

FPGA Implementations of S-box vs. T-box iterative architectures of AES

Bhupathi Kakarlapudi and Nitin Alabur

Abstract

Cryptographic standards such as AES play a major role in developing the new cryptographic standards. AES is a block cipher that has been analyzed extensively and is among the popular ones for use in symmetric key cryptography. This paper evaluates the hardware implementations of S-box and T-box architectures of the AES cipher on Xilinx Spartan3E and Virtex5 FPGA families for the key sizes of 128, 192 and 256 bits and block size of 128 bits, as per the specifications by NIST.

Index Terms—AES, S-box, T-box

I. INTRODUCTION

In 1997, NIST (National Institute of Standards and Technology) initiated a contest to develop a Federal Information Processing Standard (FIPS) that specifies an encryption algorithm capable of protecting sensitive government information well into the next century. The contest known as AES, *Advanced Encryption Standard* was intended to be used by the U.S. Government and, on a voluntary basis, by the private sector. The winner of the contest, Rijndael was finally chosen after 5 years of extensive analysis and standardization process which was open and transparent unlike its predecessor (DES) Data Encryption Standard.

The software implementations of the T-box architecture have shown significant improvement in performance over S-box architecture especially for decryption and the combined encryption and decryption unit, thus motivating us to implement the AES algorithm using T-boxes in hardware. This architecture was first implemented in hardware on the Altera CPLDs by Viktor Fischer and Milos Drutarovsky. Our emphasis is on implementing the T-box architecture on Xilinx devices and to compare the results with the S-box architecture and with other known implementations [2].

II. S-BOX ARCHITECTURE OVERVIEW

This is the general architecture proposed for AES. In this architecture, the encryption process is a set of ten rounds (for a key size of 128 bits). The encryption starts with the addition of a round key with the plaintext and through the round

operations: subbytes, shift rows, mix columns and add round key. The round operations are performed for the N_r-1 iterations (where N_r is the total number of rounds, for key sizes of 128, 192 and 256, N_r is 10, 12 and 14 respectively) and for the final iteration mix columns operation is eliminated. Similarly decryption starts with add round key operation and skips the inverse mix column during the first iteration and performs the inverse shift rows, inverse sub bytes followed by add round key operations for the remaining nine iterations.

Since this implementation uses an 8x8 look-up table to perform the sub bytes operation it is known as the S-box architecture.

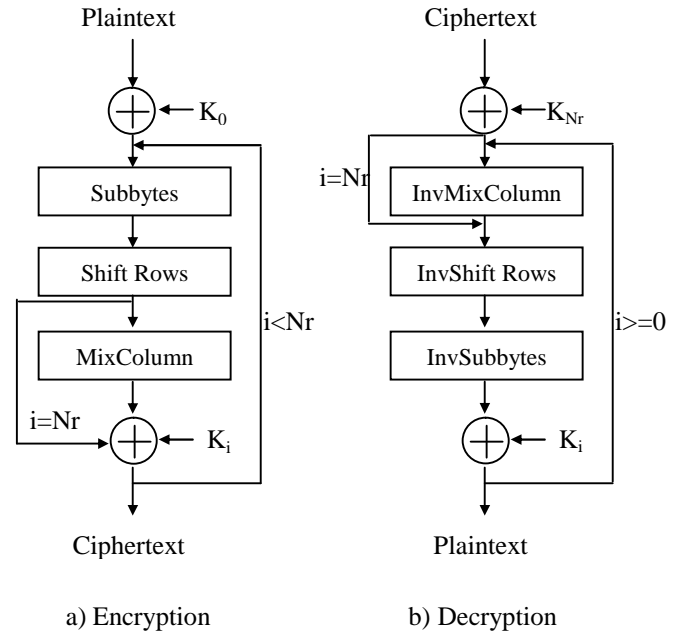


Fig.1. Structure of standard S-box Architecture of AES

Round operations:

Subbytes & Invsubbytes: This operation substitutes each byte of state input with the value from the 8x8-bit look up table of substitution box. The look up table for decryption is different from that of encryption.

Shiftrows & invshiftrows: This operation operates on the rows of state after the sub byte operation has been performed. The first row remains unchanged; the second row is circular left shifted by one byte for encryption and circular right shifted by one byte for decryption. Similarly the third and fourth rows are shifted by offsets of two and three bytes respectively.

MixColumns & InvMixcolumns: This operation takes each column of the state input, multiplies it with the elements of the matrix of size 4x4. Different matrices are used for encryption and decryption.

Encryption mix column function:

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

In the above transformation the ‘•’ represents multiplication in GF (2⁸)

Decryption inverse mix column function:

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \cdot \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

Instead of implementing this operation using the xtime function, it has been implemented with more efficient variable multiplication XOR logic. Since matrices in mix column and inverse mix column contain constant elements, these multiplications are replaced with several XOR operations that are simple to implement in FPGAs.

