

EEL4720 Final Project Report

(Note: the annotations are in blue and italic to make them more distinguishable from the slide text.)

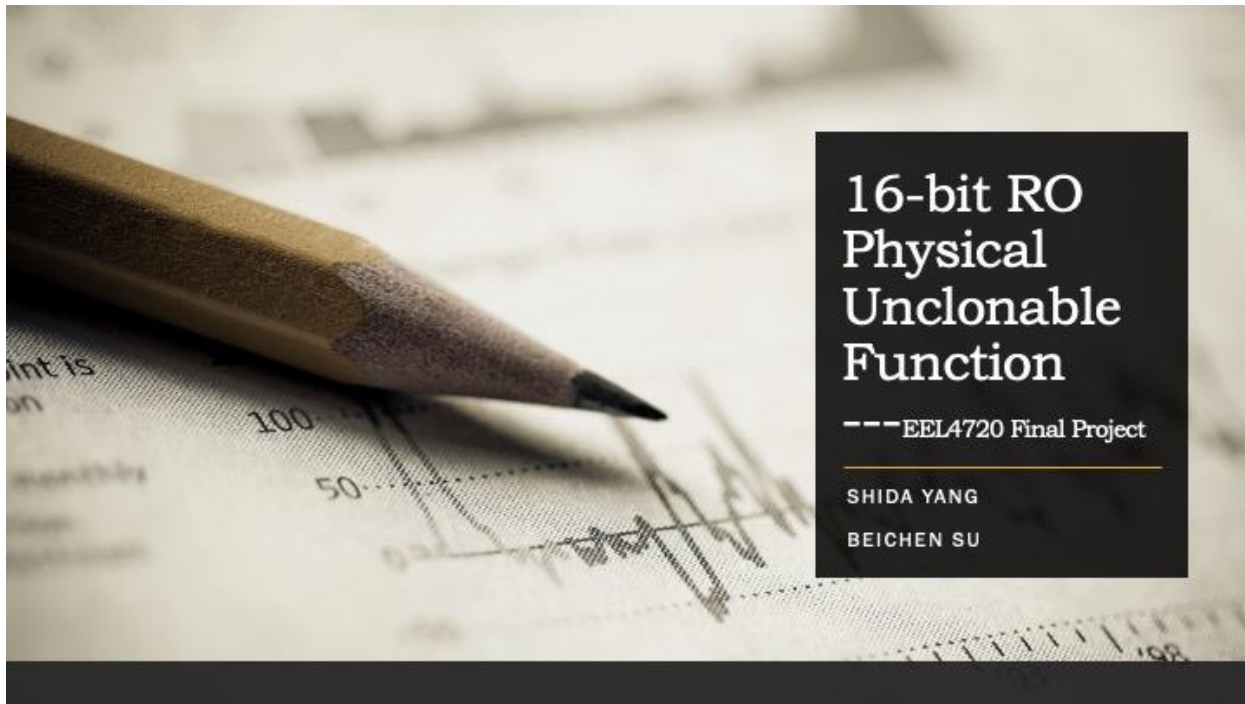


Table of Content

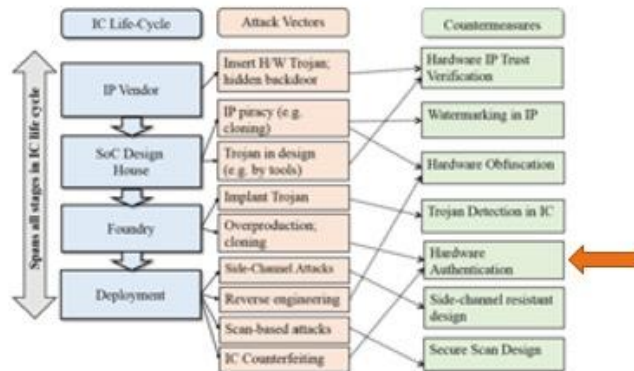
- Introduction to the application
- Design and Implementation
- Testing and demonstration
- Test Results
- Problem Encountered
- Potential Solution and New Results
- Conclusion

We discussed first five topics during progress report. We found a way to partially solve the problem afterwards, which was mainly discussed in final meeting.

