DYNAMIC MULTI-CORE INTERFERENCE MITIGATION

Dynamic Mitigation of Multi-Core Interference in Safety-Critical Systems

Moritz Meier, Institute of Flight Systems, 10.02.2026

Agenda

- Multi-Core Architecture & Interference
- Mitigation Strategies
- Dynamic Mitigation using WebAssembly
- Concept Demonstrator
- Evaluation

MULTI-CORE ARCHITECTURES

Multi-Core Architectures

- State of the Art since early 2000 in Workstations, Servers, Embedded, …
- Avionics still using Single-Core or Single-Active Core in the 2020s
- Why?

Multi-Core Interference

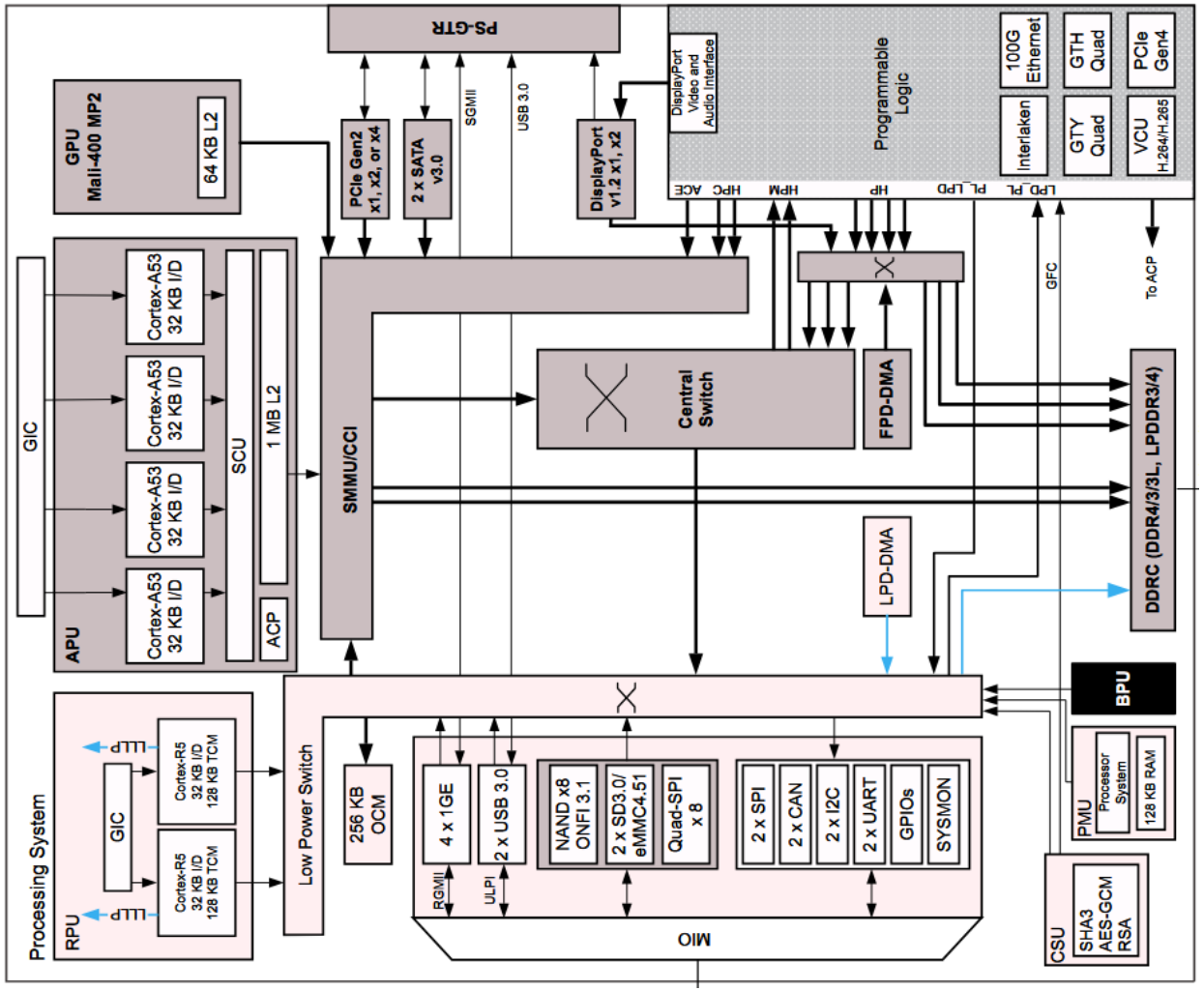
Advantage

- **More:** Compute Power
- **Less:** Power, Size, Weight
- Enabled by Caches, Interconnects
  → **Shared** Resources

