Assignment 3 Write Up

When it came to designing this assignment, I started out by referring to the previous lab assignments with regards to using ctypes. First, I copied the clock_example and kernel_module folders from Lab3. I then referred to the “main.c” file from Lab 2 and followed the instructions from that lab to wrap the “cycletimers.h” functions for use in Jupyter Notebooks. At first, I had some issues with undefined symbols when trying to test my code in Jupyter Notebooks. It turned out that I needed to rearrange the files in the Assignment3 folder I created so the compiler would link everything correctly.

For the main implementation of the assignment, I started off by copying over the CPU_select notebook from Lab 3, since it already has a way to select which CPU core to process with. I’ve also copied over the modified fib.py file from that lab since it is already set up to run with the CPU_select code. However, when testing out the cycle counting at this point, I noticed that the cycle counts were very inaccurate (cycle counts ranging from 200k - 400k for n=30 for example). I figured that there was a lot of overhead with Jupyter Notebooks running code around the fib.py file, so I further modified the fib.py file to include the ctype functions for the cycle counter. This by itself made the output of the cycle count much more realistic; with results now consistently around 44m. This still seemed fairly off, however, so I made sure to pass 1,0 in the init_counters function to disable the counter divider. Now the results are consistently at around 257m with an execution time of 0.395s at n=25. (I had to drop n from 30 since the CPU cycle counter was now overflowing and giving negative numbers.) By dividing the average cycle times by their respective average execution times, we end up with around a 650 MHz clock for the CPU, which lines up with the official spec for this PYNQ-Z2 board.

The Github repository can be found here.

In [25]:

```
import ctypes
import time
import sys
import pylab as pl
from IPython import display
import psutil
import matplotlib.pyplot as plt
import numpy as np
import os
```

In [77]:

```
# Program to pick CPU core
coreID = int(input("Which core to use? "))
param = int(input("How many terms? ")) # modified fib.py to pass term param

#beforeC = func.getCyclecount()

cmd = "taskset -c " + str(coreID) + " python3 fib.py " + str(param)
os.system(cmd)

#afterC = func.getCyclecount()

#cycleTotal = afterC - beforeC

#print('cycles spent: {}'.format(cycleTotal))
```

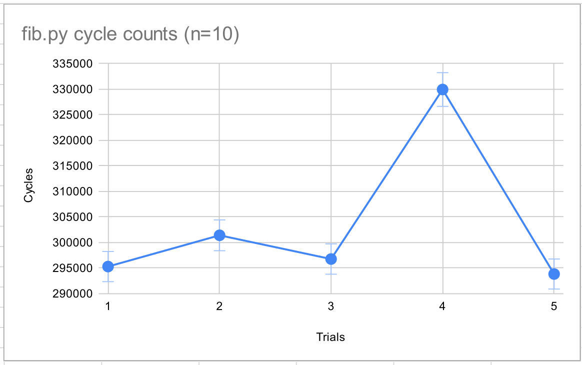
```
Which core to use? 1
How many terms? 10
time spent: 0.0005097389221191406
cycles spent: 293844
```

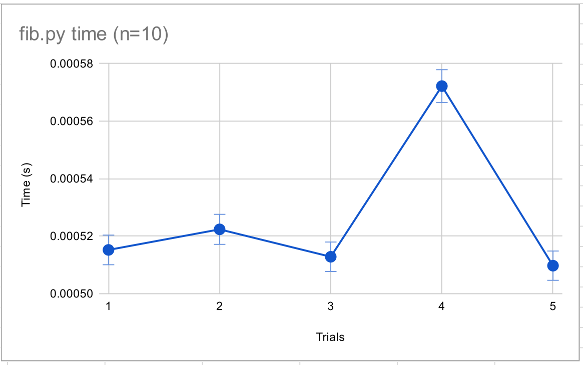
Out[77]:

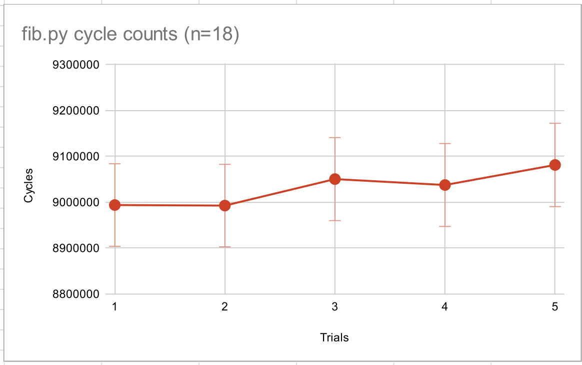
```
0
```

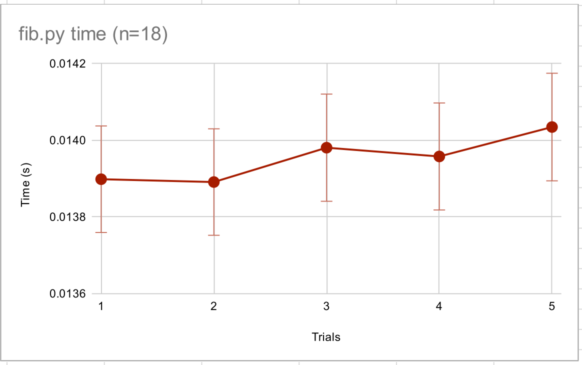
In [ ]:

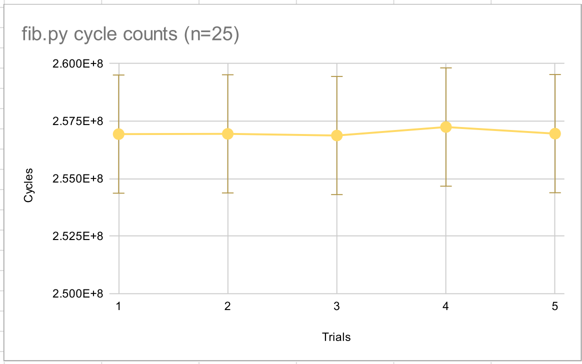
```python
#! /usr/bin/python

import ctypes
import time
import sys
func = ctypes.CDLL('./libMyLib.so')
# Program to calculate the Fibonacci sequence up to n-th term
## nterms = int(input("How many terms? ")) (moved to CPU Select notebook)
func.initPMU()
def recur_fibo(n):
   if n <= 1:
       return n
   else:
       return(recur_fibo(n-1) + recur_fibo(n-2))

tic = time.time()
beforeC = func.getCyclecount()

# check if the number of terms is valid
if int(sys.argv[1]) <= 0:
   print("Please enter a positive integer")
else:
   recur_fibo(int(sys.argv[1]))

tac = time.time()
afterC = func.getCyclecount()
cycleTotal = afterC - beforeC

print('time spent: {}'.format(tac-tic))
print('cycles spent: {}'.format(cycleTotal))

```

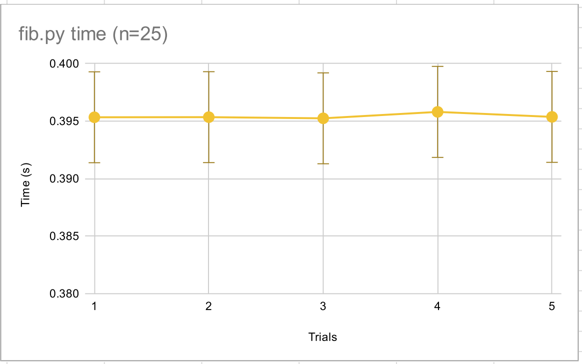

fib.py cycle counts (n=10)

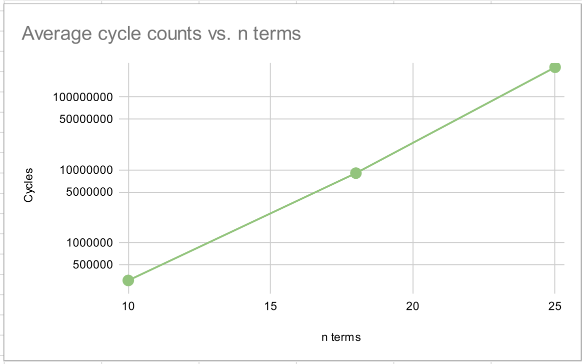

fib.py time (n=10)

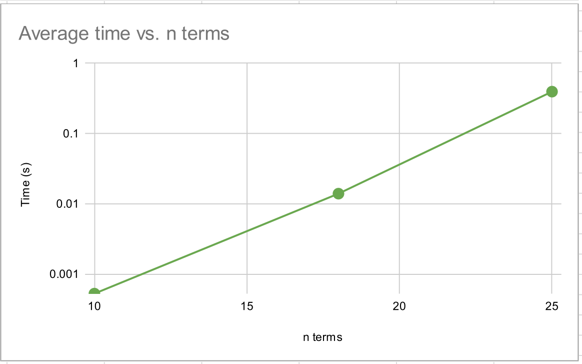

fib.py cycle counts (n=18)


fib.py time (n=18)


fib.py cycle counts (n=25)


fib.py time (n=25)


Average cycle counts vs. n terms


Average time vs. n terms