Development of a System for Easy Utilization of RISC-V Extensions Using Hypervisor Technology

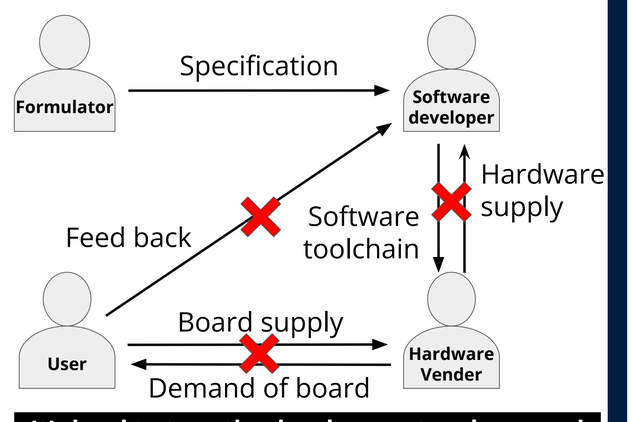
Norimasa TAKANA (takana@syssec.cs.tsukuba.ac.jp), **Yoshihiro OYAMA**
Graduate School of Science and Technology, University of Tsukuba

RISC-V Day Tokyo 2024 Summer

1. Background: Flood of Extensions

In RISC-V, modular specifications known as "extensions" are being formulated one after another.
However, **the hardware implementation has not progressed as expected**, leaving many extensions unused and in a state of limbo.
The main reasons why hardware implementations have not been progressed as desired are
1.Hardware implementation is costly.
2.Implementation of the corresponding software tool chain is essential.
3.It is difficult to predict customer demand in the strong business aspect.
The flood of such extensions is a great loss to the RISC-V community, and a major detriment to RISC-V's greatest identity as open source.

Spec not yet ratified: 73



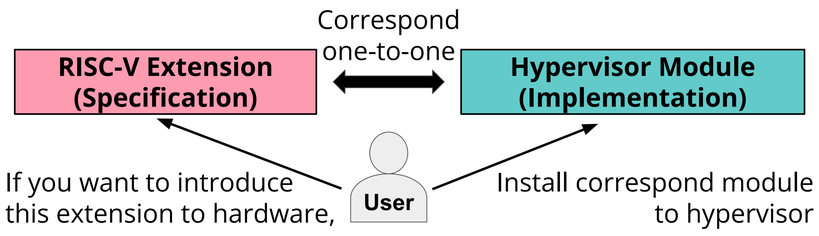
It's hard to turn the development cycle around

Reproduce RISC-V extensions in modules on a hypervisor

Modules that correspond **one-to-one** with the desired RISC-V extensions into the proposed system.
This facilitates management in software and encourages adoption and collaboration with the existing ecosystem.

support the utilization of existing extensions

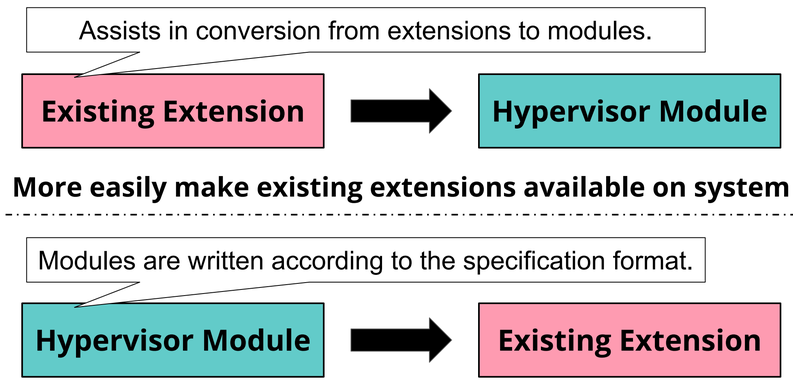
Users can easily introduce new extension environments to their hardware by installing modules.



Acceleration of development cycle

1. The number of users will increase by making it easier to try extensions.
2. **Software Developers** can easily prepare a verification environment and expect feedback from more users.
3. **Hardware vendors** can examine user demand in the environment of a well-developed software toolchain.

