

# HIGH SPEED FPGA IMPLEMENTATION OF RSA ENCRYPTION ALGORITHM

O. Nibouche<sup>1</sup>, M. Nibouche<sup>2</sup>, and A. Bouridane<sup>3</sup>

<sup>1</sup>Faculty of Informatics, The University of Ulster at Magee

<sup>2</sup>Faculty of Computing, Engineering and Mathematical Sciences  
University of the West of England

<sup>3</sup>School of Computer Science, The Queen's University of Belfast

## ABSTRACT

In this paper, new structures that implement RSA cryptographic algorithm are presented. These structures are built using a modified Montgomery modular multiplier, where the operations of multiplication and modular reductions are carried out in parallel rather than interleaved as in the traditional Montgomery multiplier. The global broadcast data lines are avoided by interleaving two operations into the same structure, thus making the implementation systolic. The results of implementation in FPGA have shown that the proposed RSA structures outperformed those structures built around a traditional Montgomery multiplier in terms of speed. In terms of area usage, an area-efficient architecture is shown in this paper that has the merit of having a high speed and a reduced area usage when compared with other architectures.

## 1. INTRODUCTION

Recently, there have been growing interest and a lot of research activity related to developing algorithms and architectures of modular multiplication for cryptography. Several reasons have fuelled such an enthusiastic work, among them, the security requirement of the booming electronic commerce, where the safety of the transactions is becoming a major concern.

Public key cryptography has shown to be an attractive technique for such applications. One of the main and best-known public key systems is the RSA cryptosystem. This cryptosystem performs fast modular exponentiation operations on numbers of hundreds of bits [1], which in turn, can be broken into a series of modular multiplication operations. Therefore, the implementation of such systems, which requires efficient architectures to compute the modular product, has motivated the development of a number of modular multiplication algorithms and architectures.

A clear distinction between the algorithms used for modular multiplication can be based upon the data format. Two major classes of algorithms can be distinguished: Most Significant Bit First (MSBF) algorithms and Least Significant Bit First (LSBF) algorithms. Montgomery's algorithm [2] makes the LSBF approach useful when performing numerous successive modular multiplication operations, such as modular exponentiation. The algorithm is used to speed up both the modular multiplication and squaring operations required during the exponentiation process. Rather than calculating the residue of the division operation as the MSBF methods do, the algorithm computes the modular product of two numbers multiplied by a scaling factor, which is relatively prime to the modulus. This allows the algorithm to perform divisions by a power of two, which is a shift operation, thus making the design of modulo multipliers easier.

In this paper, new RSA structures are presented. They are built around a modified version of Montgomery's multiplication algorithm [9]. The main feature of the proposed method is that the Montgomery's algorithm is broken into two concurrent

multiplication operations. The first one consists of a conventional multiplication operation of the two operands while the second is a reduction operation, which is needed to ensure that the partial results remain in the range required by the algorithm. This reduction operation is nothing else than a multiplication operation carried out in such a way that the least significant part of the result is the two's complement of the least significant part of the first multiplication operation [9].

The derived architectures are systolic and can be pipelined to the bit-level by interleaving multiple modular multiplication operations onto the same structure. A fully systolic bit serial parallel structure can be designed for interleaving two Montgomery's multiplication operations onto the same multiplier, which is very useful for Montgomery's exponentiation as these operations can be the multiplication and squaring operations required in the Montgomery exponentiation. Thus, avoiding the global broadcast of the data lines and making the structure systolic [5,7].

The paper is organised as follows: section 2 reviews the mathematical background of the RSA cryptosystem and the modular exponentiation operation while the Montgomery's modular multiplication algorithm is described in section 3. In section 4, the modified Montgomery's algorithm, its parallel and bit serial implementations are reviewed. The new RSA structures are shown in section 5. The first structure uses the modified Montgomery structure while the second proposed structure interleaves two modular multiplication operations into the same structure, thus making it systolic. The results of the FPGA based implementation are given in section 6. The conclusions are made in section 7.

