SRH University Heidelberg

Professional Technical Report

Hardware Implementation of Decryption By The PRESENT Block Cipher Using FPGA

Author:	PATIL, SUYOG
Matriculation Number:	11010668
Email Address:	Suyogp2112@gmail.com
Email Address:	Suyog.Patil@stud.hochschule-heidelberg.de

Supervisor:	Prof. Dr.-Ing Dejan Lazich

Contents

List of Figures

List of Tables

Chapter 1

Abstract

PRESENT is called a substitution-permutation network [SPN] with a block size of 64 bits and two different key sizes: 80 or 128 bits. PRESENT is of both practical and theoretical interest since its implementation requirements are similar to that of compact stream ciphers. Though PRESENT was originally designed with a minimal hardware footprint in mind, it is well suited for high-speed and high-throughput applications. Especially its hardware efficiency, i.e. the throughput per slice, is noteworthy.

Our goal is to design the decryption circuit of PRESENT as an 80 bit block cipher using Quartus II. We have synthesized PRESENT in modular form for better understanding and to avoid complexity.

Chapter 2

Introduction

PRESENT is a lightweight block cipher, developed in 2007 by the Orange Labs, Ruhr University Bochum and Technical University of Denmark. PRESENT was designed by Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin and C. Vikkelsoe.

The algorithm is noted for its compact size which is about 2.5 times smaller than Advanced Encryption Standard [AES].The block size is 64 bits and the key size can be 80 bit or 128 bit. The non-linear layer is based on a single 4-bit S-box which was designed with hardware optimizations in mind. PRESENT is intended to be used in situations where low-power consumption and high chip efficiency is desired.

The International Organization for Standardization and the International Electrotechnical Commission included PRESENT in the new international standard for lightweight cryptographic methods.

2.1 Block Diagram of Present Cipher Decryption

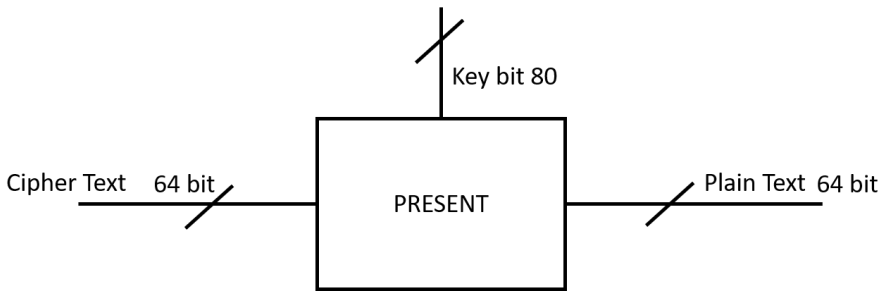

Figure 2.1: Block Diagram of Present Cipher Decryption.

PRESENT has 31 regular rounds and a final round that only consists XOR operation. One regular round consists of a key mixing step, a Inverse substitution layer, and Inverse permutation layer.

Chapter 3

Decryption

Decryption in PRESENT Cipher uses the same algorithm as Encryption, and hence it is an entirely reversible process.The only difference here would be that the application of sub-keys is reversed.

There are 3 main functions that occur in decryption of PRESENT:
1. Add Round Key
2. Inverse Substitution Layer
3. Inverse Permutation Layer.

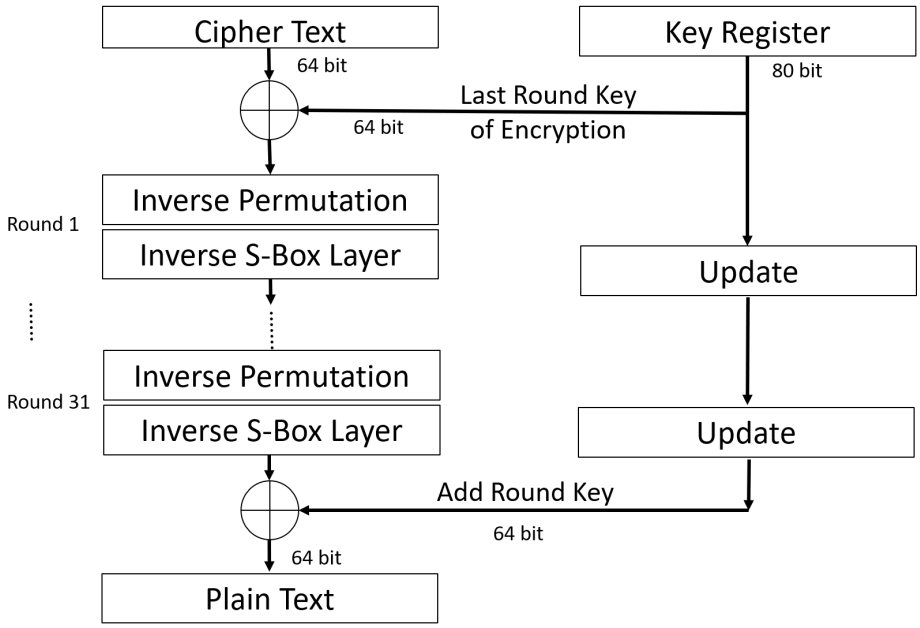

Figure 3.1: Overview of PRESENT Decryption Algorithm.

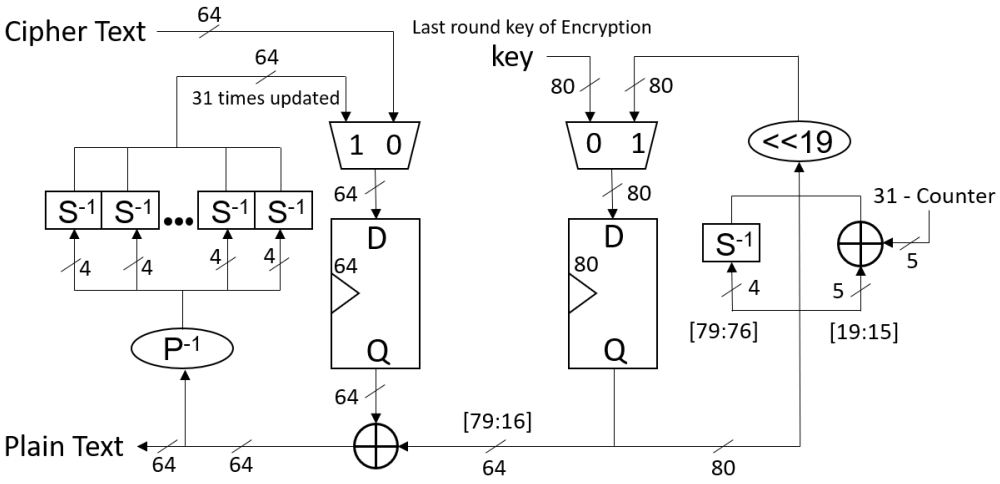

Figure 3.2: Decryption.

3.1 Add Round Key Layer

For Decryption, the add Round Key is similar as in the encryption routine. Given round key $Ki = K^i_{63}.....K^i_0$ for $1 \leq i \leq 32$ and current STATE $b_{63}...b_0$, add Round Key consists of the operation $0 \leq j \leq 63$.

In this layer, corresponding bits of data $(b63, b62, ...b0)$ and the round key $(k63, k62, ...k0)$ for the given round are XORed in a reverse order for decryption.
$b_j \rightarrow b_j \oplus K^i_j$

3.2 Inverse Substitution Layer

The inverse S-box used in PRESENT is a 4-bit to 4-bit S-box S: $F2^4 \rightarrow F2^4$. The action of this box in hexadecimal notation is given by the following table.

The inverse substitution layer consists of 20 logic gates and 4-bit input and 4-bit output (4x4): S: $F2^4 \rightarrow F2^4$. The S-box is given in hexadecimal notation according to the Table 3.1 The bit permutation used in PRESENT is given by Table 3.1. Bit i of STATE is moved to bit position P(i). In other words, the bit position 0 of the input to the Inverse P-Layer is moved to position 0, the bit position 1 of the input to the Inverse P-Layer is moved to position 4, the bit on position 2 is moved to position 8 and so on.

For getting Inverse S-box and Inverse Permutation Layer from S-box and Permutation Layer we have flipped data of S-box and Permutation Layer and rearrange it in proper order as shown in below table.

Table 3.1: S-Box and Inverse S-Box

S-Box															
0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Inverse S-Box															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
5	E	F	8	C	1	2	D	B	4	6	3	0	7	9	A

3.3 Inverse Permutation Layer

The main aim of the permutation layer is to create confusion in data which directly leads to use bit permutation method to achieve the goal. Bit i of STATE is moved to position P(i).

Table 3.2: P-Layer

P-Layer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59

48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 3.3: Inverse P-Layer

P Inverse-Layer															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63

For decryption, a reverse process to remove the effect of P Layer is required.The bit permutation in PRESENT is given by the following table.Both Encryption and Decryption details are provided in the table 3.2.

Chapter 4

Key Scheduling

For Encryption: The key schedule of PRESENT-80 consists of a 61-bit left rotation, an S-box and a XOR with a round counter. Note that PRESENT uses the same S-box for the data path and the key schedule, which allows to share resources.

The user-supplied key is stored in a key register and its 64 most significant (left-most) bits serve as the round key. The key register is rotated by 61 bit positions to the left, the left-most four bits are passed through the PRESENT S-box, and the round counter value i is exclusively-ored with bits k19 k18 k17 k16 k15 of K with the Least Significant Bit [LSB] of the round counter on the right.

The key schedule of PRESENT-128 is slightly different from the one of PRESENT-80.

For Decryption: The Key schedule of PRESENT-80 consists of 19-bit left rotation, an inverse S-box and XOR with round counter of 31 - counter and an inverse permutation layer. The decryption unit of PRESENT is very similar to encryption unit.

The first round of decryption requires that the last round key of encryption routine. For optimal performance we assume that this last round key is pre-computed and available at the beginning of decryption routine.

The user supplied key is stored in one register K and represented k79 k78 ... k0. At initial round, ie, Ki which is first round key is obtained by extracting 64 leftmost bits from register. Thus at round 1 we have Ki= k63 k62 ...k0 = k79 k78...k16

After extracting the round key Ki, the key register K= k79 k78..k0 is updated in the following manner;

1. [k79 k78...k1 k0] = [k18 k17...k20 k19]

2. [k79 k78 k77 k76] = S[k79 k78 k77 k76]

3. [k19 k18 k17 k16 k15] = [k19 k18 k17 k16 k15] xor round counter.

Thus the key register is rotated by 19 bit positions to the left, the left-most four bits are passed through the PRESENT inverse S-box, and the 31 - round counter value i is exclusively-ored with bits k19 k 18 k17 k16 k15 of K with the LSB of round counter on the right

Chapter 5

Implementation

5.1 Implementation of PRESENT-80 DECRYPTION in Quartus

The main aim of PRESENT block cipher decryption is to see if the block cipher is Side channel attack resistant.To fulfil the goal design, PRESENT block cipher has been made in Asynchronous Method of design.

PRESENT has two entities PRESENT 80 and PRESENT 128 but here implementation of PRESENT 80 is completed.

To design and simulate cipher we have used Altera Quartus II web version 15.0 multiplatform design environment. Quartus II analysis and synthesis together delivers superior placement and routing, resulting in compilation time advantage. Also, Quartus has big community support. The entire cipher logic implemented as asynchronous design means each round is been design and implemented using basic logic gates.

At the beginning of the round both the input of algorithm, ciphertext and user supplied key bits are read from the respective registers and the operation begins. Total number of I/O pins used in implementation of Only PRESENT Decryption 80 is 208 and total number of logical units used is 4156.

5.1.1 Round One

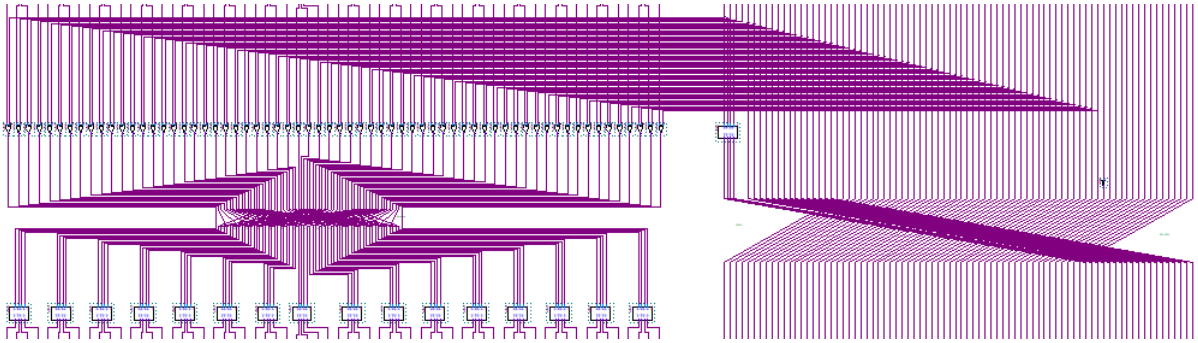

Figure 5.1: Round One

5.1.2 Permutation Inverse Layer

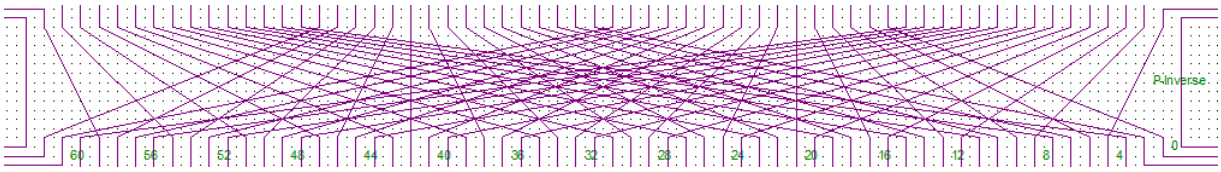

Figure 5.2: Inverse Permutation Layer

5.1.3 Key Scheduling 19 shift

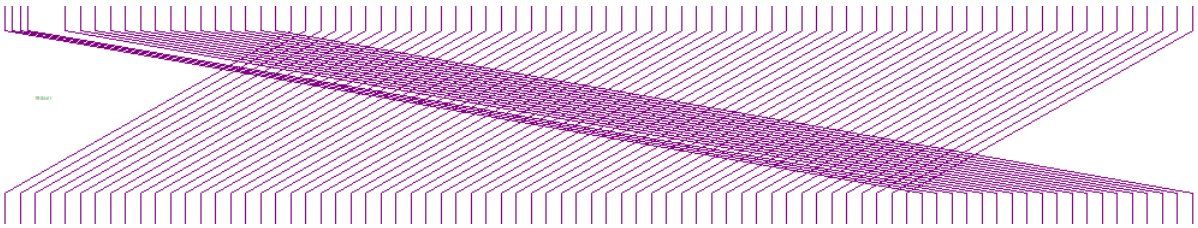

Figure 5.3: Key Scheduling 19 shift

5.1.4 S-Box Inverse

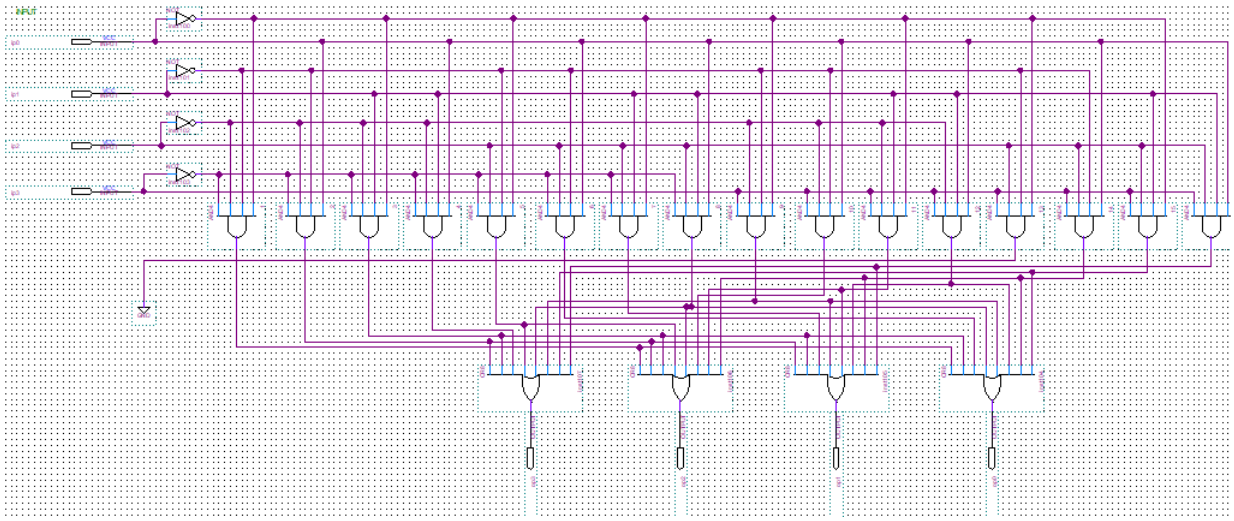

Figure 5.4: S Box Inverse

5.1.5 Last Round

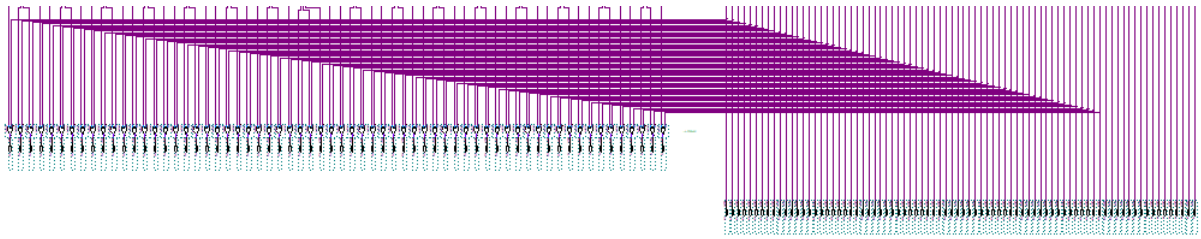

Figure 5.5: Last Round

5.1.6 Round Key

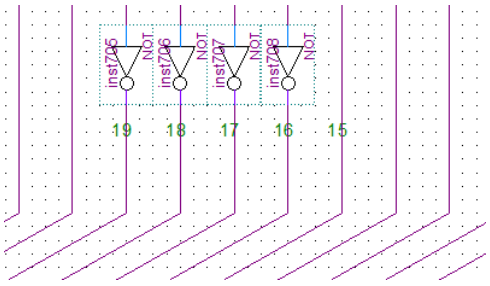

Figure 5.6: Round Key

5.1.7 Adding Plain Text and Key in Software

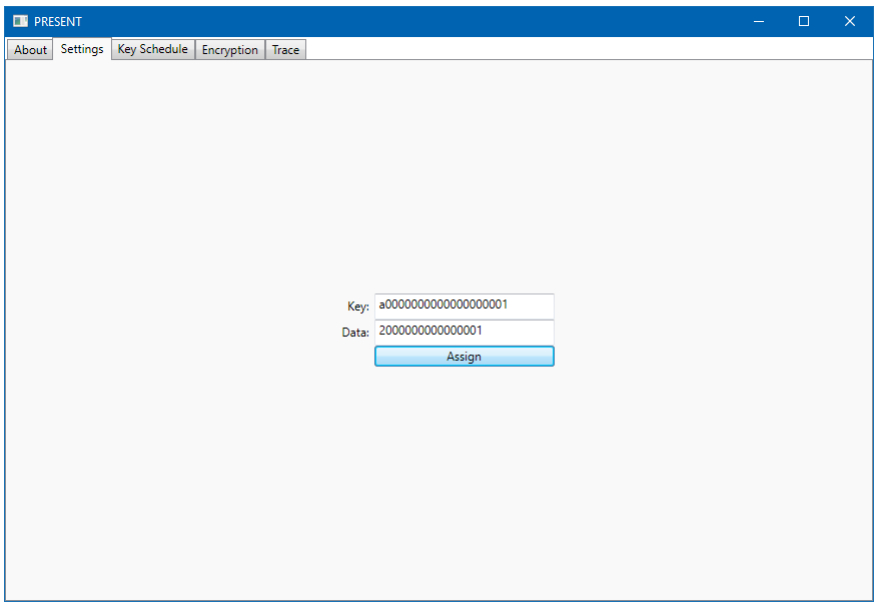

Figure 5.7: Adding Plain Text and Key in Software and Assigning

5.1.8 Getting Cipher Text and Last Round Key

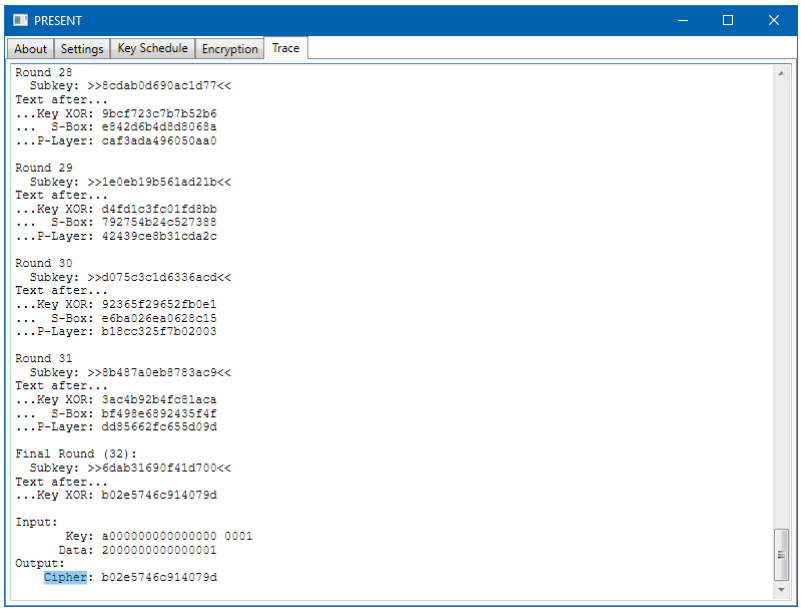

Figure 5.8: Cipher Text and Last Round Key (Input to Decryption)

5.1.9 Output of Decryption (Key)

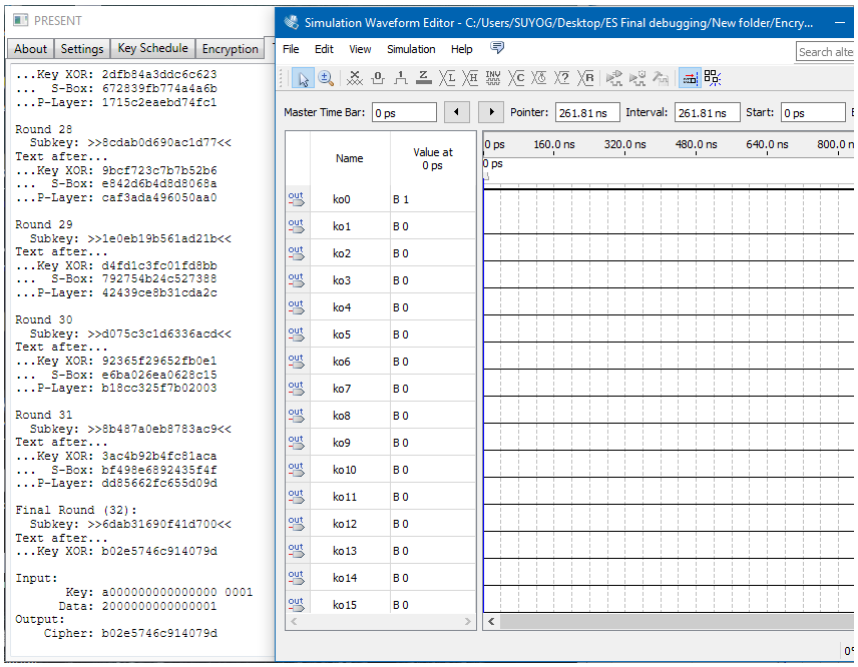

Figure 5.9: Getting Back Original Key (Which was Input key of Encryption)

