Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Enabling Hardware Randomization Across the Cache Hierarchy in Linux-Class Processors

Max Doblas[1] , Ioannis-Vatistas Kostalabros[1] , Miquel Moretó[1] and Carles Hernández[2]

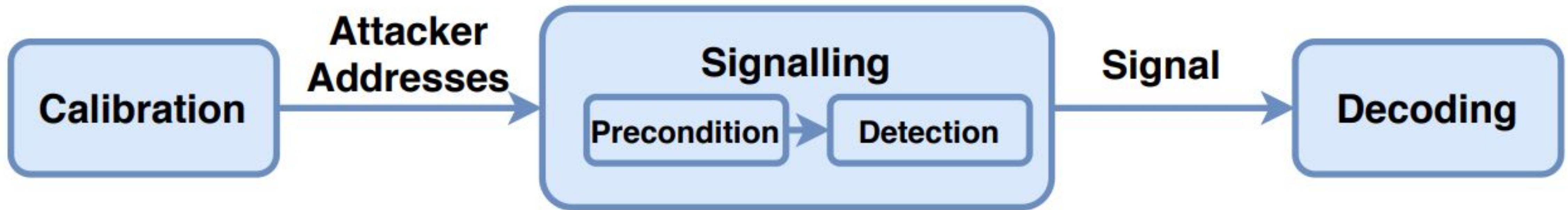
[1]Computer Sciences - Runtime Aware Architecture, Barcelona Supercomputing Center
{ max.doblas, vatistas.kostalabros, miquel.moreto } @bsc.es

[2]Department of Computing Engineering, Universitat Politècnica de València
carherlu@upv.es

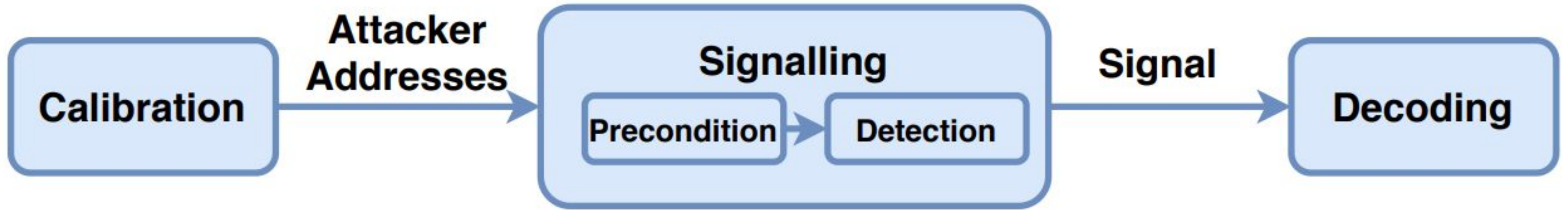
Introduction

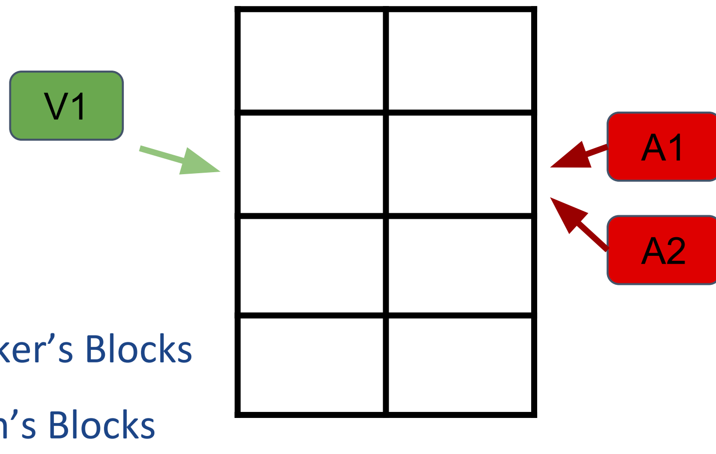
- Cache-based side channel attacks are a serious concern in many computing domains
- Existing randomizing proposals can not deal with virtual memory
  - The majority of the state-of-the-art is focussing at the LLCs
- Our proposal enables randomizing the whole cache hierarchy of a Linux-capable RISC-V processor

Cache Side Channel Attacks



Cache Side Channel Attacks



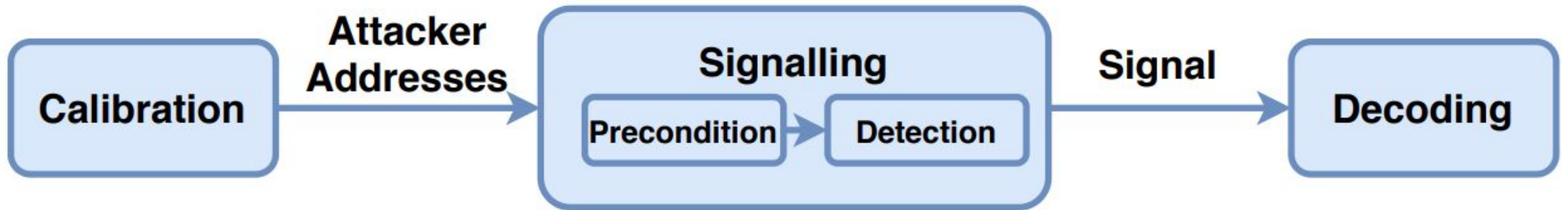


Prime+Probe Example

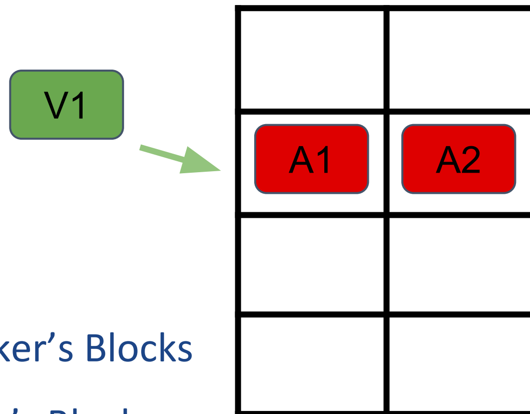
1. Calibration

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Cache Side Channel Attacks



4 sets, 2 way associative cache



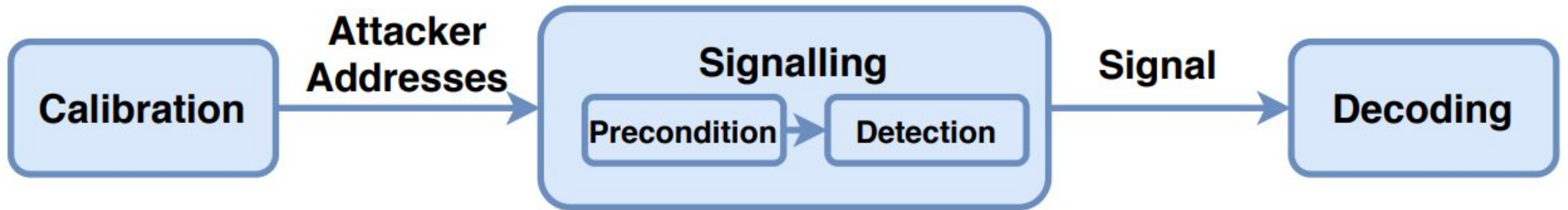
Ax Attacker's Blocks

Vx Victim's Blocks

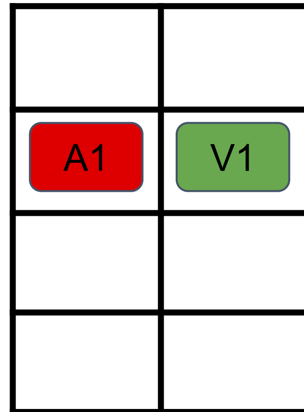
Prime+Probe Example

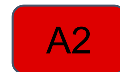
1. Calibration
2. Prime (precondition)

Cache Side Channel Attacks



4 sets, 2 way associative cache





Ax Attacker's Blocks

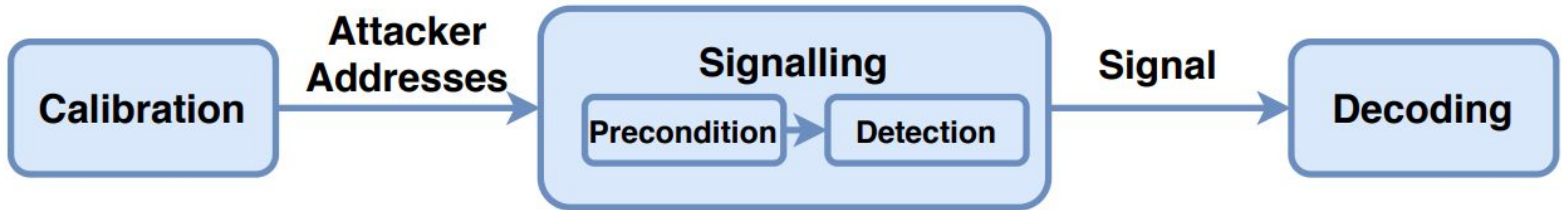
Vx Victim's Blocks

Prime+Probe Example

1. Calibration
2. Prime (precondition)
3. Wait(execution of the victim)

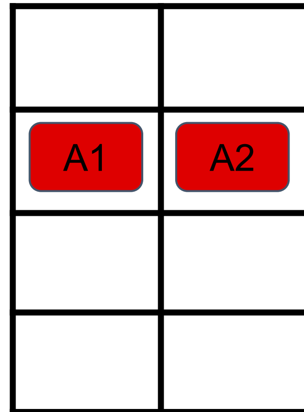
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Cache Side Channel Attacks



4 sets, 2 way associative cache

V1



Ax Attacker's Blocks

Vx Victim's Blocks

Prime+Probe Example

1. Calibration
2. Prime (precondition)
3. Wait(execution of the victim)
4. Probe (detection)

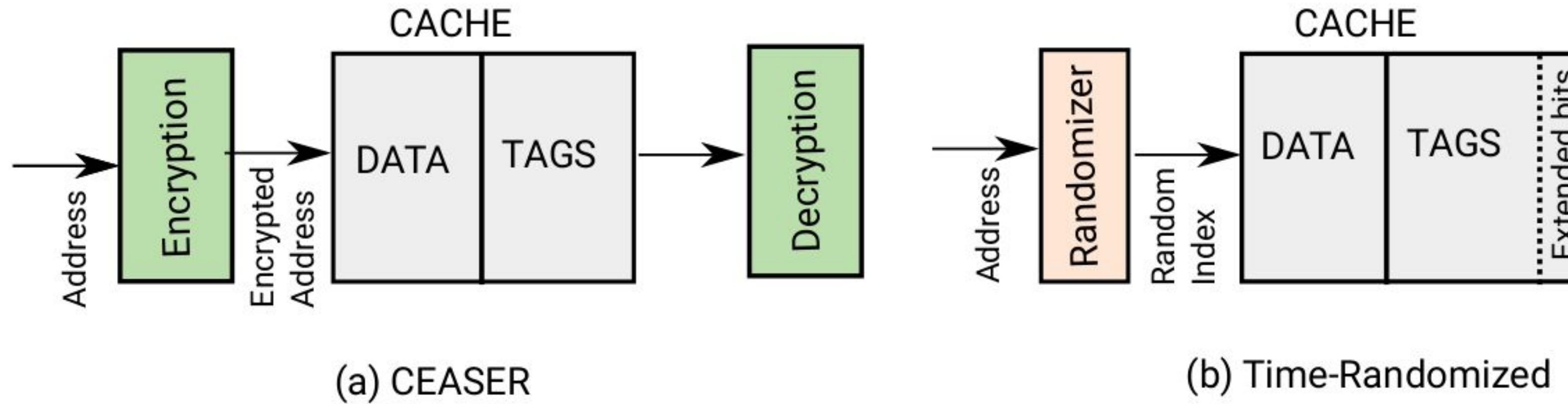
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

State of the art

Cache-layout randomization schemes

- Parametric functions that randomize the mapping of a block inside the cache
  - Use a key-value to change the hashing applied to the address
  - At every key change a new calibration has to be performed
  - Protection is provided by modifying the key frequently
- It can be used in single or multiple security domains

State of the art



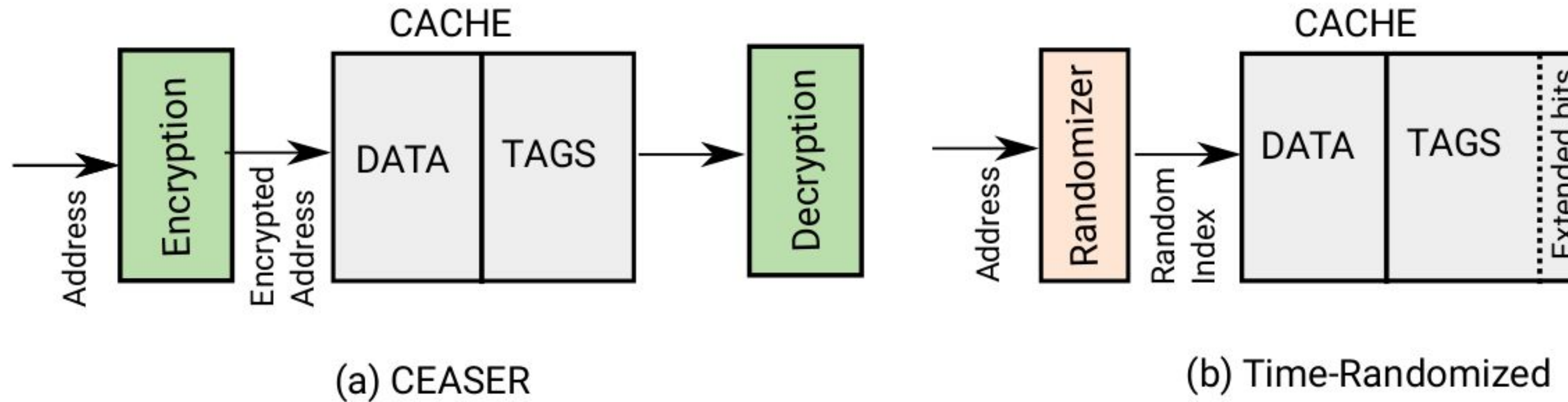
(a) CEASER

(b) Time-Randomized

- (a) Some solutions use an Encryption-Decryption scheme
  - Introduces latency -> Potential high impact in cache latency
  - Improves design simplicity by not altering the cache structure

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

State of the art

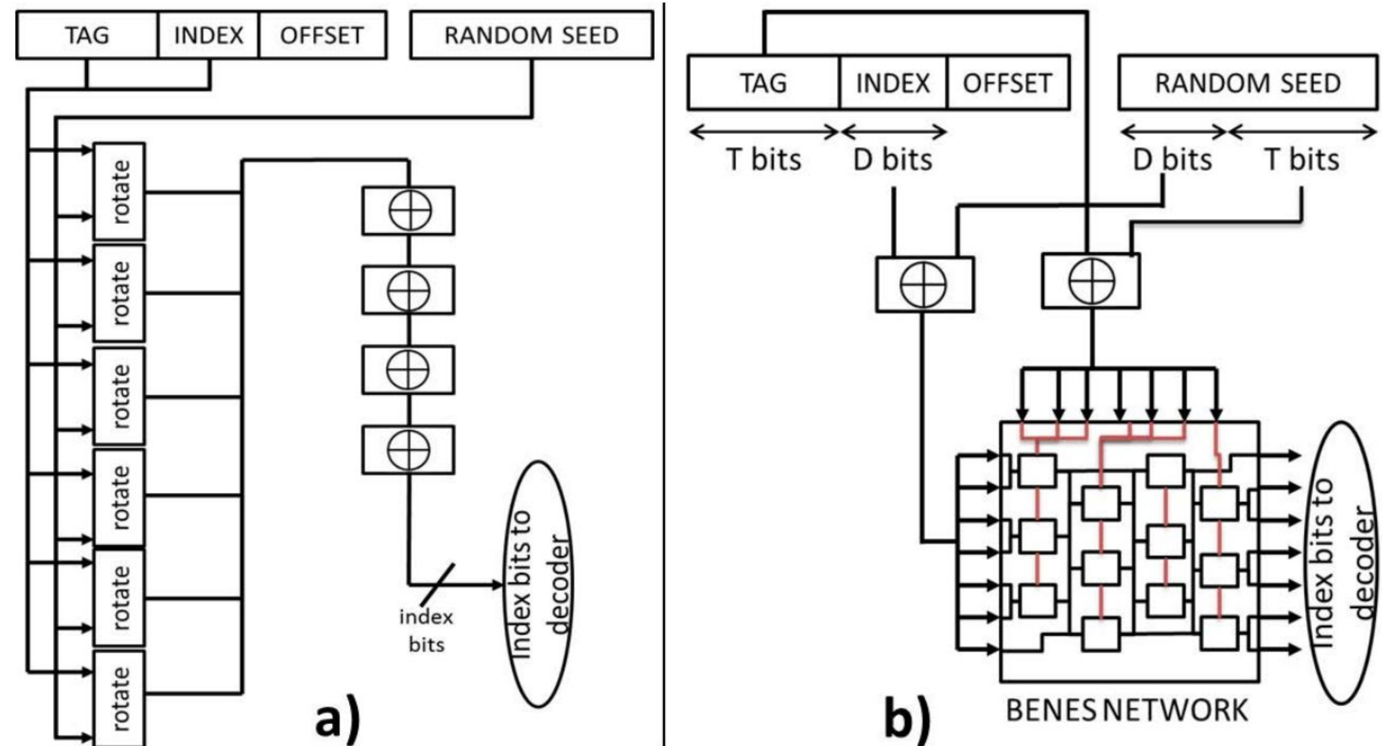


- (b) Randomization function produces the cache-set's index
  - Latency can be partially hidden-> feasible for first level caches
  - Needs to increase the Tags to recover block address
  - Extra mechanism is needed to enable the virtual memory

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Randomization Functions Quality

- Randomization functions need to balance security performance trade-off
- CEASER's LLBC
  - Inherent linearity deems it useless for SCA thwarting [1]
- Balance time randomized functions examples [2]:
  a) Hash Function
  b) Random mopdulo

