# Hardware Security

## with Microarchitecture Side Channels as Example

*Shih-Lien (Linus) Lu*

WASHINGTON STATE UNIVERSITY
EVERETT

# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
    - Technology/Transistors
    - Packaging/Board/System
        - Firmware
        - Software
    - Design process
        - Circuits, logic, RTL, architecture
    - Supply chain
- Security goals in the hardware context – CIA
    - Understand hardware attack surfaces
        - Threat
        - Vulnerability
            - Microarchitecture side-channels
    - Attack methods
    - Mitigation techniques

# Objectives of Information Security - CIA

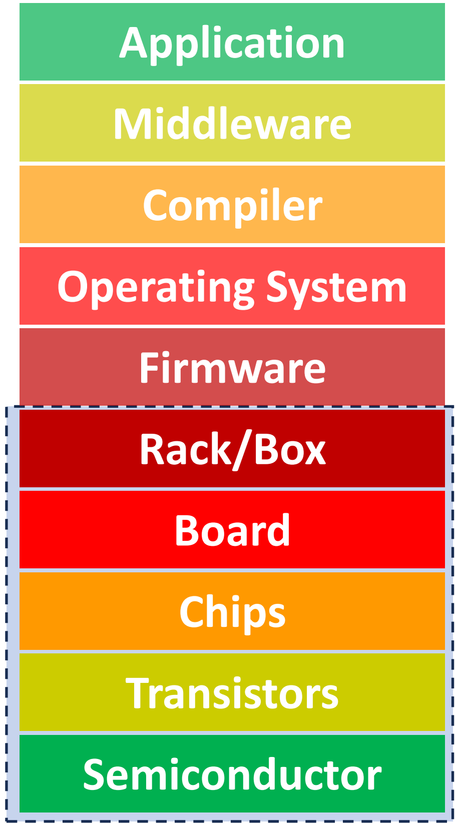
- **Confidentiality**
  - **Only authorized parties should be able to access the required data**
    - **Access mean to understand the contents of the data**
    - **Should not disclose unnecessary content**
- **Integrity**
  - **The content should only be altered by authorized users intentionally**
    - **Not tempered or degraded**
    - **Purposely or inadvertently**
  - **Non-repudiation**
- **Availability**
  - **Timely accessibility of data to authorized entities when needed**
    - **System availability**
    - **Communication channel accessibility**
    - **Data readiness**

# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
  - Technology/Transistors
  - Packaging/Board/System
    - Firmware
    - Software
  - Design process
    - Circuits, logic, RTL, architecture
  - Supply chain
- Security goals in the hardware context – CIA
  - Understand hardware attack surfaces
    - Threat
    - Vulnerability
      - Microarchitecture side-channels
  - Attack methods
  - Mitigation techniques

# Hardware Stack (vs. Software Stack)

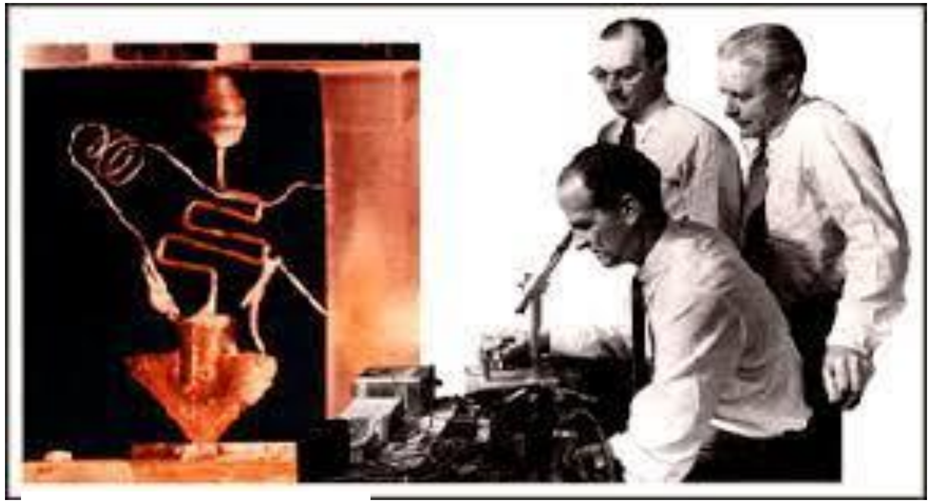
- SW stack
  - Layers/components translating an application for HW
    - Middleware includes any API/library/functions
    - OS includes UI/services/runtimes
    - An additional layer may sit between OS and firmware (hypervisor)
    - Firmware includes BIOS/drivers
- HW stack
  - Layers/components to support the execution of software
    - Rack may be pleural
    - Much of HW suppliers are outside of the US
      - Semiconductor
      - Recent resurgence of focus

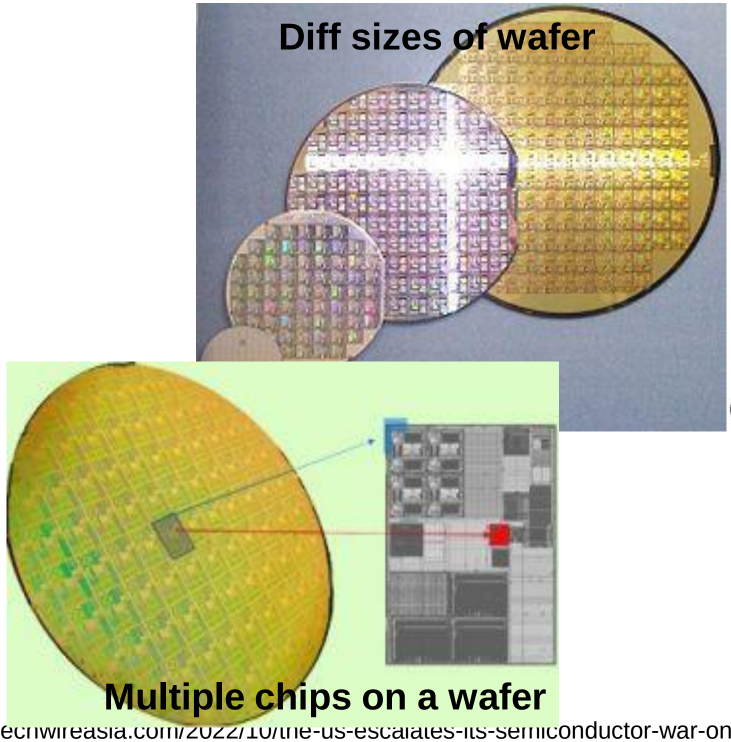


# Semiconductor Technology/Transistors

- Transistors
  - One of the important invention
- Integration of devices
  - Many transistors on a substrate
- Manufacturing is moving offshore

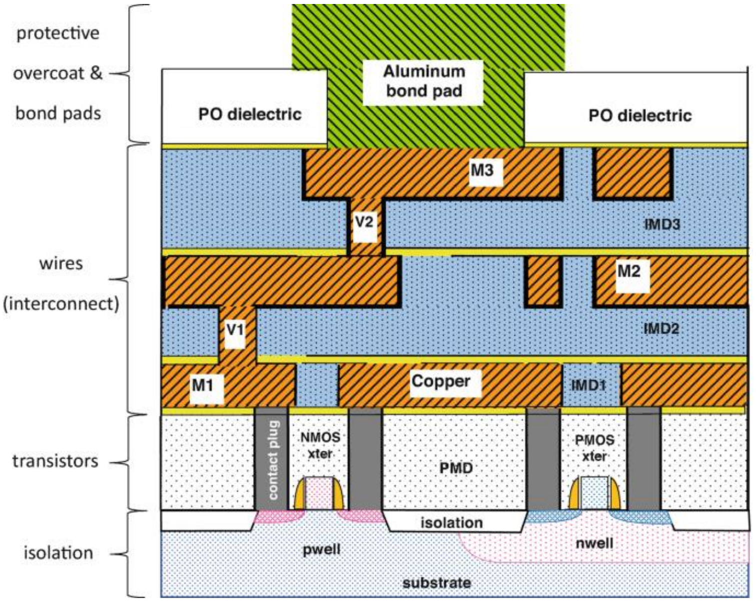
1st transistor (L) & the three "inventors" (Bardeen, Brattain, and Shockley)

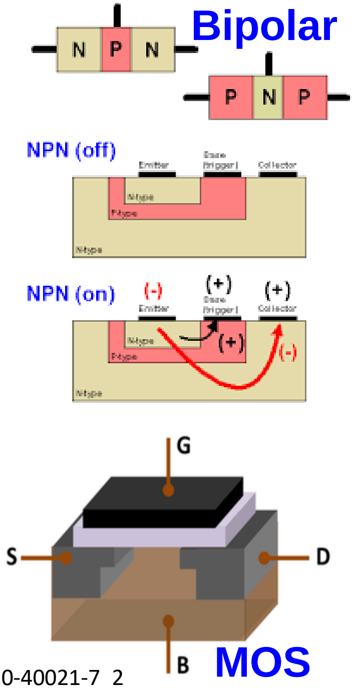




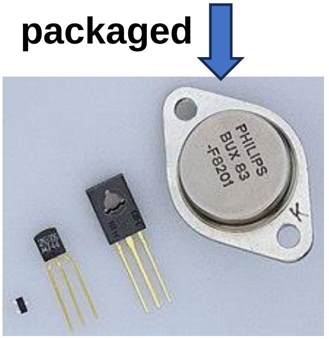
Diff sizes of wafer

Multiple chips on a wafer





packaged



Vacuum Tube

Transistor

Integrated Circuit (IC)

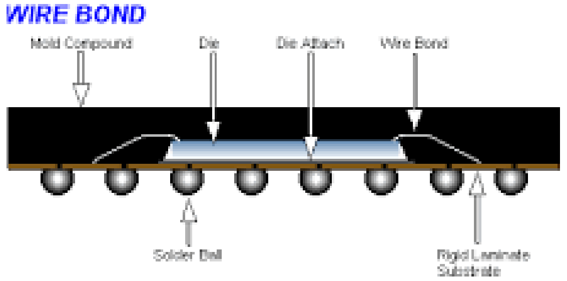
https://techwireasia.com/2022/10/the-us-escalates-its-semiconductor-war-on-china-what-happens-now/

Tigelaar, H. (2020). Overview of Integrated Circuit Manufacturing. In: How Transistor Area Shrank by 1 Million Fold. Springer, Cham. https://doi.org/10.1007/978-3-030-40021-7_2

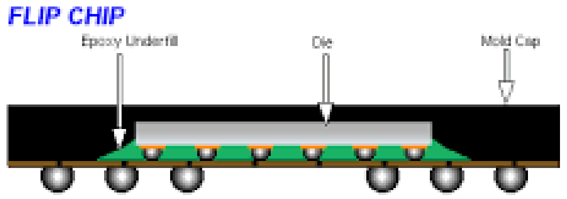
https://en.wikipedia.org/wiki/Transistor

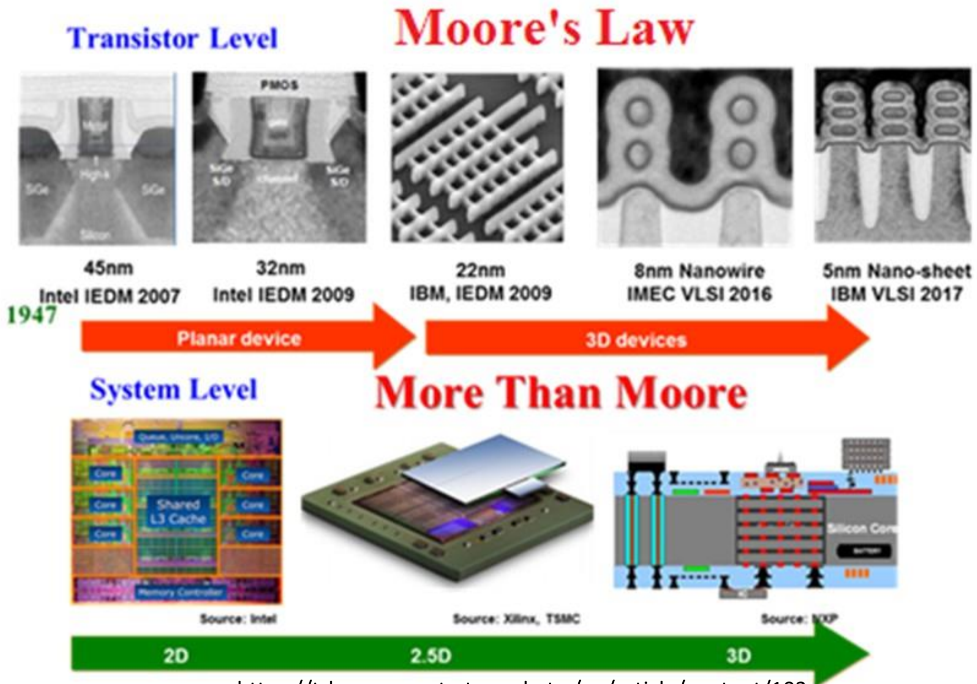
# Packaging/Assembly

- Packaging technologies matured
  - Moving off-shore
  - Supply chain
- More than Moore (MtM)
  - Packaging going 3D
  - IC die size has not shrunk with Moore's Law
  - Apply IC technology to packaging
  - Reduce physical footprint
  - Challenge - power density

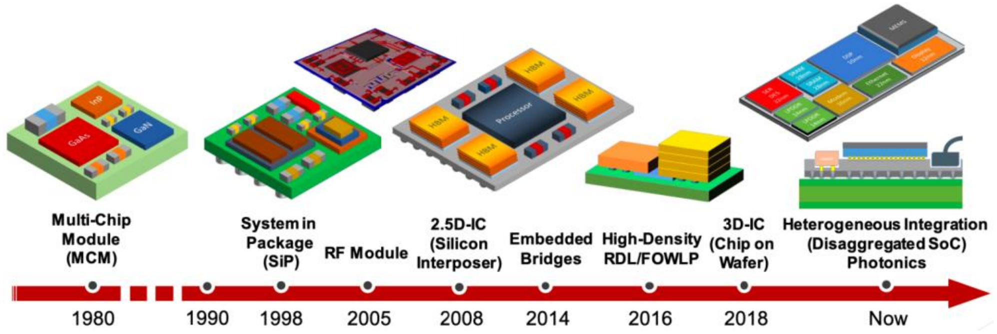


https://www.eetimes.com/the-package-interconnect-selection-quandary/





https://trh.gase.most.ntnu.edu.tw/en/article/content/103



https://semiengineering.com/more-than-moore-reality-check/
https://irle.berkeley.edu/files/2005/Offshoring-in-the-Semiconductor-Industry.pdf

# Chips to Board to Rack to Room

Packaged chips on a board

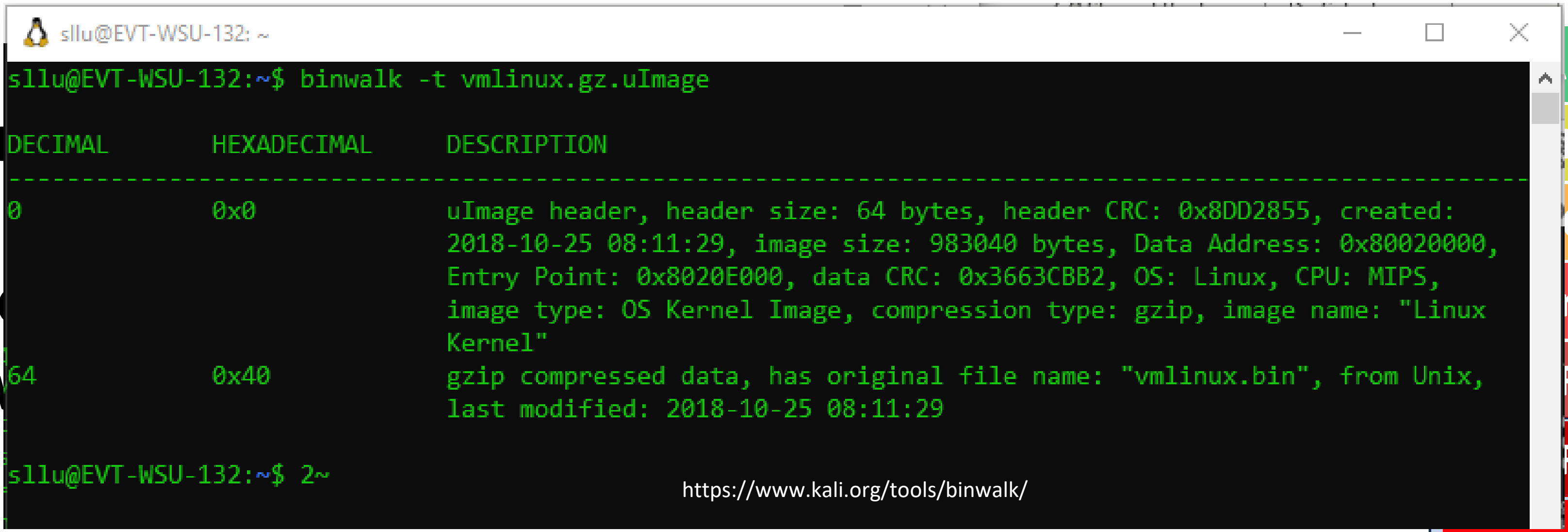
System with multiple boards

Multiple racks

Roomful of racks

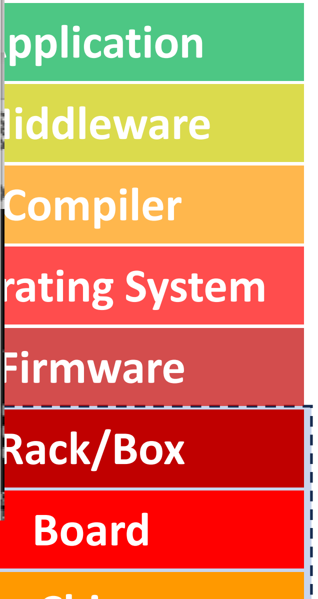
# Firmware and Software

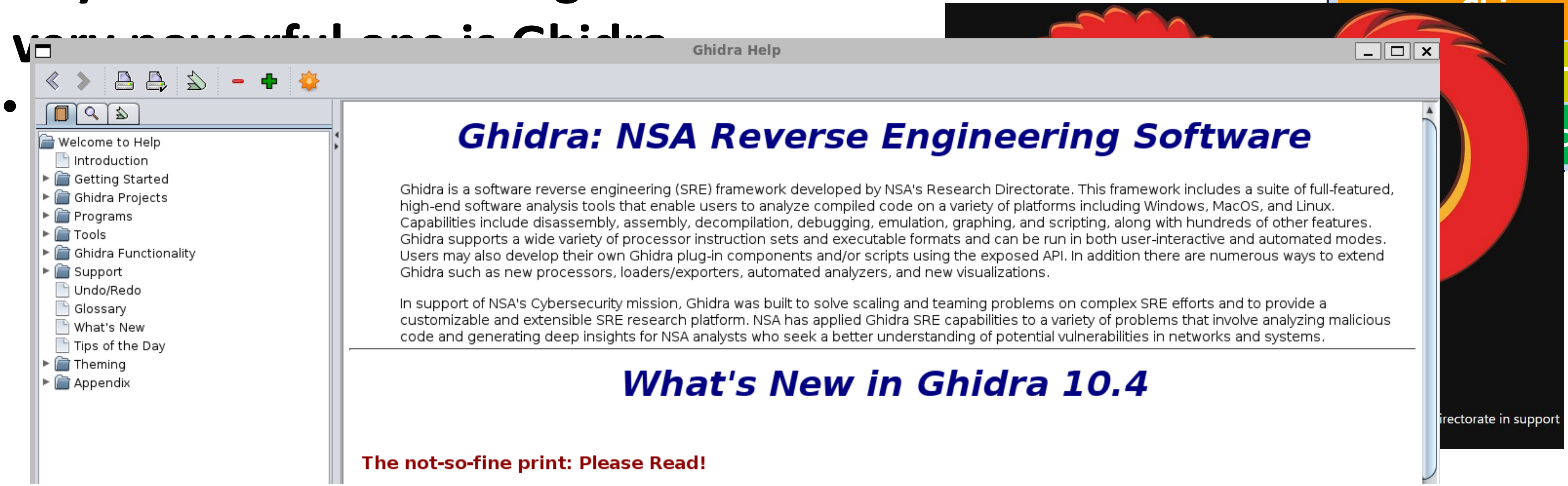
- Firmware
  - Software i
    - May inc
  - On most l
  - CptS426 w
- Software
  - Many tools to reverse engineer SW
  - A v
    -



```
sllu@EVT-WSU-132:~$ binwalk -t vmlinux.gz.uImage

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------------------------------
0             0x0             uImage header, header size: 64 bytes, header CRC: 0x8DD2855, created:
                              2018-10-25 08:11:29, image size: 983040 bytes, Data Address: 0x80020000,
                              Entry Point: 0x8020E000, data CRC: 0x3663CBB2, OS: Linux, CPU: MIPS,
                              image type: OS Kernel Image, compression type: gzip, image name: "Linux
                              Kernel"
64            0x40            gzip compressed data, has original file name: "vmlinux.bin", from Unix,
                              last modified: 2018-10-25 08:11:29

sllu@EVT-WSU-132:~$ 2~
```

https://www.kali.org/tools/binwalk/





## Ghidra: NSA Reverse Engineering Software

Ghidra is a software reverse engineering (SRE) framework developed by NSA's Research Directorate. This framework includes a suite of full-featured, high-end software analysis tools that enable users to analyze compiled code on a variety of platforms including Windows, MacOS, and Linux. Capabilities include disassembly, assembly, decompilation, debugging, emulation, graphing, and scripting, along with hundreds of other features. Ghidra supports a wide variety of processor instruction sets and executable formats and can be run in both user-interactive and automated modes. Users may also develop their own Ghidra plug-in components and/or scripts using the exposed API. In addition there are numerous ways to extend Ghidra such as new processors, loaders/exporters, automated analyzers, and new visualizations.

In support of NSA's Cybersecurity mission, Ghidra was built to solve scaling and teaming problems on complex SRE efforts and to provide a customizable and extensible SRE research platform. NSA has applied Ghidra SRE capabilities to a variety of problems that involve analyzing malicious code and generating deep insights for NSA analysts who seek a better understanding of potential vulnerabilities in networks and systems.

## What's New in Ghidra 10.4

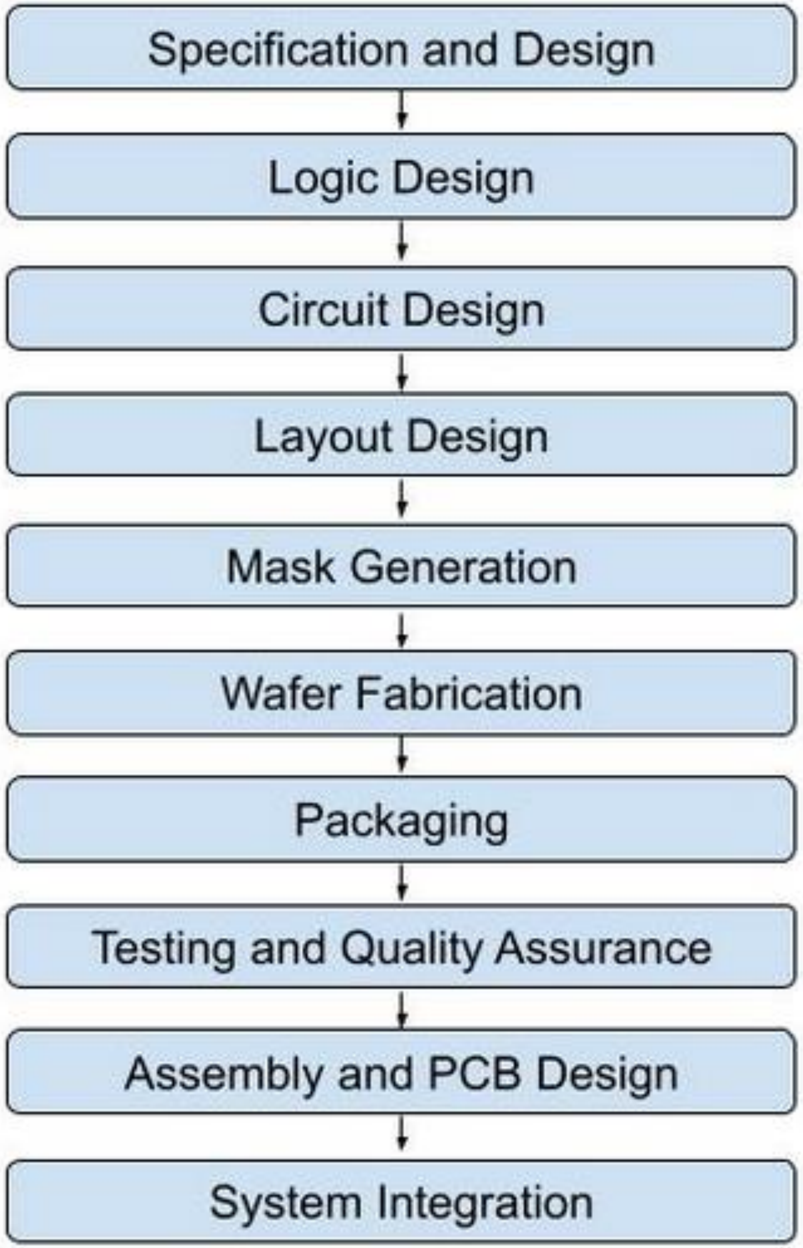
**The not-so-fine print: Please Read!**

# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
  - Technology/Transistors
  - Packaging/Board/System
    - Firmware
    - Software
  - Design process
    - Circuits, logic, RTL, architecture
  - Supply chain
- Security goals in the hardware context – CIA
  - Understand hardware attack surfaces
    - Threat
    - Vulnerability
      - Microarchitecture side-channels
  - Attack methods
  - Mitigation techniques

# Design Process

- Start with specification
- Design the architecture (microarchitecture)
- Perform logic design
  - HDL (C, System Verilog, Verilog, VHDL)
- Circuit design
  - Library cells
- Physical layout
  - GDSII
- Mask generation – MEBES and OPC
- Fabrication – many months
- Packaging – 2D, 2.5D, 3D
- Testing – pre- and post testing
- Assembly – putting parts together
- System integration



