Germany Lauterbach GmbH
France Lauterbach S.A.R.L.
Great Britain Lauterbach Ltd.
Italy Lauterbach Srl
USA Lauterbach Inc.
China Suzhou Lauterbach Technologies Co. Ltd.
Japan Lauterbach Japan Ltd.

LAUTERBACH DEVELOPMENT TOOLS

LONG-TERM TRACE ETMv3

Implementation

Lauterbach, the leading manufacturer of real-time trace tools, is introducing its new long-term trace for the ARM ETMv3. The aim of this innovation is to enable greatly extended measurement times for TRACE32 profiling and code-coverage functions.

This article describes the concepts of the long-term tracing technology as well as the technical requirements it places on the trace tool and the respective host computer.

ARM ETMv3

Tracing means recording detailed information about a program as it runs on the core. This information is usually generated by on-chip trace logic. For ARM cores, this logic element is known as the *Embedded Trace Macrocell* or ETM. The latest version of this logic, the ETMv3, can be found today in most ARM11 and Cortex cores. Since the functionality of the on-chip trace logic is the basis for the trace data, let's start with a brief introduction.

The ETMv3 generates a package-oriented trace log. The following information can be generated at program runtime and collected in trace packages:

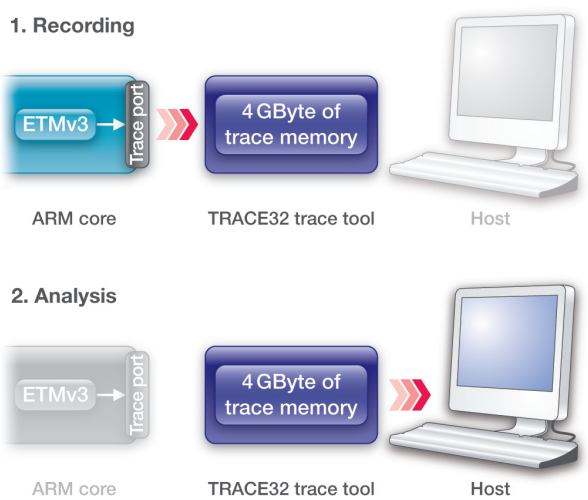
- **Program flow packages:** Contain information about the program instructions executed by the core – mainly the target addresses of jumps as well as the number of instructions executed between two jumps.
- **Data flow packages:** Contain the memory addresses read/written by the program as well as the respective data values.
- **Context-ID packages:** Contain a process/task ID in the event that an operating system is running.

The trace packages are output by the on-chip trace logic via the trace port. The trace port for the ETMv3 typically consists of 8 or 16 pins for the trace packages plus two pins for the control signals.

To minimize the bandwidth for the package output, the ETMv3 compresses the trace packages. For example, all addresses are shortened by a special algorithm. However, if the data volume is higher than the maximum bandwidth of the trace port, the FIFO buffer can overflow and some trace packages can be lost.

Compressing the trace information alone is not enough to prevent FIFO overflows. The next stage is provided by the programmability of the ETMv3. To reduce the number of trace packages, you can simply define what trace information you want generated and output. For example, no data flow information is needed for the TRACE32 profiling functions. This is very useful because it is mainly the data packages that inflate the trace volume at the port.

Classic Tracing



Classic tracing means: record first, then analyze

Classical tracing currently consists of two steps that are carried out consecutively: »

1. Recording

The trace packages are sampled at the trace port and placed in the trace memory.

2. Analysis

The trace packages are transferred from the trace memory to the host, where they are decompressed and analyzed.

The classical tracing method is limited in that only the section of the program that fits within the buffer memory can be saved for evaluation and analysis. The memory depth of the TRACE32 trace tools is currently between 1 GBytes and 4 GBytes. This allows the recording of up to 3 G trace packages.

Recording

In classical tracing, the real challenge is the recording of the trace packages. Since ARM cores today usually run at frequencies up to 1 GHz, only a fast trace port can guarantee the loss-free output of all trace packages.

