



TECHNISCHE
UNIVERSITÄT
WIEN



Rapid Prototyping Methods for custom-tailored, safe and secure RISC-V processors

D. Mueller-Gritschneider*, Johannes Geier**

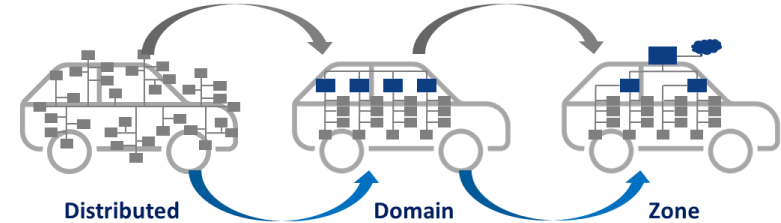
*TU Wien Informatics

** TU Munich CIT

Computing Trends: Advanced Driving Functions

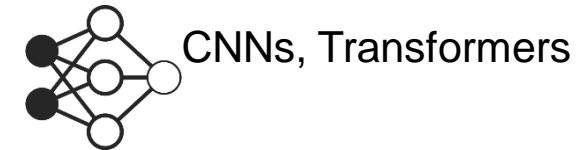
Hardware view:

- Migration from Distributed, Domain to Zone.
- Few very powerful compute platforms.
- Embedded HPC



Software View

- Rising number of SW functions on same platform
- Rising number of AI-based workloads.



Safety-critical real-time system

- Real-time constraints / deadlines.
- Functional safe and secure.



<https://www.benzinsider.com/2015/01/car-future-smartphone-wheels/>

Computing Trend tinyML

The Future of AI Is Tiny

Tiny AI reduces carbon footprints, brings deep learning at an affordable cost, creates context-aware consumer devices, cuts down data infrastructure, bolsters security, and more.

<https://www.informationweek.com/data-management/the-future-of-ai-is-tiny/>

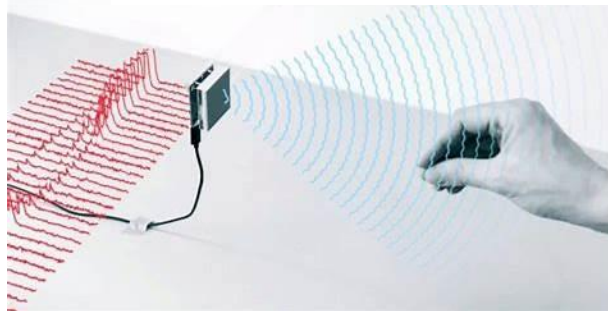
Running NN inference on low-power micro-controllers / IoT Devices

- **Audio:** Keyword Spotting (KWS) / Audio Wakeup
- **Vision:** Video Wakeup (Face Detection)
- **Radar:** Gesture Recognition
- **Accelerometer:** Activity Detection

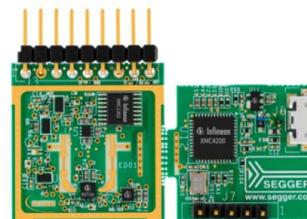
TinyML Device Shipments to Grow to 2.5 Billion in 2030, Up From 15 Million in 2020

<https://www.abiresearch.com/press/tinyml-device-shipments-grow-25-billion-2030-15-million-2020/>

Gesture Recognition with Radar Sensor



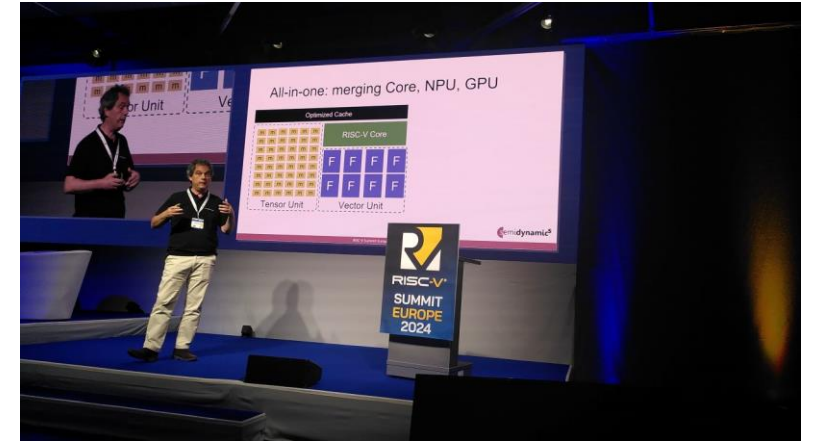
tinyML App



Platform: Infineon XMC1302 MCU
32 MHz Micro-controller CPU
32 kB Flash, 16 kB RAM

Why RISC-V?

- RISC-V is an **open standard** for an **Instruction Set Architecture (ISA)**
- Developed by UC Berkeley (2011*).
- What is an open standard ISA?
 - Everyone can implement an RISC-V processor based on the open ISA specification without having to pay royalties.
 - Implementations can be open-source or closed / commercialized.
- Community-driven (RISC-V International)
- Growing adoption in research, university curricula and industry



RISC-V Summit Europe 2024

Specifics of RISC-V

- Modular ISA design
- Building blocks: Extensions
- Examples:
 - RV32IMAF
 - RV64GV
- Encodings reserved for custom instructions (**CIs**)
 - Allows for ASIP designs e.g. for AI or security workloads

Extension	Name
I	Integer base instructions
M	Integer multiplication and division instructions
A	Atomic instructions
F	Single-precision floating-point instructions
D	Double-precision floating-point instructions
G	General (I + M + A + F + D)
C	Compressed instructions
B	Bit manipulation instructions
P	Packed-SIMD instructions
V	Vector operations instructions
N	User-level interrupt instructions
	...

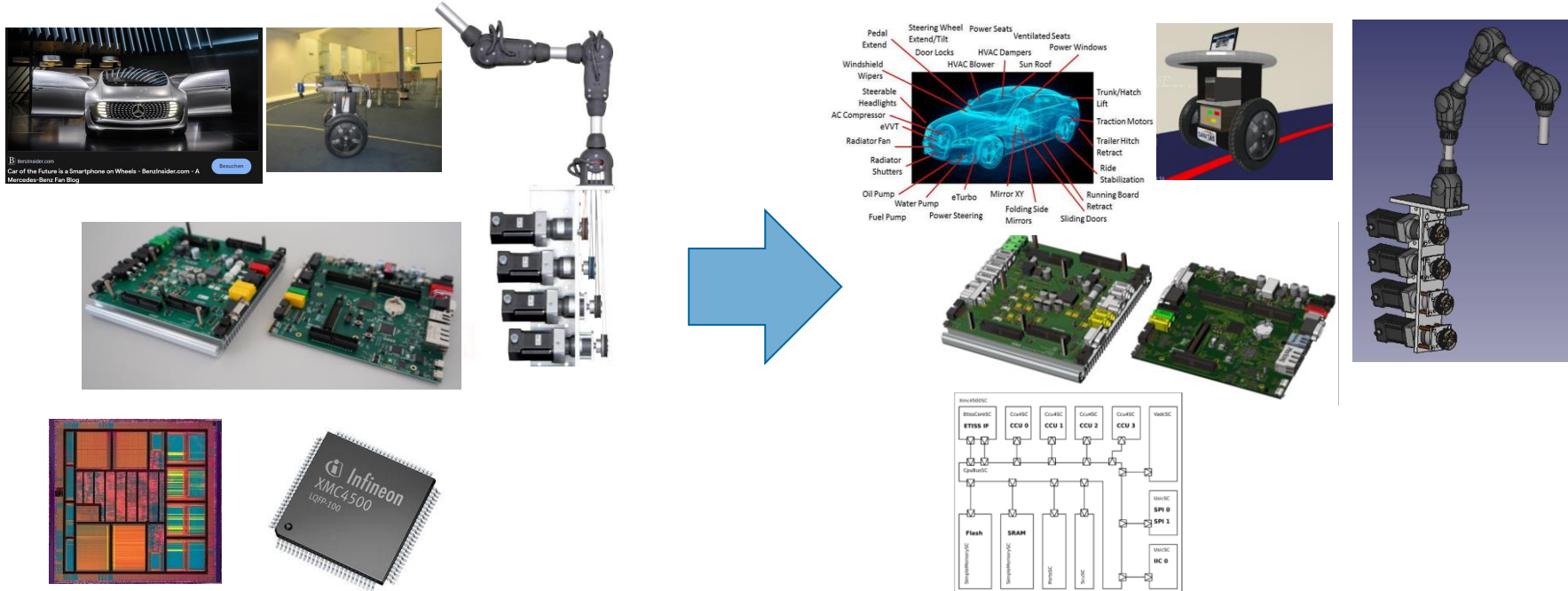
Rapid Prototyping

- RISC-V offers many choices
- Rapid prototyping methods to explore the possibilities

