High Speed AES Cipher Engine

Ms.Anuradha Balasubramaniam

Abstract— For secure data transmission cryptographic algorithms are used for many applications. This paper introduces optimized hardware implementation of area and speed improvement for the block cipher Advanced Encryption Standard (AES-128) using Field Programmable Graphic Array (FPGA). As AES has four transformations among them sub-byte and mix-column transformation are key challenges to implement in terms of area and speed. In this research proposes new method of mix-column transformation which uses logical shift and Xor operation. This hardware implementation achieves the maximum clock frequency of 188.893 MHz, is in feedback encryption modes and uses less number of slices 427.

Index Terms— AES, Cipher, Mixcolumn, Verilog.

I. INTRODUCTION

The Cryptographic algorithms are used for security services in various environments in which less number of hardware resources used (i.e. low area), high speed and Power consumption are key requirements. Wireless Local Area Networks (WLAN), Wireless Personal Area Networks (WPAN), Wireless Sensor Networks (WSN), and smart cards are examples of such technologies. In 1997, The Effort was initiated towards developing a new encryption standard, called Advanced Encryption Standard (AES) by the National Institute of Standards and Technology (NIST)[1,2].Data confidentiality or authentication services can be provided by the Advanced Encryption Standard. Data confidentiality provides protection of data from being disclosed by unauthorized parties. Data authentication means that received data has not been affected by modification, insertion or deletion, replayed and with senders authentication guaranteed. AES has many advantages. It gives protection against various attacks. Implementation of hardware has high speed, can provide more physical security and achieve higher data rate for fast applications such as routers as compared to software implementation.

The efficiency of hardware implementations of the AES candidates has been one of the major criteria used by NIST to select the new federal standard from among five final candidates. Among five finalists Rijndael was the cipher chosen for Advanced Encryption Standard. Two scientists already designed this algorithm. [3,4].In October 2000, Rijndael was accepted as AES to achieve security and privacy while processing, transferring and storing the data. Field Programmable Gate Arrays (FPGAs) are an ideal platform for performing hardware acceleration of computationally intensive operations. AES is a symmetric block cipher, one key is used in symmetric algorithms, and same key is needed for sender and receiver. A cryptographic key and algorithm are applied to a block of data at once as a group rather than to one bit at a time in block cipher Method. AES block cipher specifies two functions: encryption that generates cipher text and decryption that produces plain text. AES works on block length of 128 bits and a variable key length of 128,192 and 256 bits. The basic unit of processing in the AES algorithm is a byte.4x4 array of bytes is called state

II. LITERATURE REVIEW

The researcher proposes numerous FPGA [5] [6] and Application Specific Integrated Circuit (ASIC) [7] [8] proposals for AES. These references gives information about area efficient implementation of the AES algorithm using S box (byte substitution phase) optimizations. [9] one of the researcher of the Rijndael algorithm implemented sub byte transformation using look-up tables storing 265 byte values of s-box are stored in a ROM lookup table. Another 256 byte lookup table is required to store the inverse s-box. For FPGA implementations, this is usually a convenient option since block ROMs are already available on almost all FPGA chips. Boolean functions optimization (BFO) is second method presented. In this method, the s-box (similarly, the inverse s-box) coordinate functions are treated as 8-bit Boolean functions and they can be jointly optimized. They used the freely available Minilog program that utilizes the Espresso heuristic logic minimizer to obtain an efficient two level Boolean function implementation of the Sub Bytes and Inv-Sub-Bytes operations and third using Galois field (GF) evaluation of the multiplicative inverse. Article of author presented his work on area of AES, Using the time sharing of certain resources and iteration architecture the area optimized design is based. The sub round key is produced by 10 times iteration round the optimized single key schedule module which is designed by one S-box ROM and one inverse mix column.

For encryption and decryption mode the cryptography data is obtained by 10 times iteration round the optimized single round module which is designed by two S-boxes ROM and two mix column. This is obtained by design optimized logic circuit, timing control blocks, and state machine[10].

III. IMPLEMENTATION OF AES

A. Encryption Process:
At the initial stage of encryption and decryption process the data is mapped to the state array like (4x4) bytes. The encryption round of AES requires realization of main four functional blocks such as add round key, Sub bytes, Shift rows and mix column while the decryption round requires four inverse functional blocks such as add round key, Inv Sub bytes, Inv Shift rows and Inv mix column, which are described in the following: Basic architecture of AES is shown in figure 1
i) Add round key
ii) Sub bytes
iii) Shift rows
iv) Mix column
i) Add round key :
The add-round key takes (4x4) state byte data from encryption
process and XOR’s with sub-keys which are generated from
key expansion unit and after tenth round encrypted data will
be obtained for 128 bit data and 128 bit key. The output of add
round key is given to sub-bytes.

ii) Sub-bytes:
In this paper the sub byte transformation is implemented
using look up table based approach; storing 265 byte values of
s-box are stored in a ROM lookup table. In FPGA
implementations, this is suitable option since block ROMs are
already available on almost all FPGA chips.

iii) Shift Rows:
This operation cyclically rotates over the one byte towards left
side for second row, two bytes for third row, three bytes for
fourth row and keeps the first row is unchanged. Proposed
hardware implementation of the shift rows operation uses
combinational logic circuits which requires simply 16 OR
gates to shift 32 bits of given data whereas other
implementation uses parallel shifters and multiplexers which
are much complicated. Using multiplexer Shift rows can be
implemented by using multiplexer by the simple enable
function of registers and five 2-to-1 multiplexors of eight bits
length [14].

iv) Mix column:
This operation is based on matrix multiplication of irreducible
polynomial over GF (2^8). AES
uses a characteristic 2 finite field with 8 terms which can also
be called as Galois field GF(2^8).
The AES algorithm irreducible polynomial is given in
equation 1
\[ M(x) = x^8 + x^4 + x^3 + x + 1 \] (1)

In the polynomial representation, multiplication in GF (2^8)
is denoted by (\(*\)) which corresponds to multiplication of
polynomial of degree 8. For mix-column the multiplication
matrix is given as
\[
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix} *
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} =
\begin{bmatrix}
r_0 \\
r_1 \\
r_2 \\
r_3 \\
\end{bmatrix}
\] (2)

We try to multiply the rows with the column. Let’s take the
first, second, third and fourth row’s of the matrix and multiply
them with a’s values.
The formula goes like this
\[ R_0 = (02.d4) \{03.bf\} \{01.5d\} \{01.30\} \] (3)
\[ R_1 = (01.d4) \{02.bf\} \{03.5d\} \{01.30\} \] (4)
\[ R_2 = (01.d4) \{01.bf\} \{02.5d\} \{03.30\} \] (5)
\[ R_3 = (03.d4) \{01.bf\} \{01.5d\} \{02.30\} \] (6)

We have implemented above four rows. The key role of
multiplication by (02) and (03) is important as (01) is
basically with 1(in decimal) which ends with the original
values. No special method is required to calculate them. Here
we do need to convert them to binary form. The multiplication
of a value by x (i.e by 02) can be obtained by converting it into
binary then 1-bit left shift followed by a conditional bitwise
xor with (00011011) if the leftmost bit of the original value
(before the shift) is 1and pad on with 0's Here the conditional
bit wise xor operation used by us is x"1B" (i.e.00011011)
because the calculation GF (2m) for higher order is calculated
with the help of following equation
\[ M(x) = x^8 + x^4 + x^3 + x + 1 \] (7)
And equation (7) can be represented in binary as (00011011).
Now for row r0 of equation (3) actual results of mathematical
function are
1) \{02\}.{d4}:
First convert d4 to binary as d4=11010100; then \{d4\}.{02} =
1101 0100 << 1 (</code><span class="highlight">SS</span> on with 0's is left shift,1 is the
number of shift done)
\[ = 1010 1000 XOR 0001 1011 ( the leftmost is a 1
before shift is behind the logic) \]
\[ =10110011(ANS). \] (8)
As the irreducible polynomial is
\[ M(x) = x^8 + x^4 + x^3 + x + 1 \]
Now the multiplication by (03) can be done with the help of
(02) as suggested in book [15]. We split 03 up in its binary
form as
\[ 03 = 11 \]
\[ = 10 XOR 01 \]
2) \{03\}.{bf}:
\[ \{03\}.{bf} = \{10 XOR 01\} \{1011 1111\} \]
\[ = \{1011 1111.10\} XOR \{1011 1111, 01\} \]
\[ = \{1011 1111.10\} XOR \{1011 1111\} \]
Because \{10111111\}x1[in decimal] =1011 1111
\[ =01111110\] XOR \{00011011\} XOR \{1011 1111\}
\[ =1011010(ANS). \] (9)
3) \{01.5d\}
4) \{01.30\}
Basically multiplying with 1(in decimal) to here (5d and
30) where we end up with the original values.
Only convert it into binary form as 5d=01011101 (10)
30=000110000 (11)
Lastly equation (8),(9),(10) and (11) are added together.
Addition will be using XOR, as they are in binary form
\[ r0 = \{02.d4\} + \{03.bf\} + \{01.5d\} + \{01.30\} \]
\[ = (10110011)XOR(11011010)XOR(01011101) \]
\[ XOR\ (0011\ 0000) \]
\[ = 0000 0100 = 04 \] (in Hex)
Similarly we can calculate and implement remaining three
rows. Hardware architecture for mix-column is shown in
figure.2. To implement above four rows for Mix-column
transformation Multiplication we require logical design of
shift and XOR logical gates. Same method can be
implemented for inverse mix-column transformation by
multiplying output of shift-rows column with fixed
multiplication matrix of Mix-column where we can get first
column of mix-column output. This method is more reliable
and consumes less hardware as compare to Galois field
multiplication and adds shift method and many other
methods.

B. Key Expansion Unit
The key expansion unit is responsible for generating keys
for different rounds, which are called round keys (sub keys).
The key expansion unit consists of
1) R-con
2) Rotate word
3) Sub-bytes

The sub-keys used for encryption and decryption processes for number of rounds are pre-calculated and stored in a memory block in this paper.

The process of key generation for first round is - initial key-(a) the last column of initial key is rotated clockwise one step and passed through the S-box, (b) Result of (a) is XORed with first column of initial key and obtained result will be XORed with R-con for key expansion which generates the first column of first round key. This result will be XORed with second column of initial key which results second column of first round key, the same result again XORed with third column of initial key which gives third column of first round key. And the same result XORed with last round of initial key which result the last column of first round key. AES Encryption is shown in figure 3. This process is repeated for remaining columns’ to generate columns’ of the first round key. First round key is now treated as initial key and second round key is generated. This process continues till the generation of the tenth key. AES encryption process continues from first round to nine rounds and last round is without Mix-column transformation.

IV. IMPLEMENTATION RESULTS

As the proposed hardware implementation of AES was simulated and synthesized with Xilinx synthesis technology and implemented on Spartan 3 XC3S1500L-4FG676. As per the test data given by NIST AES implementation is verified. The implementation results of AES shown in figure 4. Table 1 and Table 2 represents the hardware implementation results for AES. Cyclic shift method is a simple reliable and less hardware approach which improves overall performance in terms of area and speed of AES. As AES cryptographic algorithm is used for lower end applications.

<table>
<thead>
<tr>
<th>Designs</th>
<th>Clk Frequency (MHz)</th>
<th>Area (Slices)</th>
<th>No.of Slices Flip Flops</th>
<th>No. of 4 Input LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing AES Encryption</td>
<td>100.099</td>
<td>540</td>
<td>132</td>
<td>1052</td>
</tr>
<tr>
<td>Proposed AES Encryption</td>
<td>188.893</td>
<td>427</td>
<td>402</td>
<td>798</td>
</tr>
</tbody>
</table>
TABLE 2

<table>
<thead>
<tr>
<th>Designs</th>
<th>Minimum input arrival time before clock</th>
<th>Maximum output required time after clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing AES Encryption</td>
<td>13.546ns</td>
<td>7.488ns</td>
</tr>
<tr>
<td>Proposed AES Encryption</td>
<td>7.585ns</td>
<td>7.165ns</td>
</tr>
</tbody>
</table>

V. SIMULATION RESULTS

Fig 4: Output Results of Implementation of AES

VI. CONCLUSION

The implementation of AES 128 bit using mix-column transformation is presented in this paper. Among shift-add method and Galois Field Method for Mix-column transformation this proposed cyclic shift method is a simple reliable and less hardware approach which improves overall performance in terms of area and speed of AES. As AES cryptographic algorithm is used for lower end applications and different embedded applications.

REFERENCES