ECE 3170 Cryptographic Hardware for Embedded Systems

Fall 2025

Assoc. Prof. Vincent John Mooney III
Georgia Institute of Technology

Lab 5, 100 pts.

Due Tuesday, December 2nd prior to 11:55pm

(please turn in electronically on Canvas)

Introduction to Power Analysis with the ChipWhisperer-NANO

In this lab, you will analyze power traces from the ChipWhisperer-NANO board and observe how plots of power traces can reveal hints about the instructions the board is carrying out.

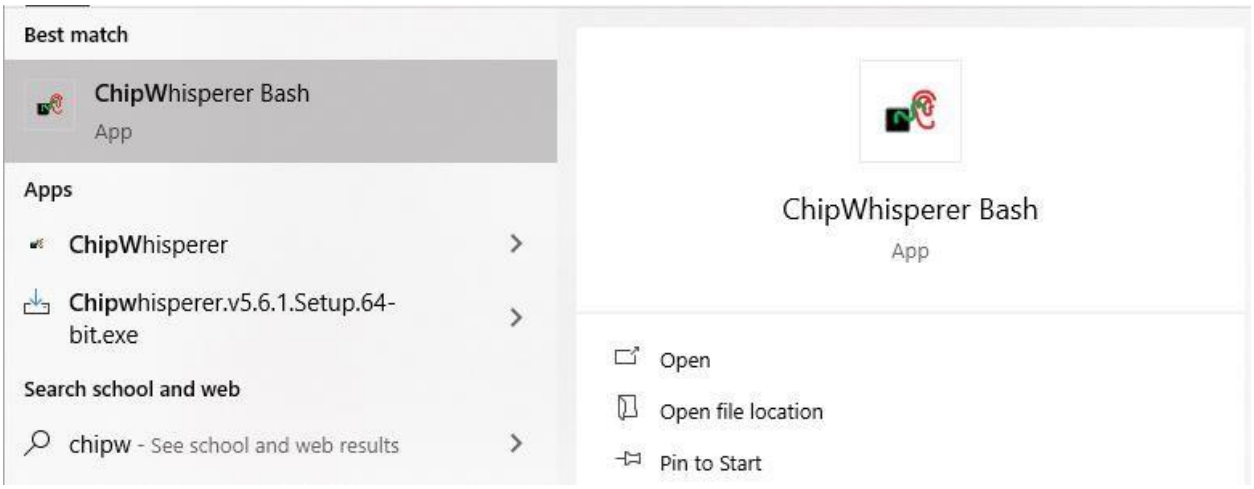
This lab will use the same software packages from the previous lab (Python, pip JupyterLab, Chocolatey). Please proceed to the next page for the steps on how to complete this lab.

1. Launch JupyterLab.

   Windows Note: This time, you will not use the command prompt, but rather the ChipWhisperer bash. The ChipWhisperer bash is installed by default from the ChipWhisperer Windows installation (https://github.com/newaetech/chipwhisperer/releases).

   To launch the ChipWhisperer bash, search for the program "ChipWhisperer Bash" and launch the program. A command window should open. You can still launch Jupyter by typing "jupyter lab." Create a Python 3 notebook for this lab.

   ChipWhisperer bash is not necessary for Mac/Linux, as a regular command window will suffice.



   With Jupyter Lab open, first enter the following code in a cell to specify your hardware setup:

```
SCOPETYPE = 'CWNANO'
PLATFORM = 'CWNANO'
```

   In your computer's file system, navigate to the directory

   (Windows)C:/Users/15055/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware.

   (Other) /Users/USERNAME/chipwhisperer/firmware/mcu
   (or wherever the chipwhisper file was downloaded)

   Create a new folder here called simpleserial-base-lab5 and **copy the contents of the simpleserial-base directory into the simpleserial-base-lab5 directory.**

2. Run the following in a new cell, **replacing YOURUSER with your system username**. **Change the path as needed based on the previous step.** You will also have to re-run this every time you make changes to your code, as this block of code builds firmware onto the board:

   %%bash -s "$PLATFORM"
   cd
   C:/Users/YOURUSER/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware/simpl eserial-base-lab5
   make PLATFORM=$1 CRYPTO_TARGET=NONE

