

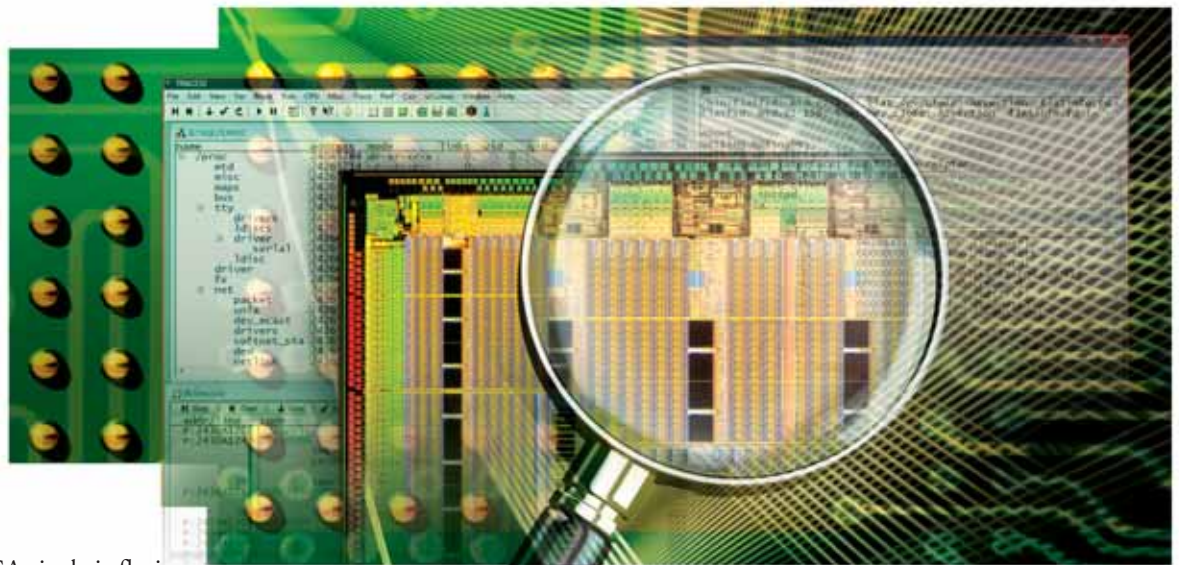
A Complete Debugging Solution for Xilinx Embedded Processors

Employ the industry-proven TRACE32 debugger to cope with the challenges of debugging advanced embedded processor systems.

by Jorge Carrillo
Staff Software Engineer
Xilinx, Inc.
jorge.carrillo@xilinx.com

Raj Nagarajan
Senior Software Engineer
Xilinx, Inc.
raj.nagarajan@xilinx.com

Oliver Oppitz
System Design Engineer
Lauterbach
oliver.oppitz@lauterbach.com



The best thing about FPGAs is their flexibility, which inspires designers to create a myriad of different designs. However, support for debugging the design is often considered last – if at all – so the debugger often has to adapt to the system.

The good news is that a debugger exists from a company that has been around for nearly 30 years in the embedded arena, gaining experience with all of the problems you can imagine and some that you do not even want to hear about. In this article, we'll show you a few examples of big and small features that Lauterbach's TRACE32 debugger offers – features that might save your day or even your project.

Flexibility in Debugging for a Flexible Platform

Speaking of flexible designs, a really interesting customer system comes to mind. It used two Xilinx® MicroBlaze™ processor cores with an internal block RAM memory

block on a Virtex™-5 LX50T device. The peculiar thing about this design was that each MicroBlaze processor had only an I-side interface to the block RAM block for instruction fetch. The second memory port of the block RAM was connected to a PCI Express interface, used to remotely change the application code and boot the processor at run time. The challenge was how to debug in a memory area that the debugger could neither read nor write, because the MicroBlaze processor itself could not perform load/store operations in this area.

We solved the problem using TRACE32's internal "virtual memory," a simulated memory inside the debugger that has an indefinite address space (64 bits) and for which memory is allocated on demand. We loaded the target program into this virtual memory and configured

the debugger internal address translation mechanism to remap access from unreadable target memory to virtual memory. This made it possible to diagnose the program even at the assembly level.

