THESIS

APPLICATIONS OF DIGITAL ADAPTIVE FILTERS TO TIME-RESOLVED OPTICAL MICROSCOPY

Submitted by

Saurabh Gupta

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2020

ABSTRACT

APPLICATIONS OF DIGITAL ADAPTIVE FILTERS TO TIME-RESOLVED OPTICAL MICROSCOPY

Phosphorescence lifetime imaging is used on several fronts, such as, skin cancer or melanoma diagnosis, and estimation of tissue oxygenation among others. Oxygen profiling is critical for mapping brain activity, apart from its use to monitor several metabolic activities, and often employs oxygen tagging molecules/probes. In this work, we describe a novel technique to recover phosphorescence lifetime using a real-time digital adaptive filter running on a field-programmable gate array (FPGA) and conclude with an important takeaway.

We also describe our strategy to mitigate relative intensity noise (RIN) in ultrafast fiber lasers, which are an attractive alternative to bulk lasers for non-linear optical microscopy due to their compactness and low cost. The high RIN of these lasers poses a challenge for pump-probe measurements such as transient absorption and stimulated Raman scattering, along with modalities that provide label-free contrast from the vibrational and electronic structure of molecules. Our real-time approach for RIN suppression uses a digital adaptive noise canceller implemented on a FPGA. We demonstrate its application to transient absorption spectroscopy and microscopy and show compatibility with a commercial lock-in amplifier. Lastly, we report the noise estimates specific to our current experimental setup.

ACKNOWLEDGEMENTS

Many people have played a crucial role in this work from the beginning to the end and I would like to extend my sincere and heartfelt gratitude through this opportunity. First and foremost, I would like to thank my advisor Jesse Wilson, for infinite help and support in the lab, for his patience, guidance and belief in me. Thanks to my peers Erkang Wang, Arya Mugdha, and Patrick Stockton for hands-on assistance with optics, my friend Hieu Bui for helping with sample prep, my mentor William Hudson for his advice and support on a large number of topics, and my graduate committee members for their time and cooperation. Next, thanks to my amazing collaborators: Steven Derrien, Jason Killgore, Randy Bartels, Neil Adames, Kevin Lear and Erin Flater. Thanks are due for Wilson and Bartels lab members for their general help around the lab. And at last but not the least, thanks to my well-wishers, friends, and family for their love and support in making this work possible.

TABLE OF CONTENTS

CHAPTER 1 – PHOSPHORESCENCE LIFETIME IMAGING

Introduction

Fluorescence/phosphorescence is a phenomenon wherein a material absorbs incident radiation and emits back light either instantly or after a short time duration. It continues to do so at a rate which reduces with time and is called the lifetime decay. Different materials/pigments have an associated characteristic lifetime which distinguishes them from one another. We can use this information to identify different types of pigments such as, a clump of melanin from skin. It can be an effective way of diagnosing melanoma or skin cancer by observing the phosphorescence lifetime of malignant melanocytes.

Optical luminescence and diffuse scattering lifetimes are conventionally measured in either the time domain or frequency domain. Time-domain techniques apply a narrow time, broad frequency excitation, while frequency-domain techniques apply a broad time, narrow frequency excitation. In principle these methods are equivalent, being related through the Fourier transform [1]. But in practice, there are numerous trade-offs between time and frequency-domain in terms of speed, signal-to-noise ratio (SNR), characterization of multi-exponential and non-exponential responses, and separability of similar lifetimes [2], [3].

Recently, a hybrid technique has emerged, which applies a pseudo-random excitation sequence that is broad in both, time and frequency[4]–[6], and has the potential to combine advantages from both approaches (Fig. 1.1). This has been implemented by storing a time series of the emission and correlating it with the excitation sequence to recover the optical impulse response [4], [5]. A faster implementation uses a high-resolution timing module to time-tag each photon arrival [7], but none of these pseudo-random approaches are fast enough for typical for laser-scan imaging framerates. This speed limit can be overcome by implementing the correlator with real time digital signal processing on a field-programmable gate array (FPGA) [8], and we note that FPGAs have also been used to improve frequency-domain lifetime imaging far beyond what is possible with analog approaches [9]. In this paper, instead of using a correlator, we recover the lifetime response of the sample with an adaptive filter [10] running on an FPGA. The adaptive filter models the optical response in real time, adjusting the model until its output matches the measured optical response. DAFT-LSM is distinct from the digital frequency domain FPGA solution in the use of pseudo-random illumination and operation of the PMT in the analog integration (not photon counting) mode. Generation of a pseudo-random binary sequence (PRBS) using a linear feedback shift register (LFSR) is shown in fig. 1.2. Here, we characterize the instrument response of the adaptive filter with a short lifetime fluorophore, AFC (7-Amino-4-trifluoromethyl coumarin), measure the phosphorescence lifetime decay of an oxygen sensing probe, $Ru(dpp)_3(PF_6)_2$, and demonstrate lifetime imaging of $Ru(BPY)_3$ crystals.

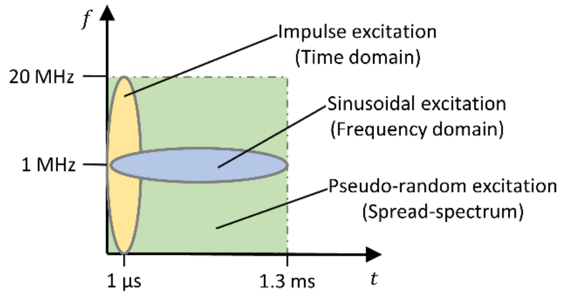


Fig. 1.1. Time domain methods use a stimulus which is narrow in time but contains a broad range of frequencies. Frequency domain methods use a stimulus with a narrow frequency spectrum but is broad in time. Pseudo-random excitation sequences are broad in both, time and frequency. Represented numbers are for illustration purpose only.

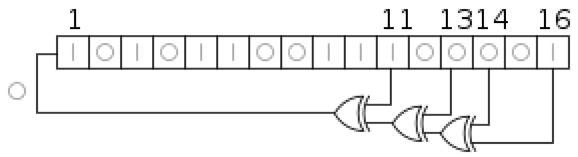


Fig. 1.2. A 16-bit Fibonacci LFSR for PRBS generation. Source: Wikipedia

Theory

In DAFT-LSM, an adaptive filter searches for an impulse response function, $h[n]$, that transforms the pseudo-random sequence, $x[n]$, through a convolution, $y = x \star h$, to match the measured output of the optical system, $d[n]$. Our implementation follows the standard least mean squares (LMS) algorithm [11] as shown in fig. 1.3. After convergence, the filter coefficients are a direct time-domain representation of the impulse response of the phosphorescent material at the focal spot of the laser beam.

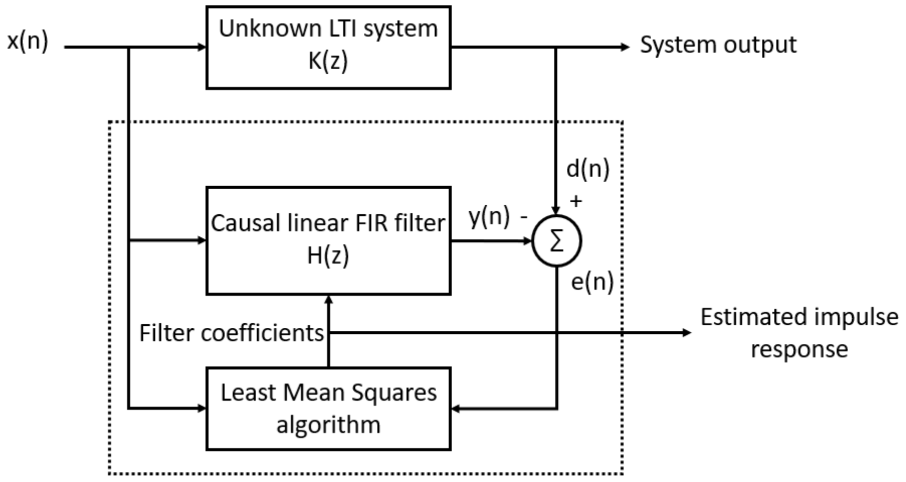


Fig. 1.3. Block diagram of the Adaptive Filter

The derivation of update factor by steepest descent is as follows:

Let the coefficients be defined by, $f(n) = 1, 2, \ .. \ , \ M-1$

Then the output of the filter is given by the convolution, $y \ = \ x \ * \ f = \sum_{k=0}^{M-1} f(k).x(n-k)$

If the desired signal is represented by d, then the error is given by, $e = \ d - y$

Let the error performance index $J(n)$ be defined by: $J(n) = e^2(n)$

It is described by a quadratic surface and has a single minimum

To minimize $J(n)$ using the steepest-descent method we move by $\frac{\partial J(n)}{\partial f(n)}$ every iteration

Which is given by,

$$\frac{\partial J(n)}{\partial f(n)} = \frac{\partial e^2(n)}{\partial f(n)}$$

$$= 2.e(n).\frac{\partial e(n)}{\partial f(n)}$$

$$= 2.e(n).\frac{\partial (d(n) - \sum_{k=0}^{M-1} f(k).x(n-k)\ )}{\partial f(n)}$$

$$= -2.e(n).x(n-k)$$

Thus, the coefficient update is given by:

$$f(n+1) = f(n) + \frac{\partial J(n)}{\partial f(n)} = f(n) - 2.x(n-k).e(n).\mu$$

Where μ is the learning rate or the convergence factor.

Implementation and Experimental Setup

The Verilog code used for our adaptive filter is based on a textbook 2-tap 8-bit coefficient design[12]. This design was modified and extended to accommodate 16-bit coefficients and 16 taps. All arithmetic is performed in signed fixed-point format, and bit truncation was replaced with signed rounding operations to achieve stable convergence behavior[13]. Source code is attached for reference in Appendix A and the bit-widths of signals are mentioned in table 1.4. We implemented this design on a DE2-115 development board (Terasic) with a Cyclone IV FPGA with a compatible AD/DA Data Conversion Card (Terasic). The design utilizes only 6% of the logic elements and 18% of the DSP multipliers on the FPGA and can be clocked at a maximum frequency of 26.95 MHz as reported by the timing analyzer.

Table 1.4. Signals and their bit-widths in $Qm.n$ notation where m is the total number of bits including the sign bit and n is the number of fractional bits.

Signal name	Format
x	Q12.11
f	Q16.10
y	Q32.21
d	Q14.11
e	Q33.21
emu	Q33.21
$xemu$	Q45.32

A system diagram of the experimental setup is shown in Fig. 1.5. A 405nm laser diode (Thorlabs LP405-SF10 on CLD1011LP mount) was modulated using a pseudo-random binary sequence generated by the FPGA. The modulated laser beam was focused on the specimen, which was placed on top of a Nikon Eclipse Ti-S microscope base, through a combination of mirrors, X-Y galvanometers (Cambridge Technology), scan lens, and tube lens. A coverslip directed a small portion of the modulated beam onto a photodetector (Thorlabs PDA36A2), which served as an input, $x[n]$, to the adaptive filter. The adaptive filter is linear shift-invariant and cannot model non-linearities and hysteresis in the laser diode. Hence, resampling the modulated beam was necessary. The emitted light from the specimen was collected by a microscope objective lens (Nikon fluor 10x/0.3 NA), and routed to a photomultiplier tube (PMT) by a dichroic mirror (Edmund 450 nm shortpass) and a long-pass filter (Chroma AT465LP) followed by a focusing lens (f =2.5 cm), and filtered with an iris (~2 mm diameter) for confocal detection. The current generated by the PMT (Hamamatsu R928) was amplified by a trans-impedance amplifier (Thorlabs TIA60) and fed to the Analog-to-Digital Converter (ADC) connected to the FPGA board. For each experiment, the PMT gain and optionally RF attenuators were used to scale the input signal amplitude to match the 512 mV peak-to-peak ADC range.

The adaptive filter implementation was first validated on an internal finite impulse response (FIR) filter to confirm convergence and find a reasonable feedback gain, μ. This test-bench FIR was programmed with an impulse response given by: $\{400, 300, 200, 100, 400, 300, 200, 100, 400, 300, 200, 100, 400, 300, 200, 100\}$. Feedback coefficients μ in decreasing powers of 2 given by, $\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$, and so on were tested. Adaptive filter performance for each case was monitored by recording the evolution of the filter impulse response after resetting its coefficients to zero. For $\mu \geq \frac{1}{8}$, the feedback gain was too large, and the filter was unstable. For $\mu \leq \frac{1}{8192}$, the correction factor, $\mu\ x[n]e[n]$, rounded to zero under our finite-precision implementation, and the filter failed to adapt at all. For $\frac{1}{16} \leq\ \mu \leq \frac{1}{4096}$, convergence behavior of the first adaptive filter coefficient is shown in Fig. 1.6, as it adapted to the first test-bench FIR coefficient, which was set to 400. We found that on setting the feedback gain, $\mu = \frac{1}{16}$, the filter converged in ~250 samples or ~12.5 μs at 20 MHz sample rate (50 ns/sample). All subsequent experiments were performed with $\mu = \frac{1}{16}$.

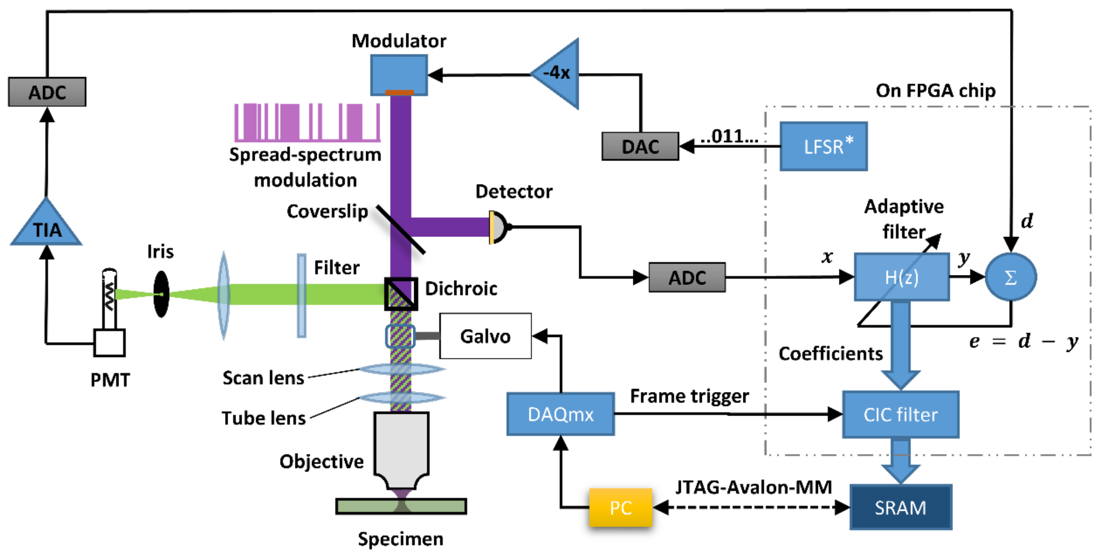


Fig. 1.5. Adaptive filter confocal imaging system. A Linear-Feedback Shift Register * (LFSR) on the FPGA generates a pseudo-random binary sequence, which modulates a laser and is focused on a specimen. The measured phosphorescence response is digitized using an ADC and serves as desired signal, d, for the adaptive filter. The modulated excitation beam is sampled by a second ADC channel and is used as the input, x, for the adaptive filter. The adaptive filter updates its impulse response to match its output, y, to the measured phosphorescence, d. After convergence, its coefficients, h, represent the impulse response of the specimen. The coefficients are downsampled to the pixel clock using a CIC filter and stored in SRAM. NI DAQmx is used to control the galvanometers and send a frame trigger to the FPGA. Final image stack $(n_x \times n_y \times 16)$ stored on the SRAM is transferred to the PC using a JTAG-Avalon-MM interface.

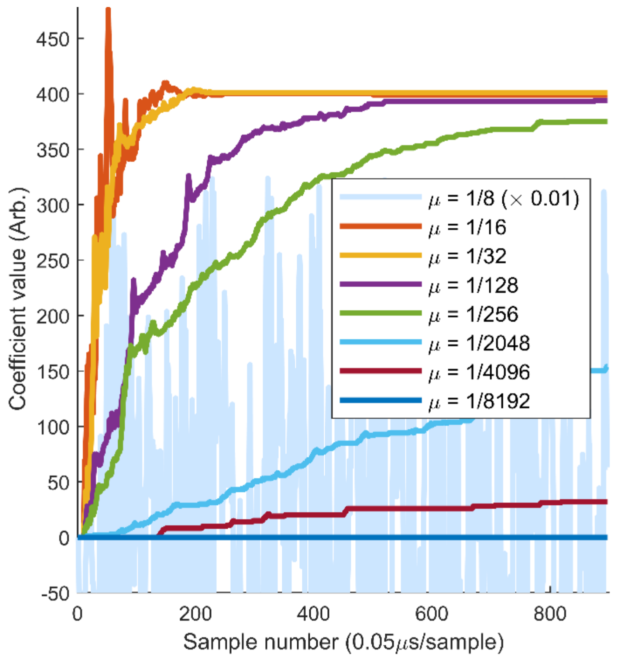


Fig. 1.6. Effect of feedback gain μ on adaptive filter convergence.

To validate the adaptive filter on luminescence lifetime measurement, we prepared a solution of a short lifetime fluorophore (109 µM AFC in DMSO, fig. 1.7), and in contrast, a relatively long lifetime dye (180 µM $Ru(dpp)_3(PF_6)_2$ in ethylene glycol, fig. 1.8)[14]. A few drops of each solution were placed in silicone wells on two separate glass slides and cover slipped. After bringing the sample into focus with 1.63 mW of modulated laser illumination, we adjusted the PMT gain and used appropriate attenuators to not exceed the range of the ADC and ran the ADC and adaptive filter at 2 MHz sample rate (0.5 µs/sample). The laser was parked at the same spot during the whole measurement. We acquired 1024 samples of the signals x, d, y, e, and h, using Quartus Signal-tap II utility. These coefficients were then averaged and plotted using MATLAB.

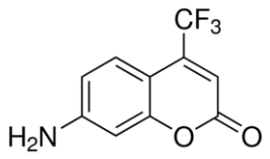


Fig. 1.7. Molecular structure of AFC from Sigma-Aldrich.

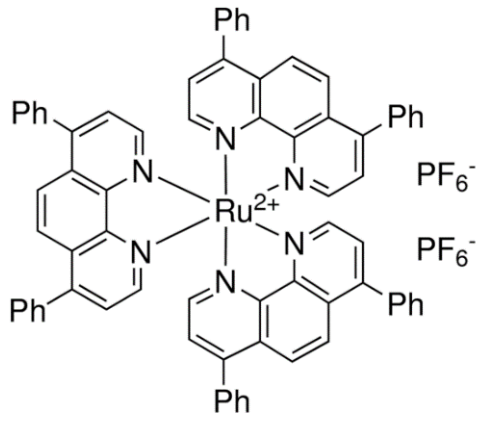


Fig. 1.8. Molecular structure of $Ru(dpp)_3(PF_6)_2$ from Sigma-Aldrich.

Results

Figure 1.10 depicts $x, d, y\ and\ f$ for AFC. Due to a short lifetime of AFC ~5 ns[15], the emitted fluorescence closely follows the modulation (Fig. 1.10b). The adaptive filter closely mimics this response (Fig. 1.10c) by adjusting its coefficients to values shown in Fig. 1.10d. As expected for a short lifetime fluorophore, the coefficients represent a delta function.

Similarly, the time domain signals for $Ru(dpp)_3(PF_6)_2$ are shown in Fig. 1.11. In this case, the long lifetime of the dye acts as a low-pass filter on the excitation sequence, as can be seen from Fig. 1.11b, appearing smoother than in Fig. 1.11a. The adaptive filter output closely matches the phosphorescence emission from the sample (Fig. 1.11c) and corresponding impulse response, h, is shown in Fig. 1.11d. An exponential resembling decay trend can be seen in the coefficients with $\tau = 2.1$ μs, in reasonable agreement with the published data[14]. Note the negative baseline in h, which is a consequence of the DC-blocking transformer on the ADC front end shown in the schematic in fig. 1.9.

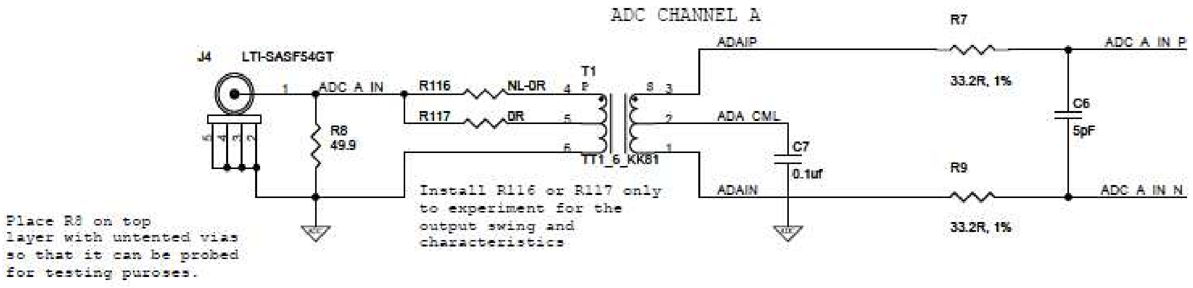


Fig. 1.9. Schematic of the ADC front-end from Terasic.

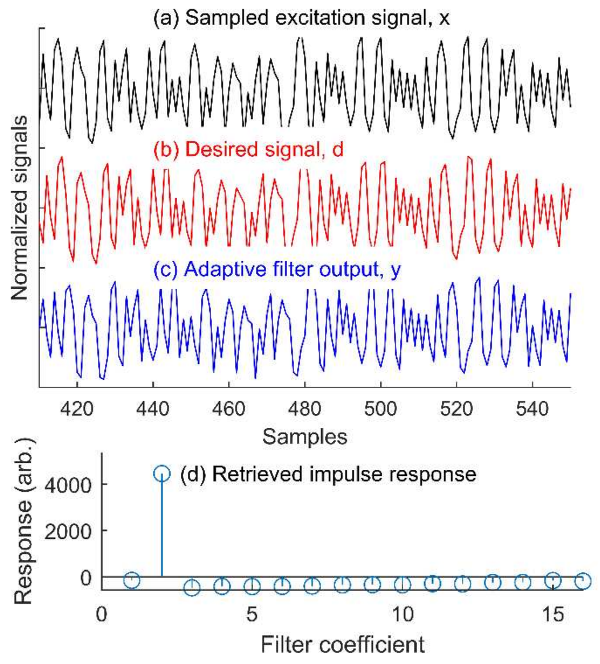


Fig. 1.10. Adaptive filter tracking and recovery of impulse response of AFC. (a) Sampled excitation signal. (b) Desired signal/phosphorescence. (c) Adaptive filter output mimics (b). (d) Retrieved impulse response resembles a delta function.

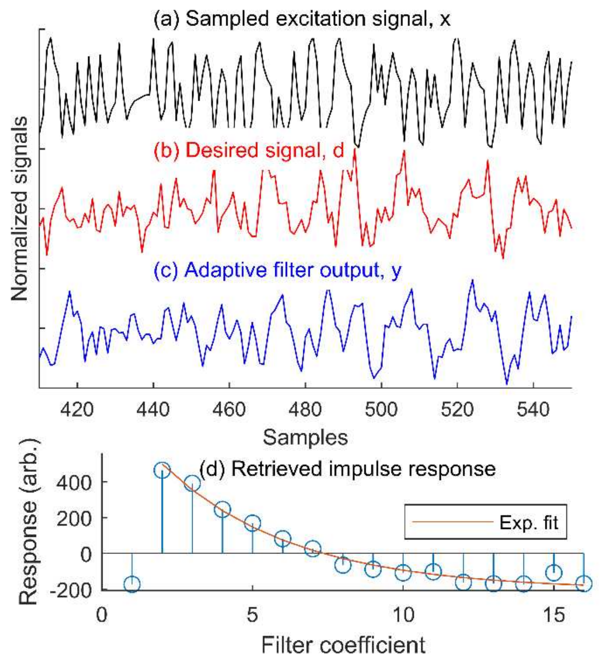


Fig. 1.11. Adaptive filter tracking and recovery of impulse response of $Ru(dpp)_3(PF_6)_2$. (a) Sampled excitation signal. (b) Desired signal/phosphorescence. (c) Adaptive filter output mimics (b). (d) Retrieved impulse response and coefficients fit to a mono-exponential function with an estimated $\tau = 2.1\ \mu s$.

