# A field-programmable gate array (FPGA)-based data acquisition system for closed-loop experiments

Ian Delgadillo Bonequi,[1] Abraham Stroschein,[1] and Lucas J. Koerner*,[1]

*Department of Electrical and Computer Engineering, University of St. Thomas, St. Paul MN, 55105, USA*

(*The author to whom correspondence may be addressed: koerner.lucas@stthomas.edu)

(Dated: 31 October 2022)

We describe a custom and open source field-programmable gate array (FPGA)-based data acquisition system developed for electrophysiology and generally useful for closed-loop feedback experiments. FPGA acquisition and processing are combined with high-speed analog and digital converters to enable real-time feedback. The digital approach eases experimental setup and repeatability by allowing for system identification and *in situ* tuning of filter bandwidths. The FPGA system includes I2C and SPI controllers, 1 GiB dynamic RAM for data buffering, and a USB3 interface to Python software. The DAQ system uses common HDMI connectors to support daughtercards that can be customized for a given experiment to make the system modular and expandable. The FPGA-based digital signal processing (DSP) is used to generate fourth-order digital infinite impulse response filters and feedback with microsecond latency. The FPGA-based DSP and an analog inner-loop are demonstrated via an experiment that rapidly steps the voltage of a capacitor isolated from the system by a considerable resistance using a feedback approach that adjusts the driving voltage based on the digitized capacitor current.

A field-programmable gate array (FPGA)-based data acquisition system for closed-loop experiments

## I. INTRODUCTION

Rapid advances in control systems theory motivate hardware developments to translate these approaches to the experimental laboratory[1]. Microcontrollers, digital signal processing (DSP) chips, and programmable logic devices such as field-programmable gate arrays (FPGAs) enable digital control systems which have inherent advantages over analog control systems. These advantages of digital control include the ability to rapidly adjust the response of filters; accommodate multiple-input multiple-output (MIMO) control, and; ease the implementation of switchable circuity for system identification of the plant to enable in situ tuning[2]. FPGA-based servos have been described for atomic, molecular, and optical (AMO) physics experiments[3],[4] and for quantum optics experimentation[5]. Biological experiments requiring feedback have been enabled by the Real-Time eXperiment Interface (RTXI)[6] which uses a Real-Time Operating System and PCIe coupled data acquisition cards to implement feedback loops with cycle rates of 100 $\mu$s. Closed-loop systems have also been applied to optogenetic neural control[7] and have used advanced control systems theory such as a state-space single-input multiple-output approach[8].

Specific to large-scale electrophysiology, closed-loop control may minimize aberrant neural activity, induce plasticity, or investigate basic neuroscience questions. An open-source hardware and software solution, the NeuroRighter, used bidirectional multi-electrode arrays (MEAs) and acquisition hardware for simultaneous recording of 8 to 64 channels sampled at 30 kHz and feedback stimulus[9]. The feedback path includes signal processing and stimulation control in an application running on a desktop computer. The system detected interictal spikes from an epileptic animal and responded with microstimulation with a mean latency of 4.4 ms[10]. A robotic system was coupled to in vitro neuron cultures using an MEA, a general-purpose DAQ card, and stimulus from a generator controlled by a desktop computer. Closed-loop control improved the hazard avoidance of the robot as compared to open-loop control[11]. The Multimed platform features 64 analog input channels with programmable gain, 10 digital I/O for stimulus triggers, and an FPGA with modular real-time processing blocks that perform functions, such as action potential detection and channel sorting[12]. A similar multichannel interface acquires 10 kHz bandwidth analog data and then performs online spike-sorting in an FPGA to enable real-time stimulus feedback[13].

Closed-loop approaches in cellular-level electrophysiology include dynamic clamp[14] and response clamp[15]; these techniques respond in real-time to measurements to mimic the complex biological environment of a neuron. A specific development, LCG, supports a suite of these

closed-loop protocols using a real-time Linux system and a COMEDI supported DAQ card[16,17]. Another real-time feedback hardware system was developed around a digital signal processor to emulate neuron models and control strategies without resorting to biological experiments[18]. At the single-cell level, ion channel studies use voltage clamp and current clamp amplifiers with closed-loop feedback that improves accuracy and speed with a target time resolution of about $150\,\mu$s[19]. Historically, partly due to bandwidth requirements, these amplifier systems have used analog feedback circuits. A few commercial digital solutions are emerging[20] for the common single microelectrode patch clamp technique. We are developing a digital feedback amplifier that targets a different single-cell electrophysiology technique termed cut-open Vaseline gap (COVG) that uses multiple electrodes for which the only available amplifier solutions are fully analog with manual control.

Here we describe a custom FPGA-based data acquisition (DAQ) system designed for multi-electrode cellular electrophysiology that is sufficiently flexible to support a variety of experimental needs. This DAQ system is built around an FPGA and high-speed data converter circuits to enable innovative approaches for real-time digital feedback, automated experimental setup, characterization of the device under test (system identification), and continuous data acquisition. This solution is unique in that four channels of 16-bit analog input are sampled at 5 MSPS and the full-resolution waveform can be downloaded to a host computer over USB-3.0. The input streams are processed on the FPGA to create signals that drive DACs for real-time feedback latency $\sim 2\,\mu$s which is considerably faster than solutions described above that implement feedback processing on a desktop computer. To expand the range of possible experiments, the system provides nonstandard channels such as a digitally controlled analog current source and I2C busses to interface with peripheral sensor chips. The DAQ system has multiple inputs and multiple outputs (MIMO) that are read and controlled digitally so that the FPGA-based control system can leverage MIMO state-space design. A standardized electrical interface is used to connect up to four daughtercards that may be customized for specific experiments. The design is open-source, modular, and extensible.

## II. SYSTEM ARCHITECTURE

### A. DAQ Hardware

The DAQ system and digital signal processing (DSP) architecture is based upon a control system requirement for a maximum latency from measurement to actuation of around 2 $\mu$s. To meet this requirement, the DAQ board uses a Xilinx Artix 7 FPGA with 180 DSP slices. Analog input is digitized by four 16-bit 5 MSPS ADCs[21] with a low-voltage differential signaling (LVDS) interface to the FPGA (AD7961). Analog output data is generated by six 14-bit 1 $\mu$s settling time DACs with a SPI interface (AD5453). Each DAC output is conditioned by a unipolar to bipolar amplifier with five programmable gain ranges (full-scale range from $\pm$200 mV to $\pm$15 V). To support system identification for control system tuning the DAQ board contains auxiliary analog input and outputs. Auxiliary analog inputs are digitized by six 16-bit 16 channel acquisition chip with programmable input amplification for each channel. This general purpose (GP) ADC samples at 1 MSPS per group of 8 channels (ADS8686). Additionally, the DAQ system has a set of two 8-channel, 16-bit DACs (GP DACs) used for voltage offset adjustment and stimulus during system identification (DAC80508). A set of these analog outputs are conditioned by unipolar to bipolar amplifiers and can optionally be used to drive a voltage to current converter (Howland current pump[22]). Fig. 1 details the architecture of the DAQ board and demonstrates the signals that are available to each daughtercard (connected via an HDMI cable). The DAQ board supports other general purpose signaling and communication protocols including I2C and digital general purpose input/output signals at voltage levels of 5.0 V, 3.3 V, 2.5 V, and 1.8 V with the 2.5 V signals supporting LVDS.

### B. Modular Daughtercards

The DAQ system supports up to four daughtercards that extend the system capability and can be designed to match a specific experiment. Daughtercards are connected using an HDMI A connector for reasonably inexpensive connectors and cables. The standardized interface from DAQ board to daughtercard includes one high-speed ADC connection, two high-speed DAC connections, auxiliary ADC and DAC connections, power ($\pm$15 V and 5 V), and a dedicated I2C bus (the pin map is tabulated in the Appendix). The I2C bus may be commonly used to configure input/output expanders on the daughtercard, such as the TCA9555, so that many slowly changing configuration

signals are available on the daughtercard to set the state of switches or relays without needing to increase the number of conductors in the cable.

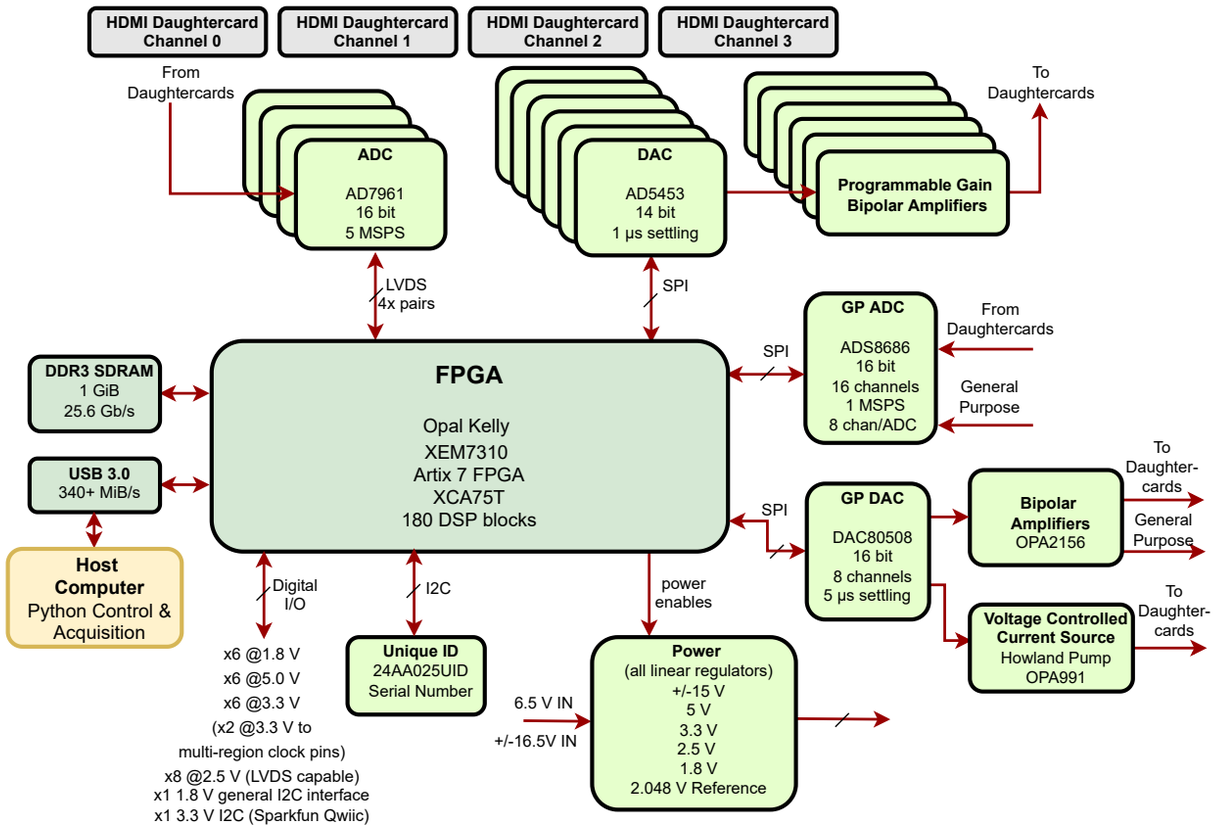



FIG. 1. Block diagram of the data acquisition board with the FPGA module (dark green) in the center. FPGA communication to a host computer (yellow) occurs over USB 3.0 at data-rates over 340 MiB/s. Daughtercard connections are shown in silver with 1x fast analog-to-digital (ADC) and 2x fast digital-to-analog (DACs), power, and I2C provided to each channel. Additional digital I/O, general purpose (GP) ADC and GP DAC channels supplement the daughtercard connectors.

## C. FPGA System

The central component of the DAQ board is an Opal Kelly XEM7310 unit, with a Xilinx Artix 7 FPGA[23]. The FPGA board has a system clock that operates at 200 MHz, as well as 1 GiB of DDR3 memory, and USB 3.0 transfer rates over 340 MiB/s. The Opal Kelly FrontPanel API and HDL endpoints allow for easy communication between the FPGA and a host computer[24] thus creating a method for software-based chip configuration and data acquisition. This FPGA communicates

A field-programmable gate array (FPGA)-based data acquisition system for closed-loop experiments with several ADCs and DACs on the DAQ board using LVDS and SPI interfaces, as well as various chips over the I2C protocol. Fig. 2 details the FPGA architecture which is described below.

