A Novel Systolic Architecture for an Efficient RSA Implementation

Nikos K. Moshopoulos and K.Z. Pekmestzi

National Technical University of Athens
Iroon Polytechneiou 9
15773 Zographou
Athens, GREECE
{nikos, pekmes}@microlab.ntua.gr

Abstract: A new systolic serial-parallel scheme that implements the Montgomery multiplier is presented. The serial input of this multiplier consists of two sets of data that enter in a bit-interleaved form. The results are also derived in the same form. The design, with minor modifications, can be used for the implementation of the RSA algorithm. The circuit yields low hardware complexity and permits high-speed operation with 100% efficiency.

1 Introduction

The core of an RSA [1] crypto-system is the modular exponentiation, which can be fragmented into a sequence of modular multiplications and squarings. These operations have to be performed in a serial pipelined way, because of the operands length (>512 bits). The most efficient algorithm for modular multiplication was presented by Montgomery [2]. One approach [3], [4] proposes a direct implementation of the Montgomery scheme by using two similar circuits: one for multiplication and one for squaring. However, it suffers from a large combinational delay. Another approach [5], [6] suggests the realization of the modular multiplication and squaring in two discrete stages: the pure product generation and the modular reduction. In this approach, the combinational delay is reduced to half, over doubling the performance.

In this paper, a new implementation of a Montgomery multiplier is presented, which is based on the direct approach achieving higher performance. The circuit is modified in an elegant way in order to realize both the modular multiplication and squaring in a bit-interleaved form. The modular exponentiation takes approximately $2n^2$ clock cycles with the minimum hardware complexity per bit, reported so far.

K. Kim (Ed.): PKC 2001, LNCS 1992, pp. 416-421, 2001

2 The Montgomery Multiplier

The Montgomery algorithm is presented below:

(Inputs)

Modulus : N (n-bits integer)

Multiplier : B (n bits integer); $B=b_{n-1}, b_{n-2}, \ldots, b_0$

Multiplicand: A (n bits integer); $A=a_{n-1}, a_{n-2}, \ldots, a_0$

(Output)

$P:=(A\cdot B\cdot 2^{-n}) \bmod N$; Modular product.

(Algorithm)

$P := 0;$

$q_0 := 0;$

for i:= 0 to n do

$P:=[(P+q_i\cdot N)/2] + b_i\cdot A;$ (1)

$q_{i+1} = P \bmod 2;$ (2)

End; {For}

Given that N is an odd number we define $N'=(N+1)/2$. Thus, (1) can be re-written as follows:

$$P=\lfloor P/2\rfloor + q_i N' + b_i A \quad (3)$$

At the ith step, the term $q_i N' + b_i A$ is computed in the circuit's upper part of Fig. 1b, while the results are shifted and accumulated in the lower part according to (3). The q_i values are derived serially during the first n cycles, while at the next n cycles the modular product P is produced. The systolic operation of this circuit requires the interleaving of the serial data b_i with zeros. Due to the internal pipelining, the feedback of q_i is delayed by two clock cycles. The zero-bit interleaving enables the synchronization of q_i with the next iteration of (3). The Montgomery product P is derived in the same bit-interleaved way. However, the idle time slot can be exploited by computing the modular product of a second number. In this manner, two modular product bits are generated in successive clock cycles without interference of their intermediate results.

In each multiplication cycle, the control line R is fed with two traveling ‘ones’, which enable the downloading of two interleaved Montgomery products into a register via a multiplexer in each cell.

The carry generated by the upper part of the (n-1)th cell must be added with the carry of the lower part, within an extra Full-Adder as shown in Fig. 1b.

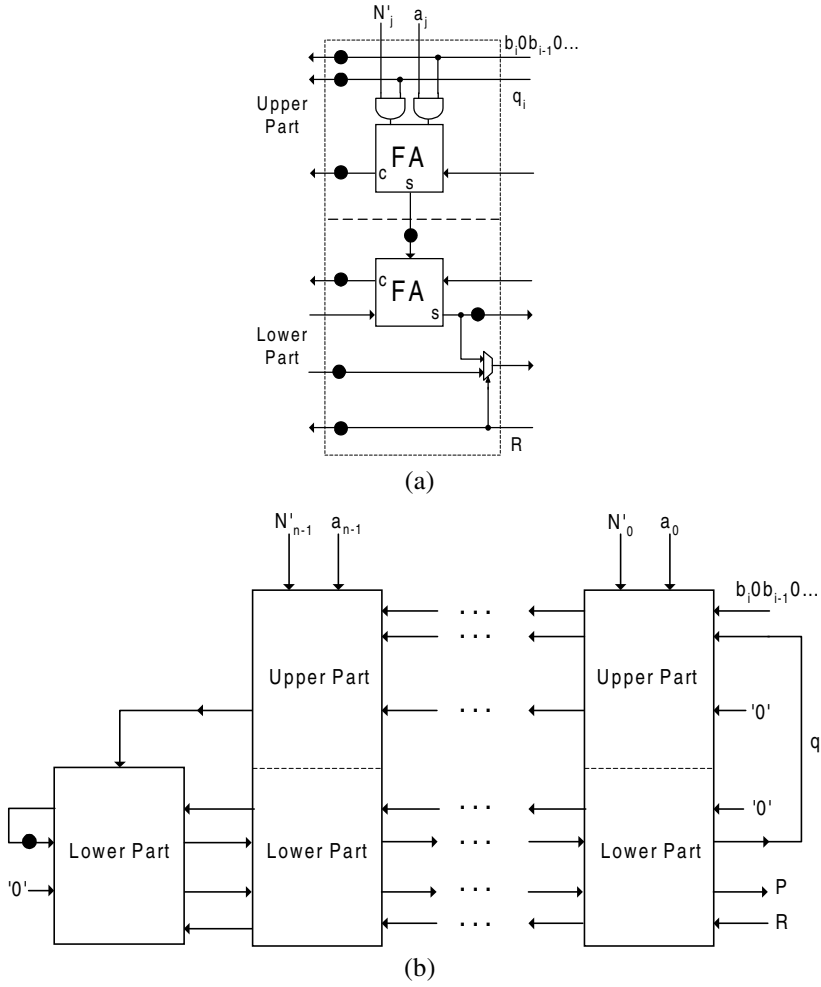

Fig. 1. (a) The basic cell (b) The proposed systolic Montgomery Multiplier

3 The Montgomery Exponentiator

The RSA algorithm can be implemented with the use of the square-and-multiply scheme.

(Inputs)

Message : M (n-bits integer)

Encryption Key : E (e_{n-1},…,e_0)

(Output)

Encrypted Message: B

(Algorithm)

A:= M; {A is an intermediate variable}

If e_0 = 0 then

B := 1;

else

B := M;

End {If}

For i:= 1 to n-1 do {n is the number of bits}

A: = $A^2 \cdot 2^{-n}$ mod N; {Mod. squaring} (4)

If e_i = 1 then

B:= $A \cdot B \cdot 2^{-n}$ mod N; {Mod. multiplication} (5)

End {If}

End; {For}

The previously presented interleaved computation of two Montgomery products in two consecutive time slots, can be of great interest regarding that, the above algorithm requires one multiplication and one squaring per iteration. The first slot can be used for the modular squaring $(A^2 \cdot 2^{-n}) mod N$ while the second for the modular multiplication $(A \cdot B \cdot 2^{-n}) mod N$. The squaring result A is produced in both serial and parallel form. The parallel form is latched and used at the next iteration as the parallel input for both operations. The latching is controlled by the R signal. The new cell is shown at Fig. 2a.

The initial value of the latches is the value of M. The P line carrying the a_i,b_i interleaved bits of $A=(A^2 \cdot 2^{-n}) mod N$ and $B=(A \cdot B \cdot 2^{-n}) mod N$ respectively, is redirected into

the serial input of the multiplier via a multiplexer, for the next iteration. This multiplexer permits the input of the initial value of B. The encrypted message is obtained after $2n^2$ clock cycles as the final value of B.

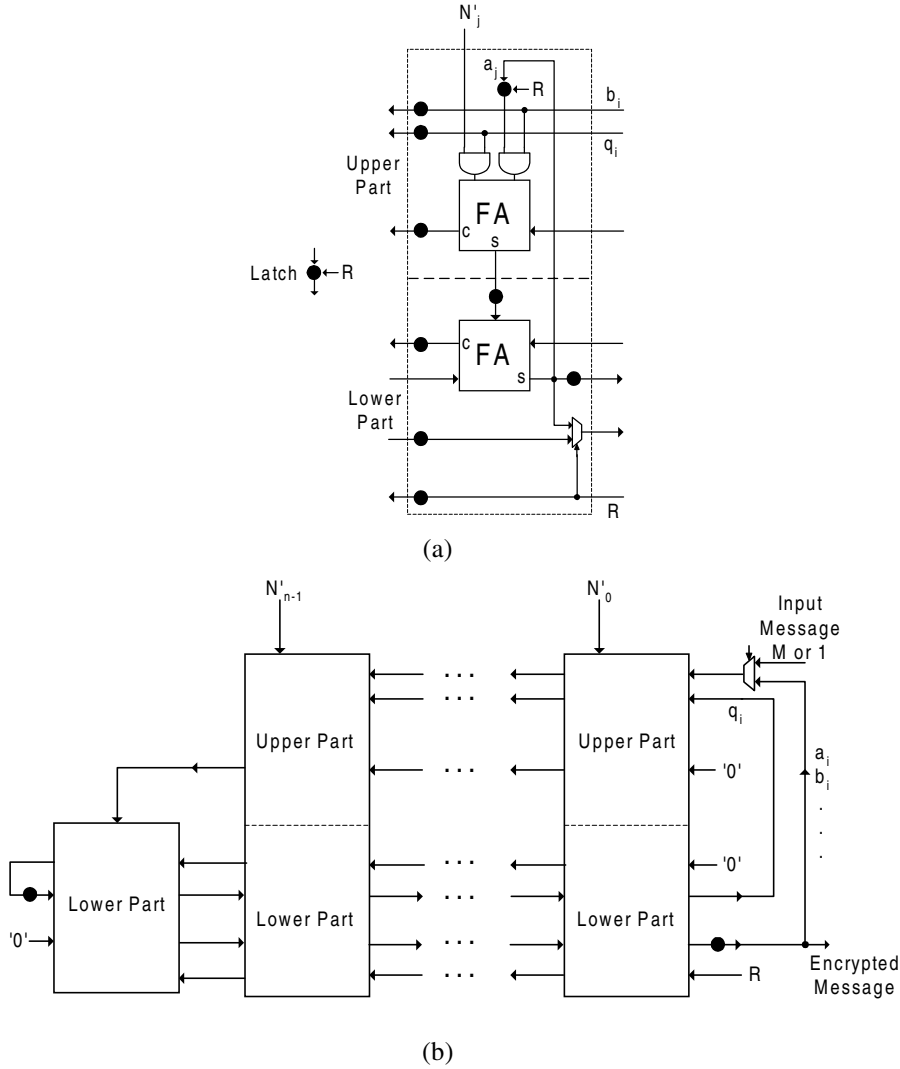

Fig. 2. (a) The basic cell (b) The proposed systolic Montgomery Exponentiator

4 Conclusions

The circuit of Fig. 2 is systolic, operates with 100% efficiency, interleaving multiplication and squaring on a bit basis, while the maximum combinational delay is equal to that of a gated Full-Adder (T_c). The utilization of the proposed design for both

squaring and multiplication, permits the application of large numbers, i.e. over 1024 bits. The critical path delay of [4] and [5] comprises two Full-Adders and some controlling logic. Therefore, it is normalized to $2T_c$. Additionally, the architecture of [6], does not include the control circuit for the RSA algorithm realization. An overall comparison in terms of hardware complexity (H), the time required for a full exponentiation (T_{exp}) and performance (C_p=H· T_{exp}), is depicted at Table 1.

Table 1. Comparison of RSA systolic arrays

Design	H*(gates)*	T_{EXP}	$C_p = H \cdot T_{EXP}$
[4]	*(4FA+8D+4G)n* = 104n	$4n^2 \cdot T_C$	$\bullet 416n^3 \cdot T_C$
[5]	*(2FA+9D+3G+8SW)n* = 117n	$4n(1.5n+2) \cdot T_C$	$\bullet 702n^3 \cdot T_C$
[6]	*(2FA+10D+4G+4SW)n* = 114n *(2FA+9D+2G+1SW)n* = 95n	$2n(n+5) \cdot T_C$	$\bullet 228n^3 \cdot T_C$
Our		$2n(n+2) \cdot T_C$	$\bullet 190n^3 \cdot T_C$

FA:Full-Adder, D:Delay Element, G:Gate, SW:Multiplexer; FA=9G, D=8G, SW=3G

The proposed design is approximately 2 and 3 times more efficient than [4] and [5] respectively. Compared to [6], our circuit's performance is about 20% higher. This is due to the direct implementation of the Montgomery algorithm, which yields a decrease of the circuit's complexity, equal to 19 gates per bit.

References

1. RIVEST, R. L., SHAMIR, A., ADLEMAN, L.: "A method for obtaining digital signature and public-key cryptosystems", Commun. ACM, 1978, VOL. 21, pp. 120-126.
2. MONTGOMERY, P. L.: "Modular multiplication without trial division", Math. Computation, 1985, VOL. 44, pp.519-521.
3. ELDRIDGE, S.E., WALTER, C. D.: "Hardware Implementation of Montgomery's modular multiplication algorithm", IEEE Trans. On Computers, 1993, VOL. 42, NO 6, pp. 693-699.
4. KORNERUP, P.,: "A systolic, linear-array multiplier for a class of right-shift algorithms", IEEE Trans. On Comput., 1994, VOL. 43, NO. 8, pp. 892-898.
5. YANG, C., CHANG, T., JEN, C.: "A new RSA cryptosystem hardware design based on Montgomery algorithm", IEEE Trans. On Circuits and Systems II, 1998, VOL. 45, NO 7, pp. 908-913.
6. C. Y. SU, S. A HWANG, P. S. CHEN, C. W. WU, "An improved Montgomery's algorithm for high-speed RSA Public-Key cryptosystem" IEEE Transactions on VLSI Systems, 1999, VOL. 7, no. 2, pp. 280-284.