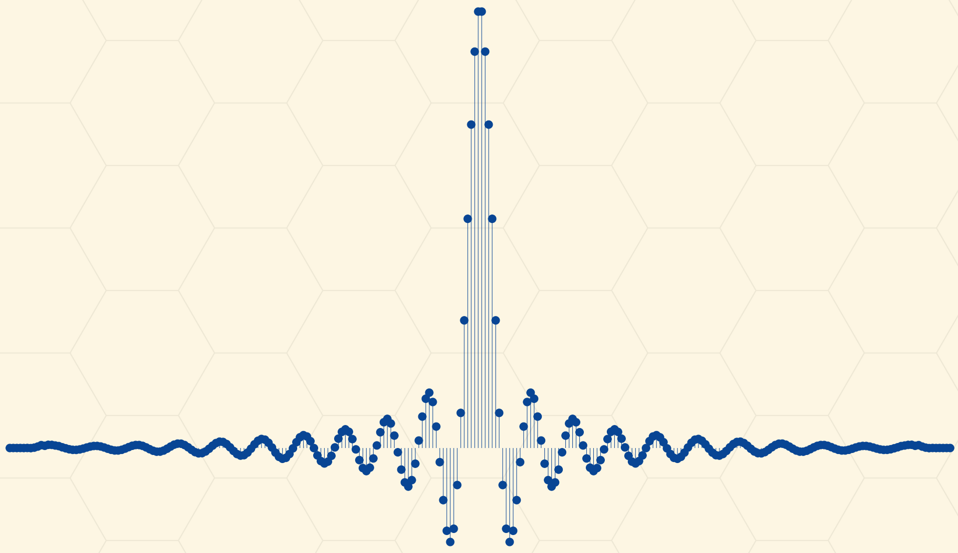
University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Front-End Signal Processing for Red Pitaya Spectrum Analyzer

Bachelor Thesis: Specifications and Planning



Degree Program:	Electrical Engineering and Information Technology
Course:	Project 6
Coaches:	Prof. Dr. Richard Gut, Michael Pichler
External Expert	Dr. Jürg Stettbacher
Team:	Raphael Frey, Noah Hüsser
Date:	August 18, 2017
Revision:	1.0.0

Institute of Microelectronics

Contents

CHAPTER 1

Initial Situation

To replace common lab tools which are usually expensive and big, Red Pitaya produces a board called STEMlab which features a capable ($125\frac{MS}{s}$, 14bit) ADC and a ZYNQ SoC from Xilinx. The idea is to record data on the board and transmit it using the network infrastructure to a PC which does all the number crunching.

To reduce the data rate and processing load, downsampling filters can be enabled to decimate the signal. The filters are necessary in order to keep the Nyquist theorem is fulfilled and to prevent aliasing. Sadly, the stock filters are not quite as capable as desired, as they do not attenuate the stopband well enough and have a frequency response not as flat as desired.

A previous project implemented new filter chains which are based on IIR filters. While those filter chains worked well in theory (i.e. their design was fundamentally sound), they did not produce satisfactory results in practice.

This project aims to analyze the previous work with respect to design and implementation of those filter chains, and to reproduce and fix the problems that exist in the earlier solution.

The structure of the idea behind the STEMlab can be seen in Figure 1.1.

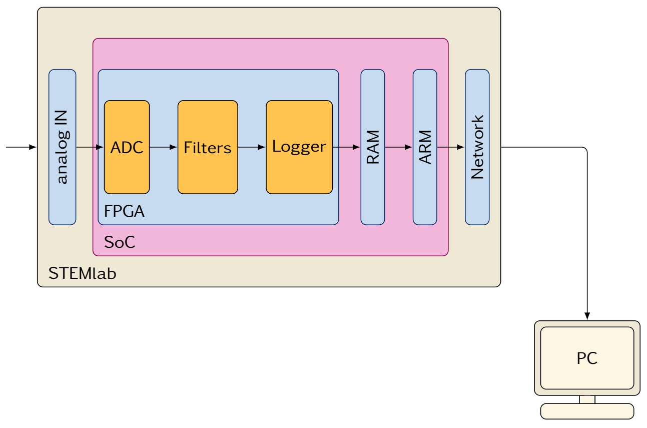


Figure 1.1: Block diagram of the STEMlab 125

CHAPTER 2 Objectives

Note: Revised goals are the result of decisions and findings during the project, which have lead to amendments in its goals.

