Anand Chari
WES 237: Assignment 2
Dining Philosophers:

I started this lab by copying over code from previous labs to set up the LEDs, blinking, and the RGB LED. I then started to write the function to act as a philosopher and gave arguments as two forks which would be represented by locks and then after the amount of time to “nap” then a state of starvation until forks were available again. I considered the thread to represent the philosopher and then created 5 locks to represent the forks. I then created 1 thread for each professor and wrote the philosopher function to try and acquire two non-blocking locks; if the function could acquire the locks, the philosopher ate then slept, otherwise the philosopher napped. This did not work. The first philosopher would eat and nap and the rest were completely blocked. I then switched the algorithm to follow a sequence of eating, napping, sleeping and using the two locks to block threads rather than allowing all to run simultaneously and the threads ran the algorithm correctly. I found that napping must not be longer than eating to avoid the threads undergoing constant starvation. I initially used 10 blink at .5 seconds each for eating, 5 blinks at 2 second each for napping but because napping was longer than eating I ran into deadlock. To counter this I increased eating to 20 blink and decreased napping to 1 second per blink. This fixed the issue.

When implementing random.randint I chose eating bounds to be 15 to 20 and napping bounds between 1 and 5 keeping the time of being on/off the same. This would mean that eating would still take 7.5 to 10 seconds while napping would take between 1 and 5 maintaining that napping would be shorter than eating. This holds our truth from above and keeps our threads from deadlock.

Youtube: https://youtu.be/sihSRZ-xMlM
Github: https://github.com/achari23/wes237assignment2

In [1]:
```
from pynq.overlays.base import BaseOverlay
import time
from datetime import datetime
base = BaseOverlay("base.bit")
import threading
btns = base.btns_gpio
import random
```

In [2]:
```
%%microblaze base.PMODB

#include "gpio.h"
#include "pyprintf.h"

//Function to turn on/off a selected pin of PMODB
void write_gpio(unsigned int pin, unsigned int val){
    if (val > 1){
        pyprintf("pin value must be 0 or 1");
    }

    gpio pin_out = gpio_open(pin);
    gpio_set_direction(pin_out, GPIO_OUT);
    gpio_write(pin_out, val);
}

void reset_gpio() {
    write_gpio(1,0);
    write_gpio(2,0);
    write_gpio(3,0);
    write_gpio(0,0);
}
```

In [3]:

```
reset_gpio()
def blink(t, d, n):
    '''
    Function to blink the LEDs
    Params:
      t: number of times to blink the LED
      d: duration (in seconds) for the LED to be on/off
      n: index of the LED (0 to 3)
    '''
    for i in range(t):
        base.leds[n].toggle()
        time.sleep(d)
    base.leds[n].off()

def blink_g(t,d):
    val = 0
    for i in range(t):
        write_gpio(2,val)
        time.sleep(d)
        if val:
            val = 0
        else:
            val = 1
    reset_gpio()
```

In [4]:

```
def philosopher(l1,l2,n):

    while True:
        f1 = l1.acquire(True)
        f2 = l2.acquire(True)
        if f1 and f2:
            print("Philosopher {} is eating!".format(n))
            if n < 4:
                blink(random.randint(15,20),.5,n)
            else:
                blink_g(random.randint(15,20),.5)
            time.sleep(0)
            l1.release()
            l2.release()
            print("Philosopher {} is finished eating and now sleeps!".format(
            if n < 4:
                blink(random.randint(1,7),1,n)
            else:
                blink_g(random.randint(1,7),1)

            time.sleep(0)
            print("Philosopher {} is awaken and is starving".format(n))
            if n < 4:
                base.leds[n].off()
            else:
                reset_gpio()
            time.sleep(0)
        if btns.read() :
            if l1.locked():
                l1.release()
            if l2.locked():
                l2.release()
            reset_gpio()
            for i in [0,1,2,3]:
                base.leds[n].off()
            break
```

