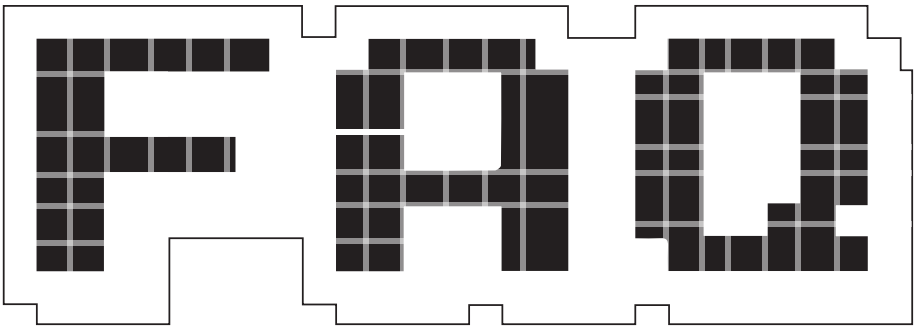


OPENRISC

Open source software is all good and well – but how about open source CPUs?

MIKE SAUNDERS

Q So, this is free software clone of a certain famous board game, with some kind of pun in the title, right?

A No, this has nothing to do with Risk. It's all about CPU cores here – and specifically, RISC (reduced instruction set computing) ones. OpenRISC is a project that designs completely open processors, which you can study, modify, and have produced in factories. Well, providing you have enough money to do that, of course...

Q Hang on a minute! Reduced instruction set? Why would I want that? Give me more, more, more instructions!

A Actually, RISC designs go back a long way, and have nothing to do with the overall power of a CPU. They're all about designing the processor so that it has fewer jobs to do – but it can do them more quickly and efficiently than in other designs. Historically, x86 CPUs have been complicated beasts, with each generation adding more and more layers of cruft onto the original design. You end up with some CPU instructions taking just a few clock cycles to execute, whereas others take far longer.

> **"RISC designs go back a long way, and have nothing to do with a CPU's overall power."**

RISC, however, opts for a much smaller range of instructions (and therefore fewer transistors on the chips), and these instructions are highly optimised. Fewer instructions doesn't mean less capability though; after all, CPUs just move numbers around in memory and perform calculations on them. And today, RISC has won: ARM chips are built with RISC architecture and are absolutely everywhere, and even though x86 CPUs are still regarded as complex (Complex, rather than Reduced Instruction Set Computing) in their architectures, modern ones break instructions down into smaller components, in a RISC-like fashion.

Q OK, but CPUs are hardware – how do you make them open source? And why would you want to?

A Yes, hardware doesn't grow on trees (unless you have some fancy wood panelling for your PC case), so initially it might seem odd to apply FOSS principles to it. But consider designs: can you take your current PC or laptop processor, get a complete specification for it, change a few parts and make your own version? Unless you're willing to pay giant licensing fees to Intel or AMD, this probably isn't an option. And even then, you wouldn't necessarily get to share your changes with the rest of the world.

Now, imagine if the entire design of your CPU were open. Imagine if the community could work together on improving its performance, features and power management. Imagine if this could work in tandem with an entirely free software stack, so that every byte and electron of your computer was free as in speech.

Q That sounds like heaven for Richard Stallman, but how many geeks really want to fiddle with CPU designs?

A Just because it seems like a black art, or an obscure subject, it doesn't mean that nobody is interested in it. On the contrary – at the time of writing, the third annual OpenRISC Conference (ORCONF 2014) was taking place in Munich, Germany. Forty developers from around the globe gathered together to discuss the current state of OpenRISC, share projects that they're working on, and contemplate ideas for the future – see page 16 for a full report.

Q So, how are OpenRISC chips made? How is development done, and who produces them?

A OpenRISC came to life in 1999, the work of a few computer science students in Slovenia, and received a small amount of financial backing in the early 2000s. Today, the OpenRISC project is much bigger and has a few chip designs (known as "cores"), but currently the focus is on the OpenRISC 1000, also known as OR1K: **http://opencores.org/or1k**. This CPU is focused on networking and embedded tasks, with special emphasis placed on simplicity and low power consumption. Its design is constructed using the Verilog hardware description language – and indeed, you can see the source code for yourself at **https://github.com/openrisc/mor1kx/tree/master/rtl/verilog**. This code is released under the GNU Lesser General Public Licence, so you're free to base your own chip on it, as long as you make your designs available for everyone else.