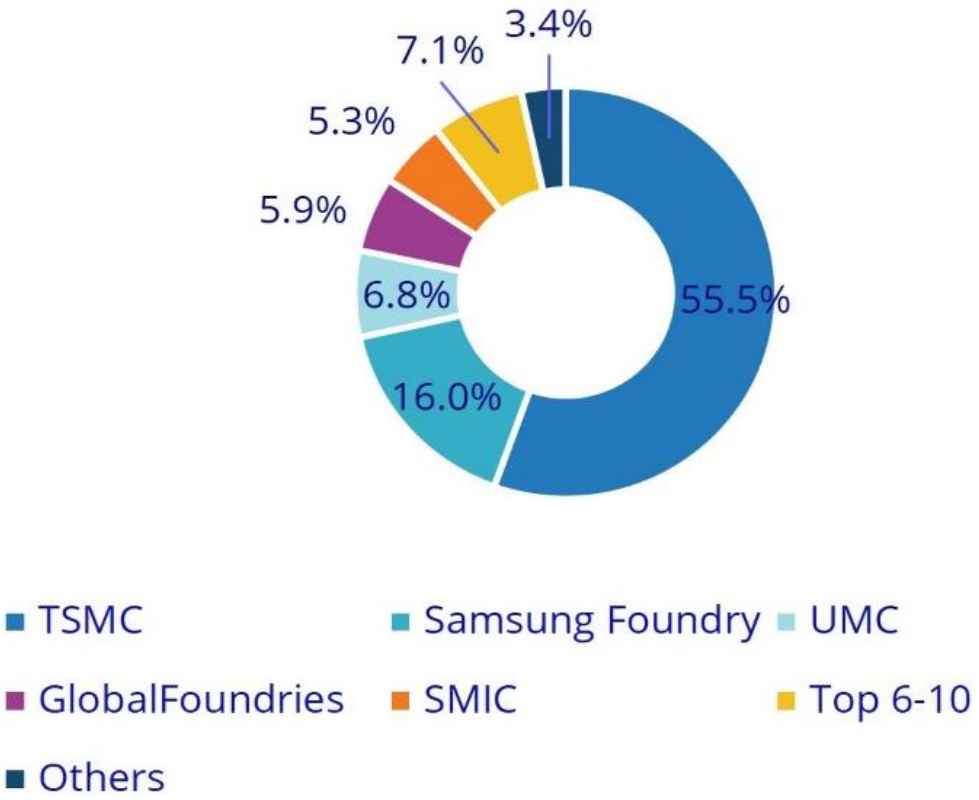
https://resources.pcb.cadence.com/blog/2023-ic-design-and-manufacturing-process

# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
    - Technology/Transistors
    - Packaging/Board/System
        - Firmware
        - Software
    - Design process
        - Circuits, logic, RTL, architecture
    - **Supply chain**
- **Security goals in the hardware context – CIA**
    - **Understand hardware attack surfaces**
        - **Threat**
        - **Vulnerability**
            - **Microarchitecture side-channels**
    - **Attack methods**
    - **Mitigation techniques**

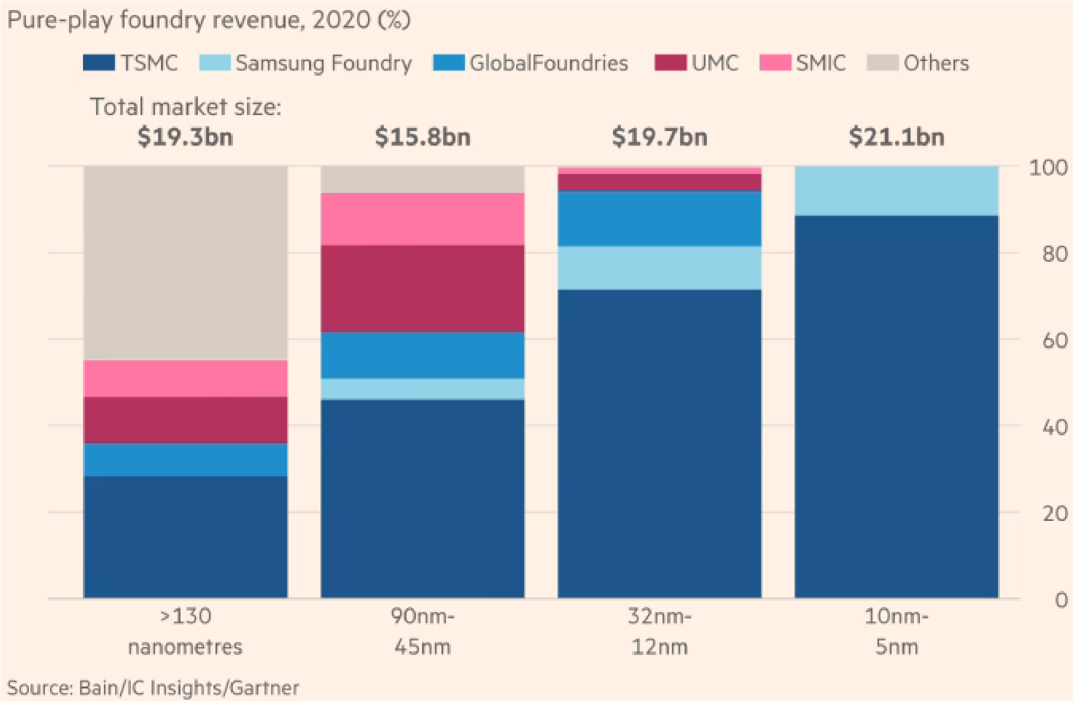
# Distribution of Semiconductor Foundry

- Total market share
  - TSMC (Taiwan Semiconductor Manufacturing Comp.) > 55%



https://www.idc.com/getdoc.jsp?containerId=prAP50994023

- Advanced nodes
  - TSMC ~ 90%
  - > 90 % for the most advanced technology



Source: Bain/IC Insights/Gartner

https://www.ft.com/content/05206915-fd73-4a3a-92a5-6760ce965bd9

**Secure/Trusted Hardware Supply Chain Plays Important Role to Achieve Security Goals**

# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
    - Technology/Transistors
    - Packaging/Board/System
        - Firmware
        - Software
    - Design process
        - Circuits, logic, RTL, architecture
    - Supply chain
- **Security goals in the hardware context – CIA**
    - **Understand hardware attack surfaces**
        - **Threat**
        - **Vulnerability**
            - **Microarchitecture side-channels**
    - **Attack methods**
    - **Mitigation techniques**

# Security Goals (CIA) in the Hardware Context

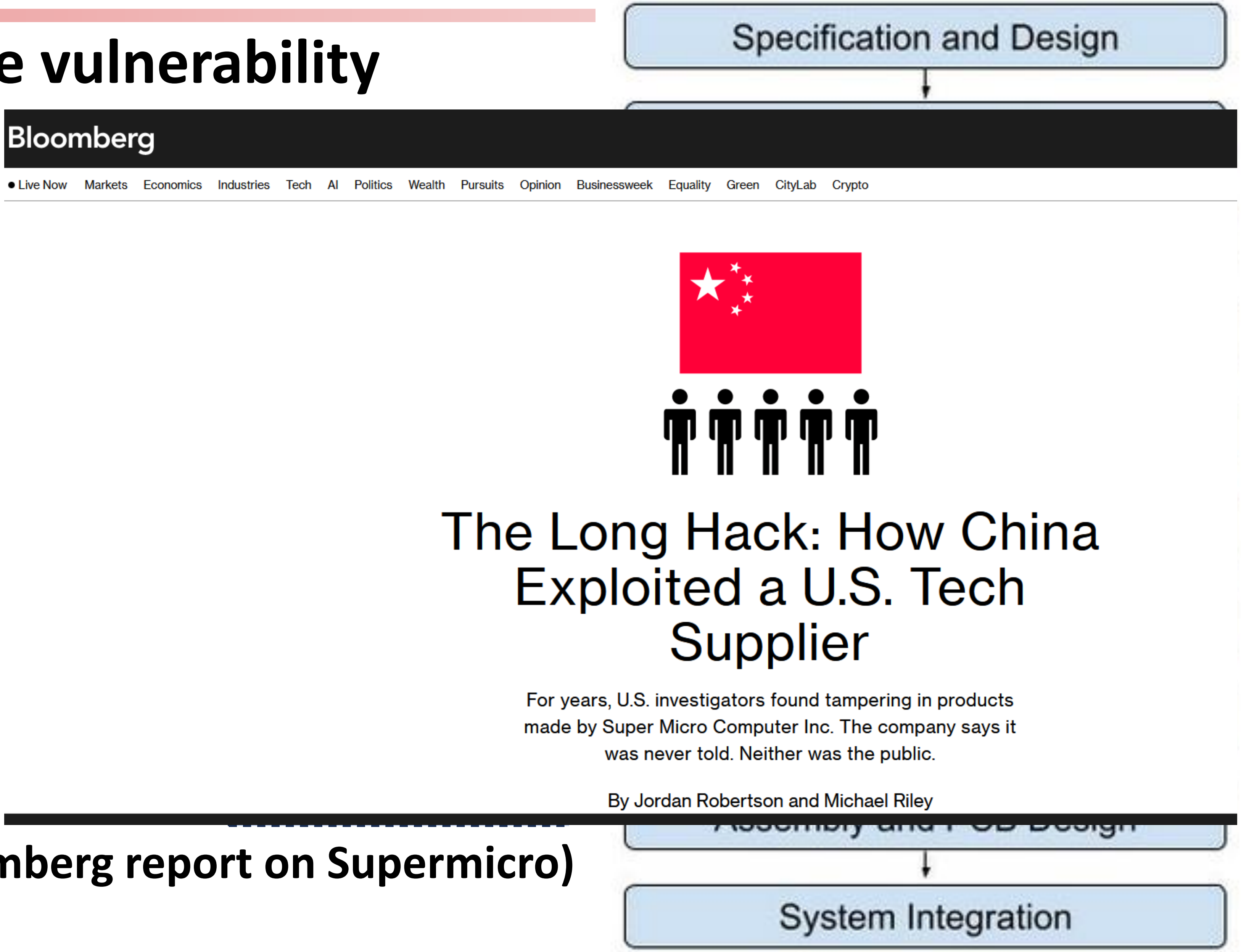
- Understand security terminology
  - Threat (threat actor)
  - Vulnerability
  - Attack
- Attack surfaces
  - Places/areas that have vulnerability
  - Attack causes violation of security goals
- Common attack methods
  - Reverse engineering
  - Fault injection
  - Information leakage through side channels
- Mitigation techniques

# Understand Security Terminology

- Threat
  - Condition/circumstances causing violation of security goals (CIA)
  - Both intentional and unintentional (e.g. disaster, power outage)
  - Range from individuals to national state
- Vulnerability
  - Weakness
    - Bugs in design
    - Features
    - Misconfiguration or mis-operation
  - Allow threat actor to act
- Attack
  - Deliberate action
  - With motive and plan

# Attack Surfaces

- Places/areas that have vulnerability
  - Actual hardware
    - System
    - Board
    - Chip
    - Interfaces
      - Between chips
      - Between boards
      - Between systems
  - Design process
    - Actual design
    - Tools used for design
    - Library cells (IPs)
  - Supply chain
    - Fabrication
    - Assembly (2018 Bloomberg report on Supermicro)
    - Integration



The Long Hack: How China Exploited a U.S. Tech Supplier

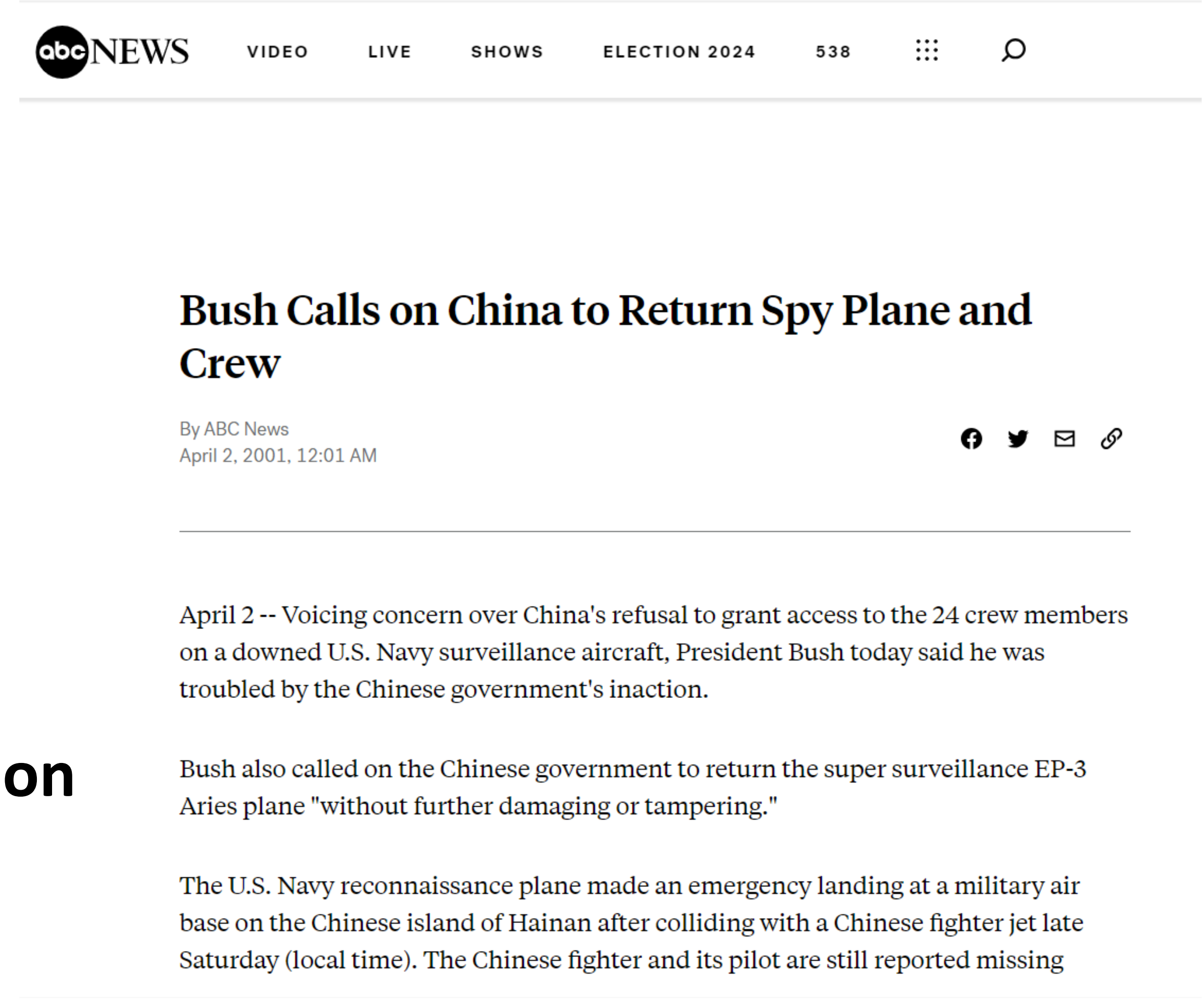
For years, U.S. investigators found tampering in products made by Super Micro Computer Inc. The company says it was never told. Neither was the public.