   If the firmware builds successfully, you should see an output like the following after running the cell:

```
SS_VER set to SS_VER_1_1
C:/Users/15055/CHIPWH~1/cw/home/portable/avrgcc/bin/make clean_objs .dep
make[1]: Entering directory 'C:/Users/15055/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware/simpleserial-base-lab2'
SS_VER set to SS_VER_1_1
rm -f -- simpleserial-base-CWNANO.hex
rm -f -- simpleserial-base-CWNANO.eep
rm -f -- simpleserial-base-CWNANO.cof
rm -f -- simpleserial-base-CWNANO.elf
rm -f -- simpleserial-base-CWNANO.map
rm -f -- simpleserial-base-CWNANO.sym
rm -f -- simpleserial-base-CWNANO.lss
rm -f -- objdir-CWNANO/*.o
rm -f -- objdir-CWNANO/*.lst
rm -f -- simpleserial-base.s simpleserial.s stm32f0_hal_nano.s stm32f0_hal_lowlevel.s
rm -f -- simpleserial-base.d simpleserial.d stm32f0_hal_nano.d stm32f0_hal_lowlevel.d
rm -f -- simpleserial-base.i simpleserial.i stm32f0_hal_nano.i stm32f0_hal_lowlevel.i
mkdir .dep
make[1]: Leaving directory 'C:/Users/15055/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware/simpleserial-base-lab2'
C:/Users/15055/CHIPWH~1/cw/home/portable/avrgcc/bin/make begin gccversion build sizeafter fastnote end
make[1]: Entering directory 'C:/Users/15055/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware/simpleserial-base-lab2'
SS_VER set to SS_VER_1_1
.
Welcome to another exciting ChipWhisperer target build!!
arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103 (release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

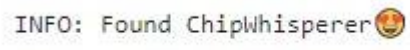
NOTE: On windows machines, WSL *may* cause conflicts preventing jupyter from using the chipwhisperer shell (instead of the wsl bash shell). This would show up as strange errors about arm-none-eabi-gcc not being found despite being installed correctly. **This can be corrected by replacing %%bash with %%sh**

3. Now, setup this ChipWhisperer using the following block of Python code in a new cell:

```
import chipwhisperer as cw
try:
    if not scope.connectStatus:
        scope.con()
except NameError:
    scope = cw.scope()

try:
    target = cw.target(scope)
except IOError:
    print("INFO: Caught exception on reconnecting to target - attempting to reconnect to scope first.")
    print("INFO: This is a work-around when USB has died without Python knowing. Ignore errors above this line.")
```

```
    scope = cw.scope()
    target = cw.target(scope)

print("INFO: Found ChipWhisperer 😍")
if "STM" in PLATFORM or PLATFORM == "CWLITEARM" or PLATFORM ==
"CWNANO":
    prog = cw.programmers.STM32FProgrammer
elif PLATFORM == "CW303" or PLATFORM == "CWLITEXMEGA":
    prog = cw.programmers.XMEGAProgrammer
else:
    prog = None

import time
time.sleep(0.05)
scope.default_setup()
def reset_target(scope):
    if PLATFORM == "CW303" or PLATFORM == "CWLITEXMEGA":
        scope.io.pdic = 'low'
        time.sleep(0.05)
        scope.io.pdic = 'high_z' #XMEGA doesn't like pdic driven high
        time.sleep(0.05)
    else:
        scope.io.nrst = 'low'
        time.sleep(0.05)
        scope.io.nrst = 'high_z'
        time.sleep(0.05)
```

If the code runs successfully, you should see the following after running the cell:



4. The following code will upload the firmware to your board. Make sure to replace YOUR USER with your system username **and update the path as neccessary**:

cw.program_target(scope, prog, "C:/Users/YOUR USER/ChipWhisperer5_64/cw/home/portable/chipwhisperer/hardware/victims/firmware/simpleserial-base-lab5/simpleserial-base-{}.hex".format(PLATFORM))

If the code runs successfully, you should see the following after running the cell. Please make sure to wait for the “Verified flash OK” message before proceeding to step 5:

```
Detected known STMF32: STM32F04xxx
Extended erase (0x44), this can take ten seconds or more
Attempting to program 4655 bytes at 0x8000000
STM32F Programming flash...
STM32F Reading flash...
Verified flash OK, 4655 bytes
```

5. Now, in a new cell, you will define a function to capture a power trace from the board:

```
def capture_trace():
    ktp = cw.ktp.Basic()
    key, text = ktp.next()
    return cw.capture_trace(scope, target, text).wave
```

6. Insert the following into a new cell:

```
wave = capture_trace()
print("✔ OK to continue!")
```

Make sure you see the “OK to continue!” message after running the cell before proceeding.

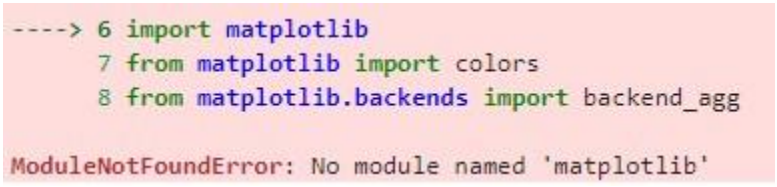
Now, open the simpleserial-base.c file located in the simpleserial-base-lab5 directory from step 1).

In this file, remove the simpleserial_put() function call located within the get_pt() function.

7. Rebuild reupload the firmware onto the ChipWhisperer board using the code in steps 2 and 4 and capture a trace with the code above. **After you have rebuilt and reuploaded the firmware,** plot your captured trace with the following code:

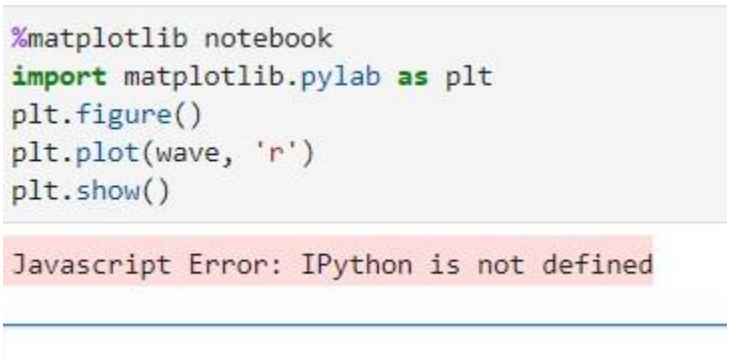
```
%matplotlib notebook
import matplotlib.pylab as plt

plt.figure()
plt.plot(wave, 'r')
plt.show()
```

Note that if you get an error like the following:



Then you will have to install the matplotlib module using the command **"python -m pip install -U matplotlib"** in your command prompt **(not the ChipWhisperer bash, just the normal Windows command prompt).**

If you get an error like the following:

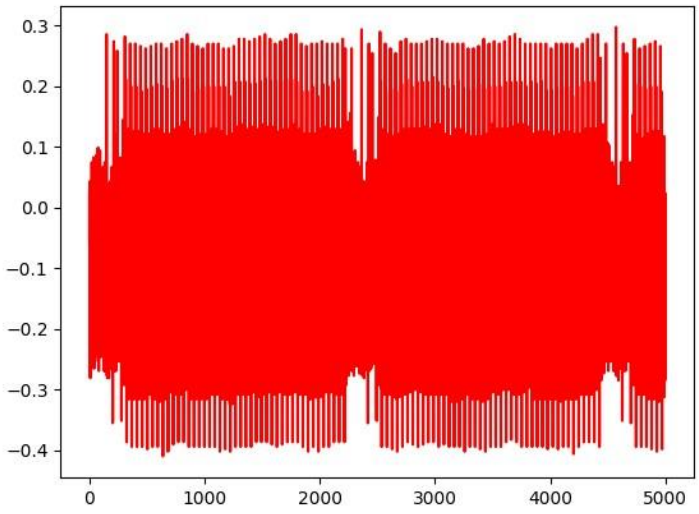


Then you will have to add the following line of code before the plt.figure() statement:

```
%matplotlib inline
```

The resulting plot should look like the one below:

[10]:
```
import matplotlib.pylab as plt

plt.figure()
plt.plot(wave, 'r')
plt.show()
```



Understanding Power Traces

In this section we will try to understand how different operations appear in a power trace and do some simple power analysis.

1. Let's begin by modifying the ChipWhisperer code to do a few addition operations.
   Open the simpleserial-base.c file in the simpleserial-base-lab5 directory from earlier. Add the following code to the get_pt() function between trigger_high() and trigger_low().

   Note: The volatile keyword is needed to prevent the compiler from optimizing the code into a single A = 0x35f instruction

```
volatile long int A = 0x2FB;
volatile long int B = 0;
B = A / 15;
B = A / 15;
B = A / 15;
B = A / 15;
B = A / 15;
```

2. Rerun the cells to recompile the c code (%%bash …) and reprogram the chip-whisperer (cw.program_target...).
3. In a new cell, write code to capture a new trace and plot the first 1200 samples using matplotlib. Can you tell where the divisions live in the power trace?
   You can select the first N entries from a python array with the following snippet:
   new_array = original_array[:N]
4. Rerun this experiment from step 1-3 with 3 divisions, then with 10. How does the power trace change each time?
5. Let's go back to 5 divisions. Try changing the divisor for one of the divisions from 15 to something else. How does this change the trace?

Required Documents to turn in

- Entire Jupyter notebook for this lab, including the outputs of running each cell.
- All figures generated with your code.
- Comment briefly your thoughts on the labs that have involved using the ChipWhisperer board and Jupyter, E.G., did you find these labs engaging? Were you able to follow and understand the instructions, or were there missing steps and errors you had difficulty troubleshooting?

This lab was written with reference to the SCA101 course from the ChipWhisperer GitHub, located at https://github.com/newaetech/chipwhisperer-jupyter/tree/c940073159c8032877e9f7b9ef852b3662c4ec02/courses/sca101