IIS5008 Hardware Security
Programming Assignment 1: Ring Oscillator PUF
(Due:2025/04/04 23:59:59)

- Version 2 (2025/03/14): Changed Fig.2 with order and counter max num.
- Version 3 (2025/03/25): Changed the description and figures of Scrambler.

Introduction

The demand for computer security in electronic products has increased due to the rapid expansion of the electronics industry. Physical Unclonable Functions (PUFs)[1], also known as physical one-way functions, are inexpensive to fabricate, inherently random, intrinsically tamper-resistant, and difficult to duplicate. This makes PUFs stand out as an ideal candidate for providing a hardware-based security design for secret key generation and storage.

In this programming assignment, you are required to implement a Feedforward Ring Oscillator PUF (FRO PUF)[2] using Verilog or SystemVerilog, **simulate** it based on the schematic diagram(0), and observe the characteristics of the PUF. Additionally, you are required to analyze the characteristics of the FRO PUF based on the collected data, evaluating its performance metrics such as uniqueness and uniformity.

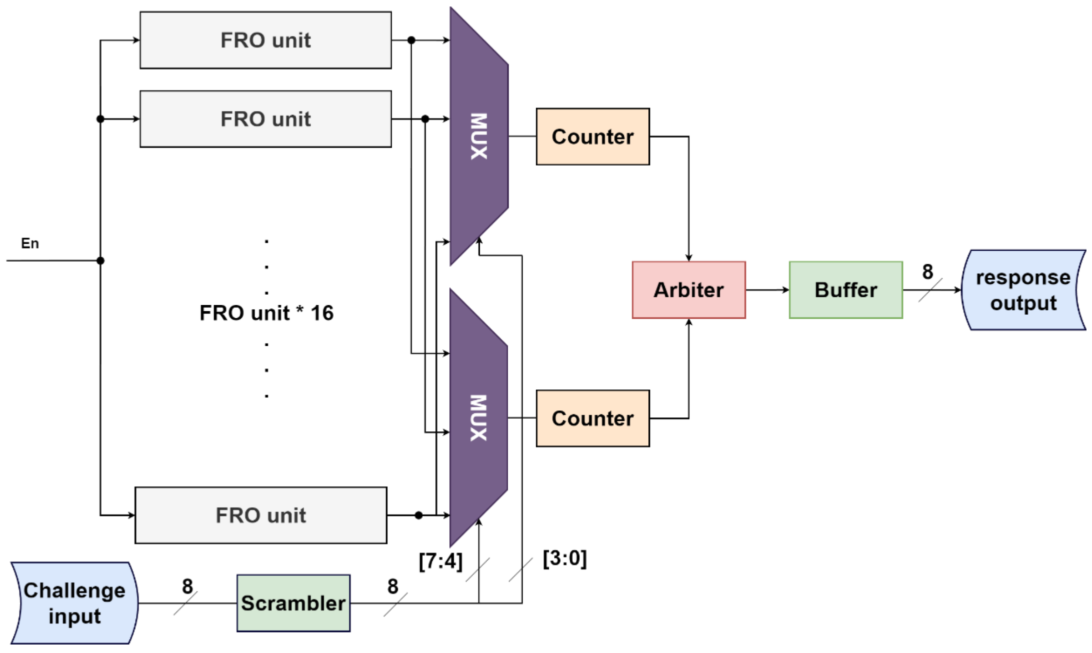

Fig 1. Overall schematic of the serial FRO PUF scheme

Background

A **Physical Unclonable Function (PUF)** is a security primitive that generates a unique device fingerprint based on the **uncontrollable process variations** introduced during chip fabrication. This intrinsic randomness makes PUFs highly resistant to duplication and tampering, making them ideal for secure key storage and authentication. A well-designed PUF should exhibit three fundamental properties:

1. **Uniqueness**: Each PUF should produce a response that is uniquely associated with the device, ensuring no two devices generate the same response for the same challenge. This property enables device authentication and differentiation.
2. **Reliability**: The PUF should produce **consistent** responses to the same challenge across different environmental conditions, such as variations in temperature and voltage. High reliability ensures that the response remains stable over time.
3. **Randomness**: The responses generated by the PUF should be highly unpredictable and resistant to modeling attacks. A strong degree of randomness prevents adversaries from reproducing or predicting responses through statistical analysis.

In challenge-response protocols, a PUF is mathematically represented as a function *R=f(C)*, where *C* is the challenge (input), and *R* is the response (output) generated by the PUF. These properties make PUFs a crucial component in hardware-based security applications.

Problem Description

You are asked to design six functional modules, (1) Feedforward Ring Oscillator, (2) 16 to 1 Multiplexer, (3) Scrambler, (4) Counter, (5) Race Arbiter, and (6) Buffer. In the following paragraph, we detail the structure and function of each module. During implementation, you **MUST** meet these requirements.

1. **Feedforward Ring Oscillator**: This part consists of 16 FROs as candidates for constructing a side of the FRO-PUF. Each FRO comprises 1 NAND gate, 3 AND gates, and 7 inverters, with a feedforward path introduced by the AND gates to influence the oscillation cycle(Fig 2). This feedforward structure increases sensitivity to process variations, enhancing the uniqueness of the PUF response.

   Since physical silicon is unavailable, process variations will be simulated by assigning different delay times to each gate. Instead of hardcoding delay values, the **delay time (#t) will be passed as a parameter DELAY to the module**, allowing flexibility in defining different FRO instances. Each of the 16 FROs should have distinct DELAY values specified in the Verilog file[4].

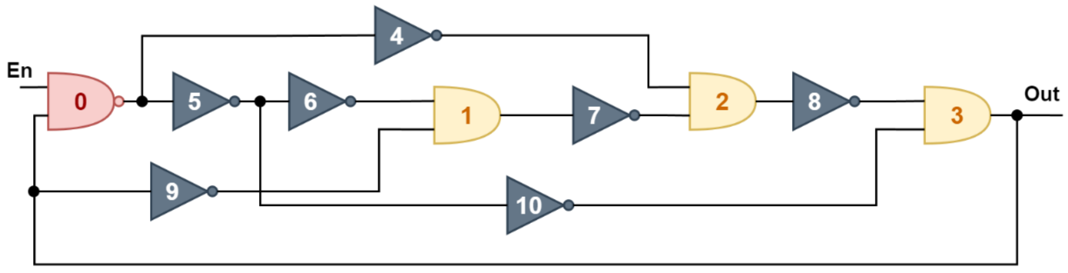

Fig 2. The structure of feedforward ring oscillator

2. **16 to 1 Multiplexer**: There are 2 MUXs, and each individually selects a candidate from 16 FROs using the select signal generated by the Scrambler module.
3. **Scrambler**: The purpose of Scrambler is to create pseudo-random selection signals for MUXs so that the unpredictability of PUF can be achieved. The scrambler receives an 8-bit challenge input and scrambles it using a Linear Feedback Shift Register (LFSR) to generate an 8-bit response key.

   In this Programming Assignment, you must implement the following Scrambler to evaluate its correctness. Additionally, you could implement other Scramblers and include them in your report, comparing their advantages and disadvantages with the original version as a **bonus**. However, **do not submit other Scramblers for correctness evaluation**.

   (1) Scrambler Overview (Fig 3)
   - The scrambler takes an 8-bit input (*In[7:0]*).
   - Function f processes the register (*reg[7:0]*), which is reset by input.
   - The register is clocked (*clk*), meaning it updates on clock cycles.
   - Function g takes the stored value and generates the final scrambled output (*Out[7:0]*).

   (2) Function f (Fig 4)
   - f defines how to update the register.
   - Each bit of *In* is involved in **resetting** the register.
   - Based on the feedback taps identified at positions *reg[7][3][2][1][0]*.
   - This new bit is typically shifted into *reg[7]* as the register shifts right (i.e., *reg[7]* moves to *reg[6]*, *reg[6]* to *reg[5]*).
   - This structure ensures data scrambling by spreading input variations across multiple register bits.

   (3) Function g (Fig 5)
   - g generates the final output using both:
   - The current input (*In[7:0]*).
   - The stored scrambled data (*reg[7:0]*).
   - The function applies an XOR operation between the register and input to produce the final scrambled output (*Out[7:0]*).

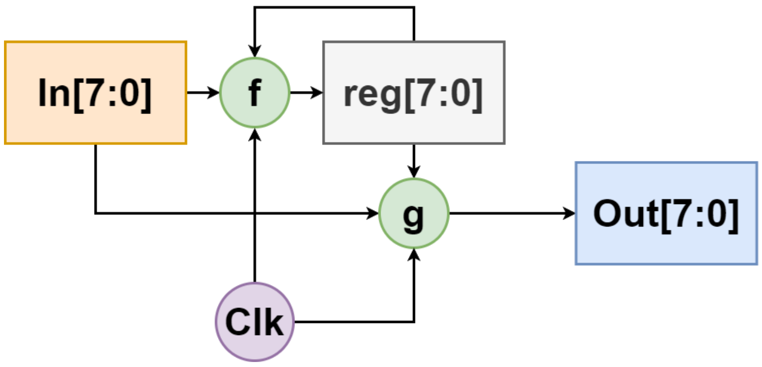

Fig 3. The structure of scrambler

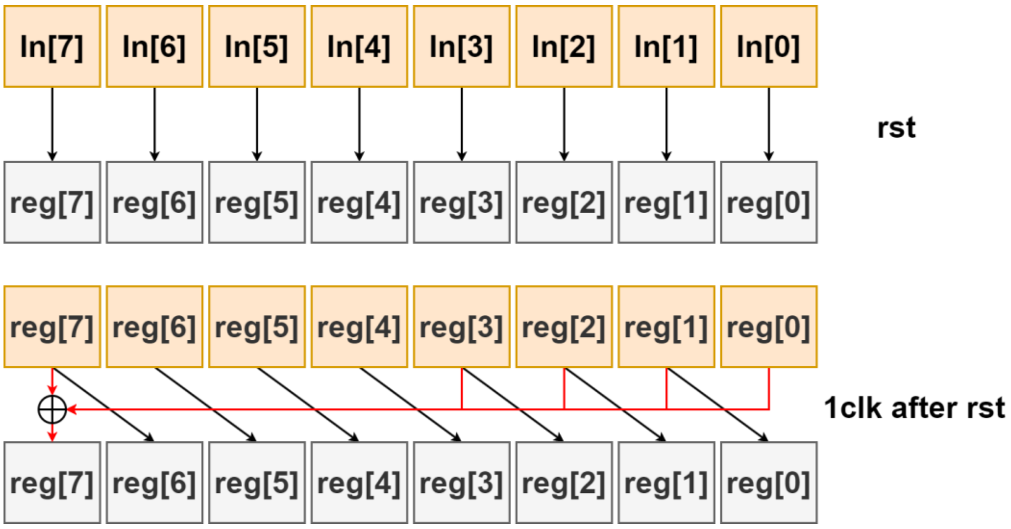

Fig 4. The structure of function f

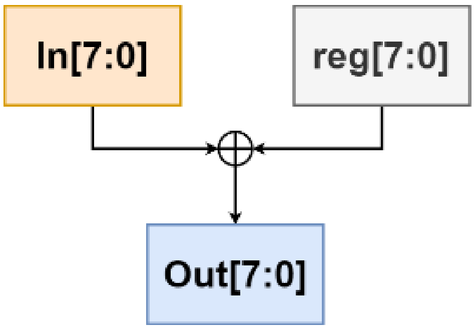

Fig 5. The structure of function g

4. **Counter**: The Counter module is responsible for counting up to a predefined value and signaling when it reaches the target.
   (1) The target count value will be determined by the parameter WIDTH, where the counter must reach 2^{WIDTH} -1 before signaling completion.
   (2) The finish signal is set to 1 when the count reaches 2^{WIDTH}-1, otherwise it remains 0.
5. **Race Arbiter**: This module determines which signal finished first. If Group A completes first, the output is set to 1. If Group B completes first, the output is set to 0. If they complete at the same time, the output is set to 1.

6. **Buffer**: This buffer stores each bit from each race and outputs an 8-bit response. It is also in charge of reset logic in order to ensure that the same input yields the same output every time it enters the same chip (Stability).

Evaluation

To assess the performance of PUF circuits, several quality metrics, including Reliability, Uniformity, and Uniqueness, have been presented. In this Programming Assignment, you should evaluate **Uniformity** and **Uniqueness**. Following will start the definition of two metrics and you **MUST** discuss the evaluation results.

1. **Uniformity (UF)**: It measures how evenly distributed the 1s and 0s are in the PUF response bits. Uniformity reflects the randomness of the response bit and is measured as a percentage of the response bit's Hamming weight (HW) according to the equation in (1.1)). 50% is the best value for uniformity:

$$Uniformity_i = \frac{1}{n}\sum_{j=1}^{n} u_{i,j} \times 100\% \qquad (1.1)$$

   where $u_{i,j}$ is the jth bit of n-bit response of ith chip. You only need to calculate your own simulated data.

2. **Uniqueness (UQ)**: This statistic assesses the difference of a PUF's responses to the same challenge (C) when implemented on several PUF chips (k). The uniqueness is calculated as the inter-chip variation of various responses using Eq. (1.2). 50% is the best value for UQ:

$$Uniqueness = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{HD(S_i, S_j)}{n} \times 100\% \qquad (1.2)$$

   where S_i is n-bit response of ith chip, S_j is n-bit responses of jth chip, k is the quantity of PUF chips, and HD(S_i, S_j) is the Hamming Distance between n-bit responses S_i and S_j.

   Fabricating simulation data is considered cheating and is strictly prohibited.

Bonus

In this section, you have two options, whichever option you choose, you must compare its advantages and disadvantages with the design in this Programming Assignment, considering factors such as uniqueness and uniformity.

