

# FPGA Embedded Test Speeds Development

FPGA embedded measurement cores and trace ports simplify your logic analyzer-based debugging and validation efforts.

by Brad Frieden  
Applications Development Engineer  
Agilent Technologies  
[brad\\_frieden@agilent.com](mailto:brad_frieden@agilent.com)

With the advent of general-purpose FPGA measurement cores, along with embedded microprocessor-focused measurement cores and trace ports, new options exist for system-level validation and debugging that can save significant time. There are some trade-offs given the choices available that can affect pin count, resources required, and the level of abstraction in captured data.

In this article, I'll consider these options and trade-offs and describe several validation and debugging situations.

## Basic Trade-Offs when Choosing Tools

One of the initial trade-offs concerns the number of FPGA pins (if any) that you can dedicate to the debugging and validation process. If you cannot dedicate pins to debugging, you will need to use the Xilinx® ChipScope™ Pro Integrated Logic Analyzer (ILA) to get internal visibility into the FPGA. The ChipScope Pro Analyzer uses an ILA core and block RAM for signal capture.



If you can spare some FPGA pins for debugging, a world of new options opens up, including a variety of measurement cores that bring signals out to pins probed with a logic analyzer or mixed-signal oscilloscope (Figure 1). Agilent Trace Cores (ATC2) are available for timing analysis, state analysis, and state analysis with pin compression.

### Example Applications

A number of applications can benefit from logic analyzer-based measurement cores and trace ports. These include:

- DDR2 memory interface debugging using a timing core
- System-level validation using a MicroBlaze™ trace core and external timing capture
- Program flow verification using an IBM PowerPC™ 405 trace port

### DDR2 Memory Interface Debugging

A recent challenge within an R&D group at Agilent Technologies involved turning on a DDR2 memory interface implemented with Xilinx Virtex™-4 FPGAs. We performed initial tests with the Xilinx ML403 development board. An ATC2 measurement timing core provided the means of tracing signals through the memory system, which led to a bug fix in the memory controller.

The digital system acquired and processed data, placing it first into DDR2 memory and then retrieving it from memory, on demand, for processing with the architecture shown in Figure 2. The memory ran at a 200-MHz clock rate. In the prototype design, something was corrupting data in the memory system.

### Our Debugging Approach

We employed a basic approach to validate and debug the memory system that first involved observing the control path, then the datapath, and then the interaction of the control and datapaths. There were a few key factors to make this approach successful:

- Access enough bits of the memory controller state machine to verify key write-and-read cycles of interest

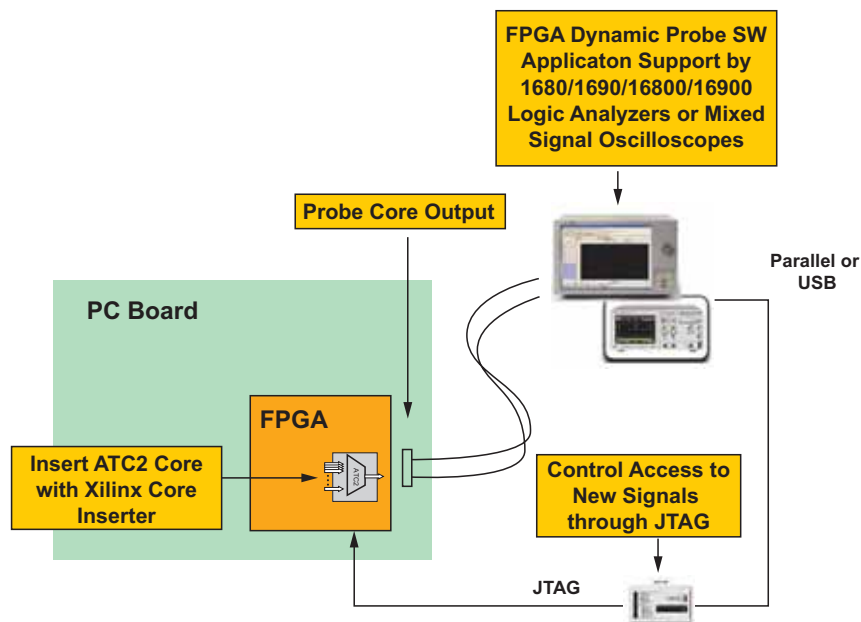


Figure 1 – FPGA Dynamic Probe measurement system with ATC2 core

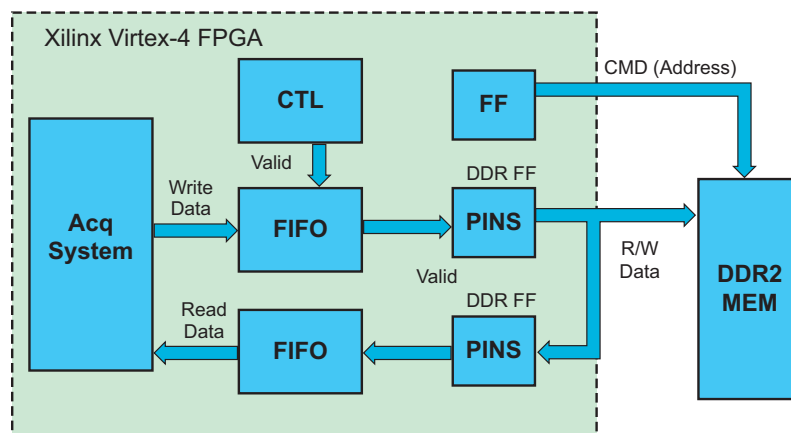


Figure 2 – Digital system with failing DDR2 memory operation

- Create a known stimulus and follow it through, including a read back
- Access at least a few bits of the data bus to trace data through
- Probe to view and ensure the rough timing between control and data signals

In this particular case, we had 10 pins available for debugging. To trace these signals, we also had to contend with several clock domains. An Agilent ATC2 Timing Core was a good choice in this application because a single timing core could probe multiple clock domains and

map those different clock domain signals out to a fixed set of pins.

One trade-off in using a timing core is that you must live with the channel-to-channel skew that forms because of the variation in path lengths to route internal FPGA signals out to FPGA pins. Agilent intentionally designed the timing core to treat those routes as “false paths” so as not to put a strain on the design tools and cause difficult timing constraints for a measurement. All place and route efforts can be focused on meeting timing constraints for the actual design.

## The Measurement System

A 16800 portable logic analyzer ran the Agilent FPGA Dynamic Probe measurement application shown in Figure 3. It was connected to the target system through JTAG for FPGA programming and MUX selection, and through flying lead probes for signal capture.

## Signal Bank Definition

We defined signal banks to check the basic memory controller state machine first and then trace signals as they flowed through the memory. Finally, the signal banks looked at both control and data to verify rough timing. We specifically defined the banks as:

Bank 0: DQaout (lowest 4 bits),  
DQbout (lowest 4 bits), Vld, DQS

Bank 1: DQain (lowest 4 bits), DQbin  
(lowest 4 bits), Vld, DQS

Bank 2: Control state machine

Bank 3: Command (address) at I/O ring

Bank 4: Write data entering FIFO

Bank 5: Write data at I/O ring

Bank 6: Read data at I/O ring

Bank 7: Read data at FIFO output

The basic state machine looked fine, as did the data written into memory. Even the data being read back from memory had no errors. The final bank measurement (Bank 0), which looked at both control and data, revealed an early strobe-to-data relationship, as shown in Figure 4. We modified the memory controller design by adjusting a register value, which in turn aligned Vld with DQS A and B data. That solved the data corruption problem.

We could have possibly considered a state core. However, each clock domain would have required its own state core, and each core its own set of FPGA pins.

## System-Level Validation

Tracing the execution of embedded processors has become more difficult in the last 10 years with the advent of caching and pipelining. However, with advances in trace technology, new embedded processors in FPGAs (such as the MicroBlaze

embedded processor in Xilinx FPGAs) have excellent visibility into their operation. This is possible through new measurement cores that probe behind the cache, combined with inverse assemblers running on a logic analyzer.

which an embedded MicroBlaze processor was part of a memory controller design. They probed both the MicroBlaze processor and external memory in an effort to validate the memory system operation. By probing the MicroBlaze processor with a

