Predicting the Output of Unpredictable Microelectronic Circuits

MSML 604 Project

March 28, 2023

Abstract

In this project, you will formulate and solve optimization problems to predict the output of a microelectronic circuit. Inside the circuit, there are two delay paths, and a comparator that compares the delay in the two paths. The comparator's output is either 0 or 1, indicating which path is faster. The routing of the delay paths in the circuit, however, is dependent on the circuit's input. You are allowed to query the circuit, but you do not have access to the internal parameters of the circuit. You do not need any background in microelectronics as the mathematical model of the circuit is already built for you. Your goal is to predict output of the output of the comparator. This is a group project, and you are encouraged to try as many formulations and numerical methods as possible as a group.

1 Introduction

The type of circuit being discussed here belongs to a type of circuits called "physical unclonable functions", or PUFs. PUFs take advantage of the manufacturing variations of microelectronic devices (e.g. transistors) to produce an unpredictable output, called the "response", to each distinct input value, which is a Boolean vector and is also called the "challenge". We call a certain challenge vector and its associated response bit/vector a challenge-response pair (CRP).

The specific type of circuit being discussed in this project is a delay-based PUF. The circuit shown in Fig. 1 on the left is the overall topology of the PUF. The leftmost "step up" indicates the starting point of both delay paths where the signal value switches from logic 0 to 1 at time 0. Each trapezoid is a multiplexer (or "mux"), which chooses its output from either of its inputs (labeled as input 0 and 1 as shown) according to the select bit (drawn on the bottom of each mux). There are two parallel muxes at each stage and they share the same select bit, which is one bit from the challenge vector. In other words, the challenge vector provides the select bits for each stage of muxes. At the end of the circuit, there is an "arbiter", or a delay comparator, which determines the response r. $r = 1$ if the upper signal arrives (i.e. switches from 0 to 1) earlier, and 0 otherwise.

Let us then take a closer look at each stage of muxes as an entirety, as shown in the right pane of Fig. 1. Each stage has two input bits and two output bits. As both muxes share the same select bit, they will either take input 0 or input 1. If the select bit is 1, then both muxes choose input 1, which means the upper input of the stage goes to the upper output, and the lower input goes to the lower output. In other words, the paths do not cross. On the other hand, when the select bit is 0, then the paths will cross. Essentially, each stage act as a switch box for the two input signals.

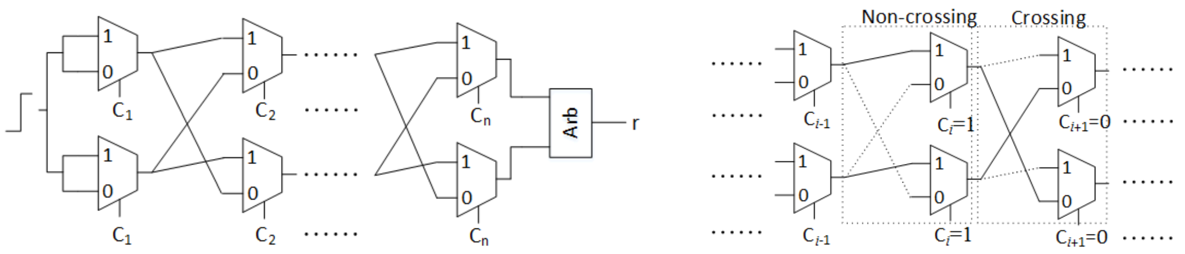

Figure 1: Structure of the delay-based PUF circuit (left) and the crossing/non-crossing scenarios of each stage (right)

There are delays when the signals propagate through each switch box. Although muxes and wires both contribute to the delay, we only consider the mux delays because they are orders of magnitude greater than the wire delays. Due to the manufacturing variation of microelectronic devices, the delay from each mux input to the output is different and is also dependent on the select bit. In other words, for each stage, if the select bit is 1 (0 resp.), the two delays from input 1 (0 resp.) to the mux outputs are added to the path delays. The delay in each path equals the sum of mux delays that the path goes through.

The outputs of PUFs are supposed to be unpredictable. More formally put, one should not have more than negligible above 0.5 probability in successfully predicting the output of the PUF of any previously unseen challenge vector, given that the PUF is queried any polynomial number of times. However, due to the linearly additive delay that we have seen in the given delay-based PUF, the unpredictable output property is not upheld. In this project, you will use the optimization theory and techniques that you have learned in this class to predict the output of this PUF to any challenge vector given a small number of queries.

2 Mathematical Model of the PUF

We denote the difference between the cumulative delay of the upper path and that of the lower path after the i^{th} stage by Δ_i, and the incremental delay difference added by the i^{th} stage in the non-crossing or crossing case by δ_i^1 or δ_i^0, respectively. Then, for each stage, the relation between the cumulative delay difference and the incremental delay difference is

$$\Delta_i = \begin{cases} \Delta_{i-1} + \delta_i^1 & \text{if} \quad c_i = 1 \\ -\Delta_{i-1} + \delta_i^0 & \text{if} \quad c_i = 0 \end{cases} \text{for } i = 1, 2, \ldots, n \tag{1}$$

where n is the number of stages in the PUF, $\Delta_0 = 0$, and $\mathbf{c} = (c_1, c_2, ..., c_n)$ is the challenge vector. With this inductive equation, the cumulative delay after the n^{th} stage can be derived as the following:

$$\Delta_n = \mathbf{\Phi} \cdot \omega \tag{2}$$

where $\mathbf{\Phi} = (\phi_1, \phi_2, ..., \phi_{n+1})$ is a function of the challenge vector, called the feature vector; and $\omega = (\omega_1, \omega_2, ..., \omega_{n+1})$ encodes the manufacturing variations of all the delay paths in each stage of multiplexers and is called the weight vector. The components of these vectors are given by

$$\phi_i = \phi_i(\mathbf{c}) = \prod_{j=i}^{n} (2c_j - 1) \ \text{ for } i = 1, 2, \ldots, n, \ \phi_{n+1} = 1 \tag{3}$$

$$\begin{aligned} \omega_i &= \frac{\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2} \text{ for } i = 2, 3, \ldots, n, \\ \omega_1 &= \frac{\delta_1^0 - \delta_1^1}{2}, \ \omega_{n+1} = \frac{\delta_n^0 + \delta_n^1}{2} \end{aligned} \tag{4}$$

We can show that (3) and (4) lead to (2) by proof of induction on n.

