Marcus Iqbal, Rob Mfon, Will Plucinsky, Dante Sawatzky
ECEC 472 - IC Design Project Final Report

Hummingbird Final Report

Specifications

The Hummingbird algorithm is a novel, ultra-lightweight encryption algorithm used for resource-constrained devices due to its hybrid structure of block cipher and stream cipher.

The exact encryption process is as follows: A 16-bit plaintext block P_{Ti} is encrypted by executing addition modulo 2^{16} of P_{Ti} and the content of the first internal state register RS_1. The result of the addition is then encrypted by the first block cipher Ek_1. This procedure is repeated in a similar manner for another three times and the output of Ek_4 is the corresponding ciphertext CT_i.

Design

The Hummingbird Encryption IC takes a 16 Bit Plain Text Input and returns a 16 Bit Cipher Text. This is completed by using a Linear Feedback Shift Register, 16 Bit State Registers, 16 Bit Block Ciphers, and 2 and 3 input 16 Bit Adders. This is depicted in the figure below.

The overall structure of the **Hummingbird** encryption algorithm (see Figure 1 (a)) consists of four 16-bit block ciphers E_{k_1}, E_{k_2}, E_{k_3} and E_{k_4}, four 16-bit internal state registers $RS1$, $RS2$, $RS3$ and $RS4$, and a 16-stage Linear Feedback Shift Register (LFSR), where the 256-bit secret key K is divided into four 64-bit subkeys k_1, k_2, k_3 and k_4 which are used in the four block ciphers, respectively. After a system initialization process with four random nonce values (see Figure 1 (b)), a 16-bit plaintext block PT_i is encrypted by first executing a modulo 2^{16} addition, denoted by $\boxplus$, of PT_i and the content of $RS1$. The result of the addition is then encrypted by the first block cipher E_{k_1}. This procedure is repeated in a similar manner for another three times and the output of E_{k_4} is the corresponding ciphertext CT_i. Moreover, following the internal state updating process described in Figure 1 (a), four internal state registers will also be updated in various and unpredictable ways.

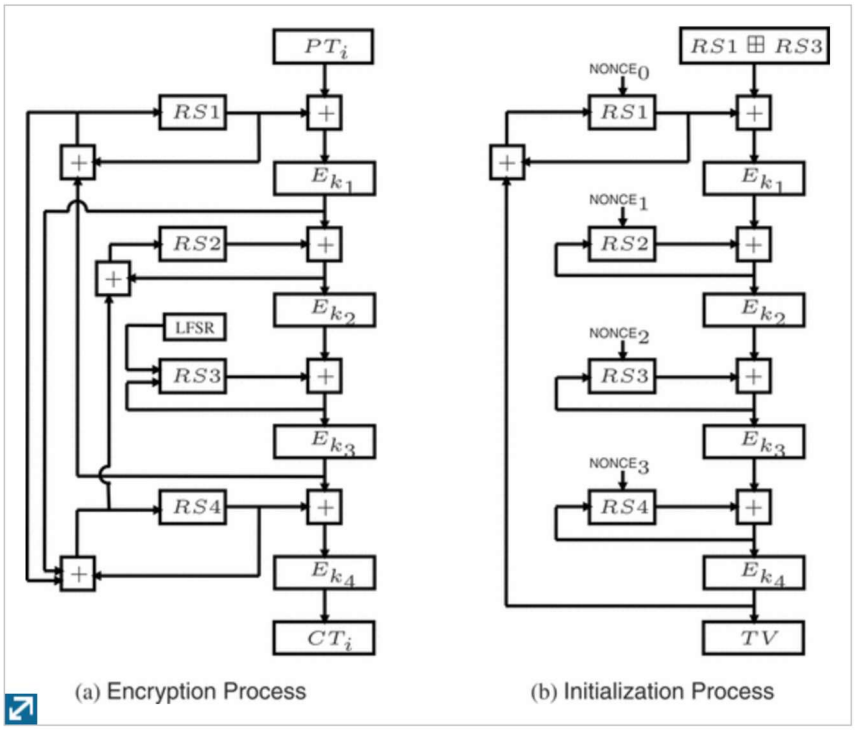


Figure 1. Encryption Block

The 16 bit plaintext word input initially passes through a 2 input, 16 bit adder with the output of the first 16 bit register, RS1. The output from this adder is then inputted into the first block cipher block, EK1, along with the global secret key. The process for the block cipher is depicted in Figure 2. The adder and block cipher process then repeats with additional semi-randomness added in through feedback loops a linear feedback shift register after the second stage.

Four S-Boxes Given in Hexadecimal Notation

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	8	6	5	F	1	C	A	9	E	B	2	4	7	0	D	3
$S_2(x)$	0	7	E	1	5	B	8	2	3	A	D	6	F	C	4	9
$S_3(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D
$S_4(x)$	0	7	3	4	C	1	A	F	D	E	6	B	2	8	9	5

Linear Transform $L : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$

$L(m) = m \oplus (m \lll 6) \oplus (m \lll 10)$

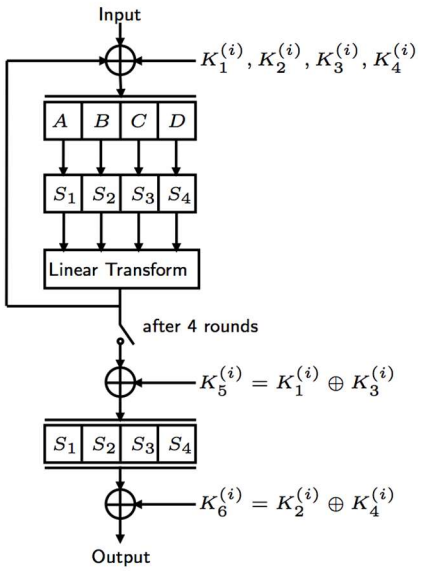


Figure 2. Block Cipher Process

Linear Feedback Shift Register

The Linear Feedback Shift Register (LFSR) generates a pseudorandom 16 bit word based on the input bit. The LFSR uses XORs and Flip-flops in order to achieve the output. There are 16 Flip-flops that are connected in series with the a clock signal as the CK input for every FF. The first FF input is the input bit touched on earlier. The output of each FF is the input for the next FF. Each FF's output is grab as a bit for the LFSR. This creates the 16 bit word. The XORs are used to add randomness to the LFSR and to add complexity to the IC. This is useful because without knowing the position of the XORs it is nearly impossible to reverse engineer the LFSR Output. The Schematic and Results of the LFSR are discussed later on.

Adders

The three input and two input, sixteen bit adders are both comprised of full adders. Sixteen in the case of the two input and thirty two in the case of the three input. The full adder was created using 2 XORs, 2 ANDs, and 1 OR gate in order to generate the `sum` and `cout` of the two inputs. The two input adder had a measured propagation delay of 200 ps, and the three input adder had a delay of 400 ps. This delay is due to the signal needing to go through every single full adder. Schematics and layouts of the adders can be found in the appendix.

16-Bit Register

The register uses 16 rising edge flip flops to store the input signal. In the schematic view of the register, as seen in Appendix A, the input and output data is sent through a bus. The clock and reset pins are set to receive an input. The layout of the register is seen in Appendix B where the blocks of the flip flops were able to be placed close together to provide a compact register. In Fig. 3 the simulation of the schematic shows that the results of the 16-Bit Register are correct. The propagation delay of the circuit is measured to be 40ps. The transistor count is 512.

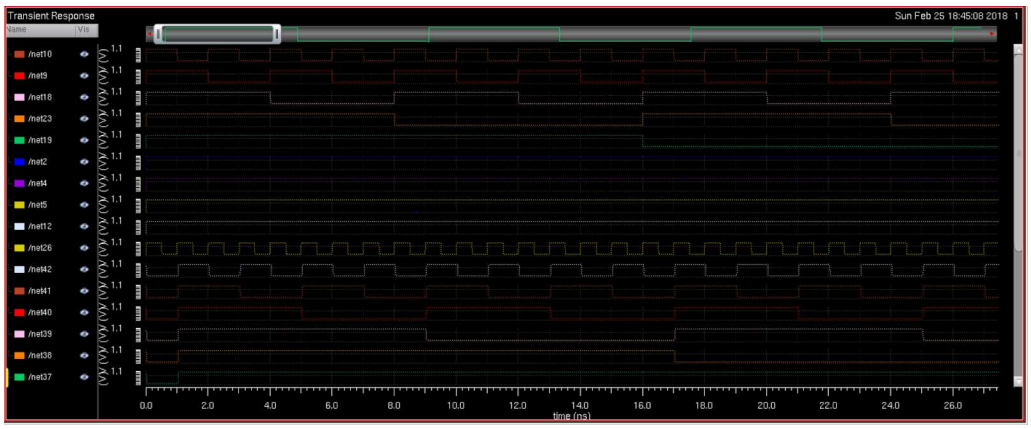


Figure 3. 16-Bit Register Simulation results

Block Cipher

As seen in figure 2, the block cipher takes a 64-bit key and a 16-bit text input. It XOR's a 16-bit chunk of the key and the text input, masks the result using an S-Box and performs a linear transform function of the result. This is repeated another three times before sending the output of this operation through another two rounds of XOR with chunks of the key with masking in between these rounds.

To implement this, several building blocks were designed and tested.

- 16-Bit XOR
  - This block was implemented using 16 2-input XOR's provided in the standard cells
- S-Block
  - Masking through the S-Block was implemented by designing a 4x16 decoder and a NAND based ROM

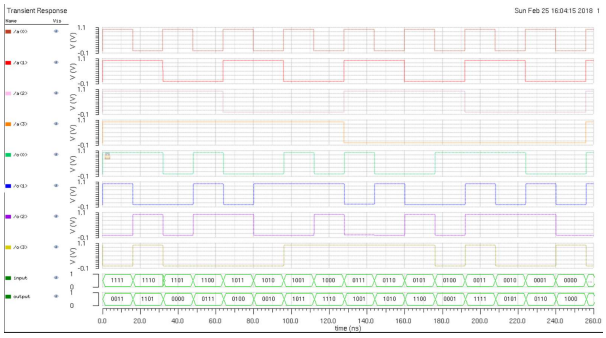


Figure 4: S-Block Functionality

- Linear Transform Block
  - The linear transform function performs a 3 input XOR between the input message, a 6-bit left-shifted version of the message, and a 10-bit left-shifted version of the message. This was implemented using the 3-input XOR's that were built for the 16-Bit XOR

The functionality of the block cipher is demonstrated in Figure 5. The propagation delay found for the block cipher is 1.26ns.

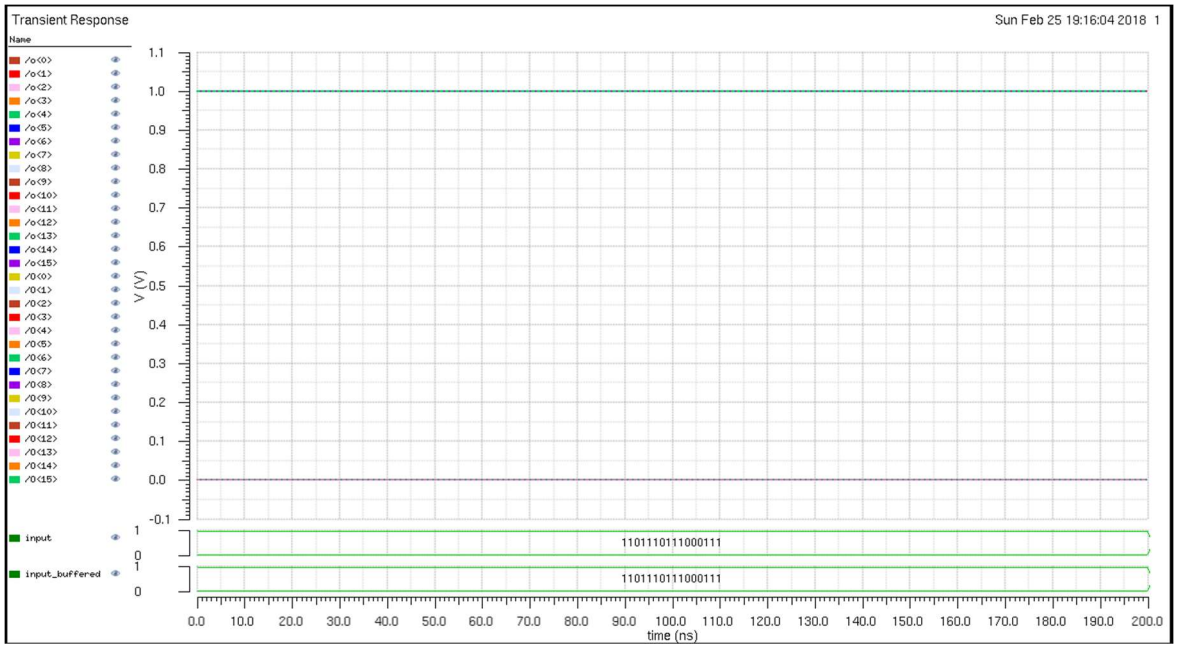


Figure 5: Block Cipher Schematic Simulation

Schematic and layout of the block cipher can be found in the appendix.

Testing Strategy and Results

In order to test the functionality of the individual block schematics, testing schematics were designed. For the full adder, which makes up the 2 and 3 input 16 bit adders, the two inputs were varied using VPULSE components and the delay time was analyzed on the simulation plot. Figure 6 shows the testing schematic, and Figure 7 shows the simulation waveform.

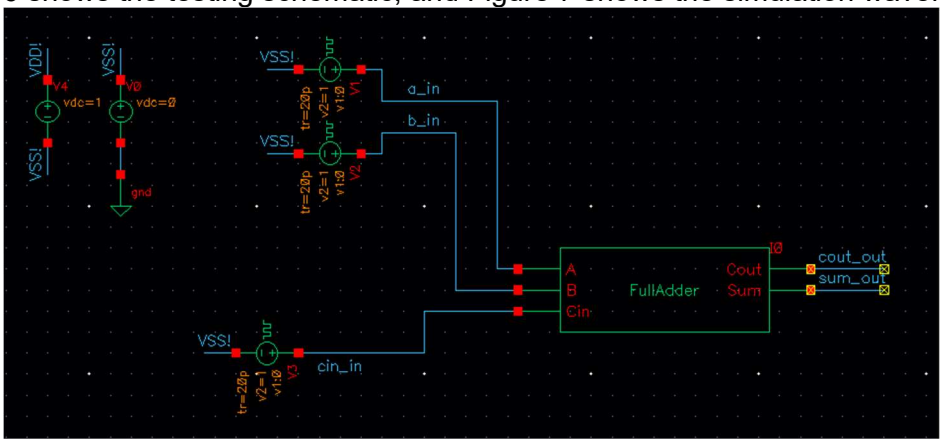


Figure 6. Full Adder Testing Schematic

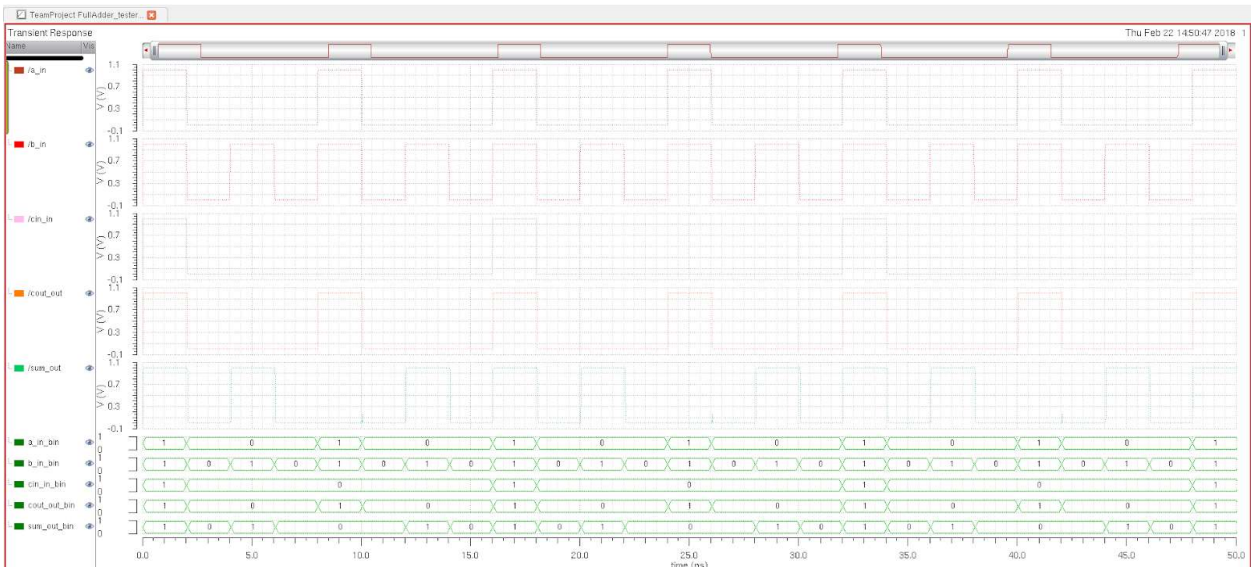


Figure 7. Full Adder Testing Schematic Simulation

Once the schematics met the design criteria for the block, the layouts were able to be designed. The layouts for the standard cells used to build the schematics were imported into the layout and connected accordingly. Due to issues in the bounding boxes of the standard cell layouts, the layouts couldn't be spaced close together or design rule check, DRC, errors would occur. An example of this placement for the full adder can be seen in Figure 8. The standard cells also had a number of grid related errors that we were unable to solve, as seen in Figure 10. Another issue we ran into with the standard cell layouts were the types of transistors they used. Although the schematics labeled the transistors as NMOS_VTL and PMOS_VTL, they were actually nmos and pmos transistors from the analog library which had an additional connection on the transistor. This created a large amount of net discrepancies when running the layout versus schematic, LVS, checks, which can be seen in Figure 11. Due to these standard cell layout issues, the DRC checks were considered passing when there were only grid errors, and LVS was considered passing when there were only net issues.

