TASKING Embedded Profiler

For Infineon AURIX Devices

AN ALTIUM BRAND

AN AFFORDABLE PROFILER FOR EVERY DESKTOP

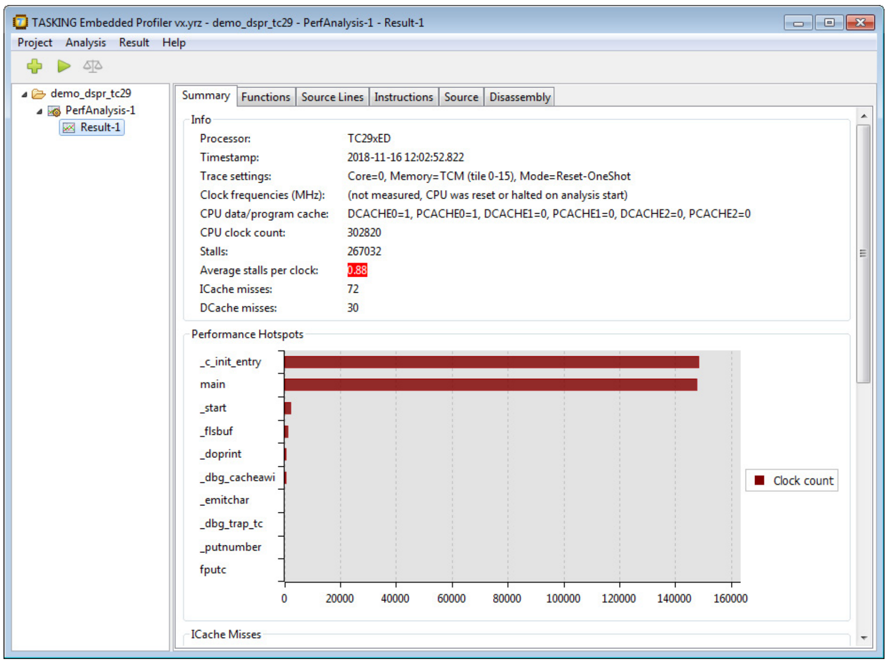
The TASKING® Embedded Profiler enables software developers of any experience level to fully utilize the processing capacity and memory resources of Infineon® AURIX™ devices and eliminate performance-killing stalls. Developed jointly with Infineon Technologies, the profiler's Smart Profiling Technology™ uses expert-level knowledge about AURIX inner-workings built into the tool. The TASKING Embedded Profiler is a 64-bit application to ensure fast operation and the handling of large trace files.

The tool features an intuitive graphical user interface (GUI) that uses information from the on-chip performance counters and symbolic information taken from the binary file to guide you to performance bottlenecks. It shows the cause of the bottleneck and provides concrete suggestions to mitigate the exposed performance issue. This wizard-like interface enables developers without expert-level knowledge of the AURIX architecture to fix the issue and squeeze the maximum performance out of the device to meet the timing requirements imposed on the function, software component, or system.

The predominant value of the TASKING Embedded Profiler is its ability to convert the massive amount of raw data into meaningful advice about how to optimize the code. The GUI immediately shows the overall system performance and the hottest functions with associated execution time and jitter, as well as the amount of instruction- and data-cache misses and pipeline stall-cycles, in both absolute and relative numbers. With a mouse click you can zoom in on the performance data, moving from system, to software component, to function, to C-statement, and finally to machine instruction scope to find the root cause of the performance issue.

Product Features and Benefits

- Developed jointly with Infineon Technologies, expert-level knowledge about AURIX™ inner-workings are built into the tool
- Affordable, non-intrusive profiler
- Allows novices and experts to fully exploit the performance potential of AURIX devices
- Identifies location and nature of inter- and intra-core performance bottlenecks
- Covers behavior of instruction pipelines, instruction- and data-caches, and memory system
- Intuitively zeroes-in on code fragments that cause performance issues
- Fits into existing software development and test flows
- Prevents use and costs of over-dimensioned hardware



Processor:	TC29xED
Timestamp:	2018-11-16 12:02:52.822
Trace settings:	Core=0, Memory=TCM (tile 0-15), Mode=Reset-OneShot
Clock frequencies (MHz):	(not measured, CPU was reset or halted on analysis start)
CPU data/program cache:	DCACHE0=1, PCACHE0=1, DCACHE1=0, PCACHE1=0, DCACHE2=0, PCACHE2=0
CPU clock count:	302820
Stalls:	267032
Average stalls per clock:	0.88
ICache misses:	72
DCache misses:	30

Figure 1: The TASKING® Embedded Profiler summary page identifies performance issues

The GUI intuitively guides you towards the root cause of performance issues such as cache misses resulting from inefficient data-structure design or trashing due to inefficient task schedules. Execution pipeline stalls can be caused by programming constructs that inhibit compiler optimizations, branch mis-predictions, or memory access delays resulting from inter-core interactions or sub-optimal data-layout in memory.

