Data Criticality in Network-On-Chip Design

Joshua San Miguel

Natalie Enright Jerger

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Network-On-Chip Efficiency

Efficiency is the ability to produce results with the least amount of waste.

- Wasted time

  NoC accounts for 30-70% of on-chip data access latency

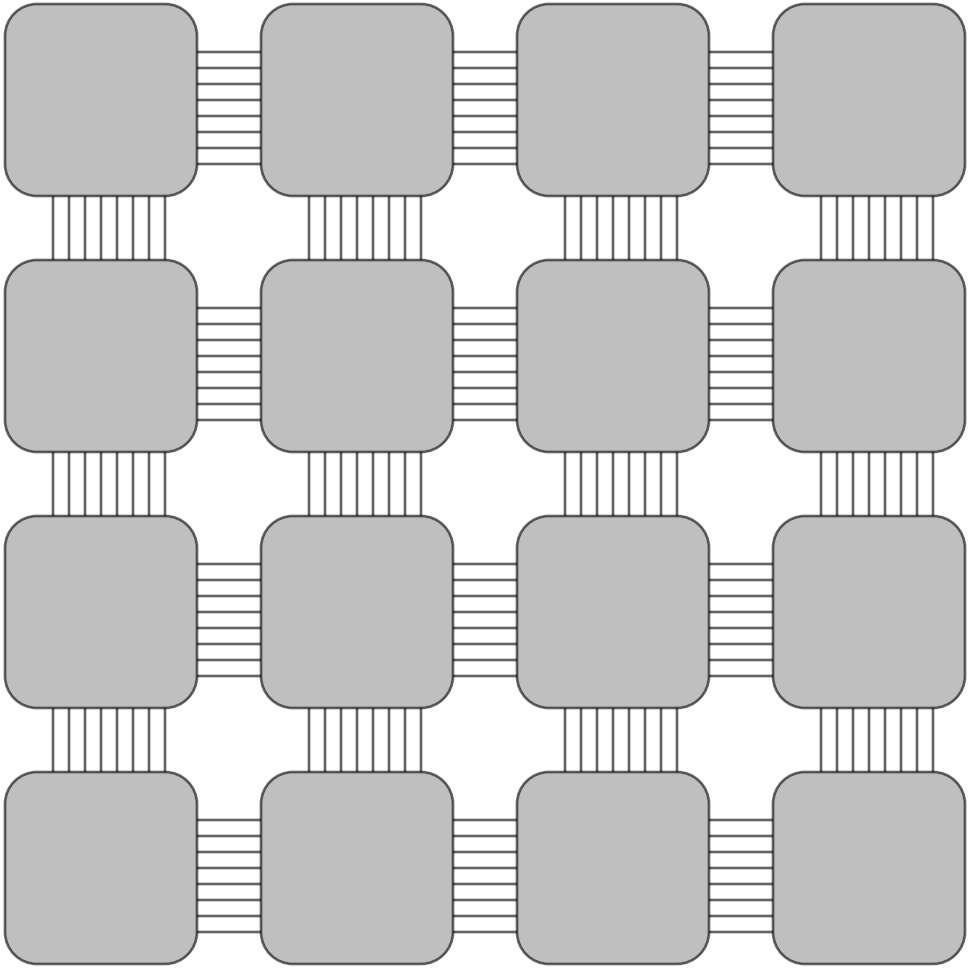
  [Z. Li, HPCA 2009][A. Sharifi, MICRO 2012]

- Wasted energy

  NoC accounts for 20-30% of total chip power

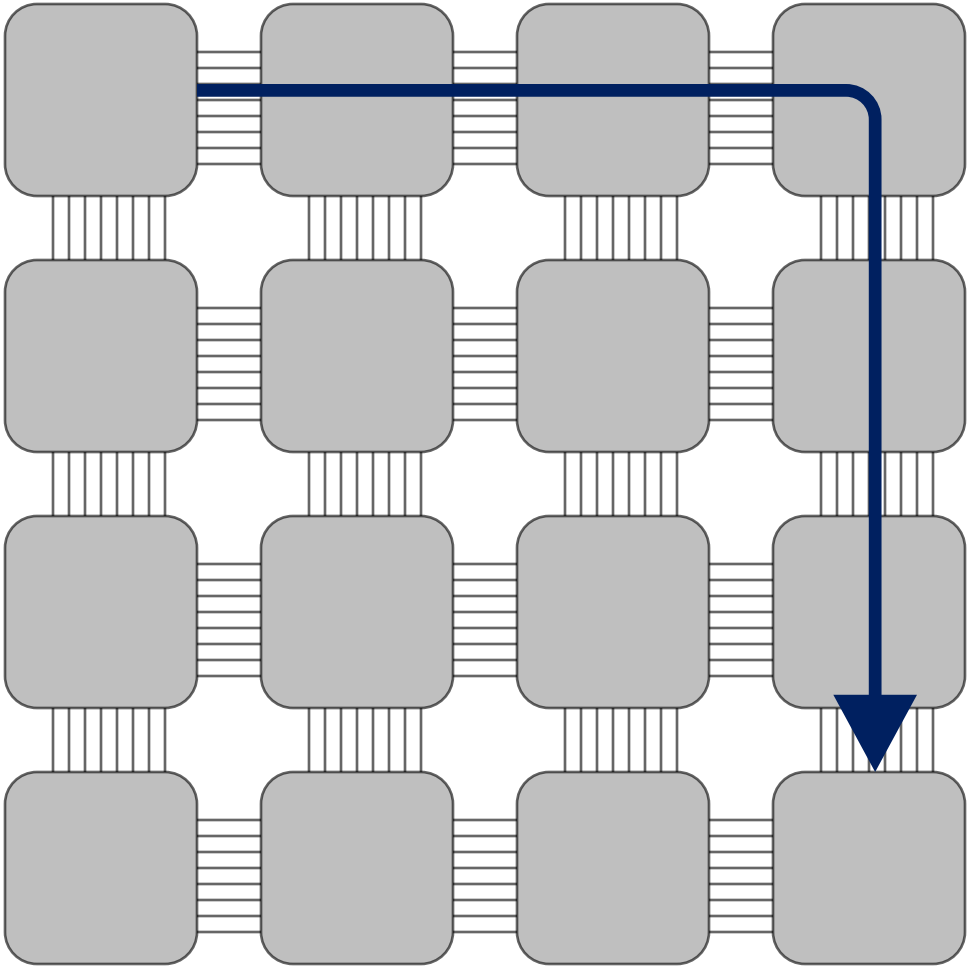
  [J. D. Owens, IEEE Micro 2007][S. R. Vangal, IEEE JSSC 2008]

