Reconfigurable Partitioned Hardware Acceleration for Safety-Critical Applications

Excerpts from Vincent Janson‘s Master Thesis

Vincent Janson, **Wanja Zaeske**

Technische Universität Braunschweig

DLR

Agenda

Motivation

Background

Approach

Demonstrator

Conclusion

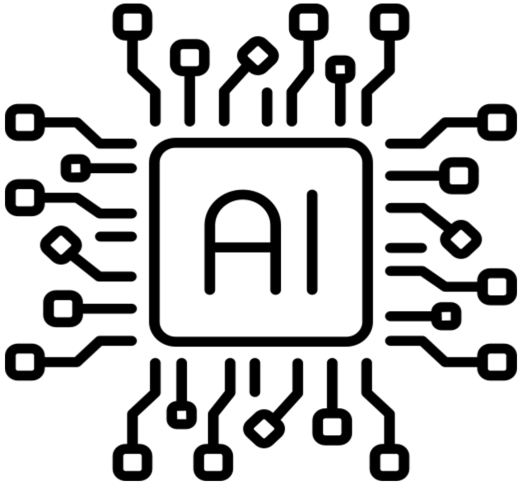
Use-Case: Transport Revolution

Geekwire.com using NASA Illustration | Lillian Gipson

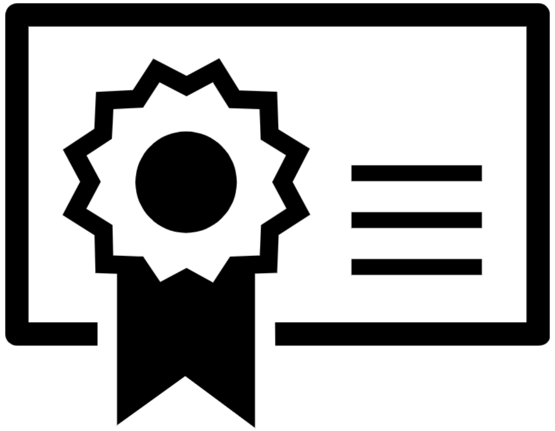
driveteslacanada.ca

Requirements & Wishes

www.vecteezy.com



flaticon.com: Triangle Squad

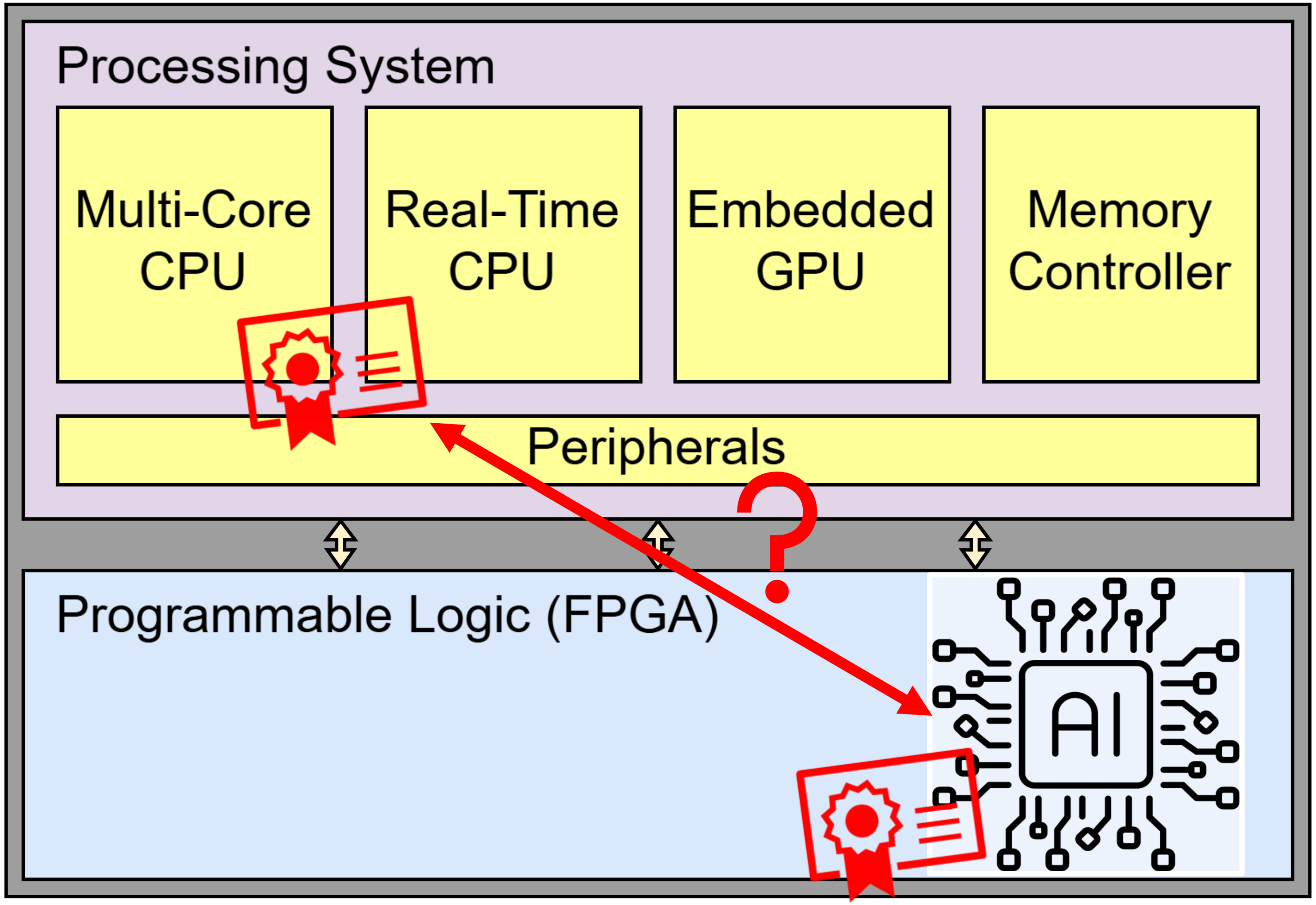


- ✓ Low-power computing platform with highest computing capacity
- ✓ Versatile platform

- ✓ Increase computing efficiency
- ✓ Simplify AI applications

- ✓ Utilize certifiable concepts/solutions
- ✓ Keep solutions simple
- ✓ Utilize by-default safety

