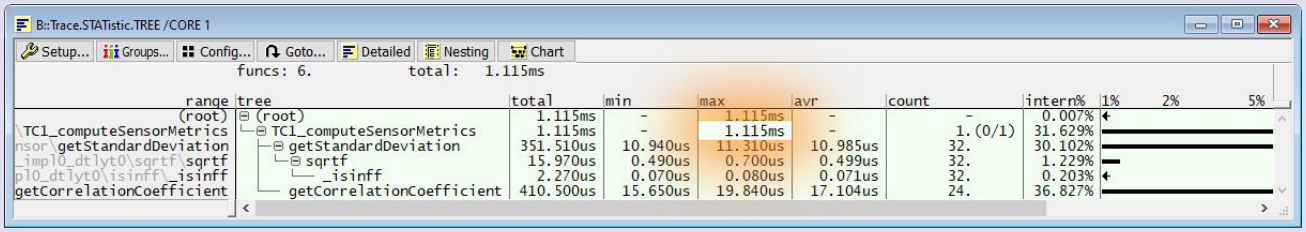
Profiling with Performance Counters for TriCore™ AURIX™

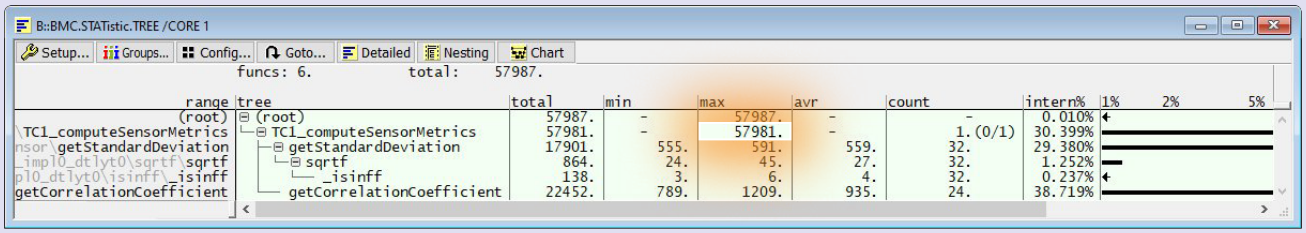
Andrea Martin, Technical Writer

Cache misses, data hazards and incorrect branch predictions have a negative impact on function runtimes. To investigate and minimize the impact of these events, the Infineon AURIX™ microcontrollers TC27x/TC29x/TC37x/TC39x, with full MCDS, offer the capability of recording these events in the trace. To demonstrate these features, Lauterbach showed that consequent analysis and adaption of the code to the processor hardware leads to a significant reduction of the function runtime.

Before Optimization



Function runtimes



Stalls per function

Stalls per Function

To demonstrate the optimization steps, we have written a test function TC1_computeSensorMetrics which performs various calculations on a sensor array. It runs on a TC397XA. For an initial overview, we configured the MCDS so that trace information is generated for core 1 which contains both the program flow and all the stall events which occurred. To get the stall events into the trace, one of the MCDS performance counters has to be set up so that a trace message is generated as soon as at least two stall events are counted. A threshold this small allows the recorded stall events to be assigned to the function causing it as accurately as possible.

For large projects it is advantageous to use a TRACE32 PowerTrace Serial trace tool for this initial overview. It can record up to 4 GByte of trace information via the AGBT. It also offers the option of streaming the trace information to a file on the host computer and significantly increasing the recording time. If only the 2 MB on-chip trace is available, several test runs have to be performed and accumulated in order to get all the required trace data for the initial overview. After the measurement is completed, a function runtime statistic as well as a statistic that shows the number of stalls per function can be displayed (see the two screenshots in the showcase above).

Stalls in Detail

Functions can now be selected individually to be examined in detail. In order to optimize these functions, it is necessary to investigate the causes of the stalls. For our example, we investigated whether the stalls were caused by instruction cache misses, incorrectly predicted branches or data cache misses. »

The MCDS provides up to 16 performance counters for this kind of in-depth diagnosis.

