A Programmable VLSI Architecture for Computing Multiplication and Polynomial Evaluation Modulo a Positive Integer

ERL-HUEI LU, LEIN HARN, MEMBER, IEEE, JAU-YIEN LEE, MEMBER, IEEE, AND WEN-YIH HWANG

Abstract—A programmable VLSI architecture with regular, modular, expansible features is designed in this correspondence for computing $AB \bmod N$, $AB + C \bmod N$, and polynomial evaluation modulo N. The size of the resultant circuit can be easily expanded to improve the security of cryptosystems without making any change to its control circuit. Furthermore, the computing procedures for all N throughout the range $0 < N < 2^{n-1}$ are identical, therefore the new circuit is well-suited for those systems in which the value of N is alternated frequently.

I. Introduction

Modular multiplications, exponentiations, and the polynomial evaluations have been widely used in cryptographic systems [1]. Modular exponentiations and polynomial evaluations can be realized by employing multiplication operations iteratively. Several multiplier architectures have been proposed for computing multiplications over $GF(2^m)$ [2], [3] or a finite ring of integers modulo a Fermat number [4]. However, these multipliers are not suitable for many public-key cryptosystems due to their lack of security [5], [6].

In 1982, Brickell [7] developed a fast modular multiplication algorithm which multiplies in $n+10$ steps. Since the delay-carry adder is used in his algorithm, the clock rate can be greatly improved. His algorithm, however, must take two steps, $D = A(2^rB) \bmod (2^rN)$ and $D/2^r = AB \bmod N$, to compute $AB \bmod N$, where the integer N is in the range $2^{n-1} \leqslant 2^rN < 2^n$. This will make some inconvenience for those cryptosystems in which the bit number of N is alternated frequently, such as in [8] and [9]. In this correspondence, Brickell's multiplication algorithm is modified for VLSI implementation. The proposed circuit can be programmed for computing $AB \bmod N$, $AB + C \bmod N$, and polynomial evaluation modulo N in one straightforward computing step, where N is any integer in the range of $0 < N < 2^n$. Furthermore, the new multiplier is regular, modular, and therefore, well-suited for VLSI implementation [10]. In addition, the size of the multiplier can be easily expanded to improve the security of cryptosystems without making any change to its control circuit. To illustrate the detailed operations of the multiplier, a timing diagram for this design is also described.

II. The VLSI Multiplier

In this section, an iterative algorithm is formulated to evaluate $E = AB \bmod N$ first. This algorithm is similar to Brickell's [7] algorithm, but it only needs n steps for computing $E = AB \bmod N$ and can be used straightforwardly throughout the range $0 < n < 2^n$. Finally, a VLSI architecture for a serial-in–serial-out multiplier is designed and its operation is illustrated in detail in this section.

A. The Algorithm for Computing $AB \bmod N$

Assume E, A, B, and N are all n-bit binary positive integers and A is restricted in the range $0 \leqslant A < N$. The binary representation of B is

$$B = b_{n-1}2^{n-1} + \cdots + b_1 2 + b_0$$

where

$$b_i = 0 \text{ or } 1, \qquad \text{for } 0 \leqslant i \leqslant n-1.$$

Therefore the multiplication operation $E = AB \bmod N$ can be rewritten as

$$E = \left[(Ab_{n-1}2^{n-1}) + \cdots + (Ab_1 2) + (Ab_0)\right] \bmod N. \quad (1)$$

In the beginning, we set $E_0 = 0$. An iterative procedure for computing (1) can be formulated as follows. First compute

$$E_1 = [(E_0 + Ab_{n-1}) \bmod N]2.$$

Then compute

$$E_2 = [(E_1 + Ab_{n-2}) \bmod N]2$$
$$\vdots$$
$$E_{n-1} = [(E_{n-2} + Ab_1) \bmod N]2.$$

