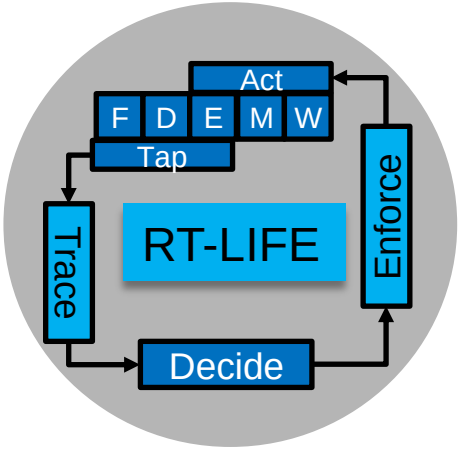
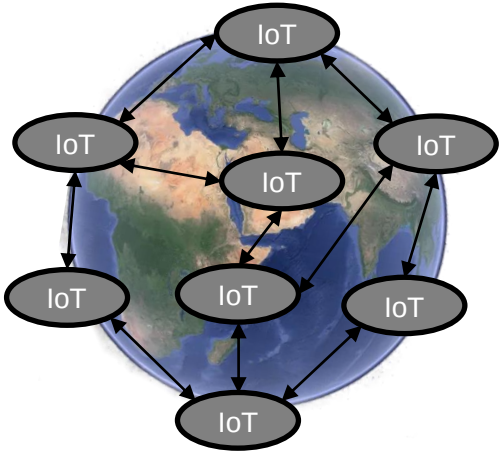
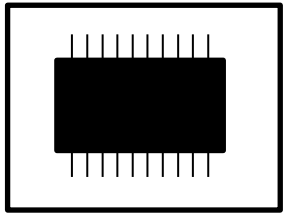
RT-LIFE

Portable RISC-V Interface for Real-Time Lightweight Security Enforcement

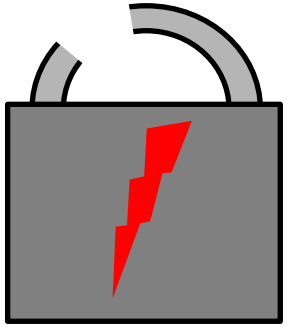


Tight Resource Constraints limit IoT Security



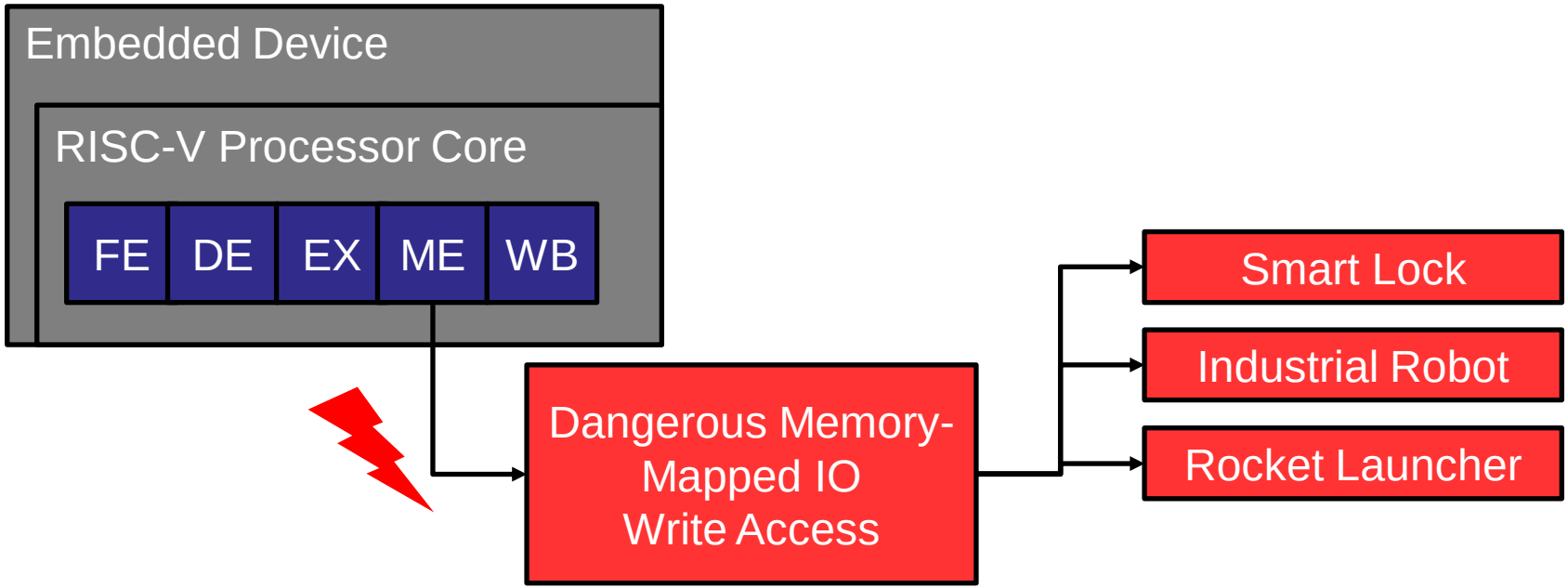


Cost, Energy, Heat,
Chip Area, Memory,
Computational Power



Security Considerations
Scenario

Real-Time embedded IoT device with potentially harmful MMIO periphery.



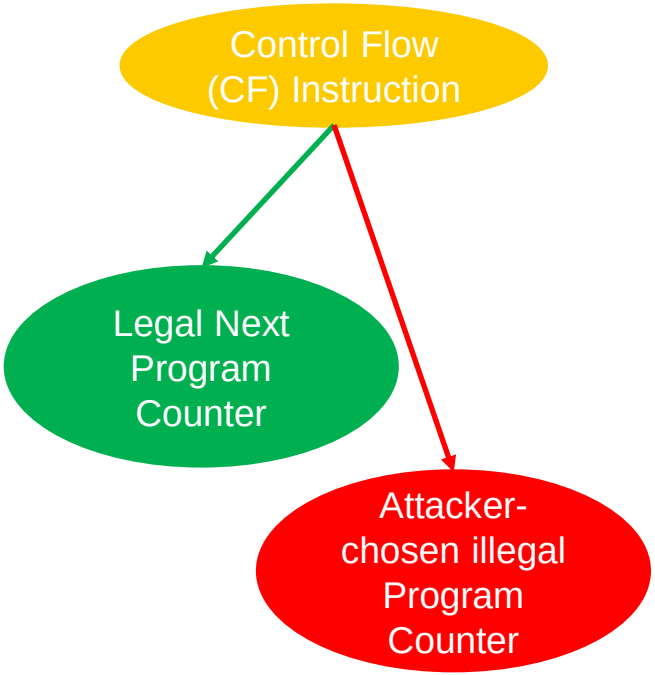
Security Arms Race (1/2) – Code Injection

```
int ORIGINAL_main(){
  int b=0;
  if(b) {
    get(R);
  }
  return 0;
}
```

```
int ATTACKER_main(){
  float evil = 1.0;
  if(evil){
    destroy_world();
  }
  return 0;
}
```

Existing mitigations: Memory Management Unit, Memory Protection Unit, Data Execution Prevention, Read-Only Memory, Address Space Layout Randomization

Security Arms Race (2/2) – Code Reuse



Without injecting new malicious code, runtime-behavior of existing code is tampered

↓

Static write protection inadequate

↓

Runtime-dynamic integrity required

Code Reuse - Return Oriented Programming (1/3)

1. Search code gadgets
2. Exploit memory error(s)
3. Manipulate return address (on stack) to concatenate gadgets

```
int calledFromMain(){
  int b=0;
  if(b) {
    get(R);
  }
  return 0;
}
```

