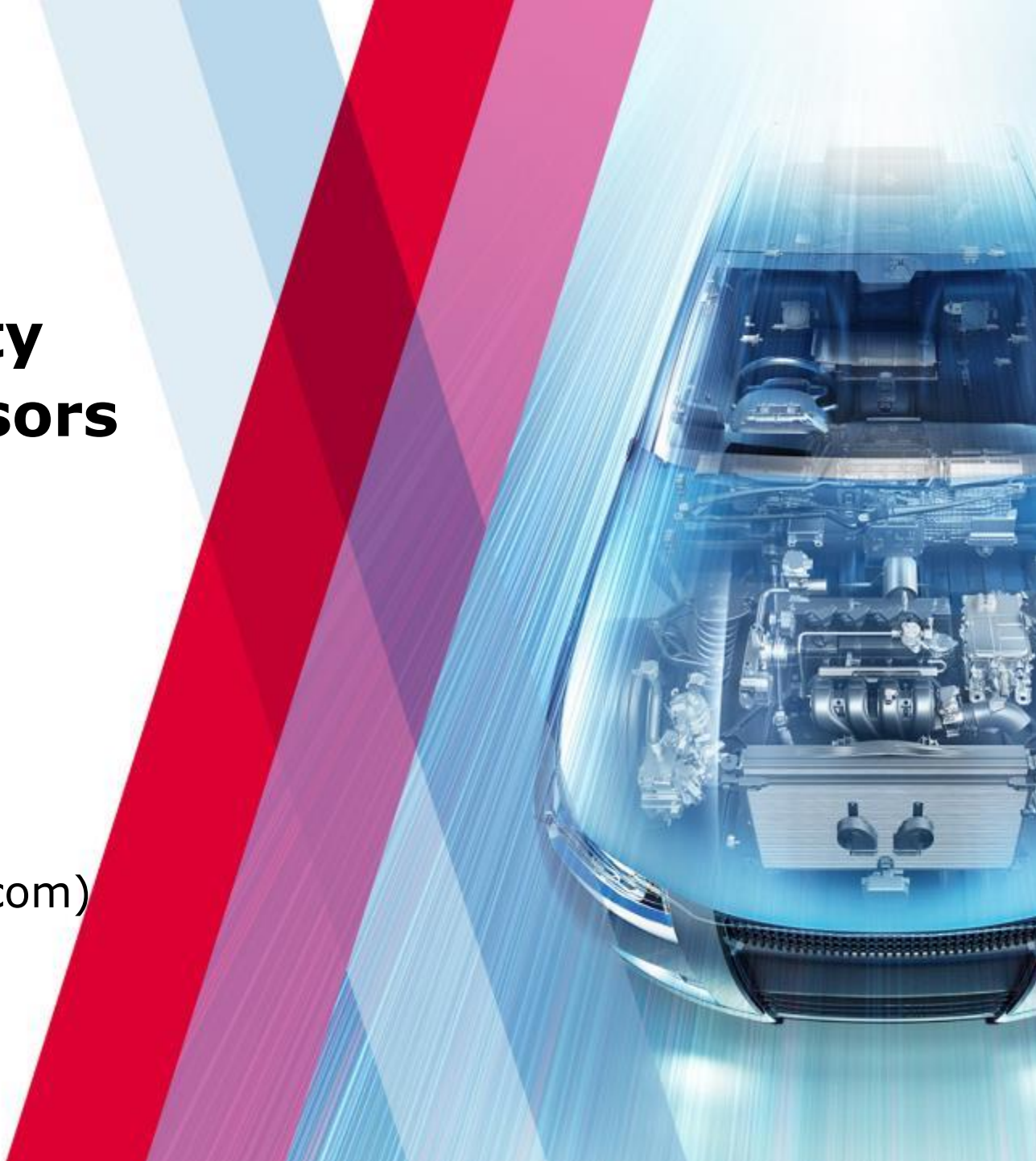
DENSO
Crafting the Core

Formal Verification of Security Properties on RISC-V Processors

Verification Futures Conference 2025 UK

Czea Sie Chuah
Christian Appold (c.appold@eu.denso.com)
Tim Leinmüller
2025/07/01



Outline

1. Introduction
2. Methodology
3. Security Standards
4. Security-Critical Functionality
5. Formal Verification
6. Conclusion

