Security:
Can we afford to have it?
Can we afford not to have it?

Daniel Gruss

2024-10-21

Graz University of Technology

Acknowledgments

This research was made possible by generous funding from:

European Research Council (ERC project FSSec 101076409), FWF SFB project SPyCoDe F8504, NSF grants 1813004, 2217020, 2316201, and research grants and gifts from Red Hat, Google, Intel, and Cisco. This work has benefitted from Dagstuhl Seminar 22341 (PEACHES). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.


side channel
= obtaining meta-data and
deriving secrets from it
CHANGE MY MIND
imgflip.com

Side Channel or not?

Side Channel or not?

- Profiling cache utilization with performance counters?

Side Channel or not?

- Profiling cache utilization with performance counters? $\rightarrow$ No

Side Channel or not?

- Profiling cache utilization with performance counters? $\rightarrow$ No
- Observing cache utilization with performance counters and using it to infer a crypto key?

Side Channel or not?

- Profiling cache utilization with performance counters? $\rightarrow$ No
- Observing cache utilization with performance counters and using it to infer a crypto key? $\rightarrow$ Yes

Side Channel or not?

- Profiling cache utilization with performance counters? $\rightarrow$ No
- Observing cache utilization with performance counters and using it to infer a crypto key? $\rightarrow$ Yes
- Measuring memory access latency with Flush+Reload?

Side Channel or not?

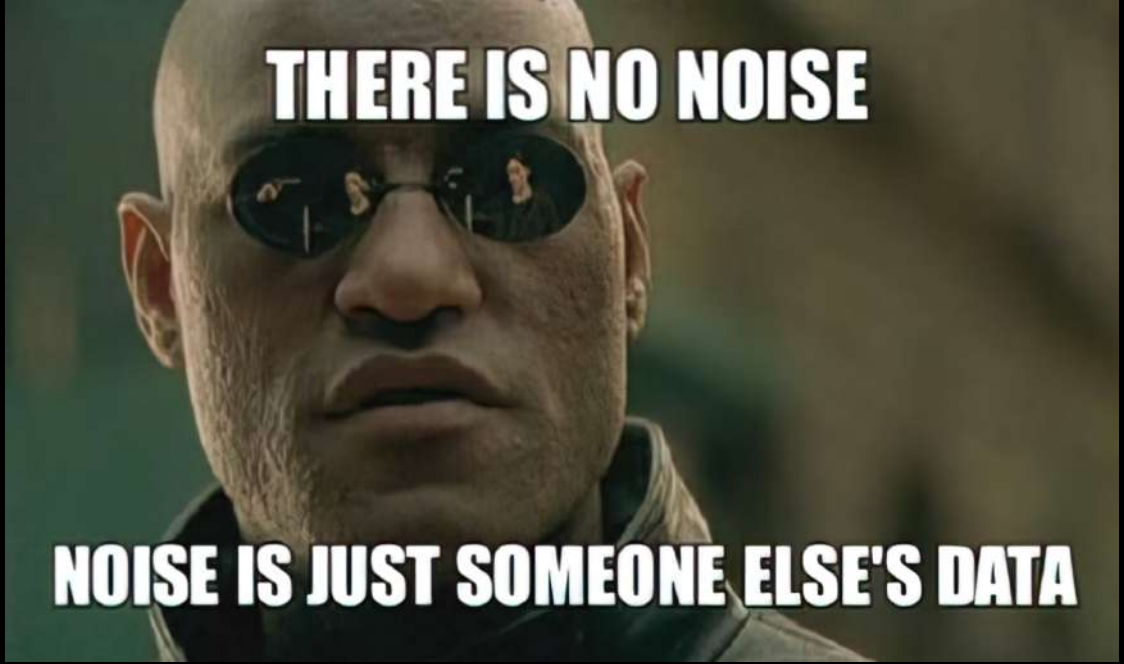
- Profiling cache utilization with performance counters? $\rightarrow$ No
- Observing cache utilization with performance counters and using it to infer a crypto key? $\rightarrow$ Yes
- Measuring memory access latency with Flush+Reload? $\rightarrow$ No

Side Channel or not?

- Profiling cache utilization with performance counters? $\rightarrow$ No
- Observing cache utilization with performance counters and using it to infer a crypto key? $\rightarrow$ Yes
- Measuring memory access latency with Flush+Reload? $\rightarrow$ No
- Measuring memory access latency with Flush+Reload and using it to infer keystroke timings?

Side Channel or not?

- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes
- Measuring memory access latency with Flush+Reload? → No
- Measuring memory access latency with Flush+Reload and using it to infer keystroke timings? → Yes


THERE IS NO NOISE
NOISE IS JUST SOMEONE ELSE'S DATA

1337 4242

FOOD CACHE

Revolutionary concept!

Store your food at home, never go to the grocery store during cooking.

Can store **ALL** kinds of food.

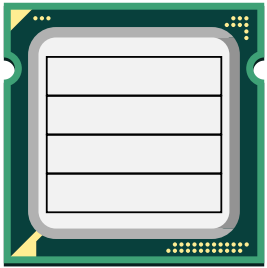
ONLY TODAY INSTEAD OF ~~$1,300~~

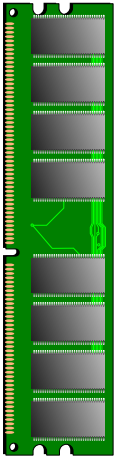
$1,299

ORDER VIA PHONE: +555 12345

CPU Cache


```
printf("%d", i);
printf("%d", i);
```

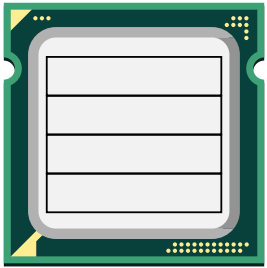


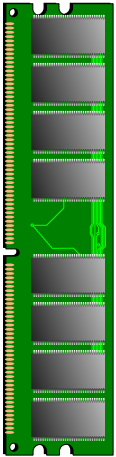


CPU Cache


```
printf("%d", i);
printf("%d", i);
```

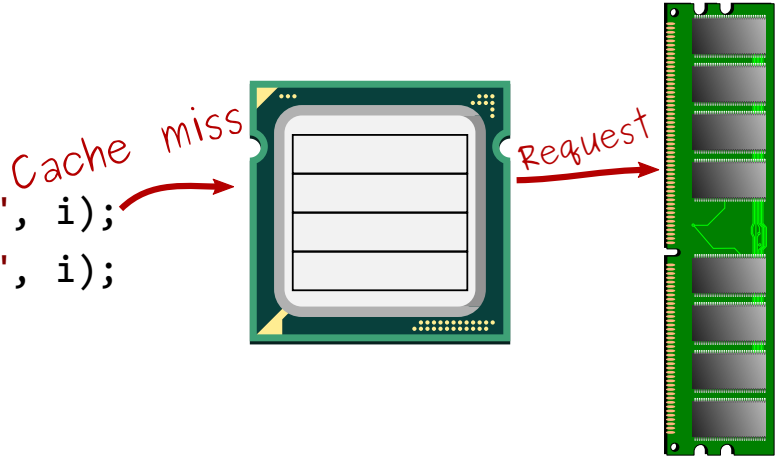
Cache miss





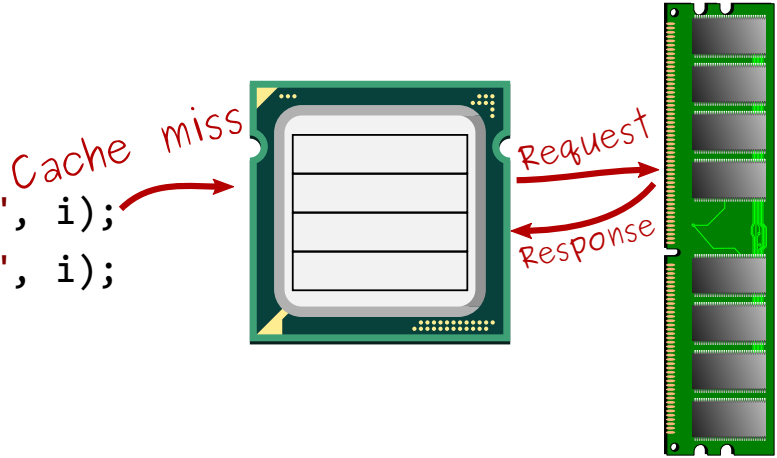
CPU Cache

```
printf("%d", i);
printf("%d", i);
```



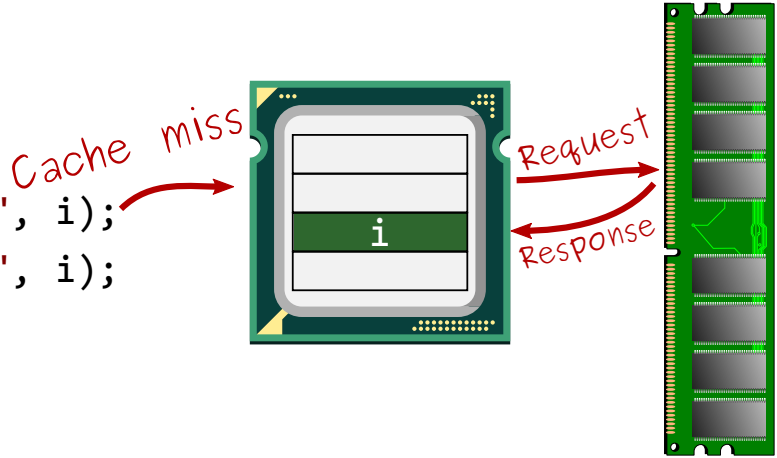
CPU Cache

```
printf("%d", i);
printf("%d", i);
```



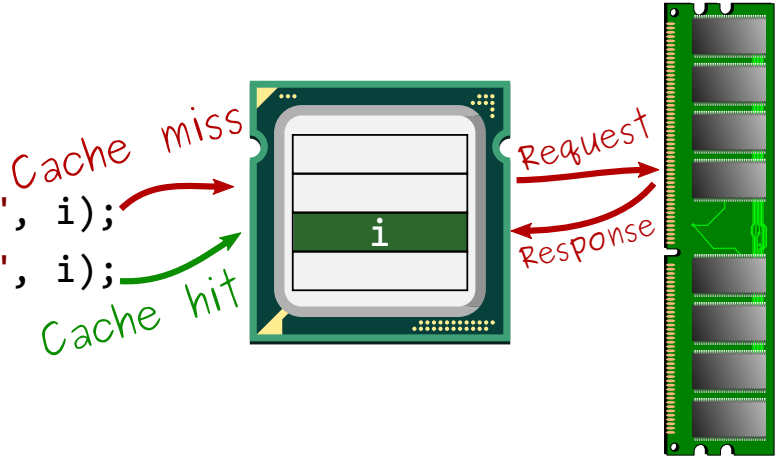
CPU Cache

```
printf("%d", i);
printf("%d", i);
```



CPU Cache

```
printf("%d", i);
printf("%d", i);
```



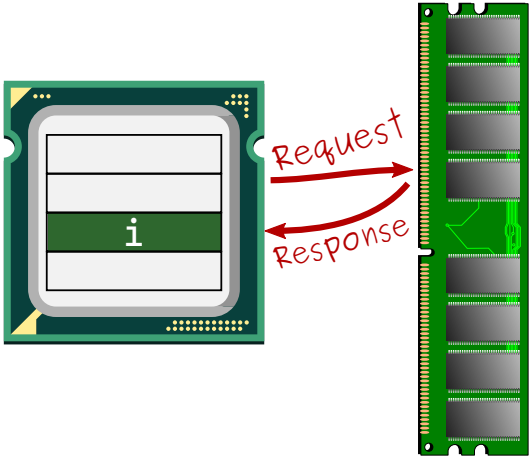
CPU Cache

DRAM access, slow

```
printf("%d", i);
printf("%d", i);
```

Cache miss

Cache hit



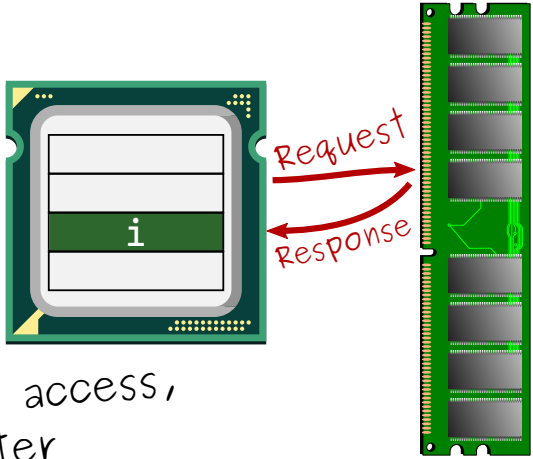
CPU Cache

DRAM access, slow

```
printf("%d", i);
printf("%d", i);
```

Cache miss

Cache hit



No DRAM access, much faster

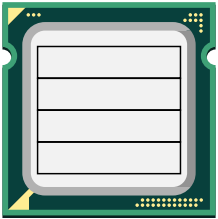
Flush+Reload

Shared Memory

ATTACKER

flush

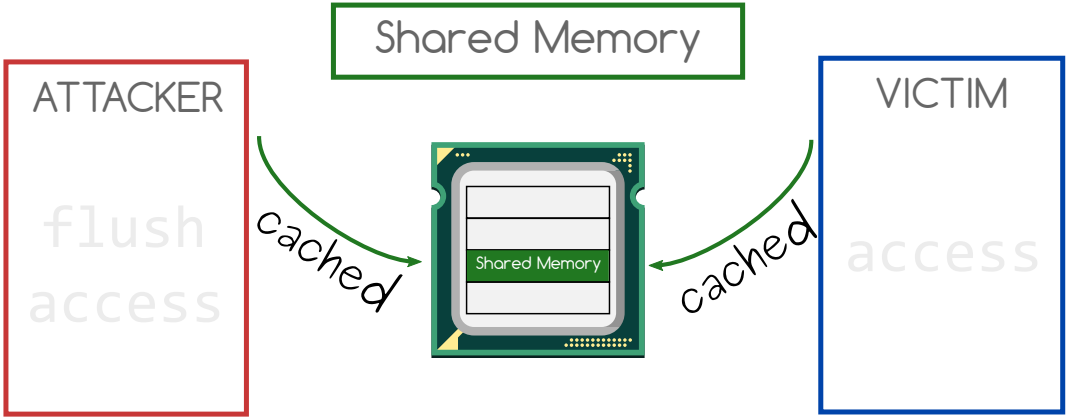
access



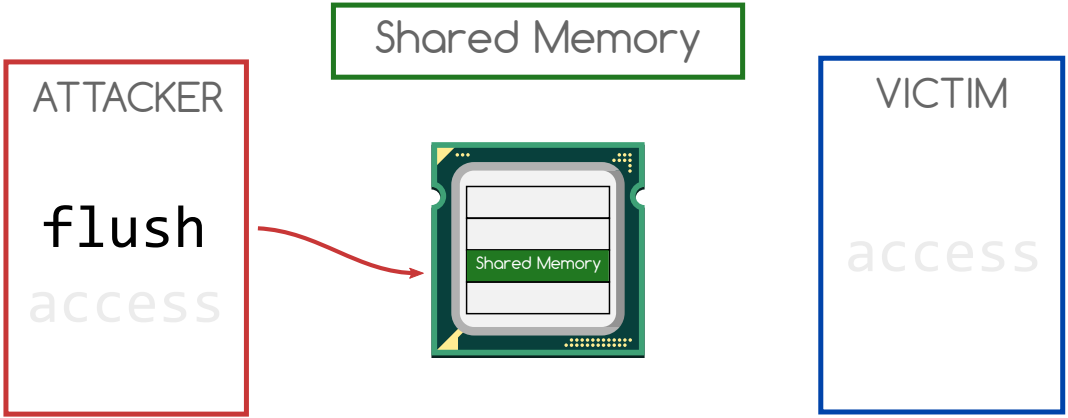
VICTIM

access

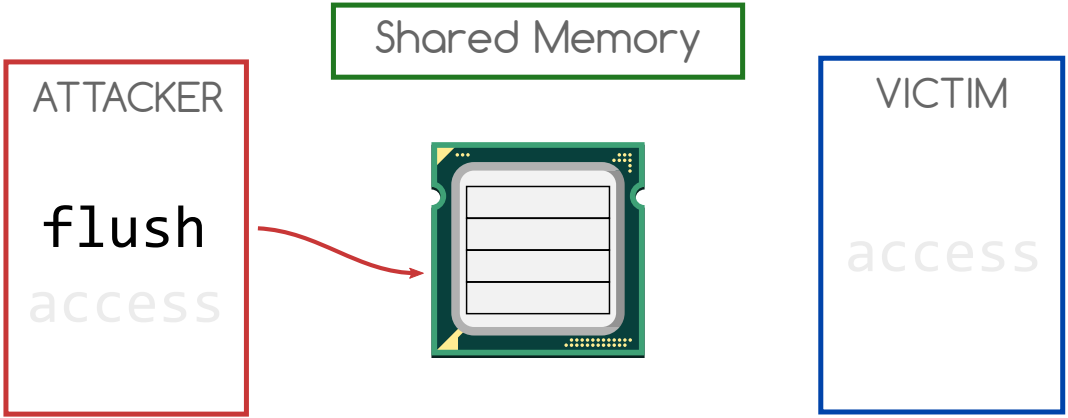
Flush+Reload



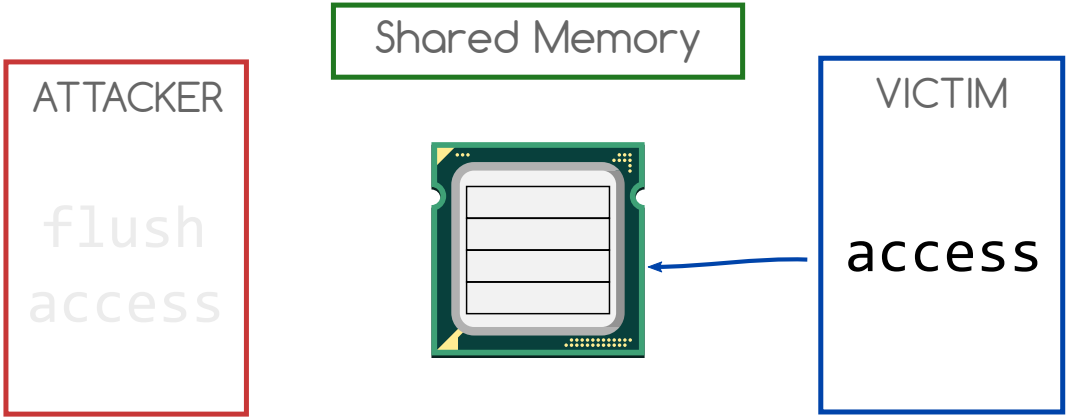
Flush+Reload



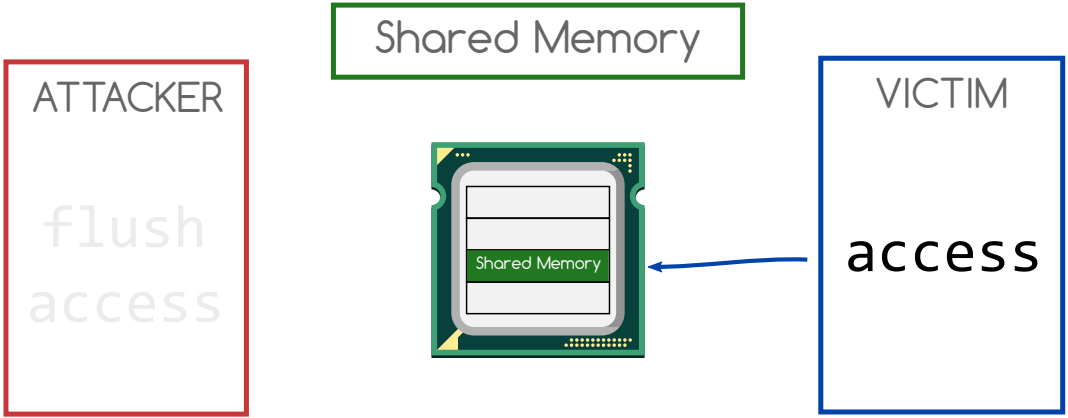
Flush+Reload



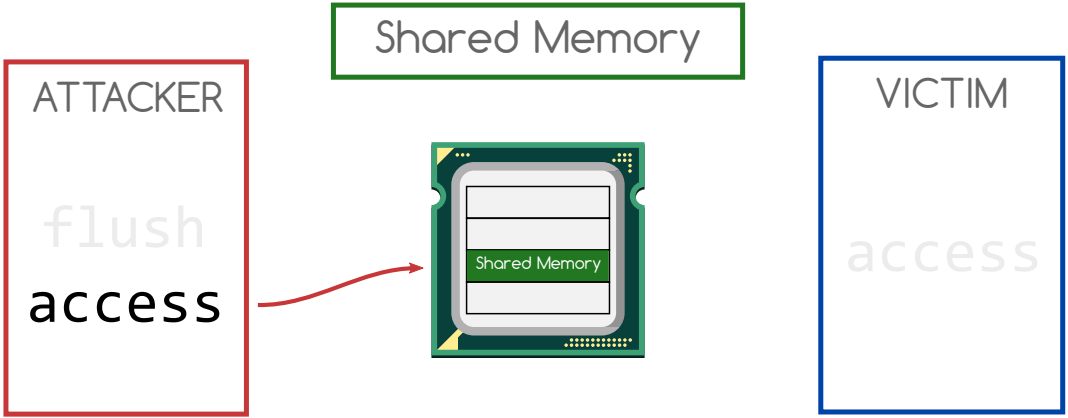
Flush+Reload



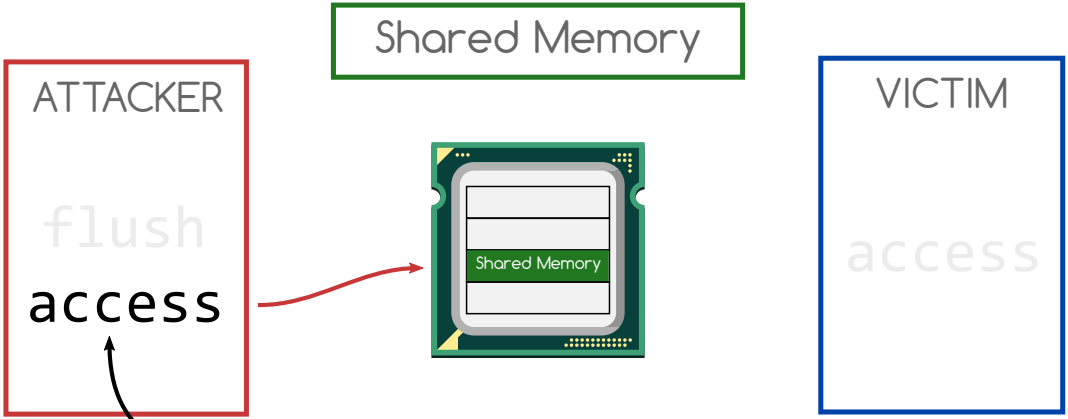
Flush+Reload



Flush+Reload

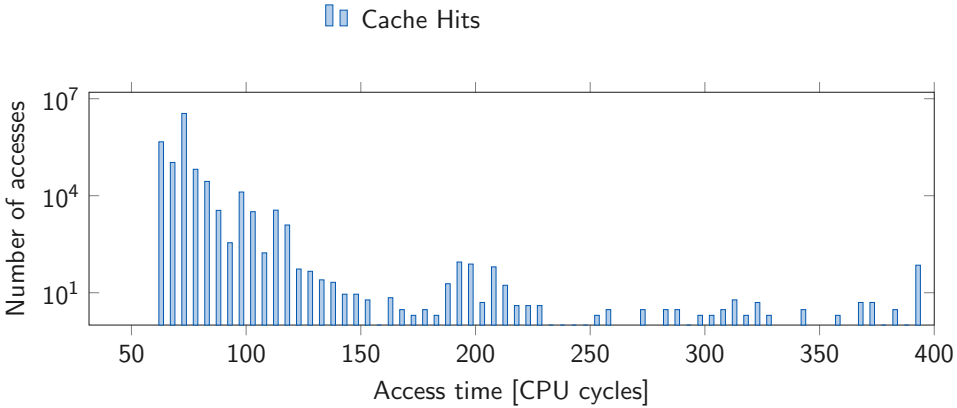


Flush+Reload

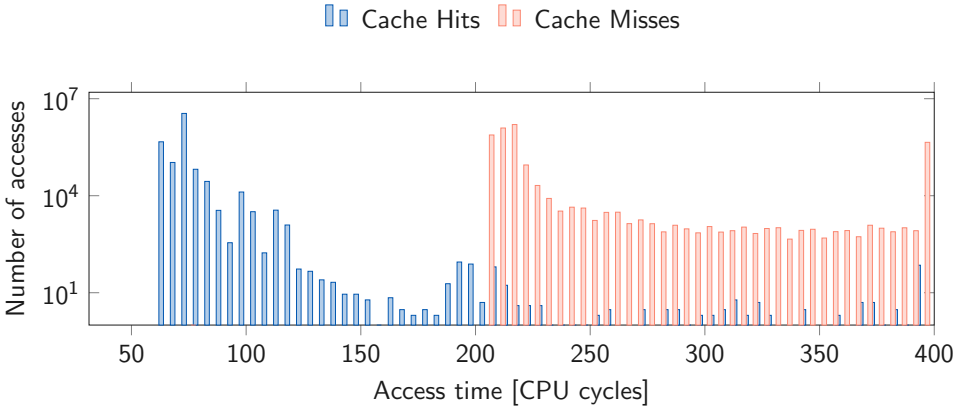


fast if victim accessed data,
slow otherwise

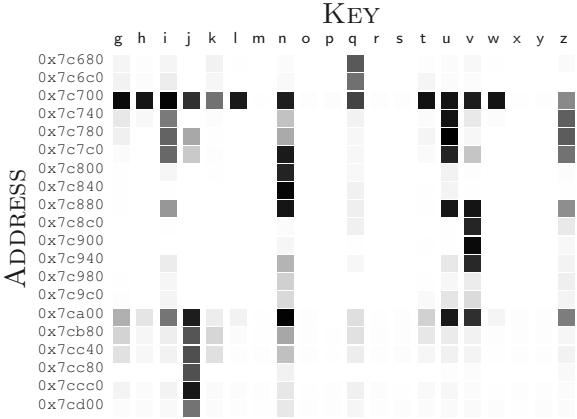
Memory Access Latency



Memory Access Latency



Cache Template



Building Meltdown

- Add a layer of indirection to test

```
char data = *(char*) 0xffffffff81a000e0;
array[data * 4096] = 0;
```

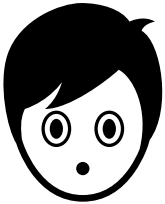
Building Meltdown

- Add a layer of indirection to test

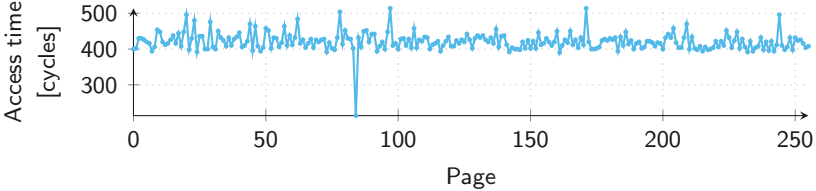
```
char data = *(char*) 0xffffffff81a000e0;
array[data * 4096] = 0;
```

- Then check whether any part of `array` is cached

Building Meltdown

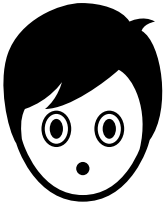


- Flush+Reload over all pages of the array

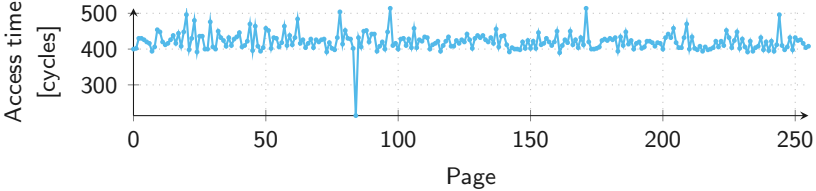


- Index of cache hit reveals data

Building Meltdown



- Flush+Reload over all pages of the array



- Index of cache hit reveals data
- Permission check is in some cases not fast enough


Kernel Address Isolation to have Side channels Efficiently Removed

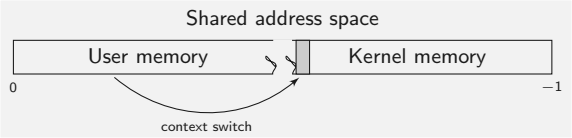
KAISER /ˈkʌɪzə/

1. [german] Emperor, ruler of an empire
2. largest penguin, emperor penguin

Kernel **A**ddress **I**solation to have **S**ide channels **E**fficiently **R**emoved

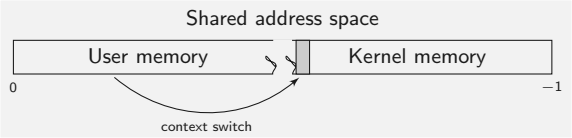
KAISER Illustration

Without KAISER:

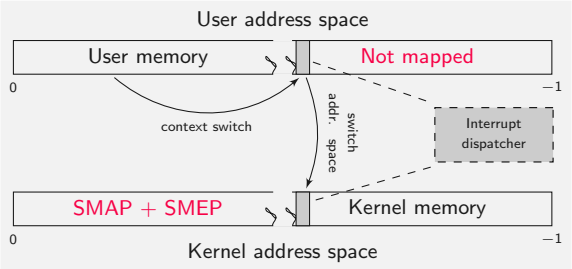


KAISER Illustration

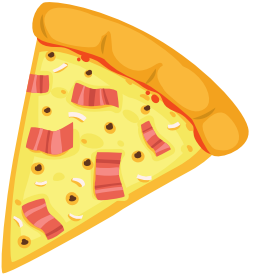
Without KAISER:

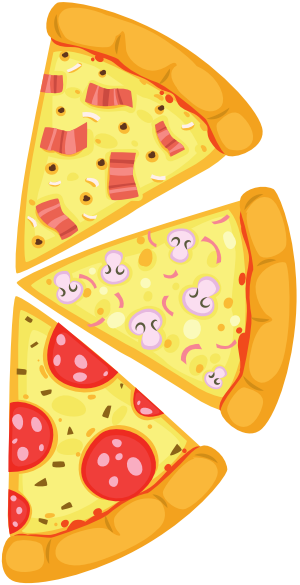


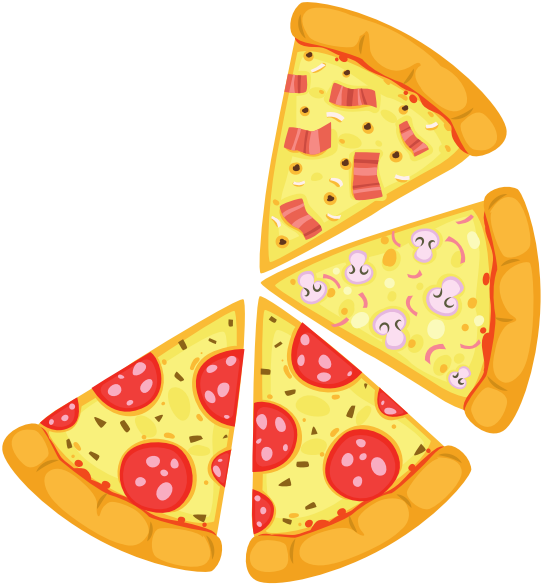
With KAISER:

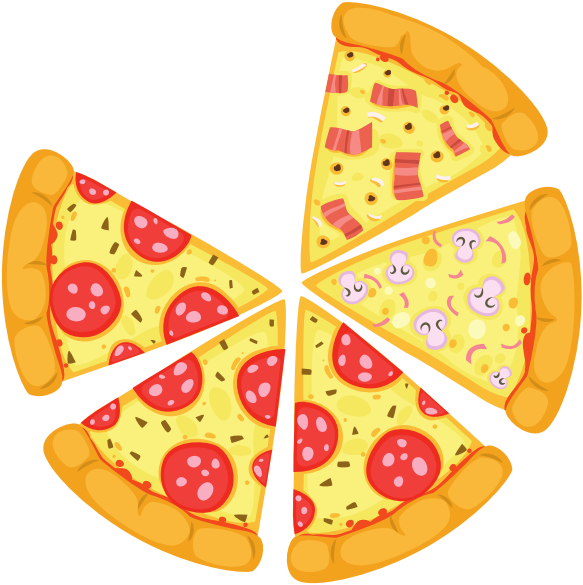


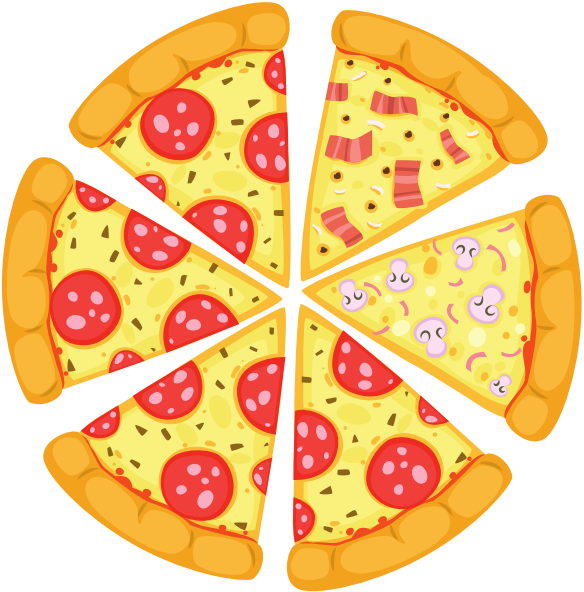

PIZZA
SPECIAL RECIPES












»A table for 6 please«

Speculative Cooking


»A table for 6 please«


PIZZA
SPECIAL RECIPES


PIZZA
SPECIAL RECIPES


Pizza

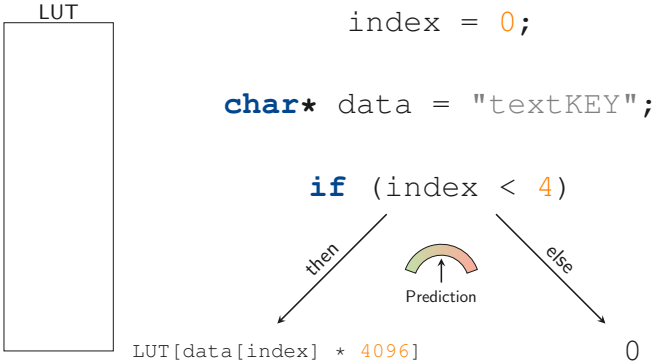

PIZZA
SPECIAL RECIPES


Pizza

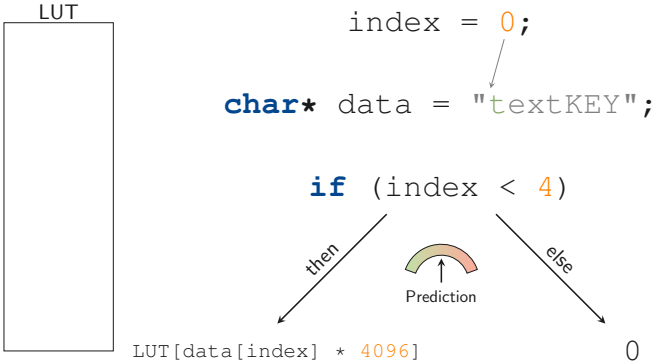

PIZZA
SPECIAL RECIPES


Pizza

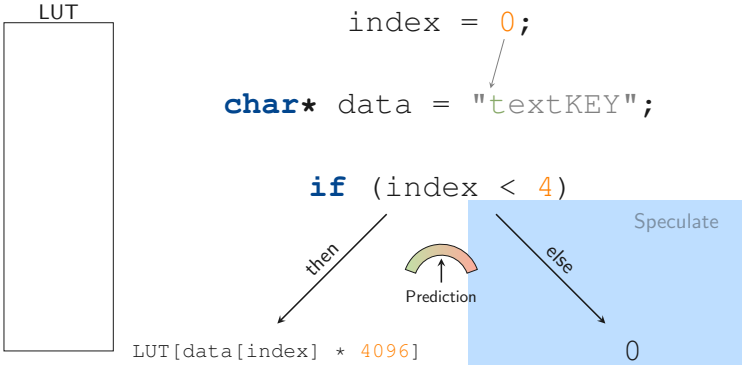
Spectre-PHT (v1)



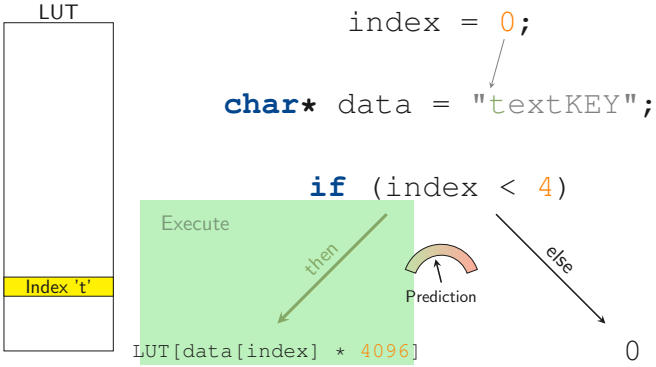
Spectre-PHT (v1)



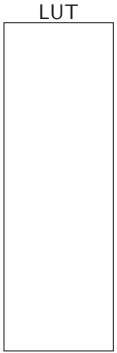
Spectre-PHT (v1)



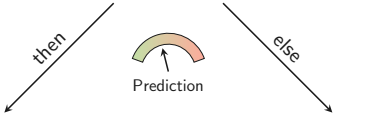
Spectre-PHT (v1)



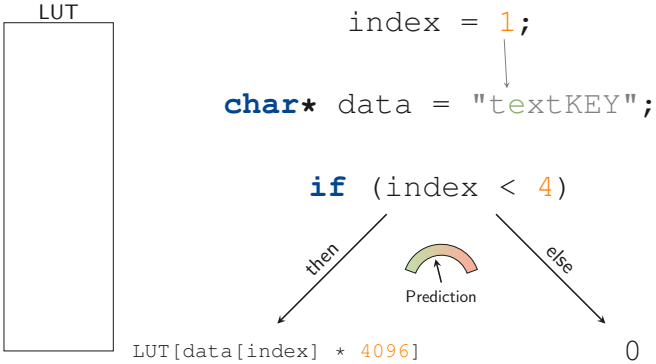
Spectre-PHT (v1)



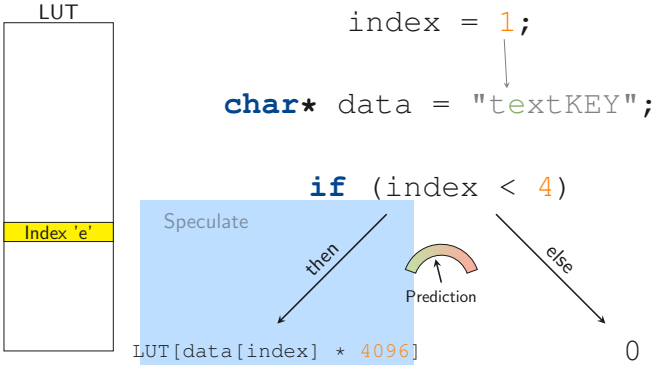

```
index = 1;

char* data = "textKEY";

if (index < 4)
```



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)


```
index = 1;

char* data = "textKEY";

if (index < 4)
```

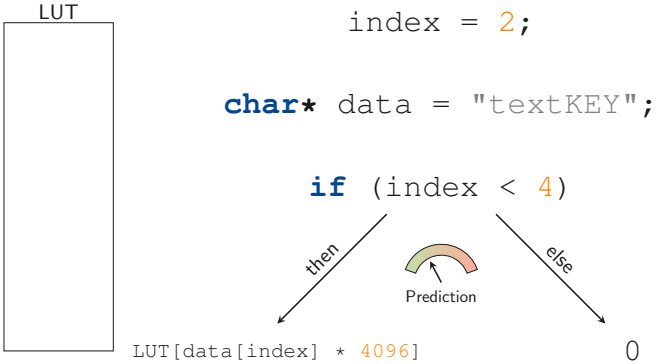
then

else

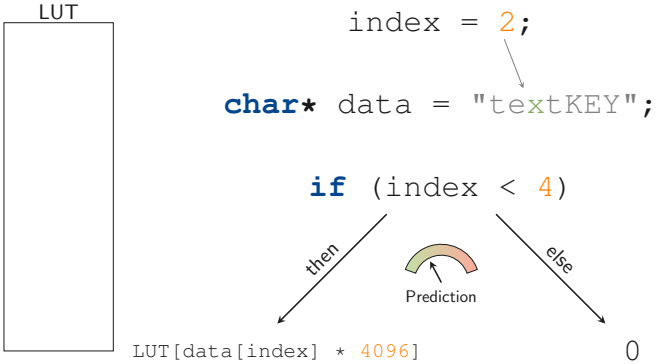
```
LUT[data[index] * 4096]
```

0

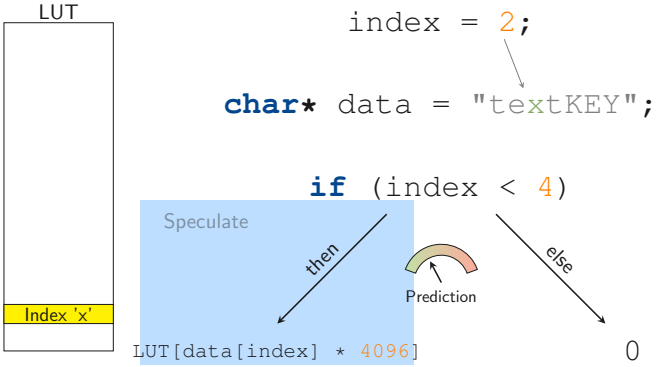
Spectre-PHT (v1)



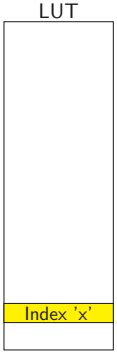
Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)




```
index = 2;

char* data = "textKEY";

if (index < 4)
```

then

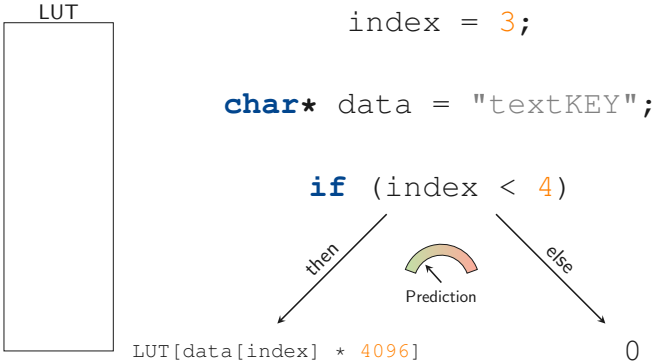
Prediction

else

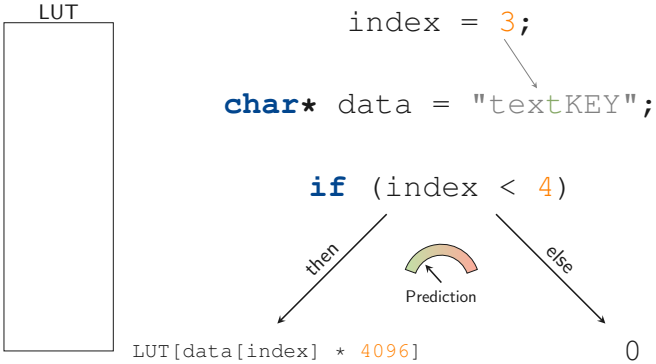
```
LUT[data[index] * 4096]
```

0

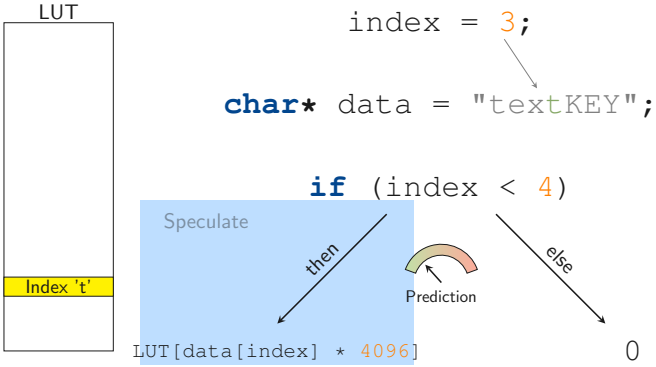
Spectre-PHT (v1)



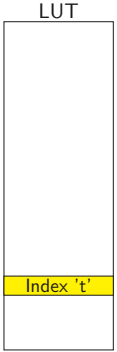
Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)




```
index = 3;

char* data = "textKEY";

if (index < 4)
```

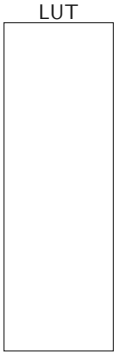
then

else

```
LUT[data[index] * 4096]
```

0

Spectre-PHT (v1)




```
index = 4;

char* data = "textKEY";

if (index < 4)
```

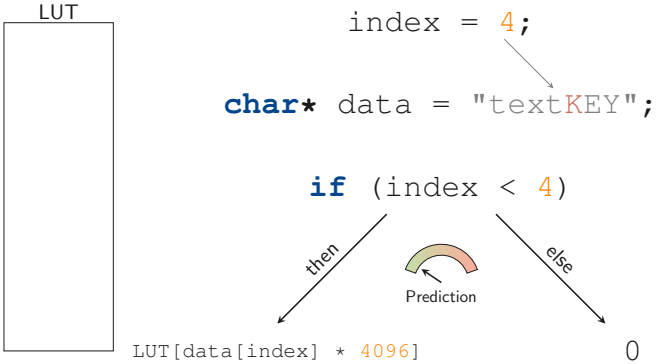
then

else

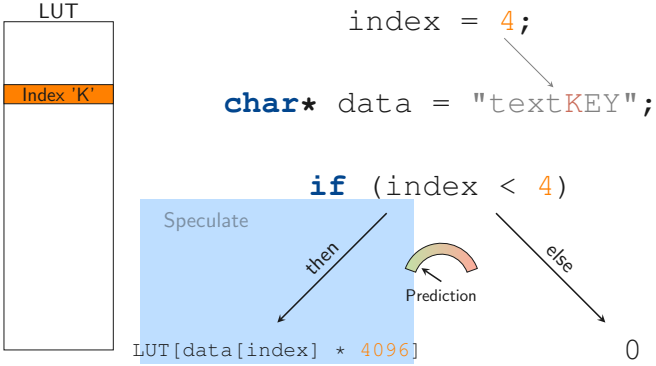
```
LUT[data[index] * 4096]
```

0

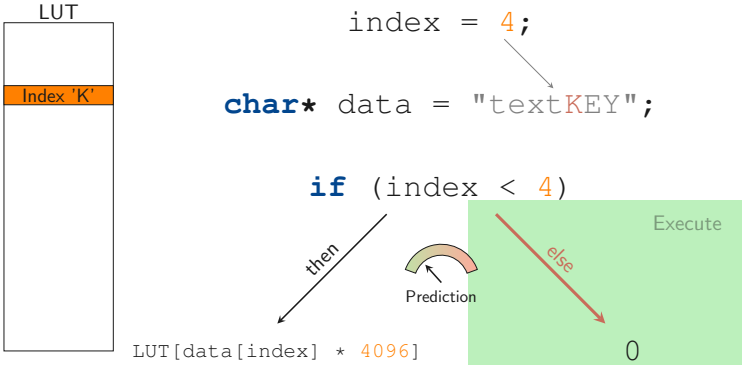
Spectre-PHT (v1)



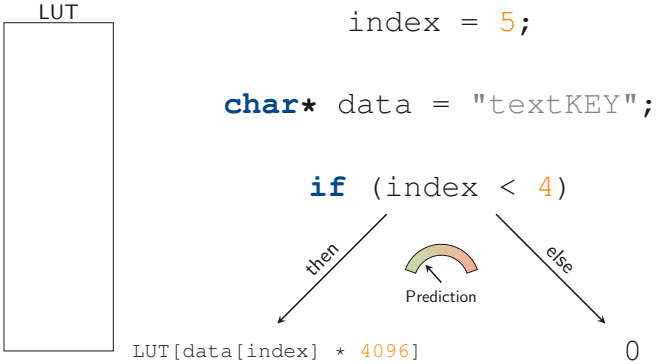
Spectre-PHT (v1)



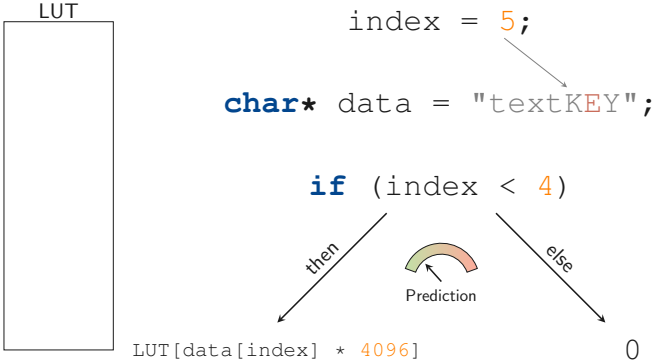
Spectre-PHT (v1)



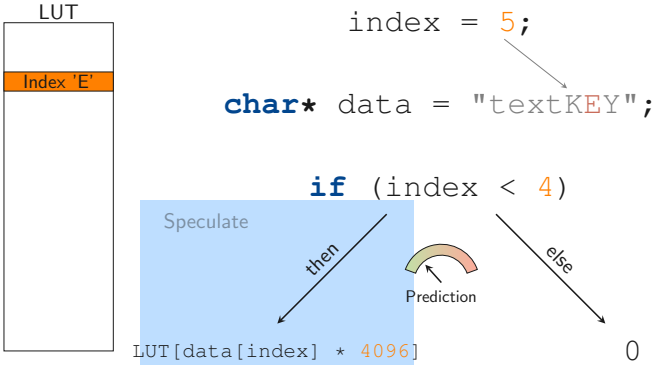
Spectre-PHT (v1)



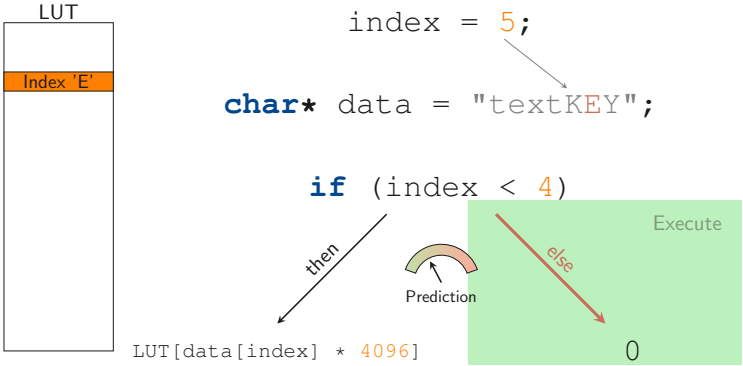
Spectre-PHT (v1)



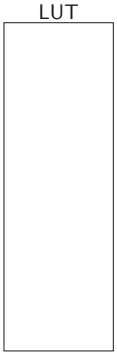
Spectre-PHT (v1)



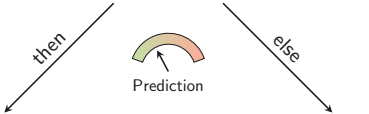
Spectre-PHT (v1)



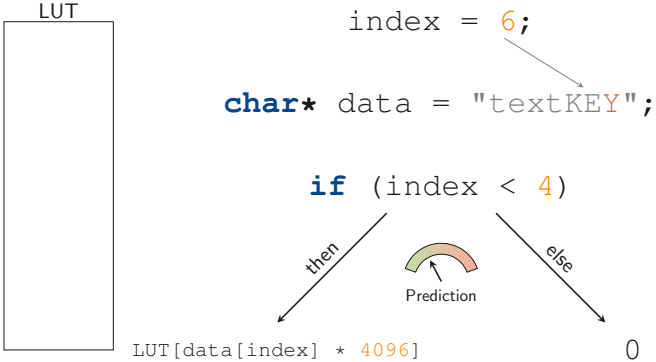
Spectre-PHT (v1)



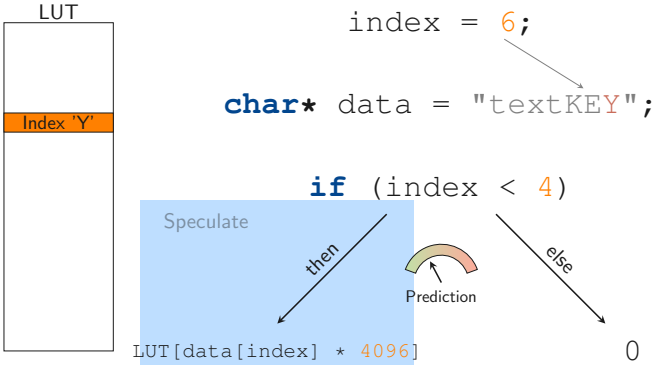

```
index = 6;

char* data = "textKEY";

if (index < 4)
```



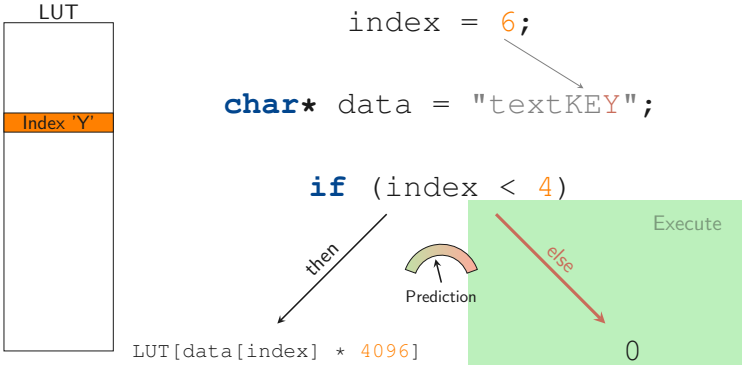
Spectre-PHT (v1)

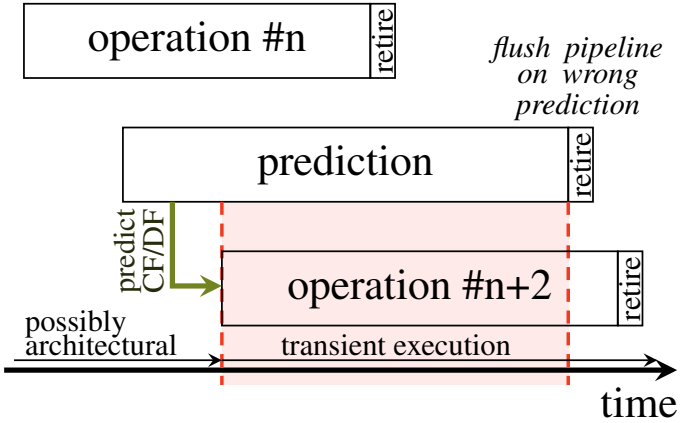


Spectre-PHT (v1)

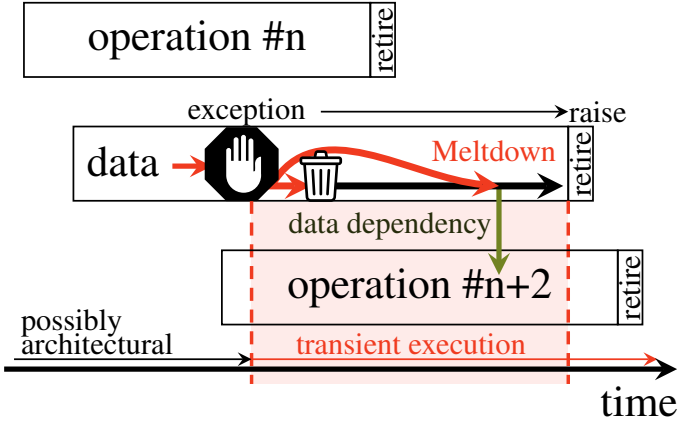


Spectre-PHT (v1)

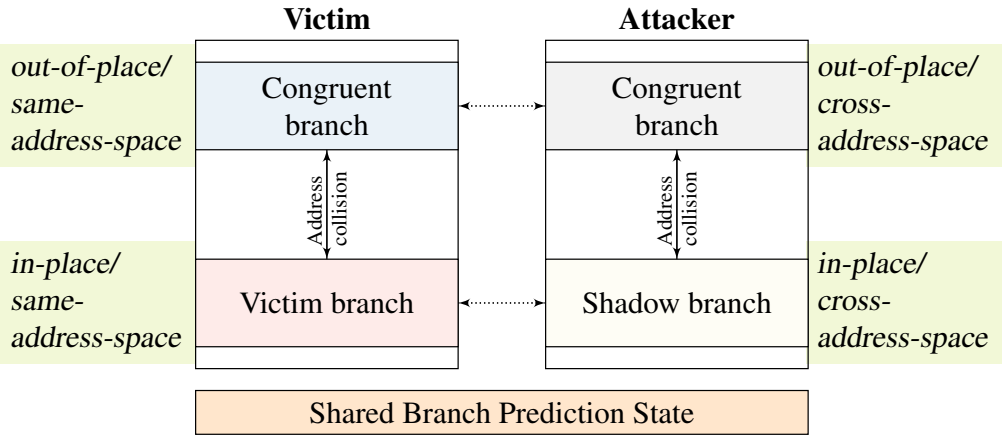



operation #n
retire
flush pipeline on wrong prediction
prediction
retire
predict CF/DF
operation #n+2
retire
possibly architectural
transient execution
time

Meltdown


operation #n
retire
exception
raise
data
Meltdown
retire
data dependency
operation #n+2
retire
possibly
architectural
transient execution
time

Mistraining Location



Performance Costs

Table 1: Reported performance impacts of countermeasures

Defense \ Impact	Performance Loss	Benchmark
InvisiSpec	22%	SPEC
SafeSpec	3% (improvement)	SPEC2017 on MARSSx86
DAWG	2–12%, 1–15%	PARSEC, GAPBS
RSB Stuffing	no reports	
Retpoline	5–10%	real-world workload servers
Site Isolation	only memory overhead	
SLH	36.4%, 29%	Google microbenchmark suite
YSNB	60%	Phoenix
IBRS	20–30%	two sysbench 1.0.11 benchmarks
STIPB	30– 50%	Rodinia OpenMP, DaCapo
IBPB	no individual reports	
Serialization	62%, 74.8%	Google microbenchmark suite
SSBD/SSBB	2–8%	SYSmark®2014 SE & SPEC integer
KAISER/KPTI	0–2.6%	system call rates
L1TF mitigations	-3–31%	various SPEC

Test - Mozilla Firefox (on lab02)

Test

file:///home/dgruss/rowhammerjs/rowhammer.html

Search

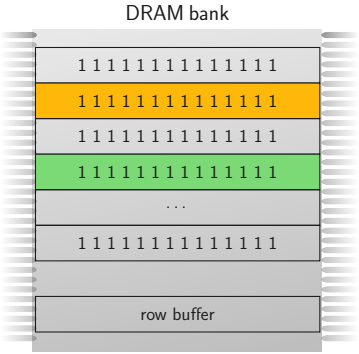
320: 12
330: 9
340: 1
350: 0
360: 1
370: 2
380: 199
390: 76
400: 72
410: 231
420: 572
1250
[!] Found flip (254 != 255) at array index 340021386 when hammering indices 339881984 and 340156416

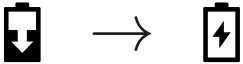
[!] Found flip (239 != 255) at array index 340022176 when hammering indices 339881984 and 340156416

[!] Found flip (191 != 255) at array index 340023138 when hammering indices 339881984 and 340156416

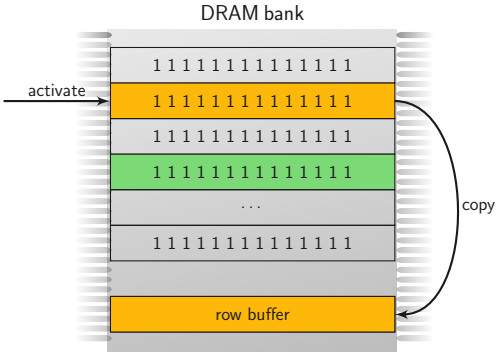
[!] Found flip (254 != 255) at array index 340025146 when hammering indices 339881984 and 340156416

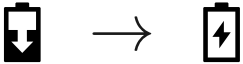
Rowhammer


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
...
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
row buffer

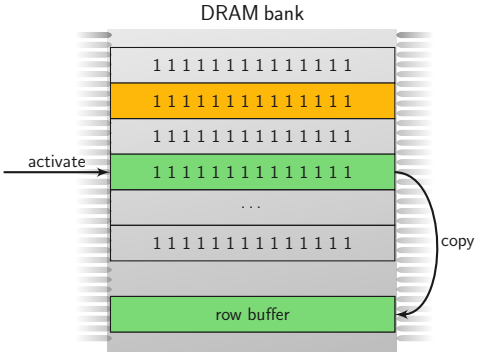


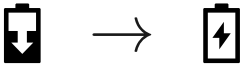
Rowhammer


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
copy
row buffer



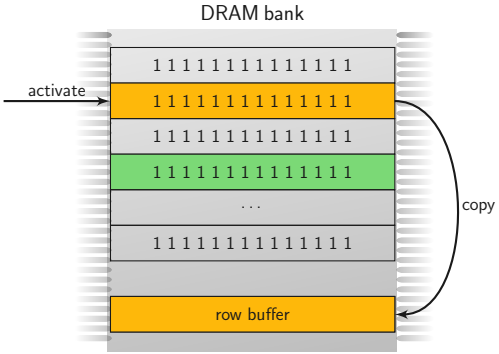
Rowhammer

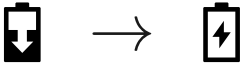




Cells leak faster upon proximate accesses → Rowhammer

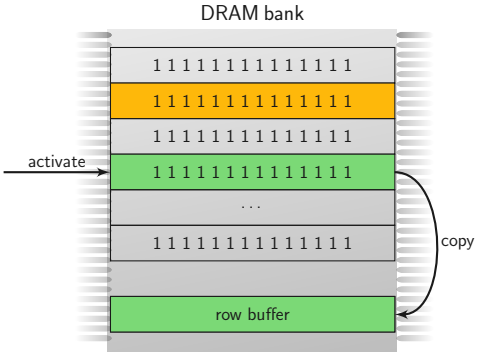
Rowhammer

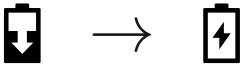




Cells leak faster upon proximate accesses → Rowhammer

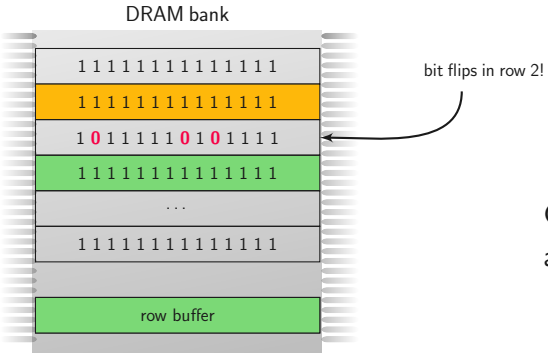
Rowhammer

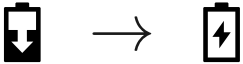




Cells leak faster upon proximate accesses → Rowhammer

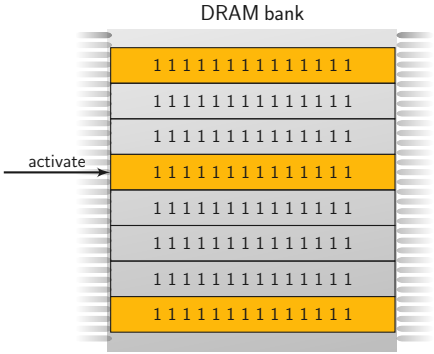
Rowhammer



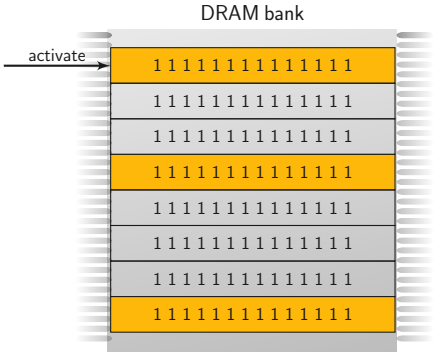


Cells leak faster upon proximate accesses → Rowhammer

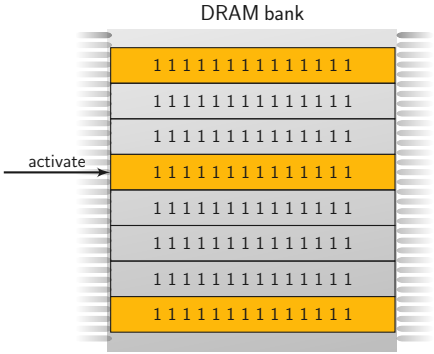
#1 - Single-sided hammering



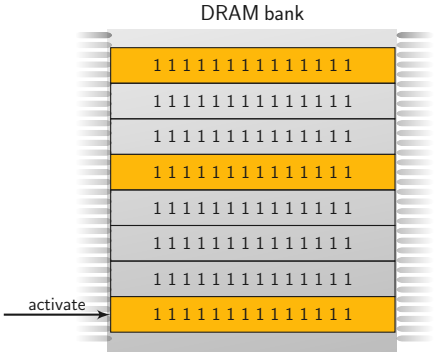
#1 - Single-sided hammering



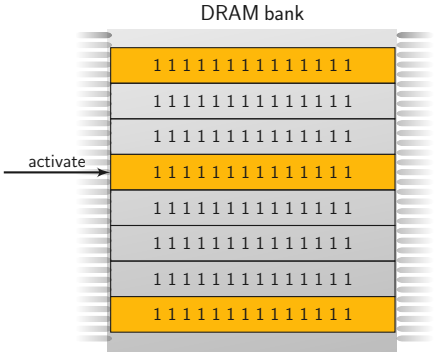
#1 - Single-sided hammering



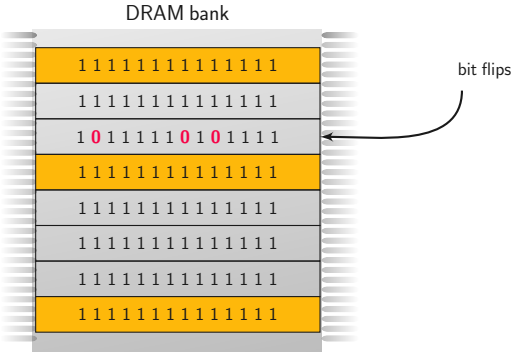
#1 - Single-sided hammering



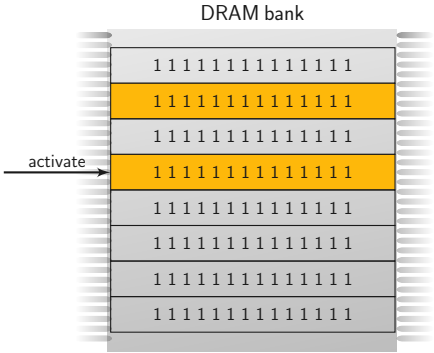
#1 - Single-sided hammering



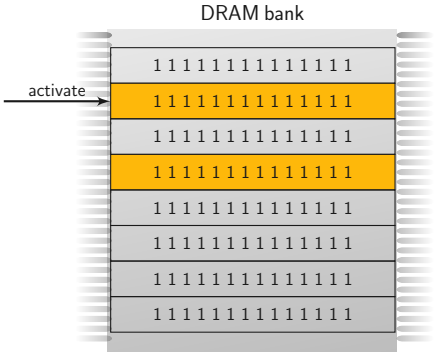
#1 - Single-sided hammering



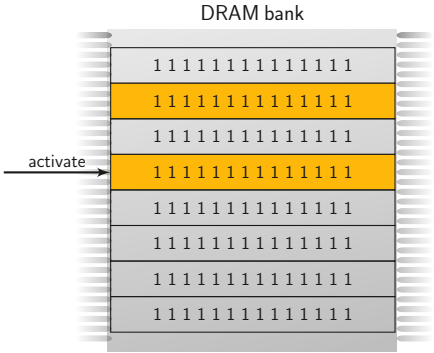
#2 - Double-sided hammering



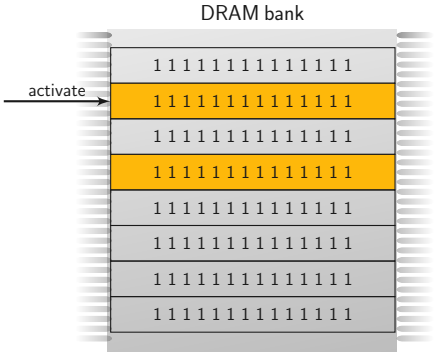
#2 - Double-sided hammering



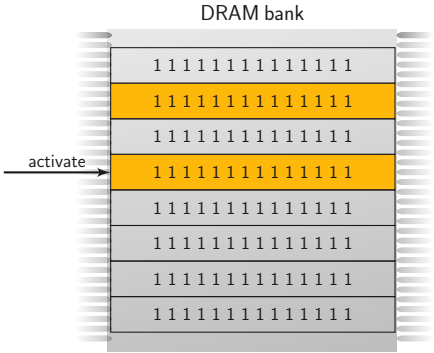
#2 - Double-sided hammering



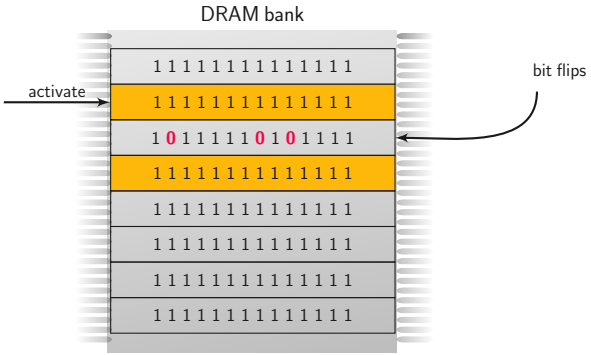
#2 - Double-sided hammering



#2 - Double-sided hammering



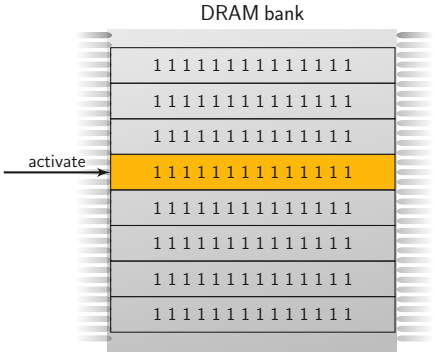
#2 - Double-sided hammering



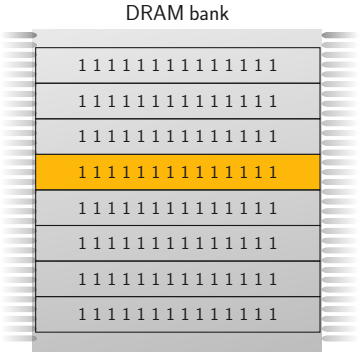
HAMMERING
TWO ROWS


HAMMERING TWO ROWS
HAMMERING A SINGLE ROW

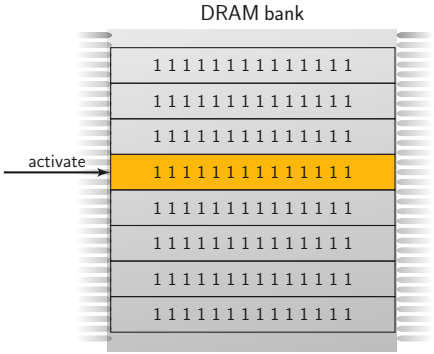
#3 - One-location hammering



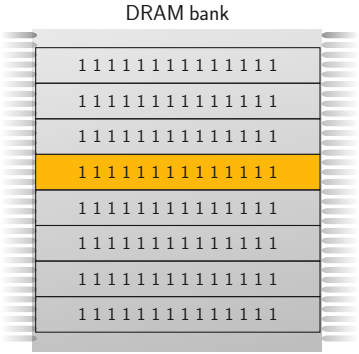
#3 - One-location hammering



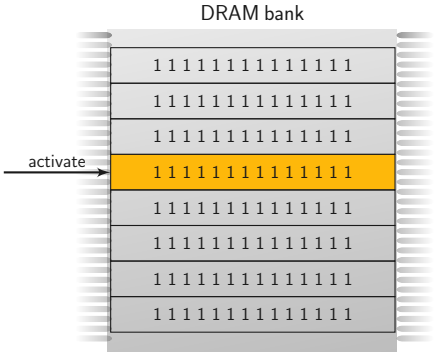
#3 - One-location hammering



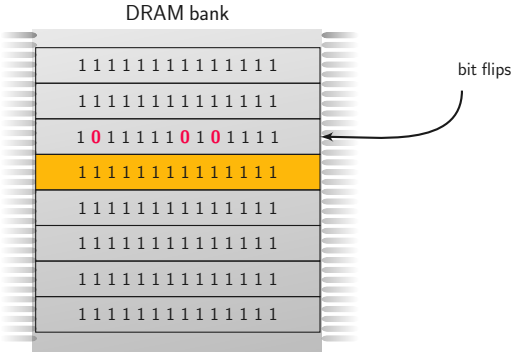
#3 - One-location hammering



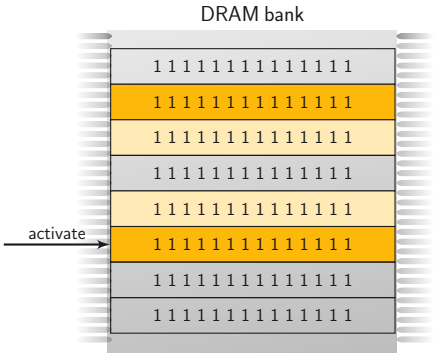
#3 - One-location hammering



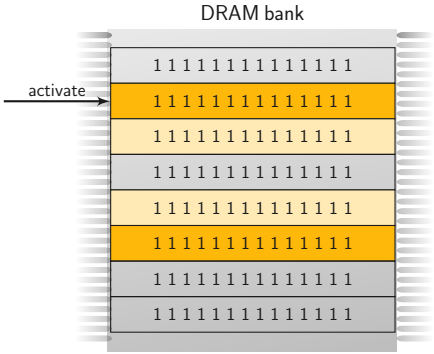
#3 - One-location hammering



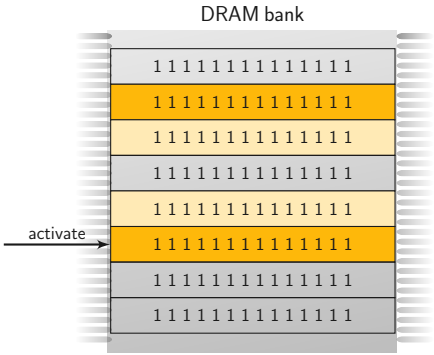
Half-Double


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

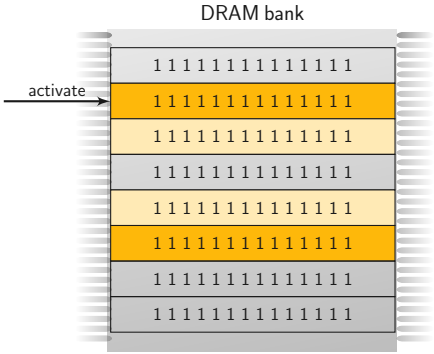
Half-Double


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

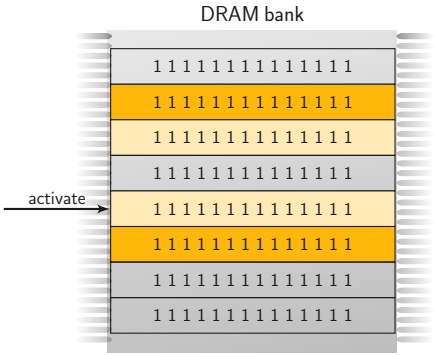
Half-Double


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

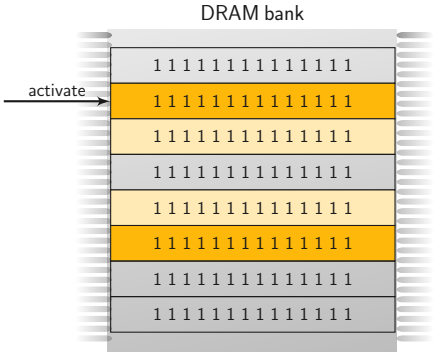
Half-Double


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

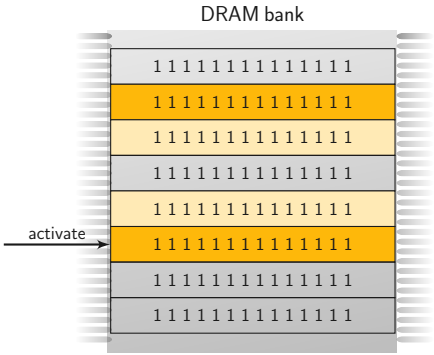
Half-Double


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

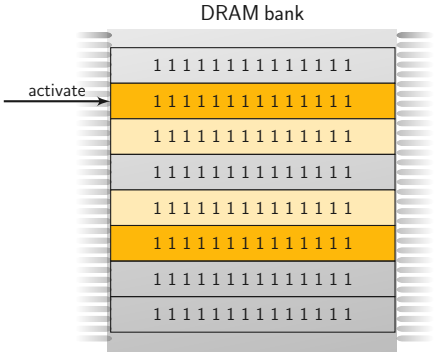
Half-Double


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

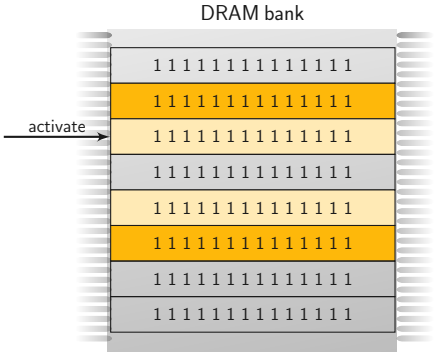
Half-Double


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

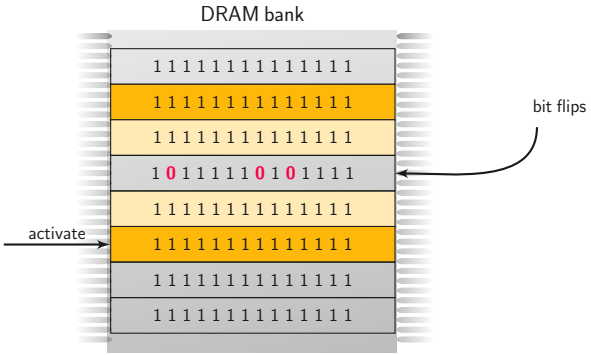
Half-Double


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

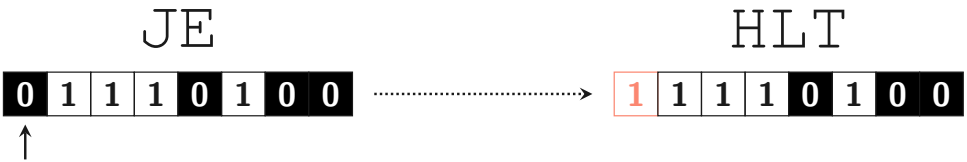
Half-Double


DRAM bank
activate
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

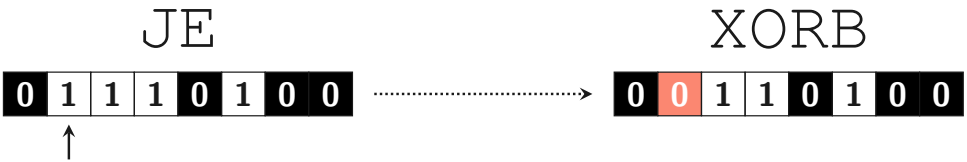
Half-Double


DRAM bank
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 1 1 1 0 1 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
bit flips
activate

Opcode Flipping - Conditional Jump



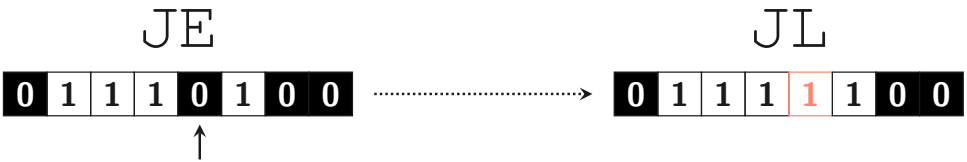
Opcode Flipping - Conditional Jump



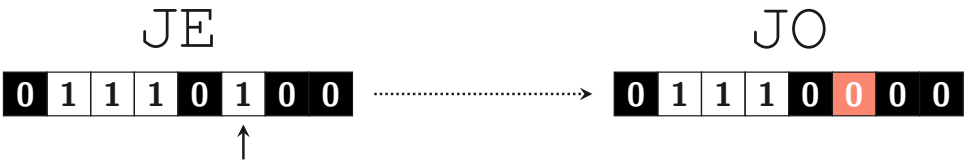
Opcode Flipping - Conditional Jump

Opcode Flipping - Conditional Jump

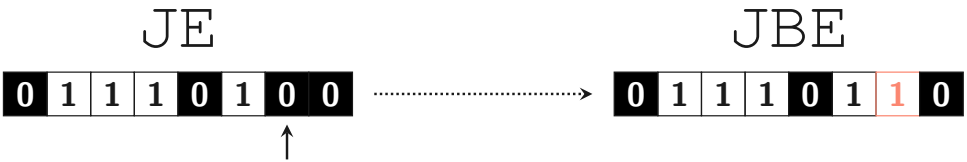
Opcode Flipping - Conditional Jump



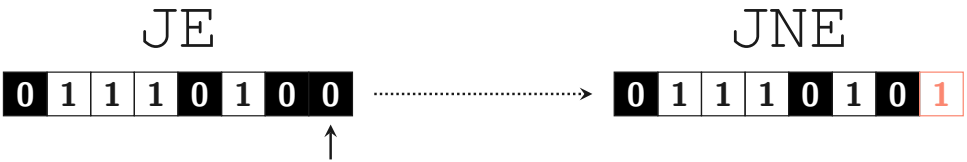
Opcode Flipping - Conditional Jump



Opcode Flipping - Conditional Jump



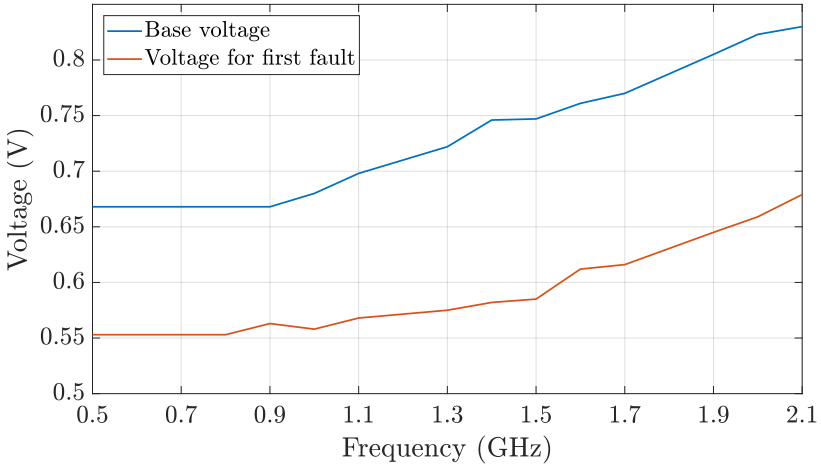
Opcode Flipping - Conditional Jump



Will it fault?


```
uint64_t multiplier = 0x1122334455667788;
uint64_t correct    = 0xdeadbeef * multiplier;
uint64_t var        = 0xdeadbeef * multiplier;

while (var == correct)
{
    var = 0xdeadbeef * multiplier;
}
uint64_t flipped_bits = var ^ correct;
```

Safe voltages



AES New Instructions - inside SGX


```
do
{
        i++;
        plaintext = <randomly generated>

        result1 = aes128_enc(plaintext);
        result2 = aes128_enc(plaintext);
} while (vec_equal_128(result1,result2) && i<iterations);
```

Plundervolt

- Should be related to undervolting

Plundervolt

- Should be related to undervolting
- From protected TEE vaults

Plundervolt

- Should be related to undervolting
- From protected TEE vaults
- Steal

Plundervolt

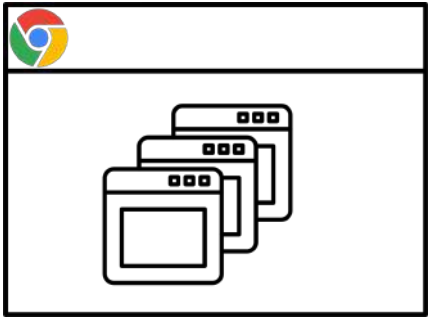
- Should be related to undervolting
- From protected TEE vaults
- Steal, corrupt

Plundervolt

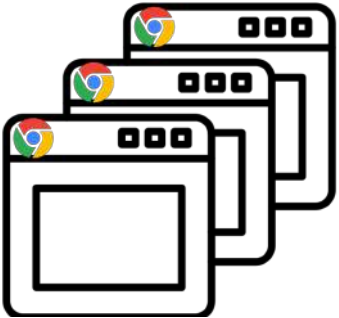
- Should be related to undervolting
- From protected TEE vaults
- Steal, corrupt, plunder, ...

PLUNDER
VOLT

Security Costs



Security Costs

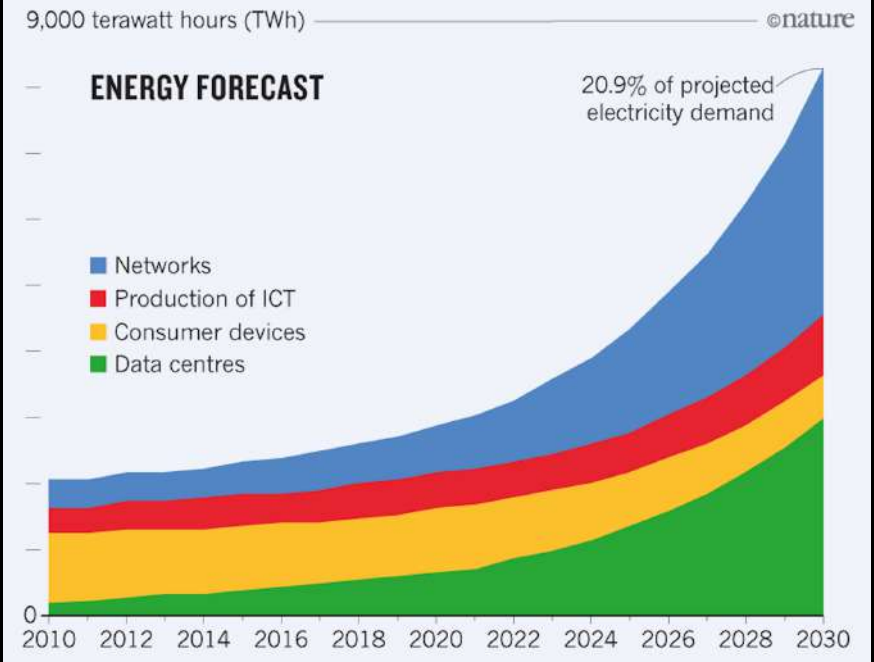


Security Costs

Security Costs

Security Costs

Security Costs


9,000 terawatt hours (TWh)
©nature
ENERGY FORECAST
20.9% of projected electricity demand
Networks
Production of ICT
Consumer devices
Data centres
0
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030

0.09%

0.40%

There are alternatives

There are alternatives to security!

How expensive is security?

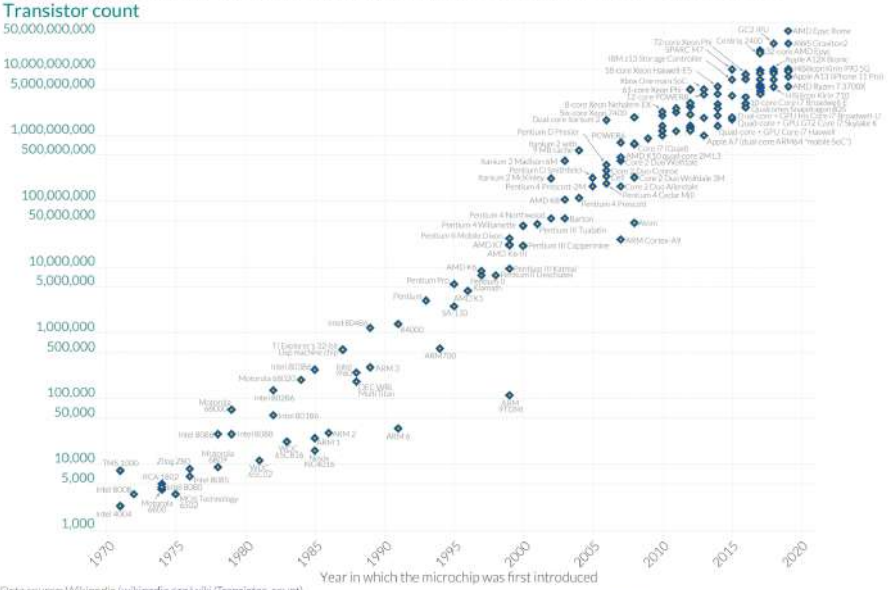

RACE FOR A VACCINE
WE WANT TO BE FIRST
WE WANT TO BE FIRST!
WE WANT TO BE FIRST!
WE WANT TO BE FIRST!
RACE TO ACT ON CLIMATE
YOU GO FIRST!
WHY SHOULD WE BE FIRST
NO, YOU GO FIRST
THEY SHOULD GO FIRST!
GUIDING


FUNCTIONALITY
SECURITY
imgflip.com

Moore's Law: The number of transistors on microchips has doubled every two years

Our World in Data

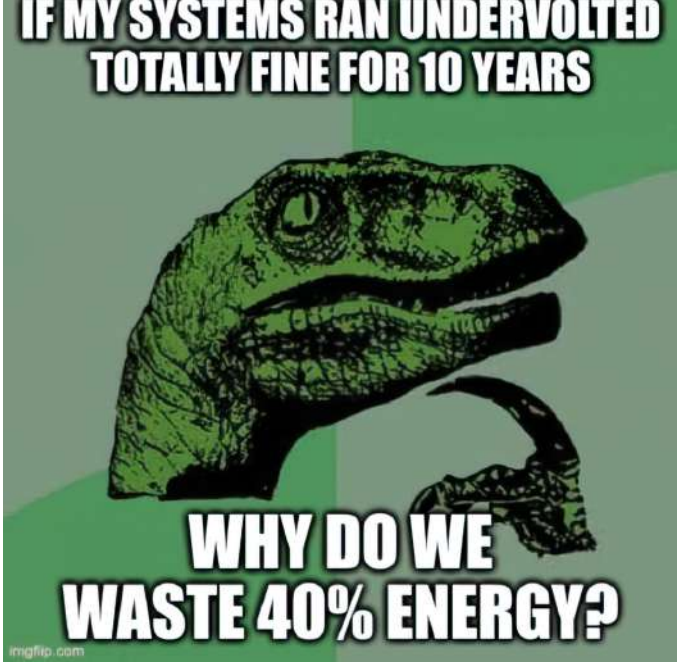
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.


THE END IS NEAR
OF MOORE'S LAW


Before
After
i7-9750H
CPU Undervolting
Huge difference!!!
Intel® Extreme Tuning Utility
Run XTU Benchmark
Cache
Other
Stress Test
Benchmarking
Profiles
App-Profile Pairing
Current Score
XTU: 1921 Marks
XTU: 2122 Marks
Compare Online
Maximum Processor Frequenc 4.15 GHz
Highest CPU Temperature 96 °C
Maximum Processor Frequenc 4.13 GHz
Highest CPU Temperature 95 °C
Package Temperature 65 °C
CPU Utilization 5 %
5 Minutes

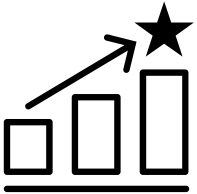

IF MY SYSTEMS RAN UNDERVOLTED
TOTALLY FINE FOR 10 YEARS
WHY DO WE
WASTE 40% ENERGY?
imgflip.com

PLUNDER VOLT

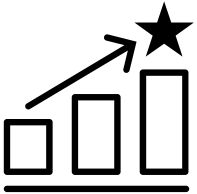
Why are problems like Rowhammer not solved already?

Common misunderstandings...

Common misunderstandings...

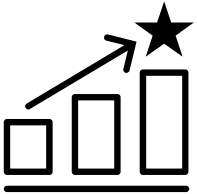


Common misunderstandings...



... create bad incentives.

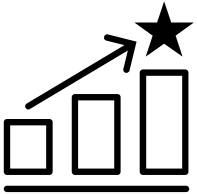
Common misunderstandings...



... create bad incentives.

- A "bit" more reliability

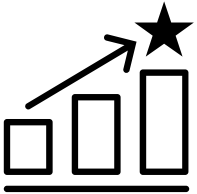
Common misunderstandings...



... create bad incentives.

- A "bit" more reliability
- Why not higher or dynamic refresh rates everywhere (e.g. TRR, PARA, ...)?

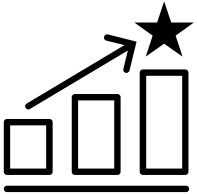
Common misunderstandings...



... create bad incentives.

- A "bit" more reliability
- Why not higher or dynamic refresh rates everywhere (e.g. TRR, PARA, ...)?
  - "just a few more targeted refreshes"

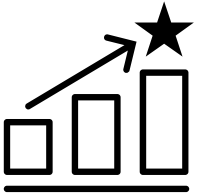
Common misunderstandings...



... create bad incentives.

- A "bit" more reliability
- Why not higher or dynamic refresh rates everywhere (e.g. TRR, PARA, ...)?
  - "just a few more targeted refreshes"
- Why not ECC everywhere?

Common misunderstandings...



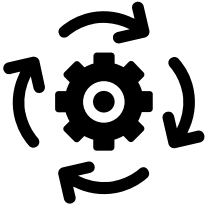
... create bad incentives.

- A "bit" more reliability
- Why not higher or dynamic refresh rates everywhere (e.g. TRR, PARA, ...)?
  - "just a few more targeted refreshes"
- Why not ECC everywhere?

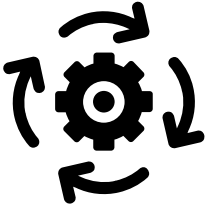
→ What incentives does it create?

Optimizing systems...

Optimizing systems...

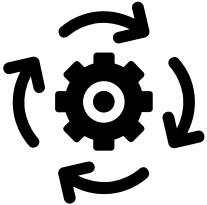


Optimizing systems...



Fundamental problem: we assume what is still reliable

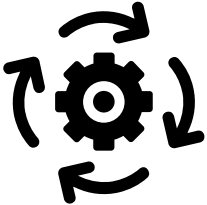
Optimizing systems...



Fundamental problem: we assume what is still reliable

- Refreshing x times per second is fine

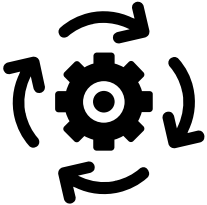
Optimizing systems...



Fundamental problem: we assume what is still reliable

- Refreshing x times per second is fine
- Normal usage, no adversary

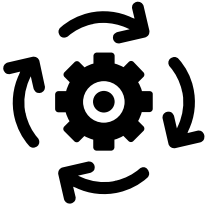
Optimizing systems...



Fundamental problem: we assume what is still reliable

- Refreshing x times per second is fine
- Normal usage, no adversary
- Assume there won't be more than n bit errors

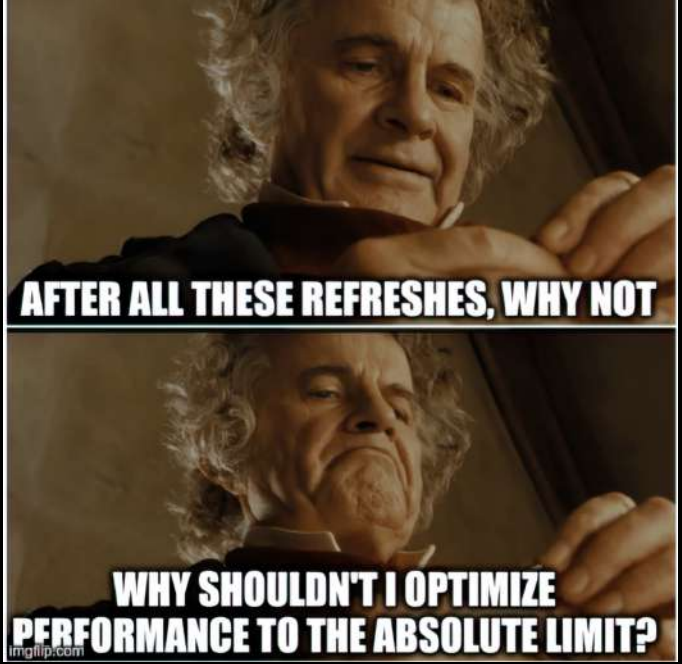
Optimizing systems...



Fundamental problem: we assume what is still reliable

- Refreshing x times per second is fine
- Normal usage, no adversary
- Assume there won't be more than n bit errors

$\rightarrow$ How far can we go with x while staying below n bit errors?


AFTER ALL THESE REFRESHES, WHY NOT
WHY SHOULDN'T I OPTIMIZE
PERFORMANCE TO THE ABSOLUTE LIMIT?
imgflip.com

Real-world example

Mobile vendors since 2018: let's add ECC by default

Real-world example

Mobile vendors since 2018: let's add ECC by default

- ECC memory $\rightarrow$ fewer bit flips $+$ more security

Real-world example

Mobile vendors since 2018: let's add ECC by default

- ECC memory $\rightarrow$ fewer bit flips + more security

Also vendors:

Real-world example

Mobile vendors since 2018: let's add ECC by default

- ECC memory $\rightarrow$ fewer bit flips + more security

Also vendors:

- Let's squeeze out the last bit of efficiency for battery runtime until just before bit flips occur

It's an optimization problem

It's an optimization problem

It's an optimization problem

It's an optimization problem

- You never know how far is still safe

It's an optimization problem

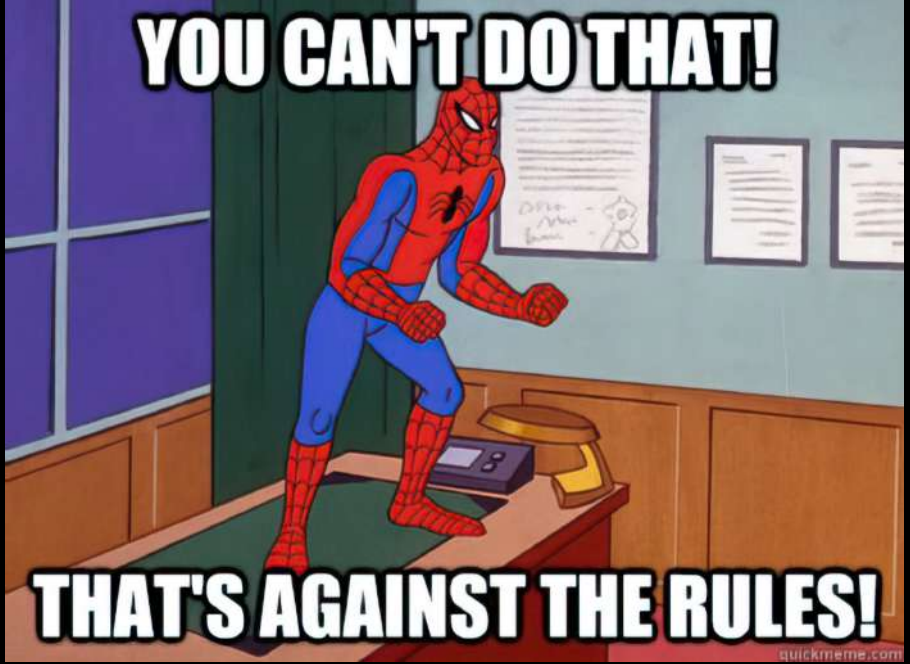
- You never know how far is still safe
- "safe" / "reliable" changes over time

It's an optimization problem

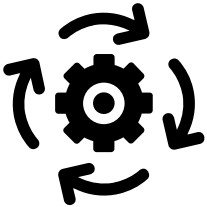
- You never know how far is still safe
- "safe" / "reliable" changes over time
- Adversary is intelligent and improves attacks over time

Security vs Reliability

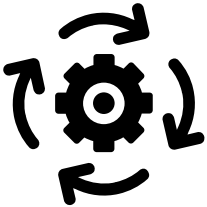
Security ~~vs Reliability~~


YOU CAN'T DO THAT!
THAT'S AGAINST THE RULES!
quickmeme.com

Security for Efficiency?



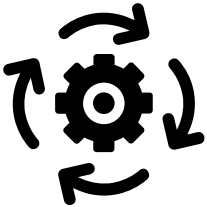
Make bit flips degrade performance **without** impacting security



Make bit flips degrade performance **without** impacting security

- Cryptographic MAC

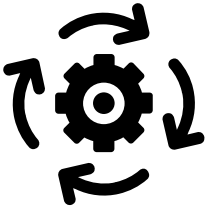
Principled Cryptographic Security and Integrity



Make bit flips degrade performance **without** impacting security

- Cryptographic MAC
- Detect **any** number of bit flips

Principled Cryptographic Security and Integrity



Make bit flips degrade performance **without** impacting security

- Cryptographic MAC
- Detect **any** number of bit flips
- Correction by **brute-force** search for correct data

CSI:Rowhammer Correction Duration

# Errors	# MAC Comp.	Avg Duration
1	17	11 ns
2	771	3.68 µs
3	33 800	124 µs
4	1.51×10^6	6.65 ms
5	6.91×10^7	261 ms
6	3.07×10^9	12.8 s
7	1.21×10^{11}	9.11 min
8	5.72×10^{12}	6.11 h

CSI:Rowhammer Security Guarantees

CSI:Rowhammer Security Guarantees

CSI:Rowhammer Security Guarantees

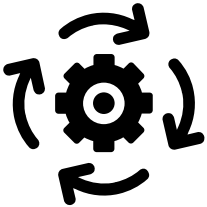
- Silent data corruption less than once per 10^9 billion years

CSI:Rowhammer Security Guarantees

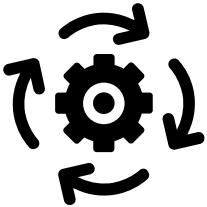
- Silent data corruption less than once per 10^9 billion years
- Second preimage after hammering for one year: $9.75 \cdot 10^{-5}$ %

CSI:Rowhammer Security Guarantees

- Silent data corruption less than once per 10^9 billion years
- Second preimage after hammering for one year: $9.75 \cdot 10^{-5}\,\%$
- Erroneous correction of 8-bit errors: $0.0161\,\%$



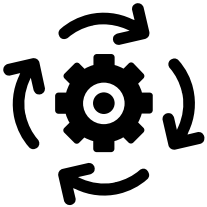
Make bit flips degrade performance **without** impacting security



Make bit flips degrade performance **without** impacting security

- Cryptographic MAC

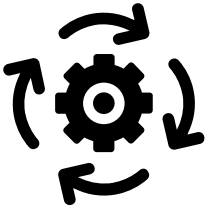
Principled Cryptographic Security and Integrity



Make bit flips degrade performance **without** impacting security

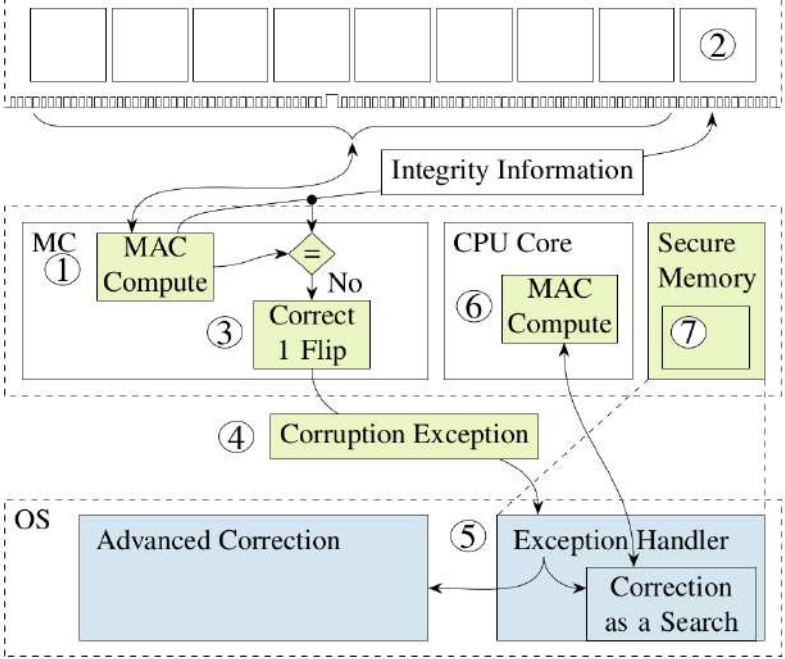
- Cryptographic MAC
- Detect **any** number of bit flips

Principled Cryptographic Security and Integrity



Make bit flips degrade performance **without** impacting security

- Cryptographic MAC
- Detect **any** number of bit flips
- Correction by **brute-force** search for correct data


2
Integrity Information
MC
1
MAC
Compute
=
No
3
Correct
1 Flip
CPU Core
6
MAC
Compute
Secure
Memory
7
4
Corruption Exception
OS
Advanced Correction
5
Exception Handler
Correction
as a Search

CSI:Rowhammer Correction Duration

# Errors	# MAC Comp.	Avg Duration
1	17	11 ns
2	771	3.68 µs
3	33 800	124 µs
4	1.51×10^6	6.65 ms
5	6.91×10^7	261 ms
6	3.07×10^9	12.8 s
7	1.21×10^{11}	9.11 min
8	5.72×10^{12}	6.11 h

CSI:Rowhammer Security Guarantees

CSI:Rowhammer Security Guarantees

CSI:Rowhammer Security Guarantees

- Silent data corruption less than once per 10^9 billion years

CSI:Rowhammer Security Guarantees

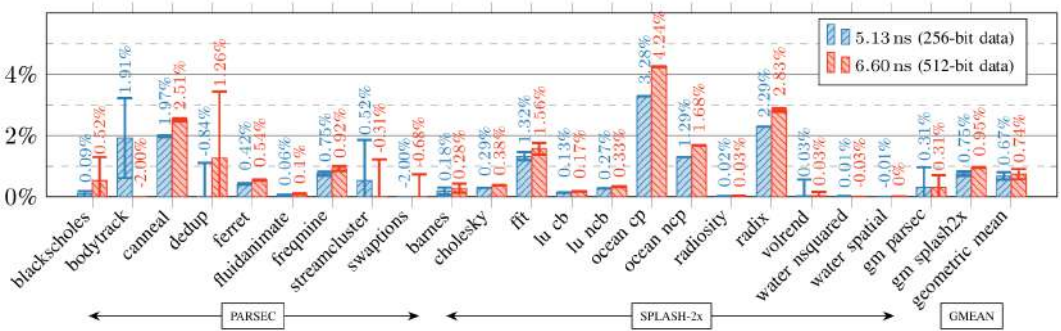
- Silent data corruption less than once per 10^9 billion years
- Second preimage after hammering for one year: $9.75 \cdot 10^{-5}$ %

CSI:Rowhammer Security Guarantees

- Silent data corruption less than once per 10^9 billion years
- Second preimage after hammering for one year: $9.75 \cdot 10^{-5}\,\%$
- Erroneous correction of 8-bit errors: $0.0161\,\%$

CSI:Rowhammer Performance

On average less than 0.75 % overhead



Overclocking

Undervolting

Intel® Extreme Tuning Utility

Monitoring | Settings | Help

System Information
Manual Tuning
• All Controls
Core
Graphics
Stress Test
Profiles

Core

Reference Clock 103,2258 MHz

Max Non Turbo Boost Ratio 34 x

Turbo Boost Short Power Max Enable: Disable / Enable

Turbo Boost Short Power Max 1,200,000 W

Turbo Boost Power Max 1,050,000 W

Turbo Boost Power Time Window 0,000976656 Seconds

Core Current Limit 300,000 A

Additional Turbo Voltage 0,00000 mV

Multipliers

1 Active Core 42 x
2 Active Cores 42 x
3 Active Cores 42 x
4 Active Cores 42 x

4 Active Cores
Default 38 x
Active 38 x
Proposed 42 x
Limits the maximum ratio that the processor can use while four cores are active.

Graphics

Processor Graphics Current Limit 300,000 A

Core	Default	Proposed
Reference Clock	101,0526 MHz	101,2258 MHz
Max Non Turbo Boost Ratio	34 x	34 x
Max Non-Turbo Boost CPU Sp...	3,436 GHz	3,510 GHz
Max Turbo Boost CPU Speed	4,042 GHz	4,335 GHz
1 Active Core	40 x	42 x
2 Active Cores	40 x	42 x
3 Active Cores	39 x	42 x
4 Active Cores	38 x	42 x
Turbo Boost Power Max	1000,000 W	1050,000 W
Turbo Boost Short Power Max	1200,000 W	1200,000 W
Turbo Boost Short Power Max...	Enable	Enable
Turbo Boost Power Time Wind...	0,000976656 S...	0,000976656 S...
Core Current Limit	300,000 A	300,000 A
Additional Turbo Voltage	0,00000 mV	0,00000 mV
Graphics	**Default**	**Proposed**
Processor Graphics Current Li...	300,000 A	300,000 A

Apply | Discard | Save to Profile

Force Reboot

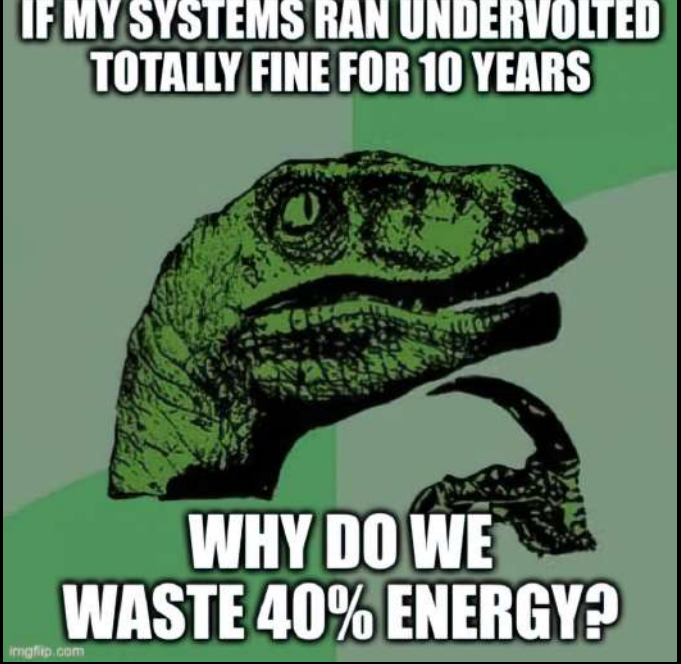
CPU Core Temperature 37 °C
CPU Utilization 3 %
Processor Frequency 3,54 GHz
Memory Utilization 2708 MB
CPU Total TDP 15 W

5 Minutes

CPU Utilization 3 %	Memory Utilization 2708 MB	CPU Core Temperature 36 °C	CPU Throttling 0%	Processor Frequency 3,54 GHz
Graphics Frequency 854 MHz	Active Core Count 1	CPU Total TDP 16 W	IACore TDP 10 W	Graphics TDP 0 W
Reference Clock Frequency 101,0 MHz	CPU Core Temperature 1 36 °C	CPU Core Temperature 2 36 °C	CPU Core Temperature 3 36 °C	CPU Core Temperature 4 35 °C
Memory Frequency 1617 MHz				

Computer Base

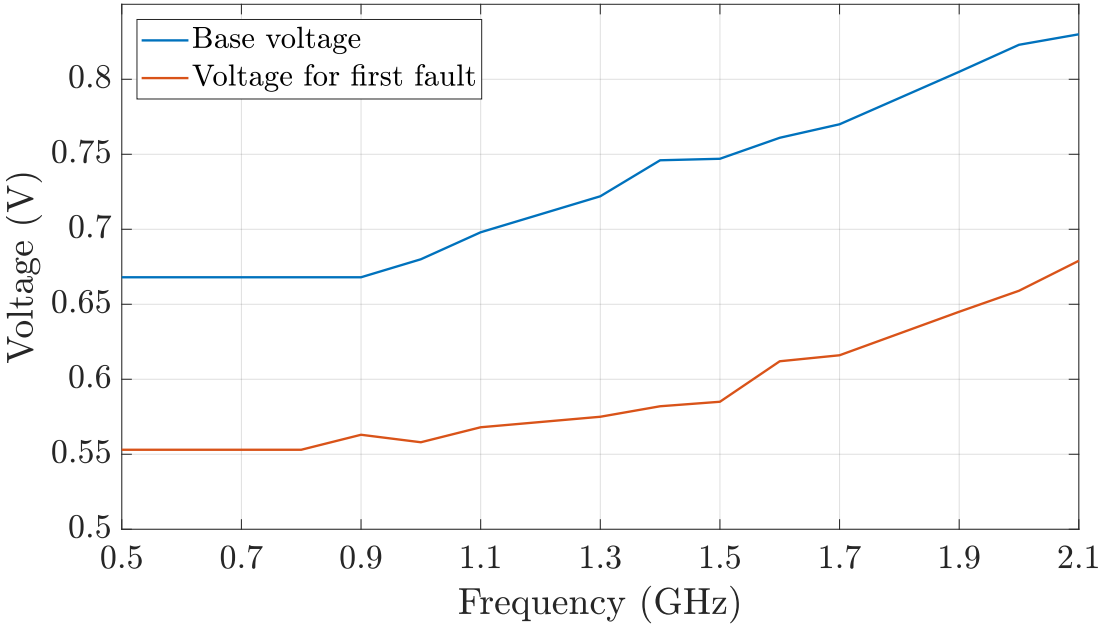

Before
After
i7-9750H
CPU Undervolting
Huge difference!!!
Intel® Extreme Tuning Utility
Run XTU Benchmark
Cache
Other
Stress Test
Benchmarking
Profiles
App-Profile Pairing
Current Score
XTU: 1921 Marks
XTU: 2122 Marks
Compare Online
Maximum Processor Frequenc 4.15 GHz
Highest CPU Temperature 96 °C
Maximum Processor Frequenc 4.13 GHz
Highest CPU Temperature 95 °C
Package Temperature 65 °C
CPU Utilization 5 %
5 Minutes


IF MY SYSTEMS RAN UNDERVOLTED
TOTALLY FINE FOR 10 YEARS
WHY DO WE
WASTE 40% ENERGY?
imgflip.com

PLUNDER VOLT

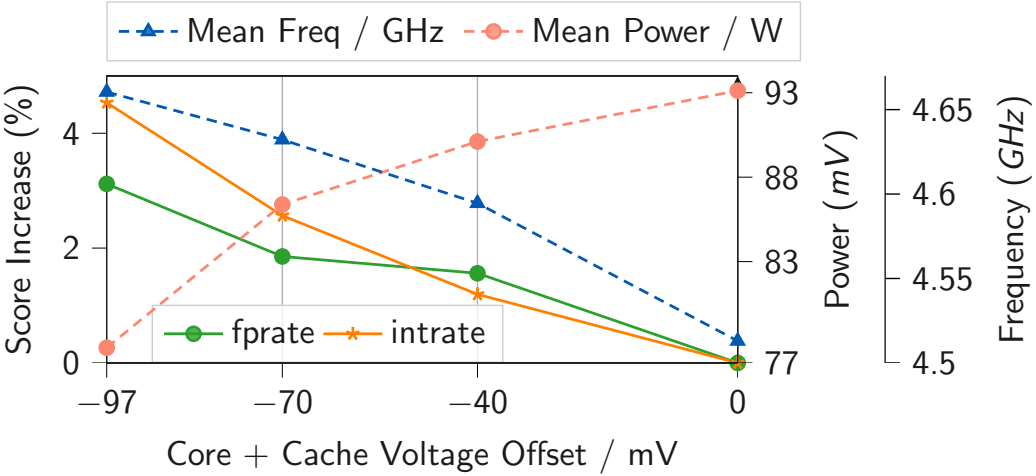
Will it fault?


```
uint64_t multiplier = 0x1122334455667788;
uint64_t correct    = 0xdeadbeef * multiplier;
uint64_t var        = 0xdeadbeef * multiplier;

while (var == correct)
{
    var = 0xdeadbeef * multiplier;
}
uint64_t flipped_bits = var ^ correct;
```


Base voltage
Voltage for first fault
Voltage (V)
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.5
0.7
0.9
1.1
1.3
1.5
1.7
1.9
2.1
Frequency (GHz)

Can we make this secure?

