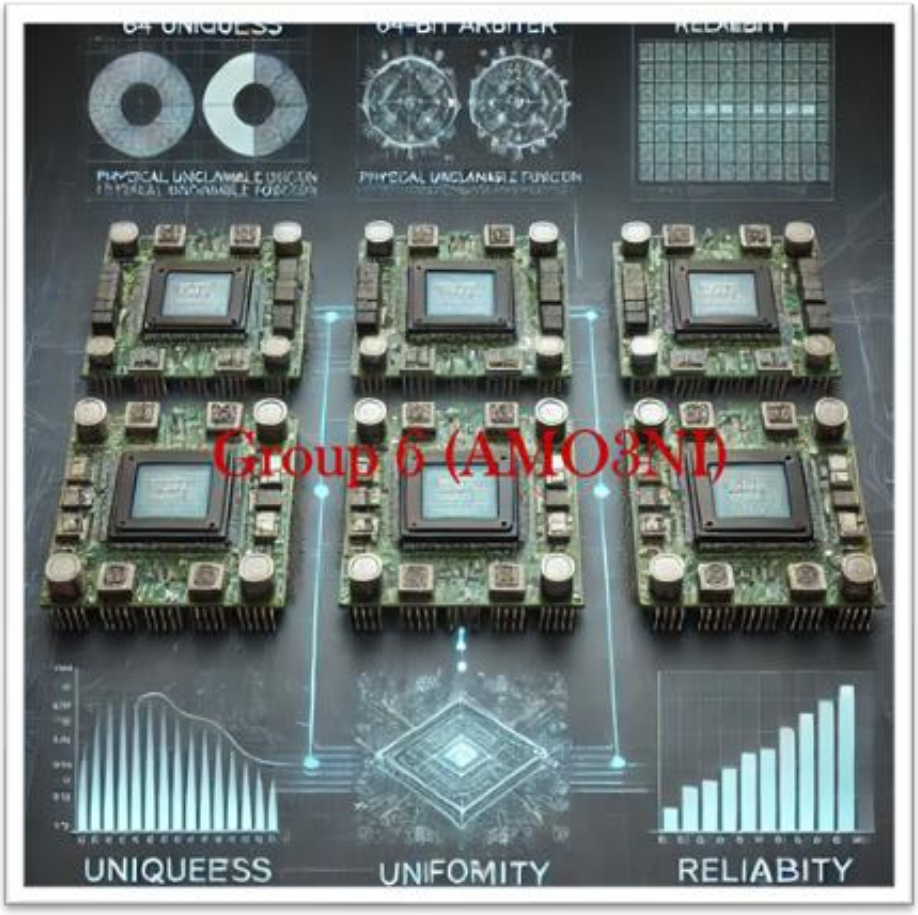


Analysis of Arbiter-PUF Using 10K CRPs

CS984- ASSIGMENT-2

Student ID	Name	Email
233560019	Mohammed Jawed	mjawed23@iitk.ac.in
233560020	Mohit Paritosh	pmohit23@iitk.ac.in
233560021	Mohit Srivastava	mohitsri23@iitk.ac.in
233560023	Nikita Shetty	narayans23@iitk.ac.in
233560025	Ajit Madhusudan Paranjape	paranjape23@iitk.ac.in

Contents

Abstract

This report provides an in-depth analysis of a 64-bit Arbiter Physical Unclonable Function (PUF) implemented on three distinct FPGA boards. The analysis evaluates three key performance metrics: **Reliability**, **Uniformity**, and **Uniqueness** using 10k Challenge-Response Pairs (CRPs) for each board.

The results indicate a high reliability of **99.75%** for **Board 3**, showcasing the PUF's consistency in producing stable responses under repeated challenges. However, the uniformity of the PUF responses is significantly below the ideal value of 50%, with values of **25.51%** for **Board 1**, **27.54%** for **Board 2**, and **21.45%** for **Board 3**. This imbalance in the distribution of '1's and '0's suggests a need for improvement in ensuring randomness in the PUF's output.

