Design and Simulation of a Hardware Logic Encryption Block for Sequential Circuits

By

Dina AlSaid , Fatema W. Saarah Hossain and Yasmeen Zayed

Supervised by

Eng. Hazem Marar

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE
in

ELECTRICAL ENGINEERING DEPARTMENT

at

PRINCESS SUMAYA UNIVERSITY FOR TECHNOLOGY

Amman, Jordan

2018/2019

This is to certify that I have examined

This copy of an engineering documentation by

Dina AlSaid, Fatema W. Saarah Hossain and Yasmeen Zayed

And have found that it is complete and satisfactory in all respect,
And that any and all revisions required by the final Examining Committee have been made

---

Eng. Hazem Marar

Acknowledgments

We would like to express our gratitude to every person who helped and supported us through the process of making this project. Great appreciation to our supervisor Eng. Hazem Marar for his guidance and patience. Special thanks to Dr. Hani Ahmad for passing on his knowledge and experience. We also would like to express our appreciation for Prof. Mohammad Mismar and Dr. Mohammad Abu-Khater for their time. Lastly, Many thanks to our friends and colleagues for their helping hands and countless gestures of kindness.

Abstract

In this project, a hardware-based encryption block is designed, that can be widely used in modern communication systems. The block is meant to be added between the components of an already existing architecture with minimal additional cost. The keys will be based on a symmetric encryption method.

The achieved design is two blocks; one for encryption and another for decryption. Both blocks are smaller than $500\mu m^2$ each and can receive and process serial data at up to 1Gb/s according to pseudo-LVDS standards. The power consumption for each block is less than 15mW. Each block requires two clocks with the same frequency but different skew, and a third clock that must be adjusted according to the type of key used.

The design is simulated on Synopsys Custom Designer based on 28/32 nm CMOS technology design rules.

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

1.1 Objectives

In this project a hardware-based encryption block is introduced. The block is designed to be added between the components of an already existing architecture, and is based on symmetric encryption.

The block design consists of an encryption circuit and a differential transmitter and receiver.

The simulation and design will be performed on Synopsys Custom Designer, using 28/32 nm CMOS technology design rules. The design will target 28/32 nm microprocessors, microcontrollers and Silicon -on-chip SOI, hence, no interfacing circuits are needed.

1.2 Cryptography and Encryption

1.2.1 Definition of Cryptography

Cryptography is a process at where data is changed to another form of text that is no longer understandable upon transmission, where the produced text is called the ciphered text. Recovering the ciphered text back to its original form is called the decryption process. A predefined key is set and used throughout the encryption and decryption processes. Keys can be hardwired [1] or a set of randomly generated keys during runtime.

1.2.2 Cryptography Features

The cryptography process has the following features that ensure confidentiality and security.

- Secrecy: The data can be decrypted only if the receiver has the encryption key and therefore data access is limited to sender and receiver.
- Authentication: The identity of the sender and the receiver will be verified.
- Integrity: During the process of encryption, the data will not be changed and will reach the receiver as it was sent.
- Availability and reliability.

1.2.3 Encryption Methods

Encryption methods [2] can be categorized based on the encryption key to three types:

- Hashing: A stream of characters or numbers is generated with fixed length from a text; the new string is called hash. What makes this type different from other encryption types is that the hash is impossible to return to the original text (depending on how well the hashing is done). This method is used to make sure that the data has not been changed after sending because any variation in the input text will result in a completely different hash.

- Symmetric: In this method, the sender and the receiver have a shared key. The confidentiality depends on keeping this key hidden from other parties. The symmetrical aspect is that the encryption and the decryption processes depend on one key therefore they are symmetrical and decryption process is the same as or the inverse of the encryption process. This type include two more divisions: block cipher and stream cipher.
- Asymmetric: this method uses two keys. These two keys are not identical but are often paired together. One of the keys can be shared which is the public key and is used in the encryption process and the other key is kept hidden and is used for decryption. This type of encryption will introduce more complications in retrieving the data but will add to the security.

These methods can be implemented in both software algorithms and hardware circuitry, and many guidelines and standards are being developed to ensure maximum security and minimum vulnerability to attacks.

1.2.4 Hardware Encryption vs. Software Encryption

Software encryption is more commonly used than hardware encryption since it is easier to use, costs less, easier to upgrade, and is more common than hardware solutions. On the other hand, the security of the data is limited to the security of the operating system, not to mention that the whole encryption process can be enabled and disabled by the user, which can introduce threats. Another drawback is that software encryption will have low performance since it uses the system's CPU in order to perform the encryption.

On the contrary, hardware encryption does not have low performance, because it does not need to access the CPU or memory of the system, this adds to the security of the user's data. Hardware encryption does not require user interference, does not need additional software that increases the vulnerability to malicious software, or does it require drivers.

However, the disadvantages of hardware encryption are high cost. Upgrading the system means that there is a need to replace the whole device and one solution may work perfectly on one device, but it may not be compatible for others, or even sometimes may not be compatible for all the components throughout the system.

1.3 Differential Signaling

1.3.1 Single Ended Signals vs. Differential Signals

Single ended signals are signals that are referred to the circuit's common ground. On the other hand, differential signals are floating with respect to the common ground and are measured as a voltage difference between two signals. Figure 1-1 shows an example of a single ended signal and a differential signal that hold the same information.

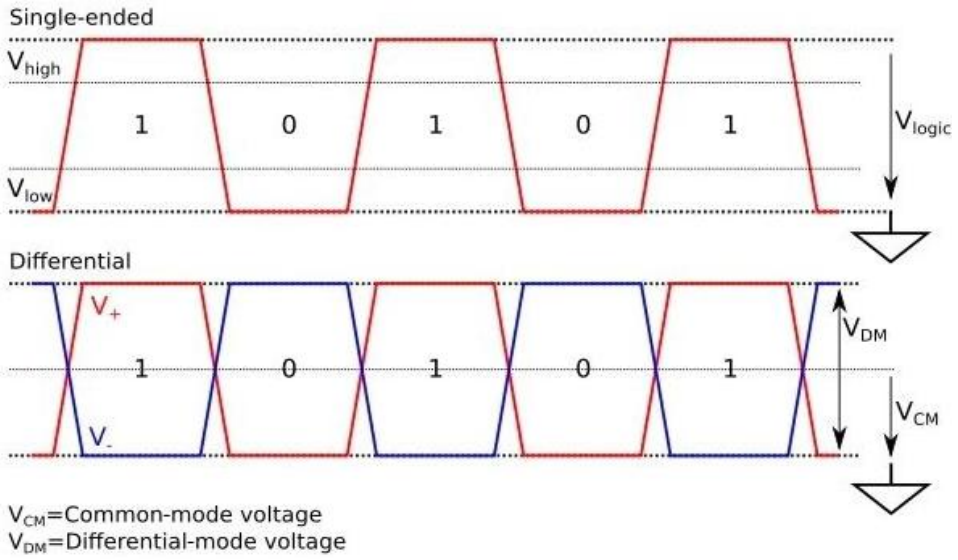


Figure 1-1 Single Ended Signal and Differential Signal

Single ended signals need one connection (wire) for every signal and a connection for the ground, while the differential signals need two connections for each signal and an extra connection for the common mode signal, making the differential transmission more expensive and harder to install.

One of the most important advantages of the differential signals is that the signal is susceptible to noise when electromagnetic or radio frequency interference are present, which is advised for the transmission of low power small signals. The noise is omitted since the two wires that transmit the signal are exposed to the same environment noise, and the fact that the information is obtained by taking the difference between V+ and V- in Figure 1-1.

1.3.2 Low Voltage Differential Signaling (LVDS)

LVDS is a standard used for differential data transmission. The LVDS supports transmission of low power differential signals and high speed (hundreds of MHz) [3].

Some of the advantages of the LVDS standard are:

- The need for small differential swing therefore reducing the power consumption.
- Low susceptibility to surrounding and self generated noise.

Table 1-1 shows some of the electrical characteristics of the LVDS [4].

Table 1-1 LVDS Standard Electrical Characteristics.

	Range
Power supply	2.5 - 5 V
Voltage swing	250 - 450 mV
Common mode voltage	1.125 - 1.375 V
Data rate	100M-1.923G bps
Clock edge rate	260 -1500 ps
Termination impedance	90 - 130±10% Ω

The data rate limit of an LVDS- based design is determined by the generator transition time characteristics, the media characteristics, the distance between the generator and the load and the required signal quality.

The importance of choosing suitable edge rates, is to keep spiking of the output signal within acceptable range (±20% $V_{steady\text{-}state}$) and reduce switching losses.

The termination impedance will prevent any reflections due to imperfect impedance matching between the transmission line (50 ohms) and the differential load termination being propagated back down the line.

1.3.3 Pseudo-LVDS

A derivation of the LVDS is the pseudo-LVDS. Pseudo-LVDS was derived to suit other applications where the characteristics of the LVDS need small changes to fit the application specifications, usually to increase the speed compared to LVDS [5].

Impedance matching will ensure the minimization of reflections and maximization of power transfer from the transmitter through the transmission lines and to the receiver, therefore not losing data and reducing errors.

1.4 Design Requirements

- 32/28nm standard CMOS technology shall be used.
- The supply voltage shall be 1V to conform to 32/28nm standard CMOS technology.
- Propagation delay of the encryption block shall not be more than 10ns.
- The designed encryption shall operate at 500MHz input signal.
- The design should be able to produce corrupted outputs if illegal keys were introduced.
- Synopsys software is used to simulate the encryption block.

1.5 Design Constraints

- Economic: If manufactured, the encryption block should cost less than $10, given that the fabrication cost for 32/28nm CMOS is between 10,000-12,000$/mm^2.
- Environmental: The encryption block should properly operate at the commercial temperature range standard, which is 0-70C.
- Manufacturability and sustainability: the encryption block circuit should be able to operate on $V_{threshold}$ with a variation of ±5%.

1.6 Document Organization

Chapter 1 consists of an introduction to cryptography and encryption, differential signaling and LVDS and Pseudo-LVDS standards. Moreover, it lists the design requirements and constraints.

Chapter 2 contains a brief review of published papers about ways to generate keys, AES standard and designs of LVDS and Pseudo-LVDS drivers and receivers.

Chapter 3 discusses the design requirements and constraints in more detail in addition to the conceptual (type of key, blocks used for encryption and data size) and physical (components needed for each block and their schematics) approaches and options for the system design, including the advantages and disadvantages of each option. Then, the options incorporated in the developed design and their schematics, stick diagrams and layouts are discussed in the developed design section.

Chapter 4 includes the simulation setup and the simulation results of the developed design as well as analysis of results and discussions about whether the design requirements are met.

Chapter 5 concludes important points in the design and any suggestions for future improvements on the design.

1.7 Project Management

1.7.1 Tasks

Table 1-2 shows how much was contributed by each member to each task; the darker the shading under a name, the greater the contribution. (A legend is provided beneath the table.)

Table 1-2 Task Division

		Dina	Fatima	Yasmeen
Research				
Documentation	Introduction			
	Literature review			
	Design			
	Analysis of results			
	Conclusions and further work			
	Final edits and revisions			
Design and Implementation	System design and planning			
	Schematics and calculations			
	XOR stage			
	Multiplexer stage			
	Flipflop stage			
	Transmitter stage			
	Receiver stage			
	System layout			
	Test benches setup			
	Simulation and testing			

Legend	
	Primary contributor (in charge)
	Secondary contributor (significant contribution)
	Contributor (occasional assistance, feedback, etc.)

2 Literature Review

The design will consist of multiple segments, each of which will serve a certain purpose. The first segment is going to be a receiver that will receive a low voltage differential signal and convert it to a single ended signal to be able to perform logic operation on it. The next segment is the encryption/decryption process that will process the data. The last segment is a transmitter that will convert the single ended data back to differential data and send it through a transmission line.

2.1 Key Generation

In [6], the cryptography process consists of three stages: key generation, encryption, and decryption.

For the first step of the key generation stage, shown in Figure 2-1, an 8-bit key is the input for eight 2:1 multiplexers along with another key through the feedback. This will make sure that the key is not used two times in a row and will increase the randomness of the generated new key.

Next the selected bits are fed into two 4-bit shift registers. the shift registers will delay and pass the bits to the next stage. The delay will be important to control the output (new key) at a desired time, and to keep track of the process.

The 8-bit output will be the select lines for eight 8:1 multiplexers, and will be chosen in circular order, as shown in Figure 2-1, with the current key as an input. Then the multiplexers' output will be passed to two 4-bit registers and the new key is ready to be used in the next stage.

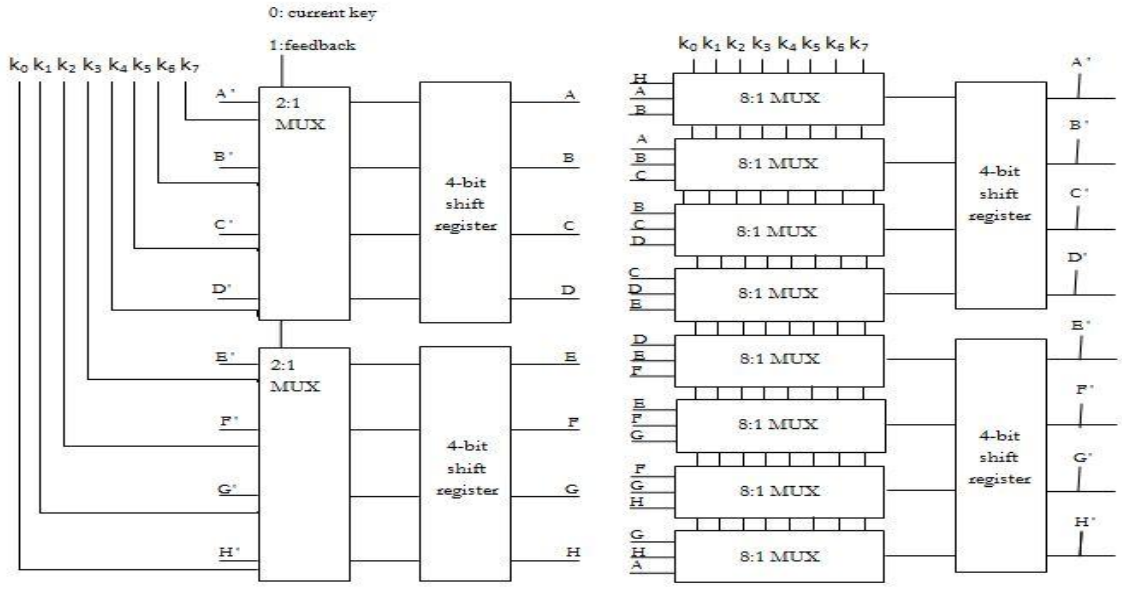


Figure 2-1 Key Generation

The second stage is the encryption stage. The new key will be XOR-ed with the data and the output (cipher text) is taken through a shift register. Then the used key will be fed-back to generate another key.

The final stage is the decryption, where the cipher text is XOR-ed with the same key the original data was encrypted with, to retrieve the original data.

The drawbacks of this approach [6] are:

- The length of the key is fixed and that will complicate the encryption of data longer than the key and will reduce the possible combinations of the key, therefore limit the randomness.
- Key has to be known by both ends of the process and that increases the possibility of it being hacked.
- The 74xx used are TTL-based [7] and therefore need an interfacing circuit that will also add other limitations.
- The key generation circuit takes up most of the size of the system, while the encryption/ decryption is made simple.
- The system has a large time delay compared to desired delay. For example the 4-bit registers alone will have at best 4.5 ns [7], and this design [6] will have three stages with shift registers and that makes the delay of these stages alone 13.5 ns.
- The encryption and the decryption circuitries should be synchronized, to make sure that the data is encrypted and decrypted by the same key at both ends.

2.2 Encryption and Decryption Method

An alternative approach is presented in [8], which features an FPGA implementation of a system based on the Advanced Encryption Standard (AES) algorithm. The standard has four basic transformations (processes). as shown in Figure 2-2.

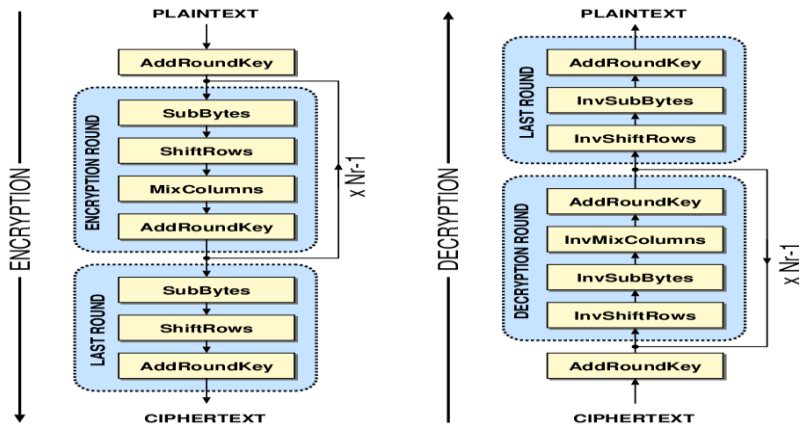


Figure 2-2 AES structure [9]

The AES is a symmetric block cipher, which has a variable key of 128, 192 or 256 bits, and fixed data length of 128 bits. The input data will be in a 4x4 matrix, called State, each element of which contains a byte.
The AES algorithm utilizes four basic transformations for encryption as follows:

- **Sub-Bytes Transformation**
  This phase involves passing each byte of the State through a pre-defined table or substitution Box (S-Box), which is a two-dimensional lookup table. Each element of the state matrix is essentially treated as the address of the S-Box element it is to be replaced by.
- **Shift-Rows Transformation**
  In this phase, a shift left and rotate operation is applied to each row of the State. The amount of shift in each row is the index of the row starting from zero (i.e. first row is not shifted, row two is shifted by one element, etc.)
- **Mix-Columns Transformation**
  The Mix-Columns phase provides diffusion by mixing the input around and performs operations splitting the matrix by columns. Each cell in each column will be multiplied by a different polynomial, so eventually each cell of the column will be mixed over all 4 cells of that column and so on.
  The design in [8] uses the mathematical properties of multiplication by a fixed constant in GF (2^8) using combinational logic circuit so this step maps one column of the input state to a new column state.
- **Add-RoundKey Transformation**
  This phase of encryption is directly operated with the AES round key: the input to the round is XOR-ed with the RoundKey. A different Round Key is generated for each round by applying transformations to the previous Round Key and the original key. The process is shown in Figure 2-3.

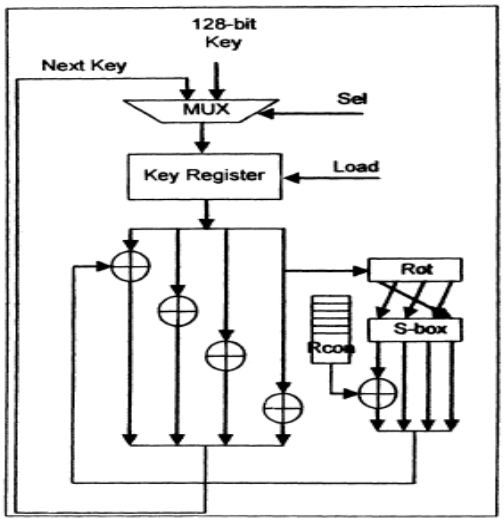


Figure 2-3 Generation of AES round key

As for the decryption, the process is reversed as shown in Figure 2-2.
Although the algorithm is efficient and secure, it is very complex and this complexity essentially adds more components to the system increasing the block's size.
Furthermore, the process's latency is high, since it needs to be done for several times (as rounds).
Moreover, the process is done using FPGAs, which implies that the hardware is relatively large, and its capabilities in terms of the operating frequency (231.97 MHz) [8] is lower than the anticipated operating frequency (500 MHz).

2.3 LVDS Interface

[10] presents low-power designs for an LVDS receiver and transmitter using 28nm CMOS technology which are meant to interface between an AM (Associative Memory chip) and an FPGA. The AM operates on a 1V supply while the FPGA can function with 1.8V or 2.5V (1.8V was chosen here to prioritize conservation of energy).

The receiver would convert the differential signals from the FPGA to a single-ended signal for the AM, and the transmitter would convert the single-ended signal from the AM (to the FPGA) to a pair of differential signals that comply with the ANSI/TIA/EIA-644 [4] specifications for LVDS data transmission format. The results from simulating the transmitter are then compared to transmitters from [11], [12], [13]and [14],which use different process technology, but follow the LVDS standard.

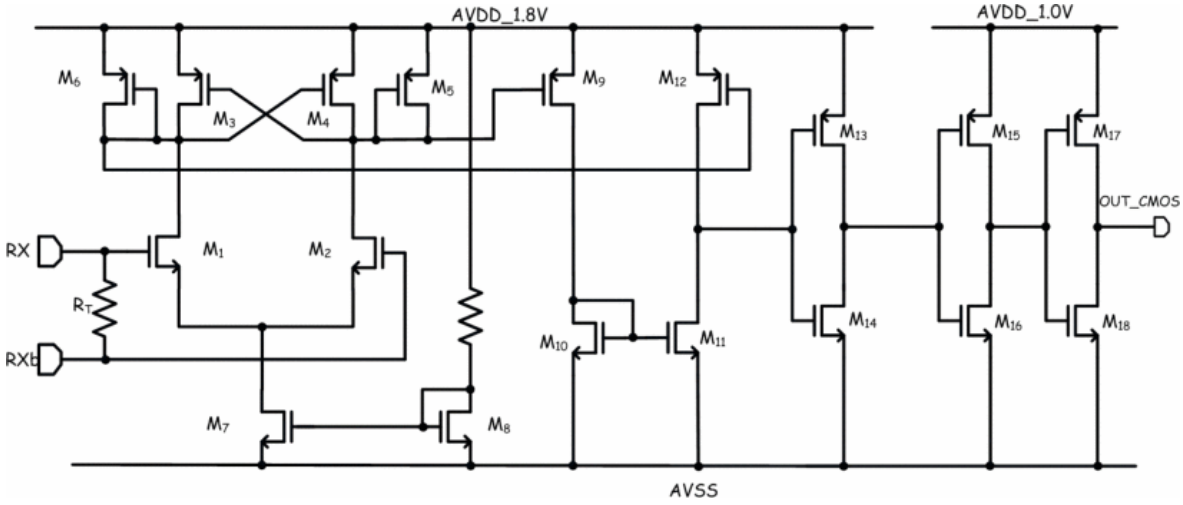
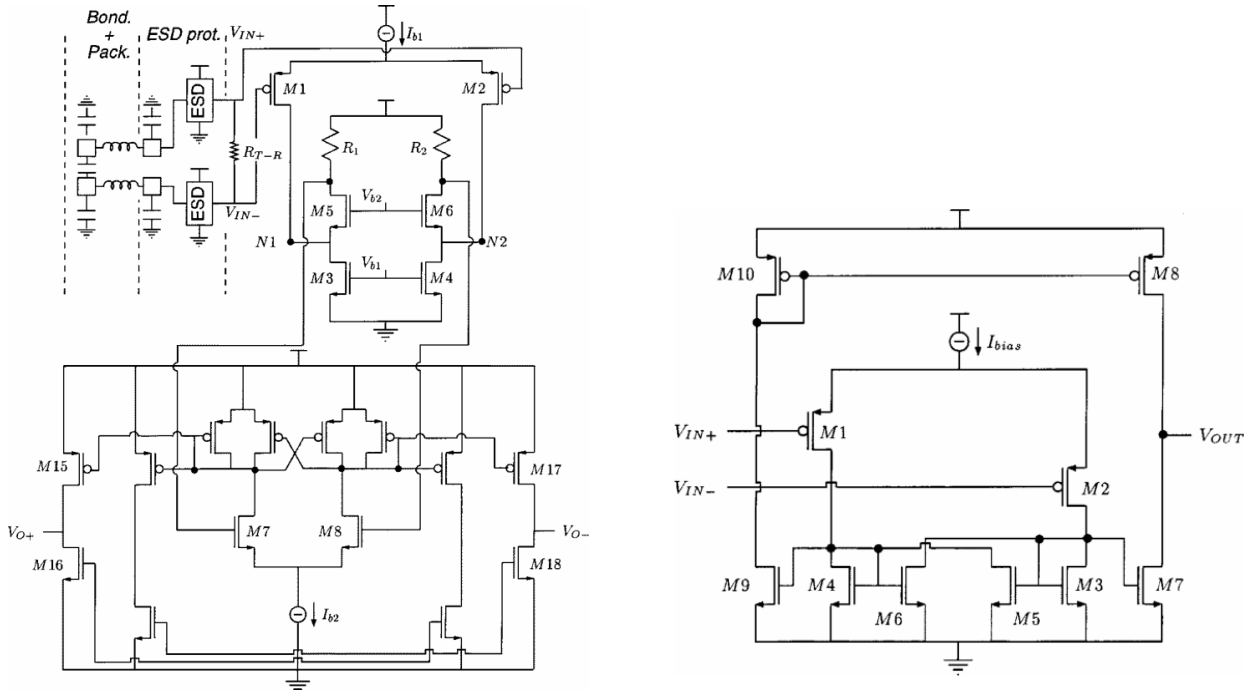


Figure 2-4 LVDS Receiver in [10]

As displayed in Figure 2-4, the receiver consists of two stages. The first stage is an amplifier with NMOS gates receiving differential input; the load is cross-coupled to enhance speed and increase gain due to the resulting positive feedback. The following stage is a level shifter, which involves three cascaded inverters, the first of which is in the 1.8V voltage-domain, while the last two are in the 1V domain. The receiver has a 900μm^2 area and the termination resistance R_T is included in the layout.



(a) (b)

Figure 2-5 LVDS Receiver Implementation in [11]

[11] implements a folded-cascode input configuration as shown in Figure 2-5(a) above; the design prioritizes maximization of gain-bandwidth product over the full input range of the LVDS standard. The cascading greatly reduces the small-signal resistance at nodes N1 and N2 and allows a large bandwidth when followed by a second stage similar to Figure 2-5 (b).The resulting receiver has an are of 0.08mm^2 on 350nm process and consumes 33mW of power.

While the area and power consumption are greater than in [10], this design is meant to handle various ranges of input rather than for a specific interface.

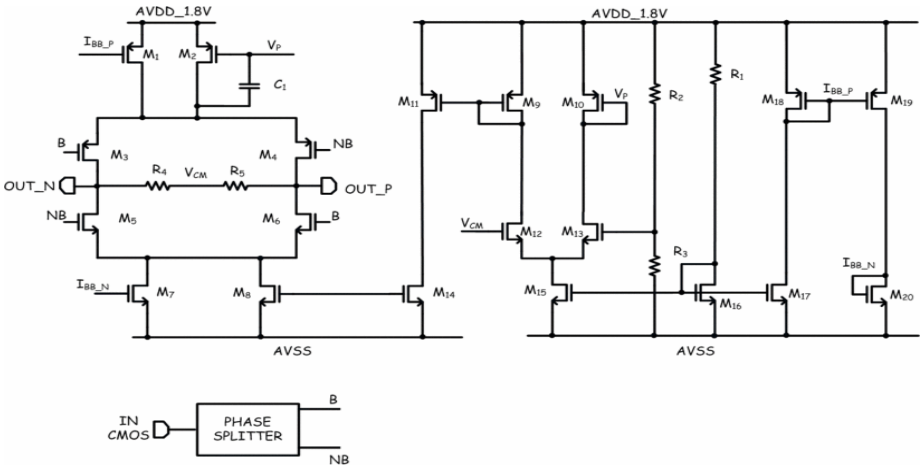


Figure 2-6 LVDS Transmitter in [10]

Figure 2-6 shows the transmitter used in [10]. First, a phase splitter is used to level-shift the $1V_{p-p}$ single-ended signal, as well as produce two signals with opposing polarities which control transistors M_3 – M_5, which form a Bridged-Switched Current Sources (BSCS) architecture to produce differential outputs. The right side of the schematic displays the feedback loop used to control M_1, M_2, M_7 and M_8 to maintain the V_{CM} according to changes sensed between R_2 and R_3. The transmitter has a $0.009mm^2$ area and typically consumes 8.7mW of power.

A similar structure is used in [10], but with a simpler control circuit to maintain common mode, as can be seen in Figure 2-7. A differential amplifier is still used to compare the achieved common mode with the required, but there is only one current source and one current sink, and only the current source is manipulated by the control circuit.

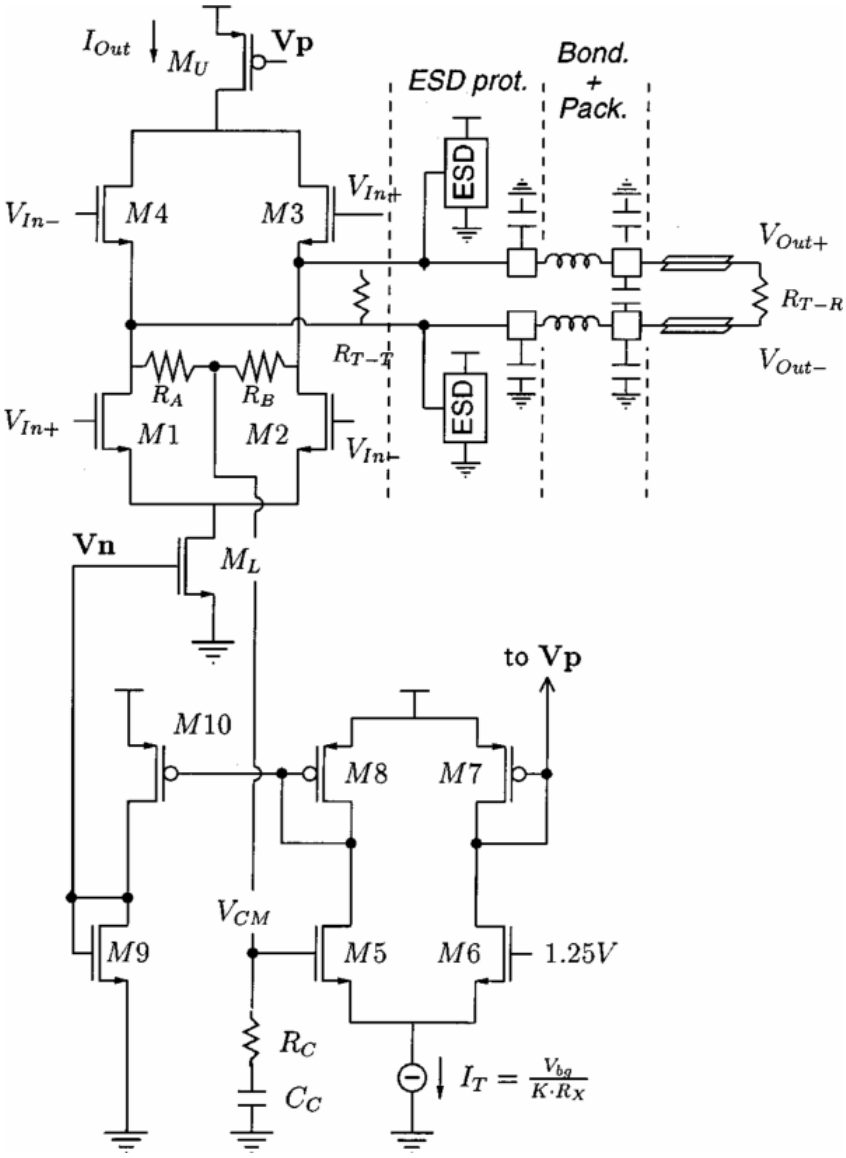


Figure 2-7 LVDS Transmitter in [11]

Table 2-1 Comparison Table Compiled in [10]

Article	Data Rate [Gbit/s]	Power [mW]	FOM [mW/Gbit/s]	Area [mm^2]	Voltage Supply [V]	CMOS Technology
[10]	1.0	8.7	8.7	0.0009	1.8/1.0	28 nm
[11]	1.2	43.0	35.8	0.175	3.3	350 nm
[12]	2.0	13.2	6.6	-	3.3	180 nm
[13]	1.4	23.0	16.4	0.11	1.8	350 nm
[14]	1.2	67.5	56.3	56.4	1.5	130 nm

While the transmitter design's 1Gbit/s data rate is slightly lower in [10]than in [11], [12], [13]and [14] the 34%~87% decrease in power consumption compensates for it more than adequately. However, the area taken up by [10] is so much smaller mainly because of the CMOS technology used. [12] did not provide the area required, so fair comparison cannot be made even though it has the next-best power conservation and can operate at the fastest data rate. [13] prioritized the optimization of both power and frequency.

3 Design

3.1 Analysis of Design Requirements

- **32/28nm CMOS technology shall be used with a standard 1V supply voltage.**

The 32nm technology node is a relatively new semiconductor manufacturing process for which the name refers to the gate length. It was first used commercially for IC manufacturing in 2010, superseding the 45nm technology from 2007. Although chips can be fabricated using technology as small as the 5nm process, cost of fabrication is very high for smaller processes. Therefore, many architectures are based on the 32nm standard, including Intel's Saltwell, Sandy Bridge (Core i3, Core i5, Core i7 processors) and Westmere architectures [15]. We will be following the design rules provided by the physical design kit for 28/32nm technology in Synopsys.

- **Propagation delay and operating frequency**

Propagation delay is the time needed for the output to change from a high state to a low state or vise versa. The block will be designed to have an overall propagation delay less than 10ns. Having in mind that the stages that operate on 500MHz should not have a propagation delay more than 2ns to operate correctly.

- **The designed encryption shall operate at 500MHz input signal**.

This means that the input stage should be able to handle a frequency of 500 MHz without any problems, this requires the propagation delay of the input not to exceed the clock's ON time. The block will be further tested to find out the maximum operating frequency.

- **The design should be able to produce corrupted outputs if illegal keys were introduced.**

If the key at the decryption side does not match the key at the encryption side, the decrypted data will be different from the encrypted data.

3.2 Analysis of Design Constraints

- **Economic:** The encryption block should cost less than $10.

  Estimating the cost for 32/28 nm CMOS fabrication to be $9,700-$10,200/mm^2 [16] for commercial purposes and around $5000 for educational purposes, the conservative allowance for the maximum area of the encryption block would be up to 2000μm^2 for using the $5000 price. This value was obtained by using Equation 3-1.

$$Area\ Area\ Allowance = \frac{Cost\ Allowance}{Cost\ per\ Area} \left[= \frac{\$10}{\$5000/mm^2}\right]$$

Equation 3-1 Area Estimation from Cost

The area of a chip must be an integer multiple of a minimum die area .Since our design is meant to be included into larger ICs, the above calculations only give an estimate of how much this system would contribute to the total fabrication cost. This approximation assumes that the IC this design is incorporated into utilizes at least 95% of the die.

Limiting the area restricts the complexity of the design, especially as the interface to convert differential signals to single-ended and then back again is included.

- **Environmental:** The encryption block should properly operate at temperature 0-70°C.

  This is a standard commercial operating range and will likely be a requirement of all components on any IC the encryption block is inserted into, as most devices are expected to undergo exposure to such temperatures.

- **Manufacturability and Sustainability:** The circuit should be able to withstand up to ±5% variations in $V_{threshold.}$.

  Foundries do not fabricate transistors with constant threshold voltage, so allowances must be made for small unintended variations in oxide thickness and RDF (random dopant fluctuation). Temperature and back-gate effect also affect the threshold voltage.

3.3 Different Designs Approaches

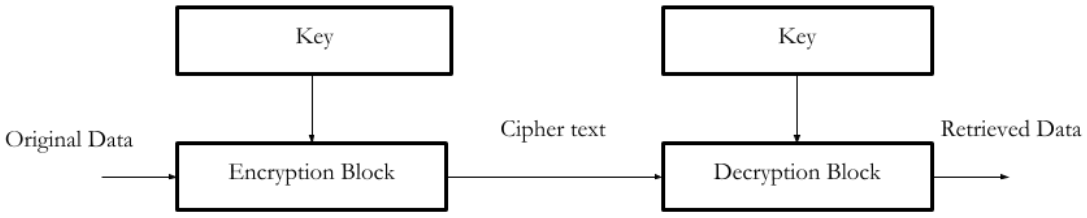


Figure 3-1 Cryptography Process Block Diagram

Figure 3-1 depicts the overall workings of the design. The data retrieved should be identical to the original data if the key entered into the decryption block is the same as the key entered into the encryption block. With incorrect keys, the decryption block should produce meaningless output.

The main elements to consider for this design are:

- The size of the data to be encrypted/decrypted (how many bits).
- The nature of the key (random or fixed).
- The number and complexity of stages used to encrypt.

3.3.1 System design approaches

3.3.1.1 The Key

The key can either be fixed or variable. Having a variable key can surely increase the security of the encryption, however a new key needs to be generated at each encryption cycle, this means that the decryption process has to produce the same key for the same encrypted sets of data. Moreover, a variable key needs a generation circuit that surely takes up a significant amount of the design's area.

On the other hand having a fixed key will reduce the complexity of the design but at the cost of reducing the security.

Another option is to use more than one key in an encryption cycle in order to increase the speed of the process, to expand the length of the data encrypted and to increase the security.

Lastly, the key length could match the data length, making it easier to perform the encryption process, otherwise, having a shorter key would require more than one cycle to finish the encryption, therefore increasing the delay, but allowing a more compact circuit.

The key can either be stored in registers within the encryption/decryption blocks or in an external EEPROM connected to both blocks. While using the EEPROM would add flexibility, it would also add significantly to the area and reduce security.

3.3.1.2 The Encryption Stages

The first approach was to follow AES. This methods advantages is that it has high security and it is widely used. On the other hand the process is complex and has many stages that need a large area to implement, not to mention the iterations needed to complete the encryption.

The second approach was to use parts of the AES, specifically the sub-byte, shift-rows and add-round-key transformations. The sub byte transformation requires a pre-defined substitution table that is made up of memory cells, therefore a large area should be provided. Therefore another approach is presented, which is only using the add-round-key and the shift-rows transformations.

For the last approach these two transformations are made simpler, by using an XOR gate instead of the add-round-key transformation, and an operation of shift and rotate instead of the shift-rows transformation. These modifications definitely reduce the security, but enables a more simple design that satisfies the design requirements.

3.3.2 Physical design approaches

3.3.2.1 Differential interface

Table 3-1 shows a comparison between two options for the differential interface. [17]

Table 3-1 A Comparison between LVDS and CML

	LVDS	**CML**
Differential Output Voltage	250-450 mV	450-800 mV
Offset Output Voltage	1.125- 1.375 V	2.425- 2.875 V
Output Rise/Fall Time	260-1500 ps	50-270 ps
Differential Input Impedance	90-132	HIGH
Maximum Signaling Rate	1.923 Gbps	1 Gbps
Technology	CMOS	BJT

Clearly the LVDS is more suitable for our requirements but does not quite fulfill them, which leads to the last option which is the pseudo-LVDS.

Pseudo-LVDS will give more flexibility to the characteristics, but impedance matching has to be taken into account to preserve the quality of the transmission.

3.3.2.2 XOR Gate Stage

This stage XORs the parallel data with the parallel key, which changes the data according to Table 3-2. The data stays the same when XOR-ed with 0, and inverted when XOR-ed with 1.

Three options were available, the first shown in Figure 3-2 was to use transmission gates because of the smaller size, but we also need inverters so the inputs are not connected directly to the diffusion causing impedance problems at high frequencies.

The addition of the inverters made us consider the next option, which is a normal pull-up and pull-down network shown in Figure 3-3, which is more robust and it does not have back driving problems.

The third option is to clock the gate so it only works when the parallel data is ready, but this will increase the power consumption and most of the time the output is floating, which does not give us the advantages of the static power consumption of the CMOS gates.

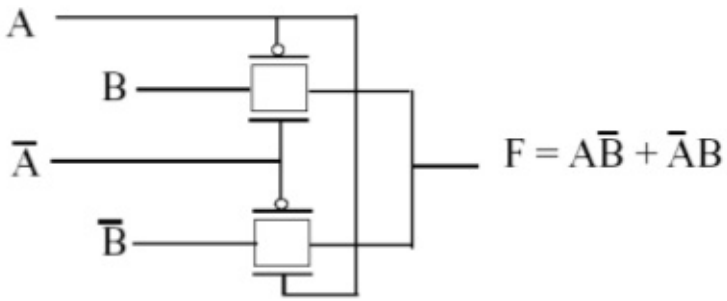


Figure 3-2 XOR Schematic Using Transmission Gates

Table 3-2 XOR Truth Table

A	B	F
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**0**

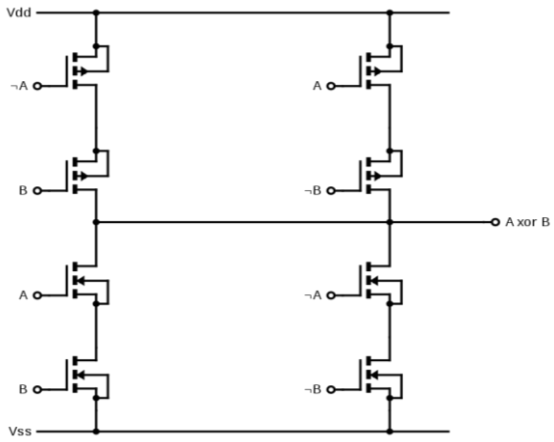


Figure 3-3 CMOS XOR Schematic

3.3.2.3 Clocks

Internally generated clocks are not usually used, in order to avoid glitches that can cause functional and timing problems and can cause unexpected output change in clocked components.

3.3.2.4 Registers

The data will be fed serially into the transmitter, and to convert it into parallel data, registers are needed. Figure 3-4 shows a 4-bit Serial-input Parallel-output (SIPO) register.

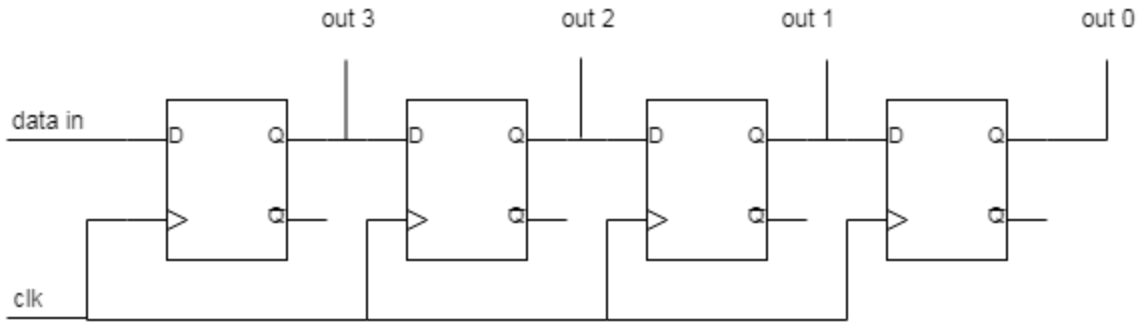


Figure 3-4 SIPO Shift Register Block Diagram

Each shift register input will be connected to a differential to single ended receiver. The length of the SIPO will affect the operating frequency of the following stages to make sure data is shifted properly before loading it to the next stage, for example if the data is 4 bits and the operating frequency is 400 MHz, the data will be ready for loading into the next stage after 4 clock cycles, this means if the next stage is clocked, the clock frequency will be 100 MHz.

These two notes should be taken into account to compromise between the area and the delay.

Before the data is transmitted, it is converted back to serial data using Parallel-input Serial-output (PISO) shift register as shown in Figure 3-5.

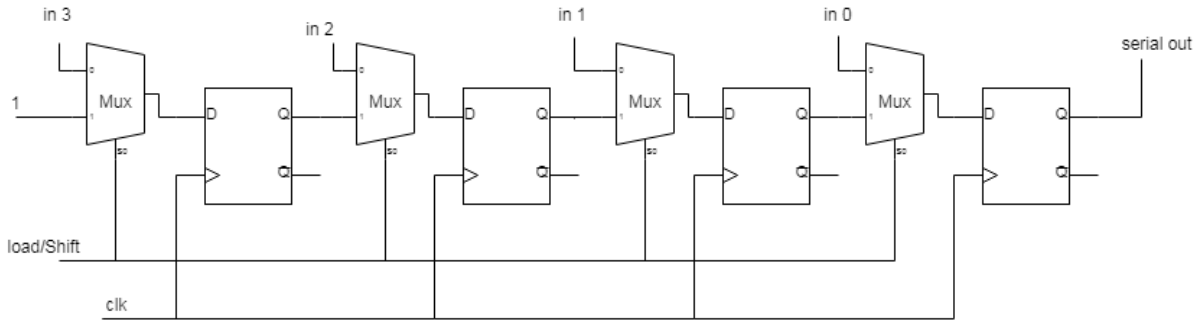


Figure 3-5 PISO Shift Register Block Diagram

The first approach for creating registers is to use latches because of their small size, but that will introduce problems since the data keeps changing while processing, and any glitches/spikes/noise in the input data will be passed to the output. Figure 3-6 shows the schematic of a latch.

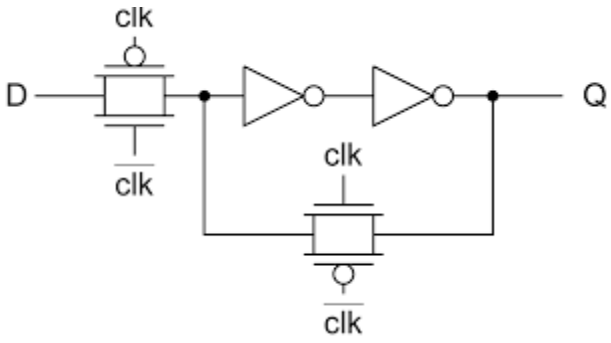


Figure 3-6 Latch Schematic

The next approach is to use two latches with inverted clocks for each register, to create a flip-flop Figure 3-7. This will help pass the data at the falling or rising edges of the clock, by taking samples of the data, which omits most glitches. But the fact that the input is connected to the diffusion of the transmission gate will introduce other problems with the impedance matching at high frequencies, because the resistance of the diffusion will change with the input state(high or low) and with the transistor's type (PMOS or NMOS). Other problems of back driving current to the input may occur too. Figure 3-7 shows the schematic of a flipflop and Table 3-3 summarizes its logical behavior.

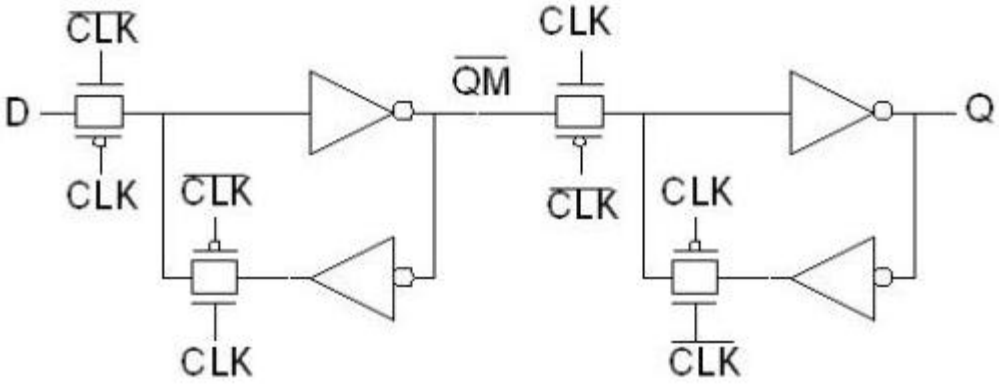


Figure 3-7 D-Flipflop Schematic

Table 3-3 DFF Truth Table

Clock	D	Q	Q_{next}	Q'_{next}
0	×	0	0	1
0	×	1	1	0
1	×	0	0	1
1	×	1	1	0
↑	0	×	0	1
↑	1	×	1	0

Legend:	↑ rising edge	× don't-care

The third approach is to add one inverter to the input or the output to solve the impedance and back driving problems, but that will invert the bits. If the inverter is added at the beginning, the output might not be able to drive multiple gates.

The fourth approach is to add two inverters at the beginning and the end as in Figure 3-8. This will give it the ability to drive more gates and provides some kind of isolation between each flipflop.

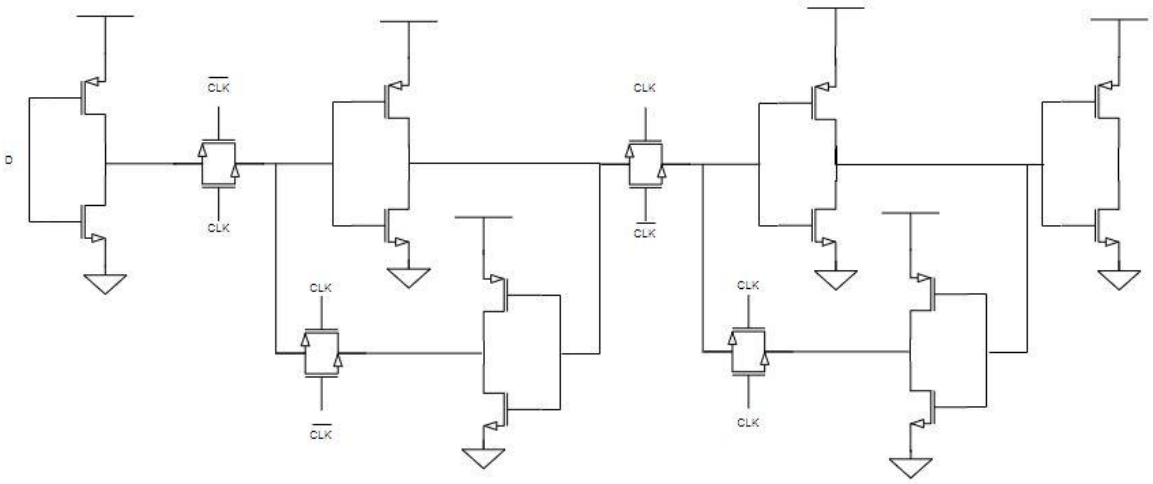


Figure 3-8 Flipflop with Driving Inverters Schematic

For shorter Euler paths in the layout design the clocks are not fed into a transmission gate as in Figure 3-8; instead, they are fed into two gating transistors (Figure 3-9).

