Webinar Series - Q&A
Part II: Introduction to SiFive RISC-V Core IP

Q:

Are there any specification or document about caches for Rocket core?

A:

The U54-MC core complex has both instruction and data caches, so it has the most comprehensive documentation for the caches. The E31 and E51 core complexes only have instruction caches, but those share similar controllers to the U54-MC. The relevant specifications are on our product pages:

https://www.sifive.com/products/risc-v-core-ip/u54-mc/
https://www.sifive.com/products/risc-v-core-ip/e51/
https://www.sifive.com/products/risc-v-core-ip/e31/

which link to the manuals, where you can find more specific specifications and the documentation

https://www.sifive.com/documentation/risc-v-core/u54-mc-risc-v-core-ip-manual/
https://www.sifive.com/documentation/risc-v-core/e51-risc-v-core-ip-manual/
https://www.sifive.com/documentation/risc-v-core/e31-risc-v-core-ip-manual/

Q:

How does the cache affect timing predictability in E31? I am thinking of deeply embedded applications

A:

Certainly when executing out of cache, timing may be variable, and we understand that is an important concern for embedded applications. This is why the Core IP includes "Instruction Tightly Integrated Memory (ITIM)", which is a variable-sized instruction scratchpad. Drew will go into more detail, but it allows you to use part of the I-Cache as a scratchpad with predictable timing. Note that this feature does not exist on the FE310-G000 silicon, but it is included in the Core IP deliverables.

Q:

Core Complex is a new name for Coreplex?

A:

Core Complex is the new technical name of the block containing the cores, the PLIC, CLINT, Debug Module, and a few other modules. In terms of the licensable IP product, we are now calling it the SiFive RISC-V Core IP, and it includes the Core Complex in addition to any desired adapters to legacy buses such as AMBA. See the blog post we made on this topic at: https://www.sifive.com/blog/2017/10/11/a-core-by-any-other-name/

Q:

What is relationship of rocket chip repository and core described here?

A:

SiFive is one of the maintainers of the open source repository, https://github.com/freechipsproject/rocket-chip. That repository is a library of components for building a RISC-V based SoC. SiFive leverages this library for its licensed Core IP as well as its open source Freedom platform, https://github.com/sifive/freedom. For licensees of the SiFive RISC-V Core IP, we also provide support, documentation, and verification.

Q:

Does the C and C++ compiler generate privileged instructions, without the user injecting assmebler instruction?

A:

The RISC-V GCC port can be used to compile code that runs in privileged mode, but it only generates user mode instructions to do so. There aren't any privileged mode instructions that it would make sense to use when compiling C code -- the only actual new instruction in the privileged-mode ISA is "sfence.vma" (which synchronizes memory accesses with the page tables), and since there's no corresponding C construct for virtual memory there's now way to emit the instruction in a sensible manner.

The vast majority of the privileged mode lives in CSRs, and as there is no C construct that would map to any of them we don't generate any code for privilege mode CSRs from C.

You can, of course, use inline assembly.

Q:

Are you going to present the U-Series in future webinars?

A:

Most likely so, but they are not on the schedule yet. Many of the things we discussed today also apply to the U-Series.