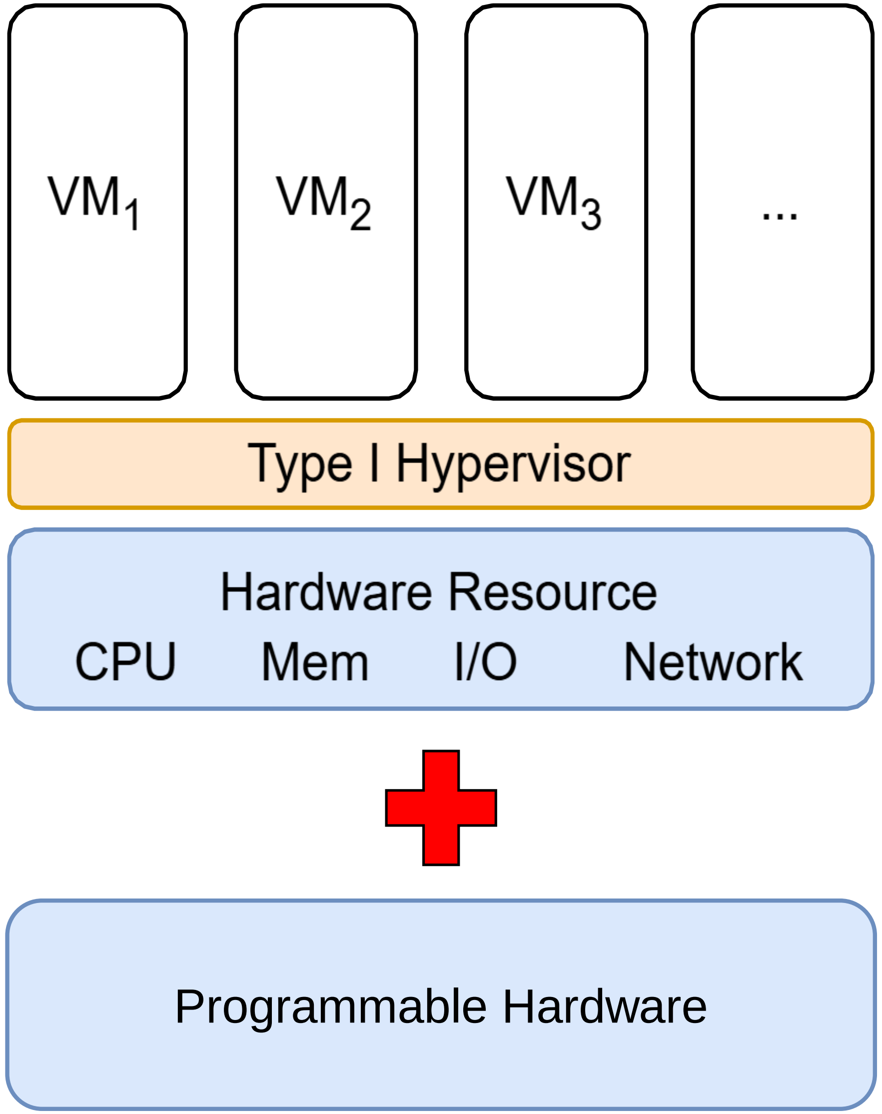
Technical Ingredients



Heterogeneous System-on-chip (SoC)

- CPU for classical software tasks
- Field Programmable Gate Array (FPGA) for programmable hardware acceleration
- **How to combine both?**

Technical Vision



Classical Approach

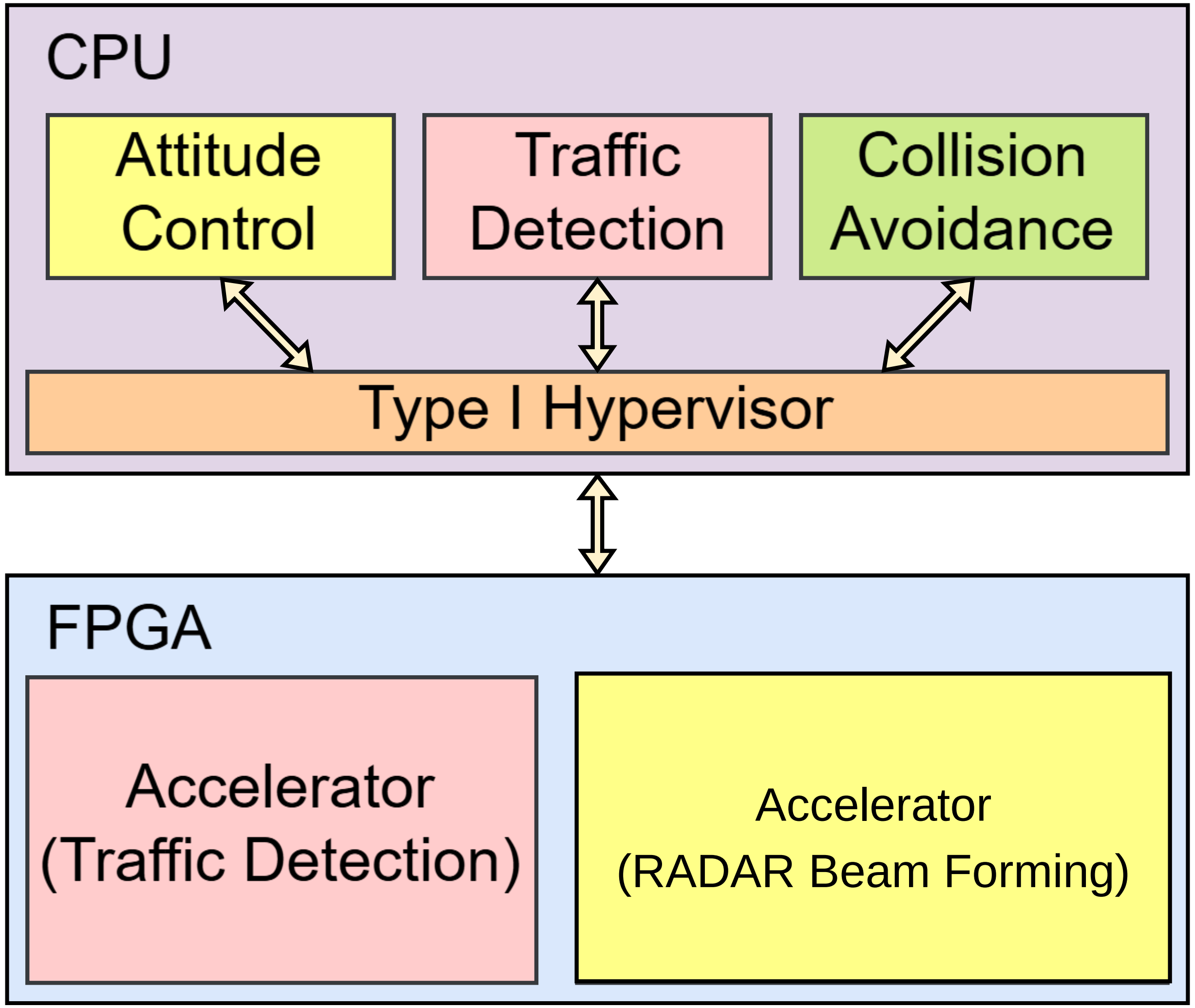
- Hypervisor/RTOS provides partitioning to applications for isolation/safety
- Partitioning as per DO-178C
- RTOS API as per ARINC 653

Idea

- Extend Hypervisor to use programmable hardware as additional resource

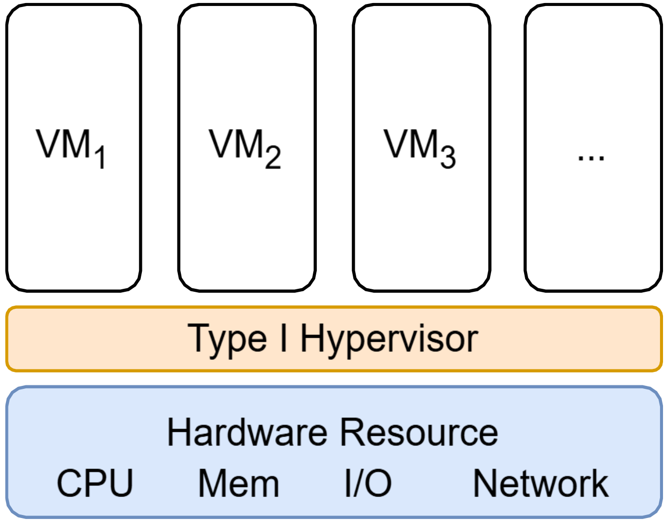
Example Architecture
Reconfigurable Computing Hypervisor

- *Heterogeneous SoCs* (Zynq UltraScale+) as computing platform
- *Hypervisor* for partitioning, application orchestration and programmable hardware interfacing
- Programmable hardware resources (*FPGA*) for hardware acceleration
- *Reconfiguration* of FPGA for adjustments in acceleration demand (different flight phase)



Agenda

Hypervisor – seL4



seL4 is a *microkernel* usable in a *hypervisor* configuration

- Lightweight (~ 10k SLOC, compiled into ~200KiB object code)
- Supports ARM v6-v8, RISC-V RV64 & x86 (32 & 64bit)
- High performance
- Implementation formally proven
- Past "academic exercise" status

https://sel4.systems/

seL4 – Microkit

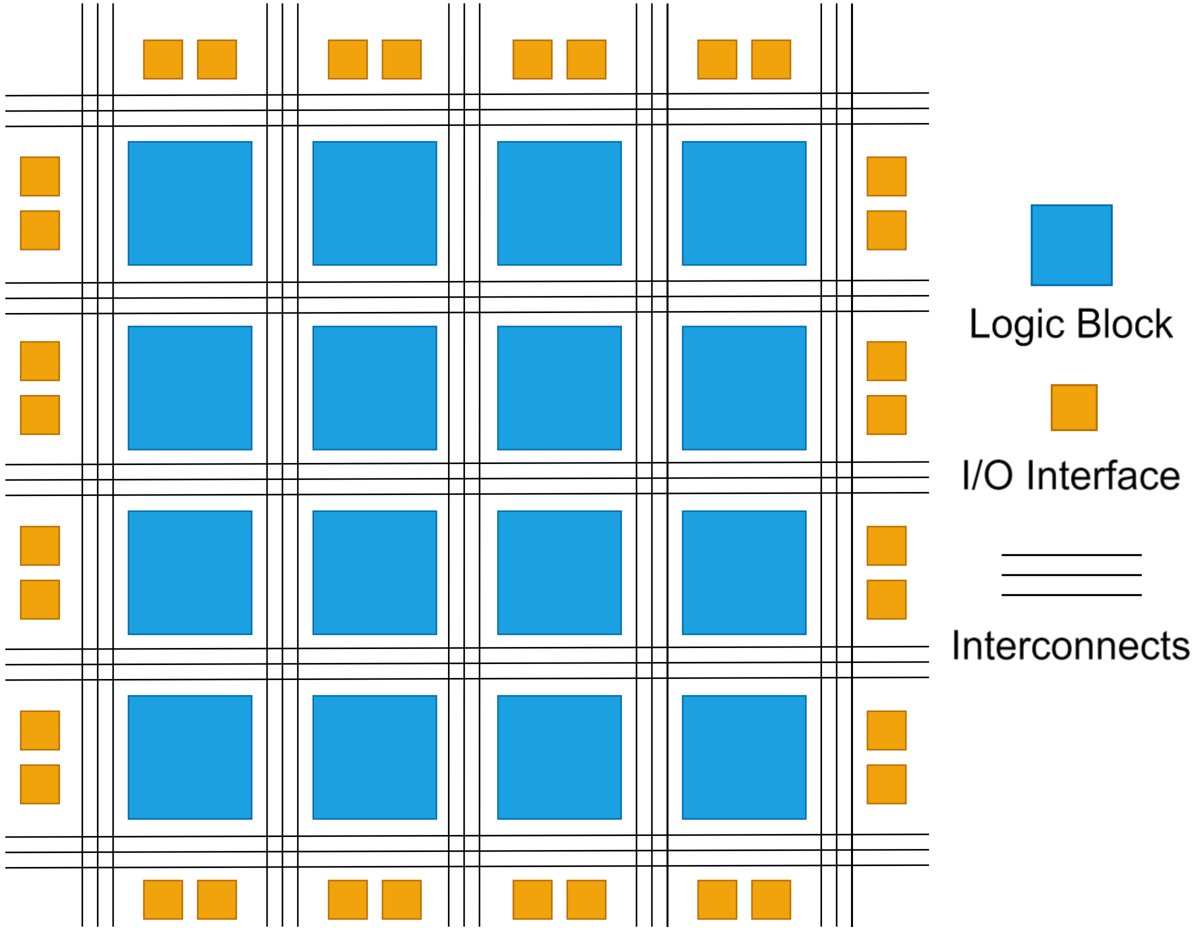
- Software Development Kit (SDK) for making seL4 easily usable
  - Introduces thin abstraction layer on top of seL4
  - Creates a bootable seL4 system image
  - Synchronous (*Protected Procedure Calls*) IPC
  - Asynchronous (*Notifications*) IPC

- System configuration through static configuration file – *System Definition File* (SDF)
  - Definition of Protection Domains (PD)
  - Memory regions and Memory Management Unit (MMU) on PD granularity
  - Inter-PD communication via communication channels
  - Interrupt handling and delegation to specific PD
  - Scheduling policy for PDs

