RESEARCH ARTICLE OPEN ACCESS

An Area-efficient Montgomery Modular Multiplier for Cryptosystems

Anizha Radhakrishnan*, Seena George**
*(M.tech scholar, Department of ECE, Sree Narayana Gurukulam College of Engineering , Kolenchery, India)
** (Asst. Professor, Department of ECE, Sree Narayana Gurukulam College of Engineering , Kolenchery, India)

Abstract
RSA is one of the most widely adopted public key algorithms at present and it requires repeated modular multiplications to accomplish the computation of modular exponentiation. A famous approach to implement modular multiplication in hardware circuits is based on the Montgomery modular multiplication algorithm since it has many advantages. To speed up the encryption/decryption process, many high-speed Montgomery modular multiplication algorithms and hardware architectures employ carry-save addition. But CSA based architecture increases the area. In this paper, in order to reduce the area of the CSA based multiplier, an area-efficient algorithm called Double Add Reduce algorithm is introduced. Then the performance analysis of the new design with the previous had done for comparison.

Keywords— Area-efficient architecture, Carry save addition, Cryptography, Double add reduce algorithm, Montgomery modular multiplier.

I. INTRODUCTION

The motivation for studying area-efficient and fast modular multiplication algorithms comes from their application in public key cryptography. Modular exponentiation is a critical operation in RSA [1], Diffie- Hellman key exchange etc. It is a popular operation for encryption in public key cryptosystems. The core operation in this method is repeated modular multiplication.

Many algorithms are available for performing fast modular multiplication. These algorithms can be grouped in to two. That is, the first category algorithms perform modular reduction followed by multiplication. Classical, Karatsuba, Schonhage-Strassen and Barrett algorithms are examples of such multiplication and modular reduction algorithms [2], [3]. The second category performs modular reduction combined with multiplication. Montgomery algorithm [4] introduced in 1985, is a famous approach for carrying out fast modular multiplication and belongs to the second category.

For a single modular multiplication Montgomery method is disadvantageous. But the advantage comes when it performs modular exponentiation. Montgomery algorithm performs modular multiplication without division. That is, it replaces division with simple bit shift operation. In many public key systems Montgomery method is used for obtaining fast modular multiplication.

Many algorithms are introduced to perform Montgomery modular multiplication in a faster way. In this paper, we are introducing a new area reducing architecture for Montgomery modular multiplier. The remainder of this paper is organized as follows: Section II briefly reviews previous algorithms and their architectures for performing Montgomery multiplication. In section III, the mathematics behind the Montgomery algorithm is explained in detail. Then we have introduced a new area efficient Montgomery algorithm in section IV. Section V gives a detailed comparison between proposed method and the previous CSA based modular multiplier. Finally we concluded this paper in section VI.

II. PREVIOUS METHODOLOGIES

Montgomery multiplication algorithm is the most efficient algorithm available. The main advantage of Montgomery algorithm is that it replaces the division operation with shift operations. During two decades many alternative forms of Montgomery algorithms are introduced. These architectures use carry save addition. The work in [5] and [13] presented two types of Montgomery algorithms which use Carry Save Adder (CSA). One of the two types used four-to-two CSA and the other used five-to-CSA. They had given a brief comparison between these two versions of Montgomery multipliers. They had found that the multiplier using four-to-two CSA architecture has shorter critical path than that of five-to-two CSA multiplier. But extra storage elements and multiplexers are required for 4-to-2 architecture which probably increases the energy consumption. The work in [6] proposed a Montgomery multiplication algorithm using pipelined carry save addition to shorten the critical path delay of five-to-two CSA. This method also required additional pipeline registers and multiplexers which will increase the area.

Ming Der Shieh presented [7] a new algorithm for high speed modular multiplication. This new Montgomery multiplier performs modular reduction in a pipelined fashion, so that the critical path delay is reduced from the four-to-two to three-to-two carry-save addition. Then it requires additional pipeline registers to store intermediate values. All the above works were not discussed about the energy consumption.