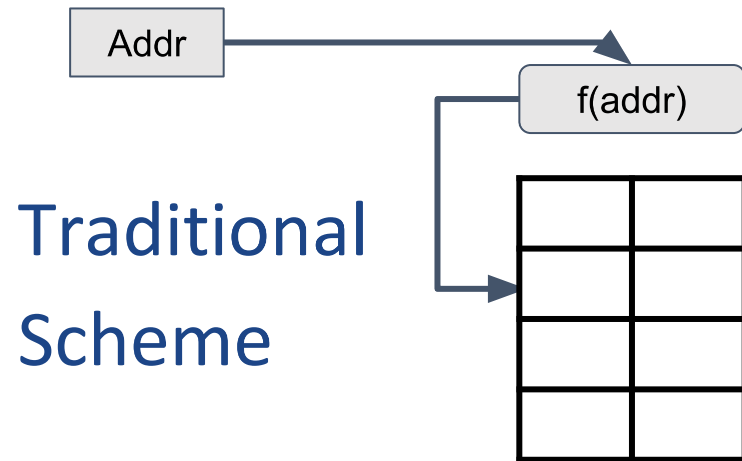


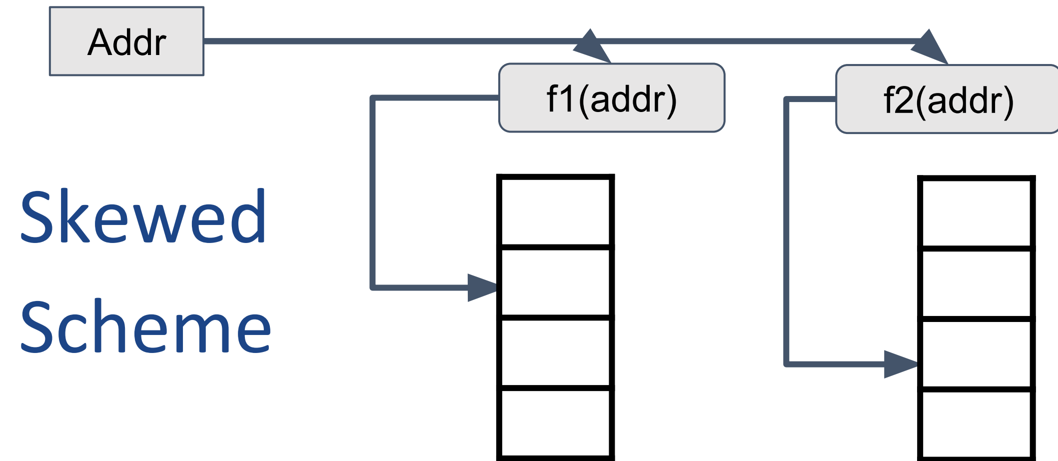
[1] R. Bodduna, V. Ganesan, P. Slpsk, C. Rebeiro, and V. Kamakoti. Brutus: Refuting the security claims of the cache timing randomization coun- termeasure proposed in ceaser. IEEE Computer Architecture Letters, 2020.

[2]D. Trilla, C. Hernández, J. Abella, and F. J. Cazorla. Cache side-channel attacks and time-predictability in high-performance critical real-time systems. In DAC, pages 98:1–98:6, 2018.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Skewed Caches





- Enhances the security of the cache
  - It is more difficult to calibrate an attack
  - Increases the resources used by multiplying the number of randomization functions.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Virtual memory Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Virtual memory Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

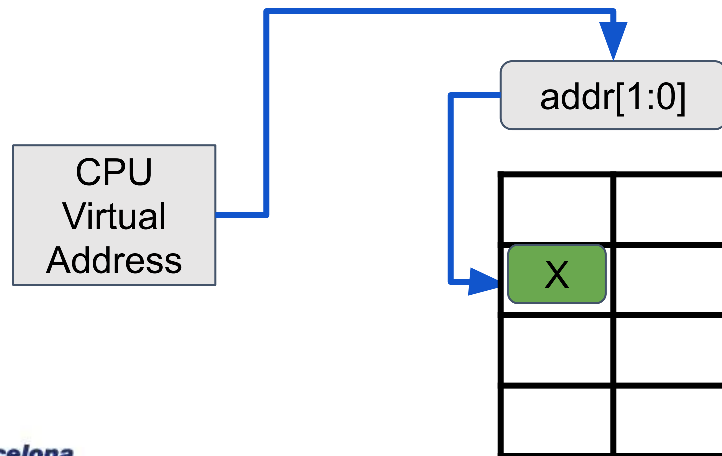
Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

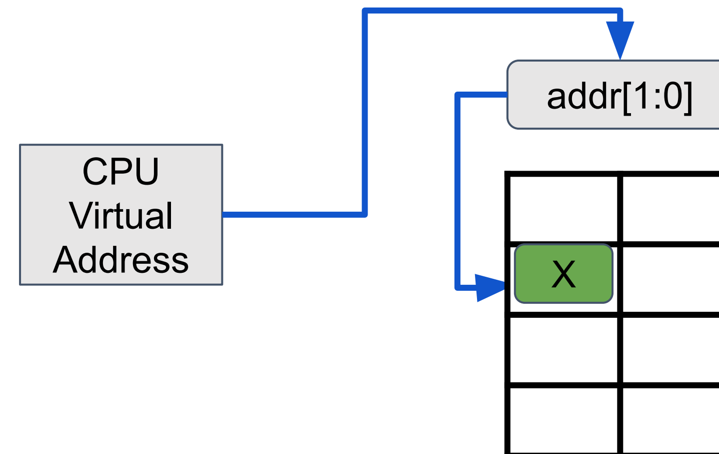
Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Process A: sb X -> 0x0001



Process B: ld 0x1001 -> r1



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Virtual memory Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

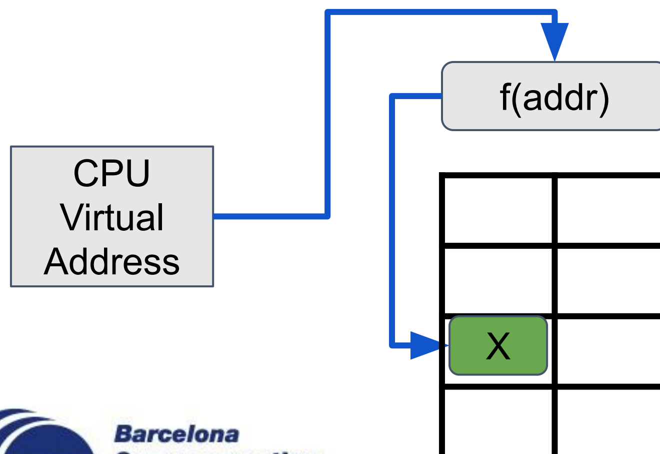
Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Proc A: sd X -> 0x0001



Virtual memory Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

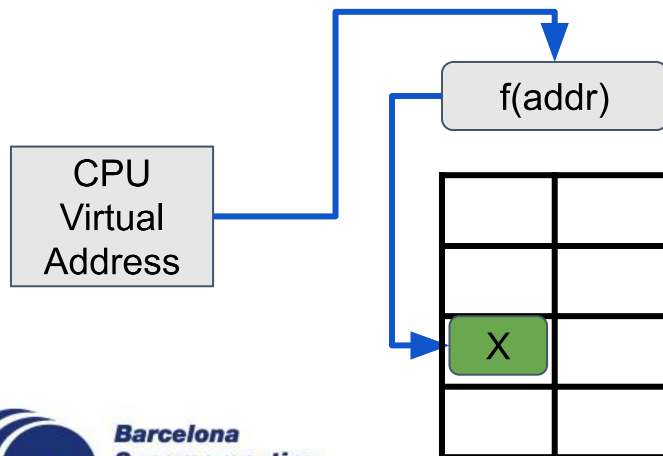
Virtual Addr	Physical Addr
0x0000	0x3000
...	...