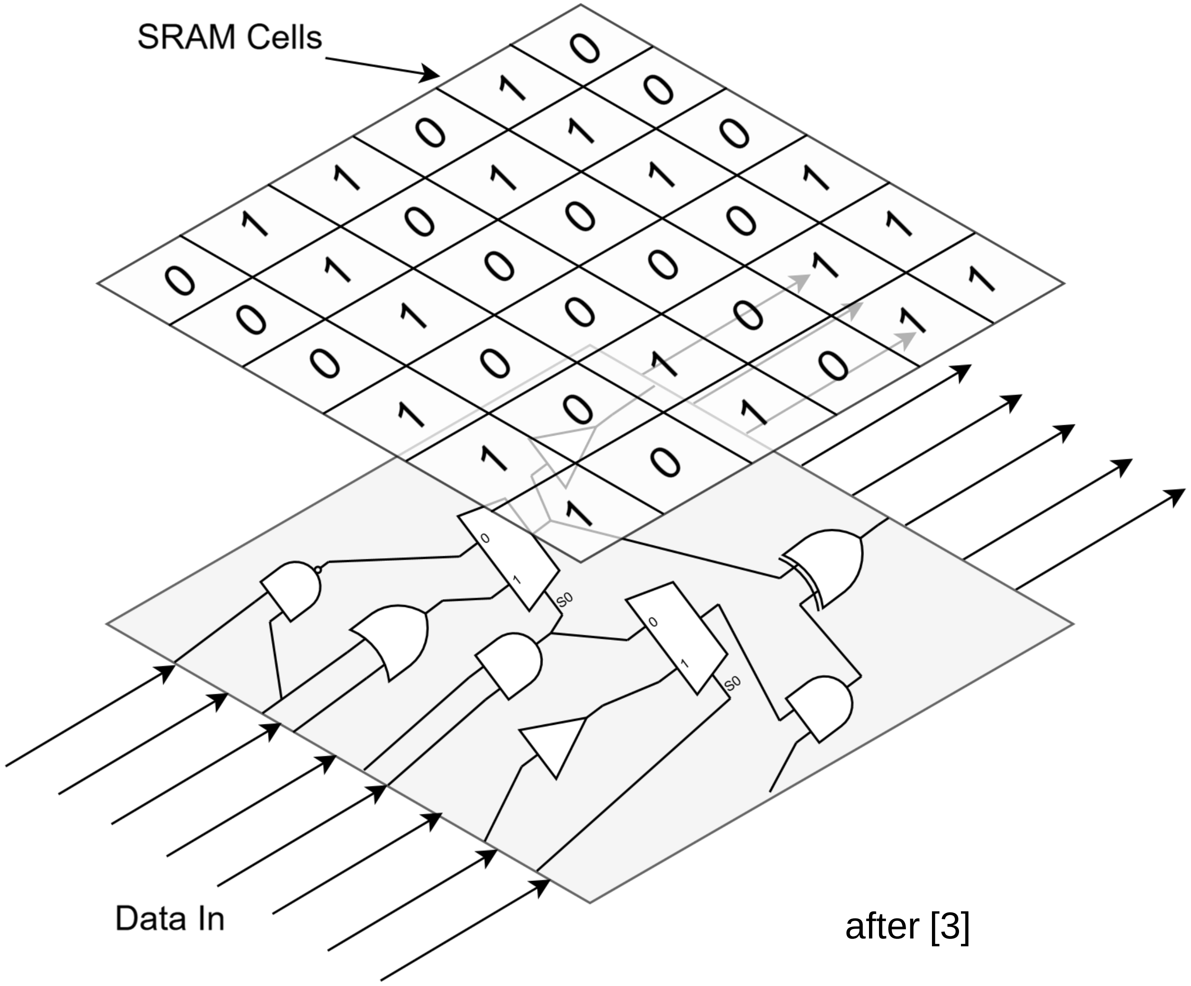
https://github.com/sel4/microkit

FPGAs in depth

- **Logic Blocks**
  - Fundamental building blocks of an FPGA containing combinational logic and registers
- **Interconnects**
  - provide wiring that connects various logic blocks



FPGA (Re-)Configuration



- **Configuration Memory**
  - (typically) SRAM memory cells
  - *bitstream* representing configuration is loaded into configuration memory

  →FPGA behavior determined by memory content

  →Change in memory = change of FPGA behavior / configured circuit

→ **Reconfiguration**
- Changing memory contents during runtime changes FPGA configuration!
- Modern FPGAs support partial bitstream reloading

Aviation Standards

- Guarantee *Safety* of Software/Hardware/System
  - Safety is represented as a probability of failure
  - *Design Assurance Levels* (*DALs*) depending on severity of failure
- Verification *Evidence* asserts Safety
  - Evidence amount is proportional to DAL