## 2. MODULAR EXPONENTIATION AND RSA CRYPTOGRAPHY

In 1977, Rivest, Shamir and Adleman [8] introduced what has become one of the most successful and widely used public key cryptosystem based on computing modular exponentiations. The security challenge brought by RSA cryptography gets its level of confidence from the difficulty of factorising large numbers. As an example [1], hundreds of computers were used worldwide for a number of months in order to factorise a 513-bit integer. For the purpose of keeping the data as secured and secret as possible, current implementations of the RSA use 1Kb key wordlength or longer [1].

As it was proposed by [8], the modulus  $M$  of the RSA algorithm is the product of two suitably generated secret prime numbers  $P$  and  $Q$ :  $M = P \times Q$ . The public exponent (also known as the encryption key)  $E$  is randomly chosen such that it is prime to  $(P-1) \times (Q-1)$ . The secret exponent (also known as the decryption key)  $D$  is computed using the extended Euclidean algorithm such that:

$$\langle E \times D \rangle_{(P-1) \times (Q-1)} = \langle 1 \rangle_{(P-1) \times (Q-1)} \quad (1)$$

where  $\langle \rangle_M$  denotes a modulo  $M$  operation. The two numbers  $P$  and  $Q$  are no longer needed. They should be discarded and never revealed. An  $n = km$  bits message is divided into  $m$  blocks of  $k$ -bit integers each. A  $k$ -bit word  $A$  is encrypted into a word  $Enc(A)$  defined by  $Enc(A) = \langle A^E \rangle_M$ . An encrypted word  $B$  is decrypted into a word  $Dec(B)$  defined by  $Dec(B) = \langle B^D \rangle_M$ .

The encryption and decryption operations are mutual inverses, i.e.  $Enc(Dec(A)) = Dec(Enc(A)) = A$  with  $0 \leq A < M$ .

A standard way for performing the modular exponentiation is by using repeated modular multiplication and squaring operations, where the exponentiation is carried out iteratively, as shown in *Algorithm 1* for the case of a LSBF scheme.

**Algorithm 1**

$$B = \langle A^E \rangle_M$$

$$E = \sum_{i=0}^{n-1} e_i 2^i, P_0 = 1, B_0 = A, B = P_n$$

for  $i = 0$  to  $n-1$

$$\{ B_{i+1} = \langle B_i^2 \rangle_M$$

if  $e_i = 1$  then  $P_{i+1} = \langle P_i B_i \rangle_M$   
else  $P_{i+1} = P_i$  }

### 3. MONTGOMERY MULTIPLICATION: BACKGROUND

Let the modulus  $M$  be an integer within the range  $[2^{n-1}, 2^n]$  and let  $R$  be  $2^n$ . Montgomery's multiplication algorithm requires  $R$  and  $M$  to be relatively prime, i.e.,  $gcd(R, M) = gcd(2^n, M) = 1$ , which is satisfied if  $M$  is odd as it is required by the algorithm. By exploiting this property, the Montgomery reduction algorithm introduces an efficient multiplication scheme, which computes the modular product,  $P$ , of two given integers,  $A$  and  $B$ , as follows [2]:

$$P = \langle ABR^{-1} \rangle_M \quad (2)$$

where  $R^{-1}$  is the inverse of  $R$  modulo  $M$ . In order to describe his algorithm, Montgomery introduced the quantity,  $M'$ , which is the inverse of  $-M$  modulo  $R$ , i.e.  $M' = \langle -M^{-1} \rangle_R$

The computation of the Montgomery multiplication is carried out using *Algorithm 2* as follows:

**Algorithm 2**

$$\{ T = AB$$

$$P = (T + \langle TM' \rangle_M) M/R$$

If  $P \geq M$  then  $P = P - M$  }

The algorithm uses the multiplication modulo  $R$  and the division by  $R$ , which are faster and simpler than the computation of  $AB \bmod M$  which involves division by  $M$ .

