Highly Area efficient AES encryption design with a new approach

Ashish Kumar Agrawal\textsuperscript{1}, Santosh Yadav\textsuperscript{2} and Kanak Kumar\textsuperscript{3}

\textsuperscript{1}\textit{M. Tech Scholar, in ECE, at Millennium Institute of Technology, Bhopal, (M.P)}

\textsuperscript{2,3}\textit{Professor, in ECE, at Millennium Institute of Technology, Bhopal, (M.P)}

Abstract: Information security has become a very critical aspect of modern computing systems. With the global acceptance of internet, virtually every computer in the world today is connected to every other. While this has created tremendous productivity and unprecedented opportunities in the world we live in, it has also created new risks for the users of these computers. The users, businesses and organizations worldwide have to live with a constant threat from hackers and attackers, who use a variety of techniques and tools in order to break into computer systems, steal information, change data and cause havoc. The paper work aims at designing and implementing a secure data communication between any two users based on the realization of advanced Symmetric-key Cryptographic algorithm called Advanced Encryption Standard (AES) on an FPGA based processor.

Introduction: Cryptographic technology is an important way to ensure information Security and is the key to information safety. Among all kinds of cryptographic algorithms, Advanced Encryption Standard Algorithm (AES) is preferred as it offers high security, efficiency, convenient usage, flexibility, and comprehensive performance. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt(decipher) information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192 and 256 bits to encrypt and decrypt data in the blocks of bits. AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext.

Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to AES is based on a design principle known as a Substitution permutation network. Unlike its predecessor, DES, AES does not use a Feistel network. AES operates on a 4 x 4 array of bytes called state in a matrix form. The algorithm consists of performing four different simple operations. These operations are: Sub Bytes, Shift Rows, Mix Columns and Add Round Key.

Design Method: AES operates on a 4x4 array of bytes (referred to as “state”). The algorithm consists of performing 4 different types of the operations.
SubBytes transformation: is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box).

<table>
<thead>
<tr>
<th>S-Box: Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>c</td>
</tr>
</tbody>
</table>

ShiftRows Transformation: The first row, $r = 0$, is not shifted. The shift value shift ($r$, Nb) depends on the row number, $r$, as follows (recall that Nb= 4):

- shift(1,4) = 1
- shift(2,4) = 2
- shift(3,4) = 3

MixColumns Transformation: The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial. In the MixColumns step, each column of the state is multiplied with a fixed polynomial $a(x)$.

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes.

Columns are considered with fixed polynomial $s(x) = a(x) \otimes s(x)$: polynomial

\[
\begin{bmatrix}
  s_{0,c} \\
  s_{1,c} \\
  s_{2,c} \\
  s_{3,c}
\end{bmatrix} = \begin{bmatrix}
  02 & 03 & 01 & 01 \\
  01 & 02 & 03 & 01 \\
  01 & 01 & 02 & 03 \\
  03 & 01 & 01 & 02
\end{bmatrix} \begin{bmatrix}
  s_{0,c} \\
  s_{1,c} \\
  s_{2,c} \\
  s_{3,c}
\end{bmatrix} \quad \text{Let}
\]

AddRoundKey Transformation: a Round Key is added to the output of MixColumn operation (state) by a simple bitwise XOR operation. For each round of operation, separate key is generated using Key Expansion.

Key Expansion: Round keys are derived from the cipher key using Rijndael's key schedule. The AES algorithm takes the Cipher Key, $K$, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb ($Nr + 1$) words. The expansion of the input key into the key schedule proceeds as per the functions Rotword(), Subword(), Rcon[i/Nk], Xor operations.
Figure 2: Proposed architecture S8 box

Tool Platform and Language used:
Tool: Xilinx ISE: It is a software tool produced by Xilinx for synthesis and analysis of HDL designs. Language used: Verilog HDL: Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction.

Platform Used: family- Vertex4, Device-XC4VLX80, Package-FF1148. Target FPGA is a Vertex FPGA because the same platform is been used by base papers.

Simulation and synthesize of Proposed work

Figure 3: Top Model

www.ijsret.org
Results: From the simulation as shown in above slides

Key: A234567ba234a234a234567ba234a234

Result: -1
Output: Cde5017b64cd7e93
Input: A234567ba234a234a234
Output^Input: 6fd15700c6f9dca7
Avalanche: 41 bit change/64 bit

Result: -2
Output: Df5ab6daed24e9c5
Input: A234567ba234a234a234
Output^Input: 7d6e14eebb5f4bf1
Avalanche: 45 bit change/64 bit

Table 3: Results for each module
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S-box 7 (S7)</td>
<td>S-box 9 (S9)</td>
<td>S-box 7 (S7)</td>
</tr>
<tr>
<td>No. of slice</td>
<td>34</td>
<td>169</td>
<td>-</td>
</tr>
<tr>
<td>Logical Time</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Overall KASUMI</td>
<td>No. of slice</td>
<td>8784</td>
<td>8770</td>
</tr>
<tr>
<td>Logical Time</td>
<td>34.01</td>
<td>-</td>
<td>33.64</td>
</tr>
</tbody>
</table>

Table 4: comparative results

**Conclusion:** The work is implemented of FPGA which makes proposed work a semicustom design as known semicustom design always lack behinds compare to full-custom design in term of Area, speed and power. In future proposed work can be implemented at transistor level (i.e. Full-custom)

**References**


[2]. Sima I., Tarmurean D., Greu V, Diaconu A.‘XXTEA, an alternative replacement of KASUMI cipher algorithm in A5/3 GSM and f8, f9 UMTS data security functions’ 9th International Conference on Communications (COMM), volume 1, pp 328-333