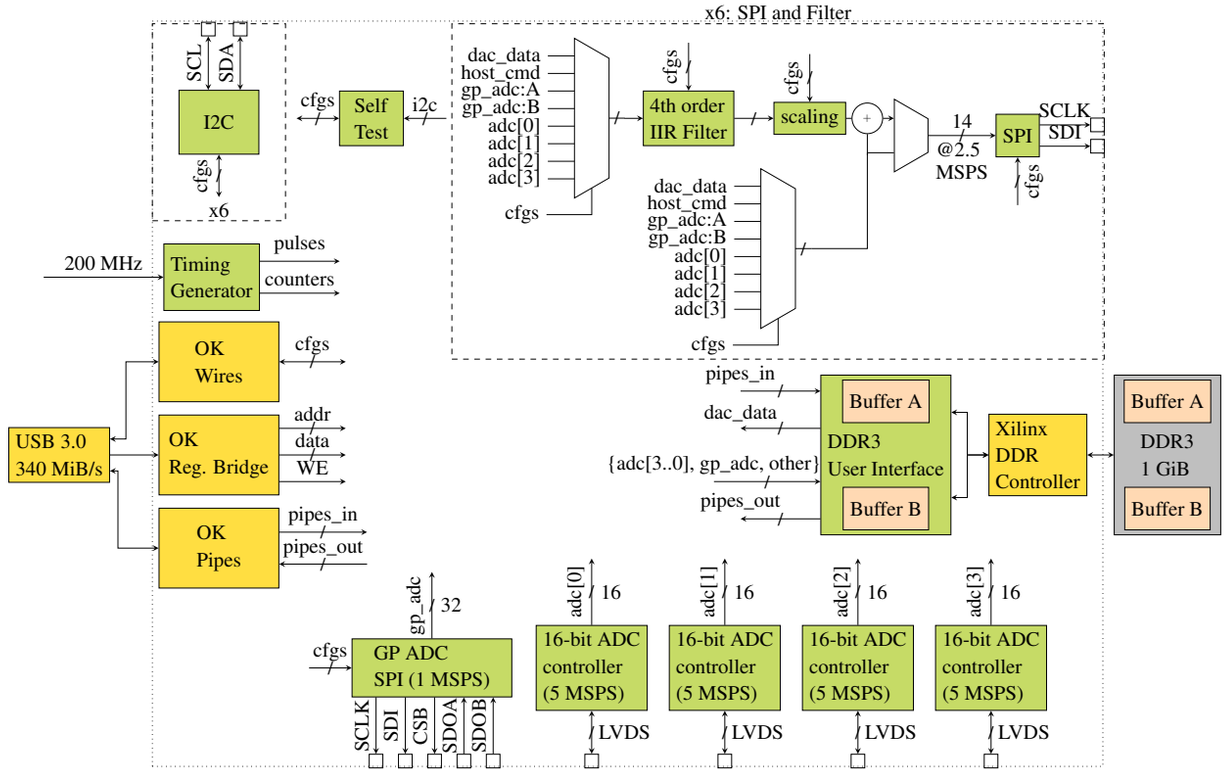



FIG. 2. Block diagram of the FPGA design that shows the main components including the USB 3.0 Host interface. The dotted line indicates the boundary of the FPGA. *cfgs* represents signals available from the USB interface over which configurations are set or read. Not all connections are shown.

### D. Analog Input LVDS Interface

Implemented on the DAQ board are four high-speed (5 MSPS) 16-bit ADCs that use an LVDS interface to the FPGA, with one cycle latency ideal for control[21]. The HDL for the ADC controller was provided by Analog Devices and enables data transfer over LVDS at the maximum 5 MSPS rate[25]. The ADC controller continuously provides data to the 4th-order filter at 5 MSPS. The filter output is continuously available to the SPI modules and can be selected as the DAC output data by the host computer through a multiplexer.

### E. Serial Peripheral Interface

A Serial Peripheral Interface (SPI) transceiver communicates with and processes data from the general purpose ADC and the DACs (see SPI block in Fig. 2). The FPGA design uses a SPI controller HDL module available from Open Cores[26]. This core is full duplex, can transmit variable word sizes up to 128 bits, and includes several useful features such as controllable configuration for transmitting MSB, or LSB first, transferring bits on rising or falling edge of the serial clock, a programmable SPI clock frequency, and up to 8 slave select lines[26]. A lightweight open-source protocol for commanding a Wishbone bus[27] converts signals from the host computer to the interface of the SPI controller. The Wishbone controller, and SPI controller cores form the backbone of the SPI communication that takes place on the FPGA. In addition to these cores, a state machine controls the interactions that occur between the SPI chips and the host computer. For example, in the case of the high-speed DACs (AD5453) used for closed-loop control, consistent SPI updates at 2.5 MSPS are required. The designed state machine transmits SPI commands at 2.5 MSPS by cycling through the required Wishbone signals to continuously output data provided by FPGA memory or ADCs for timing sensitive applications. The SPI data source can be switched to the host PC so that the host computer has full control over data sent to the DACs. In this mode, due to the communication latency of the USB interface, timing is not guaranteed.

The state machine configures the SPI controller internal registers at the startup of an experiment. Through the lightweight Wishbone controller the appropriate Wishbone commands are sent to the SPI master controller which allows the host computer the option to change a subset of the SPI controller registers, such as the SPI clock frequency. Each DAC is allocated a separate SPI controller to enable simultaneous unique updates. Each SPI controller is the only peripheral on its own Wishbone bus such that each DAC is allocated a Wishbone master core and state machine. The host computer configures the state machine to either directly pass commands to the SPI controller through the Wishbone core, or configures it to cycle through Wishbone commands at a fixed frequency, as previously discussed. The SPI controller and Wishbone controller pairs utilize few FPGA resources making this design choice feasible.

### F. DDR3 Memory

The Opal Kelly XEM7310 module includes 1 GiB of 32-bit wide onboard DDR3 SDRAM, with a peak bandwidth of 25.6 Gb/s[23]. The FPGA DDR3 memory access is implemented as a dual-port interface and is used for the reading and writing of waveforms to and from the FPGA. The DDR3 enables playback of continuous analog output through DACs, as well as continuous capture of data from the ADCs onboard the DAQ board. The architecture of the DDR3 interface includes a memory interface IP (Xilinx DDR Controller in Fig. 2). Each port of the DDR access has two FIFO buffers, one incoming and one outgoing, and a state machine controller to arbitrate access between each of the four buffers and maintain memory address pointers. Bulk data transfers are made from the host PC to the DDR3 memory, for loading DAC waveforms, and from the FPGA to the host for reading ADC data.

DDR 'buffer A' (see Fig. 2) implemented within the user interface provides data to the DACs to, in effect, create programmable function generators. The host PC has complete control over when it can write to the DDR3 memory, and when to enable reading of data to the DACs. This buffer is composed of 8 channels, with data arranged into 16-bit groups. These channels include 14-bit data for x6 AD5453 chips, and 16-bit data for x2 DAC80508 chips, with 2 bits of DAC80508 channel selection data placed in the 2 most significant bits of the 16-bit grouping for AD5453 DAC channels. 224 MiB of data are used for 5.87 s of looped playback for each of eight two-byte DAC channels giving a frequency resolution of 0.17 Hz. The second DDR3 port, 'B', buffers data gathered from four AD7961 ADC chips operating at 5 MSPS, the ADS8686 general purpose ADC operating at a maximum sampling rate of 1 MSPS, as well as data being sent to the six fast DACs (AD5453) discussed previously (operating at 2.5 MSPS). This port also stores timestamp data for a total data rate of 80 MB/s. In similar fashion to the first buffer discussed above, the host PC has complete control over when to enable ADC writing of data to DDR3 memory, as well as when to read out the data stored in the DDR3. The USB 3.0 data transfer rate averages around 200 MB/s for bulk transfers which allows for continuous data acquisition while still sending other configuration updates.

### G. Host PC Interface and Software

Opal Kelly FPGA modules provide HDL and host APIs that enable communication between the FPGA and the host computer over the USB 3.0 interface. The communication system provides single clock period duration trigger pulses, simple 32-bit wide busses for time insensitive configurations, and high-speed input and output bulk data transfers. The DAQ system uses each of these Opal Kelly endpoints. Pulse trigger signals reset modules and trigger state machines; these incoming pulses are conveniently synchronized to the proper clock domain. Incoming trigger signals prompt the host computer of certain scenarios such as newly available data. 32-bit wide input busses deliver status, enable, and configuration bits to the FPGA, and 32-bit wide output busses deliver status bits to the host computer. Bulk data is transferred from first-in first-out memories (FIFOs) that hold ADC and DAC data typically before or after buffering to the 1 GiB of DDR3 memory on the FPGA module. The host to FPGA interface includes a 32-bit data, 32-bit address register space that we use to program various FPGA configurations such as coefficients of the digital filters[24].

The host computer software that interfaces to the OpalKelly APIs is written in Python. This software organizes and abstracts chip registers, the communication protocol, and the host computer interface to each communication controller (e.g. SPI or I2C). The software uses a register map spreadsheet to create references between a data field name and a register address and bit indices. The software builds a message using the register address and bit index before sending the message over the Opal Kelly FrontPanel API to the corresponding hardware-level communication controller on the FPGA.

### H. Digital Signal Processing

The FPGA implements digital filters written in HDL. These filters operate on measurements taken by ADCs on the DAQ board. The filters were generated using the MATLAB Filter Designer Tool and translated to Verilog HDL using the MATLAB Filter Coder Tool. The filters include a processor interface to allow writing of filter coefficients, resulting in a tunable cutoff frequency.

The filters implemented on the FPGA are four pole infinite impulse response (IIR) filters with a tunable cutoff frequency and coefficients generally programmed to emulate four pole analog Bessel filters. Filter output is either sent straight to a fast DAC, or is summed with a 14-bit

command signal from the DDR3 on the Opal Kelly XEM7310 module, with the result of the sum then being sent to a fast DAC; the desired function is selected by the host computer. The filter design constraints include a 16-bit input and a 14-bit output for the fast DACs. Filter cutoff frequency is tunable from 200 kHz to 1 MHz when operating with a 5 MHz sampling rate. The rationale for selecting an IIR filter over a finite impulse response (FIR) filter for implementation within the real time control system was heavily influenced by a goal to reduce latency. While FIR filters have multiple advantages over IIR implementations, FIR filters need more taps, and thus use more memory and have a greater latency, to achieve a similar magnitude response to an IIR filter[28]. The end application prioritizes minimal filter latency so an IIR implementation was selected.

For fixed-point filter design with MATLAB tools, different cutoff frequencies input to the Filter Designer Application result in different filter internal widths and fraction lengths when a filter is converted to fixed-point format. Insufficient register widths can cause overflow and wraparound in the filter output if the cutoff frequency is modified from that used during design. To avoid overflow, the filter internal widths and fraction lengths were determined for all possible cutoff frequencies. In practice, the filters were designed for a cutoff frequency near the middle of the desired cutoff frequency range with sufficient margin, so that the filter exhibited output as expected for the extremes of the desired cutoff frequency range.

Butterworth coefficients were selected for implementation on the FPGA due to the resulting lower group delay as compared to three other options considered (Chebyshev I & II, and Elliptic). HDL code was generated using MATLAB's Filter Design HDL Coder application. The filter architecture consists of cascaded second order sections (SOS) in the Direct Form II, transposed arrangement. All coefficients are 32 bits, section inputs and outputs are 16 bits, and numerator and denominator products and accumulators are 48 and 49 bits respectively. Fig. 7 shows the simulated Bode plot for 250 kHz cutoff frequency ($f_c$).

The 4th order IIR filters include a processor interface to program coefficients and set the cutoff frequency. The filters are composed of cascaded SOS with a pipeline register between the sections and a 4-stage pipeline Xilinx multiplier was shared for all multiply operations within each of the second order sections of the filters. As compared to a multiplier for every operation, a shared pipelined multiplier reduced the DSP slice utilization for six instantiations of the filter within the FPGA from 100% to just 18% while simultaneously providing more consistent timing. Data processing steps that follow the filter are detailed in the Appendix.

## III. SYSTEM CHARACTERIZATION

### A. Data Acquisition Characterization

Fig. 3 demonstrates the fast 5 MSPS ADC with a 160 kHz sine-wave analog input signal from a waveform generator (Keysight 33500B) buffered by a fully differential ADC driver (LMP8350). The upper left inset of Fig. 3 displays a short segment of the time domain waveform while the main axes shows the FFT of 8.19 ms of data processed with a Hanning window showing a prominent peak at 160 kHz.

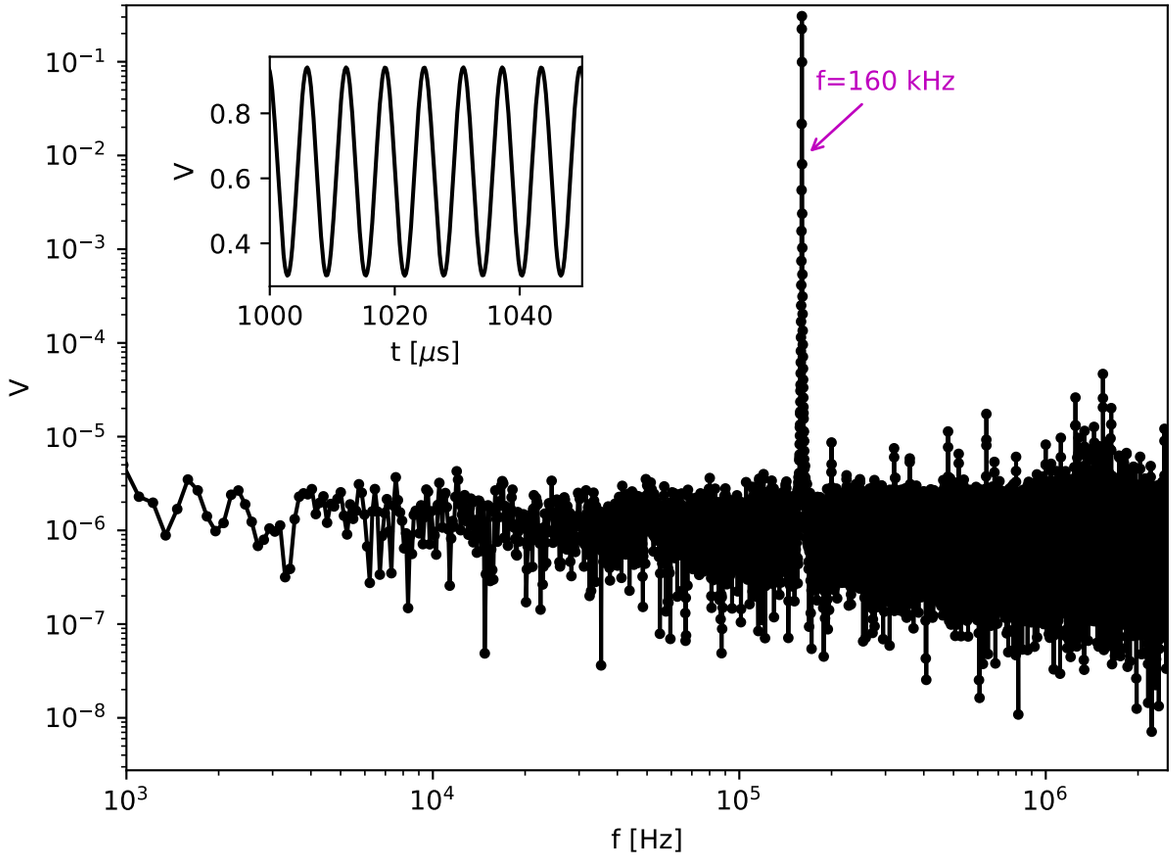



FIG. 3. Digitization of a 160 kHz sine-wave input to a fast ADC channel. The inset shows a subset of the time-domain waveform while the main axes shows the resulting FFT.

Fig. 5 shows measurements with a custom daughtercard connected to an HDMI port with a simplified schematic of the board in Fig. 4. This daughtercard controls the voltage at node $V_M$ while measuring the current flowing to $C_M$. Real-time feedback can be implemented using the

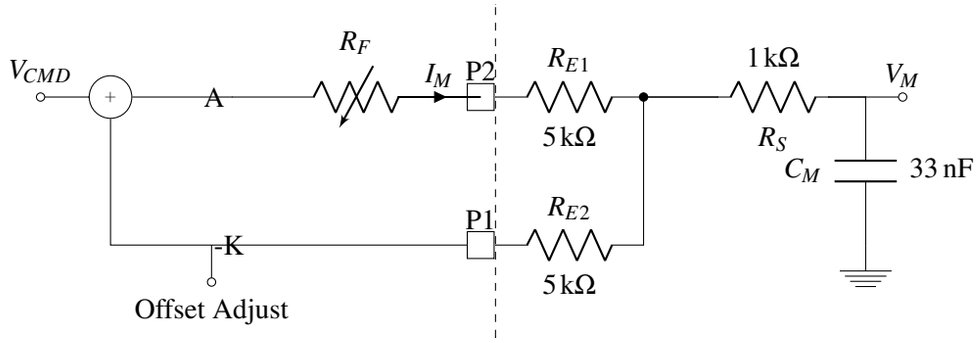



FIG. 4. The left of the dashed vertical line shows the custom daughter-card design while the right of the line is the plant to control.

measurements of the current $I_M$ and the voltage at node $P1$ (discussed in Section IV). Additional functionality of the daughtercard (not shown) includes relays near the connections to the plant that enables injection of and digitization of signals to characterize the electrical impedance of the connected plant for a calibration step before closed-loop control. The top panel of Fig. 5 presents a loop-back test of GP DAC to GP ADC with the connection made via a calibration port of the daughtercard and an uploaded DAC sine-wave at $f = 12\,\text{kHz}$. In this demonstration, the recorded ADC data lag of the programmed DAC data by $4\,\mu\text{s}$ is not significant since lag between the uploaded DAC waveform and the analog output waveform are expected. Fig. 5 bottom shows the analog feedback loop of the daughtercard controlling the capacitor voltage while digitizing the current $I_M$ at a sample rate of 5 MSPS. The differential voltage across $R_F$ is amplified by an instrumentation amplifier before driving the fast ADC on the DAQ board.

The control system design benefits from characterization of the plant and allows for online adjustments of the controller due to the digital implementation. The DAQ board is designed to measure unknown impedances in support of system identification using the GP DAC to output a single-frequency ($f_t$) sine-wave that is digitized at two points by the GP ADC. Fig. 6 demonstrates this measurement of an unknown impedance ($R_U$ in series with $C_U$) by digitizing the voltage at $V_{in}$ and $V_{mid}$ captured by two channels of the GP ADC. The FFT of these signals at the excitation frequency is used to evaluate the unknown complex impedance at $f = f_t$ as

$$Z(f) = \frac{\mathscr{F}(V_{mid})}{\mathscr{F}(V_{in} - V_{mid})/R_K}. \tag{1}$$

Results at test frequencies of 10, 15, and 20 kHz are shown in Table I for component values of $R_U = 9.866\,\text{k}\Omega$ and $C_U = 884.5\,\text{pF}$ as measured by an LCR meter (GW Instek LCR-6300).

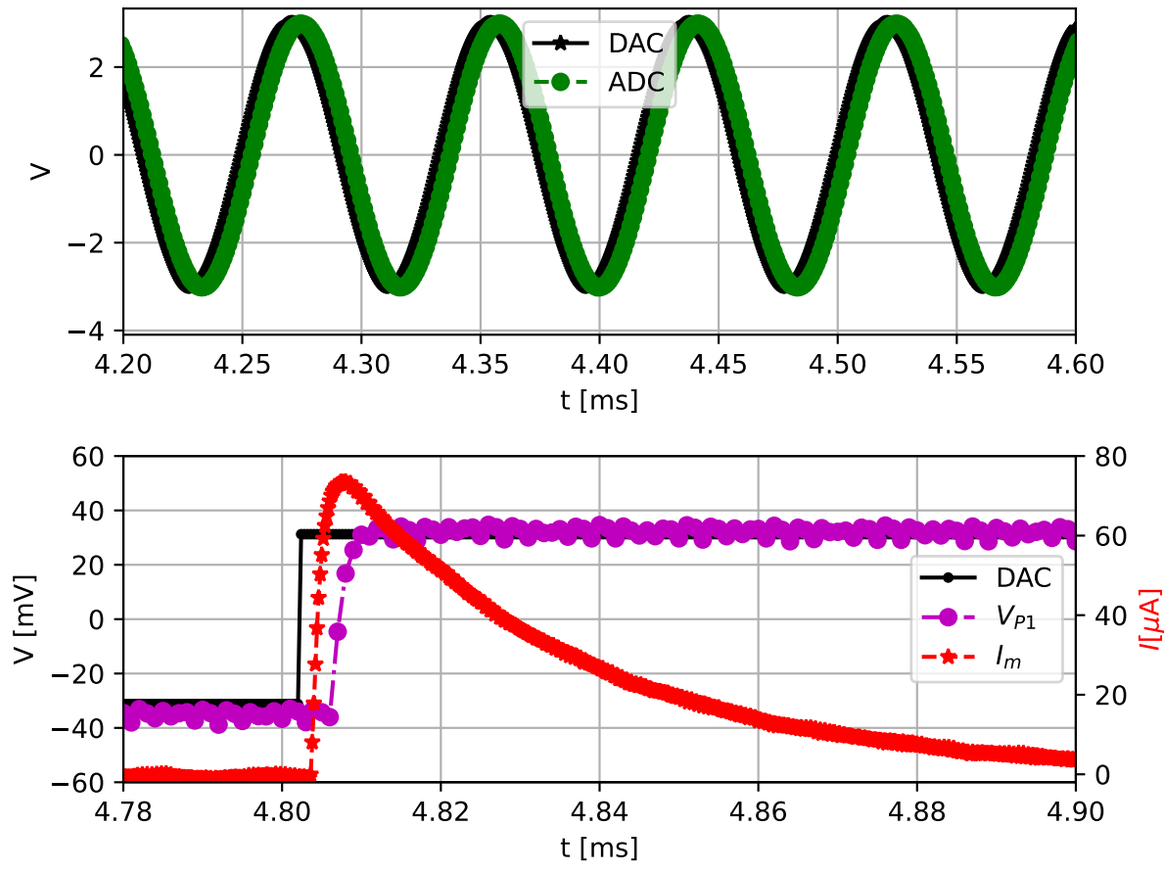



FIG. 5. (top) Loopback test of the general purpose DAC to the general purpose ADC at $f = 12\,\text{kHz}$. (bottom) The voltage clamping and current measurement application of the daughtercard architecture of Fig. 4. The voltage at node $V_{CMD}$ is set by a DAC (black), the resulting sense voltage at node $P1$ is digitized at 1 MSPS (magenta), and the membrane current $I_M$ is digitized at 5 MSPS (red). Using three separate daughtercards these waveforms (and the fast ADC FFT of Fig. 3) were collected simultaneously.

| $f$ | Re($Z$) | Im($Z$) | Re($Z$) error | Im($Z$) error |
|---|---|---|---|---|
| 10 kHz | 9977.7 Ω | -17351.7 Ω | 1.13 % | -3.57 % |
| 15 kHz | 9769.7 Ω | -11625.3 Ω | -0.98 % | -3.09 % |
| 20 kHz | 9689.7 Ω | -8767.9 Ω | -1.79 % | -2.55 % |

TABLE I. Measurements of the real and imaginary components of an unknown impedance ($Z$) using the DAQ system at three separate test frequencies ($f$). The percent error of the measurement from the calculated value using the measured component values is indicated in the two rightmost columns.

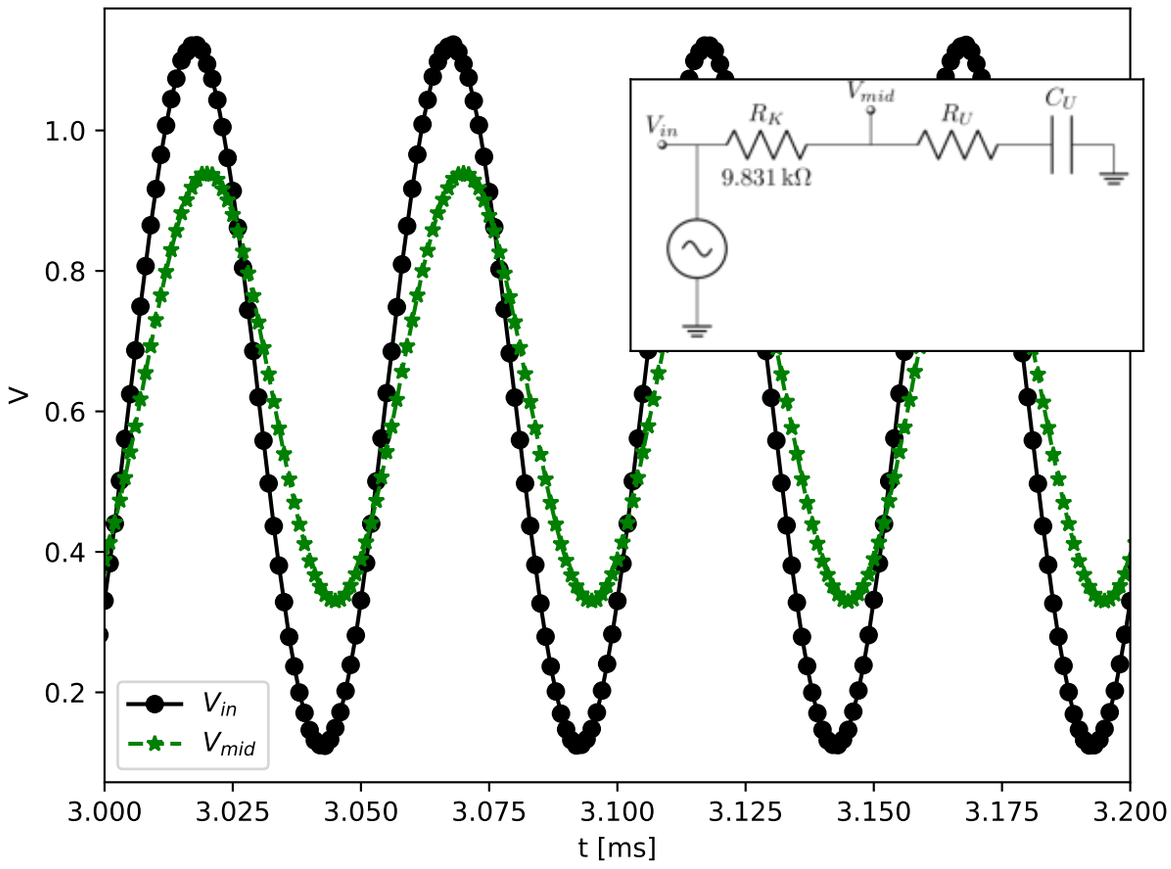



FIG. 6. Two ADC channel measurements to determine an unknown impedance using a 20 kHz excitation frequency. The circuit schematic is shown in the inset with a known resistance ($R_K$) and two unknown components $R_U$ and $C_U$.

### B. Filter Evaluation

The IIR filters implemented within the FPGA and discussed in Section II H are intended for real-time feedback. Thus it is critical to evaluate the performance of the implemented filters in comparison to theoretical models so accurate models are available for control system simulations. Filter models require measurements of the total end-to-end latency from input to output, including the entire data path of data moving through the analog electronics to ADC to filter to DAC. The following sections compare the measured transfer function of the 4th order IIR filters to the theoretical MATLAB calculated filter performance and show measurements of the latency.

To evaluate the magnitude response of the implemented 4th order IIR filters, waveforms were loaded to the DDR3 memory, passed through the filters, and recorded back to the DDR3. The host PC downloaded the DDR3 data and saved the data into .h5 files, which were read and processed offline. Sinusoidal waveforms of equal amplitude and frequency from 100 Hz to 1 MHz with 20 points per decade were tested. Due to the design of the HDL responsible for reading data from DDR3 memory and passing the data to the filters, the filter evaluation was done at a sampling rate of 2.5 MSPS (as opposed to the sampling rate of 5 MSPS used when filter data is provided by the fast ADCs). Despite this, the filters could still be accurately evaluated for their magnitude response performance since the cutoff frequency for given filter coefficients scales with the sampling frequency, and thus the magnitude response for a given set of coefficients is still predictable. These tests were conducted with filter coefficients calculated for a 4th order Butterworth filter with sampling frequency 5 MHz and a cutoff frequency of 500 kHz which, at a 2.5 MHz sampling rate, corresponds to an expected cutoff frequency of 250 kHz.

