2025

DESIGN AND VERIFICATION™

DVCON

CONFERENCE AND EXHIBITION

EUROPE

MUNICH, GERMANY

OCTOBER 14-15, 2025

Pre-Silicon Verification of Software Safety Mechanisms: A Hybrid Approach SPI and NVDLA case studies

A. Makram, M. Nabil, M. Nasser, A. Saied, S. Khourshed

SIEMENS

accellera SYSTEMS INITIATIVE

Agenda

- Motivation
- Safety Verification Foundation
- The System-Level Gap
- Our SoC Integration Approach – SPI Validation
- NVDLA AI Accelerator
- Performance results
- Future work & QA

Motivation

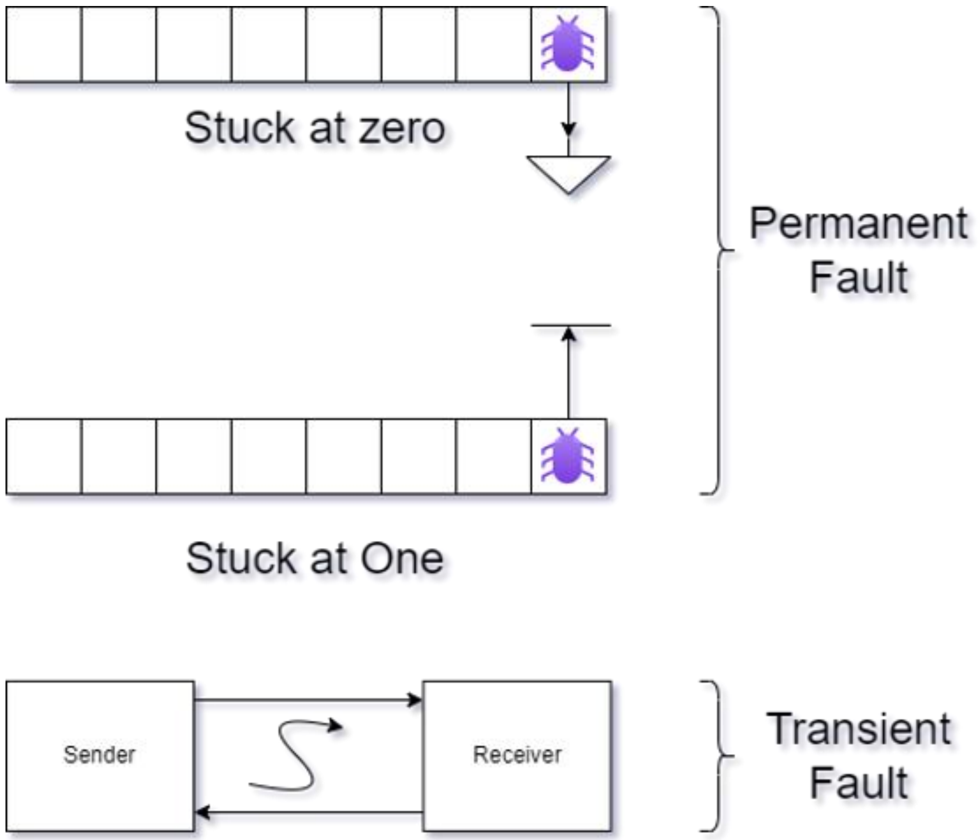
- Software Safety Mechanisms verification
- Testing complex software workloads against fault injection
- Speed up for the Fault Injection simulation .

Safety Verification Foundation

What are Random Faults?

- Random Faults due to
  - Power Supply Noise
  - Extreme Temperature conditions
  - Electromagnetic Interference (EMI)

