 doi:10.20944/preprints202106.0015.v1

LUMOS : A PYTHON INSTRUMENT CONTROL LIBRARY FOR PHOTONICS

A PREPRINT

Akhilesh S. P. Khope*
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
akhkhope@gmail.com

May 30, 2021

ABSTRACT

We present a simulation library in Python 3 called Lumos. Different optimization routines are implemented using multiple instruments. Automated measurements are implemented in this library which lead to faster measurements as compared to manual sweeps. We elaborate on the experimental methods and also example usage is mentioned.

1 Introduction

In this paper, we present a simulation library on Python 3 for analyzing and measuring photonic devices and systems. The library is based on PyVisa and simplifies testing and measurement. Fast tuning is possible with sweeping and optimization functions can tune multiple rings in a second order microring resonator (MRR) to any frequency, for example, tuning second order MRR can take a long time if done manually. We present some optimization algorithms to speed up this process. This simulation code can be used as a starter code to test any kind of photonic device or systems in lab.

This library speed up testing in the lab. This improves testing experience as visualization of plots can be done through jupyter notebooks along with the logic for control.

The application scenarios with example code covered in this work are as follows:

1. tuning of Mach Zhender and Micro Rind Resonator (MRR) switches (1^{st}, 2^{nd} and higher order MRR)
2. generating tuning maps of MRR
3. general purpose optimization routine with arbitrary number of and types of instruments for applications like vernier tunable laser, LIDAR application, higher order MRR etc.

Wavelength sweeps are time consuming if done by hand. If one uses python, new code has to be written for new applications, Lumos can speed things up by providing basic functionality.

This code was used to test a $N \times N \times M$ multi-wavelength selective crossbar switch proposed in [1, 2] and reported in [3, 4]. A switch with lesser number of devices with multi casting is also reported in [5, 6]. We also used this code for on-chip wavelength locking experiments [7, 8]. This switch can also be classified as a Elastic WDM switch [3]. Elastic WDM switches are also reported in [9, 10, 11, 12].

*Work done while at UCSB

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1

2 Lumos

Lumos uses Pyvisa [13] with NI visa backend. The library contains multiple functions that combine control of multiple instruments like Keithley Source meters, Thorlabs Photodiode, Yokogawa OSA, Yenista/EXFO Tunable Laser, OSICS Frame and attenuator and many other instruments. All code was run on Jupyter Notebook [14]. The advantage of running code on Jupyter notebooks is that all the code can run and visualized through a notebook interface on any web browser. Numpy [15], Scipy [16] and Pandas [17] packages are heavily used. All plots were implemented with matplotlib package in python [18] Anaconda with Python 3.6 [19] was used for all code. We advise users to modify the functions for their application.

Following libraries are required for every python program that uses the code in the library along with specific packages the reader wishes to use in their research.

```
import visa
import matplotlib.pyplot as plt
import numpy as np
from time import sleep
from time import time
rm = visa.ResourceManager()
rm.list_resources()
```

Some important functions from the library are given below:

1. **keithely_init**: Initializes a sourcemeter to a given voltage. Sets the compliance current to 3 mA. keithley_add is a list of GPIB id's, for ex. [22,21] or [5].

Algorithm 1 keithleyinit

procedure KEITHLEYINIT($keithley\_add, voltset, limit\_current$) ▷ initialize a list of keithleys sourcemeter 2400 GPIB numbers to voltage = voltset and a compliance limit current
  $keithley \leftarrow$visa resources of keithley_add GPIB
  for all keithleys in keithley resource list **do**
    set input buffer size to 100000
    set output buffer size to 2000
    set limit current
    set current level
    set voltage level
    set voltage range
    set function to voltage
    set function to current
    if set voltage is greater than operating voltage of the device set to default value
    set init
    set output on
    set system loc
  return $keithley$ ▷ The gcd is b

2. **keithleyiv**: performs an IV sweep with 20 measurements for each V. Average current for each V is returned.

Algorithm 2 keithleyiv

procedure KEITHLEYIV($keithley, initial, voltrange$) ▷ list of visa resources of keithleys, voltage of keithleys after the sweep, voltrange of keithleys
  $data \leftarrow emptylist$
  for each voltage in voltrange **do** ▷ We have the answer if r is 0
    for n_average number of measurements **do**
      $measureusingkeithley.read()$
    $data \leftarrow averageofn_averagemesurements$
    keithley set to initial
  return $data$ ▷ The gcd is b

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1

3. **yoko_pre**: performs a optical power vs wavelength full range sweep of the OSA

Algorithm 3 yoko_pre

procedure YOKO_PRE($yoko$)
    $yoko.writeRST$
    $initcontON$
    $senseswepoinauto$
    $initimm$
    return $None$

4. **yokorun**: returns the maximum power for a given wavelength. The function conducts a scan from wav-span/2 , wav+span/2 with a resolution of 0.02 nm. A lower span reduces total scanning time when used for wavelength sweeping.

Algorithm 4 yokorun

procedure YOKORUN(a, b)
    $yokotimeoutNone$
    $yokosensmid$
    $yokospe2x$
    $yokores0.02$
    $yokostart(wav - span)/2$
    $yokostop(wav + span)/2$
    $yokoinitimm$
    $init$
    $tracdataytra$
    $tracdataxtragetwavelength$
    $tracdataytragetpower$
    return $wavelength, power$

5. **yenista_sweep**: returns a tunable laser sweep given an input array with wavelength points. Input wavelength array points might not be equally spaced.