The Lauterbach trace tools for the parallel ETMv3 support package recording at a frequency of up to 275 MHz DDR and can therefore handle the following data rates (see Figure 1):

- 8.8 GBit/s with 16 pins for the trace package
- 4.4 GBit/s with 8 pins for the trace packages

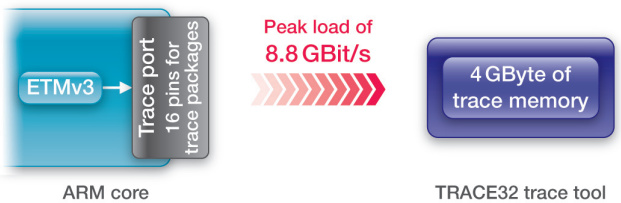


Fig. 1: The trace tool for the parallel ETMv3 supports a data rate of 8.8 GBit/s with 16 pins for the trace packages

With the serial trace tools for the ETMv3, data rates of up to 20 GBit/s can be recorded.

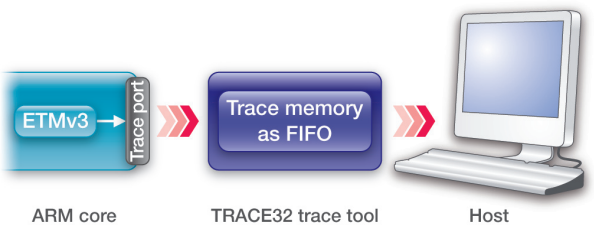
Analysis

To analyze the recorded program section, you have to transfer the trace packages from the trace memory to the host, decompress and then evaluate them.

Since the trace packages for the program flow contain no program code, this has to be added prior to analysis. The following data is used:

- The program code read by the TRACE32 software from the target system memory over the JTAG interface.
- Symbol and debug information loaded by the user for the TRACE32 software.

Long-Term Tracing



Long-term tracing means: recording and direct analysis

Long-term tracing is implemented by transferring the trace packages to the host during recording and analyzing them immediately. In this case, the trace memory of the TRACE32 trace tool is basically used as just a FIFO.

Extremely large data volumes are created during a long-term trace, so it is advisable to analyze the trace packages in parallel with the recording. Even if the trace packages are compressed before they are stored in a file, up to 5 GBytes are typically collected per hour. At the same time, you have to allow a lot of time for further analysis after recording is completed. For example, if you collect trace packages for a two-hour program run in a file, you need several hours for a subsequent conventional analysis, even on a high-powered host. »

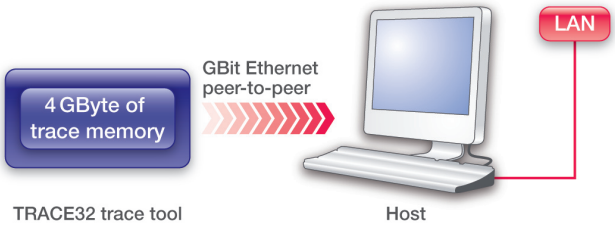


Fig. 2: Long-term tracing needs a fast peer-to-peer interface to the host

If large data volumes are to be quickly recorded, transferred, and analyzed in long-term tracing, the following conditions must be met:

- Fast host
- Fast trace tool
- Compact data formats

Fast Host

In order to be able to analyze the trace packages on the host at program runtime, you need a fast dual-core computer. Here, one core receives the trace packages while the second core evaluates the packages in parallel.

For the analysis, the program code is needed in addition to the trace packages. Since many ARM cores are not able to read the code from the target system memory at runtime, the code must be copied to the TRACE32 software before the start of the long-term trace.

Fast Trace Tool

As described above for classical tracing, the trace tool has to sample the trace packages loss-free at a fast trace port. High-speed transfer of the trace packages to the host is the new requirement for long-time tracing. For this purpose, the TRACE32 trace tool provides a GBit Ethernet interface. If the trace tool is connected peer-to-peer to the host, a transmission rate of more than 500 MBit/s can be achieved (see Figure 2).

The maximum transmission rate to the host is currently the bottleneck of long-term tracing. This means that long-term tracing will only work if the average data rate at the trace port does not exceed the maximum transmission rate to the host (see Figure 3).

