THE MARKET DEMANDS OPEN COMPUTER PROCESSORS

By **Jonathan Corbet**
January 9, 2018

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)

Is it time for open processors?
Posted Jan 9, 2018 14:55 UTC (Tue) by **armijn** (subscriber, #3653) [Link]

There is a (somewhat) successful open SPARC implementation (though I am not sure if it is actually based on OpenSPARC) and that is the LEON, which is used by ESA. That is, of course, a bit of a niche market.

Is it time for open processors?
Posted Jan 10, 2018 0:21 UTC (Wed) by **JanC_** (subscriber, #34940) [Link]

LEON is not based on on the UltraSPARC T1/T2 designs that were releases as "OpenSPARC". It's a much simpler 32-bit CPU design (SPARC v8).

They were manufactured by Atmel, and are currently being sold by Microchip, e.g.:
http://www.microchip.com/wwwproducts/en/AT697F
http://www.microchip.com/wwwproducts/en/ATF697FF
http://www.microchip.com/wwwproducts/en/AT7913E

Is it time for open processors?
Posted Jan 9, 2018 15:07 UTC (Tue) by **jcm** (subscriber, #18262) [Link]

The problem with an "open" processor is the cost. It costs over $1.2 Billion to do a ground up 4 year OoO core of the kind of competitive performance that others have built. And even then, once silicon is deployed, security issues can still be found. In short, it will *never* happen. You'll get RISC-V cores like BOOM, you'll see some wonderful IoT designs, but you'll *never* see a high end Xeon-class core unless some billionaire funds it as a pet project, and keeps investing year after year for the greater good. And no, none of the major search/cloud vendors are going to go fund this - they want to make commercial vendors compete.

> More useful would be focusing on how to make existing designs robust, especially against mitigations. I'm a huge fan of microcode, but its implementation today is very limited. There's a lot more research that can happen into how to address security in the field.

Is it time for open processors?
Posted Jan 9, 2018 15:15 UTC (Tue) by **ejr** (subscriber, #51652) [Link]

Don't bet against it quite yet. Plus, existing RISC-V research work done by grad students has produced accelerators for various problems that are competitive in speed and/or power/operation.

Is it time for open processors?
Posted Jan 9, 2018 15:49 UTC (Tue) by **epa** (subscriber, #39769) [Link]

Does your figure of $1.2 billion include manufacturing or is it just for the design stage?

Is it time for open processors?
Posted Jan 9, 2018 15:56 UTC (Tue) by **pizza** (subscriber, #46) [Link]

I think 1.2 billion is a bit high for just R&D, but I can easily see "hundreds of millions" as the price tag to design (and more importantly, adequately verify!) a modern CPU core. And that's assuming it's re-using an existing instruction set and hardware platform so you don't have to bootstrap the rest of the ecosystem around it.

Is it time for open processors?
Posted Jan 9, 2018 23:16 UTC (Tue) by **smoogen** (subscriber, #97) [Link]

I think the 1.2 billion is the case for getting a system into complete production while facing the 'realities' of the industry. The design and 'code' of the chipset to run it in an emulator may only be in the 10->100 million range.. the getting that to be something on silicon starts adding up quickly. You start running into physical problems (well yes we could do that set of adds there but the chip burns up and if we space them out we slow down this other operation). Then you have to avoid physical patents (aka use only methods from 20 years ago in a lot of cases) or spend a lot on them or hopefully have a ton of patents of your own that you can force cross-license costs which will make you pay less overall.

Is it time for open processors?
Posted Jan 9, 2018 17:25 UTC (Tue) by **excors** (subscriber, #95769) [Link]

I think manufacturing a single chip takes something on the order of a million dollars and a couple of months from when you send the completed design to the foundry. (Subsequent chips are much cheaper and quicker). If you've been employing hundreds of engineers to design it, and buying FPGAs and simulators to test it, the manufacturing sounds like a fairly trivial cost (though the latency can be annoying, particularly if you find a bug and need to start again).

Is it time for open processors?
Posted Jan 9, 2018 18:35 UTC (Tue) by **daney** (subscriber, #24551) [Link]

Think 40 mask layers * (between $80,000 and $100000 per mask) for modern processes and it adds up to more than "a million". The latency from design in to packaged chip out is indeed on the order of two months.

"Metal" fixes where you just rewire the top few layers of wires to fix minor bugs are much cheaper as you only have to regenerate a few (say six) of the masks.

The cost of employing 50-100 people for a couple of years is where the main costs to chip development lie. It is usually more than just a "CPU" design, as you need things like DRAM controllers, coherent cache fabrics, PCIe ports, packaging, thermal engineering, etc. for a usable design.

Is it time for open processors?
Posted Jan 16, 2018 9:26 UTC (Tue) by **marcH** (subscriber, #57642) [Link]

> It is usually more than just a "CPU" design, as you need things like DRAM controllers, coherent cache fabrics, PCIe ports, packaging, thermal engineering, etc. for a usable design.

For low-power, single core, embedded and highly integrated SoCs the real estate used by the CPU core can actually be small compared to all the rest. Now the relationship between number of gates and design complexity is of course not direct, however it's not completely unrelated either.

So yes: open-source SoCs would make a difference, open-sourcing CPU cores alone not so much. As long as this confusion is maintained there's nothing wrong to correct :-)

Is it time for open processors?
Posted Jan 9, 2018 18:37 UTC (Tue) by **mtaht** (✭ **supporter** ✭, #11087) [Link]

This was one of the most open source design and development efforts done to date: http://www.adapteva.com/andreas-blog/adapteva-status/

But although it was done more cheaply than anyone could have believed, by a small team and led by someone utterly brilliant, (he wrote a good paper describing deeply how the costs broke down

http://www.adapteva.com/wp-content/uploads/2013/06/hpec12...

) several important features proved too buggy to work in practice on the first chip revisions, and it looks unlikely those bugs will ever be resolved, nor will they do the 1000 core version.

Is it time for open processors?
Posted Jan 9, 2018 18:46 UTC (Tue) by **mtaht** (✭ **supporter** ✭, #11087) [Link]

oops, wrong second link. I cannot find the paper, did find some relevant slides.

https://www.parallella.org/wp-content/uploads/2017/01/hip...

Is it time for open processors?
Posted Jan 9, 2018 17:05 UTC (Tue) by **excors** (subscriber, #95769) [Link]

As software people, maybe we need to stop relying on ever-more-complex OoO hardware to make our code faster over time, and design software that can run optimally on simpler in-order CPUs instead. Then open-hardware people could make reasonably competitive CPU designs without the absurd complexity that leads to these surprising vulnerabilities and that requires the billion-dollar R&D costs.

In particular, you can probably fit two reasonable in-order cores in the same silicon area and power budget as one big out-of-order core, and get better performance from software that uses all the cores. But writing multithreaded software is really hard; occasionally that's inherent in the problem we're trying to solve, but I think in many cases it's just because our languages and tools and design patterns make it excessively difficult and error-prone. I imagine there are lots of expensive serial algorithms in the Linux kernel that could benefit from concurrency, but concurrency in C usually involves far more pain than it's worth, so that rarely happens. I don't know what languages etc would be better - I don't know if they even exist yet - but surely *something* must be possible?

GPUs show that approach works in some cases. Each 'core' can be a relatively simple in-order thing, with dozen-cycle instruction latency, no branch prediction, no per-core data cache, and can happily stall a thread for hundreds of cycles while waiting for memory. In exchange for those limitations the programmer is given thousands of cores, a programming model that makes it easy to use all those cores, and much better power efficiency and peak performance than a CPU.

I don't think you'd want to run an OS kernel on a GPU - they're a bit too extreme in prioritising throughput over latency. But maybe something halfway between GPUs and OoO CPUs (in terms of core complexity, latency, core count, etc), with a suitable programming model to make best use of it, could work much better than what we've got today.

Is it time for open processors?
Posted Jan 9, 2018 20:04 UTC (Tue) by **andresfreund** (subscriber, #69562) [Link]

I think too many use cases care about latency to a large enough that going to simple in-order cores is going to work. It's not particularly realistic to effectively parallelize tasks that only take a few ms to a large number of cores. There's certainly a lot of improvements needed to take advantage of more cores, but I think the number of cases where single-core performance is crucial will be large enough that we won't see a large move to simple in-order cores.

It's possible that we'll go more in the direction of a handful of complex OOO cores, and a lot more simple cores for the rest of the work, but that won't help against spectre like vulns.

Is it time for open processors?
Posted Jan 9, 2018 21:03 UTC (Tue) by **roc** (subscriber, #30627) [Link]

The problem is Amdahl's law. There's always some part of your workload that can't be parallelized, and given enough CPUs, that part of your workload will come to dominate performance. Thus maximizing single-threaded performance, including using speculative execution, is always important.

Is it time for open processors?
Posted Jan 9, 2018 23:15 UTC (Tue) by **daniels** (subscriber, #16193) [Link]

> The problem is Amdahl's law. There's always some part of your workload that can't be parallelized, and given enough CPUs, that part of your workload will come to dominate performance. Thus maximizing single-threaded performance, including using speculative execution, is always important.

