Cryptographic Processors

Project Report

EC383 Mini Project in VLSI Design

(January 2023 - April 2023)

By

Trivendra Tiwari (201EC163)

Bhimaraddy Yarabandi (201EC170)

Nihar Gowda S (201EE236)

under the guidance of

Dr Ramesh Kini M

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

Cryptography is the science of secure communication. It involves techniques for protecting information from unauthorized access or modification and a cryptographic processor is a specialized chip designed to perform cryptographic operations securely and efficiently. These processors are used in a wide range of applications, including secure communication, digital signatures, secure storage, and authentication.

A cryptographic processor performs cryptographic operations such as encryption and decryption, and has specialized memory for storing cryptographic keys and other sensitive information. They can also provide additional security features such as key storage and management, random number generation, and secure boot.The processors are specifically designed to provide high-speed, low-latency performance for cryptographic tasks, which are essential in many security-critical applications.

SHA-256, AES-128,192,256 and DES are all cryptographic algorithms that are commonly used in conjunction with cryptographic processors.

SHA-256

Literature survey

SHA-256 is a set of cryptographic hash function designed by the United States National Security Agency (NSA) and first published in 2001.They are built using the Merkle–Damgård construction, from a one-way compression function itself built using the Davies–Meyer structure from a specialized block cipher.

H. Gilbert at el. analyzed the security of SHA-256 against collision attacks and provided some insight into the security properties of the basic building blocks of the structure,and concluded differential and linear attacks also don't apply on the underlying structure.

Project Details

The SHA-256 (Secure Hash Algorithm 256) is a cryptographic hash function that generates a fixed-length output of 256 bits (32 bytes) regardless of the input size.The resulting hash is considered virtually impossible to reverse-engineer or predict, which makes it a valuable tool for securing sensitive data.The SHA-256 algorithm is widely used in various applications, including digital signatures, password storage, and blockchain technology. It has become the preferred hashing algorithm for many organizations due to its high level of security, reliability, and speed.

The input message is padded to ensure that its length is a multiple of 512 bits. The padding is done in such a way that the last 64 bits represent the original length of the message.The algorithm uses a 256-bit initialization vector (IV) to set the initial state of the hash function. The padded message is split into 512-bit blocks, and the algorithm processes each block in turn. For each block, the algorithm generates a message schedule consisting of 64 32-bit words. The algorithm compresses each block of the message using the message schedule and the current state of the hash function. The compression involves a series of logical and arithmetic operations, including bitwise operations, 32 bit modular addition, and cyclic shifts.After processing all blocks of the message, the final state of the hash function is used to produce the 256-bit output digest.

Implementation

The SHA256 algorithm is implemented on Quartus Prime with 4 verilog modules

1. Padding and Parsing:

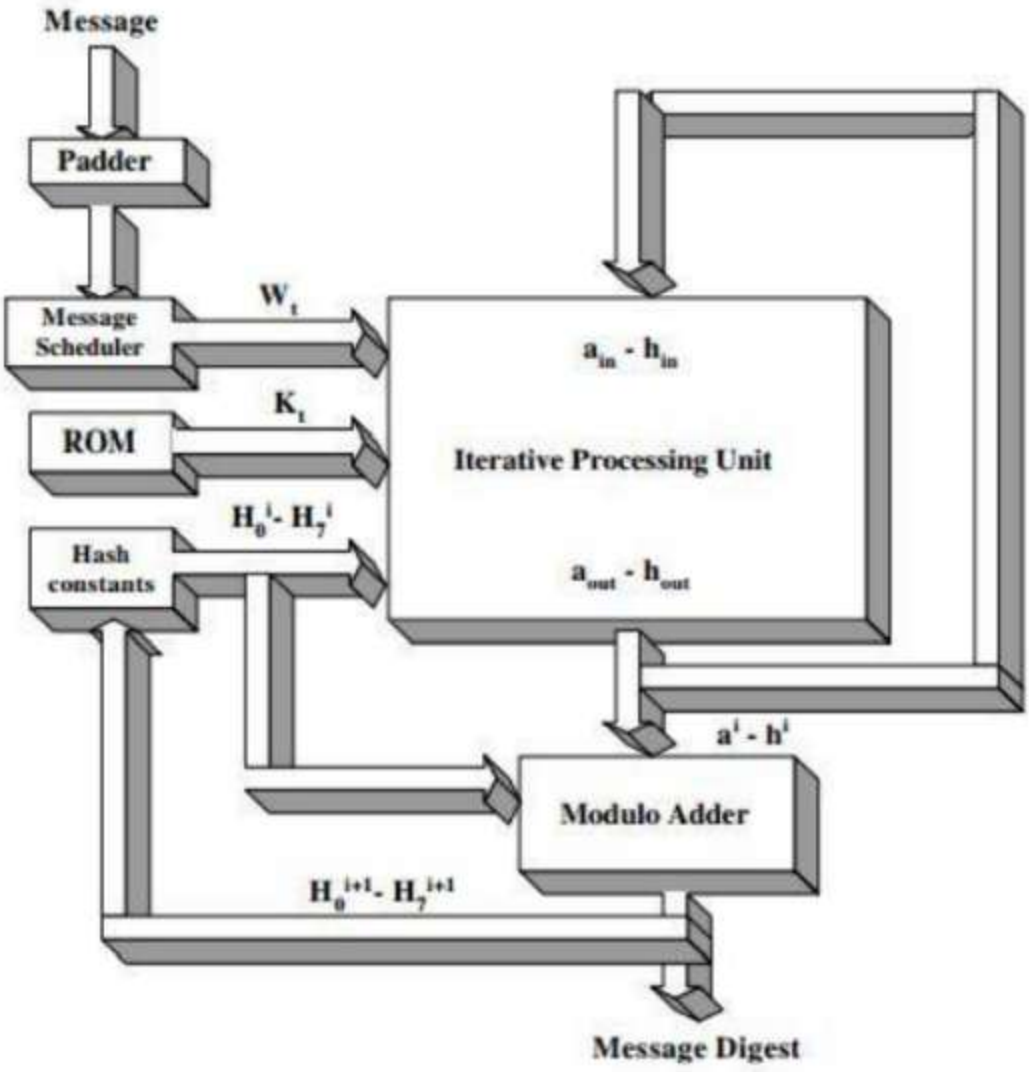
The module is appended with "1" and padded with zeros after the input data up to the length 448 modulo 512..The last 64 bits holds the size of input data .This module also issues the "padding_done" signal to the other modules that padding and parsing has been done.

2. Message Scheduler

In message scheduler we implement the 0-15 iterations for 32bit 'W' words which is all the Block_1 data obtained from padding (32*16=512 bits).After 16 'W' words we perform iterations from already received 16 words to create further 48 'W' words.

For total 64 iterations:
Message schedule, {Wt}:
for $0 \leq t \leq 15$ Wt =Mt
for $16 \leq t \leq 63$ $Wt = \sigma 1 (Wt-2) + Wt-7 + \sigma 0 (W\ t-15) + Wt-16$
semation_0(x) <= (x ROTR 7) ^ (x ROTR 18) ^ (x SHR 3)
semation_1(x) <= (x ROTR 17) ^ (x ROTR 19) ^ (x SHR 10)

Block Diagram



3. Iterative Processing

In iterative processing we initialize 8 registers each of 32bit
a, b, c, d, e, f, g, h with initial values equal to 8 intermediate initial values,
H0 = 32'h6a09e667
H1 = 32'hbb67ae85
H2 = 32'b3c6ef372

H3 = 32'ha54ff53a
H4 = 32'h510e527f
H5 = 32'h9b05688c
H6 = 32'h1f83d9ab
H7 = 32'h5be0cd19
and perform following compression functions:
for all 64 iterations:
semation_1 <= (e ROTR 6) ^ (e ROTR 11) ^ (e ROTR 25)
ch <= (e and f) ^ ((~ e) and g)
T1 <= h +semation_1 + ch + k[n] + W[n]
Semation_0 = (a ROTR 2) ^ (a ROTR 13) ^ (a ROTR 22)
maj = (a and b) ^ (a and c) ^ (b and c)
T2 = semation_0 + maj
h <= g
g <= f
f <= e
e <= d + T1
d <= c
c <= b
b <= a
a <= T1 + T2
Where K's values are pre-defined by FIPS (Federal Information & Processing Standards)

4. Message Digest

In message digest we perform addition of Initial values of "H" and values of 8 iteration registers for Block_1. After addition we concatenate values of 8 'H' each 32bit giving the out of 256 bits Hash.

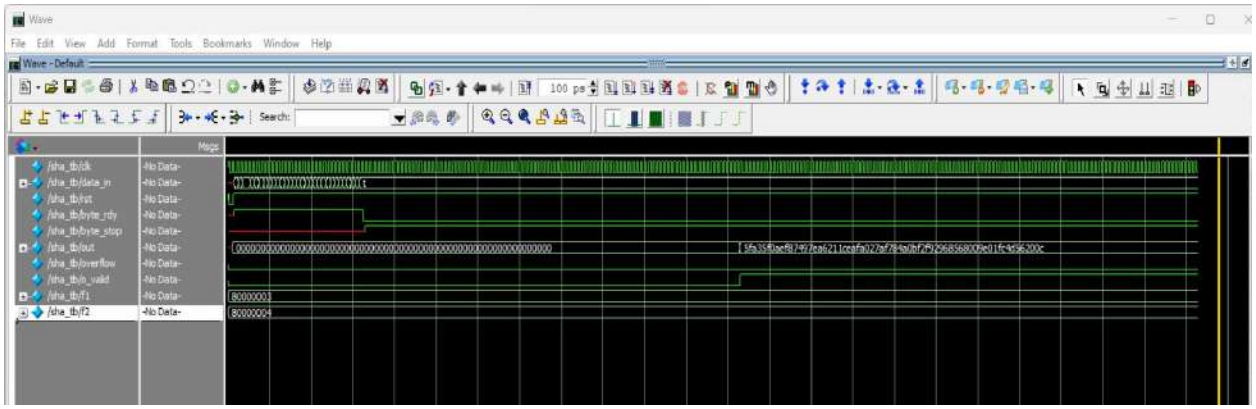
Results

The input data is

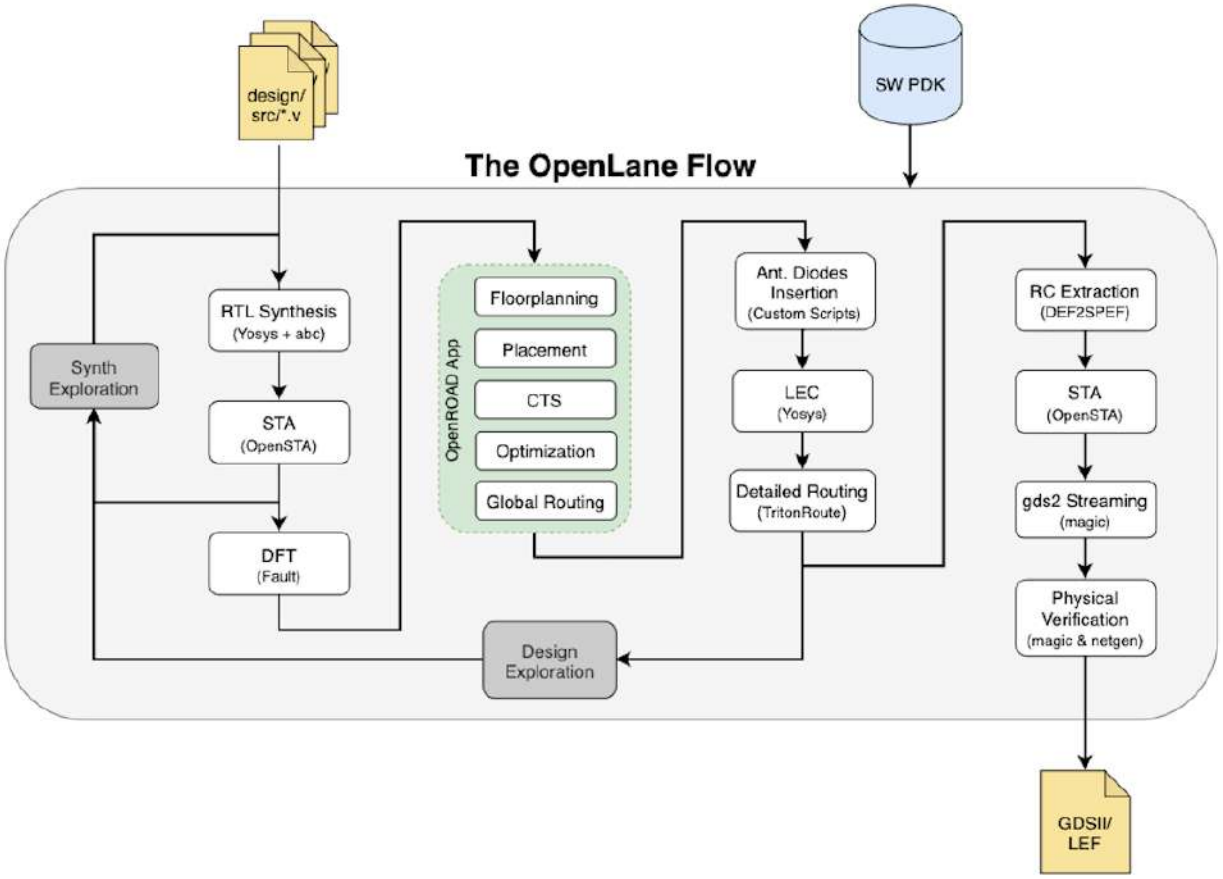
“**hello we are doing vlsi project**”

The 256bit output hash code is

“**5fa35f0aef87497ea6211ceafa027af784a0bf2f92968568009e01fc4d56200c**”



ASIC Design of SHA256 using OpenLane



Synthesis -

Openlane uses yosys open source tool for netlist synthesis and abc open source tool for technology mapping.

In the OpenLane ASIC design flow, synthesis is the step where a Register Transfer Level (RTL) design written in a hardware description language (HDL) like Verilog or VHDL is converted into a gate-level netlist. The synthesized netlist contains a list of cells, each of which represents a logic gate in the design, along with their connections. The synthesis process also optimizes the design for timing, power, and area, and generates timing reports that help in further analysis and optimization. OpenLane uses the Yosys synthesis tool for RTL synthesis, which can also perform technology mapping and optimize for area and timing constraints. After synthesis, the netlist is ready for the next step in the ASIC design flow, which is placement and routing.

Results

1) sdf file : sdf file is Standard Delay Format file . Header section of .sdf file contains several information like shown below

```
(DELAYFILE
 (SDFVERSION "3.0")
 (DESIGN "SHA")
 (DATE "Sun Mar 26 10:40:35 2023")
 (VENDOR "Parallax")
 (PROGRAM "STA")
 (VERSION "2.3.2")
 (DIVIDER .)
 (VOLTAGE 1.800::1.800)
 (PROCESS "1.000::1.000")
 (TEMPERATURE 25.000::25.000)
 (TIMESCALE 1ns)
```

Voltage = 1.8v(delay depends on supply voltage).

Timescale = 1ns (unit of time)

Date and time of report creation .

The file also contains the information about the interconnect and standard cell delay.

```
(INTERCONNECT data_in[0] _10356_.A1 (0.020:0.020:0.020) (0.009:0.009:0.009))
(INTERCONNECT data_in[0] _10408_.B (0.020:0.020:0.020) (0.009:0.009:0.009))
(INTERCONNECT data_in[0] _13467_.A1 (0.020:0.020:0.020) (0.009:0.009:0.009))
(INTERCONNECT data_in[1] _10361_.A1 (0.020:0.020:0.020) (0.009:0.009:0.009))
(INTERCONNECT data_in[1] _10413_.B (0.020:0.020:0.020) (0.009:0.009:0.009))
(INTERCONNECT data_in[1] _13498_.A1 (0.020:0.020:0.020) (0.009:0.009:0.009))
```

```
(CELL
 (CELLTYPE "sky130_fd_sc_hd__buf_1")
 (INSTANCE _10715_)
 (DELAY
  (ABSOLUTE
   (IOPATH A X (0.302:0.302:0.302) (0.204:0.204:0.204))
  )
 )
)
```

The above picture represents the delay of the sky130nm buffer cell of drive strength 1x.

2) The .v file contains the information about the technology mapping done by tool abc and this represents the gate level netlist in which gates will be represented by the standard cells.

The report section of synthesis contains the information about the number of wires used,number of standard cells of each type, core area etc.

```
Chip area for module '\SHA': 125091.222400
```

The unit of area is square microns.

And also contains the information about the skew, worst case slack, worst case negative slack etc information.

Floorplan -

Openlane uses following tools to do the floorplan

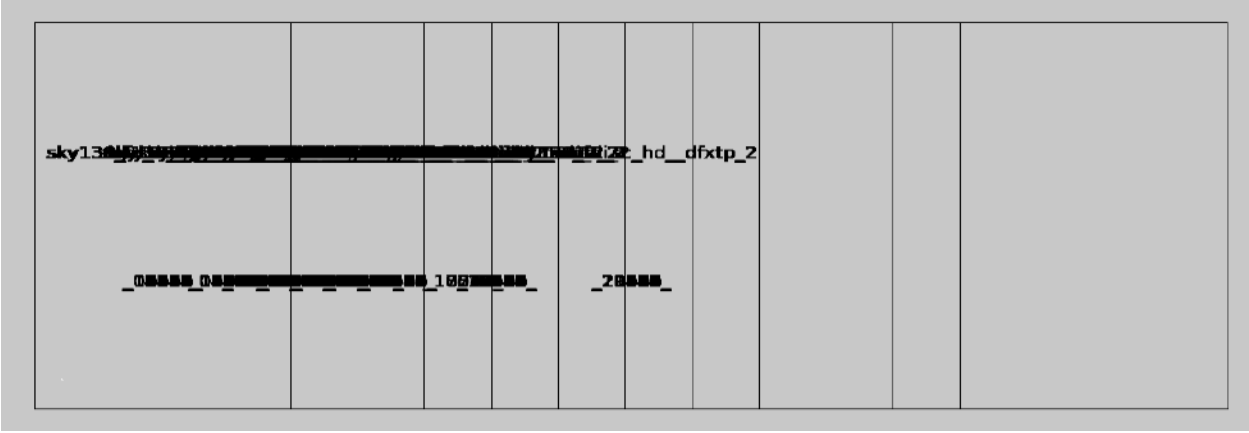
1. init_fp - Defines the core area for the macro as well as the rows (used for placement) and the tracks (used for routing)

2. ioplacer - Places the macro input and output ports

3. pdngen - Generates the power distribution network

4. tapcell - Inserts well tap and decap cells in the floorplan

1. init_fp: This is short for "initial floor plan". In chip design, a floor plan refers to the physical layout of the chip's components, including the placement of macros (large functional units, such as processors or memory blocks) and the routing of connections between them. The init_fp step defines the core area of the chip where the macros will be placed, as well as the rows (horizontal partitions) and tracks (vertical partitions) that will be used for routing.

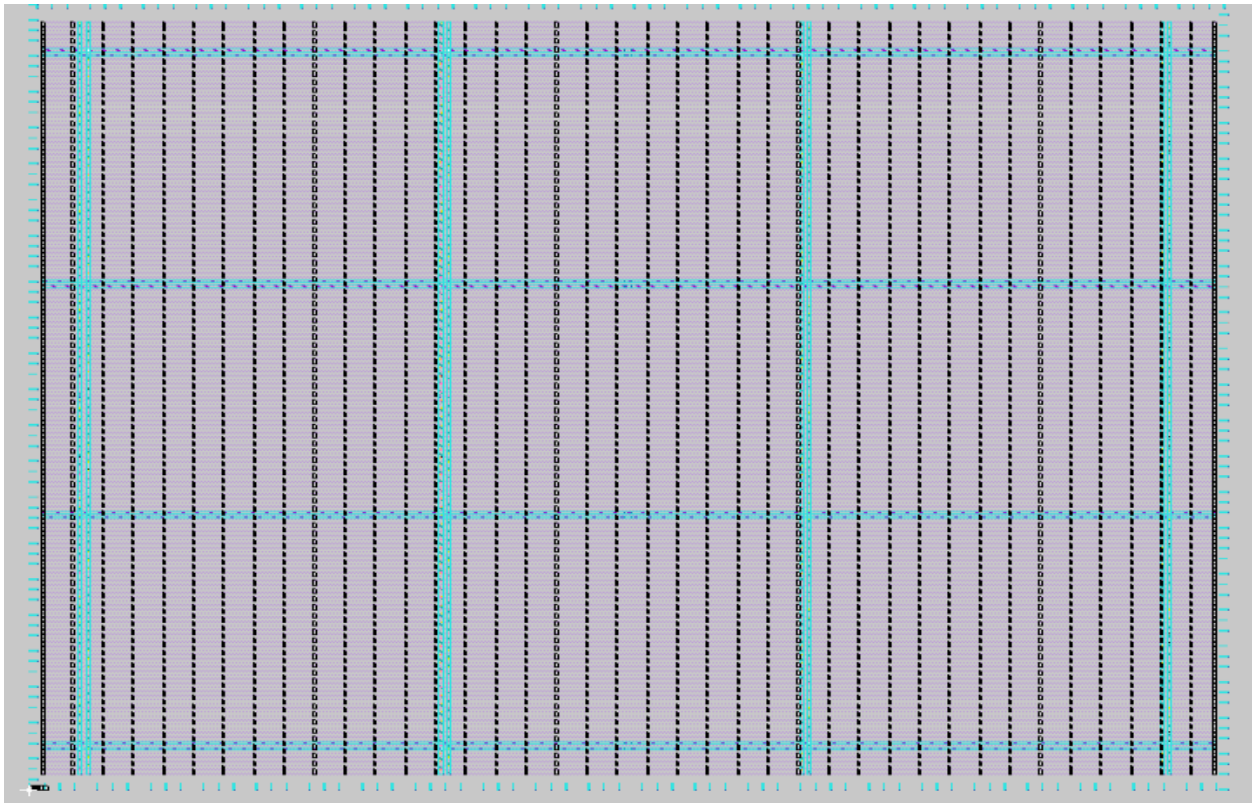


2. ioplacer: Once the core area and routing resources have been defined, the next step is to place the input and output ports of each macro. This is done by the ioplacer tool, which

determines the optimal location for each port based on factors such as signal timing, power consumption, and physical proximity to other components.

3. pdngen: After the macros and their input/output ports have been placed, the next step is to generate the power distribution network. This network ensures that each component receives the correct amount of power, while minimizing power consumption and minimizing noise that can interfere with signal quality. The pdngen tool creates the network by placing power and ground pins throughout the chip and connecting them with metal routing layers.

4. tapcell: Finally, the tapcell tool inserts well tap and decap cells into the floorplan. Well tap cells provide a connection between the substrate of the chip and the power supply, which helps to prevent parasitic effects that can degrade signal quality. Decap cells are used to smooth out fluctuations in the power supply, which can occur due to changes in the activity level of different parts of the chip. The tapcell tool determines the optimal locations for these cells based on the layout of the other components in the chip.

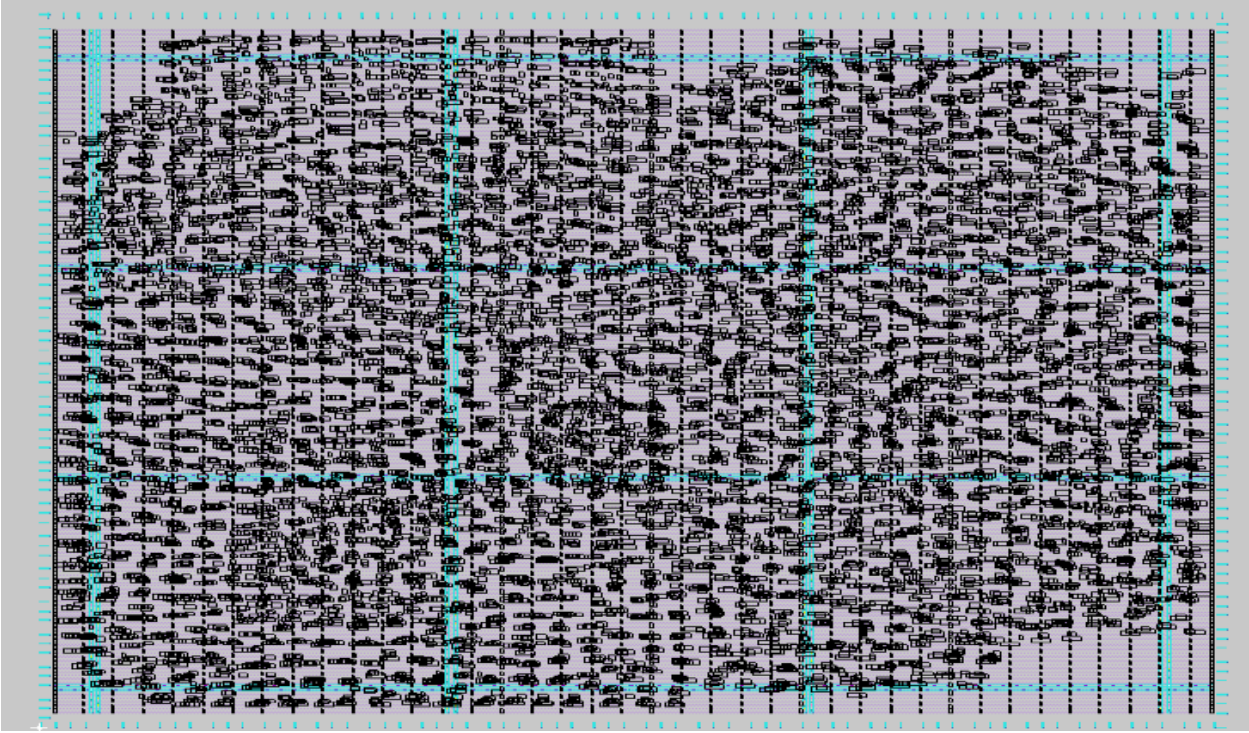


Placement -

Placement is a critical step in the physical design of a chip, where the position of each component is determined to optimize the chip's performance and minimize its area and power consumption. There are several tools used in the placement process, including RePLace, Resizer, and OpenDP.

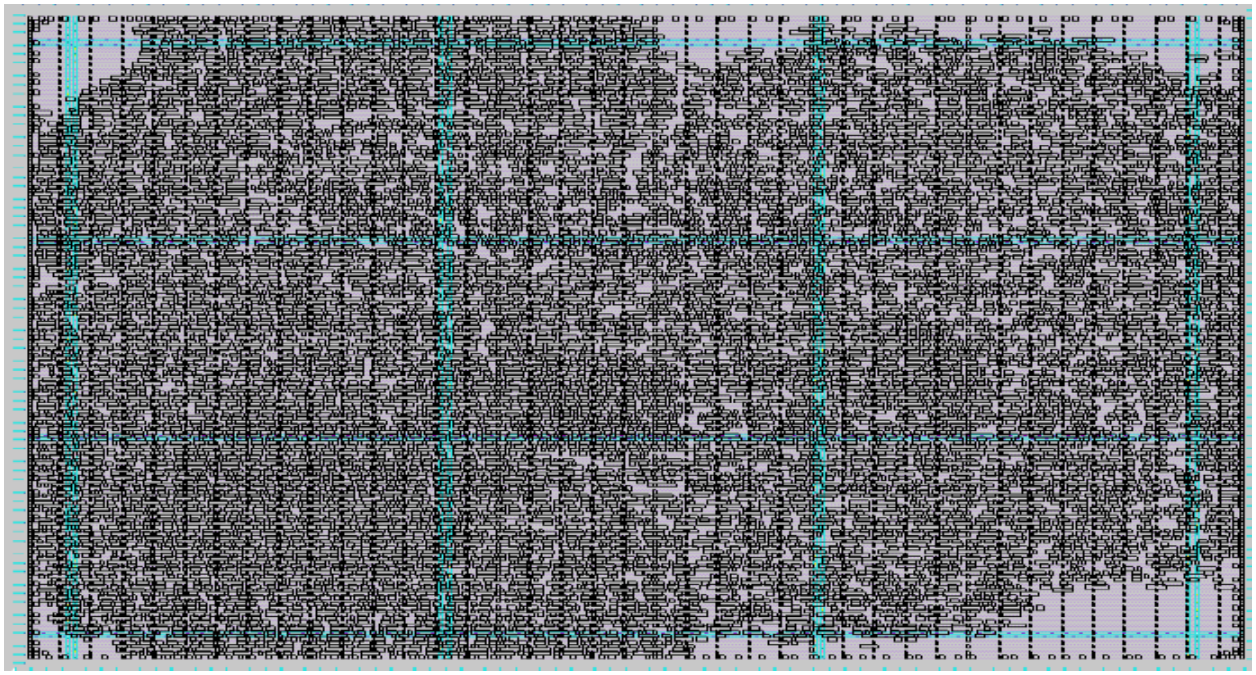
1. RePLace - Performs global placement
2. Resizer - Performs optional optimizations on the design
3. OpenDP - Performs detailed placement to legalize the globally placed components

1. RePLace: This tool performs global placement, which involves determining the approximate location of each component on the chip's surface. The goal is to minimize the total wire length of the interconnects between components, as longer wires can introduce signal delays and increase power consumption. RePLace uses algorithms such as simulated annealing and gradient descent to iteratively adjust the placement of components until a satisfactory result is achieved.

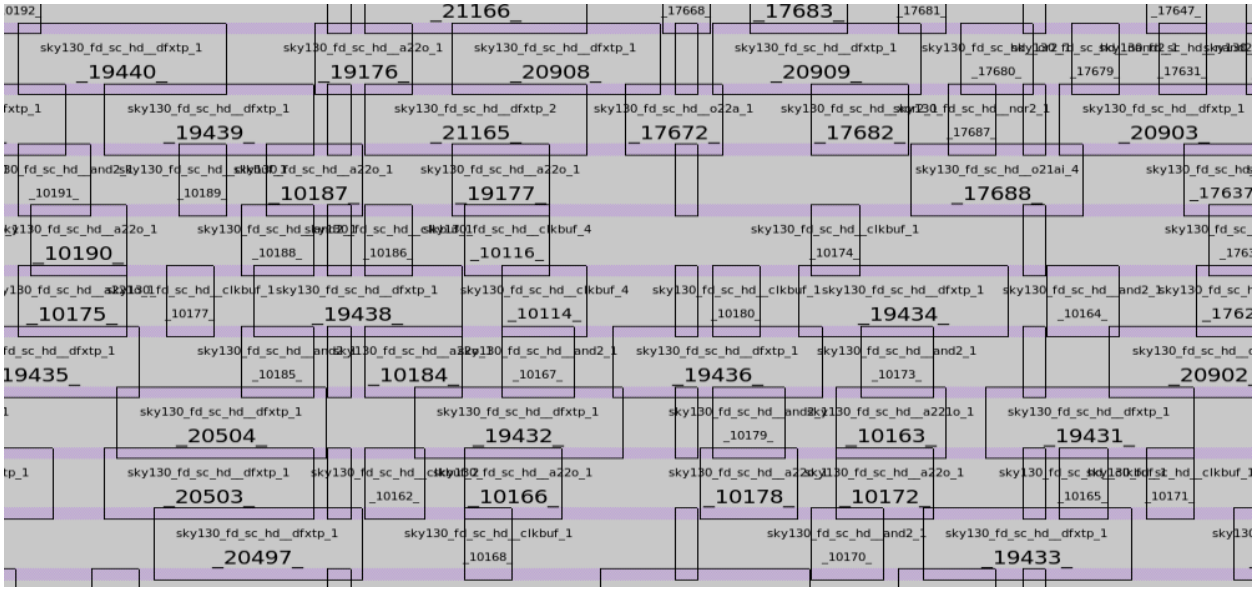


(zoomed in version of above image)

2. Resizer: After global placement, Resizer is used to perform optional optimizations on the design. These optimizations can include buffering, sizing, and cell swapping to improve timing, area, and power consumption. Resizer can also generate multiple placement solutions with different trade-offs to help designers select the optimal design.



3. OpenDP: Finally, OpenDP performs detailed placement to legalize the globally placed components. This involves ensuring that the placement is compliant with design rules, such as minimum spacing between components and metal routing layers. OpenDP also optimizes the placement of standard cells and macros to further improve the chip's performance and minimize its area and power consumption.

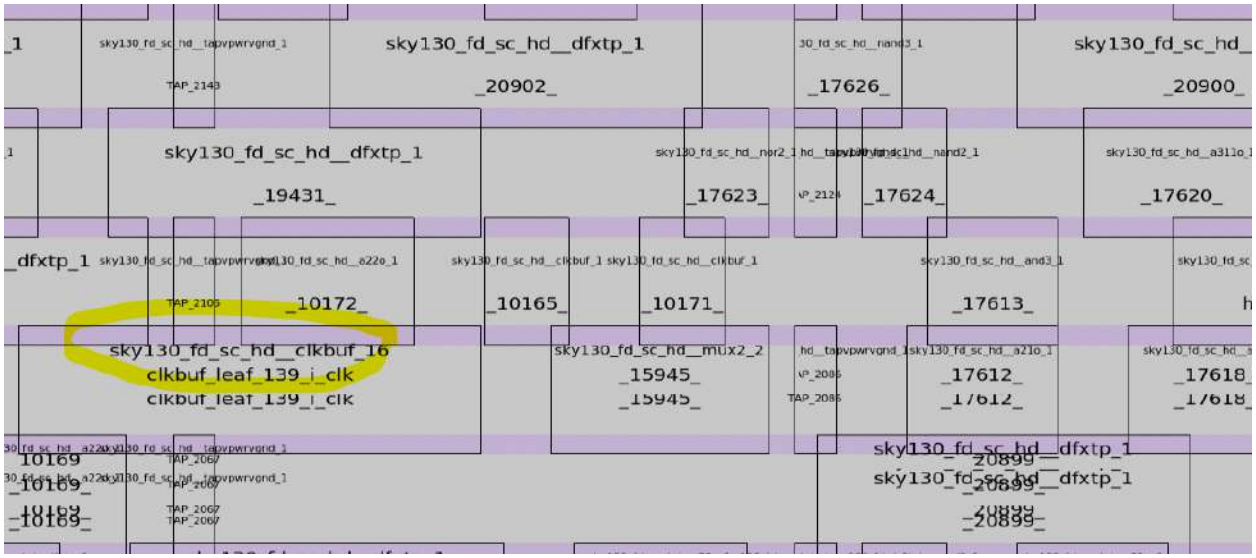


Clock Tree Synthesis (CTS) -

In digital chip design, clock distribution is a crucial part of the design process as it determines the timing accuracy and synchronicity of the entire system. The Clock Tree Synthesis (CTS) stage creates a hierarchical tree-like structure that distributes the clock signal to all sequential elements in the design with minimum clock skew and power dissipation.

One of the tools used for CTS is TritonCTS, which is a tool used for synthesizing the clock distribution network (the clock tree). It takes the logical netlist of the design and creates a clock tree with low skew, minimal delay, and low power consumption.

TritonCTS uses advanced algorithms to create the clock tree such as Steiner routing algorithms, buffer insertion, and wire sizing to meet the timing constraints, improve signal integrity, and reduce power consumption. Additionally, it performs optimization such as buffer insertion, clock gating, and power optimization, which helps to minimize clock power and reduce the dynamic power of the chip.



Clk buffer got inserted in the above step.

```
VERSION 5.8 ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN SHA ;
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 511220 521940 ) ;
ROW ROW_0 unithd 5520 10880 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_1 unithd 5520 13600 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_2 unithd 5520 16320 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_3 unithd 5520 19040 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_4 unithd 5520 21760 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_5 unithd 5520 24480 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_6 unithd 5520 27200 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_7 unithd 5520 29920 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_8 unithd 5520 32640 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_9 unithd 5520 35360 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_10 unithd 5520 38080 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_11 unithd 5520 40800 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_12 unithd 5520 43520 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_13 unithd 5520 46240 FS DO 1087 BY 1 STEP 460 0 ;
ROW ROW_14 unithd 5520 48960 N DO 1087 BY 1 STEP 460 0 ;
ROW ROW_15 unithd 5520 51680 FS DO 1087 BY 1 STEP 460 0 ;
```

In VLSI design we have studied how rows of standard cells will be abutted to each other.The ground of row0 will be abutted to ground of row1.

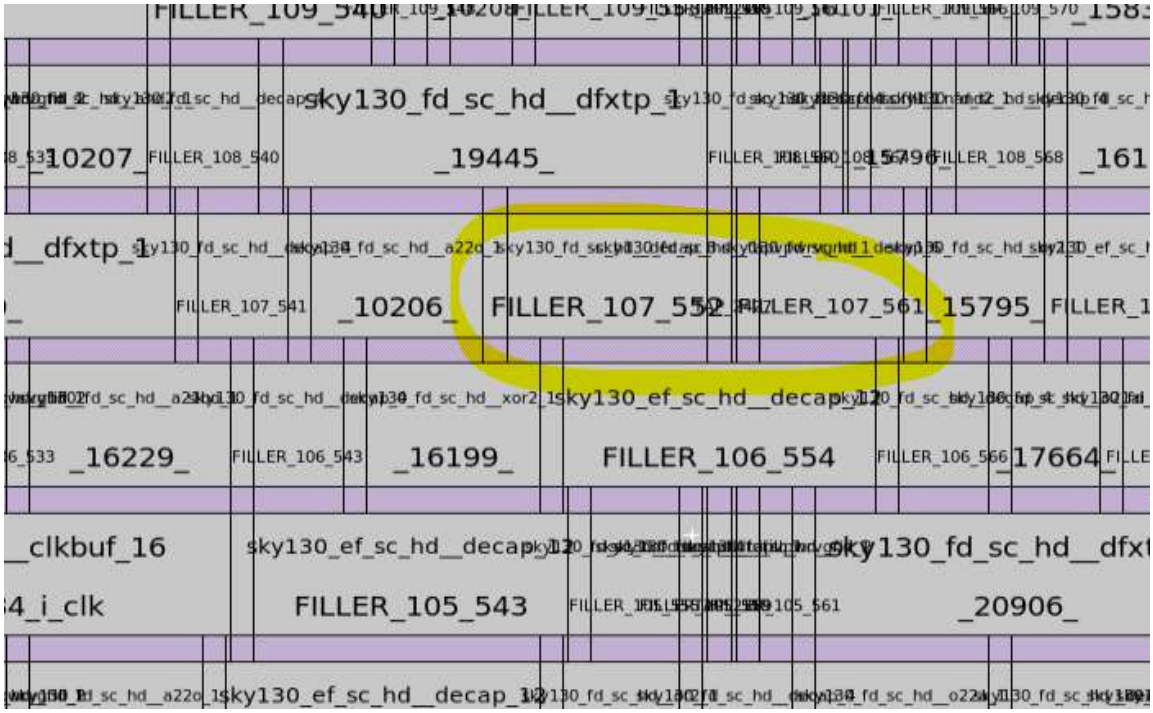
We can see that row0 is north oriented and row 1 is flipped south.

Routing -

Routing is a crucial step in the physical design of integrated circuits (ICs). It involves connecting the various components of the IC layout through metal wires to create a functioning circuit.

There are different stages involved in routing, including global routing, detailed routing, and SPEF (Standard Parasitic Extraction Format) extraction. Here's a brief overview of the three tools you mentioned:

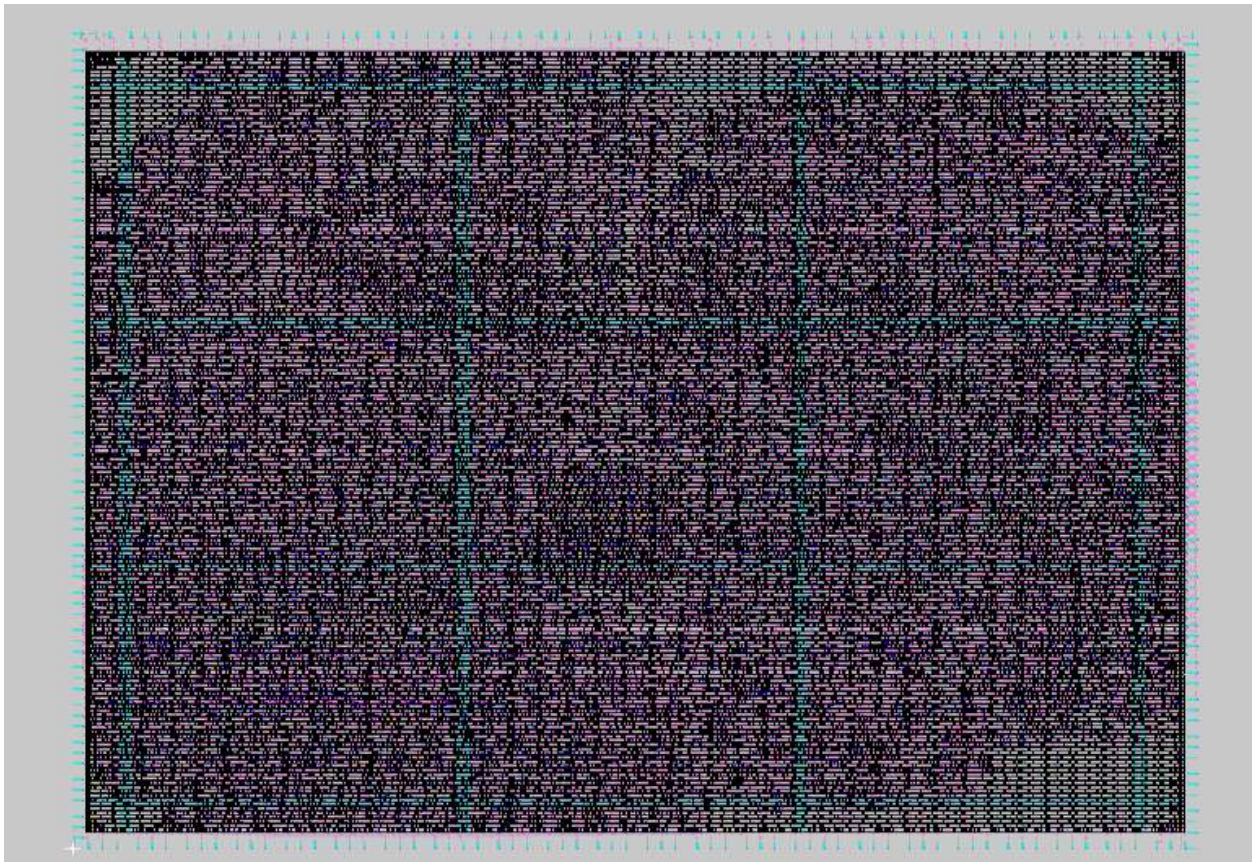
1. FastRoute is a tool used for global routing. Global routing involves determining the approximate path that interconnects various components of an IC. It is a critical step in IC design because it impacts the timing, power, and area of the circuit. FastRoute is designed to perform fast and efficient global routing, generating a guide file that is used by the detailed router.

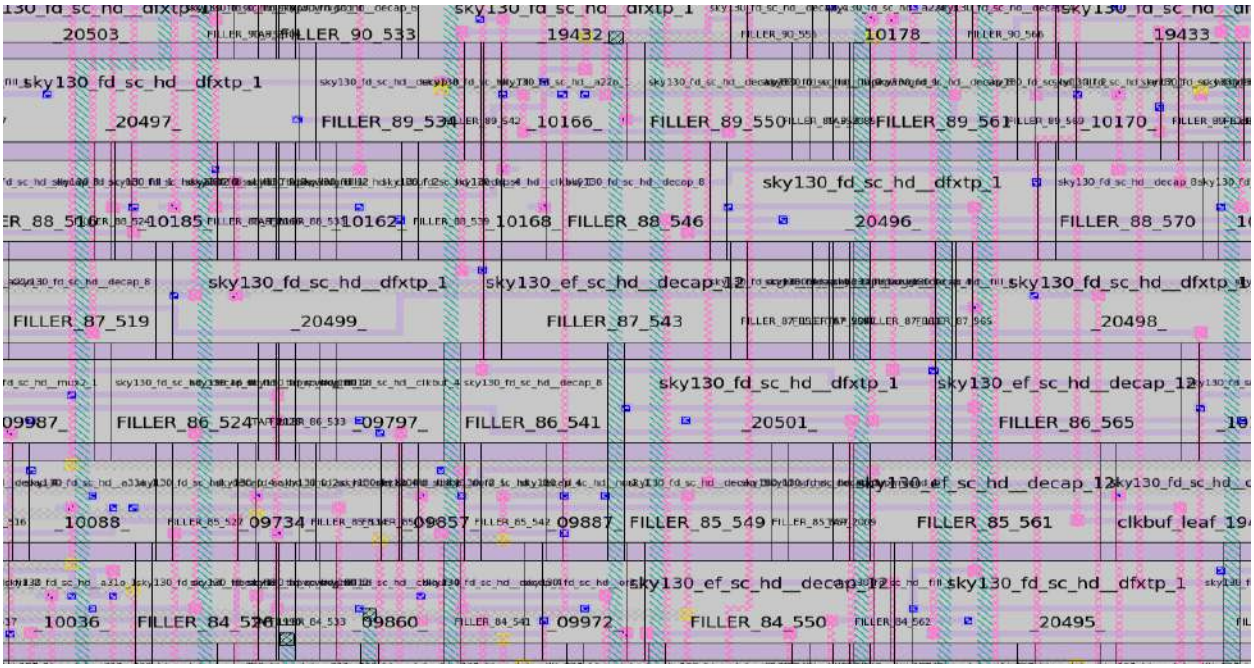


Filler cells are cells that are inserted into integrated circuits (ICs) to fill empty spaces or gaps between functional cells. They are also known as "dummy cells" or "spacer cells". Filler cells are typically used in the design of digital ICs to improve chip density, reduce timing violations, and improve manufacturing yields.

2. TritonRoute is a tool used for detailed routing. It involves routing each individual wire in the IC layout, based on the global route generated by FastRoute. TritonRoute is designed to perform fast and accurate detailed routing, ensuring that the connections between components are optimized for timing, power, and area.

3. OpenRCX is a tool used for SPEF extraction. SPEF extraction involves extracting the parasitic capacitances and resistances of the wires and other components in the circuit. These parasitics impact the timing and performance of the circuit, so it's important to extract them accurately. OpenRCX is designed to extract the SPEF format from the routed layout, which is used by other tools in the physical design flow.



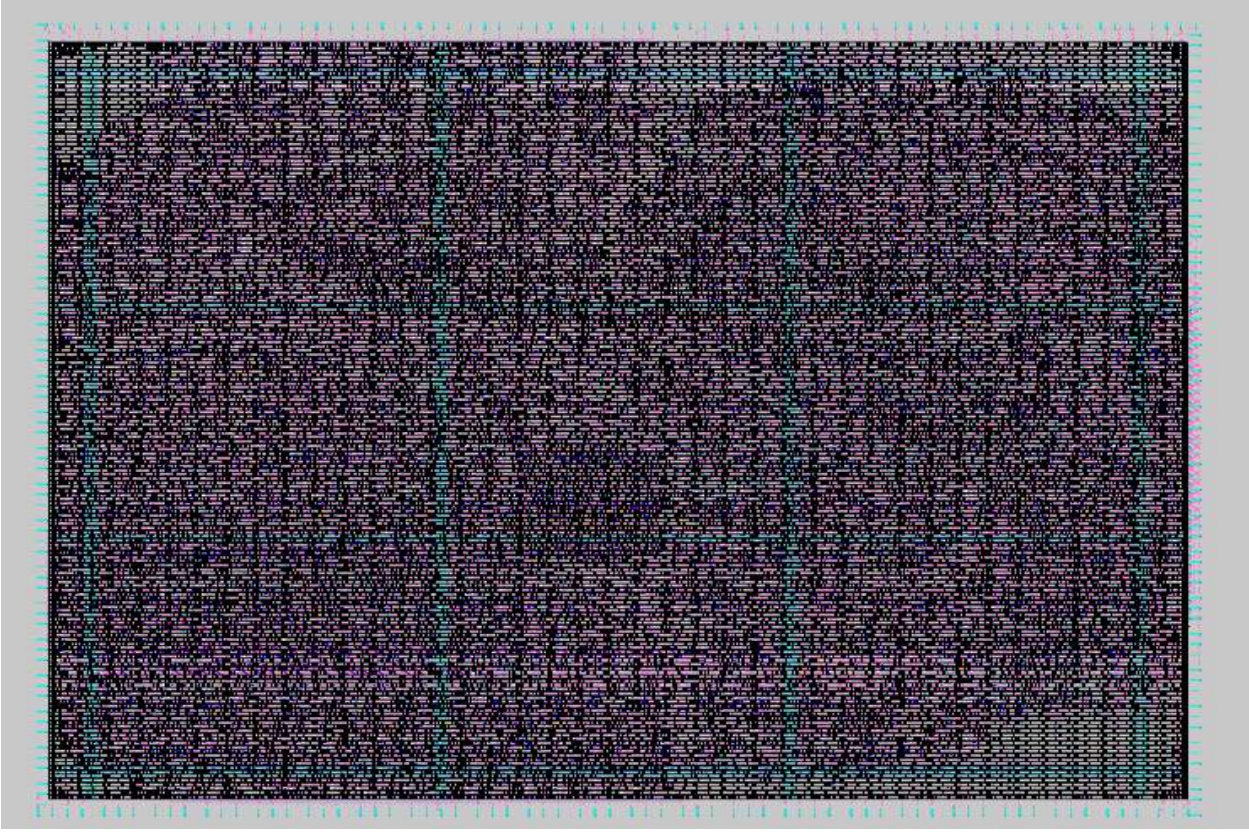

sky130_fd_sc_hd__dfxtp_1
_20503_
FILLER_90_533
_19432_
_10178_
_19433_
sky130_fd_sc_hd__dfxtp_1
_20497_
FILLER_89_534
_10166_
FILLER_89_550
FILLER_89_561
_10170_
sky130_fd_sc_hd__dfxtp_1
_10185_
_10162_
_10168_
FILLER_88_546
_20496_
FILLER_88_570
sky130_fd_sc_hd__dfxtp_1
sky130_ef_sc_hd__decap_12
sky130_fd_sc_hd__dfxtp_1
FILLER_87_519
_20499_
FILLER_87_543
_20498_
sky130_fd_sc_hd__dfxtp_1
sky130_ef_sc_hd__decap_12
FILLER_86_524
_09797_
FILLER_86_541
_20501_
FILLER_86_565
sky130_ef_sc_hd__decap_12
_10088_
_09734_
_09857_
_09887_
FILLER_85_549
FILLER_85_561
clkbuf_leaf_19
sky130_ef_sc_hd__decap_12
sky130_fd_sc_hd__dfxtp_1
_10036_
FILLER_84_526
_09860_
_09972_
FILLER_84_550
_20495_

Tapeout -

In the semiconductor industry, "tapeout" refers to the process of generating the final version of a chip's design layout in a format called GDSII, which is then used to manufacture the chip. The term originated from the practice of physically assembling the design files onto a magnetic tape before sending it to a semiconductor foundry for fabrication.

The tapeout process typically involves several stages of design verification, including circuit simulations, physical layout checks, and electrical rule checks, to ensure that the design is free of errors or manufacturing defects. Once the design is deemed ready for fabrication, the tapeout process involves converting the design files into the GDSII format, which is a standardized layout description language used by most foundries.

The final GDSII file contains all the information necessary to create the physical mask that will be used to pattern the chip's features onto a silicon wafer during the manufacturing process. Tapeout is a critical milestone in the chip design process, as it marks the point at which the design is frozen and ready for fabrication.



Signoff -

1. Magic - Performs DRC Checks & Antenna Checks
2. KLayout - Performs DRC Checks
3. Netgen - Performs LVS Checks
4. CVC - Performs Circuit Validity Checks

DES (Data Encryption Standard)

Introduction :

DES (Data Encryption Standard) is a symmetric key encryption algorithm that uses a 56-bit key to encrypt data. The algorithm works by dividing the data into 64-bit blocks and then applying a series of substitutions and permutations to each block, known as rounds. During each round, the data is mixed with a subkey derived from the main key.

The algorithm is designed to be highly secure, but due to advances in computing power, it is now considered somewhat weak by modern standards. Nonetheless, it is still widely used in some applications, and it has been a significant influence on the development of other encryption algorithms.CBC (Cipher Block Chaining) mode is a block cipher mode that adds an extra layer of security to the encryption process. In CBC mode, each plaintext block is XORed with the previous ciphertext block before encryption. This ensures that even if two plaintext blocks are the same, their corresponding ciphertext blocks will be different. The first block is XORed with an initialization vector (IV), which is a random value that is generated for each encryption process.

The IV is used to ensure that even if the same plaintext is encrypted multiple times, the resulting ciphertext will be different. CBC mode also provides some protection against certain types of attacks, such as replay attacks and dictionary attacks.

Literature survey :

Tang Songsheng at el.designed mathematical model, design principles and security analysis of block cipher, describes its typical encryption algorithm, and also proposed several methods for improving.Mathematical model includes Block cipher is use to divide the message sequence (m1,m2,m3,..,mk) into groups of equal length. Use a fixed algorithm to encrypt the message grouply under the control of the key. Long after the completion of such encryption it outputs equal length ciphertext groups. From the mathematical point of view, a block cipher with n bit and t bit keys length, is a GF(2^n) → GF(2^m)'s replacement under the control of 2^t number keys.

Mukta Sharma at el.implemented symmetric key encryption algorithm "DES-Data Encryption Standard." Highlighted on the primary flow, history, implementation, significant drawbacks, etc. Researches also showed several attacks happened up to now which included Brute-force, differential cryptanalysis, and linear cryptanalysis are among the attempted attacks.Researchers also performed the several performance analysis , The criteria' s to evaluate the performance are Architecture, Scalability, Flexibility, Security, and Limitations. The performance of encryption algorithms is based on various parameters like Encryption Time, Decryption Time, Speed, Usage, CPU Utilization, Key size, Block Size, Security, etc. Various researchers have critically analyzed DES based on the data files, file size, etc.

Project details :

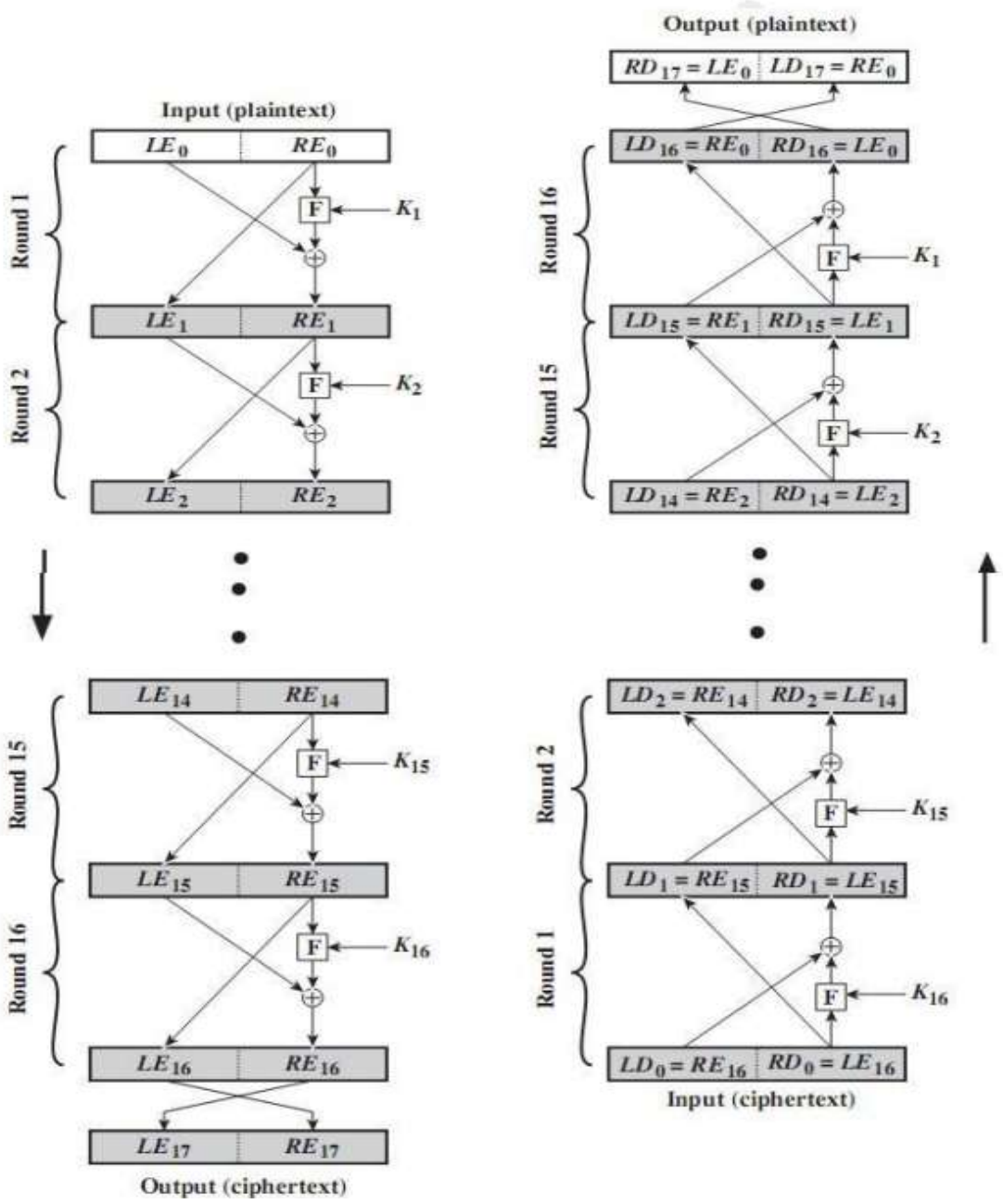
The project aims to implement DES algorithm.RTL code of the algorithm is implemented in verilog and simulated in quartus prime questasim intel simulator and results are verified by simulating testbench written in verilog.

The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher that encrypts 64-bit blocks of plaintext into ciphertext. Here are the steps of the DES algorithm:

1. Key Generation: The 64-bit secret key is transformed into 16 48-bit subkeys, one for each round of encryption. The key is first permuted using a fixed table called the Initial Permutation (IP), and then split into two 28-bit halves. Each half is then subjected to a series of circular shifts, and finally combined to generate the 48-bit subkeys.

2. Initial Permutation: The plaintext block is permuted using the Initial Permutation (IP) table, which rearranges the bits of the block according to a fixed pattern.

3. Round Function: The 64-bit plaintext block is divided into two 32-bit halves, left and right. Each round of the algorithm operates on these halves separately. The right half is expanded from 32 bits to 48 bits using the Expansion Permutation (E), and then XORed with the current subkey. The resulting 48-bit value is divided into eight 6-bit blocks, each of which is passed through an S-box (Substitution box). Each S-box substitutes its input 6 bits with a different 4-bit output, according to a fixed table. The outputs of the S-boxes are then combined into a 32-bit block using the Permutation (P) table.



4. Swap: After 16 rounds of the round function, the left and right halves of the block are swapped.

5. Final Permutation: The final 64-bit ciphertext block is generated by permuting the left and right halves of the block using the Final Permutation (FP) table, which is the inverse of the Initial Permutation.

6. Each of these steps is repeated for each 64-bit block of plaintext to be encrypted, using the same key for each block. The decryption process is the same as the encryption process, but with the subkeys applied in reverse order.

Single round of encryption :

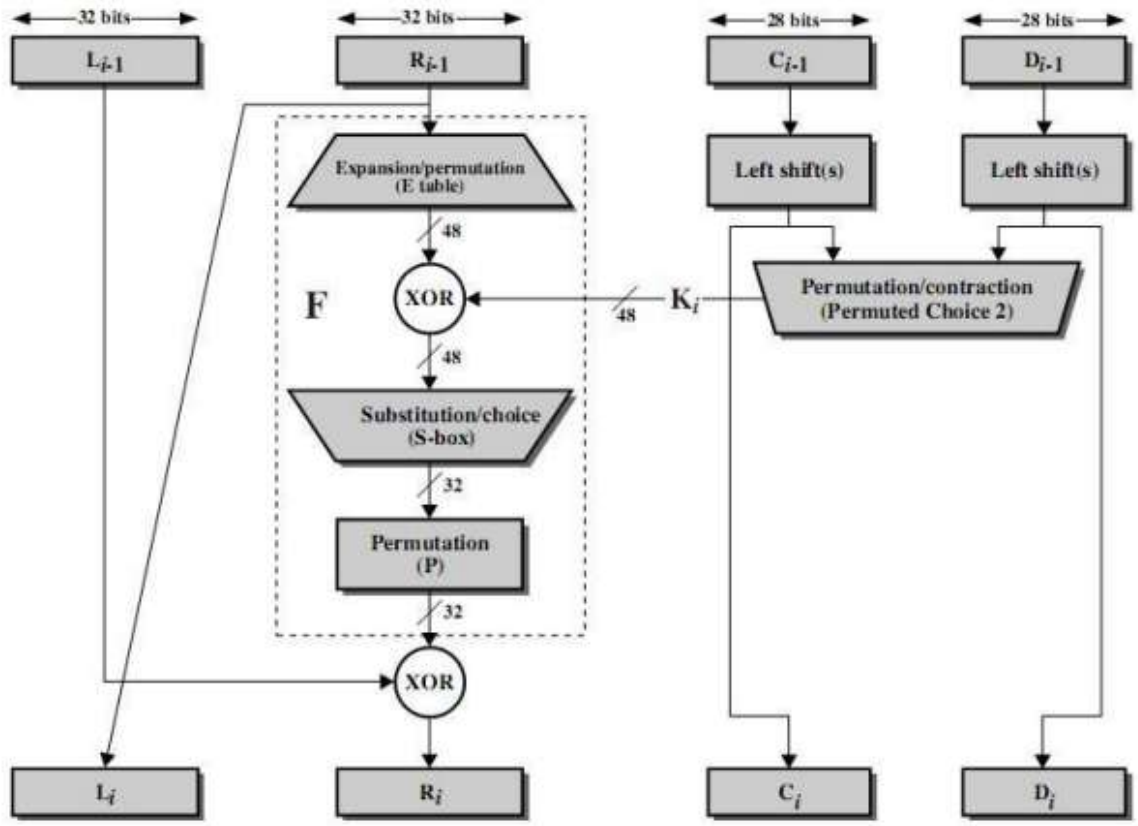


Figure 3.5 Single Round of DES Algorithm

Implementation :

Key generation module:

The key generation module in the Data Encryption Standard (DES) is responsible for generating 16 subkeys, each of which is used in one of the 16 rounds of the DES algorithm. The subkeys are derived from a 64-bit key, which is typically supplied by the user.

The key generation module uses a process called key scheduling to derive the 16 subkeys from the 64-bit key. The key scheduling process starts by permuting the 64-bit key using a fixed permutation called the "permuted choice 1" (PC1) table. This produces a 56-bit key, which is divided into two 28-bit halves.

