Modified PID & An Easy Way to Code FPGA using PyRPL

Wanlin Wang
E-mail: wwlyn@mit.edu
wwlyn@outlook.com

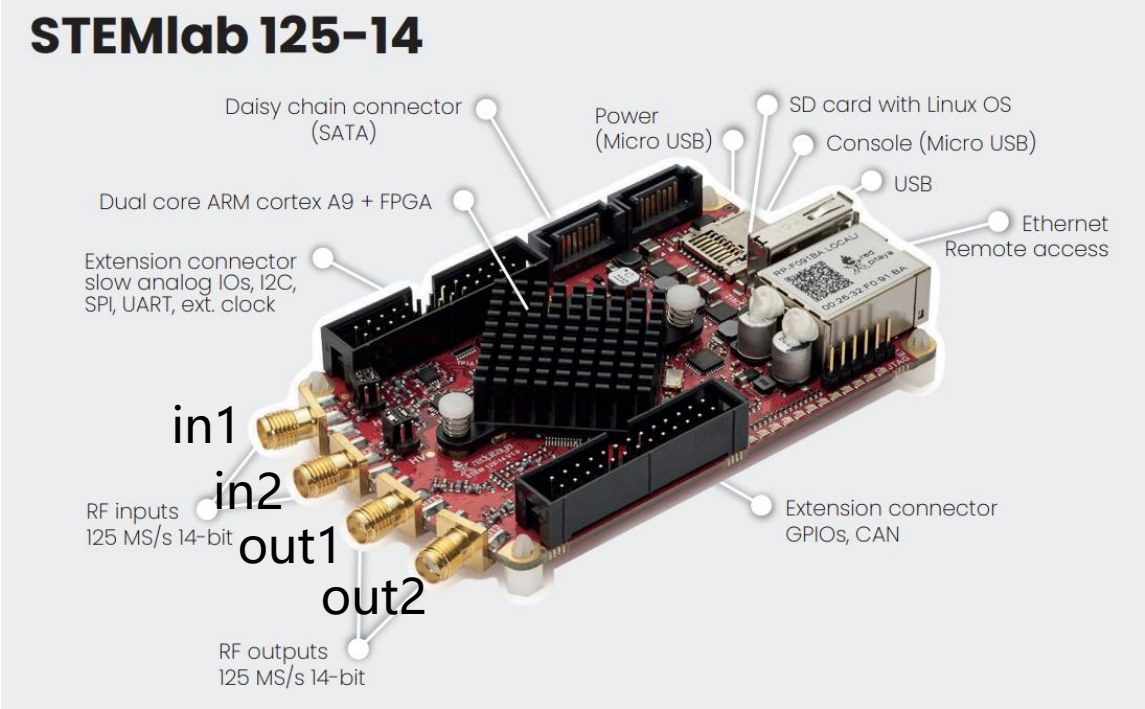
Feel free to ask questions!
More details are in the documentation.

Documentation: pyrpl_change/Documentation.md at max_hold_no_iir_improvement · wwlyn/pyrpl_change
PyRPL_change: wwlyn/pyrpl_change
Labscript: wwlyn/red_pitaya_pyrpl_pid
Official Documentation for RP: https://redpitaya.com/rtd-iframe/?iframe=https://redpitaya.readthedocs.io/en/latest/quickStart/needs.html

Outline

- An introduction to RedPitaya & PyRPL
- Modified PyRPL PID
  - 4 modifications
  - Installation, package conflict fix and basic usage
  - Labscript
- The Workflow of FPGA coding using PyRPL
  - Use digital setpoint sequence as an example
- Outlook on RedPitaya: A budget Quantum Machine?

An introduction to RedPitaya & PyRPL



The components of STEMlab 125-14

The website of RedPitaya has many functions, like oscilloscope, Vector Network Analyzer…

An introduction to RedPitaya & PyRPL

Before using RedPitaya: check the hardware feature! Eg. STEM 125-14

RF inputs	
RF input channels	2
Sample rate	125 MS/s
ADC resolution	14 bit
Input impedance	1 MΩ / 10 pF
Full scale voltage range	±1 V (LV) and ±20 V (HV)
Input coupling	DC
Absolute max. Input voltage	**LV ±6 V** **HV ±30 V**
Input ESD protection	Yes
Overload protection	Protection diodes
Bandwidth	DC - 60 MHz
Connector type	SMA

Input Modes: HV vs LV

Mode	Range	Resolution
LV	±1V	2V/2^14=0.122 mV
HV	±20V	40V/2^14=2.44 mV

An introduction to RedPitaya & PyRPL

Before using RedPitaya: check the hardware feature! Eg. STEM 125-14

RF inputs	
RF input channels	2
Sample rate	125 MS/s
ADC resolution	14 bit
Input impedance	1 MΩ / 10 pF
Full scale voltage range	±1 V (LV) and ±20 V (HV)
Input coupling	DC
Absolute max. Input voltage	**LV ±6 V HV ±30 V**
Input ESD protection	Yes
Overload protection	Protection diodes
Bandwidth	DC - 60 MHz
Connector type	SMA

RF outputs	
RF output channels	2
Sample rate	125 MS/s
DAC resolution	14 bit
Load impedance	50 Ω
Voltage range	±1 V
Short circuit protection	Yes
Output slew rate	2 V / 10 ns
Bandwidth	DC - 50 MHz
Connector type	SMA

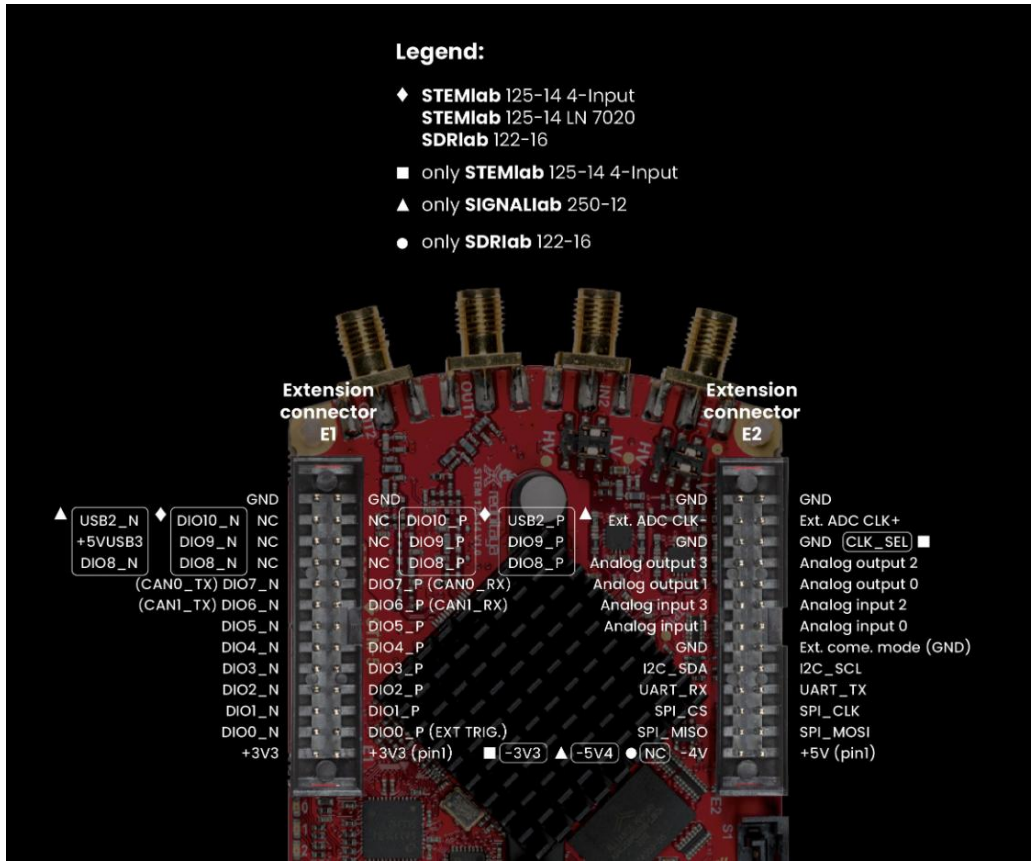
Input Modes: HV vs LV

Mode	Range	Resolution
LV	±1V	2V/2^14=0.122 mV
HV	±20V	40V/2^14=2.44 mV

How to improve DAC noise performance of STEM 125-14:
https://ln1985blog.wordpress.com/2016/02/07/red-pitaya-dac-performance/