Main

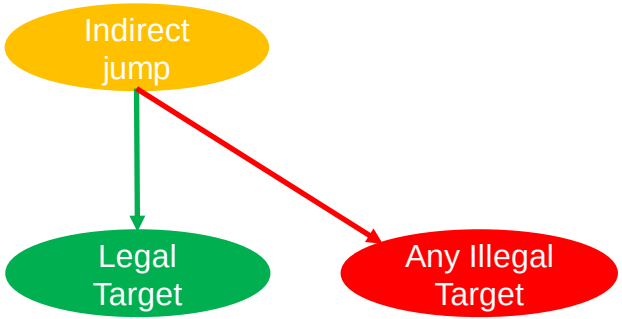
Evil

Stack		Shadow Stack
0x230 <Evil>	≠	0x80 <Main>
0x184	=	0x184

Code Reuse - Jump Oriented Programming (2/3)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Manipulate *heap* memory to tamper indirect jumps and concatenate gadgets



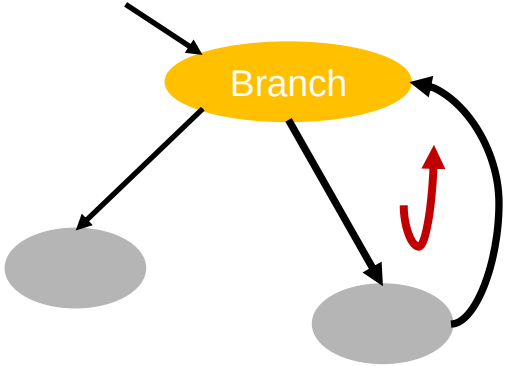
- FSM Logic constraining the Control Flow Graph
- Pointer integrity
- Artificial intelligence

…

Embedded Systems & Applications

Code Reuse - Data Oriented Programming (3/3)

- Tamper data values to slightly manipulate Control Flow (CF) without violating the Control Flow Graph
- Repurpose rarely used memory for virtual registers

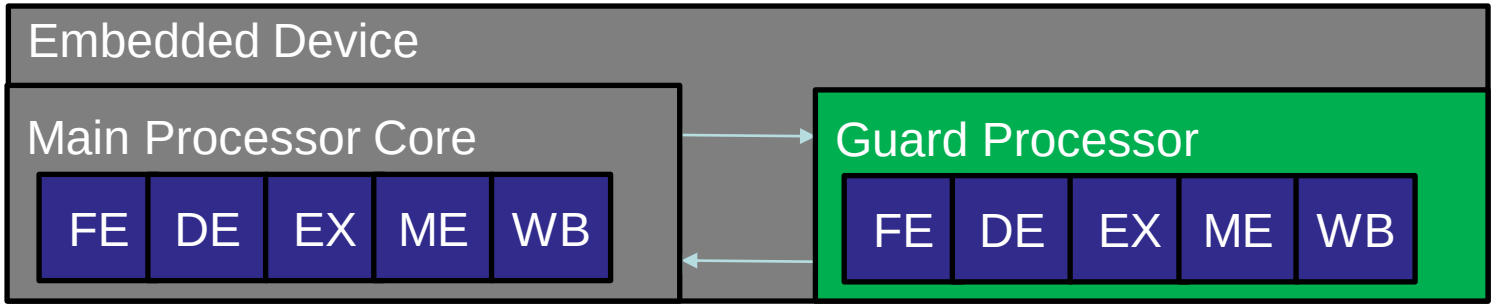


- Fine granular control flow integrity
- Data Flow Integrity
- Data Invariant Integrity

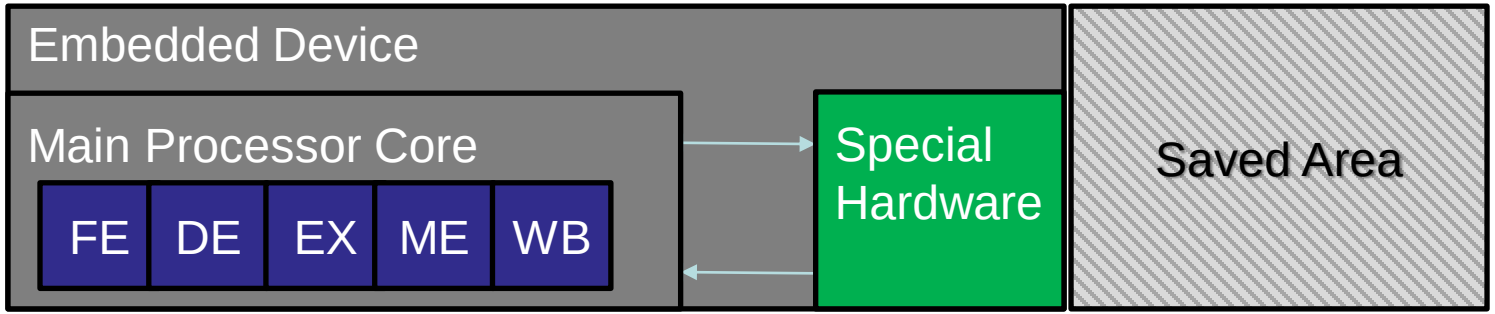
Many different attacks and *countermeasures* in industry and research!

Software / Hardware Partitioning

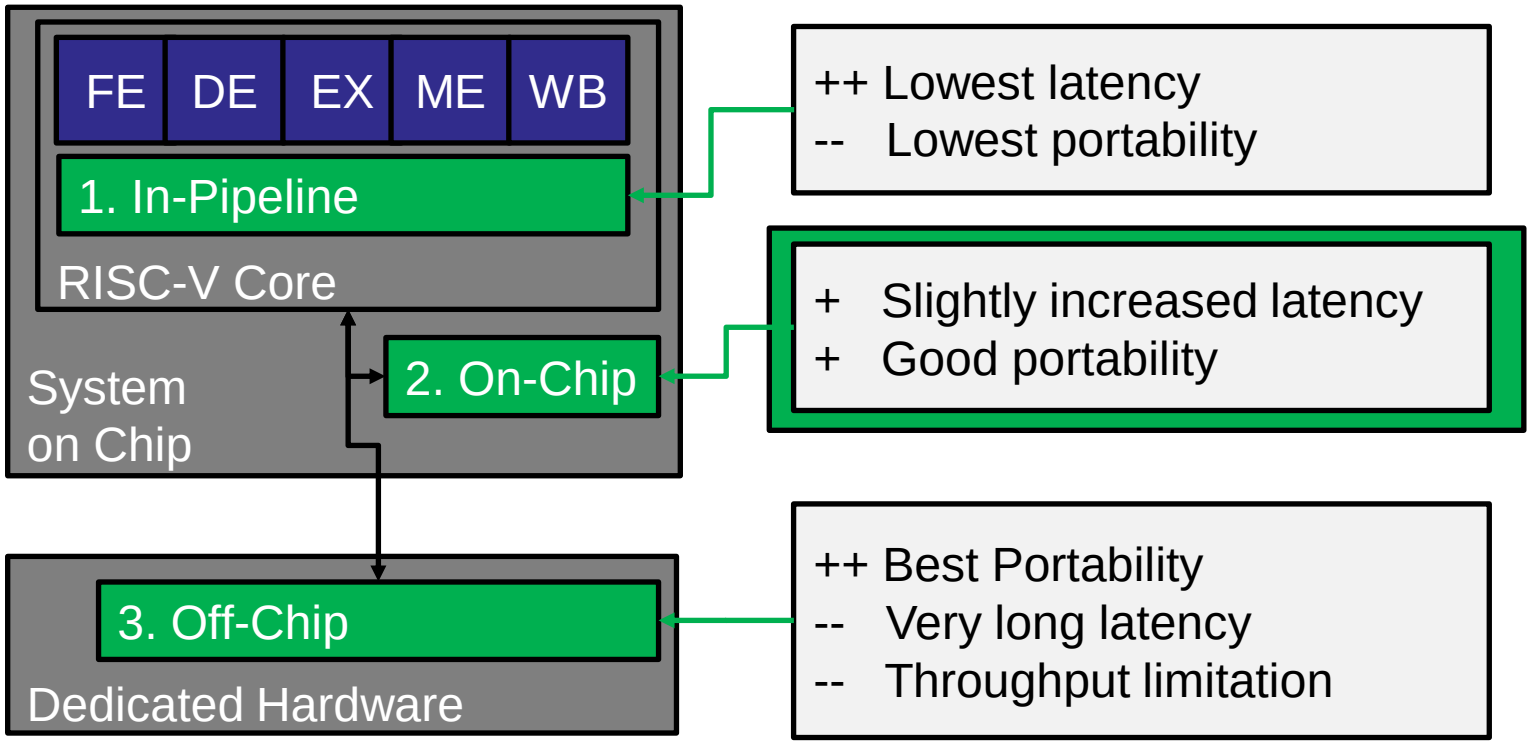
1. Software-only: Running additional Standard-ISA checking instructions
2. Enhanced Pipeline: Implementing Non-Standard-ISA checking instructions
3. Hardware-Assisted:



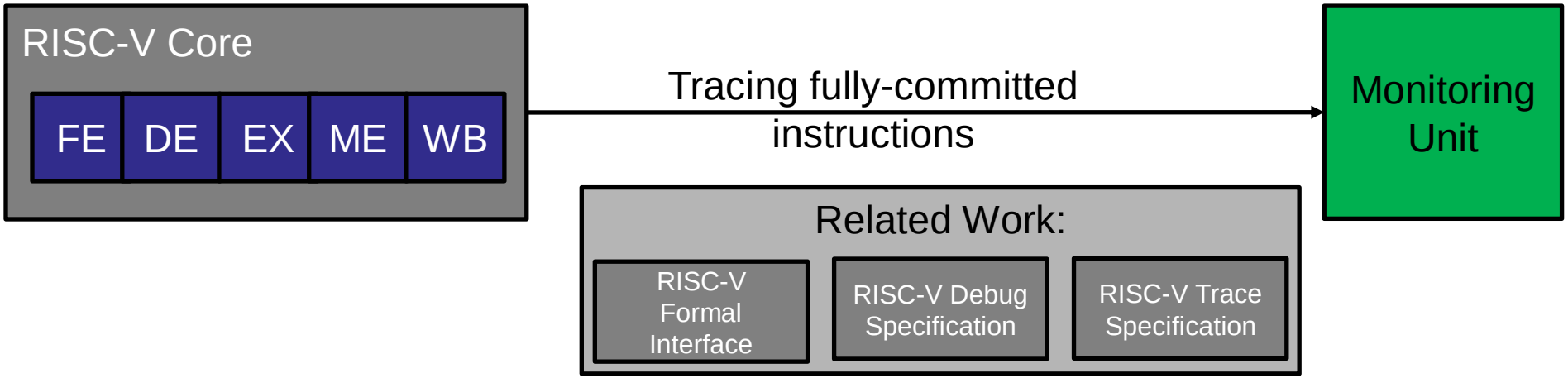
4. Specialized Hardware:



Locations of hardware Integrity modules

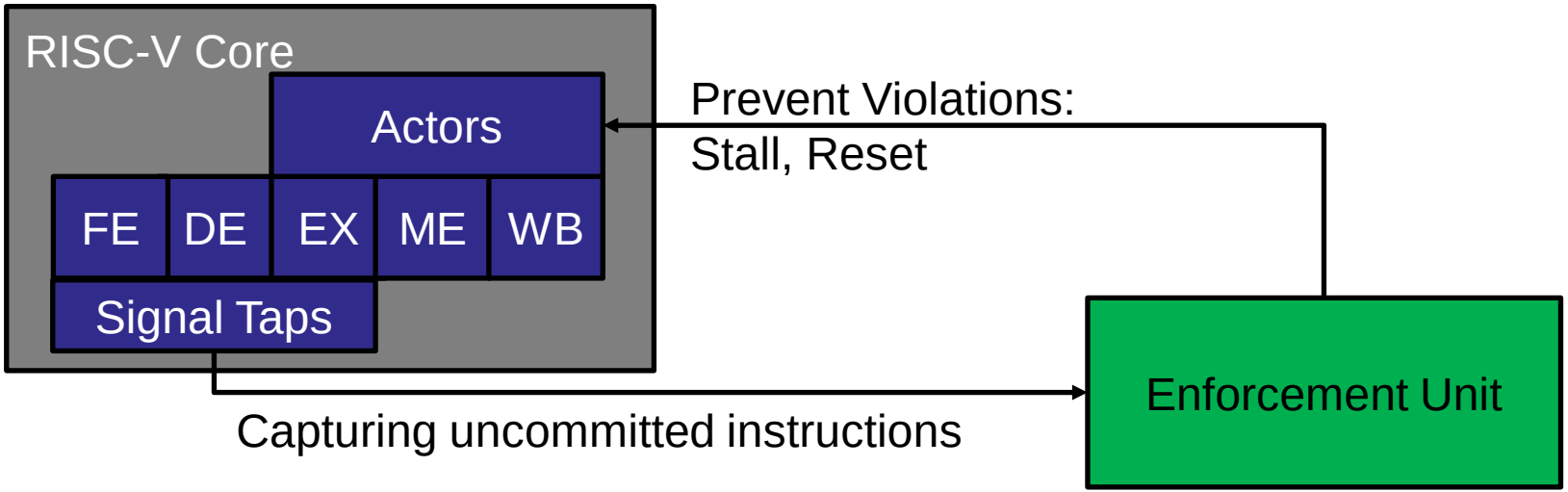


Timing: Monitoring vs. In-Time Enforcement (1/2)



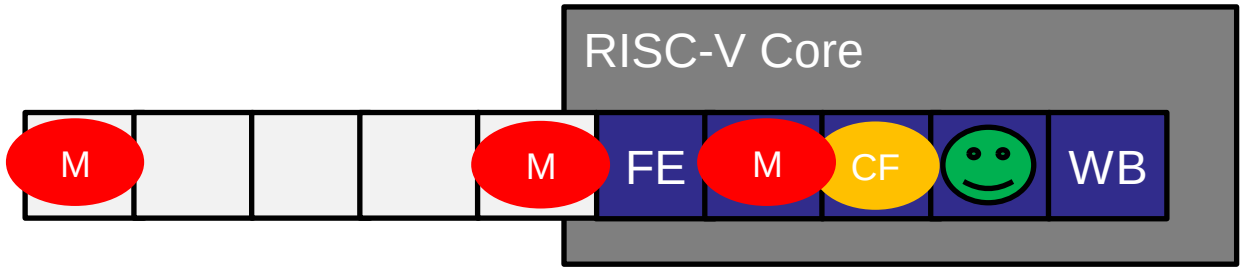
Monitoring: One-way ticket, passively monitors violations
Cannot prevent evil instructions in flight

Timing: Monitoring vs. In-Time Enforcement (2/2)



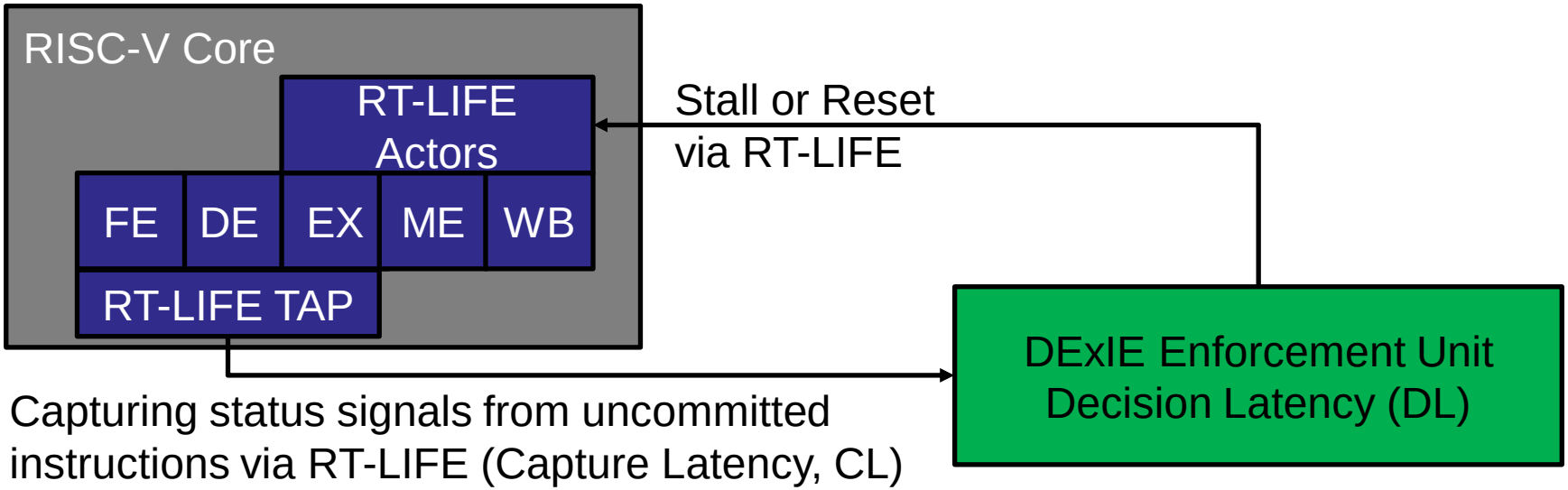
Prevention: Closed loop operation,
can actively prevent violations from taking effect