In [5]:

```
fork1 = threading.Lock()
fork2 = threading.Lock()
fork3 = threading.Lock()
fork4 = threading.Lock()
fork5 = threading.Lock()

threads = []

#phil1
t1 = threading.Thread(target=philosopher, args=(fork1, fork2, 0))
threads.append(t1)


#phil2
t2 = threading.Thread(target=philosopher, args=(fork2, fork3, 1))
threads.append(t2)


#phil3
t3 = threading.Thread(target=philosopher, args=(fork3, fork4, 2))
threads.append(t3)


#phil4
t4 = threading.Thread(target=philosopher, args=(fork4, fork5, 3))
threads.append(t4)

#phil5
t5 = threading.Thread(target=philosopher, args=(fork5, fork1, 4))
threads.append(t5)

for t in threads:
    t.start()
    print("thread {} started!".format(t))
for t in threads:
    name = t.getName()
    t.join()
    print('{} joined'.format(name))
```

```
Philosopher 0 is eating!
thread <Thread(Thread-4, started 2929906784)> started!
thread <Thread(Thread-5, started 2882065504)> started!
Philosopher 2 is eating!thread <Thread(Thread-6, started 2873672800)> started!

thread <Thread(Thread-7, started 2865280096)> started!
thread <Thread(Thread-8, started 2856887392)> started!
Philosopher 2 is finished eating and now sleeps!
Philosopher 0 is finished eating and now sleeps!Philosopher 4 is eating!
Philosopher 1 is eating!

Philosopher 2 is awaken and is starving
Philosopher 0 is awaken and is starving
Philosopher 4 is finished eating and now sleeps!Philosopher 3 is eating!
```

```
Philosopher 1 is finished eating and now sleeps!Philosopher 0 is eating!

Philosopher 1 is awaken and is starving
Philosopher 4 is awaken and is starving
Philosopher 3 is finished eating and now sleeps!
Philosopher 2 is eating!
Philosopher 0 is finished eating and now sleeps!Philosopher 3 is awaken and is
starving

Philosopher 4 is eating!
Philosopher 0 is awaken and is starving
Philosopher 4 is finished eating and now sleeps!
Philosopher 2 is finished eating and now sleeps!Philosopher 1 is eating!Philos
opher 3 is eating!


Philosopher 4 is awaken and is starving
Philosopher 2 is awaken and is starving
Philosopher 3 is finished eating and now sleeps!
Philosopher 1 is finished eating and now sleeps!Philosopher 0 is eating!
Philosopher 2 is eating!

Philosopher 1 is awaken and is starving
Philosopher 3 is awaken and is starving
Philosopher 0 is finished eating and now sleeps!Philosopher 4 is eating!
Philosopher 2 is finished eating and now sleeps!
Philosopher 1 is eating!

Philosopher 2 is awaken and is starving
Philosopher 0 is awaken and is starving
Philosopher 1 is finished eating and now sleeps!
Philosopher 4 is finished eating and now sleeps!Philosopher 3 is eating!

Philosopher 0 is eating!
Exception in thread Thread-8:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "<ipython-input-4-25fd98305a6b>", line 35, in philosopher
  File "/usr/local/share/pynq-venv/lib/python3.8/site-packages/pynq/lib/axigpi
o.py", line 226, in __getitem__
    raise IndexError()
IndexError
Philosopher 1 is awaken and is starving
Philosopher 4 is awaken and is starving
Exception in thread Thread-4:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "<ipython-input-4-25fd98305a6b>", line 13, in philosopher
RuntimeError: release unlocked lock
```

```
Thread-4 joined
Thread-5 joined
Exception in thread Thread-7:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "<ipython-input-4-25fd98305a6b>", line 14, in philosopher
RuntimeError: release unlocked lock
Philosopher 2 is eating!
Exception in thread Thread-6:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "<ipython-input-4-25fd98305a6b>", line 13, in philosopher
RuntimeError: release unlocked lock
Thread-6 joined
Thread-7 joined
Thread-8 joined
```

In [ ]: