



JUNE 23-27, 2024

MOSCONE WEST CENTER  
SAN FRANCISCO, CA, USA

# Securing Safe Customization of RISC-V Cores via Rigorous Sequential RTL Comparison



# Agenda

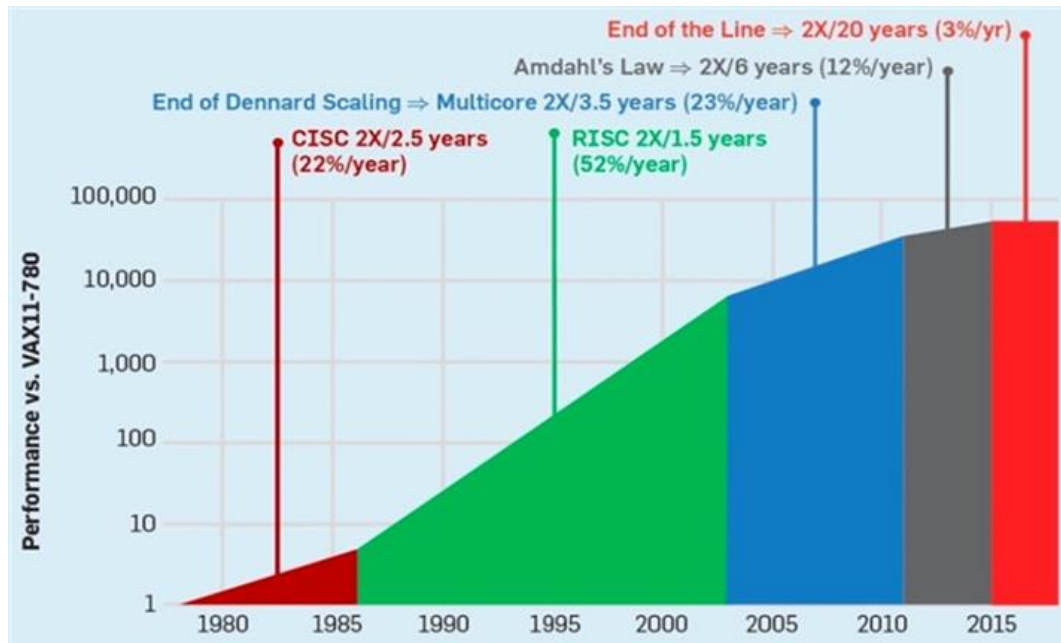
- 1 Enabling innovation with Custom Compute
- 2 Questa Processor: Importance of high-quality CPU formal verification
- 3 Ensure no-harm customization with sequential equivalence checking
- 4 Conclusion
- 5 Summary

Enabling innovation with Custom Compute

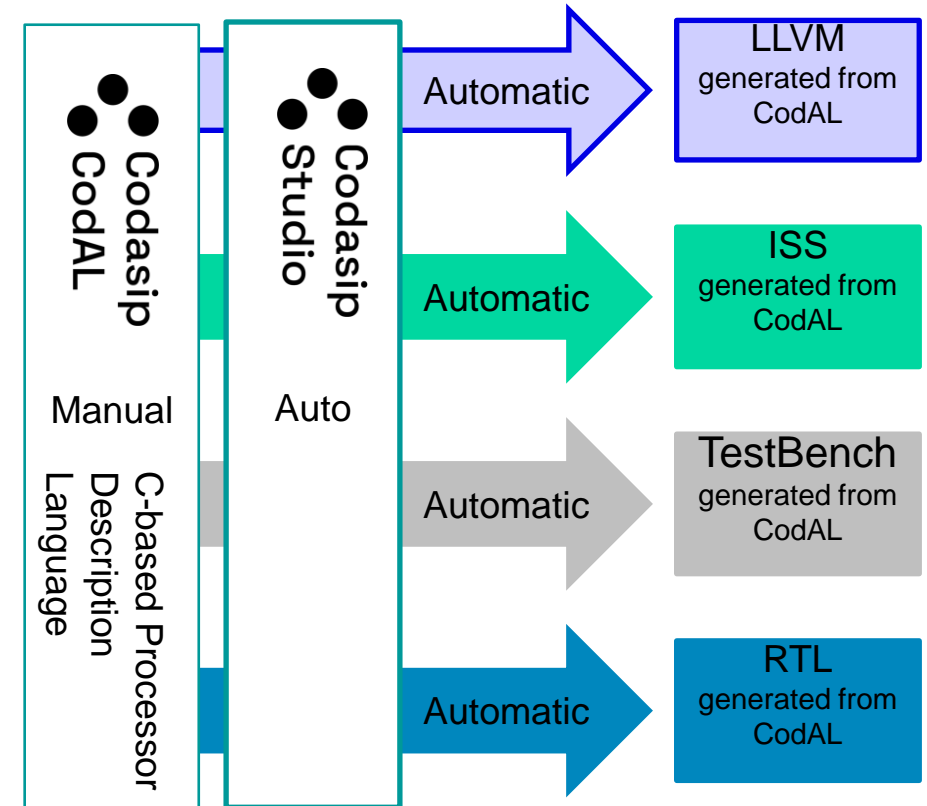
# What is customization?

## Automated Cudasip approach

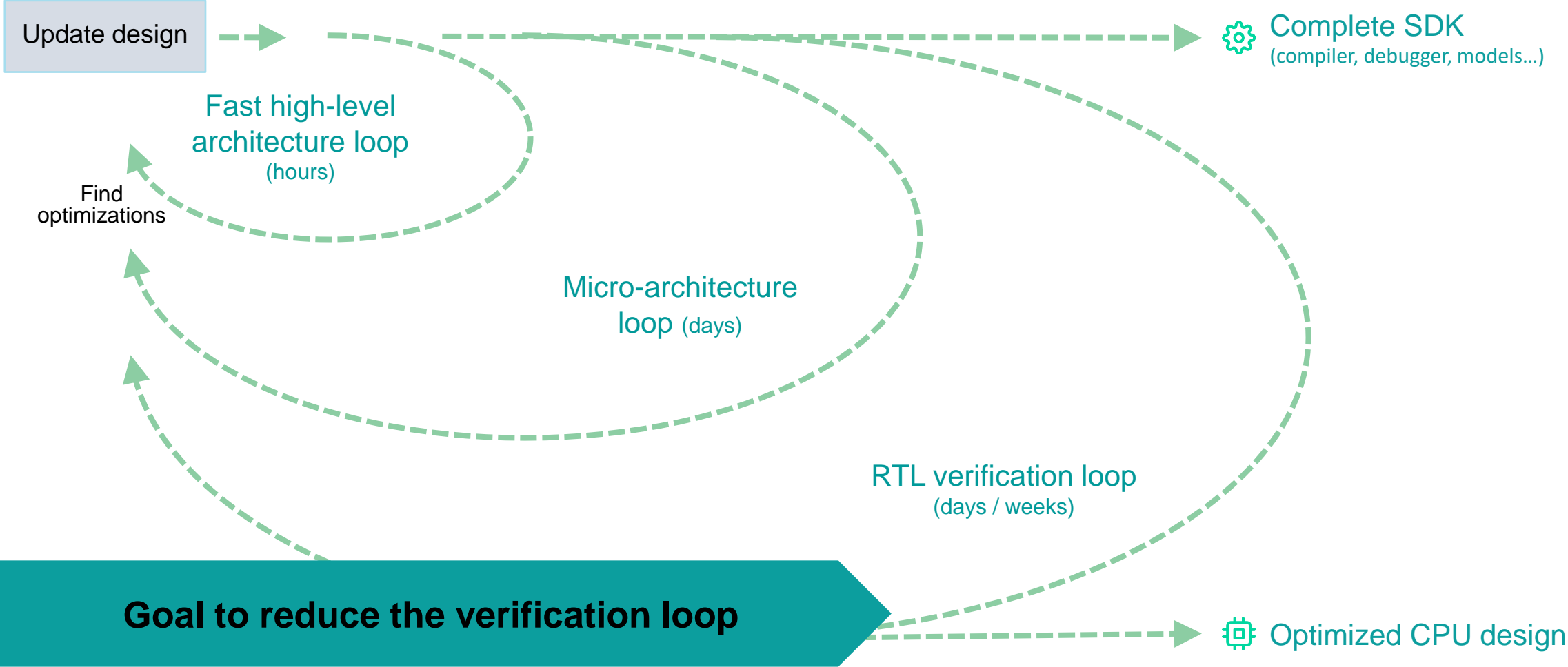
- Custom compute is one answer to the flattening of the "semiconductor laws".
- Cudasip enable custom compute by helping to build dedicated processors and getting the best possible optimization



Source: John Hennessey & David Paterson



# Get optimal results fast with an automated and iterative design process



# Full verification

The base design is fully verified using simulation and formal verification techniques

1

- FV testbenches may be complex when targeting a full ISA implementation:
  - Instruction semantics
  - Architectural register accesses
  - Event handling (exceptions, interrupts, debug, etc.)

2

- Need a formal model of the ISA
  - Siemens Questa Processor Solution provides such a model, plus a guided automated approach providing template of assertions to be used in formal verification

3

- Adding custom instructions and registers requires a full verification
  - Using the same approaches, tools, targets
  - But after adapting the tools to handle the new instructions and registers

## Verification of L31 embedded core RISC-V compliant processor

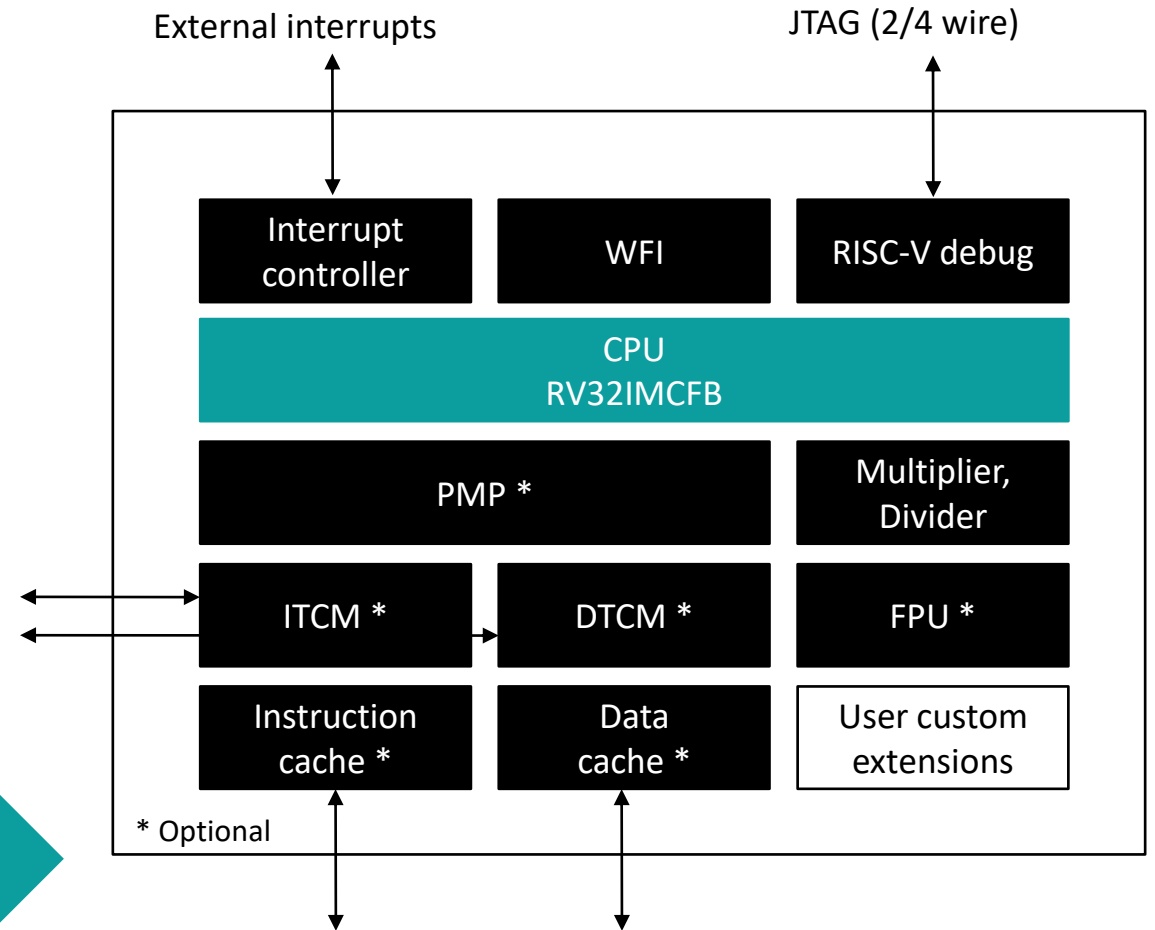
- **Industry-standard interfaces**

- AMBA for instruction and data bus
- JTAG (4pin/2pin) for debugging

- **Configurable and customizable**

- Flexible memory subsystem
- Wide selection of base ISA
- RV32IMFCZicsr\_Zifencei\_Zba\_Zbb\_Zbc\_Zbs

**Goal to have verified, tape-out quality IP**



Questa Processor

Importance of high-quality CPU formal verification



# Questa Processor



## Over 10x improvement

On verification setup and runtime



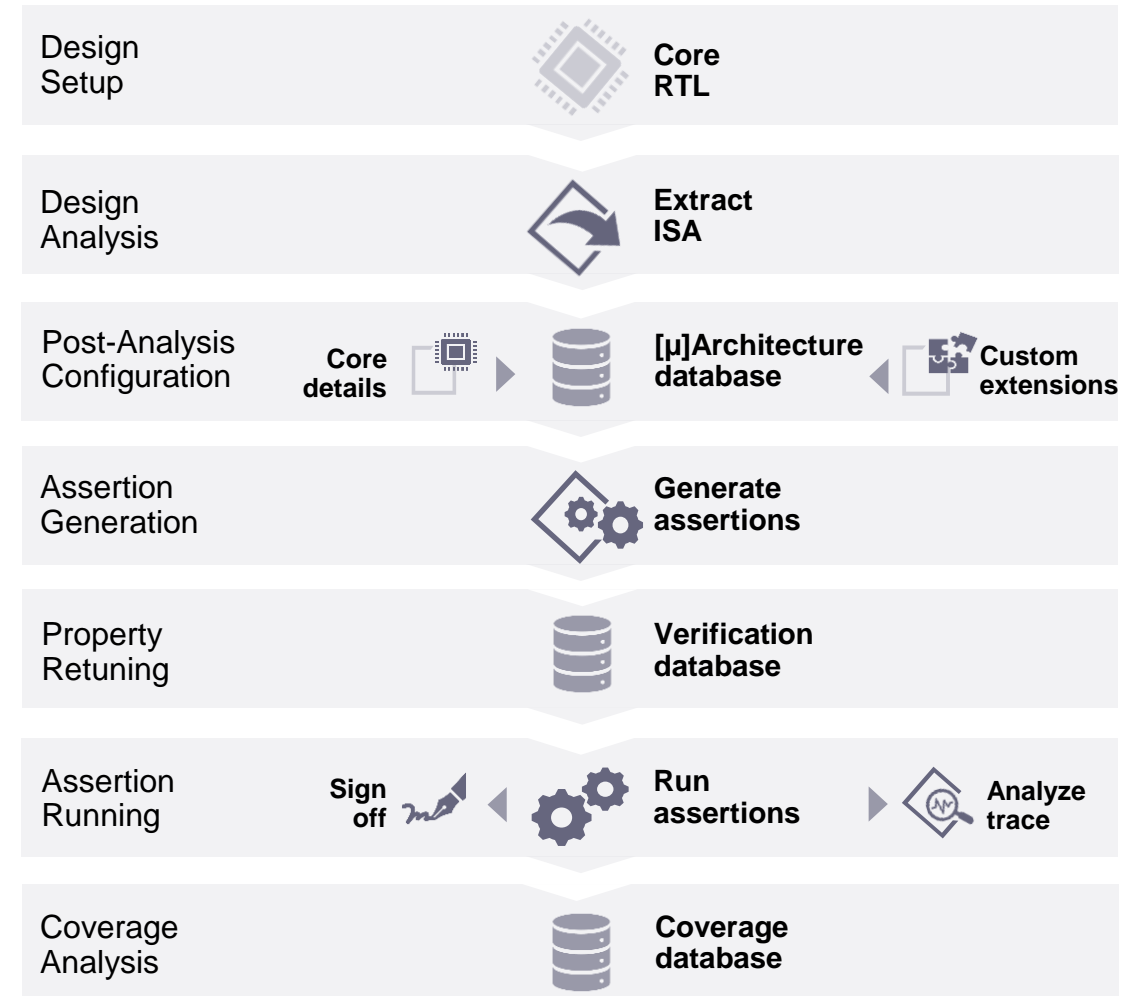
## High degree of automation

Enables automated verification of core RTL



## Exhaustive & complete verification

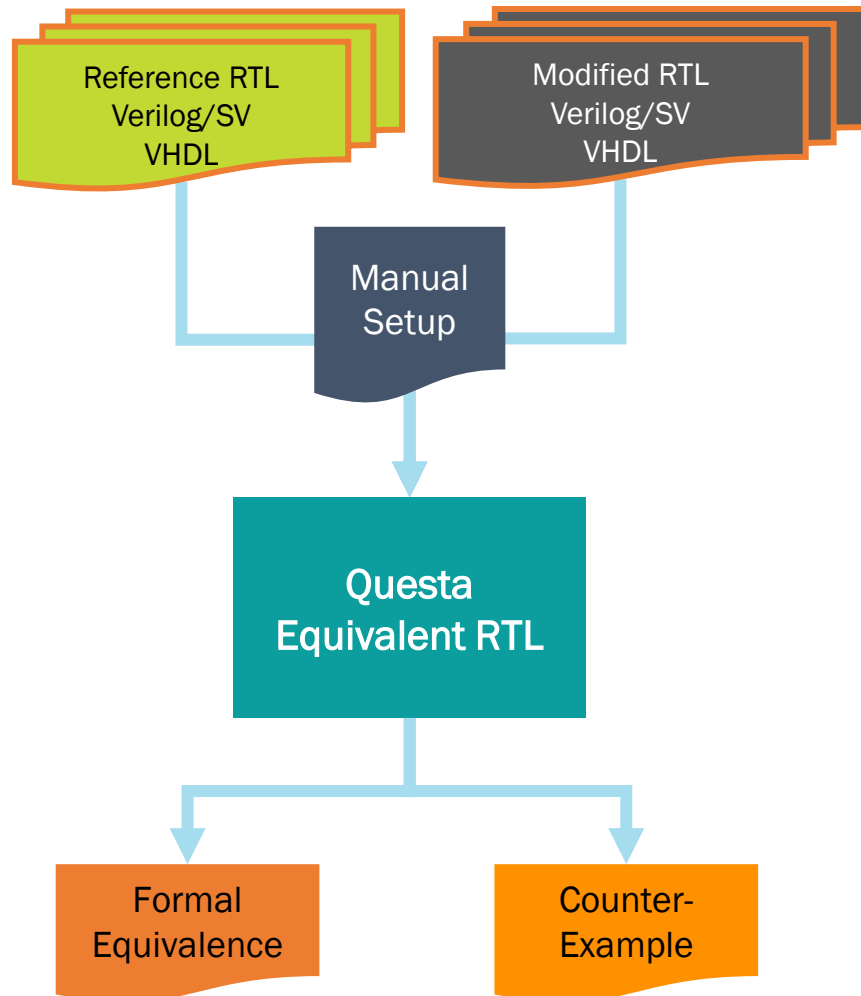
Leaves no bugs and exposes vulnerabilities



Ensure no-harm customization with  
sequential equivalence checking

## Questa Equivalent RTL

Protect known good RTL throughout optimizations and implementation changes

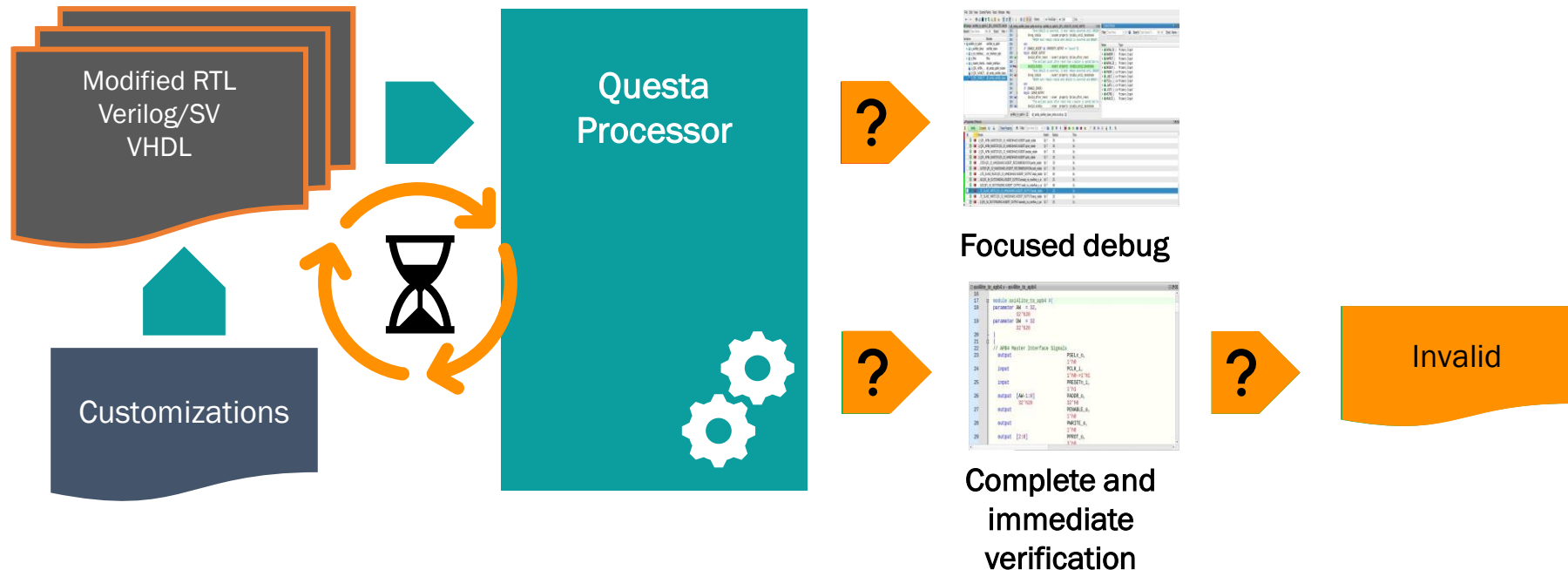


- Equivalence checking of RTL revisions
- HDL RTL language support
  - Verilog/System Verilog
  - VHDL
- Verification applications:
  - Low power clock gating
  - Fault/SEU safety mechanisms
  - Design optimization
  - RTL bug fixes and ECOs

## What is the motivation for “no-harm” verification?

### Some customers may not want to redo a full verification after adding their customizations

- Because it requires software, they don't have access to, or expertise they don't have or still building up
- Because they have already verified the logic they are adding



# Principles behind no-harm verification

## How to do no-harm verification

1

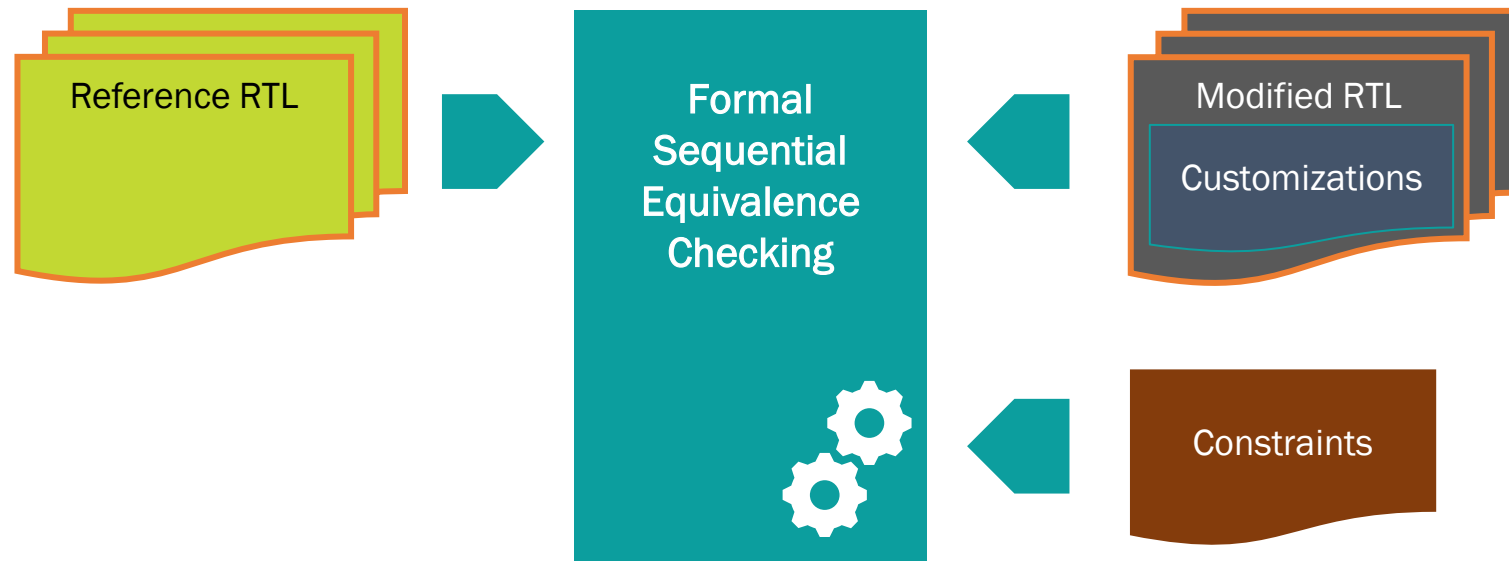
- Formal sequential equivalence checking (SLEC) between the original design, and the customized one... when not exercising the custom features

2

- Needs Sequential Equivalence Checking (not only Logic Equivalence Checking) because of changes into registers

3

- Initial setup is very simple, and sufficient for small designs / minor customizations

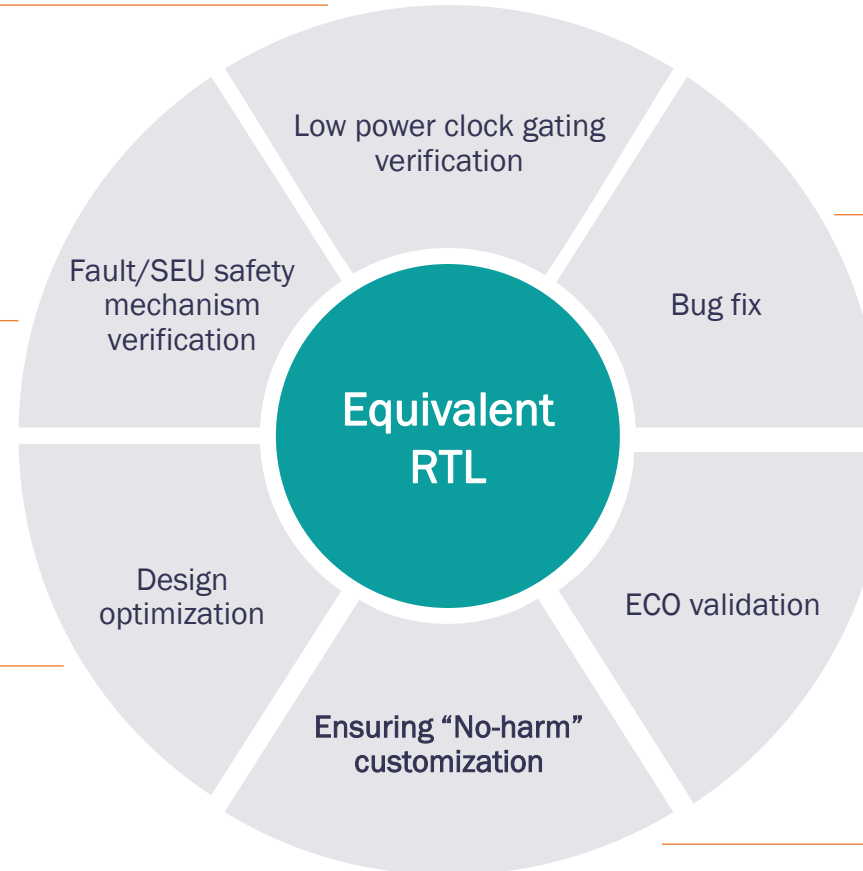


## Popular Formal Sequential Equivalence Checking use cases

With all the clock gating off,  
do they behave the same?

Does the safety mechanism  
detect and handle the  
fault/SEU?

Did your improvements break  
the design function?



Did the fix work and not  
accidentally break anything?

Did optimization  
modifications change design  
behavior?

Are implementations  
equivalent when not  
exercising the custom  
features?

