lwn.net

Is it time for open processors? [LWN.net]

By Jonathan CorbetJanuary 9, 2018

7-9 minutes

The disclosure of the Meltdown and Spectre vulnerabilities has brought a new level of attention to the security bugs that can lurk at the hardware level. Massive amounts of work have gone into improving the (still poor) security of our software, but all of that is in vain if the hardware gives away the game. The CPUs that we run in our systems are highly proprietary and have been shown to contain unpleasant surprises (the Intel management engine, for example). It is thus natural to wonder whether it is time to make a move to open-source hardware, much like we have done with our software. Such a move may well be possible, and it would certainly offer some benefits, but it would be no panacea.

Given the complexity of modern CPUs and the fierceness of the market in which they are sold, it might be surprising to think that they could be developed in an open manner. But there are serious initiatives working in this area; the idea of an open CPU design is not pure fantasy. A quick look around

turns up several efforts; the following list is necessarily incomplete.

What's out there

Consider, for example, the OpenPOWER effort, which is based on the POWER architecture. It is not a truly open-source effort, in that one has to join the club to play, but it is an example of making a processor design available for collaborative development. Products based on the (relatively) open designs are shipping. OpenPOWER is focused on the high end of the computing spectrum; chips based on this design are unlikely to appear in your handset or laptop in the near future.

Then, there is OpenSPARC, wherein Sun Microsystems fully opened the designs of the SPARC T1 and T2 processors. A few projects tried to run with these designs, but it's not clear that anybody got all that far. At this point, the open SPARC designs are a decade old and the future of SPARC in general is in doubt. Interesting things could maybe happen if Oracle were to release the designs of current processors, but holding one's breath for that event is probably not the best of ideas.

OpenRISC is a fully open design for a processor aimed at embedded applications; it has one processor (the OpenRISC 1000) in a complete state. Some commercial versions of the OpenRISC 1000 have been produced, and reference implementations (such as the mor1kx) exist. The Linux kernel

gained support for OpenRISC in the 3.1 release in 2011, and a Debian port showed up in 2014. The Debian work shut down in 2016, though. Activity around the kernel's OpenRISC code has slowed, though it did get SMP support in 2017. All told, OpenRISC appears to have lost much of the momentum it once had.

Much of the momentum these days, instead, appears to be associated with the RISC-V architecture. This project is primarily focused on the instruction-set architecture (ISA), rather than on specific implementations, but free hardware designs do exist. Western Digital recently announced that it will be using RISC-V processors in its storage products, a decision that could lead to the shipment of RISC-V by the billion. There is a development kit available for those who would like to play with this processor and a number of designs for cores are available.

Unlike OpenRISC, RISC-V is intended to be applicable to a wide range of use cases. The simple RISC architecture should be relatively easy to make fast, it is hoped. Meanwhile, for low-end applications, there is a compressed instruction-stream format intended to reduce both memory and energy needs. The ISA is designed with the ability for specific implementations to add extensions, making experimentation easier and facilitating the addition of hardware acceleration techniques.

The Linux support for RISC-V is quite new; indeed, it will only

appear once the 4.15 release gets out the door. The development effort behind it appears to be quite active, and toolchain and library support are also landing in the appropriate projects. RISC-V seems to have quite a bit of commercial support behind it — the RISC-V Foundation has a long list of members. It seems likely that this architecture will continue to progress for some time.

A solution to the hardware problem?

In response to Meltdown and Spectre, the RISC-V Foundation put out a press release promoting the architecture as a more secure alternative. RISC-V is indeed not vulnerable to those problems by virtue of not performing any speculative memory accesses. But the Foundation says that RISC-V has advantages that go beyond a specific vulnerability; the openness of its development model, the Foundation says, enables the quick incorporation of the best security ideas from a wide range of developers.

It has become increasingly clear that, while Linux may have won the battle at the kernel level, there is a whole level of proprietary hardware and software that runs below the kernel that we have no control over. An open architecture like RISC-V is thus quite appealing; perhaps we can eventually claw some of that control back. This seems like a dream worth pursuing, but getting there involves some challenges that must be overcome first.

The first of these, of course, is that while compilers can be had for free, the same is not true of chip fabrication facilities, especially the expensive fabs needed to create high-end processors. If progress slows at the silicon level — as some say is already happening — and fabrication services become more available to small customers, then it may become practical for more of us to experiment with processor designs. It will never be as easy or as cheap as typing "make", though.

Until then, we're going to remain dependent on others to build our processors for us. That isn't necessarily bad; almost all of us depend on others to build most of our software for us as well. But a higher level of trust has to be placed in hardware. Getting reproducible builds working at the software level is a serious and ongoing challenge; it will be even harder at the hardware level. But without some way of verifying underlying design of an actual piece of hardware, we'll never really know if a given chip implements the design that we're told it does.

Nothing about the RISC-V specification mandates that implementation designs must be made public. Even if RISC-V becomes successful in the marketplace, chances are good that the processors we can actually buy will not come with freely licensed designs. Large customers (those that build their own custom data centers) may well be able to insist on getting the designs too — or just create their own — but the rest of us will find ourselves in a rather weaker bargaining position.

Finally, even if we end up with entirely open processors, that will not bring an end to vulnerabilities at that level. We have a free kernel, but the kernel vulnerabilities come just the same. Open hardware may give us more confidence in the long term that we can retain control of our systems, but it is certainly not a magic wand that will wave our problems away.

None of this should prevent us from trying to bring more openness and freedom to the design of our hardware, though. Once upon a time, creating a free operating system seemed like an insurmountably difficult task, but we have done it, multiple times over. Moving away from proprietary hardware designs may be one of our best chances for keeping our freedom; it would be foolish not to try.

(Log in to post comments)