Moreover, the uniqueness between pairs of boards is also lower than expected, with values of **3.67%** between **Board 1 and Board 2**, **6.93%** between **Board 2 and Board 3**, and **4.32%** between **Board 1 and Board 3**. These low uniqueness values imply that the PUF instances on different boards are not sufficiently distinct, which could undermine the security of the system.

Overall, while the PUF demonstrates excellent reliability, the results suggest that enhancements are necessary to improve the uniformity and uniqueness, ensuring that the PUF responses are both random and distinct across different implementations. This analysis provides critical insights into the current performance of the PUF and highlights areas for future improvement.

LEARNING OUTCOMES

From this assignment, we gained a deep understanding of the practical aspects of evaluating PUFs using metrics such as uniqueness, uniformity, and reliability. we learned how to:

1. Parse and Handle Large Datasets: Efficiently manage and process large sets of CRP data files.

2. Implement and Calculate PUF Metrics: Uses Python (Jupyter *.ipynb) to calculate key performance metrics, including the application of mathematical formulas such as Hamming distance for uniqueness and majority voting for reliability.

3. Visualize Data for Better Interpretation: Leverage data visualization techniques to present findings in a clear and intuitive manner.

This experience has enhanced our understanding of PUFs, their importance in hardware security, and the methods used to evaluate their effectiveness.

1. Problem Statement

The objective of this assignment is to analyze the behavior of a 64-bit Arbiter Physical Unclonable Function (PUF) implemented on three different FPGA boards by evaluating three key metrics:

- Uniqueness
- Uniformity
- Reliability

Given:

- A set of 10k Challenge-Response Pairs (CRPs) for three different FPGA boards.
- 15 sets of CRPs for one of the boards to evaluate the reliability.

SPECIFIC TASKS

1. Uniqueness Calculation: Measure how unique the responses from one board are compared to another.
   - Uniqueness between Board 1 and Board 2.
   - Uniqueness between Board 1 and Board 3.
   - Uniqueness between Board 2 and Board 3.
2. Uniformity Calculation: Determine how balanced the responses are on each board, ideally measuring if each response bit is approximately 50% '1's and 50% '0's.
3. Reliability Calculation: Assess how consistently the same board produces the same response under the same challenges across different sets of CRPs.

2. Approach

2.1 DATA COLLECTION

- The response data is collected by parsing the CRP files. The provided files contain alternating lines of challenges and responses.
- Each board has a set of 10k responses, and for reliability analysis, 15 sets of responses are provided for Board 3.

2.2 METRIC CALCULATIONS

- **Uniformity**: Calculated as the percentage of '1' bits in the responses for each board.
- **Uniqueness**: Evaluated using the Hamming distance between the response vectors of two different boards. The uniqueness percentage is calculated based on the fraction of differing bits.
- **Reliability**: Determined by comparing the response consistency across the 15 sets of CRPs for Board 3. The reliability percentage represents the degree of consistency across these sets.

3. Implementation

3.1 DATA LOADING

A function G6_Challenge_Response_list is employed to extract the responses from the CRP files and store them in respective lists for each board.

```
def G6_Challenge_Response_list(Board_Responses, file_name):
    """
    Reads the response file and extracts the response bits, adding them to the provided list.

    Parameters:
    Board_Responses (list): List to store the responses.
    file_name (str): Path to the file containing the CRPs.
    """
    with open(file_name, "r") as file:
        File_data = file.readlines()
        for idx, line in enumerate(File_data):
            if idx % 2 != 0:  # Only process response lines (assumes every other line is a response)
                sliced_string_responses = line.split(":")[1].strip()
                Board_Responses.append(sliced_string_responses)
```

3.2 UNIFORMITY CALCULATION

Uniformity is calculated by determining the proportion of '1' bits in the response data for each board.

- Estimates how uniform the proportion of 0's and `1's is in the response bits of a PUF.
- For truly random PUF responses, this proportion must be 50%.