For an imaging test case, we scattered Ru(BPY)$_3$ crystals (fig. 1.12) on a slide followed by a coverslip and performed imaging, while the ADC and adaptive filter operated at a sample rate of 20 MHz (50 ns/sample). Average laser power incident on the sample was 245 µW with a pseudo-random modulation. The coefficients were averaged at every pixel using a Cascaded Integrator-Comb (Intel CIC filter IP core) by setting the averaging factor to be a multiple of number of channels i.e. 992. This gave us a pixel dwell time 49.6 µs. The scan line-rate for a 256 x 256 image was set using ScanImage (Vidrio Technologies), such that one set of averaged coefficients are produced per pixel by the CIC filter. A start-of-frame trigger was exported out through the NI BNC-2090A board and fed to the FPGA through GPIO pins which signaled an internal state machine to start recording coefficients. The impulse response recovered at each pixel was sequentially logged to an on-board SRAM while a pair of galvanometers raster-scanned the excitation spot through the specimen. After a frame was recorded, the SRAM contents were transferred to a host computer through a JTAG-AvalonMM interface[16], and reshaped into an image stack of dimensions $n_x \times n_y \times 16$. This stack then went through two rounds of fitting: first, by being fitted to a mono-exponential model, $A + Be^{-\frac{(t-C)}{\tau}}$. Then for a second round, by choosing the obtained fit parameters from first round of the pixel with least sum-squared error within a 3 x 3 neighborhood of the pixel of interest, as the initial guess. A resulting mono-exponential fit false color image of τ (emission lifetime) is shown in Fig. 1.13. The goodness-of-fit of the individual pixels within the image have been evaluated using the Coefficient of Determination (R^2). Pixels with $R^2 < 0.6$ have been masked in black to only display pixels with a reliable fit. The observed ~100—200 ns lifetime consistent with prior measurements of dried films of Ru(BPY)$_3$ [17]. The fact that the solid state lifetime is shorter than the ~350 ns lifetime in liquid solution may be a consequence of temperature-dependent quenching [14].

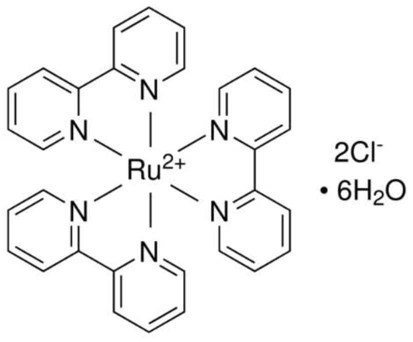


Fig. 1.12. Molecular structure of Ru(BPY)$_3$ from Sigma-Aldrich.

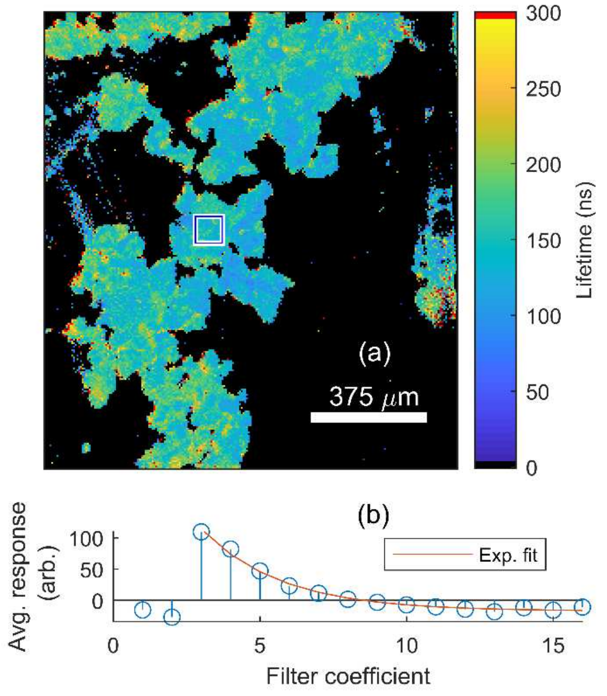


Fig. 1.13. (a) Mono-exponential fit false color phosphorescence lifetime image of Ru(BPY)$_3$ crystals. Pixels with goodness-of-fit parameter $R^2 < 0.6$ are masked black and pixels with lifetime $\geq$ 300 ns are marked in red. (b) Mono-exponential fit of averaged filter coefficients in a selected region with an estimated $\tau = 135\ ns$.

Discussion

We have shown adaptive filter measurement of luminescence lifetimes on order of 100 to 2,100 nanoseconds—enough for imaging oxygen-sensing phosphors. It does this by modeling the time-domain optical response to a pseudo-random excitation with a FIR filter, iteratively adjusting its impulse response until its output tracks the optical measurement. By implementing the adaptive filter on an FPGA, the feedback loop can operate on a 20 MS/sec data stream and converge well within the 50 µs pixel dwell time of a laser-scan imaging scenario with a 3.3 sec/frame rate. We anticipate this approach can be extended to lifetimes shorter than 10 ns by implementing a D-LMS adaptive filter, which uses pipelining to reduce the critical path delay in the feedback loop [12].

Key Takeaway

We used MATLAB to observe the system identification capability of the LMS algorithm on the unknown non-linear quantity i.e. phosphorescence, using the following photoexcitation rate equation [18],

$$\frac{dn}{dt} = k_x L(t)(1-n) - \left(k_0 + k_q Q\right)n$$

Where $n = N/N_T$ is the fraction of phosphor in excited form, k_x is the rate coefficient of excitation, $L(t)$ is the excitation light intensity, and k_0, k_q and Q are traditional Stern-Volmer parameters.

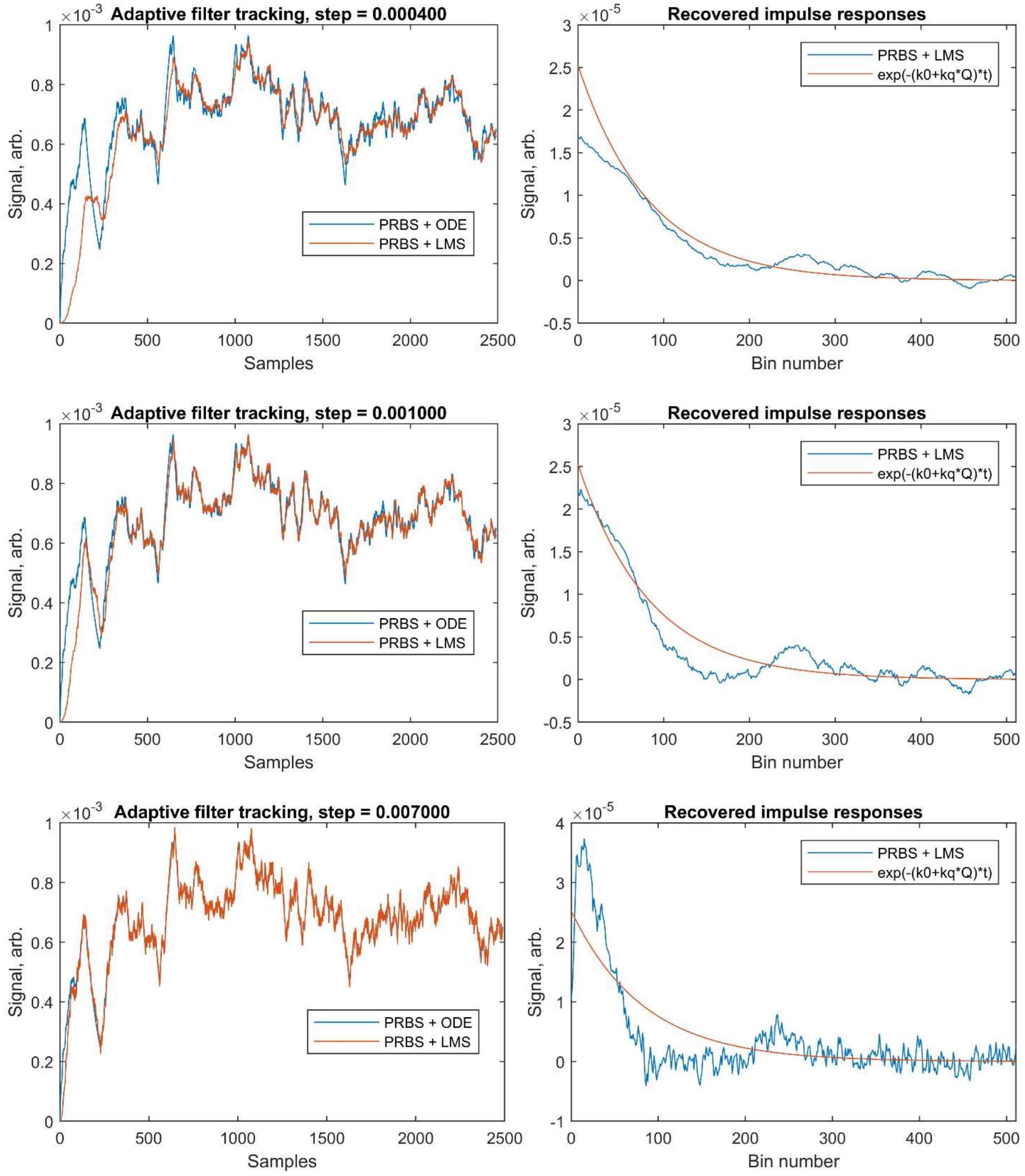


Fig. 1.14. Effect of changing step size on the recovered impulse response. Variability in the recovered impulse response shows that the method cannot recover the lifetime correctly.

Solution to the ordinary differential equation (ODE) using a PRBS stimulus showed that the adaptive filter tracked the time domain phosphorescence very well. The recovered impulse response though, was dependent on the step-size, implying that the method is inconsistent and cannot be relied upon (figure 1.14). In another experiment, we programmed a FIR filter to match the correct theoretical phosphorescence lifetime and observed the time domain response. It was found that the time domain response of the FIR filter was very different from that obtained by solving the ODE (figure 1.15). Thereby implying that a FIR system cannot model non-linear phenomena correctly. This study sets the stage for exploration of non-linear system identification methods in this domain.

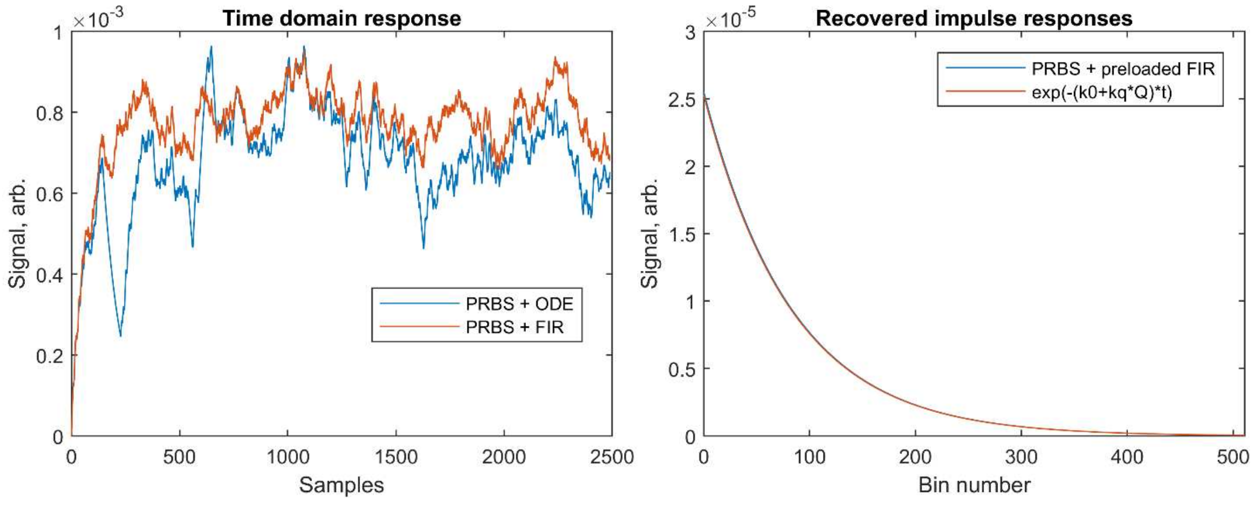


Fig. 1.15. Using a FIR filter to understand the time domain tracking behavior. On preloading FIR with the correct impulse response, its time domain response is very different from that obtained by solving the ordinary differential equation (ODE). Thereby indicating that non-linear phenomena cannot be modeled using FIR systems.

