King Abdulaziz University
College of Engineering
EE460 – Digital Design II

Section: C2
SPRING
2023

(DES Implementation in FPGA)

Report

#	Student Name	ID
1	Faisal Jehad Abdullah Abushanab	1945603

Instructor:

Dr. Mohammad H. Awedh

2023/6/20

Contents

List of Figures

List of Tables

Abstract:

This report presents the implementation of the Data Encryption Standard (DES) algorithm in an FPGA, with a focus on digital design methodologies. The main objectives of the project were to design the necessary units, including the main unit and the controller, as well as the Datapath, for the FPGA-based implementation of the DES algorithm. Additionally, the project involved incorporating UART communication for input and output handling. The implementation was thoroughly tested using simulation techniques and on an actual FPGA board. The project successfully demonstrated the functionality of both encryption and decryption processes using the DES algorithm, with support for 64-bit plaintext. The performance and correctness of the FPGA-based DES implementation were evaluated, considering factors such as throughput, latency, and resource utilization. Throughout the project, various digital design techniques and methodologies were employed, including RTL design, synthesis, and optimization. The report provides detailed insights into the design process, highlighting the key design decisions and optimizations made for efficient and reliable operation. Overall, the project served as a valuable opportunity to gain practical experience in digital design methodologies, particularly in the context of implementing cryptographic algorithms like DES in FPGAs. The successful implementation and testing of the DES algorithm in an FPGA underscore the effectiveness and applicability of FPGA-based solutions for secure data encryption and decryption.

-- **Keywords:** Data Encryption Standard (DES), FPGA (Field-Programmable Gate Array), Digital Design, Cryptographic Algorithm.

Introduction

The Data Encryption Standard (DES) algorithm is a widely recognized and historically significant symmetric-key encryption algorithm. Developed by IBM in the 1970s, DES has been widely used for secure data communication and storage. However, with the advancement of technology and increased computing power, DES is now considered relatively vulnerable to modern cryptographic attacks. Despite this, studying and implementing DES remains valuable for understanding the principles of symmetric-key encryption and digital design methodologies.

This project aims to implement the DES algorithm in an FPGA (Field-Programmable Gate Array), providing a hands-on opportunity to explore digital design techniques and gain practical experience in implementing cryptographic algorithms. The FPGA-based implementation offers the flexibility of reconfigurable hardware, allowing for efficient execution of DES operations while offering potential for customization and optimization.

The main objectives of this project include the design and implementation of the necessary units, such as the main unit and the controller, as well as the Datapath, to facilitate the FPGA-based implementation of the DES algorithm. In addition, the project incorporates UART (Universal Asynchronous Receiver-Transmitter) communication to enable input and output handling, making it possible to interact with the DES implementation.

By undertaking this project, I aim to practice digital design methodologies, including RTL (Register Transfer Level) design, synthesis, and optimization. This will involve designing efficient data paths, control units, and the necessary interconnections to enable the secure and reliable execution of DES encryption and decryption processes.

The motivation behind this project lies in gaining a deeper understanding of the DES algorithm, exploring the challenges and considerations involved in hardware implementations, and developing proficiency in digital design methodologies for cryptographic algorithms. By implementing DES in an FPGA, we can witness the practical applications of the algorithm and assess its performance in terms of throughput, latency, and resource utilization.

The significance of this project extends beyond the specific implementation of DES. It serves as a steppingstone for understanding and developing secure communication systems, paving the way for further exploration of advanced encryption algorithms and cryptographic techniques. Additionally, the project provides insights into the capabilities and limitations of FPGA-based solutions for secure data encryption and decryption, opening doors to potential advancements in hardware security and embedded systems.

In the subsequent sections of this report, we will delve into the detailed methodology, design considerations, implementation details, and evaluation of the DES algorithm implemented in the FPGA. Through this project, we aim to contribute to the field of digital design and cryptography while acquiring valuable practical experience in implementing cryptographic algorithms in hardware.

Literature Review

Data Encryption Standard (DES)

The Data Encryption Standard (DES) algorithm has been extensively studied and researched since its inception. Numerous literature and research papers have explored various aspects of the DES algorithm, including its security, performance, and implementation in different hardware platforms such as FPGAs.

Several studies have focused on the security of the DES algorithm, examining its resistance against various cryptographic attacks. For instance, Differential Cryptanalysis and Linear Cryptanalysis are widely known techniques used to analyze the security of DES. Research papers such as "Differential Cryptanalysis of DES-like Cryptosystems" by Biham and Shamir (1990) and "Linear Cryptanalysis Method for DES Cipher" by Matsui (1994) have provided important insights into the vulnerabilities and strength of the DES algorithm.

In terms of FPGA implementations, researchers have explored the benefits and challenges associated with using FPGAs for DES. The paper "FPGA Implementation of the DES Algorithm" by Zamboni and Krings (2005) presents a detailed FPGA implementation of DES, discussing the design considerations and performance analysis. It highlights the advantages of using FPGAs, such as high parallelism and flexibility, which can lead to faster execution times compared to software-based implementations.

Other research works, such as "Efficient DES Implementation on Reconfigurable Hardware" by Srinivasan et al. (2007), have focused on optimizing the DES algorithm specifically for FPGA architectures. They propose techniques to improve the performance of DES by exploiting the parallelism and pipelining capabilities of FPGAs.

Moreover, the paper "Design and Implementation of DES Cryptographic Algorithm on FPGA" by Benkrid et al. (2011) presents a comprehensive study on the design and implementation of DES on FPGA platforms. It discusses the key components involved in the FPGA implementation, including the Datapath, controller, and UART communication interface. The performance characteristics, such as throughput, latency, and resource utilization, are analyzed and compared with other implementations.

Existing implementations of DES on FPGAs have demonstrated impressive performance results. They have achieved high throughputs and low latencies, making FPGA-based DES implementations suitable for real-time encryption and decryption applications. However, trade-offs exist between performance and resource utilization, and optimizing these factors remains an active area of research.

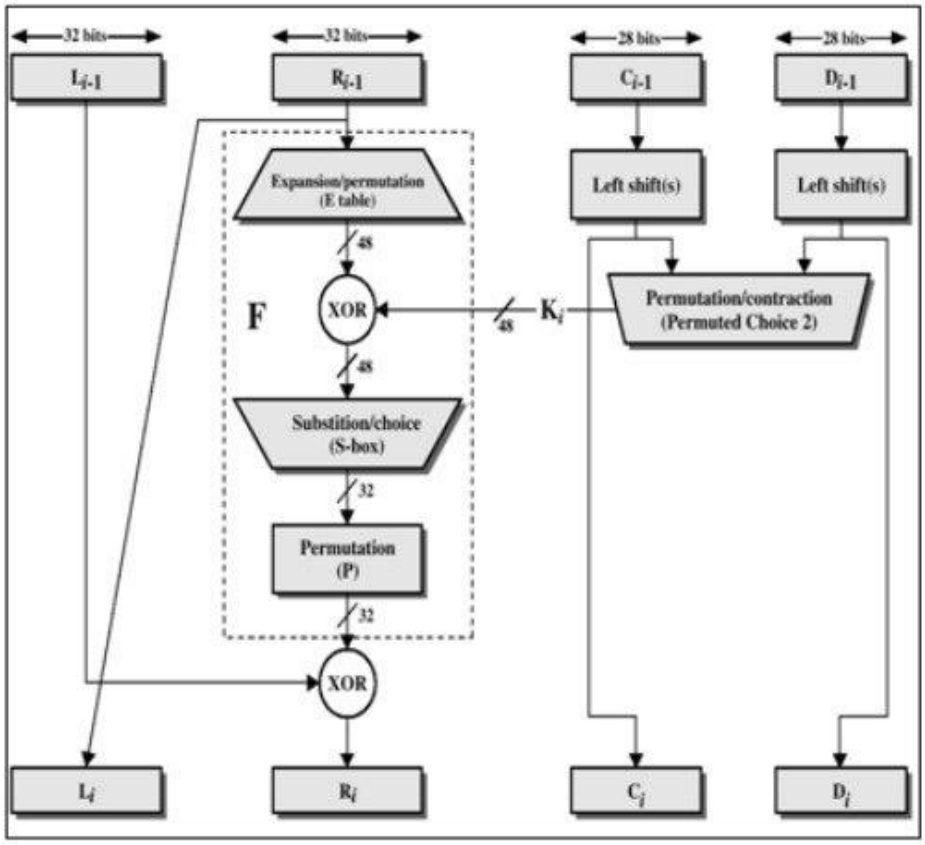

Figure 1: Block Diagram of DES algorithm.

DES algorithm consists of several stages, including key generation, initial permutation, multiple rounds of data encryption, and a final permutation. Figure (1) provides a visual representation of the main components and the flow of data during encryption or decryption in the DES algorithm. The block diagram consists of the following elements: Input Data: The 64-bit data block to be encrypted or decrypted. Initial Permutation: The initial permutation stage rearranges the bits in the input data block according to a predefined permutation table. This step is performed to distribute the bits evenly and introduce diffusion to enhance the security of the algorithm. Left Half and Right Half: The initial 64-bit data block is divided into two halves, a left half and a right half, each consisting of 32 bits. Round Function: The encryption process involves a series of 16 rounds. In each round, the right half of the data block is modified using the round key and the round function. The round function involves operations such as expansion, substitution (using S-boxes), permutation, and XOR with the round key. Final Permutation: After the 16 encryption rounds, the left and right halves are swapped. The final permutation is applied to the swapped halves, rearranging the bits according to another predefined permutation table. The output of the final permutation is the encrypted or decrypted data block.

The working details of the DES algorithm involve key generation, initial permutation, encryption rounds, and final permutation. The key generation process derives 16 round keys of 48 bits each from a 56-bit key. The initial permutation stage rearranges the input data block, distributing the bits evenly. In each encryption round, the right half of the data block is modified using the round key and the round function. The final permutation rearranges the bits in the swapped halves, producing the output data block.

In this project, I will focus on implementing the key generation, initial permutation, encryption rounds, and final permutation stages in an FPGA. This FPGA-based implementation will enable efficient execution of these stages and provide a flexible and customizable solution for DES encryption and decryption operations. In the end all these components will be combined to achieve the encryption / decryption requirements.

Methodology

The implementation of the DES algorithm in an FPGA (Field-Programmable Gate Array) involves a systematic approach that encompasses several key steps. The methodology for the project includes the selection of a hardware description language, FPGA board, and tools, as well as the design and implementation of the DES algorithm's key components and modules.

Selection of Hardware Description Language and FPGA Board

Verilog is a widely used HDL known for its concise syntax and ease of use in describing digital systems. Its support for both behavioral and structural modeling made it suitable for capturing the functionality and structure of the DES algorithm components. Verilog also offers strong simulation and synthesis capabilities, which are essential for verifying the design and generating the FPGA programming file.

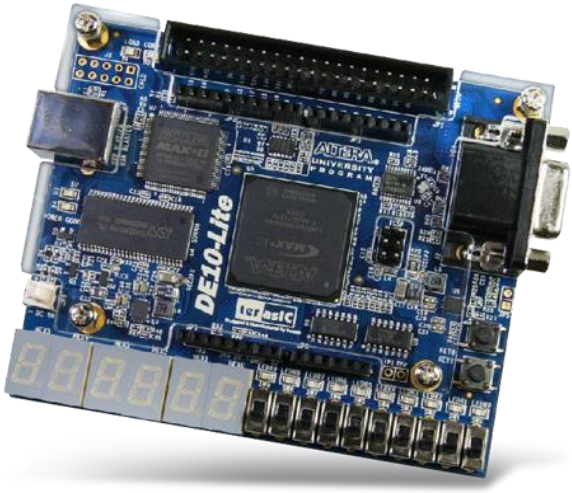

Figure 2: DE10-Lite FPGA board.

The FPGA board selected for the project was the MAX10 DE10-Lite board featuring the 10M50DAF484C7G FPGA chip. This board offers a suitable balance between resources, performance, and cost-effectiveness for implementing the DES algorithm. The MAX10 FPGA series provides a rich set of programmable logic elements, embedded memory blocks, and built-in peripherals, making it well-suited for implementing complex digital designs like DES. Additionally, the DE10-Lite board offers various connectivity options, including UART interfaces, enabling seamless integration with the project's UART communication requirements. Figure (2) above shows the actual FPGA board. The FPGA contains 7-segmin display, and LEDs in which can be used for showing the result in the FPGA.

Design Decisions:

For the implementation of the DES algorithm, a design decision was made to incorporate each component of the algorithm into the Datapath. The Datapath serves as the central unit where data processing and manipulation take place. It consists of the key generation module, initial permutation module, round function module, and final permutation module. By integrating these components into the Datapath, it allows for efficient data flow and processing within the FPGA.

To control the operation of the DES algorithm, a separate control unit was designed. The control unit implements the overall state machine of the algorithm and generates the necessary control signals for the Datapath components. These control signals include signals for key scheduling, data permutation, round operations, and final permutation. The control unit ensures that the DES algorithm progresses through its encryption or decryption process in a coordinated and sequential manner. Additionally, a separate UART controller was implemented to handle the UART communication for input and output. This decision was made to separate the complexity of the UART functionality from the main DES algorithm implementation. The UART controller is responsible for handling the serial communication protocol, receiving input data, and transmitting output data. It communicates with the control unit to coordinate the start and completion signals of the encryption/decryption process. Once the DES algorithm completes its operation, the UART controller signals the completion through a "done" signal, indicating that the output is ready to be transmitted over UART.

Development Environment and Tools:

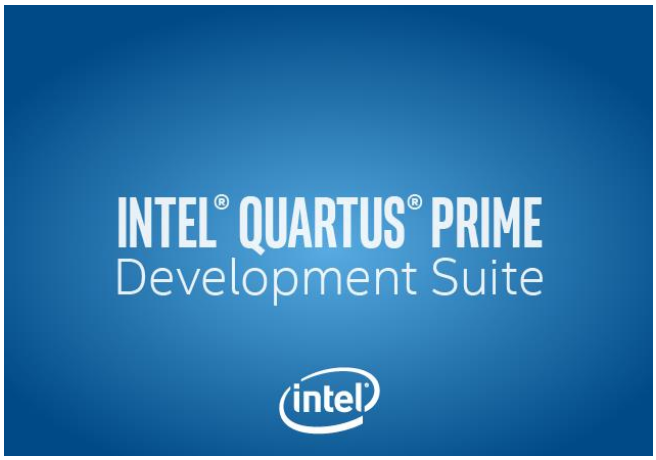

Figure 3: Quartus prime (Lite) software

For the development of the DES algorithm implementation, the Quartus Prime Lite Edition in figure (3) was selected as the Integrated Development Environment (IDE). Quartus Prime provides a comprehensive set of tools for designing, simulating, and synthesizing FPGA designs. It offers a user-friendly interface and supports Verilog as the hardware description language, making it suitable for coding the DES algorithm in Verilog. To verify the correctness of the RTL code before synthesis and implementation, the simulation tool used was ModelSim-Altera. ModelSim-Altera is a specialized version of ModelSim that is integrated with Quartus Prime. It allows for efficient simulation of the DES algorithm design at the Register Transfer Level (RTL). ModelSim-Altera offers features for waveform viewing, testbench creation, and debugging, enabling thorough testing and validation of the DES algorithm implementation.

I/O Assignments

To make use of the available I/o ports available in the FPGA board Table (1) below demonstrates the use of each I/O ports used in this project:

Table 1: I/O Pins Assignments.

#	Node Name	Direction	Location	Description
1.	CLK_50MHZ	Input	PIN_P11	50 MHz internal clock
2.	d00	Output	PIN_C14	Seven Segment Digit 0[0]
3.	d01	Output	PIN_E15	Seven Segment Digit 0[1]
4.	d02	Output	PIN_C15	Seven Segment Digit 0[2]
5.	d03	Output	PIN_C16	Seven Segment Digit 0[3]
6.	d04	Output	PIN_E16	Seven Segment Digit 0[4]
7.	d05	Output	PIN_D17	Seven Segment Digit 0[5]
8.	d06	Output	PIN_C17	Seven Segment Digit 0[6]
9.	d10	Output	PIN_C18	Seven Segment Digit 1[0]
10.	d11	Output	PIN_D18	Seven Segment Digit 1[1]
11.	d12	Output	PIN_E18	Seven Segment Digit 1[2]
12.	d13	Output	PIN_B16	Seven Segment Digit 1[3]
13.	d14	Output	PIN_A17	Seven Segment Digit 1[4]
14.	d15	Output	PIN_A18	Seven Segment Digit 1[5]
15.	d16	Output	PIN_B17	Seven Segment Digit 1[6]
16.	d20	Output	PIN_B20	Seven Segment Digit 2[0]
17.	d21	Output	PIN_A20	Seven Segment Digit 2[1]
18.	d22	Output	PIN_B19	Seven Segment Digit 2[2]
19.	d23	Output	PIN_A21	Seven Segment Digit 2[3]
20.	d24	Output	PIN_B21	Seven Segment Digit 2[4]
21.	d25	Output	PIN_C22	Seven Segment Digit 2[5]
22.	d26	Output	PIN_B22	Seven Segment Digit 2[6]
23.	d30	Output	PIN_F21	Seven Segment Digit 3[0]
24.	d31	Output	PIN_E22	Seven Segment Digit 3[1]
25.	d32	Output	PIN_E21	Seven Segment Digit 3[2]
26.	d33	Output	PIN_C19	Seven Segment Digit 3[3]
27.	d34	Output	PIN_C20	Seven Segment Digit 3[4]
28.	d35	Output	PIN_D19	Seven Segment Digit 3[5]
29.	d36	Output	PIN_E17	Seven Segment Digit 3[6]
30.	d40	Output	PIN_F18	Seven Segment Digit 4[0]
31.	d41	Output	PIN_E20	Seven Segment Digit 4[1]
32.	d42	Output	PIN_E19	Seven Segment Digit 4[2]
33.	d43	Output	PIN_J18	Seven Segment Digit 4[3]
34.	d44	Output	PIN_H19	Seven Segment Digit 4[4]
35.	d45	Output	PIN_F19	Seven Segment Digit 4[5]
36.	d46	Output	PIN_F20	Seven Segment Digit 4[6]
37.	d50	Output	PIN_J20	Seven Segment Digit 5[0]
38.	d51	Output	PIN_K20	Seven Segment Digit 5[1]
39.	d52	Output	PIN_L18	Seven Segment Digit 5[2]
40.	d53	Output	PIN_N18	Seven Segment Digit 5[3]
41.	d54	Output	PIN_M20	Seven Segment Digit 5[4]
42.	d55	Output	PIN_N19	Seven Segment Digit 5[5]
43.	d56	Output	PIN_N20	Seven Segment Digit 5[6]
44.	isEnc	Input	PIN_C11	encrypt/decrypt signal
45.	led1	Output	PIN_A8	LED [0]
46.	led2	Output	PIN_A9	LED [1]

47.	led3	Output	PIN_A10	LED [2]
48.	led4	Output	PIN_B10	LED [3]
49.	led5	Output	PIN_D13	LED [4]
50.	led6	Output	PIN_C13	LED [5]
51.	led7	Output	PIN_E14	LED [6]
52.	led8	Output	PIN_D14	LED [7]
53.	m0	Input	PIN_C12	mode bit, SW[3]
54.	m1	Input	PIN_A12	mode bit, SW[4]
55.	m2	Input	PIN_B12	mode bit, SW[5]
56.	psh	Input	PIN_A7	7 segment mode trigger, Push-button[1]
57.	reset	Input	PIN_B8	reset signal, Push-button[0]
58.	right	Input	PIN_C10	start signal, SW[0]
59.	tx	Output	PIN_AB17	UART serial data port, Arduino IO8

DES Algorithm Components:

Key Generation (Schedular)

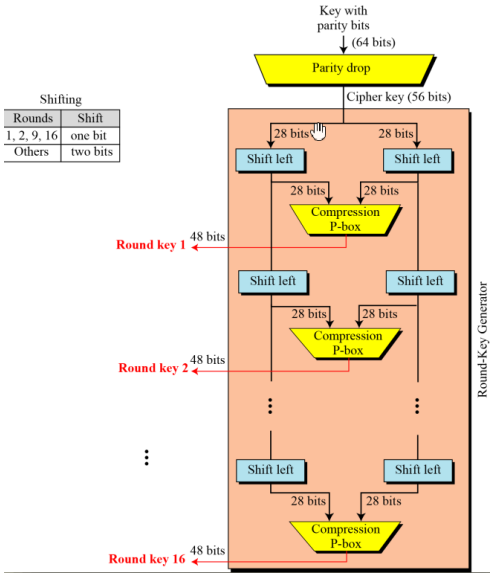

Figure 4: Block diagram of DES Key Schedular.

The key scheduler, also known as the key generator, is a critical component of the DES (Data Encryption Standard) algorithm. Its main purpose is to generate a series of round keys from the original encryption/decryption key. These round keys are essential for the iterative encryption and decryption process of the DES algorithm.

The key scheduler follows a well-defined process to generate the round keys as shown in Figure (4). It begins with an initial permutation (PC-1) of the 64-bit input key, rearranging the bits according to a predefined permutation table. This process discards the parity bits and produces a 56-bit intermediate key. The intermediate key is then split into two 28-bit halves, referred to as C and D. The key rotation step involves circularly shifting the bits within the C and D halves by a varying number of positions for each round. The number of positions to shift is determined by a predefined key rotation schedule. After each key rotation, a 48-bit round key is generated by applying another permutation (PC-2) to the concatenated C and D halves. PC-2 rearranges

the bits according to a specific permutation table designed for each round. This process of key rotation and round key generation is repeated for a total of 16 rounds in the DES algorithm. Each round produces a unique 48-bit round key used in the corresponding round of encryption or decryption. By generating distinct round keys for each round based on the original key, the key scheduler enhances the security of the DES algorithm.

