Cole Maxwell (25017014): ECE 658
Lab 5: 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.

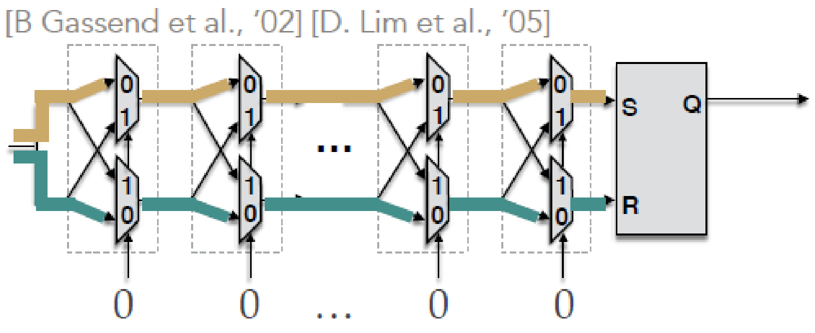


Figure 1: Gate-Level Schematic of an Arbiter PUF

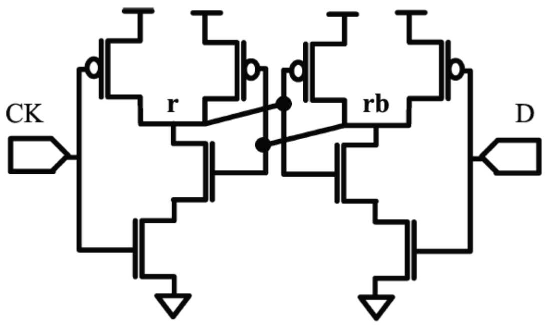


Figure 2: Transistor-Level Schematic of the Arbiter Circuit

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 the select bit in each stage can be set to 0 or 1, providing 2^m 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 could provide $2^8 = 256$ possible challenge-response pairs, and I will evaluate 4 of these challenges.

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 to compare to the layout.

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. I will check the expected waveforms and supply them in the results section.

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. Then I will combine 8-stages of the bitslice circuit with the arbiter. 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 and Simulation Delay Results

In this step, I will extract a netlist from the layout with parasitics and repeat the simulation from step 2. Finally, I will use Cscope waveforms to determine which signal arrives to the arbiter first based on the output value Q.

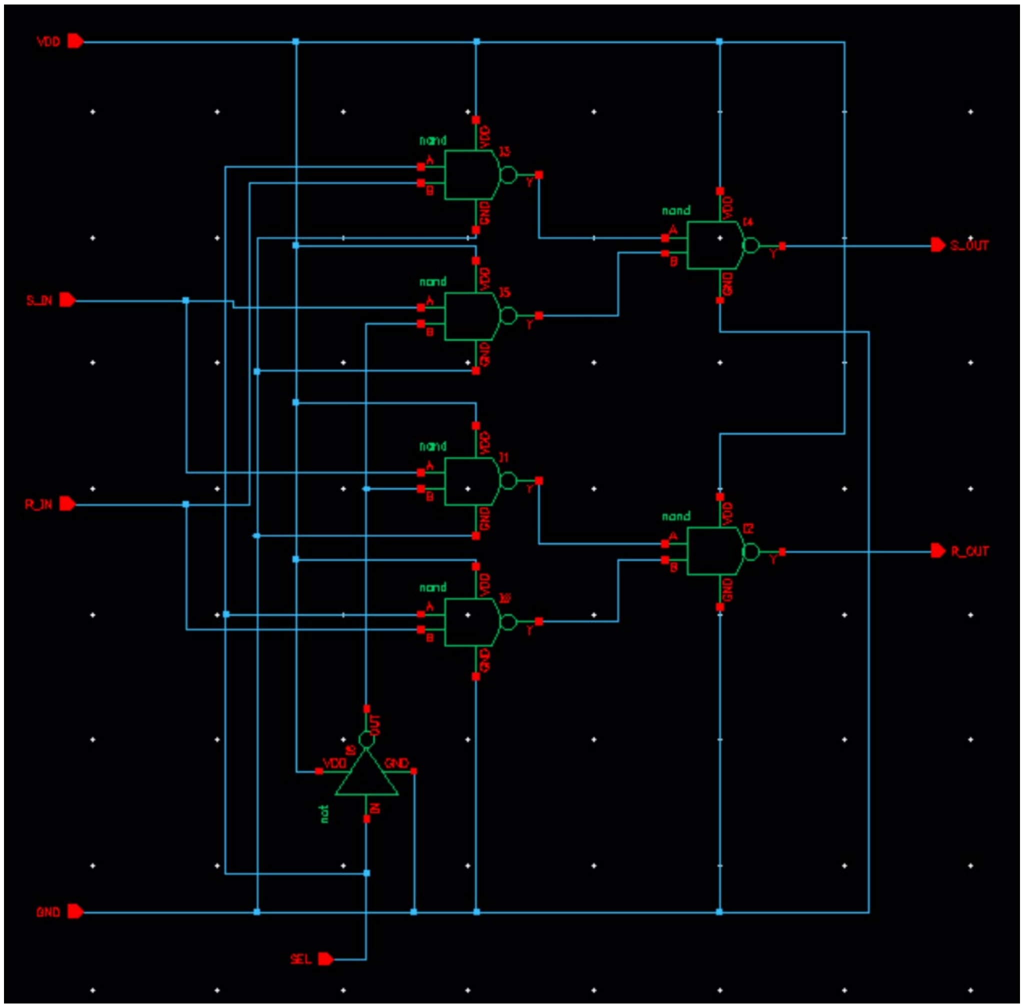
Step 5: Conclusion

Finally, I will evaluate four challenge-response pairs and reflect on the project.

