2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

How the Right Mindset Increases Quality in RISC-V Verification

Philippe Luc, Director of Verification, Codasip

Salaheddin Hetalani, Field Application Engineer, Siemens EDA

codasip SIEMENS

accellera SYSTEMS INITIATIVE

RISC-V – Why is it important to today?

- Often called “open source”… but “open standard” is better
- Standardized ISA
  - With a wide range of extensions (FP, DSP, Vector etc.)
  - Ability to customize ISA for user applications

- Strong ecosystem
  - Tools support - (LLVM, Lauterbach, IAR etc.)
  - Frameworks - (Linux, Android, AI/ML, Hypervisor…)

- Both free (open source) and commercial cores

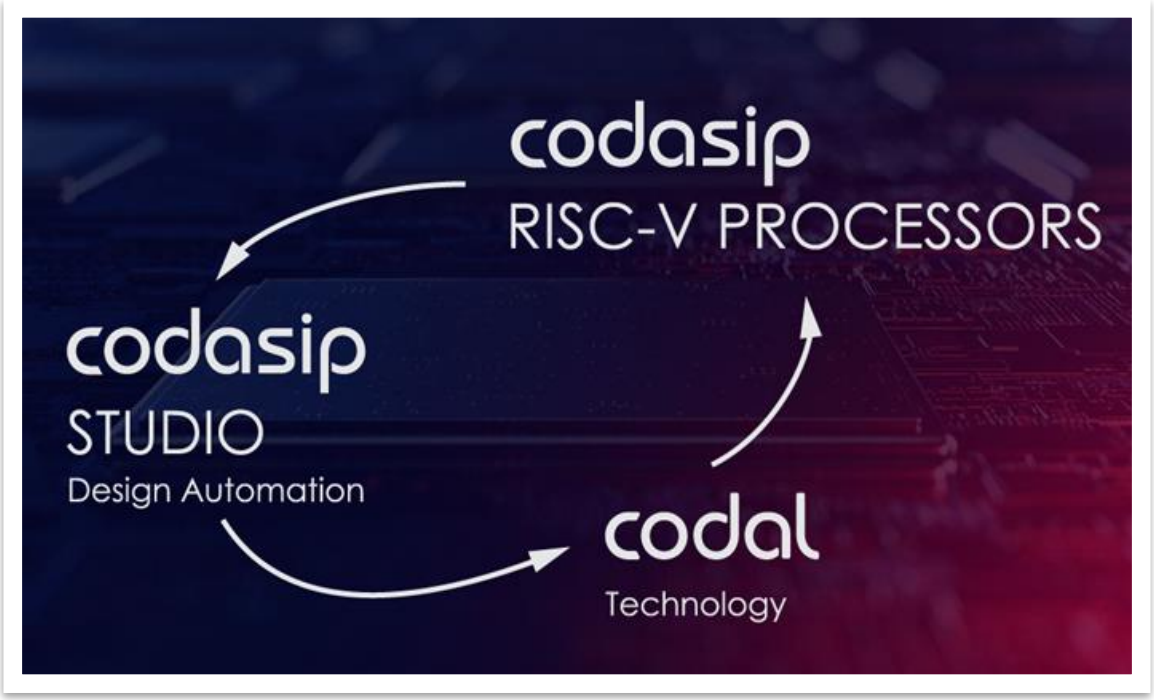
RISC-V in the future

- Market is pushing towards open standards
- Customers need a better way to address their applications
  - Additional RISC-V Extensions
  - Users doing their own customization of the ISA