DAL	Severity
A	Catastrophic
B	Hazardous
C	Major
D	Minor
E	No Safety Effect

Aviation Standards

Hypervisor
- Dev. process
- HW/SW Interaction
- …

Hardware (PL) side
- Accelerators
- Management
- Reconfiguration
- …

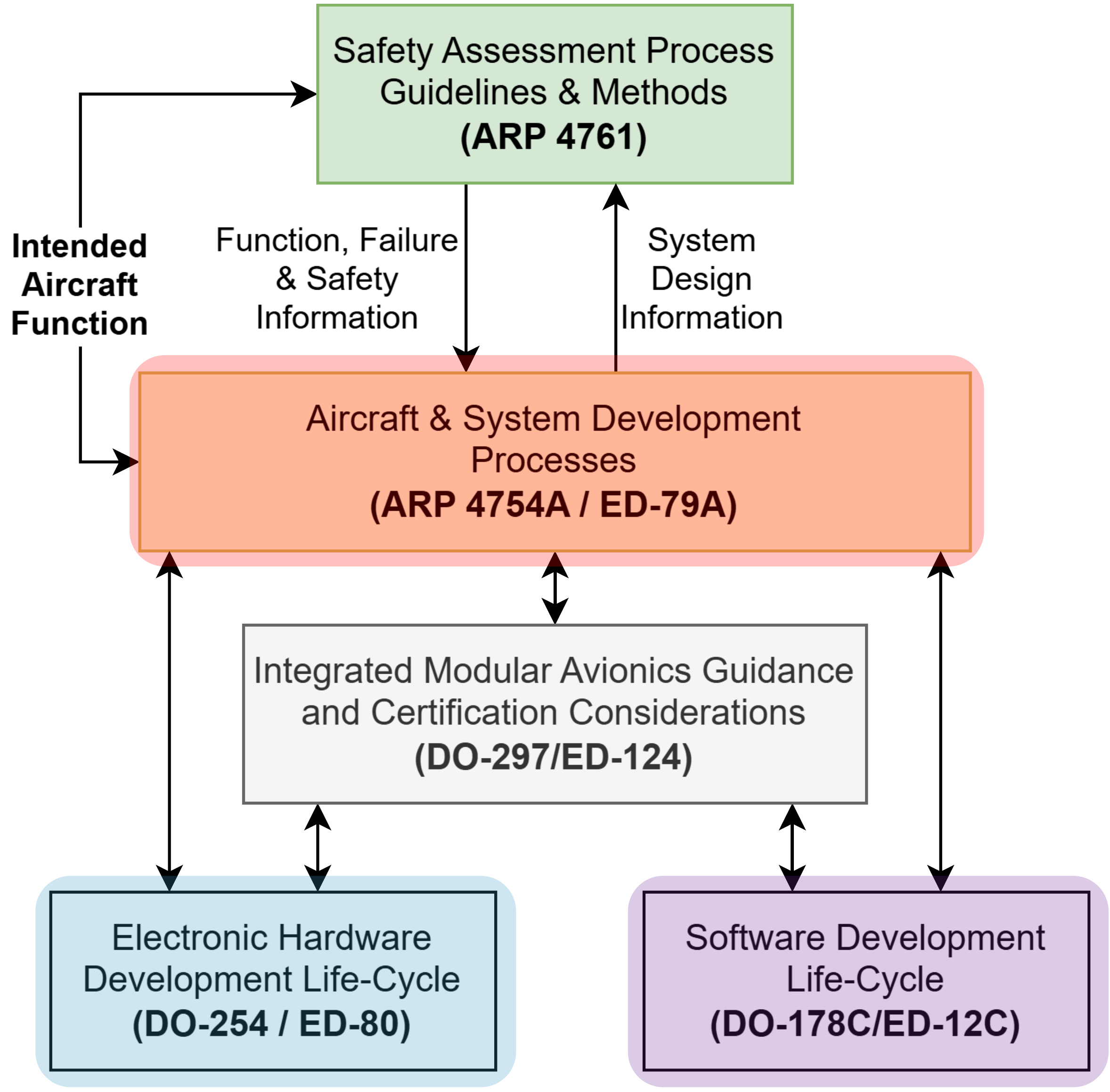


Figure is redrawn & adjusted from ARP 4754A

Software (PS) side
- seL4
- User applications
- Extension
- …

Agenda

Motivation

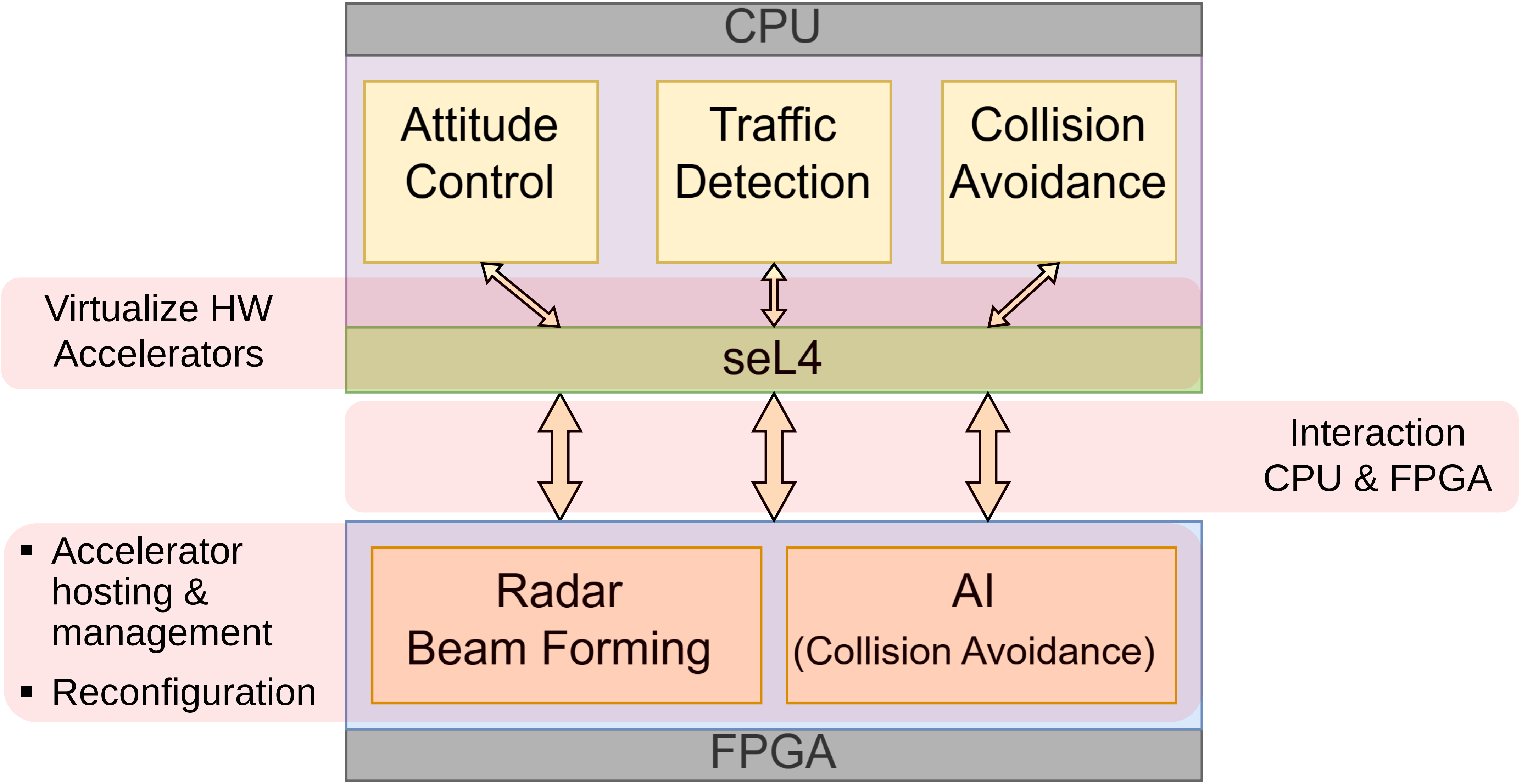
Background

Approach

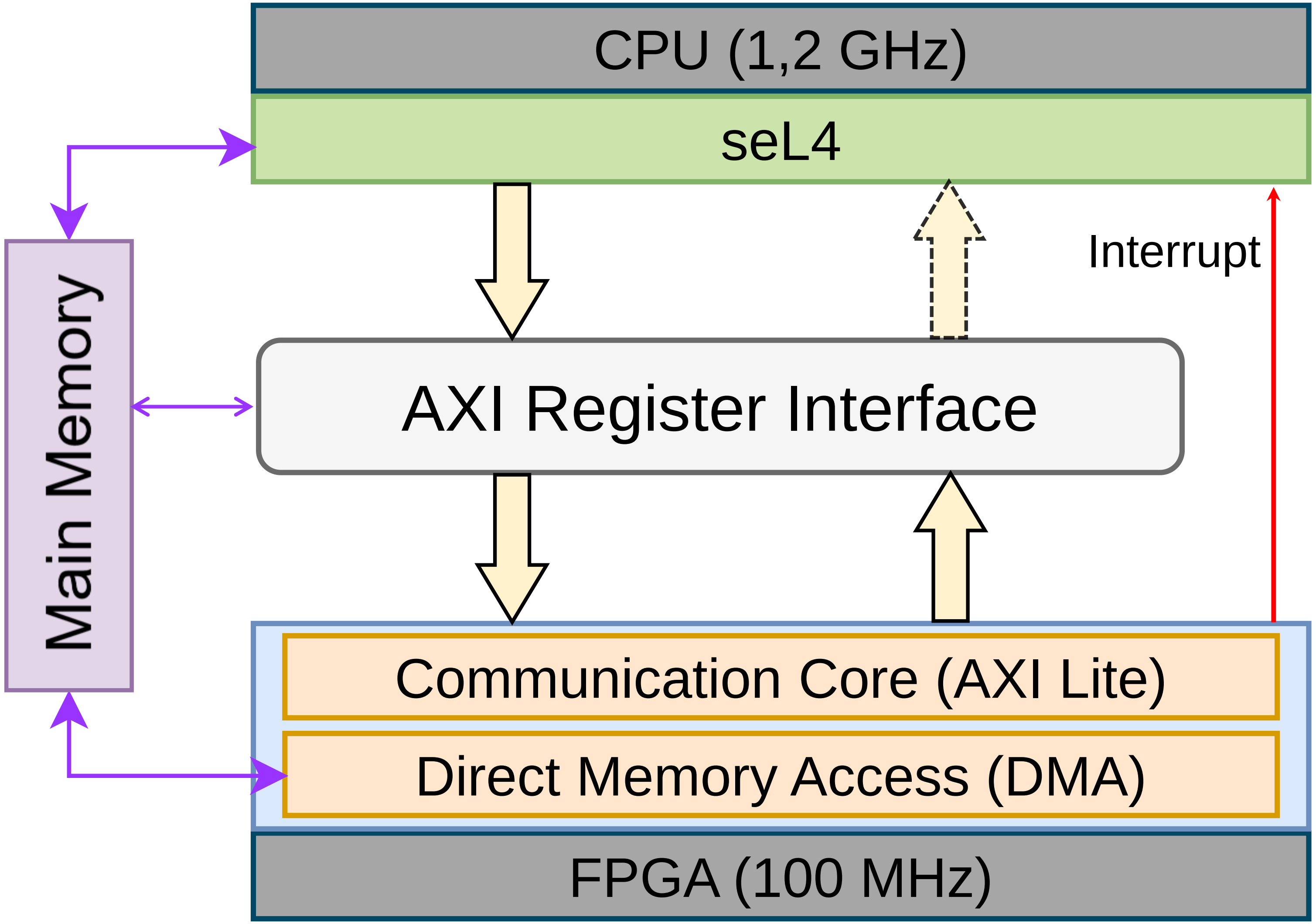
Demonstrator

Conclusion

Identifying Critical Aspects



Cross Domain Communication



Processing System – Design Questions

Technische Universität Braunschweig

DLR

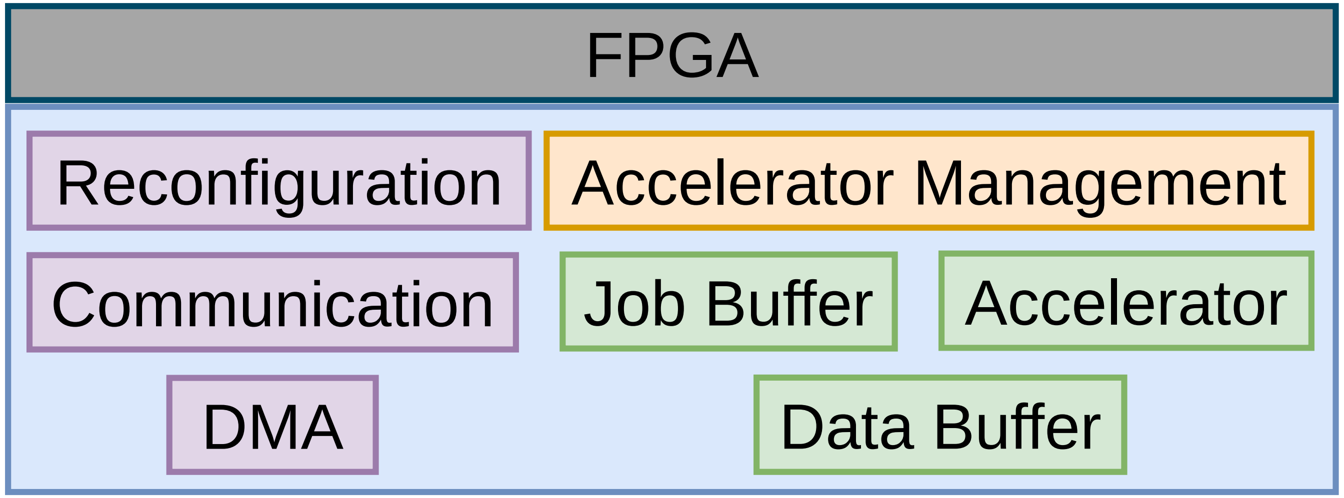
1. *How does a user PD access an accelerator core?*
   a) Directly – User PDs access accelerators directly at will
   b) Indirectly – Accelerators are accessed only via a HW accel. virtualizer

   Principle of least privilege / Resource partitioning

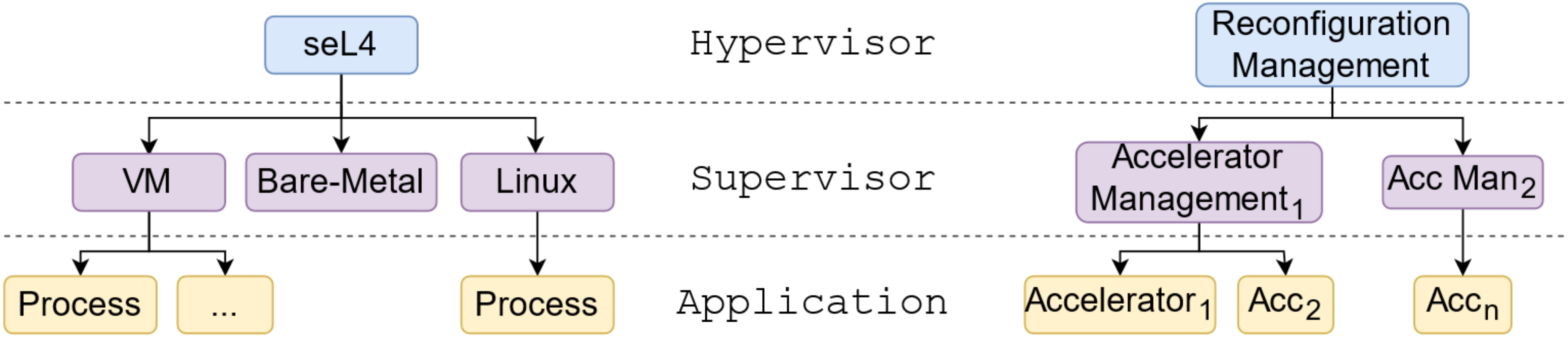
2. *How can this be implemented with seL4/Microkit?*
   a) Add functionality to seL4 kernel directly
   b) Outsource functionality to system partition/PD

   “A functionality is allowed in the kernel only if it cannot be provided in usermode”, Gernot Heiser

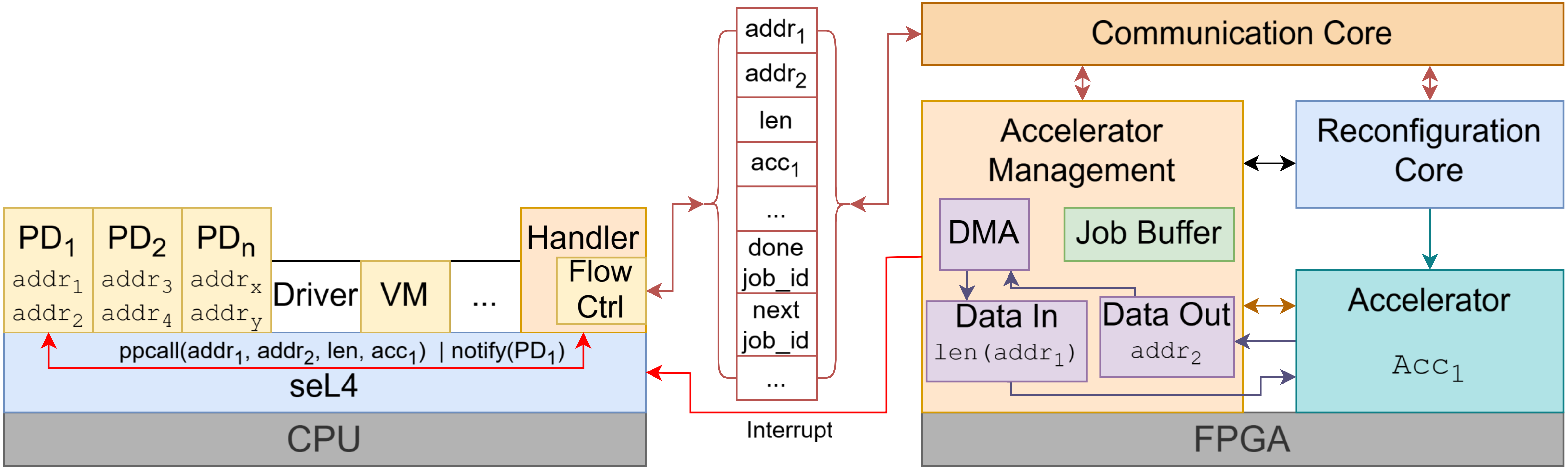
Programmable Logic – Components & Hierarchy