Deployment of the profiler in a software test environment is supported via a command-line batch mode. Typical use cases for the profiler include measuring and monitoring the performance characteristics of legacy code and newly developed software during unit and (continuous) integration testing. It can also help assure that the software will meet its timing requirements and system resource usage budget. Once a performance issue is detected, the profiler assists you in analyzing and fixing it.

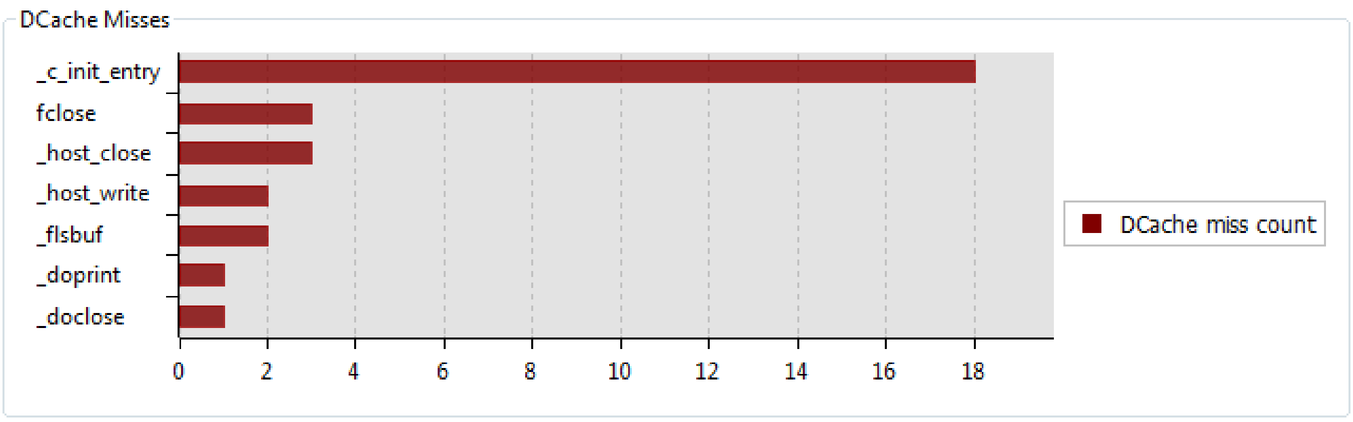


Figure 2: Drilling down from Figure 1 shows a prioritized list of the most substantial sources of stalls.

The TASKING Embedded Profiler is especially suited for use in mass-production environments where an early focus on software performance pays-off in terms of reduced hardware costs, or the opportunity to expand the feature set to increase the value of your system.

