BEN GRAS/@BJG, KAVEH RAZAVI, CRISTIANO GIUFFRIDA, HERBERT BOS
VRIJE UNIVERSITEIT AMSTERDAM

BLACKHAT USA 2018

SPYING ON YOUR NEIGHBOR: CPU CACHE ATTACKS AND BEYOND

ABOUT ME

ABOUT ME

- PhD student in VUsec VU University Research group

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security
- We do software hardening, exploitation

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security
- We do software hardening, exploitation
- Hardware attacks, side channels

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security
- We do software hardening, exploitation
- Hardware attacks, side channels
- Academic recognition but also hacker scene (Pwnies!)

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security
- We do software hardening, exploitation
- Hardware attacks, side channels
- Academic recognition but also hacker scene (Pwnies!)

ABOUT ME

- PhD student in VUsec VU University Research group
- Academic group researching systems software security
- We do software hardening, exploitation
- Hardware attacks, side channels
- Academic recognition but also hacker scene (Pwnies!)

OVERVIEW

- Side channels
- Cache attacks
- Cache defences
- Hyperthreading
- TLBleed
- Evaluation

SIDE CHANNELS

SIDE CHANNELS

SIDE CHANNELS

- Leak secrets outside the regular interface

SIDE CHANNELS

- Leak secrets outside the regular interface

RICH HISTORY - SMARTCARDS

- Power Consumption

  (FPGA Security by Shemal Shroff et al.)

- EM radiation: leak ECC bits

  (FPGA Security by Shemal Shroff et al.)

- Execution time: leak ECC, RSA bits

  (Timing Attacks on ECC by Shemal Shroff et al.)

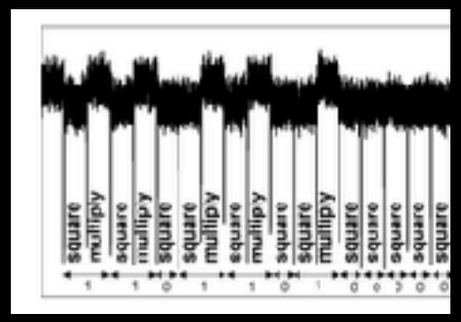
- Acoustic cryptanalysis

  (RSA Key Extraction [..] by Adi Shamir et al.)

RICH HISTORY - SMARTCARDS

- Power Consumption

  (FPGA Security by Shemal Shroff et al.)



- EM radiation: leak ECC bits

  (FPGA Security by Shemal Shroff et al.)

- Execution time: leak ECC, RSA bits

  (Timing Attacks on ECC by Shemal Shroff et al.)

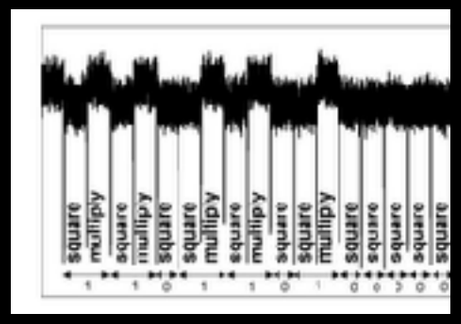
- Acoustic cryptanalysis

  (RSA Key Extraction [..] by Adi Shamir et al.)

RICH HISTORY - SMARTCARDS

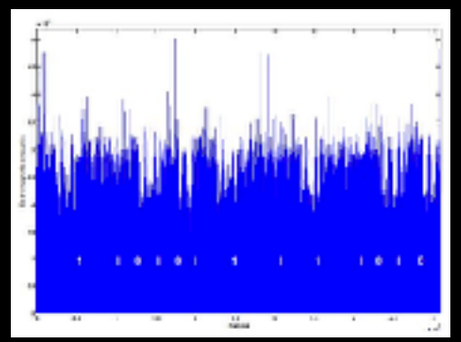
- Power Consumption

  (FPGA Security by Shemal Shroff et al.)



- EM radiation: leak ECC bits

  (FPGA Security by Shemal Shroff et al.)



- Execution time: leak ECC, RSA bits

  (Timing Attacks on ECC by Shemal Shroff et al.)

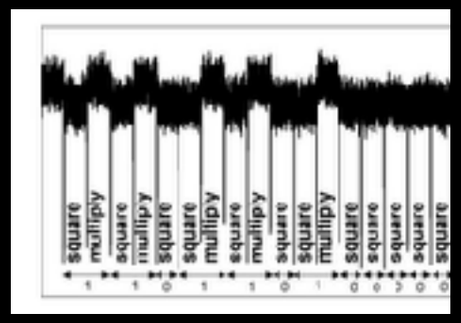
- Acoustic cryptanalysis

  (RSA Key Extraction [..] by Adi Shamir et al.)

RICH HISTORY - SMARTCARDS

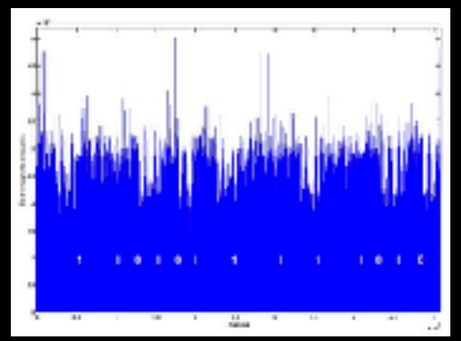
- Power Consumption

  (FPGA Security by Shemal Shroff et al.)



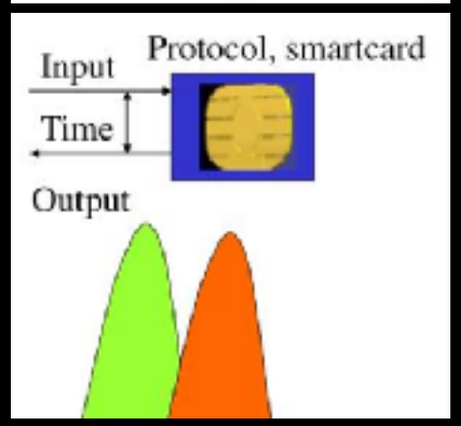
- EM radiation: leak ECC bits

  (FPGA Security by Shemal Shroff et al.)



- Execution time: leak ECC, RSA bits

  (Timing Attacks on ECC by Shemal Shroff et al.)



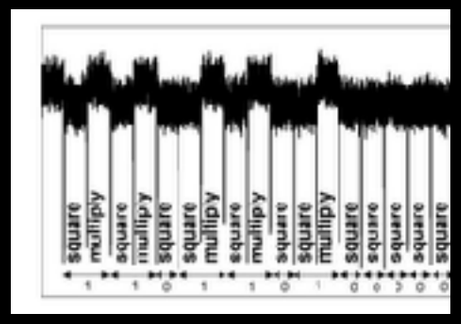
- Acoustic cryptanalysis

  (RSA Key Extraction [..] by Adi Shamir et al.)

RICH HISTORY - SMARTCARDS

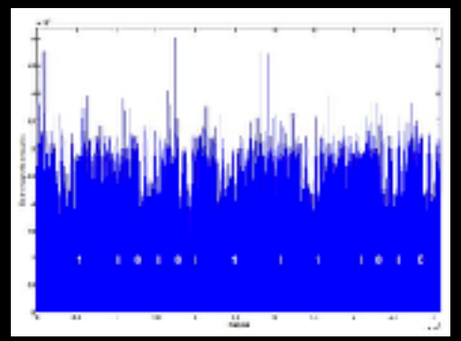
- Power Consumption

  (FPGA Security by Shemal Shroff et al.)



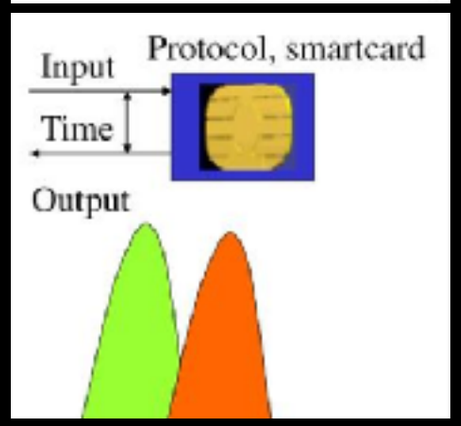
- EM radiation: leak ECC bits

  (FPGA Security by Shemal Shroff et al.)



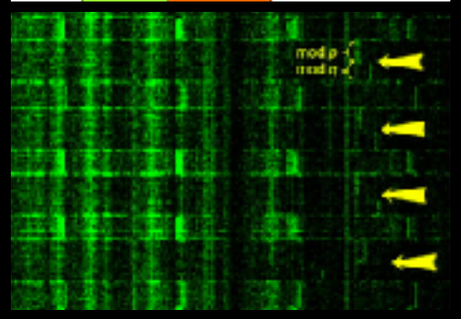
- Execution time: leak ECC, RSA bits

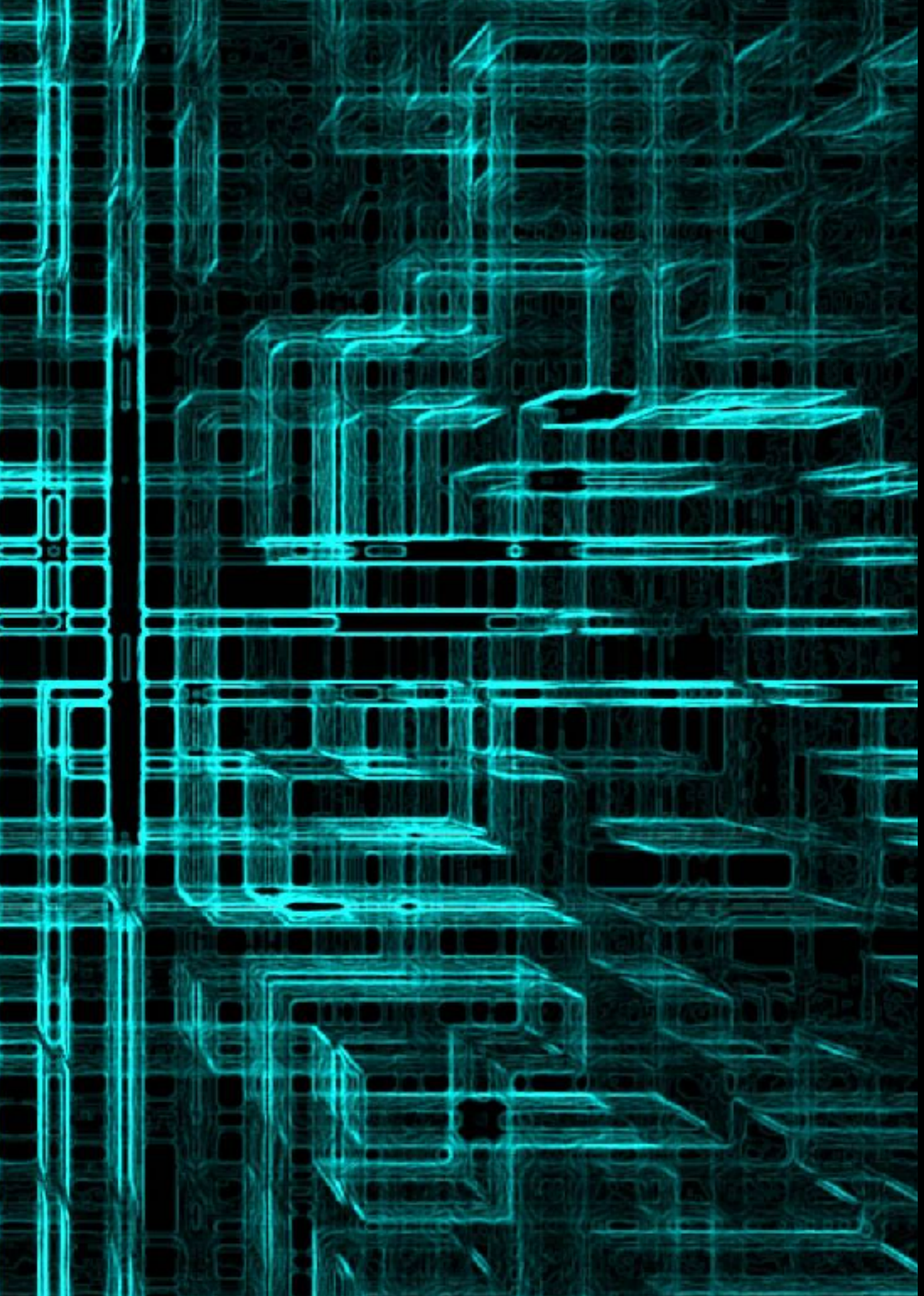
  (Timing Attacks on ECC by Shemal Shroff et al.)



- Acoustic cryptanalysis

  (RSA Key Extraction [..] by Adi Shamir et al.)





CACHE ATTACKS

CACHE: SOFTWARE EQUIVALENT

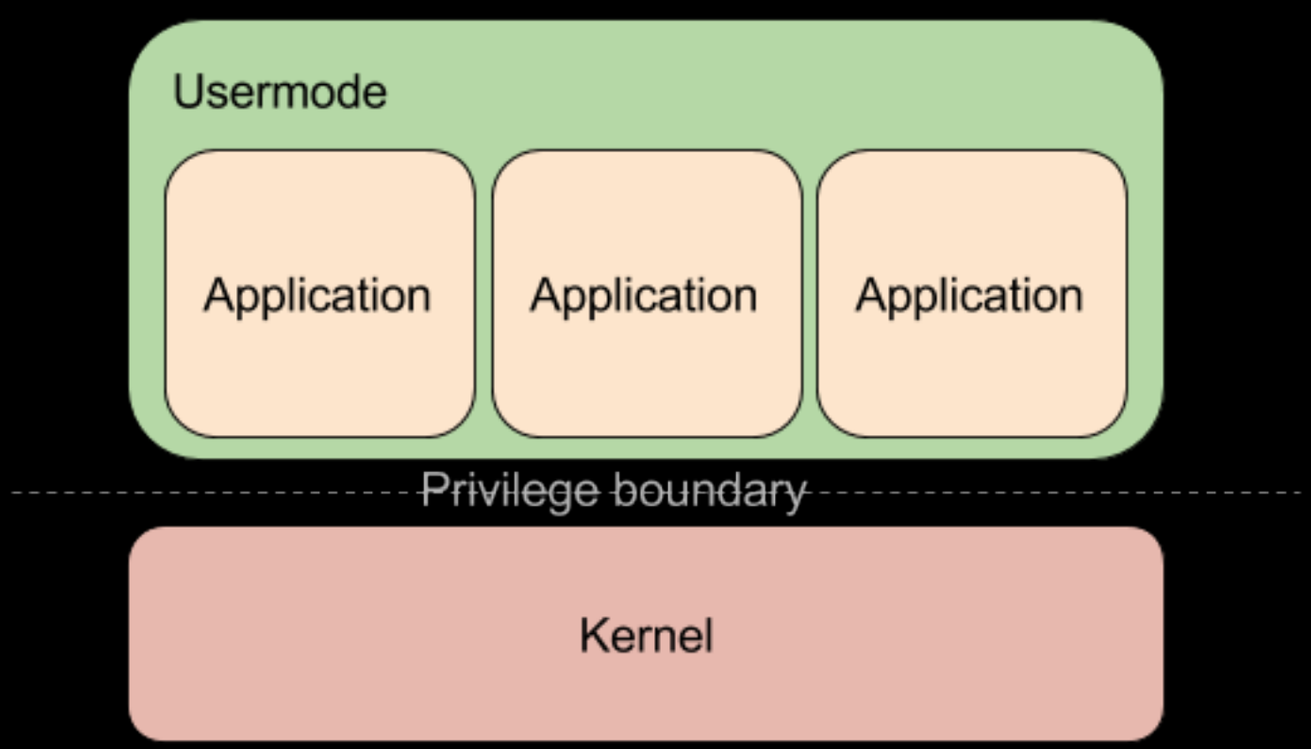
- Computing processes ought to be compartmented
- Different owners or privilege levels: *trust boundaries*

CACHE: SOFTWARE EQUIVALENT

- Computing processes ought to be compartmented
- Different owners or privilege levels: *trust boundaries*

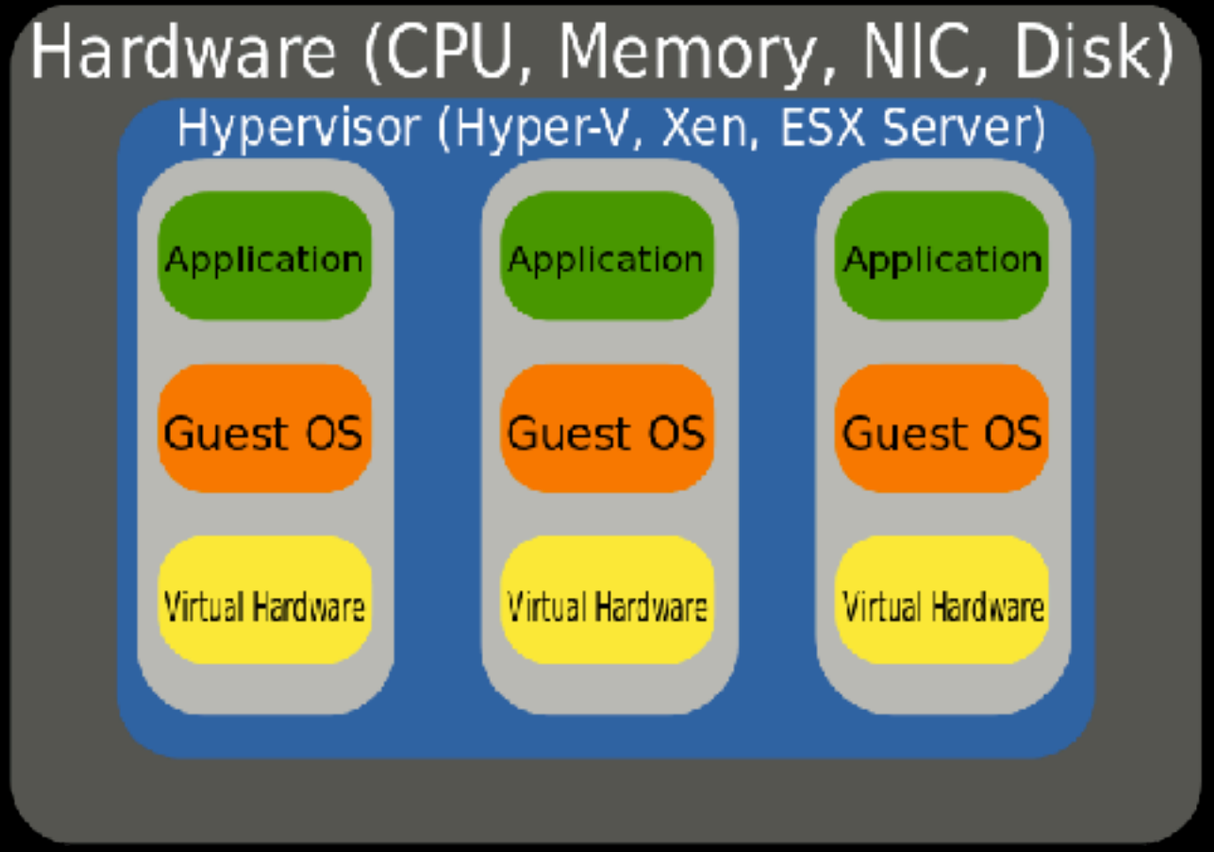
CACHE: SOFTWARE EQUIVALENT

- Computing processes ought to be compartmented
- Different owners or privilege levels: *trust boundaries*



CACHE: SOFTWARE EQUIVALENT

- Computing processes ought to be compartmented
- Different owners or privilege levels: *trust boundaries*



CACHE: SOFTWARE EQUIVALENT

- Computing processes ought to be compartmented
- Different owners or privilege levels: *trust boundaries*

CACHE: SOFTWARE EQUIVALENT

- There are **shared** resources between processes
- RAM, CPU cache, TLB, computational resources ..
- Practically always: allows signaling: covert channel
- Sometimes: allows side channel (spying)

CROSS-PROCESS SHARED STATE

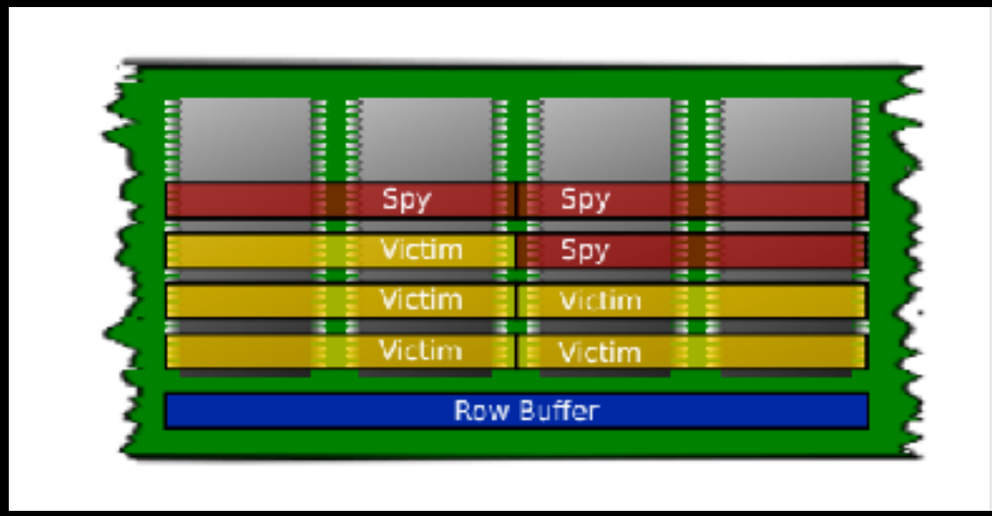
- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)

- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)

- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

CROSS-PROCESS SHARED STATE

- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)

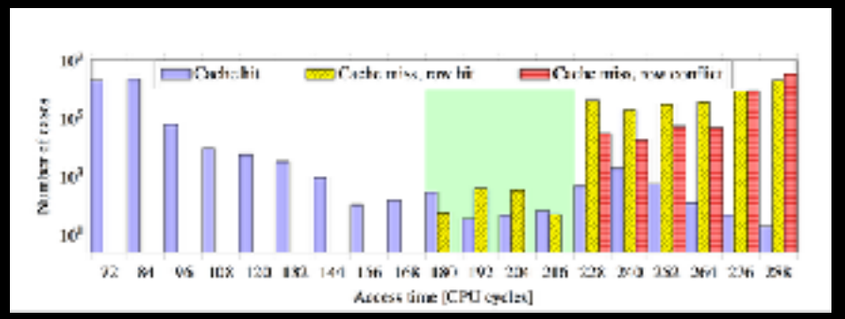


- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)

- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

CROSS-PROCESS SHARED STATE

- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)

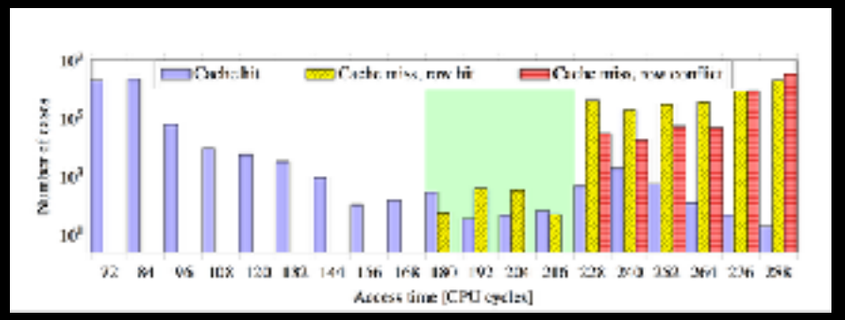


- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)

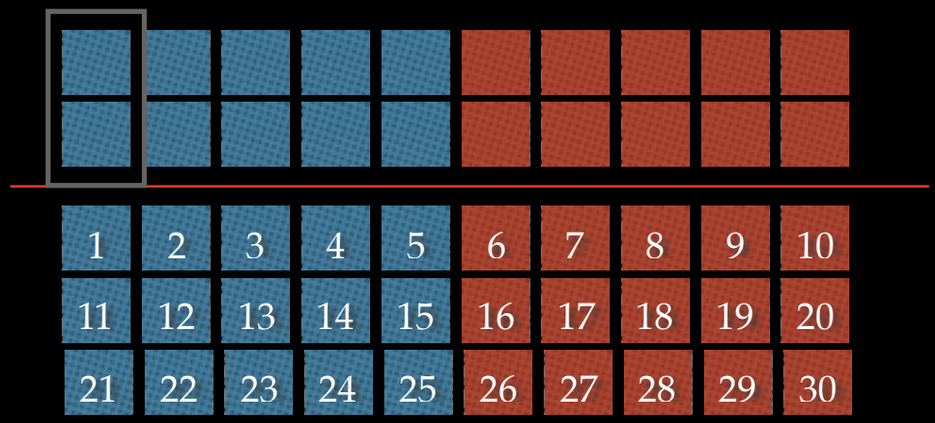
- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

CROSS-PROCESS SHARED STATE

- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)



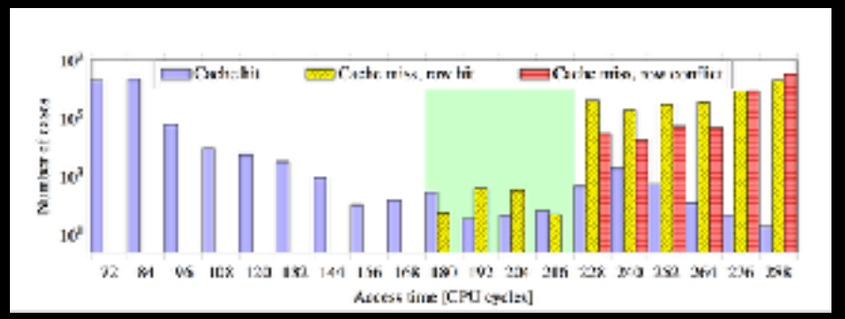
- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al. shown, many others exist)



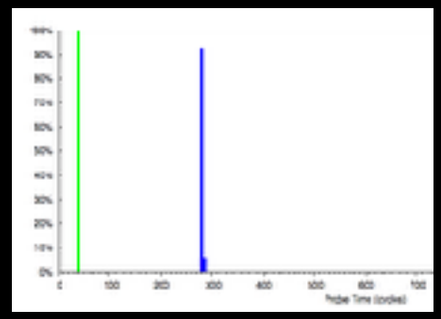
- Cache prefetch
  (Prefetch Side-Channel Attacks, by Daniel Gruss et al.)

CROSS-PROCESS SHARED STATE

- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)



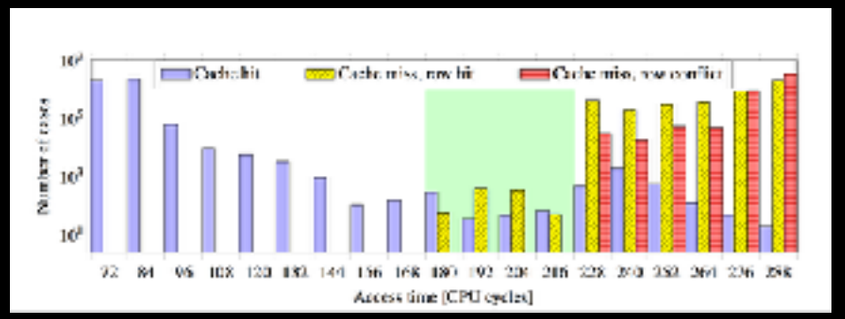
- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)



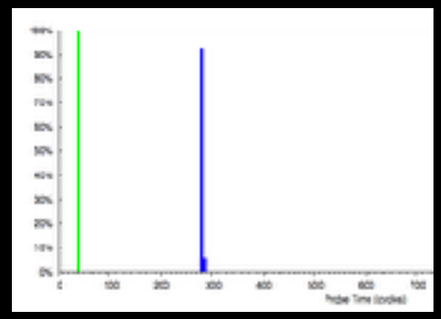
- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

CROSS-PROCESS SHARED STATE

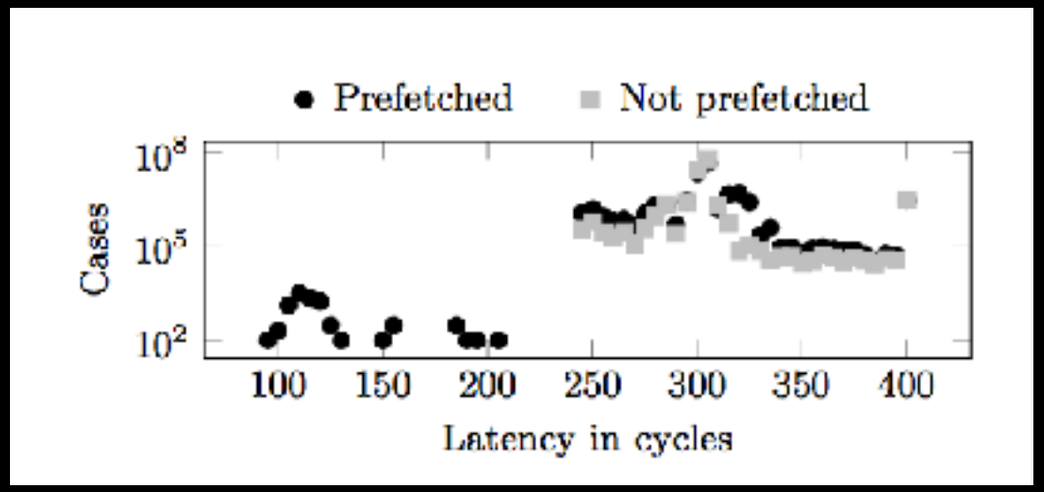
- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)



- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)

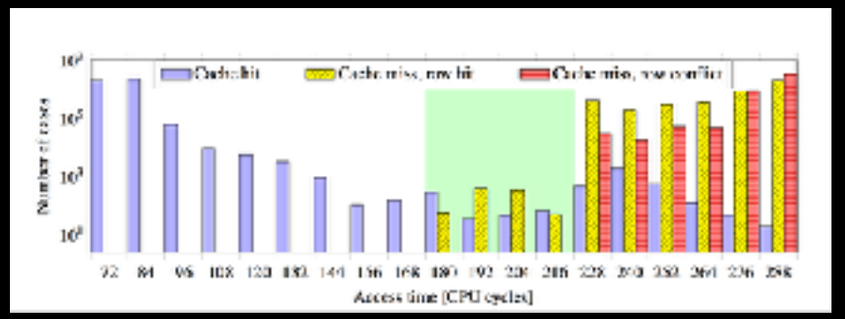


- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

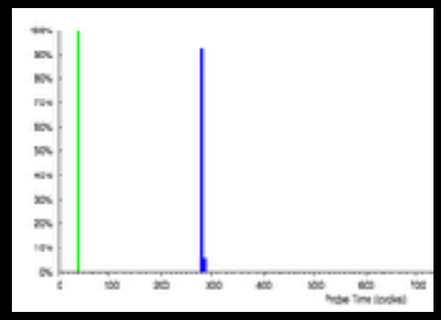


CROSS-PROCESS SHARED STATE

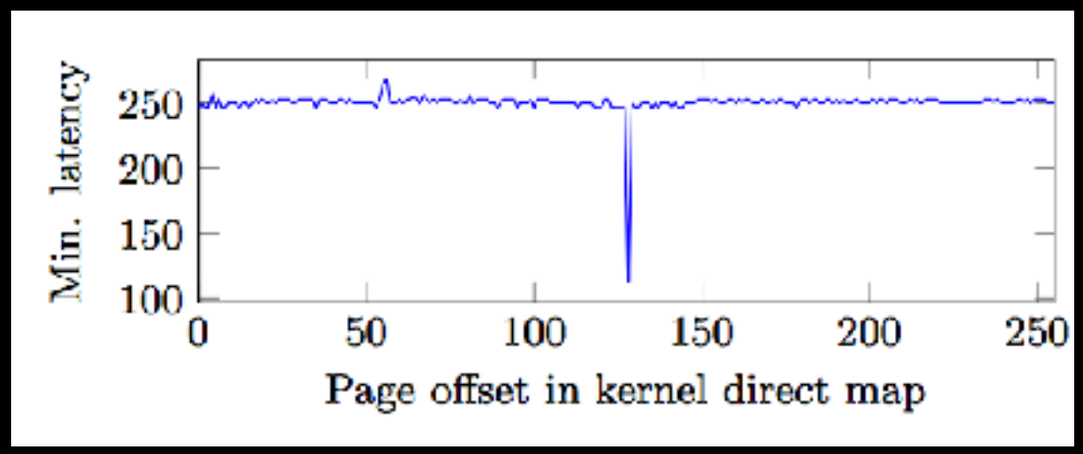
- Shared RAM row
  (DRAMA paper, by Peter Peßl et al.)



- Shared cache set
  (FLUSH+RELOAD by Yuval Yarom et al.
  shown, many others exist)



- Cache prefetch
  (Prefetch Side-Channel Attacks,
  by Daniel Gruss et al.)

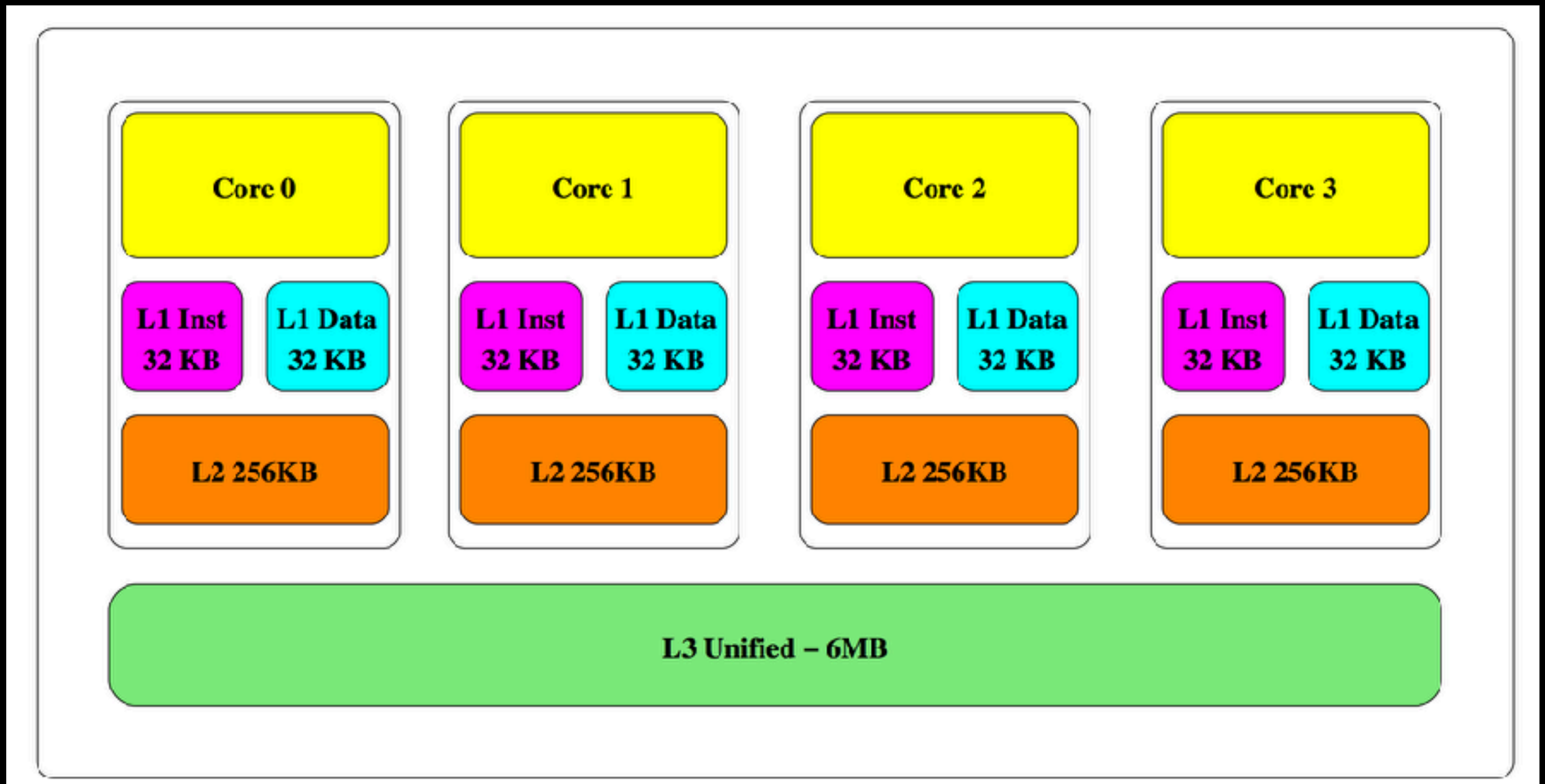


CROSS-PROCESS/VM SHARED STATE

- This is only possible because of shared resources

CROSS-PROCESS/VM SHARED STATE

- This is only possible because of shared resources



EXAMPLE: FLUSH+RELOAD

- One of several cache attacks
- Relies on shared memory
- Can be shared object (mmap()ed shared libraries)
- Or shared pages after deduplication (KSM in Linux)

EXAMPLE: FLUSH+RELOAD

- Work by Yuval Yarom, Katrina Falkner
- Memory access patterns can betray secrets
- Because access patterns frequently depend on secrets
- Example: RSA keys. *(n,e,d)* Private: *d*. *n=pq* and *d* are 1024 bits or more

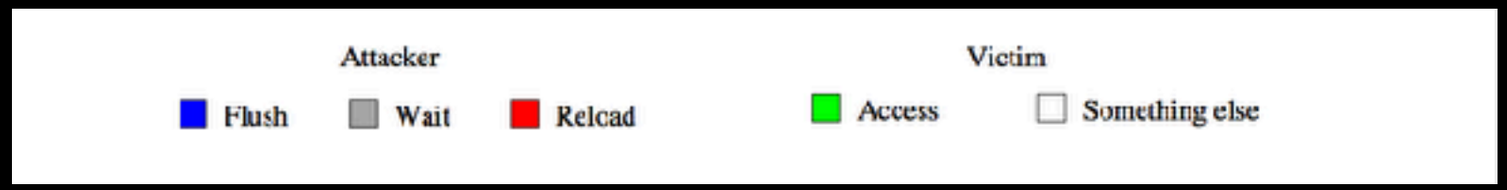
EXAMPLE: FLUSH+RELOAD

- Signing is: computing $m^d \pmod n$
- Often square-and-multiply **depending on** bits in *d*

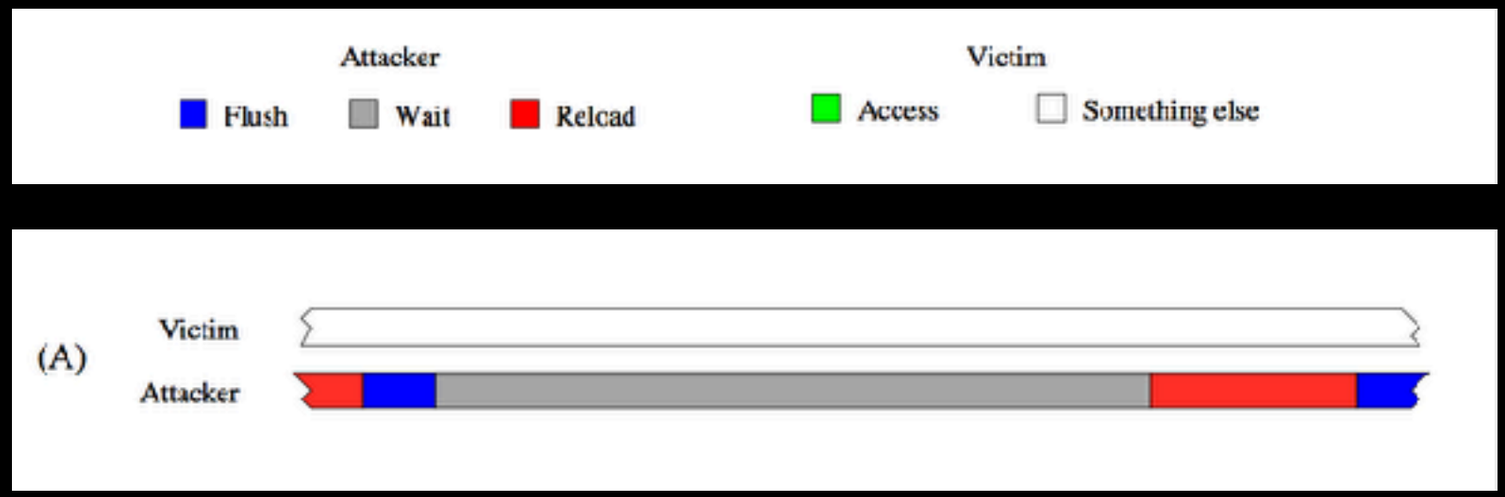
- Shared cache activity betrays memory access patterns
- Quickly probing the cache can betray the bits in *d*