Finally

$$E_n = (E_{n-1} + Ab_0) \bmod N = E.$$

From the above procedure, it is evident that one needs to evaluate $E_{i+1} = [(E_i + Ab_{n-i-1}) \bmod N]2$ iteratively. However, since we have $0 \leqslant E_i = [(E_{i-1} + Ab_{n-i}) \bmod N]2 < 2N$ and $0 \leqslant A < N$, the inner sum $E_i + Ab_{n-i-1}$ is restricted to be in the range $0 \leqslant E_i + Ab_{n-i-1} < 3N$. Therefore, the result of the modular addition $(E_i + Ab_{n-i-1}) \bmod N$ can be divided into three different cases as

$$(E_i + Ab_{n-i-1}) \bmod N = \begin{cases} E_i + Ab_{n-i-1}, & \text{if } 0 \leqslant E_i + Ab_{n-i-1} < N. \\ E_i + Ab_{n-i-1} - N, & \text{if } N \leqslant E_i + Ab_{n-i-1} < 2N. \\ E_i + Ab_{n-i-1} - 2N, & \text{if } 2N \leqslant E_i + Ab_{n-i-1} < 3N. \end{cases} \quad (2)$$

To realize the above operation in hardware, a parallel adder is used for computing $E_i + Ab_{n-i-1}$; then the binary two's complement method [11, pp. 190–193] is used to subtract N or $2N$ from the sum of $E_i + Ab_{n-i-1}$. The subtraction operations can be implemented concurrently by using two parallel adders. Finally, the carriers generated from the most significant bits of these two adders are used to choose the appropriate result in (2). The result is then multiplied by 2 using a left-shift operation to obtain the intermediate product E_{i+1}. From $0 \leqslant E_i + Ab_{n-i-1} < 3N$, we know that a $(n+2)$-bit multiplier size is needed for the n-bit integer computation.

Manuscript received June 25, 1987; revised September 21, 1987. This work was supported by the National Science Council, Republic of China under Contract NSC 76-0404-E006-07.

E.-H. Lu, J.-Y. Lee, and W.-Y. Hwang are with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, Republic of China.

L. Harn was with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, Republic of China on leave from the Computer Science Department, University of Missouri, Kansas City, MO 64110.

IEEE Log Number 8718250.

0018-9200/88/0200-0204$01.00 ©1988 IEEE

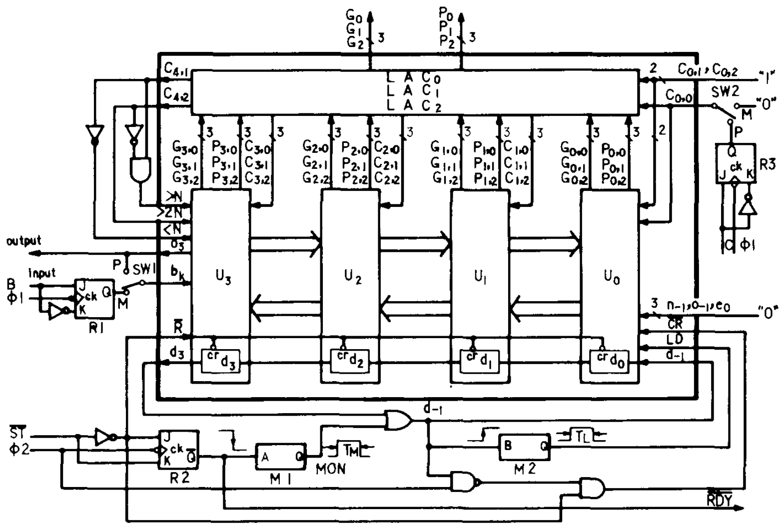

Fig. 1. Connection diagram for computing $AB \bmod N$, $AB + C \bmod N$, and polynomial evaluation modulo N.

