EIT, Lund University

IC Project 1, ETIN35

Low-Power MCU for Edge Computing

PULPino

Author
Kristoffer Westring
Kristoffer.westring@eit.lth.se

LUND
UNIVERSITY

Contents

1 Information

The PULPino platform is an open-source microcontroller unit (MCU) built around the RISC-V instruction set architecture (ISA), making it an excellent educational tool for understanding the principles of MCU design and operation within the context of edge computing.

For more information on the PULPino platform, including its architecture, features, and how to get started with development, refer to the following resources:

- PULPino platform GitHub repository: `https://github.com/pulp-platform/pulpino`
- PULPino datasheet: `https://pulp-platform.org/docs/pulpino_datasheet.pdf`
- Local PULPino files: `/usr/local-eit/cad2/riscv/pulpino`

To gain a deeper understanding of the communication protocols used in PULPino, it is recommended to read about the APB (Advanced Peripheral Bus) protocol. A basic understanding of the bus protocol is required during implementation of a new peripheral.

- Overview of the APB protocol: `http://verificationprotocols.blogspot.com/2017/03/apb-protocol.html`
- Detailed APB protocol specification: `http://web.eecs.umich.edu/~prabal/teaching/eecs373-f12/readings/ARM_AMBA3_APB.pdf`

Important!

The projects can grow quite large, make sure to delete unnecessary files. This usually makes itself known by throwing a **Disk Quota Exceeded** message in the terminal. If you run out of disk space, try:

- Run **quota -s** to check your current disk quota. Remove files accordingly.
- Place your project on the local disk, however it is **not** backed up! You can create `/export/space/nobackup/<your-username>`. In this directory the you should not be limited by your quota. You will **only** be able to access this directory on the computer from which you created the directory. Make sure to backup your projects either on a USB stick or on GitHub or a similar service.

This information is crucial for the completion of your project.

2 Getting started

In order to get started with the project, you will need to locate the necessary files, and get a basic understanding of the structure of the project. The project is quite large and you don't need to understand every module in detail, but a general knowledge of the structure is required.

2.1 Setting up the project

All the necessary files for this project are located in `/usr/local-eit/cad2/riscv/`. You will need to copy the file containing the environment variables setup, `setup2022.efd`, the Modelsim initialization file containing the libraries, `modelsim_st65.ini`, as well as the PULPino directory, `pulpino`.

```
cd
mkdir etin35_project
cd etin35_project
cp -r /usr/local-eit/cad2/riscv/setup2022.efd /usr/local-eit/cad2/riscv/modelsim_st65
    .ini /usr/local-eit/cad2/riscv/pulpino ./
```

Each time you start a new terminal, or open a new tab in the terminal, you will have to source `setup2022.efd` if you want to execute Modelsim, synthesize, or compile something.

```
source setup2022.efd
```

If you get an error regarding syntax error, enter the bash terminal by typing **bash** before sourcing the setup file.

You will also need to update the RTL files by pulling them from GitHub remote repository. For this, run the following script below.

```
cd pulpino/
python update-ips.py
```

If you get an error regarding **remote**, when running **python update-ips.py**, open the python script with the same name in a text editor, and make the following changes

Listing 1: Potential error when running **update-ips.py**

```
Using remote git server https://github.com, remote is https://github.com/pulp-
    platform
From https://github.com/pulp-platform/IPApproX
* branch          verilator -> FETCH_HEAD
Traceback (most recent call last):
File "update-ips.py", line 91, in <module>
ipdb.update_ips(remote = remote)
TypeError: update_ips() got an unexpected keyword argument 'remote'
```

Listing 2: Snippet from **update-ips.py** to resolve error

```
# updates the IPs from the git repo
```

```
#ipdb.update_ips(remote = remote) # Before changes
ipdb.update_ips()
```

At this point, the python script should work and the IP:s should be pulled. The terminal will show print whether or not each ip has been successfully updated. The ip `adv_dbg_if` will not be updated, but this **should not** be an issue.

