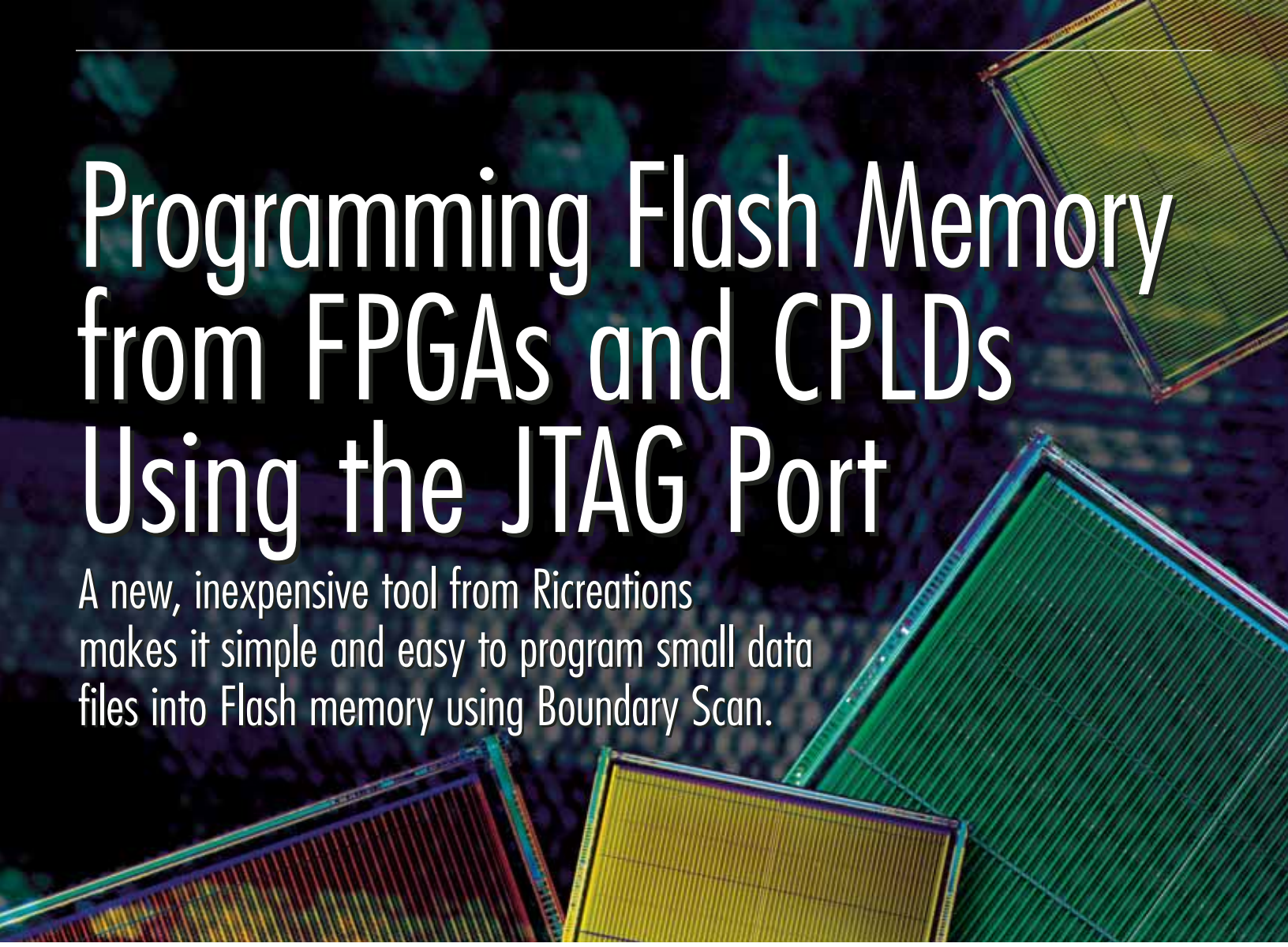

Programming Flash Memory from FPGAs and CPLDs Using the JTAG Port

A new, inexpensive tool from Ricreations makes it simple and easy to program small data files into Flash memory using Boundary Scan.

by Rick Folea
Job Title????????
Company????????
rfolea@UniversalScan.com

The first prototype of a processor board with Flash memory on it always poses a bit of a problem: How do you get the first chunk of code/boot loader/RTOS into the PROM? You could pre-program the PROM before populating the board, but that assumes the code is ready in time and won't require any changes.

Most designers and lab technicians don't have access to or can't afford the high-end JTAG tools available today. Furthermore, they usually don't want to take the time to build the tests required to do the scan testing and Flash programming anyway. So, what do you do?

We have added a new tool has been added to the Universal Scan™ our JTAG test suite that makes Flash programming from your Xilinx FPGA or CPLD a snap. You just tell the Universal Scan tool which Xilinx pins are connected to the PROM, select the data file to put in the PROM, and then press PROGRAM.

That's it. What's more, the Universal Scan tool is compatible with your Xilinx parallel port download cable, so you don't even need special hardware to do it.

JTAG Background: How Does it Work?

TTThe I/Os on all Xilinx FPGAs and CPLDs are connected to a giant shift register around the boundary of the device. From the JTAG port, you can shift test vectors into this boundary register using the TDI pin and then apply those vectors to the I/Os, independent of the logic inside the part. In fact, the part doesn't even have to be configured for this to work.

You can also unobtrusively monitor the I/O cells while your device is running by instructing the Boundary Scan chain to capture the state of the I/O cells, and then shift the result out on the TDO pin.

Simply stated, you would follow these steps to program your PROM:

1. Shift a vector into the JTAG chain to setup the address, data, and chip-enables (CEs) and apply it to the pins.
2. Shift the same vector into the JTAG chain to enable the write-enable (WE) signal to the PROM and apply it to the pins.
3. Shift the same vector into the JTAG chain with WE disabled and apply that to the pins.

Repeat these steps a few million times, throw in an occasional command or two to the PROM, and you're done. Sounds easy,

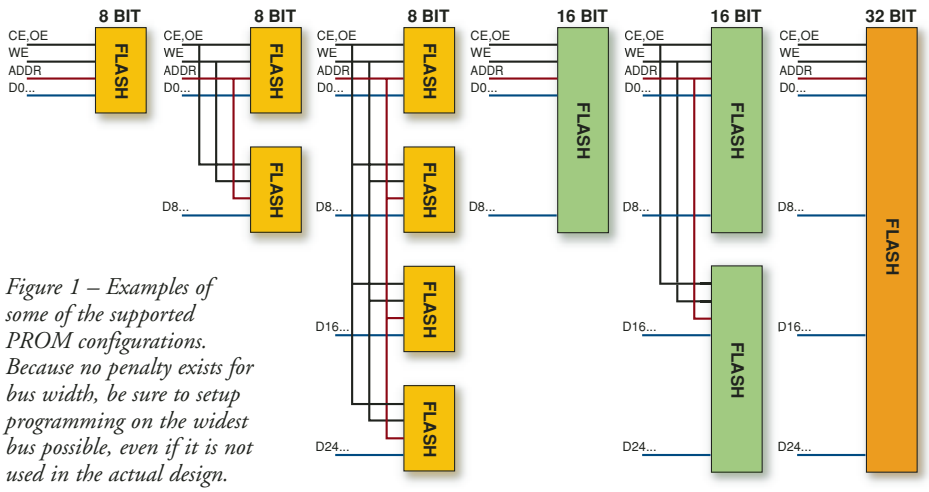

Figure 1 – Examples of some of the supported PROM configurations. Because no penalty exists for bus width, be sure to setup programming on the widest bus possible, even if it is not used in the actual design.

right? Unfortunately dealing with the low-level details of the JTAG state machine is tedious and difficult.

Fortunately, Universal Scan takes care of all of these details for you, and knocks the PROM programming effort down to the absolute simplest model possible. You don't need any netlists, test executives, test vectors, special hardware, or anything else normally associated with JTAG test development.

A Simple Solution

Universal Scan supports any bus configuration: 8-, 16-, and 32-bit PROM data buses built from 8-, 16-, or 32-bit PROMS. For example, you can have a 32-bit bus that comprises four 8-bit PROMS in parallel, and Universal Scan will program all four in parallel.