1. Implement a Ring Oscillator PUF (RO PUF) that differs from this architecture (e.g., as seen in [3]).
2. Use a different Scrambler.

Requirement

You have to write this program in Verilog or SystemVerilog and use NC-Verilog 、 as the simulation platform. You MUST implement the assignment independently. If you refer to any external materials, you must cite them properly and demonstrate a thorough understanding of the content.

You must test your FRO PUF system with at least five different challenge inputs. All .v files should be submitted through ee-class. Please compress all your .v files as a zip file, and upload the .zip file to ee-class. The files you need to submit are as follows:

1. Your design, including Top.v 、 FRO.v 、 Scrambler.v 、 MUX_8to1.v 、 Counter.v 、 Race_Arbiter.v 、 Buffer.v 、 testbench and other modules (.v files) designed by your own if you have. You have to put all .v files in a .zip file named StudID_PA1.zip (ex: 123456789_PA1.zip) and upload to ee-class.
2. A report named as StudID_Name_PA1.pdf (ex: 123456789_陳聿廣_PA1.pdf) Note that the only acceptable report file format is .pdf, in either Chinese or English. In the report, you have to include at least:
   (1) How to compile and execute your program
   (2) Describe your design philosophy and every module
   (3) Please capture your simulation waveform and explain it (ex. where is your Response)
   (4) Show your calculation about **Uniformity** and **Uniqueness** and your analysis
   (5) **(Bonus)** A comparative analysis of the differences, advantages, and disadvantages of FRO [2] and RO [3]
3. All Verilog/SystemVerilog files must maintain consistent module declarations with the provided template, including **parameter declarations, inputs, and outputs**. Any deviation from this format will result in a deduction in the grading.

Grading

The grading is as follows:

1. Correctness of your code: 50%
   According to the content of the example ZIP file, there should be at least the specified input and output as stated in the file. Every module should meet its respective requirements.
2. The analysis of your PUF Design (**Uniformity** and **Uniqueness**): 30%
   (1) Explanation of Data Collection
   - Clearly explain **how the data was obtained** (e.g., testing methods, data sources, environmental conditions).
   - Provide details on the sampling method (e.g., number of PUF instances tested, number of test rounds, input conditions used).
   - If **automated data collection** was implemented using a specific program

or tool, describe the process and tools used.

(2) Presentation of Collected Data

- Clearly present **specific data samples** (e.g., tables, histograms, scatter plots).
- Ensure the data is presented in a clear and structured format with sufficient supporting data points.
- If there are outliers, they should be noted and reasonably explained.

(3) Interpretation of Results

- Explain the significance of the data
- Compare the design with the ideal case and provide reasonable hypotheses for any discrepancies.

(4) Justification of Observed Values

- Discuss why the values appear as they do. What factors influence the results?
- For example, how do architectural choices, circuit characteristics, testing conditions affect the outcome?
- If the results deviate from expectations, provide a reasonable explanation (e.g., possible hardware inaccuracies, entropy sources).

3. The report format and content: 10%

   This section will be ranked across the entire class. Reports that use visual elements such as diagrams and tables to enhance clarity and improve readability will receive higher scores.
4. Demo session: 10%
5. Bonus: 10%

   The bonus analysis includes a comparison between this architecture and the other Ring Oscillator PUF (RO PUF) like [3] or different Scramblers, evaluating their respective advantages and disadvantages. conduct a complete analysis to earn the full 10%

Please submit your assignment on time. Otherwise, the penalty rule will apply:

1. Within 24hrs delay: 20% off
2. Within 48hrs delay: 40% off
3. More than 48hrs: 0 point

Contact

For all questions about PA1, please send Email to 林威宏. Email to bb232399@g.ncu.edu.tw with the title [HW Security PA1].

Reference

[1] F. Zerrouki, S. Ouchani, H. Bouarfa, A survey on silicon PUFs, J. Syst. Archit.

127(2022), 102514.

[2] T. -K. Dang, R. Serrano, T. -T. Hoang and C. -K. Pham, "A Novel Ring Oscillator PUF for FPGA Based on Feedforward Ring Oscillators," 2022 19th International SoC Design Conference (ISOCC), Gangneung-si, Korea, Republic of, 2022, pp. 87-88, doi: 10.1109/ISOCC56007.2022.10031300.

[3] Suh, G.E., Devadas, S., 2007. Physical unclonable functions for device authentication and secret key generation, in: 2007 44th ACM/IEEE Design Automation Conference, IEEE. pp. 9–14.

[4] SystemVerilog ring oscillator simulation with different delays (https://electronics.stackexchange.com/questions/614234/systemverilog-ring-oscillator-simulation-with-different-delays)