Algorithm 5 yenista_sweep

procedure YENISTA_SWEEP(a, b) ▷
    $data \leftarrow emptylist$
    $yenistawritepower$
    $yenistawriteenable$
    for $wavinWLnm$ **do** ▷ We have the answer if r is 0
        $yenistawritewavelength$
        $dataappendyoko_run(yoko, wav, wavelengthspan = 0.1)$
    return $maxpowerofeverywavelengthsweep$ ▷

6. **CE_ring_OSA_lock**: through and drop version. Genetic algorithm for multivariate optimization is used. This is a blackbox optimization method.

Algorithm 6 CE_ring_OSA_lock

procedure CE_RING_OSA_LOCK(a, b) ▷
    $yenistawritewav$
    $mean \leftarrow [guess, guess]$
    $std \leftarrow [std\_guess, std\_guess]$
    while std[0]>res or std[1]>res **do** ▷
        initialize v1 to random normal mean[0],std[0],n1
        initialize v2 to random normal mean[1],std[1],n1
        filter v1 and v2 to values less than device operating voltage and round to n_decimal decimal places
        drop voltages from v1 v2 to the same size
        change v1 v2 to the same size
        for each voltage on v1 and v2 **do**
            set first keithley voltage to v1
            set second keithley voltage to v2
            measure output powers for each v1, v2 and record max power
            get elite samples for both v1 and v2
            $mean \leftarrow meanofelitev1, v2roundedton\_decimal$
            $std \leftarrow stdofelitev1, v2roundedton\_decimal$
    set first keithley voltage to v1
    set second keithley voltage to v2
    return $mean, std$ ▷

7. **hill_climb**: Through and drop port optimization functions optimize the power on the Yokogawa OSA and Thorlabs Powermeter PM100USB.

Algorithm 7 hill_climb

procedure HILL_CLIMB($yenista, yoko, keithley, wav, guess = 4, std_guess = 0.01, n_iter = 300$) ▷
    keithley[0] voltage set to guess
    keithley[1] voltage set to guess
    set yenista to wav
    $bestpair \leftarrow [guess, guess]$
    run full yokogawa osa sweep yoko_run(yoko)
    sleep for 1 s for instrument delay
    for n_iter **do**
        if i==0: set v1, v2 to guess
        else: set v1, v2 to random normal with std_guess around bestpair[0] and bestpair[1], round to n_decimal
        set keithley[0] to voltage v1
        set keithley[1] to voltage v2
        sleep(0.1)
        if i==0: bestvalue to max(yoko_run(yoko,wav,0.1)[1]) bestpair to [v1,v2] print(bestvalue)
        else: value to max(yoko_run(yoko,wav,0.1)[1])
        if value>bestvalue: bestvalue to value bestpair to [v1,v2]
        keithley[0] set to str(bestpair[0])
        keithley[1] str(bestpair[1])

8. **gen_wav_arr**: helper function for a fine. and coarse tuning. This function generates points that are close together at the maximum and coarser away from the maxima. This can used as input into yenista_sweep for a faster sweep.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1

Algorithm 8 gen_wav_arr

procedure GEN_WAV_ARR($maxwav, res_fine = 0.04, res_coarse = 0.2, span = 2$) ▷
  $r \leftarrow a \bmod b$
  for $r \neq 0$ **do** ▷
    $a \leftarrow b$
    $b \leftarrow r$
    $r \leftarrow a \bmod b$
  return $WLnm$ ▷

3 Results generated by using Lumos and Example Usage

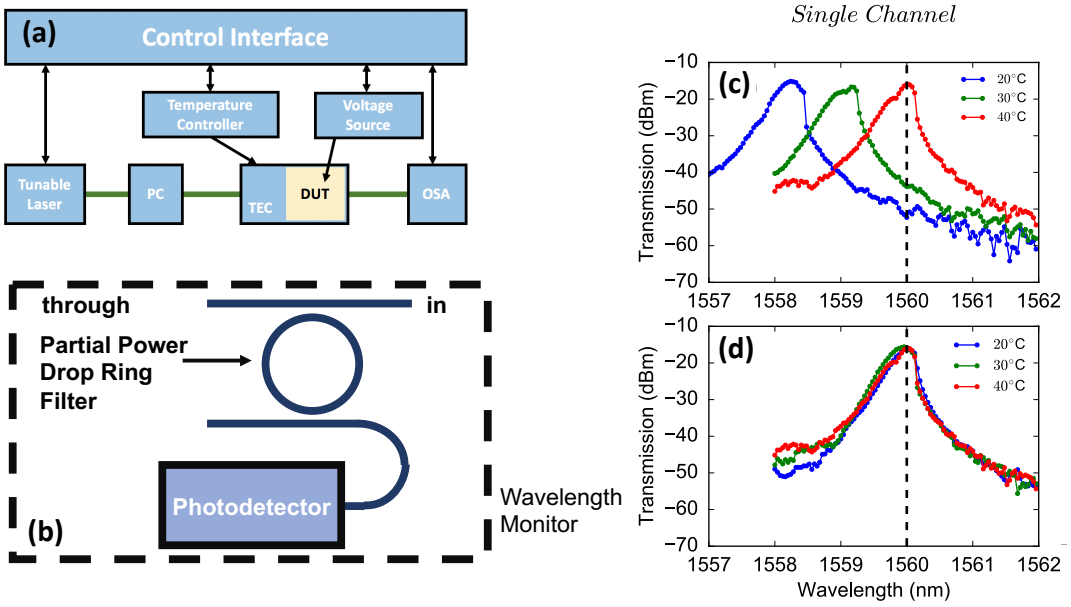

Figure 1: (a) Diagram of test setup PC: Polarization Controller, TEC: Thermo Electric Cooler, DUT: Device Under Test, OSA: Optical Spectrum Analyzer, (b) wavelength monitor on which the experiment was run, (c) single channel locking at different temperatures. [8]

In Fig. 1 (a)-(d) show the experiment in [8, 7]. In this experiment, we lock MRRs in a switch to an on chip wavelength monitor. We use optimization algorithms mentioned in the library to get optimum transmission at the drop port for a single wavelength input. In Fig. 2 (a)-(e) we do the same as Fig. 1 with two channel input. In both these experiments, we had to control Tunable Laser, OSA and multiple Keithley source meters 2401 through a single function.

3.1 Code used to optimize a second order ring spectrum and plot a wavelength vs transmission sweep

```
import lumos as lms
import visa
import matplotlib.pyplot as plt
#% matplotlib inline # remove below comment if not using code on jupyter notebook
import numpy as np
from time import sleep
rm = visa.ResourceManager()
rm.list_resources()
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
gpib_numbers = {}
gpib_numbers["keithley_heater_mrr"] = [23,24]
gpib_numbers["yenista_tunable_laser"] = [25]
gpib_numbers["yokogawa_osa"] = [26]

rm_resources_dict = {}
for key in gpib_numbers:
    # check if GPIB0 or GPIB1 according to which gpib to usb adapter is used
    if key == "keithley_heater_mrr":
        rm_resources_dict[key] = []
        rm_resources_dict[key].append(rm.open_resource("GPIB0::"+str(i)+"::INSTR"))
    else:
        rm_resources_dict[key] = rm.open_resource("GPIB0::"+str(i)+"::INSTR")


# optimize power at the monitor photodiode
# Random Stochstic Hill Climbing
# number of iterations for the optimization algorithm
n_iter = 300
# guess voltage
guess = 2
# standard deviation of guess voltage
std_guess = 0.3
res = 0.01
wav = 1550 #nm
# the following function sets the MRR heaters to the correct voltage
lms.CE_ring_OSA_lock(rm_resources_dict["yenista_tunable_laser"],
                     rm_resources_dict["yokogawa_osa"],
                     rm_resources_dict["keithley_heater_mrr"],
                     wav=wav,
                     guess = guess,
                     std_guess = std_guess,
                     res = res)

# sweep with yenista TL and yokogawa OSA at 0.2 nm resolution
WLnm = np.arange(1530,1565,0.2)
# power in dBm
power_tunable_laser = 0
transmission_dBm = lms.yenista_sweep(rm_resources_dict["yenista_tunable_laser"],/
                                     rm_resources_dict["yokogawa_osa"],/
                                     WLnm = WLnm,/
                                     power = 0)
plt.plot(WLnm,transmission_dBm)
plt.xlabel("Wavelength (nm)")
plt.ylabel("Transmission (dBm)")
plt.show()
```

3.2 Code used to lock a second order MRR to an on-chip wavelength monitor

```
import lumos as lms
import visa
import matplotlib.pyplot as plt
#% matplotlib inline # remove below comment if not using code on jupyter notebook
import numpy as np
from time import sleep
rm = visa.ResourceManager()
rm.list_resources()

gpib_numbers = {}
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
gpib_numbers["keithley_monitor_ring"] = [21]
gpib_numbers["keithley_monitor_pd"] = [22]
gpib_numbers["keithley_heater_mrr"] = [23,24]
gpib_numbers["yenista_tunable_laser"] = [25]
gpib_numbers["yokogawa_osa"] = [26]

rm_resources_dict = {}
for key in gpib_numbers:
    # check if GPIB0 or GPIB1 according to which gpib to usb adapter is used
    if key == "keithley_heater_mrr":
        rm_resources_dict[key] = []
        rm_resources_dict[key].append(rm.open_resource("GPIB0::"+str(i)+"::INSTR"))
    else:
        rm_resources_dict[key] = rm.open_resource("GPIB0::"+str(i)+"::INSTR")

# set monitor MRR to correct voltage
# v1 according to map of v1 vs wavelength
rm_resources_dict["keithley_monitor_ring"][0].write(':SOUR:VOLT'+str(v1))
rm_resources_dict["keithley_monitor_pd"][0].write(':SOUR:VOLT'+str(v2))
for i in range(len(rm_resources_dict["keithley_heater_mrr"])):
    rm_resources_dict["keithley_heater_mrr"][i].write(':SOUR:VOLT'+str(0))

# optimize power at the monitor photodiode
# Random Stochstic Hill Climbing
# number of iterations for the optimization algorithm
n_iter = 300
# guess voltage
guess = 4
# standard deviation of guess voltage
std_guess = 0.01
wav = 1550 nm
# the following function sets the MRR heaters to the correct voltage
lms.hill_climb_pd(rm_resources_dict["yenista_tunable_laser"],/
              rm_resources_dict["keithley_monitor_pd"],/
              rm_resources_dict["keithley_heater_mrr"],/
              wav = wav,/
              guess = guess,/
              std_guess = std_guess,/
              n_iter = 300)

# sweep with yenista TL and yokogawa OSA at 0.2 nm resolution
WLnm = np.arange(1530,1565,0.2)
# power in dBm
power_tunable_laser = 0
transmission_dBm = lms.yenista_sweep(rm_resources_dict["yenista_tunable_laser"],/
                                rm_resources_dict["yokogawa_osa"],/
                                WLnm = WLnm,/
                                power = 0)
plt.plot(WLnm,transmission_dBm)
plt.xlabel("Wavelength (nm)")
plt.ylabel("Transmission (dBm)")
plt.show()
```

3.3 Code used to generate tuning map of a second order MRR

General purpose optimizer and its applications for Vernier Tunable laser or on chip LIDARs

The laser in [20] paper has two rings and a phase section. The rings have to be tuned for the laser to lase at the right frequency and phase section has to be tuned together with it to optimize the output power. For LIDAR, the laser has

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1

to be tuned with the phase section as described in the previous line and multiple tunable grating elements have to be tuned with it to adjust the angle along two axes of the output light. For both these applications the testing can be hugely simplified by using the optimization routine built into Lumos. One can simultaneouly control multiple instrument resources (more than 200 for LIDAR) which can be sourcemeters, lasers , OSAs etc and change their values according to genetic algorithms coded into the library for automatic optimized output. New instrument resources can also be added to the library by following the example template given below. The optimization routine requires that each instrument should have a min and max voltage or current set.

```
def optimization_power_out(instr_resources, instr_volt_curr_range, monitor_resource,
↪   optimization_method, tol = 1e-3, n_iter = 300):
    # optimize multiple current and voltage sources to get the max or min output power
    return
```

For the locking experiment, one needs to set the wavelength monitor MRR to the correct voltage then use the optimization algorithm to tune the drop ring to the correct voltages for second order MRR while optimizing the power received at the wavelength monitor photodiode. This requires control of 4 source meters, while setting the tunable laser to the correct wavelength. Initializing and setting the compliance is done through keithley_init. Setting the correct voltages and wavelength is done through pyvisa. Optimizing the power dropped at the wavelength monitor photodiode is done through hill_climb_pd

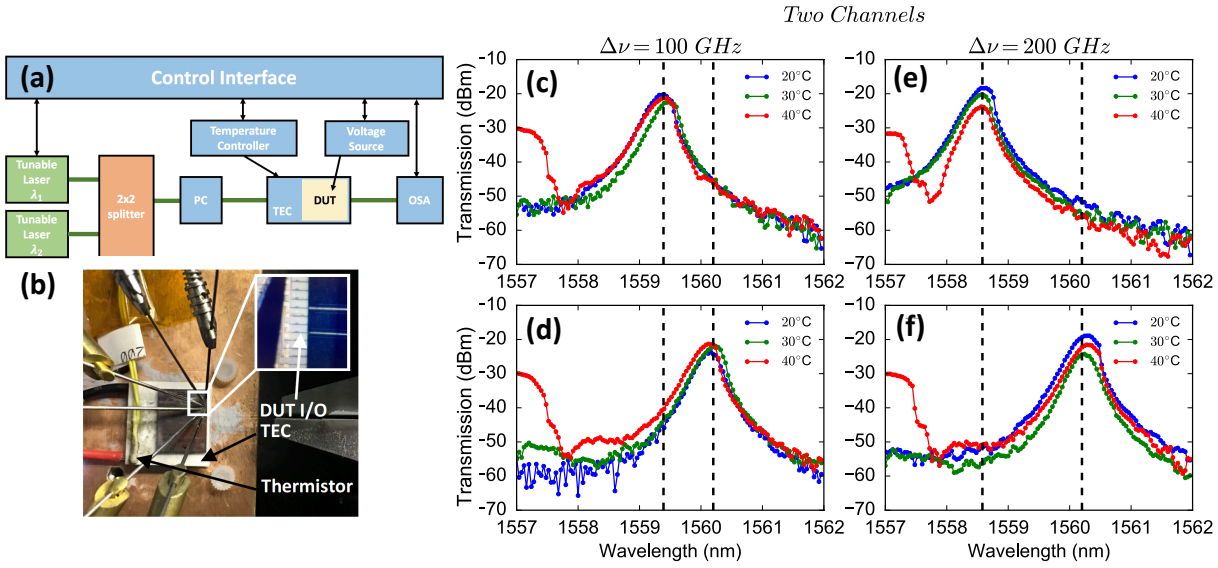

Figure 2: (a) Test setup for two wavelength locking. PC: Polarization Controller, TEC: Thermo Electric Cooler, DUT: Device Under Test, OSA : Optical Spectrum Analyzer, (b) Photo of Photonic Integrated Circuit with probes, (c),(d) locking of two channels at 100 GHz spacing and (e),(f) 200 GHz spacing [8].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1