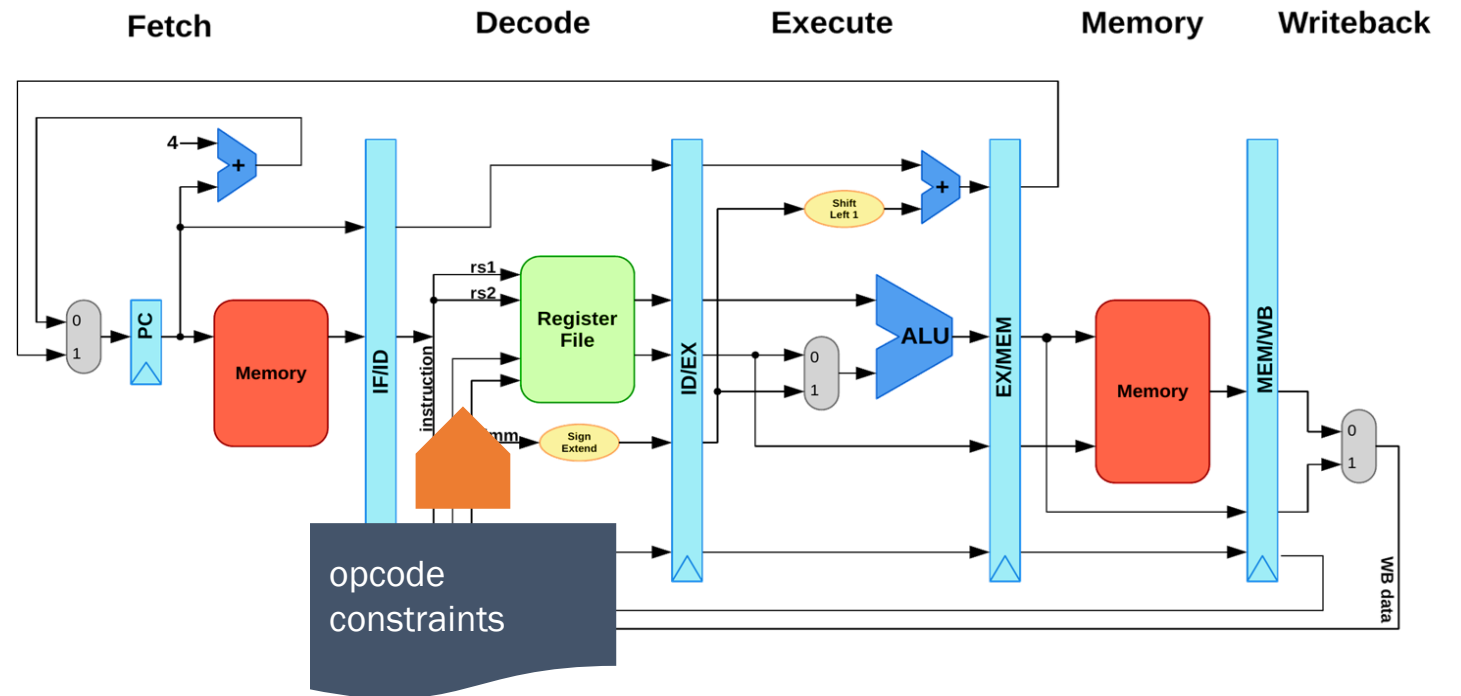
## Harmless customization on L31 (RV32IMCFB) embedded core

### We used the technique to verify the customization of the L31 embedded core

The customizations we want to verify are new instructions:

- MAC and MACS: impact decoder, existing MUL block and FSM, adds a 3rd read port on register file
- SQRT: impacts decoder, add a new functional unit (a new module)

The constraints to forbid the execution of these instructions are on the decoder inputs rather than on the AHB input

