REFERENCES

[1] K. De, C. Natarajan, D. Nair, and P. Banerjee, "RSYN: A system for automated synthesis of reliable multilevel circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 186–195, June 1994.

[2] N. K. Jha and S.-J. Wang, "Design and synthesis of self-checking VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 879–887, June 1993.

[3] R. Leveugle and G. Saucier, "Optimized synthesis of concurrently checked controllers," *IEEE Trans. Comput.*, vol. 39, pp. 419–425, Apr. 1990.

[4] M. Diaz, J. C. Geffroy, and M. Courvoisier, "On-set realization of fail-safe sequential machines," *IEEE Trans. Comput.*, vol. C-23, pp. 133–138, Feb. 1974.

[5] H. Y. H. Chuang and S. Das, "Design of fail-safe sequential machines using separable codes," *IEEE Trans. Comput.*, vol. C-27, pp. 249–252, Mar. 1978.

[6] M. Nicolaidis, R. O. Duarte, S. Manich, and J. Figueras, "Fault-secure parity prediction arithmetic operators," *IEEE Design Test Mag.*, April/June 1997.

[7] *IEEE Standard VHDL Reference Manual*, IEEE Standard 1076, 1987.

[8] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[9] S. Tarnick, "Controllable self-checking checkers for conditional concurrent checking," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 547–553, May 1995.

[10] V. S. Iyengar and L. L. Kinney, "Concurrent fault detection in microprogrammed control units," *IEEE Trans. Comput.*, vol. C-34, pp. 810–821, Sept. 1985.

[11] T. Nanya and T. Kawamura, "A note on strongly fault-secure sequential circuits," *IEEE Trans. Comput.*, vol. C-36, pp. 1121–1123, Sept. 1978.

[12] J. E. Smith and G. Metze, "Strongly fault secure logic networks," *IEEE Trans. Comput.*, vol. C-27, pp. 491–499, June 1978.

[13] J. E. Smith and P. Lam, "A theory of totally self-checking system design," *IEEE Trans. Comput.*, vol. C-32, pp. 831–844, Sept. 1983.

[14] C. Bolchini, F. Salice, and D. Sciuto, "The design of totally self-checking sequential circuits," Dept. Electronics, Politecnico di Milano, Italy, Rep. 96-019, Feb. 1996.

[15] "SIS: A system for sequential circuit synthesis," Dept. Elec. Eng. Comp. Sci., Univ. Calif., Berkeley, CA, Memo. UCB/ERL M92/41, 1992.

[16] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: State assignment of finite state machines for optimal two-level logic implementations," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 905–924, Sept. 1990.

[17] C. Bolchini, R. Montandon, F. Salice, and D. Sciuto, "A state encoding for self-checking finite state machines," in *Proc. IEEE VLSI'95*, Chiba, Japan, pp. 711–716.

[18] C. Bolchini, F. Salice, and D. Sciuto, "Redundant faults in TSC networks: Definition and removal," in *Proc. DFT '96*, Boston, MA, pp. 277–285.

[19] ——, "Design of totally self-checking checkers for a class of constant hamming distance codes," *J. Microelectron. Syst. Integrat.*, vol. 5, no. 2, pp. 85–100, June 1997.

[20] M. A. Marouf and A. D. Friedman, "Design of self-checking checkers for Berger codes," in *Proc. Int. Symp. Fault-Tolerant Comput.*, June 1978, pp. 179–184.

Two Systolic Architectures for Modular Multiplication

Wei-Chang Tsai, C. Bernard Shung, and Sheng-Jyh Wang

***Abstract*—This article presents two systolic architectures to speed up the computation of modular multiplication in RSA cryptosystems. In the double-layer architecture, the main operation of Montgomery's algorithm is partitioned into two parallel operations after using the precomputation of the quotient bit. In the non-interlaced architecture, we eliminate the one-clock-cycle gap between iterations by pairing off the double-layer architecture. We compare our architectures with some previously proposed Montgomery-based systolic architectures, on the basis of both modular multiplication and modular exponentiation. The comparisons indicate that our architectures offer the highest speed, lower hardware complexity, and lower power consumption.**