2.2 RTL simulation

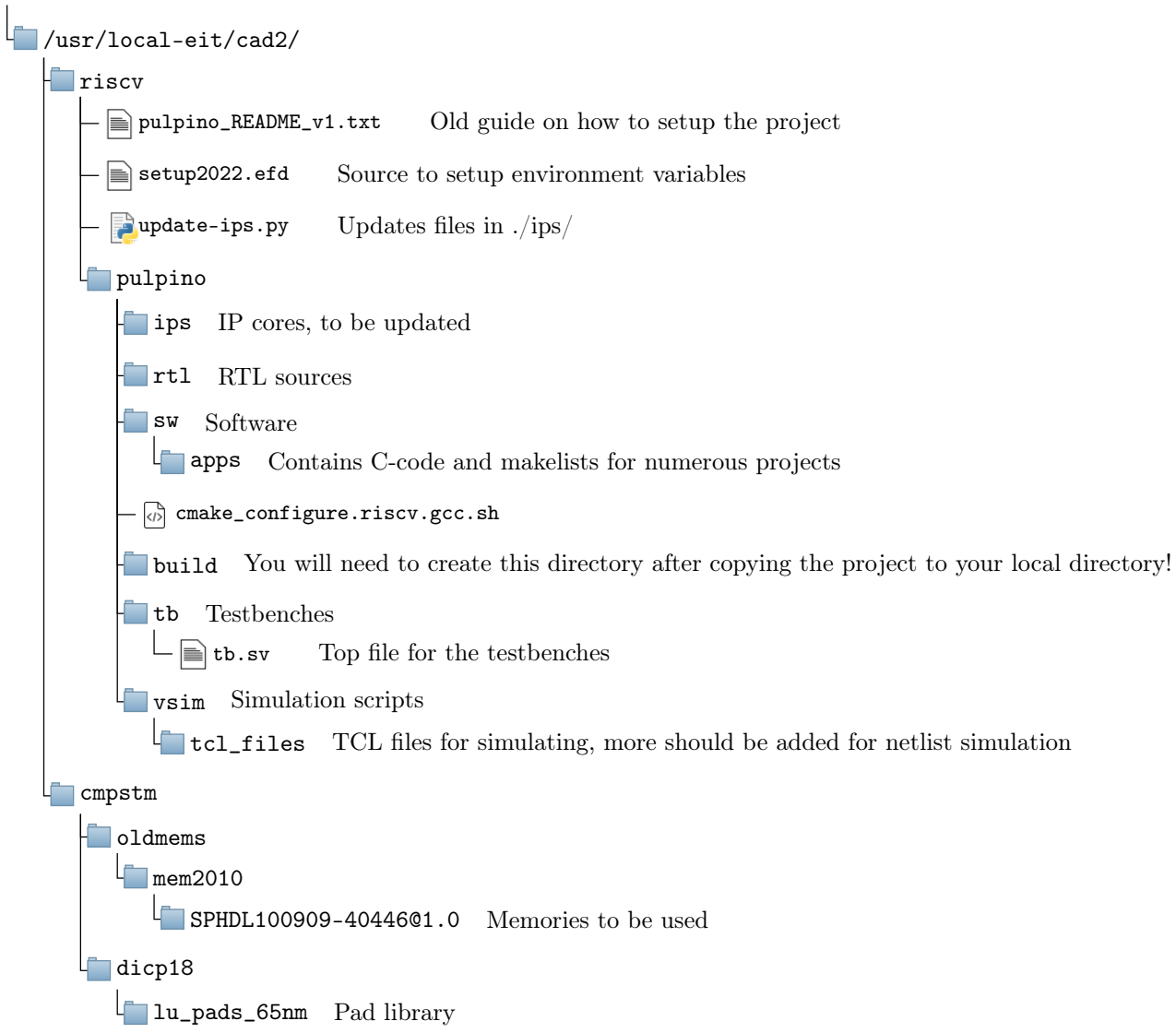
An helloworld-project has been created for you, which will be used as the source for the simulation. Before you can simulate you need to start the RISCV GNU Compiler Collection (**RISCV GCC**). This will create a makefile for you, which is used to simplify the compilation and simulation startup.

```
cd sw/
mkdir build
cp cmake_configure.riscv.gcc.sh build/
cd build/
source cmake_configure.riscv.gcc.sh
make helloworld.read
make vcompile
make helloworld.vsim (you can also try: make helloworld.vsimc)
```

It is recommended that you open `cmake_configure.riscv.gcc.sh` and `Makefile` in a text editor to understand what knobs you can turn and what the commands executed above, actually does. Try making your own application and see if you are able to run the code in the simulation!

2.3 Important files and directories

This directory tree shows the location of some of the more important directories and files. Note that the list is **not** a complete list over which files you will need to look and and modify, but is meant to act like a starting reference to help you navigate the project. You will need to gain a deeper understanding from experience.


