ABSTRACT: Wired and wireless communications requires secure transaction of critical data maintaining proper validation, authentication and confidentiality. The Advanced Encryption Standard (AES) is a Federal Information Processing Standard (FIPS), which is a cryptographic algorithm that is used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. AES works over a finite field. Finite field multiplication is the main operation for most of the cryptographic applications. Redundant representation allows multiplication and squaring to be achieved more simply than other representations. The main idea in the redundant representation is to represent the finite Galois field. In this project, a modified version of redundant representation which works for any finite fields to design arithmetic circuits with small complexity has been studied. The representation of the mix column of the AES algorithm is modified using the modified redundant representation. The computations of the gates used to design the arithmetic circuits of the mix column of the AES algorithm are determined. The computation results are verified with the theoretical computations. Here Xilinx 14.1 version software is used for simulation and optimization of the synthesizable VHDL code of the AES algorithm.

1. INTRODUCTION
Efficient hardware implementations of the arithmetic operations in the finite field have been widely studied in coding theory, computer algebra, and cryptographic applications. Multiplication of the field elements is the most important arithmetic operation since it is an essential building block in applications. The measure of efficiency in hardware implementations is the number of AND gates and XOR gates. The representation of field elements has great impact on the performance of the finite-field arithmetic. Redundant representation allows multiplication and squaring to be achieved more simply than other representations. In cryptographic applications, finite fields, represented by low weight polynomials, are desired due to the efficiency of finite-field operations.

The main idea in the redundant representation is to represent the finite field GF[2^m] a subring of the quotient polynomial ring GF[2^n/x^n-1], provided that n > m and there is a subring of GF[2^n/x^n-1] which is isomorphic to GF[2^m] as a ring. This representation allows a kind of parallel multiplier design yielding small complexity for multiplication in GF[2^m] for some m. For a better complexity, it is necessary to choose the smallest n satisfying the above condition.

1.1 THE PURPOSE OF CRYPTOGRAPHY
Cryptography is the science of writing in secret code and is an ancient art. The new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet. Within the context of any application-to-application communication, there are some specific security requirements.

1.2 AES ALGORITHM
The redundant representations are represented in the AES algorithm to reduce the number of gates used in the process of encryption and decryption. The National Institute of Standards and Technology (NIST), solicited proposals for the Advanced Encryption Standard (AES). The AES is a Federal Information Processing Standard (FIPS), which is a cryptographic algorithm that is used to protect electronic data. 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 blocks of 128 bits. The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three
stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm.

In AES algorithm the encryption and decryption process has the following stages to encipher or decipher the given plain text, Encryption process:
1. Byte Substitution.
2. Shift rows
3. Mix column
4. Add Round Key
Decryption process:
1. Inverse Byte Substitution
2. Inverse Shift rows
3. Inverse Mix Column
4. Key Generation

The AES algorithm works over finite field. Finite field multiplication is the main operation for most of the cryptographic applications. The AES algorithm has the finite field multiplication over the finite field GF(2^8) in the Mix column and Inverse mix column. The measure of efficiency in hardware implementations is the number of AND gates and XOR gates.

In the MixColumns transformation, we need to implement constant multiplication of \{02\} and \{03\} in GF(2^8). Assuming \(X\) is a byte in the State, \(\{02\}X\) can be implemented by shifting and bit-wise XOR operations, and \(\{03\}X\) can be computed by \(\{(02)X\}\) where \(X\) denotes Exclusive-OR. If \(X\) is expressed in binary form as \({x_7}, {x_6}, {x_5}, {x_4}, {x_3}, {x_2}, {x_1}, {x_0}\), \(\{02\}X\) can be calculated by \(\{(02)X\} = \{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\}\) \(\{02\} = 0, 0, 0, x_7, x_0, 0, 7, x_7\). Since \(x_i = x_i\), only needs 4 XOR gates to implement the MixColumns transformation for one column of the State. The InvMixColumns transformation is more complicated. The number of gates used for the Mix column of AES algorithm occupies more space in the implementation process. Because of the large number of gates the calculation process becomes more complicated in the AES algorithm. Each round performs the Mix column process and it get increases until all the rounds are performed. Therefore the time complexity and space complexity increases in AES algorithm process.