High peak loads are not critical since they can be buffered by the trace memory. »

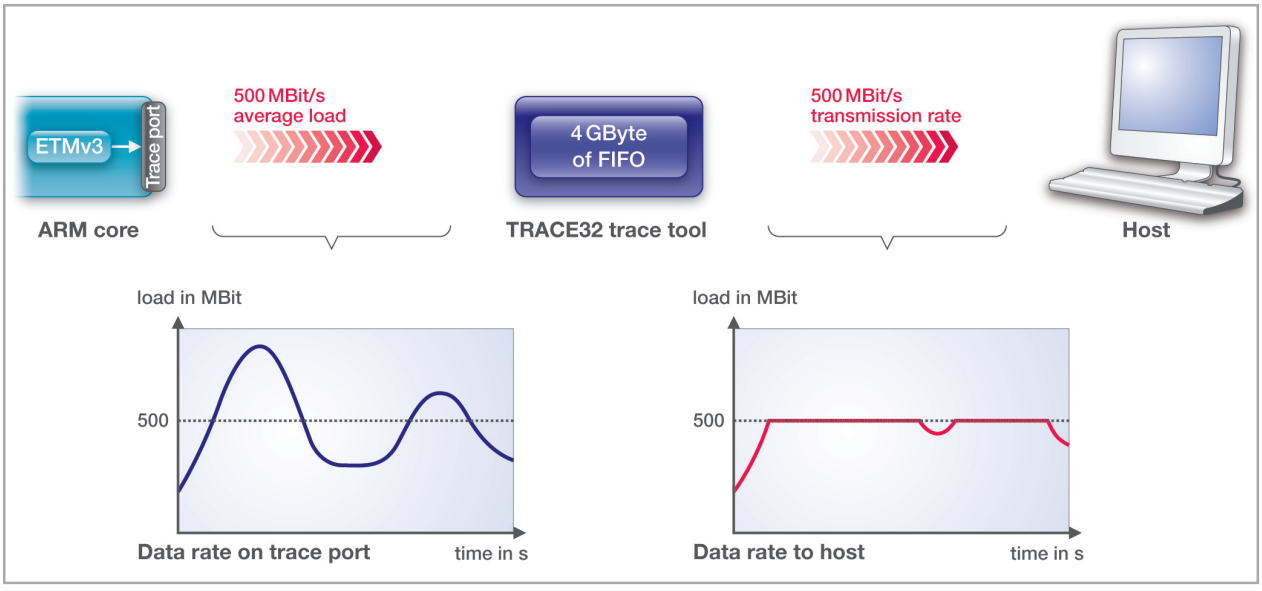


Fig. 3: Long-term tracing works for this example if the average load at the trace port does not exceed 500 MBit/s

Software	Mobile Terminal	Floating Point Arithmetic	HDD Controller
Trace information per instruction	0.8 Bit	2.2 Bit	4.3 Bit
Core	Cortex-A	ARM11	ARM9
Core frequency	500 MHz	300 MHz	450 MHz
Trace port frequency DDR	166 MHz	75 MHz	150 MHz
RTOS	Linux	—	—
Average data rate at trace port	340 MBit/s	406 MBit/s	798 MBit/s

Compact Data Formats

Since the maximum transmission rate to the host is limited, it is important to keep the data volume as compact as possible. The data volume can be influenced in two ways:

1. Optimal programming of ETMv3
2. Compact buffering of trace packages

1. Optimal programming of ETMv3

You can directly influence the data rate at the trace port by programming the ETMv3 so that trace packages are only generated for analysis-relevant information. The data flow packages that represent a high load for the trace port are not usually needed for profiling and code coverage.

The other factors influencing the average data rate at the trace port unfortunately have to be considered as unchangeable:

ARM core frequency: The higher the ARM core frequency, the more trace data per second.

Software on the target system: A software program that makes a large number of jumps and finds data/instructions in the cache generates more trace packages per second than a software program that pro- »

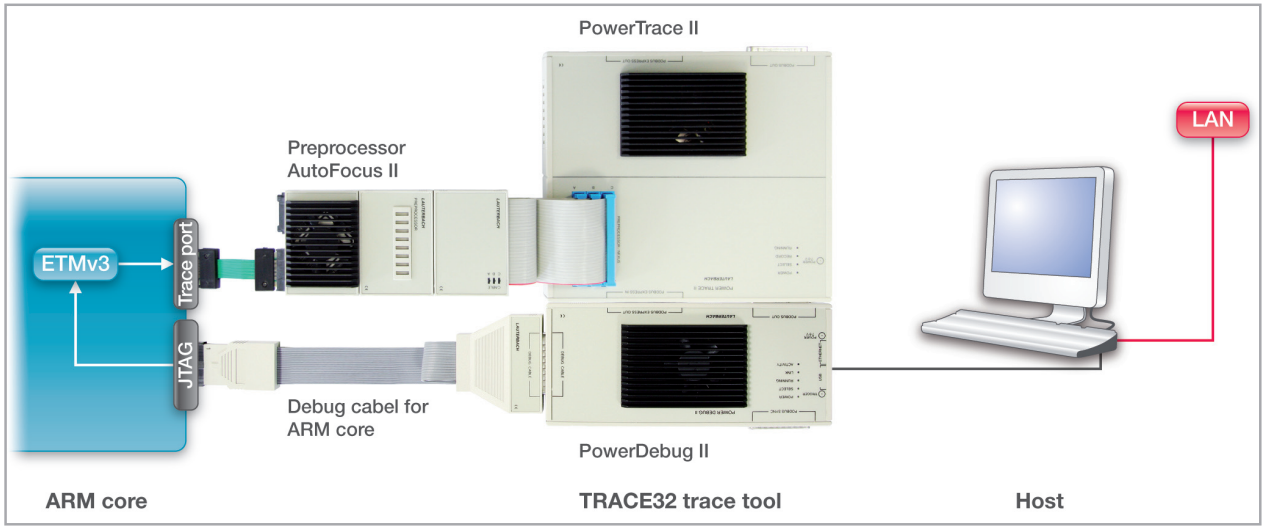


Fig. 4: The Lauterbach trace tool for long-term tracing with the ETMv3

cesses many sequential instructions and often has to wait for the availability of data/instructions.

The table on page 4 shows a few examples of average data rate measurements at the trace port. It is surprising that the data rate is greatly influenced by the software running on the core. The core frequency and the core architecture do not play as significant of a role.

2. Compact buffering

The firmware of the TRACE32 trace tool has been enhanced so that, the optimal packing density of the packages in the trace memory is achieved with 8 pins for the output of the trace packages.

Summary

In the TRACE32 software, the configuration and analysis of the long-term trace runs under the name of *Real-Time Streaming* – RTS for short.

A Lauterbach trace tool for long-term tracing of the parallel ETMv3 consists of the following TRACE32 products (see Figure 4):

PowerDebug II: Provides the GBit Ethernet interface to the host and transmits the trace packages.

Debug cable for the ARM core: Programs the ETMv3 over the JTAG interface.

PowerTrace II: Stores the trace packages, max. trace depth currently 4 GBytes.

Preprocessor AutoFocus II: Samples the trace packages at the parallel trace port and transfers them to the trace memory.

With long-term tracing, Lauterbach has taken an important step toward a trace technology that permits an almost unlimited analysis of the program run. It is fair to assume that both the processing power of the hosts and hard-disk capacity will increase constantly during the next few years, so we expect even more comprehensive analysis options will become available.

LONG-TERM TRACE ETMv3

Code Coverage-Analysis and Long-Term Tracing

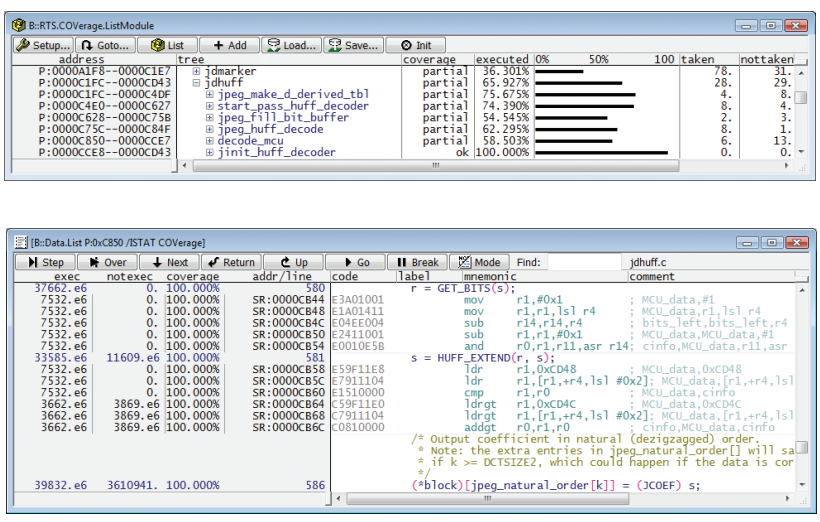


Fig. 5: Lists showing code coverage (overview and detailed)

One application for long-term tracing is checking whether all of the program code is processed during a system test.