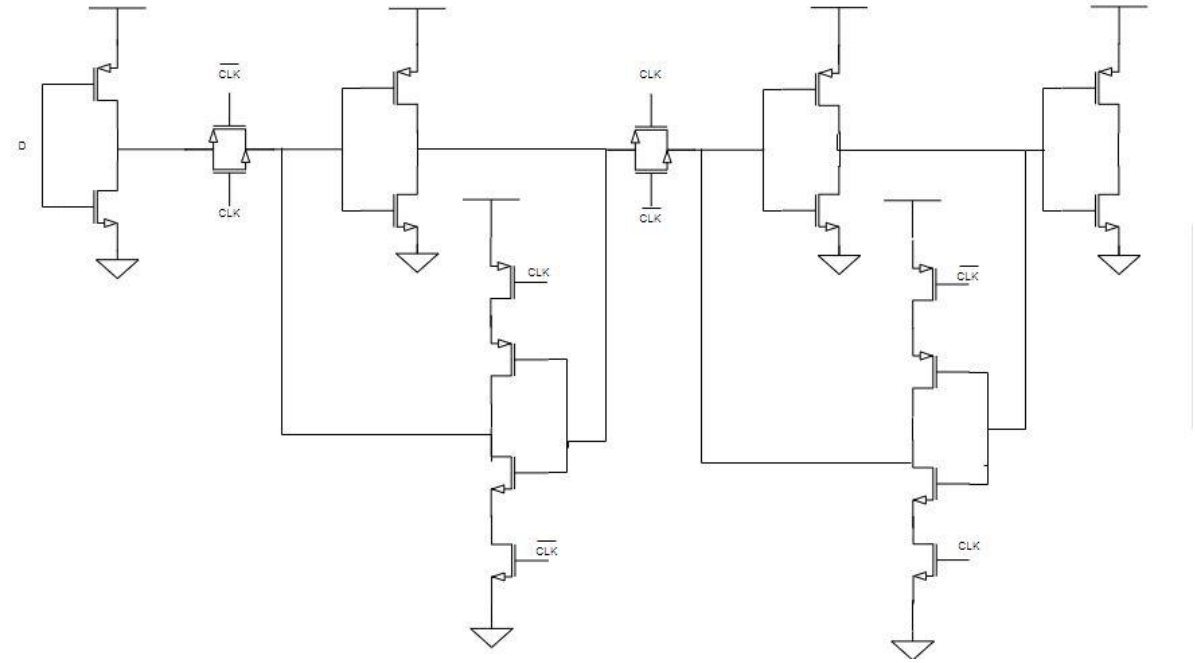


Figure 3-9 D-Flipflop Schematic with Gated Clocks

As for the layout design, the first option was to create a separate diffusion layer for each transistor. This will increase the area needed for the design.

Another option is for the routing: to use the extra polysilicon layers between transistors to be able to use only one metal layer, but the polysilicon layer has high resistivity; hence, this option is disregarded. However, one must take into account the fabrication process used for implementing the design when deciding the number of metals used. Some fabrication processes support only one metal layer; therefore, using the polysilicon layer in this situation is helpful as a routing option.

The other option was to use multiple layers of metals, which helps somewhat with reducing the area and significantly reduces the length of the metal paths.

3.3.2.5 Multiplexer (MUX)

The MUX is used in the PISO shift register to alternate between loading data and shifting data from the previous parallel stage. A clock is connected to the select line of the MUX to choose between shifting and loading as in Figure 3-5. Table 3-4 shows the truth table of the MUX.

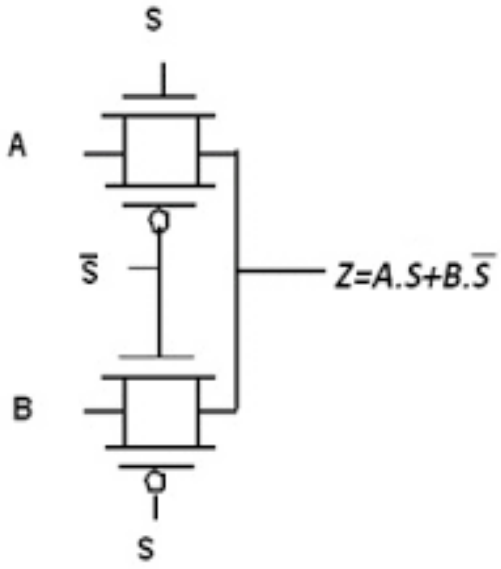


Figure 3-10 Multiplexer Schematic Using Transmission Gates

The first approach was to use transmission gates Figure 3-10 to create the MUX instead of a normal CMOS gate Figure 3-11, because it had fewer transistors. But because of problems like connecting the input to the diffusion and back driving, two inverters were needed which increased the number of transistors, so we reconsidered using a normal CMOS gate because it is more robust.

Table 3-4 MUX Truth Table

S	A	B	Z
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

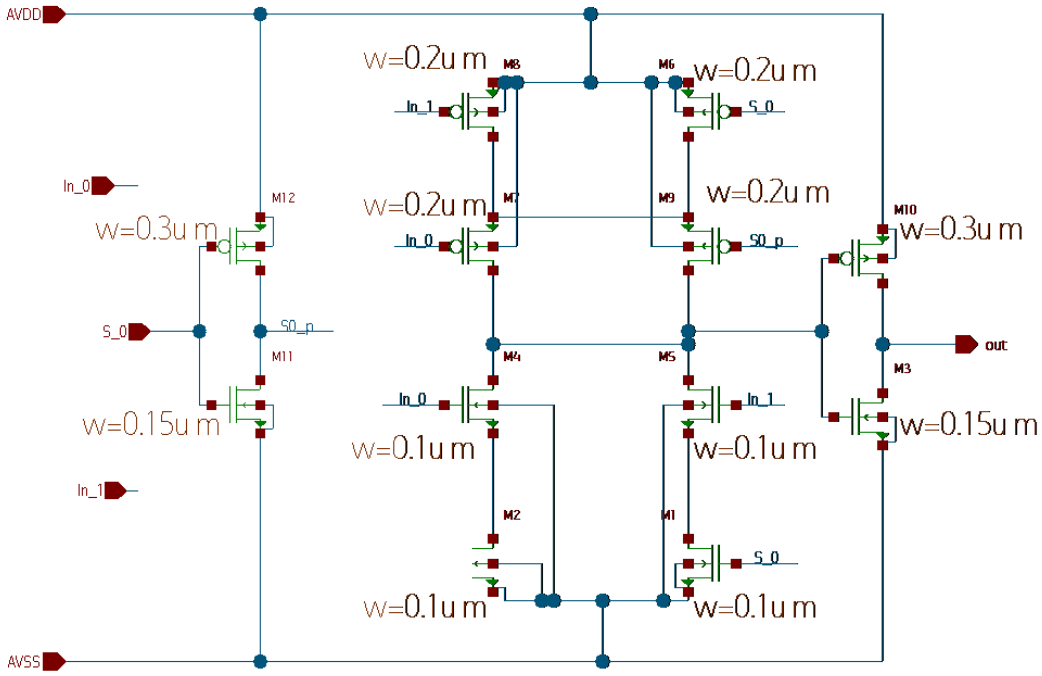


Figure 3-11 CMOS Multiplexer schematic

3.4 Developed Design

The developed design processes serial input data. The serial data is then converted to 8-bit parallel data the converted back to serial before transmission.
The following sections will discuss the components and sub-blocks used for creating the encryption and decryption blocks.

For both blocks, the first component is a Pseudo-LVDS receiver to convert the differential data to a single-ended signal that can be processed by the subsequent stages and the last stage is the Pseudo-LVDS transmitter, which converts the signal back to a differential signal.

Encryption begins by a serial-to-parallel shift register, fed into an XOR with the key bits coming from a Parallel-input Parallel-output register. Then, the data is shifted to the right by 4 bits (hardwired shift) before being fed into a parallel-to-serial shift register, which is the last stage before the transmitter.
Decryption is implemented in a similar manner. Although it is traditional to shift the data first and then XOR with key for decryption, we found it simpler to use the same circuit as encryption, but with a hard-wired 4-bit shift to the right for the keys. The effective result is identical in both cases. Had the shift been any amount other than 4, it would have been necessary to reverse the hardwired shift before the parallel-to-serial shift register.

3.4.1 Pseudo-LVDS driver and receiver

The expected nature of the received data is differential; therefore, a differential receiver is needed to convert the differential signal into a single ended signal.

Pseudo-LVDS was mainly used to choose a new common mode for the differential signal that does not exceed the power supply used for 28nm technology; LVDS common mode is 1.125 while the voltage supply for the 28nm technology is 1V.

Table 3-5 shows the intended electrical characteristics of the receiver used in our encryption and decryption blocks.

3.4.1.1 Receiver

Table 3-5 Pseudo-LVDS Receiver: Desired Electrical Characteristics

Characteristics	**value**
Power supply	0.9-1.1 V
Minimum Differential voltage swing	±150 mV
Common mode voltage and Tolerance	500 ±50 mV
Minimum slew rate	0.9 V/ns

The design in Figure 3-12 was developed based on a H-bridge amplifier architecture which is not traditionally used for generating a single-ended output. Loading between the transistor drains at the bridge has been removed as there is no need to maintain a differential swing.

When differential input voltage [$V_d = V_{inH} - V_{inL}$] is positive, M2 and M3 are conducting and the output must be high, so the small-signal impedance on M2 must be significantly higher than that of M3. When V_d is negative, current flows through M1 and M4 and output must be low so M1 has a significantly larger impedance than M4. M5 and M6 act as current sink and source and are biased by supply and ground respectively.

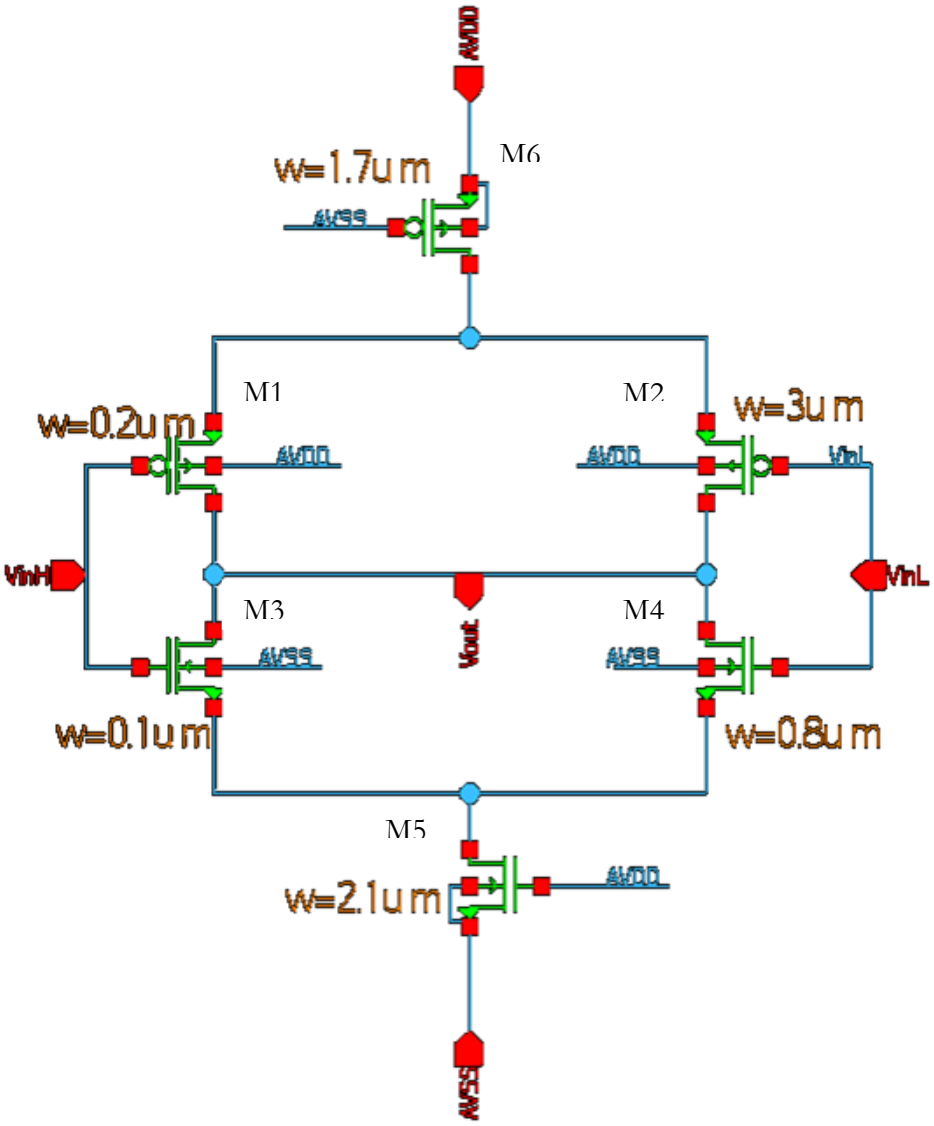


Figure 3-12 Developed Pseudo-LVDS reciever schematic

The transistors were sized with area-optimization in mind. M3 must have the smallest width as it is an NMOS with a large resistance and 100nm is the smallest gate-width allowed by the design rules. M2, M4, M5 and M6 were sized to have the maximum collective widths possible to fit well on the layout next to the remaining encryption circuit. Optimal width ratios for the required output level were found through simulating the test bench over varied parameters. W_{M5}:W_{M6} affects the DC offset of the output while W_{M2}:W_{M4} controls the output level when it is high.

Figure 3-14 and Figure 3-13 show the stick diagram of the receiver and the corresponding layout. The current source and sink are on the left side of the layout.

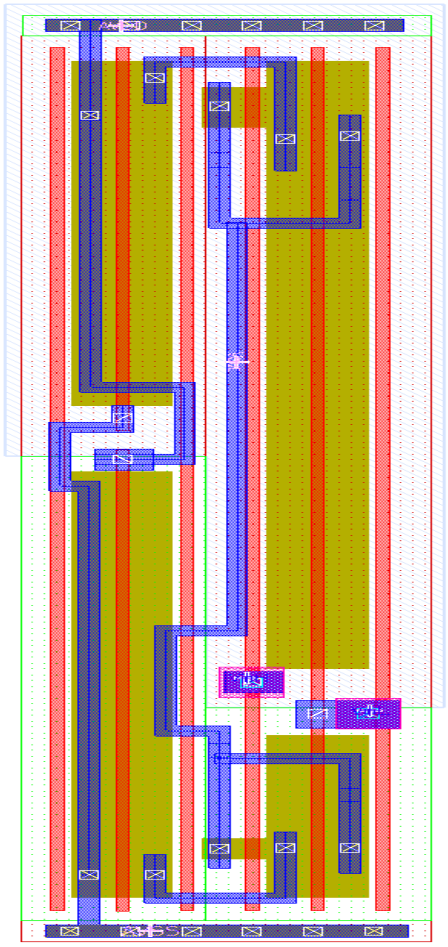


Figure 3-13 Developed Pseudo-LVDS receiver layout

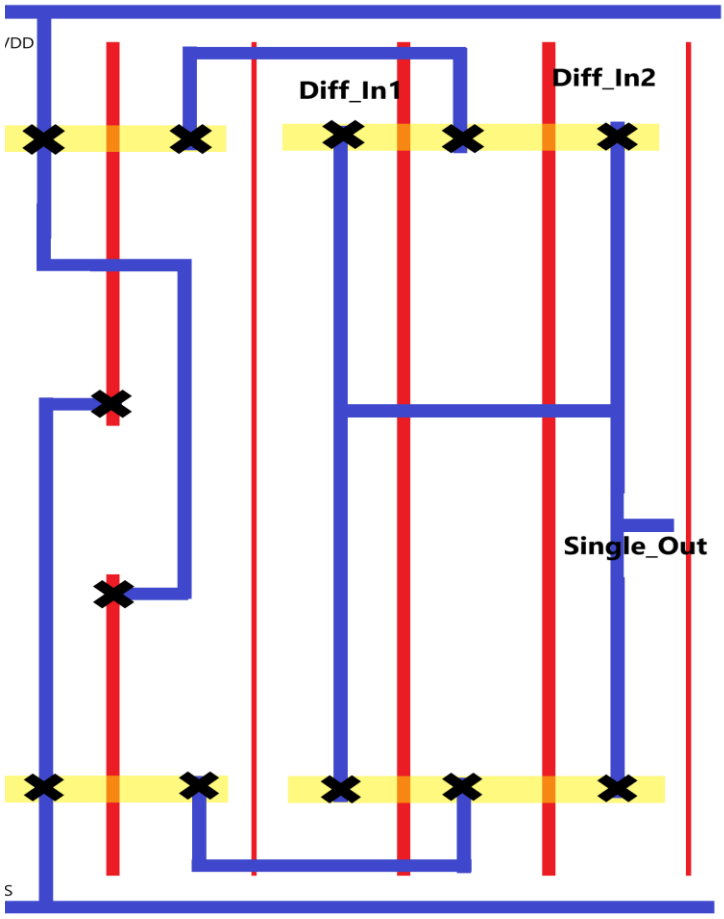


Figure 3-14 Developed Pseudo-LVDS receiver stick diagram

3.4.1.2 Transmitter

The final design for the pseudo-LVDS transmitter is based on the architecture discussed in [10], but without any feedback control for the V_{CM} as the common mode spiking was already small enough to meet standards across all corners.

Table 3-6 shows the intended electrical characteristics of the driver used in our encryption and decryption blocks.

Table 3-6 Pseudo-LVDS driver: Desired Electrical characteristics

Characteristics	**value**
Power supply	0.9-1.1 V
Differential voltage swing	±200 mV
Common mode voltage level tolerance	500 ±50 mV
Maximum Common mode variation	150 mV_{pp}
Maximum variation from steady state (V_{ss})	±20% V_{ss}
Built-in resistor	100±10% Ω

One of the most important issues with pseudo-LVDS transmitters is impedance matching. The output levels must be maintained for loads as small as 50Ω differentially, which is a standard impedance for transmission lines. The bridge resistors are 50Ω each, so the equivalent impedance across the bridge becomes 50Ω when a transmission line is connected to each output.

After the output finishes rising or falling, the current is always flowing through the current source (M9) and current sink (M1), the bridge resistors and an NMOS and PMOS that are on opposite sides of the bridge. So, the resistors and transistors essentially act as voltage dividers when voltage levels are steady. For a typical output of 0.5±0.25V, each transistor was sized to have about 16 Ω drain-source resistance. Gate width is 100µm for each NMOS and 258.5µm for each PMOS.

The design rules allow a maximum of 3.5µm gate width, so the transistors need 50 fingers with the widths shown in Figure 3-15, which is a simplified view of the final circuit. Due to contradictory design rules, each transistor was replaced by 25 parallel transistors with two fingers each as shown in Figure 3-16.