By Jordan Robertson and Michael Riley

https://www.theregister.com/2021/02/12/supermicro_bloomberg_spying/

# Common Attack Methods

- Reverse engineering
  - System
    - firmware
  - Board
  - Chip - FPGA
    - Zeroization
- Fault injection
  - Perturbation
    - Temperature
    - Voltage
    - Clock
  - Causing system to leak information
- Side channels
  - Physical
  - Microarchitectural



## Bush Calls on China to Return Spy Plane and Crew

By ABC News
April 2, 2001, 12:01 AM

April 2 -- Voicing concern over China's refusal to grant access to the 24 crew members on a downed U.S. Navy surveillance aircraft, President Bush today said he was troubled by the Chinese government's inaction.

Bush also called on the Chinese government to return the super surveillance EP-3 Aries plane "without further damaging or tampering."

The U.S. Navy reconnaissance plane made an emergency landing at a military air base on the Chinese island of Hainan after colliding with a Chinese fighter jet late Saturday (local time). The Chinese fighter and its pilot are still reported missing

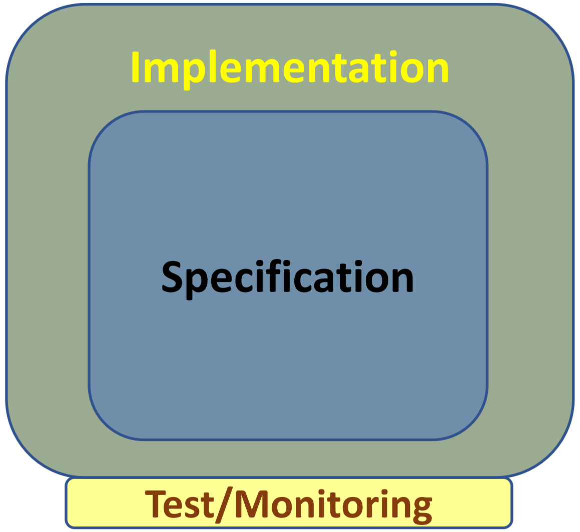
https://abcnews.go.com/International/story?id=81317

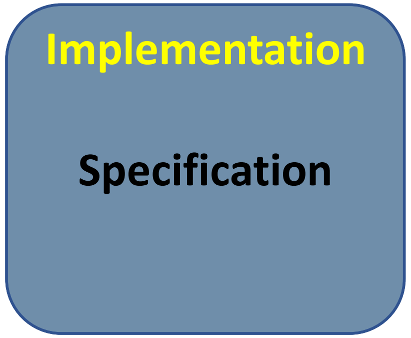
# Outline

- Security goals - CIA
- Hardware stack (vs. software stack)
  - Technology/Transistors
  - Packaging/Board/System
    - Firmware
    - Software
  - Design process
    - Circuits, logic, RTL, architecture
  - Supply chain
- Security goals in the hardware context – CIA
  - Understand hardware attack surfaces
    - Threat
    - Vulnerability
      - **Microarchitecture side-channels**
  - Attack methods
  - **Mitigation techniques**

# Microarchitecture Side-Channels

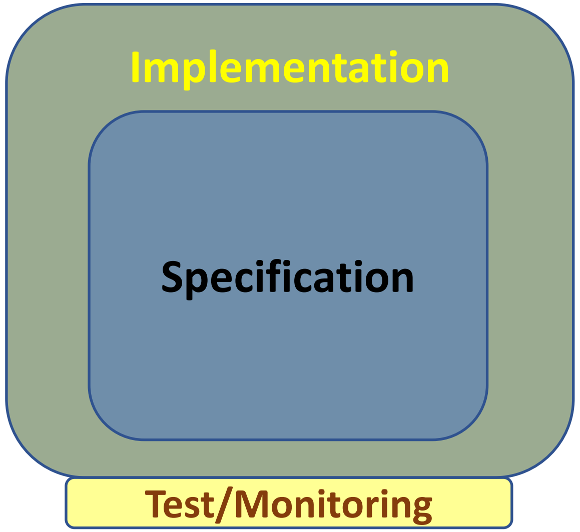
- Why is security challenging (especially in HW context)?
  - Correctness
    - Met specification (e.g. ISA spec)
  - No vulnerability means …
    - No extra features beyond the specification
    - No side-channels
      - Speculation
      - Transient states
      - Physical manifestation
  - But we want …
    - Performance
    - Observability
    - Testing/debugging
    - …





# What is Microarchitecture? Why?

- Implementation of Instruction Set Architecture (ISA)
  - Ex. Of ISA x86-64, ARMv8-A, RISC-V, MIPS
  - ISA is the specification
- Multiple implementations of one ISA to get:
  - Performance
  - Power efficiency
  - Observability
  - Testability
  - Security!
  - Etc.
- SW interactions with the implementation
  - Observable



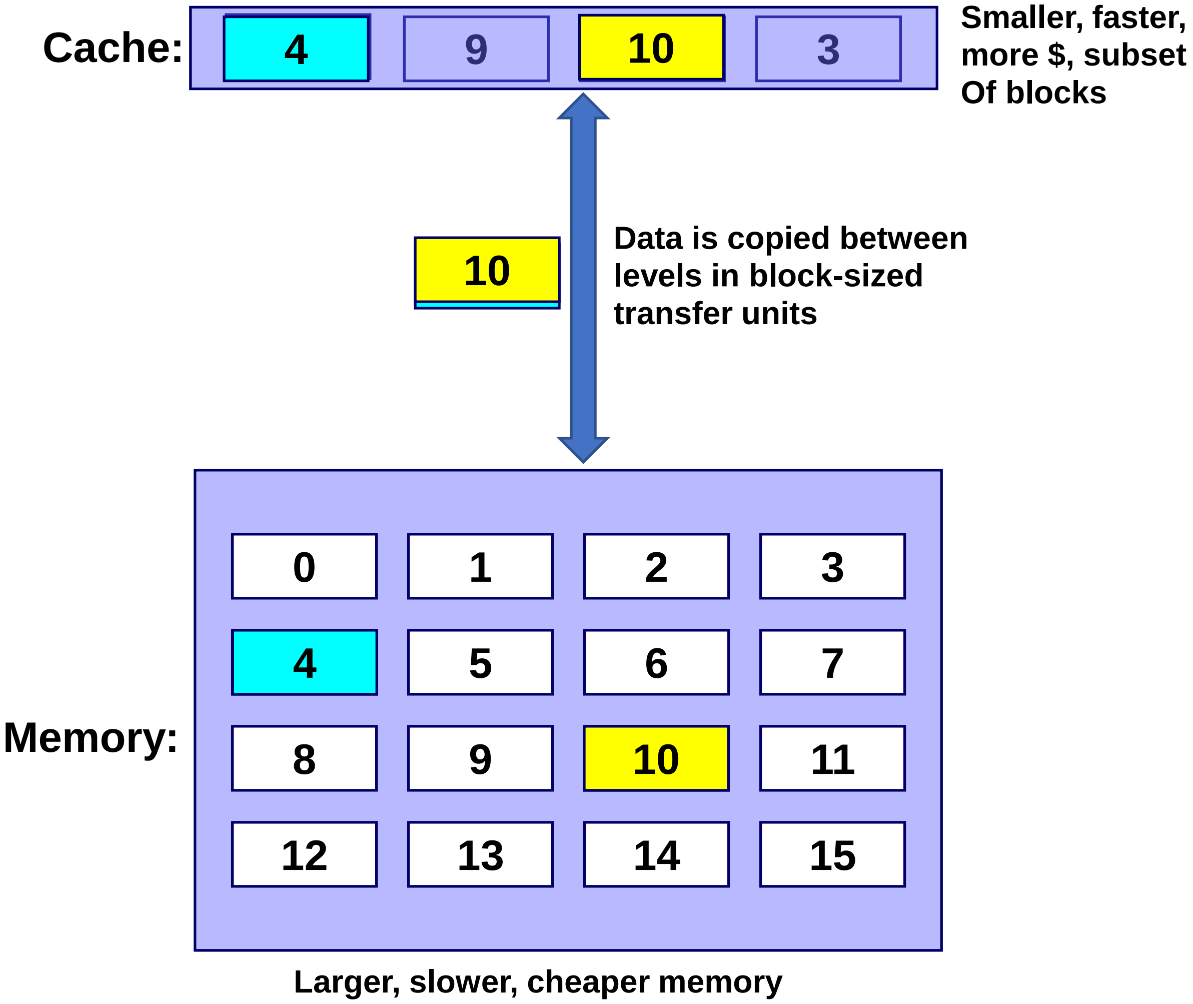
# Microarchitectural Side-Channels

- Why this is so insidious?
  - Need no physical access
    - Cloud: multi-tenant
    - Threat agent is a “legitimate” tenant of the cloud
  - Shared resource as the covert channel
    - Host has no explicit control of the share resources
    - Behavior can be affected/observed by another tenant
  - Conflicting goals – performance vs. security
  - Slowing down of Moore’s law → more microarchitecture optimization
    - Increasing attack surfaces
- Cause
  - Speed up common cases
    - Lower latency when the “technique” works
    - Longer latency otherwise
  - Performance monitoring mechanisms

**Passive Information Gathering & Active Information Leakage**

# Started with Cache

- High-level intro to cache



Smaller, faster, more $, subset Of blocks

Data is copied between levels in block-sized transfer units

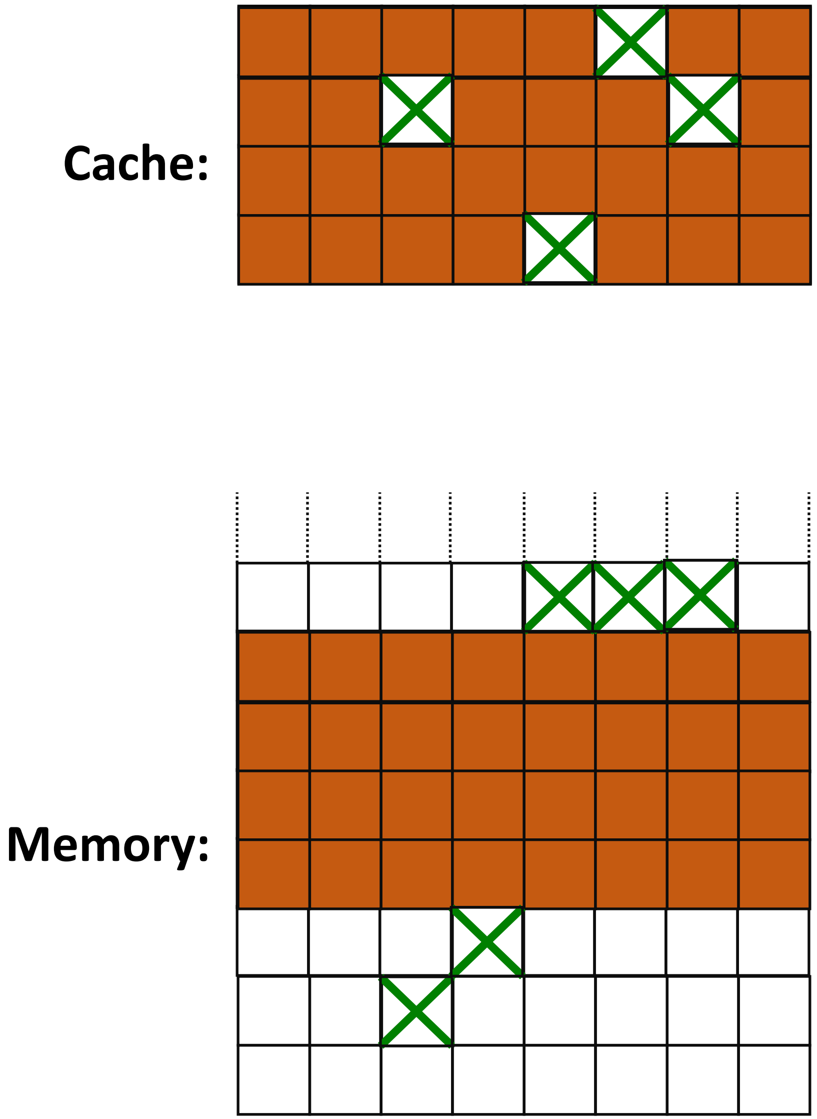
Larger, slower, cheaper memory

- Principles behind caching
  - Temporal locality
    - Re-use of specific data in time
  - Spatial locality
    - Use of data close to each other
    - Example loop:
      - for ( …..) { sum=sum+x[i];}
- When CPU read
  - Cache Hit
    - Fast – lower latency
  - Cache Miss
    - Slow – longer latency