Introduction

□ Nowadays, people rely more and more on their digital devices, which stored so many important privacy information like radio-frequency identification (RFID). **Hardware security is becoming increasingly important.**

□ Our application is mainly focused on hardware authentication



In the whole process of IC lifecycle, there are a lot of potential attack vectors exist along with its countermeasures. Our application is mainly focus on Hardware Authentication

Introduction

□ In traditional methods, secret keys are stored digitally in a nonvolatile memory which is always vulnerable based on hardware implementation and key storage.

□ For extremely resource contained platform such as radio-frequency identification (RFID), even simple cryptographic operations can be too costly.

□ Software-only protection is not enough. Non-volatile memory technologies are vulnerable to invasive attack as secrets always exist in digital form.

□ adversaries can physically extract secret keys from EEPROM while processor is off

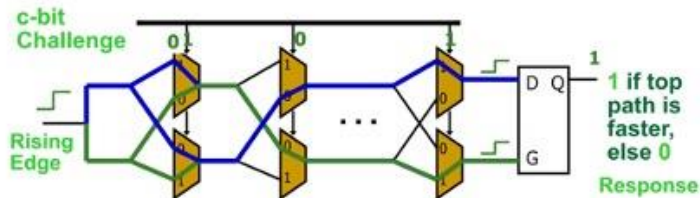
□ Due to the differences between each devices (length, width, oxide thickness) in circuit and system (**Process Variation**), each IC has unique properties.

Introduction

Physical Unclonable Function (PUF) is a function that is:

- ☐ Based on a physical system
- ☐ Easy to evaluate (using the physical system)
- ☐ Outputs looks like random function
- ☐ Unpredictable even for an attacker with physical access

Path delays in an IC are statistically distributed due to random manufacturing variations

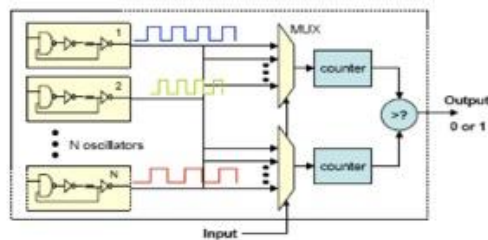


This is the structure of the basic physical unclonable function, what we designed are shown below.

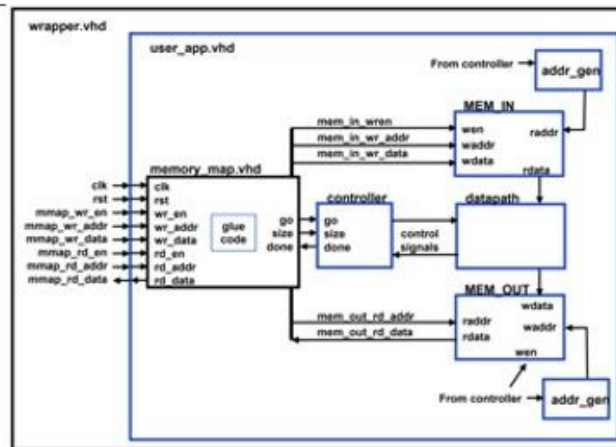
Introduction

Ring-Oscillator (RO) PUF relies on delay loops and counters instead of MUX and arbiters because it will be more stable on FPGA.

Our design is implementing a 16-bit RO PUF. Input (Challenge) is 16 bit and the output is 1 bit. We store 32 different 16-bit challenges, generated by a random generator function, in RAM and pass through all challenges one by one to form a 32-bit response sequences.

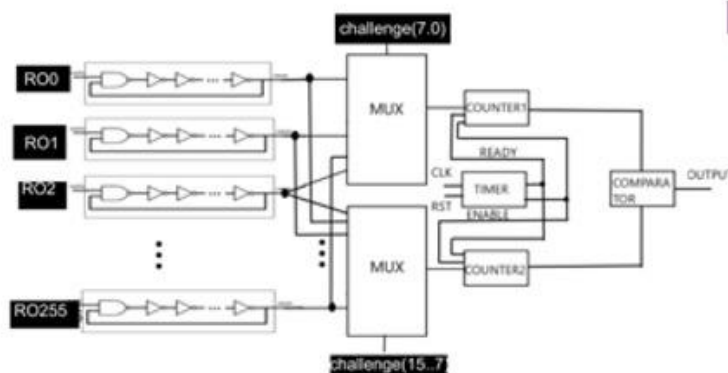


Design and Implementation



The `user_app` design is basically same as our lab 5, the only differences will be the data path, which is shown below. Later in the report, we will show how we are testing for this design. There are $25 \times 32 \times 6 = 4800$ challenges input, so we need to use RAM to store the input and output.

