PID controller and lock-in amplifier FPGA toolkit

Travis Casey

May 2021

1 Introduction

The need to control physical systems is prevalent throughout experimental physics. In particular, atomic, molecular, and optical physics require very precise temperature and laser frequency control for their experiments. All lasers exhibit some amount of frequency drift over time which inhibits the accuracy and effectiveness of their results. To address this problem and other similar ones, a stabilization scheme using feedback is typically employed.[1,2]

The basic function of a feedback controller is typically to measure some quantity of a system and compare it to the desired value (the setpoint). The controller uses the difference of these values, known as the error signal, to correct the system performance. Likely the most popular such control scheme is known as the proportional-integral-derivative (PID) controller.[3] The PID controller is straightforward to understand and implement, can control a wide variety of physical systems, and can be finely optimized by adjustment of its parameters. However, in terms of the control variable (e.g. temperature or frequency), it requires the error signal to have opposite sign across the setpoint. In other words, when the control variable is too low and the error is negative, then the quantity must be positive when the control variable is too high. Sometimes such a quantity is not immediately apparent in the system.

One such system is controlling the frequency of a laser to match an absorption peak, whether from an optical cavity or atomic spectroscopy. Ideally, the frequency is set such that the absorption is maximized (at the top of the peak). However, both increasing and decreasing frequency at the peak causes the absorption to drop, making it invalid as an error signal for a PID controller. One solution to this problem is called the Pound-Drever-Hall (PDH) technique.[4]

The PDH technique utilizes a lock-in amplifier to create a usable error signal from the absorption data. The lock-in amplifier consists of an oscillator, a mixer, and a low-pass filter. The oscillator modulates the external system (such as the laser frequency) and then mixes the output with a sine wave of the same frequency. For components at the same frequency, this creates a DC component and a component at twice the modulation frequency. Furthermore, the DC component is phase-sensitive. By passing the result through a low-pass filter to remove all oscillating terms, the remaining DC signal indicates the magnitude and phase of components of the input signal with the same frequency as the oscillator. By itself, the lock-in amplifier allows for detection of signals even in noisy environments. The PDH technique employs it to compute a derivative-like signal of the absorption peak, creating an odd function about the desired frequency; this makes it a suitable error signal for the PID controller.

This laser frequency control requires a variety of instruments, some of which are oscillators, a PID controller, and a lock-in amplifier.[4] The purpose of this paper is to present an implementation of these components on a field programmable gate array (FPGA). A digital implementation offers several benefits over the analog counterparts. The FPGA offers a cost-effective and highly customizable alternative that retains the precision and effectiveness of dedicated instruments.

The primary benefit of FPGA-based designs is the simple implementation of microcircuitry for a wide variety of purposes. This is done using a hardware description language (HDL); the language in this implementation is Verilog. The programmability and reprogrammability of the hardware allows for a single FPGA to be employed for multiple purposes or to be easily customized to a specific application.[2] The project detailed in this report offers internal oscillators, a PID controller, a lock-in amplifier and filters that can be employed in various setups to suit applications from basic signal filtering to PDH frequency locking. Additionally, parameters can be set and adjusted easily thanks to the integration of the FPGA with a microprocessor.

2 Hardware

The board used in this project is the Red Pitaya STEMlab 125-14 which offers both the Xilinx Zynq 7010 SoC FPGA and a dual-core ARM Cortex-A9 microprocessor on a single small board. The FPGA has an internal clock speed of 125 MHz, and is connected with two full-speed analog-to-digital converters (ADC) and digital-to-analog converters (DAC), operating at 125 MS/s. Both the ADCs and the DACs have 14-bit resolution. The ADCs have input ranges of ± 1 V or ± 20 V which can be configured by the user via a jumper. The DACs have an output range of ± 1 V. A variety of other peripherals are available such as GPIO pins and daisy-chain connectors.

The inclusion of both a microprocessor and a FPGA on a single device greatly improves performance and increases its versatility. The programmable hardware of the FPGA can handle the bulk of fast and complex data operations, while the processor can be used for interfacing with the FPGA, recording and processing data, and communicating with other computers. In particular, the processor can read and write to the FPGA memory registers, allowing for data retrieval and real-time parameter adjustments. This setup allows the user to interact with the FPGA via an external interface with no reprogramming of the FPGA itself. Communication with the processor is typically handled through the ethernet port on the board. Support for wireless connection is also offered with a WiFi adapter. The user can use secure shell (SSH) to access the Linux terminal and thereby interface with the device. Finally, the documentation and software for the Red Pitaya is open-source, easily allowing users to customize its operations, as is done in this project.

3 Theory

As described previously, the main function of the project is to create a digital implementation of instruments used in laser frequency stabilization. However, they can also be employed in other ways, such as temperature control or digital filtering. The instruments are a PID controller, a lock-in amplifier, and oscillators.

3.1 PID Controller

The PID controller is a commonly-used device designed to control a physical variable to a desired value. Fig. 1 is the basic layout of a PID controller. The measured value, which is the property of the system to control, is typically referred to as the process variable, $y(t)$. The desired value for this variable is called the setpoint, $r(t)$, and the difference between these two is referred to as the error signal, $e(t)$. The PID controller uses this feedback to control some element of the system, the control variable. There must be a correlation between the control variable and the process variable or the controller will not work. The error signal provides an indication of how far the process variable is from the setpoint and in which direction the control variable must be altered to drive the process variable to the setpoint.

The PID output uses the combination of three different responses to the error signal to control the system.[3] The first is the proportional signal, whose response is just the error signal with some gain, K_p. This is the main driving signal that always seeks to move the process variable towards the setpoint. However, control with only proportional gain may result in a steady-state error, as the response is weak at low error values. The second component is the integral response, which is the integral of the error signal over time with a gain of K_i. The benefit of the integrator is to remove the steady-state error in the signal, so that the error signal will actually go to zero. Finally there is the derivative response with a gain of K_d. This component responds to sharp movements in the error signal, and helps to reduce problems like oscillations in the output or overshoot of the setpoint. Mathematically, the output of the PID controller, $u(t)$, is as follows:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(t)dt + K_d\, e'(t).$$

Discretization of this process is straightforward. Note that n is the number of clock cycles since the start and ΔT is the time between clock cycles. For this FPGA, ΔT is 8 ns.

$$u[n] = K_p e[n] + K_i\, \Delta T(e[n] + e[n-1] + e[n-2] + \ldots) + K_d \frac{(e[n] - e[n-1])}{\Delta T}.$$

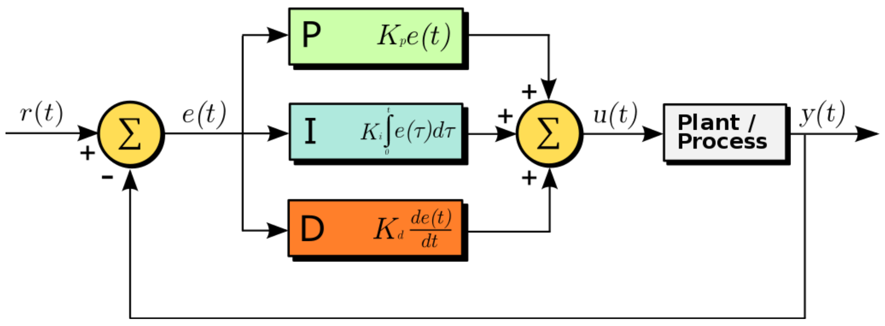


Figure 1: Schematic of a basic PID Controller. Available from Wikimedia Commons.