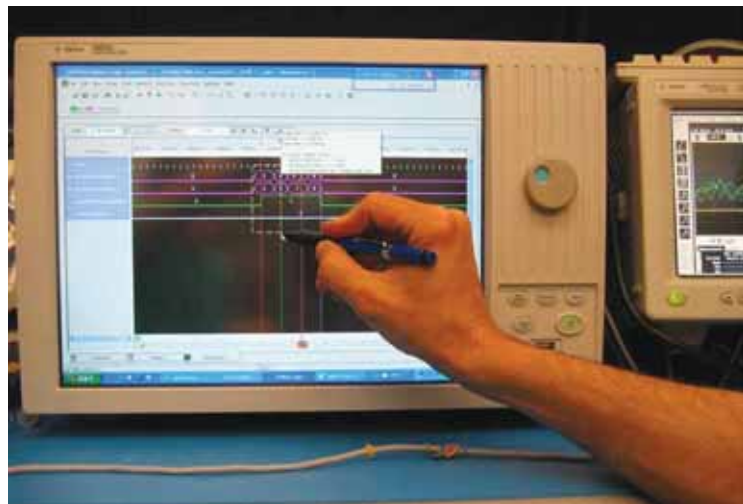


Figure 3 – Agilent 16800 series portable logic analyzer with FPGA Dynamic Probe



Figure 4 – Vld present one clock cycle early and misaligned to DQS A and B data

Additionally, with such a thorough connection to microprocessor signals, tight time correlation is possible between microprocessor activity and external events such as an FPGA interface to external memory. This helps immensely with system-level validation and debugging.

## A MicroBlaze Debugging Example

Designers in our Agilent R&D lab recently validated the operation of a system in

MicroBlaze trace core, they were able to trace program flow, including write and read commands from the MicroBlaze embedded processor. Timing measurements on the external memory helped determine whether they had accomplished proper memory interfacing.

The designers inserted the MicroBlaze trace core (MTC) into the design using Xilinx CORE Generator™ software from within the Embedded Development

Kit (EDK). In doing so, they had to make some trade-offs. If they allocated more FPGA pins, they could access more MicroBlaze signals, and vice versa. In the end, they chose to dedicate 20 FPGA pins to the measurement. That let them view key address, control, and data signals sufficient to do inverse assembly.

Next, one of the designers walked through steps to connect to the FPGA,

program it with an instrumented design, import bus/signal names, and perform automatic FPGA pin mapping. This allocation allowed him to not only view program flow, but also see data reads and writes to memory and registers. He accomplished this with the FPGA Dynamic Probe/MicroBlaze trace tool kit interface.

He was then able to perform pin mapping, in which the logic analyzer systemati-

cally toggles each of the FPGA debugging pins. This allows the logic analyzer to scan across all of its channels to find each toggling signal. The logic analyzer is then able to take advantage of pin-to-channel mapping. With this information, the logic analyzer automatically assigns MicroBlaze signal names to the proper logic analyzer channels. Figure 5 shows the resulting trace. The designer thus saved a great deal of setup time and eliminated the potential for errors that exist when using a manual process.

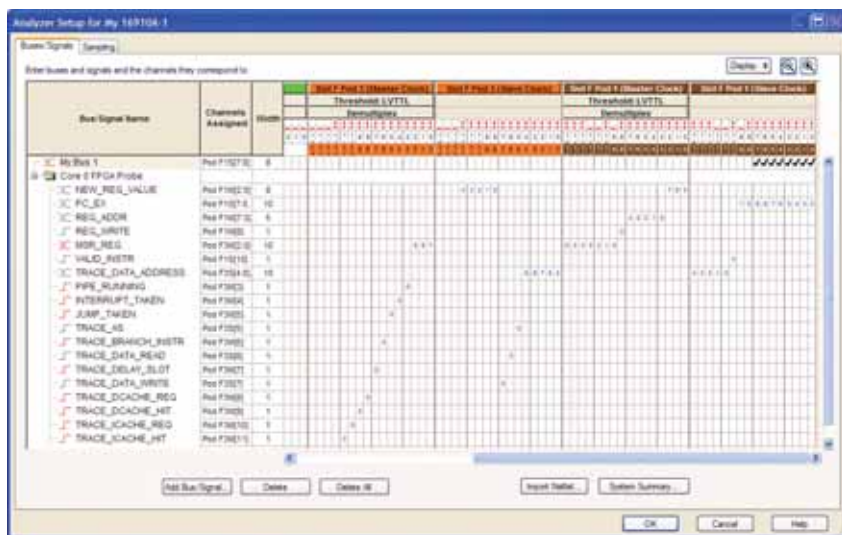


Figure 5 – Automatic logic analyzer pod and channel assignments for MicroBlaze trace

### Tying in External Memory Measurements

An interesting methodology ensued, as described in Figure 6, in which memory writes and reads were driven by a simple program running on the MicroBlaze processor, which ultimately led to hardware timing measurements on the memory. First, the logic analyzer inverse assembler traced program flow, including memory read and write commands. The logic analyzer could then search forward from a write or read command to find actual memory writes and reads in the hardware. Careful triggering on the external memory control lines (RAS, CAS, WE) allowed the logic analyzer to capture fine timing measurements between control lines and data signals (DQ, DQS, DDR\_Clock).

