Cole Maxwell (25017014): ECE 658

Lab 5: TBD

December 2020

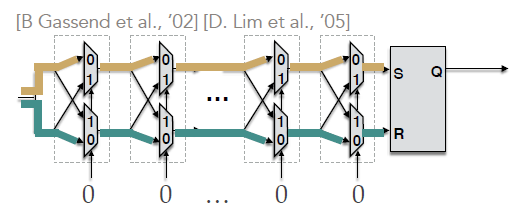
Bitslice Implementation of an Arbiter PUF

**Report Section 1: Problem Statement with Objectives**

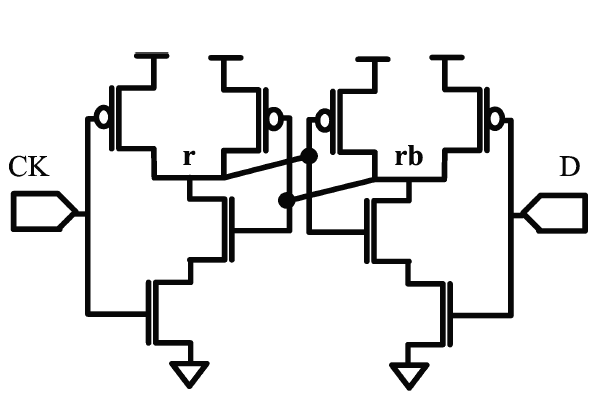
Given the circuitous, multi-step route that newly fabricated ICs take from manufacturer to consumer, it is crucial to verify the authenticity of the delivered chip. The global distribution network that ICs are subject to leaves room for counterfeiting by illegitimate parties, an unfortunate reality that costs the semiconductor market an estimated $5B per year.

One technique to ensure the authenticity of ICs is to use Physical Unclonable Functions (PUFs). A PUF provides a biometric signature for an IC that can be documented immediately after manufacturing and verified to match once the chip arrives at its destination. Like using a fingerprint to verify someone’s identity, an effective PUF must be specific to that IC instance and must not change over time. To ensure the uniqueness of each ICs signature, a PUF should be determined solely by random, unpredictable process variations that are intrinsic to fabrication.

In this report, I will design and evaluate the layout of an Arbiter PUF based on the following schematics provided in Module 9 of the course.



*Figure 1: Gate-level Schematic of an Arbiter PUF*



*Figure 2: Transistor-level Schematic of the Arbiter Circuit*

*Figure 1* shows a series of *m* stages, where each identical stage contains two 2:1 multiplexers, and the abutment of the bitslices will serve as the challenge circuit for each challenge-response pair in the design. *Figure 2* shows the transistor-level schematic for the arbiter circuit, which is implemented as two cross-coupled NAND gates. The arbiter is placed at the end of the bitslices and is used to determine the response in each challenge-response pair.

The Arbiter PUF is considered a *strong PUF*, because it the select bit in each stage can be set to 0 or 1, providing 2m combinations of possible challenges to be issued, each generating its own response. Looking again at *Figure 1*, there are four paths within each stage, and the values of the select bit for each stage (given by the challenge) will determine two paths through the bitslices that end at the arbiter circuit. Theoretically, if each path has the same length (and the same delay), we would expect the signals to arrive to the arbiter at the same time. However, due to process variations during fabrication, the delay of each path will be slightly different, and either signal S or signal R will arrive first.

It is the role of the arbiter to record which signal arrives first and output Q accordingly. If R arrives first, the arbiter’s output, Q, will be set to 0. If S arrives first, Q will be 1. Thus, if the layout is created so that each path is measured to be exactly the same length (and have the same delay), the process variations will solely determine which signal arrives first. The challenge-response pairs generated after manufacturing provides a fingerprint unique to this circuit.

Since manufacturing is not within scope, the goal of this report is to create a layout of one stage of the circuit as well as the arbiter, where the length of each is as close as possible. After abutting 8 stages together and extracting a netlist, a series of challenges can be issued using the .VEC function in HSPICE, and the delays and responses can be measured to see how well the paths match. A 128-stage Arbiter PUF would be a better design, but for the purposes of this lab, I will use 8 stages that provide 28 = 256 possible challenge-response pairs.

**Report Section 2: Experimental Plan**

Step 1: Create Schematics and Symbols

I will create schematics for the arbiter circuit and one stage of the PUF using Cadence Schematic L. Then, I will create a symbol for each schematic, and create another 8-stage schematic.

Step 2: Check Functionality of Schematics

In this step, I will verify that the design works correctly by extracting a SPICE netlist and using HSPICE to perform transient simulation using the .VEC feature.

Step 3: Layouts

In this step, I will generate layouts for the arbiter circuit and one stage of the PUF using Cadence Layout XL with the goal of creating consistent path lengths. The results from this step will be images of the layouts with measurements annotated and screen captures of the design that passes DRC and LVS.

Step 4: Parasitics Extraction

In this step, I will extract a netlist from the layout with parasitics and repeat the simulation from step 2.

Step 5: Delay Results

Finally, I will use Cscope waveforms to determine which signal arrives to the arbiter first based on the output value Q, and repeatedly adjust the layout to get the delays to match as best as possible.

**Report Section 3: Results and Analysis**

TBD: