Wireless Pers Commun (2018) 100:1427–1458
https://doi.org/10.1007/s11277-018-5647-z

Novel Reversible Design of Advanced Encryption Standard Cryptographic Algorithm for Wireless Sensor Networks

P. Saravanan[1] · **P. Kalpana**[1]

Published online: 23 March 2018

Abstract The quantum of power consumption in wireless sensor nodes plays a vital role in power management since more number of functional elements are integrated in a smaller space and operated at very high frequencies. In addition, the variations in the power consumption pave the way for power analysis attacks in which the attacker gains control of the secret parameters involved in the cryptographic implementation embedded in the wireless sensor nodes. Hence, a strong countermeasure is required to provide adequate security in these systems. Traditional digital logic gates are used to build the circuits in wireless sensor nodes and the primary reason for its power consumption is the absence of reversibility property in those gates. These irreversible logic gates consume power as heat due to the loss of per bit information. In order to minimize the power consumption and in turn to circumvent the issues related to power analysis attacks, reversible logic gates can be used in wireless sensor nodes. This shifts the focus from power-hungry irreversible gates to potentially powerful circuits based on controllable quantum systems. Reversible logic gates theoretically consume zero power and have accurate quantum circuit model for practical realization such as quantum computers and implementations based on quantum dot cellular automata. One of the key components in wireless sensor nodes is the cryptographic algorithm implementation which is used to secure the information collected by the sensor nodes. In this work, a novel reversible gate design of 128-bit Advanced Encryption Standard (AES) cryptographic algorithm is presented. The complete structure of AES algorithm is designed by using combinational logic circuits and further they are mapped to reversible logic circuits. The proposed architectures make use of Toffoli family of reversible gates. The performance metrics such as gate count and quantum cost of the proposed designs are rigorously analyzed with respect to the existing designs and are properly tabulated. Our proposed reversible design of AES algorithm shows considerable improvements in the performance metrics when compared to existing designs.

✉ P. Saravanan
dpsaravanan@yahoo.com

[1] Department of Electronics and Communication Engineering, PSG College of Technology, Peelamedu, Coimbatore, Tamil Nadu 641 004, India

Keywords Reversible logic · Security · Cryptography · AES algorithm · Toffoli gates · Low power · Wireless sensor networks · Power analysis attacks

1 Introduction

The reversible logic has demonstrated promising results in emerging applications of computing paradigm such as quantum computation and nanotechnology [1, 2]. Reversible logic was first related to power when Landauer stated that information loss due to functional irreversibility leads to power dissipation [3]. This principle is further supported by Bennett that zero power dissipation can be achieved only when the circuit contains reversible gates [4]. A cryptographic algorithm is an essential part in building secure systems to protect information against attacks. A well-known cryptographic algorithm is the Data Encryption Standard [5], which has been widely adopted in many security products. However, serious considerations arise for long-term security because of the relatively short key word length of only 56 bits and due to the highly successful cryptanalysis attacks.

In November 2001, the National Institute of Standards and Technology (NIST) of the United States chose the Rijndael algorithm as the suitable Advanced Encryption Standard (AES) [6] to replace the DES algorithm. Since then, many hardware implementations have been proposed in literature. Some of the works target field programmable gate arrays (FPGA) [7] as the implementation platform and some target application-specific integrated circuits (ASIC) [8]. But both FPGA and ASIC implementations consume power from its source which leads to information leakage through side channel attacks.

Quantum computing is a new paradigm for implementing cryptographic algorithms using reversible gates [9]. The primary reason for this is the increasing demands for low power and increased security for the computing devices. As our computing demands become more complex, the power requirements tend to increase. This further leads to a prominent side channel attack namely power analysis attacks on cryptographic systems, notably categorized as simple power analysis (SPA), differential power analysis (DPA) and high order power analysis (HO-PA) attacks. Cryptographic systems implemented with reversible gates ideally consume zero power and hence thwarts all side channel attacks related to power analysis. Energy efficient implementations of reversible building blocks to counterattack power analysis attacks were proposed in [10]. By using proper charge sharing mechanism, resistance to power analysis attacks has been achieved in their proposed implementations.

In Ref. [11], an attempt has been made to apply reversible logic to develop secure cryptosystems against power analysis attacks. A prototype of reversible Arithmetic and Logic Unit (ALU) for crypto-processor was presented in that work. Reversible Montgomery multipliers were proposed to resist power analysis attacks in crypto hardware [12]. The various functional blocks in AES have been synthesized using Toffoli family of reversible gates in [13]. The reversible gate design of AES SubBytes and Inverse SubBytes transformations were proposed in [14]. The reversible gate design of finite field architectures for Elliptic Curve Cryptography was proposed in [15].

In the current work, a well optimized reversible gate design of 128-bit AES cryptographic algorithm is presented. First, the key transformations in the algorithm are synthesized by using conventional logic gates. Then the transformations are mapped to reversible logic and are synthesized by using reversible logic gates. The reversible logic synthesis is optimized by reusing the existing reversible gates wherever applicable so that the performance metrics are improved.

The paper is organized as follows. The background of reversible logic gates and their quantum cost are given in Sect. 2. Section 3 highlights the transformation steps in AES cryptographic algorithm. The proposed reversible building blocks of AES algorithm are explained in Sect. 4. The performance analysis of the proposed reversible design with the existing designs is given in Sect. 5. Section 6 concludes the paper with necessary references.

2 Background on Reversible Logic Gates

Reversible logic marks a promising new direction where all operations are performed in an invertible manner. That is, in contrast to traditional logic, all computations can be reverted (i.e. the inputs can be obtained from the outputs and vice versa). In addition, reversible logic gates are bijective transformations on the inputs, i.e. number of input and outputs are equal and every distinct input gives a distinct output. Reversible gates do not allow any fanout or feedback. The commonly used reversible gates are NOT, controlled-NOT alias CNOT (also called as Feynman gate), controlled–controlled-NOT alias CCNOT (also called as Toffoli gate), SWAP and Fredkin gates as shown in Fig. 1. Among these, NOT, CNOT and Toffoli gates are basically n-bit controlled Toffoli gate with control lines n = 0, 1 and 2 respectively. The behavior of few reversible gates is defined as follows:

NOT: $a' = 1 \oplus a$
CNOT: $a' = a$, $b' = a \oplus b$
TOFFOLI: $a' = a$, $b' = b$, $c' = c \oplus ab$
SWAP: $b' = a$, $a' = b$
FREDKIN: $a' = a$, $b' = a'b + ac$, $c' = ab + a'c$.

The reversible logic synthesis can be done either with Toffoli gate family or Fredkin gate family since both are universal gates. In our proposed reversible design, Toffoli family of reversible gates has been used for reversible logic synthesis. The quantum cost of Toffoli gate with 0, 1 and 2 control lines is 1, 1, and 5 respectively. The performance metrics considered for reversible gate design are number of ancilla inputs, number of garbage outputs, number of reversible gates used, its quantum cost and delay in terms of number of stages. Ancilla inputs (Constants with either 0 or 1) and Garbage outputs are information that is not needed for the actual computation. They are required since the

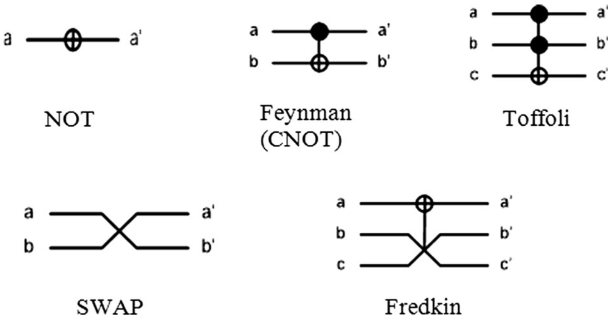


Fig. 1 Reversible gates

reversibility necessitates an equal number of outputs and inputs. Quantum cost denotes the effort needed to transform a reversible circuit to a quantum circuit [16].

3 Background on AES Algorithm

The AES algorithm (FIPS 2001) is a symmetric block cipher that processes data blocks of 128 bits using a cipher key of length 128, 192, or 256 bits. Each data block consists of a 4×4 array of bytes called the state S, on which the basic operations of the AES algorithm are performed. In the encryption process, after an initial round key addition, a round function consisting of four different transformations—SubBytes, ShiftRows, MixColumns and AddRoundKey—is applied to the data block in the encryption process.

The SubBytes transformation is a nonlinear byte substitution that operates independently on each byte of the state S using a substitution table (S-Box). The ShiftRows operation is a circular shifting on the rows of the state with different numbers of bytes (offsets). The MixColumns transformation mixes the bytes in each column of the state by the multiplication with a fixed polynomial modulo $x^4 + 1$. AddRoundKey is an XOR operation that adds a round key to the state S in each iteration, where the round keys are generated during the key expansion phase. The round function is performed iteratively 10, 12, or 14 times (Nr), depending on the key length of 128, 192 or 256-bits respectively.

The MixColumns transformation is not applied to the final round. The complete AES encryption steps are outlined in Fig. 2. The AES decryption steps can be performed by two ways such as inverse cipher and equivalent inverse cipher. In the Inverse Cipher process, shown in Fig. 3, the sequence of the transformations differs from that of the encryption, while the sequence of the key schedules for encryption and decryption remains the same. In the equivalent inverse cipher process, shown in Fig. 4, the sequence of transformations is same as the encryption. This is accomplished with a change in the key schedule.

4 Proposed Reversible Building Blocks of AES Transformations

In this research, the major transformations of AES algorithm such as SubBytes [14], MixColumns, ShiftRows, AddRoundKey and Key scheduler are deduced using conventional logic gates. Then, conventional logic gates are mapped to reversible logic gates and are reused wherever possible in order to optimize the performance metrics.

The proposed reversible gate design of AES algorithm utilizes Toffoli family of reversible gates for their logic synthesis and is optimized in terms of reduced number of ancilla inputs, garbage outputs, gate count, quantum cost and delay. The functional verification of the proposed reversible gate designs is carried out by writing Verilog code for each reversible gate and integrating them. Xilinx ISE 13.2 is used for simulation purpose.

4.1 SubBytes and InvSubBytes Transformations

In the encryption module, the SubBytes transformation is a non-linear transformation, which computes the multiplicative inverse of each byte of the state S in $GF(2^8)$ with irreducible polynomial $P(x) = x^8 + x^4 + x^3 + x + 1$ followed by an affine transformation. The transformation in the decryption module performs the inverse of the corresponding transformation in the encryption module.

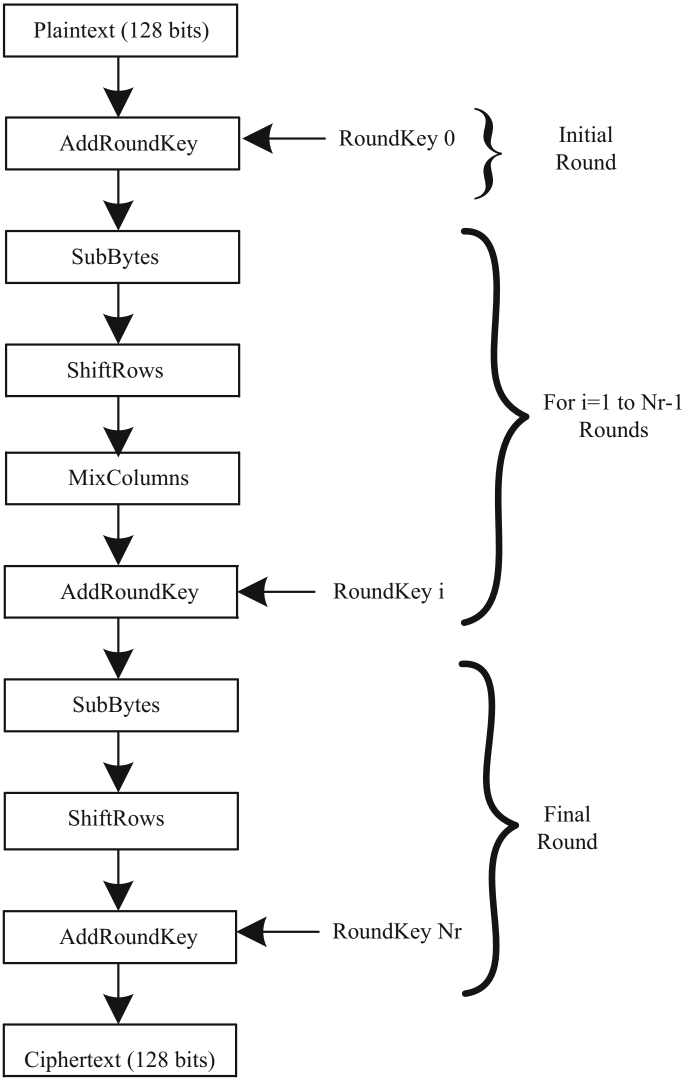


Fig. 2 AES Encryption process

4.1.1 Design Approaches

The SubBytes and InvSubBytes transformations can be implemented by two different approaches. They can be either constructed as a single circuit whose input–output relation is directly equivalent to the SubBytes transformation known as Look-up Table (LUT) approach or constructed as a multiplicative inversion circuit and an affine transformation circuit independently. Then, these two circuits are cascaded to design the SubBytes transformation known as composite field arithmetic approach.