- Third party implementation
- Third party with adaption
  - Reconfig: ZyPR [2]
  - Comm: AXI Lite
  - DMA: Alex Forencich [3]
- Thesis implementation
  - Finite State Machine



Envisioned System Architecture



Agenda

Motivation

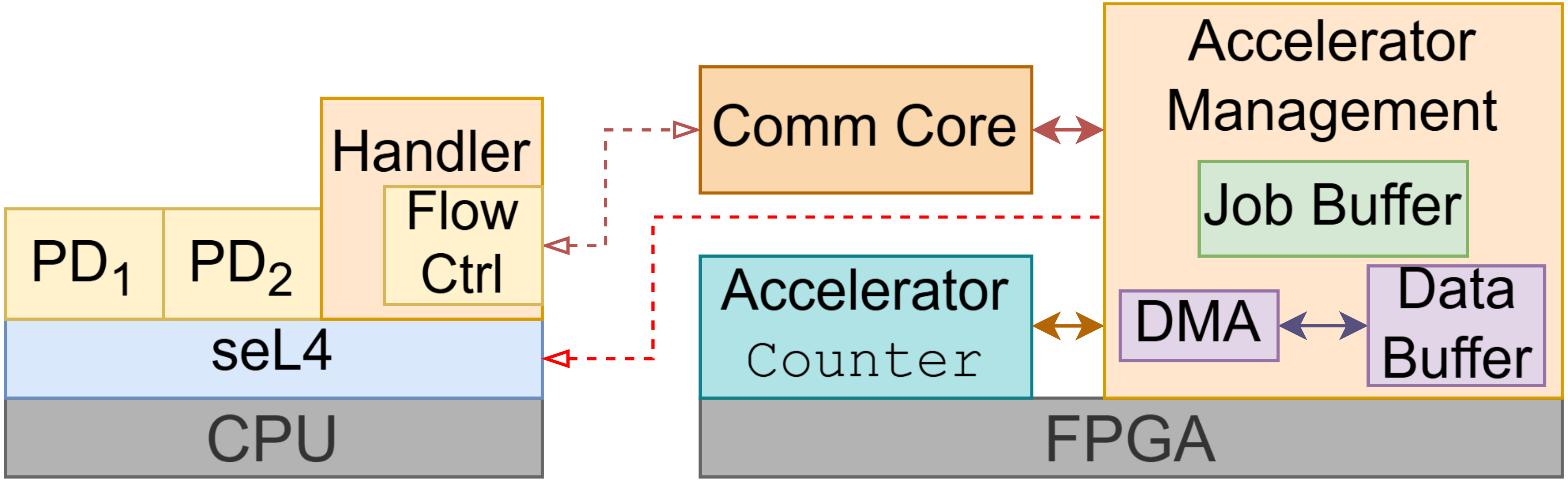
Background

Approach

Demonstrator

Conclusion

Demonstrator System Architecture



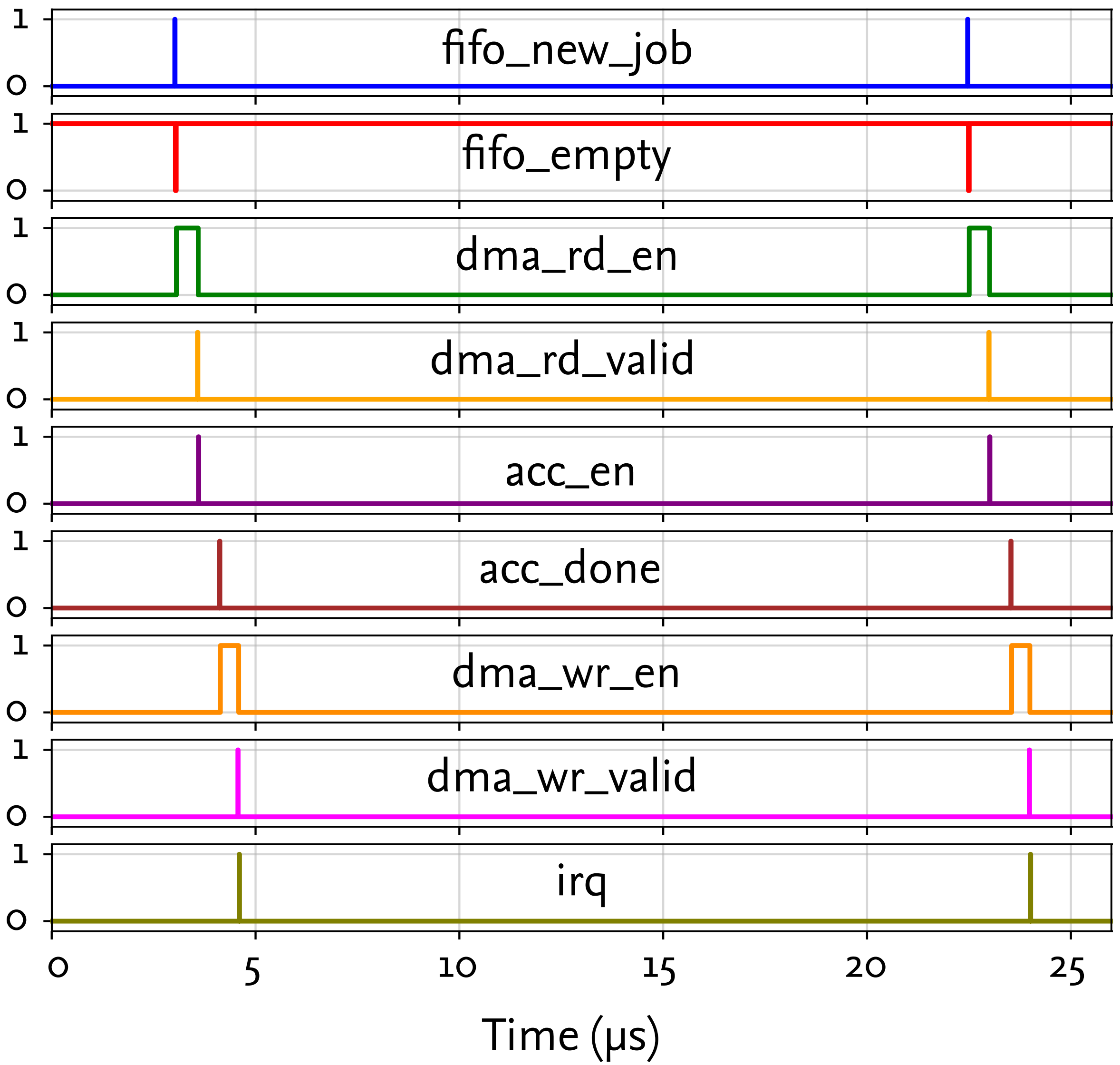
- Limitations
  - No HW accel. reconfiguration
  - Dummy accelerator → simple counter
  - Dummy PDs → just submit jobs/read results
  - DMA → only 64 bytes roundtrip through FPGA

