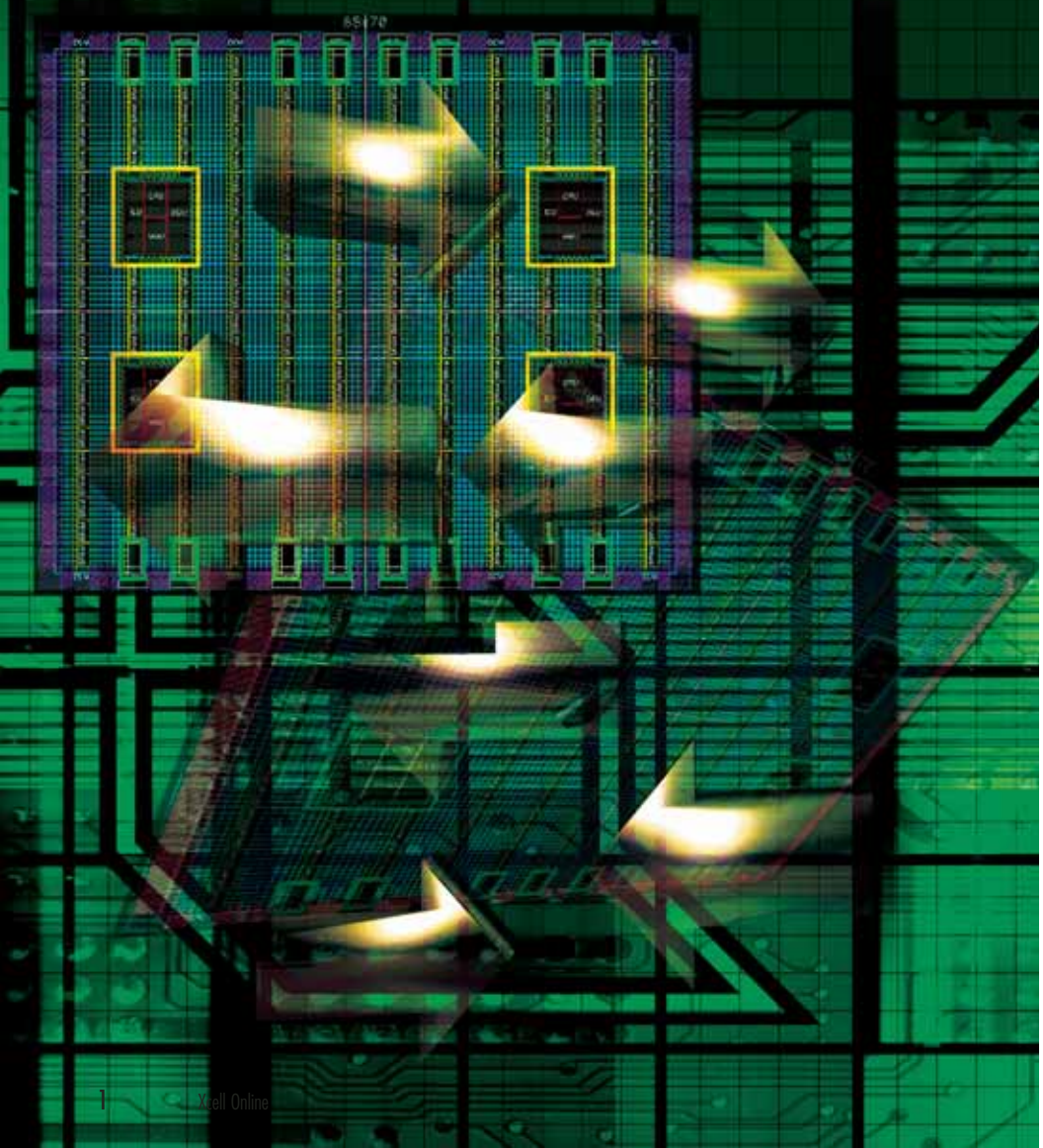
Using the JTAG Interface as a General-Purpose Communication Port



The existing JTAG programming interface in Xilinx FPGAs provides a flexible debug and diagnostic interface.

by Derek Wallace
Staff Engineer
Silicon and Software Systems
derek.wallace@s3group.com

You are probably quite familiar with using the JTAG interface for configuring Xilinx® FPGAs. You may also be familiar with the JTAG interface for various other functions, such as using the Xilinx ChipScope™ Pro tool for debugging your designs. However, have you ever considered using this same interface as a general-purpose communication port to your design?