Fig. 7 shows the measured magnitude response calculated as the difference between the maximum and minimum values normalized to the amplitude of the sinusoidal waveform input. The Bode plot demonstrates excellent correspondence of the -3 dB point between measurements and theoretical response (double precision coefficients) at a sampling rate of 2.5 MSPS. These measurements prove that the filter output measured on the hardware matches well to the expected theoretical performance.

Filter and complete datapath latency were measured using an oscilloscope and DDR recordings of data entering and leaving the filter. Datapath latency arises from: 1) the fast ADC conversion and data transfer time; 2) fast DAC SPI transaction and DAC settling time; 3) amplifier bandwidth and slew-rate at the DAC output; 4) data transfer through the filter; 5) filter latency depending on

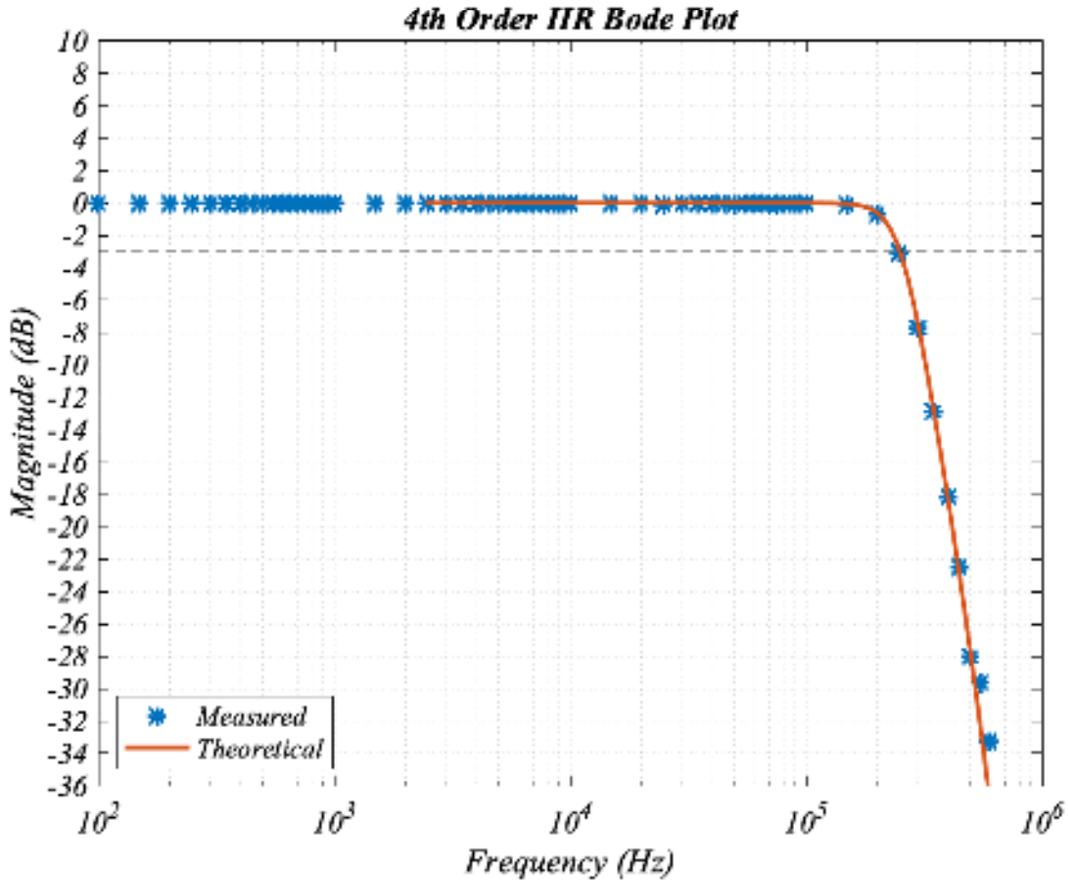



FIG. 7. Bode plots of the experimental and theoretical IIR filters. The dotted line indicates the -3 dB point and intersects with the cutoff frequency location of the Bode plots. The red line is the theoretical frequency response of the 4th order IIR implementation, as calculated by MATLAB using double precision Butterworth coefficients with $f_c$ = 250 kHz. The blue dots indicate the frequency response as calculated from digital filter data captured via FPGA onboard DDR3 memory and shows near perfect correspondence with the theoretical trace.

the cutoff frequency; and, 6) latency of analog circuits that generate the ADC input (for example, those on the custom daughtercard). Datapath latency characterization proceeded by inputting a step response from a waveform generator (Keysight 33500B) buffered by a fully differential ADC driver (LMP8350) into a fast ADC channel with the digitized signal output to a fast DAC channel. The time difference between input and output signals were measured with an oscilloscope. The latency results, differentiated by filter configuration, are:

- Filter bypassed (includes latency sources 1, 2, 3): 0.80 $\mu$s rise start; 1.30 $\mu$s settling time.

A field-programmable gate array (FPGA)-based data acquisition system for closed-loop experiments

- Filter enabled, unity-gain coefficients (latency sources 1-4): $1.44\,\mu$s rise start; $1.94\,\mu$s settling time.
- Filter enabled, $f_c = 500\,$kHz (latency sources 1-5): $2.00\,\mu$s rise start; $3.00\,\mu$s settling time.

## IV. EXAMPLE EXPERIMENTAL APPLICATION

The applications of an open-source DAQ system with real-time feedback are numerous. In the following section, we demonstrate an electrophysiology experimental application for which the DAQ system was originally designed.

In the context of electrophysiology experiments, a bath clamp circuit is used to set, or clamp, the membrane voltage, while measuring the membrane current to study the electrical properties of ion channels. Our data acquisition system is designed for a paricular electrophysiology technique named Cut-Open Vaseline Gap (COVG). For this application a custom daughtercard described in the schematic of Fig. 4 has been designed and fabricated.

COVG is a voltage clamp electrophysiology technique used to characterize ion channel electrical behavior that provides a unique combination of high-speed/low-noise needed to resolve fast protein dynamics and the rare ability to control both intra- and extra-cellular solutions[29]. Despite these advantages, experimentalists regard the COVG technique to be challenging because previously developed manually operated amplifier systems require considerable experience and skill. The development presented here leverages modern digital electronics to make the COVG technique more accessible and repeatable. The electronic system must control the voltage across the cell membrane while measuring the current through the cell membrane with a dominant time constant to control of $33\,\mu$s set by the access resistance ($R_s = 1\,\text{k}\Omega$) and the membrane capacitance of $C_M = 33\,$nF (Fig. 4).

For the following experiments the custom daughtercard creates an analog feedback loop that clamps the membrane voltage (set by $V_{CMD}$) while measuring the membrane current resolved across $R_F$ (see Fig. 4). The daughtercard design allows for I2C I/O expanders to control relays and multiplexers that enable the experimentalist to disconnect, connect, or ground electrodes. This digital control of the electrode state will ease setup as other amplifiers often require the user to physically insert and remove electrodes from the conductive solution to verify proper setup. Digital configuration bits set the value of the sense resistor $R_F$ to tradeoff clamp speed and current noise.

To evaluate the transient voltage clamping of the daughtercard and DAQ system two daughtercard outputs were connected to a network of resistors and capacitors that emulate the resistance of electrodes and the capacitance of the cell membrane. An electrical schematic of this cell model is shown to the right of the dashed line in Fig. 4. A DAC on the main DAQ board generates a step function command voltage at the node $V_{CMD}$ in Fig. 4. This waveform is generated by the host computer and loaded to the FPGA DDR3 memory. The daughtercards are configured for closed-loop analog feedback and to route the differential voltage taken across $R_F$ to a fast ADC on the DAQ board, allowing the FPGA access to digitized current measurements. These digitized current measurements are routed to the 4th order IIR filters within the FPGA. Initial tests of voltage clamping performance without digital feedback measured a rise time (10% - 90%) of $\sim$70 $\mu$s in the membrane voltage ($V_M$) response as set by the 33 $\mu$s time constant created by $R_S$ and $C_M$ and measured by the general purpose ADC.

With the analog feedback proven, similar experiments were conducted to leverage the DAQ system capabilities by applying real-time digital feedback to improve the voltage clamping speed. The feedback signal added to the control loop is the digitized membrane current which is used to implement series resistance compensation[30,31]. The method uses the FPGA (see the 'SPI and filter' block in Fig. 2) to filter the measured current ($f_c = 500$ kHz), compute the voltage drop across the resistance introduced by the solution, scale it by a fraction $\alpha$, and then sum this resulting compensation voltage with the target membrane voltage command signal in real time. The compensation voltage is calculated as:

$$V_{COMP} = \alpha(I_M R_s). \tag{2}$$

By over-driving the command voltage, the voltage drop across the series resistance, $R_S$, is compensated for, resulting in improved rise times of the membrane voltage step response. In theory, when $\alpha$ is set to one, the membrane voltage should follow command signals exactly, however in practice, when $\alpha$ approaches one, the control loop becomes unstable with positive feedback. To test the feedback performance, the $\alpha$ factor was varied from 0.05 to 0.95 (5% - 95% compensation) by writing via the register bridge, to the post-filter data processing blocks detailed in the Appendix. The membrane voltage was monitored with an oscilloscope and digitized using the DAQ system general purpose ADC, for offline observation of the effects of the magnitude of series resistance compensation ($\alpha$).

Fig. 8 shows digitized waveforms of $I_M$ (5 MSPS), $V_M$ (1 MSPS), and $V_{CMD}$ (2.5 MSPS) versus

time for select $\alpha$ values and Table II summarizes calculated step response metrics for the membrane voltage, $V_M$. The rise time (10-90%) of the membrane voltage improved to $\sim$23 $\mu$s with 50% $R_S$ compensation. This indicates significant improvement in the clamping speed of the analog control loop, when digital feedback signals were added to the analog feedback. When $\alpha$ is increased to values beyond 0.8, significant overshoot ($\sim$ 50%) is observed in the membrane voltage response, and with $\alpha$ of 0.95, the membrane voltage oscillates, indicating that the feedback loop reached instability.

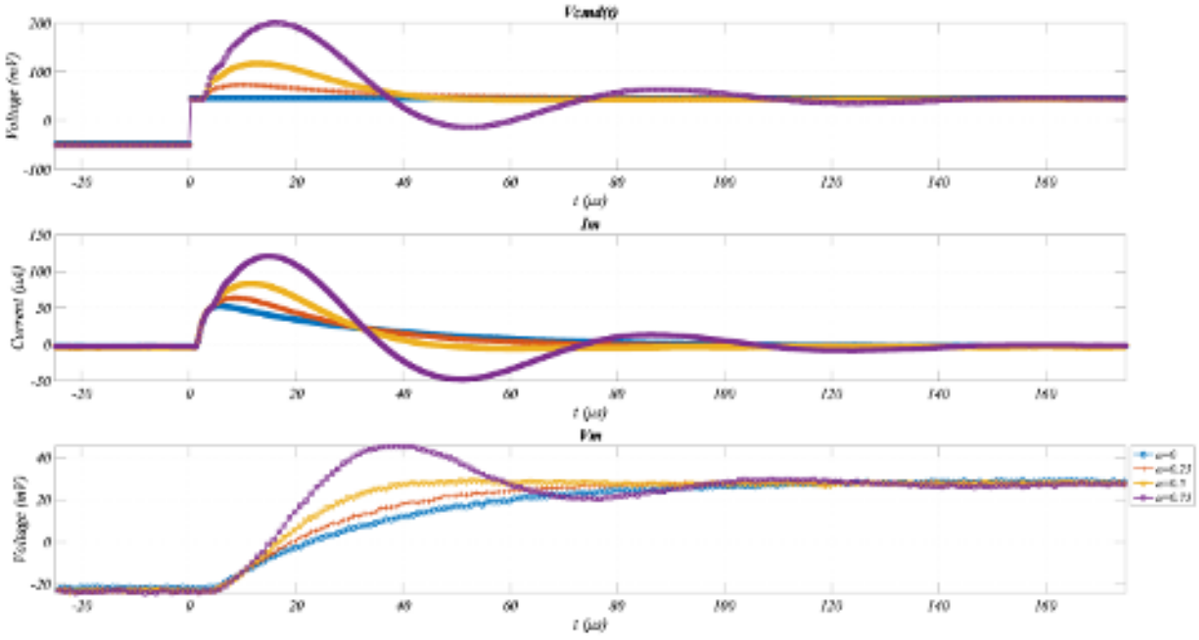



FIG. 8. Top: command voltage (CMD); center: membrane current ($I_M$); and bottom: membrane voltage ($V_M$) with select $\alpha$ values. From these traces, the effects of increasing $\alpha$ are readily observed. Rise times (10% - 90%) of the membrane voltage, $V_M$, improve significantly as $\alpha$ increases. At $\alpha > 0.5$ the voltage response degrades due to overshoot, extended settling time, and ultimately instability.

