

Easing the path to deployment in ML4Sys through FPGAs

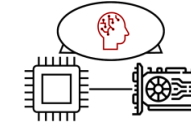
Vision Talk

Maximilian Jakob Heer, Benjamin Ramhorst, Gustavo Alonso
{maximilian.heer, benjamin.ramhorst, alonso}@inf.ethz.ch



Overview – what to expect from this talk

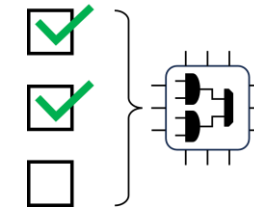
1. ML4Sys – entering a new era of computer systems



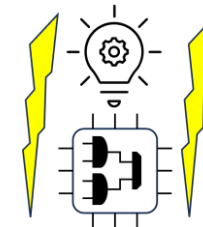
2. FPGA-shells + ML-compilers: A match made in heaven



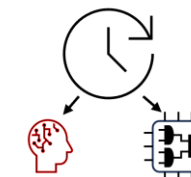
3. ML4Sys-challenges and how we solve them on FPGA



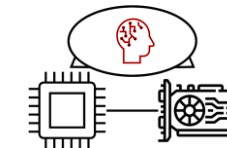
4. Promising use cases for ML4Sys on FPGAs



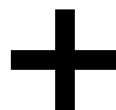
5. What's gonna happen next – a glimpse of the future



ML4Sys – entering a new era of computer systems



The Era of Machine Learning
Increasingly powerful ML-models
ubiquitous in any domain



The Era of Complex Systems
Ever more complicated computer systems
for more complicated applications

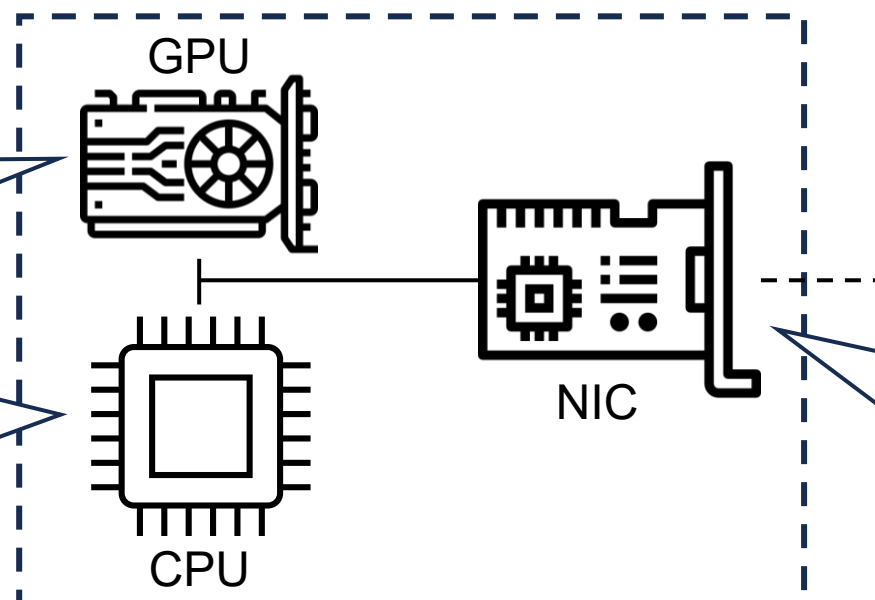
Use Machine Learning to enhance the capabilities of Computer Systems (**ML4Sys**) and make them *adaptive to their own workloads, reduce complexity and get rid of the demands of manual optimization.*

ML used for...

- ... power management,
- ... fast workload modeling for more efficient task allocation,
... and much more.

ML used for...

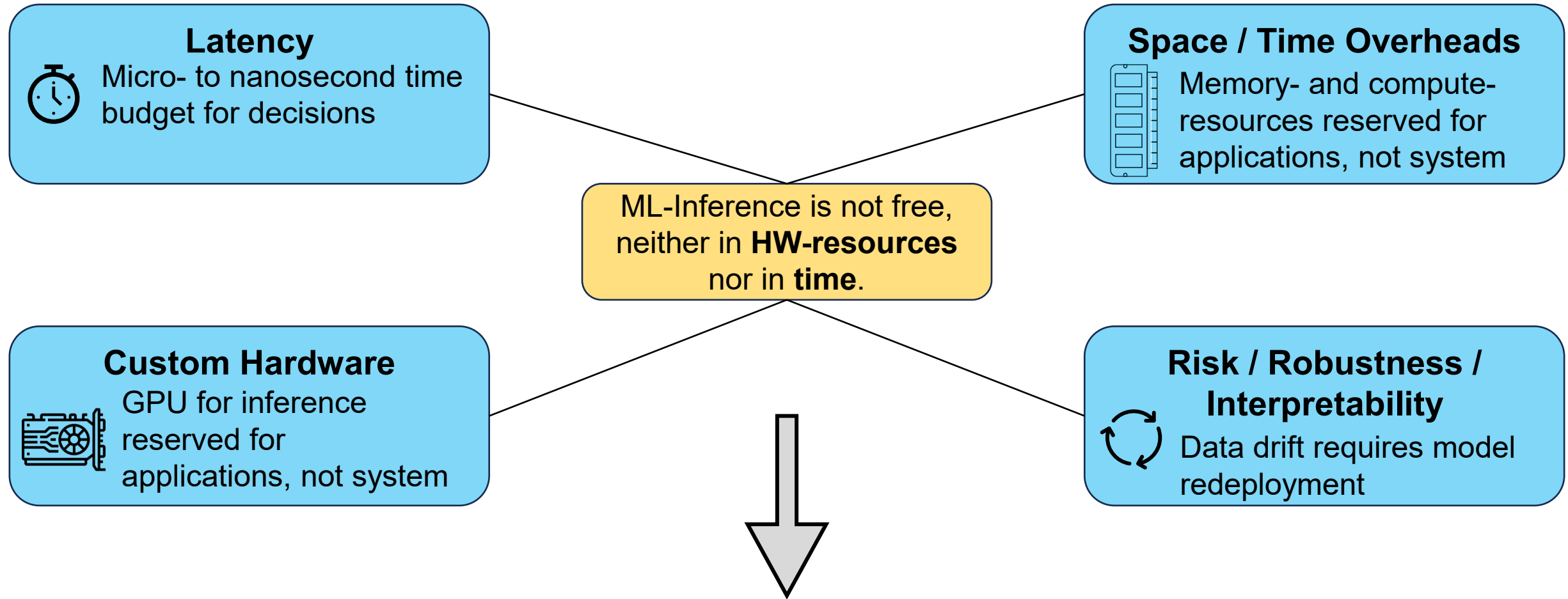
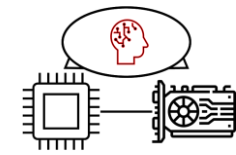
- ... computer architecture and chip design (EDA),
- ... branch predictions,
- ... cache replacement and prefetching policies,
- ... garbage collection,
... and much more.



ML used for...

- ... fast and precise network modeling,
- ... efficient congestion control and load balancing in multi-path environments,
- ... traffic scheduling,
... and much more.

ML4Sys sounds great – what's missing?

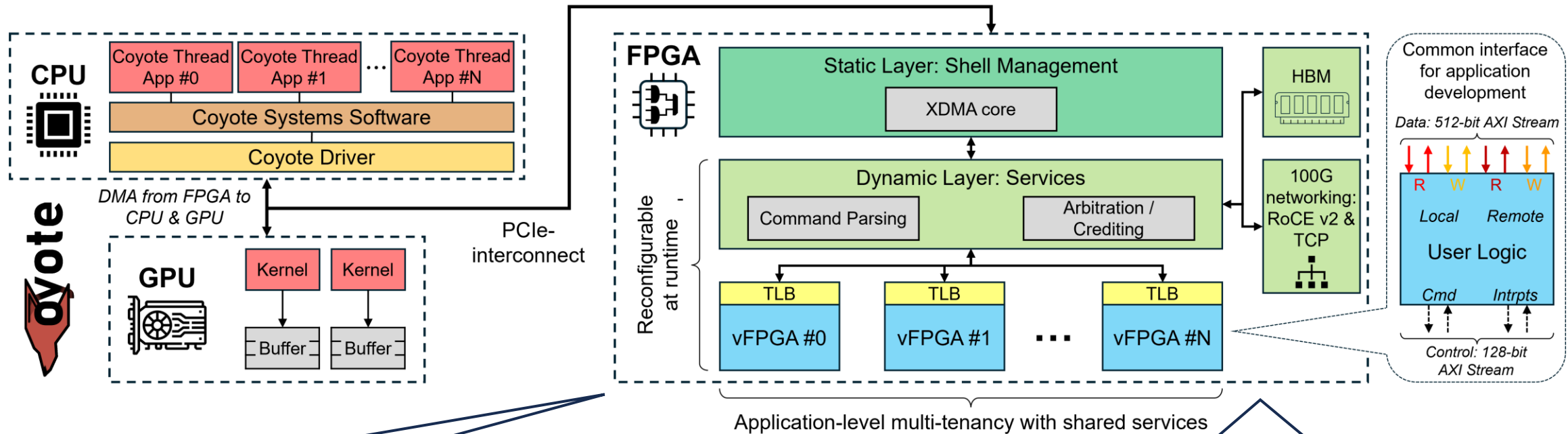


What if ultrafast ML-inference could be integrated right into an existing system directly on the accelerator with only minimal resource overhead and easy replacability?

Coyote: Setting the stage for ML4Sys on FPGAs



What if we could have a complete Computer Systems to our hands on FPGA, with every single aspect up for customization?



Completeness: Coyote covers all relevant aspects of a computer system:

- Multi-Tenancy
- Networking (RoCE v2)
- Memory Management

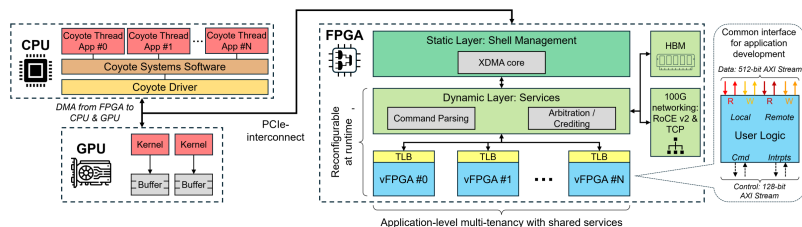
Accessibility: All system components are open for customization by the user

- Performance Separation
- Network Management
- Arbitration & Scheduling

Reconfigurability: During deployment, the different parts of the FPGA-shell can be swapped out „hot“.

FPGA-Shells and ML-compilers: A match made in heaven

FPGA-shells (Coyote v2 etc.)



Advantages of FPGA-shells for ML4Sys:

- **Feature-completeness** for computer systems.
- **Directly exposed interfaces** for customization.
- FPGA-fabric for **both system components** and **ML-inference**.

ML-compilers (hls4ml, FINN)



Advantages of FPGA-ML-compilers:

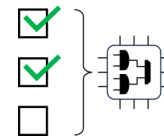
- **Ultra-fast ML-inference**.
- **Compatible** with High-Level ML-frameworks.
- **Ressource-efficient** implementation of ML-inference.
- **Adaptive to various interfaces** in hardware.



Repurpose ML-compilers from edge deployments to systems research to make best use of beneficial properties in both domains.


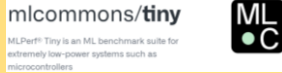
→ FPGAs become the **premier platform** for ML4Sys-research due to these compilers


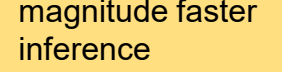
ML4Sys-challenges and how we solve them on FPGAs



Latency

ML-inference on FPGAs allows for ultra-low latency.

FPGA  →  MLPerf Tiny is an ML benchmark suite for extremely low-power systems such as microcontrollers.

Embedded CPU  →  Orders of magnitude faster inference

Typical system times:

- DRAM: 50-100 ns
- Page Walks: 100-400 ns
- PCIe: 80-150 ns

Space Overhead

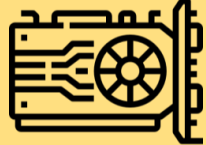
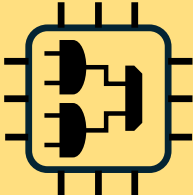
FPGA-shells leave plenty of room for ML-models, ML-compilers are resource-efficient.

 10-20% overall resources

System-relevant ML-models are small and fit within the shell.


Custom Hardware

The FPGA-fabric replaces the dedicated GPU.

 → 

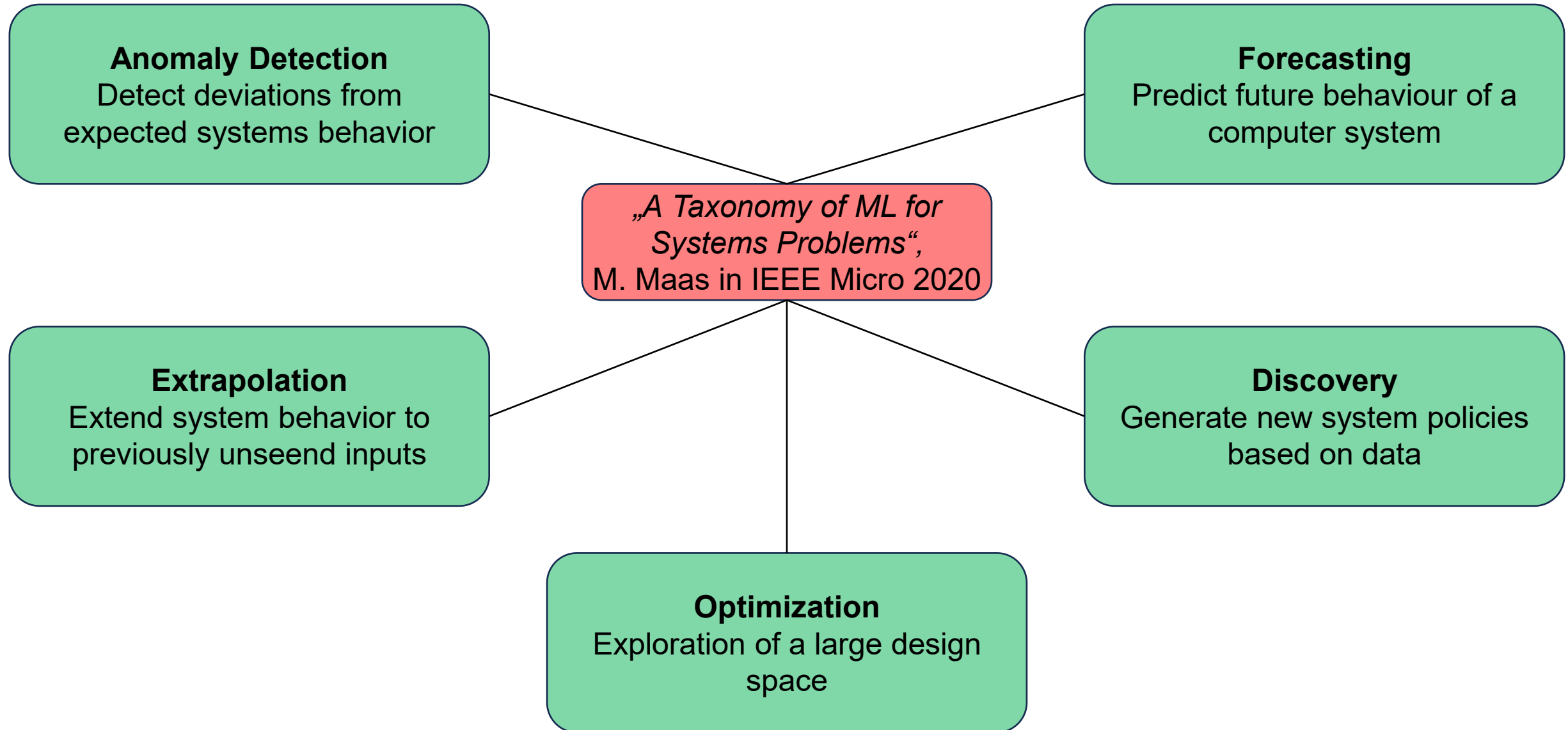
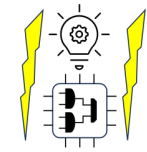
Distribution Shifts

Partial reconfiguration on FPGAs allows to quickly swap out deployed ML-models.