Agenda

- **ETISS: Simulation of Functional Behavior with CIs**
- **Seal5: LLVM Compiler Support for CIs**
- CorePerfDSL: Software Performance Simulation
- Safety in the presence of HW faults

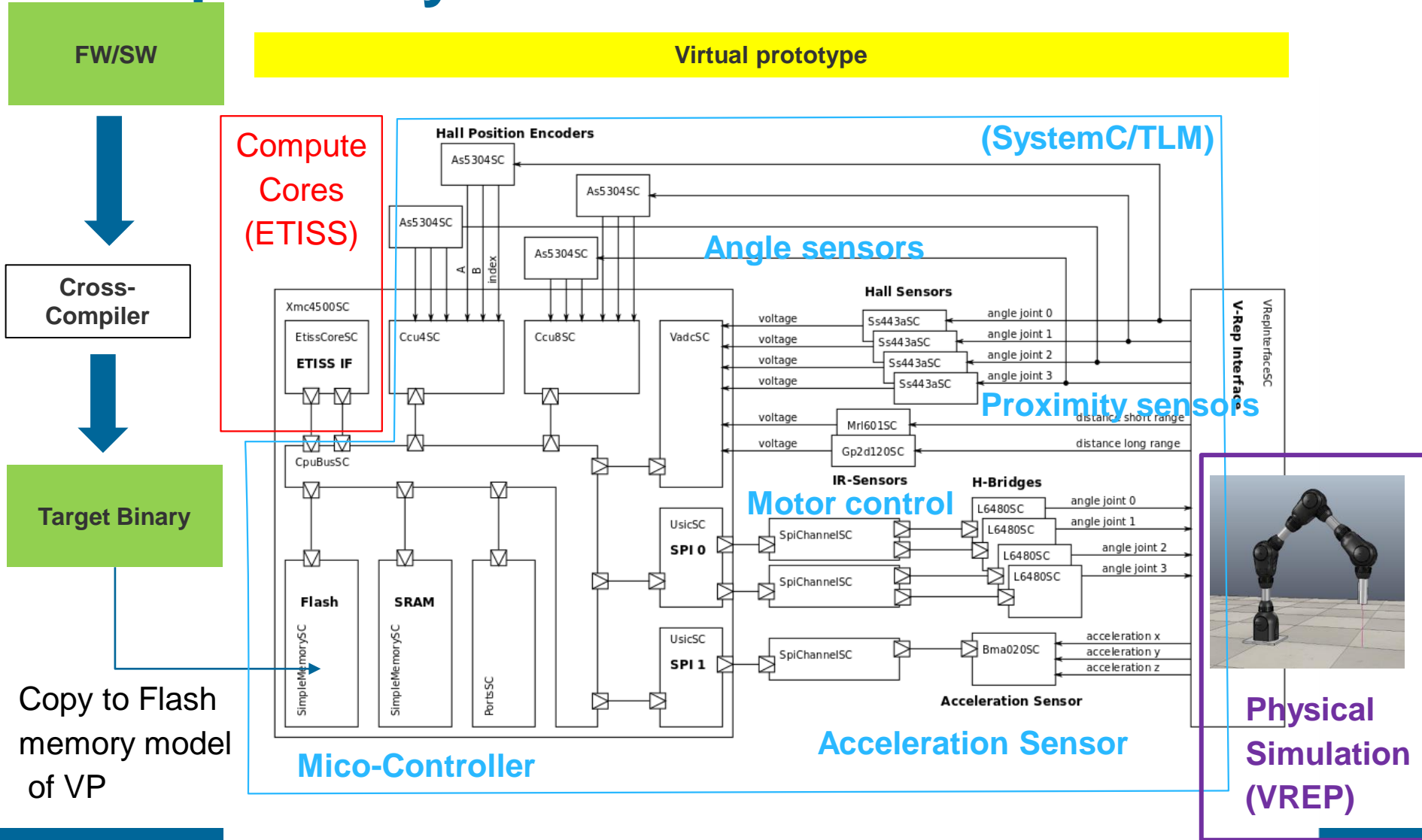
Pre-Silicon Simulators: Virtual Prototypes (VPs)



- Computer model of an System-on-Chip (SoC) or Embedded System
- Sufficiently detailed model of SoC to allow an emulation of the embedded software

- Early software development
- Architectural Exploration
- Early Validation

Example of SystemC VP: Robotic Arm

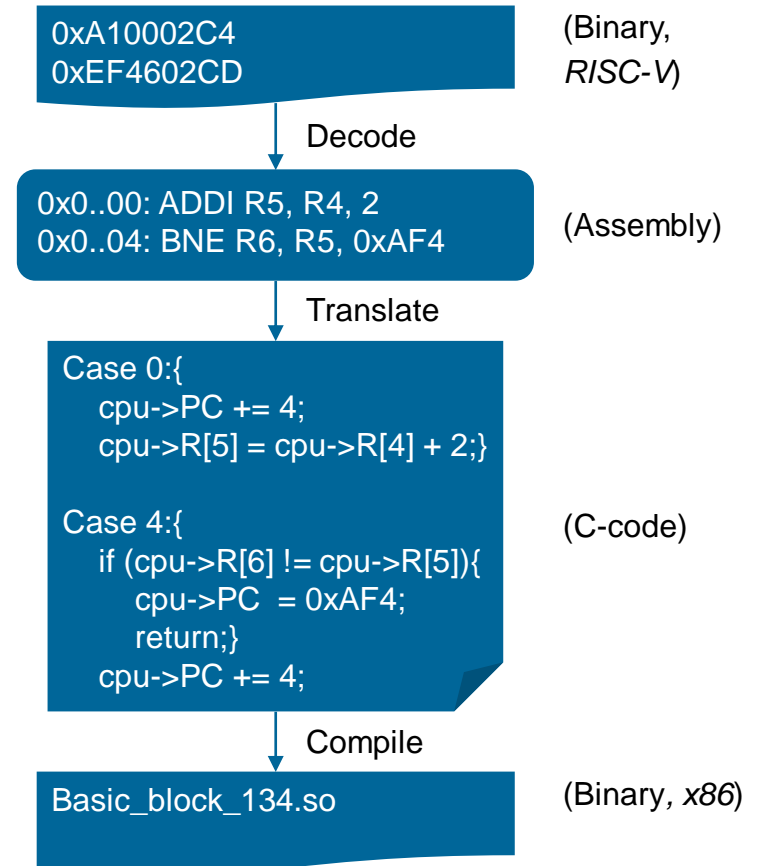


ETISS: Extensible Translating Instruction Set Simulator

Key Features:

- ISA-independent
- Good RISC-V Support (32/64)
- Open Source, permissive (BSD3)

- Fast binary translation
- Simulation speed: ~100+ MIPS
- High flexibility:
 - Stand-alone (C++)
 - Part of Virtual Prototype (SystemC/TLM)
- **Plugin-mechanism for simulator extensions**
- **Support for custom instructions**



M2ISAR

■ CoreDSL2 Description (Input):

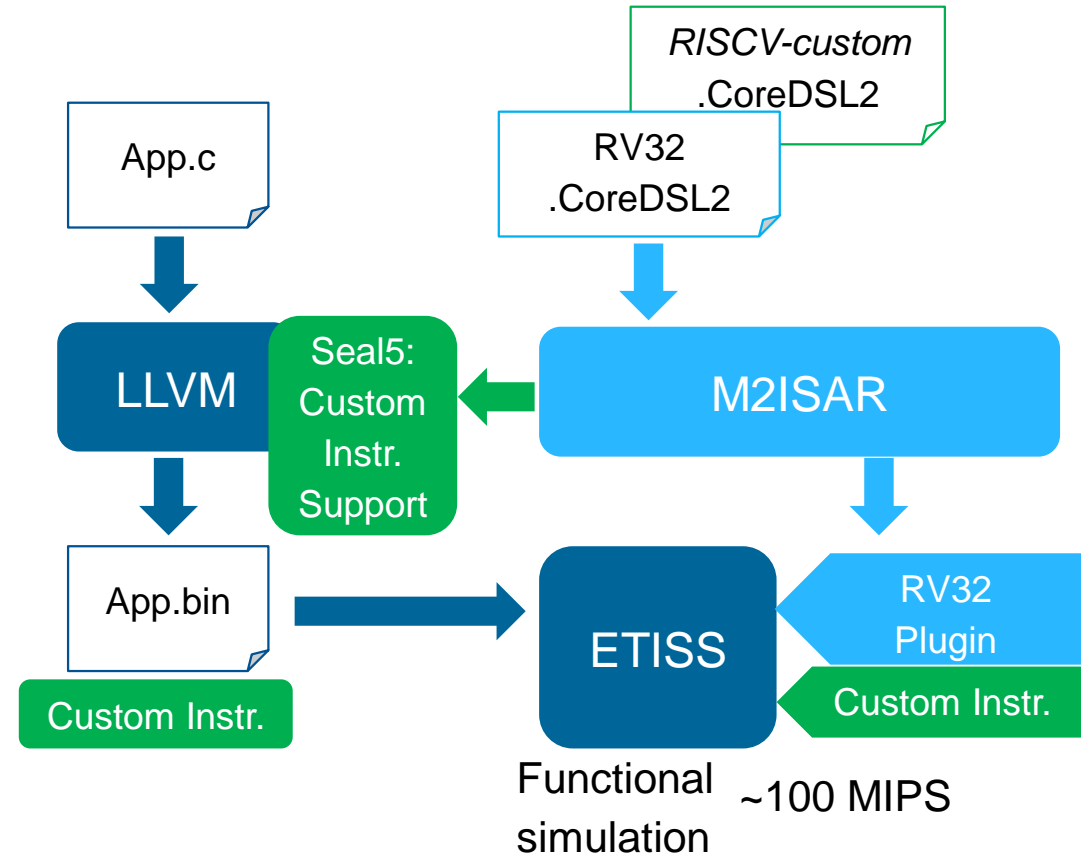
- C-style DSL, Open Spec
- Open Source Examples: RV32,RV64

■ Output:

- ETISS Simulator Support
 - Base ISA Simulator
 - Simulation Support for CIs
- LLVM Patch (Seal5)
 - Compiler Support for CIs

CoreDSL2 Definition for RISC-V Add Instruction

```
ADD {  
    encoding: b0000000 | rs2[4:0] | rs1[4:0] | b000 | rd[4:0] | b0110011;  
    args_disass: "{name(rd)}, {name(rs1)}, {name(rs2)}";  
    if(rd != 0) X[rd] <= X[rs1] + X[rs2];  
}
```



Results for Core-V Extension (CV32E40P)

CORE-V Extension: ~200 Custom instr.
Comparison (large set of benchmarks):

OpenHW Group Community LLVM

Manual implemented
Compiler uses 16%
of CoreV Instructions

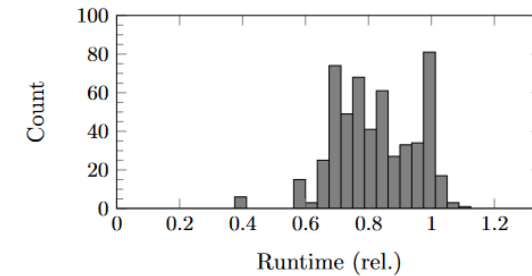
Average
performance gain
over plain RV32 is
15% speedup

Seal5 generated LLVM from CoreDSL2

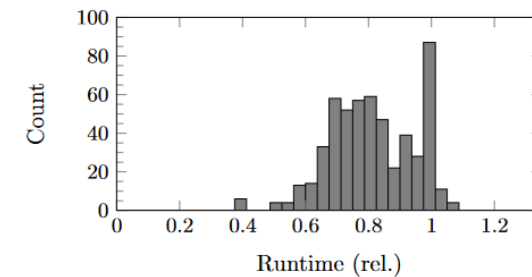
Seal5 semi-
automatic generated
compiler uses 33% of
CoreV Instructions

Average
performance gain
over plain RV32 is
24% speedup with
SIMD

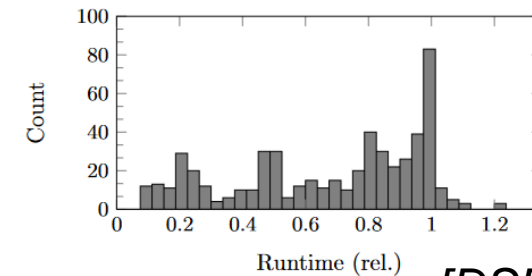
OpenHW Group Community LLVM



Seal5 generated LLVM (no SIMD)



Seal5 generated LLVM (with SIMD)



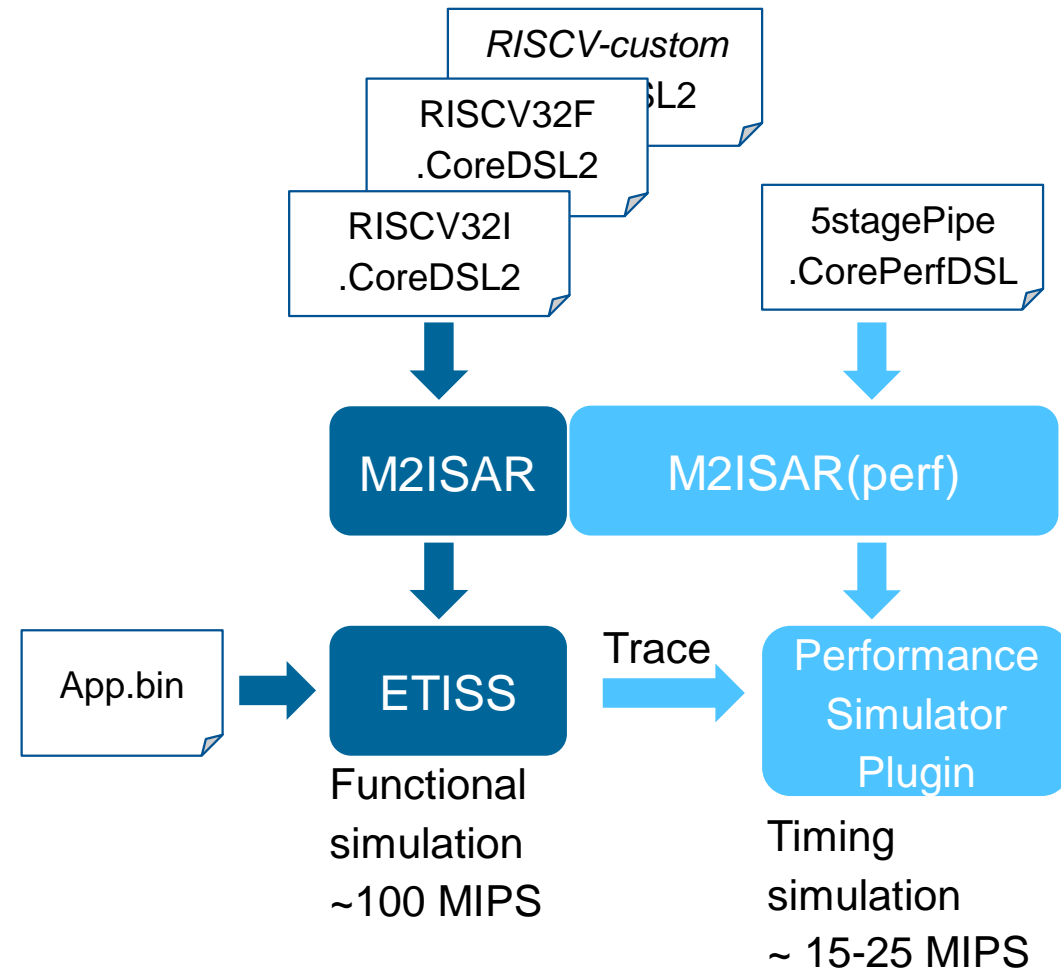
[DSD24 – Matter-V]

