Lab 1: Physical Unclonable Functions

Derek Caprio & Igil Duro
Department of Computer Science & Engineering
University of South Florida
Tampa, FL

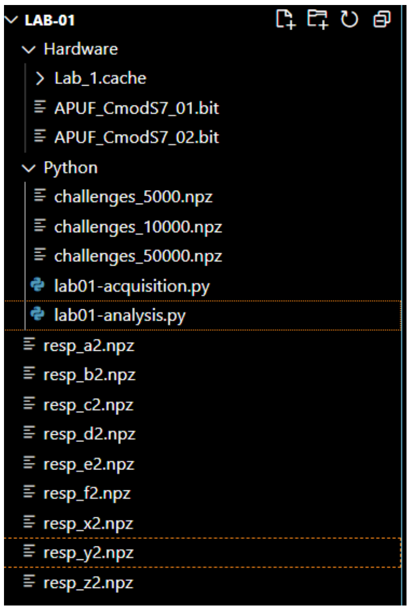
I. Introduction

For this lab we evaluate the reproducibility and uniqueness of an Arbiter PUF (APUF). To determine this, a string of bits called a challenge is written into the APUF, and an output string of bits is produced. From this output we find the Hamming distance (HD) by calculating the percentage of response bits that are different from the same challenge. Reproducibility is a measure of how well an APUF is able to produce the same output if the same challenge is written in multiple times. Uniqueness is a measure of how different outputs are if a challenge is fed to two different APUFs. Ideally, the HD for reproducibility should be close to 0% and uniqueness should be close to 50%

II. Methods

For this experiment, we were provided three files, APUF_CmodS7_01.bit, APUF_CmodS7_02.bit and a challenges_5000.npz. We used the APUF files to program a CmodS7 board using Xilinx Vivado v2020.2. Then using Python we wrote three scripts. One script (challenge_responses.py) to write challenges to the APUF over a serial connection, read the response back in, and save to a npz file. The second script (analysis.py) analyzes the data from the response files to assess the reproduciblity and uniqueness of the APUF. The mean and standard deviations of the HDs are also calculated. Finally, the third script generates (plots.py) bar charts using matplotlib to show how many bits of the 16-bit responses have changed. Three responses are generated for each partner and an additional three using the second bitstream for intrachip variation.

Igil's response files from the challenges_5000.npz file generated by the APUF_CmodS7_01 bitstream file are labelled as resp_a2.npz, resp_b2.npz, and resp_c2.npz. Derek's response files generated from the same challenges on the same bitstream are labeled as resp_x2.npz, resp_y2.npz, and resp_z2.npz. Derek's response files from the same challenges_5000.npz file but generated by the APUF_CmodS7_02 bitstream for intrachip comparison are labeled as resp_d2, resp_e2, and resp_f2.



```
LAB-01
  Hardware
    Lab_1.cache
    APUF_CmodS7_01.bit
    APUF_CmodS7_02.bit
  Python
    challenges_5000.npz
    challenges_10000.npz
    challenges_50000.npz
    lab01-acquisition.py
    lab01-analysis.py
  resp_a2.npz
  resp_b2.npz
  resp_c2.npz
  resp_d2.npz
  resp_e2.npz
  resp_f2.npz
  resp_x2.npz
  resp_y2.npz
  resp_z2.npz
```

Fig. 1. Use the "figure" directive to insert a single-column image.

III. Results

A. Reading Check

1. What is the purpose of the arbiter in the APUF? - In an APUF, the purpose of an arbiter is to decide which signal arrives first. Outputting a 0 or 1.

2. Why might the PUF response bit change when different challenges are applied to the same circuit on the same FPGA? - When a challenge is a applied to a PUF, the challenge bits control if top and bottom paths swap at each stage. So if different challenges are being applied to the circuit the the path an input pulse takes will vary. This can result in delays in each stage since one path may be slightly faster than the other. Resulting in a different output.

3. Why might the PUF response bits change when the same challenges are applied to the same circuit on different FPGAs? - If the same challenges are being run through the same circuit on different FPGAs, the response bits can change due to the result of process variation. Meaning transistors in ICs can have slight variations of each other even in the same IC design. These variations result in propagation delays which then cause different outputs that PUFs make use of to produce unique outputs from each other.