Introduction

Christian Appold – Software R&D

Introduction

DENSO is a globally acting tier 1 automotive supplier

- **One of the three largest tier 1 automotive suppliers worldwide**

Product portfolio:

- Thermal systems (e.g. car air conditioning)
- Powertrain systems (e.g. direct injection pump)
- Electrification systems (e.g. steering motors)
- Advanced Driver Assistance Systems
- Sensors, Semiconductors (e.g. rain sensor)
- Factory automation (e.g. robots)

35 European sites:

- Headquarters: Kariya, Japan
- **165 000 employees** worldwide
- 190 companies belong to DENSO group
- 44 billion euro consolidated net sales
- **3,6 billion euro R&D investments**
- **41 000 patents worldwide**

DENSO in Germany:

- Company name: DENSO AUTOMOTIVE Deutschland GmbH
- **Headquarters: Eching (close to Munich)**
- 820 employees
- Further branches: Aachen, Frankfurt, Köln, Stuttgart, Wolfsburg

RISC-V processor activities:

- DENSO member of RISC-V International
- **Development and selling of own RISC-V processors**
- 32-bit and 64-bit processors

Introduction

- Processors are ubiquituous and embedded in nearly every electronic device
- Security is very important for upcoming applications, e.g. autonomous driving or factory automation
- In past years several famous bugs and security-vulnerabilities in processors found
- Design flaws can be exploited by attackers with drastic consequences
- Reliable pre-silicon verification of security-critical functionality needed
  - Simulation not exhaustive and corner case bugs can be missed

Formal Verification required for Security-Critical functionality

- **We investigated how to perform this formal verification with Jasper**

Introduction

- RISC-V processors targetted by attackers easier due to openness of ISA
- Dispersion of RISC-V processors strongly growing, several small companies exist

Security verification guidance needed

- No previous work did a detailed analysis of RISC-V specifications for security-critical functionality
- **Our Contribution helps:**
  1. **Identification of large set of security-critical functionality for RISC-V processors**
  2. **Derivation of properties for security-critical functionality correctness**
  3. **Optimization of Cadence Jasper usage for verifying the properties**

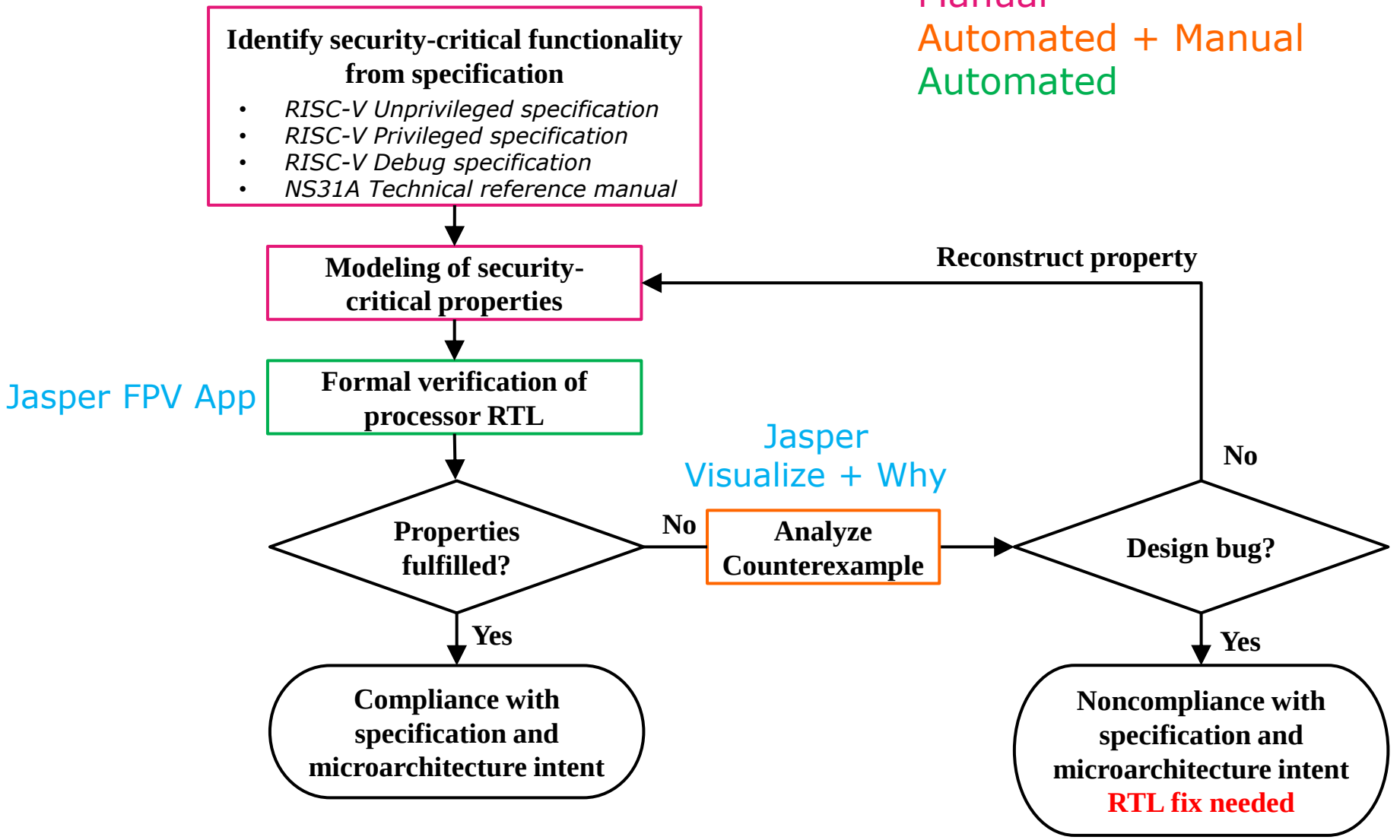
- Our full property set for security-critical functionality published in paper:
  - *Formal Verification of Security Properties on RISC-V Processors*
  - Published 2023 at 21st ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE)

Methodology

Methodology

Workflow

Manual
Automated + Manual
Automated



Methodology

- We identified security-critical functionality by investigating
  - RISC-V architecture for protection mechanisms for security (e.g. PMP)
  - Existing malicious attacks from literature
    - E.g. attacks from following paper:
      - *SoK: Eternal War in Memory (2013 IEEE Symposium on Security and Privacy)*
  - Identified approach to enumerate possible other attacks

Security Standards

Christian Appold – Software R&D

Security Standards

- Common Criteria is an international standard for independent security evaluation and certification
  - For IT products implemented as hardware, firmware or software
  - It introduces seven different levels of evaluation (EAL1 to EAL7)
  - **EAL7 is the highest assurance level and requires formal methods for design and implementation verification**
    - **for extremely high-risk environments where utmost security is critical**

Common Criteria requires formal verification when utmost security is critical
- **We give guidance how to do this for security-critical processor functionality**

Security Standards

- Common Weakness Enumeration (CWE)
  - A community-developed list of common software or hardware weaknesses
  - For hardware it lists weaknesses around concepts that are frequently encountered in hardware design
- There is overlap between weaknesses described in CWE and our work
  - RISC-V specification describes what and how functionality should be implemented
    - some of the functionality protects against CWE weaknesses

Our properties prove security-critical functionality behavior which protects against CWE elements

- Example CWE category where our properties help protecting against issues:
  - Privilege Separation and Access Control Issues, e.g.
    - incorrect defult permissions (Our Properties: Mode Transition, Exception and Interrupt)
    - improper isolation of shared resources on System-on-a-Chip (SoC) (Our Properties: CSR, Debug Operation, Mode Transition, PMP)

Security-Critical Functionality

Christian Appold – Software R&D

Security-Critical Functionality

- We identified security-critical functionality and grouped it in 9 categories:

Instruction Execution CFU, CCF, DoS *Check instruction flow through pipeline*	**CSRs** CCF, DoS, PE *Comply with CSR access rules*	**Debug Operation** CCF, PE *Comply with Debug CSR access rules*
Exception and Interrupt CCU, DoS, PE *Proper handling required*	**Mode Transition** PE *Mode transition rules need to be met*	**PMP** CFU, CCF, DoS *Access control rules for memory regions need to be met*
Control Flow CCF, DoS *Correct setting of program counter*	**Register Update** CFU, CCF, DoS *Correct target register is updated*	**Memory Access** CFU, CCF, DoS *Value and address memory transfers as intended*

Our properties can detect bugs in the hardware

- malicious behavior exploiting them can be triggered e.g. by software execution

Possible security issues, when properties not checked:

- change of functionality (CFU)
- change of control flow (CCF)
- denial of service (DoS)
- privilege escalation (PE)