But there was another challenge: for stepping through the program, especially over high-level language lines and conditional branches, one normally uses software breakpoints. With no access to the block RAM from the MicroBlaze processor, this was out of the question. The solution came in the form of a "map.break" command, used to force the debugger to exclusively use hardware breakpoints for the given address range. Other flavors of the map command allow you to specify the data width of memories, change the endianness, or completely deny the debugger from accessing an address range with critical peripheral registers.

Small but Useful Features

Lauterbach's experience with JTAG debuggers and emulators shows in both the features of the TRACE32 debugger as well as in many small and unexpected details. Given that Lauterbach develops all software with its own tools, this is not surprising. And daily problems often inspire the most useful features.

Did you ever want to disable the annoying timer interrupt handler during a debugging session or invert a conditional branch without restarting? Ever wanted to patch a loop to see if the core executes correctly from external memory? The built-in assembler does just that.

Another typical scenario during debugging: How often did you just step one line too far and had to start all over? The register-restore feature will undo the last steps.

TRACE32 displays memory by permanently rereading the memory 10 times per second, even while the processor is stopped. Why does it do this? Perhaps a second MicroBlaze processor in your system, which you kept running, is causing some data corruption and you would like to also monitor this. Perhaps your sometimes unstable memory now causes tell-tale flickering on the screen. Or perhaps your JTAG interface is really not so stable at 20 MHz. These are things you would certainly like to know. On the other hand, TRACE32 guarantees that it will only access memories when required.

Having touched on the topic of peripherals, we should also mention the peripheral register files. These specify the location, width, and even bit-wise encoding of memory-mapped registers, grouping them into a tree of related registers. In this way the peripheral registers are easily accessible for inspection and modification: you can disable your DMA controller with a click instead of digging in the target manuals for the correct bit. The debugger comes with specifications for standard peripherals, but you can modify the files for your purposes using a simple text editor. For the Xilinx tool chain, an add-on is in the works to generate these files on the fly as part of the Xilinx build process.

Flexible IDE

TRACE32 offers a powerful graphical user interface (GUI), but its command-line origins still show in two highly useful features: the debugger command line at the bottom of the screen and the fact that practically every function of the GUI is also accessible through commands – and thus from scripts. This automates all of your routine tasks, including target configuration, laying out windows, and arranging them on multiple virtual screens. Best of all, the windows do not have a docking behavior like many IDEs but can be freely positioned and resized, even overlapping each other. There are also couplings with various IDEs, so you can invoke TRACE32 directly from your Eclipse environment, for example.

and will work with any design created with the Xilinx Embedded Development Kit (EDK). For PowerPCs, dedicated debugging connectors are also supported.

In the context of multi-core systems, the synchronized starting and stopping of cores becomes an issue. For targets supporting this in hardware, like in multi-MicroBlaze processor configurations, the debugger uses hardware features to achieve cycle-accurate synchronization; otherwise the synchronization is done in software. The integrated scripting language is multicore-aware, allowing control of all GUIs from a master script for connecting debuggers to their respective cores, resetting them, and downloading and starting application programs.



Figure 1 – Lauterbach TRACE32 debugging and trace cable connected to Xilinx ML507 board

Attaching to Multi-Core Targets

Another interesting feature is Lauterbach's intuitive approach to debugging multiple cores in the target. Just start an instance of the GUI for each core and have them share a debugging cable; this also works for heterogeneous systems with PowerPC and MicroBlaze cores or for another system from the 50-plus processor architectures supported by TRACE32 (Figure 1).

TRACE32 attaches to the same JTAG connector used by the Xilinx platform cable

Real-Time Program Flow and Data Trace

The main feature of the real-time trace is recording the program flow, defined as each and every instruction the processor executed as well as data transactions. For MicroBlaze processors, this is done through the Xilinx MicroBlaze trace core (XMTC) that comes with Xilinx Platform Studio. XMTC includes a trace encoder, which contains an input interface that attaches to a MicroBlaze processor trace port comprising approximately 200 unencoded signals.