CHAPTER 2 – RINS: THE REAL-TIME RELATIVE INTENSITY NOISE SUPPRESSOR

Introduction

Transient absorption and stimulated Raman scattering microscopy are nonlinear optical methods that obtain chemical contrast from electronic and vibrational dynamics [19]. Applications have ranged from label-free molecular contrast in biological tissues to defect characterization and charge transport mapping in novel 2-dimensional nanomaterials [19]–[22]. Both techniques rely on sensing miniscule perturbations to a probe laser beam, induced by a pump beam, and are thus highly susceptible to laser relative intensity noise (RIN). In the case of bulk laser sources (e.g. a Ti:Al_2O_3 ultrafast oscillator), most of the RIN can be rejected by modulating the pump at > 1 MHz and employing lock-in detection on the probe [23]. In the case of fiber laser sources, however, broad bandwidth and high-frequency RIN [24]–[26] necessitates active noise cancellation through balanced detection, for example, custom radio frequency (RF) analog electronics involving a variable-gain amplifier and PID controller [27]. Compensating for such noise by slow scanning and long averaging at each pixel introduces other problems. This averaging strategy precludes high-speed imaging and is ineffective on $1/f$ noise [28]. In addition, the heat deposited by long pixel dwell times can damage the sample. Recently, we introduced a software-based scheme which uses a high-speed analog to digital converter (ADC) and adaptive filtering for active RIN cancellation [29]. Compared with the analog electronics solution, the software approach is convenient in terms of sharing, replication, and fine-tuning and requires neither custom analog circuits, nor a dedicated hardware lock-in amplifier (LIA). However, as it was limited by ADC noise and based on post-processing of acquired data, it was impractically slow for all but proof-of-concept imaging tests. These speed limitations can be overcome by implementing the adaptive filter on a field-programmable gate array (FPGA) for real-time noise cancellation [30]. FPGAs, due to their capacity for low latency and high bandwidth computations, are well-suited for real-time signal processing on data streams directly from high-bandwidth (i.e. 10-500 MSPS) ADCs, enabling for example, low cost and energy efficient software-defined radio [31] and lock-in amplifiers for scientific applications [8], [32]–[34].

(Note: the information presented here appears with permission from the author's and in accordance with the publisher's license, reference [29] *and* [35]*.)*

Here, we implement adaptive laser RIN suppressor (RINS) in real-time on a development board (Red Pitaya STEMlab 125-14) that includes high-speed ADC and digital-to-analog (DAC) converters alongside an FPGA with an on-board microprocessor (Xilinx Zynq 7010 SoC). The device is set up as a drop-in denoiser that is inserted between the photodetectors and a lock-in amplifier and coded using high-level synthesis (HLS). For this implementation, we performed lock-in detection in software and show compatibility with a commercial lock-in amplifier. We demonstrate its application to transient absorption microscopy of a crystalline powder. Compared to our previous all-software implementation [29], this FPGA implementation has a lower noise floor and provides real-time RIN cancellation, reducing the image acquisition time from ~6 hours down to ~35 minutes.

Theory

A balanced detector can eliminate noise that is correlated between the signal and reference arms (common-mode noise) such as laser RIN. When the electronic noise floor is below the shot noise floor, a balanced detector will reduce the overall noise floor to 3 dB above the shot noise floor [36]. This limit arises from the separate shot noise contributions of the two detectors. Unlike the RIN, these add together at the balanced detector's output because they are uncorrelated. Likewise, electronic noise (photodiode amplifier noise, ADC input noise) is also uncorrelated. Therefore, when the electronic noise floor is above the shot noise floor, a balanced detector will reduce the overall noise floor to 3 dB above the electronic noise floor. We take this theoretical limit into consideration when tuning the adaptive filter's performance.

For a given adaptive filter length L we select a step size μ such that the noise floor of $e = y - d$ is at 3 dB above the electronics noise floor (the theoretical limit for an electronic noise limited balanced detector), which we estimate from the power spectrum at $f > 15\ \mathrm{MHz}$, above the photodetector cutoff. This turns out to be $\mu \approx 0.8/L$ for our conditions from our software-based study. As can be seen in fig. 2.1 below, larger μ has an advantage in tracking transmitted probe intensity more closely, but also erodes the signal and increases high-frequency noise. This increased noise for large μ is also a symptom of the LMS gradient descent overshooting and oscillating about the optimum filter coefficients [11]. The best SNR enhancement appears to be at $L = 8$, $\mu = 0.1$. Filters with $L \geq 16$ (not shown) were found to be less stable under our conditions and fail to further enhance SNR.

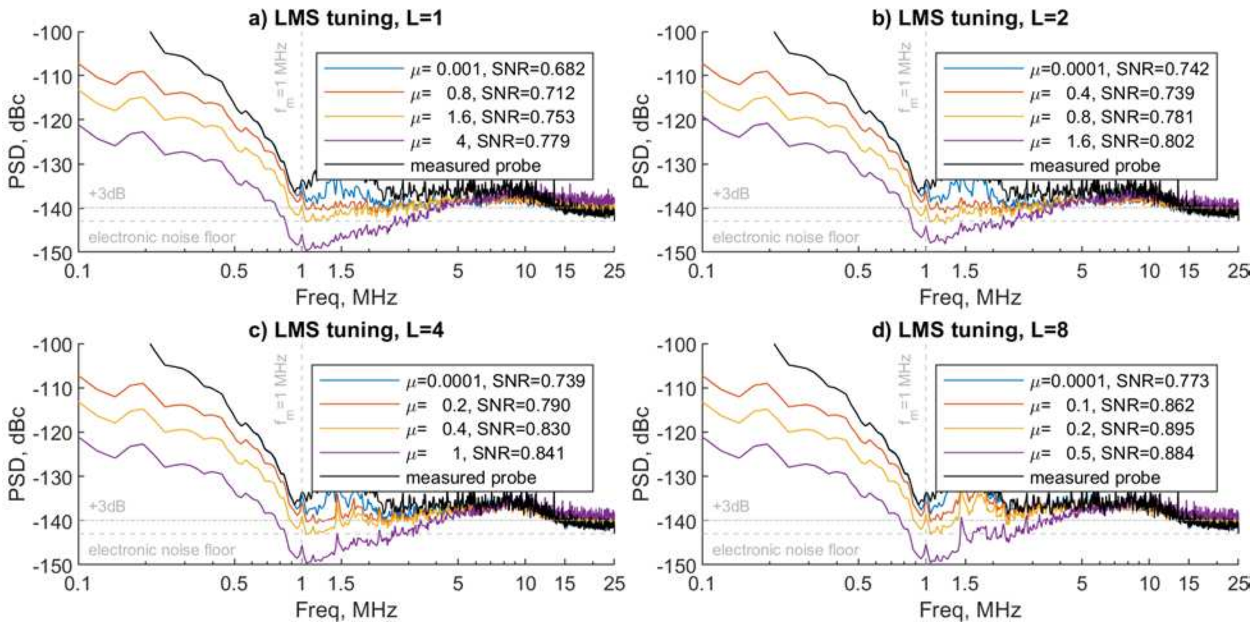


Fig. 2.1. ANC-enhanced PSDs for BGO particle imaging experiments. PSDs shown are max projections across all scan lines of the image, for a single repetition. The adaptive filter noise canceling performance is evaluated with respect to different filter lengths L and step sizes μ.

Implementation

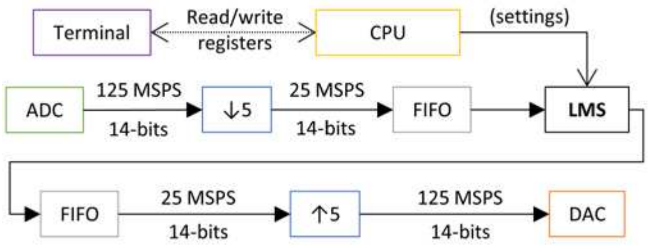


Fig. 2.2. System diagram of IP components and interconnects placed on the FPGA. Digitized datastream from ADC is downsampled, passed through an LMS adaptive filter, upsampled, then passed to the DAC for an analog output. Clock domain crossings are handled with FIFO buffers. The on-chip CPU controls parameters of the LMS module through an AXI-lite interface and can be accessed through a command-line terminal.

We made use of Pavel Demin's software-defined radio project as a starting point [37]. A high-level system diagram of our implementation is shown in fig. 2.2. An 8-tap digital adaptive noise canceller (ANC) was implemented using the least mean squares (LMS) algorithm. Individual modules were connected using an IP integrator (Xilinx Vivado Hlx 2018.2). All signals pass between modules using the AXI streaming protocol. Flow of data within the FPGA modules is as follows: Two 14-bit ADCs on-board operating at 125 MSPS digitize x and d which are low-pass filtered with cut-off at ~8.2 MHz then decimated

(Xilinx FIR compiler 2; M=5). The resulting 25 MSPS datastream is then passed to individual FIFOs for a clock domain conversion from 125 MHz to 25 MHz. The HLS produced adaptive filter IP (called LMS module) is clocked at 25 MHz, which receives this x and d. The IP produces an error output at 25 MSPS which is passed through a FIFO to convert clock domain from 25 MHz to 125 MHz. Clock domain crossing FIFOs are implemented using Xilinxs' FIFO generator (BRAM implementation with independent clocks using 8-sync stages). An interpolator (Xilinx FIR compiler 2; L=5) is used to upsample the error output before passing to a DAC on-board operating at 125 MSPS. A gain setting is implemented by right shifting a 14-bit window, effectively amplifying the error signal before outputting via a DAC. This serves the purpose of bringing the signal above the noise floors of the DAC and the next physical device downstream (i.e. a lock-in amplifier). The LMS algorithm step-size and output gain is manipulated using registers controlled by the CPU. A program running on the CPU in a loop continuously reads/updates these registers as provided by the user via a command-line terminal. The LMS module constantly reads these parameters every clock cycle through an AXI-lite interface.

The LMS module was coded in C/C++ using the HLS methodology, which speeds hardware design by making it easier to validate the algorithm, automatically handling pipelining and loop unrolling to meet timing requirements and reducing the effort to change parameters and explore design space. Pre-processor directives *(#pragma)* were used to convey to the HLS compiler details about parallelization and logic implementation. *#pragma HLS pipeline* $II = 1$ sets the design throughput to one sample per clock cycle, and ensures the adaptive filter updates its output and filter coefficients before the next sample arrives at the input. *#pragma array_partition* maps arrays (i.e. the tapped delay line $x[n]$ ... $x[n-7]$ and filter coefficients $f[0]$ ... $f[7]$) into multiple registers rather than one large memory (block RAM) for simultaneous access. *#pragma HLS unroll* exposes parallelism by enabling all the filter taps to be executed in the same clock cycle.

Internal details of the LMS module are illustrated in fig. 2.3 using a signal flow diagram. The bit-widths are represented in a $Qn.m$ format where n is the total number of bits and m is the number of fractional bits after an assumed decimal position. Arithmetic was performed in signed fixed-point format using Xilinxs' *ap_int* datatype. While fixed-point arithmetic is significantly faster than floating-point, it comes at the risk of round-off and overflow errors due to limited precision and range. To minimize this risk, the number of integer and fractional bits for the filter coefficients, products, and accumulators were selected with the assistance of the MATLAB fixed-point toolbox. We selected the minimum bit-widths that closely matched the output of a floating-point simulation of LMS filtering on pre-recorded pump-probe data. This approach generally prevented overflow, except in areas where transmissivity was significantly higher than the pre-recorded data. To address it, an additional guard bit can be added to the coefficients or electronic (after sample) / optical (before sample) attenuation of the probe beam can avoid coefficient overflow.

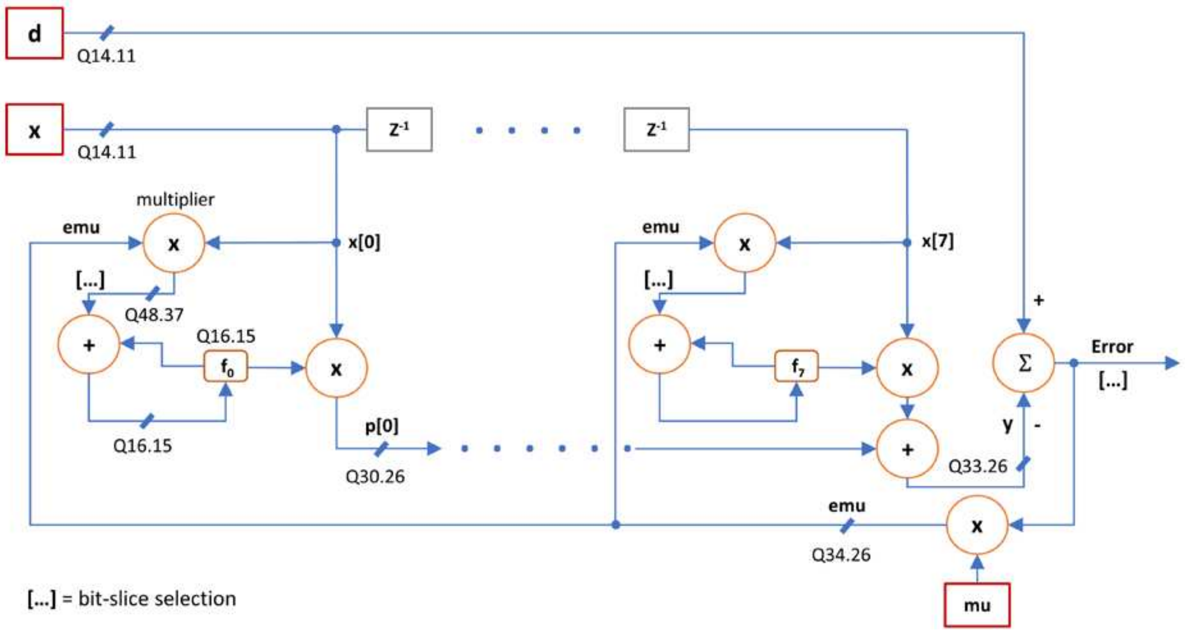


Fig. 2.3. Signal flow diagram of the LMS module.

The HLS design was simulated using C/C++ test-benches to verify functional correctness, and that the output matches the MATLAB simulation. The HLS tool synthesized a hardware description of the module and a register-transfer level (RTL) model of the logic implementation. This generated RTL was then verified using C/C++ test-benches via co-simulation, which simulates the behavior of the scheduled hardware as it would run on the FPGA in the presence of a clock. The timing report showed that the module could run at a clock rate of 25 MHz with the requested sample initiation interval (II) of 1 clock cycle.

Experimental Setup

The pump-probe microscope setup is shown in figure 2.4 below. Pump and probe pulses at 530 nm and 480 nm, respectively, with a cross-correlation of 800 fs, were generated by a two-color laser source described in reference [38]. Pump and probe power out of the two-color laser source were both 10 mW. The pump was modulated with a square wave at 1.5 MHz with an acousto-optic modulator (AOM). Before the microscope, a 50/50 beam splitter directed a portion of the probe beam towards a reference photodiode (PDA36A, Thorlabs), which was connected to ADC CH1. After the microscope, the transmitted probe beam was detected with a second photodiode and connected to ADC CH2. In all the experiments using RINS, the feedback coefficient was set to, $\mu = 2 \times 10^{-6}$ and total gain from ADC to DAC was $2^5 = 32$.

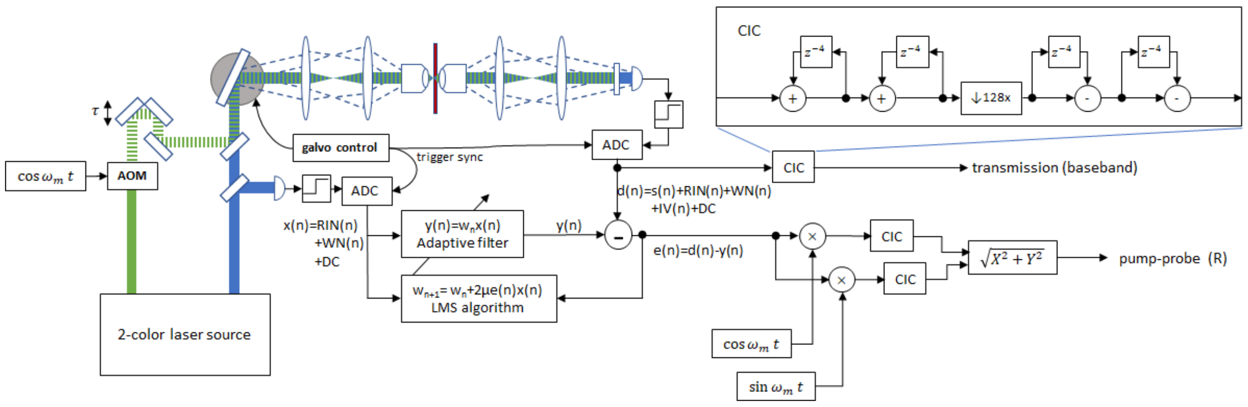


Fig. 2.4. A conventional pump-probe microscope is employed to generate probe signal, $d(n)$, and reference, $x(n)$. The probe and reference signals feed into the FPGA to produce the output $e(n)$, which contains the pump-probe signal minus estimated RIN. Then a software LIA and CIC filter are used to demodulate the pump-probe signal from $e(n)$.

The Red Pitaya DAC output was then either re-digitized by a data acquisition (DAQ) device (Analog Discovery Studio, Digilent) for processing with a software lock-in algorithm or fed into a hardware lock-in device (Moku, Liquid Instruments). For image acquisition, we used TTL synchronization from y-scan mirror to trigger the DAQ acquisition while a 3.5 KHz resonant scan mirror completed the x-scan. Translation along y-axis was achieved by stepping a mechanical stage by 1 µm after one acquisition, repeated for 100 lines. Image field of view was 75 x 100 µm^2. To further improve SNR, we captured and averaged data for a total of 10 passes. Data without RINS was obtained by directly digitizing the output using DAQ, from the probe photodetector.

A MATLAB script was used as a lock-in amplifier (LIA) to extract modulated amplitude at the modulation frequency. The modulation reference was derived from the TTL sync output of the function generator driving the pump AOM. The pump modulation TTL sync was captured on the DAQ analog CH2. This square wave was then converted to in-phase (I) and quadrature (Q) sinusoids by a narrow band-pass FIR filter at the fundamental, followed by a Hilbert transform and a normalization step to eliminate amplitude variations. The resulting complex vector was rotated by multiplication with $e^{i\Phi}$ to bring the lock-in X channel in phase with absorptive signals (i.e. two-photon absorption, excited-state absorption). The real and imaginary parts of the reference oscillator were then individually mixed with the output to obtain the I and Q products. These I and Q products were then passed through a CIC decimator with $R = 128, M = 4$, and $N = 2$ to yield X and Y lock-in channels. A simple MATLAB code snippet that demonstrates this process is as follows:

```
% Setup CIC filter
decimator = dsp.CICDecimator(128,4,2);
% Band-pass parameters
Fs = 50;  % Sampling Frequency
N      = 100;    % Order
Fstop1 = 1.4;    % First Stopband Frequency
```

```
Fpass1 = 1.45;  % First Passband Frequency
Fpass2 = 1.55;  % Second Passband Frequency
Fstop2 = 1.6;   % Second Stopband Frequency
Wstop1 = 1;     % First Stopband Weight
Wpass  = 1;     % Passband Weight
Wstop2 = 1;     % Second Stopband Weight
% Calculate coefficients using FIRLS function and obtain filter object
b  = firls(N, [0 Fstop1 Fpass1 Fpass2 Fstop2 Fs/2]/(Fs/2), [0 0 1 1 0 0], [Wstop1
Wpass Wstop2]);
Hd = dfilt.dffir(b);
. . .
filt_TTL = filter(Hd, TTL); % filter TTL sync
filt_TTL_hilbert = hilbert(filt_TTL);
% Normalize each data point
filt_TTL_hilbert_norm = filt_TTL_hilbert./abs(filt_TTL_hilbert);
osc_shifted = filt_TTL_hilbert_norm * exp(i*phase_offs);
% Lock-in
I = Signal.*real(osc_shifted);
Q = Signal.*imag(osc_shifted);
decimator.reset();
lia_x = decimator(I);
decimator.reset();
lia_y = decimator(Q);
```

Results

Spectroscopic (Non-Imaging) Measurement Result

Figure 2.5 shows the pump-probe delay scan of a single, uniform $Bi_4Ge_3O_{12}$ crystal, acquired with and without RINS in place. At each probe delay, we acquired a 327.68 μs window of the signal using DAQ and calculated the power spectral density (PSD) with MATLAB's *pwelch()* function with a Blackman window of length 4096. During these measurements, the resonant x-scanner was enabled to prevent heat from building up at the focal spot and give the adaptive filter some minor transmissivity variations to keep up with. As expected for our conditions (530 nm pump, 480 nm probe), the signal at the pump modulation frequency (1.5 MHz) indicates a non-degenerate two-photon absorption that traces the cross-correlation of the pump and probe pulses [29].

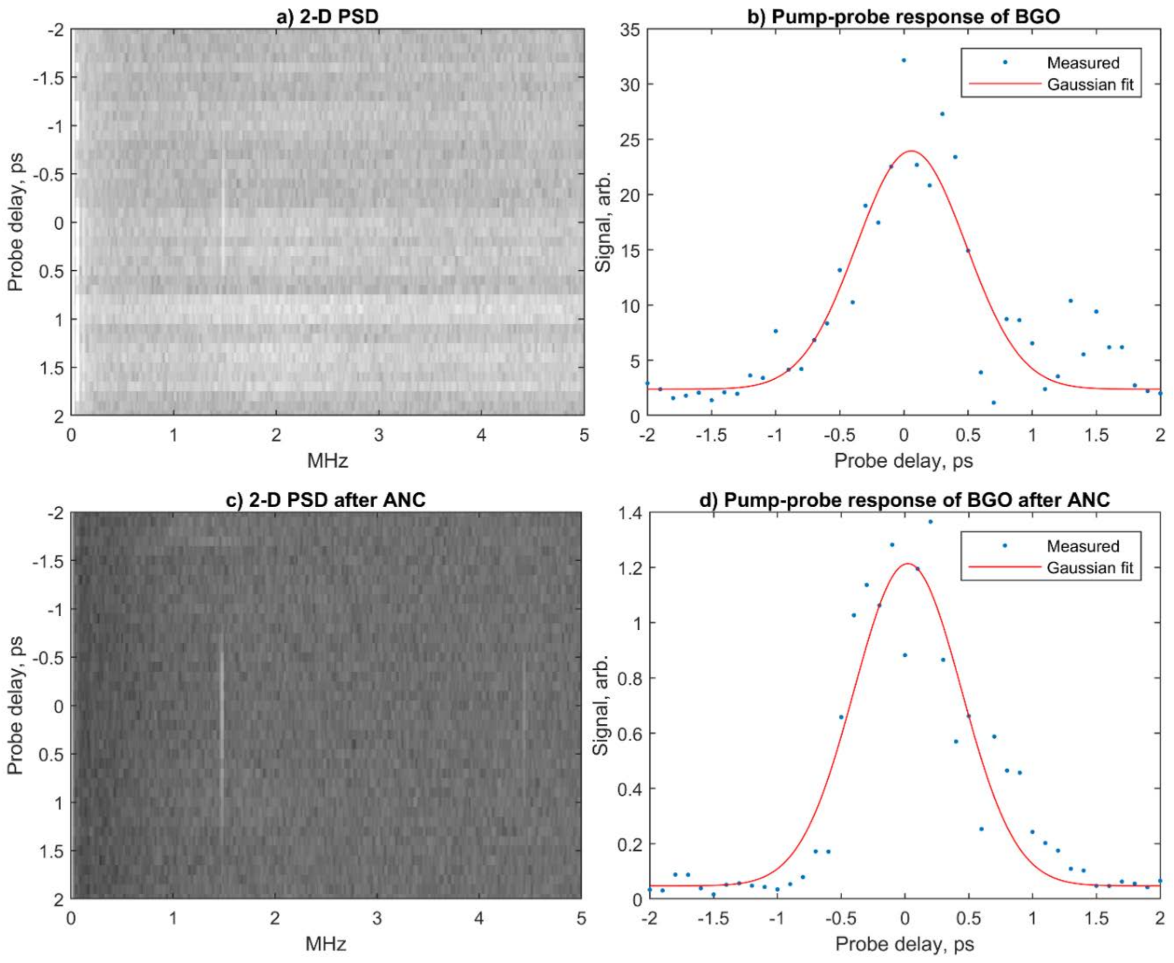


Fig. 2.5. Pump-probe delay scan of 2-photon absorption response in BGO. a) Power spectral density (PSD) of probe photodiode signal with respect to pump-probe delay, without RINS. b) PSD at 1.5 MHz pump modulation frequency, without RINS. c) PSD of probe with RINS, showing lower noise floor and making the 4.5 MHz pump modulation harmonic visible. d) PSD with RINS at 1.5 MHz pump modulation frequency.