Add RoundKey: This operation performs logical XOR operation of the state with the round key.

III. T-BOX ARCHITECTURE OVERVIEW

The T-box approach allows the computation of the entire iteration of AES using only table look-ups called T-boxes and XOR operations. These precomputed T-box look up tables represent the combined operation of the subbytes and the mix column transformations [3]. Compared to the 8x8 bits wide S-box look up tables, the T-box tables are of the size of 8 x 32 bits. The mathematical description described below explains how T-box tables and the corresponding AES round operations are obtained:

T-box Tables derivation

State Input

| | | | |
|----------------|----------------|-----------------|-----------------|
| S ₀ | S ₄ | S ₈ | S ₁₂ |
| S ₁ | S ₅ | S ₉ | S ₁₃ |
| S ₂ | S ₆ | S ₁₀ | S ₁₄ |
| S ₃ | S ₇ | S ₁₁ | S ₁₅ |

Mixcolumn encryption operation to the first column of state Input

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \begin{pmatrix} 02*S_0 + 03*S_1 + 01*S_2 + 01*S_3 \\ 01*S_0 + 02*S_1 + 03*S_2 + 01*S_3 \\ 01*S_0 + 01*S_1 + 02*S_2 + 03*S_3 \\ 03*S_0 + 01*S_1 + 01*S_2 + 02*S_3 \end{pmatrix}$$

S4, S8, S12
S5, S9, S13
S6, S10, S14
S7, S11, S15
First row elements
Second Row
Third Row
Fourth Row

↓
↓
↓
↓
T₀
T₁
T₂
T₃

T-Box Tables for encryption:

$$T_0[a] = \begin{pmatrix} 02.S[a] \\ S[a] \\ S[a] \\ 03.S[a] \end{pmatrix} \quad T_1[a] = \begin{pmatrix} 03.S[a] \\ 02.S[a] \\ S[a] \\ S[a] \end{pmatrix}$$

$$T_2[a] = \begin{pmatrix} S[a] \\ 03.S[a] \\ 02.S[a] \\ S[a] \end{pmatrix} \quad T_3[a] = \begin{pmatrix} S[a] \\ S[a] \\ 03.S[a] \\ 02.S[a] \end{pmatrix}$$

T-Box Tables for Decryption:

$$T_0^{-1}[a] = \begin{pmatrix} 0E.S[a] \\ 09.S[a] \\ 0D.S[a] \\ 0B.S[a] \end{pmatrix} \quad T_1^{-1}[a] = \begin{pmatrix} 0B.S[a] \\ 0E.S[a] \\ 09.S[a] \\ 0D.S[a] \end{pmatrix}$$

$$T_2^{-1}[a] = \begin{pmatrix} 0D.S[a] \\ 0B.S[a] \\ 0E.S[a] \\ 09.S[a] \end{pmatrix} \quad T_3^{-1}[a] = \begin{pmatrix} 09.S[a] \\ 0D.S[a] \\ 0B.S[a] \\ 0E.S[a] \end{pmatrix}$$

In the above resultant matrix representation each column has fixed constants except that state inputs are updated with next row elements for the following iteration. First row elements of state input are always multiplied with first column

constants of the multiplication matrix. Similarly, the second row elements with second column of matrix and so on. The first row of the state input uses table T0, the second row uses the table T1, the third row uses the table T2 and the fourth row uses the table T3.

Round functions computation:

32-bit round output is obtained by giving the state input values to the T-box tables after the shift row operation. As shown in the example

$$\begin{aligned}
 e_0 &= T_0[00] \oplus T_1[05] \oplus T_2[10] \oplus T_3[15] \oplus K_1[0-31] \\
 e_1 &= T_0[04] \oplus T_1[09] \oplus T_2[14] \oplus T_3[03] \oplus K_1[32-63] \\
 e_2 &= T_0[08] \oplus T_1[13] \oplus T_2[02] \oplus T_3[07] \oplus K_1[64-95] \\
 e_3 &= T_0[12] \oplus T_1[01] \oplus T_2[06] \oplus T_3[11] \oplus K_1[96-127]
 \end{aligned}$$

Complete round function can be obtained by concatenating the 32-bit round outputs e0, e1, e2, e3.

Encryption & Decryption in T-box architecture

Round outputs of encryption and decryption iterations are computed from the look-up tables. Last round in both the cases is performed in a special way.