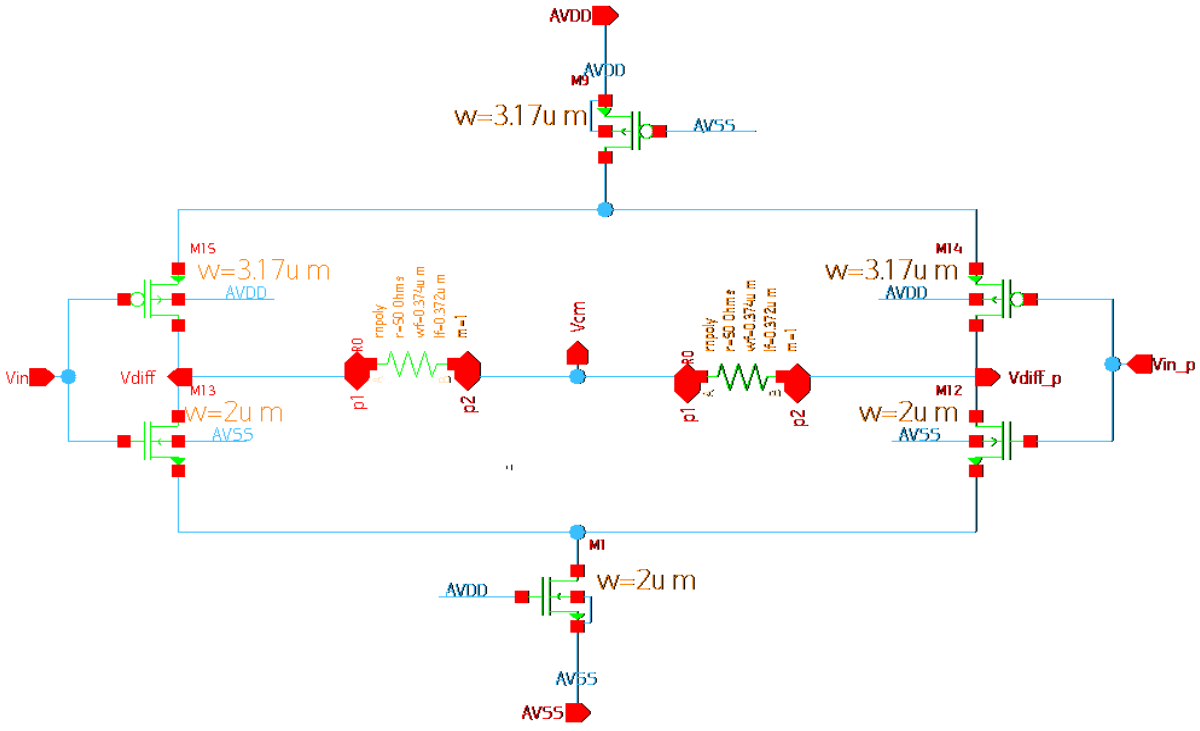


Figure 3-15 Symbolic Schematic of Developed Pseudo-LVDS Transmitter

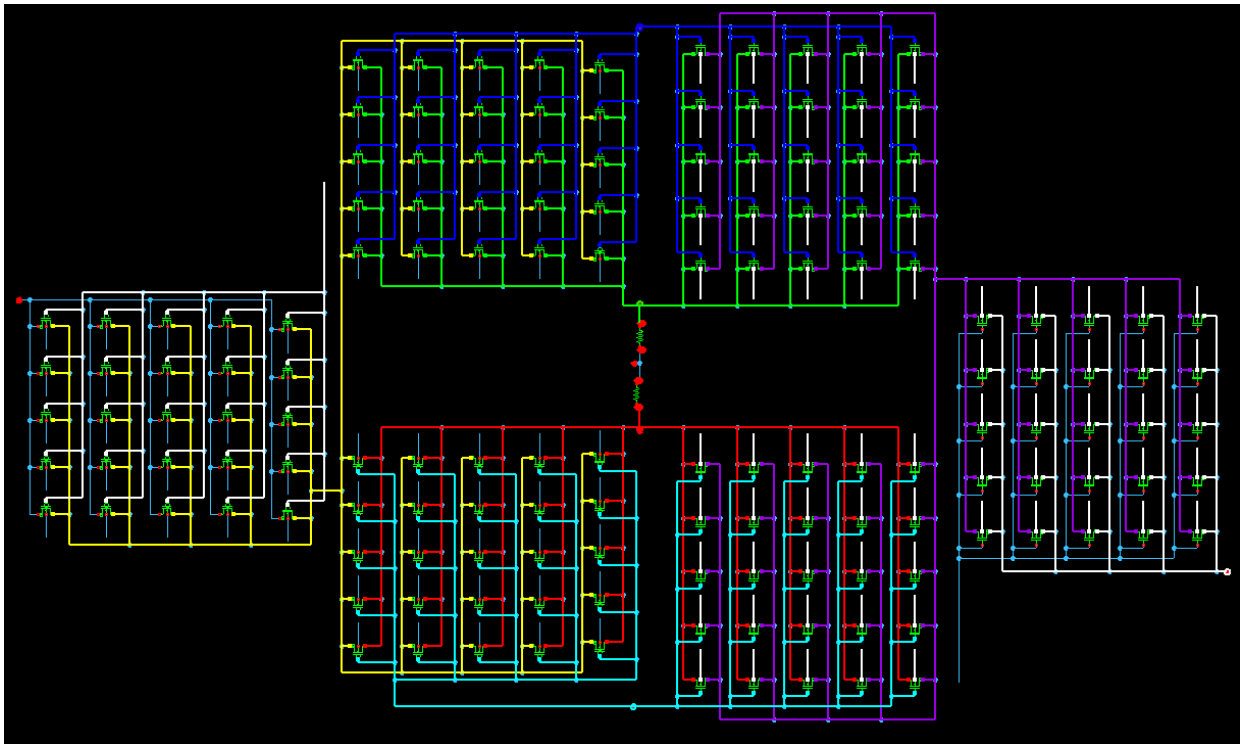


Figure 3-16 Detailed Schematic of Developed Pseudo-LVDS Transmitter

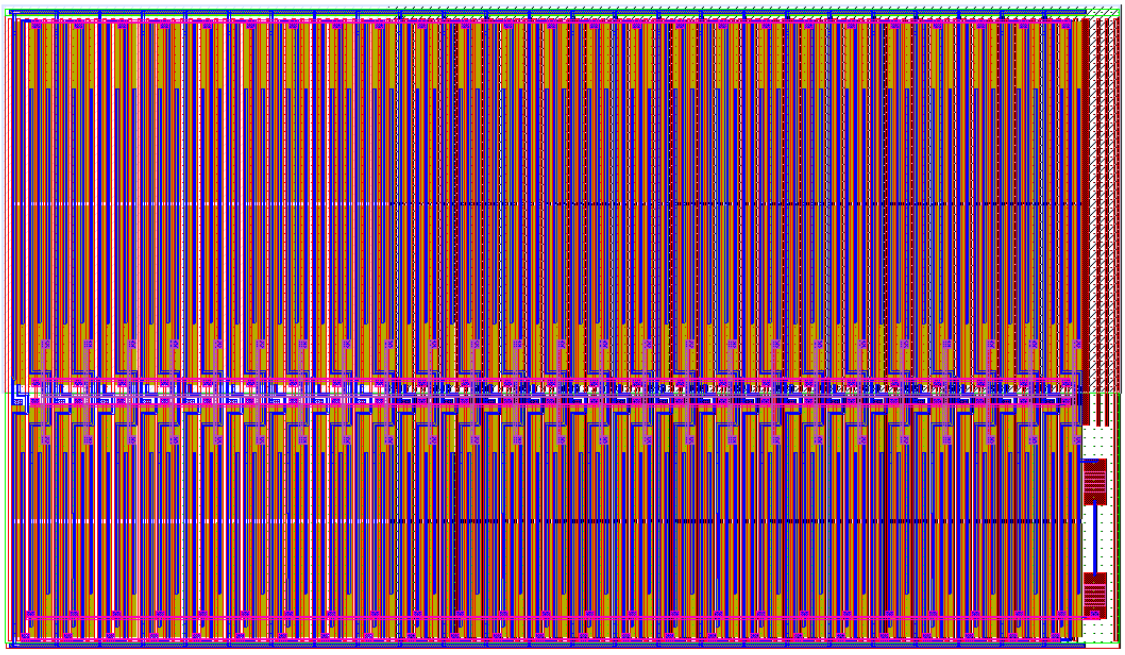


Figure 3-17 Full Layout of Developed Pseudo-LVDS Transmitter

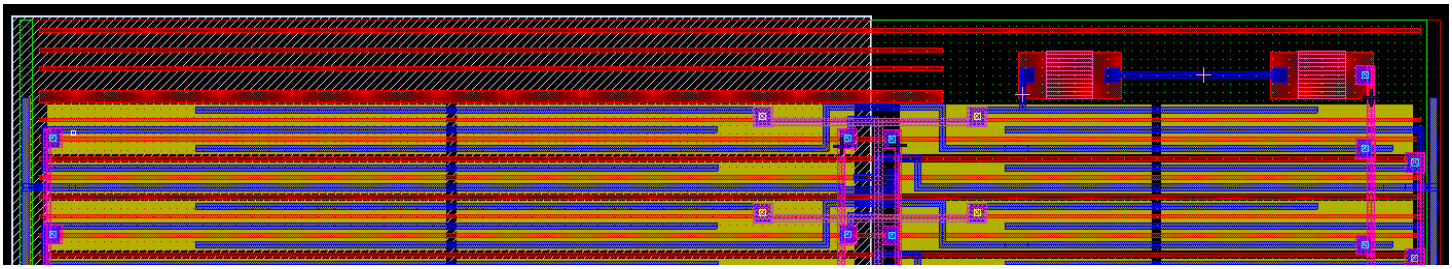


Figure 3-18 Developed Pseudo-LVDS Transmitter

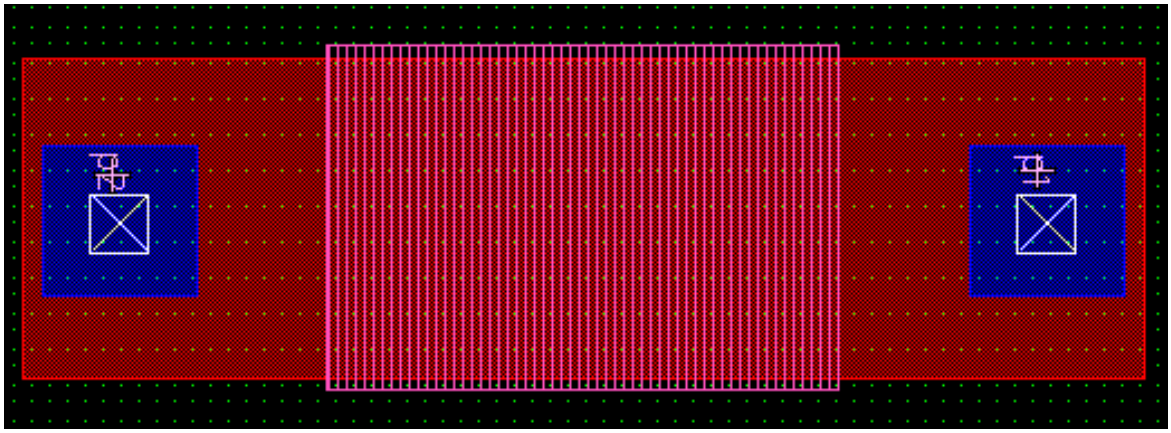


Figure 3-19 Fifty Ohms Resistor Layout

3.4.1.3 Driving Inverters

Since the transmitter has much higher input capacitance than the last inverter in the flipflops, connecting the PISO output directly to the transmitter input would result in exceeding the maximum 10ns total delay stated in the requirements. For optimum delay, we would need to use a series of 5 (N_{opt}) inverters, each 2.9 (A) times larger than the previous, starting with the inverter at the end of the D-flipflop according to the equations below [18]:

$$N_{opt} = \ln\left(\frac{C_{load}}{C_{in}}\right) \qquad A = \sqrt[N_{opt}]{\left(\frac{C_{load}}{C_{in}}\right)}$$

Equation 3-2 Number and Sizes of Inverters Producing Optimal Delay

C_{load} is the capacitance at the input of the transmitter, and C_{in} is at the input of the last inverter in any D flipflop. Since input capacitance is directly proportional to gate area and all transistors in our designs have 30nm gate length, we can replace the capacitances with the sum of widths of all transistor gates connected to the respective inputs.

Fitting the optimal set of inverters into the layout would exceed the area and cost constraints, so we opted to use a single inverter that is 10 times larger than the D flip flop inverter with $0.05C_{load}$ input capacitance.

3.4.2 D-Flipflop

The flipflop is used as a building block for the registers. The used flipflop uses extra two inverters at the beginning and the end, in addition to two inverters for feeding in the clock, which isolate the input and the output and the input clock. Figure 3-20 shows the schematic of the flipflop designed.

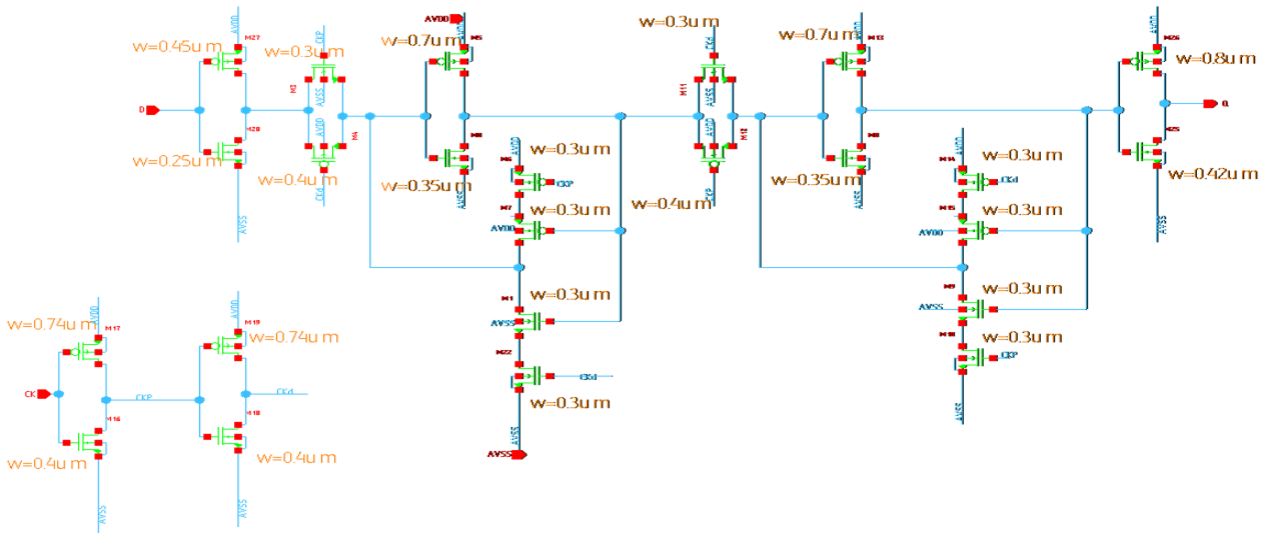


Figure 3-20 Developed D-Flipflop Schematic

Figure 3-21 and Figure 3-22 show the developed stick diagram and the corresponding layout for the D-flipflop.

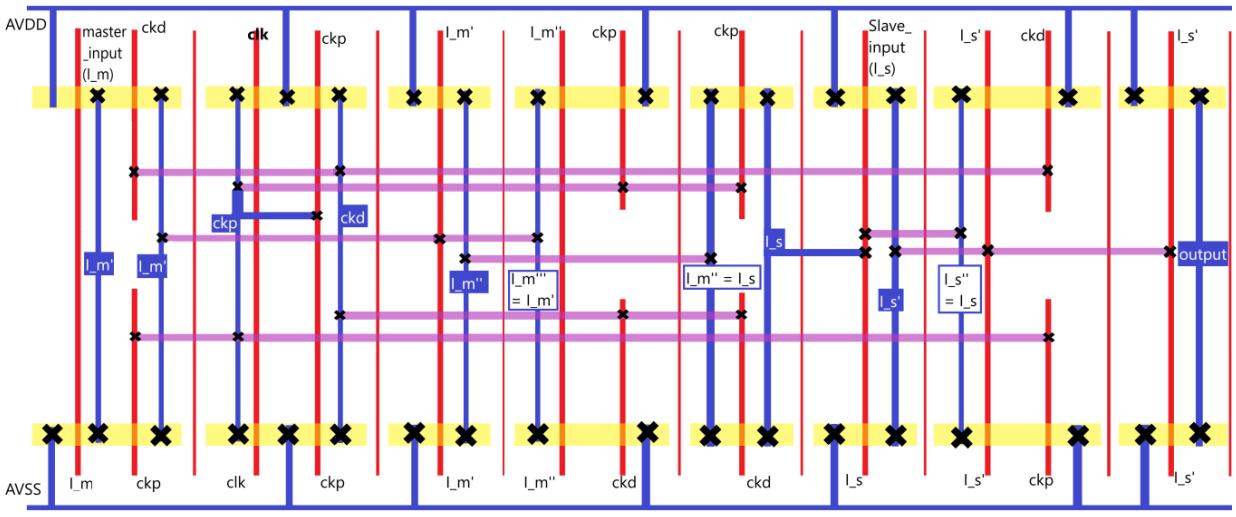


Figure 3-21 Developed D-Flipflop Stick Diagram

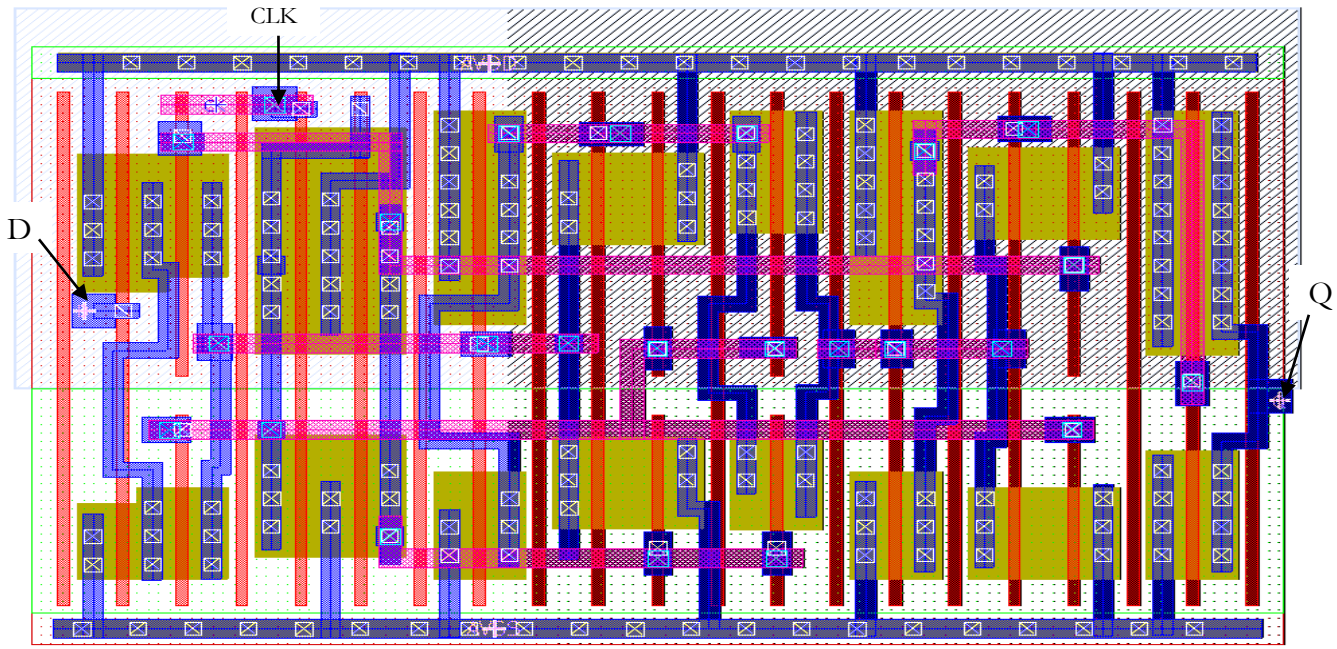


Figure 3-22 Developed D-Flipflop Layout

For the PISO register's output, the output is fed into the transmitter, and the transmitter's input is the signal and its inversion. To produce the inverted signal another flipflop with an extra inverter is added parallel to the last flipflop and its output is fed into the transmitter. Refer to Figure 3-23, Figure 3-24 and Figure 3-25. As the inverter is added at the input, this flipflop requires almost twice as much hold time as the normal D flipflop; this must be considered when skewing the PISO clock. Since no changes are made to the slave latch, propagation delay should not be affected, which is required by the transmitter input in order to minimize variations in common mode.

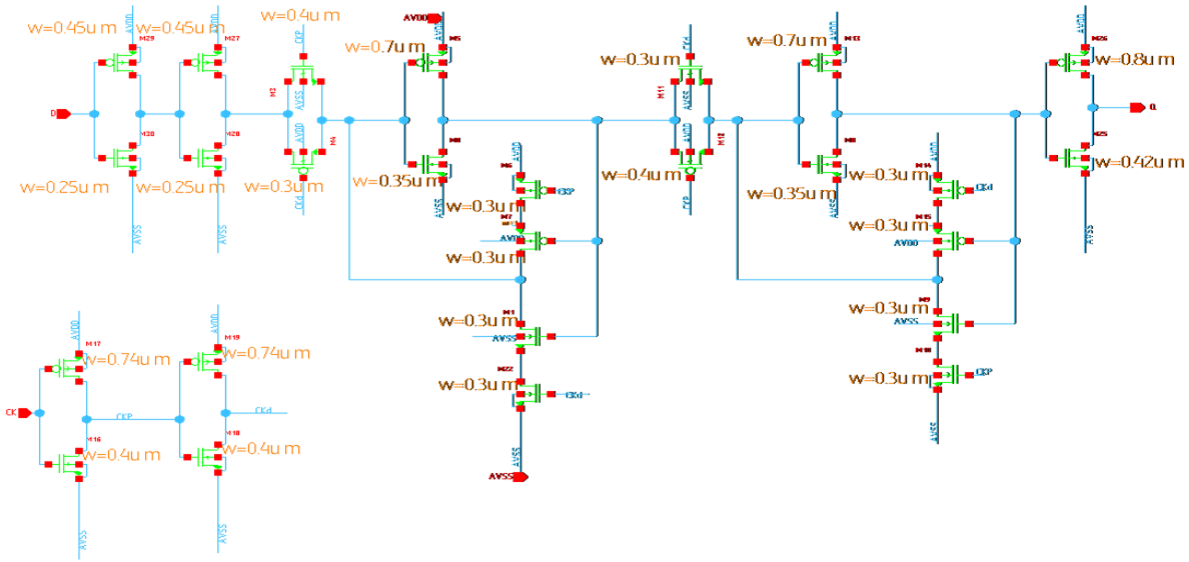


Figure 3-23 Developed D-Flipflop with Inverted Output Schematic

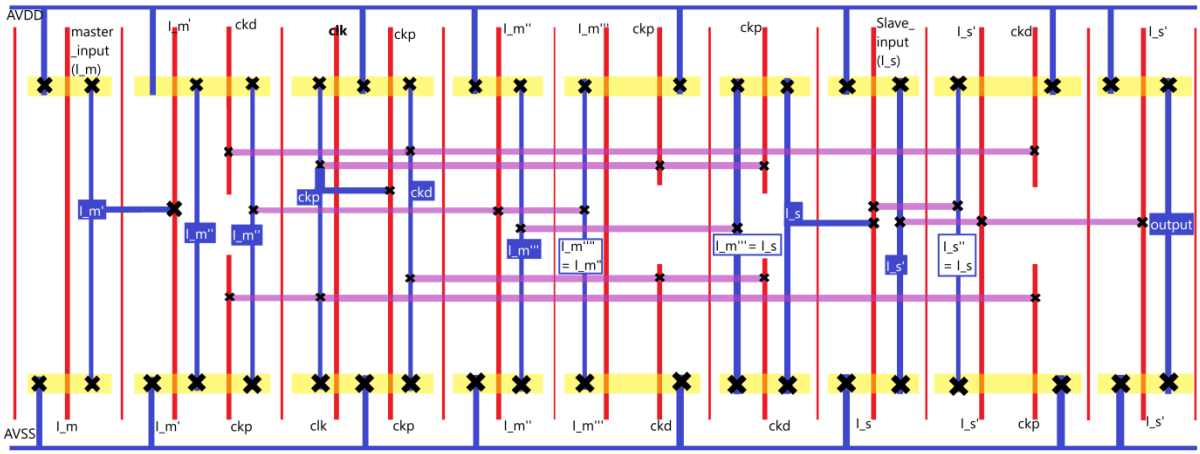


Figure 3-24 Developed D-Flipflop with Inverted Output Stick Diagram

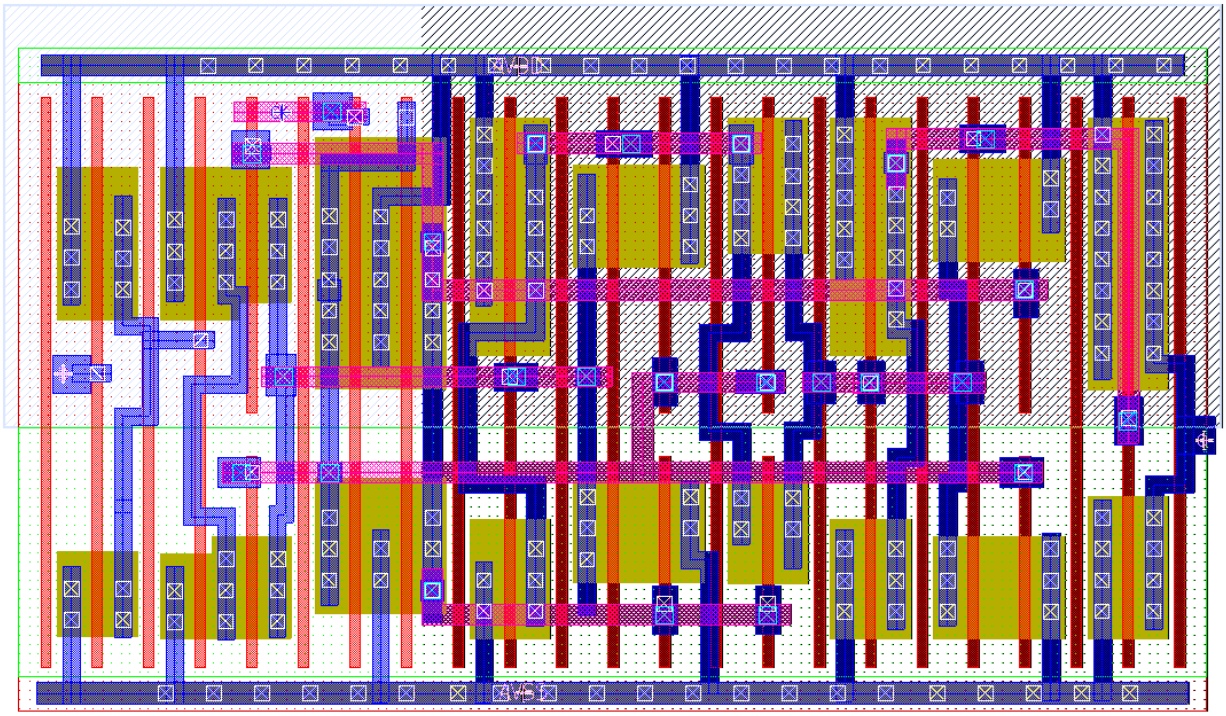


Figure 3-25 Developed D-Flipflop with Inverted Output Layout

3.4.3 Multiplexer

The MUX used is a CMOS MUX with pull-up and pull-down networks. We tried to reduce the needed inverters for the inputs by adding an inverter at the output and inverting the inputs in the pull up network. Figure 3-26 shows the developed design for the MUX.

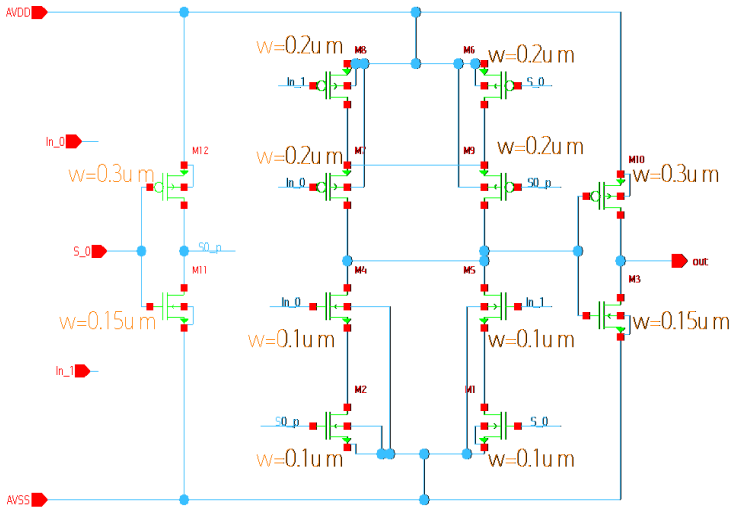


Figure 3-26 Developed Multiplexer Design

Figure 3-27 and Figure 3-28 show the stick diagram and the corresponding layout for the MUX.

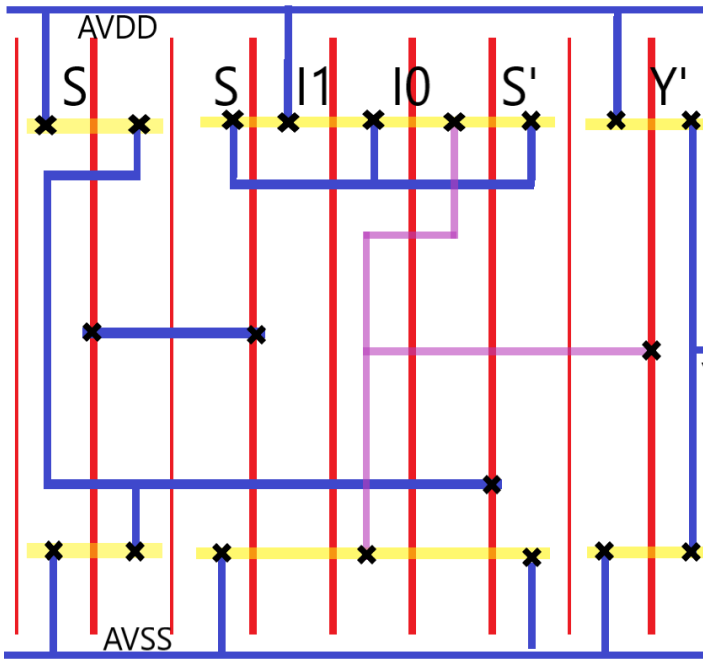


Figure 3-27 Developed Multiplexer Stick Diagram

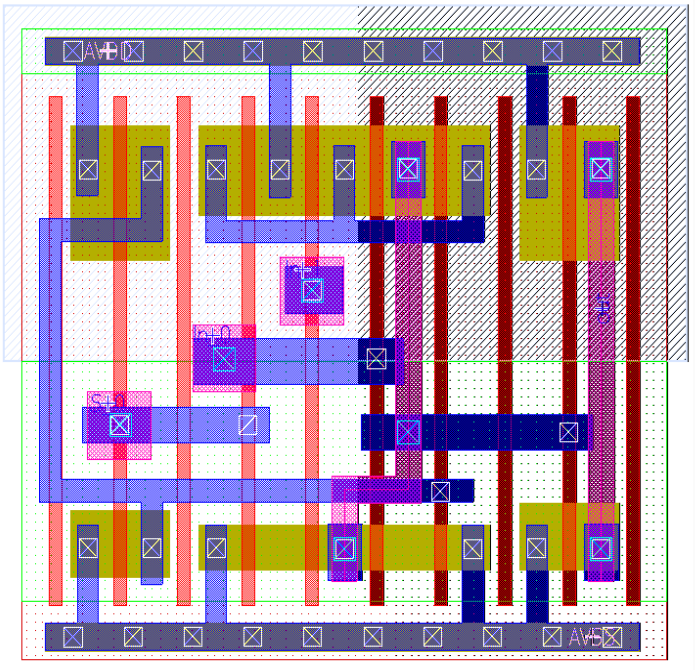


Figure 3-28 Developed Multiplexer Layout

3.4.4 Registers

3.4.4.1 Serial input parallel output shift register (SIPO)

An 8-bit register is used for storing the data in the design. It consists of 8 D-Flipflops. The output of the first flipflop is shifted to the next flipflop at each clock edge, this means the system needs eight clock cycles at least to convert 8 serial bits into parallel bits and start the encryption process.