You can use the JTAG interface of Virtex™-4, Virtex-II, Virtex-II Pro, and Spartan™-3 devices as a low-footprint general-purpose communication port, providing powerful and flexible test and debug capability to your design.

Using the JTAG interface has the following advantages:

- No additional connectors are required on the PCB, as the JTAG is used for both download and communication
- It does not require voltage-level translation (as RS232 would)
- The JTAG controller is already built into the FPGA, minimizing user design

JTAG Interface Overview

JTAG is a half-duplex serial communication interface. Any device that supports the JTAG interface will implement a JTAG controller. Data is shifted in through TDI and shifted out via TDO. The TAP controller is clocked by TCK and uses TMS as a control signal to control the overall process.

Communication with the JTAG controller is a two-step process. First, an instruction is shifted into the instruction register (IR). This instruction is decoded and selects one of the data registers (DR) for shifting data into (on TDI) and at the same time selecting the same DR as the source of data for TDO. The selected DR is then accessed. Accessing the DR always requires shifting N bits of data into the register on TDI and receiving N bits of data from the register on TDO (where N is the size of the register).

Each DR can be read only, write only, or read/write. If a DR is read only, the N bits that are shifted in during an access are discarded and the N bits shifted out are returned on the TDO line. In the case of a write-only data register, the N bits shifted in are captured in the register and the N bits shifted out are meaningless.

For more detailed information on using the JTAG controller in Xilinx FPGAs, see Xilinx application note XAPP139, "Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary Scan," or "Reconfiguring Block RAMs," a Xilinx TechXclusive.

Extending JTAG with Custom Instructions

The Xilinx JTAG controller supports a number of custom JTAG instructions and corresponding data registers. One such example is the CFG_IN instruction and its DR. This CFG_IN DR is used during the configuration of the FPGA over the JTAG interface.

USER Custom Instructions

Another example is the USER custom instructions. These instructions allow the JTAG controller to communicate with user logic inside the FPGA. Virtex-4 devices support up to four such instructions: USER(4:1). Spartan-3, Virtex-II, and Virtex-II Pro FPGAs support two instructions.

Figure 1 is a block diagram illustrating how the JTAG controller connects to the FPGA fabric in a Xilinx FPGA. You should first instantiate the BSCAN component from the Xilinx UniSim libraries (specific components are available for Virtex-4, Virtex-II, and Spartan-3 devices). This component enables your design to communicate with the JTAG controller. You must then implement one of the USER DRs.