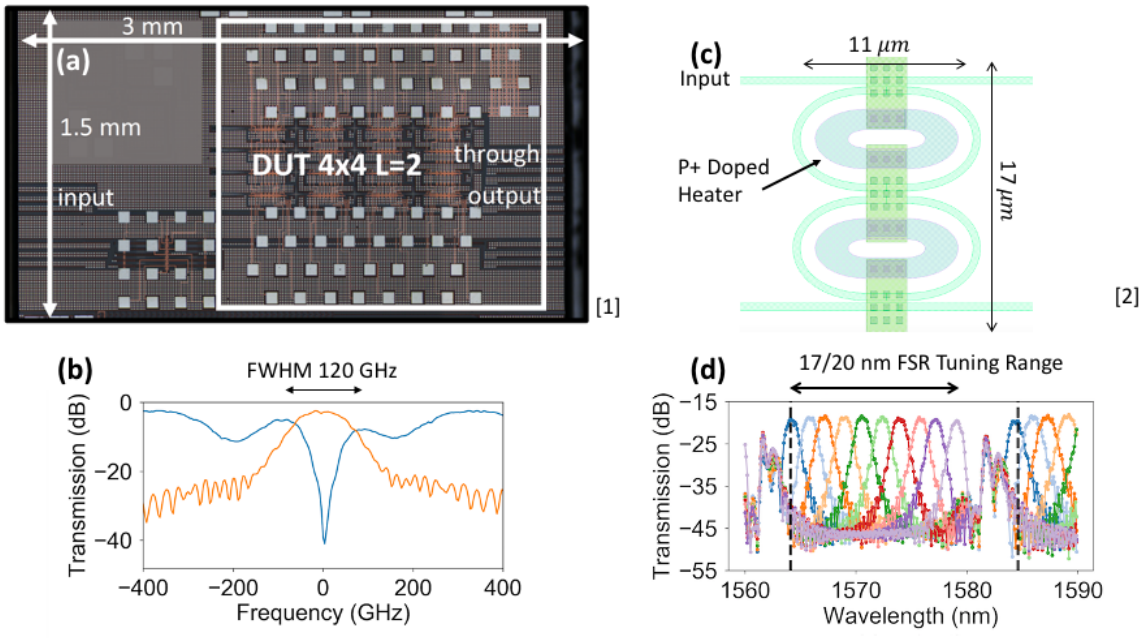

Figure 3: (a) Die shot of 4×4 PIC, (b) MRR spectrum generated through yenista_sweep function, (c) MRR used in the experiment, (d) Tuning of MRR in a switch cell where at each voltage the spectrum is optimized [21, 22]

If one uses OSA to optimize the ring spectrum, one could use heater_set and scipy optimize methods instead of the custom hill climb methods. One should ideally use through port spectrum, but as we didn't have access to one we optimized the drop port spectrum.

References

[1] Akhilesh SP Khope, Adel AM Saleh, John E Bowers, and Rod C Alferness. Elastic wdm crossbar switch for data centers. *IEEE Optical Interconnects (OI) Conference*, pages 48–49, 2016.

[2] Adel AM Saleh, Akhilesh SP Khope, John E Bowers, and Rod C Alferness. Elastic wdm switching for scalable data center and hpc interconnect networks. In *2016 21st OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS)*, pages 1–3. IEEE, 2016.

[3] Akhilesh SP Khope, Mitra Saeidi, Raymond Yu, Xinru Wu, Andrew M Netherton, Yuan Liu, Zeyu Zhang, Yujie Xia, Garey Fleeman, Alexander Spott, et al. Multi-wavelength selective crossbar switch. *Optics express*, 27(4):5203–5216, 2019.

[4] A. S. Prabhu Khope. *Multi-wavelength Selective Crossbar Switch*. PhD thesis, UC Santa Barbara., 2019.

[5] Akhilesh SP Khope, Roger Helkey, Songtao Liu, Sairaj Khope, Rod C Alferness, Adel AM Saleh, and John E Bowers. Scalable multicast hybrid broadband-crossbar wavelength selective switch: proposal and analysis. *Optics Letters*, 46(2):448–451, 2021.

[6] Akhilesh S. P. Khope, Roger Helkey, Songtao Liu, Adel A. M. Saleh, Rod C. Alferness, and John E. Bowers. A scalable multicast hybrid broadband crossbar wavelength selective switch for datacenters. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1585–1587, 2021.

[7] Akhilesh SP Khope, Andrew M Netherton, Takako Hirokawa, Nicolas Volet, Eric J Stanton, Clint Schow, Roger Helkey, Adel AM Saleh, John E Bowers, and Rod C Alferness. Elastic wdm optoelectronic crossbar switch with on-chip wavelength control. In *Photonics in Switching*, pages PTh1D–3. Optical Society of America, 2017.

[8] Akhilesh SP Khope, Takako Hirokawa, Andrew M Netherton, Mitra Saeidi, Yujie Xia, Nicolas Volet, Clint Schow, Roger Helkey, Luke Theogarajan, Adel AM Saleh, et al. On-chip wavelength locking for photonic switches. *Optics letters*, 42(23):4934–4937, 2017.

[9] Tae Joon Seok, Jianheng Luo, Zhilei Huang, Kyungmok Kwon, Johannes Henriksson, John Jacobs, Lane Ochikubo, Richard S Muller, and Ming C Wu. Mems-actuated 8× 8 silicon photonic wavelength-selective switches with 8 wavelength channels. In *2018 Conference on Lasers and Electro-Optics (CLEO)*, pages 1–2. IEEE, 2018.

[10] Yuta Goebuchi, Masahiko Hisada, Tomoyuki Kato, and Yasuo Kokubun. Optical cross-connect circuit using hitless wavelength selective switch. *Optics express*, 16(2):535–548, 2008.