3.4.4.2 Parallel input parallel output register (PIPO)

The key is stored in the PIPO register and feeds the XOR at each clock cycle. If the key is not hardwired or fixed in some other way, clock frequency of the PIPO must be according to Equation 3-3 below.

$$f_{pipoCLK} = \frac{1}{8N} f_{sipoCLK}$$

Equation 3-3 Key Register's Clock Frequency

For the above equation, $f_{pipoCLK}$ is the clock frequency of the PIPO, $f_{sipoCLK}$ is the clock frequency of the SIPO and N is a positive integer. If this is followed, each key is used to encrypt N bytes of data.

Furthermore, it is best if every positive edge of the PIPO clock coincides with a positive edge of the SIPO clock to ensure that the data and the key are per synchronized. Otherwise, if the key is fixed, then the same clock can be used for PIPO and SIPO, and it does not introduce any problems.

3.4.4.3 Parallel input Serial output register (PISO)

After the encryption is done, the data needs to be converted back to serial form. As in Figure 3-5, there is a MUX to control the loading of the data from the previous stage and the shifting to the successive DFF. The first step in the conversion is to load the parallel data in the flipflops when the load/shift$^-$ signal is high.

When the load/shift$^-$ signal gets low the data is shifted to the next flipflop. Since our parallel inputs are data processed after being shifted out by the SIPO, the first load pulse must rise after every eighth positive edge of the SIPO clock and fall before the next positive edge of the PISO clock. The PISO clock also needs to be skewed enough to allow enough time for the data in each parallel path to be processed.

3.4.5 XOR Gate

This stage XORs the parallel data with the parallel key, which changes the data, but in order to make it harder to guess the key from XOR-ing the encrypted data with the original data, the outputs are shifted by for bits by changing the wire placement (i.e. the first bit is connected to the fifth bit in the PISO, the second bit is connected to the sixth… etc.).

Figure 3-29 shows the schematic of the XOR gate developed.

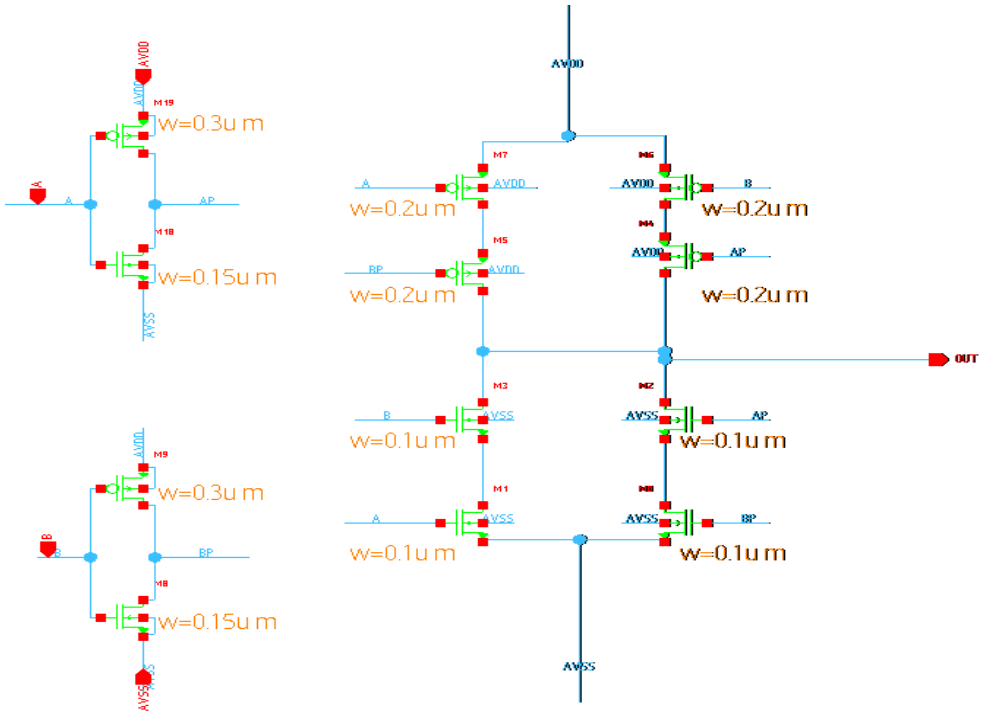


Figure 3-29 Developed XOR Schematic

Figure 3-30 and Figure 3-31 show the developed stick diagram and the corresponding layout.

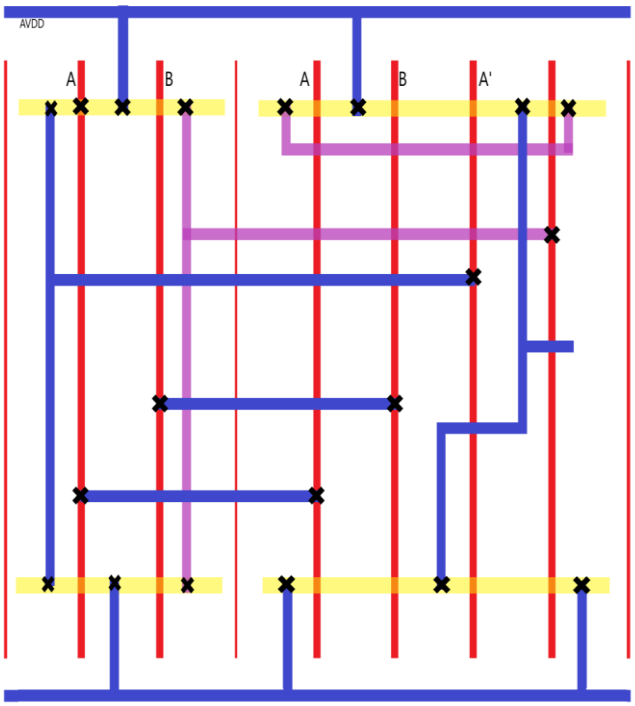


Figure 3-30 Developed XOR Stick Diagram

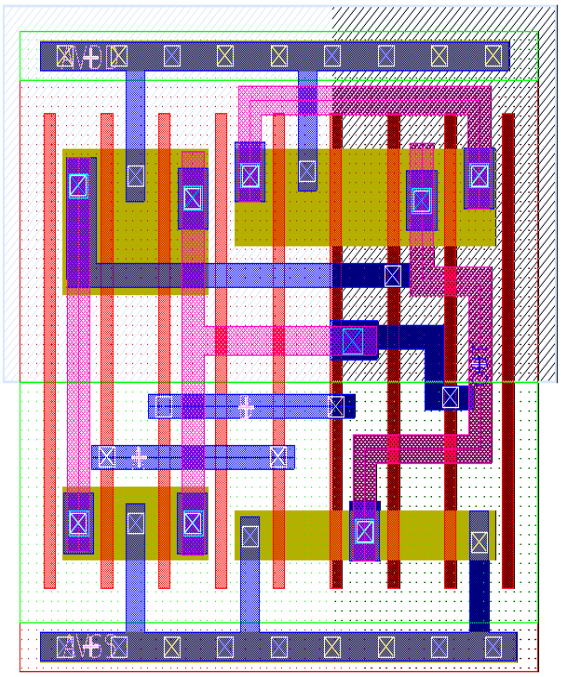


Figure 3-31 Developed XOR Layout

3.4.6 Completed Blocks

Our encryption and decryption blocks are identical in every way except for the wiring. Table 4-3 on page 51 lists the area and number of each component used and Figure 3-32 and Figure 3-35 show the encryption block schematic and layout. Appendix A contains the netlist generated by the SAE tool in Synopsys for the schematic.

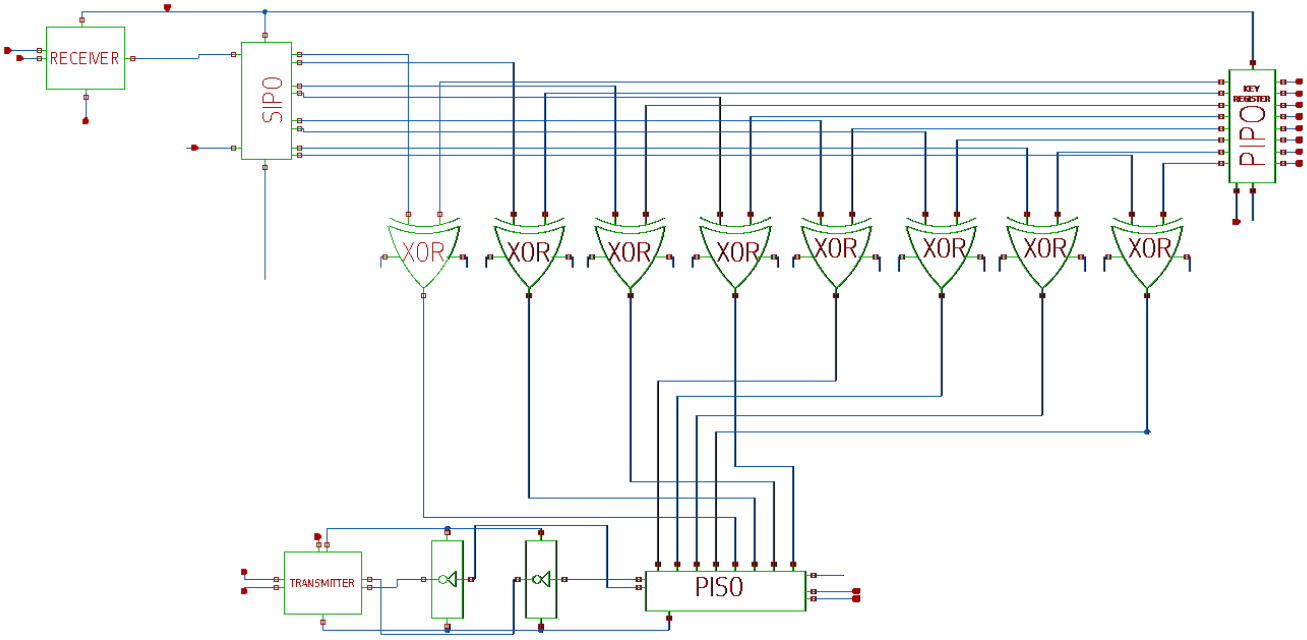


Figure 3-32 Developed Encryption Block Schematic

The decryption block is almost identical to the encryption block; the only difference is that the decryption key is shifted by four at the input, as shown in Figure 3-33 and Figure 3-34.

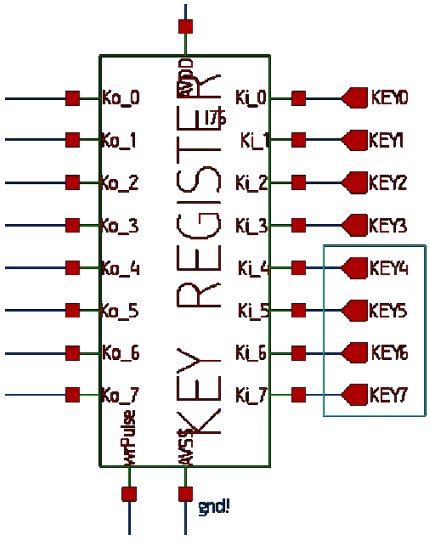


Figure 3-33 Encryption Key Register

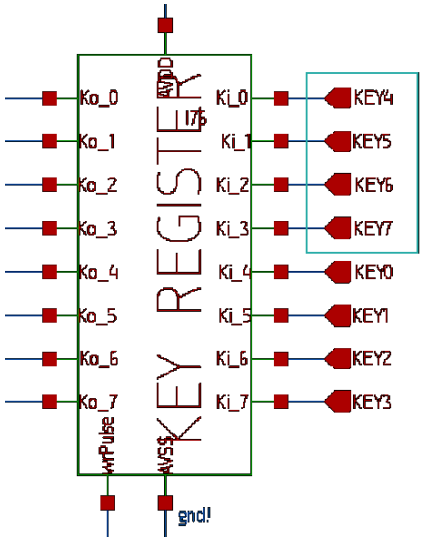


Figure 3-34 Decryption Key Register

Figure 3-35 is labeled to show the general placement of all the components. The key registers form a row at the top, followed the SIPO in the next row, then the PISO multiplexers are interleaved with the XORs and the last row before the transmitter contains all the normal PISO DFFs. The transmitter visibly takes up most of the area and the receiver is to its left while the driving inverters and the inverting DFF from the PISO are to its right.

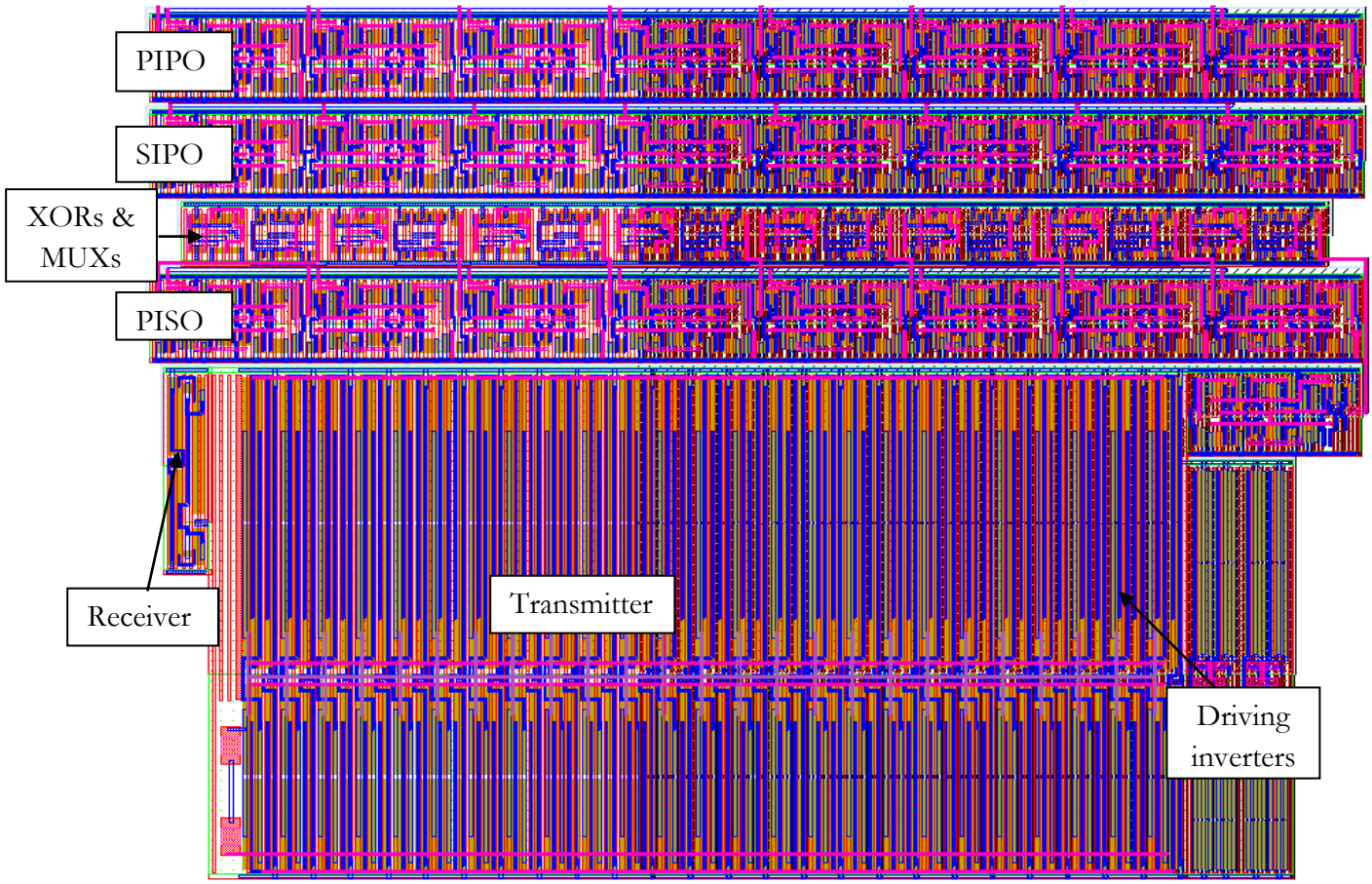


Figure 3-35 Developed Encryption Block Layout

4 Results

4.1 Simulation Setup

After the conceptual development of the system was complete, physical design began by developing each component separately and testing individual functionality and limits before moving on to the sub-blocks, and then the complete block. Test benches are set up with preceding stages included and capacitive loads based on the input of the next stage. Since layout parasitic extractions were not available and we could not run post-layout simulations, we doubled the expected capacitive loads which were estimated using the following equations for capacitance of transistor's gate from [18]:

$$C_G = \frac{\varepsilon_{ox}}{t_{ox}} WL \ , \quad \varepsilon_{ox} = (dielectric\ constant) \times \varepsilon_o$$

Where ε_o is the permittivity of the free space.

$$\varepsilon_o = 8.85418782 \times 10^{-12}$$

Equation 4-1 Gate Capacitance

Where W and L are the width and length of the gate, dielectric constant is assumed to be 3.9 and t_{ox} was approximated to be around 1.5×10^{-9} from [19]. The input capacitances for our components were calculated to be:

0.50 fF for a D flipflop
0.32 fF for an XOR or MUX
5.00 fF for a driving inverter
180 fF for the transmitter
0.21 fF for the VinH input of the receiver
1.71 fF for the VinL input of the receiver

Every component was tested across PVT corners to ensure that requirements and constraints have been adhered to. The three sets of conditions simulations were done under are:

- **FF process with 1.1 V supply at 70ºC**: the best-case scenario is that fast transistors (with lowered $V_{threshold}$) are working with a 10% rise in the power supply. This is expected to have the smallest delays but have the largest power consumption.
- **TT process with 1.0 V supply at 25ºC**: this is the nominal operating point and should result in delay and power values somewhere between the other two cases.
- **SS process with 0.9 V supply at 0ºC:** the worst-case scenario is that slow transistors (with raised $V_{threshold}$) are working with a 10% decrease in the power supply. This is expected to have the largest delays but have the smallest power consumption.

4.2 Test benches, Logical Functionality and Simulation Results

This section includes different test benches and their results. For clarity, the following color-code is followed **unless stated otherwise**:

Table 4-1 Waveforms Legend

Color	Corner
Red	SS process with 0.9 V supply at 0°C
Dark Blue	FF process with 1.1 V supply at 70°C
Black	TT process with 1.0 V supply at 25°C

4.2.1 Receiver test bench and resulting waveforms

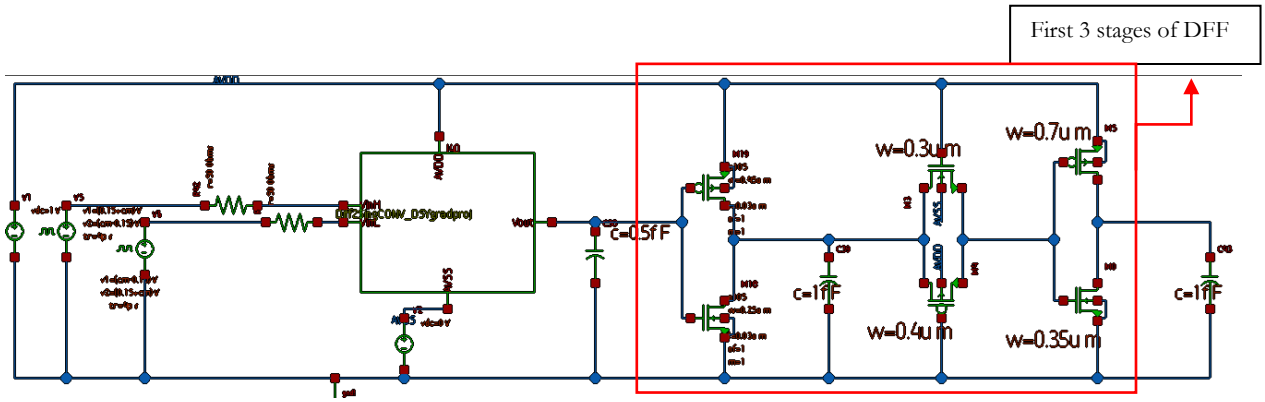


Figure 4-1 Receiver Test bench

The receiver uses two inputs to receive the differential signal (as shown on the left side of Figure 4-1 above) and produces a single-ended signal. A 50Ω resistance has been added at each input to represent a transmission line's impedance. Since the output swing varies greatly at different inputs and corners, the first three stages of a D flipflop have been added at the output to ensure that the produced signal can be processed by the SIPO.

Figure 4-2 shows the resulting eye diagram of the receiver at typical operation, when the input differential swing is ±200 mV and the common mode voltage is 0.5V at 1 Gb/s. The eye diagram iteration shows that the output signal steady state does not reach 1V or 0V, but the eye opening is large enough for the level to be detected and corrected in the next stage. Although it looks too ideal for maximum data rate, this is only the typical operation and performance worsens across PVT corners and with variations in input voltage levels.

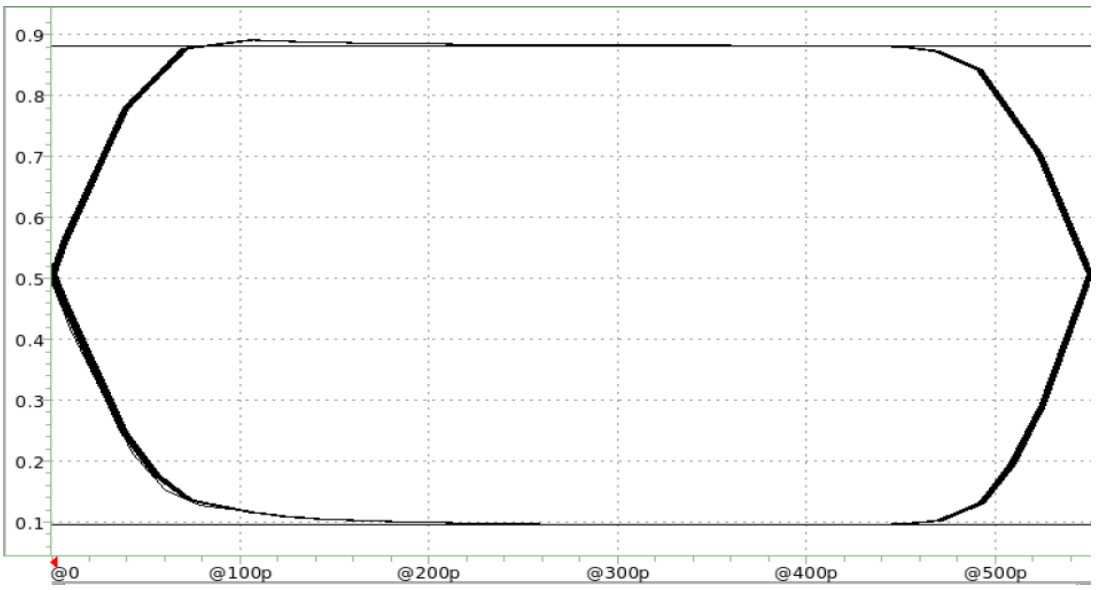


Figure 4-2 Eye Diagram for Receiver Typical Operation

Then the input differential swing is varied to find the minimum and maximum differential swing the receiver can sense (Figure 4-3). Then, it was tested for common mode level tolerance, to find the maximum and minimum input common modes that do not create errors in the output signal (Figure 4-4). Since the receiver's output never reaches ground or supply, functionality is decided by whether the first three stages of the DFF can correctly interpret the data.