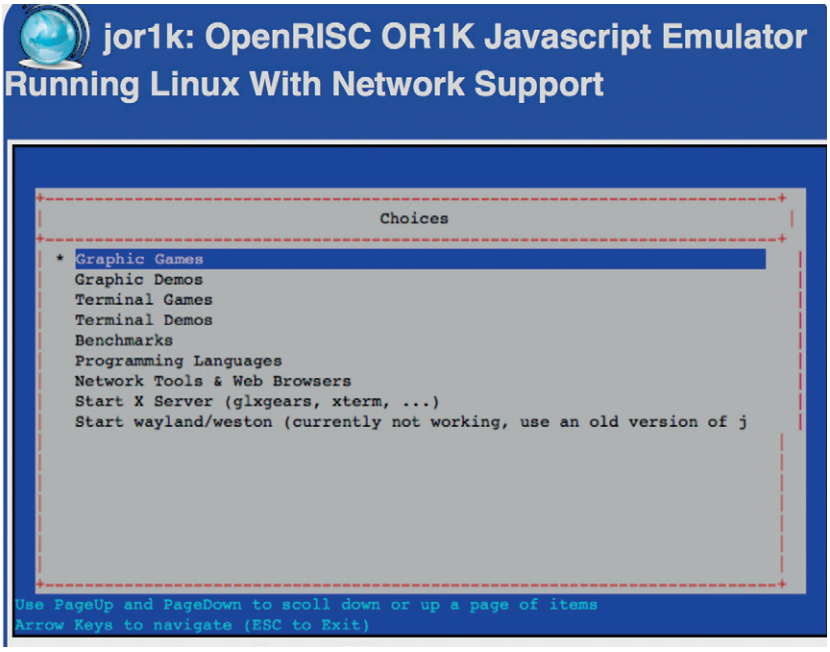
For development purposes, the OR1K design can be implemented in FPGA (field-programmable gate array) development boards, which means you can hook up an OpenRISC processor to various peripherals and create a fully working computer. The GNU toolchain has been ported so that it can produce OpenRISC binaries, and work is underway to get *LLVM/Clang* to the same state as well. The Linux kernel itself has been able to run on OpenRISC since version 3.1, and a few other real-time operating systems such as eCos have OpenRISC ports as well.

OpenRISC is popular in embedded devices, and at ORCONF one developer talked about a project using OpenRISC chips in TV set-top boxes to convert satellite video data to internet streaming formats. Other embedded chips had been considered, but it was important for the set-top box manufacturer that the chips could run the Linux kernel, so OpenRISC was chosen.

Another area in which OpenRISC is well known is academia. The Technical University of Munich uses it in research (which is why the conference was held in the city), while many other universities teach courses based on it.

Q **What about the chip in the real world, though – are there any commercial implementations?**

A You bet! This isn't just a play toy for hackers and students.



Experiment with OpenRISC in your browser: try the JavaScript-based emulator (running a Linux kernel) at http://s-macke.github.io/jor1k.

OpenRISC has been implemented in many SoC (system on a chip) designs – that is, chips that include all the components needed for a standalone computer. Most notably, Samsung has used it in various digital TV models, while the Allwinner A31, a SoC used in a large range of mobile phones, tablets and smart TVs, has an OpenRISC core doing power management work.

OpenRISC has even gone into space as part of the TechEdSat project, which is a satellite designed by students at San Jose State University – it needed to be cheap, so OpenRISC was a natural choice. It could be used in many other devices and projects as well, that we don't even know about, just as Linux and the BSDs are often chugging away inside networking and embedded hardware, without any fanfare.

Q **Is OpenRISC finished, or is it still a work in progress?**

A With 80,000 lines of Verilog behind the design, there's still plenty of room for tweaks, performance improvements and power savings. A team of developers is beavering away on a new CPU pipeline – that is, the series of stages used to process instructions. Ideally, it will support out-of-order execution, along with dynamic branch prediction. Another topic that came up at the conference was improving the 64-bit version of the chip. There's even some interest in porting FreeBSD to OpenRISC, although we'd wager that NetBSD will get there first, given that it already runs on a staggering number of platforms...

Q **OK, you've sufficiently whetted my appetite, and now I want to make big bucks as a CPU designer. Where do I start?**

A Well, we can't guarantee that you'll be the Chief CPU Architect at Intel after playing around with OpenRISC for a few months, but it's a great way to dip your toes into the vast wold of processor design. A good place to begin is the Getting Started guide at **http://kevinmehall.net/openrisc/guide** – and note the requirements in particular. You should have a solid grounding in the workings of a Linux installation, along with knowledge of Verilog and C (the language used to code the Linux kernel).

The guide also explains how to run *or1ksim*, an OpenRISC CPU emulator, and how to use the chip on an FPGA development board. You can pick up one of these boards for around £50 to £100; see **http://opencores.org/or1k/FPGA_Development_Boards** for a list of boards that work with OpenRISC. But if you really want to dive in at the deep end, try the Architecture Manual at **http://tinyurl.com/qj6pjfc** – just bear in mind that it's 358 pages long!

< TIM BRAY />

Graham Morrison meets the co-author of XML, a fierce privacy advocate, an encryption hacker and the owner of many hats.

Watching Tim Bray talk to an audience is a little intimidating. He talks fast and every word counts. And he wants action – he wants his audience to change the world After founding companies, co-authoring the XML specification, working at Sun Microsystems and then Google (leaving because he famously didn't want to leave Canada for Silicon Valley), Tim has seen, thought and talked about most things to do with technology. He's even making his own security contributions to the amazing open source Android email application, *K-9*. His keynote at OSCON 2014 was about threats – threats to our privacy, threats to our online freedoms and threats to our data, and "Now is the time for sensible, reasonable, extreme paranoia," as he puts it. Which is exactly what we wanted to talk about when we met with him.

How do we balance the advantages of big data with healthcare systems and privacy? And how can PGP help with this?

Tim Bray: Well Larry (Wall, the creator of the Perl programming language) certainly puts things into sharp perspective. You know, if you're willing to trust people with your health data maybe we can save a lot of lives, and yet there are a substantial number of people who are reasonably paranoid about trusting their data to anybody. So if you're going to do that, you should be transparent about how you're going to handle the data, how you'd protect it and who you'd share it with, and so on.

But I don't think that's anything like a complete answer, because the vast majority of people don't really have the education or tools to understand what the data protection options are, and what the consequences of sharing are. And we shouldn't expect them to. It's a hard area of expertise and not one that we should expect civilians to have. So I think there's actually quite a strong role for the public sector in here to establish regulations, because this is not an issue of technology, it's purely an issue of policy. Now in the US, they have such regulations, the most visible of which is called HIPAA, which is widely used and strictly enforced in the healthcare sector. And, whereas I think almost anybody would agree that it's reasonable to have such regulations, the HIPAA itself is not actually very good. In particular, it does things like mandate the use of specific technologies such as passwords when, quite possibly, better alternatives to passwords would be more secure and less onerous are available.

So I would probably appeal to a combination of market forces and public regulation to assume the default action is the right action, is the ethical action, is the action that produces the least surprise on users. If we've learnt one thing, no matter how many information popups you put there saying what you're going to do with the information and what the consequences are, a high proportion of people will say 'ah, where's the OK button, I just wanna get through this'. And whether you like it or not, that's just the way people are. So I think that we, as a profession, have a responsibility to do the ethical thing by default and not surprise people with what happens to the information they share. As you point out, OpenPGP is a well established, fully debugged cryptography framework that is highly implementable and has been implemented lots of times, and yet its penetration among the non-geek community is more or less zero percent. If we're not going to make it easy to use by default, then we're not going to have privacy by default and I think our profession owes the community, the people that use our stuff, privacy by default.

If you look at HTTPS (ignoring Heartbleed), even people without technical knowledge can

> "We owe Mr Snowden a lot of credit for exposing people to the fact that their governments are doing things of questionable legality, morality and effectiveness."

understand that a site is encrypted, at least in principle. The certificate sharing is all done transparently and there's an authority that deals with the authentication of the certificate. Maybe that's what we need?

TB: Whenever anyone mentions HTTPS, flawed though it is, I have to push back a little bit. Yeah, it's possible to screw up the installation and deployment and so on, so that it doesn't work properly. But when it is deployed properly, which is not that hard, there is not a known instance of HTTPS data being brute-force decrypted. It is plenty good enough. And yes I agree with you that people are starting to be educated about what the little lock means.

> "We, as a profession, have a responsibility to do the ethical thing by default."

One of the things that I'm disappointed about though is the community of developers that are still offering services over HTTP without even allowing HTTPS let alone requiring HTTPS. What they should all do is go to their legal department and ask a legal opinion, 'Hey, we're offering our services in such a way that anyone sent onto the company can spy on us – is that legally OK?' Well, you know what, it very probably isn't!

LV Even on Android, we don't know of an email client with decent S/MIME PGP support.

TB: So *K-9* is getting it.

LV But it's taken a couple of years to get it into the latest alpha.

TB: Well, one of the problems with cryptography is key management, and how do you find the key for the person you want to send a secure message to. Historically, the cypherpunk community who invented all this stuff were a little bit guilty of letting the perfect be the enemy of the good, and so they set up this web of trust mechanism, which never worked – ordinary people won't go to key signing parties. I'm sorry, it's not going to happen.

So I've been super interested in this thing called keybase.io, which provides a pretty nice solution to the problem of acquiring a key with good reasons to believe in it. And, what's great about it is that you don't even have to trust keybase.io. They present everything


"I don't care if Apple has signed it, I want to download an app that has previously been audited and found to be safe."

that you need to know in proofs that you can independently verify if you want to. I thought the difficulty of actually discovering and acquiring keys might be the single largest remaining logjam, or one of them anyhow, in making quality cryptography available. Now I think we're ready to make that one go away, and so I'm optimistic.

LV GnuPG for email clients can work completely transparently as far as key exchange and key discovery go. You don't even have to know anything about the keys.
TB: And particularly in the case of keybase, you just go 'I'm looking for a key for you', and it says 'Oh, here's a key, which is also controlled by the person who controls your Twitter account and your GitHub account'. OK, that's good evidence, I'll believe that that key is a good one to encrypt my message with.

LV When do you think this became such a big issue? Has it always been an issue, or has it become more an issue as the internet has become so much a part of our lives?
TB: I would offer a lot of credit to Ed Snowden, who started this conversation. I think a lot of people, security professionals and people close to metal, kind of knew what was going on. Knew that, you know, government agencies were in fact scooping up a huge proportion of internet traffic. We did not know they were tapping into Google's data centre.

LV And you worked at Google for a while...
TB: Yeah, and I tell you, the people at Google were livid when that one came to the surface. So I think we owe Mr Snowden a lot of credit for exposing the larger population to the fact that their governments are doing things that are of questionable legality, morality and effectiveness.

LV In the UK – thanks to the European Court of Justice – it is illegal to harvest data from ISPs on a large scale and that's why they're rushing through these very quick surveillance laws to make it 'not' illegal.
TB: Right, the DRIP [Data Retention and Investigatory Powers] law. We're having a similar problem in Canada, although one of our courts found that it is illegal for ISPs to provide usage data without a warrant, which is a change in the landscape.

LV Do you think it's too late? Do you think the cat's out of the bag?
TB: Oh no, not in the slightest. We're still in the first generation of people who live on the internet. We can make the experience more private and safe for all the people that are going to come after us.

