UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DEVELOPMENT AND TESTING OF AN FPGA-CONTROLLED SWITCHED-INTEGRATOR CURRENT AMPLIFIER FOR USE IN SCANNING TUNNELLING MICROSCOPY

A THESIS
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

VINCENT PALMER
Norman, Oklahoma
2023

DEVELOPMENT AND TESTING OF AN FPGA-CONTROLLED SWITCHED-INTEGRATOR CURRENT AMPLIFIER FOR USE IN SCANNING TUNNELLING MICROSCOPY

A THESIS APPROVED FOR THE DEPARTMENT OF ENGINEERING PHYSICS

BY THE COMMITTEE CONSISTING OF

Dr. Lloyd A. Bumm, Chair

Dr. Michael B. Santos

Dr. Clifford W. Fitzmorris

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Lloyd Bumm. You were the first person I spoke to or met at the University of Oklahoma, and without your guidance and advice I would not have been able to take the leap to leave the military, return to college, and pursue a graduate degree. The fact that I ended up in your research group seems almost pre-destined as a result, and this thesis would not have been possible without the guidance, education, and support you provided. To my committee members Dr. Santos and Dr. Fitzmorris, thank you for giving me a solid foundation in electronics and engineering, advice, and technical insight along the way, and finally for helping me complete this thesis. To Steven, thank you for the hours of conversation, for listening to me rage, and for being the least lame vegetarian/runner combo I've ever met. And last, but certainly not least, I would like to thank my beautiful wife Becca. Your support these years after I left a promising career to pursue my dreams has never wavered. I could not have asked for a better partner in crime, and I plan on spending the rest of my life paying you back for everything.

Abstract

The scanning tunnelling microscope (STM) is a very powerful analytic tool capable of achieving atomic resolution. Unfortunately, the STM is restricted to samples that are sufficiently conductive to allow adequate tunneling current for feedback control. The amplifier used to measure the tunneling current is the critical limiting component. If the amplifier could be made more sensitive, the STM could be operated at lower tunneling currents allowing lower conductivity samples to be studied. Most amplifiers used in STM employ a resistor feedback design, which become unstable at high gain necessitating a tradeoff between gain and bandwidth. One way to circumvent that stability problem is to use a capacitor feedback design (switched integrator), which does not exhibit the same stability problem. This comes at the expense of added complexity because the output is the integral of the current and needs to be periodically reset. In this project, a switched-integrator current amplifier is constructed and explored. It consisted of an analog switched integrator controlled by a field-programmable-gate-array (FPGA) with a 16-bit analog-to-digital converter and an 18-bit digital-to-analog converter. A viable prototype was created which allowed for the exploration of the gain, phase, and time delay of such systems. This exploration helped further characterize the important design considerations and trade-offs necessary for such a system. A design sequence is proposed that allows for optimal planning based on the desired tunneling current and system bandwidth.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview of Scanning Tunnelling Microscopy

The scanning tunnelling microscope (STM) can image surfaces with atomic resolution and has proven to be an extremely powerful tool in modern physics.[1] The technique was developed by Gerd Binnig and Heinrich Rohrer while working at IBM and demonstrated for the first time in 1983 (along with Gerber and Weibel),[2] earning them the 1986 Nobel Prize in Physics.[3]

In STM, an atomically sharp probe tip, typically made of tungsten or a platinum-iridium alloy, is positioned less than a nanometer above the surface of the material by a piezoelectric actuator. Once the probe tip is close enough, the electron wavefunctions of the probe tip and the sample will overlap, and quantum tunneling between the two will become possible. By applying a bias voltage between the probe tip and the sample, a net quantum tunnelling current will result. Because the tunnelling current is strongly dependent on the probe tip-sample separation, the tunneling current can be used to maintain the separation which avoids physical contact between the probe tip and the sample. One common STM operational mode achieves this by operating at a constant tunneling current. Negative feedback by the piezoelectric actuator is used to maintain the tunneling current at an operator-selected set point, and thus a nominally constant probe tip-sample separation. The difference between setpoint current and the measured current is the error signal which drives the piezoelectric actuator. For example, if the tunneling current is above the set point, the separation is too small, and the probe tip is moved away from the surface. If the current is below the set point, the separation is too large, and the probe tip is moved closer to the surface. This balance is maintained throughout the time when the STM is operating. This feedback minimizes the error signal and holds the tunneling current constant. To image the sample surface, the tip is scanned across the material in a raster pattern, and the tip height at constant tunneling current is recorded for each point in the pattern, generating an electronic topographic map of the sample. The corresponding z-coordinate can then be mapped as a function of the x- and y- coordinate to create an image of the sample surface.[4] An example of an STM image is given in Figure 1.1. The amplifier that measures the tunneling current is a critical component of the STM and is the focus of this project.

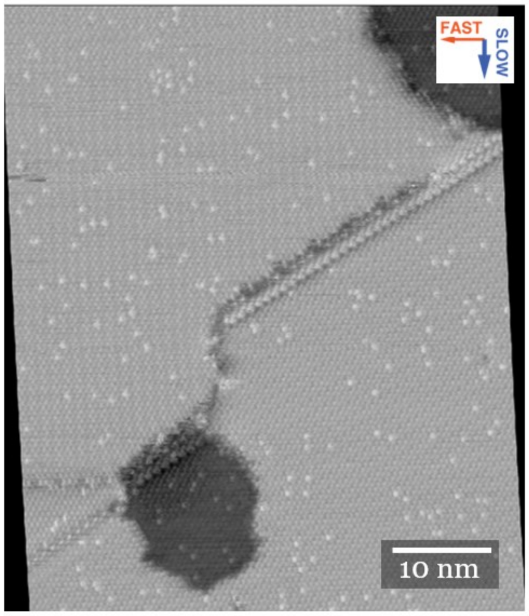


Figure 1.1:STM example image. DHCT-corrected STM image of alkanethiol-on-Au(111) self-assembled monolayers. The dark regions are topographically lower than the lighter regions and represent regions of gold vacancy islands. The monolayer is composed of a decanethiol host (10 carbon backbone) containing a small fraction of longer undecanethiol guest molecules (11 carbon backbone). The bright spots represent the longer chains on a sea of smaller chains.[5]

While it was initially believed that the technique mapped solely the topography of the surface of a material, in 1996 Diebold showed that the STM map was in fact a map of the electronic structure of the material.[6789] Thus, an STM image represents the convolution of the physical and electronic properties of the surface. By changing the imaging conditions (e.g., sample bias or the set-point current), it is possible to probe different electronic states or atoms within a sample. This ability to provide atomic resolution of materials surfaces has proven extremely valuable and has even recently allowed for the study of quantum superposition of electrons in monolayer graphene.[10]

1.2 Motivation

Controlling an STM requires a sophisticated set of electronics and a computer to process the measurements into a gray-scale image like in Figure 1.1, and a simplified diagram of these components is show in Figure 1.2. The tunnelling current must be measured and conditioned, most commonly by a current-to-voltage amplifier, to a voltage level that can interface with the

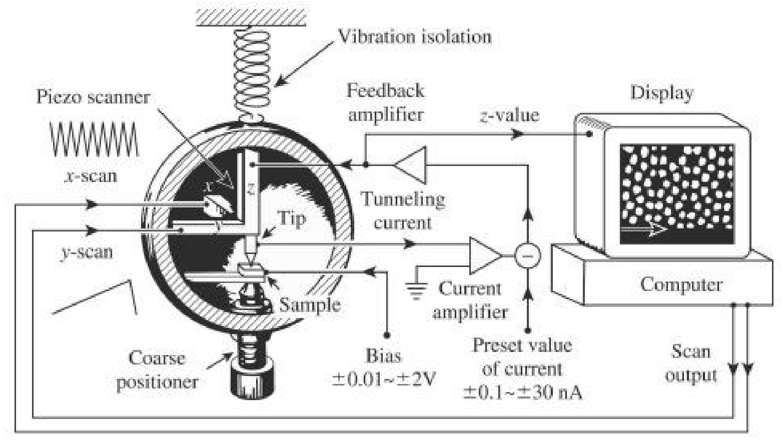


Figure 1.2: Diagram of a scanning tunnelling microscope. The tunnelling current is amplified to interface with the STM control electronics which raises or lowers the tip, and the tip height in relation to the x and y position on the sample is stored to generate the contour map from Figure 1.1.[1]

STM control electronics. The performance of this current amplifier is a key design consideration for the STM.

Standard STM set-ups use transimpedance amplifiers, which use resistors as the feedback mechanism. The advantages of this set-up follow.

1. Simple to use: The gain, and therefore the voltage generated by the tunnelling current, can be easily manipulated by changing the resistance of the feedback resistor in accordance with Ohm's law.
2. Ease of processing: Since the output voltage is directly proportional to the input current, extracting the tunnelling current requires little processing.
3. History of success: Transimpedance amplifiers have historically worked very well for samples with relatively high conductance. They have been employed for imaging with tunnelling currents as low as 0.1 pA.

However, two issues exist that severely limit resistor-feedback systems. The first is that the parasitic capacitance in the feedback resistor results in a pole of the transfer function that limits the bandwidth of the amplifier. The second is that the input capacitance creates a zero in the transfer function which can lead to oscillation. This forces a compromise between bandwidth and gain. One way around this is to use a capacitor as the feedback element instead of a resistor, which has been used successfully in the design of patch clamp amplifier.[11][12] These systems are more stable than their resistor feedback counterparts but are more complicated to implement. This type of feedback may allow for the detection of smaller tunnelling currents without sacrificing bandwidth, which would allow for application of STM to samples that are intractable by current systems due to low surface conductance. Section 2.2 discusses the theoretical implementation of these current amplifiers, and their actual usage is discussed in the following chapters.

1.3 Overview of the Project

The intent of this project was to explore the feasibility of using an integrating (capacitor-feedback) amplifier for use as an STM current amplifier. The ultimate objective is to achieve STM operation with tunneling currents below 1 pA with a bandwidth of 2 kHz or greater with a current signal to noise ratio (SNR) greater than 10. To interface with existing STM control electronics, the output voltage must be proportional to the current. Because the current amplifier is part of the STM feedback loop, the phase margin of the system will need to be >45° to assure stable operation, which places limits on the group delay to below 200 μs.

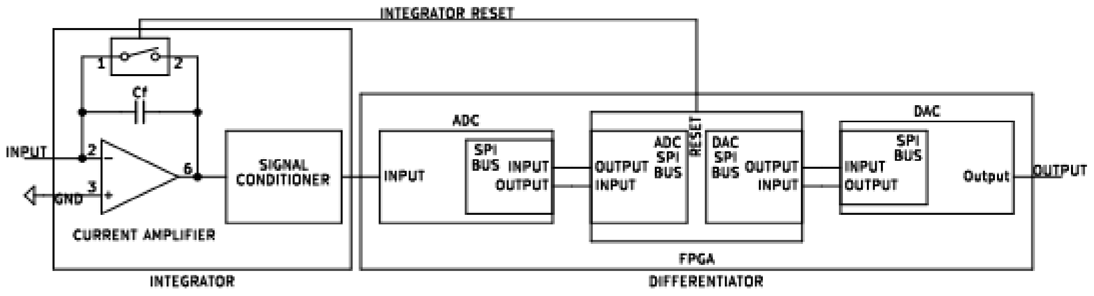


Figure 1.3: Conceptual diagram of the capacitor feedback operational amplifier.

There are two required components for this design, the integrator, and the differentiator. In this project, the integrator was implemented on a printed circuit board with additional support components to provide the necessary input current and output voltage conditioning. Differentiation was accomplished by digitizing the signal with an analog-to-digital converter (ADC), differentiating it with a field-programmable gate array (FPGA), and outputting the processed signal with a digital-to-analog converter (DAC). Additionally, the FPGA controlled the ADC, DAC, and the reset switch across the capacitor in the integrator. A simplified diagram of the general set-up is shown in Figure 1.3.

1.4 Scope of the Thesis

This thesis will cover the theory, general design, prototyping, and implementation of a switched integrator capacitor feedback operational amplifier system that could be used for STM. Chapter 1 introduced the theory of STM, the motivation for studying capacitor-feedback operational amplifiers, and the general overview of the project. The information in Chapter 2 lays the theoretical groundwork for understanding the limitations of traditional STM current amplifiers, and how capacitor-feedback can work around these limitations. Chapter 3 breaks down the hardware and components used in the system to a greater extent and discusses what each piece of the system does. Chapters 4 and 5 break down into detail the two major components. Chapter 4 describes the differentiator, specifically the theory behind the low-pass differentiating filter used as well as the software that controls the ADC, DAC, reset switch, and implementation of the filter. Chapter 5 covers the detailed PCB board implementation of a precise switched integrator. This chapter shows that the system works as intended and highlights the important design criteria revealed by these tests for future implementation. Chapter 6 summarizes the results from Chapters 5, discusses a possible design sequence based on the necessary design criteria, and recommends possible improvements for future similar systems.

1.5 References

1. Binnig, G.; Rohrer, H., Scanning tunneling microscopy from birth to adolescence. *Reviews of Modern Physics.* **1987,** *49***,** 615-625.
2. Binnig, G.; Rohrer, H.; Gerber. C.; Weibel, E., Surface studies by scanning tunnelling microscopy. *Phys. Rev. Lett.* **1982,** *49***,** 57-61.
3. The Nobel Prize in Physics 1986. NobelPrize.org. Nobel Prize Outreach AB 2023. Fri. 7 Apr 2023. https://www.nobelprize.org/prizes/physics/1986/summary/.
4. Chen, J., *Introduction to Scanning Tunnelling Microscopy.* Second Edition, Oxford University Press, 2008.
5. Yothers, M., High-precision measurements of alkanethiol self-assembled monolayer structure with scanning tunneling microscopy. Univ Oklahoma. PhD Dissertation. 2020. https://shareok.org/handle/11244/325434.
6. Diebold, U.; Anderson,J.; Ng, K.; Vanderbilt, D., Evidence for the Tunneling Site on Transition-Metal Oxides: TiO2. *Phys. Rev. Lett*. **1996**, *77***,** 1322-1325.
7. Hembacher, S.; Giessibl, F. J.; Mannhart, J.; Quate, C. F., Revealing the hidden atom in graphite by low-temperature atomic force microscopy. *Proc. Natl. Acad. Sci.* **2003,** *100* (22), 12539-12542.
8. Feenstra, R. M., Scanning tunneling microscopy and spectroscopy of gold on the GaAs(110) surface. *J. Vac. Sci. Technol. B* **1989,** *7* (4), 925-930.
9. Bumm, L. A.; Arnold, J. J.; Dunbar, T. D.; Allara, D. L.; Weiss, P. S., Electron transfer through organic molecules. *J. Phys. Chem. B* **1999,** *103* (38), 8122-8127.
10. Liu, X.; Farahi, G.; Chiu, C.; Papic, Z.; Watanabe, K.; Kenji, T.; Takashi, Z.; Zaletel, M.; Yazdani, A., Visualizing broken symmetry and topological defects in a quantum Hall ferromagnet. *Science*. **2021,** *375,* 6578, 321-326.
11. Finkel, A. S., Progress in instrumentation technology for recording from single channels andsmall cells. In *Cellular and Molecular Neurobiology: A Practical Approach*, Chad, J.; Wheal, H., Eds. Oxford UniversityPress: 1991.
12. Sigworth, F. J., Electronic Design of the Patch Clamp. In *Single-Channel Recording*, Sakmann, B.; Neher, E., Eds. Springer US: Boston, MA, 1995; pp 95-127.

Chapter 2: STM Current Amplifiers

The current amplifier is a key component of the STM that imposes limits on the gain and bandwidth of the tunnelling current, which is fundamental in measuring the tunnelling current in STM. Bandwidth determines the response time of the STM, and sufficient bandwidth is required required for the STM to be able to accurately follow the contours of the sample and react to variations in the surface without crashing. The gain limits the tunnelling current (and therefore the conductance of materials) that can be studied. These amplifiers that convert current to voltage are called transimpedance amplifiers. Converting current into voltage is useful because most electronic systems encode information as voltages, making it easier to interface with other components.

The main factor that needs to be controlled to ensure a successful operational amplifier set-up is the signal-to-noise ratio. If the noise on a signal is too high, it may be difficult, if not impossible, to recover the desired signal in a timely manner. Increasing the gain of a system will also increase the amplitude of contributing noise sources. Different amplifier configurations will have different noise sources and responses to them. This chapter discusses the noise sources in transimpedance amplifiers, how different amplifier configurations work, and the resulting noise analysis that goes along with those amplifier configurations.

2.1 Noise Sources in Transimpedance Amplifiers

The three major noise sources relevant to STM amplifiers are shot noise, thermal noise, and kTC noise. Understanding what causes them and the effect they have on the measurement allows for systems to be designed that manage noise sources to improve the signal-to-noise ratio.

2.1.1 Shot Noise

Shot noise results from the discrete nature of electric charge. Current results from the movement of electrons. When current is high, the movement of the electrons follows the law of high numbers, the relative number of statistical fluctuations will be statistically small. However, at lower currents, the relative number of statistical fluctuations will increase, causing noise in the current. The equation for RMS shot noise is

$$N_s = \sqrt{2eIB},$$

where *e* is the electron charge, *I* is the average dc current, and *B* is the frequency bandwidth.[1] The resulting SNR is

$$SNR_s = \sqrt{\frac{I}{2eB}}.$$

Shot noise is an inherent property of the current being measured. The SNR at 1 nA, 1pA, and 1 fA in 2 kHz bandwidth gives SNRs of 1200, 40, and 1.2, respectively. At high current, the effect of shot noise is generally less relevant, and if other noise sources dominate can be neglected. Shot noise can begin to dominate as other noise sources are reduced or when measuring a very small current.

Because the shot noise is fundamental to the tunneling current, considering the shot noise alone estimates the expected current noise from a noiseless current amplifier. Using the target parameters from Section. 1.3 (SNR of 10 in a 2 kHz bandwidth), the smallest practical current the STM can operate at is 64 fA. In practice the STM loop bandwidth is effectively lower than this target because the STM feedback is primarily an integrator, and the loop can be slowed below the 2 kHz bandwidth to reduce the noise.

2.1.2 Thermal Noise

Thermal noise, also known as Johnson-Nyquist noise, is a phenomenon that arises from the thermal motion of electrons. The fluctuation of electrons within the resistor gives at any instant a voltage across the resistor. The RMS voltage noise that results in a resistor with resistance *R* is

$$N_J^V = \sqrt{4k_BTBR},$$

and the resulting RMS current noise is

$$N_J^I = \sqrt{\frac{4k_BTB}{R}}.$$

2.1.3 kTC Noise

Ideal capacitors are lossless devices. However, Johnson-Nyquist noise on the rest of the circuit will impress a voltage fluctuation on the capacitor. This results in thermal noise on the capacitor known as kTC noise.[2] The RMS voltage noise is

$$N_K^V = \sqrt{k_B T / C},$$

and the charge noise is

$$N_K^Q = \sqrt{k_B T C}.$$

An interesting consequence of this is the presence of reset noise left on a capacitor when a switch is opened. The random thermal motion of the charges is then frozen onto the plates of the capacitor, resulting in the presence of noise while the capacitor is discharging, which can be the limiting source of noise for capacitive sensors.[3]

2.2 Amplifier Types

Response to the above noise sources is an important consideration when developing an STM. While there are numerous possible amplifier configurations that could be used in an STM, the focus of this thesis will be limited to those based on operational amplifiers. While many STMs use resistor feedback for their operational amplifiers, capacitive feedback has many characteristics that may be useful in low tunneling current applications. This next section differences between these two types of feedback.

2.2.1 Resistor Feedback

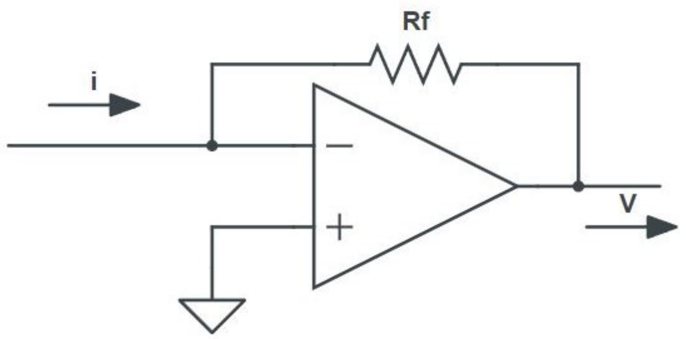


Figure 2.1: Simplified diagram of a resistor feedback operational amplifier. Most STM's use current amplifiers with resistor feedback.

In the resistor feedback operational amplifier shown in Figure 2.1, the relationship between the current into the inverting terminal and voltage across the feedback resistor can be easily found from Ohm's law and the op amp golden rules[4]

$$V = -IR.$$

These types of operational amplifiers are sufficient for samples with relatively high conductance. The gain in a resistor-feedback amplifier is proportional to R, and the Johnson-Nyquist voltage noise is proportional to $\sqrt{R}$. Increasing the feedback resistor increases the gain faster than the noise, thus increasing the signal-to-noise ratio. However, the choice amplifier of gain must consider the entire circuit.

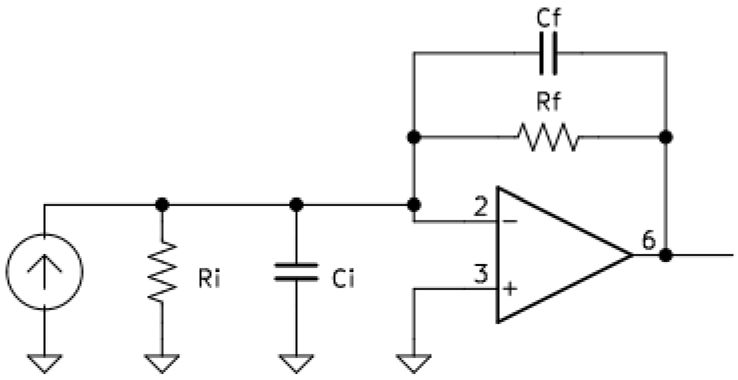


Figure 2.2: Elements contributing to stability and bandwidth in transimpedance amplifiers.

The schematic in Figure 2.2 shows a resister feedback amplifier, along with the feedback resistors parasitic capacitance, input capacitance, and input resistance. Resistors and the circuit wiring contribute parasitic stray capacitance, and the input capacitance and resistance are important elements in the response and stability of the amplifier. The stray capacitance and the feedback resistor contribute a pole at

$$f_P = \frac{1}{2\pi R_f C_f},$$

which results in a 3dB frequency reduction of the bandwidth at f_P.[5] When the denominator of the pole is large, the bandwidth becomes small. Thus, at a fixed C_f, there must be a trade-off between the gain and the bandwidth, since increasing R to increase the gain will decrease the bandwidth.[1] Additionally, the parallel sum of the capacitances and the resistances contributes a zero at

$$f_Z = \frac{R_f + R_i}{2\pi R_f R_i (C_f + C_i)},$$

which can affect stability and, in the worst case, cause the system to oscillate. Operational amplifier stability is discussed in section 2.3. A typical value for stray capacitance that has been reported in STM amplifiers is 0.5 pF.[1]

Since the voltage gain of the resistor feedback operational amplifier is also dependent on the resistance of the feedback resistor, this also inherently limits the conductance of materials that can be studied; if the conductance is too low, the gain-limited current-to-voltage amplifier will not be able to generate a large enough current to drive the feedback mechanism of the STM. There have been numerous methods developed to maintain an acceptable bandwidth while sampling low-conductance materials in STM. A few are discussed below.

1. Atomic Force Microscopy (AFM): AFM is another type of scanning probe microscopy that can measure forces as small as 10^{-18} N.[6] It has proved very successful in sampling many different materials but has roughly 10-100x less resolution.[7]
2. Cascading Amplifiers: Running the current through multiple stages of amplifiers, rather than one stage, can increase the frequency response of the amplifier, but results in an increase in noise because the internal noise of each amplifier is amplified by the subsequent stages.[89] For example, a two-stage amplifier can be used, where the first stage will typically have high gain and low bandwidth for stability and the second stage recovers the lost bandwidth from the first stage.
3. Another option is the use of composite amplifiers, where the last stage completes a feedback loop to the first stage. These designs can take the best elements of each amplifier while reducing their negative effects.[10]
4. Feedback Loop Compensation: A team in 2020 was able to achieve 28 kHz bandwidth with 10 times lower Johnson-Nyquist noise than traditional transimpedance amplifiers. They did this by applying a compensating capacitor and resistor to the feedback mechanism.[11] Another team devised a circuit with an operational amplifier that resulted in an additional "negative capacitance" seen across the feedback element of a transimpedance amplifier.[12]

2.2.2 Capacitor Feedback

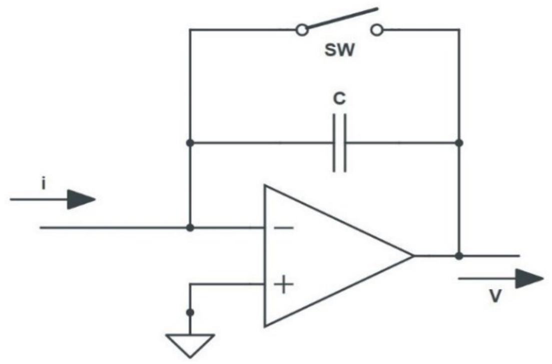


Figure 2.3: Schematic of a switched integrator operational amplifier. The switch is used in parallel with the feedback capacitor to allow the capacitor to discharge when nearing saturation.

Figure 2.2 shows the layout of a switched capacitor feedback operational amplifier. Unlike the resistor feedback system, when a capacitor with capacitance *C* is used as the feedback mechanism for these current-to-voltage amplifiers, the relationship between the input current *I* and the output voltage *V* is given by[4]

$$V_{out} = -\frac{1}{C}\int I_{in}\, dt.$$

Thus, the voltage depends on the sum total of the charge built up on the capacitor. So, whereas in a resistor feedback system a constant current into the amplifier would result in a constant voltage output, a constant current into a capacitor feedback system would result in a linearly increasing voltage output. The output would continue to increase until the amplifier saturates. Saturation needs to be avoided by the periodic discharge of the capacitor. This is accomplished with the in parallel with the capacitor, which keeps the capacitor in the range necessary to accurately measure the current being amplified. For defining the gain of the integrator, we consider the output quantity to be $\frac{dV_{out}}{dt}$ because it is instantaneously related to the input quantity I_{in}. The gain of the amplifier can be seen as the inverse of the capacitance of the feedback capacitor by taking the time derivative of the above equation, yielding the relation

$$\frac{\left(\frac{dV_{out}}{dt}\right)}{I_{in}} = -\frac{1}{C} = A_I,$$

where A_I is the gain of the integrator.

The STM feedback that drives the raising and lowering of the tip depends on the tunnelling current through the sample, not its integral. Therefore, the voltage signal needs to be differentiated to interface with the STM control electronics. While differentiating the signal adds another component, these amplifiers have improved noise performance compared to resistor feedback systems, particularly at the high frequencies and gains necessary to analyze samples with low conductance.[13][14][15][16]

Capacitor feedback systems are popular in the biological sciences through patch clamp amplifiers,[17] but these instruments require modifications and are usually too bulky to be mounted close to the scanning tip which increases the system noise.[18] However, other capacitive feedback systems have been developed and shown to be viable for different situations.[19]

2.3 Stability

Time delays in the system can cause stability issues with the STM. As the tip moves over the surface of the sample, the piezo needs to be able to respond to these changes. As the voltage from the changing tunnelling current varies, the amplifier will try to respond. Time delays cause the op amp to fail to detect that it has reached the correct output voltage, which causes it to overshoot. The op amp then must try to correct back to the desired output voltage. If the delay is small enough compared to the frequency of the input signal, this will result in damped successive feedback oscillations that eventually settles on the correct output. If the time delay is results in a phase approaching 180^{o} and a gain greater than 1, then positive feedback can result, causing the system to continuously oscillate and never settle to the correct value.[1]

2.4 References

1. Chen, J., *Introduction to Scanning Tunnelling Microscopy.* Second Edition, Oxford University Press, 2008.
2. Sarpeshkar. R; Delbruck, T.; Mead C., White noise in MOS transistors and resistors. *IEEE Circuits and Devices Magazine* **1993,** *9,* 6.
3. Fowler, B.; Godfrey, M.; Mims, S., Reset Noise Reduction in Capacitive Sensors. *IEEE Transactions On Circuits And Systems*, **2006,** *53*, 8.
4. Horowitz, P.; Hill, W., *The Art of Electronics.* Third Edition, Cambridge University Press, 2015.
5. Graeme, J., *Photodiode Amplifiers Op Amp Solutions.* McGraw-Hill, 1995.
6. Binnig, G.; Quate, C.; Gerber, C. Atomic Force Microscope. *Phys. Rev. Lett*. **1986,** *56*, 930.
7. Umbach, C. C.; Blakely, J. M., Development of a sub-picoampere scanning tunneling microscope for oxide surfaces. *Appl. Surf. Sci.* **2001,** *175-176*, 746-752.
8. Steer, M., *Fundamentals of Microwave and RF Design.* North Carolina State University Libraries, 2019.
9. Petersen, J. P.; Kandel, S. A., Circuit design considerations for current preamplifiers for scanning tunneling microscopy. *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena* **2017,** *35* (3), 033201.
10. Gift, S.; Maundy, B., Versatile composite amplifier configuration *International Journal of Electronics*, **2015,** *102*, 6, 993–1006.
11. Štubian, M.; Bobek, J.; Setvin, M.; Diebold, U.; Schmid, M., Fast low-noise transimpedance amplifier for scanning tunneling microscopy and beyond. *Rev. Sci. Instrum.* **2020,** *91* (7), 074701.
12. Xie, K.; Liu, Y.; Li, X.; Guo, L.; Zhang, H. Expanding the bandwidth of the ultra-low current amplifier using an artificial negative capacitor. *Review of Scientific Instruments*. **2004,** *75*, 2.
13. Burr-Brown, Noise Analysis of FET Transimpedance Amplifiers. *Burr-Brown Application Bulletin* **1994,** *SBOA060.*
14. Baker, B. C., Improved Noise Performance of the ACF2101 Switched Integrator. Ibid. **1993**, *SBOA032*.
15. Baker, B. C., Comparison of Noise Performance between a FET Transimpedence Amplifier and a Switched Integrator. *Burr-Brown Application Bulletin* **1994,** *SBOA034*.
16. Raczkowski, K.; Piasecki, T.; Rudek, M.; Gotszalk, T., Design and evaluation of precise current integrator for scanning probe microscopy. *Meas. Sci. Technol.* **2017,** *28* (3), 034013.
17. Sigworth, F. J., Electronic Design of the Patch Clamp. In *Single-Channel Recording*, Sakmann, B.; Neher, E., Eds. Springer US: Boston, MA, 1995; pp 95-127.
18. Carlà, M.; Lanzi, L.; Pallecchi, E.; Aloisi, G., Development of an ultralow current amplifier for scanning tunneling microscopy. *Rev. Sci. Instrum.* **2004,** *75* (2), 497-501.
19. Umbach, C. C.; Blakely, J. M., Development of a sub-picoampere scanning tunneling microscope for oxide surfaces. *Appl. Surf. Sci.* **2001,** *175-176*, 746-752.

Chapter 3: System Design

Before diving into the specific construction of the system, it is important to understand generally how each component works together as well as the general design philosophy of the system as a whole. Since the voltage across a switched integrator is proportional to the integral of the current into the input, the signal needs to be differentiated to extract the tunnelling current. This was accomplished by first integrating the input signal with the switched integrator. The output of the integrator is conditioned first by an anti-aliasing filter with a differential input. The former removes frequencies above the Nyquist frequency of the digital sampling. The latter suppresses common mode noise. It is then converted to a fully differential signal and input to the 16-bit analog-to-digital converter (ADC). Digital signal processing and differentiating is accomplished in the FPGA using a differentiating finite impulse response (FIR) filter and outputting the differentiated signal with a digital-to-analog converter (DAC). The output voltage can then be compared with the error voltage in the feedback electronics to determine the necessary adjustments to the tip height. A block diagram showing the general relationship of each component is shown in Figure 3.1. Each component is described in more detail in the following sections of this chapter.

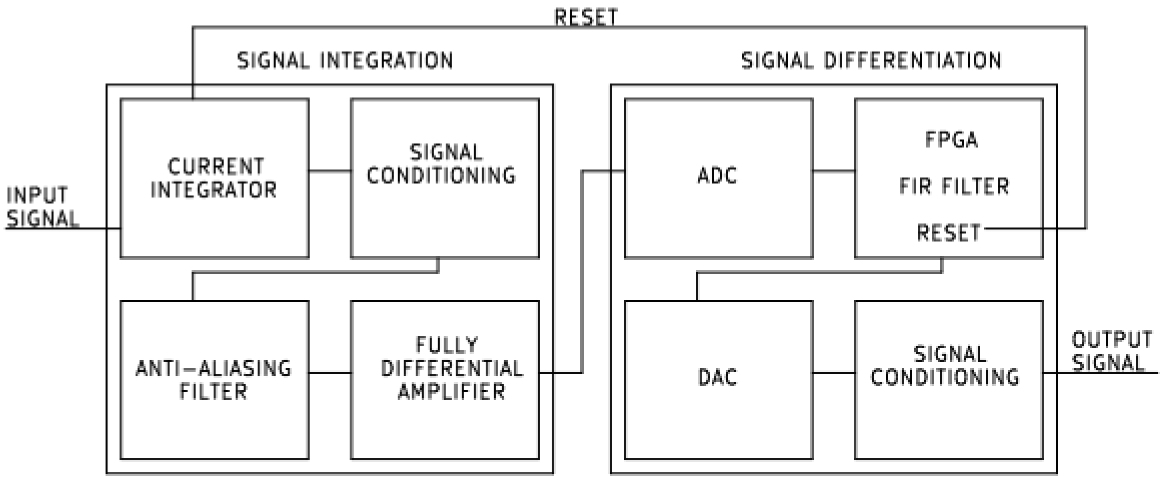


Figure 3.1: Block diagram of components of signal integrator and signal differentiator.

3.1 Signal Integration

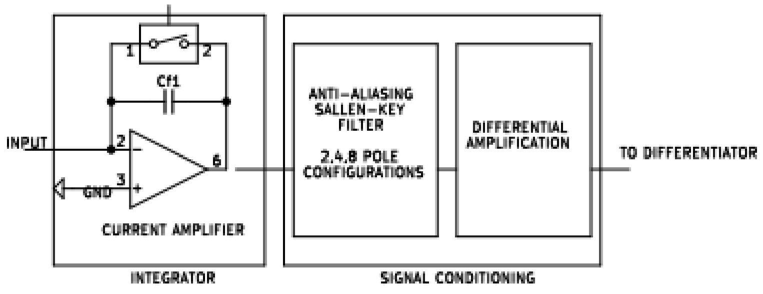

INPUT
2
3
6
GND
Cf1
1
2
CURRENT AMPLIFIER
INTEGRATOR
ANTI-ALIASING SALLEN-KEY FILTER
2,4,8 POLE CONFIGURATIONS
DIFFERENTIAL AMPLIFICATION
SIGNAL CONDITIONING
TO DIFFERENTIATOR

Figure 3.2: Block diagram of input integration components

Figure 3.2 shows the three major elements responsible for integrating and conditioning the input from the STM. The first, and most important component of this system is the switched integrator. It contains the capacitor feedback amplifier, the switch to discharge the capacitor, and all the additional electronics to process and amplify the signal. The tested switched integrator systems are covered in Chapters 5 and 6.

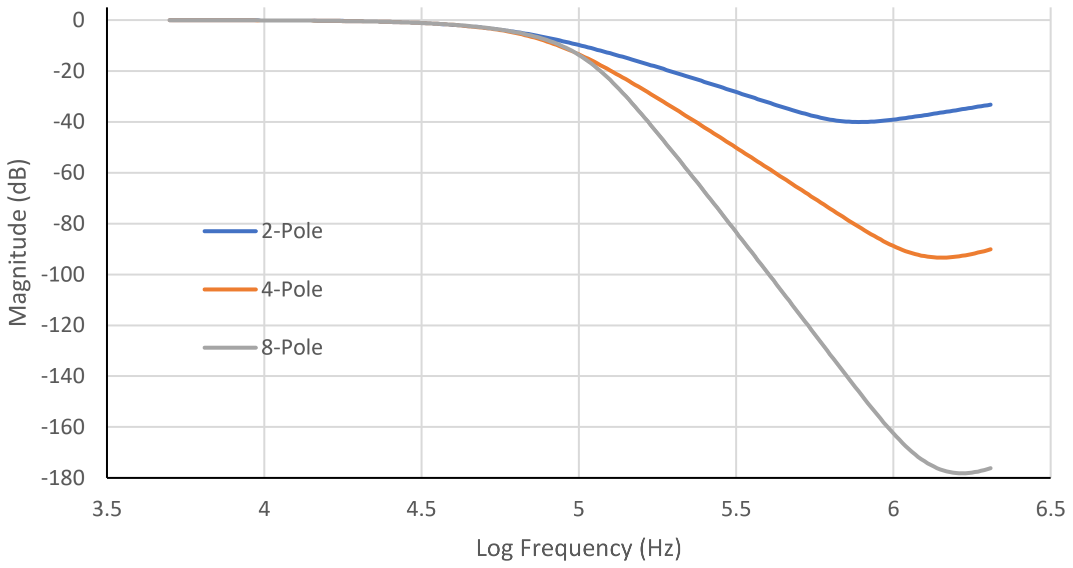


Figure 3.3: Magnitude response of the Sallen-Key anti-aliasing filters. While all three filters have the same pass-band and stop-band, the attenuation provided by the 8-pole filter is significantly stronger than the other two.

Next the signal is processed through an anti-aliasing filter. Aliasing occurs when a signal over the Nyquist frequency (discussed in Chapter 4) is sampled, causing false frequencies to appear in a signal.[1] High frequency noise signals can cause the appearance of features in the data that are not actually there. The anti-aliasing filter is a low-pass filter, preventing signals with frequencies above the Nyquist frequency from entering the ADC. The system is jumper selectable for 2, 4, and 8-pole Sallen-Key active low-pass filter as well as an 8-pole clock-tunable Bessel filter. The schematics are included in Appendix A.3, and plots of the Sallen-Key filters performance are shown in Figure 3.3 and Figure 3.4. The 8-pole filter has the strongest attenuation, reaching −87 dB at the Nyquist frequency of 333 kHz while introducing a 10 µs group delay. This time delay can combine with other time delays in the system to produce a frequency dependent phase shift which can cause oscillations of the STM control loop at frequencies where the phase reaches 180°, as discussed in Section 2.2.1.

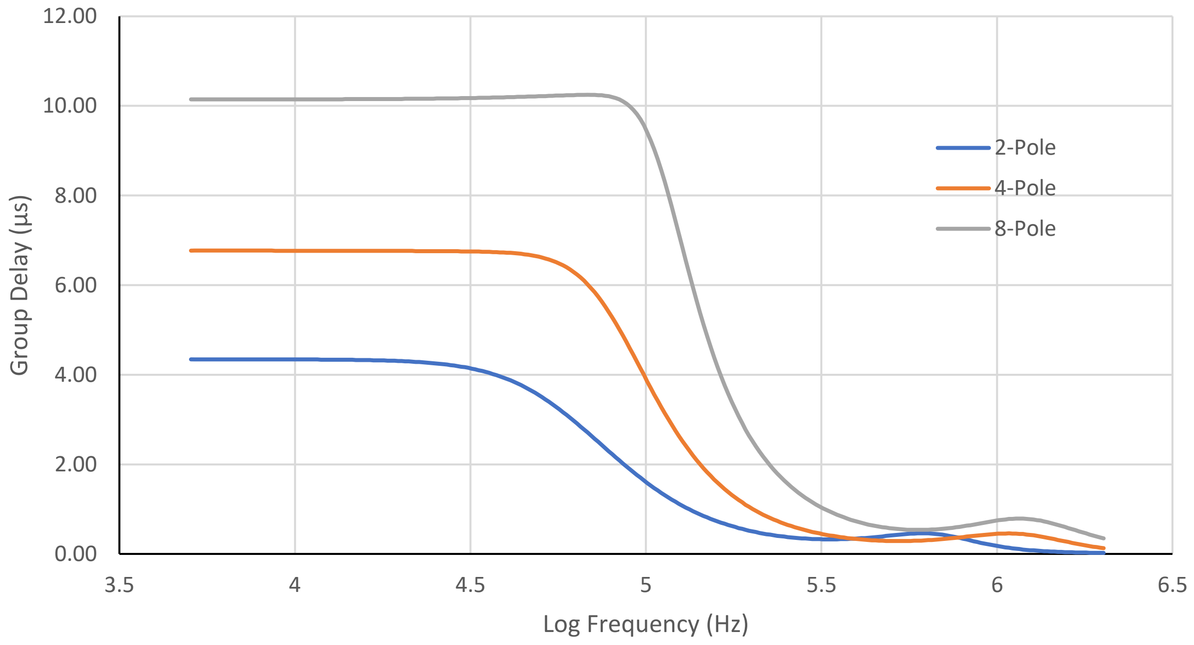


Figure 3.4: Group delay of the Sallen-Key anti-aliasing filters. The consequence of the increased attenuation provided by the 8-pole filter is the increase in the group delay.

The signal is then amplified by a fully differential amplifier as required by the ADC, which also provides immunity to common mode noise such as ground loops. The fully differential amplifier used was the ADA4940-1ARZ differential amplifier[2], which converts the single-ended signal out of the integrator into a fully-differential signal to be processed by the ADC. This board also contains components that condition the output signal, which will be discussed in the next section. The schematics for the set-up used are included in Appendix A.4.2. From here, the signal is then transmitted to the differentiator.

3.2 Signal Differentiation

Figure 3.5 shows a circuit diagram of the set-up of the ADC, DAC, and FPGA. Communication between these components used a serial peripheral interface (SPI) bus, which is designed for short distances. To communicate between the PCB boards, ribbon cables with alternating signal and ground wires connections were implemented to reduce crosstalk and control the impedance as well as added 100 Ω source resistors to attenuate the signal reflections. After the processed signal is output by the DAC, it is first conditioned by a AMP03GS precision, unity gain differential amplifier[3] and is then buffered by a unity gain line driver OP27G low-noise precision amplifier.[4] An external power source providing ±15V was also built to power the entire set up.

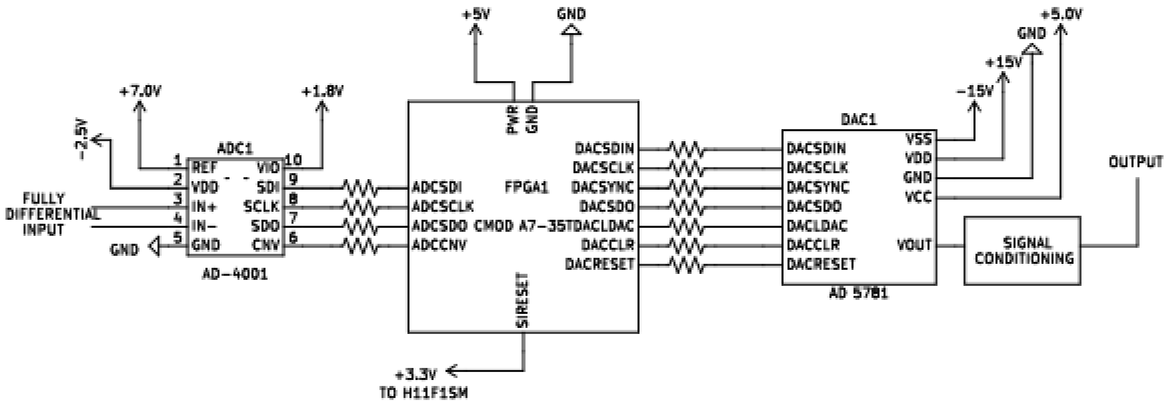


Figure 3.5: Schematic of the ADC, FPGA, and DAC.

The ADC used was an Analog Devices AD4001. The AD4001 is a low power, high accuracy analog to digital converter. It outputs a 16-bit integer which has a range of −5V to +5V, yielding a resolution of 152.6 µV. It can convert and output a maximum of 2 MSPS and communicates via a modified SPI protocol. Figure 3.6 shows the functional block diagram of the AD4001.