***Index Terms*—Cryptosystems, Montgomery, RSA, systolic.**

I. INTRODUCTION

In electronic commerce, secure transactions are becoming a major concern. Public-key cryptography [1], [2] has been shown to be an attractive technique for such applications. The RSA cryptosystem, one of the best known public-key systems, performs *modular exponentiation*, $M^E \pmod N$, which can be decomposed into a series of *modular multiplications*, $AB \pmod N$. There have been a lot of research activities related to the algorithms or architectures of modular multiplications for RSA [3]–[10]. Among them, Montgomery's modular multiplication algorithm has several advantages when implemented in VLSI [6]. The radix-2 Montgomery's algorithm for the modular multiplication, $A \times B \pmod N$, is described as follows, where A, B, and N are m-bit numbers (N is an odd number) and $A = \sum_{i=0}^{m-1} a_i 2^i$:

Montgomery's Algorithm for Modular Multiplication (Radix 2)

$R_{-1} = 0$;
for $i = 0$ to $m - 1$
  $q_i = R_{i-1} + a_i B \pmod 2$;
  $R_i = (R_{i-1} + a_i B + q_i N)/2$;
end.

Here, R_i represents the partial result. The fast computation of R_i inherently needs a quick enough broadcast of q_i.

There have been several Montgomery-based systolic VLSI architectures proposed for modular multiplications that can avoid the problem of broadcasting q_i [8]–[10]. In this paper, we propose two systolic architectures to speed up the computation of modular multiplication. In the first architecture, called the *double-layer* architecture, the main operation of Montgomery's algorithm is partitioned into two parallel operations after the precomputation of the quotient bit q_i. By combining precomputation and partitioning, the clock period is halved and the speed of this architecture is improved by a factor of two if the unpartitioned circuit is not optimized. In the second architecture, called the *non-interlaced* architecture, the operations in the double-layer architecture are properly paired off, and their timing is arranged to remove the one-clock-cycle gap, which is inherent in the double-layer architecture. The speed is improved then, after optimizing the combined general cell of this non-interlaced architecture.

Manuscript received June 11, 1998; revised March 29, 1999.

The authors are with National Chiao Tung University, Hsinchu 30010, Taiwan, R.O.C.

Publisher Item Identifier S 1063-8210(00)00768-X.

II. Double-Layer Architecture

The main algorithm of a straightforward systolic implementation of Montgomery's algorithm can be expressed as

for $i = 0$ to $m - 1$

$q_i = (R_{i-1})_0 \oplus a_i b_0$;

$(R_i)_{j-1} + 2(Carry1_i)_j + 2(Carry2_i)_j$

$= (R_{i-1})_j + a_i b_j + q_i n_j + (Carry1_i)_{j-1} + (Carry2_i)_{j-1}$,

$j = 0 \sim m - 1$;

end.

Fig. 1(a) shows the data dependency between bits and between iterations. $(R_{i-1})_j$ is the result from the previous iteration, while $(Carry1_i)_{j-1}$ and $(Carry2_i)_{j-1}$ are the carries propagated from the adjacent lower bit of the same iteration.

In our architecture, the computation of $(R_i)_{j-1}$ is partitioned into the following two operations to shorten the clock period

$$(P_i)_j + 2(CP_i)_j = a_i b_j + q_i n_j + (CP_i)_{j-1} \quad (1)$$

$$(R_i)_{j-1} + 2(CR_i)_j = (R_{i-1})_j + (P_i)_j + (CR_i)_{j-1} \quad (2)$$

where $(CP_i)_j$ and $(CR_i)_j$ are the carries when calculating $(P_i)_j$ and $(R_i)_{j-1}$, respectively.

When q_i is available, $a_i b_j$ and $q_i n_j$ are computed in the first stage to generate $(P_i)_j$. Then $(R_i)_{j-1}$ is calculated in the second stage after $(P_i)_j$ and $(CR_i)_{j-1}$ are ready. As shown in Fig. 1(b), two kinds of simple cells are used, respectively, to calculate $(P_i)_j$ and $(R_i)_{j-1}$. The upper layer cells compute $(P_i)_j$, and the lower layer cells compute $(R_i)_{j-1}$.

