Micro-Architectural Attacks on Cyber-Physical Systems

07/24/2019

Heechul Yun

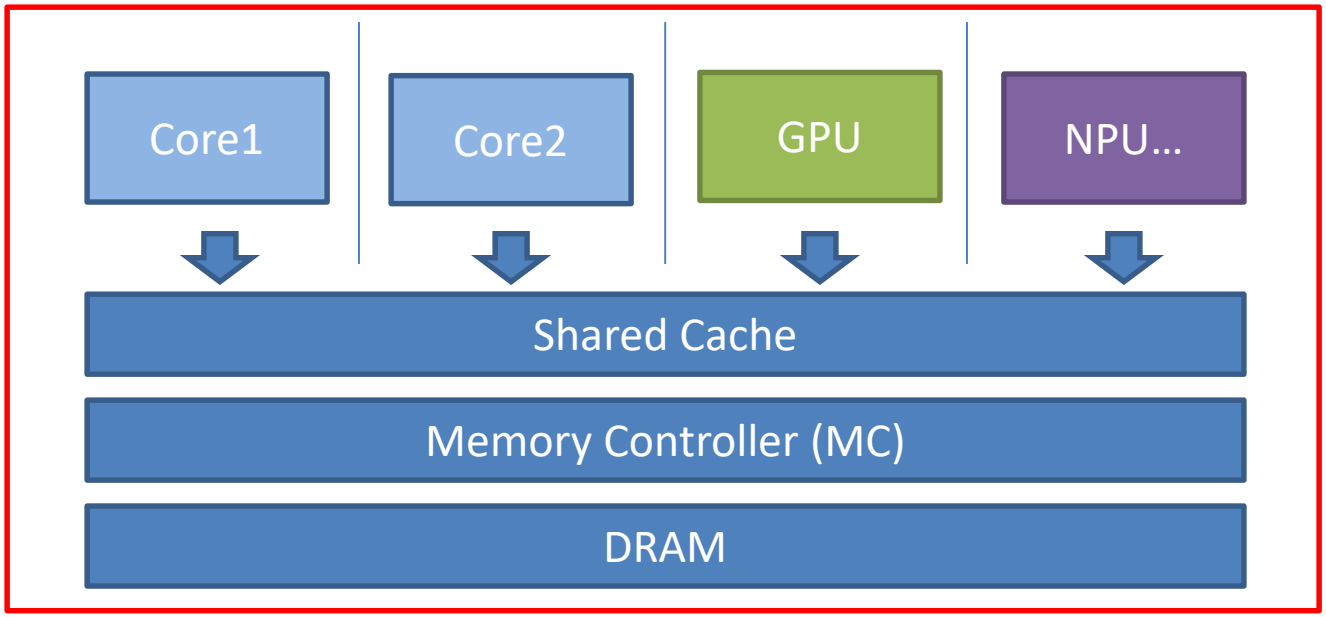
University of Kansas

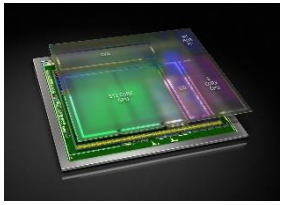
KU THE UNIVERSITY OF KANSAS

Modern Cyber-Physical Systems

- Cyber Physical Systems (CPS)
  - Cyber (Computer) + Physical (Plant)
- Real-time
  - Control physical process in real-time
- Safety-critical
  - Can harm people/things
- **Intelligent**
  - Can function autonomously

Modern System-on-a-Chip (SoC)





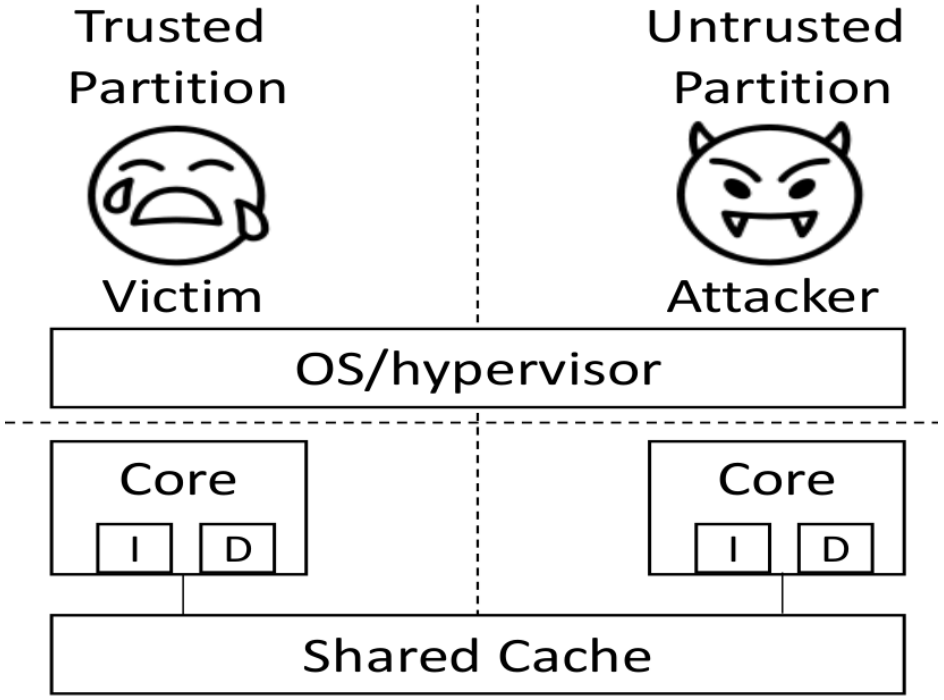
- Integrate multiple cores, GPU, accelerators
- Good performance, size, weight, power
- **Introduce new challenges in real-time, security**

Micro-Architectural Attacks

- Micro-architectural hardware components
  - E.g., cache, tlb, DRAM, OoO engine, MSHRs, …
- **Can affect execution timing**
  - E.g., delay critical real-time tasks
- **Can leak secret**
  - E.g., Meltdown, Spectre
- **Can alter data**
  - E.g., RowHammer

KU THE UNIVERSITY OF KANSAS

1. Denial-of-Service Attacks



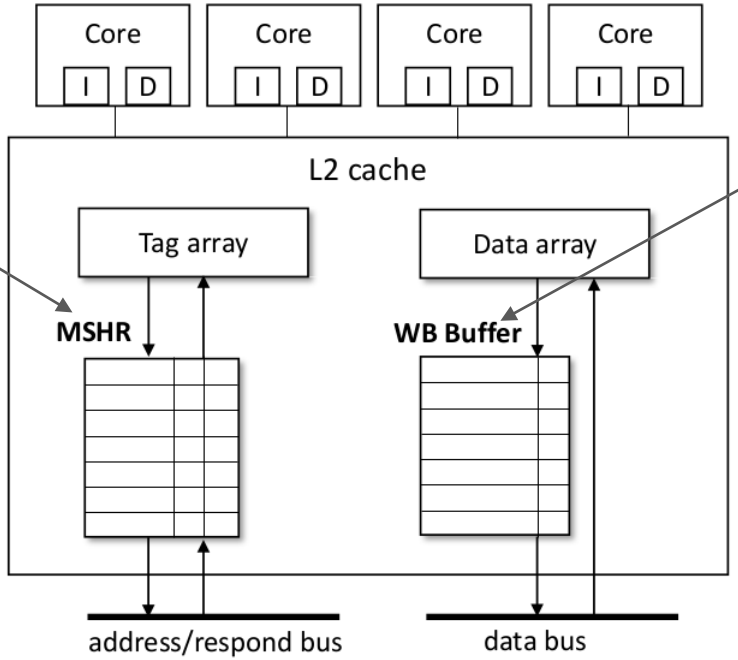
- Attacker's goal: increase the victim's task **execution time**
- The attacker is on different core/memory/cache partition
- The attacker can only execute non-privileged code.

M. G. Bechtel and H. Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *RTAS*, 2019

Non-Blocking Cache

Miss Status Holding Registers[1]

- Track outstanding cache misses.



Writeback Buffer[2]

- Holds evicted dirty lines (writebacks).
- Prevents cache refills from waiting.

- We identified cache internal structures that are potential DoS attack vectors

[1] P. K. Valsan, H. Yun, F. Farshchi. "Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems." In *RTAS*, 2016
[2] M. G. Bechtel and H. Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *RTAS*, 2019

Cache DoS Attacks

```
for (i = 0; i < mem_size; i += LINE_SIZE)
{
        sum += ptr[i];
}
```

Read Attacker
(target **MSHRs**)

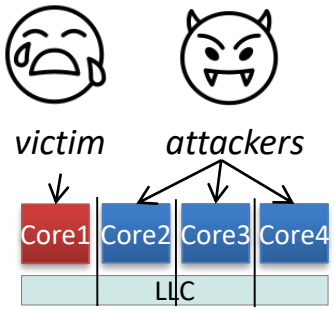
```
for (i = 0; i < mem_size; i += LINE_SIZE)
{
        ptr[i] = 0xff;
}
```

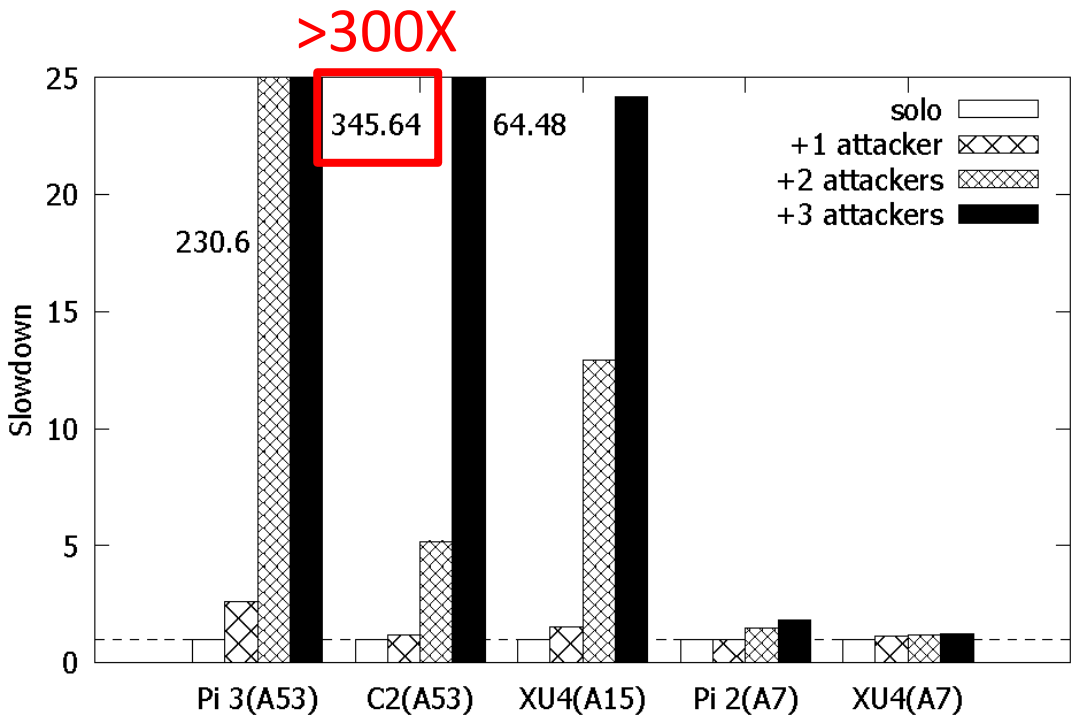
Write Attacker
(target **WBBuffer**)

- Denial-of-Service (DoS) attacks targeting internal hardware structures of a shared cache.
  - Block the cache → delay the victim's execution time

M. G. Bechtel and H. Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *RTAS*, 2019

Effects of Cache DoS Attacks





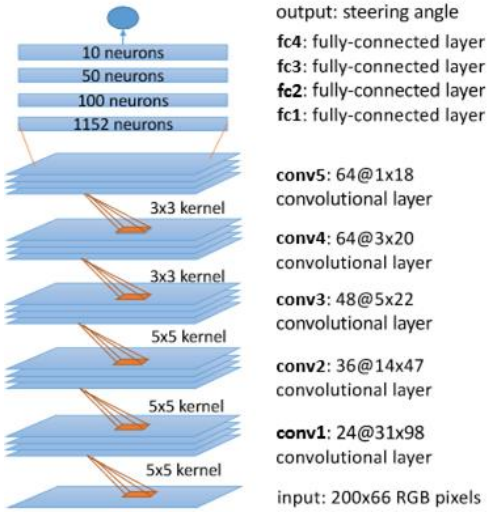
- Observed worst-case: >300X (times) slowdown
  - On popular in-order multicore processors
  - Due to contention in cache write-back buffer