Results

✓ Functional behavior as expected

× Non-functional behavior worse than desired:

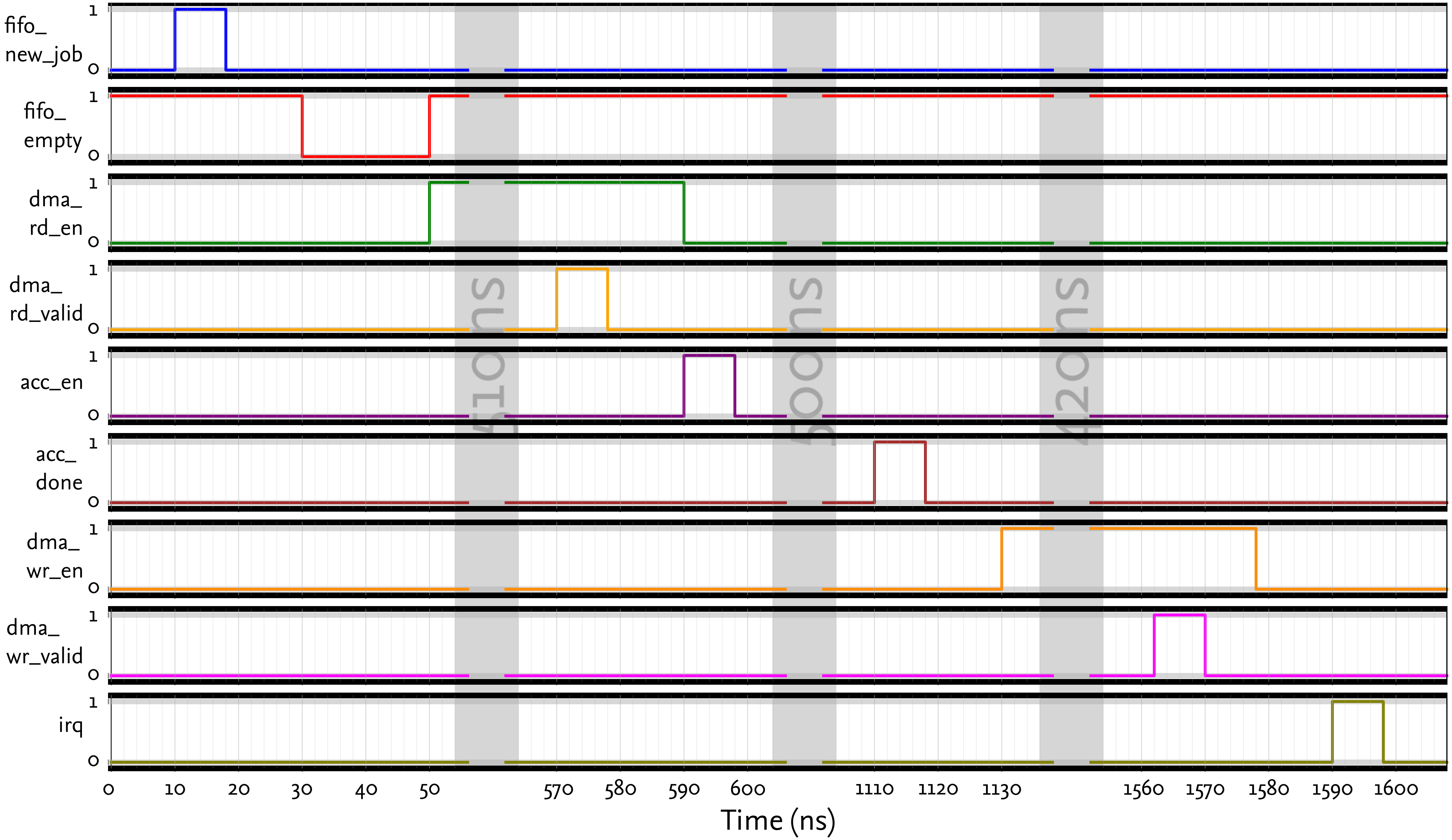
- Delay in PS to PL signaling (AXI register)
- Delay in PL to PS signaling (IRQ)
- Slow DMA read/write



Closer look at PL/PS Communication

Technische Universität Braunschweig

DLR



Vincent Janson, Wanja Zaeske, 3. Sep 2025

Lessons Learned & Demonstrator TODOs

- Synchronous debug printing is **slow**
  - UART-printf @ 115k2 baud caused 41 dB performance attenuation
- Significant overhead per PL/PS communication
  - Larger DMA transfers are advisable
- Microkit is excellent for experimentation
  - Reasonably simple implementation
  - ~ 1.1k SLOC for entire PS
- Logic analyzer is indispensable to debug real-time PL/PS interaction

- TODO
  - Proper benchmarking & performance tweaking
    - Current DMA implementation reaches ~120 MiB/s, but 380 MiB expected
  - Integration of partial bit-stream reloading for PL runtime reconfiguration

Agenda

Motivation

Background

Approach

Demonstrator

Conclusion

Conclusion

Takeaways

- Hypervisor principles are applicable to HW accelerators
- Microkernel approach compatible with use-case
- High performance PL/PS interfaces are tricky

Achievements

✓Low-Latency PL & PS interaction
  - Min roundtrip per accelerator job <4µs

✓Virtualizer for HW. Accelerators

✓Partitioned in HW and SW
  - Aiding certification

✓PL-Reconfiguration prepared
  - Implementation pending

Literature

[1] T. Xia, Y. Tian, J.-C. Prévotet, and F. Nouvel, "Ker-ONE: A new hypervisor managing FPGA reconfigurable accelerators," Journal of Systems Architecture, Article vol. 98, Sep 2019, doi: 10.1016/j.sysarc.2019.05.003.

[2] Li, Xiangwei & Fei, Cheng & Maskell, Douglas. (2018). FPGA Overlays: Hardware-based Computing for the Masses. 10.15224/978-1-63248-144-3-12.

[3] Clive Maxfield. The Design Warrior’s Guide to FPGAs: Devices, Tools and Flows. Newnes, USA, 1st edition, 2004. ISBN: 0750676043.

[4] Gernot Heiser and Kevin Elphinstone. L4 Microkernels: The Lessons from 20 Years of Research and Deployment. ACM Trans. Com-put. Syst., 34(1), apr 2016. ISSN: 0734-2071.DOI: 10.1145/2893177.