Embedded Systems Laboratory Technische Universität Kaiserslautern
apl. Prof. Dr.-Ing. Dominik Stoffel Fachbereich Elektrotechnik und Informationstechnik
Dipl.-Ing. Thomas Fehmel Entwurf informationstechnischer Systeme

Assignment 1

The goal of this first warm-up assignment is to give you an overview of the computing platform and the accompanying software tools used in this lab. You will obtain the documentation needed to become familiar with the platform. The hardware and software is provided as source code. You are encouraged to look into it. **The source code is NOT public domain. Please, do not distribute it in any form. Please, do not upload it on github!**
The hardware design is written in VHDL (Very High Speed Integrated Circuit Hardware Description Language). You will have to write VHDL code yourself in this lab. (Note that this is an advanced lab. We do not give an introduction into VHDL, but assume that you have fundamental knowledge in VHDL coding.)
You may also write your designs in Verilog. However, the supervisors and lab assistants will most likely not be able to help you with problems related to Verilog.
The computing 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. Assembly programs can be translated to machine code with an assembler (see below).

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. All the following explanations assume a Linux environment. If you wish to work on a Windows or Mac environment you are free to do so. However, again, the supervisors and lab assistants cannot provide support for these environments.

Git

`Git` is a version control system[1] used to store and track your development of the system-on-chip hardware and software. You will only need a few basic `Git` commands in this lab (see Fig. 1). The following steps will guide you through the steps needed to set up your `git` repository.
The `Git` repository is located on a server in our network, and `Git` utilizes `SSH` to access it, utilizing `ssh`-RSA keys for authentication. 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 obtain an initial copy of the repository. [2] The `<GROUP NUMBER>` is a two digit decimal 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 permissions 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 or new 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 -m"<Commit Message>"
```

creates a commit of your changes and annotates it with your given message. If no message was given via the command line, an editor will open to prompt you to enter a commit message. This is preferable if you want to enter multi-line commit messages to elaborate on changes or additions to your code. The common 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 following lines. Git will not accept a commit without or an empty message.

[1] `http://git-scm.com/`
[2] The “\” escapes the line break so the shell interprets it as one command. When writing the command in one line, do not use it.

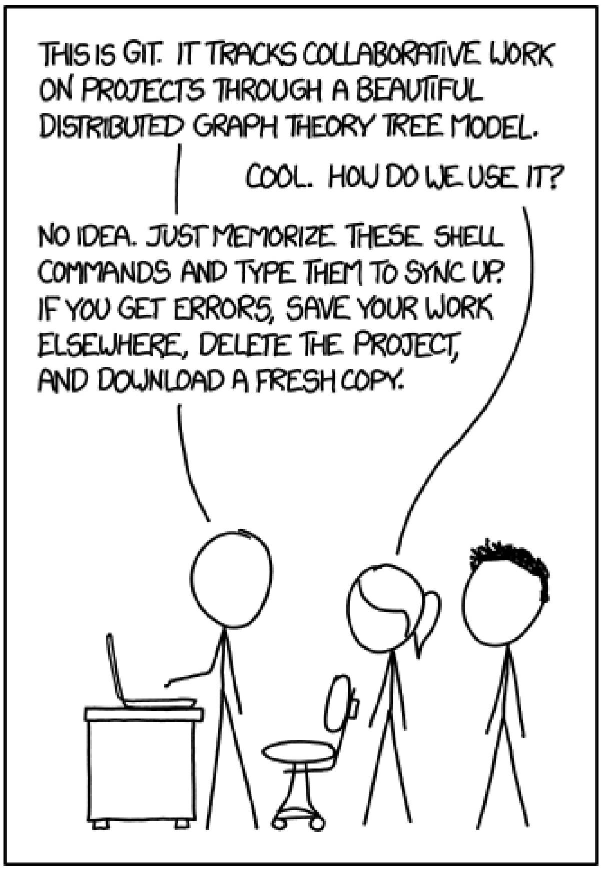
THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Figure 1: Randell Monroe's XKCD on Git

To learn how to further use git, e.g., undoing changes made to files, rewinding commits, etc., consult the on-line resources for git [3]. It may happen that during the semester we need to make updates to the computing platform source code. Such changes will be distributed via commits in the `master` branch, and we will inform you about it. You can incorporate these changes into your own copy 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 of your group account to your own computer.

a) Your task is to clone the repository, and make sure you have the branch which is named after your lab account. Create and edit a file `group_members` in the project's root directory. Insert the names (and only the names) of your group members into this file, seperated by newlines. Then, add and commit the file.
Commits are only created in the local repository. Run the command

```
$ git push
```

to copy your commits to the remote repository.

Documentation & Assembler

The documentation for the platform manual is provided in source code form (LaTeXand SVG files) and as a finished PDF.

[3]Again: `http://git-scm.com/`

The platform processor LT16x32 uses a custom assembler, which is also provided as source code.

b) Open the subdirectory `documentation` in the project's root directory and open the manual `soc.pdf`. Familiarize yourself with its general content, you will need to revisit it later.
Optional: There is a Makefile that you can use to compile the documentation into a PDF file. You can re-compile the manual by entering the `documentation/sources` directory and execute:

```
$ make
```

c) Follow the instructions in the manual to build the assembler executable.
Afterwards, 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.

Computing 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. Please, 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. You are free to use this editor or another editor of your choice.
We will not give an introduction into the suite here. Please, make yourself familiar with it on your own. There are tutorials, in both text and video form, available on the internet.
The ISE design suite is installed on the lab PCs, but it is also available on the Xilinx web page (`http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html`), with a 'free' student license to use it. If you plan to download and install it on your personal computer beware that the download is quite large (many gigabytes).

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

```
$ ise
```

in a new terminal window.
Familiarize yourself with the graphical user interface and 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. To circumvent a problem in the ISE Design suite's behavior, load the `soc/top/top.vhd` separately first, and take care to add sources in the `soc/testbench/` directory for simulation only. Load all sources 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 VHDL 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 was configured for. To ensure this, open a terminal in the directory of your ISE project and use the command

```
$ 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. View and study the bus interactions and the 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 gate-level logic descriptions from the RTL design, and also the 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 FPGA board.
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 not already set).
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 which file is to be loaded during elaboration into the instruction memory.
There are several ways to change the target program that is to be written into the instruction memory. The first 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 code which is not always desirable. Testbenches that instantiate "lt16soc_top" should always set the "programfilename" generic to the test program used.
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`) However, this change cannot be tracked by `git` and is therefore discouraged.

c) Synthesize the design with the `blinky.ram` program. Download it to the board with iMPACT. Observe the result and demonstrate it to the lab assistant. In addition, each participant should be ready to be quizzed about the properties of the test program, platform and SoC.

Question 1.3 - Challenge

Each warmup assignment 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 to choose is up the participant. The animation should feature at least 8 'frames'.
Hint: Lookup tables Lookup tables are a straightforward way to solve such problems.