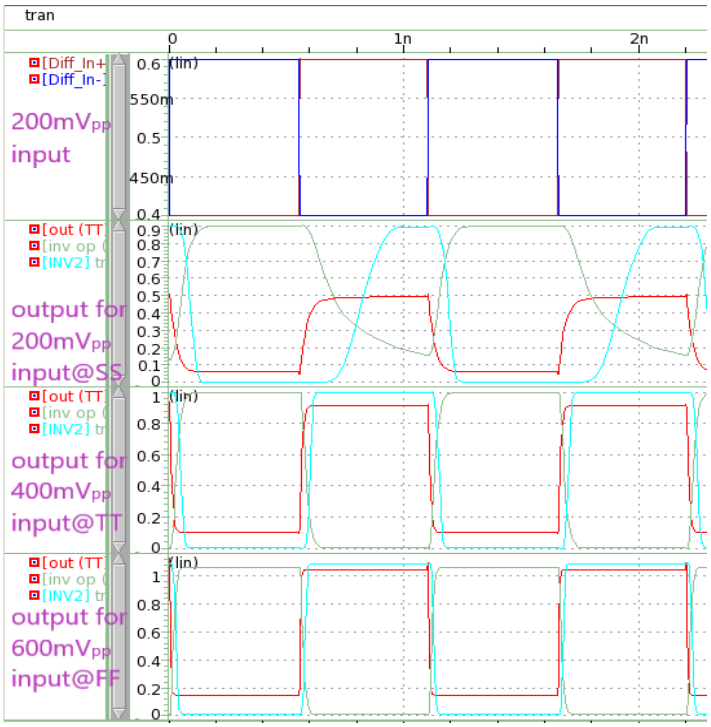


Figure 4-3 Receiver Operation with Different Input Swings

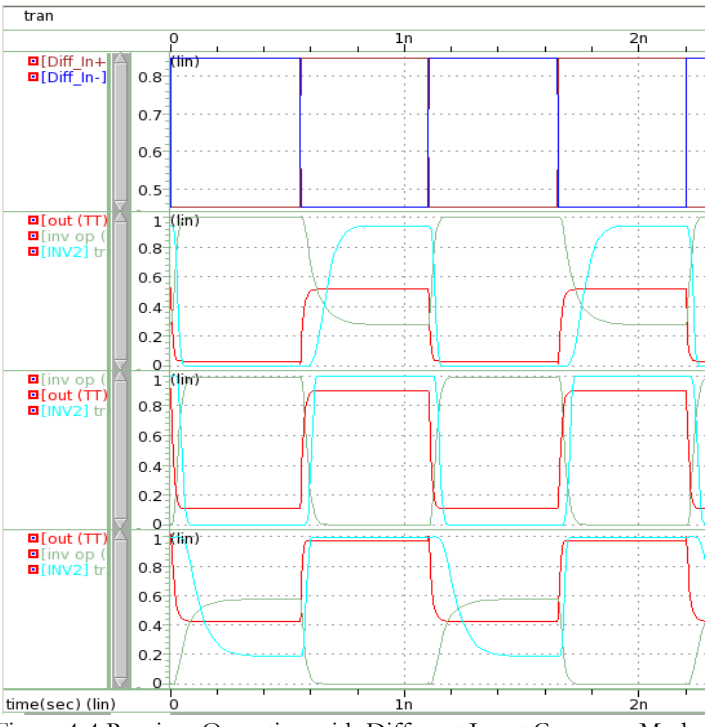


Figure 4-4 Receiver Operation with Different Input Common Modes

4.3 D Flipflop test bench and results

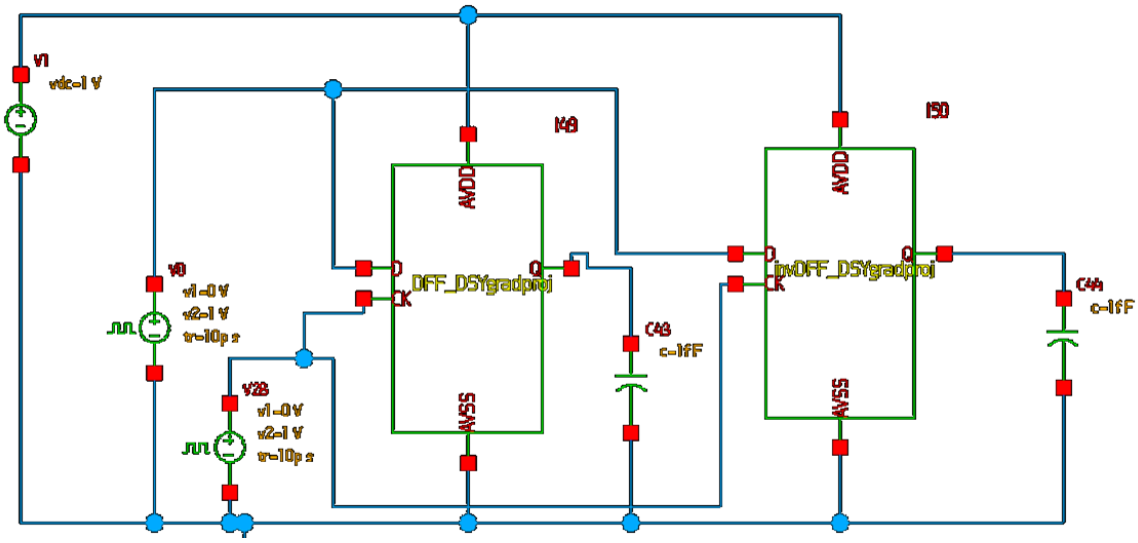


Figure 4-5 D flipflop test bench

The normal D flipflop and the one with inverted output are tested together. Figure 4-6 and Figure 4-7 show the typical operation with different hold times. In Figure 4-6, there is only 20ps between the DFF clock's rising edge and the previous change in data. While that is enough for the normal DFF, the inverting DFF does not capture the third bit.

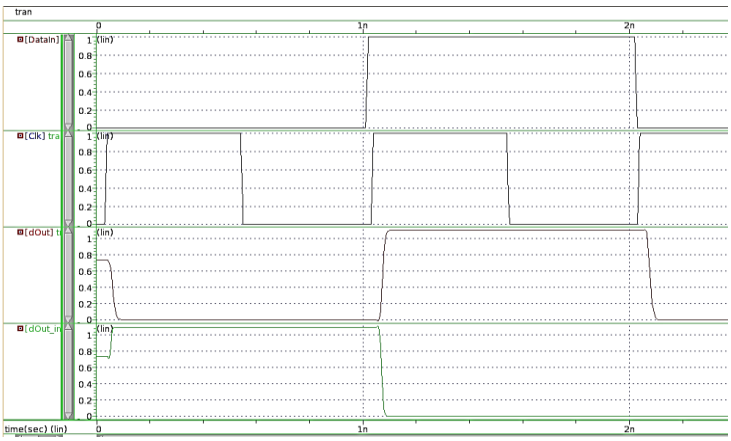


Figure 4-6 Typical D flipflop operation with insufficient hold time

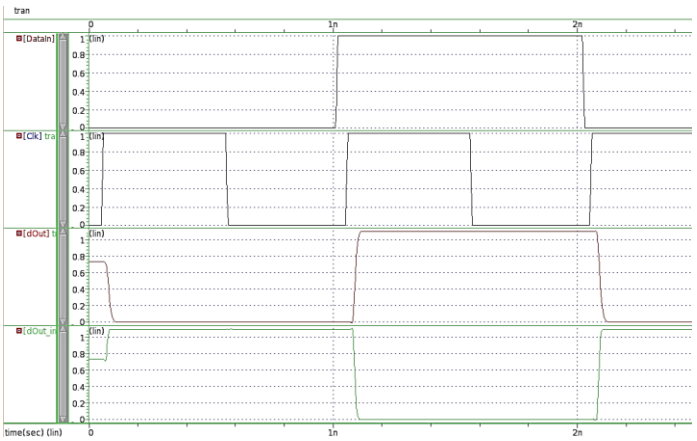


Figure 4-7 Typical D flipflop operation with adequate hold time

Table 4-2 Required hold times for DFF operation

Corner	SS	TT	FF
Normal D flipflop	15.3 ps	9.7 ps	4.9 ps
Inverting D flipflop	34.7 ps	18.5 ps	10.4ps

4.3.1 A Single Parallel Path

One path from the 8 parallel paths in the encryption/decryption circuit is examined in this test bench (Figure 4-8).

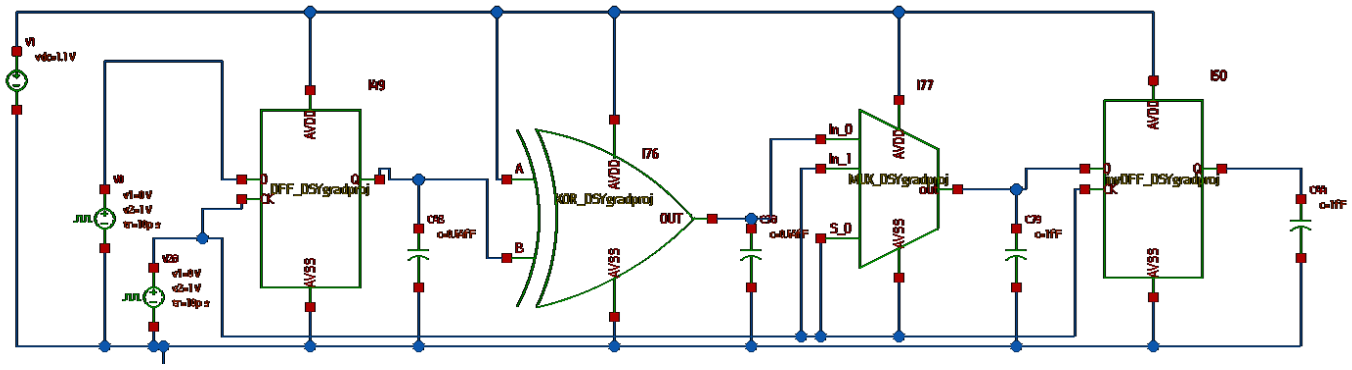


Figure 4-8 Test bench for a parallel path to process a single bit

The path is tested at all corners with a 2 GHz clock and the results can be seen on the left side of Figure 4-9. The system works well and the small glitches at the XOR and the MUX outputs are omitted, since the last stage is a flipflop that samples the data at the rising edge. These glitches or spikes appear at the falling edge of the clock and do not pose a problem as data is sampled at rising edges. This simulation was used to measure propagation delay of and power consumption of the XOR and MUX.

Circuit operation is still satisfactory at 5GHz but XOR and MUX outputs begin to lose definition as shown on the right side of Figure 4-9; this is considered the highest operating frequency, so only FF and SS corners are simulated just to prove functionality.

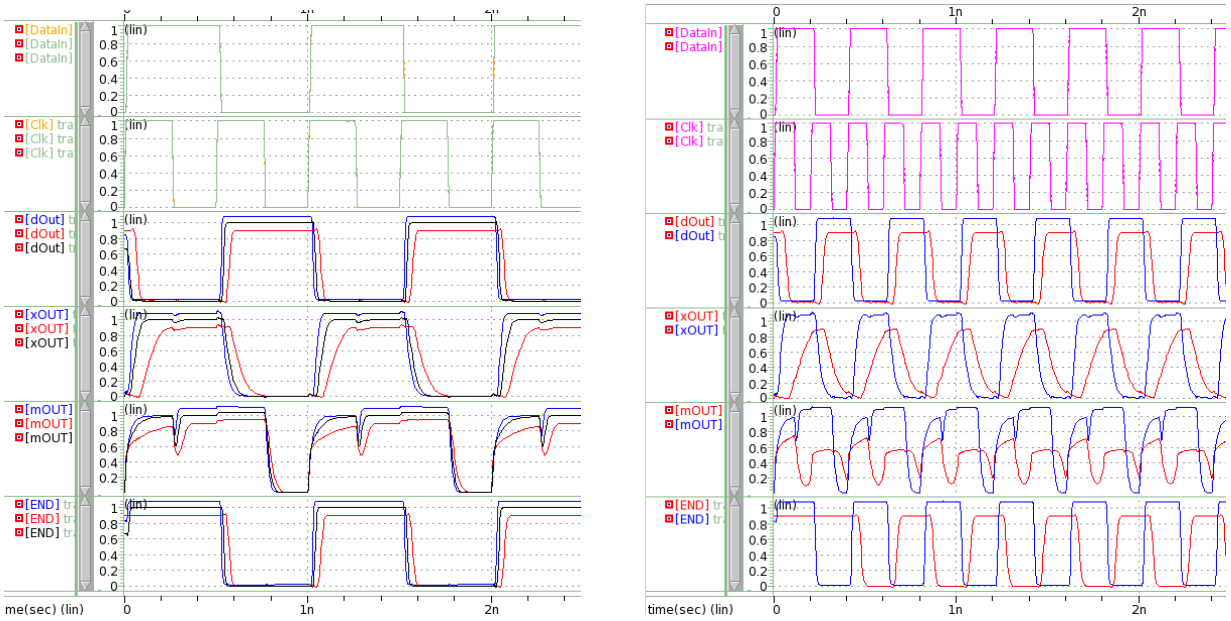
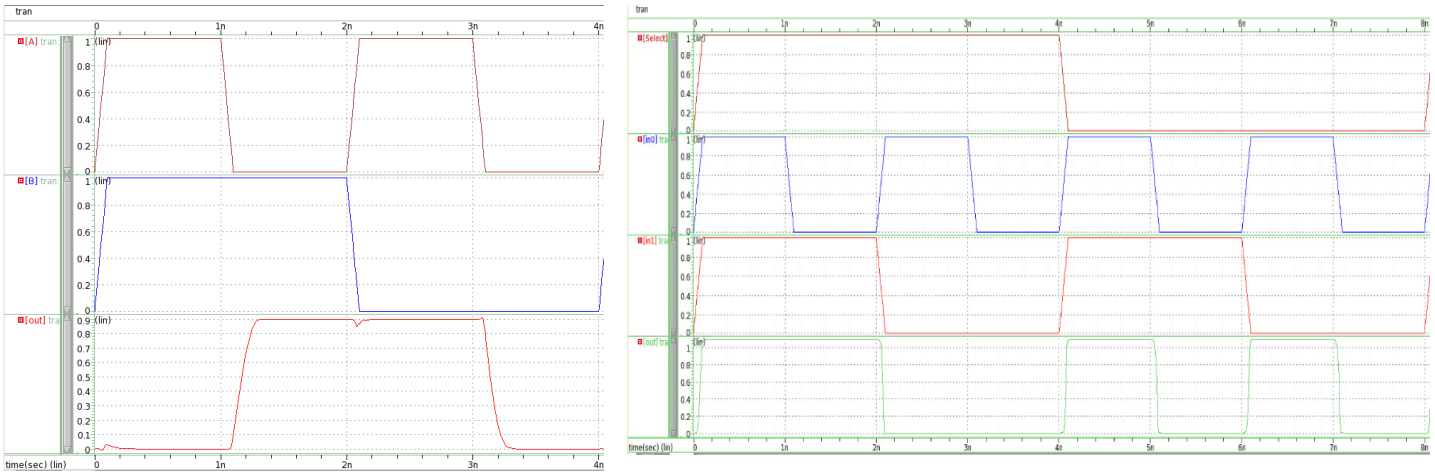


Figure 4-9 Single-bit System at 2 Gb/s and 5 Gb/s.

The MUX and the XOR gates were also tested individually at typical conditions to ascertain logical functionality, the results of which can be seen below. The bottom waveform is the output for the respective component. For the MUX waveforms in Figure 4-10 (b), the top panel has the select line (S_0) which is high for the first half of the simulation so In_1 (third panel from the top) is passed to the output and when S_0 is low, In_0 is passed on. The XOR results in Figure 4-10 (a) are also as expected: the output is high when the two inputs are different, but low otherwise.



(a) (b)

Figure 4-10 Logical functionality of (a) XOR and (b)Multiplexer

4.3.2 Functionality of Shift Registers

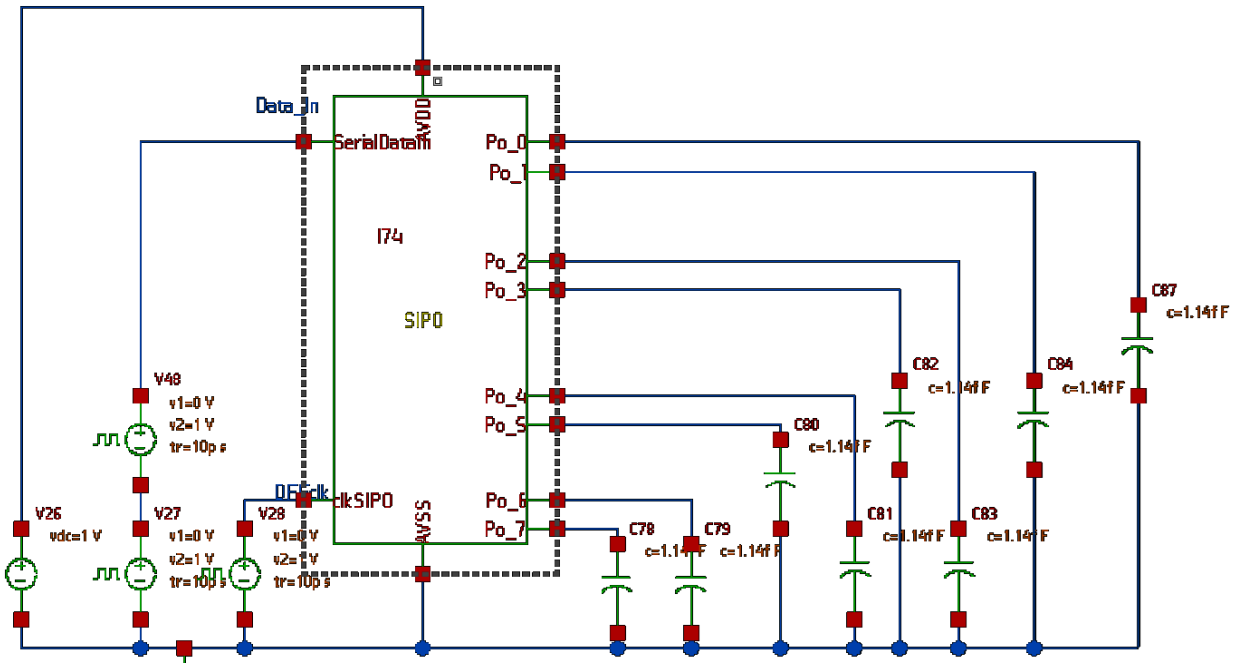


Figure 4-11 SIPO test bench

Figure 4-11 shows the simulation settings for the SIPO shift register. The same settings are used for the PISO. Each capacitor has the collective input capacitance of one DFF and two XORs to double the capacitive load as mentioned before.

The output of the SIPO is obtained at the 8th clock cycle as shown in Figure 4-12, so the delay of the block depends heavily on the number of bits processed – smaller number of bits lead to less delay. In this case, the clock edges follow the data edge very closely and that is sufficient. However, if the data has a low slew rate or does not rise or fall properly as seen with the receiver, it is better to allow as much slack as possible.

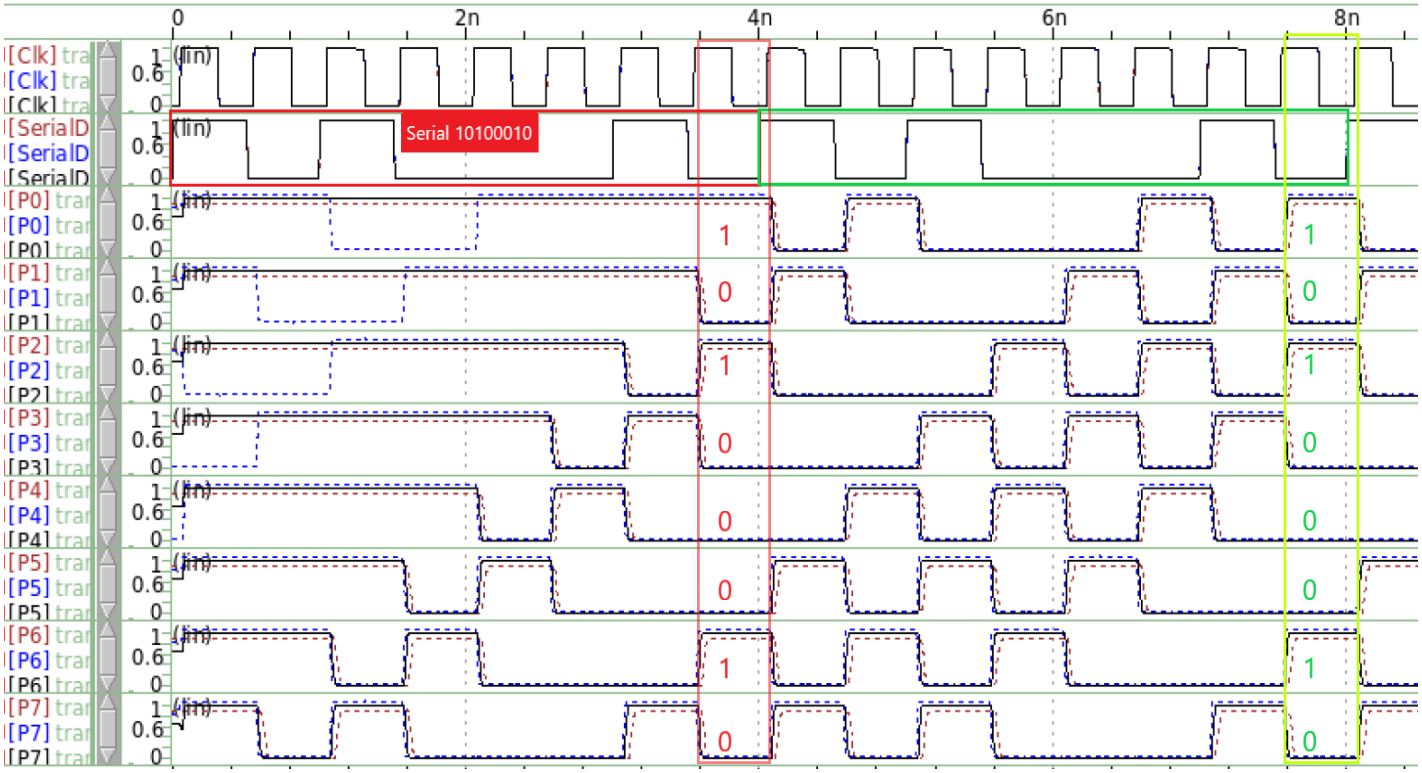


Figure 4-12 SIPO functionality across all corners

The PISO was tested by feeding in the output of a SIPO and then looking at the two serial outputs as shown in Figure 4-13. The original serial data and resulting PISO outputs are enclosed in blue boxes. Only the first and last parallel data are shown since the serial outputs are clearly identical or inverse in logical value to the original serial data. The green box highlights how the load signal rises after the parallel data is ready, remains high long enough for the PISO DFFs to capture said data and falls before the next rising edge of the PISO clock so that the loaded data is shifted forward and out bit by bit.

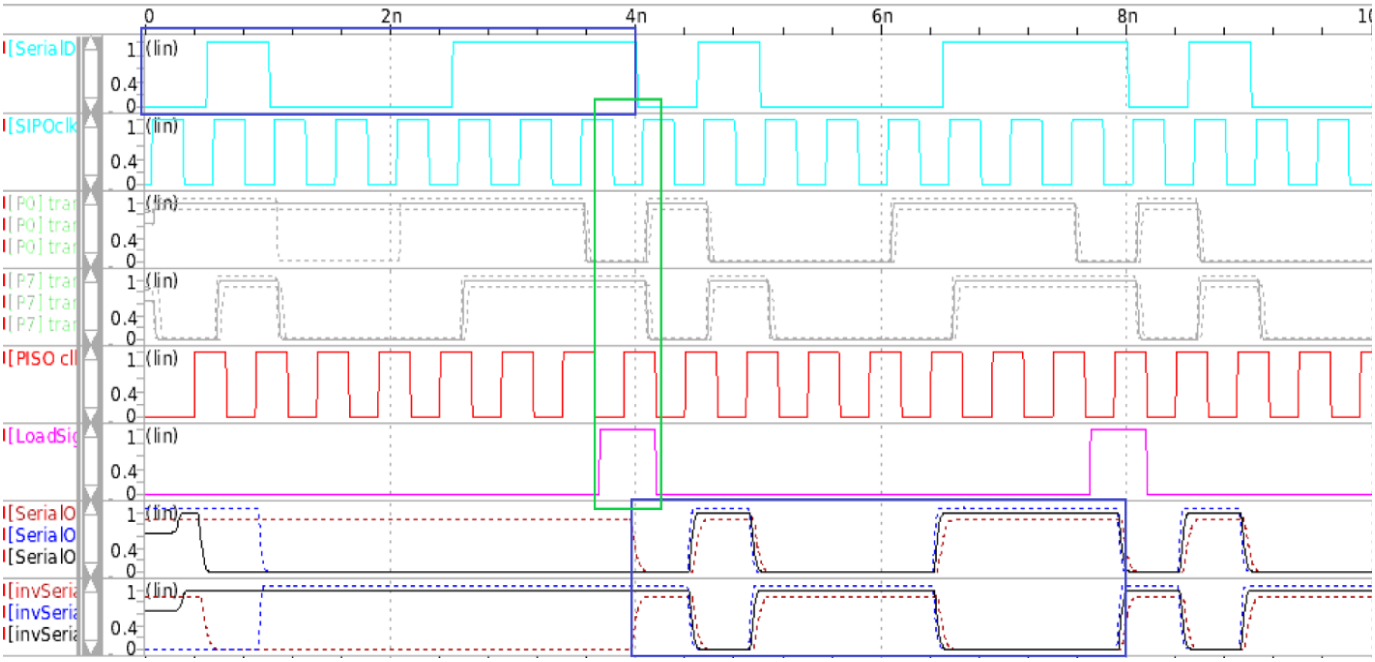


Figure 4-13 PISO functionality across all corners

4.3.3 Transmitter test bench and results

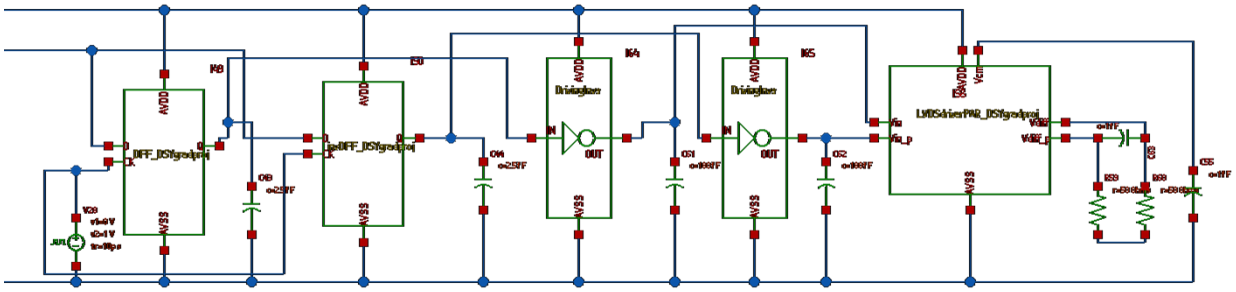


Figure 4-14 Transmitter test bench

The transmitter test bench is depicted in Figure 4-14. The inputs are generated by a receiver (not shown above) followed by the last two parallel flipflops of a PISO and two driving inverters (one for each input). The outputs are loaded with 50Ω resistors and a 1fF capacitor in accordance with the pseudo-LVDS standards. This test bench is also used to collect data on the driving inverters.

