American Journal of Applied Sciences 11 (5): 851-856, 2014
ISSN: 1546-9239

doi:10.3844/ajassp.2014.851.856 Published Online 11 (5) 2014 (http://www.thescipub.com/ajas.toc)

CARRY SAVE COMMON MULTIPLICAND MONTGOMERY FOR RSA CRYPTOSYSTEM

[1]Rupali Verma, [2]Maitreyee Dutta and [3]Renu Vig

[1]Computer Science and Engineering, PEC University of Technology, Chandigarh, India
[2]Computer Science and Engineering, National Institute of Technical Teachers' Training and Research, Chandigarh, India
[3]University Institute of Engineering and Technology, Panjab University, Chandigarh, India

Received 2014-01-13; Revised 2014-02-13; Accepted 2014-03-08

ABSTRACT

RSA public key cryptosystem provides encryption and digital signatures. With growth of key size an efficient design of RSA in terms of area, frequency, throughput and power consumption is hard to achieve. Also with the different type of attacks possible, a need for secure RSA cryptosystem which is attack resistant has arisen. This study presents RSA design with Montgomery powering ladder and proposed carry save common multiplicand Montgomery on FPGAs. Since the modular exponentiation is based on Montgomery powering ladder therefore it is power attack resistant. Common multiplicand Montgomery modular multiplication reduces the complexity by computing once the common operations in modular squaring and modular multiplication. The proposed carry save common multiplicand Montgomery modular multiplication maintains intermediate results in carry save form and utilizes the DSP slices to convert the redundant results into binary at the end of the modular multiplication. The proposed RSA design implemented on FPGAs is efficient in terms of area, frequency, power consumption and is power attack resistant.

Keywords: Carry Save, Common Multiplicand, FPGA, Montgomery, RSA

1. INTRODUCTION

RSA is a popular public key cryptosystem (Rivest *et al*., 1978). The security of RSA lies in large size operands which are 1024 bits or more. RSA encryption and decryption are modular exponentiation functions. Classical binary exponentiation methods- left to right and right to left perform modular squaring in each iteration but modular multiplication only when exponentiation bit is one. Montgomery powering ladder has a regular structure with parallel modular squaring and modular multiplication and prevents the implementation attacks due to its regular behavior (Joye and Yen, 2002). Common multiplicand multiplication takes the advantage of parallel modular squaring and multiplication and reduces the complexity by computing once the reductions on common multiplicand. Common multiplicand Montgomery design suitable for hardware implementation is proposed in (Wu *et al.,* 2013). Their word based radix 2 and radix 4 architectures have been presented by the authors in (Wu *et al.,* 2013). Various architectures: Systolic arrays and carry save designs (McIvor *et al.,* 2004; Fournaris, 2010) for Montgomery modular multiplication (Montgomery, 1985) are in literature. Carry save designs provide the advantage of high frequency at cost of large area when implemented on FPGAs. This is due to the mapping of carry and sum bit on different LUTs. A high performance fault attack and simple power attack resistant modular exponentiation with carry save Montgomery modular multiplication is proposed in (Fournaris, 2010). It employs carry save logic in all its inputs, outputs, intermediate values and computations. It is optimized in terms of area, frequency and throughput and is attack resistant. The work in this study aims in power attack resistant efficient RSA design with low power consumption so that it is energy efficient design. To achieve it, the RSA is based on Montgomery powering ladder, carry save common multiplicand Montgomery modular multiplication. It uses 2 DSP slices for redundant to binary conversion at end of common multiplicand Montgomery modular multiplication. Section 2 gives a brief introduction to

Corresponding Author: Rupali Verma, Computer Science and Engineering, PEC University of Technology, Chandigarh, India

common multiplicand Montgomery modular multiplication. The proposed carry save common multiplicand Montgomery modular multiplication for RSA is presented in section 3. Section 4 presents its architecture. Section 5 presents the modular exponentiation for RSA based on Montgomery powering ladder and carry save common multiplicand Montgomery. Section 6 gives the implementation results and comparison with related carry save designs in literature. Section 7 concludes the paper.

2. COMMON MULTIPLICAND MONTGOMERY MODULAR MULTIPLICATION

Common multiplicand Montgomery modular multiplication takes the advantage of the common multiplicand in the modular squaring and modular multiplication and divides them into two parallel processes (Wu *et al*., 2013). Let R and P be k bit numbers, n is k bit modulus and MMM is Montgomery modular multiplication.

$$MMM(R, P, n) = R \cdot P \cdot 2^{-k} \bmod n \quad (1)$$

$$MMM(P, P, n) = P \cdot P \cdot 2^{-k} \bmod n \quad (2)$$

Equation 1 and 2 represent modular multiplication and modular squaring respectively where P is the common multiplicand. A common multiplicand approach performs the two independent operations by common modular reduction on P as in Equation 3 and two separate accumulations in Equation 4 and 5 (Wu *et al*., 2013) where r_i and p_i are i^{th} digit of R and P respectively:

$$T_i = P.2^{-i} \bmod n, \text{for } i = 1, 2, \ldots, k \quad (3)$$

$$X = \sum_{i=1}^{k} r_i P 2^{-i} \bmod n \quad (4)$$

$$Y = \sum_{i=1}^{k} p_i P 2^{-i} \bmod n \quad (5)$$

Algorithm 1. Common multiplicand Montgomery modular multiplication
Input: P and R are k+g bit numbers:
$g = 1+\lceil \log_2(k+1) \rceil$
$P = \sum_{i=0}^{k+g-1} p_i 2^i$, $R = \sum_{i=0}^{k+g-1} r_i 2^i$
Modulus, n with $2^{k-1} < n < 2^k$, gcd (n, 2) = 1
Output: X = P. R. $2^{-(k+2g)} \bmod n$, $Y = P^2 2^{-(k+2g)} \bmod n$ with $0 \le X < 2^{k+g}$, $0 \le Y < 2^{k+g}$
1: X: = 0, Y: = 0;
2: T: = P;
3: for i = 1 to k+2g do
4: q[i]: = $T_0 \bmod 2$;
5: T: = (T+ q[i] n)/2;
6: if g+1≤ i ≤ k+2g then
7: X:= X+ $r_{k+2g-i} \cdot T$, Y:= Y+ $p_{k+2g-i} \cdot T$;
8: end if;
9: end for;
10. return X, Y.

Algorithm 1 is common multiplicand Montgomery modular multiplication proposed by authors (Wu *et al*., 2013). Algorithm 2 is the proposed carry save method for common multiplicand Montgomery modular multiplication. All the intermediate addition operations of large numbers are done with carry save adders. The input operands to algorithm are in binary form. To convert the results from redundant to binary few extra cycles are required. Also it is essential to perform the conversion of result from redundant to binary at the end so that in successive common multiplicand Montgomery modular multiplication in exponentiation, the accumulation of partial products can start from most significant bit of multiplier.

3. PROPOSED CARRY SAVE COMMON MULTIPLICAND MONTGOMERY MODULAR MULTIPLICATION

Algorithm 2 takes input P, R and n, computes modular reduction on T which is initialized to the common multiplicand P. To reduce the iteration time the various steps are parallelized by making them independent computations. Step 5 performs modular reduction whereas step 8 performs accumulation. Steps (4, 5) and 8 are pipelined for parallel computation.

Algorithm 2. Proposed Carry Save Common Multiplicand Montgomery Modular Multiplication (CSCMMM)
Input: P and R are both (k+g) bit numbers with
$g = 1+\lceil \log_2(k+1) \rceil$
$P = \sum_{i=0}^{k+g-1} p_i 2^i$, $R = \sum_{i=0}^{k+g-1} r_i 2^i$
Modulus, n with $2^{k-1} < n < 2^k$ and gcd (n, 2) = 1
Output: X = P. R. $2^{-(k+2g)} \bmod n$, $Y = P^2 2^{-(k+2g)} \bmod n$ with $0 \le X < 2^{k+g}$, $0 \le Y < 2^{k+g}$
1: X1: = 0, X2:= 0, Y1: = 0, Y2: = 0;
2: T1[1]: = 0, T2[1]: = P;
3: For i = 1 to k+2g do

4: q[i]: = $(T1[i]_0 \oplus T2[i]_0)$ mod 2;
5: T1[i+1], T2[i+1]: = (T1[i]+T2[i]+q[i]·n)/2;
6: end for;
7: for i = g+2 to k+2g+1 do
8: X1,X2:= X1+X2+$r_{k+2g-(i-1)}$.(T1[i]+T2[i]);
parallel
Y1,Y2:= Y1+Y2+$p_{k+2g-(i-1)}$.(T1[i]+T2[i])
9: end for;
10: X: = X1+X2; Y: = Y1+Y2; /* conversion from redundant to binary */
11: return X, Y.

The for loop of step 3 runs for k+2g iterations with each iteration computing quotient q[i] and T1[i+1],T2[i+1]. This loop computes modular reduction on common multiplicand P and has delay of 1 XOR, 1 full adder and 2:1 MUX. The computed T1, T2 values are added in successive iteration. Therefore the accumulation of partial product starts from g+2 iteration. Hence for loop of step 7 runs from i = g+2 to k+2g+1. The multiplier bits for partial product accumulation are taken from k+g-1 to 0:

$$\text{When } i = g+2, k+2g-(i-1) = k+g-1$$

$$\text{and } i = k+2g+1, k+2g-(i-1) = 0$$

The accumulation of partial products in X1, X2 and Y1, Y2 are computed in parallel with delay of 2 full adders and 2:1 MUX.

4. ARCHITECTURE OF CARRY SAVE COMMON MULTIPLICAND MONTGOMERY MODULAR MULTIPLICATION

Figure 1 shows the architecture of carry save common multiplicand Montgomery modular multiplication. It consists of:

- I/O Interface
- Control unit
- Registers
- Counter
- Common reduction unit
- X, Y accumulation units
- Adders

The I/O interface takes three inputs P, R and n and gives two outputs X and Y in binary. The control unit controls the sequence of computations to achieve modular multiplication. Common reduction unit computes quotient and reduction on common multiplicand. Common reduction unit, X and Y accumulation units are pipelined so that common reduction and accumulation are computed in parallel. The counter keeps track of the computations. Adders convert the result from redundant to binary. The number of cycles in conversion from redundant to binary depends on the adder and its implementation.

5. RSA MODULAR EXPONENTIATION

Algorithm 3 is the modular exponentiation based on Montgomery powering ladder and common multiplicand Montgomery modular multiplication to compute M^e mod n (Wu *et al*., 2013). M is converted to Montgomery domain and R is reassigned pre-computed value Z. This is done to have one modular multiplication unit in exponentiation. If exponent bit is set then:

$R = P.\ R.\ 2^{-(k+2g)} \bmod n$, $P = P^2\ 2^{-(k+2g)} \bmod n$.

If exponent bit is zero then values are

$P = R.P.\ 2^{-(k+2g)} \bmod n$, $R = R^2\ 2^{-(k+2g)} \bmod n$.

At step 8 and 9 the result is converted to integer domain and stored in C respectively.

Hence $C = 1.\ R\ .2^{-(k+2g)} \bmod n$.

Algorithm 3. RSA modular exponentiation with Montgomery powering ladder and common multiplicand Montgomery modular multiplication

Input: $0 \le M < n < 2^k$, $e = \sum_{i=0}^{k-1} e_i\ 2^i$,

$\lambda = 2^{(2k+4g)} \bmod n$, $\delta = 2^{k+2g} \bmod n$

$Z = \delta \bmod n$ (pre-computed value), gcd (n, 2) = 1

Output: $C = M^e \bmod n$, $0 \le C \le n$.

1. P, R = CSCMMM (M, λ, n)
// to convert M into Montgomery domain
2. R = Z;
3. for i = k-1 to 0 do
4. if e_i = 1 then R, P = CSCMMM(P,R, n);
5. else P, R = CSCMMM(R, P, n);
6. end if;
7. end for;
8. P,R = CSCMMM(1, R, n);
// convert to integer domain
9. C: = P;
10. return C.

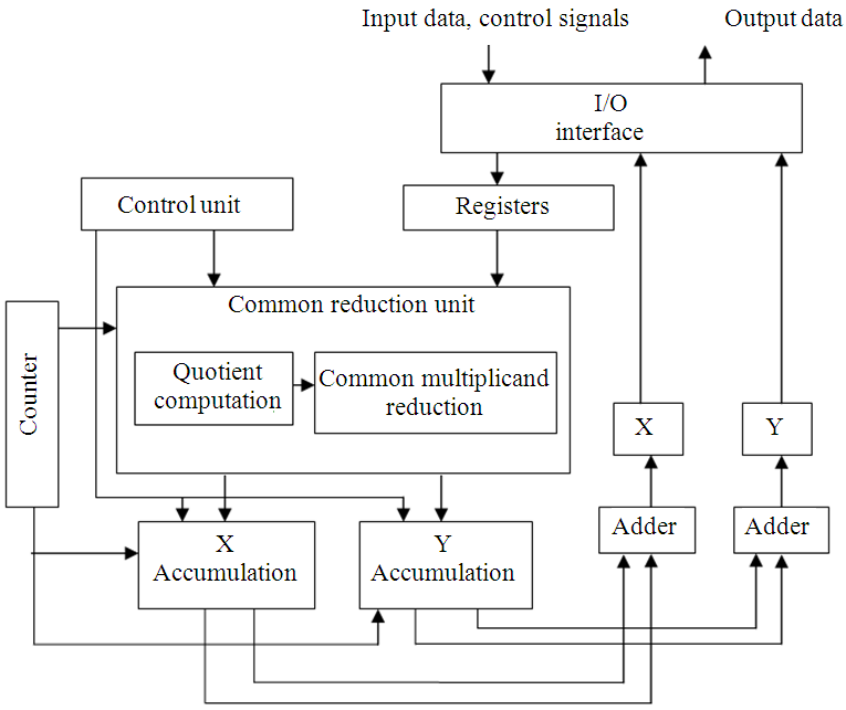

Fig. 1. Architecture of carry save common multiplicand montgomery

6. IMPLEMENTATION RESULTS

RSA modular exponentiation with Montgomery powering ladder and carry save common multiplicand Montgomery modular multiplication is coded in VHDL and synthesized in Xilinx ISE design suite 12.4. The target device is xc5vlx50t (package ff65 target speed -3). The size of operands is 1024 bits and encryption exponent is $e = 2^{16}+1$. **Figure 2** shows DSP48E chosen for addition. To convert the results from redundant to binary (algorithm 2) two DSP48E are used that work in parallel to add X = X1+X2 and Y = Y1+Y2. 48 bit operands and carry bit are taken in each cycle and added to give 48 bit result and 1bit carry out that becomes carry in for next cycle. For RSA 1024 bits the operand size in common multiplicand is 1036(1024+12) bits which requires approximately 22 cycles for addition using DSP48E.

Using IP core and architecture wizard DSP48E is selected and the instruction:

$$C + CONCAT + CARRYIN$$

Is given and CARRYOUT is selected which is shown in **Figure 2**.

Table 1 gives the number of cycles for RSA 1024 bit modular exponentiation and taking encryption exponent $e = 2^{16}+1$. For 1024 bits, g has value 12 (Wu *et al*., 2013). For 17 bit exponent the total calls for Common multiplicand Montgomery are 17+1 (from integer to Montgomery domain) +1 (Montgomery to integer domain). The total cycle count for RSA exponentiation is 20368.**Table 2** gives area results in terms of slice registers, LUTs and DSP48Es. These results are obtained from place and route report generated in Xilinx ISE 12.4. **Table 3** gives results of RSA modular exponentiation in terms of area, frequency, throughput, power and simple power attack. Throughput of the proposed RSA design is calculated by the formula Equation 6:

$$\text{Throughput} = \frac{\text{Bit length} \times \text{Frequency}}{\text{Number of Cycles}} \quad (6)$$

The power consumption is generated in Xilinx x power analyzer. The power consumption of our RSA 1024 bits is 43 mW. RSA in our work is based on Montgomery powering ladder, hence it is simple power attack resistant.

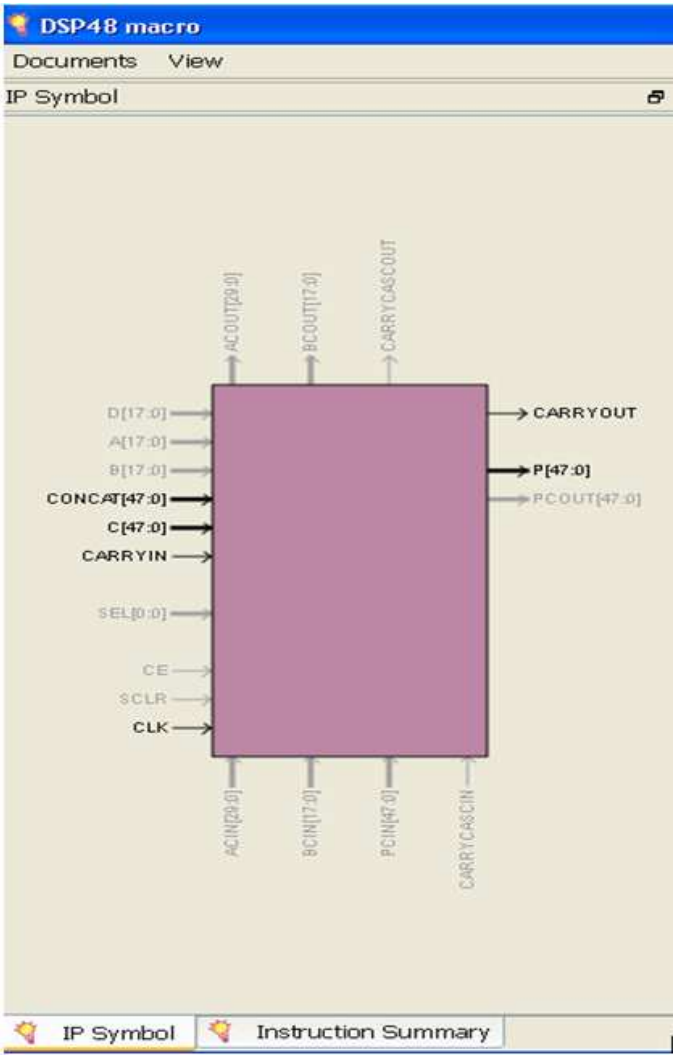

Fig. 2. DSP48E to add 48 bit data with carry in

Table 1. Cycle count in RSA 1024 bit modular exponentiation

Design	Exponentiation cycles	No of cycles in proposed Montgomery	Total cycle count
RSA	19	1050+22(addition)	20368

Table 2. Detailed Area results of RSA 1024 bit modular exponentiation (Virtex XC5VLX50T)

Slice Registers	LUTs	DSP48Es
14948	12447	2

Table 3. RSA 1024 bit modular exponentiation (Virtex XC5VLX50T)

	Area (Slices)	Freq (MHz)	Throughput (Mbps)	Power (mW)	SPA
Fournaris, 2010	7158	274.00	14.50	-	Yes
Our	4977	327.38	16.45	43	Yes

