Thread Execution Analysis with Trace

{{#switchcategory:MSP430=<McuHitboxHeader/>|C2000=<McuHitboxHeader/>|Stellaris=<McuHitboxHeader/>|TMS570=<McuHitboxHeader/>|MCU=<McuHitboxHeader/>|MAVRK=<MAVRKHitboxHeader/>|<HitboxHeader/>}}

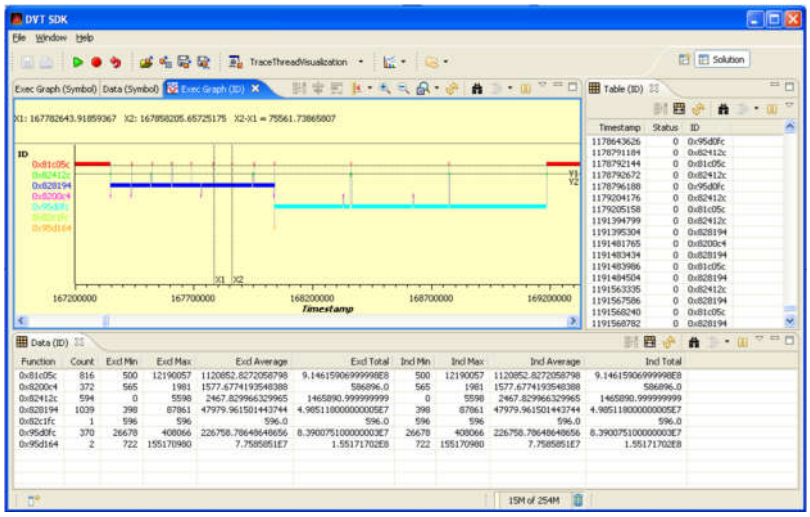
Contents

Thread Execution Analysis

Overview

Thread Execution Analysis is a method of using XDS560 Trace to create a cycle accurate representation of an applications execution at the thread level. From the data gathered from this analysis, we can see the thread duration, along with the order that the threads execute. We can diagnose cases where some threads may be starved of execution time, or where priority issues might cause a task to not be executed at all.

The Thread Execution Analysis script generates a text output that can be displayed in a graphical package, or processed by another script. A graphical display is shown at right.



Methodology

Thread Execution Analysis is implemented by capturing a thread Id and a timestamp each time a thread context switch occurs. This requires a slight modification tot he application. A hook function needs to be used that can be called by the Operating System each time thread context is switched. The function needs to write the id of the new thread to a global variable. All writes tot he global variable are then captured via trace with an associated time stamp. The output generated by the script outputs a timestamp with an associated thread ID for every context switch.

Capturing Thread Execution Analysis Data

Configure trace in the Trace Control menu as desired. Typically, "Stop on Buffer Full" mode is used with Thread Execution Analysis, but it can also be used with "Circular Buffer" mode. If using an XDS560T, configure the desired trace buffer size. A larger buffer will capture much more data, but the data will take a longer time to process. A smaller buffer won't get nearly as much of the application, but will be post processed very quickly. The compression of data when capturing via this method will not yield nearly as much data as in some other cases, so the penalty for using a very large trace buffer will be much smaller.

The Unified Breakpoint Manager (UBM) Plugin in Code Composer Studio can be easily configured to capture Thread Execution Analysis data

- From the Breakpoint window, create a new Trace job.
- Edit the properties of the job, and select Trace Type->Standard, Actions->User Script, and Script Type->Thread Aware Profiling.
- Expand the Script Type Option
- Set the Memory Write address to the global memory location where the Thread ID is getting written (This can be a numeric or symbolic address)
- Specify the access sizes that you want to capture. Typically leaving them all selected is fine.
- Click OK to save the configuration and ensure that the job is enabled in the breakpoint window.
- Run the application. You should see trace data being captured in the Trace Display Window.

Decoder Prerequisites

In order for the Thread Execution Analysis application to be able to process the data, the following fields must be in the data passed to the script

- Write Data
- Trace Status
- Cycles

Command Line

One of the following commands can be used to process the trace data. Note that full or relative paths to each file supplied must be provided if all files are not in the current directory.

td.exe XDS560_RecTraceData.bin -app <out file> <trace_decoder_options> | perl trace_thread_execution_log.pl -c=<trace/code> -n

or

-bin XDS560_RecTraceData.bin -app <out file> <trace_decoder_options> | trace_thread_execution_log.exe -c=<trace/code> -n

Trace Script Download

The script package can be downloaded at the following location https://www-a.ti.com/downloads/sds_support/applications_packages/trace_csv_scripts/index.htm

Note that in order to use the stand alone trace decoder, you must have version 3.0.0 of the scripting package or later.

Trace Script Frequently Asked Questions

Generic Scripting FAQS

- Q: Why do I sometimes see "UNKNOWN", "UNKNOWN" in the output for functions/filenames
  - A: The function/filename symbols are determined from the output of the ofd6x.exe (Object File Dump utility), which generates a list of functions and filenames from the .out file, along with their staring and ending addresses. If "UNKNOWN" values are showing in the output, it's because trace captured program execution that had a program address outside the ranges specified by the OFD utility. Common causes might be code that has been dynamically loaded/allocated which wouldn't have associated information in the .out file. You can usually determine the exact cause by looking in the file generated by ofd6x.exe and the .map file and determining why the offending program address is not defined.

Thread Aware Profiling Specific FAQS

{{ 1. switchcategory:MultiCore= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **Thread Execution Analysis with Trace** here.	Keystone= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **Thread Execution Analysis with Trace** here.	C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **Thread Execution Analysis with Trace** here.*	*For technical s please post yo questions at http://e2e.ti.cor Please post on comments abo article **Thread Execution An with Trace** her* }}

Links

Amplifiers & Linear	DLP & MEMS	Processors	Switches & Multiplexers
Audio	High-Reliability	• ARM Processors	Temperature Sensors & Control ICs
Broadband RF/IF & Digital Radio	Interface	• Digital Signal Processors (DSP)	Wireless Connectivity
Clocks & Timers	Logic	• Microcontrollers (MCU)	
Data Converters	Power Management	• OMAP Applications Processors	

{{#switchcategory:MSP430=<McuHitboxFooter/>|C2000=<McuHitboxFooter/>|Stellaris=<McuHitboxFooter/>|TMS570=<McuHitboxFooter/>|MCU=<McuHitboxFooter/>|MAVRK=<MAVRKHitboxFooter/>|<HitboxFooter/>}}