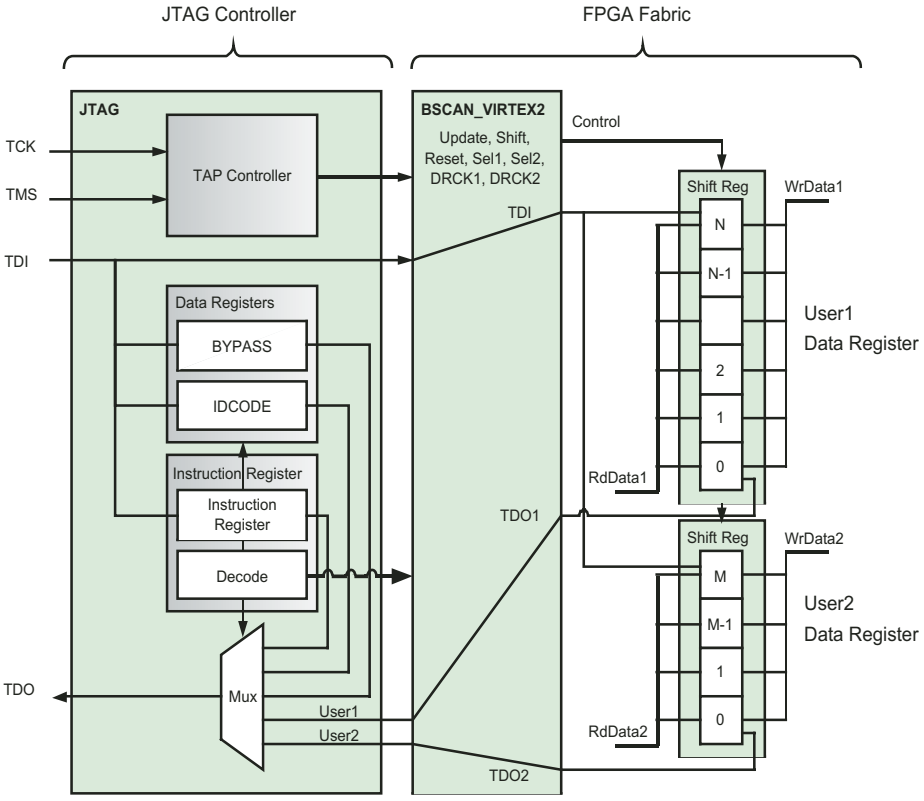


Figure 1 – JTAG interface to FPGA fabric

Implementing a USER DR

A USER DR is just a shift register with optional load. The BSCAN component provides all of the control signals needed to clock, enable, and load the shift register. A write-only USER DR requires a shift register, while a read/write USER DR requires a loadable shift register.

The size of the USER DR is up to you. It can be as long or as short as you require for your target application. Additionally, each USER DR may be a different size. Implementing a read/write 32-bit USER DR requires only 16 slices, a tiny fraction of the available resources in even the smallest Spartan-3 device. (A write-only 32-bit USER DR can be implemented in as little as six slices using SRLs.)

JTAG Communication Port

The complexity of modern FPGA designs, coupled with the commercial pressure for fast turnarounds, requires innovative methodologies for testing and debug. With these challenges in mind, S3 has developed a simple architecture that uses the existing JTAG interface to enable communication with your design. You can easily adapt this architecture, General-purpose Native jtAg Tester (GNAT), for any FPGA design, providing a very low footprint yet powerful communication port.

Flexible Architecture

In both laboratory and factory test envi-

ronments, we have found many varied uses for GNAT:

- Replacement for external components used during the development phase but removed during production (such as LEDs and switches)
- As a half-duplex communication channel to an embedded processor such as PicoBlaze™ or MicroBlaze™ soft-core processors
- Enabling test/diagnostic modes
- Reading status/debug registers from design
- Updating PicoBlaze instruction ROM while the FPGA is "live"
- Updating embedded ROM/RAM data coefficients while the FPGA is "live"
- Controlling the hardware remotely from a PC by emulating external switches (like the push-button reset)
- Controlling a multiplexer to select trigger/data signals that route to the ChipScope Pro tool
- Controlling a multiplexer to select signals that route to external pins for monitoring
- Programming/uploading external memories that interface to the FPGA, such as SDRAM

GNAT Architecture

Figure 2 is a block diagram of the GNAT architecture. In this example one of the USER DRs is being used by the Xilinx ChipScope Pro tool, while another is being used by the GNAT component.

The GNAT component is a configurable block that implements a variable-length DR. One side of the GNAT component communicates with the BSCAN_VIRTEX2 block while the other provides a very simple read/write bus interface to multiple GNAT peripherals. The GNAT component provides a unique Chip Select signal for each peripheral as well as read/write control signals and write data. A readable GNAT peripheral is responsible for providing read data.

