Systolic Modular Multiplication

*Shimon Even**

Bellcore
445 South St.
Morristown, NJ 07960-1910

Abstract. *A simple systolic array for achieving the effect of modular reduction, in linear time, is described. This circuit, in conjunction with Atrubin's multiplier, performs modular multiplication in linear time.*

1. Introduction

With the increasing interest in using number theoretic cryptographic systems, the quest for fast and inexpensive circuits for performing modular multiplication of long integers continues. It is possible to perform integer multiplication, as well as dividing by a fixed modulus, in logarithmic time. However, the corresponding circuits tend to be complex.

A systolic array, for performing multiplication of long integers, was proposed by Atrubin [A] over 25 years ago. The two positive integers to be multiplied are represented in binary. They are fed serially to the first cell of the array, least significant bit first. The product is supplied serially by the first cell, least significant bit first, without delay. Thus, the time required to get the product is linear. The structure of the cell of Atrubin's array is very simple, and it uses no long-distance communications; i.e. each cell (a finite automaton) communicates only with its neighbors. Thus, very high clock rates are possible.**

I propose a systolic array, with similar characteristics, which performs, in effect, modular reduction. This array is based on a modular reduction system proposed by Montgomery [M].

The design of a chip for performing number-theoretic cryptographic operations, based on these two systolic arrays, is simple, and allows very high clock rates. I believe that these two advantages make such a chip competitive with currently known designs;

---

* On leave from Computer Science Dept., Technion, Haifa, Israel 32000. Csnet: even@cs.technion.ac.il

** Note that if the two multiplicands are of length n, the product is of length $2 \cdot n$, and no pipelining is possible.

see, for example, the relevant abstracts from EuroCrypt 90 [DK, OSA].

2. Montgomery's modular system

Let N be an odd integer, the *modulus*. We wish to be able to perform modular multiplications quickly. Montgomery suggested a system which avoids "regular" division, and replaces it by an operation which requires less time. We describe the system in the binary case*.

Assume all integer representations are in binary; i.e. the radix is 2. Let n be the number of bits in the representation of N; i.e. $2^{n-1} < N < 2^n$. Let $R = 2^n$. For every $x \in \mathbf{Z}_N$ let $\bar{x} \equiv x \cdot R \pmod{N}$. $\bar{x}$ is called the *image* of x. Throughout, a representation of an image has at most n bits; it may exceed N, but is nonnegative.

The idea is to do all the modular operations with images. Thus, given $\bar{x}$ and $\bar{y}$, images of x and y, respectively, we want to compute $\bar{z}$, where $z \equiv x \cdot y \pmod{N}$. However, by multiplying $\bar{x}$ and $\bar{y}$, we get

$$\begin{aligned} T &= \bar{x} \cdot \bar{y} \\ &\equiv x \cdot y \cdot R^2 \pmod{N} \\ &\equiv \bar{z} \cdot R \pmod{N}, \end{aligned}$$

and we need to divide T by R in order to get $\bar{z}$. This division is meaningful, in the ring of integers modulo N, since R and N are relatively prime. Note that there are at most $2n$ bits in the representation of T, and we want the representation of $\bar{z}$ to be nonnegative and have at most n bits. Observe that division by R, modulo N, is also useful for returning from an image to its source.

For a given binary representation K of an integer, let K_0 be its least significant bit.

The computation of $T \cdot R^{-1} \bmod N$ is performed by the following procedure:

```
begin
    M ← T;
    for i=1 to n do
    begin
        M ← M + N·M_0;
        Shift-right M, one bit
    end
    if M ≥ 2^n then M ← M − N
end
```

* The system is slightly modified, following a suggestion of Ami Litman, in order to simplify its description.

Observe that since N is odd, the addition of $N \cdot M_0$ to M, causes the new M_0 to be 0, and this 0 is then shifted away, affecting a division by 2. Denote by $M_0^{(i)}$ the least significant bit of M, just before the i-th application of the loop. It follows that the value of M, upon the termination of the inner loop, satisfies

$$M = \tfrac{1}{2}(\cdots\tfrac{1}{2}(\tfrac{1}{2}(T + M_0^{(1)} \cdot N) + M_0^{(2)} \cdot N)\cdots + M_0^{(n)} \cdot N).$$