For a hardware implementation, a systolic array can be derived from the bit-wise version of Montgomery multiplication as shown in *Algorithm 3* [2]:

**Algorithm 3**

$$0 < A = \sum_{i=0}^n a_i 2^i < M, 0 < B < R, P_i = 0$$

{ for  $i = 0$  to  $n-1$

$$q_i = \langle P_{i-1} + a_i B \rangle_2$$

$$P_i = P_{i-1} + a_i B + q_i M / 2 }$$

If  $P_{n-1} \geq M$  then  $P_{n-1} = P_{n-1} - M$

*Algorithm 3* interleaves the multiplication steps with the reduction steps and the final result is  $\langle AB2^{-n} \rangle_M$ .

Several structures have been presented in the literature for the implementation of Montgomery's algorithm [3-7]. These architectures, which implement the interleaved *Algorithm 3*, are either based on the use of Carry Save Adders (CSAs) or Carry Propagate Adders (CPAs). Figure 1 depicts a bit serial implementation of this algorithm based on a CSA multiplication scheme and using two rows of gated Full Adders (gFAs). The first row is used for the multiplication part of the algorithm while the second part implements the reduction step. The loops within the bit serial structure suggest that the structure can no further be pipelined. Furthermore, the serial inputs are broadcast to all the cells. Such a global distribution lowers the clock frequency, and therefore should be eliminated.

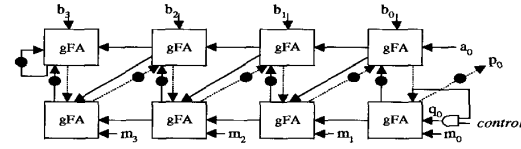


Figure 1. A bit serial Montgomery Multiplier

### 4. A MODIFIED NON-INTERLEAVED MONTGOMERY'S MULTIPLIER

As shown by *Algorithm 3*, the Montgomery's modular multiplication algorithm is equivalent to two interleaved conventional multiplication operations, where the second one is a reduction operation which is required to keep the partial results within the range  $[0, M+B]$ .

In this section, a modified Montgomery's algorithm and its structure are presented. This is achieved by breaking the initial algorithm into two multiplication operations that can be carried out simultaneously. Let  $T$  be the product of  $A \times B$ , i.e.:

$$T = AB = T_0 + T_1 R \quad (3)$$

where  $T_0$  and  $T_1$  are the least significant word and most significant word of  $T$ , respectively [9]. *Algorithm 4* is the non-interleaved version of Montgomery algorithm:

**Algorithm 4:**

$$P'_1 = 0, c_0 = 1, P''_1 = 0$$

For  $i = 0$  to  $n-1$

$$\{ \text{Step 1: } P'_i = P'_{i-1} + 2a_i B$$

$$\text{Step 2: } P'_i + c_i = \bar{T}_{0,i} + 2c_{i+1}$$

$$\text{Step 3: } q_i = (P'_{i,i} \oplus \bar{T}_{0,i})$$

$$\text{Step 4: } P''_i = P'_{i-1} + q_i M \}$$

$$\text{Step 5: } P = (P'_{n-1} + P''_{n-1}) / 2^n$$

$$\text{Step 6: If } P \geq M \text{ then } P = P - M \}$$

where  $\oplus$  is the bit-wise exclusive OR operation,  $\bar{T}_{0,i}$ ,  $P''_i$ , and  $P'_i$  are the  $i^{\text{th}}$  bit of  $\bar{T}_0$ ,  $P''_i$ , and  $P'_i$  respectively and where  $\bar{T}_0$  is two's complement of  $T_0$ . The modular multiplication in this algorithm is broken into two concurrent multiplication operations and computes Montgomery's multiplication by using  $\bar{T}_0$ , the two's complement of  $T_0$ , to select the reduction value. The results from the two multipliers are then added and a division by  $R$  follows. Step 1 is concerned with the calculation of the product  $AB$ . In step 2, the two's complement of  $T_0$  is computed.