Cache/Cache Hierarchy is a Microarchitecture Technique

# Prime+Probe [Per05, OST06]

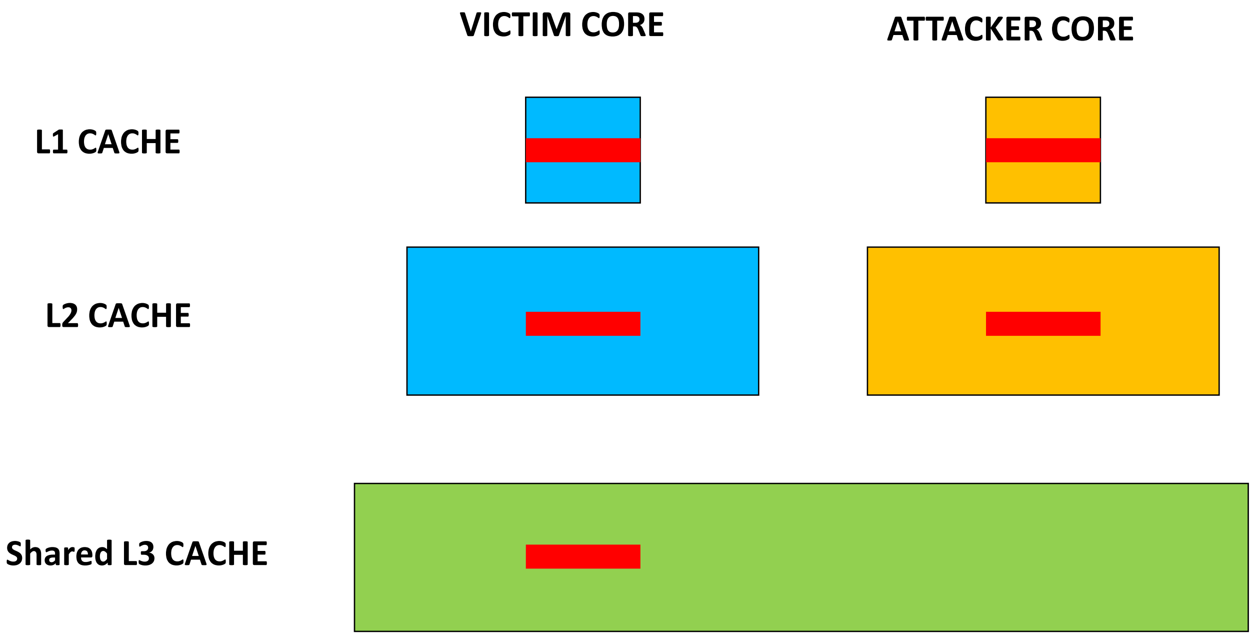
- Attacker chooses a cache-sized memory buffer
- Attacker accesses all the lines in the buffer, filling the cache with its data
- Victim executes, evicting some of the attackers lines from the cache
- Attacker measures the time to access the buffer
  - Accesses to cached lines is faster than to evicted lines
  - Learn victim’s address



[Per05] C. Percival, "Cache Missing for Fun and Profit", BSDCan, 2005
[OST06] D. A. Osvik, A. Shamir and E. Tromer, "Cache Attacks and Countermeasures: The Case of AES", CT-RSA 2006

# Flush+Reload [Y&F14]

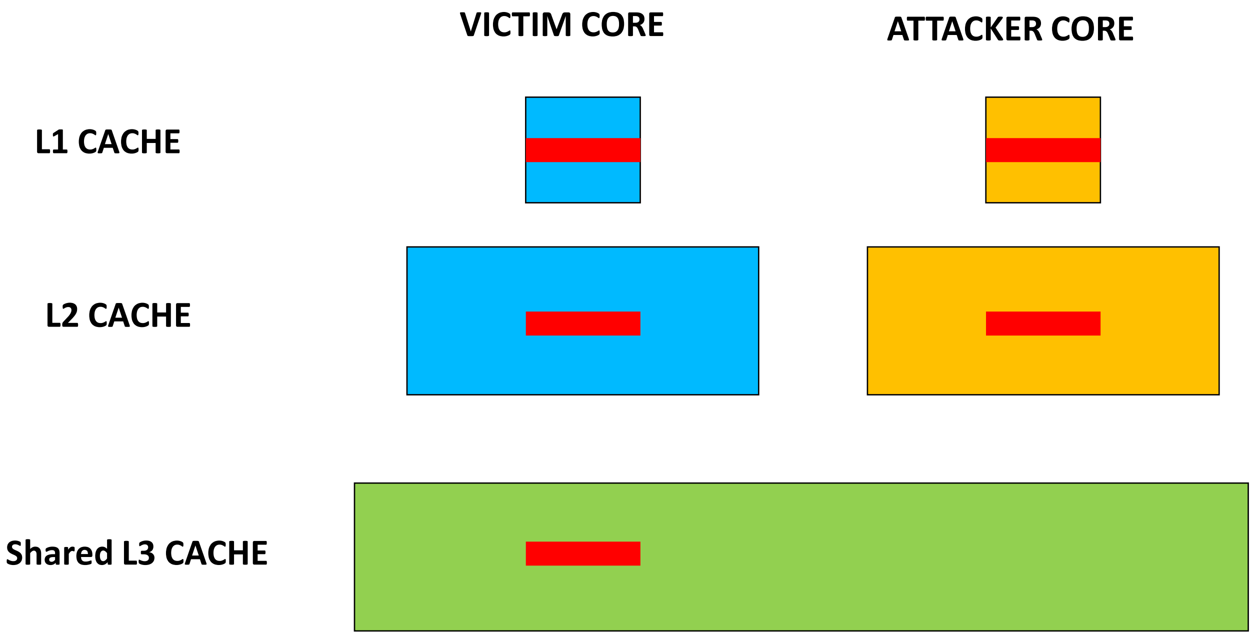
- Work on cache lines (vs. cache sets in Prime+Probe)
- Needs physical memory sharing with victim
- Attacker use "`CLFLUSH`" to evict line
- Due to inclusive property victim private cache line evicted also



Y. Yarom & K. Falkner, "FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack," 23rd USENIX Sec Symp, 2014

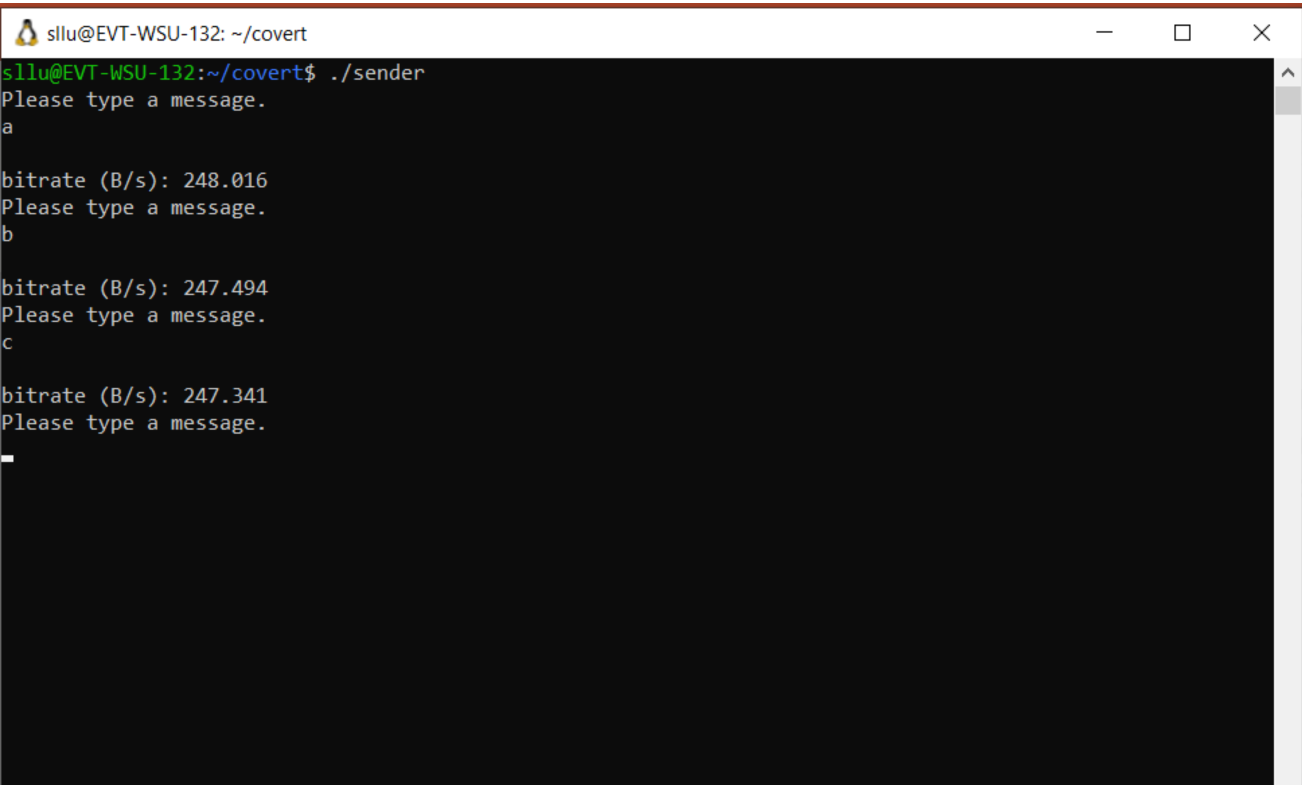
# Flush+Reload (Cont.)

- Attacker waits
- Victim reload the cache line
- Attacker reload and use read time stamp counter to time (rdtsc)
  - If fast then line in L3 (used by victim) else in DRAM (not used by victim)

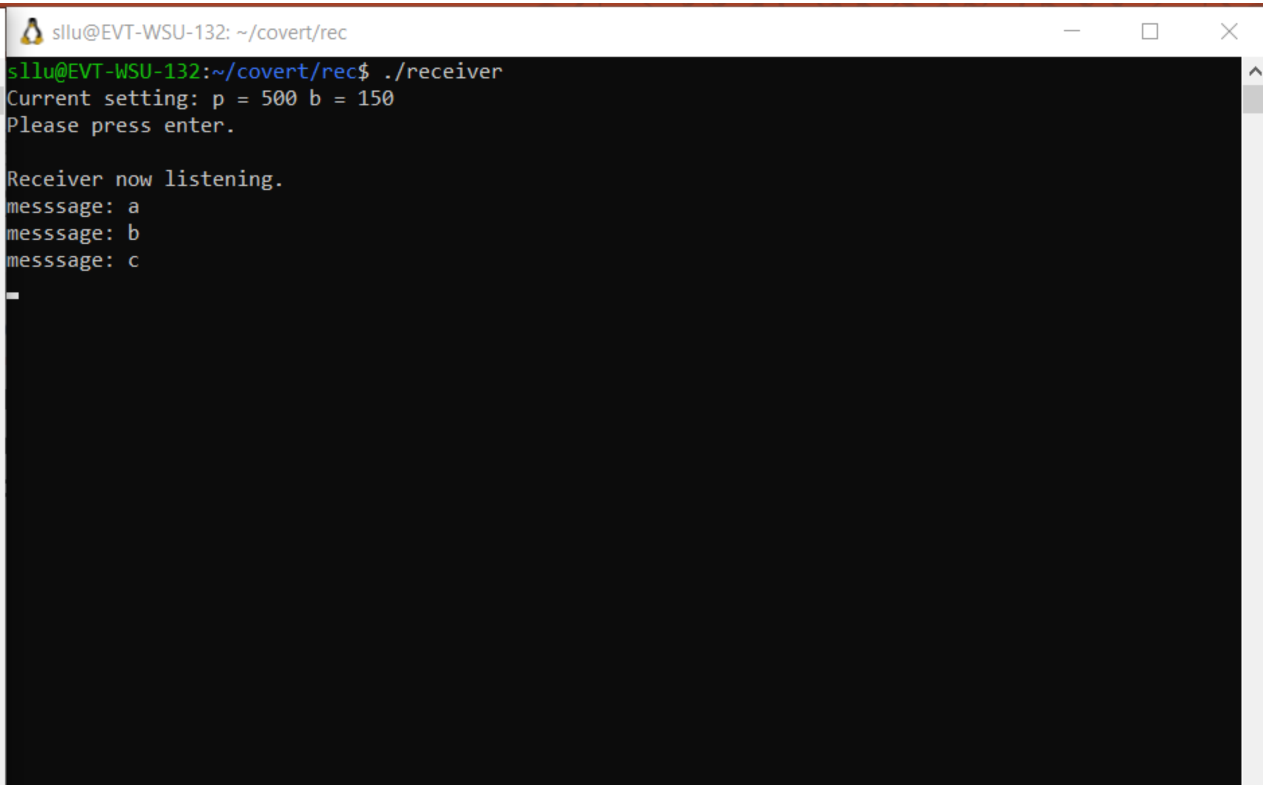


# A Demo of Cache Covert Channel

- What can threat actor do with cache side channels?
  - Cryptographic key leakage
  - A simple way to establish a covert channel to leak data