Agenda

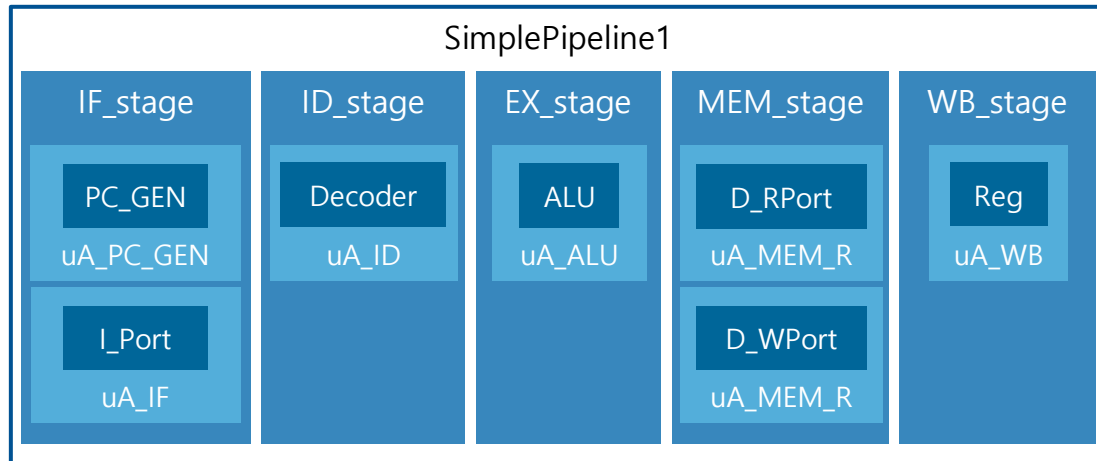
- ETISS: Simulation of Functional Behavior with CIs
- Seal5: LLVM Compiler Support for CIs
- **CorePerfDSL: Software Performance Simulation**
- Safety in the presence of HW faults

Plugin-Example: Performance Simulator

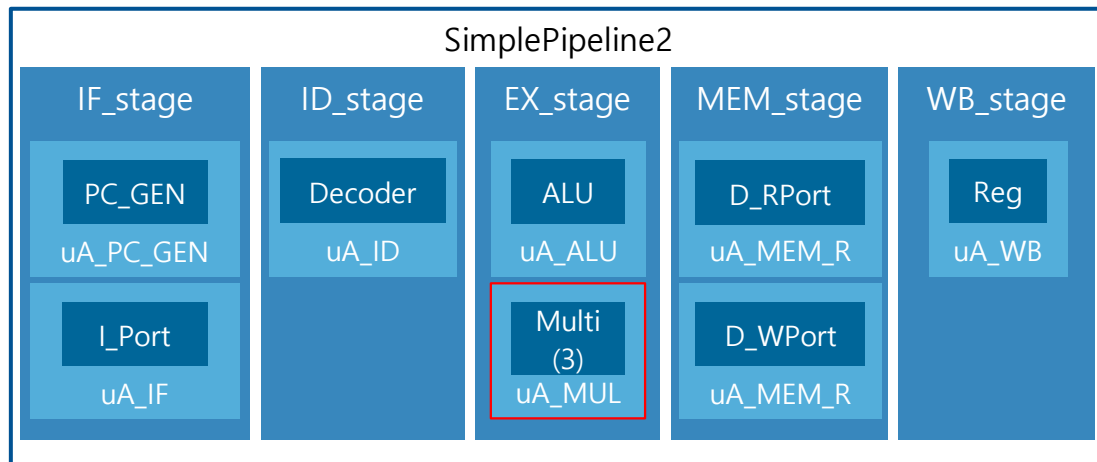
- Target: Estimate cycle-approximate software performance at ISS speed.
- CorePerfDSL: Domain Specific Language to flexibly describe the processor pipeline
 - Stages, resources, forwarding paths
 - Branch prediction, caches, ...
- Performance Simulator: Does not simulate data in pipeline stages, only the timing



CorePerfDSL : Simple Pipeline Model



IF	ID	EX	MEM	WB
ADD				
SUB	ADD			
	SUB	ADD		
		SUB	ADD	
			SUB	ADD
				SUB



IF	ID	EX	MEM	WB
MUL				
SUB	MUL			
	SUB	MUL		
			MUL	
		SUB	MUL	
			SUB	MUL
				SUB

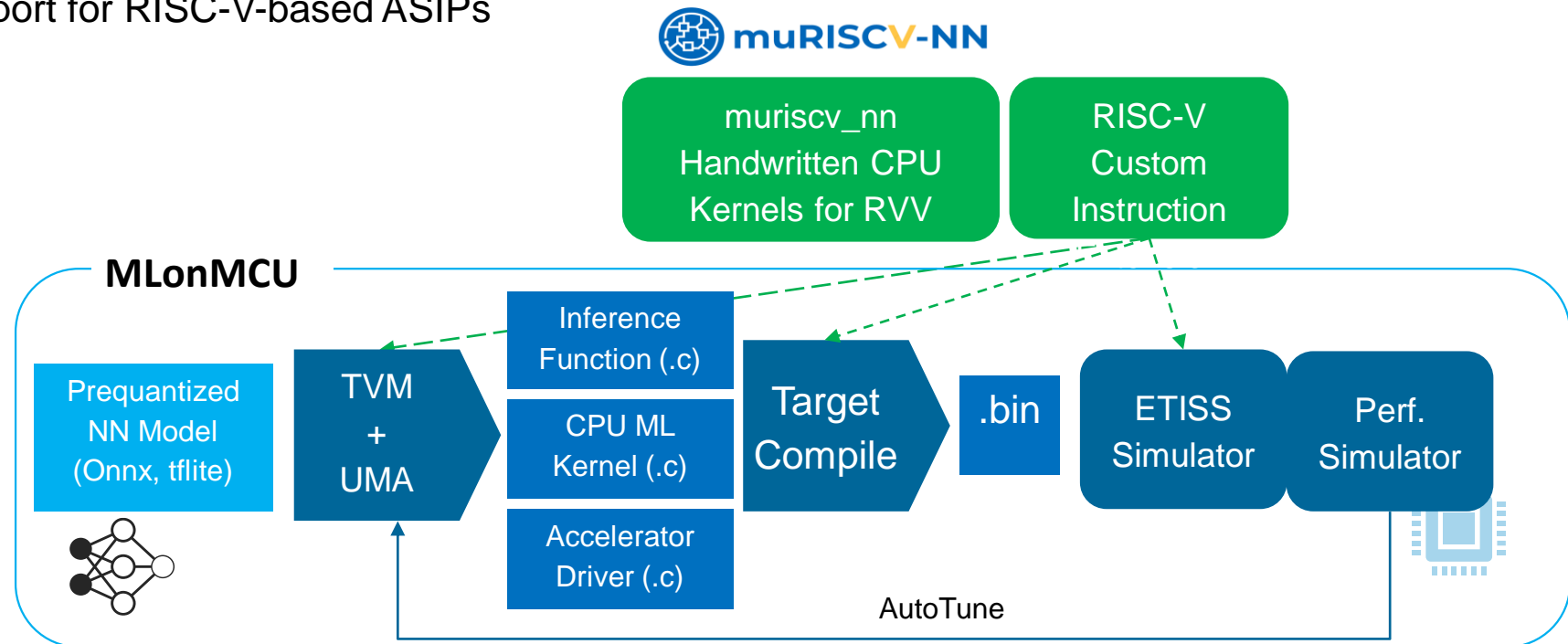
Results for CV32E40P and CVA6 (CODES/ISSS24)

Benchmark	CV32E40P									CVA6						
	Num. Instr.	RTL Cycles	CPI	ISS, CPI=1 Error	ISS, CPI=1.29 Error	Perf.Sim. Cycles	CPI	Error	Num. Instr.	RTL Cycles	CPI	ISS, CPI=1 Error	ISS, CPI=1.45 Error	Perf.Sim. Cycles	CPI	Error
aha-mont64	4,534,817	5,313,414	1.17	14.7%	10.1%	5,313,364	1.17	0.000941%	1,918,426	2,643,667	1.38	27.4%	5.22%	2,634,848	1.37	0.334%
crc32	4,183,376	4,881,112	1.17	14.3%	10.6%	4,881,062	1.17	0.00102%	4,184,058	6,106,933	1.46	31.5%	0.656%	5,932,284	1.42	2.86%
cubic	6,854,333	8,189,299	1.19	16.3%	7.97%	8,189,249	1.19	0.000611%	3,743,102	5,364,025	1.43	30.2%	1.18%	5,594,859	1.49	4.30%
edn	3,447,366	4,132,445	1.20	16.6%	7.61%	4,132,395	1.20	0.00121%	3,367,939	4,141,912	1.23	18.7%	17.9%	3,818,705	1.13	7.80%
huffbench	2,302,258	2,963,665	1.29	22.3%	0.211%	2,963,615	1.29	0.00169%	2,297,463	3,366,148	1.47	31.7%	1.03%	3,343,952	1.46	0.659%
matmult-int	3,602,152	4,377,284	1.22	17.7%	6.16%	4,377,234	1.22	0.00114%	4,334,701	5,252,905	1.21	17.5%	19.7%	4,878,131	1.13	7.13%
minver	2,509,246	3,581,566	1.43	29.9%	9.62%	3,581,516	1.43	0.00140%	2,287,078	3,572,886	1.56	36.0%	7.18%	3,488,838	1.53	2.35%
nbody	3,177,393	3,783,510	1.19	16.0%	8.33%	3,783,460	1.19	0.00132%	2,248,434	3,173,887	1.41	29.2%	2.72%	3,146,675	1.40	0.857%
nettle-aes	4,406,553	4,628,360	1.05	4.79%	22.8%	4,628,310	1.05	0.00108%	5,333,799	5,509,595	1.03	3.19%	40.4%	5,478,349	1.03	0.567%
nettle-sha256	3,965,640	3,991,463	1.01	0.647%	28.2%	3,991,413	1.01	0.00125%	4,022,440	4,192,764	1.04	4.06%	39.1%	4,545,013	1.13	8.40%
nsichneu	2,241,955	3,619,320	1.61	38.1%	20.1%	3,619,270	1.61	0.00138%	2,566,060	6,248,656	2.44	58.9%	40.5%	6,556,321	2.56	4.92%
picojpeg	3,595,305	4,215,381	1.17	14.7%	10.0%	4,215,331	1.17	0.00119%	3,642,847	5,029,281	1.38	27.6%	5.03%	4,752,829	1.30	5.50%
qrduino	2,822,182	3,487,235	1.24	19.1%	4.40%	3,487,185	1.24	0.00143%	3,088,750	5,073,925	1.64	39.1%	11.7%	4,711,397	1.53	7.14%
sglib-combined	2,344,372	3,290,199	1.40	28.7%	8.08%	3,290,149	1.40	0.00152%	2,394,797	3,814,301	1.59	37.2%	8.96%	3,963,025	1.65	3.90%
slre	2,376,079	2,920,434	1.23	18.6%	4.96%	2,920,384	1.23	0.00171%	2,470,357	3,535,047	1.43	30.1%	1.33%	3,446,210	1.40	2.51%
st	3,969,153	4,905,662	1.24	19.1%	4.37%	4,905,612	1.24	0.00102%	2,728,276	3,987,953	1.46	31.6%	0.801%	3,912,536	1.43	1.89%
statemate	2,098,097	2,305,566	1.10	9.00%	17.4%	2,305,516	1.10	0.00217%	1,876,459	2,431,164	1.30	22.8%	11.9%	2,395,081	1.28	1.48%
ud	923,462	2,034,580	2.20	54.6%	41.4%	2,034,530	2.20	0.00246%	1,384,912	2,279,751	1.65	39.3%	11.9%	2,076,639	1.50	8.91%
wikisort	1,056,343	1,413,159	1.34	25.2%	3.57%	1,413,109	1.34	0.00354%	924,018	1,366,640	1.48	32.4%	1.96%	1,336,376	1.45	2.21%
Average	3,179,478	3,896,508	1.29	20.0%	11.9%	3,896,458	1.29	0.00148%	2,884,943	4,057,444	1.45	28.9%	12.1%	4,000,635	1.43	3.88%

CV32E40P	RTL	ISS CPI=1	ISS CPI=1.29	Perf. Sim	CVA6	RTL	ISS CPI=1	ISS CPI=1.29	Perf. Sim
Avg. Error [%]	-	20.0	11.9	0.00148	Avg. Error [%]	-	28.9	12.1	3.88
Sim. Perf. [MIPS]	0.2	104	104	24	Sim. Perf. [MIPS]	0.01	106	106	15

tinyML / Edge Machine Learning Benchmarking

- **MLonMCU target:** Benchmarking tinyML deployment
 - TVM and TFLite micro support
 - ETISS simulator as HW and tuning backend
 - Growing Support for RISC-V-based ASIPs



Agenda

- ETISS: Simulation of Functional Behavior with CIs
- Seal5: LLVM Compiler Support for CIs
- CorePerfDSL: Software Performance Simulation
- **Safety in the presence of HW faults**

Functional Safety Aspects

■ ISO/PAS 21448


- Safety of the Intended Functionality (SOTIF) - Situational awareness: Outlier objects, undiscovered scenes,...
- Algorithmic Safety Techniques: Plausibility, Fallback, etc.

■ ISO 26262

- Systematic faults during development („bugs“)
 - Safety Design Flow, Verification
- Safety in the presence of random HW faults
 - Fault Tolerance (ASIL level A-D)
Error detection, handling, recovery, correction

If undetected:
Silent data corruption (SDC)

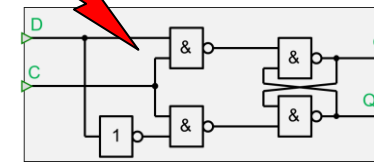
sum=64 → sum=65600



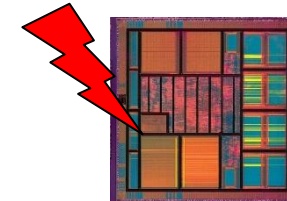
```
for(i=0;i<n;i++){
    sum+=c[i]*b[i];
}
WriteREG(sum,Addr);
```

Corruption of
computation

'0' → '1'