Problem

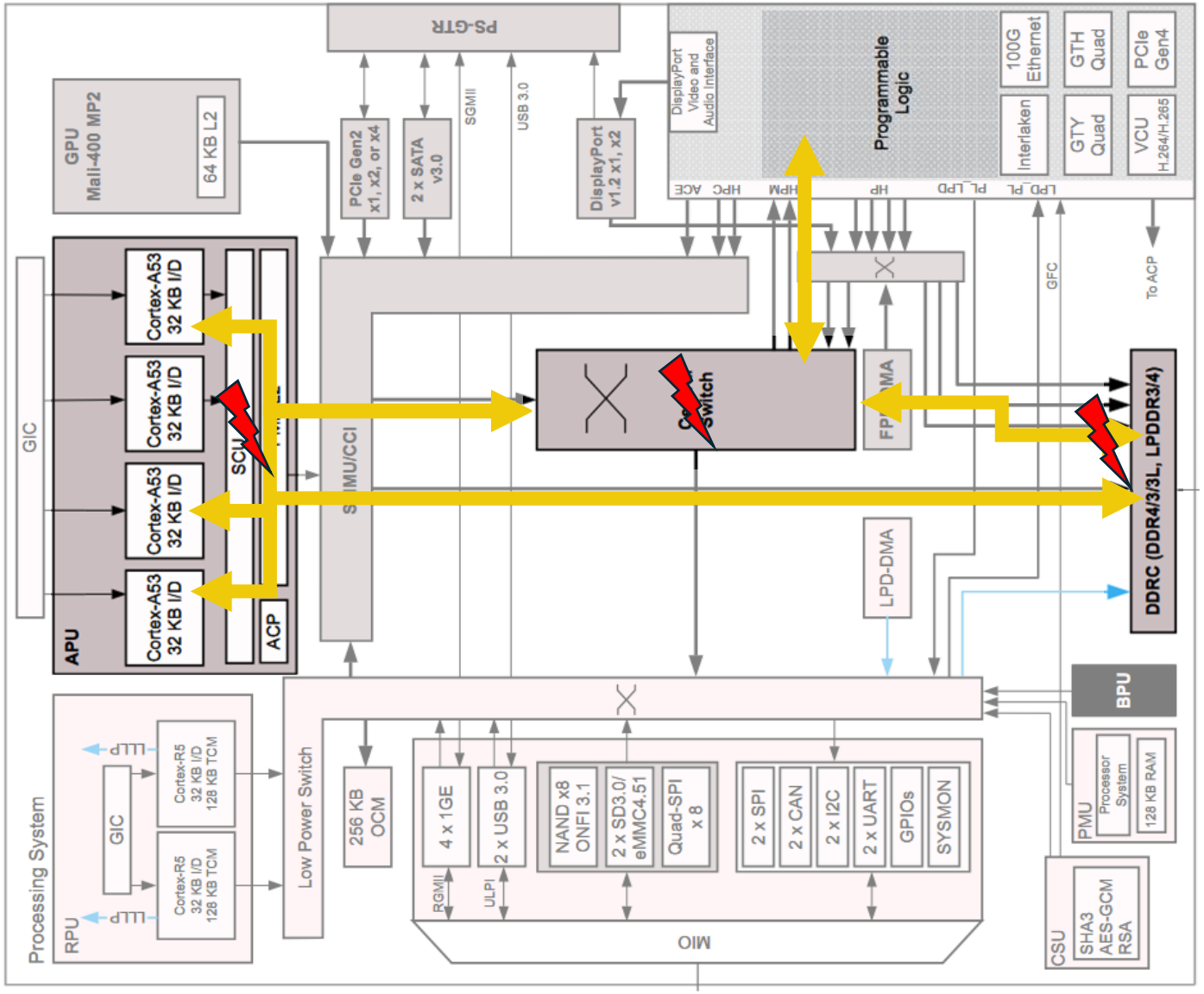
- **Shared** Resources need Arbitration / Allocation policies

→ Temporal Interference between Cores



AMD/Xilinx Zynq UltraScale+ MPSoC Embedded Design Methodology Guide (UG1228)

Multi-Core Interference



MITIGATION STRATEGIES

Mitigation Strategies

Static Mitigation

- Cache Partitioning
- Schedulability Analysis
- Disabling Secondary Cores (Single-Active Core)

