Application Note: AN00191

Getting Started with Binary Analysis in xTIMEcomposer Studio

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.

To get started, simply double click on *Getting Started with binary analysis 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

None

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.

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 Load the binary into the binary analysis perspective

In the *project explorer* view, expand the *Binaries* node and double click on the binary (*.xe) file. This will load the binary into the analysis tools and switch perspective to the most recently used tool. For the purposes of this tutorial, ensure that the *Analyze Binary* button in the toolbar is selected.

Alternatively, click on the *Analyze* button in the left-hand toolbar, then ensure that the *Analyze Binary* button in the toolbar is selected. The *Load binary* toolbar button can then be used to load the required binary.

2.3 Overview of the available information

The *Binary* view consists of 4 tabs:

- **Resource Usage**
  Provides a breakdown of the overall and per tile resource usage
- **Function Table**
  Lists the global functions, addresses, sizes and stack usage
- **Data Table**
  Lists the global data objects, addresses and sizes
- **I/O Table**
  Shows the port/xlink to pin mapping for the package targeted
- **Call Graph**
  Shows the functions call graphs for each of the tiles

Note that for this application, the overall reported number of logical cores used is 1.

2.4 Update the application code

Return to the *Edit* perspective by selecting the *Edit* button in the left-hand toolbar. In the *project explorer* view, open *src/main.xc*, then uncomment the following lines:

```
//par {
//    printA();
//    printB();
//}
```

Rebuild the application, and reload the resulting binary into the analysis tool. Note that as expected, the resource usage reported for the number of logical cores used has now increased to 2.

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 printA() {
    printstrln("A");
}

void printB() {
    printstrln("A");
}

int main() {
    //par {
    //    printA();
    //    printB();
    //}
    return 0;
}
```