In addition to the control methods outlined above, we are exploring more advanced control methods. One promising method uses a proportional-integral (PI) controller in a loop-shaping approach wherein the existing analog feedback transfer function is treated as a forward path in a control loop that is closed digitally with direct feedback of the membrane voltage. Measurements of this method show rise times, in response to a membrane voltage step command, of $\sim$16 $\mu$s, with minimal overshoot, and zero steady-state error. Future work will add a Luenberger observer

| | RiseTime [$\mu$s] | Overshoot [%] | Settling Time [$\mu$s] |
|---|---|---|---|
| $\alpha$=0 | 66.2 | 1.7 | 99.5 |
| $\alpha$=0.25 | 46.4 | 0.2 | 74.9 |
| $\alpha$=0.5 | 22.4 | 3.7 | 33.8 |
| $\alpha$=0.75 | 13.5 | 37.3 | 110 |

TABLE II. Rise time, percent overshoot, and settling time of the membrane voltage step response versus the magnitude of series resistance compensation. The measurements correspond to the waveforms of $V_M$ from Fig. 8. Rise time is from 10-90% and settling time is the time to reach and stay within $\pm$5% of the final value.

that allows for virtual feedback of membrane voltage — a necessary feature for electrophysiology experiments where membrane voltage cannot be directly measured.

End-to-end latency was also evaluated for this experimental configuration since analog circuitry, including an instrumentation amplifier that buffers the voltage across $R_F$, contributes additional latency. The latency from ADC to DAC, was apparent as a temporary plateau at the leading edge of the commanded voltage. This discontinuity was measured using oscilloscope traces and captured by digitized waveforms. Some clock cycles are required before the filtered data is available and added to the command signal, and thus an observable flat region on the rising edge of the square wave is inevitably produced prior to adding the series resistance compensation from the transient current spike onto the command signal. Fig. 9 shows an oscilloscope capture of the end-to-end latency with vertical markers measuring a latency of $\sim$2.5 $\mu$s. Recorded DAC waveforms suggest a similar result with a plateau of $\sim$5 sampling clocks, which at the sampling period of 400 ns, corresponds to $\sim 2\,\mu$s of end-to-end latency.

## V. CONCLUSIONS

An open-source data acquisition system that supports real-time feedback using FPGA-based DSP has been designed and presented. The system is designed to be easily translated to a broad range of experimental needs by being modular and open source. The off-the-shelf FPGA module provides an API for USB data transfers between a host computer and the FPGA. The design files for the custom DAQ board that connects to the FPGA module are maintained on GitHub[32] and were created using KiCad, an open source printed circuit board design suite. The supporting

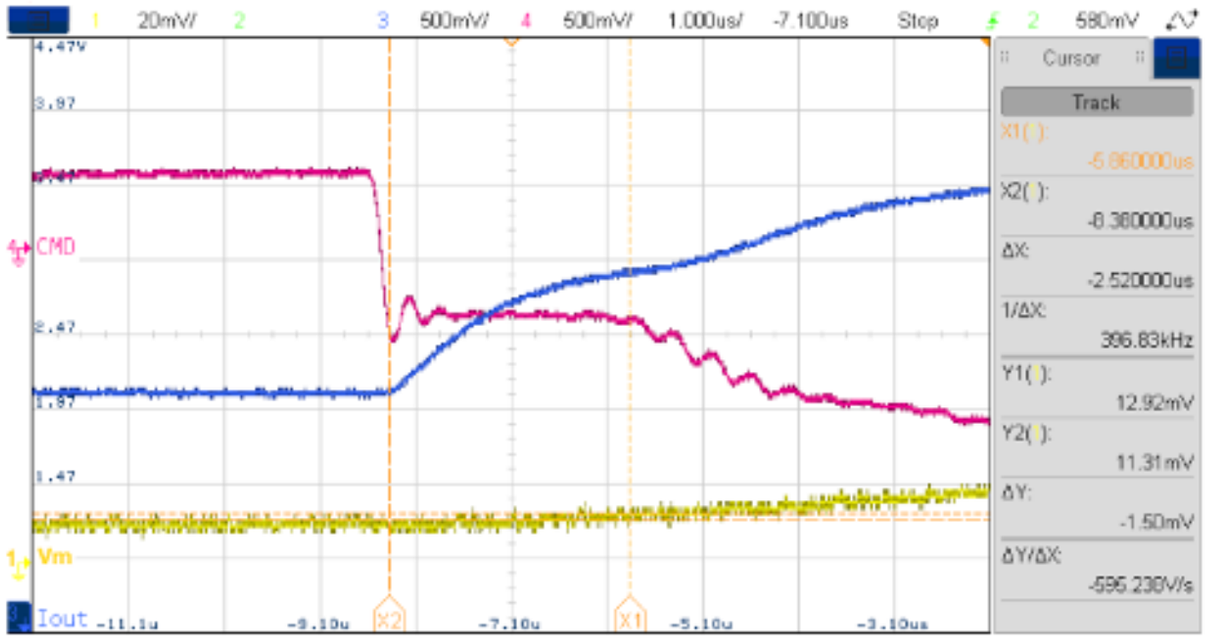



FIG. 9. End-to-end filter datapath latency measured by oscilloscope. The magenta trace represents the $V_{CMD}$ signal sent through the filter. The initial edge transition of the signal represents the raw $V_{CMD}$ signal before digital feedback has had time to take effect. As the $V_{CMD}$ signal moves through the analog circuitry, it then creates a spike in $I_M$ measured current, which is scaled and summed back with the signal, at which point, the flat region after the initial edge transition begins to ramp up again. It follows that from this flat region, end-to-end latency of the filter data path can be accurately measured.

FPGA code[33] and Python software[34] are also open source. Contributions, extensions, and customization of the entire system from users are anticipated and will be supported. The DAQ system has six SPI interfaces driven by data that is filtered within the FPGA to control fast analog output DACs while four channels of analog input for dedicated feedback are available at 16-bit resolution and 5 MSPS. We have designed and evaluated efficient digital IIR filters that were demonstrated in a real-time feedback application targeting electrophysiology. The numerous general purpose analog inputs and outputs (I/O), as well as I2C busses and digital I/O create a system that supports *in situ* calibrations for improved control system performance. The calibration capability was experimentally demonstrated via an impedance analyzer. This DAQ system meets experimental needs for microsecond latency feedback.

## ACKNOWLEDGMENTS

This work is partially funded by NIH R15 grant R15NS116907 to PI L. J. Koerner. Additional funding from the University of St. Thomas (UST) Undergraduate Research Opportunities (UROP) program and the UST School of Engineering. L.J.K. thanks Dr. Gilman Toombes for inspiration and unwavering support.

## AUTHOR DECLARATIONS

### Conflict of Interest

The authors have no conflicts to disclose.

### Author Contributions

**Ian Delgadillo Bonequi**: Methodology, Software, Validation, Formal Analysis, Writing - Original & Draft. **Abraham Stroschein**: Methodology, Software, Validation, Writing - Review & Editing. **Lucas Koerner**: Conceptualization, Methodology, Software, Validation, Formal Analysis, Writing - Original & Draft, Supervision, Project administration, Funding acquisition.

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in figshare at `http://doi.org/10.6084/m9.figshare.20521944.v1`, reference number[35].

## Appendix A: Daughtercard pin description

| Pin # | Pin Name | Description |
|---|---|---|
| 1 | fast DAC 2 / GP DAC 2 | Analog input, $\sim 1\,\mu$s settling time, programmable full-scale range. Or analog input from DAC8050, 16 bit, $5\,\mu$s settle, 5 V full-scale. |
| 2,5,8,11,17 | Gnd | |
| 3 | GP DAC 1 | Analog input from DAC8050, 16 bit, $5\,\mu$s settle, 5 V full-scale. |
| 4 | fast ADC+ | Differential analog output (positive) to AD796x ADC. |
| 6 | fast ADC- | Differential analog output (negative) to AD796x ADC. |
| 7 | fast DAC 1 | Analog input, $\sim 1\,\mu$s settling time, programmable full-scale range. |
| 9 | GP ADC 1 | Analog output to ADS8686 1 MSPS (programmable PGA for full-scale range up to $\pm 10$ V). |
| 10 | +15 V | Power from linear regulator. |
| 12 | VCM | Reference voltage from the AD796x buffered on the DAQ board (2.5 V). Sets the common-mode voltage of differential ADC driver. |
| 13 | $-15$ V | Power from linear regulator |
| 14 | auxiliary | Resistor jumper on the DAQ board sets analog in, analog out, or digital I/O (3.3 V levels). |
| 15 | SCL | I2C clock. Pull-up resistor on DAQ board, 3.3 V level. |
| 16 | SDA | I2C data. Pull-up resistor on DAQ board, 3.3 V level. |
| 18 | 5 V | Power from linear regulator. |
| 19 | GP ADC 2 | Analog output to ADS8686 1 MSPS (programmable PGA for full-scale range up to $\pm 10$ V). |

TABLE III. Daughtercard pin map used for the HDMI-A connector. Output indicates from the daugthercard to the DAQ board. GP indicates general purpose.

### Appendix B: Post-Filter Data Processing Blocks

Additional processing blocks act on the filter output data. The signed, 14-bit filter output is processed with optional operations, which include a scaling and offset, down-sampling operation, and summing with a separate 2.5 MSPS signal. After scaling, the filter output is converted to an unsigned value with offset to match the unsigned fast DACs. To use minimal resources and maximize clock frequency the signed to unsigned conversion was hand-coded as a bit-shift operation

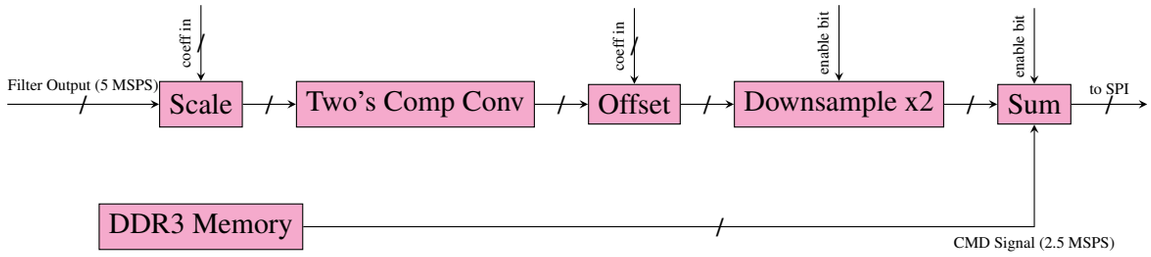



FIG. 10. Architecture of the post-filter scaling and summing blocks.

combined with addition of an offset. A summary description of the filter output to DAC datapath is listed below. Fig. 10 shows a block diagram of this post-filter datapath.

1. Two 14 bit coefficients are loaded using the Opal Kelly Register Bridge. One of the coefficients is as an optional offset and the other is a scaling factor with 13 bit fraction length.

2. The 14 bit filter output which varies from -1 to 1 is multiplied by the unsigned scaling coefficient (between 0 and 1). Thus, the filter data (14-bit, 12-bit fraction) remains between -1 to 1 after scaling with maximum and minimum outputs of 0x1000 and 0x3000, respectively.

3. The sign bit of the output is checked and then the data is bit-shifted left by one position. If the sign bit is set, 0x2000 is subtracted from the output to the right of the sign bit, if the sign bit is not set, 0x1FFF is added to the output. Then, the first 14 bits of the result are taken as the output. Using these rules, maximum and minimum outputs of the filter become 0x3FFF and 0x0000, respectively, which fit the maximum and minimum output hexadecimal codes of the 14-bit fast DAC.

4. After the filter data is scaled and converted to unsigned binary representation, the aforementioned 14-bit offset coefficient is added.

5. Data is optionally downsampled by a factor of 2 and optionally summed with a signal stored in the DDR3 memory and read out at 2.5 MSPS.

## REFERENCES

[1]S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2019).

[2]K. J. Åström and B. Wittenmark, *Adaptive Control: Second Edition* (Courier Corporation, 2013).

[3]S. J. Yu, E. Fajeau, L. Q. Liu, D. J. Jones, and K. W. Madison, "The performance and limitations of FPGA-based digital servos for atomic, molecular, and optical physics experiments," Review of Scientific Instruments **89**, 025107 (2018), publisher: American Institute of Physics.

[4]D. R. Leibrandt and J. Heidecker, "An open source digital servo for atomic, molecular, and optical physics experiments," Review of Scientific Instruments **86**, 123115 (2015), publisher: American Institute of Physics.

[5]B. M. Sparkes, H. M. Chrzanowski, D. P. Parrain, B. C. Buchler, P. K. Lam, and T. Symul, "A scalable, self-analyzing digital locking system for use on quantum optics experiments," Review of Scientific Instruments **82**, 075113 (2011).

[6]Y. A. Patel, A. George, A. D. Dorval, J. A. White, D. J. Christini, and R. J. Butera, "Hard real-time closed-loop electrophysiology with the Real-Time eXperiment Interface (RTXI)," PLoS Computational Biology **13**, e1005430 (2017).

[7]M. F. Bolus, A. A. Willats, C. J. Whitmire, C. J. Rozell, and G. B. Stanley, "Design strategies for dynamic closed-loop optogenetic neurocontrol *in vivo*," Journal of Neural Engineering **15**, 026011 (2018).

[8]M. F. Bolus, A. A. Willats, C. J. Rozell, and G. B. Stanley, "State-space optimal feedback control of optogenetically driven neural activity," Journal of Neural Engineering (2020), 10.1088/1741-2552/abb89c.

[9]J. Rolston, R. Gross, and S. Potter, "Closed-loop, open-source electrophysiology," Frontiers in Neuroscience **4** (2010), 10.3389/fnins.2010.00031.

[10]J. D. Rolston, R. E. Gross, and S. M. Potter, "NeuroRighter: Closed-loop multielectrode stimulation and recording for freely moving animals and cell cultures," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2009) pp. 6489–6492, iSSN: 1558-4615.

[11]S. Buccelli, J. Tessadori, I. Colombi, L. Martines, M. Bisio, A. Averna, M. Semprini, V. Pasquale, and C. Michela, "Closed-loop electrophysiology: Past, present and future perspectives and applications," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (2018) pp. 1–5.

