Low Complexity Digit Serial Multiplier for Finite Field using Redundant Basis

Jyothi Leonore Dake and Sudheer Kumar Terlapu

Department of ECE, Shri Vishnu Engineering College for Women (Autonomous), Bhimavaram −534202, Andhra Pradesh, India; dakeleonore@gmail.com, skterlapu@gmail.com

Abstract

Cryptography has been increasingly used due to its rapid trends. In recent days, it has been used mostly in communication and in financial transactions through automated machines or internet. The applications of cryptography and coding theory require finite field operations and are realized based on the finite field computations. In this paper efficient digit-serial multiplier over finite field is implemented and is obtained by using Redundant Basis (RB), intend to present high-throughput multiplier. Area and power are the two factors which obtain less when compared to the previous multipliers. The digit-serial multiplier for 32-bit is implemented using Verilog HDL and synthesized to know its better performance in terms of area and power compared to previous multipliers.

Keywords: Cryptography, Digit-Serial, Finite FIELD, Redundant Basis

1. Introduction

Many applications of cryptography are realized based on the finite field operations and which has vast use in recent days4. Finite field is which consists of finite number of fields and also known as Galois field, denoted as GF (p^n). Addition and multiplication are two operations of finite field mostly used and multiplication is commonly used for many applications based on field computations3. To obtain multipliers over finite fields, uses basis and they are standard basis which is also known as the polynomial (PB), normal (NB), dual (DB) and redundant (RB)10. The dual basis is which need conversion and normal basis include squaring, redundant basis is also as normal basis not only include squaring but also have modular reduction to obtain low-complexity multipliers6.

In this, a low-complexity multiplier for finite field by using redundant representation is implemented. From the Signal Flow Graph (SFG) the Processor Space-Flow Graph (PSFG) is obtained, from that the digit-serial/parallel multipliers are presented previously. By modifying that in order to consume low complexity the new multiplier is implemented to have good performance. The constitution of the paper is as follows: Existing RB digit-serial multipliers are presented in section 2. Proposed RB digit-serial multiplier is mentioned in section 3. Simulation results and comparison are mentioned in section 4. Conclusions are presented in section 5.

2. Existing RB Digit-Serial Multipliers

2.1 Mathematical description

Let A, B belongs to GF (2^n) can be demonstrated in the form of RB:

\[ A = \sum_{i=0}^{n-1} a_i x^i \]  
\[ B = \sum_{i=0}^{n-1} b_i x^i \]  

Thus \( a_i, b_i \) belongs to GF(2). Let A and B are inputs and product is C, is demonstrated as follows:
Low Complexity Digit Serial Multiplier for Finite Field using Redundant Basis

\[ C = A \cdot B = \sum_{i=0}^{n-1} (x^i b_i) \cdot A \] (3)

\[ = \sum_{i=0}^{n-1} \left( \sum_{j=0}^{n-1} b_{(i-j)h} a_j \right) x^i \] (4)

Where \((I - j)\) denotes modulo \(n\) reduction. Define

\[ C = \sum_{i=0}^{n-1} c_i x^i, \text{ where } c_i \in GF(2) \]

we have:

\[ c_i = \sum_{i=0}^{n-1} b_{(i-j)h} a_j \] (5)

Alternately, we can write (5) in a bit-level matrix-vector form:

\[
\begin{bmatrix}
  c_0 \\
  c_1 \\
  \vdots \\
  c_{n-1}
\end{bmatrix} =
\begin{bmatrix}
  b_0 & b_{n-1} & \cdots & b_1 & a_0 \\
  b_1 & b_0 & \cdots & b_2 & a_1 \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  b_{n-1} & b_{n-2} & \cdots & b_0 & a_{n-1}
\end{bmatrix}
\] (6)

From (6), shifted form of the input bits \(B\) can be defined as follows:

\[ B^0 = \sum_{i=0}^{n-1} b_0^i x^i = b_0 x \oplus b_1 x \oplus \cdots \oplus b_{n-1} x^{n-1} \] (7)

\[ \cdots \cdots \cdots \cdots B^{n-1} = \sum_{i=0}^{n-1} b_i^{n-1} x^i = b_1 + b_2 x + \cdots + b_0 x^{n-1} \] (8)

Where, \(b_0^{i+1} = b_{i+1}^{n-1} \quad b_j^{i+1} = b_{j-1}^{i+1} \), for \(1 \leq j \leq n - 2 \). (9)

The recursions on (9) can be extended further to have:

\[ b_j^{i+s} = \begin{cases} b_{n-s+j}^i, & \text{for } 0 \leq j \leq n - 2 \\ b_i^i, & \text{other wise} \end{cases} \] (10)

Where \(1 \leq s \leq n - 1\), Let \(Q\) and \(P\) are two integers alike \(n = QP + r\), where \(0 \leq r \leq P\). For ease, assume \(r = 0\), and decompose the input operand \(A\) into \(Q\) number of bit-vectors \(A_u\) for \(u = 0, 1, \cdots, Q-1\), as follows:

\[ A_0 = [a_0 a_Q \cdots a_{n-Q}] \] (11)

\[ A_{Q-1} = [a_{Q-1} a_{3Q-1} \cdots a_{n-1}] \] (12)

Identically, we can produce \(Q\) units of shifted vector operands \(B_u\) for \(u = 0, 1, \cdots, Q-1\), as follows:

\[ B_0 = [B_0^0 B_0^1 \cdots B_0^{n-1}] \] (13)

\[ B_{Q-1} = [B_{Q-1}^0 B_{Q-1}^1 \cdots B_{Q-1}^{n-1}] \] (14)