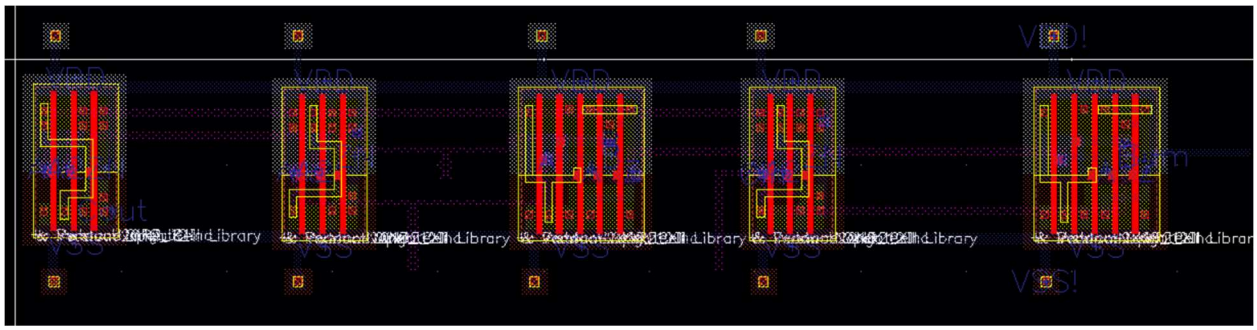


Figure 8. Full Adder Standard Cell Placement

Below is the results of the Linear Feedback Shift Register. As you can see the output of the bits are correct but reversed. This was discussed by our group. This was a simple error of a checkbox when bussing the output. The system bussed the bits as Out<15:0> rather than Out<0:15>. The propagation delay for the LFSR is 25.9 ps.

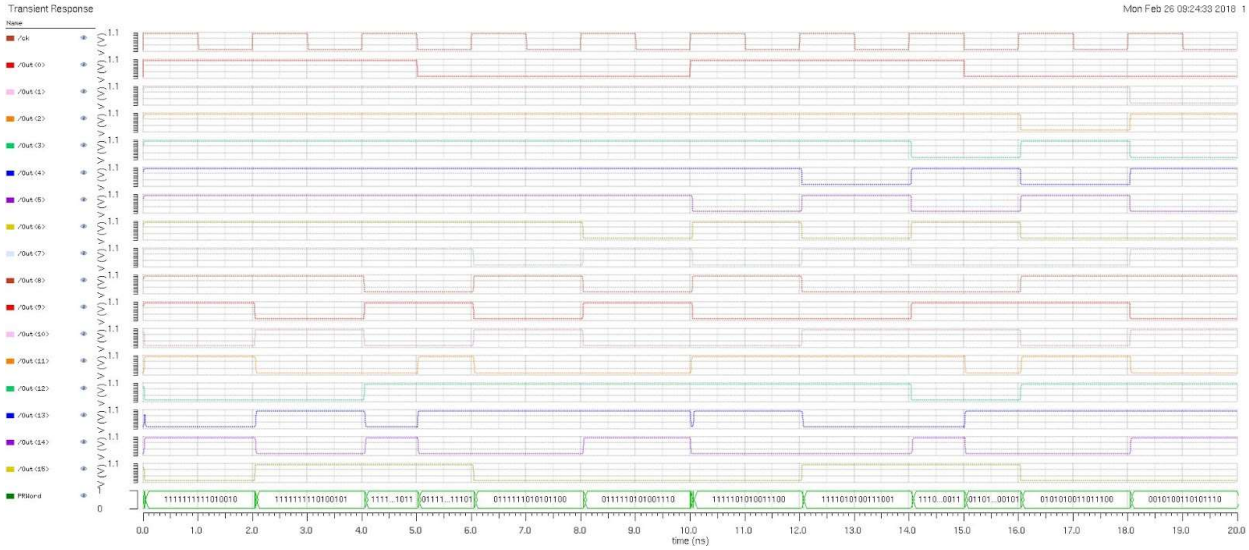


Figure 9. Results of Linear Feedback Shift Register

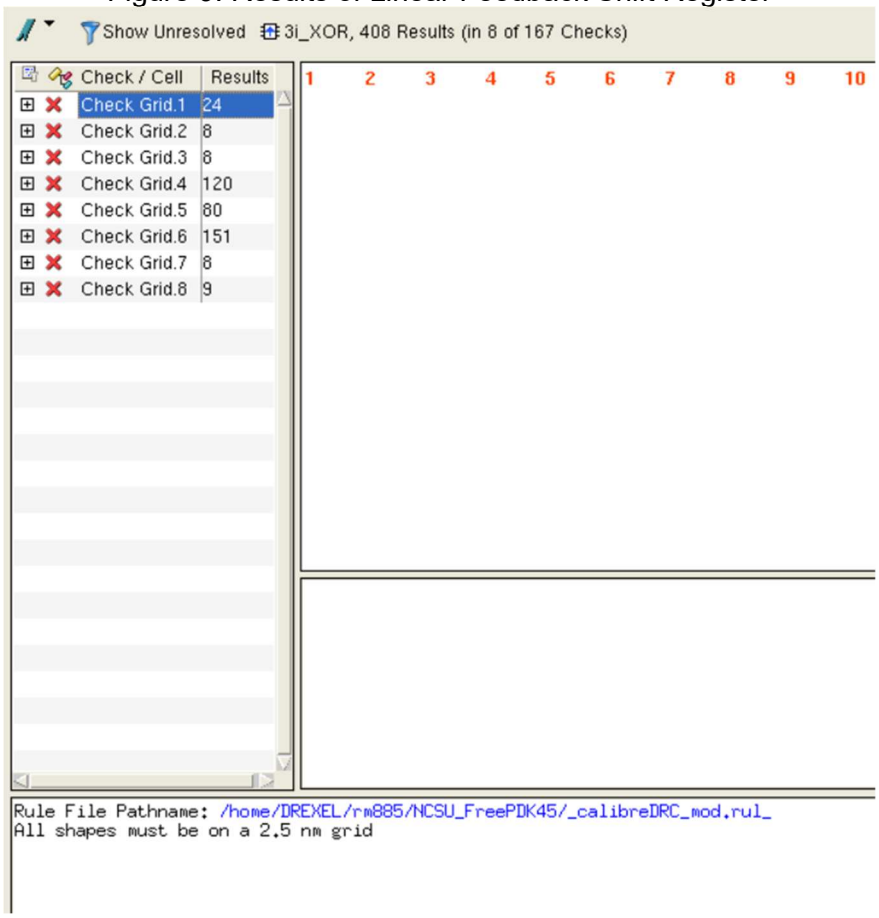


Figure 10. Grid DRC Errors

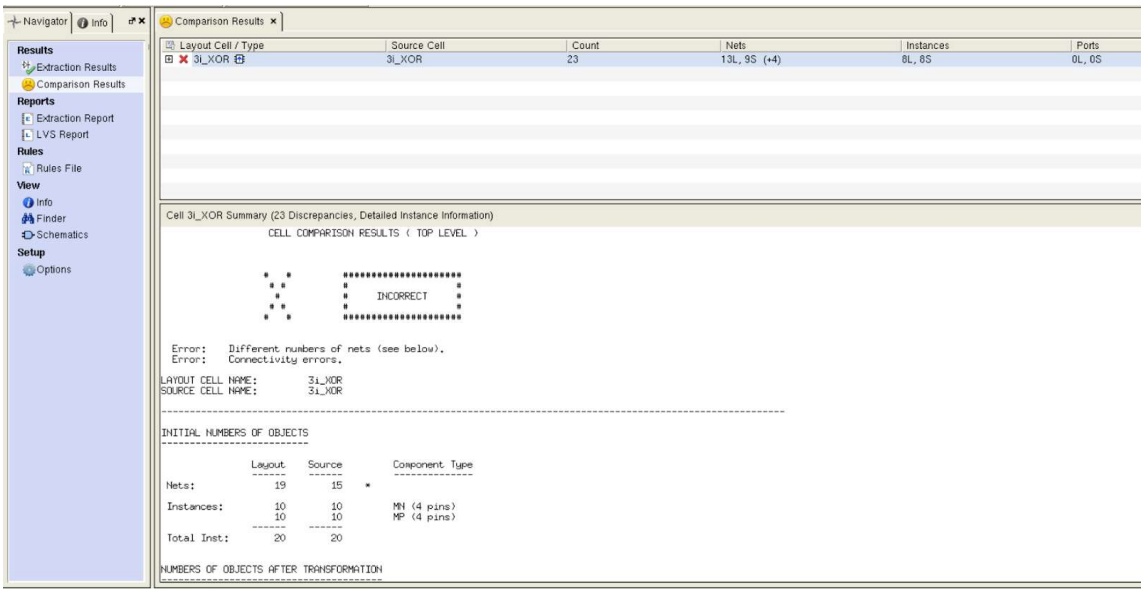


Figure 11. LVS Errors

We ran our clock conservatively during the simulation in order to ensure we didn't have any timing issues along the critical path as the simulation took a long time to simulate. The critical path was the pathway through the block ciphers and the adders. This could have been optimized by performing a more thorough analysis of the individual components that make up that pathway in order lower the propagation delay. In the block cipher for example we could have pipelined the rounds rather than them being separate for robustness.

Final Transistor Count: 35,454

Areas of Potential Optimization

Oversight

Complexity of Circuit - Additional time should be spent in optimizing the layouts of each of the components in the circuit as well as the layout of the final circuit (Layout of the final circuit and the full block cipher weren't completed). In hindsight, this project could have greatly benefited from developing our own standard cells instead of using the ones provided by the OpenCell library. A significant amount of time was spent in attempting to correct errors that weren't fully understood by the team members.

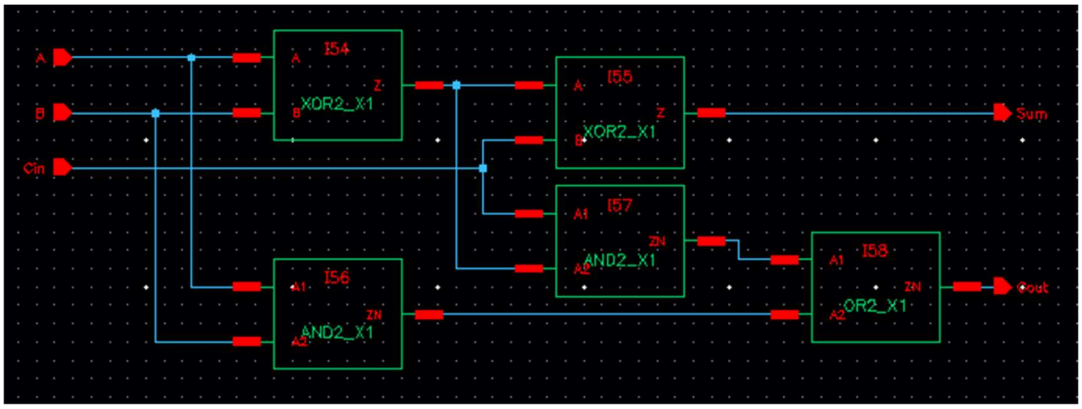
Lack of Knowledge

Power Analysis - Due to the long time it takes to simulate the full circuit (6h, 20 min with non switching inputs for a 8ns simulation), we weren't able to perform power analysis on the schematic of the full circuit in order to have an accurate measurement of the power our circuit would consume.

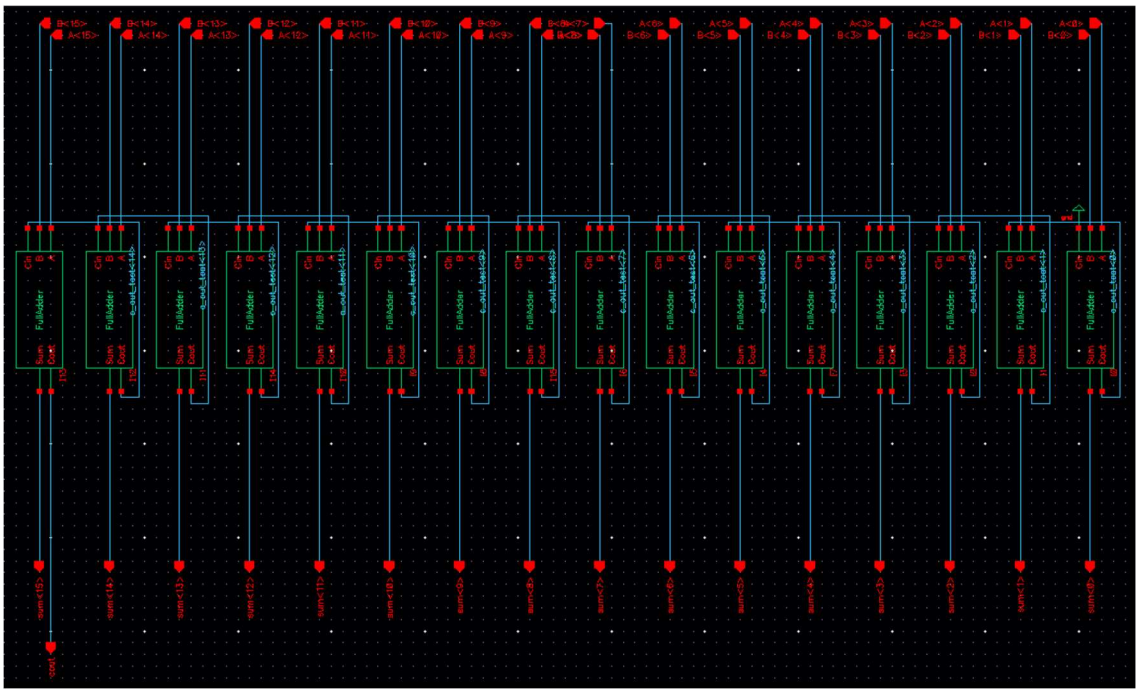
C# Simulation - A software implementation of the hummingbird algorithm would have ensured that the circuit designed works as expected. In our current state, we can ensure that there's a high degree of randomness that is applied to the input plaintext, and that it would be hard to decrypt the ciphertext without knowing the key.