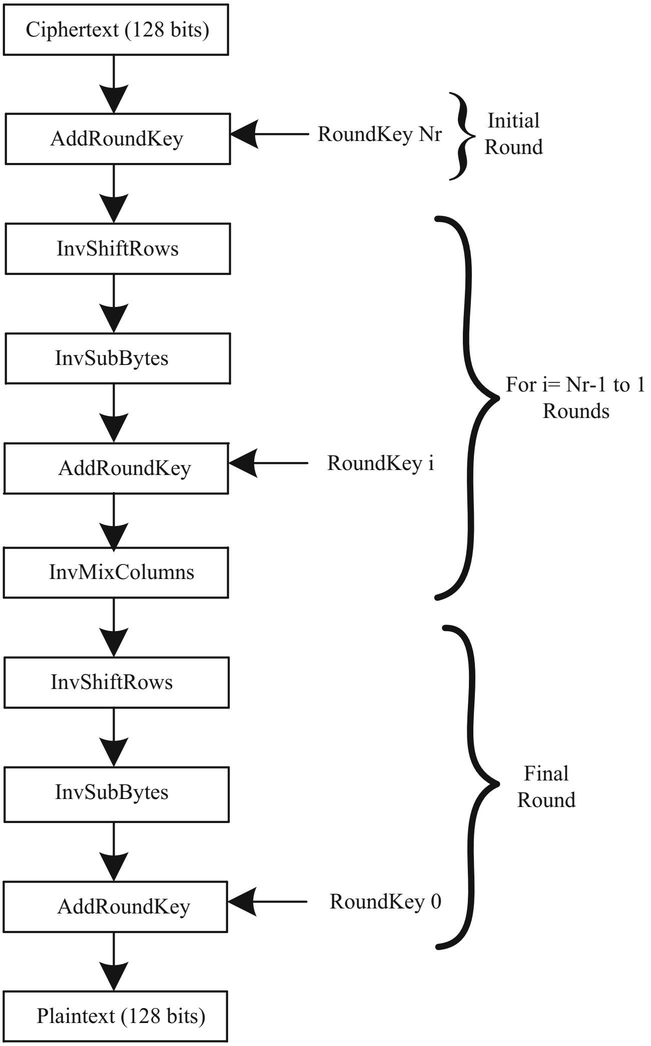


Fig. 3 AES decryption using inverse cipher process

4.1.2 Composite Field Arithmetic Approach

Composite fields are frequently used in the design of Galois Field arithmetic where arithmetic operations rely on table lookups. Composite fields also known as sub-field arithmetic can be used to reduce lookup-related costs. Hence, another approach to design SubBytes and InvSubBytes transformations is the use of composite field arithmetic in the design of multiplicative inversion module. The idea of applying composite field arithmetic to AES algorithm is explored in detail in [17–19]. Applying composite field arithmetic, the elements of large-order fields are mapped to those of small-order fields in which the field operations can be carried out in a simpler way with reduced hardware cost. The functional blocks to design multiplicative inverse module in GF(2^8) is shown in Fig. 5.

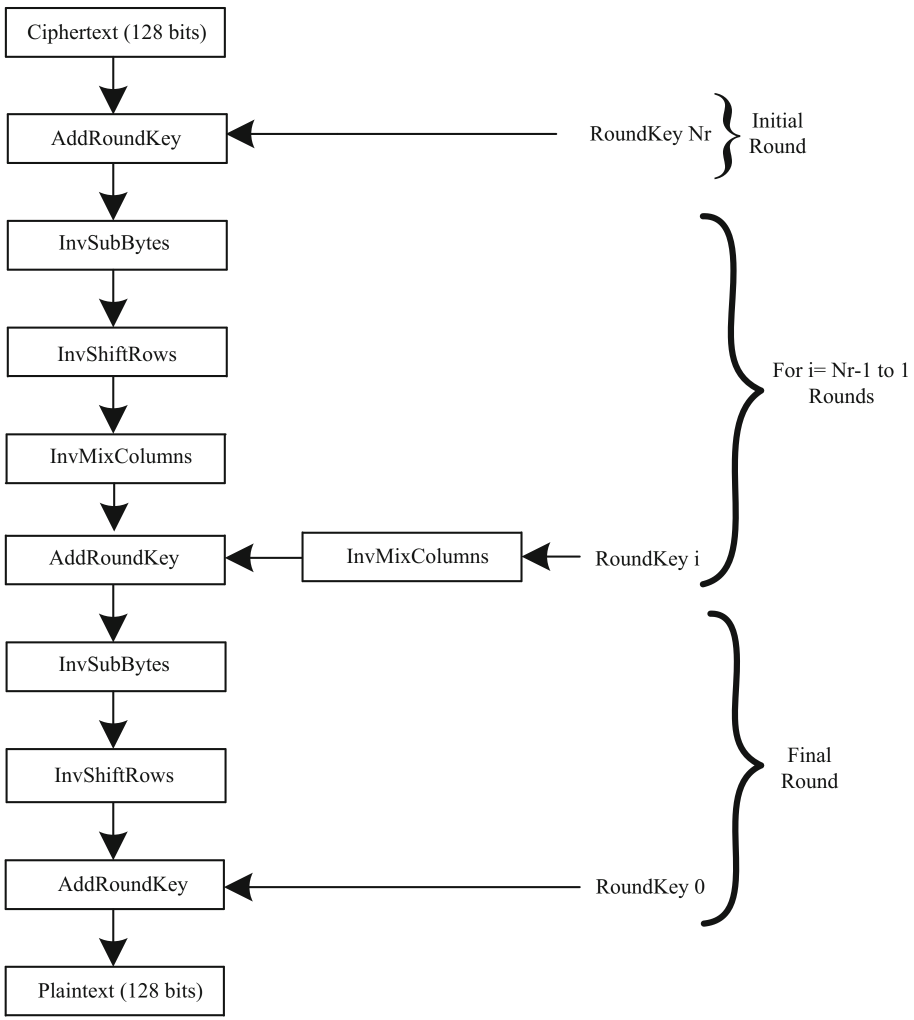


Fig. 4 AES decryption using equivalent inverse cipher process

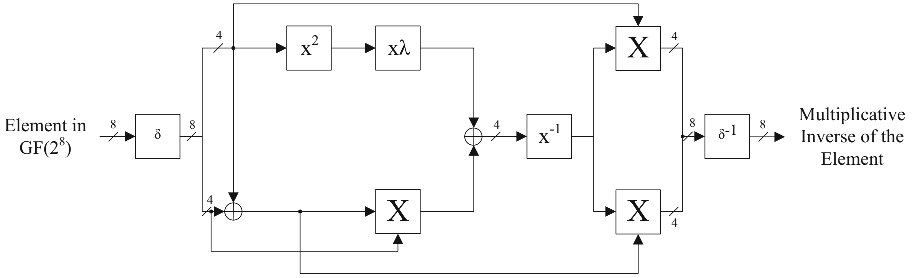


Fig. 5 Basic building blocks to design multiplicative inverse module in GF(2^8)

The details of each block in the GF(2^8) multiplicative inversion module are illustrated below.

δ → Isomorphic mapping from GF(2^8) to composite fields
x^2 → Squarer in GF(2^4)
xλ → Multiplication with constant λ in GF(2^4)
⊕ → Addition operation in GF(2^4)
x^{-1} → Multiplicative inversion in GF(2^4)
X → Multiplication operation in GF(2^4)
δ^{-1} → Inverse isomorphic mapping to GF(2^8).

The SubBytes transformation can be performed by taking the multiplicative inverse in GF(2^8) followed by affine transformation and the InvSubBytes transformation can be performed by inverse affine transformation followed by multiplicative inverse in GF(2^8). The sequence of steps to carry out both transformations is shown below.

SubBytes transformation: multiplicative inversion in GF(2^8) followed by Affine transformation.

InvSubBytes transformation: inverse affine transformation followed by Multiplicative inversion in GF(2^8).

4.1.3 Proposed Reversible Multiplicative Inverse Module in GF(2^8)

Since reversible gate designs are functionally reversible, it is sufficient to design either forward isomorphic mapping or inverse isomorphic mapping. The number of XOR operations required in the forward isomorphic mapping is 24 whereas inverse isomorphic mapping takes only 23 XOR operations. Hence, inverse isomorphic mapping function has been designed by using reversible gates in this research and the same design can be used for forward isomorphic mapping also.

The corresponding reversible gate design requires 23 CNOT gates which results in a quantum cost of 23. This calculation is based on the assumption that each XOR operation requires one CNOT gate when the reversible logic synthesis is performed by conventional design using one-to-one mapping from logic operations to equivalent reversible gates. However, in the proposed design, the gate count and quantum cost is optimized by properly reusing the existing reversible gates. The proposed reversible inverse isomorphic mapping block takes only 15 CNOT gates and has a quantum cost of 15 which gives 35% savings in both gate count and quantum cost compared to the conventional design. The proposed design has zero ancilla inputs, zero garbage outputs and has delay of 13 as shown in Table 1. The reversible gate design of forward/inverse isomorphic mapping block is shown in Fig. 6.

The reversible squarer and multiplication with constant λ block in GF(2^4) is shown in Fig. 7. The squarer block takes 4 CNOT gates and multiplication with constant λ block takes 4 CNOT gates when they are synthesized separately by using reversible gates. But the proposed reversible gate design of the merged squarer and multiplication with constant λ block takes only 4 CNOT gates and has a quantum cost of 4 which gives 50% savings in gate count and quantum cost compared to the individual designs. Also, it takes zero ancilla input, zero garbage output and has a delay of 3 as shown in Table 1.

The addition operation in GF(2^4) can be performed by simple logical XOR operation. The reversible gate design of the GF(2^4) adder block is shown in Fig. 8, which takes 4

Table 1 Performance analysis of proposed reversible building blocks of multiplicative inverse module in GF(2^8)

Name of the block	No. of ancilla inputs	No. of garbage outputs	No. of reversible gates	Quantum cost	Delay
IsoMap/InvIsoMap	0	0	CNOT—15	15	13
Squarer and multiplication by constant λ	0	0	CNOT—4	4	3
Adder (XOR block)	0	4	CNOT—4	4	1
Multiplication in GF(2^4)	13	17	CNOT—25 CCNOT—9	70	18
Multiplicative inverse in GF(2^4)	8	8	CNOT—14 CCNOT—8	54	19

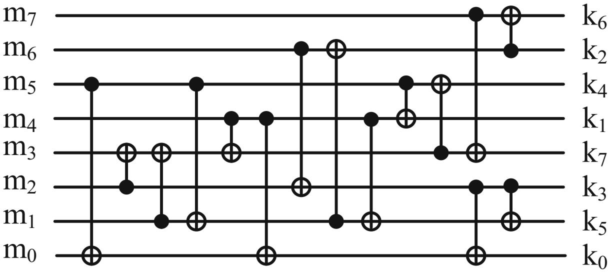


Fig. 6 Reversible IsoMap/InvIsoMap block

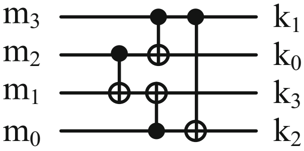


Fig. 7 Reversible squarer and multiplication with constant block

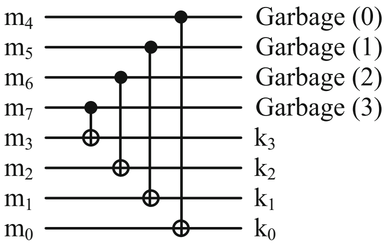


Fig. 8 Reversible adder in GF(2^4)

CNOT gates and has a quantum cost of 4. Also, it takes zero ancilla inputs, 4 garbage outputs and has a delay of 1 as shown in Table 1.

The reversible gate design of GF(2^4) multiplier block requires 9 AND operations and 46 XOR operations. The reversible logic synthesis by one-to-one mapping takes 9 CCNOT gates and 46 CNOT gates which results in a quantum cost of 91. The proposed design is optimized by reusing the reversible gates properly so that the reversible GF(2^4) multiplier block takes only 9 CCNOT gates and 25 CNOT gates and has a quantum cost of 70 which results in 38% savings in Gate count and 23% savings in quantum cost. The proposed design takes 13 ancilla inputs, 17 garbage outputs and has a delay of 18 as shown in Table 1. The reversible gate design of the multiplication in GF(2^4) is shown in Fig. 9.

The Multiplicative inverse in GF(2^4) can be calculated by three different approaches namely Square–Multiply approach, Multiple-Decomposition approach and Direct Mapping approach [18]. The computation of multiplicative inverse using Square and multiply approach is shown in Fig. 10. The corresponding reversible gate design can be obtained by using reversible squarer block and reversible GF(2^4) multiplier. The reversible gate design of multiplicative inverse in GF(2^4) using Square and Multiply approach takes 70 CNOT, 18 CCNOT gates and has a quantum cost of 160. Also, it takes 34 ancilla inputs, 42 garbage outputs and has a delay of 56 as shown in Table 2.

In multiple-decomposition approach, the multiplicative inverse in GF(2^4) can be computed by using composite field approach. In this case, the element in GF(2^4) is decomposed into a composite field of GF(2^2)^2. The computation of multiplicative inverse in composite field GF(2^2)^2 is shown in Fig. 11 [18]. The reversible gate design of multiplicative inverse in composite field GF(2^2)^2 is shown in Fig. 12. The reversible gate design takes 22 CNOT, 9 CCNOT gates, and has a quantum cost of 67. Also, it takes 14 ancilla inputs, 14 garbage outputs, and has a delay of 19 as shown in Table 2.

Let $m = \{m_3 m_2 m_1 m_0\}_2$ be an element in GF(2^4) and its inverse is given by $m^{-1} = \{m_3^{-1} m_2^{-1} m_1^{-1} m_0^{-1}\}_2$ which can be obtained by direct mapping approach as given in Eq. (1) [18].

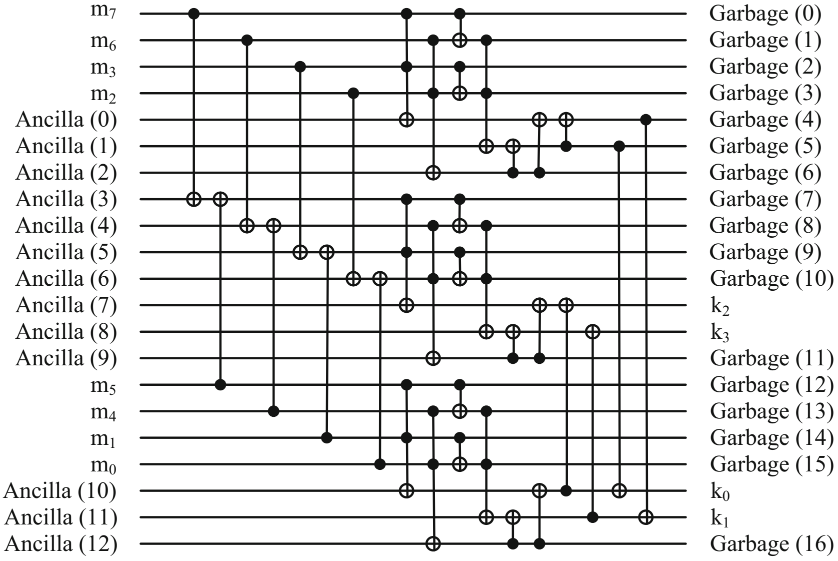


Fig. 9 Reversible multiplier in GF(2^4)

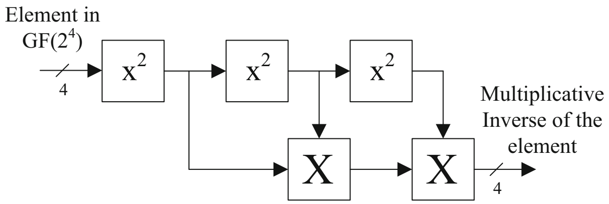


Fig. 10 Multiplicative inverse using square and multiply approach

Table 2 Performance comparison of reversible multiplicative inverse modules in GF(2^4)

Name of the approach	No. of ancilla inputs	No. of garbage outputs	No. of reversible gates	Quantum cost	Delay
Square–multiply	34	42	CNOT—70 CCNOT—18	160	56
Multiple decomposition	14	14	CNOT—22 CCNOT—9	67	19
Direct mapping	8	8	CNOT—14 CCNOT—8	54	19

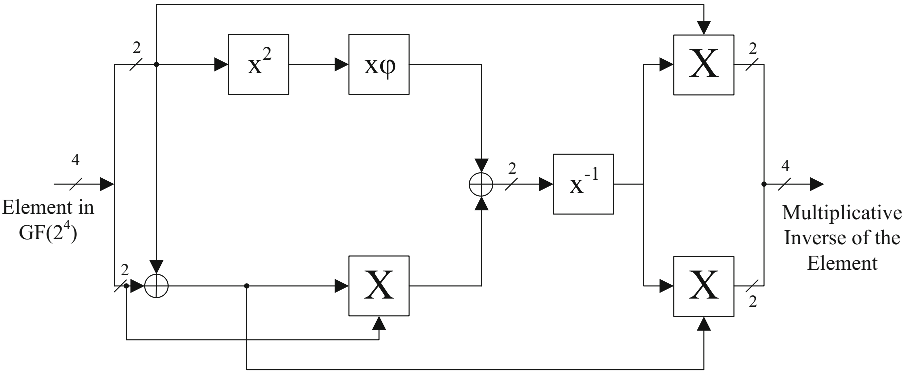


Fig. 11 Multiplicative inverse using multiple decomposition approach

$$
\begin{aligned}
m_3^{-1} &= m_3 \oplus m_3m_2m_1 \oplus m_3m_0 \oplus m_2\\
m_2^{-1} &= m_3m_2m_1 \oplus m_3m_2m_0 \oplus m_3m_0 \oplus m_2 \oplus m_2m_1\\
m_1^{-1} &= m_3 \oplus m_3m_2m_1 \oplus m_3m_1m_0 \oplus m_2 \oplus m_2m_0 \oplus m_1\\
m_0^{-1} &= m_3m_2m_1 \oplus m_3m_2m_0 \oplus m_3m_1 \oplus m_3m_1m_0 \oplus\\
&\quad m_3m_0 \oplus m_2 \oplus m_2m_1 \oplus m_2m_1m_0 \oplus m_1 \oplus m_0
\end{aligned} \tag{1}
$$

By analysing Eq. (1), it can be inferred that the direct mapping approach requires 25 AND operations and 21 XOR operations to calculate the multiplicative inverse. The reversible logic synthesis of Eq. (1) using one-to-one mapping takes 25 CCNOT gates, 21 CNOT gates and has a quantum cost of 146. In the proposed reversible gate design of GF(2^4) multiplicative inversion module, the reversible gates are properly reused so that it takes only 8 CCNOT and 14 CNOT gates with a quantum cost of 54 as shown in Fig. 13.

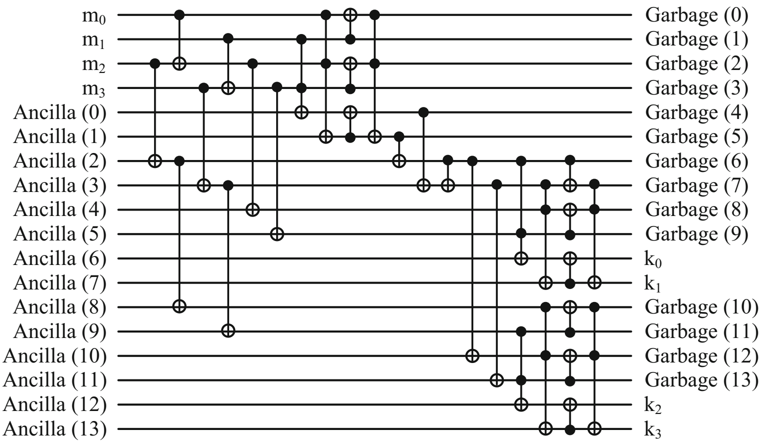


Fig. 12 Reversible gate design of multiplicative inverse module using multiple decomposition approach

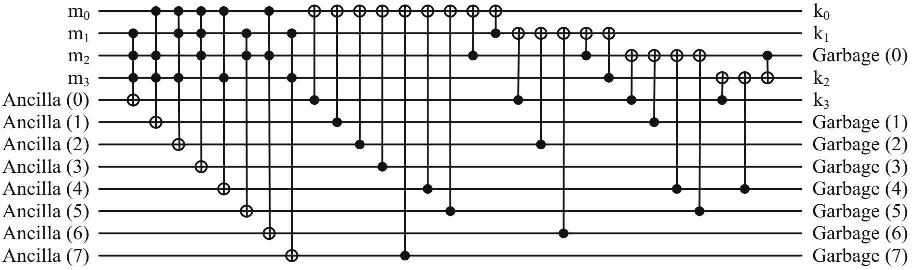


Fig. 13 Reversible gate design of multiplicative inverse module using direct mapping approach

The reusability of reversible gates results in 52% savings in gate count and 63% savings in quantum cost. The proposed design takes 8 ancilla inputs and 8 garbage outputs and has a delay of 19 as shown in Table 2.

From Table 2, it can be inferred that the multiplicative inverse in GF(2^4) can be efficiently computed with direct mapping approach, since it takes less number of reversible gates and the quantum cost involved is also less. This is because, the composite field approach will not give optimum results when the order of the field involved is small such as GF(2^4). Hence, in the proposed reversible SubBytes and InvSubBytes transformations, the multiplicative inverse in GF(2^4) is calculated by direct mapping approach.

The performance metrics of the reversible building blocks of GF(2^8) multiplicative inversion module are given in Table 1. The proposed reversible GF(2^8) multiplicative inversion module requires two reversible IsoMap/InvIsoMap block, two reversible GF(2^4) adder, one reversible Squarer and Multiplication by constant λ block, three reversible GF(2^4) multiplier, and one reversible GF(2^4) multiplicative inversion module as shown in Fig. 5 [18]. The performance metrics of the proposed reversible GF(2^8) multiplicative inversion module are tabulated in Table 3.

Table 3 Performance analysis of proposed reversible multiplicative inverse module in GF(2^8)

Name of the block	No. of ancilla inputs	No. of garbage outputs	No. of reversible gates	Quantum cost	Delay
IsoMap/InvIsoMap	0	0	CNOT—30	30	26
Squarer and multiplication by constant λ	0	0	CNOT—4	4	3
Adder (XOR block)	0	8	CNOT—8	8	2
Multiplication in GF(2^4)	39	51	CNOT—75 CCNOT—27	210	36
Multiplicative inverse in GF(2^4)	8	8	CNOT—14 CCNOT—8	54	19
Proposed reversible GF(2^8) multiplicative inversion module	47	67	CNOT—131 CCNOT—35	306	83

4.1.4 Proposed Reversible Affine Transformation Block

The reversible gate design of affine transformation is shown in Fig. 14. Since the design is functionally reversible, it is enough if either affine transformation or inverse affine transformation is considered. In this research, reversible gate design of affine transformation has been carried out which actually requires 4 NOT operations and 32 XOR operations. The reversible logic synthesis using one-to-one mapping approach requires 4 NOT and 32 CNOT gates with a quantum cost of 36. By properly reusing the existing reversible gates, the proposed reversible affine transformation block is optimized to 4 NOT and 21 CNOT gates with a quantum cost of 25. This optimization results in 31% savings in both gate count and quantum cost. The proposed design takes zero ancilla inputs, zero garbage outputs and has a delay of 21 as shown in Table 4.

4.1.5 Proposed Reversible SubBytes/InvSubBytes Transformation Blocks

The reversible SubBytes transformation can be obtained by cascading reversible GF(2^8) multiplicative inversion module and reversible affine transformation block as shown in Fig. 15 [18]. The proposed reversible SubBytes transformation takes 4 NOT gates, 152 CNOT gates, 35 CCNOT gates and has a quantum cost of 331. Also, it takes 47 ancilla

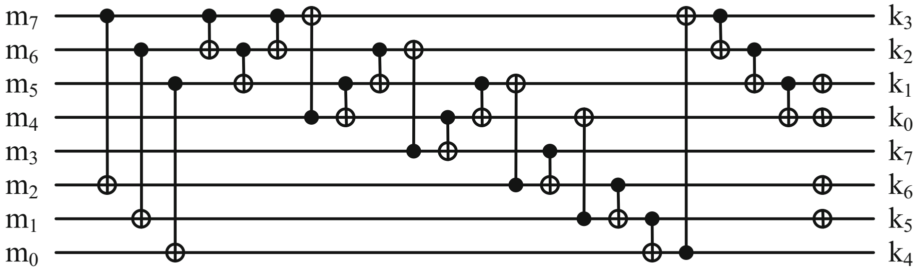


Fig. 14 Reversible affine transformation block

Table 4 Performance analysis of proposed reversible subbytes/invsubbytes transformation module

Name of the block	No. of ancilla inputs	No. of garbage outputs	No. of reversible gates	Quantum cost	Delay
Multiplicative inverse in GF(2^8)	47	67	CNOT—131 CCNOT—35	306	83
Affine transformation	0	0	NOT—4 CNOT—21	25	21
Proposed reversible SubBytes/ InvSubBytes transformation module	47	67	NOT—4 CNOT—152 CCNOT—35	331	104

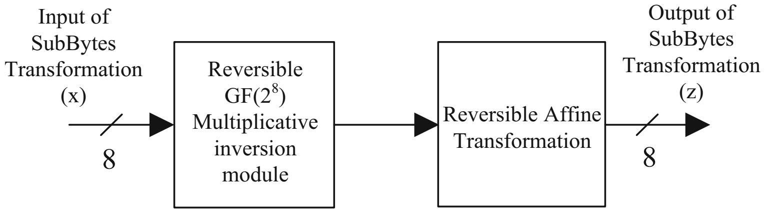

Input of SubBytes Transformation (x)
8
Reversible GF(2^8) Multiplicative inversion module
Reversible Affine Transformation
Output of SubBytes Transformation (z)
8

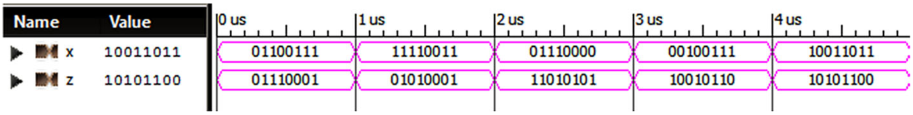
Fig. 15 Block diagram of the proposed reversible SubBytes transformation

inputs, 67 garbage outputs and has a delay of 104 as shown in Table 4. The simulation output of the proposed reversible SubBytes transformation is shown in Fig. 16.

The reversible InvSubBytes transformation can be obtained by cascading reversible affine transformation and reversible GF(2^8) multiplicative inversion module as shown in Fig. 17 [18]. The performance metrics of the proposed reversible InvSubBytes transformation is similar to reversible SubBytes transformation since the same building blocks are used for both transformations. The simulation output of the proposed reversible InvSubBytes transformation is shown in Fig. 18. The proposed reversible InvSubBytes transformation takes 4 NOT gates, 152 CNOT gates, 35 CCNOT gates and has a quantum cost of 331. Also, it takes 47 ancilla inputs, 67 garbage outputs and has a delay of 104. The performance metrics of the proposed reversible SubBytes/InvSubBytes transformation are given in Table 4.

4.1.6 Performance Analysis

Table 5 summarizes the performance improvement in the proposed reversible SubBytes/InvSubBytes transformation module compared to the conventional reversible designs. An important point to be noted here is that the conventional reversible designs are the ones



Name	Value
x	10011011
z	10101100

Fig. 16 Functional verification of the proposed reversible SubBytes transformation block

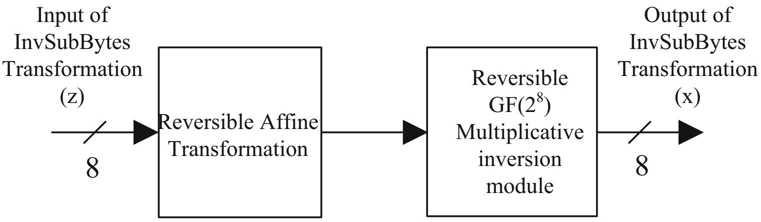
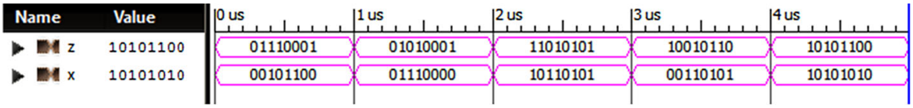


Fig. 17 Block diagram of the proposed reversible InvSubBytes transformation



Name	Value	0 us	1 us	2 us	3 us	4 us
z	10101100	01110001	01010001	11010101	10010110	10101100
x	10101010	00101100	01110000	10110101	00110101	10101010

Fig. 18 Functional verification of the proposed reversible InvSubBytes transformation block

Table 5 Performance comparison of reversible subbytes/invsubbytes transformation modules

Name of the block	Gate count		Quantum cost		% Reduction	
	Conventional design	Proposed design	Conventional design	Proposed design	Gate count	Quantum cost
IsoMap/InvIsoMap	46	30	46	30	35	35
Squarer and multiplication by constant λ	8	4	8	4	50	50
Adder (XOR block)	8	8	8	8	–	–
Multiplication in GF(2^4)	153	102	261	210	33	20
Multiplicative inverse in GF(2^4)	46	22	146	54	52	63
Affine transformation	36	25	36	25	31	31
Proposed reversible SubBytes/ InvSubBytes transformation	297	191	505	331	36	35

obtained by direct one-to-one mapping of logic operations to reversible gates. The two important performance metrics known as gate count and quantum cost are analysed in this research to show the improvement in the proposed reversible designs.

The proposed design of reversible SubBytes/InvSubBytes transformation module shows 36% reduction in gate count and 35% reduction in quantum cost compared to the conventional reversible designs. In addition, the proposed reversible gate design shows 35% reduction in gate count and 97% reduction in quantum cost compared to the existing reversible SubBytes and InvSubBytes transformation module [13] as shown in Table 6.

4.2 MixColumns and InvMixColumns Transformations

The MixColumns transformation operates on the state column-by-column, treating each column as a four-term polynomial (FIPS 2001). In this transformation, each byte of a column is replaced by a function of all the bytes in the same column. Here, each

Table 6 Performance improvement in the proposed reversible subbytes/invsubbytes transformation modules

Functional block	Gate count		Quantum cost		% Reduction	
	[13]	Proposed design	[13]	Proposed design	Gate count	Quantum cost
Reversible SubBytes/InvSubBytes transformation	294	191	11,602	331	35	97

transformation is based on the polynomial c(x) = {03}x^3 + {01}x^2 + {01}x + {02}. This polynomial is co-prime to $x^4 + 1$ and therefore invertible with d(x) = {0B}x^{3-} + {0D}x^2 + {09}x + {0E} as the inverse. So, the InvMixColumns transformation is constructed with the polynomial d(x) where d(x) = c^{-1}(x).

4.2.1 Proposed Reversible MixColumns Transformation

The structure of MixColumns transformation is shown in Fig. 19. The whole design of the architecture is divided into two parts namely block-1 and block-2. The combined block is the integration of block-1 and block-2. The reversible block-1 implements the following operations such as X_{bcd}, X_{acd}, X_{abd}, X_{abc} and X_{ab}, X_{bc}, X_{cd}, X_{da} in the MixColumns transformation of AES crypto core in a bit-by-bit basis using reversible gates. Here, the notation X_{bcd} indicates $b \oplus c \oplus d$ and X_{ab} indicates $a \oplus b$. It takes one bit from all the four elements of the column in the state array and gives eight valid outputs as shown in Fig. 20. The reversible gate design of block-1 is made with 12 CNOT gates and it takes 4 ancilla inputs and zero garbage outputs. Also, it has a quantum cost of 12 with 8 units delay as given in Table 7.

The block-2 performs XTime operation and gives the final output of MixColumns transformation. It takes sixteen valid inputs which come out from different block-1 and performs the GF(2^8) multiplication and gives eight valid outputs which are nothing but the

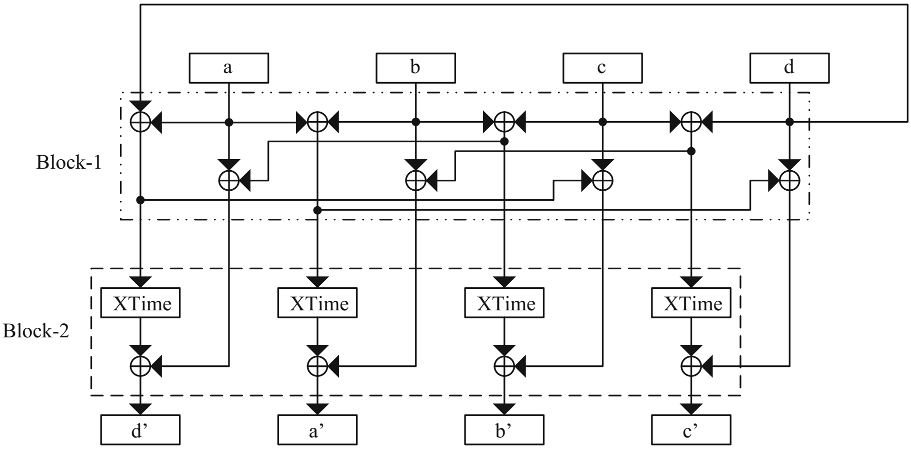


Fig. 19 Structure of MixColumns transformation

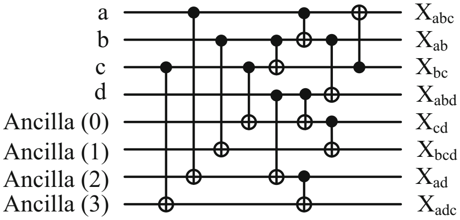


Fig. 20 Reversible gate design of block-1

Table 7 Performance analysis of reversible mixcolumns transformation block

Metrics	Block 1	Block 2	Proposed reversible block
Ancilla inputs	4	0	32
Garbage outputs	0	8	32
No. of reversible gates	12	11	140
Quantum cost	12	11	140
Delay	8	11	19

final output of the MixColumns transformation of AES crypto core. The reversible gate design of block-2 is carried out with 11 CNOT gates and it takes zero ancilla input and 8 garbage outputs as shown in Fig. 21. In addition, it has a quantum cost of 11 with 11 units of delay as given in Table 7.

The previously mentioned sub-blocks block-1 and block-2 are instantiated 8 times and 4 times respectively and combined together to get the final design of the MixColumns transformation in AES crypto core as shown in Fig. 22. The reversible gate design of the combined block is made of 140 CNOT gates and it takes 32 ancilla inputs and 32 garbage outputs. In addition, it has a quantum cost of 140 with 19 units of delay as given in Table 7.

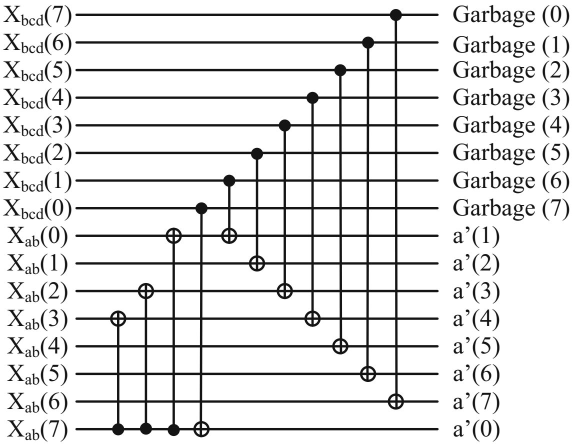


Fig. 21 Reversible gate design of block 2

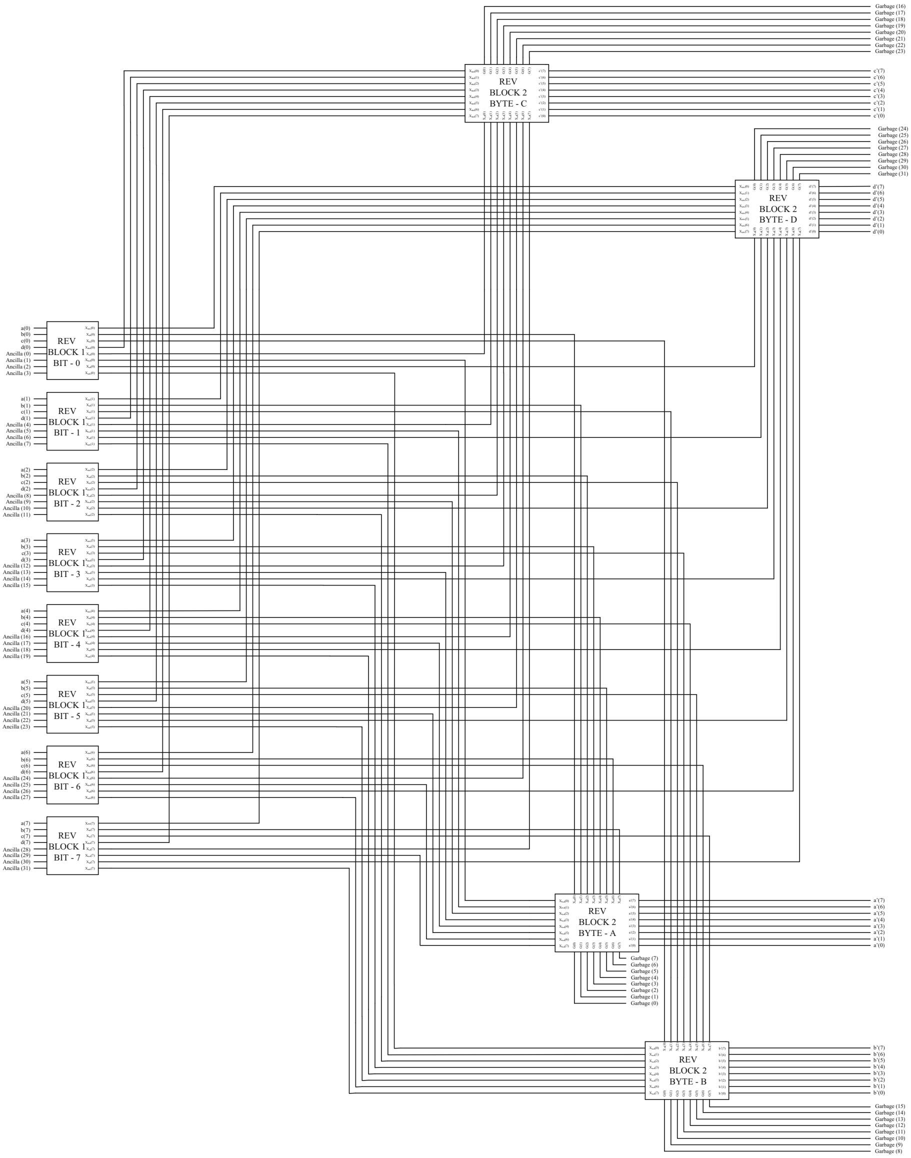


Fig. 22 Proposed reversible MixColumns transformation block

4.2.2 Design of Reversible InvMixColumns Transformation

The reversible gate design of InvMixColumns transformation is made by decomposing it into five blocks such as reversible MixColumns transformation block, reversible inverse block-1, reversible inverse block-2, reversible inverse block-3 and reversible X4Time block as shown in Fig. 23. The reversible inverse blocks 1, 2 and 3 consist of similar gate level structure. These blocks take two sets of 8-bit inputs and perform XOR operations

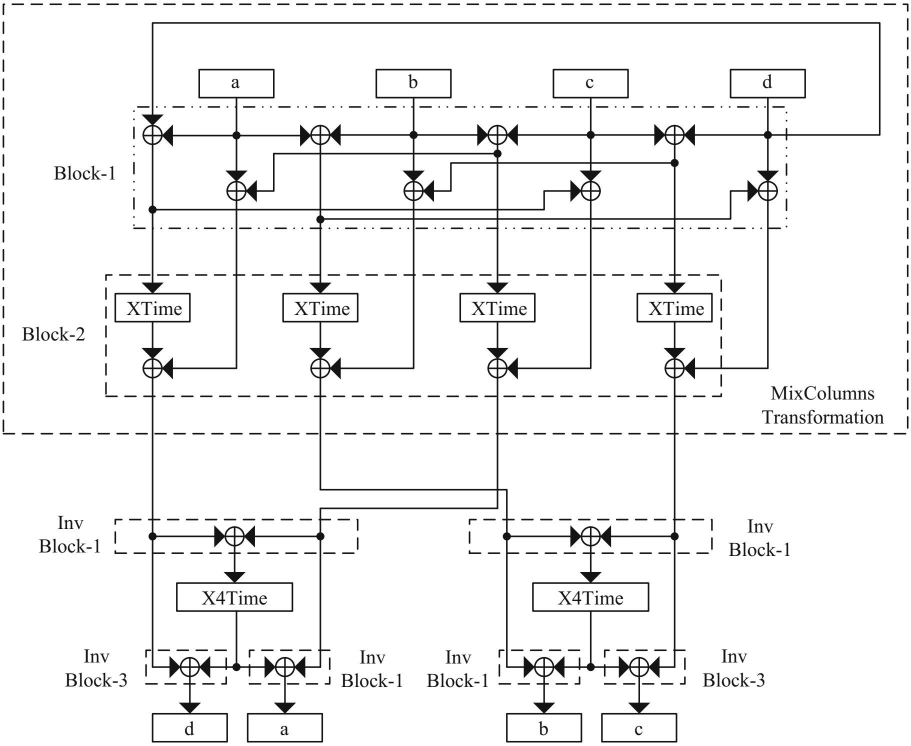


Fig. 23 Structure of InvMixColumns transformation

among them by using CNOT gates. The blocks produce 8-bit XORed outputs and the remaining 8-bit outputs are directly taken from one of the 8-bit inputs.

The MixColumns transformation in AES algorithm is synthesized by using reversible gates in Sect. 4.2.1. The same reversible design can also be used in reversible InvMixColumns transformation block in order to reduce the hardware cost. The reversible MixColumns transformation block requires 140 CNOT gates with a quantum cost of 140. In addition, it takes 32 ancilla inputs, 32 garbage outputs and has a delay of 19 units as shown in Fig. 22. The reversible inverse blocks 1, 2 and 3 perform bitwise XOR operations on their two sets of inputs using CNOT gates and produces XORed outputs as shown in Fig. 24.

Each reversible inverse block requires 8 CNOT gates with a quantum cost of 8 and has unit delay. The number of ancilla input and garbage output will be different for each reversible inverse blocks. Reversible inverse block-1 does not require any ancilla inputs and garbage outputs whereas reversible inverse block-2 requires 8 ancilla inputs and it has zero garbage output. Reversible inverse block-3 has 8 garbage outputs with zero ancilla input. The reversible X4Time block which computes the multiplication with constant term 04 can be designed by cascading two XTime blocks [20]. Instead, the X4Time block which saves one XOR operation can be designed directly as shown in Fig. 25.

The corresponding reversible gate design of X4Time block is shown in Fig. 26. It requires 5 CNOT gates with quantum cost of 5 and has a delay of 5 units. This reversible block has zero ancilla input and garbage output. The previously mentioned sub-blocks such

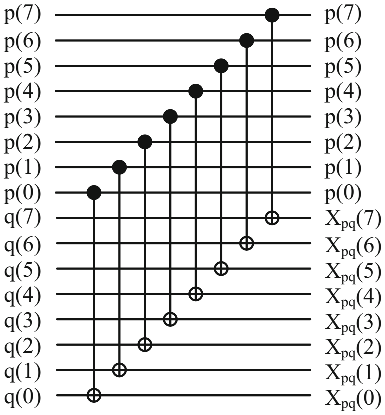


Fig. 24 Reversible gate design of inverse blocks 1, 2 and 3

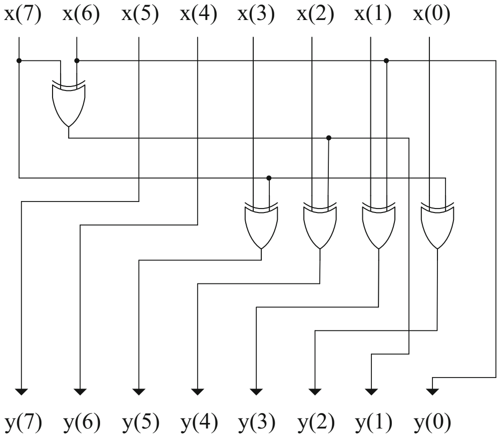

x(7) x(6) x(5) x(4) x(3) x(2) x(1) x(0)
y(7) y(6) y(5) y(4) y(3) y(2) y(1) y(0)

Fig. 25 Structure of X4Time block in GF(2^8)

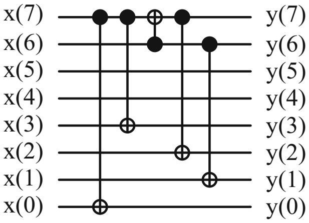


Fig. 26 Reversible gate design of X4Time block

as reversible MixColumns transformation block, reversible inverse blocks 1, 2 and 3, and reversible X4Time block are instantiated once, 4 times, 2 times, 2 times and 2 times respectively, and combined together to get the final design of the reversible InvMixColumns transformation block in AES crypto core as shown in Fig. 27. The reversible gate design of the combined block is made with 214 CNOT gates and it takes 48 ancilla inputs and 48 garbage outputs. In addition, it has a quantum cost of 214 with 28 units of delay as given in Table 8.

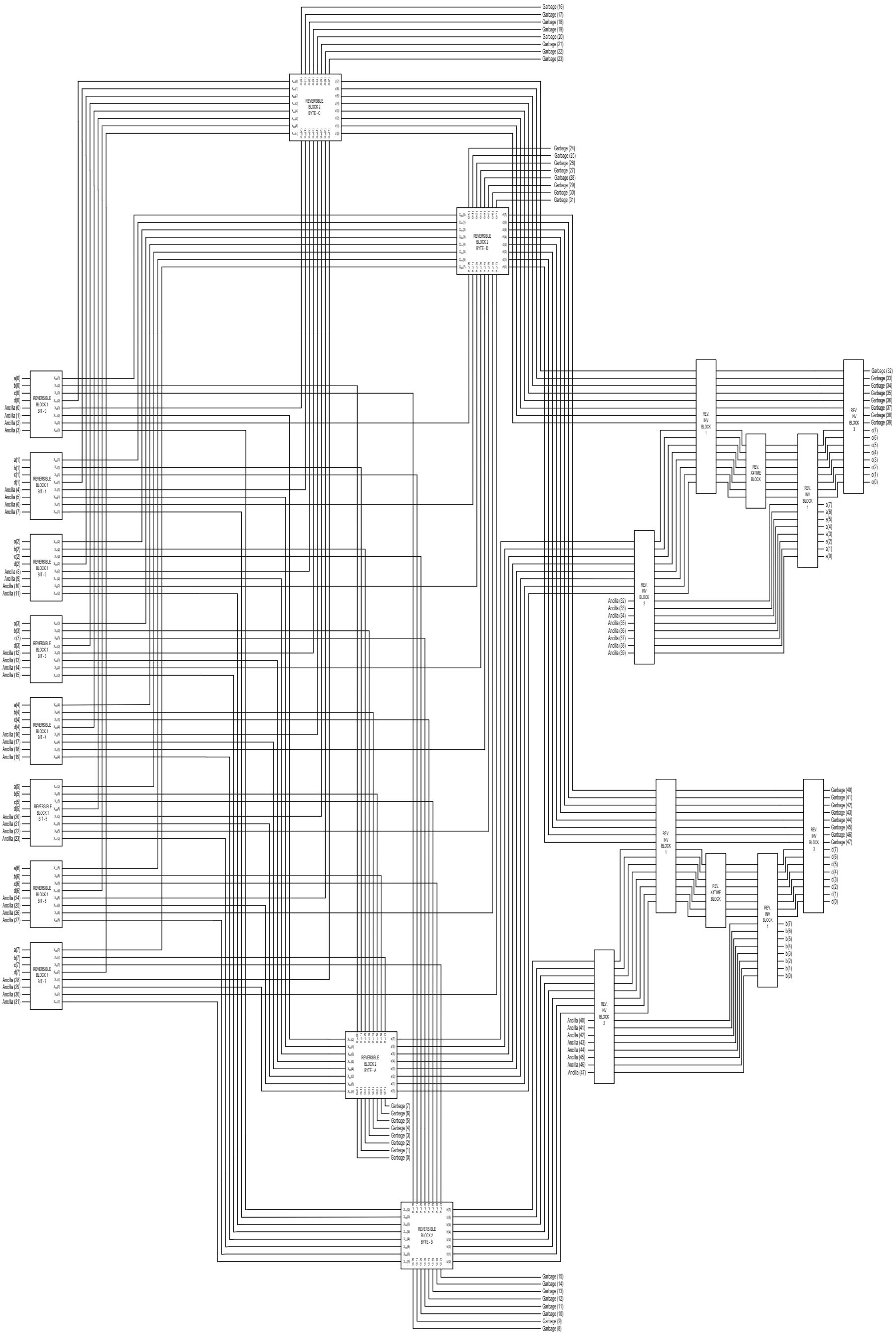


Fig. 27 Proposed reversible InvMixColumns transformation block

Table 8 Performance analysis of reversible invmixcolumns transformation module

Metrics	Reversible MixColumns design	Inverse block 1	Inverse block 2	Inverse block 3	X4Time	Proposed reversible block
Ancilla inputs	32	0	8	0	0	48
Garbage outputs	32	0	0	8	0	48
No. of reversible gates	140	8	8	8	5	214
Quantum cost	140	8	8	8	5	214
Delay	19	1	1	1	5	28

4.2.3 Performance Analysis

This section describes the performance analysis of the proposed reversible gate design of MixColumns and InvMixColumns transformations. Table 9 compares the performance metrics of different reversible MixColumns transformation designs. The proposed reversible InvMixColumns design gives 7% reduction in gate count and quantum cost compared to existing design [22]. The simulation output of both reversible MixColumns and InvMixColumns transformations are shown in Figs. 28 and 29 respectively for three random sets of inputs.

Since gate count and quantum cost are the important metrics for reversible gate design, they are considered here for comparison with existing designs. The proposed reversible MixColumns design gives 10% reduction in gate count and quantum cost compared to existing design [20]. Table 10 compares the performance metrics of different reversible InvMixColumns transformation designs. The proposed reversible MixColumns transformation module shows 81% reduction in gate count and 93% reduction in quantum cost compared to the existing design [13] as given in Table 11.

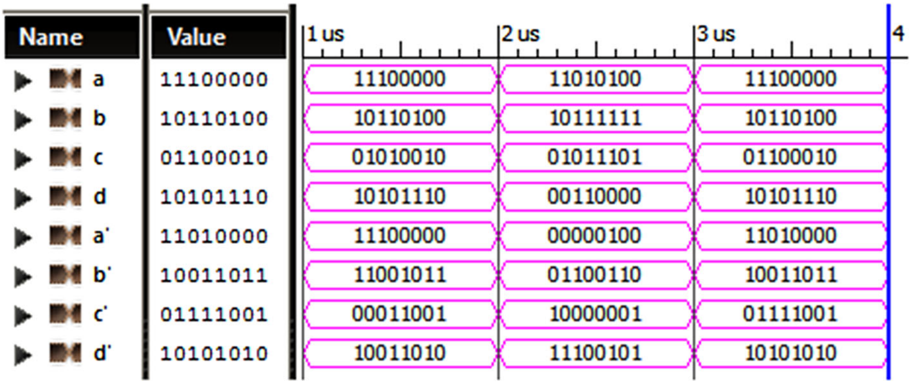
4.3 Shiftrows and InvShiftRows Transformations

4.3.1 InvShiftRows Transformation

The ShiftRows transformation provides a simple permutation of the data, whereas the other transformations involve substitutions. Further, since the state is treated as a block of

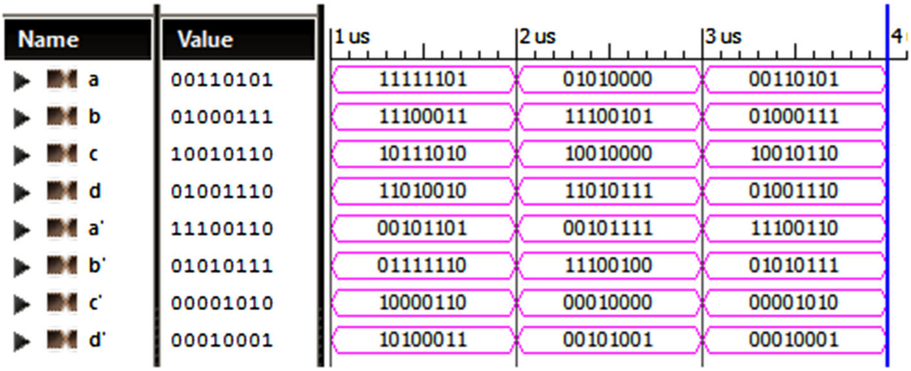
Table 9 Performance comparison of reversible mixcolumns transformation modules

Metrics	[21]	[20]	Proposed design	% Reduction
No. of reversible gates	CNOT—212	CNOT—156	CNOT—140	10
Quantum cost	212	156	140	10
Delay	10	13	19	–
Ancilla input	32	32	32	–
Garbage output	72	40	32	–



Name	Value	1 us	2 us	3 us
a	11100000	11100000	11010100	11100000
b	10110100	10110100	10111111	10110100
c	01100010	01010010	01011101	01100010
d	10101110	10101110	00110000	10101110
a'	11010000	11100000	00000100	11010000
b'	10011011	11001011	01100110	10011011
c'	01111001	00011001	10000001	01111001
d'	10101010	10011010	11100101	10101010

Fig. 28 Functional verification of the proposed reversible MixColumns transformation block



Name	Value	1 us	2 us	3 us
a	00110101	11111101	01010000	00110101
b	01000111	11100011	11100101	01000111
c	10010110	10111010	10010000	10010110
d	01001110	11010010	11010111	01001110
a'	11100110	00101101	00101111	11100110
b'	01010111	01111110	11100100	01010111
c'	00001010	10000110	00010000	00001010
d'	00010001	10100011	00101001	00010001

Fig. 29 Functional verification of the proposed reversible InvMixColumns transformation

Table 10 Performance comparison of reversible invmixcolumns transformation modules

Metrics	[18]	[20]	[22]	Proposed design	% Reduction
No. of reversible gates	CNOT—265	CNOT—232	CNOT—230	CNOT—214	7
Quantum cost	265	232	230	214	7
Delay	37	23	30	28	–
Ancilla input	80	48	48	48	–
Garbage output	72	56	48	48	–

columns, it is this step which provides the diffusion of values between columns. The ShiftRows step operates on the rows of the state. It cyclically shifts the bytes in each row. The first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are left shifted by two and three respectively.

Table 11 Performance comparison of reversible mixcolumns transformation module

Functional block	Gate count		Quantum cost		% Reduction	
	[13]	Proposed design	[13]	Proposed design	Gate count	Quantum cost
Reversible MixColumns design	736	140	1952	140	81	93

4.3.2 InvShiftRows Transformation

InvShiftRows is the inverse of the ShiftRows transformation. The bytes in the last three rows of the state are cyclically shifted over different numbers of bytes. The first row is not shifted. Each byte of the second row is shifted one to the right. Similarly, the third and fourth rows are left shifted by two and three respectively. Both ShiftRows and InvShiftRows transformations involve cyclic shifting of bytes of data either to the right or to the left direction. Hence, these transformations do not require any specific reversible gates to perform the operations.

4.4 AddRoundKey Transformation

In the AddRoundKey Transformation, roundkey is added by combining each byte of the state with the corresponding byte of the roundkey by bitwise XOR operation. For each round, the roundkey is derived from the main key using Rijndael's key schedule; each roundkey is the same size as the state. This is the only step which makes use of the key and obtains the result. So, this step is used at the end of each round. The transformation can be efficiently implemented in reversible logic using a CNOT gate for every bit. A total of 128 CNOT gates are required for realizing this transformation. Table 12 shows the results in terms of number of gates, quantum cost, delay, number of garbage outputs and number of constant inputs for AddRoundKey transformation.

4.5 Proposed Reversible Key Scheduler

The Key Scheduler is responsible for generating the round keys which are used in the AddRoundKey transformation. The Key Scheduler module has three essential steps:

1. The RotWord() step which cyclically shifts each byte in a word one byte to the left can be implemented by wiring alone.

Table 12 Performance analysis of AddRoundKey transformation module

Functional block	Number of gates	Quantum cost	Delay	Garbage output	Ancilla input
AddRoundKey transformation	CNOT—128	128	1	128	0

2. The SubWord() step which applies SubBytes transformation to each of the four bytes in a word requires the same S-boxes as used in the main encryption flow.
3. The Rcon is a constant word array and only the leftmost byte in each word is nonzero. As only one byte of Rcon is nonzero, and thus, only 8 CNOT gates are required to XOR the Rcon value with the output which is obtained from the previous step.
4. An 128-bit XOR operation is done by using 128 CNOT gates for the final step.

Reversible logic gate design of key scheduler for 128-bit AES algorithm is shown in Fig. 30 in which i varies from 1 to Nr. For each round, the round keys are generated by varying i from 1 to 10. Table 13 gives the performance metrics in terms of number of reversible gates, quantum cost, delay, number of garbage outputs and number of ancilla inputs required for designing RotWord operation, SubBytes transformation, xoring with round constant (rcon), xor operation and complete Key Scheduler using reversible logic gates. The RotWord operation is permutation of bits, so it does not require reversible gates.

The functional verification of the reversible key scheduler is shown in Fig. 31. The performance improvement of the proposed reversible key scheduler is given in Table 14. The proposed reversible key scheduler shows 33% reduction in gate count and 97% reduction in quantum cost compared to the existing design [13].

5 Performance Analysis of Reversible AES Design

5.1 Proposed Reversible AES Encryption Module

The 128-bit AES encryption algorithm takes 128-bit plaintext as input alongwith an 128-bit key and produces 128-bit ciphertext as output. The number of rounds (Nr) is 10 for 128-bit keys. The reversible logic gate design of 128-bit AES encryption requires 11 AddRoundKey transformations, 10 SubBytes transformations, 9 MixColumns transformations and 10 ShiftRows transformations. The Fanout module is designed using 128 CNOT gates with one unit delay and has a quantum cost of 128, garbage outputs of 128 and zero ancilla inputs. The reversible gate design of AES encryption algorithm is shown in Fig. 32.

From Fig. 32, it can be observed that the proposed reversible gate design of AES encryption module requires reversible AddRoundKey block, reversible SubBytes transformations, reversible MixColumns transformations and reversible Key schedulers. As ShiftRows involve cyclic shifting of state elements, they do not require any reversible logic

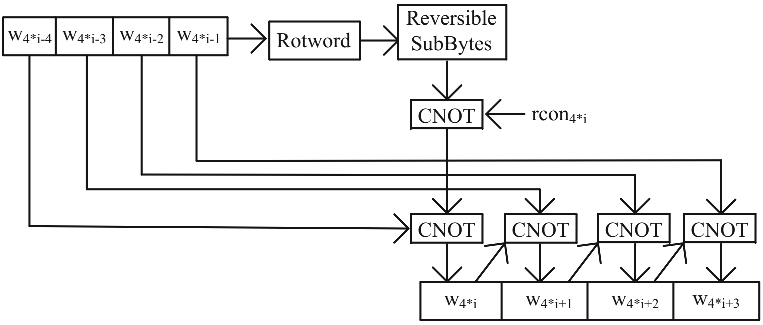


Fig. 30 Proposed reversible key scheduler

Table 13 Performance analysis of proposed reversible key scheduler

Functional block	Number of gates	Quantum cost	Delay	Garbage output	Ancilla input
RotWord	0	0	0	0	0
SubBytes transformation	CNOT—152 NOT—4 CCNOT—35	331	104	67	47
XOR with rcon	CNOT—8	8	1	8	0
XOR operation	CNOT—32	32	1	32	0
Key scheduler	CNOT—2592 NOT—64 CCNOT—560	5456	109	1232	752

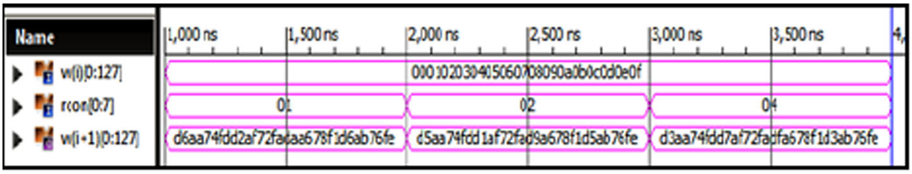

Name
1,000 ns
1,500 ns
2,000 ns
2,500 ns
3,000 ns
3,500 ns
4,
w(i)[0:127]
rcon[0:7]
w(i+1)[0:127]
000102030405060708090a0b0c0d0e0f
01
02
04
d6aa74fdd2af72fadaa678f1d6ab76fe
d5aa74fdd1af72fad9a678f1d5ab76fe
d3aa74fdd7af72fadfa678f1d3ab76fe

Fig. 31 Functional verification of the proposed reversible key scheduler

Table 14 Performance comparison of reversible key schedulers

Functional block	Gate count		Quantum cost		% Reduction	
	[13]	Proposed design	[13]	Proposed design	Gate count	Quantum cost
Reversible key scheduler	4832	3216	185,760	5456	33	97

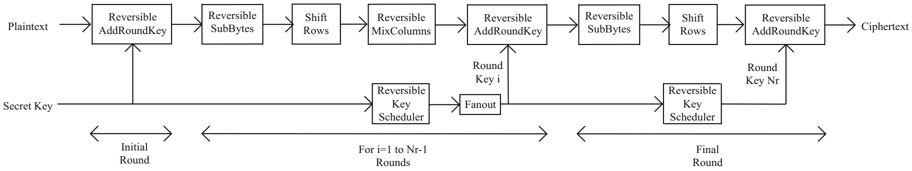
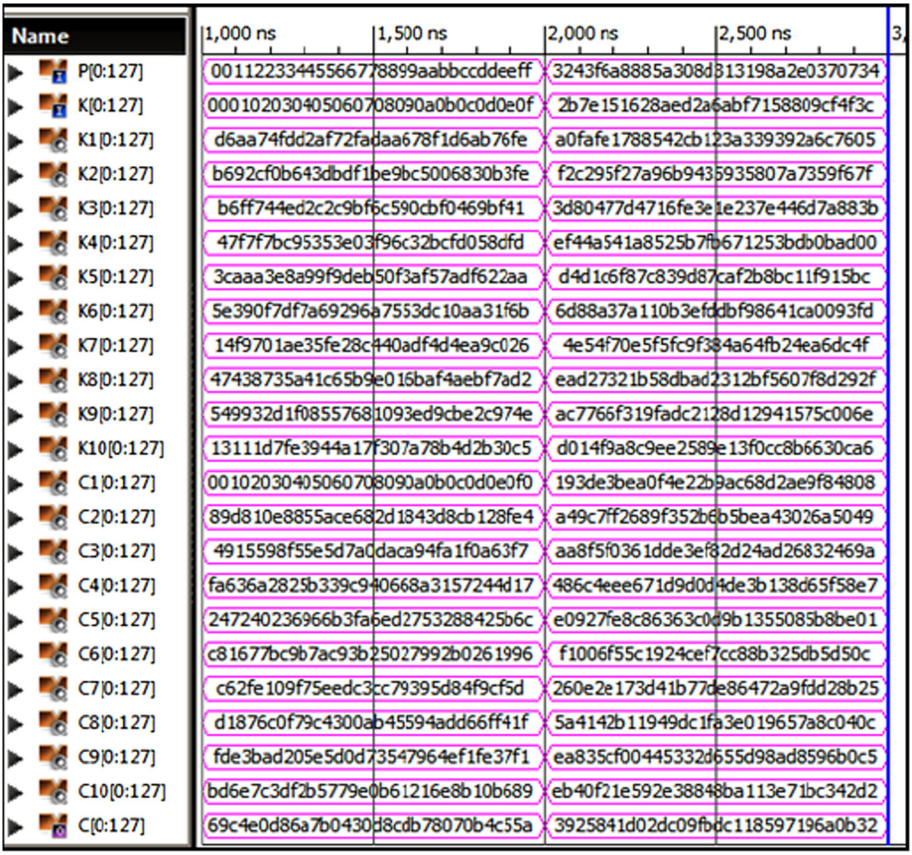


Fig. 32 Proposed reversible AES encryption module

gates. The remaining transformations are properly synthesized using reversible logic gates and the optimized designs are used to build the proposed reversible AES encryption module. Table 15 gives the performance metrics of proposed reversible gate design of 128-bit AES encryption module. The functional verification of the proposed reversible 128-bit AES module is shown in Fig. 33.

Table 15 Performance analysis of proposed reversible AES encryption module

Functional block	Number of gates	Quantum cost	Delay	Garbage output	Ancilla input
Round 0 (initial round)	CNOT—2720 NOT—64 CCNOT—560	5584	109	1360	752
Rounds 1–9	CNOT—52,560 NOT—1152 CCNOT—10,080	104,112	1918	24,192	14,688
Final round	CNOT—2560 NOT—64 CCNOT—560	5424	105	1200	752
128-AES encryption	CNOT—57,840 NOT—1280 CCNOT—11,200	115,120	2132	26,752	16,192



```
P[0:127]     00112233445566778899aabbccddeeff  3243f6a8885a308d313198a2e0370734
K[0:127]     000102030405060708090a0b0c0d0e0f  2b7e151628aed2a6abf7158809cf4f3c
K1[0:127]    d6aa74fdd2af72fadaa678f1d6ab76fe  a0fafe1788542cb123a339392a6c7605
K2[0:127]    b692cf0b643dbdf1be9bc5006830b3fe  f2c295f27a96b9435935807a7359f67f
K3[0:127]    b6ff744ed2c2c9bf6c590cbf0469bf41  3d80477d4716fe3e1e237e446d7a883b
K4[0:127]    47f7f7bc95353e03f96c32bcfd058dfd  ef44a541a8525b7fb671253bdb0bad00
K5[0:127]    3caaa3e8a99f9deb50f3af57adf622aa  d4d1c6f87c839d87caf2b8bc11f915bc
K6[0:127]    5e390f7df7a69296a7553dc10aa31f6b  6d88a37a110b3efddbf98641ca0093fd
K7[0:127]    14f9701ae35fe28c440adf4d4ea9c026  4e54f70e5f5fc9f384a64fb24ea6dc4f
K8[0:127]    47438735a41c65b9e016baf4aebf7ad2  ead27321b58dbad2312bf5607f8d292f
K9[0:127]    549932d1f08557681093ed9cbe2c974e  ac7766f319fadc2128d12941575c006e
K10[0:127]   13111d7fe3944a17f307a78b4d2b30c5  d014f9a8c9ee2589e13f0cc8b6630ca6
C1[0:127]    00102030405060708090a0b0c0d0e0f0  193de3bea0f4e22b9ac68d2ae9f84808
C2[0:127]    89d810e8855ace682d1843d8cb128fe4  a49c7ff2689f352b6b5bea43026a5049
C3[0:127]    4915598f55e5d7a0daca94fa1f0a63f7  aa8f5f0361dde3ef82d24ad26832469a
C4[0:127]    fa636a2825b339c940668a3157244d17  486c4eee671d9d0d4de3b138d65f58e7
C5[0:127]    247240236966b3fa6ed2753288425b6c  e0927fe8c86363c0d9b1355085b8be01
C6[0:127]    c81677bc9b7ac93b25027992b0261996  f1006f55c1924cef7cc88b325db5d50c
C7[0:127]    c62fe109f75eedc3cc79395d84f9cf5d  260e2e173d41b77de86472a9fdd28b25
C8[0:127]    d1876c0f79c4300ab45594add66ff41f  5a4142b11949dc1fa3e019657a8c040c
C9[0:127]    fde3bad205e5d0d73547964ef1fe37f1  ea835cf00445332d655d98ad8596b0c5
C10[0:127]   bd6e7c3df2b5779e0b61216e8b10b689  eb40f21e592e38848ba113e71bc342d2
C[0:127]     69c4e0d86a7b0430d8cdb78070b4c55a  3925841d02dc09fbdc118597196a0b32
```

Fig. 33 Functional verification of the proposed reversible AES encryption module

5.2 Proposed Reversible AES Decryption Module

The top-level structure of 128-bit AES decryption algorithm takes 128-bit ciphertext as input along with an 128-bit key and produces an 128-bit plaintext as output. There are four distinct operations: InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey. After an initial AddRoundKey transformation, Nr-1 rounds are performed which consists of a sequence of four operations. The number of rounds is 10 for 128-bit key length. The final round excludes InvMixColumns transformation. The reversible gate design of 128-bit AES decryption module using Inverse Cipher is shown in Fig. 34. It requires 11 AddRoundKey transformations, 10 InvSubBytes transformations, 9 InvMixColumns transformations and 10 InvShiftRows transformations.

The reversible gate design of 128-bit AES decryption module using Equivalent Inverse Cipher is shown in Fig. 35. It requires 11 AddRoundKey transformations, 10 InvSubBytes transformations, 18 InvMixColumns transformations and 10 InvShiftRows transformations. The performance comparison of 128-bit AES decryption using inverse cipher and equivalent inverse cipher is given in Table 16. In the equivalent inverse cipher, the mixroundkeys are used. The mixroundkeys are the modified roundkeys obtained by applying InvMixColumns to the roundkeys. Thus, the quantum cost and number of reversible gates required to design the reversible equivalent inverse cipher are more compared to the reversible inverse cipher. Hence, in the proposed reversible gate design of AES decryption module, inverse cipher method is used to decrypt the ciphertext. Table 17 gives the performance metrics of proposed reversible gate design of 128-bit AES decryption module. The functional verification of the proposed reversible 128-bit AES module is shown in Fig. 36.

5.3 Performance Improvement

The proposed reversible SubBytes/InvSubBytes transformation module of AES crypto core shows 36% reduction in gate count and 35% reduction in quantum cost compared to the conventional reversible designs. In addition, the proposed design shows 35% reduction in gate count and 97% reduction in quantum cost compared to the existing design of reversible SubBytes and InvSubBytes transformation module.

The proposed reversible gate design of MixColumns and InvMixColumns transformation of AES crypto core based on CNOT gates show 81 and 93% reduction in gate count and quantum cost respectively compared to existing designs. The proposed reversible key scheduler of AES crypto core shows 33% reduction in gate count and 97% reduction in quantum cost compared to the existing design. The performance improvement in all the proposed designs is mainly due to the reuse of already available reversible gates to the maximum extent. The complete AES encryption process takes 16,192 ancilla inputs,

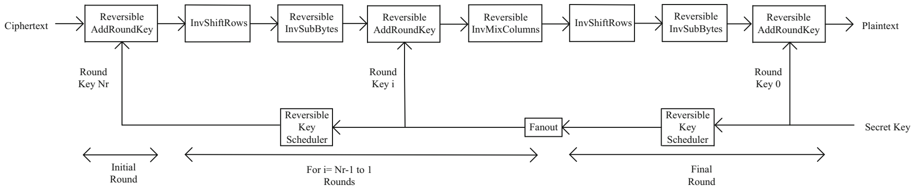


Fig. 34 Proposed reversible AES decryption module using inverse cipher method

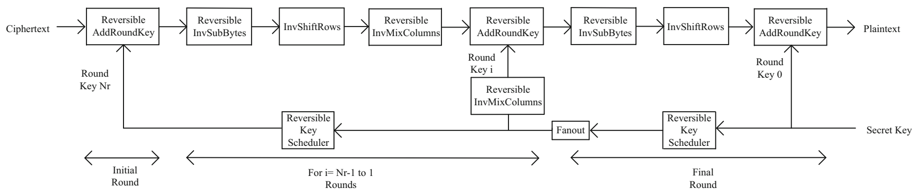


Fig. 35 Proposed reversible AES decryption module using equivalent inverse cipher method