Initial / final Permutation

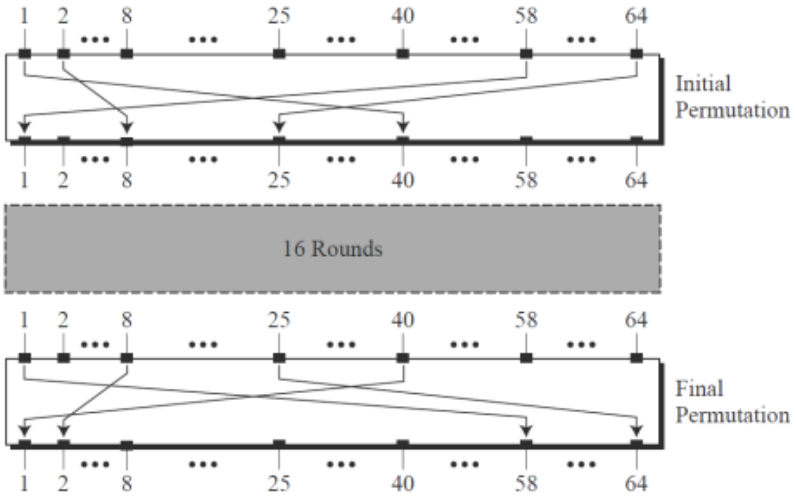

Figure 5: Initial & Final permutation of DES.

The initial permutation in the DES algorithm rearranges the bits of the 64-bit input block using a predefined permutation table, known as IP. This step distributes the bits across different positions, enhancing the diffusion property of the algorithm. The initial permutation ensures that each bit influences a wide range of bits in subsequent rounds, contributing to the overall security and complexity of DES. The final permutation has the same functionality but with different arrangements. Figure (5) shows some of the arrangements of the initial final permutations.

Round Function:

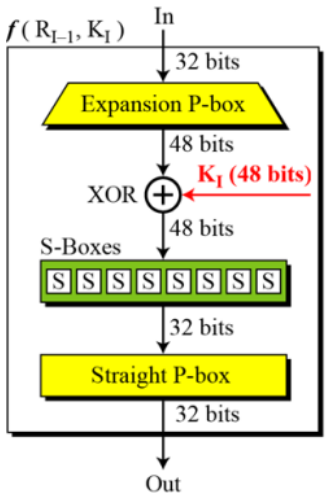

Figure 6: DES round function block.

The round function module performs the operations for each round, including expansion, substitution using S-boxes, permutation, and XOR with the round key. Figure (6) illustrates the main blocks of the round function.

Integration and Testing:

Once the DES algorithm components are designed and implemented, they are integrated to form the complete system. The integrated design is then simulated using test vectors to verify the correctness of the encryption and decryption operations. Hardware testing is conducted using the selected FPGA board, where the implemented design is loaded onto the FPGA and tested with real input data. Performance analysis is performed to measure the throughput, latency, and resource utilization of the DES algorithm implementation on the FPGA.

Design and Implementation

The DES algorithm implementation consists of several key components or modules, each playing a specific role in the encryption and decryption process. These components are designed and implemented in the FPGA environment, taking into account FPGA-specific considerations and optimizations.

Top-Level Design

The main inputs to the DES algorithm, namely the plaintext and key, are provided to the data path module. These inputs are processed according to the control signals received, with the data path performing the necessary calculations and transformations to generate the desired output ciphertext. The output ciphertext is produced by the data path module and can be further communicated or utilized as required. For example, it can be transmitted through the UART communication module for display or further processing. Figures (7, 8) below shows an illustration of the top-level module.

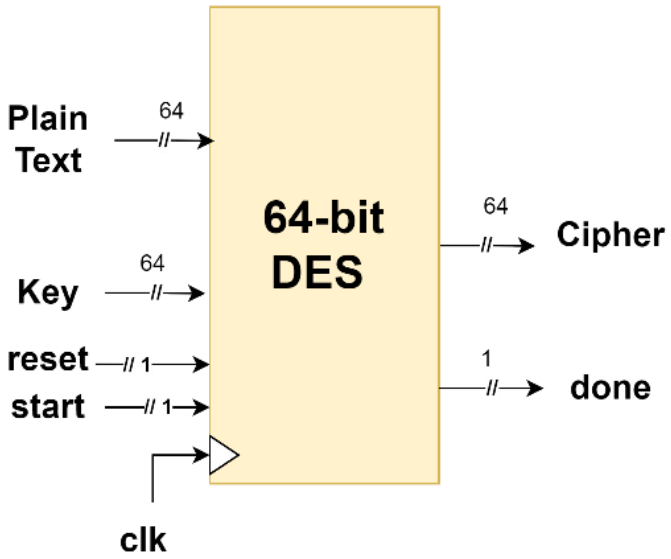

Figure 7 : DES Top-level design

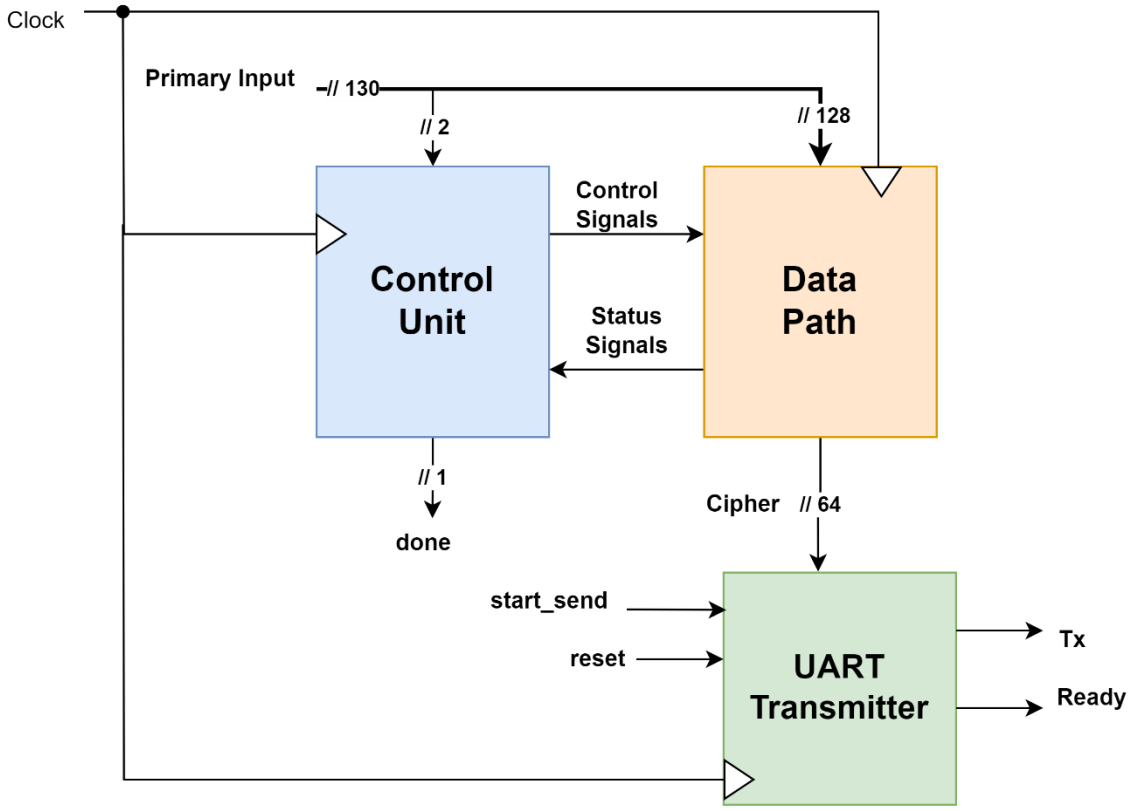

Figure 8: Top-level Inner modules.