Appendix

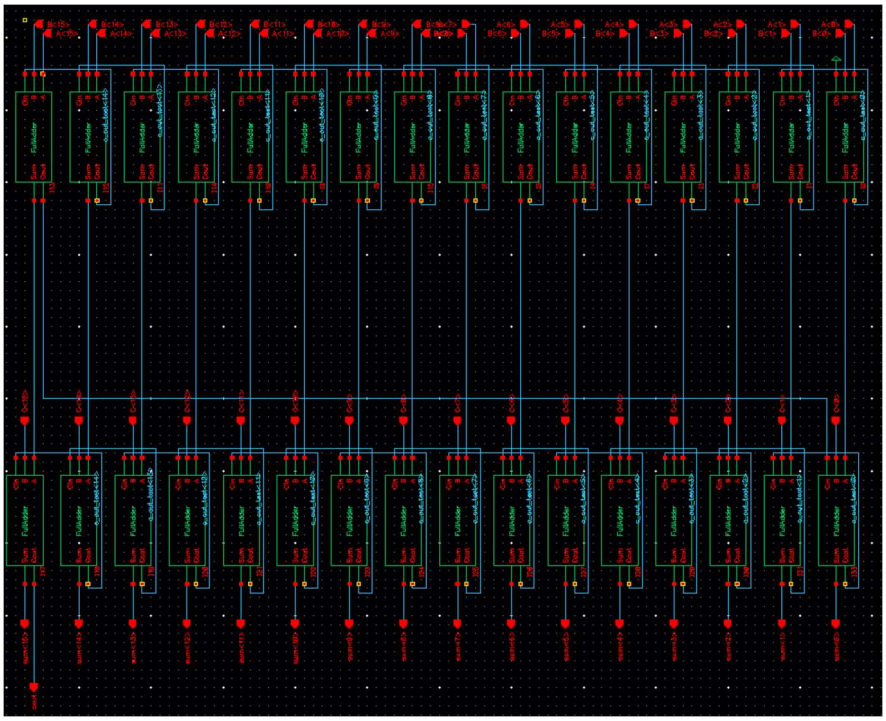
Appendix A: Schematics



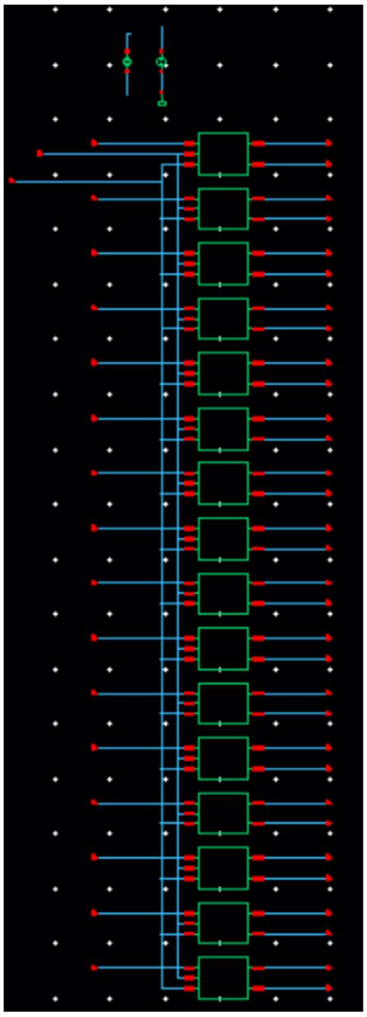
2-Bit Full Adder Schematic



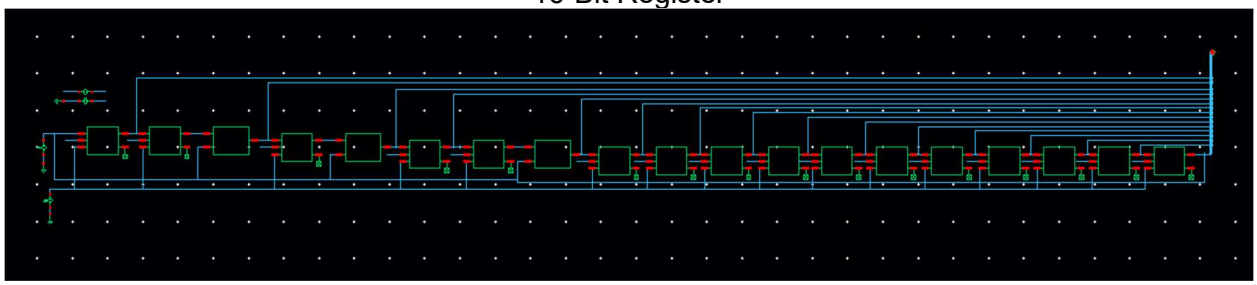
2 input 16-Bit Adder



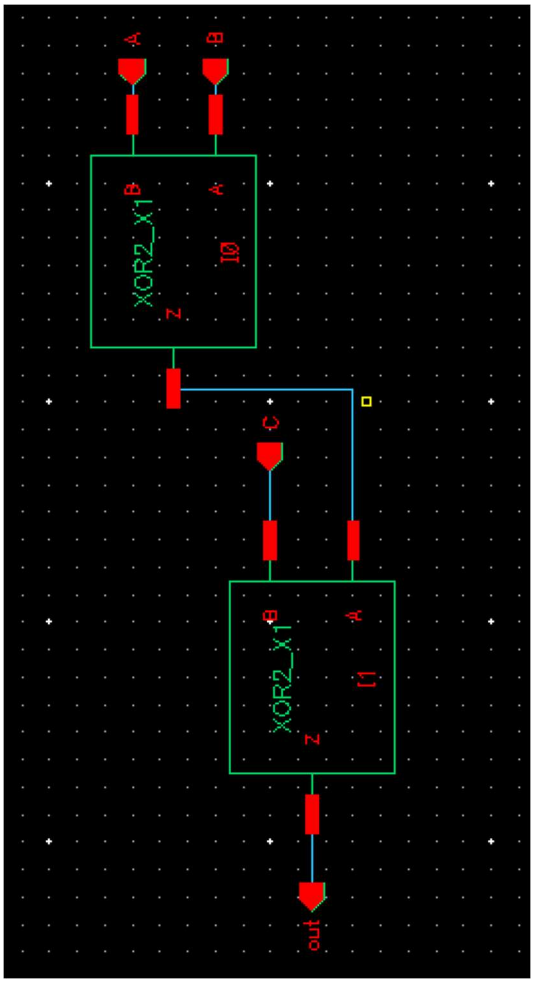
3 Input 16-Bit Adder



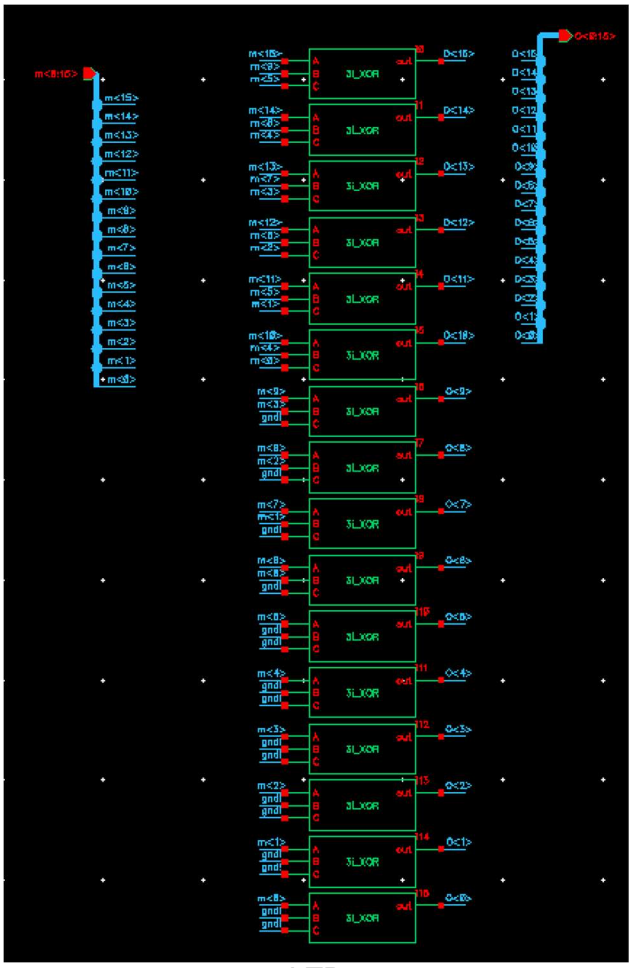
16-Bit Register



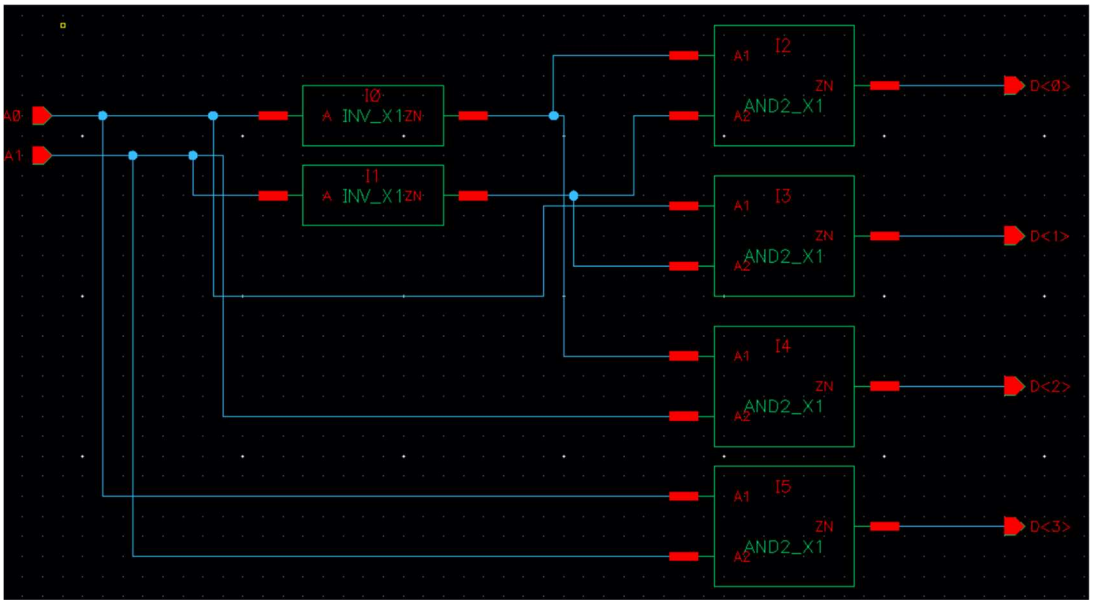
LFSR



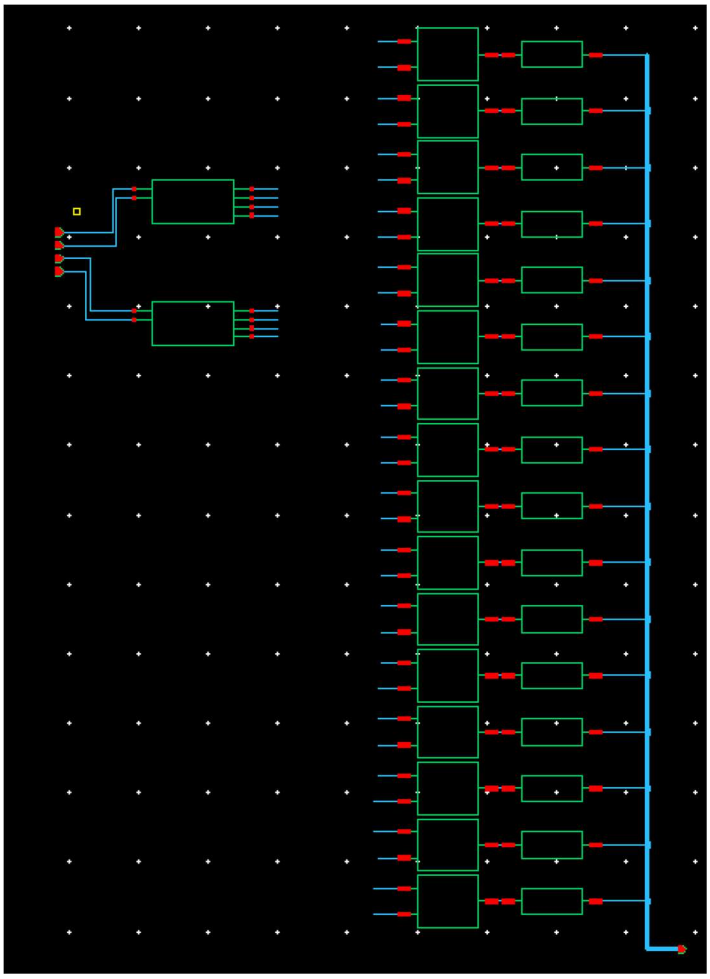
3 Input XOR



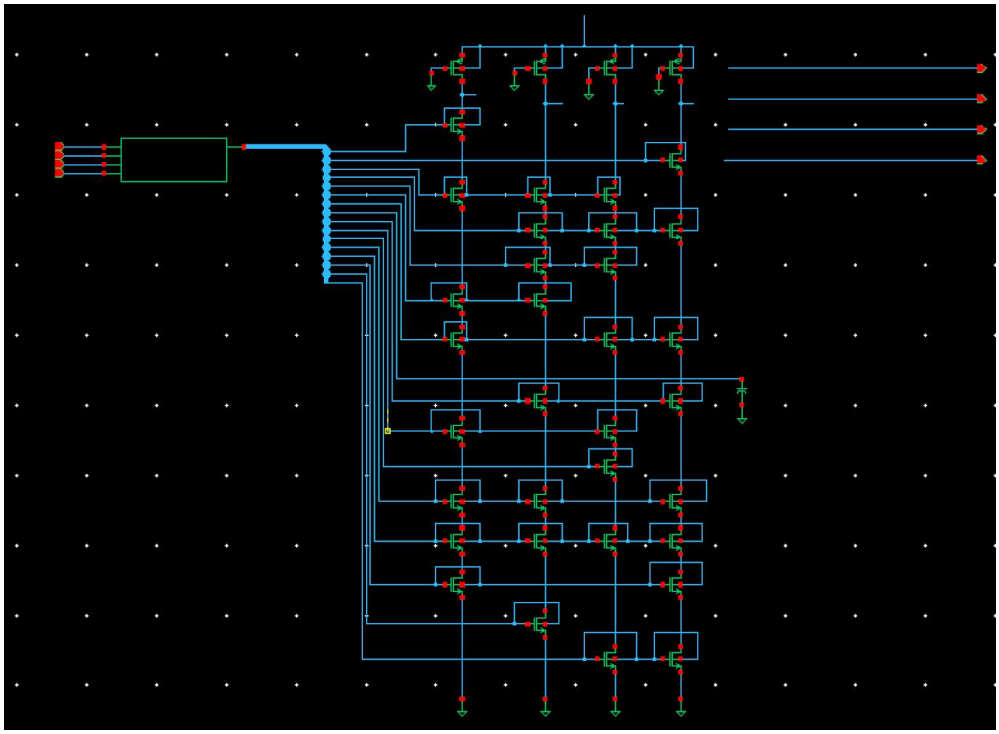
LTB



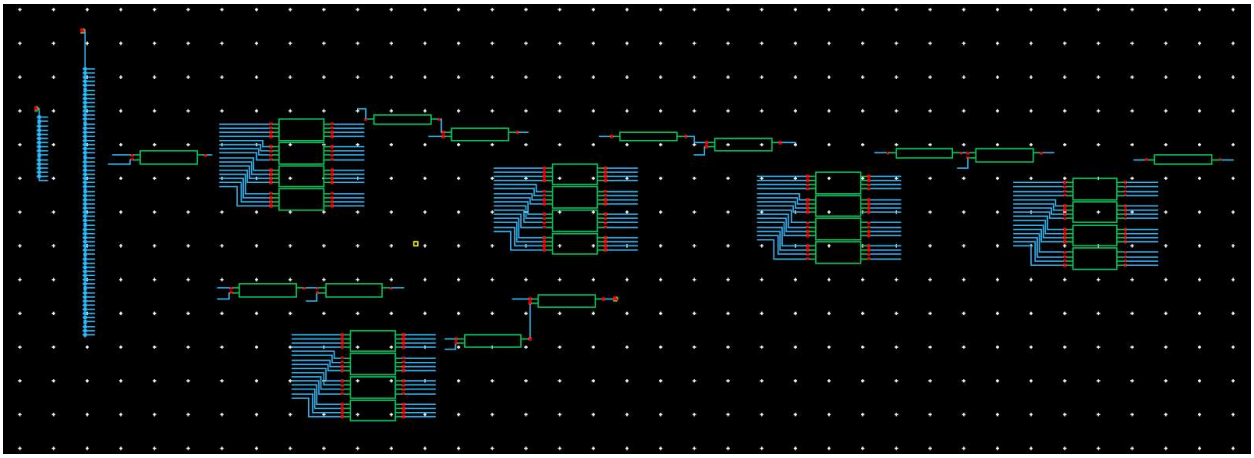
2x4 Decoder



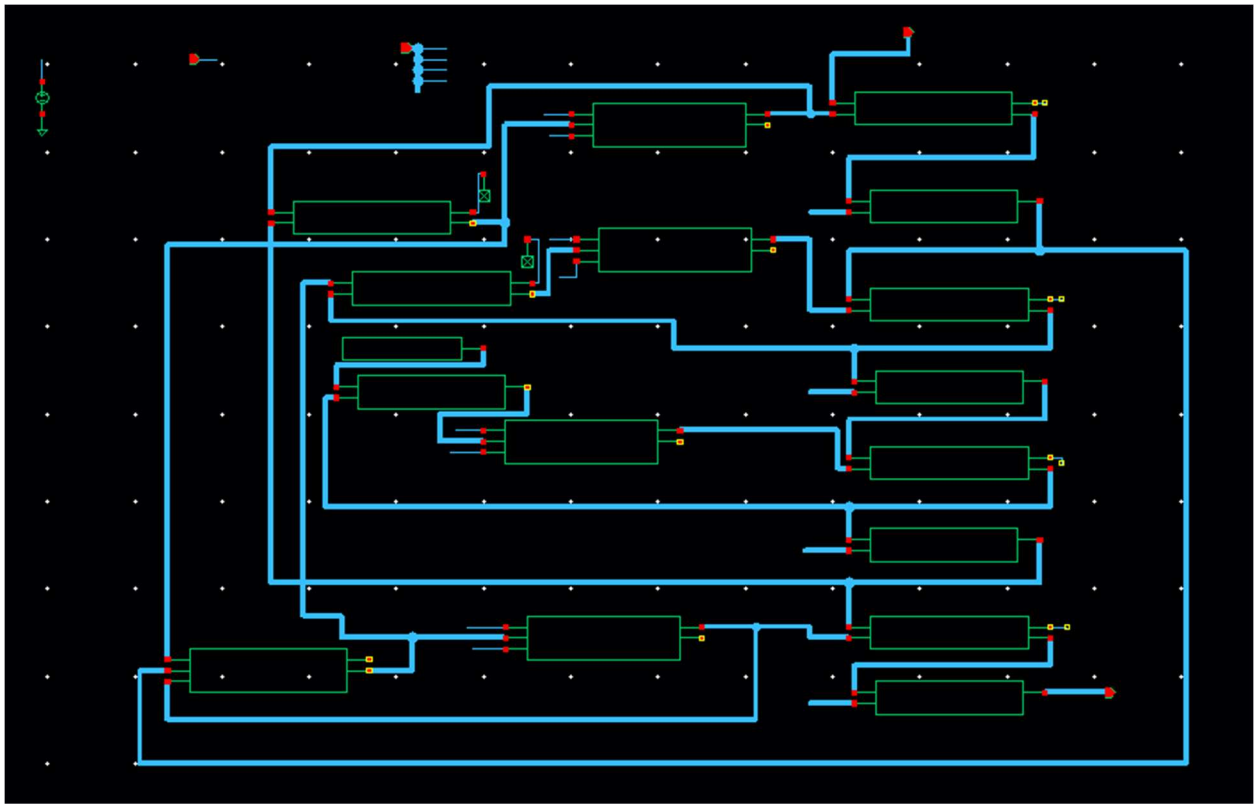
4x16 Decoder



S Block



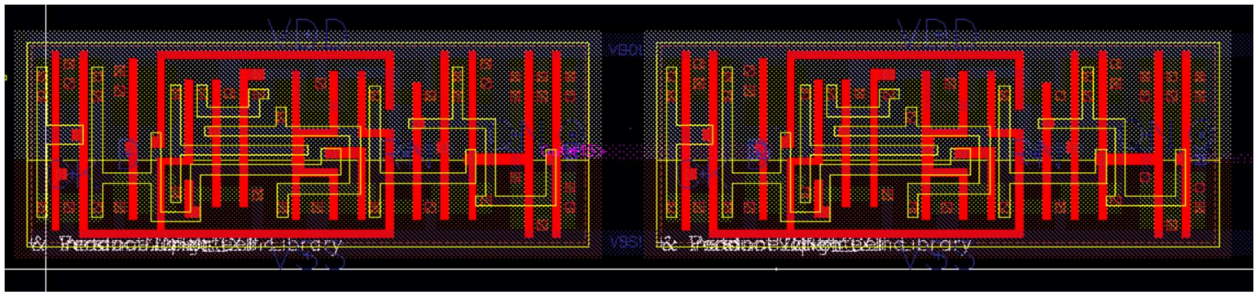
Block Cipher



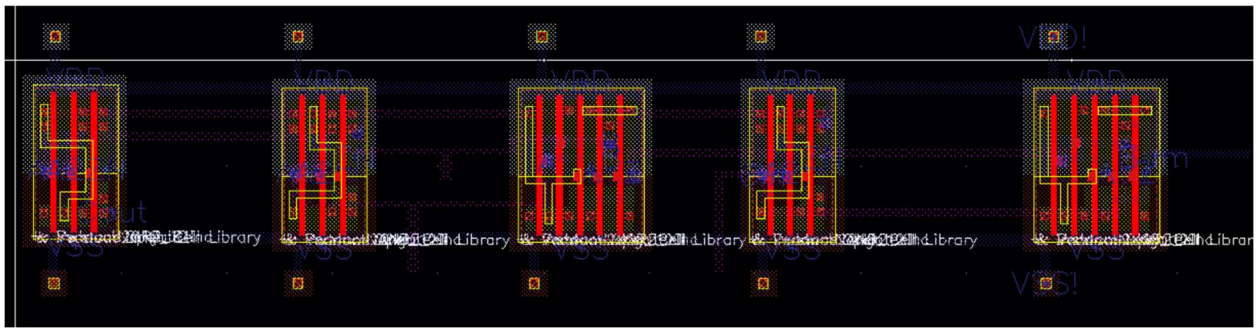
Encryption Block

Appendix B: Layouts

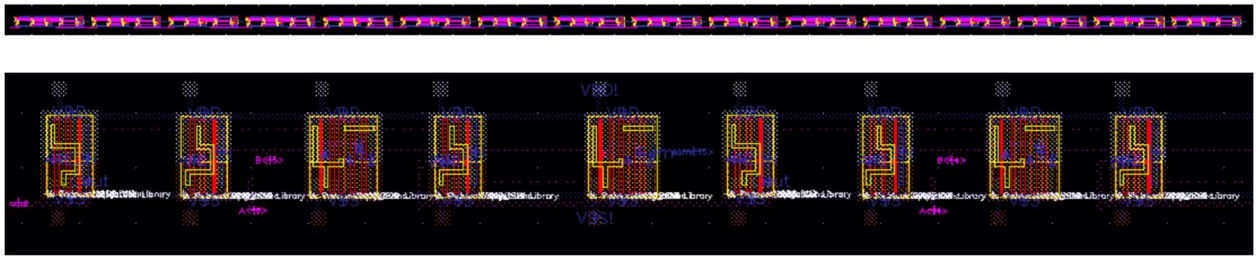
16-Bit Register



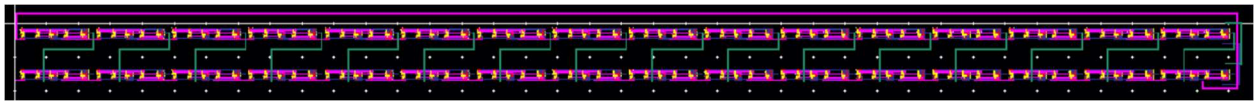
Linear Feedback Shift Register

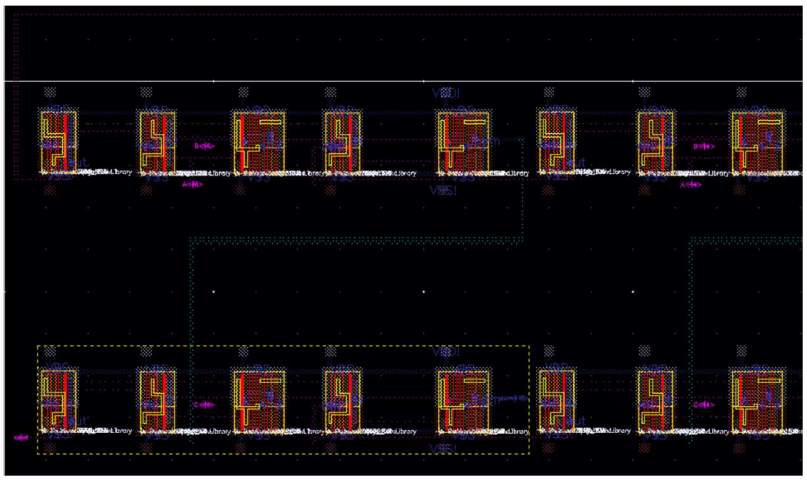


2-Bit Full Adder

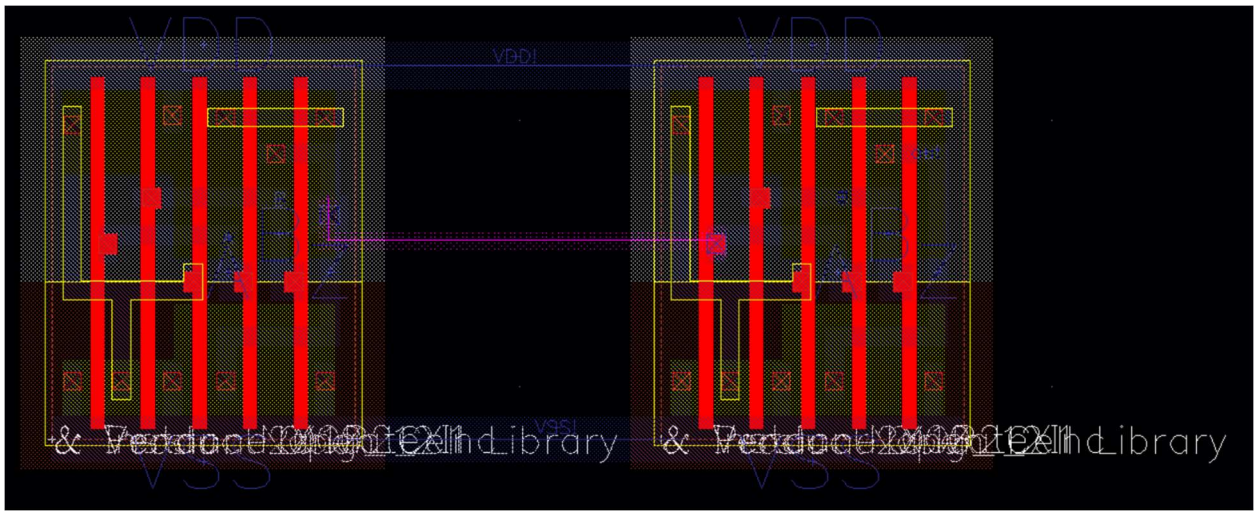


2 Input 16-Bit Adder

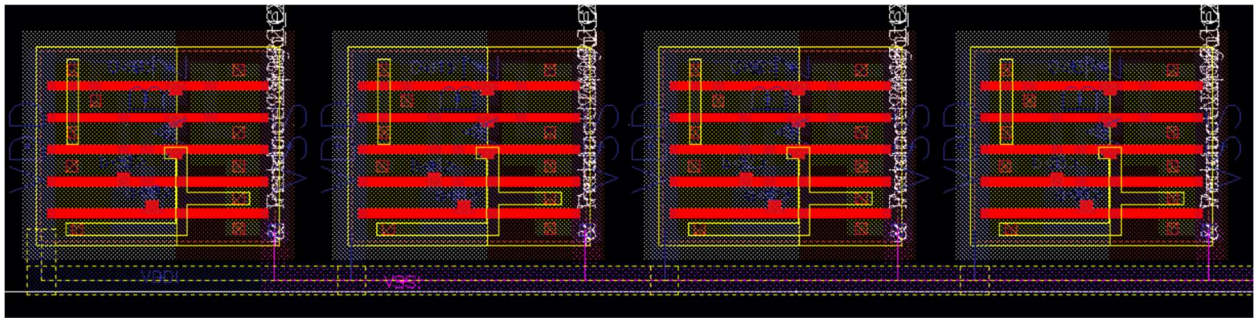


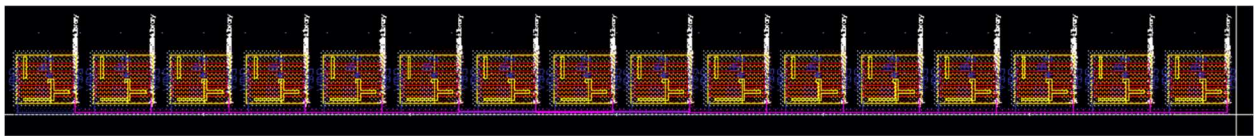


3 Input 16-Bit Adder

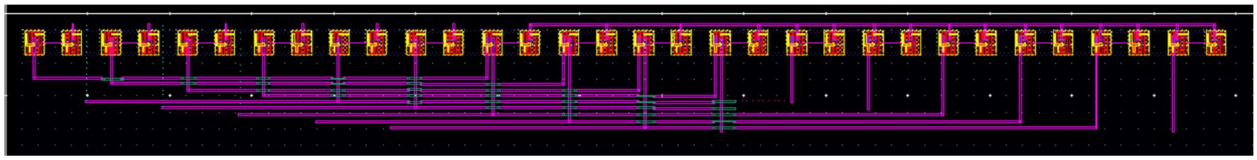


