MODULAR MULTIPLICATION ARCHITECTURE FOR PUBLIC KEY CRYPTOSYSTEM

Keon-Jik Lee, Byeong-Jik Lee*, Chong-Won Park
Computer & Software Laboratory, ETRI, Daejeon 305-350 KOREA
**Dept. of Computer Engineering, Kyungpook National Uni., Daegu 702-701 KOREA*

ABSTRACT

A new processing element structure and bit level systolic multiplier is proposed for the efficient implementation of Montgomery's multiplication algorithm in RSA cryptosystem. By reorganizing and analyzing the recursive equation of Montgomery algorithm at the Boolean operator level, the critical path delay of the proposed PE is shorter compared with other designs. The critical path delay of the proposed PE is reduced by 20% and 29% when compared to a Walter PE and Guo's bit-serial PE, respectively. Furthermore, the area requirement of the proposed PE is 10% smaller than that of a Walter PE. With a continuous data input, the proposed array can produce multiplication results at a rate of one per $n+3$ cycles with a latency of $3n + 6$ cycles, where n is the bit size of the modulus. The proposed architecture is highly regular, nearest-neighbor connected, and thus well suited for VLSI implementation.

KEYWORDS

Modular multiplication, Cryptography, Systolic array

1. INTRODUCTION

It is extensively recognized that security issues will play a crucial role in many future communication and computer systems. The fundamental security requirements include authentication, confidentiality, integrity and non-repudiation. For physical security reasons as well as for performance, it is often advantageous to implement cryptographic algorithms in hardware. To provide such security services, most systems use public key (i.e., asymmetric key) cryptography algorithms such as the RSA algorithm (Rivest, et al. 1978), Diffie-Hellman algorithm (Diffie and Hellman, 1976), etc. The characteristic advantage of public key cryptosystems is that they acquire security without requiring transmission of secret information (i.e., a private key) regarding how to decode a received data (i.e., encrypted data). This is in contrast to symmetric key cryptosystems that demand a secure key transmission.

Among them, RSA cryptosystem is the best known and most widely used in public key cryptosystems. The core arithmetic operation of RSA is modular exponentiation, which can be accomplished by a sequence of modular multiplications. However, the modular exponentiation of several hundreds bits (usually > 500 bits) makes it difficult for the RSA cryptosystem to achieve a high throughput. Accordingly, methods which can speed up the implementation of the modular exponentiation are of considerable practical significance. The key point to implementing modular exponentiation is the development of an efficient algorithm for modular multiplication with large numbers under limited hardware resources. A commonly used method for performing modular multiplication is Montgomery's modular multiplication algorithm (MMM) (Montgomery, 1985). Several approaches have been proposed to design hardware architecture for MMM based on a bit-serial computation (Guo, et al, 1999; Koc, et al, 1996; Kornerup, 1994; Orup, 1995; Shin, 1998; Walter, 1993).

In this paper, we present a systematic design methodology of new bit level systolic multiplier based on analyzing and reconstructing the recursive equation of Montgomery algorithm at the Boolean operator level. In the proposed design, a higher clock rate can be applied because the critical path delay is shorter than in other designs.

2. PROPOSED ALGORITHM FOR MONTGOMERY MODULAR MULTIPLICATION

The RSA cryptosystem performs the computation of $C \equiv M^E$ mod N, where C is ciphertext, M is plaintext, E is encryption key, and N is modulus. In many RSA implementations, for the exponent E with long word-length, the square-and-multiply method is usually used to compute the modular exponentiation (Knuth, 1981; Menezes, et al, 1997). This algorithm performs modular exponentiations by using squaring and multiplying cyclically. The computation of $T \equiv A \cdot B$ mod N plays a very important role in modular exponentiation. To perform this computation efficiently, Montgomery multiplication algorithm is widely used.