3.2 Pound-Drever-Hall Technique

Precise laser frequency stabilization is essential in a number of experimental fields. In atomic physics, locking the frequency of a laser to the absorption frequency of atoms or the resonant frequency of a stable cavity is desirable. This is typically done with the Pound-Drever-Hall (PDH) technique.[4]

When passing laser light through a collection of atoms, if the light is at the proper frequency, some of it will be absorbed by the atoms to transition energy levels. This is the basic idea of absorption spectroscopy. By scanning the laser over a range of frequencies and measuring the output of the laser after passing through the atoms, one can observe dips where the light of certain frequencies was absorbed. However, due to thermal movement of the atoms, the frequencies absorbed may not be their actual resonant frequency. This is due to the Doppler effect altering the light frequency relative to those atoms. The effect is a Doppler-broadened spectrum where the peaks are actually Gaussian distributed about the resonant frequency. These wider peaks may overpower and conceal other resonances, such as the fine or hyperfine structure of the atoms.[5]

One solution to this is saturated absorption spectroscopy. The idea is to split a laser into 2 components of different intensities. The first component, known as the pump beam, is higher intensity and passes through the atoms as usual. The second component is the probe beam and has lower intensity. It passes through the atoms in the opposite direction of and overlapping the pump beam. This setup is shown in Fig. 2. A photodiode then measures the probe beam. When the laser frequency is exactly at a resonance of the atoms, both the pump and probe beams can only be absorbed by atoms with near zero velocity relative to the lasers. The stronger pump beam saturates the absorption of these atoms, so the probe beam is less absorbed at the resonant frequency. When the laser frequency is off resonance, the two beams can only be absorbed by different atoms since they must be moving at a velocity of the opposite direction, so this effect does not happen. By taking the difference of the absorbed spectrum and the typical doppler-broadened spectrum, narrow peaks can be obtained at the resonant frequency of the atomic sample.[5]

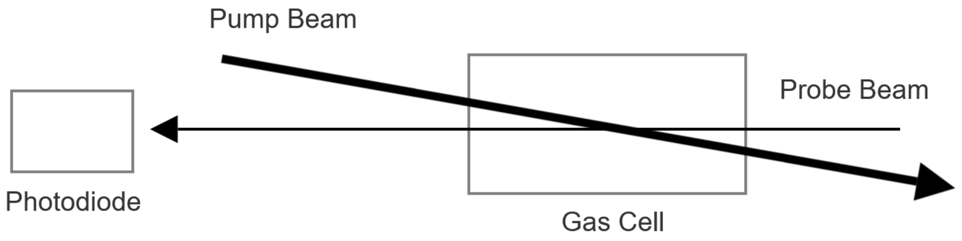


Figure 2: The setup for saturated absorption spectroscopy. The pump and probe beams are derived from the same laser and thus have equal frequencies but different intensities. The photodiode measures the intensity of the probe beam, which will have a Doppler-free absorption spectrum as the laser frequency scans across resonance.

The goal of the PDH technique is to lock the frequency of a laser to one of these resonant frequencies. From here, the process is identical to locking the frequency to the resonance of a stable cavity. Ideally a feedback controller, such as a PID controller, could be used to lock the frequency of the laser as it can handle laser frequency drift from a number of sources. However, the peak from saturated absorption spectroscopy is not a suitable error signal for the PID as it is an even function about the resonant frequency. If the frequency drifts off resonance, the PID would not be able to tell if the frequency needs to be increased or decreased.

The solution is to use a derivative-like signal of the peak, which would be odd about the resonant frequency. To obtain this signal, we can slightly modulate the laser frequency (or phase). If it is on the increasing side of the peak, then there will be oscillations on the absorption signal that are in phase with the modulation. Alternatively, if the laser frequency is currently on the decreasing side, the oscillations will be 180° out of phase. We can then use a lock-in amplifier to obtain this data in a usable form. By using a mixer to multiply that data with a reference signal of the same frequency and phase, the result is a DC signal that is proportional to the cosine of the phase difference and an oscillating signal at twice the frequency. The oscillating term is removed using a low-pass filter, and the resulting signal, known as the error signal, is an odd, linear function about the resonant frequency. A PID controller can then be used to lock the frequency to resonance, where the error signal is zero.[4]

3.3 Lock-In Amplifier

The lock-in amplifier is a powerful tool to extract signals of known phase and frequency from noisy environments. As stated before, it also finds use in the PDH technique.

The amplifier consists of an oscillator, a mixer, and a low-pass filter, as shown in Fig. 3. The oscillator produces a sine wave of a certain frequency, ω_r and phase ϕ. The internal wave can be phase-shifted and is multiplied with the input to the amplifier. In practice, the oscillator is typically also used to modulate the system being measured. This creates an input of a known frequency, which allows for detection even in noisy environments. To see the function of the amplifier, consider a component of the input that has frequency ω_i. Then the result after the mixer is:

$$\sin(\omega_r t + \phi)\sin(\omega_i) = \frac{1}{2}(\cos((\omega_r - \omega_i)t + \phi) - \cos((\omega_r + \omega_i)t + \phi)).$$

The mixer output contains components at the sum and difference of the two frequencies. If the frequencies are equal, then one component is at DC and the other component is at twice their frequency. Passing the result through a low-pass filter to remove the oscillating term yields a DC signal proportional to $\cos(\phi)$. If, however, the two frequencies do not match, both components will still be oscillating. After passing through the low-pass filter, the resultant signal may be greatly attenuated, especially for frequencies far from the modulation frequency. In this way, the amplifier selectively amplifies signal components of the appropriate frequency.