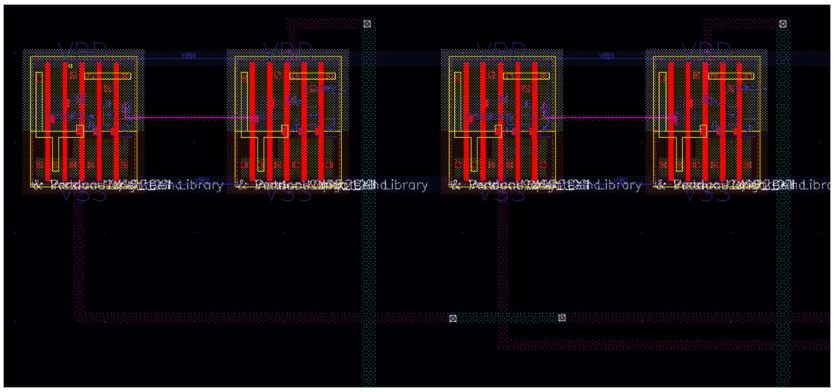
3 Input XOR



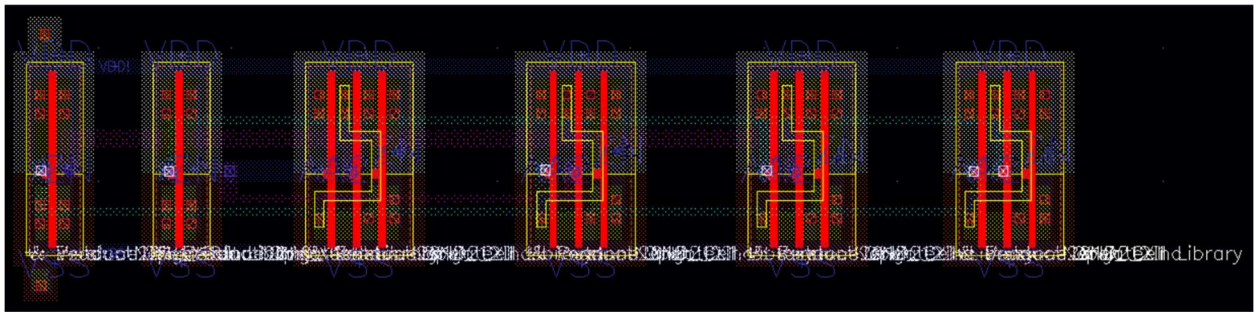


16-Bit XOR

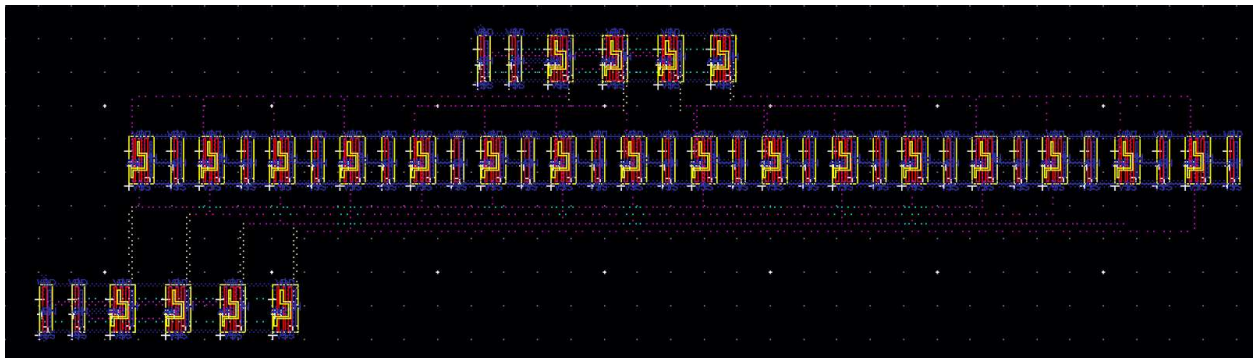




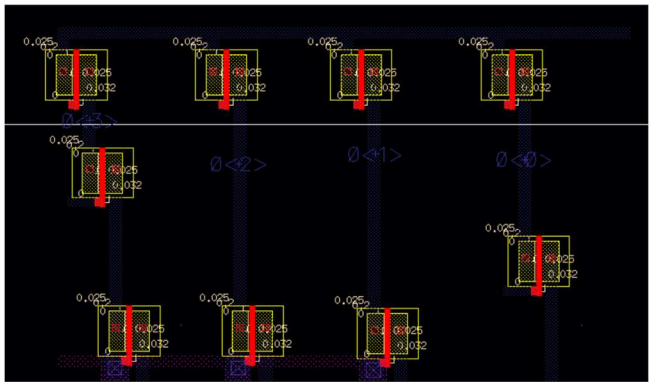
Linear Transform Block



2x4 Decoder

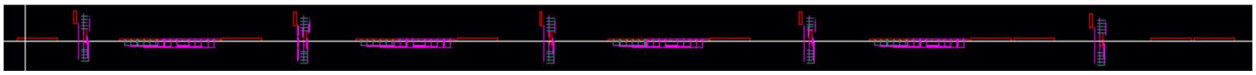


4x16 Decoder

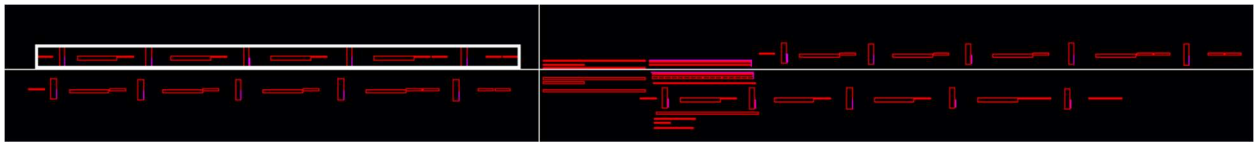


S-Block Lookup Table

S Block



Block Cipher



Encryption Block