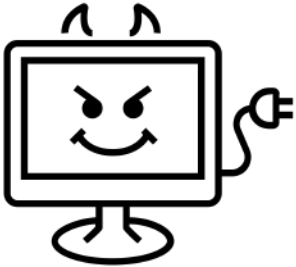
Performance Improvement and Power Savings (as a graph)



Performance Improvement and Power Savings

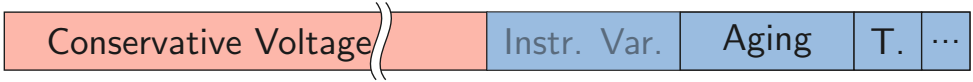
CPU	V_{off}	Score	Power	Freq.	Energy Eff.
i5-1035G1	−70 mV	+6.0 %	−0.1 %	+8.5 %	+6.1 %
	−97 mV	+7.9 %	−0.5 %	+12 %	+8.4 %
i9-9900K	−70 mV	+2.2 %	−7.2 %	+2.6 %	+10 %
	−97 mV	+3.8 %	−16 %	+3.3 %	+23 %
7700X*	−70 mV	+1.4 %	−9.8 %	+1.8 %	+12 %
	−97 mV	+1.9 %	−15 %	+1.8 %	+20 %

Problem: Reliability Issues



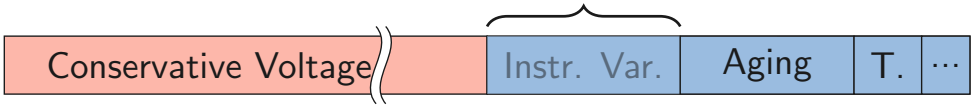
Problem: **Security** Issues

CPU Supply Voltage



CPU Supply Voltage

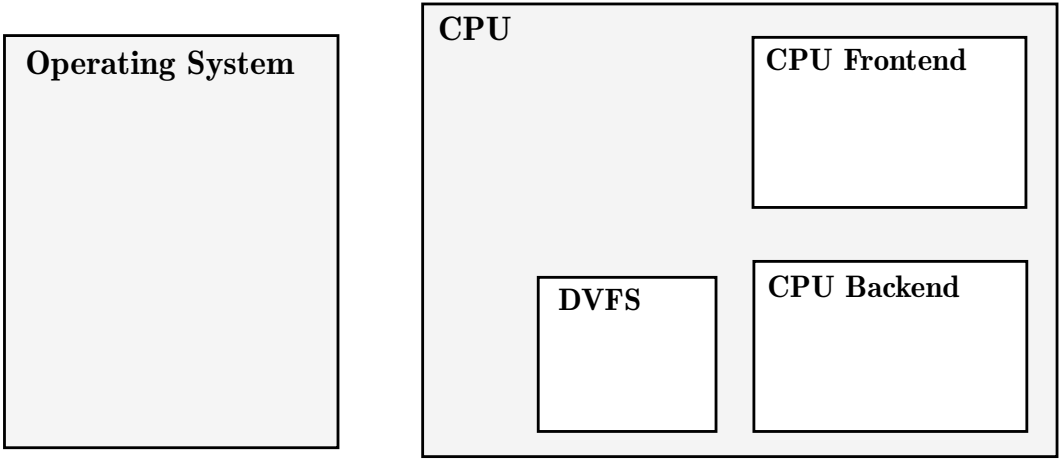
Up to a 150 mV variation in instruction voltage requirement.



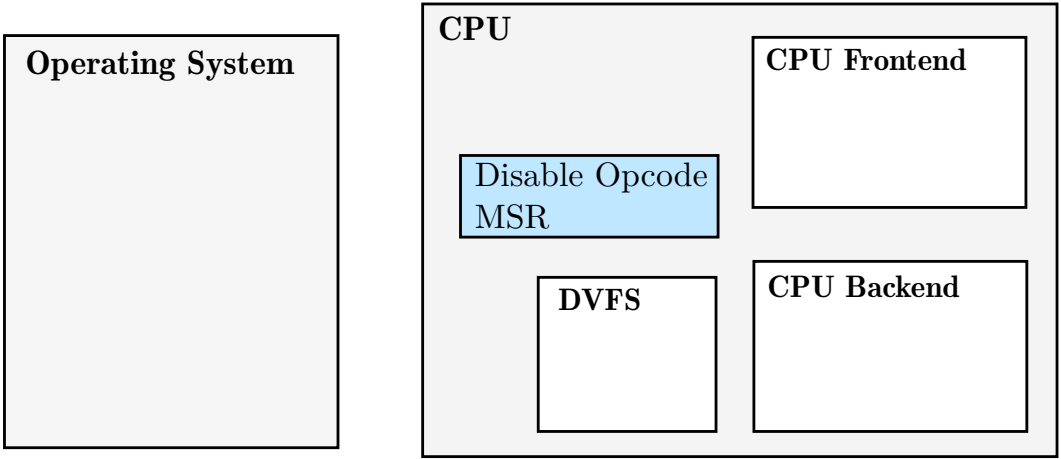
Some Instruction Produce Faulty Results

Instruction	IMUL	VOR*	AESENC	VXOR*	VANDN*	VAND*	VSQRTPD	VPCLMULQDQ	VPSRAD	VPCMP*	VPMAX*	VPADDQ
Number of Faults	**79**	47	40	40	30	28	24	16	9	5	3	1

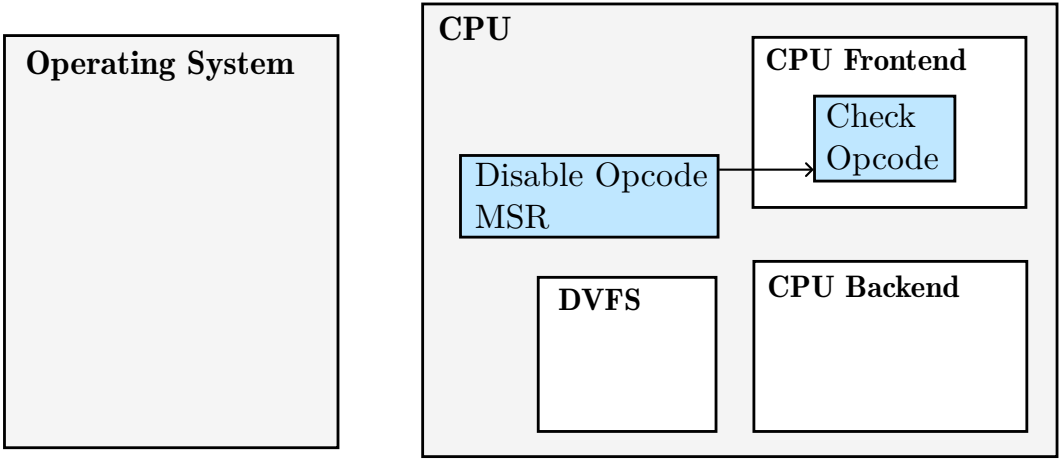
Overview



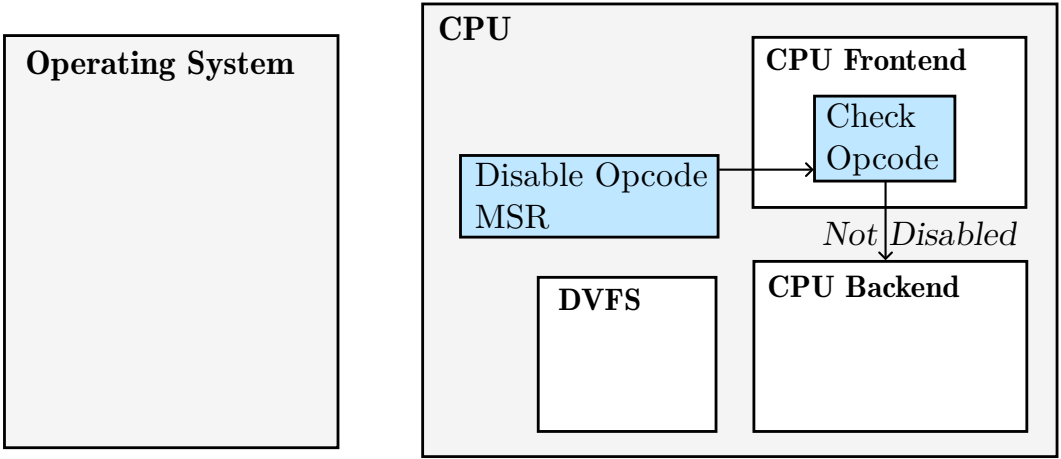
Overview


Operating System
CPU
CPU Frontend
Disable Opcode MSR
DVFS
CPU Backend

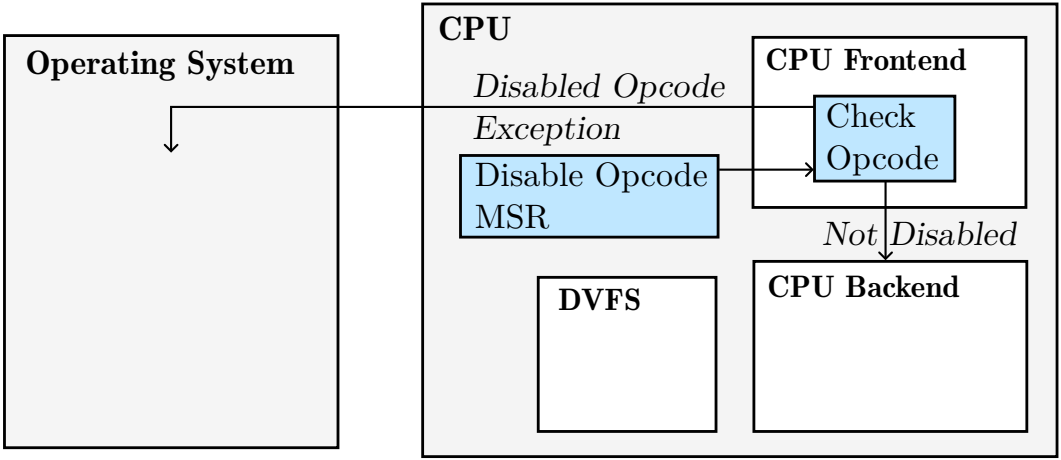
Overview


Operating System
CPU
CPU Frontend
Check Opcode
Disable Opcode MSR
DVFS
CPU Backend

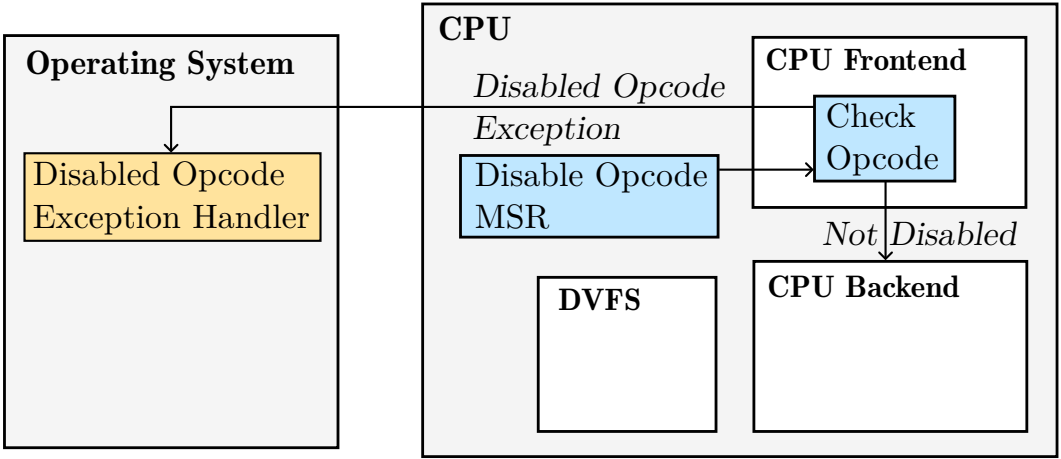
Overview


Operating System
CPU
CPU Frontend
Check Opcode
Disable Opcode MSR
Not Disabled
DVFS
CPU Backend

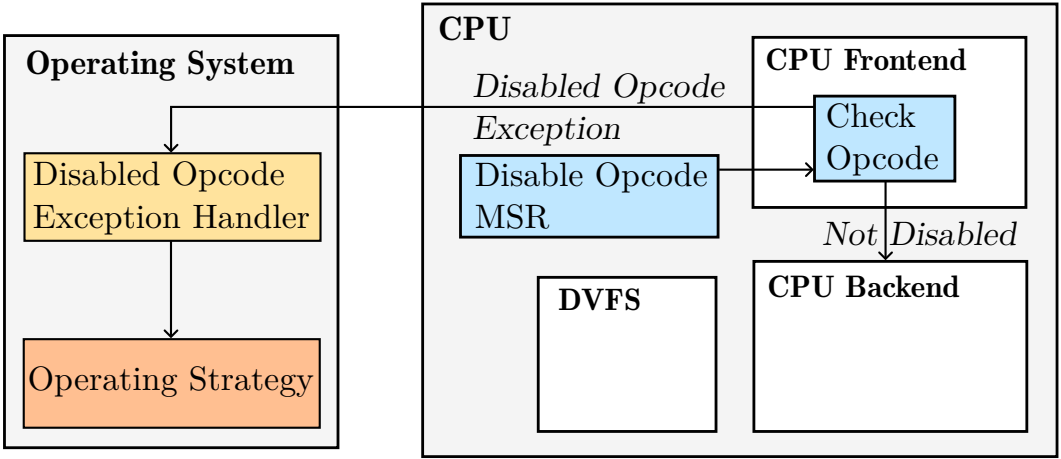
Overview


Operating System
CPU
CPU Frontend
Disabled Opcode Exception
Check Opcode
Disable Opcode MSR
Not Disabled
DVFS
CPU Backend

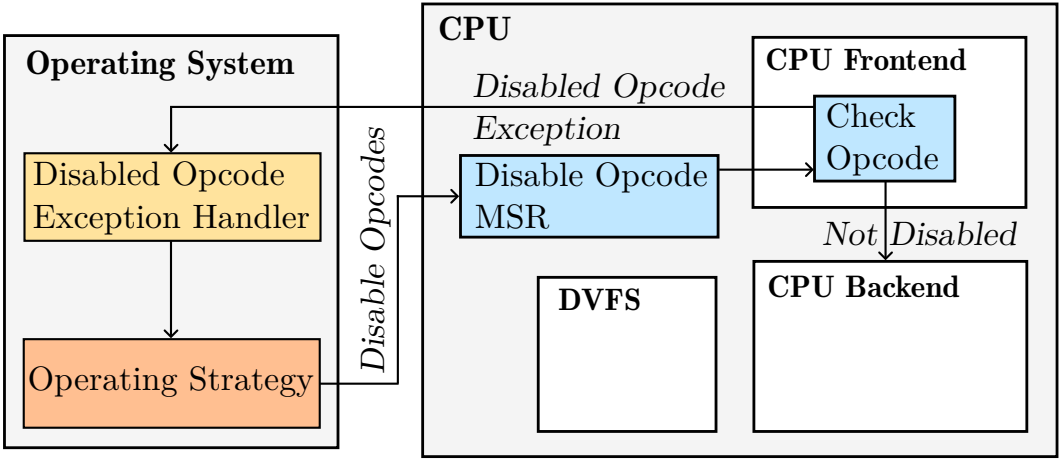
Overview


Operating System
Disabled Opcode Exception Handler
CPU
Disabled Opcode Exception
Disable Opcode MSR
CPU Frontend
Check Opcode
Not Disabled
DVFS
CPU Backend

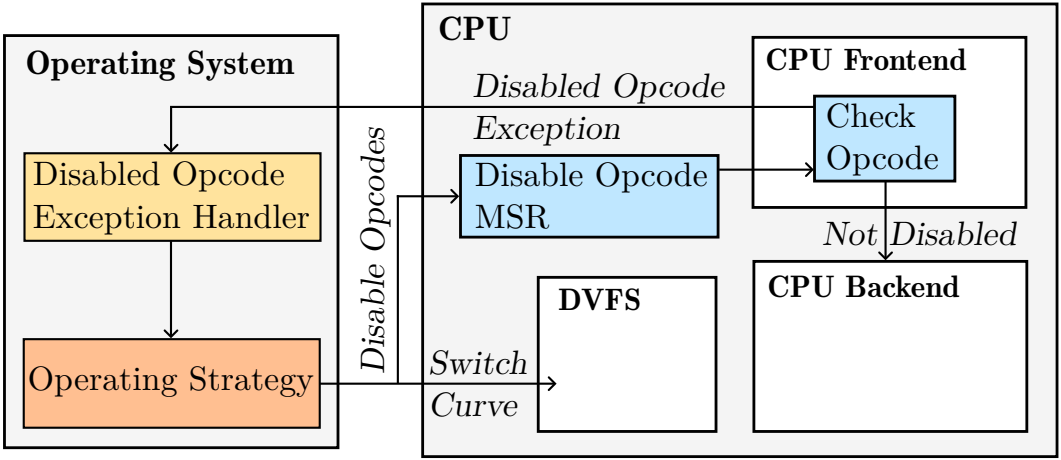
Overview


Operating System
Disabled Opcode Exception Handler
Operating Strategy
CPU
CPU Frontend
Disabled Opcode Exception
Check Opcode
Disable Opcode MSR
Not Disabled
DVFS
CPU Backend

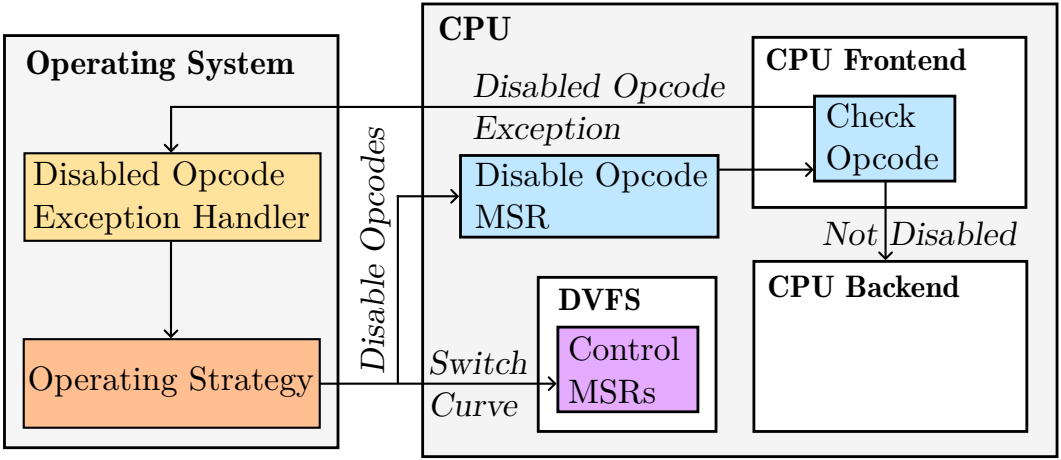
Overview


Operating System
Disabled Opcode Exception Handler
Operating Strategy
Disable Opcodes
CPU
Disabled Opcode Exception
Disable Opcode MSR
CPU Frontend
Check Opcode
Not Disabled
DVFS
CPU Backend

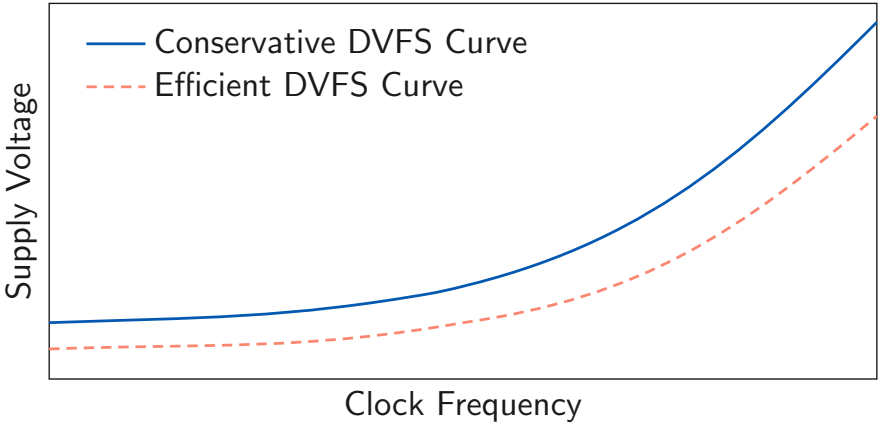
Overview


Operating System
Disabled Opcode Exception Handler
Operating Strategy
Disable Opcodes
CPU
Disabled Opcode Exception
Disable Opcode MSR
Switch Curve
DVFS
CPU Frontend
Check Opcode
Not Disabled
CPU Backend

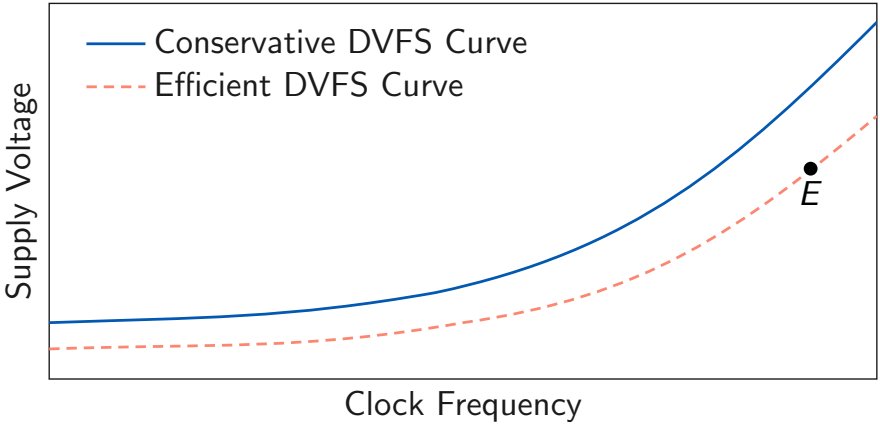
Overview


Operating System
Disabled Opcode Exception Handler
Operating Strategy
Disable Opcodes
CPU
Disabled Opcode Exception
Disable Opcode MSR
CPU Frontend
Check Opcode
Not Disabled
CPU Backend
DVFS
Control MSRs
Switch Curve