→ Causes a lot of performance degradation

→ Often only mitigates a specific Interference Channel

→ Requires intricate knowledge of Hardware Implementation

Mitigation Strategies

- Trade-Off has to be made

Mitigation Strategies

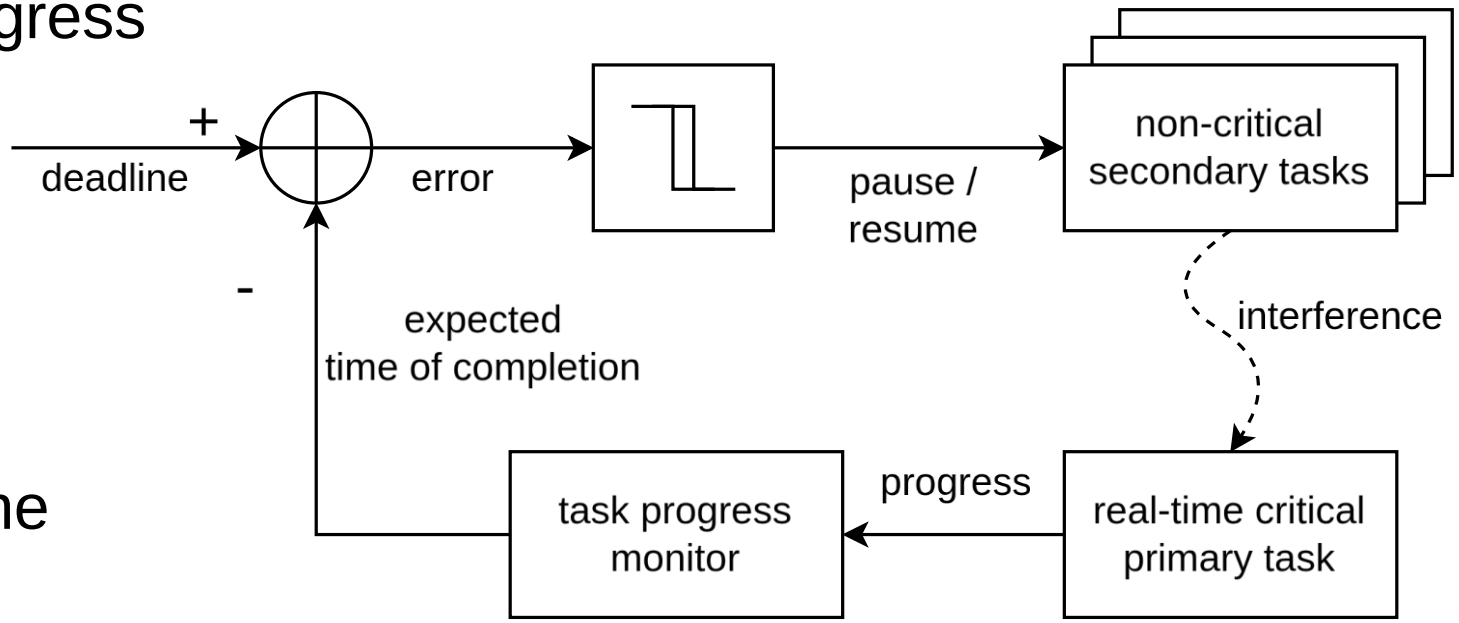
Naïve Idea

- Interference degrades Execution Speed
- Execution Speed is observable
- Interference is caused by Software Execution
- Software Execution is controllable

Mitigation Strategies

Dynamic Mitigation

- Measure Computational Progress
- Extrapolate throughput to exp. Time-of-Completion
- Control-Loop to keep deadline
- Activate Mitigation if necessary



Mitigation Strategies

Advantages

- Independent of interference channel mechanism
- Catches all interference caused by software
- No hardware knowledge required

Disadvantages

- Only one Critical Task can run at any time
- All other Tasks have to be considered Non-Critical

Mitigation Strategies

Dynamic Mitigation

How to measure the Computational Progress?