An introduction to RedPitaya & PyRPL

Before using RedPitaya: check the hardware feature! Eg. STEM 125-14



Extension connector	
Digital IOs	16
Digital voltage levels	3.3 V
Analog inputs	4
Analog input voltage range	0 – 3.5 V
Analog input resolution	12 bit
Analog input sample rate	100 kS/s
Analog outputs	4
Analog output voltage range	0 – 1.8 V
Analog output resolution	8 bit
Analog output sample rate	≲ 3.2 MS/s
Analog output bandwidth	≈ 160 kHz
Communication interfaces	I2C, SPI, UART, CAN
Available voltages	+5 V, +3V3, -4 V
External ADC clock	No [2]

The extension module is not only for trigger, but…

An introduction to RedPitaya & PyRPL

README | Code of conduct | MIT license

PyRPL =
Python user interface + FPGA bottom !

build passing | build failing | codecov 12% | python 2.7 | 3.4 | 3.5 | 3.6 | pypi v0.9.3.6 | downloads 23k | docs failing
gitter join chat | license GPLv3

Download Now
SourceForge - Trusted for Open Source

PyRPL (Python RedPitaya Lockbox) turns your RedPitaya into a powerful DSP device, especially suitable as a digital lockbox and measurement device in quantum optics experiments.

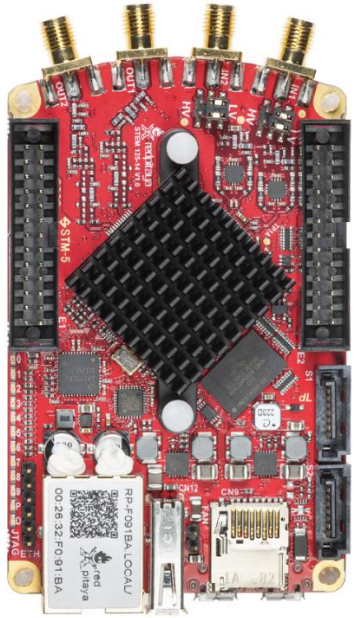
An introduction to RedPitaya & PyRPL

Cheap when we want fast, user-friendly feedback!

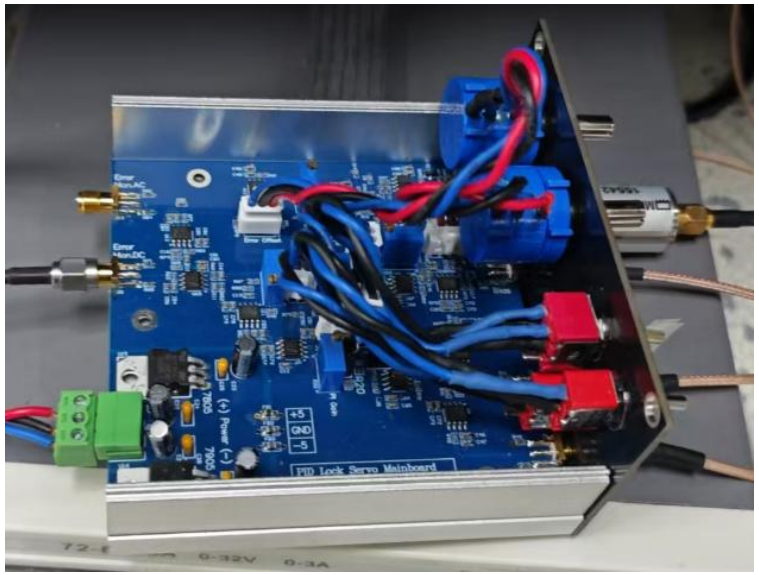
- **If we only need user-friendly, Raspberry Pico, stm32… is good and cheaper options;**
- **If we need fast, analog PID is faster and cheaper.**

- **And for fast fancy feedback.**

VS



VS



Modified PyRPL PID: Basic Usage

Basic Usage:

RedPitaya implements standard PID control:

$$\text{Output}(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

where $e(t) = \text{input} - \text{setpoint}$.

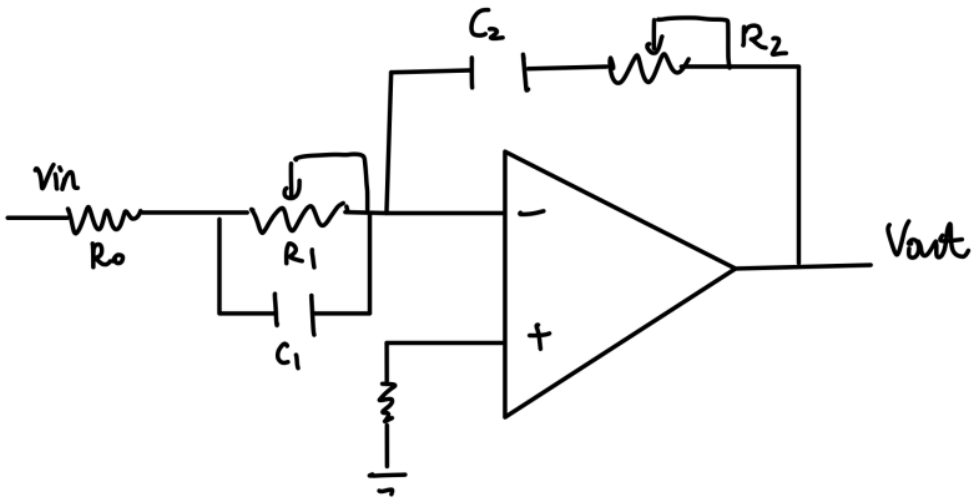
Two Basic Operation Modes:

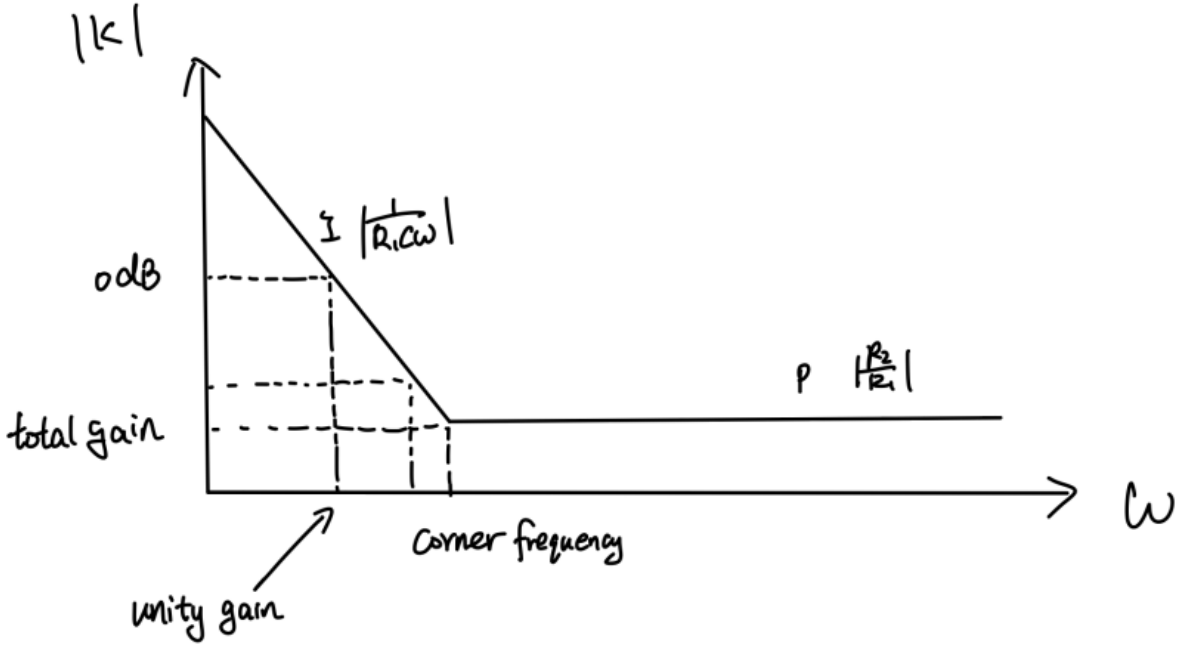
Mode	Configuration
Digital Setpoint	Single PID module
Analog Setpoint	Dummy module + functional PID (differential mode)

