# 수정 및 보다 향상된 성능의 몽고메리 모듈러 곱셈기 제안

신준범 [0]　　이광형
한국과학기술원 전자전산학과
{jbshin, khlee}@monami.kaist.ac.kr

# Correction and further improvements of Montgomery Modular Multiplier

Jun-Bum Shin [0]　　H. Lee-Kwang
Department of Computer Science, KAIST

요약

Operator-level optimization of a systolic array for Montgomery Modular Multiplication (MMM) algorithm is presented in this paper. The proposed systolic array is faster than that of C.D. Walter by 40%. Compared with J.B. Shin et al.'s, it is 25% faster.

## 1. Introduction

Reducing the length of sequential operations is an important task when designing special purpose hardwares such as modular multipliers[1]. Based on C.D. Walter's systolic array, J.B. Shin et al. improved it by decomposing arithmetic operations into boolean ones[2]. Though the decomposition resulted in a faster modular multiplier by 20% than Walter's, the parallelism among the boolean operators remains unused. Therefore, we propose a further improved modular multiplier, which utilizes *operator-level parallelism* of boolean operations.

## 2. MMM and Walter's systolic array

Walter's systolic array is a parallel implementation of MMM (Montgomery modular mutiplication algorithm). Given three numbers $A$, $B$, and $M$, where $A = \sum_{i=0}^{i=n-1} A[i]2^i$, $B = \sum_{i=0}^{i=n-1} B[i]2^i$, and $M = \sum_{i=0}^{i=n-1} M[i]2^i$, the base of MMM, $P_n$, is represented by the following algorithm:

$P_0 = 0$
**for** $i = 0$ to $n-1$ **do**
  $Q[i] \leftarrow (P_i + A[i] \times B) \bmod 2$
  $P_{i+1} = (P_i + A[i] \times B + Q[i] \times M)$ div 2
**end for**

Assuming that $R = 2^n$, we can obtain modular multiplication from $P_n$ because $P_n = ABR^{-1} \bmod M$. Walter transformed this algorithm into a systolic array by the following recursive equation, which is obtained by transforming MMM

$$P_{i+1}[j-1] + 2 \times \mathrm{carry}_i[j] = P_i[j] + A[i]B[j] + Q[i]M[j] + \mathrm{carry}_i[j-1] \quad (1)$$

He decomposed the value $\mathrm{carry}_i[j]$ into two different variables $C_1[i,j]$ and $C_2[i,j]$, which are one bit values. With applying carry prediction technique to those values, Walter showed that $P_i[j]$ can be obtained by five steps of sequential boolean operations.

## 3. J.B. Shin et al.'s optimization with modification

J.B. Shin et al. proposed an improved systolic array based on Walter's. By transforming the given arithmetic recursive equation into a boolean form, they found that the length of sequential operations could be further reduced by decomposing the variable $C_2[i,j]$ into two separate variables, $d_1[i,j]$ and $d_2[i,j]$. They proposed a set of recursive equations with these variables and showed that these equations can be computed at four steps, compared with the five step sequential operations of the original one.

We found that the recursive equations given in [2] have some errors. Since $A[i] \oplus B[j]$(resp. $Q[i] \oplus M[j]$) stands for $A[i] \times B[j] \bmod 2$(resp. $Q[i] \times M[j] \bmod 2$) in all equations they gave, the term $A[i] \oplus B[j]$(resp. $Q[i] \oplus M[j]$) must be transformed to $A[i] \wedge B[j]$(resp. $Q[i] \wedge M[j]$). The modified equations are shown in (2)-(5).

$$\begin{aligned} &P_{i+1}[j-1] \\ &= ((A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j])) \\ &\quad \oplus (P_i[j] \oplus C_1[i,j-1]) \end{aligned} \quad (2)$$

$$\begin{aligned} &C_1[i,j] \\ &= (((A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j])) \\ &\quad \wedge (P_i[j] \oplus C_1[i,j-1])) \\ &\quad \oplus (((A[i] \wedge B[j]) \wedge (Q[i] \wedge M[j])) \\ &\quad \oplus ((d_1[i,j-1] \vee d_2[i,j-1]) \\ &\quad \vee (P_i[j] \wedge C_1[i,j-1])) \end{aligned} \quad (3)$$

$$\begin{aligned} &d_1[i,j] \\ &= (((A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j])) \\ &\quad \wedge (P_i[j] \oplus C_1[i,j-1])) \\ &\quad \wedge (((A[i] \wedge B[j]) \wedge (Q[i] \wedge M[j])) \\ &\quad \oplus ((d_1[i,j-1] \vee d_2[i,j-1]) \\ &\quad \vee (P_i[j] \wedge C_1[i,j-1])) \end{aligned} \quad (4)$$

$$
\begin{aligned}
&d_2[i,j] \\
&= (((A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j])) \\
&\quad \wedge((d_1[i,j-1] \vee d_2[i,j-1]) \\
&\quad\ \vee(P_i[j] \wedge C_1[i,j-1])) \qquad (5)
\end{aligned}
$$

## 4. Further optimization

In this paper, we further reduce *CPL*, by utilizing *operator-level parallelism*. For example, we can evaluate the boolean expression $a \oplus b \oplus c$ either by $\mathtt{xor}(\mathtt{xor}(a,b),c)$ or by $\mathtt{xor}(a,\mathtt{xor}(b,c))$, where the function $\mathtt{xor}(x,y)$ evaluates $x \oplus y$. In this case, we can evaluate the boolean expression at one step by using the function $\mathtt{xor}(a,b,c)$ because the function $\mathtt{xor}(a,b,c)$ can be implemented as a logic gate or an instruction, which takes about the same time with $\mathtt{xor}(x,y)$ [3,4]. By using such an operator-level optimization, i.e., the optimization in evaluating a sequence of boolean operations, we can propose a systolic algorithm with *CPL* 3.

Before using this, we modify the equations in (2)-(5). The rules they use to transform arithmetic equations to boolean form are

$$A \times B \bmod 2 = A \wedge B \qquad (6)$$

$$A + B \bmod 2 = A \oplus B \qquad (7)$$

$$A + B \ \mathtt{div}\ 2 = A \wedge B \qquad (8)$$

where $A$ and $B$ are all 1-bit integers. In equations (3)-(5), the or-operator($\vee$) is used instead of xor-operator($\oplus$) only when the two input values($A$ and $B$) are not both 0 and in this case $A \vee B$ equals to $A \oplus B$. To use *operator-level* optimization we change all or-operator($\vee$)'s to xor-operator($\oplus$)'s. Optimization procedure is the following :

*step 1. Change all or-operator($\vee$)'s in Fig. ?? to xor-operator($\oplus$)'s and make an index in each gate(Fig. 1).*

*step 2. Use 3-input* xor*-gate to utilize* operator-level parallelism, *i.e.* $C_2$ *and the output of* $\mathrm{AND}_{G13}$ *is directly transferred to* $\mathrm{XOR}_{G32}$*(Fig. 2).*

*step 3. Shift down* $\mathrm{AND}_{G13}$, $\mathrm{XOR}_{G14}$, $\mathrm{XOR}_{G23}$, *and* $\mathrm{AND}_{G31}$ *by 1-level(Fig. 2 and 3).*

*step 4. Label the output of* $\mathrm{AND}_{G21}$, $\mathrm{XOR}_{G32}$, $\mathrm{XOR}_{G23}$, *and* $\mathrm{AND}_{G33}$ *which transformed to the gates in level-4 as* $s_1$, $s_2$, $s_3$, *and* $s_4$ *respectively(Fig. 3).*

*step 5. Remove the parameters* $C_1$ *and* $C_2$ *in* input *and* output *level and make* $s_1$, $s_2$, $s_3$, *and* $s_4$ *be input and output parameters(Fig. 4).*

After *step 5* we get a dependency graph with CPL 3. The systolic algorithm corresponding with Fig. 4 is the following :

$$
\begin{aligned}
&P_{i+1}[j-1] \\
&= (A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j]) \qquad (9) \\
&\quad \oplus((s_2[i,j-1] \oplus s_4[i,j-1]) \oplus P_i[j]) \\
&s_1[i,j] \\
&= (A[i] \wedge B[j]) \wedge (Q[i] \wedge M[j]) \qquad (10)
\end{aligned}
$$