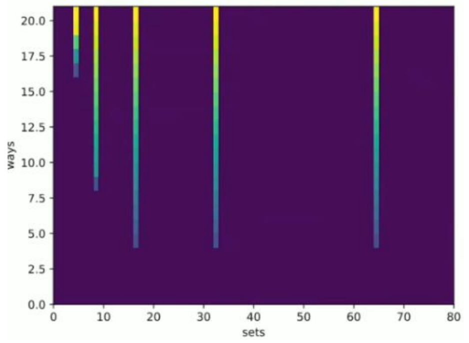
```
sllu@EVT-WSU-132:~/covert$ ./sender
Please type a message.
a

bitrate (B/s): 248.016
Please type a message.
b

bitrate (B/s): 247.494
Please type a message.
c

bitrate (B/s): 247.341
Please type a message.
```

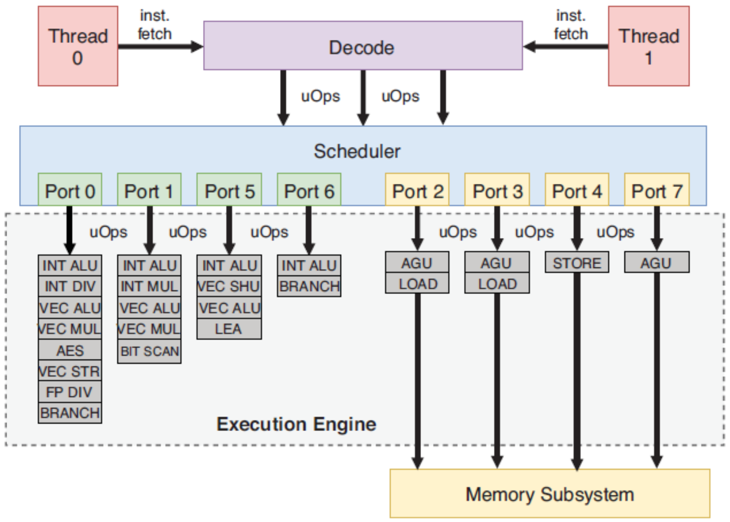


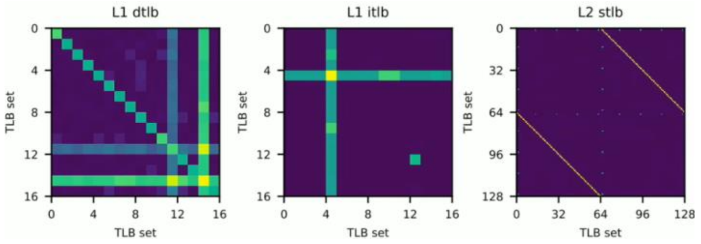
```
sllu@EVT-WSU-132:~/covert/rec$ ./receiver
Current setting: p = 500 b = 150
Please press enter.

Receiver now listening.
messsage: a
messsage: b
messsage: c
```

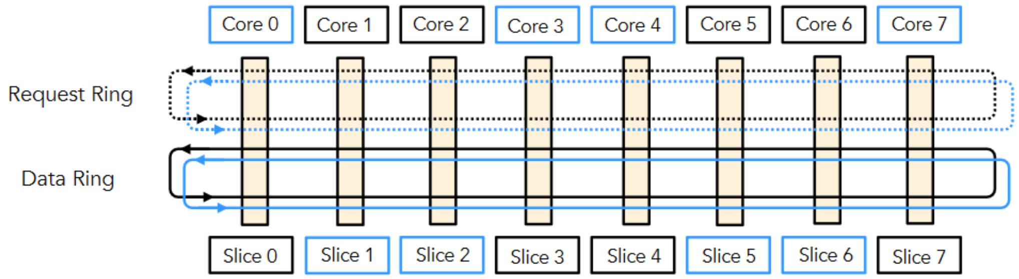
# Other On-Chip Shared Structures

- TLBs [Gra 18]
  - L1iTLB, L1dTLB, L2TLB contention
  - Reverse engineered by experimenting with page walks and perf cntr
  - Access set of L1dTLB and observe sharing between two threads
  - Observe EdDSA ECC key multiplication
- Functional Units [Ald 19]
  - Port contention of SMT
- On-chip networks [PLF 20]
  - Ring contention
    - Overlapped segments





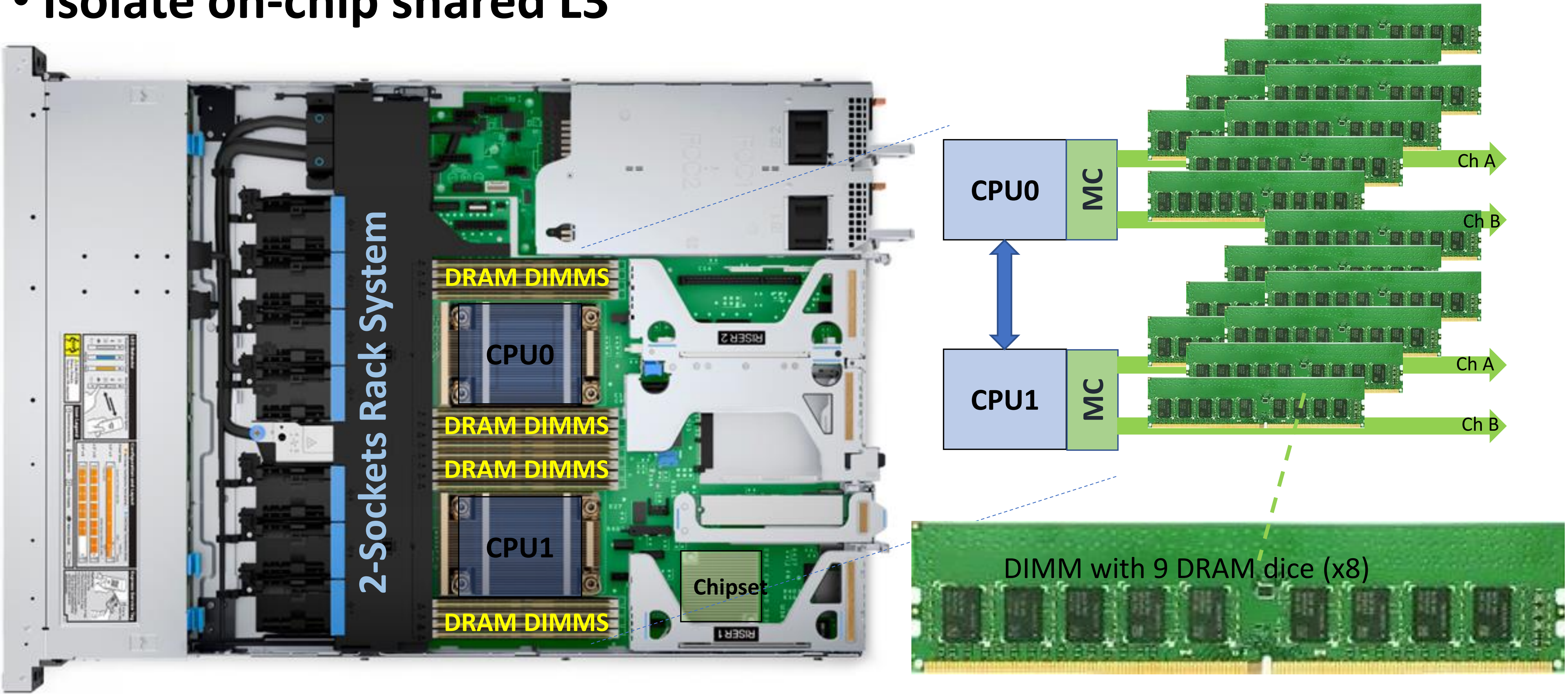




Translation Leak-aside Buffer: Defeating Cache Side-channel Protections with TLB Attacks, 27th USENIX Security Symp. 2018
Port Contention for Fun and Profit. In 2019 IEEE Symposium on Security and Privacy (SP). 2019
Lord of the Ring(s): Side Channel Attacks on the CPU On-Chip Ring Interconnect Are Practical, 30th USENIX Security Symp. 2021