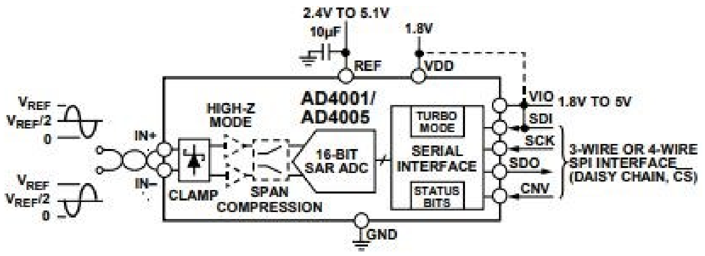


Figure 3.6: Functional block diagram of the AD4001 Analog-to-Digital Converter.[5]

The DAC used was an Analog Devices AD5781. The AD5781 is a low power, high accuracy digital-to-analog converter. It takes in a 24-bit instruction, consisting of a 4-bit register access command, an 18-bit integer corresponding to the desired output voltage in the range of −10 V to +10 V, and two additional bits that are not used. This yields a resolution of 76.3 μV. It can accept a maximum clock frequency of 35 MHz and communicates via SPI. Figure 3.7 shows the functional block diagram of the AD5781.

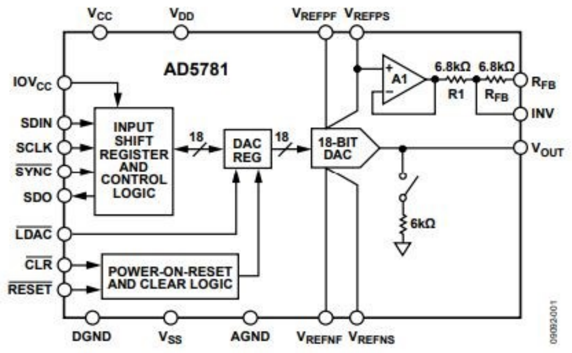


Figure 3.7: Functional block diagram of the AD5781 Digital-to-Analog Converter.[6]

The AD4001 and the AD5781 were controlled via a CMOD-A7 35T which is built around a Xilinx Artix-7 FPGA. The VHDL-2008 code used to control both components works in parallel to minimize downtime between reading data in from the ADC and outputting it to the DAC. Additionally, the low-pass differentiating FIR filter discussed later in Chapter 4, as well as the integrator reset controller, was programmed into the FPGA. The code controlling these three processes is included in the Appendix. The CMOD-A7 was modified from its standard 12 MHz clock to an 80 MHz clock, which theoretically would allow up to 1.33 M samples per second. Through experimentation, it was determined that the maximum reliable throughput of this set-up of 667 k samples per second was achieved by slowing the clock down to 20 MHz. We believe the limitation was SPI inter-board connection. Figure 3.8 shows a picture of the CMOD-A7 35-T.

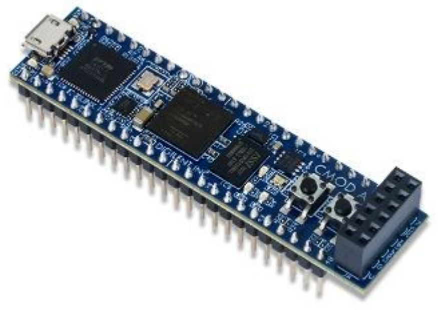


Figure 3.8: Picture of the CMOD-A7 35T.[7]

3.3 References

1. Finlay, D. J .; Dodwell, P.C.; Caelli, T.M.; The wagon-wheel effect. *Perception*. **1984,** *105* (13), 237–248.
2. Analog Devices, ADA4940-1/ADA4940-2 Datasheet Rev E., Norwood MA **2018**. https://www.analog.com/media/en/technical-documentation/data-sheets/ADA4940-1_4940-2.pdf.
3. Analog Devices, AMP03 Datasheet Rev E., Norwood MA, **2003**. https://www.mouser.com/datasheet/2/609/AMP03-1503769.pdf.
4. Analog Devices, OP27 Datasheet Rev H., Norwood MA, **2015**. https://www.analog.com/media/en/technical-documentation/data-sheets/OP27.pdf.
5. Analog Devices, AD4001/AD4005 Datasheet Rev C., Norwood MA, **2021**. https://www.analog.com/media/en/technical-documentation/data-sheets/ad4001-4005.pdf.
6. Analog Devices, AD5781 Datasheet Rev E., Norwood MA, **2018**. https://www.analog.com/media/en/technical-documentation/data-sheets/AD5781.pdf.
7. Digilent, CMOD A7 Reference Manual, Pullman, WA, Date Unknown, https://digilent.com/reference/programmable-logic/cmod-a7/reference-manual.

Chapter 4: Low-Pass Differentiation

4.1 Standard FIR Filters

The output of the capacitor feedback operational amplifier is an example of a continuous time signal. Thus, this system takes an input system *I(t)*, and outputs a transformation of that data *O(t).*

$$O(t) = \mathrm{T}\{I(t)\}.$$

This transformation can be accomplished through the implementation of a carefully constructed filter. Finite impulse response filters are filters that have no feedback into the filter and whose impulse response only accounts for data within a specific duration. The general formula used for a FIR filter is

$$O(n) = \sum_{i=0}^{N} c_i I(n-i),$$

where *N* is the number of taps/samples used to filter the data, and *c* is the *i*th coefficient, or impulse response. Outside of these values, *c* is zero, which means that for a sampling time of t_s, the total impulse response duration is $t_s\ (N\text{-}1)$. At the next sampling instance, *I(0)* is added to this register as *i=0*. The other values in the register are shifted down by one; *i=0* becomes *i=1*, *i=1* becomes *i=2*... until the last value at *i = N-1*, which is shifted out of the register.

In a standard direct-form FIR filter, when viewed in the time domain, the values of c_i create a moving "window" through which certain values are used to process the signal, and outside of the window the values are discarded. This mathematically is represented by a convolution

$$O(t) = I(t) * W(t),$$

where *I(t)* is the input signal and *W(t)* is the window function.[1] Different window functions can be convolved with the input signal depending on the desired frequency response of the filter[2].

The values of the coefficients in a FIR filter can be designed such that it can perform as a variety of filters, e.g., high pass, low pass, or in this application, a low-pass differentiator. Proper design of the filter coefficients can allow certain frequency bands through while attenuating others. For a switched integrator STM, a low pass differentiating filter is desirable to help eliminate high frequency noise while still effectively extracting the tunnelling current. Additionally, the characteristics of the FIR filter can be tuned such that it acts as a linear phase filter in the desired frequency region, which ensures that the group delay for the system will be constant, preventing signals with different frequencies from shifting relative to one another.

Increasing the number of taps in a FIR filter will generally improve the desired characteristics, to include the sharpness of the transition from pass band to stop band and the amount of attenuation in the cut-off band. However, this will introduce a time delay equal to roughly half of the total impulse response duration. This time delay needs to be considered along with other time delays in the system, such as the anti-aliasing filter delay from Section 3.1.1. If the total time delay results in a phase shift of 90° or greater, this can lead to instability and oscillation of the STM control loop as discussed in Section 2.2.1.

The Nyquist frequency is another critical component to ensure proper design and functioning of a FIR filter. The Nyquist frequency of a filter is the maximum input signal frequency that filter can sample without aliasing; sampling signals above this frequency can result in the appearance of false signals. The Nyquist frequency is half of the sampling frequency, meaning if a filter samples a signal at 2 MHz, the Nyquist frequency of that filter would be 1 MHz.[3] Thus, increasing the sampling rate (which corresponds to decreasing the time between samples, or the sampling time, of a filter) will increase the bandwidth of the filter.

4.2 Low-Pass Differentiating FIR Filter

An interesting algebraic property of convolutions which can easily be seen from the definition of a convolution is that the derivative operator is communicative, i.e.

$$\tfrac{d}{dt}I(t) * W(t) = I(t) * \tfrac{d}{dt}W(t).$$

Thus, the derivative of an input signal *I(t)* can be extracted by running it through a filter with the derivative of the desired window function.[4] This property is useful, since as discussed in Section 2.1.1, the output of a capacitor feedback operational amplifier is related to the integral of the current into it. The differentiating FIR filter thus also allows for the recovery of the input tunnelling current. A consequence of this is that one of the key features of a differentiating FIR filter is the coefficients are anti-symmetric.

The gain of a differentiator in the time domain is proportional to the RC time constant

$$V_{out}(t) = RC\left(\frac{dV_{in}}{dt}\right).$$

Unit analysis of this gain shows that the gain has units of seconds. The gain of a non-inverting differentiator can be expressed in the frequency domain as

$$\frac{V_{out}(\omega)}{V_{in}} = j\omega RC,$$

which shows that gain is proportional to frequency and has an output phase shift of +90°. In the linear response region, this means that high frequencies will be amplified more than low frequencies. The required dynamic range of gain can create challenges in the required accuracy of the coefficients and the dynamic range of the DAC.

For a practical differentiator, the frequency range of differentiation must be limited. The FIR filter gain also needs to fall below the least significant bit of the DAC at or below the Nyquist sampling frequency to prevent aliasing noise from being amplified by the differentiator. Steep fall-off to and strong attenuation in the stop band generally requires many coefficients but this requirement can be relaxed due to the additional attenuation provided by the pre-ADC low-pass anti-alias filter.

4.3 Software Design

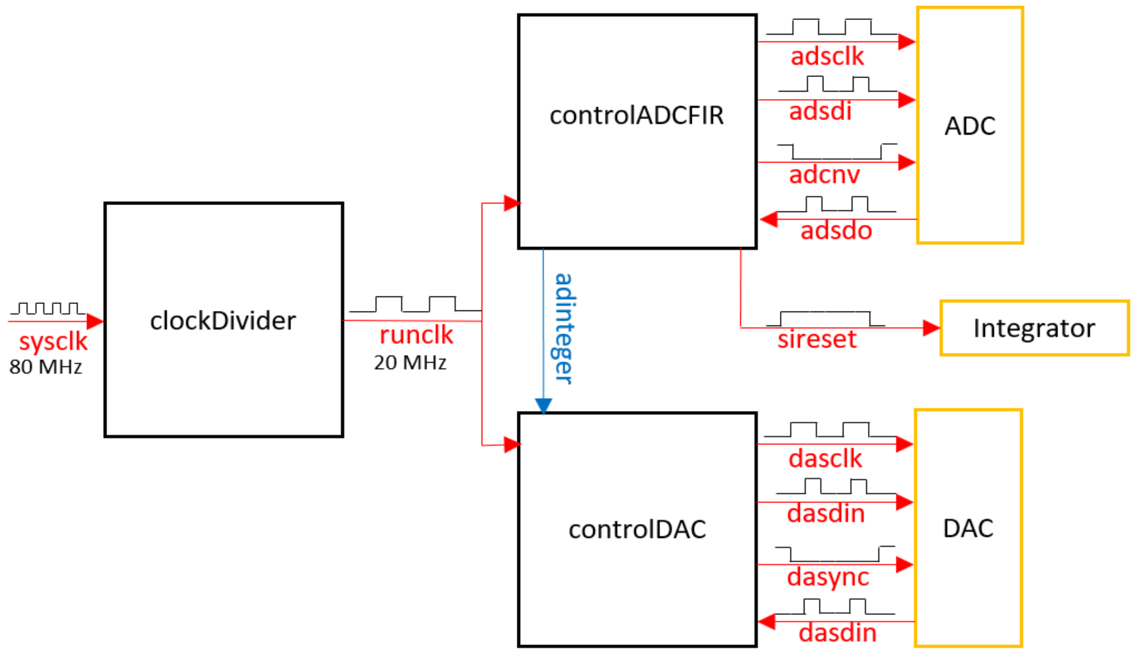


Figure 4.1: VHDL-2008 code block diagram.

The VHDL-2008 code is included in Appendix A.2. It features three major processes; one to generate the 20 MHz clock, one to control the ADC and the FIR filter, and one to control the DAC. Figure 4.1 shows the block diagram of the relationship between these three processes. Figure 4.2 shows the global timing diagram showing the coordination between the ADC, the DAC, and the FIR filter. The DAC is the rate limiting step, requiring 30 clock cycles to output one transmission. This process runs in parallel to first the combined ADC and FIR filter.

Process clockDivider: This process takes the 80 MHz and divides it down to 20 MHz, which is used to run the rest of the operations. Additionally, it generates a 2 MHz clock to control the anti-aliasing Bessel filter if it is in use.

Process controlADCFIR: The ADCNV pin is taken low on the first clock cycle, signaling the ADC to begin outputting its 16-bit signal. The ADC takes 20 clock cycles to output its 16-bit binary signal to the FPGA on the ADSDO pin, which stores it as the corresponding decimal integer. Upon completion of the ADC output, the ADCNV pin is taken high, which switches the ADC from outputting data to converting the input voltage into a new integer. This integer is then input into the FIR filter register, which takes five clock cycles to process the data. On clock cycle 26, the FPGA completes the transformed signal. Additionally, when the ADC outputs a value corresponding to a specified min or max limit, this is means capacitor is saturated beyond the limits that the ADC can measure and needs to be reset, which results this process sending a signal to the switch to initiate a reset. A buffer of ±4.8 V was used as the limits to keep the data

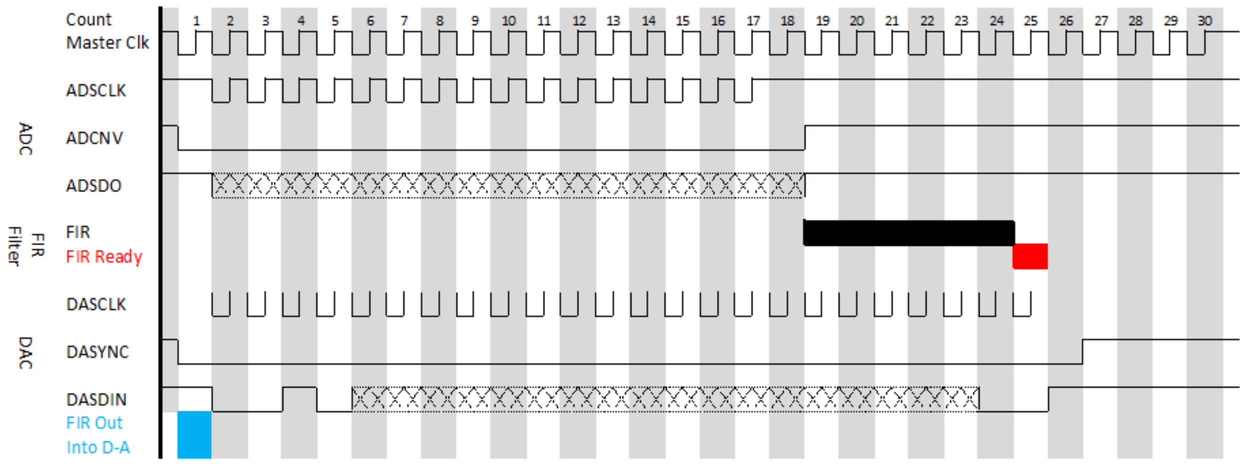


Figure 4.2: Differentiator master timing diagram. While the DAC is outputting processed data, the ADC is receiving new data to run through the FIR filter or converting data for the next transmission.

well within the operational limits of the ADC. The process for this is discussed in detail in Section 4.4.

Process controlDAC: On the first clock cycle of the subsequent iteration of the algorithm, the output of the FIR filter is shifted into the process controlling the DAC. On the second clock cycle, the DASYNC pin of the DAC is taken low to signal an incoming signal transmission. Over the next 24 clock cycles, the output of the FIR filter is converted into an 18-bit binary signal and output on the DASDI pin, along with the appropriate register access code. On clock cycle 28, the DASYNC pin is taken high to signal completion of a signal transmission and is held high long enough for the DAC to process and output the corresponding voltage before the whole cycle repeats again. Thus, on iteration *n* of the algorithm, the DAC is outputting the result of the *n-1* FIR processed ADC conversion.

The ADC can be used without configuration based on its standard settings, but the DAC requires that the configuration register be accessed to appropriately configure it before it begins converting transmitted values. At start-up, controlDAC outputs 1000 configuration transmissions to ensure DAC has received configuration data, and then begins outputting values to DAC for it to convert. This was initially done while designing the code for troubleshooting simplicity purposes; during power-on, the DAC and FPGA receive power at the same time. Identifying exactly when either component is ready was not attempted. Instead, hard-coding numerous configuration transmissions ensured the DAC receives at least one configuration message before controlDAC starts attempting to transmit data. It was not removed from the code because it was not found to negatively affect the performance of the system and because the total time for 1000 transmissions with a 20 MHz clock is only 1.5 ms.

4.4 Capacitor Reset

Since the ADC is limited to ±5 V, the capacitor will need to be reset before the ADC receives voltages at these limits. For the ADC, this corresponds to the integers 32767 (+5 V) and 32768 (−5 V) since it uses two's complement to express negative numbers. If controlADCFIR, detects that the ADC output is between 31400 (+4.8 V, rounded down) and 34100 (−4.8 V, rounded up), it will begin outputting a +3.3 V signal to the switch for 2 ms, during which time capacitor switch will close allowing the capacitor to reset. The 0.2 V buffer around the two listed values accounts for any offset errors in the input voltage and prevents the ADC from being stuck at the rails without being able to reset. During the reset time, the clock signals to the ADC and DAC will not be output, effectively freezing the input and output. Once 2 ms have passed, the entire process will restart. 2 ms was chosen as a safe value during testing that will was short enough that it did not significantly distort data in the expected operating ranges, but long enough that the capacitor had more than enough time to discharge fully. It is possible that as the system is eventually revised, this time could be reduced significantly.

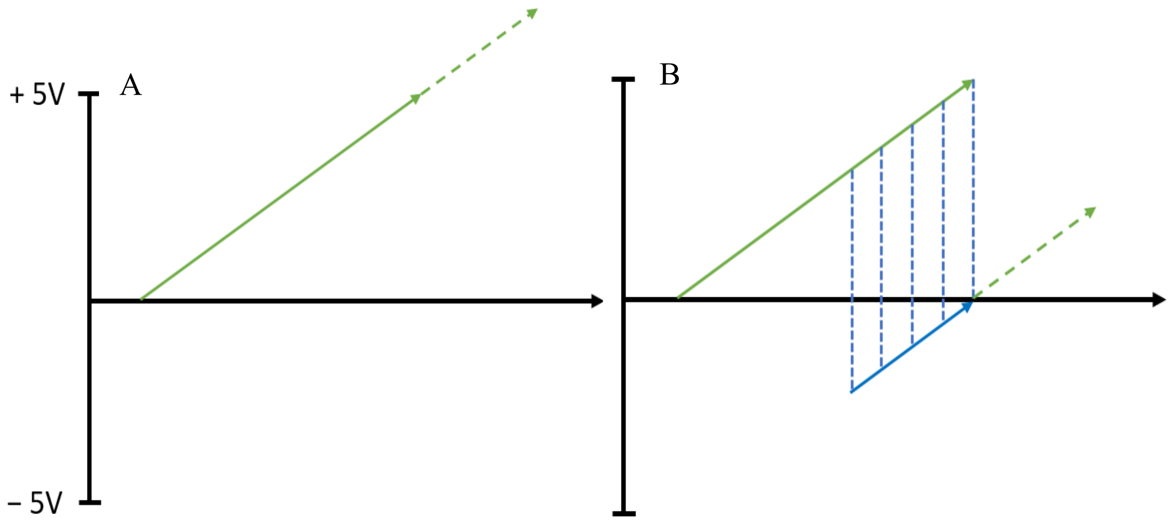


Figure 4.3: Graphical description of the correction made to the FIR register upon resetting. (A) A constant tunnelling current into the integrator will generate a linear voltage ramp (solid green line) that can exceed the limits of the ADC (dashed green line). (B). Resetting the integrator will introduce a false discontinuity in the output of the integrator, which will degrade the signal when differentiated. Subtracting the most recent value stored in the FIR register from every value in the register (dashed blue lines) will translate the signal so that the slope is preserved, but the discontinuity is removed (solid blue line).

After the reset, the fully discharged capacitor will start back at 0 V and resume integrating the signal. This will introduce a discontinuity in the output of the integrator that is not representative of the current input. If the old values in the FIR filter are used, this large discontinuity will propagate through the differentiator, resulting in a spike in the output of the differentiator until it has propagated through the FIR filter. If the values in the FIR filter are reset entirely to zero, there will be a delay as the filter refills. Because this is a differentiator, a continuous slope is expected in the data values in the filter and can be accomplished by patching the discontinuity. This patch is implemented on each reset by subtracting the most recent value in the FIR filter from all the data values. This adjusts the values in the register so that the most recent value is 0 V which will stitch with the next value after reset (close to zero) and continue the same slope as the signal before reset. A graphical representation of the process is found in Figure 4.3.

4.5 Design of Filter for Testing

Identifying the conditions under which a filter will be used is important for understanding how to optimize the filter for desired performance. The ideal differentiator would have a linear response up to the Nyquist frequency, and then a step-response drop off to extremely high attenuation. However, this is not possible, and trade-offs must be made to maximize performance.

1. Sharp cut-off: In order to achieve a very sharp cut-off, the number of coefficients in the filter must be very high. The trade-off to this is that the time delay of the filter increases linearly with the order of the filter, and since the STM requires real-time feedback, the time delay cannot be allowed to grow too large without resulting in control loop instability.
2. Large pass-band region: In order to have a pass-band that accommodates a large range of frequencies, the coefficients will have a range in magnitude. Coefficients with very high magnitude can cause the intermediate values within the FIR register to overflow due to integer limitations, and very small coefficients can reduce the intermediate values to below the resolution of the ADC or DAC and cause loss of information. An additional complication is the fact that VHDL-2008 is incompatible with floating-point numbers, so the floating-point coefficients had to be converted by multiplying the coefficients by a large enough power of 10 to achieve at most 6 integer digits per coefficient, and then dividing the sum by the same power of 10 at the end. If the disparity of the coefficient values is large enough, some of the smallest coefficients will express to zero, affecting the order of the filter and resulting in significant loss in information.
3. High attenuation: In order to achieve high attenuation in the stop-band, the order of the filter must be high, which circles back to the issues discussed above.