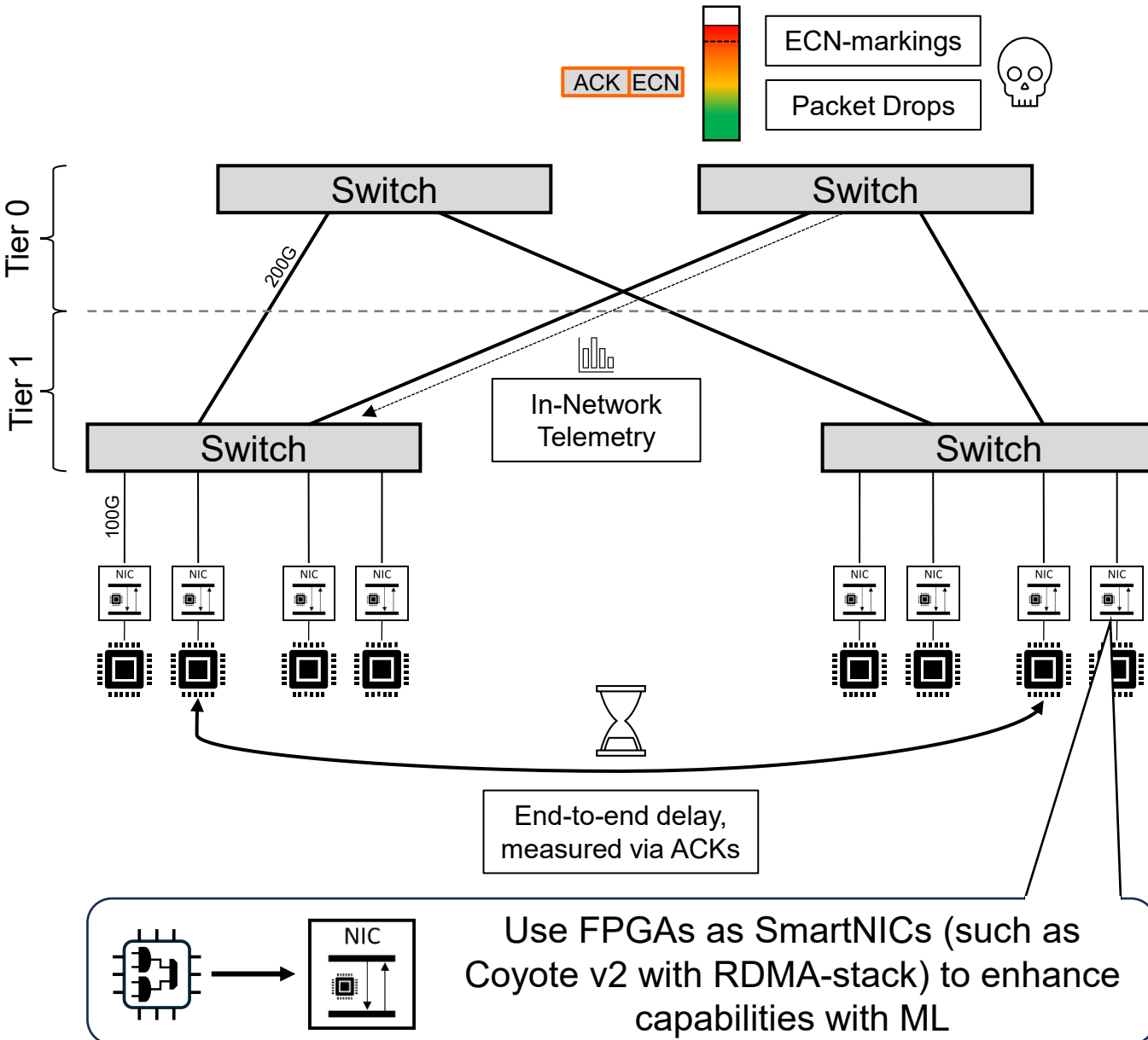
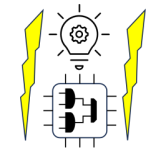
 57ms per partial reconfiguration

Fast enough for hot-swapping of deployed ML-models in the system.

Application fields of ML in Systems



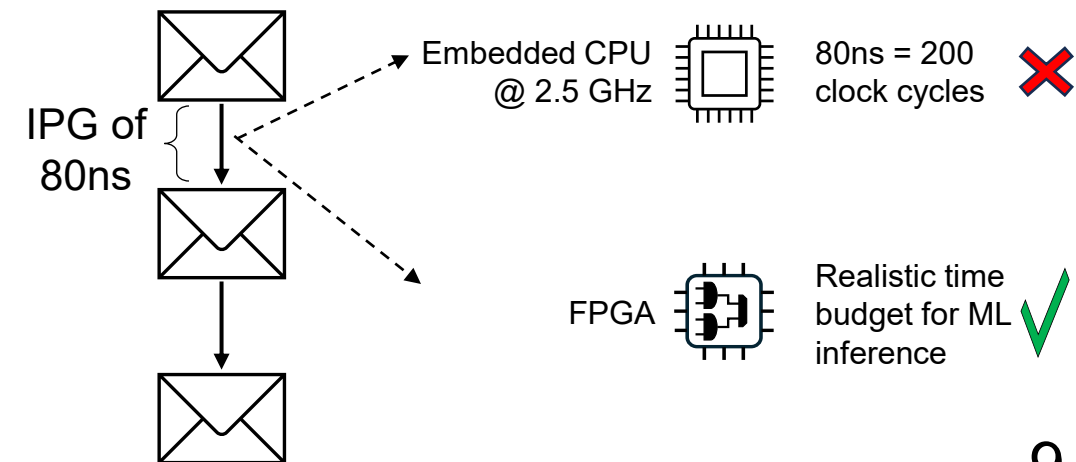
Forecasting: ML-based Congestion Control



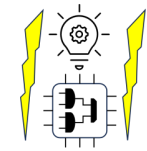
Congestion Control is one of the key problems of datacenter networking:

- Based on network signals, control the pace of packets released from a NIC to the network.
- Trend #1: Include ever more multi-modal congestion signals.
- Trend #2: ML seems to be a good fit for the problem.

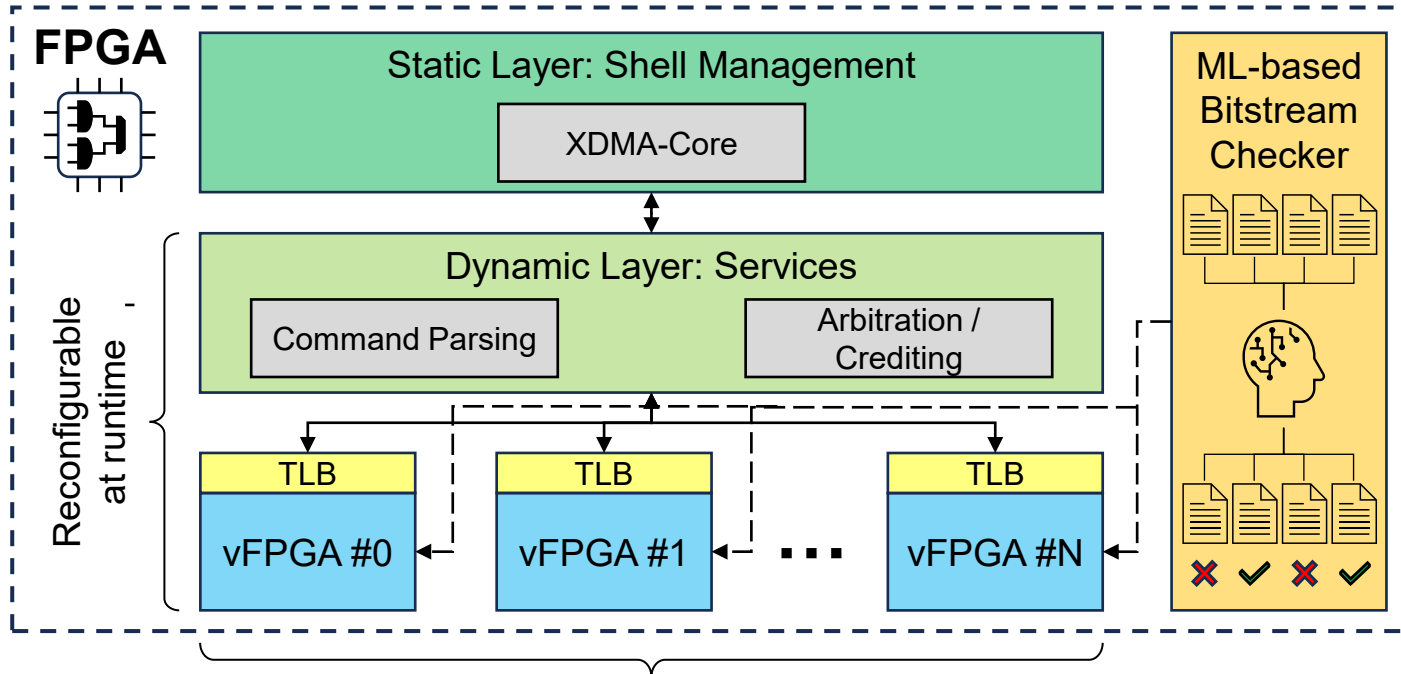
Example of 400G high-performance traffic with 4k MTU packets:



Anomaly detection: ML-based safe multi tenancy






Why didn't FPGAs „as-a-service“ in the datacenter become more popular so far?



ML can effectively screen bitstreams for potential threats.
BUT: Inference runs on external hardware.

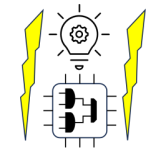


User-provided bitstreams are inherently unsafe:

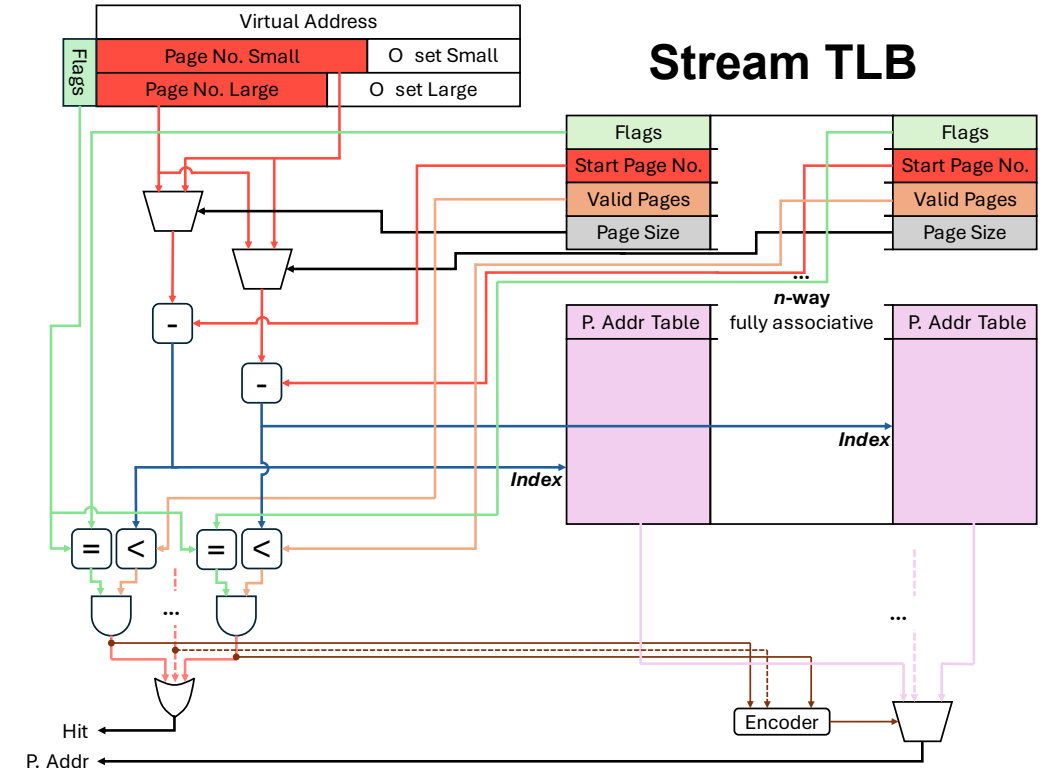
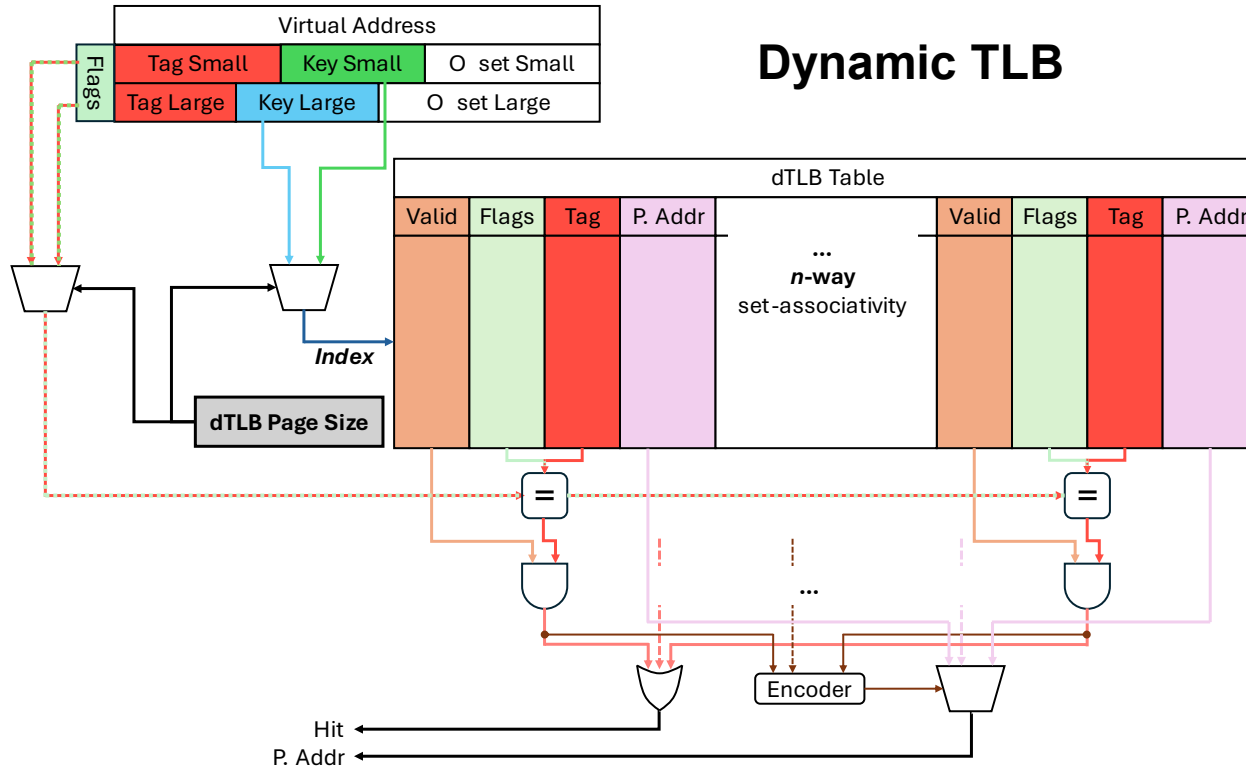
- Exposure of the lowest level possible with all consequences 
- Voltage-caused destruction of the FPGA 
- Eavesdropping between neighboring applications 