However, this partition alone does not improve the computation speed at all. Note that q_i has to be computed before P_i is computed bit by bit. On the other hand, due to the divide-by-two operation in Montgomery's algorithm, the computation of $(R_i)_0$ depends on $(P_i)_1$. According to these dependencies, there is a four-clock-cycle delay between $(R_{i-1})_0$ and $(R_i)_0$, as shown in Fig. 2. Therefore, even though the partition has shortened the clock cycle, the number of clock cycles between adjacent iterations is also doubled. To speed up the computation, we need some precomputations to break the dependencies.

We observed that both q_i and $(CP_i)_0$ can be precomputed from $a_i b_0$, $(R_{i-2})_1$, $(P_{i-1})_1$, and $(CR_{i-1})_0$ directly; $(P_i)_0$ can also be computed from $(R_{i-2})_1$, $(P_{i-1})_1$, and $(CR_{i-1})_0$. The precomputations of q_i, $(P_i)_0$, and $(CP_i)_0$ can be calculated in parallel with the computation of $(R_{i-1})_0$. With these precomputations, the timing sequence can be rearranged as shown in Fig. 3. Now, the delay time between successive iterations is reduced down to two clock cycles. We observe that there is no impact on the speed of the architecture since the number of logic levels can be adjusted according to the maximum critical path of unit cells. The operation speed is enhanced by a factor of two if the unpartitioned circuit is not optimized. Even though plenty of efforts have been made before to speed up the computation of Montgomery's algorithm using the concept of precomputation, none of them has discovered the possibility of combining both precomputation and partitioning to double the speed in a systolic architecture. Note that there is still a one-clock-cycle gap between $(R_{i-1})_0$ and $(R_i)_0$. This is due to the inherent characteristics of the divide-by-two operation in Montgomery's algorithm. This effect will be explained in detail in Section III.

Fig. 4(a) shows the two-dimensional (2-D) systolic implementation of the double-layer architecture, in which B-cell computes $(P_i)_j$, C-cell computes $(R_i)_j$, and A-cell is a modified version of B-cell to take care of precomputation. The outputs of A-cell and B-cell are fed into both the C-cell of the same iteration and the A-cell and B-cell of the next bit, while the output of the C-cell is fed into the C-cell corresponding to the previous bit of the next iteration. The inputs of the C-cell, which are used to generate $(R_i)_0$, are also connected to the A-cell of the next iteration for precomputation. By projecting the 2-D systolic array along the y direction, we get the one-dimensional (1-D) systolic array as shown in Fig. 4(b). In the 1-D systolic structure, each cell calculates the corresponding bit of R_i for every i, $i = 0 \sim m - 1$. The HSPICE simulation of the three cells (A, B, C-cells) shows that the circuit delay is less than 2 ns, if using TSMC 0.35-μm CMOS technology models.

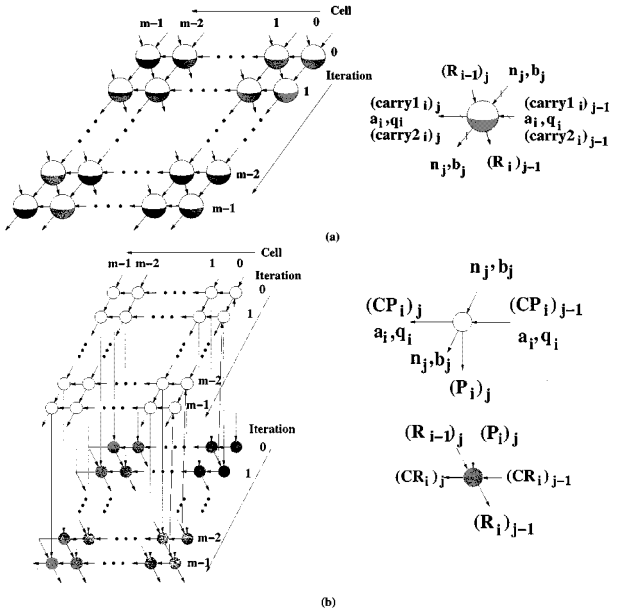

Fig. 1. Data dependency of Montgomery's algorithm. (a) Original architecture. (b) Double-layer architecture.