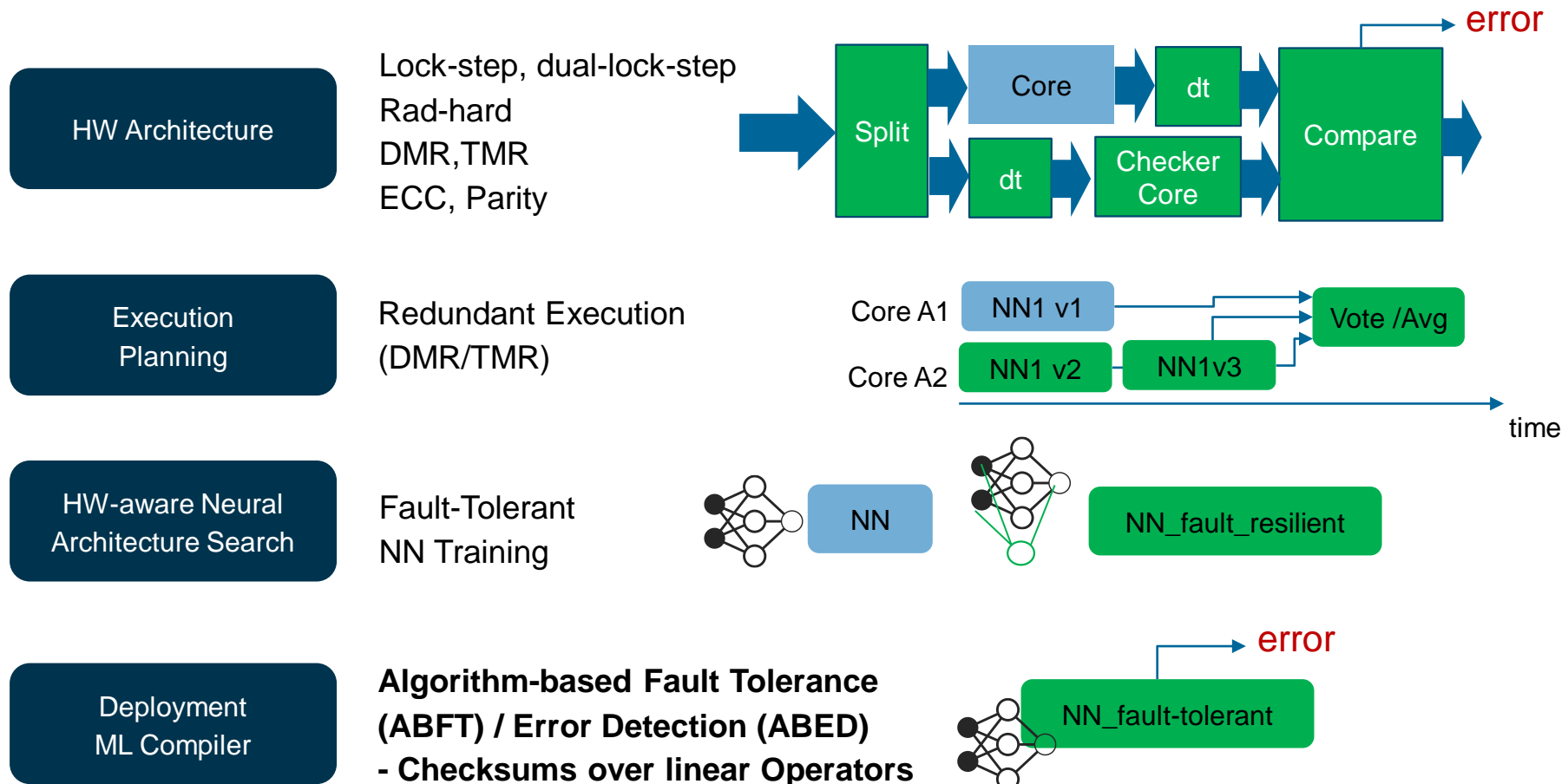


Bit Flip at
Circuit Level



Radiation-
induced
Soft Error

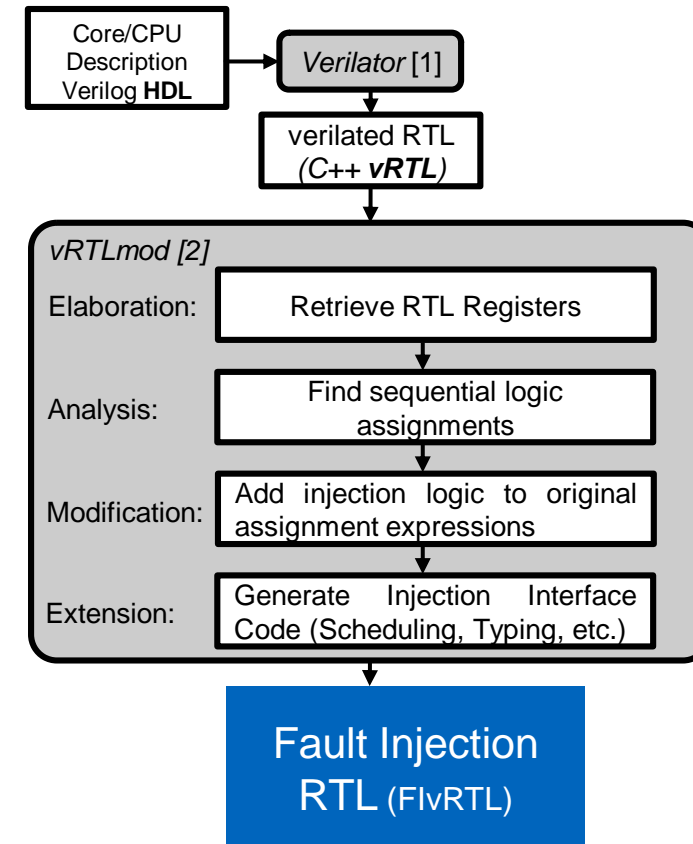
Exploration of Fault Tolerance for AI Workloads



Fault Injection Tools to obtain SDC Rates

- **ETISS Plugin** - ISS Level Fault Injection
 - Bit flips in architectural registers, PC
 - Instruction Skip Error Model
- **vRTLmod** - Verilator RTL Modifier for Flip-Flop-level Fault Injection
 - Bit flips in all micro-architectural registers
 - Small overhead (~10%) compared to plain vRTL

design	variables	bits	CPS ¹ [10 ³]		penalty [%]
			vRTL	vRTLmod	
cv32e40p [3, 4]	419	12272	352.3	300.0	~15
cv32e40s [5]	435	23602	123.4	109.6	~11
cva6 [6, 7]	2239	713056	10.55	9.636	~9



[CF23]



Accelerated Fault Simulation: Differential/Mixed-level

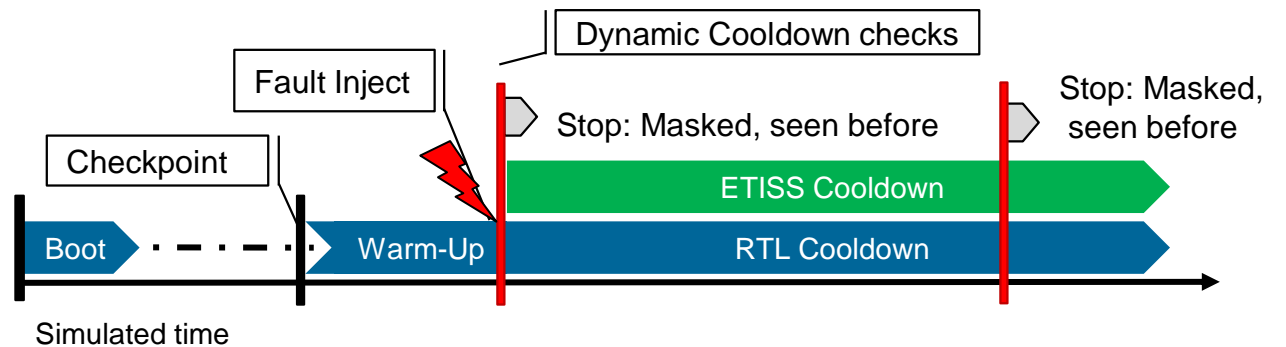
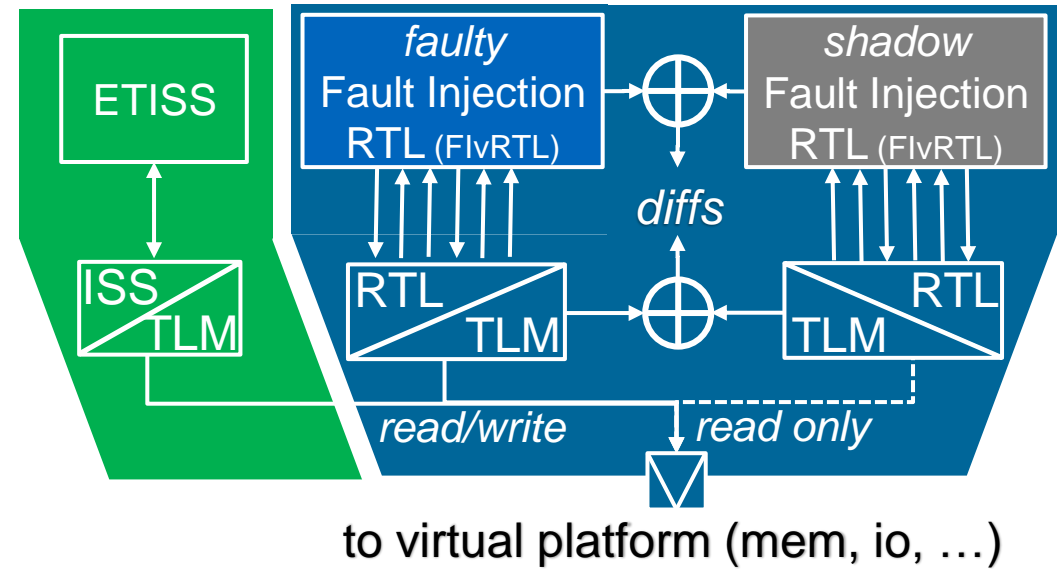
■ Boot Checkpoints