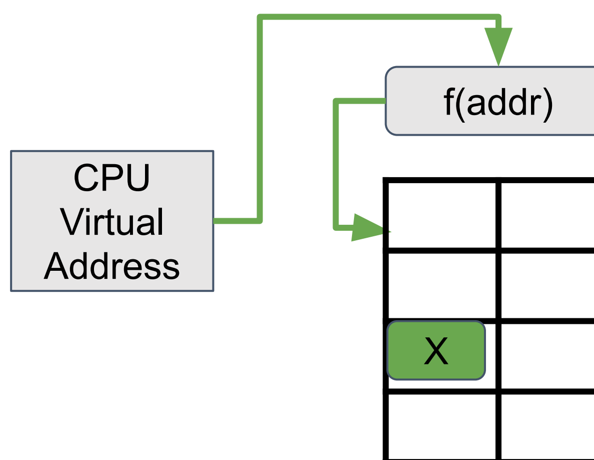
Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Proc A: sd X -> 0x0001

Proc B: ld 0x1001 -> r1





Miss

Virtual memory Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

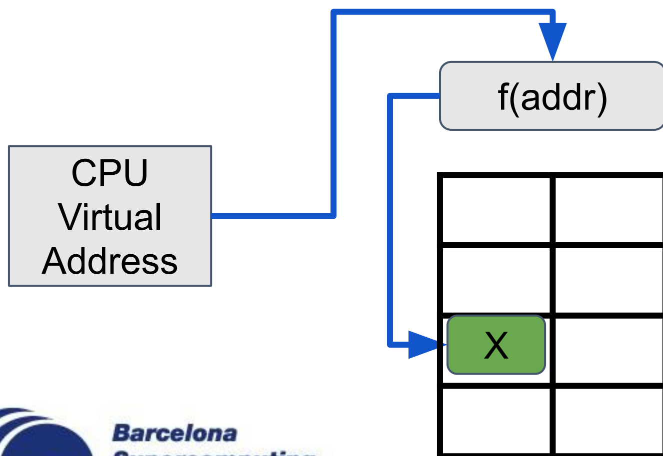
Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

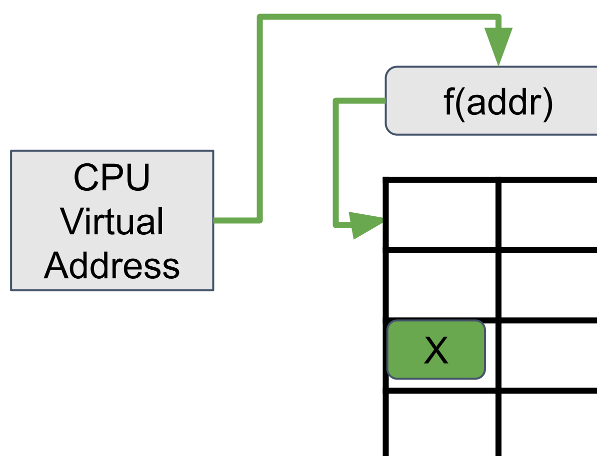
Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Proc A: sd X -> 0x0001

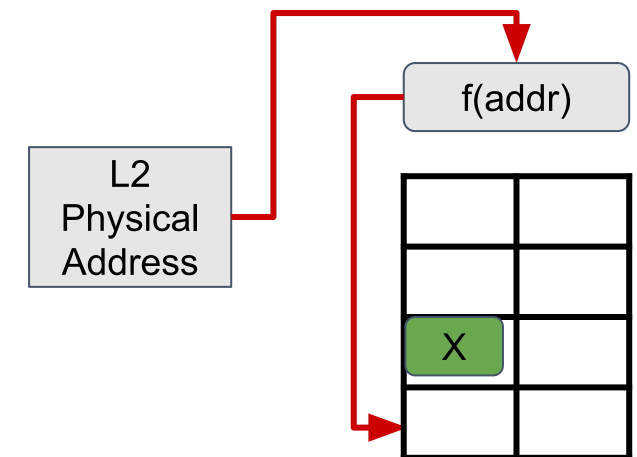


Proc B: ld 0x1001 -> r1



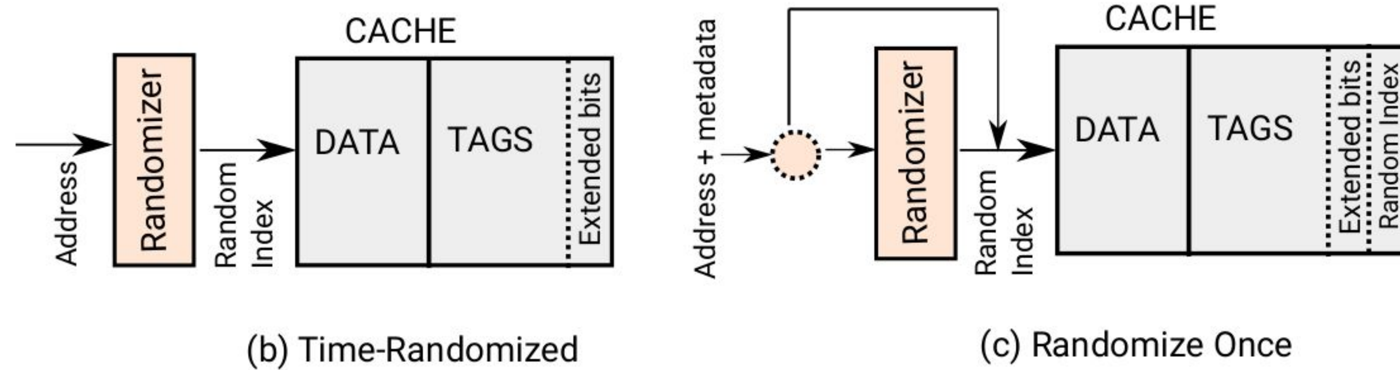
Miss

Coherency protocol access to addr 0x3001



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Proposal



- Adds supports the coherence protocol in finding any valid block.
  - Even after a key or a page-table's translation modification.
- Every cache, keeps track of the valid blocks in the lower level cache.
  - This tracking is done by storing the last random index used by the lower level cache for every valid block.
  - Using this information, the cache probes any block of the lower level cache.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

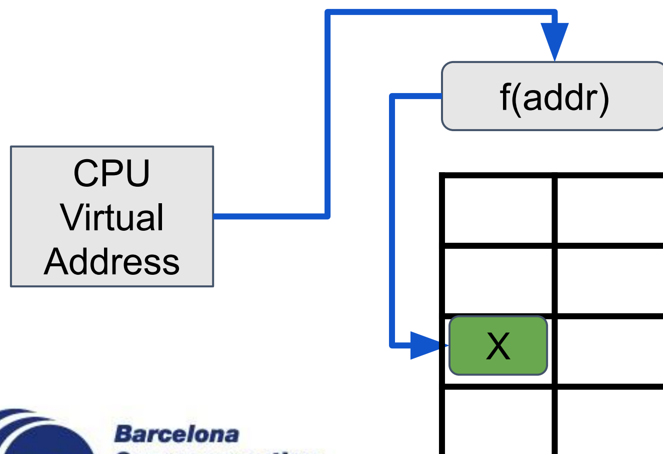
Virtual Addr	Physical Addr
0x0000	0x3000
...	...

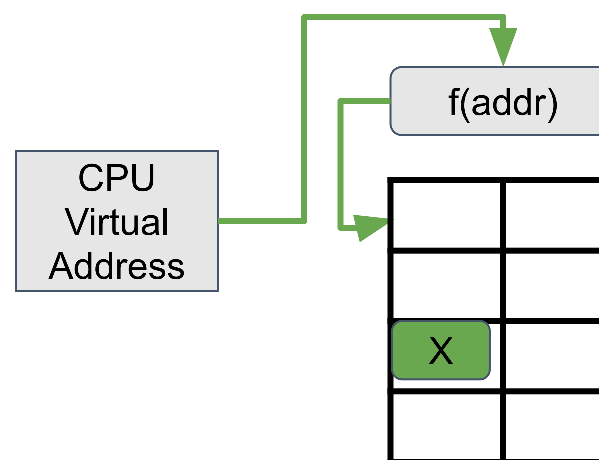
Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Proc A: sd X -> 0x0001

Proc B: ld 0x1001 -> r1





Miss

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

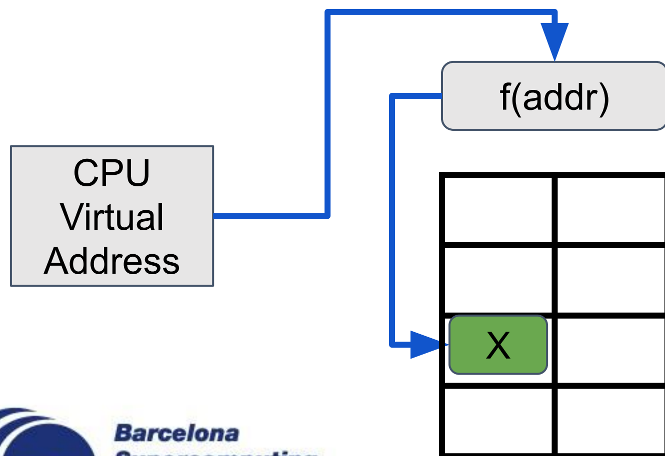
Page Table B

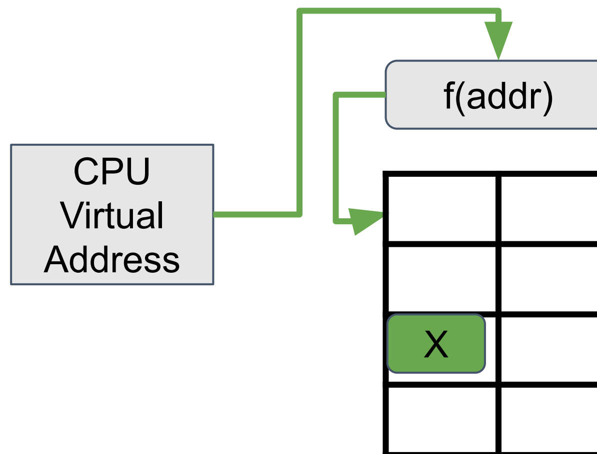
Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Proc A: sd X -> 0x0001

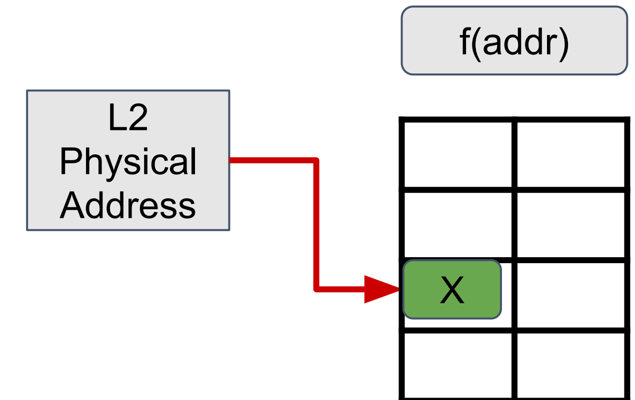
Proc B: ld 0x1001 -> r1

Coherency protocol access to addr 0x3001





Miss



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

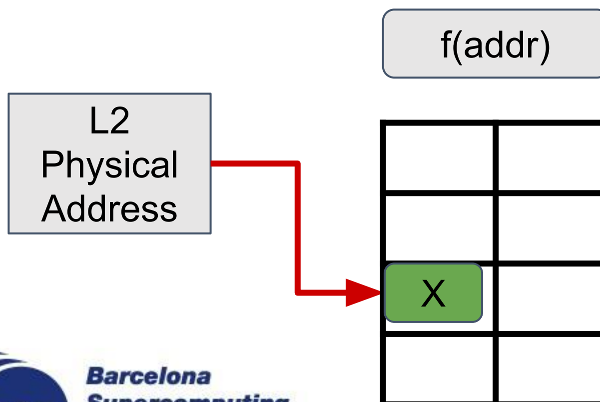
Virtual Addr	Physical Addr
0x0000	0x3000
...	...