Preventing the Worst-Case Attack

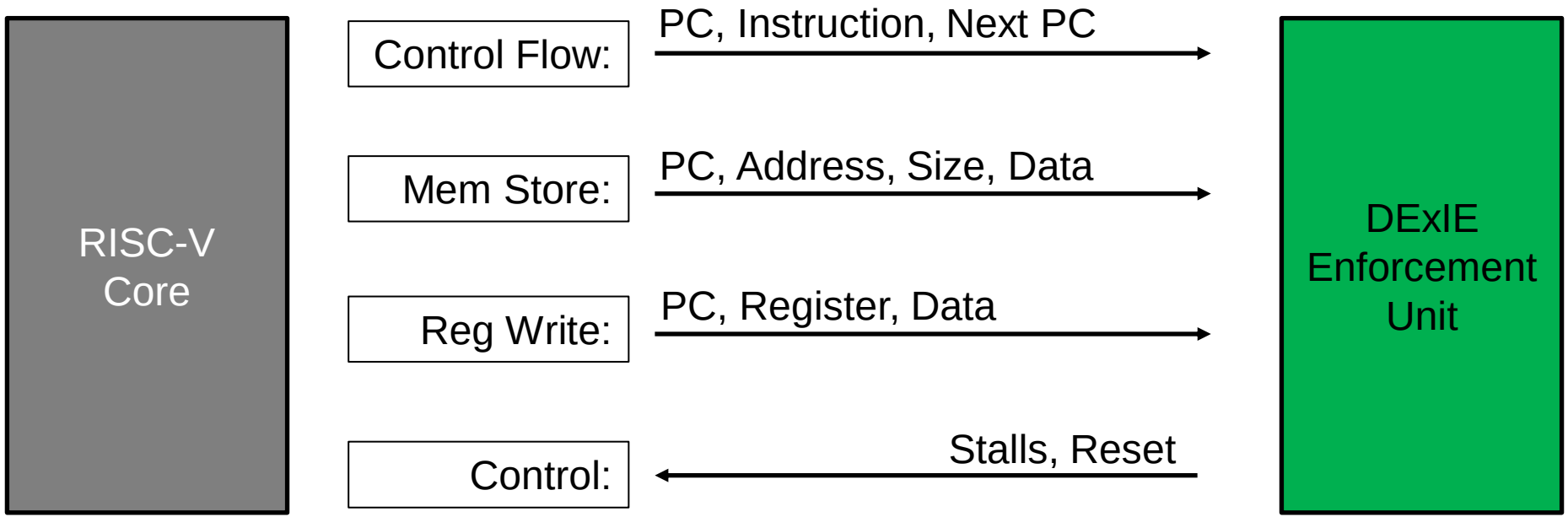


Timing is critical!