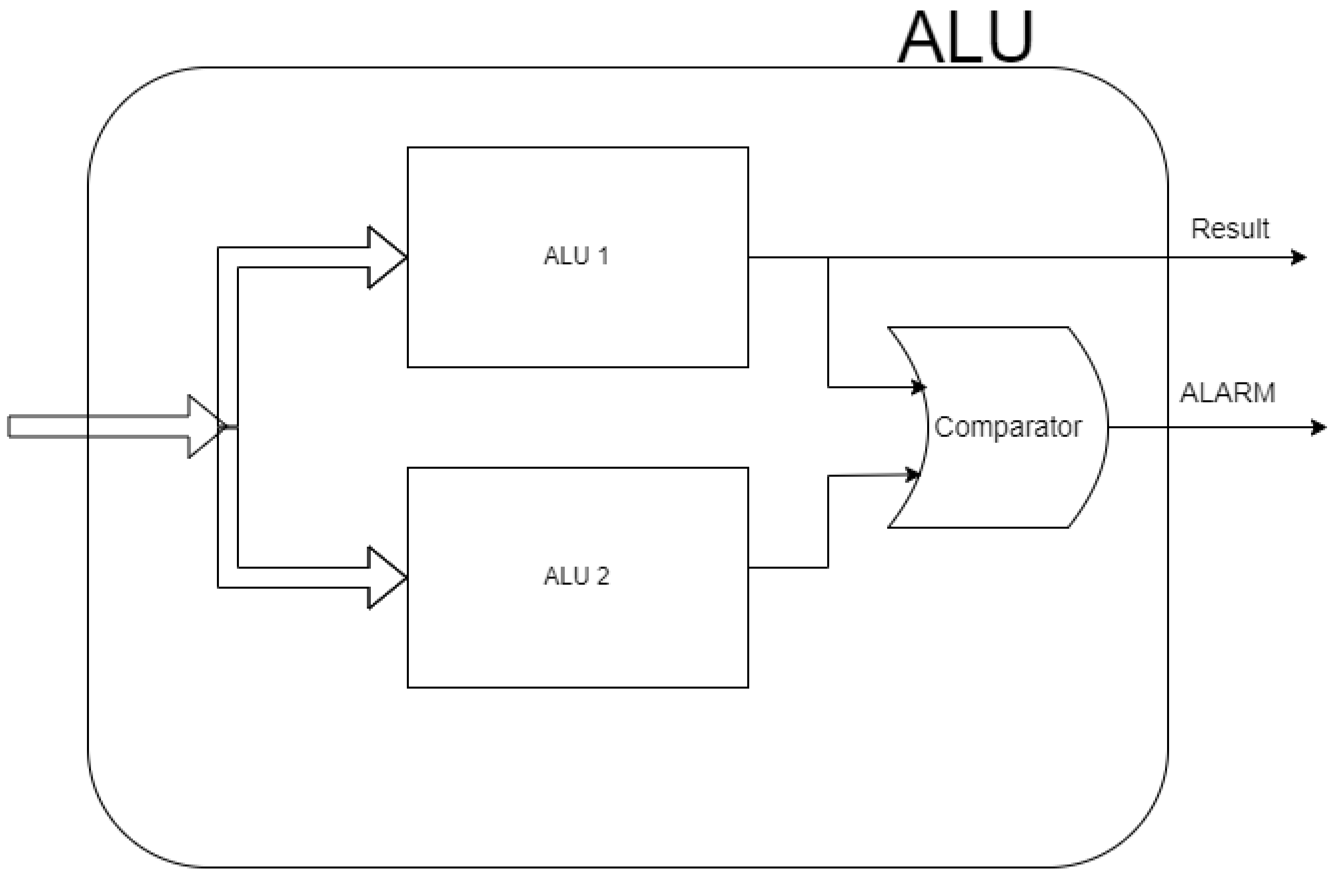
- Modeling Faults in Digital IPs:
  - Stuck at zero or one.



Safety Verification Foundation

What Are Safety Mechanisms?

- **Safety Mechanisms**
  - HW SMs
  - SW SMs
- **HW safety mechanism**
  - Implemented in RTL
  - Example:
    - Duplication and comparator.
    - Triplication and voter.
- **ALARM**
  - Interrupt indicating detection of a fault.

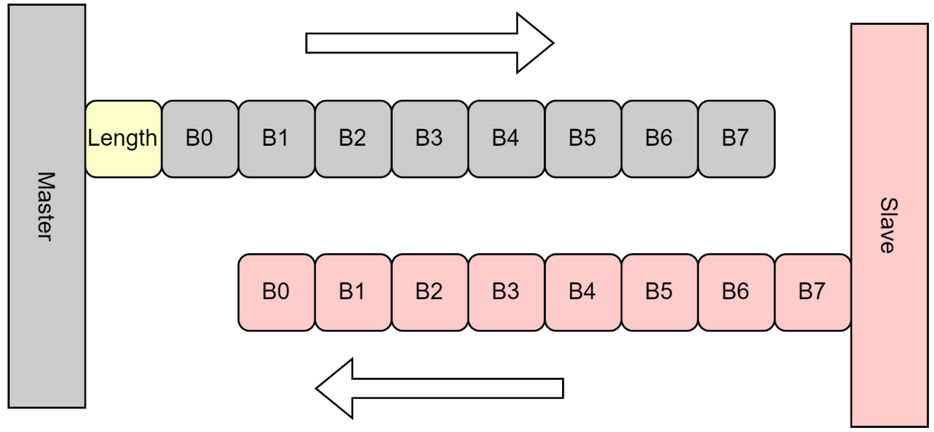


Safety Verification Foundation

Software Safety Mechanisms

- **SW SMs** are implemented on the CPU as software.

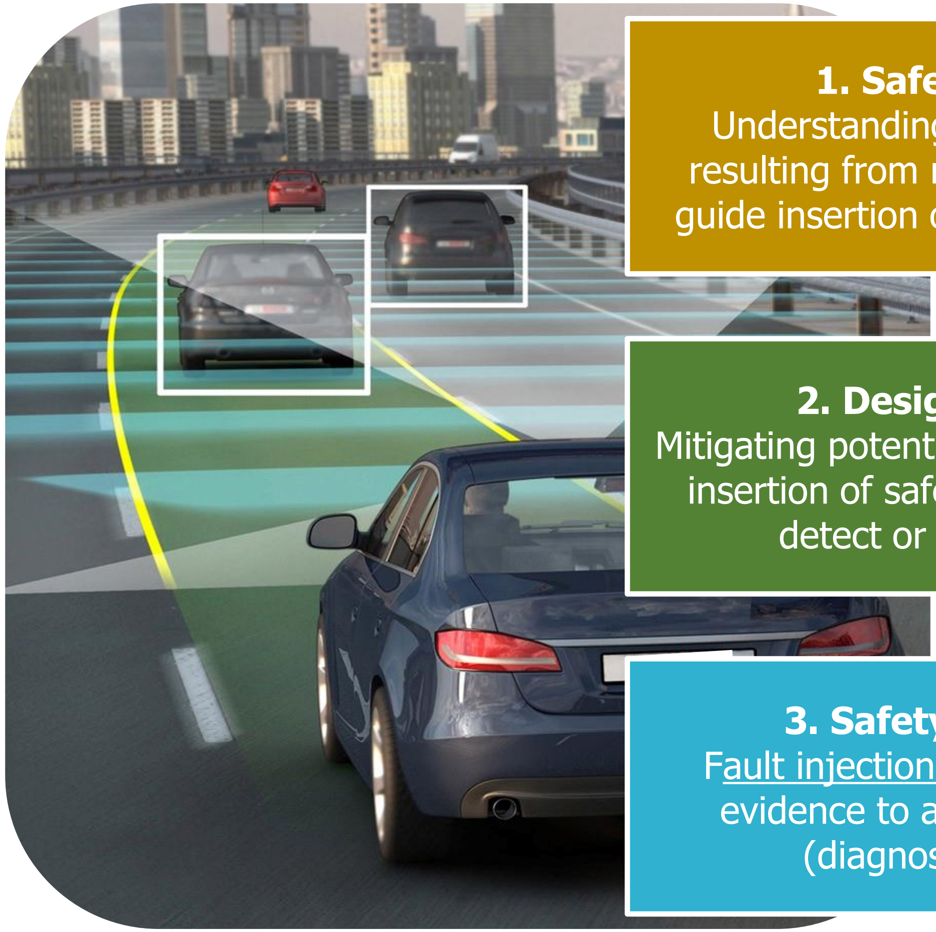
- **SW safety mechanism**
  1. Read after Write
  2. Information redundancy
     - Checksum
     - CRC
  3. Monitoring (watchdog timer)