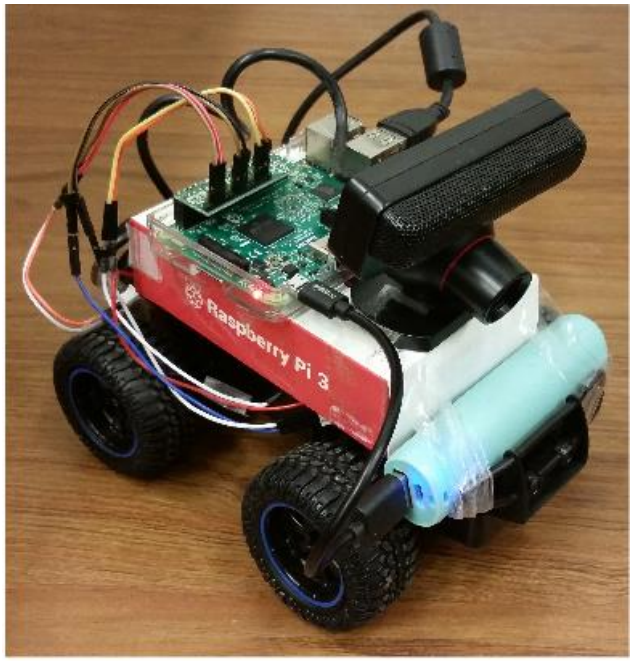
M. G. Bechtel and H. Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *RTAS*, 2019

DeepPicar

- A **low cost**, small scale replication of NVIDIA's DAVE-2
- **Uses the exact same DNN**
- Runs on a Raspberry Pi 3 in **real-time**

Item	Cost ($)
Raspberry Pi 3 Model B	35
New Bright 1:24 scale RC car	10
Playstation Eye camera	7
Pololu DRV8835 motor hat	8
External battery pack & misc.	10
Total	70

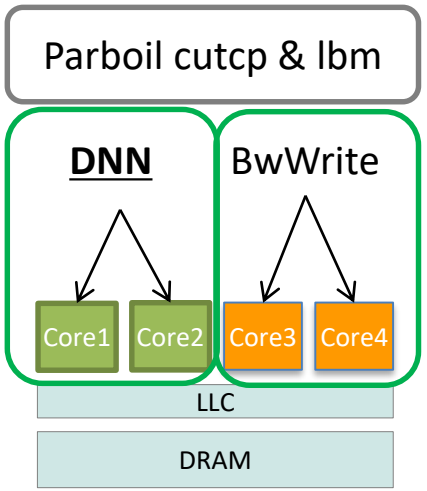




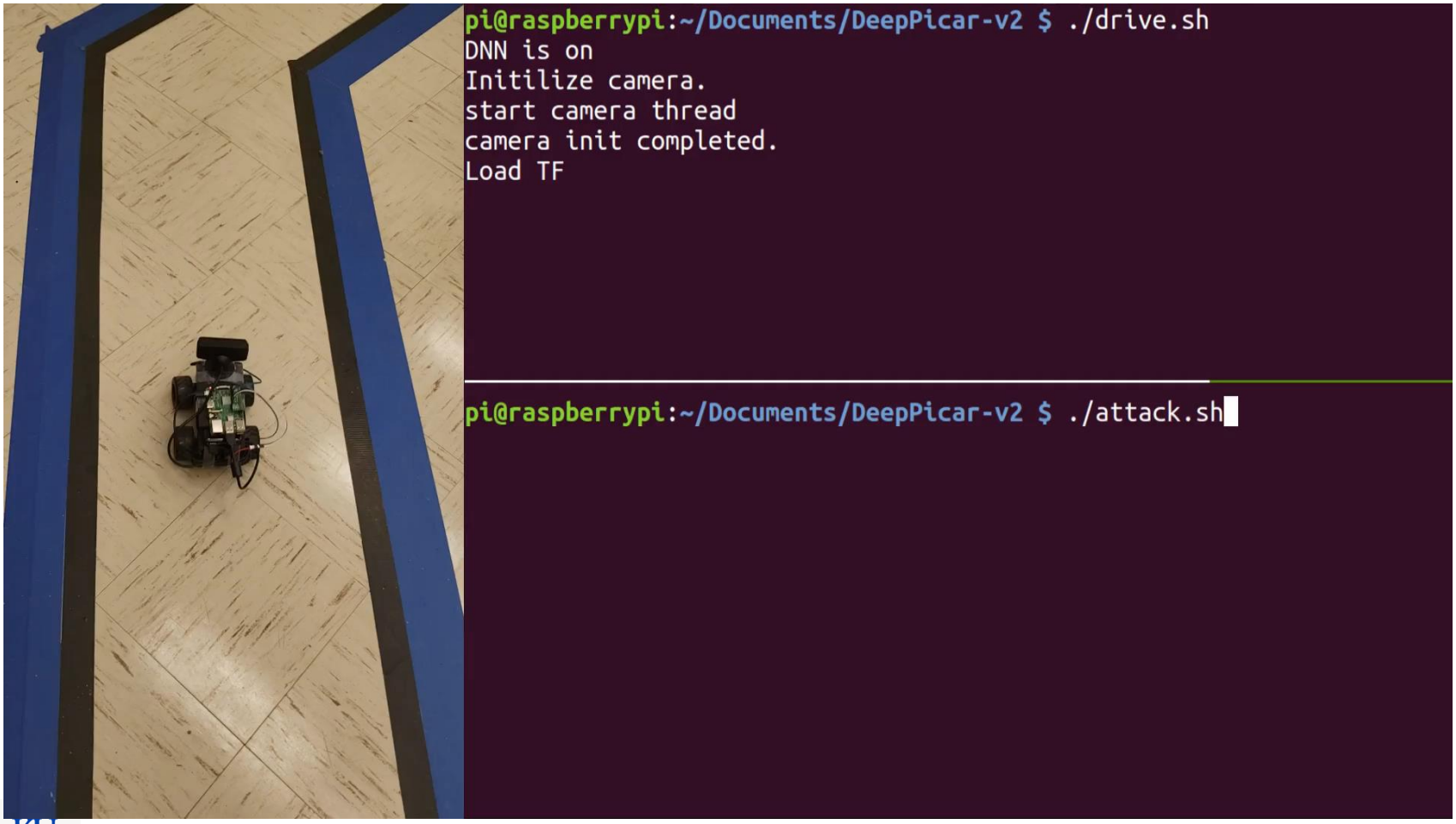
Experiment Setup

- DNN control task of DeepPicar (real-world RT)
- IsolBench BwWrite benchmark (synthetic RT)
- Parboil benchmarks (real-world BE)