For this code-coverage analysis of the trace data, the TRACE32 software provides a list of all modules/functions and their code coverage. Additionally, a statistical summary of the execution of conditional instructions is displayed. Each function can be analyzed in detail: For linear code, you can see how often a command was run during the test (exec). For conditional instructions, you can also observe how often a command was skipped because its condition was not met (notexec).

LONG-TERM TRACE ETMv3

Profiling and Long-Term Tracing

For time-critical functions, maximum times are often defined and have to be checked during the system test. The long-term tracing feature makes this check easy and causes of timeouts to be quickly found.

Verification

First is the check whether time-critical functions exceed their maximum time. For this purpose, the ETMv3 should be programmed so that it generates only trace packages for the program flow and the context ID. There are two reasons for this:

1. In this way, the data rate at the trace port is kept as low as possible.

2. FIFO overflows preventing an exact analysis of function nesting are prohibited.

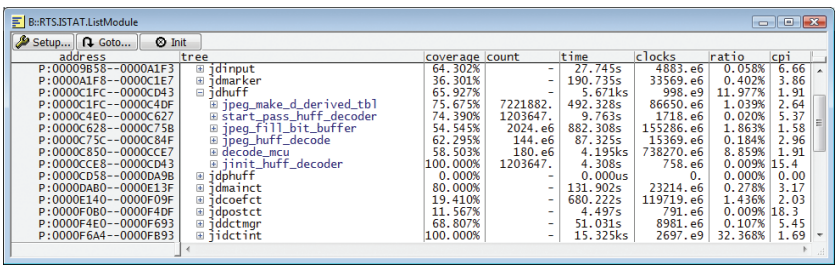


Fig. 6: Analysis of time behavior of modules and functions

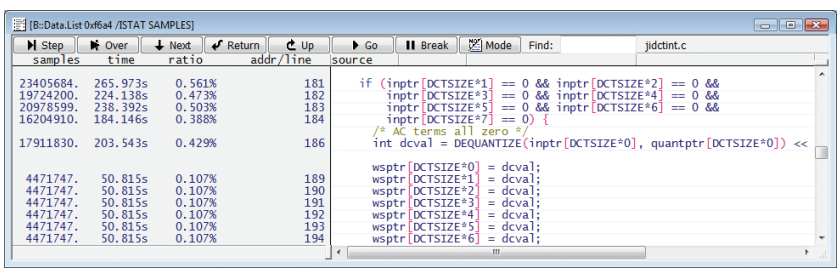


Fig. 7: Details of time behavior of program lines of a function

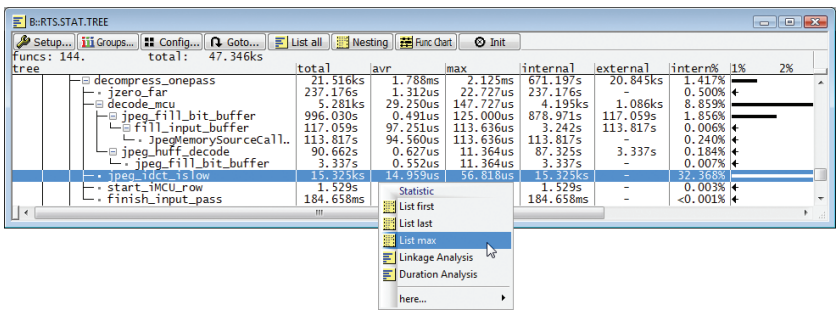


Fig. 8: If necessary, details on the longest function run can quickly be displayed

After the long-term trace is started, the TRACE32 software analyzes the time behavior of the functions. The following are analyzed: the runtime and number of clock cycles during the complete test run, the function's share of the total runtime, and the average "clocks per instruction" (see Figure 6).

A detailed analysis of the individual function also shows the time behavior of the program lines (see Figure 7).

If the analysis intermittently exceeds the defined maximum time, the cause must of course be found.

Troubleshooting

The long-term trace can be configured so that the trace packages are saved in a file at program runtime. With a data volume of 5 GBytes/hour, about four days of the program runtime can be recorded on an average hard disk. Using fast, sophisticated search functions, the TRACE32 software can then search the trace recording for the excessive function run and show it in a detailed analysis (see Figure 8).