Prevention with RT-LIFE

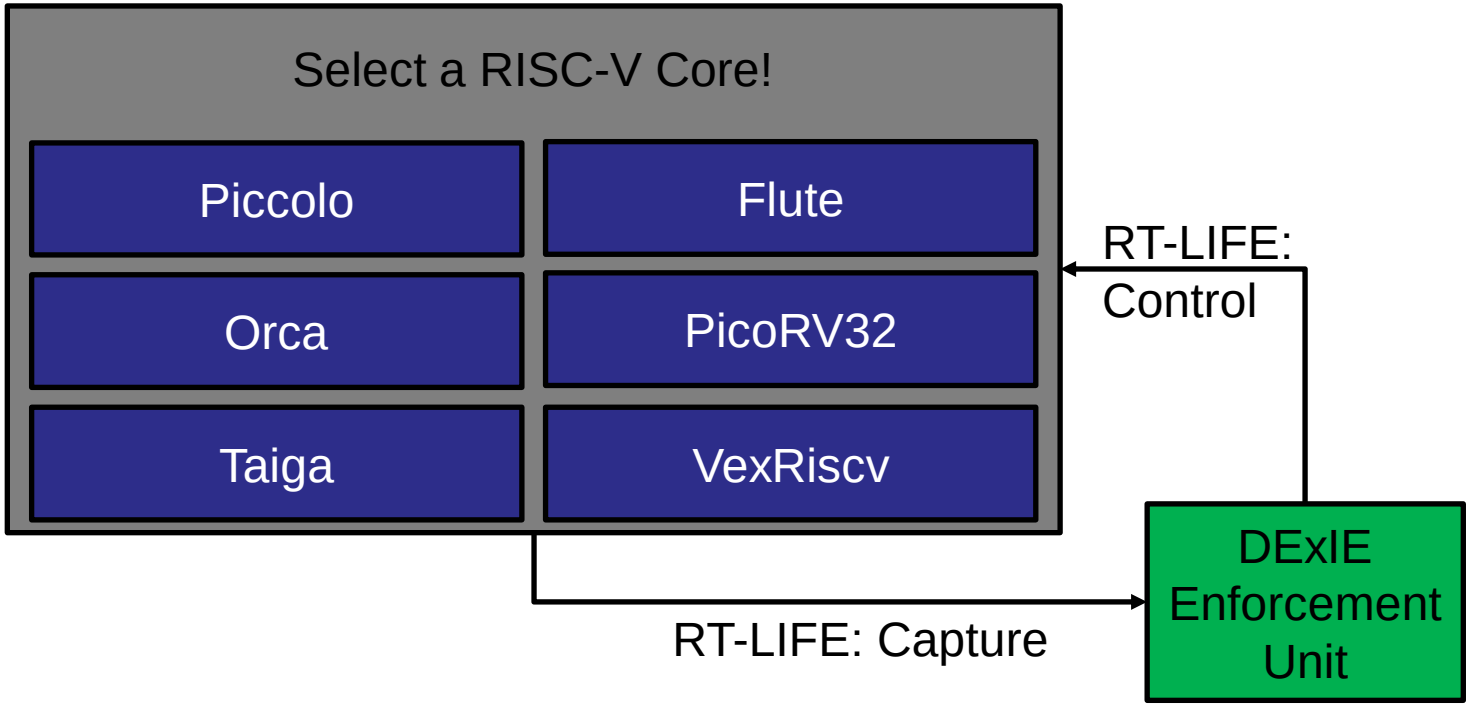


Architecture: Interface signal specification



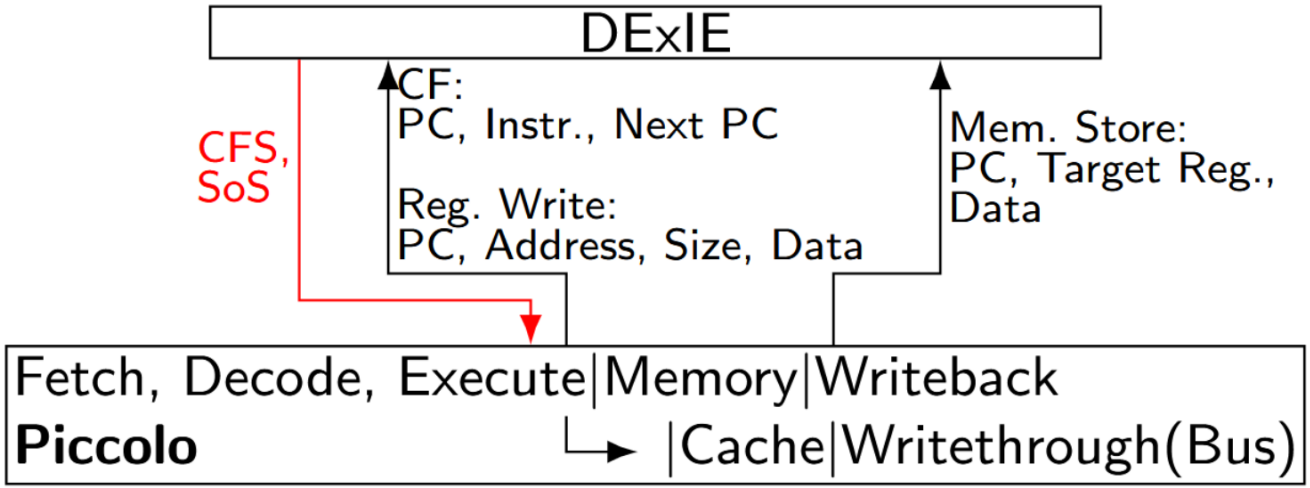
Design Space Exploration

Covering 6 RISC-V cores

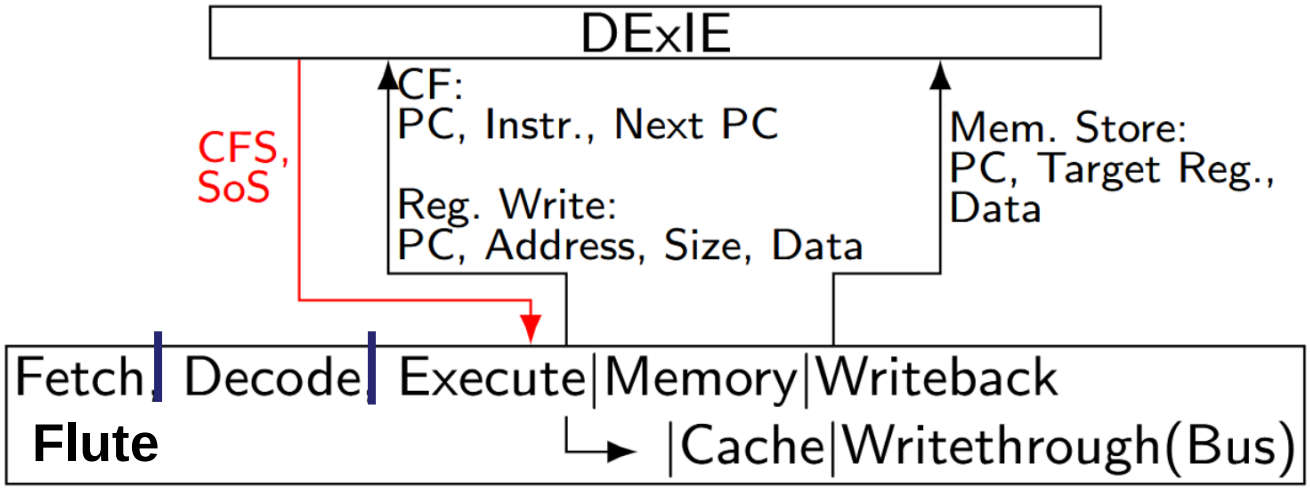


Microarchitecture (1/6)

Piccolo



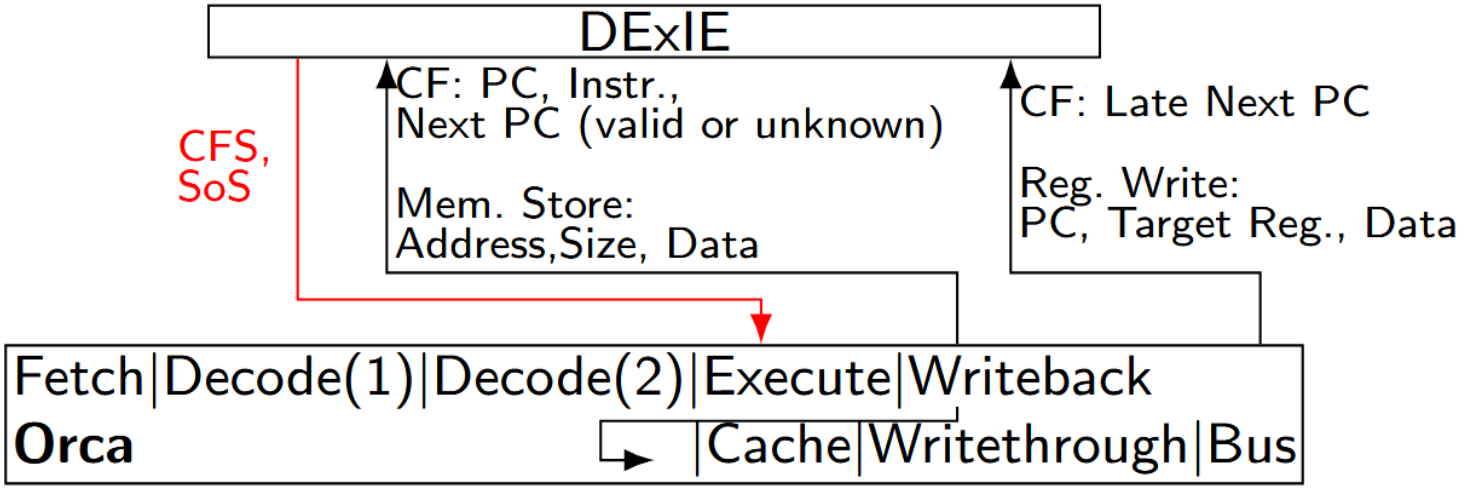
Microarchitecture (2/6)
Flute



Flute is similar to Piccolo, but separates more pipeline stages

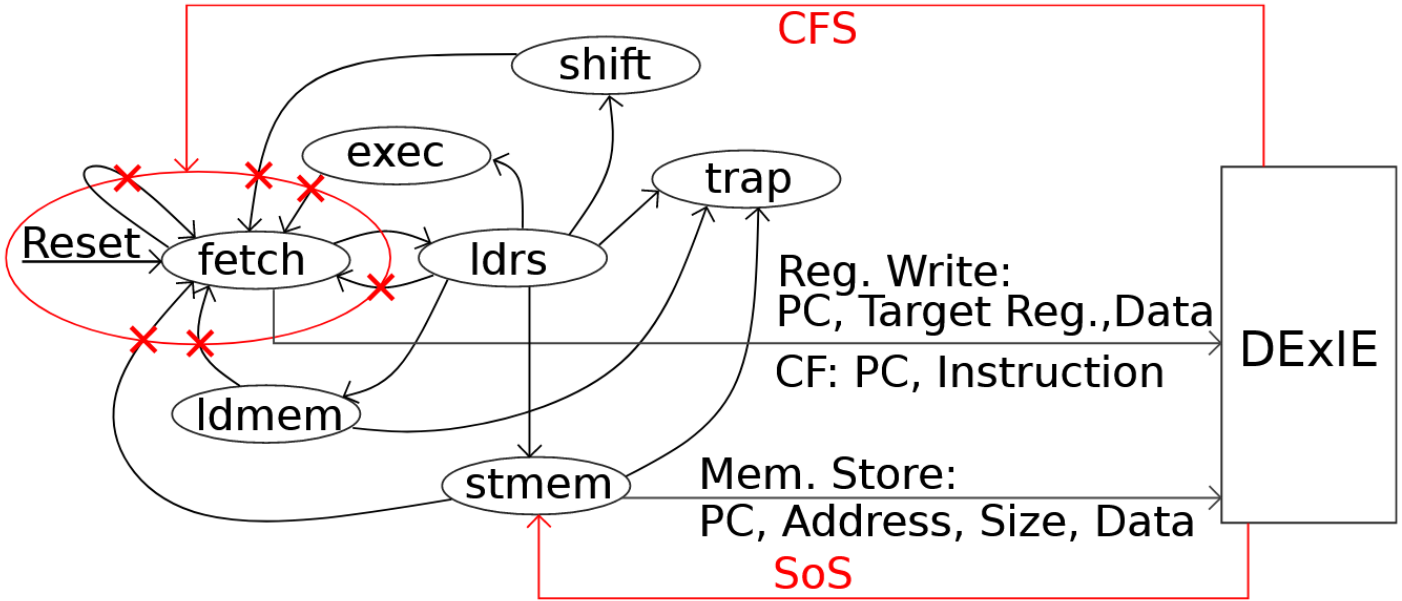
Microarchitecture (3/6)