5.1.10 Output of Decryption (Key)

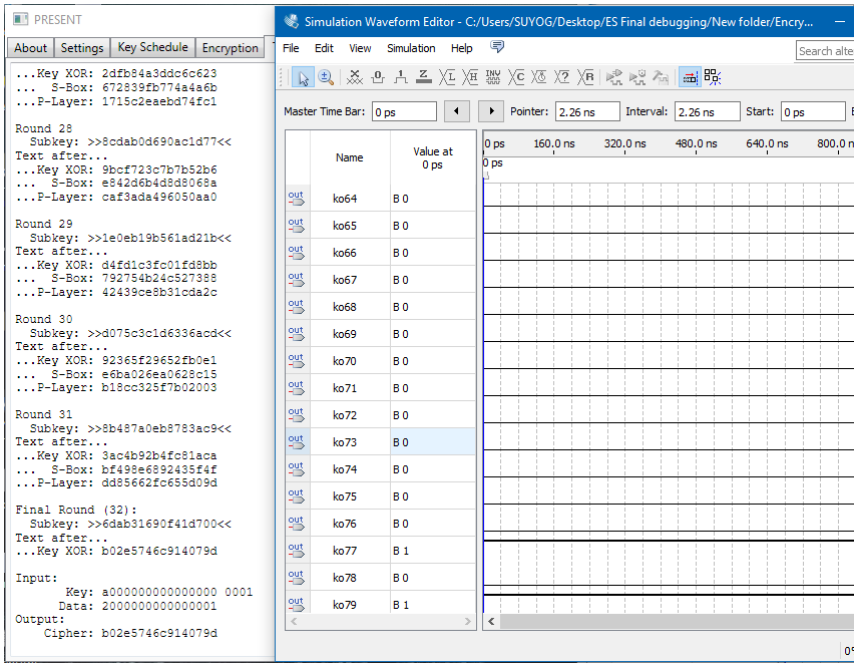

Figure 5.10: Getting Back Original Key (Which was Input key of Encryption)

5.1.11 Output of Decryption (Plain Text)

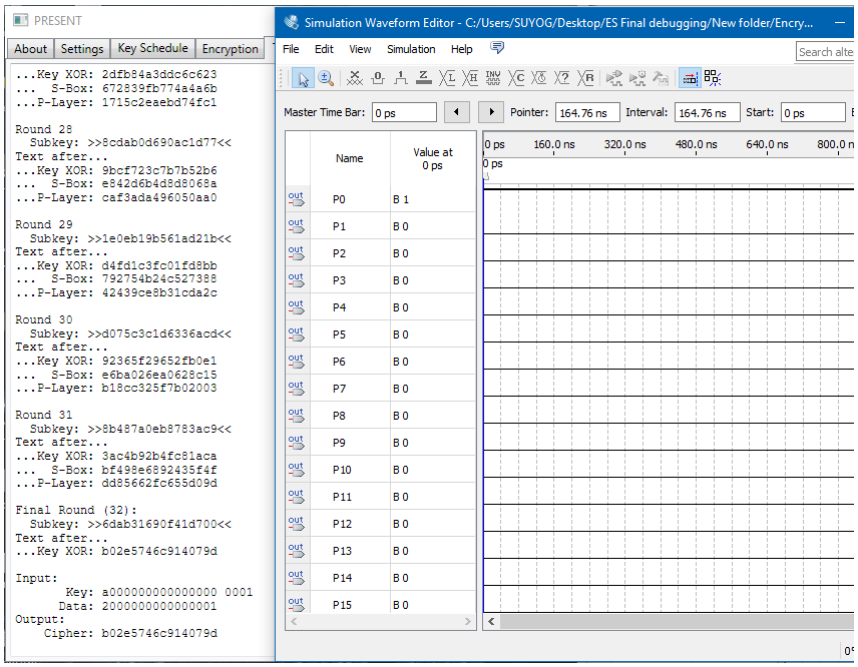

Figure 5.11: Getting Back Plain Text (Which was Input of Encryption)

