Application Note: AN00197

Getting Started with GPROF in xTIMEcomposer Studio

This application note shows how to get started with GPROF using the xTIMEcomposer studio. It shows you how to run an application with GPROF profiling enabled. It then explains how to analyze the resulting profile using the built-in profiling view.

To get started, simply double click on *Getting Started with GPROF in xTIMEcomposer Studio* in the Examples view, and click finish in the resulting import dialog. The sample project will then be imported and you will be switched to the *XMOS Edit* perspective. The getting started pdf is then accessable from the *doc/pdf* folder at the top level of the imported project.

Required tools and libraries

- xTIMEcomposer Tools - Version 14.0

Required hardware

This application note is designed to run on any XMOS multicore microcontroller or the XMOS simulator.

Prerequisites

None

1 Overview

1.1 Introduction

Our comprehensive development tools suite provides everything you need to write, debug and test applications based on xCORE multicore microcontrollers. The full xTIMEcomposer tool set includes unique capabilities such as the xSCOPE logic analyzer and XMOS Timing Analyzer, that let you get the best performance from the deterministic xCORE architecture. With our collection of libraries and examples, it's easy to create and deliver xCORE applications.

xTIMEcomposer features:

- Eclipse graphical environment + plus command line tools
- LLVM C, C++ and xC compilers
- xDEBUG: GDB multicore debugger
- xSIM: Cycle accurate simulator
- xSCOPE: In-circuit instrumentation + real-time logic analyzer
- XTA: Static timing analysis
- Multiple platform support: Windows, OS X, Linux
- Enterprise/Community editions: Tools support for everyone

This application note shows how to get started with GPROF using the xTIMEcomposer studio. It shows you how to run an application with GPROF profiling enabled. It then explains how to analyze the resulting profile using the built-in profiling view.

2 Getting Started

Ensure you are in the edit perspective by clicking on the *Edit* perspective button on the left hand side toolbar.

In the *Project Explorer* view you will see the *app_getting_started_with_gprof* project. Open *src/main.xc* by double clicking.

This application calls 2 functions: *functionA* and *functionB*. Each function consists of a loop, and the iterations are set such that *functionA* will take roughly 10 times longer to execute than *functionB*.

2.1 Build the application

To build the application, select 'Project -> Build Project' in the menu, or click the 'Build' button on the toolbar. The output from the compilation process will be visible on the console.

2.2 Create and launch a run configuration

To create/view a GPROF profile you first need to create a run configuration. The xTIMEcomposer allows multiple configurations to exist, thus allowing you to store configurations for running on different targets/with different runtime options and arguments.

Right-click on the generated binary in the *Project Explorer* view, and select *Run As -> Run Configurations*. In the resulting dialog, double click on *xCORE Application*, then perform the following operations:

- On the *Main* tab select the desired target, or alternatively check the *simulator* option.
- On the *Simulator* tab, check *Enable Gprof output*.

Now select the *Run* button to launch the application. On completion, a number of *.gprof profiling files will be written to the top level of the project, one for each logical core present in the system. The perspective will automatically switch to *GProf Profiling*, and a dialog will appear listing the avaiable *.gprof files.

- Select *tile[0]_core0.gprof* and click *Open*.

You will then be prompted to choose the binary corresponding to this run. The correct binary should be selected by default.

- Select *OK*.

The *gprof* view will then be populated with the profile for logical core 0, which in this case corresponds to the main thread of execution.

Expand the *app_getting_started_with_gprof.xc* node in the tree. This will reveal the time spent in each function of this application during execution. As expected, roughly 90% of the time is spent in *functonA*, and 10% in *functionB*.

Expand the *functionA* node in the tree. This will reveal the time spent in each line of this function. Double-click on the top line. This will bring up the corresponding line in the editor. As you can see, roughly 60% of the time of this function is spent performing the loop management.

3 References

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide

4 Full source code listing

4.1 Source code for main.xc

```
// Copyright (c) 2015, XMOS Ltd, All rights reserved

int functionA() {
    int result = 0;
    for (unsigned int i = 0; i < 10000; ++i) {
        ++result;
    }
    return result;
}

int functionB() {
    int result = 0;
    for (unsigned int i = 0; i < 1000; ++i) {
        ++result;
    }
    return result;
}

int main() {
    int total = 0;
    total += functionA();
    total += functionB();
    return 0;
}
```