B. The VLSI Architecture for Computing $AB \bmod N$

From the previous discussion, it is evident that a circuit to iteratively compute $E_{i+1} = [(E_i + Ab_{n-i-1}) \bmod N]2$ is needed in the hardware realization of the multiplier. An overall VLSI architecture for a 4-bit multiplier with the control circuit is shown in Fig. 1. This multiplier can be used for the 2-bit integer computation. It contains four operational cells, three lookahead carry generator (LAC) cells, and a control circuit. The detailed circuit for each operational cell U_i is shown in Fig. 2 while the circuit for the LAC cell can be found in [11, pp. 205–215]. Those LAC's provide carry lookahead capability for three 4-bit parallel adders in the 4-bit multiplier. Expanding the size of this multiplier is analogous to the expanding of a lookahead carry adder. In Fig. 1, P_i and G_i denote the group propagate and group generate outputs of LAC_i, respectively. $P_{i,j}$, $G_{i,j}$, and $C_{i,j}$ denote the propagate variable, generate variable, and input carry of adder unit j in the operational cell i, respectively. It is not too difficult to understand the functions of other connections in the bold rectangular block when compared with the operational cell circuit. As mentioned before, the multiplier can also be used for computing $AB + C \bmod N$ or evaluating polynomial modulo N. The switches $SW1$ and $SW2$, shown on the left and right side of Fig. 1, respectively, are used for function selection. They are at point M when computing $AB \bmod N$.

In this 4-bit VLSI circuit, right after A and N shifted into their respective registers a_j and n_j, and the registers e_j (which are used for storing the intermediate products E_i) are cleared, the computation of the modular multiplication starts and will be accomplished within four iterative steps. During each iteration, the sum $E_i + Ab_{4-i-1}$ will be evaluated first by the adder 0. Later on the differences $(E_i + Ab_{4-i-1}) - N$ and $(E_i + Ab_{4-i-1}) - 2N$ will be evaluated concurrently by adders 1 and 2. Since the binary two's complement method is used for implementing subtractions, not only the complement of each bit in registers n_j should be connected to adders 1 and 2 for subtract N and $2N$ respectively, but also the input of n_{-1} must be connected to ZERO state (when N is loaded) and the carry inputs $C_{0,1}$ and $C_{0,2}$ are connected to ONE state. The carries from the most significant

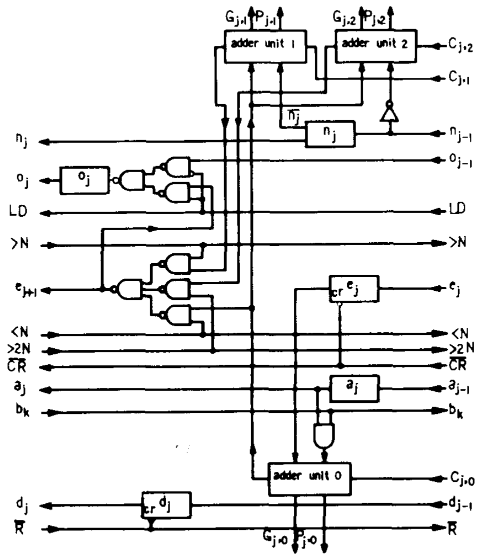

Fig. 2. Circuit for each operation cell U_j shown in Fig. 1.

bit of these two adders, $C_{4,1}$ and $C_{4,2}$, are decoded to form three control signals ($< N$, $> N$, and $> 2N$) and will be used to decide which is the correct value of (2). The result of (2) is left-shifted to double its value and thus we have the intermediate product E_{i+1}. To realize the shift operations, 0 must be loaded into the e_0 register simultaneously. Therefore the input of e_0 is connected to ZERO state. The above operations can be completed within one clock cycle.

After iterating the same procedure four times (the last iteration is without the left-shift operation), the control signal LD is high. The final product E is loaded into output registers $\underline{o_j}$. Then E is serially shifted out at the same clock rate. The $\overline{CR}$ signal is derived from the control circuit to clear the e_j registers so that the multiplication can be executed continously.

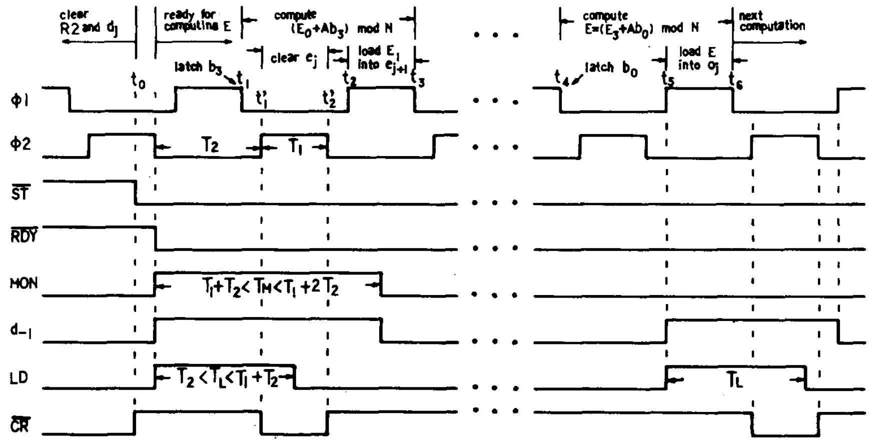

Fig. 3. Timing diagram.

As shown in Fig. 1, the four-stage shift registers d_j serve as a ring counter and the control signals LD and $\overline{CR}$ are periodic pulses with a period of four clock cycles. One remarkable feature for this kind of design is that the external control circuit is always the same for different sizes of multiplier, i.e., the multiplier can be expanded without changing the control circuit.

To clarify the whole operational procedure, a timing diagram is included in Fig. 3. The timing scheme of this design is based on two-phase nonoverlapping clocks, namely $\phi 1$ and $\phi 2$. In this multiplier design, when $\phi 1$ is high, the master flip-flops (FF) in registers e_j and o_j and all slave FF in d_j are enabled. Conversely, when $\phi 2$ is high, all master FF in d_j and all slave FF in e_j and o_j are enabled. Once A and N are shifted serially into their corresponding registers, the circuit starts to compute $AB \bmod N$. The computing procedures are listed as follows.

1) Before t_0: Since $\overline{ST}$ (command of start) is high, register 2 ($R2$) and all registers d_j as shown in Fig. 1 are cleared.

2) At t_0: $\overline{ST}$ is forced to be low at t_0 by other systems, such as a general-purpose computer. $\overline{RDY}$, the output $\overline{Q}$ of $R2$, will switch to low after the first falling edge of $\phi 2$, indicating that the multiplier is ready for computing $E = AB \bmod N$. Data b_3, b_2, b_1, and b_0 (since $B < N < 2^2$ in general, both b_3 and b_2 equal zero) can then be shifted from other system to register 1 ($R1$) with the same rate as $\phi 1$. The signals MON, d_{-1}, LD, and $\overline{CR}$ are then generated by the control circuit. In the control circuit, $M1$ and $M2$ are denotations of monostable multivibrators 1 and 2, respectively.

3) Between t_1' and t_2': $\overline{CR}$ signal is low to clear registers e_j.

4) At t_1: Latch b_3 in $R1$.

5) Between t_1 and t_3: Compute $(E_0 + Ab_3) \bmod N$. The carries $C_{4,1}$ and $C_{4,2}$ from LAC's of adders 1 and 2 are encoded to form the three control signals $< N$, $> N$, and $> 2N$. Then the correct value $0 \leqslant (E_0 + Ab_3) \bmod N < N$ is extracted.

6) Between t_2 and t_3: Load the value $(E_0 + Ab_3) \bmod N$ into the master FF of e_{j+1} in the next stage (left shift) to form the intermediate product E_1.