Figure 1 shows example PROM configurations supported by Universal Scan. Universal Scan also supports direct connections between the Xilinx part and the PROM enables, or indirect enables via memory-mapped I/O. (Perhaps your PROM CEs are derived from address lines in a PLD that is not in the JTAG chain, as shown in Figure 2.)

As Figure 2 also illustrates, PROM signals don't have to come from a single device; they can be spread out among any of the devices in the chain.

Limitations and Design Considerations

All this shifting of data around the JTAG chain mentioned previously is very time consuming. To demonstrate this point, let's take a simple example of a Xilinx Spartan-IIE™ device in a 456-pin FBGA package (XC2S300E-FG456). Assume all of the PROM pins are connected directly to this part.

This device has roughly 1,200 Boundary Scan cells in the giant shift register around the boundary of the device. If we take the worst-case scenario of writing one byte at a time to the PROM (no buffered writes), then we need to:

1. Shift the 1,200 bits into the chain to setup data, address, and CEs.
2. Shift the 1,200 bits into the chain to enable the WE signal.
3. Shift the 1,200 bits into the chain to disable the WE signal.

If we use an Intel® algorithm for writing a single byte, the example must be preceded with a command, which doubles the overhead. Thus, a total of 7,200 bits must shift around the JTAG chain just to write one byte/word, as shown in Figure 3. And that doesn't include adding a command to check the results of the operation.

Now, if we assume we have a small 20 KB boot loader we want to put in the Flash memory, then we'd need to repeat this 7,200 bit shift operation 20,000 times.

Because you typically get only a few hundred kilohertz bit rate out of a standard parallel port, that little 20 KB chunk of data takes about 10 minutes to program. If you happen to have a larger FPGA or a larger data file, it will take even longer. It all depends on the total length of the JTAG chain and the size of the data file.

Because all of the data is shifted in serially and then applied to a giant latch in parallel, there is no penalty for bus width. An 8-bit data bus programs at the same rate as a 16- or 32-bit bus. So if you are using a PROM that supports an 8- or 16-bit wide data bus as an 8-bit device, go ahead and connect the unused data lines and the BYTE control line to the FPGA. Even though they aren't used in the final design, you can use them to program the PROM through JTAG and cut the programming time in half. This works with 16- and 32-bit devices as well.

Other ways you can minimize programming time include:

- Connecting the PROM to the smallest JTAG device you can (the one with the shortest boundary register)

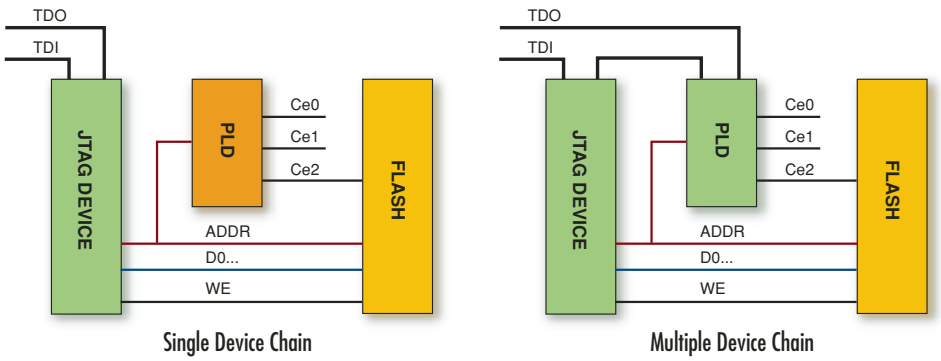

Figure 2 – Although it simplifies things to have all PROM pins connected to a single JTAG part, it is not a requirement. With Universal Scan you can program both memory-mapped PROMs and PROMs with signals from multiple JTAG devices.