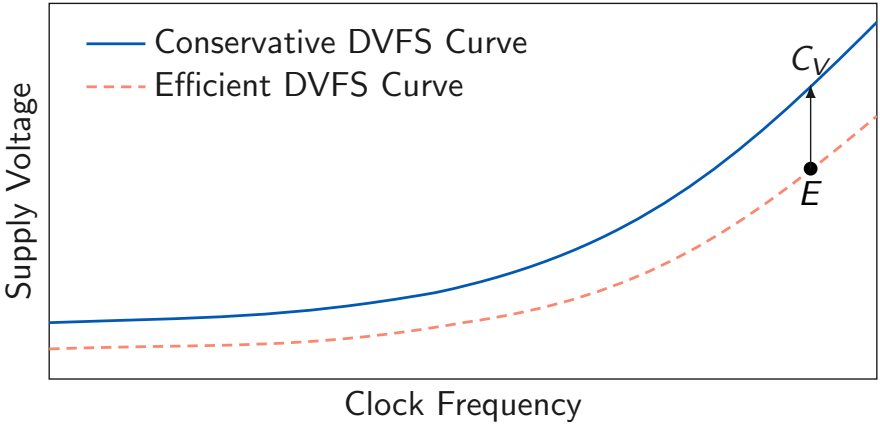
DVFS Curves and Operating Strategies



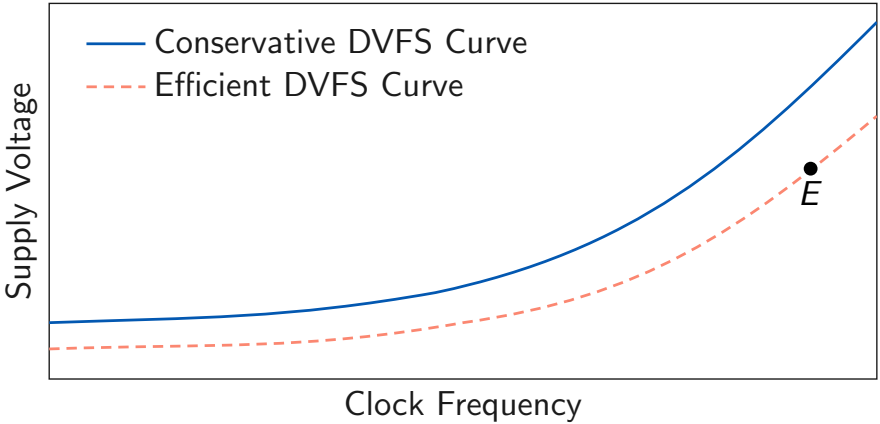
DVFS Curves and Operating Strategies



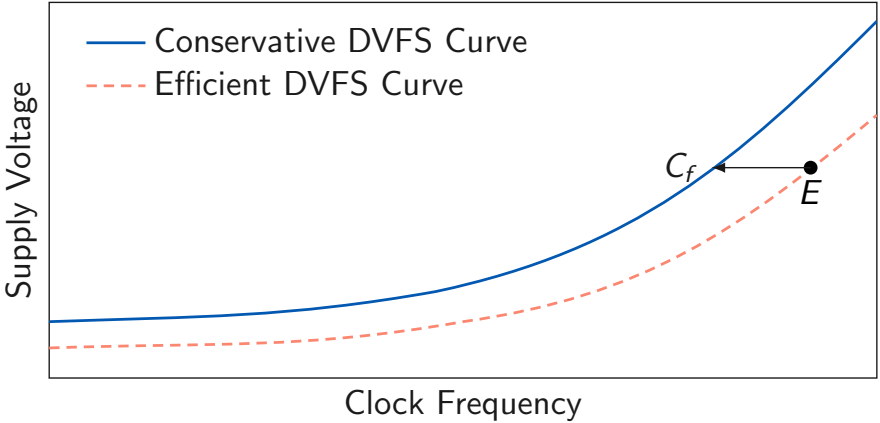
DVFS Curves and Operating Strategies



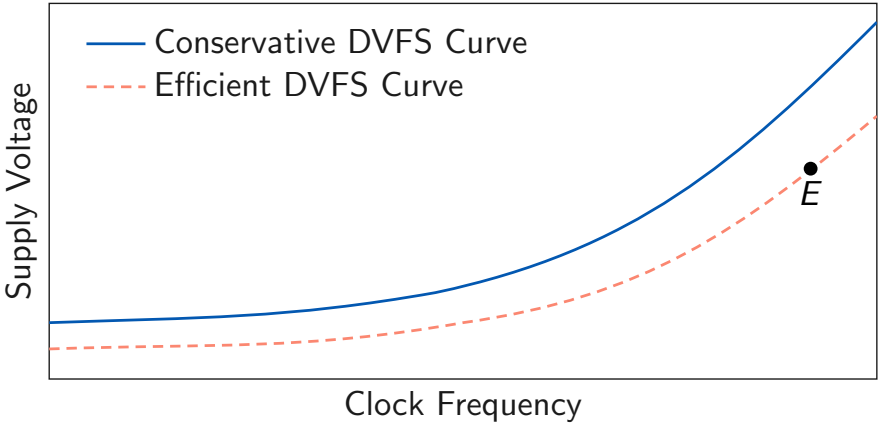
DVFS Curves and Operating Strategies



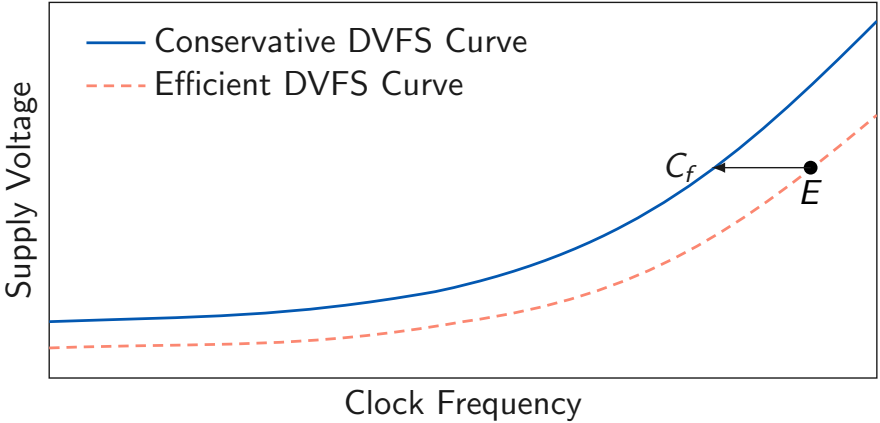
DVFS Curves and Operating Strategies



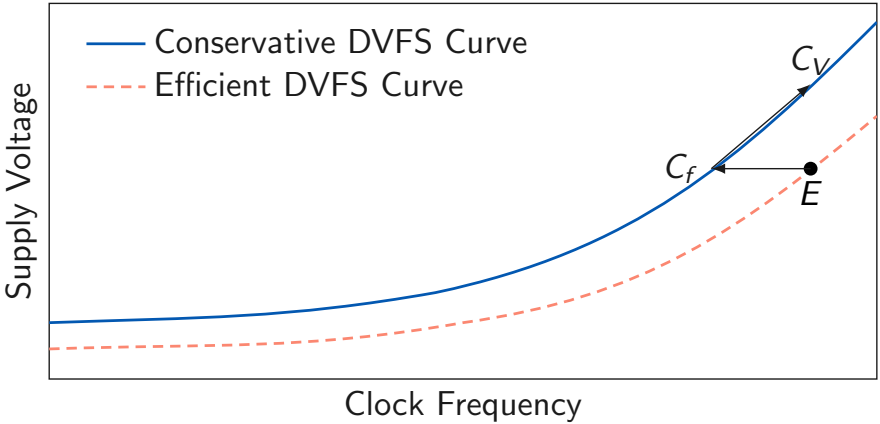
DVFS Curves and Operating Strategies



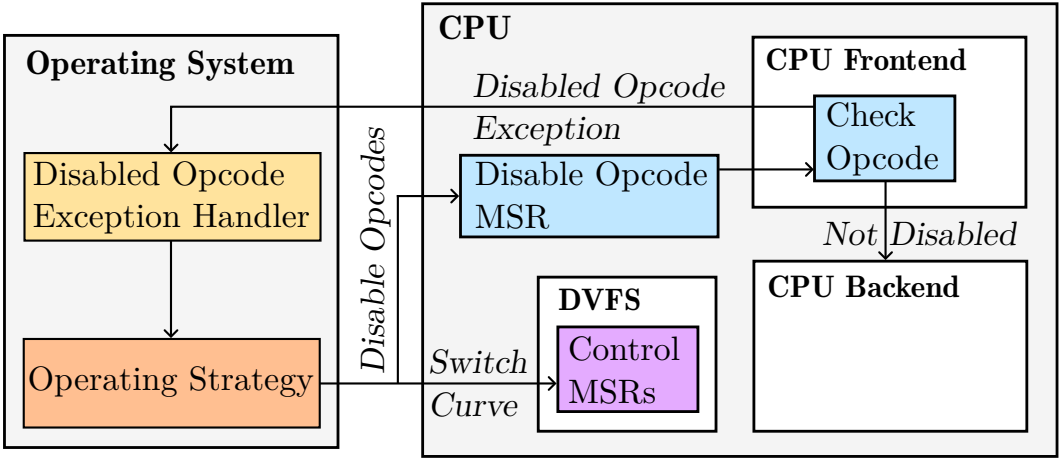
DVFS Curves and Operating Strategies



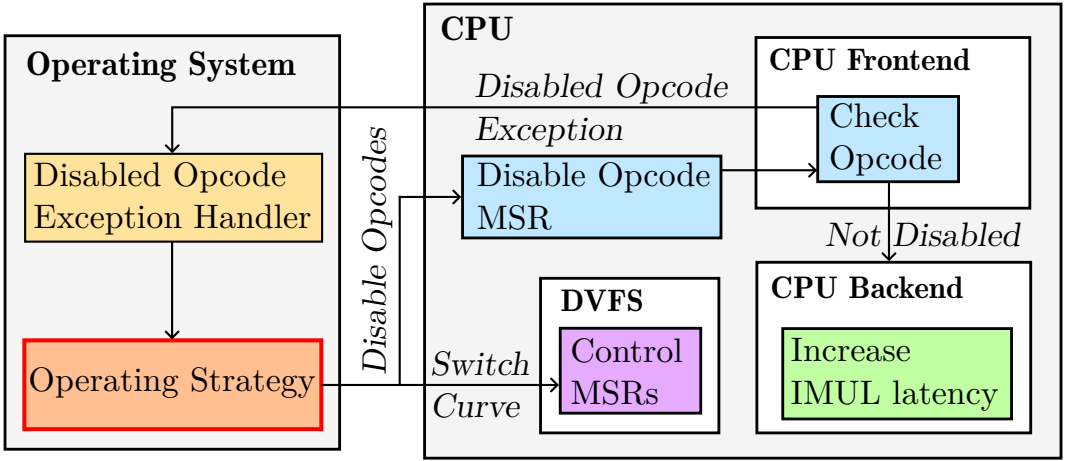
DVFS Curves and Operating Strategies



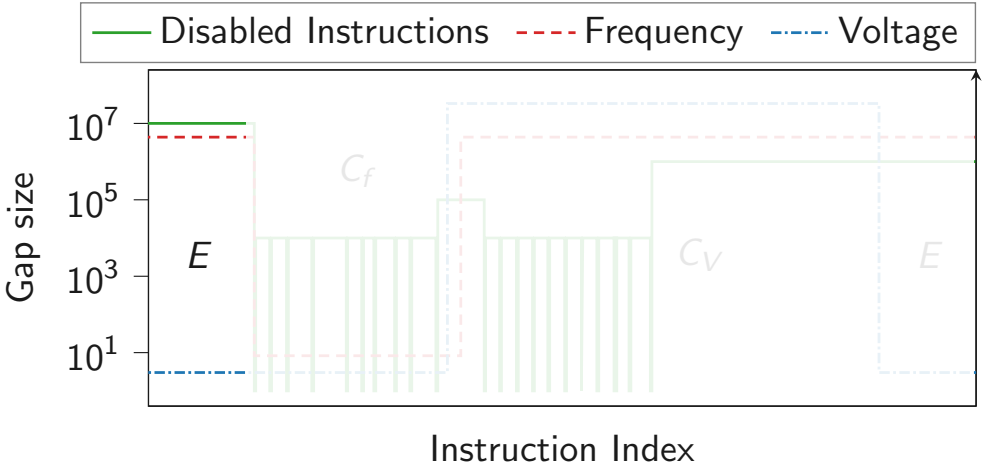
Operating Strategy



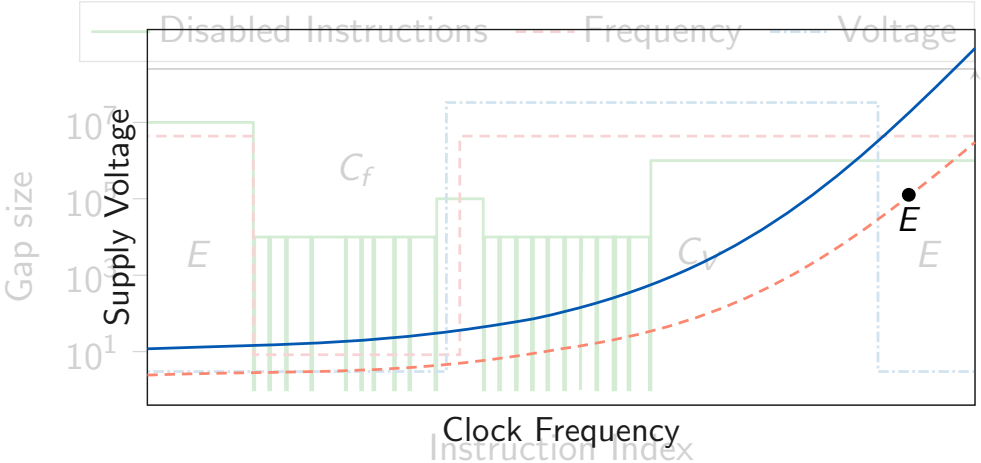
Operating Strategy



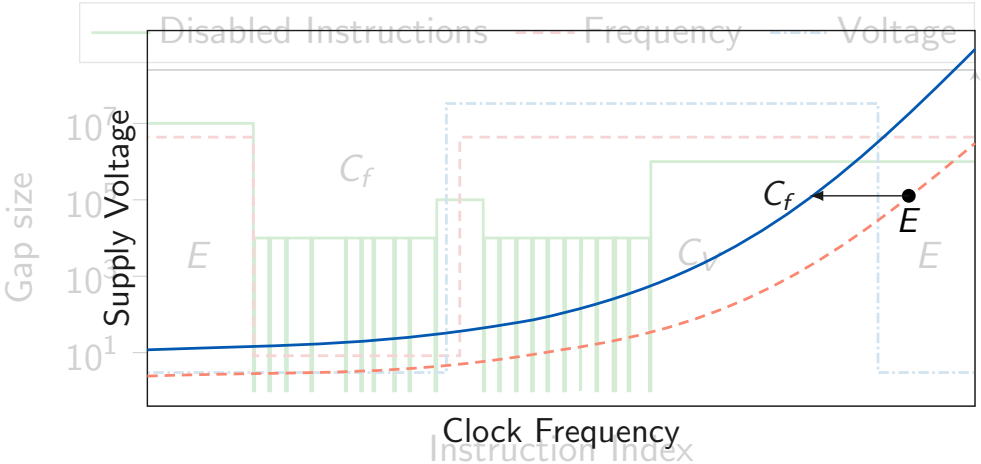
fV Operating Strategy



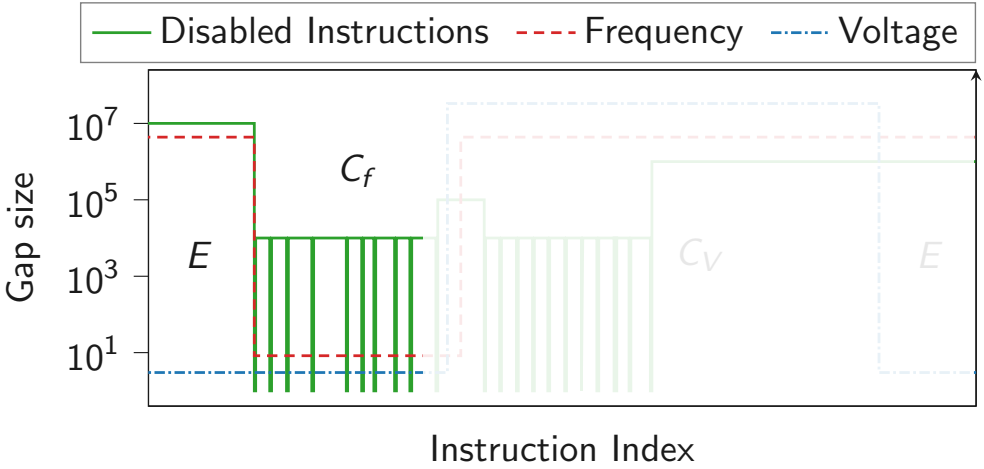
fV Operating Strategy



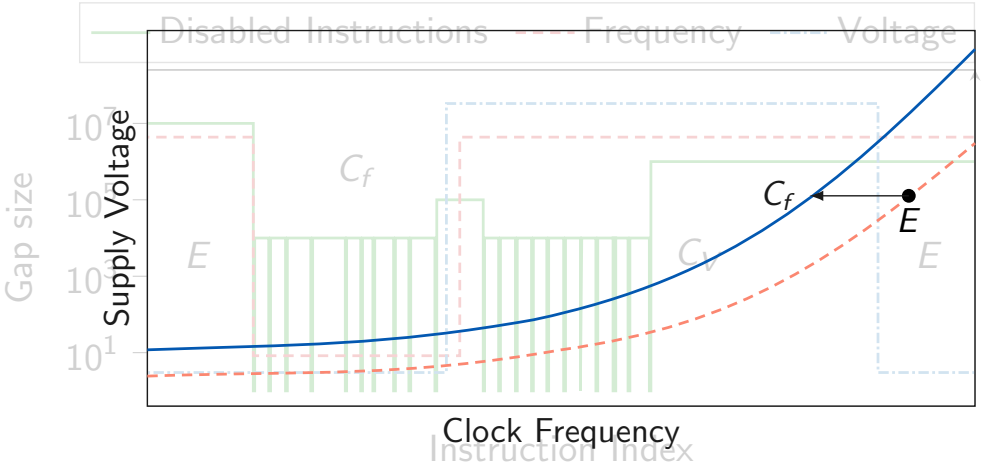
fV Operating Strategy



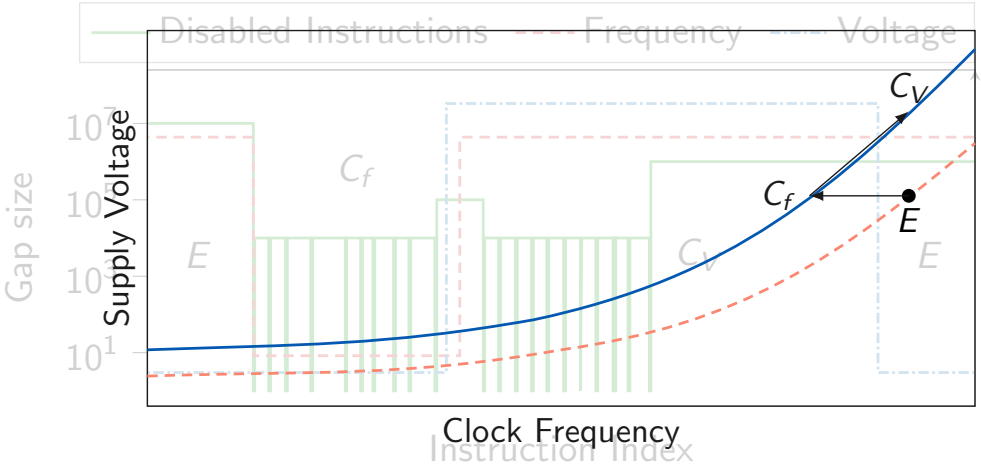
fV Operating Strategy



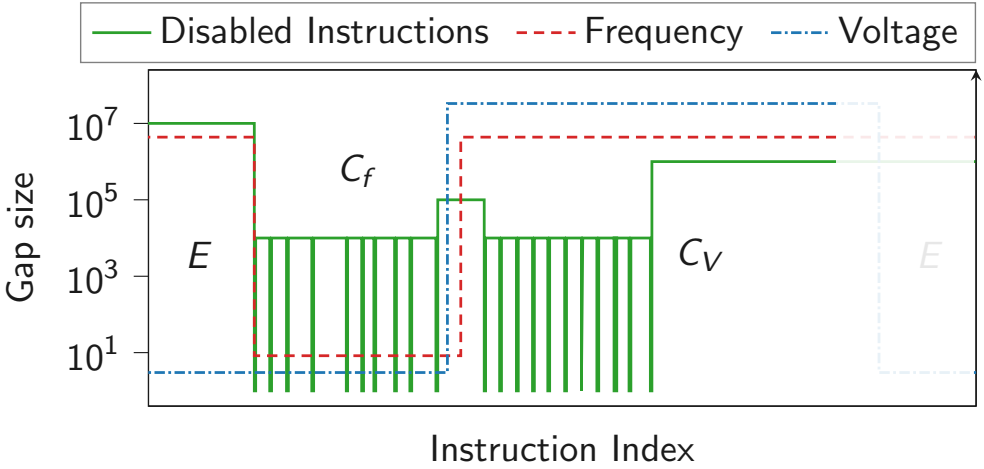
fV Operating Strategy



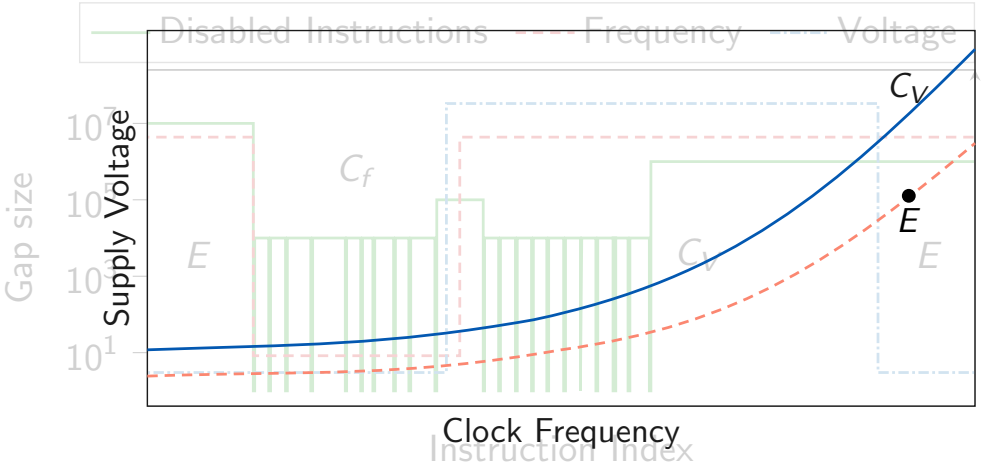
fV Operating Strategy



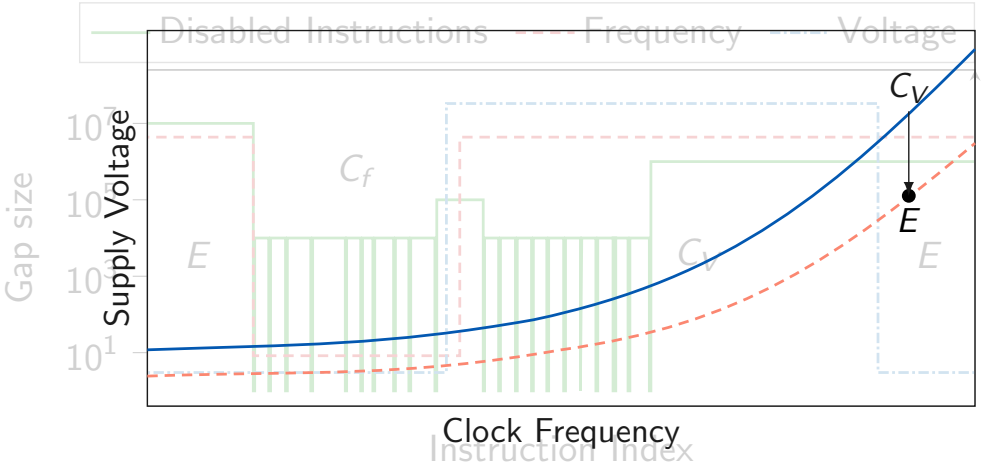
fV Operating Strategy



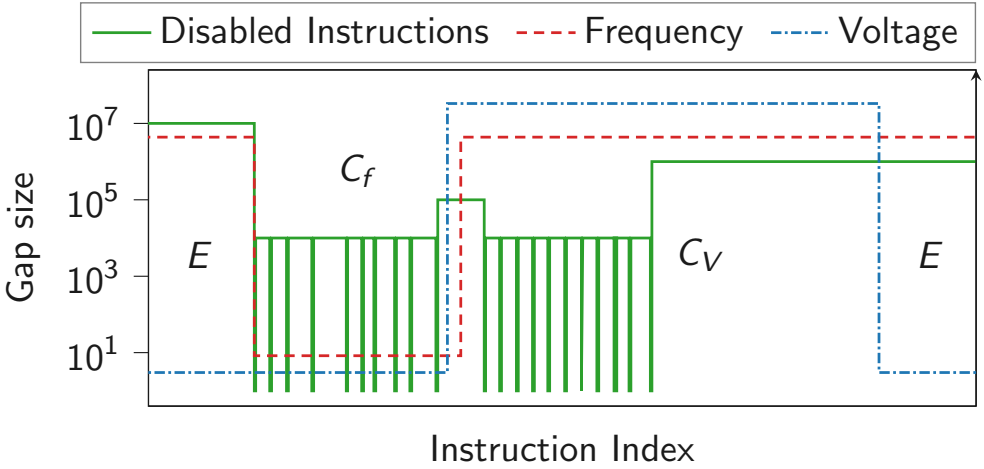
fV Operating Strategy



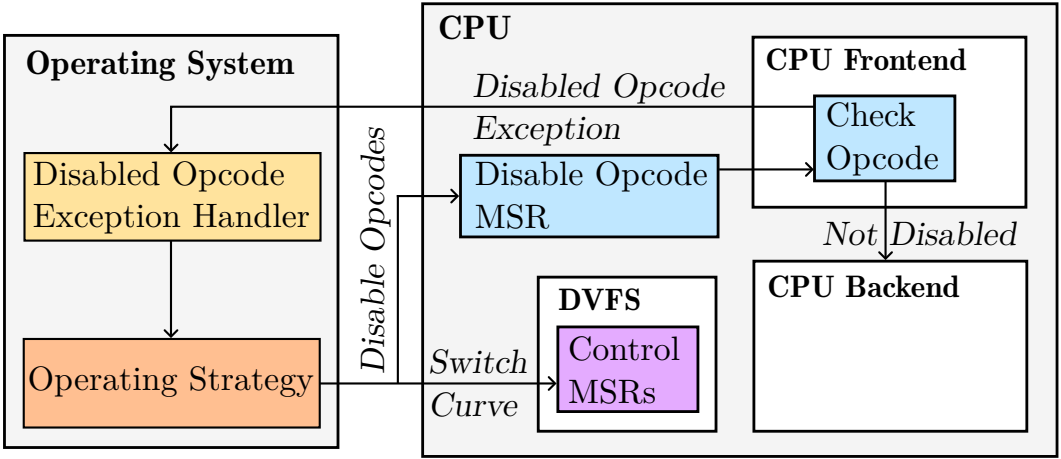
fV Operating Strategy



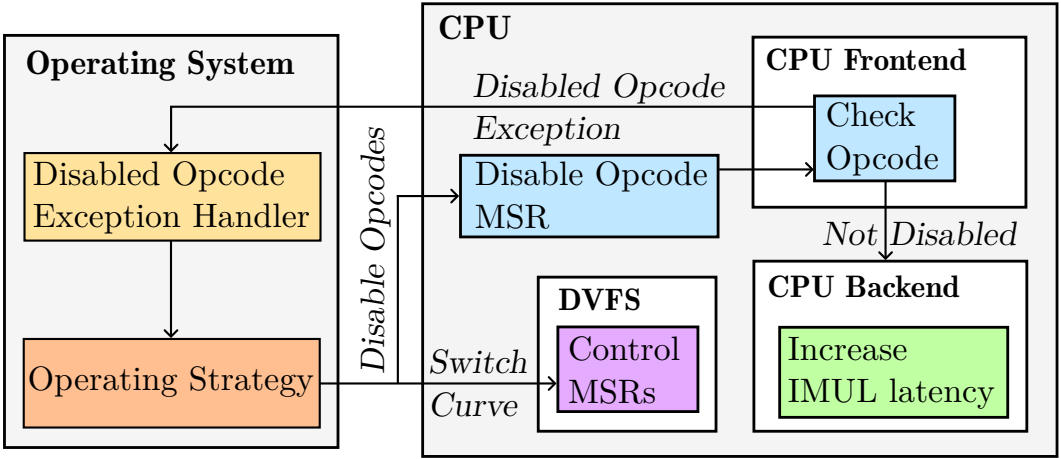
fV Operating Strategy



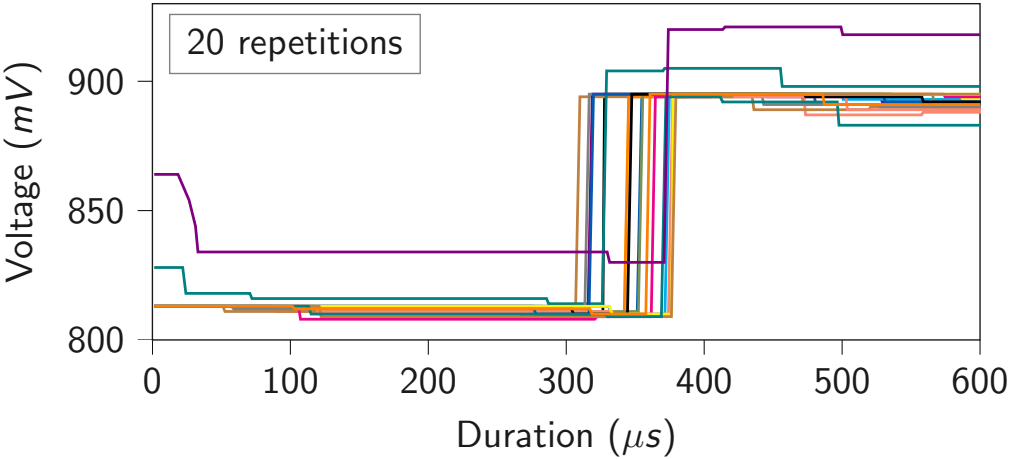
Increase IMUL Latency



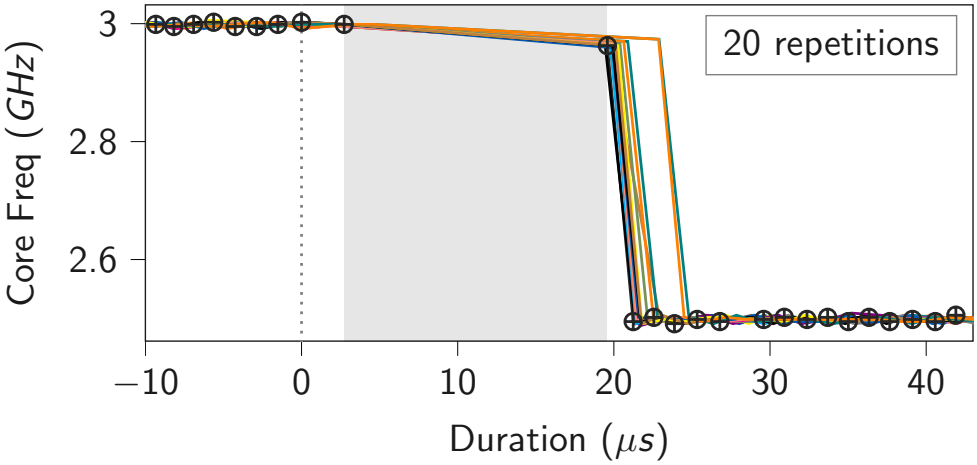
Increase IMUL Latency



Voltage Change Delay



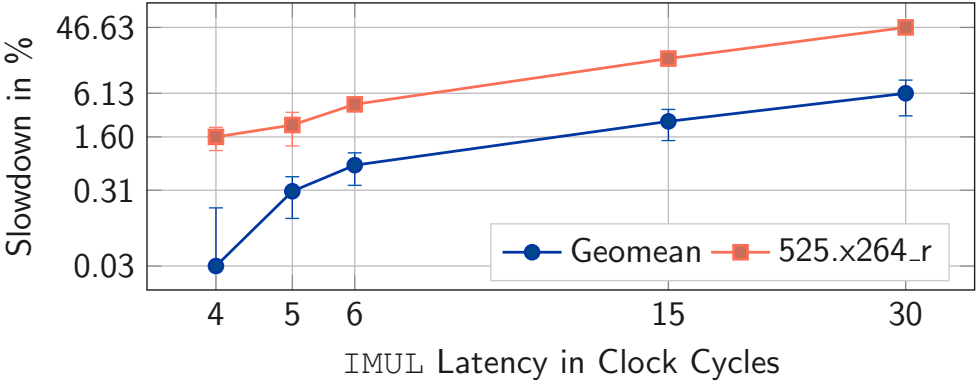
Frequency Change Delay



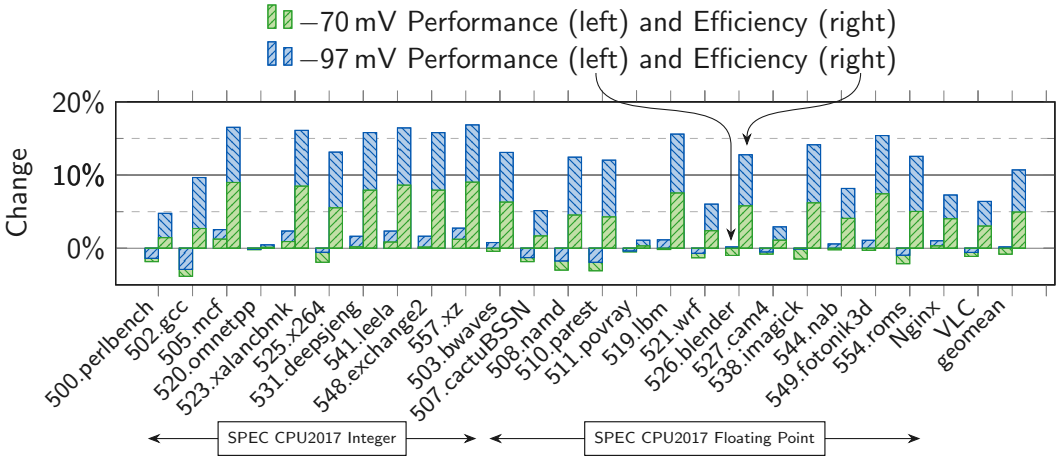
Performance Improvement and Power Savings

CPU	V_{off}	Score	Power	Freq.	Energy Eff.
i5-1035G1	−70 mV	+6.0 %	−0.1 %	+8.5 %	+6.1 %
	−97 mV	+7.9 %	−0.5 %	+12 %	+8.4 %
i9-9900K	−70 mV	+2.2 %	−7.2 %	+2.6 %	+10 %
	−97 mV	+3.8 %	−16 %	+3.3 %	+23 %
7700X*	−70 mV	+1.4 %	−9.8 %	+1.8 %	+12 %
	−97 mV	+1.9 %	−15 %	+1.8 %	+20 %

Increasing the IMUL Latency



SPEC CPU2017 Results



More Results

CPU	$cores$ OS		70 mV Undervolt						97 mV Undervolt					
			$SPEC_{gmean}$	$SPEC_{median}$	525.x264	$SPEC_{noSIMD}$	Nginx	VLC	$SPEC_{gmean}$	$SPEC_{median}$	525.x264	$SPEC_{noSIMD}$	Nginx	VLC
$\mathcal{A}_1$	fV	Pwr	−5.62 %	−7.05 %	−7.05 %	−7.05 %	−3.55 %	−3.88 %	−9.75 %	−10.9 %	−12.1 %	−14.8 %	−5.81 %	−6.30 %
		Perf.	−0.25 %	−1.31 %	−1.31 %	+2.97 %	+0.50 %	−0.39 %	+0.80 %	+1.35 %	0.06 %	+3.45 %	+1.20 %	+0.18 %
		Eff.	+5.70 %	+6.18 %	+6.18 %	+10.8 %	+4.20 %	+3.63 %	+11.7 %	+13.7 %	+13.8 %	+21.4 %	+7.44 %	+6.92 %
$\mathcal{A}_4$	fV	Pwr	−4.62 %	−0.11 %	−6.92 %	−7.41 %	−0.97 %	−1.00 %	−8.87 %	−8.67 %	−13.1 %	−16.2 %	−1.57 %	−1.57 %
		Perf.	−3.93 %	−0.04 %	−7.87 %	+1.82 %	−0.26 %	−0.58 %	−3.58 %	−3.47 %	−7.25 %	+1.84 %	−0.14 %	−0.53 %
		Eff.	+0.72 %	0.07 %	−1.01 %	+9.97 %	+0.72 %	+0.43 %	+5.80 %	+5.70 %	+6.70 %	+21.6 %	+1.45 %	+1.05 %
$\mathcal{A}_\infty$	e	Pwr	−7.50 %	−7.58 %	−5.40 %	−7.50 %	−7.24 %	−7.24 %	−12.3 %	−12.4 %	−10.3 %	−16.6 %	−12.1 %	−12.1 %
		Perf.	−41.6 %	−11.8 %	+6.16 %	+1.42 %	−98.5 %	−91.9 %	−41.9 %	−11.9 %	+6.10 %	+1.42 %	−98.5 %	−91.9 %
		Eff.	−36.9 %	−4.51 %	+12.2 %	+9.63 %	−98.3 %	−91.2 %	−33.7 %	+0.58 %	+18.3 %	+21.6 %	−98.3 %	−90.7 %
$\mathcal{B}_\infty$	f	Pwr	−8.14 %	−7.80 %	−7.80 %	−9.13 %	−4.42 %	−4.43 %	−11.5 %	−10.8 %	−10.8 %	−14.1 %	−6.71 %	−6.73 %
		Perf.	−7.82 %	−7.83 %	−9.25 %	+0.42 %	−2.50 %	−2.52 %	−10.3 %	−10.8 %	−12.2 %	+0.58 %	−2.30 %	−2.33 %
		Eff.	+0.34 %	−0.03 %	−1.57 %	+10.5 %	+2.01 %	+2.00 %	+1.40 %	0.05 %	−1.57 %	+17.1 %	+4.73 %	+4.72 %
