Revised Project Plan

02125 Bachelorproject

by Rasmus Wiuff **s163977**

12th of May

1 About the project

The project aims to develop a framework written in Java, to facilitate digital circuit verification. This framework will be developed based on defined use cases and will lead to a user-friendly workflow, formed by relevant inputs from users of verification software. The framework will utilise a simple, yet powerful backend to drive the simulations of the DUT (Device Under Test). Lastly, the framework is optimised for concurrent execution.

2 Goals and work packages

2.1. Main goals

The goals of the framework are as such:

1. Simple Workflow: It should be easy to set up and run tests.
2. High Performance: Efficient backends and concurrent execution is essential.
3. Versatility: Method overloading, clever type definitions and functional programming should make the user able to write versatile tests for multiple implementations at a time.

2.2. Work packages

In order to keep track of the project, a list of work packages is defined.

1. Testing Frontend: Working frontend that communicates with a dummy backend.
2. Toolchain Definition: A functioning API between front- and backend.
3. Concurrency: Parallel execution of tests.

3 Tools and technical knowledge

During the project, a list of tools and concepts will be investigated and applied.

3.1. CI/CD

The project will be managed with Gradle[1] on a git repository (hosted on GitHub). When set up, this ensures automatic compiling and unit test execution, enabling a continuous development cycle.

3.2. Concurrency in Java

Java has classes for creating, managing and joining threads, however coroutines provide higher performance. This is not natively supported by Java, however there are promising projects and approaches to be found online[2].

[1] gradle.org

[2] See “Coroutines in pure Java” by esoco on Medium

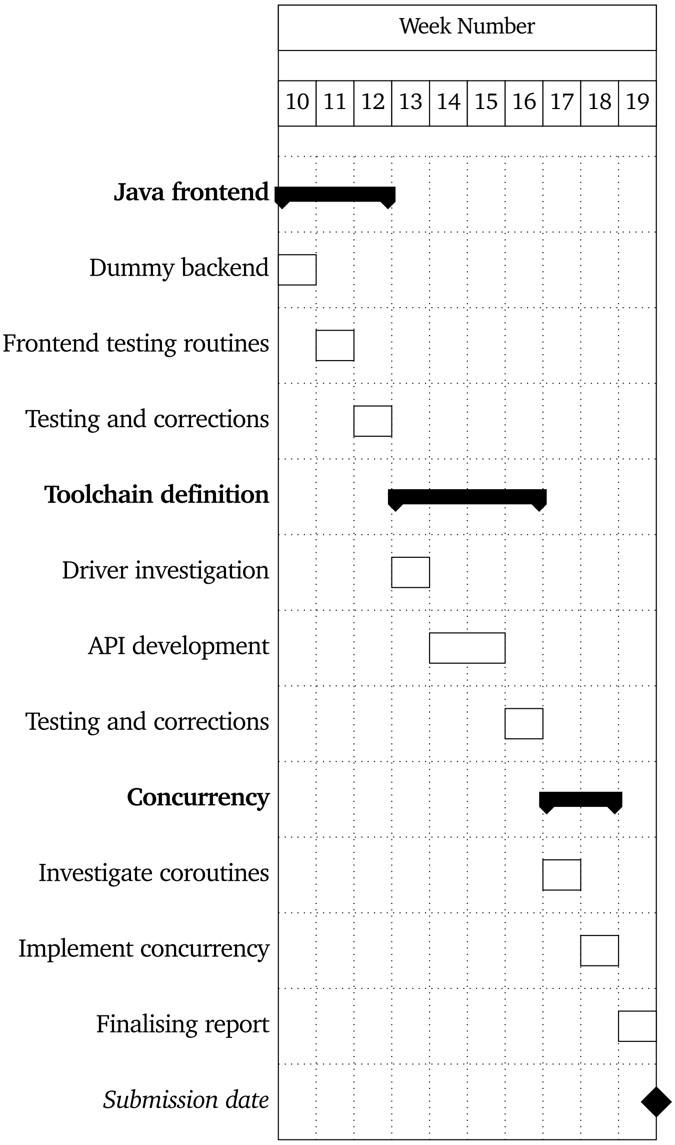
3.3. Verilator and drivers

Verilator[3] is needed to simulate and test the DUT. In order to drive Verilator, this project will investigate some of the current Hardware Description Languages (HDL's) and their drivers. The new driver from Chisel, svsim[4] and the simulator from SpinalHDL[5] are under consideration.

4 Timeline

Fig. 1 outlines the project, subdivided into tasks.

Figure 1: *Project timeline*



[3] www.veripool.org/verilator
[4] svsim on GitHub
[5] SpinalHDL on GitHub

5 Self-evaluation

During this project I had the following findings:

5.1. Coursework

It seemed that I regrettably signed up for a rather intense course besides this project. In the future, I will have a more focused approach to the larger projects (i.e. the Master's Thesis).

5.2. Choice of topic

I found the topic quite interesting. I have previously worked with embedded devices and low-level programming, so this might have been a very good match for me as a middle ground between software and hardware, or rather, software supporting hardware development.

5.3. Project work

I find these project largely more enjoyable, than regular study practices. Even though I did not reach all the goals for my project, I find myself trying more novel approaches and working harder at the problems, when I have this larger influence of the projects scope and goals.