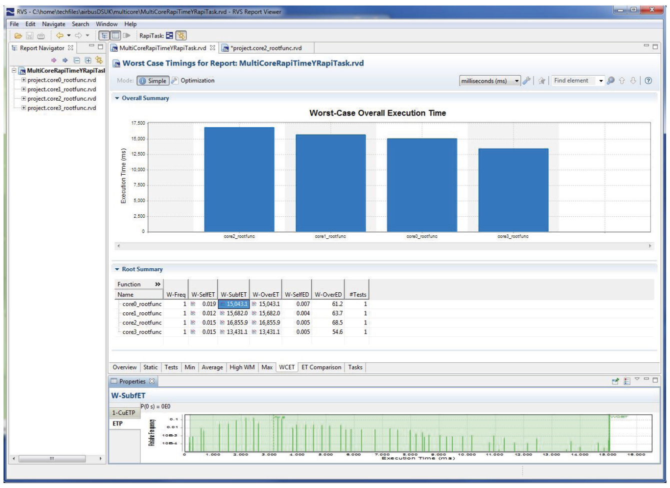
Multicore verification with Rapi**Time**, Rapi**Task** and RTEMS®

To show that your application meets hard real-time requirements, you need to know the worst-case response time of time-critical threads within the application. To establish this, you need to satisfy two conditions: use an RTOS with deterministic scheduling behaviour and know the worst case execution time (WCET) for each thread.

The first condition is met by RTEMS with real-time processes (RTPs). The second can be met by Rapi**Time** WCET analysis of individual threads. Rapi**Task** can also be used to identify and analyze the different task in a multicore environment. Part of the Rapita **Verification Suite** (R**VS**), Rapi**Time** goes beyond the capabilities of conventional performance profilers in several ways. Specifically, Rapi**Time**:

- Collects distributions of execution time measurements taken directly from a multicore target.
- Allows the level of detail for measurements to be configured on the level of lines or blocks of code, functions or entire subsystems.
- Combines measurements with a structural model of the application's source code to determine predicted WCET.
- Displays the predicted worst-case path through the code.
- Shows which lines of code have the biggest impact on WCET.



Root Summary

Name	W-Freq	W-SelfET	W-SubfET	W-OverET	W-SelfED	W-OverED	#Tests
core0_rootfunc	1	0.019	15,043.1	15,043.1	0.007	61.2	1
core1_rootfunc	1	0.012	15,682.0	15,682.0	0.004	63.7	1
core2_rootfunc	1	0.015	16,855.9	16,855.9	0.005	68.5	1
core3_rootfunc	1	0.015	13,431.1	13,431.1	0.005	54.6	1

Figure 1. Multicore Rapi**Time** report

Rapi**Task** lets users:

- View a high-level overview of multi-core software system scheduling.
- Locate rare timing events in multi-core systems, such as race conditions and interference.
- Identify on which cores in multicore system each task is executing on.
- Understand the workload (e.g. CPU usage) of the system

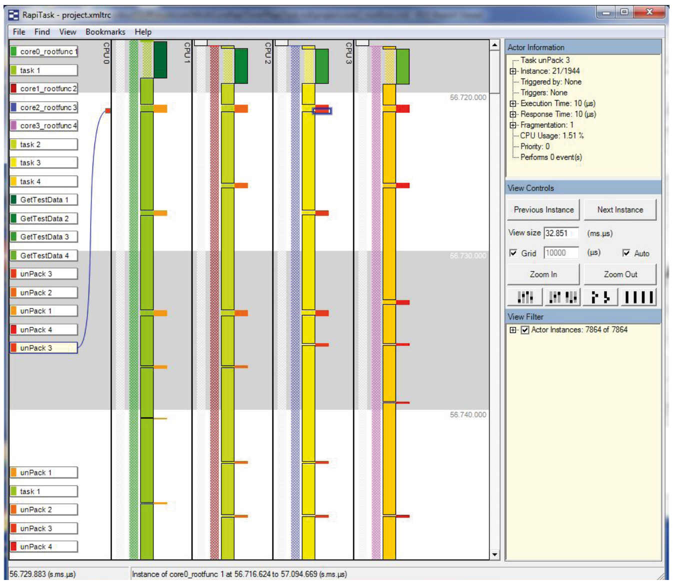


Figure 2. Rapi**Task** showing RTEMS running on a 4 core processor

How Rapi**Time** works

Figure 3 shows how R**VS** fits into the build process. The instrumenter embeds instrumentation points (Ipoints) into code between pre-processing and compiling the code.

Using data from the instrumenter, the structural analysis generates a complete structural model of the application. During testing, trace data in the form of instrumentation point identifiers and timestamps are collected. Once the trace data has been collected, it is combined with the structural model in the Rapi**Time** Database, which can be examined in the Report Viewer.

R**VS**'s trace collection is via an open interface. This allows the trace to be captured in a variety of ways where it is captured by a logic analyzer or Rapita's data logger, **RTB**x.

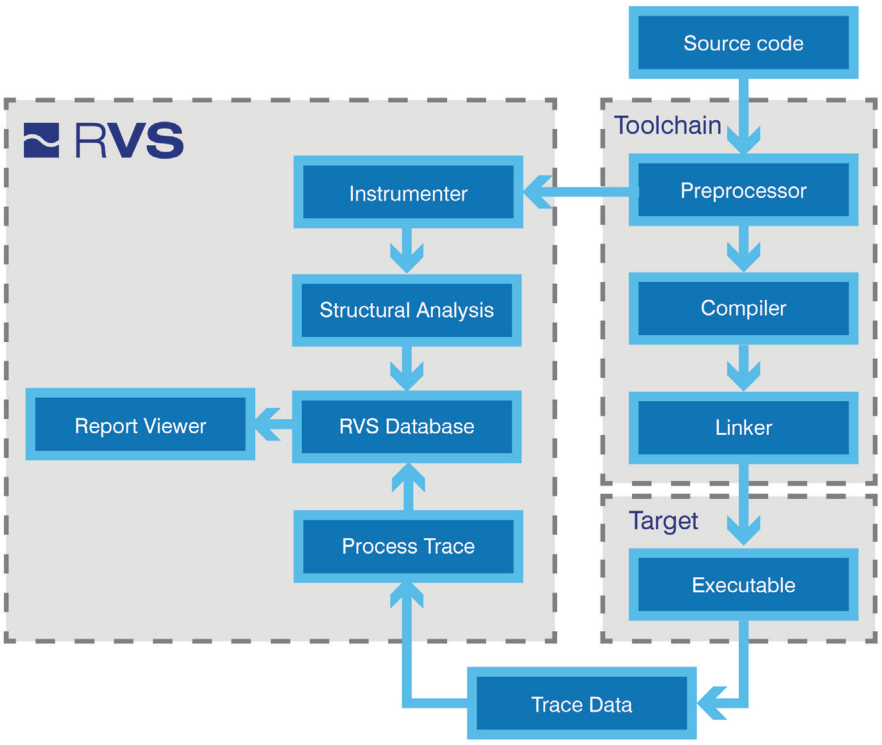


Figure 3. R**VS** data collection workflow

For software capture, an on-target buffer is used. At each instrumentation point, the identifier and timestamp are recorded in the buffer. RTEMS tasks can spool trace data off the target for later analysis.

Target integration kits can be provided to support different trace interfaces.

Multicore WCET analysis and Scheduling analysis with Rapi**Time** and Rapi**Task**

Collecting the execution time from a specific task within RTEMS means eliminating the time spent executing other (pre-empting) tasks and interrupts from any time measurements.

For each core, a trace file is generated. This contains information about all of the tasks which were executed on the core. Eliminating time in other tasks requires an instrumented context switch mechanism. This is achieved by adding an instrumented function into several hooks which are called when a switch context operation occurs, such as init, create, switch or stop. In such functions, the involved tasks are identified and captured to enrich the analysis.

In situations where the execution time of several tasks is of interest, Rapi**Time** can collect a trace of multiple tasks concurrently. This can then be demultiplexed into reports on separate tasks.

Using the same mechanism and the same produced trace, Rapi**Task** can provide useful information about the multi core system under analysis; a high-level task overview, rare timing events and timing behavior.

About Rapita Systems

Rapita Systems provides customized on-target verification solutions which reduce the cost of measuring and optimizing the timing performance of large real-time software systems in the avionics and automotive electronics markets.

The Rapita **Verification Suite** (R**VS**), which includes Rapi**Time** and Rapi**Cover**, is the essential collection of on-target timing verification, optimization and code coverage measurement tools for real-time embedded systems. It is the only product on the market that can tell users exactly where to focus optimization effort to minimize worst-case execution time.

Using R**VS**, customers have cut the worst-case execution time of large scale, legacy applications by up to 50% with only a few days effort, and significantly reduced unnecessary testing and instrumentation overheads. Rapita's software supports Windows and Linux and Ada, C and C++ projects.

For information about Rapi**Time** and the timing information it can provide, see the Rapi**Time** brochure and the Rapi**Time** White Paper available at www.rapitasystems.com/downloads.

Glossary

Execution time:

The time spent executing a specific thread between two points. This time excludes time spent executing any pre-empting threads or interrupts.

Instrumentation point identifiers (Ipoint Ids):

Constant values that allow an external view of the application's progress to be made. With Rapi**Time**, Ipoint Ids do not need to be unique.

Response time:

The elapsed time between some initial stimulus to a system and that system's response. In the case of a multi-threaded application, this may include the time spent executing several tasks and/or interrupts.

Worst-case execution time (WCET):

The longest possible execution time for a given thread.

Worst-case response time (WCRT):

The longest possible response time that can occur for a given system. Note that for non real-time applications this can be unbounded.