Design and Implementation



Our design includes 256 ring oscillators, each path includes 30 not gates and a nand gate. The outputs of ring oscillator connect to two MUX, each of them takes an 8-bit input. The output of the MUX will be connected to the counters to calculate the times the output becomes 1. After certain time period, which is

controlled by timer, the comparator will compare the results from two counters. The output will be 1 if counter1 larger than counter2, 0 otherwise.

Test and Demo (Method and Goal)

We will use intra-Hamming Distance and inter-Hamming Distance to test our design.

What is Hamming Distance?

- Given two numbers, calculate the fraction of bits that are different
- For example, use two 8-bit number
 - 0b11001010
 - 0b10001011
 - 2 different bits, so $HD=2/8=0.25$

Test and Demo (Method and Goal)

Intra-HD

- Give same set of input to the same chip multiple times
- Compare the HD between the outputs
- Tells how reliable the design is
- Ideally close to 0 because a reliable design should have the same output for the same input

$$\overline{\text{Intra} - d_{HD}}(k) = \frac{2}{k(k-1)} \sum_{m=1}^{k-1} \sum_{n=m+1}^k \frac{HD(R_{i,m}, R_{i,n})}{n}$$

Inter-HD

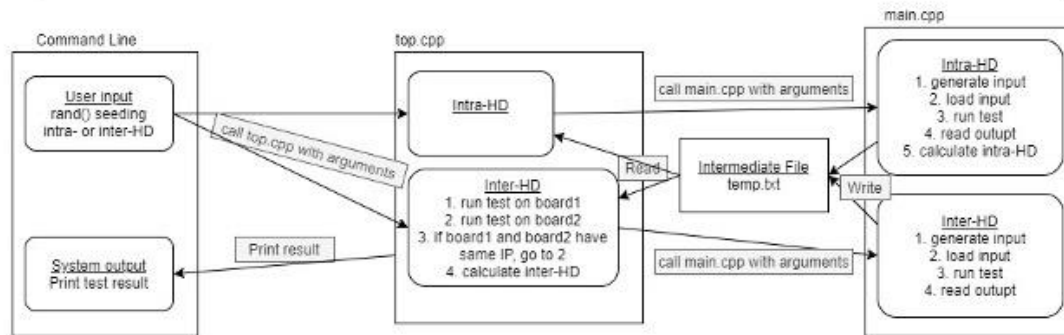
- Give same set of input to different chips
- Compare the HD between the outputs across chips
- Tells how unique each chip is when implementing the same design
- Ideally close to 0.5 because each bit has two possible outcomes, so half of the bits should be different on average

$$\overline{\text{Inter} - d_{HD}}(k) = \frac{2}{k(k-1)} \sum_{m=1}^{k-1} \sum_{n=m+1}^k \frac{HD(R_{i,m}, R_{j,n})}{n}$$

$R_{x,y} = y^{\text{th}}$ response of board x

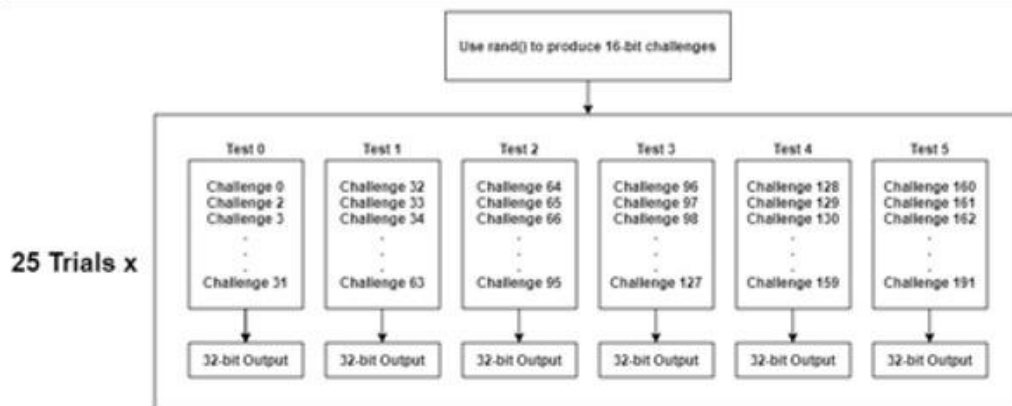
To see how our design performs, we evaluate two criteria. The intra-HD tells how reliable our design is, so it should ideally be zero. The inter-HD tells how unique each FPGA is when implementing our design, so it should be close to 0.5.