B. Analysis of Results

Ideally, an APUF should have an interchip hamming distance of 50% and an intrachip hamming distance of 0%. However, implementations on FPGAs can cause deviations from the ideal behavior. Additionally, we appear to also be

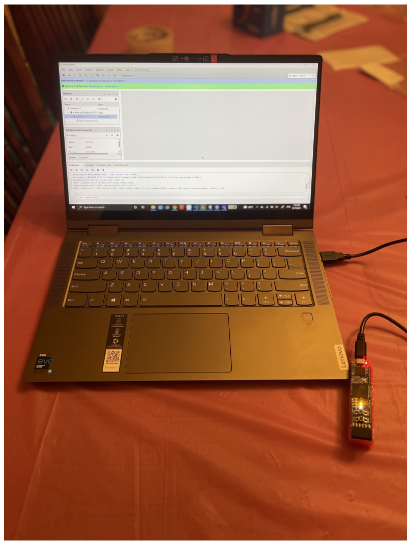


Fig. 2. Use the “figure” directive to insert a single-column image.

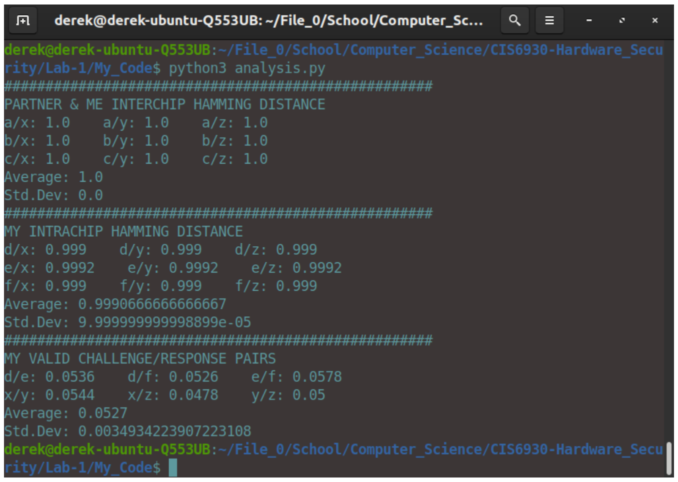
experiencing other errors leading to non-ideal behavior, as will be described.

Figures 3 and 5 shows the results of running the analysis.py file. This script compares all combinations of responses in regard to inter-chip, intrachip with different APUF bitstreams, and intrachip with the same bit-stream. Notice that the hamming distance for valid challenge/response pairs have an average of 0.0527. This is the section which compares the three responses to the same challenges on the same APUF bitstream.A, b, and c are used with one bitstream, d, e, and f are used with one bitstream, and x, y, and z with the other. So, we would expect to see a small Hamming distance as we indeed do. However, the partner to partner interchip HD has an average of 1.0 with no deviation. This is not the behavior we would expect as we would ideally see an HD of 50% here. Instead we see that every bit has changed. Finally, when looking at the intrachip HD, we see another average very close to 1.0 and with very little deviation. This is again unexpected. Ideally, this could be close to 0 and would mean that the chip can produce the same responses under different conditions.

Figures 6, 4, and 7 demonstrate our attempt at counting how many bits are different between responses. Unfortunately, there are unresolved errors in the implementation. However, the intrachip and partner to partner interchip comparisons do match what we see with the HDs in Figure 3, namely that all bits are being changed. But because these HDs do not demonstrate ideal behavior, the graphs do not either. In regard to Figure 7 for the responses to the same challenges, there must be some error in creating the graph as this does not reflect what we see in Figure 3.

IV. Discussion

As described in Section III, many of the results were not what we expected. Some of this may be due to the nature of implementing an APUF on changeable hardware such as the FPGA, but others may be due to errors in our implementation.



```
derek@derek-ubuntu-Q553UB:~/File_0/School/Computer_Science/CIS6930-Hardware_Secu
rity/Lab-1/My_Code$ python3 analysis.py
####################################################
PARTNER & ME INTERCHIP HAMMING DISTANCE
a/x: 1.0    a/y: 1.0    a/z: 1.0
b/x: 1.0    b/y: 1.0    b/z: 1.0
c/x: 1.0    c/y: 1.0    c/z: 1.0
Average: 1.0
Std.Dev: 0.0
####################################################
MY INTRACHIP HAMMING DISTANCE
d/x: 0.999    d/y: 0.999    d/z: 0.999
e/x: 0.9992    e/y: 0.9992    e/z: 0.9992
f/x: 0.999    f/y: 0.999    f/z: 0.999
Average: 0.9990666666666667
Std.Dev: 9.999999999998899e-05
####################################################
MY VALID CHALLENGE/RESPONSE PAIRS
d/e: 0.0536    d/f: 0.0526    e/f: 0.0578
x/y: 0.0544    x/z: 0.0478    y/z: 0.05
Average: 0.0527
Std.Dev: 0.0034934223907223108
derek@derek-ubuntu-Q553UB:~/File_0/School/Computer_Science/CIS6930-Hardware_Secu
rity/Lab-1/My_Code$
```

Fig. 3. Hamming distance means and standard deviations

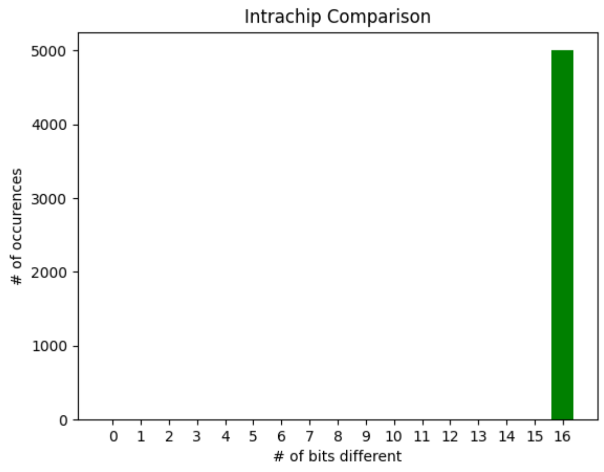
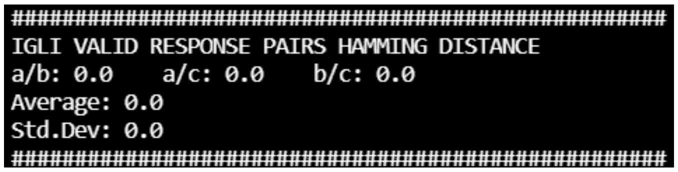


Fig. 4. Intrachip variations

We are able to verify the functionality of the APUF that it is able to generate the same responses for the same challenges, as seen in the means and standard deviations of valid responses to the same challenges on the same chip, but the intrachip behavior between two different APUF bitstreams and partner to partner interchip variation both have much larger hamming distances than expected. In addition, we suspect the bar graphs are not being generated correctly, but need to experiment more to determine whether this is caused by an error in the hamming distance calculations or the graphing itself.



```
############################################################
IGLI VALID RESPONSE PAIRS HAMMING DISTANCE
a/b: 0.0    a/c: 0.0    b/c: 0.0
Average: 0.0
Std.Dev: 0.0
############################################################
```

Fig. 5. Valid Response Pairs

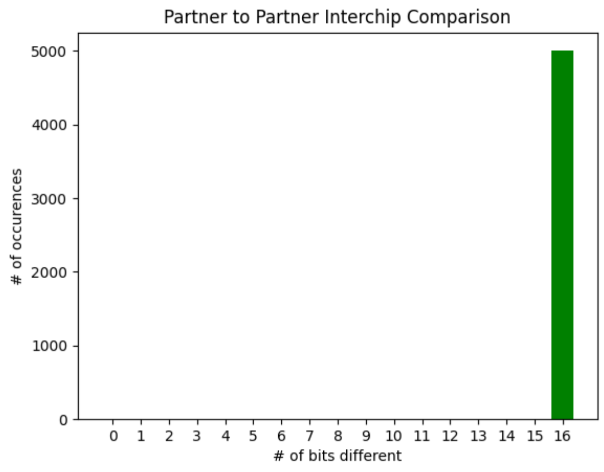


Fig. 6. Interchip variations

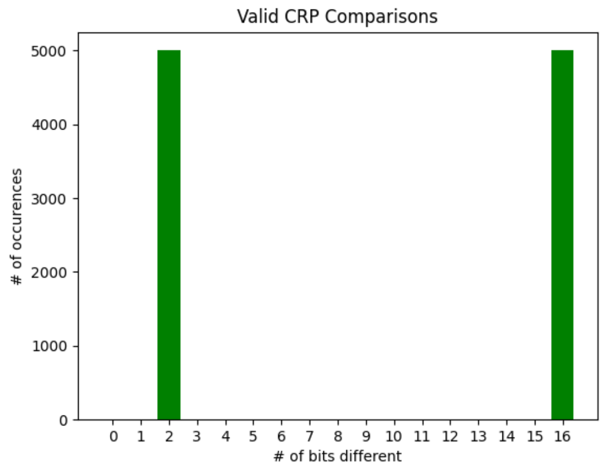


Fig. 7. Valid response variations

V. Conclusion

In this lab we evaluated the properties of an APUF using several challenges, on several APUF bitsreams, and between two different FPGAs. Hamming distances, means and standard deviations between hamming distances were used for analyzing the behavior of the APUFs. Results were also graphed to show how many bits are dissimilar between responses, however these graphs need more work and are not currently reliable. The APUF behaved as expected when receiving the same challenges, but the behavior was much less than ideal when handling different APUF implementations.