Considering the STM operation conditions, it is desirable to have high gain in the low frequency region to ensure that the ADC and DAC will be able to resolve the data. Extremely high attenuation was not prioritized because additional attenuation was achieved with the anti-aliasing filter. Numerous filters were used and tested throughout this project, and the filter described below was chosen for the tests in Chapter 5 because of the considerations discussed above. It was generated with the Filter Designer tool available in MATLAB[5] with the goal of achieving −90 dB attenuation below the Nyquist frequency, and with a linear response region up to 30 kHz to give ample bandwidth in which to test. Figure 4.4 shows the impulse response of this filter, and Figure 4.5 shows the magnitude response, both generated by MATLAB. Note that the coefficients are anti-symmetric as expected of a differentiator. The table of coefficients is included in Appendix A.1.

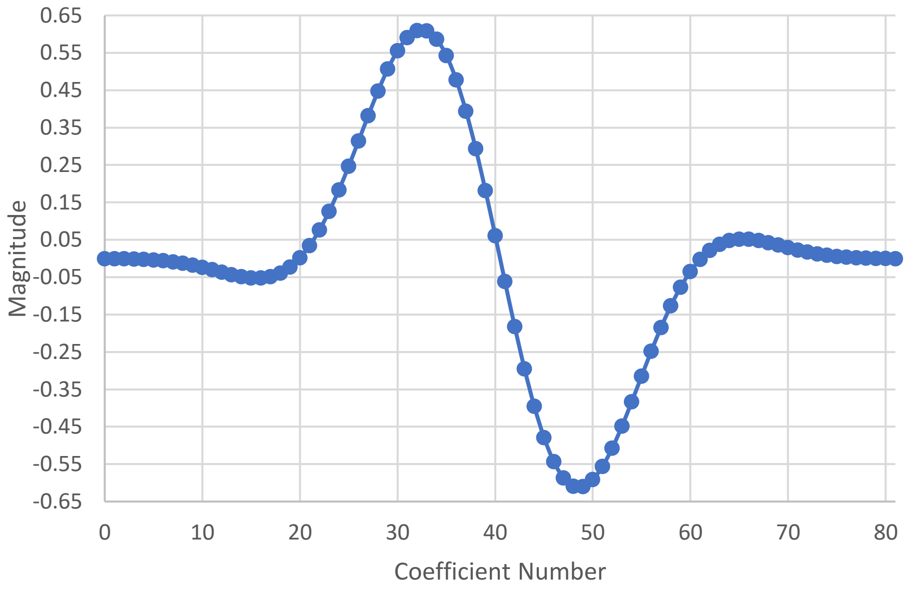


Figure 4.4: Impulse response of 81st order low-pass differentiating direct form FIR filter.

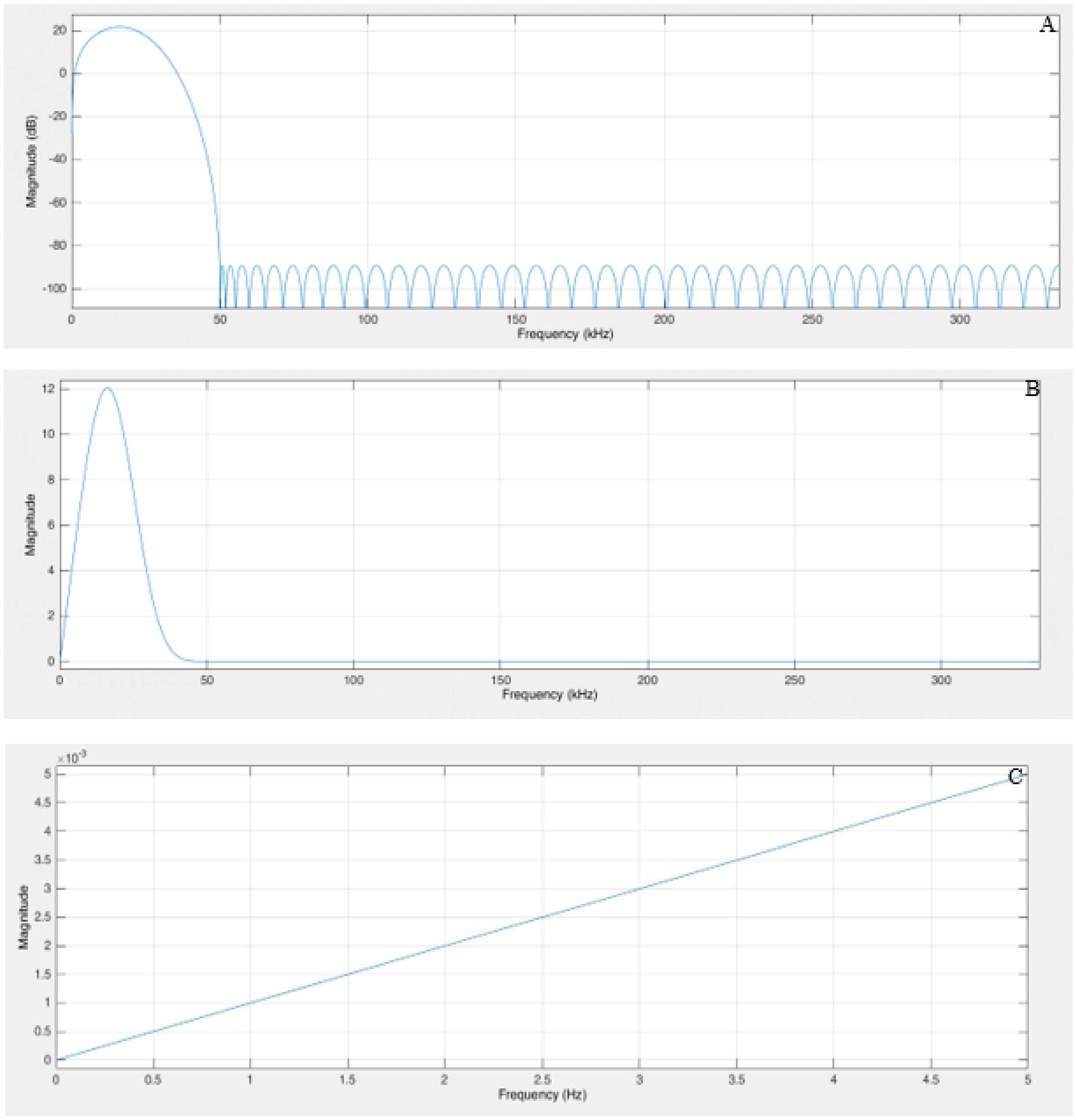


Figure 4.5: Magnitude response of 81st order direct-form differentiating FIR filter. (A) Magnitude (dB) vs frequency up to Nyquist frequency. (B) Magnitude (linear) vs frequency up to Nyquist frequency. (C) Magnitude (linear) vs frequency up to 5 Hz. This filter was determined through trial and error, and achieved the necessary compromises discussed in Section 4.5 to optimize performance for the tests in Chapter 5.

4.6 References

1. McClellan, J.; Schafer, R.; Yoder, M., *Signal Processing First.* Pearson Education Inc. 2003.
2. Prabhu, K., Window Functions and Their Applications in Signal Processing. CRC Press. 2014.
3. Shannon, C., Communication in the Presence of Noise. *Proceedings of the IEEE*, **1998,** *86*, 2.
4. Read, R., The convolution theorem and its applications. *Structural Medicine-Protein Crystallography Course*, University of Cambridge, 2009 https://www-structmed.cimr.cam.ac.uk/Course/Convolution/convolution.html.
5. MathWorks, Using Filter Designer, 2023 https://www.mathworks.com/help/dsp/ug/using-filter-designer.html.

Chapter 5: PCB Switched Integrator

There are trade-offs to be considered when designing a switched-integrator system. There is a trade-off between gain and system reset frequency dependent on the feedback capacitor. A smaller feedback capacitor yields higher integrator gain but will require frequent resets. Each reset introduces a deadtime that blinds the amplifier to changes of input current during the reset interval. To minimize disruption of STM operation, the resets must be carefully orchestrated into the STM control loop operation. See Section 2.2.2.

There is also a trade-off between time delay and noise attenuation related to the number of coefficients in the FIR filter. While attenuating the high frequency noise is important to prevent aliasing from interfering with the current signal, the FIR filter does not have to provide all the attenuation. This can allow fewer coefficients to be used, which can significantly decrease the time delay of the system. Additionally, the linear relation between gain and frequency requires that the system be carefully tuned to the expected sampling conditions.

The effect of these design considerations will be discussed and shown in relation to the PCB design discussed in this chapter. The circuit diagram is found in Figure 5.1.

5.1 Switched Integrator Test Circuit Design

The board is powered by ±15V, and is protected with a system of zener diodes and capacitors to prevent damaging the board if power is improperly connected. Two current sources were included in order to allow for the possibility of more complex inputs for future testing. Each signal input is protected by TVS diodes.[1] The J13 and J19 jumpers allow for selecting the input signal polarity. For these tests, the J13 jumper was configured to invert the polarity of the input signal to negate the inversion of the current amplifier. The signals are then conditioned by an AMP03G differential amplifier. The current source uses a 10 M resistor followed by a 1000x current divider. An input of 1 V provides 100 pA to the integrator. The integrator output is buffered and conditioned by an OP27G line driver. These OP27G amplifiers have a 10 kΩ resistor and 10 nF capacitor that contribute a pole at 1.6 kHz that contribute to the measured magnitude and phase of the integrator component.

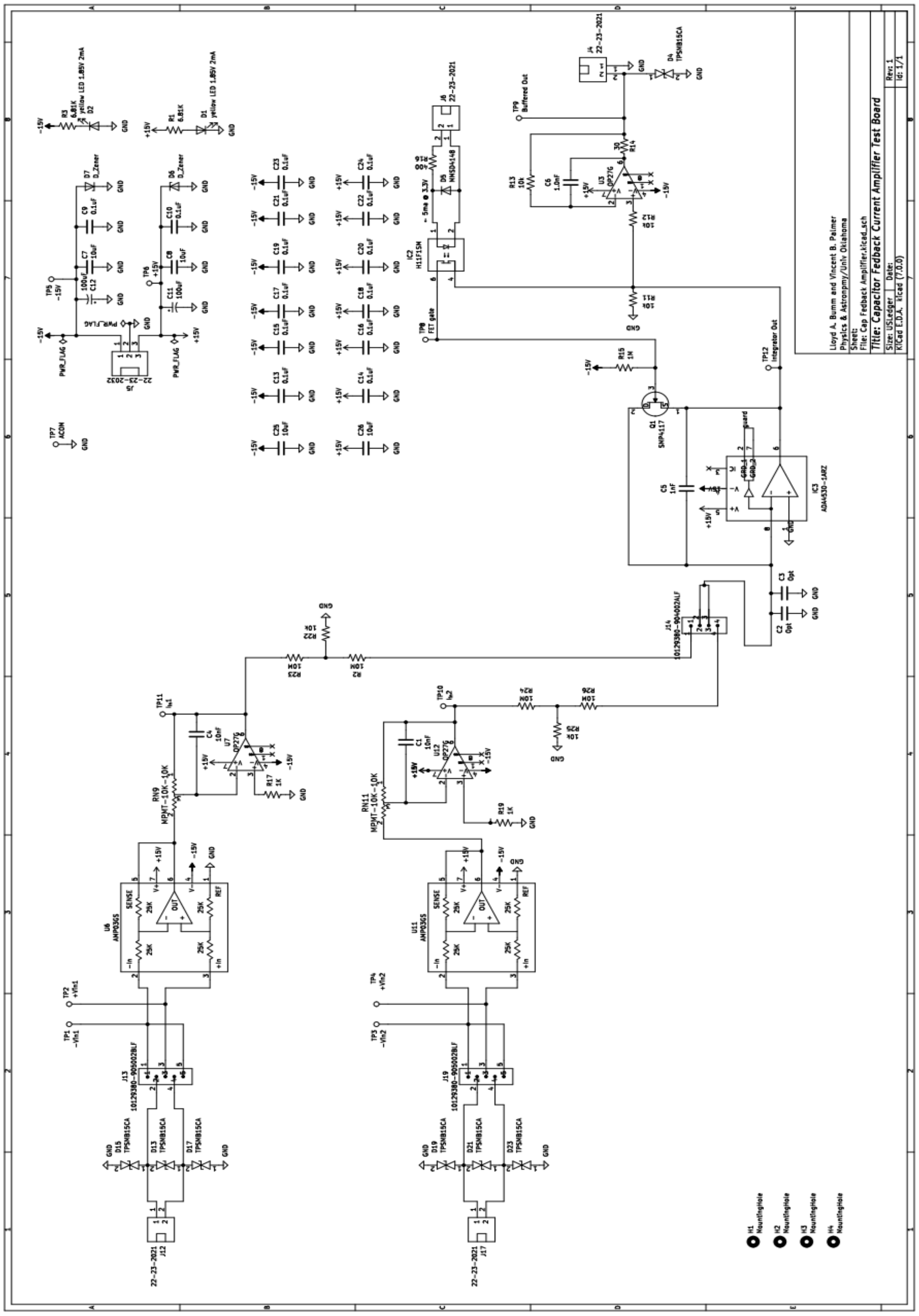

Lloyd A. Bumm and Vincent B. Palmer
Physics & Astronomy/Univ Oklahoma
Sheet:
File: Cap Fedback Amplifier.kicad_sch
Title: Capacitor Fedback Current Amplifier Test Board
Size: USLedger
Date:
KiCad E.D.A. kicad (7.0.0)
Rev: 1
Id: 1/1
ADA4530-1ARZ
SMP4117
H11F1SM
OP27G
AMP03GS
Buffered Out
Integrator Out
FET gate

Figure 5.1: Circuit diagram of PCB layout. Note this board was designed for the ADA4530 but was modified to accept the ADA4622 due to survivability issues experienced during testing.

The PCB was designed to use the ADA4530-1ARZ femtoampere input bias current electrometer amplifier as the integrator,[2] because of its very small input current offset and voltage noise.However, during initial testing we were not able to get the ADA4530 to work. Instead, the ADA4622,[3] a precision JFET transistor op amp with a high bandwidth (−3dB = 15.5 MHz) and slew rate (~20 V/μs) was substituted instead The feedback capacitor C5 was initially planned to be a 1 nF capacitor, but during testing it was found that the gain was too small, and and after testing that will be discussed in the next sections, was replaced with a 5 pF capacitor. This means that the expected gain of the amplifier is

$$A_I = \frac{1}{5 \times 10^{-12}\ \mathrm{F}} = 200\ \times 10^9 \mathrm{F}^{-1}.$$

The SMP4117[4] acted as the switch, and was opened and closed by the H11F1SM[5], which receives the +3.3 V signal from the FPGA prior to the ADC reaching its limits and closes the switch of the SMP4117, allowing the capcitor to discharge. The PCB was designed with a provision for loading the input with capacitance for testing, C2 and C3. For these tests C1 and C2 were populated by MMBD301-TP schottky diodes to protect the inverting input from exceeding ±0.35 V. The output of the switched integrator is buffered by an OP27G and output to the anti-aliasing filter.

5.2 System Performance Analysis

All tests were performed with a Stanford Research Systems DS345 30 MHz synthesized function generator.[6] The signals out of the function generator, integrator, and differentiator were analyzed with a Keysight DSOX1204G oscilloscope.[7] Also note that a small DC offset in the range of −0.12 V was applied to the function generator output waveforms to compensate for an apparent input offset current in the integrator. This offset was manually adjusted for each measurement to cancel the DC drift of the integrator output.

5.2.1 Demonstration of System Performance

Figure 5.2 demonstrates the ability of this system to effectively process various signal waveforms. Triangle and square waves at 100 Hz were input to the system with max amplitude of ±2 V for the function generator. The output from the integrator were parabolas and triangle waves respectively, matching the expected shape and sign of the integral of triangle and square waves.

Next, the relative accuracy of the gain of both the integrator and differentiator can be checked by comparing the rough magnitudes of the signals resulting from the input square wave. For the 4 V peak-to-peak square wave, the integrator output slope has a magnitude of 40 V/s, and the differentiator output has a magnitude of 80 mV peak-to-peak. The resulting gains are 2×10^{11} F^{-1} for the integrator, 1×10^{-3} s for the differentiator, in line with the expected values and yields

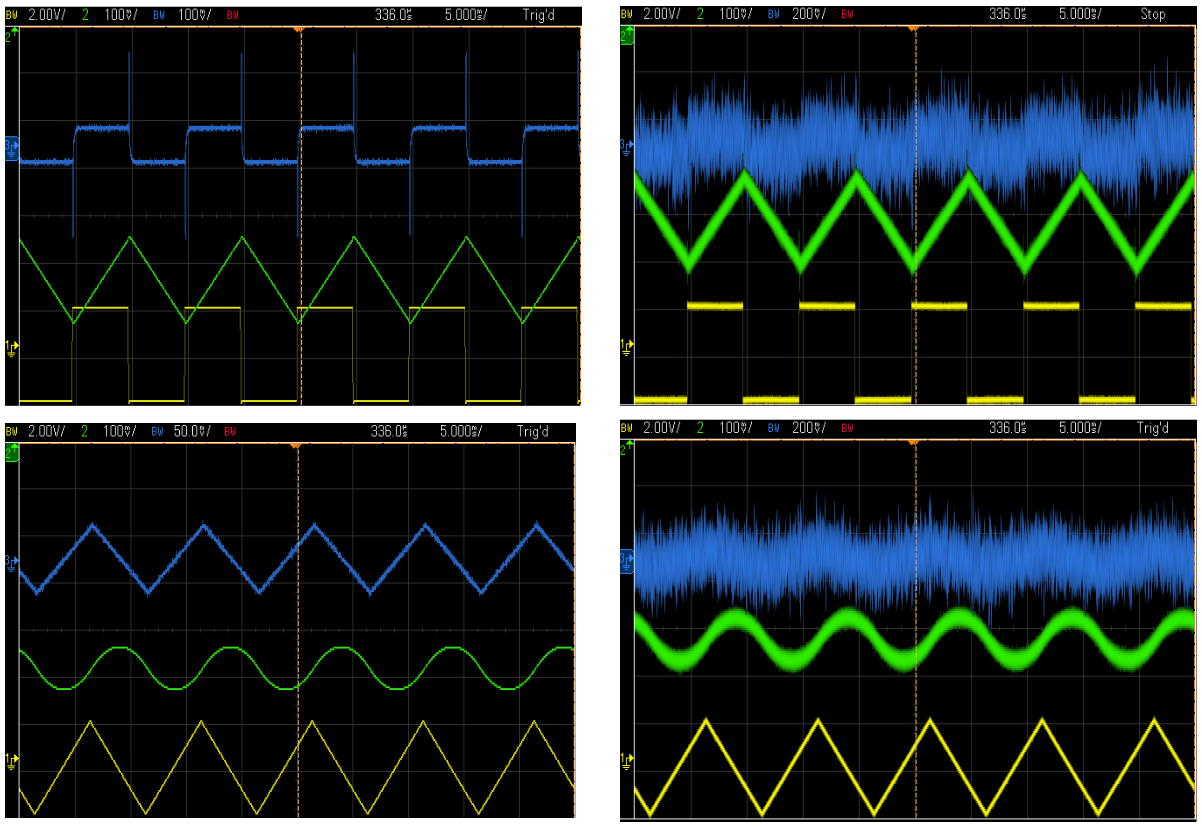


Figure 5.2: Demonstration of integration and differentiation of input signal. Yellow is the input signal, green is the output of the integrator, and blue is the output of the differentiator. Bottom two images show a 100 Hz triangle wave with ~2V amplitude, top two images show a 100 Hz square wave with same amplitude. Images on the left are after averaging over 65536 samples. Images on the right are without averaging.

an overall transimpedance gain of 2×10^8 V/A. The frequency response of the above component gains is checked in the next section.

When the output of the differentiator is averaged over a large number of samples, the original input waveform is recovered. Without averaging however there appears to be a significant amount of noise on both the differentiator and the integrator. Additionally, averaging the signal over so many samples is not ideal for STM application, as the STM requires real-time feedback. Comparing the averaged and raw data out of the integrator shows a slight fuzziness which indicates noise on the integrator. Since differentiators are highly susceptible to noise, it is possible that the noise on the integrator is being amplified by the gain of the differentiator and will need to be mitigated to produce cleaner signals.

5.2.2 Magnitude vs Frequency Response

Next, the gain of both the integrator and the differentiator were measured individually across a wider range of frequencies and compared to the theory. For these tests, the gain was computed in the time domain with an input sine wave represented by the formula

$$V_{in} = V_I \sin 2\pi f t.$$

Given that the 1000x current divider outputs 100 pA of current for every 1 V of input signal, and defining the resulting transadmittance gain as

$$A_g = 1 \times 10^{-10} \left(\frac{A}{V}\right),$$

the equation for the current coming into the amplifier is

$$I_{in} = V_I A_g \sin 2\pi f t.$$

The voltage output of the integrator is another sinusoid, and since it is related to the integral of the input current, it takes the form,

$$V_{out} = V_O \cos 2\pi f t,$$

Plugging these equations into the equation shown in Section 2.2.2 for the gain on an integrator in the time domain yields

$$\frac{\left(\frac{dV_{out}}{dt}\right)}{I_{in}} = \frac{2\pi f V_O}{V_I A_g} = A_I.$$

The voltage out of the integrator was measured at every half decade of frequency from 1 Hz to 3000 Hz and is plotted in Figure 5.3 to determine the gain. Below 3000 Hz, the gain matches very closely with the expected value of 2×10^{11} F^{-1} but begins to deviate above 3000 Hz due to the roll off of the gain of due to U12 and U7 that begins around 1.6 kHz. This shows that the integrator is behaving as expected.