Test and Demo (Data and Control Flow)



Because we need data from two different boards to calculate inter-HD, I used a top.cpp program to help run the code on the FPGA and process the output data from different boards.

Test and Demo (main.cpp – Input/Output)



Since each challenge produces one 1-bit output, we group 32 challenges together to produce a 32-bit output. We have $6 \times 32 = 192$ challenges, so we can produce 6 different 32-bit outputs. This test is performed 25 times so that we can get the average value for inter-HD and intra-HD.

Test and Demo (main.cpp – Intra-HD)

- 25 outputs in the same test from 25 trials
 - Output 0 ~ Output 24
- Pairing output (every possible combination)
 - Output 0 – Output 1, Output 0 – Output 2, Output 0 – Output 3, Output 0 – Output 4, ...Output 0 – Output 23, Output 0 – Output 24
 - Output 1 – Output 2, Output 1 – Output 3, Output 1 – Output 4, ...Output 1 – Output 23, Output 1 – Output 24
 - Output 2 – Output 3, Output 2 – Output 4, ... Output 2 – Output 23, Output 2 – Output 24
 -
 - Output 23 – Output 24
- For each pair, calculate intra-HD
 - Find number of different bits (n), divide by total number of bits (32)
 - $n/32$
- Average the intra-HDs

Test and Demo (top.cpp – Inter-HD)

- Keep doing test until get results from two different boards
- 25 outputs from board 1, 25 outputs from board 2
- Pair them up just like in the previous slide
 - First element in the pair is from board 1, second element is from board 2
- For each pair, calculate inter-HD
 - Find number of different bits (n), divide by total number of bits (32)
 - $n/32$
 - Same algorithm for calculating Hamming Distance, just the pairs are from different boards
- Average the inter-HDs

These two slides show how the outputs are paired when calculating the hamming distances.

Test and Demo (Command Line)

- `./top.o RAND_SEED INTRA_OR_INTER`
 - `RAND_SEED`: make sure the same set of input is given to the board
 - `INTRA_OR_INTER`: 0=inter-HD, 1=intra-HD
- Printed Result
 - 6 tests, each test is the average of 25 trials



This is how we use the terminal to run our test.

Note: when `RAND_SEED=0`, I use `time(NULL)` as the seed, so it will be complete random.

Result (Intra-HD)

```
[shidayang@ece-b312-recon proj_sw]$ ./top.o 1125 1
Doing Intra-HD Test:
-----
Start running Board
.....
Finish running Board1
-----
Board: 192.168.1.110:
Test 0: 0.0025
Test 1: 0.006875
Test 2: 0.0025
Test 3: 0.01625
Test 4: 0.0025
Test 5: 0
[shidayang@ece-b312-recon proj_sw]$
```

```
[shidayang@ece-b312-recon proj_sw]$ ./top.o 1125 1
Doing Intra-HD Test:
-----
Start running Board
.....
Finish running Board1
-----
Board: 192.168.1.107:
Test 0: 0.0025
Test 1: 0.015625
Test 2: 0.0025
Test 3: 0.0025
Test 4: 0.0154167
Test 5: 0.0025
[shidayang@ece-b312-recon proj_sw]$
```

Result (Inter-HD)