- In Software: Checkpoints
  - In known Intervals (fixed amount of Compute between Checkpoints)

  → Intrusive to the Application

- In Hardware: Performance Monitoring Unit
  - I.e. via Instruction Counting

  → Causes tight coupling between Software and Hardware

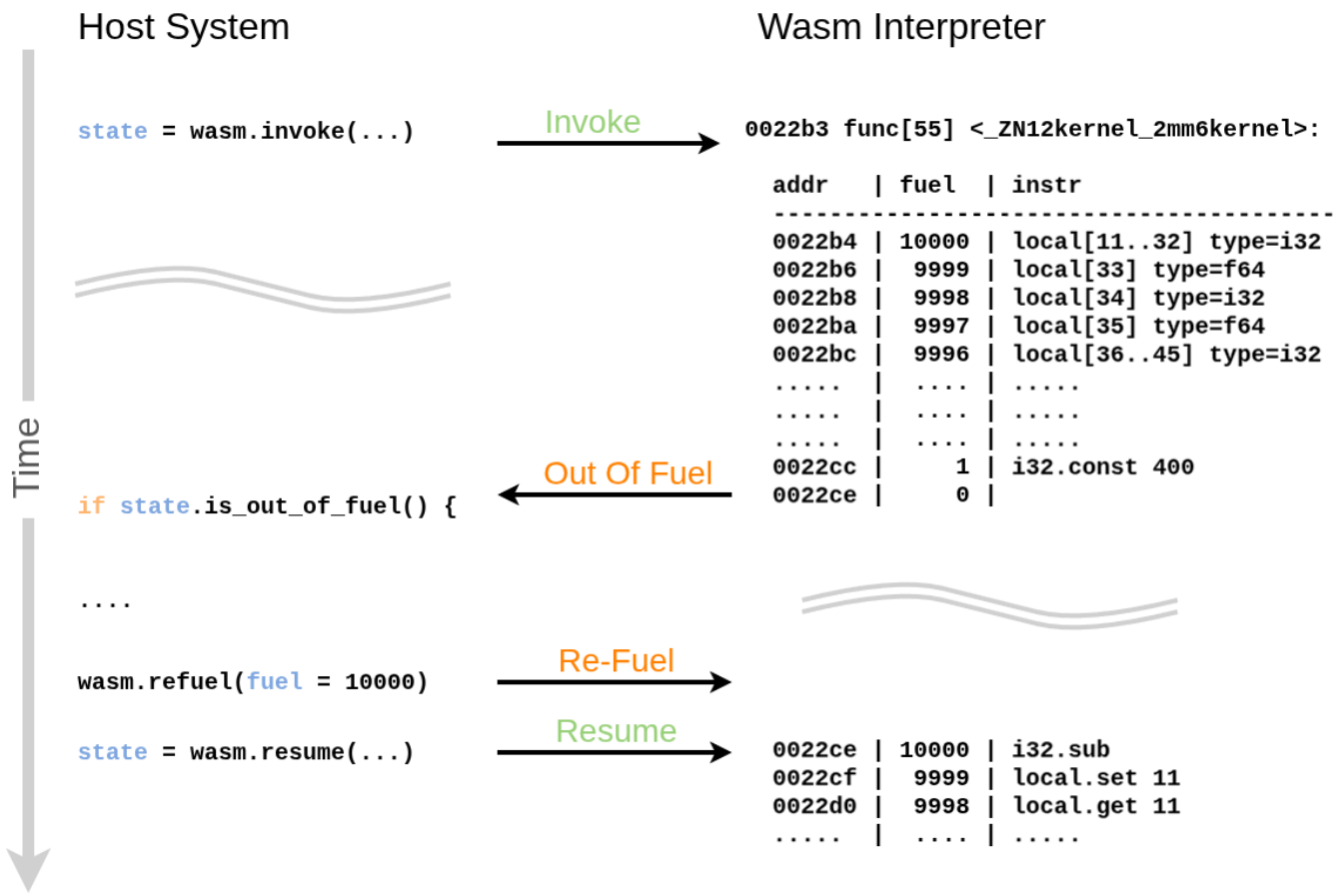
DYNAMIC MITIGATION WITH WEBASSEMBLY

WebAssembly

- Virtual Instruction Set Architecture
  - Not Programming Language
- Open, Formal Specification
- Strict Spatial Isolation
- Strict Type-Checking and Validation at Runtime

→ Potentially valuable for Avionics?