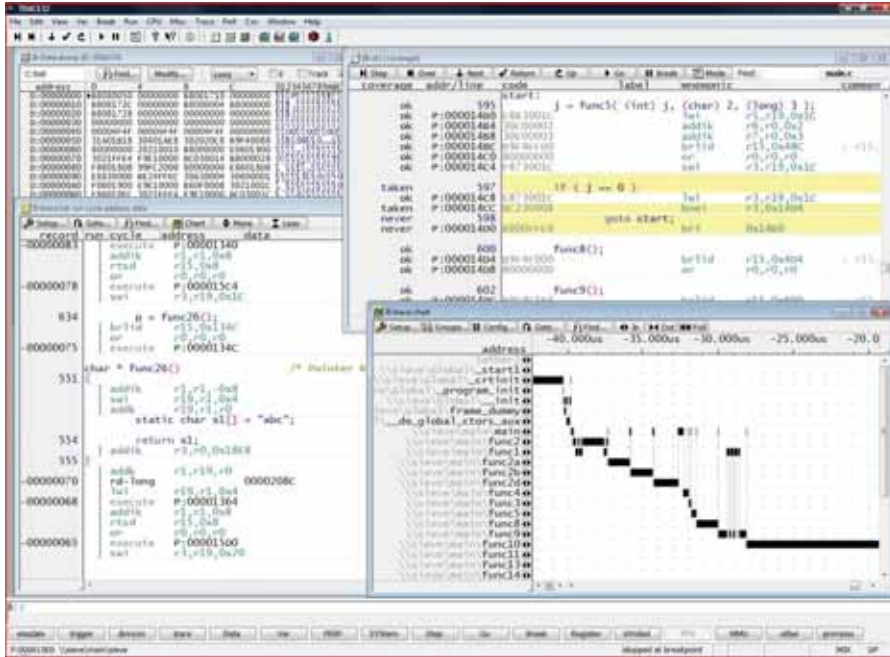


Figure 2 – Lauterbach TRACE32 IDE showing trace, code coverage, and function call chart views



Figure 3A – Lauterbach Mictor MicroBlaze trace adapter for Xilinx Spartan-3E FPGA board



Figure 3B – Lauterbach Mictor MicroBlaze trace adapter for Xilinx MLx boards

It also includes an output interface, which provides an encoded trace in 21 signals.

The trace hardware provides up to 512 MB of external high-speed trace memory, which is used instead of the scarce on-chip memory resources for storing trace information. Trace functionality is also supported on PowerPC architectures. More advanced features include statistical analysis of function and task run times, variable access, and code coverage analysis (Figure 2, Figure 3A and 3B).

Operating System Support

On the MicroBlaze processor, TRACE32 comes with a so-called kernel awareness module for μ CLinux and Linux. For PowerPCs, many more operating systems are supported, including QNX, VxWorks, and Nucleus PLUS. The extensions make the debugger aware of the kernel-related data structures in the target. They allow debugging on the process level using process-specific breakpoints and program control. Other features include full MMU support; real-time, non-intrusive display of Linux system resources like loaded kernel modules or mounted file systems; statistical evaluation and graphical display of task run times; and task-related evaluation of function run times.

Conclusion

The Lauterbach TRACE32 provides a comprehensive debugging solution for PowerPC and MicroBlaze processors on all Xilinx device families. Future Xilinx-related additions will be to enhance the debugging cable so that the Xilinx ChipScope™ analyzer can use it for target accesses jointly with the debugger and downloading FPGA bitstreams through the debugger itself.

For more information, please visit www.lauterbach.com.

Xilinx Global Trade Show Calendar

Xilinx participates in numerous trade shows and events throughout the year. This is a perfect opportunity to meet our silicon and software experts, ask questions, see demonstrations of new products, and hear other customer success stories.

For more information and the most current schedule, visit www.xilinx.com/events/.

April 14-17, 2008

NAB 2008
Las Vegas, NV

April 15-17, 2008

Embedded Systems Conference
Silicon Valley
San Jose, CA

April 15-17, 2008

Special Event Effects
Symposium
Long Beach, CA

April 21-22, 2008

China International Medical
Electronics Technology
Shenzhen, China

May 14, 2008

Linley Tech Seminar:
High-Speed Interconnects
San Jose, CA

May 14-16, 2008

Embedded Systems Expo 2008
Tokyo, Japan