[11] Ryotaro Konoike, Keijiro Suzuki, Shu Namiki, Hitoshi Kawashima, and Kazuhiro Ikeda. Ultra-compact silicon photonics switch with high-density thermo-optic heaters. *Optics express*, 27(7):10332–10342, 2019.

[12] Xian Xiao, Roberto Proietti, Gengchen Liu, Hongbo Lu, Yu Zhang, and SJ Ben Yoo. Multi-fsr silicon photonic flex-lions module for bandwidth-reconfigurable all-to-all optical interconnects. *Journal of Lightwave Technology*, 38(12):3200–3208, 2020.

[13] PyVISA Authors. Pyvisa: Control your instruments with python. `https://pyvisa.readthedocs.io/`, 2016.

[14] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

[15] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[16] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}. 2014.

[17] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[18] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[19] Michel F Sanner et al. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61, 1999.

[20] Martijn JR Heck. Highly integrated optical phased arrays: photonic integrated circuits for optical beam shaping and beam steering. *Nanophotonics*, 6(1):93–107, 2017.

[21] Akhilesh S.P. Khope, Songtao Liu, Andy Netherton, Zeyu Zhang, Sairaj Khope, Roger Helkey, Adel A.M. Saleh, Rod C. Alferness, and John E. Bowers. Experiments on multiwavelength selective crossbar switches. In *2020 International Conference on Information Science and Communications Technologies (ICISCT)*, pages 1–5, 2020.

[22] Akhilesh SP Khope, Songtao Liu, Zeyu Zhang, Andrew M Netherton, Rebecca L Hwang, Aaron Wissing, Jesus Perez, Franklin Tang, Clint Schow, Roger Helkey, et al. 2 λ switch. *Optics Letters*, 45(19):5340–5343, 2020.

4 Appendix : Functions

Initialize Keithley Source meters Model 2401

```
#initialize the Keithleys
def keithley_init(keithley_add,voltset = 2):
    keithley = []
    for i in keithley_add:
        keithley.append(rm.open_resource("GPIB0::"+str(i)+"::INSTR"))
    for i in range(len(keithley)):
        keithley[i].write('*RST')
        keithley[i].Inputbuffersize = 100000
        keithley[i].OutputBuffersize = 2000
        limit_current  = 3 #magnitude of current in mA
        keithley[i].write(':SENS:CURR:PROT '+str(limit_current)+ 'E-3')
        keithley[i].write(':SOUR:CURR:LEV 0')
        keithley[i].write(':SOUR:VOLT:LEV 0')
        # change the level from 0 if required
        #keithley.write(':SOUR:CURR:RANG 1')
        keithley[i].write(':SOUR:VOLT:RANG 7')
        keithley[i].write('SOUR:FUNC VOLT') # source voltage
        keithley[i].write('SENS:FUNC "CURR"')
        #voltset = 2 #magnitude of voltage in V
        if voltset<10:
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
            keithley[i].write(':SOUR:VOLT '+str(voltset))
        else:
            keithley[i].write(':SOUR:VOLT '+str(2))
        keithley[i].write(':INIT')
        keithley[i].write(':OUTP ON')
        keithley[i].write(':SYST:LOC')
    return keithley
```

IV sweep of Keithley Source meters Model 2401

```
def keithley\_iv(keithley,initial,voltrange):
    # give an initial voltage initial so that after the code is run the keithley is set
    ↪  back to its initial voltage
    data = []
    avg_meas = np.zeros(20)
    for i in voltrange:
        keithley.write(':SOUR:VOLT '+str(i))
        sleep(0.1)
        for j in range(20):
            meas = keithley.read().strip().split(",")
            avg_meas[j] = [float(meas[k]) for k in range(len(meas))][1]
        data.append(np.mean(avg_meas))
    keithley.write(':SOUR:VOLT '+str(initial))
    return data
```

Yokogawa OSA AQ6370

```
# should run this before running sweep
def yoko_pre(yoko):
    yoko.write('*RST')
    yoko.timeout = None
    yoko.write(':INIT:CONT ON')
    yoko.write(':SENS:SWE:POIN auto')
    yoko.write(':init:imm;')
# for repeated runs during a sweep
def yoko\_run(yoko,wav,span):
    yoko.timeout = None
    yoko.write(':SENS:SENS MID')
    yoko.write(':SENS:SWE:SPE 2x')
    yoko.write(':SENS:BWID:RES 0.02nm')
    yoko.write(':SENS:WAV:STAR '+str(wav-span/2)+'nm')
    yoko.write(':SENS:WAV:STOP '+str(wav+span/2)+'nm')
    yoko.write(':init:imm;')
    yoko.write(':INIT')
    yoko.write(':TRAC:DATA: Y? TRA;')
    sleep(0.1)
    data1 = yoko.read()
    yoko.write(':TRAC:DATA: X? TRA;')
    sleep(0.1)
    data2 = yoko.read()
    data1 = np.fromstring(data1,dtype = float,sep=',')
    data2 = 1e6*np.fromstring(data2,dtype = float,sep=',')
    data = [data2,data1]
    return data
```