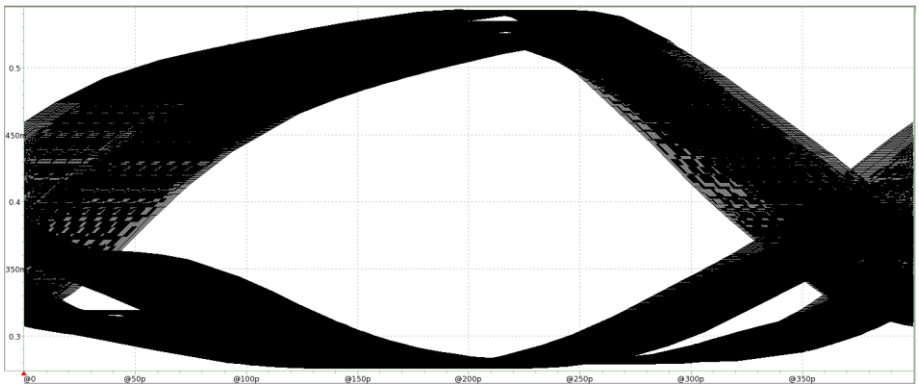


Figure 4-15 Driver's eye diagram at SS at 2.5Gb/s

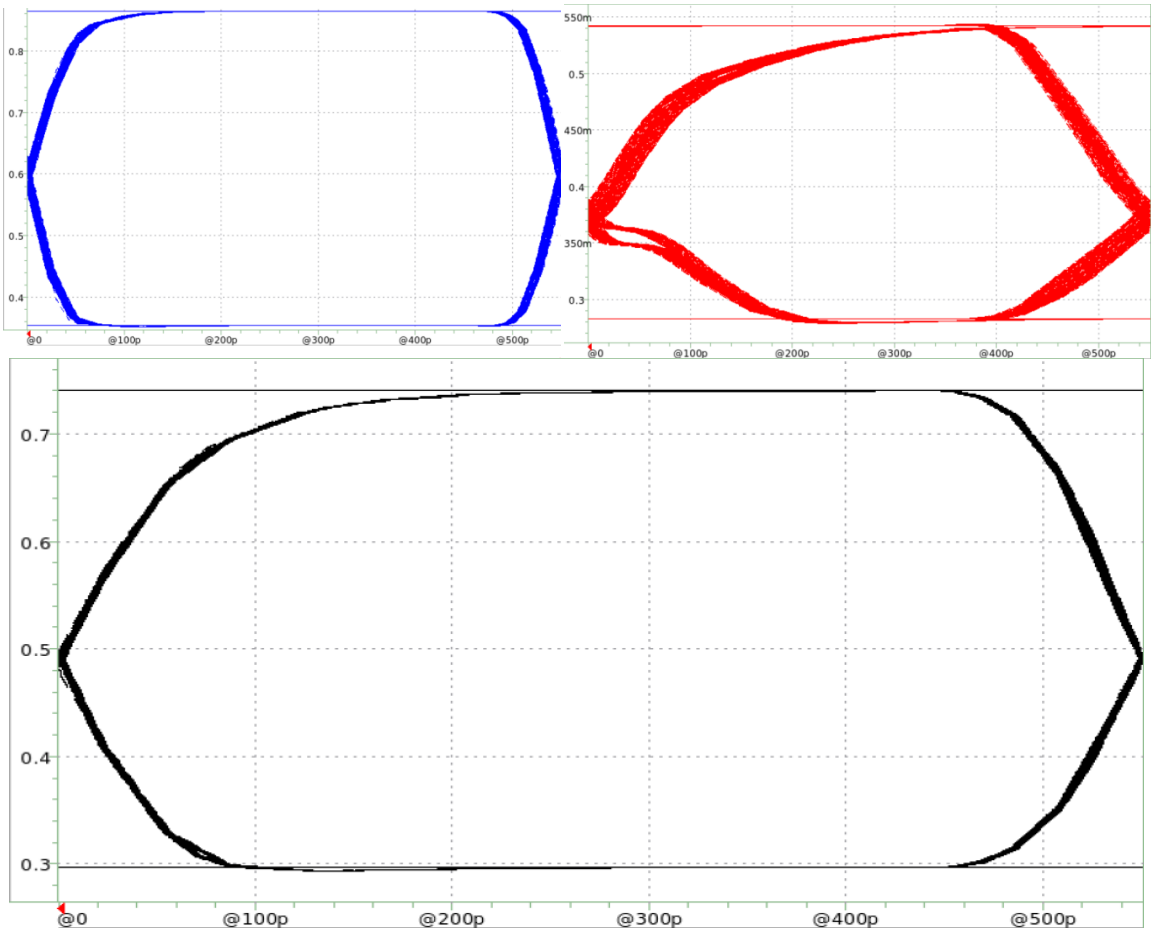


Figure 4-16 Driver's eye diagrams at 1.8 Gb/s

The eye diagram in Figure 4-15 shows that the output just barely meets the pseudo-LVDS standards, and this is the worst case at the highest data rate (2.5 Gb/s) tolerated by the receiver. Figure 4-16 shows the output eye diagrams at the three different corners at 1.8 Gb/s – the

maximum data rate for the receiver. A summary of the results from measuring these eyes is shown in Table 4-7 in the Analysis of Results section.

4.3.3.1 Sizing the Transistors

The following circuit was used to empirically find the sizing of the transistors in the driver. First the matching W_p:W_n ratio was assumed to be 2 and a simulated sweep was run over W_n to find the 100µm NMOS width for 16Ω. Then, W_n was fixed and the W_p was swept as a varied multiple of W_n to find the correct matching ratio of 1.58.

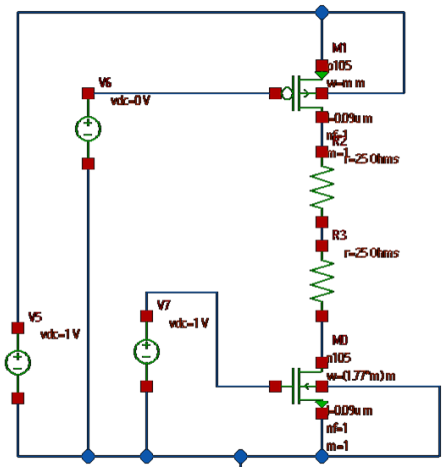


Figure 4-17 Transistor Sizing Test Bench

4.3.4 Functionality of Completed Block

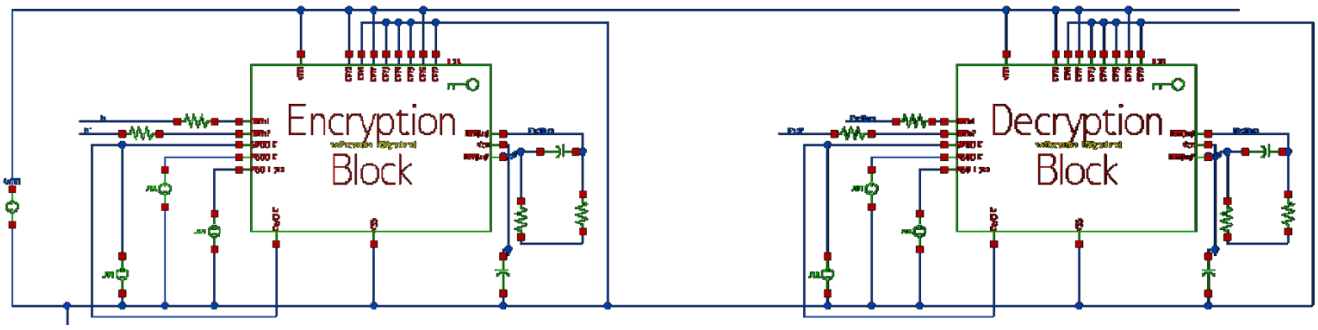
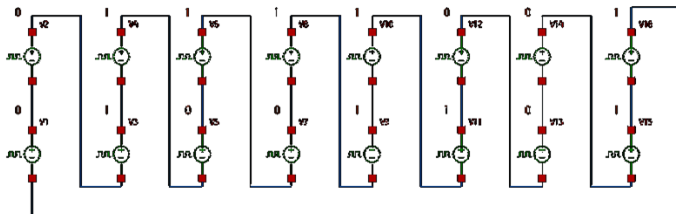


Figure 4-18 Test bench for Encryption and Decryption

The encryption and decryption blocks are tested together. Although each component can perform at 1.8Gb/s or above, it appears that some bits are lost when the whole system is forced to operate at this speed. Since the maximum total delay (required to remain below 10ns) is exceeded with

clock frequencies below 900 MHz, we have chosen to simulate the above circuit at 1 GHz to prove functionality, that the system can run at 500 MHz (since that is a lower frequency) and that the propagation delay is bellow 10ns.



This string of voltage pulse sources is used to generate 16 bits of repetitive single-ended serial data which is fed to a DFF and an inverting DFF to the transmitter through driving inverters (similarly to the driver test bench in Figure 4-14) to create the differential input of the encryption block.

Figure 4-19 Generating 16-bit Serial Data

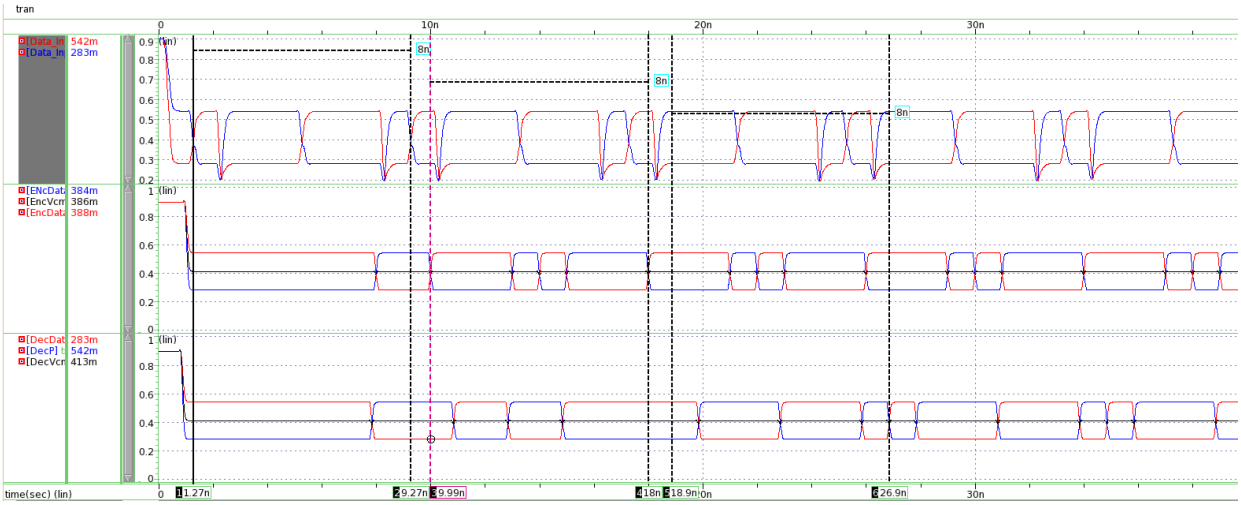


Figure 4-20 Encrypting with 0x00 key at SS

The key inputs can be connected to any kind of key-generation block with an 8-bit output that is synchronized with the data as described by Equation 3-3. For the testbench, the key is fixed by wiring, so the same SIPO clock can be used for the key register as well.

At first, all key inputs are connected to ground to give a 0x00 key for both encryption and which causes all XOR gates to act as buffers for the data, allowing the 4-bit shift to be tested As we can see in Figure 4-20, even under the worst conditions, a 1000 1110 input (in top panel) is encrypted with only a 4-bit shift to be 1110 1000 (in middle panel) and then decrypted back to 1000 1110 in the bottom panel.

Then, both keys are wired to be 0011 0110, and simulation is done over all corners (Figure 4-21) to show that encryption and decryption will work under extreme conditions. This is repeated with a different input data and only at TT but with the same keys as shown in Figure 4-22.
Following this, the decryption key is changed to 0010 0100, while the encryption key remains as before and Figure 4-23 shows that if illegal keys are used for decryption, output is corrupted and cannot be recovered unless both the actual and illegal key are known and the corrupted data is re-encrypted with the illegal key and the then decrypted properly with the original key.

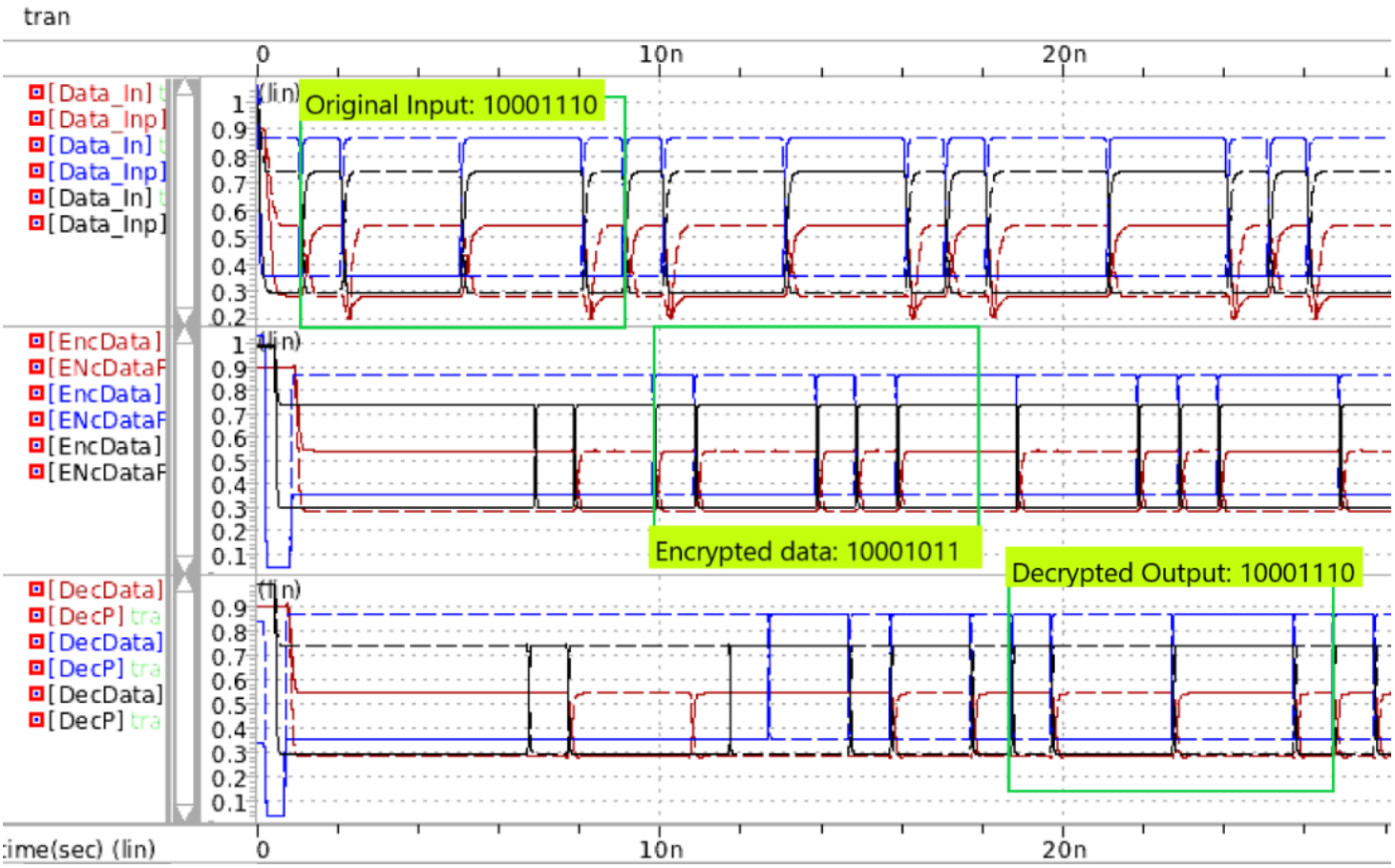


Figure 4-21 Encryption and decryption over PVT corners

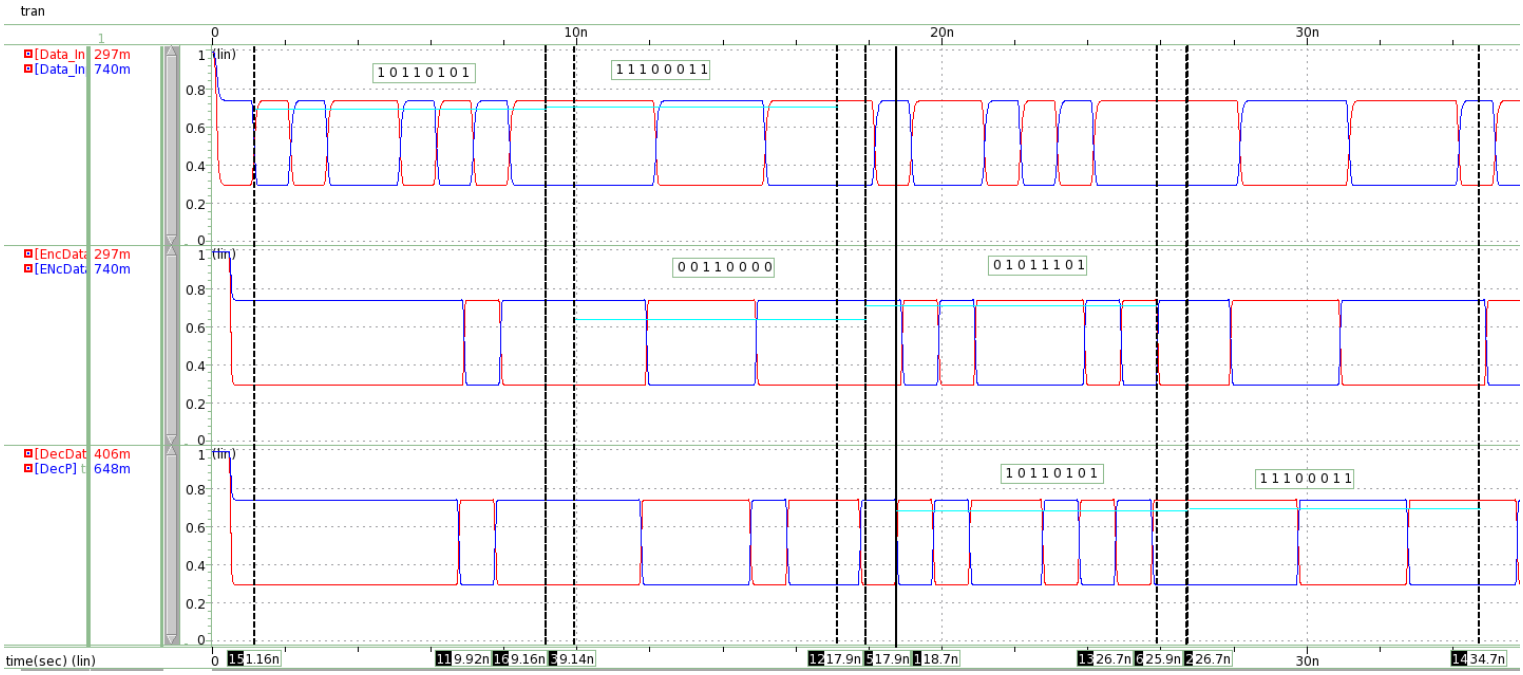


Figure 4-22 Successful decryption

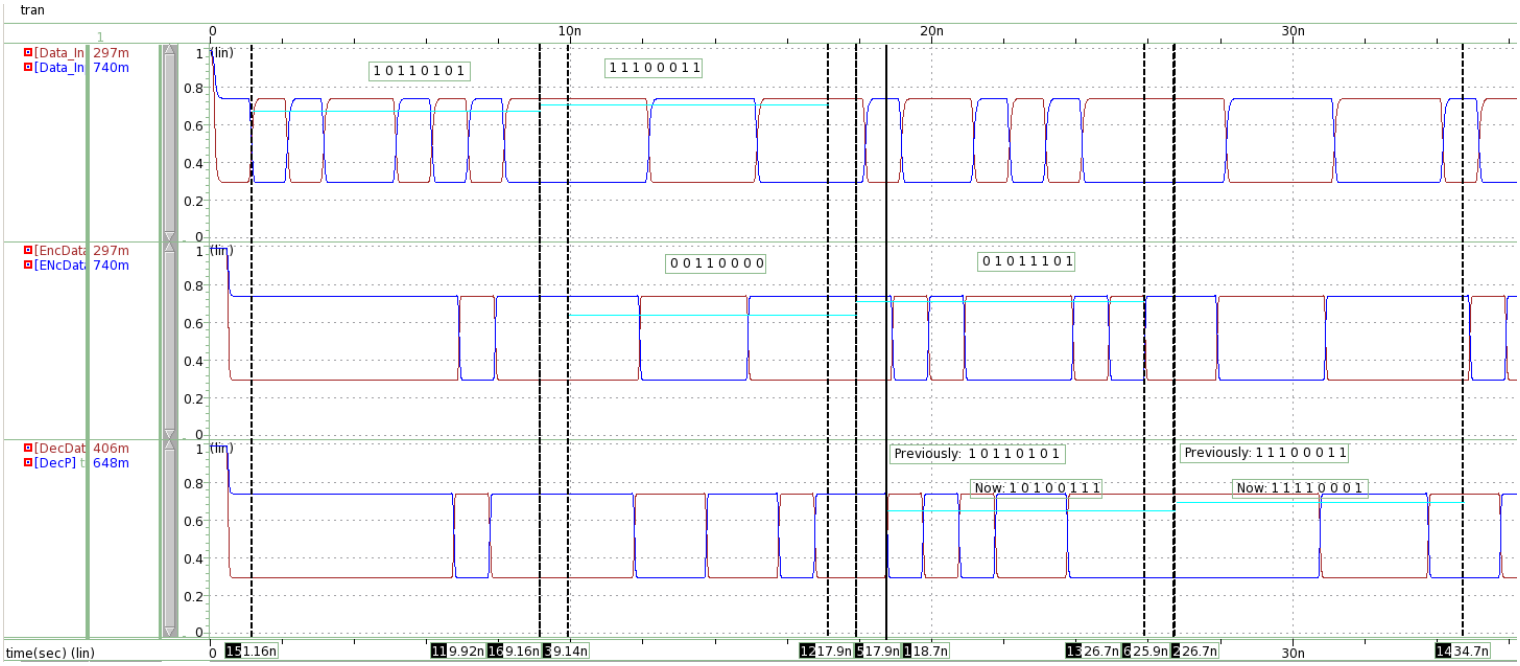


Figure 4-23 Unsuccessful decryption with illegal keys produces corrupted output data

4.4 Analysis of Results

4.4.1 Area

The table below shows a breakdown of how area is used in the layout. When small components are placed next to each other, area can be reduced by combining the N-implants and P-implants of neighboring components and sharing dummy polys. The area reduction is a percentage calculated using the equation below:

$$Area\ Reduction = \frac{(Area\ per\ unit\ \times\ Units\ used) - Total\ area\ in\ block}{(Area\ per\ unit\ \times\ Units\ used)} \times 100\%$$

Equation 4-2 Percentage Area Reduction

However, the total area is still greater than the collective areas of all units since components could not be placed in a way that uses 100% of the area and there is some extension required for routing. Also, the transmitter – which takes up over 47% of the total area –has very little reduction since there is only one unit of it.

Table 4-3 Component Count and Layout Area

	Area per Unit (μm^2)	Units Used	Total Area in Block (μm^2)	Area Reduction
XOR	**2.026** (1.466 x 1.382)	8	**14.418** (10.433 x 1.382)	11.0%
MUX	**2.356** (1.619 x 1.455)	8	**17.736** (11.681 x 1.467)	5.90%
D flipflop	**6.832** (3.286 x 2.079)	24	**102.079** (49.1x2.079)	37.7%
D flipflop w/ inverted output	**7.470** (3.593 x 2.079)	1	**7.470** (3.532x2.079)	0.00%
Driver/transmitter	**225.259** (19.803 x 11.375)	1	**224.065** (19.698 x 11.375)	0.53%
Driving inverters	**10.697** (1.148 x 9.318)	2	**20.267** (2.175 x 9.318)	5.27%
Receiver	**4.755** (1.028 x 4.625)	1	**4.755** (1.028 x 4.625)	0.00%
Complete Block			**476.646** (24.666 x 19.324)	-4%

$$Cost\ Contribution = Cost\ per\ area \times Area = (\$5000/mm^2) \times (477\mu m^2) = \$2.385$$

Equation 4-3 Cost Estimation from Area

The total area is 476.646μm^2 in the end, which will lead an estimated cost contribution of $2.385 per block according to above calculations. For both blocks together, the cost would be doubled to $4.77, which is well within the constraints.

4.4.2 Power Consumption

Table 4-4 below shows the individual and total power consumption across the three corners. As expected, power consumption is highest at FF and lowest at SS.

The transmitter consumes 75%~87% of the total power since it presents a very low equivalent impedance between supply and ground. The driving inverters also consume significantly greater

amounts of power that the rest of the components while the XORs and multiplexers use very little power since their transistors are smallest in number as well as size.