EXAMPLE: FLUSH+RELOAD

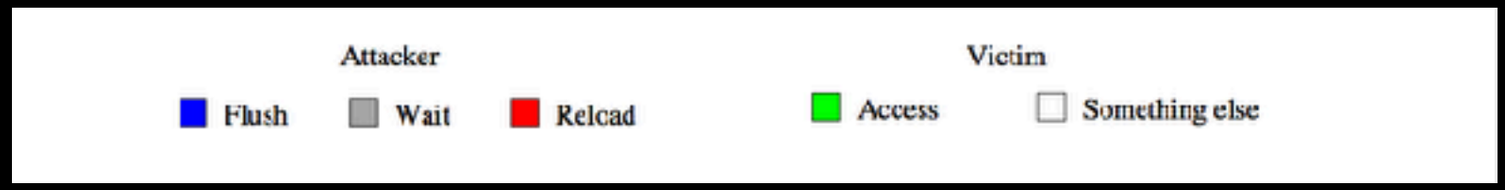
EXAMPLE: FLUSH+RELOAD

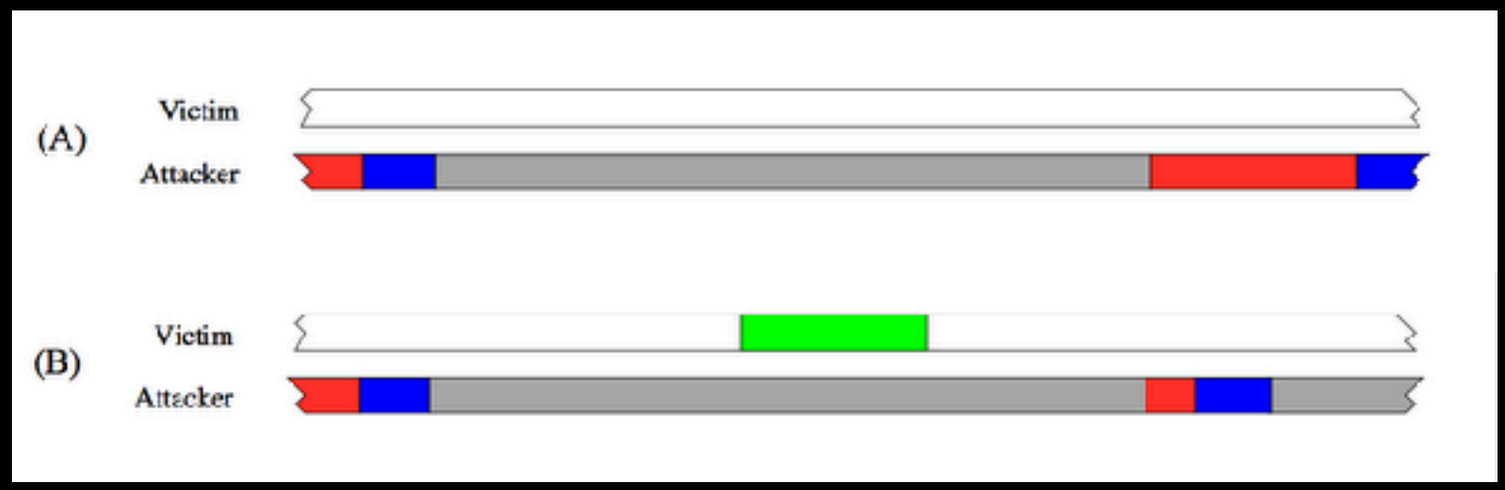


EXAMPLE: FLUSH+RELOAD

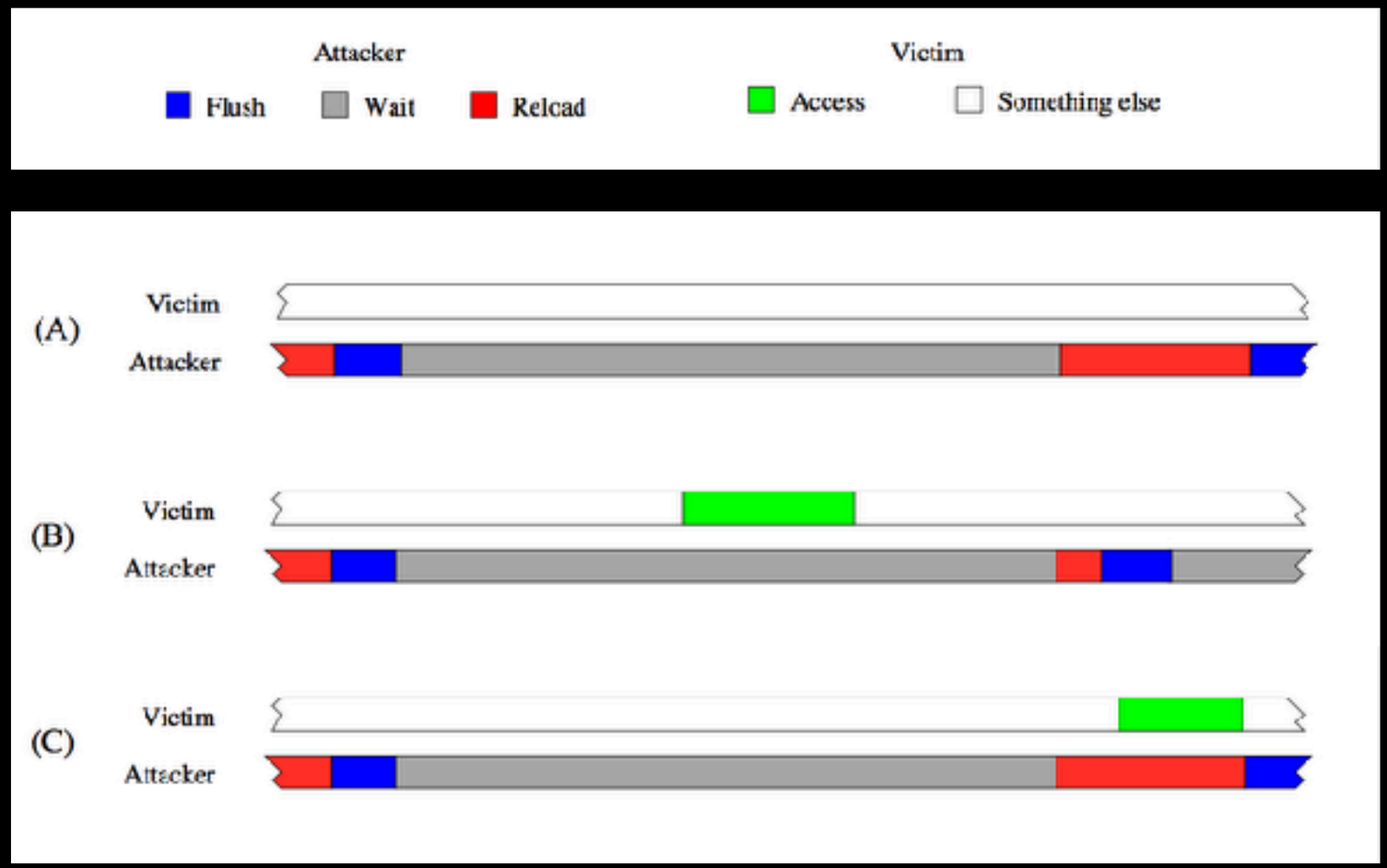


EXAMPLE: FLUSH+RELOAD

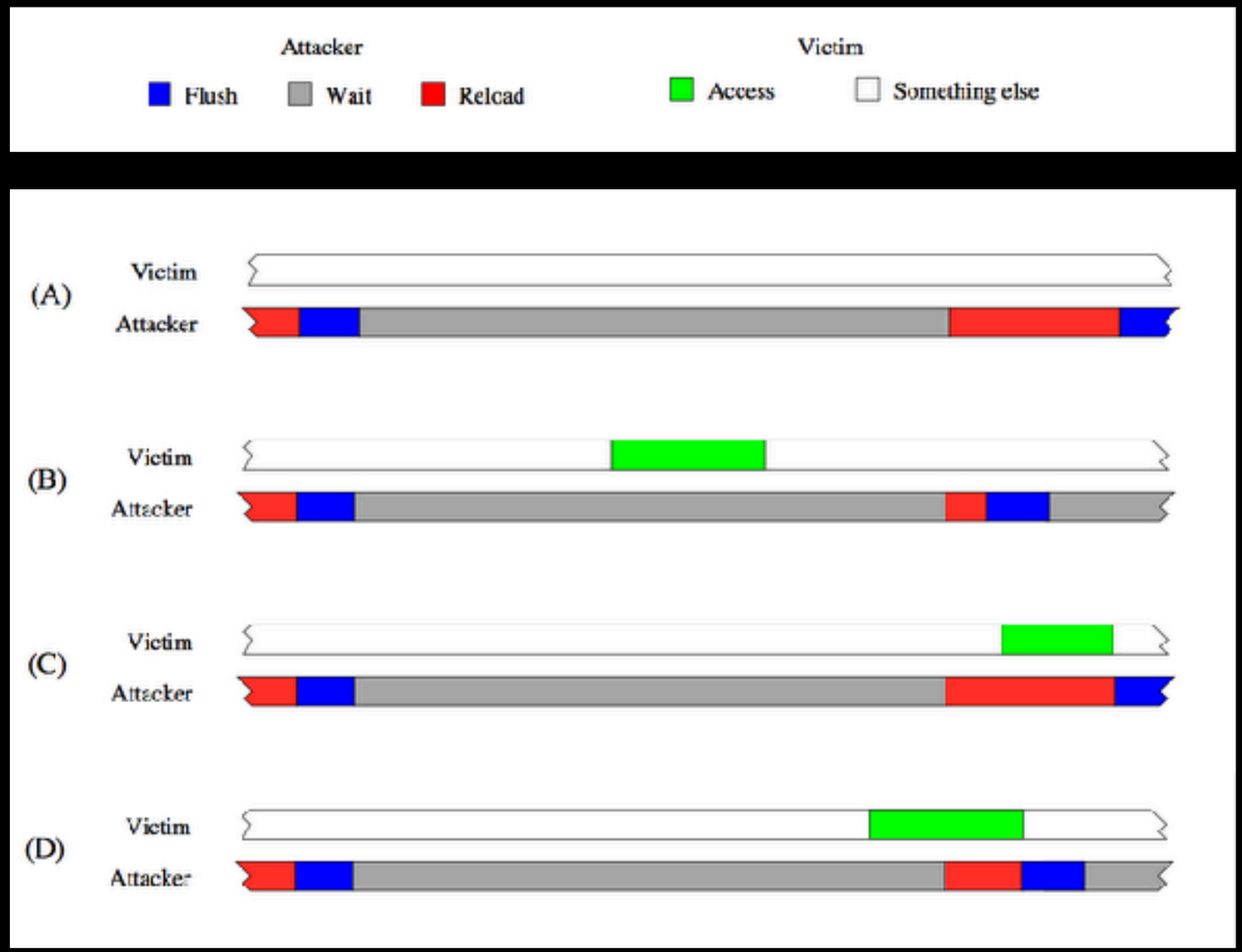




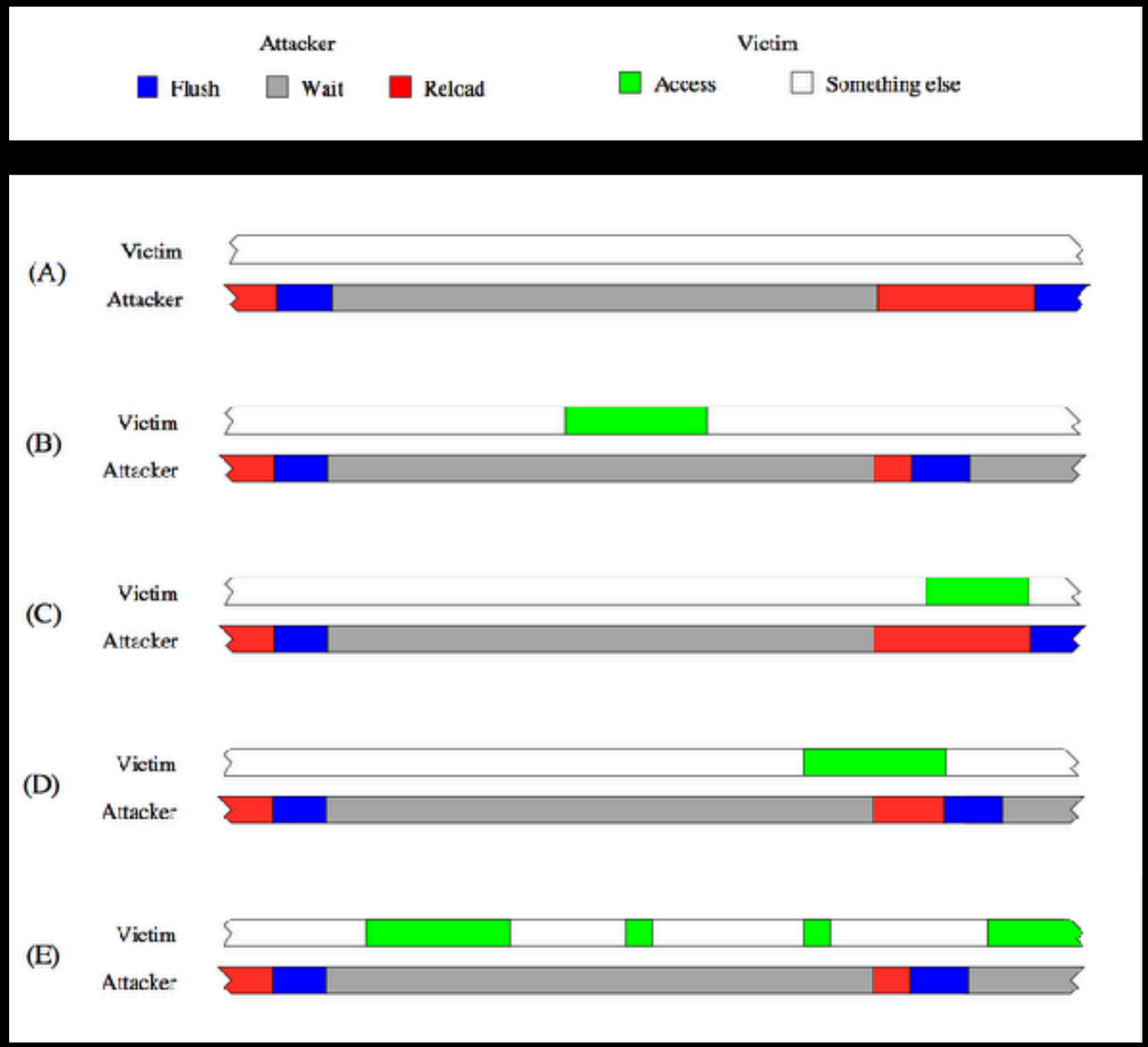
EXAMPLE: FLUSH+RELOAD



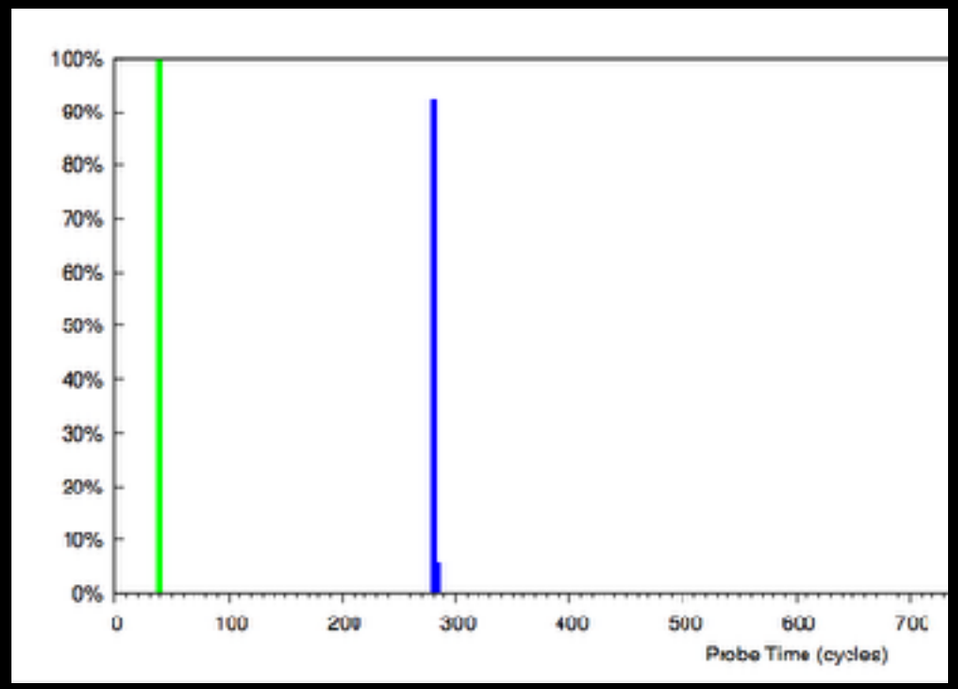
EXAMPLE: FLUSH+RELOAD



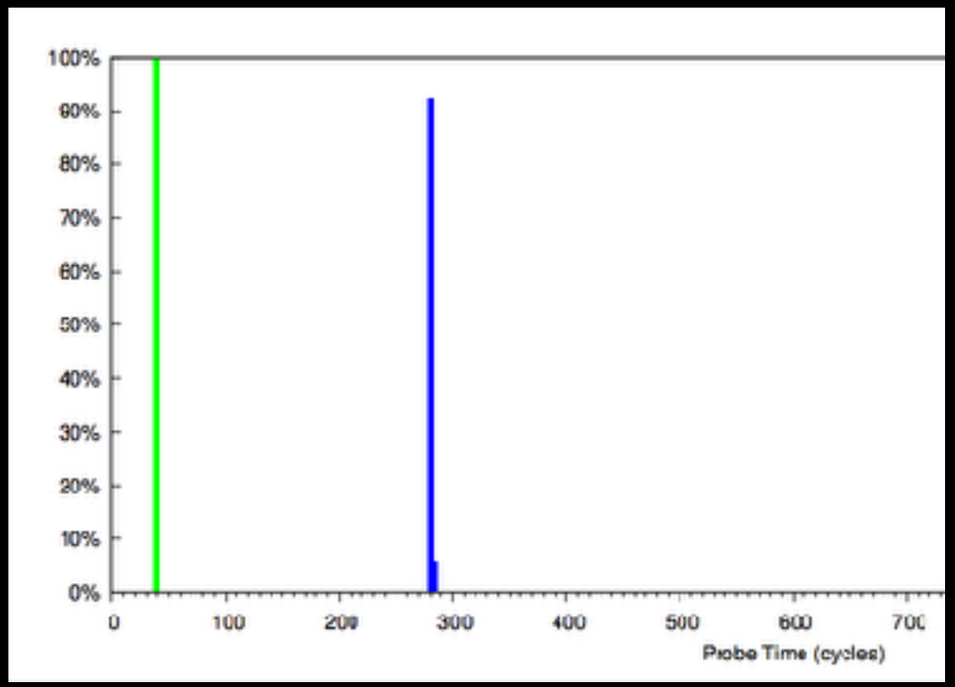
EXAMPLE: FLUSH+RELOAD



EXAMPLE: FLUSH+RELOAD



EXAMPLE: FLUSH+RELOAD



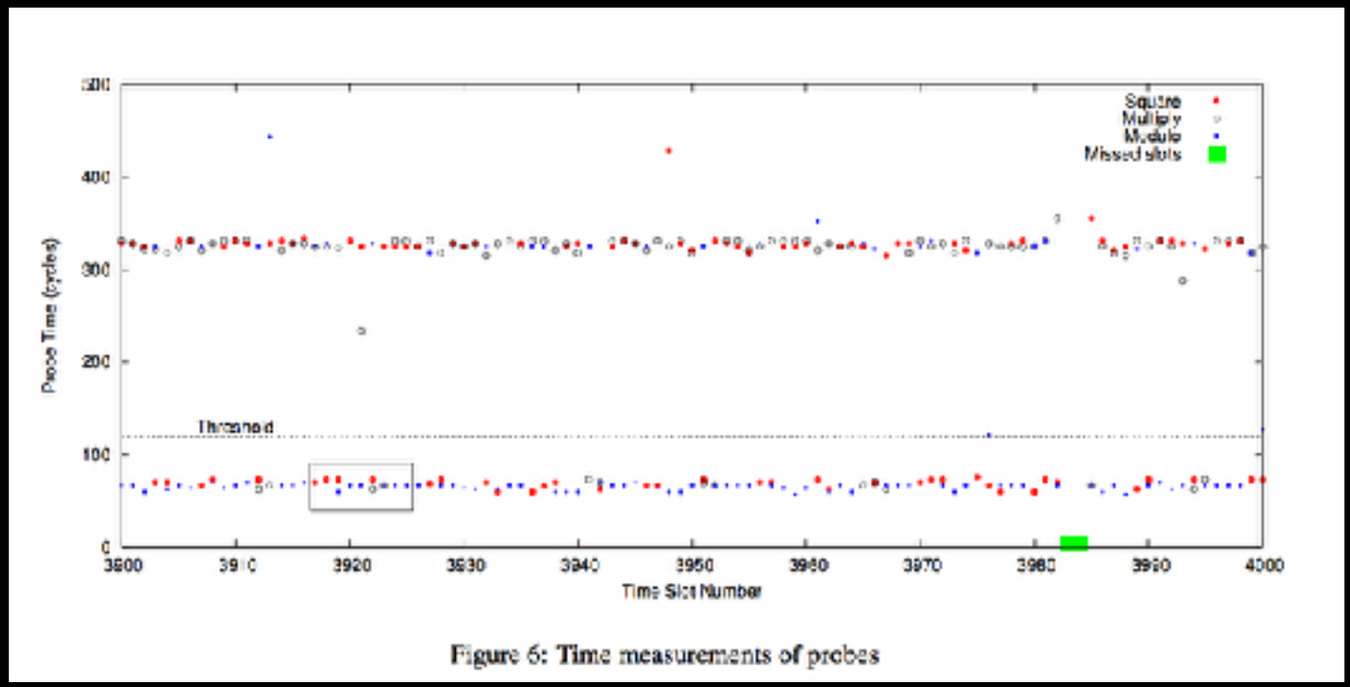
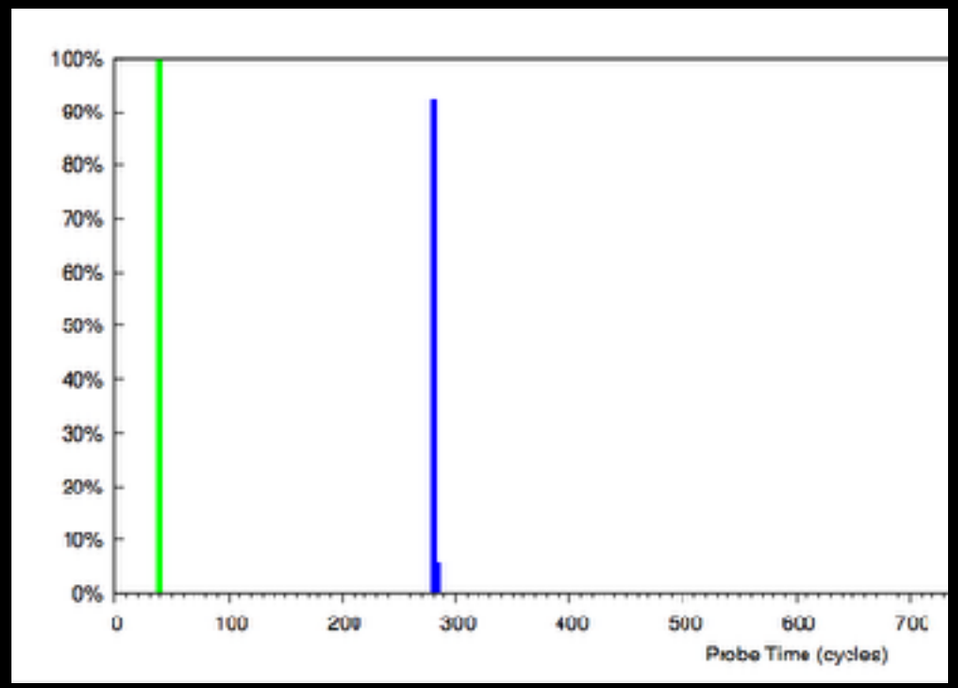


Figure 6: Time measurements of probes

EXAMPLE: FLUSH+RELOAD



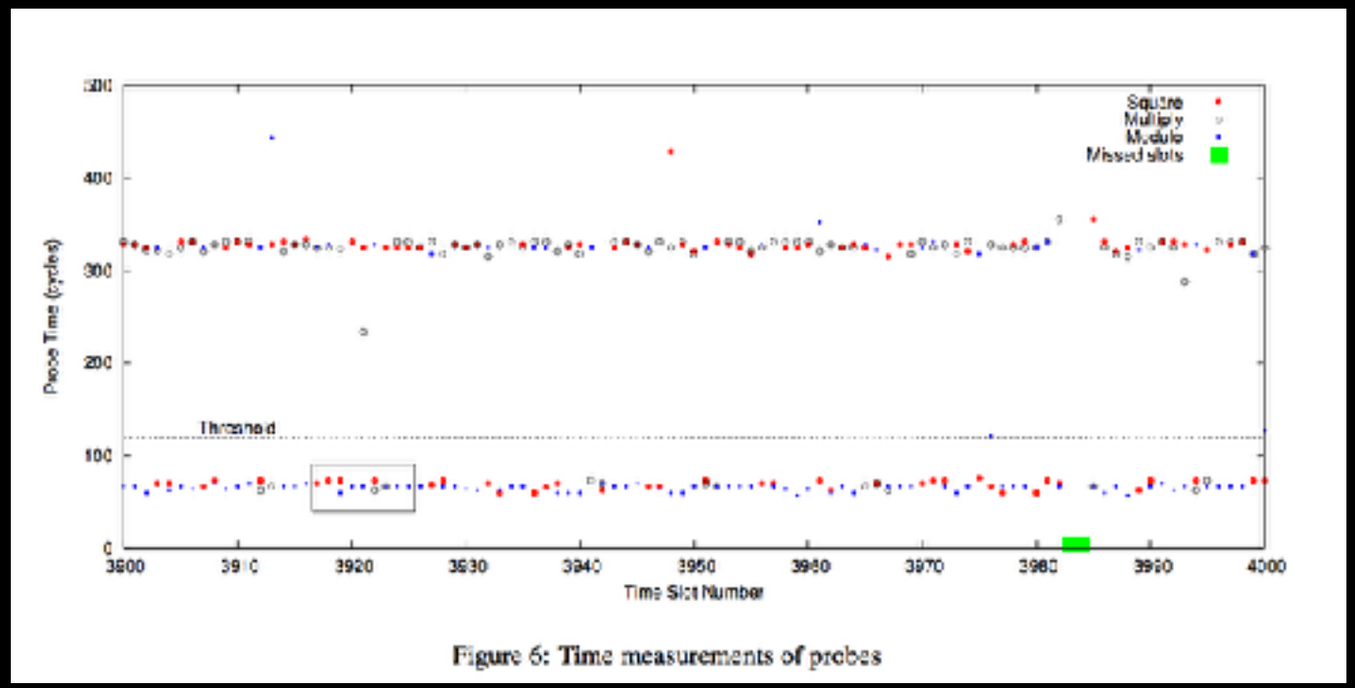
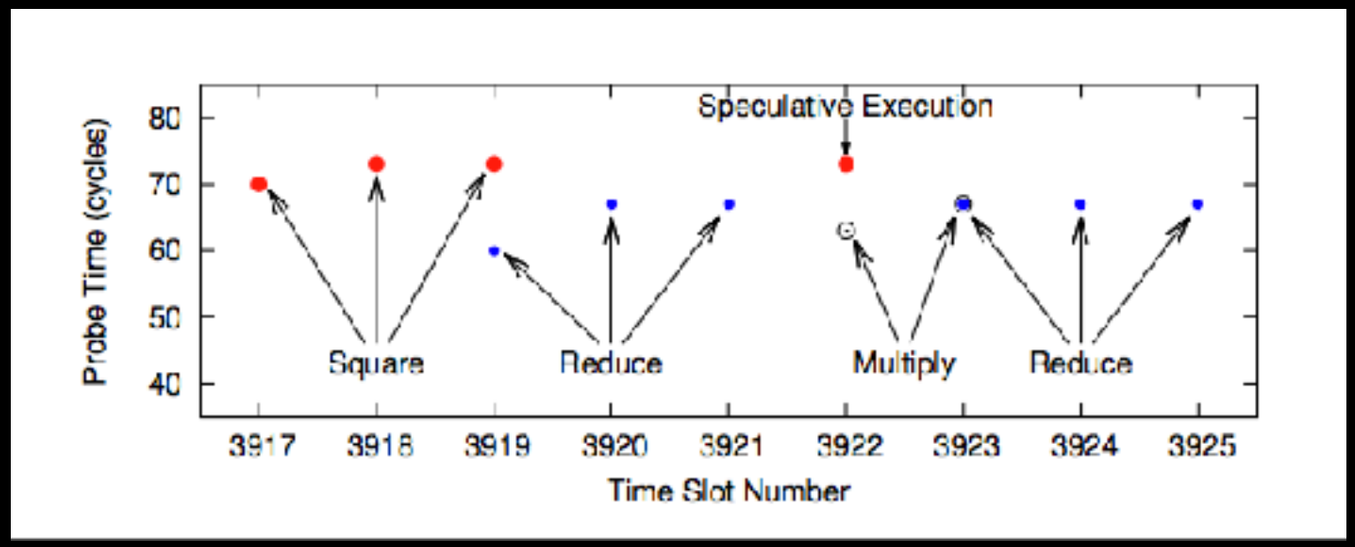
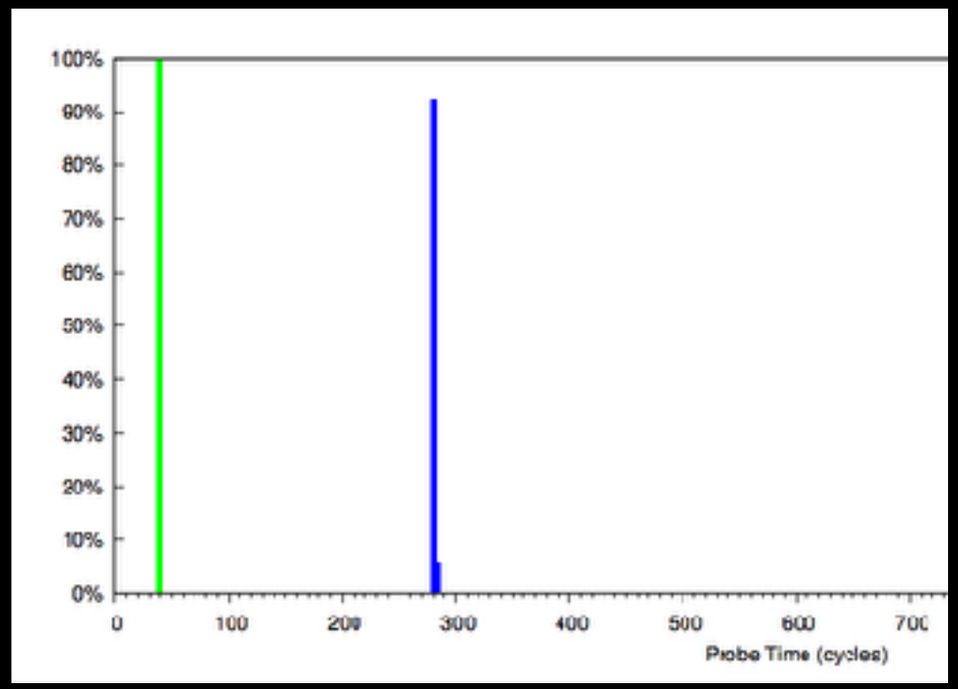


Figure 6: Time measurements of probes



EXAMPLE: FLUSH+RELOAD



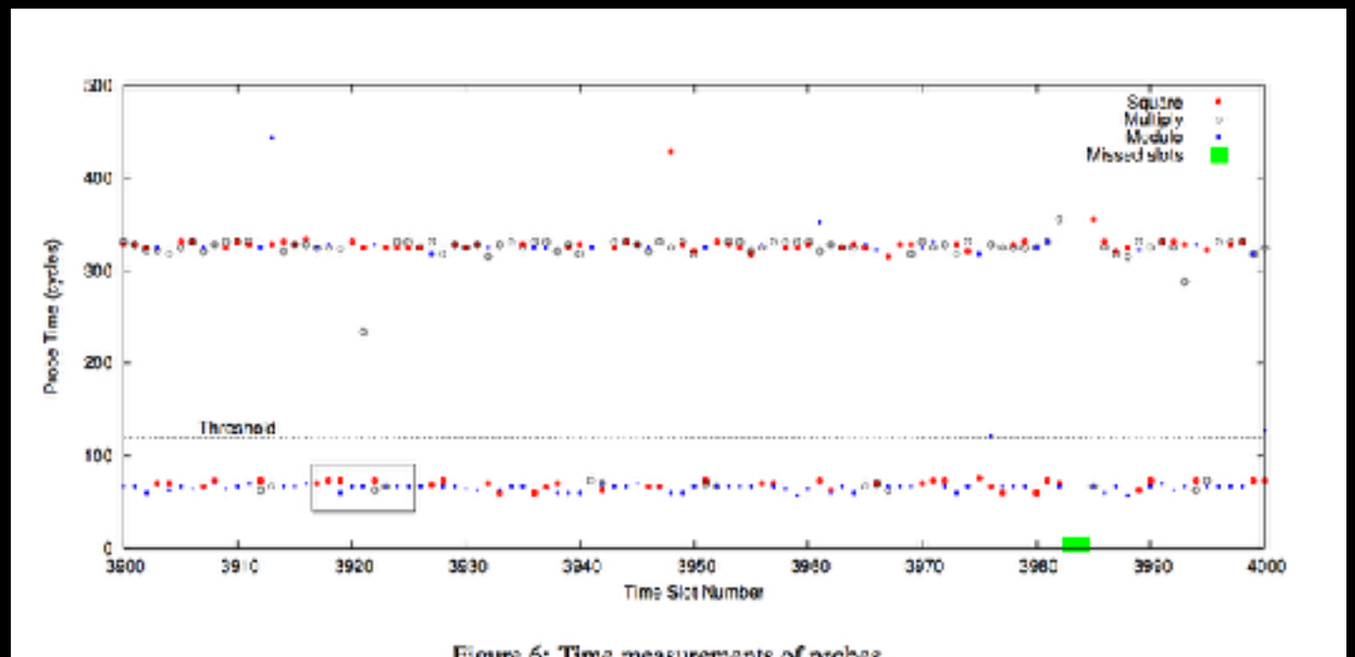
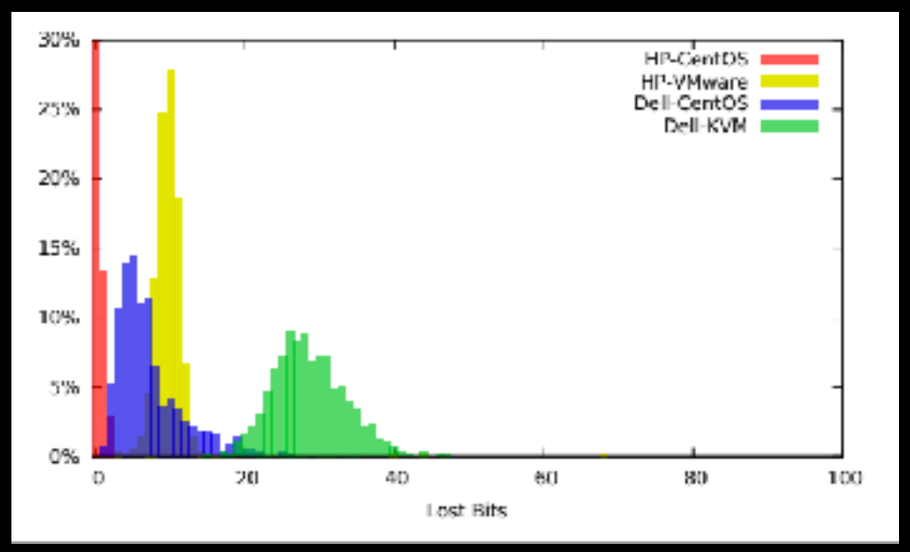
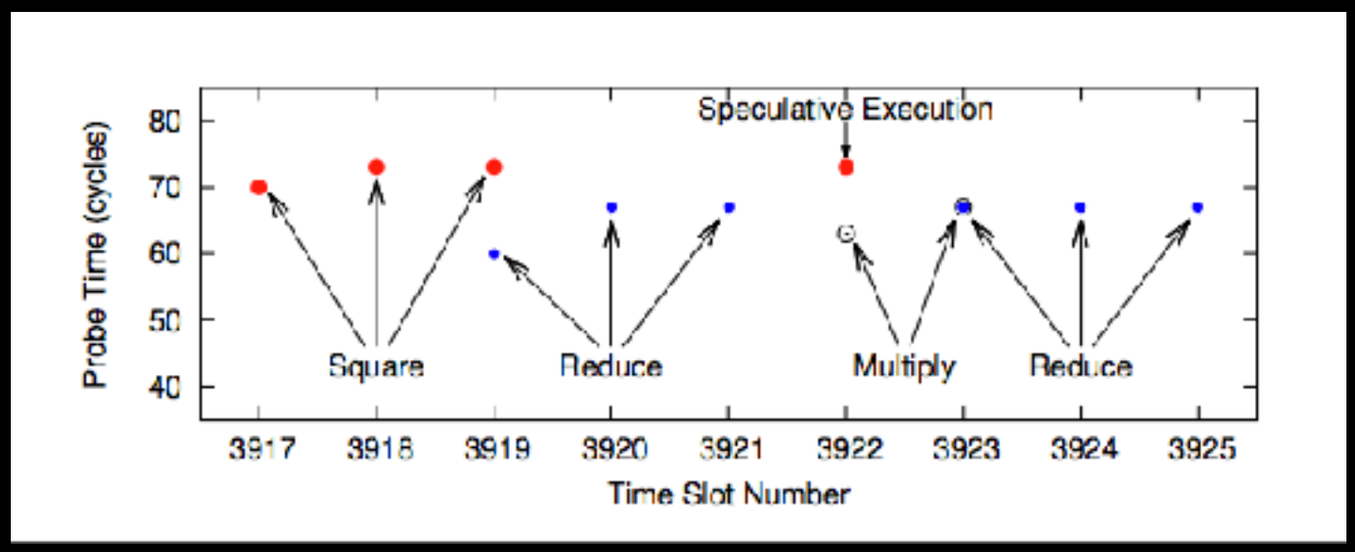


Figure 6: Time measurements of probes





EXAMPLE: FLUSH+RELOAD

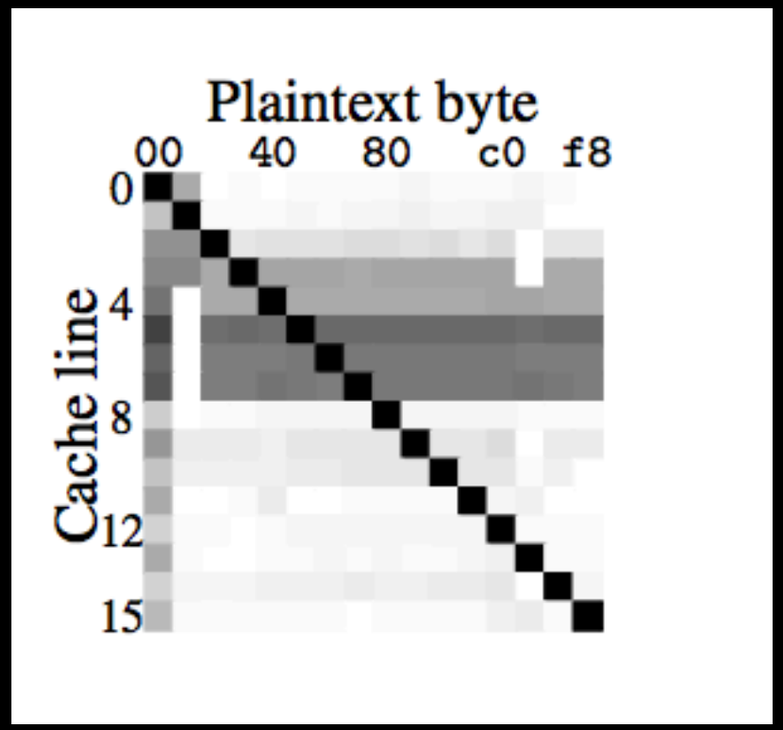
- Can also attack AES implementation with T tables
- A table lookup happens $T_j\,[x_i = p_i \oplus k_i\,]$
- p_i is a plaintext byte, k_i a key byte