	Task	WCET (C ms)	Period (P ms)	# Threads
RT	t^{rt}_{dnn}	**34**	**100**	**2**
	t^{rt}_{bww}	**220**	**340**	**2**
BE	t^{be}_{cutcp}	∞	N/A	4
	t^{be}_{lbm}	∞	N/A	4



Effect of Co-Scheduling



https://youtu.be/Jm6KSDqlqiU

2. Speculative Execution Attacks

- Attacks exploiting microarchitectural side-effects of executing speculative (transient) instructions
- Many variants

Attack	Description
Variant 1 (Spectre) [16]	Bounds Check Bypass
Variant 1.1 [15]	Bounds Check Bypass Store
Variant 1.2 [15]	Read-only Protection Bypass
Variant 2 (Spectre) [16]	Branch Target Injection
Variant 3 (Meltdown) [18]	Supervisor Protection Bypass
Variant 3a [12]	System Register Bypass
Lazy FP [24]	FPU Register Bypass
Variant 4 [9]	Speculative Store Bypass
ret2spec [20]	Return Stack Buffer
L1 Terminal Fault [11, 26]	Virtual Translation Bypass

No hardware support planned in near future

P. Kocher et al., "Spectre attacks: Exploiting speculative execution," In *IEEE S&P*, 2019. (originally published in arXiv archive in Jan. 2018)

Spectre Attack (Variant 1)

```
if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}
........
```

- Assume `x` is under the attacker's control
- Attacker trains the branch predictor to predict the branch is in-bound

P. Kocher et al., "Spectre attacks: Exploiting speculative execution," In *IEEE S&P*, 2019.

Spectre Attack (Variant 1)

```
if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}
.........
```

1. [ACCESS]

- Speculative execution of the first line **accesses** the secret (`val`)

P. Kocher et al., "Spectre attacks: Exploiting speculative execution," In *IEEE S&P*, 2019.

Spectre Attack (Variant 1)

```
if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}
.........
```

2. [TRANSMIT]

- Speculative execution of the second, secret dependent load **transmits** the secret to a *microarchitectural state (e.g., cache)*

P. Kocher et al., "Spectre attacks: Exploiting speculative execution," In *IEEE S&P*, 2019.

Spectre Attack (Variant 1)

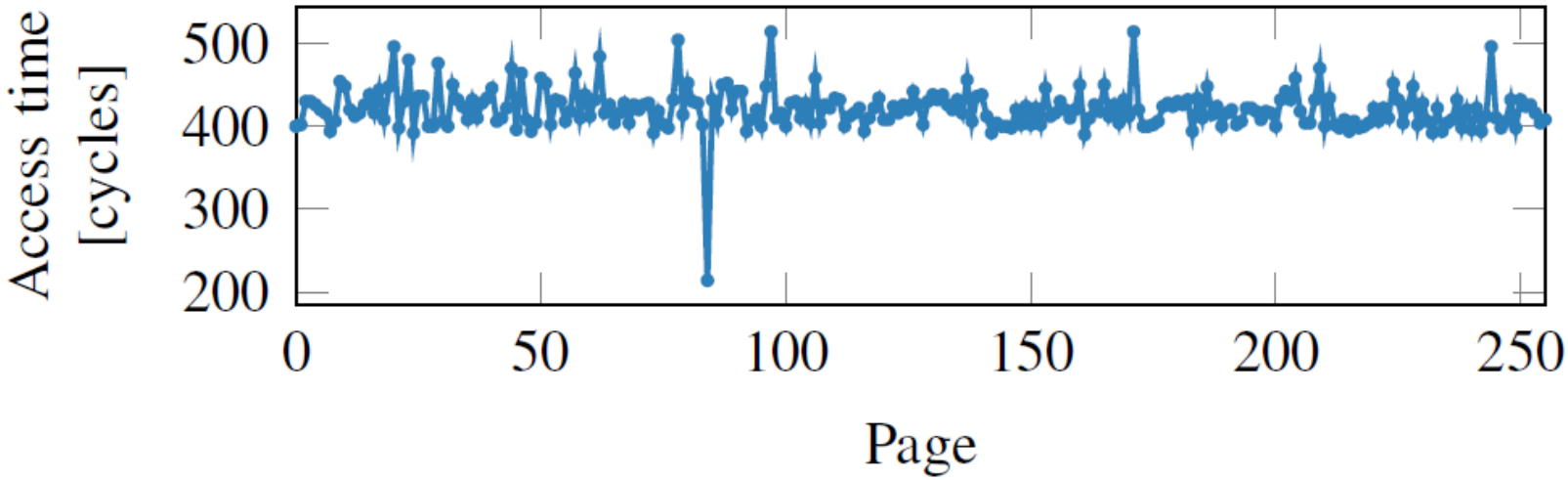
```
if(x < array1_length){
    val = array1[x];
    tmp = array2[val*512];
}
........
```

3. [RECEIVE]

- Attacker **receives** the secret by measuring *timing differences* (cache hit vs. miss) among the elements in the probe array

