University of Maryland Baltimore County
CMPE/ENEE 691
Hardware Security
Spring 2023

HW7: PUFs
Due Date: 13/04/2023

Deliverables:

1) Your transistor level Design (.sp file).
2) A ReadMe describing how to run your design.
3) A Makefile to automate the run the PUFs if applicable.
4) Output files (10 files totally) showing the digital voltage values of the outputs of each PUF for each challenge.
5) The 10 analog output files (PUF1_out.txt … PUF10_out.txt)
6) The simulation output (**.lis)** file
7) The Evaluation metrics: Uniformity, Intra Hamming Distance for each PUF, and Inter Hamming Distance between each 2 PUFs.

In this homework, you are to design an arbiter-PUF with a 16-bit challenge and an 8-bit output. The design is done in the transistor level and simulation is performed using Synopsys HSPICE. Please check the Tutorial uploaded in blackboard to learn more about using HSPICE. You can find the definitions of PUF metrics in "PUF_Evaluation_Metrics.pdf" uploaded on blackboard.

For this homework, please take the following steps:

1) Design an arbiter-PUF with a 16-bit challenge and an 8-bit output. The PUF should consist of 8 arbiter chains, each providing a 1-bit output, for a total of an 10-bit output. An arbiter chain is shown at the end of this document.
2) Simulate 8 process-varied samples of your PUF (PUF$_1$, PUF$_{2,}$ … PUF$_8$). Please follow Notes 1 and 2 for this step.
3) Evaluate the PUFs using "uniformity", "intra-Hamming Distance" and "inter-Hamming Distance" metrics.

Note 1: Use the GAUSS model to introduce process variation to *LMIN, vth_pmos, vth_nmos, toxm_pmos and toxm_nmos*, and use SWEEP MONTE = 8 to run all 8 PUF samples. Refer to the "Process Variations" section in the HSPICE tutorial (pages 10 and 11) for more information.

The simulation results will be written to the output (**.lis**) file. The results for each run will be headed with **monte carlo index = x** where x is the number of the Monte Carlo run (the PUF sample number). Open the output file and search for the word 'monte' to find the results for monte = 1, 2, 3, … 10. Extract the output voltage values of each run into a separate file (e.g PUF1_out.txt, PUF2_out.txt … PUF3_out.txt). The voltage values should be put in the space-separated format shown in Figure 1.

```
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
```

```
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
v0 v1 v2 v3 v4 v5 v6 v7
```

Figure 1: analog output file layout

where v0 is the output of the 1st arbiter chain, v1 is the output of the 2nd arbiter chain … v7 is the output of the 8th arbiter chain.

v0 v1 v2 v3 v4 v5 v6 v7 are the output voltages of the 1st challenge
v0 v1 v2 v3 v4 v5 v6 v7 are the output voltages of the 2nd challenge
v0 v1 v2 v3 v4 v5 v6 v7 are the output voltages of the 3rd challenge
…
v0 v1 v2 v3 v4 v5 v6 v7 are the output voltages of the 15th challenge

An example of the analog output file (PUF1_out.txt) you need to produce is shown in Figure 2. Note that you need a total of 8 of these files, 1 for each PUF run.

```
0.0000e+00 0.000e+00 4.395e-05 4.395e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
2.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 1.100e+00 4.393e-05
4.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
6.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.4000e-08 0.000e+00 1.100e+00
8.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 2.2000e-08 0.000e+00 1.100e+00
0.0000e+00 0.000e+00 4.395e-05 4.395e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
2.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 1.100e+00 4.393e-05
4.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
6.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.4000e-08 0.000e+00 1.100e+00
8.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 2.2000e-08 0.000e+00 1.100e+00
0.0000e+00 0.000e+00 4.395e-05 4.395e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
2.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 1.100e+00 4.393e-05
4.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.100e+00 4.394e-05 4.395e-05
6.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 1.4000e-08 0.000e+00 1.100e+00
8.0000e-09 0.000e+00 1.100e+00 4.394e-05 4.395e-05 2.2000e-08 0.000e+00 1.100e+00
```

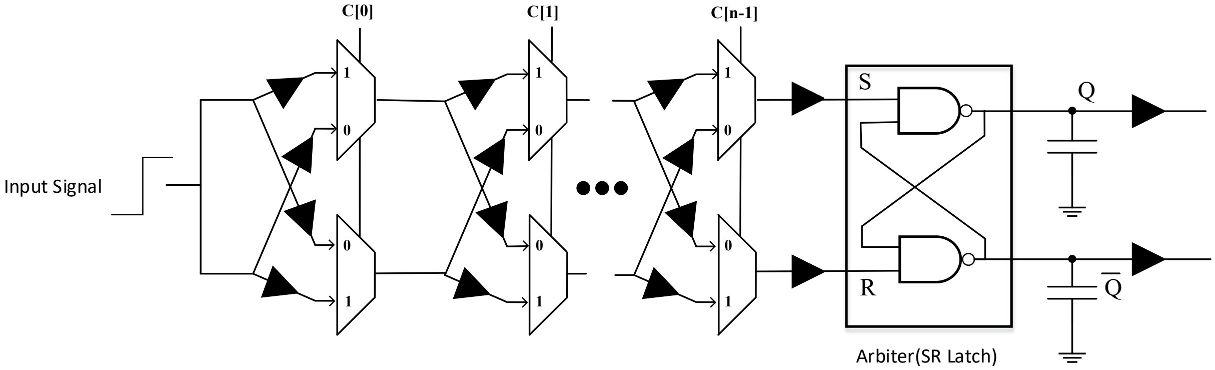
Figure 2: analog output file example

As can be seen in Figure 2, the values you will put in PUF1_out.txt … PUF8_out.txt are analog values. You can change them to digital values manually or by using the provided script "AtD.py". Note that this script will only work on the specified matrix format in Figure 1.

Note 2: You need to apply the challenges which given in "challenges.txt". To implement this in your SPICE script, connect your challenge input pins to PWL voltage sources (each challenge bit will be controlled by its own PWL source). You can use the provided script "CtP.py" to generate the PWL statements from the digital values in "challenges.txt". Each PWL statement (for Vc0 to Vc15) describes a voltage source to be used to supply a challenge pin.

Useful hints for your implementation:

1- Please implement the arbiter PUF as follows (by adding buffers) to be able to model the effect of process variation in wires (via added buffers):



2- Please use *'BUF_X3M'* buffer in the gate.txt file you are given to realize the buffers shown in the above picture.

3- It is recommended to use step = 0.5 ns in the command *'.TRAN'*.

4- The PUF response is extracted from node *'Q'* as shown in the picture.

5- In the class slides, A DFF was used as an arbiter. As the process variations may not be enough for the 16-bit PUF you implement, please use an SR-Latch based arbiter (shown in the above picture) instead of the DFF one in the slides.

6- As a load use the two capacitors shown in the picture, each 100 Femtofarad (fF).

7- In the gate.txt the transistor level model of all gates including Multiplexer, Buffer, SR latch are given. You do not need to design these gates yourself. You can just instantiate them from this file.

8- Please make an appropriate *input pulse* for the input of your PUF, i.e., each 5ns the challenge is changed, so you should consider this time for making an appropriate input pulse.

Please note that the buffers at the Q and $\bar{Q}$ output of PUF are not connected to any other node.