Orca

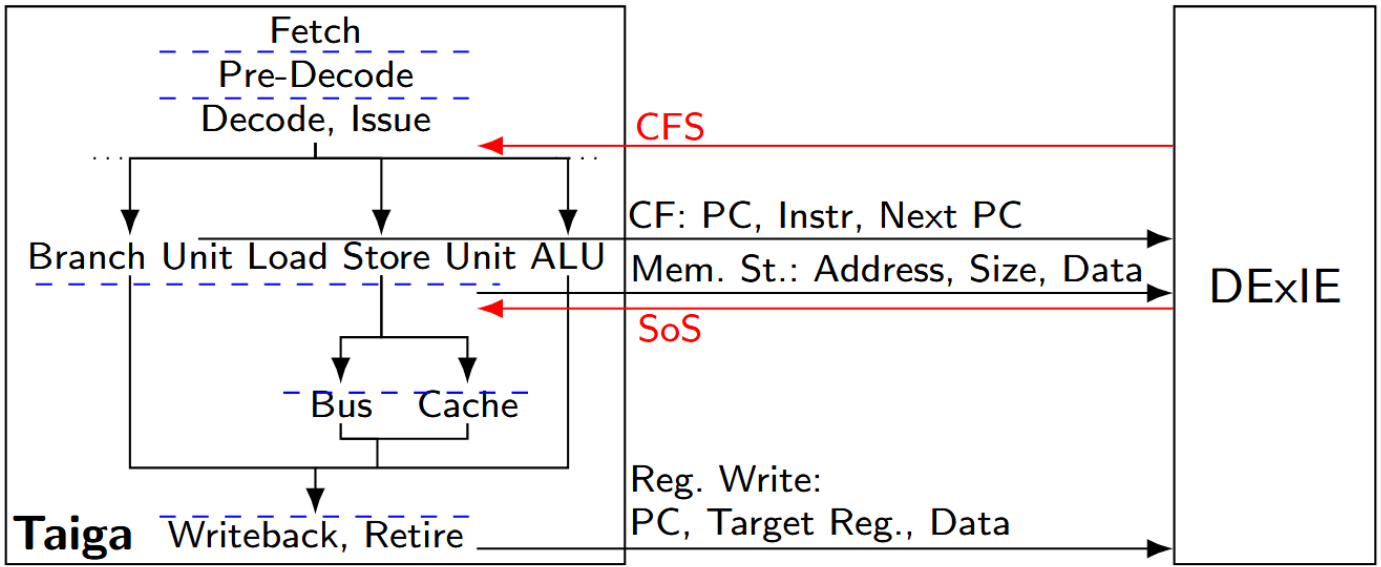


Microarchitecture (4/6)

PicoRV32

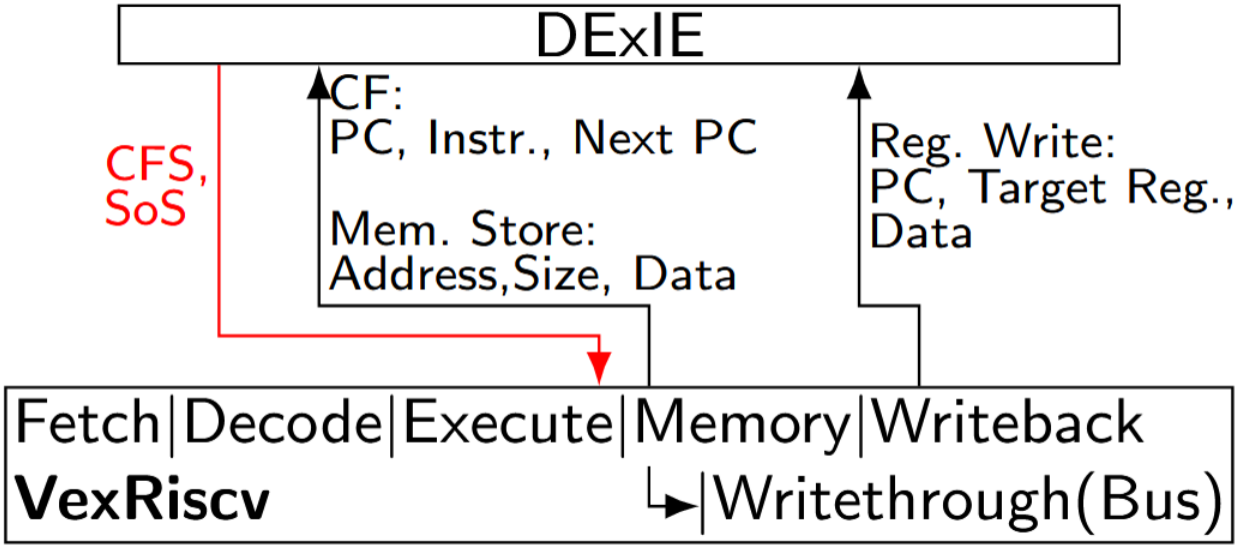


Microarchitecture (5/6)
Taiga

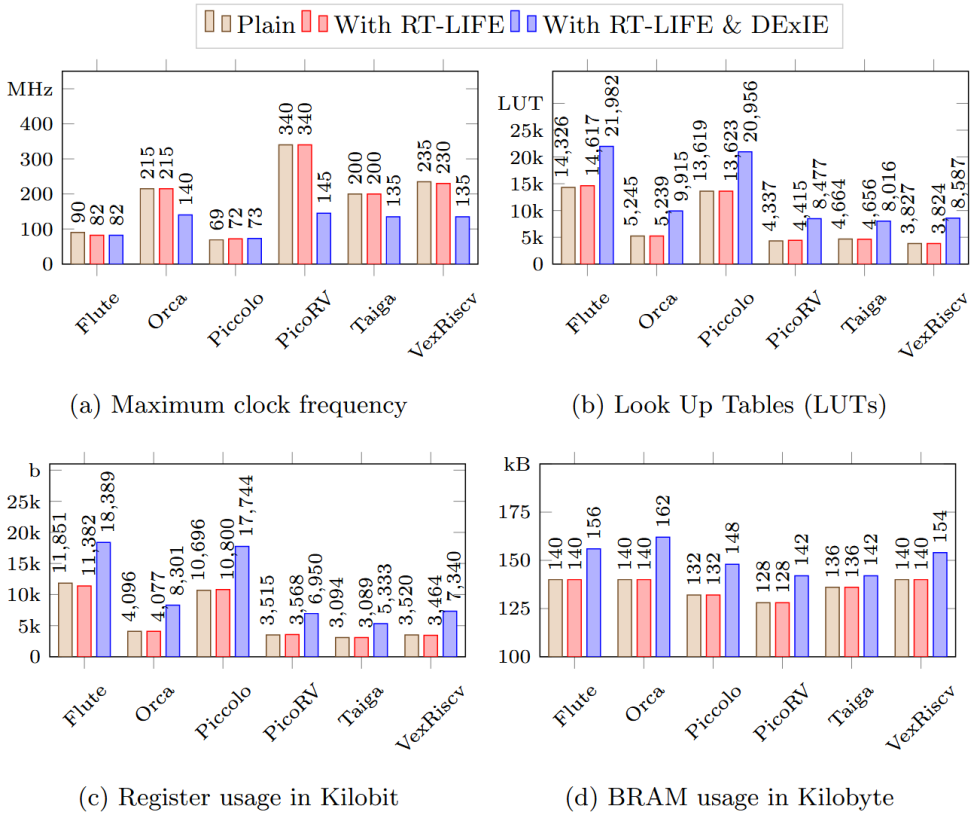


Microarchitecture (6/6)

VexRiscv



Evaluation



(a) Maximum clock frequency

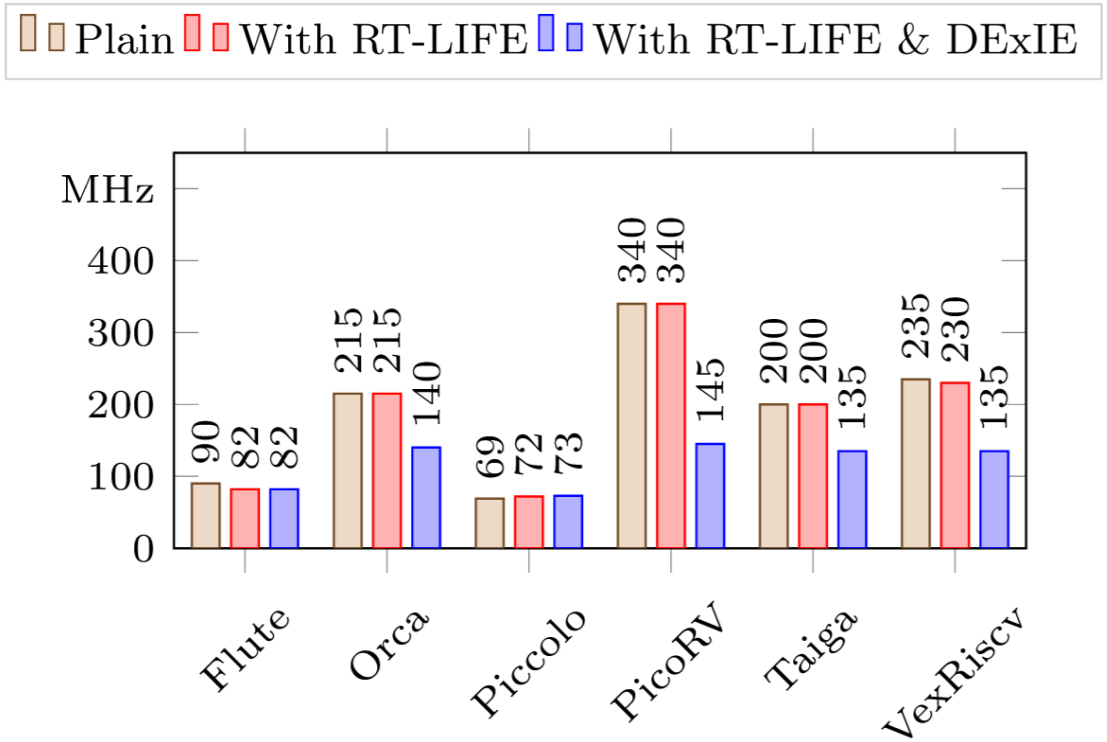
(b) Look Up Tables (LUTs)

(c) Register usage in Kilobit

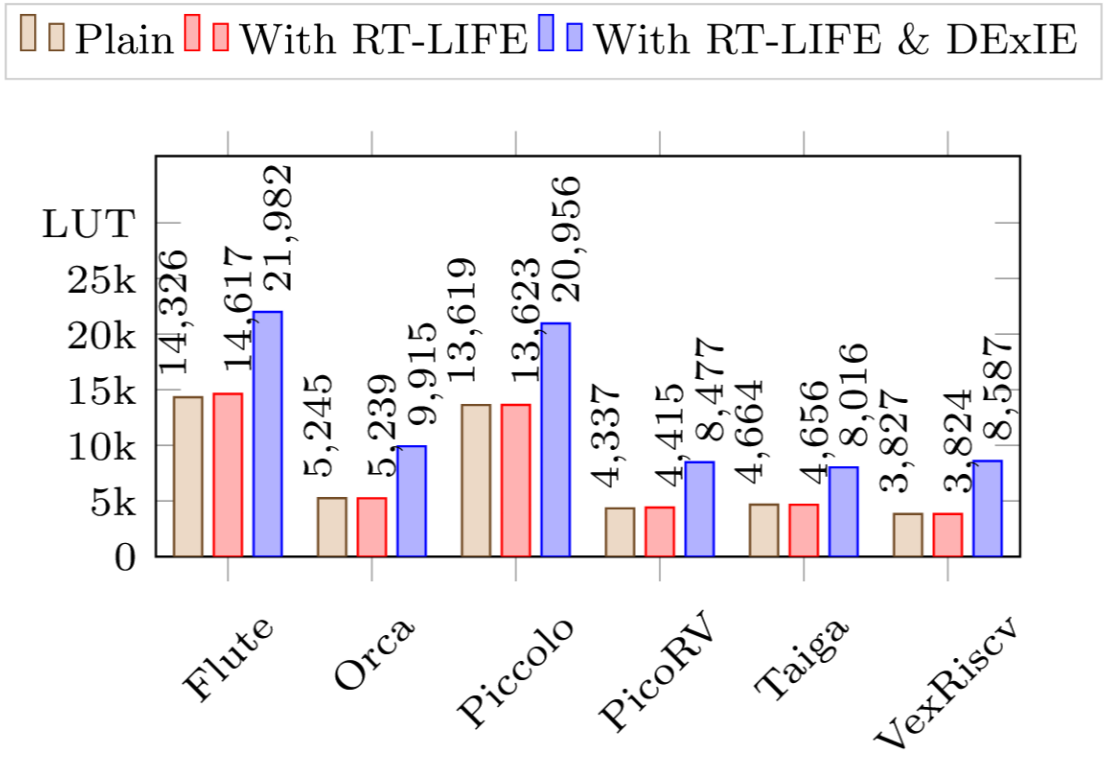
(d) BRAM usage in Kilobyte

Evaluation (1/4)

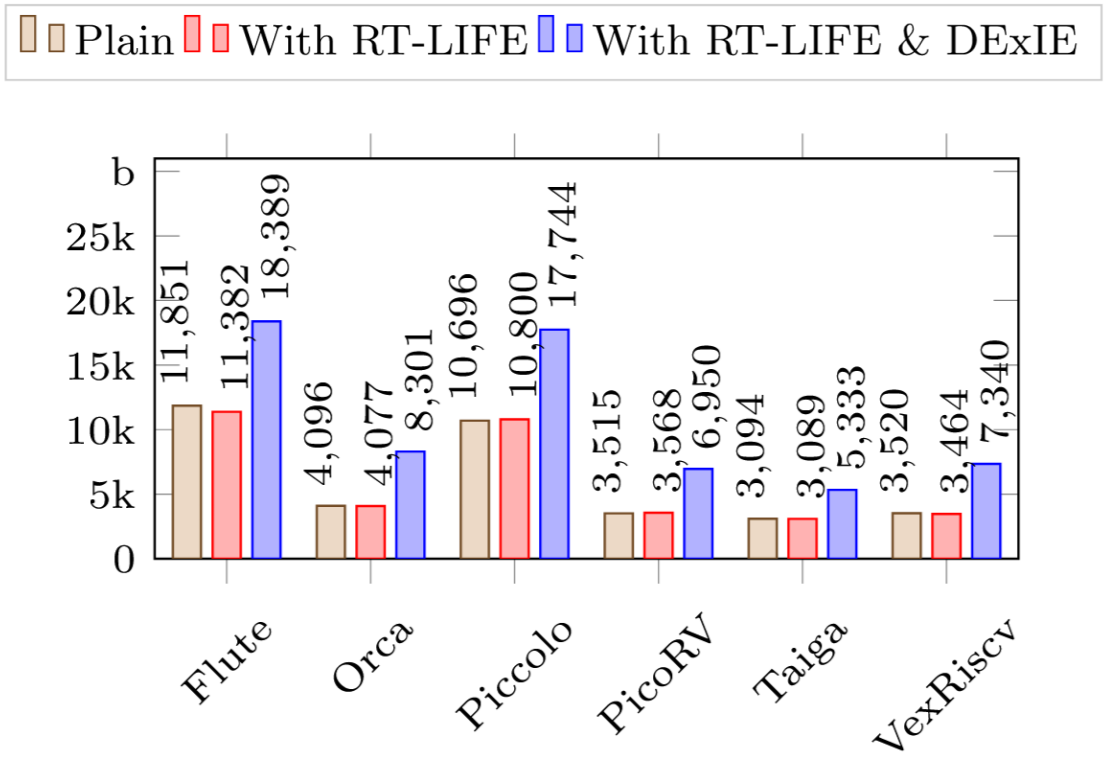
Maximum Clock Frequency in MHz



Evaluation (2/4)
Look Up Tables (LUTs)