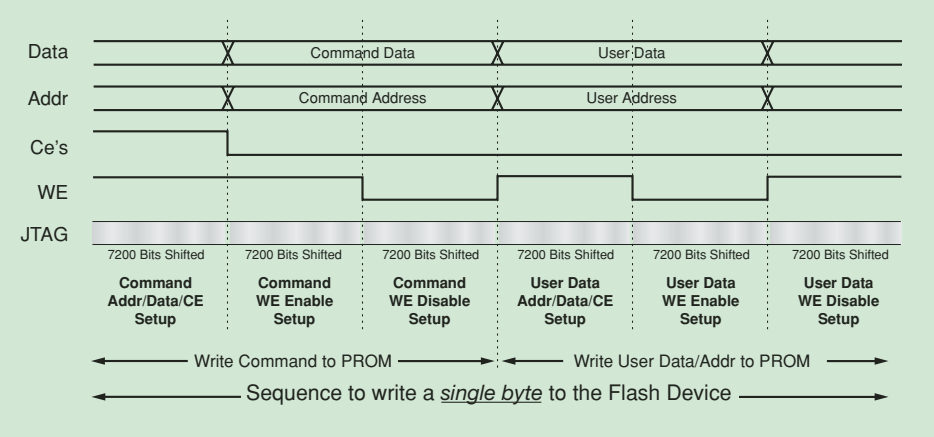

Figure 3 – Programming Flash memories is slow because each and every bus transition requires shifting the data through the entire JTAG chain. This shows the un-optimized example described in the text.

Check out these Xilinx approved suppliers if you are interested in full high-end or high-speed JTAG testing tools:

• JTAG TTechnologies	*www.jtag.com*
• Acculogic	*www.acculogic.com*
• Corelis	*www.corelis.com*
• Goepel	*www.goepel.com*
• Assett-Intertech	*www.assett-intertech.com*
• Intellitech	*www.intellitech.com*
• Flynn	*www.flynn.com*

- Putting any JTAG parts not connected to the PROM into BYPASS mode to shorten the overall length of the JTAG chain
- Trying to connect all PROM signals to only one of the devices in the JTAG chain (maximizes the number of devices you can put into BYPASS)
- Choosing a Flash memory that supports buffered writes – these don't require that you write a command before every byte/word write-cycle and nearly doubles the throughput rate of the programming operation

Also, be sure to connect all PROM signals to the JTAG chain so that you can control every aspect of the PROM's functionality through Boundary Scan – and don't forget to connect the VPEN signal to a pin under JTAG control.

Although this article focuses on Xilinx FPGAs and CPLDs, this method will work exactly the same way with any JTAG-enabled device: processors, DSPs, Ethernet switches, microcontrollers, and others.

If things don't go according to plan, it's easy to debug any issues you might be having with the JTAG chain or Flash programming, as the Flash programmer is part of the Universal Scan JTAG debugging tool. You can use Universal Scan to manually toggle signals between the PROM and your Xilinx device to isolate the issue quickly and efficiently.

Conclusion

The Universal Scan tool enables you to easily program small data files into Flash memory using Boundary Scan. Universal Scan does not replace high-end JTAG tools, which are great if you need to program large data files quickly or in large quantities. But if you want an inexpensive, simple, and flexible JTAG Flash memory programming tool for prototype and general lab development using small data files, then Universal Scan may be the perfect solution.

Universal Scan is available now as a free upgrade to Universal Scan 6.0 users with active registrations. A free, fully functional trial is also available on the Web at www.UniversalScan.com. Download it and start programming Flash from your Xilinx devices today.

You'll find more information on this tool on the Xilinx website at *www.xilinx.com*, under Products & Services > System Resources > Configuration Solutions > Automatic Test Equipment (ATE) and Boundary Scan Tools. You can also call your local Xilinx distributor for information or arrange a live demo at your business. Σ

For more information about Boundary Scan, please consult these resources:

Intel *(www.intel.com)*

Intel, "Designing for On-Board Programming Using the IEEE 1149.1 (JTAG) Access Port" Application Note #AP-630.

Intel, "Introduction to On-Board Programming with Intel Flash Memory" Application Note #AP-624.

Xilinx *(www.xilinx.com)*

Folea, Rick, "Got the BGA Blues?" Xcell Journal – Issue 46, March 2003.

Xilinx, "A Quick JTAG ISP Checklist" Application Note XAPP104.

Xilinx, "Using BSDL Files for Spartan-3 FPGAs" Application Note XAPP476.

Xilinx, "Using the XC9500/XL/XV JTAG Boundary Scan Interface" Application Note XAPP069.

Amazon *(www.amazon.com)*

Parker, Kenneth. June 2003. The Boundary-Scan Handbook. Kluwer Academic Publishers; 3rd edition.