Wavelength sweep with Yenista Tunable laser
Used with Tunics T100S-HP.

```
#yenista wav sweep WLnm is wavelength array in nm
def yenista_sweep(yenista,yoko,WLnm,power = 0):
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
    data = []
    yenista.write('P=%f\n'%power)
    yenista.write('ENABLE')
    for j in range(len(WLnm)):
        yenista.write('L=%f\n'%WLnm[j])
        if j==0:
            yoko_pre(yoko)
        sleep(0.2)
        data.append(yoko\_run(yoko,WLnm[j],0.1))
        sleep(0.1)
    return [max(data[k][1]) for k in range(len(data))]
```

Cross entropy algorithm: drop port

```
# CE OSA ring locking function
def CE_ring_OSA_lock(yenista,yoko,heater,wav,guess = 2,std_guess = 0.3,res = 0.01):
    n1 = 100 # total number of samples per loop
    n2 = 10  # elite samples
    yenista.write('L=%f\n'%wav)
    mean = [ guess,guess]
    std = [ std_guess,  std_guess]
    print([mean,std])
    # std limit of 0.01 V can be changed
    while std[0]> res or std[1] > res:
        v1 = np.random.normal(mean[0],std[0],n1)
        v1 = np.round(v1[(v1<10)],3)
        v2 = np.random.normal(mean[1],std[1],n1)
        v2 = np.round(v2[(v2<10)],3)
        #v1 = np.concatenate((y,v1))
        #v2 = np.concatenate((y,v2))
        if len(v1)<len(v2):
            v2 = v2[:len(v1)]
        else:
            v1 = v1[:len(v2)]
        maxv = np.zeros(len(v1))
        # CE loop to measure detected powers for each voltage across two heaters
        for j in range(len(v1)):
            heater[0].write(':SOUR:VOLT '+str(v1[j]))
            heater[1].write(':SOUR:VOLT '+str(v2[j]))
            if j==0:
                yoko_pre(yoko)
                sleep(0.2)
            sleep(0.1)
            temp = yoko\_run(yoko,wav,0.3)
            maxv[j] = max(temp[1])
        temp2 =v1[np.argsort(maxv)[-n2:]]
        temp3 =v2[np.argsort(maxv)[-n2:]]
        mean = np.round(np.mean([temp2,temp3],1),3)
        std = np.round(np.std([temp2,temp3],1),3)

    heater[0].write(':SOUR:VOLT '+str(mean[0]))
    heater[1].write(':SOUR:VOLT '+str(mean[1]))
    return [mean,std]
```

Cross entropy algorithm through port

```
def CE_ring_OSA_lock_through(yenista,yoko,heater,wav,guess = 2,std_guess = 0.3,res =
↪  0.01):
    n1 = 100 # total number of samples per loop
    n2 = 10  # elite samples
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
    yenista.write('L=%f\n'%wav)
    mean = [ guess,guess]
    std = [ std_guess,  std_guess]
    print([mean,std])
    # std limit of 0.01 V can be changed
    while std[0]> res or std[1] > res:
        v1 = np.random.normal(mean[0],std[0],n1)
        v1 = np.round(v1[(v1<10)],3)
        v2 = np.random.normal(mean[1],std[1],n1)
        v2 = np.round(v2[(v2<10)],3)
        #v1 = np.concatenate((y,v1))
        #v2 = np.concatenate((y,v2))
        if len(v1)<len(v2):
            v2 = v2[:len(v1)]
        else:
            v1 = v1[:len(v2)]
        maxv = np.zeros(len(v1))
        # CE loop to measure detected powers for each voltage across two heaters
        for j in range(len(v1)):
            heater[0].write(':SOUR:VOLT '+str(v1[j]))
            heater[1].write(':SOUR:VOLT '+str(v2[j]))
            if j==0:
                yoko_pre(yoko)
                sleep(0.2)
            sleep(0.1)
            temp = yoko\_run(yoko,wav,0.3)
            maxv[j] = min(temp[1])
        temp2 =v1[np.argsort(maxv)[-n2:]]
        temp3 =v2[np.argsort(maxv)[-n2:]]
        mean = np.round(np.mean([temp2,temp3],1),3)
        std = np.round(np.std([temp2,temp3],1),3)

    heater[0].write(':SOUR:VOLT '+str(mean[0]))
    heater[1].write(':SOUR:VOLT '+str(mean[1]))
    return [mean,std]
```

Random Stochastic Hill Climbing

```
def hill_climb(yenista,yoko,keithley,wav,guess = 4,std_guess = 0.01,n_iter = 300):
    global iter_power
    keithley[0].write(':SOUR:VOLT '+str(guess))
    keithley[1].write(':SOUR:VOLT '+str(guess))
    yenista.write('L=%f\n'%wav)
    bestpair = [guess,guess]
    yoko_pre(yoko)
    sleep(1)
    for i in range(n_iter):
        if i==0:
            v1 = guess
            v2 = guess
        else:
            v1 = bestpair[0] + np.round(np.random.normal(0,std_guess),3)
            v2 = bestpair[1] + np.round(np.random.normal(0,std_guess),3)
        keithley[0].write(':SOUR:VOLT '+str(v1))
        keithley[1].write(':SOUR:VOLT '+str(v2))
        sleep(0.1)
        if i==0:
            print(i)
            bestvalue = max(yoko\_run(yoko,wav,0.1)[1])
            bestpair = [v1,v2]
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
            print(bestvalue)
        else:
            value = max(yoko\_run(yoko,wav,0.1)[1])
            if value>bestvalue:
                bestvalue = value
                bestpair = [v1,v2]
        if i%50==0:
            print(i)
            print(bestvalue)
            print(bestpair)
        iter_power.append(bestvalue)
    keithley[0].write(':SOUR:VOLT '+str(bestpair[0]))
    keithley[1].write(':SOUR:VOLT '+str(bestpair[1]))
    print(i)
    print(bestvalue)
    print(bestpair)
    return
```