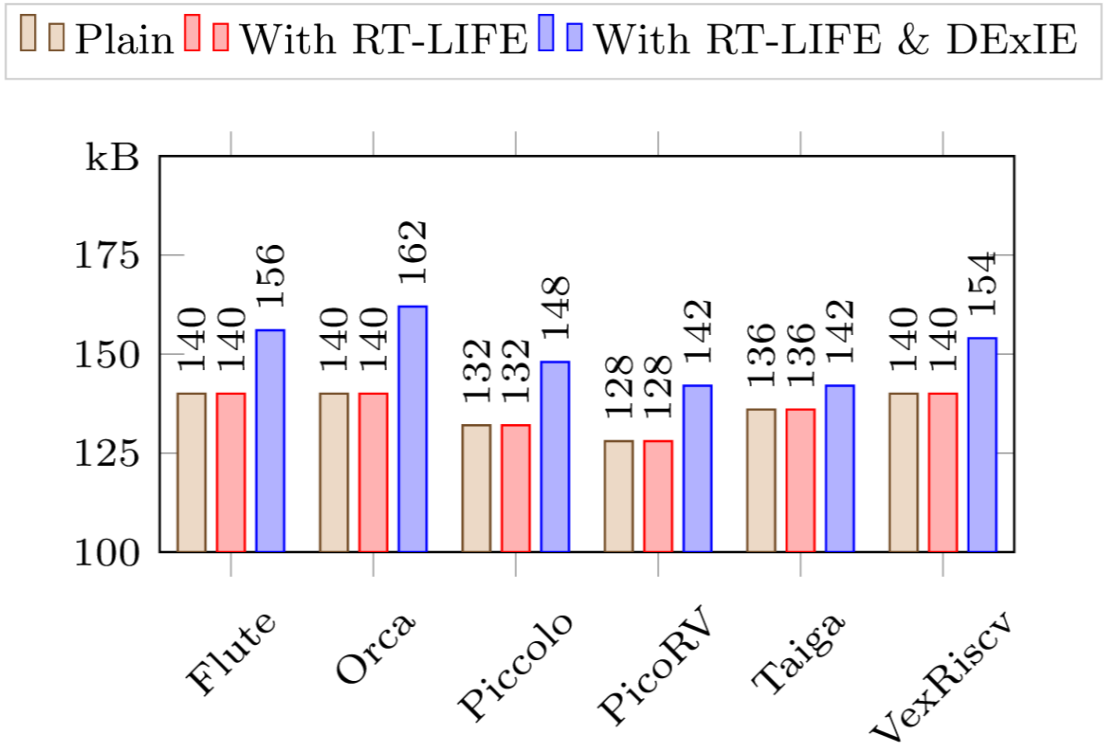


Evaluation (3/4)
Register Usage in Bit



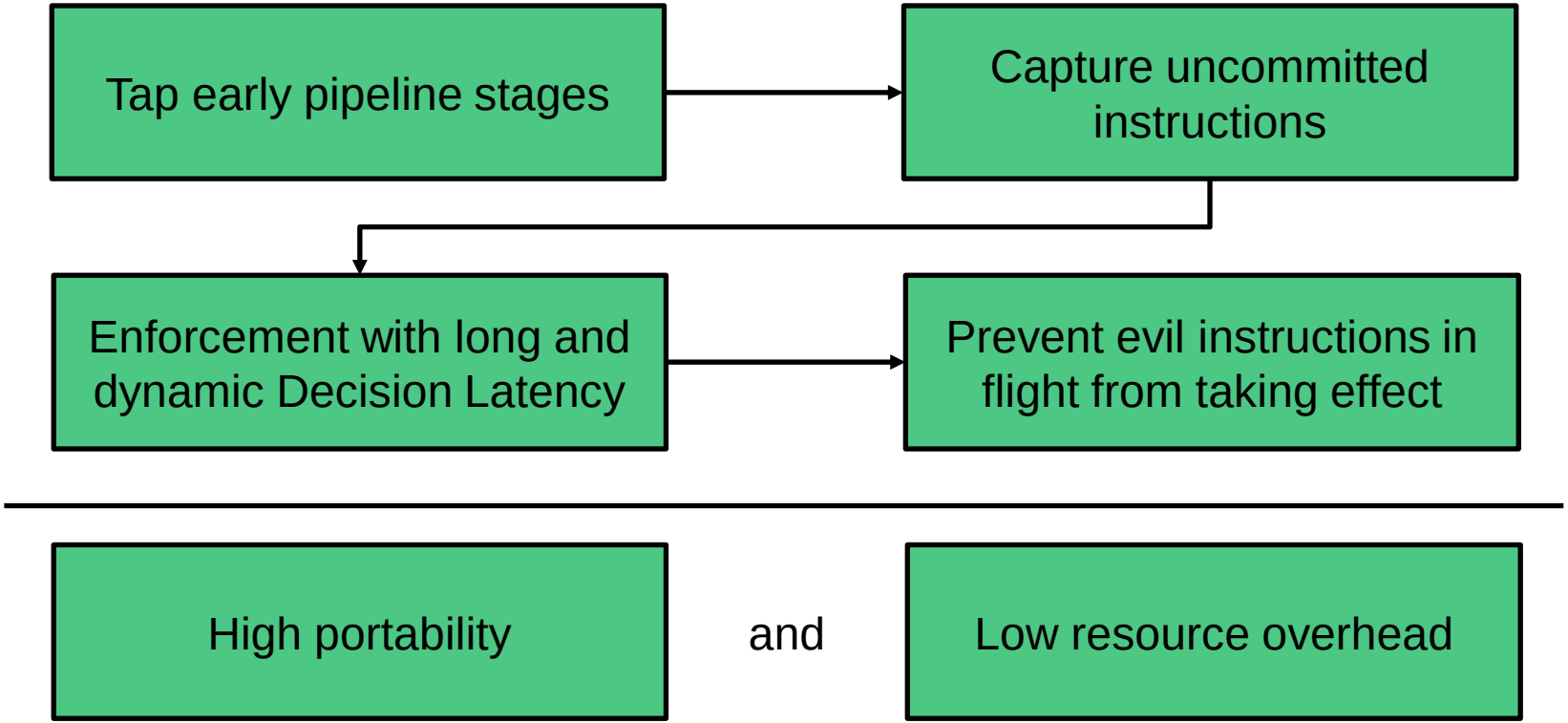
Evaluation (4/4)

BRAM Usage in Kilobyte



Conclusion

RT-LIFE: Real-Time Lightweight Integrity Enforcement Interface



Future Work

TECHNISCHE UNIVERSITÄT DARMSTADT

Attack Prevention
for
Out-of-order Cores

Reduced Capture Latency
via
Branch Prediction

Dynamic Attack Responses
via
Low-latency Interrupts

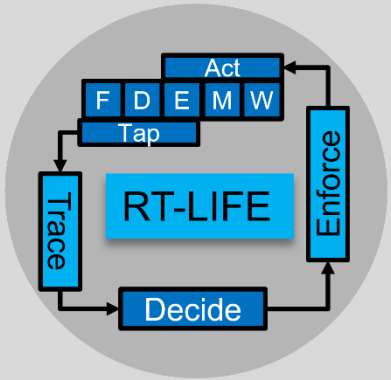
Runtime-Dynamic Security
Enforcement Units

Embedded Systems & Applications

Open Source

TECHNISCHE UNIVERSITÄT DARMSTADT

RT-LIFE on GitHub



https://github.com/esa-tu-darmstadt/RT-LIFE

Made with TaPaSCo

TASK PARALLEL SYSTEM COMPOSER
TU DARMSTADT
TAPASCO
SINCE 2015

https://github.com/esa-tu-darmstadt/tapasco

Embedded Systems & Applications