Hopefully autotools won't be with us for much longer.

Is it time for open processors?
Posted Jan 16, 2018 19:15 UTC (Tue) by **hkario** (subscriber, #94864) [Link]

unfortunately the current applications running in the BrowserOS only recently aren't forced on a single CPU and each get a thread by themselves

not that it helped much for performance

Is it time for open processors?
Posted Jan 9, 2018 21:13 UTC (Tue) by **tshow** (subscriber, #6411) [Link]

> I imagine there are lots of expensive serial algorithms in the Linux kernel that could benefit from concurrency, but concurrency in C usually involves far more pain than it's worth, so that rarely happens. I don't know what languages etc would be better - I don't know if they even exist yet - but surely *something* must be possible?

That's been the mantra for the last 30 years at least, but rather like practical fusion, it always seems to be a decade away.

Aside from anything else, if parallelism was being held back by C-family languages, you'd think someone would have written a library with a simple API to wrap the problem up and isolate it from the calling language.

One of the advantages of pure functional languages is that they make parallelism easier (the lack of side effects makes isolating functions simpler), but in practical terms the difference hasn't been enough that the world has flipped to LISP or ML-family languages.

Besides, a lot of programming problems are stubbornly serial.

I think if anything the best hope is probably a combination of simplified faster cores and a re-examination of the primitives modern programming languages need to do their jobs. Who knows what performance we could wring from hardware that was built with modern compilers in mind?

Is it time for open processors?
Posted Jan 10, 2018 8:23 UTC (Wed) by **smurf** (subscriber, #17840) [Link]

> I think if anything the best hope is probably a combination of simplified faster cores
> and a re-examination of the primitives modern programming languages need to do their jobs.

Modern processor cores already are as fast as it gets … and what could be more simple, conceptually, than "load a value, do something else while that load stalls, then do something with the value"?

The reason some current processors only have two hyperthreads is probably because more don't increase speed, due to switching costs, large caches, and instruction-level parallel execution. All of this also speeds up single-threaded programs, which is why a simpler, possibly-more-hyperthreaded processor is unlikely to win any real-world performance awards.

Is it time for open processors?
Posted Jan 10, 2018 9:45 UTC (Wed) by **renox** (subscriber, #23785) [Link]

> Modern processor cores already are as fast as it gets … and what could be more simple, conceptually, than "load a value, do something else while that load stalls, then do something with the value"?

Not having this? Have a look at the Mill CPU (https://en.wikipedia.org/wiki/Mill_architecture), it tries to get the same performance than modern CPU on regular code without having the complexity of OoO CPU.

It's only slideware currently though unfortunately and its single address space design would need big changes in current softwares so I'm not very optimistic..

Is it time for open processors?
Posted Jan 10, 2018 13:53 UTC (Wed) by **smurf** (subscriber, #17840) [Link]

> It's only slideware currently though unfortunately and its single address space design
> would need big changes in current softwares so I'm not very optimistic..

That's one problem.

The other: Compare to what Transmeta tried to do and what they ended up actually accomplishing before folding. The Mill idea is an order of magnitude more ambitious, which IMHO translates to an equally extensive uncertainty WRT the achievable results.

Is it time for open processors?
Posted Jan 20, 2018 6:49 UTC (Sat) by **igodard** (guest, #105242) [Link]

Mill team here. A bug-compatible x86 (Transmeta) was vastly more ambitious than anything we ever considered. Yes, the Mill is different, but the bulk of the difference is simplification. "Please excuse this long letter, but I did not have the time to write a short one" - Blaise Pascal

Is it time for open processors?
Posted Jan 10, 2018 14:55 UTC (Wed) by **jcm** (subscriber, #18262) [Link]

Good luck to them. They have absolutely no chance whatsoever.

Is it time for open processors?
Posted Jan 10, 2018 15:14 UTC (Wed) by **mtaht** (✭ **supporter** ✭, #11087) [Link]

One really remarkable thing I've learned in life: *Sometimes* Don Quixote does win.

Is it time for open processors?
Posted Jan 11, 2018 20:47 UTC (Thu) by **joib** (subscriber, #8541) [Link]

To begin with, I think it's important that we as a society fund wild out-there stuff, even if most of it "fails". Because the alternative is incrementalism, which we have enough of already, thank you very much. So in the grand scheme of things, I think it's perfectly Ok if we spend some 10's of millions of $$$ on Mill to see if the idea flies.

That being said, the Mill "belt" seems like a somewhat clever mix between "normal" register based cpu's and a stack machine. OTOH I'm not convinced it avoids the problems in making a superscalar pipelined stack machine, in that you get a very strict ordering requirement due to the results being pushed on top of the stack (or onto the end of the belt in Mill). Further, their IPC claims seem, er, really far out there; I don't understand how they can credibly claim such numbers. Add in the fact that after all these years they still have nothing more than powerpoints to show, so I'm a bit skeptical.

But if it works out, hey, awesome!

Is it time for open processors?
Posted Jan 16, 2018 9:39 UTC (Tue) by **marcH** (subscriber, #57642) [Link]

> Aside from anything else, if parallelism was being held back by C-family languages, you'd think someone would have written a library with a simple API to wrap the problem up and isolate it from the calling language.

The memory model has to be built in the language. C/C++ finally got one after 40 years.
http://www.hboehm.info/c++mm/

https://doc.rust-lang.org/book/first-edition/concurrency....

Is it time for open processors?
Posted Jan 9, 2018 21:14 UTC (Tue) by **jcm** (subscriber, #18262) [Link]

Nice ideas, but the reality is that the industry has convinced itself that only software matters. Nobody cares about hardware, and nobody understands it at anything like the scale of the number of software engineers who can write python to run on brawny cores. Certain players in the industry have spent years literally sucking the competency out of others in an effort to benefit from the extreme logical extent of abstraction. This will never be fixed. The only path is open and fair competition from alternatives provided by companies who can make a decent living enough to invest in taking on incumbents.

Is it time for open processors?
Posted Jan 9, 2018 22:58 UTC (Tue) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

With industry consolidation (Intel buying Altera, Broadcom buying Qualcomm, Synoptics buying up a zillion toolmaker), I am pessimistic about the future of complex hardware design.

Is it time for open processors?
Posted Jan 9, 2018 23:15 UTC (Tue) by **jcm** (subscriber, #18262) [Link]

Me too. I have been for years.

Is it time for open processors?
Posted Jan 10, 2018 0:34 UTC (Wed) by **tpo** (subscriber, #25713) [Link]

Does anybody in this thread have a figure on how much out of order execution buys you? Is that in the order 50% or of a factor of 2 or 10* or ... ?

50% or a factor two is not huge and I would guess that this could be gained by less fat and wasteful software stacks. I think modern programs do *huge* memory I/O due to fat stacks and OO, so I think most programs are RAM I/O bound (i.e. word processing does basically what it did 30 years ago, except for the much prettier glyphs, but is using 10^2 - 10^3 times more RAM for the "same task").

Wrt to parallelisation I think the only "workable" paradigm is message passing (what I understand of Rust's parellelizing paradigm I would consider as message passing).

But, as far as I understand, message passing is even more demanding on I/O since the OS needs to be copying a lot of data, so that'd be even worse. The only solution to that problem I can see is something like HPs "The machine" concept where you have a lot of cores with a lot of local RAM and extremely fast message passing between the cores. Which as an abstraction maps somewhere between "each core has it's own OS" and "each core is a process".

All this is my reasoning as a non-practitioning bystander looking at the various working and imagined concepts.

Are there papers looking at these concepts quantitatively and holisticaly? I.e. price per "processing unit", total computing throughputs, I/O throughputs, estimates of the degrees of freedom in applications (in how many parallel parts could typical workloads be possibly split?)?

> That is, do we know for sure that fat sequential CPUs will always win with today's workloads and the paradigms that we are aware of?

Is it time for open processors?
Posted Jan 10, 2018 1:59 UTC (Wed) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

The OO gain is about 5x or more on many loads. 100x is not unheard of for carefully tuned code.

Is it time for open processors?
Posted Jan 10, 2018 14:58 UTC (Wed) by **jcm** (subscriber, #18262) [Link]

Yea, OoO benefit is *huge*. And there's nothing wrong with OoO. The problem comes when you take the implementation a little far and separate your permissions checking from your other logic (and handle the exception at retirement) in the name of speed.

OoO has limits. Intel's general approach for a while has been to shove progressively large reorder windows into their cores. You get to do that for a few generations before it stops buying you increased performance. I've been gleefully looking forward to how they handle trying to exceed 224 entries in flight, because it won't let them publish PowerPoint slides showing much benefit. They'll probably still go and build it tho.