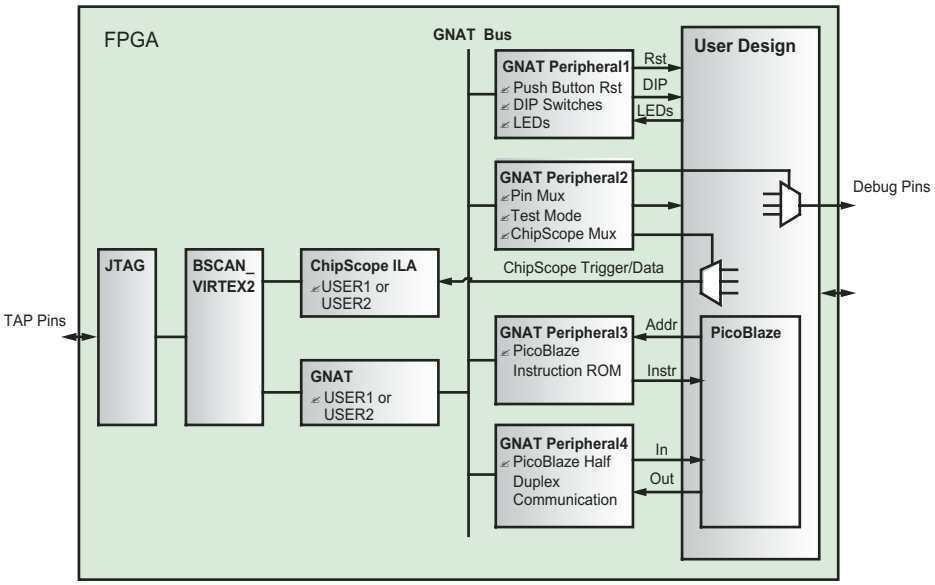


Figure 2 – GNAT architecture

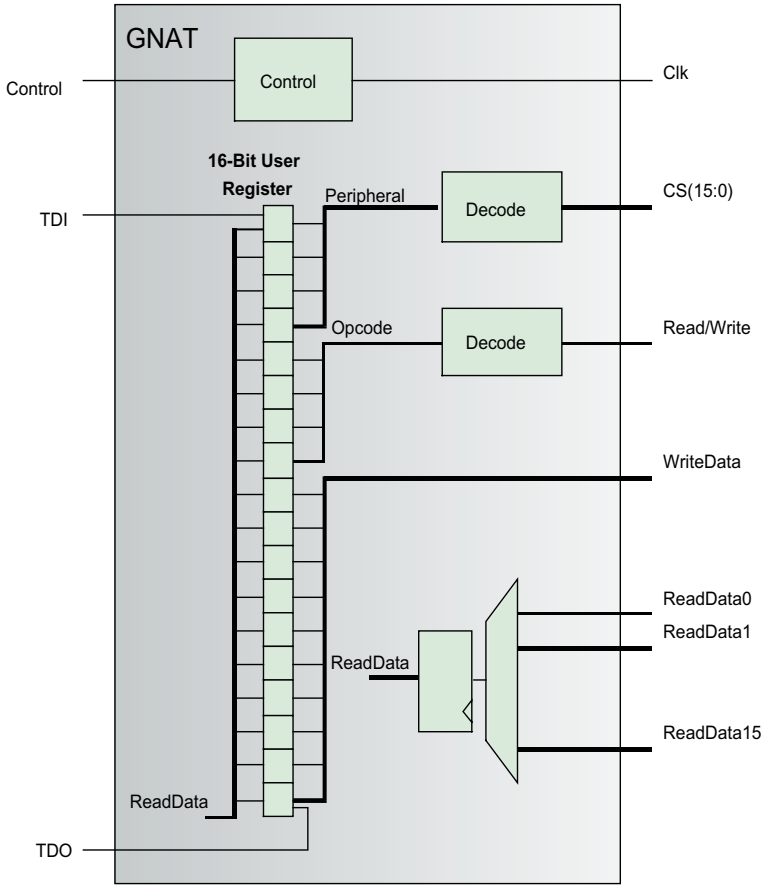


Figure 3 – GNAT component

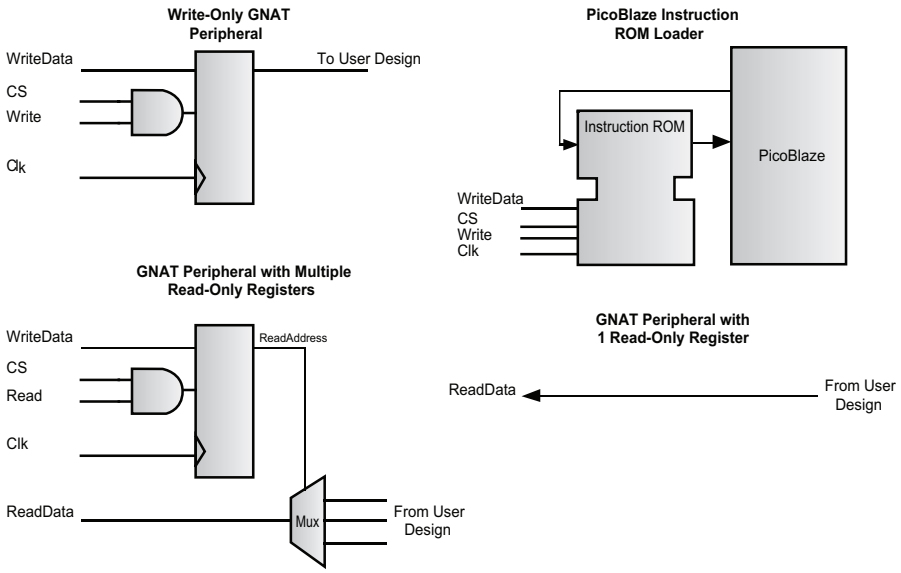


Figure 4 – GNAT peripherals

```
#============================================================
# JTAG Chain
# -----------------
#     Device0   Device1     Device2
# TDI –> 18V02 –> XCV50E –> XC2V4000 -> TDO
#
# NOTE: binary data is shifted in lsb first
#============================================================

source $env(XILINX)/chipscope/tcljtag.tcl
set handle    [jtag_open]
jtag_lock     $handle
jtag_autodetect  $handle

# Shift the USER2 Instruction (b00011) into the Instruction Register of XC2V4000
jtag_shiftir   $handle -buffer "110000" -endstate RTI -device 2

# WRITE: Shift the 16 bit GNAT command into the USER2 Data Register of XC2V4000
#    Peripheral: 0x2
#    Opcode  : 0x0 (write)
#    Data    : 0x34
jtag_shiftdr   $handle -buffer "0010110000000100" -endstate RTI -device 2

jtag_unlock  $handle
```

Figure 5 – Sample TCL to write to a GNAT peripheral

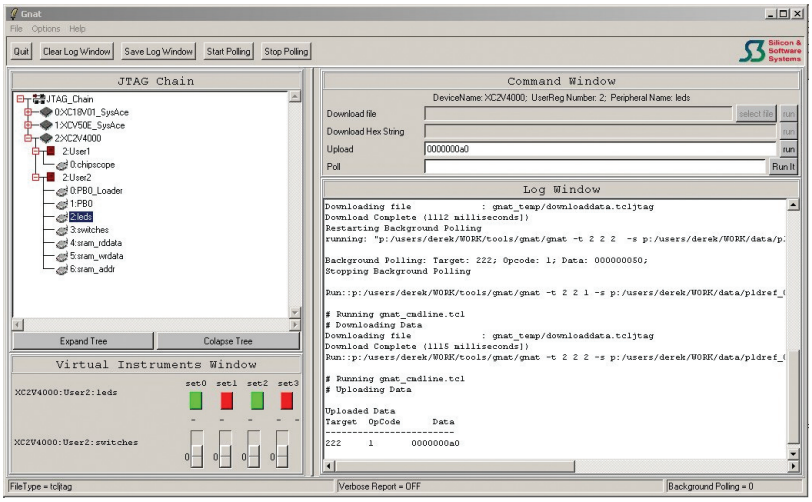


Figure 6 – GNAT GUI

GNAT Component

Figure 3 is a detailed view of the GNAT component. In this case a 16-bit USER DR has been defined. The upper 8 bits of the register are used to control access to the GNAT peripherals; 4 bits identify the peripheral and 4 bits are allocated as an opcode (read/write). The lower 8 bits are for data communication.