# DRAM in a Computer System

- Non-inclusive L3
- Isolate on-chip shared L3

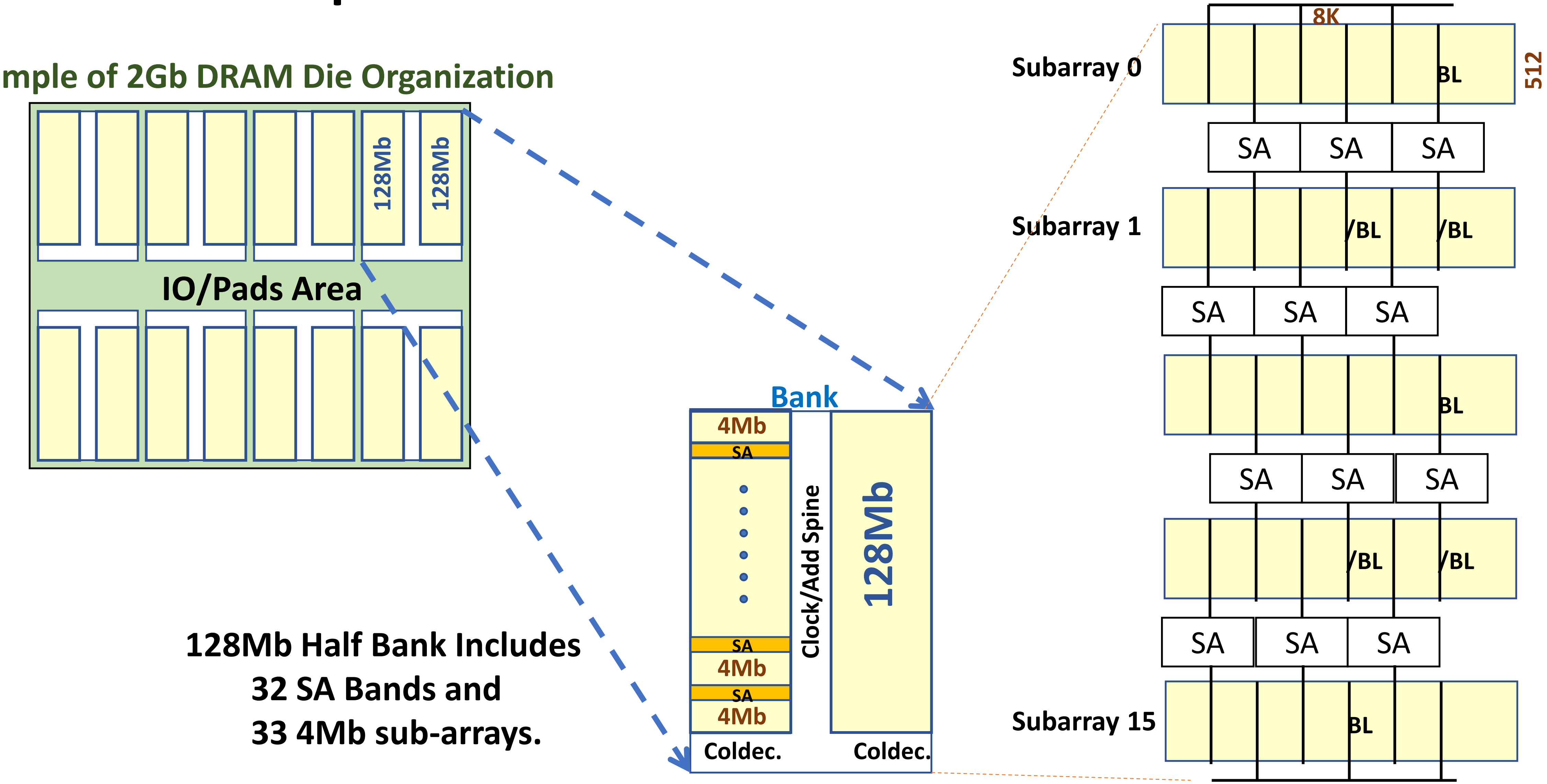


# DRAM Die Internal

- **Banks are independent**

**Example of 2Gb DRAM Die Organization**

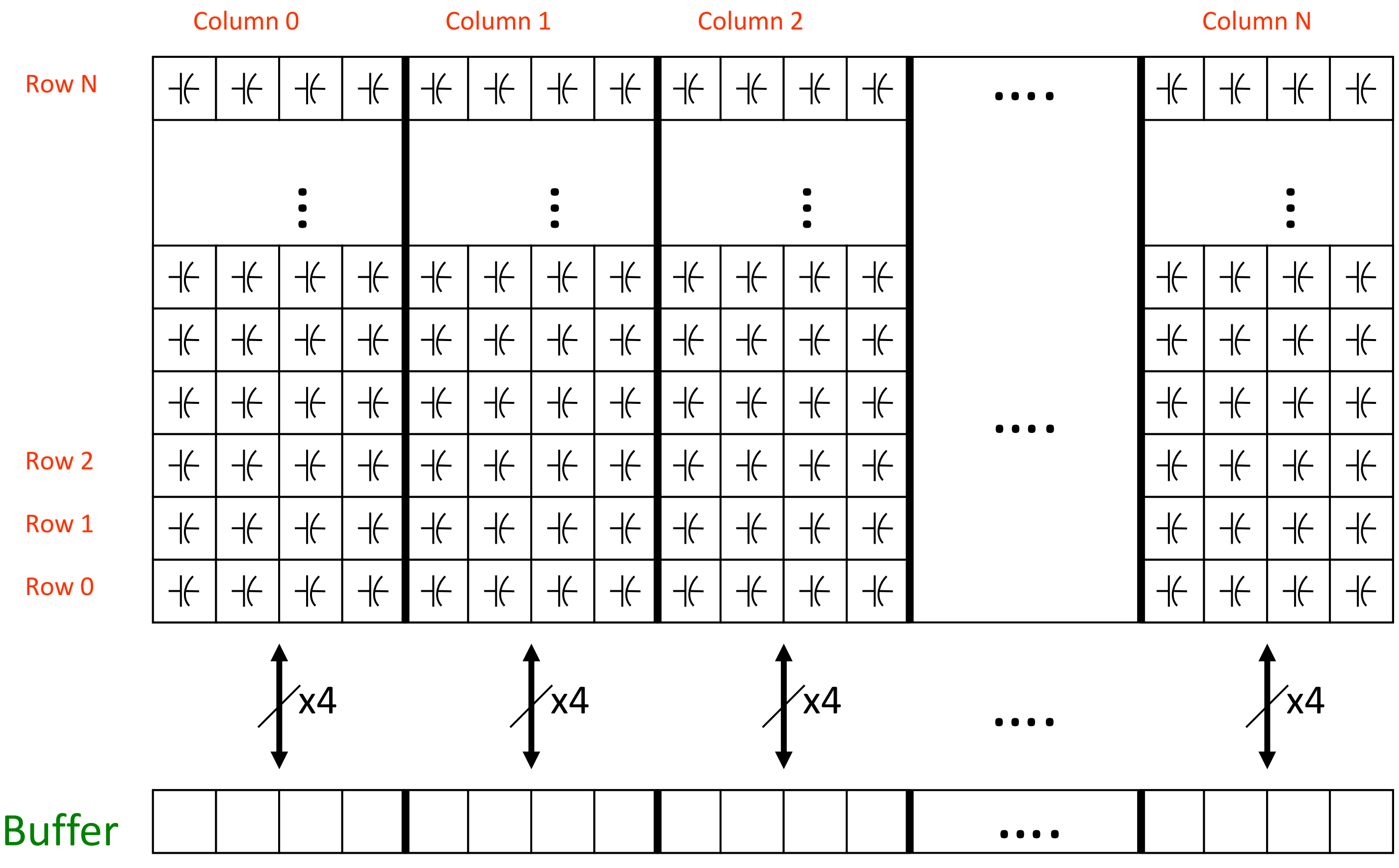
**OPEN BL in Commodity DRAM**



**128Mb Half Bank Includes
32 SA Bands and
33 4Mb sub-arrays.**

# DRAM Internal Abstraction – Page of 8K

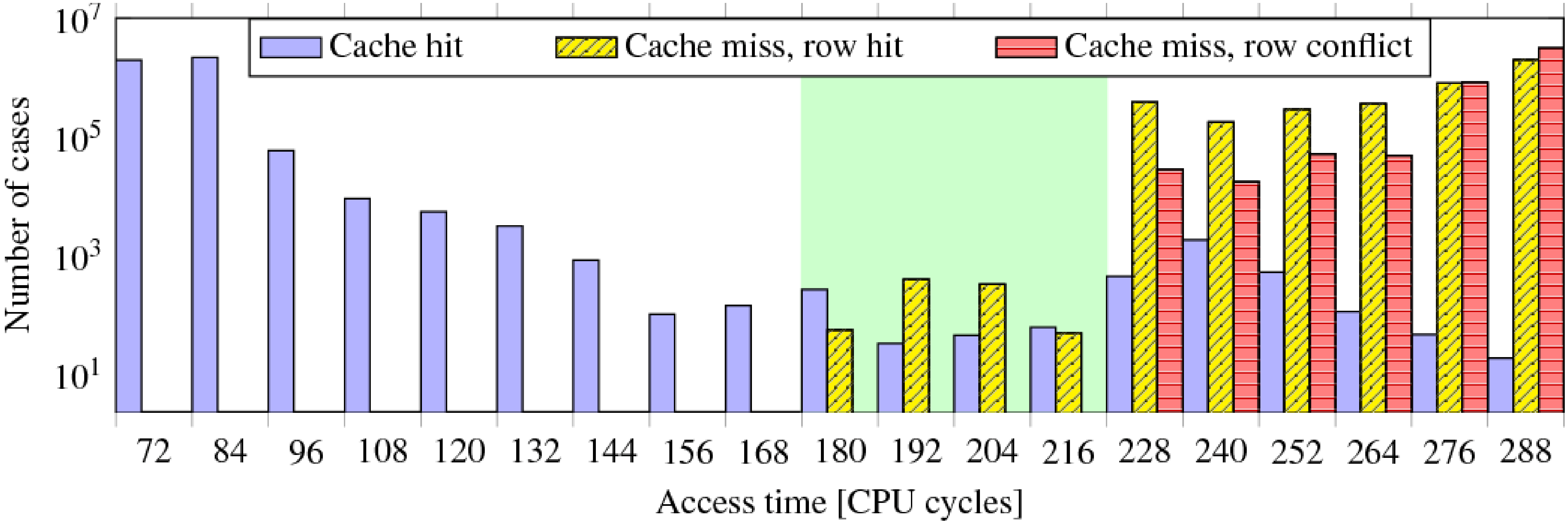
- ACT – activate a row/page (tRAS)
- Read a portion of bits out (burst)
- Page policy
  - Close page
  - Open page
    - Locality
- Access to same bank slow
- Access to different banks
  - faster



Page Size = number of bits

# Access Time Disparity Due to Row Buffer [PGM16]

- **Behavior of memory access time**
  - **Row hits – lower latency (180-216 cycles no row conflicts)**
  - **Row miss/conflicts – higher latency**



P. Pessl et. al, “DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks,” 25th USENIX Sec Symp, 2016

# Vulnerability in Transient State

- Speculative execution
  - Common performance improvement technology in modern processors
    - Branch prediction
    - Speculatively load
  - Software security requirement
    - CPUs runs instructions correctly
    - Discard mistakes and re-run
  - “Private” data are in the HW speculatively (will get flushed eventually)
    - Observable
    - Leaking private information
- Famous examples
  - Meltdown and Spectre [Koch19]
  - Google project zero

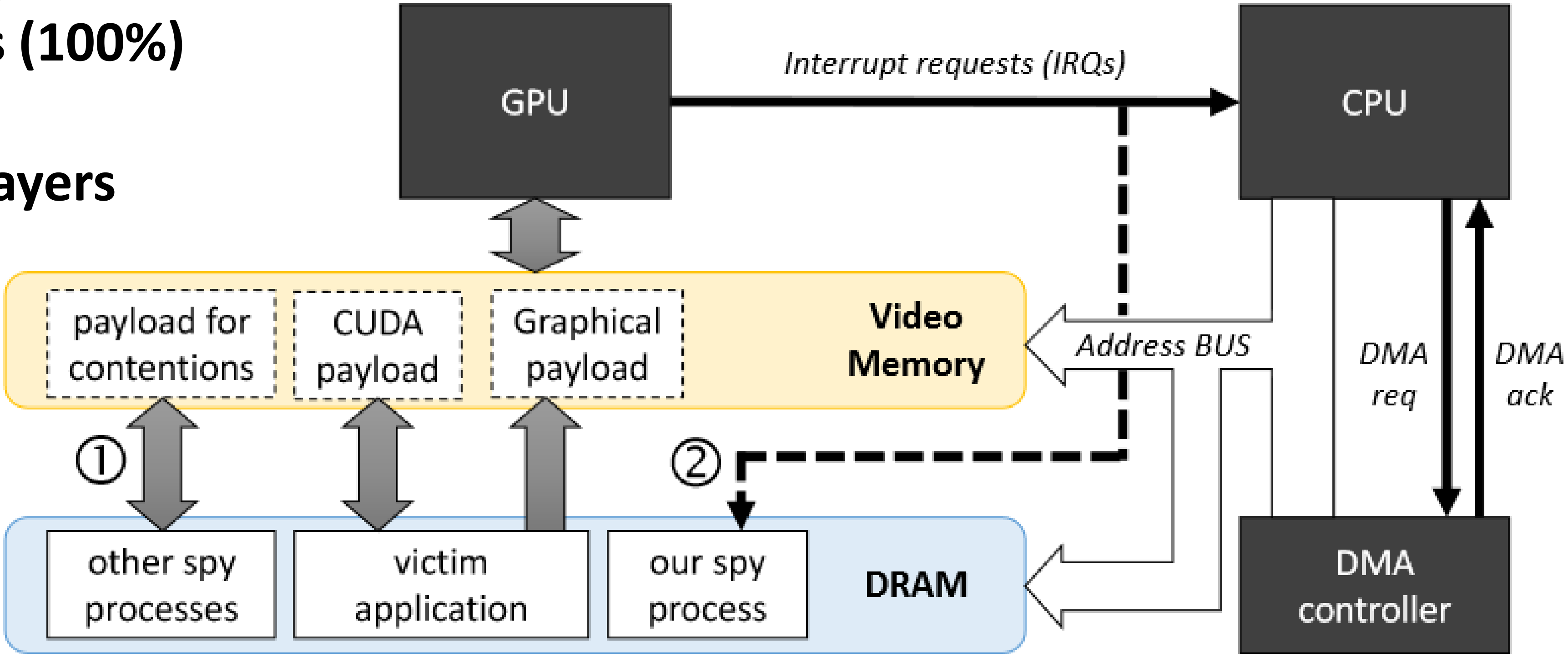
https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html

https://nsarchive.gwu.edu/sites/default/files/documents/4345434/Meltdown-Attack-Spectre-2017-Unclassified-spectre.pdf

https://meltdownattack.com/

# Interrupt Requests Side-Channels [Ma 2015]

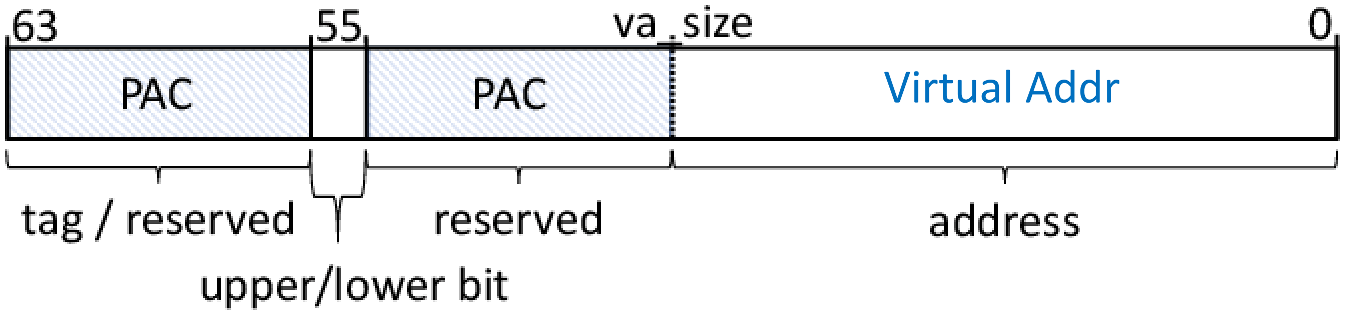
- No active contented process - ①
- Passive reading aggregated graphics interrupt counts (OS) - ②
  - Infer from statistics collected to predict GPU tasks (ML)
  - Events (prediction rate)
    - GUI apps (99.95%)
    - GPGPU workloads (100%)
    - Webpage visits
    - Different video players
    - PDF documents

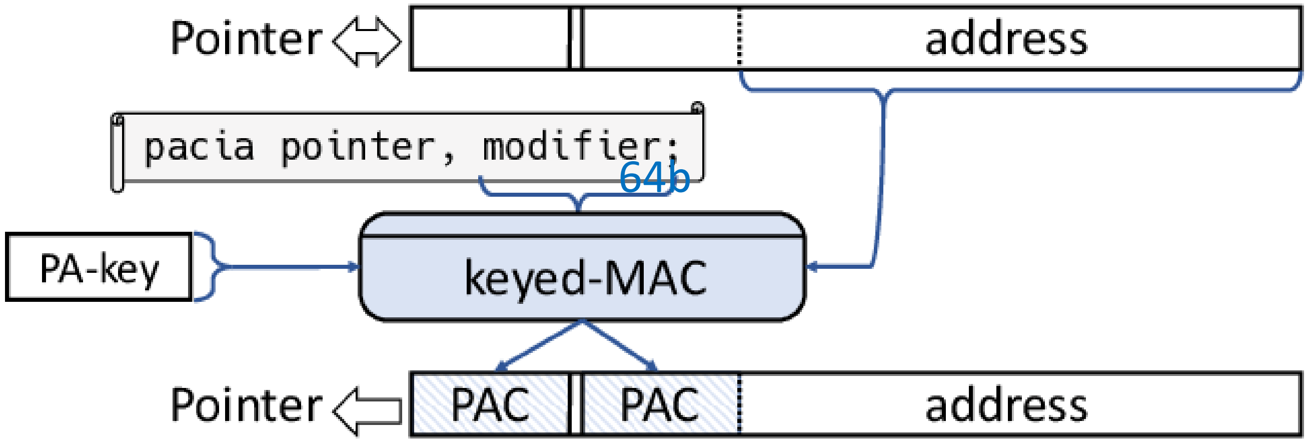


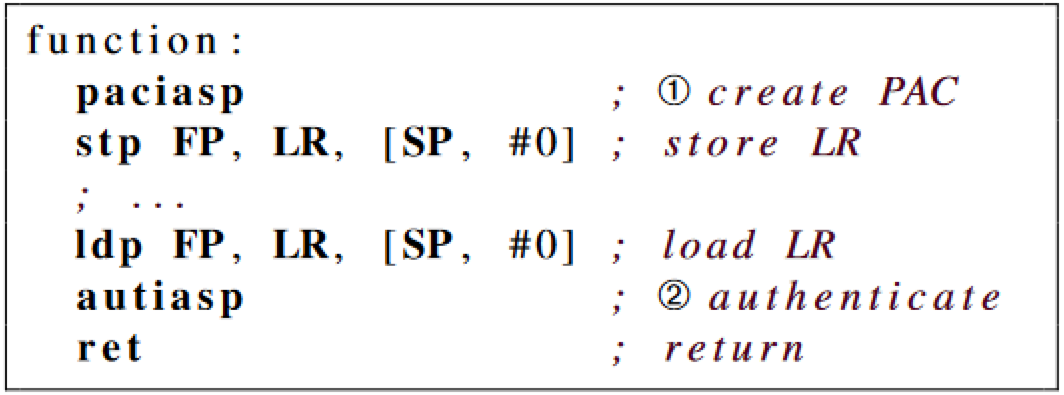
On the Effectiveness of Using Graphics Interrupt as a Side Channel for User Behavior Snooping, IEEE T. on Dep & Sec computing, v. 19, 2022
https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=7752&context=sis_research