Random Stochastic Hill Climbing : through port

```
def hill_climb_th(yenista,yoko,keithley,wav,guess = 4,std_guess = 0.01,n_iter = 300):
    keithley[0].write(':SOUR:VOLT '+str(guess))
    keithley[1].write(':SOUR:VOLT '+str(guess))
    yenista.write('L=%f\n'%wav)
    bestpair = [guess,guess]
    yoko_pre(yoko)
    sleep(1)
    for i in range(n_iter):
        if i==0:
            v1 = guess
            v2 = guess
        else:
            v1 = bestpair[0] + np.round(np.random.normal(0,std_guess),3)
            v2 = bestpair[1] + np.round(np.random.normal(0,std_guess),3)
        keithley[0].write(':SOUR:VOLT '+str(v1))
        keithley[1].write(':SOUR:VOLT '+str(v2))
        sleep(0.1)
        if i==0:
            print(i)
            bestvalue = max(yoko\_run(yoko,wav,0.1)[1])
            bestpair = [v1,v2]
            print(bestvalue)
        else:
            value = max(yoko\_run(yoko,wav,0.1)[1])
            if value<bestvalue:
                bestvalue = value
                bestpair = [v1,v2]
        if i%50==0:
            print(i)
            print(bestvalue)
            print(bestpair)
    keithley[0].write(':SOUR:VOLT '+str(bestpair[0]))
    keithley[1].write(':SOUR:VOLT '+str(bestpair[1]))
    print(i)
    print(bestvalue)
    print(bestpair)
    return
```

Fine coarse array generation for laser scan

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
def gen_wav_arr(maxwav,res_fine = 0.04,res_coarse = 0.2,span = 2):
    temp1 = np.arange(maxwav-span,maxwav+span,res_fine)
    temp2 = np.arange(maxwav-26-span,maxwav-26+span,res_fine)
    temp3 = np.arange(np.max(temp2),np.min(temp1),res_coarse)
    WLnm = np.concatenate((np.concatenate((temp2,temp3)),temp1))
    return WLnm
```

Stochastic Hill Climb with Thorlabs Photodiode

```
def hill_climb_pd(yenista,pd,keithley,wav,guess = 4,std_guess = 0.01,n_iter = 300):
    keithley[0].write(':SOUR:VOLT '+str(guess))
    keithley[1].write(':SOUR:VOLT '+str(guess))
    keithley[0].read().strip().split(",")
    keithley[1].read().strip().split(",")
    yenista.write('L=%f\n'%wav)
    bestpair = [guess,guess]
    yoko_pre(yoko)
    sleep(1)
    for i in range(n_iter):
        if i==0:
            v1 = guess
            v2 = guess
        else:
            v1 = bestpair[0] + np.round(np.random.normal(0,std_guess),3)
            v2 = bestpair[1] + np.round(np.random.normal(0,std_guess),3)
        keithley[0].write(':SOUR:VOLT '+str(v1))
        keithley[1].write(':SOUR:VOLT '+str(v2))
        keithley[0].read().strip().split(",")
        keithley[1].read().strip().split(",")
        if i==0:
            meas = PD[0].read().strip().split(",")
            curr = [float(meas[k]) for k in range(len(meas))][1]
            bestvalue = -curr
            bestpair = [v1,v2]
        else:
            meas = PD[0].read().strip().split(",")
            curr = [float(meas[k]) for k in range(len(meas))][1]
            value = -curr
            if value>bestvalue:
                bestvalue = value
                bestpair = [v1,v2]
        if i%50==0:
            print(i)
            print(bestvalue)
            print(bestpair)
    keithley[0].write(':SOUR:VOLT '+str(bestpair[0]))
    keithley[1].write(':SOUR:VOLT '+str(bestpair[1]))
    keithley[0].read().strip().split(",")
    keithley[1].read().strip().split(",")
    print(i)
```

Sample Scipy optimize code for microring resonators

```
maxwav = 1556.15
print(maxwav)
x0 = [2,2]
yoko_pre(yoko)
def heater_set(x,yenista = yenista,yoko = yoko,keithley = heater1,wav = maxwav):
    yenista.write('L=%f\n'%wav)
    global iter_power
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 June 2021 doi:10.20944/preprints202106.0015.v1


```
    print(str(x[0]))
    print(x[1])

    if abs(x[0])>9:
        x[0] = 9
    if abs(x[1])>9:
        x[1] = 9
    keithley[0].write(':SOUR:VOLT '+str(x[0]))
    keithley[1].write(':SOUR:VOLT '+str(x[1]))
    sleep(0.1)
    power = max(yoko\_run(yoko,wav,0.1)[1])
    iter_power.append(power)
    return power
from scipy.optimize import minimize
iter_power = []
res = minimize(heater_set, x0,args = (yenista,yoko,heater1,1537)\

,method='Nelder-Mead',tol = 0.05,options={'disp': True})
```