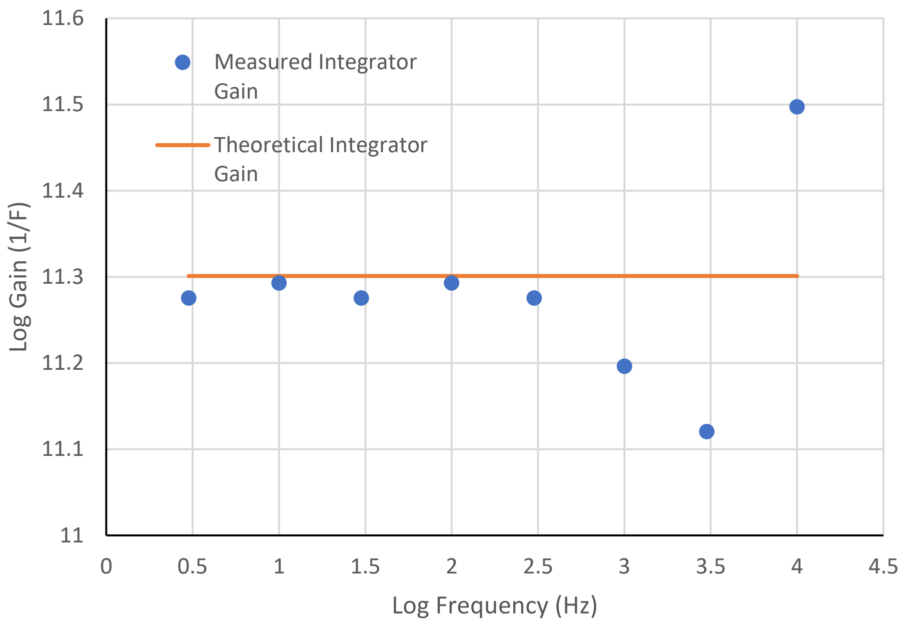


Figure 5.3: Measured gain of the integrator. Below 3000 Hz, the amplifier gain was within 10% of the theoretical value based off the feedback capacitance and can be seen to decrease as frequency increases as expected. Above 3000 Hz, the poles from U7 and U12 cause the gain to deviate.

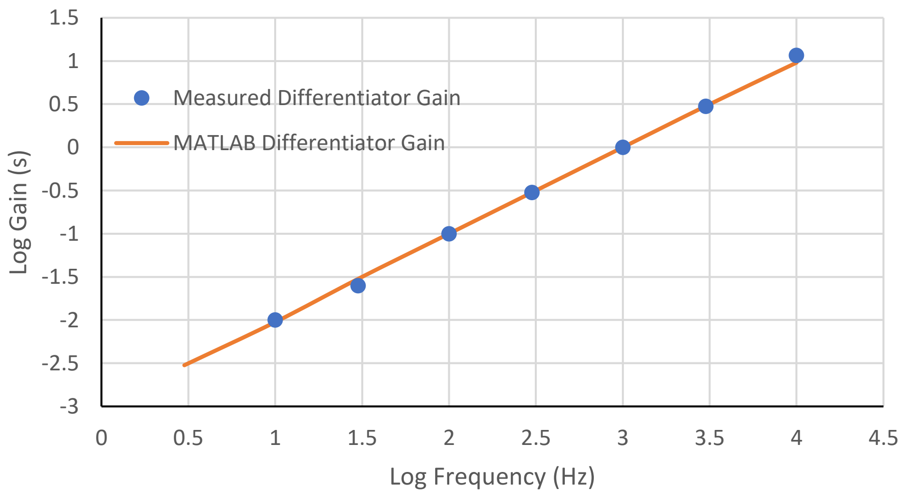


Figure 5.4: Measured gain of the differentiator. The slope of the linear differential gain was measured to follow the same relationship as predicted from the MATLAB filter designer tool.

The gains of the differentiator were measured via the same means and compared to the predicted values generated by the MATLAB Filter Designer tool. The measured gain vs frequency relationship for the differentiator is found in Figure 5.4. The values were found to nearly match the MATLAB values, indicating that the differentiator is behaving as expected and that the Filter Designer tool can be used to produce filters with desired gains in specific frequency bands with this implementation. The discrepancy at low frequencies is most likely due to precision limitations of the oscilloscope.

5.2.3 Phase Response and Time Delay

Next, the phase of both the integrator and the differentiator were measured compared and compared to the theory. In the frequency domain, the gain of a non-inverting integrator is

$$\frac{V_{out}}{V_{in}} = \frac{1}{j\omega RC}.$$

This shows that the expected phase response of this integrator is −90º. To measure the phase response of the integrator, the time delay between an input sine wave and the output of the integrator were compared to the period of the input sine wave. As shown in Figure 5.5, the phase

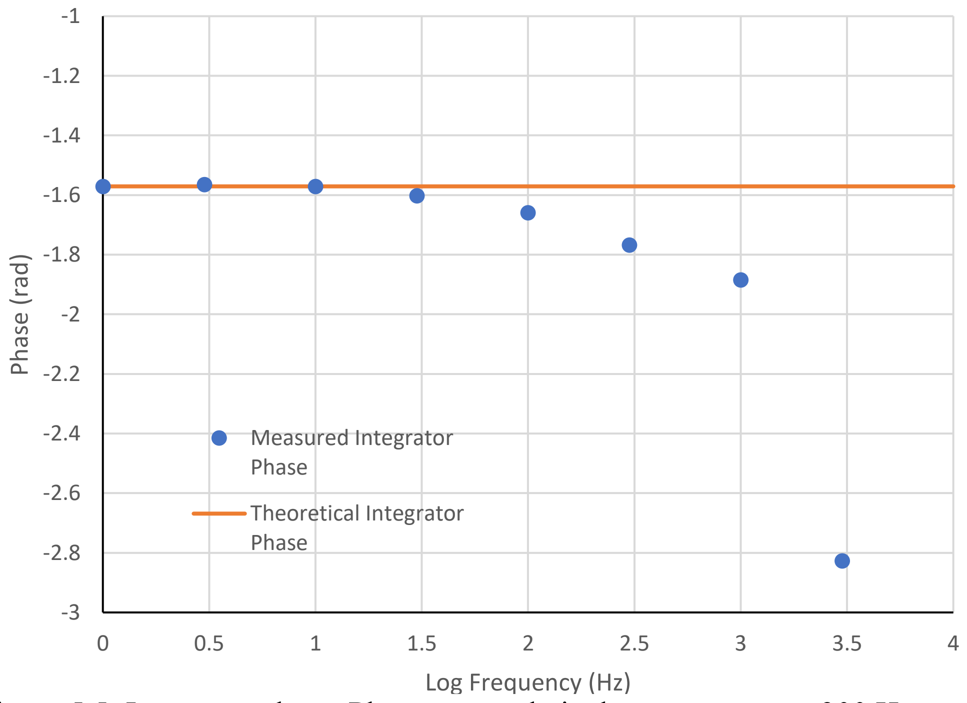
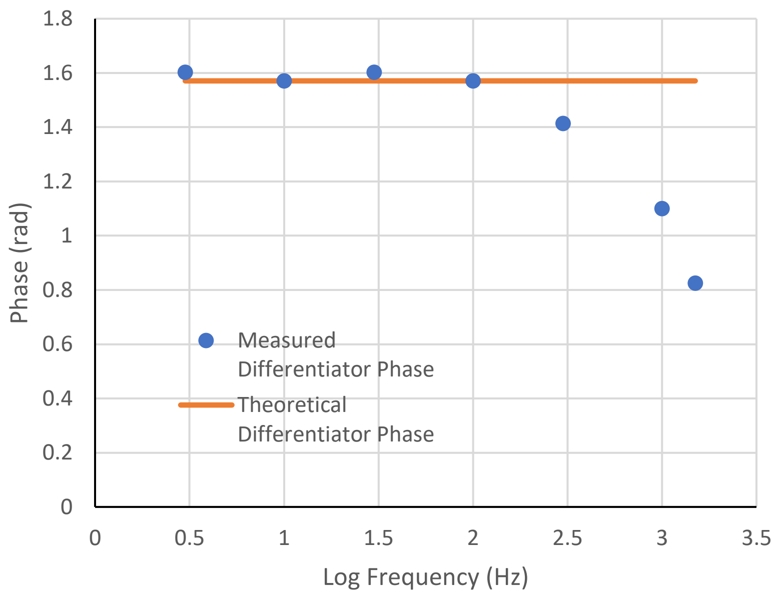


Figure 5.5: Integrator phase. Phase stays relatively constant up to 300 Hz at –90º as expected. Beyond that, the effect of the additional poles contributed by U12 and U7 affect the shape of the graph.

is relatively constant up to 300 Hz at roughly −90º. The effect of the pole at 1.6 kHz on U7 and U12 can be seen above 1 kHz.

Figure 5.6 shows the differentiator phase as a function of frequency compared to the standard value for a differentiator. As expected, the values start near +90º, but decrease as frequency increases, which can be explained by the group delay of the FIR filter compared to the input frequency. The group delay of the differentiator can be extracted from the slope of linear plot, yielding a group delay of 83 μs.



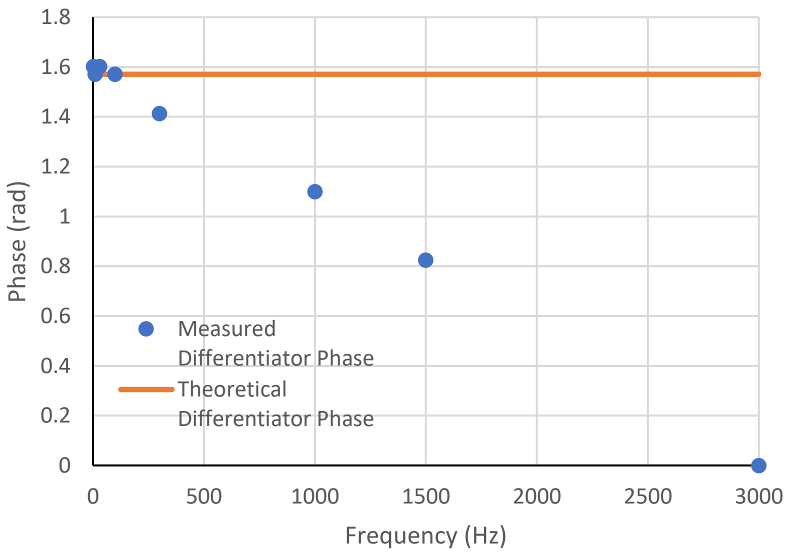


Figure 5.6: Differentiator phase. (Top) Phase vs log frequency. (Bottom) Phase vs linear frequency. Phase stays relatively constant up to 300 Hz at 90º as expected. Beyond that, the phase begins to decrease because of the time delay of the FIR filter.

Figure 5.7 shows the relationship between the input, integrator, and differentiator waveforms with an increase in frequency. At low frequencies, the differentiator appears to be in phase with the input waveform, with the integrator 90º out of phase. This is because of the opposing phase response of the differentiator and integrator, which cancel each other out. As frequency increases however, the group delay causes the differentiator to not fully correct back in phase with the input signal. This is due to the FIR filter and the anti-aliasing filter contributing time delays in the system that are not noticeable at low frequencies but become more apparent at high frequencies.

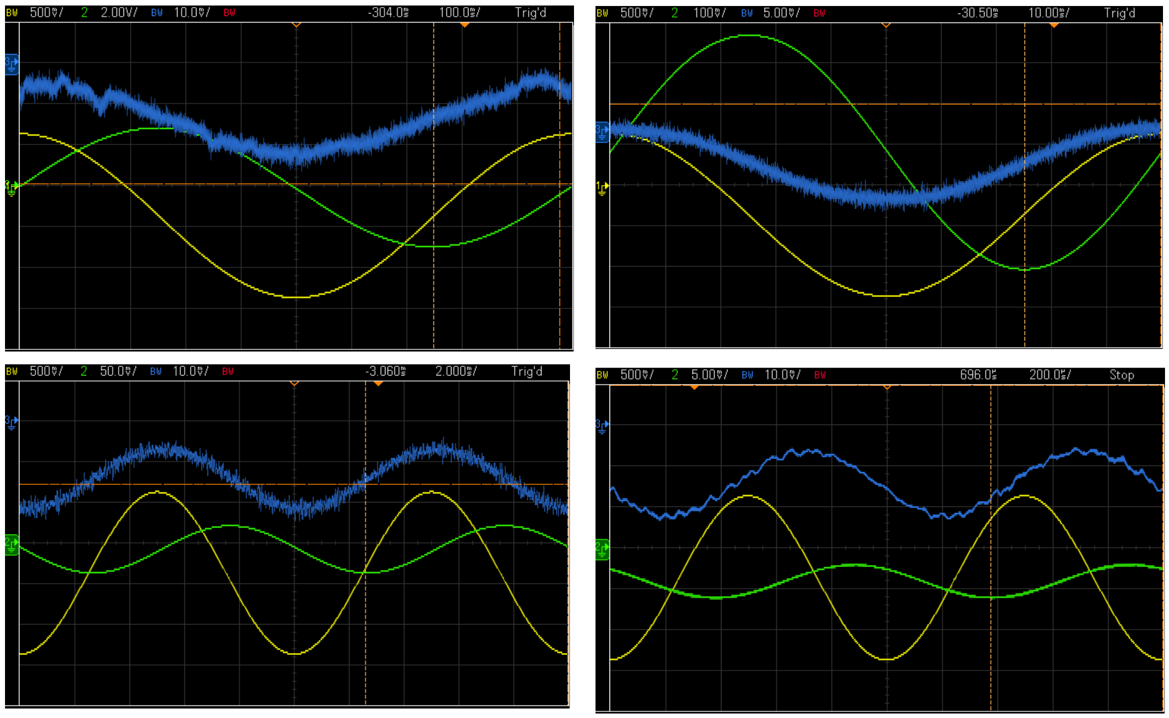


Figure 5.7: Phase of entire system. At low frequencies, the phase of the integrator and differentiator cancel out, leaving the differentiated signal in phase with the input. As frequency increases, the time delay of the FIR filter causes the differentiator signal to fall out of phase with the input. (Top left) 1 Hz. (Top right) 10 Hz. (Bottom left) 100 Hz. (Bottom right) 1000 Hz.

To measure and verify the total system group delay, a sawtooth wave was used as the input because the discontinuity is easy to correlate to the output of the differentiator. Measuring the time delay shown in Figure 5.8 gives a total time delay from input of 80 µs. This is very close to the measured time delay from Figure 5.6 of 83 µs, again showing the system is working as expected.

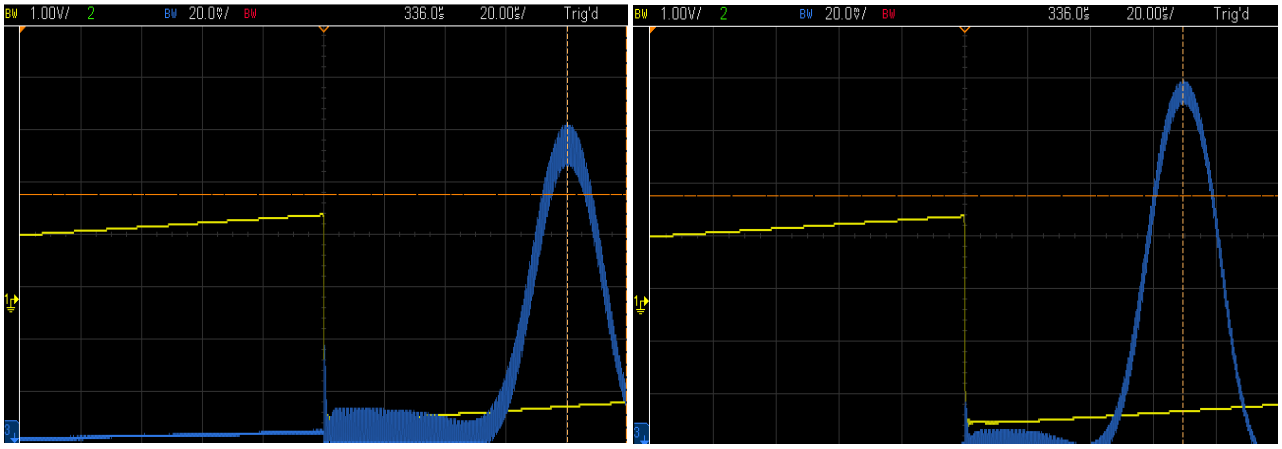


Figure 5.8: Determination of group delay. With the Sallen-Key filter (left) and without (right). The time delay is increased by 10 µs compared to the rest of the system, in line with Figure 3.4. It also matches closely with the measured group delay of 83 µs from Figure 5.6.

5.2.4 Low-Frequency, Low-Amplitude Analysis

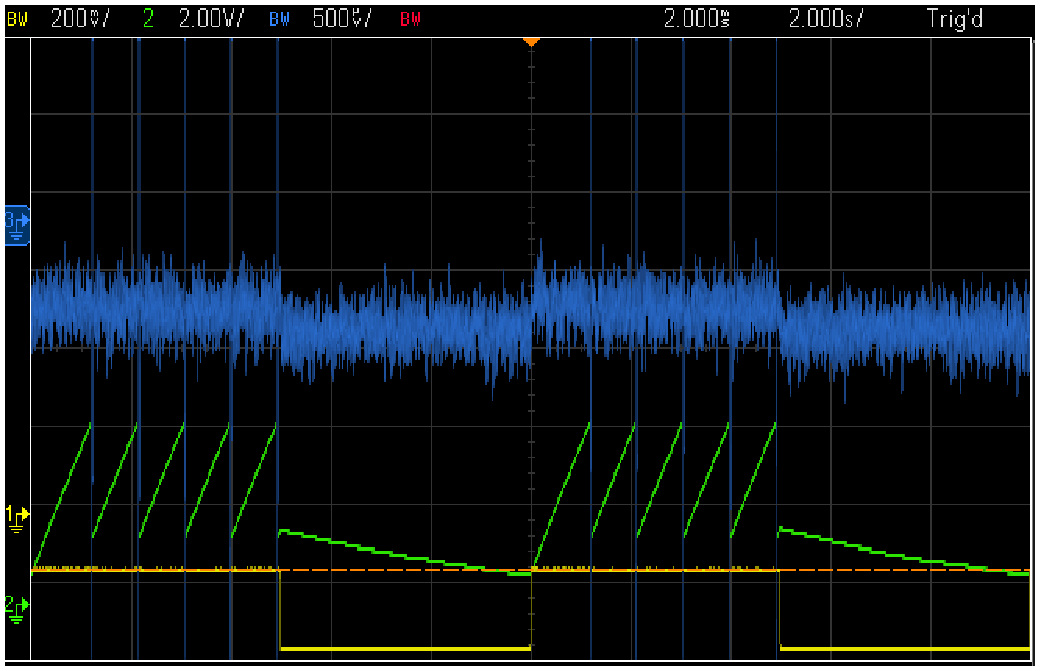


Figure 5.9: Performance of system at low amplitudes and frequencies. Input was a 0.1 V square wave at 0.1 Hz. Signals were processed with boxcar averaging.

As the tunneling currents expected in these STM operations are very small, the expected performance of the system is better tested at low-frequency and low amplitude. Figure 5.9 shows the performance of the filter under a 0.1 V, 0.1 Hz square wave. At this frequency, the resets are apparent in the capacitor, occurring several times per cycle. The difference in slopes highlight the drift inherent in the system, and shows that even with the offset discussed earlier applied, the drift could not be fully cancelled out. While the signal on the integrator is very sharp, there is still a high amount of noise on the differentiator. This was unexpected, as reducing the amount of noise out of the integrator should reduce the noise on the differentiator. A possible explanation for this was the effect of small slopes combined with the resolution of the ADC. If the input from the integrator results in a high slope, then there will be significant changes in the voltage between samples. However, if the voltage does not increase beyond the resolution of the ADC between samples, then the ADC will output the same voltage in between samples, resulting in the slope being transformed into a straight line. This could cause the ADC output to bounce between zero and very small numbers due to random fluctuations, which when compared to the small voltages it is trying to output would result in an increase in the signal-to-noise ratio. Figure 5.10 demonstrates this graphically and shows that the combination of the sampling time and the input voltage will determine if the ADC is able to resolve input voltages reliably.

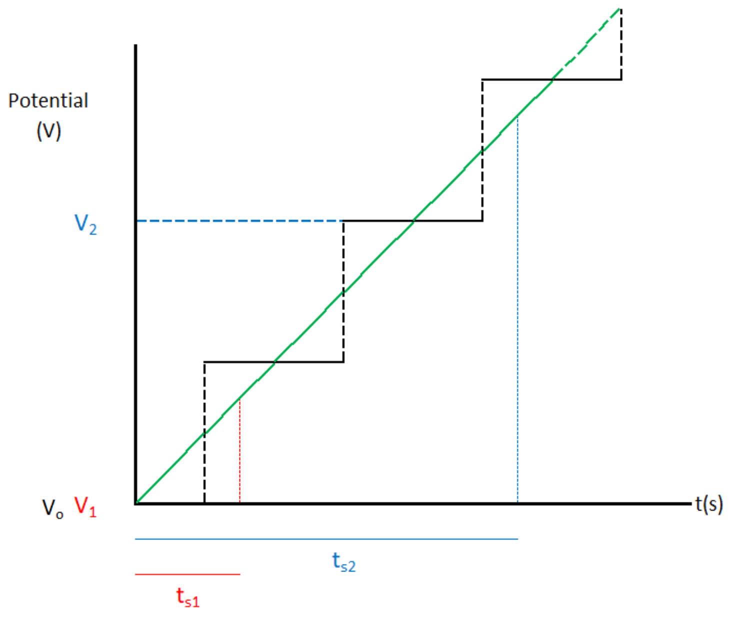


Figure 5.10: Relationship between sampling time and ADC resolution. The green line represents the actual voltage being input to the system. The black line represents the digitized voltage the ADC will interpret in a certain range. V_0 is the voltage sampled prior to the current sampling. The slope of the voltage being sampled does not change between the two sampling times, but long sampling times (t_{s2}), allow for the voltage to change enough that the second voltage (V_2) is substantially different from V_0, and thus a slope can be measured. Short sampling times (t_{s1}) can result in the next sampling resulting in the ADC outputting a voltage (V_1) with the same value as V_0. Thus, the FIR filter would interpret this as a straight line.

As the slope could not be changed substantially to test this, the sampling time was instead increased by decreasing the clock frequency to 400 kHz while keeping the rest of the code and system unchanged. This increased the sampling time from 1.5 μs to 75 μs. This noticeably decreased the noise on the differentiator, so much so that averaging was not required to recover the initial signal reliably and accurately, as seen in Figure 5.11. Thus, the initial assumption that increasing the sampling frequency of the FIR filter would reduce noise is not always accurate and increasing it too much may be counterproductive when trying to measure sub-picoamp tunneling currents. This shows that for a specific tunneling current, the system gain sets a limit on the sampling time of the filter, and how that sampling time is allocated between the different relevant parameters needs to be considered when designing a similar system.