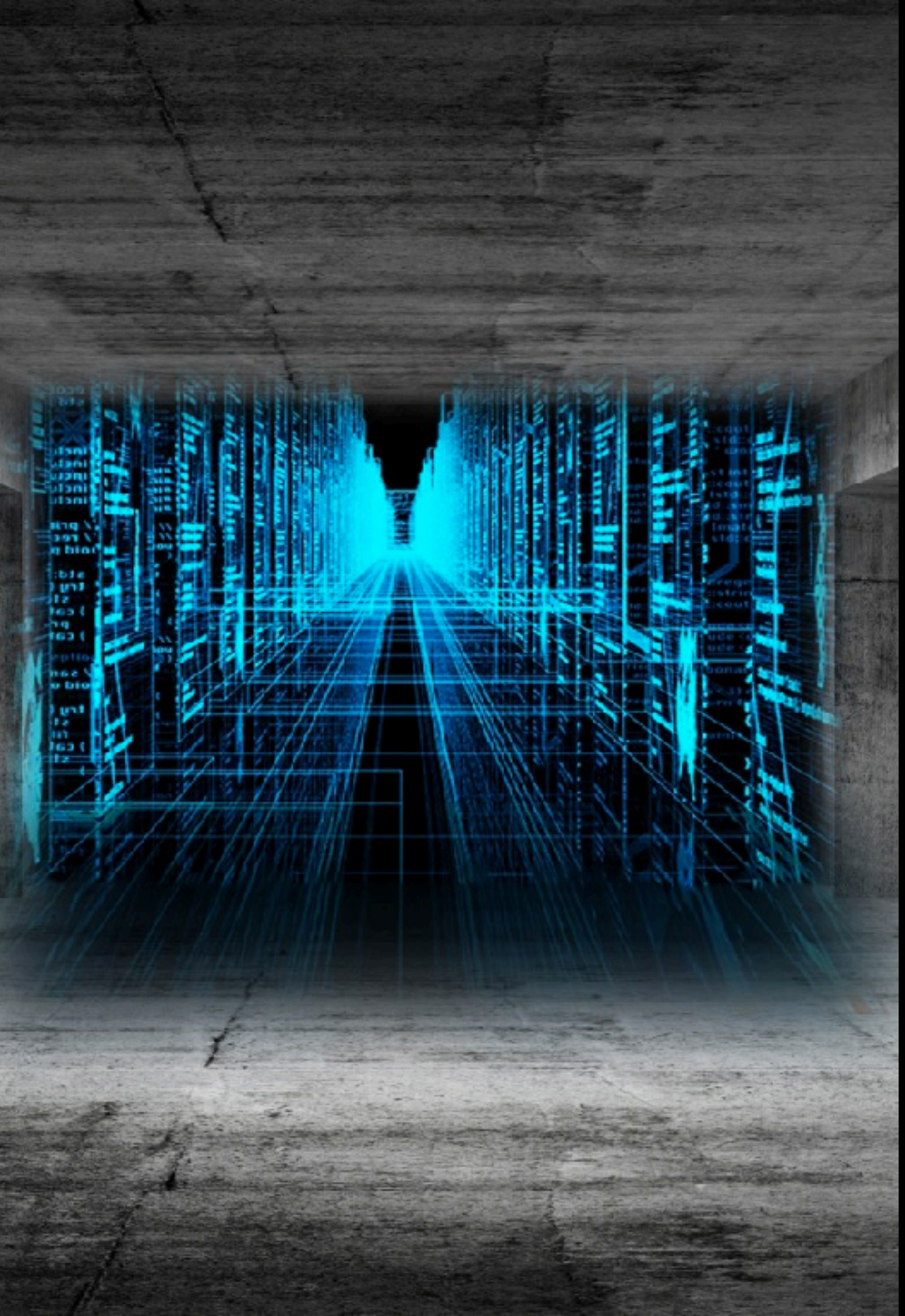
EXAMPLE: FLUSH+RELOAD

- Again: secrets are betrayed by memory accesses
- Known plaintext + accesses = key recovery

EXAMPLE: FLUSH+RELOAD

- Again: secrets are betrayed by memory accesses
- Known plaintext + accesses = key recovery





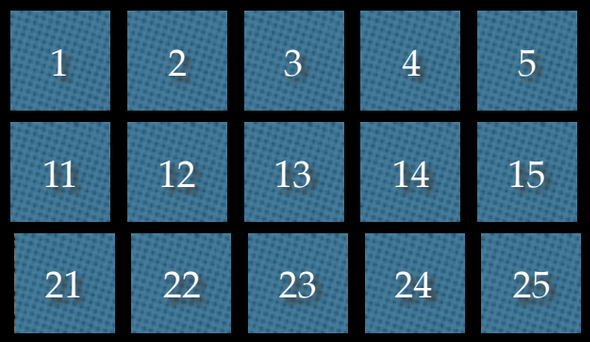
CACHE DEFENCES

CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this

CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this

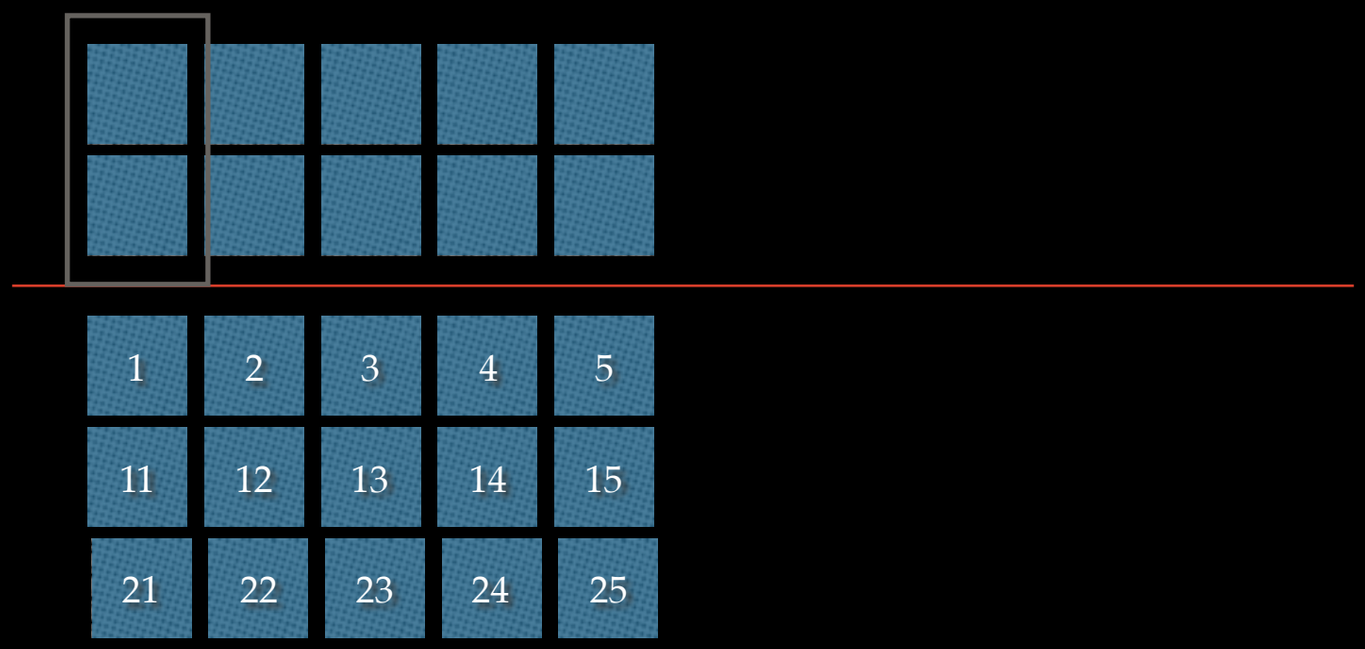


CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this

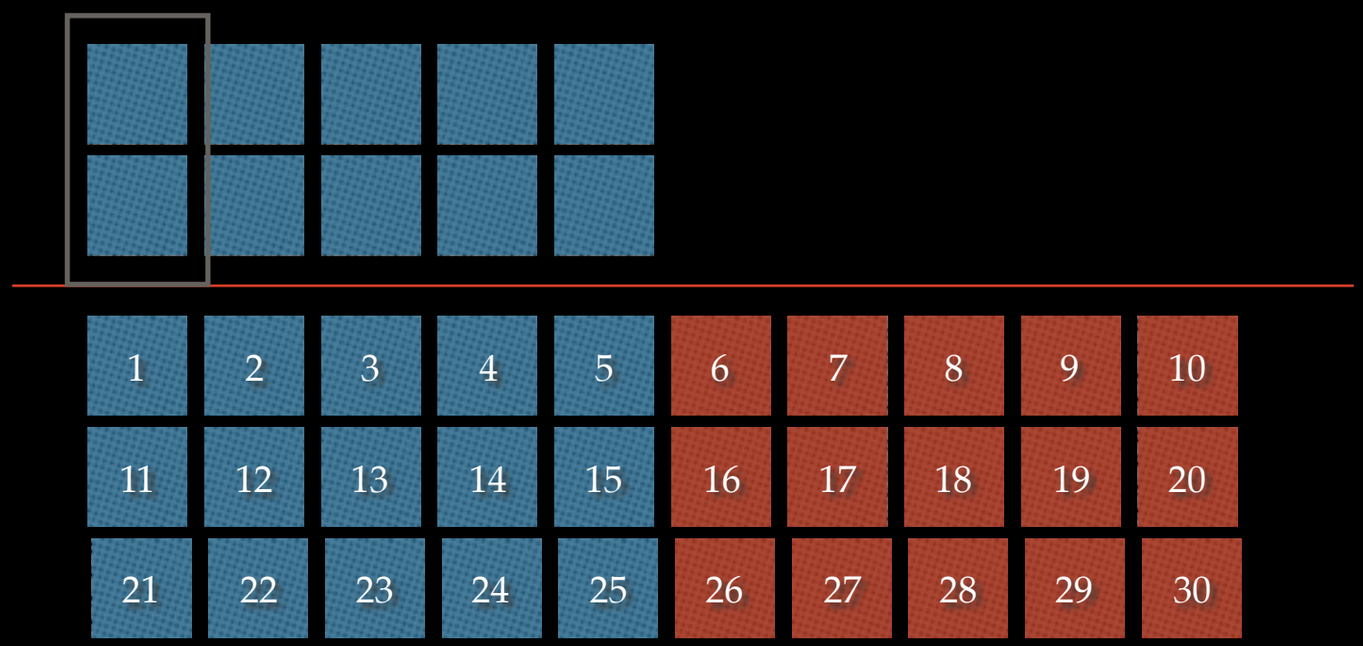
CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this



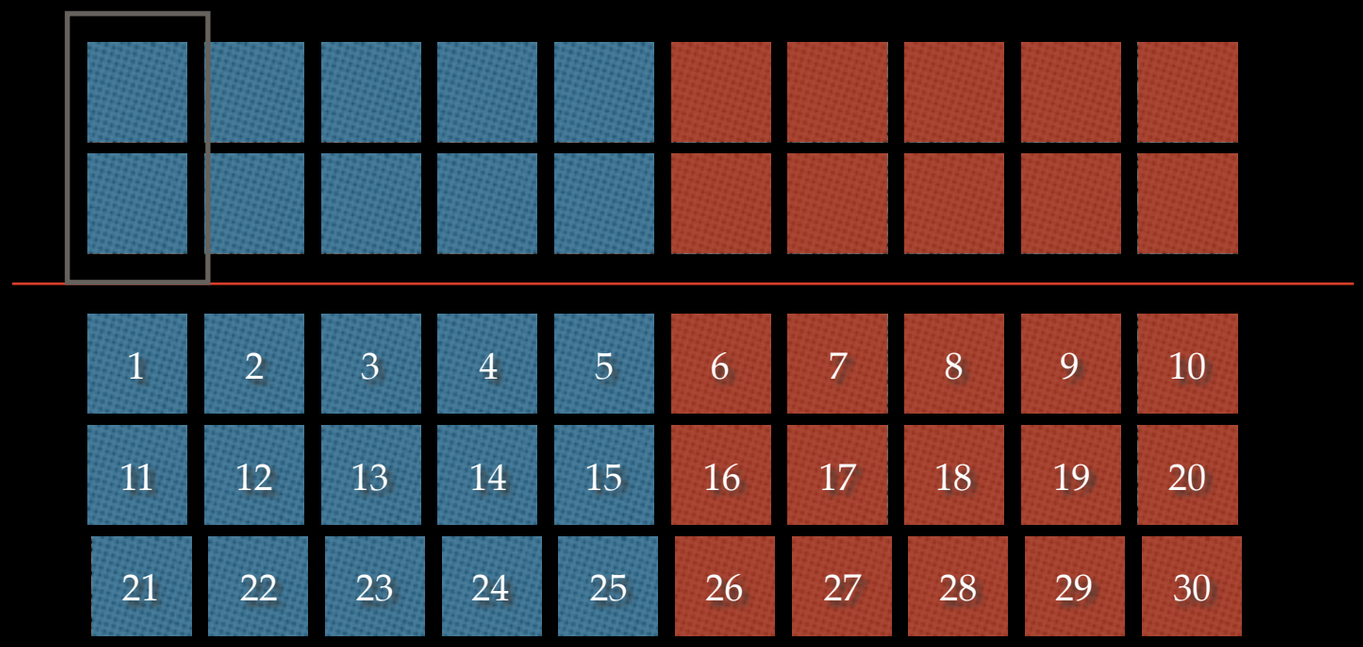
CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this



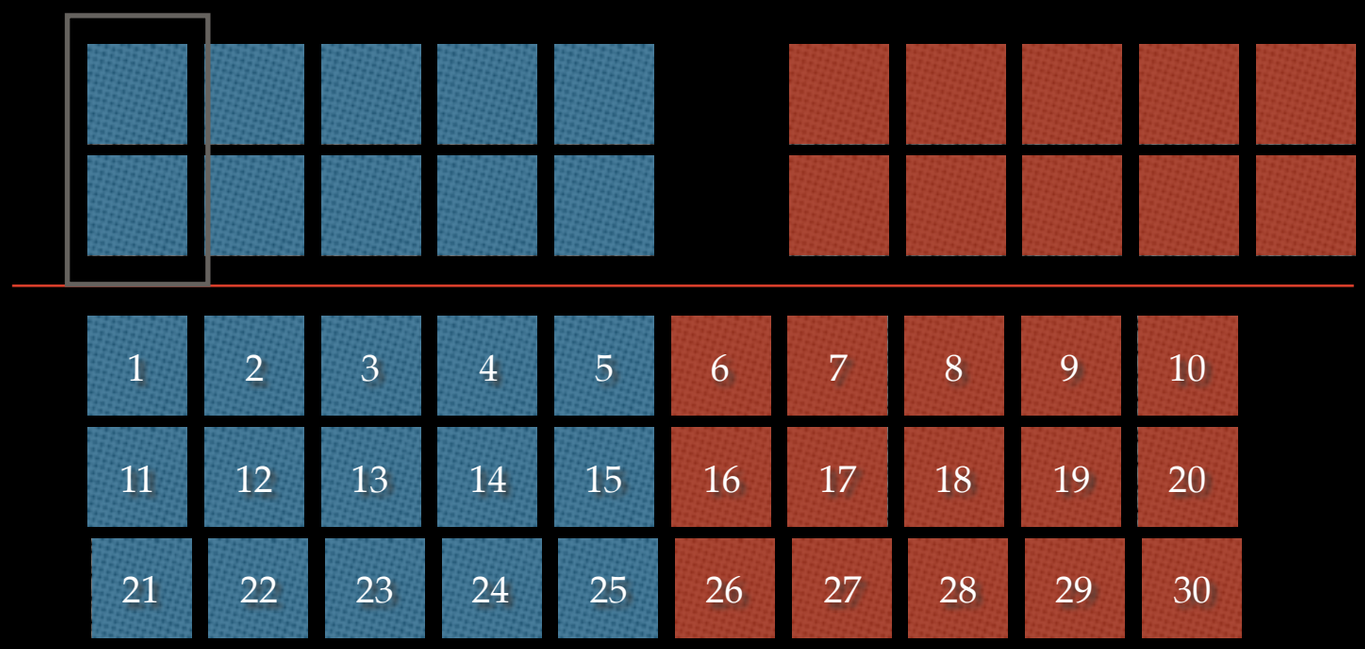
CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this



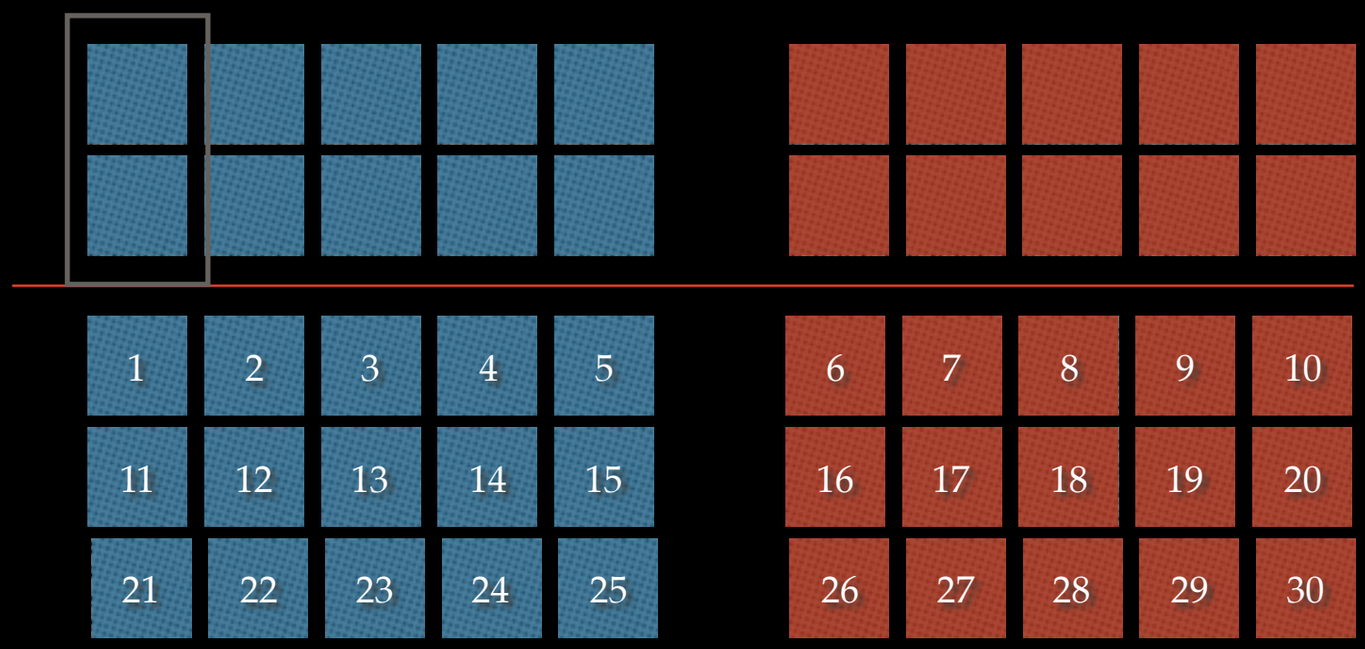
CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this



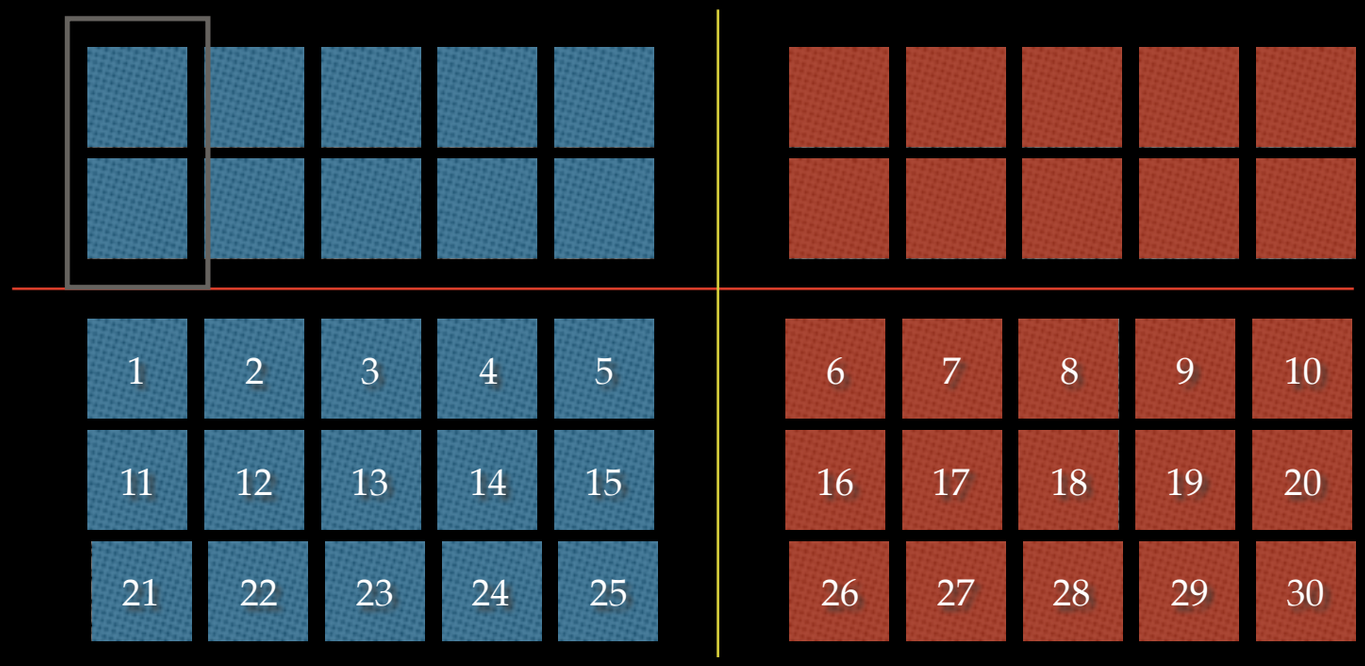
CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this



CACHE COLOURING

- Figure out page colors
- These map to shared cache sets
- Do not share same colors across security boundaries
- Kernel arranges this

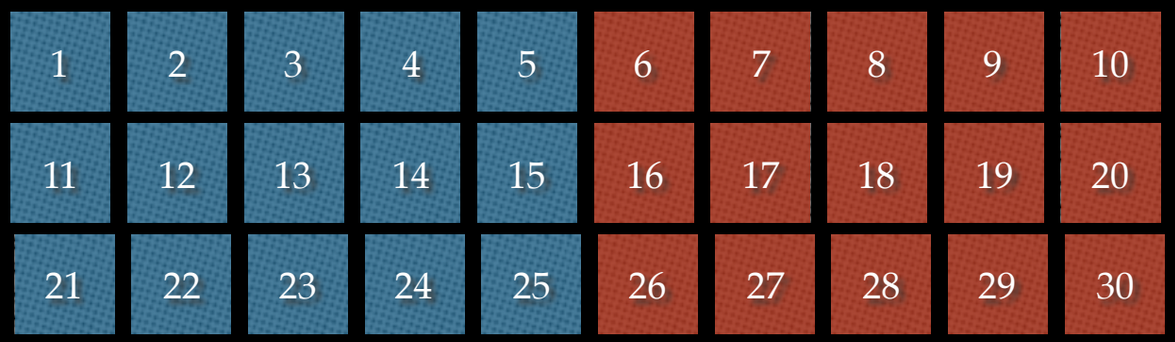


CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature

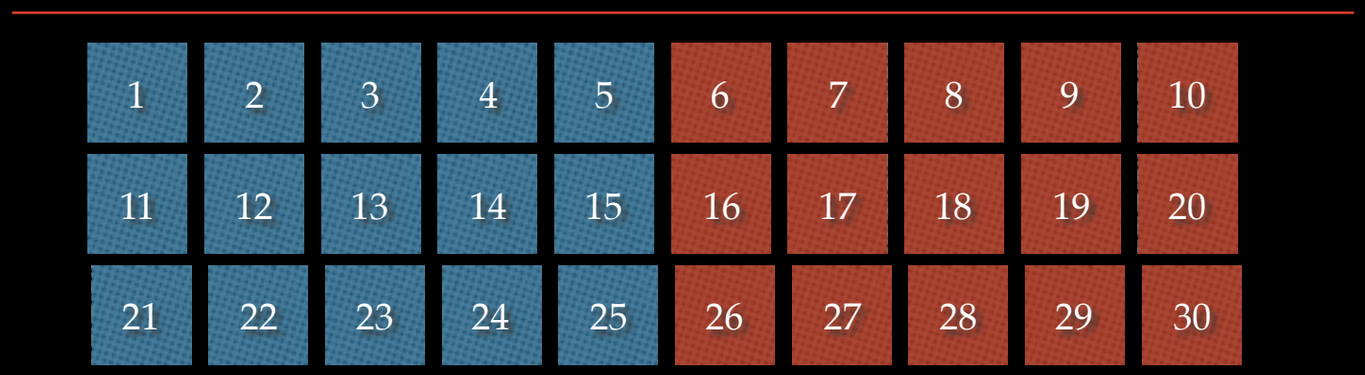
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



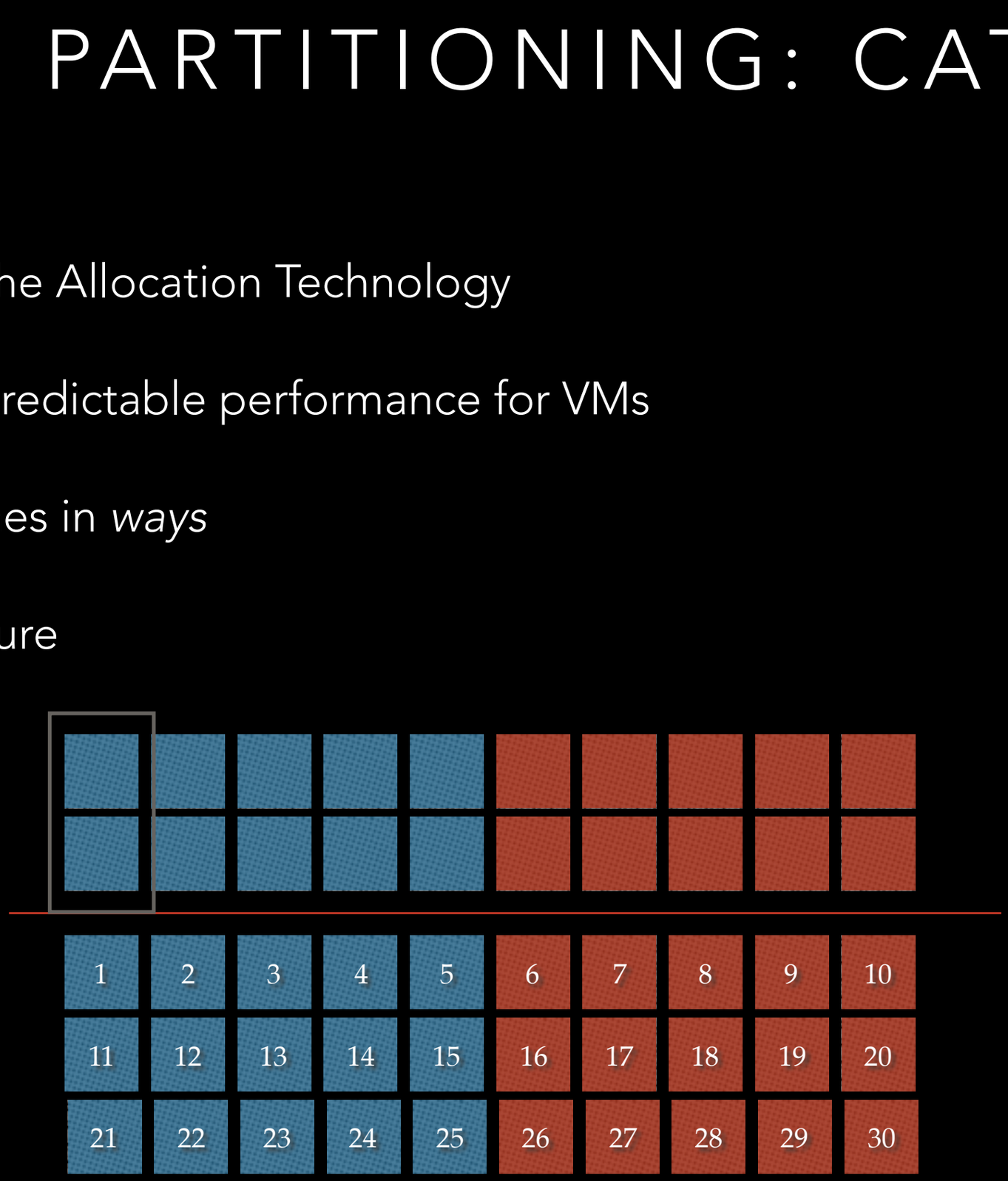
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