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

In each of the 16 rounds of key scheduling, the two 28-bit halves are rotated left by a certain number of bits, which depends on the round number. Then, the rotated halves are each permuted using a fixed permutation called the "permuted choice 2" (PC2) table. This produces a 48-bit subkey, which is used in the corresponding round of the DES algorithm.

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES encrypt module

1. Permutation: The 64-bit plaintext is first permuted using a fixed permutation called the initial permutation (IP) table. This rearranges the bits of the plaintext in a specific way.

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

2. Splitting: The permuted plaintext is split into two 32-bit halves, called the left half (L0) and the right half (R0).

3. Rounds: The encryption process involves 16 rounds of operations, each of which involves manipulating the left and right halves of the plaintext using a subkey derived from the original key. The operations in each round are:

- Expansion: The right half of the plaintext (Ri-1) is expanded from 32 bits to 48 bits using a fixed permutation called the expansion (E) table.

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- Substitution: The expanded right half is XORed with the current subkey (Ki) and then split into eight 6-bit chunks. Each 6-bit chunk is substituted using a fixed 4x16 substitution (S) box, which replaces the 6 input bits with 4 output bits.

- Permutation: The output of the substitution step is then permuted using a fixed permutation called the permutation (P) table.

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

- Mixing: The permuted output of the substitution step is then XORed with the left half of the plaintext (Li-1).

- Swapping: After all 16 rounds have been completed, the left and right halves of the final round output are swapped.

4. Final permutation: The swapped left and right halves of the final round output are then permuted using an inverse initial permutation called the inverse permutation (IP) table, which produces the 64-bit ciphertext.

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

CBC mode in encryption and decryption :

CBC (Cipher Block Chaining) is a mode of operation for block ciphers, such as AES and DES. In CBC mode, each block of plaintext is first XORed with the previous block of ciphertext before being encrypted. This helps to introduce randomness and prevent patterns in the plaintext from being preserved in the ciphertext.

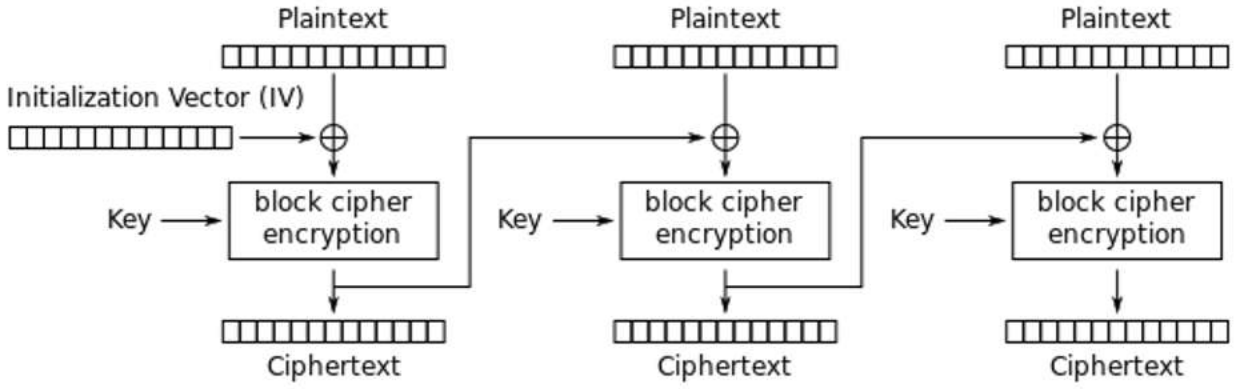
Here's how CBC encryption works:

1. Initialization: An initialization vector (IV) is chosen randomly and used to encrypt the first block of plaintext.

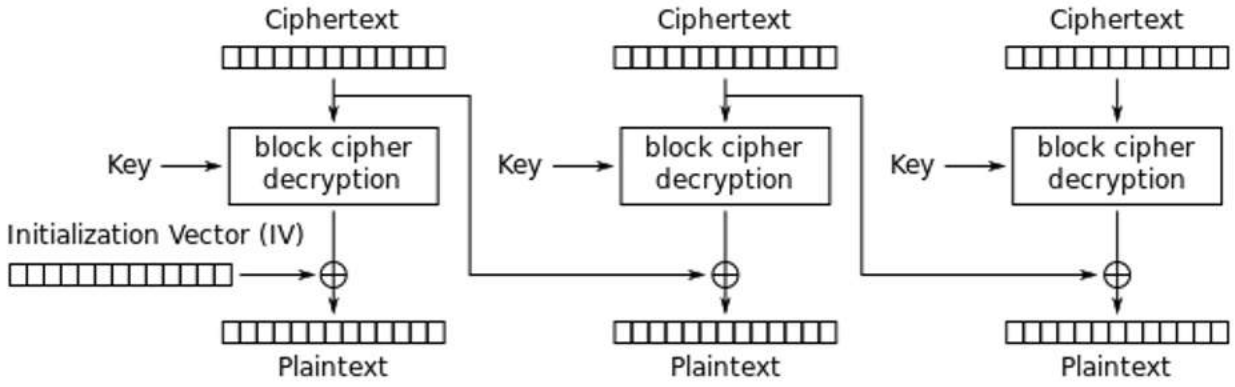
2. Block chaining: For each subsequent block of plaintext, the plaintext is first XORed with the previous block of ciphertext. If this is the first block, then the IV is used as the "previous block". The resulting block is then encrypted using the block cipher.

3. Output: The resulting ciphertext blocks are concatenated to form the final ciphertext.

Decryption in CBC mode works similarly, but in reverse. Each block of ciphertext is first decrypted using the block cipher, and then XORed with the previous block of ciphertext to recover the original plaintext block.



Cipher Block Chaining (CBC) mode encryption



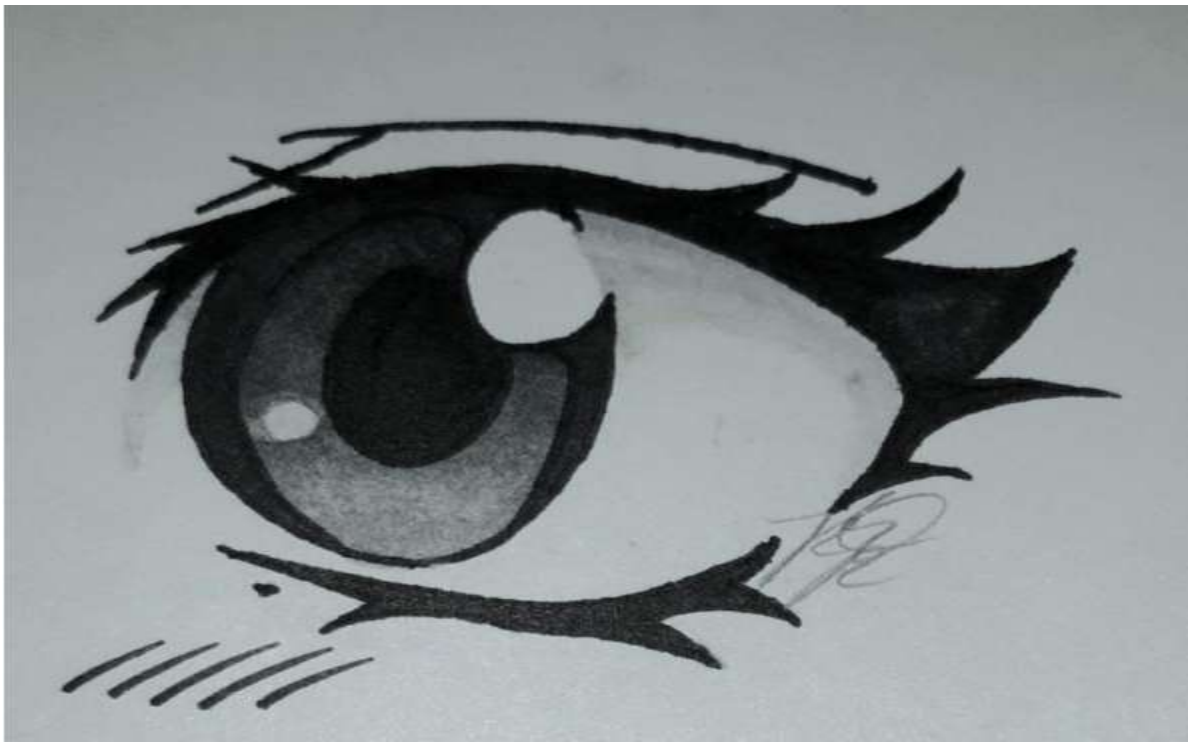
Cipher Block Chaining (CBC) mode decryption

Results :

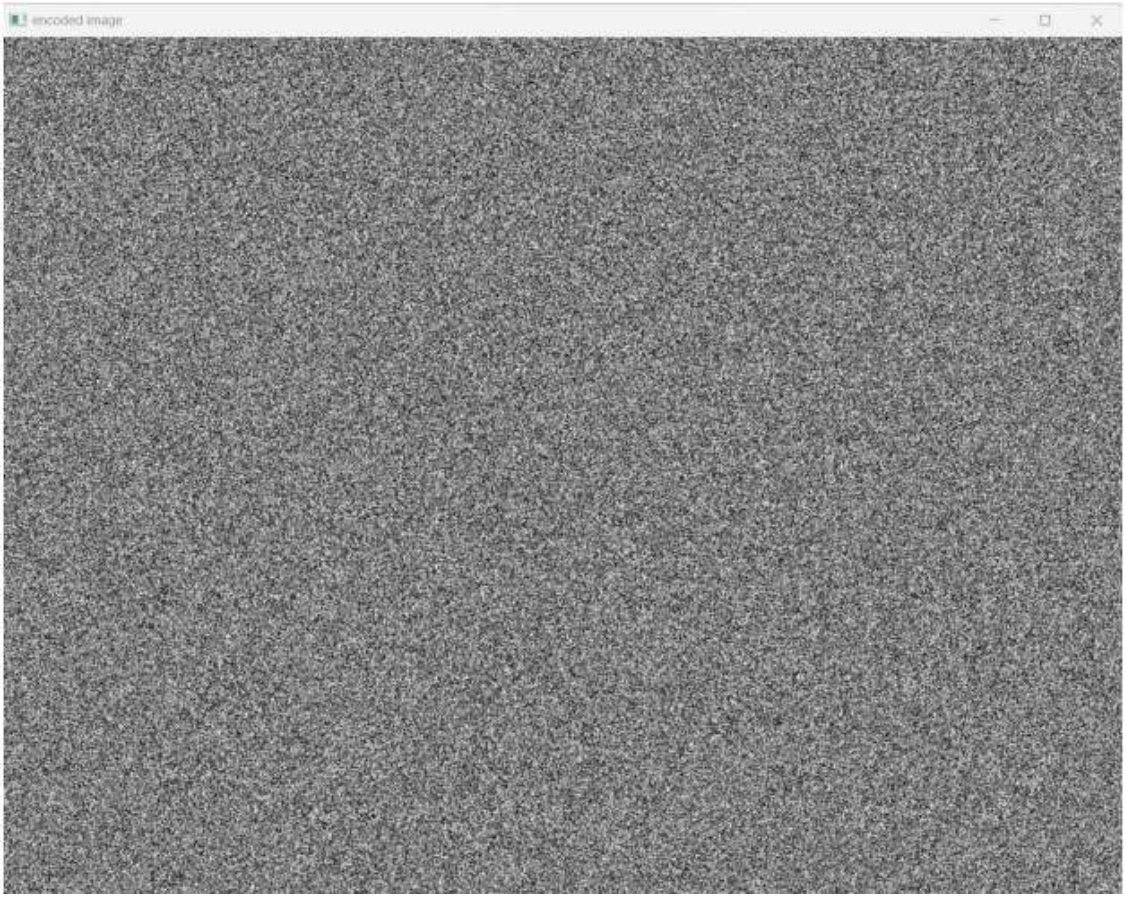
We used the DES cryptographic algorithm in CBC mode to encrypt given 1024*1024 grayscale image.We used the DES decryption algorithm implemented by us to decrypt the image into original form.

(“images attached here are cropped to format the document,these are not 1024*1024 pixel images.The real images are attached with this file“).

