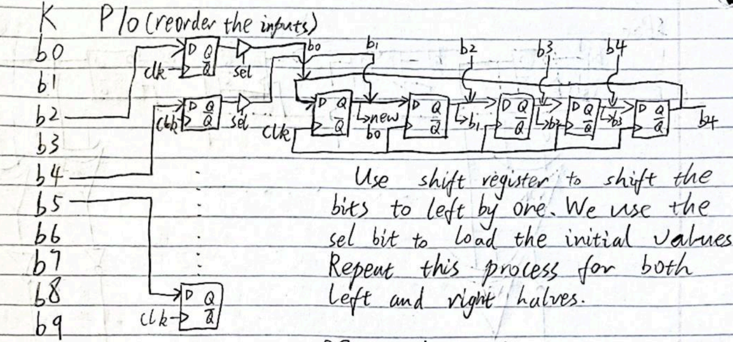
CSE 463/563M Digital Integrated Circuit Design & Architecture

SDES Hardware Encryption

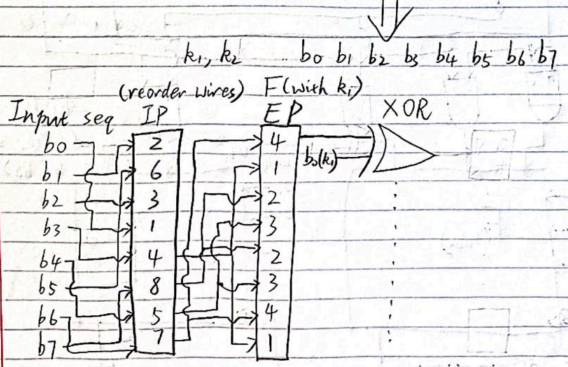
Connor Irwin (463)
Nick Song (563)
Ryan Kropp (563)

563 Final project Part 1

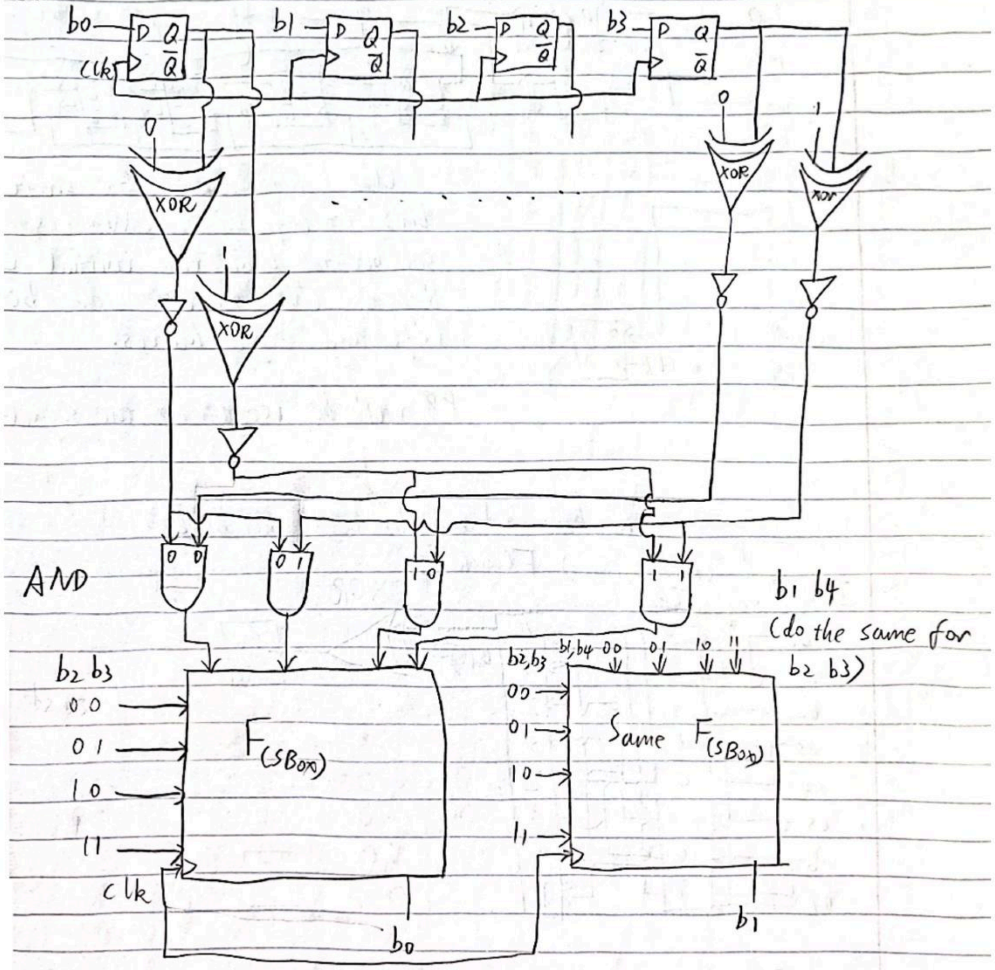


Use shift register to shift the bits to left by one. We use the sel bit to load the initial values. Repeat this process for both left and right halves.

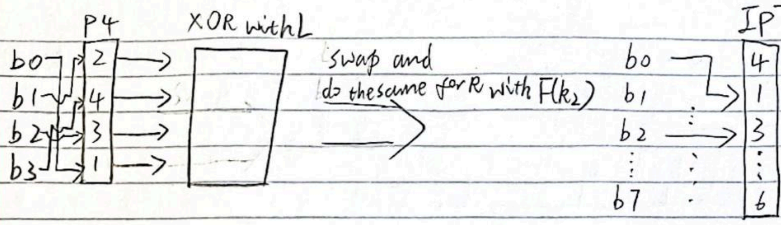
P8 will be reordering the signals again



Box



Repeat this process for b4, b5, b6, b7
and we can get a 4-bit output



P4
XOR with L
swap and
do the same for R with F(k2)
IP^-1
b0
b1
b2
b3
2
4
3
1
b0
b1
b2
b7
4
1
3
6

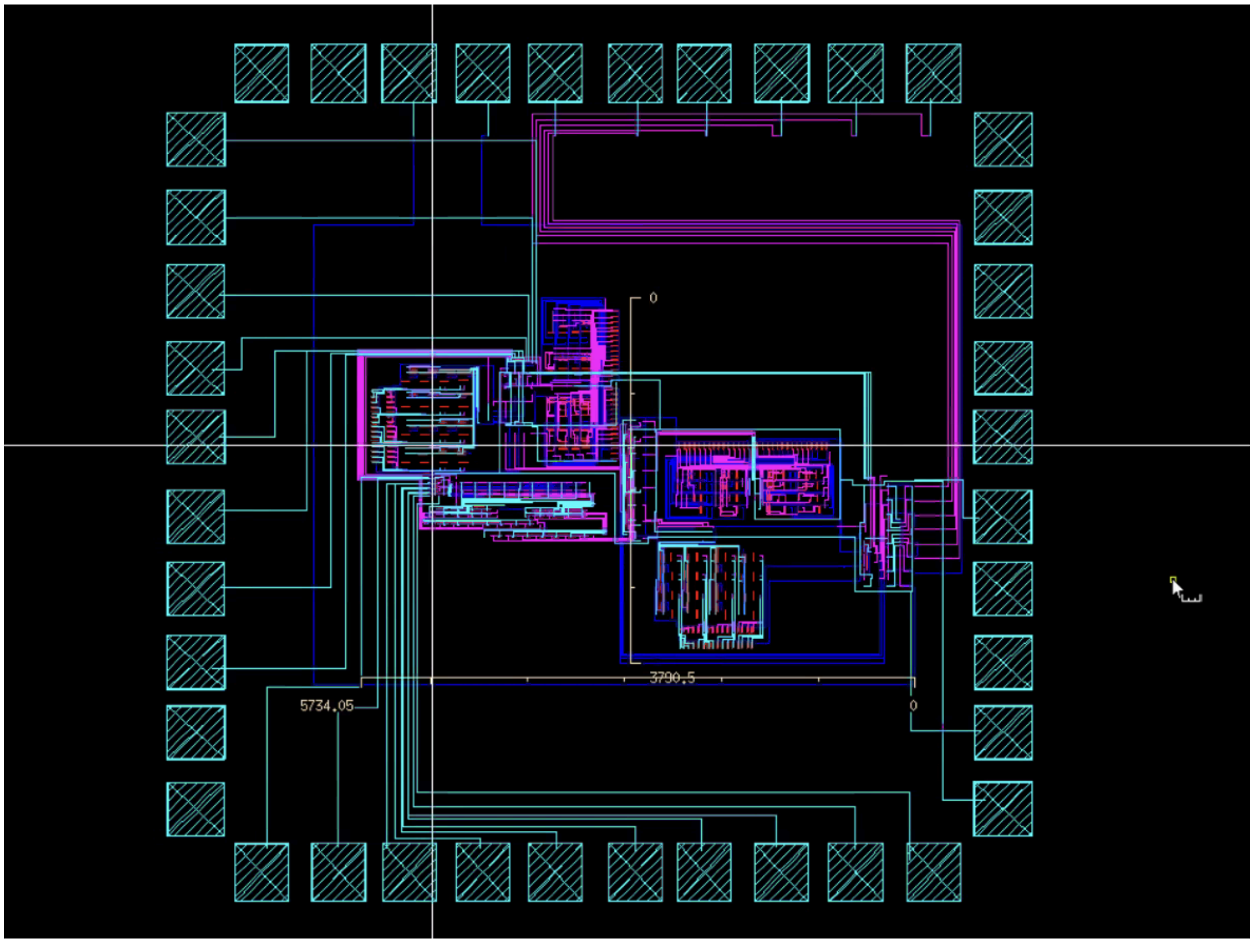


Figure 1. Whole Layout

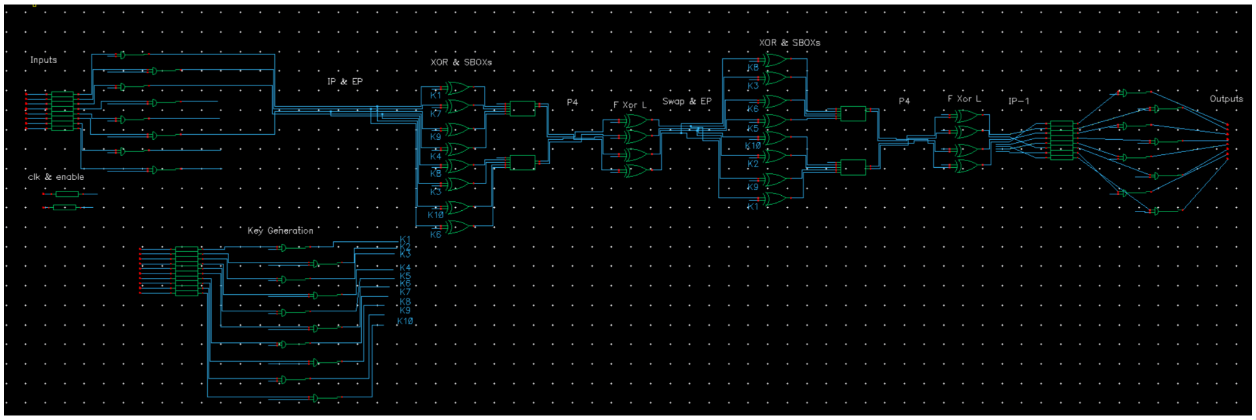


Figure 2. Whole Schematic

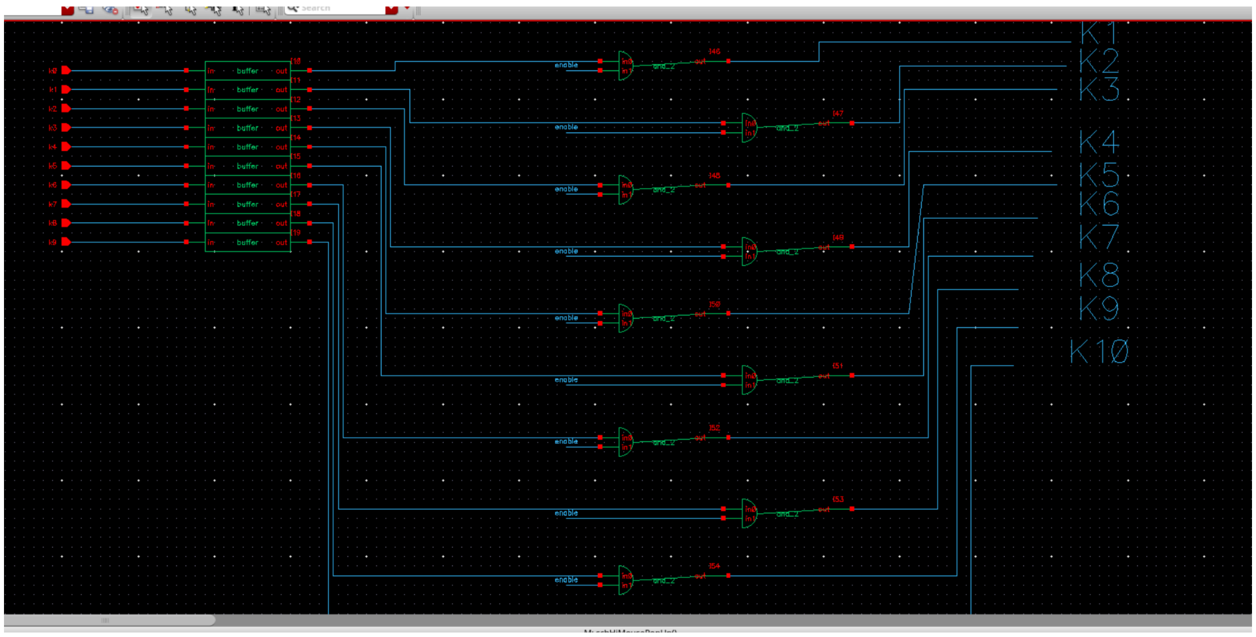


Figure 3. Key generation schematic

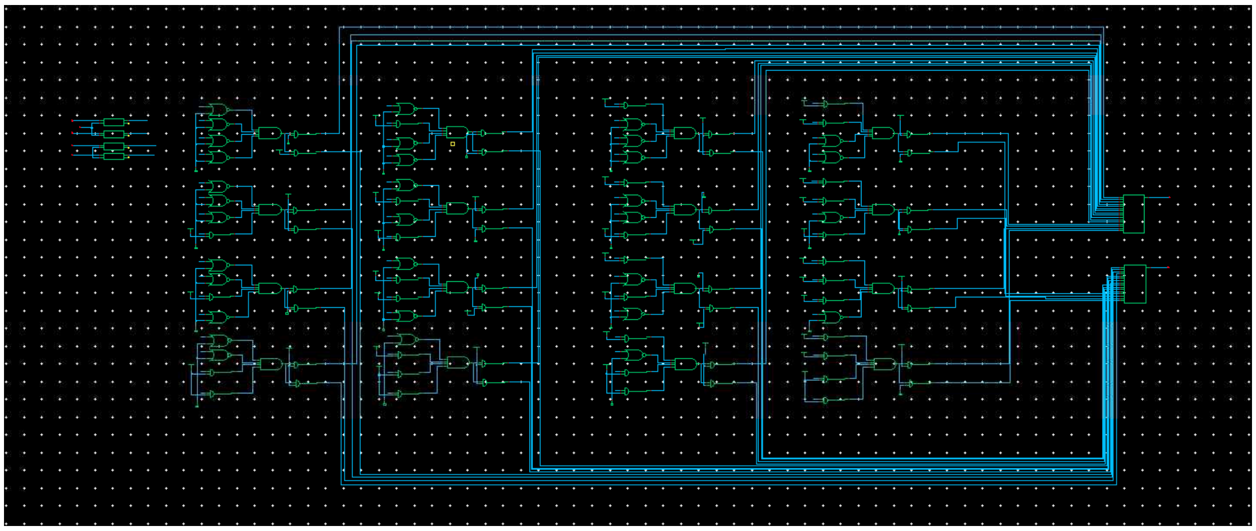


Figure 6. Sbox 1 schematic

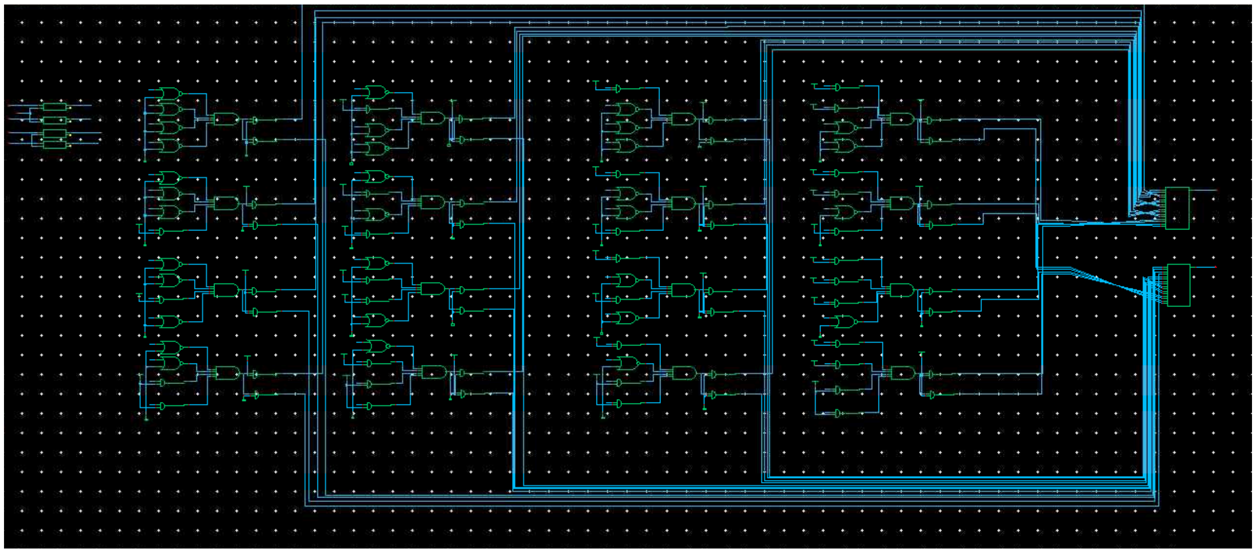


Figure 7. Sbox 2 schematic

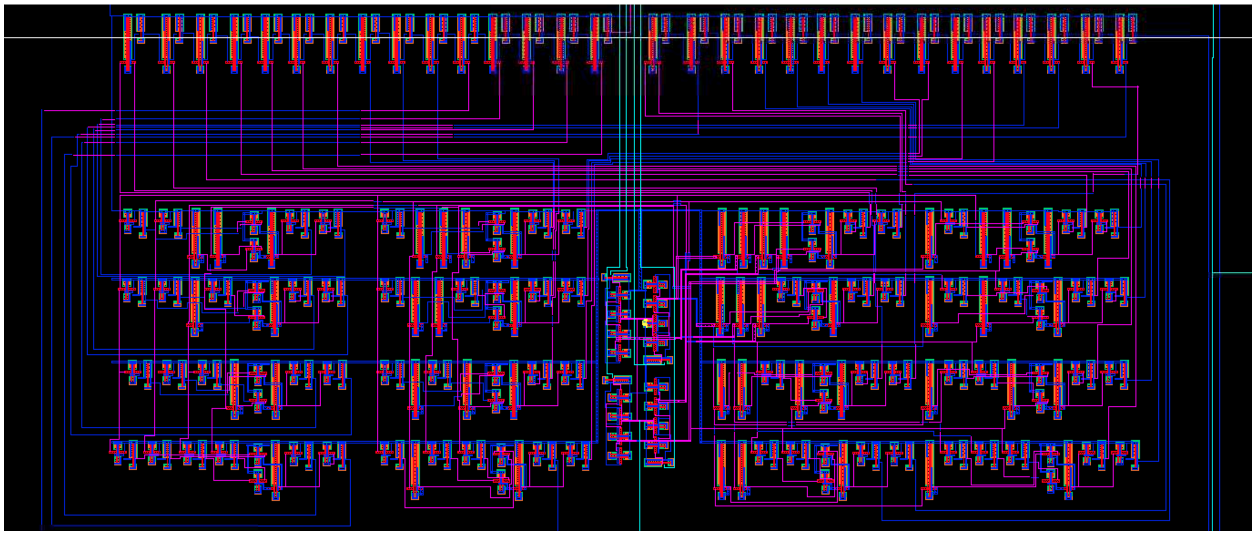


Figure 8. Sbox 1 Layout

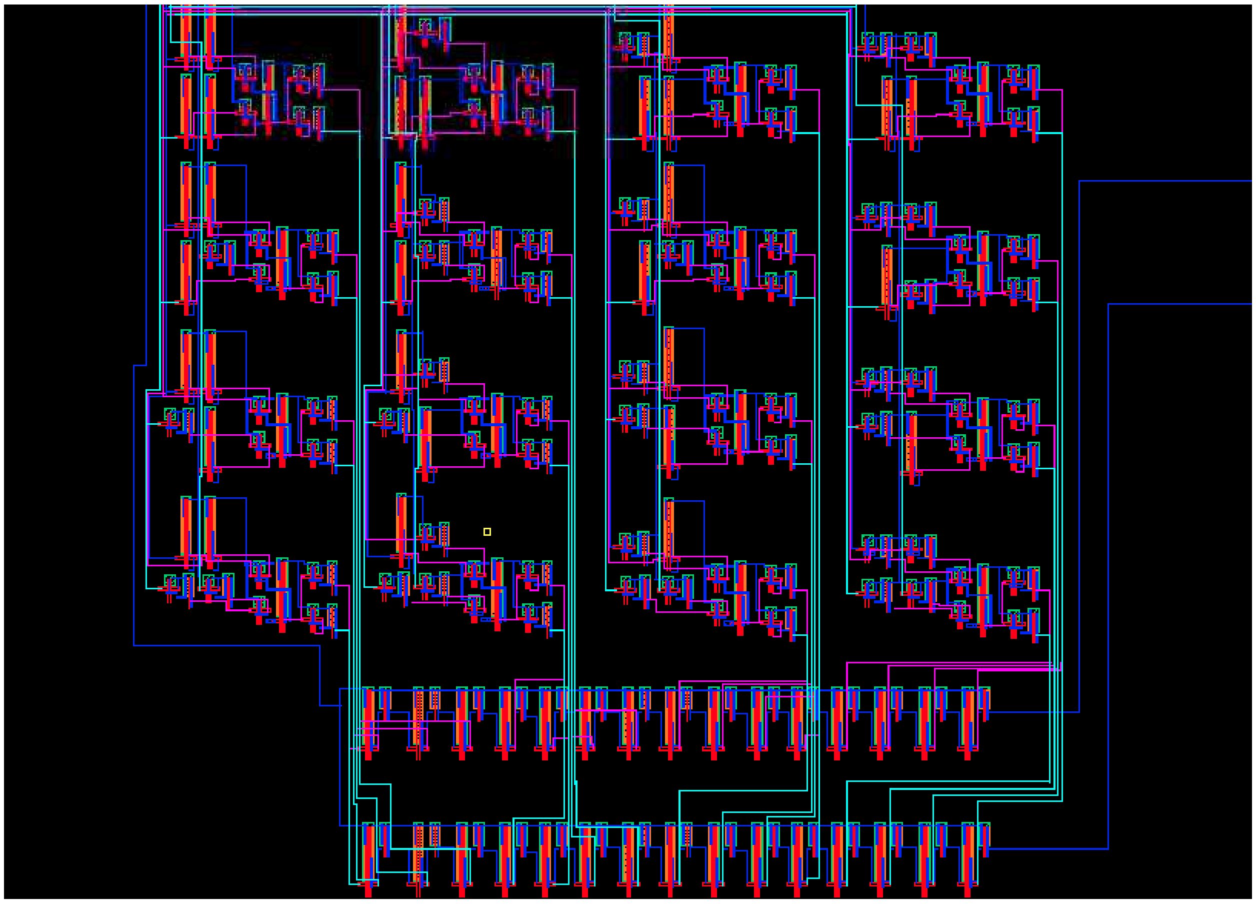


Figure 9. Sbox 2 layout

Our Design

For this project, we implemented the S-DES, or “simplified data encryption standard,” algorithm in hardware. The algorithm consists of various data permutations and bitwise operations used to encrypt an 8-bit long word of data with an associated 10-bit long key. The following are the algorithm’s steps and how we constructed them in hardware:

<u>Key Generation</u>: To generate our two keys k1 and k2, the given 10-bit key is run through a permutation P10, then cut in two and each half is shifted either 1 or 3 bits left independently, depending on whether we’re making k1 or k2, and, finally, they’re reduced to 8-bit words via the P8 permutation. In hardware, we realized these operations via simple wire routing–if bit X was meant to finish in position Y for a given step, we simply routed it to that position for the next stage’s input.

Initial Permutation: Similarly to our key generation, this was implemented using wire routing.

The following operations are most of the remainder of the algorithm, and all member operations of the function f_k.

Expansion Permutation: The right half of the initial permutation is fed into f_k first, and the first step is E/P–it gets expanded to 8 bits via a permutation. We implemented this with wire routing.

XOR with Key 1: The output of E/P is fed into a bit-wise xor with the first key. We created a separate block for these, and each pair of bits was xored using the appropriate bits of key 1 from the key generation stage.

Sboxes: The output of the XOR operation is then fed into our sboxes. The sboxes take these two separate four bit inputs, and each output the two appropriate bits. This pair of two bits is then combined to receive a 4 bit number.

XOR with original input: The output of these sboxes is then put through XOR with the appropriate original bits from the input.

XOR with Key 2: The output of these xors, after being put thorough a permutation by rearranging the outputs, is fed into a series of xors with key 2.

Sboxes part 2: Again, the sboxes are utilized to map a pair of 4 bit inputs to a pair of 2 bit inputs.

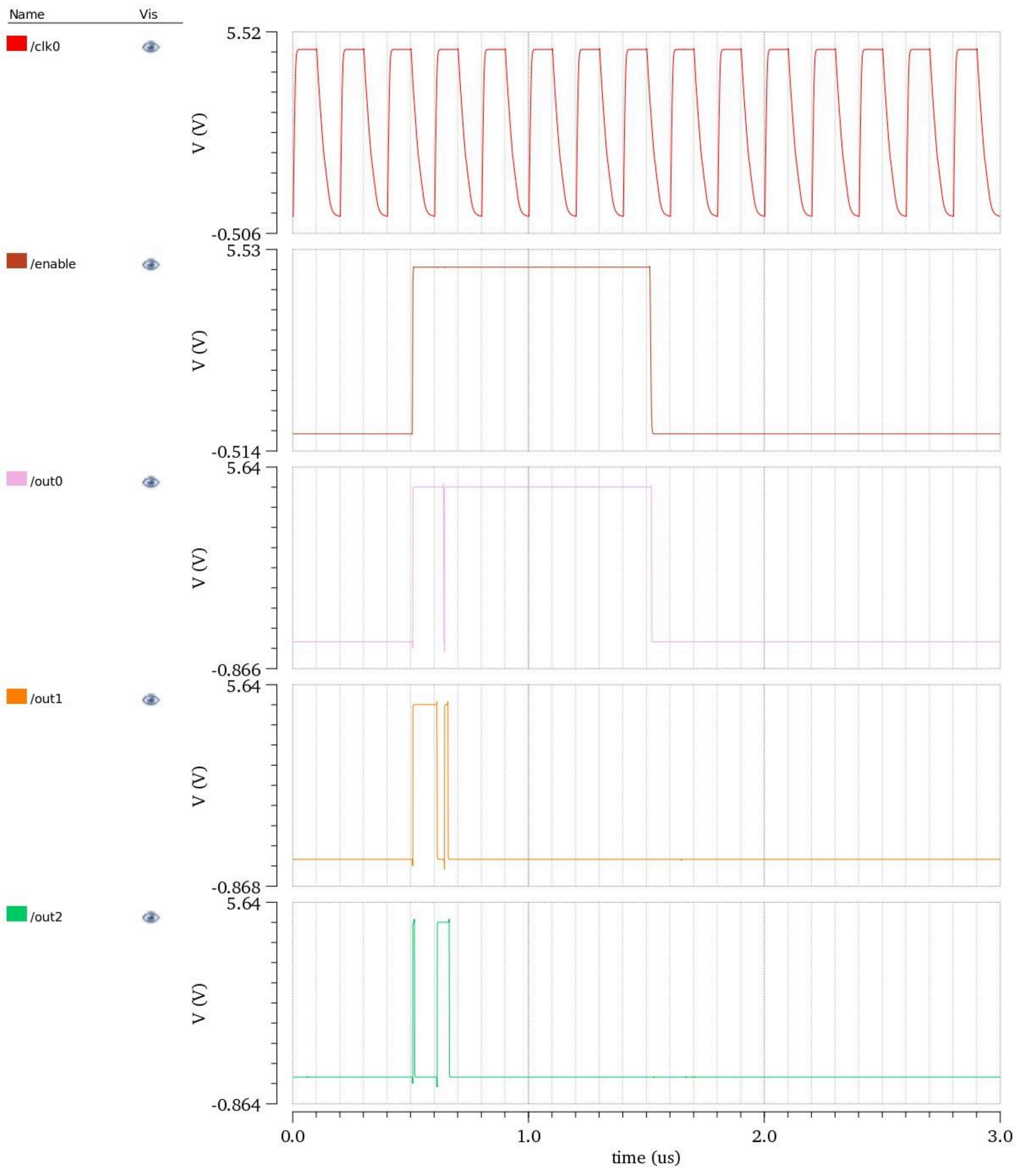
Final steps: Output of the sboxes is then fed into a series of xors along with the appropriate pattern of original bits. The output of these xors is then put through permutation along with the outputs from the xors in the first step. This new 8 bit permutation is our final result, and is fed through buffers and the enable logics before returning the encrypted output.

Sbox 1 & 2: To implement Sboxes, the group implemented logic to identify which box in the Sbox grid would need to be accessed. For example, if the team was looking for "1000" in Sbox 2, an and gate would be used to look for the most significant 1, while nand would be used to look for the following zeros. This way, the pattern could be correctly identified. Then, the outputs of these gates are fed into an and gate. This and gate will only output 1 if all other inputs are 1 (box "1111"). The output of this 4 input and gate are fed into two individual and gates. This way, if we desire a 1 from this specific pattern, we pull the other pin of the and gate high. And if we desire a 0 as the other output, we can pull the other gate input low. This will give us the desired output by following this methodology for each cell in the sbox. These implementations can be seen in the Sbox schematics above. For implementing the layout, we simply connected our previous gate layouts according to the created schematic.

Simulations

CSE563_final_project des schematic 19:51:22 Mon May 6 2024

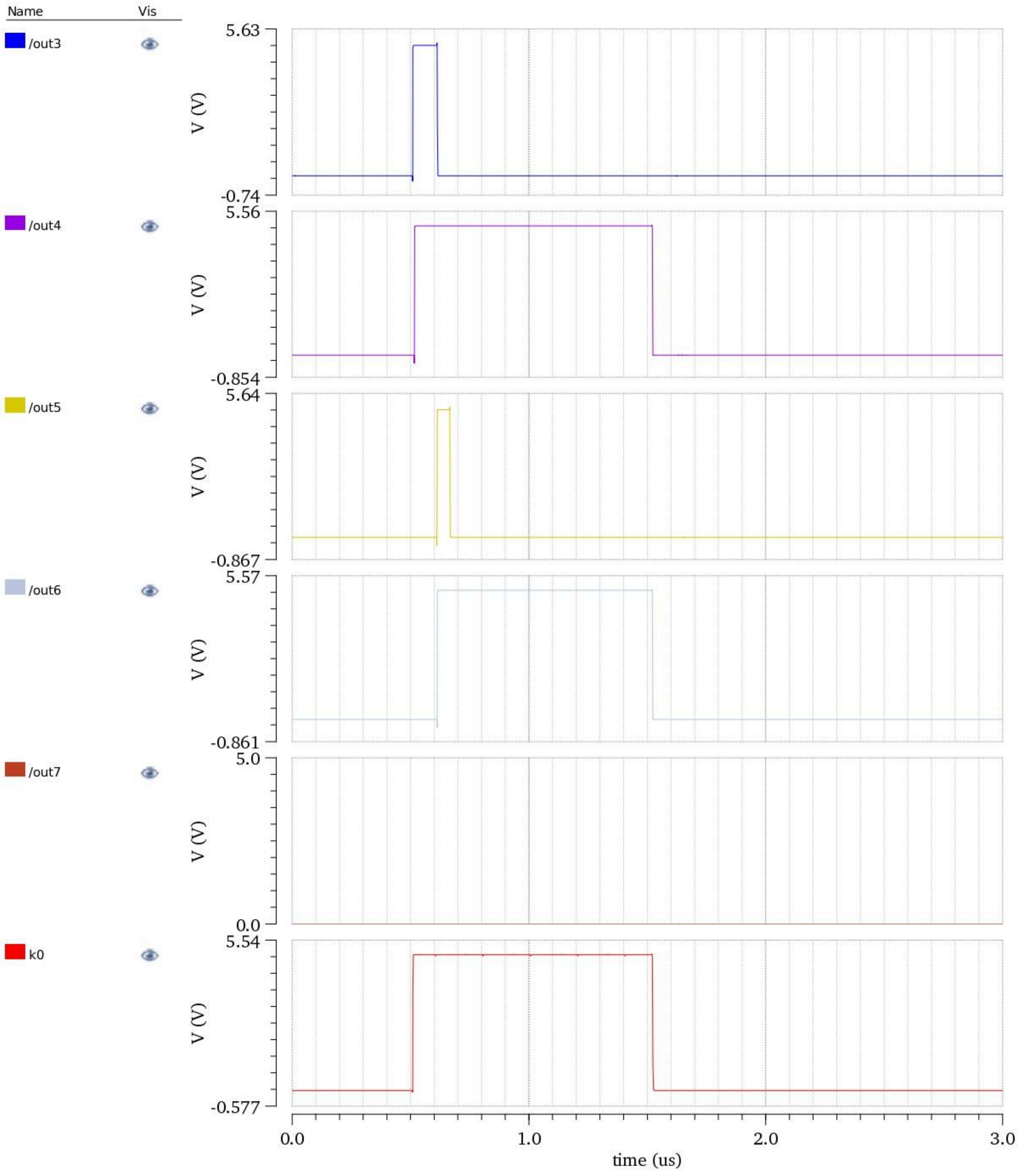
Transient Response **Mon May 6 19:40:58 2024**



Printed on
by qinzhounick

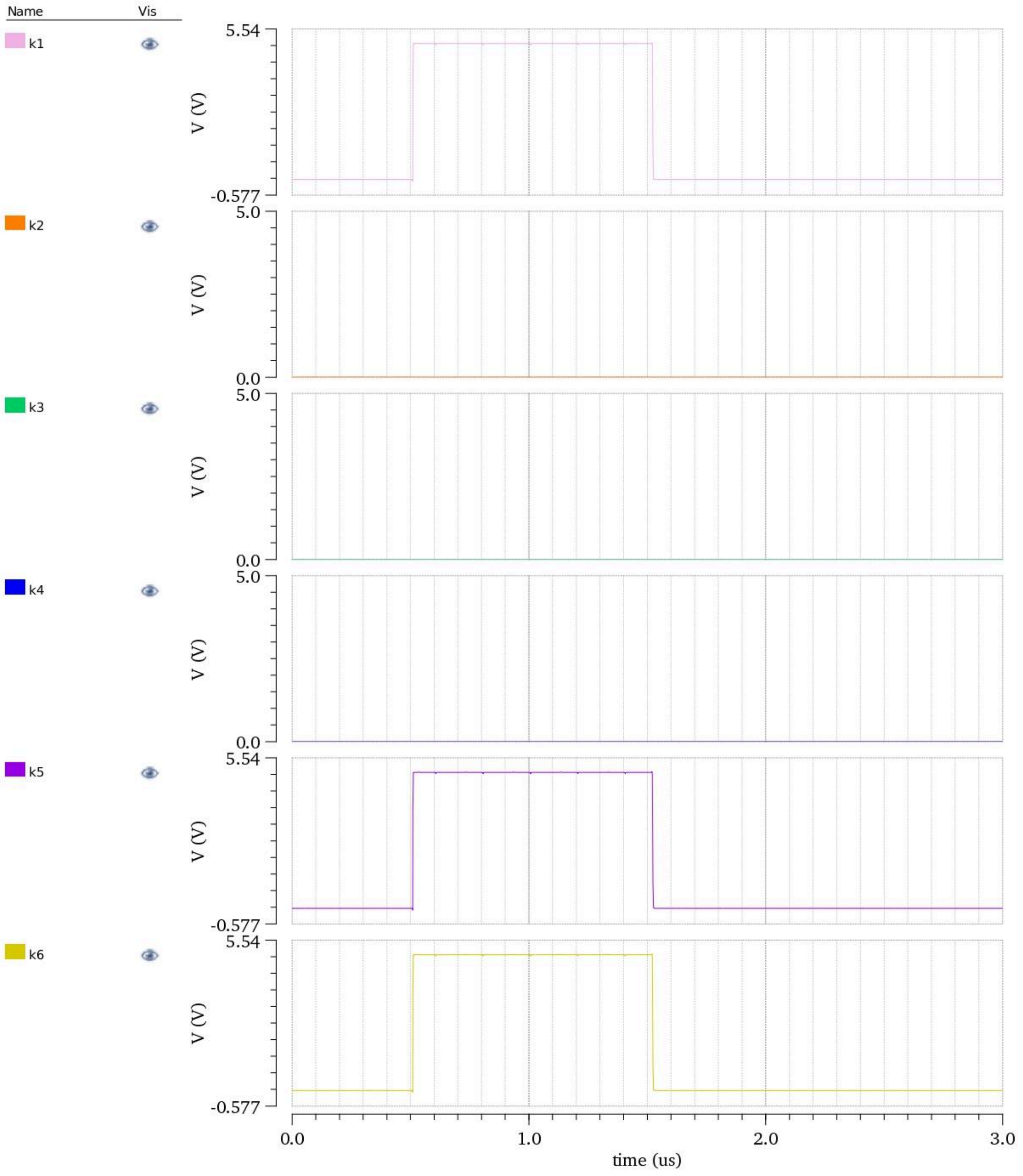
Transient Response

Mon May 6 19:40:58 2024

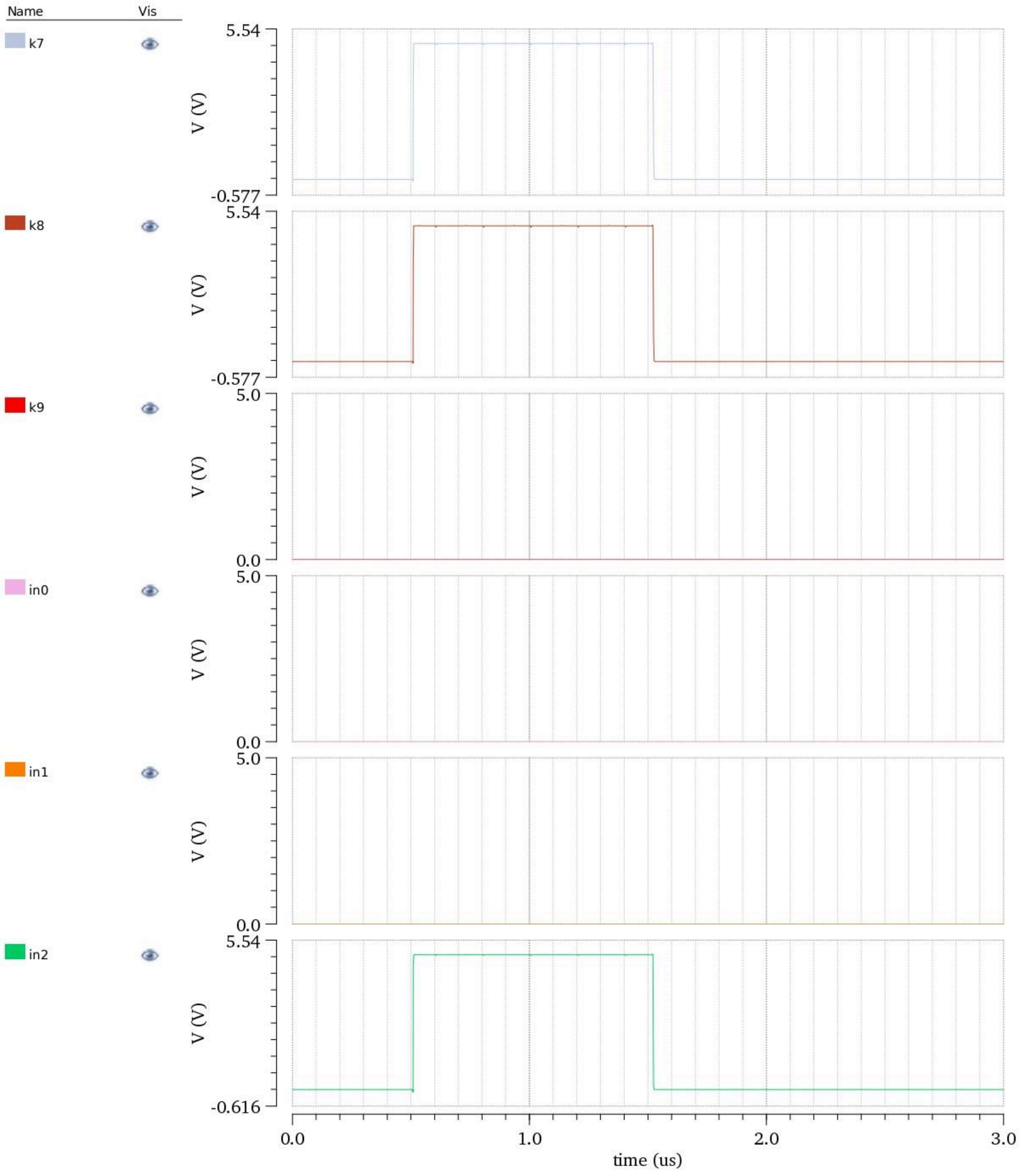


Transient Response

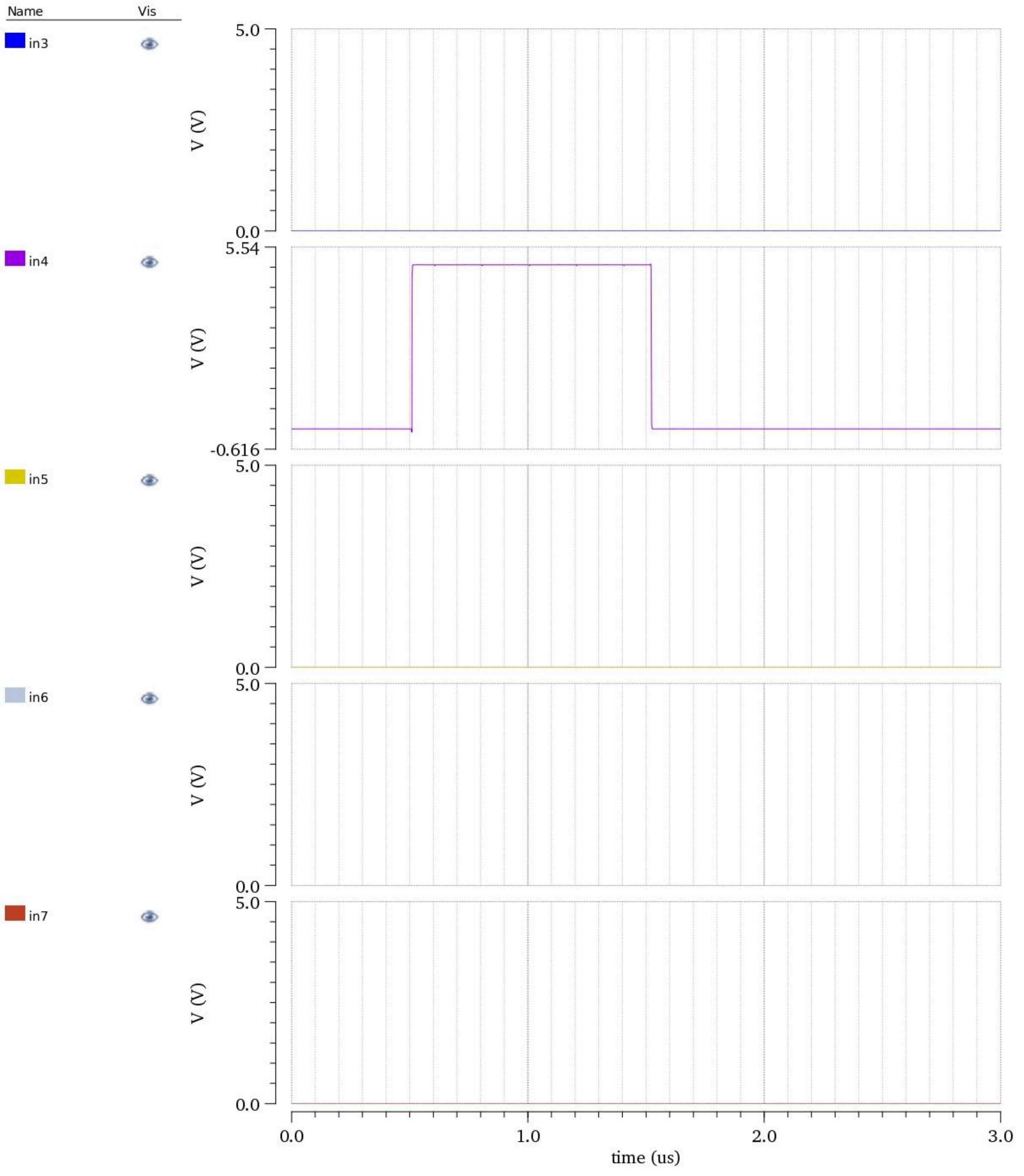
Mon May 6 19:40:58 2024



Transient Response **Mon May 6 19:40:58 2024**



Transient Response **Mon May 6 19:40:58 2024**



Operation Specifics

All basic gates utilized in this homework assignment were designed in homework assignments with a threshold voltage of 3.3 Volts. This was achieved throughout the semester by calculating a correct W / L ratio between the nmos and pmos. It is also important to note our design has a 2 cycle clock delay in the worst case scenario. Outputs are not asserted at the same time, this is the worst case scenario. This is because of the flip flop in the sbox stage of our design.

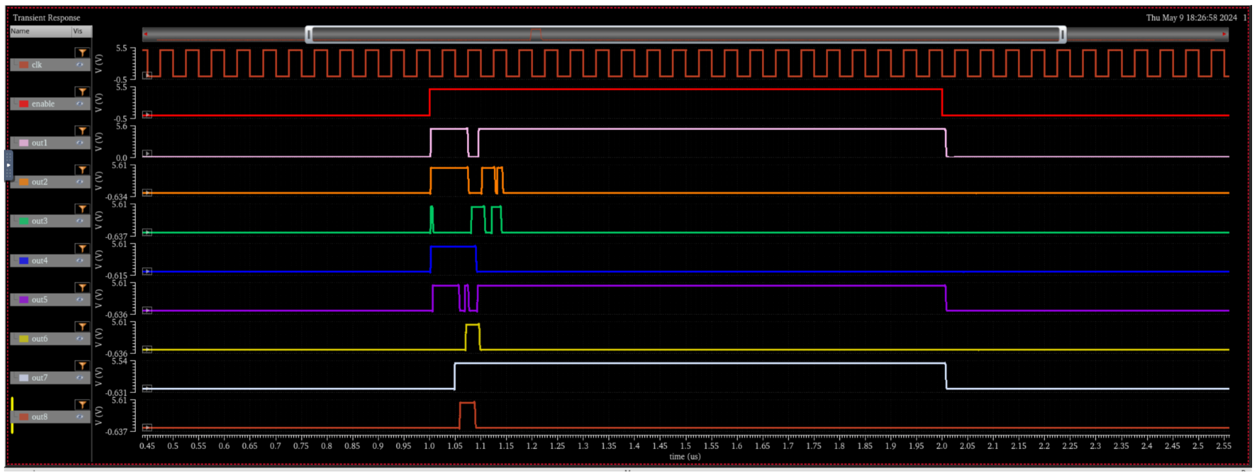
Future Work

Once the team completed the layout, the LVS netlist failed to match. Nets will need to be double checked in order to correctly simulate our layout. Additionally, there are some fluctuations in our output which we are investigating further.Some output noise can be seen on logical lows on the outputs in our simulations as well, which we will need to investigate further.

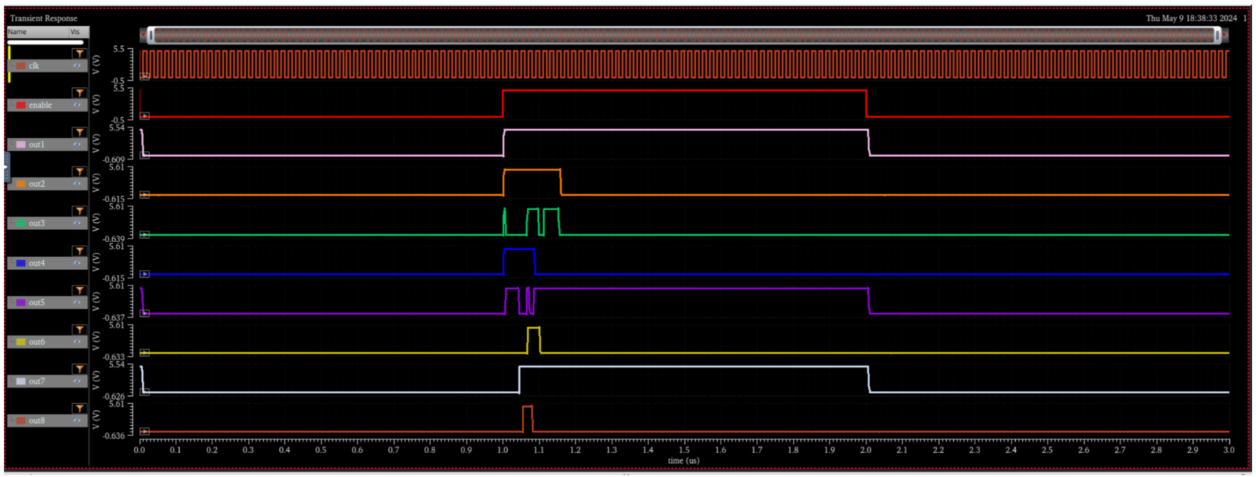
563 Analysis

We initially set the clock period to be 100 ns. We also tested our design with clock periods of 50 ns, 20 ns, and 1 ns. Our design functions as expected with a clock period of 100ns, when the clock gets faster, the output signals take longer to stabilize, and the delay time is longer. Below are the simulation outputs with only clk, enable, 8 bits output for readability.

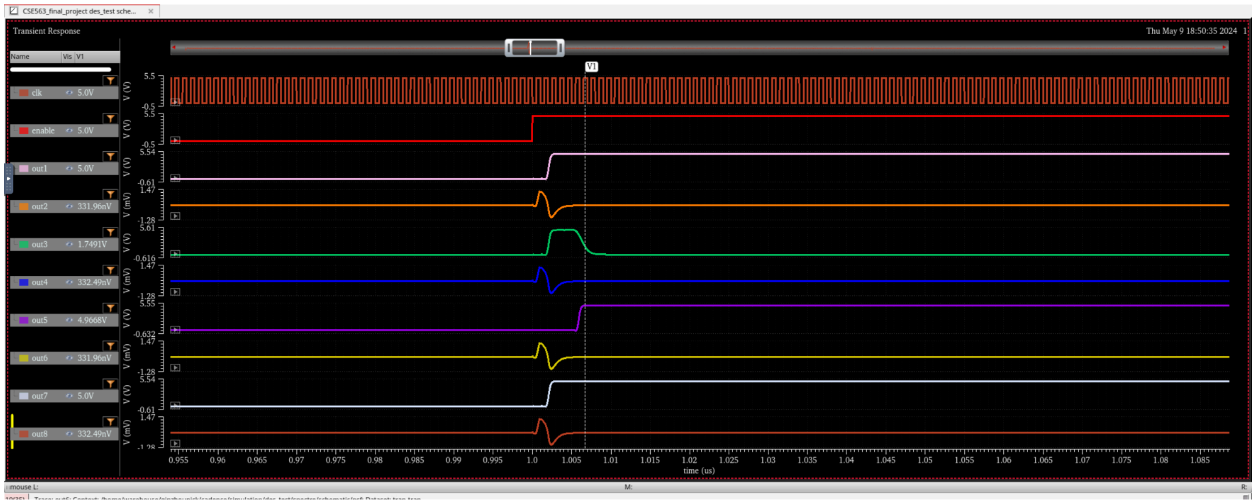
- 50 ns clk,
    - Worst case: 3 clk periods delay



- 20 ns clk
    - Worst case: 8 clk periods delay



- 1 ns clk
    - Worst case: 8 clk periods delay


CSE563_final_project des_test sche...
Transient Response
Thu May 9 18:50:35 2024
clk 5.0V
enable 5.0V
out1 5.0V
out2 331.96nV
out3 1.7491V
out4 332.49nV
out5 4.9668V
out6 331.96nV
out7 5.0V
out8 332.49nV
V1
time (us)