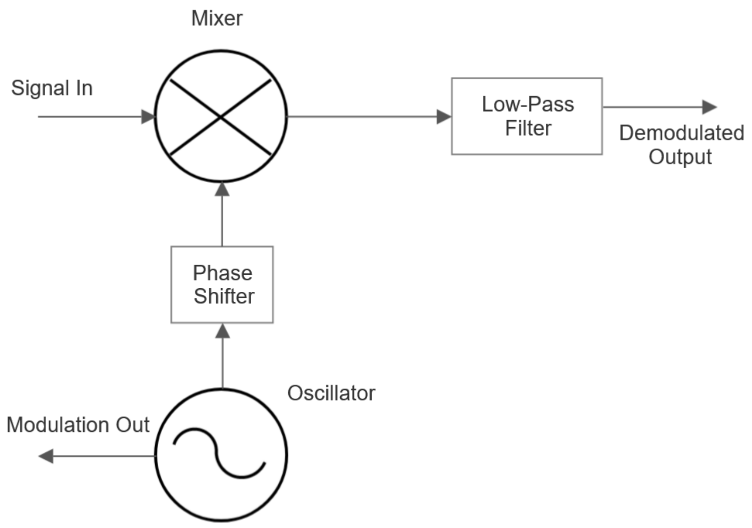


Figure 3: Schematic of a lock-in amplifier. Sometimes a second oscillator or the phase shifter is used for the reference signal to optimize amplification. The modulation out signal is can be used to modulate an external system.

3.4 Low-Pass Filter

The low-pass filter is a circuit that is designed to significantly attenuate signals at high frequency while not affecting those with low frequency. The most basic low-pass filter is the passive first-order RC filter as seen in Fig. 4a. Applying Kirchoff's laws yields the the following differential equation for the circuit:

$$v_{in}(t) - v_{out}(t) = RC\frac{dv_{out}}{dt}.$$

The RC value characterizes the response of the system. In a clearer form, the filter has a cutoff frequency at which signals are attenuated to half power. Lower frequency signals will be less attenuated and higher frequencies will be more attenuated. The cutoff frequency f_c is:

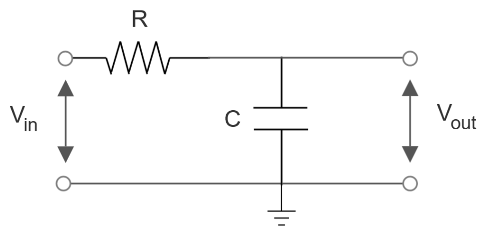
$$f_c = \frac{1}{2\pi RC}.$$

Letting the input variable, v_{in} be x and the output variable, v_{out} be y, the discretization of this is as follows:

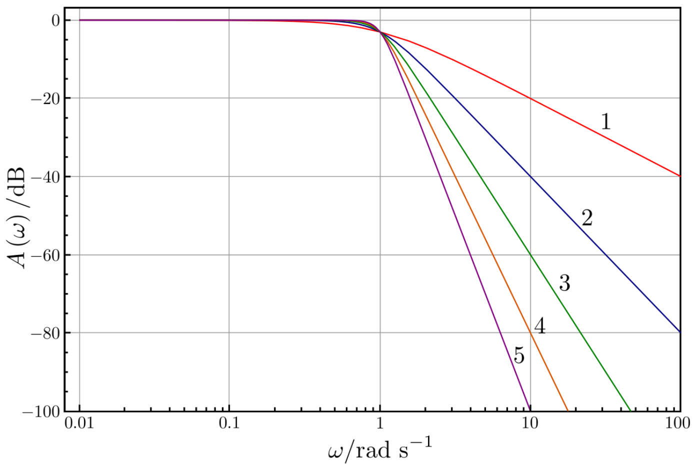
$$x[n] - y[n] = RC\frac{y[n] - y[n-1]}{\Delta T}.$$

Again, ΔT is the time between clock cycles. Rearranging to solve to for $y[n]$ we find:

$$y[n] = \frac{\Delta T}{RC + \Delta T}x[n] + \frac{RC}{RC + \Delta T}y[n-1].$$



(a) Schematic of a first-order RC low-pass filter.



(b) Bode plot of the first five orders of low-pass filter with cutoff frequency 1. The attenuation is in decibels, reflecting the relative power attenuated. In particular, higher order filters provide a much sharper cutoff.

Figure 4: Low-pass filter design and characteristics.

This equation simplifies greatly after defining the parameter α.

$$\alpha = \frac{\Delta T}{RC + \Delta T},$$

$$y[n] = \alpha x[n] + (1 - \alpha) y[n-1].$$

Finally, we would like to know the cutoff frequency f_c of the digital filter in terms of α. This works out to be:

$$f_c = \frac{\alpha}{(1-\alpha)2\pi\Delta T}.$$

This is the digital design of the filter implemented. The lock-in amplifier uses a third-order low-pass filter, which is achieved by running the signal through three successive low-pass filters. The resulting filter provides greater attenuation of signals with frequencies higher than the cutoff frequency, as seen in Fig. 4b.

4 Module Implementation

The project constitutes a number of modules implemented on the FPGA, along with an external remote interface. Each module is based on the function of a common laboratory instrument such as a filter or a controller. The connections between each module in the FPGA design are controlled by multiplexers (mux) which allows for user designation of how the modules work together. For example, the PID controller and lock-in amplifier modules can be used independently, or the output of the lock-in amplifier can be set to the error signal input of the PID controller for laser frequency locking.

4.1 Multiplexer

The multiplexer is a simple module implemented on the FPGA designed to take in multiple input signals and a switch value to designate which input it feeds its one output. The multiplexers used in this project are 16-input or 32-input, so the switch value is from 0 to 15 or 0 to 31, and directly designates which input the multiplexer outputs. Unused inputs are set to 0.

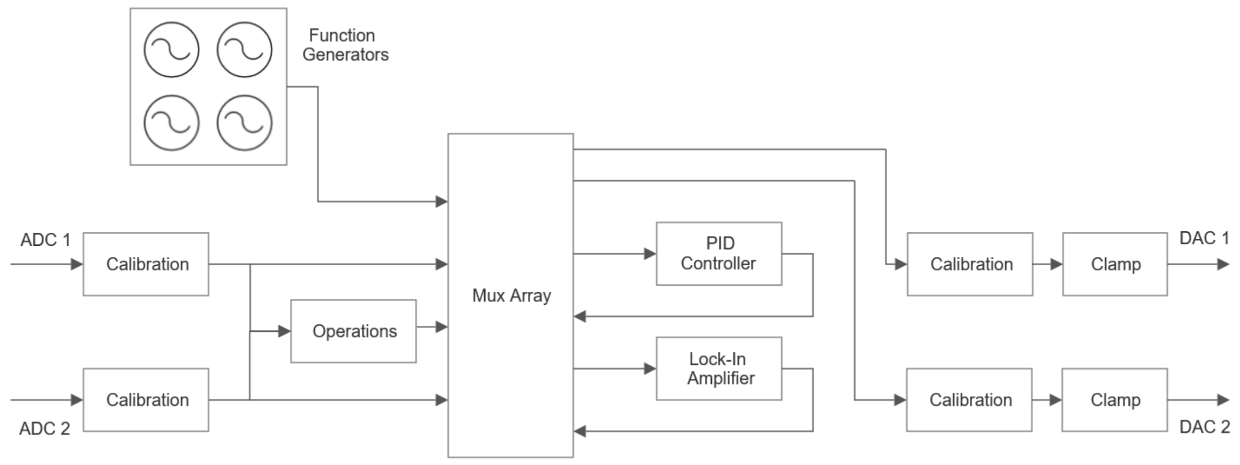


Figure 5: A simplified diagram of the modules and connections between them in this project. The multiplexer (mux) array takes the ADC inputs, the input operations, the oscillators, the lock-in amplifier, the PID controller and more as inputs, then outputs them to other modules or to the DAC as defined by the user. This allows for handling of potentially hundreds of combinations

4.2 Calibration

Input and output calibration for the ADCs and DACs are offered as a module. The module adds an offset to the incoming or outgoing data, then applies a gain to it. The offset and the gain are both user-defined. The module also has an enable/disable parameter; when disabled, the output is simply the uncalibrated input. Calibration values are stored on the Red Pitaya to easily set these parameters on startup.

To recalibrate the inputs, disconnect any voltage to the corresponding ADC and set the offset to the average of the input measurements. To calibrate the gain, set the input to a known voltage and monitor the input voltage reading. Adjust the gain until the average input voltage is the same as the known voltage.

To recalibrate the outputs, an external voltage measurement is required. Set the output voltage to zero then measure the actual output. Adjust the offset until the actual reading is zero. Then set the output voltage to some nonzero value and adjust the gain until the actual reading matches that value.

4.3 Input Operations

Directly after input calibration, the operations module calculates several operations of the two inputs. These are: the inverted values of each input, the sum of the inputs, and the difference of the inputs (in either order). These values are then outputted to multiplexers to be used as inputs for other modules or even to be outputted. These basic operations can help with signal processing (such as using the difference for saturated absorption spectroscopy), and other operations can easily be added as needed.

4.4 PID Controller

The PID controller module takes in a number of inputs. Primarily there is the feedback input measured from the system. There is also the setpoint, which can be either input, an operation of those inputs (see above), or a user-defined value. This is set via multiplexer. Other parameters are the proportional, integral, and derivative gains. Finally there is the minimum and maximum output values of the PID control, which is used for the integrator. The controller contains 5 blocks.

The first block calculates the error signal, which is simply the difference of the setpoint and the measured value. This error signal is then used as an input to the proportional, integral, and derivative blocks. Next, the proportional block takes the error signal and multiplies it by the proportional gain modifier. This value is then sent to the final summing block.

The integral block calculates the sum of the error signal multiplied by the integral gain at each clock tick. This is the discrete approximation of the integral up to a factor of the clock period. If the measured value begins far away from the setpoint, by the time the setpoint is reached, the integral output may overpower the other blocks in a phenomena known as integral windup. To deal with this, the integrator memory is capped at the amount corresponding to the maximum and minimum output of the PID control. This causes the integrator output to immediately change as the error signal crosses zero, allowing less overshoot and faster response times.

The derivative block calculates the difference of the current error signal and the error signal of the previous clock cycle. This is the discrete derivative up to a factor of the clock frequency.

The final block sums the three components and relegates the output to within its minimum and maximum values. This value should then be output to control the external system.

4.5 Function Generators

The function generator module is a necessity for the lock-in amplifier, and can also be sent directly to one of the outputs of the Red Pitaya. The function generator creates sine waves with a customizable frequency, phase, and amplitude. The sine wave is generated by using a table of sine wave values from the first quarter wavelength of a sine wave. This table is initialized to the FPGA RAM on startup. The function generator module's purpose is to choose which entry to output to create sine waves of various frequencies and phases. To do this, it has a counter that counts along the number of entries in the sine wave table, and a quadrant tracker that keeps track of what quarter of the sine wave to output. For example, the second quadrant requires the counter to refer to the opposite entry of the table as the sine wave is decreasing.

The frequency aspect is controlled by the period setting. The period is the number of clock cycles that the module should wait before iterating the counter. A period of 0 outputs the sine wave at full frequency, which a period of 1 outputs it at half frequency. In general, a period of n generates a sine wave at $1/(n+1)$ of the full frequency. It should be noted that this design has quite large steps between frequencies, especially near full frequency (e.g. if full frequency is 50 kHz, the next step is 25 kHz). At very low frequencies, the sine wave can also have noticeable steps due to the digitization of the sine wave.

To mitigate these issues, there are two types of function generators present in the design. The first is a low frequency module that has a memory file containing 625 points along the first quarter of a sine wave. This translates to a 2500 point sine wave. At full frequency, the 125 MHz clock outputs this as a 50 kHz sine wave. At much lower frequencies, this allows for many frequency options with good resolution. The other type is a high frequency module, whose memory file contains 31 points. The operation is the same as the low frequency module, resulting in a full frequency of 1 MHz. As needed, additional frequency ranges can be defined by altering the module to accommodate memory files with more or less points.

The phase of the sine wave is controlled by adding the phase to current value of the counter. If this causes the counter to exceed its maximum value, then this also alters the quadrant.

The memory files that generate the sine waves correspond to a sine wave of 2 Vpp, which is the maximum output of the Red Pitaya. Amplitude is handled as a multiplier to the end result, and thus always has magnitude less than or equal to one.

There are two low frequency and two high frequency function generator modules present in the design, allowing for two sine waves of the same type to be used simultaneously. For example, one wave can be sent to an output to modulate a system, and another wave of the same frequency but different phase can be used in the lock-in amplifier.

4.6 Low-Pass Filter

The low-pass filter module is a straightforward implementation of the discrete first-order low-pass filter. It takes as an input a user-defined integer shift parameter corresponding to the alpha value. The alpha value can be translated to cutoff frequency or RC value as discussed in the theory section.

4.7 Lock-In Amplifier

The lock-in amplifier module consists of a mixer, a third-order low-pass filter, and a scaling factor. The mixer multiplies the 14-bit input with the 14-bit reference sine wave. This results in a 28-bit number which is sent to the low-pass filters. By retaining 28 bits of accuracy rather than cutting down to the typical 14 bits, we can filter with a much lower cutoff frequency without losing resolution. In order to provide sufficient noise filtering, the data is passed through 3 of the low-pass filter modules, resulting in a third-order filter. Finally, the 28-bit signal needs to be reduced to 14-bit. However, the optimal method to do this is very dependent on the current setup and data. To avoid any unnecessary loss of information, a user-defined scaling factor is applied to the data. If the data saturates the ± 1 V range, reducing the scaling factor will help. Otherwise, increasing the scaling factor may improve resolution.

4.8 Output Clamp

The output clamp module is the final module that the output data passes through before the DAC. The module restricts the output to a user-defined range. If the input value exceeds the maximum, it is set to the maximum value. The opposite holds for the minimum value. This allows for low-voltage or asymmetrical applications.

5 Using the Red Pitaya

The simplest way to use the Red Pitaya is with the included remote interface, shown in Fig. 6. The interface is written in Python with the TkInter package and uses the Paramiko package to communicate with the Red Pitaya through secure shell (SSH). It allows enabling or disabling the board and setting/saving of all parameters. See the appendix for more information about parameters and their functions.