Network-On-Chip Efficiency



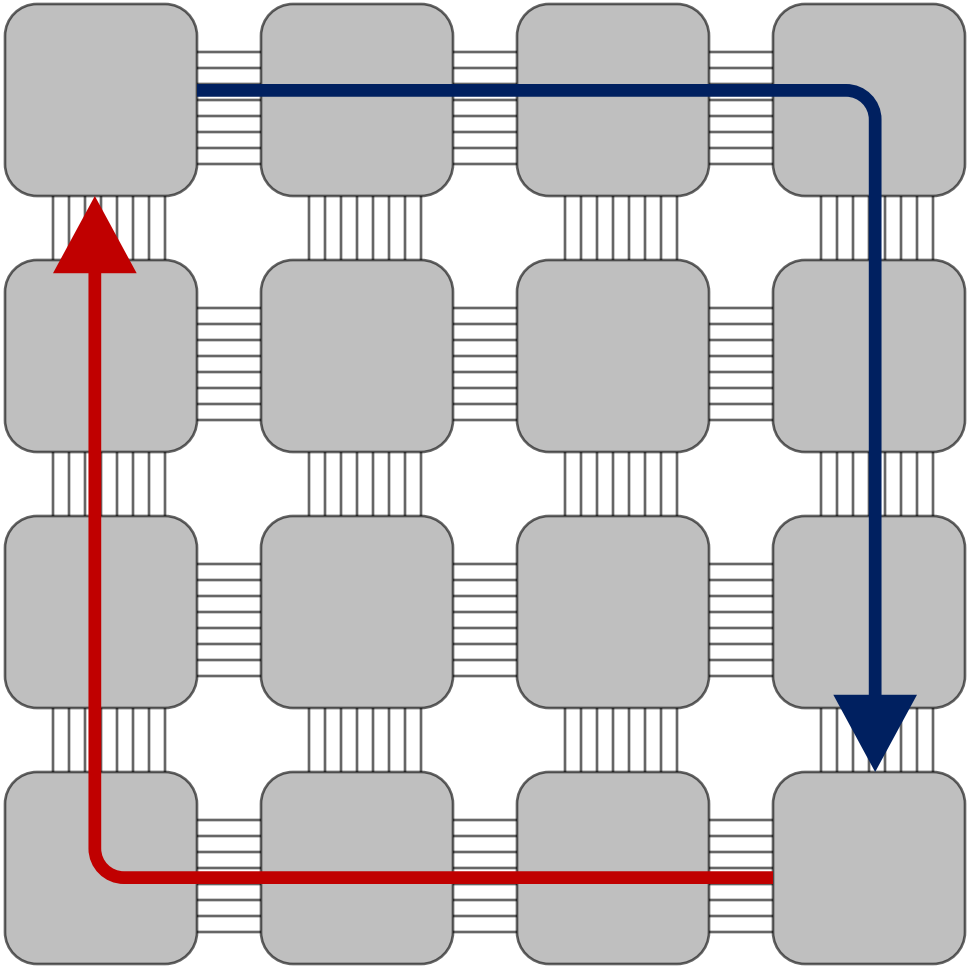
Network-On-Chip Efficiency

load request

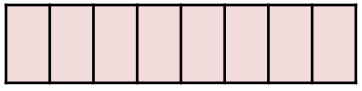


Network-On-Chip Efficiency

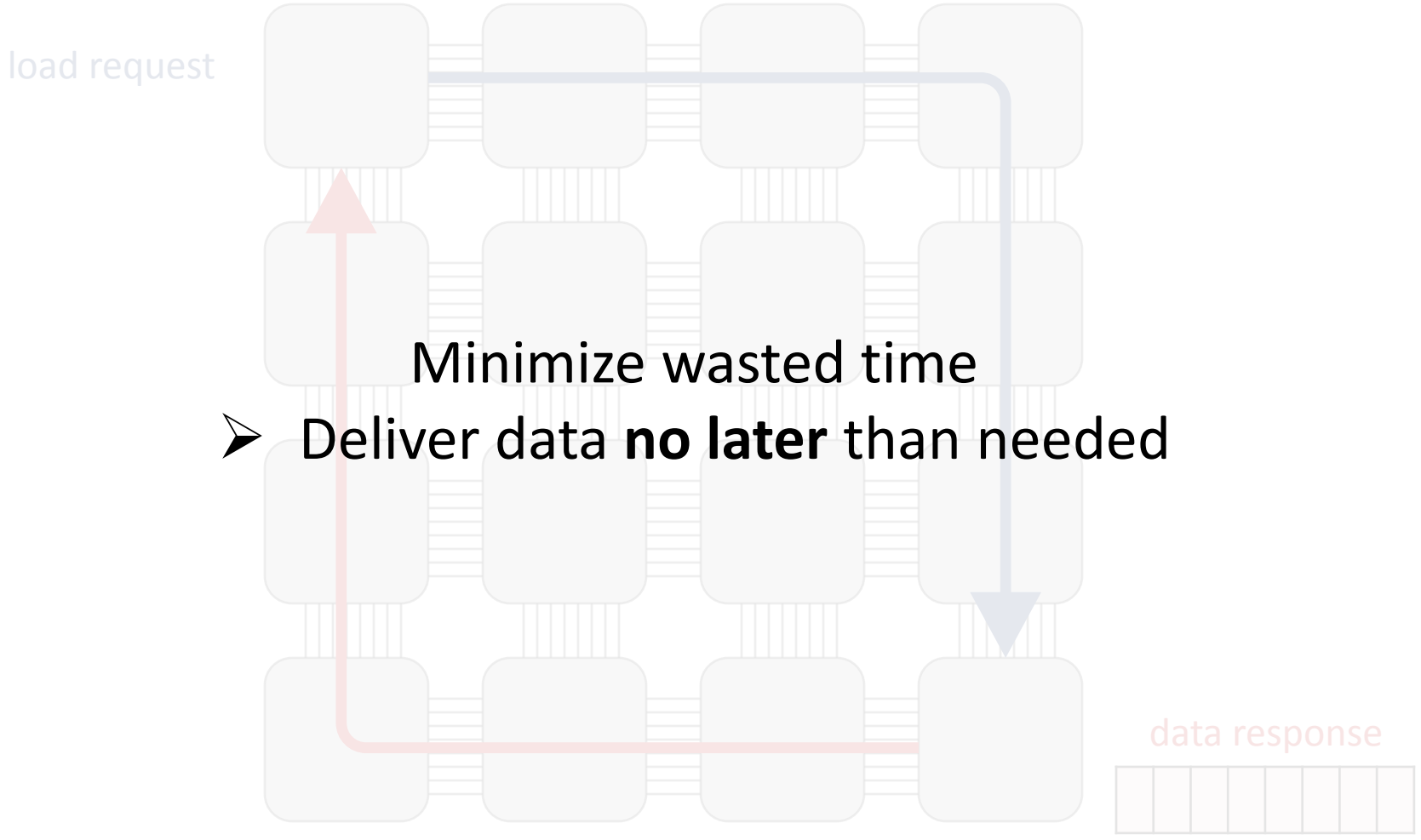
load request



data response



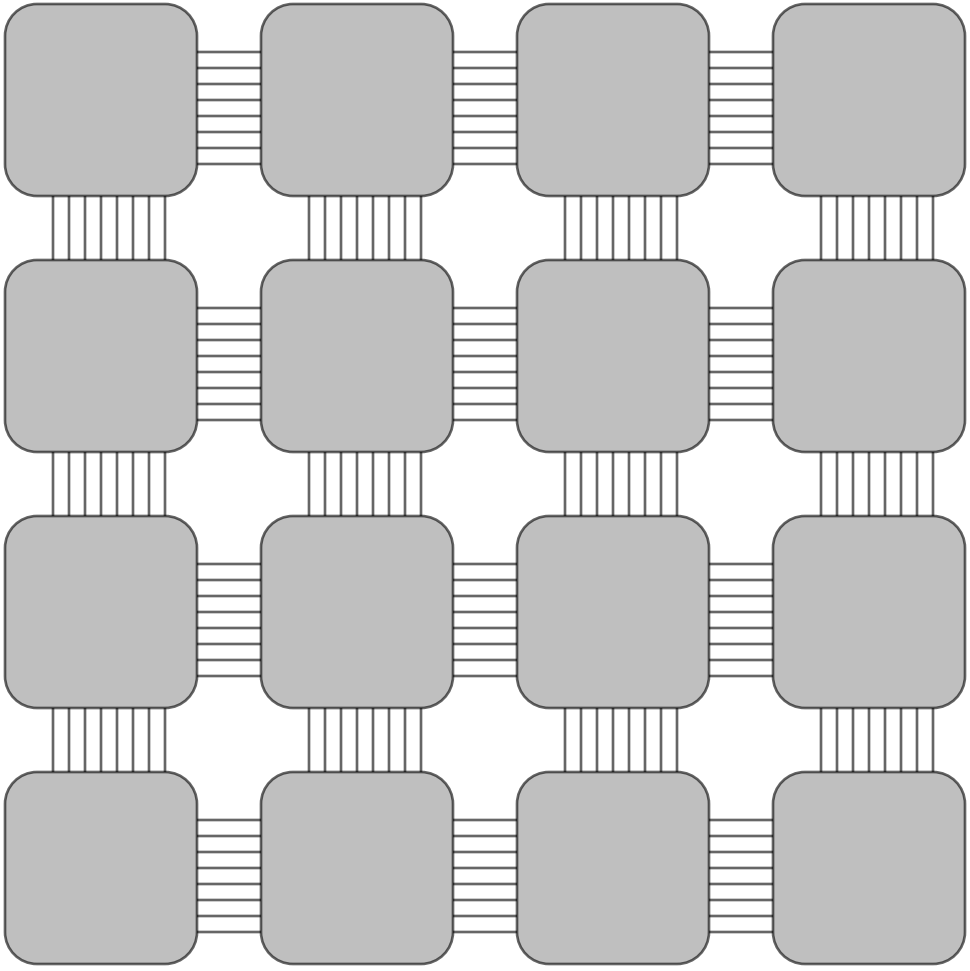
Network-On-Chip Efficiency



Minimize wasted time

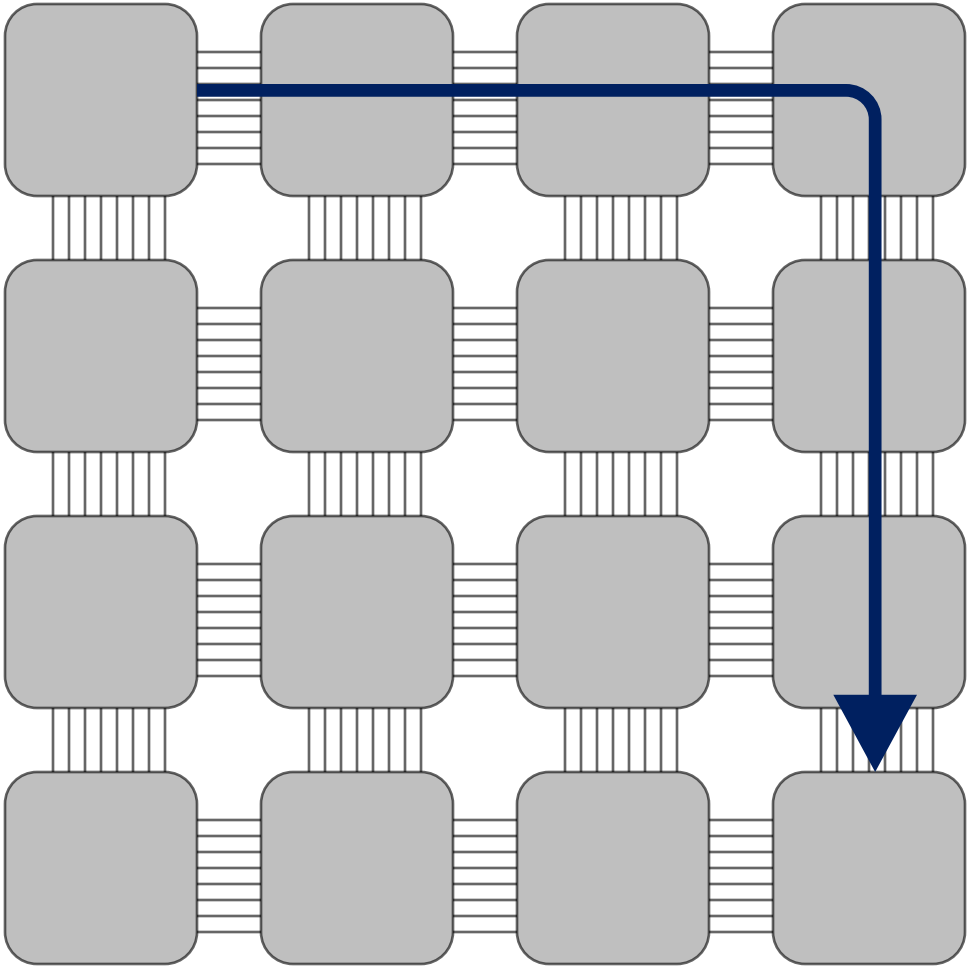
- Deliver data **no later** than needed

Network-On-Chip Efficiency



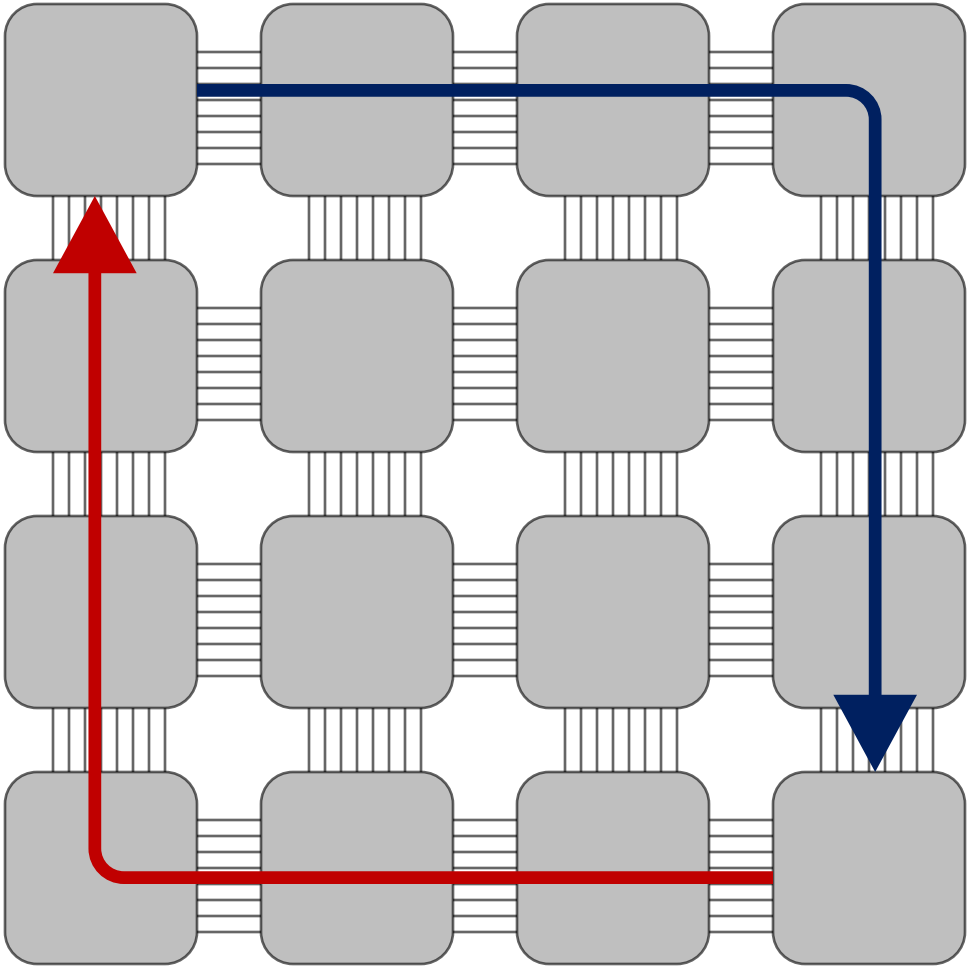
Network-On-Chip Efficiency

load request
0x2

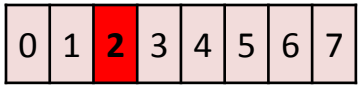


Network-On-Chip Efficiency

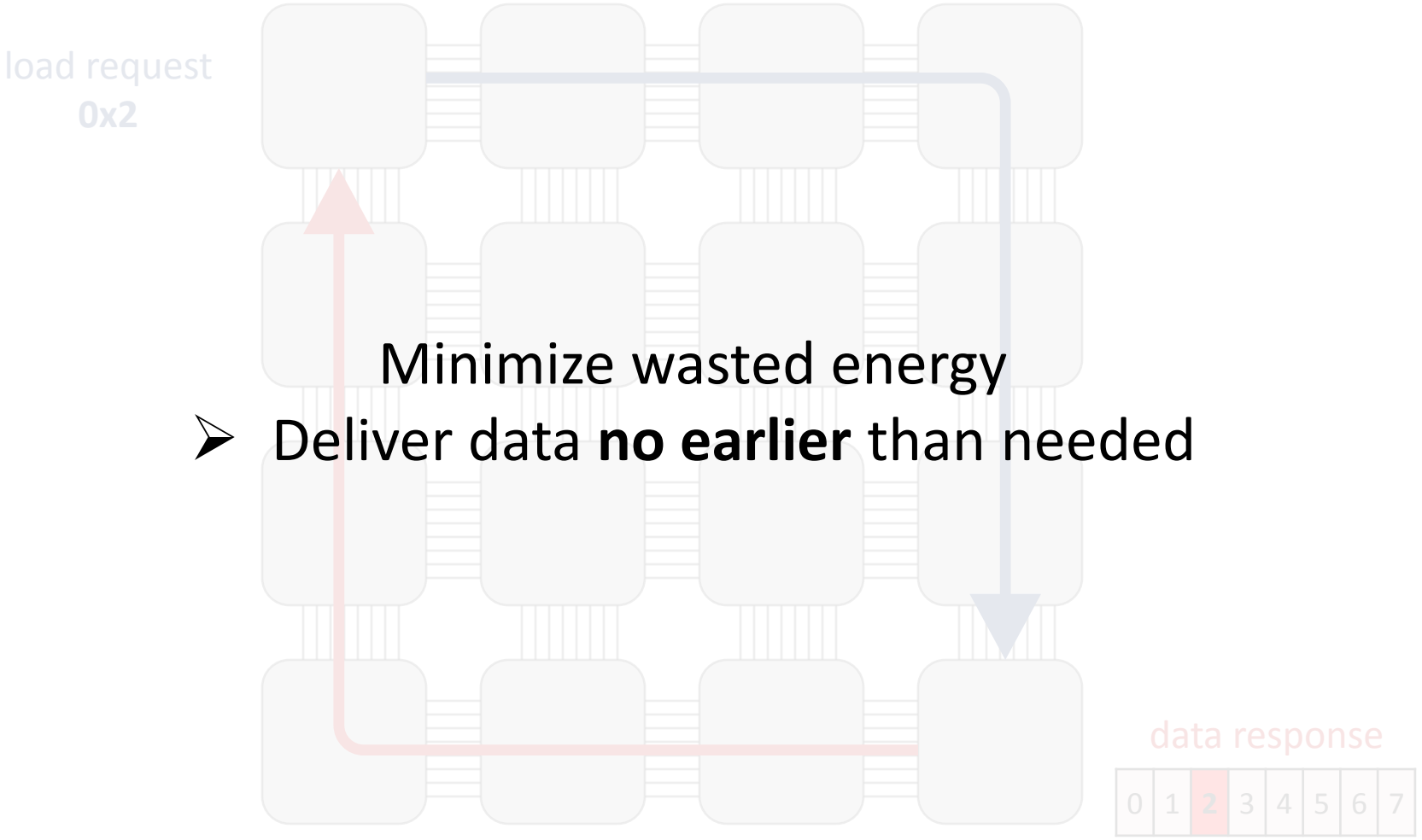
load request
0x2



data response



Network-On-Chip Efficiency



Minimize wasted energy

- Deliver data **no earlier** than needed

Network-On-Chip Efficiency

Why store data in blocks of multiple words?

- Exploit spatial locality in applications
- Avoid large tag arrays in caches
- Improve row buffer utilization in DRAM

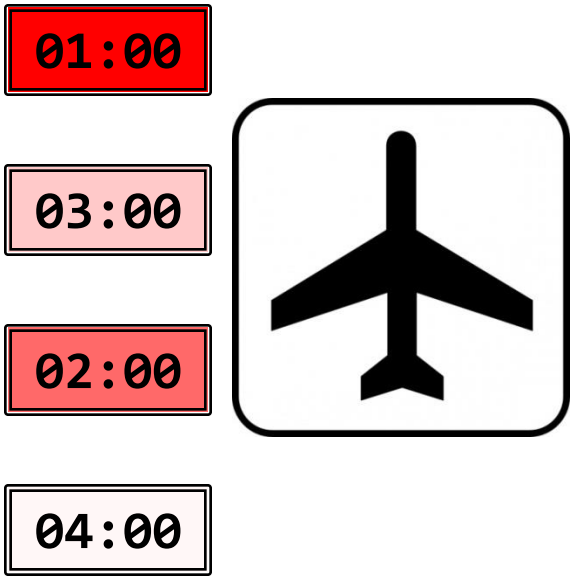
Network-On-Chip Efficiency

Why store data in blocks of multiple words?

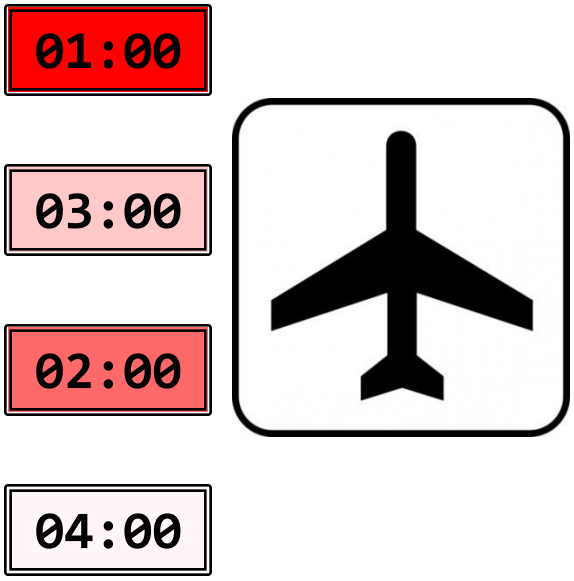
- Exploit spatial locality in applications
- Avoid large tag arrays in caches
- Improve row buffer utilization in DRAM

Store data at a coarse granularity, but move data at a fine granularity.

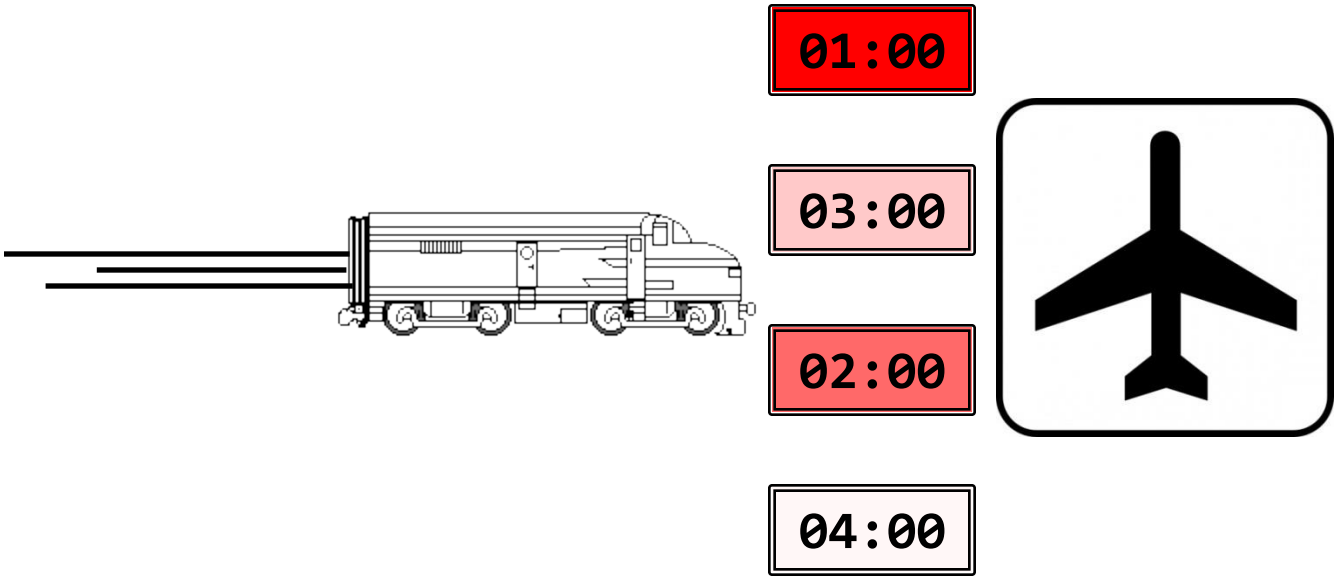
Network-On-Chip Efficiency



Network-On-Chip Efficiency

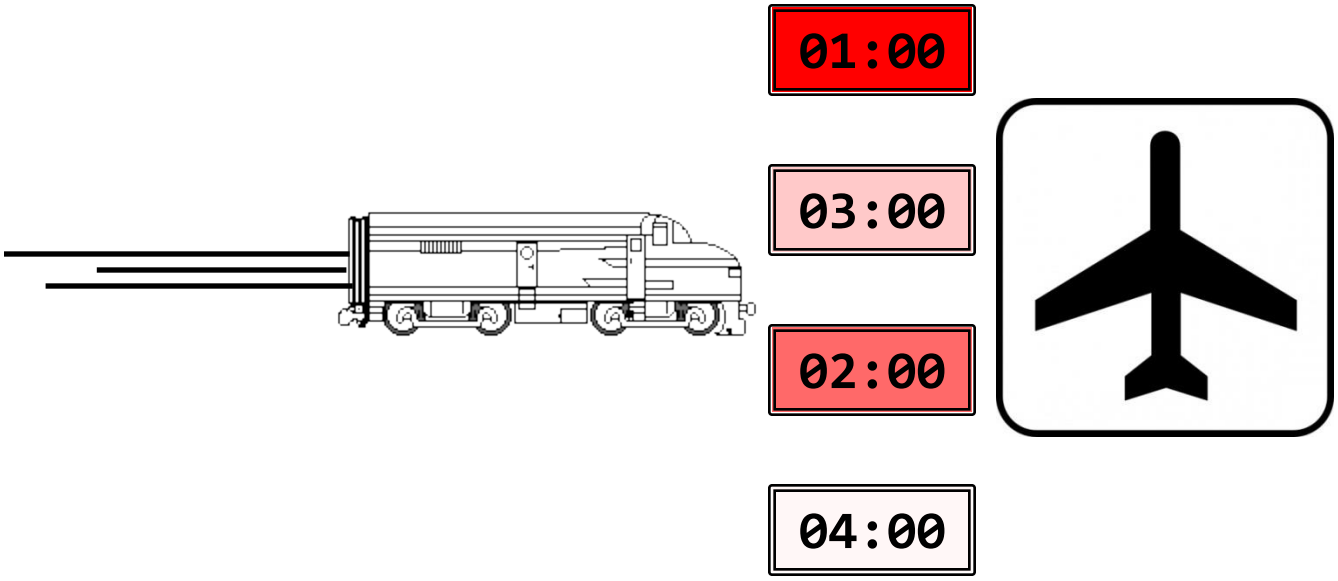


Network-On-Chip Efficiency

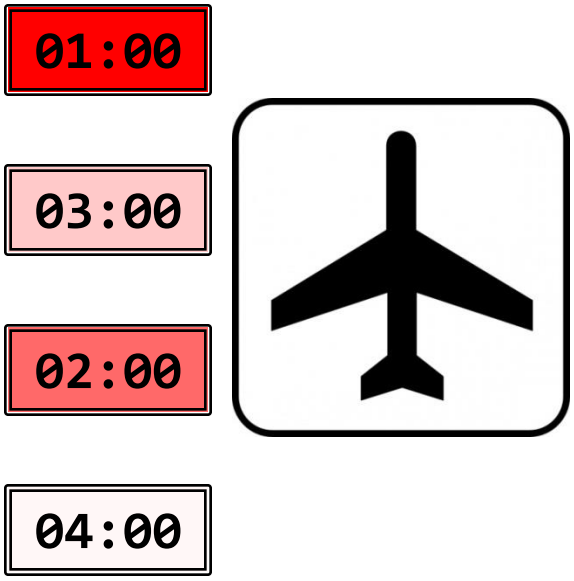


Network-On-Chip Efficiency

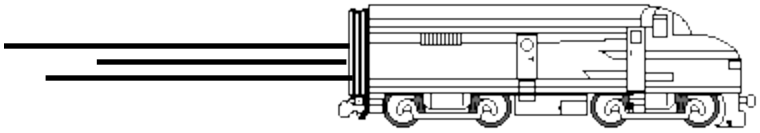
Arrive **no later** than needed. But expensive.
Wasted money since some arrive too early.



Network-On-Chip Efficiency



Network-On-Chip Efficiency



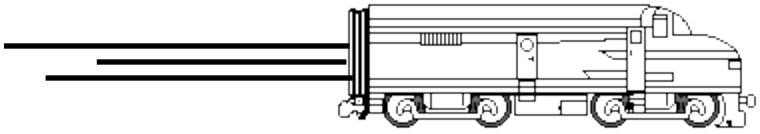
01:00

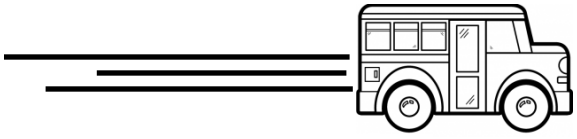
03:00

02:00

04:00

Network-On-Chip Efficiency





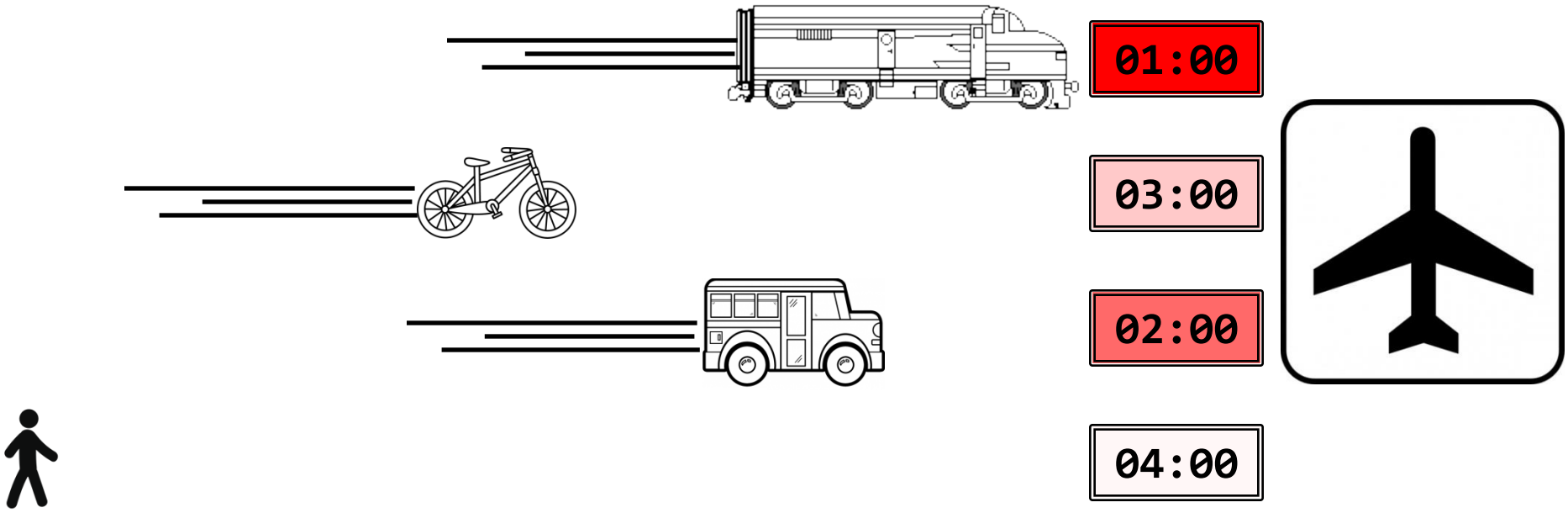
01:00

03:00

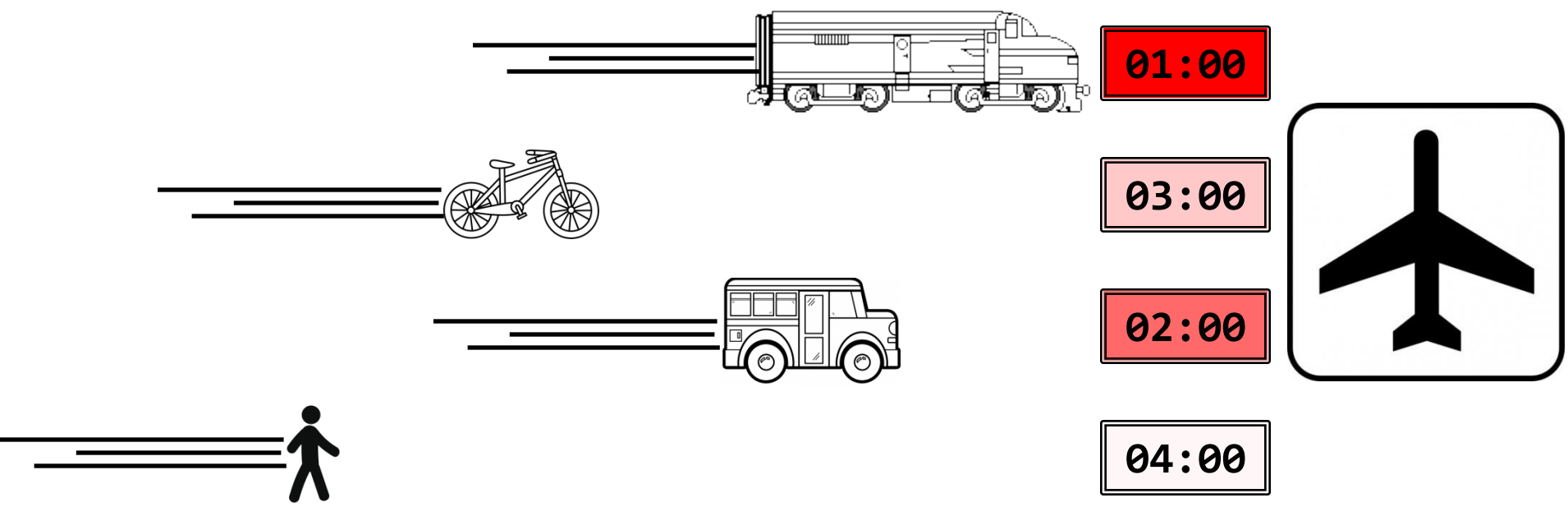
02:00

04:00

Network-On-Chip Efficiency

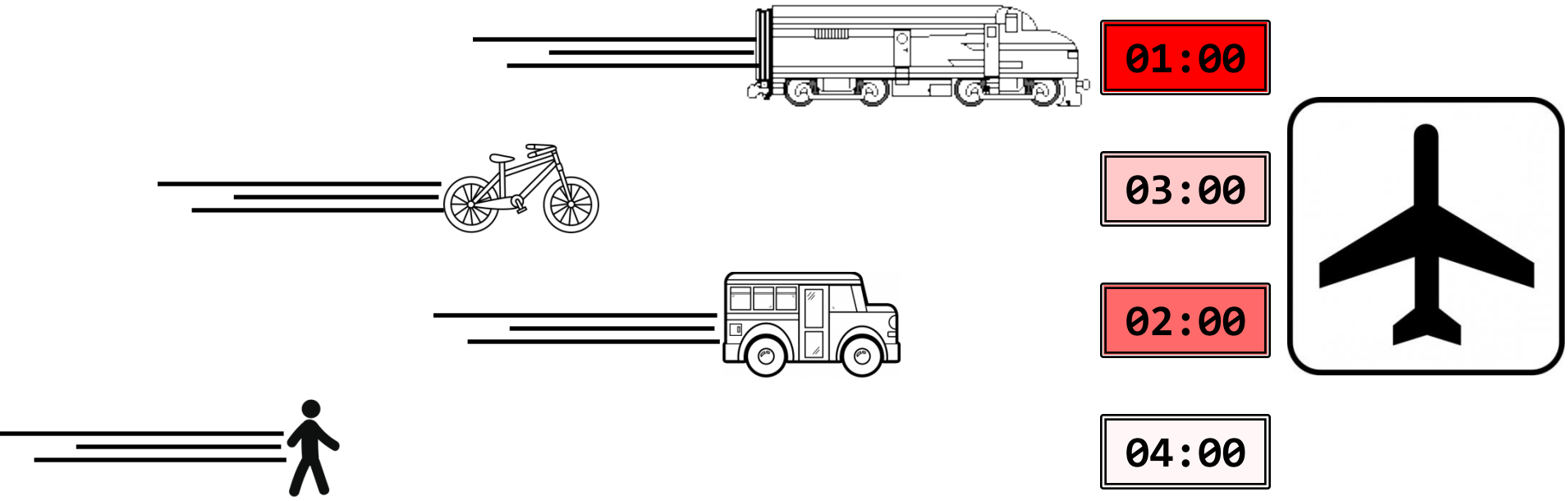


Network-On-Chip Efficiency



Network-On-Chip Efficiency

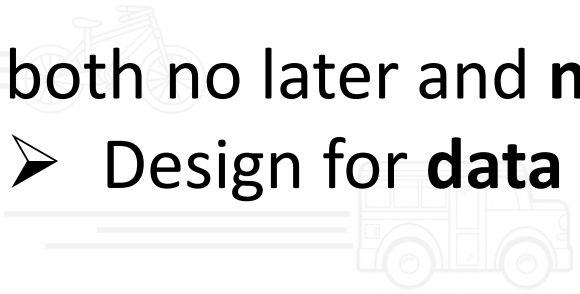
Spend just enough money to arrive
both no later and **no earlier** than needed.



Network-On-Chip Efficiency

Spend just enough money to arrive
both no later and **no earlier** than needed.



Deliver data both no later and **no earlier** than needed

- Design for **data criticality**

Outline

Defining Criticality

- Data Criticality
- Data Liveness

Measuring Criticality

- Energy Wasted

Addressing Criticality

- NoCNoC

Data Criticality

Data Criticality is the promptness with which an application uses a data word after fetching it from memory.

Critical: used immediately after being fetched.

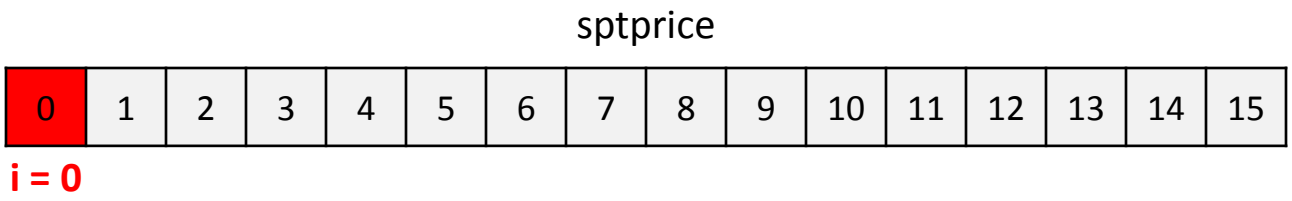
Non-critical: used some time later after being fetched.

Data Criticality – blackscholes

```
for (i++) {
        ... = BlkSchlsEqEuroNoDiv(sptprice[i]);
}
```

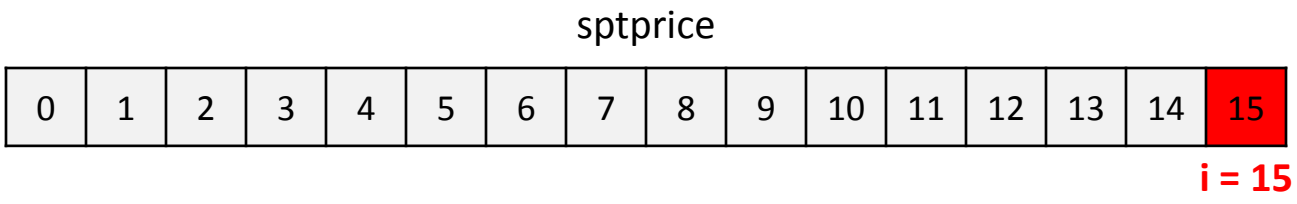
Data Criticality – blackscholes

```
for (i++) {
        ... = BlkSchlsEqEuroNoDiv(sptprice[i]);
}
```



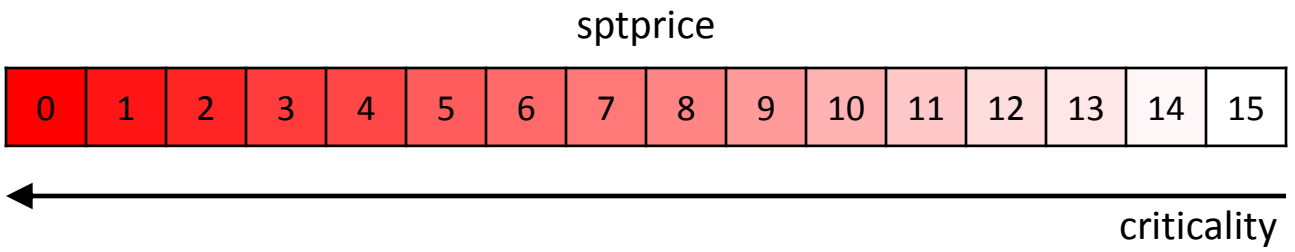
Data Criticality – blackscholes

```
for (i++) {
        ... = BlkSchlsEqEuroNoDiv(sptprice[i]);
}
```



Data Criticality – blackscholes

```
for (i++) {
        ... = BlkSchlsEqEuroNoDiv(sptprice[i]);
}
```



Data Criticality – fluidanimate

```
for (iparNeigh++) {
      if (borderNeigh) {
            pthread_mutex_lock();
            neigh->a[iparNeigh] -= ...;
            pthread_mutex_unlock();
      } else {
            neigh->a[iparNeigh] -= ...;
      }
}
```

Data Criticality

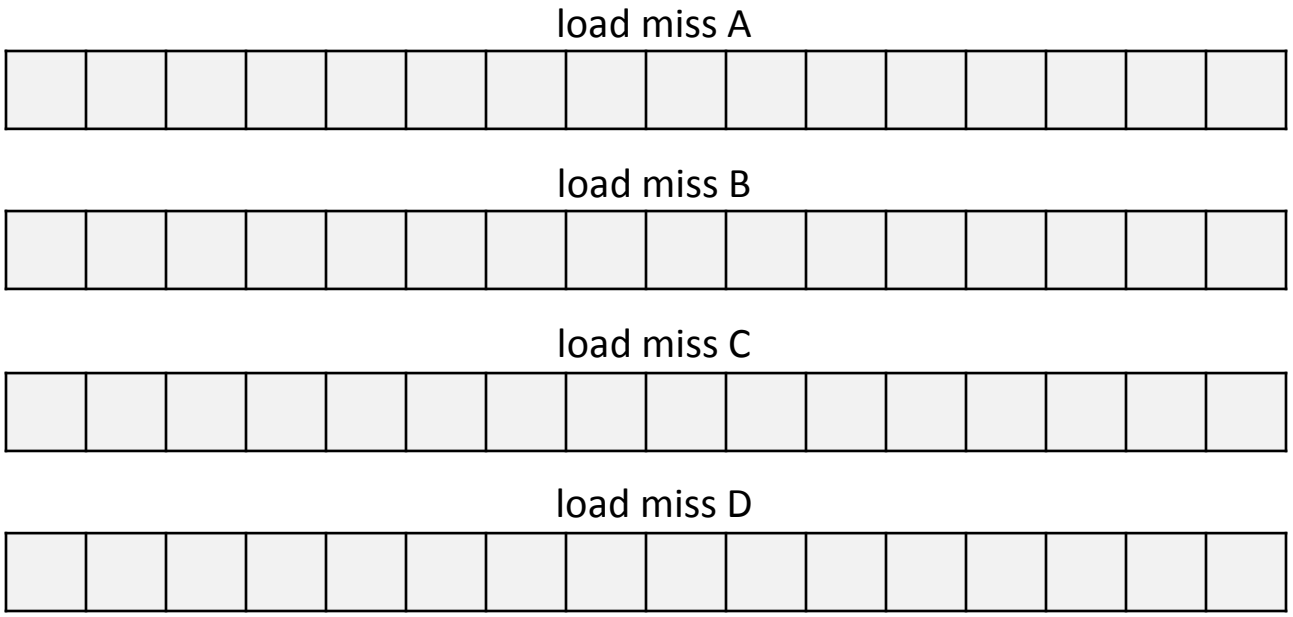
Data criticality is an inherent consequence of spatial locality and is exhibited by most (if not all) real-world applications.

Examples of non-criticality:

- Long-running code between accesses
- Interference due to thread synchronization
- Dependences from other cache misses
- Preemption by the operating system

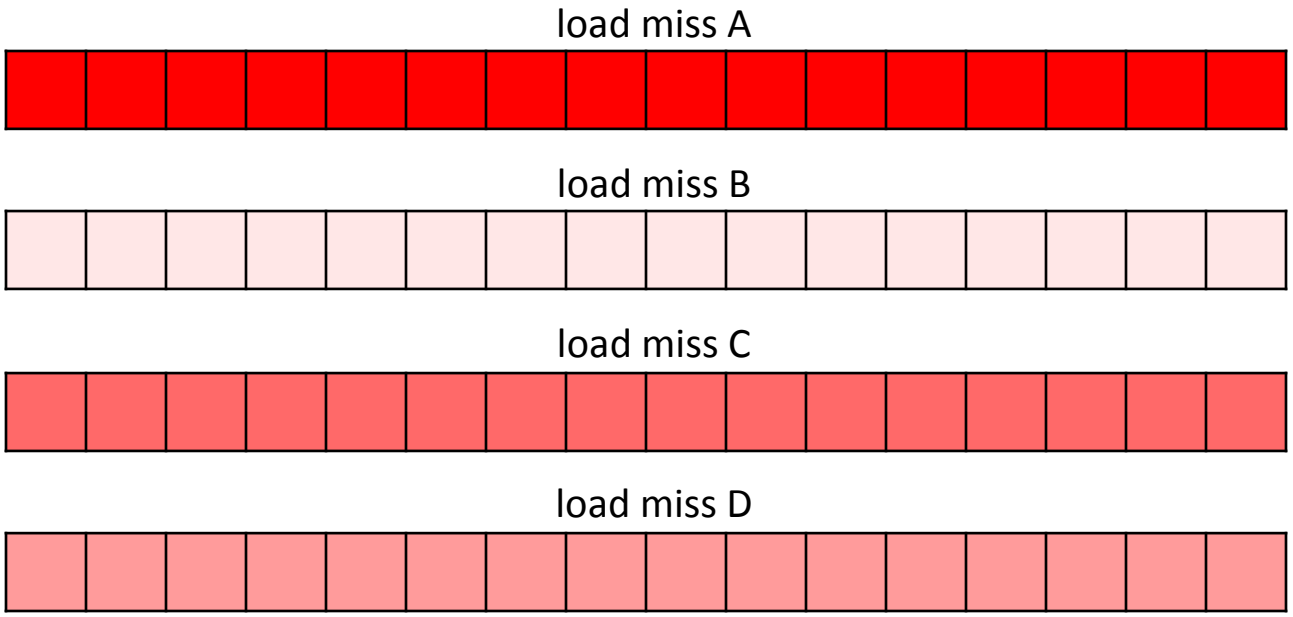
Data Criticality vs. Instruction Criticality

Instruction (or packet) criticality

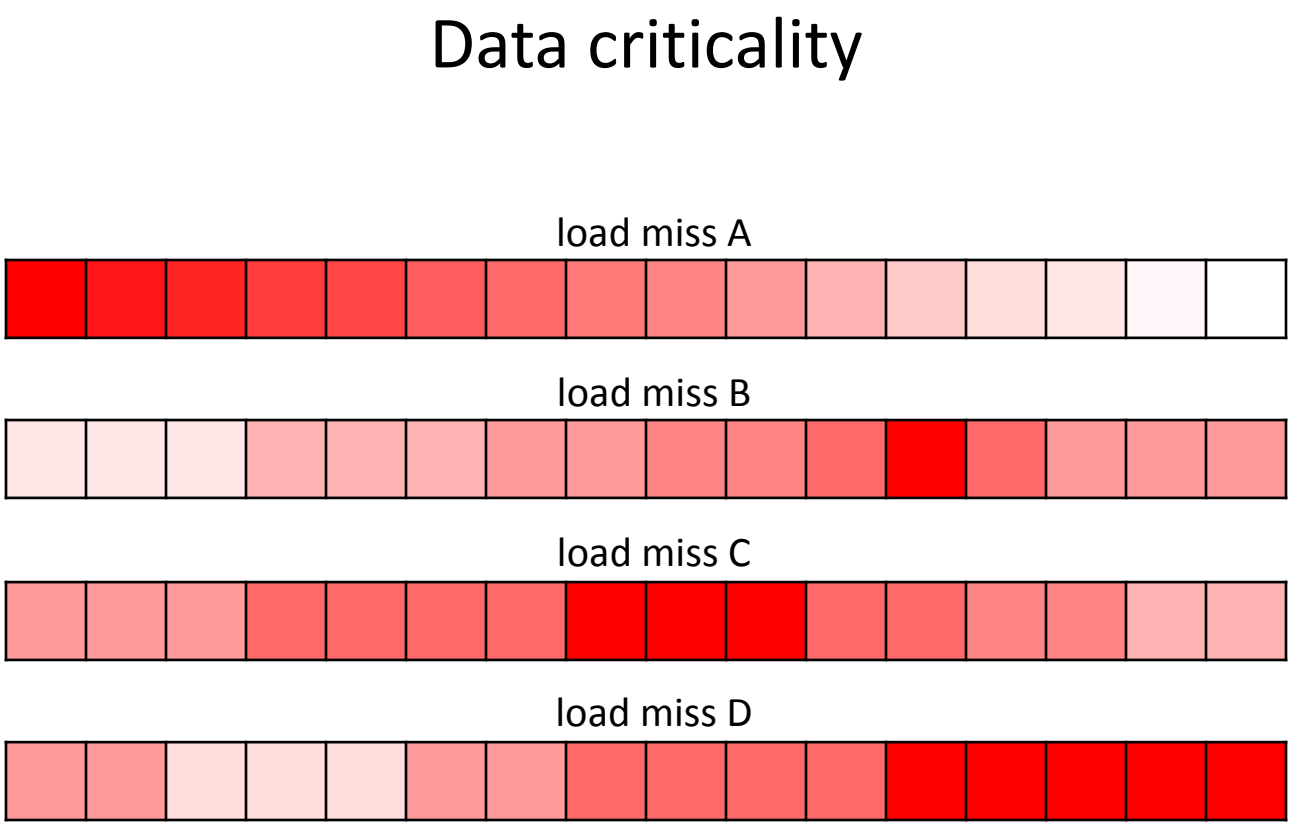


Data Criticality vs. Instruction Criticality

Instruction (or packet) criticality



Data Criticality vs. Instruction Criticality



Data Liveness

Data Liveness describes whether or not an application uses a data word at all after fetching it from memory.

Live-on-arrival (live): used at least once during its cache lifetime.

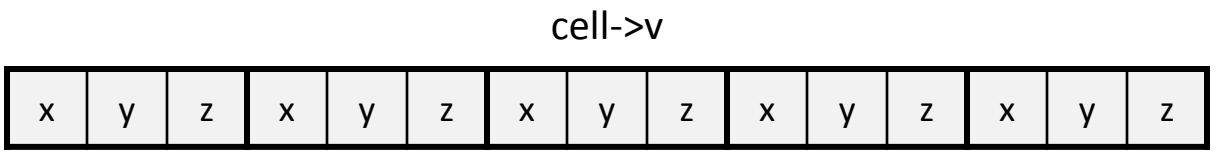
Dead-on-arrival (dead): never used during its cache lifetime.

Data Liveness – fluidanimate

```
for (iz++)
for (iy++)
for (ix++)
for (j++) {
      if (border(ix)) ... = cell->v[j].x;
      if (border(iy)) ... = cell->v[j].y;
      if (border(iz)) ... = cell->v[j].z;
}
```

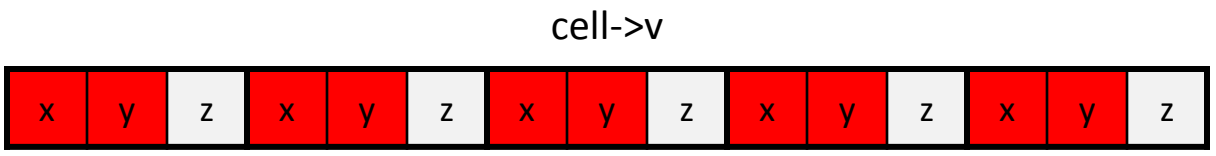
Data Liveness – fluidanimate

```
for (iz++)
for (iy++)
for (ix++)
for (j++) {
      if (border(ix)) ... = cell->v[j].x;
      if (border(iy)) ... = cell->v[j].y;
      if (border(iz)) ... = cell->v[j].z;
}
```



Data Liveness – fluidanimate

```
for (iz++)
for (iy++)
for (ix++)
for (j++) {
      if (border(ix)) ... = cell->v[j].x;
      if (border(iy)) ... = cell->v[j].y;
      if (border(iz)) ... = cell->v[j].z;
}
```



Data Liveness

Data liveness measures the degree of spatial locality in an application.

Examples of dead words:

- Unused members of structs
- Irregular or random access patterns
- Heap fragmentation
- Padding between data elements
- Early evictions due to invalidations, cache pressure or poor replacement policies

Outline

Defining Criticality

- Data Criticality
- Data Liveness

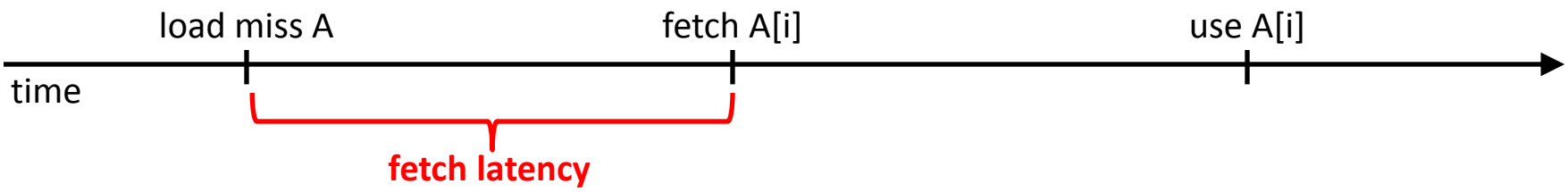
Measuring Criticality

- **Energy Wasted**

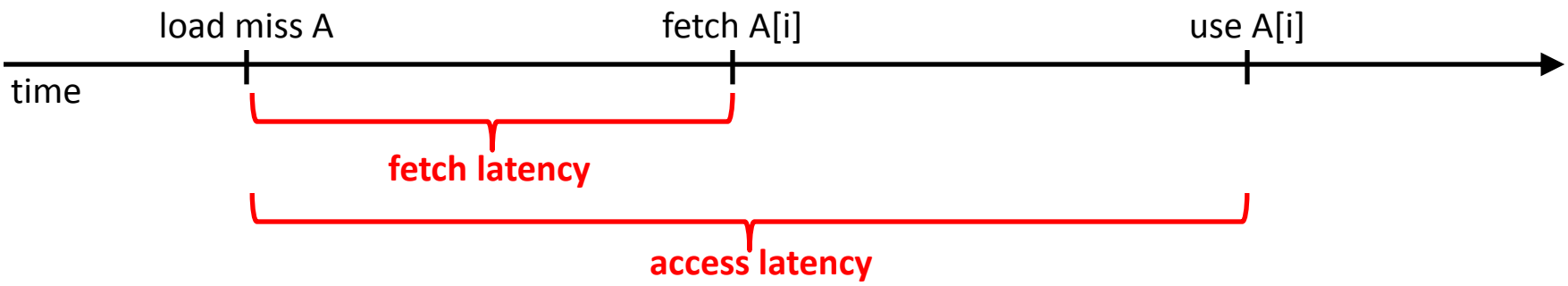
Addressing Criticality

- NoCNoC

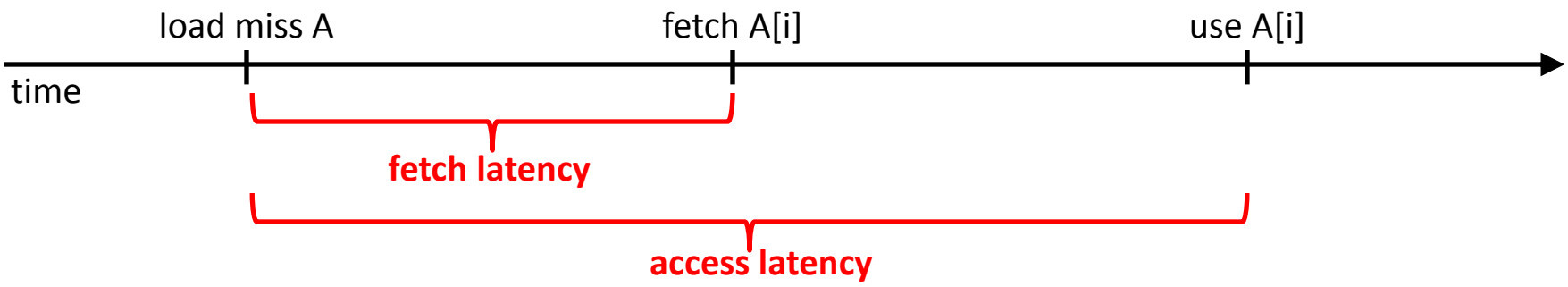
Measuring Criticality



Measuring Criticality



Measuring Criticality



$$non\text{–}criticality = \frac{access\ latency}{fetch\ latency}$$

1x for critical words, >1x for non-critical words

Measuring Criticality

Full-system simulations:

- FeS2, BookSim, DSENT
- 16 2.0 GHz OoO cores
- 64 kB private L1 per core, 16-word cache blocks
- 16 MB shared distributed L2

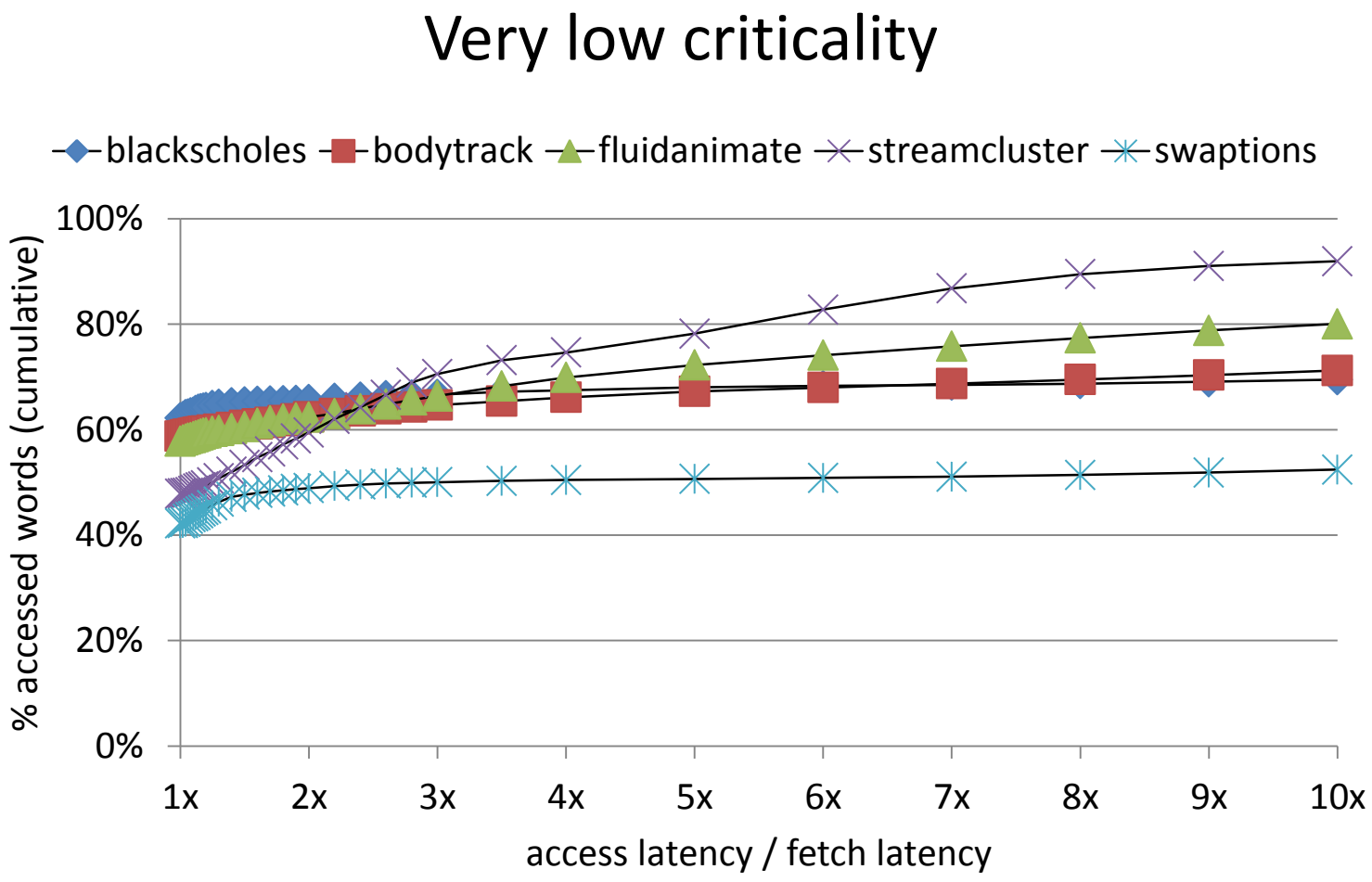
Baseline NoC configuration:

- 4 x 4 mesh, 2.0 GHz, 128-bit channels
- X-Y routing, 3-stage router pipeline, 6 4-flit VCs per port

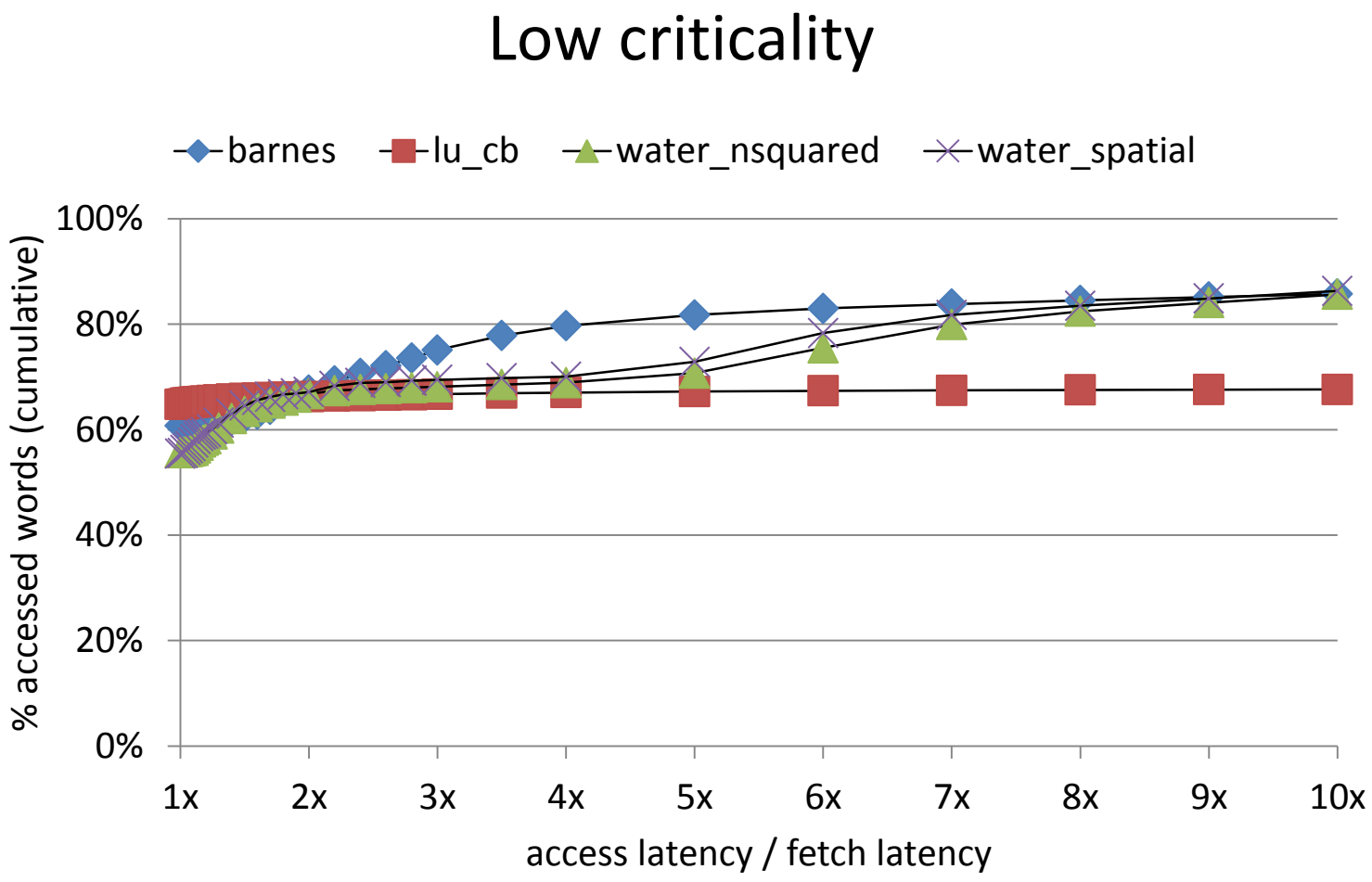
Applications:

- PARSEC and SPLASH-2

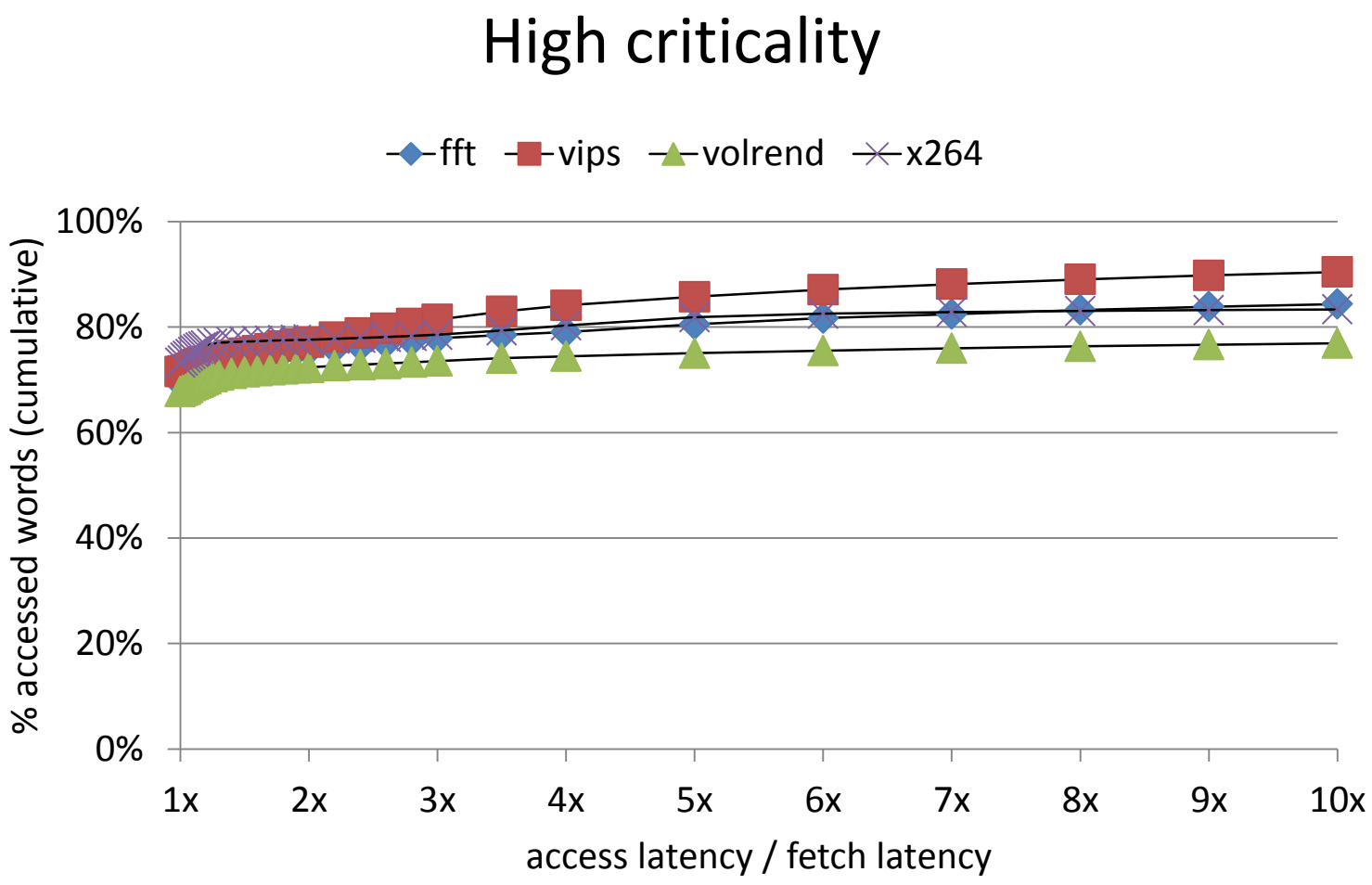
Measuring Criticality



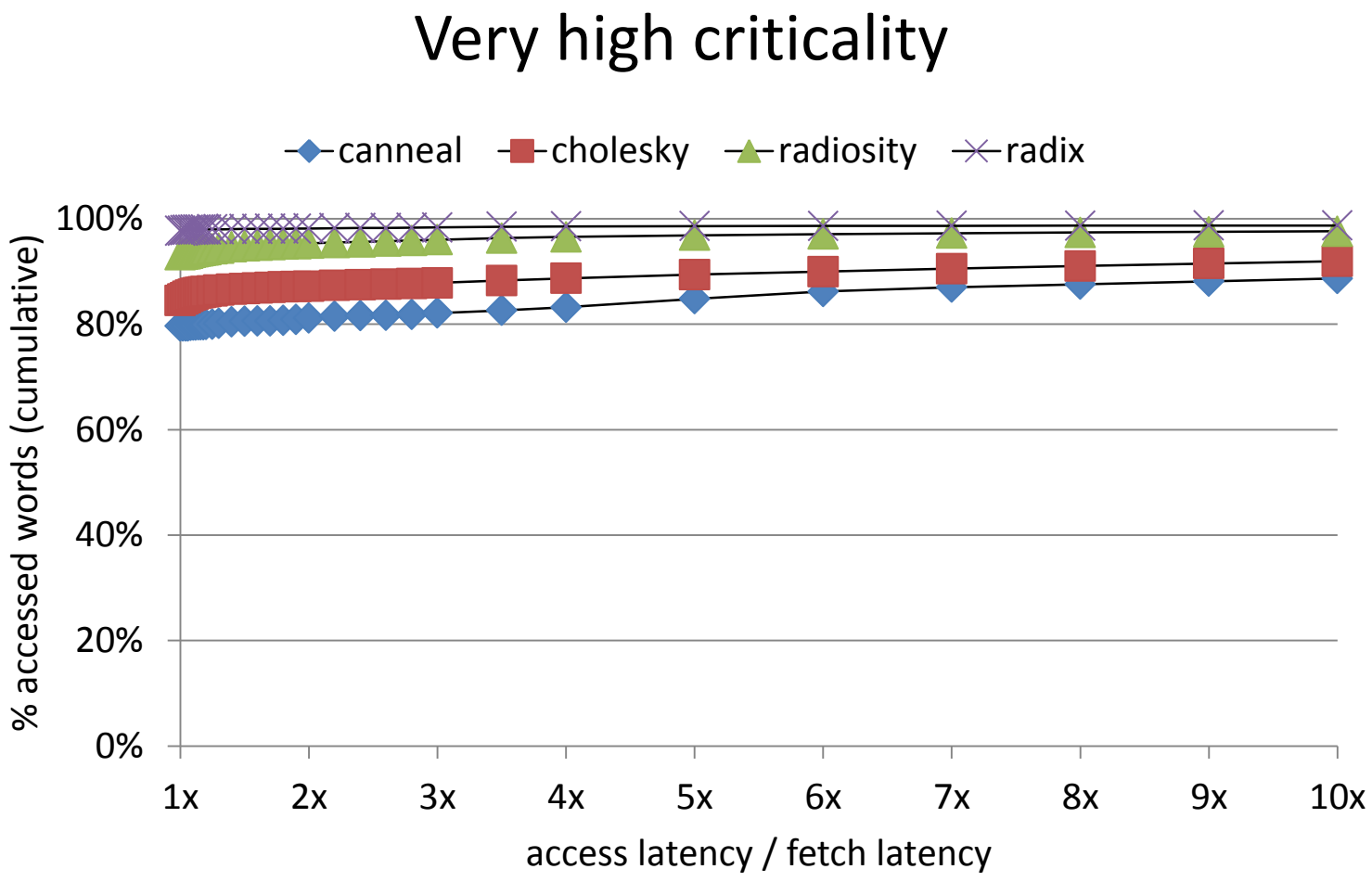
Measuring Criticality



Measuring Criticality



Measuring Criticality



Measuring Criticality – Energy Wasted

Estimate energy wasted due to non-criticality

- Model an ideal NoC where for each word:

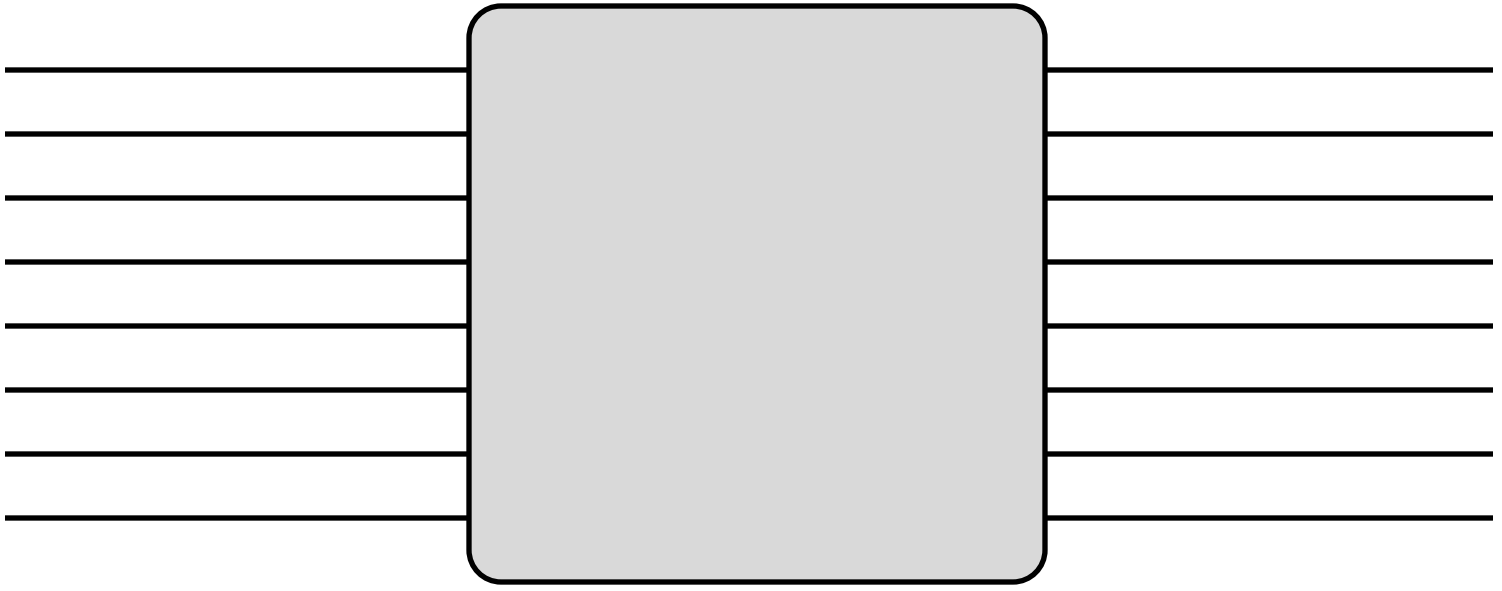
$$fetch\ latency \approx access\ latency$$