LV But even with all the data anonymised, there's no one accepting anyone's word that it's anonymised, and even when it is anonymised, you only need something like three trig points to find people. There's just so much data out there. It seems to us like the cat's already out the bag.
TB: Well sure, you can take that attitude. And also you can say that anyone who carries a mobile phone is revealing their location at all times. But I think the notion that the cat's out the bag and we should all roll over and give up is just barshit. I think that every time we increase the proportion of the traffic that is privacy-enabled, we are doing a public service. We are driving up the cost to those who have been abusing the existing transparency, and I think that's something we should do. We have no excuse for not doing it.

LV But we're in the minority. The new generation are used to Facebook and everything about their lives being online.
TB: From what I hear, the younger generation is increasingly less Facebook-centric and one of the reasons for that is they don't like that

creepy feeling that their information is leaking. Just because an opinion one holds is in the minority, doesn't mean we should give up and stop pursuing it. And, like I said earlier, I don't think we should expect the general population to have educated opinions about internet safety any more than they have educated opinions about emission control systems in automobiles or air traffic control regulations, and that kind of thing. It's our responsibility as professionals to ascertain what the most beneficial behaviour is and build things in that way.

> **"I'm not arguing in favour of absolute privacy; there can never be such a thing."**

LV It's one thing to say it, but we've known about this for a long time and it hasn't happened.

TB: I'm disturbed by your counsels of dispair. The proportion of things being transmitted over HTTPS is monotonically increasing. And we still encounter people saying 'oh, my stuff is shareware, it doesn't need to be HTTPS'. And to that I say, 'well, yeah, but the decision as to whether something needs to be private is complicated and very context dependent, and most people aren't equipped to make it, so why don't you just do the safe thing and make everything private. And that's an unanswerable argument now that HTTPS is very low cost and low difficulty to deploy. I'm quite optimistic actually. I'm not arguing in favour of absolute privacy; there can never be such a thing. All we can ever do is move the needle and increase the proportion of things that are privacy enabled, and I think we're doing that.

I notice that there are excellent *OpenPGP* libraries available now for Ruby and Python and JavaScript and Node. There are not for Java itself or for .Net, which is a problem that we need to address. I know that people are working hard over at Google and some other places about trying to do crypto in the browser. Which is, if you can do it, a super interesting enabler of a bunch of interesting things, but there's a severe question of trust. When can you trust an app that you download over the air? Those problems are not insuperable.

LV Do you think we can counteract the new app-centric frontier by using stuff like Firefox OS?

TB: It is clearly the case that the browser is becoming less central as part of the software delivery ecosystem because a very high proportion of things are migrating into native mobile apps. And to some extent that's happening for reasons that are actually good. Native mobile apps offer more responsiveness and a lot of UX options that are not available in browser apps. They also have an important advantage that we're not giving enough credit to, which is that they also have a superior developer experience. However, there are two important advantages of the browser-centric world that we're losing. One of course is the speed of update. If you have a bug that you need to fix fast, in the case of Android you're looking at a latency of hours and in the case of Apple you're looking at a totally unpredictable latency of potentially weeks, whereas if you're web-centric you can fix an applet in minutes. And that's huge. Secondly, the issue you raised is important to me. I think the internet is better when anyone can deploy anything on it without asking for permission. And anyone can go and get it. And I do not like the app store as intermediaries through which you go to find anything. I'm less offended by the Google app store than by Apple's because of its exclisivity – when Apple says we're not going to sell something, they're not saying they're not going to sell it, they're saying you can't have it. If Google doesn't like something, well you can still get it off the project's website, right? How much impact is something like Firefox OS gonna have? Well, I hate to say it, but that probably depends on price performance of the hardware it's running on.

Tim's a fan of the way apps make things easier for developers – but not of the disadvantages they present to users.

LV We lament the loss of what the internet had at the beginning of its life.

TB: Think about it this way, the volume of traffic on the internet is unimaginably vast. So whether you are an over enthusiastic government agency or whether you are a hacker trying to steal credit card numbers, every time we can impose a non-zero cost on surveillance, the volume is so high that there will be entire class of surveillance or an entire class of crime, that it becomes uneconomic. And, given that, why would we not do it? Every time you increase the proportion of traffic with HTTPS up by 5%, you drive a certain class of undesirable behaviours into uneconomic territory. You don't have to win to win, you just have to make things better to improve the environment.