Design and Evaluation of Novel Effective Montgomery Modular Multiplication Architecture

Maryam Moayedi¹ and Abdalhossein Rezaie²*

¹ACECR institute of higher education, Isfahan branch, Isfahan 84175-443, Iran
²Academic Center for Education, Culture and Research (ACECR), Isfahan University of Technology (IUT) branch, Isfahan, Iran

¹mrm.moayedi@gmail.com, ²rezaie@acecr.ac.ir

Abstract

Secure communication is a challenging issue in modern industries and critical infrastructures. The core technology used for securing the communication is cryptography. Modular multiplication is an important operation in cryptosystems. This paper investigates a novel modular multiplication algorithm and architecture. In the proposed algorithm and architecture, the parallel architecture and compact SD technique are utilized to improve the performance of modular multiplication operation and cryptosystems. The proposed architecture is implemented on Xilinx Virtex 5 FPGA. The complexity analysis results and FPGA implementation results show that the proposed modular multiplication algorithm and architecture provide improvement on the total computation time and area×time complexity compared to other modified modular multiplication algorithms and architectures.

Keyword: Security, modular multiplication architecture, FPGA, parallel computation, compact SD

1. Introduction

Secure communication is a challenging issue in modern industries such as gas and oil and critical infrastructures such as public transport and power generation systems [1-5]. The core technology utilized to secure communication in these networks is cryptography [4, 1]. Elliptic Curve Cryptosystem (ECC) and RSA are two well-known cryptosystems, which are utilized for securing network communication [6, 7]. The basic operations in these cryptosystems are scalar multiplication and modular exponentiation. These operations are performed by repeating modular multiplication operation [6-10]. As a result, the performance of modular multiplication algorithm/architecture plays an important role in the performance of the cryptosystems. High-performance cryptosystem is hard to achieve without using the hardware acceleration [11].

Montgomery modular multiplication algorithm is one of commonly used modular multiplication algorithm. It is because Montgomery modular multiplication replaces the trial division with a series of simple shift and binary addition operations, which are simple in hardware implementations [11]. Although hardware implementation of the binary Montgomery modular multiplication is simple, but it is time-consuming operation. To improve the performance of Montgomery modular multiplication algorithm and architecture, several hardware implementation method and computational techniques have been developed that can be categories into four groups: using high-radix technique [11-17], using systolic array architecture [18-20], using carry-save addition architecture [11, 16, 21, 22, 23], and using scalable architecture [9, 12, 24, 25, 26, 27].

*Corresponding Author
Carry-save addition architectures and high-radix technique are two well-known methods which reduce the total computation time of the modular multiplication [16, 17, 21]. Using the high-radix technique, the number of required clock cycle is reduced at the expense of the critical path overhead [11]. Rezai and Keshavarzi proposed the variable Length Montgomery modular multiplication (VLM3) [11], which reduces the critical path overhead. They proposed a new integer representation in [11]. Using this new integer representation, the required high-radix partial multiplication is simplified to binary partial multiplication. In addition, the zero chain multiplication, the required addition, and the nonzero digit multiplication in several clock cycles are simplified as one multi-bit shift and one binary addition in only one clock cycle. They also improved this method by developing compact SD modular multiplication algorithm and architecture [17]. They eliminated the final comparison and subtraction by changing the number of iterations and utilized circuit.

On the other hand, Neto et. al., [7] have presented a method to partition the Montgomery modular multiplication architecture. In this method, the multiplier bits are distributed in such a way that it was equivalent to multiplication in $2^{k}$-radix. Using this method, the multiplication is performed in $\frac{n}{k}$ clock cycles.

This paper presents and evaluates novel modular multiplication algorithm and architecture. The proposed modular multiplication algorithm and architecture has the following distinctive characteristics:

a) It utilizes parallel computation to reduce the total computation time.

b) It utilizes high-radix technique to reduce the number of required clock cycles.

c) It uses the compact SD encoding to reduce the critical path compared to conventional high-radix technique.

d) It presents the complexity analysis.

e) It presents the FPGA implementation results.