Last round in Encryption: Since mix columns operation has to be eliminated in the last round, this round is treated differently. In this round, sub byte values are needed instead of T-box outputs. Additional memory space is not needed to implement an additional sub byte table, as one byte outputs of the sub byte transformation can be extracted from the four-byte outputs of the T-box transformation corresponding to the same one-byte input as:

$$S[a]=\text{byte}(1,T_0[a])=\text{byte}(2,T_1[a])=\text{byte}(3,T_2[a])=\text{byte}(0,T_3[a])$$

Last round in Decryption:

In the last round of decryption, the InvMixcolumns operation is not executed. As a result, the outputs of the Invsubbytes transformation $S^{-1}[a]$ must be computed. Given the value of $T_0^{-1}[a]$, $S^{-1}[a]$ can be computed as follows [1]:

$$\begin{aligned}
 S^{-1}[a] &= 0E^{-1}.\text{byte}(0,T_0^{-1}[a]) = 09^{-1}.\text{byte}(1,T_0^{-1}[a]) = \\
 &= 0D^{-1}.\text{byte}(2,T_0^{-1}[a]) = 0B^{-1}.\text{byte}(3,T_0^{-1}[a])
 \end{aligned}$$

Where $\text{byte}(n,X)$ represents the n^{th} byte of a variable X. Based on the above equations each bit of $S^{-1}[a]$ can be computed using four different equations, each giving the same value. As a result, for each bit, we can choose an equation with the smallest number of terms [1].

Bits of $S^{-1}[a]$ are computed as follows:

$$\begin{aligned}
 x &= \text{byte}(0,T_0^{-1}[a]) \\
 y &= \text{byte}(1,T_0^{-1}[a]) \\
 z &= \text{byte}(3,T_0^{-1}[a])
 \end{aligned}$$

$$\begin{aligned}
 S_7^{-1} &= y_7 \oplus y_4 \oplus y_1 \\
 S_6^{-1} &= y_6 \oplus y_3 \oplus y_0 \\
 S_5^{-1} &= y_5 \oplus y_2 \\
 S_4^{-1} &= y_4 \oplus y_1 \\
 S_3^{-1} &= z_3 \oplus z_2 \oplus z_1 \\
 S_2^{-1} &= x_6 \oplus x_5 \oplus x_0 \\
 S_1^{-1} &= x_7 \oplus x_5 \oplus x_4 \\
 S_0^{-1} &= y_5 \oplus y_2 \oplus y_0
 \end{aligned}$$

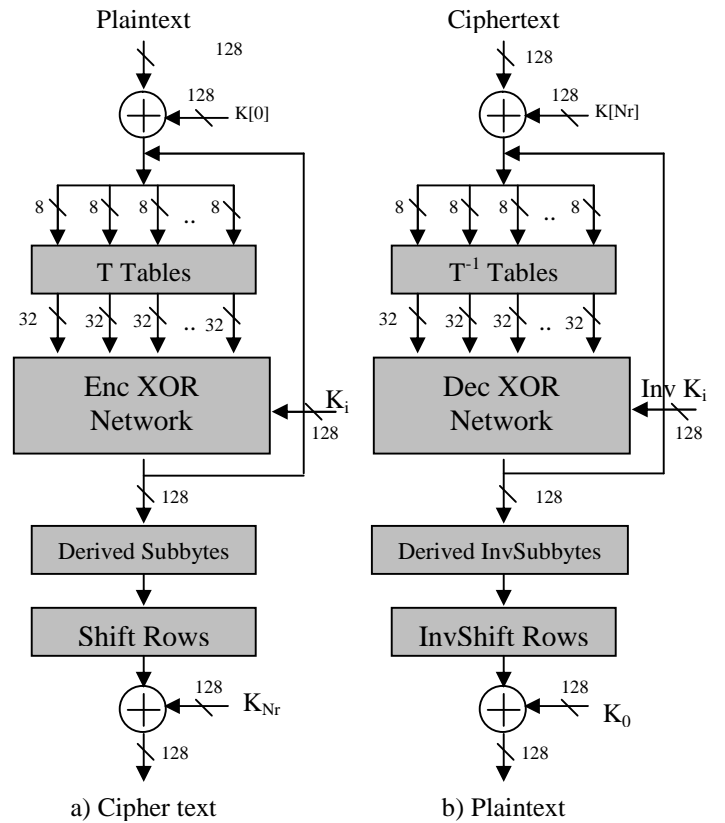


Fig.2. Structure of T-box Architecture of AES a) encryption algorithm b) Decryption algorithm.

Decryption algorithm shown in figure 2 for the T-box differs from that of the standard decryption. Except for the first and the last round keys, inverse round keys are used in the intermediate iterations in this decryption algorithm. Round modification for decryption is shown in figure 3.

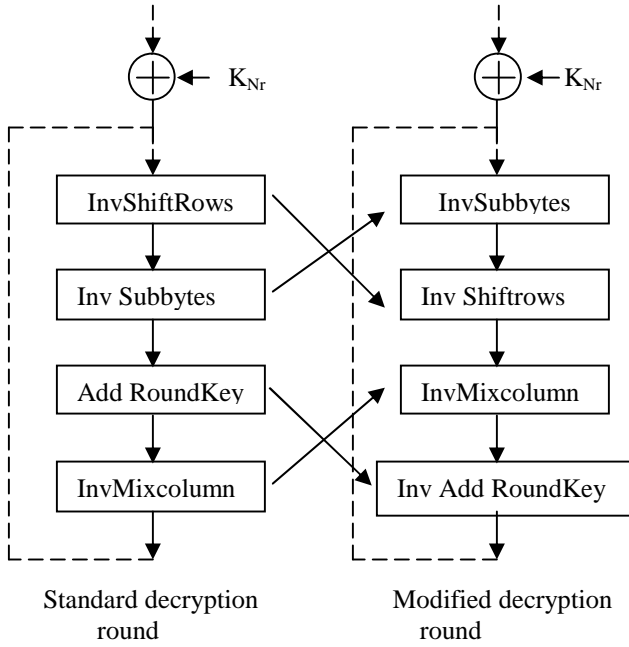


Fig 3. Deriving the modified decryption round

The operations Inverse Shift Rows and Inverse Subbytes can be swapped without affecting the result and since Inverse Mixcolumn is a linear transformation, the following equation is valid

$$\text{Inverse Mixcolumn } (d \oplus K) = \text{Inverse Mix Column } (d) \oplus \text{Inverse Mix Column } (K)$$

The reason for round modification in the case of T-box decryption is because the T-box outputs include sub bytes and mix column operations. This modification is very similar to the encryption round operation sequence and hence this method is appropriate for efficient implementation of this architecture.

IV. BASIC ITERATIVE ARCHITECTURE OF S-BOX & T-BOX

Basic Iterative Architecture:

In this architecture, input data is fed through the multiplexer and it is stored in the register and then passed through a one round combinational logic, the result of the combinational logic is fed back to the circuit through the multiplexer and stored in the register. This architecture can only encrypt one block of data at a time and the number of clock cycles necessary to encrypt a single block of data is equal to the total number of cipher rounds.

S-Box Basic Iterative Architecture:

The block diagram of the encryption/decryption unit based on S-boxes in the basic iterative architecture is shown in the

figure. Only one register is present in the basic iterative architecture. This register can be placed either before or after the combined Subbytes and InvSubbytes transformation, where encryption and decryption data paths converge. The critical path is located in the decryption circuit, and includes Invshift rows, Addroundkey, InvMixcolumns, a 3-to-1 multiplexer and Inv subbytes. This architecture, requires 11,13, and 15 clock cycles in order to process one block of data for 128, 192, and 256-bit keys respectively.

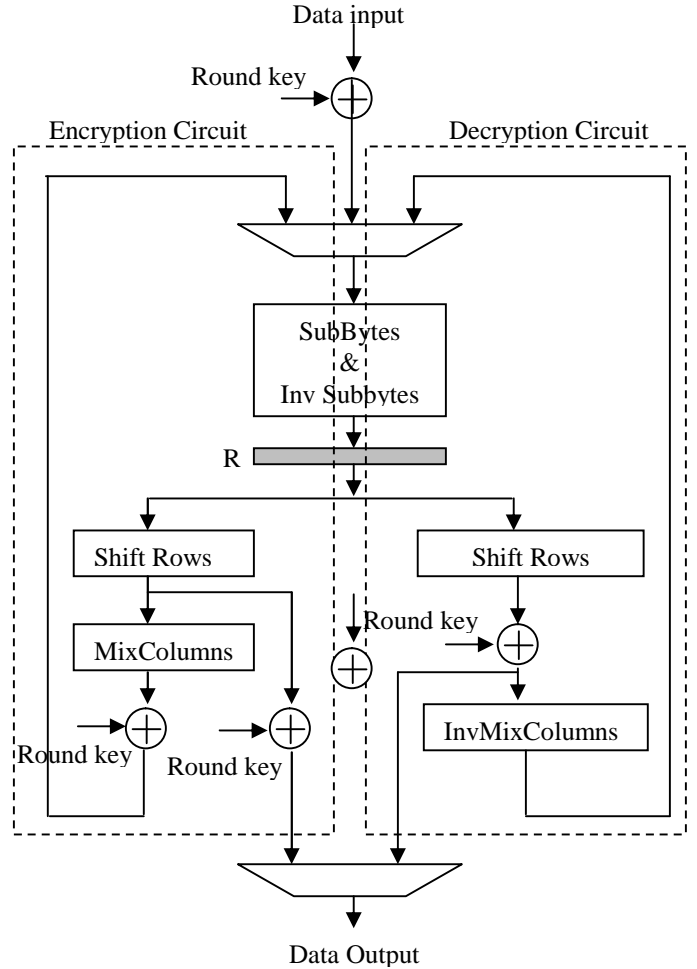


Fig 4: S-box Based Basic Iterative architecture

T-Box Basic Iterative Architecture:

The block diagram of the encryption/decryption unit based on T-boxes in the basic iterative architecture is shown in the figure. The critical path is located in the decryption circuit and includes T-boxes lookups, XOR network and Inv Mixcolumn for the round key. This architecture, also requires 11,13, and 15 clock cycles in order to process one block of data for 128, 192, and 256-bit keys respectively.

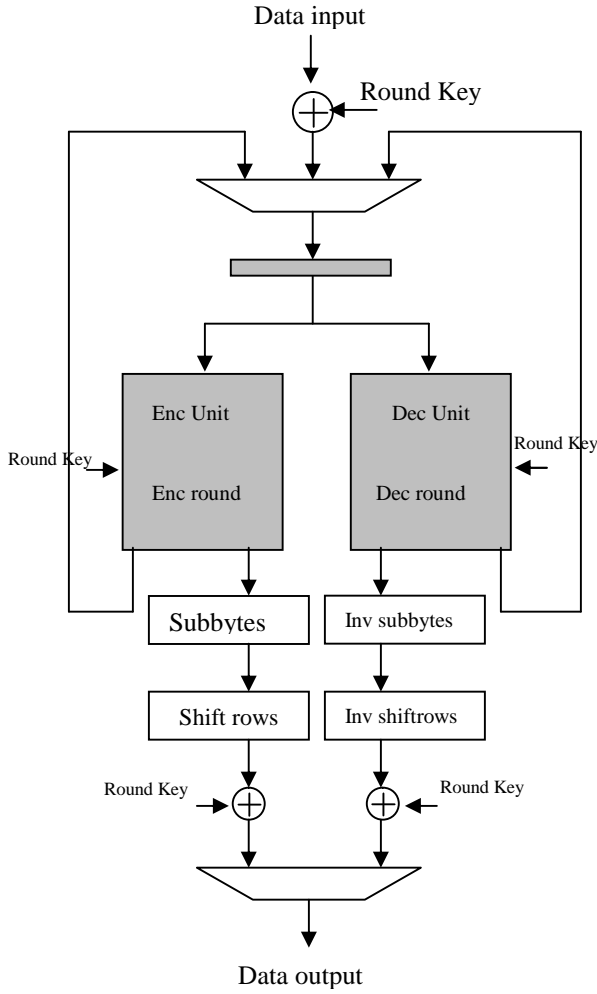
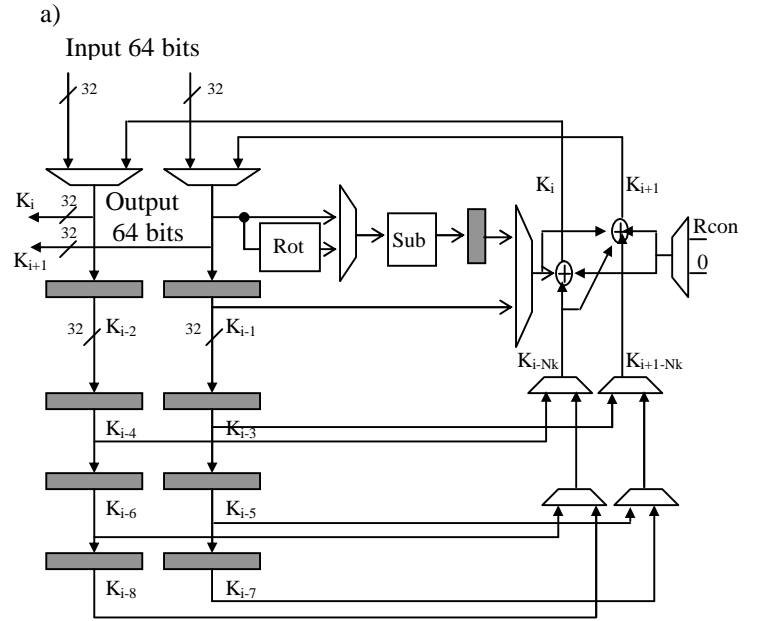


Fig 5: T-Box basic iterative architecture

V. KEY SCHEDULING

An AES key scheduling unit can either generate round keys on-the fly or it can store them in an internal key memory during the key setup phase, and then read them from this memory whenever they are required by the encryption /decryption unit.

The key scheduling unit [1] supports all three key sizes described in the standard, i.e., 128, 192, and 256 bit keys, this unit is referred to as 3-in-1 design. It requires a key setup phase, during which the round keys are computed and stored in internal memory. This unit produces 64-bit (a half of a round key) per clock cycle. The operation of the circuit is described by formulae given in Fig.6 that follow the pseudocode. The unit is capable of computing two 32-bit words of the key (K_i and K_{i+1}) per clock cycle, independent of the size of the main key.



b)
for $i \bmod Nk = 0$

$$K_i = K_{i-Nk} \oplus \text{Sub}(\text{Rot}(K_{i-1})) \oplus \text{Rcon}_{i/Nk}$$

$$K_{i+1} = K_{i+1-Nk} \oplus K_{i-Nk} \oplus \text{Sub}(\text{Rot}(K_{i-1})) \oplus \text{Rcon}_{i/Nk}$$

for $(Nk=8)$ and $(i \bmod Nk) \neq 0$

$$K_i = K_{i-Nk} \oplus \text{Sub}(K_{i-1})$$

$$K_{i+1} = K_{i+1-Nk} \oplus K_{i-Nk} \oplus \text{Sub}(K_{i-1})$$

otherwise

$$K_i = K_{i-Nk} \oplus K_{i-1}$$

$$K_{i+1} = K_{i+1-Nk} \oplus K_{i-Nk} \oplus K_{i-1}$$

Ref.[1], Figure 6. The 3-in-1 key scheduling unit of AES

a) Main circuit b) Formulae describing the operation of the circuit.

Since each round key is 128 bits long (the size of the input block), two clock cycles are required to calculate each round key. Therefore, this key scheduling unit is not designed for computing sub keys on the fly. Instead, all round keys corresponding to the new main key are computed in advance and stored in the key memory. This computation can be performed in parallel while encrypting data using previous main key, therefore key scheduling does not impose any performance penalty.

VI. TEST VECTORS AND DESIGN VERIFICATION

For testing the designs, test vectors provided by NIST in the fips 197 publication [3] have been used. The publication contains the intermediate state values, which is very helpful in the process of debugging.

Aldec Active HDL 7.2 has been used for the functional simulation. All the designs have been verified and implemented using the Xilinx ISE 10.1 Foundation.

VII. INTERFACE

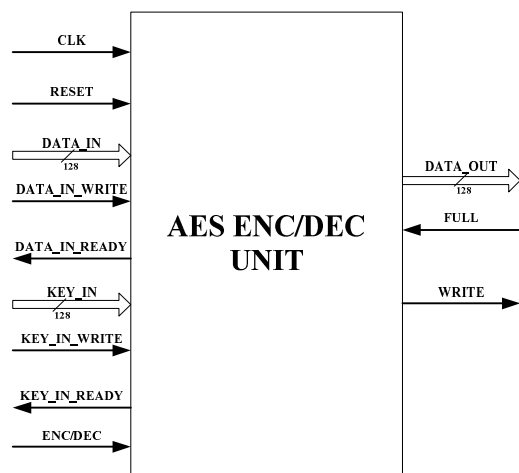


Fig7. Interface for implementations in Virtex5

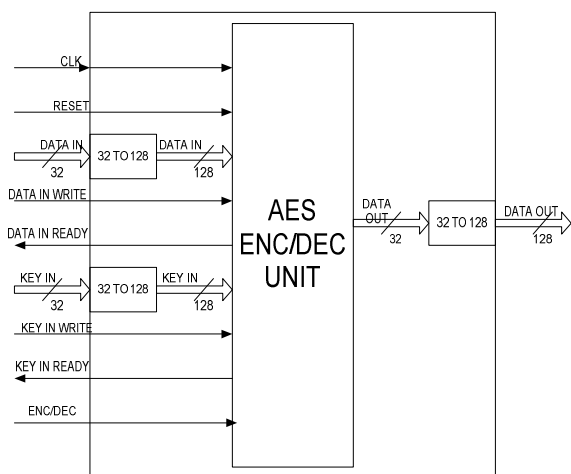


Fig 8. Interface for implementations in Spartan 3E

The interfaces shown in figure 7 and 8 are for the AES implementations with a 128 bit key

VIII. RESULTS OF IMPLEMENTATION IN XILINX VIRTEX 5 FPGA

All the AES architectures have been mapped into Xilinx FPGA device xc5vlx30-3ff676 with speed grade -3.

S-box results

The results of mapping, for the key sizes of 128bits, 192 bits and 256 bits based on the 8x8bit S-boxes are summarized in the tables below:

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 54.624 | 2343.292 | 264 | 8876.107 |
| 192 | 63.728 | 2008.536 | 264 | 7439.023 |
| 256 | 65.344 | 1958.863 | 264 | 7448.152 |

Table 1: Encryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 58.944 | 2171.552 | 331 | 6560.582 |
| 192 | 68.768 | 1861.33 | 331 | 5623.356 |
| 256 | 78.592 | 1628.664 | 331 | 4920.436 |

Table 2: Decryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 83.7 | 1529.271 | 633 | 2415.91 |
| 192 | 94.906 | 1348.702 | 641 | 2104.06 |
| 256 | 127.12 | 1006.922 | 622 | 1618.846 |

Table 3: Encryption and decryption

The tables 1, 2, 3 show the results of S-box implementations with encryption only, decryption only, and for both encryption and decryption in one unit.

When compared to the encryption only design, the decryption unit has a very low throughput and a low throughput to area ratio. The reason for low throughput is the inverse mix column operation that requires almost 80% more resources when compared to that of the mix column operation. Hence this additional combinatorial logic adds to the critical path hence increasing the latency and decreasing the throughput.

The combined encryption-decryption unit when compared to the other implementations of the S-box units has the lowest throughput to area ratio.

Compared to the results obtained by [2] on the high performance Altera Apex FPGA, the throughputs of the encryption, decryption and the combined units implemented in Xilinx are approximately higher by a factor of 2.43, 1.84, and 1.58 respectively.

T-Box results

Following are the results for the 8x32 bit T-boxes shown in the tables below:

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 61.044 | 2,096.848 | 762 | 2,751.769 |
| 192 | 71.204 | 1,797.651 | 767 | 2,343.744 |
| 256 | 83.712 | 1,529.051 | 760 | 2,011.911 |

Table 4: Encryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 79.416 | 1,611.765 | 931 | 1,731.220 |
| 192 | 92.652 | 1,381.513 | 931 | 1,483.903 |
| 256 | 105.888 | 1,208.824 | 931 | 1,298.415 |

Table 5: Decryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 108.888 | 1,175.519 | 1696 | 693.113 |
| 192 | 125.44 | 1,020.408 | 1693 | 602.722 |
| 256 | 141.104 | 907.132 | 1686 | 538.038 |

Table 6: Encryption and decryption

The tables 4, 5, 6 show the results of T-box implementations for encryption only, decryption only and the combined encryption and decryption unit.

Similar to the S-box implementation results, the encryption unit has the highest throughput and throughput to area ratio where as the combined implementation has the lowest throughput and the throughput to area ratio and the decryption unit is in between.

IX. RESULTS OF IMPLEMENTATION ON XILINX SPARTAN 3E FPGA

The basic interface has been modified to implement both architectures in the low cost Xilinx Spartan 3E FPGA. The ports for key input, data input and the data output are all 32 bits wide.

The target devices for the AES designs have been chosen based on the smallest device a design fits into. The table below describes the target devices used for each design.

| Implementation | Target Device |
|------------------|------------------|
| S-box Encryption | xc3s250e-5cp132 |
| S-box Decryption | xc3s250e-5cp132 |
| S-box Combined | xc3s500e-5cp132 |
| T-box Encryption | xc3s500e-5cp132 |
| T-box Decryption | xc3s1200e-5ft256 |
| T-box Combined | xc3s1600e-5fg320 |

Table 7: Target device

S-box results

The results of mapping, for the key sizes of 128bits, 192 bits and 256 bits based on the 8x8bit S-boxes for implementation in Spartan3E are summarized in the tables below:

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 194.18 | 659.182 | 1329 | 495.998 |
| 192 | 211.464 | 605.303 | 1349 | 448.705 |
| 256 | 227.28 | 563.181 | 1349 | 417.481 |

Table 8: Encryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 225.62 | 567.325 | 1505 | 376.960 |
| 192 | 239.976 | 533.386 | 1504 | 354.645 |
| 256 | 267.6 | 478.325 | 1505 | 317.824 |

Table 9: Decryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 300.56 | 425.871 | 3019 | 141.063 |
| 192 | 317.526 | 403.116 | 2913 | 138.385 |
| 256 | 360.672 | 354.893 | 3019 | 117.553 |

Table 10: Combined encryption and decryption

The tables 8, 9, 10 show the implementation results for the S-box designs of encryption, decryption, and combined encryption and decryption unit.

In terms of throughput to area ratio, the encryption has the best ratio, and the combined unit has the lowest ratio. The decryption unit has a ratio higher than the combined unit and lower than that of the encryption unit.

When compared to the encryption unit, the decryption unit requires a larger area and hence a lower throughput due to a larger latency. This increase in area is due to the inverse mix columns function that requires more computations compared to that of the mix columns operation.

T-Box results

Following are the results for the 8x32 bit T-boxes shown in the tables below:

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 242.04 | 528.838 | 4416 | 119.755 |
| 192 | 266.244 | 480.762 | 4416 | 108.868 |
| 256 | 285.888 | 447.727 | 4416 | 101.387 |

Table 11: Encryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 299.64 | 427.179 | 7798 | 54.780 |
| 192 | 313.94 | 407.721 | 7795 | 52.305 |
| 256 | 342.48 | 373.744 | 7795 | 47.946 |

Table 12: Decryption only

| Key size | Latency (ns) | Throughput (Mbps) | Area (slices) | Throughput /Area $\times 10^3$ |
|----------|--------------|-------------------|---------------|--------------------------------|
| 128 | 340.62 | 375.785 | 11,687 | 32.154 |
| 192 | 379.016 | 337.716 | 11,687 | 28.896 |
| 256 | 401.664 | 318.674 | 11,687 | 27.267 |

Table 13: Combined encryption and decryption

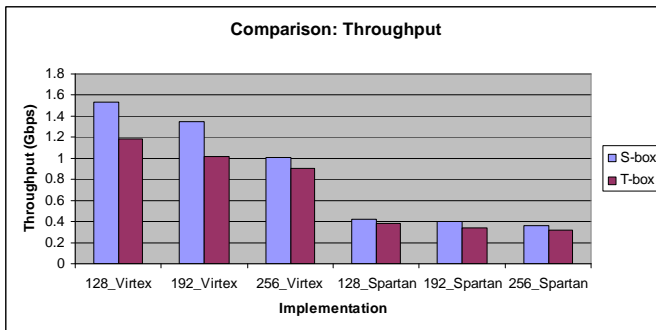
The tables 11, 12, 13 show the results of T-box implementations for encryption, decryption, and the combined encryption and decryption unit.