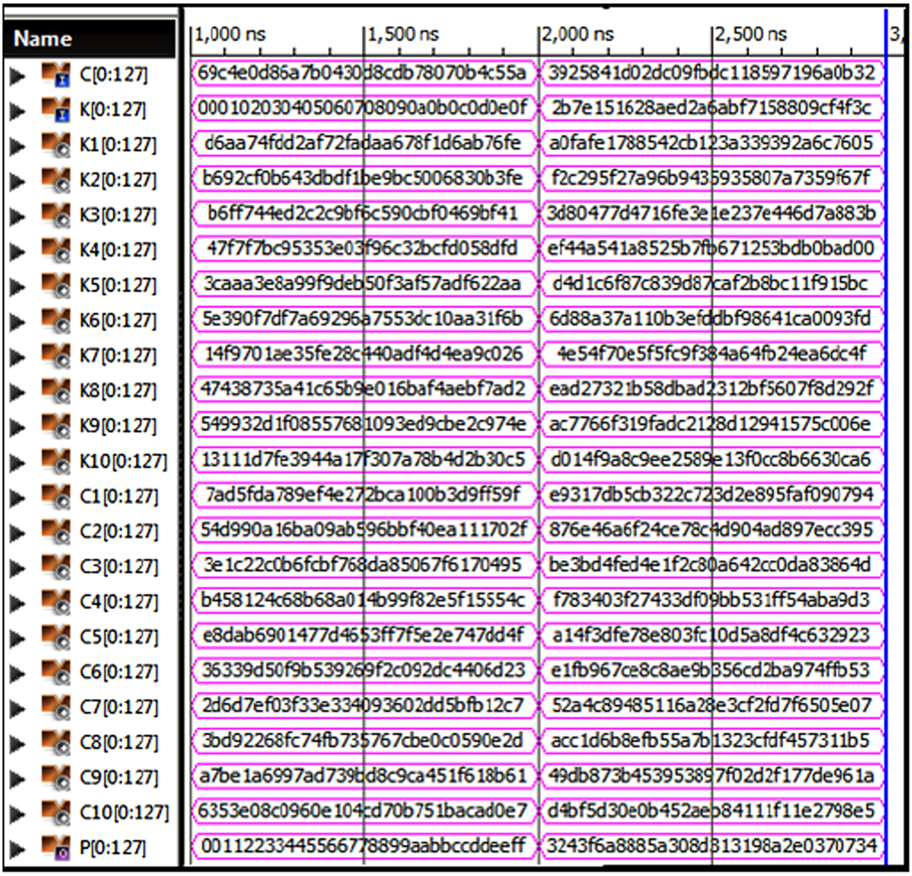
Table 16 Performance comparison of reversible AES decryption modules

Parameters‘	Inverse cipher method	Equivalent inverse cipher method
Number of gates	CNOT—60,504 NOT—1280 CCNOT—11,200	CNOT—62,430 NOT—1280 CCNOT—11,200
Quantum cost	117,784	119,710
Delay	1412	1440
Garbage output	27,328	27,760
Ancilla input	16,768	17,200

Table 17 Performance analysis of proposed reversible AES decryption module

Functional block	Number of gates	Quantum cost	Delay	Garbage output	Ancilla input
Round 0 (initial round)	CNOT—2720 NOT—64 CCNOT—560	5584	109	1360	752
Rounds 1–9	CNOT—55,224 NOT—1152 CCNOT—10,080	106,776	1198	24,768	15,264
Final round	CNOT—2560 NOT—64 CCNOT—560	5424	105	1200	752
AES decryption	CNOT—60,504 NOT—1280 CCNOT—11,200	117,784	1412	27,328	16,768

26,752 garbage outputs with a delay of 2132 and a total of 70,320 reversible logic gates with a quantum cost of 115,120. The complete AES decryption process takes 16,768 ancilla inputs, 27,328 garbage outputs with a delay of 1412 and a total of 72,984 reversible logic gates with a quantum cost of 117,784. In summary, our proposed AES reversible design gives 36% reduction in gate count and 97% reduction in quantum cost when compared to the existing design [13] as shown in Table 18. This considerable reduction in the performance metrics is mainly due to the reuse of the already existing gates in the structures.



```
C[0:127]     69c4e0d86a7b0430d8cdb78070b4c55a   3925841d02dc09fbdc118597196a0b32
K[0:127]     000102030405060708090a0b0c0d0e0f   2b7e151628aed2a6abf7158809cf4f3c
K1[0:127]    d6aa74fdd2af72fadaa678f1d6ab76fe   a0fafe1788542cb123a339392a6c7605
K2[0:127]    b692cf0b643dbdf1be9bc5006830b3fe   f2c295f27a96b9435935807a7359f67f
K3[0:127]    b6ff744ed2c2c9bf6c590cbf0469bf41   3d80477d4716fe3e1e237e446d7a883b
K4[0:127]    47f7f7bc95353e03f96c32bcfd058dfd   ef44a541a8525b7fb671253bdb0bad00
K5[0:127]    3caaa3e8a99f9deb50f3af57adf622aa   d4d1c6f87c839d87caf2b8bc11f915bc
K6[0:127]    5e390f7df7a69296a7553dc10aa31f6b   6d88a37a110b3efddbf98641ca0093fd
K7[0:127]    14f9701ae35fe28c440adf4d4ea9c026   4e54f70e5f5fc9f384a64fb24ea6dc4f
K8[0:127]    47438735a41c65b9e016baf4aebf7ad2   ead27321b58dbad2312bf5607f8d292f
K9[0:127]    549932d1f08557681093ed9cbe2c974e   ac7766f319fadc2128d12941575c006e
K10[0:127]   13111d7fe3944a17f307a78b4d2b30c5   d014f9a8c9ee2589e13f0cc8b6630ca6
C1[0:127]    7ad5fda789ef4e272bca100b3d9ff59f   e9317db5cb322c723d2e895faf090794
C2[0:127]    54d990a16ba09ab596bbf40ea111702f   876e46a6f24ce78c4d904ad897ecc395
C3[0:127]    3e1c22c0b6fcbf768da85067f6170495   be3bd4fed4e1f2c80a642cc0da83864d
C4[0:127]    b458124c68b68a014b99f82e5f15554c   f783403f27433df09bb531ff54aba9d3
C5[0:127]    e8dab6901477d4653ff7f5e2e747dd4f   a14f3dfe78e803fc10d5a8df4c632923
C6[0:127]    36339d50f9b539269f2c092dc4406d23   e1fb967ce8c8ae9b356cd2ba974ffb53
C7[0:127]    2d6d7ef03f33e334093602dd5bfb12c7   52a4c89485116a28e3cf2fd7f6505e07
C8[0:127]    3bd92268fc74fb735767cbe0c0590e2d   acc1d6b8efb55a7b1323cfdf457311b5
C9[0:127]    a7be1a6997ad739bd8c9ca451f618b61   49db873b453953897f02d2f177de961a
C10[0:127]   6353e08c0960e104cd70b751bacad0e7   d4bf5d30e0b452aeb84111f11e2798e5
P[0:127]     00112233445566778899aabbccddeeff   3243f6a8885a308d313198a2e0370734
```

Fig. 36 Functional verification of proposed reversible AES decryption module using inverse cipher method

Table 18 Performance improvements in proposed reversible AES design

Functional block	Gate count		Quantum cost		% Reduction	
	[13]	Proposed design	[13]	Proposed design	Gate count	Quantum cost
SubByte module	294	191	11,602	331	35	97
MixColumns module	736	140	1952	140	81	93
AddRoundKey module	128	128	128	128	–	–
Key scheduler module	4832	3216	185,760	5456	33	97
Reversible AES design	109,664	70,320	3749,408	115,120	36	97

6 Conclusion

A novel reversible gate design of complete 128-bit AES algorithm is presented. Since the reversible logic gates ideally consume zero power and their quantum computing based implementation is less sensitive to power analysis attacks, they are exploited to construct the AES algorithm in this work. The Toffoli family of reversible gates are used in the proposed designs and the reversible logic gates are reused as much as possible in order to optimize the performance metrics in the proposed structures. The proposed reversible gate design of AES algorithm gives 36% reduction in gate count and 97% reduction in quantum cost when compared to the existing design. Hence, the proposed design can be effectively used to protect confidential data in low power and secure applications such as wireless sensor networks.

References

1. Merkle, R. C. (1993). Two types of mechanical reversible logic. *Nanotechnology, 4*(2), 114–131.
2. Younism, S. G., & Knight, T. F. (1994). Asymptotically zero energy split-level charge recovery logic. In *Proceedings of international workshop on low power design* (pp. 177–182).
3. Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development, 5*(3), 183–191.
4. Bennett, C. (1973). Logical reversibility of computation. *IBM Journal of Research and Development, 17*(6), 525–532.
5. Schneier, B. (1996). *Applied cryptography*. New York: Wiley.
6. National Institute of Standard and Technology (NIST). (2001). Advanced Encryption Standard (AES), FIPS-197.
7. Chodowiec, P., & Gaj, K. (2003). Very compact FPGA implementation of the AES algorithm. In *Proceedings of cryptographic hardware and embedded systems* (pp. 319–333).
8. Chih-Pin, S., et al. (2003). A high-throughput low-cost AES processor. *IEEE Communications Magazine, 41*(12), 86–91.
9. Saravanan, P., & Kalpana, P. (2013). A novel and systematic approach to implement reversible gates in quantum dot cellular automata. *WSEAS Transactions on Circuits and Systems, 12*(10), 307–316.
10. Saravanan, P., & Kalpana, P. (2014). Energy efficient reversible building blocks resistant to power analysis attacks. *Journal of Circuits, Systems and Computers, 23*(9), 1450127-1–1450127-40.
11. Thapliyal, H., & Zwolinski, M. (2006). Reversible logic to cryptographic hardware: A new paradigm. In *49th IEEE international midwest symposium on circuits and systems* (Vol. 1).
12. Nayeem, N. M., Jamal, L., & Babu, H. M. H. (2009). Efficient reversible montgomery multiplier and its application to hardware cryptography. *Journal of Computer Science, 5*(1), 49–56.
13. Datta, K., Shrivastav, V., Sengupta, I., & Rahaman, H. (2013). Reversible logic implementation of AES algorithm. In *Proceedings of 8th international conference on design & technology of integrated systems in nanoscale era* (pp. 140–144).
14. Saravanan, P., & Kalpana, P. (2015). Design of SubBytes and InvSubBytes transformations of AES algorithm using power analysis attack resistant reversible logic gates. *Australian Journal of Basic and Applied Sciences, 9*(1), 8–18.
15. Saravanan, P., & Kalpana, P. (2015). Performance analysis of reversible finite field arithmetic architectures over GF(p) and GF(2m) in elliptic curve cryptography. *Journal of Circuits, Systems and Computers, 24*(8), 1550122–1550150.
16. Robert, W. (2011). An introduction to reversible circuit design. In *Saudi international electronics, communications and photonics conference*.
17. Rudra, A., Dubey, P. K., Jutla, C. S., Vijay Kumar, Rao, J. R., & Rohatgi, P. (2001). Efficient Rijndael encryption implementation with composite field arithmetic. In *Proceedings of 3rd international workshop on cryptographic hardware and embedded systems* (pp. 175–188).
18. Zhang, X., & Parhi, K. K. (2004). High-speed VLSI architectures for the AES algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12*(9), 957–967.
19. Mui, E. (2007). *Practical implementation of Rijndael S-Box using Combinational logic*. <http://www.xess.com/static/media/projects/Rijndael_SBox.pdf>.

20. Fischer, V., Drutarovsky, M., Chodowiec, P., & Gramain, F. (2005). InvMixColumn decomposition and multilevel resource sharing in AES implementations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13*(8), 989–992.
21. Hua, L., & Friggstad, Z. (2005). An efficient architecture for the AES mix columns operation. *IEEE International Symposium on Circuits and Systems, 5,* 4637–4640.
22. Nalini, C., Anandmohan, P. V., Poornaiah, D. V. (2010). Mix/InvMixColumn decomposition and resource sharing in AES. In *International conference on industrial and information systems* (pp. 166–171).

Dr. P. Saravanan completed his B.E. in Electrical and Electronics Engineering in the year 2007 securing first class with distinction at Thiagarajar College of Engineering, Madurai and M.E. VLSI Design with first class and distinction at PSG College of Technology, Coimbatore in 2009. He completed his Ph.D. at Anna University, Chennai in 2015. He joined the Department of Electronics and Communication Engineering, PSG College of Technology, Coimbatore as an Assistant Professor in 2009 and he is currently serving as an Associate Professor. His main areas of research interest are VLSI Design, Verification and Testing, Hardware Security and Multi-scale Modeling of Nanoelectronic Devices. He has published many national, international journals and attended conferences related to his area. His credits also include nearly 5 years of industrial experience as well. He is a member of IEEE, IETE, ISTE, ISSS and VLSI Society of India.

Dr. P. Kalpana completed her B.E. in Electronics and Communication Engineering (First class with distinction) in the year 1992 and M.E. in Applied Electronics in 2002 from PSG College of Technology. She completed her Ph.D. in the area of VLSI Testing in 2009 from Anna University, Chennai. She is currently working as Professor in the department of ECE, PSG College of technology. She has organized International Guest faculty workshop, Short term courses and Conferences in the area of VLSI design. She was the Coordinator for the VLSI—SMDP II project funded by Ministry of Electronics and Information Technology, New Delhi during 2009–2013 and Principal Investigator for a TEPP project funded by DSIR. She is a Life member of ISTE and IETE. She has published 35 papers in national and international journals and conference publications.