Embedded Systems Laboratory
apl. Prof. Dr.-Ing. Dominik Stoffel
Dipl.-Ing. Thomas Fehmel

Technische Universität Kaiserslautern
Fachbereich Elektrotechnik und Informationstechnik
Entwurf informationstechnischer Systeme

Assignment 1

The first warm-up assignment is aimed at giving an overview of the used platform and the supplied tools. Documentation is supplied to get familiar with the platform. In addition, all sources are open and can be inspected at will.
As you will notice, the hardware design is written in VHDL (Very High Speed Integrated Circuit Hardware Description Language). Since this is an advanced lab, we do not deem it necessary to provide an introduction to VHDL or teaching it. You may write your designs in Verilog. However, the supervisors and lab assistants will most likely not be able to help you when any problems related to Verilog arise.
The used platform is called the *LT16 System on Chip*, which is built around the *LT16x32* processor. The processor uses its own unique instruction set. For programming, consult the LT16x32 Platform Manual, which gives an overview and functional description of the used assembly language. The assembly language can be translated with a provided assembler program (see below). Every student should be aware that a certain knowledge of assembly language *in general* and programming skills are necessary to complete this lab.
It should be noted that it is specifically *not* the task of the lab assistants and supervisors to teach the necessary skills in programming and basic hardware design. These skills are considered prerequisites to the lab.

Question 1.1

The first task is to set up the working environment for the lab exercises on your lab PC, laptop or home PC. The lab PCs use a Linux based operating system, so all following descriptions assume a Linux environment. If you wish to work within your own Windows or Mac environment, you are free to do so. The lab assistants and supervisors will not provide support for problems with non-lab hardware and software.

Git

Git is a version control system[1] used to store and track the development of the System-on-Chip.
Your lab account is supplied with the `ssh`-key needed to access the remote repository. Open a terminal in your home directory, and use the command

```
$ git clone -b esylab<GROUP NUMBER> \
    git@bordeaux.eit.uni-kl.de:lt16lab-<GROUP NUMBER> lt16lab
```

to get an initial copy of the repository. The `<GROUP NUMBER>` is a two digit number associated with your group. This creates a local copy of the repository, and checks out the designated branch for your group, in a directory `lt16lab` (relative to the place the command was executed).
There is also a branch called `master`, which holds an unmodified version of the “vanilla” SoC. You will find that you do not have write permission on the master branch, only on your assigned group branch.
During your work in the lab, you can keep track of your changes by using git. The command

```
$ git add <file> [<file> ...]
```

lets you add changed files to be committed. This is also called "staging".
Once you have added the modified files you wish to stage in a commit, the command

```
$ git commit
```

will initiate the commit process. If none was given via the command line, an editor will open to prompt you to enter a commit message. The default way to do this is to have a short commit title in the first line, and a longer description of what was committed in the consecutive lines. Git will not accept a commit without or an empty message.
To learn how to further use git, i.e. undoing changes made to files, rewinding commits, etc. consult the on-line resources for git [2]. If there is an update to the system, it will be distributed via a commit in the `master` branch.

[1] `http://git-scm.com/`
[2] `http://git-scm.com/`

You can incorporate these changed by performing a `merge`. Before performing a merge for the first time, read the git manual on the topic.
If you want to clone the git repository on your home PC or laptop, you must copy the `ssh`-key from `~/.ssh/` directory.

a) Your task is to clone the repository, and make sure you have the branch which is named after your lab account checked out. Create and edit a file `group_members` in the projects root directory. Insert the names (and only the names) of your group members into this file, separated by newlines. Then, add and commit the file. Afterwards, run the command

```
$ git push
```

to transfer your changes to the remote repository.

Documentation & Assembler

The documentation for the platform manual is provided as latex and scalar vector graphics files. A Makefile to compile them into a convenient pdf is provided.
The platform processor LT16x32 uses a custom assembler, which is also provided as source.

b) Enter the subdirectory `documentation` in the project root and execute the command

```
$ make
```

to assemble the manual `soc.pdf` from the latex files. Follow the instructions in the manual to build the executable assembler. Build all provided sample programs in the `<project root>/programs/` directory by executing the `make` command in that folder.

Question 1.2

The goal of this task is to make you familiar with the structure of the ESyLab-Soc platform and the use of the hardware design tools.

The Platform

The ESyLab-SoC is a lightweight embedded platform, consisting of a processor core, the LT16x32, a Wishbone bus interconnect module, an interrupt controller, memory controllers and peripheral I/O controllers.
A description of the system is given in the platforms manual. Read it to become acquainted with the structure of the top-level entity.

Xilinx ISE

The Xilinx ISE Design Suite is a collection of tools to simulate and synthesize hardware designs for Xilinx FPGA targets. It comes with an integrated editor, which need not neccessarily use. The ISE design suite is available on the Xilinx webpage (`http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html`), with a 'free' student license to use it. Feel free to download and install it on your personal Laptop/PC.
We will not give an introduction into the suite here, but expect you to make yourself familiar with it on your own. There are tutorials, both written and video, available on the Internet, we will not officially recommend any specific one.

a) Create a directory `ise` in your home directory.
Execute the command

```
$ ise
```

in a new terminal. Create a new project inside the `~/ise/` directory.
Now, load all VHDL sources of the platform from the `soc` directory in the git project root to the ISE project. Do this by using "Add Source" (do not use "Add Copy of Source") in the ISE integrated development environment. this way, your changes can be tracked with git.
For the elaboration of your design it is necessary that the `.ram` file which the instruction memory is configured to load is present under the path it is was configured for. To ensure this, open a terminal in the directory of your ISE project and use the commands

```
$ ln -s <path to git project root>/programs/ .
```

to create a symbolic link from the directory with the assembled programs to the ise project directory.

Simulation

The Xilinix ISE features an integrated hardware simulator called ISim. It can be started via the ISE IDE or separately in the terminal console.

b) Select the provided "warmup1.vhd" test bench for simulation and simulate its behavioral model. Run the simulation until termination. Inspect the bus interactions and input/output behavior.
Check the `blinky.asm` program file, which is used by the test bench, to find the reason for the witnessed behavior.
Hint: Check the contents of the processor's register file during simulation.

Synthesis

The process of synthesis in the Xilinx Suite creates logic networks from the design description and the subsequent configuration for the specified target FPGA. This process is performed in several steps.
After synthesis, the iMPACT tool, which is also part of the design suite, is used to load the configuration to the the FPGA-board.

c) If not done already, import `top.ucf` from the `soc/top` directory in the git project root into your ISE project. Switch the design view from Simulation to Implementation and set "lt16soc_top" as top module (if it is not already).
You will notice that the top level entity has a generic named "programfilename" with a default value which is used during synthesis. This generic constant is used to specify to the instruction memory which file is loaded during elaboration into the instruction memory.
In order to change the target program to be written into the instruction memory, there are several ways. The most obvious one is to change either the default value in the entity definition of `top.vhd`, or the value in the instantiation of the instruction memory. Both methods have the downside that this is a change to the source, which is not always desirable.
A third way is offered by the design suite (this is optional):
When the top level entity is selected in the design view, the synthesis process properties (under "Synthesize - XST") can be altered by selecting "Process Properties..." in the context menu of the process title. Change the property display level to "Advanced" and search for the "generics" switch name. Its value can be a list of key-value associations of generics of the top-level entity, separated by a pipe ('|') character. For more information, consult the Xilinx Website (`http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/pp_db_xst_synthesis_options.htm`)
Synthesize the design with the `blinky.ram` program. Load it to the board with iMPACT. Observe the result and demonstrate it to the lab assistant. In addition, be ready to be quizzed about the properties of the platform and SoC.

Question 1.3 - Challenge

Each Warmup has an additional *optional* task, a "*Challenge*". *Challenges* are individual tasks, not group tasks, and should only be worked on if the mandatory part of the assignment has already been completed.

The challenges not only serve to test the participant's coding and design abilities, but also his/her ability to interpret goal formulations and develop solutions without a step-by-step guide.
The *challenge* for this assignment task is to rewrite the `blinky` program to display an alternative animation. What sort of animation is up to the participant. Imagine something like the lights from Knigth-Rider's KITT. The animation should feature at least 8 'frames'.
Recommendation: Lookup tables are a straightforward way to solve such problems.