In terms of throughput to area ratio, the encryption has the best ratio, and the combined unit has the lowest ratio. The decryption unit has a ratio higher than the combined unit and lower than that of the encryption unit.

The reason for the decryption unit to have a larger area and a lower throughput is due to the presence of the inverse mix columns function operated on the keys for rounds 1 to Nr-1. Since this operation takes place on the fly, it adds to the area and the delay of the decryption unit.

X. COMPARISON OF THE S-BOX AND THE T-BOX IMPLEMENTATION RESULTS

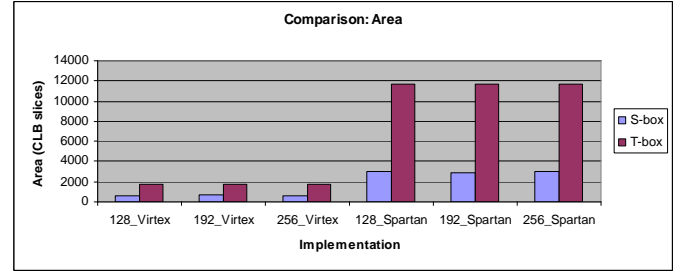
Throughput



Graph 1: S-box and T-box throughputs for the combined encryption and decryption units for different key sizes

Graph 1 shows the throughputs obtained for implementation of both the architectures of the combined encryption and decryption unit with three different key sizes. Again, the S-box encryption implementation has the better performance in both the devices in terms of throughput.

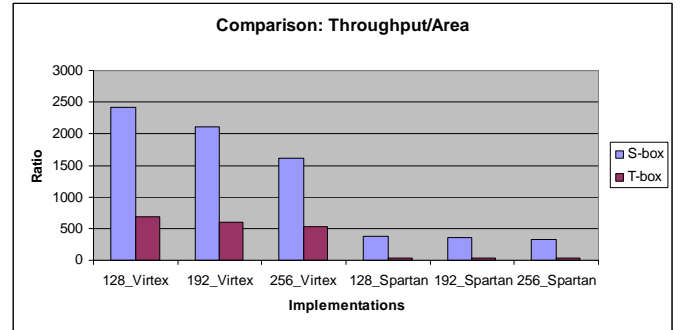
Area



Graph 2: Area for different implementations and key sizes

According graph 2, the implementation with the largest area is the T-box combined encryption and decryption circuit implemented in the Xilinx Spartan 3E device. The S-box implementation in the Virtex5 device has the smallest area

Throughput vs Area



Graph 3: Throughput vs Area ratio for different implementations and key sizes

Graph 3 shows that the best throughput to area ratio is obtained for the S-box architecture with a 128 bit key input.

Compared with the results of the T-box implementations, the S-box implementations have better performance in terms of area, latency and the throughput to area ratio.

The Xilinx ISE 10.1 tool implements the ROM used for the T-box look up tables in the LUTs and has not implemented any of the tables in the BRAMs. This resulted in larger area and latency for the T-box implementations compared to the corresponding S-box implementations with identical key sizes, hence dropping the throughput to area ratio very low.

Although the earlier implementations [2] of the T-box architecture have larger area, they have very low latency compared to the corresponding S-box implementations.

XI. CONCLUSION

The two different architectures implemented have different performance characteristics. The S-box implementations outperform the T-box implementation, giving excellent performance in terms of high throughput and low area.

According to the results obtained, our results do not show improvement in performance for T-box implementation since the T tables for encryption and decryption are implemented as logic and not as BRAMs. By ensuring that the tables are stored in BRAMs, the desired performance can be achieved with the T-box implementations.

The performance of these architectures can further be improved by pipelining the existing designs and converting the S-box look up tables to pure combinational logic.

REFERENCES

- [1] FPGA and ASIC Implementations of AES, by Kris Gaj and Pawel Chodowiec (to be published as a chapter of a textbook on cryptographic Engineering.)
- [2] Viktor Fischer and Milos Drutarovsky, "Two methods of Rijndael Implementation in reconfigurable hardware," CHES 2001.
- [3] FIPS 197: Advanced encryption standard. National Institute of Standards and Technology, 2001 available at <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [4] J. Daemen and V. Rijmen. AES proposal: Rijndael. Technical Report, 1999, available at <http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf>.
- [5] J. Daemen and V. Rijmen. The design of Rijndael: AES - The Advanced Encryption Standard. Number ISBN 3-540-42580-2. Springer-Verlag, 2002.