/usr/local-eit/cad2/
riscv
pulpino_README_v1.txt
Old guide on how to setup the project
setup2022.efd
Source to setup environment variables
update-ips.py
Updates files in ./ips/
pulpino
ips
IP cores, to be updated
rtl
RTL sources
sw
Software
apps
Contains C-code and makelists for numerous projects
cmake_configure.riscv.gcc.sh
build
You will need to create this directory after copying the project to your local directory!
tb
Testbenches
tb.sv
Top file for the testbenches
vsim
Simulation scripts
tcl_files
TCL files for simulating, more should be added for netlist simulation
cmpstm
oldmems
mem2010
SPHDL100909-40446@1.0
Memories to be used
dicp18
lu_pads_65nm
Pad library

3 Starting the project

At this point, you need to start planning out the necessary work required to fulfill the tasks in the project, see Ch. 4. Some of the files in the project may not be synthesizable; it is your job to understand why and how this can be resolved.

If you have not done so yet, you need to look through the directories and understand which part of the modules does what.

Start off by swapping the systems existing memories to the SRAM provided, see Ch. 2.3, which is 2048x8. Figure out where the RAM is instantiated, and create a wrapper so that you can swap the memory to fit the 8096x32 RAM of the existing architecture. Take note of which functions the memories require, e.g., *is it byte-enabled? Does it require an enable signal? etc.*

After swapping the memories, make sure that the RTL simulation is still valid, when using the behavioral model of the memories. *This is a large project, in order to be able to structure all the different parts, please use multiple .tcl scripts for things that can be automated, like setting up the libraries in modelsim etc.*

Setup the tcl scripts needed for the synthesis. Look at your scripts for the previous project and reuse code! You need to figure out which RTL files is needed, and in which order they need to be compiled. Make sure that the synthesis is successfully finished. Read the reports!

Now it is time to simulate the netlist. Figure out where the libraries are set for Modelsim. It is a good idea to create a bash script for simulating the synthesis, to avoid manual edits when changing from behavioral to netlist simulation.

4 Tasks

4.1 Grade 3

1. Run *Hello World* program as an RTL simulation to verify functionality
2. Adapt 2048x8-bit RAM modules to simulate a 32kB, 32-bit wide memory with byte-enable functionality, compatible with the Pulpino platform's requirements
3. Switch from FLL to External Clock Source
4. Remove peripherals to limit the number of pads to 32. JTAG and SPI will be used, do not remove these
5. Define Separate Clock Domains for every clock signal in the ASIC flow

6. Simulation
   - RTL, initial verification
   - Gate level, synthesis validation (Post-synthesis)
   - Layout, Physical Verification (Post-layout)
7. Benchmark matrix multiplication on RISC-V vs accelerator
   - Performance
   - Power

4.2 Grade 4

1. All tasks required for grade 3.
2. Replace the timer peripheral with the matrix multiplier accelerator, created as a part of this course.
   - This requires the creation of a wrapper from which the accelerator can be attached to the APB bus [1].
3. Write an application in C for integration and testing of the accelerator in the MCU.
4. Configure the floorplan and re-run the ASIC flow.
5. Physical verification.

Tip: Create an APB testbench prior to attaching the peripheral to the system. It may be easier to test the functionality of the accelerator outside of the SoC, though this is **not** required for this course, only a suggestion.

[1] APB Protocol: `http://verificationprotocols.blogspot.com/2017/03/apb-protocol.html`

4.3 Grade 5

For grade 5, you do **not** need to perform the tasks for grade 4, i.e., implementing the matrix multiplier accelerator to the PULPino is not required.

1. All tasks required for grade 3.
2. Create a convolution accelerator.
   - Filter size 3×3
   - Filter size 5×5

   Note: The filter size is given by supervisor! Before initiating the work for grade 5, contact your supervisors to get the filter size of your convolution accelerator.
3. Replace the timer peripheral with the matrix multiplier accelerator, created as a part of this course.
4. Write an application in C for integration and testing of the accelerator in the MCU.
5. Configure the floorplan and re-run the ASIC flow.
6. Physical verification.

5 Contact details

The TA's for the project **Low-Power MCU for Edge Computing** in the course **ETIN35** for this period, and their contact details are listed below. Besides the weekly group meetings, you can either contact the TA's via email or Teams, or you can visit their room during office hours.

Sergio Castillo
Main Supervisor
sergio.castillo_mohedano@eit.lth.se
Room Number: E:2339

Kristoffer Westring
Co-Supervisor
kristoffer.westring@eit.lth.se
Room Number: E:2339

Alex Allfjord
Co-Supervisor
Mail: alex.allfjord@eit.lth.se
Room Number: E:2339