The Montgomery modular multiplication algorithm introduced by P. L. Montgomery speeds up the modular multiplications and squarings required for exponentiation. It computes the Montgomery product MMM(A, B) = $A \cdot B \cdot R^{-1}$ mod N, where $R = r^{n+2}$. It is a method for multiplying two integers modulo N, while avoiding division by N. Multiplication modulo r and division by r are both intrinsically fast operations, since r is a power of 2. The Montgomery algorithm for performing modular multiplication $A \cdot B$ mod N can be described as Figure 1.

```
Input:    $A, B, r, N$
Output:   $T = A \cdot B \cdot r^{-(n+2)} \bmod N$
$T = 0$
for $i = 0$ to $n + 1$ do
          $Q_i = (T + A_i B) \bmod r$
          $T = (T + A_i B + Q_i N) \text{ div } r$
end
```

Figure 1. Montgomery Modular Multiplication

The result T is ABR–1 mod N, where N is an odd number of n-bit, A and B are two arbitrary integers falling within the range of [0, 2N], r = 2 and $R = r^{n+2}$. Note that there is n + 2 passes through the for i loop. In Walter (1993), the output T can be directly used as the next input B in modular exponentiation. But the output T cannot be used as the input A because A is n-digit number. Thus, the algorithm was modified in order to take A with n + 1 digits as the next input as shown in Figure 1. Here, all numbers are represented in base r = 2. Increasing the base reduces the number of digits in the multiplicand A, multiplier B, and modulus N and so reduces the number of clock cycles in the algorithm. However, the depth of hardware that has to be driven in a single clock cycle is increased as well, so that a slower clock must be used. The bit level algorithm of Figure 1 is as follows:

```
Input:        $A, B, r, N$
Output:       $T = A \cdot B \cdot r^{-(n+2)} \bmod N$
for $i = 0$ to $n + 1$ do
    for $j = 0$ to $n + 2$ do
        if ($j = 0$) then
            $Q_i = (T_{i,j} + A_i B_j) \bmod 2$
        $T'$            $= (T_{i,j} + A_i B_j + C0_{i,j}) \bmod 2$
        $C0_{i,j+1}$    $= (T_{i,j} + A_i B_j + C0_{i,j}) \text{ div } 2$
        $T_{i+1,j-1}$   $= (T' + Q_i N_j + C1_{i,j}) \bmod 2$
        $C1_{i,j+1}$    $= (T' + Q_i N_j + C1_{i,j}) \text{ div } 2$
    end
end
```

Figure 2. Bit level algorithm for Montgomery Modular Multiplication

where the initial value of T is zero and whenever the inner j loop starts, the values of $C0$ and $C1$ are reset to zero. After n + 2 iterations of the outer i loop, the result $T = [T_{n+2,\,n}, \ldots, T_{n+2,\,0}]$ is an (n + 1)-bit number and can be directly used as a new operand without reduction in modular exponentiation. Each intermediate result

$T_{i+1,\, j-1}$ for $0 \le i \le n$, $1 \le j \le n+2$, being in the range of $[0, 3N]$, needs $(n + 2)$-bit to represent itself (i.e., the inner j loop needs $n + 3$ iterations).

The main goal of the current work is to identify a design that combines the smallest input/output delay with the lowest expense (uses the smallest possible number of gates). Note that cost minimization can be seen to increase the delay by introducing extra levels of gates, whereas delay minimization tends to increase the cost by increasing the number of gates and the number of inputs required for each gate. In many cases, however, these two goals can be synchronized. Since the goal of delay minimization is to reduce the delay on the longest input/output path, called the critical path, there is no reason why all the noncritical paths could not be optimized for cost, as long as the delay of the noncritical paths does not exceed the delay of the critical path. Therefore, we begin by minimizing the delay of the critical path and then proceed to minimize the cost of the noncritical paths. With Boolean operators, the algorithm of Figure 2 can be rewritten as follows:

```
Input :    $A, B, r, N$
Output:    $T = A \cdot B \cdot r^{-(n+2)} \bmod N$
**for** $i = 0$ **to** $n + 1$ **do**
    **for** $j = 0$ **to** $n + 2$ **do**
        **if** $(j = 0)$ **then**
            $Q_i = T_{i,j} \oplus (A_i \wedge B_j)$
        $T' \quad = (T_{i,j} \oplus C0_{i,j}) \oplus (A_i \wedge B_j)$
        $C0_{i,j+1} = ((T_{i,j} \oplus C0_{i,j}) \wedge (A_i \wedge B_j)) \vee (T_{i,j} \wedge C0_{i,j})$
        $T_{i+1,j-1} = T' \oplus ((Q_i \wedge N_j) \oplus (D0_{i,j} \vee D1_{i,j}))$
        $D0_{i,j+1} = T' \wedge ((Q_i \wedge N_j) \oplus (D0_{i,j} \vee D1_{i,j}))$
        $D1_{i,j+1} = (Q_i \wedge N_j) \wedge (D0_{i,j} \vee D1_{i,j})$
    **End**
**End**
```

