A Combined Circuit for Multiplication and Inversion in $\mathrm{GF}(2^m)$

Katsuki Kobayashi, *Student Member, IEEE*, and Naofumi Takagi, *Senior Member, IEEE*

***Abstract*—A combined circuit for multiplication and inversion in $\mathbf{GF(2^m)}$ is proposed. In order to develop a combined circuit, we start with combining the most significant bit first multiplication algorithm and the modified extended Euclid's algorithm by focusing on the similarities between them. Since almost all hardware components of the circuits are shared by multiplication and inversion, the combined circuit can be implemented with significantly smaller hardware than that necessary to implement both multiplication and inversion separately. By logic synthesis, the area of the proposed circuit is estimated to be approximately over 15% smaller than that of previously proposed combined multiplication/division circuits.**

***Index Terms*—Galois field, inversion, multiplication.**

I. Introduction

Galois field $\mathrm{GF}(2^m)$ plays important roles in error-correcting codes and cryptography [1]. Such applications can be accelerated with fast dedicated circuits for arithmetic operations in $\mathrm{GF}(2^m)$. Among the basic arithmetic operations in $\mathrm{GF}(2^m)$, multiplication and multiplicative inversion/division are much more time-consuming than addition and subtraction. Thus, various circuits for multiplication and inversion/division in $\mathrm{GF}(2^m)$ have been proposed [2]–[5].

Multiplication and inversion in $\mathrm{GF}(2^m)$ are employed for elliptic curve cryptography (ECC), which is one of the major public key cryptosystems. Since m is very large in ECC [1],[6], realization of circuits for both operations yields large area. Thus, the reduction of hardware of these circuits is important for area-restricted devices like portable ones.

In this paper, we propose a combined circuit for multiplication and inversion in $\mathrm{GF}(2^m)$. In the circuit, multiplication and inversion are carried out with the most significant bit (MSB) first multiplication algorithm and the inversion algorithm proposed in [4], which is based on the extended Euclid's algorithm, respectively. In order to develop a combined circuit based on these algorithms, we start with combining them by focusing on the similarities between them. Since almost all hardware components of the circuit are shared by multiplication and inversion, the circuit can be implemented with significantly smaller hardware than that necessary to implement both multiplication and inversion separately.

Manuscript received January 31, 2008; revised April 30, 2008. Current version published December 10, 2008. This work was supported in part by the VVLSI Design and Education Center (VDEC), The University of Tokyo, in collaboration with Synopsys, Inc. and Rohm, Inc. This paper was recommended by Associate Editor P. K. Meher.

The authors are with the Department of Information Engineering, Graduate School of Information Science, Nagoya University, Nagoya 464-8603, Japan (e-mail: katsu@takagi.i.is.nagoya-u.ac.jp; ntakagi@takagi.i.is.nagoya-u.ac.jp).

Digital Object Identifier 10.1109/TCSII.2008.2003347

Compared with previously proposed combined circuits for multiplication and division, the circuit to be proposed has several advantages. For example, although the area complexity of [7] is $O(m^2)$, that of the circuit to be proposed is $O(m)$. The circuits proposed in [8] and[9] are based on the Stein's binary greatest common divisor (GCD) algorithm for division and need to reverse the order of the coefficients of inputs and output polynomials for multiplication. Thus, the circuit proposed in [8] has extra area for such pre- and postcomputation, and how to implement such computation is not described in [9]. In contrast, the circuit proposed here does not need such computations.

We have synthesized the circuit to be proposed using 0.18 μm CMOS standard cell library, for several m's, and estimated its area and critical path delay. The area of the circuit and registers is significantly smaller than that of the previously proposed combined multiplication/division circuits [8], [9].

This paper is organized as follows. The next section explains algorithms for multiplication and inversion in $\mathrm{GF}(2^m)$ on which the circuit to be proposed is based. Section III combines these algorithms to develop a combined circuit. Section IV proposes a combined circuit and estimates its area and critical path delay by logic synthesis.

II. Preliminaries

Let

$$G(x) = x^m + g_{m-1}x^{m-1} + \cdots + g_1 x + 1$$

be an irreducible polynomial on $\mathrm{GF}(2)$, where $g_i \in \{0,1\}$. Then, we can represent an arbitrary element in $\mathrm{GF}(2^m)$ defined by $G(x)$ as

$$A(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$$

where $a_i \in \{0,1\}$.

Multiplication " $\cdot$" in $\mathrm{GF}(2^m)$ is defined as a polynomial multiplication modulo $G(x)$ on $\mathrm{GF}(2)$. Multiplicative inverse $B^{-1}(x)$ of $B(x)$ in $\mathrm{GF}(2^m)$ is defined as the element that satisfies

$$B(x) \cdot B^{-1}(x) = 1.$$

A. *MSB-First Multiplication Algorithm in* $\mathrm{GF}(2^m)$

We combine the MSB-first multiplication algorithm with the inversion algorithm shown later, because they are suited to be combined. The MSB-first multiplication algorithm in $\mathrm{GF}(2^m)$ is as follows, where b_j denotes the jth coefficient of polynomial $B(x)$. The operation " $\times$" denotes polynomial multiplication on GF(2). The notation $\deg(\cdot)$ denotes the degree of a polynomial.