P. Kocher et al., "Spectre attacks: Exploiting speculative execution," In *IEEE S&P*, 2019.

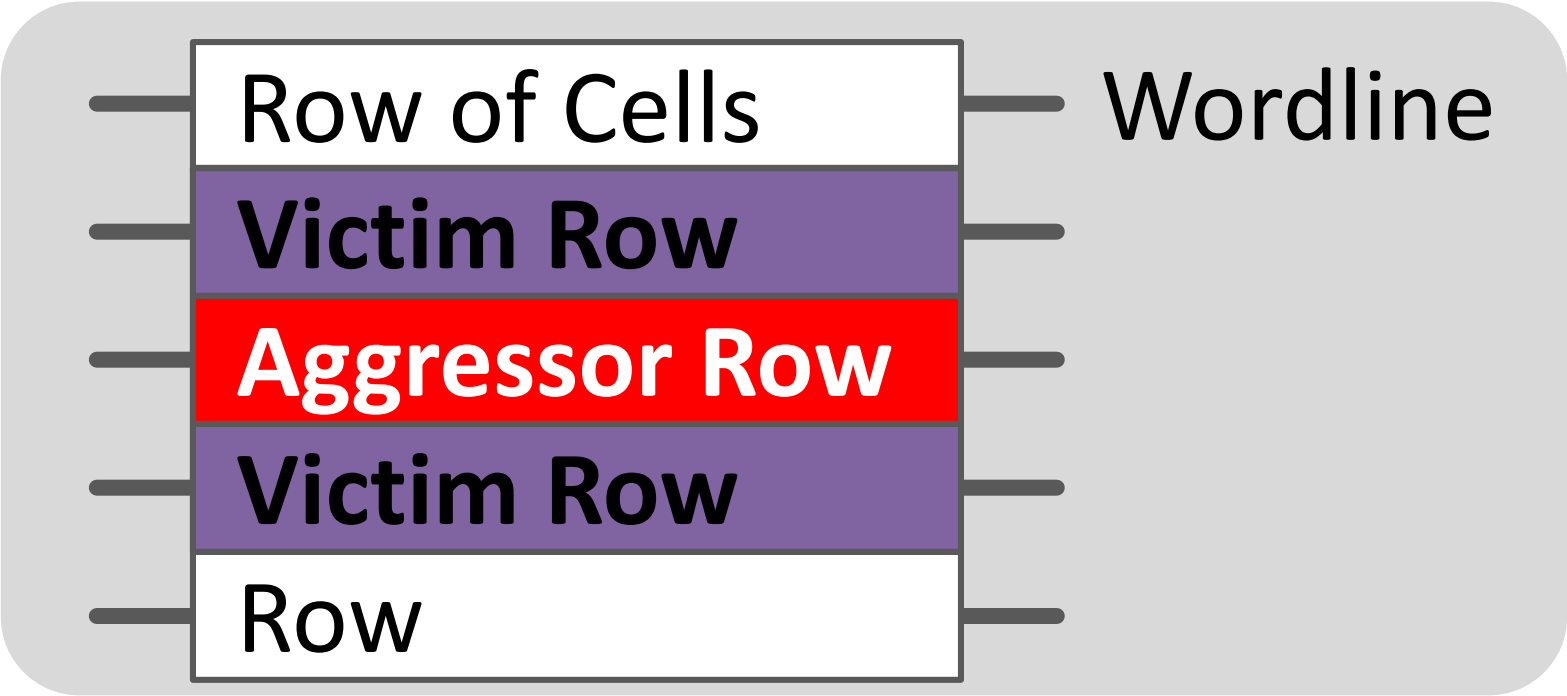
Cache Timing Channels



- Leak secret via timing differences
  - Fast (cache-hit): victim accessed it
  - Slow (cache-miss): victim didn't access it.
- Methods: Flush+Reload, Prime+Probe, etc.

Image source: M. Lipp et al., "Meltdown," In *USENIX Security*., 2018.

3. RowHammer Attacks



- **R**epeatedly opening and closing a DRAM row can induces **bit flips** in adjacent rows storing sensitive data (e.g., page table)

Credit: This slide is from Dr. Yoongu Kim's presentation slides of the following paper:
"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," In *ISCA*, 2014

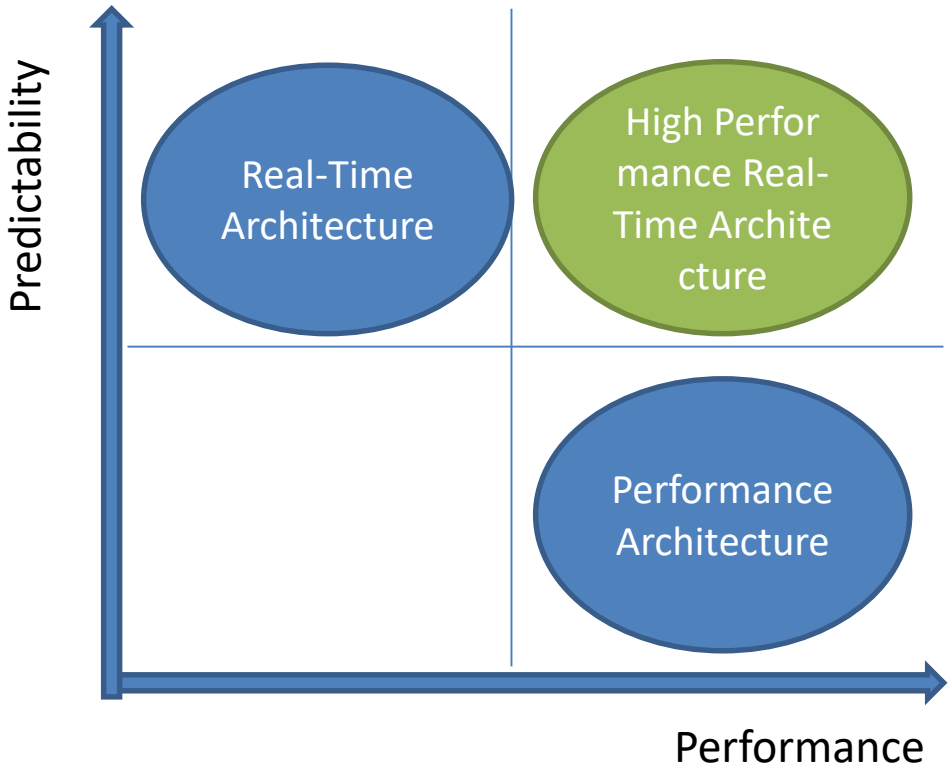
Isolation

- Traditionally about memory isolation
  - Prevent unauthorized access to memory
  - Hardware support: MPU, MMU
- What we need
  - **Prevent influence between domains**
  - Not only for real-time systems
  - But also for security[1]
- What hardware architecture/OS do we need?