The response bit is determined by the arbiter, which simply compares the delay of the two paths and determines which path is faster, i. e. whether Δ_n is positive or negative:

$$r = \begin{cases} 1 \text{ if } \Delta_n > 0 \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

This model is provided to you in the form of MATLAB code so you do not have to recreate it.

3 MATLAB Code for this Project

A MATLAB code framework for the project is given to you. "predict.m" contains the flow of this problem. The PUF parameters are generated and the syntax of querying the PUF is given to you. You need to work two sections:

1. Generate your training dataset. You can use the "puf_query" function to obtain the correct response of a challenge vector. Keep track of the number of training samples using the "training_size" variable. Generation of the training data at this stage does not count towards your training time.
2. Estimate the weight values in the PUF. The estimated value is denoted by w_0. You are allowed to query the PUF in the training process. This query will count toward the training time, and you also need to increment the "training_size" variable each time you query the PUF.

"puf_query.m" contains the mathematical model of the PUF, and "weight_diff.txt" contains a list of random numbers that the weights are taken from.

The time that your code takes to obtain the estimated weight w_0 is calculated. The test data will be generated randomly. The targeted success rate is 99%. For each 0.01% of success rate below 99%, 1 second of penalty time will be added to your training time.

4 Project Requirements and Grading

4.1 Due date and time

Thursday, April 27 at 11:59 pm.

4.2 Group project

This is a group project. Each group can have no more than 4 people. Solo groups are allowed. However, collaboration is encouraged because the variety of algorithms attempted is a grading metric. You must sign up for groups on Canvas.

4.3 Deliverables

The deliverables of this project include a report (.pdf), code (.m), and a recorded presentation (video or voice-embedded PowerPoint formats).

4.3.1 Report

Your report should have the following contents.

1. Formulation. Describe how you are going to estimate w. Write down the optimization problems involved. For example, linear programming, quadratic programming, SVM, neural networks all count as different formulations.
2. Numerical methods. Describe the numerical methods that you are going to use to solve the optimization problem. You are allowed to use built-in functions of MATLAB and other MATLAB packages you can find. For example, using a built-in MATLAB function or a self-written gradient descent function to solve for the same optimization problem count as two different methods.
3. Experiment results. Document the results (success rate, training time, and training set size), as well as anything else that you deem is of interest, for all the formulations and algorithms that.
4. Team evaluation (non-solo groups only). Document the responsibility of each team member. Every member also needs to write a short evaluation (1 2 sentences) about each team member including themselves.

4.3.2 Code

You need to submit your version of "predict.m" that contains ONE formulation and algorithm that you believe has the best performance. Please include a separate README file if your code needs anything more than hitting the "Run" button to run. All the submitted code will be run on the same computer with the same MATLAB version, and the performance will be recorded.

Notice that the actual weights w is exposed in the code. While you can feel free to look at it, the only way you can use it in the code is to use it in the "r =puf_query(c,w)" function.

4.3.3 Presentation

Please prepare a presentation about 10 minutes long that introduces ALL the formulations and numeric methods that you have used.

4.4 Grading

You grade will include the following parts:

1. **Completeness of deliverables: 25%.** If you submitted all the 3 items and they all look reasonable, you will get the full points. If your submission is incomplete (i.e. contents missing from the report, or your code does not run), points will be deducted.

2. **Diversity of solutions: 25%.** Your first formulation and numeric method will earn 10 and 5 points, respectively. Each additional formulation and method will earn up to 10 and 5 points, respectively, depending on their dissimilarity with your other solutions. Those that are not validated by experiment results will only earn half of the points. Your points may exceed 25 in this category, in which case you get extra credit!

3. **Performance: 25%.** This score will be based on your submitted code. There are two main performance metrics: effective training time and number of training samples. Any solution that is *Pareto-optimal* will get full pints. Your points will be "$25 - 5\times$ # of points your solution is sub-optimal to" if it is not optimal. See illustration in Fig. 2.

4. **Contribution: 25%.** This measures your contribution in your group. You will get full points as long as you made reasonable contribution to your group.

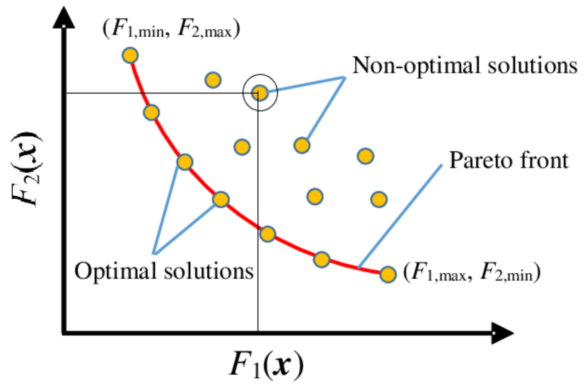

Figure 2: Demonstration of Pareto optimality. Your solution is Pareto optimal if no other solution is both faster and takes fewer samples than yours. The circled data pint is sub-optimal to (or dominated by) the 4 datapoints inside the rectangle.

5 Good luck!