Is it time for open processors?
Posted Jan 10, 2018 15:45 UTC (Wed) by **excors** (subscriber, #95769) [Link]

> there's nothing wrong with OoO

Can you realistically have an OoO CPU without speculative execution? (I'd assume not since OoO execution benefits from having lots of instructions in flight, and most programs don't exclusively use long sequences of instructions with no branches, so you'd lose too much performance if you didn't speculate across branches.)

Spectre seems to indicate there *is* something fundamentally wrong with speculative execution (hence with OoO). CPUs will execute sequences of instructions that do not match the program they're meant to be running. Execution is observable through a wide variety of side channels. It doesn't matter how rigorously we prove the security of our software, and how it avoids revealing any secrets over those side channels, if the CPU is going to execute something arbitrarily different. It therefore becomes impossible to write secure software.

Is it time for open processors?
Posted Jan 10, 2018 17:37 UTC (Wed) by **Jonno** (subscriber, #49613) [Link]

> Spectre seems to indicate there *is* something fundamentally wrong with speculative execution

No, Spectre only indicates that there is something fundamentaly wrong with *some* speculative execution.

In the RISC-V community, the current consensus seems to be that only the following 4 categories of speculation are potentially problematic:

- Using a speculated register value as a memory address
- Using a speculated register value as a branch condition
- Using a speculated register value as a jump/branch target
- Using a speculated register value in a variable-time instruction.

If you avoid these, all other speculation should be fine, notably including prefetch, branch prediction, fixed-time instructions, and using retired registers as memory adresses / branch conditions / jump/branch targets / variable-time instruction arguments.

Is it time for open processors?
Posted Jan 10, 2018 17:42 UTC (Wed) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

You can have CPUs with speculative execution but not speculative memory fetches. I.e. CPU will be able to reorder and/or parallelize code like this to better utilize available ALUs:
> a = b + c
> d = a << 2
> e = k + 1

But it won't do any memory fetches.

Is it time for open processors?
Posted Jan 10, 2018 22:43 UTC (Wed) by **brouhaha** (subscriber, #1698) [Link]

Or you can have OoO with speculative execution, but allow speculative loads ONLY from cache. Without doing a lot of simulation, I don't know how much performance that would gain compared to having no speculative loads. Obviously it depends on the data cache hit rate.

Is it time for open processors?
Posted Jan 13, 2018 13:15 UTC (Sat) by **ianmcc** (subscriber, #88379) [Link]

I'd imagine that loading something into cache would be a major performance benefit of speculative execution. We'll have to wait and see what they come up with, but I reckon it will be some kind of separate cache for speculative execution.

Is it time for open processors?
Posted Jan 13, 2018 14:20 UTC (Sat) by **excors** (subscriber, #95769) [Link]

And a separate L2 cache, and L3, and eDRAM, and TLBs? and some other magic to deal with reading data that's currently dirty in another core's cache?

> I don't understand how it could be feasible to read memory without it being observable in some way.
>
> (Even reading from L1 cache might be observable, if it modifies some LRU-replacement state inside the cache.)

Is it time for open processors?
Posted Jan 10, 2018 17:11 UTC (Wed) by **tpo** (subscriber, #25713) [Link]

I think there *is* something fundamentaly wrong with the concept of OoO execution and that is that it makes it hard to reason about computing. The abstraction is not well defined, it's not orthogonal, its boundaries and also it's state machine are unclear. As excors said, Spectre is a symptom of this.

If f.ex. you are confronted with the question:

* how much power does this instruction consume?
* how much time and how many cylcles will it take to execute?
* can it cause an interrupt or be interrupted?
* etc.

then every answer will contain a lot of "it depends" and in the end you won't be certain anyway because an execution of a single CPU instruction is an extremely complex undertaking. Or in other words in the equation:

outputs =f_cpu_instruction(inputs)

you have really no idea what exactly the opaque inputs are and what the visible or invisible outputs will be, if you consider the equation to consist of more than just the opcode and it's parameters. A modern CPU really is a "strange machine".

So the real question IMHO is again: are we sure that complexity and not well defined behavior really are the price we have to pay for performance?

Is it time for open processors?
Posted Jan 10, 2018 21:17 UTC (Wed) by **farnz** (subscriber, #17727) [Link]

The challenge is that moving the "it depends" from silicon to software is known to be a difficult undertaking.

We know from HP's old Dynamo work (a PA-RISC JIT interpreter running on a PA-RISC system) that optimizing code based on runtime information can provide extra performance not available if you optimize based only on information available at compile time, and that the benefit of doing so can outweigh the cost of a JIT interpreter, let alone dedicated silicon that does the job.

We know from the fates of VLIW processors, Transmeta and Intel's Itanium that's it's hard to produce a chip design that's simple to reason about (Itanium, at least, exposed most of the complexity to software, and was easy to reason about at the assembly level, because all the hard stuff that makes modern x86/ARM/POWER etc difficult to reason about is in the software instead of the hardware). Further, the lack of a FPGA Mill (or, indeed, anything other than simulation) implies that they're finding it hard to design a predictable CPU that performs well on the workloads people care about.

Given all this history, I'd expect there to be at least one PhD's worth of advances in the state of the art required, if not multiple, before we can get to a point where high performance CPUs are simple enough to be predictable. I suspect this is why the Spectre researchers expect it to keep haunting us - we are currently facing the choice between sacrificing a significant chunk of both performance and energy efficiency but being secure, or trying to patch the known side channels as we spot them. Neither is a particularly nice place to be in; still, if you ever fancy doing a PhD in computer architecture, this would be a great problem to tackle :)

Is it time for open processors?
Posted Jan 11, 2018 21:12 UTC (Thu) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

"Further, the lack of a FPGA Mill (or, indeed, anything other than simulation) implies that they're finding it hard to design a predictable CPU that performs well on the workloads people care about".

Um, no, they've been ready to start towards an FPGA implementation for a while and have stalled out for lack of the funding required (they estimate about 10m). It's kind of hard to explain the Mill feature set and long term payoff to VCs that would rather find the next big thing in web services without the "heavy semi" level of investment required to make a new chip from the ground up.

I am kind of hoping that post-spectre "because, security" might now be a more valid argument to those with a long term viewpoint or money at risk.

Confession: I am an unabashed Mill fan. It's way less baroque than the itanium was. I'm not going to sit here and write about my favorite top 10 features here but have always encouraged frustrated folk to have a beer, queue up a talk, and inspect the architecture and instruction set, and dream a little. Am I alone in remembering how fun that was in the days Byte Magazine stalked the earth?

https://millcomputing.com/docs/

Even though their business model for the chip logic itself does not have open source as part of it presently, certainly most of the tools will become so (example their compiler is LLVM based).

No matter if they succeed or fail, the ideas they've put forth are worth thinking about.

That said, there is no way that the Mill would enter a Xeon-like market in its first

3 generations - but it could displace 100s of millions of virtual-memory-less DSPs and compete with arm on the low end after it first taped out in a reasonable process. I worry a lot about the security and reliability of all the 260+ cpus in a modern car, for example. I worry about the constant decline in real-time functionality we're seeing in mainstream cpus (and the related rise of unsecured co-processors sharing memory space). Does a tesla need KPTI?

I also am a fan of more software engineers getting interested in EE (programmers and EEs need to start going to the same parties again because they barely share a common language nowadays) . I wish efforts like DARPA's CRAFT program to find ways to accelerate chip design and validation was better funded. I'm a strong supporter of the RISC-V effort (not just to have an open source set of chips but as teaching tools for the next generation of EEs, and in particular, develop better hardware design languages). R&D into software methods in support of better hardware design used to be a vibrant and interesting field back in the 80s. R&D into micro-kernels and capabilities has been mostly dead for years...

After being thoroughly exposed the security exposures getting worse by the day while doing the cerowrt project, I almost dropped out of the bufferbloat project and went to work on the Mill.

Is it time for open processors?
Posted Jan 12, 2018 9:54 UTC (Fri) by **farnz** (subscriber, #17727) [Link]

Honestly, given the cost of doing an FPGA implementation, that sounds like an excuse - I'm not expecting the first FPGA implementation to be particularly high end (it's just a proof of concept, after all). They're spending money on salaries for people to write the simulations already - they can afford to spend that same money on people writing HDLs instead, and a Stratix 10 dev kit is under $10,000. 2.8 million logic elements won't be enough for a high end CPU design, but should be plenty to show that their lowest end design is workable at a reasonable clock speed (say 800 MHz or so). And I've watched all of Ivan's talks - I'm not convinced that it'll actually be possible to implement all of the Mill in an acceptably performant fashion (for an FPGA implementation, competitive with a soft core in the same FPGA - say a low end ARM Cortex); I suspect that it'll be very hard to design something that matches the functional simulations that we've seen and that manages a decent clock speed under timing closure.

I'm not expecting amazing results from an FPGA design, after all - but I've seen enough of the industry to know that if you can't demo your digital logic in anything other than a simulator you wrote, there's likely to be significant issues actually transposing the design from simulation to hardware. The most common foul-up I've seen is cases where, once you implement in hardware, you can't make timing closure without a hugely long critical path; this won't necessarily show in a functional simulation, because you don't have to simulate gate delays.

It's also worth remembering that this class of attack relies on untrusted code sharing the same device as your code; as long as the critical car electronics is kept separate from infotainment, reporting etc, and only connected via Ethernet/CANbus or similar communications methods, they don't apply. Thus, the autonomous driving functionality of a car doesn't have to care about these attacks - while they may well be possible, if the only code running on the autonomous CPUs and the motor/steering control CPUs is trusted and does not accept arbitrary user input (which rules out a music player being part of the trusted domain - you get a blob of data from the user), then the attack is harmless; same can apply within companies, where it doesn't matter if the payroll process can, by being attacked, reveal payroll information to the pay increase process on the same system.

Is it time for open processors?
Posted Jan 12, 2018 12:45 UTC (Fri) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

I think the Mill long ago passed the "show me the gates" stage also.

Although the costs of devkits like the stratix-10 are now quite reasonable, perhaps the ultrascale (50m gates) would be better.

I also don't think it could be a competitive softcore. An FPGA POC would shed doubts and bootstrap software development, only - both direly needed though!

* The width of various busses is a problem
* The stack spiller is complicated
* dozen other things like cache design are not off the shelf

I wouldn't even hazard a guess as to the achievable clock rate in an FPGA. Something greater than 0 would be nice.

Is it time for open processors?
Posted Jan 12, 2018 16:07 UTC (Fri) by **farnz** (subscriber, #17727) [Link]

I wouldn't expect it to be a competitive softcore - as far as I'm concerned, the point of building something like the Mill as a softcore filling an entire Stratix 10 is to show that the performance results you get from simulation are achievable in real hardware, too, and that when you say that you can scale up performance in an ASIC, you're not pulling numbers from your fundament.

Also, to bring this full circle, the point of including Mill in the list was to show that it's not easy to do well at high performance predictable processors. Transmeta failed with an existing (well-understood) ISA and JIT to the predictable CPU; Itanium failed despite having two big tech companies (HP and Intel) pushing it hard. Mill is doing the start from scratch and design everything for predictability thing, and despite over a decade of work from serious engineers (none of the Mill team strike me as clueless dreamers), they've still not got product. So, of the efforts to be high performance and predictable, the only one that has shipped product and not yet been canned is Nvidia's Denver series - but even that's now being paired with "traditional" OoO cores, and we don't yet know whether Carmel is a traditional OoO core, or a Denver JIT core.

Basically, this stuff is *hard*, and so far, there's no constructive proof that we can do a predictable core that's as fast for real workloads as a traditional OoO core with all the tricks.

Is it time for open processors?
Posted Jan 16, 2018 18:27 UTC (Tue) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

just for the record, the mill folk published a paper on spectre/meltdown vs mill:

https://millcomputing.com/blog/wp-content/uploads/2018/01...

I am long-term curious about novel attacks against the PLB management code, however, I'm too low on sleep to care for a good long time.

Is it time for open processors?
Posted Jan 16, 2018 19:37 UTC (Tue) by **excors** (subscriber, #95769) [Link]

That paper seems to get decreasingly confident towards the end. The introduction states simply:

> The Mill CPU is not vulnerable to the Spectre and Meltdown attacks. The Mill is an in-order machine and Spectre and Meltdown as described take advantage of speculative execution on out-of-order machines.

but later they say that it was effectively vulnerable to Spectre because of a compiler feature (it could generate code that performed speculative reads).

Given that code always gets transformed through multiple levels of software/firmware/microcode/hardware before finally executing any operations, and modern "in-order" CPUs (Denver, the Mill, etc) seem to still use speculation to improve performance and just implement it slightly higher in the stack of transformers than traditional OoO CPUs, "in-order" vs "out-of-order/speculative" seems a fairly artificial distinction - what matters for security is the combined result of the entire stack. (At least it's easier to disable speculation if it's happening at a higher level than hardware, since you can push a software/firmware/microcode update instead of new silicon, but it might still suffer the same performance impact.)

They also say:

> The Mill has not been vulnerable to Spectre variant 2 because the Mill has a very short pipeline and low mispredict penalty, so loads erroneously issued will be revoked in time before they have any side effects. This was an unanticipated side effect of the Mill design

which seems to indicate that being an in-order machine *doesn't* make the hardware fundamentally immune to Spectre anyway, it was just a lucky result of their current pipeline design.

Is it time for open processors?
Posted Jan 20, 2018 8:24 UTC (Sat) by **igodard** (guest, #105242) [Link]

The Mill architecture is not vulnerable to Meltdown or Spectre. In the course of verifying that assertion we found a bug in the compiler that would cause it to sometimes schedule loads without the predicates that were supposed to guard them. We fixed that.

The Mill makes no guarantee that system software is bug free; no compiler does. There are bug reports that I filed against gcc that are still open a decade later. If a compiler turns source "(x+y)*z" into machine code that computes "x+(y*z)" then that is a bug in the compiler, not a flaw in the architecture. Likewise, if a compiler turn source "if (b) x = y;" into "t0 = y; if (b) x = t0;" then that is a bug in the compiler, not a flaw in the architecture.

That was the bug we found. The correct machine code computes "b? t0 = y; if (b) x = t0;". It's been fixed.

Is it time for open processors?
Posted Jan 20, 2018 10:09 UTC (Sat) by **farnz** (subscriber, #17727) [Link]

Arguably, this is where the Mill design shows its strength; because almost all the speculation is done in the compilers, you only need to update software as and when a new Spectre-like problem is found. Further, where there's no security boundary between two chunks of code (hence no worry about Spectre - why use a side-channel attack when you can read directly?), Mill can compile in extra speculation, whereas Intel has to leave that speculation out just in case.

Is it time for open processors?
Posted Jan 21, 2018 9:06 UTC (Sun) by **smurf** (subscriber, #17840) [Link]

> if a compiler turn source "if (b) x = y;" into "t0 = y; if (b) x = t0;" then that is a bug in the compiler

Depends on the memory model. If there's no way y might be in inaccessible memory and it's not marked as volatile, I wouldn't consider that change to be a bug.

Is it time for open processors?
Posted Jan 20, 2018 8:06 UTC (Sat) by **igodard** (guest, #105242) [Link]

Show me the gates?

There are critical paths in development as in CPUs. There's no point in doing FPGA work until you have proven in sim that a concept is correct and is what you want and fits with the rest. But you have to feed the sim, so you need a tool chain. And even when you have it all in house, if the sim or FPGA exposes nifty ideas then you can't publish either until the patents issue. And there are iterations at each stage as you find things that don't work, or get better ideas. Heavy semi takes a long time, just like cement plants and steel mills. It takes the Intels a long time too.

Showing gates to tire-kickers is at the very end of that.

Is it time for open processors?
Posted Jan 20, 2018 8:45 UTC (Sat) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

> then you can't publish either until the patents issue
Until you _file_ for a patent.

Is it time for open processors?
Posted Jan 20, 2018 7:50 UTC (Sat) by **igodard** (guest, #105242) [Link]

I wish we could move salaries from simulator to FPGA, but you apparently missed the memo - the Mill project has always been bootstrap, and there never have been salaries for anyone. Our part-timers are dedicated, but still not full time, and still paid only sweat-equity.

Is it time for open processors?
Posted Jan 20, 2018 10:04 UTC (Sat) by **farnz** (subscriber, #17727) [Link]

That just emphasizes my core point; doing a high performance CPU design that's also highly predictable is *hard*. You've been working on this for around 10 years now, and still aren't in a position to make money from it. When you combine that experience with Intel's Itanium, and Transmeta's long term fate, it's clear that the reason that commercially successful high performance designs are all either massively multithreaded (Nvidia GPUs, Intel Xeon Phi) or vulnerable to Spectre (high end ARM, Intel Core) is not that there's a silver bullet we're all overlooking, but that even with very clever people working on a new design, it's a really hard problem, and it's really hard to go from "interesting ideas" to "demonstrable ability to build a CPU that could be competitive given money".

I included Mill because you're attempting something interesting in the "CPUs that are completely predictable" space, and you're not getting huge wins quickly - you're evidence that high performance processor design is fundamentally hard, even if you throw away "conventional wisdom". Given your experience, Itanium, and Transmeta, I feel confident in saying that there's no easy way to design fast CPUs that don't do speculative out of order execution; there are hard ways, but then we're looking at engineering tradeoffs.

Is it time for open processors?
Posted Jan 11, 2018 3:50 UTC (Thu) by **areilly** (subscriber, #87829) [Link]

Many loads? I imagine that there are some that might get to a factor of five, but I'm interested to know what the factor-100 ones look like.

There are plenty of other workloads, of the dense numerical or media-processing variety where out-of-order doesn't buy very much at all, which is why most embedded DSPs are in-order VLIW-shaped, and why the much smaller in-order Cortex A53 is rarely more than a factor of two slower (per cycle) than the much bigger out-of-order cores, on that sort of workload.

I'd be quite interested to know how many of those factor-100 workloads survive latency/throughput tradeoffs against flock-of-chickens style systems, or custom hardware.

Anyway, the world is an interesting place, with many different workloads, so it's hard to generalize. In the absence of much extra cost, of course everyone will buy the fastest single-thread option. But if the cost goes up, as a result of work-arounds, things can change. Or if ways to speculate safely turn out to be less expensive than thought, then it's all back on again.

Is it time for open processors?
Posted Jan 11, 2018 14:59 UTC (Thu) by **excors** (subscriber, #95769) [Link]

Cortex-A53 (in-order) vs Cortex-A57 (out-of-order) does seem to be about a factor of 2 difference in typical benchmarks at the same frequency. (And a greater-than-2 difference in power and area). I guess the problem is that it's hard to push an in-order design significantly further than the A53. You could add more execution units but you won't be able to use them since you can't extract any more parallelism from typical code. You need a short pipeline (since long ones are expensive to stall, and in-order stalls a lot), but that makes it hard to run at higher frequencies. The memory system needs to prioritise latency over bandwidth and capacity. Etc. Meanwhile OoO designs *can* go significantly further than the A57, as demonstrated by high-end Intel CPUs (or even by newer ARM cores) - the main benefit doesn't come directly from OoO itself but from all the other optimisations that become viable once you have OoO.

Is it time for open processors?
Posted Jan 11, 2018 19:45 UTC (Thu) by **areilly** (subscriber, #87829) [Link]

Don't get me wrong: I'm a big fan of out-of-order designs. It's incredibly cool that it can be made to work at all, and you can't say no to the straight-line speed, when it's on offer. Certainly takes a lot of the effort off careful processor-specific optimization at the code and compiler level. But if it turned out that we've been chasing an illusion, and not being able to speculate loads that miss in the cache knocks most of the performance benefit off (I've seen credible measurement of 37% performance loss on Haswell from ISBS MSR protections for Spectre, for compilation tasks) then a re-think might be in order.

I don't agree that A53 is as far as in-order can be pushed, although it is a beautifully balanced design for its niche. It's only two-way scalar, and its SIMD units are only 64-bits wide. There's plenty of room to grow. Even within the ARM world, the Denver cores are 7-ish way scalar in-order and clock as fast as any of the other mobile devices. The in-order SPARC cores were never as fast in single-thread benchmarks as out-of-order cores of the day, but they weren't as much slower than the (older) intel Core parts after the Spectre mitigations are enabled.

I remember the discussions when Alpha was going wide but still in-order, and competing against early out-of-order MIPS cores. The compilers were happy enough to compile some speculation into the code, by turning short contitionals into execute-both-branches-and-conditional-move-result style. Probably not possible to do that over a 200-instruction speculation window like current cores, but you'll get some of the way. And they weren't building wide SIMD vectors back then.

Is it time for open processors?
Posted Jan 11, 2018 23:49 UTC (Thu) by **excors** (subscriber, #95769) [Link]

I believe Cortex-A53's NEON is actually 128-bit wide (or 2x64), for most float and int operations.

The 7-wide vs 2-wide comparison seems a little misleading - I think Denver doesn't have many more execution units than A53, the main difference is that Denver can theoretically decode and dispatch to all 7 at once (when using its custom VLIW ISA, and apparently only in unlikely cases where all 7 micro-ops can fit into one 32B bundle; and apparently it needs an average of 1.8 micro-ops per ARM instruction, so it can practically do maybe 3 ARM instructions per cycle).

It sounds like Denver's ARM-to-VLIW optimiser actually does something equivalent to speculative execution (statically predicting branches and emitting speculative micro-ops to keep the execution units busy), so could be vulnerable to Spectre. At least the optimiser is just software so they could fix it fairly easily (at some performance cost).

In some ways I think that reinforces my (not-very-well-thought-through) argument :-) . Denver's hardware might be technically in-order, but they use lots of the normal out-of-order techniques (with most of the same costs and risks) to achieve just 3 ARM-instructions per cycle. Presumably they wouldn't have bothered with all that complexity if they could have reached the same much-better-than-A53 performance with a pure in-order design.

Is it time for open processors?
Posted Jan 12, 2018 1:22 UTC (Fri) by **excors** (subscriber, #95769) [Link]

Oh, in addition to the static speculation performed by the optimiser, apparently the Denver hardware continues executing speculatively after a cache miss (in the hope that subsequent instructions with not-quite-correct data will pull useful things into the cache) then rolls back once the data arrives. That sounds like a second vector for Spectre, and one that can't be simply disabled in the optimiser.

(This is based on the information in https://piazza.com/class_profile /get_resource/hzbgxhrhoe3...)

Is it time for open processors?
Posted Jan 12, 2018 2:27 UTC (Fri) by **jcm** (subscriber, #18262) [Link]

Forward speculation like that is called "runahead". I believe Intel do similar on some of their cores. As do others. And yes, it's ripe for analysis.

Is it time for open processors?
Posted Jan 11, 2018 19:44 UTC (Thu) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

Stuff with a lot of pointer chasing, like linear algebra on large sparse matrices.

Is it time for open processors?
Posted Jan 11, 2018 19:59 UTC (Thu) by **areilly** (subscriber, #87829) [Link]

True, that is probably the biggest win for designs that can speculate through loads, but that's precisely the sort of code that is going to be hit on the head by the Spectre mitigations anyway. I can imagine an HPC installation (or, indeed, a big-iron database installation) coming to the conclusion that they're going to rely on the fact that they don't run user-provided code on their isolated machines, and their normal malware protection protocols are up to the task: they just won't enable the mitigations. I certainly wouldn't if I was running something on a single-function embedded system.

If your twisty pointer-chasing sparse linear algebra is going to run on someone's cloud infrastructure though, they'll have the mitigations turned on, and it will run badly anyway. So you can re-code to speculate explicitly, in software, (Itanium style), or demand a discount and just turn on some more processors. There have been some nice research papers about running code like that on pairs of cores, (hyper-thread style), with one core of the pair scampering ahead and speculating the memory pattern, so that the speculated lines are in cache, while the other grinds away doing the math, directly out of cache. I doubt that there are many compilers that can construct that sort of code automatically, because the Out-of-order cores did such a great job on their own. Now's the time to dig them up, perhaps.

Is it time for open processors?
Posted Jan 11, 2018 20:16 UTC (Thu) by **joib** (subscriber, #8541) [Link]

HPC systems I'm familiar with tend to have hundreds of accounts, all with shell access (via ssh). And users who are scientists, and not computer experts (many struggle with things like ssh keys etc.). They are a phished account + local root vuln. away from pressing the "nuke from orbit" button. Presumably it's different for supercomputers used for nuclear weapons research..

As for the dual thread idea, maybe you're thinking of "scout threads" that Sun was investigating back in the day?

Is it time for open processors?
Posted Jan 11, 2018 20:27 UTC (Thu) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

SPARC actually explored the "multiple threads" idea. Basically, instead of waiting for a load to finish they switch to another thread. This worked reasonably well only for embarrassingly parallel workloads like web servers.

Is it time for open processors?
Posted Jan 11, 2018 20:21 UTC (Thu) by **joib** (subscriber, #8541) [Link]

> like linear algebra on large sparse matrices.

Luckily for linear algebra we have solutions that don't require speculation for performance, namely vector processing (I mean "real" vector ISA's not the short vector packed SIMD extensions popular in current microprocessors). Or GPU-style SIMT, if that floats your boat. Sure, in the short term it's going to be painful, but we know how to engineer ourselves out of this particular hole.

Many other workloads aren't so fortunate.

Is it time for open processors?
Posted Jan 11, 2018 20:23 UTC (Thu) by **Cyberax** (✭ **supporter** ✭, #52523) [Link]

It doesn't help with sparse matrices, unfortunately. I also encountered the same problems when I was doing bioinformatics, namely DNA assembly.

Is it time for open processors?
Posted Jan 11, 2018 20:34 UTC (Thu) by **joib** (subscriber, #8541) [Link]

> It doesn't help with sparse matrices, unfortunately.

Vectorizing sparse matrix operations tends to be the textbook example for the usefulness of scatter/gather in vector ISA's.

I guess today scatter/gather isn't as awesome as is used to be, as with current arithmetic/bw ratios on mainstream CPU's the CPU becomes memory bound anyway. But with less opportunity for speculation, perhaps there's place for a comeback (AVX-512, ARM SVE, and the RISC-V V extension all have them).

Is it time for open processors?
Posted Jan 17, 2018 15:15 UTC (Wed) by **mstone_** (subscriber, #66309) [Link]

unfortunately the consumers of such products don't have the pocketbooks to support the development of such products--that's why everyone is now running commodity CPUs.

Is it time for open processors?
Posted Jan 10, 2018 16:57 UTC (Wed) by **jezuch** (subscriber, #52988) [Link]

I think Intel tried that with Larrabee. And Sony with Cell. If that's the future then these efforts were significantly before their time :)

Is it time for open processors?
Posted Jan 10, 2018 17:48 UTC (Wed) by **excors** (subscriber, #95769) [Link]

Larrabee failed because it was meant to be a GPU, and it took years to realise their architecture had terrible performance when used as a GPU. But it led to Xeon Phi which has been used in quite a few supercomputers, so it seems the general concept can work for some software.

With Cell, I got the impression the main problem was that the SPEs were complex and unfamiliar - they probably weren't a fundamentally flawed architecture, but game developers didn't have the experience or tools to use them effectively, so they were a huge pain in practice. And they didn't give much better performance than a standard 3-core CPU plus GPU, so they weren't worth the costs. But nowadays game developers are having to deal with 8-core CPUs and GPGPU, and probably expecting future consoles to go even wider, so they're designing engines with task-based parallelism that can easily scale to large numbers of cores and can mix CPU and GPU tasks, so they should be better able to cope with a Cell-like system now.

Is it time for open processors?
Posted Jan 11, 2018 3:40 UTC (Thu) by **jimzhong** (subscriber, #112928) [Link]

I think with the help of powerful optimizing compilers, in-order processors can be competitive with OoO ones.

Is it time for open processors?
Posted Jan 11, 2018 10:16 UTC (Thu) by **farnz** (subscriber, #17727) [Link]

That was what the Intel Itanium was supposed to be - an in-order core that was as powerful as the best OoO cores they could build. It didn't work out that way - the "powerful optimizing compiler" never got good enough to be as fast on the in-order core as OoO cores were when you applied the same compiler power.

Further, the HP Dynamo work gives us reason to believe that runtime optimization will always be necessary to get peak performance; on an OoO processor of the late 90s, they demonstrated that a JIT recompiling native code to native code was faster than just running the native code directly.

I suspect that there are multiple PhDs worth of work to go from where we are today (complexity in OoO hardware, which "JITs" the native code into fast executing code), to a world where in-order CPUs are as fast as OoO CPUs; we need a good runtime JIT to convert a useful intermediate form to something that can be executed, and we need to know what the intermediate form and the something that can be executed should look like.

And looking around at the market, this isn't something that's not been tried: Itanium failed (no-one even wrote a useful JIT for it); Transmeta failed (maybe it could have done better if the code morphing had been open, but I doubt it); Mill hasn't yet got as far as an FPGA model, let alone real silicon (so may never get to a high performance implementation - if you can't do hardware, only simulations, then you run the risk of real hardware massively underperforming for reasons your simulation didn't take into account, like thermals).

Is it time for open processors?
Posted Jan 11, 2018 21:39 UTC (Thu) by **areilly** (subscriber, #87829) [Link]

Nvidia's Denver core is the current poster child for this idea. It gets past the slow startup problem by having a single-issue hardware decode path, so the JIT doesn't have to fire up unless it's busy code. Worked competitively in my old Nexus-9. Would be nice if you could compile your dense linear algebra directly to the VLIW core of you wanted to, imo.

Is it time for open processors?
Posted Jan 12, 2018 10:00 UTC (Fri) by **farnz** (subscriber, #17727) [Link]

> It will be interesting to see what happens with Denver, yes - Nvidia has already gone from "Denver1 is the only CPU core you need" (in the K1) to "Denver2 is paired with ARM's Cortex-A57 so that Denver2 only provides the high-performance cores" (in the X2). If Carmel is effectively Denver3, then that validates the idea (to a degree), but if it's a more "traditional" ARM core, Denver can be added to the list of "couldn't quite make it work out" variants on the "fast in-order core, with complexity in software".

Is it time for open processors?
Posted Jan 11, 2018 4:43 UTC (Thu) by **dvdeug** (subscriber, #10998) [Link]

I've thought about going back to 1970 and give a lecture to a conference on programming languages on modern programming languages. The hardest question would be concurrency. Go's going back to Algol 68 and coroutines, Scala supports threads and actors <i>and</i> parallel data structures, and then we've got stuff like Spark and Hadoop on the large scale and direct CPU vector support at the lowest level. That's a huge range of options and remarkably little consensus about which ones are better. We're already pushing people towards more parallelism, given that quad-cores are becoming standard on desktops, but it's really hard.

Is it time for open processors?
Posted Jan 12, 2018 6:44 UTC (Fri) by **paulj** (subscriber, #341) [Link]

That's exactly the premise behind the UltraSPARC T1 and T2: Simple in-order core. Instead of trying to extract parallelism out of a single execution stream, it instead optimises for the software explicitly being coded for parallelism by creating threads. If a thread of execution stalls on memory, the core switches to another.

The benefit to ditching all the logic and long pipelines to support OoO and speculative execution, and using simple, in-order execution, is that you can pack a lot more of those cores into the same transistor budget.

So you get a CPU that scales up really to handle loads that scale-out via threads of execution. The downside is that any single thread of execution is (much) slower than on a highly pipelined, speculative, OoO processor.

Is it time for open processors?
Posted Jan 9, 2018 23:46 UTC (Tue) by **immibis** (subscriber, #105511) [Link]

How much does it cost to get an OS kernel into production in billions of devices?

Is it time for open processors?
Posted Jan 10, 2018 0:05 UTC (Wed) by **balkanboy** (subscriber, #94926) [Link]

You can always force a paradigm shift by introducing a good crisis just like this one w/Spectre & Meltdown, particularly if it affects large groups of people among which are also CPU manufacturer shareholders. One doesn't get Intel to budge unless there's something in it for them or you introduce a real threat to their existing, business-as-usual model.

I know I shouldn't be, but I am kind of glad this happened - it provides a major kick in the ass for Intel and makes AMD looks good - they needed a boost like this in addition to a stellar new CEO, Lisa Su.

Is it time for open processors?
Posted Jan 13, 2018 7:20 UTC (Sat) by **alison** (subscriber, #63752) [Link]

> "you'll *never* see a high end Xeon-class core unless some billionaire
> funds it as a pet project, and keeps investing year after year for the
> greater good."

sed -i -e 's/some billionaire/Huawei/g' and the plan starts to sound possible.

Is it time for open processors?
Posted Jan 13, 2018 7:22 UTC (Sat) by **alison** (subscriber, #63752) [Link]

> "you'll *never* see a high end Xeon-class core unless some billionaire
> funds it as a pet project, and keeps investing year after year for the
> greater good."

sed -i -e 's/some billionaire/Huawei/g' and the plan starts to sound possible.

Is it time for open processors?
Posted Jan 21, 2018 9:38 UTC (Sun) by **lkcl** (guest, #60496) [Link]

well, by accident i have encountered a potential solution, there. a couple of months ago i contacted the head of the shakti team in india, and was surprised that he was extremely enthusiastic to hear from me. whilst extremely busy with an experimental 20nm low-power tape-out, he did have time to communicate that he had basically been given UNLIMITED resources by the Indian Government to, and i quote, "Piss All Over ARM And Intel".

due to the sheer volume of the market in india he is being taken seriously by various companies who have offered him FREE access to tools, FREE access to top-end foundries for experimental (MVP) samples at 20nm, 28nm and 40nm (one of each), and he in turn has offered the open hardware and free software community ACCESS to that opportunity.

so all the costs normally associated with getting a processor out the door are GONE.

how d'ya like them apples? :)

http://rhombus-tech.net/riscv/shakti/m_class/

Is it time for open processors?
Posted Jan 9, 2018 15:29 UTC (Tue) by **joib** (subscriber, #8541) [Link]

> Meanwhile, for low-end applications, there is a compressed instruction-stream format intended to reduce both memory and energy needs.

FWIW, to nitpick, this isn't specifically for low-end applications.

In fact, one of the stated reasons why the compressed extension is an extension and not part of the base ISA is that for the lowest of the low end, the extra complexity in the decoder might not be worth it.

Is it time for open processors?
Posted Jan 9, 2018 15:51 UTC (Tue) by **ken** (subscriber, #625) [Link]

Does there exist even a a single out of order open implementation of a CPU?

Is it time for open processors?
Posted Jan 9, 2018 16:07 UTC (Tue) by **mtaht** (✭ **supporter** ✭, #11087) [Link]

https://github.com/ucb-bar/riscv-boom is an OOO risc-v core.

Building a risc-v core is almost as simple as typing make, btw, after you install the java/scala /chisel dependencies. You'd need a supported FPGA board (the Zynq series is good) and a few proprietary tools to finalize and write the code, and your cpus will grind for hours or days compiling it all, but that's it.

Ironically, once you do all that, you now have a co-processor living in the much-the-same physical memory space as the main processor, with all the security headaches that entails.

Is it time for open processors?
Posted Jan 9, 2018 17:19 UTC (Tue) by **ejr** (subscriber, #51652) [Link]

The RISC-V folks have silicon as well. And that chip uses an in-order RV32I for very effective, state-of-the-art power management. The relevant publications are on either Krste's or Bora's pages.

Is it time for open processors?
Posted Jan 9, 2018 20:39 UTC (Tue) by **aleXXX** (subscriber, #2742) [Link]

This is written in Scala ? I expected VHDL or something...
How do you describe hardware using a more or less normal programming language ?

Is it time for open processors?
Posted Jan 9, 2018 21:43 UTC (Tue) by **nybble41** (subscriber, #55106) [Link]

> How do you describe hardware using a more or less normal programming language ?

By using an "embedded Domain-Specific Language" library (in this case, Chisel[1]) which generates Verilog when the Scala program is run, similar to CλaSH for Haskell[2] or MyHDL for Python[3]. What these all have in common is that they allow you to use your language of choice as a preprocessor or "template engine" to generate low-level HDL from relatively high-level descriptions.

[1] https://chisel.eecs.berkeley.edu/
[2] http://www.clash-lang.org/
[3] http://www.myhdl.org/

Is it time for open processors?
Posted Jan 9, 2018 22:26 UTC (Tue) by **mtaht** (✭ **supporter** ✭, #11087) [Link]

Chisel is one of the few bright spots in hardware construction languages to date, especially as it has been used to build real processors, and comes with an increasingly large library of common chip components. SystemC has grown in popularity too (no open source implementation, however). Underneath it all, verilog and VHDL are better than in the 80s, but still a crufty mess that you are lucky to get something that works 1 time in 10. A lot of parameterizing is adhoc and driven by scripts. Extensive validation and simulation is needed of the result.

Open sourced verilog designs are like open sourcing the buggy, uncommented, assembly language version of the high level program compiled with -O3.

I am partial to asynchronous circuits (stuff without a central clock is lower power, doesn't need explicit power management, and emits less noise, good for sensitive radios).
The http://www.async.caltech.edu/Pubs/PDF/chpasync2012.pdf chp design language and compiler were open sourced a ways back (and is somewhere on github I think but so far can't find it). We've seen a few interesting new chips built around async logic recently in the AI space.

Is it time for open processors?
Posted Jan 11, 2018 15:48 UTC (Thu) by **kpfleming** (subscriber, #23250) [Link]

There's this too: an asynchronous DSP for media processing workloads. Vastly lower power consumption.

http://www.octasic.com/product/oct2224w/

Is it time for open processors?
Posted Jan 10, 2018 3:07 UTC (Wed) by **pabs** (subscriber, #43278) [Link]

There are open tools for FPGA synthesis too:

https://symbiflow.github.io/

More FPGA links here:

https://wiki.debian.org/FPGA

Is it time for open processors?
Posted Jan 10, 2018 7:58 UTC (Wed) by **michaeljt** (subscriber, #39183) [Link]

Doesn't an open source CPU on an FPGA just shift the closed source part down? I might be wrong, as I don't know much about FPGAs. And the Zynq series seem to have a (presumably closed-source) ARM inside them - does that replace part of what would other be done in programmable logic?

Is it time for open processors?
Posted Jan 10, 2018 16:47 UTC (Wed) by **somlo** (subscriber, #92421) [Link]

> Doesn't an open source CPU on an FPGA just shift the closed source part down?

True, but the closed part is just a large grid of mostly identical configurable logic blocks (CLBs) with a programmable interconnect that builds your hardware design more or less like on a nano-scale "breadboard". As such, there should be much less magic in those closed bits than there would be in a whole closed ASIC.

> And the Zynq series seem to have a (presumably closed-source) ARM inside them

That's a hybrid ASIC/FPGA, where they added a pre-optimized-in-silicon "hard IP core" for applications that frequently require an ARM chip to free up generic CLBs for other uses. Not interesting/useful if one's goal is to have an open CPU design running on the FPGA. Ultimately, one can simply choose to ignore the closed hard IP core(s) and just utilize the generic CLBs for everything.

Is it time for open processors?
Posted Jan 10, 2018 23:07 UTC (Wed) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

I mentioned the Zynq series with the arm processor in them because they are an easier way for programmers to get into futzing with an FPGA and their related toolchains, with a lot of fairly cheap boards out there with decent linux support.

It's a lot easier to poke at problems in a FPGA assist when you already have a working processor on-board.

Pure FPGA work requires you reach for the logic analyzer immediately.

Is it time for open processors?
Posted Jan 12, 2018 12:56 UTC (Fri) by **mtaht** (✯ **supporter** ✯, #11087) [Link]

boom v2: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-20...

Is it time for open processors?
Posted Jan 9, 2018 16:07 UTC (Tue) by **jebba** (✯ **supporter** ✯, #4439) [Link]

Berkeley Out-of-Order Machine (BOOM), RISC-V:

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-20...

Is it time for open processors?
Posted Jan 9, 2018 17:13 UTC (Tue) by **palmer** (subscriber, #84061) [Link]

Well, I certainly hope it is :)

Is it time for open processors?
Posted Jan 9, 2018 18:43 UTC (Tue) by **joib** (subscriber, #8541) [Link]

There was recently this article about a 17-year old high school student (!) doing his own integrated circuits in his parents garage: https://spectrum.ieee.org/semiconductors/devices/the-high... His plan is apparently to recreate the Intel 4004, which isn't that far off from a 6502 in terms of complexity (IIRC around 2k vs. 3.5k transistors). So, open source C64 here we come!!111

The remaining problem, to port Linux, a desktop environment, and firefox.. :-/

Is it time for open processors?
Posted Jan 9, 2018 18:51 UTC (Tue) by **joib** (subscriber, #8541) [Link]

I forgot to mention, in the RISC-V world there is the "Micro-riscy" from the pulp project (http://www.pulp-platform.org/) that comes in at 11.6 kGE (so roughly 45k transistors in CMOS) which I would guess is out of reach for a garage non-cleanroom fab like said high school student has.

Is it time for open processors?
Posted Jan 15, 2018 13:23 UTC (Mon) by **gtg** (subscriber, #84695) [Link]

You mean a bit like this guy? www.megaprocessor.com ;-)

Is it time for open processors?
Posted Jan 15, 2018 20:00 UTC (Mon) by **joib** (subscriber, #8541) [Link]

Wow! That's, uh, quite cool! :) Thanks for the link.

Is it time for open processors?
Posted Jan 16, 2018 16:26 UTC (Tue) by **nix** (subscriber, #2304) [Link]

Actually it probably generates quite a lot of heat. Its power draw is going to be... high. :)

Is it time for open processors?
Posted Jan 9, 2018 18:57 UTC (Tue) by **excors** (subscriber, #95769) [Link]

> RISC-V is indeed not vulnerable to [Meltdown and Spectre] by virtue of not performing any speculative memory accesses.

Is that true? Surely it depends on the details of any particular implementation of the RISC-V ISA, and they could choose to do speculative memory accesses, or could check access permissions slightly too late in their pipeline, just like implementations of x86 and of ARM.

The post from RISC-V only says that the Rocket processor and all "announced RISC-V silicon" are not vulnerable, which is very different to saying "RISC-V is not vulnerable". BOOM sounds like it might be vulnerable to Spectre.

Is it time for open processors?
Posted Jan 9, 2018 19:05 UTC (Tue) by **joib** (subscriber, #8541) [Link]

While Meltdown and Spectre are fascinating and on everybodys minds right now (and Spectre-style attacks will likely be with us for a long time), the vast majority of vulnerabilities are still "normal" software ones such as buffer overflows.

Speaking of hardware, if we aren't rewriting everything in Rust or some other memory safe language, one way to make plain C safer would be to use hardware-enforced bounds checking such as the CHERI project (https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/), which adds hardware support for fat pointers that can be used as capabilities (for a quick overview, https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/201...).

Of course, such an approach suffers from a chicken-and-egg problem; if mainstream architectures aren't supporting it, people won't port software for it either.. *sigh*

Is it time for open processors?
Posted Jan 9, 2018 22:30 UTC (Tue) by **ballombe** (subscriber, #9523) [Link]

I do not know if we need an open processor, but we need a processor with separate kernel/user address space, as sparc and s390 do.

Is it time for open processors?
Posted Jan 9, 2018 22:42 UTC (Tue) by **taintedbit** (subscriber, #108080) [Link]

As a bit of a digression, printers (of the inkjet and laser variety, not the 3D variety) also seem like they would benefit from a successful open-source hardware project.

Printers are notorious for taking control away from and acting hostile towards their owners and users (e.g., DRM embedded in consumables, buggy proprietary software packages, invisible tracking dots, remotely exploitable security flaws, and more). Additionally, although I am not an electronics expert, it seems plausible to me that someone could build an open-source printer from a kit in their home.

Every few years I do a few quick searches on the topic, and although a few incomplete projects have come and gone, I have not yet seen any lasting, comprehensive, and accessible open printer project. The main reason for this seems to be a combination of fears about patents, laws against reverse engineering, lack of demand (which I find hard to believe), and unexpected technical complexity. Hopefully I have simply missed the existence of such a project, but if not, I hope that the current discussions about open-source hardware might inspire some people with the necessary expertise, both technical and legal, to investigate the subject.