Modified PyRPL PID: Improve the bandwidth of PID

Improve the bandwidth of PID

- **Problem:** PyRPL limits **Ki** to 38kHz (unity gain frequency when Kp, Kd=0), restricting PID bandwidth.
- **Cause:** Limited GAINBITS (24 bits) for Ki register.
- **Solution:** Increase GAINBITS by 1 bit → doubles Ki upper limit.





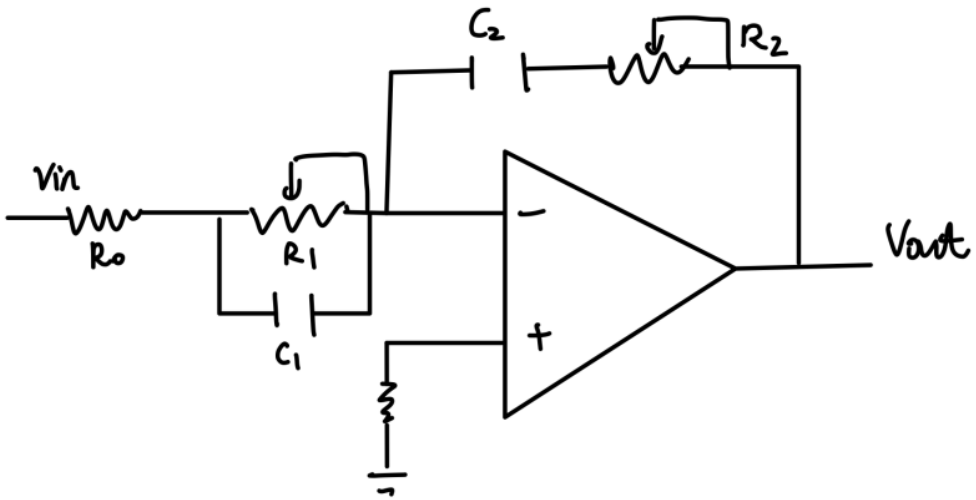
$$K(j\omega) = -\frac{R_2}{R_1}\left(\frac{C_1}{C_2}\frac{R_1}{R_2} + \frac{1}{j\omega R_2 C_2} + j\omega C_1 R_1\right)$$

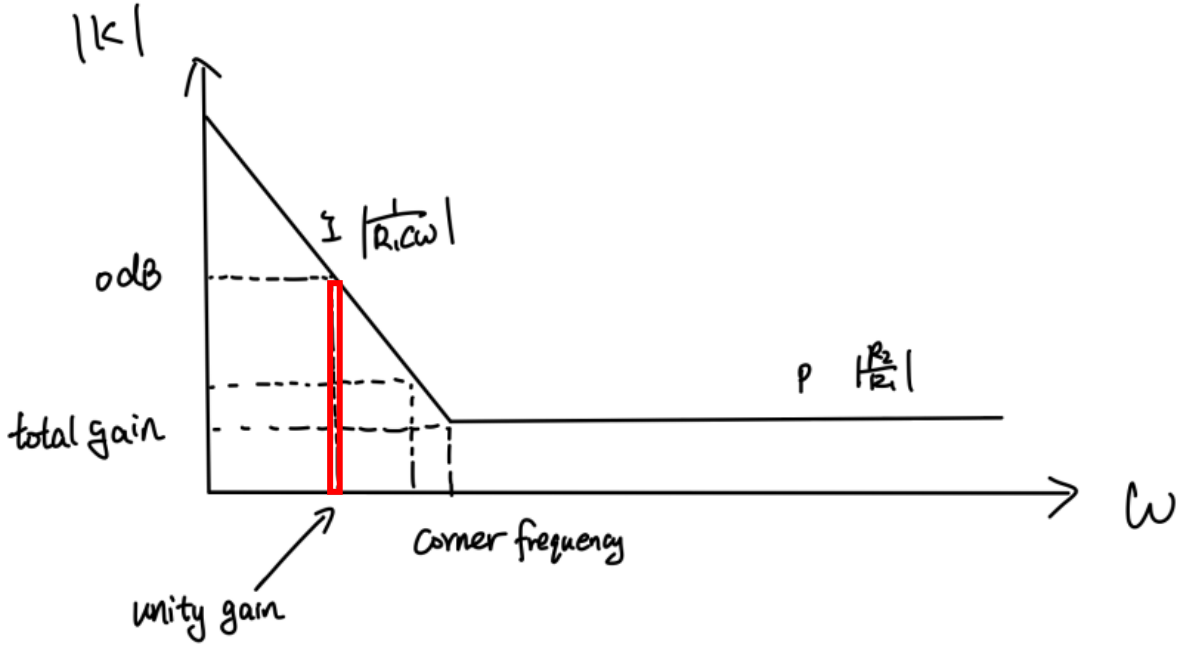
$$\text{Output}(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

Modified PyRPL PID: Improve the bandwidth of PID

Improve the bandwidth of PID

- **Problem:** PyRPL limits **Ki** to 38kHz (unity gain frequency when Kp, Kd=0), restricting PID bandwidth.
- **Cause:** Limited GAINBITS (24 bits) for Ki register.
- **Solution:** Increase GAINBITS by 1 bit → doubles Ki upper limit. Binary!





$$K(j\omega) = -\frac{R_2}{R_1}\left(\frac{C_1}{C_2}\frac{R_1}{R_2} + \frac{1}{j\omega R_2 C_2} + j\omega C_1 R_1\right)$$

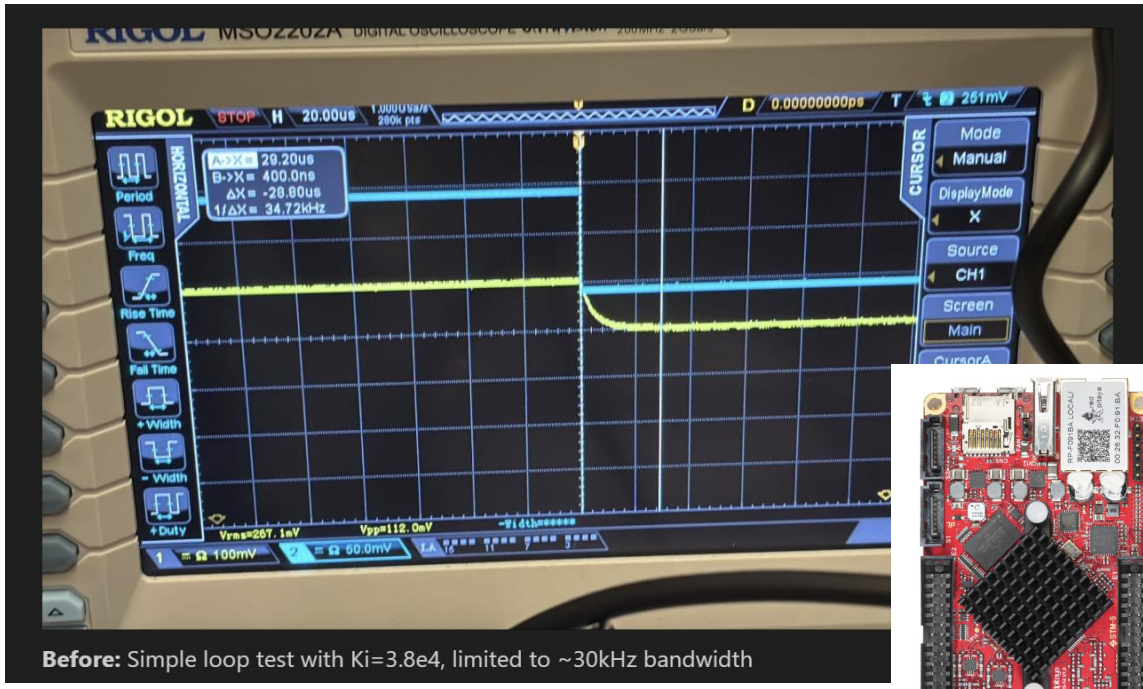
=1 (0dB)