	e	Pwr	−9.18 %	−8.02 %	−10.8 %	−9.18 %	−9.79 %	−9.79 %	−14.4 %	−13.3 %	−15.9 %	−14.4 %	−14.9 %	−14.9 %
		Perf.	−26.4 %	−5.12 %	+14.5 %	−0.54 %	−95.7 %	−79.8 %	−26.1 %	−5.25 %	+18.5 %	0.01 %	−95.7 %	−79.8 %
		Eff.	−19.0 %	+3.15 %	+28.3 %	+9.51 %	−95.3 %	−77.6 %	−13.7 %	+9.26 %	+40.9 %	+16.8 %	−95.0 %	−76.2 %
$\mathcal{C}_\infty$	fV	Pwr	−5.64 %	−7.05 %	−7.05 %	−6.12 %	−3.56 %	−4.03 %	−9.78 %	−11.2 %	−12.1 %	−14.1 %	−5.83 %	−6.55 %
		Perf.	−0.85 %	−1.92 %	−1.92 %	+3.53 %	+0.33 %	−1.12 %	+0.19 %	+0.19 %	−0.55 %	+3.79 %	+1.03 %	−0.57 %
		Eff.	+5.07 %	+5.53 %	+5.53 %	+10.3 %	+4.04 %	+3.03 %	+11.0 %	+12.8 %	+13.1 %	+20.8 %	+7.28 %	+6.40 %

%	+10.8 %	+4.20 %	+3.63 %	+11.7 %	+13.7 %	+13.8 %	+21.4 %	+7.44 %	+6.92 %
%	−7.41 %	−0.97 %	−1.00 %	−8.87 %	−8.67 %	−13.1 %	−16.2 %	−1.57 %	[redacted]
%	+1.82 %	−0.26 %	−0.58 %	−3.58 %	−3.47 %	−7.25 %	+1.84 %	−0.14 %	−0.53 %
%	+9.97 %	+0.72 %	+0.43 %	+5.80 %	+5.70 %	+6.70 %	+21.6 %	+1.45 %	+1.05 %
%	−7.50 %	−7.24 %	−7.24 %	−12.3 %	−12.4 %	−10.3 %	−16.6 %	−12.1 %	−12.1 %
%	+1.42 %	−98.5 %	−91.9 %	−41.9 %	−11.9 %	+6.10 %	+1.42 %	−98.5 %	−91.9 %
%	+9.63 %	−98.3 %	−91.2 %	−33.7 %	+0.58 %	+18.3 %	+21.6 %	−98.3 %	−90.7 %
%	−9.13 %	−4.42 %	−4.43 %	−11.5 %	−10.8 %	−10.8 %	−14.1 %	−6.71 %	−6.73 %
%	+0.42 %	−2.50 %	−2.52 %	−10.3 %	−10.8 %	−12.2 %	+0.58 %	−2.30 %	−2.33 %
%	+10.5 %	+2.01 %	+2.00 %	+1.40 %	0.05 %	−1.57 %	+17.1 %	+4.73 %	+4.72 %
%	−9.18 %	−9.79 %	−9.79 %	−14.4 %	−13.3 %	−15.9 %	−14.4 %	−14.9 %	−14.9 %
%	−0.54 %	−95.7 %	−79.8 %	−26.1 %	−5.25 %	+18.5 %	0.01 %	−95.7 %	−79.8 %
%	+9.51 %	−95.3 %	−77.6 %	−13.7 %	+9.26 %	+40.9 %	+16.8 %	−95.0 %	−76.2 %
%	−6.12 %	−3.56 %	−4.03 %	−9.78 %	−11.2 %	−12.1 %	−14.1 %	−5.83 %	−6.55 %
%	+3.53 %	+0.33 %	−1.12 %	+0.19 %	+0.19 %	−0.55 %	+3.79 %	+1.03 %	−0.57 %
%	+10.3 %	+4.04 %	+3.03 %	+11.0 %	+12.8 %	+13.1 %	+20.8 %	+7.28 %	+6.40 %

%	+10.8 %	+4.20 %	+3.63 %	+11.7 %	+13.7 %	+13.8 %	+21.4 %	+7.44 %	+6.92 %
%	−7.41 %	−0.97 %	−1.00 %	−8.87 %	−8.67 %	−13.1 %	−16.2 %	−1.57 %	[redacted]
%	+1.82 %	−0.26 %	−0.58 %	−3.58 %	−3.47 %	−7.25 %	+1.84 %	−0.14 %	−0.53 %
%	+9.97 %	+0.72 %	+0.43 %	+5.80 %	+5.70 %	+6.70 %	+21.6 %	+1.45 %	+1.05 %
%	−7.50 %	−7.24 %	−7.24 %	−12.3 %	−12.4 %	−10.3 %	−16.6 %	−12.1 %	−12.1 %
%	+1.42 %	−98.5 %	−91.9 %	−41.9 %	−11.9 %	+6.10 %	+1.42 %	−98.5 %	−91.9 %
%	+9.63 %	−98.3 %	−91.2 %	−33.7 %	+0.58 %	+18.3 %	+21.6 %	−98.3 %	−90.7 %
%	−9.13 %	−4.42 %	−4.43 %	−11.5 %	−10.8 %	−10.8 %	−14.1 %	−6.71 %	−6.73 %
%	+0.42 %	−2.50 %	−2.52 %	−10.3 %	−10.8 %	−12.2 %	+0.58 %	−2.30 %	−2.33 %
%	+10.5 %	+2.01 %	+2.00 %	+1.40 %	0.05 %	−1.57 %	+17.1 %	+4.73 %	+4.72 %
%	−9.18 %	−9.79 %	−9.79 %	−14.4 %	−13.3 %	−15.9 %	−14.4 %	−14.9 %	−14.9 %
%	−0.54 %	−95.7 %	−79.8 %	−26.1 %	−5.25 %	+18.5 %	0.01 %	−95.7 %	−79.8 %
%	+9.51 %	−95.3 %	−77.6 %	−13.7 %	+9.26 %	+40.9 %	+16.8 %	−95.0 %	−76.2 %
%	−6.12 %	−3.56 %	−4.03 %	−9.78 %	−11.2 %	−12.1 %	−14.1 %	−5.83 %	−6.55 %
%	+3.53 %	+0.33 %	−1.12 %	+0.19 %	+0.19 %	−0.55 %	+3.79 %	+1.03 %	−0.57 %
%	+10.3 %	+4.04 %	+3.03 %	+11.0 %	+12.8 %	+13.1 %	+20.8 %	+7.28 %	+6.40 %

Conclusion

Conclusion

- Decade-old problems like Rowhammer can be solved with principled security

Conclusion

- Decade-old problems like Rowhammer can be solved with principled security
- Adding security can increase efficiency

Conclusion

- Decade-old problems like Rowhammer can be solved with principled security
- Adding security can increase efficiency
- New and unexplored area that needs a lot more research

Security:
Can we afford to have it?
Can we afford not to have it?

Daniel Gruss

2024-10-21

Graz University of Technology