An Expandable Montgomery Modular Multiplication Processor

Adnan A.A. Gutub Alaaeldin A. M. Amin
Computer Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, SAUDI ARABIA

Email: *gutub@ccse.kfupm.edu.sa*

Abstract

A design for an expandable modular multiplication hardware is proposed. This design allows for cascading the hardware if larger moduli are required. The proposed design uses Montgomery modular multiplication algorithm [5].

1 Introduction

Several public-key cryptographic systems [1] make heavy use of modular multiplication. The security of these systems depends on the size of the encryption/decryption key. Larger key sizes have better security. The most popular such algorithm is the RSA [2] algorithm. Reported RSA hardware implementations, e.g. [3, 4, 5, 6, 7, 8, 9, 11, 12], are designed for fixed key sizes. If larger key sizes are needed to improve system security, the hardware must be redesigned.

The goal of this paper is to develop an expandable Montgomery modular multiplication hardware implementation where duplication of well defined bit-sliced hardware will adapt the system to larger numbers. To incorporate such flexibility, hardware and performance overheads are expected. The proposed expandable design depends mainly on a systolic multiplier used by Sauerbrey [6]. This model has been redesigned and modified for expandability.

In the following, the basic systolic multiplier is described, Montgomery product algorithm and its implementation are reviewed, and modifications for expandability are detailed.

2 The Systolic Multiplier

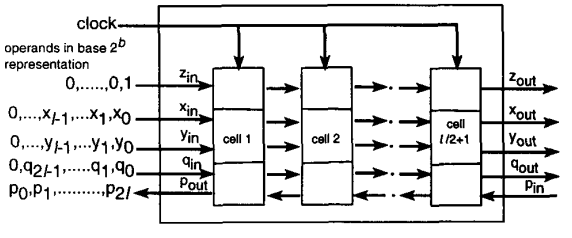

Figure 1: The word-serial multiplier (systolic array)

The systolic multiplier consists of a set of cascaded identical cells as shown in Figure 1. This multiplier performs the operation: $p = x.y + q$, in a word-serial manner, where x, y and q have l-words, of b-bits each. In other words, each of x, y and q are numbers of l-digits in base 2^b. The time required for a complete operation is $2l$ clock cycles [6]. The control input z is used to indicate the beginning of the operation.

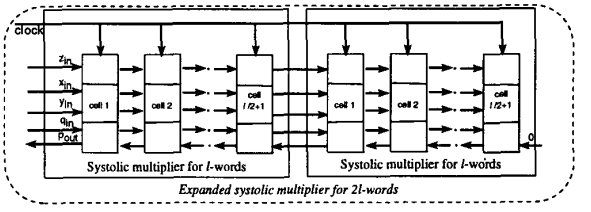

Figure 2: Expandability of the systolic multiplier

To multiply two operands of l-words, the required number of cascaded cells is $\lceil l/2 \rceil + 1$ [6]. This systolic multiplier is chosen since it can be easily expanded. Figure 1, shows a multiplier for l-digit numbers. If the numbers to be multiplied are increased in size to $2l$-digits, the only required modification on the design is to add another identical systolic multiplier in cascade, as shown in Figure 2. Clarification and modeling of each cell in the systolic multiplier is given in the following subsection.

2.1 The Basic Cell

The basic cell of the systolic multiplier is designed to perform the algorithm shown in Figure 3. Each cell consists of 4 b-bit parallel multipliers, 3 adders, 10 latches/registers, ten AND gates, two OR gates and an XOR gate. Three different size latches/registers are used in each cell (Figure 4). Single bit registers, are used to hold values of Z2 and Z1 which control the state of the cell. Seven b-bit registers are used to hold xe, ye, xo, yo, xt, yt and p. One $(b+2)$-bit register to hold S, where the extra 2-bits are required to account for the carry (S-out:=S+qin+u+p). The b-bit parallel multipliers compute intermediate products. Each multiplier uses (b^2) 2-input AND gates, (b) Half-adders, and $(b^2 - 2b)$ Full-adders.

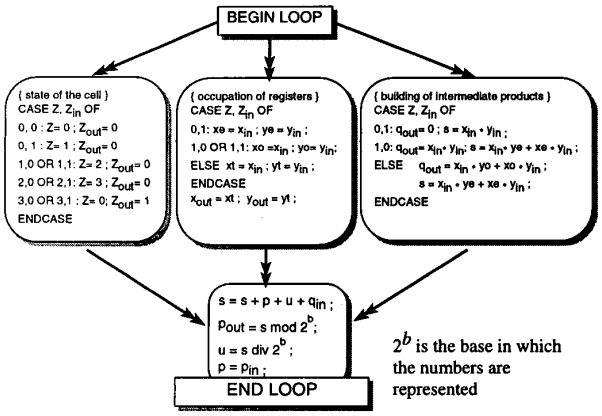

Figure 3: The algorithm of the basic cell

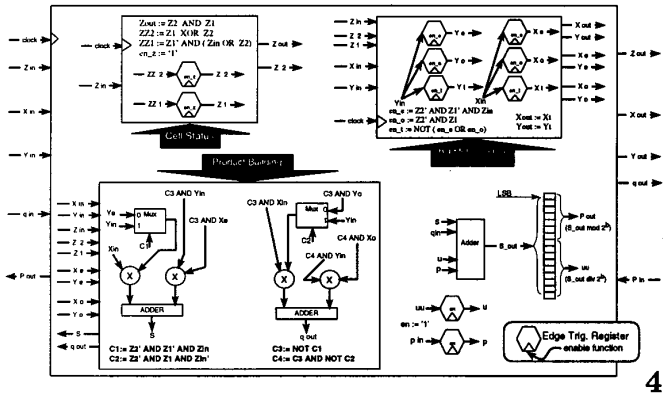

Figure 4: Hardware design of the cell

3 Montgomery Product Algorithm

Peter Montgomery [5] in 1985, came up with a method to compute modular multiplication without trial division. To compute:$z = x.y \ mod \ N$, the following steps are followed:

1. Choose $R > N$ such that $R = 2^k$, where k is the number of bits in N, accordingly R is relatively prime to N.
2. Compute R^{-1} and N' such that $RR^{-1} \ mod \ N = 1$, and $N' = -N^{-1} \ mod \ R$.
3. Transform x and y to x' and y' (Montgomery's-representation), where $x' = xR \ mod \ N$ and $y' = yR \ mod \ N$.
4. Calculate Montgomery-product: $z' = MP(x', y') = x'y'R^{-1} \ mod \ N$.
5. Transform z' to the normal representation z, i.e. $z = z'R^{-1} \ mod \ N = MP(z', 1)$.

Since they need to be computed only once, steps 1, 2 and 3 can be calculated through software. The Montgomery-product $z' = MP(x', y') = x'y'R^{-1} \ mod \ N$ is computed as follows:

1. $P = x' \times y'$;
2. $U = P + N \times (P \times N' \ mod \ R)$;
3. $S = U/R$;
4. $MP = S \ (if \ S < N)$ or $MP = S - N \ (if \ S \geq N)$;

Since R is a power of 2 number ($R = 2^k$), the $mod \ R$ and division by R operations can be inexpensively computed. However, this method is suitable only if many MP(x',y') operations are required to offset the initial cost of converting x and y to the Montgomery's representations x' and y'. This is the case with the modular exponentiations needed by the RSA algorithm. To accommodate large moduli, computing MP(x',y') is organized in a word-serial manner, as shown in Figure 5. This algorithm is well-suited for a systolic multiplier implementation similar to the one described in the previous section.

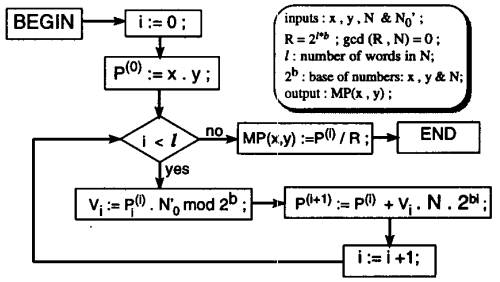

Figure 5: The MP-algorithm (Montgomery Product)

4 Montgomery Product Parallel Implementation

To implement the Montgomery product (MP) algorithm (Figure 5) using the systolic multiplier shown in Figure 1, a signal flow graph is developed to describe the flow of data. The signal flow graph for a 4-word number is shown in Figure 6.

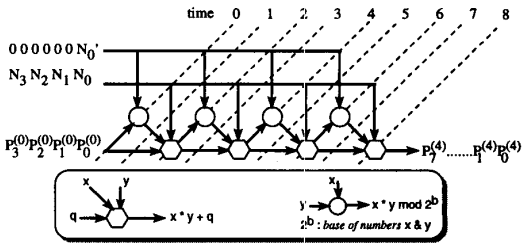

Figure 6: The signal flow graph

Two types of processors are required, one is a parallel multiplier, and the other is a systolic multiplier. The output starts coming out after $2l$ clock cycles. However, the first l-digits of the output will be discarded to account for the

division by R, and accordingly the MP result will be serially available after $3l$ clock cycles.

The hardware implementation derived from the signal flow graph (for $l = 4$ words) is shown in Figure 7. The full MP result will be available after $4l$ clock cycles. Two types of registers are used for proper data synchronization: T and 2T, which delay data by one and two clock cycles respectively. The overall number of registers required for an l-words design is $(6l - 3)$. The number of parallel multipliers used is l, while the number of systolic multipliers used is $l + 1$. The extra systolic multiplier is not shown in Figure 7, but it is necessary to compute $p^{(0)}$.

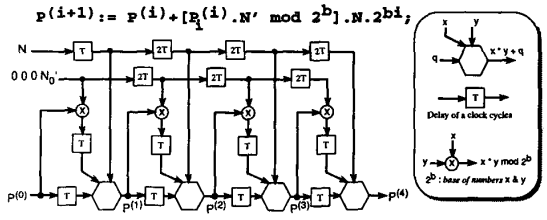

Figure 7: The signal flow graph MP implementation (parallel hardware)

4.1 Expandability of the parallel MP Implementation

To expand the design shown in Figure 7, not only should the number of systolic multipliers be increased, but also the size of each systolic multiplier must increase. This is due to the fact that the number of cascaded stages of the systolic multipliers depends on the number of words/digits (l). Thus, such design does not allow for regular linear expandability.

For example, if the design needs to be expanded to handle $2l$-words, the expansion is performed in both the horizontal and the vertical directions. The horizontal expansion is to increase the number of systolic multipliers to $2l$, while the vertical expansion increases the number of cells in each systolic multiplier to accommodate the $2l$-words. Thus, linear expandability for such architecture is not possible (Figure 8).

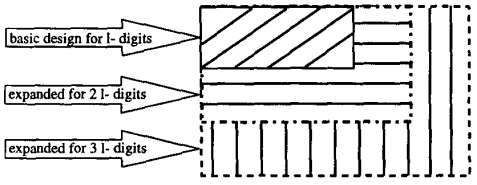

Figure 8: Expandability of the parallel implementation

5 The Expandable MP Design

For regular linear expandability of the MP implementation, the design must be reorganized for serial rather than parallel processing. This is achieved by projecting all systolic multipliers into one and projecting all parallel multipliers into one, i.e. using only one parallel multiplier and one systolic multiplier as shown in Figure 9.

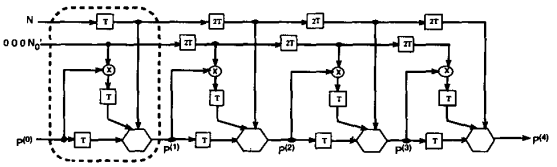

Figure 9: Projecting all multipliers into one

Consider the case where l = 4-words, and $p^{(0)}$ is precomputed, let the words of $p^{(0)}$ be fed in a word-serial manner to the projected processor and the outputs be saved in an output register, as shown in Figure 10. After $2l$ clock cycles all words of $p^{(1)}$ will be available in the output register. Then, $p^{(1)}$ is serially fed to the processor with the other inputs N and N'_0, properly synchronized. After additional $2l$ clock cycles, all digits of $p^{(2)}$ will be available in the output register. Likewise, $p^{(3)}$ and $p^{(4)}$ are computed using the same procedure allowing $2l$ clock cycles for each.

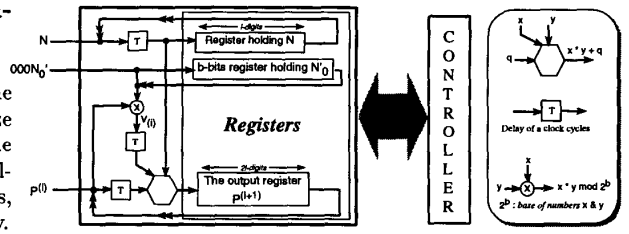

Figure 10: The expandable serial MP implementation

The time required for each $p^{(i+1)}$ to be available at the output register is $2l + 1$ clock cycles, including an extra cycle for proper synchronization. The digits of N must be synchronized to those of $p^{(i)}$ and delayed by two more clock cycles. The single word V_i (Figure 5) is calculated using a parallel multiplier to multiply N'_0 with $p_i^{(i)}$, and discarding the most significant word.

To allow for expandability, the controller is built using shift registers and multiplexors since shift registers can be easily expanded, as shown in Figure 11. The systolic multiplier is made expandable by passing its cascading signals as external interface signals as shown in Figure 2.

An expandable modular multiplier system will consist of a basic processor chip which can operate on l-digit operands, and a number of expansion chips, each allows processing of an additional l-digits. The basic MP-processor for a key size of l-digits, consists of a systolic multiplier, 9 b-bit registers, a

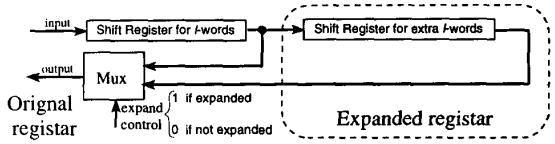

Figure 11: Expandable shift registers design

b-bit parallel multiplier, and a number of multiplexors. The expansion chip, however, does not need the b-bit parallel multiplier and uses fewer number of registers.

For a key size of l-digits, the latency time required to start getting the MP output is: $(2l+1)(l+1) = 2l^2+3l+1$ clock cycles.

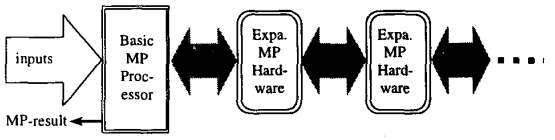

Figure 12: The expandable MP system

6 Conclusion

In this paper, a new hardware model of an expandable modular multiplication system is proposed. The new hardware depends mainly on a systolic multiplier reported by Sauerbrey [6]. Targeting the maximum possible speed, Montgomery modular multiplication algorithm has been adopted. This method computes modulo multiplication without trial division, which is the most time consuming operation in modulo multiplications.

References

[1] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, John Wiley & Sons, New York, 2nd edition, 1996.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of ACM*, vol. 21, no. 2, pp. 120-126, February 1978.

[3] Ernest F. Brickell, "A Survey of Hardware Implementations of RSA," *Advances in Cryptology-CRYPTO'89 Proceedings*, NewYork, Springer Verlog, 1990, pp. 368-370

[4] Po-Song Chen, Shih-Arn Hwang, and Cheng-Wen Wu, "A Systolic RSA Public Key Cryptosystem," *IEEE International Symposium on Circuits and Systems, ISCAS'96* , NewYork, 1996, pp. 408-411.

[5] Cetin Kaya Koc, Tolga Acar, and Burton S. Kaliski, Jr. "Analyzing and Comparing Montgomery Multiplication Algorithms," *IEEE Micro*, June 1996, pp. 26-33.

[6] Jorg Sauerbrey, "A Modular Exponentiation Unit Based on Systolic Arrays," *Advances in Cryptology, AUSCRYPT'92*, Gold Coast, Queensland, Australia, December 1992, pp. 505-516.

[7] C. N. Zhang, "An Improved Binary Algorithm for RSA," *Computers & Mathematics with Applications*, March 1993, pp. 15-24.

[8] G. Orton, M. Roy, P. Scott, L. Peppard and S. Tavares, "VLSI Implementation of Public-Key Encryption Algorithms," *in Advances in Cryptology: CRYPTO'86 Proceedings*, A. M. Odlyzko, Ed. 1987, pp. 277-301.

[9] Holger Sedlak, "The RSA Cryptography Processor," *Advances in Cryptology: EUROCRYPT'87 Proceedings*, C. Pomerance, Ed. NewYork 1988, pp. 95-105.

[10] F. Hoornaert, M. Decroos, J. Vandewalle and R. Govaerts, "Fast RSA-Hardware: Dream or Reality ?," *Advances in Cryptology : EUROCRYPT'88 Proceedings*, 1988, pp. 257-264.

[11] F. Al-Tuwaijry & S. Barton, "A High Speed RSA Processor," *Sixth International Conference on Digital Processing of Signals in Communications*, September 2-6 1991, Longhborough, UK, IEE 1991, pp. 210-214.

[12] H. Orup, E. Svendsen and E. Andreasen, "VICTOR An Efficient RSA Hardware Implementation," *Advances in Cryptology-EUROCRYPT'90: Workshop on The Theory and Application of Cryptographic Techniques*, May 21-24 1990, pp. 245-252.