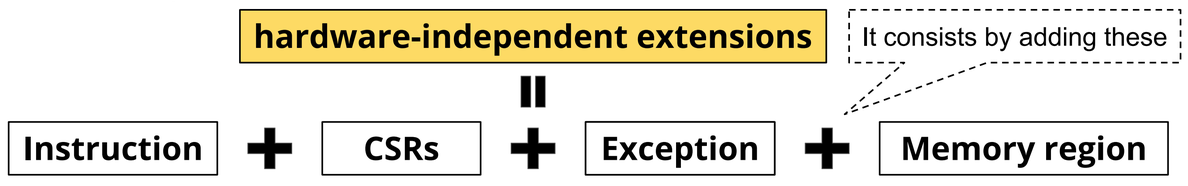
Enabling hypervisor modules tightly coupled to the specification



2. Target Extension: "hardware-independent"

We first target at specialized **hardware-independent** extensions, many involve custom instructions and the addition of CSRs, Such as:

- Zicfiss: For security mechanism called Shadow Stack.
- Smmtt: For setting access permissions to OS-related memory areas.
- Ssdbltrp: For double trap.



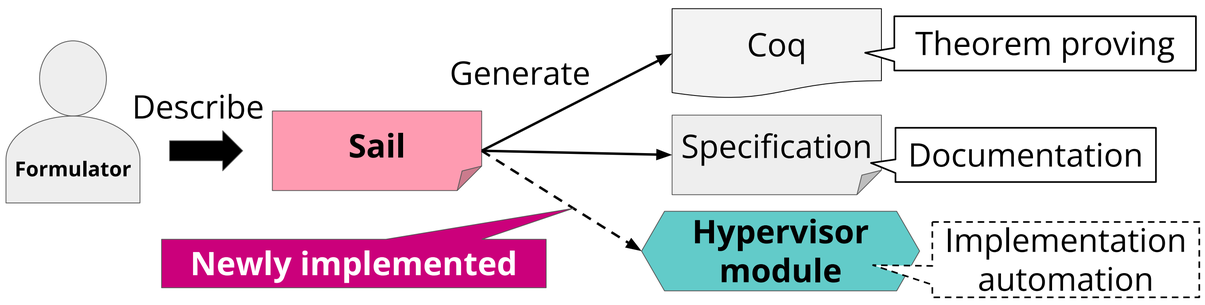
These extensions are hardware-independent and can be fully reproduced by the hypervisor.

3. Implementation: Trap and emulate

Emulation is achieved by trapping the newly introduced instructions, CSRs, exceptions, and memory areas on the hypervisor side.

4. Utilizing Sail: Convert to spec to the module

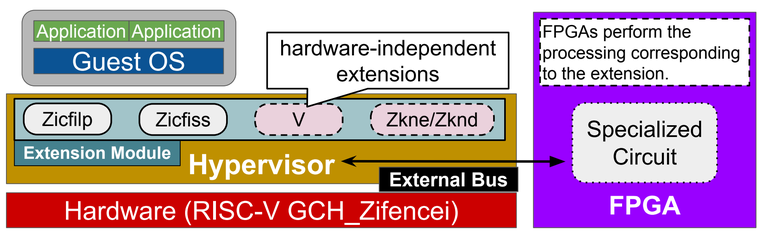
To achieve tight coupling of the extension and hypervisor modules, we utilize Sail [1], a language for defining the instruction-set architecture (ISA) semantics of processors.



[1]: https://github.com/rems-project/sail

5. Future Works: Impl, experiment, etc...

This study is still in the idea and basic implementation stages.
In addition to the ideas presented, we are also considering the following.



Others

- Increasing the number of supported extensions.
- Evaluation on real board.
- Measurement of execution speed overhead.

We plan to present detailed demonstration and experimental results at the next conference.

6. Acknowledgement

This research is partially supported by Exploratory IT Human Resources Project (MITOU Program) of Information-technology Promotion Agency, Japan (IPA) in the fiscal year 2024.