[12]A. Pirog, Y. Bornat, R. Perrier, M. Raoux, M. Jaffredo, A. Quotb, J. Lang, N. Lewis, and S. Renaud, "Multimed: An Integrated, Multi-Application Platform for the Real-Time Recording and Sub-Millisecond Processing of Biosignals," Sensors **18**, 2099 (2018).

[13]J. Park, G. Kim, and S.-D. Jung, "A 128-Channel FPGA-Based Real-Time Spike-Sorting Bidirectional Closed-Loop Neural Interface System," IEEE Transactions on Neural Systems and Rehabilitation Engineering **25**, 2227–2238 (2017).

[14]A. A. Sharp, M. B. O'Neil, L. F. Abbott, and E. Marder, "The dynamic clamp: artificial conductances in biological neurons," Trends in Neurosciences **16**, 389–394 (1993).

[15]A. Wallach, D. Eytan, A. Gal, C. Zrenner, and S. Marom, "Neuronal response clamp," Frontiers in Neuroengineering **4** (2011), 10.3389/fneng.2011.00003.

[16]D. Linaro, J. Couto, and M. Giugliano, "Real-time Electrophysiology: Using Closed-loop Protocols to Probe Neuronal Dynamics and Beyond," Journal of Visualized Experiments: JoVE , 52320 (2015).

[17]D. Linaro, J. Couto, and M. Giugliano, "Command-line cellular electrophysiology for conventional and real-time closed-loop experiments," Journal of Neuroscience Methods **230**, 5–19 (2014).

[18]W. Liu, S. Chang, J. Wang, and C. Liu, "A real-time hardware experiment platform for closed-loop electrophysiology," IEEE Transactions on Neural Systems and Rehabilitation Engineering **30**, 380–389 (2022).

[19]P. Weerakoon, E. Culurciello, K. G. Klemic, and F. J. Sigworth, "An Integrated Patch-Clamp Potentiostat With Electrode Compensation," IEEE Transactions on Biomedical Circuits and Systems **3**, 117–125 (2009).

[20]J. Dolzer, "Patch Clamp Technology in the Twenty-First Century," (Springer US, New York, NY, 2021) pp. 21–49.

[21]Analog Devices, "Data Sheet AD7961 16-bit 5 MSPS PULSAR Differential ADC," (2014), accessed 2022-08-15.

[22]Texas Instruments, "AN-1515 A Comprehensive Study of the Howland Current Pump," Tech. Rep. AN-1515 (Texas Instruments, 2013) accessed 2022-07-27.

[23]Opal Kelly, "XEM7310," (2022), https://opalkelly.com/products/xem7310/, Accessed 2/6/2022.

[24]Opal Kelly, "FrontPanel HDL, Opal Kelly Documentation Portal," (2022), https://docs.opalkelly.com/fpsdk/frontpanel-hdl/, Accessed 8/15/2022.

[25]Analog Devices, "AD7961 Native FMC Card & Xilinx Reference Design," (2021), accessed 2022/8/15 https://wiki.analog.com/resources/fpga/xilinx/fmc/ad7961.

[26]S. Srot, "SPI controller core," (2016), https://opencores.org/projects/spi, Accessed: 2022-02-5.

[27]D. Gisselquist, "Debug Bus Interface(s)," (2021), accessed: 8/15/2022

https://github.com/ZipCPU/dbgbus/blob/master/hexbus/rtl/hbexec.v.

[28]M. Francis, "Infinite Impulse Response Filter Structures in Xilinx FPGAs (WP330 v1.2)," (2009), Xilinx White Paper WP330.

[29]A. Pantazis and R. Olcese, "Cut-Open Oocyte Voltage-Clamp Technique," in *Encyclopedia of Biophysics*, edited by European Biophysical Societies, G. Roberts, and A. Watts (Springer Berlin Heidelberg, Berlin, Heidelberg, 2019) pp. 1–9.

[30]A. J. Sherman, A. Shrier, and E. Cooper, "Series Resistance Compensation for Whole-Cell Patch-Clamp Studies Using a Membrane State Estimator," Biophysical Journal **77**, 2590–2601 (1999).

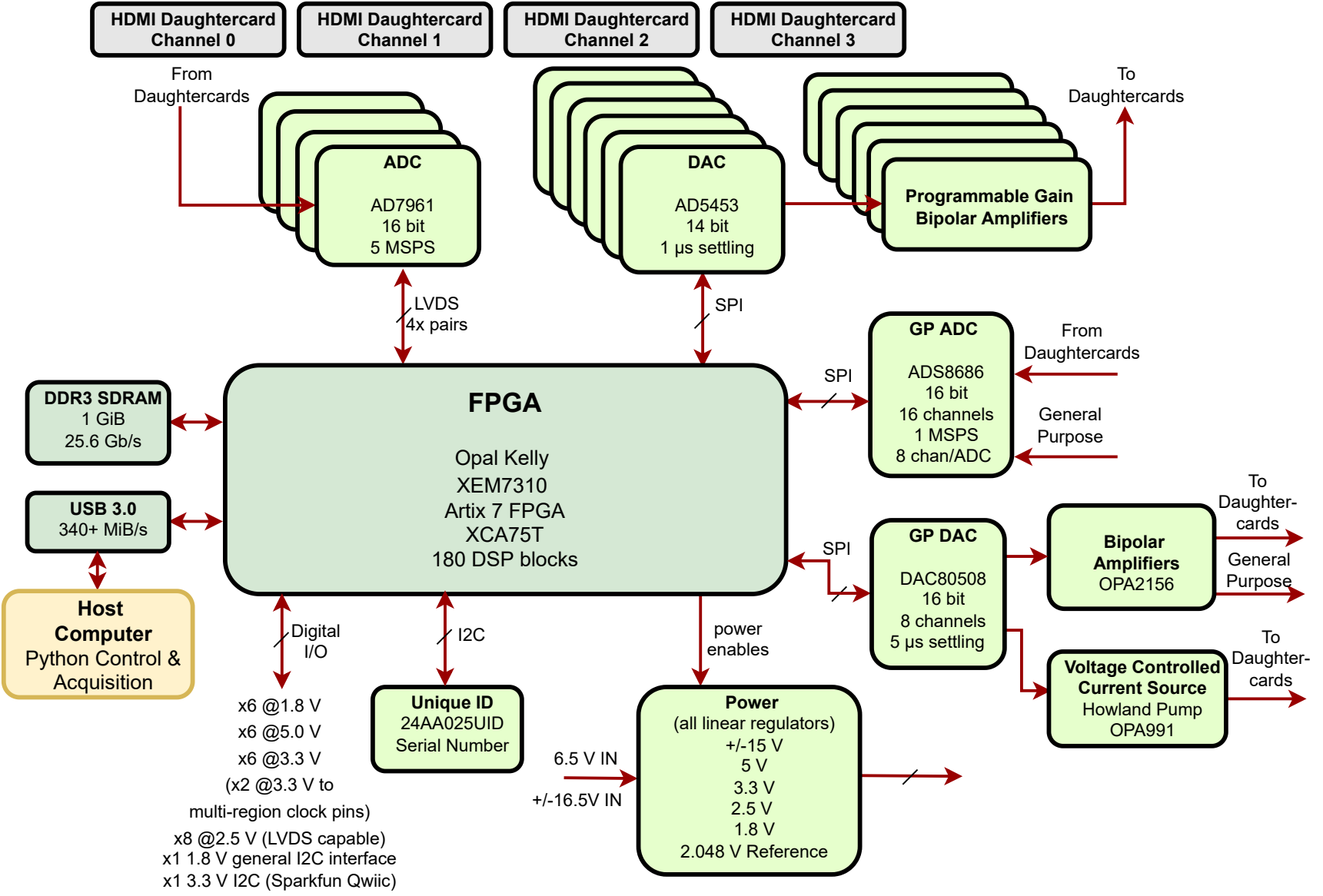
[31]F. Sigworth, "Design of the EPC-9, a computer-controlled patch-clamp amplifier. 1. Hardware," Journal of Neuroscience Methods **56**, 195–202 (1995).

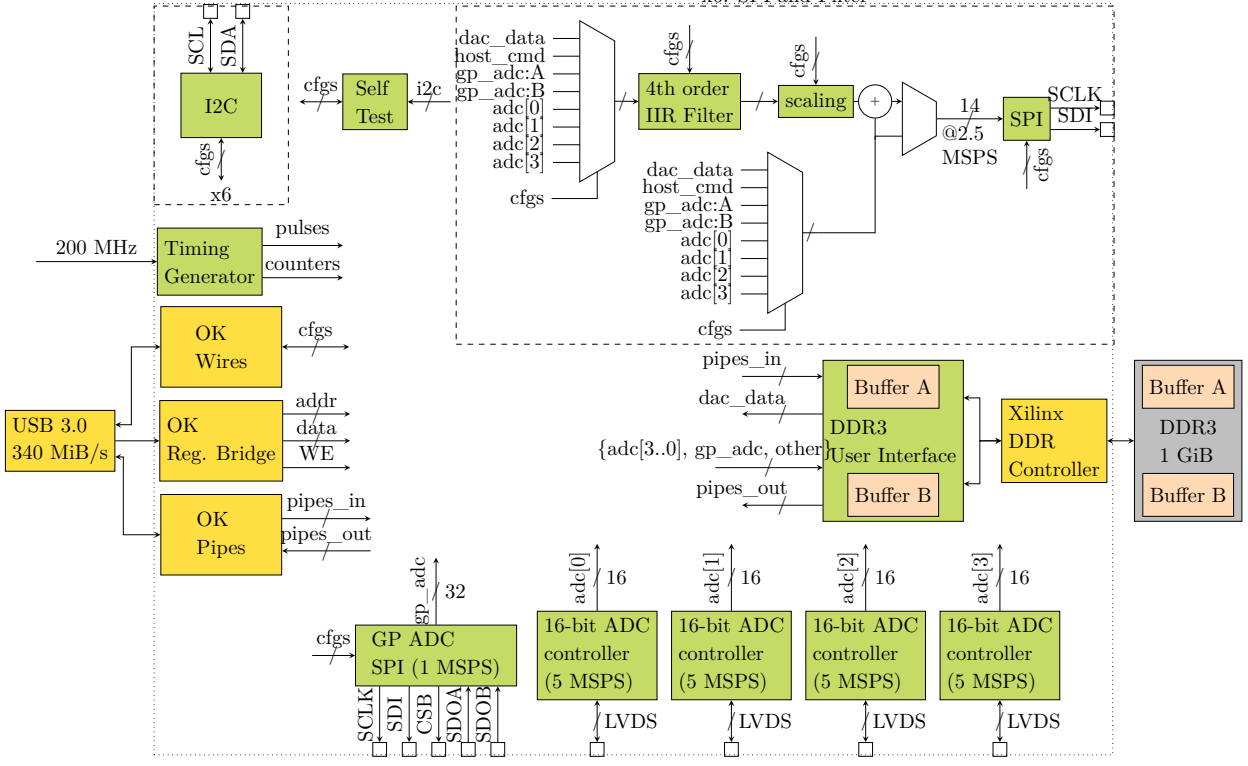
[32]L. Koerner, "Real-Time Feedback DAQ Board," (2022), https://github.com/lucask07/open_covg_daq_pcb, Accessed 2022/8/16.

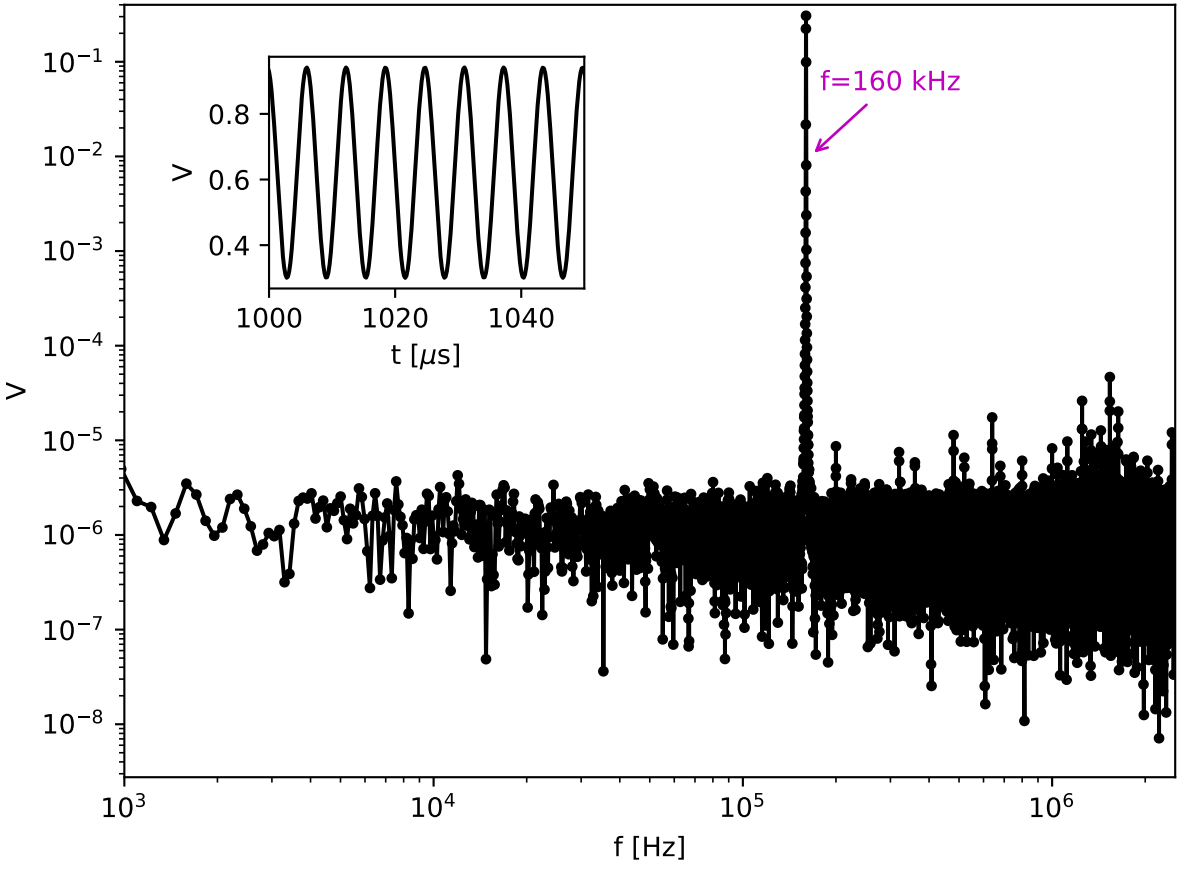
[33]L. Koerner, "FPGA and Python experiment code for the digital ion channel amplifier project." (2022), https://github.com/lucask07/covg_fpga, Accessed 2022/8/16.

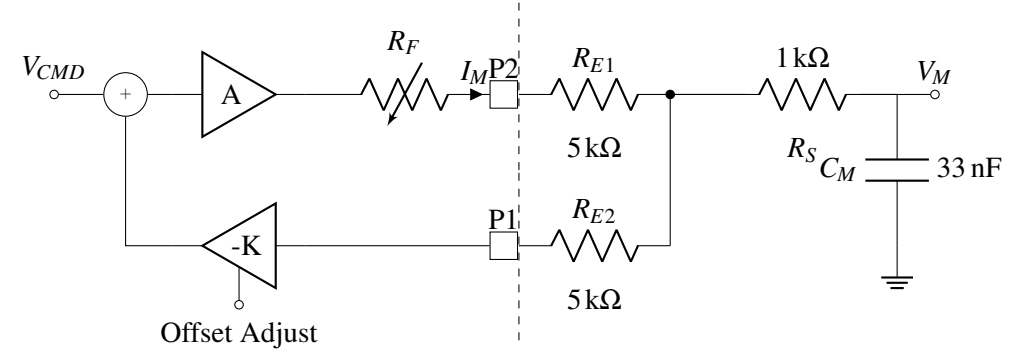
[34]A. Stroschein, "Pyripherals," (2022), https://github.com/Ajstros/pyripherals, Accessed 2022/8/16.

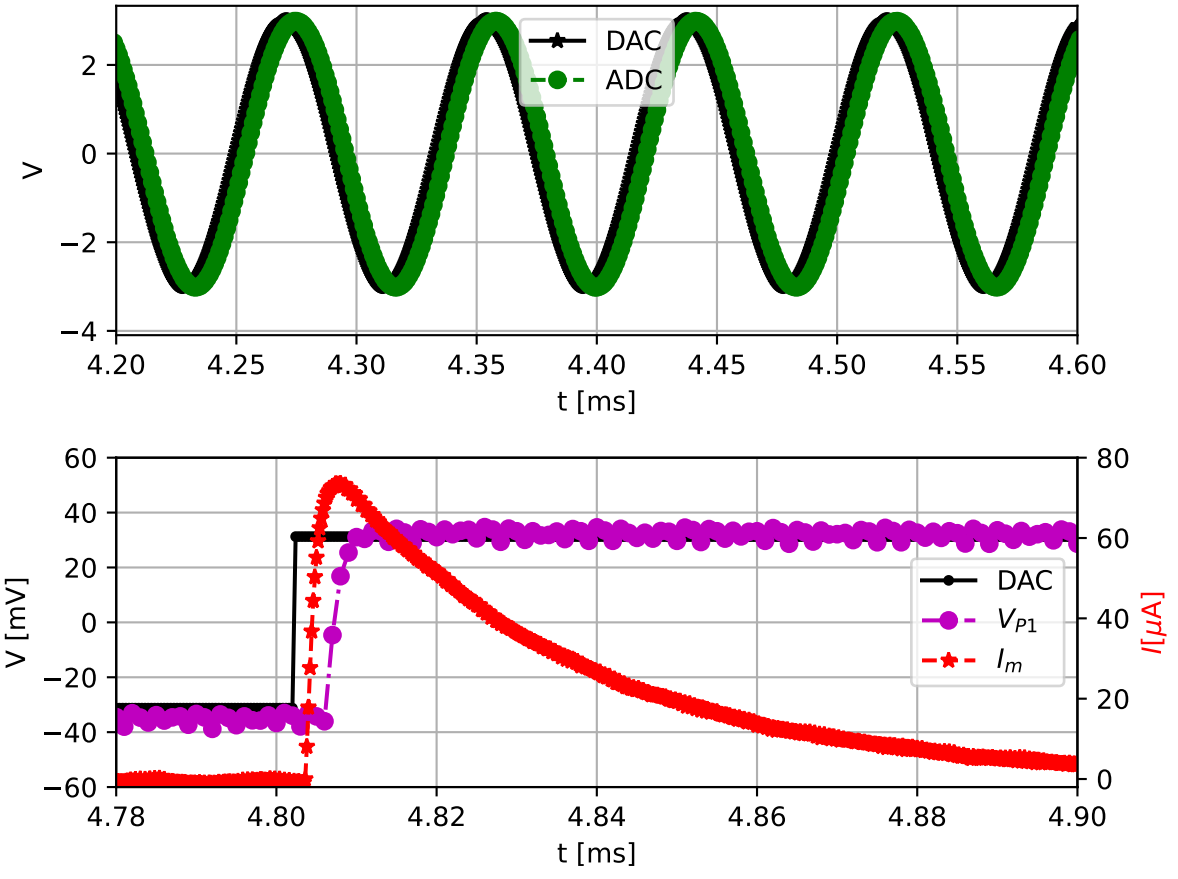
[35]L. Koerner, A. Stroschein, and I. Delgadillo Bonequi, "FPGA-based closed-loop data acquisition system demonstration data," (2022), 10.6084/m9.figshare.20521944.v1, https://figshare.com/articles/dataset/FPGA-based_closed-loop_data_acquisition_system_demonstration_data/20521944/1 Accessed 2022/8/20.

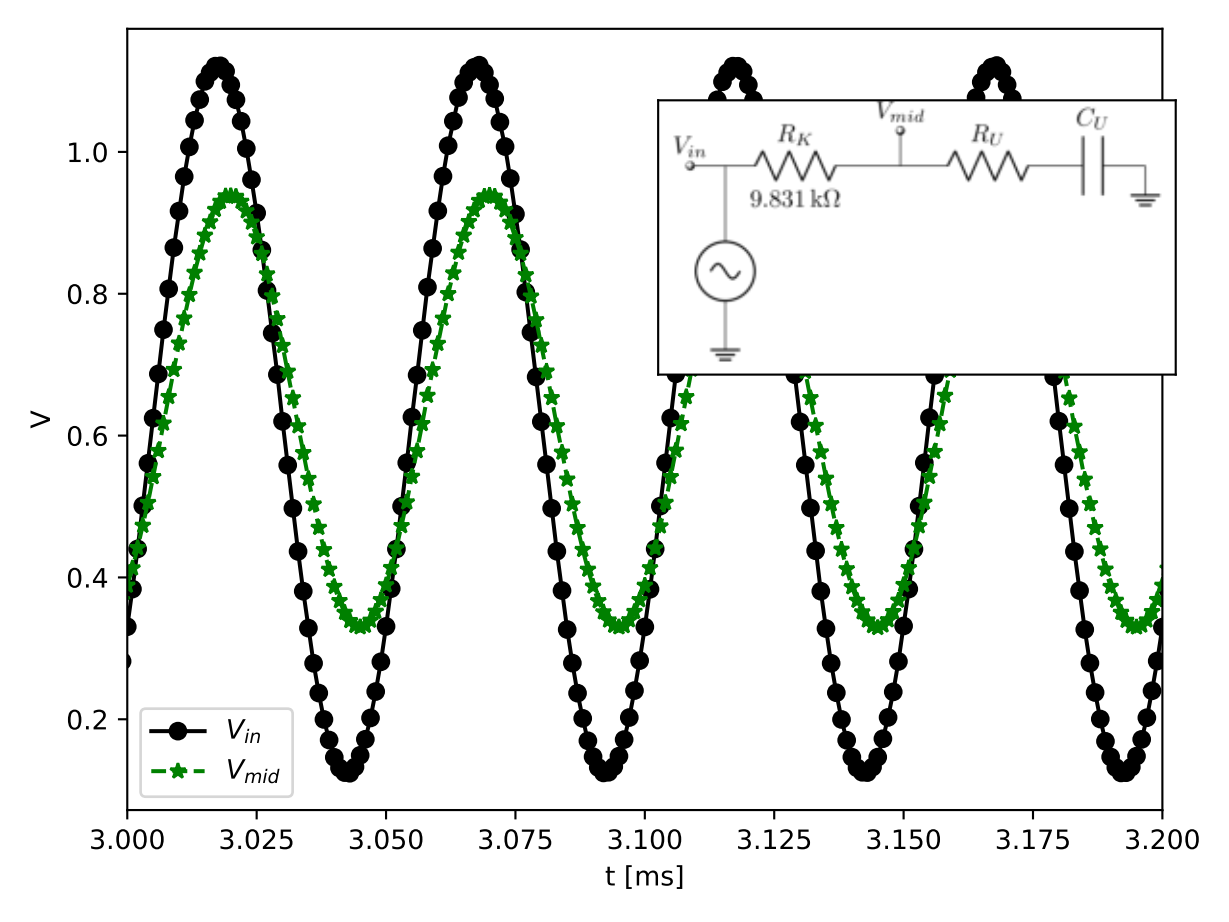

HDMI Daughtercard Channel 0
HDMI Daughtercard Channel 1
HDMI Daughtercard Channel 2
HDMI Daughtercard Channel 3
From Daughtercards
ADC
AD7961
16 bit
5 MSPS
DAC
AD5453
14 bit
1 µs settling
Programmable Gain Bipolar Amplifiers
To Daughtercards
LVDS
4x pairs
SPI
GP ADC
ADS8686
16 bit
16 channels
1 MSPS
8 chan/ADC
From Daughtercards
General Purpose
DDR3 SDRAM
1 GiB
25.6 Gb/s
FPGA
Opal Kelly
XEM7310
Artix 7 FPGA
XCA75T
180 DSP blocks
USB 3.0
340+ MiB/s
GP DAC
DAC80508
16 bit
8 channels
5 µs settling
Bipolar Amplifiers
OPA2156
To Daughter-cards
General Purpose
Host Computer
Python Control & Acquisition
Digital I/O
I2C
power enables
Voltage Controlled Current Source
Howland Pump
OPA991
To Daughter-cards
x6 @1.8 V
x6 @5.0 V
x6 @3.3 V
(x2 @3.3 V to multi-region clock pins)
x8 @2.5 V (LVDS capable)
x1 1.8 V general I2C interface
x1 3.3 V I2C (Sparkfun Qwiic)
Unique ID
24AA025UID
Serial Number
6.5 V IN
+/-16.5V IN
Power
(all linear regulators)
+/-15 V
5 V
3.3 V
2.5 V
1.8 V
2.048 V Reference