Security-Critical Functionality

- Control Flow category:
  - Prove correct setting of Program Counter (PC)
  - Each setting of PC is verified, several possibilites (one property for each)
    - reset PC, standard PC increment (PC+4), interrupt, exception, due to last occuring instruction (jal, jalr, branch, mret, debug return)
  - Possible security issues when not checked:
    - change of control flow (e.g. PC modification to attacker code execution)
    - denial of service (e.g. disable proper instruction flow at all)

Property Example: Correct PC after mret instruction (FAILING, found bug)

```
assert property (@(posedge clk) disable iff (rst)
        !firstInstruction && write_into_wb_pipeline_register && last_instruction_mret
        |-> instruction_PC_to_wb_pipeline_register == mepc_register_PC);
```

PC checks at writing PC in writeback (wb) stage pipeline register

mepc stores trap return PC

Security-Critical Functionality

- Control and Status Register (CSR) category:
  - Properties check compliance with CSR access rules
  - Possible security issues when not checked:
    - change of control flow (PC for exception return modified)
    - denial of service (e.g. interrupts (permanently) disabled)
    - privilege escalation (e.g. modify stored previous privilege level)

CSR read and write operand behavior

Register operand				
Instruction	rd	rs1	read CSR?	write CSR?
CSRRW	x0	-	no	yes
CSRRW	!x0	-	yes	yes
CSRRS/C	-	x0	yes	no
CSRRS/C	-	!x0	yes	yes
Immediate operand				
Instruction	rd	uimm	read CSR?	write CSR?
CSRRWI	x0	-	no	yes
CSRRWI	!x0	-	yes	yes
CSRRS/CI	-	0	yes	no
CSRRS/CI	-	!0	yes	yes

bugs here

Property Example: (FAILING, found bug)

- CSR write access is done only for CSR instructions with register and immediate operand as shown in the table
- Example for mstatus CSR, used also for the other CSRs

```
assert property (@(posedge clk) disable iff (rst)
    !(CSR_instruction && additional_Conditions_Fulfilled) |-> !(WriteEnable_mstatus_register))
```

Formal Verification

Christian Appold – Software R&D

Formal Verification

- Verification executed on 16-core AMD server with 2.2GHz each and 128 GB RAM
- All verification runs executed with RISC-V processor core as top-level module
- Approximate design information:

  Total number (Sum of Bits)
  - Gates: 84000 (495000)
  - RTL lines: 49000
- Did one Jasper verification run for each property category
  - all properties in a category verified by Jasper in parallel
- Jasper contains formal verification engines for full-proofs and bug-hunting
- Use of several different engines in parallel for a verification run
  - automatic choosing of most suitable verification engine by Jasper

Formal Verification

- We identified 1146 properties and grouped them under 9 categories
- Achieved full-proof for passing properties
- Runtime for one property:
  - Control Flow:

    < 24h with blackboxing
  - Other properties: < 4000s
- 3 design bugs found

Formal Verification

Categories	Properties		
	Assertion	*Pass*	*Fail*
Instruction Execution	10	10	0
CSR	394	280	114
Debug Operation	14	14	0
Exception and Interrupt	87	87	0
Mode Transition	13	13	0
PMP	574	574	0
Control Flow	9	8	1
Register Update	33	33	0
Memory Access	12	12	0
Total	**1146**	**1031**	**115**

- Bug 1 and Bug 2 occur repeatedly for each CSR
- Bug 1:
  - violation of specification, no read to CSR for CSR instructions when target register (rd) is register 0
- Bug 2:
  - violation of specification
    - no write to CSR for CSR instructions when source register 1 (rs1) is register 0 or immediate operand is 0
- Bug 3:
  - wrong PC taken in debug mode, when mret instruction in writeback stage and exception occurs

Conclusion

Conclusion

- Proliferation of RISC-V processors strongly growing
  - Expected that number of discovered security vulnerabilities in them will grow
  - Methods to ensure high security-level urgently needed
- We identified, categorised and formally verified a large set of security-critical properties for RISC-V processors
- Formal verification scales well for our property set
- Identified design bugs, which have not been found by other verification approaches
- **Our work gives guidance for RISC-V security hardening**
- **Helps to significantly increase the security-level of RISC-V processors**
- **Properties and presented security formal verification used in our processor development**

DENSO

Crafting the Core