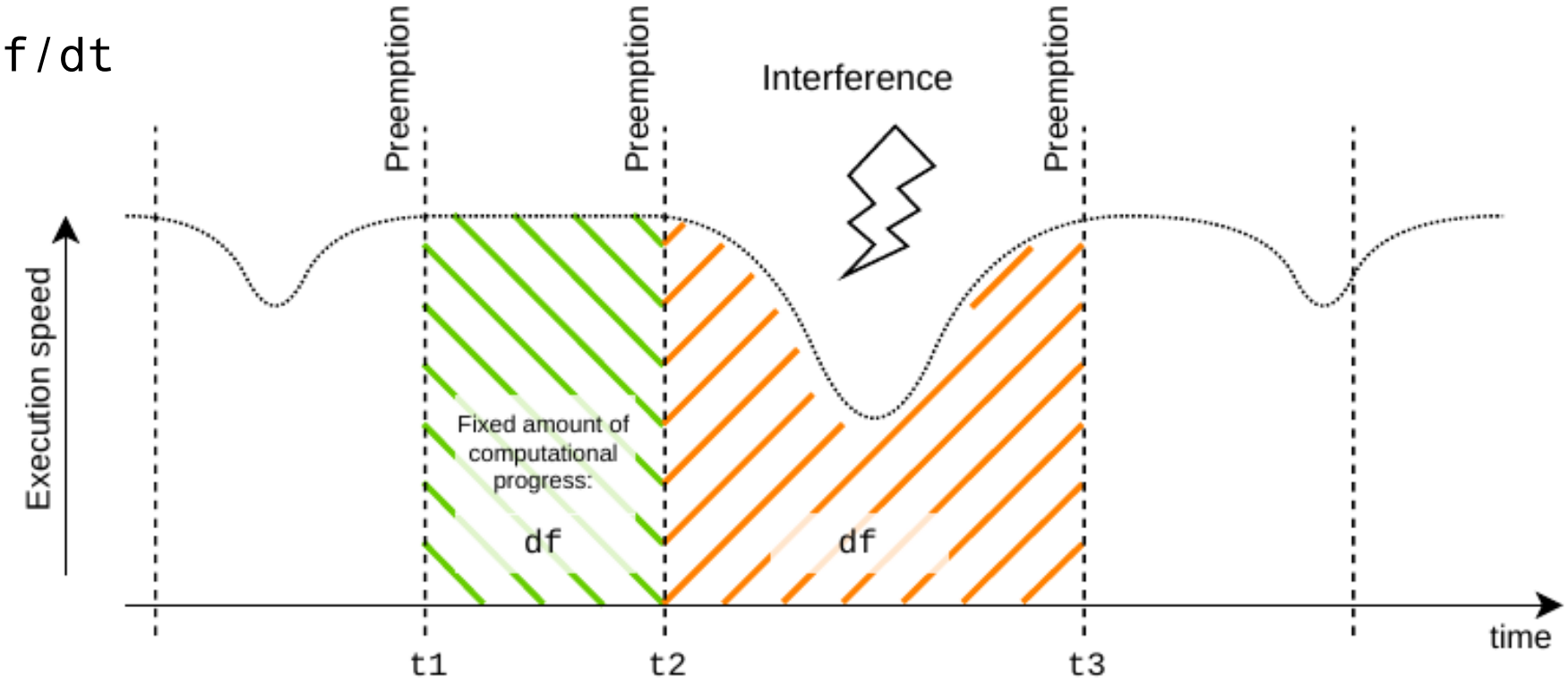
Fuel Mechanism



Dynamic Mitigation with WebAssembly

- Wasm Fuel Consumption equals Computational Progress

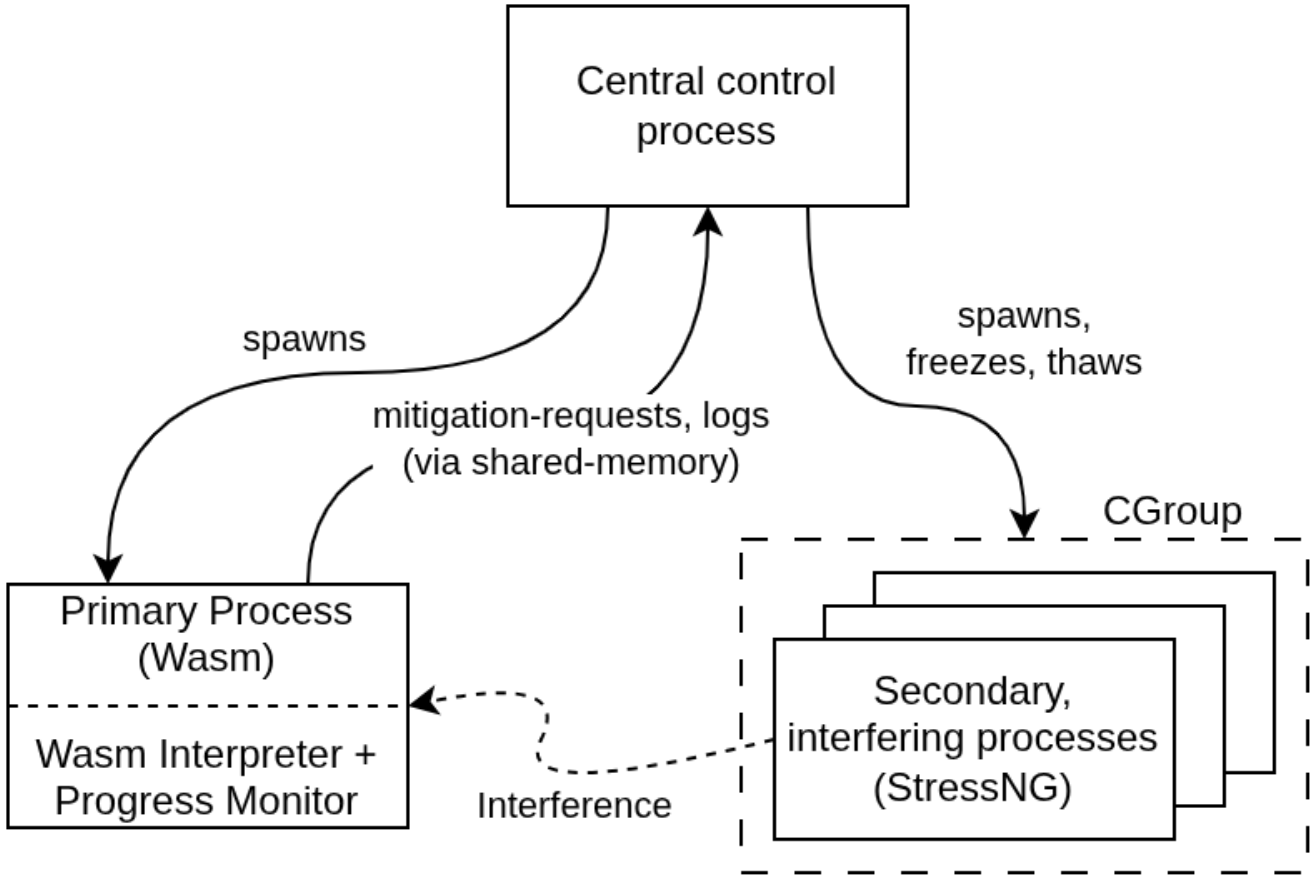
```
df := const. Amount of Fuel
throughput = df/dt
```



CONCEPT DEMONSTRATOR

Concept Demonstrator

- Linux based
- DLR Wasm Interpreter with a Resumable Fuel Mechanism
- StressNG for heavy interference



→ Goal: Show Qualitative Feasibility

EVALUATION

Evaluation