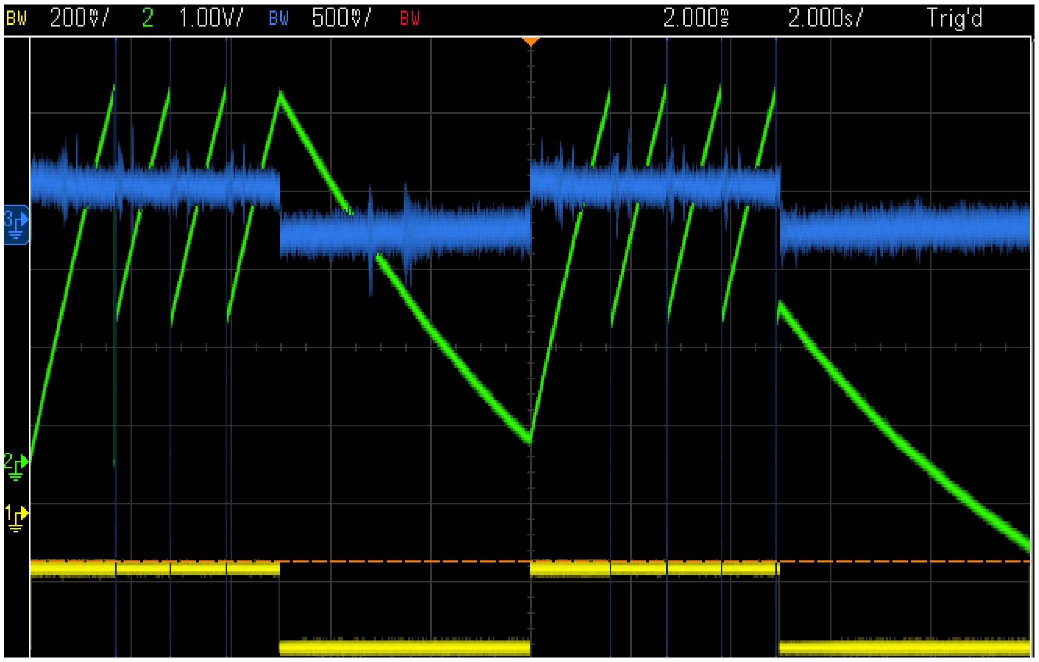


Figure 5.11: Performance of system at 13.3 k samples per second. Input was a 0.1 V square wave at 0.1 Hz. Signals were processed without any averaging. The increased gain of the filter at low frequencies resulted in the smoothest signal processed yet with the highest signal-to-noise ratio.

Decreasing the clock frequency also had the effect of increasing the differentiator gain, which can be seen due to the increase in the amplitudes of the differentiator signal between the 20 MHz clock (±250 μV) to the 400 kHz clock (±500 mV). This is expected, as decreasing the clock frequency increases the time between samplings, which is analogous to the original filter sampling a signal with a higher slope.

5.3 References

1. Littlefuse, TPSMB Series TVS Diode Datasheet, 2020 https://www.littelfuse.com/~/media/electronics/datasheets/tvs_diodes/littelfuse_tvs_diode_tpsmb_datasheet.pdf.pdf
2. Analog Devices, ADA 4530-1 Datasheet Rev B., Norwood MA, 2017. https://www.mouser.com/datasheet/2/609/ADA4530_1-3120025.pdf
3. Analog Devices, ADA 4622-1/ ADA 4622-2/ADA 4622-4 Datasheet Rev F., Norwood MA, 2022. https://www.analog.com/media/en/technical-documentation/data-sheets/ada4622-1-4622-2-4622-4.pdf
4. InterFET, 2N4117/A, 2N4118/A, 2N4119/A N-Channel JFET, 2020. https://www.mouser.com/datasheet/2/676/jfet_2n4117_2n4118_2n4119a_interfet-2888010.pdf
5. ON Semiconductor, Photo FET Optocouplers H11F1M, H11F2M, H11F3M, 2007. https://www.mouser.com/datasheet/2/308/1/H11F3M_D-2314223.pdf
6. Stanford Research Systems, Synthesize Function Generator Rev 2.3, 2016. https://www.thinksrs.com/downloads/pdfs/manuals/DS345m.pdf
7. Keysight InfiniVision 1000 X-Series Oscilloscopes Datasheet, 2020 https://www.keysight.com/us/en/assets/7018-06411/data-sheets/5992-3484.pdf

Chapter 6: Conclusion

6.1 Summary of Results

While the desired SNR from Chapter 1 has not yet been demonstrated, there were two major successes resulting from this project.

First, a working prototype was created which has been demonstrated to function across a range of frequencies and operating conditions, allowing for future testing in pursuit of the desired SNR at sub-picoamp tunneling currents. The gain and phase of the integrator and differentiator performed as expected, and the system can be easily modified to accommodate a variety of different components. The analysis of Chapter 5 can then be repeated to test the effect the changes have on the performance of the system.

Second, the results of Chapter 5 revealed the important design criteria for this system, and careful analysis of the dependance of each of these individual components leads to a design roadmap for future iterations. Figure 6.1 captures the design progression, starting with a desired tunneling current and bandwidth. The tunneling current dictates the required gain necessary to amplify the signal into a detectable region, which determines the necessary feedback capacitance, which determines the reset characteristics of the system. The bandwidth sets the sampling time and provides limitations which guide the filter frequency characteristics, which allows for the generation of a filter with specific performances. The integrator gain and the sampling time must result in a signal amplitude that can be effectively resolved by the ADC and DAC. Once these components have been chosen, the noise of the system can be analyzed and checked to see if the resulting SNR is high enough.

The system was able to recover an input signal on the order of 1 pA with significant averaging, but noise appears to be a significant concern with this current configuration. Identifying the noise sources and mitigating the most significant contributors, particularly out of the integrator, will be crucial for minimizing the noise of the system as whole. With some targeted analysis and optimization, this system may be able to achieve the current and SNR goals set out in Chapter 2. The next section identifies some of the issues, constraints, and improvements that were identified during this project.

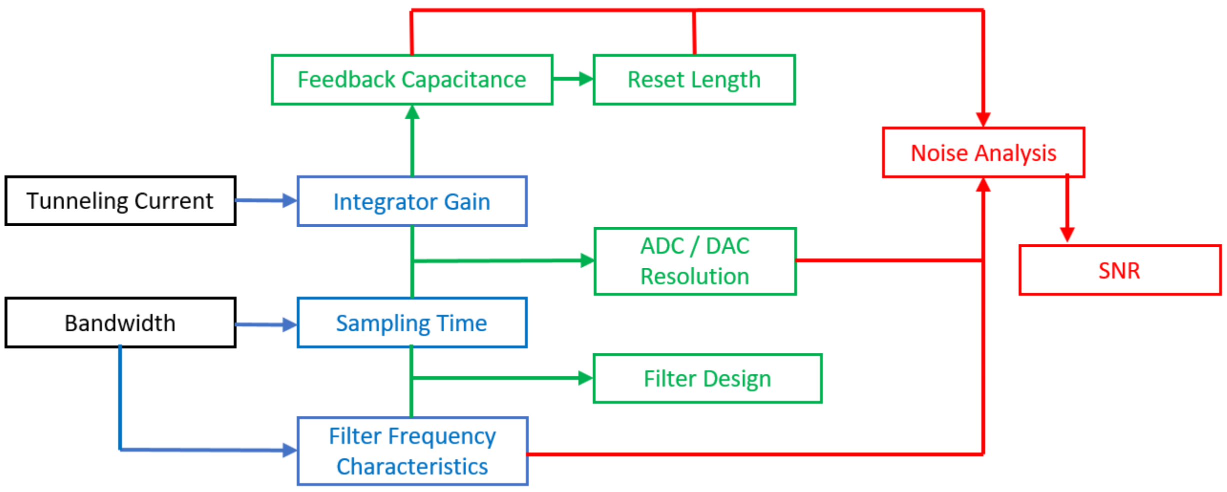


Figure 6.1: Design sequence for future iterations. Items in black are the initial design constraints. Items in blue are the intermediate results driven by the chosen constraints. Items in green are the necessary design outputs resulting from the intermediate results. Items in red are the final analyses that must be performed to determine if the chosen design will accomplish the desired end goal.

6.2 Limitations and Possible Improvements

6.2.1 VHDL-2008 Constraints

VHDL-2008 only supports integer math, but the filter coefficients had many decimal digits. This required that the coefficients be significantly truncated. This was accomplished by multiplying the coefficients by a large enough power of 10 to achieve at most 6 integer digits per coefficient, and then dividing the sum by the same power of 10. The reason for the limitation on 6 digits was that very large numbers, especially if added together, can cause integer overflow, and resulted in unusual operations and performance. To generate optimal filters, the spread in the total number of digits in the integer coefficients to a minimum, with a good goal being the same order as the number of digits used by the ADC, so that the largest coefficients don't overflow the system, but the smallest coefficients are not truncated to zero and the order of the filter is reduced.

To allow for filters that do not meet the above criteria if necessary, and to improve upon some other limitations of this design, future iterations should take advantage of the Vivado IP catalog. The IP catalog has both a floating-point arithmetic and FIR Compiler module which could be useful to simplify/optimize the above operations. This was not implemented in this design because of issues interfacing the Vivado IP integrator with the VHDL-2008 code.

6.2.2 Clocking Constraints and Resolution Limitations

The FPGA, ADC, and DAC were all chosen based on their availability as a proof of concept. Increasing the speed of the clock would reduce the time delay or allow more precise data (more bits) to be output in the same amount of time, assuming the ADC resolution can support the decrease in sampling time. Additionally, ADC and DACs with higher resolution would allow for more precise tuning of the small tunnelling currents. Additional analysis showed that the integer math limitation discussed in Section 6.2.1 does not have a significant effect on accuracy of the ADC and DAC, as the rounding inherent in the conversion resolution ends up washing out the effect of the lost decimal digits. However, if the truncation process has the effect of reducing the order of the filter significantly because several coefficients were too small in comparison to the largest coefficients, the information loss of the filter can reduce the systems accuracy and result in additional noise. With the AD4001, any current resulting in a voltage with a change in magnitude of less than 152.6 μV between samples would contribute to the signal degradation discussed in Section 5.2.4. Increasing the ADC resolution would allow for an increase in the bandwidth. Additionally, the IP catalog has a clocking wizard which can generate clocks with larger range of possible frequencies compared to the process clockDivider, which could be used to generate clocks that can maximize the throughput based on the components used.

6.2.3 Reset Considerations

While a larger capacitor would simplify operations, it is necessary to use as small a capacitor as possible to have sufficient gain to sample small tunnelling currents. Excessive resetting can be prevented in this configuration if the tunneling currents are small enough and any input bias is accounted for. A simple RC time constant cannot be measured easily for this system to determine the optimal reset length, so additional analysis measuring the discharge rate should be performed to optimize reset length.

There are a few steps that can be taken to help reduce the number of resets needed:

1. Careful calibration and compensation of the input signal should help prevent input signal offset from saturating. This could potentially be done by providing low frequency feedback to keep the integrator centered. This could be checked with a carefully calibrated sine wave generator; if a small enough perfectly balanced sine wave is used as an input, then the ascending and descending branches should cancel out the charge on the capacitor plates and resetting won't occur. If the sine wave is offset, then excess charge will build up in the direction of the offset and the capacitor will saturate. One possible improvement as well would be using the planned ADA4530-1 amplifier instead of the ADA4622. The ADA4622 has an average input bias current of 2 pA and may be the source of the drift noticed. The ADA4530-1 is designed to have an ultra-low input current of on average 1 fA, an improvement of 2000x.
2. Careful calibration in the choice of feedback capacitor. As large a capacitor as possible should be used so that the rate of change of the voltage for a given tunnelling current is small, preventing the voltage from exceeding the ADC limits.
3. Multi-Stage Amplification: The first stage is the integrator, which would be outfitted with as large of a capacitor as possible while still providing sufficient gain. The second stage would then be a low noise, high gain amplifier to provide the additional amplification necessary. Typically, it is desirable for the first stage in cascading amplifiers to have the most gain, but it may be possible to reach a compromise with this configuration that will provide the necessary dynamic range and gain.
4. Using an ADC/DAC with as large of a range as possible. A larger ADC range would allow more charge to build up on the capacitor before necessitating a reset, but this may increase the amount of discharge time needed as well.

Additionally, a point of improvement for the VHDL-2008 code is that this code did not implement a planned delay between closing the reset switch and resuming ADC operations. Reset noise on the capacitor could lead to transient voltages that would degrade the signal after resetting; pausing for a short period to allow the transients to dissipate would prevent this.

6.2.4 Parallel Filter Design

The different FIR filters used in these tests demonstrated a need to optimize filters based on the expected conditions being sampled. Attempting to use a filter outside of the conditions for which it was optimized can cause the performance to suffer dramatically. One way to maximize performance over a wider range of conditions would be to program multiple parallel FIR filters into the FPGA all running at the same time. The output could be dynamically chosen to be the one corresponding to the slope of the signal being sampled, or crudely approximated based on the difference between the two most recent samples. If the frequency of the system is high, then the slope will be high, allowing the use of a filter optimized for that frequency range. Conversely, if the frequency is low, then a filter optimized for lower frequencies would be used.

6.2.5 Noise

Many of the signals tested showed significant noise, which will affect the SNR achieved by the system. While the ability of the oscilloscope to take an average of a signal was useful for demonstrating the operation of the system, averaging the sample in this way is not ideal for STM operations due to the need for real-time feedback. The next step would be to conduct detailed noise analysis to determine the limiting noise sources in this system and to determine what components can be changed to improve the SNR. Figure 5.9 hints that the limiting noise source may be somewhere on the integrator. Additionally, improving the design so that an ultra-low input bias current amplifier like the ADA5530-1 can be used may significantly reduce noise on the integrator, which will prevent that noise from being amplified by the differentiator.

Appendix

A.1 FIR Filter

Table A.1: List of coefficients in 81st order direct-form differentiating FIR filter for PCB integrator testing.

Number	Full Value	Expressed Coefficient	Number	Full Value	Expressed Coefficient
0	-0.0000795002155406707000	-8	41	-0.0615025433614420000000	-6150
1	-0.0002349294080794460000	-23	42	-0.1818597356483660000000	-18186
2	-0.0005587738607812480000	-56	43	-0.2944377863607940000000	-29444
3	-0.0011429443002243000000	-114	44	-0.3945866431230330000000	-39459
4	-0.0021079725373220300000	-211	45	-0.4784162517855410000000	-47842
5	-0.0035937748897407300000	-359	46	-0.5430172226578300000000	-54302
6	-0.0057485206108828400000	-575	47	-0.5866066106966640000000	-58661
7	-0.0087104837895356200000	-871	48	-0.6085906238714820000000	-60859
8	-0.0125833095123488000000	-1258	49	-0.6095415201029370000000	-60954
9	-0.0174049555914985000000	-1740	50	-0.5910949997812200000000	-59109
10	-0.0231129555688742000000	-2311	51	-0.5557790910148760000000	-55578
11	-0.0295082498281096000000	-2951	52	-0.5067925706185040000000	-50679
12	-0.0362229303887307000000	-3622	53	-0.4477523721734430000000	-44775
13	-0.0426953595989090000000	-4270	54	-0.3824325505195080000000	-38243
14	-0.0481596731712490000000	-4816	55	-0.3145148339799230000000	-31451
15	-0.0516529195442990000000	-5165	56	-0.2473699596262230000000	-24737
16	-0.0520454016089867000000	-5205	57	-0.1838831588651290000000	-18388
17	-0.0480955463442876000000	-4810	58	-0.1263337341038220000000	-12633
18	-0.0385300715548540000000	-3853	59	-0.0763316065762705000000	-7633
19	-0.0221462408052410000000	-2215	60	-0.0348102548847928000000	-3481
20	0.0020686869474078100000	207	61	-0.0020686869474078100000	-207
21	0.0348102548847928000000	3481	62	0.0221462408052410000000	2215
22	0.0763316065762705000000	7633	63	0.0385300715548540000000	3853
23	0.1263337341038220000000	12633	64	0.0480955463442876000000	4810
24	0.1838831588651290000000	18388	65	0.0520454016089867000000	5205
25	0.2473699596262230000000	24737	66	0.0516529195442990000000	5165
26	0.3145148339799230000000	31451	67	0.0481596731712490000000	4816
27	0.3824325505195080000000	38243	68	0.0426953595989090000000	4270
28	0.4477523721734430000000	44775	69	0.0362229303887307000000	3622
29	0.5067925706185040000000	50679	70	0.0295082498281096000000	2951
30	0.5557790910148760000000	55578	71	0.0231129555688742000000	2311
31	0.5910949997812200000000	59109	72	0.0174049555914985000000	1740
32	0.6095415201029370000000	60954	73	0.0125833095123488000000	1258
33	0.6085906238714820000000	60859	74	0.0087104837895356200000	871
34	0.5866066106966640000000	58661	75	0.0057485206108828400000	575
35	0.5430172226578300000000	54302	76	0.0035937748897407300000	359
36	0.4784162517855410000000	47842	77	0.0021079725373220300000	211
37	0.3945866431230330000000	39459	78	0.0011429443002243000000	114
38	0.2944377863607940000000	29444	79	0.0005587738607812480000	56
39	0.1818597356483660000000	18186	80	0.0002349294080794460000	23
40	0.0615025433614420000000	6150	81	0.0000795002155406707000	8

A.2 FPGA Code (VHDL-2008)

```
--------------------------------------------------------------------------------------------------------------------------------------------
---- Company: University of Oklahoma
---- Student: Vincent Palmer
-------
---- Create Date: 08/26/2023
---- Last Updated Date: 06/12/2023
---- Design Name: AD4001 ADDA
---- Module Name: 4001_Top
---- Project Name: SEM ADDA
---- Target Devices: AD4001 Eval Board, CMOD A7 35-T FPGA, AD5781 Eval Board
---- Tool Versions:
---- Description: This project takes input from the AD4001 ADC and sends it to the
---- CMOD FPGA,  which processes it via a FIR filter and sends the differentiated
---- data to AD 5781 DAC
---- Dependencies:
-------
---- Revision:
---- Revision 0.01 - File Created
---- Additional Comments:
-------
--------------------------------------------------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity ad4001_top is
    Port (
--          Ports for the AD4001
            adsdi : out STD_LOGIC := '1';
            adsdo : in STD_LOGIC;
            adsclk : inout STD_LOGIC := '0';
            adcnv : out STD_LOGIC := '1';

--          Ports for the AD5781.
            dasdin : out STD_LOGIC;
            dasdo : inout STD_LOGIC;
            dasclk : inout STD_LOGIC;
            dasync : out STD_LOGIC;
            daldac: out STD_LOGIC;
            daclr: out STD_LOGIC;
            dareset: out STD_LOGIC;

--          Port to establish clock
            sysclk : in std_logic;
 --         Ports to set LEDs appropriately, and outputting to the PMOD.
 --         Used in troubleshooting, not in actual operations
            led1: inout std_logic;
            led2: inout std_logic;
            led0_r: inout std_logic;
            led0_g: inout std_logic;
            led0_b: inout std_logic;
            ja4: out std_logic := '1';
            ja0: out std_logic := '1';

 --         Port to reset integrator when capped
            sireset: out std_logic := '0';

 --         Port for the UART transmisson
            uart_txd_in: inout std_logic);

 end ad4001_top;

 architecture RTL of ad4001_top is

     constant ad_data_length: integer := 16;--AD4001 uses 16 bit data transfer
     constant ad_init_length: integer := 16;
     constant da_data_length: integer := 24; --AD5781 uses 24 bit data transfer
     signal cnvcount : natural range 0 to 25 := 0; --Counter that tells when
                                                   --CNV will be low or high
     signal adcount: natural range 0 to 15 := 15; --Counter for number of bits
                                                  --in AD transmission
     signal adinteger: integer range -268435455  to 268435455 := 0;
            -- Integer corresponding to binary output from AD
     signal transmissioncount : integer range 0 to 1000 := 0;
     signal transmitusb : integer range 0 to 1 := 0;

 --  Signals for creating sclk. Used in process clk divider, which divides the
 --  clock on the FPGA by a hex number to give you a slower clock.
     signal ClkCnt_a:std_logic_vector(27 downto 0);
     signal divisor: std_logic_vector(3 downto 0);
     signal reset: std_logic_vector(27 downto 0);
     signal ck_div_a: std_logic;
     signal runclk : std_logic := '0';
     signal divclk: std_logic;

 --  Counters to ensure correct number of bits sent via UART
     signal screenoutsig : std_logic_vector(9 downto 0) := "1111111111";
     signal usbuart : std_logic_vector (5 downto 0);


 begin

 --  CMOD17-35T has an internal clock of 12MHz, modified to support 40 MHz clock
 --  40 MHz= sysclk. Generates a slower runclk that synchronizes SPI interface
```

```
clockDivider: process(sysclk)
        begin
            divisor  <=X"1";--Results in a 20 MHz clock
            reset <= X"000_0000";  --clock counter reset state value
            if sysclk'event and sysclk='1' then
                ClkCnt_a <= ClkCnt_a + '1';
                if ClkCnt_a=divisor then
                    ClkCnt_a <= reset;
                   ck_div_a <= not ck_div_a;
                end if;
            end if;
            runclk <= ck_div_a;
        end process;

controlADCFIR: process(runclk, adinteger, transmissioncount, transmitusb, sysclk)
    variable acount : integer range 0 to 30 := 0;--Synchronize AD and DA by
                                                --cycling through 30 runclk cycles
    variable aoutsig: std_logic; --'1' or '0' corresponding to value currently on
                                 -- ADSDO, corresponds to binary integer
    variable adintegert: integer range 0 to 65535 := 0; --Temporary hold while
                                                       --determining adinteger
    variable adintegerx: integer range 0 to 65535 := 0; --Temp hold to determine
                                                       --if AD converter is railed
    variable adintegersig1: std_logic := '0'; --Checks to see if incoming value is
                                             --greater than v +4.8 V
    variable adintegersig2: std_logic := '0'; --Checks to see if incoming value is
                                             --greater than  to -4.8 V
    variable firout: integer range -131071 to 131071 := 0; --Output of FIR Filter
    variable adintegers: integer range -65535 to 65535 := 0; -- Output of AD 4001
                                                            --corrected for sign
    variable sicount: integer range 0 to 40000 := 0; -- Counter that drives how long

                                                    -- sireset is held high

    variable lpfir: integer_vector(81 downto 0); --Outputs of AD4001 held in an
                                                --array to be multiplied by the
                                                --corresponding coefficients
    variable lpfirold: integer_vector(81 downto 0) ; --Temporary hold for outputs
                                                    --of AD4001 held in array to
                                                    --allow shifting of old data
                                                    --into next spot in array
    --Results of multiplying lpfir value by corresponding coefficient
    variable lpfirint0: integer range -2100000000   to 2100000000;
    variable lpfirint1: integer range -2100000000   to 2100000000;
    variable lpfirint2: integer range -2100000000   to 2100000000;
    variable lpfirint3: integer range -2100000000   to 2100000000;
    variable lpfirint4: integer range -2100000000   to 2100000000;
    variable lpfirint5: integer range -2100000000   to 2100000000;
    variable lpfirint6: integer range -2100000000   to 2100000000;
    variable lpfirint7: integer range -2100000000   to 2100000000;
    variable lpfirint8: integer range -2100000000   to 2100000000;
    variable lpfirint9: integer range -2100000000  to 2100000000;
    variable lpfirint10: integer range -2100000000  to 2100000000;
    variable lpfirint11: integer range -2100000000  to 2100000000;
    variable lpfirint12: integer range -2100000000  to 2100000000;
    variable lpfirint13: integer range -2100000000  to 2100000000;
    variable lpfirint14: integer range -2100000000  to 2100000000;
    variable lpfirint15: integer range -2100000000  to 2100000000;
    variable lpfirint16: integer range -2100000000  to 2100000000;
    variable lpfirint17: integer range -2100000000  to 2100000000;
    variable lpfirint18: integer range -2100000000  to 2100000000;
    variable lpfirint19: integer range -2100000000  to 2100000000;
    variable lpfirint20: integer range -2100000000  to 2100000000;
    variable lpfirint21: integer range -2100000000  to 2100000000;
    variable lpfirint22: integer range -2100000000  to 2100000000;
    variable lpfirint23: integer range -2100000000  to 2100000000;
    variable lpfirint24: integer range -2100000000  to 2100000000;
    variable lpfirint25: integer range -2100000000  to 2100000000;
    variable lpfirint26: integer range -2100000000  to 2100000000;
    variable lpfirint27: integer range -2100000000  to 2100000000;
    variable lpfirint28: integer range -2100000000  to 2100000000;
    variable lpfirint29: integer range -2100000000  to 2100000000;
    variable lpfirint30: integer range -2100000000  to 2100000000;
    variable lpfirint31: integer range -2100000000  to 2100000000;
    variable lpfirint32: integer range -2100000000  to 2100000000;
    variable lpfirint33: integer range -2100000000  to 2100000000;
    variable lpfirint34: integer range -2100000000  to 2100000000;
    variable lpfirint35: integer range -2100000000  to 2100000000;
    variable lpfirint36: integer range -2100000000  to 2100000000;
    variable lpfirint37: integer range -2100000000  to 2100000000;
    variable lpfirint38: integer range -2100000000  to 2100000000;
    variable lpfirint39: integer range -2100000000  to 2100000000;
    variable lpfirint40: integer range -2100000000  to 2100000000;
    variable lpfirint41: integer range -2100000000  to 2100000000;
    variable lpfirint42: integer range -2100000000  to 2100000000;
    variable lpfirint43: integer range -2100000000  to 2100000000;
    variable lpfirint44: integer range -2100000000  to 2100000000;
    variable lpfirint45: integer range -2100000000  to 2100000000;
    variable lpfirint46: integer range -2100000000  to 2100000000;
    variable lpfirint47: integer range -2100000000  to 2100000000;
    variable lpfirint48: integer range -2100000000  to 2100000000;
    variable lpfirint49: integer range -2100000000  to 2100000000;
    variable lpfirint50: integer range -2100000000  to 2100000000;
    variable lpfirint51: integer range -2100000000  to 2100000000;
    variable lpfirint52: integer range -2100000000  to 2100000000;
    variable lpfirint53: integer range -2100000000  to 2100000000;
    variable lpfirint54: integer range -2100000000  to 2100000000;
    variable lpfirint55: integer range -2100000000  to 2100000000;
    variable lpfirint56: integer range -2100000000  to 2100000000;
    variable lpfirint57: integer range -2100000000  to 2100000000;
    variable lpfirint58: integer range -2100000000  to 2100000000;
    variable lpfirint59: integer range -2100000000  to 2100000000;
    variable lpfirint60: integer range -2100000000  to 2100000000;
    variable lpfirint61: integer range -2100000000  to 2100000000;
    variable lpfirint62: integer range -2100000000  to 2100000000;
    variable lpfirint63: integer range -2100000000  to 2100000000;
    variable lpfirint64: integer range -2100000000  to 2100000000;
    variable lpfirint65: integer range -2100000000  to 2100000000;
    variable lpfirint66: integer range -2100000000  to 2100000000;
    variable lpfirint67: integer range -2100000000  to 2100000000;
    variable lpfirint68: integer range -2100000000  to 2100000000;
    variable lpfirint69: integer range -2100000000  to 2100000000;
    variable lpfirint70: integer range -2100000000  to 2100000000;
    variable lpfirint71: integer range -2100000000  to 2100000000;
    variable lpfirint72: integer range -2100000000  to 2100000000;
    variable lpfirint73: integer range -2100000000  to 2100000000;
    variable lpfirint74: integer range -2100000000  to 2100000000;
    variable lpfirint75: integer range -2100000000  to 2100000000;
    variable lpfirint76: integer range -2100000000  to 2100000000;
    variable lpfirint77: integer range -2100000000  to 2100000000;
    variable lpfirint78: integer range -2100000000  to 2100000000;
    variable lpfirint79: integer range -2100000000  to 2100000000;
    variable lpfirint80: integer range -2100000000  to 2100000000;
    variable lpfirint81: integer range -2100000000  to 2100000000;

    variable lpfirouts: integer range -2100000000  to 2100000000; --Direct output
                      --of FIR filter (with sign) before correcting for twos
                      --complement
    variable lpcoeff0: integer := 0;--Coefficients in FIR Filter. Note, all are
    variable lpcoeff1: integer := 0;--multiplied by a common power of ten to
    variable lpcoeff2: integer := 0;-- corrent floating point numbers to integers.
    variable lpcoeff3: integer := 0;--This common term is divided out
    variable lpcoeff4: integer := 0;--later on in the filter
    variable lpcoeff5: integer := 0;
    variable lpcoeff6: integer := 0;
    variable lpcoeff7: integer := 1;
    variable lpcoeff8: integer := 1;
    variable lpcoeff9: integer := -1;
    variable lpcoeff10: integer := -4;
    variable lpcoeff11: integer := -6;
    variable lpcoeff12: integer := -3;
    variable lpcoeff13: integer := 11;
    variable lpcoeff14: integer := 31;
    variable lpcoeff15: integer := 39;
    variable lpcoeff16: integer := 6;
    variable lpcoeff17: integer := -75;
    variable lpcoeff18: integer := -162;
    variable lpcoeff19: integer := -158;
    variable lpcoeff20: integer := 30;
    variable lpcoeff21: integer := 367;
    variable lpcoeff22: integer := 625;
    variable lpcoeff23: integer := 458;
    variable lpcoeff24: integer := -307;
    variable lpcoeff25: integer := -1358;
    variable lpcoeff26: integer := -1866;
    variable lpcoeff27: integer := -963;
    variable lpcoeff28: integer := 1430;
    variable lpcoeff29: integer := 4033;
    variable lpcoeff30: integer := 4608;
    variable lpcoeff31: integer := 1451;
    variable lpcoeff32: integer := -4809;
    variable lpcoeff33: integer := -10544;
    variable lpcoeff34: integer := -10640;
    variable lpcoeff35: integer := -1776;
    variable lpcoeff36: integer := 14516;
    variable lpcoeff37: integer := 31362;
    variable lpcoeff38: integer := 39637;
    variable lpcoeff39: integer := 33184;
    variable lpcoeff40: integer := 12933;
    variable lpcoeff41: integer := -12933;
    variable lpcoeff42: integer := -33184;
    variable lpcoeff43: integer := -39637;
    variable lpcoeff44: integer := -31362;
    variable lpcoeff45: integer := -14516;
    variable lpcoeff46: integer := 1776;
    variable lpcoeff47: integer := 10640;
    variable lpcoeff48: integer := 10544;
    variable lpcoeff49: integer := 4809;
    variable lpcoeff50: integer := -1451;
    variable lpcoeff51: integer := -4608;
    variable lpcoeff52: integer := -4033;
    variable lpcoeff53: integer := -1430;
    variable lpcoeff54: integer := 963;
    variable lpcoeff55: integer := 1866;
    variable lpcoeff56: integer := 1358;
    variable lpcoeff57: integer := 307;
    variable lpcoeff58: integer := -458;
    variable lpcoeff59: integer := -625;
    variable lpcoeff60: integer := -367;
    variable lpcoeff61: integer := -30;
    variable lpcoeff62: integer := 158;
    variable lpcoeff63: integer := 162;
    variable lpcoeff64: integer := 75;
    variable lpcoeff65: integer := -6;
    variable lpcoeff66: integer := -39;
    variable lpcoeff67: integer := -31;
    variable lpcoeff68: integer := -11;
    variable lpcoeff69: integer := 3;
    variable lpcoeff70: integer := 6;
    variable lpcoeff71: integer := 4;
    variable lpcoeff72: integer := 1;
    variable lpcoeff73: integer := -1;
    variable lpcoeff74: integer := -1;
    variable lpcoeff75: integer := 0;
    variable lpcoeff76: integer := 0;
    variable lpcoeff77: integer := 0;
    variable lpcoeff78: integer := 0;
    variable lpcoeff79: integer := 0;
    variable lpcoeff80: integer := 0;
    variable lpcoeff81: integer := 0;

        begin
           adsclk <= runclk when ((acount >= 3) and (acount <= 18)
                     and (sicount = 0)) else '0'; --Synchronizes AD clock to
                                                  --runclk for 16 clock cycles
                                                  --when sireset is low
           adcnv <= '0' when ((acount >= 2) and (acount <= 20)) else '1';
               --Synchronizes ADCNV to runclk for 19 clock cycles when
               --sireset is low
           aoutsig := adsdo when ((acount >= 3) and (acount <= 18)) else '0';
               -- Pulls value on ADSDO into FPGA for calculations
           adintegersig1 := '1' when ((adintegerx > 31400) or sicount > 0) else '0';
               --Checks to see if AD input is above value correspondiong to +4.8,
               --rounded to nearest hundred
           adintegersig2 := '1' when ((adintegerx < 34100) or sicount > 0) else '0';
               --Checks to see if AD input is above value correspondiong to -4.8,
               --rounded to nearest hundred
           sireset <= '1' when (((adintegersig1 = '1')
                      and (adintegersig2 = '1'))) else '0';
               --If value is above +4.8V and -4.8V simultaneously-> initiate reset

           if (rising_edge(sysclk)) then --If sireset is triggered, start counting
                                         --to determine when to stop resetting
               sicount := sicount + 1 when sireset = '1';
           end if;
           if (falling_edge(sysclk)) then --Resets SI count so that normal
                                          --operations can resume
               sicount := 0 when sicount = 39999 else sicount;
           end if;

           if falling_edge(runclk) then --Increase acount to synchronize with ADSCLK

                                        --and AD SPI
               acount := acount + 1 when acount < 30 else 0;
           end if;
           if rising_edge(runclk) then --Converts binary into corresponding decimal
                                       --value, depending on clock count.
                                       --Two's complement, no sign
               if acount >= 3 and acount <= 18 then
                   adintegert := (adintegert + 2**(18-acount))
                                 when aoutsig = '1' else adintegert;
               end if;
           end if;
           if acount = 0 then--Reset adingetert to start fresh
               adintegert := 0;
           end if;
           if acount = 19 then--Store value in adintegert to check if
                              --SI needs to be reset
               adintegerx := adintegert when sireset = '0' else 0;
           end if;
       if acount = 20 then--Corrects twos complememnt to account for sign,
                          --easier to manipulate and apply to FIR filter
           adintegers := -1*(65536-adintegert)
                         when adintegert > 32767 else adintegert;
       end if;
           if acount = 21 then--Update FIR register with new value
               --If resetting, seeds FIR register with corrected values to
               --prevent delay in refilling register and to patch the
               --slopes together
             lpfir(0) :=  adintegers when sireset = '0' else lpfir(0) - lpfir(0);
             lpfirold(1) := lpfir(1) when sireset = '0' else lpfir(1) - lpfir(0);
             lpfirold(2) := lpfir(2) when sireset = '0' else lpfir(2) - lpfir(0);
             lpfirold(3) := lpfir(3) when sireset = '0' else lpfir(3) - lpfir(0);
             lpfirold(4) := lpfir(4) when sireset = '0' else lpfir(4) - lpfir(0);
             lpfirold(5) := lpfir(5) when sireset = '0' else lpfir(5) - lpfir(0);
             lpfirold(6) := lpfir(6) when sireset = '0' else lpfir(6) - lpfir(0);
             lpfirold(7) := lpfir(7) when sireset = '0' else lpfir(7) - lpfir(0);
             lpfirold(8) := lpfir(8) when sireset = '0' else lpfir(8) - lpfir(0);
             lpfirold(9) := lpfir(9) when sireset = '0' else lpfir(9) - lpfir(0);
             lpfirold(10) := lpfir(10) when sireset = '0' else lpfir(10) - lpfir(0);
             lpfirold(11) := lpfir(11) when sireset = '0' else lpfir(11) - lpfir(0);
             lpfirold(12) := lpfir(12) when sireset = '0' else lpfir(12) - lpfir(0);
             lpfirold(13) := lpfir(13) when sireset = '0' else lpfir(13) - lpfir(0);
             lpfirold(14) := lpfir(14) when sireset = '0' else lpfir(14) - lpfir(0);
             lpfirold(15) := lpfir(15) when sireset = '0' else lpfir(15) - lpfir(0);
             lpfirold(16) := lpfir(16) when sireset = '0' else lpfir(16) - lpfir(0);
             lpfirold(17) := lpfir(17) when sireset = '0' else lpfir(17) - lpfir(0);
             lpfirold(18) := lpfir(18) when sireset = '0' else lpfir(18) - lpfir(0);
             lpfirold(19) := lpfir(19) when sireset = '0' else lpfir(19) - lpfir(0);
             lpfirold(20) := lpfir(20) when sireset = '0' else lpfir(20) - lpfir(0);
             lpfirold(21) := lpfir(21) when sireset = '0' else lpfir(21) - lpfir(0);
             lpfirold(22) := lpfir(22) when sireset = '0' else lpfir(22) - lpfir(0);
             lpfirold(23) := lpfir(23) when sireset = '0' else lpfir(23) - lpfir(0);
             lpfirold(24) := lpfir(24) when sireset = '0' else lpfir(24) - lpfir(0);
             lpfirold(25) := lpfir(25) when sireset = '0' else lpfir(25) - lpfir(0);
             lpfirold(26) := lpfir(26) when sireset = '0' else lpfir(26) - lpfir(0);
             lpfirold(27) := lpfir(27) when sireset = '0' else lpfir(27) - lpfir(0);
             lpfirold(28) := lpfir(28) when sireset = '0' else lpfir(28) - lpfir(0);
             lpfirold(29) := lpfir(29) when sireset = '0' else lpfir(29) - lpfir(0);
             lpfirold(30) := lpfir(30) when sireset = '0' else lpfir(30) - lpfir(0);
             lpfirold(31) := lpfir(31) when sireset = '0' else lpfir(31) - lpfir(0);
             lpfirold(32) := lpfir(32) when sireset = '0' else lpfir(32) - lpfir(0);
             lpfirold(33) := lpfir(33) when sireset = '0' else lpfir(33) - lpfir(0);
             lpfirold(34) := lpfir(34) when sireset = '0' else lpfir(34) - lpfir(0);
             lpfirold(35) := lpfir(35) when sireset = '0' else lpfir(35) - lpfir(0);
             lpfirold(36) := lpfir(36) when sireset = '0' else lpfir(36) - lpfir(0);
             lpfirold(37) := lpfir(37) when sireset = '0' else lpfir(37) - lpfir(0);
             lpfirold(38) := lpfir(38) when sireset = '0' else lpfir(38) - lpfir(0);
             lpfirold(39) := lpfir(39) when sireset = '0' else lpfir(39) - lpfir(0);
             lpfirold(40) := lpfir(40) when sireset = '0' else lpfir(40) - lpfir(0);
             lpfirold(41) := lpfir(41) when sireset = '0' else lpfir(41) - lpfir(0);
             lpfirold(42) := lpfir(42) when sireset = '0' else lpfir(42) - lpfir(0);
             lpfirold(43) := lpfir(43) when sireset = '0' else lpfir(43) - lpfir(0);
             lpfirold(44) := lpfir(44) when sireset = '0' else lpfir(44) - lpfir(0);
             lpfirold(45) := lpfir(45) when sireset = '0' else lpfir(45) - lpfir(0);
             lpfirold(46) := lpfir(46) when sireset = '0' else lpfir(46) - lpfir(0);
             lpfirold(47) := lpfir(47) when sireset = '0' else lpfir(47) - lpfir(0);
             lpfirold(48) := lpfir(48) when sireset = '0' else lpfir(48) - lpfir(0);
             lpfirold(49) := lpfir(49) when sireset = '0' else lpfir(49) - lpfir(0);
             lpfirold(50) := lpfir(50) when sireset = '0' else lpfir(50) - lpfir(0);
             lpfirold(51) := lpfir(51) when sireset = '0' else lpfir(51) - lpfir(0);
             lpfirold(52) := lpfir(52) when sireset = '0' else lpfir(52) - lpfir(0);
             lpfirold(53) := lpfir(53) when sireset = '0' else lpfir(53) - lpfir(0);
             lpfirold(54) := lpfir(54) when sireset = '0' else lpfir(54) - lpfir(0);
             lpfirold(55) := lpfir(55) when sireset = '0' else lpfir(55) - lpfir(0);
             lpfirold(56) := lpfir(56) when sireset = '0' else lpfir(56) - lpfir(0);
             lpfirold(57) := lpfir(57) when sireset = '0' else lpfir(57) - lpfir(0);
             lpfirold(58) := lpfir(58) when sireset = '0' else lpfir(58) - lpfir(0);
             lpfirold(59) := lpfir(59) when sireset = '0' else lpfir(59) - lpfir(0);
             lpfirold(60) := lpfir(60) when sireset = '0' else lpfir(60) - lpfir(0);
             lpfirold(61) := lpfir(61) when sireset = '0' else lpfir(61) - lpfir(0);
             lpfirold(62) := lpfir(62) when sireset = '0' else lpfir(62) - lpfir(0);
             lpfirold(63) := lpfir(63) when sireset = '0' else lpfir(63) - lpfir(0);
             lpfirold(64) := lpfir(64) when sireset = '0' else lpfir(64) - lpfir(0);
             lpfirold(65) := lpfir(65) when sireset = '0' else lpfir(65) - lpfir(0);
             lpfirold(66) := lpfir(66) when sireset = '0' else lpfir(66) - lpfir(0);
             lpfirold(67) := lpfir(67) when sireset = '0' else lpfir(67) - lpfir(0);
             lpfirold(68) := lpfir(68) when sireset = '0' else lpfir(68) - lpfir(0);
             lpfirold(69) := lpfir(69) when sireset = '0' else lpfir(69) - lpfir(0);
             lpfirold(70) := lpfir(70) when sireset = '0' else lpfir(70) - lpfir(0);
             lpfirold(71) := lpfir(71) when sireset = '0' else lpfir(71) - lpfir(0);
             lpfirold(72) := lpfir(72) when sireset = '0' else lpfir(72) - lpfir(0);
             lpfirold(73) := lpfir(73) when sireset = '0' else lpfir(73) - lpfir(0);
             lpfirold(74) := lpfir(74) when sireset = '0' else lpfir(74) - lpfir(0);
             lpfirold(75) := lpfir(75) when sireset = '0' else lpfir(75) - lpfir(0);
             lpfirold(76) := lpfir(76) when sireset = '0' else lpfir(76) - lpfir(0);
             lpfirold(77) := lpfir(77) when sireset = '0' else lpfir(77) - lpfir(0);
             lpfirold(78) := lpfir(78) when sireset = '0' else lpfir(78) - lpfir(0);
             lpfirold(79) := lpfir(79) when sireset = '0' else lpfir(79) - lpfir(0);
             lpfirold(80) := lpfir(80) when sireset = '0' else lpfir(80) - lpfir(0);
```

```
  lpfirold(81) := lpfir(81) when sireset = '0' else lpfir(81) - lpfir(0);
end if;
if acount = 22 then --Multiply values in FIR register with corresponding
                    --coefficient
    lpfirint0 := (lpcoeff0*lpfir(0));
    lpfirint1 := (lpcoeff1*lpfir(1));
    lpfirint2 := (lpcoeff2*lpfir(2));
    lpfirint3 := (lpcoeff3*lpfir(3));
    lpfirint4 := (lpcoeff4*lpfir(4));
    lpfirint5 := (lpcoeff5*lpfir(5));
    lpfirint6 := (lpcoeff6*lpfir(6));
    lpfirint7 := (lpcoeff7*lpfir(7));
    lpfirint8 := (lpcoeff8*lpfir(8));
    lpfirint9 := (lpcoeff9*lpfir(9));
    lpfirint10 := (lpcoeff10*lpfir(10));
    lpfirint11 := (lpcoeff11*lpfir(11));
    lpfirint12 := (lpcoeff12*lpfir(12));
    lpfirint13 := (lpcoeff13*lpfir(13));
    lpfirint14 := (lpcoeff14*lpfir(14));
    lpfirint15 := (lpcoeff15*lpfir(15));
    lpfirint16 := (lpcoeff16*lpfir(16));
    lpfirint17 := (lpcoeff17*lpfir(17));
    lpfirint18 := (lpcoeff18*lpfir(18));
    lpfirint19 := (lpcoeff19*lpfir(19));
    lpfirint20 := (lpcoeff20*lpfir(20));
    lpfirint21 := (lpcoeff21*lpfir(21));
    lpfirint22 := (lpcoeff22*lpfir(22));
    lpfirint23 := (lpcoeff23*lpfir(23));
    lpfirint24 := (lpcoeff24*lpfir(24));
    lpfirint25 := (lpcoeff25*lpfir(25));
    lpfirint26 := (lpcoeff26*lpfir(26));
    lpfirint27 := (lpcoeff27*lpfir(27));
    lpfirint28 := (lpcoeff28*lpfir(28));
    lpfirint29 := (lpcoeff29*lpfir(29));
    lpfirint30 := (lpcoeff30*lpfir(30));
    lpfirint31 := (lpcoeff31*lpfir(31));
    lpfirint32 := (lpcoeff32*lpfir(32));
    lpfirint33 := (lpcoeff33*lpfir(33));
    lpfirint34 := (lpcoeff34*lpfir(34));
    lpfirint35 := (lpcoeff35*lpfir(35));
    lpfirint36 := (lpcoeff36*lpfir(36));
    lpfirint37 := (lpcoeff37*lpfir(37));
    lpfirint38 := (lpcoeff38*lpfir(38));
    lpfirint39 := (lpcoeff39*lpfir(39));
    lpfirint40 := (lpcoeff40*lpfir(40));
    lpfirint41 := (lpcoeff41*lpfir(41));
    lpfirint42 := (lpcoeff42*lpfir(42));
    lpfirint43 := (lpcoeff43*lpfir(43));
    lpfirint44 := (lpcoeff44*lpfir(44));
    lpfirint45  := (lpcoeff45*lpfir(45));
```

```
        lpfirint46 := (lpcoeff46*lpfir(46));
        lpfirint47 := (lpcoeff47*lpfir(47));
        lpfirint48 := (lpcoeff48*lpfir(48));
        lpfirint49 := (lpcoeff49*lpfir(49));
        lpfirint50 := (lpcoeff50*lpfir(50));
        lpfirint51 := (lpcoeff51*lpfir(51));
        lpfirint52 := (lpcoeff52*lpfir(52));
        lpfirint53 := (lpcoeff53*lpfir(53));
        lpfirint54 := (lpcoeff54*lpfir(54));
        lpfirint55 := (lpcoeff55*lpfir(55));
        lpfirint56 := (lpcoeff56*lpfir(56));
        lpfirint57 := (lpcoeff57*lpfir(57));
        lpfirint58 := (lpcoeff58*lpfir(58));
        lpfirint59 := (lpcoeff59*lpfir(59));
        lpfirint60 := (lpcoeff60*lpfir(60));
        lpfirint61 := (lpcoeff61*lpfir(61));
        lpfirint62 := (lpcoeff62*lpfir(62));
        lpfirint63 := (lpcoeff63*lpfir(63));
        lpfirint64 := (lpcoeff64*lpfir(64));
        lpfirint65 := (lpcoeff65*lpfir(65));
        lpfirint66 := (lpcoeff66*lpfir(66));
        lpfirint67 := (lpcoeff67*lpfir(67));
        lpfirint68 := (lpcoeff68*lpfir(68));
        lpfirint69 := (lpcoeff69*lpfir(69));
        lpfirint70 := (lpcoeff70*lpfir(70));
        lpfirint71 := (lpcoeff71*lpfir(71));
        lpfirint72 := (lpcoeff72*lpfir(72));
        lpfirint73 := (lpcoeff73*lpfir(73));
        lpfirint74 := (lpcoeff74*lpfir(74));
        lpfirint75 := (lpcoeff75*lpfir(75));
        lpfirint76 := (lpcoeff76*lpfir(76));
        lpfirint77 := (lpcoeff77*lpfir(77));
        lpfirint78 := (lpcoeff78*lpfir(78));
        lpfirint79 := (lpcoeff79*lpfir(79));
        lpfirint80 := (lpcoeff80*lpfir(80));
        lpfirint81 := (lpcoeff81*lpfir(81));
    end if;
    if acount = 23 then--Sum all values together
        lpfirouts := (lpfirint0 + lpfirint1+ lpfirint2 + lpfirint3
                    + lpfirint4 + lpfirint5 + lpfirint6 + lpfirint7
                    + lpfirint8 + lpfirint9+ lpfirint10 + lpfirint11
                    + lpfirint12 + lpfirint13 + lpfirint14+ lpfirint15
                    + lpfirint16 + lpfirint17 + lpfirint18 + lpfirint19
                    + lpfirint20 + lpfirint21 + lpfirint22 + lpfirint23
                    + lpfirint24 + lpfirint25 + lpfirint26 + lpfirint27
                    + lpfirint28 + lpfirint29 + lpfirint30 + lpfirint31
                    + lpfirint32 + lpfirint33 + lpfirint34 + lpfirint35
                    + lpfirint36 + lpfirint37 + lpfirint48 + lpfirint49
                    + lpfirint40 + lpfirint41 + lpfirint42 + lpfirint43
                    + lpfirint44 + lpfirint45 + lpfirint46 + lpfirint47
                    + lpfirint48 + lpfirint49 + lpfirint50 + lpfirint51
                    + lpfirint52 + lpfirint53 + lpfirint54 + lpfirint55
                    + lpfirint56 + lpfirint57 + lpfirint58 + lpfirint59
                    + lpfirint60 + lpfirint61 + lpfirint62 + lpfirint63
                    + lpfirint64 + lpfirint65 + lpfirint66 + lpfirint67
                    + lpfirint68 + lpfirint69 + lpfirint70 + lpfirint71
                    + lpfirint72 + lpfirint73 + lpfirint74 + lpfirint75
                    + lpfirint76 + lpfirint77 + lpfirint78 + lpfirint79
                    + lpfirint80 + lpfirint81)/100000;
                    --Dividing by 100000 correcots for correcting floating
                    --point  to integers
           end if;

            if acount = 24 then--Store current values in register to a new
                                --location in order to shift and update the
                               --register at the start of next cycle
              lpfir(1) := lpfir(0);
              lpfir(2) := lpfirold(1);
              lpfir(3) := lpfirold(2);
              lpfir(4) := lpfirold(3);
              lpfir(5) := lpfirold(4);
              lpfir(6) := lpfirold(5);
              lpfir(7) := lpfirold(6);
              lpfir(8) := lpfirold(7);
              lpfir(9) := lpfirold(8);
              lpfir(10)  := lpfirold(9);
              lpfir(11) := lpfirold(10);
              lpfir(12) := lpfirold(11);
              lpfir(13) := lpfirold(12);
              lpfir(14) := lpfirold(13);
              lpfir(15) := lpfirold(14);
              lpfir(16) := lpfirold(15);
              lpfir(17) := lpfirold(16);
              lpfir(18) := lpfirold(17);
              lpfir(19) := lpfirold(18);
              lpfir(20) := lpfirold(19);
              lpfir(21) := lpfirold(20);
              lpfir(22) := lpfirold(21);
              lpfir(23) := lpfirold(22);
              lpfir(24) := lpfirold(23);
              lpfir(25) := lpfirold(24);
              lpfir(26) := lpfirold(25);
              lpfir(27) := lpfirold(26);
              lpfir(28) := lpfirold(27);
              lpfir(29) := lpfirold(28);
              lpfir(30) := lpfirold(29);
              lpfir(31) := lpfirold(30);
              lpfir(32) := lpfirold(31);
              lpfir(33) := lpfirold(32);
              lpfir(34) := lpfirold(33);
              lpfir(35) := lpfirold(34);
```

```
lpfir(36) := lpfirold(35);
lpfir(37) := lpfirold(36);
lpfir(38) := lpfirold(37);
lpfir(39) := lpfirold(38);
lpfir(40) := lpfirold(39);
lpfir(41) := lpfirold(40);
lpfir(42) := lpfirold(41);
lpfir(43) := lpfirold(42);
lpfir(44) := lpfirold(43);
lpfir(45) := lpfirold(44);
lpfir(46) := lpfirold(45);
lpfir(47) := lpfirold(46);
lpfir(48) := lpfirold(47);
lpfir(49) := lpfirold(48);
lpfir(50) := lpfirold(49);
lpfir(51) := lpfirold(50);
lpfir(52) := lpfirold(51);
lpfir(53) := lpfirold(52);
lpfir(54) := lpfirold(53);
lpfir(55) := lpfirold(54);
lpfir(56) := lpfirold(55);
lpfir(57) := lpfirold(56);
lpfir(58) := lpfirold(57);
lpfir(59) := lpfirold(58);
lpfir(60) := lpfirold(59);
lpfir(61) := lpfirold(50);
lpfir(62) := lpfirold(51);
lpfir(63) := lpfirold(52);
lpfir(64) := lpfirold(53);
lpfir(65) := lpfirold(54);
lpfir(66) := lpfirold(55);
lpfir(67) := lpfirold(56);
lpfir(68) := lpfirold(57);
lpfir(69) := lpfirold(58);
lpfir(70) := lpfirold(59);
lpfir(71) := lpfirold(70);
lpfir(72) := lpfirold(71);
lpfir(73) := lpfirold(72);
lpfir(74) := lpfirold(73);
lpfir(75) := lpfirold(74);
lpfir(76) := lpfirold(75);
lpfir(77) := lpfirold(76);
lpfir(78) := lpfirold(77);
lpfir(79) := lpfirold(78);
lpfir(80) := lpfirold(79);
lpfir(81) := lpfirold(80);
```

```
            end if;

            if acount = 25 then --Corrects sign so that output is in twos
            complement
                                --for DA operations
                if lpfirouts >=0 and lpfirouts <= 65530 then
                    firout := lpfirouts;
                elsif lpfirouts <0 and lpfirouts  >= -65536
                    then firout := 131073 + lpfirouts;
                elsif lpfirouts >
                    65530 then firout
                    := 65530;
                elsif lpfirouts < -
                    65536 then firout
                    := 65537;
                end if;
            end if;
            if acount = 26 then--Final integer to output to DAC
                    adinteger <= firout;
            end if;
        end if;
    end process;

controlDAC: process(runclk, adinteger)
    variable dcount: integer range 0 to 30; --Synchronizes DA clock to runclk
    for
                                        --24 clock cycles when sireset is
    low variable doutsig: std_logic; -- Outputs signal to DASDI
    corresponding to
                              --bit in dabits
    variable dabits: std_logic_vector (160 downto 0); --Stores binary bits
                                                  --to output to DA
    variable fromad: integer range 0 to 262143:=0;
       --Scaled value from adinteger to account for +-10V range
    variable fromadtemp: integer; --Temp hold updating fromad as bits have
                               --been outputted
    variable converthold: integer; --Converts fromadtemp to binary
    variable configswitch: std_logic;
       --Determines whether to send configuration data or voltage
       --data should be sent
    variable configcount: integer range 0 to 1000:= 0;
        --Send 1000 configuration signals prior to beginning conversion

    begin
        dasclk <= runclk when (((dcount >= 3) and (dcount <= 26)) )else '0';
            -- Synchronizes DA clock to runclk for 24 clock cycles
            --when sireset is low
        dasync <= '0' when (((dcount >= 2) and (dcount <= 28)))
                    else '1';
```

```
daldac <= '0' ;
--Unused, must be held low due to chosen configuration
daclr <= '1';
--Unused, must be held high due to chosen configuration
dareset <= '1';
--Unused, must be held high due to chosen configuration
configswitch := '1' when configcount < 1000 else '0';
    --Once configcount reaches 1000, drops low to allow
    --outputting voltages
dasdin <= doutsig when (((dcount >= 3) and (dcount <= 27)) )else '1';
    --Sends signal corresponding to binary integer
    --to DASDIN
    doutsig := '1' when dabits(dcount) = '1' else'0';
    --Signal corresponding to bits in output data

if rising_edge(runclk) then
    fromadtemp := fromad when dcount = 2;
        --Pull fromad in to perform calculations
    dcount := dcount + 1 when dcount < 30 else 0;
        --Increase dcount to synchronize with DASCLK and DA SPI
    converthold := fromadtemp /(2**(24-dcount)) when
                   (dcount >= 7 and dcount <= 24) else 0;
        --If fromadtemp > 2**(24-dcount), will have a remainder
        --of 1 -> needs to output
        --binary 1. If <, remainder 0, output 0
end if;
if falling_edge(runclk) then
    fromadtemp := (fromadtemp - 2**(24-dcount))
                  when (converthold >= 1) else fromadtemp ;
        --Subtract outputted values from fromadtemp
end if;

if configswitch = '1' then --When
    configuring if
    falling_edge(runclk) then
        configcount := configcount + 1 when dcount
    = 0; end if;
    if dcount = 0 then
        dabits(3) := '0'; --First four -> access configuration
        register dabits(4) := '0';
        dabits(5) := '1';
        dabits(6) := '0';
            dabits(7) := '0'; --
            Configuration data dabits(8) :=
            '0';
            dabits(9) := '0';
            dabits(10) := '0';
            dabits(11) := '0';
            dabits(12) := '0';
            dabits(13) := '0';
```

```
                dabits(14) := '0';
                dabits(15) := '0';
                dabits(16) := '0';
                dabits(17) := '0';
                dabits(18) := '1';
                dabits(19) := '1';
                dabits(20) := '0';
                dabits(21) := '1';
                dabits(22) := '1';
                dabits(23) := '0';
                dabits(24) := '0';
                dabits(25) := '1';
                dabits(26) := '0';
        end if;

    else--configswitch = '0' -
        >Outputting data if dcount = 0
        then
            dabits(3) := '0'; --Command to output corresponding
            voltage dabits(4) := '0';
            dabits(5) := '0';
            dabits(6) := '1';
            --Remaining DA bits calculted below
            dabits(25) := '0'; --Not Used
            dabits(26) := '0'; --
        Not Used end if;
        if dcount = 1 then
            --AD4001 -> 16 bit accuracy, 0 = 0V, 32767 = +5V,
            --    32768 = -5V, 65535 = -152.6 uV (Loops around)
            --AD5781 -> 18 bit accuracy, 0 = -10V, 131072 = 0V,
            --    262143 = +10V (Linear from from -10V to +10V)
            --Below if loop corrects for difference in precision and
            --configuration
            if adinteger <= 65536
            then
                fromad := ((adinteger*2) +
            131072);
            elsif adinteger >= 65537 then
                fromad := (adinteger-
            65537)*2;
            end if;
        end if;
        if dcount >= 7 and dcount <= 24 then
            --Output '1' or '0' corresponding to
            remainder if (converthold >= 1) then
                dabits(dcount) := '1';
            else
                dabits(dcount) := '0';
            end if;
        end if;
        end if;
    end process;

end RTL;
end Behavioral;
```

A.3 Additional Electronics Schematics

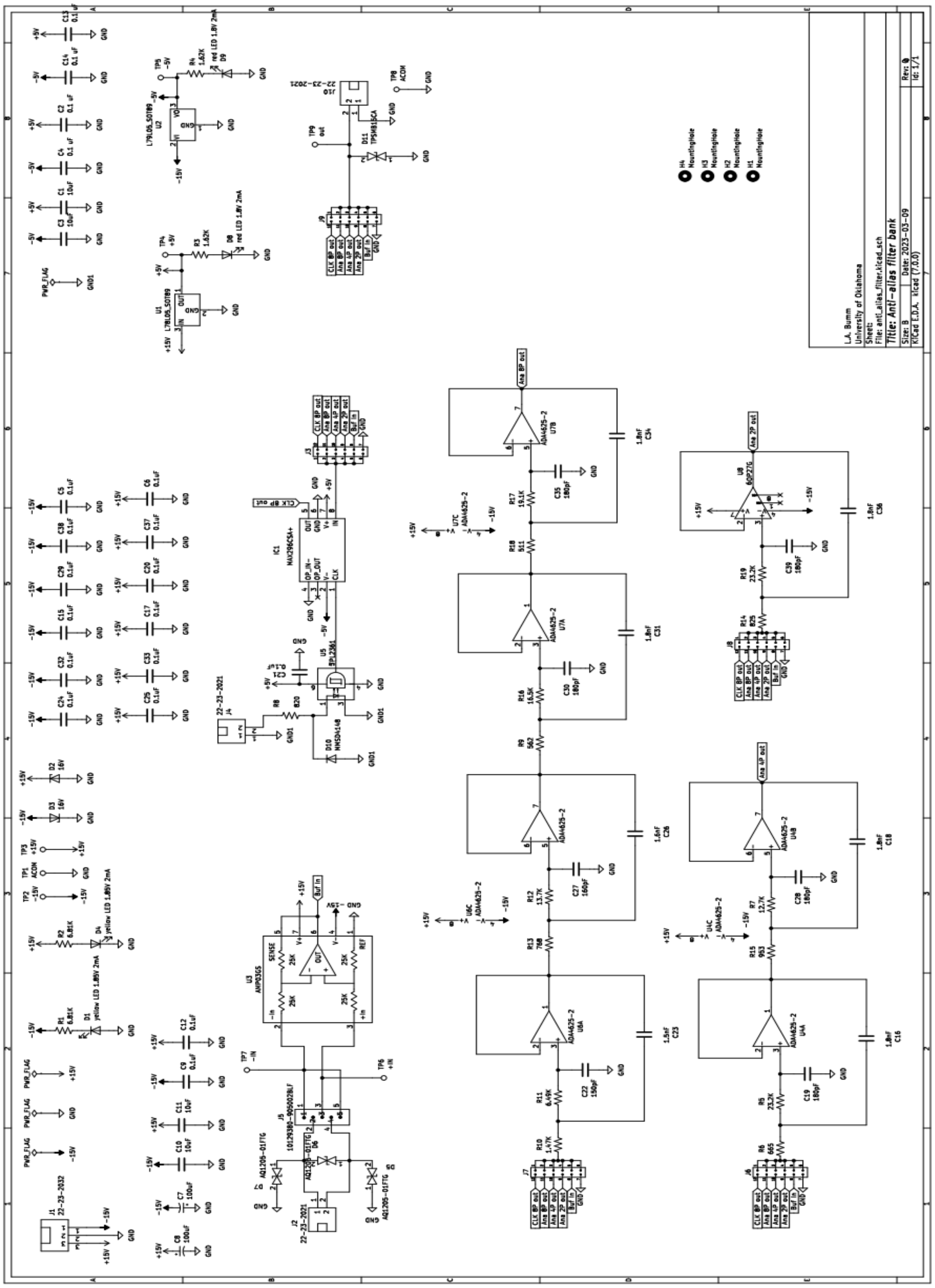


Figure A.1: Schematic of anti-aliasing filter.

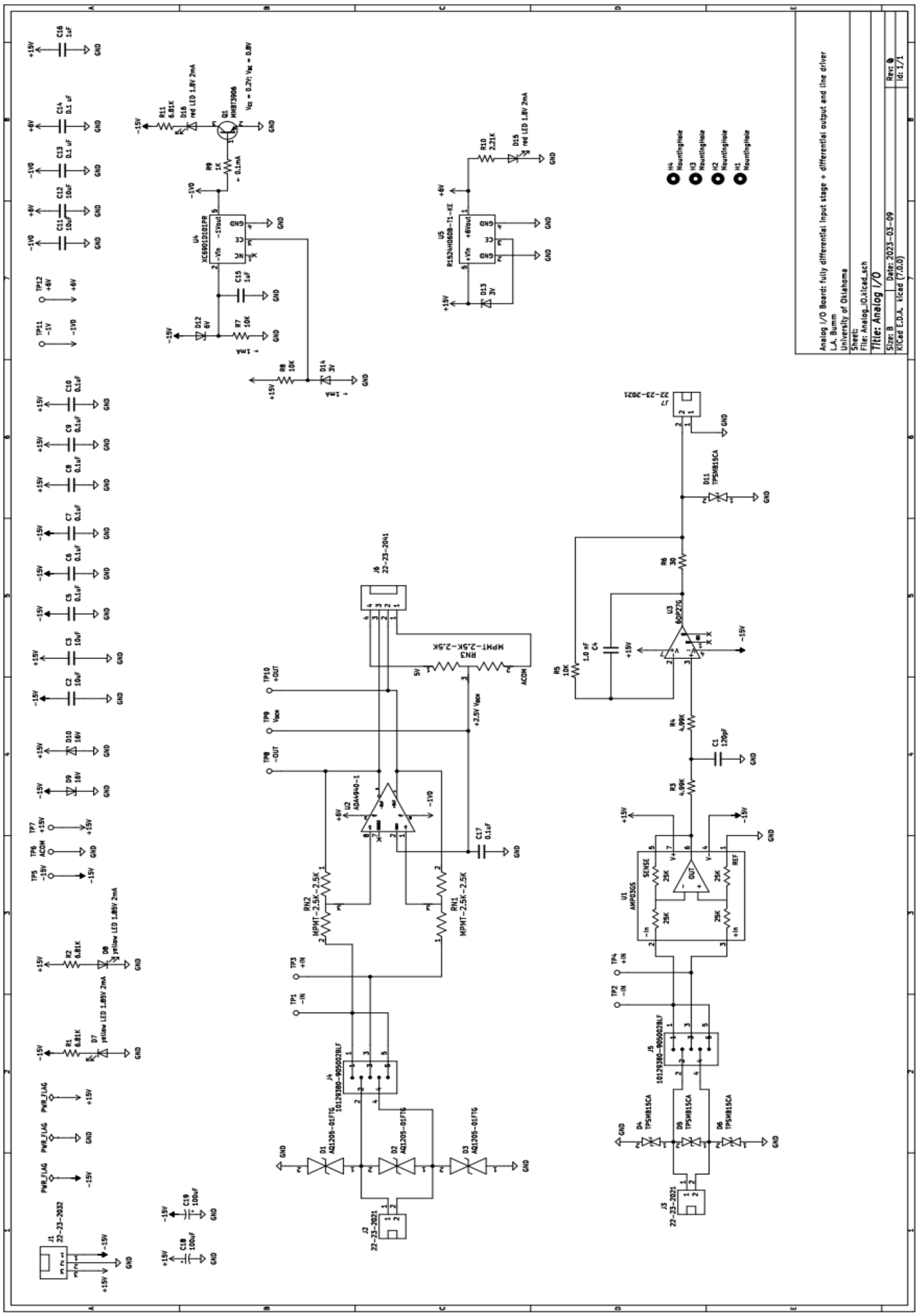


Figure A.2: Schematic of differential amplifier.