To trace the MicroBlaze execution as well as the external memory timing, the designer “split” the logic analyzer acquisition module into two independent analyzers, a unique ability of Agilent logic analyzers. One analyzer was tied to the MicroBlaze clock, while the other analyzer generated its own clock in timing mode.

He then attached high-performance flying lead probes to the DDR memory. With this connection, the logic analyzer could look for a control value of x04 (RAS=H, CAS=L, WE=L) to find a write and x05 to find a read.

### Tracing Memory Write at Multiple Levels

He first viewed a memory write at the assembly and source level, where the MicroBlaze signal TRACE\_DATA\_WRITE went high after a system reset. The program executed by the MicroBlaze processor wrote successive values in successive memory loca-

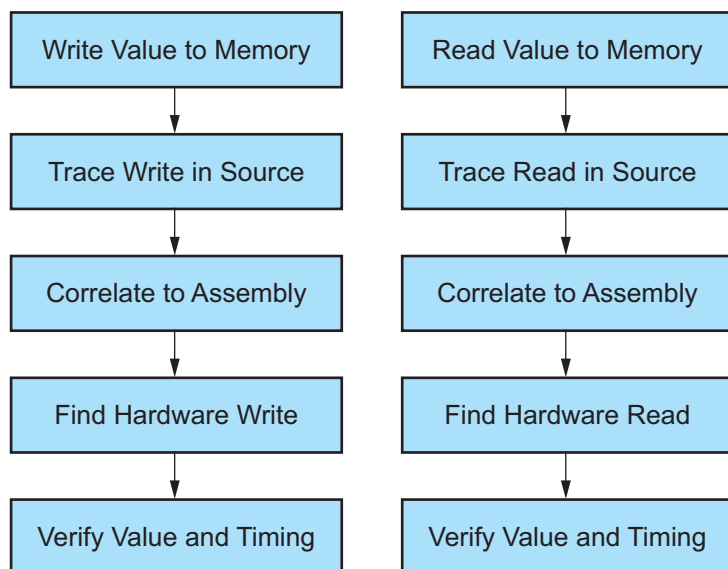


Figure 6 – Process for validation of proper writes and reads to and from DDR memory

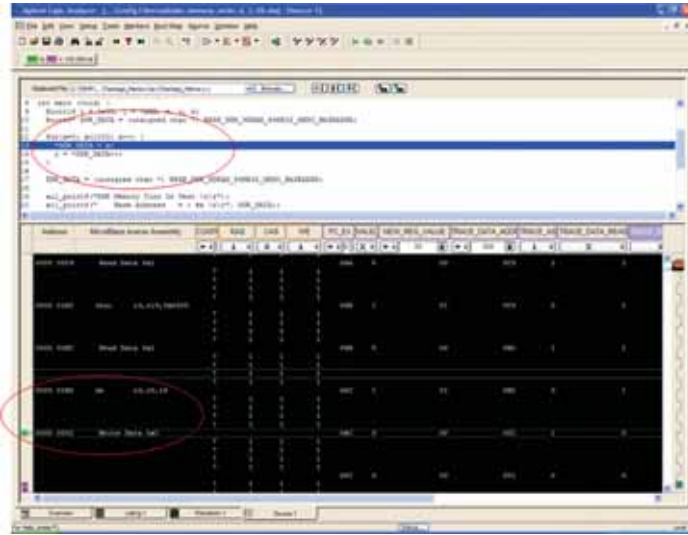


Figure 7 – MicroBlaze write command seen in program flow at assembly and source level



Figure 8 – 4-GHz timing analyzer memory write value and timing validation

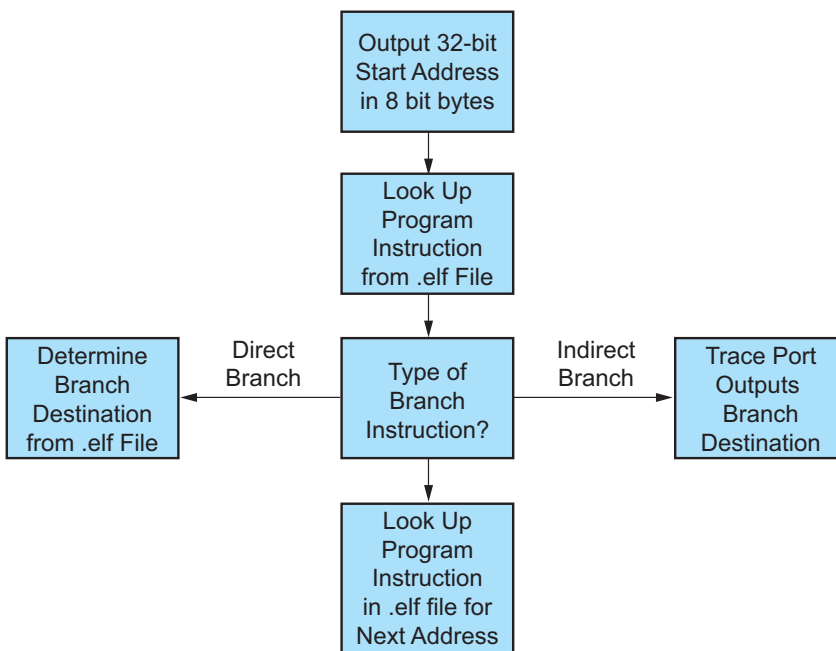


Figure 9 – Process for trace port decoding on embedded IBM 405 processor

tions. The designer turned the cache off to force the writes and reads into and out of the external memory. Figure 7 shows the program flow with the MicroBlaze processor after the designer stepped through the source code; he used the down green arrow in the logic analyzer interface until a memory write appeared.

At this point, he was able to make a search on the external memory controller line data for the next write condition (CONTROL = x04). The result from this search pointed the logic analyzer to precisely the right time to sort out memory timing and verify that it was correct. The trace of a memory write in hardware is shown in Figure 8 with specific write timing validation measurements. Notice that on the rising edge of DQS and CLK, the data bits were stable and a value of “1” was written to memory. A similar process verified that a memory write had correct data and timing.

### Program Flow Verification with IBM PPC 405

A surprisingly different scenario existed with a system that incorporated an embedded IBM 405 processor core, in which a different designer needed a basic trace to verify program flow. The system incorporated a trace port for the 405. This allowed tracing program flow while measuring only a few pins on the FPGA. In the previous example with the MicroBlaze embedded processor, control, address, data, and register information were all measured and accessible. Here, only partial control and address information was accessed through the trace port, so significant “reconstruction” of the trace was necessary.

### Making “Sense” Out of “Nonsense”

The designer took advantage of a logic analyzer trace reconstruction process (described in Figure 9) to determine program flow. Through a combination of information, a logic analyzer inverse assembler can reconstruct the trace. A number of things are required – knowing a starting address for the program being executed, having access to the source files and the executable (.elf) file, and obtaining occa-

sional branch trace messages. Those branch trace messages give clues about how program branches are taken. For example, if a direct branch is taken, then the inverse assembler must look into the .elf file to see where that branch leads.

There is a subtle difference between this 405 trace and that of the MicroBlaze processor. It is difficult to make a time correlation between the 405 program flow trace and any external trace, such as the trace from probing external memory. If the cache is turned on, then no guaranteed correlation exists between the reconstructed 405 trace and such external measurements. With the cache turned off, correlation is achieved to a similar level as with the MicroBlaze embedded processor.


### Conclusion

Measurement cores used in conjunction with logic analyzers can be very helpful to validate and debug modern digital systems, including:

- General debugging of FPGA logic or memory systems
- Analysis of program flow in embedded MicroBlaze processors
- Sophisticated analysis of internal FPGA activity time-correlated to external system operation

Additionally, if you are using embedded IBM 405 processors, you can view program flow for validation purposes by using a trace port and analysis software on a logic analyzer to reconstruct the processor activity. It takes some FPGA pins and a little thinking about how to best trigger or search for the data of interest. But in the end, you can save many hours of development time through these debugging methods.

You can find more detailed accounts of these debugging examples in a recent *Programmable Logic DesignLine* four-part series, available at [www.pldesignline.com/news/191901655](http://www.pldesignline.com/news/191901655).

For more information, contact your local Agilent sales representative, or visit [www.agilent.com/find/logic](http://www.agilent.com/find/logic) and [www.agilent.com/find/xilinxfpga](http://www.agilent.com/find/xilinxfpga). 

# WHAT'S NEW



To complement our flagship publication *Xcell Journal*, we've recently launched three new technology magazines:

- *Embedded Magazine*, focusing on the use of embedded processors in Xilinx® programmable logic devices.
- *DSP Magazine*, focusing on the high-performance capabilities of our FPGA-based reconfigurable DSPs.
- *I/O Magazine*, focusing on the wide range of serial and parallel connectivity options available in Xilinx devices.

In addition to these new magazines, we've created a family of Solution Guides, designed to provide useful information on a wide range of hot topics such as *Broadcast Engineering*, *Power Management*, and *Signal Integrity*.

Others are planned throughout the year.



See all the new publications on our website.

[www.xilinx.com/xcell](http://www.xilinx.com/xcell)