2.1 Must Haves

1. Analysis of the previous project
   a) What was the exact strategy?
   b) Why did it fail?
   c) How can potential mistakes be fixed?
   d) Is the project salvageable?
2. Comparing and finding a solution to implement the desired filter chains
   a) Start over from the beginning, or
   b) Continue working on the existing solution.
3. A complete specification for all the filter chains with respect to
   a) Decimation Rates
   b) Attenuation in stopband
   c) Maximum ripple in passband
4. A complete simulation of the filter chains either in Matlab or another tool.
5. Filter implementation on the FPGA
6. Verification of the performance of the implemented filters

2.2 Must Haves (revised)

After analysis of the previous project and the overall STEMlab ecosystem, these are the modified/amended objectives. For the full analysis, see the main project report.

1. Develop our own STEMlab toolchain including
   a) Build system
   b) Good documentation
   c) Vivado TcL scripts
2. Develop our own oscilloscope application
   a) A server on the STEMlab
   b) A scope application in JavaScript

2.3 Nice to Haves

1. Additional scoping application features
   a) Automatic SNR calculation
   b) Automatic THD calculation
   c) Multiple trace sources
   d) Different types of triggers

2.4 Do Not Implement (revised)

1. IIR filter chains

CHAPTER 3

Objects of Delivery

- Specifications (periodically, 18.08.2017)
- Working Implementation of the Filter Chains (18.08.2017)
- Working Implementation of the JavaScript Scoping Application (18.08.2017)
- Factsheet (31.08.2017)
- Technical Report (18.08.2017), including Developer and User Guides
- Working Repository (containing READMEs, Toolchain, Virtual Machine, Calculations, Simulations, Measurements, etc.) (18.08.2017)
- Presentation (15.09.2017)

CHAPTER 4

Concept (revised)

4.1 Toolbox (revised)

Since it has become apparent that the STEMlab project base led by Red Pitaya is going through a major overhaul and it is uncertain whether this will change in the future, we will develop our own toolbox.

We will implement a toolchain that can build an ARM Linux image with all the necessary components. This will be done based on the original Red Pitaya patches to make the Linux run on the ZYNQ and some scripts of Open Source projects which we will adapt to our needs and a more general toolchain.

We will use GNU/Linux as a Platform and Matlab and Vivado for filter design and implementation, respectively.

For the JavaScript Application we will use node.js, yarn, rollup and mithril.js for the implementation.

4.2 Realization (revised)

The preceding project implemented filter systems by cascading several IIR filters. While IIR filters tend to use fewer resources for a given transition band width than FIR filters, it still turned out that resource usage was too high to implement filter cascades for both channels present on the STEMlab.

Consequently, the decision was made to exploit CIC filters in this project. They are very computationally efficient and can be used as initial stages in a filter cascade, followed by FIR filters to shape the passband response and clean up any undesired effects which may have been introduced into the frequency response of the system. Two types of CIC filters are used: One with a decimation ratio of 25, and another with a decimation ratio of 125. Both are then followed by their respective compensation FIR filter and more FIR filters for the final rate reduction.

As it has been decided to develop an entirely self-dependent solution of the project, general components on the FPGA such as an ADC controller or a data logger are required. For the ADC part an IP from Pavel Demin will be utilized (see the project report for further remarks and source). For the logger part an IP core that Mr. Hüsser developed in a previous project will be used.

For the ARM Core, a Linux will be set up that can communicate with the logger core via a GNU/Linux kernel module. A server application will be developed which will run on the Linux to send the data written to the RAM by the logger to the network. This will be done using WebSockets.

The oscilloscope web application will be implemented using JavaScript. WebSockets are used to transmit and receive data to and from the network. As build systems, yarn and rollup will be used. To implement the GUI side of the application, mithril.js will be deployed.