Frame start Header	Source address	Destination address	Length	DATA	Checksum

Safety Verification Foundation

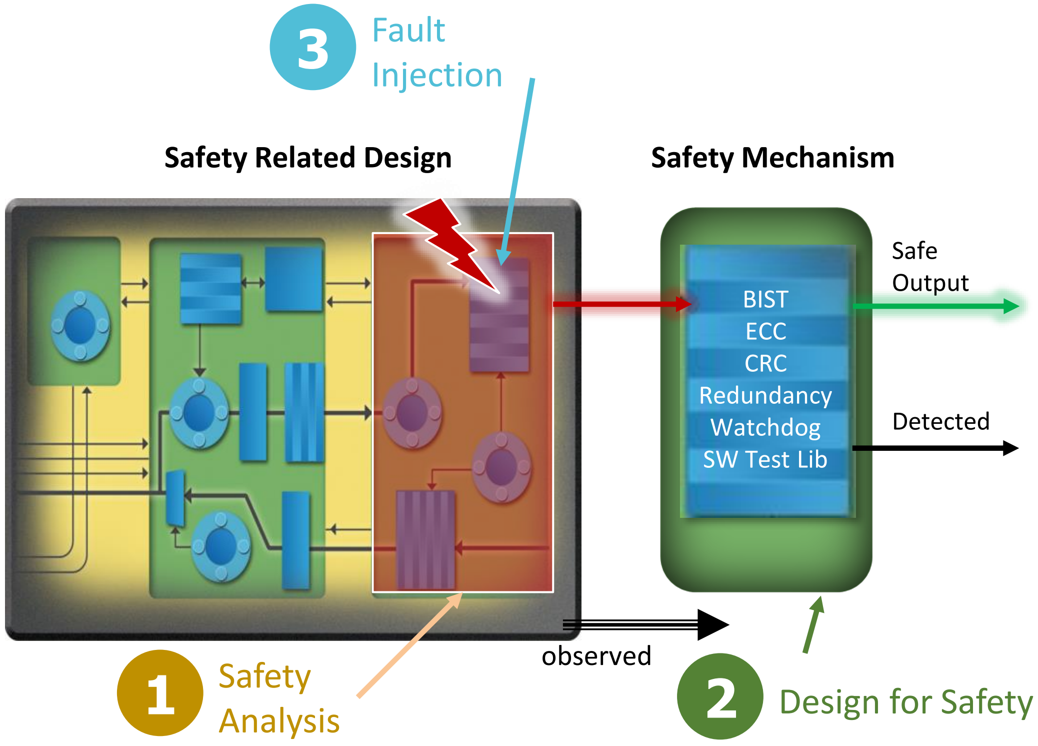
Safety Flow



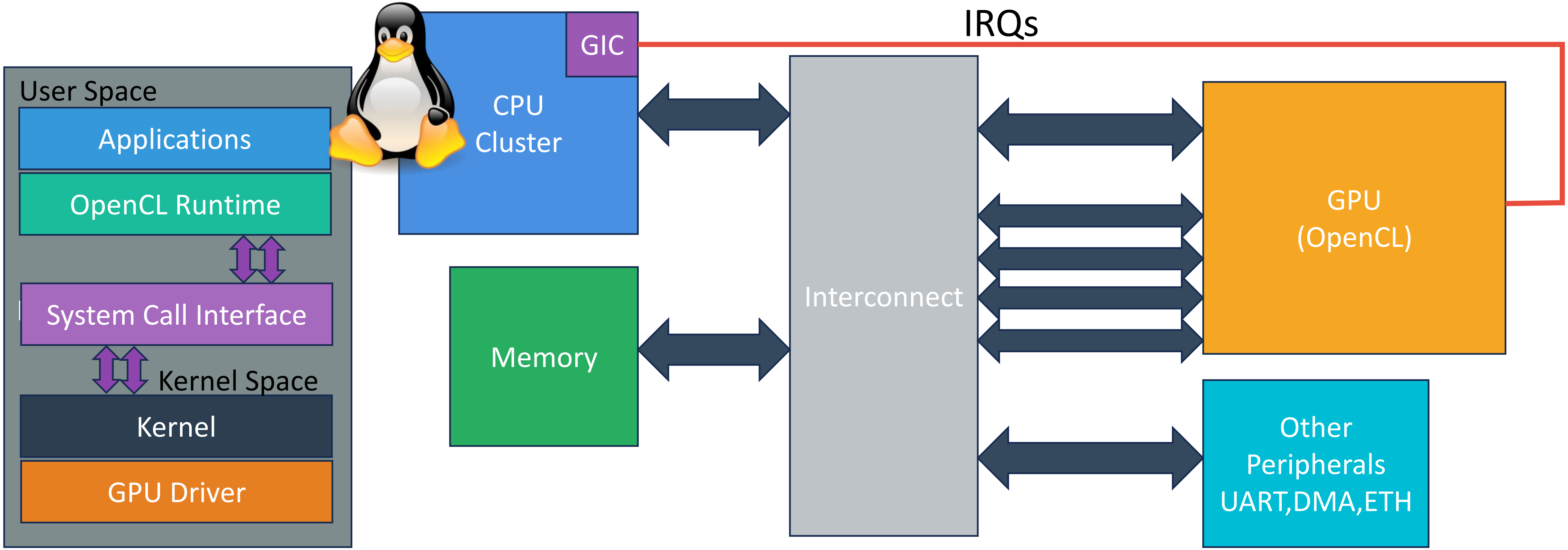
1. Safety Analysis
Understanding the failure modes resulting from random HW faults to guide insertion of safety mechanisms

2. Design for Safety
Mitigating potential failures through the insertion of safety mechanisms that detect or correct failures

3. Safety Verification
<u>Fault injection campaign</u> providing evidence to achieve compliance (diagnostic coverage)



Safety Verification Foundation
Example for Real SOC



The System-Level Gap

❑Requires Ready RTL for the whole SOC IPs.

- Several months of delay to the fault campign

❑Long simulation time for SOCs

- +8 hours of Linux booting

❑Longer time for simulating actual and complex software stacks

❑Ex. GPU/NPU workloads

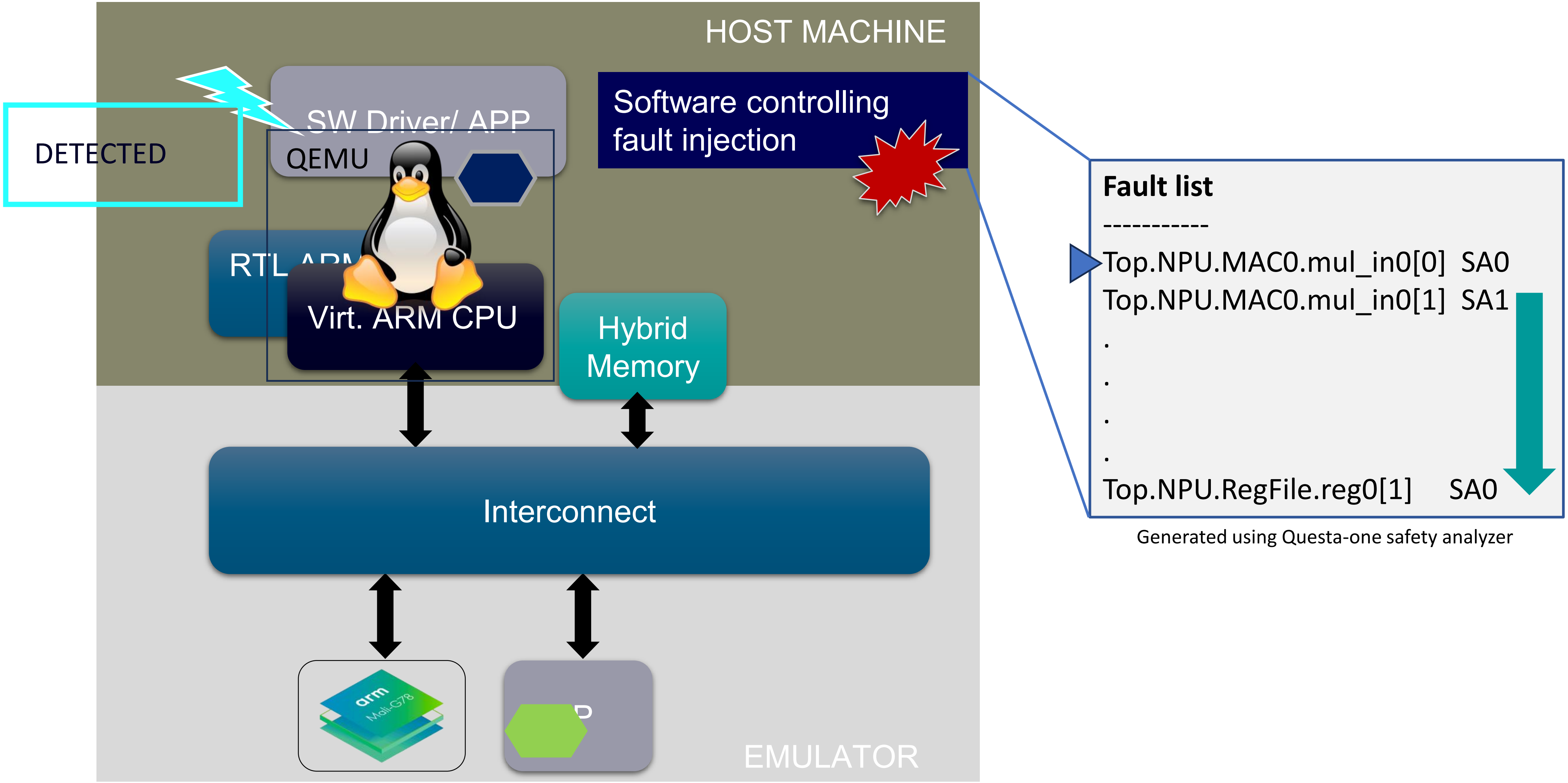
Closing the Gap: Shift Left
Enable Fault injection in Hybrid platform