Thus,

$$\begin{aligned} T &= 2(\cdots(2(2 \cdot M - M_0^{(n)} \cdot N) - M_0^{(n-1)} \cdot N)\cdots) - M_0^{(1)} \cdot N. \\ &= 2^n \cdot M - N \cdot (M_0^{(n)} \cdot 2^{n-1} + M_0^{(n-1)} \cdot 2^{n-2} + \cdots + M_0^{(1)}). \end{aligned} \tag{1.1}$$

and

$$T \equiv R \cdot M \pmod{N},$$

or,

$$M \equiv T \cdot R^{-1} \pmod{N}.$$

Further, let us show that equation (1.1) indicates the number of bits in the representation of M. Denote

$$\mu \triangleq M_0^{(n)} \cdot 2^{n-1} + M_0^{(n-1)} \cdot 2^{n-2} + \cdots + M_0^{(1)}.$$

Now, equation (1.1) can be rewritten as

$$M \cdot 2^n = T + N \cdot \mu.$$

Since $\mu < 2^n$ and $T < 2^{2n}$,

$$M \cdot 2^n < 2^{2n} + N \cdot 2^n,$$

or

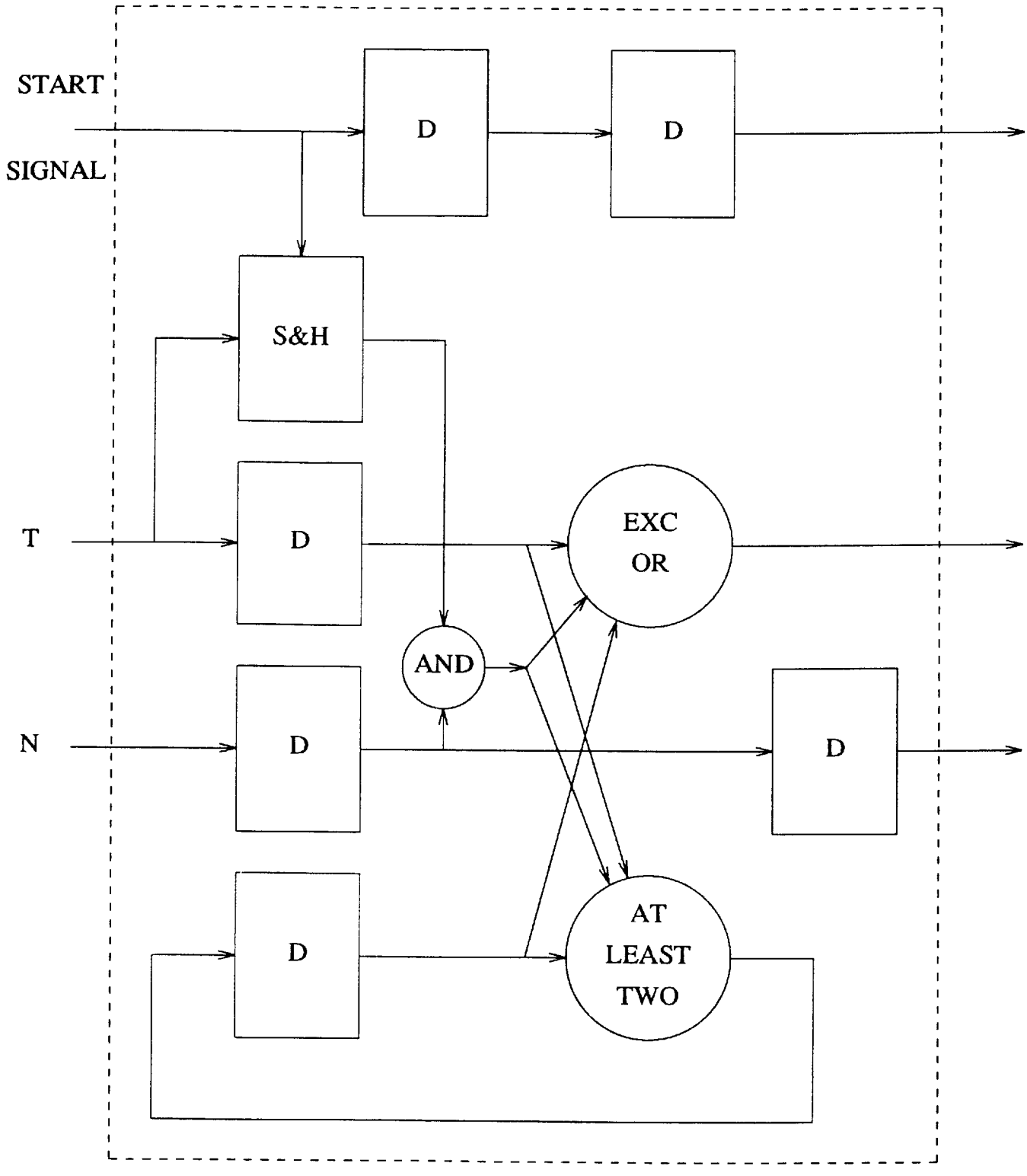
$$M - N < 2^n.$$

If $M < 2^n$, then no correction of its representation is necessary. Otherwise, a subtraction of N leaves it positive, and its representation uses at most n bits. This completes the proof of validity of the procedure.