- Skip cycles before Fault Injection point
- Small number of Warm-Up cycles

■ Dynamic Cooldown

- **Stop: Masked / seen before:** If error is known or is fully masked
- **ISS Cooldown:** if error effect is captured in ISS state
- **RTL Cooldown:** else

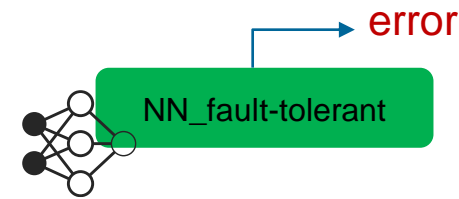
Speed ups ~4-6x:
2 Weeks -> 3 days
– no accuracy loss



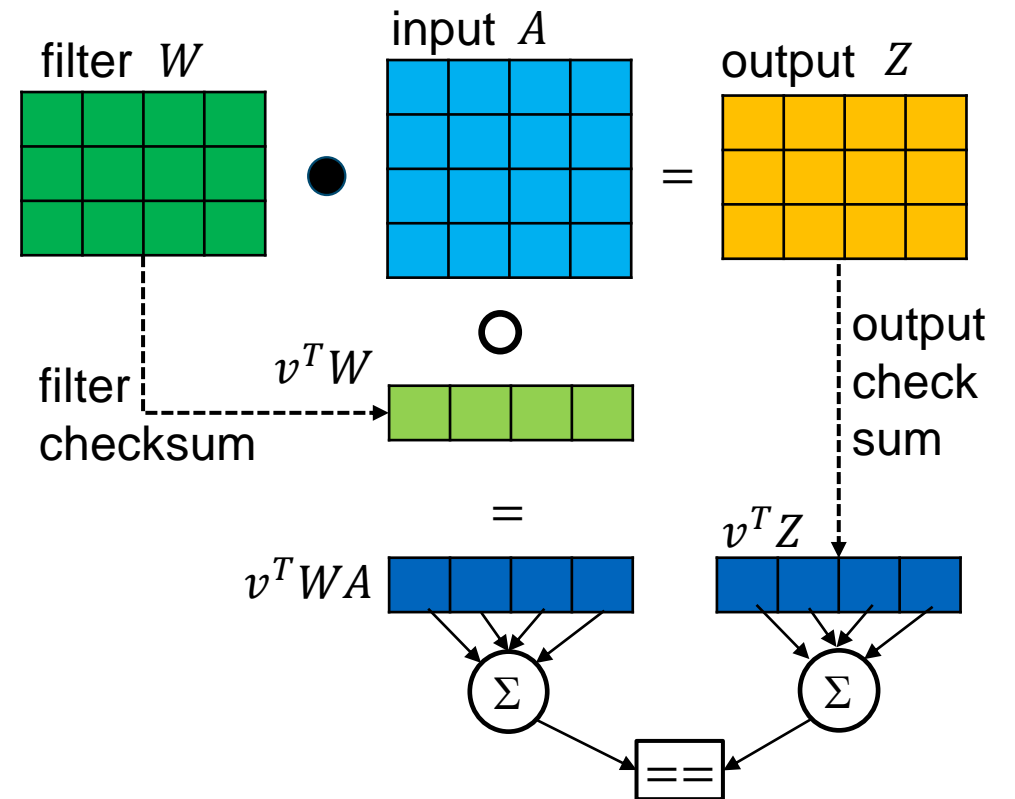
Algorithm-based Error Detection for AI Workloads

Deployment
ML Compiler

Algorithm-based Fault Tolerance
(ABFT) / Error Detection (ABED)
- Checksums over linear Operators



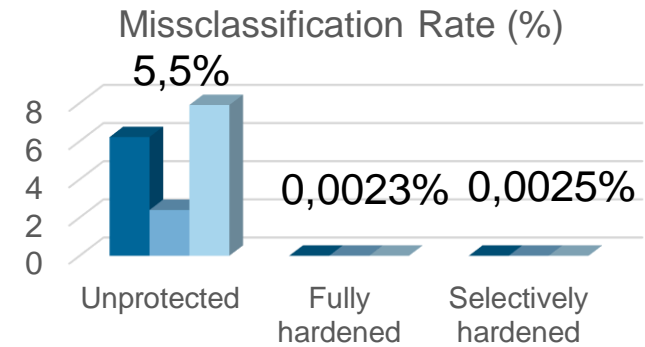
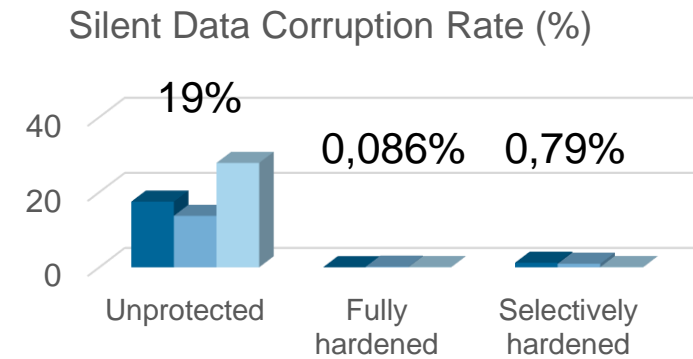
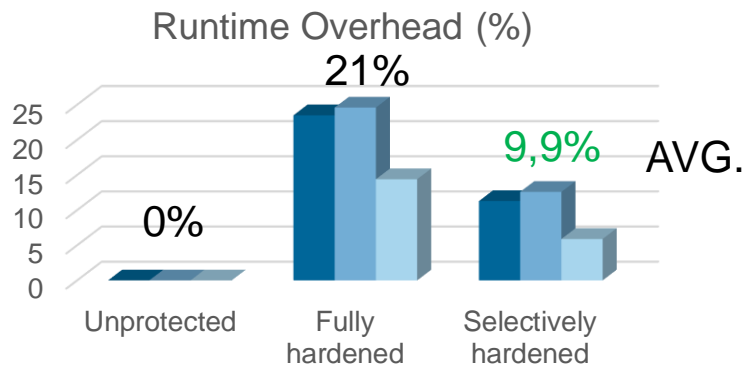
- ABFT uses checksums to find random errors in linear algebra operations (e.g. matrix-matrix-multiply) [1]
- NVIDIA: Filter and input fmap checksum (FIC) [2]
 - Efficient implementation for convolutions
- Less than 2x runtime / energy overhead



- [1] Huang et al. "Algorithm-based fault tolerance for matrix operations," IEEE Transactions on Computers, 1984.
- [2] Hari et al. "Making convolutions resilient via algorithm-based error detection techniques," IEEE Transactions on Dependable and Secure Computing, 2022.

Exploration of Selective Hardening for AI Workloads

- Three tinyML neural networks :
AWW, VWW, ResNET
- Mixture of instruction duplication and NVIDIA FIC (ABFT method)
- ETISS Fault Injection to obtain SDC rates (RV32 CPU Model)



2200x Improvement

[DATE23]

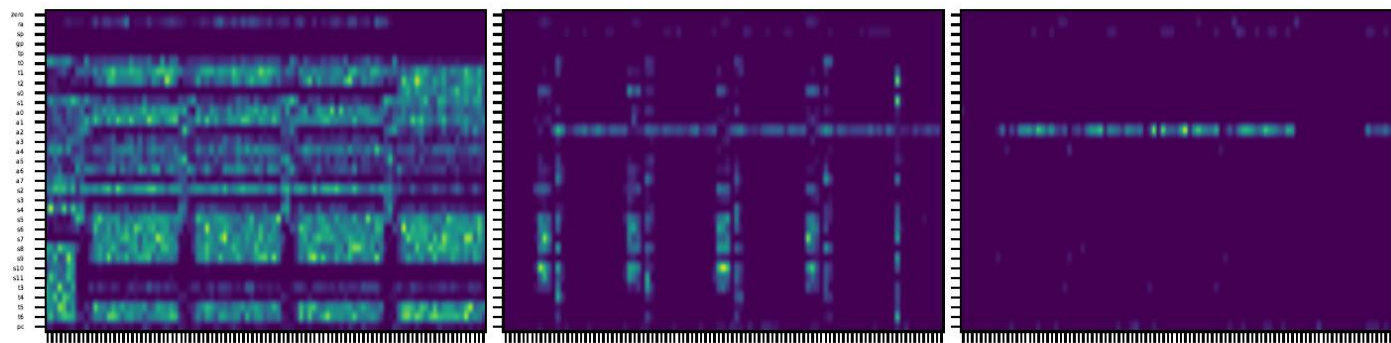
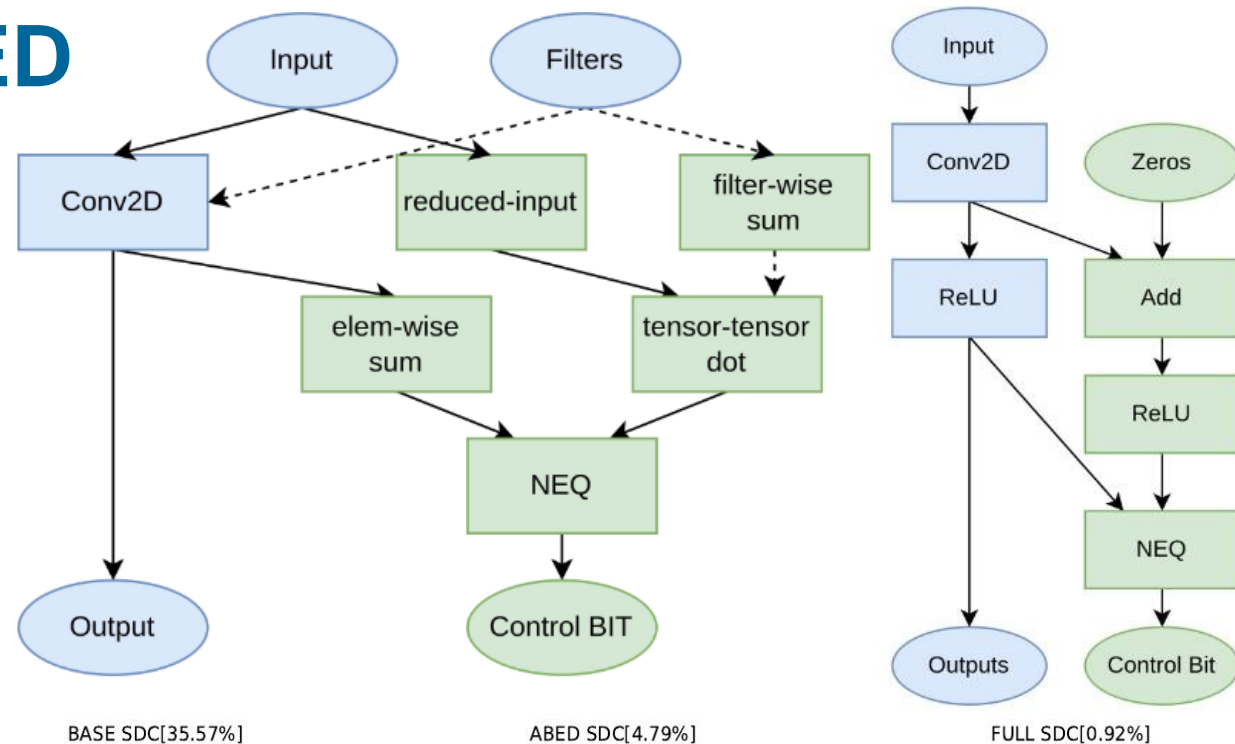
Automated Graph-level ABED

State-of-the-art: Implement ABED in modified kernel library (CUDA, TensorRT, ...)

Ours: Express ABED in computational Graph with standard operators
→ Let ML-Compiler automatically add ABED at graph-level

Idea: Add “cheap” operators to verify computation.

- Avoid Dual-Module Redundancy (DMR)
- Checksums for linear operators
- Fallback to DMR for non-linear (activations)



AWW tinyML on RISC-V 32-bit

Agenda

- ETISS: Simulation of Functional Behavior with CIs
- Seal5: LLVM Compiler Support for CIs
- CorePerfDSL: Software Performance Simulation
- Safety in the presence of HW faults
- **Open Source Tools and References**

Open Source ESL Tools for RISC-V

RISC-V Processor Simulation and Fault Injection

- **ETISS**: Extensible Translating Instruction Set Simulator (ETISS) + **M2ISAR**: ETISS Generator for RISC-V Custom Extensions
 - <https://github.com/tum-ei-eda/etiss>
 - <https://github.com/tum-ei-eda/M2-ISA-R>
 - [The extendable translating instruction set simulator \(ETISS\) interlinked with an MDA framework for fast RISC prototyping \(RAPID 2017\)](#)
- **vrtlmod**: Fault Injection Environment based on Verilator
 - <https://github.com/tum-ei-eda/vrtlmod>
 - [vRTLmod: An LLVM based Open-source Tool to Enable Fault Injection in Verilator RTL Simulations \(Computing Frontiers 2023\)](#)

RISC-V LLVM Compiler extensions

- **Seal5**: LLVM Patch Generator for RISC-V Custom Extensions
 - <https://github.com/tum-ei-eda/seal5>
 - [Seal5: Semi-automated LLVM Support for RISC-V ISA Extensions Including Autovectorization \(to appear MATTER-V/DSD 2024\)](#)
- **COMPAS**: LLVM fault tolerance for RISC-V (e.g. instruction duplication)
 - <https://github.com/tum-ei-eda/compas-ft-riscv>
 - [Compas: compiler-assisted software-implemented hardware fault tolerance for risc-v \(MECO 2022\)](#)

tinyML / Edge Machine Learning on RISC-V

- **MURISCV-NN**: ML Kernels for RISC-V (CMSIS-NN port for RISC-V P+V)
 - <https://github.com/tum-ei-eda/muriscv-nn>
 - [muRISCV-NN: Challenging Zve32x Autovectorization with TinyML Inference Library for RISC-V Vector Extension \(Computing Frontiers 2024\)](#)
- **MLonMCU**: tinyML Deployment with TVM on RISC-V
 - <https://github.com/tum-ei-eda/mlonmcu>
 - [MLonMCU: TinyML Benchmarking with Fast Retargeting \(CODAI 2022, appeared CODAI 2023\)](#)

Thanks

- Contributors
 - Marc Greim
 - Uzair Sharif
 - Rafael Stahl
 - Philipp van Kempen
 - Karsten Emrich
 - Conrad Foik
 - Johannes Geier
 - Leonidas Kontopoulos
 - Jefferson Parker Jones
 - ...

Thank you for your attention.



Contact

daniel.mueller-gritschneder@tuwien.ac.at
<https://ti.tuwien.ac.at/ecs>



Johannes.Geier@tum.de

