A panel discussion at the Electronics Design Show took a look at some of the issues involved in using FPGAs in embedded systems. By Graham Pitcher.

While there’s nothing new about the use of FPGAs in electronics products, many engineers are only just beginning to explore how the devices could help to improve their designs. 

There are a number of issues to be considered, including: how do you choose the right device for the job?; what are the tools issues?; what about design verification? and what’s the future for FPGAs?

Looking to address these questions, New Electronics assembled a panel at the recent Electronics Design Show conference, in association with NMI’s FPGA Frontrunners group.

FPGA design tips
So what are the panel’s tips for choosing an FPGA? Pete Leonard from Renishaw uses FPGAs widely. He said: “You could choose them for versatility, but you might need to look at one device from a particular family. But you have to remember that, at the end of the day, you have to produce something and sell it, so understand what the peripherals will interface with and what you will put into it.”

ARM’s Andrew Gardner says his group uses the largest possible device. “My group develops custom FPGA based hardware, so we have to put a lot of thought into device selection, finding a device with the right number of look up tables (LUTs), the right memory and interfaces.

“But it makes sense to put thought into the future. For example, we have upgraded our FPGA board so that we can use pin compatible devices in the future.”

Responding to a question from the floor, Gardner noted: “LUTs are cheap; it’s the interconnect that’s expensive and that can differentiate between FPGAs.”

Moderator Adam Taylor picked up on the LUT point, asking whether companies had capacity targets. “For example,” he asked, “do you want to fill the device or might you want 60% or 80% utilisation?”

AptCore’s Leon Wildman said utilisation involved a number of trade offs. “We look to see if the design is memory, pin, logic or gate bound. By changing the architecture, you can trade these off. For example, you can move memory off chip or you could use a serial interface, rather than a parallel one, if you’re pin bound. And you can always do more in software to get your design into a smaller part.”

But he counselled that this approach might take longer. “If time to market is important, you may want to spend more on a larger device.”

But, as a contributor from the floor noted: “When a device is full, you get a lot of timing problems. When you move something, that causes problems.”

Leonard noted that, in some designs, he had seen 96% device utilisation.

What about design tools and design approaches? “I’ve been exploring the use of Xilinx’ SDSoC environment, which allows you to use a C/C++ approach to programming Zynq devices,” said Taylor. This high level synthesis approach provides a way to generate RTL from C. “The FPGA world is pushing towards high level synthesis because there are more software engineers in the world.”

John Older, business development manager with ITDev, asked from the floor: “Can you get the best from an FPGA if you’re programming this way? Can you get the performance?”

Gardner said that if you’re designing for the web, you don’t use C. “We want to make efficient use of
verification in simulation because it’s easier, even if it means you get into the lab later.”

Continuing, Wildman said: “Use SystemVerilog for verification and develop coverage driven verification testbenches. Use constrained random testing at the block and system level, randomising things like packet length and headers. Collect coverage on corner cases; the things that might make your design trip over. Randomised testing can expose bugs you hadn’t thought about.”

How long should you spend verifying your design? Leonard said that, from his experience, 70 to 80% of design time is taken up proving what you’ve put in place works. “We design in RTL,” he noted, “designing and verifying in simulation. It allows us to check timings and the relationships between signals. “The beauty is that we can synthesise to an FPGA target, then get a back annotated model with timing delays. We can bring the propagation delays through the FPGA back into a behavioural testbench and check the timings are correct.”

Gardner said that working at the gate level allows designers to verify the synthesised output. “We will spend a lot of time hand crafting, maybe in RTL simulation. We have to trust that the synthesis tool will make an accurate job of turning the design into gates.”

Wildman added: “Design for test, manufacture and verification should be considered at the beginning. If you have four modes and four features, that means there are 16 functions to verify. Five and five means 25. Keep your design as simple as possible to reduce the number of corner cases. “Design from the spec, then write your verification plan from the spec. And it’s better to write the verification plan first.”

Third party IP And what about using third party IP? Taylor said he had used third party IP and integrated it successfully. “It does take a bit of understanding, but there’s no reason why you shouldn’t use it.”

Gardner advised: “Understand what you’re implementing and what rights you have to that IP. Be careful.”

Leonard said: “Have a good legal team which can review restrictions on use. But it’s important that engineers are aware of their legal obligations. You do need a definition of the interface, the functionality and performance, as well as an implementation, so you know how large the synthesised design is on a particular device.”

Wildman: “There is a cost and you need to understand it. It might be free on a trial basis, but not in the final product.”

“There’s copyright,” Taylor concluded, “but also copyleft when it comes to open source software. It’s an interesting minefield.”

hardware, so a high level approach is probably more appropriate.”

Wildman noted: “VHDL is massively parallel and, just like hardware, everything can happen at the same time. C, by contrast, is serial. It needs a different mindset to program in parallel.”

Renishaw’s Leonard said high level synthesis had been tried in research. “But our focus is on functionality,” he said. “If you don’t have much real estate, you have to make sure each register counts. High level synthesis might work if you have 1million gates, but will the device perform as you want it to? That needs some engineering ability.”

Advice for those coming into the FPGA world? Wildman said: “It’s easy for someone to ‘knock up’ some RTL, put it into an FPGA and then find out it doesn’t work when it gets to the lab. Don’t debug in the lab; do verification in simulation because it’s easier, even if it means you get into the lab later.”

Continuing, Wildman said: “Use SystemVerilog for verification and develop coverage driven verification testbenches. Use constrained random testing at the block and system level, randomising things like packet length and headers. Collect coverage on corner cases; the things that might make your design trip over. Randomised testing can expose bugs you hadn’t thought about.”

“High level synthesis might work if you have 1million gates, but will the device perform as you want it to? That needs some engineering ability.” Pete Leonard

www.newelectronics.co.uk 13 December 2016