7) Between t_3 and t_4: Repeat steps 4–6 for b_2 and b_1.

8) At t_4: Latch b_0.

9) Between t_4 and t_6: Compute the final product $E = (E_3 + Ab_0) \bmod N$.

10) Between t_5 and t_6: Load the final product E into the master FF of output registers o_j.

11) After t_6: Clear all e_j registers, then continue to execute the next modular multiplication.

There are several issues in this design which need to be clarified. Once the final product is loaded into the output registers o_j, it can be shifted out serially. The pulse widths of MON and LD must be within the ranges of $T_1 + T_2 < T_M < T_1 + 2T_2$ and $T_2 < T_L < T_1 + T_2$ (as shown in Fig. 3) in order to generate the signals $\overline{CR}$ and LD properly, otherwise there will be some problems. For example, either the width of $\overline{CR}$ (or LD) is too short to clear e_j registers (load data into o_j register), or too long to work properly.

III. Computing $AB + C \bmod N$ and Evaluating Polynomials Using the Multiplier

Evaluating a polynomial is a fundamental operation in some group key-sharing systems [12]–[14]. The VLSI architecture for computing multiplication proposed in Section II can also be applied to a polynomial evaluation design. By the use of Horner's rule, polynomial evaluation can be achieved by computing $AB + C \bmod N$ iteratively.

A. Computing $AB + C \bmod N$

The algorithm introduced in Section II can be modified for computing $AB + C \bmod N$. Assume the binary representation of C is $C = c_{n-1}2^{n-1} + \cdots + c_1 2 + c_0$ and the restriction on A, B, and N is the same as the preceding. Then the operation $F = AB + C \bmod N$ can be rewritten as

$$F = \left[(Ab_{n-1} + c_{n-1})2^{n-1} + \cdots + (Ab_1 + c_1)2 + (Ab_0 + c_0) \right] \bmod N. \quad (3)$$

When compared to (1), we know that it is necessary to evaluate $F_{i+1} = [(F_i + Ab_{n-i-1} + c_{n-i-1}) \bmod N]2$ iteratively for computing (3). The inner sum is always within the range $0 \leqslant F_i + Ab_{n-i-1} + c_{n-i-1} < 3N$. Therefore, we can use the previous multiplier for computing $AB + C \bmod N$, if $SW2$ is switched to point P and SW_1 to point M. The serial inputs $0, 0, c_{n-1}, \cdots, c_1, c_0$ are latched in register 3 ($R3$) at the clock rate of $\phi 1$; i.e., for each iteration of computing $F_{i+1} = [(F_i + Ab_{n-i-1} + c_{n-i-1}) \bmod N]2$, the carry input of the parallel adder 0 at its least significant is c_{n-i-1} rather than ZERO state. The detailed computing procedures are analogous to the steps described in Section II.

B. Evaluating Polynomials

Consider that

$$P(X) = C_m X^m + C_{m-1} X^{m-1} + \cdots + C_1 X + C_0 \bmod N \quad (4)$$

is a polynomial with degree m, where the coefficients of this polynomial belong to the set $\{0,1,\cdots,N-1\}$, and at least one coefficient is not zero. Using Horner's rule and letting $X = A$, (4) can be rewritten as

$$\begin{aligned} P(A) &= (\cdots((0A + C_m)A + C_{m-1})A + \cdots + C_1)A + C_0 \bmod N \\ &= A(\cdots A(A(A0 + C_m) + C_{m-1}) + \cdots + C_1) + C_0 \bmod N \end{aligned} \quad (5)$$

From (5), the value $P(A)$ can be obtained by iteratively computing $(AB + C) \bmod N$ for $m+1$ times. It is obvious that the VLSI architecture for the multiplier described previously can be used to perform polynomial evaluation if the switches $SW1$ and $SW2$ are set to point P.

