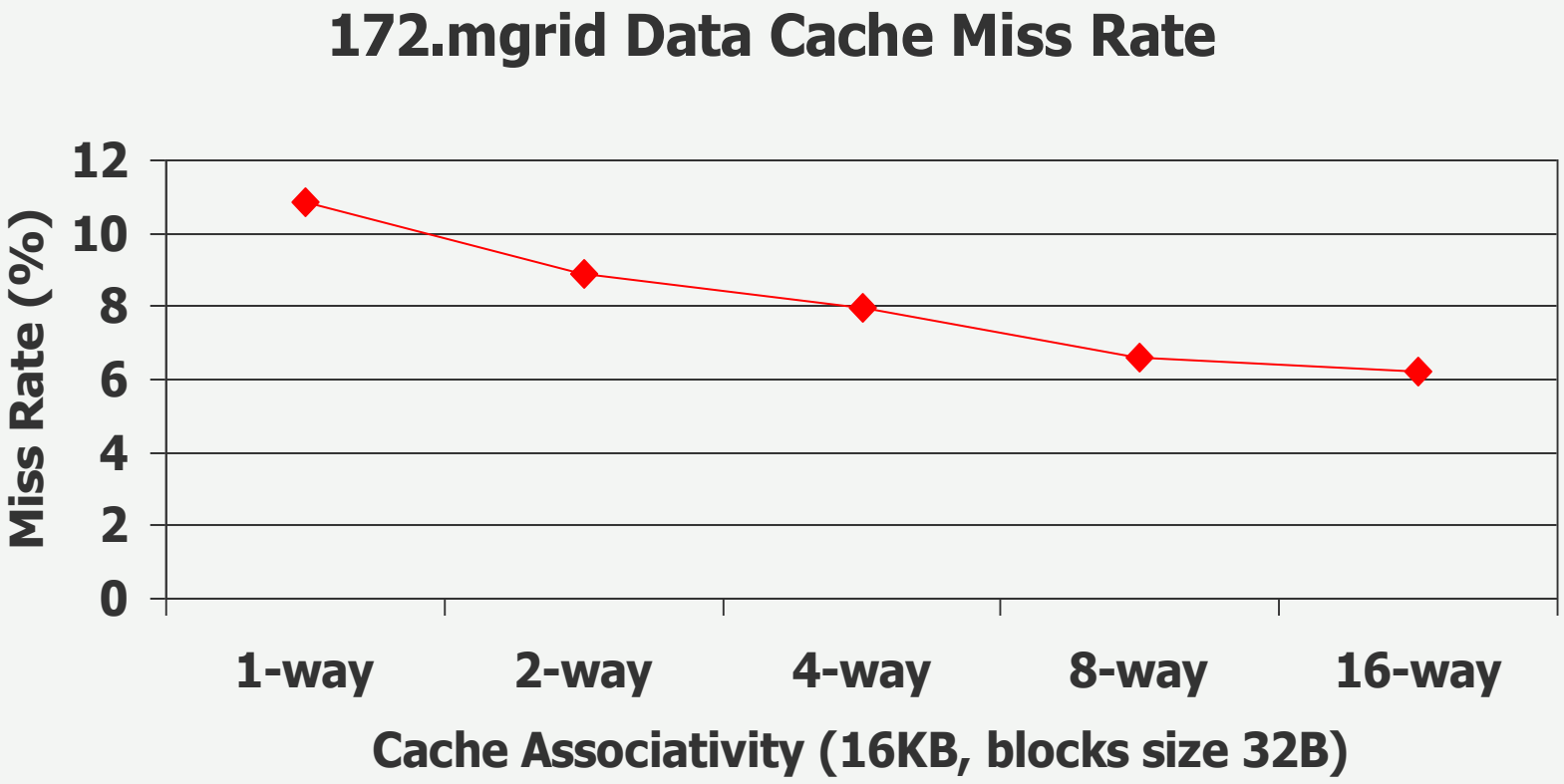
Direct-mapped Cache Operations: Minimum Hit Time



Set-associative Cache Operations: Delayed Hit Time



Set-associative Caches Reduce Miss Ratios



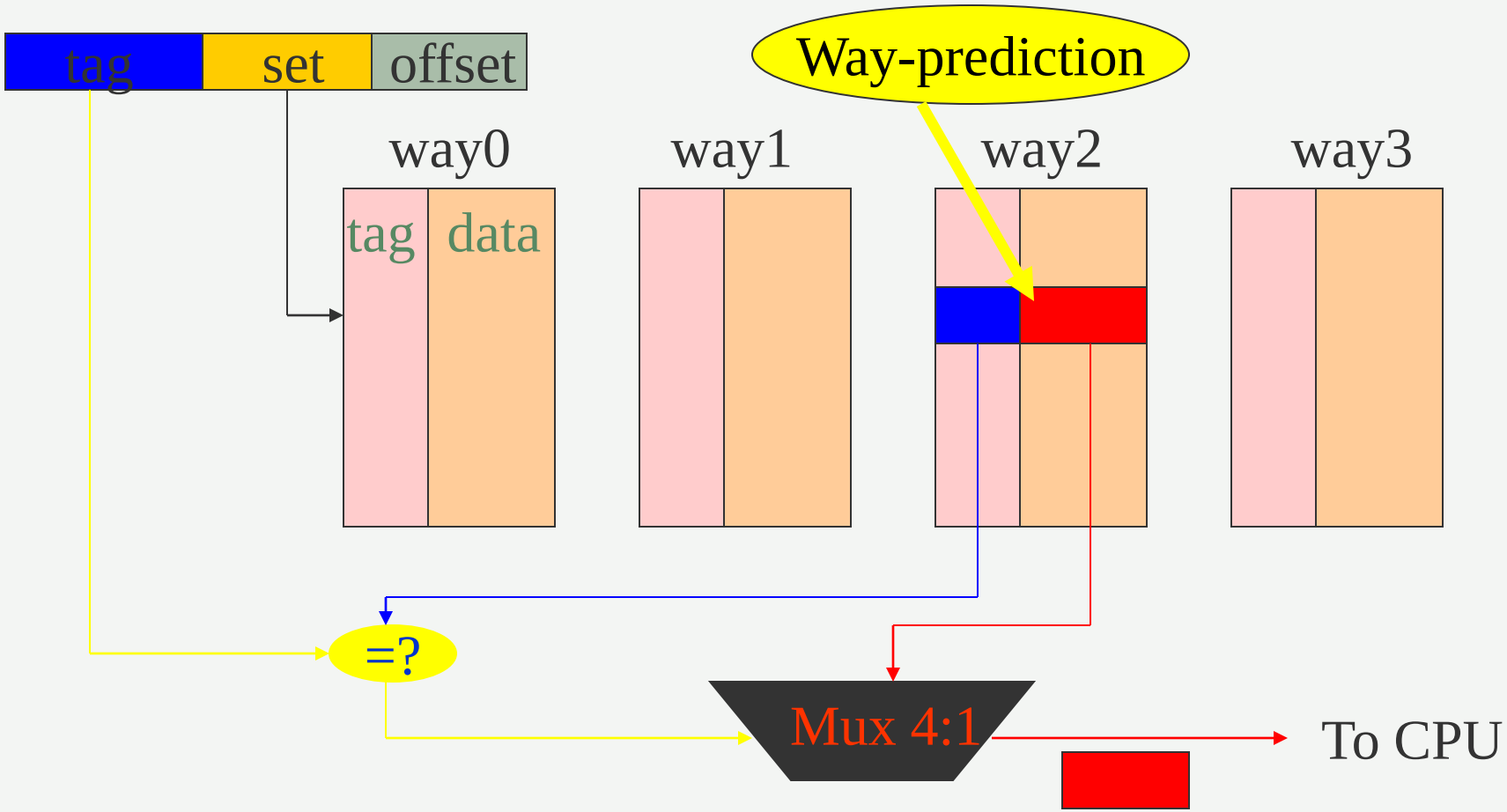
172.mgrid: SPEC 2000, multi-grid solver

Trade-offs between High Hit Ratios (SA) and Low Access Times (DM)

- Set- associative cache achieves high hit-ratios: 30% higher than that of direct-mapped cache.
- But it suffers high access times due to
  - Multiplexing logic delay during the selection.
  - Tag checking, selection, and data dispatching are sequential.
- Direct-mapped cache loads data and checks tag in parallel: minimizing the access time.
- Can we get both high hit ratios and low access times?
- The Key is the Way Prediction: speculatively determine which way is the hit so that only that way is accessed.

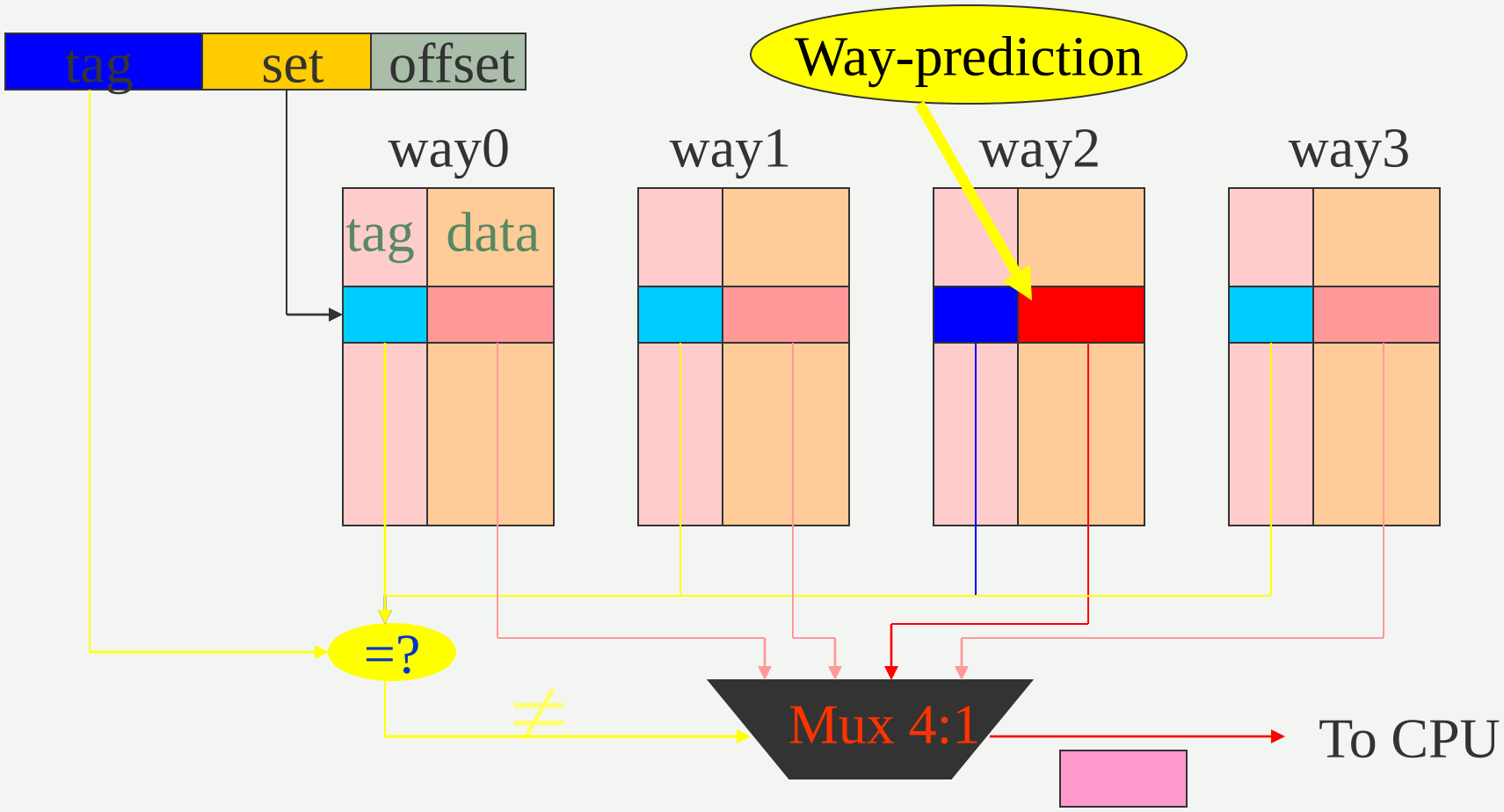
Best Case of Way Prediction: First Hit

Cost: way prediction only



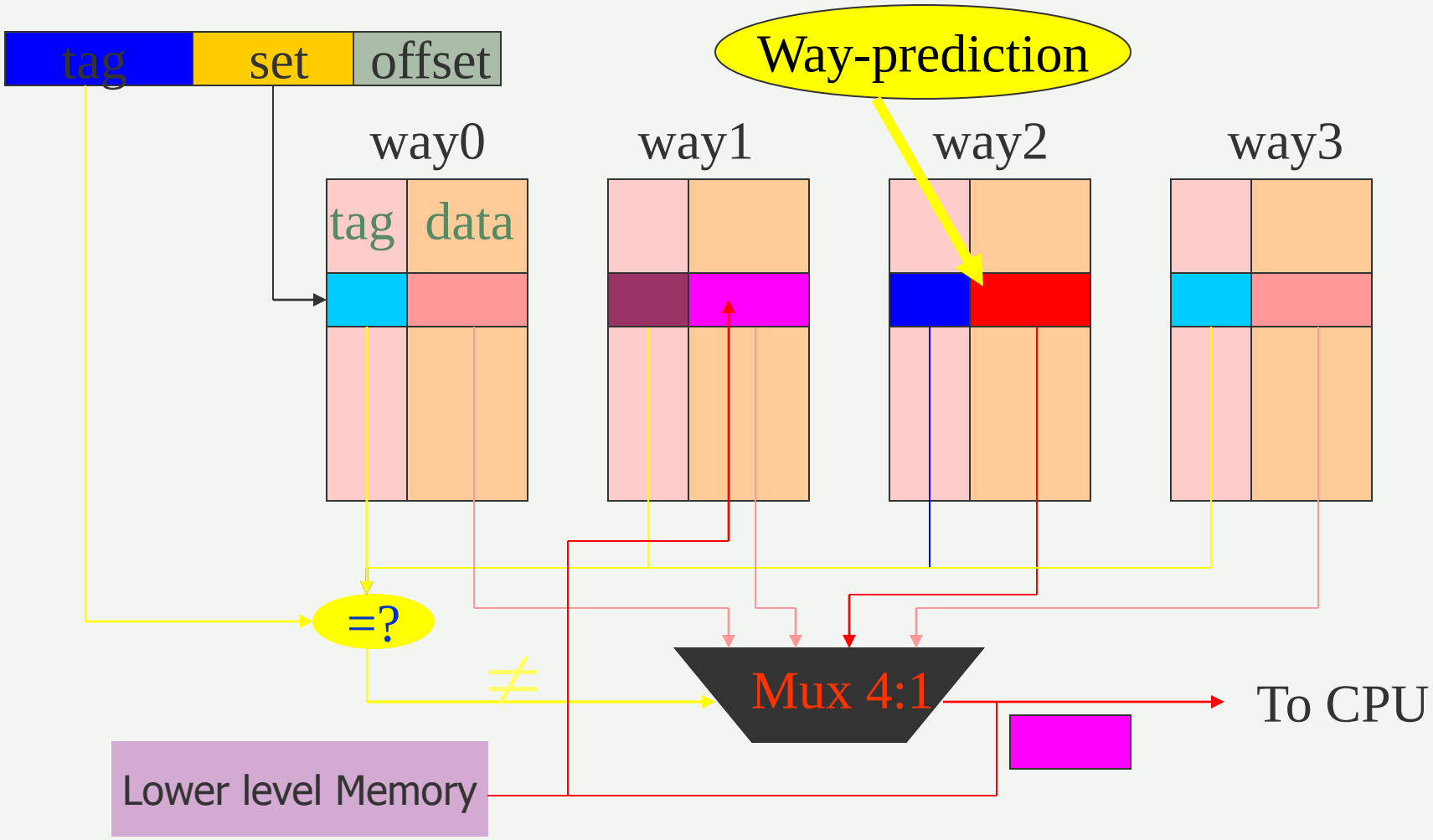
Way Prediction: Non-first Hit (in Set)

Cost: way prediction + selection in set



Worst Case of Way-prediction: Miss

Cost: way prediction + selection in set + miss



MRU Way Predictions

- Chang, et. al., ISCA'87, (for IBM 370 by IBM)
- Kessler, et. al., ISCA'89. (Wisconsin)

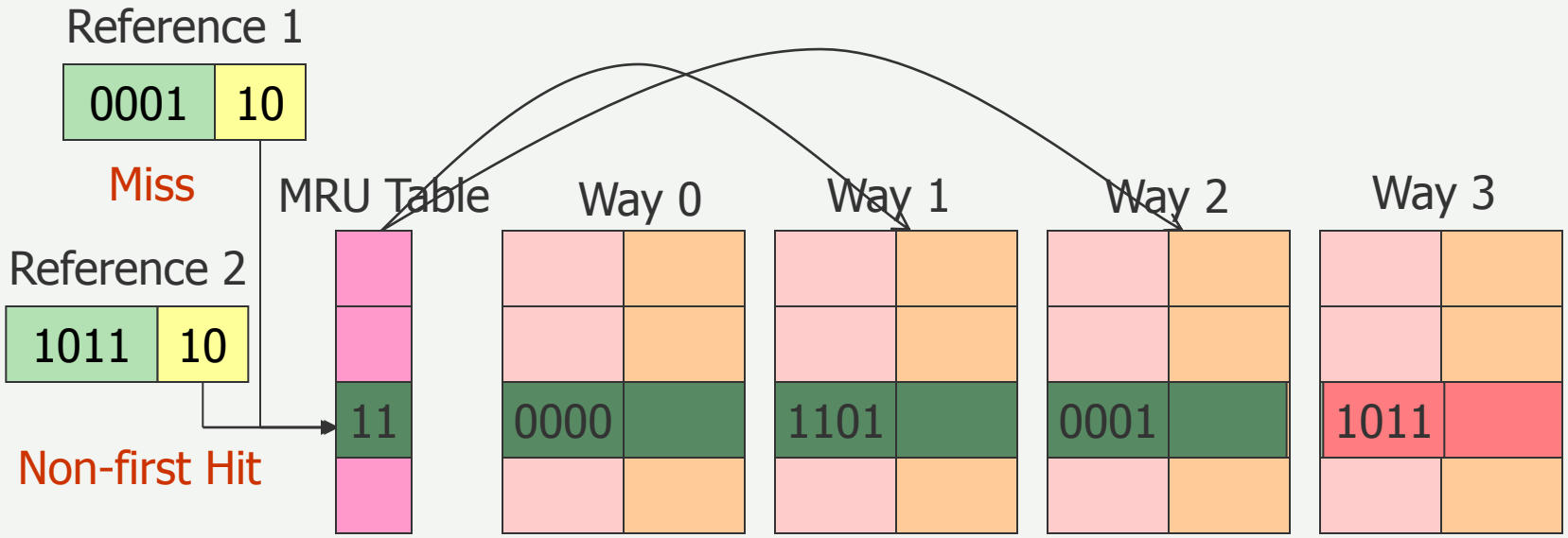
- Mark the Most Recent Use (MRU) block in each set.

- Access this block first. If hits, low access time.

- If the prediction is wrong, search other blocks.

- If the search fails in the set, it is a miss.

MRU Way Prediction



Limits of MRU Set-Associative Caches

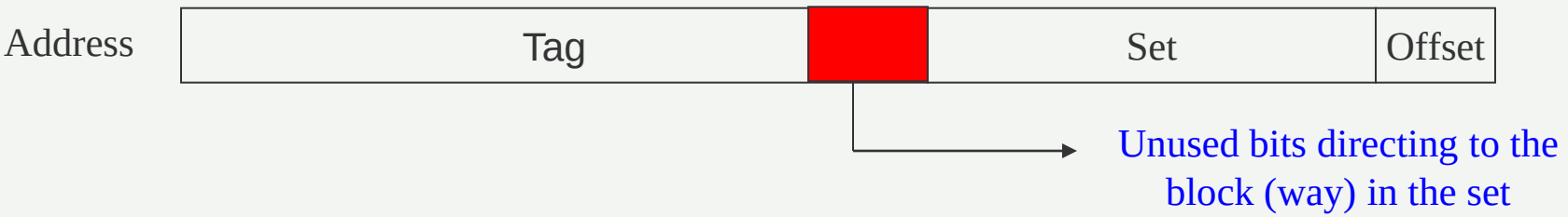
- First hit is not equivalent to a direct-mapped hit.
  - MRU index is fetched before accessing the block (either cache access cycle is lengthened or additional cycle is needed).
- The MRU location is the only search entry in the set.
  - The first hit ratio can be low in cases without many repeated accesses to single data (long reuse distance), such as loops.
- MRU cache can reduce access times of set-associative caches by a certain degree but
  - It still has a big gap with that of direct-mapped cache.
  - It can be worse than set-associative cache when first-hits to MRU locations are low.

Multi-column Caches: Fast Accesses and High Hit Ratio

- Zhang, et. al., IEEE Micro, 1997. (Tongji, and Ohio State)

- Objectives:
  - Each first hit is equivalent to a direct-mapped access.
  - Maximize the number of first hits.
  - Minimize the latency of non-first-hits.
  - Additional hardware should be simple and low cost.

Multi-Column Caches: Major Location



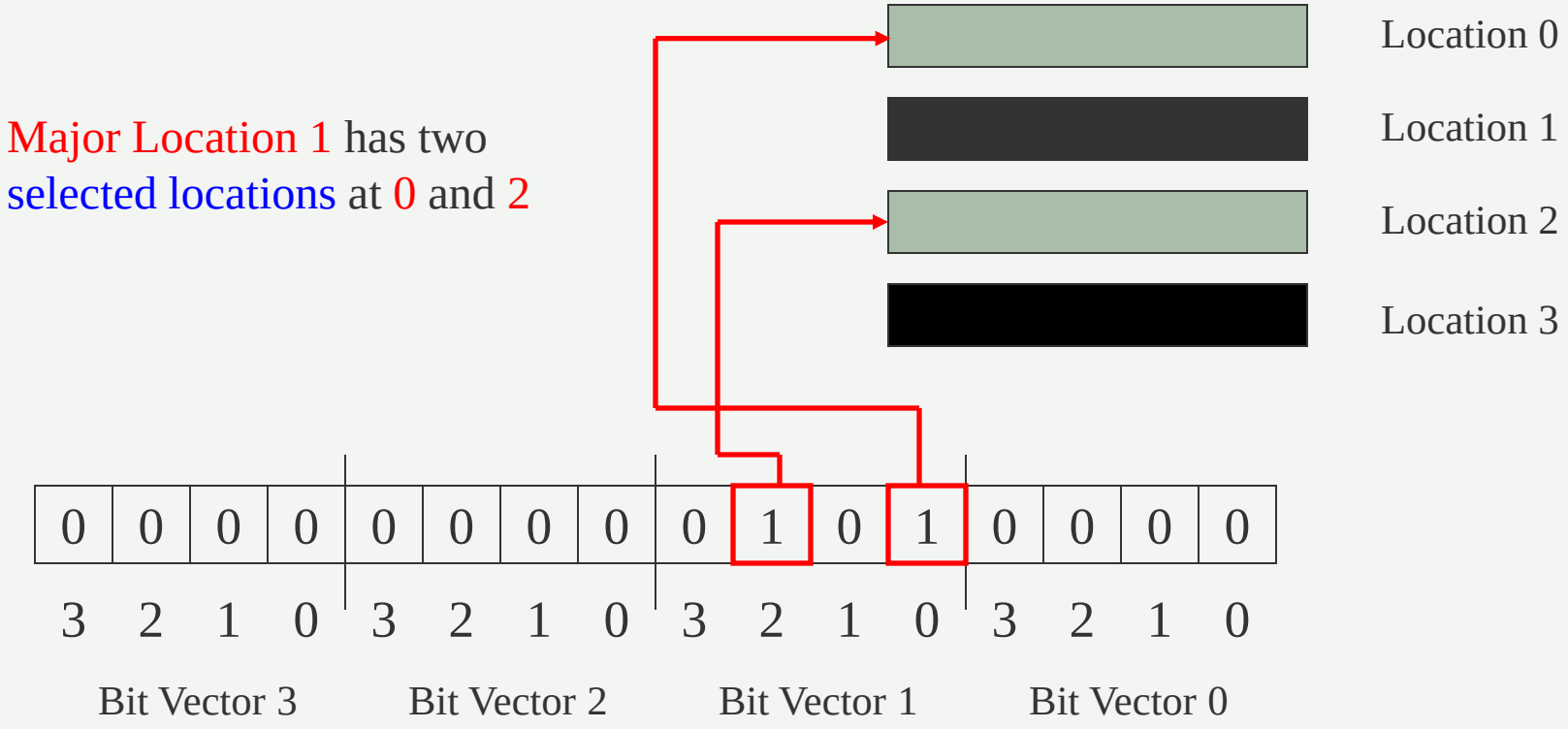
- The unused bits and ``set bits'' generate a direct-mapped location: Major Location.
- A major location mapping = direct-mapping.
- Major location only contains a MRU block either loaded from memory or just accessed.

Multi-column Caches: Selected Locations

- Multiple blocks can be direct-mapped to the same major location, but only MRU is the major.
- The non-MRU blocks are stored in other empty locations in the set: Selected Locations.
- If ``other locations'' are used for their own major locations, there will be no space for selected ones.
- Swap:
  - A block in selected location is swapped to major location as it becomes MRU.
  - A block in major location is swapped to a selected location after a new block is loaded in it from memory.

Multi-Column: Indexing Selected Locations

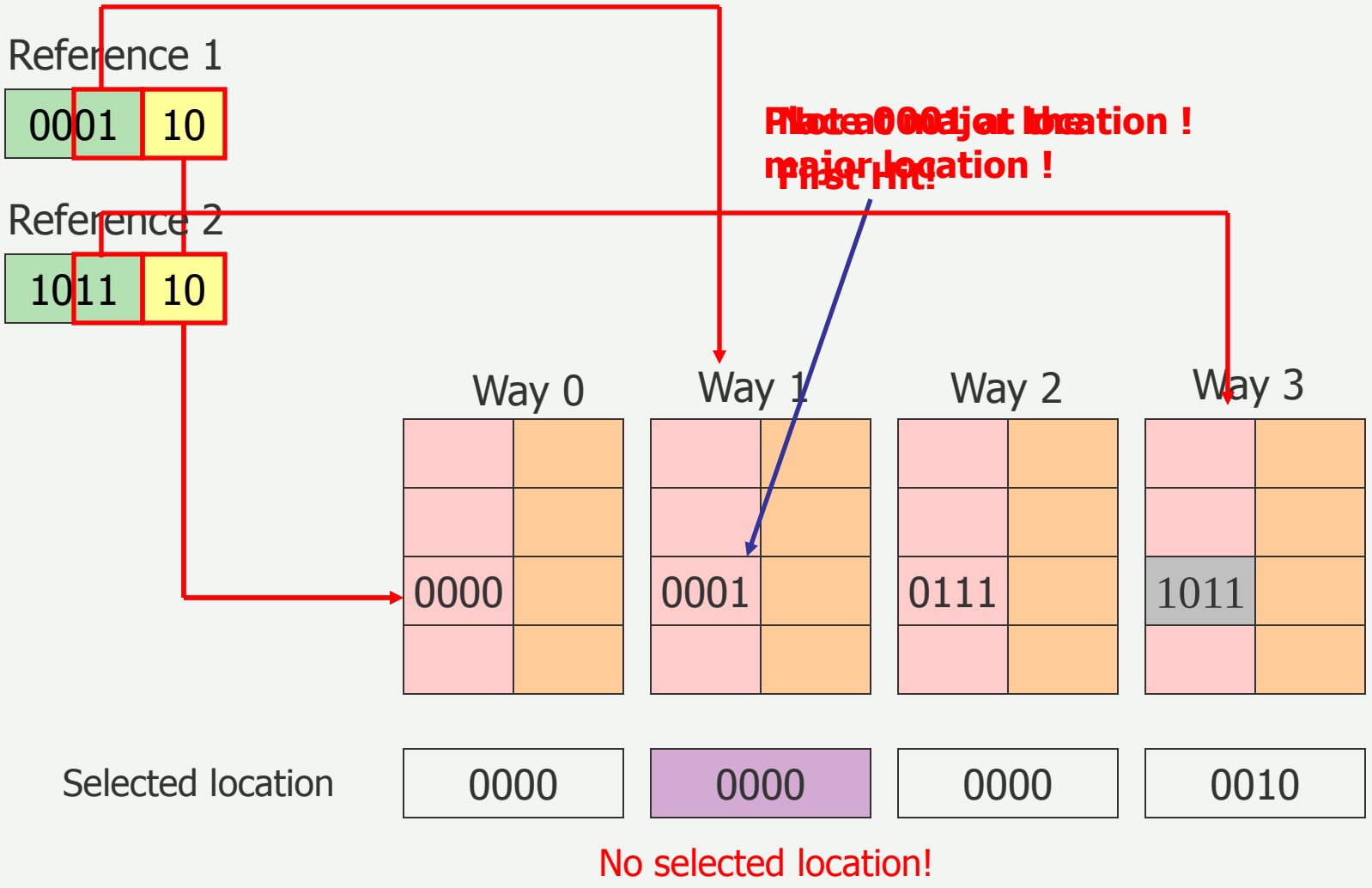
- The selected locations associated with its major location are indexed for a fast search.



Summary of Multi-column Caches

- A major location in a set is the direct-mapped location of MRU.
- A selected location is the direct-mapped location but non-MRU.
- An selected location index is maintained for each major location.
- A ``swap'' is used to ensure the block in the major location is always MRU.

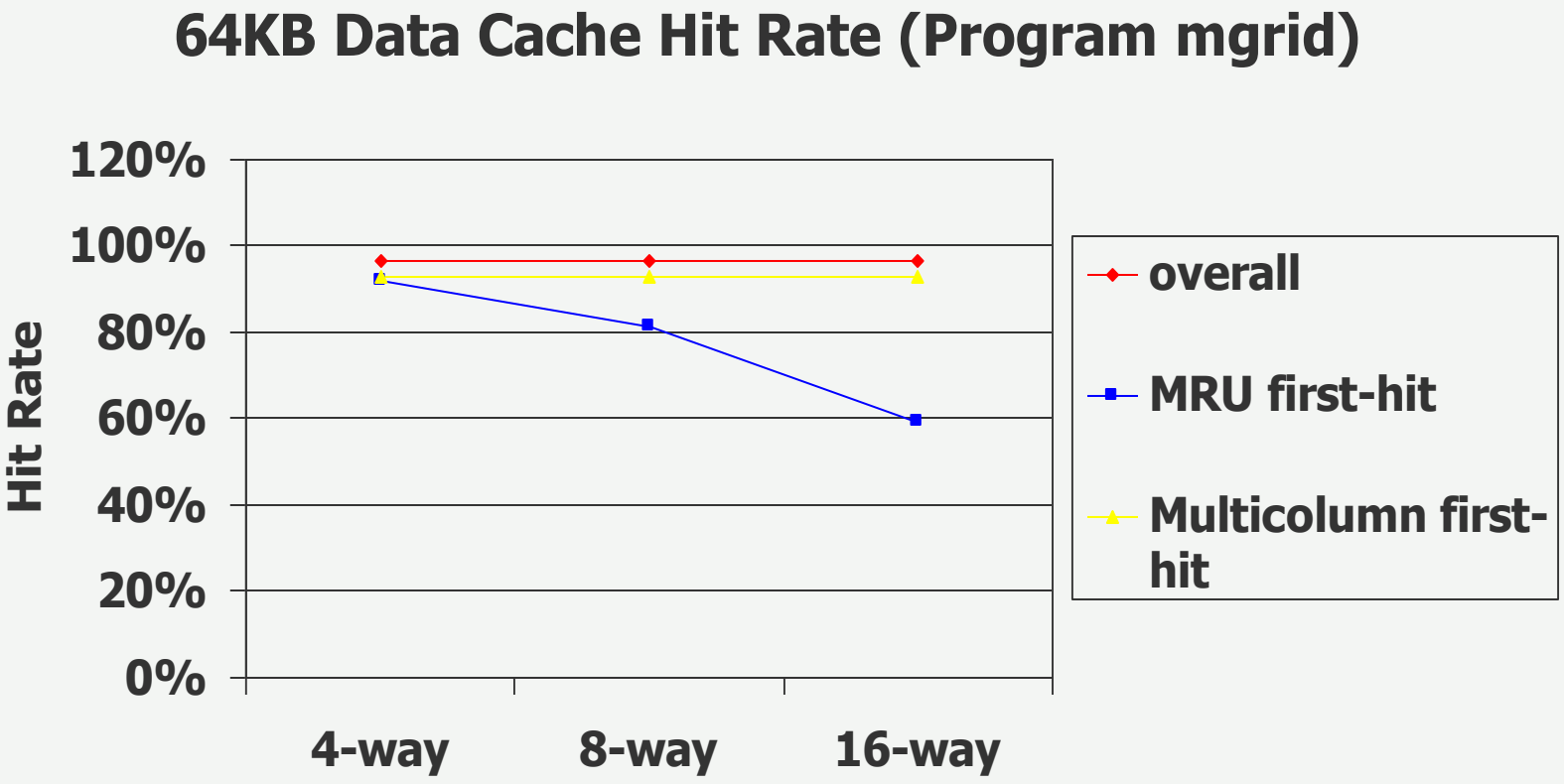
Multi-column Cache Operations



Performance Summary of Multi-column Caches

- Hit ratio to the major locations is about 90%.
- The hit ratio is higher than that of direct-mapped cache due to high associativity while keeps low access time of direct-mapped cache in average.
  - First-hit is equivalent to direct-mapped.
  - Non-first hits are faster than set-associative caches.
- Not only outperform set associative caches but also
  - Column-associative caches (two way only, ISCA'93, MIT).

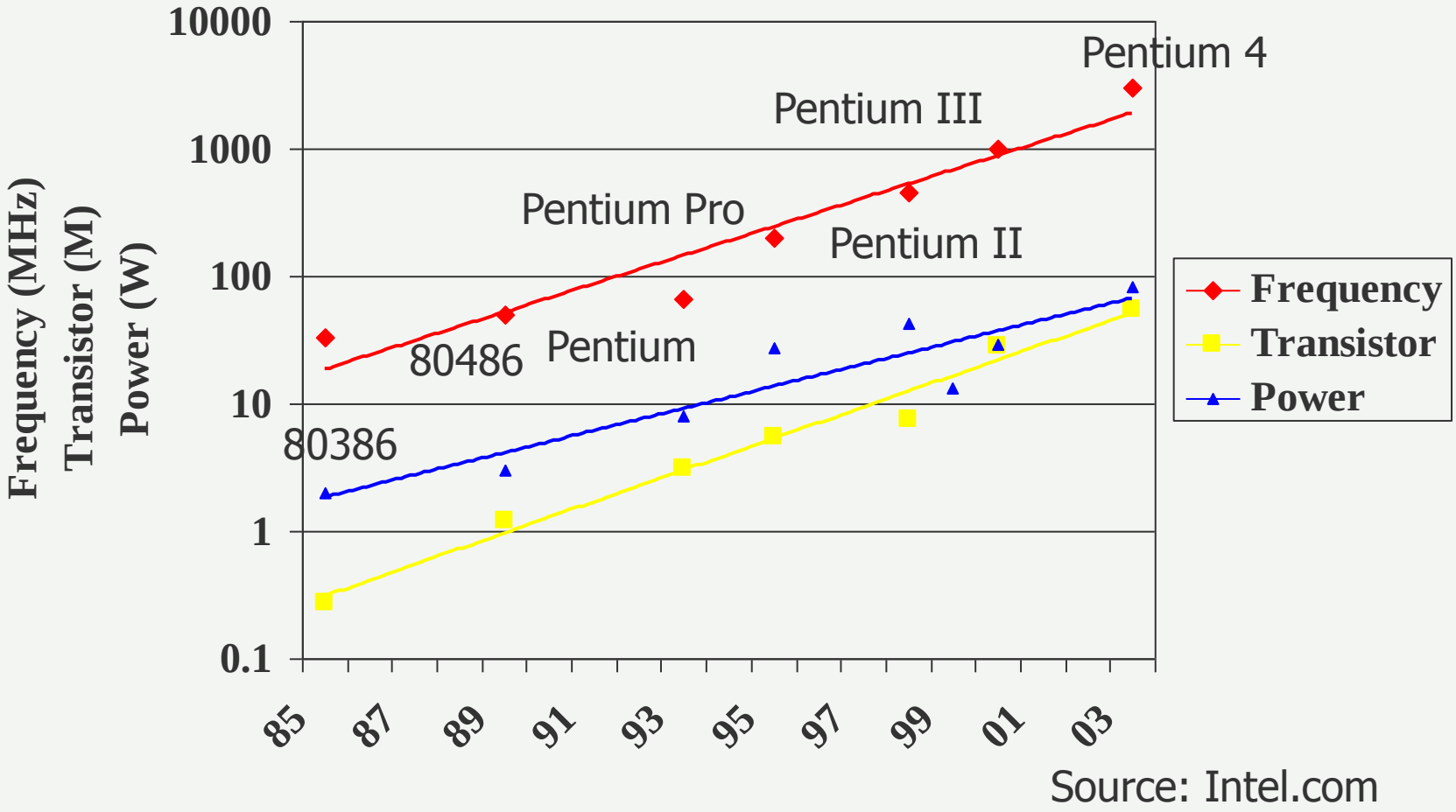
Comparing First-hit Ratios between Multicolumn and MRU Caches



Some Complexity of Multi-column Caches

- Search of the selected locations can be
  - sequential based on the vector bits of each set, or
  - parallel with a multiplexor for a selection.
- If a mapping finds its major location is occupied by a selected location in another major location group,
  - either search the bit vectors to set the bit to 0, or simply
  - Replace it by the major location data. When the selected location is searched, it will be a miss.
- The index may be omitted by only relying on the swapping.
- Partial indexing by only tracing one selected location.

Multi-column Technique is Critical for Low Power Caches



Source: Intel.com

Importance of Low-power Designs

- Portable systems:
  - Limited battery lifetime
- High-end systems
  - Cooling and package cost
    - > 40 W: 1 W → $1
    - Air-cooled techniques: reaching limits
  - Electricity bill
  - Reliability

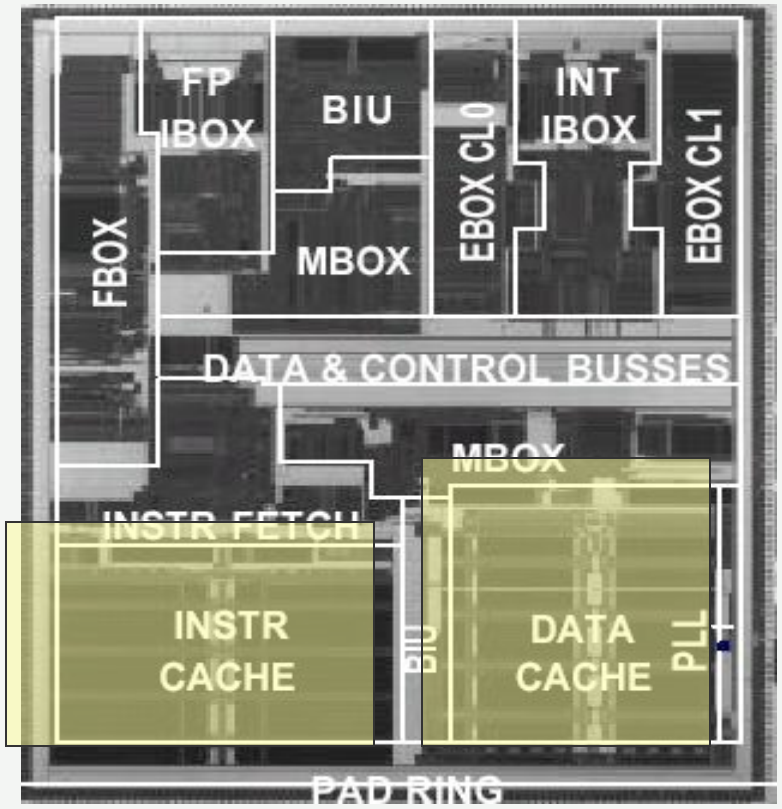
Low-power Techniques

- Physical (CMOS) level
- Circuit level
- Logic level
- Architectural level
- OS level
- Compiler level
- Algorithm/application level

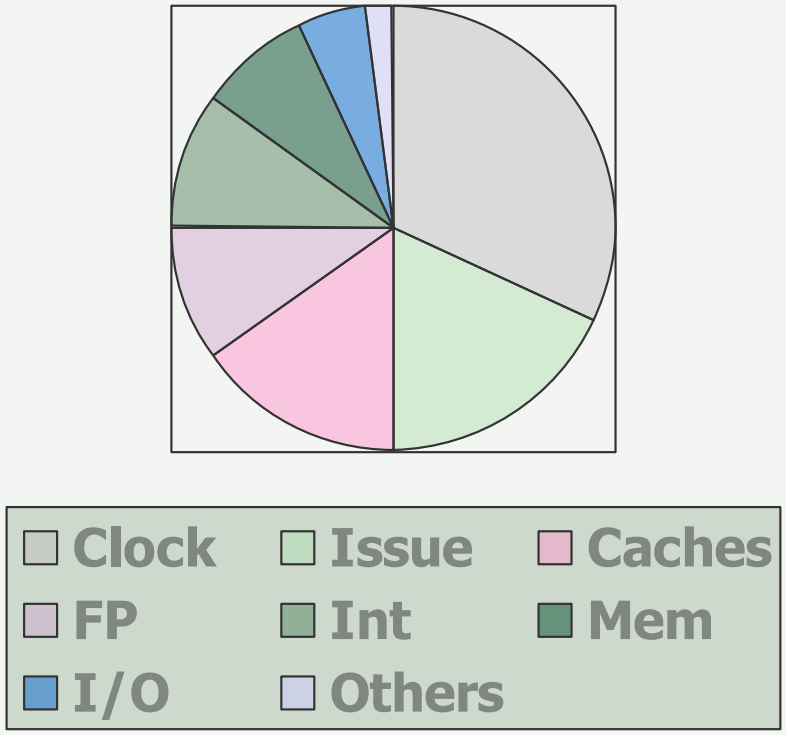
Tradeoff between Performance and Power

- Objects for general-purpose system
  - Reduce power consumption without degrading performance
- Common solution
  - Access/activate resources only when necessary
- Question
  - When is necessary?

On-chip Caches: Area & Power (Alpha 21264)

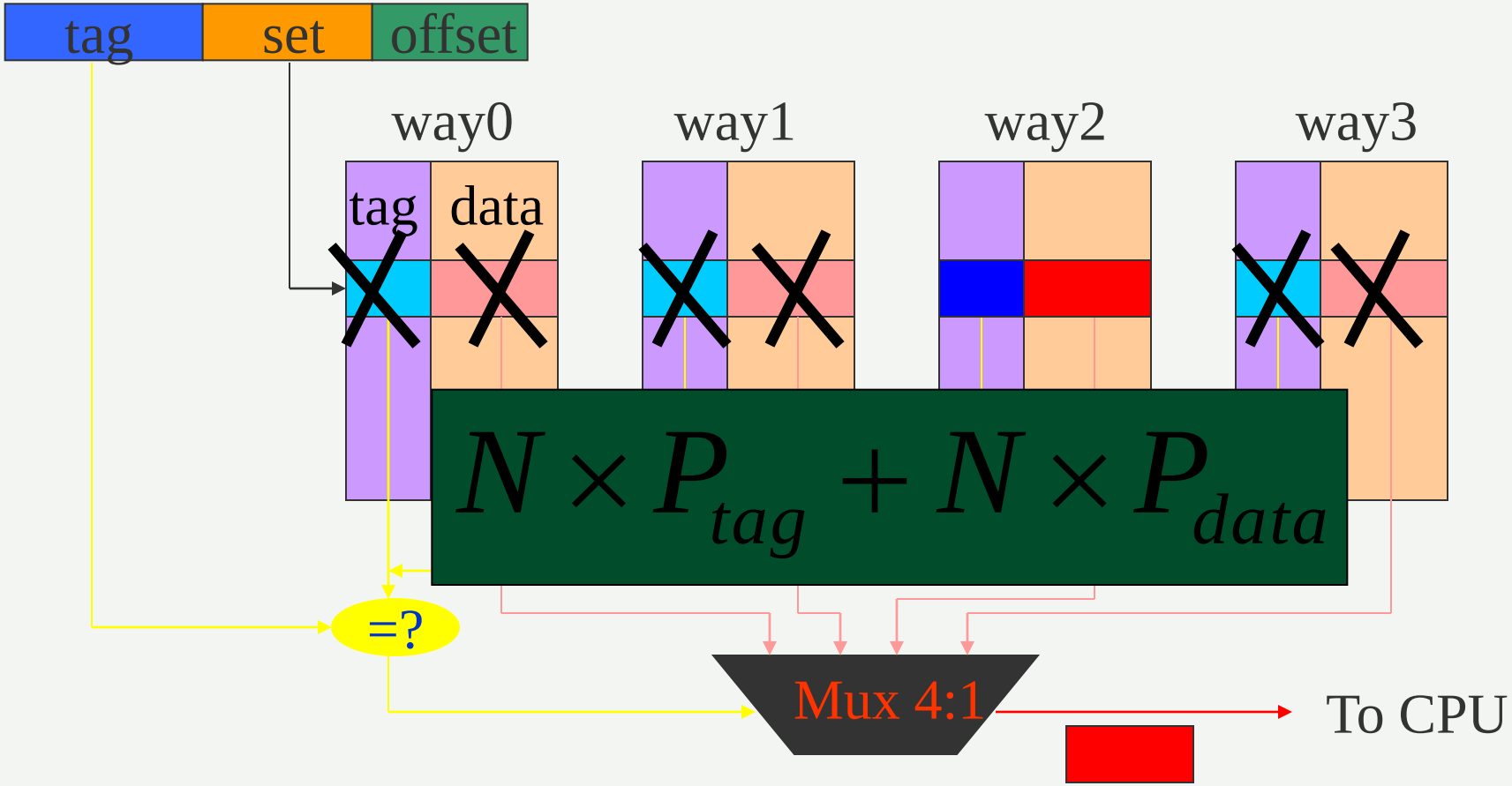


Power Consumption



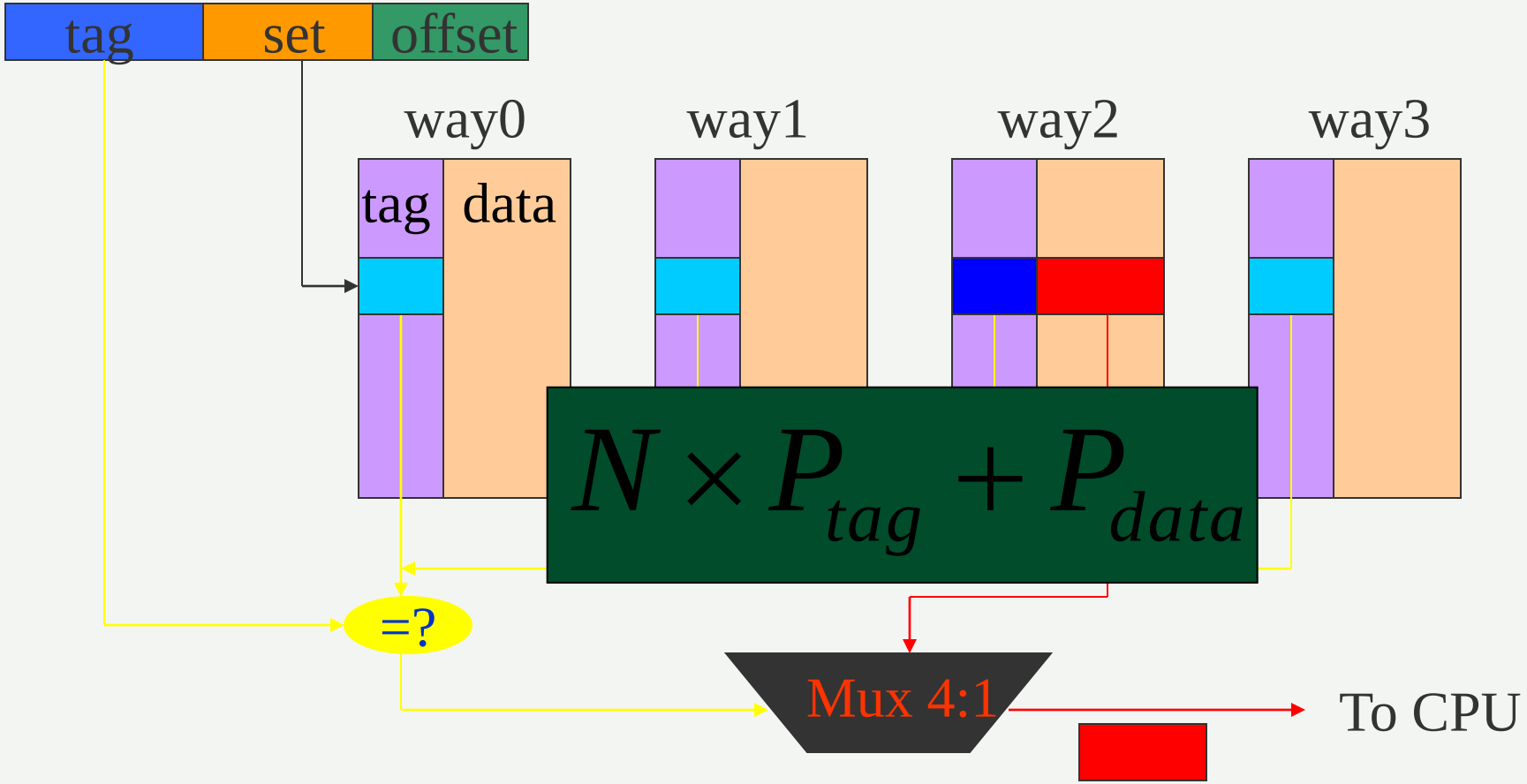
Source: CoolChip Tutorial

Standard Set-associative Caches: Energy Perspective



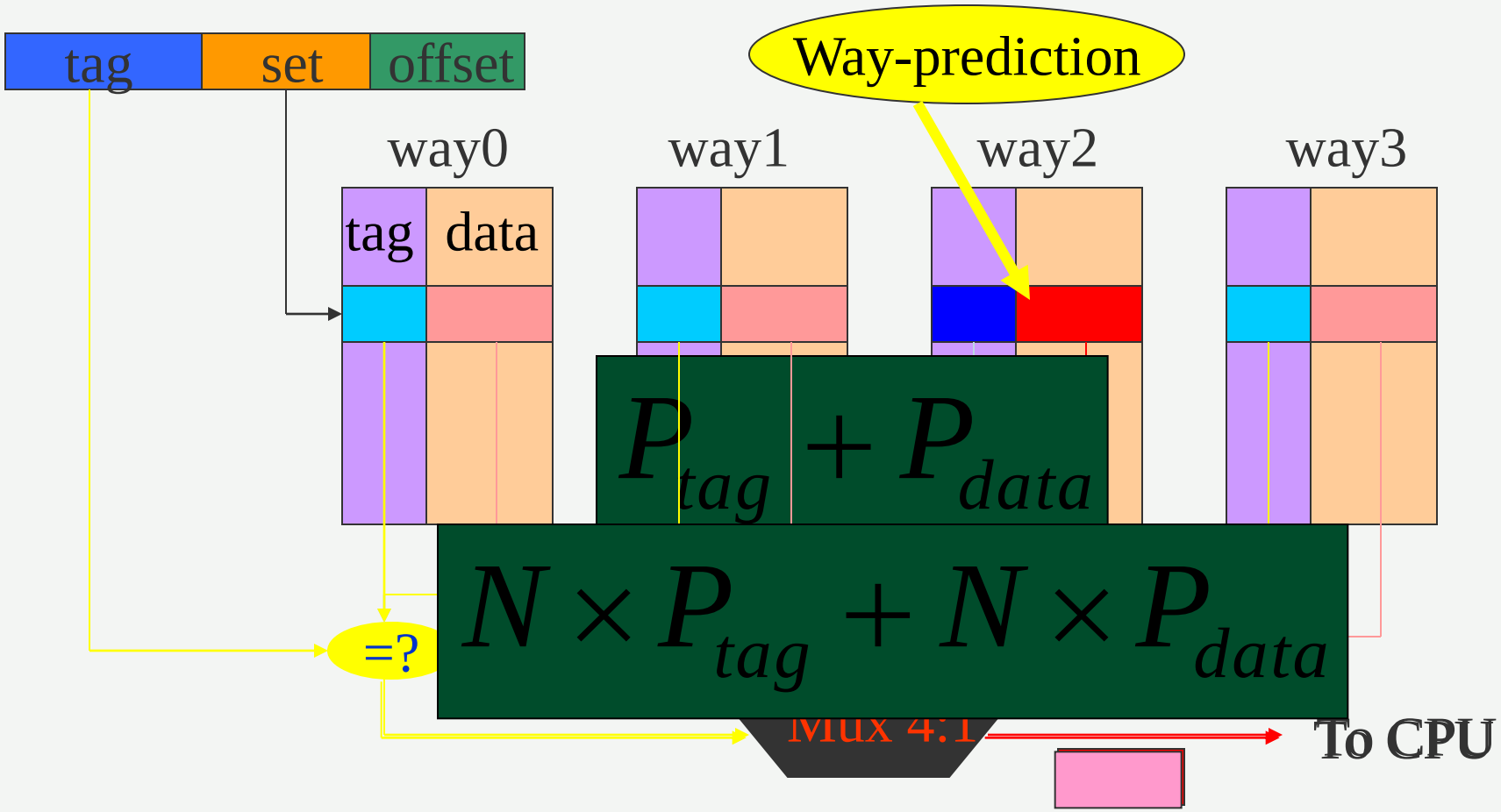
N : cache associativity, P_{tag} : tag power, P_{data} : data power

Phased Cache: tag checking first



N : cache associativity, P_{tag} : tag power, P_{data} : data power

Way-prediction Cache

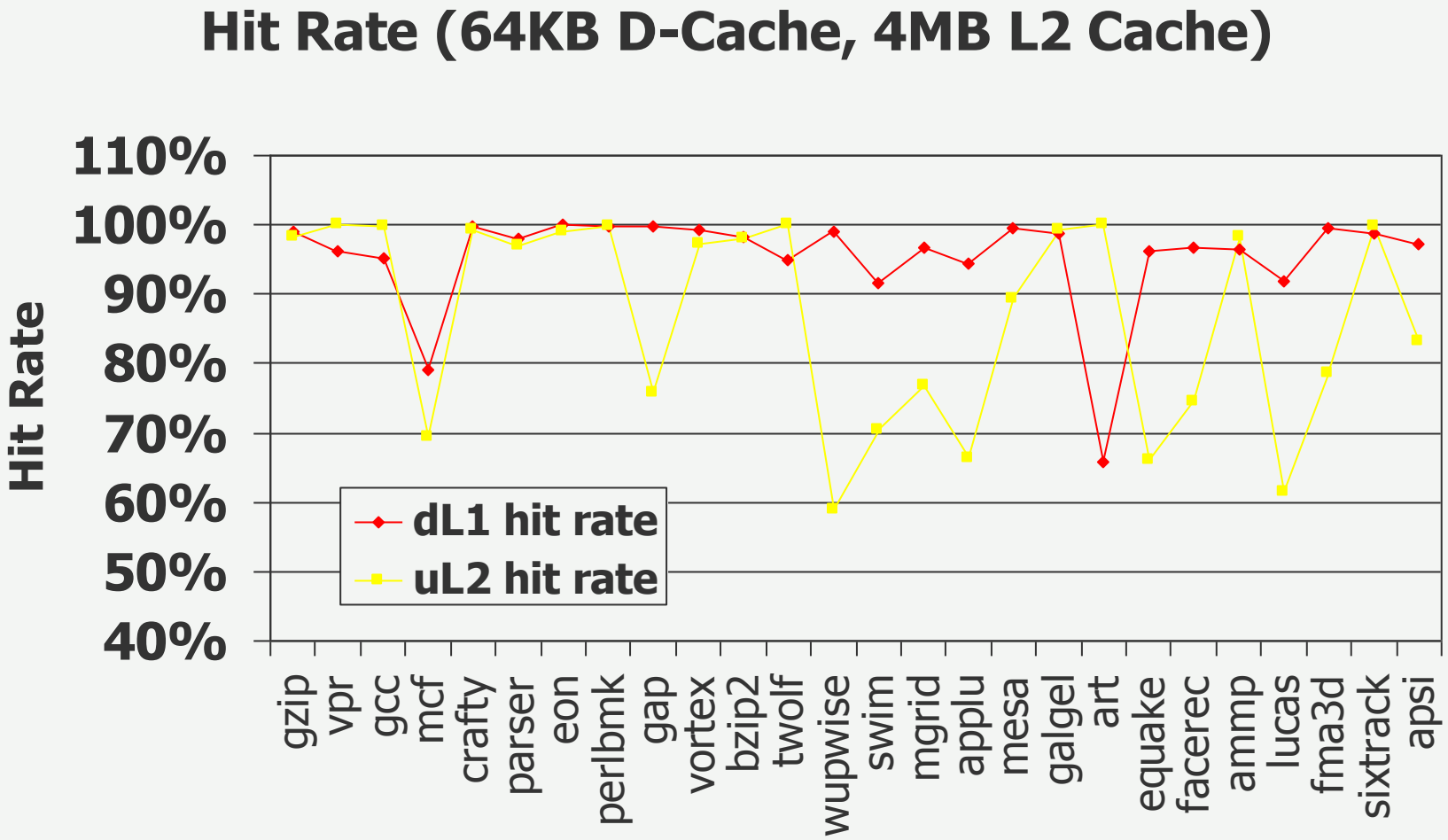


N : cache associativity, P_{tag} : tag power, P_{data} : data power

Limits of Existing Techniques

- Way-prediction caches
  - Benefits from cache hits $(P_{tag} + P_{data})$
  - Effective for programs with strong locality
- Phased caches
  - Benefits from cache misses $(N \times P_{tag})$
  - Effective for programs with weak locality

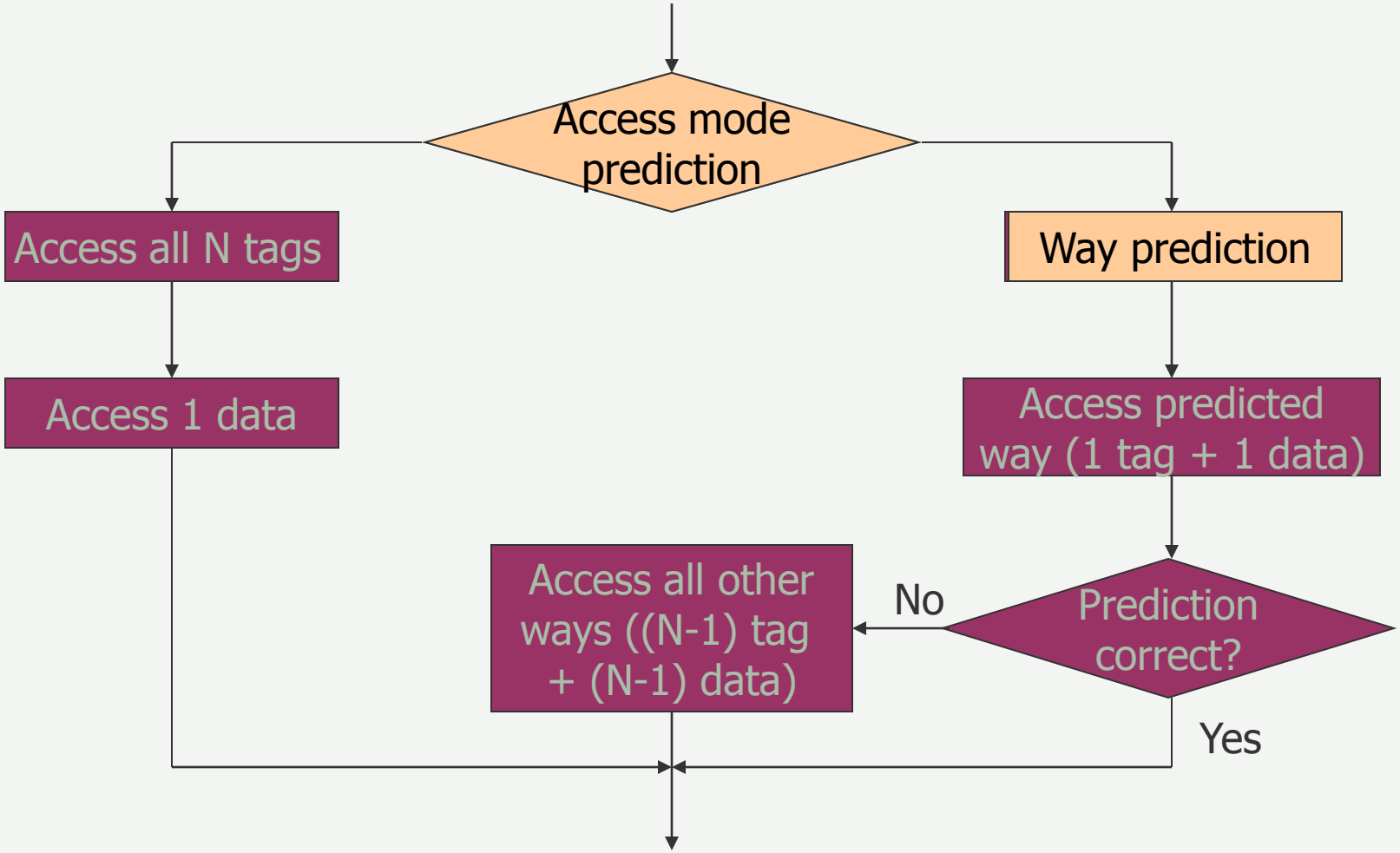
Cache Hit Ratios are Very Different



Cache Optimization Subject to Both Power and Access Time

- Objectives
  - Pursue lowest access latency and power consumption for both cache hit and miss.
  - Achieve consistent power saving across a wide range of applications.
- Solution
  - Apply way-prediction to cache hits and phase cache to misses.
  - Access mode prediction (AMP) cache.
  - Zhu and Zhang, IEEE Micro, 2002 (Ohio State)

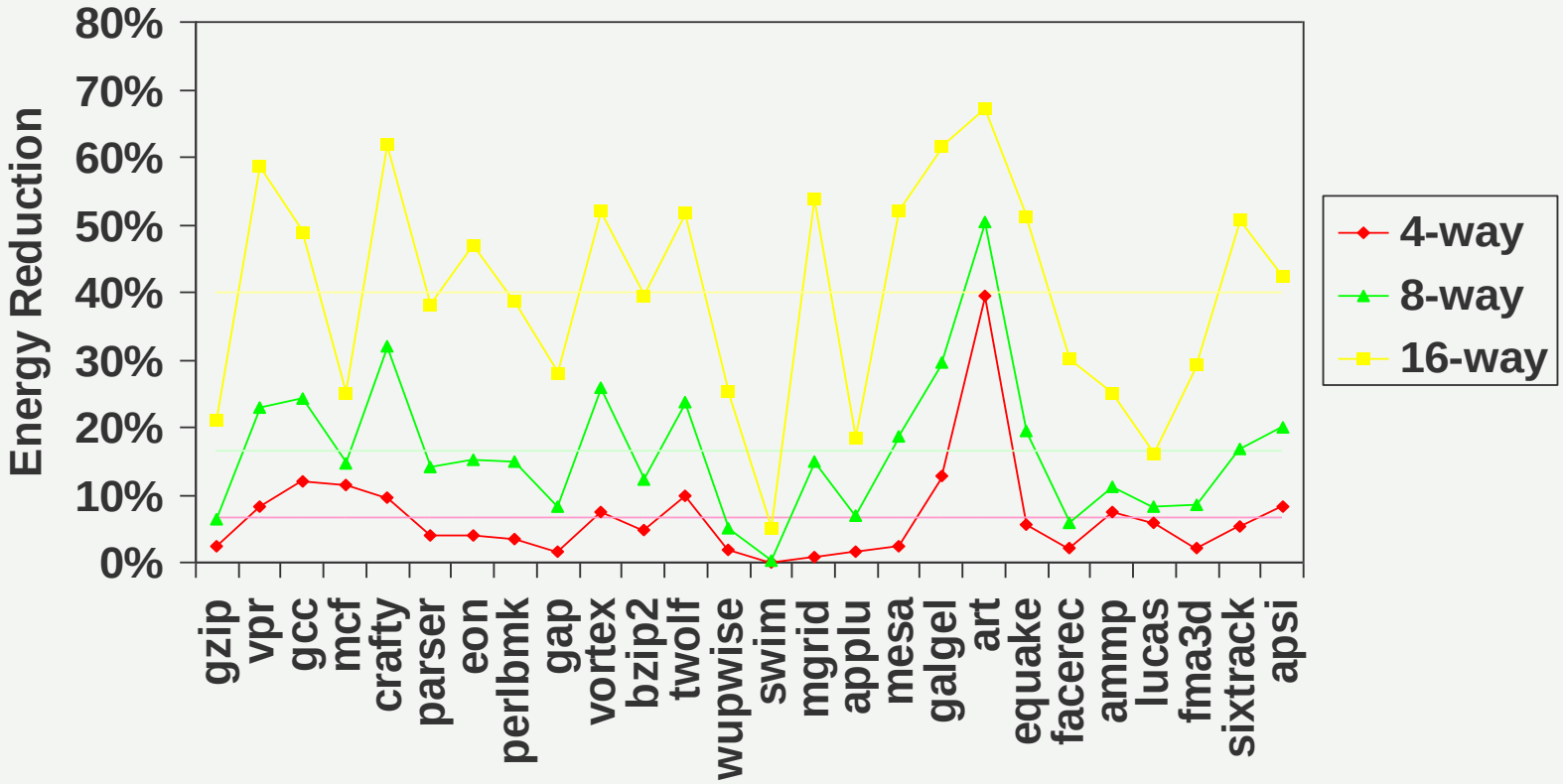
AMP Cache


Access mode prediction
Access all N tags
Access 1 data
Way prediction
Access predicted way (1 tag + 1 data)
Prediction correct?
No
Yes
Access all other ways ((N-1) tag + (N-1) data)

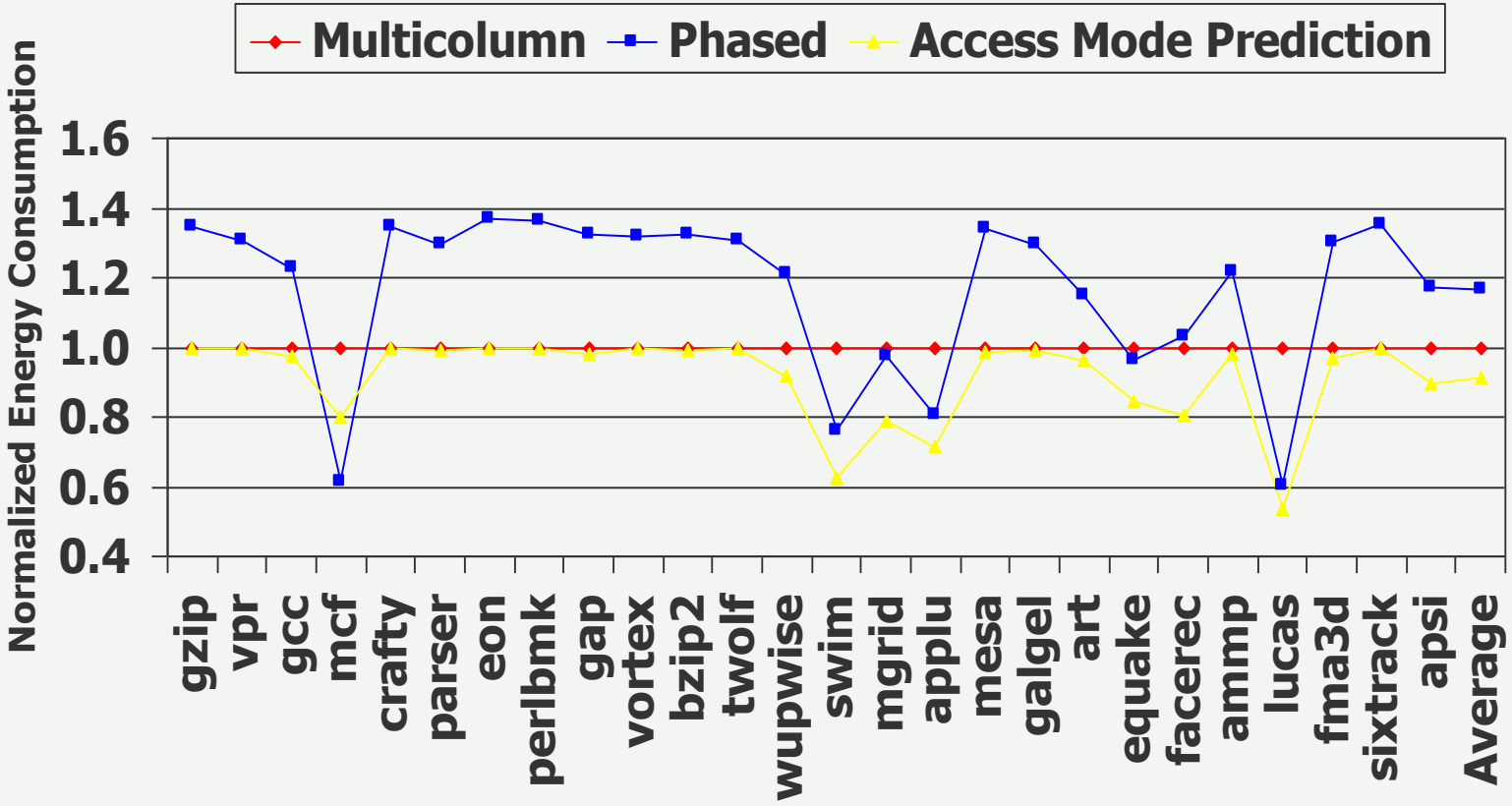
Prediction and Way Prediction

- Prediction
  - Access predictor is designed to predict next access be a hit or a miss.
  - The prediction result is used to switch between phase cache and way prediction technique.
  - Cache misses are clustered and program behavior is repetitive.
  - Branch prediction techniques are adopted.
- Way Prediction
  - Multi-column is found the most effective.

Energy Consumption: Multi-column over MRU Caches



Energy Consumption



Conclusion

- Multi-column cache fundamentally addresses the performance issue for both high hit ratio and low access time.
  - major location mapping is dominant and has the minimum access time (=direct-mapped access)
  - swap can increase the first hit ratios in major locations.
  - Indexing selected locations make non-first-hits fast.
- Multicolumn cache is also an effective way prediction mechanism for low powers.