Table 4-4 Power Consumption in μW for each component

	FF	TT	SS
D flipflops in SIPO	146	41.4	24.3
XOR	7.51	1.75	0.790
MUX	5.53	1.95	0.515
D flipflops w/inverted output	157	40.2	25.1
Driver/transmitter	12300	7960	3410
Driving inverters	355	181	200
Receiver	108	18.5	14.1
Complete Block	14600	9000	4460

4.4.3 Propagation Delay

Table 4-5 shows the individual and overall propagation delays over the three corners. As expected, delay is greatest at SS and smallest at FF. The driving inverters contribute the most to the total delay since there is only one inverter in place of five. D flipflops in the key register (PIPO) have smaller delay than other DFFs since their outputs are connected only to the XOR, so they face smaller capacitance loads. The last two DFFs in the PISO have a larger delay since they are loaded by the large driving inverters.

The total delay for the first bit to be encrypted or decrypted is 8.9ns or less when the system operates on 1GHz clocks. The system is not expected to run on clocks faster than 500MHz and the total delay will always be a little more than 8 clock cycles, so the delay will be greater at lower frequencies. However, it is obvious that the total propagation delay due to circuitry is definitely less than 10ns, so it can be said that requirement is met.

Table 4-5 Propagation delay for each component

	SS	TT	FF
D flipflops in PIPO	41.4 ps	24.3 ps	14.6 ps
D flipflops in PISO and SIPO	71.4 ps	37.4 ps	23.6 ps
Inverting DFF and last DFF in PISO	95.5 ps	50.5 ps	31.3 ps
XOR	55.3 ps	28.6 ps	15.7 ps
MUX	71.3 ps	37.1 ps	19.0 ps
Driver/transmitter	87.5 ps	31.8 ps	11.9 ps
Driving inverters	159 ps	91.5 ps	47.7 ps
Receiver	52.8 ps	23.7 ps	10.2 ps
Complete Block (with 1GHz clock)	8.8 ns	8.85 ns	8.9 ns

4.4.4 Achieved Electrical Characteristics of LVDS Interface

Table 4-6 and Table 4-7 show the electrical characteristics achieved by the developed design. The transmitter operates better than expected, but the receiver's sensitivity is less than intended. It should be able to sense input differential swings as small as ±150 mV, but it needs ±200 mV or more to produce a distinguishable single-ended signal. It would have been better to use an architecture similar to Figure 2-5 (b) in page 12; the increase in area would not be noticeable since the transmitter is so large. However, the current receiver design works with the current transmitter design even with the signals attenuated somewhat by 50Ω impedance.

Table 4-6 Achieved Electrical Characteristics of Receiver

Characteristics	Value
Power supply	0.9-1.1 V
Minimum Differential voltage swing	±200 mV
Common mode voltage and Tolerance	500 ±150 mV
Minimum slew rate	1 V/ns

Table 4-7 Electrical Characteristics of Transmitter Output

	SS at 2.5Gb/s	**SS at 1.8Gb/s**	**TT at 1.8Gb/s**	**FF at 1.8Gb/s**
Minimum bit period(ps)	300	500	530	540
Minimum output amplitude(mV)	120	125	225	250
Maximum output amplitude(mV)	180	140	230	265
Common mode(mV)	400	420	520	605
Maximum V_{CM} variation(mV)	±50	±30	±20	±30

4.5 Validation of design requirements within the realistic constrains.

Table 4-8 Validation Summary

	Parameter	Required	Achieved	
Requirements	CMOS technology	32/28 nm	✓	
	Operate at Supply Voltage	1V	✓	
	Maximum Propagation Delay	10ns	8.65ns	✓
	Maximum Input Frequency	500 MHz	1 GHz	✓
	Simulation Software	Synopsys Custom Designer	✓	
	Output for Illegal Keys	Corrupted	Corrupted	✓
Economic constraints	Price	$10	$4.77	✓
Manufacturability and Sustainability constraints	Temperature Range	0 to 70ºC	0 to 70ºC	✓
	$V_{threshold}$ Variation	±5%	±5%	✓

All original requirements have been met within the imposed constraints.

5 Conclusion and Future Work

5.1 Conclusion

This project focuses on hardware security that could be reliable and cheap in the same time. Although the encryption method may seem simple, but there could be many variations where the complexity can be increased with the same design.

The design's compactness surely plays a role in opening a window for a wide variety of improvements by adding extra components that enhance the security even more. Moreover, the compatibility of the design with devices using an LVDS differential interface makes it easier to add the block to other blocks using the same 28 nm CMOS technology.

5.2 Future work

Many enhancements can be made to the design, such as generating random keys that adds significantly to the security. Adding extra stages, like lookup tables, to the encryption process to follow the full AES is also an option; although it would double the area. Furthermore, variable shift could be added without adding much to the area.

Area can also be reduced greatly if only four bits are processed in parallel rather than eight, or if the LVDS interface is not required. However, if the interface is required then the sensitivity of the receiver should be improved, and some effort might be made to decrease the power consumption of the transmitter.

If cost constraints are ignored, a reliable design might process 16 bits at a time with variable shift and an S-box as well as the interface with improved receiver. This might provide good security while remaining within a reasonably small area.

Due to errors faced in the LPE test in the simulation process, calculating accurate capacitances and delays was not possible, but when these errors are overcome in the future, adjustments on the layouts of the components would be made to achieve more optimized and realistic results.

References

[1] Park, Keun Young; Kim, Yong Soo; Kim, Juho, "External memory protection mechanism based on encryption using revocable hardwired key," in *1st IEEE Global Conference on Consumer Electronics*, Tokyo, 2012.

[2] Farhana Sheikh; Leonel Sousa, Circuits and Systems for Security and Privacy, CRC Press, Taylor and Francis Group, 2016, pp. 3-5.

[3] S. Kim, B.-S. Kong, C.-g. Lee, J.-H. Kim, Y.-H. Jun and C. Kim, "A 6-Gbps/pin 4.2mW/pin Half-Deuplex Pseudo-LVDS Transceiver," *IEEE,* pp. 484-487, 2006.

[4] Telecommunications Industry Association(TIA), "Electrical characteristics of low-voltage differential-signaling (LVDS) interface circuits," 2000.

[5] S. V. Kondratenko, "Design and characterization of high-speed CMOS pseudo-LVDS transceivers," *Journal of Physics: Conference Series 675,* 2016.

[6] Rastogi, Rohit; Mishra, Rishabh; Sharma, Sanyukta; Nigam, Anshika; Arya, Pratyush, "Security of Data Transmission Using Logic Gates and Crypt Analysis," in *International Conference on Computing for Sustainable Global Development*, Ghaziabad, India, 2015.

[7] Texas Instruments, The TTL Data Book, standard TTL, Schottky, low power Schottky circuits., 1985.

[8] Arshad Aziz; Nassar Ikram, "An Efficient FPGA Based Sequential Implementation of Advanced Encryption Standard," National University of Sciences & Technology (NUST), Karachi, Pakistan.

[9] Muhammad Farhan Wali; Muhammad Rehan, "Effective Coding and Performance Evaluation of the Rijndael Algorithm (AES)," in *Student Conference on Engineering Sciences and Technology*, Karachi, 2005.

[10] G. Traversi, F. De Canio, V. Liberali and A. Stabile, "Design of LVDS Driver and Receiver in 28 nm CMOS Technology for Associative Memories," in *6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, 2017.

[11] A. Boni, A. Pierazzi and D. Vecchi, "LVDS I/O Interface for Gb/s-per-Pin Operation in 0.35um CMOS," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* vol. 36, pp. 706-711, 2001.

[12] X. Wang, L. Wu and Y. Liu, "Low-power LVDS I/O interface for above 2Gb/s-per-pin operation," *Journal of Electronics (China),* vol. 26, no. 4, p. 525–531, 2009.

[13] M. Chen, J. Silva-Martinez, M. Nix and M. E. Robinson, "Low-Voltage Low-Power LVDS Drivers," *IEEE Journal of Solid-State Circuits,* vol. 40, no. 2, pp. 472 - 479, 2005.

[14] Y. K. C, Y. M. F, P. K. T and Y. T. H, "1.2 Gbps LVDS interface," *Proceeding of the International Symposium on Integrated Circuits ISIC,* 2007.

[15] "32 nm lithography process- WikiChip," WikiChip, 2016. [Online]. Available: https://en.wikichip.org/wiki/32_nm_lithography_process#. [Accessed 19 Dec 2018].

[16] "General prices," Europractice, 2012. [Online]. Available: http://www.europractice-ic.com/docs/190415_MPW2019-general-v5.0.pdf. [Accessed 19 December 2018].

[17] Cypress Semiconductor Corporation, "A Comparison of CML and LVDS for High-Speed Serial Links," Cypress Semiconductor Corporation, 2002.

[18] B. Jacob, "ENEE 359a Digital VLSI Design: Transistor Sizing & Logical Effort," University of Maryland ECE Dept., 2004. [Online]. Available: https://ece.umd.edu/class/enee359a/enee359a-sizing.pdf. [Accessed 28 April 2019].

[19] "32nm BSIM4 model card for bulk CMOS," Predictive Technology Model, 30 September 2005. [Online]. Available: http://ptm.asu.edu/modelcard/32nm_bulk.pm. [Accessed 28 April 2019].

Appendix A: Encryption Block Netlist

```
*  Generated for: HSPICE
*  Design library name: GPpComponents
*  Design cell name: hwEncryption_DSYgradproj
*  Design view name: schematic


.option PARHIER = LOCAL
.option PORT_VOLTAGE_SCALE_TO_2X = 1

.option ARTIST=2 PSF=2
.temp 25

*Custom Compiler Version N-2017.12-SP1-4
*Thu May  9 04:51:18 2019


********************************************************************************
* Library          : GPpComponents
* Cell             : DFF_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt dff_dsygradproj avdd avss ck d q
xm28 net107 d avss avss n105 w=0.25u l=0.03u nf=1 m=1
xm25 q net102 avss avss n105 w=0.42u l=0.03u nf=1 m=1
xm18 ckd ckp avss avss n105 w=0.4u l=0.03u nf=1 m=1
xm11 net98 ckd net97 avss n105 w=0.3u l=0.03u nf=1 m=1
xm10 net72 ckp avss avss n105 w=0.3u l=0.03u nf=1 m=1
xm9 net97 net102 net72 avss n105 w=0.3u l=0.03u nf=1 m=1
xm8 net102 net97 avss avss n105 w=0.35u l=0.03u nf=1 m=1
xm16 ckp ck avss avss n105 w=0.4u l=0.03u nf=1 m=1
xm3 net107 ckp net78 avss n105 w=0.3u l=0.03u nf=1 m=1
xm22 net89 ckd avss avss n105 w=0.3u l=0.03u nf=1 m=1
xm1 net78 net98 net89 avss n105 w=0.3u l=0.03u nf=1 m=1
xm0 net98 net78 avss avss n105 w=0.35u l=0.03u nf=1 m=1
xm19 ckd ckp avdd avdd p105 w=0.74u l=0.03u nf=1 m=1
xm26 q net102 avdd avdd p105 w=0.8u l=0.03u nf=1 m=1
xm27 net107 d avdd avdd p105 w=0.45u l=0.03u nf=1 m=1
xm17 ckp ck avdd avdd p105 w=0.74u l=0.03u nf=1 m=1
xm15 net97 net102 net69 avdd p105 w=0.3u l=0.03u nf=1 m=1
xm14 net69 ckd avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm13 net102 net97 avdd avdd p105 w=0.7u l=0.03u nf=1 m=1
xm12 net98 ckp net97 avdd p105 w=0.4u l=0.03u nf=1 m=1
xm7 net78 net98 net94 avdd p105 w=0.3u l=0.03u nf=1 m=1
xm6 net94 ckp avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm5 net98 net78 avdd avdd p105 w=0.7u l=0.03u nf=1 m=1
xm4 net107 ckd net78 avdd p105 w=0.4u l=0.03u nf=1 m=1
.ends dff_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : SIPO
* View             : schematic
```

```
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt sipo avdd avss po_0 po_1 po_2 po_3 po_4 po_5 po_6 po_7 serialdatain
+ clksipo
xi2 avdd avss clksipo po_6 po_5 dff_dsygradproj
xi1 avdd avss clksipo po_7 po_6 dff_dsygradproj
xi0 avdd avss clksipo serialdatain po_7 dff_dsygradproj
xi3 avdd avss clksipo po_5 po_4 dff_dsygradproj
xi6 avdd avss clksipo po_2 po_1 dff_dsygradproj
xi5 avdd avss clksipo po_3 po_2 dff_dsygradproj
xi4 avdd avss clksipo po_4 po_3 dff_dsygradproj
xi7 avdd avss clksipo po_1 po_0 dff_dsygradproj
.ends sipo

********************************************************************************
* Library          : GPpComponents
* Cell             : DrivingInvs
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt drivinginvs avdd avss in out
xm4 out in avss avss n105 w=2.1u l=0.03u nf=2 m=1
xm0 out in avss avss n105 w=2.1u l=0.03u nf=2 m=1
xm2 out in avss avss n105 w=2.1u l=0.03u nf=2 m=1
xm30 out in avss avss n105 w=2.1u l=0.03u nf=2 m=1
xm5 out in avdd avdd p105 w=4u l=0.03u nf=2 m=1
xm3 out in avdd avdd p105 w=4u l=0.03u nf=2 m=1
xm1 out in avdd avdd p105 w=4u l=0.03u nf=2 m=1
xm29 out in avdd avdd p105 w=4u l=0.03u nf=2 m=1
.ends drivinginvs

********************************************************************************
* Library          : GPpComponents
* Cell             : XOR_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt xor_dsygradproj a avdd avss b out
xm8 bp b avss avss n105 w=0.15u l=0.03u nf=1 m=1
xm18 ap a avss avss n105 w=0.15u l=0.03u nf=1 m=1
xm3 out b net15 avss n105 w=0.1u l=0.03u nf=1 m=1
xm2 out ap net11 avss n105 w=0.1u l=0.03u nf=1 m=1
xm1 net15 a avss avss n105 w=0.1u l=0.03u nf=1 m=1
xm0 net11 bp avss avss n105 w=0.1u l=0.03u nf=1 m=1
xm9 bp b avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm19 ap a avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm7 net31 a avdd avdd p105 w=0.2u l=0.03u nf=1 m=1
xm6 net27 b avdd avdd p105 w=0.2u l=0.03u nf=1 m=1
xm5 out bp net31 avdd p105 w=0.2u l=0.03u nf=1 m=1
xm4 out ap net27 avdd p105 w=0.2u l=0.03u nf=1 m=1
.ends xor_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : invDFF_DSYgradproj
```

```
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt invdff_dsygradproj avdd avss ck d q
xm30 net110 d avss avss n105 w=0.25u l=0.03u nf=1 m=1
xm28 net107 net110 avss avss n105 w=0.25u l=0.03u nf=1 m=1
xm25 q net102 avss avss n105 w=0.42u l=0.03u nf=1 m=1
xm18 ckd ckp avss avss n105 w=0.4u l=0.03u nf=1 m=1
xm11 net98 ckd net97 avss n105 w=0.3u l=0.03u nf=1 m=1
xm10 net72 ckp avss avss n105 w=0.3u l=0.03u nf=1 m=1
xm9 net97 net102 net72 avss n105 w=0.3u l=0.03u nf=1 m=1
xm8 net102 net97 avss avss n105 w=0.35u l=0.03u nf=1 m=1
xm16 ckp ck avss avss n105 w=0.4u l=0.03u nf=1 m=1
xm3 net107 ckp net78 avss n105 w=0.3u l=0.03u nf=1 m=1
xm22 net89 ckd avss avss n105 w=0.3u l=0.03u nf=1 m=1
xm1 net78 net98 net89 avss n105 w=0.3u l=0.03u nf=1 m=1
xm0 net98 net78 avss avss n105 w=0.35u l=0.03u nf=1 m=1
xm29 net110 d avdd avdd p105 w=0.45u l=0.03u nf=1 m=1
xm19 ckd ckp avdd avdd p105 w=0.74u l=0.03u nf=1 m=1
xm26 q net102 avdd avdd p105 w=0.8u l=0.03u nf=1 m=1
xm27 net107 net110 avdd avdd p105 w=0.45u l=0.03u nf=1 m=1
xm17 ckp ck avdd avdd p105 w=0.74u l=0.03u nf=1 m=1
xm15 net97 net102 net69 avdd p105 w=0.3u l=0.03u nf=1 m=1
xm14 net69 ckd avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm13 net102 net97 avdd avdd p105 w=0.7u l=0.03u nf=1 m=1
xm12 net98 ckp net97 avdd p105 w=0.4u l=0.03u nf=1 m=1
xm7 net78 net98 net94 avdd p105 w=0.3u l=0.03u nf=1 m=1
xm6 net94 ckp avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm5 net98 net78 avdd avdd p105 w=0.7u l=0.03u nf=1 m=1
xm4 net107 ckd net78 avdd p105 w=0.4u l=0.03u nf=1 m=1
.ends invdff_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : MUX_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt mux_dsygradproj avdd avss in_0 in_1 s_0 out
xm11 s0_p s_0 avss avss n105 w=0.15u l=0.03u nf=1 m=1
xm5 net55 in_1 net54 avss n105 w=0.1u l=0.03u nf=1 m=1
xm4 net55 in_0 net18 avss n105 w=0.1u l=0.03u nf=1 m=1
xm3 out net55 avss avss n105 w=0.15u l=0.03u nf=1 m=1
xm2 net18 s0_p avss avss n105 w=0.1u l=0.03u nf=1 m=1
xm1 net54 s_0 avss avss n105 w=0.1u l=0.03u nf=1 m=1
xm12 s0_p s_0 avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm10 out net55 avdd avdd p105 w=0.3u l=0.03u nf=1 m=1
xm9 net55 s0_p net56 avdd p105 w=0.2u l=0.03u nf=1 m=1
xm8 net56 in_1 avdd avdd p105 w=0.2u l=0.03u nf=1 m=1
xm7 net55 in_0 net56 avdd p105 w=0.2u l=0.03u nf=1 m=1
xm6 net56 s_0 avdd avdd p105 w=0.2u l=0.03u nf=1 m=1
.ends mux_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : PISO
```

```
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt piso avdd avss dffclk in0 in1 in2 in3 in4 in5 in6 in7 load_shiftp sout
+ sout_p
xi21 avdd avss dffclk net146 sout_p invdff_dsygradproj
xi13 avdd avss dffclk net111 net118 dff_dsygradproj
xi14 avdd avss dffclk net109 net120 dff_dsygradproj
xi15 avdd avss dffclk net107 net122 dff_dsygradproj
xi16 avdd avss dffclk net104 net125 dff_dsygradproj
xi17 avdd avss dffclk net102 net126 dff_dsygradproj
xi18 avdd avss dffclk net100 net128 dff_dsygradproj
xi19 avdd avss dffclk net98 net114 dff_dsygradproj
xi20 avdd avss dffclk net146 sout dff_dsygradproj
xi5 avdd avss net114 in0 load_shiftp net146 mux_dsygradproj
xi6 avdd avss avss in7 load_shiftp net111 mux_dsygradproj
xi7 avdd avss net118 in6 load_shiftp net109 mux_dsygradproj
xi8 avdd avss net120 in5 load_shiftp net107 mux_dsygradproj
xi9 avdd avss net122 in4 load_shiftp net104 mux_dsygradproj
xi10 avdd avss net125 in3 load_shiftp net102 mux_dsygradproj
xi11 avdd avss net126 in2 load_shiftp net100 mux_dsygradproj
xi12 avdd avss net128 in1 load_shiftp net98 mux_dsygradproj
.ends piso

********************************************************************************
* Library          : GPpComponents
* Cell             : Diff2SingCONV_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt diff2singconv_dsygradproj avdd avss vinh vinl vout
xm13 vout vinl net72 avss n105 w=0.8u l=0.03u nf=1 m=1
xm12 vout vinh net72 avss n105 w=0.1u l=0.03u nf=1 m=1
xm1 net72 avdd avss avss n105 w=2.1u l=0.03u nf=1 m=1
xm9 net78 avss avdd avdd p105 w=1.7u l=0.03u nf=1 m=1
xm14 vout vinh net78 avdd p105 w=0.2u l=0.03u nf=1 m=1
xm15 vout vinl net78 avdd p105 w=3u l=0.03u nf=1 m=1
.ends diff2singconv_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : KeyReg
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt keyreg avdd avss ki_0 ki_1 ki_2 ki_3 ki_4 ki_5 ki_6 ki_7 ko_0 ko_1 ko_2
+ ko_3 ko_4 ko_5 ko_6 ko_7 wrpulse
xi0 avdd avss wrpulse ki_0 ko_0 dff_dsygradproj
xi1 avdd avss wrpulse ki_1 ko_1 dff_dsygradproj
xi2 avdd avss wrpulse ki_2 ko_2 dff_dsygradproj
xi3 avdd avss wrpulse ki_3 ko_3 dff_dsygradproj
xi4 avdd avss wrpulse ki_4 ko_4 dff_dsygradproj
xi5 avdd avss wrpulse ki_5 ko_5 dff_dsygradproj
xi6 avdd avss wrpulse ki_6 ko_6 dff_dsygradproj
xi7 avdd avss wrpulse ki_7 ko_7 dff_dsygradproj
```

```
.ends keyreg

********************************************************************************
* Library          : GPpComponents
* Cell             : 50ohmRES_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt _50ohmres_dsygradproj p1 p2
xr0 p1 p2 rnpoly w=0.374u l=0.372u m=1
.ends _50ohmres_dsygradproj

********************************************************************************
* Library          : GPpComponents
* Cell             : LVDSdriverPAR_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
.subckt lvdsdriverpar_dsygradproj avdd avss vcm vdiff vdiff_p vin vin_p
xi492 vdiff vcm _50ohmres_dsygradproj
xi491 vcm vdiff_p _50ohmres_dsygradproj
xm415 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm414 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm413 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm412 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm411 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm410 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm409 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm408 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm407 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm406 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm405 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm404 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm403 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm402 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm401 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm400 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm399 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm398 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm397 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm396 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm395 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm394 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm393 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm392 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm391 vdiff_p vin_p net202 avss n105 w=4u l=0.03u nf=2 m=1
xm390 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm389 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm388 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm387 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm386 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm385 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm384 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm383 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm382 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm381 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
```

xm380 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm379 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm378 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm377 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm376 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm375 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm374 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm373 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm372 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm371 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm370 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm369 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm368 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm366 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm364 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm365 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm367 vdiff vin net202 avss n105 w=4u l=0.03u nf=2 m=1
xm356 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm363 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm359 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm357 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm361 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm362 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm360 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm358 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm355 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm343 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm342 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm353 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm344 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm345 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm346 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm352 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm349 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm348 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm354 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm341 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm347 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm350 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm351 net202 avdd avss avss n105 w=4u l=0.03u nf=2 m=1
xm490 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm489 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm488 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm487 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm486 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm485 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm484 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm483 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm482 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm481 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm480 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm479 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm478 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm477 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm476 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm475 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm474 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm473 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1

```
xm472 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm471 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm470 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm469 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm468 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm467 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm466 vdiff vin net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm465 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm464 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm463 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm462 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm461 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm460 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm459 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm458 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm457 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm456 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm455 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm454 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm453 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm452 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm451 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm450 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm449 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm448 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm447 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm446 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm445 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm444 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm443 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm442 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm441 net237 avss avdd avdd p105 w=6.34u l=0.03u nf=2 m=1
xm440 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm439 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm438 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm437 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm436 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm435 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm434 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm433 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm432 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm431 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm430 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm429 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm428 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm427 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm426 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm425 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm424 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm423 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm422 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm421 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm420 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm419 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm418 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm417 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
xm416 vdiff_p vin_p net237 avdd p105 w=6.34u l=0.03u nf=2 m=1
.ends lvdsdriverpar_dsygradproj
```

```
********************************************************************************
* Library          : GPpComponents
* Cell             : hwEncryption_DSYgradproj
* View             : schematic
* View Search List : hspice hspiceD schematic spice veriloga
* View Stop List   : hspice hspiceD
********************************************************************************
xi74 vdd vss net617 net595 net593 net591 net589 net587 net585 net583 data_in
+ sipoclk sipo
xi98 vdd vss net719 net732 drivinginvs
xi97 vdd vss net715 net731 drivinginvs
xi94 net576 vdd vss net583 net697 xor_dsygradproj
xi87 net615 vdd vss net617 net702 xor_dsygradproj
xi89 net579 vdd vss net593 net695 xor_dsygradproj
xi90 net580 vdd vss net591 net696 xor_dsygradproj
xi88 net578 vdd vss net595 net701 xor_dsygradproj
xi93 net575 vdd vss net585 net698 xor_dsygradproj
xi91 net581 vdd vss net589 net700 xor_dsygradproj
xi92 net574 vdd vss net587 net699 xor_dsygradproj
xi46 vdd vss pisoclk net700 net699 net698 net697 net702 net701 net695 net696
+ piso_l_shp net719 net715 piso
xi96 vdd vss diffin1 diffin2 data_in diff2singconv_dsygradproj
xi76 vdd vss key0 key1 key2 key3 key4 key5 key6 key7 net615 net578 net579 net580
+  net581 net574 net575 net576 keyclk keyreg
xi99 vdd vss vcm diffout1 diffout2 net732 net731 lvdsdriverpar_dsygradproj


.option opfile=1 split_dp=1
.option probe=1


.end
```