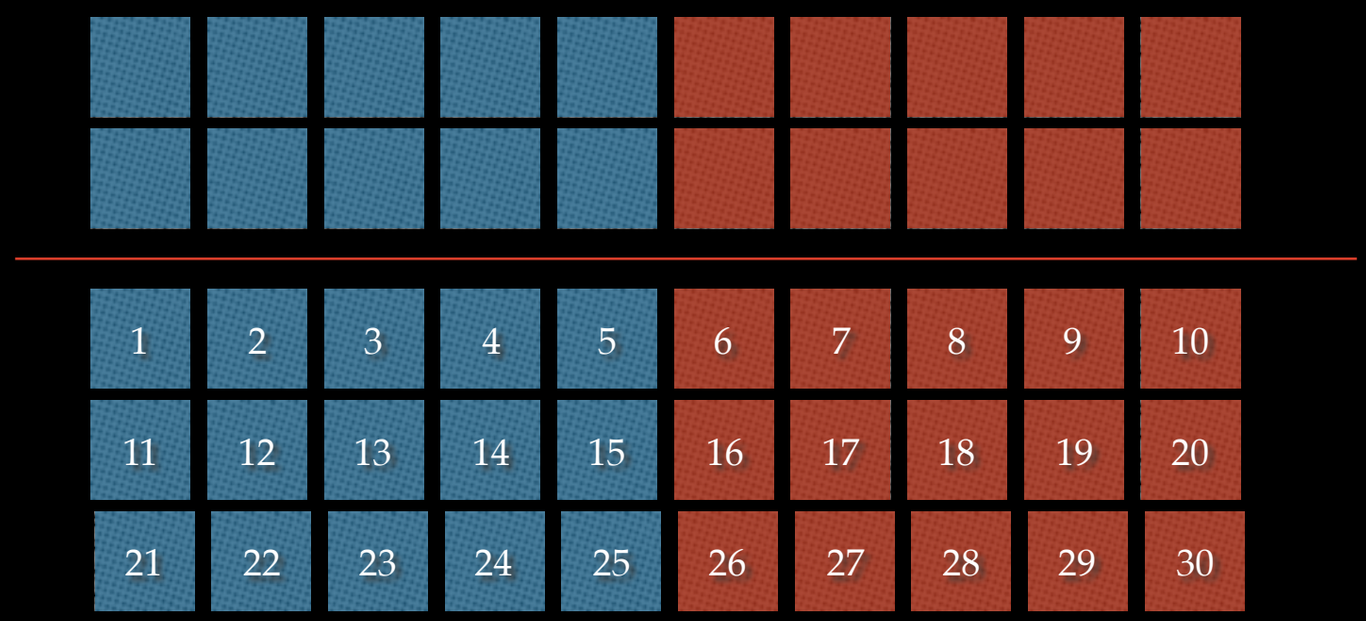
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



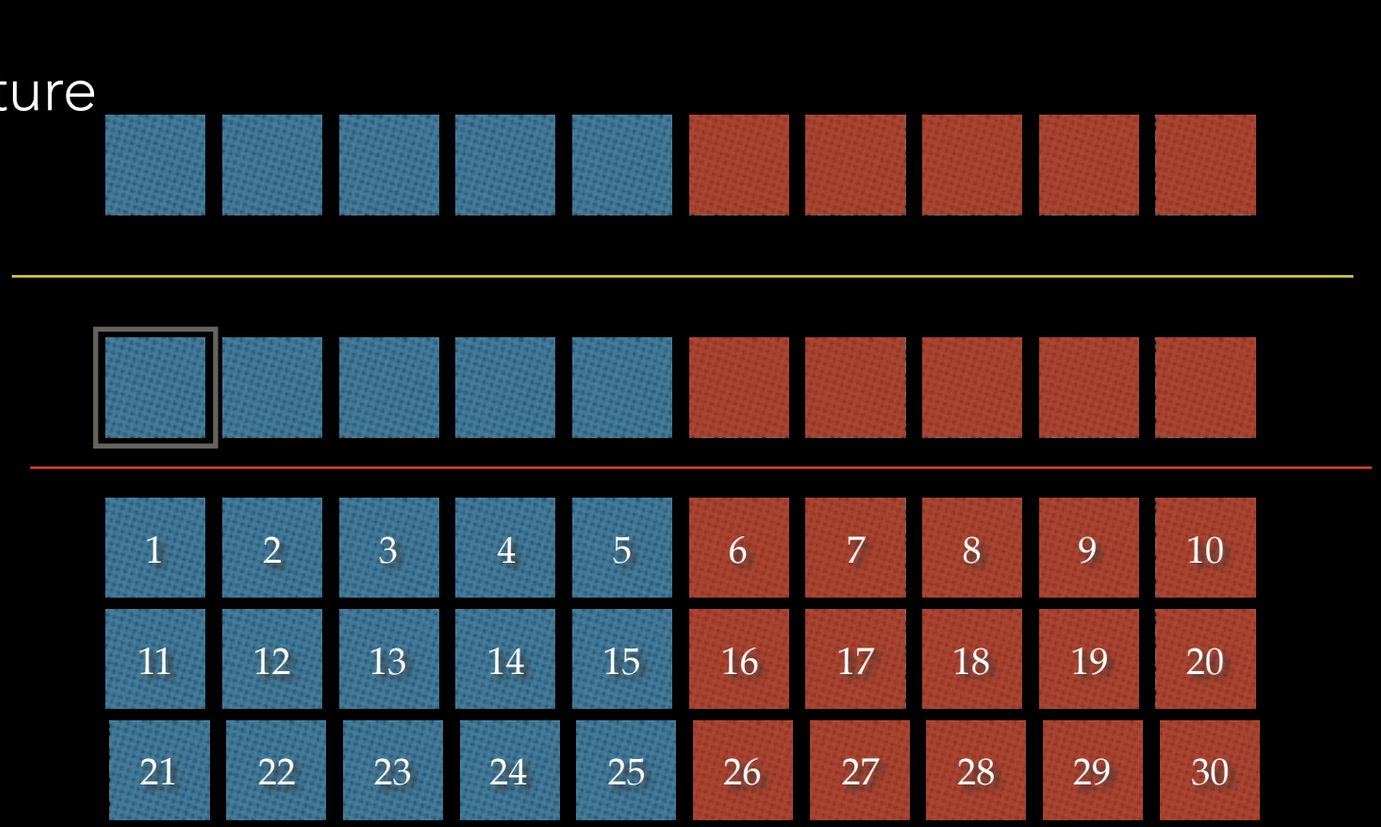
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



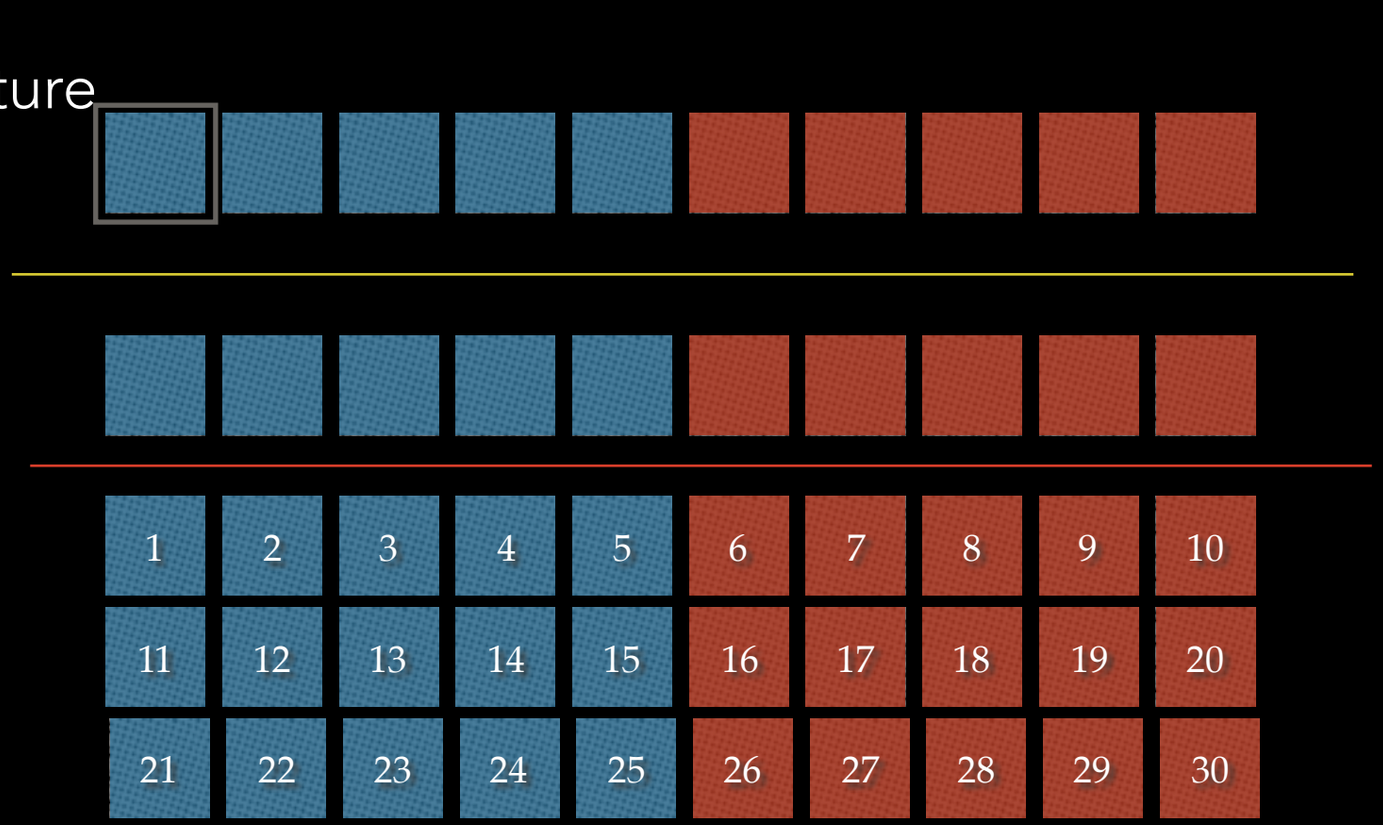
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



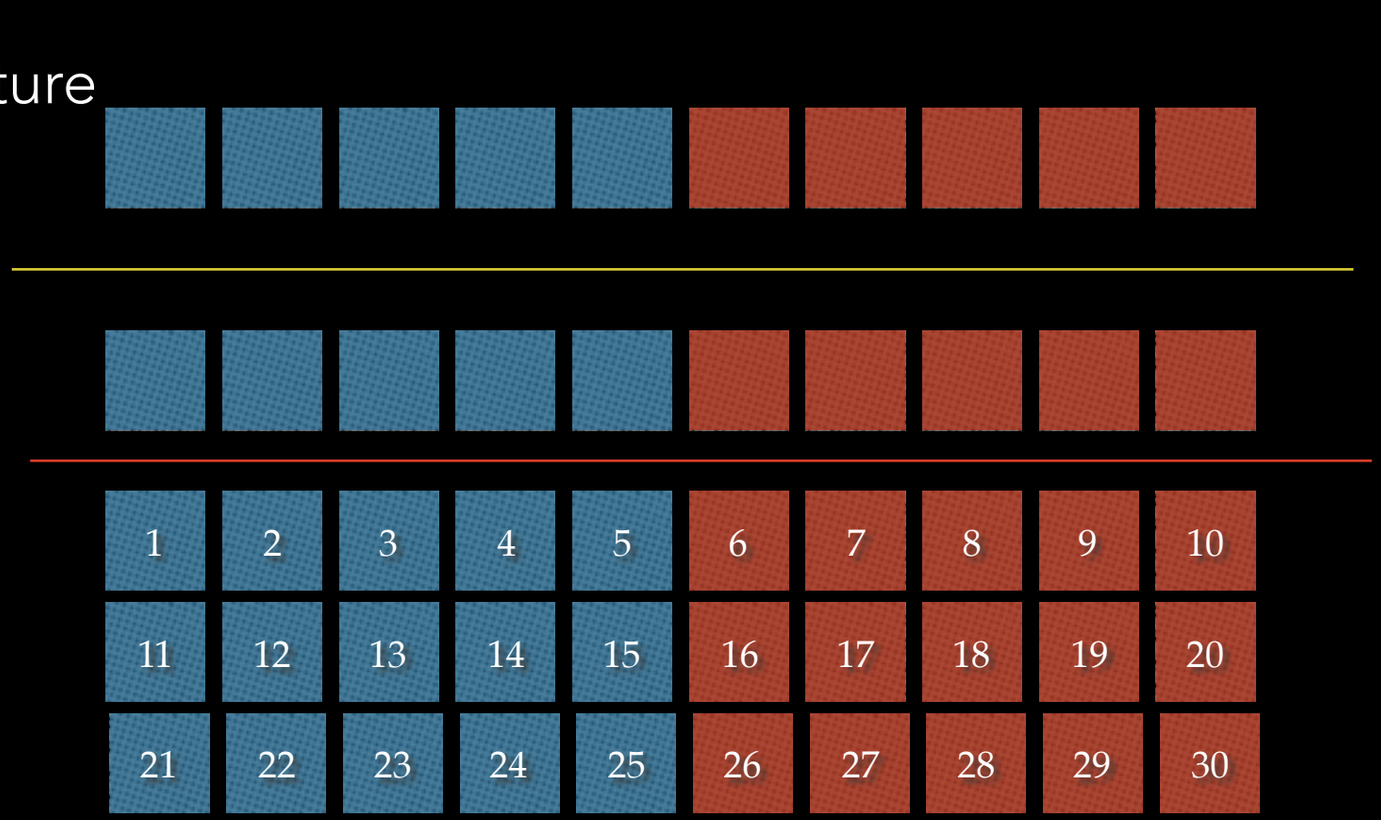
CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



CACHE PARTITIONING: CAT

- Intel CAT: Cache Allocation Technology
- Intended for predictable performance for VMs
- Partitions caches in *ways*
- Hardware feature



CACHE PARTITIONING: TSX

CACHE PARTITIONING: TSX

- Intel TSX: Transactional Synchronization Extensions

CACHE PARTITIONING: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory

CACHE PARTITIONING: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity

CACHE PARTITIONING: TSX

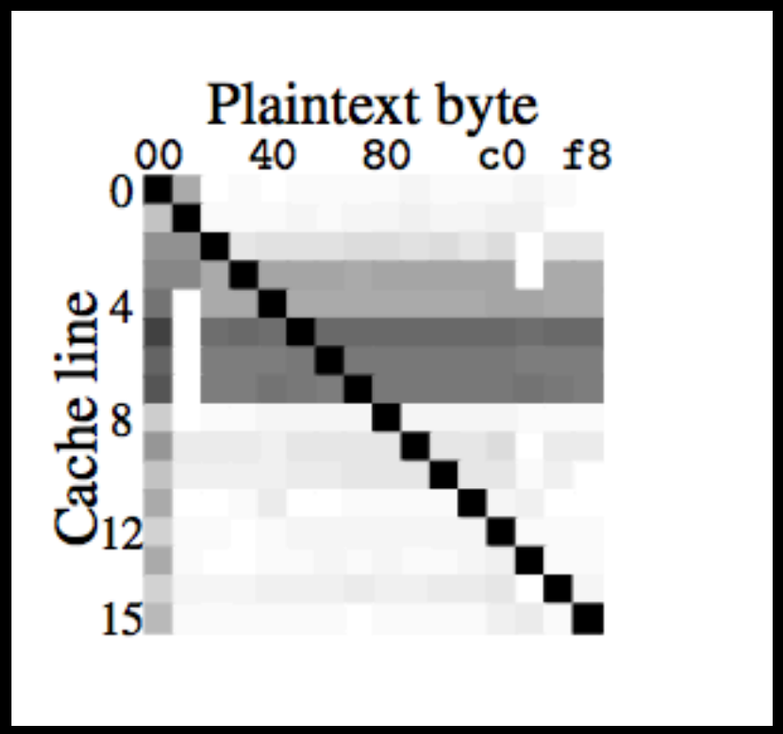
- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort

CACHE PARTITIONING: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
- We can use this as a defence

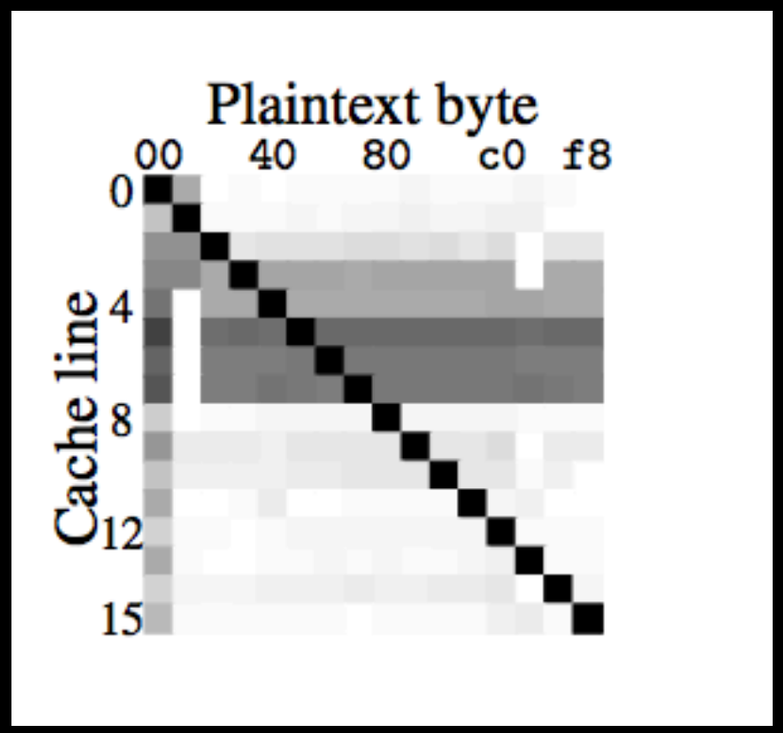
CACHE PARTITIONING: TSX

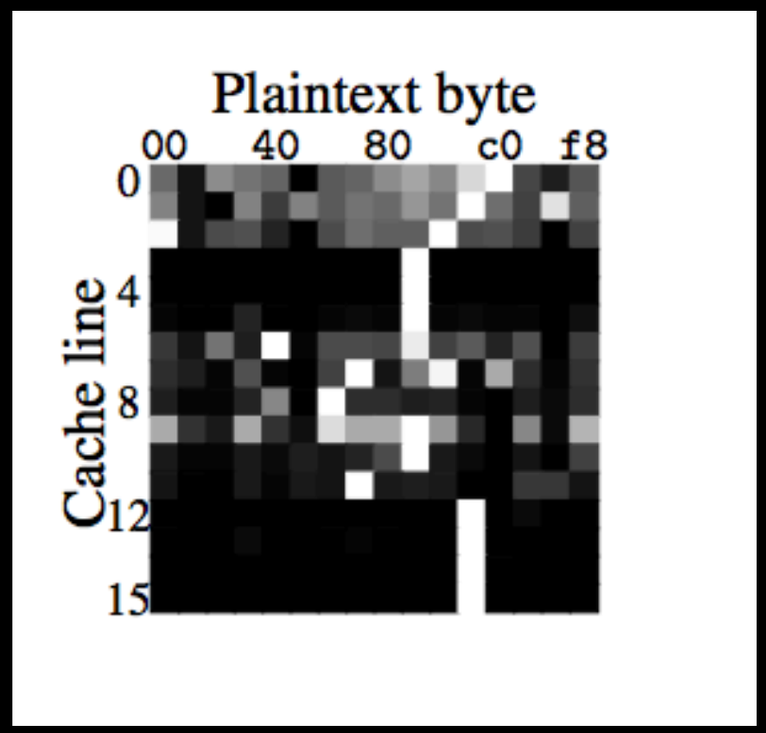
- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
- We can use this as a defence



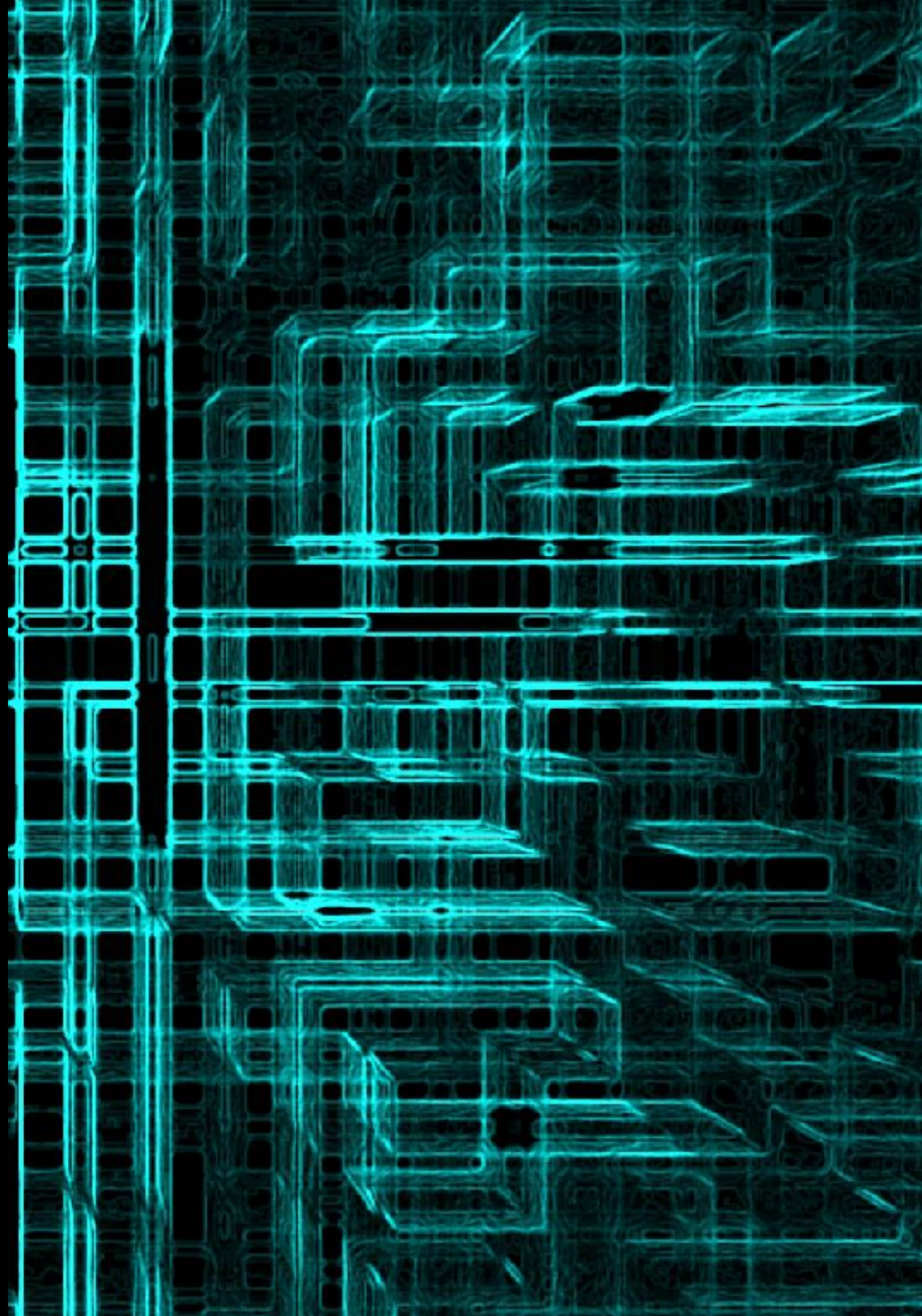
CACHE PARTITIONING: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
- We can use this as a defence





HYPER THREADING



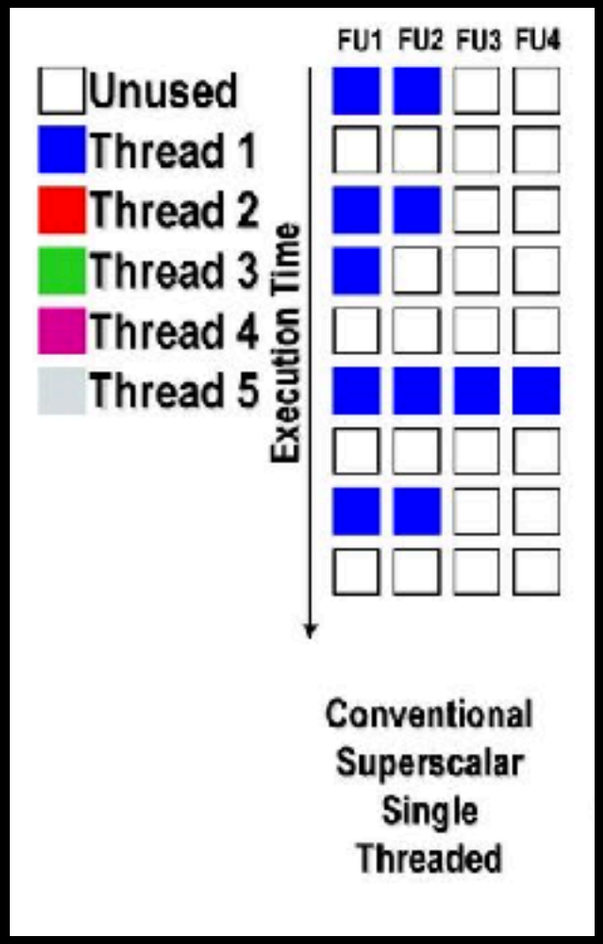
SUPERSCALAR CPU UTILISATION

SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation

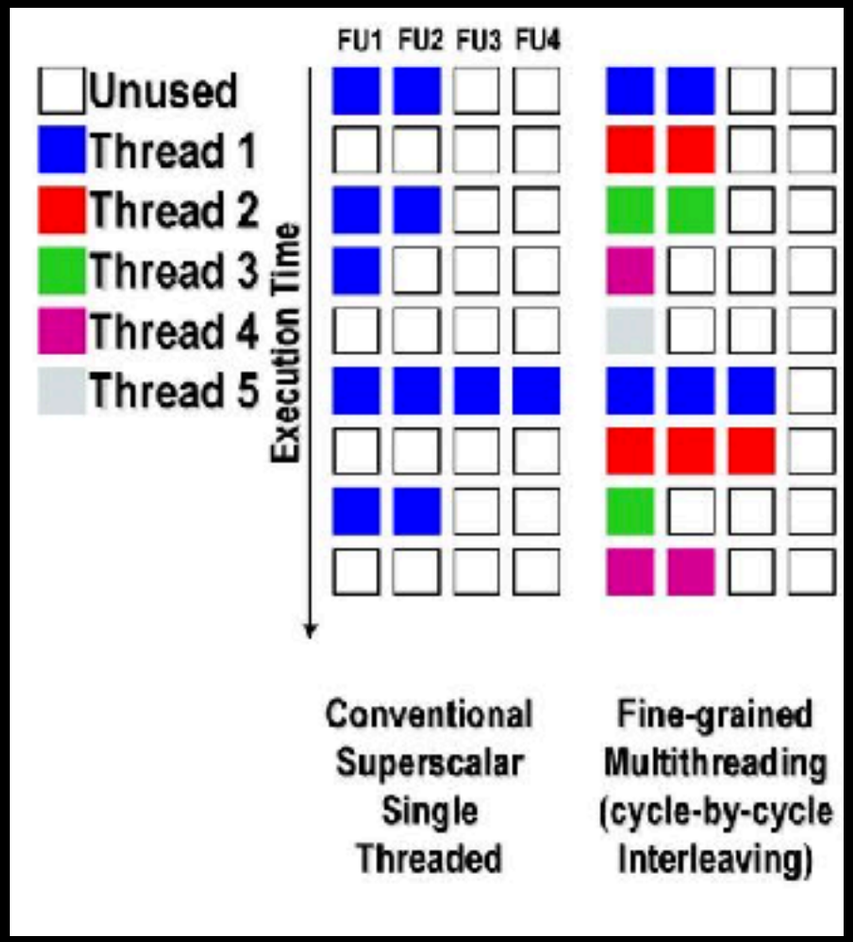
SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation



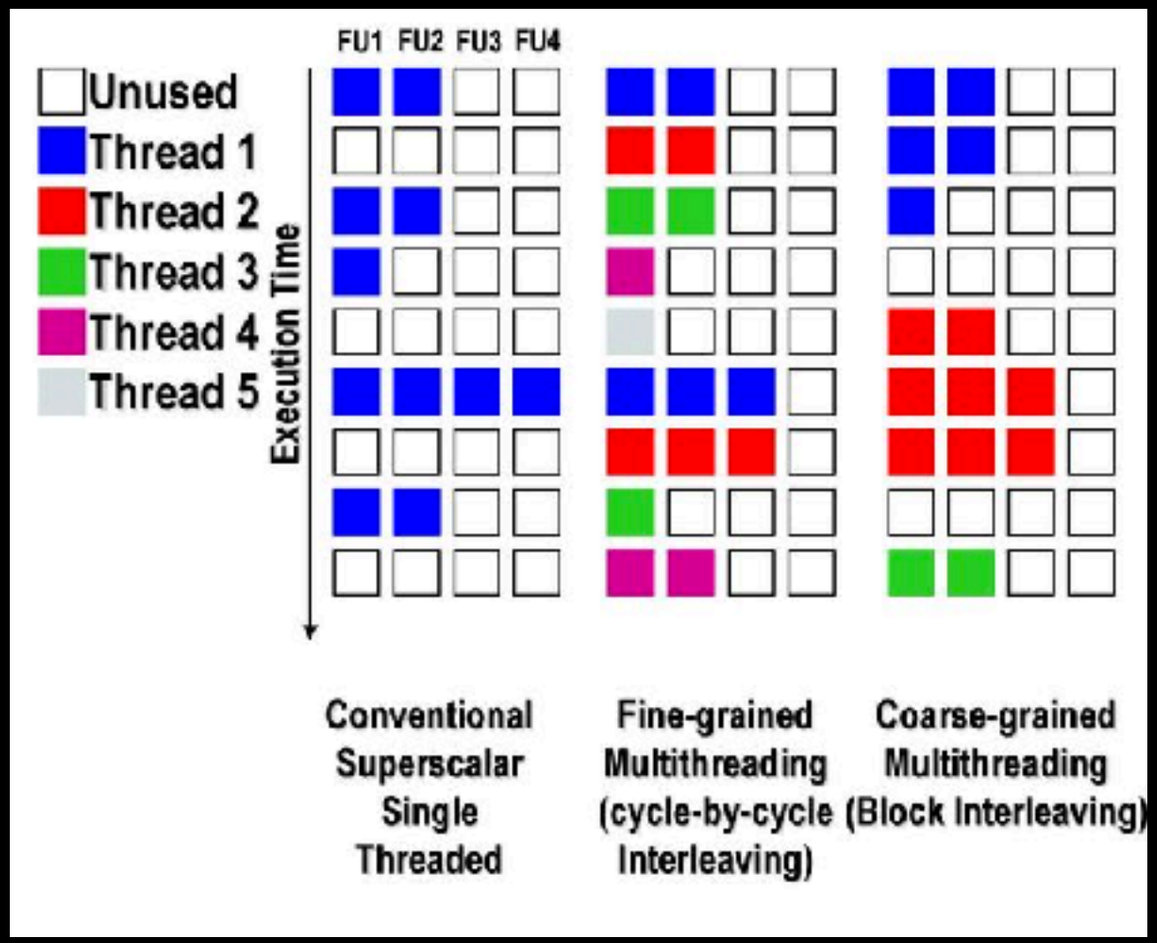
SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation



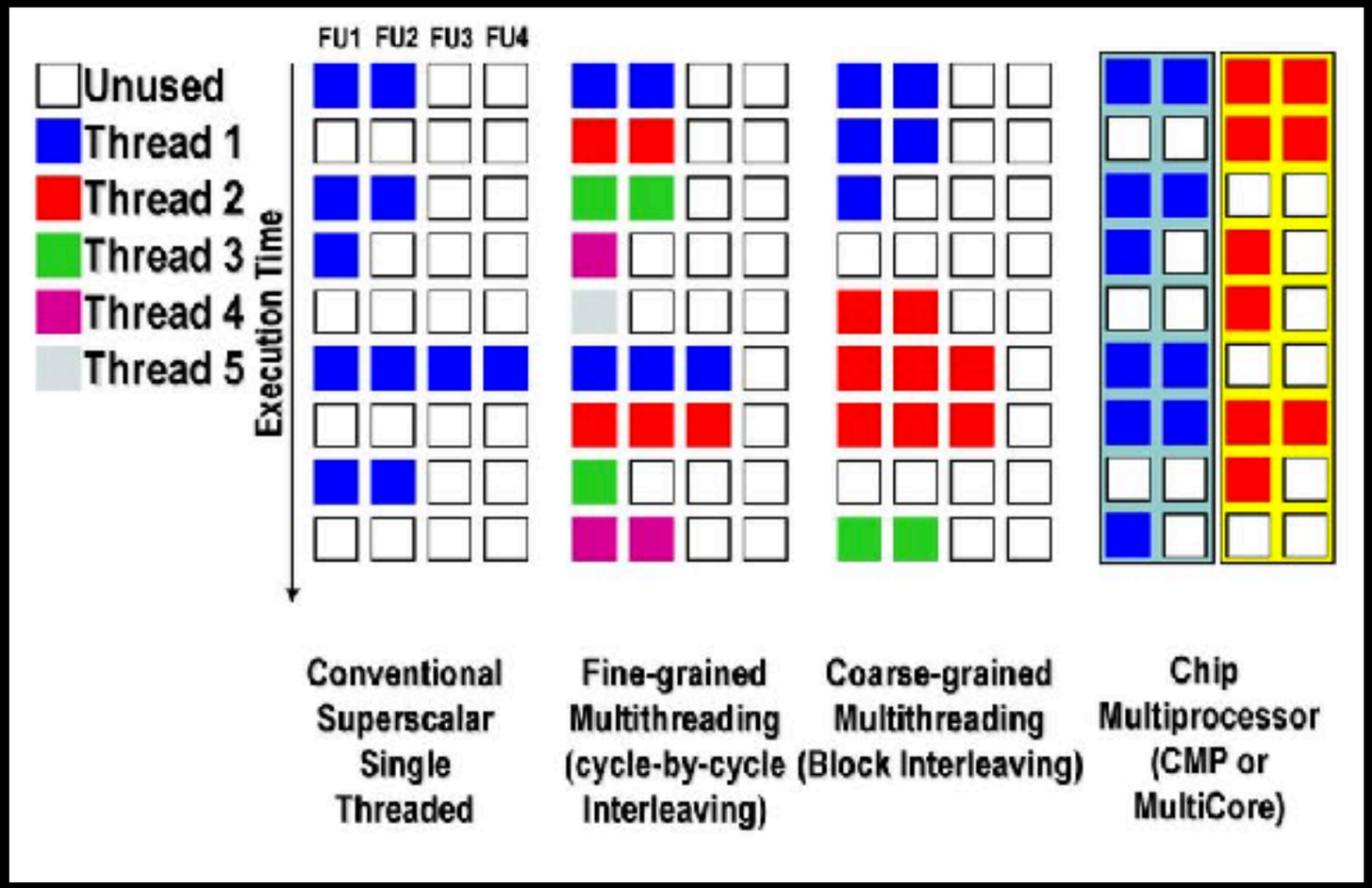
SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation



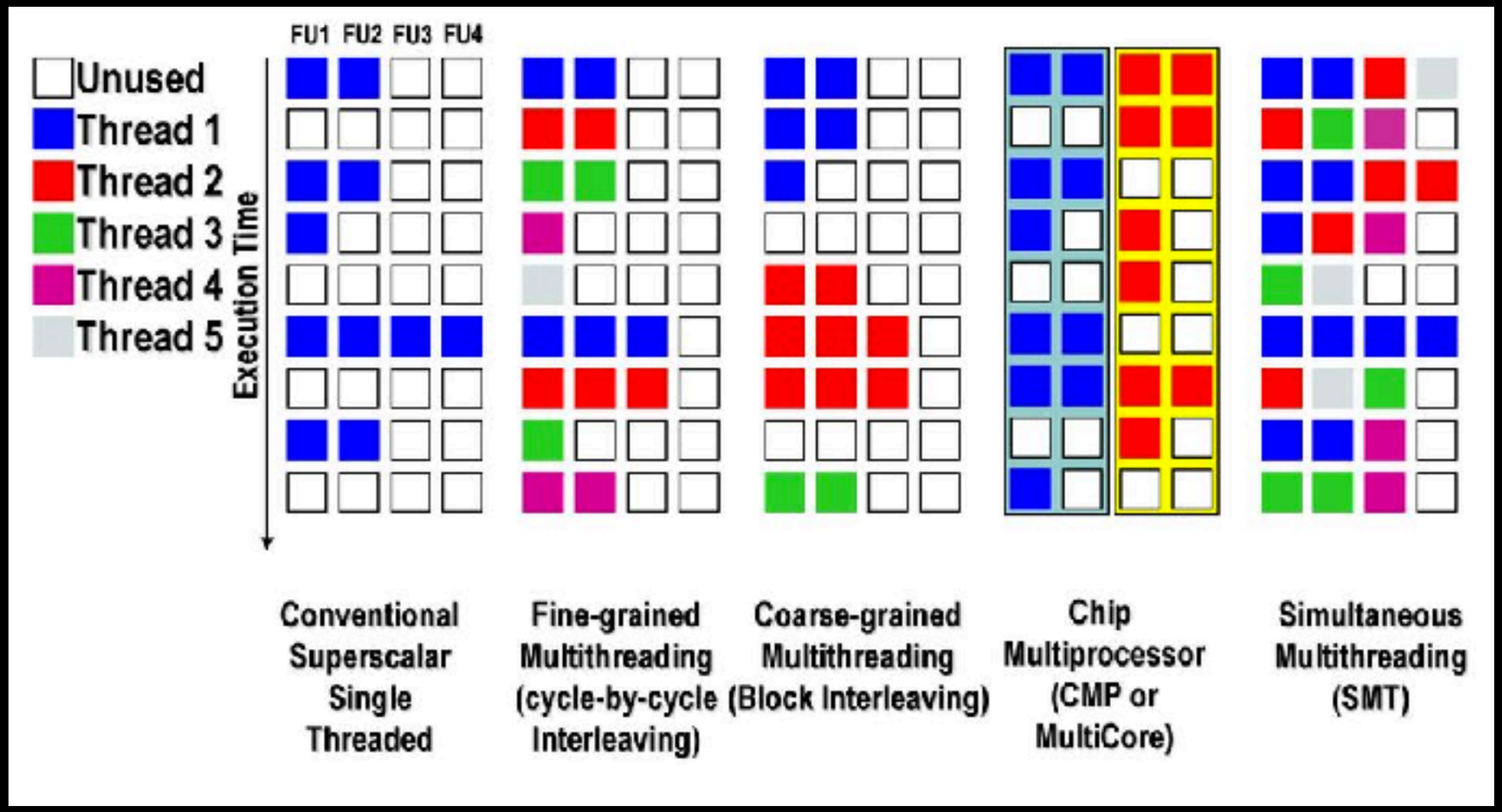
SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation

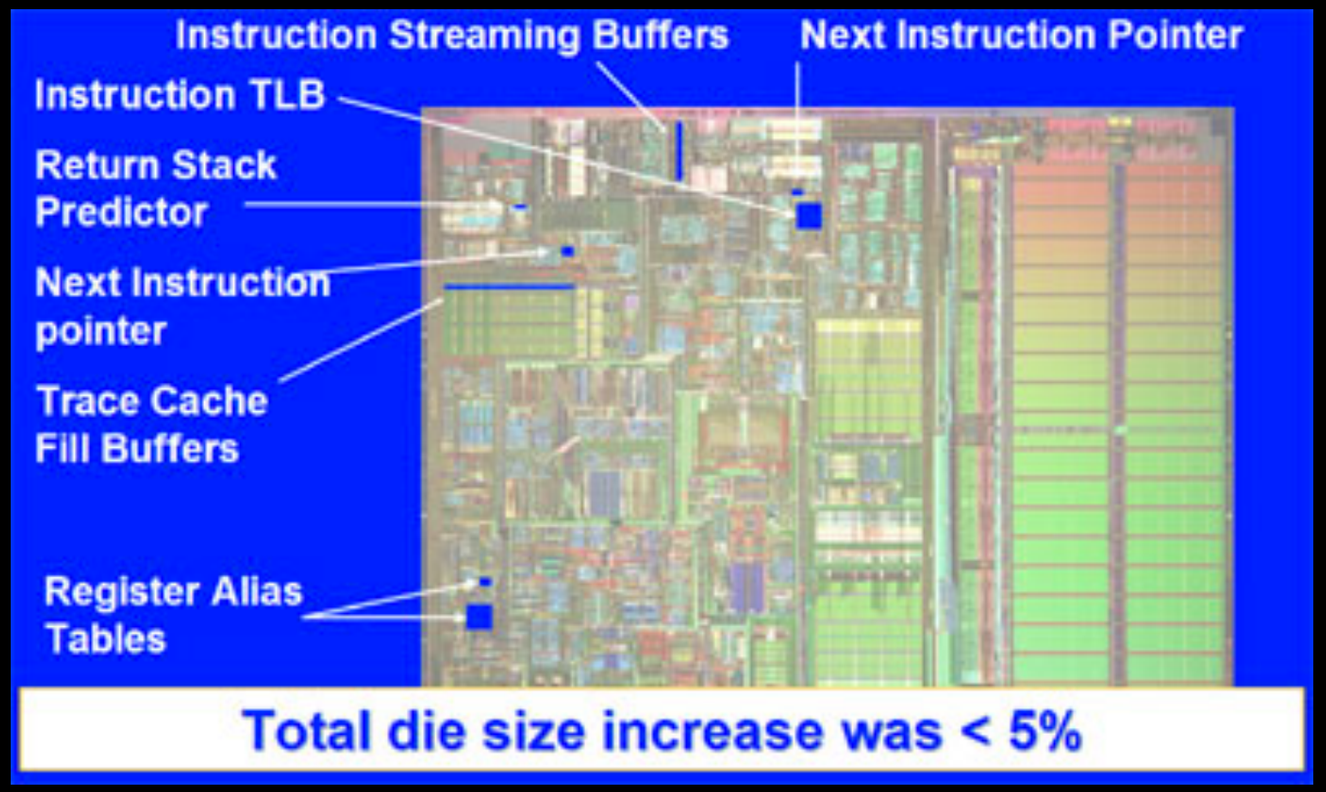


SUPERSCALAR CPU UTILISATION

- Share functional units (FU): increase utilisation

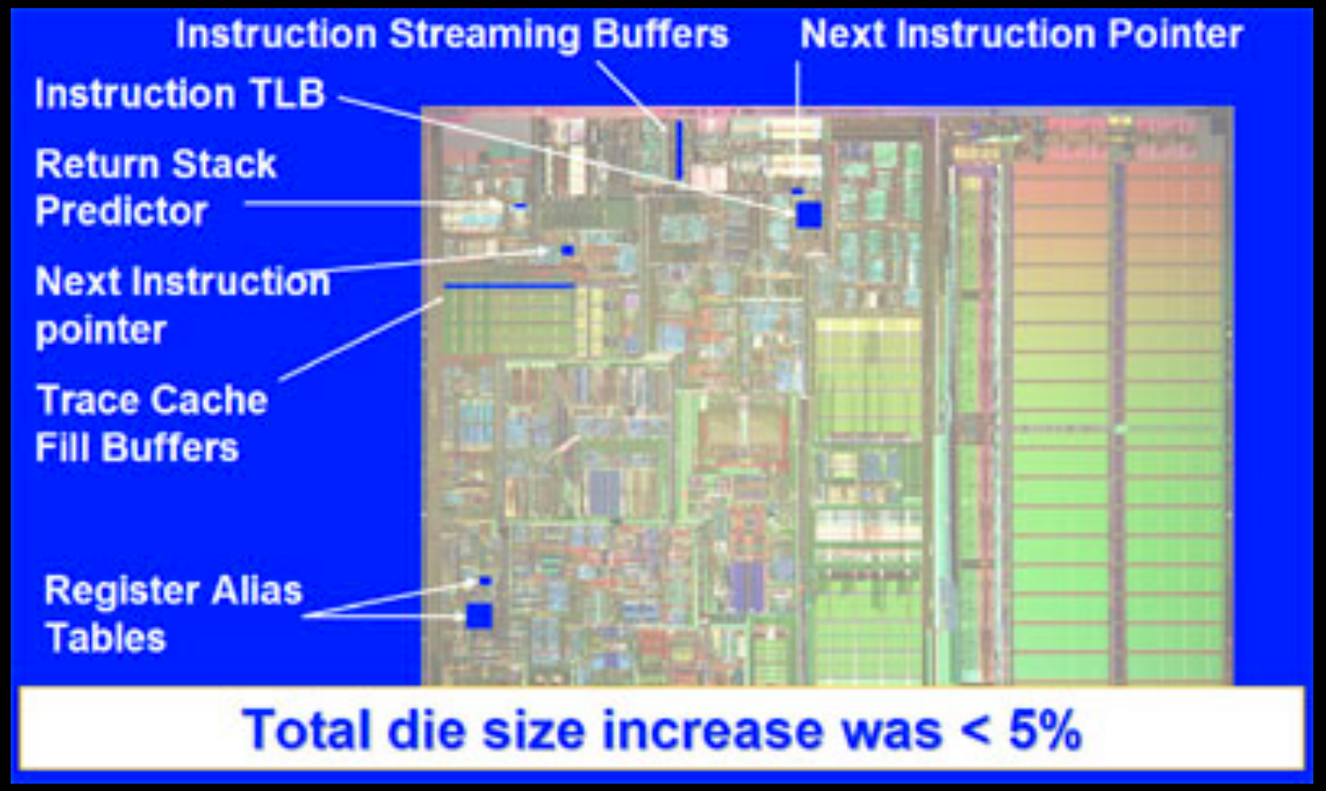


HYPER THREADING



HYPER THREADING

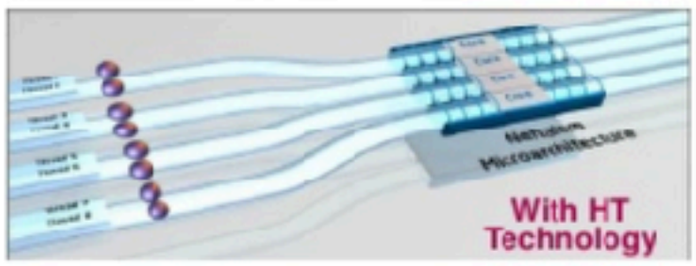
- Low investment, high utilisation yield



HYPER THREADING

Intel® Hyper-Threading Technology

- Nehalem is a scalable multi-core architecture
- Hyper-Threading Technology augments benefits
  - Power-efficient way to boost performance in all form factors: higher multi-threaded performance, faster multi-tasking response



	Hyper-Threading		Multi-cores
	Shared or Partitioned	Replicated	Replicated
Register State		X	X
Return Stack		X	X
Reorder Buffer	X		X
Instruction TLB	X		X
Reservation Stations	X		X
Cache (L1, L2)	X		X
Data TLB	X		X
Execution Units	X		X

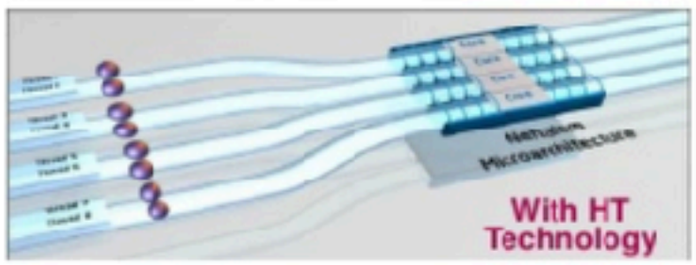
- Next generation Hyper-Threading Technology:
  - Low-latency pipeline architecture
  - Enhanced cache architecture
  - Higher memory bandwidth

HYPER THREADING

- Significant resource sharing: good and bad

Intel® Hyper-Threading Technology

- Nehalem is a scalable multi-core architecture
- Hyper-Threading Technology augments benefits
  - Power-efficient way to boost performance in all form factors: higher multi-threaded performance, faster multi-tasking response

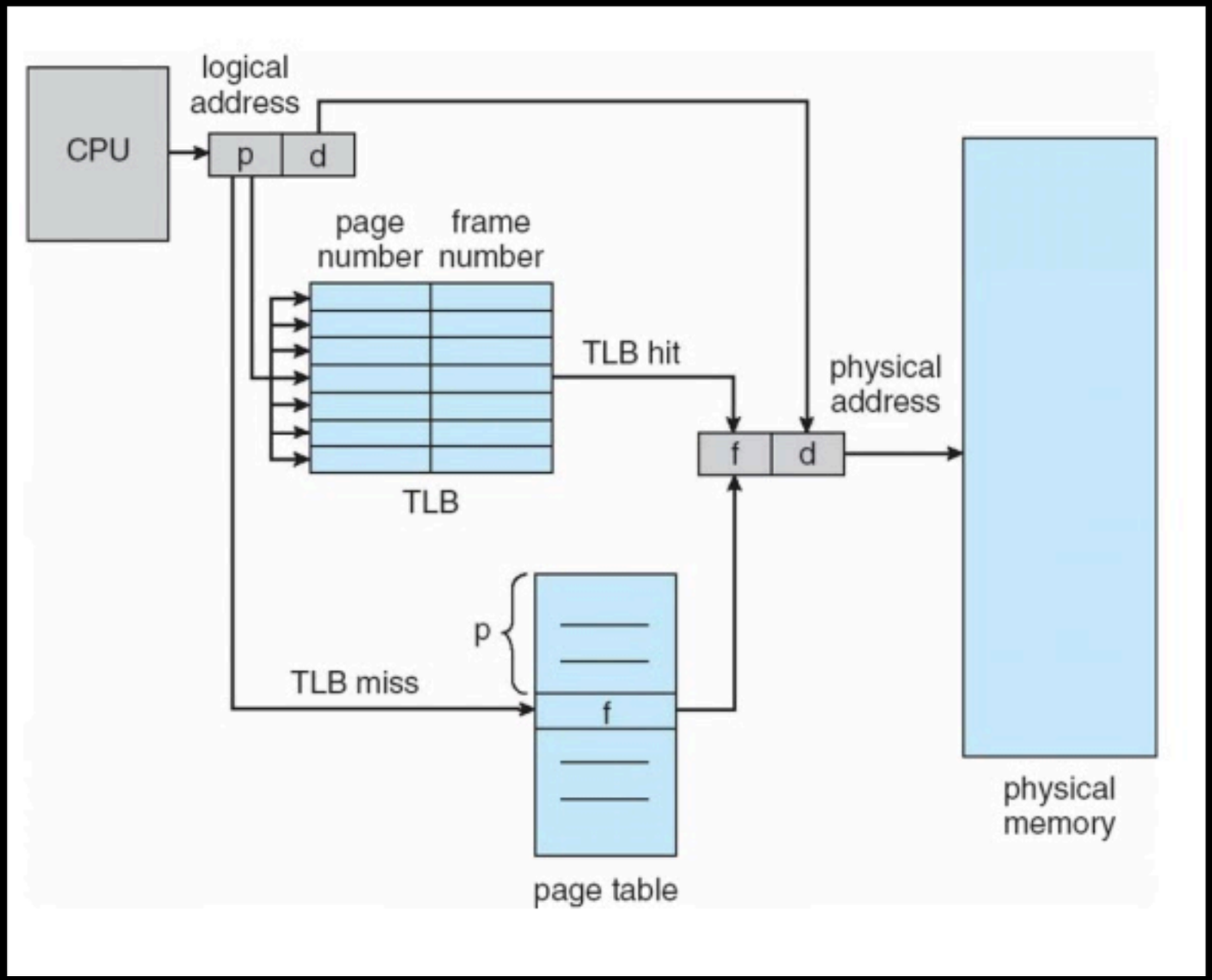


	Hyper-Threading		Multi-cores
	Shared or Partitioned	Replicated	Replicated
Register State		X	X
Return Stack		X	X
Reorder Buffer	X		X
Instruction TLB	X		X
Reservation Stations	X		X
Cache (L1, L2)	X		X
Data TLB	X		X
Execution Units	X		X

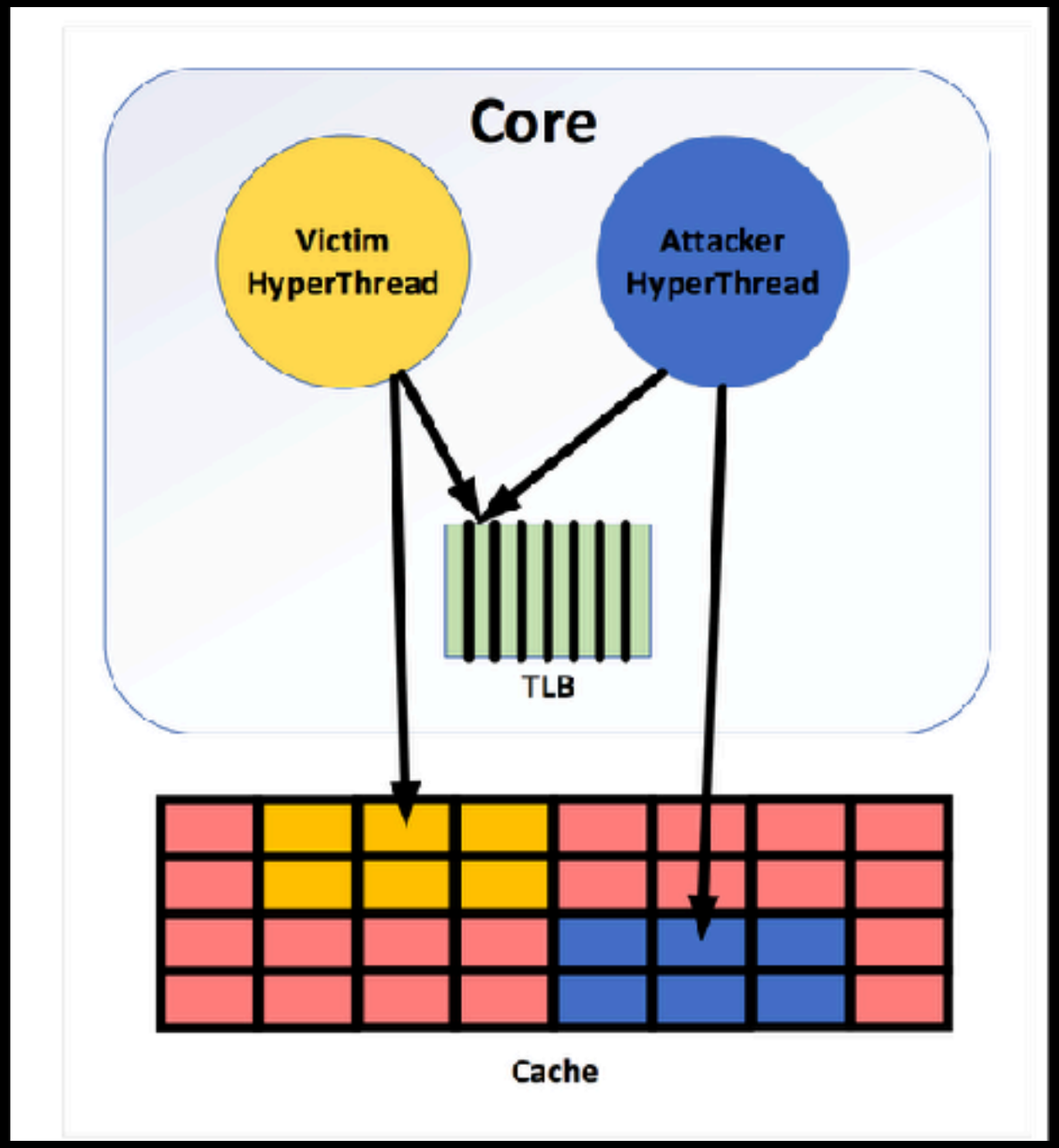
- Next generation Hyper-Threading Technology:
  - Low-latency pipeline architecture
  - Enhanced cache architecture
  - Higher memory bandwidth

TLBLEED

TLBLEED: TLB


CPU
logical address
p
d
page number
frame number
TLB hit
physical address
f
d
TLB
p
TLB miss
f
page table
physical memory

TLBLEED: TLB AS SHARED STATE



TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Other structures than cache shared between threads?
- What about the TLB?
- Documented: TLB has L1iTLB, L1dTLB, and L2TLB
- They have sets and ways
- Not documented: structure

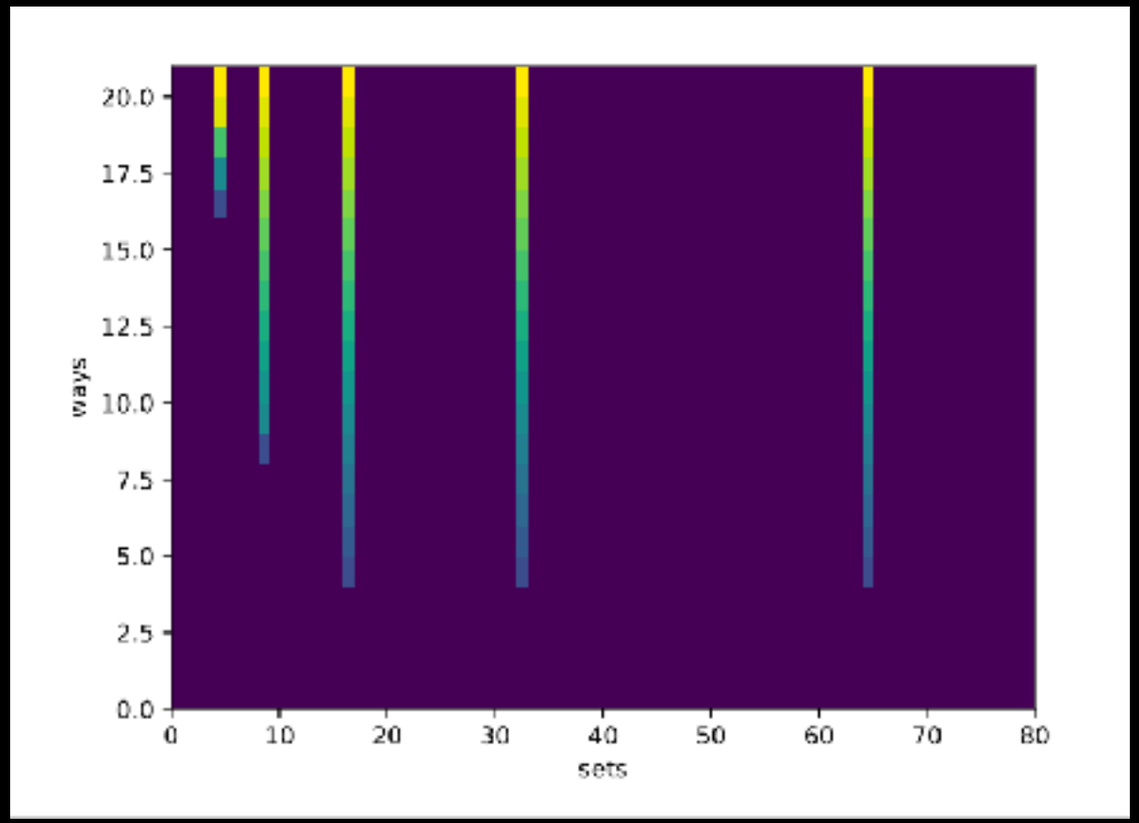
TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Let's experiment with performance counters
- Try linear structure first
- All combinations of ways (set size) and sets (stride)
- Smallest number of ways is it
- Smallest corresponding stride is number of sets

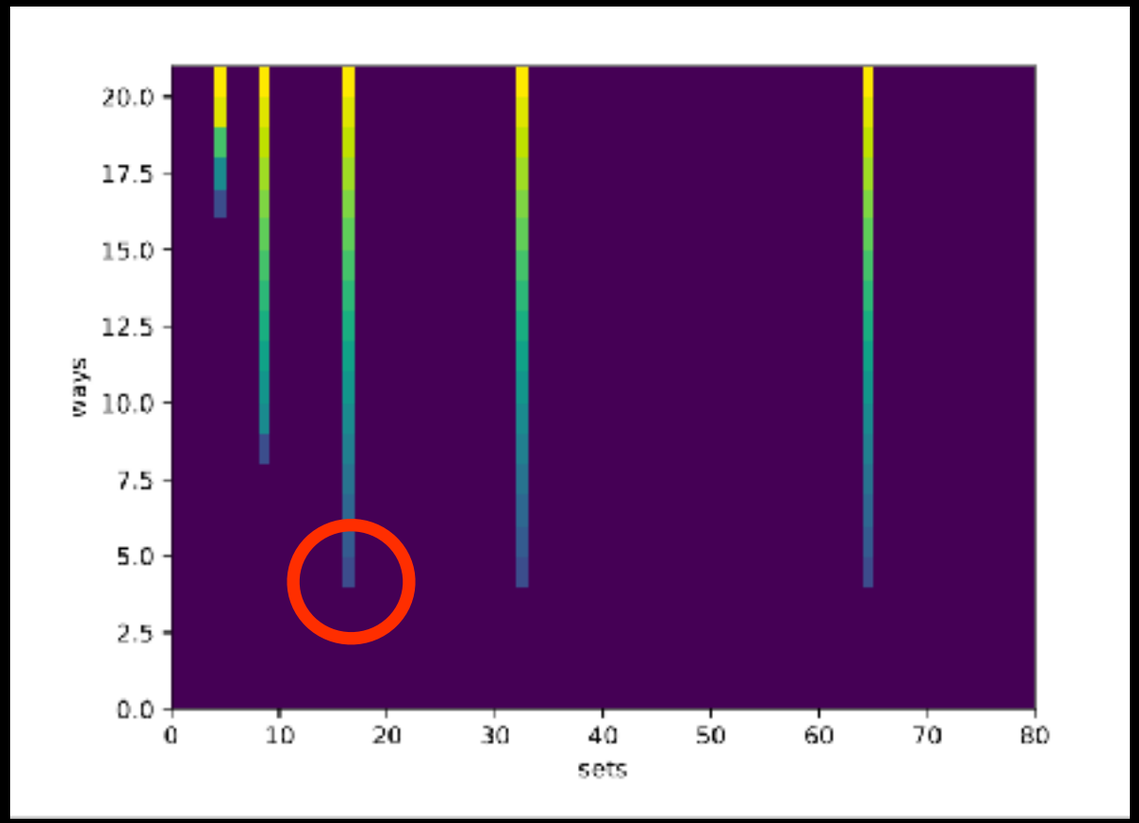
TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Try linear structure first
- All combinations of ways (set size) and sets (stride)
- Smallest number of ways is it
- Smallest corresponding stride is number of sets



TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Try linear structure first
- All combinations of ways (set size) and sets (stride)
- Smallest number of ways is it
- Smallest corresponding stride is number of sets



TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- For L2TLB:
  We reverse engineered a more complex hash function

TLBLEED: TLB AS SHARED STATE

- For L2TLB:
  We reverse engineered a more complex hash function

- Skylake XORs 14 bits, Broadwell XORs 16 bits

TLBLEED: TLB AS SHARED STATE

- For L2TLB:
  We reverse engineered a more complex hash function

- Skylake XORs 14 bits, Broadwell XORs 16 bits

- Represented by this matrix, using modulo 2 arithmetic

TLBLEED: TLB AS SHARED STATE

- For L2TLB:
  We reverse engineered a more complex hash function

- Skylake XORs 14 bits, Broadwell XORs 16 bits

- Represented by this matrix, using modulo 2 arithmetic

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters

TLBLEED: TLB AS SHARED STATE

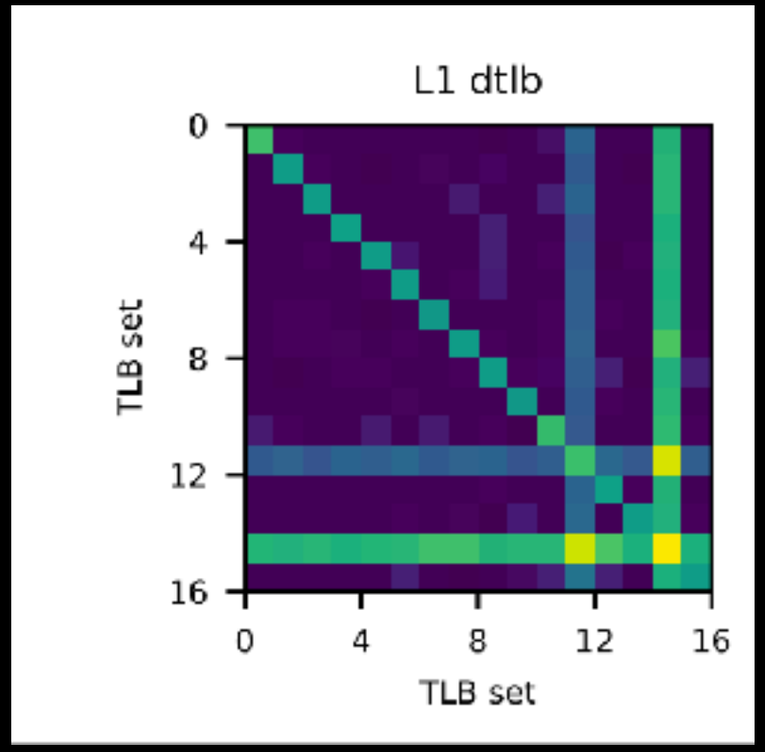
- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?

TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?
- Let’s experiment with misses when accessing the same set

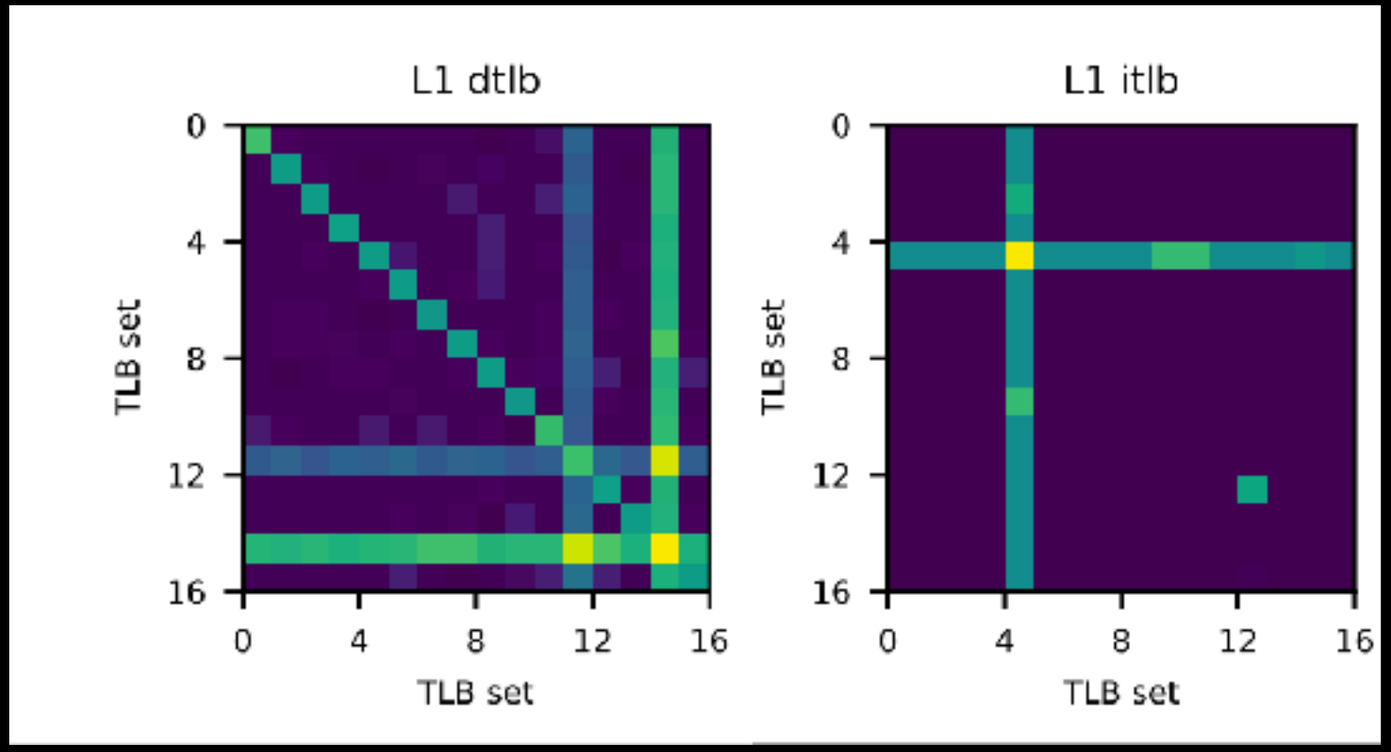
TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?
- Let’s experiment with misses when accessing the same set



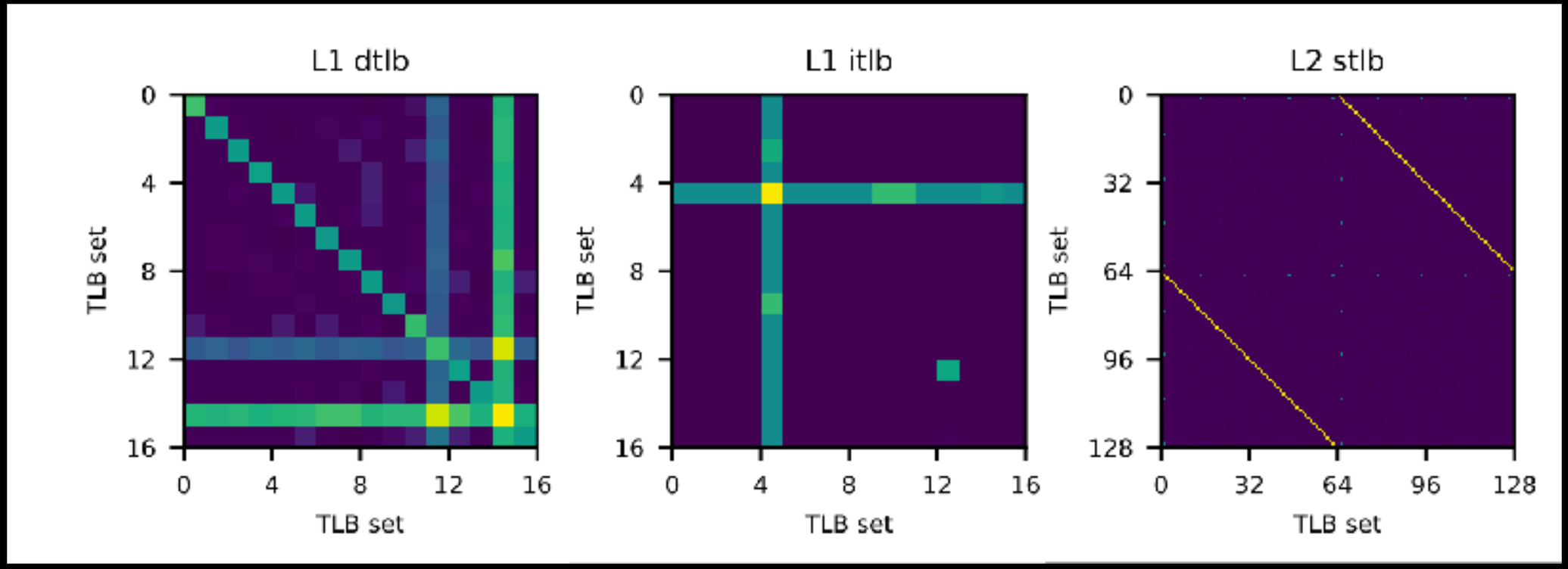
TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?
- Let’s experiment with misses when accessing the same set



TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?
- Let’s experiment with misses when accessing the same set



TLBLEED: TLB AS SHARED STATE

		L1 dTLB					L1 iTLB					L2 sTLB				
Name	year	set	w	pn	hsh	shr	set	w	pn	hsh	shr	set	w	pn	hsh	shr
Sandybridge	2011	16	4	7.0	lin	✓	16	4	50.0	lin	✗	128	4	16.3	lin	✓
Ivybridge	2012	16	4	7.1	lin	✓	16	4	49.4	lin	✗	128	4	18.0	lin	✓
Haswell	2013	16	4	8.0	lin	✓	8	8	27.4	lin	✗	128	8	17.1	lin	✓
HaswellXeon	2014	16	4	7.9	lin	✓	8	8	28.5	lin	✗	128	8	16.8	lin	✓
Skylake	2015	16	4	9.0	lin	✓	8	8	2.0	lin	✗	128	12	212.0	XOR-7	✓
BroadwellXeon	2016	16	4	8.0	lin	✓	8	8	18.2	lin	✗	256	6	272.4	XOR-8	✓
Coffeelake	2017	16	4	9.1	lin	✓	8	8	26.3	lin	✗	128	12	230.3	XOR-7	✓

TLBLEED: TLB AS SHARED STATE

- We find more TLB properties
- Size, structure, sharing, miss penalty, hash function

		L1 dTLB					L1 iTLB					L2 sTLB				
Name	year	set	w	pn	hsh	shr	set	w	pn	hsh	shr	set	w	pn	hsh	shr
Sandybridge	2011	16	4	7.0	lin	✓	16	4	50.0	lin	✗	128	4	16.3	lin	✓
Ivybridge	2012	16	4	7.1	lin	✓	16	4	49.4	lin	✗	128	4	18.0	lin	✓
Haswell	2013	16	4	8.0	lin	✓	8	8	27.4	lin	✗	128	8	17.1	lin	✓
HaswellXeon	2014	16	4	7.9	lin	✓	8	8	28.5	lin	✗	128	8	16.8	lin	✓
Skylake	2015	16	4	9.0	lin	✓	8	8	2.0	lin	✗	128	12	212.0	XOR-7	✓
BroadwellXeon	2016	16	4	8.0	lin	✓	8	8	18.2	lin	✗	256	6	272.4	XOR-8	✓
Coffeelake	2017	16	4	9.1	lin	✓	8	8	26.3	lin	✗	128	12	230.3	XOR-7	✓

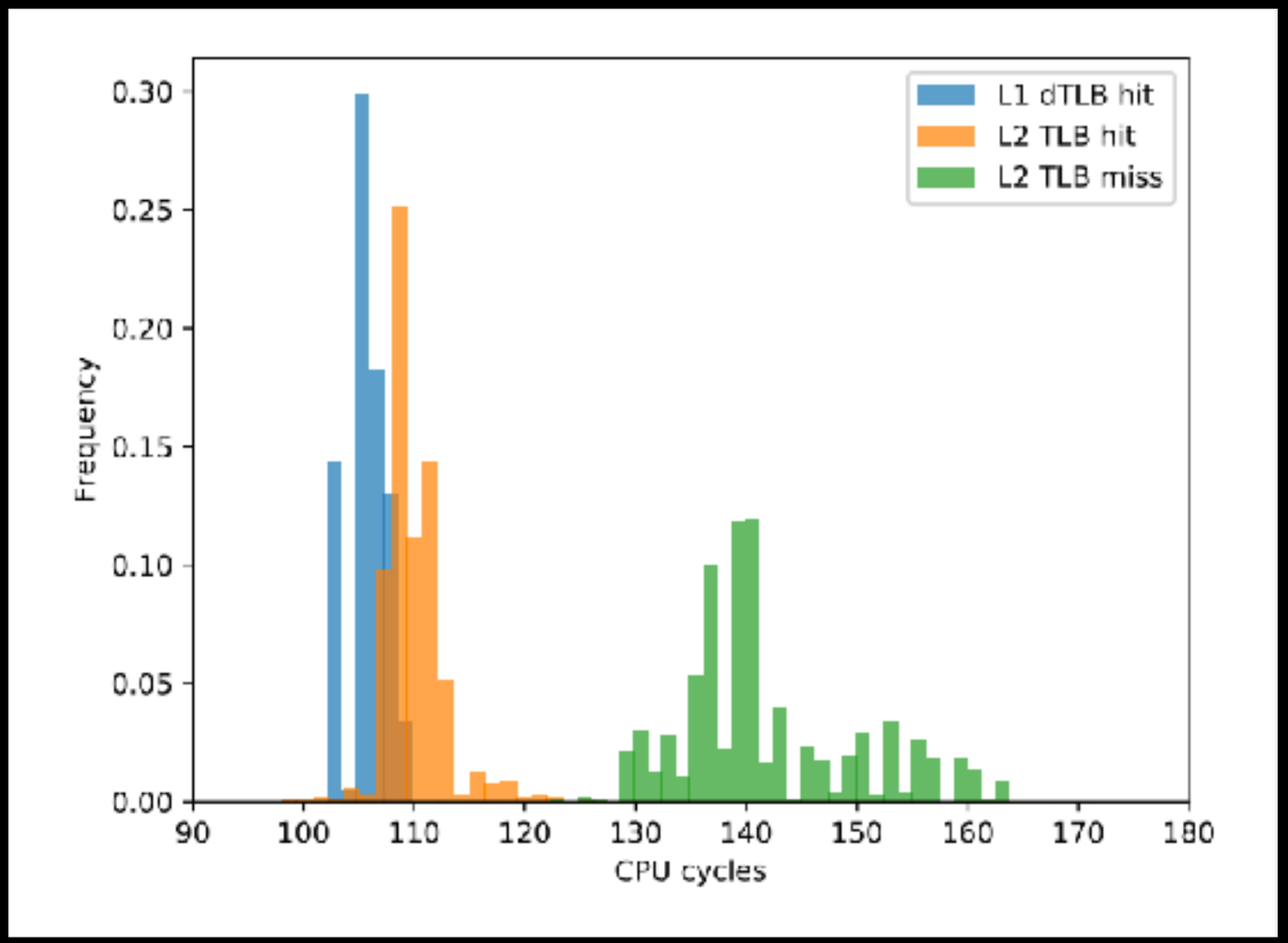
TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Can we use only latency?
- Map many virtual addresses to same physical page

TLBLEED: TLB AS SHARED STATE

- Can we use only latency?
- Map many virtual addresses to same physical page



TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Let’s observe EdDSA ECC key multiplication

```
void _gcry_mpi_ec_mul_point (mpi_point_t result,
 gcry_mpi_t scalar, mpi_point_t point,
 mpi_ec_t ctx)
{
 ...
 for (j=nbits-1; j >= 0; j--) {
  _gcry_mpi_ec_dup_point (result, result, ctx);
  if (mpi_test_bit (scalar, j))
   _gcry_mpi_ec_add_points(result,result,point,ctx);
 }
 ...
}
```

TLBLEED: TLB AS SHARED STATE

- Let's observe EdDSA ECC key multiplication
- Scalar is secret and ADD only happens if there's a 1

```
void _gcry_mpi_ec_mul_point (mpi_point_t result,
 gcry_mpi_t scalar, mpi_point_t point,
 mpi_ec_t ctx)
{
 ...
 for (j=nbits-1; j >= 0; j--) {
  _gcry_mpi_ec_dup_point (result, result, ctx);
  if (mpi_test_bit (scalar, j))
   _gcry_mpi_ec_add_points(result,result,point,ctx);
 }
 ...
}
```

TLBLEED: TLB AS SHARED STATE

- Let's observe EdDSA ECC key multiplication
- Scalar is secret and ADD only happens if there's a 1
- Like RSA square-and-multiply

```
void _gcry_mpi_ec_mul_point (mpi_point_t result,
 gcry_mpi_t scalar, mpi_point_t point,
 mpi_ec_t ctx)
{
 ...
 for (j=nbits-1; j >= 0; j--) {
  _gcry_mpi_ec_dup_point (result, result, ctx);
  if (mpi_test_bit (scalar, j))
   _gcry_mpi_ec_add_points(result,result,point,ctx);
 }
 ...
}
```

TLBLEED: TLB AS SHARED STATE

- Let’s observe EdDSA ECC key multiplication
- Scalar is secret and ADD only happens if there’s a 1
- Like RSA square-and-multiply
- But: we can not use code information! Only data..!

```
void _gcry_mpi_ec_mul_point (mpi_point_t result,
 gcry_mpi_t scalar, mpi_point_t point,
 mpi_ec_t ctx)
{
 ...
 for (j=nbits-1; j >= 0; j--) {
  _gcry_mpi_ec_dup_point (result, result, ctx);
  if (mpi_test_bit (scalar, j))
   _gcry_mpi_ec_add_points(result,result,point,ctx);
 }
 ...
}
```

