EE 465 Lab Exam 1

Created by Rory Gowen

January 18, 2017

Introduction

This document describes the setup of a Verilog Project that can be used to measure the magnitude/frequency response of a filter and to help with the setup of a circuit to determine MER of an RC filter. These circuits must be setup for the first lab exam.

Setting Up a Project for the Lab Exam

Create a Verilog Project for the DE2-115 (EP4C115F29C7) and include the following verilog files:

- lab_exam_1.v
- lab_exam_1.sdc
- EE465_filter_test_baseband.qxp

Set the `lab_exam_1.v` file to be the top level module. Connect the filter to be tested to the `EE465_filter_test_baseband` module in the `lab_exam_1` module.

After you have completed all of the steps for this exam create a signal tap file to display the following signals:

- the impulse created by you Impulse Generation Circuit
- the output of your transmit filter
- the output of your receiving filter
- any signals that you need to determine the MER of your cascaded filters

Use the sample_clk_ena as the clock for signal tap. You may also include any other signals that will be of help to you.

Marks will be based upon:

- Your Transmit filter meeting the specifications provided in the Deliverable 1 document
- The number of multipliers uses in your transmit filter (the fewer the better)
- Your implementation of the MER circuit
- Questions about your design choices

EE465 Filter Test Module – (EE465_filter_test_baseband.qxp)

The `EE465_filter_test_baseband` module provides the following functions for testing a filter:

- Generates the following clock outputs:
  - system_clk: clock_50 divided by 2
  - sample_clk_ena: pulse of duration (1/system_clk), frequency of clock_50 divided by 8
  - symbol_clk_ena: pulse of duration (1/system_clk), frequency of clock_50 divided by 32
- Generates a 2-bit pseudo-random symbol sequence based upon a 22-bit maximum length LFSR (`lfsr_value`)
- Generates a 1s17 signal containing a 4-ASK signal that has been upsampled by 4. (`input_to_filter_1s17`)
- Scales the amplitude of `input_to_filter_1s17` by a factor of $2^{-\texttt{filter\_input\_scale}}$.
- Upsamples the output of the filter being tested by a factor of 4 and anti-aliases the upsampled output of the filter.
- Provides a 14-bit value that can be connected directly to the DAC on the DE2-115 daughter card.

The ports list of the module is given in table 1 with a general description of the function of each connection in the port list.

It should be noted that not all filters to be tested will require all of the port connections. The number of connections that you need will depend upon whether or not you have implemented a multiplierless filter or not. For timing considerations the clocks generated by the module should be used throughout the project containing the module.

A basic block diagram of the module is given in figure 1.

Table 1: Port list and Description for EE465_filter_test_baseband Module

Port Connection Name	**Direction**	**bit format**	**Description**
clock_50	input	clock	Provides a master clock to the module clock is expected to be 50 MHz.
reset	input	1-bit	Active high reset. Resets the module and starts the LFSR.
output_from_filter_1s17	input	1s17	The output of the filter should be connected to the module here.
filter_input_scale	input	2-bit	Scales the input signal to the filter (`input_to_filter_1s17`) by $2^{-\texttt{filter\_input\_scale}}$. This signal can be used to help debug overflow issues.
input_to_filter_1s17	output	1s17	A 4 times upsampled signed value that can be connected to the filter input. The upsampled signal is a full-scale and zero-stuffed.
lfsr_value	output	2-bit	Provides 2-bits from the LFSR that can be used for testing certain implementations of filters. This value is sampled at a frequency of system_clk Hz.
output_to_DAC	output	1u13	Provides a signal that can be directly connected to the DAC.
system_clk	output	clock	Provides a clock at a frequency of $\frac{clock\_50}{2}$
sample_clk_ena	output	clock	Provides a clock pulse of duration $\frac{1}{system\_clk}$, frequency of $\frac{clock\_50}{8}$
symbol_clk_ena	output	clock	Provides a clock pulse of duration $\frac{1}{system\_clk}$, frequency of $\frac{clock\_50}{32}$

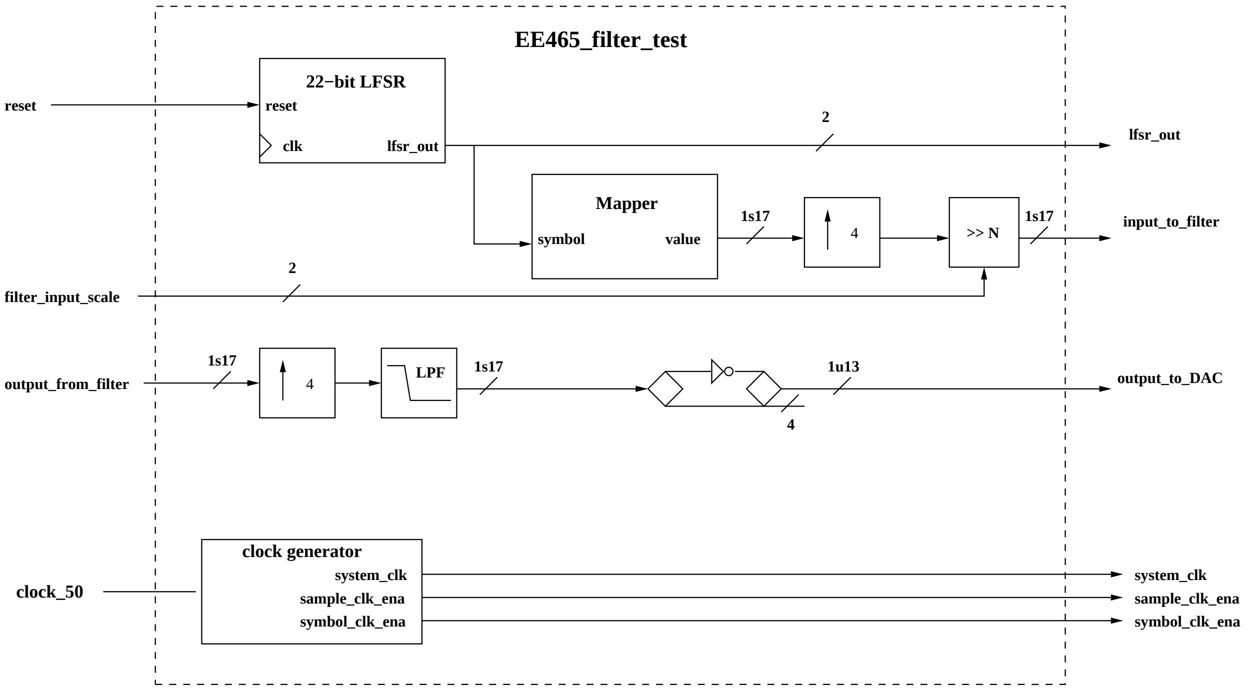

Figure 1: EE465_filter_test_baseband Block Diagram

Adding Your Filter to the Project

The filter to be tested should be instantiated in place of the **ee465_gold_standard_srrc** module in `lab_exam_1.v`. Your particular design may not require the same signals as the **ee465_gold_standard_srrc** module. You will have to change connections as appropriate for your filter.

You may notice that the **ee465_gold_standard_srrc** uses two clocks **sample_clk_ena** and **system_clk**. The reason for this to to help align the clock edges in circuits that use multiple sampling rates. The filters that you have currently designed may not require the use of both clock signals. Your filter may work correctly by using the **sample_clk_ena** signal only.

Later on in this class you will want to add the **system_clk** to your filter designs. If you want to add the system_clk to your current design the process is quite simple. There is an example of the change below. *This topic will be discussed in more detail in class at a later date.*

Any circuit that uses a clock edge (i.e. a register) such as the one in the code that follows:

```
always@ (posedge sample_clk_ena)
        delay <= sig_in;
```

Would be replaced with:

```
always@ (posedge system_clk)
        if (sample_clk_ena == 1'b1)
```

```
                delay <= sig_in;
        else
                delay <= delay;
```

Testing Your Filter

Once your filter has been instantiated and connected to the ee465_filter_test_baseband module compiled and put on a DE-115 board you may view the output of your filter on a signal analyzer.

Connect the output of **DAC_B** to the signal analyzer and set the frequency range to be 0 MHz to 3.125 MHz. Turn trace averaging on and press **KEY[3]** on the DE2-115 board to start the random data sequence.

Measure the peak power of your filter relative to the stop band power using a delta marker to ensure that your filter meets the requested specifications. If it does not then you will need to do further work on your filter design.

In order to get the best measurement results from the signal analyzer the filter being tested should have the maximum output possible without overflow errors occurring. This allows for the largest dynamic measurement range of the signal analyzer when measuring your filter characteristics. *You may want to adjust your filter coefficients to achieve the maximum output gain possible*

Determining the MER of your filters

You will need to create an impulse generation circuit that will aid in determining the ISI of your filters. You will need to cascade both the transmit SRRC filter (your windowed filter) and the receive SRRC filters (non-windowed original specification filter) and pass an impulse through the filters.

At the request of professor Salt, I have been asked to not give the block diagram of the MER measurement circuit to you. You will have to devise a circuit to measure the MER of the filters on your own. *Hint: You may implement a circuit that operates the same way as your Matlab script for calculating MER, however in this case you will most likely not be able to normalize you received power to 1. It may be helpful to create circuits that measure the peak output power and the ISI of the cascaded filters.*

The MER calculation does not need to be done in verilog (i.e. you do not need to display the dB result in Signal Tap). You only need values in Signal Tap that you can then use to hand calculate the MER.

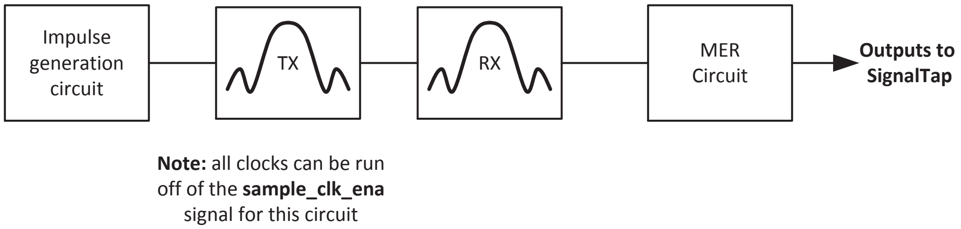
Impulse generation circuit
TX
RX
MER Circuit
Outputs to SignalTap
Note: all clocks can be run off of the sample_clk_ena signal for this circuit

Figure 2: MER test circuit