SiFive

RISC-V 工具鏈的進化之路：回顧與展望

The Evolution of the RISC-V Toolchain: A Year in Review and the Road Ahead

Kito Cheng

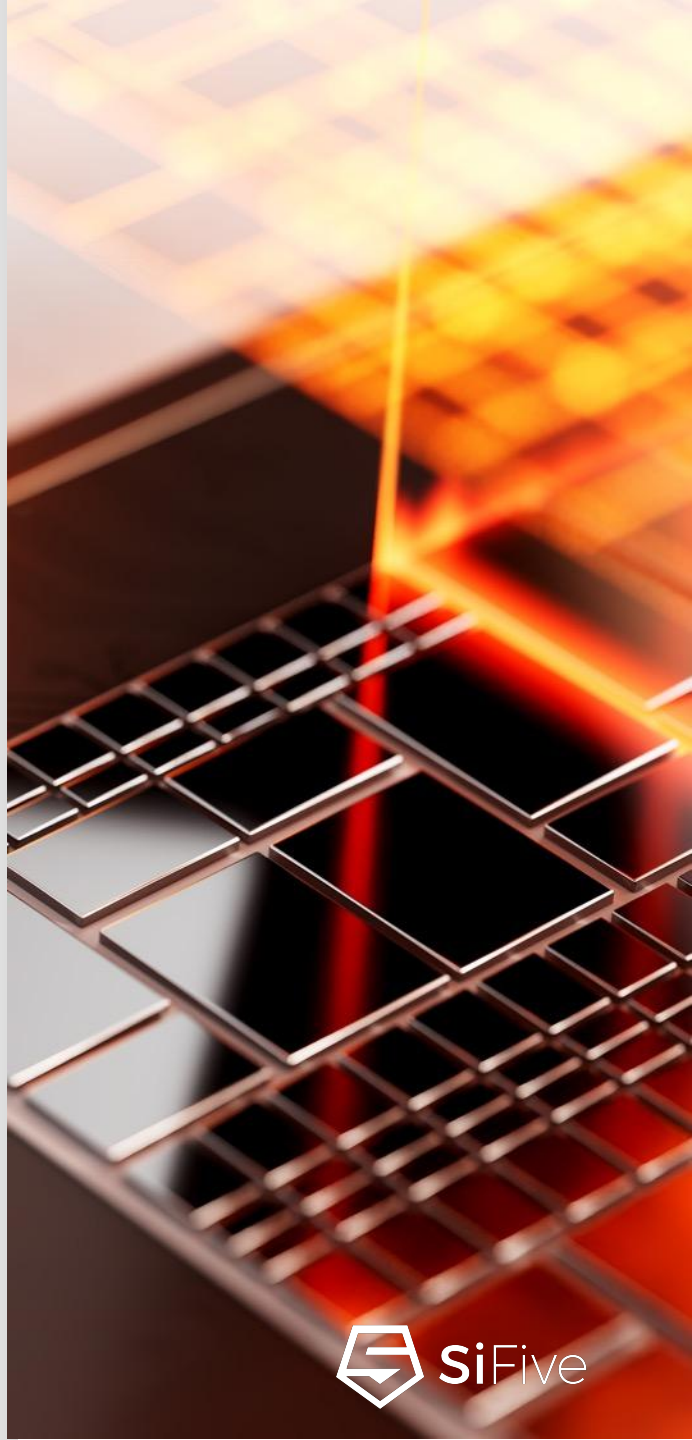
Collaborative Effort Across the Ecosystem

- Maintainers and contributors come from many companies
- The strength of the RISC-V toolchain comes from the community behind it.