RSA exponentiation based on carry save Montgomery modular multiplication was proposed by the authors (McIvor *et al*., 2004) and its implementation results on Virtex 2 FPGAs was presented. They used carry save adders for addition of operands during modular multiplication and there was no conversion of results from carry-save to binary at end of modular multiplication. The proposed carry save common multiplicand Montgomery in this work requires a format conversion from carry save to binary at end since each successive modular multiplication in exponentiation starts the accumulation of partial products from the most significant bit of multiplier. Our proposed modular multiplication uses two DSP48E for addition. It adds 48 bits in one cycle and requires 22 cycles for addition of 1036 bits. The proposed RSA is implemented on virtex 5 FPGAs. Its implementation on virtex 2 FPGAs is not possible due to lack of DSP slices in virtex 2 FPGAs. Compared to RSA (Fournaris, 2010) which is based on carry save Montgomery modular multiplication and is attack resistant, our RSA is efficient in terms of area, frequency and throughput. Also the implementation of RSA with Montgomery powering ladder naturally protects it from many implementation attacks (Joye and Yen, 2002). The power consumption of our RSA design is very less as compared to the power consumption of 1024 bit modular multiplication in (Ye *et al*., 2013). The addition cycles in our work for redundant to binary conversion can be further reduced by using fast adders presented in (Zicari and Perri, 2010). The use of reversible logic in Montgomery modular multiplication to prevent power attacks was presented in (Nayeem *et al*., 2009). The performance of common Multiplicand Montgomery modular multiplication with reversible adder proposed in (Haghparast and Navi, 2008) can be analyzed.

7. CONCLUSION

In this study, RSA modular exponentiation based on Montgomery powering ladder and carry save common multiplicand Montgomery modular multiplication uses DSP48E to convert result from carry save to binary at end of modular multiplication. The design is efficient in terms of area, throughput and power. Also the design is power attack resistant. The throughput is inversely proportional to the cycle count. The number of cycles of carry save common multiplicand Montgomery modular multiplication can be reduced with the use of efficient adders used to convert redundant results to binary. Also area results can be improved by efficiently mapping the carry save design on FPGAs.

8. REFERENCES

Fournaris, A.P., 2010. Fault and simple power attack resistant RSA using Montgomery modular multiplication. Proceedings of the IEEE International Symposium on Circuits and Systems, May 30-Jun. 2, IEEE Xplore Press, Paris, pp: 1875-1878. DOI:10.1109/ISCAS.2010.5537879

Haghparast, M. and K. Navi, 2008. A Novel reversible BCD adder for nanotechnology based systems. Am. J. Applied Sci., 5: 282-288. DOI: 10.3844/ajassp.2008.282.288

Joye, M. and S.M. Yen, 2002. The Montgomery powering ladder. Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems, Aug. 13-15, Springer-Verlag Berlin Heidelberg, CA, USA., pp: 291-302. DOI: 10.1007/3-540-36400-5_22

McIvor, C., M. McLoone and J.V. McCanny, 2004. Modified Montgomery modular multiplication and RSA exponentiation techniques. Proceedings of the Computers and digital Techniques, Nov.18-18, IEEE Xplore Press, pp: 402-408. DOI: 10.1049/ip-cdt:20040791

Montgomery, P.L., 1985. Modular multiplication without trial division. Math. Computat., 44: 519-521.

Nayeem, N.M., L. Jamal and H.M.H. Babu, 2009. Efficient reversible Montgomery multiplier and its application to hardware cryptography. J. Comput. Sci., 5: 49-56. DOI: 10.3844/jcssp.2009.49.56

Rivest, R.L, A. Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM., 21: 120-126. DOI: 10.1145/359340.359342

Wu, T., S. Li and L. Liu, 2013. Fast, compact and symmetric modular exponentiation architecture by common-multiplicand Montgomery modular multiplications. Intergrat. VLSI J., 46: 323-332. DOI: 10.1016/j.vlsi.2012.09.002

Ye, J.H., T.W. Hung and M.D. Shieh, 2013. Energy-efficient architecture for word-based Montgomery modular multiplication algorithm. International Symposium on VLSI Design, Automation and Test, Apr. 22-24, IEEE Xplore Press, Hsinchu, pp: 1-4. DOI: 10.1109/VLDI-DAT.2013.6533882

Zicari, P. and S. Perri, 2010. A fast carry chain adder for virtex-5 FPGAs. Proceedings of the 15th IEEE Mediterranean Electrotechnical Conference on MELECON, Apr. 26-28, IEEE Xplore Press, Valletta, pp: 304-308. DOI: 10.1109/MELCON.2010.5476275