Figure 3. The Proposed Montgomery Multiplication

By reorganizing and analyzing the recursive equation of Montgomery algorithm at the Boolean operator level, the critical path delay is shorter compared with other designs. For a further comparison, the following assumptions, as discussed in detail in Gajski (1997), are made: $T_{XOR} = 4.2\Delta$, $A_{XOR} = 14\Phi$, $T_{AND} = 2.4\Delta$, $A_{AND} = 6\Phi$, $T_{OR} = 2.4\Delta$, $A_{OR} = 6\Phi$, where T_{GATE} and A_{GATE} are the time and area requirements of a 2–input gate and where Δ and Φ are the delay and area of an inverter circuit. As shown in Figure 3, the longest input/output path, called the critical path, is from input T to output T (XOR-XOR-XOR path), which requires 12.6Δ. The critical path of Figure 2, on the other hand, is from input A to output $C1$ (AND-XOR-XOR-XOR-AND-OR path), which requires 19.8Δ. Accordingly, it is clear that the implementation of Figure 3 is faster than that of Figure 2 by 36%.

When compared with Walter (1993), the performance efficiency of MMM can be determined based on the computational speed of each PE. The critical path of a Walter PE consists of an AND-OR-XOR-AND-OR path, which requires 13.8Δ. Therefore, the proposed PE is faster than that of a Walter PE by 9%. When comparing the area requirements of the respective PEs, the proposed PE needs 104Φ (six AND, two OR, and four XOR operators), whereas a Walter PE needs 124Φ (seven AND, two OR, and five XOR operators). Therefore, the cost of the proposed PE is 16% lower than that of a Walter PE. To compare with Shin et al. (1998), the area requirement of a PE is same as that of a Walter PE, but the critical path of a PE consists of an AND-XOR-AND-XOR path, which requires 13.2Δ. Therefore, the proposed PE is faster than that of a Shin's PE by 5%.

3. BIT LEVEL SYSTOLIC MULTIPLIER ARCHITECTURE

The algorithm proposed in the previous section can be illustrated by a two-dimensional dependence graph (DG) as shown in Figure 4, where $n = 3$. This DG consists of $(n + 2) \times (n + 3)$ basic cells. The circuit of the basic cell is shown in Figure 4, where each cell is composed of four 2-input XOR gates, six 2-input AND gates, and two 2-input OR gates. As described in the previous section, the critical path is from input $T_{i,j}$ to output $T_{i+1,j-1}$.