The results show that the proposed modular multiplication algorithm and architecture provides an improvement in comparison with recent modified Montgomery modular multiplication algorithms and architectures. As a result, the proposed modular multiplication algorithm and architecture have a huge potential to become efficient algorithm and architecture for implementation of high-performance modular multiplication operation.

The remaining of this paper is organized as follows: in Section 2, the background of the proposed modular multiplication algorithm is investigated. Section 3 presents the proposed modular multiplication algorithm. The hardware implementation of the proposed modular multiplication is provided in Section 4. In addition, Section 4 compares the proposed modular multiplication architecture to other modified modular multiplication architectures. Finally, Section 5 provides conclusion of this paper.

2. Preliminaries

Montgomery modular multiplication algorithm [28] is commonly used algorithm to perform modular multiplication. This algorithm improves the performance of modular multiplication by replacing the trial division by the modulus with a simple addition and right shift operations. It should be noted that these operation are simple for hardware implementation [16, 17, 29]. Algorithm 1 illustrates the binary Montgomery modular multiplication algorithm.
### Algorithm 1. The Binary Montgomery Modular Multiplication (BM3) algorithm

Input: X, Y, M;
Output: S(n) = X × Y mod M;
1. \( S(0) = 0 \);
2. For \( i = 0 \) to \( n-1 \)
3. \( q_i = (S(i) + x_i Y) \mod 2 \);
4. \( S(i+1) = (S(i) + x_i Y + q_i M) / 2 \);
5. End for
6. If \( S(n) \geq M \) then \( S(n) = S(n) - M \);
7. Return \( S(n) \);

The inputs of this algorithm are multiplier \( X \), multiplicand \( Y \), and modulus \( M \) that are \( n \)-bit integers. The output of this algorithm is \( S(n) = (X \times Y) \mod M \). In this algorithm, \( S(i) \) denotes \( S \) in the \( i \)th iteration, \( x_i \) denotes the \( i \)th bit of \( X \). The output of this algorithm is calculated in \( n \) clock cycles. Therefore, this modular multiplication algorithm is time-consuming algorithm \([11, 17]\). To further improve the performance of Montgomery modular multiplication algorithm, several computational techniques and hardware implementation have been proposed such as \([7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] \). One of the efficient modular multiplication algorithms is K-Partition Montgomery Modular Multiplication (KPM3) algorithm \([7]\). The KPM3 algorithm is based on the Montgomery Modular Multiplication Partition (M3P) algorithm. Algorithm 2 shows the M3P algorithm.

### Algorithm 2. The M3P algorithm \([7]\)

Input: \( M, X, Y, j, K \);
Output: \( Z_{pj} = \text{MMP}(j, X, Y, M) = Y \times R_j \mod M \);
1. \( S_{pj}[0] = 0 \);
2. For \( i = 0 \) to \( n/K \)
3. \( a = S_{pj} + X_j^i \times Y_r \);
4. \( q_{K-1,...,0} = a \times Y_{1,0} \times (R_{K-1,...,0} \mod 2^K) \);
5. \( S_{pj}(i+1) = (a + q_{K-1,...,0}) / 2^K \);
6. End for
7. Return \( Z_{pj} = S_{pj} \times Y_r \);

In this algorithm, the inputs are \( X \), \( X = \sum_{i=0}^{n-1} x_i 2^i = \sum_{i=0}^{k-1} x_p i \) where \( x_p = \sum_{i=0}^{k-1} x_i 2^i \), multiplicand \( Y \), and modulus \( M \), which are \( n \)-bit integers. The final output of the M3P algorithm is a partial multiplication \( Z_{pj} = (X_p \times Y \times 2^{-n}) \mod M \).

### Algorithm 3. The KPM3 algorithm \([7]\)

Input: \( X, Y, M, j, k \);
Output: \( Z = \text{KPM3}(X, Y, M) = Y \times R^{-1} \mod M \);
1. For each partition \( j \) do in parallel
2. \( Z_{pj} = \text{M3P}(j, X, Y, M) \);
3. End for
4. \( Z = 0 \);
5. For \( j = 0 \) to \( k \)
6. \( Z = Z + Z_{pj} \);
7. for \( l = 0 \) to \( 1 \)
8. If \( Z \geq M \) then
9. \( Z = Z - M \);
10. End if
11. End for
12. End for
13. Return \( Z \);
In this algorithm, the partitioning method, which is shown in line 1 to 3, denotes a way to perform K number of M3P in parallel. However, this processing can be serially performed, but it requires K-1 times longer. It should be noted that the addition of partial results, $Z_p$, which is denoted in lines 4 to 11, is serially performed. So, it requires K-1 clock cycles. However, it can be performed in parallel at expense of area overhead.

Another efficient modular multiplication algorithm is compact SD modular multiplication algorithm [17]. Algorithm 4 shows the compact SD modular multiplication algorithm.

Algorithm 4. The compact SD modular multiplication algorithm [17]

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>For $i=0$ To $J+1$, /* $J$ shows the number of digits in compact SD representation/*</td>
</tr>
<tr>
<td>2.</td>
<td>$k=\text{Length}^{(i)}$;</td>
</tr>
<tr>
<td>3.</td>
<td>If $\text{Length}^{(i)} = 3$ Then $P := S(i)$, $k = k - 1$;</td>
</tr>
<tr>
<td>4.</td>
<td>Else</td>
</tr>
<tr>
<td>5.</td>
<td>If $Type_i = 0$ Then $P := S(i) + Y$;</td>
</tr>
<tr>
<td>6.</td>
<td>Else $P := S(i) - Y$;</td>
</tr>
<tr>
<td>7.</td>
<td>$q_i = P_{k-0} (2^{k+1} - M_{0_{-k}}^{-1}) \mod 2^{k+1}$;</td>
</tr>
<tr>
<td>8.</td>
<td>$S(i+1) = (P + qM)/2^{k+1}$;</td>
</tr>
<tr>
<td>9.</td>
<td>End for</td>
</tr>
<tr>
<td>10.</td>
<td>Return $S(J+2)$;</td>
</tr>
</tbody>
</table>

In the compact SD modular multiplication algorithm, the inputs are multiplier $X_{\text{CSD}}$, multiplicand $Y$, and modulus $M$ where $X_{\text{CSD}}$ denotes the multiplier representation in the compact SD representation that is calculated using algorithm 3 in [17]. The output of the compact SD modular multiplication is $S(J+2) = (XY2^{-n}) \mod M$, and $P_{k-0} = P \mod 2^{k+1}$.

It should be noted that in the compact SD representation, each digit includes a number of consecutive zero bits, which is followed by a nonzero digit. So, using compact SD representation for the multiplier, the calculation high radix partial multiplication of the $P=S(i)+X^iY$ is simplified to binary partition multiplication.

Although these modified Montgomery modular multiplication algorithms are efficient modular multiplication algorithms, but the performance of these algorithms can be improved as will be described latter.

3. The Proposed Modular Multiplication Algorithm

This section presents novel modified Montgomery modular multiplication algorithm and architecture. The basic block diagram of the proposed modular multiplication algorithm is shown in Figure 1.
As it is shown in Figure 1, the proposed architecture is composed of four main blocks:

(a) A converter for converting the multiplier X from binary representation to compact SD representation. The compact SD encoding algorithm and architecture, which are developed in [17], is utilized to perform this operation.

(b) A partitioning scheme to divide the compact SD representation of X to K partition.

(c) Compact SD modular multiplication architecture.

(d) An adder for addition the K-partition results.

Algorithm 5 shows the proposed modular multiplication algorithm for K=2.