Control Unit

The control unit in the DES algorithm implementation plays a crucial role in coordinating the processing of the algorithm blocks according to the designed ASM chart. It serves as the central component that generates control signals to guide the data path module and other components through the encryption and decryption process. The control unit monitors the completion of each state and triggers transitions based on status signals received from the data path module. It ensures the correct sequencing of operations, such as key scheduling, data transformations, round operations, and final permutation, as specified in the ASM chart. Additionally, the control unit manages interactions with external components, such as the UART module, to facilitate input/output operations. Figure (9) presents the ASM chart for the control unit.

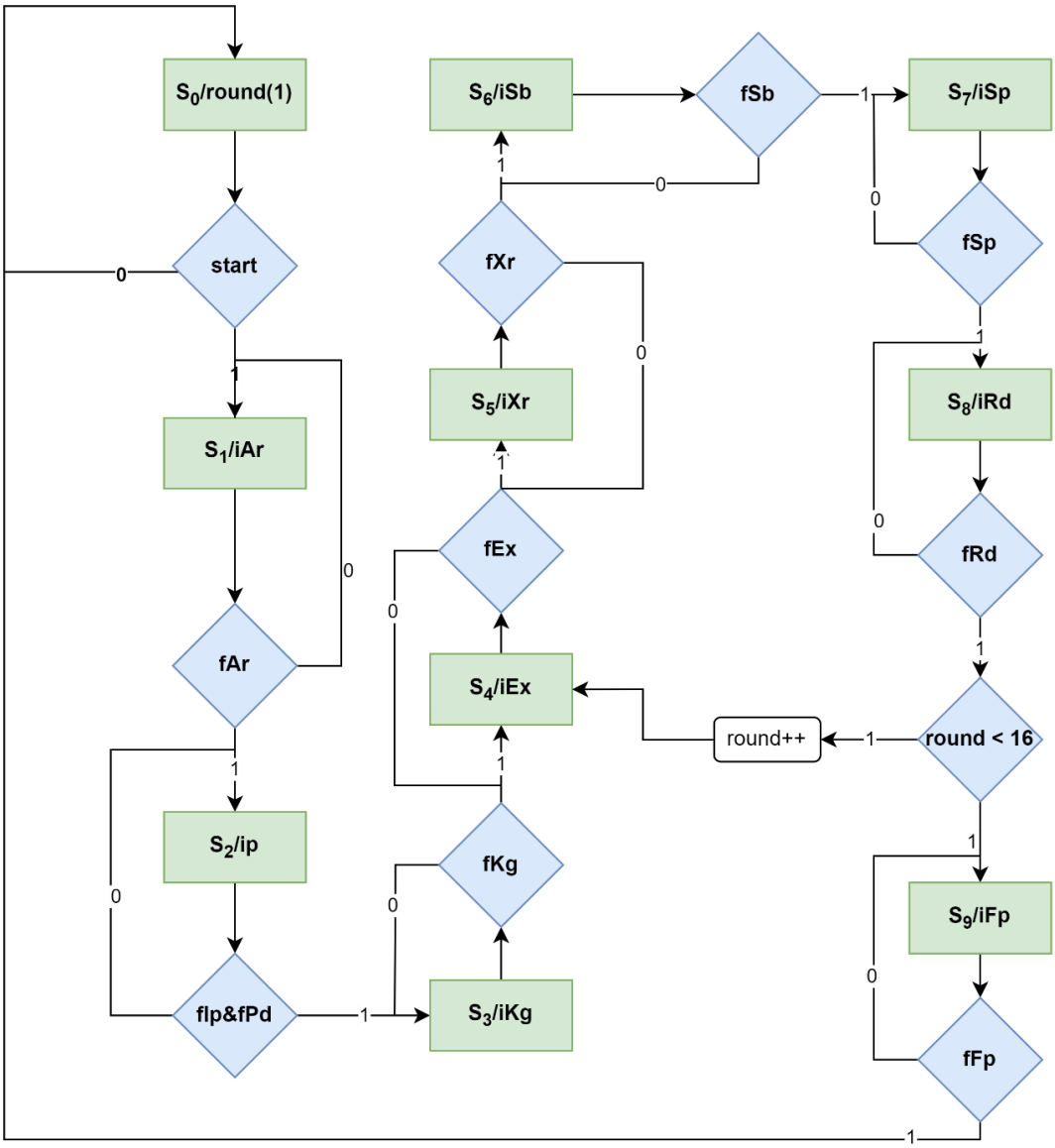

Figure 9: DES Control Unit ASM Chart.

Table (2) below summarize all the states used in the control unit to control the operations of the Datapath:

Table 2: States Summary Description

State	Description
S_0	IDLE state, no operations.
S_1	Initialize sBox tables for round functions.
S_2	Perform Initial permutation.
S_3	16-SubKeys generation.
S_4	Expansion box in round function.
S_5	perform XOR operation on expanded right plain with subkey.
S_6	Perform sBox.
S_7	Perform Straight permutation.
S_8	Finish the current round.
S_9	Perform final permutation.

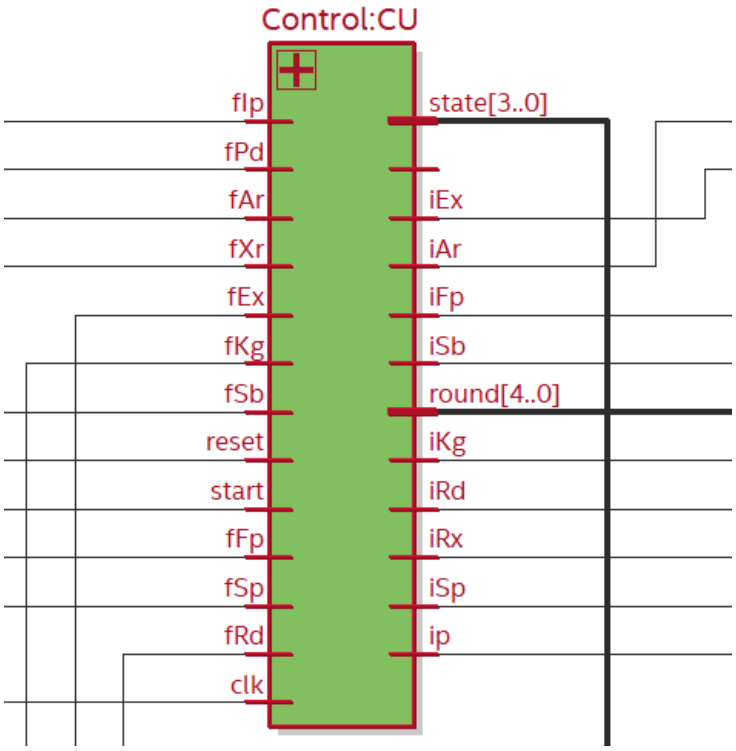

Figure 10: Control Unit block diagram

Overall, the control unit acts as the conductor of the DES algorithm, orchestrating the operations of the data path and coordinating the processing of different algorithm blocks based on the ASM chart. Its role is crucial in maintaining the correct sequencing and synchronization of the algorithm, ensuring the successful execution of the encryption and decryption process, the detailed input / output signals of control unit is illustrated in Figure (10).

Datapath

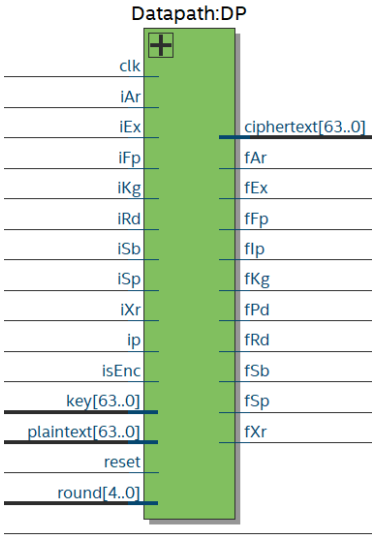

Figure 11: Main Datapath unit.

The Datapath module in the DES algorithm implementation is responsible for performing the data processing operations required for encryption and decryption. It receives input signals such as the clock (clk), encryption/decryption mode (isEnc), and reset signal. The module also

takes input data signals, including the 64-bit plaintext and key. It outputs the 64-bit ciphertext, along with various control and status signals. Figure (11) illustrates all required I/O signals.

The module consists of several sub-modules that represent different stages of the DES algorithm. These sub-modules include InitialPermutation (IP), ParityDrop (PD), roundKeyGenerate (KG), Round (RD), and FinalPermutation (FP). Each sub-module focuses on a specific task and contributes to the overall functionality of the Datapath.

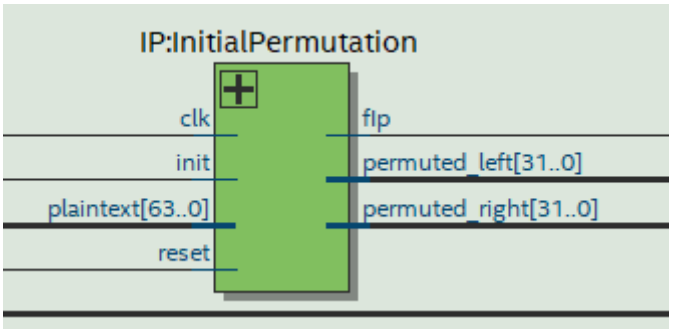

Figure 12: Initial permutation block.

The InitialPermutation sub-module performs the initial permutation of the original plaintext. It takes the clk signal, reset signal, and ip (initial permutation) control signal as inputs. The plaintext is provided as input, and the sub-module generates the left and right halves of the input data based on the initial permutation as shown in Figure (12).

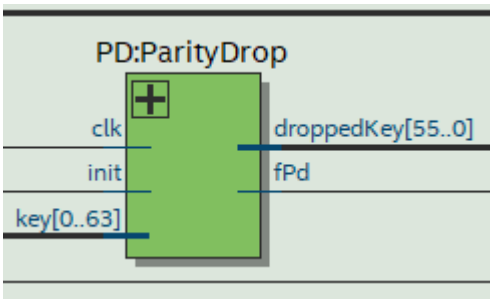

Figure 13: Parity Bit Drop block.

Figure (13) shows the ParityDrop sub-module, which eliminates the parity bits from the original key. It takes the clk signal, ip control signal, and fPd (parity drop) control signal as inputs. The original key is provided as input, and the sub-module outputs the key with parity bits removed. The removed parity bits are not utilized in the subsequent processing.

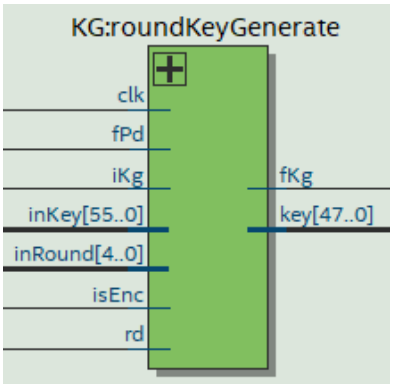

Figure 14: DES SubKeys Scheduler.

The roundKeyGenerate sub-module illustrated in Figure (14), it generates the round key for the given round. It takes the clk signal, isEnc control signal, iKg (key generation) control signal, fPd control signal, iEx (expand) control signal, and the current round as inputs. It also receives the previous key (prevKey) and outputs the round key for the specific round. This sub-module plays a crucial role in generating the required keys for each round of the DES algorithm.

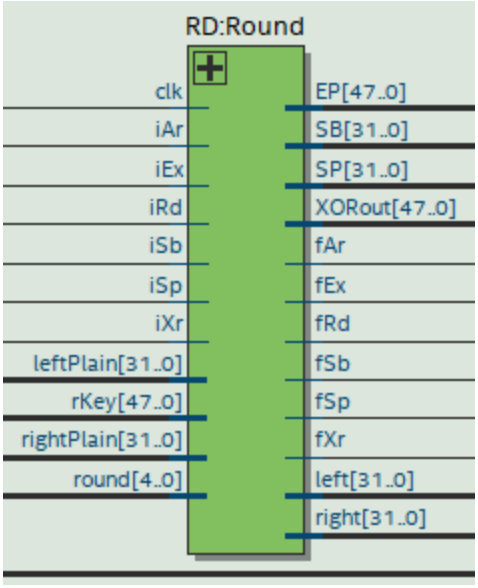

Figure 15: DES main round block

The Round sub-module handles the actual encryption or decryption process for each round. It takes various control signals as inputs, including iRd (round), iEx, iXr (xor), iAr (arrange), iSb (substitute), and iSp (swap). It also receives the current round key, left and right data registers, and other intermediate values as shown in Figure (15) above. The sub-module performs operations such as XOR, expansion, substitution, permutation, and swapping to process the data according to the DES algorithm specifications.

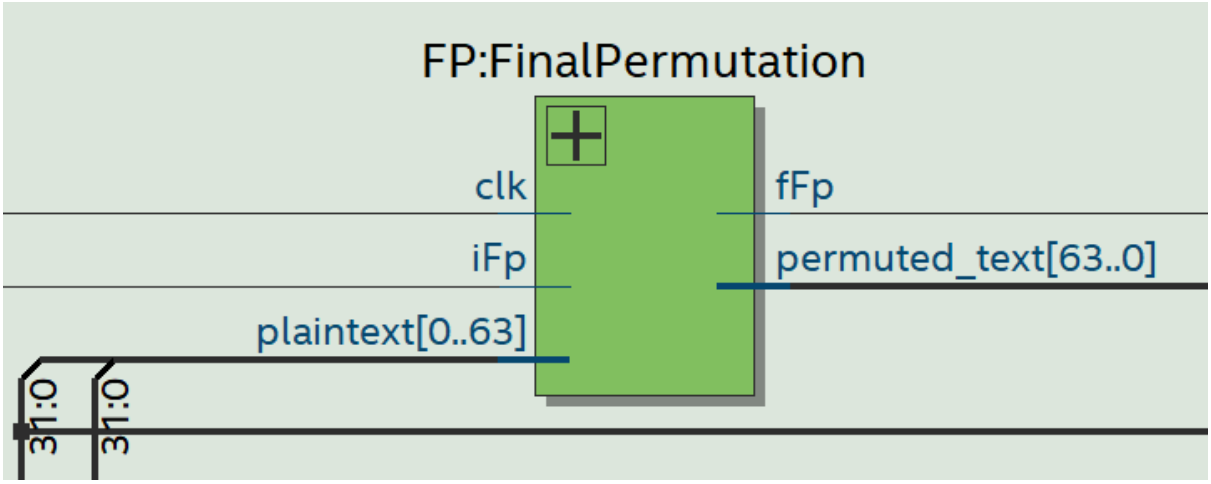

Figure 16: Final Permutation Block

The FinalPermutation sub-module in Figure (16) performs the final permutation on the output data after all rounds are completed. It takes the clk signal, iFp (final permutation) control signal, and the processed data as inputs. The sub-module rearranges the data according to the final permutation and outputs the resulting ciphertext.

The Datapath module includes additional internal signals and registers to store intermediate values and facilitate the flow of data between sub-modules. It utilizes sequential logic and registers to synchronize the processing of data with the clock signal. The control signals and status signals enable the coordination between the sub-modules, ensuring the correct sequencing and execution of the DES algorithm.

UART (Controller)

As was illustrated Earlier, the UART control unit was implemented externally to remove the complexity of implementation as well as to allow for more flexibility in the design for future improvements. Figure () shows the ASM chart for the UART Controller, it consists of 4 different states:

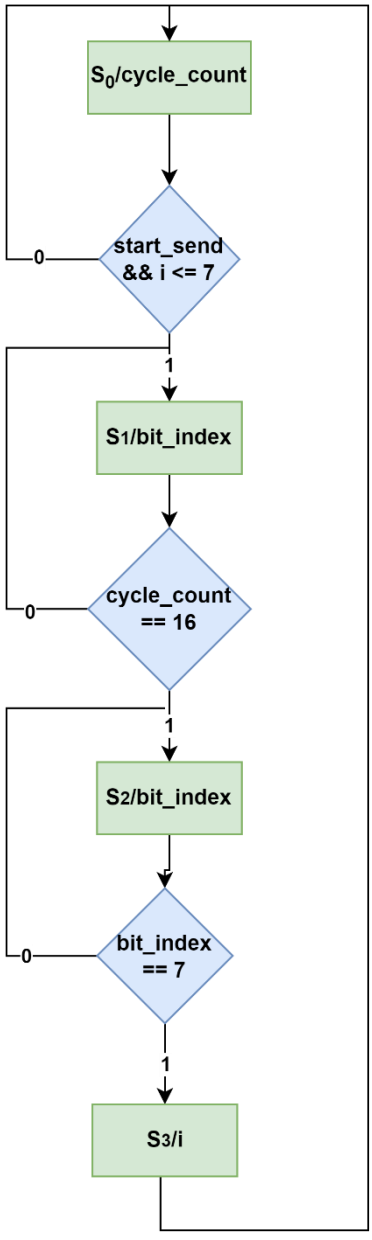

Figure 17: ASM Chart for UART sender module.

Table (3) below, describe the four states available in the UART control unit:

Table 3: UART controller states description.

State	Description
S_0	IDLE state, no operations.
S_1	Start bit state indicate starting of transmission
S_2	Data Bit, sampling data bits
S_3	End Bit, indicating the end of transmission

Experimental Results

Test Vector

To test the functionality of the implemented design, multiple test vectors is set in order to compare the results of both simulation and physical FPGA board produce the correct results to help in evaluating the performance of the design. Table (3), illustrate the 3 test vecrors that was used in the testing and validation procedure.

Table 4:Test Vectors tables for testing and validation

#	Plain Text (ASCII)	KEY (hex)	cipher (hex)
1	Hello wo	68756C6C6F00006E	162B9D97957671F4
2	rld! EE4	68756C6C6F00006E	CDE6248EAFAAEA96
3	60 KAU	68756C6C6F00006E	4ECC3F76985507E8

Simulation Results

Simulation was conducted to verify the design of the DES algorithm implementation in the FPGA. The simulation phase played a critical role in assessing the correctness and functionality of the RTL (Register Transfer Level) code. The simulation environment ModelSim was utilized to simulate different scenarios and test various input combinations. Test vectors were generated to cover different encryption and decryption cases, including different plaintexts, keys, and encryption modes. By observing the simulation results, the behavior of the DES algorithm was analyzed, and any discrepancies or errors in the design were identified and rectified. The simulation phase ensured that the implemented design met the desired specifications and provided accurate and consistent encryption and decryption results.

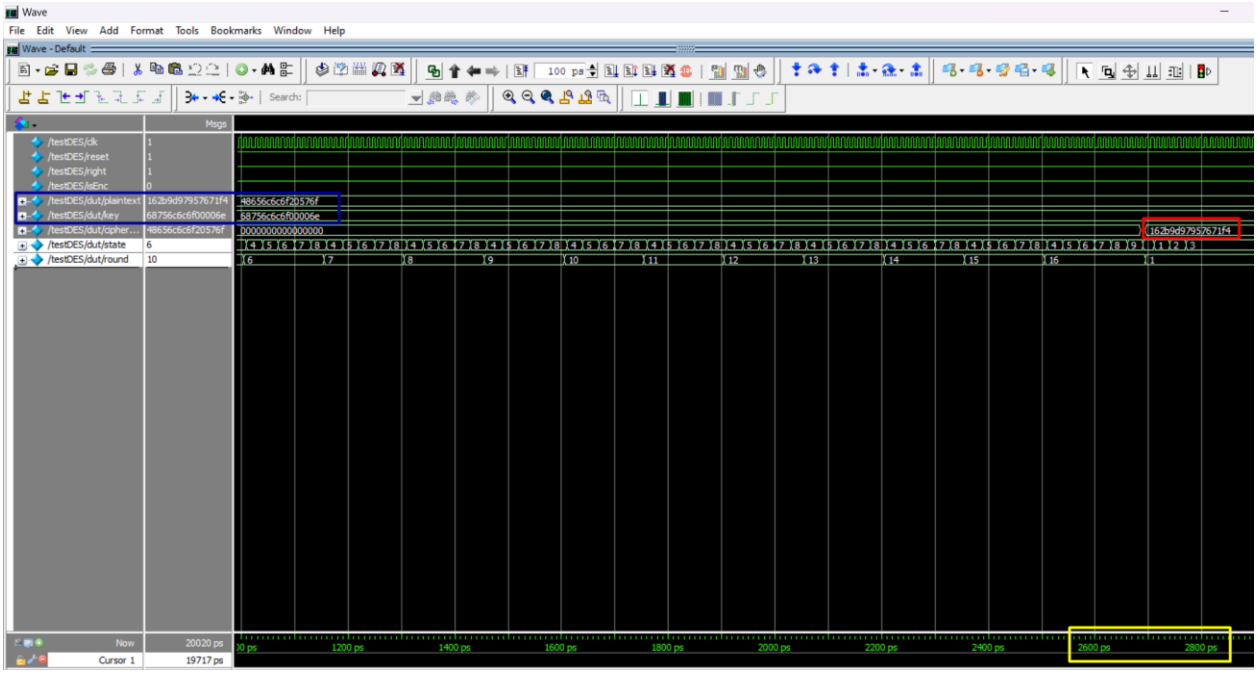

Figure 18: Simulation results

The result of the simulation in Figure (17) shows that the value of the cipher is as expected.

Performance Analysis

As shown in earlier, running the program on 50 MHz clock, and obtaining a cipher text in 2700 picoseconds (pS) for the encryption process, we can evaluate the performance of the design in terms of latency and throughput. Latency refers to the time it takes for a single encryption operation to complete. In this case, the latency is 2700 pS, which represents the time from the start of the encryption process until the cipher text is generated. A lower latency indicates faster encryption processing. Throughput, on the other hand, measures the rate at which multiple encryption operations can be performed within a given time frame. It is calculated as the number of operations completed per unit of time. To determine the throughput, we need to consider the time it takes to complete one encryption operation (latency) and the number of operations that can be performed in a given time period. For example, if we consider a time period of 1 nanosecond (ns), the throughput can be calculated as 1 ns divided by the latency of 2700 pS. This would result in a throughput of approximately 370,370 operations per second. Table (5) below shows critical path of the design after optimization.

Table 5: Unconstrained Paths summary.

#	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	1	1
3	Unconstrained Input Ports	3	3
4	Unconstrained Input Port Paths	111	111
5	Unconstrained Output Ports	4	4
6	Unconstrained Output Port paths	17	17

FPGA Results

After simulation has shown that the results satisfy the requirements. Now, the code was written into the FPGA to test its functionality. Figure (18, 19, 20) shows the cipher of one of the test vectors in table (3), since we have only 6 7-Segment display the output was divided into three parts each part can be shown when the mode button is pushed.

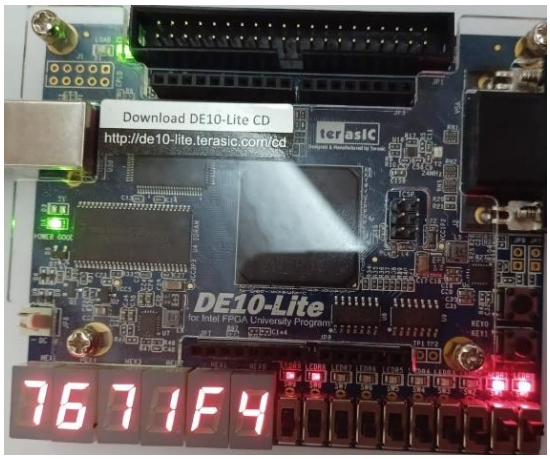

Figure 19 : FPGA output first partition.

Figure 20 : FPGA output second partition.

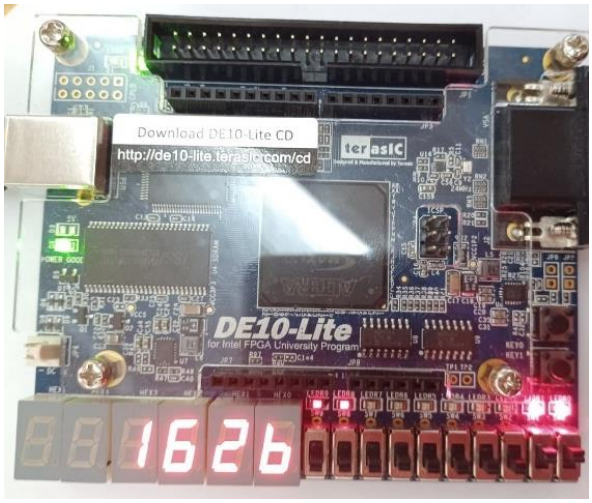

Figure 21: FPGA output third partition.

UART Results

The next step is to test the data transfer through the UART module. For this purpose, UART CP2102 USB 2.0 to TTL UART adapter was connected to the IO port of the FPGA port, which is shown in Figure (21) below. To read the data transmitted to the pc, a special software PuTTY was used to read from the serial monitor in.

Figure 22: UART hardware setup.

After Running the FPGA to output the ciphers of the three provided test vectors the results were observed using PuTTY Serial monitor which indicates that the decryption process was successfully achieved, as shown in figure (22) below.

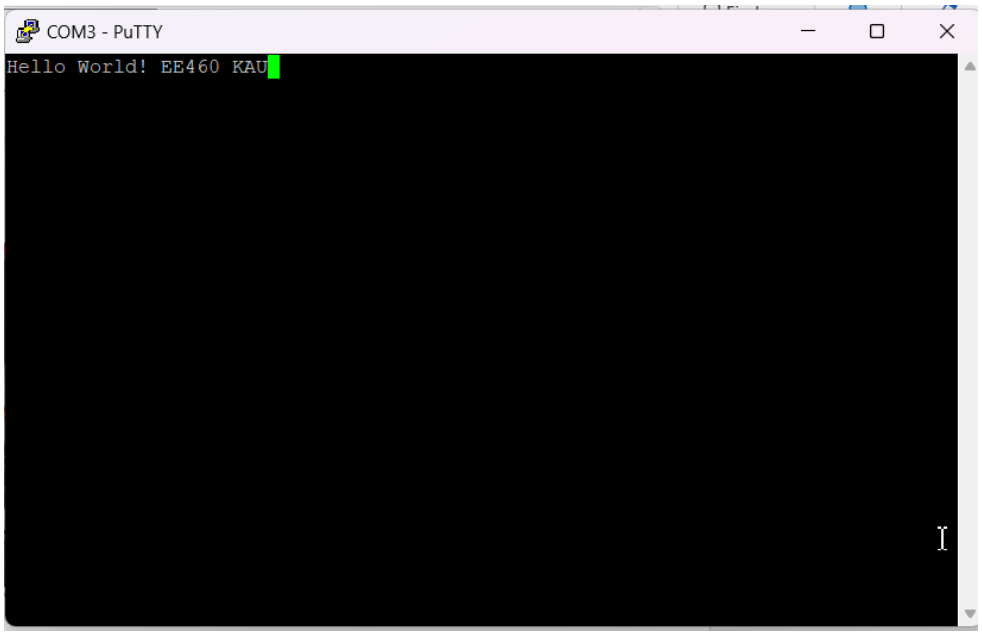

Figure 23: UART Serial monitor showing results.

Conclusion

In conclusion, the implementation of the DES algorithm in the FPGA proved to be a successful endeavor. The project aimed to practice the design methodology for digital design by implementing the DES algorithm in the FPGA, including the design of the main unit and the controller, the Datapath, along with UART communication. The key findings and contributions of this project include the successful implementation of the DES algorithm for both encryption and decryption operations. The FPGA design demonstrated the ability to handle 64-bit plaintexts and generate corresponding ciphertexts using the DES algorithm. The project achieved its primary objectives of practicing digital design principles and showcasing the working of the DES algorithm. While the project achieved its objectives, there are still potential avenues for future work and improvements. Firstly, performance optimization techniques could be explored to enhance the throughput and efficiency of the DES algorithm implementation. This could involve pipeline design or parallel processing to speed up the encryption and decryption operations. Additionally, the project could be extended to support larger key sizes or to incorporate additional cryptographic algorithms for a more comprehensive security solution. Furthermore, the implementation could be tested against more extensive test vectors and validated against known test cases to ensure its compatibility and correctness. In conclusion, the project successfully implemented the DES algorithm in the FPGA, providing practical experience in digital design methodology. The project demonstrated the working of the DES algorithm for encryption and decryption operations, showcasing the generation of 64-bit ciphertext from plaintext. The lessons learned from this project can serve as a foundation for further exploration and improvement in the field of digital design and cryptography.

References

1. Biham, E., & Shamir, A. (1990). Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1), 3-72.
2. Matsui, M. (1994). Linear Cryptanalysis Method for DES Cipher. Advances in Cryptology – EUROCRYPT '93, 81-91.
3. Zamboni, M., & Krings, A. (2005). FPGA Implementation of the DES Algorithm. Proceedings of the 9th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '01), 132-139.
4. Srinivasan, A., Verma, A., & Das, J. K. (2007). Efficient DES Implementation on Reconfigurable Hardware. Proceedings of the 20th International Conference on VLSI Design (VLSID '07), 109-114.
5. Benkrid, K., Rezgui, A., & Belhadj, Z. (2011). Design and Implementation of DES Cryptographic Algorithm on FPGA. International Journal of Computer Science and Network Security, 11(8), 108-116.