Semester Project

Architectural exploration of low-pass FIR filters using MATLAB HDL Coder

Student: Nikos Stylianou

AEM: 2917

Course: ECE 494 - Processor Design

June 19, 2023

Contents

1 Introduction

A low-pass FIR (*Finite Impulse Response*) filter is a type of digital filter used in signal processing to attenuate or remove high-frequency components from a signal, allowing only low-frequency components to pass through. FIR filters are characterized by their impulse response, which is a finite duration sequence of coefficients. The basic operation of a low-pass FIR filter involves convolving the input signal with the impulse response of the filter. The filter coefficients determine the filter's frequency response, and specifically in a low-pass filter, they emphasize the passage of low-frequency signals and suppress higher frequencies The design of a low-pass FIR filter involves determining the appropriate filter order (length) and selecting the filter coefficients. The filter order determines the length of the filter's impulse response and affects the sharpness of the filter's frequency cutoff. Higher filter orders generally result in steeper roll-off characteristics but require more computational resources.

For this project, I will be focused in exploring

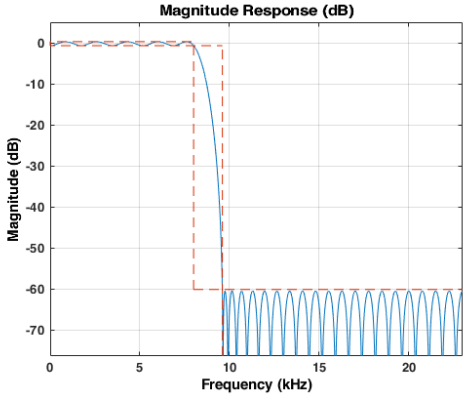

Figure 1: FIR magnitude response

the architecture of an FIR filter with the following characteristics:

- Sampling frequency: 46 kHz
- Pass frequency: 8 kHz
- Stop frequency: 9.6 kHz
- Number of coefficients:
  - Minimum coefficients possible
  - 20 coefficients
  - 30 coefficients
- Arithmetic:
  - Single precision floating point
  - Fixed point 24 bits
  - Fixed point 16 bits
  - Fixed point 8 bits
- Optimizations:
  - 1 stage multiplier pipelining
  - 2 stage multiplier pipelining
  - Multiplier-less using CSD
  - Distributed Arithmetic

The implemented filter has a magnitude response as figure 1 shows.

2 Filter Creation

Before creating any HDL code, the filter has to be normally implemented in MATLAB. Using `designfilt` command, every parameter of the filter is filled with the help of GUI. We call this function 3 times, one for each different order of filter, thus creating 3 different FIR filters. Using

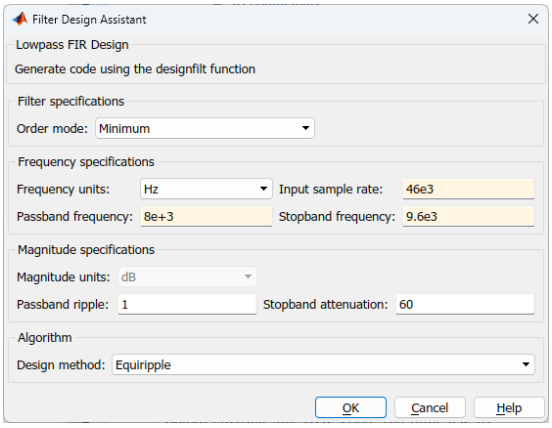

Figure 2: MATLAB's design assist tool for creating lowpass FIR filters

`fvtool`, all three filters can be compared and be checked for compliance.

For HDL Coder to work, all filters must be designed by `fdesign.lowpass()` and `design()` functions, thus all filters created above must be re-created. The first function creates the specification for each filter containing useful information such as passband/stopband frequency, sample rate, etc. In order to actually create the filter, `design()` function has to be called and it returns `dsp.FIRFilter` data type, meaning that the wanted filter is successfully created. Note that, we make use of two specific toolboxes provided by Mathworks, called *DSP System Toolbox* and *DSP HDL Toolbox* that provide those "easy" methods of creating filters alongside with optimizations like pipelining, CSD and others needed for this project.

After each filter is created, `fdhdltool` has to be called with filter specifications and numeric type needed as arguments. Every aspect of the generated filter can be tuned from there such configuring architectures, multiplier pipelining, test bench generator and more. This task has to be executed for each implemented filter.

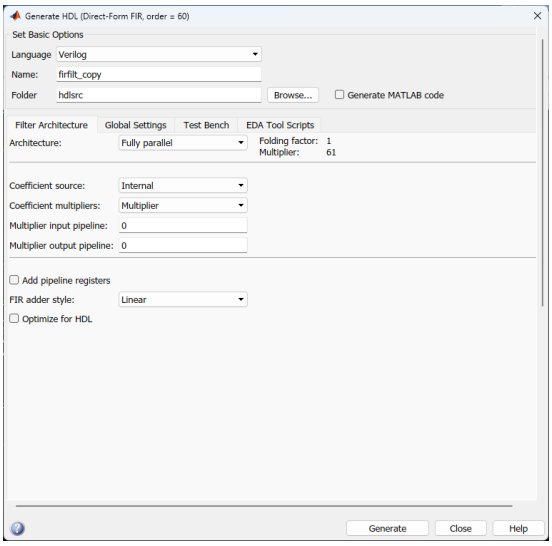

Figure 3: MATLAB's tool for creating HDL filters.

3 Filter Synthesis

Once creating every filter with its own configuration, each HDL file has to be synthesized and/or implemented in order to compare each architecture. For this process, Xilinx Vivado seems to be the best option to use as other great EDA tools like Synopsys and Cadence suites weren't available. Since I'm using Vivado, I should as well target an FPGA that might have access such as the Zedboard. Zedboard isn't a very big FPGA in terms of memory size, but, hopefully, it might be able to implement some filters at the end.

3.1 Importing MATLAB HDL files

In Vivado, a new project is created for each filter architecture. Files are imported using the standard GUI procedure while also adding constraints for the target device mentioned above. MATLAB produces source HDL and test-bench for each filter created.