The combination of the BSCAN_VIRTEX2 and the GNAT blocks now provide a very simple but powerful communication mechanism to your design. And because it is all configurable, you can define the size of the USER DR and the definition of all bits that best meet your needs.

GNAT Peripheral

Figure 4 illustrates some example GNAT peripherals. The GNAT bus provides a very simple mechanism to read/write from or to a GNAT peripheral.

Software Support

Xilinx provides all of the tools required for you to write simple scripts to access the USER registers through the JTAG interface. ISE™ tools provide a custom TCL shell (xtclsh), while the Xilinx ChipScope Pro tools provide a TCL script (Figure 5) (tcljtag.tcl) that creates a number of high-level TCL commands to control the JTAG interface.

With these tools, it is easy to develop scripts or GUI applications to provide a powerful debug environment. Figure 5 is an example TCL script that writes the value 0x34 to a GNAT peripheral in device 2 of the JTAG chain. Figure 6 is an example GUI that we at S3 developed to access multiple GNAT peripherals in multiple devices.

Conclusion

This article has presented how the JTAG interface in Xilinx FPGAs can be used as a general-purpose communication port to your design. A simple lightweight, yet flexible architecture, GNAT, illustrates how you can quickly and easily add JTAG debug capability to your design with minimal FPGA resource overhead. For more information on GNAT, visit *www.s3group.com/design_expertise/fpga/*.