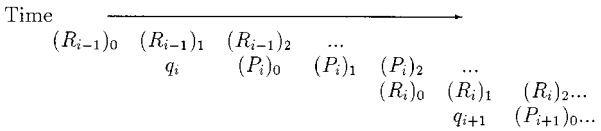

Fig. 2. Timing sequence of $(P_k)_l$ and $(R_k)_l$.

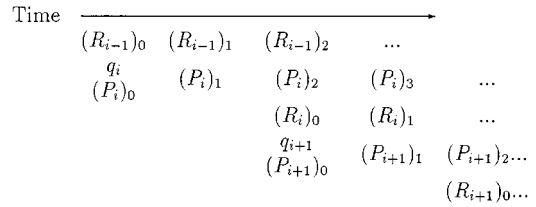

Fig. 3. Timing sequence of $(P_k)_l$ and $(R_k)_l$ after removing some data dependencies.

III. Non-Interlaced Architecture

As mentioned in Section I, there is a one-clock-cycle gap in a bit-by-bit architecture. Here, we use a typical systolic architecture to explain this effect. Fig. 5 shows the spatial connection and temporal dependency of this systolic array. Here, we use dashed lines to indicate

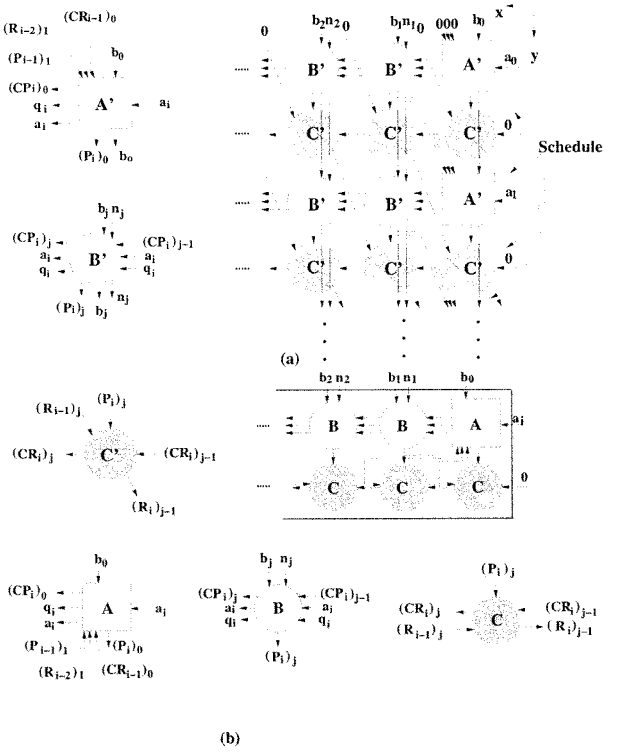

Fig. 4. Two-dimensional and one-dimensional structures of the double-layer systolic array.

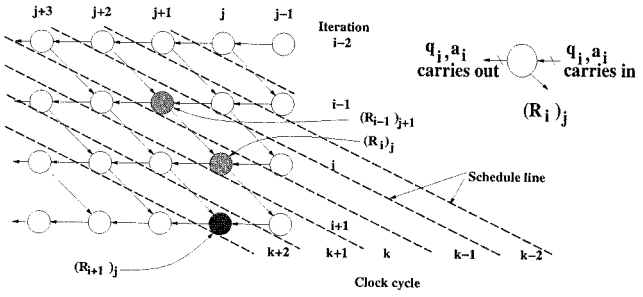

Fig. 5. Feasible arrangement of schedule lines (larger than 135°).

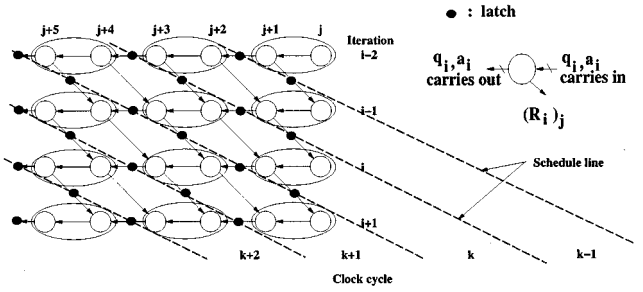

Fig. 6. Pair-off style of the non-interlaced architecture.

the schedule lines. Assume that the jth circle at the ith row does the computation of $(R_i)_j$ and that the arrows between circles indicate the data dependency. Due to the divide-by-two operation in Montgomery's algorithm, $(R_{i-1})_{j+1}$ has to be computed before the computation of $(R_i)_j$. This is why some arrows have a –45° orientation. Due to this fact, the angle of the schedule lines must be larger than 135° and smaller than 180° for a feasible implementation without violating the data flows. Fig. 5 shows a feasible arrangement of a bit-by-bit architecture. Under this arrangement, $(R_{i-1})_{j+1}$ is computed before $(R_i)_j$, and there is no time conflict. However, if $(R_i)_j$ is computed in the kth cycle, $(R_{i+1})_j$ will be computed in the $(k+2)$th cycle. Up to now, no feasible bit-by-bit architecture has been found that computes $(R_{i+1})_j$ in the $(k+1)$th cycle. Hence, there is always a one-clock-cycle gap between adjacent iterations in a feasible bit-by-bit architecture. Both Walter's and Kornerup's designs have this problem [8], [9].

In this section, we propose the non-interlaced architecture to eliminate the one-clock-cycle gap by pairing off the computation of $(R_i)_j$'s and rearranging the operation timing. Here, we group $(R_i)_j$ and $(R_i)_{j+1}$ together, $(R_i)_{j+2}$ and $(R_i)_{j+3}$ together, and so on (see Fig. 6). With this arrangement, assume that both $(R_i)_j$ and $(R_i)_{j+1}$ are computed in the kth-clock cycle. Then $(R_{i+1})_j$ can be computed right in the $(k+1)$th-clock cycle. There is no clock-cycle gap between $(R_i)_j$ and $(R_{i+1})_j$. Nevertheless, further consideration of this straightforward grouping is necessary due to some data dependency between $(R_i)_j$'s. Note that the pair of $(R_i)_j$ and $(R_i)_{j+1}$ depends on the pair of $(R_{i-1})_{j+2}$ and $(R_{i-1})_{j+3}$; the pair of $(R_{i-1})_{j+2}$ and $(R_{i-1})_{j+3}$ depends on the pair of $(R_{i-2})_{j+4}$, and $(R_{i-2})_{j+5}$, etc. It may initially appear infeasible to have all of them computed within the same clock cycle. Fortunately, $(R_{i-1})_{j+2}$ is computed within the first half of the kth-clock cycle (because both $(R_{i-1})_{j+1}$ and $(R_{i-2})_{j+3}$ have already been computed in the $(k-1)$th-clock cycle) while $(R_i)_{j+1}$ is to be computed within the second half of the kth-clock cycle. Similarly, $(R_{i-2})_{j+4}$ is computed within the first half of the clock cycle, while $(R_{i-1})_{j+3}$ is computed within the second half. This lag in time makes it possible to have all $(R_i)_j$, $(R_i)_{j+1}$, $(R_{i-1})_{j+2}$, $(R_{i-1})_{j+3}$, $(R_{i-2})_{j+4}$, ... computed in the same clock cycle. Therefore, after removing the latch between $(R_{i-1})_{j+2}$ and $(R_i)_{j+1}$, the latch between $(R_{i-2})_{j+4}$ and $(R_{i-1})_{j+3}$, and so on, all these operations can be computed in time.

After pairing off $(R_i)_j$'s, the one-clock-cycle gap is removed while the length of the clock period is doubled. The delay time between iterations is not changed, and there is no improvement in speed so far. However, by applying optimization on the merged pairs, the critical path can be shortened and the operation speed can thus be improved. Even though the speed optimization tends to increase the number of logic gates, the overall area of the non-interlaced architecture actually decreases slightly compared to the double-layer architecture. This is because the number of flip-flops in the non-interlaced architecture is reduced by about one half only. On the other hand, Kornerup's design [9] can also be thought to have paired off the straightforward systolic array in a slightly different way. His merging can save about the same amount of flip-flops as the non-interlaced architecture. However, the one-clock-cycle gap still exists in his design.