Several previous works [8], [9] have developed techniques to reduce the power/energy consumption of Montgomery multipliers. In [8], some latches named glitch blockers are located at the outputs of some circuit modules to reduce the spurious transitions and the expected switching activities of high fan-out signals in the radix-4 scalable Montgomery multiplier. Shiann Rong Kuang [9] tried to reduce the energy consumption of CSAs and registers in the CSA-based Montgomery multipliers via a new technique. They modified the CSA based Montgomery multiplier algorithm and as a result of this, the number of clock cycles required to complete the multiplication is largely decreased. To achieve further energy reduction, they have adjusted the internal structure of barrel register full adder and then applied the gated clock design technique. But this energy efficient algorithm increased the total area of the design.

In order to reduce the area, we are presenting a new algorithm which can modify the CSA based algorithm in [9]. Before introducing the area efficient algorithm, let us see the mathematics behind the Montgomery modular multiplication.

III. MONTGOMERY MODULAR MULTIPLICATION

Modular multiplication of two integers X and Y, simply performs,

$$Z = X \cdot Y \bmod M \qquad (1)$$

Here X, Y and M are n - bit numbers and M should be greater than X and Y. Instead of computing X. Y, the Montgomery multiplication [10] algorithm computes,

$$Z' = MP\,(X, Y, M) = X \cdot Y \cdot 2^{-n} \bmod M \qquad (2)$$

MP denotes Montgomery Product and sometimes the equation (2) can also be represented as shown below.

$$Z' = MP\,(X, Y, M) = X \cdot Y \cdot R^{-1} \bmod M \qquad (3)$$

Here $R = 2^n$ and R^{-1} is the multiplicative inverse of R mod M. That is,

$$R \cdot R^{-1} \bmod M = 1 \qquad (4)$$

In order to perform Montgomery multiplication the numbers should be converted to the Montgomery domain. The conversion to the Montgomery domain is very simple. The conversion from integer domain to Montgomery domain is shown below.

$$X' = X \cdot R \bmod M \qquad (5)$$
$$Y' = Y \cdot R \bmod M \qquad (6)$$

Here X' and Y' are the Montgomery forms of X and Y. After completing the multiplication in Montgomery domain, we have to convert it back to the integer domain to obtain the final result. The conversion from the Montgomery product Z' to Z is shown below:

$$Z = MP\,(Z', 1, M) = Z' \cdot 1 \cdot R^{-1} \bmod M \qquad (7)$$

These conversions are required only at the starting point and ending point. It is important to note here that to find the modular product the Montgomery algorithm does not perform any division. On the other hand, it performs just addition and bit shift operation. The pseudo code [11], [12] for finding Montgomery product S is given below.

```
Algorithm 1
MP1(X,Y,M) {
Initially S = 0;
for i = 0 to n-1 {
        $S = S + X_i \cdot Y$ ;
        If S is odd then S = S + M ;
        S = S/2;  }
   If $S \geq M$ then S = S – M;  }
```

Here we can see that algorithm requires division by 2. Division by 2 is equivalent to shifting the bits to right which is fast and easy in hardware. Thus we can avoid the division. Only requirement is to perform simple conversion from Montgomery domain done before and after multiplication.

IV. PROPOSED METHODOLOGY

The work in [9], proposed an energy-efficient four-to-two CSA based Montgomery algorithm. While reducing energy, area of the multiplier is slightly increased. In this paper, we propose an area efficient, fast and low power algorithm called Double Add Reduce (DAR) algorithm which can replace the CSA based multiplier.

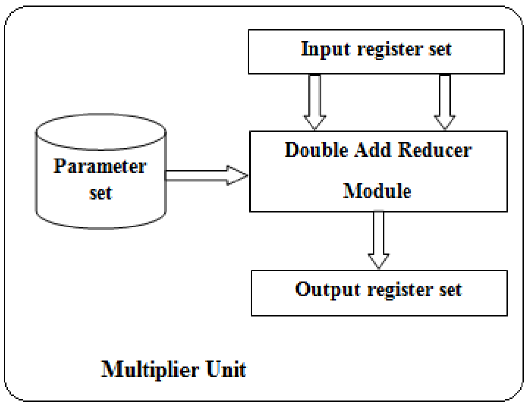

Fig.1. Block diagram of multiplier based on DAR algorithm

Figure 1 shows the block diagram of Montgomery modular multiplier. Here main block is a double add reduce block.