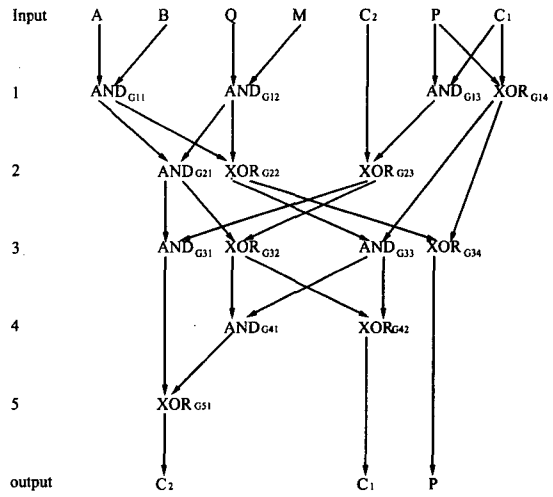


Figure 1. Result of Step 1 in optimization procedure

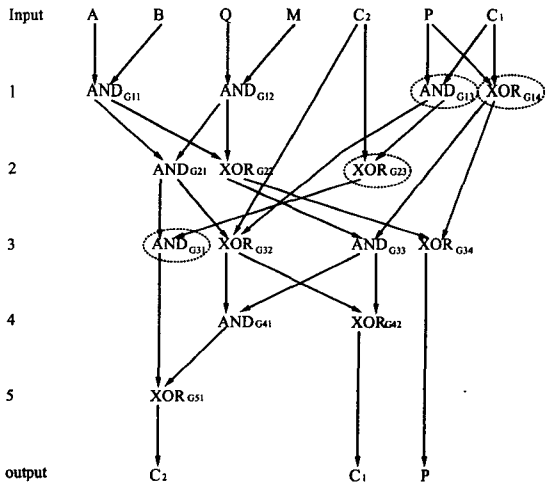


Figure 2. Result of Step 2 in optimization procedure

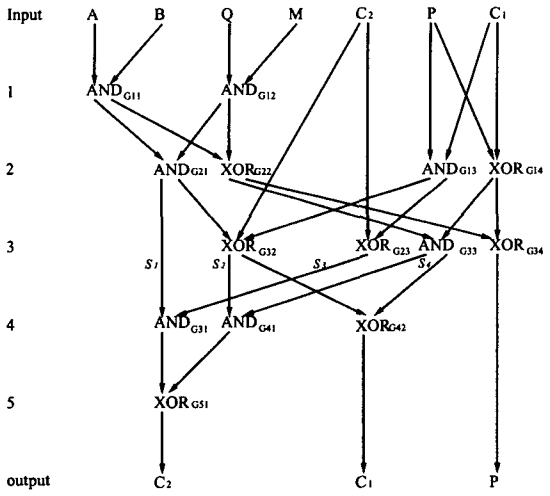


Figure 3. Result of Step 3 in optimization procedure

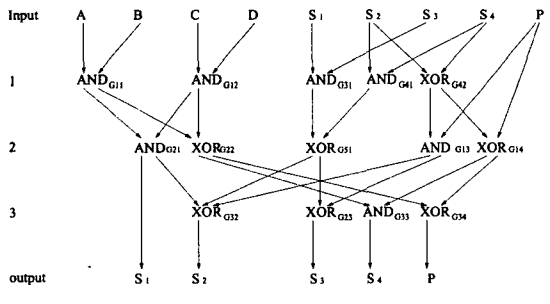


Figure 4. Final result : the dependency graph of proposed systolic algorithm with CPL 3

$$
\begin{aligned}
& s_2[i,j] \\
& = \mathbf{xor}[(A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j]), \\
& \quad ((s_1[i,j-1] \wedge s_3[i,j-1]) \\
& \quad\quad \oplus (s_2[i,j-1] \wedge s_4[i,j-1])), \\
& \quad ((s_2[i,j-1] \oplus s_4[i,j-1]) \wedge P_i[j])]
\end{aligned} \tag{11}
$$

$$
\begin{aligned}
& s_3[i,j] \\
& = ((s_1[i,j-1] \wedge s_3[i,j-1]) \\
& \quad \oplus (s_2[i,j-1] \wedge s_4[i,j-1])) \\
& \quad \oplus((s2[i,j-1] \oplus s4[i,j-1]) \wedge P_i[j])
\end{aligned} \tag{12}
$$

$$
\begin{aligned}
& s_4[i,j] \\
& = (A[i] \wedge B[j]) \oplus (Q[i] \wedge M[j]) \\
& \quad \oplus((s_2[i,j-1] \oplus s_4[i,j-1]) \oplus P_i[j])
\end{aligned} \tag{13}
$$

In the above equations, three-input xor-operator is used in equation (11) as predicted in *step 2* of transformation. The initial conditions with this systolic algorithm are $P_0[j] = 0$ for all $j$, $s_1[i,0] = 0$ for all $i$, $s_2[i,0] = 0$ for all $i$, $s_3[i,0] = 0$ for all $i$, and $s_4[i,0] = 0$ for all $i$. For the PEs with index $[i,0]$, instead of computing $P_{i+1}[-1]$, $Q[i]$ is computed with the logic, $Q[i] = P_i[0] \oplus (A[i] \wedge B[0])$.

The interpretation of equations (9)-(13) is that with the notations used in [1], $C_2[i,j] = (s_1[i,j] \wedge s_3[i,j]) \oplus (s_2[i,j] \wedge s_4[i,j])$ and $C_1[i,j] = s_2[i,j] \oplus s_4[i,j]$. For [2], $d_1[i,j] = s_1[i,j] \wedge s_3[i,j]$, $d_2[i,j] = s_2[i,j] \wedge s_4[i,j]$, and $C_1[i,j] = s_2[i,j] \oplus s_4[i,j]$.

## 5. Conclusion

The operator-level parallelism has enabled to achieve significant improvements which achieved by utilizing the parallelism of associative relation among boolean operators. The proposed systolic array is faster by 25% than that of J.B. Shin et al.'s. Compared with the original one(C.D. Walter's), it is 40% faster. Also we fixed some flaws in [2].

For further research, we could implement the proposed modular multiplier in an efficient way.

*Acknowledgement:*

This work was supported by the Korea Science and Engineering Foundation(KOSEF) through the Advanced Information Technology Research Center(AITrc).

## REFERENCES

1. C.D. Walter, Systolic modular multiplication, *IEEE Trnas. Computers,* C42(3):376-378, 1993.
2. J.B. Shin, J.K. Kim, and H. Lee-Kwang. Optimization of montgomery modular multiplication algorithm for systolic array. *The institution of Electrical Engineers Electronics Letters,* 34(19):1830-1831, 1998.
3. John Markus, *Guidebook of electronic circuits.* McGraw-Hill, 1974
4. Alfred V. Aho, Ravi Sethi, and Jeffery D. Ullman. *Compilers: principles, techniques, and tools.* Addison Wesley, 1986.