ML-based bitstream checking directly on the FPGA via hls4ml / FINN makes the FPGA a **completely independent multi-tenancy platform** and allows to offer „**FPGA-as-a-service**“ in the public datacenter.

Optimization: ML-based Memory Management

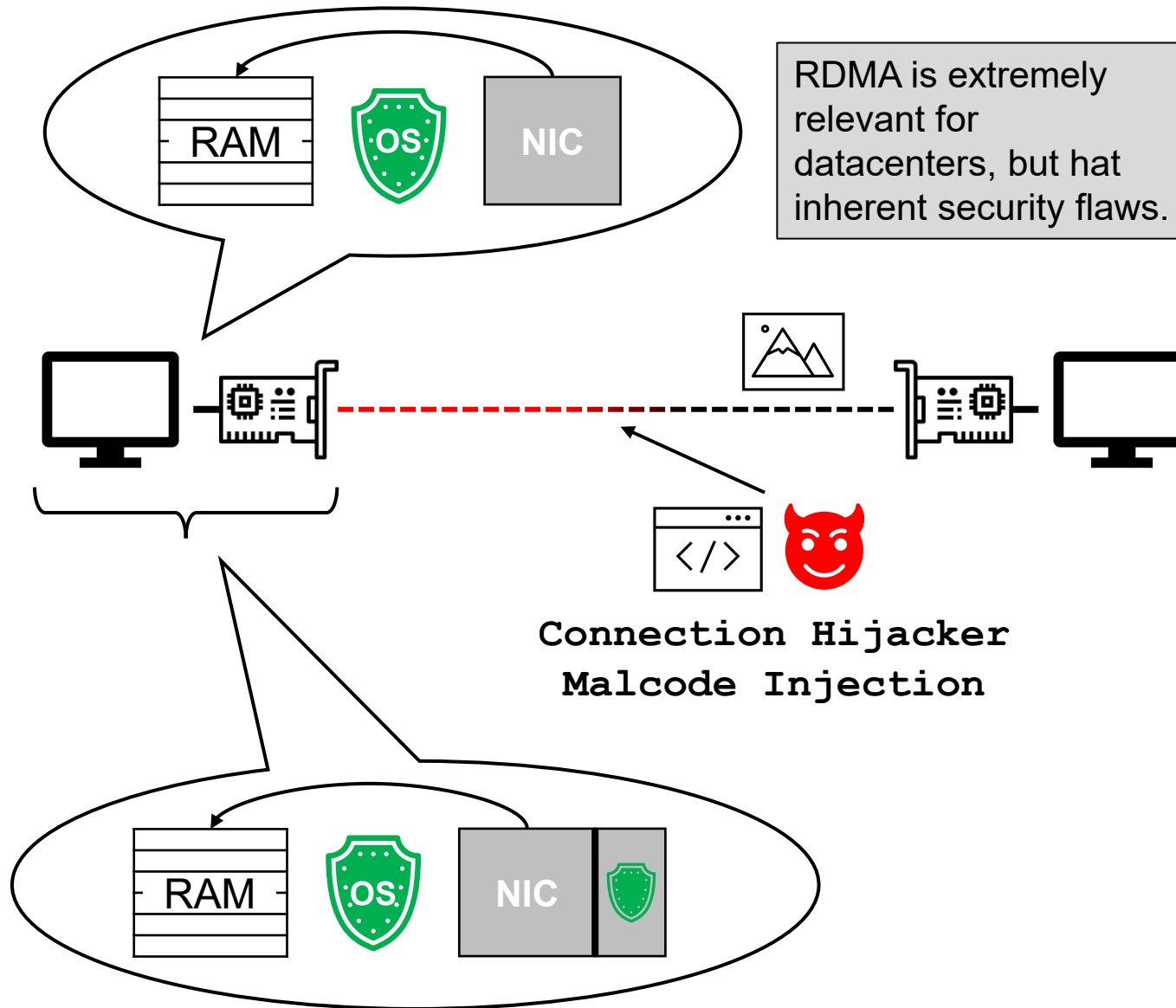
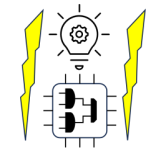