5.1.12 Output of Decryption (Plain Text)

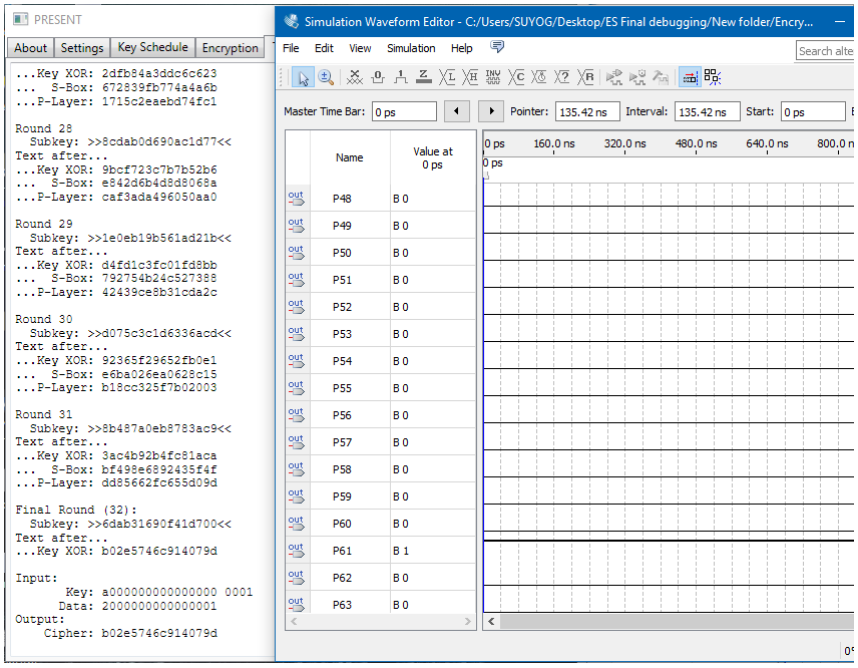

Figure 5.12: Getting Back Plain Text (Which was Input of Encryption)

Chapter 6

Observations

The implementation consist of Present Encryption Decryption consists of 8066 logic elements and 288 I/O pins. For Decryption we need last Round key of Encryption and cipher text which we get after Encryption.
We need to subtract last round key of Encryption with counter for 31 times for getting back Plain text.
Sbox Inverse and P Inverse has to be calculated from Sbox and P layer of Encryption. By flipping data of sbox and P layer and arranging in proper order we get Sbox inverse and P inverse layer. In Encryption we have done 61 times left shift so in decryption we need to do 19 times left shift in Round One.

Chapter 7

Conclusion

Present is an Ultra Light block cipher, therefore was mostly used in many small hardware applications such as RFID's. Present has been designed to be secured cryptographically by mixing Round key in implementation and to avoid any channel attacks. It has 80 bit key and 128 bit key for creating more complexity in hardware.
During implementation, some difficulties that were encountered:
1.We were required to design the entire implementation of encryption in order to design and verify Decryption.
2. Implementation of 32 round keys of encryption being a tedious process.
3. Design of inverse S-box for decryption.
4. Limited information available for reference regarding the inverse s-box implementation.
4. We were able to check the decryption output with reference to the encryption output.
5. No tool or simulator available to readily check the decryption output.
6. When design file is converted to symbol block, pins were not ordered as per design file and had to be re-arranged.
7. When compared to encryption where the same symbol blocks can be used multiple times as for implementation, in decryption the round-keys need to be implemented at the symbol block stage and not prior or after it. Hence the symbol blocks cannot be replicated and used for multiple rounds.
For decryption, we have given output of encryption that is Cipher text as input to decryption, and the last round key of encryption to first round of decryption to verify the decryption implementation. When running this implementation, the output of decryption is verified against the Plain-text and first key of the encryption with matching results. The output was compared with Encryption simulator - "PRESENT Animation.exe" [Horst Gortz Institute for IT Security, Ruhr University Bochum, Germany] and cross-verified with each stage of encryption.

Bibliography

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.

[2] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[3] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, "Design space exploration of present implementations for fpgas," in *2009 5th Southern Conference on Programmable Logic (SPL)*, April 2009, pp. 141–145.

[4] T. Korte, "www.lightweightcrypto.org/present," in *Horst Gortz Institute for IT Security*, Ruhr University Bochum Germany.

[5] X. Zhao, T. Wang, and S. Guo, "Improved side channel cube attacks on present."