Measuring Criticality – Energy Wasted

Estimate energy wasted due to non-criticality

- Model an ideal NoC where for each word:

$$fetch\ latency \approx access\ latency$$

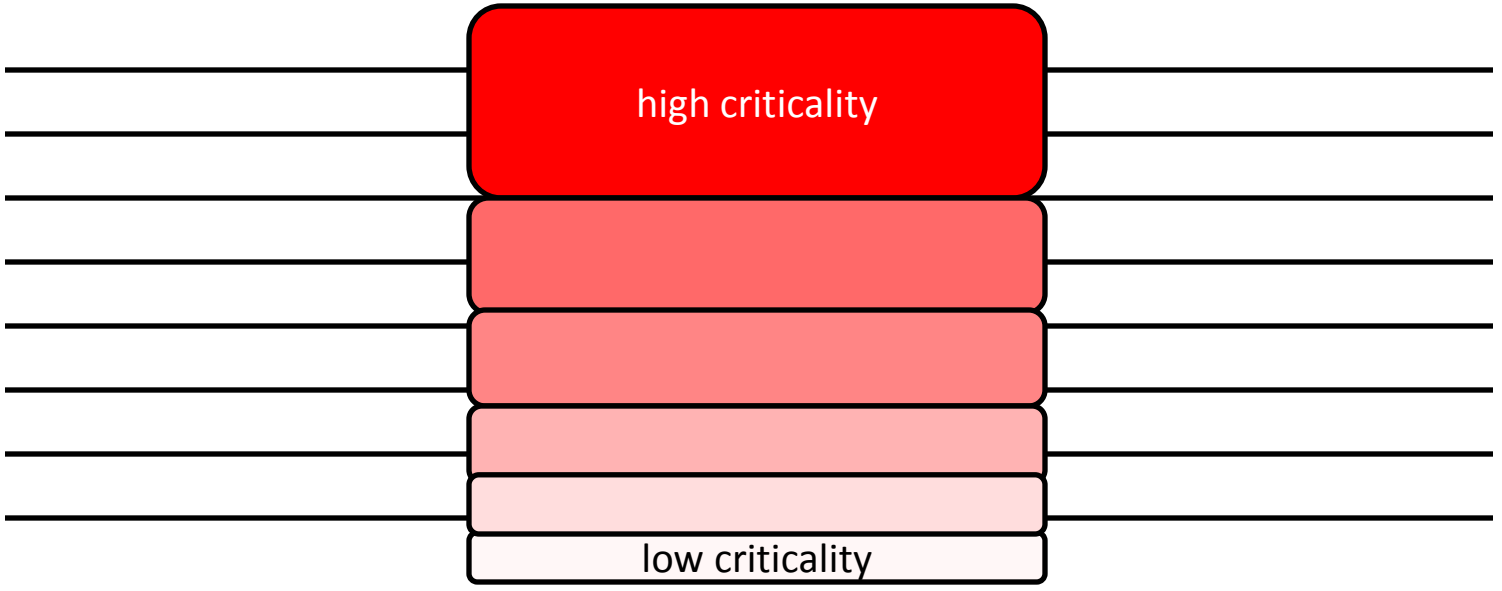


Measuring Criticality – Energy Wasted

Estimate energy wasted due to non-criticality

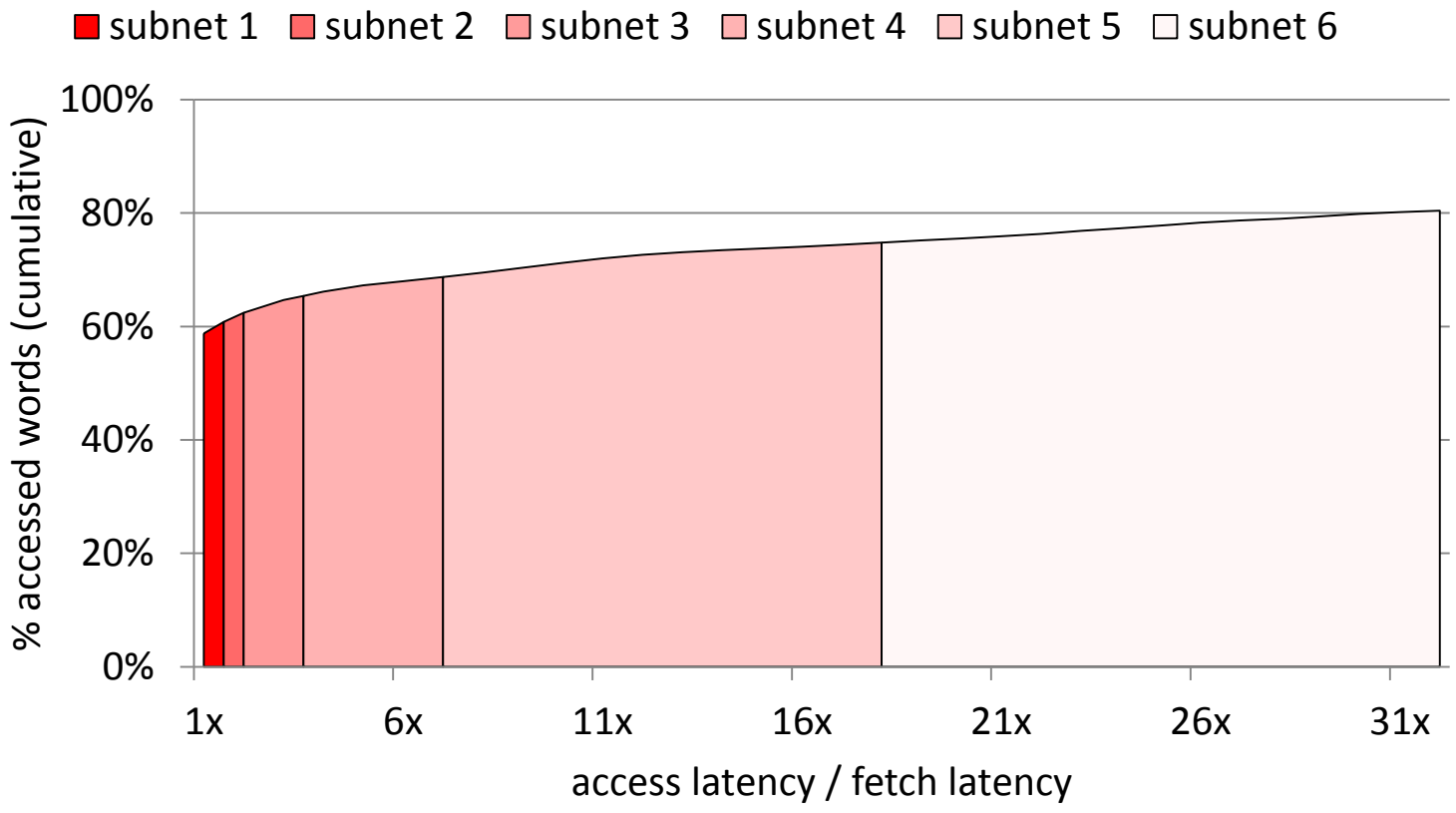
- Model an ideal NoC where for each word:

$$fetch\ latency \approx access\ latency$$



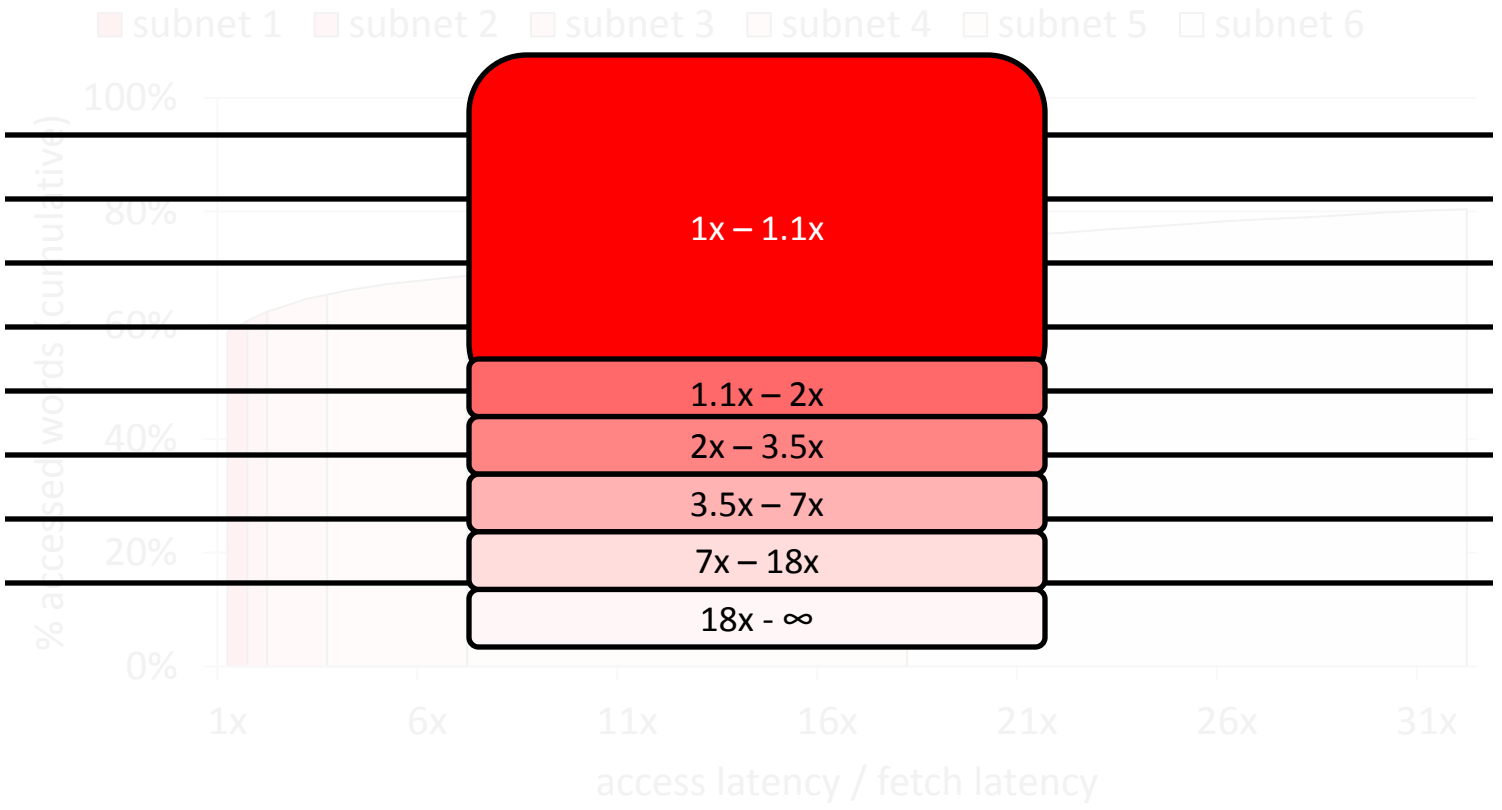
Measuring Criticality – Energy Wasted

e.g., bodytrack:



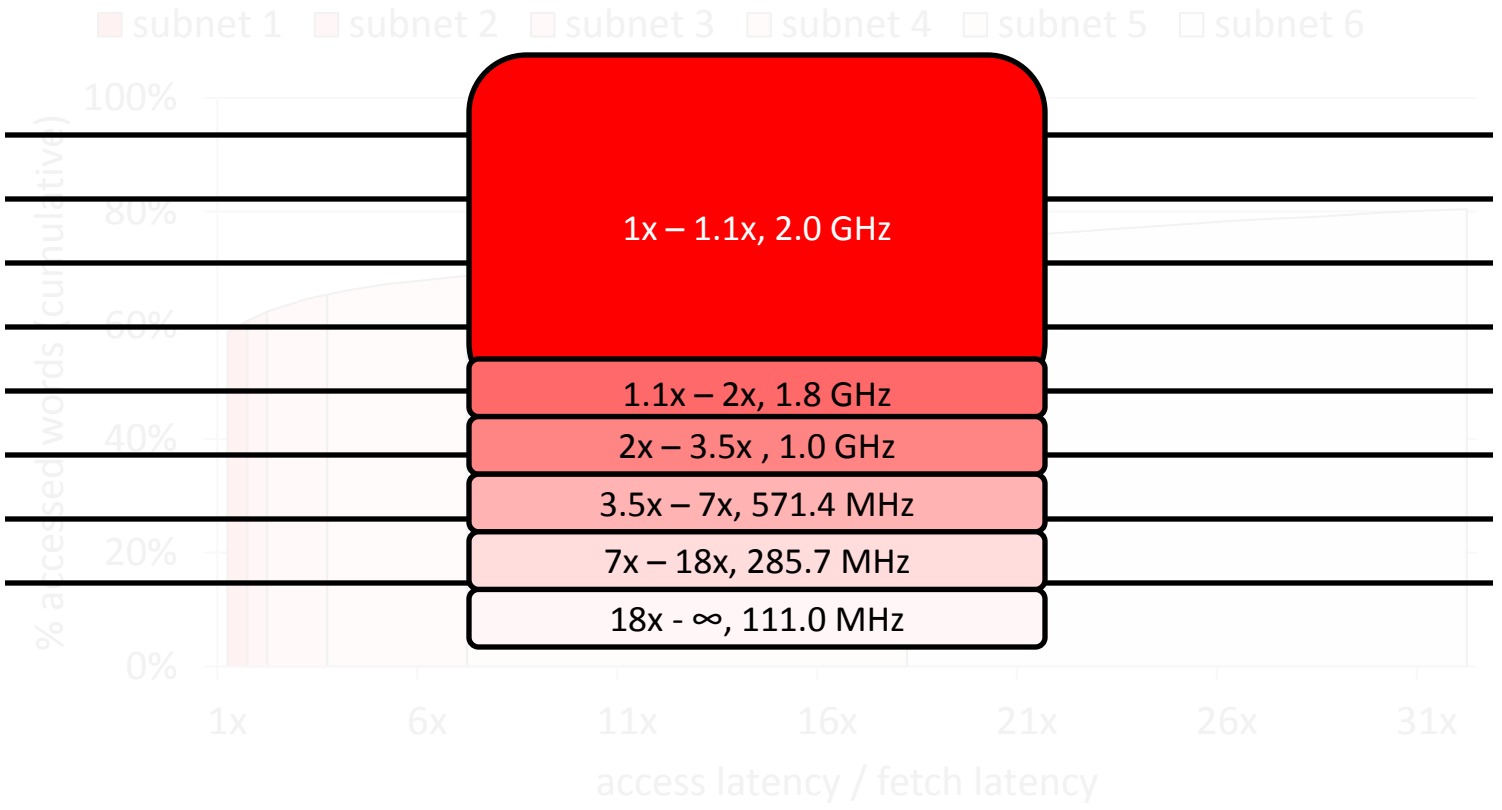
Measuring Criticality – Energy Wasted

e.g., bodytrack:

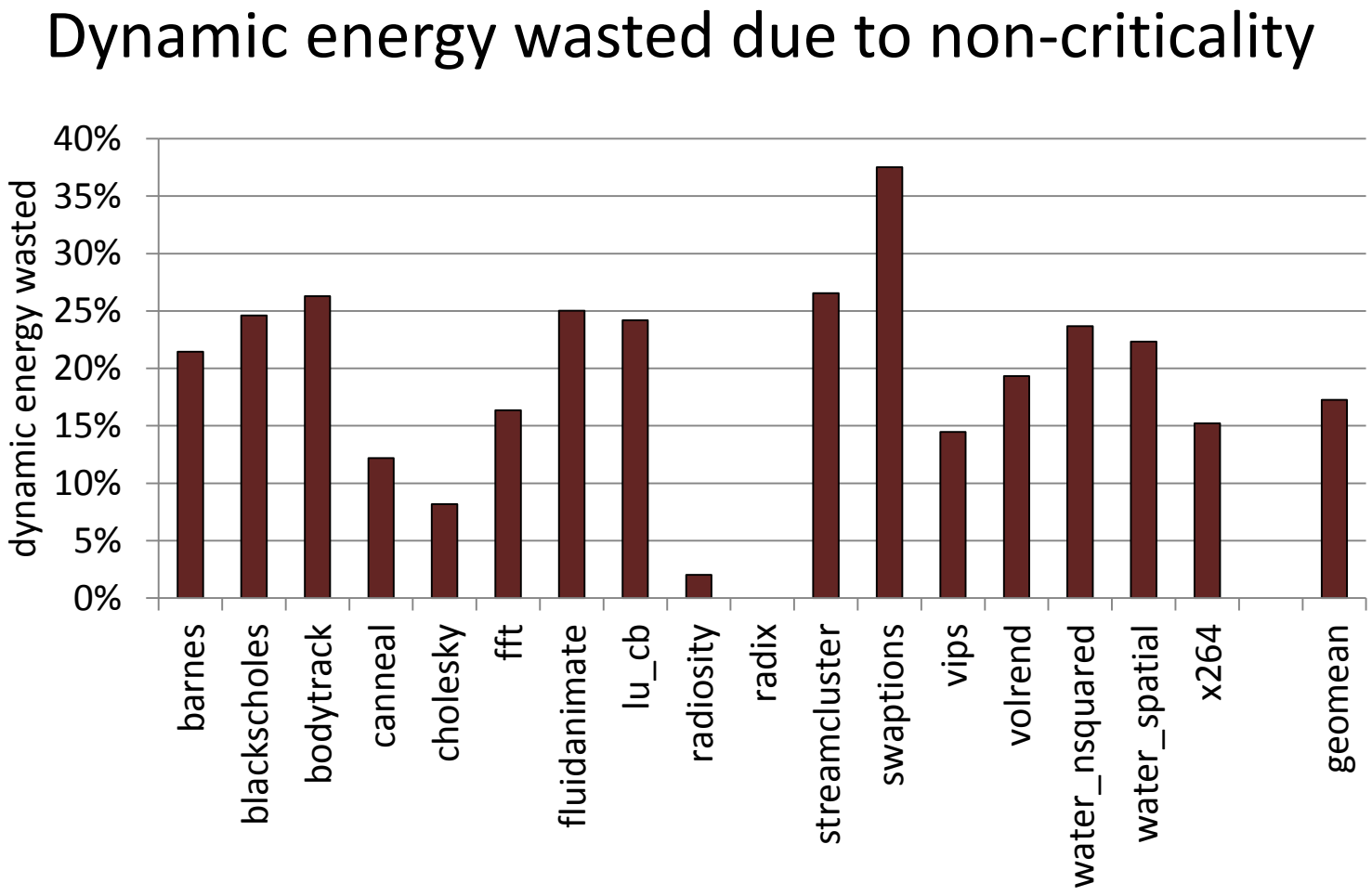


Measuring Criticality – Energy Wasted

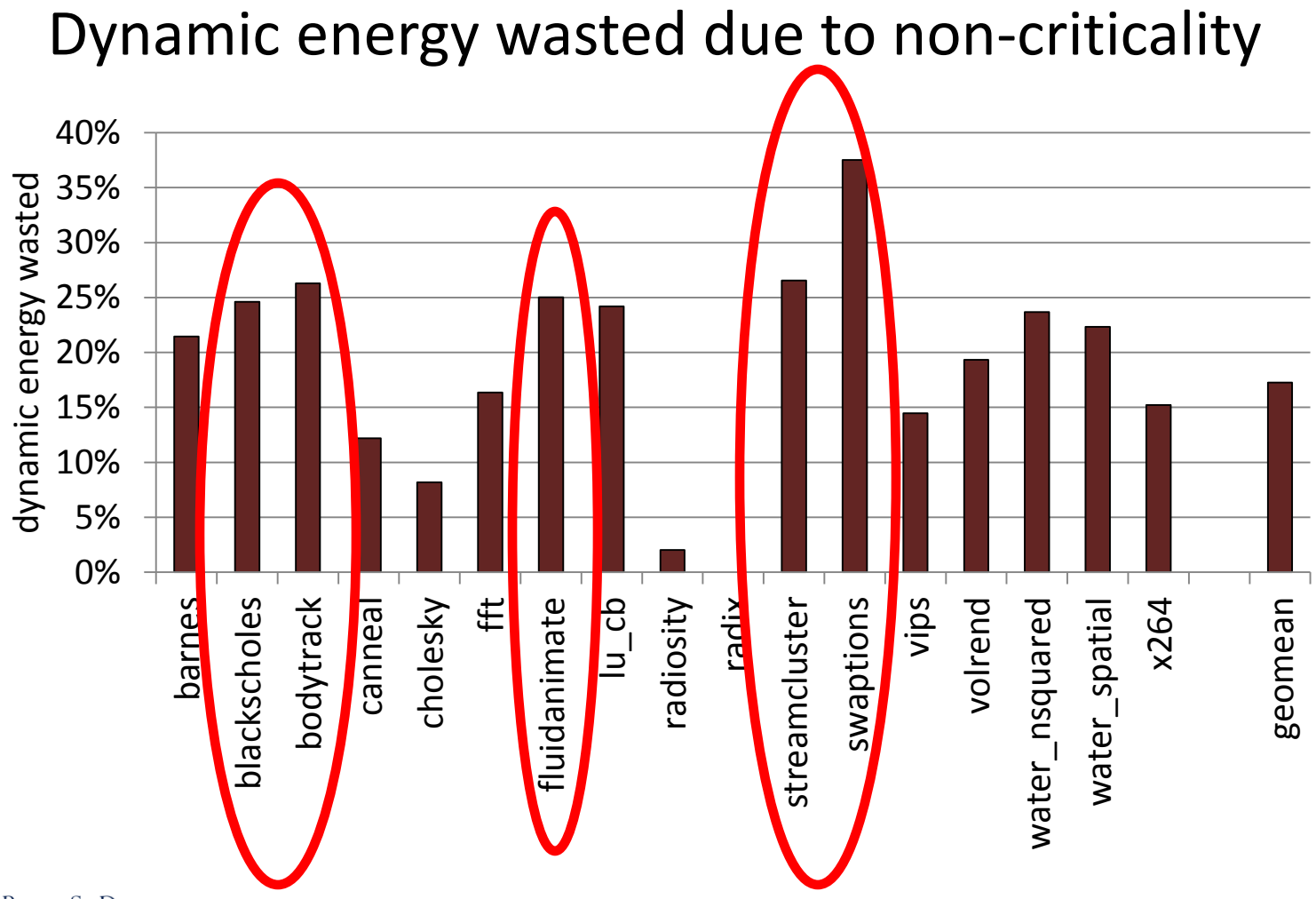
e.g., bodytrack:



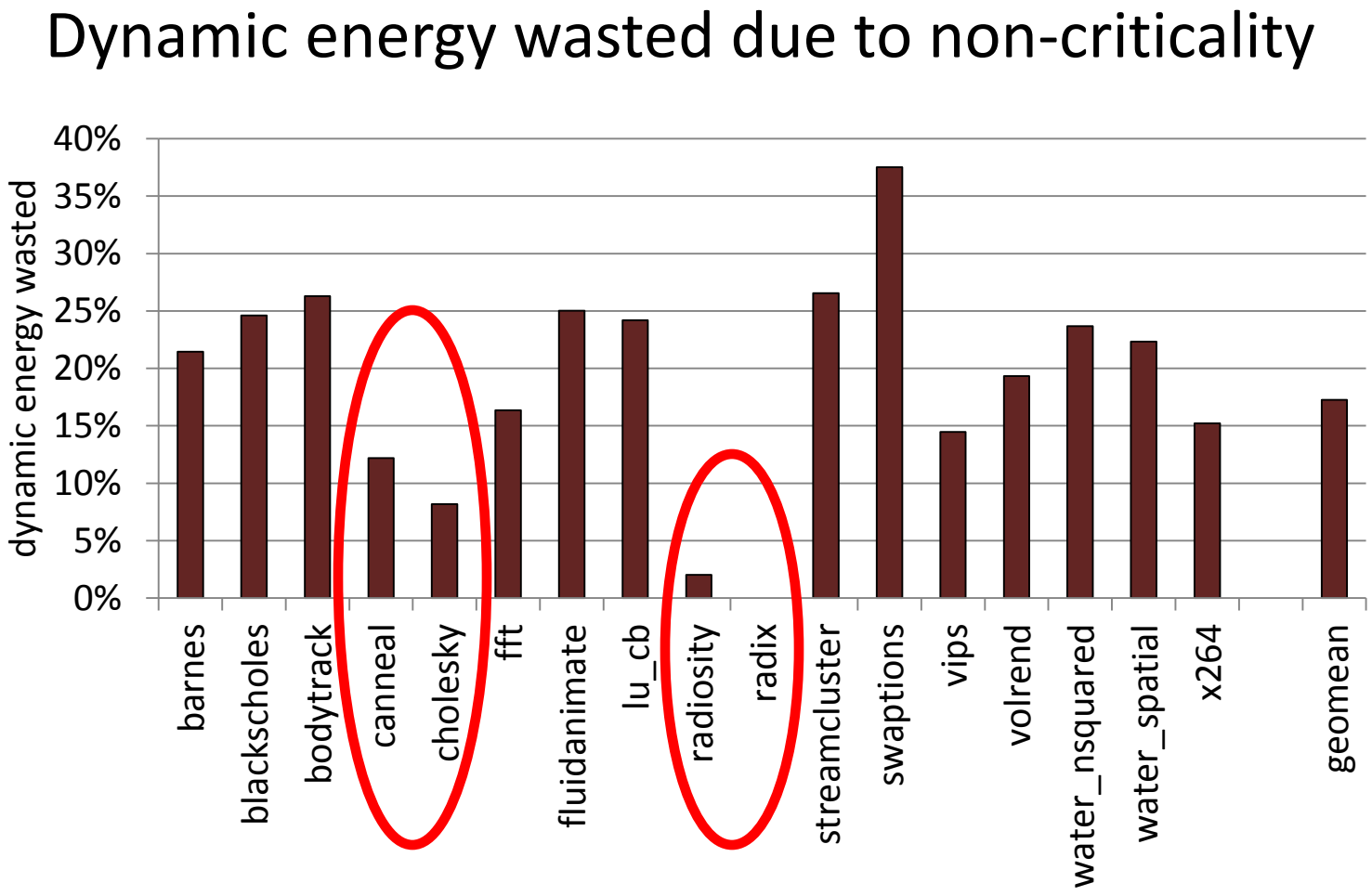
Measuring Criticality – Energy Wasted



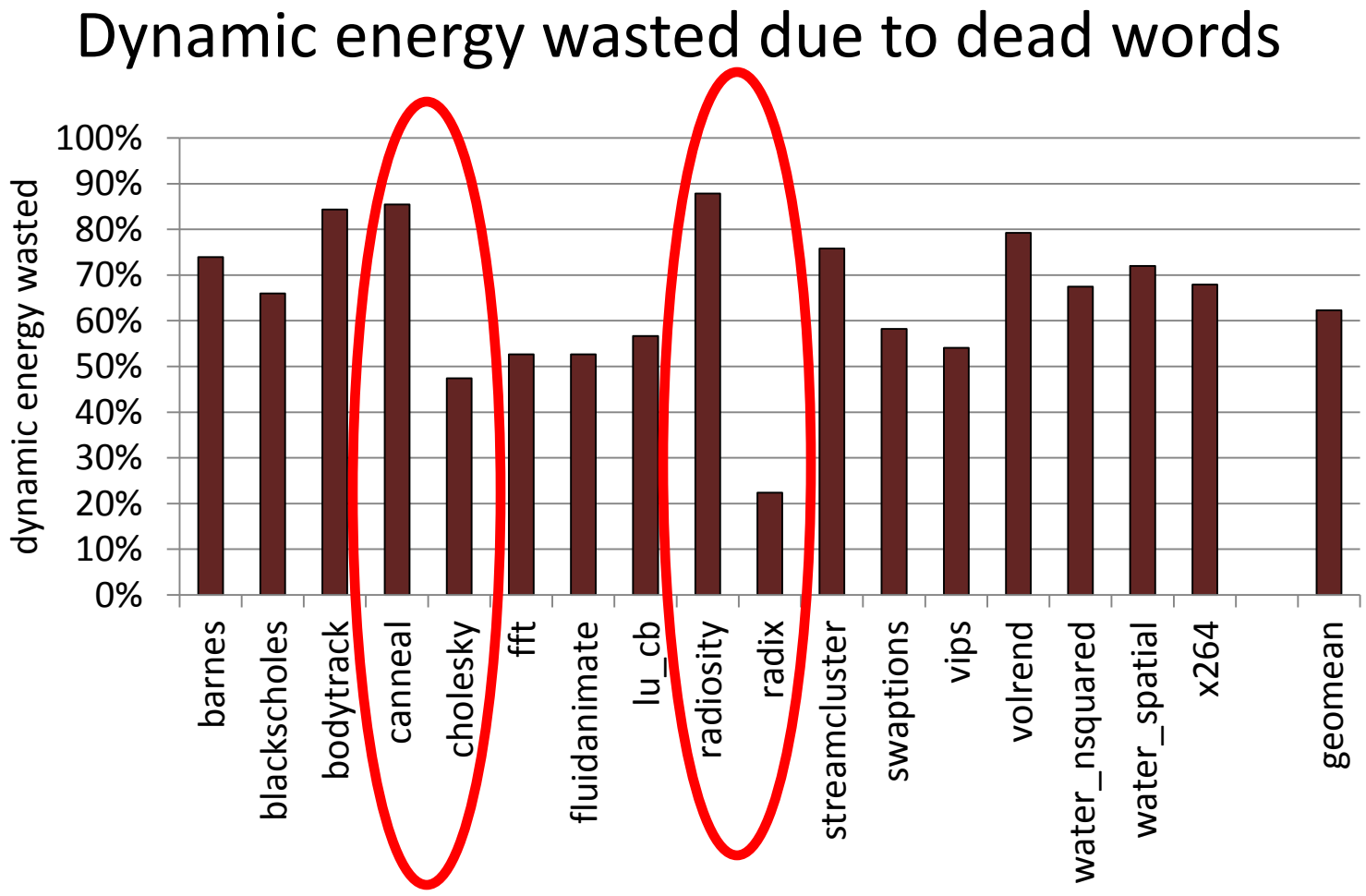
Measuring Criticality – Energy Wasted



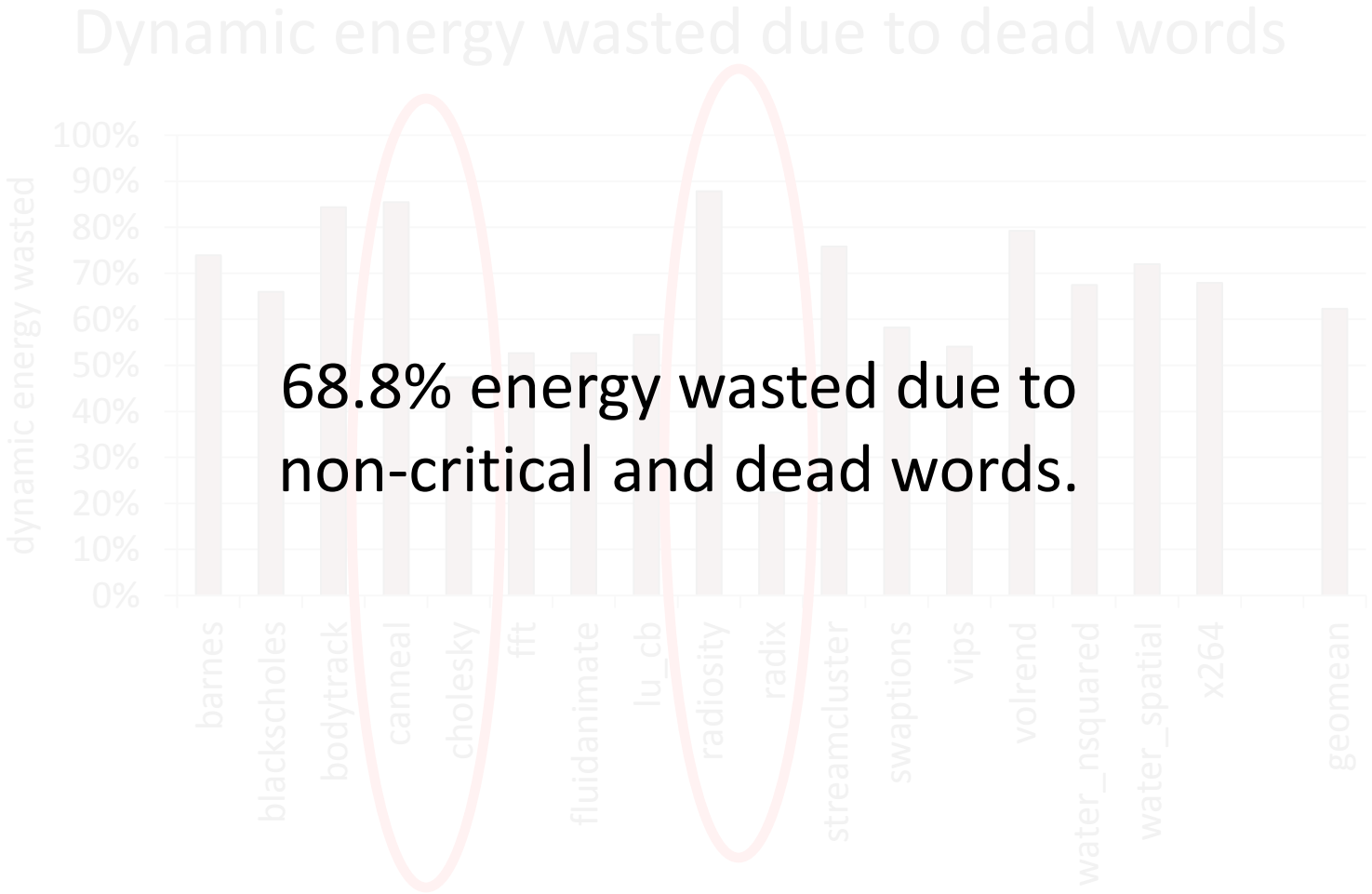
Measuring Criticality – Energy Wasted



Measuring Criticality – Energy Wasted



Measuring Criticality – Energy Wasted



68.8% energy wasted due to non-critical and dead words.

Outline

Defining Criticality

- Data Criticality
- Data Liveness

Measuring Criticality

- Energy Wasted

Addressing Criticality

- **NoCNoC**

Addressing Criticality

A criticality-aware NoC design needs to:

1. Predict the criticality of a word prior to fetching it.
2. Separate the fetching of words based on their criticality.
3. Reduce energy consumption in fetching low-criticality words.
4. Eliminate the fetching of dead words.

NoCNoC (Non-Critical NoC)

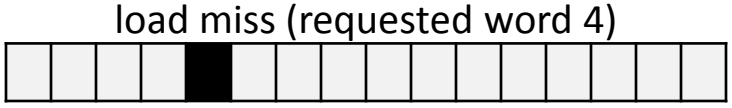
A proof-of-concept, criticality-aware design

NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.

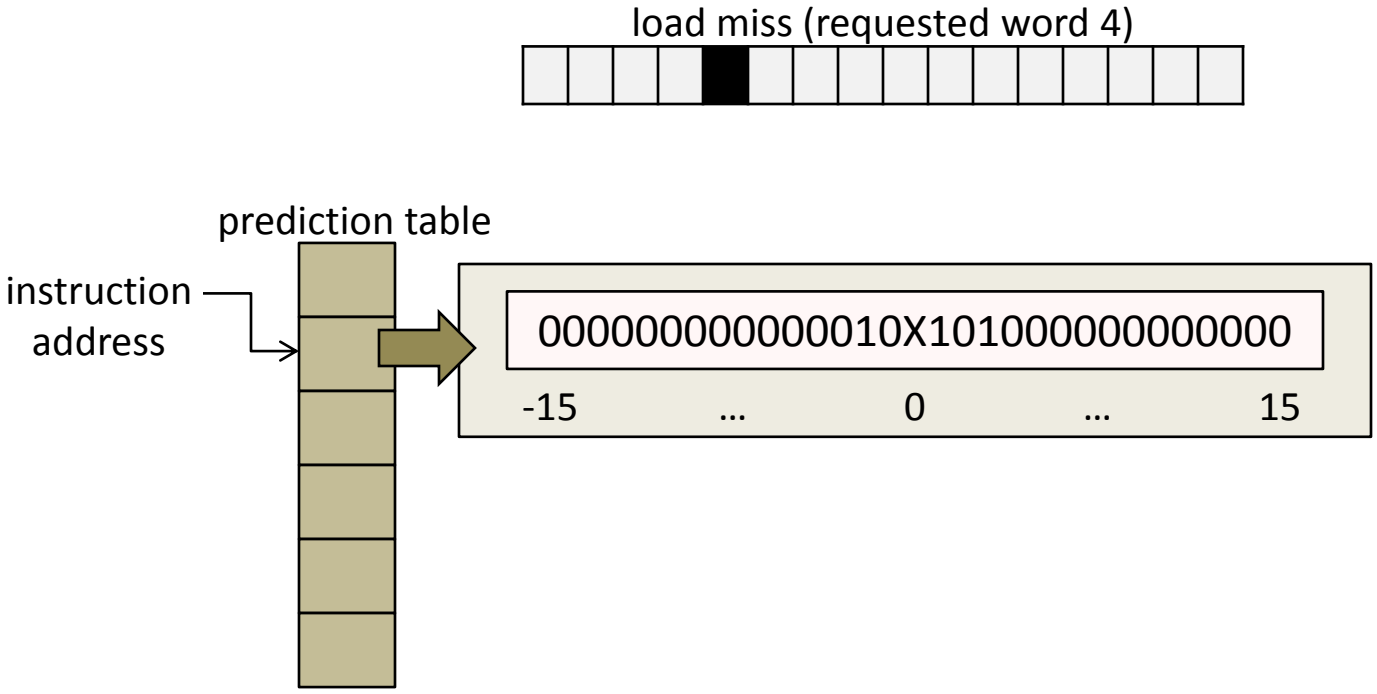
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



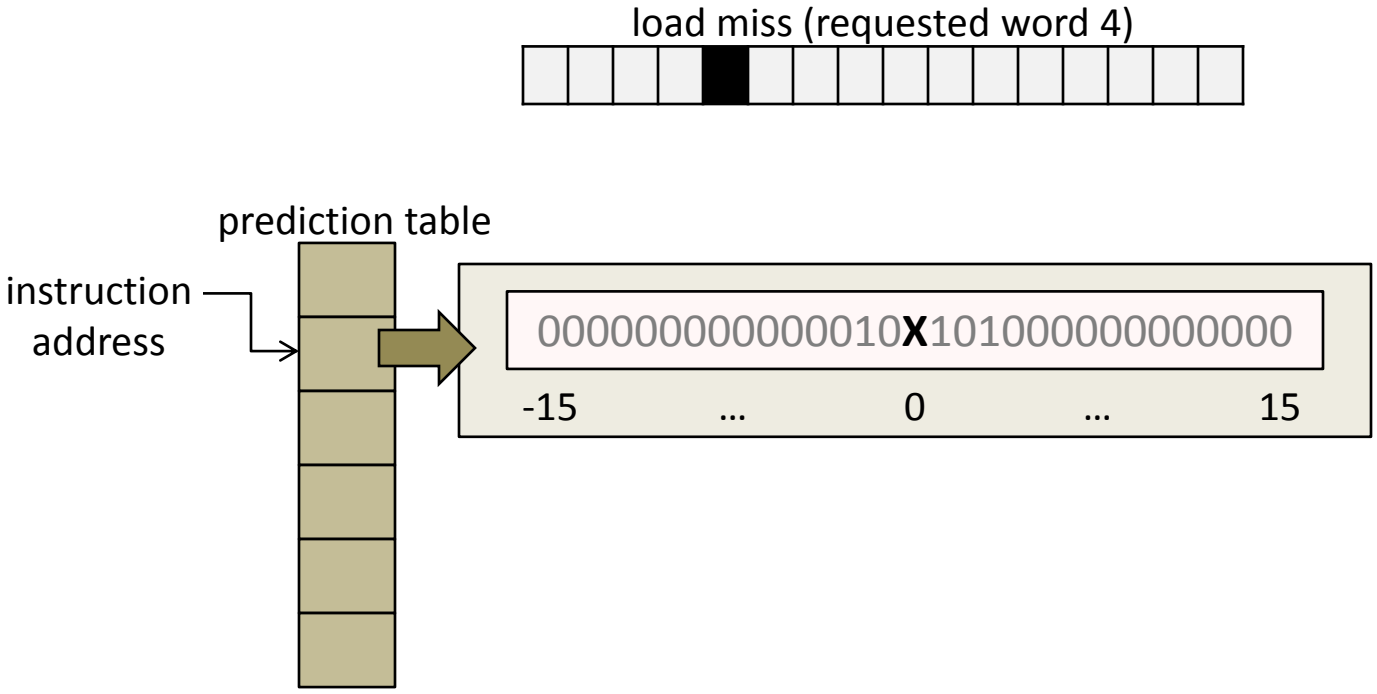
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



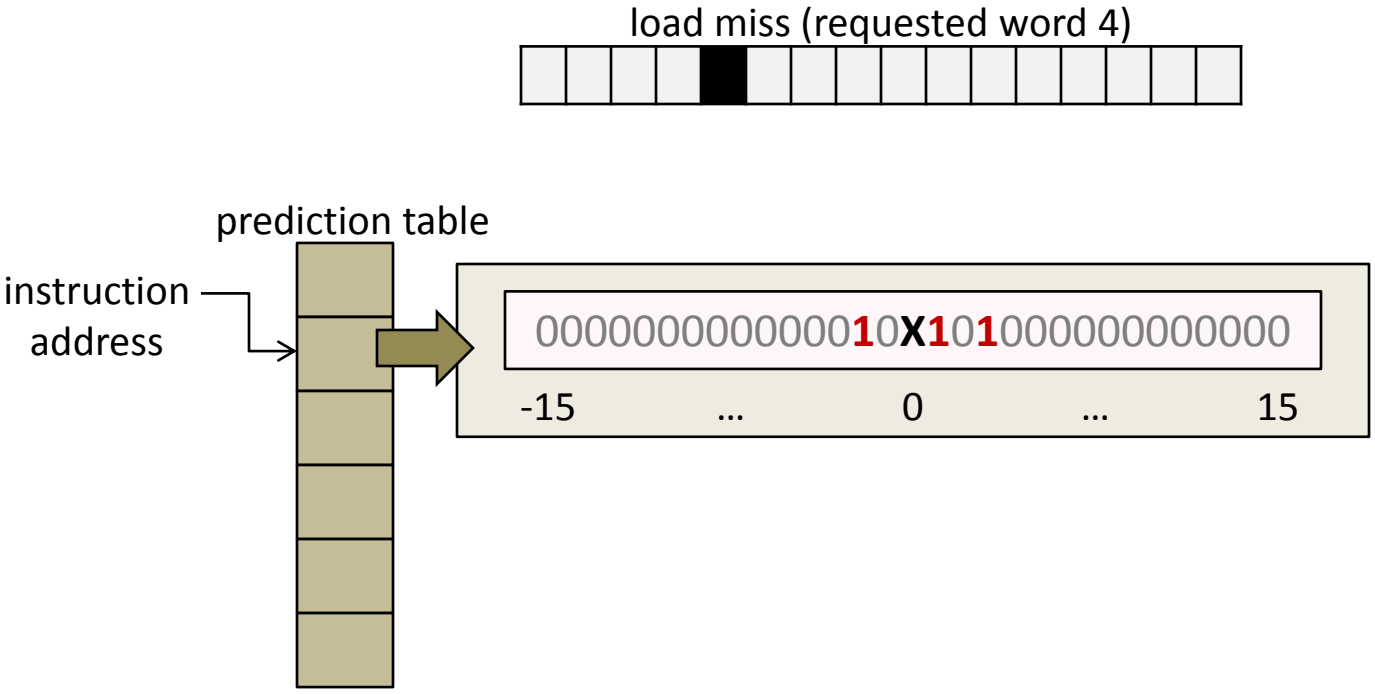
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



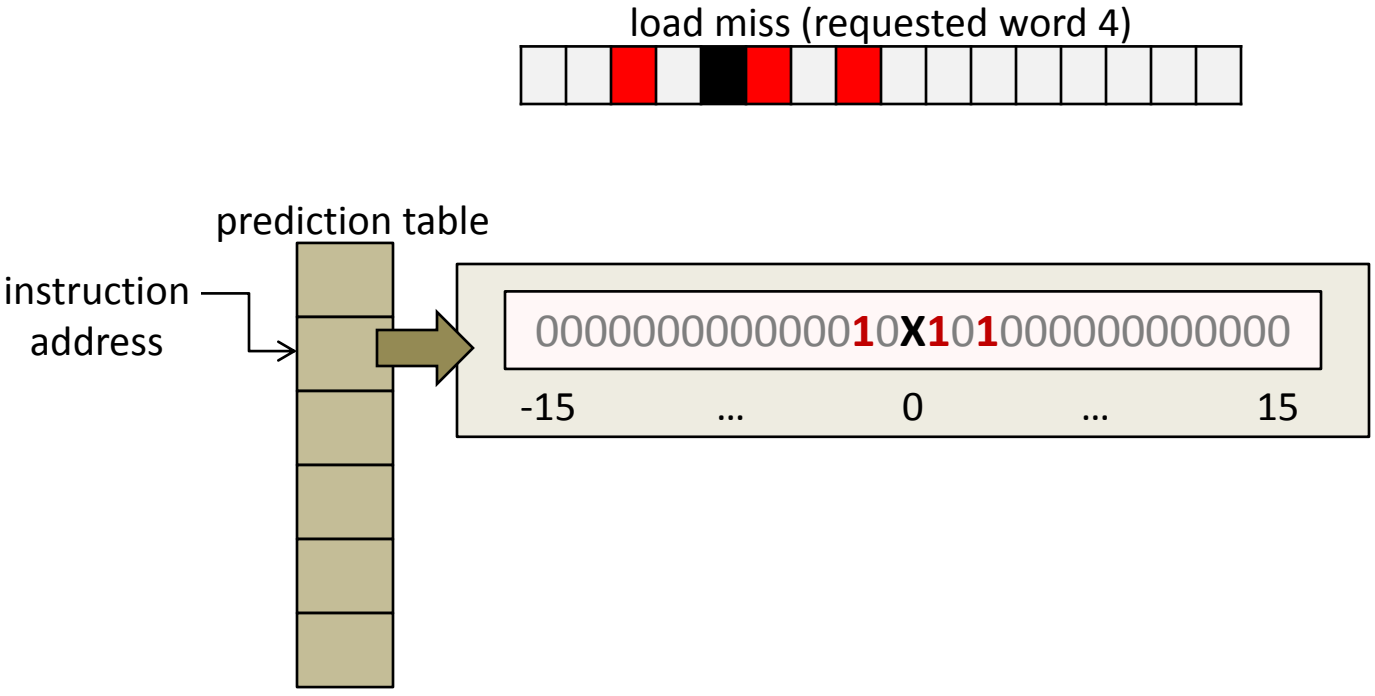
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



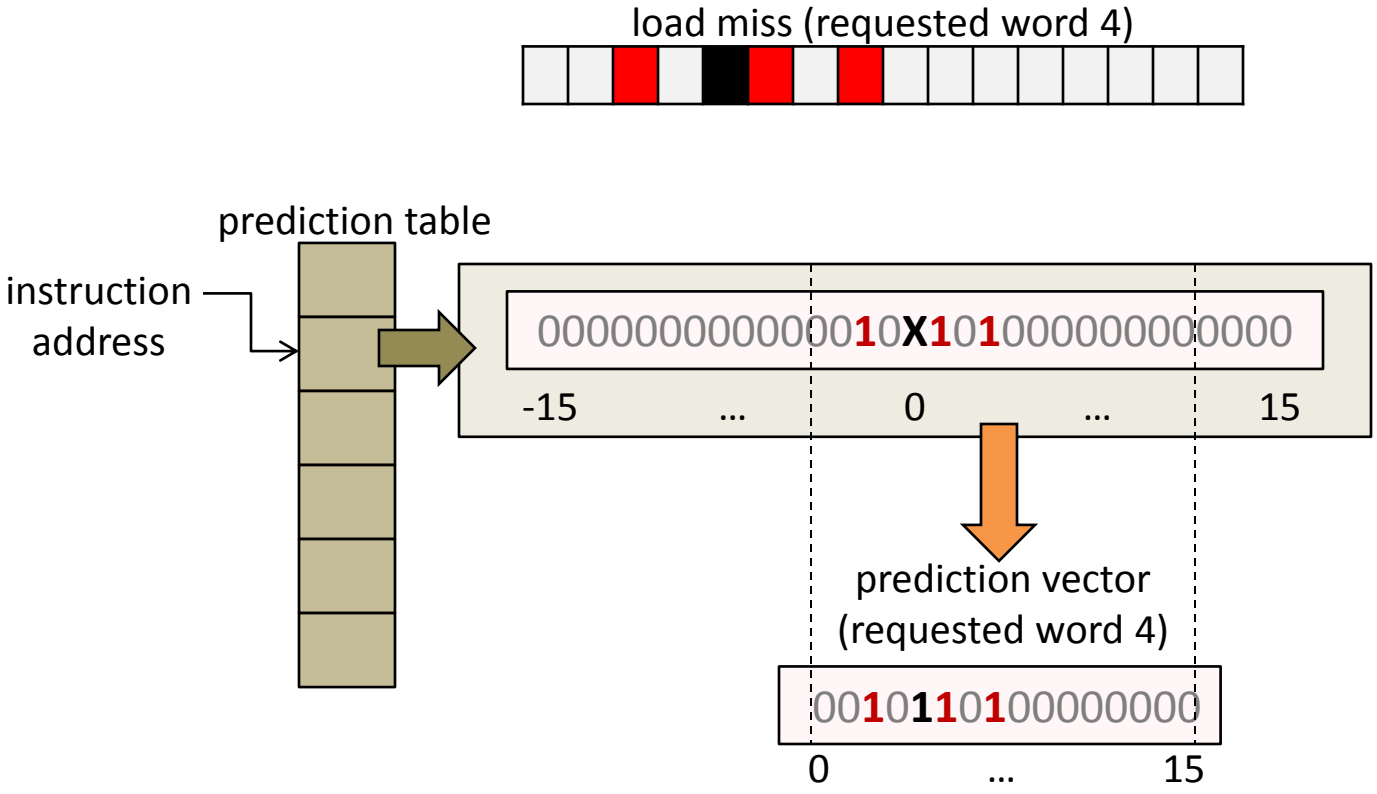
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



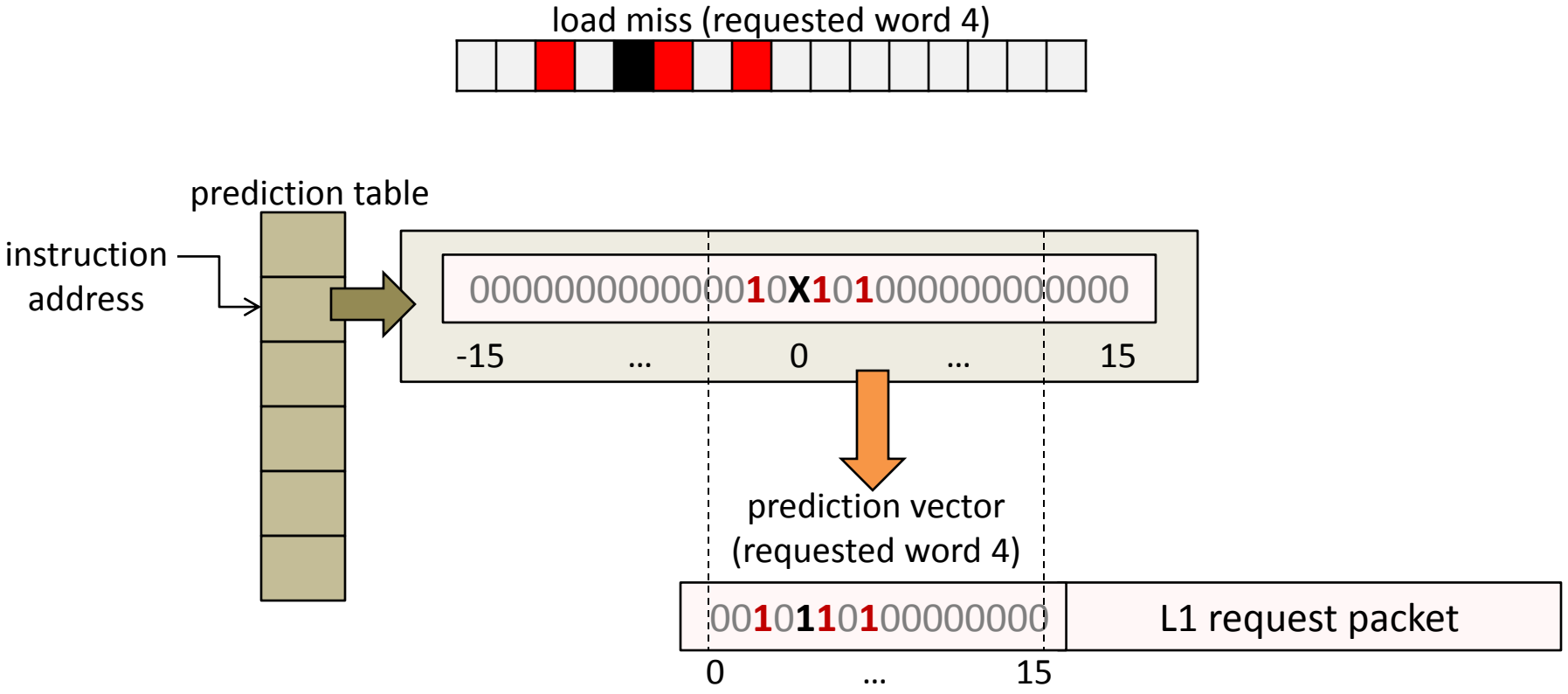
NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.



NoCNoC

1. Predict the criticality of a word prior to fetching it.
   - Binary predictor: either critical or non-critical.

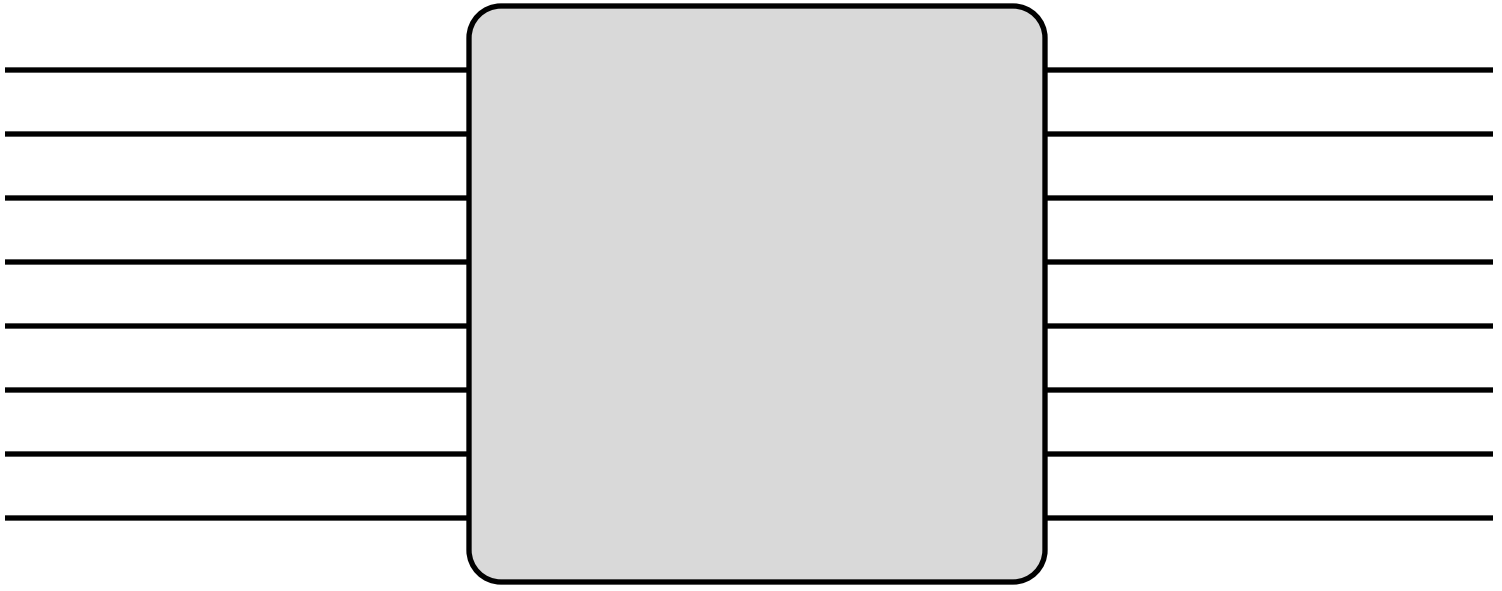


NoCNoC

2. Separate the fetching of words based on their criticality.
   - Two physical subnetworks: critical and non-critical.

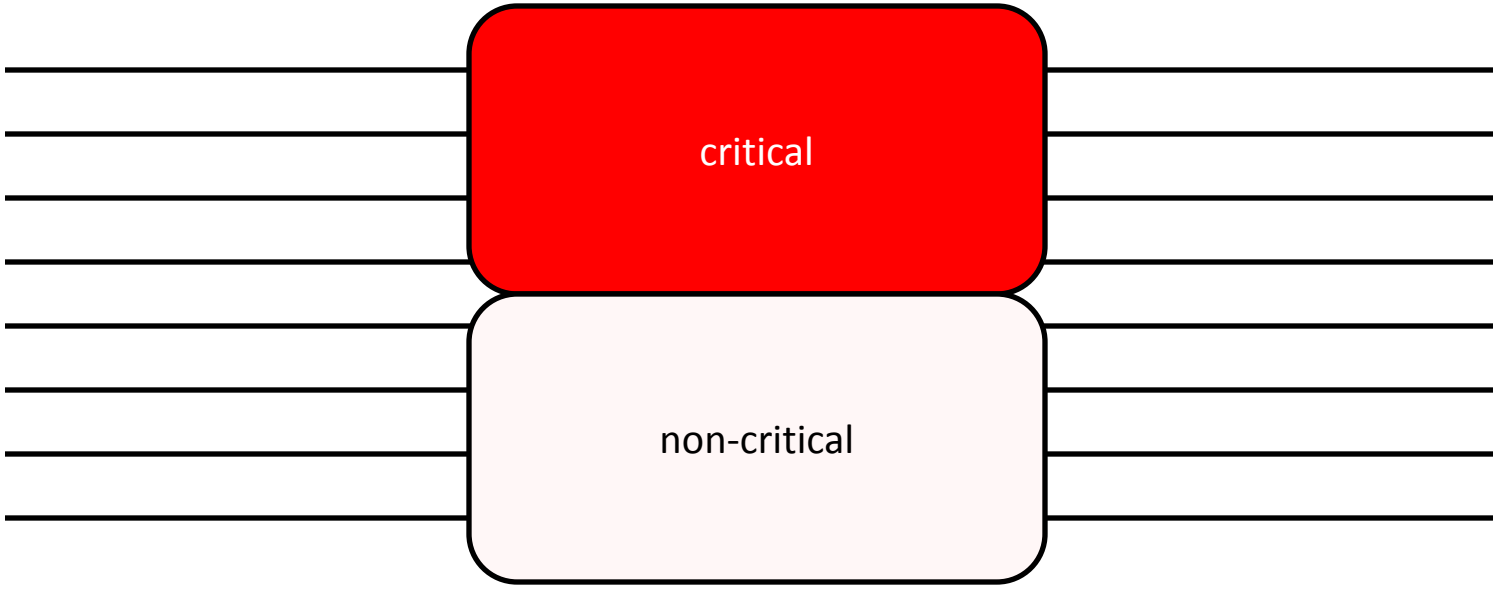
NoCNoC

2. Separate the fetching of words based on their criticality.
   - Two physical subnetworks: critical and non-critical.



NoCNoC

2. Separate the fetching of words based on their criticality.
   - Two physical subnetworks: critical and non-critical.

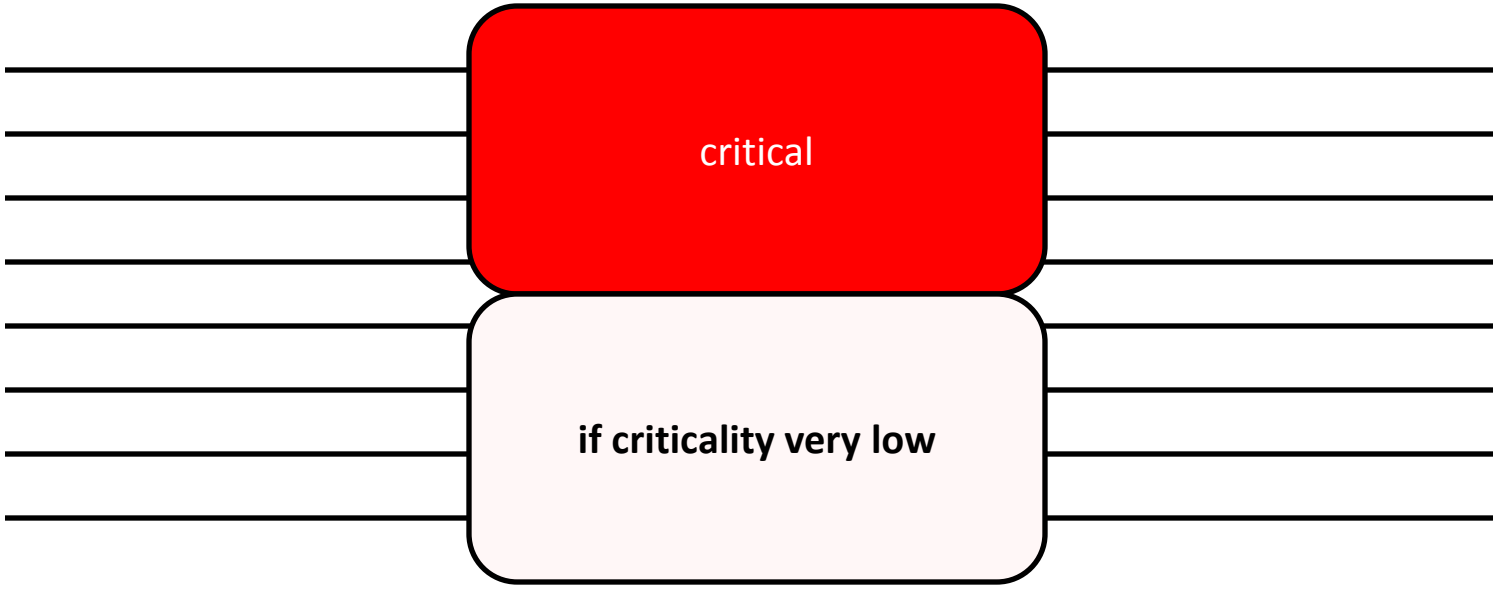


NoCNoC

3. Reduce energy consumption in fetching low-criticality words.
   - DVFS in non-critical subnetwork.

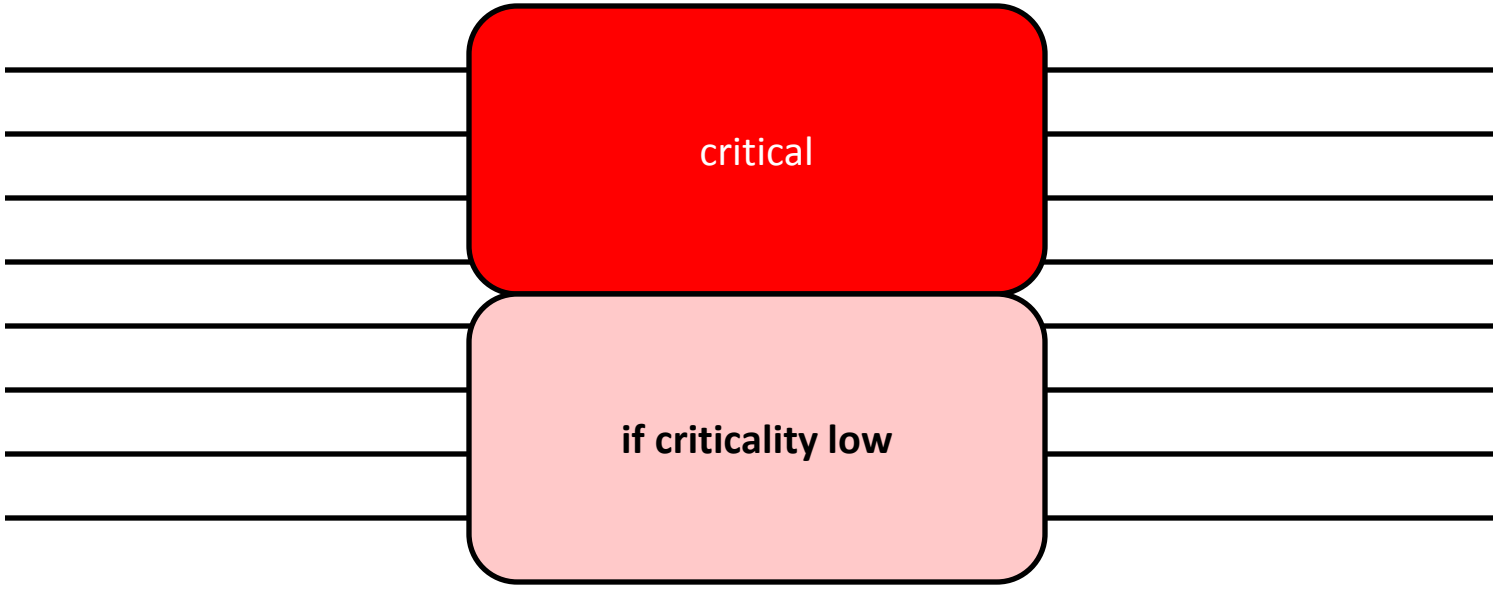
NoCNoC

3. Reduce energy consumption in fetching low-criticality words.
   - DVFS in non-critical subnetwork.



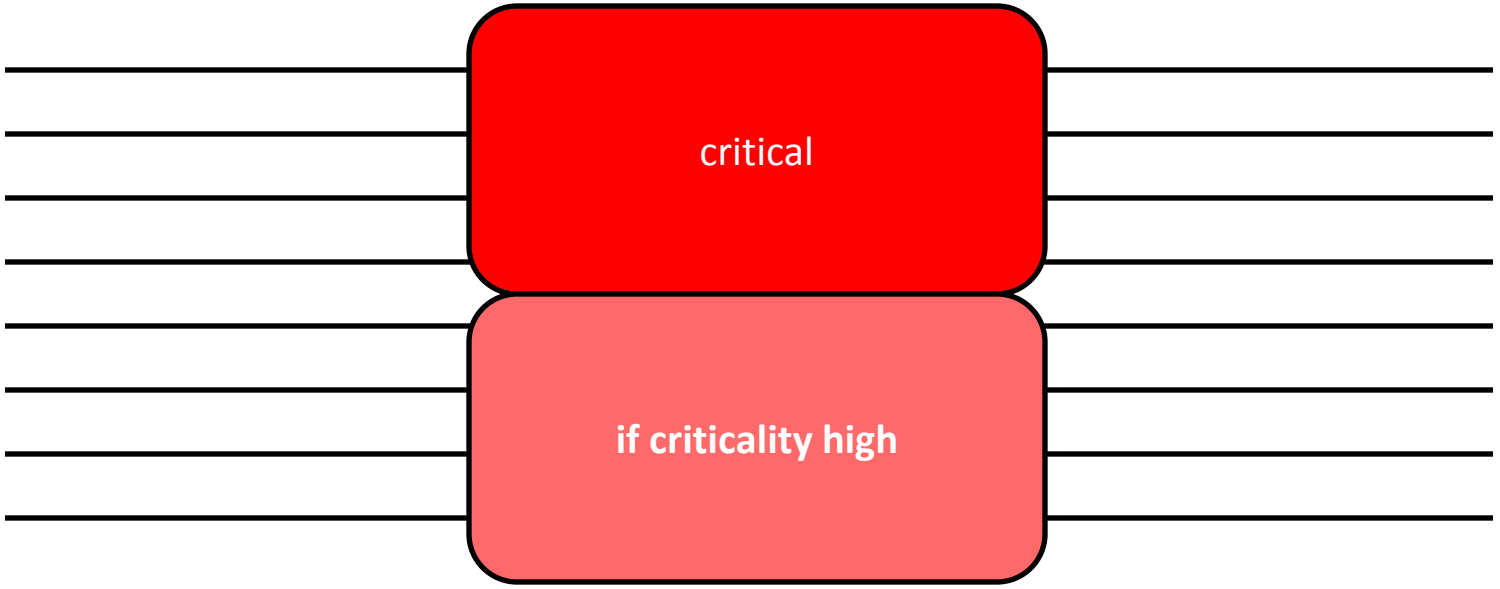
NoCNoC

3. Reduce energy consumption in fetching low-criticality words.
   - DVFS in non-critical subnetwork.



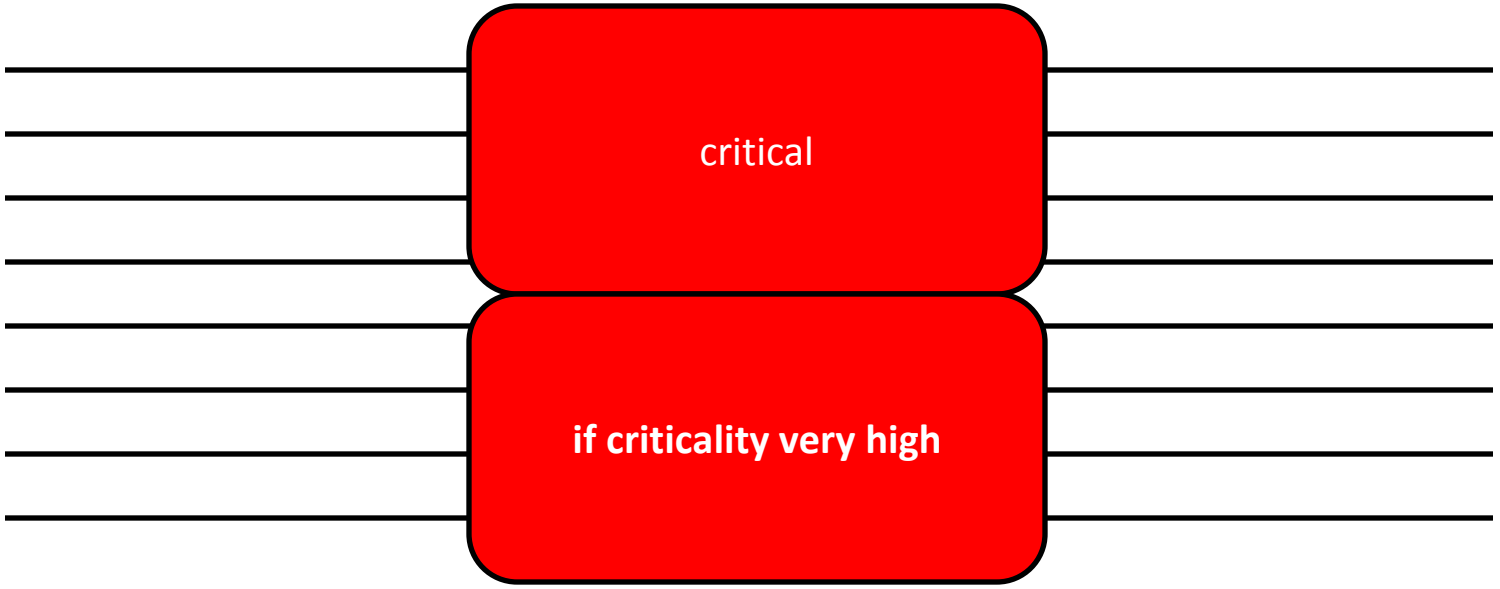
NoCNoC

3. Reduce energy consumption in fetching low-criticality words.
   - DVFS in non-critical subnetwork.



NoCNoC

3. Reduce energy consumption in fetching low-criticality words.
   - DVFS in non-critical subnetwork.

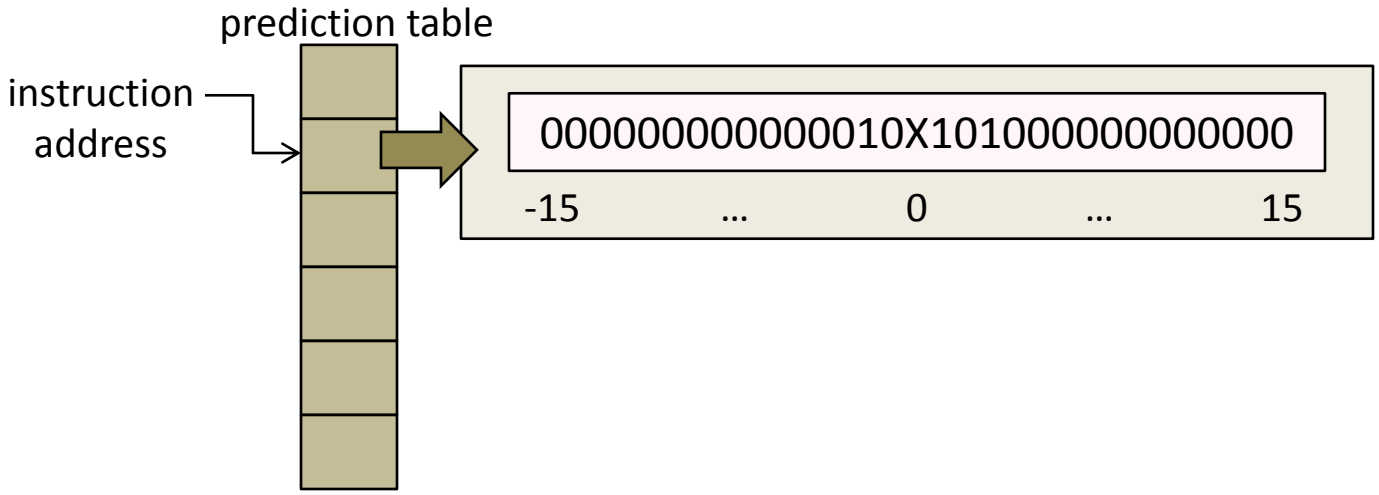


NoCNoC

4. Eliminate the fetching of dead words.
   - Binary predictor: either live or dead.

NoCNoC

4. Eliminate the fetching of dead words.
   - Binary predictor: either live or dead.

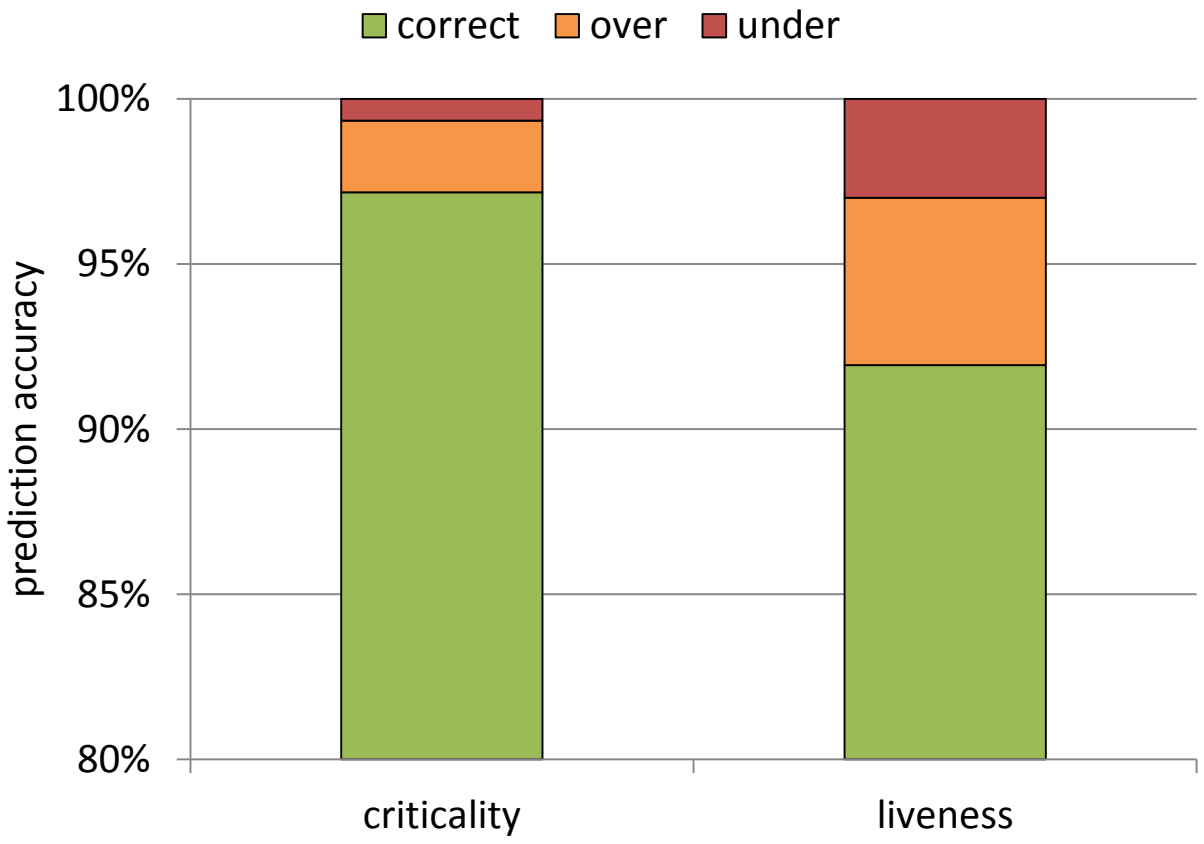


NoCNoC

More details and results in the paper:

- DVFS scheme
- Prediction tables
- Comparison to instruction criticality (Aergia [R. Das, ISCA 2010])

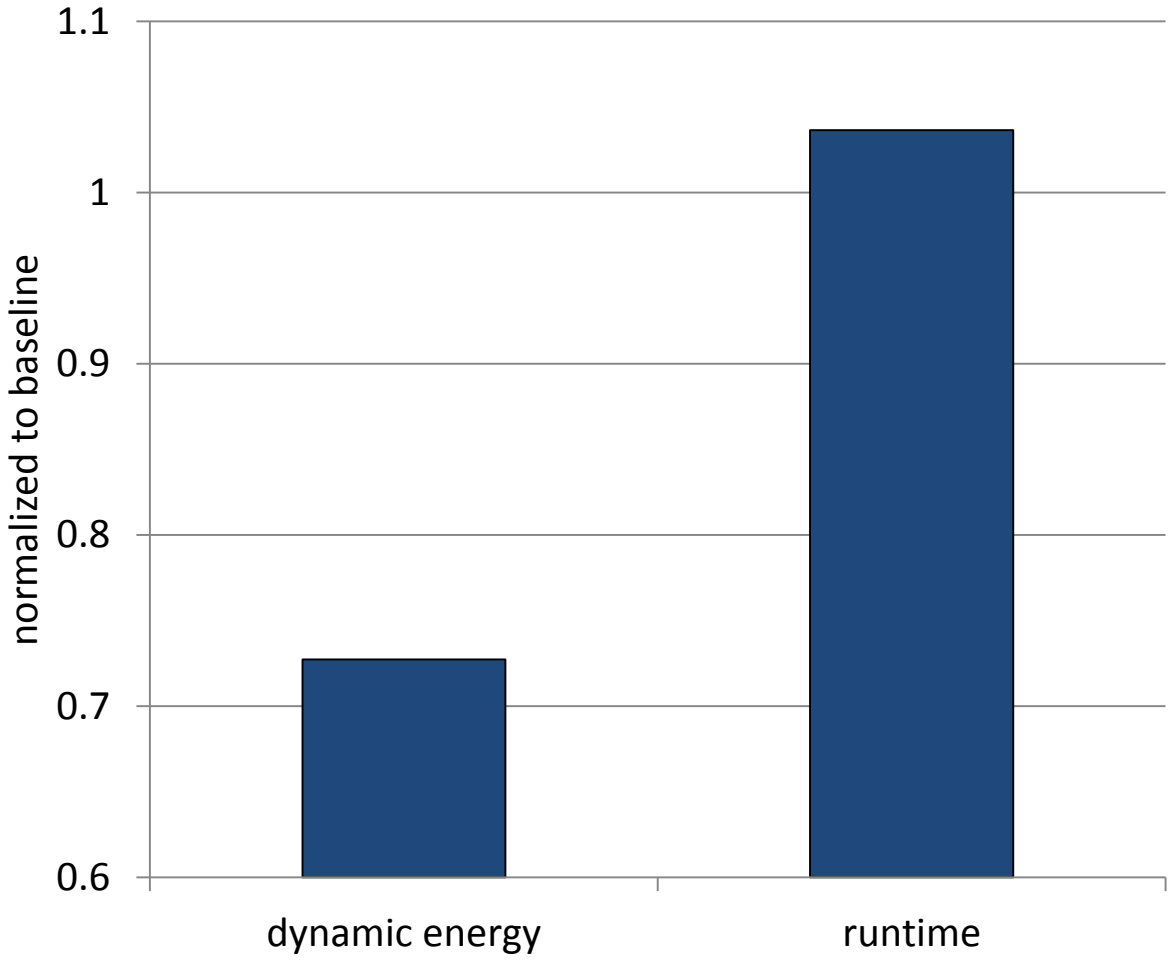
NoCNoC – Prediction Accuracy



NoCNoC – Prediction Accuracy

Outperforms prior liveness predictor (70% accuracy) [H. Kim, NOCS 2011]

NoCNoC – Performance and Energy



Conclusion

Define **Data Criticality**

- Deliver data both no later and **no earlier** than needed.

Measure Criticality

- 68.8% energy wasted due to non-critical and dead words.

Address Criticality

- NoCNoC, a proof-of-concept, criticality-aware design.

Thank you