On multiple precision based Montgomery Multiplication without Precomputation of $N_0' = -N_0^{-1}$ *mod W*

H. Ploog, D. Timmermann

Department of Electrical Engineering and Information Technology
University of Rostock, Richard-Wagner-Str-31, 18119 Rostock, Germany
hp@e-technik.uni-rostock.de

Abstract

An efficient implementation of modular exponentiation, i.e., the main building block of many public key cryptographic devices, is achieved by algorithmic optimization of the Montgomery modular multiplication algorithm based on multiple precision such that pre-computation of $N_0' = - N_0^{-1}$ mod W can be avoided. This can be attained by modifications of the multiplier used.

1. Introduction

Many cryptographic schemes, such as RSA scheme [1], ElGamal scheme, etc., are based on modular exponentiation of long integers (up to 2048 bits). A widely used method to perform modular exponentiation (ME) is the 'square and multiply'-technique [2]. In 1985, P. L. Montgomery proposed an algorithm for modular multiplying $A \cdot B$ mod N without trial division [3]. In [4] different modular reduction algorithms for large integers are compared with respect to their performance and the conclusion was drawn that modular exponentiation based on Montgomery's algorithm has the best performance.

In this paper we propose a new architecture achieving low latency for the Montgomery modular multiplication algorithm, which does not require the precomputation of $N_0' = - N_0^{-1}$ *mod W* and thereby reducing the number of clock cycles necessary for modular multiplication. Hence, we also speed up modular exponentiation.

2. Montgomery multiplication

Let A, B be elements of Z_N, where Z_N is the set of integers between [0, N-1]. Let R be an integer relatively prime to N, e.g. gcd(R, N)=1, and $R > N$. Then, the Montgomery algorithm computes

$$\text{MonProd}(A, B) = A \cdot B \cdot R^{-1} \bmod N.$$

Since the algorithm works for any R being coprime to N it is more advantageous if R is a power of 2, e.g. $R=2^x$, since a division by a power of 2 is a simple shift.

For computing the Montgomery product we need two additional values, N' and R^{-1} being the modular inverse of R mod N, i.e., $R \cdot R^{-1} = 1$ (mod N). To calculate R^{-1} the extended Euclidian algorithm is used. N' is an integer with the property $R \cdot R^{-1} - N \cdot N' = 1$. The algorithm for Montgomery Multiplication is given below:

Algorithm 1: computes $u = \bar{a} \cdot \bar{b}\, R^{-1}$ mod N

```
1. $t = \bar{a} \cdot \bar{b}$
2. $m = t \cdot N'$ mod $R$
3. $u = (t + m \cdot N) / R$
4. if $u \geq N$ then $u = u - N$
```

In many cryptographic devices large intermediate results are stored in RAM instead of flip-flops to get small footprint architectures. This is often the case in smart cards or FPGA based solutions. The width w of the RAM is therefore a given design constraint [5].

Many optimizations have been reported over the last years. Dussè and Kalinski first noted that in the case of multiple-precision it is possible to use $N_0' = -N_0^{-1}$ *mod W* (N_0' and N_0 are the w least significant bits of N' and N', respectively) instead of N'[6].

Koc et al. analyzed several implementations of the Montgomery algorithm with respect to their performance on general-purpose-processors [7]. It can be shown that the Finely Integrated Product Scanning method (FIPS) is best suited for hardware implementation due to its low clock cycle count. The FIPS method interleaves the computation of $a \cdot b$ and $m \cdot n$ and requires $2s^2+s$ multiplications, with $s=\lceil n/w \rceil$. The FIPS method executes the Montgomery multiplication row-wise from the right to the left. Therefore, a $2w+1+\log_2 s$-bit wide accumulator is required

2.1 Preparing the multiplier for reduction

We now take a closer look at line 2 of algorithm 1 which stated that $m = t_0 \cdot N_0'$ mod W, with t_0 (and N_0') representing the w-lower bits of $t = a \cdot b$ (and N') and W being the RAM width. It can be observed that for W=2 (binary case) one does not have to perform a modulo reduction of $t_0 \cdot N_0'$ mod 2, since we have simple 1 bit x 1 bit multiplication which can be realized using a single AND-gate, e.g. $m_{0,0} = t_{0,0}$ AND $N_{0,0}'$. This property is widely used for implementing Montgomery multiplication using systolic arrays [8]. The system modulus N in the RSA-scheme is always odd, since N is the product of two large primes. Let N be an odd integer. Then N' is also odd since $N_{0,0} \cdot N_{0,0}' = -1 \bmod 2$.