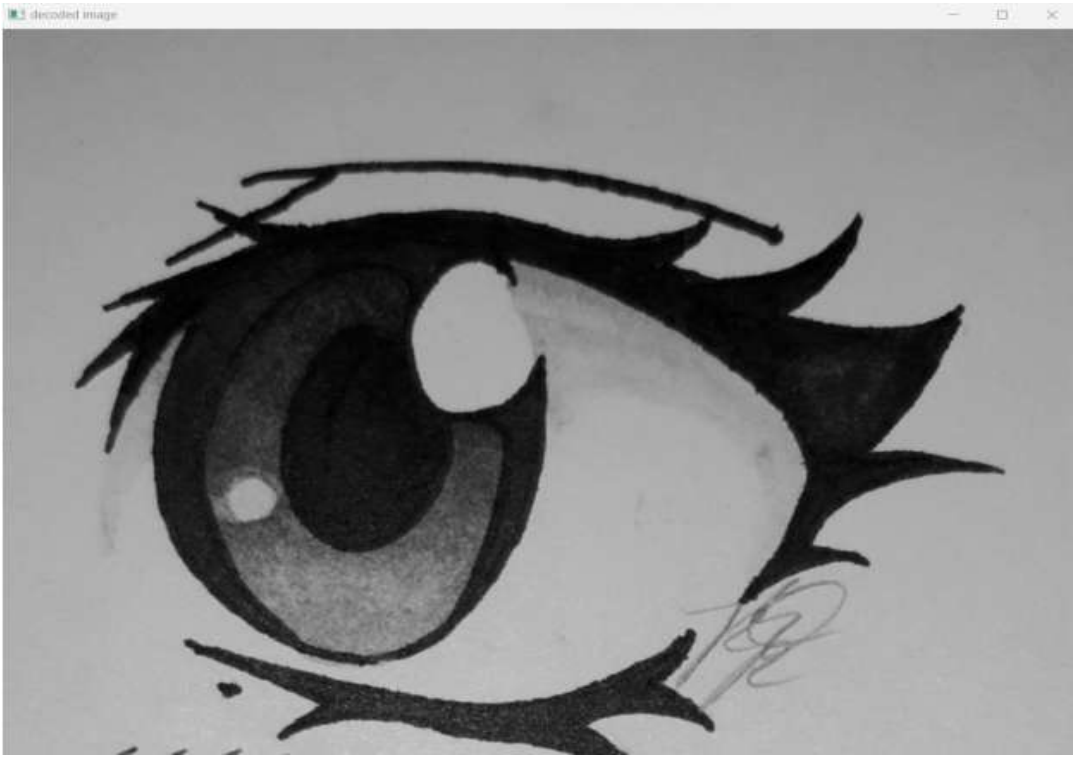
Original image :



Encrypted image:



Decrypted image :



AES(Advanced Encryption Standard)

Introduction :

AES (Advanced Encryption Standard) is a widely used cryptographic algorithm that provides a high level of security and is widely used in various applications such as online banking, e-commerce, and secure communication. The AES algorithm was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and was selected as the official encryption standard by the U.S. National Institute of Standards and Technology (NIST) in 2001.

The AES algorithm uses symmetric-key encryption, which means that the same secret key is used for both encryption and decryption. The AES algorithm supports three different key sizes: 128-bit, 192-bit, and 256-bit. The larger the key size, the stronger the encryption and the more secure the data.The AES algorithm operates on a block of plaintext data, typically 128 bits in size. The algorithm uses a series of mathematical operations, including substitution, permutation, and mixing, to transform the plaintext block into a ciphertext block. The same operations are used in reverse order during decryption to recover the original plaintext.

AES-128 uses a 128-bit key, AES-192 uses a 192-bit key, and AES-256 uses a 256-bit key. While all three key sizes are considered secure, AES-256 provides the strongest encryption and is recommended for the most sensitive applications.

Literature survey :

AES is the most effective algorithm to provide security in message transmission.Flevina Jonese D’souza et al.implemented AES algorithm with hybrid approach of Dynamic Key Generation and Dynamic S-box Generation which is survive from most of the possible attacks like Brute-force Attack, Differential Attack, Algebraic Attack and Linear Attack.In hybrid approach first researchers added more complexity in data to increase Confusion and Diffusion in Cipher text by using Dynamic Key Generation and then by using Dynamic S-Box Generation they made difficult for attacker to do any down study of static set of S-box.

Singh et al. studied three encryption algorithms like Rivest-Shamir-Adleman, Data Encryption Standard, Triple Data Encryption Standard and Advanced Encryption Standard to analyze the effectiveness of cryptography based on speed, time, and throughput. Researchers proved that Advanced Encryption Standard is a better algorithm than Data Encryption Standard, Triple Data Encryption Standard and Rivest-Shamir-Adleman for communication security.

Mahajan et al. surveyed the performance of existing encryption techniques like Advanced Encryption Standard, Data Encryption Standard and Rivest-Shamir-Adleman algorithms. Based

on the investigation researchers determined that Advanced Encryption Standard algorithm consumes least encryption and Rivest-Shamir-Adleman consumes longest encryption time. Also, the Decryption of Advanced Encryption Standard algorithm is better than other algorithms. From the simulation result, it is estimated that the Advanced Encryption Standard algorithm is greatly better than Data Encryption Standard and Rivest-Shamir-Adleman algorithm.

Project details :

The project aims to implement AES algorithms of different key lengths like 128,192 and 256 bit key lengths.RTL code of the algorithm is implemented in verilog and simulated in quartus prime questasim intel simulator and results are verified by simulating testbench written in verilog.

AES Algorithm :

The AES (Advanced Encryption Standard) algorithm consists of several steps that are performed in sequence to encrypt and decrypt data. Here are the general steps of the AES algorithm:

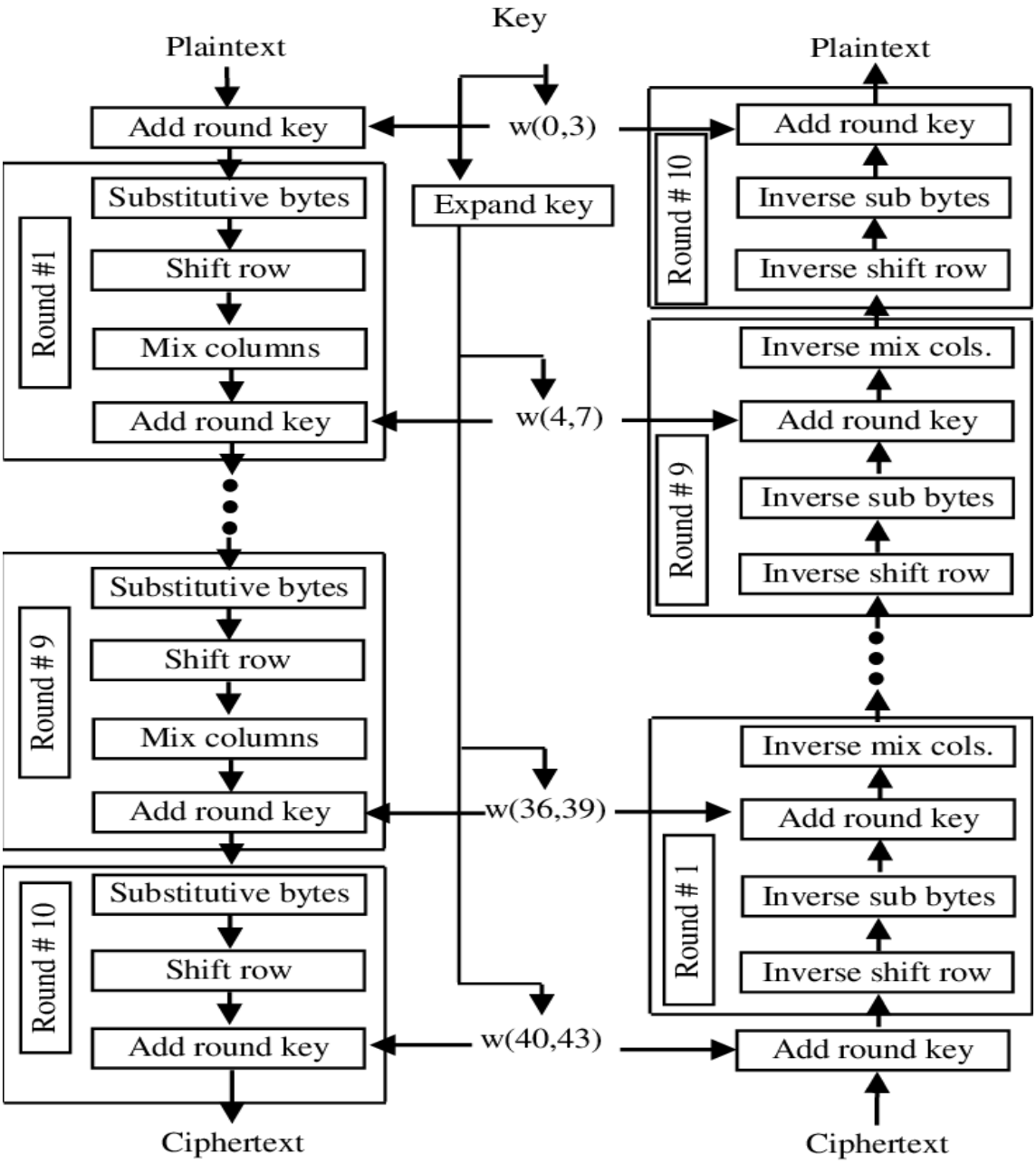
1. Key Expansion: The original secret key is expanded into a set of round keys, one for each round of encryption and decryption. This process involves applying a series of mathematical operations to the original key to generate the round keys.

2. Initial Round: In the first round, the plaintext is combined with the first round key using a bitwise XOR operation.

3. Rounds: There are a fixed number of rounds that are performed based on the key size. In each round, the ciphertext from the previous round is processed using a series of mathematical operations that include byte substitution, shifting rows, and mixing columns. Each round uses a different round key that is generated during the key expansion phase.

4. Final Round: In the final round, the ciphertext is processed using byte substitution and shifting rows. The final round does not include the mixing columns step.

5. Output: The final output of the encryption process is the ciphertext, which is the encrypted form of the original plaintext. To decrypt the ciphertext, the reverse operations are performed using the same round keys in reverse order.


Key
Plaintext
Add round key
w(0,3)
Expand key
Round #1
Substitutive bytes
Shift row
Mix columns
Add round key
w(4,7)
Round # 9
Substitutive bytes
Shift row
Mix columns
Add round key
w(36,39)
Round # 10
Substitutive bytes
Shift row
Add round key
w(40,43)
Ciphertext
Plaintext
Add round key
Round # 10
Inverse sub bytes
Inverse shift row
Inverse mix cols.
Add round key
Round # 9
Inverse sub bytes
Inverse shift row
Inverse mix cols.
Add round key
Round # 1
Inverse sub bytes
Inverse shift row
Add round key
Ciphertext

Implementation :

AES encryption

The design and implementation of AES encryption algorithm in RTL consists of 5 modules :

Key expansion module :

The key expansion module takes the original secret key as input and generates a set of round keys that are used in each round of encryption and decryption. The key expansion process involves applying a series of mathematical operations to the original key to generate a set of round keys.The key expansion module begins by copying the original key into the first set of round keys. Then, the module applies a series of operations to each subsequent set of round keys to generate the next set of round keys.

The key expansion process involves four main steps:

1. Key Schedule Core: This step involves performing a set of mathematical operations on a portion of the previous round key. This operation involves rotating the bytes of the previous round key, performing byte substitution using the AES S-box, and XORing the result with a constant value that depends on the current round.

2. Key Schedule Round: This step involves generating a new round key by XORing the output of the Key Schedule Core step with a portion of the previous round key.

3. Final Round Key: This step involves generating the final round key, which is used in the final round of encryption or decryption. The final round key is generated by applying the Key Schedule Core step one more time and XORing the result with a portion of the previous round key.

4. Key Expansion: This step involves repeating the Key Schedule Round step for a fixed number of rounds, depending on the key size, to generate a complete set of round keys.

Add round key module :

The Add Round Key module takes as input the state array, which is a matrix of bytes representing the data being encrypted or decrypted, and a round key, which is a matrix of bytes generated by the Key Expansion module. The module then performs a bitwise XOR operation between the state array and the round key to produce the output.

The Add Round Key module is the first step in each round of encryption or decryption and is essential for providing security to the data. The XOR operation ensures that the data is mixed with the key in a way that makes it difficult for an attacker to decrypt the data without the correct key.

Sub bytes module :

The Substitution Bytes module takes as input the state array, which is a matrix of bytes representing the data being encrypted or decrypted, and replaces each byte in the state array with a corresponding byte from the AES S-box. The AES S-box is a fixed table that contains a non-linear substitution value for each possible byte value.

The Substitution Bytes module is a simple operation that involves looking up a value from the AES S-box for each byte in the state array.

Shift rows :

the data being encrypted or decrypted, and shifts the bytes in each row of the matrix to the left. The amount by which the bytes are shifted depends on the row number, with the first row being shifted by 0 bytes, the second row being shifted by 1 byte, the third row being shifted by 2 bytes, and the fourth row being shifted by 3 bytes.

Mix columns :

The Mix Columns module takes as input the state array, which is a matrix of bytes representing the data being encrypted or decrypted, and applies a matrix multiplication to each column of the matrix. The matrix used for the multiplication is a fixed matrix called the MixColumns matrix. The multiplication is done in the Galois field GF(2^8), which is a finite field that is used in AES to provide a high degree of diffusion in the data being encrypted or decrypted.