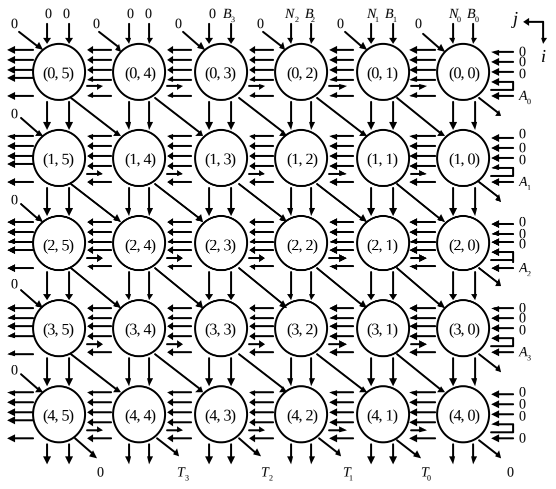

Figure 4. DG for proposed MMM with $n = 3$

In this DG, B and M enter the array from the top, while A enters the array from the right. The cell with index $[i, 0]^T$ computes $C0_{i,j+1}$, $D0_{i,j+1}$, $D1_{i,j+1}$, and $Q_i = T_{i,0} \oplus (A_i \wedge B_0)$ where the initial values of $C0_{i,j}$, $D0_{i,j}$, and $D1_{i,j}$ are '0', respectively, whereas the others computes $C0_{i,j+1}$, $D0_{i,j+1}$, $D1_{i,j+1}$, and $T_{i+1,j-1}$. In the jth row of the array, the value $T_{i,j}$ is computed with the partial product generated by the $(j-1)$-th row to give a new partial product that is passed on to the $(j+1)$-th row.

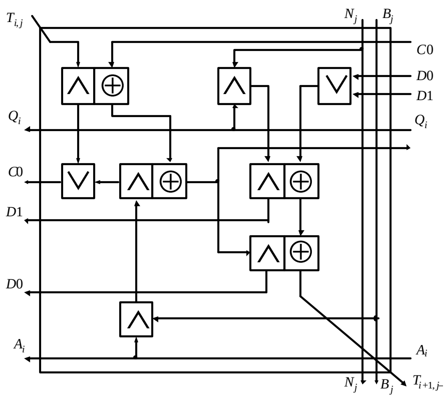

Figure 5. Circuit of the (i, j) cell in Figure 4

The initial positions and index points of each variable are as follows: A_i enters index $[i, 0]^T$ and flows in the direction $[0, 1]^T$. Since Q_i is used in all the cells of the row, it is sent on a horizontal path to all the cells in the i-th row. That is, Q_i is computed at index $[i, 0]^T$ and flows in the direction $[0, 1]^T$. The carry $C0$, $D0$, and $D1$ enter index $[i, 0]^T$ with an initial value of '0', respectively, thereafter they are computed at all index points, and flow in the direction $[0, 1]^T$. N_j and B_j enter index $[0, j]^T$ and flow in the direction $[1, 0]^T$. $T_{i,j}$ enters index $[0, j]^T$ with an initial value of '0', it is then computed at all index points, and flows in the direction $[1, -1]^T$. Since the values of $T_{i,j}$ for all i is computed concurrently and depends on the values computed in the preceding row, it is clear that operations are performed row by row.

The cells in the i-th row of the array perform the i-th iteration of the proposed algorithm, where the cell at position (i, j) computes the bit $T_{i+1,j-1}$ (see Figure 4). The final result T emerges from the bottom row of the array after $n + 2$ iterations. As shown in Figure 5, the (i, j)-th cell receives N_j and B_j as its input from the $(i-1, j)$-th cell, $T_{i,j}$ from $(i-1, j+1)$-th cell, and $C0$, $D0$, $D1$, Q_i, and A_i from $(i, j-1)$-th cell.

By projecting the DG in Figure 4 in a east direction following the projection procedure and cut-set systolization (Kung, 1988), a new bit serial systolic multiplier can be easily derived. The result is shown in

Figure 6, where $n + 2$ basic cells are used and "•" denotes a 1-bit delay element. This array is controlled by a control sequence 011···1 of length $n + 3$. The architecture of each basic cell is shown in Figure 7.

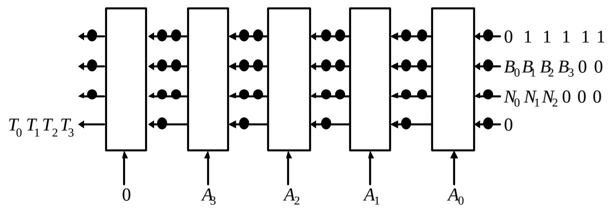

Figure 6. A bit serial systolic multiplier with $n = 3$

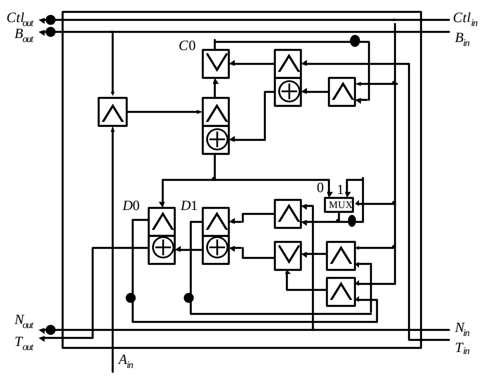

Figure 7. Circuit of each basic cell in Figure 6

According to the projection, the bits B_is and N_is enter the array in serial form with the least significant bit (LSB) first, while the bits A_is should reside inside the array, i.e., A_i should be at one cell to be ready for execution. In Figure 4, since each Q_i $(0 \le i \le n + 2)$ must be broadcast to all the cells in the first, second, …, and $(n + 2)$-th row, respectively, some extra circuitry is needed to latch such bits. Accordingly, in the basic cell shown in Figure 7, a 2-to-1 multiplexer (MUX) and 1-bit latch are added for this purpose. The operations of the extra circuitry are controlled by a control sequence Ctl = 011···1 of length $n + 3$. When the control signal Ctl is in logic 0, each Q_i $(0 \le i \le n + 2)$ is loaded into each PE. Furthermore, three 2-input AND gates are added to the cell in Figure 7 owing to the fact that three zeros must be fed to each row of the circuit in Figure 4 from the rightmost cell. When the control signal Ctl is in logic 0, these AND gates generate zeros.

If the input data come in continuously, this array can produce results at a rate of one per $n + 3$ cycles with a latency of $3n + 6$ cycles. The output result emerges from the left hand side of the array in serial form with the LSB first. It is also possible to incorporate extra one 2-to-1 MUX and one 1-bit latch into each cell in Figure 6 so that A may also enter the array in serial form with the LSB first.

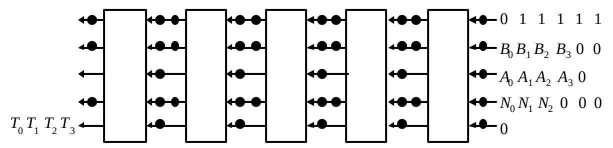

Figure 8. A modified version of the array shown in Figure 6

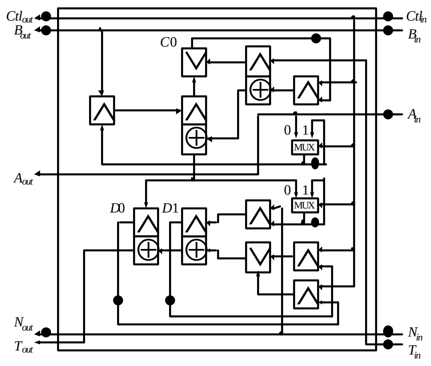

Figure 9. Circuit of each basic cell in Figure 8

The resulting array and its basic cell are shown in Figures 8 and 9, respectively. In this array, when the control signal *Ctl* is in logic 0, each A_i ($0 \le i \le n + 1$) is loaded into each PE. The basic cell of the Figure 9 is composed of four 2-input XOR gates, nine 2-input AND gates, two 2-input OR gates, two 2-to-1 MUXs, and thirteen 1-bit latches.

4. ANALYSIS AND CONCLUSION

We have presented a low-cost high-speed bit serial-in-serial-out systolic multiplier based on Montgomery algorithm to improve the computation speed of modular multiplication that is the core arithmetic operation in RSA cryptosystem. Efficient implementation needs an extensive analysis of a problem. A new PE structure for the efficient implementation of Montgomery algorithm is accomplished by reorganizing and analyzing the recursive equation of Montgomery algorithm at the Boolean operator level. As a result, the modified Montgomery algorithm efficiently reduces the critical path delay, thereby increasing the clock rate. For a further comparison, the following assumptions, as discussed in detail in Gajski (1997), are made: $T_{XOR} = 4.2\Delta$, $A_{XOR} = 14\Phi$, $T_{AND} = 2.4\Delta$, $A_{AND} = 6\Phi$, $T_{OR} = 2.4\Delta$, $A_{OR} = 6\Phi$, $T_{MUX} = 5.8\Delta$, $A_{MUX} = 18\Phi$, where T_{GATE} and A_{GATE} are the time and area requirements of a 2–input gate, and Δ and Φ are the delay and area of an inverter circuit, respectively. Also, the time and area requirement of 1-bit latch is 1.4Δ and 8Φ, respectively.

In Table 1, we compare the proposed architecture with some previously developed Montgomery-based systolic arrays. Comparison with related work based on the available data shows that the proposed method yields comparable or better results. The propagation delay of the proposed PE is reduced by 20%, 14%, and 29% when compared to a Walter PE, Shin PE, and Guo et al., respectively. Furthermore, the area requirement of the proposed PE is reduced by 10%, 18%, and 5% when compared to a Walter PE, Shin PE, and Guo et al., respectively. From Table 1, the proposed architecture has the lowest critical path delay, when compared with other works. The proposed design takes the same latency and throughput as the circuits in table, yet requires less computing time because of a shorter critical path. Since the critical path delay in modular multiplication is shorter compared with other methods, the proposed system yields a faster implementation, plus a higher clock rate can be also applied. As described in McCanny (1986), a system with unidirectional data flow gains advantages over a system with bi-directional data flow in terms of chip cascadability, fault tolerance.

Table 1. Comparison of the related bit-serial systolic multipliers

Circuit Item	Walter (1993)	Shin *et el.* (1998)	Guo *et el.* (1999)	Proposed method
Number of basic cells	$n + 2$	$n + 2$	$n + 2$	$n + 2$
Throughput (1/cycles)	$1/(n + 3)$	$1/(n + 3)$	$1/(n + 3)$	$1/(n + 3)$
Latency (cycles)	$3n + 6$	$3n + 6$	$3n + 6$	$3n + 6$
Data flow	Unidirectional	Unidirectional	Unidirectional	Unidirectional
Maximum cell delay	$2T_{XOR2} + 4T_{AND2} + 2T_{OR2} + 2T_{MUX2}$	$2T_{XOR2} + 4T_{AND2} + T_{OR2} + 2T_{MUX2}$	$3T_{XOR2}+4T_{AND2} +2T_{OR2}+2T_{MUX2}$	$4T_{XOR2} + 2T_{AND2} +T_{MUX2}$
Cell complexity	7 AND_2 gates 2 OR_2 gates 5 XOR_2 gates 12 1-bit latches 4 MUX_2	7 AND_2 gates 2 OR_2 gates 5 XOR_2 gates 13 1-bit latches 5 MUX_2	7 AND_2 gates 2 OR_2 gates 5 XOR_2 gates 12 1-bit latches 3 MUX_2	9 AND_2 gates 2 OR_2 gates 4 XOR_2 gates 13 1-bit latches 2 MUX_2
Basic components and Their numbers	**AND_2 gate:** $7n$+14 **OR_2 gate:** $2n$+4 **XOR_2 gate:** $5n$+10 **1-bit latch:** $12n$+24 **MUX_2:** $4n$+8	**AND_2 gate:** $7n$+14 **OR_2 gate:** $2n$+4 **XOR_2 gate:** $5n$+10 **1-bit latch:** $13n$+26 **MUX_2:** $5n$+10	**AND_2 gate:** $7n$+14 **OR_2 gate:** $2n$+4 **XOR_2 gate:** $5n$+10 **1-bit latch:** $12n$+24 **MUX_2:** $3n$+6	**AND_2 gate:** $9n$+18 **OR_2 gate:** $2n$+4 **XOR_2 gate:** $4n$+8 **1-bit latch:** $13n$+26 **MUX_2:** $2n$+4
AT- product	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Number of control signals	1	1	1	1

It is worthwhile to note that Walter and Shin *et al.* did not consider the feedback of the previous multiplication result in the modular exponentiation. To solve this problem, one processing element should be added to the architecture of a bit serial-in-serial-out implementation. It should be noted that Walter and Shin et al.'s implementation is not the design of the bit-parallel or bit-serial systolic array but the design of the dependence graph and cell circuit.

