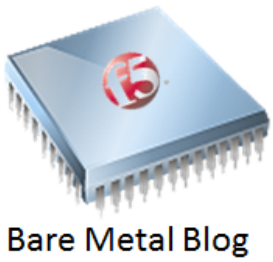
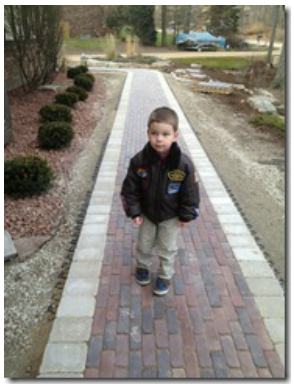
Bare Metal Blog: FPGAs The Benefits and Risks

Don MacVittie, 2012-20-11

#BareMetalBlog The use of FPGAs, the risks, trade-offs, and benefits to IT.



I was talking with the team working on our yard – they're putting in new sidewalks and a patio, amongst other things – and we got on the subject of gutters. When we bought this house, it came with no gutters, and that has, over time, caused some serious damage to the base of the house. Wood and plaster do not take it well when water pours down on them at the rate that, oh, say melting snow in the spring sends it down. So I had them get us an estimate for gutters on the entire house. Some of the work they're estimating is running the gutters right to the storm drain, which is not normally cheap, but they had both the front and back yards all ripped up, so it is a good time to do it, both cheaper and less messy, since the mess is already there.



So I told them to do it, because I don't want the sod they're going to lay to be ripped up in a year when we decide to put the gutters on, and certainly don't want them to rip up the patio and sidewalks they're putting in now just to lay pipe later – that would be nearly impossible.

And that, in a nutshell, is the same reason why FPGAs are used in a lot of high-tech firms. If the device is my yard/sidewalks, and I have to choose between a custom ASIC versus an FPGA, the custom ASIC would require me to rip up the yard later, while the FPGA is planning ahead for change.

Let me explain. With an FPGA, the circuits are programmed. Not like software, but code sets up the circuits, and then they are pretty equivalent to having them be hard-wired. With an ASIC, they really are hard-wired. So six months later, a change to the system – be it added functionality or fixes to existing logic – will be far easier with an FPGA than an ASIC. With an FPGA, the design file is opened, the changes made and tested, then the config is compiled and delivered to manufacturing. At that point, the devices produced with the new config file will have the new functionality. With ASICs, you change the design, send it to a manufacturing shop, wait for the shop to produce a small run (working it into their schedule that is), test the result, and then do a full production run. Then the new ASIC has to be put on the assembly line to replace the old ASIC. The difference is astronomical in terms of time required and even more so in terms of cost.

Of course there are some trade-offs. Every architectural choice results in trade-offs, and anyone who tells you differently is indeed trying to sell you something, and they don't want to admit the trade-offs used to produce what they're selling.

One of the big concerns out there about FPGAs is that they're less secure. In the most vague, general sense, this is true. But in practical use scenarios, it most certainly isn't. Here are the concerns, and why they're over-rated (note that these notes are adapted from responses to my questions put to Clint Harames of F5's most excellent FPGA team, I cannot vouch for other production except to say the other teams I was involved with outside of F5 were similar):

- It's field programmable! What if it gets modified? In F5's case, none of the programmability is accessible from the outside. There is no Ethernet or coding hack that can reprogram it, because that functionality is not accessible. Other vendors work to a differing standard, so definitely worth checking, though I would remind you that it is almost never going to be as easy to hack an FPGA as it is to hack software or COTS hardware.
- Okay, but can't it be erased and destroy the device? In theory yes (though erasing it is only effective until the next boot – non-destructive, so-to-speak), but if "modify" functionality is not accessible, then it can't be erased easily. The caveat is that there is of course a reset pin on the chip, but if the ne'er-do-well has physical access to your device, time to disassemble the device, and a handy pinout for the FPGA chip you're using, I'm going to guess you have bigger problems than whether they can reset your FPGA.
- If it's programmable, can't the program be read out and modified? Again, that functionality can be enabled on the chip, and you can check with your device manufacturer to see if they leave it enabled for production devices. Remember, it is a twofold story here, in F5's case, we don't generally want to reprogram production devices and don't want to make reverse engineering our product any easier than it has to be, while we want to protect you from someone modifying a production device. So when the design is done and meets all test criteria, we at F5 turn access to this functionality off completely before shipping product is produced. Definitely worth checking with your vendor to find out what they are doing.

Again, your vendor may do things differently, if, for some reason they need the ability to reprogram the FPGA in your device.

For you, the IT staffer, the benefits are pretty straight-forward. The device you purchase will be closer to “up to date” because of the time-to-market benefits of FPGAs, it will be cheaper because of the reduced up-front costs (note that like everything involving costs, economies of scale can change the “cheaper” part to be untrue, depending upon the costs involved), and the resulting device will be far, ***far*** faster than the equivalent processing done on a general purpose CPU. In the end, it is hardware doing the processing, and FPGAs have concurrency that general purpose CPUs can only match with a huge number of cores, even then since the OS handles the scheduling on a general purpose CPU, many cores does not normally make up the performance difference.

There are some who think the advent of virtualization and virtualized appliances should curb the use of FPGAs, as the virtual version has to include all the functionality. While this is, on the surface, a reasonable argument, it has a flaw. FPGAs are MUCH faster than software will ever be, let alone a VM running on a host with who-knows-how-many other VMs sharing its resources. So in cases like F5, where there is a hardware and a software version, the key is to be able to run in both. TMOS, F5’s OS for traffic management, uses hardware if available, software if not. This offers the best of both worlds – acceptable traffic management in a VM, and high-performance traffic management in hardware.

Next time I’ll delve into specific functionality that on our hardware platforms is implemented in FPGA, and how that helps you do your job in IT, today was more of a “what are the risks, what are the benefits” in a generic sense.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com