PLCT Lab

What's New in Last GCC/LLVM Toolchain Release

- More Stable Vectorization Support
- New Extension Support
- Profile Support for -march
- Function Multiversioning



More Stable Vectorization Support

- GCC:
  - **GCC 14**: First version with auto-vectorization support for RISC-V
  - **GCC 15**: Broader support and many bug fixes

- LLVM:
  - Introduced vector-specific optimizations (e.g., **VLOpt**)
  - Improved vector code generation
  - See also
    - Igalia Blog – LLVM Vector Codegen Improvements
      - https://blogs.igalia.com/compilers/2025/05/28/improvements-to-risc-v-vector-code-generation-in-llvm/
  - Don't miss this afternoon's session:
    "Introduce the implementation of LLVM Loop Vectorizer"

New Extension Support

- Control-Flow Integrity (CFI):
  - Zicfilp, Zicfiss
  - Full CFI support is still in progress

- Atomics
  - New extension: Zabha

- S-mode Extensions
  - Many supervisor-level extensions are now supported (not listed individually)

- All extensions from RVA23 and RVB23 has supported!

Profile Support for -march

- GCC and LLVM now support profile-based -march syntax
  - (GCC must wait until GCC 16)
  - Syntax:
    - -march=<profile-name>[_<extra-extensions>]
      - e.g. -march=rva23u64_zbkb
  - Profiles provide a short form for common sets of extensions
    - Example:
      - -march=rva23u64 instead of
        -march=rv64imafdcbv_zicsr_zicntr_zihpm_ziccif_... (and so on...)

Function Multiversioning

- RISC-V now supports the following function attributes!
  - __attribute__((target("...")))
  - __attribute__((target_clone("...")))
  - __attribute__((target_version("...")))
- Enables function multiversioning — choose the best implementation at runtime
  - Similar to what is available on AArch64 and x86
- Target strings may include extensions like "zve64d", "zbb", etc.
- Spec detail here:
  - https://github.com/riscv-non-isa/riscv-c-api-doc/blob/main/src/c-api.adoc

Function Multiversioning: target

- target attribute for single function with specific extensions

```
__attribute__((target("arch=+v"))) int foo(void) {
  return 0;
}
__attribute__((target("arch=+zbb"))) int bar(void) {
  return 1;
}
```

Function Multiversioning: target_clone

- target_clone attribute for single function with different extensions .

```
__attribute__((target_clones("arch=+v;priority=2",
                             "default",
                             "arch=+zbb;priority=1")))
int foo(int a)
{
  return a + 5;
}

int bar() {
  // foo will be resolved by ifunc
  return foo(1);
}
```

Function Multiversioning: target_version

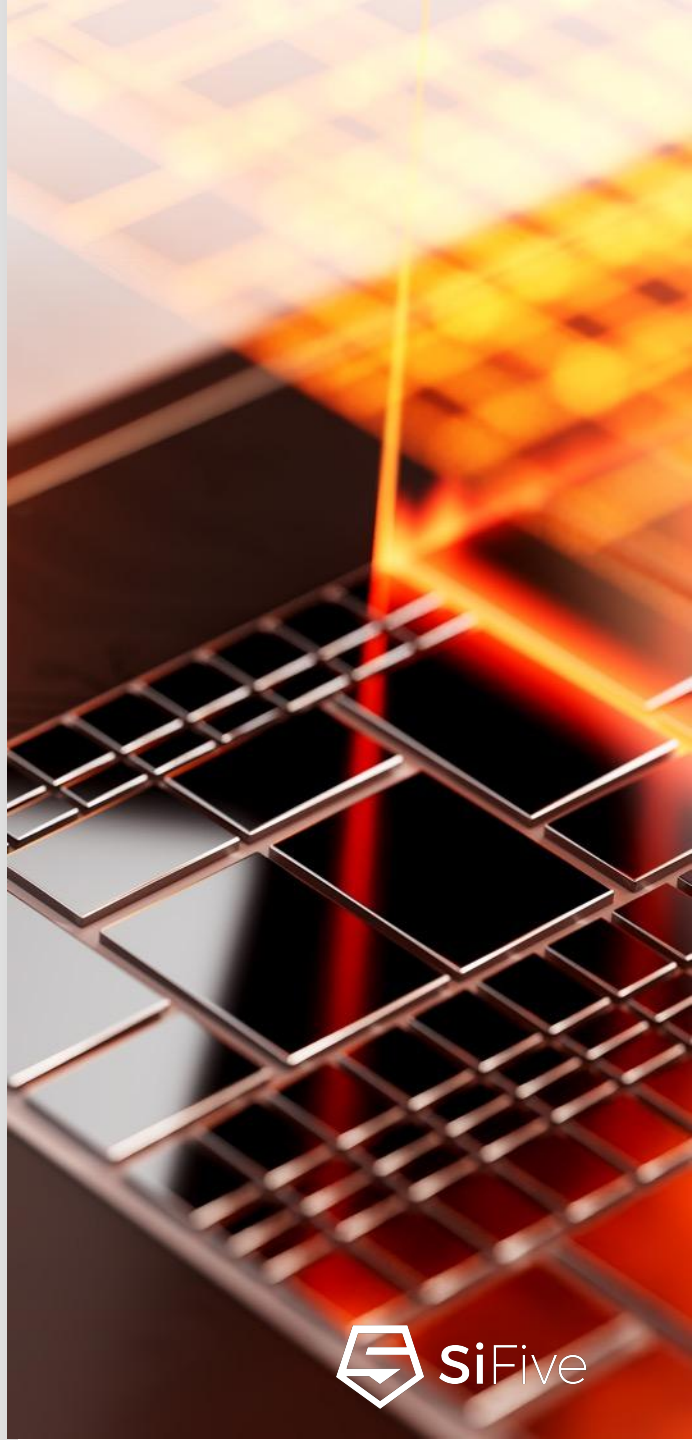
- target_version attribute for single function with different implementation for different extension.

```
__attribute__((
target_version("arch=+v;priority=1")))
int foo(int a)
{
  return a + 5;
}

__attribute__((
target_version("arch=+zbb;priority=2")))
int foo(int a)
{
  return a + 5;
}
```

```
__attribute__((target_version("default")))
int foo(int a)
{
  return a + 5;
}

int bar() {
  // foo will be resolved by ifunc
  return foo(1);
}
```

What's New in Next GCC/LLVM Toolchain Release

- Complete CFI Support
- More New Extension Support
- Vectorization improvement!
- Matrix Extension Support



Complete CFI Support

- Complete shadow stack and landing pad support
  - binutils and glibc!
- And function signature scheme landing pad!

More New Extension Support

- Zilsd: Load/store Pair
- Zibi: Branch with immediate
- And also more vendor extension on upstream
  - Qualcomm
  - Andes
  - T-head

Vectorization improvement!

- Again, don’t miss this afternoon’s session!
  “Introduce the implementation of LLVM Loop Vectorizer”

Matrix Extension Support

- Initial support on LLVM for SiFive matrix extension
  - Work in progress.
  - It's SiFive specific, but the infrastructure could be shared with other future matrix extensions.

RISC-V
Toolchains Need You

- Let’s keep improving the RISC-V ecosystem, together.

> Whether it's contributing code, reporting bugs, testing new features, or sharing performance data — **every bit of help makes a difference.**

Standardizing the RISC-V Toolchain Ecosystem

- https://github.com/riscv-non-isa/riscv-toolchain-conventions
  Defines shared behavior and user-facing conventions for open-source compilers
- https://github.com/riscv-non-isa/riscv-c-api-doc
  Defines intrinsics, predefined macros, and RISC-V-specific C/C++ attributes
- https://github.com/riscv-non-isa/riscv-asm-manual
  Documents and standardizes assembly syntax and conventions
- https://github.com/riscv-non-isa/rvv-intrinsic-doc
  Defines vector intrinsics and language extensions for RVV
- https://github.com/riscv-non-isa/riscv-elf-psabi-doc
  Define the psABI for RISC-V

THANK YOU