4.1 Double Add Reduce (DAR) algorithm

The DAR block computes Montgomery product using DAR algorithm which is shown below:

```
Algorithm 2
MP2(X,Y,M)
{
$S_{-1} = 0$;
$Y = 2 \times Y$;
For i=0 to n {
        $q_i = S_{i-1} \bmod 2$;                    (LSB of $S_{i-1}$)
        $S_i = (S_{i-1} + q_i.M + X_i.Y) / 2$; }
Return $S_n$;
}
```

The multiplicand X is shifted up by 1 bit to simplify the calculation of q_i. This implies that the result is a factor of 2 too large, and so the for loop must be executed an additional time to eliminate this factor.

4.2 Architecture

The architecture based on DAR algorithm is shown in figure 2. That is, the algorithm can be executed with the help of two adders. To remove the dependency of q_i on the addition of $X_i.Y$ and S_{i-1}, Y can be shifted up 1 bit, thus forcing the LSB of $S_{i-1} + X_i.Y$ to always be zero, as illustrated in figure 2.

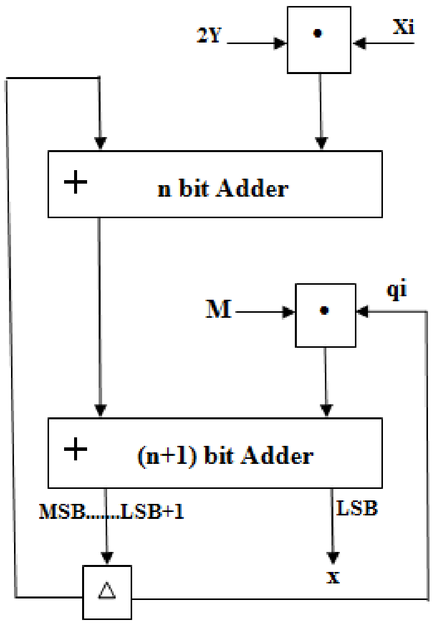

Fig.2. Architecture based on DAR algorithm

However, this extra factor of 2 must be removed by an extra iteration, meaning that the number of clock cycles is equal to n+1.

V. RESULT ANALYSIS

In order to verify the area efficiency, we have captured our new design in VHDL and implemented into the Xilinx Spartan 3E series of FPGA. For performance comparison, we have also captured the CSA based multiplier design in VHDL and implemented into the FPGA. Following section gives the performance results of these implementations.

5.1 Detailed area analysis

During implementation, we have generated device utilization report using Xilinx ISE 8.1i software tool. The design summary report generated by the Xilinx ISE software tool can be used to analyze the area utilized by the two versions of the Montgomery modular multiplier. Table I gives a detailed area analysis of the two implementations.

TABLE I. AREA ANALYSIS OF THE TWO IMPLEMENTATIONS

	CSA based modular multiplier (8- bit)	**DAR based modular multiplier (8- bit)**
Number of slice flipflop used	64	23
Number of 4 input LUTs used	144	60
Total equivalent gate count for design	1527	595

Table I shows that proposed DAR based modular multiplier (8- bit) considerably reduces the total area used by the design.

5.2 Timing and power analysis

From the simulation results of both CSA multiplier (8 bit) and DAR multiplier (8 bit), we have seen that after applying inputs the later one takes less time for producing the output. For same inputs Modified multiplier gives the result in less time. From the Figure 3, CSA based multiplier takes 2.185 µs for generating the output while from the Figure 4, DAR based multiplier takes only 1.745 µs. After applying inputs, DAR based modular multiplier takes less time to produce output.

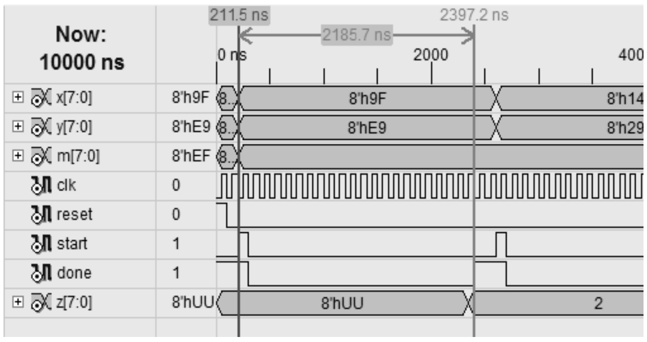

Fig.3. Simulation result for 4-to-2 CSA based modular multiplier

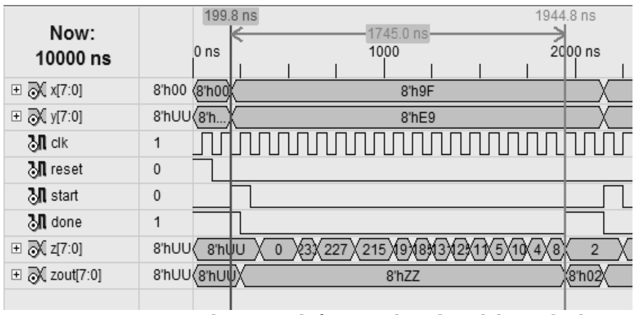

Fig.4. Simulation result for DAR based modular multiplier

The power summary provides information about the usage of voltage and other power resources for your entire design or for a single design module. During synthesis we have generated the power data choosing device type Spartan 3 and package tq144. These two multipliers consumes almost same power (18mW).

VI. CONCLUSION

This paper presented an efficient algorithm and its corresponding architecture to reduce the area without affecting the speed of a Montgomery modular multiplier. Experimental results showed that the proposed method is indeed capable of reducing the area of 8 bit Montgomery modular multiplier up to 61% compared to the previous CSA based design. In addition to this, our new design increased the speed to 20% and having almost same power consumption. Here we cannot observe significant change in power consumptions of the two versions of multipliers. In future, we will try to develop efficient architecture to carry out fast modular multiplication with low energy consumption.

VII. ACKNOWLEDGMENT

We would like to express our sincere gratitude to the Management and staff, SNG College of Engineering for providing the facilities.

REFERENCES

[1] Taek-Won Kwon, Chang-Seok You," *Two implementation methods of a 1024-bit RSA Cryptoprocessor based on modified Montgomery algorithm*", 0-7803-6685-9/01/$10.00 2001 IEEE.

[2] Andre Weimerskirch and Christof Paar, "*Generalizations of the Karatsuba Algorithm for Efficient Implementations*", IEEE Int.Conf. Comput. Design, 2005.

[3] Antoon Bosselaers, Rene Govaerts and Joos Vandewalle," *Comparison of three modular reduction functions*", ° Springer-Verlag 1993 oct 25.

[4] Peter L. Montgomery, "*Modular multiplication without trivial division", Mathematics of computation*, Vol.44, No. 170. (Apr 1985) ,pp. 519-522.

[5] C. McIvor, M. McLoone, and J. V. McCanny, "*Modified Montgomery modular multiplication and RSA exponentiation techniques*," IEE Proc. Comput. Digit. Tech., vol. 151, no. 6, pp. 402–408, Nov. 2004.

[6] K. Manochehri and S. Pourmozafari, "*Fast Montgomery modular multiplication by pipelined CSA architecture*," in Proc. IEEE Int. Conf. Microelectron., Dec. 2004, pp. 144–147.

[7] M.-D. Shieh, J.-H. Chen, W.-C. Lin, and H.-H.Wu, "*A new algorithm for high-speed modular multiplication design*," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 9, pp. 2009–2019, Sep. 2009.

[8] H.-K. Son and S.-G. Oh, "*Design and implementation of scalable low-power Montgomery multiplier*," in Proc. IEEE Int. Conf. Comput. Design, 2004, pp. 524–531.

[9] Shiann-Rong Kuang, Jiun-Ping Wang," *Energy-Efficient High-Throughput Montgomery Modular Multipliers for RSA Cryptosystems*", IEEE Transactions on very large scale integration (VLSI) systems, vol. 21, no. 11, november 2013.

[10] J. C. Neto, A. F. Tenca, and W. V. Ruggiero, "*A parallel k-partition method to perform Montgomery multiplication*," in Proc. IEEE Int. Conf. Appl.-Specif. Syst., Arch. Process., Sep. 2011, pp. 251–254.

[11] Ambika R, Hamsavahini R, " *A Survey on Hardware Architectures for Montgomery Modular Multiplication Algorithm*", IJETCAS 13-342; © 2013.

[12] Kooroush Manochehri, Saadat Pourmozafari," *Montgomery and RNS for RSA hardware implementation*", Computing and Informatics, Vol. 29, 2010, 849–880.

[13] Alan Daly, Ciaran McIvor, William Marnane," *Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures*", November, 2003.