Fig. 7(a) shows the 2-D structure of the non-interlaced architecture. With this arrangement, the contiguous iterations can be successively initiated without the one-clock-cycle gap. Fig. 7(b) shows the 1-D projection of Fig. 7(a) along the y direction. The D-Cell represents the least significant bit cell; the F-Cell represents the general cell; and the E-cell represents the modified F-Cell for precomputation. For iteration i, the jth general cell computes the $(2j-2)$th and $(2j-3)$th bits of R [i.e., $(R_i)_{2j-2}$ and $(R_i)_{2j-3}$] within a clock cycle according to the following:

$$4(CP_i)_j + 2(P_i)_{2j} + (P_i)_{2j-1} = 2a_i b_{2j} + 2q_i n_{2j} + a_i b_{2j-1} + q_i n_{2j-1} + (CP_i)_{j-1} \quad (3)$$

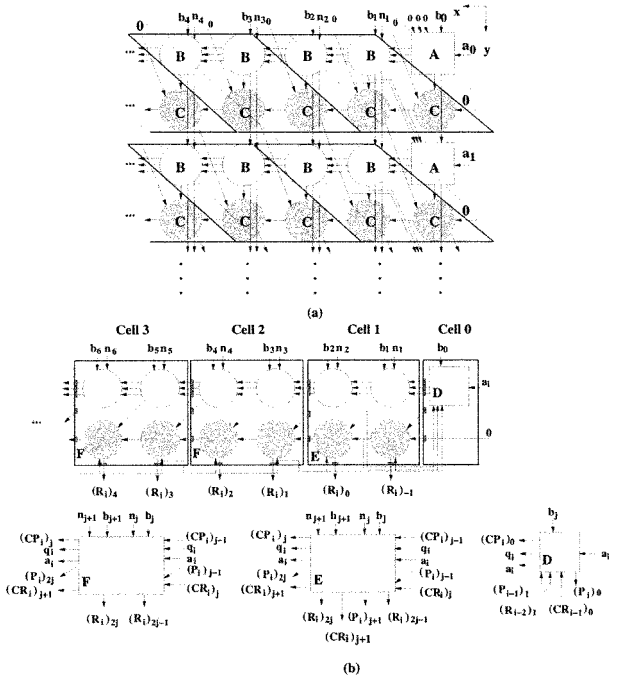

Fig. 7. Two-dimensional and one-dimensional structures of the non-interlaced systolic array.

and

$$\begin{aligned} 4(CR_i)_{j-1} + 2(R_i)_{2j-2} + (R_i)_{2j-3} \\ = 2(R_{i-1})_{2j-1} \\ + 2(P_i)_{2j-1} \\ + (R_{i-1})_{2j-2} + (P_i)_{2j-2} \\ + (CR_i)_{j-2}. \end{aligned} \tag{4}$$

In these two equations, $(P_i)_{2j-1}$ comes from (3) and is used by (4) within the general cell (F-Cell). On the other hand, $(R_{i-1})_{2j-1}$ comes from the output of the $(j+1)$th cell and is calculated at the same time as the jth cell is computing $(R_i)_{2j-2}$ and $(R_i)_{2j-3}$. To speed up the computation, we optimize the whole combinational part of this cell. The HSPICE simulation of the three cells (D-, E-, F-cells) shows that the circuit delay is less than 2.7 ns, if using TSMC 0.35-μm CMOS technology models.

IV. Comparison for Modular Multiplication and Exponentiation

The double-layer architecture has the one-clock-cycle gap problem, while the non-interlaced architecture does not. When computing a modular multiplication in the double-layer architecture, the one-clock-cycle gaps can be utilized to simultaneously compute another independent modular multiplication. On the other hand, it is more efficient to compute a sequence of dependent modular multiplications using the non-interlaced architecture.

Tables I–III show various comparisons for a 1024-bit modular multiplication among several Montgomery-based systolic arrays. In Table I, we compare the cycle time and the number of clock cycles. The simulation was performed using SPICE with TSMC 0.35-μm CMOS technology models. It can be seen that our double-layer architecture has the shortest cycle time and our non-interlaced architecture has the smallest number of cycles. Table II shows the comparison of the hardware complexity for computing 1024-bit modular multiplications. This table does not contain the hardware requirement for the control part of the systolic arrays. However, the control schemes of these architectures are roughly the same due to the similar data formats and operation orders. It can be seen that Kornerup's design and our non-interlaced architecture have lower hardware complexity, due to the reduced number of flip-flops. Table III shows the comparison of power consumption for computing 1024-bit modular multiplications. We assume that the clock power is one quarter of the total power. We also show a comparison at a normalized frequency at 278 MHz, which is the maximum working frequency of (the slowest) Kornerup's design. Once again, Kornerup's design and our non-interlaced architecture have lower power consumption than most architectures, due to the reduced number of flip-flops.

Tables IV and V show various comparisons for computing a 1024-bit modular exponentiation. We assume that the added control circuitry for modular exponentiation does not affect the critical path of the modular multiplication core. In Table IV, therefore, we use the same simulated

TABLE I
Comparison of the Cycle Time, Cycle Number, and Computation Time

Author	Critical Path	Cycle time	N_{mul}	T_{mul}
Walter [8]	$max(4T_{AND/OR}+T_{XOR}, 2T_{AND/OR}+2T_{XOR})$	2.4ns	$\simeq 2m$	$4.9\mu s$
Kornerup [9]	$T_{AND/OR}+4T_{XOR}$	3.6ns	$\simeq 2m$	$7.4\mu s$
Chen et al. [10]	$T_{AND/OR}+2T_{XOR}$	2.0ns	$\simeq 4m$	$8.2\mu s$
Double-layer	$T_{AND/OR}+2T_{XOR}$	2.0ns	$\simeq 2m$	$4.1\mu s$
Non-interlaced	$T_{AND/OR}+3T_{XOR}$	2.7ns	$\simeq m$	$2.8\mu s$

$T_{AND/OR}$: Time delay of an AND or OR gate.
T_{XOR}: Time delay of an XOR gate.
$T_{FA} \approx 2T_{XOR}$.
N_{mul}: number of clock cycles per modular multiplication.
T_{mul}: time per modular multiplication (1024 bits).

TABLE II
Comparison of Area and the Transistor Count

Author	Area	Transistor count
Walter [8]	$9mA_{AND/OR}+5mA_{XOR}+5mA_{DFF}$	$\simeq 240$K
Kornerup [9]	$2mA_{AND/OR}+2mA_{FA}+3mA_{DFF}$	$\simeq 172K$
Chen et al. [10]	$2mA_{AND/OR}+2mA_{FA}+7mA_{DFF}$	$\simeq 240$K
Double-layer	$6mA_{AND/OR}+4mA_{XOR}+6mA_{DFF}$	$\simeq 240$K
Non-interlaced	$12.5mA_{AND/OR}+6mA_{XOR}+3.5mA_{DFF}$	$\simeq 209$K

$A_{AND/OR}$: Area of an AND or OR gate.
A_{XOR}: Area of an XOR gate.
A_{FA}: Area of a full adder.
A_{DFF}: Area of a D type flipflop.

TABLE III
Comparison of the Power Among Several Systolic Arrays

Author	$Power_{max-speed}$	Power at 278MHz
Walter [8]	0.787W	0.557W
Kornerup [9]	$0.387W$	$0.387W$
Chen et al. [10]	1.224W	0.735W
Double-layer	1.134W	0.681W
Non-interlaced	0.803W	0.593W

$Power_{max-speed}$: Power at maximum working frequency. Assume the clock power is about 25%.

TABLE IV
COMPARISON OF THE CYCLE NUMBER AND COMPUTATION TIME

Author	Cycle time	N_{exp}	1024-bit RSA speed
Walter [8]	2.4ns	$\simeq 2m^2$	203Kbps
Kornerup [9]	3.6ns	$\simeq 2m^2$	136Kbps
Chen et al. [10]	2.0ns	$\simeq 4m^2$	122Kbps
Double-layer	2.0ns	$\simeq 2m^2$	244Kbps
Non-interlaced	2.7ns	$\simeq 1.5m^2$	241Kbps

N_{exp}: Average number of clock cycles per modular exponentiation.

TABLE V
COMPARISON OF HARDWARE COMPLEXITY IN TERMS OF TRANSISTOR COUNT

Author	Core	Registers	Control	Total
Walter [8]	$\simeq$ 240K	$\simeq$ 147K	$\simeq$ 147K	$\simeq$ 534K
Kornerup [9]	$\simeq$ 344K	$\simeq$ 147K	$\simeq$ 98K	$\simeq$ 589K
Chen et al. [10]	$\simeq$ 240K	$\simeq$ 147K	$\simeq$ 135K	$\simeq$ 522K
Double-layer	$\simeq$ 240K	$\simeq$ 147K	$\simeq$ 147K	$\simeq$ 534K
Non-interlaced	$\simeq$ 209K	$\simeq$ 123K	$\simeq$ 98K	$\simeq$ 430K

cycle time from Table I. With the cycle time and the average number of clock cycles per modular exponentiation, we obtain the throughput rate of different architectures. It can be seen that our double-layer and non-interlaced architectures outperform all the other architectures in throughput rate.

Table V shows the estimated transistor counts among several systolic arrays to perform modular exponentiation. In Kornerup's design, two copies of the modular multiplication hardware core are required to perform modular exponentiation. The other architectures only require one copy of the hardware core. Table V also includes the hardware complexity estimates of the data registers (to store the RSA-specific constants and variables), and the control circuitry. In Table V, it can be seen that our double-layer architecture and non-interlaced architecture compare favorably in hardware complexity for a 1024-bit RSA operation. Since power consumption is closely related to the hardware complexity, we conjecture that the power consumption of our proposed architectures is also among the lowest.

V. CONCLUSION

In this paper, we proposed two new systolic architectures for improving the computation speed of modular multiplication. In the double-layer architecture, we increase the throughput rate by decomposing the main operation and adding some precomputation hardware. In the non-interlaced architecture, we eliminate the one-cycle gap between iterations by pairing off the double-layer architecture. We compare our architectures with some previously proposed Montgomery-based systolic architectures. The comparisons indicate that our architectures offer the highest speed, low hardware complexity, and low power consumption among these architectures.

REFERENCES

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.
[2] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.
[3] G. R. Blakley, "A computer algorithm for the product AB modulo M," *IEEE Trans. Comput.*, vol. 32, pp. 497–500, 1983.
[4] E. F. Brickell *et al.*, "A fast modular multiplication algorithm with application to two key cryptography," in *Advances in Cryptology—EUROCRYPT'82*, E. F. Chaum *et al.*, Eds. New York: Plenum, 1983, pp. 51–60.
[5] Ç. K. Koç and C. Y. Hung, "Bit-level systolic arrays for modular multiplication," *J. VLSI Signal Process.*, vol. 3, pp. 215–223, 1991.
[6] P. L. Montgomery, "Modular multiplication without trial division," *Math. Computat.*, vol. 44, pp. 519–521, Apr. 1985.
[7] C. D. Walter, "Still faster modular multiplication," *Electron. Lett.*, vol. 31, no. 4, pp. 263–264, Feb. 1995.
[8] ——, "Systolic modular multiplication," *IEEE Trans. Comput.*, vol. 42, pp. 376–378, Mar. 1993.
[9] P. Kornerup, "A systolic, linear-array multiplier for a class of right-shift algorithms," *IEEE Trans. Comput.*, vol. 43, pp. 892–898, Aug. 1994.
[10] P. S. Chen, S. A. Hwang, and C. W. Wu, "A systolic RSA public key cryptosystem," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, May 1996, pp. 408–411.
[11] M. Shand and J. Vuillemin, "Fast implementation of RSA cryptography," in *Proc. 11th IEEE Symp. Computer Arithmetic*, 1993, pp. 252–259.
[12] S. E. Eldridge and C. D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 42, June 1993.
[13] C. Yang, C. Jen, and T. Chang, "The IC design of a high speed RSA processor," in *Proc. IEEE Asia Pacific Conf. Circuits and Systems'96*, Nov. 1996, pp. 33–36.