Figure 2.6 shows the power spectrum of the RINS output along with the electronics noise floor and shot noise floor. The total SNR of the pump-probe signal after RINS is +15 dB. The total gain (32x) through the FPGA places the shot noise floor on par with the input noise of the DAQ. But as can be seen from the red line, the overall noise floor of the Red Pitaya ADC, DAC and LMS module places the electronics noise floor around 12 dB above the shot noise floor. From this we conclude that the Red Pitaya ADC noise floor is the limiting factor in shot noise limited detection.

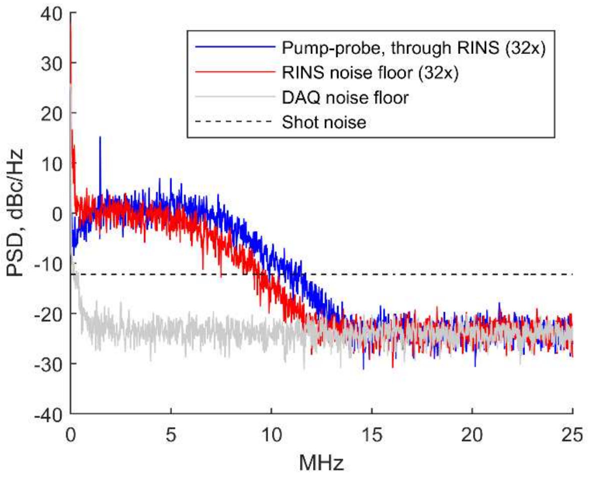


Fig. 2.6. Power spectra of RINS output, electronics noise floor, and shot noise floor. Units in dBc/Hz, referenced to the RINS system electronics noise floor.

Imaging Result

To demonstrate the ability of RINS to maintain balance during a high-speed imaging scenario, a sample of crushed BGO was placed under the objective, then a spacer was placed around the particles with a coverslip on top. Figure 2.7 shows imaging results at 0 ps and 2 ps probe delays, both with and without RINS in place. Each scan line was acquired for 10 repetitions, and the resulting lock-in outputs and PSDs for each scan line were averaged across all repetitions. Consistent with our previous findings, the lock-in, without ANC, sees a significant amount of high-frequency RIN within its passband (magenta box, Fig. 2.7 b, d). As a result, an image is formed of the sample transmissivity and has no dependence on probe delay (Fig. 2.7 a, e). By contrast, with RINS, noise is significantly reduced across a broad range of RF frequencies (Fig. 2.7 d, h), making the 1.5 MHz pump modulation and its 4.5 MHz harmonic clearly visible in the PSD. In addition, the lock-in output recovers a clear dependence on probe delay (Fig. 2.7 c, g), consistent with the delay scan (Fig. 2.5).

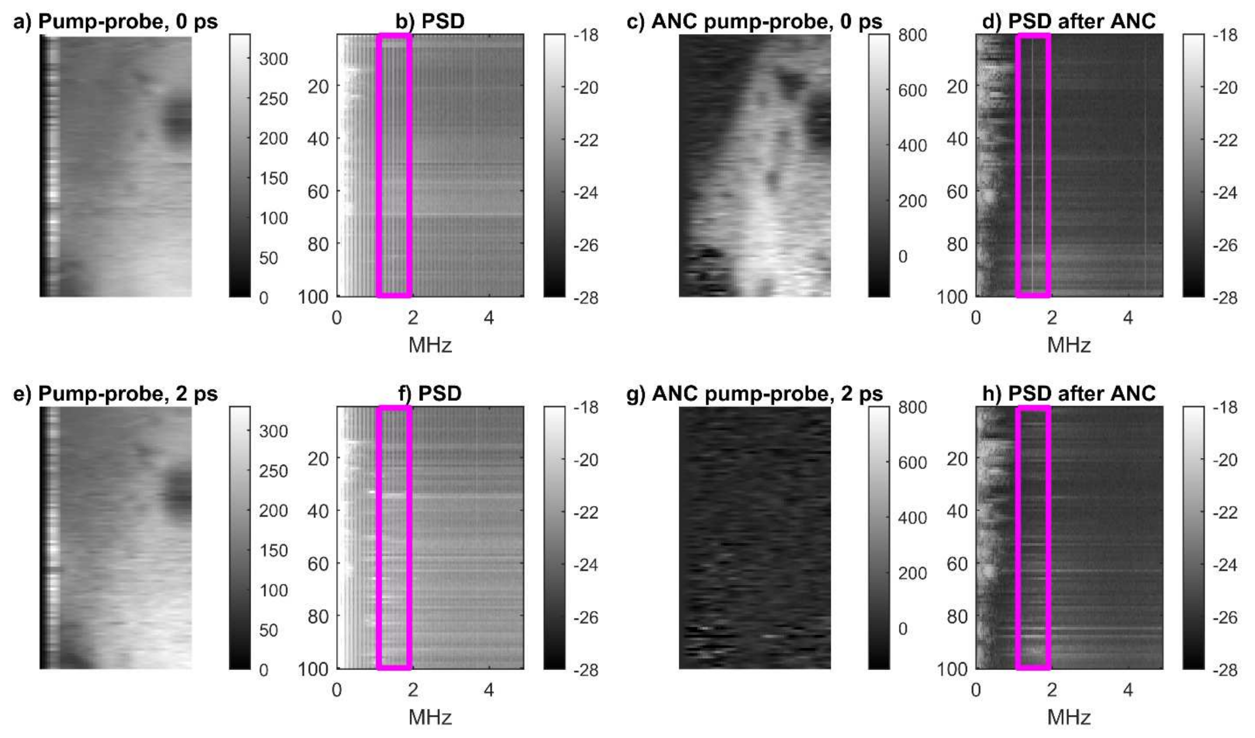
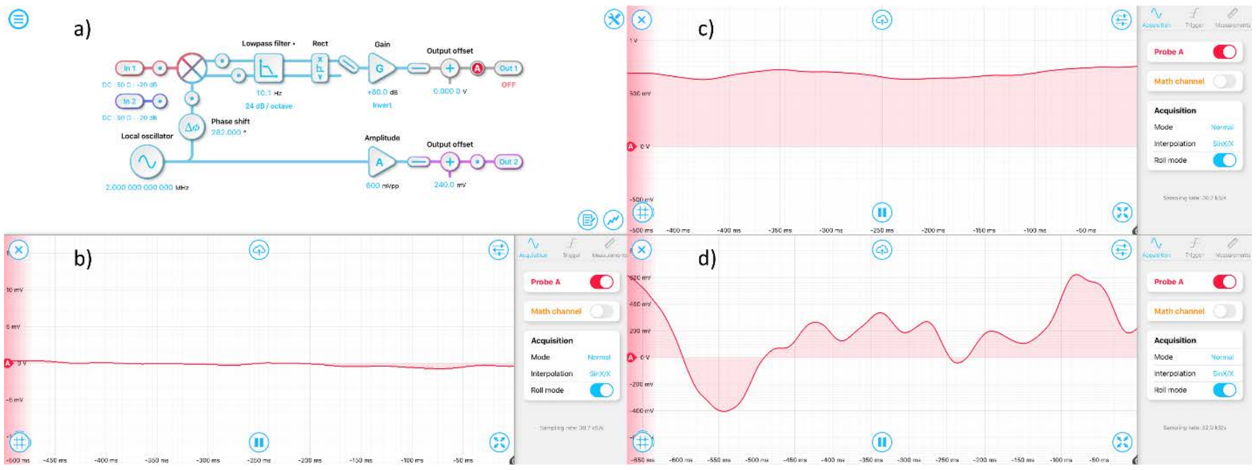


Fig. 2.7. Imaging results at $\boldsymbol{\tau = 0\ ps}$ probe delay (top row) and at $\boldsymbol{\tau = 2\ ps}$ probe delay (bottom row).

Compatibility with Commercial Lock-In Amplifiers

Finally, we tested the RINS system as a drop-in pre-filtering device in front of a commercial lock-in amplifier (Moku, Liquid Instruments). As before, the sample is monolithic BGO, and probe delay is set to $\tau = 0$. From fig. 2.8, the pump-probe signal is noticeably stable after RIN has been removed from the probe, corresponding to a significant reduction in noise.



Lowpass filter: 10.1 Hz, 24 dB / octave
Gain: +80.0 dB, Invert
Output offset: 0.000 0 V
Out 1: OFF
In 1: DC : 50 Ω : -20 dB
In 2: DC : 50 Ω : -20 dB
Phase shift: 282.000 °
Local oscillator: 2.000 000 000 000 MHz
Amplitude: 600 mVpp
Output offset: 240.0 mV

Probe A: on
Math channel: off
Acquisition
Mode: Normal
Interpolation: SinX/X
Roll mode: on

Fig. 2.8. RIN suppression prefiltering effects on output of commercial lock-in amplifier. a) Lock-in configuration. b) Probe, reference connected to RINS, RINS output then connected to lock-in, pump beam blocked. c) Pump unblocked, showing pump-probe signal through RINS. d) Pump unblocked, probe detector connected directly to lock-in, without RINS.

Conclusion

To summarize, our results on BGO visibly show an enhancement in SNR, obtained in real-time using the RINS system which produces an analog output compatible with a conventional lock-in amplifier. Though the front-end ADC noise floor of Red Pitaya prevented shot noise-limited detection, the device enabled transient absorption imaging under conditions that are impossible to image using a lock-in amplifier alone (i.e. high levels of RIN combined with effects due to fast, resonant scanning of the beam). Data collection time was pronounceably reduced from ~6 hours down to ~35 minutes compared to our previous all-software implementation. We anticipate this plug-and-play RIN denoising device to find applications in any experimental technique that relies on detecting small perturbations to a probe laser, such as transient absorption, stimulated Raman scattering, and photothermal microscopy.

Thermal Noise, Shot Noise and Quantization Noise

The ADC on the Red Pitaya 125-14 is LTC2145-14 (Linear Technology) which has an input impedance, $R = 1\ M\Omega$. The thermal noise at $45°C$ is calculated as follows [39],

$$Thermal\ noise = 4k_BTR = 1.76 * 10^{-14}\ V^2/Hz$$

Where k_B is Boltzmann constant in joules/kelvin.

Shot noise for $Iavg = 245\ \mu A$ [Iavg = 0.37 Vavg / PDA36A TIA gain factor 1510 V/A] is given by [40],

$$Shot\ noise = 2eI * gain^2 = 1.79\ * 10^{-16}\ V^2/Hz$$

And the quantization noise approximated by a sawtooth error is given by [41],

$$Quantization\ noise = \frac{q^2}{12 * 60\ MHz\ ADC\ bandwidth} = 2.1 * 10^{-17}\ V^2/Hz$$

Where $2^{-13}\ V$ is the quantization step.