# Security Enhancement Structure - PAC Vulnerability

- What is PAC (Pointer Authentication Codes)?
  - A “security” feature from ARM to guard against pointer integrity
    - ARMv8.3-A
    - Unused address bits
    - Dynamically instrumented
      - Add PAC (pacia__)
      - Authenticate (autia__)
  - Usage example – func call
    - Entry – create a PAC in LR
    - Exit – check LR





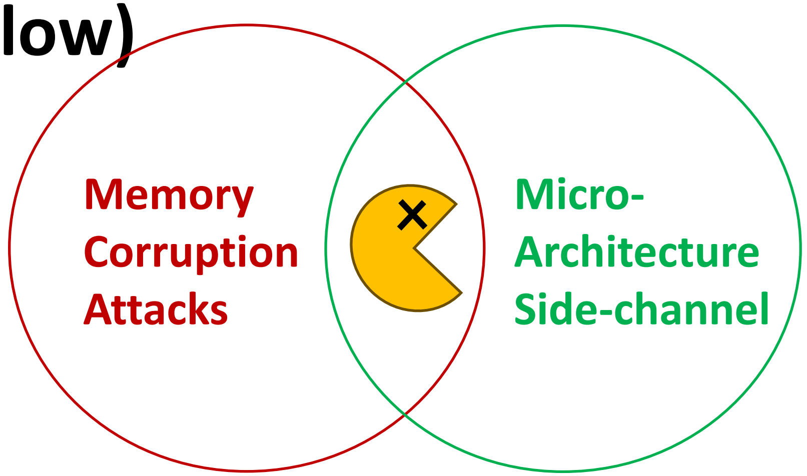


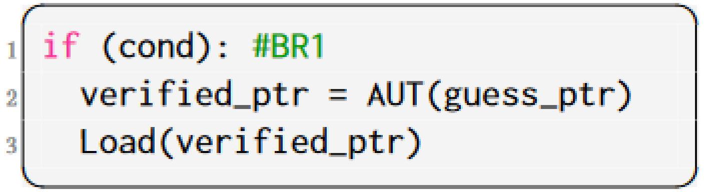
```
function:
  paciasp                    ; ① create PAC
  stp FP, LR, [SP, #0]       ; store LR
  ; ...
  ldp FP, LR, [SP, #0]       ; load LR
  autiasp                    ; ② authenticate
  ret                        ; return
```

PAC it up: Towards Pointer Integrity using ARM Pointer Authentication, USENIX Security 2019

# Apple M1 PACMAN Attack [Rav 22]

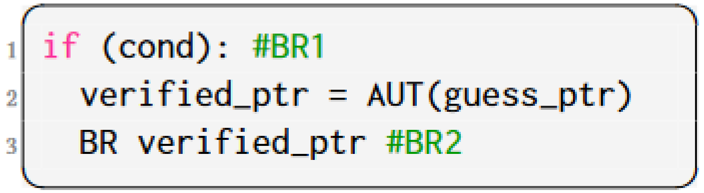
- Leveraging microarchitecture side-channel & memory corruption
  - PAC protects illegal memory access (buffer overflow)
    - Verify the signature (hash)
    - If different system halts
  - But … there is the speculation state!
    - Guess the hash in speculation state





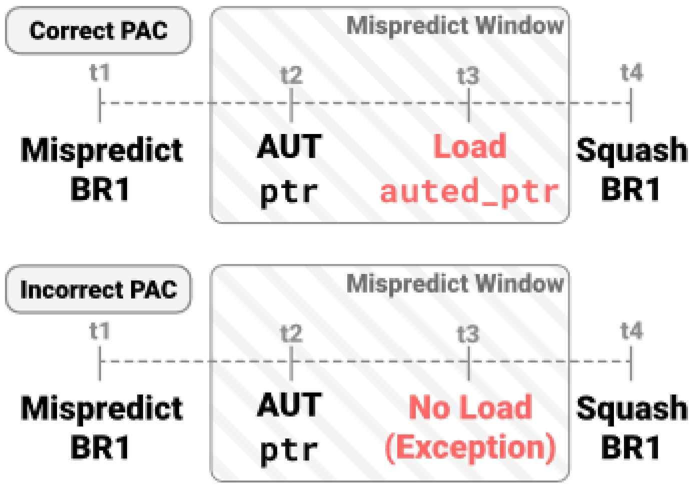
```
if (cond): #BR1
  verified_ptr = AUT(guess_ptr)
  Load(verified_ptr)
```

(a) A data PACMAN gadget.

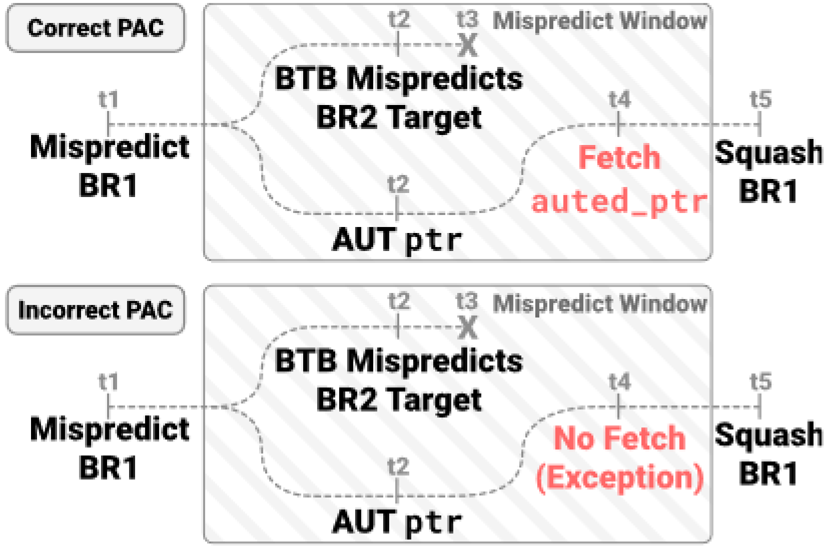


```
if (cond): #BR1
  verified_ptr = AUT(guess_ptr)
  BR verified_ptr #BR2
```

(b) An instruction PACMAN gadget.



(c) Timeline for data access leak.



(d) Timeline for instruction fetch leak.

PACMAN: Attacking ARM Pointer Authentication with Speculative Execution, ISCA 2022

# Outline

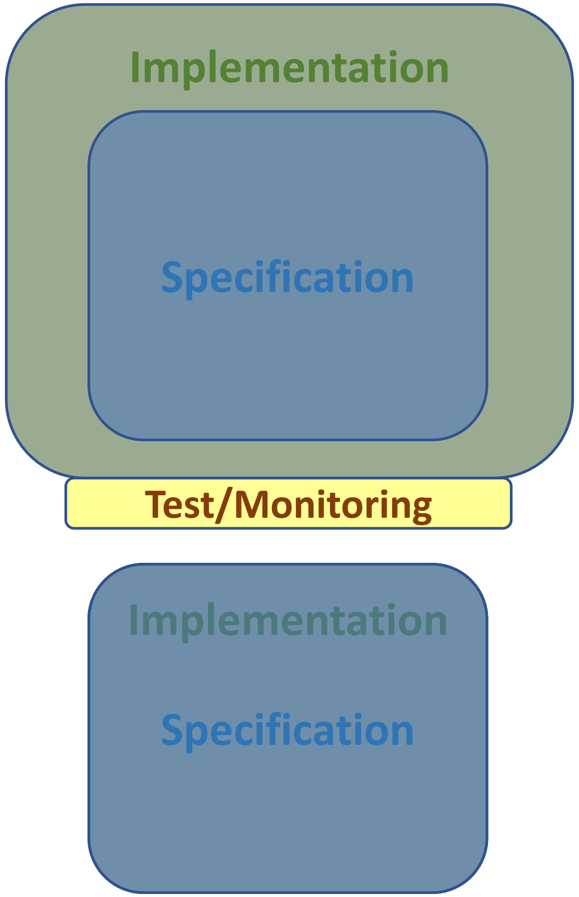
- Security goals - CIA
- Hardware stack (vs. software stack)
  - Technology/Transistors
  - Packaging/Board/System
    - Firmware
    - Software
  - Design process
    - Circuits, logic, RTL, architecture
  - Supply chain
- Security goals in the hardware context – CIA
  - Understand hardware attack surfaces
    - Threat
    - Vulnerability
      - Microarchitecture side-channels
  - Attack methods
  - **Mitigation techniques**

# Mitigation Starts with Understanding

- Classification
  - Side-channels enabled by
    - Shared structures
    - Speculation
    - Performance counters
  - Channel types
    - Persistent
      - States that persist until next events
    - Ephemeral
      - Transient states – attacker/victim must cohabit
  - Attack types
    - Active
      - Affects victim
    - Passive

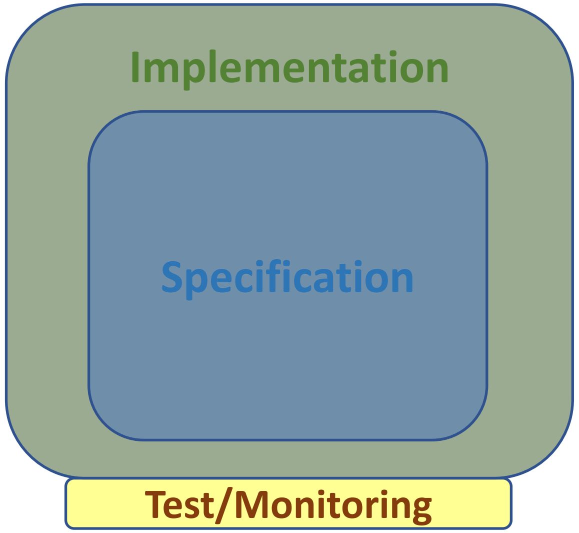
# Mitigation

- Complete clean slate design
  - Compatibility challenge
  - Cost – formal verification?
- Mitigation
  - Partition
    - Strick isolation
    - No shared structures
      - Performance impact
  - Obfuscation
    - Add noise



# Possible Obfuscation Approaches

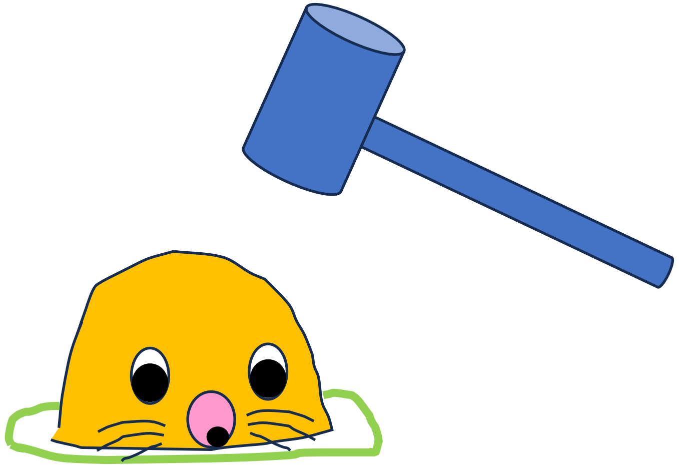
- Static analysis/data analytic
  - Given an implementation
  - Formalize all potential modulation
    - To mitigate one must find the modulation
    - Aid in reverse engineering
  - Build graph - evaluate
- Dynamic monitoring
  - Detection of active leakage
    - Patterns?
    - Does it need performance counters?
- Dynamic randomization - obfuscation
  - Source side
    - Intelligent fuzzing of software
  - System side
    - Insertion of disturbance
      - Maximize effect of noise while minimize effects on performance



https://web.eecs.umich.edu/~barisk/public/morpheus.pdf

# Summary

- Security is important when there is no trust
- Security is challenging
- Microarchitecture side channel is particularly dangerous
  - No need for physical access
  - Against design goals
  - Increasing attack surfaces
- Possible approach
  - Static
  - Dynamic
  - Need efficient and general solution



# Q & A