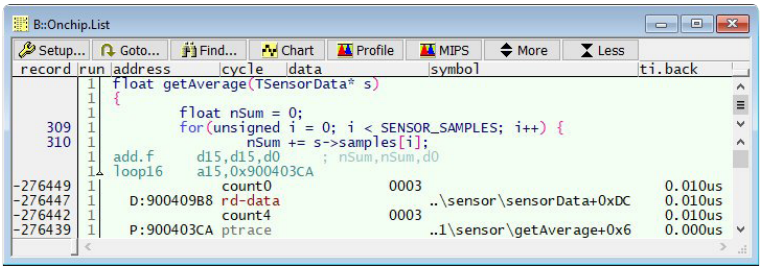
To be able to carry out the necessary optimizations quickly it is important to identify which instructions are causing the individual stalls. The classic method of reconstructing the program flow based on the trace messages for all branches is not fine-grained enough for this objective. The MCDS has to be set up in such a way that it generates a program trace message per MCDS clock. Infineon calls this SYNC trace.

Let's recap the whole thing.

- More event messages are needed for an in-depth diagnosis.
- More program trace messages are needed to identify the offending instructions.
- Additional trace messages like a read address message might also be required for a detailed analysis.

Our tests have shown that such a measurement generates so many trace packets that the AGBT port is almost always overloaded. Therefore the on-chip trace must be used. To make optimum use of the available on-chip trace memory, it is recommended that the MCDS is configured so that both the SYNC trace messages and event trace messages are only generated for the function currently being examined. This is possible by simply setting the appropriate breakpoints.

In our test example we proceeded step by step. First, we examined the data cache misses and optimized our test function so that it used the data cache more efficiently.

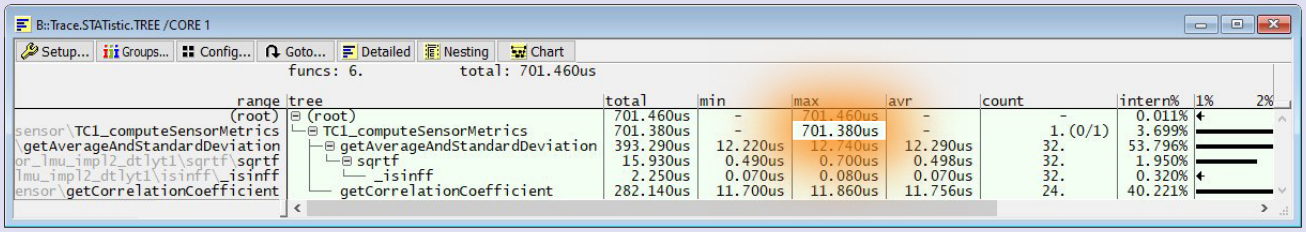


Next, we examined the instruction cache misses and the incorrectly predicted branches with the goal of reducing the pipeline stalls to the unavoidable minimum. We verified the result of our optimizations again and again by means of the TRACE32 statistic commands. In the end, we were able to reduce the number of stalls for our test function TC1_computeSensorMetrics from 57981 to 19479 and thus significantly improve the runtime of the function (see the showcase below).

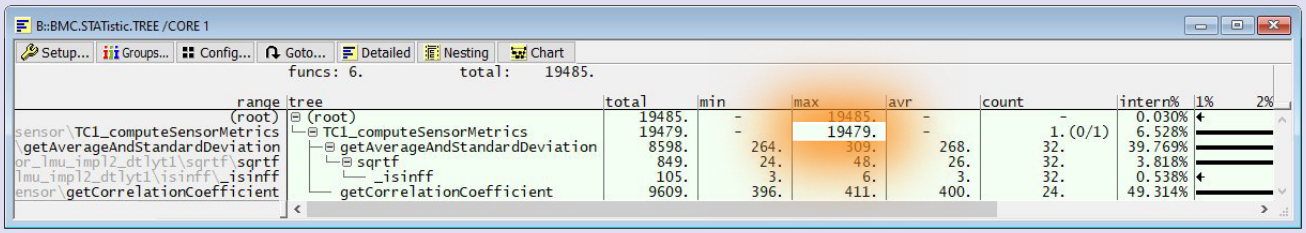
Conclusion

The possibility of recording the program execution and the performance counters in the trace enables new options for optimizing function runtimes. To be able to optimize code at this level highlights the need for engineers to have a good working knowledge of the underlying processor hardware and features. To use these new options, a TRACE32 Release Software of at least 09/2020 is needed. This also contains our test example bmc_trace_demo.cmm in the demo directory. As always, our tools and analysis techniques and options are adapted and expanded based upon customer feedback and demands.

After Optimization



Function runtimes



Stalls per function