The pixel dwell time is 2.56 $\mu s/pixel$, which is a bandwidth of 390.625 KHz. The RMS values of the quantities above, within this bandwidth are,

$$Thermal\ noise, rms = 82.9\ \mu V$$

$$Shot\ noise, rms = 8.36\ \mu V$$

$$Quantization\ noise, rms = 2.86\ \mu V$$

These calculations show that our experimental setup is currently limited by thermal noise.

Future Directions

To achieve shot noise limited detection, an ADC which has its effective noise floor below the shot noise limit is a must. An appropriate ADC can be selected by carefully taking into consideration the noise contribution from the following sources: thermal noise, quantization noise, differential non-linearity noise, and ADC jitter noise.

REFERENCES

[1] C. M. McGraw, G. Khalil, and J. B. Callis, “Comparison of time and frequency domain methods for luminescence lifetime measurements,” *J. Phys. Chem. C*, vol. 112, no. 21, pp. 8079–8084, 2008, doi: 10.1021/jp711867u.

[2] E. Gratton, S. Breusegem, J. Sutin, Q. Ruan, and N. Barry, “Fluorescence lifetime imaging for the two-photon microscope: time-domain and frequency-domain methods,” *J. Biomed. Opt.*, vol. 8, no. 3, p. 381, 2003, doi: 10.1117/1.1586704.

[3] A. T. N. Kumar, S. B. Raymond, B. J. Bacskai, and D. A. Boas, “Comparison of frequency-domain and time-domain fluorescence lifetime tomography.,” *Opt. Lett.*, vol. 33, no. 5, pp. 470–472, 2008, doi: 10.1364/OL.33.000470.

[4] N. G. Chen and Q. Zhu, “Time-resolved optical measurements with spread spectrum excitation,” *Opt. Lett.*, vol. 27, no. 20, p. 1806, 2002, doi: 10.1364/ol.27.001806.

[5] Q. Zhang, H. W. Soon, H. Tian, S. Fernando, Y. Ha, and N. G. Chen, “Pseudo-random single photon counting for time-resolved optical measurement,” *Opt. Express*, vol. 16, no. 17, p. 13233, 2008, doi: 10.1364/oe.16.013233.

[6] Q. Zhang and N. Chen, “Pseudo-random single photon counting system: a high speed implementation and its applications,” *Des. Qual. Biomed. Technol. IV*, vol. 7891, no. 1, p. 78910G, 2011, doi: 10.1117/12.874390.

[7] Q. Zhang, L. Chen, and N. Chen, “Pseudo-random single photon counting system: a high speed implementation,” *Biomed. Opt. Express*, vol. 1, no. 1, p. 41, 2010, doi: 10.1117/12.874390.

[8] J. W. Wilson, J. K. Park, W. S. Warren, and M. C. Fischer, “Flexible digital signal processing architecture for narrowband and spread-spectrum lock-in detection in multiphoton microscopy and

time-resolved spectroscopy," *Rev. Sci. Instrum.*, vol. 86, no. 3, p. 033707, 2015, doi: 10.1063/1.4916261.

[9] R. A. Colyer, C. Lee, and E. Gratton, "A Novel Fluorescence Lifetime Imaging System That Optimizes Photon Efficiency," *Microsc. Res. Tech.*, vol. 71, no. 3, pp. 201–213, 2008.

[10] B. Friedlander, "System Identification Techniques for Adaptive Signal Processing," *Circuits Syst. Signal Process*, vol. 1, no. I, pp. 3–41, 1982.

[11] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 4th ed. New York: Springer, 2013.

[12] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 4th ed. New York: Springer, 2014.

[13] W. T. Padgett and D. V. Anderson, *Fixed-Point Signal Processing.* San Rafael, CA: Morgan & Claypool, 2009.

[14] K. J. Morris, M. S. Roach, W. Xu, J. N. Demas, and B. A. DeGraff, "Luminescence lifetime standards for the nanosecond to microsecond range and oxygen quenching of ruthenium(II) complexes," *Anal. Chem.*, vol. 79, no. 24, pp. 9310–9314, 2007, doi: 10.1021/ac0712796.

[15] B. R. Gayathri, J. R. Mannekutla, and S. R. Inamdar, "Effect of binary solvent mixtures (DMSO/water) on the dipole moment and lifetime of coumarin dyes," *J. Mol. Struct.*, 2008, doi: 10.1016/j.molstruc.2008.02.020.

[16] D. W. Hawkins, "Altera JTAG-to-Avalon-MM Tutorial," 2012. .

[17] E. L. Sciuto *et al.*, "Photo-physical characterization of fluorophore Ru(bpy)32+ for optical biosensing applications," *Sens. Bio-Sensing Res.*, vol. 6, pp. 67–71, 2015, doi: 10.1016/j.sbsr.2015.09.003.

[18] A. S. Golub, A. S. Popel, L. Zheng, and R. N. Pittman, “Analysis of Phosphorescence Decay in Heterogeneous Systems: Consequences of Finite Excitation Flash Duration,” *Photochem. Photobiol.*, vol. 69, no. 6, pp. 624–632, Jun. 1999, doi: 10.1111/j.1751-1097.1999.tb03338.x.

[19] Y. Zhu and J. X. Cheng, “Transient absorption microscopy: Technological innovations and applications in materials science and life science,” *J. Chem. Phys.*, vol. 152, p. 020901, 2020, doi: 10.1063/1.5129123.

[20] M. C. Fischer, J. W. Wilson, F. E. Robles, and W. S. Warren, “Invited Review Article: Pump-probe microscopy,” *Rev. Sci. Instrum.*, vol. 87, no. 3, p. 031101, 2016, doi: 10.1063/1.4943211.

[21] C. Zhang, D. Zhang, and J. X. Cheng, “Coherent Raman Scattering Microscopy in Biology and Medicine,” *Annual Review of Biomedical Engineering*. 2015, doi: 10.1146/annurev-bioeng-071114-040554.

[22] E. M. Grumstrup, M. M. Gabriel, E. E. M. Cating, E. M. Van Goethem, and J. M. Papanikolas, “Pump-Probe Microscopy: Visualization and Spectroscopy of Ultrafast Dynamics at the Nanoscale,” *Chem. Phys.*, vol. 458, pp. 30–40, 2015, doi: 10.1016/j.chemphys.2015.07.006.

[23] P. Tian and W. S. Warren, “Ultrafast measurement of two-photon absorption by loss modulation.,” *Opt. Lett.*, vol. 27, no. 18, pp. 1634–1636, 2002, doi: 10.1364/OL.27.001634.

[24] N. Coluccelli, V. Kumar, M. Cassinerio, G. Galzerano, M. Marangoni, and G. Cerullo, “Er/Tm:fiber laser system for coherent Raman microscopy,” *Opt. Lett.*, vol. 39, no. 11, pp. 3090–3093, 2014, doi: 10.1364/OL.39.003090.

[25] P. Qin *et al.*, “Reduction of timing jitter and intensity noise in normal-dispersion passively mode-locked fiber lasers by narrow band-pass filtering,” *Opt. Express*, vol. 22, no. 23, pp. 28276–28283, 2014, doi: 10.1364/OE.22.028276.

[26] I. L. Budunoğlu, C. Ülgüdür, B. Oktem, and F. Ö. Ilday, “Intensity noise of mode-locked fiber

lasers," *Opt. Lett.*, vol. 34, no. 16, pp. 2516–2518, 2009, doi: 10.1364/OL.34.002516.

[27] C. W. Freudiger, W. Yang, G. R. Holtom, N. Peyghambarian, X. S. Xie, and K. Q. Kieu, "Stimulated Raman scattering microscopy with a robust fibre laser source," *Nat. Photonics*, vol. 8, no. 2, pp. 153–159, 2014, doi: 10.1038/nphoton.2013.360.

[28] J. W. Wilson and R. A. Bartels, "Rapid birefringent delay scanning for coherent multiphoton impulsive raman pump-probe spectroscopy," *IEEE J. Sel. Top. Quantum Electron.*, vol. 18, no. 1, pp. 130–139, 2012, doi: 10.1109/JSTQE.2011.2106113.

[29] E. Wang, S. Gupta, and J. W. Wilson, "Adaptive noise cancelling for transient absorption microscopy," *J. Biomed. Opt.*, no. Accepted., 2020.

[30] R. Finger, F. Curotto, R. Fuentes, R. Duan, L. Bronfman, and D. Li, "A FPGA-based Fast Converging Digital Adaptive Filter for Real-time RFI Mitigation on Ground Based Radio Telescopes," *Publ. Astron. Soc. Pacific*, vol. 130, no. 984, p. 25002, 2017, doi: 10.1088/1538-3873/aa972f.

[31] R. Akeela and B. Dezfouli, "Software-defined Radios: Architecture, state-of-the-art, and challenges," *Computer Communications*. 2018, doi: 10.1016/j.comcom.2018.07.012.

[32] E. Flater *et al.*, "Error estimation and enhanced stiffness sensitivity in contact resonance force microscopy with a multiple arbitrary frequency lock-in amplifier (MAFLIA)," *Meas. Sci. Technol.*, 2020.

[33] G. A. Stimpson, M. S. Skilbeck, R. L. Patel, B. L. Green, and G. W. Morley, "An open-source high-frequency lock-in amplifier," *Rev. Sci. Instrum.*, 2019, doi: 10.1063/1.5083797.

[34] D. M. Harcombe, M. G. Ruppert, and A. J. Fleming, "A review of demodulation techniques for multifrequency atomic force microscopy," *Beilstein J. Nanotechnol.*, vol. 11, pp. 76–91, 2020, doi: 10.3762/bjnano.11.8.

[35] S. Gupta, E. Wang, S. Derrien, and J. W. Wilson, "RINS: A FPGA-based Real-time Relative Intensity Noise Suppressor in Pump-probe Microscopy," *Rev. Sci. Instrum.*

[36] P. C. D. Hobbs, "Reaching the shot noise limit for $10," *Opt. Photonics News*, vol. 2, no. 4, pp. 17–23, 1991, doi: 10.1364/OPN.2.4.000017.

[37] P. Demin, "Red Pitya Notes." .

[38] S. R. Domingue, R. A. Bartels, A. J. Chicco, and J. W. Wilson, "Transient absorption imaging of hemes with 2-color, independently tunable visible-wavelength ultrafast source," *Biomed. Opt. Express*, vol. 8, no. 6, pp. 2807–2821, 2017.

[39] "Johnson noise." https://en.wikipedia.org/wiki/Johnson–Nyquist_noise.

[40] "Shot noise." https://en.wikipedia.org/wiki/Shot_noise.

[41] "Quantization noise." https://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf.

APPENDIX A

Verilog code for LMS adaptive filter in chapter 1

```
module adaptive_fir
#(parameter       W1 = 12,W2 = 32,L=16 )
(input clk,
input reset,
input signed [11:0] x_in,
input signed [13:0] d_in,
input [7:0] mu_in,
output reg signed [31:0] y_out,
output reg signed [32:0] e_out,
output reg signed [15:0] f_0,
output reg signed [15:0] f_1,
output reg signed [15:0] f_2,
output reg signed [15:0] f_3,
output reg signed [15:0] f_4,
output reg signed [15:0] f_5,
output reg signed [15:0] f_6,
output reg signed [15:0] f_7,
output reg signed [15:0] f_8,
output reg signed [15:0] f_9,
output reg signed [15:0] f_10,
output reg signed [15:0] f_11,
output reg signed [15:0] f_12,
output reg signed [15:0] f_13,
output reg signed [15:0] f_14,
output reg signed [15:0] f_15
);

//regs and wire declarations:
reg signed [11:0] x [0:L-1];
reg signed [15:0] f [0:L-1];
reg signed [13:0] d ;
wire signed [31:0] d_extended;
reg signed [27:0] p [0:L-1];
wire signed [32:0] emu;
reg signed [44:0] xemu [0:L-1];
reg signed [15:0] xemu_truncd [0:L-1];
reg signed [16:0] xemu_round [0:L-1];
wire signed [31:0] y;
wire signed [32:0] e;
wire signed [29:0] delta_round_factor = (mu_in == 8'd0) ? 30'sd0 : 30'sd1 <<<
(mu_in - 1'd1);

initial begin
      d <= 14'd0;
      x[0] <= 12'd0;
      x[1] <= 12'd0;
      x[2] <= 12'd0;
      x[3] <= 12'd0;
      x[4] <= 12'd0;
      x[5] <= 12'd0;
```

```
        x[6] <= 12'd0;
        x[7] <= 12'd0;
        x[8] <= 12'd0;
        x[9] <= 12'd0;
        x[10] <= 12'd0;
        x[11] <= 12'd0;
        x[12] <= 12'd0;
        x[13] <= 12'd0;
        x[14] <= 12'd0;
        x[15] <= 12'd0;

        f[0] <= 16'd0;
        f[1] <= 16'd0;
        f[2] <= 16'd0;
        f[3] <= 16'd0;
        f[4] <= 16'd0;
        f[5] <= 16'd0;
        f[6] <= 16'd0;
        f[7] <= 16'd0;
        f[8] <= 16'd0;
        f[9] <= 16'd0;
        f[10] <= 16'd0;
        f[11] <= 16'd0;
        f[12] <= 16'd0;
        f[13] <= 16'd0;
        f[14] <= 16'd0;
        f[15] <= 16'd0;

         f_0 <= 16'd0;
         f_1 <= 16'd0;
         f_2 <= 16'd0;
         f_3 <= 16'd0;
         f_4 <= 16'd0;
         f_5 <= 16'd0;
         f_6 <= 16'd0;
         f_7 <= 16'd0;
         f_8 <= 16'd0;
         f_9 <= 16'd0;
         f_10 <= 16'd0;
         f_11 <= 16'd0;
         f_12 <= 16'd0;
         f_13 <= 16'd0;
         f_14 <= 16'd0;
         f_15 <= 16'd0;

end

always @(posedge clk or negedge reset)
if (!reset)
begin
        d <= 14'd0;
        x[0] <= 12'd0;
        x[1] <= 12'd0;
        x[2] <= 12'd0;
        x[3] <= 12'd0;
        x[4] <= 12'd0;
        x[5] <= 12'd0;
        x[6] <= 12'd0;
        x[7] <= 12'd0;
        x[8] <= 12'd0;
        x[9] <= 12'd0;
        x[10] <= 12'd0;
        x[11] <= 12'd0;
        x[12] <= 12'd0;
        x[13] <= 12'd0;
        x[14] <= 12'd0;
        x[15] <= 12'd0;

        f[0] <= 16'd0;
        f[1] <= 16'd0;
        f[2] <= 16'd0;
        f[3] <= 16'd0;
        f[4] <= 16'd0;
        f[5] <= 16'd0;
        f[6] <= 16'd0;
        f[7] <= 16'd0;
        f[8] <= 16'd0;
        f[9] <= 16'd0;
        f[10] <= 16'd0;
        f[11] <= 16'd0;
        f[12] <= 16'd0;
        f[13] <= 16'd0;
        f[14] <= 16'd0;
        f[15] <= 16'd0;

         f_0 <= 16'd0;
         f_1 <= 16'd0;
         f_2 <= 16'd0;
         f_3 <= 16'd0;
         f_4 <= 16'd0;
         f_5 <= 16'd0;
         f_6 <= 16'd0;
         f_7 <= 16'd0;
         f_8 <= 16'd0;
         f_9 <= 16'd0;
         f_10 <= 16'd0;
         f_11 <= 16'd0;
         f_12 <= 16'd0;
         f_13 <= 16'd0;
         f_14 <= 16'd0;
         f_15 <= 16'd0;

  end
  else  begin
        d <= d_in;

        x[0] <= x_in;
        x[1] <= x[0];
        x[2] <= x[1];
        x[3] <= x[2];
        x[4] <= x[3];
        x[5] <= x[4];
        x[6] <= x[5];
        x[7] <= x[6];
```

```
x[8] <= x[7];
x[9] <= x[8];
x[10] <= x[9];
x[11] <= x[10];
x[12] <= x[11];
x[13] <= x[12];
x[14] <= x[13];
x[15] <= x[14];

xemu_round[0] <= xemu [0][37:21] + 1'b1 ;
xemu_round[1] <= xemu [1][37:21] + 1'b1 ;
xemu_round[2] <= xemu [2][37:21] + 1'b1 ;
xemu_round[3] <= xemu [3][37:21] + 1'b1 ;
xemu_round[4] <= xemu [4][37:21] + 1'b1 ;
xemu_round[5] <= xemu [5][37:21] + 1'b1 ;
xemu_round[6] <= xemu [6][37:21] + 1'b1 ;
xemu_round[7] <= xemu [7][37:21] + 1'b1 ;
xemu_round[8] <= xemu [8][37:21] + 1'b1 ;
xemu_round[9] <= xemu [9][37:21] + 1'b1 ;
xemu_round[10] <= xemu [10][37:21] + 1'b1 ;
xemu_round[11] <= xemu [11][37:21] + 1'b1 ;
xemu_round[12] <= xemu [12][37:21] + 1'b1 ;
xemu_round[13] <= xemu [13][37:21] + 1'b1 ;
xemu_round[14] <= xemu [14][37:21] + 1'b1 ;
xemu_round[15] <= xemu [15][37:21] + 1'b1 ;

xemu_truncd[0] <= xemu_round[0][16:1];
xemu_truncd[1] <= xemu_round[1][16:1];
xemu_truncd[2] <= xemu_round[2][16:1];
xemu_truncd[3] <= xemu_round[3][16:1];
xemu_truncd[4] <= xemu_round[4][16:1];
xemu_truncd[5] <= xemu_round[5][16:1];
xemu_truncd[6] <= xemu_round[6][16:1];
xemu_truncd[7] <= xemu_round[7][16:1];
xemu_truncd[8] <= xemu_round[8][16:1];
xemu_truncd[9] <= xemu_round[9][16:1];
xemu_truncd[10] <= xemu_round[10][16:1];
xemu_truncd[11] <= xemu_round[11][16:1];
xemu_truncd[12] <= xemu_round[12][16:1];
xemu_truncd[13] <= xemu_round[13][16:1];
xemu_truncd[14] <= xemu_round[14][16:1];
xemu_truncd[15] <= xemu_round[15][16:1];


f[0] <= f[0] + xemu_truncd[0];
f[1] <= f[1] + xemu_truncd[1];
f[2] <= f[2] + xemu_truncd[2];
f[3] <= f[3] + xemu_truncd[3];
f[4] <= f[4] + xemu_truncd[4];
f[5] <= f[5] + xemu_truncd[5];
f[6] <= f[6] + xemu_truncd[6];
f[7] <= f[7] + xemu_truncd[7];
f[8] <= f[8] + xemu_truncd[8];
f[9] <= f[9] + xemu_truncd[9];
f[10] <= f[10] + xemu_truncd[10];
f[11] <= f[11] + xemu_truncd[11];
f[12] <= f[12] + xemu_truncd[12];
```

```
      f[13] <= f[13] + xemu_truncd[13];
      f[14] <= f[14] + xemu_truncd[14];
      f[15] <= f[15] + xemu_truncd[15];

       f_0 <= f[0];
       f_1 <= f[1];
       f_2 <= f[2];
       f_3 <= f[3];
       f_4 <= f[4];
       f_5 <= f[5];
       f_6 <= f[6];
       f_7 <= f[7];
       f_8 <= f[8];
       f_9 <= f[9];
       f_10 <= f[10];
       f_11 <= f[11];
       f_12 <= f[12];
       f_13 <= f[13];
       f_14 <= f[14];
       f_15 <= f[15];

      y_out <= y;
      e_out <= e;

end

always @(*) begin
integer i;
for (i=0; i<L; i=i+1) p[i] <= x[i] * f[i];
end

assign  y = p[0] + p[1]+ p[2] + p[3] +p[4]+ p[5] + p[6] + p[7] + p[8] + p[9]
+ p[10] + p[11] + p[12] + p[13] + p[14] + p[15] ;

assign d_extended = d <<< 10;
assign e = (d_extended-y);


assign emu  =  ((e + delta_round_factor) >>> mu_in);   //Delta control

always @(*) begin
integer i;
for (i=0; i<L ; i = i+1) xemu[i] <= emu * x[i];
end

endmodule
```

APPENDIX B

System diagram from chapter 2.

APPENDIX C

Adaptive filter HLS code from chapter 2

```
//**** IP designer: Saurabh Gupta, CSU, 2020 ****************

#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include <hls_stream.h>
#include <ap_int.h>
#include "./include/LMS.h"

//#define SIMULATE 1

void LMS(hls::stream<ap_uint<16> > &x_in, hls::stream<ap_uint<16> > &d_in,
            hls::stream<ap_uint<16> > &e_out,
            volatile ap_int<32> gain, volatile ap_int<32> mu) {
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE axis off port=x_in
#pragma HLS INTERFACE axis off port=d_in
#pragma HLS INTERFACE axis off port=e_out
#pragma HLS INTERFACE s_axilite port=gain
#pragma HLS INTERFACE s_axilite port=mu

      ap_int < 8 > j;
      static ap_int < 33 > y;
      static ap_int<14> x[L];
      #pragma HLS array_partition variable=x
      static ap_int<16> f[L];
      #pragma HLS array_partition variable=f
      static ap_int <34> emu;
      static ap_int<34> e;
      ap_int < 14 > x_input;
      ap_int < 14 > d_input;
      ap_uint < 16 > read_x;
      ap_uint < 16 > read_d;
      static ap_int < 29 > rescaled_d;
      ap_int<32> shift_factor;
      static ap_int<32> MU;
      static ap_int<16> correction_factor;
      static ap_int<47> xemu[L];
      #pragma HLS array_partition variable=xemu
      static ap_int<17> xemu_round[L];
      #pragma HLS array_partition variable=xemu_round
      static ap_int<16> xemu_truncd[L];
      #pragma HLS array_partition variable=xemu_truncd

#ifdef SIMULATE
      int i = 0;
      do {
#endif
#pragma HLS pipeline II=1
            shift_factor = gain;
            MU = mu;

            // 1) tapped delay line
            read_x = x_in.read();
            read_d = d_in.read();
            x_input = read_x.range(13, 0);
            d_input = read_d.range(13, 0);
            for (j = L - 1; j > 0; j--) {
#pragma HLS loop_tripcount min=7 max=7
#pragma HLS UNROLL
                  x[j] = x[j - 1];
            }
            x[0] = x_input;

            // 2) convolution
            y = 0;
            for (j = 0; j < L; j++) {
#pragma HLS loop_tripcount min=8 max=8
#pragma HLS UNROLL
                  y+= x[j] * f[j];
            }

            // 3) error calculation
            rescaled_d = (d_input.to_int() << 15); // rescale d
            e = rescaled_d.to_int() - y.to_int();
            // feedback loop update of coefficients
            correction_factor = (MU==0? 0 : 1 << (MU-1));
            emu = (correction_factor.to_int() + e.to_int()) >> MU;

            for (j = 0; j < L; j++) {
#pragma HLS loop_tripcount min=8 max=8
#pragma HLS UNROLL
                  xemu[j] = x[j] * emu;  // Q14.11 * Q34.26 = Q47.37
                  xemu_round[j] = xemu[j].range(37,21).to_int() + 1;
                  xemu_truncd[j] = xemu_round[j].range(16,1);
                  f[j] = f[j].to_int() + xemu_truncd[j].to_int();
            }
            printf("%d, ", e.to_int());
            e_out.write(e.range(28-shift_factor, 15-shift_factor).to_int());

#ifdef SIMULATE
      i++;
      } while (i < N_SAMPS);
#endif
}
```

APPENDIX D

C code for a CIC filter

```
// Partial code for CIC filter with R=10 , N=4 , M=1
// By: Saurabh Gupta, CSU, 2020
       integrator1 = integrator1 + input ;
       integrator2 = integrator2 + integrator1 ;
       integrator3 = integrator3 + integrator2 ;
       integrator4 = integrator4 + integrator3 ;

       downsample_clock++ ;
       if ((downsample_clock % 10) == 0) //modulo 10
       {
            comb1 = integrator4 - last_integrator4 ;
      comb2 = comb1 - last_comb1 ;
      comb3 = comb2 - last_comb2 ;
      output = (int)((comb3 - last_comb3)>>1) ; // scaling factor

      last_integrator4 = integrator4 ;
      last_comb1 = comb1;
      last_comb2 = comb2;
      last_comb3 = comb3;
       }
```