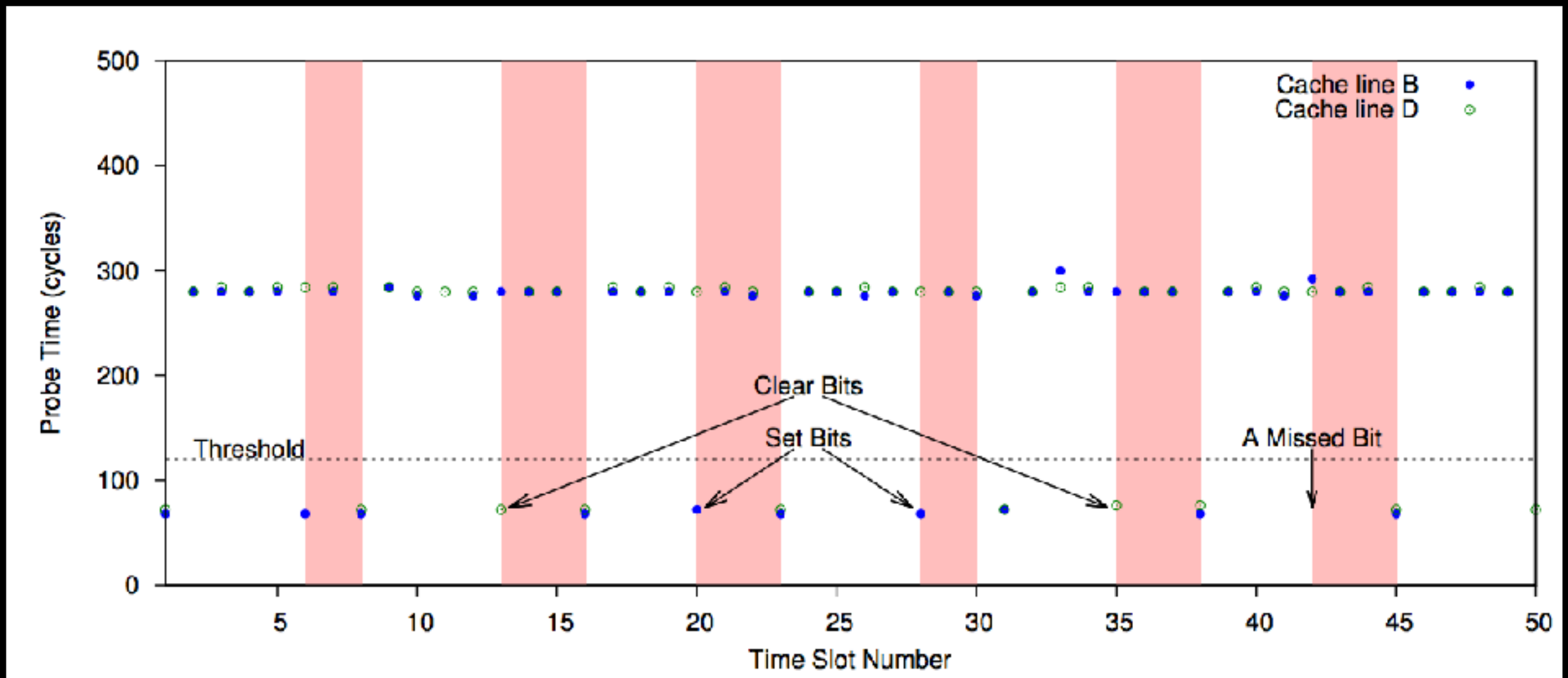
TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Typical cache attack relies on *spatial separation*
- I.E. different cache lines

TLBLEED: TLB AS SHARED STATE

- Typical cache attack relies on *spatial separation*
- I.E. different cache lines



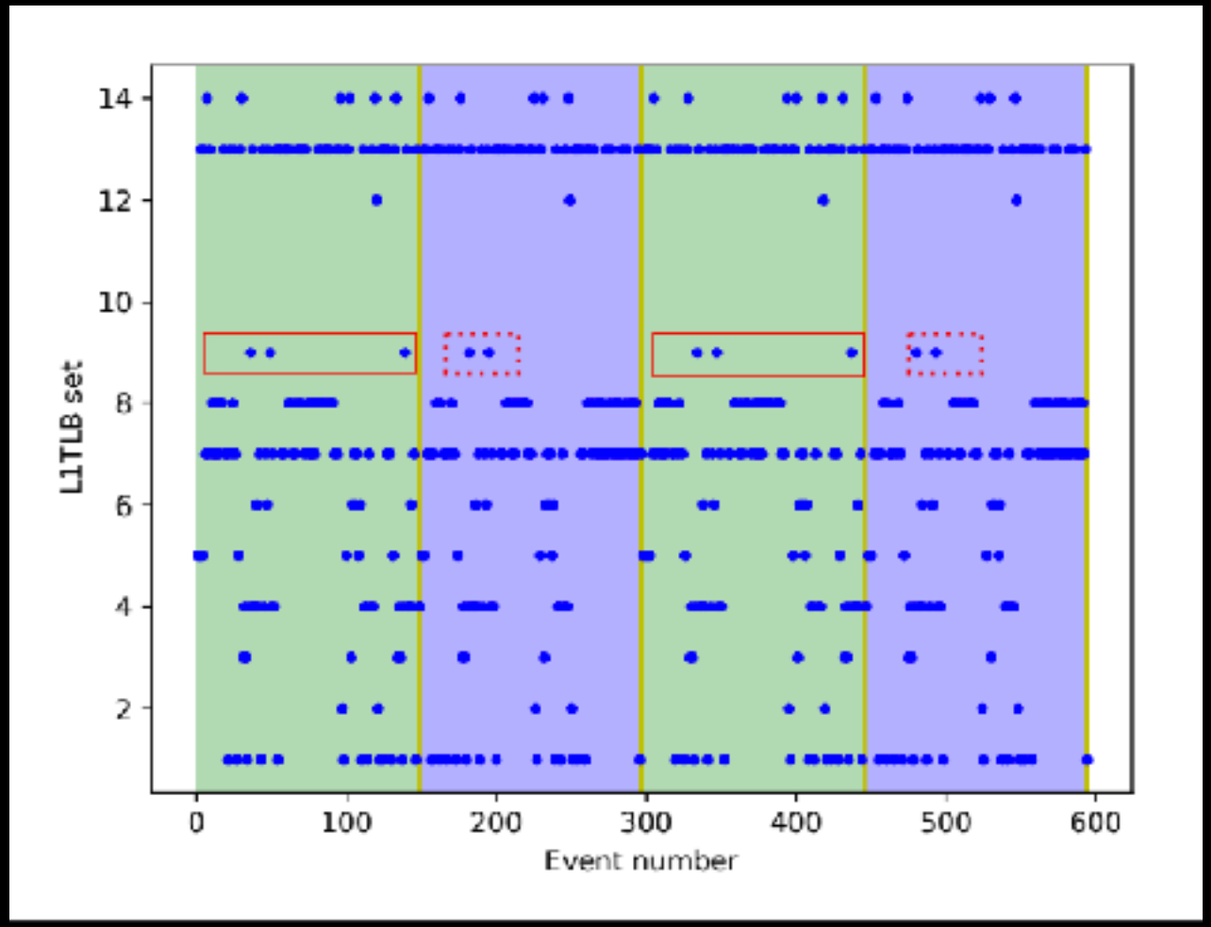
TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

- Let’s find the spatial L1 DTLB separation
- There isn’t any
- Too much activity in both blue/green cases

TLBLEED: TLB AS SHARED STATE

- Let’s find the spatial L1 DTLB separation
- There isn’t any
- Too much activity in both blue/green cases



TLBLEED: TLB AS SHARED STATE

TLBLEED: TLB AS SHARED STATE

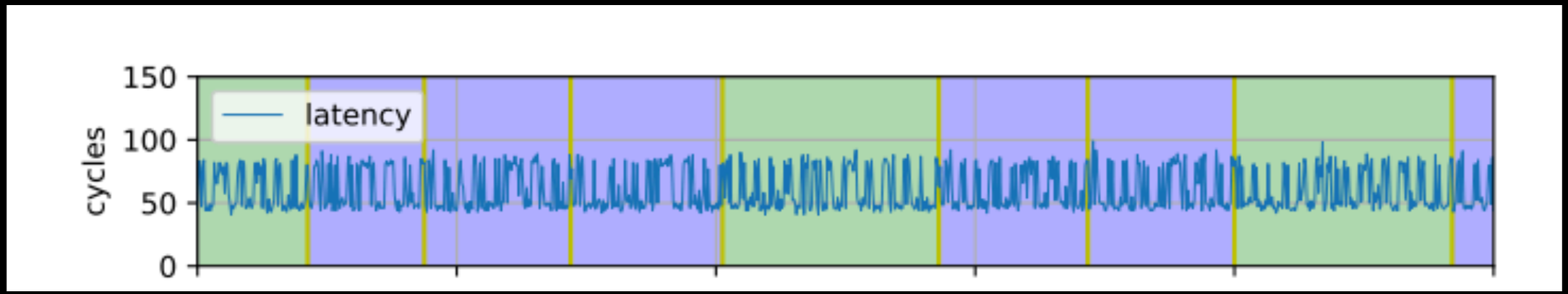
- Monitor a single TLB set and use temporal information

TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference

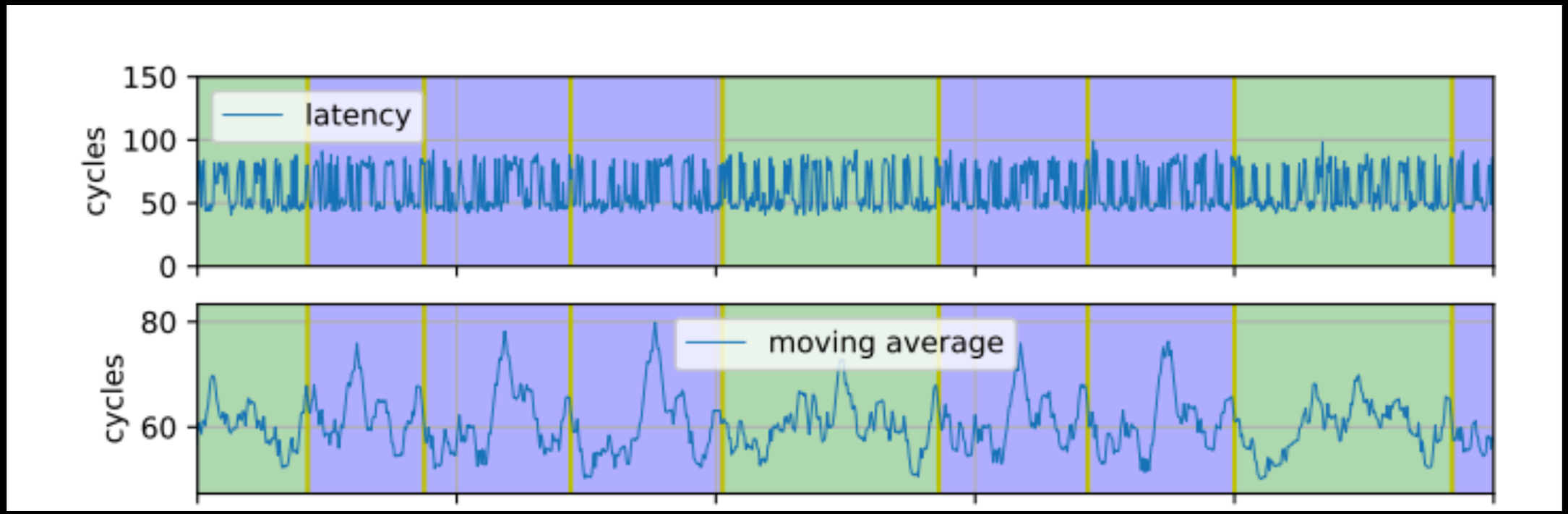
TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference



TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference

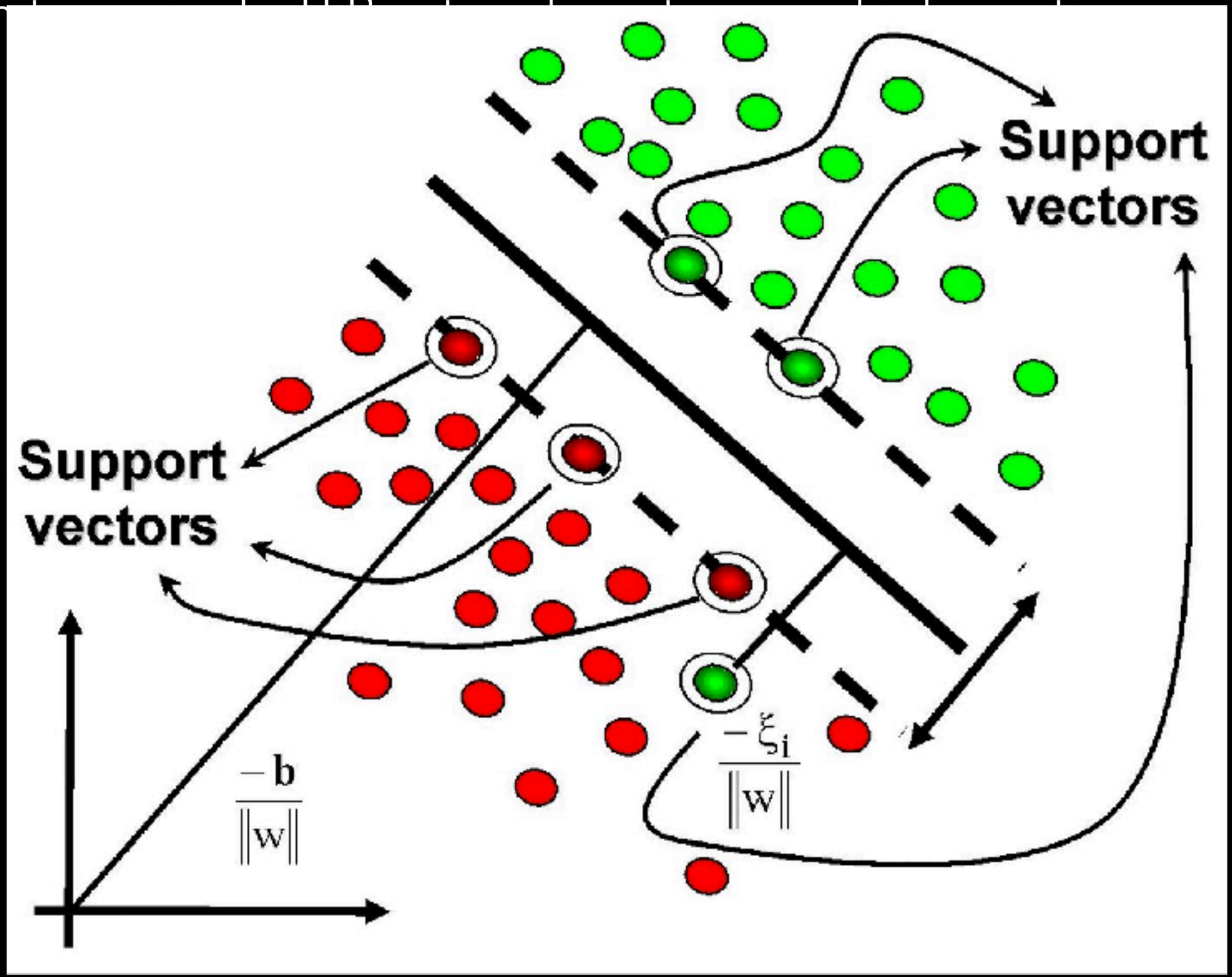


TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference

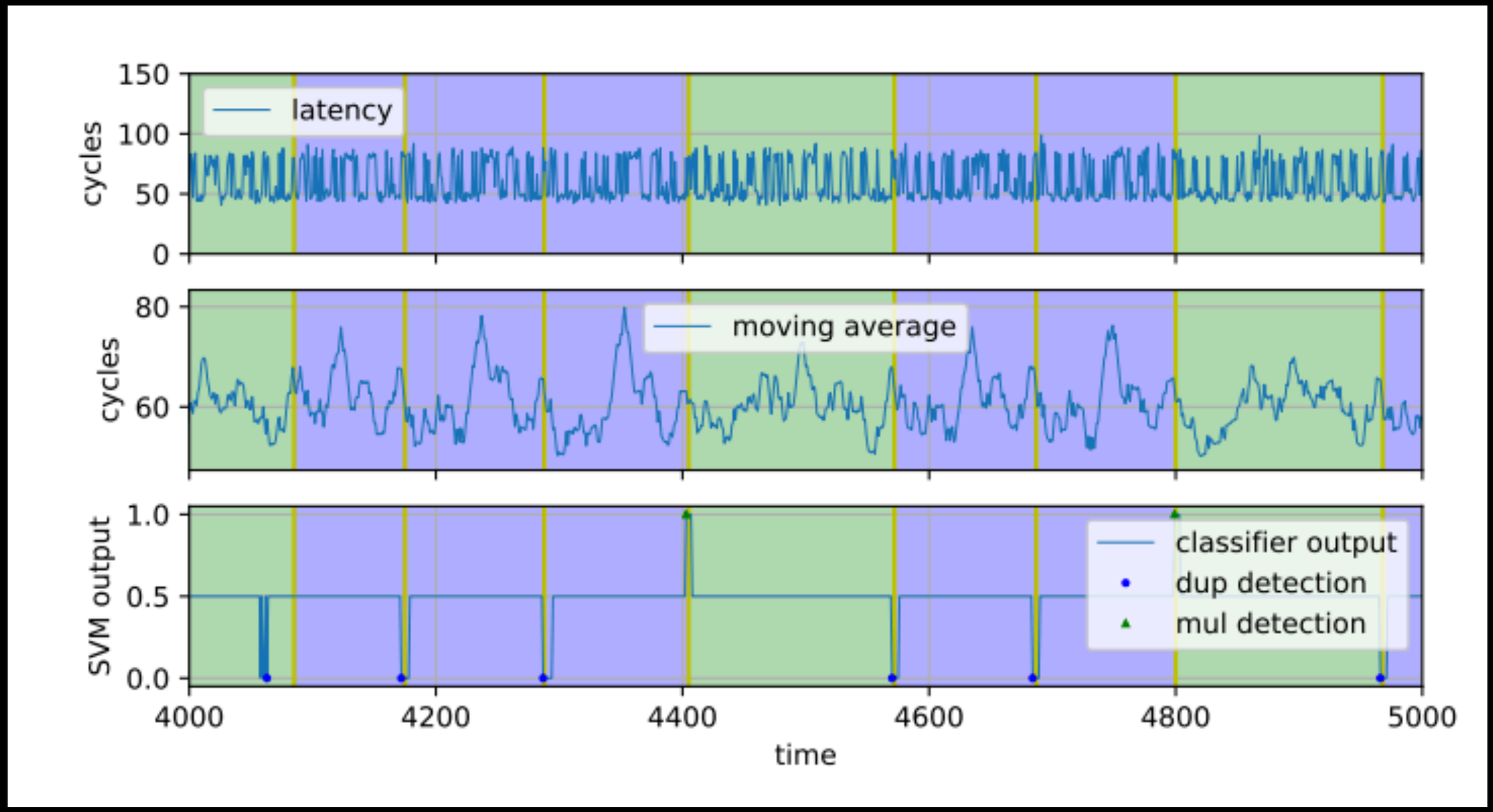
TLBLEED: TLB AS SHARED STATE

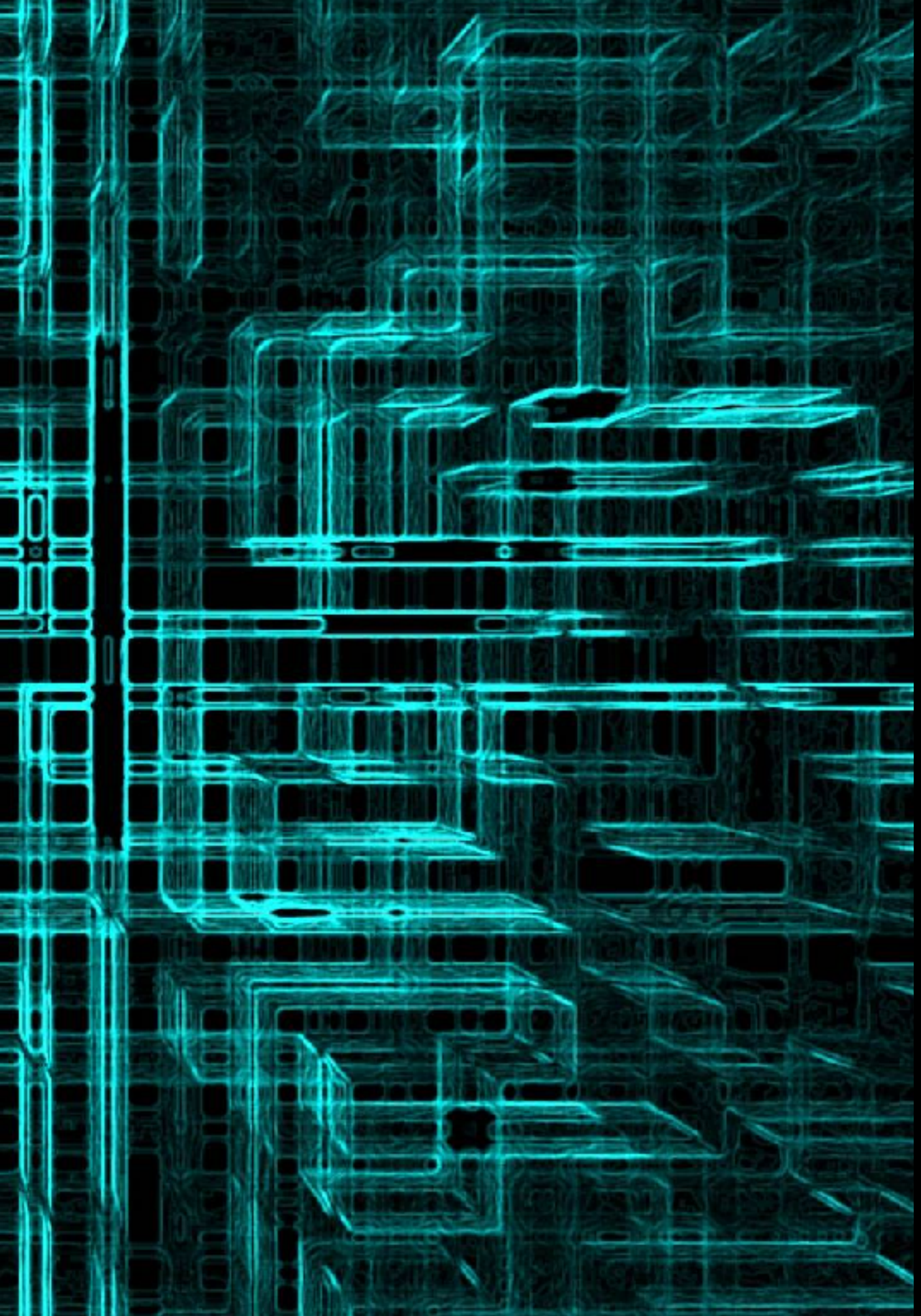
- Mon
- Use



TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference





EVALUATION

TLBLEED RELIABILITY

TLBLEED RELIABILITY

Microarchitecture	Trials	Success	Median BF
Skylake	500	0.998	$2^{1.6}$
Broadwell	500	0.982	$2^{3.0}$
Coffeelake	500	0.998	$2^{2.6}$
Total	1500	0.993	

TLBLEED RELIABILITY

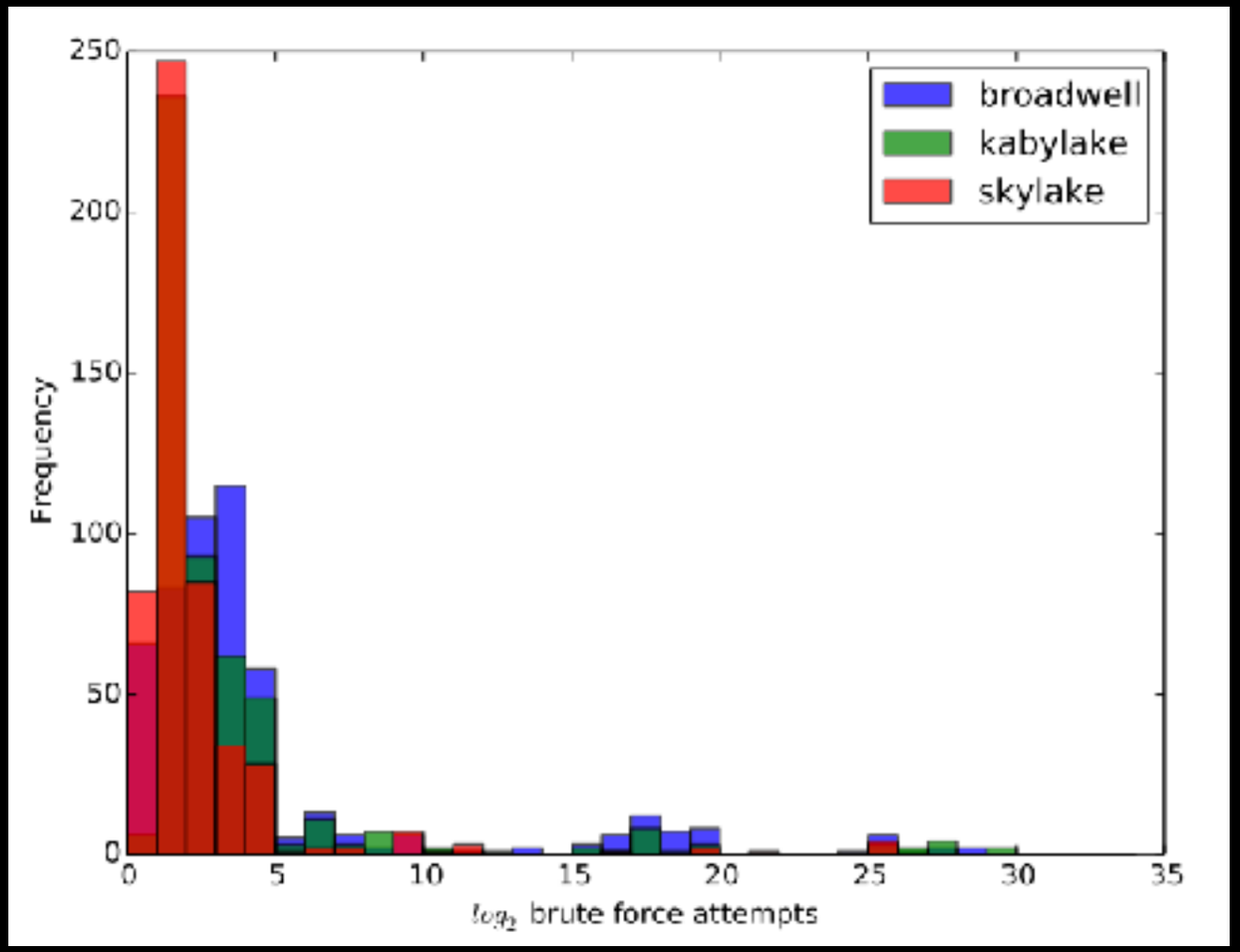
Microarchitecture	Trials	Success	Median BF
Skylake	500	0.998	$2^{1.6}$
Broadwell	500	0.982	$2^{3.0}$
Coffeelake	500	0.998	$2^{2.6}$
Total	1500	0.993	

Microarchitecture	Trials	Success	Median BF
Broadwell (CAT)	50	0.94	2^{12}
Broadwell	500	0.982	$2^{3.0}$

TLBLEED RELIABILITY

Microarchitecture	Trials	Success	Median BF
Skylake	500	0.998	$2^{1.6}$
Broadwell	500	0.982	$2^{3.0}$
Coffeelake	500	0.998	$2^{2.6}$
Total	1500	0.993	

Microarchitecture	Trials	Success	Median BF
Broadwell (CAT)	50	0.94	2^{12}
Broadwell	500	0.982	$2^{3.0}$



CREDIT

- Work also by Kaveh Razavi, Cristiano Giuffrida, Herbert Bos
- Some diagrams in these slides were taken from other work: FLUSH+RELOAD, Prefetch, DRAMA
- Yuval Yarom, Katrina Falkner, Peter Peßl, Daniel Gruss

CONCLUSION

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defences

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defences
- @bjg @gober

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defences

- @bjg @gober
- @vu5ec

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defences

- @bjg @gober
- @vu5ec
- www.vusec.net

CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defences

- @bjg @gober
- @vu5ec
- www.vusec.net

- Thank you for listening