Debugging AMP and SMP Systems

Many multi-core processors can be used as either AMP or SMP systems. Depending on the mode of operation, different debug and trace concepts are applicable. In the following article, we describe how these concepts can be applied using TRACE32 to debug an ARM Cortex-A9 MPCore.

Debug Concepts

In discussion with our customers, we realize again and again that there are many varying interpretations of the two terms:

- **AMP – asymmetrical multiprocessing**
- **SMP – symmetrical multiprocessing**

Therefore we think it is worth taking the time to explain how these terms are used at Lauterbach and what effect they have on the configuration and usage of a TRACE32 debugger.

As the term “multiprocessing” implies, multiple cores are working together in an embedded system. What is crucial for debugging is how the system tasks are distributed to the individual cores.

Debug Concept for AMP Systems

In AMP systems, each core is assigned a specific task. How the tasks are distributed is determined in the design phase of the system. In addition to a standard controller (usually RISC architecture) specialized accelerators (DSPs or customized cores) are frequently chosen.

When debugging AMP systems, an individual TRACE32 instance is started for each core (see Figure 4). There are two reasons for this:

1. An AMP system can contain different core architectures.
2. Each core processes a separate part of the application. This means that the majority of the symbol and debug information is assigned exclusively to the corresponding core.

However, because the cores do not work independently, but perform the application task together and in parallel, it must be possible to start and stop all cores simultaneously. This is the only way to test the interaction between the cores and to monitor and control the entire application. There are different ways to start and stop all cores simultaneously. Ideally, the multi-core processor will support this through internal synchronization logic. If this logic is missing, TRACE32 takes over the synchronization process. A special algorithm calculates JTAG command sequences to control all cores as promptly as possible. »

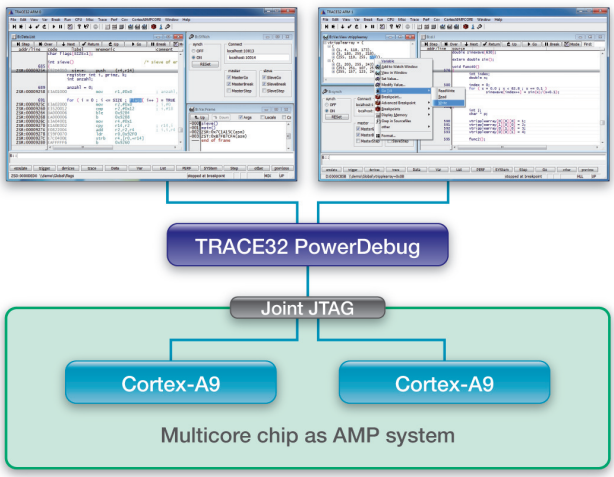

Fig. 4: When debugging AMP systems, an individual TRACE32 instance is started for each core.

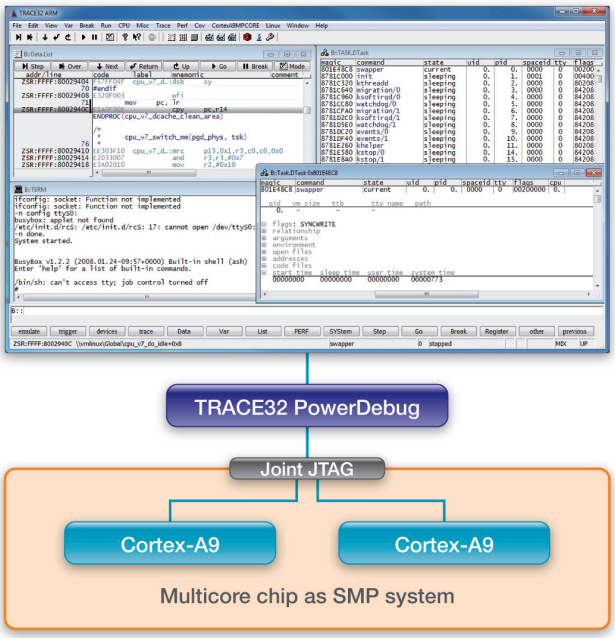

Fig. 5: When debugging SMP systems, a single TRACE32 instance is started for all cores.