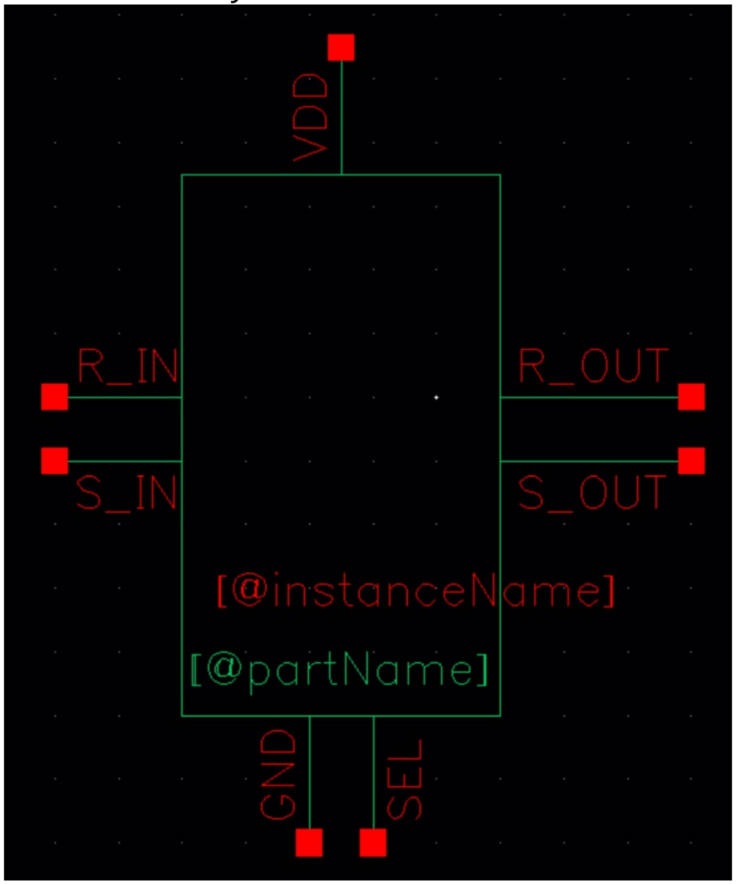
Report Section 3: Results and Analysis

Step 1: Create Schematics and Symbols

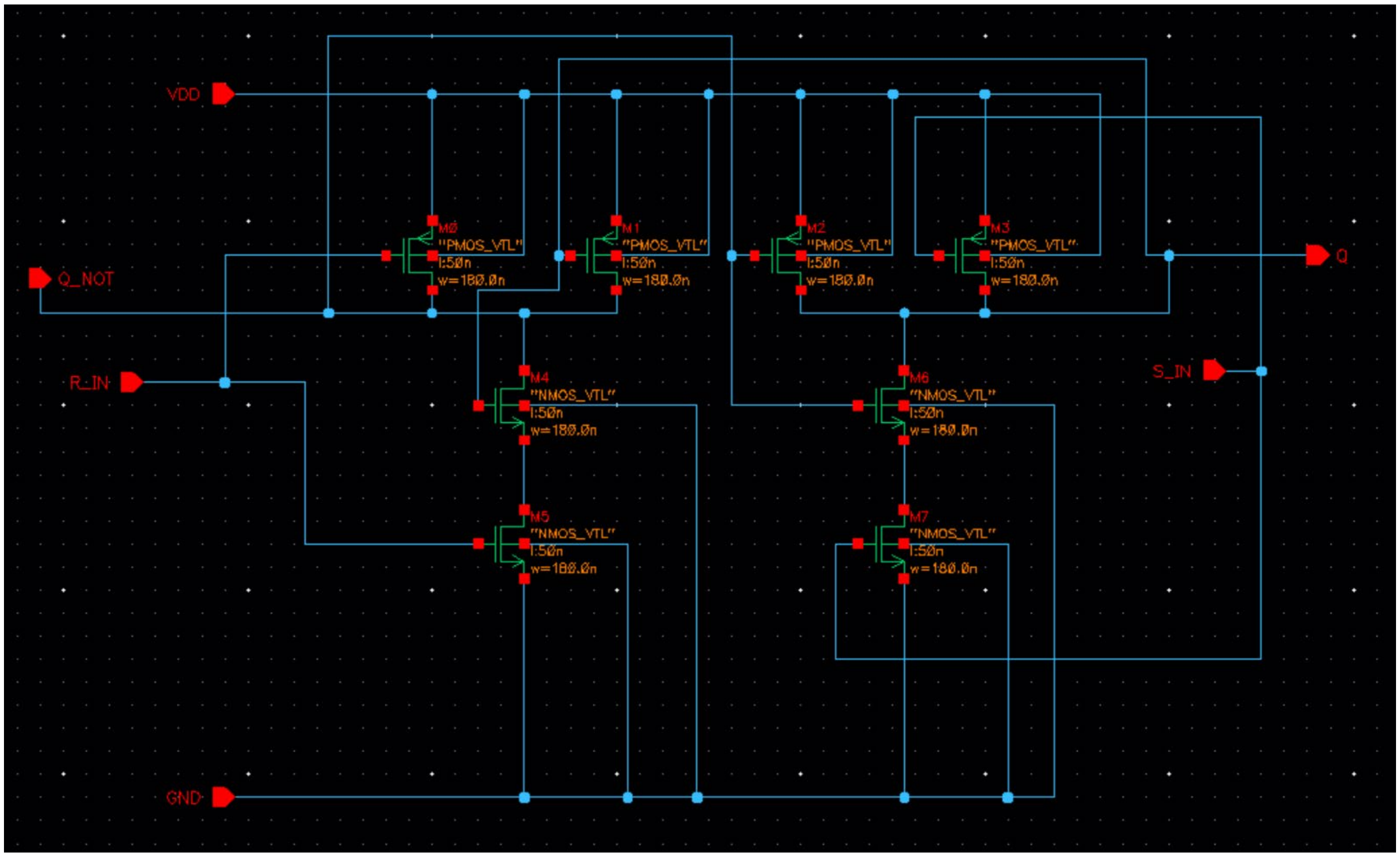
MUX Bitslice Schematic:



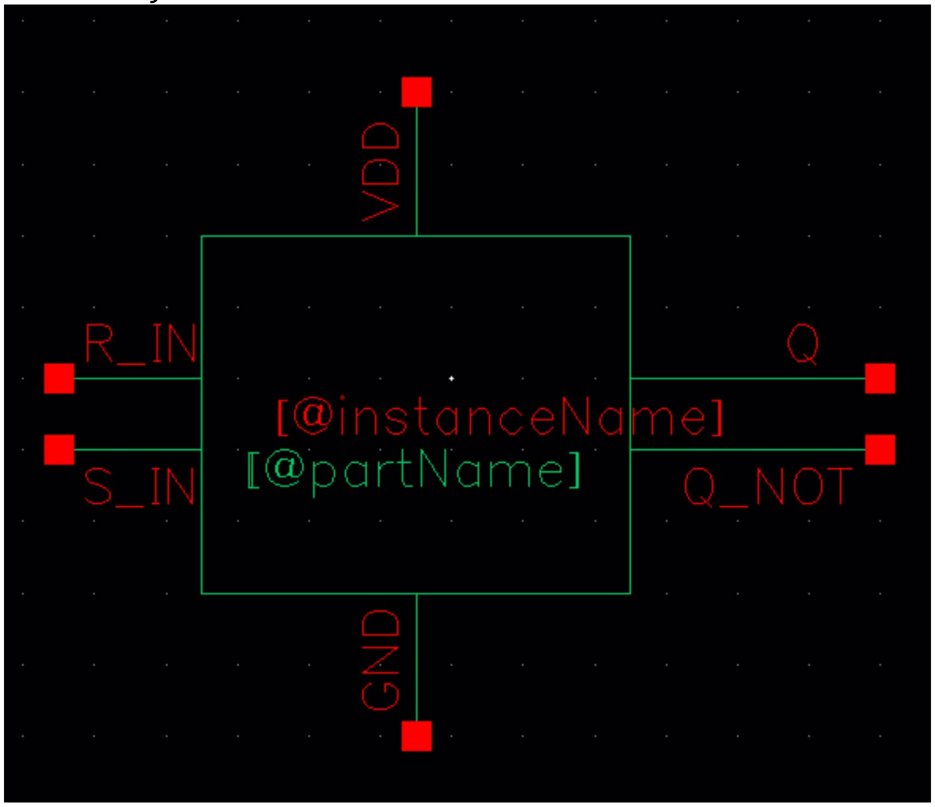
MUX Bitslice Symbol:



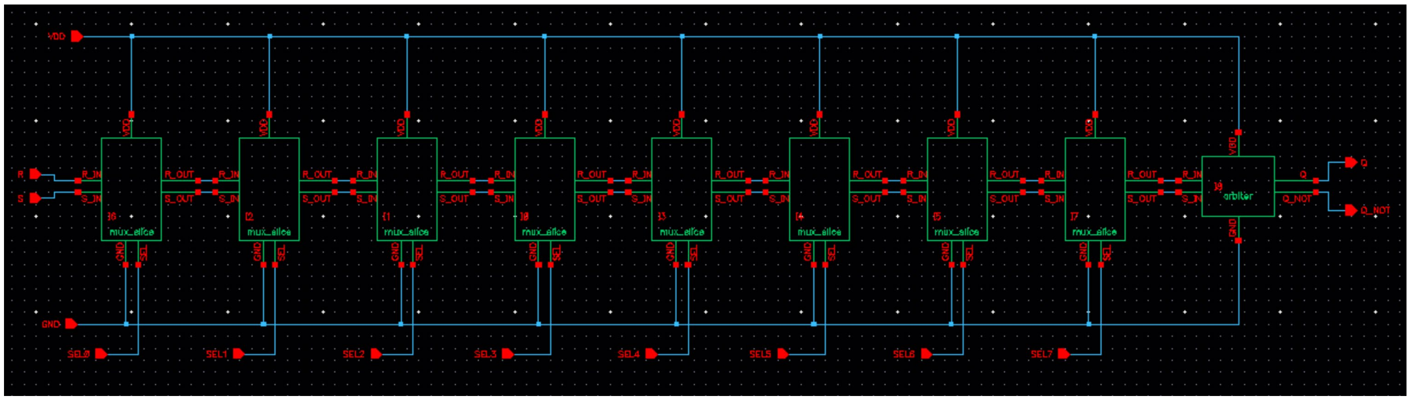
Arbiter Schematic:


VDD
Q_NOT
R_IN
GND
S_IN
Q
M0 "PMOS_VTL" l:50n w=180.0n
M1 "PMOS_VTL" l:50n w=180.0n
M2 "PMOS_VTL" l:50n w=180.0n
M3 "PMOS_VTL" l:50n w=180.0n
M4 "NMOS_VTL" l:50n w=180.0n
M5 "NMOS_VTL" l:50n w=180.0n
M6 "NMOS_VTL" l:50n w=180.0n
M7 "NMOS_VTL" l:50n w=180.0n

Arbiter Symbol:

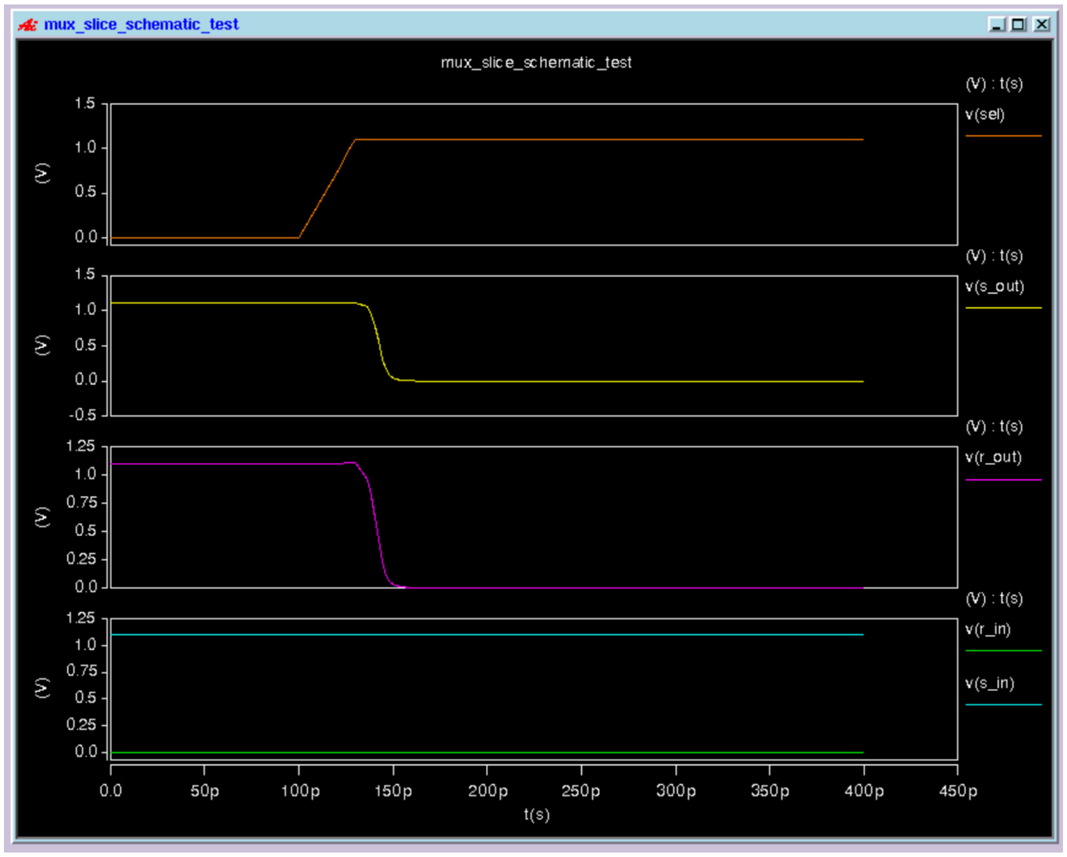


8-stage Arbiter PUF Schematic:

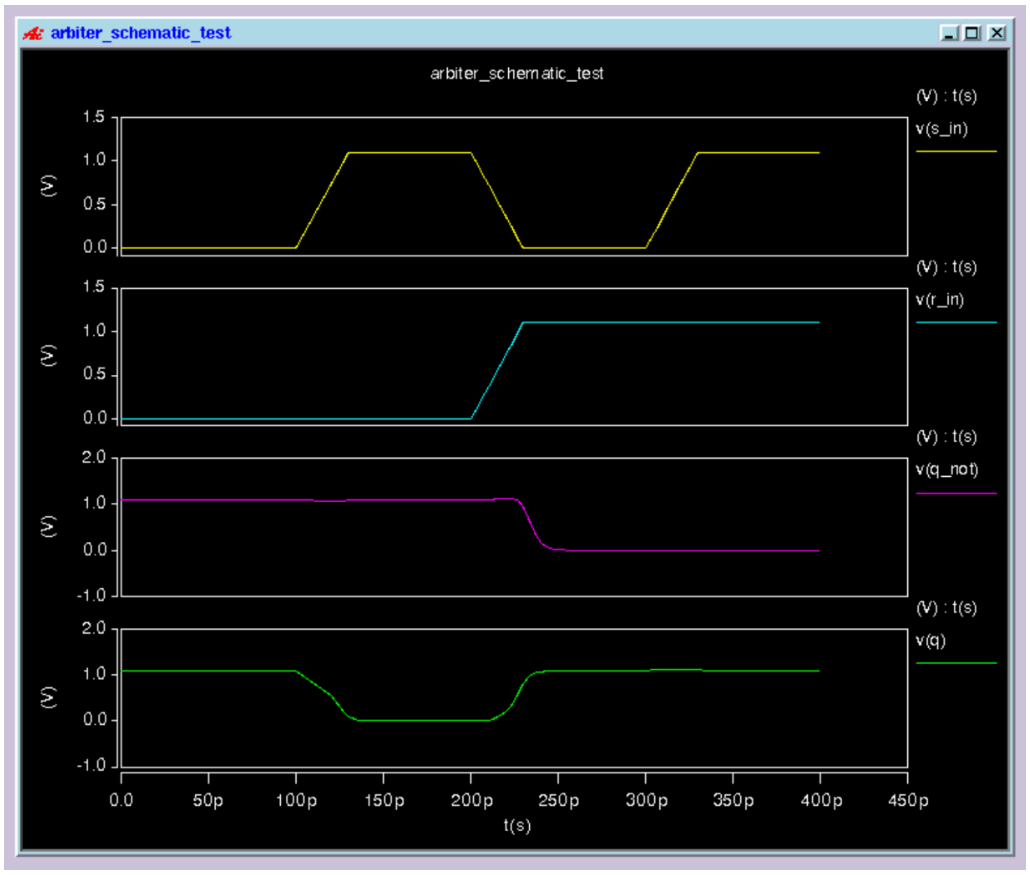


Step 2: Check Functionality of Schematics

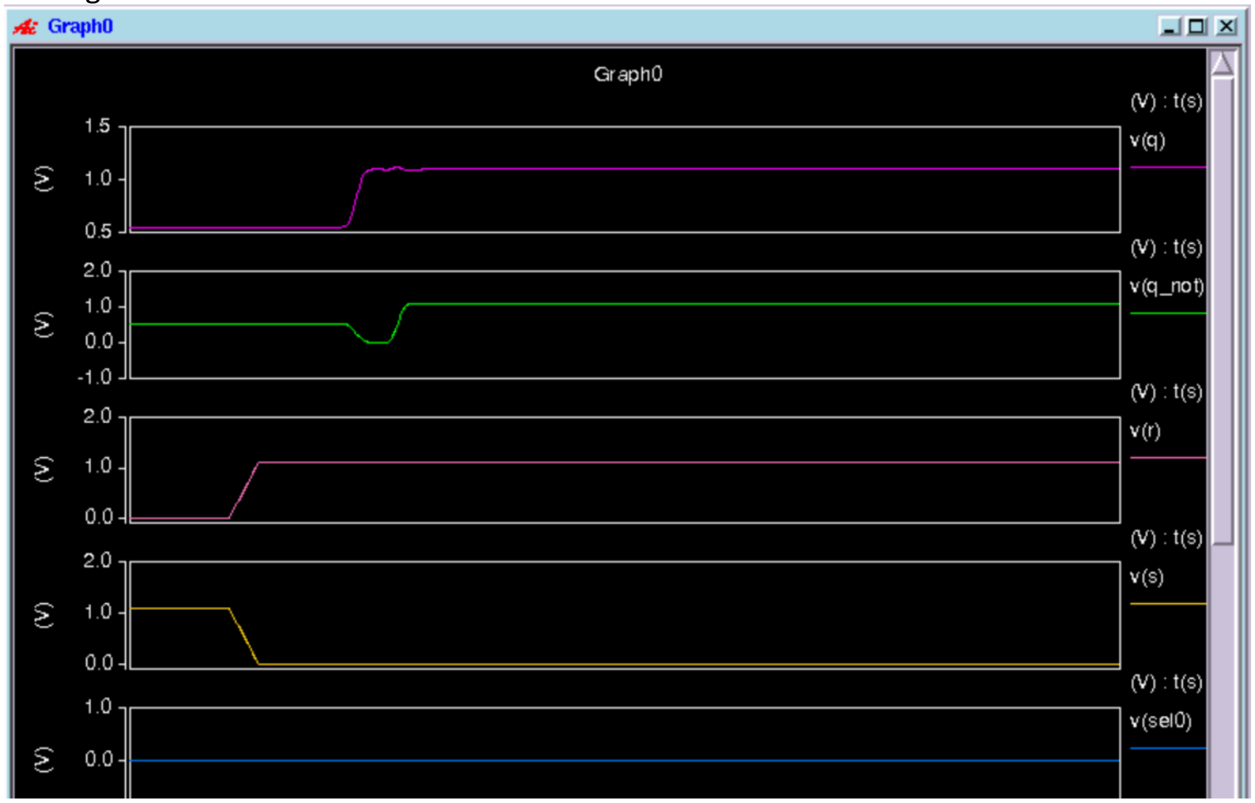
MUX Bitslice Schematic Waveforms:



Arbiter Schematic Waveforms:

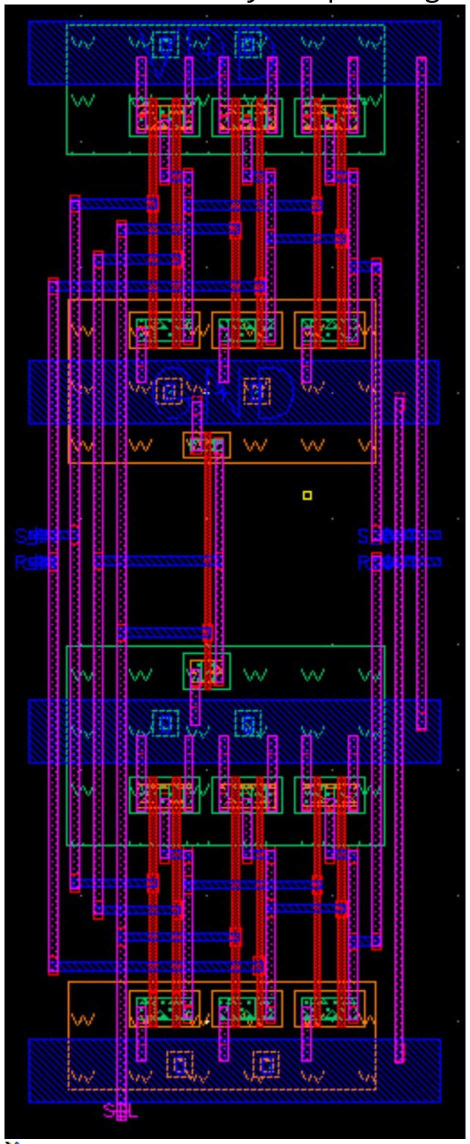


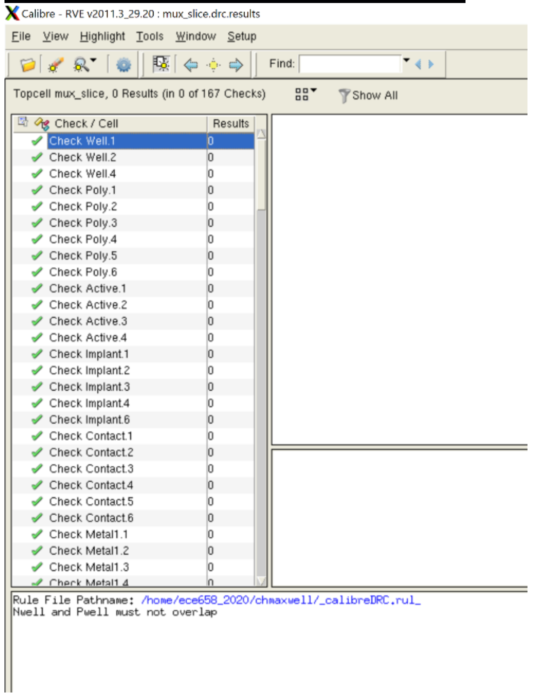
8-stage Arbiter PUF Schematic Waveforms:



Step 3: Layouts

MUX Bitslice Layout passing DRC & LVS:



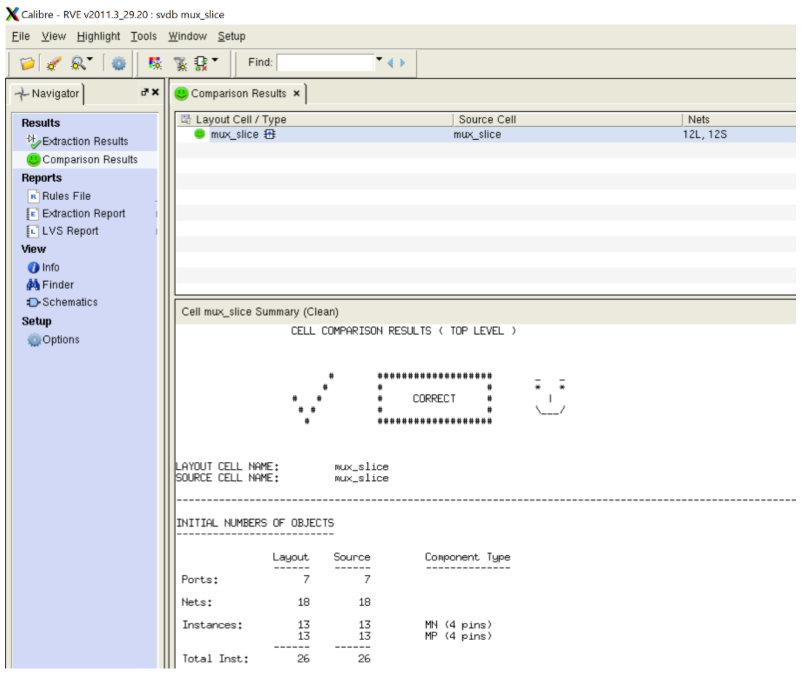


Calibre - RVE v2011.3_29.20 : mux_slice.drc.results
File View Highlight Tools Window Setup
Find:
Show All

Topcell mux_slice, 0 Results (in 0 of 167 Checks)

Check / Cell	Results
✓ Check Well.1	0
✓ Check Well.2	0
✓ Check Well.4	0
✓ Check Poly.1	0
✓ Check Poly.2	0
✓ Check Poly.3	0
✓ Check Poly.4	0
✓ Check Poly.5	0
✓ Check Poly.6	0
✓ Check Active.1	0
✓ Check Active.2	0
✓ Check Active.3	0
✓ Check Active.4	0
✓ Check Implant.1	0
✓ Check Implant.2	0
✓ Check Implant.3	0
✓ Check Implant.4	0
✓ Check Implant.6	0
✓ Check Contact.1	0
✓ Check Contact.2	0
✓ Check Contact.3	0
✓ Check Contact.4	0
✓ Check Contact.5	0
✓ Check Contact.6	0
✓ Check Metal1.1	0
✓ Check Metal1.2	0
✓ Check Metal1.3	0
✓ Check Metal1.4	0

```
Rule File Pathname: /home/ece658_2020/chmaxwell/_calibreDRC.rul_
Nwell and Pwell must not overlap
```

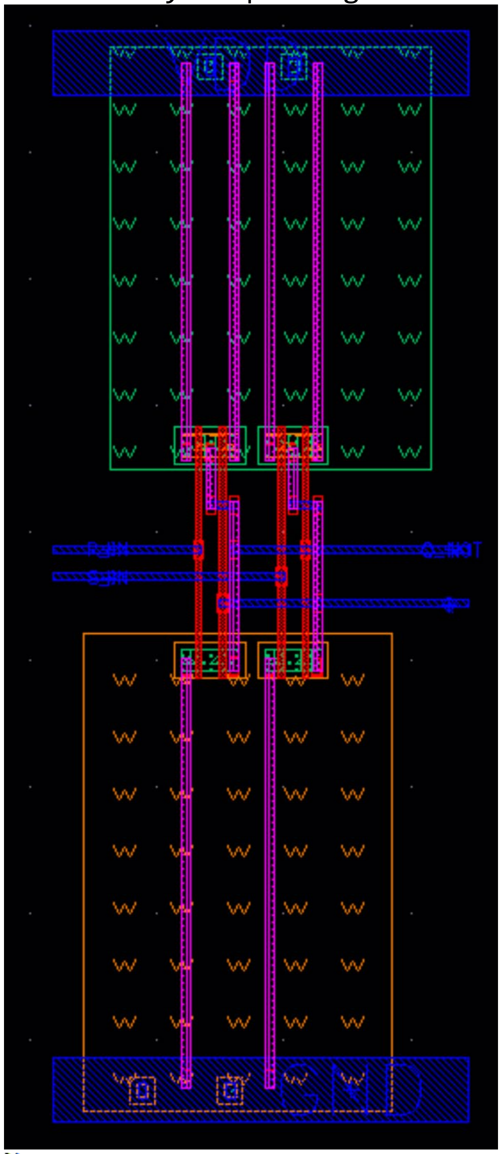


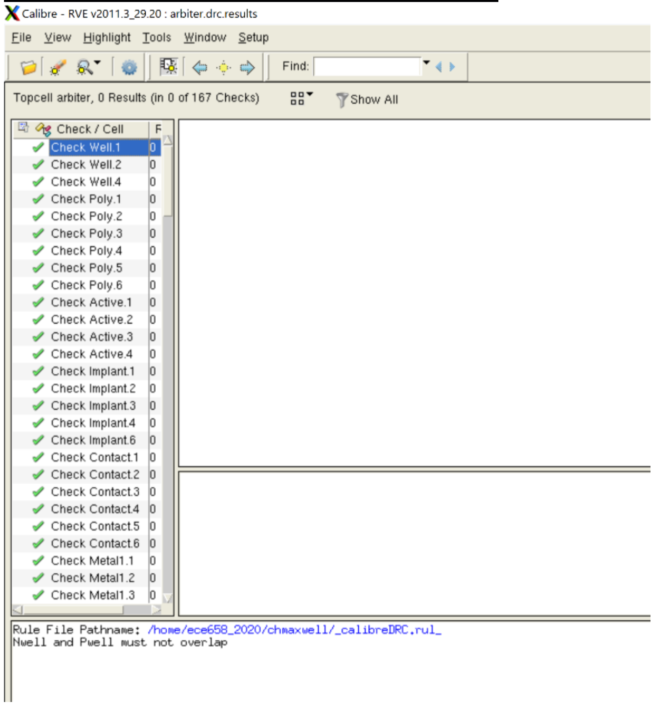
Layout Cell / Type	Source Cell	Nets
● mux_slice	mux_slice	12L, 12S

Cell mux_slice Summary (Clean)

```
                    CELL COMPARISON RESULTS ( TOP LEVEL )


                         #         ###################       _   _
                        #          #                 #        *   *
                    #   #          #     CORRECT     #          |
                     # #           #                 #        \___/
                      #            ###################


LAYOUT CELL NAME:         mux_slice
SOURCE CELL NAME:         mux_slice

--------------------------------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:              7         7

 Nets:              18        18

 Instances:         13        13         MN (4 pins)
                    13        13         MP (4 pins)
                ------    ------
 Total Inst:        26        26
```

Arbiter Layout passing DRC & LVS:

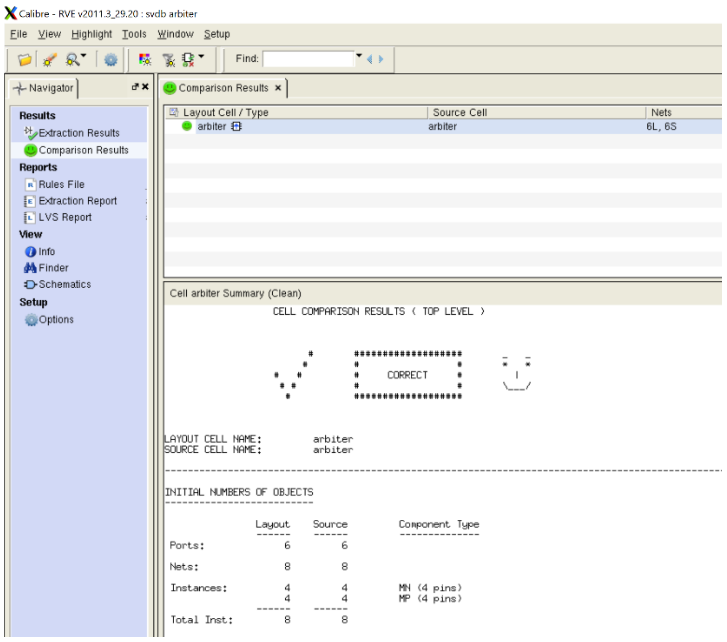



Calibre - RVE v2011.3_29.20 : arbiter.drc.results
File View Highlight Tools Window Setup
Find:
Show All

Topcell arbiter, 0 Results (in 0 of 167 Checks)

Check / Cell	F
✓ Check Well.1	0
✓ Check Well.2	0
✓ Check Well.4	0
✓ Check Poly.1	0
✓ Check Poly.2	0
✓ Check Poly.3	0
✓ Check Poly.4	0
✓ Check Poly.5	0
✓ Check Poly.6	0
✓ Check Active.1	0
✓ Check Active.2	0
✓ Check Active.3	0
✓ Check Active.4	0
✓ Check Implant.1	0
✓ Check Implant.2	0
✓ Check Implant.3	0
✓ Check Implant.4	0
✓ Check Implant.6	0
✓ Check Contact.1	0
✓ Check Contact.2	0
✓ Check Contact.3	0
✓ Check Contact.4	0
✓ Check Contact.5	0
✓ Check Contact.6	0
✓ Check Metal1.1	0
✓ Check Metal1.2	0
✓ Check Metal1.3	0

```
Rule File Pathname: /home/ece658_2020/chmaxwell/_calibreDRC.rul_
Nwell and Pwell must not overlap
```

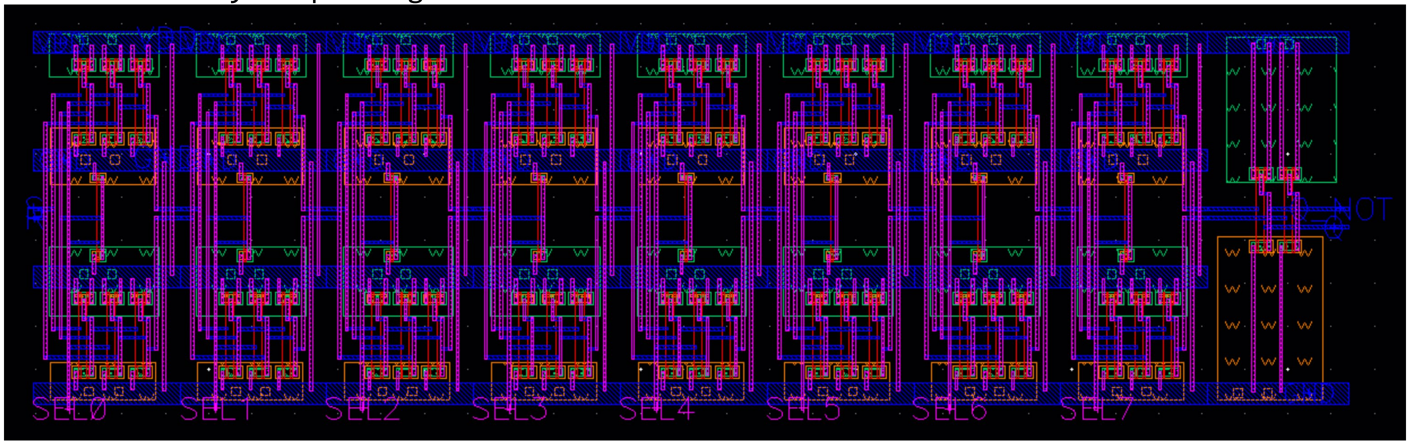


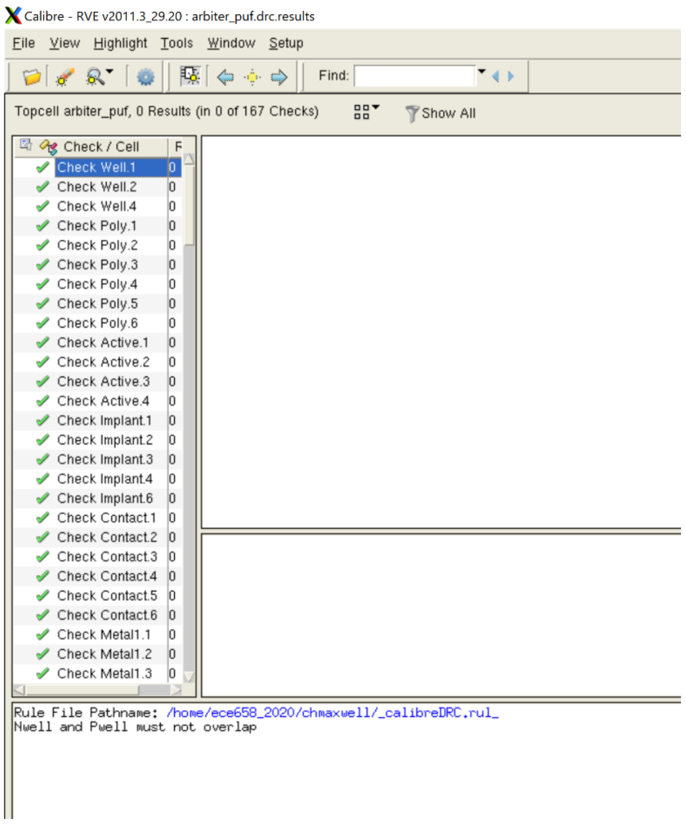
Layout Cell / Type	Source Cell	Nets
● arbiter	arbiter	6L, 6S

Cell arbiter Summary (Clean)

```
                    CELL  COMPARISON RESULTS ( TOP LEVEL )


                         #         ###################       _   _
                        #          #                 #        *   *
                   #   #           #     CORRECT     #          |
                    # #            #                 #        \___/
                     #             ###################


LAYOUT CELL NAME:         arbiter
SOURCE CELL NAME:         arbiter

--------------------------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                Layout    Source         Component Type
                ------    ------         --------------

 Ports:            6         6

 Nets:             8         8

 Instances:        4         4           MN (4 pins)
                   4         4           MP (4 pins)
                ------    ------
 Total Inst:       8         8
```

Arbiter PUF Layout passing DRC & LVS:



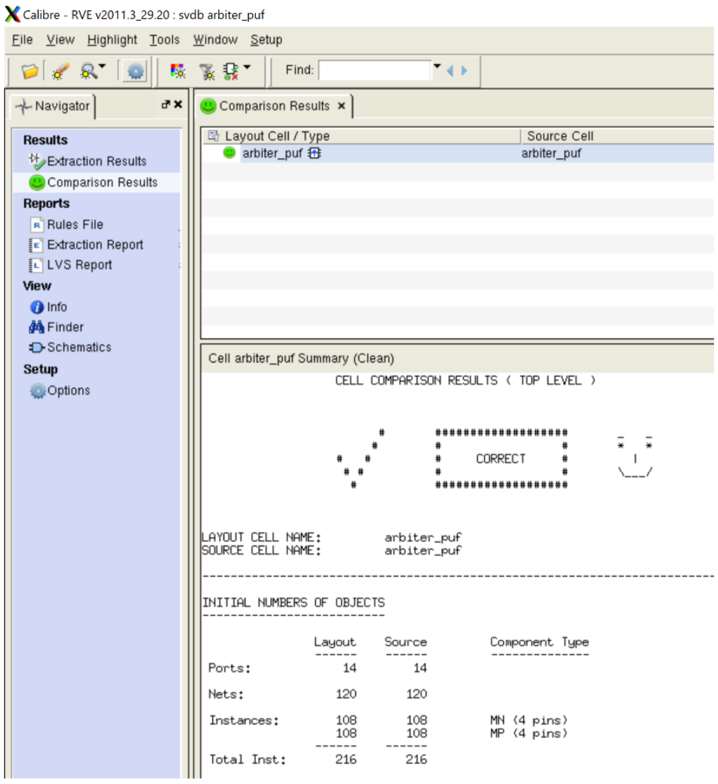


Calibre - RVE v2011.3_29.20 : arbiter_puf.drc.results
File View Highlight Tools Window Setup
Find:
Show All
Check / Cell

Topcell arbiter_puf, 0 Results (in 0 of 167 Checks)

Check / Cell	F
Check Well.1	0
Check Well.2	0
Check Well.4	0
Check Poly.1	0
Check Poly.2	0
Check Poly.3	0
Check Poly.4	0
Check Poly.5	0
Check Poly.6	0
Check Active.1	0
Check Active.2	0
Check Active.3	0
Check Active.4	0
Check Implant.1	0
Check Implant.2	0
Check Implant.3	0
Check Implant.4	0
Check Implant.6	0
Check Contact.1	0
Check Contact.2	0
Check Contact.3	0
Check Contact.4	0
Check Contact.5	0
Check Contact.6	0
Check Metal1.1	0
Check Metal1.2	0
Check Metal1.3	0

```
Rule File Pathname: /home/ece658_2020/chmaxwell/_calibreDRC.rul_
Nwell and Pwell must not overlap
```



Layout Cell / Type	Source Cell
arbiter_puf	arbiter_puf

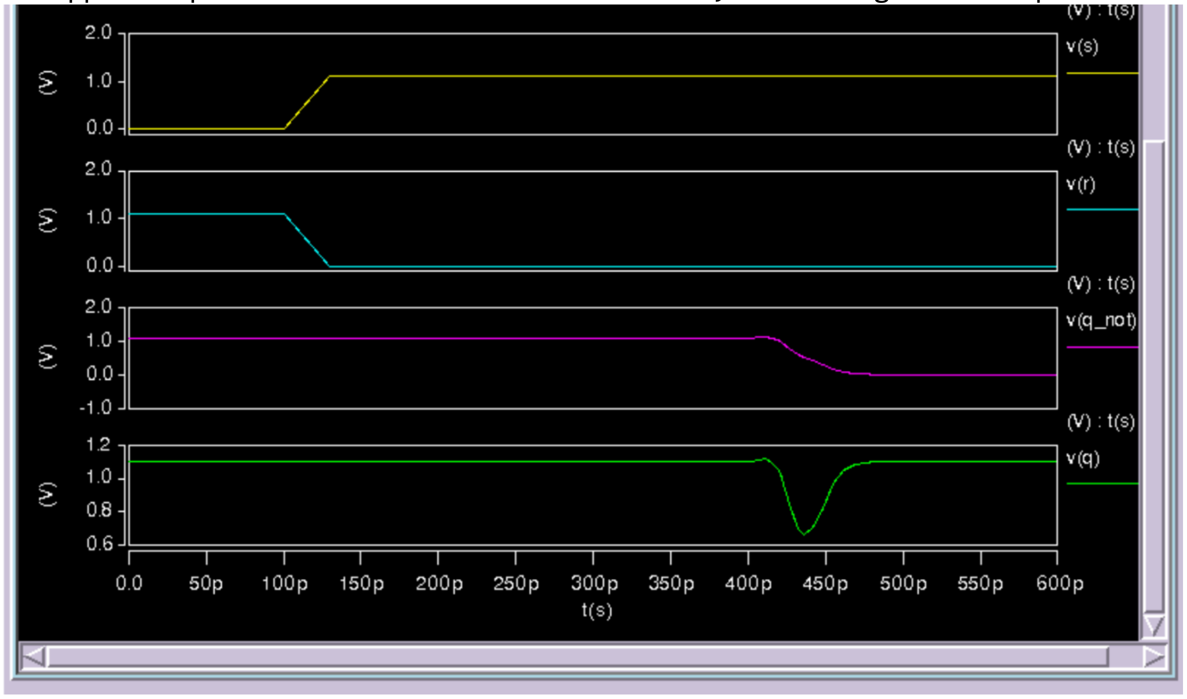
Cell arbiter_puf Summary (Clean)

```
                    CELL COMPARISON RESULTS ( TOP LEVEL )


                 #        ###################       _   _
                #         #                 #        *   *
           #   #          #     CORRECT     #          |
            # #           #                 #        \___/
             #            ###################


LAYOUT CELL NAME:        arbiter_puf
SOURCE CELL NAME:        arbiter_puf

--------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:           14        14

 Nets:           120       120

 Instances:      108       108           MN (4 pins)
                 108       108           MP (4 pins)
                ------    ------
 Total Inst:     216       216
```

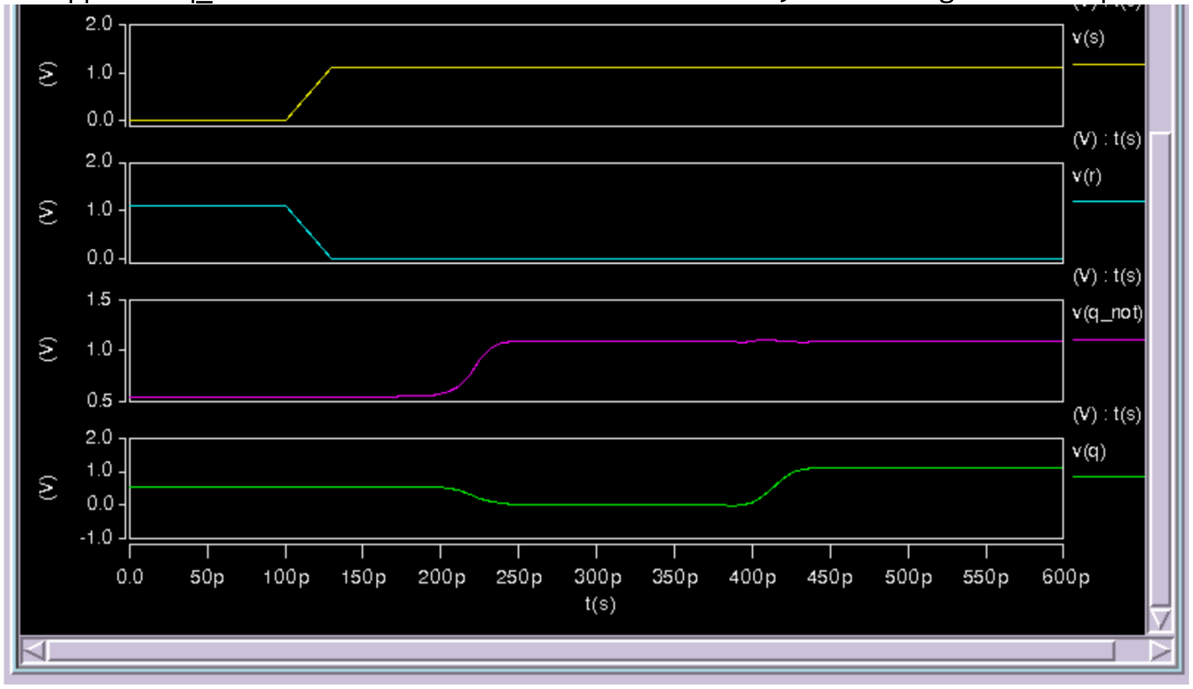
Step 4: Parasitics Extraction and Simulation Delay Results

