A Systolic, High Speed Architecture for an RSA Cryptosystem

K.Z. PEKMESTZI AND N.K. MOSHOPOULOS
Electrical and Computer Engineering Department, National Technical University of Athens, Heroon Polytechneiou 9, 15773 Zographou, Athens, Greece

Received May 23, 2000; Revised February 15, 2001; Accepted March 2, 2001

Abstract. An architecture based on the RSA public key cryptography algorithm is presented. The circuit includes two components, one for modular squaring and one for modular multiplication. Each component is based on the Montgomery algorithm and implements the modular operations using two modified serial-parallel multipliers. A full modular exponentiation is completed every \( n(n + 3) \) clock cycles. All circuits are systolic, operate with 100% efficiency and their maximum combinational delay is equal to one gated Full-Adder. Thus, high-speed performance is achieved while the low cell hardware complexity enables an efficient VLSI implementation.

Keywords: public key cryptography, RSA, Montgomery algorithm, modular multiplication, modular squaring

1. Introduction

Bit-serial arithmetic is often used to reduce the area and the wiring to a reasonable level. This need becomes compulsory when long number operations are involved. Recent attacks on cryptography algorithms, such as the RSA [1], imply that security is achieved with bit lengths greater than 512. The basic operation of the RSA algorithm is the modular exponentiation, which in turn can be separated into modular multiplication and modular squaring. The most efficient way to perform modular multiplication is the Montgomery algorithm [2], which converts successive subtractions and comparisons to normal additions. This algorithm is suitable for hardware implementation due to the large number of arithmetic operations.

The serial-parallel multiplier is a strong candidate for long number multiplication because of its simple structure. However, it has the disadvantage that between two successive multiplications, one zero word must be inserted, reducing the operational efficiency of the circuit to 50%. The full product requires \( 2n \) clock cycles. Thus, successive multiplications can be computed every \( 2n \) cycles. Given that, the data input needs only \( n \) cycles, an \( n \)-bit padding of zeros is necessary in order to pump the high order part of the product out of the circuit. Additionally, the serial-parallel multiplier is not systolic because long lines for signal broadcasting are required. A systolic scheme was presented by Lyon [3], which truncates the result and its latency is proportional to the bit length of the multiplier. Thus, no application of the Montgomery algorithm is practical. Another architecture, which operates with 100% efficiency and retains a low delay, is presented by Even [4]. However, it requires extra hardware and is not systolic. The hardware complexity is also increased in the design presented at Jenne et al. [5], which operates with 50% efficiency.

Considering the hardware realization of the RSA scheme, based on the Montgomery algorithm, an efficient architecture is presented in Eldridge et al. [6], which suffers from a large combinational delay. The same drawback appears in Kornerup [7], while in Yang et al. [8] the hardware complexity per bit is increased due to the use of large multiplexers. A good approach is presented in Su et al. [9], with a limitation on the maximum operational bit rate.

In this paper, an architecture for modular multiplication based on Montgomery’s algorithm is presented, which includes two systolic, 100% operation efficient, serial-parallel multipliers. In this algorithm, the modular multiplication is separated into multiplication and
reduction. A modular squaring circuit is also presented with the same features. Each of the above circuits requires \( n + 3 \) clock cycles for the completion of an operation, where \( n \) denotes the bit length of the encryption process. The hardware complexity is retained low while the maximum combinational delay is equal to one gated Full-Adder. The presented schemes of the modular multiplier and squarer are combined with minor amendments, for the realization of an RSA based system, which operates at very high speed with 100% efficiency.

2. The RSA Realization

The RSA algorithm consists of successive modular exponentiations. Several algorithms have been proposed, which try to reduce the time needed for such operations. One of the most widely used is the square and multiply algorithm. This method performs modular exponentiation by combining modular multiplication and squaring. Let \( M \) and \( C \) denote the plain and the encrypted message respectively. \( E = (e_0, e_1, \ldots, e_{n-1}) \) denotes the encryption key, \( N \) is the modulus and \( Z \) is an intermediate variable. The algorithm, with a simplification in the first iteration, is described below:

\[
\begin{align*}
Z &: = M; \\
\text{If} \; e_0 &= 0 \; \text{then} \\
& \quad C := 1; \\
\text{else} \\
& \quad C := M; \\
\text{End } \{If\} \\
\text{For } i &:= 1 \; \text{to} \; n - 1 \; \text{do} \{n \; \text{is \; the \; number \; of \; bits}\} \\
& \quad Z := Z^2 \mod N; \{\text{Modular squaring}\} \\
& \quad \text{If} \; e_i = 1 \; \text{then} \\
& \quad \quad C := C \cdot Z \mod N; \quad \{\text{Modular multiplication}\} \\
& \qquad \text{End } \{If\} \\
\text{End; } \{\text{For}\} \\
\text{Return } C;
\end{align*}
\]

The square and multiply algorithm is composed of a modular squaring and a conditional modular multiplication per iteration. The squaring operation can be observed as a modular multiplication of a binary number with itself. A hardware implementation, which integrates both operations within the same circuit, namely a modular multiplier, is possible. Such a scheme, requires reduced hardware recourses, but operates with 50% efficiency. An alternative design, which is 100% efficient, can be realized by implementing the modular multiplication and the squaring operations in two separate circuits.

A projection of the square and multiply algorithm throughout the steps required for the generation of the encrypted message \( C \), is shown in Fig. 1. Depending on the value of \( e_i \), a multiplication occurs between the squaring result and the previous product, or between the previous product and 1, for \( e_i = 1 \) and \( e_i = 0 \), respectively. Thus, if \( e_i = 0 \), the multiplication is bypassed, according to the algorithm.

3. Modular Multiplication Based on the Montgomery Algorithm

An efficient algorithm regarding the realization of modular multiplication is presented in [2]. The algorithm produces a result between 0 and \( 2 \cdot N - 1 \), provided that \( 2^{n-2} \leq N < 2^{n-1} \). However, a final subtraction of \( N \) from the final result may be necessary.

The integration of the Montgomery algorithm into the square and multiply method needs extra initializations and a final transformation, which can be pre- and post-calculated without affecting the overall performance. A modification of this algorithm, is shown below:

\[
\begin{align*}
\text{(Inputs):} & \\
& \text{Modulus: } N \ (n \text{-bit integer}) \\
& \text{Multiplicand: } B \ (n + 1 \text{-bit integer}) \\
& \text{Multiplier: } A \ (n + 1 \text{-bit integer})
\end{align*}
\]