Efficient Squaring Algorithm for Embedded RISC Processors

Feng-Fu Su, Ren-Junn Hwang and Loang-Shing Huang
Department of Computer Science and Information Engineering TamKang University
Tamsui, Taipei County, Taiwan 251, R.O.C.
E-mail:lsh876@ms27.hinet.net

Abstract*-Squaring X^2 is a special case of multiplication that plays an important role to several public-key cryptosystems such as the RSA and ECC cryptosystems. This paper proposes an efficient squaring algorithm for embedded RISC processors. In order to improve the performance, we utilize the feature (multiply/accumulate unit) of the embedded RISC processors and minimize the number of external memory accesses. Our squaring algorithm is 59-72% faster than Yang et al.′s for the range of bit-length from 1024 to 8192 by Texas Instruments TMS320C55x DSP.*

Keywords: Squaring algorithm, Exponentiation algorithm, Embedded system.

1. Introduction

Squaring, i.e. multiplying a number by itself, is the main operation in the exponentiation. There are many cryptographic methods, including RSA cryptosystem [5], elliptic curve cryptography [3], and so on, that are based on the exponentiation computation. The standard procedure for exponentiation operation requires many multiplications and squarings. The squaring takes most of the computation cost of an exponentiation operation. The times of squaring in the computation of X^e are dependent on the bit length of e. The exponent of RSA computation should be a large integer by security consideration. Therefore, squaring large integer is a key factor in the performance of many public key cryptosystems.

In squaring a large integer, i.e. $X^2 = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)_b{}^2$, many cross-product terms of the form $x_i \times x_j$ and $x_j \times x_i$ are equivalent. They need to be computed only once and then left shifted in order to be doubled. An n-digit squaring operation is performed using only $(n^2 + n)/2$ single-precision multiplications. Consequently, the squaring operation is more efficient than the multiplication operation. In 2004, Yang et al. proposed a new efficient squaring algorithm that fixes the error-indexing bug of the Guajardo-Parr squaring algorithm [2, 8]. However, their algorithm needs many external memory accesses in implementing it. This algorithm is not suitable for applying to embedded RISC processors.

In the embedded RISC processor, such as digital signal processor (DSP), the time consuming of multiply instruction is the same as load/store instruction [1]. If the squaring algorithm requires fewer multiply instructions or fewer load/store instructions, it enhances the computation performance. We improve the squaring algorithm by minimizing the number of external memory accesses to enhance the computation performance of the embedded RISC processors in the related computation. Our implementation result shows that the performance of our improved squaring algorithm is nearly 2.5 times faster in comparison with the Yang et al.′s squaring algorithm on the embedded RISC processors.

The rest of this paper is organized as follows: in section 2, we first review the Yang et al.′s squaring algorithm. In section 3, we present an efficient squaring algorithm. The details of our implementation results are described in section 4. Finally we conclude this paper in section 5.

2. Yang et al.′s squaring algorithm

In 2004, Yang et al. proposed an efficient squaring algorithm, Algorithm 1, to avoid both the improper carry handing bug of the standard squaring algorithm and the error-indexing bug of the Guajardo-Paar squaring algorithm [8].

Algorithm 1: Yang et al.′s squaring algorithm
Input: $X = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)_b$
Output: $Z = (z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b$

```
1. $(z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b \leftarrow (0, 0, \ldots, 0, 0)_b$
2. for $i = 0$ to $n-1$
2.1 $c \leftarrow 0$
2.2 for $j = i+1$ to $n-1$
2.2.1 $(c, s) \leftarrow z_{i+j} + x_j x_i + c$
2.2.2 $z_{i+j} \leftarrow s$
2.3 $z_{i+n} \leftarrow c$
3. $Z \leftarrow 2Z$
4. $c \leftarrow 0$
5. for $i = 0$ to $n-1$
5.1 $(c, s) \leftarrow z_{2i} + x_i x_i + c$, $z_{2i} \leftarrow s$
5.2 $(c, s) \leftarrow z_{2i+1} + c$, $z_{2i+1} \leftarrow s$
6. Return $Z = (z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b$
```

The algorithm computes $Z = X^2$. The capital letters, such as X, Z, represent multiple precision integers. For example, X is a multiple precision integer which can be written as an array $(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)_b$ consisting of n digits (b is the digital base, $0 \le x_i < b$). The lowercase letters, such as x, z, c, s, denote single precision integers. Yang et al. claimed that their algorithm is accurate and efficient [8]. However, we find that the steps 2.2, 2.3, 3, and 5 of algorithm 1 require many memory accesses. Memory access is time-consuming because it must switch on highly capacitive address and data buses, row and column decode logic, and data lines with a high fan-out [4]. Yang et al.′s squaring algorithm is not adopted to implement on the embedded RISC processors.

3. Our efficient squaring algorithm

In this section, we propose a new efficient squaring algorithm for the embedded RISC processor. In the embedded RISC processor, the time consuming of multiply instructions are the same as load/store instructions. Load and store instructions are more expensive than other instructions that involve just register accessing. Our improved squaring algorithm, Algorithm 2, minimizes the number of external memory accesses to enhance the efficiency of performing related operation in the embedded RISC processor.

```
Algorithm 2: Our improved squaring algorithm
Input: $X = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)_b$
Output: $Z = (z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b$
1. $(z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b \leftarrow (0, 0, \ldots, 0, 0)_b$, $(p, q, r) \leftarrow (0, 0, 0)$
2. for $i = 1$ to $n-1$
2.1 for $j = 0$ to $\lfloor (i-1)/2 \rfloor$
2.1.1 $(p, q, r) \leftarrow (p, q, r) + 2x_j x_{i-j}$
2.2 $z_i \leftarrow r$
2.3 $r \leftarrow q$, $q \leftarrow p$, $p \leftarrow 0$
3. for $i = n$ to $2n-3$
3.1 for $j = (i-n+1)$ to $\lfloor (i-1)/2 \rfloor$
3.1.1 $(p, q, r) \leftarrow (p, q, r) + 2x_j x_{i-j}$
3.2 $z_i \leftarrow r$
3.3 $r \leftarrow q$, $q \leftarrow p$, $p \leftarrow 0$
4. $z_{2n-1} \leftarrow q$
5. for $i = 0$ to $n-1$
5.1 $(q, r) \leftarrow z_{2i} + x_i x_i$, $z_{2i} \leftarrow r$
5.2 $(q, r) \leftarrow z_{2i+1} + q$, $z_{2i+1} \leftarrow r$
6. Return $Z = (z_{2n-1}, z_{2n-2}, \ldots, z_1, z_0)_b$
```

Our improved algorithm is used to implement long integer squaring on processors with a multiply/accumulate (MAC) unit [6]. Most embedded RISC processors (DSPs) feature a multiply/accumulate (MAC) unit with a word "wide" accumulator so that a certain number of products can be accumulated without loss of precision. The triple (p, q, r) of Algorithm 2 represents registers because of $2x_j x_{i-j}$ being the result of a triple-precision integer. The operations of Steps 2.3 and 3.3 is just a digit right-shift of (p, q, r). The most costly computation of Algorithm 2 is the execution of Steps 2.1 and 3.1. After finishing Step 2.1 or 3.1, we can get one result (i.e. r) and output it (Step 2.2 or 3.2). Because of only one memory access (Step 2.2 and 3.2 totally need $2n$ times) and then get one result, our proposed algorithm is faster than Algorithm 1 (Step 2.2.2 of Algorithm 1 totally needs $\frac{n(n-1)}{2}$ times).

4. Implementation results

To measure the performance of our improved squaring algorithm together with Yang et al.′s, we implemented these two algorithms on the Texas Instruments TMS320C55x DSP [7]. The DSP includes two MACs, four independent 40-bit accumulators, a 40-bit ALU, a 16-bit ALU, a 40-bit shifter, and so on. The program codes are implemented by assembler language. In the given experimental analysis, the multiplier X is from 1024 to 8192 bits long. The bit length of RSA computation should be larger than 1024 by security considerations. Numbers of CPU clock cycles for realization of these two squaring algorithm are given in Table 1. The forth column of Table 1 shows our improved squaring algorithm is 59-72% faster than Yang et al.′s for the range of bit-length from 1024 to 8192.

In other words, our improved algorithm needs only 41% of the computational cost needed by the Yang et al.′s for a 1024-bit squaring. It is noteworthy that our algorithm can significantly improve the squaring performance for the embedded RISC processors. That is to say, the speed of our improved squaring algorithm is almost 2.5 times faster than that of the Yang et al.′s.

5. Conclusion

This paper proposes an efficient squaring algorithm that is suitable for the embedded RISC processors. Our squaring algorithm is based on minimizing the number of external memory accesses. Our computational performance analysis shows that our squaring algorithm is 59-72% faster than Yang et al.′s for the range of bit-length from 1024 to 8192 on the Texas Instruments TMS320C55x family of digital signal processors. In a word, the speed of our squaring algorithm is almost 2.5 times faster than that of the Yang et al.′s. It is noteworthy that our algorithm can significantly improve the squaring performance for the embedded RISC processors.

ACKNOWLEDGEMENTS

This work was partially supported by the iCAST project sponsored by the National Science Council, Taiwan, under the grants no.97-2221-E-032-019.

References

[1] Johann Groβschädl. Roberto M. Avanzi, Erkay Savas, and Stefan Tillich, 'Energy-efficient software implementation of long integer modular arithmetic', *CHES 2005, LNCS 3659*, 2005, pp.75-90
[2] J. Guajardo and C. Paar, 'Modified squaring algorithm', Available from URL: http://citeseer.ist.psu.edu/672729.htm
[3] Koyama K, Maurer U, Okamoto, and Vanstone SA, 'New public-key schemes based on elliptic curves over the ring Z_n', *Proc. CRYPTO'91*, Santa Barbara, 1991, pp.252-266
[4] K. Roy and M.C. Johnson, 'Software design for low power', *Lower power design in deep submicron electronics, vol. 337 of NATO Advanced science institutes series*, chapter 6.3, 1997, pp.433-460
[5] R. Rivest, A. Shamir, and L. Adleman, 'A method for obtaining digital signature and public-key cryptosystems', *Commun. of ACM*, 1978, vol.21, no.2, pp.120-126
[6] S.R. Dussé and B. S. Kaliski, 'A cryptographic library for Motorola DSP 56000', *Proc. EUROCRYPT '90*, 1991, pp.203-213
[7] Texas Instruments, Inc., 'TMS320C5510', Available from URL: http://www.compactpci-systems.com/products/search/fm/id/?6812
[8] Wu-Chuan Yang, Peng-Yueh Hseih, and Chi-Sung Laih, 'Efficient squaring of large integers', *IEICE Trans. Fundamentals*, 2004, vol.E87-A, no.5, pp.1189-1192

Table 1. Numbers of CPU clock cycles for realizing two squaring algorithms

Length (bits)	Yang et al.′s (clock cycles)	Ours (clock cycles)	Speedup (%)
1024	5392	2221	59
2048	18981	6509	66
4096	70725	21197	70
8192	272516	75149	72