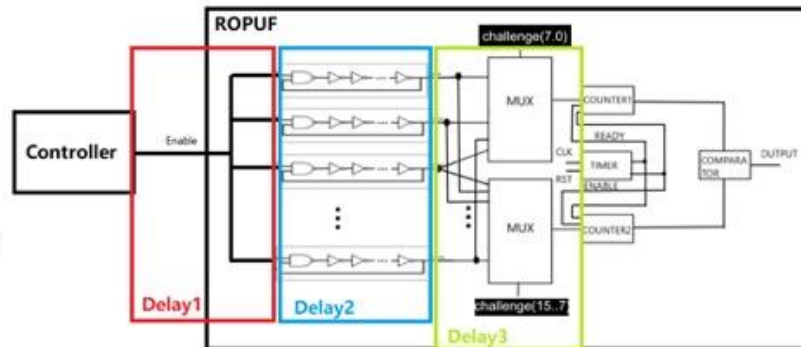
- The average inter-HDs we measured from our boards are far from ideal
 - Should be around 0.5 ideally
 - Our data ranges from 0.03 to 0.18
 - The implemented design favors certain ROs over others (explained in the next slide)

```
[shidayang@ece-b312-recon proj_sw]$ ./top.o 845 0
Doing Inter-HD Test:
-----
Start running Board1
.....
Finish running Board1
-----
Start running Board2
.....
Finish running Board2
-----
Board1: 192.168.1.103
Board2: 192.168.1.108
Test 0: 0.03125
Test 1: 0.185
Test 2: 0.0911458
Test 3: 0.069375
Test 4: 0.1875
Test 5: 0.165208
[shidayang@ece-b312-recon proj_sw]$
```

These are our initial testing results. The intra-HD is very good, so our design is reliable. However, the inter-HD is far from 0.5, so our design does not reflect the uniqueness of each FPGA.

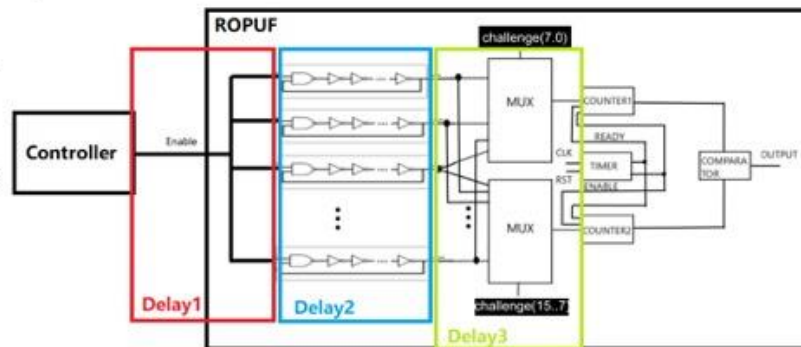
Problem Encountered

- The output bit ultimately depends on the **difference** between the propagation delays along different paths
 - The choice of path is determined by the challenge



Problem Encountered

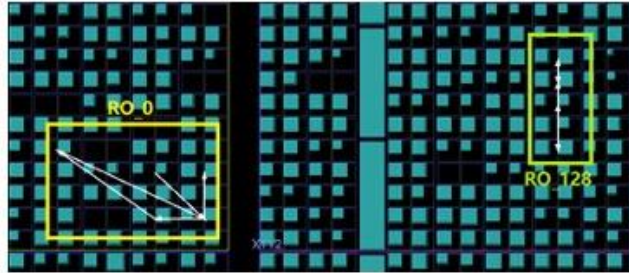
- Total delay = Delay1 + Delay2 + Delay3 + PV
 - Delay1: enable signal from the controller to the ROs
 - Delay2: signal traveling through the internal routing of the ROs
 - Delay3: RO output to counters
 - PV: process variation
- Successful ROPUF design should be symmetric enough to keep Delay1, Delay2, and Delay3 the same so that the output only depends on PV



The output ultimately depends on the difference between the delays when the signal goes through different paths. In a successful ROPUF design, PV should be the only factor that affect the output, so the paths from the controller to the 256 ROs (Delay1) should be identical, the paths inside each RO (Delay2) should be identical, and the paths from the ROs' outputs to the counters (Delay3) should be identical.

Problem Encountered

- The internal routing of each Ring-Oscillator is different when using automatic place and route
 - If a challenge is 0b10000000 00000000
 - Mux1 selects RO_128, Mux0 selects RO_0
 - Internal routing lengths are different, making Delay2 different



Problem Encountered

- Internal Routing (Delay2) is a major source of delay because we have many NOT gates in each RO
- Ideal cases, when putting design on different boards, only PV should change
- The difference between Delay2's along different paths makes the PV insignificant
- No matter which board you run the design on, the output will favor RO_128 in this case


Board1	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	25	15
	Delay3	5	1
	PV	4	3
	Total Delay	36	24
Board2	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	25	15
	Delay3	5	1
	PV	1	4
	Total Delay	33	25

The table on the right side is used to be an example, to explain why symmetric designs are so important, and the data were made up rather than extracted. The process variation will be hard to measure since that is the feature of the physical devices.