4.3 Testing

To test the performance of the filters, a series of tests will be performed.

The following metrics will be calculated from the measured data.

- The RMS frequency response
- The SNR frequency response
- The attenuation in the stopband

Optional testing would involve a comparison to the reference implementation and to a commercial spectrum analyzer.

CHAPTER 5

Project Timeline (revised)

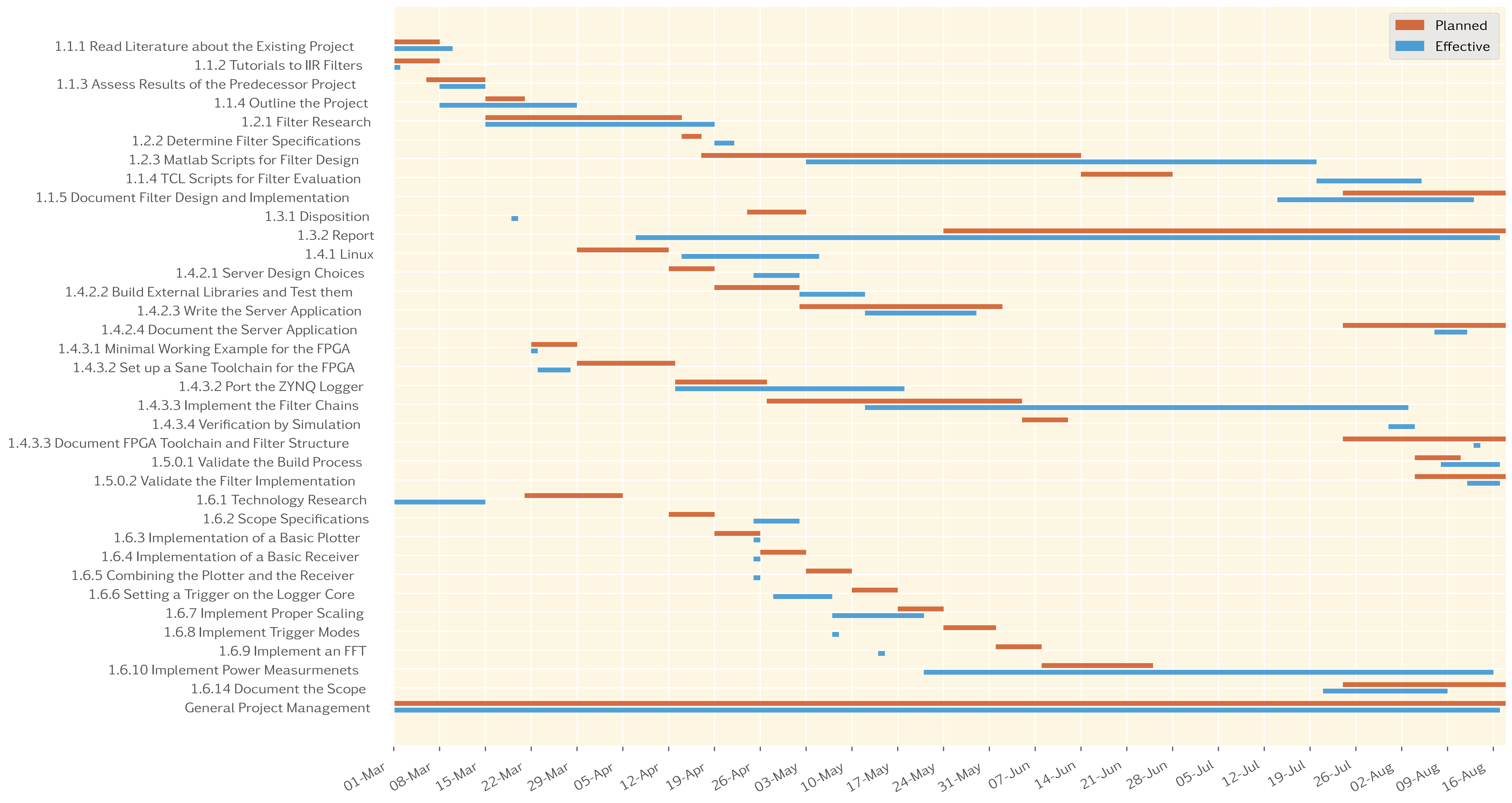


Figure 5.1: The timetables of the project. The planned and effective time spans are visible.