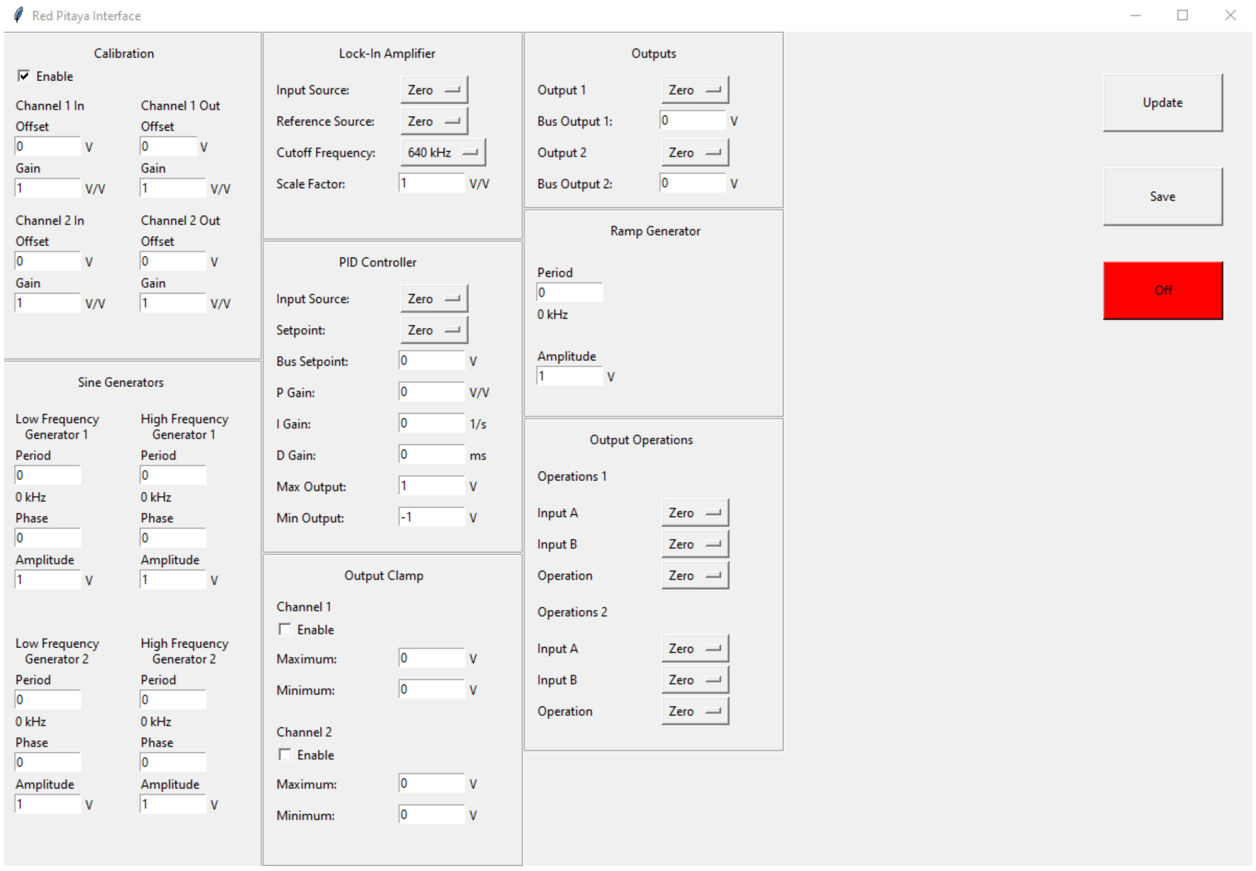
Upon starting the interface, it attempts to connect the Red Pitaya. If it can, it sets the FPGA to the bitstream on the Red Pitaya in the `/lock/` directory. It then attempts to read a memory file in `/lock/` on the board to initialize parameters. The interface allows saving of parameters between sessions by writing to this file. If the file is not found, it will set default parameters and create the file.

Further development on each layer of the project is possible and can help improve and customize functionality. An integrated synthesis environment (ISE) is required to synthesize the HDL to the bitstream that the FPGA uses. The Xilinx ISE WebPack is available for free and covers all necessary functionality. The Red Pitaya Master environment is available online with the basic FPGA connections and setup. The HDL modules included in this project are imported to the v0.94 HDL project. Only the `red_pitaya_top.sv` HDL file is modified from the Red Pitaya Master environment in order to integrate the new modules. To implement a design on the FPGA, first use the ISE to generate a bitstream. Transfer this bitstream to the

Red Pitaya and concatenate it to `/dev/xdevcfg`. To set the new bitstream as the default for the interface, replace the `red_pitaya_top.bit` in the `/lock/` directory with a bitstream of the same name.

The Red Pitaya can also be directly controlled using SSH. There are a number of command line utilities available for the Red Pitaya. In particular, the `monitor` utility allows reading/writing to the FPGA registers (as long as they are properly configured in the current bitstream). The FPGA registers used for the project (i.e. the parameters or the output data) are detailed in Appendix. Note that parameters are typically sent in hexadecimal.

Finally, the remote interface can be easily modified in Python. If adding new parameters, first ensure that they are properly set in the current bitstream. Add the TkInter widgets to accommodate setting these parameters, then set them to `monitor` the corresponding register when the update button is pressed using Paramiko.



Calibration
[✓] Enable
Channel 1 In — Offset: 0 V; Gain: 1 V/V
Channel 1 Out — Offset: 0 V; Gain: 1 V/V
Channel 2 In — Offset: 0 V; Gain: 1 V/V
Channel 2 Out — Offset: 0 V; Gain: 1 V/V

Sine Generators
Low Frequency Generator 1 — Period: 0; 0 kHz; Phase: 0; Amplitude: 1 V
High Frequency Generator 1 — Period: 0; 0 kHz; Phase: 0; Amplitude: 1 V
Low Frequency Generator 2 — Period: 0; 0 kHz; Phase: 0; Amplitude: 1 V
High Frequency Generator 2 — Period: 0; 0 kHz; Phase: 0; Amplitude: 1 V

Lock-In Amplifier
Input Source: Zero
Reference Source: Zero
Cutoff Frequency: 640 kHz
Scale Factor: 1 V/V

PID Controller
Input Source: Zero
Setpoint: Zero
Bus Setpoint: 0 V
P Gain: 0 V/V
I Gain: 0 1/s
D Gain: 0 ms
Max Output: 1 V
Min Output: -1 V

Output Clamp
Channel 1
[ ] Enable
Maximum: 0 V
Minimum: 0 V
Channel 2
[ ] Enable
Maximum: 0 V
Minimum: 0 V

Outputs
Output 1: Zero
Bus Output 1: 0 V
Output 2: Zero
Bus Output 2: 0 V

Ramp Generator
Period: 0; 0 kHz
Amplitude: 1 V

Output Operations
Operations 1
Input A: Zero
Input B: Zero
Operation: Zero
Operations 2
Input A: Zero
Input B: Zero
Operation: Zero

Figure 6: The interface used to control the Red Pitaya. It was created in Python using the TkInter package and communicates with the Red Pitaya via SSH using the Paramiko package. The red off button enables/disables the output. The update button sets the parameters entered in the interface to the FPGA. Finally, the save button saves the parameters currently entered for the next time using the interface.

