[Music]

Paul Cutler
Welcome to the Circuit Python Show. I'm your host, Paul Cutler. This episode I'm joined by Seth Kerr. Seth is an embedded software engineer and open source hardware developer. Seth has designed the Icy Blue Feather and FPGA development board for mobile and power conscious applications. Seth, welcome to the show!

Seth Kerr
Hey, it's good to be here. I'm really excited to come on.

Paul Cutler
I'm glad you're here too. How did you first get started with computers and electronics?

Seth Kerr
My origin story with electronics is a lot like everybody else's. You know, parents had a computer, you're really excited about it. You know, growing up in the 90s and then early 2000s when the internet was, you know, coming full force, I mostly got started with the doing IRC chat bots back in the IRC days. And I sort of put it down for a little bit, but then a couple years ago, I was like, "Oh, I gotta really do something with my life." So I decided to go back to school for computer engineering and took my electronics class. And I was hooked. I started building transistor adders and stuff on breadboards and one thing led to another. And I'm here designing custom circuit boards and trying to make things a lot easier for people who started in not too different of a place that I did.

Paul Cutler
Speaking of custom circuit boards, you have a crowd supply campaign running for the Icy Blue FPGA feather. Let's start at the beginning. For listeners like me, what is an FPGA?

Seth Kerr
An FPGA is a field programmable gate array. So the easiest way I tend to think of it as if you put transistors in a circuit board, it's like that, but you're doing it with words. So FPGAs, they use a language model called hardware definition languages. And so what those do is you can describe functionality. So if I want to add two numbers together I can describe that process of adding it together and these tools that we use for programming FPGAs they will synthesize. So take what you have described and turn it into an actual circuit that's programmed onto these chips to do exactly, or in most cases exactly, what you describe to do. What are the advantages to using an FPGA over a typical microcontroller? So an FPGA is sort of like a blank slate compared to a microcontroller. So a microcontroller you kind of expect to have everything that you need to make your project go, you just need to code it. whereas in FPGA it gives you the ability to customize your approach. If you want to do something specific and really fast, an FPGA would be a perfect tool to allow you to do that. It doesn't have to worry about, you know, am I waiting on other things to finish up to be able to do what I need to do. It can just do it. Compared to a microcontroller, it's not as

Paul Cutler
like general purpose. It's very specific. What is your process to designing a board? Do you use KiCad or EagleCad? For the longest time I've used EagleCad and that's where I cut my teeth in school

Seth Kerr
because I've got a free student license and I figured out everything that I needed to with that. But I'm slowly working on transitioning to KiCad because you know things cost money for for Fusion and Eagle. And Eagle is not being regularly updated because they're trying to push

everybody to Fusion 360 Electronics Designer. But, you know, KiCad offers just like a great open source and very, you know, progressively improved platform. In that process, you know, I usually think about what I want to build. So like, if I'm, for instance, if I wanted to make a board here actually that I can share. So if I wanted to make this Bluetooth board, for instance, I think about, "Okay, what chip am I using? What are some constraints that I have? What are features or things that I want to include? With that, what number of PCB layers do I need to make that design function properly?" And so from there, I go through and I pick parts that are going to fit those needs. And then I start by just going through the data sheets and laying out the connections for those individual parts. And then, you know, going back to the main microcontroller or whatever chip is going to be the host on the board. And I get those things all connected up and then go into the PCB design. I figure out the design I want. So like, in case of like a feather. So like I have a feather, a pre-designed layout for feather footprints. I don't ever have to worry about the measurements or anything or where headers or, or USB are going to go. And then I can just lay my parts out, how they're best going to route. And then I, I route them up, figure out power scheme. Uh, and then, then I'm done. Send it off to fab.

Paul Cutler
So now that we know a little about what an FPGA is, tell me more about the icy blue FPGA feather.

Seth Kerr
So the IceBlue FPGA Feather is my attempt to bring a very prolific and very easy to use format to the FPGA world. And it was kind of already done. There's an Orange Crab FPGA Feather. It's a fantastic feather. But the various Gs that can be maybe a little bit intimidating thinking about having to program it, or like, oh, you've got DDR3 memory on that. It's a really cool board, but it's not quite at that beginner novice level that someone would hope to get into. And so I decided that I was going to go the traditional approach, use an FT232 USB bridge, which is from FTDI. And that allows us to have only one mode to program. So there's some spy flash on there that you can load your bitstream onto through this USB chip. And it sort of removes the guesswork of, oh, well, am I programming the FPGA directly? Am I programming Flash? And so it leaves out a whole chunk of things that might be difficult to understand when getting into FPGAs. So the first thing is removing some of these barriers that make it confusing. And then the second idea was, using this feather format, we can make it part of an ecosystem. So if people are very familiar with Adafruit, they know that there's maybe 10 dozen feather wings out there that can be used with like I squared C or spy interfaces or just regular digital GPIO. So what that does is just it creates a sort of reward where it's like if you figure out what you're doing and you process, work through the examples that you would have learning how to use an FPGA. You'd be like, OK, well, I know how to set up an I2C interface now. I want to use this feather wing that has a display on it. I want to figure out how to drive a display really fast with an FPGA. So there's a door open for you for a project. Or in the 1-bit squared Discord, somebody had mentioned they had made a keyboard with an FPGA, which is really novel. You wouldn't think you're like, Oh, I'll just use a microcontroller, but microcontrollers don't inherently have matrix scanning as something that they're good at. But in FPGA, it's all real-time IO. It's very fast. So you can do matrix scanning faster and more accurately than you could on a microcontroller. So they made an FPGA powered keyboard and I was like, Oh wow, that's really cool. So thinking about what are the things that I can do, this feather opens it up because of the ecosystem that it's part of. And so that's like these driving motivations as I've been working on it and refining it, is all these cool projects that people are going to be able to do simply because of the format that it's in and the process for programming it is so much simpler. There's no guesswork. There's only one way to do it. That's sort of what it is, what the motivation is behind it. It also has some features on it that are what you can expect for something that's geared towards people trying to make something or use something for the first time. It's got some LEDs on it, and it's got the traditional RGB LED because everybody loves RGB. I've got tons of examples already ready for people to use on GitHub that are set up.