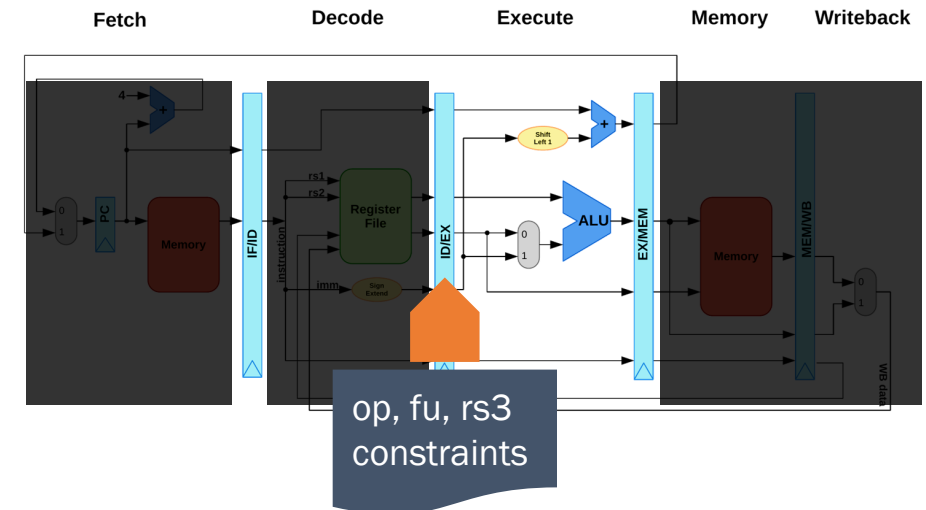
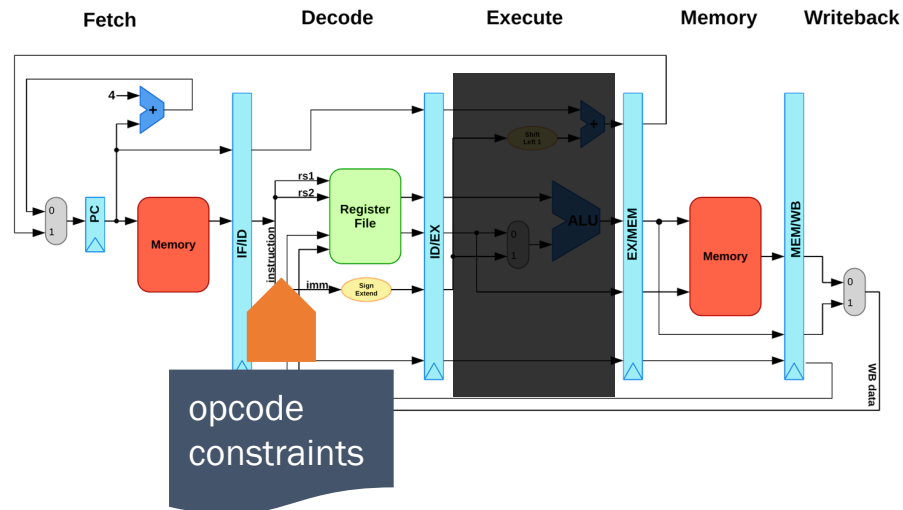


# No-harm on Cudasip L31 embedded core

## Quick solving scalability issues

**Need to split, at least into 2 large blocks execute stage, and the rest using selective black-boxing**

- The initial constraints on the decode inputs are applied to verify the full CPU where the execute stage is black-boxed
- New constraints are devised on the inputs of execute stage when everything except this block is black-boxed.
- Need to formally verify that the original constraints imply the new ones





## How to enable verifiable customization

**Deficiencies to get proof convergence when the customization changes the semantics of critical signals inside the design, although it has no impact on visible outputs**

1

Without customization

```
enum fu_op : uint8
{
    OP_ADD,
    ...
    OP_SW,
    OP_SH,
    ...
    OP_WFI
};
```

2

Bad customization

```
enum fu_op : uint8
{
    OP_ADD,
    ...
    OP_SW,
    OP_SQRT, // custom
    OP_SH,
    ...
    OP_WFI
};
```

3

Good customization

```
enum fu_op : uint8
{
    OP_ADD,
    ...
    OP_SW,
    OP_SH,
    ...
    OP_WFI,
    OP_SQRT // custom
};
```

- Instruction decode produces a set of control signals, including "operation" and "functional unit" of type enum.
- Customization adds new operation and unit values
- If added "in the middle of the existing ones", the semantics of these signals change and SLEC is not able to converge
- Solution: add them "after the existing ones"

# Harmless customization on L31 (RV32IMCFB) embedded core

## Results

This customization of L31 could be verified with the no-harm FSEC technique in a couple of days. It consists in the following steps, each fully converging:

Checks	Designs	Constraints for no custom exec	Run time
Setup sanity	Customized vs Customized	Yes	20 s (proof)
No-harm can catch bugs	Original vs Customized	No	6s (fail)
No-harm on execute block	Original vs Customized	Yes, on execute inputs	270 s (proof)
No-harm on other blocks	Original vs Customized	Yes, on decode inputs	43 s (proof)
Constraint propagation	Customized	As assumes on decode, asserts on execute	5s (proof)

The full FV of a CPU similar to L31 using the Questa Processor Solution is estimated to 2-3 months effort.

# Summary

## Custom compute is increasingly important

### Key takeaways

**Verification methodology to allow customization of existing IPs with confidence.**

Later applied the same technique successfully after extra customizations (addition of crypto instructions, post-increment load, etc.)

**One part of the solution is “no-harm” verification using Questa Equivalent RTL**

- Has the advantage of being very simple to setup and use
- Scalability problem can be mitigated by using simple formal techniques
- Simple guidelines are necessary mitigate potential issues with intrusiveness

We are improving the methodology with tool support to "bound" the customizations to verifiable ones

# Thank you!

## Q&A

Keerthi Devraj – [keerthi.devraj@siemens.com](mailto:keerthi.devraj@siemens.com)

Laurent Arditi – [laurent.arditi@codasip.com](mailto:laurent.arditi@codasip.com)

Nicolae Tusinschi – [nicolae.tusinschi@siemens.com](mailto:nicolae.tusinschi@siemens.com)