In fact, the algorithm uses a function  $f$  given by:  $\bar{T}_0 = R - T_0 = \langle f(T_0, M) \rangle_R$ . The function  $f$  is computed in both steps 3 and 4. The results of step 1 and step 4 are added and a division by  $R$  follows. As in the original algorithm (Algorithm 2), a last reduction is required (step 6).

A parallel implementation of the proposed modified algorithm (Algorithm 5) is depicted in Figure 2. The first multiplier uses a conventional multiplier while the Least Significant (LS) cells of the second multiplier are built around a Half Adder (HA) with some additional gates. The LSB cell at the  $i^{th}$  row of the second multiplier receives the bit  $\bar{T}_{0,i}$  and computes the bit  $q_i$ , which is broadcast to the remaining cells of the row. The LSB cell also computes a carry bit which is fed to the row below. However, the overall delay of the cell does not exceed the delay of a gated FA. In addition to the two multipliers, the parallel structure uses a two's complement circuit and an adder to compute the final result. A bit serial parallel implementation of Algorithm 4 is shown in Figure 3. The algorithm is implemented using two bit serial multipliers, a serial adder for use to perform the addition of step 5, and a serial two's complement circuit that is used to two's complement the product  $AB$ . The clock path of the bit serial multiplier of Figure 3 is equivalent to that of a gated FA and a latch. However, the data lines are broadcast to all the cells. Such a global distribution lowers the clock frequency, especially for large operands, which are usually used in cryptography. In Figure 4, a bit-serial architecture that is fully systolic is depicted. The global distribution lines are avoided. This has been carried out by interleaving two modular multiplication operations onto the same structure and by pipelining the feedback loops [9]. This technique has been used in many works [7, 9], and the number of operations interleaved onto the same structure depends on the number of latches added to the feedback loops. For RSA cryptography, the two interleaved modular multiplication operations are the two operations involved in the modular exponentiation. In this way, the serial multiplier can be used as a modular exponentiation structure.

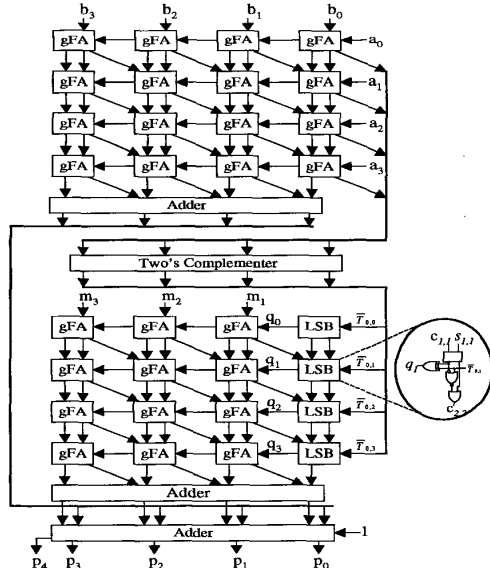


Figure 2. A parallel non-interleaved implementation of Montgomery algorithm

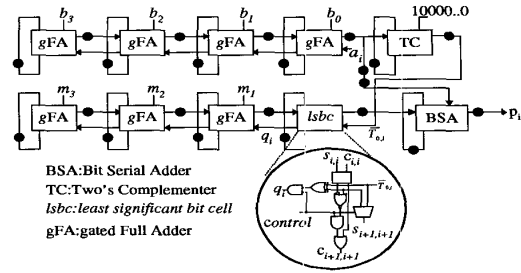


Figure 3. A serial implementation of the modified Montgomery algorithm

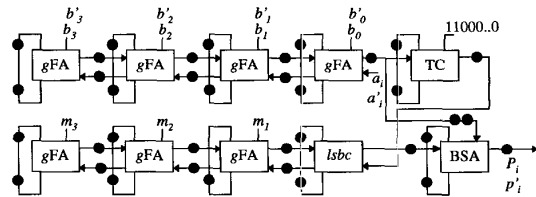


Figure 4. A systolic serial implementation of the modified Montgomery algorithm

### 5. NEW HIGH SPEED RSA ARCHITECTURES:

This section presents two new RSA serial structures built around the non-interleaved multipliers of figures 3 and 4. Figure 5 depicts the first structure that uses Figure 3 modular multiplier. As in Algorithm 1, two multipliers are required for squaring and multiplication. For the squarer, both a parallel and serial registers are required. For the multiplier in Figure 5, only a parallel register is required to store the data for the next iteration. For this multiplier, the serial input comes from a multiplexer that controls the multiplication operation. If  $e_i = 1$ , then the output of the squarer is fed to the multiplier. Otherwise, a scaling factor equal to  $\langle 2^n \rangle_M$  is fed to the multiplier. It is cancelled by the  $\langle 2^{-n} \rangle_M$  factor introduced by the Montgomery algorithm, which means the operation is equivalent to a multiplication by 1. Scaling factors must also be fed to the parallel registers at the beginning of the encryption/decryption operation. Therefore,  $\langle 2^n A \rangle_M$ , and  $\langle 2^n \rangle_M$  are loaded into squarer parallel register and the multiplier parallel register, respectively. As with all RSA structures that use Montgomery multiplier, a final multiplication by one is required at the end to cancel the effect of the  $\langle 2^n \rangle_M$  scaling factor. The second RSA structure proposed in this paper is depicted in Figure 6. This structure uses a single multiplier that is shown in Figure 4. Therefore, the modular multiplication and modular squaring operations are interleaved into the same structure. When carrying out the modular exponentiation in this way, two parallel registers are required in addition to a serial register that provided the serial input to the multiplier. To switch between the two parallel registers, a row of multiplexers is required.

### 6. EVALUATION

In this section, a comparison between the different RSA structures presented in this paper is carried out. The comparison is made in terms of the hardware usage and frequency for different wordlength. The structures of Figures 5 and 6 have been

implemented in an FPGA XCV-100E device. An RSA structure that uses the "conventional" Montgomery multiplier of Figure 1 has also been implemented. In addition, a 2-bit digit architecture derived from Figure 5 structure by unfolding (an unfolding factor of 2) has also been implemented. For illustration purposes, the chosen wordlengths are 64, 96, and 128 bits.

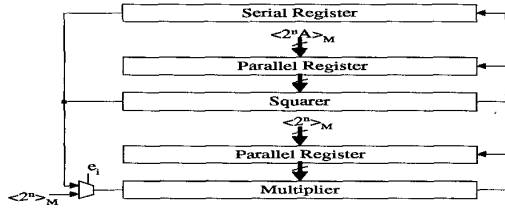


Figure 5. An RSA structure built around the modified Montgomery algorithm

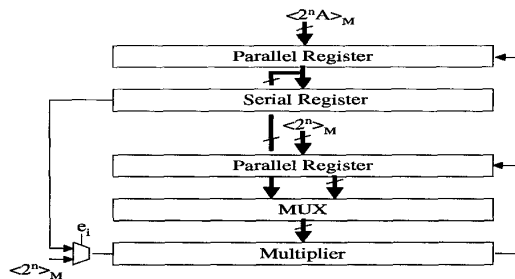


Figure 6. An area-efficient RSA structure

Figure 7 shows the performances of the different structures in terms of speed. Arc1, which is the conventional RSA structure, shows similar speed performances as Arc2, which is the unfolded architecture from Figure 5 structure. However, Arc1 processes one bit per clock cycle while Arc2 processes 2 bits of data per clock cycle. Therefore, despite working at almost the same clock frequency, the proposed Arc2 has doubled the throughput rate of Arc1, which the original Montgomery algorithm. Arc 3 has a higher clock frequency than Arc1. This is due to the fact the clock path has been reduced in Arc3 from two adders to a single adder. Furthermore, the effect of removing the global broadcast lines is clear on Figure 7. Arc4, which is the structure of Figure 6, has the highest frequency as it only uses nearest neighbour links. However, its throughput rate is half of the throughput rate of Arc1 and Arc3. In terms of area usage, Arc4 has the lower hardware usage when implemented in FPGA. This structure is area efficient because it only requires one multiplier. Arc1 and Arc 3 both require two multipliers and show similar area usage.

### 7. SUMMARY

In this paper, new structures that implement the RSA cryptographic algorithm are presented. These structures are built using a modified Montgomery modular multiplier, where the operations of multiplication and modular reductions are carried out in parallel rather than interleaved as in the traditional Montgomery multiplier. The implementation results have shown that the proposed architectures can reach twice the throughput rate of an RSA structure that uses the original Montgomery algorithm. An area-efficient RSA structure that also exhibits a high clock frequency is obtained by interleaving two multiplication operations into the same structure.

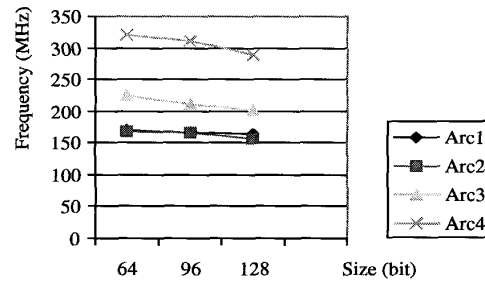


Figure 7. Frequency performances

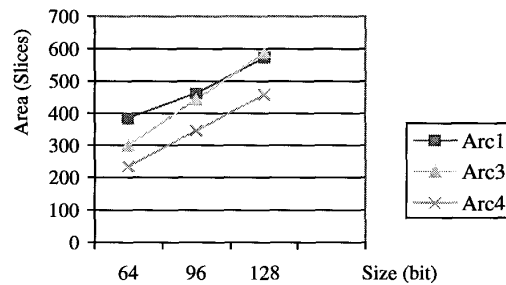


Figure 8. Hardware usage performances

### 8. REFERENCES

- [1] M. Shnad and J. Vuillemin, 'Fast implementation of RSA cryptography', Proceedings of the 11th IEEE Symposium on Computer Arithmetic, 1993.
- [2] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comp.*, vol. 44, no. 170, pp. 519-521, 1985.
- [3] J. Guo and C. Wang, "A novel digit-serial systolic array for modular multiplication" in *Proc. IEEE ISCAS '98*, vol. 2, pp. 177-180, 1998.
- [4] C. D. Walter, "Systolic Modular Multiplication" *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 376-378, 1993.
- [5] W. T. Tsai, C. B. Shung, and S. Wang, "Two Systolic Architectures for Modular Multiplication" *IEEE Trans. VLSI Systems.*, vol. 8, no. 1, pp. 103-107, 2000.
- [6] A. F. Tenca and C. K. Koc, 'A scalable architecture for Montgomery multiplication. *Cryptographic Hardware and Embedded Systems*, C. K. Koc and C. Paar, editors, Lecture Notes in Computer Science No. 1717, pages 94-108, 1999.
- [7] W. L. Freking and K. K. Parhi, "Ring-Planarized Cylindrical Arrays with Application to Modular Multiplication" *IEEE Workshop on Signal Processing Systems Design & Implementation (SIPS 2000)*, Lafayette, Louisiana, USA, pp 497-506.
- [8] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining Digital Signatures and Public Key Cryptosystems," *Comm. of ACM*, 21 (2), pp. 120-126, Feb. 1978.
- [9] O. Nibouche, A. Bourriadne, and M. Nibouche. "Bit-level Montgomery structures" to appear in *IEE Proceeding-Computers and Digital Techniques*.