Sbox module :

The S-box module is a lookup table that contains a fixed set of substitution values for each possible byte value. In AES, the S-box is a 256-byte table that is generated using a mathematical formula based on the irreducible polynomial x^8 + x^4 + x^3 + x + 1. Each entry in the S-box is calculated using a combination of bitwise operations and modular arithmetic.

AES decryption

The decryption algorithm begins with the Add Round Key module, where the last round key is added to the ciphertext. This is followed by the InvShiftRows module, which is the inverse of the ShiftRows module used in encryption. The InvSubBytes module, which is the inverse of the Substitution Bytes module used in encryption, is then applied to the state array.

Next, the decryption algorithm uses the MixColumns module, which is the inverse of the Mix Columns module used in encryption, to mix the columns of the state array. However, the last round of decryption does not use the MixColumns module.

After the MixColumns module, the decryption algorithm again applies the Add Round Key module using the round keys in reverse order. This process is repeated for each round of decryption, except for the first and last rounds which have slightly different steps.

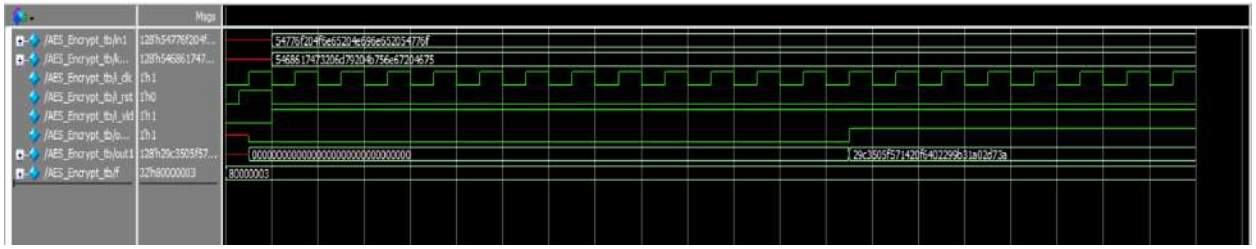
The decryption algorithm is closely related to the encryption algorithm because it uses the same modules in reverse order. This means that the key expansion module is also the same for both encryption and decryption. The only difference is that the round keys are used in reverse order during decryption.

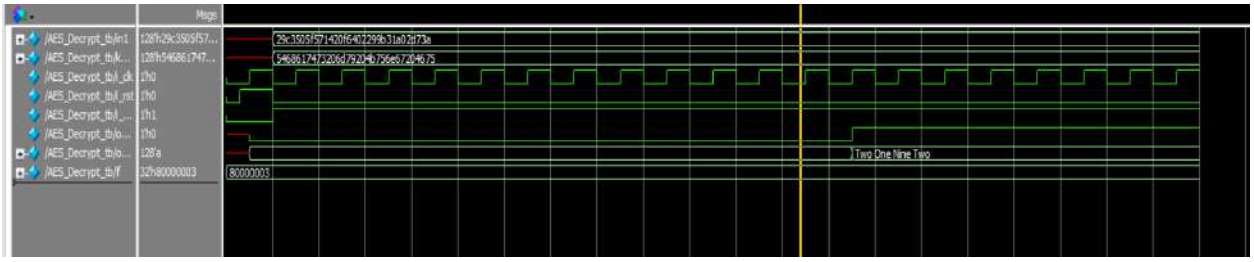
Results

Simulation

Case 1 : message : Two One Nine Two
Key : 128'h54_68_61_74_73_20_6D_79_20_4B_75_6E_67_20_46_75

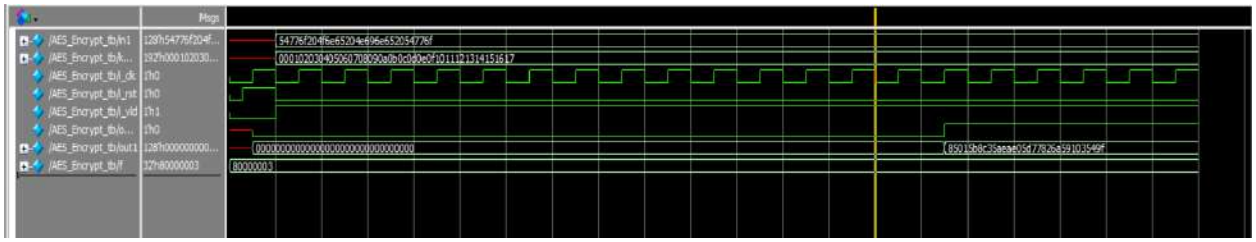
Encoded output : 29c3505f571420f6402299b31a02d73a.
Decoded output : Two One Nine Two.

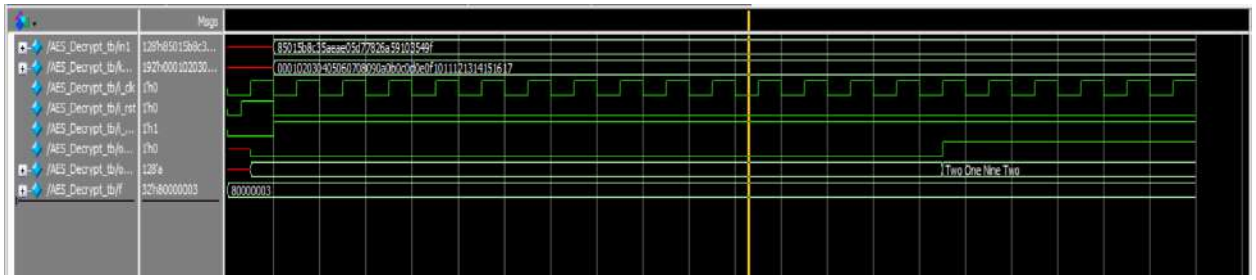




Case 2 : message : Two One Nine Two
Key : 192'h_000102030405060708090a0b0c0d0e0f1011121314151617;

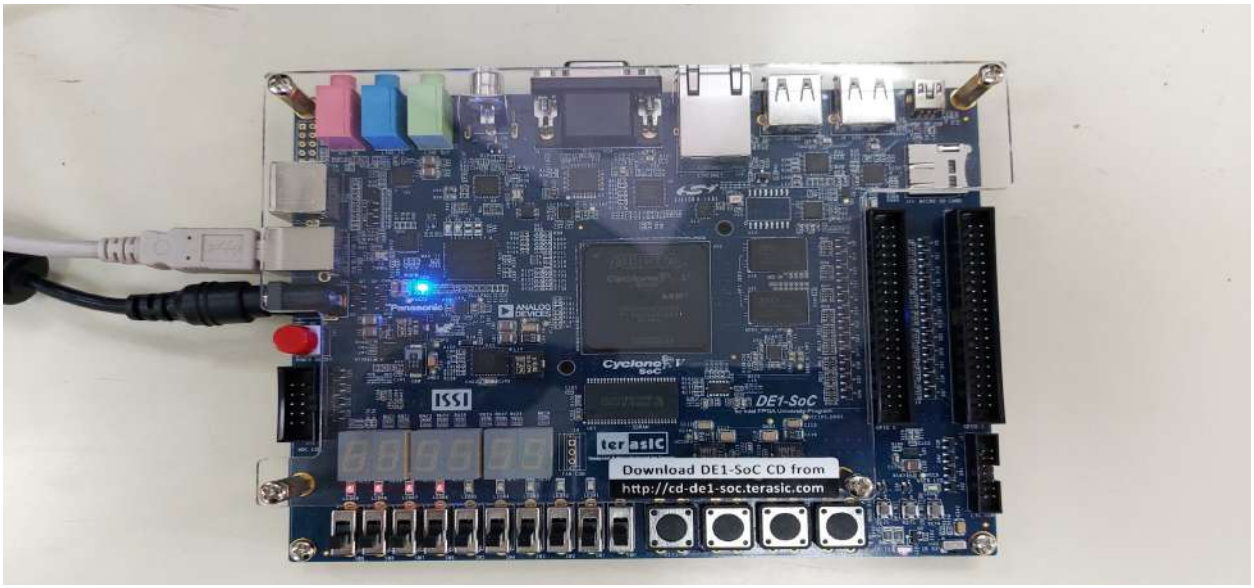
Encoded output :85015b8c35aeae05d77826a59103549f.
Decoded output : Two One Nine Two.



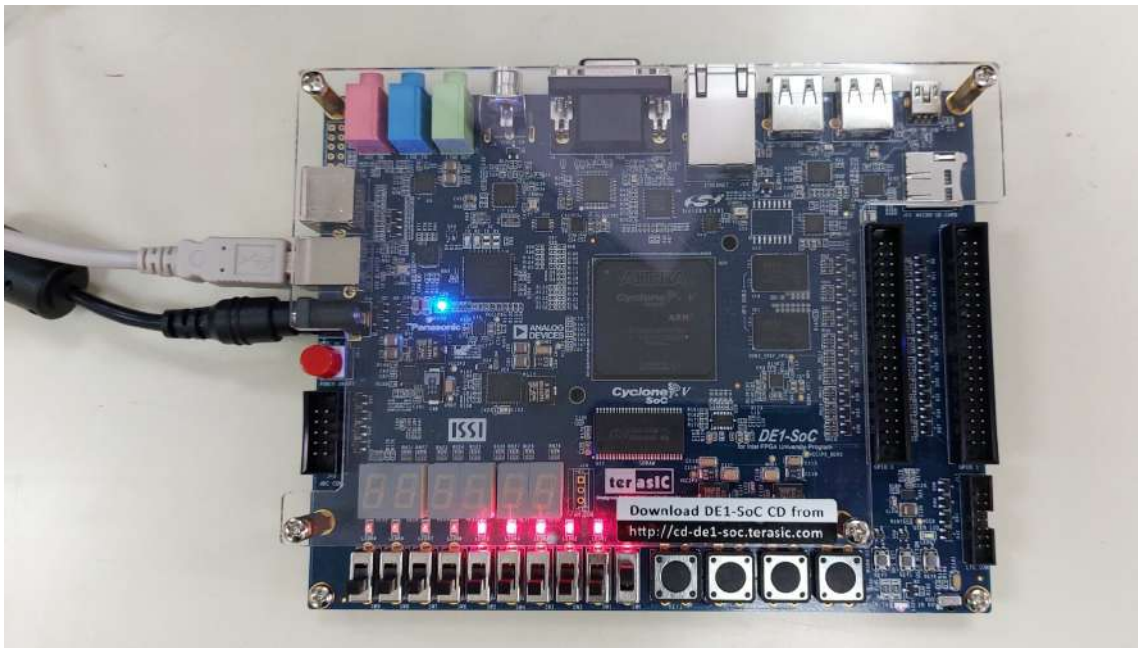


Testing was done on the **DE1-SOC** kit by implementing the wrapper module which generates the required inputs to the system and if tests passed it turns on the LEDs.

Enable turned off



Enable turned on



Conclusion :

In conclusion, the Cryptographic Processor project has successfully implemented the Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Secure Hash Algorithm (SHA) using Verilog language. These cryptographic algorithms are widely used in securing communication and data transmission, and their implementation in hardware form enhances their speed and efficiency.

Additionally, the SHA algorithm was designed using OpenLane, which is an open-source digital ASIC design flow. The use of OpenLane allowed for the creation of an optimized and high-performance SHA design, while also providing a transparent and accessible development process.

Overall, the successful implementation of these cryptographic algorithms in Verilog and OpenLane has demonstrated the effectiveness of hardware-based encryption and highlights the importance of secure communication in modern-day systems. Future work could involve further optimizing the designs for faster speeds and lower power consumption, as well as exploring the implementation of other encryption algorithms.

Reference(s) :

[1] Flevina Jonese D'souza, Dakshata Panchal, "Advanced Encryption Standard (AES) Security Enhancement using Hybrid Approach" in International Conference on Computing, Communication and Automation (ICCCA2017) .

[2] Tang Songsheng , Ma Xianzhen, "Research of typical block cipher algorithms" in the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE).

[3] Mukta Sharma and Dr. R.B. Garg , "DES: The Oldest Symmetric Block Key Encryption Algorithm" in Proceedings of the SMART -2016, IEEE Conference ID: 39669 .

[4] Seung-Jo Han, Heang-Soo Oh, Jongan Park, " The improved Data Encryption Standard (DES) Algorithm"

[5] FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION (FIPS PUB 180-4) http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[6] National Institute of Standards and Technology (NIST) http://www.nic.funet.fi/pub/crypt/cryptography/symmetric/aes/nist/Rijndael.pdf

[7] Chanjuan Li, Qingguo Zhou, Yuli Liu, Qi Yao (Cost-efficient Data Cryptographic Engine Based on FPGA) http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5992044