Algorithm 5. The proposed modular multiplication algorithm

Input: X, Y, M
Output: Z = XY \mod M

1. Compute X_{CSD} using algorithm 3 in [17] (result in t digit)

2. Distribute X_{CSD} to K partition such that
   \[ X_{CSD} = \sum_{i=0}^{t-1} x_{CSD,i} 2^i = \sum_{i=0}^{K-1} x_{CSD,p_i} 2^{2i+jk} \]
   and \[ x_{CSD,p_i} = \sum_{j=0}^{t-1} x_{CSD,j+iK} 2^{i+jk} \]

3. For each partition j do in parallel (j from 0 to K-1)
In the proposed modular multiplication algorithm, the inputs are multiplier \( X \), multiplicand \( Y \), and modulus \( M \) that are \( n \)-bit integers. In step 1 of the proposed modular multiplication algorithm, the multiplier \( X \) is converted from binary representation to the compact SD representation using algorithm 3 in [17]. The output of this step is an integer in which each digit contains consecutive zero bits followed by a nonzero bit. After that, the \( K \)-partition method is applied to the encoded multiplier. The output of this step for \( K=2 \) decomposed the multiplier to two partitions, \( X_{\text{CSD0}} \) and \( X_{\text{CSD1}} \). Using the compact SD modular multiplication architecture, two partial results, \( Z_{P0} \) and \( Z_{P1} \), are computed in parallel. Finally, the results of the \( K \) partitions are added to compute the final result, \( S=X.Y \mod M \).

The utilized compact SD convertor for hardware implementation of the proposed modular multiplication is shown in Figure 2.

**Figure 2. The Utilized Architecture for the Compact SD Convertor in the Proposed Modular Multiplication Algorithm**

In addition, Figure 3 shows the utilized architecture for the compact SD modular multiplication in the proposed modular multiplication architecture.
4. Results and Discussion

This section compares the computational complexity analysis of the proposed modular multiplication algorithms and the proposed modular multiplication architecture with other modified modular multiplication algorithms and architectures.

4.1. The Computational Complexity Analysis

Our computational complexity analysis shows that the average Hamming weight for n-bit multiplier with K-partition in the proposed modular multiplication algorithm is \( \frac{n}{3K} \). However, the average Hamming weight for n-bit multiplier in the compact SD modular multiplication algorithm [17] is \( \frac{n}{3} \), and the average Hamming weight for n-bit multiplier

---

**Figure 3. The Utilized Architecture for the Compact SD Modular Multiplication in the Proposed Modular Multiplication Architecture**
with K-partitions in the KPM3 algorithm [7] is \( \frac{n}{K} \). Table 1 compares the computational complexity (average Hamming weight) of the multiplier in the proposed modular multiplication algorithm and other modified modular multiplication algorithms [17] and [7] for 256-, 512-, 1024- and 2048-bit modulus length.

**Table 1. Comparative Table for the Computational Complexity**

<table>
<thead>
<tr>
<th>Ref</th>
<th>Computation cost</th>
<th>n=256</th>
<th>n=512</th>
<th>n=1024</th>
<th>n=2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7] for K=2</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>[7] for K=3</td>
<td>86</td>
<td>171</td>
<td>342</td>
<td>683</td>
<td></td>
</tr>
<tr>
<td>[7] for K=4</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>86</td>
<td>171</td>
<td>342</td>
<td>683</td>
<td></td>
</tr>
<tr>
<td>This paper for K=2</td>
<td>43</td>
<td>86</td>
<td>171</td>
<td>342</td>
<td></td>
</tr>
<tr>
<td>This paper for K=3</td>
<td>29</td>
<td>57</td>
<td>114</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>This paper for K=4</td>
<td>22</td>
<td>43</td>
<td>86</td>
<td>171</td>
<td></td>
</tr>
</tbody>
</table>

Based on our analysis that is shown in Table 1, the proposed modular multiplication algorithm provides an improvement in terms of computation complexity in comparison with other modified modular multiplication algorithms [17] and [7].