The proposed architecture is systolic and exploits pipelining and parallelism possible in order to obtain high speed and throughput. The proposed array involves unidirectional data flow and local interconnection, and is highly regular and modular. As a result, it is well suited to VLSI implementation. It is also worth noting that a bit-parallel system produces a much better throughput performance than a bit-serial one, although it involves much more circuit complexity. To improve this trade-off between throughput performance and hardware complexity, the adoption of digit-serial architecture of Aggoun et al. (1998) would seem to be a good approach. In Figure 10, the proposed array was simulated and verified using the Mentor Graphics VHDL (ChipSim) V16.00.00.12 simulator, where $n = 3$; Inputs: *A*: $(1101)_2$, *B*: $(1011)_2$, and *N*: $(111)_2$, Output *T*: $(110)_2$.

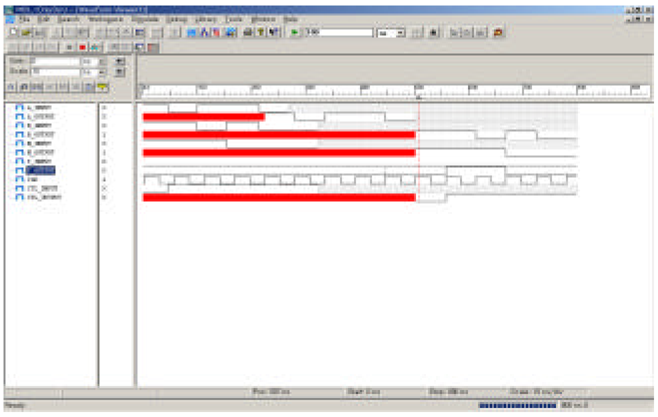

Figure 10. Simulation result

REFERENCES

Aggoun, A. et al, 1998. Bit-level pipelined digit-serial array processors, *IEEE Trans. Circuits and Systems*, Vol. 45, No. 7, pp. 857–868.

Diffie, W. and Hellman, M. E., 1976. New directions in cryptography, *IEEE Trans. Inform. Theory*, Vol. 22, No. 6, pp. 644–654.

Gajski, D. D., 1997. *Principles of digital design*, Prentice Hall.

Guo, J. H. et al, 1999. Design and implementation of an RSA public-key cryptosystem, *Proc. of IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp. 504–507.

Knuth, D. E., 1981. *The Art of Computer Programming, vol. 2*: *Seminumerical Algorithms, 2nd ed.* Addition-Wesley.

Koc, C. K. et al, 1996. Analyzing and comparing Montgomery multiplication algorithm, *IEEE Micro. Chip, Systems, Software and Applications*, Vol. 16, pp. 26–33.

Kornerup, P., 1994. A systolic linear array multiplier for a class of right-shift algorithms, *IEEE Trans. Computers*, Vol. 43, pp. 892–898.

Kung, S. Y., 1988. *VLSI array processors*, Prentice Hall.

McCanny, J. V. et al, 1986. Use of unidirectional data flow in bit-level systolic array chips, *Electron. Lett.*, Vol. 22, pp. 540–541.

Menezes, A. J. et al, 1997. *Handbook of applied cryptography*, CRC Press.

Montgomery, P. L., 1985. Modular multiplication without trial division, *Math. Comp*., Vol. 44, No. 170, pp. 519–521.

Orup, H., 1995. Exponentiation, Modular multiplication and VLSI implementation of high-Speed RSA cryptography, Ph.D. thesis, University of Aarhus.

Rivest, R. L. et al, 1978. A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126.

Shin, J. B. et al, 1998. Optimization of Montgomery modular multiplication algorithm for systolic arrays, *Electron. Lett.*, Vol. 34, pp. 1830–1831.

Walter, C. D., 1993. Systolic modular multiplication, *IEEE Trans. Comput*., Vol. 42, No. 3, pp. 376–378.