Using this property of N' we get $m_{0,0} = t_{0,0}$. Figure 1 depicts a 4-bit Montgomery multiplier during the reduction cycle, where $t=(t_7, t_6, ..., t_0)$, $n=(n_3, n_2, n_1, n_0)$, and $u= (u_4, u_3, ..., u_0)$. The dotted lines in Figure 1 are m_j, correspondingly. Obviously, we do not need N_0' to compute the reduction. Figure 1 is showing the finally implemented multiplier.

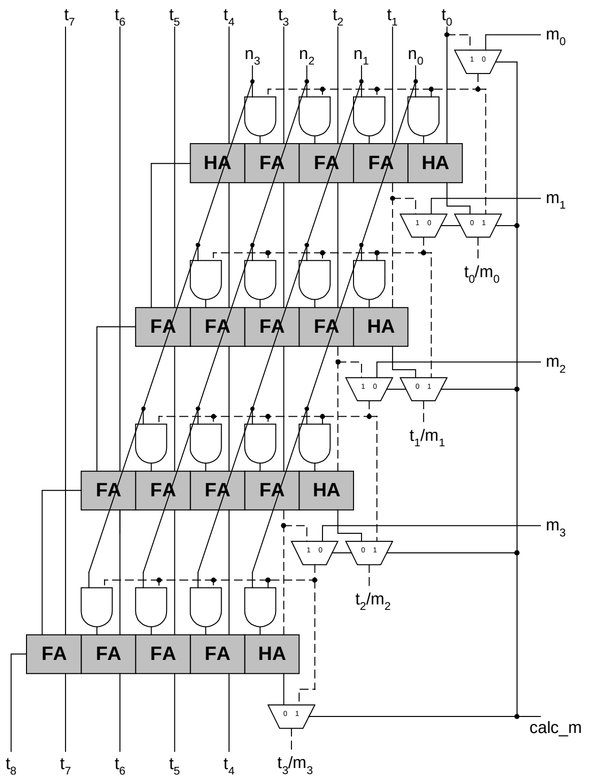

Figure 1. Proposed multiplier architecture

The additional input *calc_m* is set to one if we want to calculate m and is set to zero if we want to perform normal multiplication. By means of the proposed architecture of the optimized multiplier we do not have to precompute and store $N_0' = -N_0^{-1} \bmod W$. Obviously, this comes not for free. The price to pay is a state-machine to control the multiplier.

2.2 Evaluation and summary

A multiple-precision Montgomery multiplication using the FIPS implementation requires at least $2s^2+s$ multiplications of two $w \cdot w$ - words. Employing the optimized implementation we can reduce the number of multiplications to $2s^2$, since the calculation of $m = t_0 \cdot N_0' \bmod W$ can be avoided. The architecture proposed in this paper allows to compute 1743 modular multiplications per second (n=512, w=16, f=3.58 MHz), at least 21 times the number of modular multiplication presented in [9] .

We present a novel multiplier architecture suitable for smart card implementation to perform a modular exponentiation based on Montgomery multiplication. A VHDL description of an arithmetic co-processor for computing modular exponentiation based on the optimized Montgomery multiplication was developed. This model could be parameterized by w and N. The proposed architecture reduces the number of $w \cdot w$-bit multiplication in one Montgomery multiplication by s. However, a state-machine to control the multiplier has to be implemented. We are currently working forward to pipeline the architecture so that a higher radix W can be used. The full paper is available upon request due to the authors.

3. References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp.120-127, February 1978

[2] D. Knuth, "The Art of Computer Programming: Volume 2, Seminumerical Algorithms", 2nd edition, Addison-Wesley, 1981

[3] P. L. Montgomery, "Modular multiplication without trial division," Mathematics of Computation, Vol. 44, No. 170, April 1985, pp. 519-521.

[4] A. Bosselaers, R. Govaerts, and J. Vandewalle, "Comparison of three modular reduction functions," Advances in Cryptology – CRYPTO`93, pp. 166-174, Springer-Verlag, 1993

[5] H. Ploog, D. Timmermann, "FPGA based architecture evaluation of cryptographic coprocessors for smart cards," FCCM'98 Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, April 1998

[6] S. R. Dussé, B. S. Kaliski Jr., "A cryptographic library for the Motorola DSP56000," Advances in Cryptology – EUROCRYPT 90, pp. 230-244, Springer-Verlag, 1990

[7] C. K. Koc, T. Acar, B. S. Kaliski Jr., "Analyzing and comparing Montgomery Multiplication Algorithms," IEEE Micro, pp. 26-33, June 1996

[8] A. Tiountchik, E. Trichina, "RSA Acceleration with Field Programmable Gate Arrays," Information Security and Privacy, 4th Australasian Conference, April 1999, Proceedings, Springer LNCS 1587

[9] J.-F. Dhem, D. Veithen, J.-J. Quisquater, ”*SCALPS: Smart Card for Limited Payment Systems,*” IEEE Micro, vol. 16, no. 3, June 1996