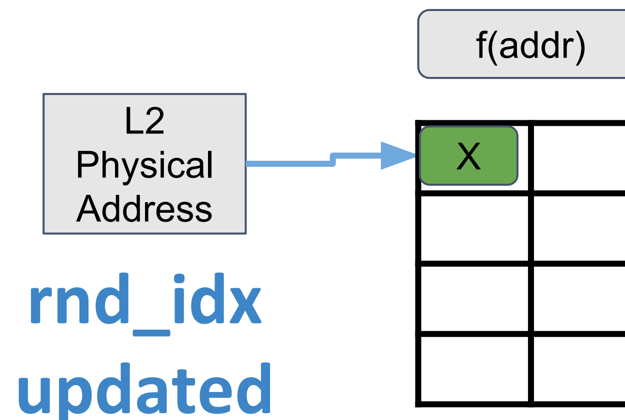
Page Table B

Virtual Addr	Physical Addr
0x1000	0x3000
...	...

Coherency protocol invalidating addr 0x3001

Coherency protocol provides X





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Example: Shared data

- Two processes A and B
  - Two different Page Tables
  - Shares data on 0x3000
  - First level caches are VIPT

Page Table A

Virtual Addr	Physical Addr
0x0000	0x3000
...	...

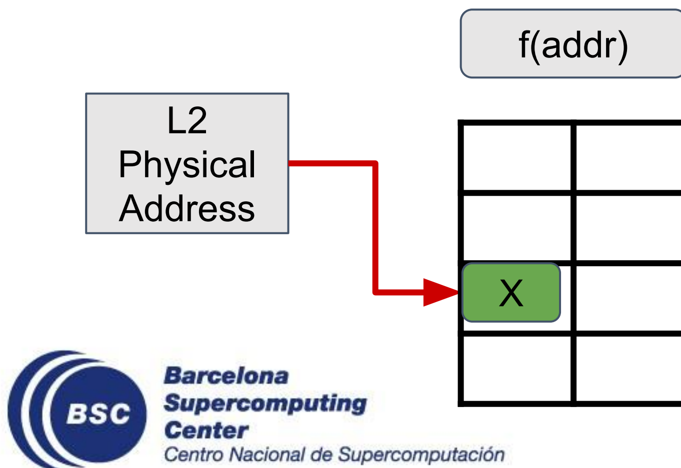
Page Table B

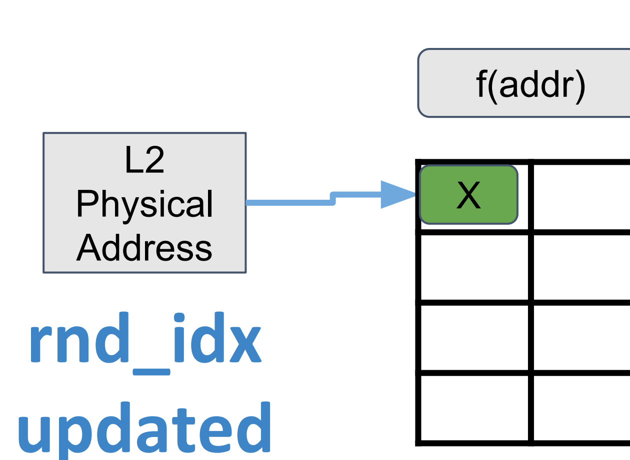
Virtual Addr	Physical Addr
0x1000	0x3000
...	...

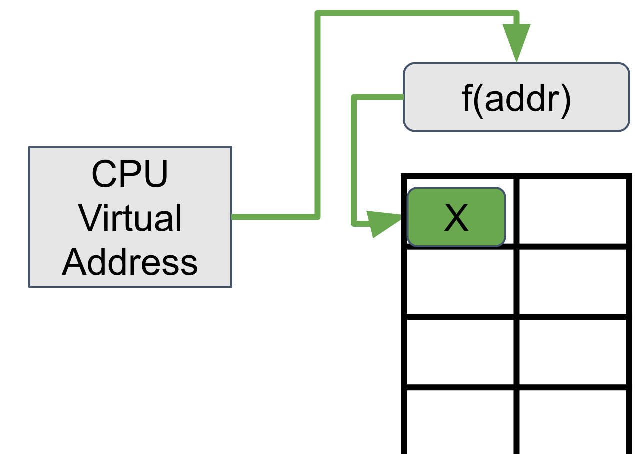
Coherency protocol invalidating addr 0x3004