6 PID Testing

To test the effectiveness of the FPGA-based implementation of the PID controller we used it as a temperature controller for an aluminum block. The full temperature control system can be seen in Fig. 7. The test system was an aluminum block heated by a Marlow RC3-2.5 thermoelectric cooler (TEC). The TEC alternatively cools or heats depending on the current it receives, making it an ideal actuator for a PID setup. The sensor was a TH10K thermistor on the block which, when part of a voltage divider, created a voltage signal correlated to the current temperature of the block. 40°C was the desired setpoint, which was converted into its voltage equivalent from the thermistor voltage divider. The output current of the Red Pitaya was insufficient to drive the TEC, so an operational amplifier was used to drive the TEC. The op amp used was the OPA548 evaluation module, which had many of the components previously installed. The gain was set to approximately 10 by switching a resistor on the evaluation board.

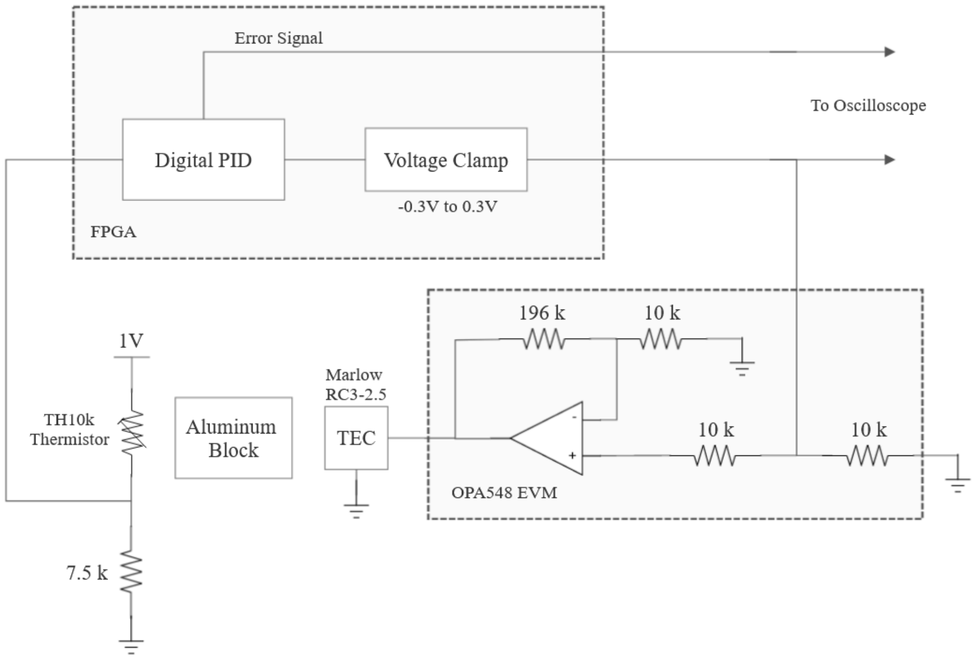


Figure 7: The experimental setup for temperature control of an aluminum block. The temperature sensor is a voltage divider with a thermistor, pictured to the left of the block. The FPGA acts as the PID controller and the OPA548 evaluation board is the driver for the TEC. The error signal and PID output are monitored with an oscilloscope.

The parameters of the PID controller were tuned to optimize the response of the system. In particular, reducing setpoint overshoot, a fast settling time, and long-term stability are important factors to be considered. To this end, parameters were tuned to their greatest values that did not cause the system to be unstable so as to improve responsiveness. From there they were tweaked to improve the response. The parameters found were:
Proportional Gain: 512
Integral Gain: $18.4s^{-1}$
Derivative Gain: 2.30×10^{-6} s

In order to provide a benchmark to compare the Red Pitaya with, a commercial temperature controller was applied to the same system.The controller was an Arroyo TECPak 585, using the same thermistor, TEC and aluminum block. The temperature data from each controllers was collected with a second thermistor placed near the first with resistance independently measured using a Tektronix DMM4050 precision multi-

meter. The Arroyo parameters were found using its AutoTune feature. They were:
Proportional Gain: 3.85
Integral Gain: $1.79 \times 10^{-2} s^{-1}$
Derivative Gain: 108 s

The response of the two systems was very similar. Fig. 8 shows the data for the two systems. Qualitatively, there were differences in the rise time (how long it takes to reach the setpoint) and the Red Pitaya slightly overshot the setpoint. However, these differences can be adjusted by modifying parameters. Both controllers displayed very stable temperature within just two minutes.

As for the long-term stability, the two systems also had very similar results. The RMS of the steady-state temperature for the Red Pitaya was 0.0059° C while the RMS of the Arroyo controller was 0.0080° C. Both are quite precise for temperature control.

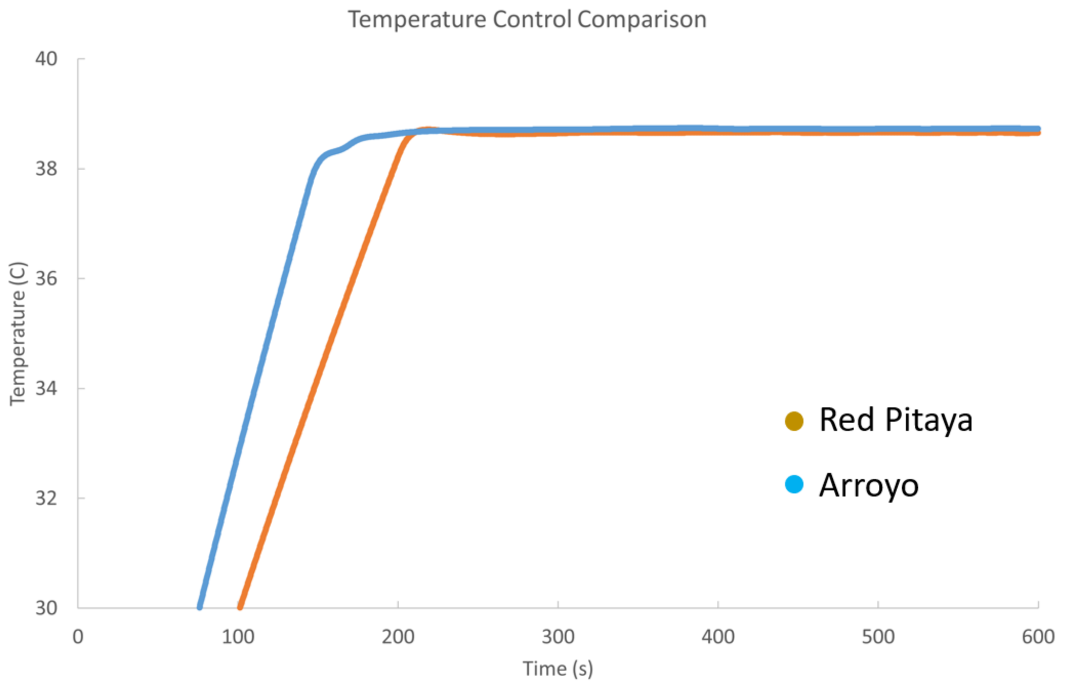


Figure 8: Temperature data for the comparison of the Red Pitaya and the Arroyo temperature controller. The TEC, thermistor, and aluminum block were the same for both controllers. The data was collected with an independent thermistor and a digital multimeter.