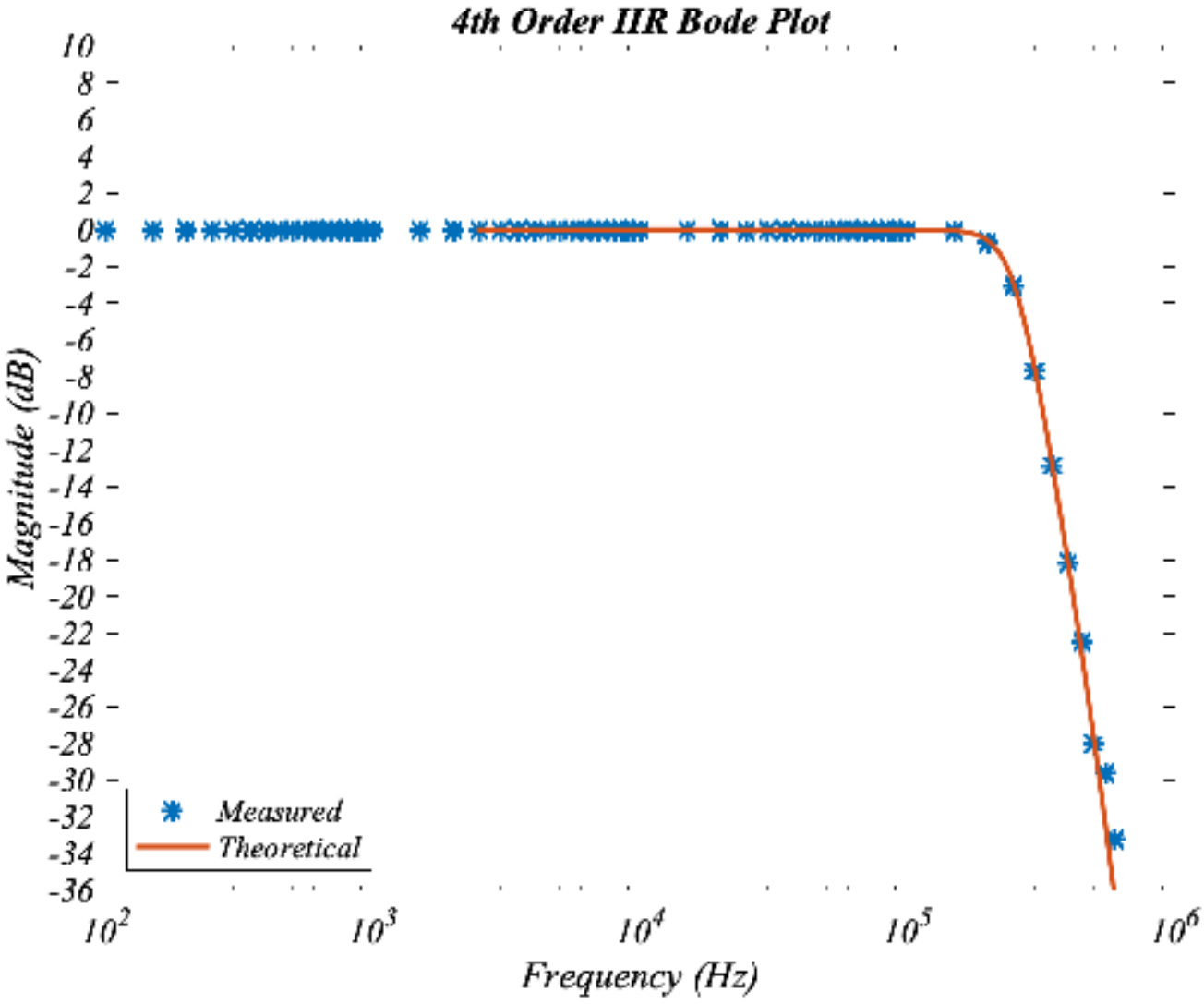

x6: SPI and Filter
SCL
SDA
I2C
cfgs
x6
cfgs
Self Test
i2c
dac_data
host_cmd
gp_adc:A
gp_adc:B
adc[0]
adc[1]
adc[2]
adc[3]
cfgs
4th order IIR Filter
cfgs
scaling
+
14
@2.5 MSPS
SPI
SCLK
SDI
cfgs
dac_data
host_cmd
gp_adc:A
gp_adc:B
adc[0]
adc[1]
adc[2]
adc[3]
cfgs
200 MHz
Timing Generator
pulses
counters
OK Wires
cfgs
USB 3.0 340 MiB/s
OK Reg. Bridge
addr
data
WE
OK Pipes
pipes_in
pipes_out
pipes_in
dac_data
{adc[3..0], gp_adc, other}
pipes_out
Buffer A
DDR3 User Interface
Buffer B
Xilinx DDR Controller
Buffer A
DDR3 1 GiB
Buffer B
gp_adc
32
cfgs
GP ADC SPI (1 MSPS)
SCLK
SDI
CSB
SDOA
SDOB
adc[0]
16
16-bit ADC controller (5 MSPS)
LVDS
adc[1]
16
16-bit ADC controller (5 MSPS)
LVDS
adc[2]
16
16-bit ADC controller (5 MSPS)
LVDS
adc[3]
16
16-bit ADC controller (5 MSPS)
LVDS

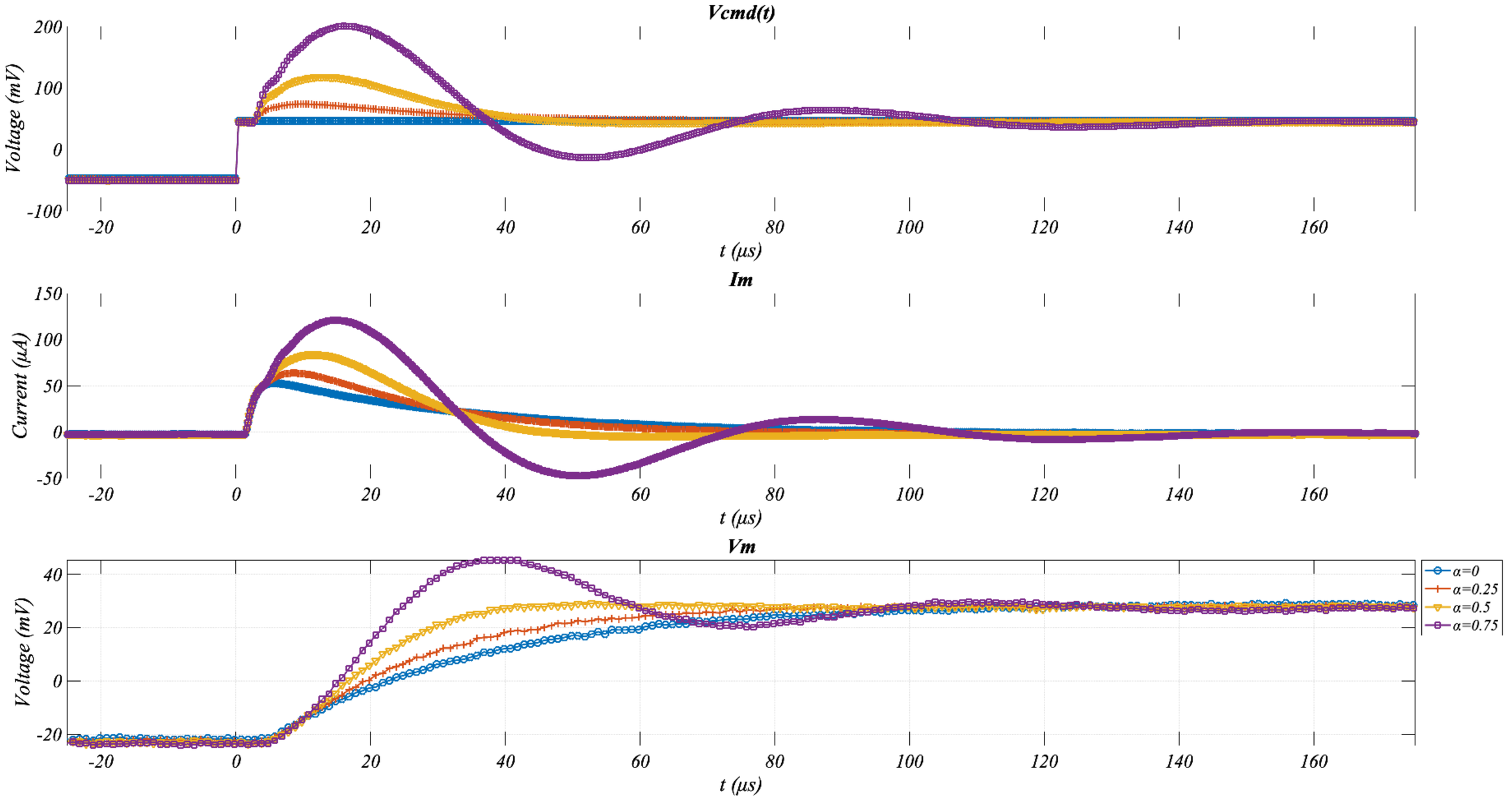

f=160 kHz
V
f [Hz]
t [$\mu$s]
V

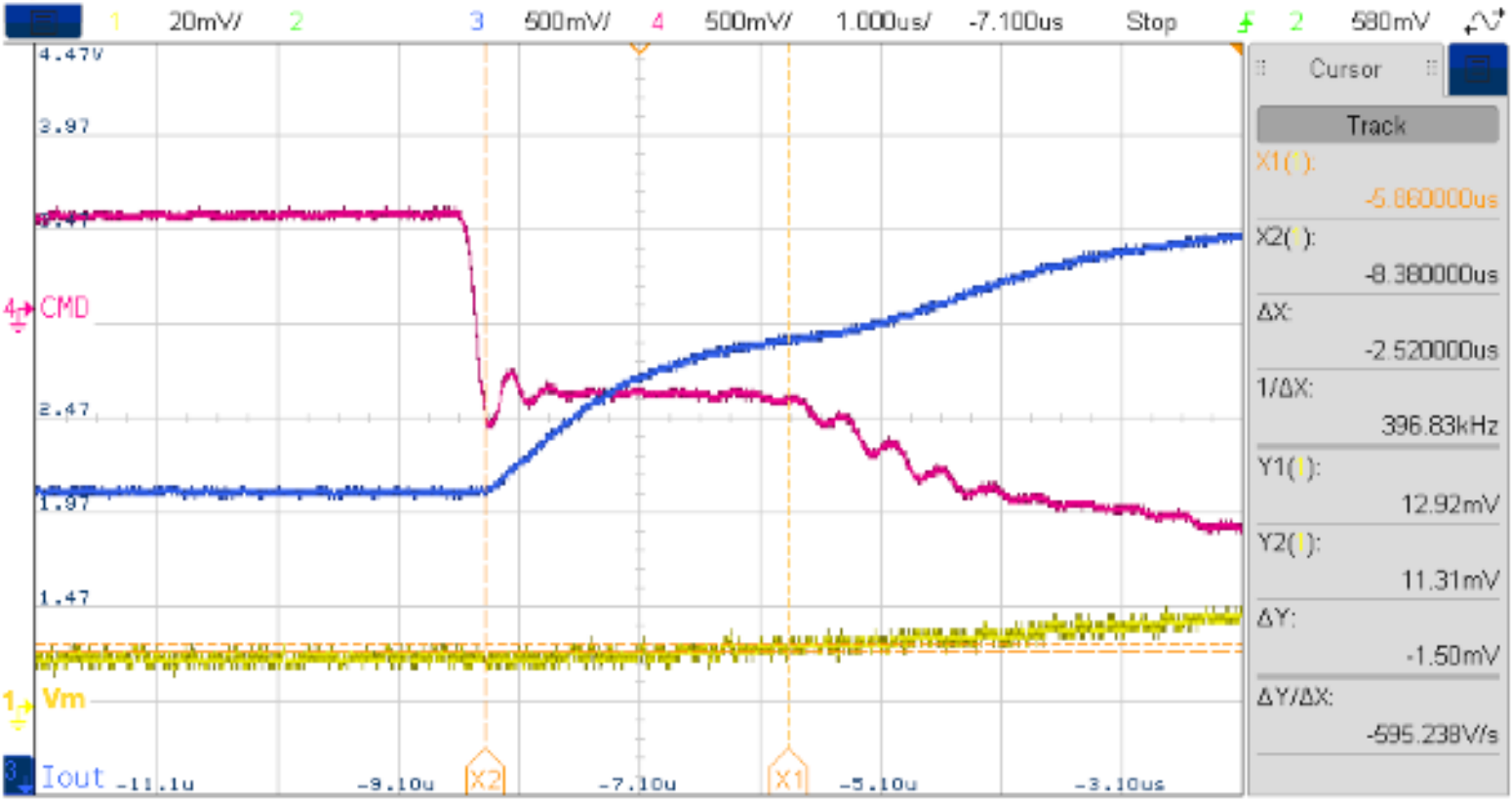

$V_{CMD}$
+
A
$R_F$
$I_M$
P2
$R_{E1}$
5 kΩ
1 kΩ
$V_M$
$R_S$
$C_M$
33 nF
-K
P1
$R_{E2}$
5 kΩ
Offset Adjust

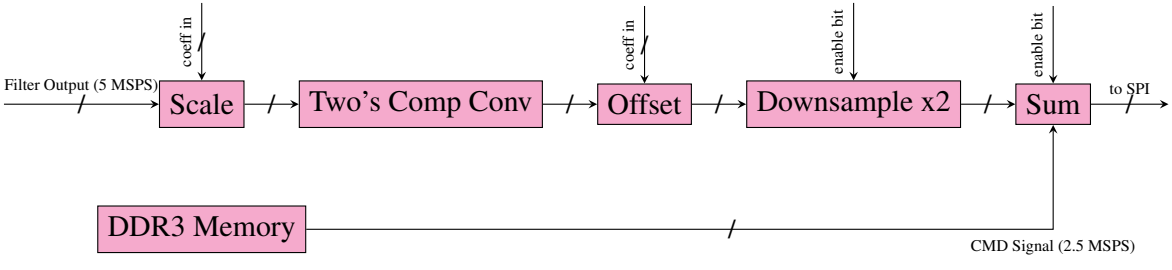

DAC
ADC
V
t [ms]
V [mV]
$I[\mu A]$
$V_{P1}$
$I_m$


1.0
0.8
0.6
0.4
0.2
V
3.000
3.025
3.050
3.075
3.100
3.125
3.150
3.175
3.200
t [ms]
$V_{in}$
$V_{mid}$
$V_{in}$
$R_K$
9.831 kΩ
$V_{mid}$
$R_U$
$C_U$


4th Order IIR Bode Plot
Magnitude (dB)
Frequency (Hz)
Measured
Theoretical
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
-24
-26
-28
-30
-32
-34
-36
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$


Vcmd(t)
Voltage (mV)
200
100
0
-100
-20
0
20
40
60
80
100
120
140
160
t (μs)
Im
Current (μA)
150
100
50
0
-50
-20
0
20
40
60
80
100
120
140
160
t (μs)
Vm
Voltage (mV)
40
20
0
-20
-20
0
20
40
60
80
100
120
140
160
t (μs)
α=0
α=0.25
α=0.5
α=0.75


1 20mV/ 2 3 500mV/ 4 500mV/ 1.000us/ -7.100us Stop 2 580mV
Cursor
Track
X1( ):
-5.000000us
X2( ):
-8.380000us
ΔX:
-2.520000us
1/ΔX:
396.83kHz
Y1( ):
12.92mV
Y2( ):
11.31mV
ΔY:
-1.50mV
ΔY/ΔX:
-595.238V/s
4.47V
3.97
2.47
1.97
1.47
CMD
Vm
Iout
-11.1u
-9.10u
X2
-7.10u
X1
-5.10u
-3.10us


coeff in
coeff in
enable bit
enable bit
Filter Output (5 MSPS)
Scale
Two's Comp Conv
Offset
Downsample x2
Sum
to SPI
DDR3 Memory
CMD Signal (2.5 MSPS)