Coherency protocol provides X

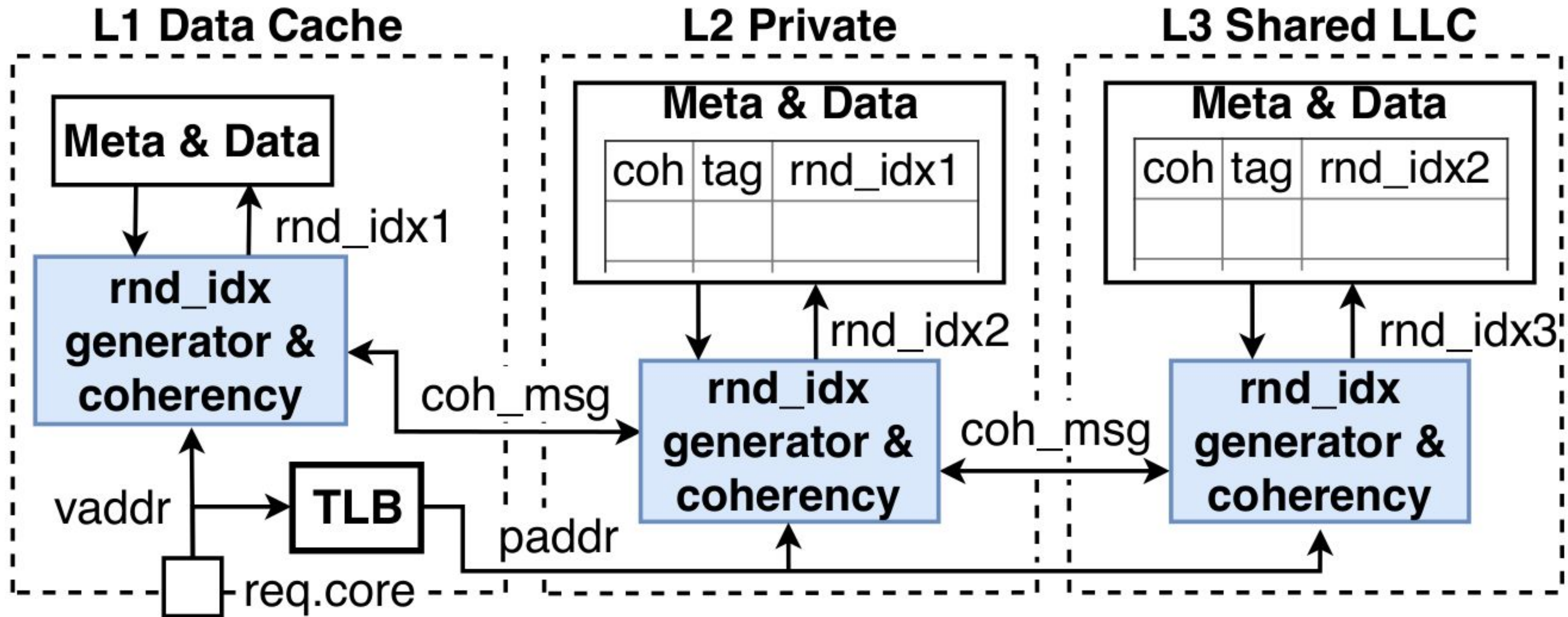
Proc B: ld 0x1001 -> r1







Example of a Three Level Cache Hierarchy



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

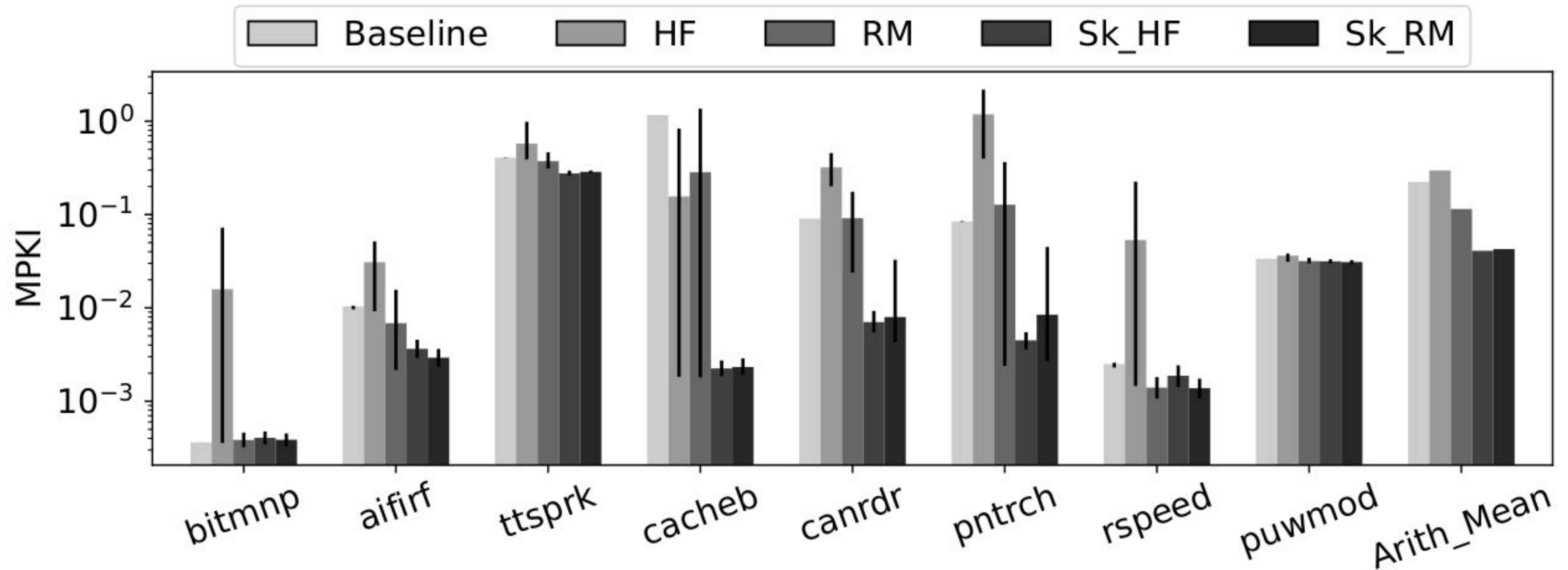
Implementation on a RISC-V Core

We have implemented this mechanism in the lowRISC SoC.

- There are two different randomizers on the first level cache .
  - Hash function and Random modulo.
- L2 incorporates the directory which track the L1 Blocks .
- Both caches have been augmented with tag array extensions to handle collisions produced by the randomizers.
- The Coherency protocol has been modified.
  - Able to issue probe requests using the random index stored.

Performance Evaluation

- We used the non-floating point benchmarks from the EEMBC suite.
  - 1000 iterations with 1000 different randomized keys.
- The hash function version has a very small impact on performance.
  - Other configurations increase the performance in this benchmarks.



Security Evaluation

- NIST STS testing proves uniform set distribution.
- Non-linear randomization function.
  - Thwarts linear cryptanalysis attacks.
- Security vulnerability analysis based on the cost of attack calibration

Processor	L1	L1 (skewed)	L2	L2(skewed)
Rocket	4288	68608	35456	2269184
Neoverse	17152	274432	2269184	18153472
Skylake	17728	1134592	67584	1081344

Number of attacker accesses to build eviction set

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Resources Evaluation

FPGA resources utilization for different configurations of the caches

		LUTs	FF	CLAs
L1	Baseline	3249	2514	82
	HF	4587 (+41.2%)	2598 (+3.3%)	87 (+6.1%)
	RM	3553 (+6.0%)	2598 (+3.3%)	87 (+6.1%)
	HF Skewed	7862 (+142.0%)	2676 (+6.4%)	87 (+6.1%)
	RM Skewed	3718 (+14.4%)	2676 (+6.4%)	87 (+6.1%)
L2	Baseline	11047	3778	85
	Others	13607 (23.2%)	3999 (+5.8%)	93 (+9.4%)
Total	Baseline	15301	7636	199
	HF	19199 (+25.5%)	7941 (+4.0%)	212 (+6.5%)
	RM	18055 (+18.0%)	7941 (+4.0%)	212 (+6.5%)
	HF Skewed	22474 (+46.9%)	8019 (+5.0%)	212 (+6.5%)
	RM Skewed	18330 (+25.5%)	8019 (+5.0%)	212 (+6.5%)

- The HF has a higher cost.
- In the RM case, randomization module consumes very few resources.

Conclusions

- Novel randomization mechanism for the whole cache hierarchy.
- Enables the use of virtual and physical addresses.
- Maintains cache coherency.
- Has a small impact on performance and consumed resources.
- We achieved integration into a RISC-V processor capable to boot Linux.
- Achieved increased security against cache-based side-channel attacks.

Future work

- Analyze implications and implementation of more complex coherence protocols.
- Implement our proposal in a complex processor design.
- Enable the utilization of multiple security domains.

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Thank you

max.doblas@bsc.es