Finally, there are several points to be addressed. First, since the input o_{-1} is connected to the ZERO state, the registers o_j can be reset to ZERO before the signal $\overline{ST}$ arrives. Thus the first computation $(A0 + C_m) \bmod N$ can be completed within $n+2$ clock cycles by using a $(n+2)$-bit multiplier if $0 < N < 2^n$. The overall computation time for evaluating a polynomial $P(X)$ is $(n+2)(m+1)$ clock cycles. The control circuit for this operation contains one additional counter which is not shown in Fig. 1. The purpose of this counter is to indicate when the polynomial value $P(A)$ is available.

IV. Conclusion and Remarks

In this correspondence, a programmable VLSI architecture for computing $AB \bmod N$, $AB + C \bmod N$, and polynomial evaluation is proposed. The system constructed by the proposed architecture can be easily expanded. For example, by incorporating four 64-bit chips of this type and three additional LAC chips (such as SN74182), we can construct a 256-bit system. With this expansible property, the security of cryptosystems can be improved to the desired level by increasing the key length. The circuit inside the bolded rectangular block, which is shown in Fig. 1, can be fabricated in a 40-pin package. Using 2.5-μm CMOS process technology, the chip area of a 64-bit circuit is about 5.82×6.21 mm, and the estimated maximum clock rate is higher than 6 MHz at 5-V power-supply voltage. If the control circuit is also included in a single chip, a 64-pin package is required to maintain the expansible feature. In general, it needs $n+2$ clock cycles for this new design to compute $AB \bmod N$ or $AB + C \bmod N$, if the computing circuit is $(n+2)$-bit size. To complete an evaluation of a polynomial with mth degree, this architecture needs $(n+2)(m+1)$ clock cycles. These requirements of clock cycles can be minimized to n and $n(m+1)$, respectively, if outputs of the two registers o_{n-1} and d_{n-1} in this $(n+2)$-bit circuit are individually connected to two pins of a package.

Acknowledgment

It is a pleasure to acknowledge Dr. T. K. Truong and Dr. I. S. Hsu for their helpful suggestions.

References

[1] D. E. R. Denning, *Cryptography and Data Security*. Reading, MA: Addison-Wesley, 1983.

[2] C. S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," *IEEE Trans. Comput.*, vol. C-33, pp. 357–360, Apr. 1984.

[3] P. A. Scott, S. E. Tavares, and L. E. Peppard, "A fast VLSI multiplier for GF(2^m)," *IEEE J. Selected Areas Commun.*, vol. SAC-4, pp. 62–66, Jan. 1986.

[4] J. J. Chang, T. K. Truong, H. M. Shao, I. S. Reed, and I. S. Hsu, "The VLSI design of a single chip for the multiplication of integers modulo a Fermat number," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-33, pp. 1599–1602, Dec. 1985.

[5] S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106–110, Jan. 1978.

[6] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587–594, July 1984.

[7] E. F. Brickell, "A fast modular multiplication algorithm with application to two key cryptography," *Advances in Cryptology, Proceedings of Crypto'82*. New York: Plenum, 1983, pp. 51–60.

[8] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 208–210, Mar. 1983.

[9] A. Shamir, "Embedding cryptographic trapdoods in arbitrary knapsack systems," *Inform. Process. Lett.*, vol. 17, pp. 77–79, Aug. 1983.

[10] M. J. Foster and H. T. Kung, "The design of special-purpose VLSI chips," *Computer*, vol. 13, pp. 26–40, Jan. 1980.

[11] H. Taub, *Digital Circuits and Microprocessors*. New York: McGraw-Hill, 1982.

[12] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, Nov. 1979.

[13] F. Luccio and S. Mazzone, "A cryptosystem for multiple communication," *Inform. Process. Lett.*, vol. 10, pp. 180–183, July 1980.

[14] D. E. Denning and F. B. Schneider, "Master keys for group sharing," *Inform. Process. Lett.*, vol. 12, pp. 23–25, Feb. 1981.