Formula

$$uniformity_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l}$$

where $r_{i,l}$ is the l-th binary bit of an n-bit response from a chip i.

Simplified formula according to approach use in this assignment:

$$\text{Uniformity} = \left(\frac{\text{Sum of '1' bits}}{\text{Total number of bits}}\right) \times 100$$

Implementation

```
def G6_Calculate_Uniformity(Board):
    """
    Calculates the uniformity of a PUF based on the response bits.

    Parameters:
    Board (list): List of response bits for a board.

    Returns:
    float: The uniformity percentage.
    """

    sum_responses = sum(int(bit) for row in Board for bit in row)
    count_responses = len(Board) * len(Board[0])
    uniformity = (sum_responses / count_responses) * 100
    return round(uniformity, 2)
```

3.3 UNIQUENESS CALCULATION

Uniqueness between two boards is determined by calculating the Hamming distance between their respective response vectors.

- represents the ability of a PUF to uniquely distinguish a particular chip among a group of chips of the same type.
- Ideal value is 50%

Formula

$$uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\%$$

where:

- $HD(R_i, R_j)$ is Hamming Distance between n-bit signature of chip *i* and *j*.
- *K* is the number of chip containing PUF under interest.

Implementation

```
def G6_Calculate_Uniqueness(Board_x, Board_y):
    """
    Calculates the uniqueness between two boards based on their response bits.

    Parameters:
    Board_x (list): List of response bits for board X.
    Board_y (list): List of response bits for board Y.

    Returns:
    float: The uniqueness percentage.
    """
    hamming_distance = sum(x != y for x, y in zip(Board_x, Board_y))
    uniqueness = (hamming_distance / len(Board_x)) * 100
    return round(uniqueness, 2)
```

3.4 RELIABILITY CALCULATION

Reliability is calculated by comparing the responses from 15 different CRP sets for Board 3. The reliability metric indicates the consistency of the board's responses.

How efficient a PUF is in reproducing the response bits.

- Employ intra-chip HD among several samples of PUF response bits to evaluate it.
- The same n-bit response is extracted at a different operating condition (different ambient temperature or different supply voltage)
- Ideal value is 100%

Formula

$$reliability = (1 - \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R_{i,t})}{n}) \times 100\%$$

Where:

- R_i is n-bit response of PUF instance *i* at normal operating conditions.
- $R_{i,t}$ is the *t*-th sample of R_i .

Simplified Formula based on assignment:

$$\text{Reliability} = \left(\frac{\text{Sum of agreement ratios}}{\text{Total number of CRP sets}}\right) \times 100$$

Where the agreement ratio for each CRP is calculated as:

$$\text{Agreement Ratio} = \left(\frac{\text{Number of times the majority bit appears}}{\text{Total number of CRP sets}}\right) \times 100$$

Implementation

```python
def G6_Reliability_Of_PUF(base_filename):
    """
    Calculates the reliability of the PUF for a particular board by comparing multiple sets of responses.

    Parameters:
    base_filename (str): The base filename for the multiple sets of CRPs.

    Returns:
    float: The reliability percentage.
    """
    all_responses = []
    for i in range(1, 16):
        filename = f"{base_filename}{i}.txt"
        responses = []
        G6_Challenge_Response_list(responses, filename)
        all_responses.append(responses)

    reliability = 0
    for i in range(10000):  # Assumes 10K CRPs
        bits = [int(all_responses[j][i]) for j in range(15)]
        majority_vote = 1 if bits.count(1) > bits.count(0) else 0
        agreement_ratio = (bits.count(majority_vote) / 15) * 100
        reliability += agreement_ratio

    return round(reliability / 10000, 2)
```

3.5 RESULTS PRESENTATION

The results are printed and visualized using bar charts and line plots to enhance interpretability.

Results Visualization

- **Uniformity**: Visualized using a bar chart, with distinct colors for each board.
- **Uniqueness**: Displayed using another bar chart, showing the uniqueness percentage between different board pairs.
- **Reliability**: Represented by a line plot across 15 CRP sets for Board 3.

```
# 1. Uniformity Bar Chart with different colors

uniformity_data = pd.DataFrame({
    'Boards': ['Board 1', 'Board 2', 'Board 3'],
    'Uniformity (%)': [board1_uniformity, board2_uniformity, board3_uniformity]
})

# Create the barplot with updated parameters
ax = sns.barplot(x='Boards', y='Uniformity (%)', data=uniformity_data, hue='Boards', palette='husl', legend=False)

# Add value annotations on top of each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 3),
                textcoords='offset points')
# Add a title and show the plot
plt.title('Uniformity of Responses for Each Board')
plt.show()
```

```
# 2. Uniqueness Bar Chart with different colors
uniqueness_data = pd.DataFrame({
    'Comparison': ['Board 1 vs. Board 2', 'Board 2 vs. Board 3', 'Board 1 vs. Board 3'],
    'Uniqueness (%)': [uniqueness_Board1_Board2, uniqueness_Board2_Board3, uniqueness_Board1_Board3]
})

ax =sns.barplot(x='Comparison', y='Uniqueness (%)', data=uniqueness_data, hue='Comparison', palette='coolwarm', legend=False)

# Add value annotations on top of each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 3),
                textcoords='offset points')

plt.title('Uniqueness Between Board Pairs')
plt.show()
```

4. Analysis of Results

```
********************  Group 6 OutPut ********************
The Reliability for Board 3 is: 99.75%
The Uniformity for Board 1 is: 25.51%
The Uniformity for Board 2 is: 27.54%
The Uniformity for Board 3 is: 21.45%
The Uniqueness between Board 1 and Board 2 is: 3.67%
The Uniqueness between Board 2 and Board 3 is: 6.93%
The Uniqueness between Board 1 and Board 3 is: 4.32%

********************  End  ********************
```

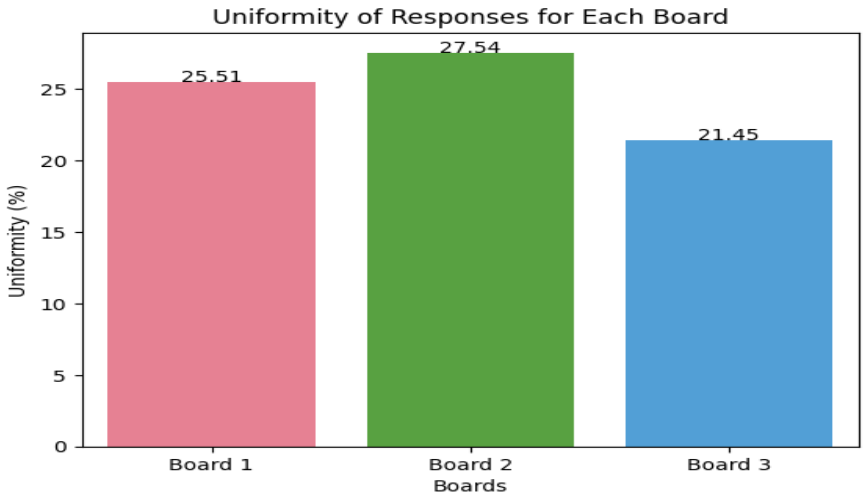
4.1 RELIABILITY FOR BOARD 3

99.75%

Interpretation: The reliability metric indicates that Board 3 is highly consistent in generating the same response when presented with the same challenge. A reliability of 99.75% suggests that the PUF on Board 3 is extremely stable and reliable, which is critical for secure applications.

4.2 UNIFORMITY

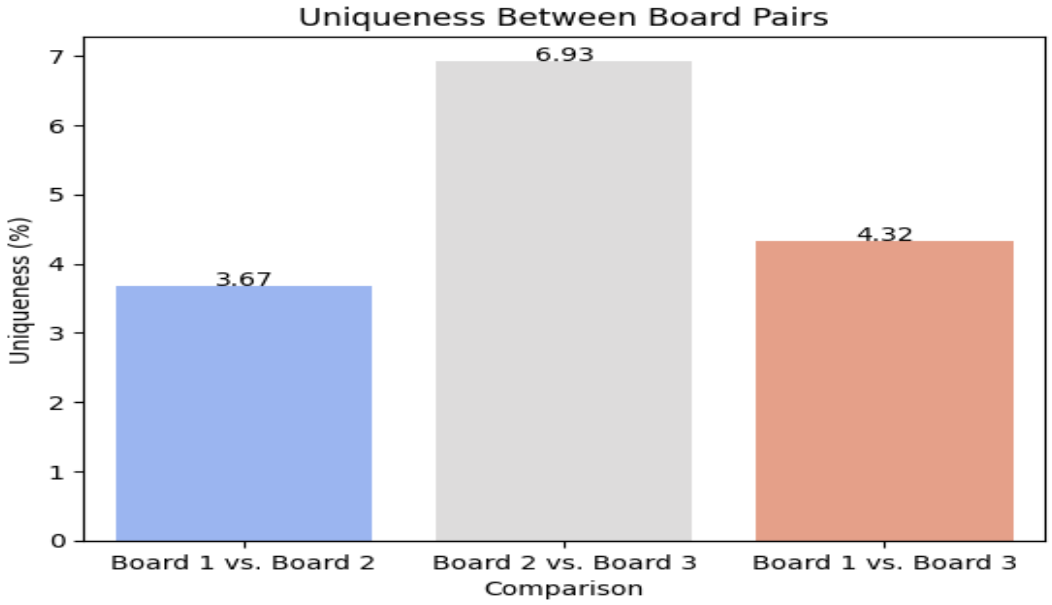
- Board 1: 25.51%
- Board 2: 27.54%
- Board 3: 21.45%



Interpretation: The uniformity percentages are lower than the ideal 50%, indicating that the distribution of '1's and '0's in the response bits is imbalanced. In an ideal PUF, the uniformity should be close to 50% to ensure randomness. The results suggest that the PUF responses are skewed, which may affect the overall security and randomness of the PUF.

4.3 UNIQUENESS

- Board 1 vs. Board 2: 3.67%
- Board 2 vs. Board 3: 6.93%
- Board 1 vs. Board 3: 4.32%



Interpretation: The uniqueness values are lower than expected, as ideal uniqueness should be close to 50%. Low uniqueness indicates that the PUF responses from different boards are very similar, which is not desirable. High uniqueness is essential for distinguishing between different PUF instances on different boards, ensuring that each PUF is unique and not clonable.

4.4 CONCLUSION

Reliability is excellent, demonstrating that the PUF on Board 3 consistently produces stable responses.

Uniformity needs improvement as the response bits are not evenly distributed, which could impact the randomness and security of the PUF.

Uniqueness is significantly below the ideal value, suggesting that the PUFs on different boards are not sufficiently distinct, potentially compromising the security by reducing the effectiveness of the PUFs in distinguishing different devices.

5.References

[1] CS984: Introduction to Hardware Security, Week-5

[2] Mukhopadhyay, D., & Chakraborty, R.S. (2014). Hardware Security: Design, Threats, and Safeguards (1st ed.). Chapman and Hall/CRC. (Indian Institute of Technology Kharagpur,West Bengal, Indian textbook)

[3] S joshi, S P. Mohanty, E. Kougianos, Everything You wanted to know about PUFs, Nov.2017, https://ieeexplore.ieee.org/document/8103099, PP. 38 -46

[4] A Theoretical Model to Link Uniqueness and Min-Entropy for PUF Evaluations, https://pureadmin.qub.ac.uk/ws/portalfiles/portal/156143741/main.pdf

[5] A. Maiti, V. Gunreddy, and P. Schaumont, A System-atic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. New York, NY:Springer New York, 2013, pp. 245–267