Coyote v2 offers multiple different TLB-designs for different workload purposes (configuration memory, data stream memory etc.). The correct allocation of user data to one of these TLBs is a key optimization problem.

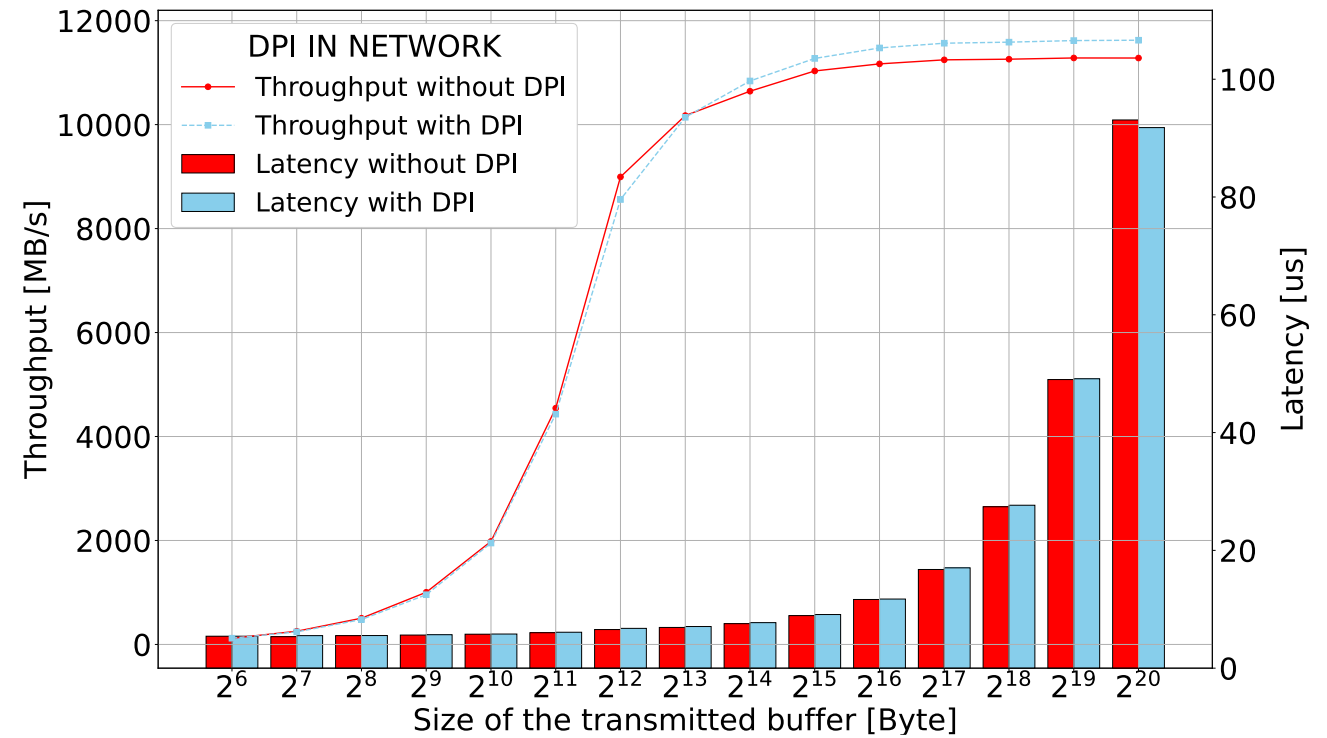
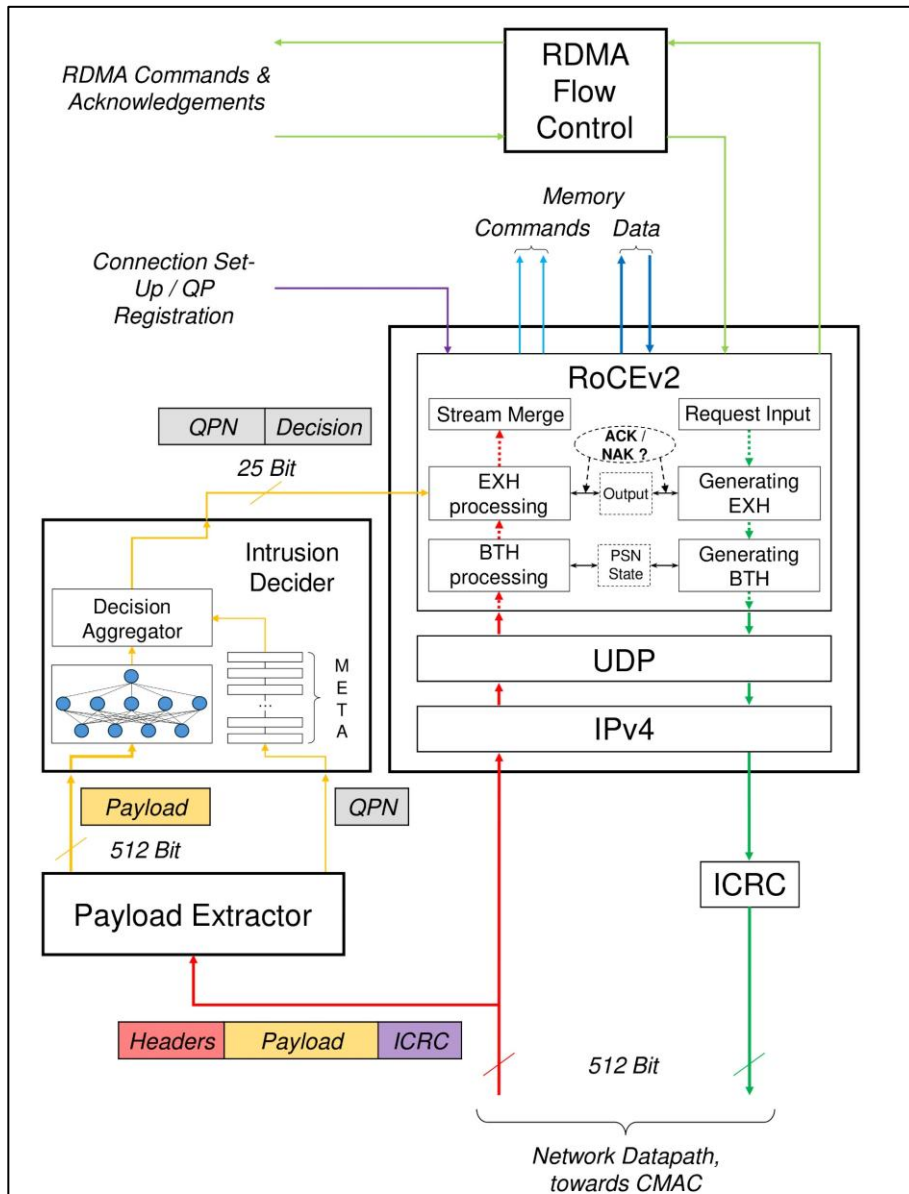
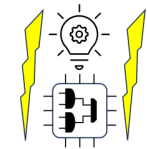