KU THE UNIVERSITY OF KANSAS

[1] Q Ge, Y Yarom, T Chothia, G Heiser. "Time Protection: the Missing OS Abstraction". In *EuroSys*, 2019

Real-Time **AND** Real-Fast



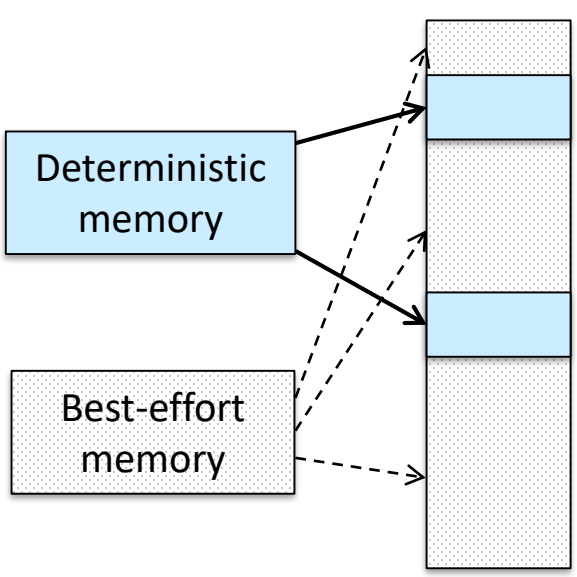
- Strong isolation **AND** high performance

How?

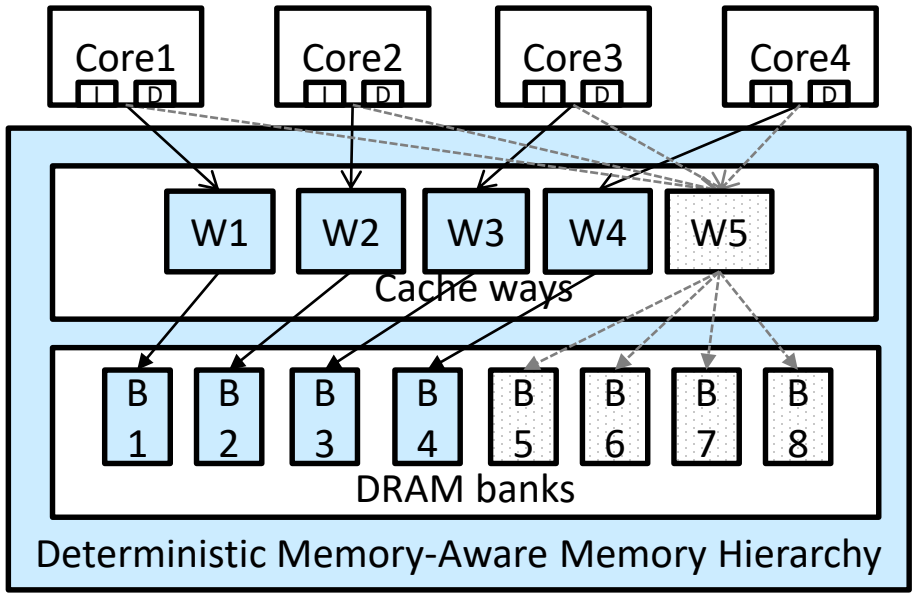
- Embrace complexity for high performance
  - Non-blocking cache, prefetcher, out-of-order execution engine, split-transaction bus, …
- Cross-layer OS/HW collaborative approach
  - Need to re-think existing abstractions
  - Need new SW/HW contracts to reason and control all things that affect timing

Deterministic Memory

- Declare all or part of address space as deterministic memory
- DM-aware **end-to-end resource management**



Application view (logical)



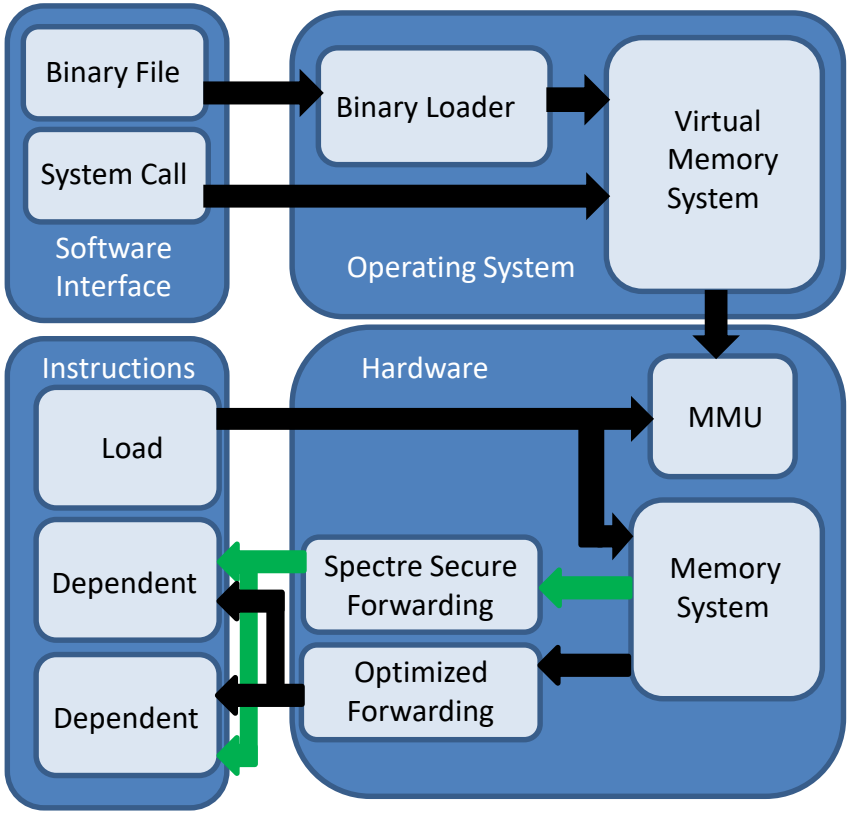
System-level view (physical)

Data-centric cross-layer approach for **real-time**

F. Farshchi, P. K. Valsan, H. Yun. "Deterministic memory abstraction and supporting multicore system architecture." In *ECRTS*, 2018

SpectreGuard

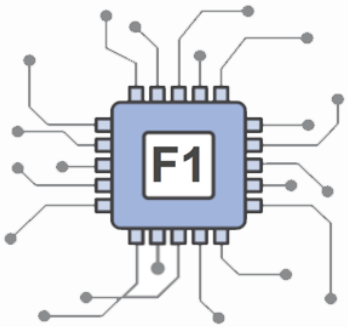
- Step 1: Software tells OS what **data** is secret
- Step 2: OS updates the page table entries
- Step 3: Load of the secret data is identified by MMU
- Step 4: secret data forwarding is **delayed** until safe



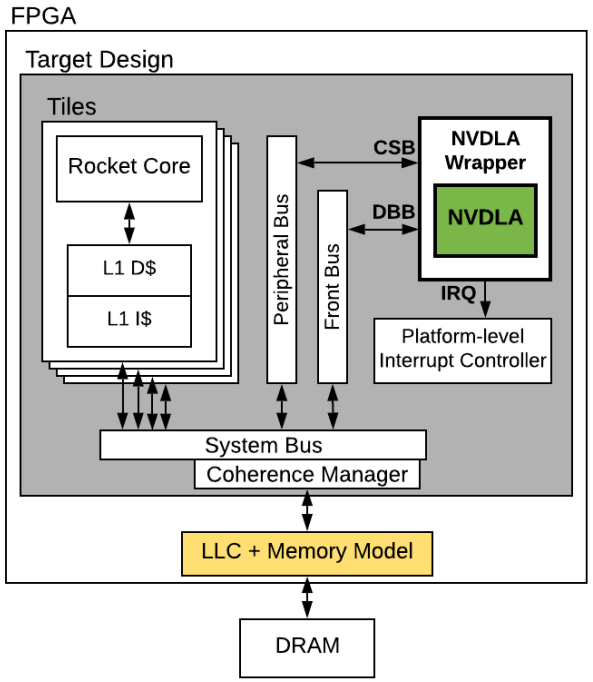
Data-centric cross-layer approach for **security**

J. Fustos, F. Farshchi, H. Yun. "SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks.." In *DAC*, 2019

RISC-V + NVDLA SoC Platform



amazon web services

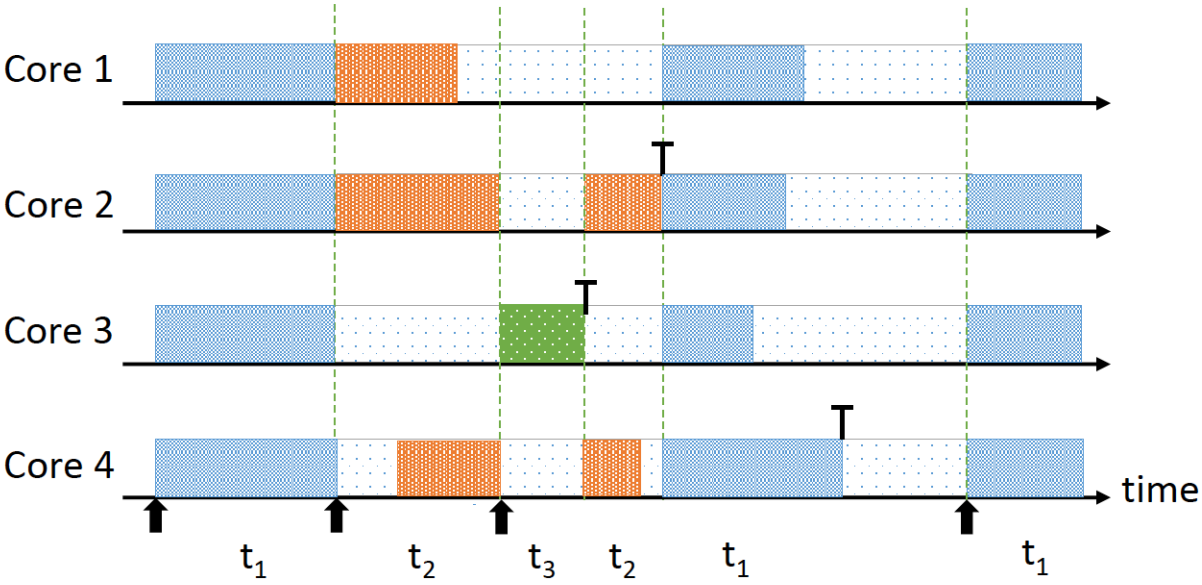


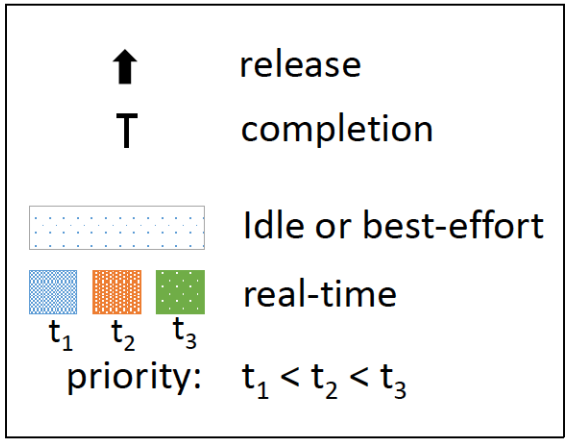
- Full-featured quad-core SoC with hardware DNN accelerator on Amazon FPGA cloud

Open-source hardware: big research opportunity!

F. Farshchi, Q Huang, H. Yun. "Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim." In *EMC^2*, 2019

RT-Gang



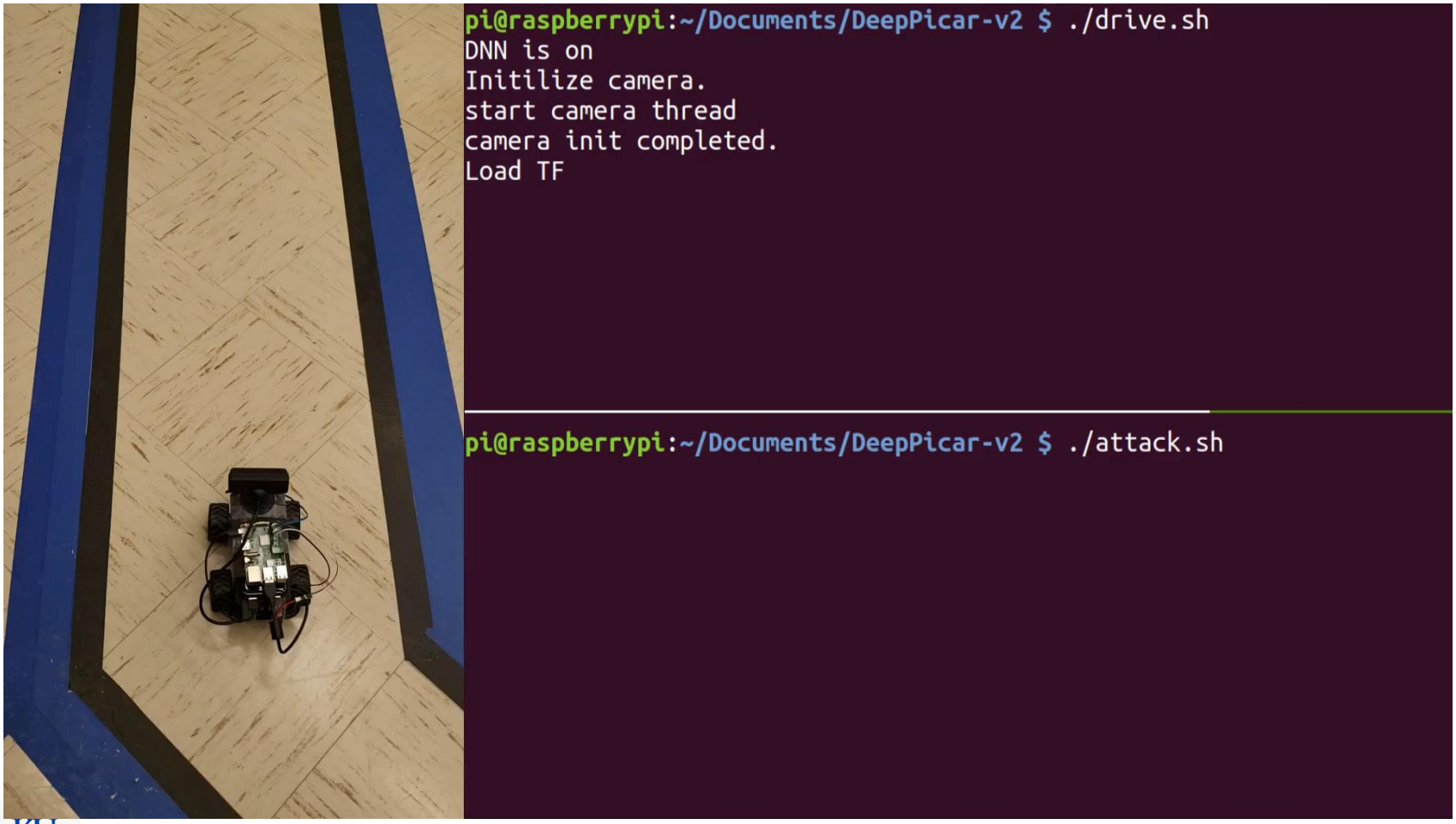


- **One parallel real-time task---a gang---at a time**
  - Eliminate inter-task interference by construction
- Schedule best-effort tasks during slacks **w/ throttling**

OS can do a lot more on COTS hardware

W. Ali and H. Yun. "RT-Gang: Real-Time Gang Scheduling Framework for Safety-Critical Systems." In *RTAS*, 2019

RT-Gang



https://youtu.be/pk0j063cUAs

Conclusion

- Micro-architectural attacks are a serious threat for intelligent CPS
  - Can leak secret (confidentiality)
  - Can alter data (integrity)
  - Can affect real-time performance (correctness)

- We need **better computing infrastructure** for safe, secure, and intelligent CPS
  - And we can **build** one

KU THE UNIVERSITY OF KANSAS

Thank You!

Acknowledgement:

This research is supported by NSA Science of Security initiative contract #H98230-18-D-0009 and NSF CNS 1718880, 1815959.

KU THE UNIVERSITY OF KANSAS

Recent Publications

1. [C] Jacob Michael Fustos, Farzad Farshchi, and Heechul Yun. SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks. *Design Automation Conference (**DAC**)*, 2019
2. [C] Waqar Ali and Heechul Yun. RT-Gang: Real-Time Gang Scheduling Framework for Safety-Critical Systems. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (**RTAS**)*, 2019.
3. [C] Michael Garrett Bechtel and Heechul Yun. Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (**RTAS**)*, 2019 **Outstanding Paper Award**
4. [W] Farzad Farshchi, Qijing Huang, and Heechul Yun. Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim. *Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (**EMC^2**)*, 2019.
5. [C] Michael Garrett Bechtel, Elise McEllhiney, Minje Kim, Heechul Yun. DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (**RTCSA**),* 2018
6. [C] Waqar Ali, Heechul Yun. Protecting Real-Time GPU Applications on Integrated CPU-GPU SoC Platforms. *Euromicro Conference on Real-Time Systems (**ECRTS**)*, 2018
7. [C] Farzad Farshchi, Prathap Kumar Valsan, Renato Mancuso, Heechul Yun. Deterministic Memory Abstraction and Supporting Multicore System Architecture. *Euromicro Conference on Real-Time Systems (**ECRTS**)*, 2018
8. [J] Prathap Kumar Valsan, Heechul Yun, Farzad Farshchi. Addressing Isolation Challenges of Non-blocking Caches for Multicore Real-Time Systems. ***Real-time Systems***, Vol: 53, Issue: 5, pp: 673–708, 2017
9. [J] Heechul Yun, Waqar Ali, Santosh Gondi, Siddhartha Biswas. BWLOCK: A Dynamic Memory Access Control Framework for Soft Real-Time Applications on Multicore Platforms. ***IEEE Transactions on Computers***, Vol: 66, Issue: 7, pp: 1247-1252, 2017
10. [C] Prasanth Vivekanandan, Gonzalo Garcia, Heechul Yun, Shawn Keshmiri. A Simplex Architecture for Intelligent and Safe Unmanned Aerial Vehicles. *IEEE Intl. Conf. on Embedded and Real-Time Computing Systems and Applications (**RTCSA**)*, 2016. **Best Student Paper Nomination**
11. [C] Prathap Kumar Valsan, Heechul Yun, Farzad Farshchi . Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems. In *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (**RTAS**)*, 2016. **Best Paper Award**
12. [C] Heechul Yun, Gang Yao, Rodolfo Pellizzoni, Marco Caccamo, and Lui Sha. Memory Bandwidth Management for Efficient Performance Isolation in Multi-core Platforms, ***IEEE Transactions on Computers***, Vol 65, Issue 2, 2016, pp. 562 – 576. **Editor's Pick of the year 2016**

Full List: http://www.ittc.ku.edu/~heechul/pub.html

KU THE UNIVERSITY OF KANSAS