Application Note: AN00193

Getting Started with Debugging in xTIME-composer Studio

This application note shows how to get started with debugging using the xTIMEcomposer studio. It shows you how to create and launch a debug configuration, set breakpoints, single step and inspect variable values during execution.

To get started, simply double click on *Getting Started with debugging 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 binary analysis using the xTIMEcomposer studio. It shows you how to open a binary in the binary analysis tool, browse the resource usage, modify the source code and see the change reflected in the analysis output.

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_debugging* project. Open *src/-main.xc* by double clicking.

This application contains 2 tasks, a producer task and a consumer task, each connected via a channel. The producer sends data to one end of the channel, and the consumer reads the data from the other end and prints it to the console.

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 Insert some breakpoints

Before you start insert breakpoints on the following lines:

Breakpoint (1) In function **producer**

```
c <: i;
```

Breakpoint (2) In function **consumer**

```
printintln(data);
```

A breakpoint is inserted by double clicking in the left-hand side gutter of the corresponding line.

2.3 Create and launch a debug configuration

To start debugging you first need to create a debug 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 *Debug As -> Debug Configurations*. In the resulting dialog, double click on *xCORE Application*. On the *Main* tab, select the required target, or check the *simulator* option if you have no hardware connected.

Now select the *Debug* button to launch the application. The perspective will automatically switch to the debug perspective, and execution will stop on the breakpoint (1) in **producer**.

The current location can be seen in in the *Debug* view. This view displays a list of all the currently running tasks, and show the backtrace for each, the current tasks backtrace being expanded by default.

Now remove the breakpoint on the current line by double clicking on the breakpoint icon in the gutter, the click the *Resume* button in the main toolbar. Execution will be continued until you hit breakpoint (2) in **consumer**.

Now remove breakpoint (2). Click on the *Step Over* button in the toolbar. You'll see the '0' appear on the console.

Hover over the *data* variable. The popup shows it current value. Alternatively, the values of all the locals/globals can be seen in the *Variables* view.

Resume execution. You'll see the remainder of the output in the *Console* view.

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) 2016, XMOS Ltd, All rights reserved

#include <print.h>

void producer(chanend c) {
    for (unsigned int i = 0; i < 10; ++i) {
        c <: i;
    }
}

void consumer(chanend c) {
    int data = 0;
    c :> data;
    printintln(data);
}

int main() {
    chan c;
    par {
        producer(c);
        consumer(c);
    }
    return 0;
}
```