Is it time for open processors?
Posted Jan 10, 2018 0:06 UTC (Wed) by **pabs** (subscriber, #43278) [Link]

Speaking of which, the GNU origin story involves printer software. Does anyone know if it was printer drivers or software running on the printer itself?

Is it time for open processors?
Posted Jan 10, 2018 1:15 UTC (Wed) by **pizza** (subscriber, #46) [Link]

I don't see it happening, for a simple reason -- the "open source" printers will be competing with commercial printers that are already sold at or below cost thanks to cutthroat competition, to say nothing of the secondhand market for the better-made stuff.

(And I say this as someone who writes FOSS printer drivers...)

Is it time for open processors?
Posted Jan 10, 2018 2:23 UTC (Wed) by **karkhaz** (subscriber, #99844) [Link]

Printers are sold below cost because the ink is sold at extortionate prices. Anybody with any sense would invest in a printer that was priced the other way round, i.e. a more expensive printer that uses non-locked-in and presumably cheaper consumables.

Is it time for open processors?
Posted Jan 10, 2018 2:33 UTC (Wed) by **rahulsundaram** (subscriber, #21946) [Link]

>Anybody with any sense would invest in a printer that was priced the other way round, i.e. a more expensive printer that uses non-locked-in and presumably cheaper consumables.

That really depends on how often you print. Cheap printer and expensive ink is perfectly fine if you print occasionally and like the convenience of a printer of home but don't print often enough for the running cost of ink to be a problem.

Is it time for open processors?
Posted Jan 10, 2018 7:52 UTC (Wed) by **michaeljt** (subscriber, #39183) [Link]

> Cheap printer and expensive ink is perfectly fine if you print occasionally and like the convenience of a printer of home but don't print often enough for the running cost of ink to be a problem.

I believe that inkjet printer nozzles can (could?) be destroyed by dried out ink if they are used too infrequently. Might be wrong though.

Is it time for open processors?
Posted Jan 10, 2018 16:48 UTC (Wed) by **nybble41** (subscriber, #55106) [Link]

> I believe that inkjet printer nozzles can (could?) be destroyed by dried out ink if they are used too infrequently.

That has been my experience. I only have occasional need for hardcopies (a few times a year—tax forms and the like, and one-off projects), and eventually switched to an HP Color LaserJet CM1312nfi at home after observing that I was replacing ink cartridges roughly every other print job due to the ink drying out. The laser was more expensive up front—and replacement toner isn't cheap either—but since toner doesn't expire nearly as quickly as ink I can actually use the entire cartridge, drastically reducing my cost-per-page. I've had my laser printer for nearly a decade now and only had to replace the black toner once. As a bonus, the quality is higher for text and diagrams, and it even manages passable photos.

Is it time for open processors?
Posted Jan 11, 2018 2:55 UTC (Thu) by **JanC_** (subscriber, #34940) [Link]

I switched to a monochrome laser for the same reason, as pretty much everything I printed didn't really _need_ any colours (and those aren't really all that more expensive than inkjets nowadays).

Is it time for open processors?
Posted Jan 17, 2018 15:28 UTC (Wed) by **mstone_** (subscriber, #66309) [Link]

Yup, I had my old laserjet 4L for something close to 20 years with only a few toner cartridge replacements. Finally broke down and replaced with a color laser when it got too hard to find a replacement cartridge & parts. Tried an inkjet for a while, basically needed a new ink cartridge every time I printed, and that gets really expensive really fast. The inkjet had beautiful quality for pictures, and the ability to do full bleed output (hard on a laser), but I don't generally need display-quality pictures "right now", so outsourcing that rare need is a no-brainer.

Is it time for open processors?
Posted Jan 10, 2018 16:08 UTC (Wed) by **khim** (subscriber, #9252) [Link]

You assume people are rational. They are not. And, worse, producers tend to be rational while final consumers are not (because irrational makers go bankrupt while irrational consumers don't).

Thus we have what we have.

Is it time for open processors?
Posted Jan 18, 2018 16:39 UTC (Thu) by **massimiliano** (subscriber, #3048) [Link]

You assume people are rational. They are not. And, worse, producers tend to be rational while final consumers are not (because irrational makers go bankrupt while irrational consumers don't).

Thus we have what we have.

Now, *this* should go into the "quotes" section!

Programmable hardware
Posted Jan 10, 2018 3:57 UTC (Wed) by **songmaster** (subscriber, #1748) [Link]

Given the availability of really big high-speed FPGA chips today it surprises me that we don’t hear about them being used as the basis for an open CPU design (and I’m not talking about those chips that already have an ARM or similar core built into them, that isn’t the point of this idea). I understand that they aren’t going to be as fast as a custom-developed chip, but raw chip speeds have been going up much more slowly in recent years and the ability to reprogram the hardware ought to make for some interesting ideas.

I could see multiple CPUs on different chips and each programmed or optimized for a different workload or part of the problem — that one is currently running Python bytecode, there’s a JVM over here with a couple of cores, and the security processor over there is doing TLS and SSH. Of course the result is going to need an OS that is decidedly not SMP or even big.LITTLE. Any takers?

Programmable hardware
Posted Jan 10, 2018 16:53 UTC (Wed) by **somlo** (subscriber, #92421) [Link]

IIRC there's a partial port of Fedora to RISC-V, and there's even a way to run it on an FPGA board using a Xilinx Artix7 chip: https://fedoraproject.org/wiki/Architectures/RISC-V

What's missing right now is a way to build the FPGA bitstream from Verilog (or Chisel) sources using completely open tool chains, although there's some work in progress to address that (see https://symbiflow.github.io/ mentioned in an earlier comment).

Programmable hardware
Posted Jan 10, 2018 22:55 UTC (Wed) by **brouhaha** (subscriber, #1698) [Link]

Soft-core processors in an FPGA are about two orders of magnitude slower than the fastest single-core x86 performance. Soft-core processors are great for some things, but replacing general-purpose processors is for the most part not one of them.

Soft-core processors are mostly useful if you're going to have an FPGA anyhow, for some other reason, and can throw in a low-ish performance processor little or no extra hardware cost.

As a hobby I've developed some 8-bit soft core processors, including one compatible with the RCA CDP1802, which was the first 8-bit CMOS single-chip microprocessor, circa 1976 (famous for use in the COSMAC ELF microcomputer). On recent FPGAs, my 1802 core runs at least 70 times the maximum instruction execution rate of the original, but that's with 40 years of hardware advancement.

Is it time for open processors?
Posted Jan 10, 2018 7:13 UTC (Wed) by **sampablokuper** (subscriber, #53150) [Link]

No mention of LowRISC?

Must watch by bunnie on open hardware
Posted Jan 10, 2018 13:25 UTC (Wed) by **guerby** (subscriber, #108731) [Link]

IMHO a very good tour of remaining issues around open hardware processors by bunnie :

Keynote Address: Impedance Matching Expectations Between RISC-V and the Open Hardware Community
blog : https://blog.hackster.io/death-of-moores-law-makes-open-h...

video : https://www.youtube.com/watch?v=zXwy65d_tu8
slides : https://riscv.org/wp-content/uploads/2017/05/Wed1100-impe...
author blog : https://www.bunniestudios.com/

Is it time for open processors?
Posted Jan 15, 2018 20:38 UTC (Mon) by **metasequoia** (subscriber, #119065) [Link]

Since a processor would be an international non-profit effort, a standing problem comes up - no mechanisms exist which can protect the rights of people, or FOSS and its 'gift to everybody' in what now amounts to an intentionally amoral supranational legislative space where corporate rights have been elevated above those of nations, and people and their rights and interests are intentionally absent, in order to enforce a lock down that allows only one possible future for the planet, one where everything is owned and nothing shared, a corporate centric world view so instnctively wrong to most people that its been hidden from sight while what amounts to a global oligarchy works feverishly to lock it into place irreversibly in as many ways as possible.

To carve demonstrably bad, undemocratic policy in stone forever, basically. This seems to me like a kind of madness.

There may also be current secret law, involving hardware, which if it is the case, I suspect would be because these agreements likely - in order to give all nation states equal rights - when dealing with multinational corporations, the agreements likely require some back door mechanisms they can all be given equally.

This is I suspect the case.. In these deals, there is no standing for any representatives of the whole worlds people as people are only instantiated through the country they are the market of.

See the problem?

We, the people, no longer have representation in this sphere. Its become "We the corporations". This is not in the abstract, its been done physically. Sure, we have governments, but their primary responsibility increasingly, because people were never made aware of the fact that this was being done, are to act on behalf of their corporations. In particular the newer agreements being promoted clearly are a sort of second enclosure transferring everything of value to corporations. For example, creation of any new public services of all kinds seem to be prohibited "except services supplied in the exercise of governmental authority": "a service supplied in the exercise of governmental authority' means any service which is supplied neither on a commercial basis, nor in competition with one or more service suppliers." So almost no public services end up qualifying for protection.

There is an urgent need for new structures to represent people in this sphere.