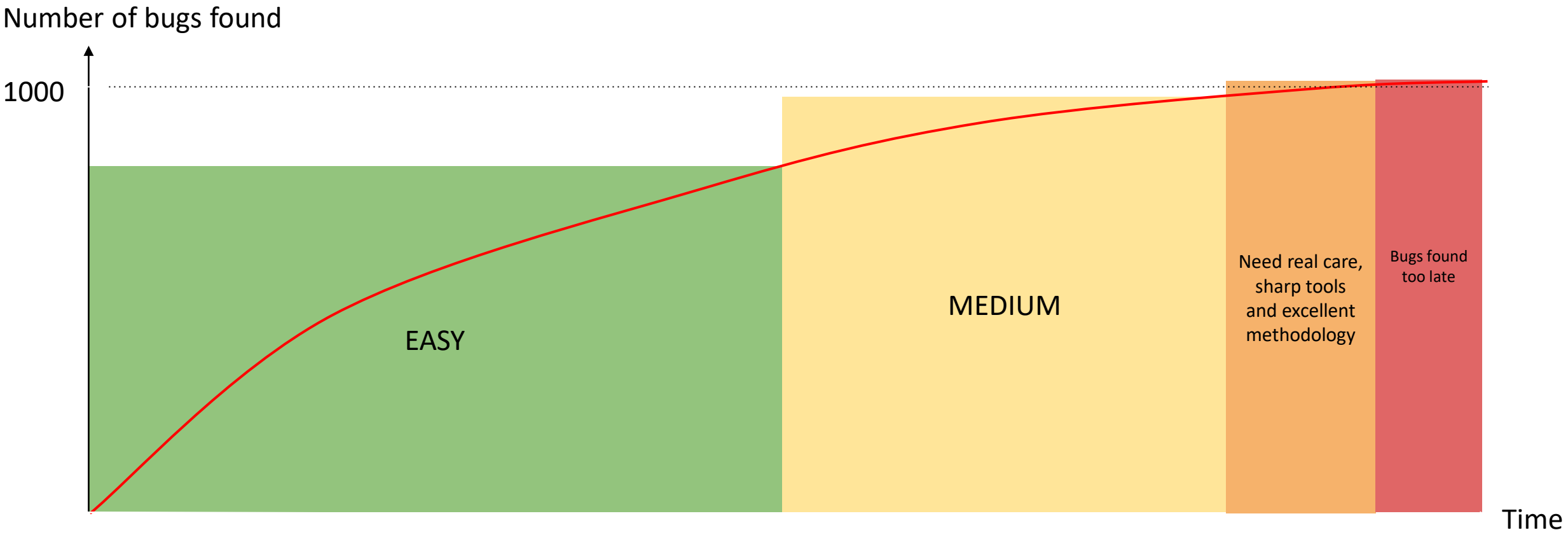
Codasip supports the open ISA and has automation for customization

Codasip

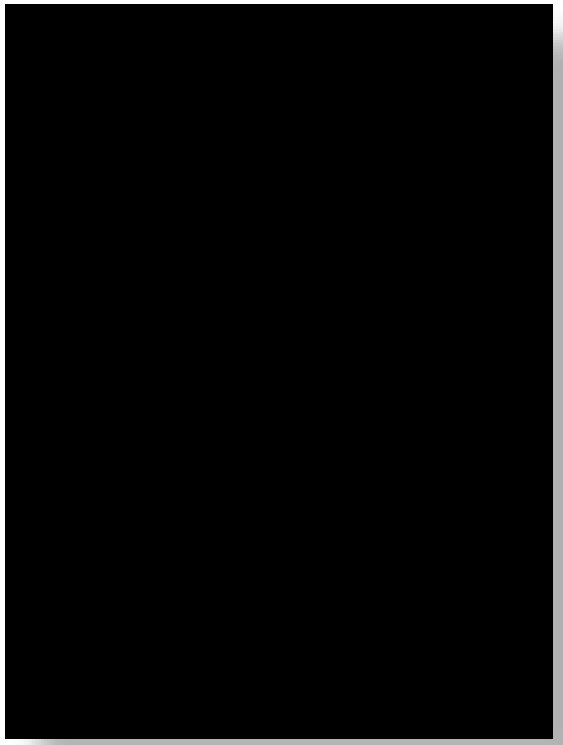
We do RISC-V with a twist



Bugs' life


Number of bugs found
1000
EASY
MEDIUM
Need real care, sharp tools and excellent methodology
Bugs found too late
Time

Where to search for bugs



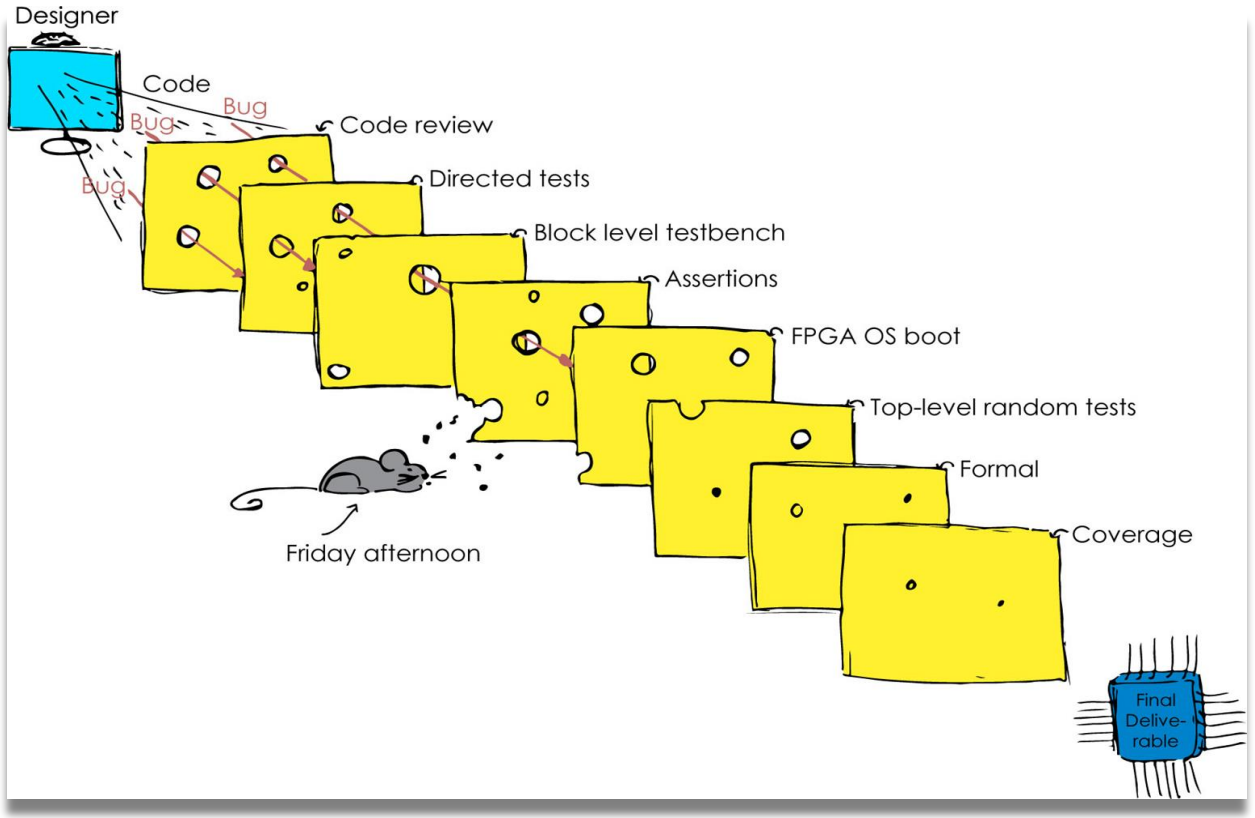
Everywhere

Lighted spaces

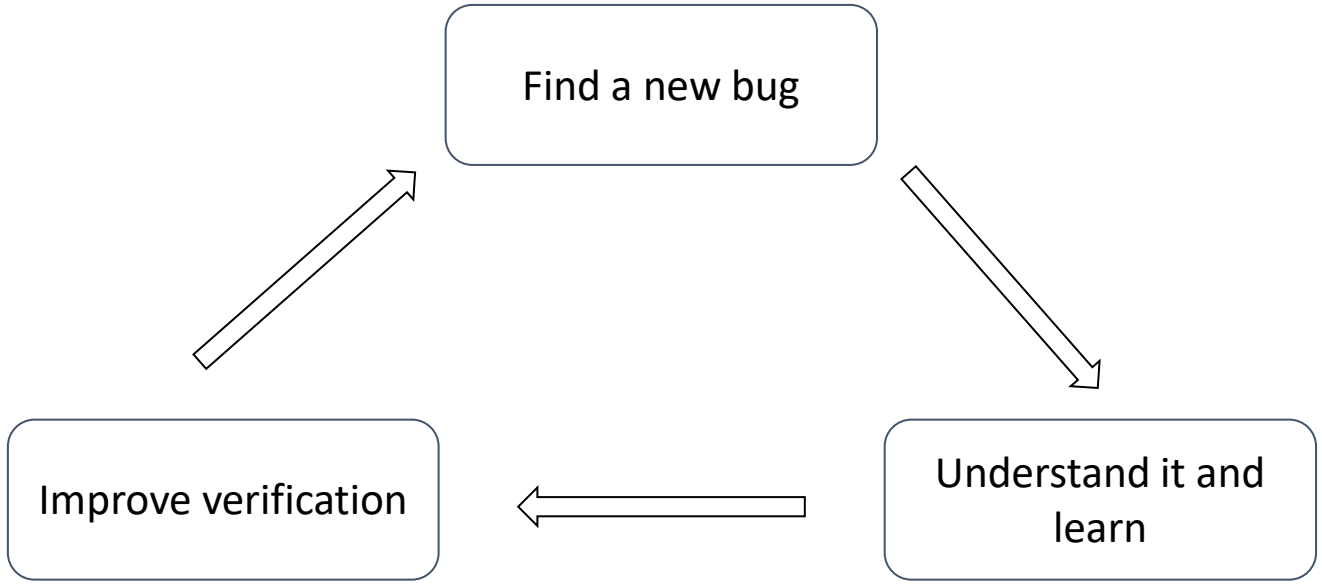
Customer request: Make an unbreakable chain

- RISC-V is open standard
  - Same baseline ISA
  - Optional standard extensions
  - Custom extensions
  - Different microarchitectures
- Test the chain
  - Pull it
  - Shake it
- Check it is pulled and shaken

Verification flow: plugging the holes



Fix bugs, and improve verification



Codasip strategy with Siemens

- Codasip started using formal methodology thoroughly
  - Consistency checks
  - Siemens “Processor Verification App”
    - Dedicated OneSpin formal checker for RISC-V architecture

```
property RESET_p;
reset_sequence |=>
    t##1 !(ifu.pma_fetch_s_fault_o_Q|ifu.pmp_fetch_s_fault_o_Q) and
    during(t, nxt(t,4), rst_n)
implies
    t##3 Ready2Execute and
    t##0 !fetch_req_valid ##1 fetch_req_valid && fetch_req_PC=={l31_32IMFCB_UPTIDpbid_ca_core_t.p_boot_addr_Q[31:2],2'b00} and
    t##4 pc_check({l31_32IMFCB_UPTIDpbid_ca_core_t.p_boot_addr_Q[31:2],2'b00}) and
    t##4 all_regs_valid(Arch) && Arch.X[0].data=='0 and
    t##4 RISCV_CSR_reset_state(Arch.CSR) and
    t##0 !dmem_req_valid [*4] and
    t##3 right_hook;
endproperty
```

- For a first use Codasip tried in a core that was in development

OneSpin's industry proven solutions

The content of this article was presented at DVCon 2007 and is posted with DVCon's permission.

Complete Formal Verification
of **TriCore2** and Other Processors

Infineon technologies

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Complete Formal Verification
of a Family of **Automotive DSPs**

BOSCH
Invented for life

RENESAS

Formal Verification Applied to the Renesas
MCU Design Platform

51 DESIGN AUTOMATION CONFERENCE

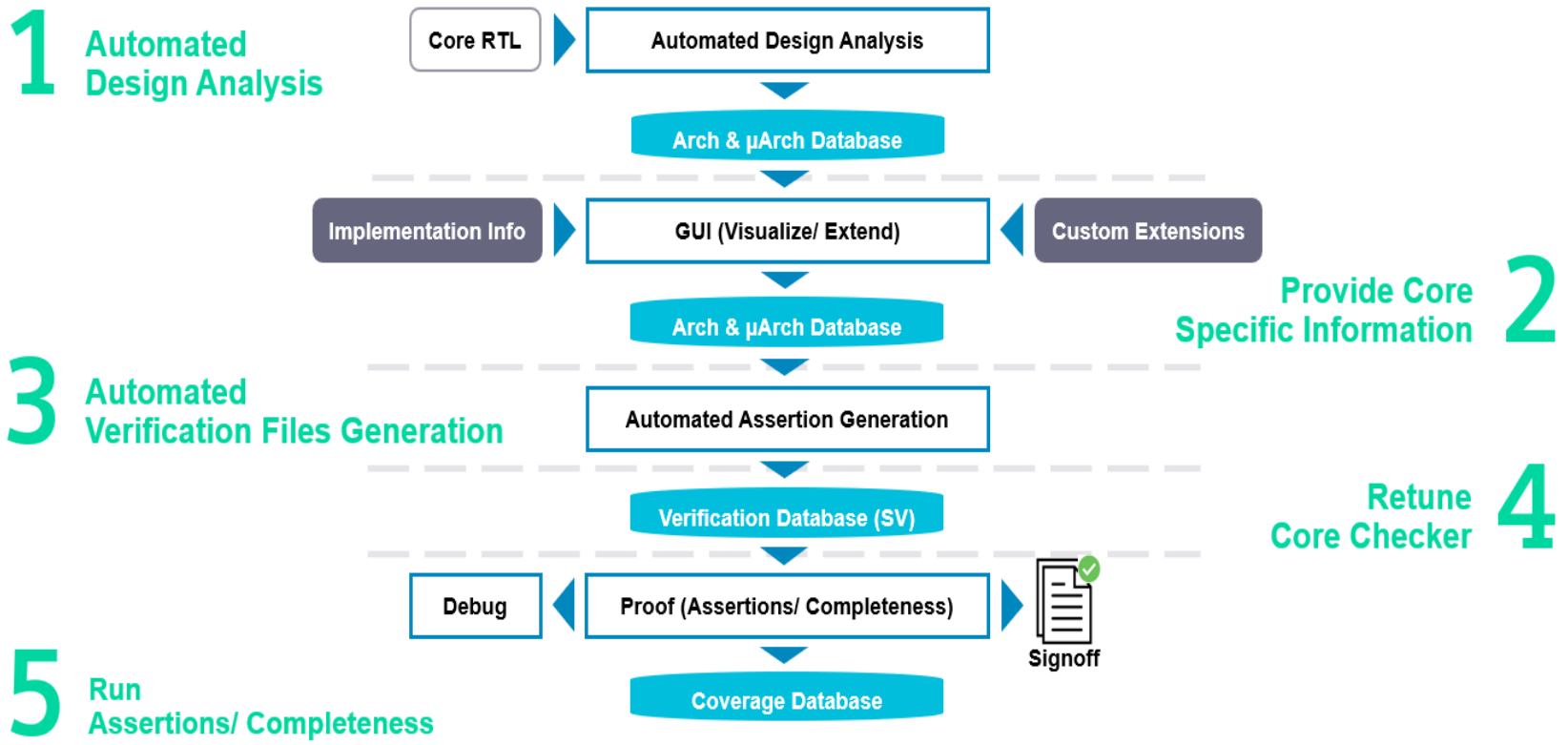
ECI EDAPTIVE COMPUTING, INC.

Complete Formal Verification of **RISC-V**
Processor IPs for **Trojan-Free Trusted ICs**

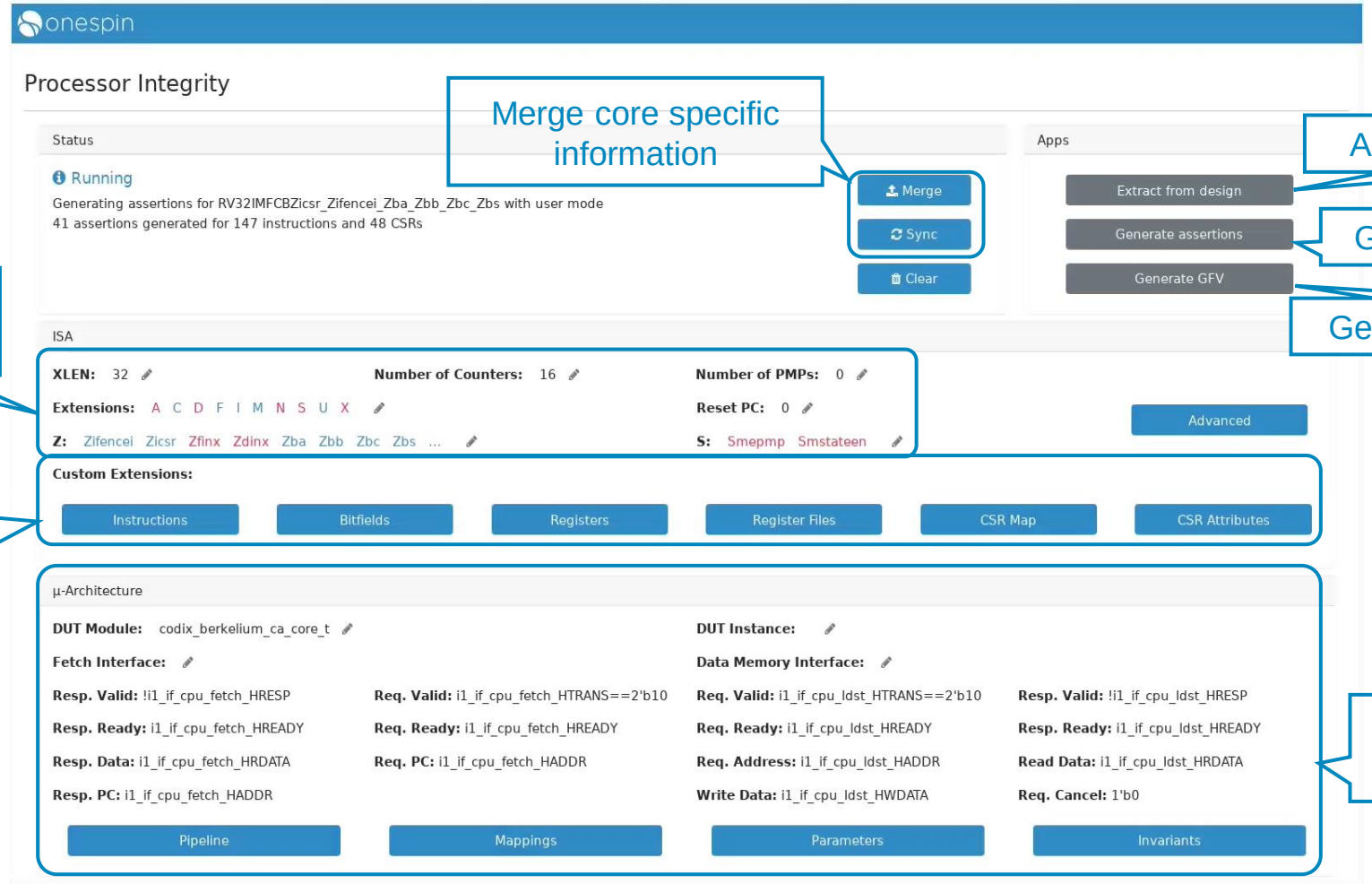
GOMACTech
2019

GOMAC Tech

OneSpin's processor verification app – Flow



Processor verification app – GUI



Merge core specific information

Automated design analysis

Generate assertions

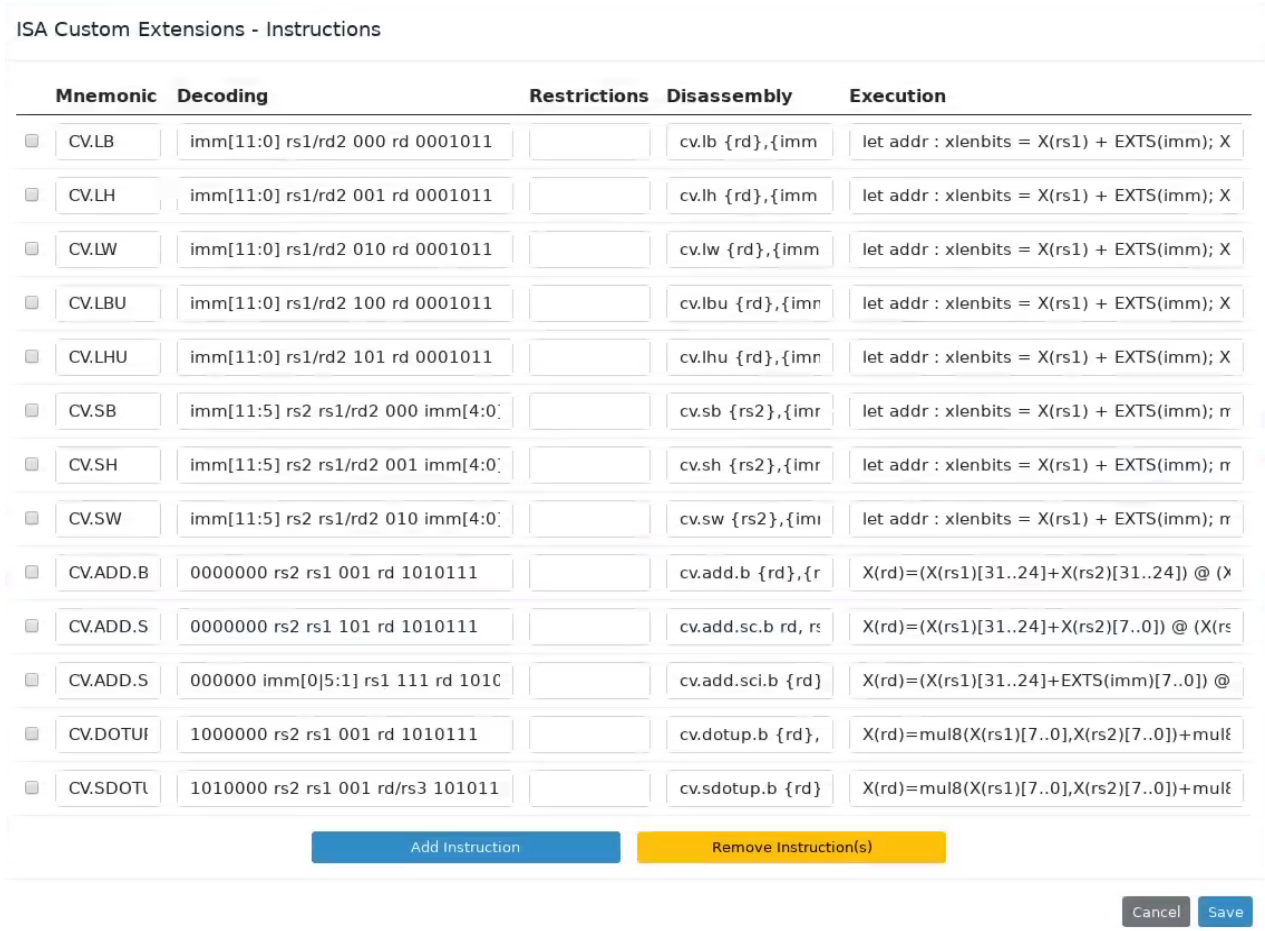
Generate completeness plan

Design ISA information

Custom/ User extensions

Design micro-architecture information

Processor verification app – Custom Instructions



Processor verification app – Generated Assertions

Name	Proof Status	Witness Status	Case Split Status	Validity	Source
!	! <any stat	! <any statu	! <any status	! <any	!
Constraints					
Properties					
RV_chk.ops.BUBBLE_a	open	open	open	up_to_date	core_checker.sv:735
RV_chk.ops.INTR_Handle_a	open	open	open	up_to_date	core_checker.sv:740
RV_chk.ops.RESET_a	open	open	open	up_to_date	core_checker.sv:734
RV_chk.ops.RV32I.ARITH_a	open	open	open	up_to_date	core_checker.sv:745
RV_chk.ops.RV32I.BRANCH_a	open	open	open	up_to_date	core_checker.sv:744
RV_chk.ops.RV32I.Debug.EBREAK_ForcedEntry_a	open	open	open	up_to_date	core_checker.sv:757
RV_chk.ops.RV32I.Debug.EBREAK_HaltReq_a	open	open	open	up_to_date	core_checker.sv:756
RV_chk.ops.RV32I.EBREAK_BreakPoint_a	open	open	open	up_to_date	core_checker.sv:753
RV_chk.ops.RV32I.ECALL_a	open	open	open	up_to_date	core_checker.sv:752
RV_chk.ops.RV32I.FENCE_a	open	open	open	up_to_date	core_checker.sv:749
RV_chk.ops.RV32I.JUMP_a	open	open	open	up_to_date	core_checker.sv:743
RV_chk.ops.RV32I.MEM_a	open	open	open	up_to_date	core_checker.sv:746
RV_chk.ops.RV32I.MEM_MultiAccess_a	open	open	open	up_to_date	core_checker.sv:747
RV_chk.ops.RV32I.WFI_a	open	open	open	up_to_date	core_checker.sv:750
RV_chk.ops.RV32I.xRET_a	open	open	open	up_to_date	core_checker.sv:751
RV_chk.ops.RV32M.DIV_a	open	open	open	up_to_date	core_checker.sv:792
RV_chk.ops.RV32M.MUL_a	open	open	open	up_to_date	core_checker.sv:791
RV_chk.ops.RVC.ARITH_a	open	open	open	up_to_date	core_checker.sv:905
RV_chk.ops.RVC.BRANCH_a	open	open	open	up_to_date	core_checker.sv:904
RV_chk.ops.RVC.JUMP_a	open	open	open	up_to_date	core_checker.sv:903
RV_chk.ops.RVC.MEM_a	open	open	open	up_to_date	core_checker.sv:906
RV_chk.ops.RVC.MEM_MultiAccess_a	open	open	open	up_to_date	core_checker.sv:907
RV_chk.ops.RVX.instr_CV_LB_2_a	open	open	open	up_to_date	custom_extensions.sv:1802
RV_chk.ops.RVX.instr_CV_LB_a	open	open	open	up_to_date	custom_extensions.sv:1801
RV_chk.ops.RVZicsr.CSRx_a	open	open	open	up_to_date	core_checker.sv:777
RV_chk.ops.RVZifencei.FENCE_I_a	open	open	open	up_to_date	core_checker.sv:773
RV_chk.ops.XCPT_IF_ID_a	open	open	open	up_to_date	core_checker.sv:737
RV_chk.ops.XCPT_MEM_a	open	open	open	up_to_date	core_checker.sv:738
RV_chk.ops.XCPT_WB_a	open	open	open	up_to_date	core_checker.sv:739
SVA Named Properties					
SVA Sequences					

Interrupts

I - Base Integer

M

C

X

Zicsr & Zifencei

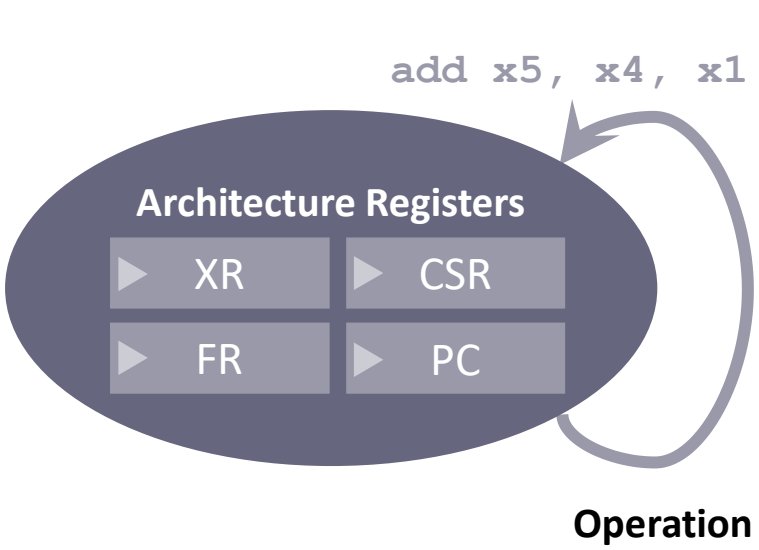
Exceptions

RV32IMC_Zicsr_Zifencei

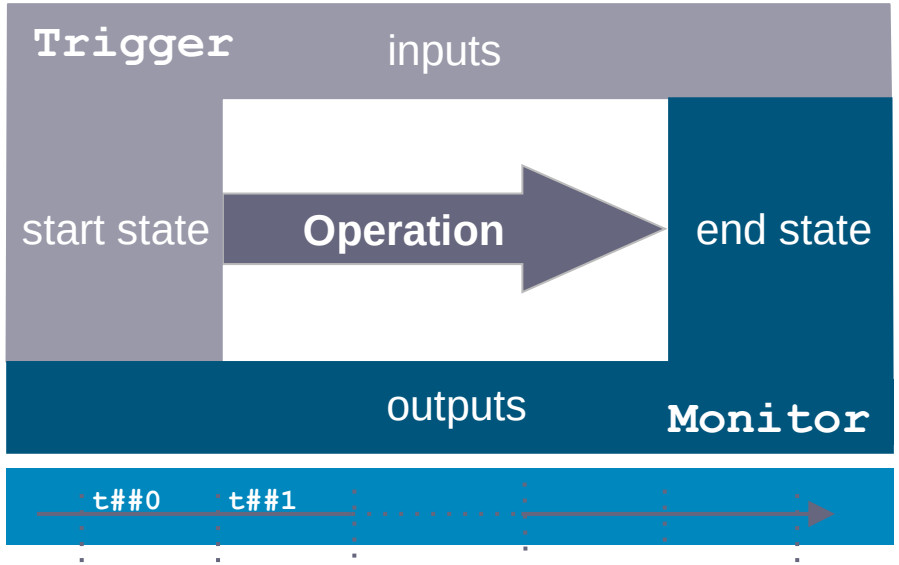
27

Assertions

OneSpin 360's unique modelling principle



Operation




```
property add_p;
    t_id ##0 start_state and
    t_id ##0 add_instr   and
    ...
implies
    ...
    t_wb ##1 x5==result  and
    t_wb ##0 end_state;
endproperty

add_a: assert property(add_p);
```

Operational Assertion

Siemens findings at Codasip (1/2)

Consistency Checks/Dead Code - Found a signal that was hardwired

```
// "Virtual" EX1 stage handling
if (s_ex1_clear_i)
    pipe.EX1.clear();
else if (s_ex1_stall_i)
    pipe.EX1.stall();
```

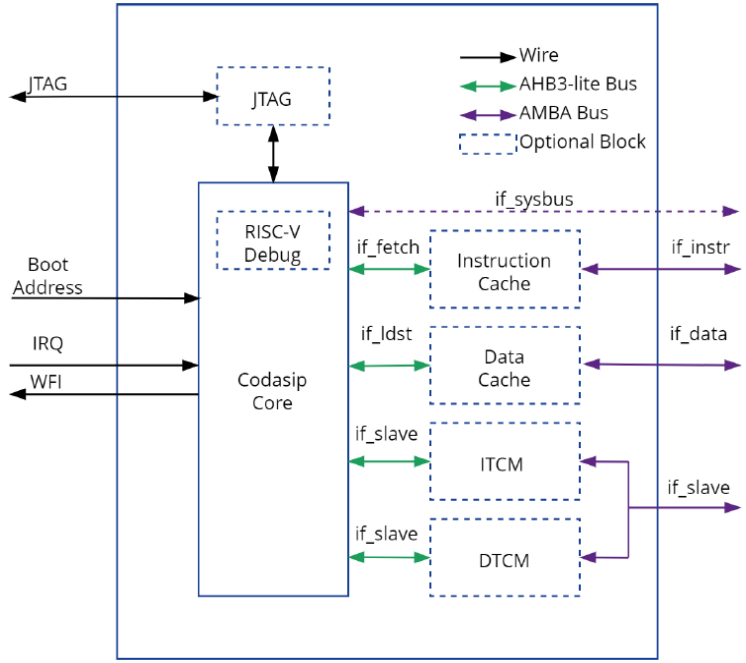
```
// pipeline controller
plc.s_flush_i     = s_flush;
plc.s_id_clear_i  = s_id_clear;
plc.s_ex_stall_i  = s_ex_stall;
plc.s_ex_clear_i  = s_ex_clear;
plc.s_ex1_stall_i = s_ex1_stall;
plc.s_ex1_clear_i = s_ex1_clear;
```

```
// EX1 stage
s_ex1_stall = ex1_stall;
s_ex1_clear = false;
```

Siemens findings at Codasip (2/2)

- Illegal instruction exceptions not raised
- Illegal CSR counter increment
- Legal instructions treated as illegal
- Wrong settings of floating-point flags, memory accesses, and program counter
- Storing the wrong address in mtval in case of a misaligned memory access exception.

RV32IMFCZicsr_Zifencei_Zba_Zbb_Zbc_Zbs



Conclusion

- Consistency checks and “Super” linting can find errors earlier
- OneSpin Processor Verification App allows a very efficient, additional way to verify
- Fixing issues earlier in the verification process than standard UVM
- Shorter verification times than the standard functional based approach
- Good complement for any verification methodology

Questions