- Can WebAssembly Fuel be used to detect Multi-Core Interference?
- Is the Dynamic Mitigation Strategy effective?
- How efficient is the Dynamic Mitigation?

Evaluation

Detect Interference

- Empirical Evaluation of Computational Throughput

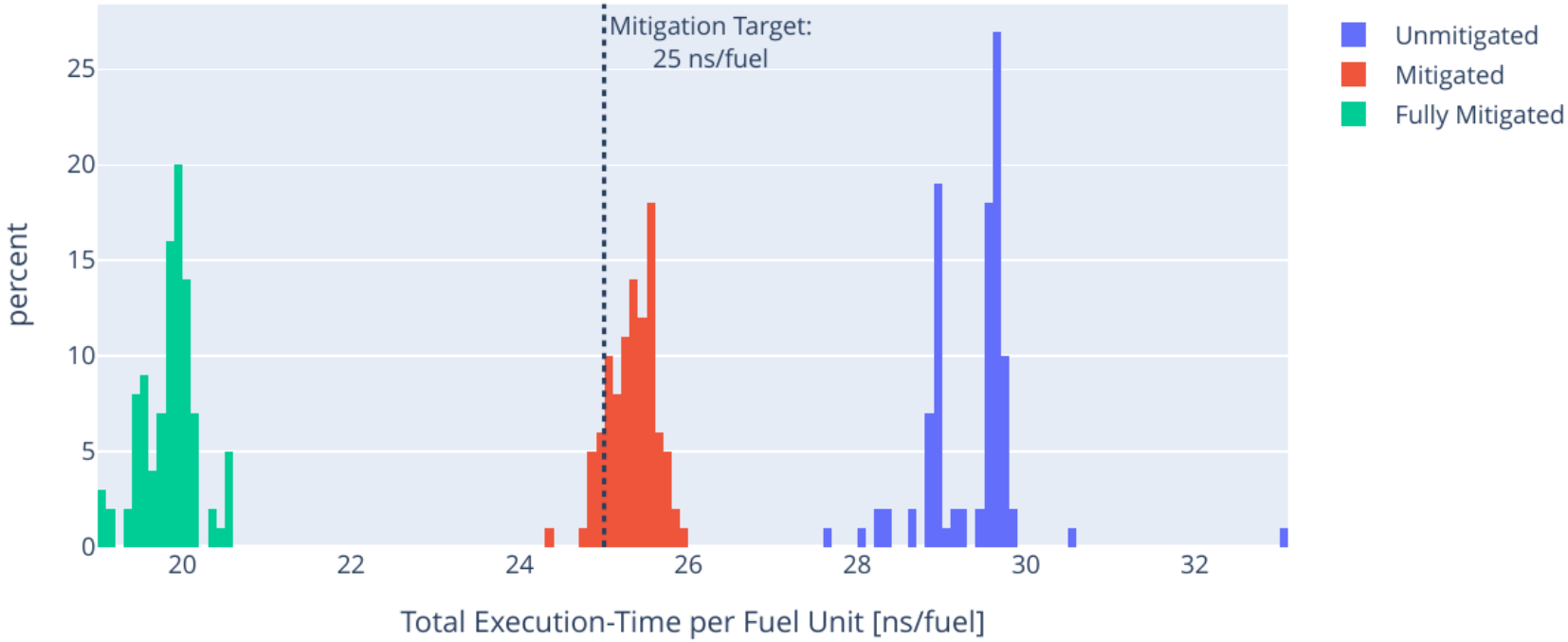
Description	Without Interference	With Interference
Mean Throughput	19.8 ns / fuel	29.3 ns / fuel
Median Throughput	9.8 ns / fuel	9.9 ns / fuel
Median Throughput, only Outliers	310 ns / fuel	610 ns / fuel