CHAPTER 6

Work Packages (revised)

6.1 General Prerequisite Tasks

Contains tasks to get an overview of the previous project's current state and to define further steps and requirements to this project which follow from that.

6.1.1 Read Literature about the Existing Project

ANCESTORS:	None
ASSIGNEE:	Raphael Frey, Noah Hüsser
DESCRIPTION:	Read the preceding project's documentation and general information about the STEMlab board to get familiar with the project and its general state. Based on this, the specifications will be determined and the project requirements (this document) will be revised to reflect the new findings.

6.1.2 Tutorial to IIR Filters

ANCESTORS:	None
ASSIGNEE:	Raphael Frey, Noah Hüsser
DESCRIPTION:	Complete the lab course about IIR filters and their bit growth problems and fixes provided by Mr. Pichler.

6.1.3 Assess Results of the Predecessor Project

ANCESTORS:	Read Literature about the Existing Project
ASSIGNEE:	Raphael Frey, Noah Hüsser
DESCRIPTION:	Define what the current project status is and where we can connect to the predecessor project.

6.1.4 Outline the Project

Ancestors:	6.1.3
Assignee:	Raphael Frey, Noah Hüsser
Description:	Define what the requirements for the project are and which of those are a must and which a nice to have.

6.2 Filter Design

Contains the work packages relating to filter design in Matlab and filter implementation in the FPGA tool chain.

6.2.1 Filter Research

Ancestors:	6.1.4
Assignee:	Raphael Frey
Description:	Research the digital filter technologies which will be needed for this project. In particular, this includes FIR and CIC filters, since Xilinx provide predefined blocks for these filters in their toolchain. Besides the general knowledge on digital filters, the Xilinx toolchain must be researched in order to understand the FIR and CIC filter blocks and allow their usage.

6.2.2 Determine Filter Specifications

Ancestors:	6.2.1, 6.1.4
Assignee:	Raphael Frey
Description:	Specify the minimum and nice to have requirements for filters. Document this well.

6.2.3 Matlab Scripts for Filter Design

Ancestors:	6.2.2
Assignee:	Raphael Frey
Description:	Matlab scripts which design various filter chains. These consist of one or several dispatcher scripts where the filter chains are specified in terms of their frequency band properties, and scripts which actually design the filters according to these specifications. Specifically, scripts for generating CIC filters and their compensation filters, as well as FIR filters are needed. For FIR filters, the resulting coefficients are to be saved in files so that they can be loaded by the Xilinx tool chain. If possible, Matlab's parallelism should be exploited to reduce filter design times.

6.2.4 TCL Scripts for Filter Evaluation

Ancestors:	6.2.3
Assignee:	Raphael Frey
Description:	In order to assess the resource usage of the filter chains which are designed by Matlab, the respective filter blocks (CIC, FIR) need to be implemented in Vivado and a bitstream must be compiled. Since the number of filters generated by package 6.2.3 can be very large (dozens or even hundreds of filters), this process must be automated to be of any use. For this, Tcl scripts are used. For FIR filters, the scripts load the coefficient files which have been generated by Matlab. Vivado will generate usage reports which can then be automatically post-processed.

6.2.5 Document Filter Design and Implementation

Ancestors:	6.2.4
Assignee:	Raphael Frey
Description:	Documentation for the filter design tool chain. The documentation should be sufficiently detailed so that the tool chain can be used by a person who wishes to create new filters for a new bitstream.

6.3 Documentation

6.3.1 Disposition

Ancestors:	None
Assignee:	Noah Hüsser
Description:	Create a general outline of the thesis document.

6.3.2 Report

Ancestors:	None
Assignee:	Raphael Frey, Noah Hüsser
Description:	Continuously check, restructure and update the report to current new insights. Compile and round out the actual thesis document after all the different parts have been written.

6.4 Firmware

6.4.1 Linux

Ancestors:	None
Assignee:	Noah Hüsser
Description:	Boot an Ubuntu Linux onto the Red Pitaya.The target here is to have a ARM Linux distribution which boots fine on the STEMlab. The official sources from the STEMlab project should be used as they contain a lot of patches for the STEMlab hardware. A standalone toolchain has to be developed as the compiled versions do not contain the right device tree (DT) and board support package (BSP). Compiling in the right DT and BSP is not part of this task.

6.4.2 Server Application

6.4.2.1 Server Design Choices

Ancestors:	None
Assignee:	Noah Hüsser
Description:	It has to be determined what technology should be used to provide the server side of the data transmission. The server needs to be able to perform IOCTL calls and to serve data with the chosen protocol over the network. It has to run on the arm GNU/Linux and shouldn't have too many dependencies. A C-like language will most likely serve best since it's close to the kernel.

6.4.2.2 Build External Libraries and Test Them

Ancestors:	6.4.2.1
Assignee:	Noah Hüsser
Description:	A library for networking and another for JSON are needed. The requirements will be decided by the specifications of the scope application.

6.4.2.3 Write the Server Application

Ancestors:	6.4.2.2
Assignee:	Noah Hüsser
Description:	A server application has to be written that reads data from the Logger and transmits it over the network. It needs to be able to configure the logger via RPCs that are mapped to `IOCTL`s. It should read data from `/dev/logger` which is presented by the kernel module.

6.4.2.4 Document the Server Application

Ancestors:	6.4.2.2
Assignee:	Raphael Frey, Noah Hüsser
Description:	Write documentation for the server application build process.

6.4.3 FPGA

6.4.3.1 Minimal Working Example

ANCESTORS:	None
ASSIGNEE:	Noah Hüsser
DESCRIPTION:	Get a minimal working example project running on the Red Pitaya. This requires importing the cores from Pavel Demin's project as well as compiling the Linux for the STEMlab. Correct functionality of the Red Pitaya hardware, particularly the ADC, is to be verified via Anton Potochnik's frequency counter.

6.4.3.2 Set up a Sane Toolchain for the FPGA

ANCESTORS:	6.1.3,6.4.3.1
ASSIGNEE:	Raphael Frey
DESCRIPTION:	Depending on the previous project's results, an entirely new Toolchain is required or not. Either way, Vivado can be very finicky and we need to have a sane toolchain for the entire FPGA build process so we can work cleanly.

6.4.3.3 Port the ZYNQ Logger

ANCESTORS:	6.4.3.2
ASSIGNEE:	Noah Hüsser
DESCRIPTION:	Port the Zynq Logger to the Red Pitaya. The main challenges will be the porting of its interface (for which a block `axis2datalanes` is developed), as well as understanding and properly implementing the device tree and the kernel module which is needed for the Linux OS to interface with the logger hardware.

6.4.3.4 Implement the Filter Chains

ANCESTORS:	6.4.3.3
ASSIGNEE:	Raphael Frey
DESCRIPTION:	Implement the filter chains as designed in Matlab on the FPGA. This requires correct functionality of the FIR and CIC filter blocks.

6.4.3.5 Verification by Simulation

ANCESTORS:	6.4.3.4
ASSIGNEE:	Raphael Frey
DESCRIPTION:	Simulate all the FPGA blocks. Simulate the filters with a sine to make sure the filter work as expected.

6.4.3.6 Document FPGA Toolchain and Filter Structure

Ancestors:	6.4.3.5
Assignee:	Raphael Frey
Description:	Document the implemented filter chains and surrounding FPGA structure.

6.5 Validation

6.5.1 Validate the Build Process

Ancestors:	6.4.3.5, 6.4.2.3, ??
Assignee:	Raphael Frey
Description:	Build the entire project (Linux, bitstream), flash result onto the Red Pitaya and verify correct functionality.

6.5.2 Validate the Filter Implementation

Ancestors:	6.4.3.5
Assignee:	Noah Hüsser
Description:	Validate the specified frequency response of the filters and the attenuation in the passband.

6.6 Front-end

6.6.1 Technology Research

Ancestors:	None
Assignee:	Noah Hüsser
Description:	Research the possibilities to design and implement a nice front-end for the STEMlab recording system. The STEMlab Project has a built-in web interface which allows the display of signals and rudimentary calculations. Furthermore Prof. Gut has implemented a spectrum analyzer in Java which interfaces with the STEMlab. Both approaches are feasible and should be evaluated thoroughly. Furthermore it has to be evaluated how the data should be transferred to the front-end. This decision correlates a lot with the selection of a programming language since e.g. web technologies do not support UDP. Finally, existing libraries have to examined very carefully. There are a lot of libraries around data transmission and GUI design, but not all of them are viable solutions.

6.6.2 Scope Specifications

Ancestors:	6.6.1
Assignee:	Noah Hüsser
Description:	Specify the scope's requirements and nice-to-haves.

6.6.3 Implementation of a Basic Plotter

Ancestors:	6.6.1
Assignee:	Noah Hüsser
Description:	At first a simple plotter is implemented. It should not do more than plot a given vector of data points on a canvas. The plotting should be done with high performance at 60 frames per second independently of the size of the data vector. Of course more data will slow down the plotting, but it should perform well enough that vectors up to 64 kilosamples in size plot nicely. For this task it is recommended to use existing libraries which make use of the GPU such as OpenGL.

6.6.4 Implementation of a Basic Receiver

Ancestors:	6.6.16.4.2.3
Assignee:	Noah Hüsser
Description:	A simple receiver which receives recorded samples over the network should be created. For this task a simple emitter which emits test samples without actually recording anything should be implemented. In further tasks it can be used conveniently without the need of a running STEMlab board. The receiver should simply receive samples as packets with lengths of powers of two over the network. Again, a good, high-performance and publicly accepted library should be used to reduce workload and ensure reliability.

6.6.5 Combining the Plotter and the Receiver

Ancestors:	6.6.3, 6.6.4
Assignee:	Noah Hüsser
Description:	The plotter and the receiver should be combined to display vectors received over the network. This should happen fluently without any delays.

6.6.6 Setting a Trigger on the Logger Core

Ancestors:	6.6.5
Assignee:	Noah Hüsser
Description:	The application should have a ability to set a trigger type for any channel with the possibility to do this using a GUI.

6.6.7 Implement Proper Scaling

Ancestors:	6.6.6
Assignee:	Noah Hüsser
Description:	The GUI should not only plot a qualitatively correct signal but also show its quantitative properties. A grid should be programmed into the GUI which can be used to read metrics of a signal. A panel on the GUI should display the current grid size. It should be possible to scale signals using the mouse or the GUI. Grids should adjust with the current scale.

6.6.8 Implement Trigger Modes

Ancestors:	6.6.6
Assignee:	Noah Hüsser
Description:	It should be possible to use the scope application in *normal, auto,* and *single* mode. Those modes should behave as follows:

normal	send a packet when a trigger was fired and repeats this procedure
auto	send a packet when a trigger was fired or a timeout occurred and repeat
single	send a packet after a first trigger and does not repeat this procedure

6.6.9 Implement an FFT

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	An FFT should be calculated and displayed live.

6.6.10 Implement Power Measurements

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	Calculate the power in the signal. It has to be possible to calculate the power between two markers to limit the calculation on a certain part of the signal.

6.6.11 Implement SNR Calculation

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	The SNR of the signal between two markers has to be calculated automatically. As a reference use the Java application of Prof. Gut.

6.6.12 Implement THD Calculation

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	The THD of the signal spectrum has to be calculated automatically. As a reference, use the Java application of Prof. Gut.

6.6.13 Implement Dual Channel Possibilities

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	The entire transmission and plotting of all the signals should be implemented for dual channel capabilities. The FPGA code supports up to eight channels whilst only supplying two to the server since the STEMlab board only supports two physical channels.

6.6.14 Document the Scope

Ancestors:	6.6.2
Assignee:	Noah Hüsser
Description:	Document the entire oscilloscope application and how to use it.