The product \(C = AB\) which is obtained from (6) are broke down into products of \(Q\) factors \(A_u\) and \(B_u\) for \(u = 0, 1, \ldots, Q-1\) as:

\[ C = \sum_{u=0}^{Q-1} B_u A_u^T \] (15)

Where \(C_u\) denotes:

\[ C_u = B_u A_u^T \] (16)

2.2 Structure-I for RB Digit-Serial Multiplier

From the PSFG the structures for RB digit serial multiplier are designed. The Structure-I for RB digit serial multiplier is shown in Figure 1. Bit-Permutation Module (BPM), Partial Product Generation Module (PPGM) and finite field accumulator module are the three modules in this structure. To fed BPM output top number of partial product generation units, it rewrite the input bits of \(B\). The task of \(M\) node and \(A\) node is carries out by AND, XOR and register cells in PPGM and the \(n\)-bit parallel accumulation units are presented in finite field accumulator, which consists of the XOR and register cells. After the inputs \(a, b\) fed to PPGM it performs multiplication of inputs bits concurrently and then obtained result is added by XOR cell. The partial products obtained from PPGM are fed to finite field accumulator module and it acquires successive output.

2.3 Modification of Structure –I for RB Digit-Serial Multiplier

We have \((P=kd+l), \) for any \(p\) integer value, where \(0 \leq l < d\) and \(d < P\). For simple, we assume \(l=0\), however can easily extended to the cases where \(l \neq 0\). Define \(0 \leq h \leq k - 1\), and \(0 < f < d - 1\), such that (17) can be as:

\[ C_u = \sum_{h=0}^{k-1} \sum_{f=0}^{d-1} B_{u+hQ} a_{u+hQ} \] (17)

The PSFG is reformed by (17) to obtain proper structure-I for RB digit-serial multiplier in Figure 2. To form this structure a couple of shifting nodes, a couple of mul-
multiplication nodes and a couple of addition nodes of PSFG are united to achieve PPGM of \( \frac{P}{2} \) PPGUs and these nodes are accomplished by new PPGU. The functioning of this structure is same as the structure-I.

![Figure 1. Structure-I for RB digit serial multiplier.](image1)

2.4 Structure-II for RB Digit-Serial Multiplier

In Figure 3 it is shown the Structure-II for RB digit-serial multiplier. It is also obtained from the PSFG and which \((P-1)\) of A nodes are connected in serial and are formed to obtain \((P-2)\) number of A nodes into the pipeline form. And the pipeline forms of A nodes are formed by the pipeline XOR tree. And there is no need of padding 0 at input instead to meet the time requirement. It is same as structure-I only differ at the PPGM it contains the pipeline XOR tree.

2.5 Structure-III for Digit-Serial RB Multiplier

The structure-III for RB digit serial multiplier is shown in Figure 4. In this structure the operations such as bit-addition and bit-multiplication can be done in parallel to increase the throughput of the structure, but it is with two register cells. \((P+1)\) PPGUs are exists in this structure and at \((P+Q+1)\) clock cycles the first output is obtained. And the final output is achieved at \(Q\) cycles.

3. Proposed RB Digit Serial Multiplier

The proposed structure is shown in the Figure 5 it utilized redundant representation and is obtained from the PSFG. The proposed RB digit serial multiplier has bit-permutation, partial product generation and finite field accumulator modules.

![Figure 2. Modified structure –I for RB digit serial multiplier.](image2)

![Figure 3. Structure –II for RB digit serial multiplier.](image3)

In the proposed structure the input bits of B are distributed to the partial product generation block, for the first time the original bits of B are fed to the AND gate and which is multiplied with the input bits of the A. Then the bit shift inputs of B are fed to the AND gates of the PPGM. The outcome obtained is added by the XOR gate, this can be done in parallel and it is fed to finite field accumulator.
block. In this accumulator block the outcome acquired is accumulated and preserved by the XOR cell and register cell to get the final output.

Figure 4. Structure –III for RB digit serial multiplier.

Figure 5. Proposed Structure for RB digit serial multiplier.

4. Simulation Results and Comparison

Figure 6 shows the simulation result of proposed RB digit serial multiplier. It is with inputs a, b, clock and reset and output. Here it carryout bit shifting of B inputs and multiplied to ‘A’ inputs. It can be done in concurrent to acquire output, inputs a, b are initialized with 32 bits to obtain output. mul, shift_n, pu1, au and si1 shown are the internal signals of the multiplier (Table 1).

Table 1. Comparison table.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Number of slice registers</th>
<th>Power (w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing Structure-I</td>
<td>1191</td>
<td>0.748</td>
</tr>
<tr>
<td>Existing modified Structure-I</td>
<td>831</td>
<td>0.746</td>
</tr>
<tr>
<td>Existing Structure-II</td>
<td>799</td>
<td>0.744</td>
</tr>
<tr>
<td>Existing Structure-III</td>
<td>1927</td>
<td>0.752</td>
</tr>
<tr>
<td>Proposed Structure</td>
<td>75</td>
<td>0.743</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, Proposed RB digit serial multiplier is implemented and simulated. And the existing multipliers are also simulated to compare with the proposed structure. They are implemented for 32-bit, by comparison it is evident that the proposed structure achieved good performance in accordance usage of hardware and consuming of power. So, it can be efficient in designing of cryptographic devices and also in coding theory. And proved that it procure small area and power.

6. References