→ WebAssembly Fuel can be used to detect Interference

Evaluation

Efficacy 1 / 2

- Statistical Evaluation of N = 100 Runs

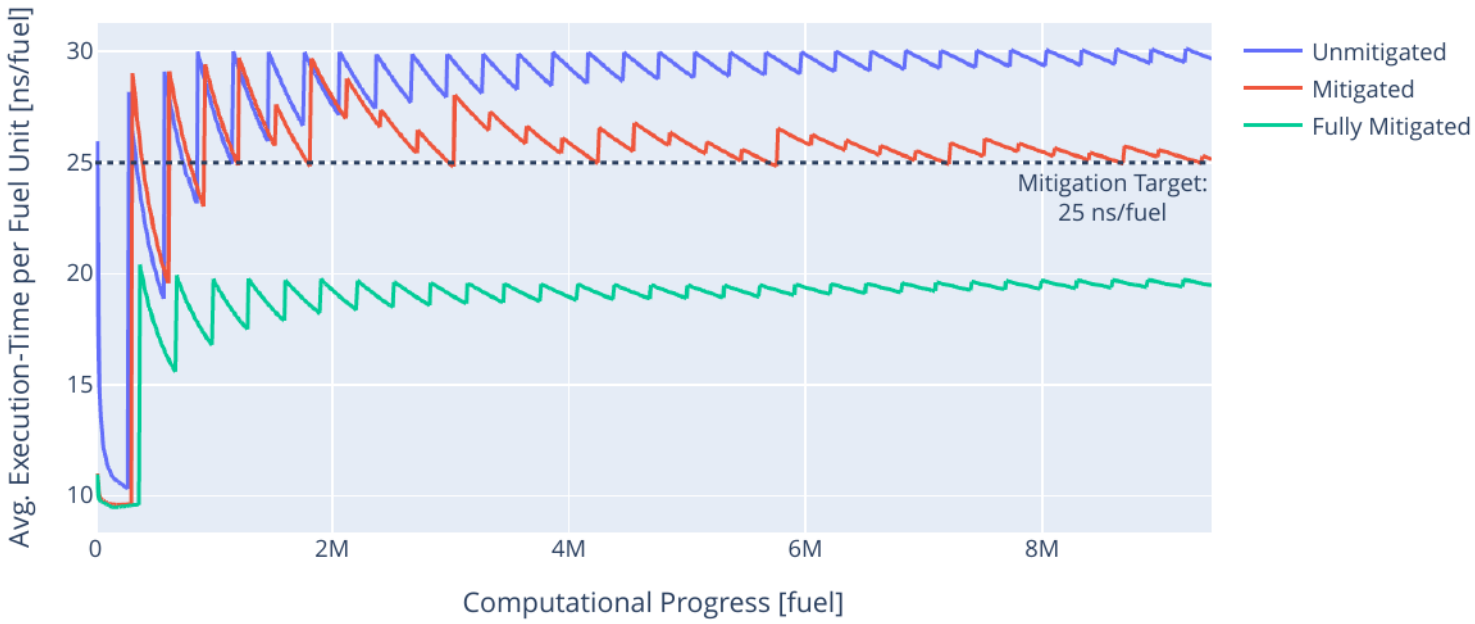


→ Dynamic Mitigation is effective

Evaluation

Efficacy 2 / 2

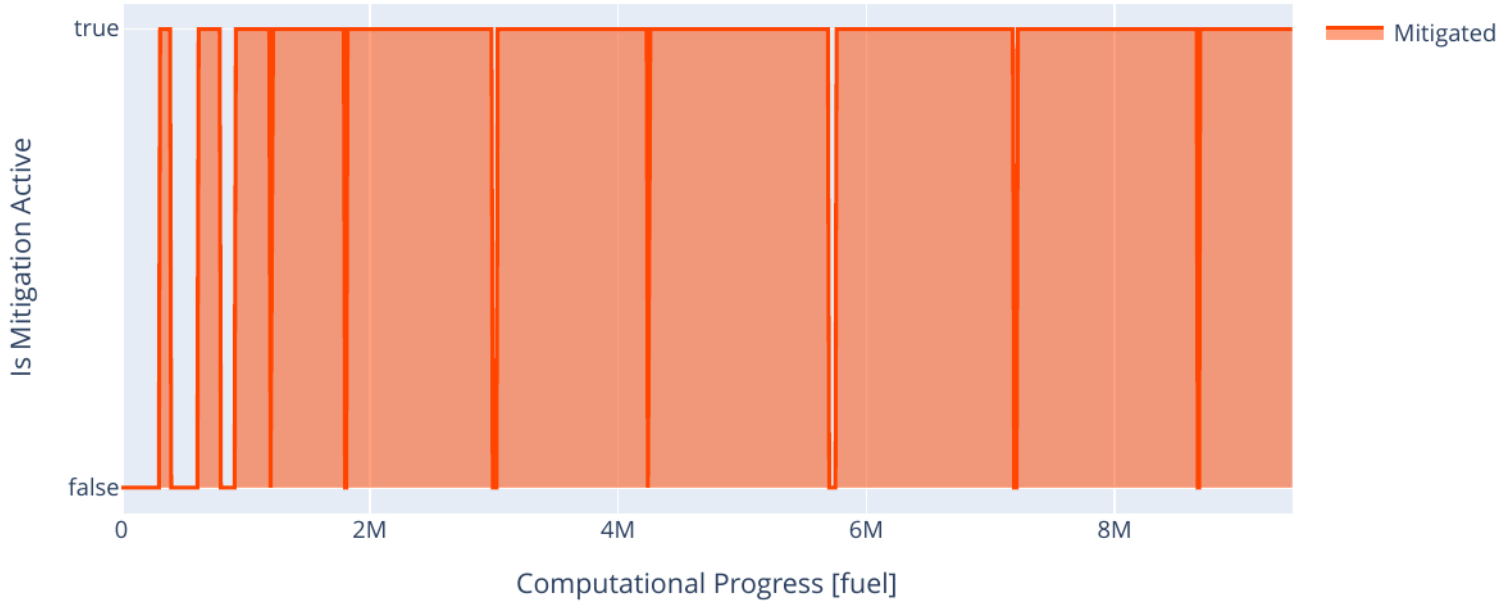
- Example: Single Run



Evaluation

Efficiency

- Mitigation Activity



→ Less than 10% of Compute on secondary Core remaining

LIMITATIONS & OUTLOOK

Limitations

General

- Significant Performance Impact due to Interpretation
- Only a single Task at any time can be considered “Critical”
- Interference originating from the Hardware (DMA, etc) can not be mitigated

Demonstrator Specific

- Assumption of 1 Wasm Instr. ~ 1 Fuel Unit not accurate
- Current Primary and Secondary Test Tasks not realistic
- Linux based Demonstrator not a suitable Testbed to answer Questions quantitatively

Outlook

- Dynamic Mitigation is promising, but much more Research is necessary
- Future Work:
  - Port Demonstrator to Bare-Metal setup with proper Tracing
  - Switch Test Tasks for more realistic, or more nuanced Tasks
- Next Research Requestion:

→ Does this Dynamic Mitigation Approach entail any Temporal Guarantees?

Imprint

Topic: **Dynamic Mitigation of Multi-Core Interference in Safety-Critical Systems**

Date: 2026-02-19

Author: Moritz Meier

Wanja Zaeske

Umut Durak

Institute: Flight Systems (Braunschweig)