3. A Systolic Array for $T \cdot R^{-1} \pmod{N}$

The systolic array for computing $T \cdot R^{-1} \pmod{N}$ consists of n identical cells, $C_1, C_2, \cdots, C_n$. The design of the cell is depicted in the diagram. The inputs to C_1 are supplied from the outside, and for every $1 \le i < n$, the outputs of C_i are the corresponding inputs to C_{i+1}. The T-output line of the last cell, C_n, is the output of the whole array.

The D boxes are *delay flip-flops*; i.e. their output at time $t+1$ is equal to their input at time t. There is a global clock whose pulses reach each of the flip-flops, but its lines and ports are not shown in the diagram.



Typical cell of systolic array

for computing T/R mod N.

The *start signal*, as seen by C_1, is the sequence $1 \cdot 0^{2n-1}$, i.e. a 'one' pulse followed by $2n-1$ 'zero' pulses. Clearly, C_i gets the 'one' at time $2 \cdot i - 1$.

The *sample and hold* flip-flop, S&H, samples the input entering from the l.h.s., when it gets the 'one' from the top, and holds it. All flip-flops are set to produce a 'zero'

output, at time $t=1$, by a *reset* signal which reaches them; the reset lines and ports are not shown in the diagram.

The EXC-OR (exclusive or) circuit and the AT-LEAST-TWO circuit, in conjunction with the lower-left delay flip-flop, constitute the serial adder function of the cell; the current bit of the sum $(M + N \cdot M_0)$ is the output of the EXC-OR, the carry bit is the output of the AT-LEAST-TWO -- it is delayed by lower-left delay flip-flop. The adder adds the two numbers which appear serially and simultaneously, least significant bit first, on the T and N lines. If the first bit on T, sampled at time $2 \cdot i - 1$, is 'one', then starting from time $2 \cdot i$, the AND gate is enabled. In this case, the number appearing on the N line goes through the enabled AND gate and is added serially to the number appearing on the T line. If the first bit on T is 'zero' then the AND gate remains disabled and the number on the T line goes through the adder unchanged, since the disabled AND gate blocks the number on the N line from entering the adder. In either case, the first output bit on the T line appears at time $2 \cdot i$ and is always 'zero'. However, it is ignored by C_{i+1}, which samples its input T line at time $2 \cdot i + 1$, thus affecting a shift-right operation on the output of C_i. The bits of N are delayed twice, and thus the least significant bit of N appears at the output together with the least significant bit of the number on the T line, after its "right-shift".

Thus, the cell performs the operations of one instance of the inner loop of the algorithm, and a systolic array of n cells performs all its iterations. The remaining operation, of subtracting N, if necessary, is not shown. It requires storing all $n+1$ bits of M, and going through a one-cell subtractor, if necessary. The total time to do a modular multiplication consists, therefore, of the following addends: Feeding the two multiplicands to Atrubin's array, the first output bit comes out immediately (or one tick later, depending on the particular design). The last bit of output, from the systolic array proposed here, comes out $3 \cdot n$ ticks later. Subtracting N, if necessary, can then start without further delay, and the final result can already be fed back to Atrubin's multiplier, for the next step of the modular exponentiation. Thus, the number of clock ticks per one modular multiplication is, effectively, $3 \cdot n$.

Remark

Ami Litman has shown me a method for removal of the need to store the $n+1$ bits before starting the subtraction. Using his method, effectively, one modular multiplication takes $2n+4$ ticks, instead of $3n$.

4. Acknowledgments

I would like to thank Yacov Yacobi for getting me interested in the subject, Ami Litman for useful discussions, Arjen K. Lenstra and Richard F. Graveman for commenting on an earlier version.

5. References

[A] Atrubin, A. J., "A One-Dimensional Real-Time Iterative Multiplier", *IEEE Tran. on Electronic Computers*, Vol. 14, 1965, pp. 394-399.

[DK] Dusse, S. R., and B. S. Kaliski Jr., "A Cryptographic Library for the Motorola DSP 56000", *EuroCrypt 90 - Abstracts*, May 21-24, 1990, Scanticon, Arhus, Denmark, pp. 213-217.

[OSA] Orup, H., Svendsen, E., and E. Andreasen, "VICTOR, An Efficient RSA Hardware Implementation", *EuroCrypt 90 - Abstracts*, May 21-24, 1990, Scanticon, Arhus, Denmark, pp. 219-227.

[M] Montgomery, P. L., "Modular Multiplication Without Trial Division", *Math. of Computation*, Vol. 44, 1985, pp. 519-521.