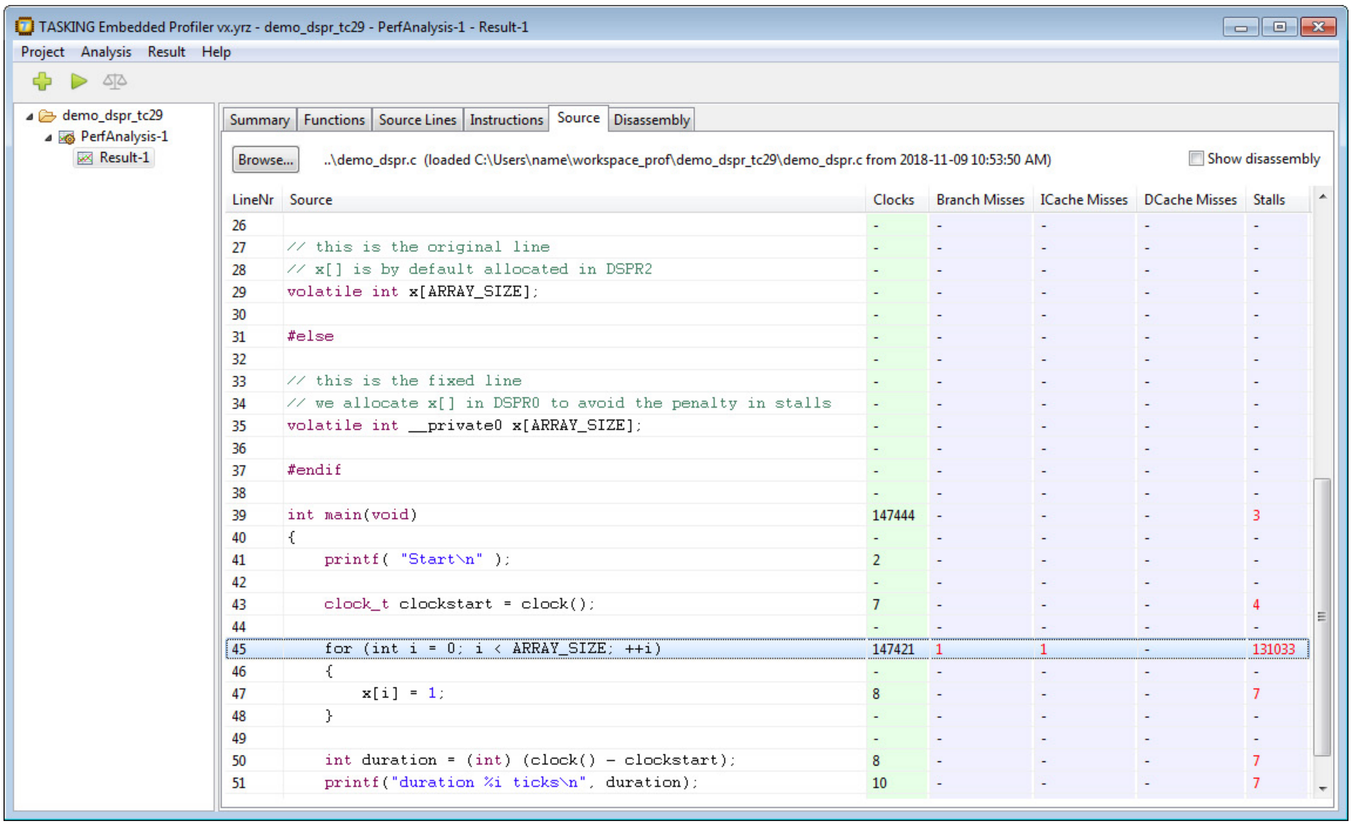
The profiler is especially suited for use in mass-production environments where an early focus on software performance pays off in terms of reduced hardware costs, or the opportunity to expand the feature set to increase the value of your system.

COLLECTING AND ANALYZING PERFORMANCE DATA

The TASKING Embedded Profiler can be configured to collect performance data at different levels of granularity. At the most dense level, the following data is acquired per machine instruction executed:

- Number of clock cycles
- Instruction- and/or data-cache miss events
- Branch-prediction failure events
- Number of stall-cycles due to memory access delays or pipeline hazards.

Performance data can be gathered for either the whole application or for selected functions only. Performance data is acquired in either a truly non-intrusive "one-shot mode" where the data acquisition stops once the trace buffer is full, or in "continuous" mode in which case the application can be slowed down once performance data buffers tend to overflow and are transferred from the target to the host system.

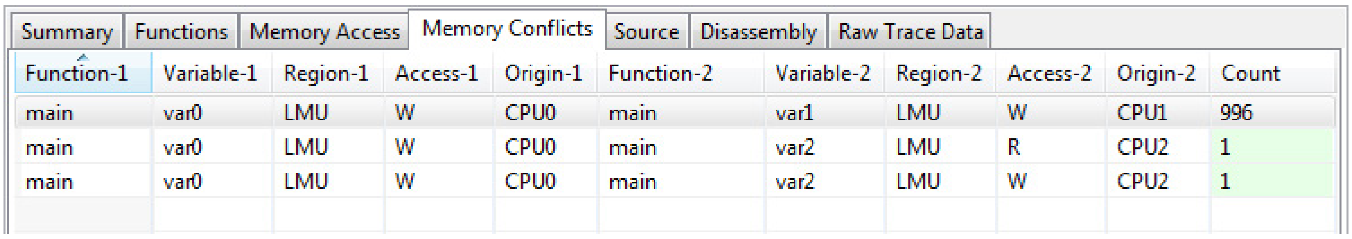


..\demo_dspr.c (loaded C:\Users\name\workspace_prof\demo_dspr_tc29\demo_dspr.c from 2018-11-09 10:53:50 AM)

LineNr	Source	Clocks	Branch Misses	ICache Misses	DCache Misses	Stalls
26		-	-	-	-	-
27	`// this is the original line`	-	-	-	-	-
28	`// x[] is by default allocated in DSPR2`	-	-	-	-	-
29	`volatile int x[ARRAY_SIZE];`	-	-	-	-	-
30		-	-	-	-	-
31	`#else`	-	-	-	-	-
32		-	-	-	-	-
33	`// this is the fixed line`	-	-	-	-	-
34	`// we allocate x[] in DSPR0 to avoid the penalty in stalls`	-	-	-	-	-
35	`volatile int __private0 x[ARRAY_SIZE];`	-	-	-	-	-
36		-	-	-	-	-
37	`#endif`	-	-	-	-	-
38		-	-	-	-	-
39	`int main(void)`	147444	-	-	-	3
40	`{`	-	-	-	-	-
41	`    printf( "Start\n" );`	2	-	-	-	-
42		-	-	-	-	-
43	`    clock_t clockstart = clock();`	7	-	-	-	4
44		-	-	-	-	-
45	`    for (int i = 0; i < ARRAY_SIZE; ++i)`	147421	1	1	-	131033
46	`    {`	-	-	-	-	-
47	`        x[i] = 1;`	8	-	-	-	7
48	`    }`	-	-	-	-	-
49		-	-	-	-	-
50	`    int duration = (int) (clock() - clockstart);`	8	-	-	-	7
51	`    printf("duration %i ticks\n", duration);`	10	-	-	-	7

Figure 3: Drilling further down from Figure 2 identifying the C statements that caused the DCache misses/stalls.

Small local changes to the software can cause far-reaching interferences, especially in complex multi-core systems. An attempt to fix a performance issue may result in a local improvement, but (unforeseen) side effects can have negative impacts in other places. Therefore, the profiler offers a means to compare the analysis results before and after applying a code change. This enables you to see and understand the effects of your optimization efforts, including its undesired side effects, and act accordingly.



Function-1	Variable-1	Region-1	Access-1	Origin-1	Function-2	Variable-2	Region-2	Access-2	Origin-2	Count
main	var0	LMU	W	CPU0	main	var1	LMU	W	CPU1	996
main	var0	LMU	W	CPU0	main	var2	LMU	R	CPU2	1
main	var0	LMU	W	CPU0	main	var2	LMU	W	CPU2	1

Figure 4: The Memory Access Conflicts tab identifies when two variables from different

Developers can also choose to display raw trace data. Raw trace data is useful, for example, to understand why stall-cycles are assigned to instructions that do not access memory. This can be the case when an instruction is the target of a branch. The Raw Trace Data tab has a search field you can use to search the address column.

TARGET DEVICE RESOURCE USAGE

The TASKING Embedded Profiler uses the Multi-Core Debug Solution (MCDS) for on-chip trace support. The following on-chip trace memory types are supported:

- Trace Calibration Memory (TCM)
- Extended Trace Memory (XTM)
- miniMCDS Trace Memory (TRAM)

DEVICE SUPPORT

The TASKING Embedded Profiler can be used in combination with TriCore™/AURIX emulation devices and with production devices equipped with a mini-MCDS. The following devices are supported:

- TC23xED
- TC26xED
- TC27xED
- TC29x
- TC29xED
- TC38x
- TC39xAED
- TC39xED

The profiler connects to the target via the 64-bit Device Access Server (DAS) driver (v7.0.6) and Device Access Port (DAP) miniWiggler.

HOST PLATFORMS

Microsoft Windows* 64-bit

SUMMARY

The TASKING Embedded Profiler is a cost-efficient tool that enables non-expert users to fully exploit the computational and memory resources of Infineon AURIX devices. Most embedded software must comply with strict timing requirements. Integrating the profiler into your software development and test flow offers a means to address timing and performance issues immediately on first occurrence. As a result, either the cost of the hardware used can be reduced, or additional features can be implemented without increasing the hardware cost.