$$\text{Output}(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

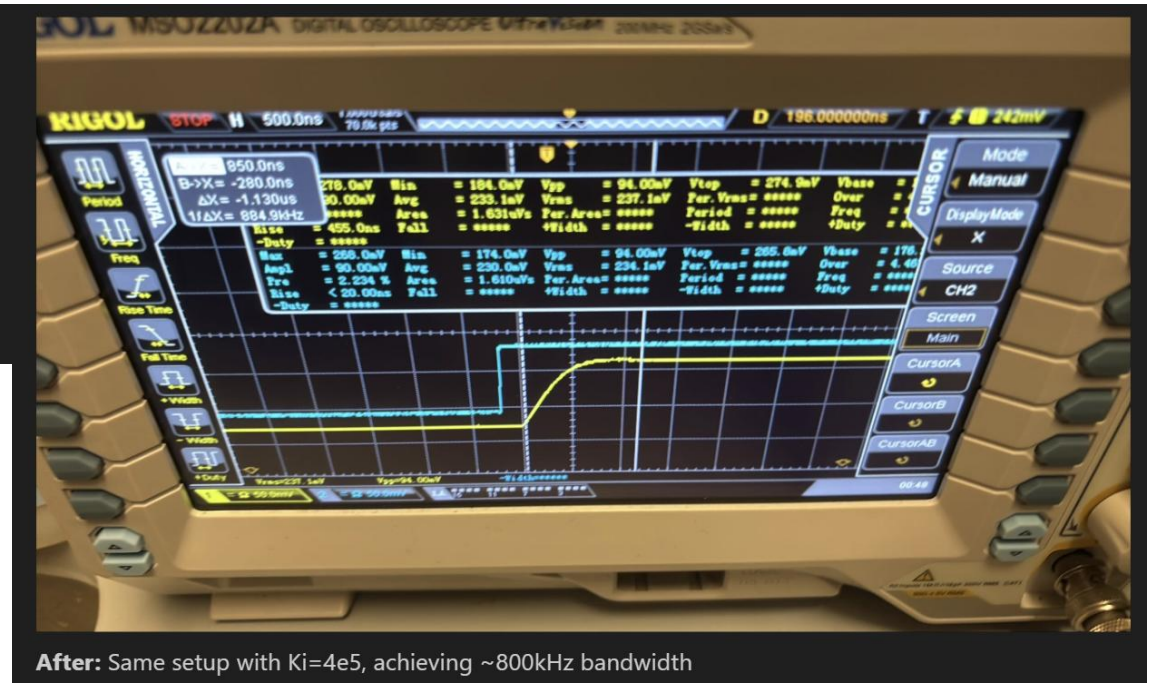
Modified PyRPL PID: Improve the bandwidth of PID

Improve the bandwidth of PID

- **Problem:** PyRPL limits **Ki** to 38kHz (unity gain frequency when Kp, Kd=0), restricting PID bandwidth.
- **Cause:** Limited GAINBITS (24 bits) for Ki register.
- **Solution:** Increase GAINBITS by 1 bit → doubles Ki upper limit.



Before: Simple loop test with Ki=3.8e4, limited to ~30kHz bandwidth

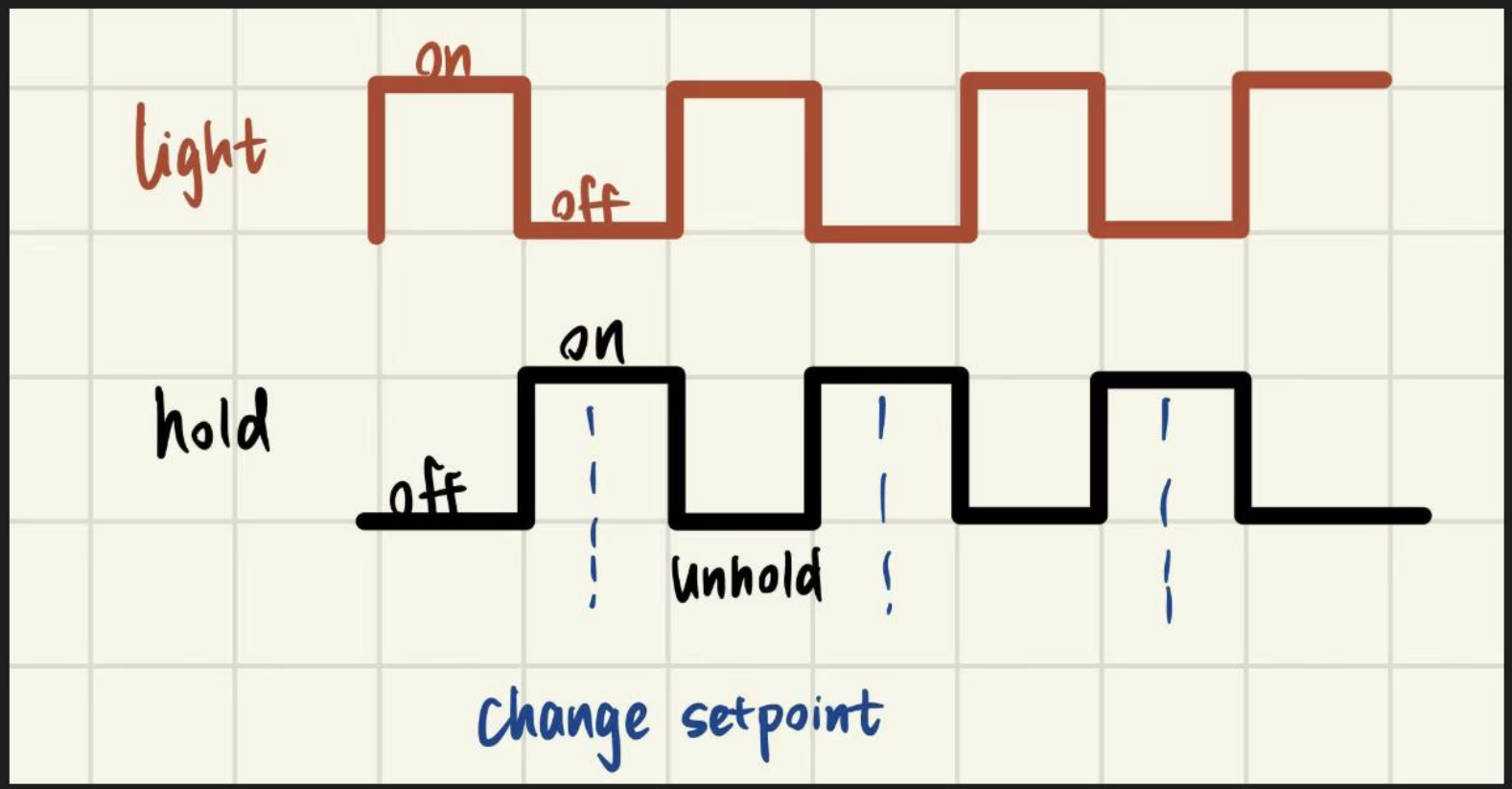


After: Same setup with Ki=4e5, achieving ~800kHz bandwidth

Modified PyRPL PID: Hold using external trigger

1. Hold using external trigger
2. Keep the last output when hold

Already have in `max_hold_no_iir_improvement` branch?

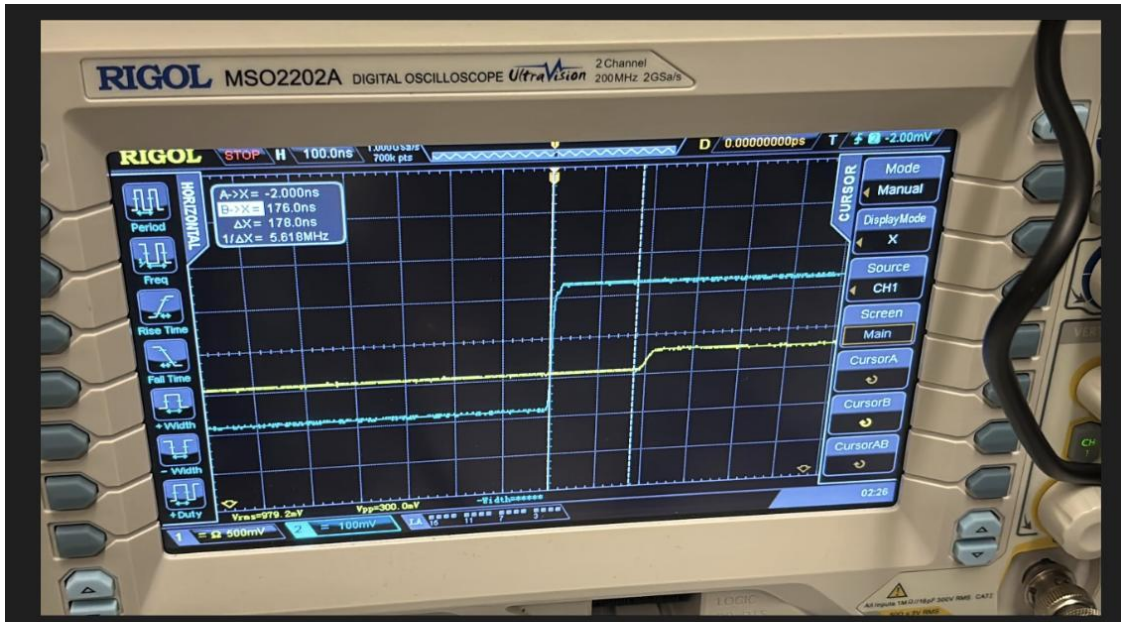


When the light is off, we don't want the lockbox to work otherwise if there is DC signal, it will saturate. And we also want to change the setpoint during the hold and when unhold, the lockbox will lock to another setpoint.

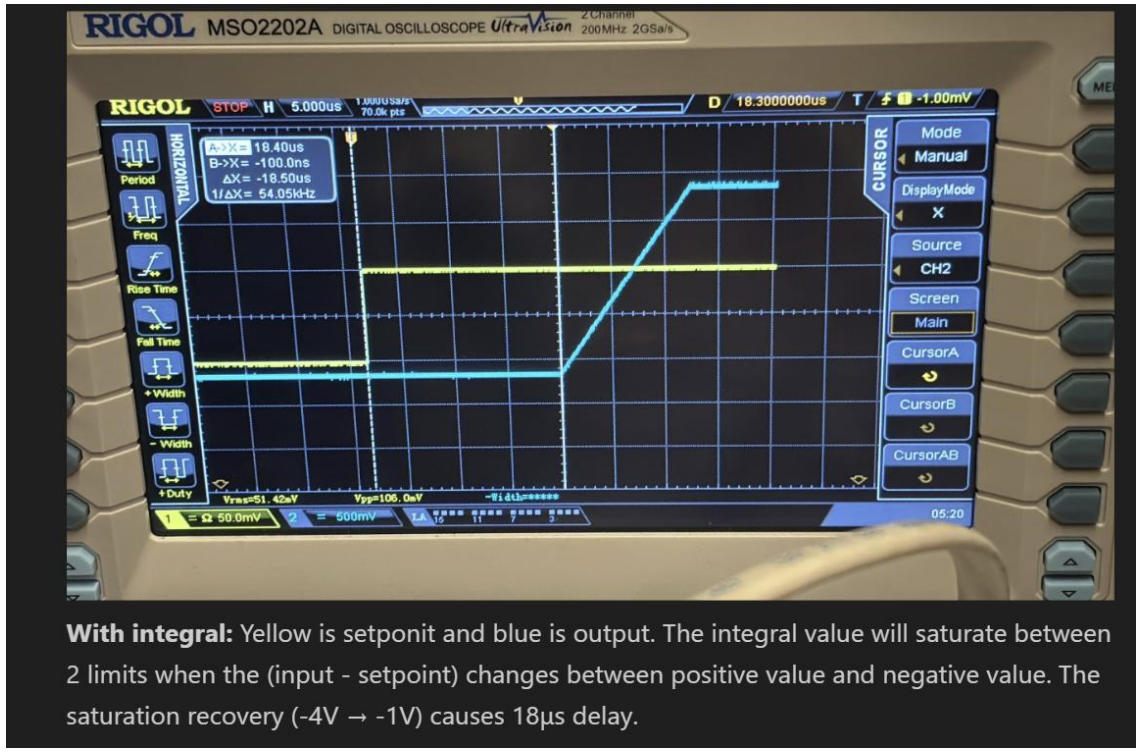
Modified PyRPL PID: Hold using external trigger

Hold using external trigger

- **Problem:** When unhold, we want PID lock to another setpoint, but it cannot function well.
- **Root Cause:** Integral range [-4V,+4V] exceeds output limits [-1V,+1V], causing delayed response in hold/unhold sequences.
- **Solution:** Change the integral range to [-1V,+1V] while keep the caculation resolution.



With Proportional only: Blue is input and yellow is output. The output will just be Kp * input. Pure proportional control shows ~180ns delay time



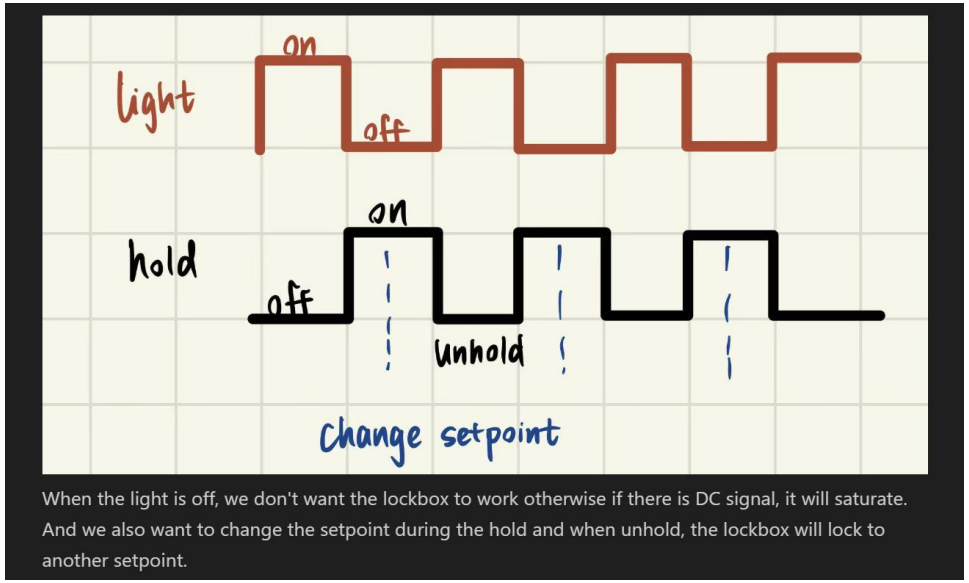
With integral: Yellow is setponit and blue is output. The integral value will saturate between 2 limits when the (input - setpoint) changes between positive value and negative value. The saturation recovery (-4V → -1V) causes 18µs delay.

Modified PyRPL PID: Hold using external trigger

Keep the last output when hold

- It just set Kp=0, keep the integral term. Works for small Kp and stable locks (error ≈ 0), but causes dramatic output jumps when:
- Using high Kp (bandwidth-limited systems);
- Error signals have large fluctuations

$$\text{Output}(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$



When the light is off, we don't want the lockbox to work otherwise if there is DC signal, it will saturate. And we also want to change the setpoint during the hold and when unhold, the lockbox will lock to another setpoint.

Modified PyRPL PID: Digital Setpoint Sequence

Use external trigger to step digital setpoint in a pre-loaded sequence
(if the setpoint is not continuously changing)

Why digital over analog setpoint sequences?

Issue	Analog Setpoint	Digital Setpoint
Noise	~20mV AC noise (no impedance matching)	Fixed digital values
Stability	Oscillates with certain P&I values	No oscillation issues at the same P&I, can push bandwidth further
ADC/DAC Noise	Uses both (2× noise)	Uses DAC only (1× noise)

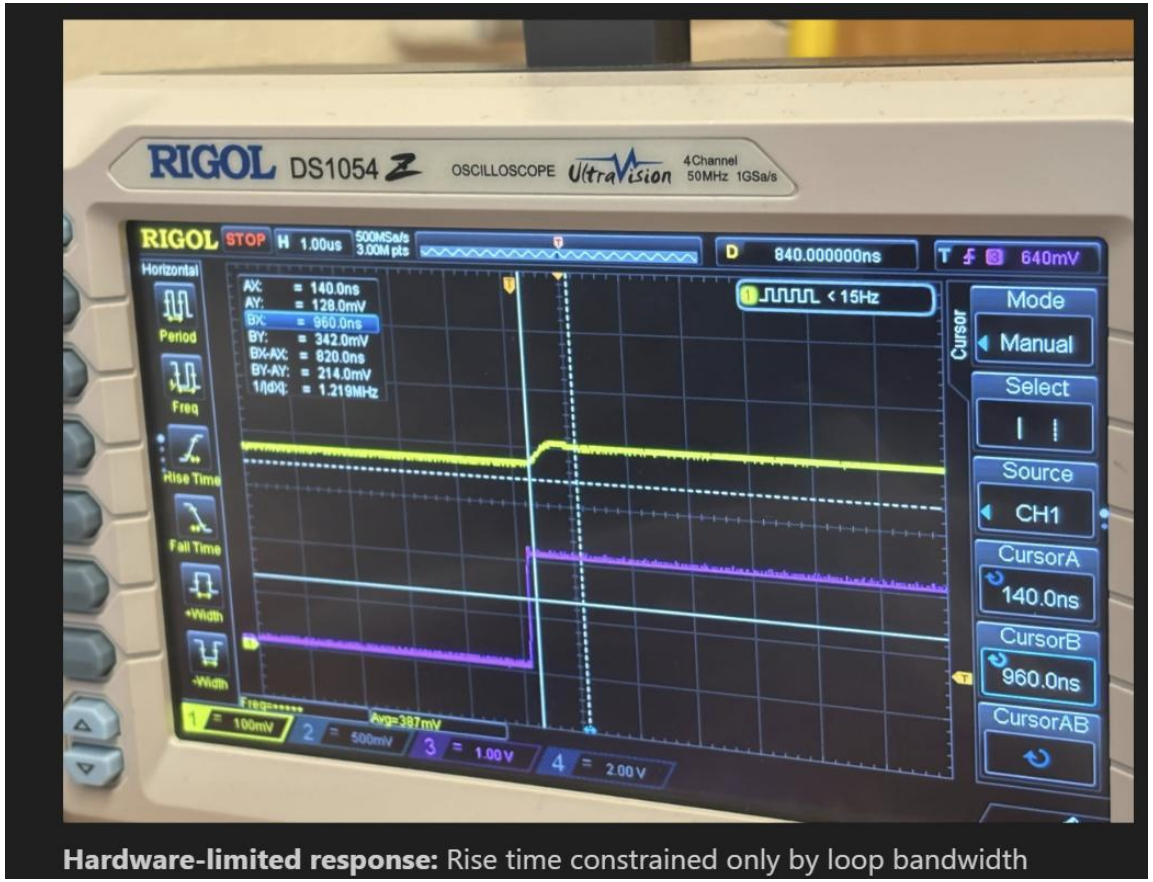
Implementation: Pre-loaded digital setpoint sequence with external rise detection.

Modified PyRPL PID: Digital Setpoint Sequence

Use external trigger to step digital setpoint in a sequence

Tested in the simplest PID loop which just connects input and output of RP.

Digital sequence operation: Purple = external trigger, Yellow = PID input(=output)



Hardware-limited response: Rise time constrained only by loop bandwidth

Later use this as the example of FPGA coding workflow!

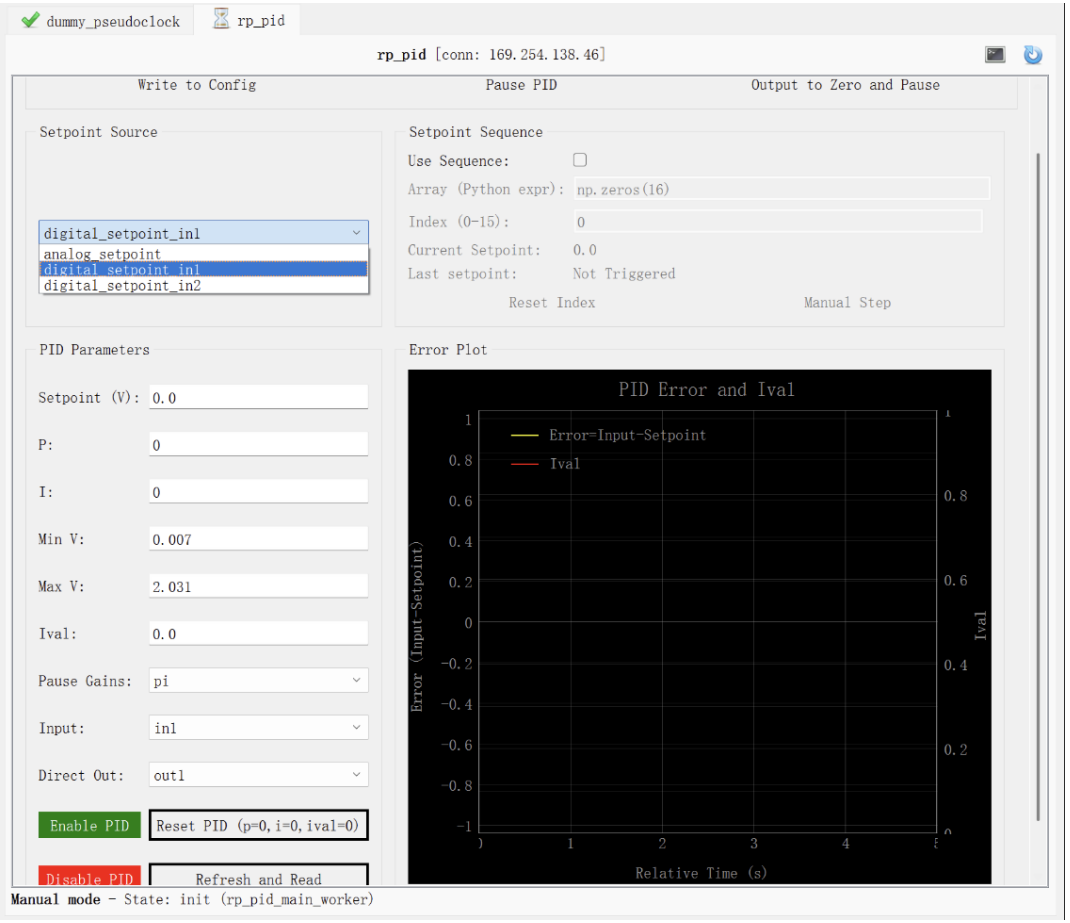
Modified PyRPL PID: Demonstration

Installation, package conflict fix (on python 3.9) and basic usage

- **Directly install my modified PyRPL**
- **Explore other interesting branch of original PyRPL**

Labscript

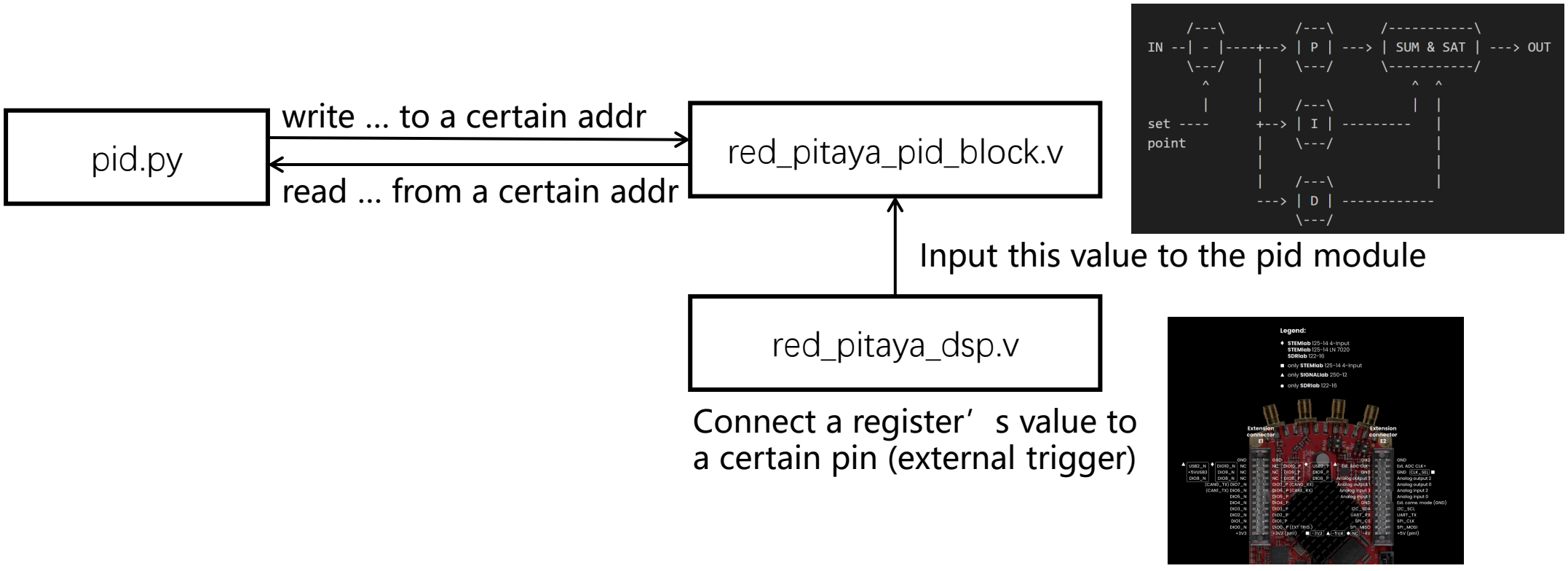
- **Blacs**
- ***Run 2 blacs at the same time**
  (a little bit tricky, details are in the documentation)



The Workflow of FPGA coding using PyRPL: Demonstration

Use external trigger to change digital setpoint covers:

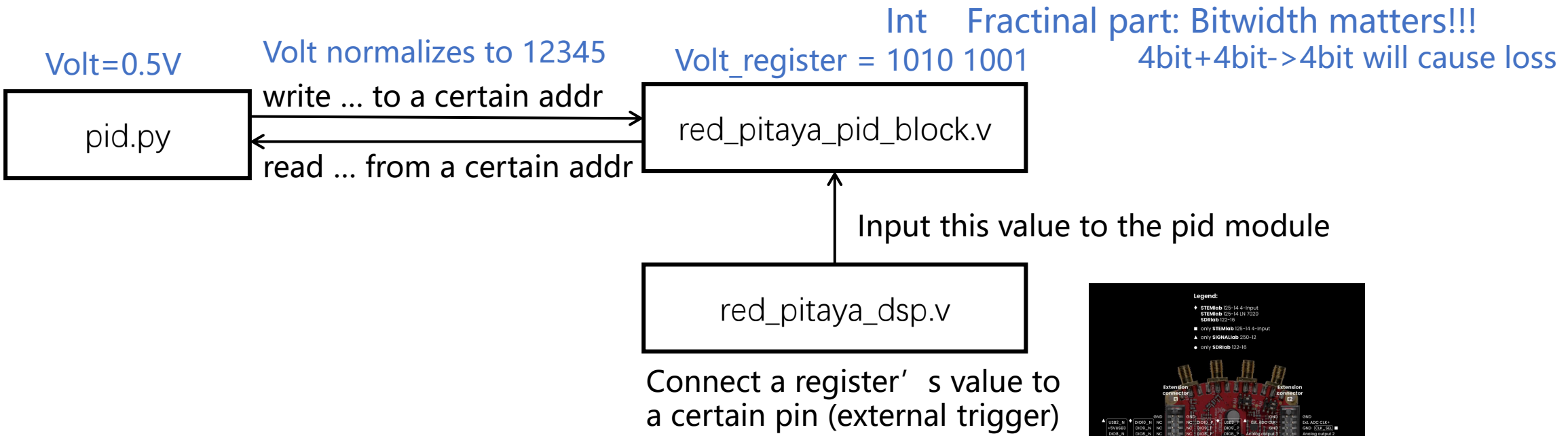
- **External trigger connection** **--red_pitaya_dsp.v**
- **Test the rise of the trigger** **--red_pitaya_pid_block.v**
- **Write and read certain registers (including basic value types & array)** **--pid.py & red_pitaya_pid_block.v**
- **Calculation logic of FPGA (a kind of decision tree)** **--red_pitaya_pid_block.v**



The Workflow of FPGA coding using PyRPL: Demonstration

Use external trigger to change digital setpoint covers:

- **External trigger connection** **--red_pitaya_dsp.v**
- **Test the rise of the trigger** **--red_pitaya_pid_block.v**
- **Write and read certain registers (including basic value types & array)** **--pid.py & red_pitaya_pid_block.v**
- **Calculation logic of FPGA (a kind of decision tree)** **--red_pitaya_pid_block.v**