Challenge-Response Pairs determined by waveforms:

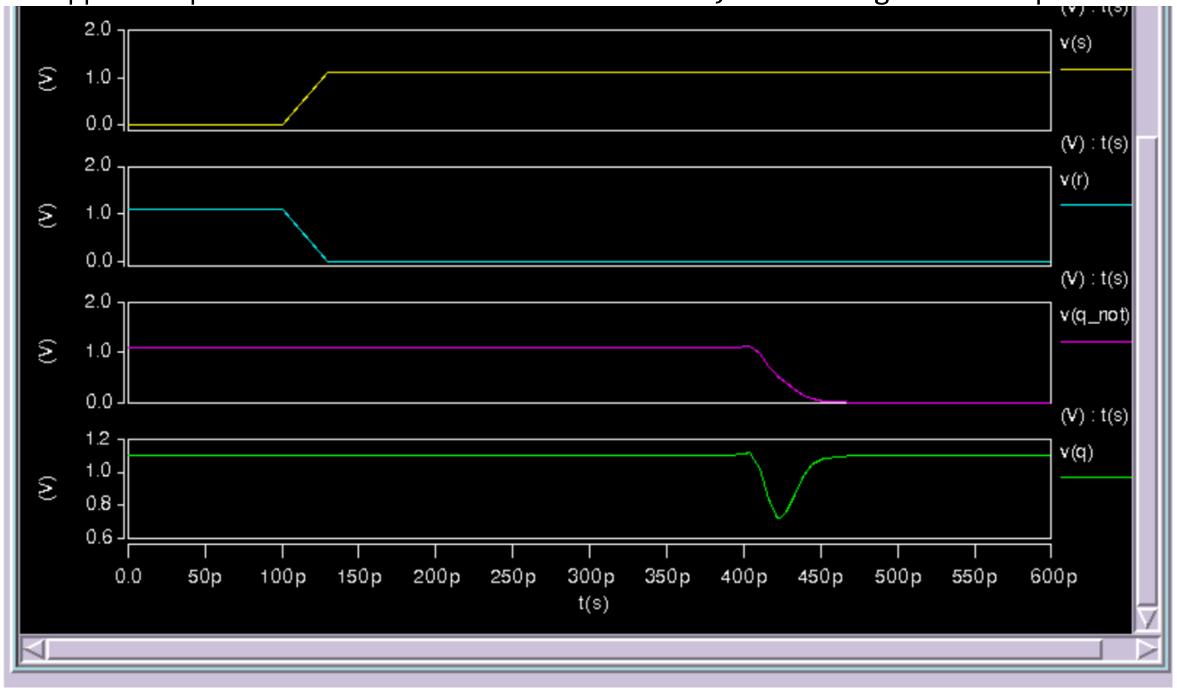
The first challenge was setting all select bits to zero: 00000000
It appears q makes a 1 to 0 transition first, resulting in a response of 1.



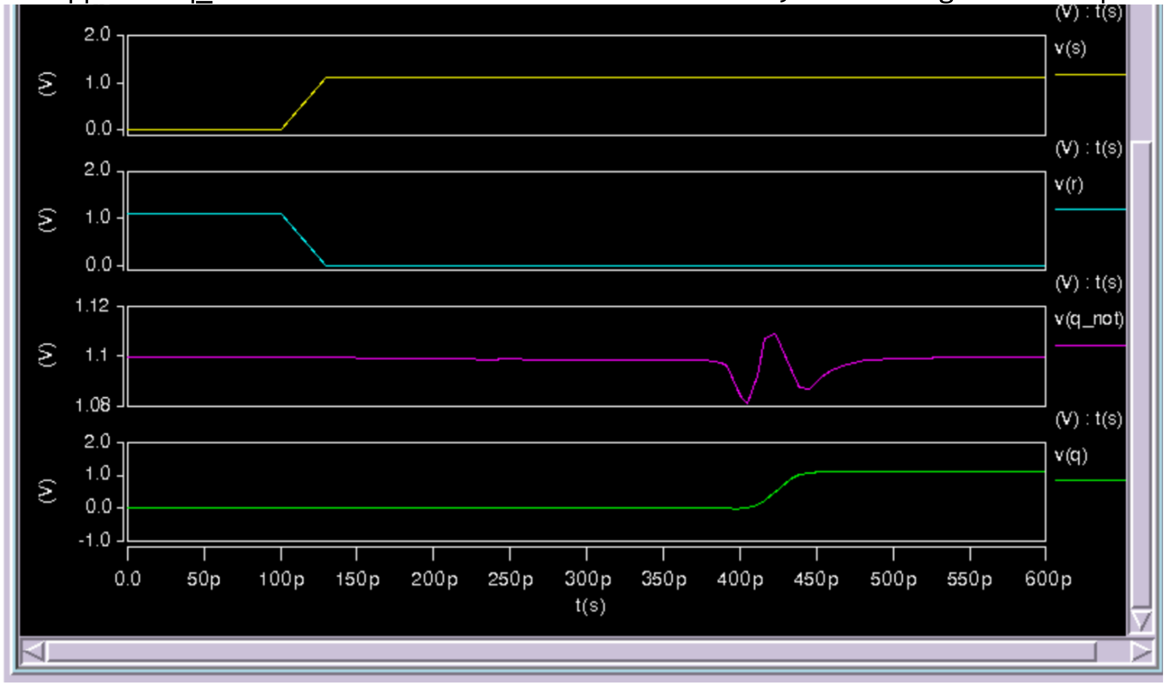
The second challenge was setting all select bits to one: 11111111
It appears q_not makes a 0 to 1 transition first, resulting in a response of 0.



The third challenge was alternating select bits: 01010101
It appears q makes a 1 to 0 transition first, resulting in a response of 1.



The fourth challenge was alternating select bits in the opposite manner: 10101010
It appears q_not makes a 1 to 0 transition first, resulting in a response of 0.



Step 5: Conclusion

After generating a netlist with parasitics from the final layout, two of the four challenges generated a 1 response, and the other two generated a 0 response. The delays between paths seen by the arbiter are quite small, and sometimes hard to distinguish. These results illustrate that the paths through the 8-stage PUF are close in length, and the delays seem to be determined by the parasitics and the specific layout I implemented.

To expand this project, I would work on minimizing the difference in path delays through the bitslice so that the characterization of the circuit would depend solely on process variations during fabrication. Then, I would use a 128-stage path and generate challenge-response pairs for all 2^{128} possible combinations to fully characterize the circuit.

The most difficult part of this process was designing the bitslice with paths of equal length, since the SELECT bit for each stage must be accessible from either the bottom or the top. This limitation was due to the abutment design choice, which forced me to use interconnect of different lengths to reach the bottom and top MUXs with the SELECT signals. I tried to minimize this issue by creating three layers in each stage, where the middle layer (containing only the inputs/outputs and one inverter) was equidistant from the bottom and top MUXs.