they've already been tested. And then as well, a bunch of project boards as well that will have all the code. Well, I say code, but it's kind of just like the HDL files. They'll all be ready for people to look at, to dissect, and hopefully inspire people to dig deeper and start digging into the Feather ecosystem.

Paul Cutler
- You kind of touched on it, but within the Feather ecosystem, how could you use the Icy Blue Feather with another board running CircuitPython, for example.

Seth Kerr
- So the example that I like to give for this is, so the FPGA that's on the IC Blue Feather, it has what are called DSP blocks, so digital signal processing blocks. And essentially what they are is they're these multiplier blocks that can multiply incredibly fast. And they've also got these accumulators, which essentially you multiply and accumulate. you can run some very fast integer-based digital signal processing. And so one thing that most microcontrollers are not good at is digital signal processing. Unless they have specific hardware for that, like hardware floating point units, it may be a little bit slower to do it on a microcontroller. So one thing that people could do, interfacing the icy blue feather and another feather-- So like say the Feather RP2040, for instance. And this is a really good example because the RP2040 has two 8-bit PIO registers on it. And those can send data single clock cycle and receive data single clock cycle, which is essentially like, it's almost like having an FPGA on a microcontroller. But essentially what this allows you to do is you could take eight pins on your Feather, set eight pins on the IC blue, and send data back and forth incredibly fast, do your DSP. So like if you're trying to run a very, like a FFT, Fast Fourier Transform, you don't have to worry about doing that on your RP2040. You can continue collecting your data, sending it on, and not have to worry about wasting clock cycles, you know, doing your FFT. You can offload that to your FPGA. And that's a huge advantage that you would have with using the IcyBlue within the Feather ecosystem is the ability to do these tasks very quickly off chip without having to worry about, "Oh, am I squeezing the most resources out of my microcontroller?"

Paul Cutler
You touched on HDL earlier. For those used to programming with CircuitPython, how different is it to program the IcyBlue Feather?

Seth Kerr
It can be a little bit of a learning curve just because you're going from a mindset of telling hardware that's already pre-configured how you want it to run. So it's already converted into instructions that you don't have to worry about. With HDLs, you're telling it how exactly it's going to run, what you're doing with the data that comes in and out, what you're doing with the data that's interlinked and shared between described modules. In terms of the environments that you use and whatnot, you can use VS Code to write your VHDL or your Verilog code. The examples that I have set up in the GitHub repository, they are all Verilog, which I tend to like Verilog a little bit more because it's not really a good comparison, but it's considered closer to C than VHDL is. But I tend to find it a little bit easier to work with. It tends to give you this feeling that it's a little bit more understandable. It's not as-- like, the language is a little bit easier, the words that they use to describe what you're doing. But with that in mind, you can-- like I said, you can code it with VS Code. And then there's makefiles that are very easy to understand what's going on. If you want to set up a new project, you essentially just copy the makefile from another project. And then you just make sure that your file name and the module name in your top level file match inside the makefile. And then you just do make build and then make program flash. Some of them have a slightly different make command just because I maybe have multiple examples in the directory. But for the most part, what happens inside those commands is there's not really anything-- not much that needs to change. Really trying to make sure that the barriers to starting out are as low as possible.

Paul Cutler
Well, we're almost out of time, but before we go, I'd like to ask each guest one last question. You're about to start a new project. Which board do you reach for?

Seth Kerr
I'm kind of biased towards my own stuff, but one of my favorite boards to use is the BREAD 2040, which is a USB-C RB 2040 board that I made that is like slightly smaller than a feather. It has a NeoPixel and everything on it. It's just a cute little board. It's very BREAD board friendly and it just works for a lot of stuff that I'm doing. If people want to learn more about you or the Icy Blue Feather, where should they go? So the icy blue feather currently is in pre-launch on Crowd Supply. So if you go to Crowd Supply and go to the pre-launch page, you'll see the icy blue there. I also have a website where I'm trying to get back into more blogging and whatnot around the company. So that's oakdevtech.

Paul Cutler
I'll make sure to link to that in the show notes as well. Seth thanks so much for being on the show.

Seth Kerr
Thanks for having me. Appreciate it.

Paul Cutler
Thank you for listening to The Circuit Python Show. For show notes and transcripts, visit circuitpythonshow.com. Until next time, stay positive.

Subscribe!