It should be noted that the computational complexity in the proposed modular multiplication algorithm is reduced compared to [17] and [7] by about

\[ (1 - \frac{n}{K}) \times 100 = (1 - \frac{1}{K}) \times 100 \] (50% for K=2, 66.7% for K=3, and 75% for K=4) \hspace{1cm} (1)

\[ (1 - \frac{n}{K}) \times 100 = (1 - \frac{1}{3}) \times 100 = 66.7\% \hspace{1cm} (2) \]

Table 2 summarizes these computational complexity for the proposed modular multiplication algorithm compared to other modified modular multiplication algorithms [17] and [7].

**Table 2. The Computation Complexity Improvement**

<table>
<thead>
<tr>
<th>Ref</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>[17] for K=2</td>
<td>50%</td>
</tr>
<tr>
<td>[17] for K=3</td>
<td>66.7%</td>
</tr>
<tr>
<td>[17] for K=4</td>
<td>75%</td>
</tr>
<tr>
<td>[7]</td>
<td>66.7%</td>
</tr>
</tbody>
</table>

4.2. Hardware Implementation Results

The proposed architecture has been coded in VHDL and synthesized using Xilinx ISE14.1. Table 3 compares the proposed modular multiplication architecture to other modified multiplication architectures [211], [16], [22], [17] for 1024-bit length modulus.

**Table 3. Comparative Table for the FPGA Implementation Results for 1024 Bit Modulus**

<table>
<thead>
<tr>
<th>Ref</th>
<th>Time((\mu s))</th>
<th>Area (slice)</th>
<th>AT (slice×ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[21]</td>
<td>11.61</td>
<td>55702</td>
<td>646.7</td>
</tr>
<tr>
<td>[16] d=4</td>
<td>1.18</td>
<td>5702</td>
<td>6.7</td>
</tr>
<tr>
<td>[22]</td>
<td>5.7</td>
<td>1793</td>
<td>10.2</td>
</tr>
<tr>
<td>[17]</td>
<td>0.851</td>
<td>6091</td>
<td>5.2</td>
</tr>
<tr>
<td>This paper</td>
<td>0.420</td>
<td>11891</td>
<td>5.00</td>
</tr>
</tbody>
</table>
In this table, Time denotes the total computation time in μs, Area denotes the number of occupied FPGA slices, and AT denotes Area×time metrics in slices×ms.

Based on FPGA results, that are shown in Table 3; the total computational time and AT of the proposed modular multiplication architecture are reduced compared to other modified modular multiplication architecture at the expense of area overhead.

5. Conclusion

In modern industries and critical infrastructure, secure communication is an important issue. The increasing demands on data communications make the implementation of the cryptosystems a crucial research topic. ECC and RSA are well-known public-key cryptosystems. The basic operation in these cryptosystems is modular multiplication. This paper presented and evaluated a novel modular multiplication algorithm and architecture. The proposed modular multiplication algorithm and architecture used K-partition architecture and compact SD modular multiplication advantages to perform modular multiplication operation. The proposed modular multiplication architecture was implemented on Xilinx Virtex 5 FPGA. The complexity analysis results and FPGA implementation results demonstrated that the proposed Montgomery modular multiplication algorithm and architecture provided improvement on the other resulting total computation time and area×time compared to other modified modular multiplication algorithms and architectures.

References


Authors

Maryam Moayedi, she is an M.S. student at Academic Center for Education, Culture and Research (ACECR) institute of higher education, Isfahan, Iran. She received B.S degree in electronic engineering from Islamic Azad University, Najafabad, Isfahan, Iran in 2010. Her area of interests includes cryptography algorithm and its application.

Abdolfossein Rezai, he is an assistant professor in Academic Center for Education, Culture and Research (ACECR), Isfahan University of Technology (IUT) branch, Isfahan, Iran. He received Ph.D degree from Semnan University, Semnan, Iran in 2013, M.S. and B.S. degree from Isfahan University of Technology (IUT), Isfahan, Iran in 1999, and 2003, respectively. His research interests include network security, cryptography algorithm and its application, and neural network implementation in nanoelectronics.