Solution

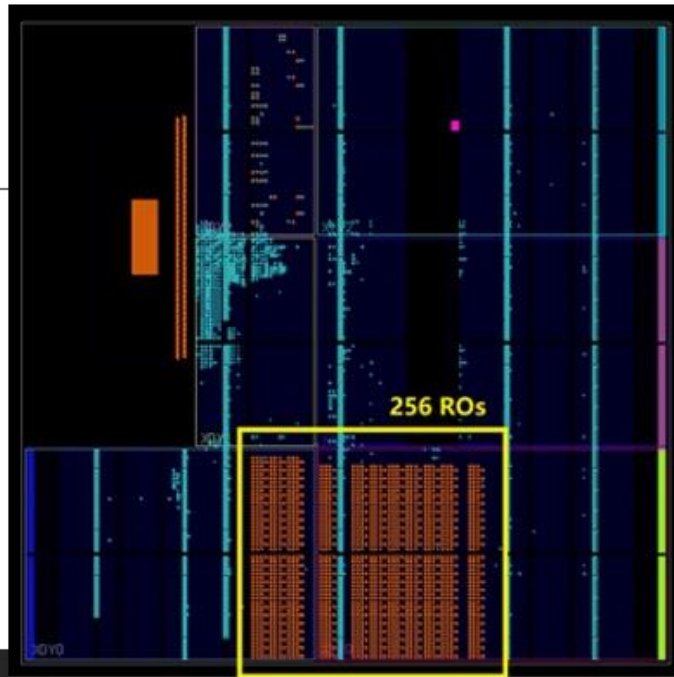
- Use Tcl commands to manually place and route the Ros
 - Use `unplace_cell` to clear an area
 - Use `place_cell` to place the LUTs of each RO identically

[Link to the command file](#)



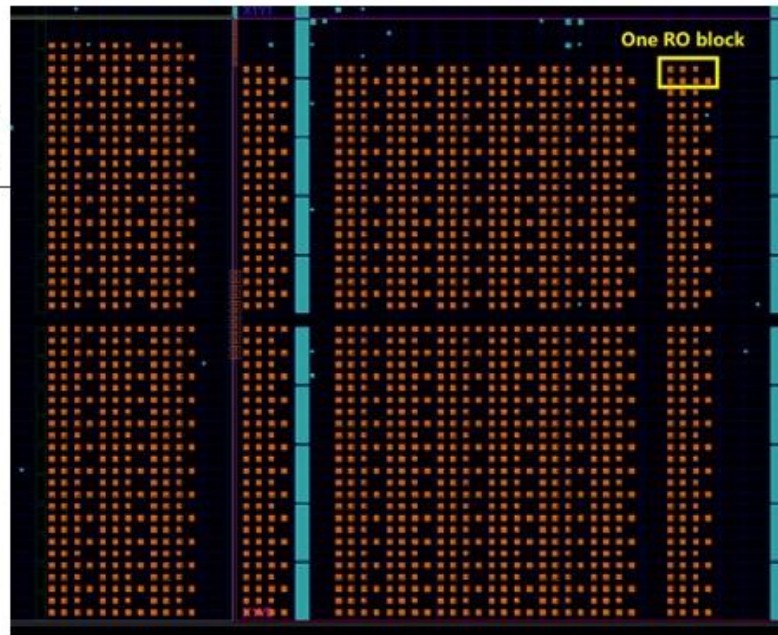
*Note: I will attach how I did this in another file
([place_and_route_using_tcl_commands.pdf](#))*

Solution (Placement)



Solution (Placement)

Need to skip some columns
because we want to place
blocks at locations where
the available routing
resources are identical



This shows the placement of the LUTs of the ROs.

Solution (Routing)

- A, B, and C are three adjacent tiles locate in different columns
- Each RO needs two tiles
- Choose AB vs. BC results in different available internal routing resources
- Since ROs in other columns have available routing resources like BC does, we have to skip column A and use BC



We had to skip some columns to find identical routing resources.

New Result (Intra-HD)

After making these changes, the intra-HDs are still good

```
[shidayang@ece-b312-recon proj_sw]$ ./top.o 0 1
Doing Intra-HD Test:

Start running Board
.....
Finish running Board

Board: 192.168.1.104:
Test 0: 0.0391667
Test 1: 0.025
Test 2: 0.005
Test 3: 0.02125
Test 4: 0.016875
Test 5: 0.0216667
[shidayang@ece-b312-recon proj_sw]$
```

After changing our design, the inter-HD is still good.

New Result (Inter-HD)

```
shidayang@ece-b312-recon proj_sw]$ ./top.o 845 0
Doing Inter-HD Test:
Start running Board1
.....
Finish running Board1
Start running Board2
.....
Finish running Board2
Board1: 192.168.1.103
Board2: 192.168.1.108
Test 0: 0.03125
Test 1: 0.185
Test 2: 0.0911458
Test 3: 0.069375
Test 4: 0.1875
Test 5: 0.165208
shidayang@ece-b312-recon proj_sw]$
```

Before change

```
shidayang@ece-b312-recon proj_sw]$ ./top.o 0 0
Doing Inter-HD Test:
Start running Board1
.....
Finish running Board1
Start running Board2
.....
Finish running Board2
Board1: 192.168.1.107
Board2: 192.168.1.104
Test 0: 0.319896
Test 1: 0.239688
Test 2: 0.343021
Test 3: 0.399583
Test 4: 0.383229
Test 5: 0.281146
shidayang@ece-b312-recon proj_sw]$
```

After change

We also see a significant improvement in the inter-HD. However, it is still not perfect.

Possible Improvements and Limitations

- PV has a greater change to dominate the output
- But Delay1 and Delay3 can sometimes dominate, making inter-HD less than 0.5
- Possible solution:
 - Change placement so that the routing between the **controller** and each **RO** are the same (decrease difference between Delay1's)
 - Change placement so that the routing between the **RO** and the **counter** are the same (decrease difference between Delay3's)
 - Choose FPGAs with greater process variation (increase difference between PV)

Board1	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	15	15
	Delay3	5	1
	PV	4	3
	Total Delay	36	24
Board2	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	15	15
	Delay3	5	1
	PV	1	4
	Total Delay	23	25

The reason is that our change only solved the problem for Delay2, but Delay1 and Delay3 are still different. Even though PV has a greater change to dominate now, Delay1 and Delay3 may also dominate sometimes. The solution is to change the placement of our design to make all the Delay1's identical and all the Delay3's identical. Or, we can also find FPGAs with greater PV.

Possible Improvements and Limitations

- Limitations (why we did not make those improvement):
 - To change placement, we need to consider the routing resources as well. If the ROs are not placed column by column, it is hard to keep track of the internal routing of 256 ROs manually
 - We are limited to the kind of boards we have

Board1	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	15	15
	Delay3	5	1
	PV	4	3
	Total Delay	36	24
Board2	Delay Type	RO_0 Path	RO_128 Path
	Delay1	2	5
	Delay2	15	15
	Delay3	5	1
	PV	1	4
	Total Delay	23	25

The solutions are not feasible at this point given the reasons in the slide.

Conclusion

- We build a 16-bit Ring Oscillator Physical Unclonable Function with great complexity in both hardware structure and software algorithm, which can be used in hardware security protection in the real world.
- Currently it is highly reliable and relatively special for each devices, which can be shown from our test output and data analysis. We are satisfied about our progress and result.

Our design is highly reliable, but it does not show the uniqueness of each FPGA perfectly. However, we identified the issue and partially solved it after doing some research. Even though our design is not perfect, it is still useable in real world. Since the difference between the inter-HD and the intra-HD is significant enough, we can easily set a threshold (like 0.05) to determine if the FPGA that responses to the challenges is the FPGA we expect.

Reference

- [1] Tehranipoor and C. Wang (Eds.), Introduction to Hardware Security and Trust, *Springer*, 2011
- [2] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. 2002. Physical one-way functions. *Science* 297(2002), 2026–2030.
- [3] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. 2002. Silicon physical random functions. In *CCS*. 148–160.
- [4] M. Majzoobi, M. Rostami, F. Koushanfar, D.S. Wallach, and S. Devadas. 2012. Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. In *IEEE Symposium on Security and Privacy Workshops (SPW)*. 33 – 44.