7 PDH Testing

The next test was applying the PDH technique with the Red Pitaya. The goal was to use the Red Pitaya on a complex and precise experiment, and show that it can be used to lock laser frequency. The purpose of the experiment was to lock the frequency of a laser to a resonant frequency of a rubidium sample. The experimental setup is shown in Fig. 9.

The laser is set up for saturated absorption spectroscopy of the rubidium gas cell, with the probe beam being detected by a photodetector. Not pictured in Fig. 9 is a second beam of the same intensity as the probe passing through the gas cell and to a photodetector. These photodetectors are connected to the inputs of the Red Pitaya. The difference of those voltages yields the narrow saturated absorption peaks as the laser scans across the resonant frequencies, as seen in Fig. 10. We want to lock the frequency of the laser at the tip of these peaks. To do so we will use the PDH technique.

The laser light is modulated by an electro-optic modulator (EOM) controlled by the Red Pitaya oscillator. An oscillator of the same frequency and adjustable phase is mixed with the absorption peaks then demodulated with the lock-in amplifier. The result is seen in Fig. 11. The phase should be tuned to maximize this signal (i.e. in phase). The error signal contains the linear error signal we need for the PID controller to lock the laser frequency on the resonance. Since this error signal is a derivative-like signal of the absorption peaks, we want to lock onto the frequency at which that error signal is zero to be at the resonant frequencies.

When trying to control the laser frequency with the error signal in Fig. 11, it is important to stay within the linear region. If the frequency escapes this region, the PID will not be able to tune it back to resonance. Thus it is important to have a responsive and stable controller for the laser frequency. As of writing this, initial testing produced the error signal seen in Fig. 11 and that signal was used to lock the laser frequency using an external PID controller. Unsuccessful attempts were made using the Red Pitaya PID controller to lock the laser frequency, but tuning and testing is ongoing.

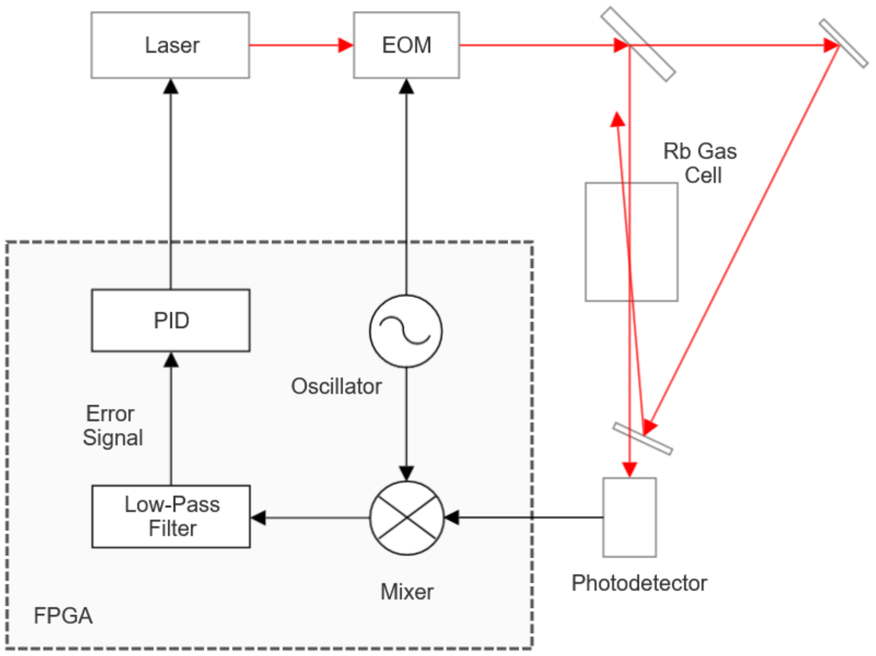


Figure 9: FM spectroscopy experimental setup. The laser's output is phase-modulated by an electro-optic modulator (EOM) controlled by the Red Pitaya oscillator. The resultant light is set on the rubidium gas cell as described in the theory section. The result is measured by a photodetector and demodulated within the Red Pitaya. Finally, the demodulated error signal is used by the PID controller to fix the laser frequency at resonance.

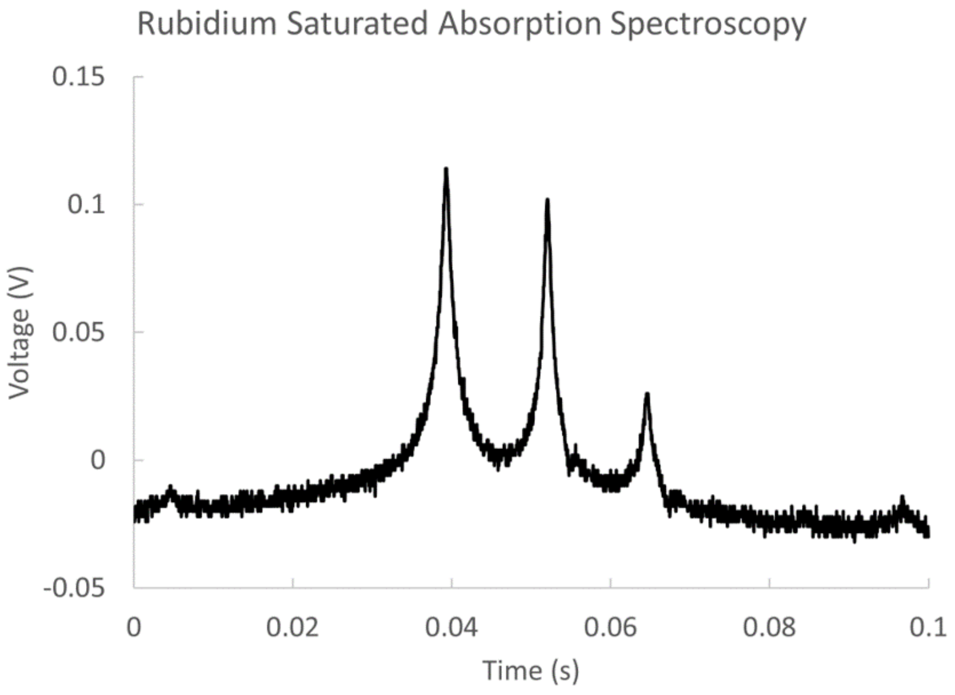


Figure 10: The saturated absorption peaks of Rubidium. The data was recorded with an oscilloscope monitoring the difference of the photodetector outputs. The narrow peaks at resonant frequencies are clearly visible.

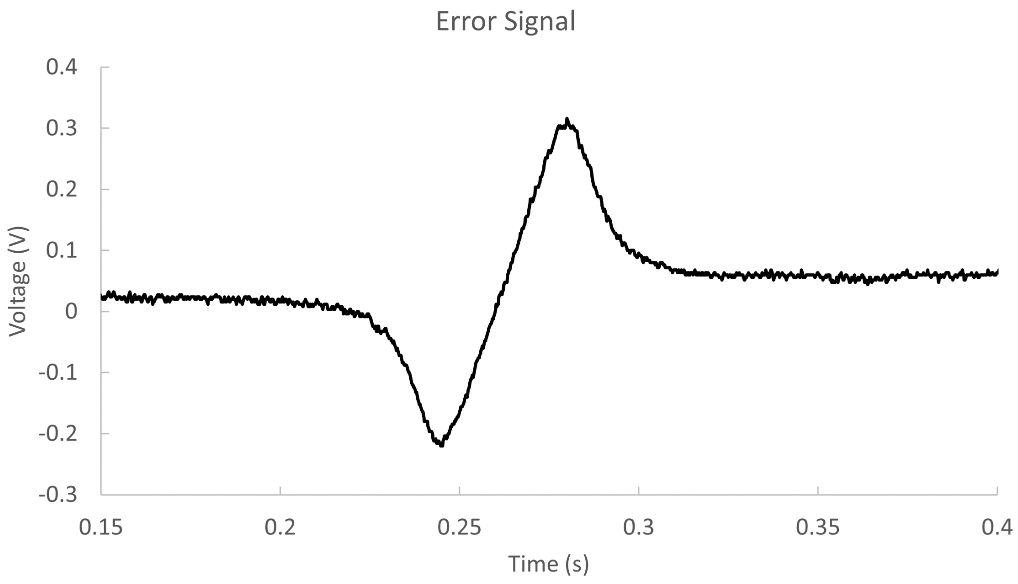


Figure 11: The error signal extracted from the Rubidium spectroscopy. The data was recorded with an oscilloscope from the output of the FPGA. The linear region is clearly visible, corresponding to a resonant frequency in the Rubidium spectrum. This region is suitable for controlling the laser frequency with a PID controller.

8 Conclusion

The digital implementation of a PID controller, oscillators, and a lock-in amplifier on the Red Pitaya provides a cost-effective and powerful alternative to dedicated devices. The reprogrammability of the FPGA means the system can be optimized and adapted for a variety of other uses.

Further work on the system can include supporting software, such as re-locking programs or automatic parameter tuning. Other possible features are high-speed data acquisition, touchscreen interface, recreations of other laboratory instruments, and more.

9 References

[1]J. I. Thorpe, K. Numata, and J. Livas, "Laser frequency stabilization and control through offset sideband locking to optical cavities," Opt. Express **16**, 15980-15990 (2008).

[2]A. Schwettmann, J. Sedlacek, and J. P. Shaffer, "Field-programmable gate array based locking circuit for external cavity diode laser frequency stabilization," Rev. Sci. Instrum. **82**, 103103 (2011).

[3]J. Bechhoefer, "Feedback for physicists: A tutorial essay on control," Rev. Mod. Phys. **77**, 783 (2005).

[4]E. D. Black, "An introduction to Pound-Drever-Hall laser frequency stabilization," American Journal of Physics **69**, 79 (2001).

[5]D. W. Preston, "Doppler-free saturated absorption: Laser spectroscopy," American Journal of Physics **64**, 1432 (1996).

10 Appendix

Name	Description	Permissible Values	Memory Registers
Calibration Module			
Enable	Enables or disables the calibration.		04, 10, 1C, 28
Offset	The amount subtracted from the input for each channel.	[-1, 1] V	08, 14, 20, 2C
Gain	The gain applied to the input after the offset.	[-8, 8] V/V	0C, 18, 24, 30
Sine Generators Module			
Period	Frequency control parameter. Represents clock cycles per value. Actual frequency is displayed below the text box.	0, 1, 2, ... 63999	34, 40, 4C, 58
Phase	Phase in degrees.	[0, 360)	38, 44, 50, 5C
Amplitude	Amplitude of the sine wave.	[-1, 1] V	3C, 4C, 54, 60
Ramp Generator Module			
Period	Frequency control parameter. Represents clock cycles per value. Actual frequency is displayed below the text box.	0, 1, 2, ... 63999	64
Amplitude	Amplitude of the ramp.	[-1, 1] V	68
Lock-In Amplifier Module			
Input Source	Choose the input for the lock-in amplifier.		A0
Reference Source	Choose the reference modulation.		A4
Cutoff Frequency	The cutoff frequency for the third-order low-pass filter.		6C
Scale Factor	Scale factor applied to the lock-in output. If the signal is saturated, try reducing this factor.		70
PID Controller Module			
Input Source	Choose the input for the PID controller.		A8
Setpoint	Choose the setpoint source. Bus Setpoint allows setting a value for the setpoint in the next parameter.		AC
Bus Setpoint	If Bus Setpoint is selected for the Setpoint, this value will be the setpoint.	[-1, 1] V	D0
P Gain	Proportional gain parameter.	[-4096, 4096] V/V	74
I Gain	Integral gain parameter.	[-235, 235] 1/s	78
D Gain	Derivative gain parameter.	[-8, 8] ms	7C
Max/Min Output	PID output range. Caps the integrator memory as well.	[-1, 1] V	80, 84
Output Clamp Module			
Enable	Enable or disable each clamp. Disabled clamp has an output range of -1 V to 1V.		88, 94
Maximum/Minimum	Defines allowed output range.	[-1, 1] V	8C, 90, 98, 9C
Output Operations Module			
Inputs	Define the two inputs to be used for the operation.		B0, B4, B8, BC
Operations	Choose the type of operation on those inputs.		C0, C4

Outputs Module			
Outputs	Choose the source of the ouput to the DAC. Bus Ouput 1/2 allows setting a value for the output in the next parameter.		C8, CC
Bus Ouput	If Bus Output 1/2 is selected for the Output, this value will be the output.	[-1, 1] V	D4, D8
Read-Only Data			
Inputs	ADC inputs after calibration.		E4, E8
Ouputs	Outputs to DAC.		DC, E0
PID Error	Error signal from PID.		EC