Debug Concept for SMP Systems

In contrast to AMP systems, where the tasks assigned to each core are predefined, the assignment in SMP systems is flexible. In an SMP system, the system designer no longer assigns tasks to cores. An SMP operating system does this instead. All cores must be the same type to enable tasks to be assigned freely to each core as required.

Task assignment is performed dynamically, meaning that the assignment depends on the current system status. A task unit that can be assigned by an operating system is called a “task” or “thread”. In simple terms, a task that needs to be processed is assigned to a core that is currently free.

For debugging SMP systems, only **one** TRACE32 instance is opened and all cores are controlled from this one point (see Figure 5 on the previous page). Because the developer is primarily concerned with debugging a single task, the TRACE32 user interface displays the state of the entire SMP system from the perspective of this single task or from the perspective of the core where the task is running. Of course the visualization can be switched to other tasks or cores if required.

TRACE32 assumes a function that is similar to an SMP operating system. It organizes the debugging of all cores so that developers do not need to look into the details of the SMP system. For example, if a breakpoint is set, TRACE32 makes sure that the breakpoint is entered in all cores. This is necessary because at the time when the breakpoint is set, it’s not yet clear which core will execute the program section with the breakpoint. If a core stops at a breakpoint, all other cores are also stopped automatically. The display in TRACE32 switches to the task or core that hits the breakpoint. If the program is restarted, all cores start running together.

Debugging SMP systems with TRACE32 is easy. After a TRACE32 instance is started and configured for the SMP system, the developer essentially use it as if he were debugging only one core.

Trace Concepts

TRACE32 analyzes and displays trace information in different ways, depending on whether the trace data was generated by an AMP system or an SMP system. For AMP systems, trace analysis is largely performed on each core independently. The trace information for an SMP system, however, can be analyzed for a single task, a single core, or for the entire system, depending on the type of query.

Trace Concept for AMP Systems

Because debugging individual cores of an AMP system is performed over separate TRACE32 instances, trace information is also displayed on these individual user interfaces. AMP systems can consist of different types of cores, so different trace protocols might be used. As the individual trace streams are displayed in the separate user interfaces, they can be individually decoded and analyzed.

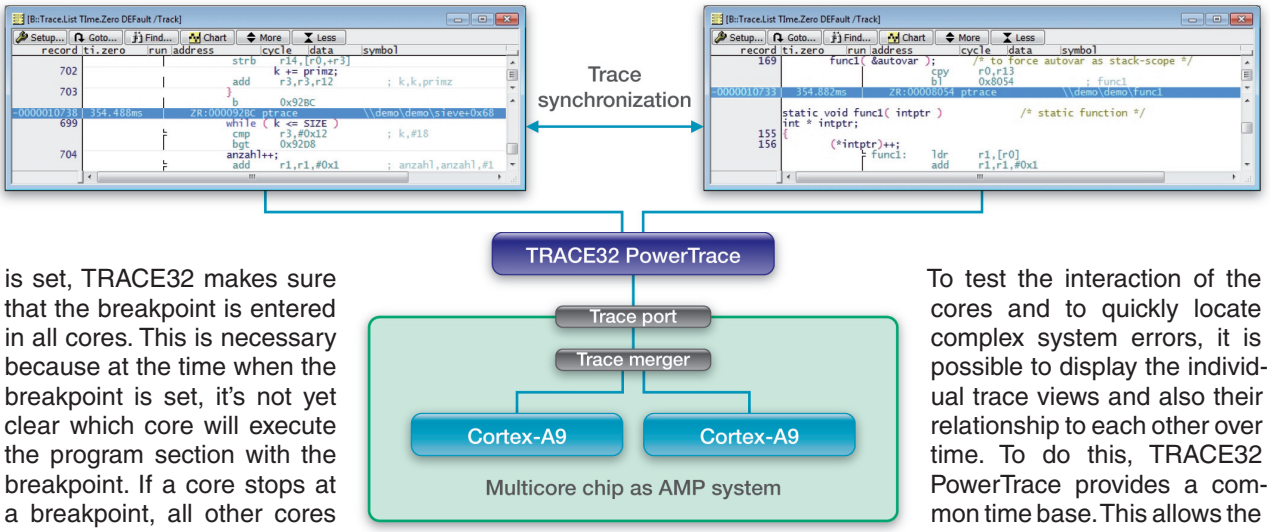

Fig. 6: When tracing AMP systems, the trace information for each core is displayed on a separate user interface. Time synchronization of the user interfaces is possible.

To test the interaction of the cores and to quickly locate complex system errors, it is possible to display the individual trace views and also their relationship to each other over time. To do this, TRACE32 PowerTrace provides a common time base. This allows the developer to select a point in time in the trace view on one user interface and see exactly »

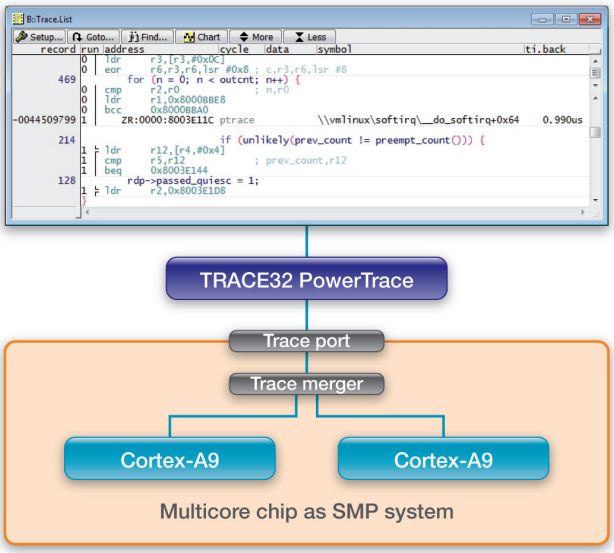

Fig. 7: When tracing SMP systems, the information for all cores is stored in a shared trace memory.

which command was being executed by another core at approximately the same time (see Figure 6).

Trace Concept for SMP Systems

All information about the programs processed on an SMP system is stored in a shared trace memory for all cores (see Figure 7). One of the advantages of TRACE32 is that it provides different views of this information.

To locate errors in a task or for task-specific runtime measurements, trace information can be displayed specifically for an individual task.

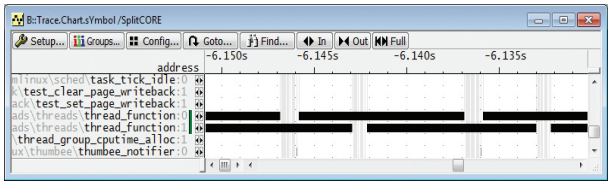

Fig. 8: The trace analysis shows which cores processed the individual program sections.

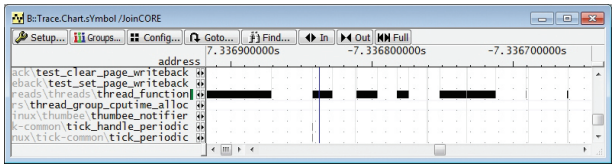

Fig. 9: The trace analysis analyses the SMP system as a whole; which core processed which program section is unimportant.

If you want to know information such as “Which cores processed my task?” or “What is the run-time load of my cores?”, it can be useful to view the trace information for all cores at the same time. Figure 8 shows an example of this view. The core number (0 or 1) indicates the cores on which the individual program sections ran.

In order to examine the SMP system as a whole, it is not necessarily to know which core processed which task or program section. TRACE32 also provides display options for this type of view of the SMP system (see Figure 9).

During 2010, Lauterbach will continue to enhance the preparation and display of trace information from SMP systems. This will include new analysis functions based upon feedback from existing users and also new concepts currently in development.