KTH Information and
Communication Technology

IL2212 EMBEDDED SOFTWARE

Laboratory 1
Communication in a Bare-Metal Multiprocessor

January 29, 2016

1 Objectives

The laboratory introduces a multiprocessor soft-core that will be used during this laboratory course. Students will receive the necessary design files for the processor and initial scripts to download the multiprocessor and the software for each core to an FPGA board.

During this laboratory students shall

- identify the architecture of the multiprocessor analyzing the provided Altera design files
- investigate possibilities to optimize the memory footprint of an application
- implement an application that demonstrates communication between the different cores and I/O-handling
- write a short report presenting the features and limitations of the processor and possible mechanisms for interprocessor communication

Students should take this laboratory very serious, since laboratory 2 will require a very good understanding of the same multiprocessor.

2 Deadlines

Students have to respect the following deadlines:

1. The finalized report has to be submitted as PDF-file not later than February 8, 12:00. See the detailed instructions on the course web.

2. The demo-application needs to be finalized before the first laboratory session so that it can be demonstrated already at the beginning of the laboratory session.

3 General Guidelines

Read the entire laboratory manual in detail before you start with the preparation tasks. Complete the preparation tasks before your lab session in order to be allowed to start the laboratory exercises.

It is very important that students are well-prepared for the labs, since both lab rooms and assistants are expensive and limited resources, which shall be used efficiently. The laboratory will be conducted by groups of two students. However, each student has to understand the developed source code and the preparation tasks. Course assistants will check that students are well-prepared.

Note: All program code shall be well-structured and well-documented. The language used for documentation is English.

4 Preparation Tasks

4.1 Installation

1. Please use the virtual machine that has been prepared for the course. The exact installation instructions are available at the course web page.

2. There is a `git`-repository on the KTH Github server including the soft core and scripts and source files to run a few examples. Add this repository as a remote to your private repository using the instructions on the course web.

3. The repositiory includes software examples for the multiprocessor in the `app/`-folder. Study and run the scripts for the examples `hello_world`, `hello_ucosii` and `hello_mpsoc` inside the Nios 2 Shell (accessible from the Desktop).

   NOTE: You do not need to synthesize the hardware. The required files (`*.sof`, `*.sopcinfo`, `*.jdi`, `*.qsys`) are already available in the `hardware/`-folder of the git-repository!

4.2 Multiprocessor Architecture

The first step is to identify the architecture of the multiprocessor, which is described in a QSYS-file with the suffix `*.qsys`. To do this use the QSYS-tool, which is available via Quartus. In the QSYS-tool you can see the cores, peripherals and their interconnection.

4.3 Code Optimization

The multiprocessor has very limited memory resources in order to fit in the Altera DE2 FPGA board. Thus, for efficient programs, code size optimization is essential. The scripts available in the repository provide already a few optimizations, but additional optimzations should be investigated to further reduce the code size. Study the Altera documentation to understand which optimizations are available and how they can be included in the provided scripts.

4.4 Demo Application

One important deliverable of this laboratory is to design and implement a demo-application. This demo-application shall

- demonstrate that processors can communicate with each other. You should use all the architectural features identified for passing messages between processors (e.g. shared memory).
- use the performance counter to measure execution time
- demonstrate all I/O-peripherals, but it is not required to use interrupt from the I/O-units
- be optimized for code size to achieve a very small memory footprint in particular with respect to the onchip-memories.

As a starting point, take a look at the `hello_mpsoc` application and the corresponding script `run_hello_mpsoc.sh` and understand what they do. Then, develop an own idea for an application and implement it. You are expected to change the functionality on all processors.

4.5 Technical Specification

Students shall deliver a technical specification of the multiprocessor in an electronic document. The following requirements apply to the document. The document shall

- present the features and limitations of the multiprocessor
- include a diagram illustrating the architecture of the multiprocessor with all peripherals and their interconnect. It shall also include the types and sizes of the memories.
- suggest mechanisms for passing messages between processors safely.
- use the IEEE double column format and should have three to four pages. A link to Word and LaTeX-templates for this format is available on the course web page.

- include a table about the memory footprint of the compiled source code on each CPU.
- describe the demo-application in terms of functionality, performance and cost.

5 Laboratory Tasks

During the laboratory, students will

- demonstrate their applications
- answer questions on
    1. their report on the technical specification
    2. the multiprocessor architecture
    3. the demo-application including code optimization techniques.

6 Examination

In order to pass the laboratory the student must

- have completed the preparation tasks of Section 4 before the lab session
- have demonstrated the preparation tasks in a convincing way for the laboratory staff
- have delivered a well-written and informative technical specification