Utilization of a FPGA-deployed ML-model to automatically assign incoming data allocations to the optimal TLB for the use case.

Anomaly detection: ML-based Deep Packet Inspection

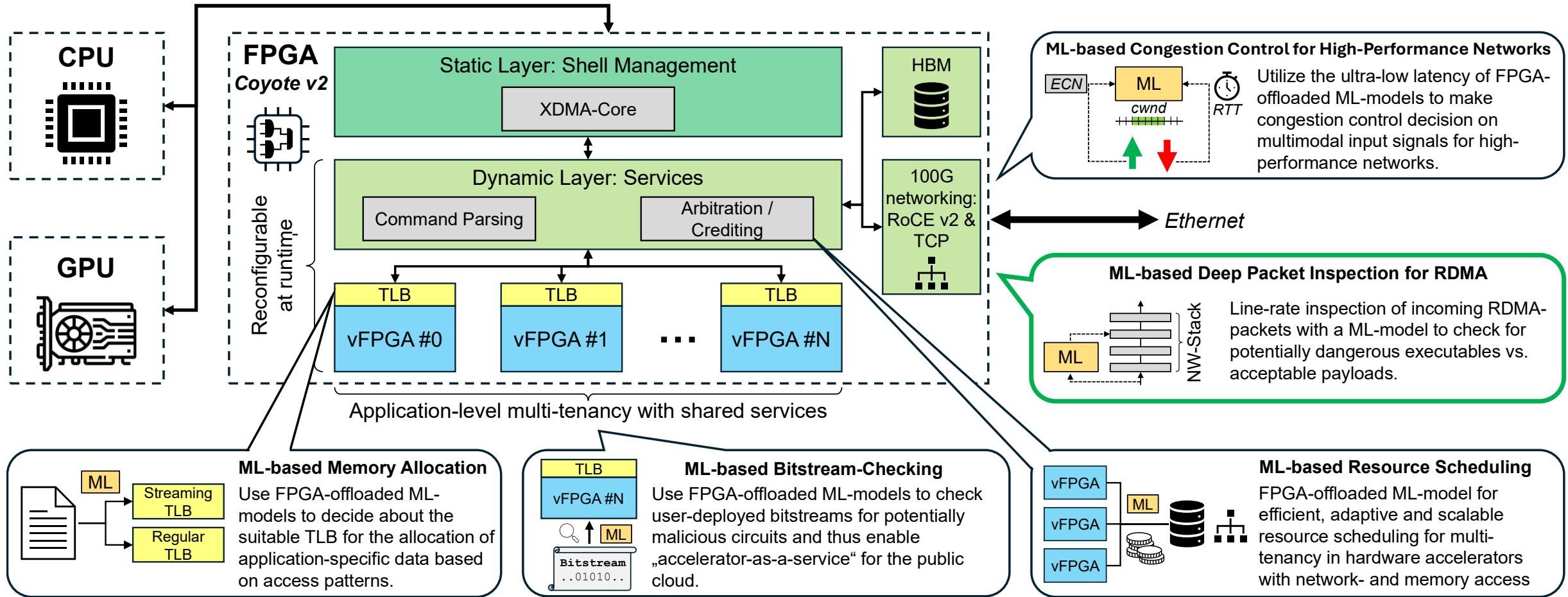
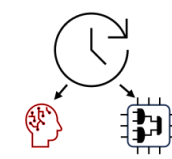


Anomaly detection: ML-based Deep Packet Inspection



Model	Accuracy [%]	FPR [%]	FNR [%]
SR	95.31 ± 0.33	5.04 ± 0.53	4.33 ± 0.35
Binary	87.19 ± 5.22	15.22 ± 14.2	10.43 ± 12.34
Ternary	97.83 ± 0.16	1.74 ± 0.37	2.59 ± 0.38
8-bit quantized	97.88 ± 0.13	1.76 ± 0.29	2.48 ± 0.24

Overview and next steps



Next steps from here on:

- Extract required training data from the described use cases
- Implement models (model architecture, data compression etc.)
- Evaluate by comparison with traditional, non-ML system techniques