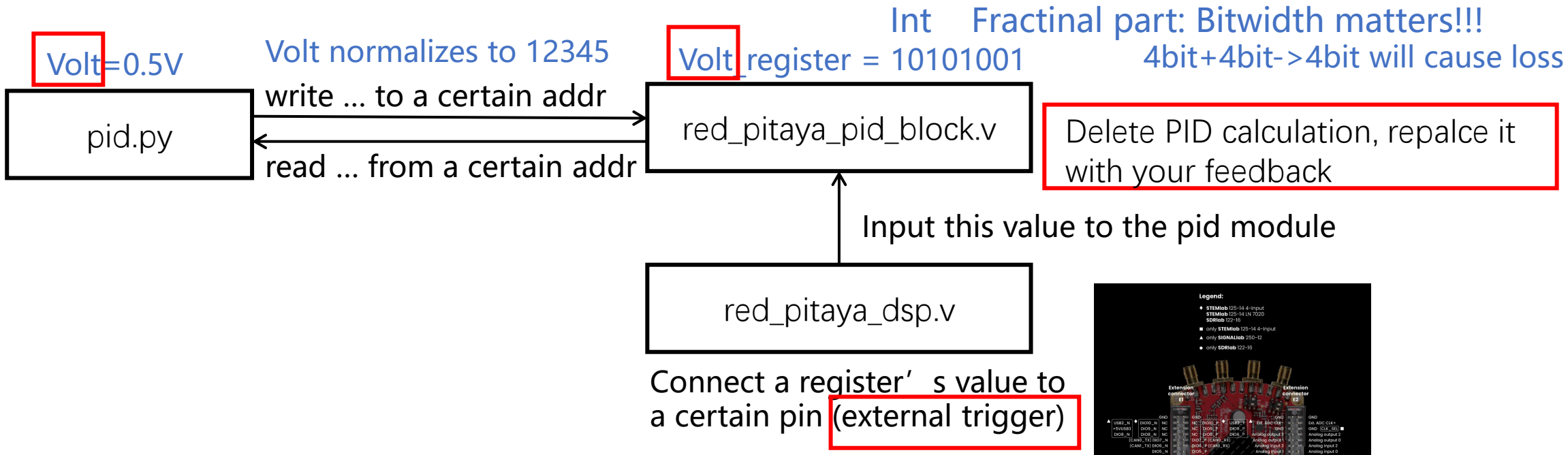


Built-in AI recommended!!!

Outlook on RedPitaya: A budget Quantum Machine?

Easy to code

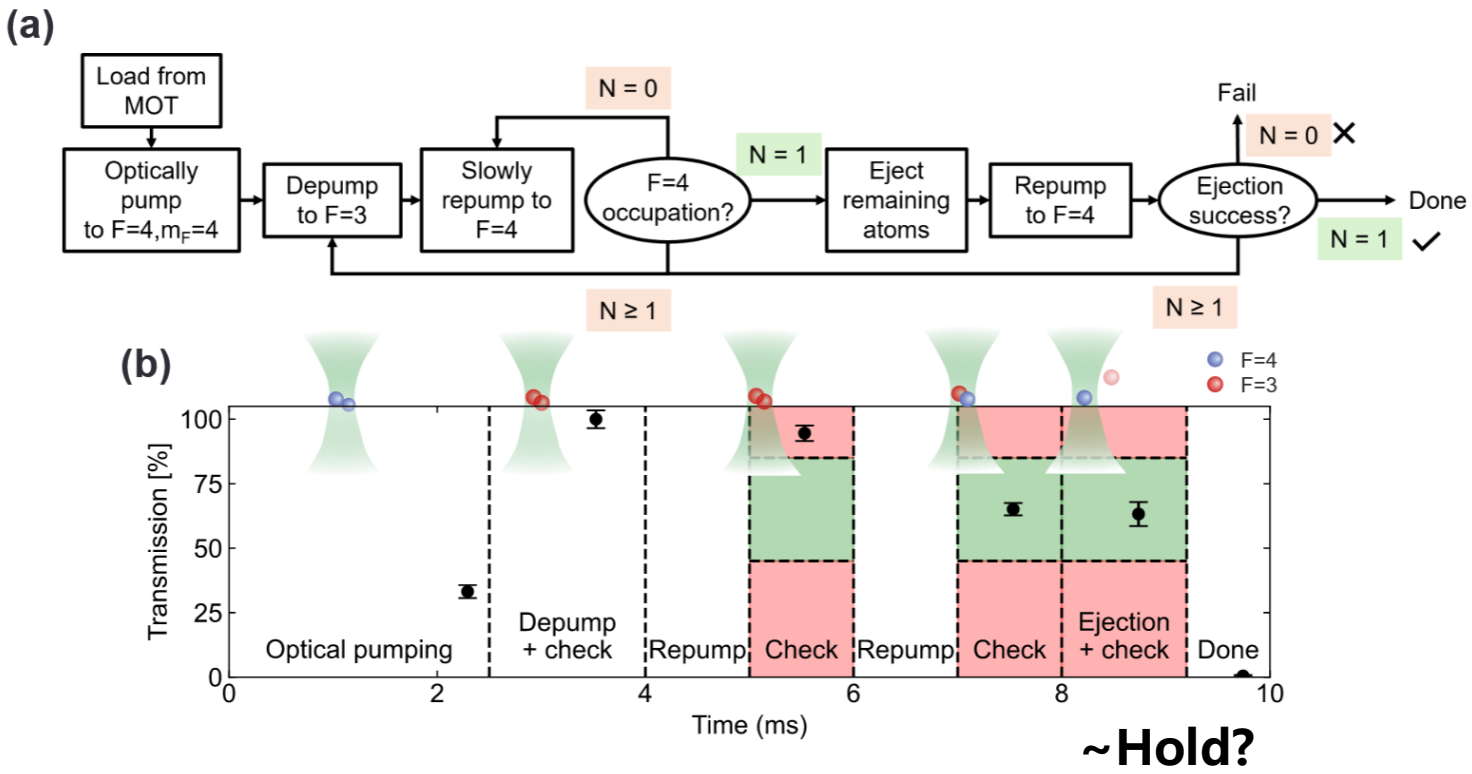
- **We can just change the registers, calculation logic and trigger of PID module, let "PID" do other feedbacks!**
- **Skip the difficulty to build the connection & wiring of different module, which I think it's the most difficult thing when we code FPGA.**



Outlook on RedPitaya: A budget Quantum Machine?

Fast feedback: A budget Quantum Machine?

- **8ns clock time**
- **Other hardware feature? ADC/DAC, channels, input/output range…**
- **Python package conflict (fixed in python 3.9)**
- **Calibration issue! (partly fixed)**



~Hold?

[2411.12622] Cavity-enabled real-time observation of individual atomic collisions

Outlook on RedPitaya: A budget Quantum Machine?

Calibration issue!

- **PyRPL cannot read RedPitaya's calibration data, causing ADC/DAC offsets.**

Outlook on RedPitaya: A budget Quantum Machine?

Calibration issue!

- **PyRPL cannot read RedPitaya's calibration data, causing ADC/DAC offsets.**
- **Digital setpoint: Manually offset calibration needed for precise physical values.**
- **Analog setpoint: No calibration needed because the difference between 2 inputs matters, and finally it will go to 0.**


```
def phy2dig_setpoint_in1(physical_value):
    k1 = (HALF_IN1 - ZERO_IN1)/0.5
    b1 = ZERO_IN1
    return k1 * physical_value + b1
```

Calibration from python
(slow but doesn't matter for pre-loading)

Outlook on RedPitaya: A budget Quantum Machine?

Calibration issue!

- **PyRPL cannot read RedPitaya's calibration data, causing ADC/DAC offsets.**
- **Digital setpoint: Manually offset calibration needed for precise physical values.**
- **Analog setpoint: No calibration needed because the difference between 2 inputs matters, and finally it will go to 0.**
- **Advanced fast feedback systems: We just need to apply the linear function (y=kx+b) two more times in the FPGA calculation logic or add the offset to our algorithm.**
- **Maybe sometimes we can calibrate all the technique offset?**


```
def phy2dig_setpoint_in1(physical_value):
    k1 = (HALF_IN1 - ZERO_IN1)/0.5
    b1 = ZERO_IN1
    return k1 * physical_value + b1
```

Calibration from python
(slow but doesn't matter for pre-loading)

Thanks for listening!

Wanlin Wang
E-mail: wwlyn@mit.edu
wwlyn@outlook.com

Feel free to ask questions!
More details are in the documentation.

Documentation: pyrpl_change/Documentation.md at max_hold_no_iir_improvement · wwlyn/pyrpl_change
PyRPL_change: wwlyn/pyrpl_change
Labscript: wwlyn/red_pitaya_pyrpl_pid
Official Documentation for RP: https://redpitaya.com/rtd-iframe/?iframe=https://redpitaya.readthedocs.io/en/latest/quickStart/needs.html