QEMU

- Emulated CPU models
  ARM, RISC, X86 …
- No need to RTL of other IPs
- Virtual devices
  - vUART, vETH,
- Support different OS.

Veloce

- Highly scalable for large designs.
- Full debugging capabilities .
- Full control on the RTL on runtime.
  - Fault injection in runtime



Generated using Questa-one safety analyzer

Case Study 1 – SPI Controller

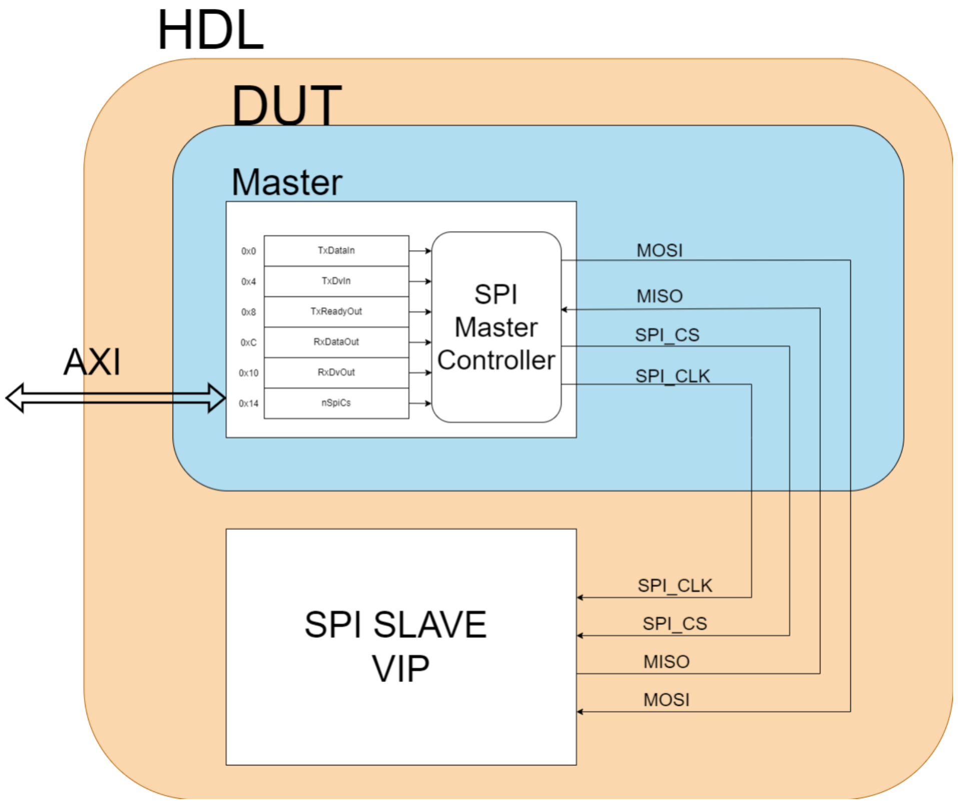


Fig. 2: SPI Architecture

- 82 fault scenarios (stuck-at).
- Aligned results with ISO 26262.
- Combined SMs→ 96.3% detection.

SM	Faults Injected	Faults Detected	Detection (%)
CRC	82	40	48.7
RAW	82	53	65.0
Timeout	82	36	44.0
Timeout+CRC	82	75	91.4
Timeout+RAW	82	79	96.3

Case Study 2 – NVDLA

- Nvidia Deep Learning Accelerator.
- Used in Nvidia Jetson.
- Complex software stack to run CNNs as Yolo, Lenet and Resnet.
- RTL and Virtual parts.

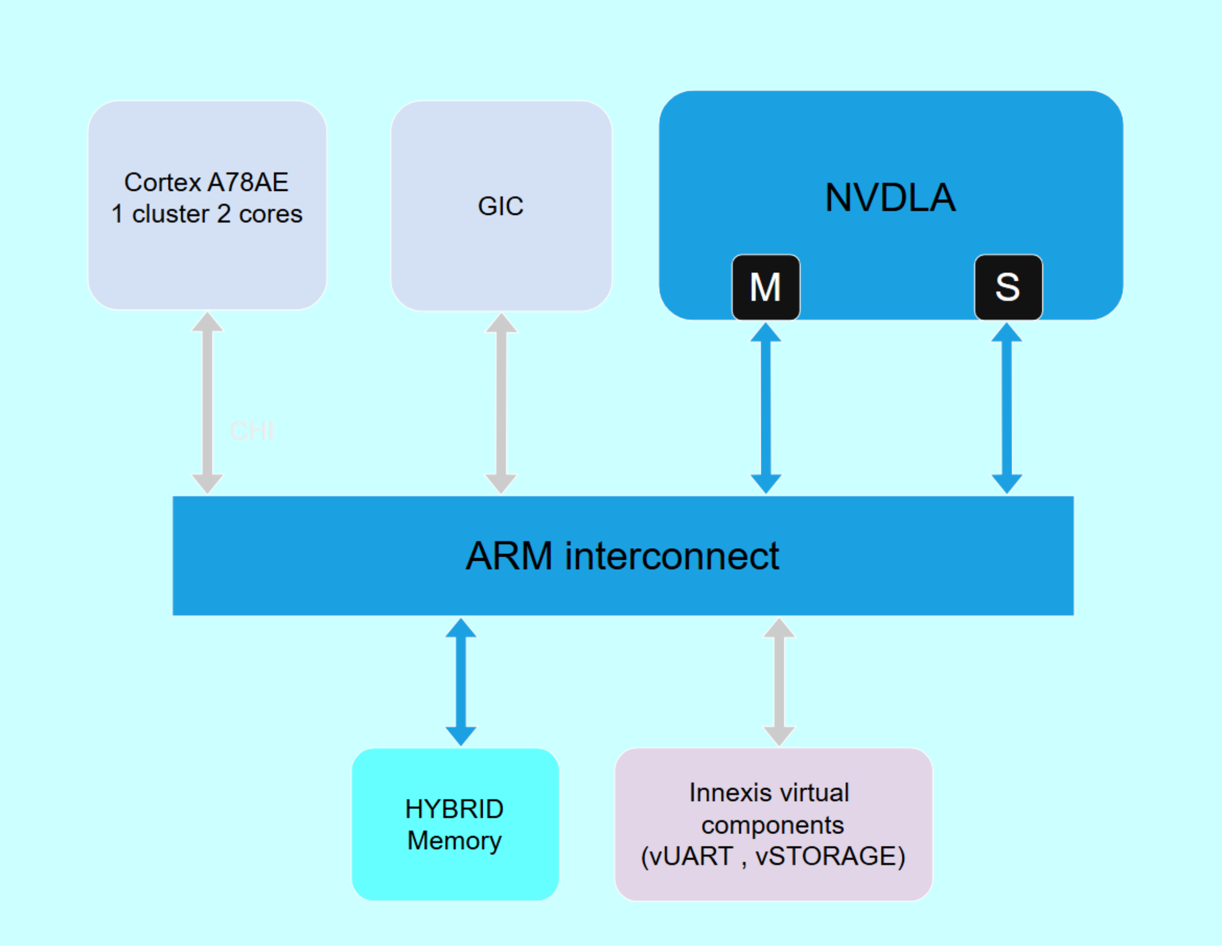
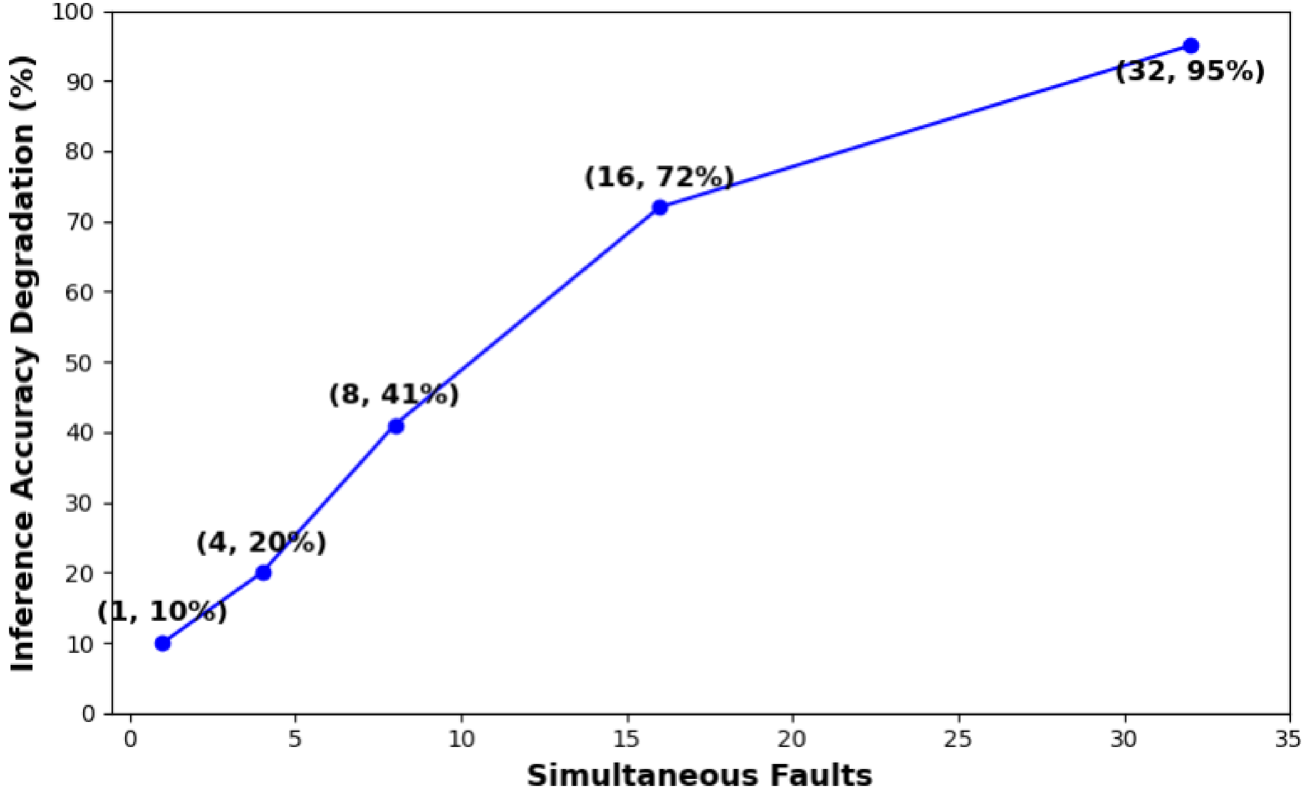


Fig. 3: Integration of NVDLA with hybrid platform

Case Study 2 – NVDLA

- FI on different **1500** signal and register in the RTL.
- Multiple-point FI:
  - 1 fault → 10% accuracy loss
  - 32 faults → 95% accuracy loss
- **NVDLA demonstrate moderate resilience against single faults.**



Accuracy Degradation vs Fault Injection Density

```
# ls /sys/bus/platform/devices/20000000.nvdla
driver            drm               of_node           subsystem
driver_override   modalias          power             uevent
# lsmod | grep opendla
opendla                81920  0
drm                   401408  2 opendla
# head -1 /proc/interrupts
           CPU0       CPU1
# cat /proc/interrupts | grep nvdla
 22:         18          0     GICv3 121 Level     20000000.nvdla
# devmem 0x20000000
0x00010001
# cat output.dimg
0 0 119 0 0 0 0 0 0 0 #
```

Device discovery

Driver discovery

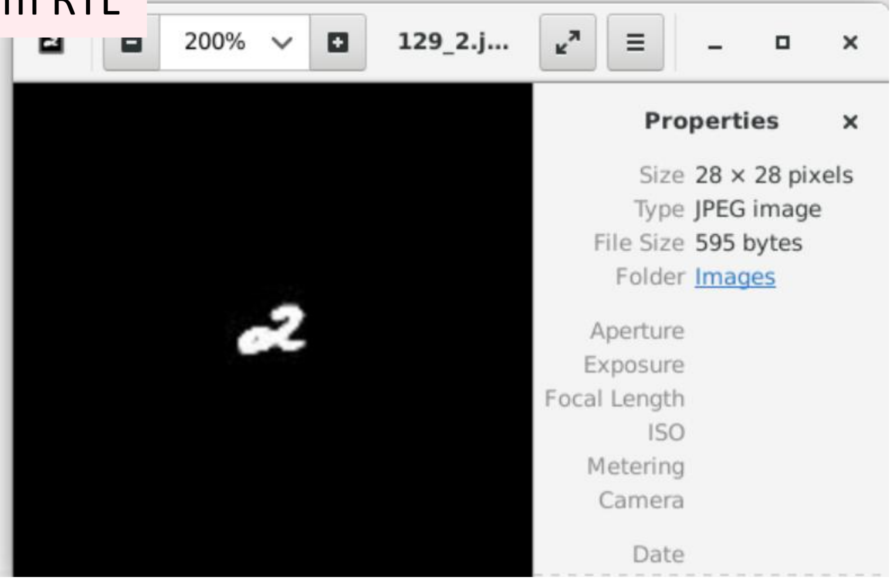
NVDLA Interrupt check

18 interrupts to core 0 during inference.

Read NVDLA ID from RTL

Class 2

Image detected is 2



Performance Results

- 10x speedup over full emulation.
- 1000x speedup over simulation.
- 1000-Fault Campaigns
  - 1 day in Hybrid platform.
  - 2 weeks in fully emulated platform.

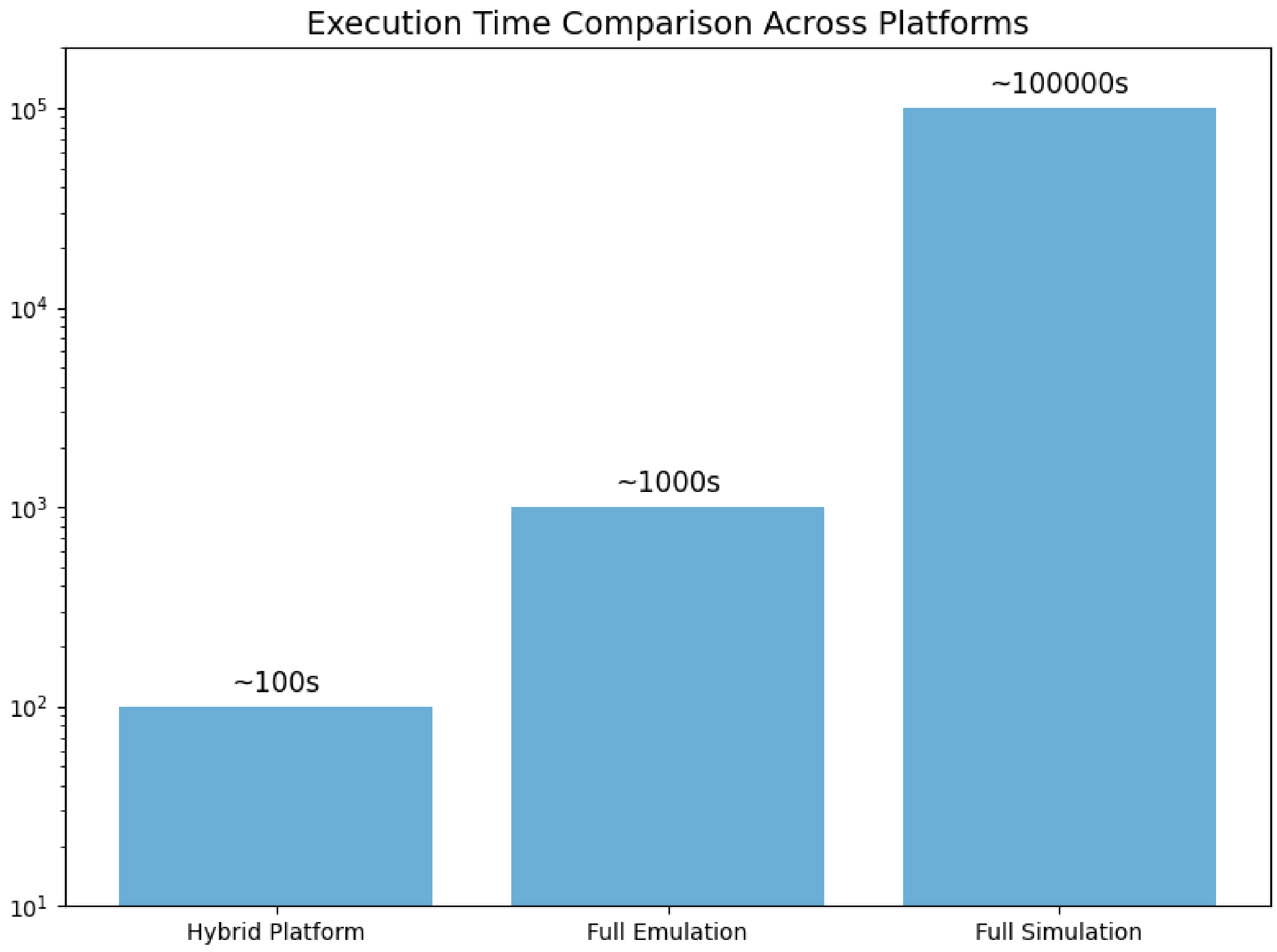


Fig. 5: Runtime Performance Across Verification Platforms

Performance Results
Comparison with FPGA Approaches

	Resources	Run Time	Manual RTL Modifications	Fault Injection Capabilities	Recompilations Required
FPGA SOC[1]	30% (Limited)	~34ms	Manually done	Limited	Many compilations
Hybrid approach	0.6% (Scalable)	1 min 24 sec	No need	Full design	One compilation

Comparison based on NVDLA inference on CiFar10 using ResNet18 CNN.

(1) Late Breaking Result: FPGA-Based Emulation and Fault Injection for CNN Inference Accelerators

Key Takeaways

- Hybrid approach enables early, accurate software safety verification before full SoC availability.
- Implemented SMs aligned with the ISO26262.
- Scales to complex IPs like NVDLA.
- Significant runtime speedup against different verification platforms.
- One-time compilation supports entire safety campaign.

Future Work

- Advanced analysis of AI accelerators against fault injection.
- We would like to try different safety mechanisms.

Thank You

Questions ?

Ahmed Makram
ahmed.makram@siemens.com