**Pseudocode for mix column:**

```
MixColumn(byte state[4,Nc], Nc)
begin
    byte t[4] for c = 0 step 1 to Nc - 1
    for r = 0 step 1 to 3
        t[r] = state[r,c]
    end for
    for r = 0 step 1 to 3
        state[r,c] = FFmul(0x02, t[r]) xor
                      FFmul(0x03, t[(r + 1) mod 4]) xor
                      t[(r + 2) mod 4] xor t[(r + 3) mod 4]
    end for
end for
```

1.3. MIX COLUMN:
A substitution that makes use of arithmetic over GF(2^8). Mix column operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. MixColumns transformation is performed on the State column-by-column. Each column is considered as a four-term polynomial over GF(2^8) and multiplied by \(a(x)\) modulo \(x^4 + 1\).

This step replaces each byte of a column by a function of all the bytes in the same column.

More precisely, each byte in a column is replaced by two times that byte, plus three times the the next byte, plus the byte that comes next, plus the byte that follows. The words _next_ and _follow_ refer to bytes in the same column, and their meaning is circular, in the sense that the byte that is next to the one in the last row is the one in the first row. [By two times and three times, we mean multiplications in GF(2^8) by the bit patterns 00000010 and 00000011, respectively.]

In the MixColumns transformation, we need to implement constant multiplication of \{02\} and \{03\} in GF(2^8). Assuming \(X\) is a byte in the State, \(\{02\}X\) can be implemented by shifting and bit-wise XOR operations, and \(\{03\}X\) can be computed by \(\{(02)X\} X\) where \(X\) denotes Exclusive-OR. If \(X\) is expressed in binary form as \({x_7}, {x_6}, {x_5}, {x_4}, {x_3}, {x_2}, {x_1}, {x_0}\), \(\{02\}X\) can be calculated by \(\{(02)X\} = \{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\}\) \(\{02\} = 0, 0, 0, x_7, x_0, 0, 7, x_7\). Since \(x_i = x_i\), only needs 4 XOR gates to implement. The block diagram shows the straightforward way to calculate \(S0\), in the MixColumns transformation. Since \(\{01\}X = X\), instead of \(\{01\}X\) is used in this.

Calculations:
- Calculation of \(S'_{1,c}\), \(S'_{2,c}\), and \(S'_{3,c}\) can be done by connecting appropriate \(\{02\}X\), \(\{03\}X\), or \(X\) of \(S'_{1,c}\), \(S'_{2,c}\), and \(S'_{3,c}\) to the last row of XOR gates in the critical path has 4 XOR gates and a total of \(4 \times 8 + 4 \times 4 = 144\) 2-input XOR gates is needed to implement the MixColumns transformation for one column of the State. The InvMixColumns transformation is more complicated. The number of gates used for the Mix column of AES algorithm occupies more space in the implementation process. Because of the large number of gates the calculation process becomes more complicated in the AES algorithm. Each round performs the Mix column process and it get increases until all the rounds are performed. Therefore the time complexity and space complexity increases in AES algorithm process.

**Figure 1** Encryption Process
Inverse Mixing of Columns Transformation InvMixColumns() is the inverse of the MixColumns() transformation presented. InvMixColumns() operates on the State column-by-column, treating each column as a four term polynomial. The columns are considered as polynomials over GF (28) and multiplied modulo x4 + 1 with a fixed polynomial a-1(x), given by

\[ a^{-1}(x) = 0\{0\}x^3 + 0\{0\}x2 + 0\{0\}x + 0\{e\}. \]

2. METHODOLOGY

To decrease the number of gates in the mix column of the AES algorithm the redundant representation is used. The main idea in the redundant representation is to represent the finite field GF(2m) a subring of the quotient polynomial ring GF(2n)/(xn-1), provided that n > m and there is a subring of GF(2m)/(x^n-1) which is isomorphic to GF(2m) as a ring. This representation allows a kind of parallel multiplier design yielding small complexity for multiplication in GF(2m) for some m. For a better complexity, it is necessary to choose the smallest n satisfying the above condition. Let p be an arbitrary prime number and m > 2 be a positive integer. Note that the finite field GF(p^n) can be considered as a ring as well. Let n > m be an integer and consider the finite quotient ring GF(p^n)/{(x^n-1)}. If there exists a subring of the quotient ring GF(p^n)/(x^n-1), which is isomorphic to GF(p^n) as a ring, then it is well known that we can represent GF(p^n) using the ring representation of the quotient ring GF(p^n)/(x^n-1). Such a representation of GF(p^n) is called a redundant representation or a polynomial ring representation.

2.1. MULTIPLICATION USING MODIFIED REDUNDANT REPRESENTATION AND COMPLEXITY

Let f(x) be an irreducible polynomial of degree k in GF(p^n)[x]. Multiplication in GF(p^n)[x]/(x^n-1) is computed as a multiplication of polynomials with modulo x^n-1 reduction. A simple and generic design for finite-field multiplication in GF(p^n) is the schoolbook method. Consider two k-term polynomials a(x)=Σaix^i \[\{i\} \leq k\] and b(x)=Σbjx^i \[\{i\} \leq k\]. By using the schoolbook multiplication method, one can compute C(x)=a(x)b(x) Assume that the reduction polynomial, f(x), is binomial. Then, x^i+j = x^i+j \[\{i\} \leq k, j \leq k\]

Then, for binomials, the schoolbook multiplication method can be performed at most k^2 multiplications and k(k-1) additions in GF(p). By using the same idea, if the reduction polynomial, f(x), is trinomial or pentanomial, then the number of additions is (k-1)(k+1) or (k-1)(k+3) respectively.

2.1.1 MULTIPLICATION USING MODIFIED METHOD

Obtain an arithmetic design for finite-field multiplication in GF(p^n) assuming an arithmetic design for finite-field multiplication in the intermediate field GF(p^k) in the case that p=2. The same method works for the general characteristic p. The multiplication in GF(p^n)/(x^k-1) can be performed with a linear feedback shift register with feedback polynomial (x^k-1). Here c(x)=a(x)b(x) c(x)=ΣΣ a(i) b(j) (mod p) x^i+j. One can determine the number of AND gates and XOR gates for the extension degree multiple of k by using the following formulas for binary fields. These are the upper bounds for the multiplication. AND or XOR refers to the number of multiplications or additions in GF(2) to perform a multiplication in GF(2^n), respectively. #AND=AND, n^2  
#XOR= XOR, n^2+k, n, (n-1)  

3. SOFTWARE TOOLS

VHDL (VHSIC hardware description language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits.

The manipulations are carried out in the VHDL language using Xilinx software. Xilinx ISE is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize (“compile”) their designs, perform timing analysis, examine RTL diagrams, simulate a design’s reaction to different stimuli, and configure the target device with the programmer.

4. OUTPUT FOR MIX COLOUMN AND MIX COLUMN USING REDUNDANT REPRESENTATION:

4.1. MIX COLUMN

![Image](image1.png)

4.2. MIX COLUMN USING REDUNDANT REPRESENTATION

![Image](image2.png)
5. DISCUSSION ABOUT THE PROPOSED WORK

The main idea in the redundant representation is to represent the finite field GF(2<sup>n</sup>) a subring of the quotient polynomial ring GF(2<sup>n</sup>)/<x=x<sup>2</sup>-1>, provided that n > m and there is a subring of GF(2<sup>m</sup>)/<x=x<sup>2</sup>-1> which is isomorphic to GF(2<sup>n</sup>) as a ring. This representation allows a kind of parallel multiplier design yielding small complexity for multiplication in GF(2<sup>n</sup>) for some m. For a better complexity, it is necessary to choose the smallest n satisfying the above condition. Let p be an arbitrary prime number and m > 2 be a positive integer. Note that the finite field GF(p<sup>m</sup>) can be considered as a ring as well. Let n > m be an integer and consider the finite quotient ring GF(p)[x]/<x=x<sup>2</sup>-1>. If there exists a subring of the quotient ring GF(p)[x]/<x=x<sup>2</sup>-1>, which is isomorphic to GF(p<sup>n</sup>) as a ring, then it is well known that we can represent GF(pm) using the ring representation of the quotient ring GF(p)[x]/<x=x<sup>2</sup>-1>. Such a representation of GF(p<sup>m</sup>) is called a redundant representation or a polynomial ring representation. Let f(x) be an irreducible polynomial of degree k in GF(p)[x] Multiplication in GF(pK)=GF(p)[x]/f(x) is computed as a multiplication of polynomials with modulo reduction. A simple and generic design for finite-field multiplication in GF(pK) is the schoolbook method. Consider two k-term polynomials a(x)=Σx<i><ai>, b(x)=Σx<i><bi>. By using the schoolbook multiplication method, one can compute C(x)=a(x).b(x) Assume that the reduction polynomial, f(x), is binominal. Then, X<sup|i</sup><sup>2</sup>=X<sup|i</sup> if i+j<k else X<sup|i+j</sup> if i+j>k Then, for binomials, the schoolbook multiplication method can be performed at most k<sup>2</sup> multiplications and k(k-1) additions in GF(p). By using the same idea, if the reduction polynomial,F(x) is trinominal or pentominal, then the number of additions Is (k-1)(k+1) or (k-1)(k+3) respectively.

obtain an arithmetic design for finite-field multiplication in GF(pK) assuming an arithmetic design for finite-field multiplication in the intermediate field GF(p)<sup>m</sup> in the case that p=2 The same method works for the general characteristic p. The multiplication in GF(p<sup>m</sup>)[x]/<x=x<sup>2</sup>-1> can be performed with a linear feedback shift register with feedback polynomial (x<sup>k</sup>-1). Here c(x)=a(x).b(x) c(x)≡Σ_{i=0}^{m-1} (b(0+i)(mod nk))x<sup>i</sup> One can determine the number of AND gates and XOR gates for the extension degree multiple of k by using the following formulas for binary fields. These are the upper bounds for the multiplication. AND or XOR refers to the number of multiplications or additions in GF(2) to perform a multiplication in GF(2<sup>n</sup>), respectively, #ANDs= AND<sup>n</sup>-1 #XOR= XOR<sup>n</sup>-1+k.n<sup>n</sup>-1

5.1. VALUES OF N

<table>
<thead>
<tr>
<th>N</th>
<th>AND</th>
<th>OR</th>
<th>%AND</th>
<th>%OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>3172</td>
<td>3660</td>
<td>70.7</td>
<td>64.5</td>
</tr>
<tr>
<td>67</td>
<td>4489</td>
<td>4422</td>
<td>52.8</td>
<td>41.2</td>
</tr>
</tbody>
</table>

6. CONCLUSION

Optimized and Synthesizable VHDL code is developed for the implementation of both encryption and decryption process. Each program is tested with some of the sample vectors provided by NIST and output results are perfect with minimal delay. Adding data pipelines and some parallel combinational logic in the key scheduler and round calculator can further optimize this design. The modified version of redundant representation which works for any finite fields to design arithmetic circuits with small complexity has been studied. The representation of the mix column of the AES algorithm is modified using the modified redundant representation. The computations of the gates used to design the arithmetic circuits of the mix column of the AES algorithm are determined. The computation results are verified with the theoretical computations. Here Xilinx 14.1 version software is used for simulation and optimization of the synthesizable VHDL code of the AES algorithm.

REFERENCES


