RAPITA SYSTEMS LTD

Product brief

Zero Overhead Instrumentation with Rapi-Time

Rapi**Time** (part of Rapita Verification Suite, RVS) uniquely provides analysis of timing and code coverage from target measurements. Key to its approach is instrumentation of the source code. A new enhancement is the ability for certain development environments to instrument source code without additional instructions being added into the object code - zero overhead instrumentation.

Introduction to Rapi**Time**

Rapi**Time** is an automated performance measurement and timing analysis tool. Targeted at real-time, embedded applications, Rapi**Time** provides code coverage metrics, performance measurement, determination of worst-case execution time (WCET), and guidance for optimization efforts.

In general terms Rapi**Time** works as follows:

- As the application is executed on the target, an execution trace is collected. This trace is a sequence of time-stamped values that show which parts of the application are executed.
- Rapi**Time** identifies the parts of the source code have and have not been executed (code coverage), and performance metrics for each part of executed code.
- Using the performance and coverage data, Rapi**Time** can also predict WCET, and identify the best locations for optimization.

The execution trace is created by instrumenting the source code with expressions (instrumentation points, or Ipoints) that indicate that a specific section of code has been executed. On many platforms the code for the instrumentation can be implemented with as few as one machine instruction per instrumentation point, thus leaving a very small overhead. However, there are situations where even a small overhead is undesirable (see "Why no instrumentation?", overleaf).

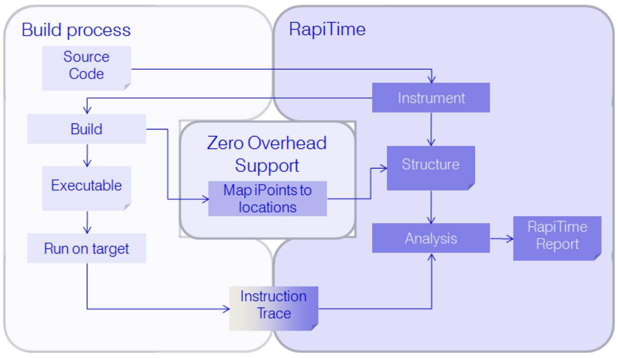

Figure 1: Zero Overhead Instrumentation

Zero overhead instrumentation

In some situations it is possible to provide "zero overhead instrumentation". Although this approach does insert Ipoints into the source code, they are labels only. This means no additional code appears in the executable.

Two key technologies are required to make this work:

- Compiler support for mapping labels at source code level to memory locations.
- Debugger/data logger support for collecting a time-stamped trace of locations of code statements executed.

The sidebar on the next page shows tools compatible with zero overhead instrumentation.

How it works

Figure 1 shows the zero overhead instrumentation process. The steps in the process are as follows:

1. Rapi**Time** inserts Ipoints into the source code ("Instrument").
2. The build process expands Ipoints to labels in the object file at build time ("Build")
3. Rapita's support for zero overhead instrumentation automatically updates the structural model to use the memory locations of

the labels instead of Ipoint IDs ("Map Ipoints to locations")
4. The application is executed with some technology for recording a timestamped trace of locations of executed instructions ("Run on target")
5. Rapi**Time** combines the trace file with the structure to produce a report ("Analysis").

Why no instrumentation?

Inserting a small amount of code into an application to implement instrumentation usually provides an excellent analysis solution. Rapi**Time**'s analysis can even compensate for the slight increase in time required to execute instrumentation. However there are cases where even this cannot be tolerated. Two examples of this are:

Critical applications. Inserting instrumentation code into the application for measurement purposes introduces the remote possibility of different behavior between measured and deployed code. Development processes for high integrity systems (e.g. DO-178B for avionics) require you to prove that this does not occur. Typically this is achieved by running tests twice:

- Once with instrumentation, to perform measurements.
- Once without instrumentation, to prove that the application's behavior is unaffected by the instrumentation.

With zero overhead instrumentation, it is possible to deploy the same code that is used for measurements, avoiding the need for duplicated tests.

Memory limited applications. In some situations there is very little additional room for new code. In these situations, even the small increase due to instrumentation code is undesirable.

Summary

Zero overhead instrumentation gives you all of the benefits of Rapi**Time**'s analysis without introducing instrumentation code into your application.

Not every environment supports zero overhead instrumentation. However, where your development environment does support it, Rapita Systems will be pleased to provide a Rapi**Time** integration service or to support your own integration activity to make use of zero overhead instrumentation. Please contact us for further information.

Supported technologies

Compiler. Zero overhead instrumentation will work with any compiler that can:

- Insert assembly labels at source level (for example using inline assembly strings);
- Provide somewhere that the address of located labels can be extracted from (e.g. map or ELF files).

Rapita Systems is developing support for specific compilers. Please contact us for information.

Debugger/Data Logger. Zero overhead instrumentation can be implemented with the following debugger/data logger solutions:

- iSYSTEM TraceGT debugger connected to CPUs with Nexus or ARM ETM;
- Lauterbach PowerTrace debugger connected to CPUs with Nexus or ARM ETM;
- Rapita Systems RTBx data logger or Logic Analyzer connected to the address bus of a CPU without on-chip program cache;
- CPU simulator with support for time stamped instruction trace.

About Rapita Systems

Rapita Systems Ltd 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.

RVS, 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 WCET.

Using RVS, 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.

Our software supports Microsoft Windows (XP, 2000, Vista and Windows 7) and Linux.

The zero overhead instrumentation concept was developed in MERASA, a research project funded by the 7th European Framework Programme (FP7) under contract no 216415. More information can be found at http://www.merasa.org/