[Algorithm MSB]

(*The MSB-first Multiplication Algorithm in* $\mathrm{GF}(2^m)$)

Input: $A(x), B(x)$: $\deg(A(x)), \deg(B(x)) \leq m-1$

$G(x)$: $\deg(G(x)) = m$ and irreducible

Output: $A(x) \cdot B(x) (= A(x) \times B(x) \bmod G(x))$

```
1: $P(x) := 0$;
2: **for** $i := 1$ **to** $m$ **do**
3:  **if** $b_{m-1} = 0$ **then**
4:    $P(x) := x \times P(x) \bmod G(x)$;
5:  **else**
6:    $P(x) := (x \times P(x) + A(x)) \bmod G(x)$;
7:  **end if**
8:  $B(x) := x \times B(x)$;
9: **end for**
10: *output* $P(x)$ *as the result.*
```

Note that, in the execution of the algorithm, although the degree of $B(x)$ exceeds $(m-1)$, the variables b_k's are not referred to, for $k \geq m$. Therefore, when we implement the algorithm as a circuit, an m-bit register is sufficient to store $B(x)$.

B. Yan and Sarwate's Inversion Algorithm in $\mathrm{GF}(2^m)$

For a large field, the extended Euclid's algorithm is often employed for inversion/division because it can be implemented easily. Actually, various circuits for inversion/division in $\mathrm{GF}(2^m)$ based on the extended Euclid's algorithm have been proposed. In the combined circuit to be proposed, inversion is carried out with the algorithm proposed by Yan and Sarwate [4], which is based on the extended Euclid's algorithm and shown later. Here, r_j denotes the jth coefficient of the polynomial $R(x)$. δ is a variable for finding out timing of a swap of the two polynomials, $R(x)$ and $S(x)$, for mutual division.

[Algorithm YS]

(*Yan and Sarwate's Inversion Algorithm in* $\mathrm{GF}(2^m)$ [4])

Input: $B(x)$: $\deg(B(x)) \leq m-1$

$G(x)$: $\deg(G(x)) = m$ and irreducible

Output: $B^{-1}(x)$

```
1: $R(x) := x \times B(x)$; $S(x) := G(x)$;
2: $U(x) := 1$; $V(x) := 0$; $\delta := -1$;
3: **for** $i := 1$ **to** $2m-1$ **do**
4:  **if** $r_m = 0$ **then**
5:    $R(x) := R(x) \times x$;
6:    $U(x) := U(x) \times x$;
7:  **else**
8:   **if** $\delta \geq 0$ **then**
9:     $R(x) := (R(x) + S(x)) \times x$;
10:    $U(x) := (U(x) + V(x)) \times x$;
11:   **else**
12:     $\begin{bmatrix} R(x) \\ S(x) \end{bmatrix} := \begin{bmatrix} (R(x)+S(x)) \times x \\ R(x) \end{bmatrix}$;
13:     $\begin{bmatrix} U(x) \\ V(x) \end{bmatrix} := \begin{bmatrix} (U(x)+V(x)) \times x \\ U(x) \end{bmatrix}$;
14:     $\delta := -\delta$;
15:   **end if**
16:  **end if**
17:  $\delta := \delta - 1$;
18: **end for**
19: *output* $V(x)/x^{m-1}$ *as the result.*
```

Note that, at the end of the kth iteration, all u_j and v_j are 0 for j's outside the range $[\max(0, k-m), k]$, where u_j and v_j denote the jth coefficients of the polynomials $U(x)$ and $V(x)$, respectively. Therefore, when we implement the algorithm as a circuit, two $(m+1)$-bit registers are sufficient to store $U(x)$ and $V(x)$ by using a cyclic left shift instead of a logical left shift. At the same time, we employ x^2 as the initial value of $U(x)$ so that the result $B^{-1}(x)$ will be stored in the register for $V(x)$.

TABLE I
OPERATIONS OF ALGORITHM MSB IN AN ITERATION

b_{m-1}	$B(x)$	$P(x)$
0	$B(x) := x \times B(x)$;	$P(x) := x \times P(x) \bmod G(x)$;
1	$B(x) := x \times B(x)$;	$P(x) := (x \times P(x) + A(x)) \bmod G(x)$; $\left(P(x) := x \times (P(x) + x^{-1} \times A(x)) \bmod G(x);\right)$

III. Combined Algorithm for Multiplication/Inversion in $\mathrm{GF}(2^m)$

In order to develop a combined circuit, we start with combining algorithms MSB and YS by focusing on the similarities between them. Tables I and II show the operations in an iteration of algorithms MSB and YS, respectively.

The polynomial $B(x)$ in algorithm MSB and the polynomial $R(x)$ in algorithm YS determine which operation is carried out in an iteration. Therefore, we consider merging the two polynomials first. We represent the obtained polynomial as $BR(x)$.

TABLE II
OPERATIONS OF ALGORITHM YS IN AN ITERATION

r_m	$\delta \geq 0$	$R(x), S(x)$	$U(x), V(x)$
0	*don't care*	$R(x) := x \times R(x);$	$U(x) := x \times U(x) \bmod (x^{m+1}+1);$
1	*false*	$\begin{bmatrix} R(x) \\ S(x) \end{bmatrix} := \begin{bmatrix} x \times (R(x)+S(x)) \\ R(x) \end{bmatrix};$	$\begin{bmatrix} U(x) \\ V(x) \end{bmatrix} := \begin{bmatrix} x \times (U(x)+V(x)) \bmod (x^{m+1}+1) \\ U(x) \end{bmatrix};$
	true	$R(x) := x \times (R(x)+S(x));$	$U(x) := x \times (U(x)+V(x)) \bmod (x^{m+1}+1);$

TABLE III
OPERATIONS OF ALGORITHM MULINV IN AN ITERATION

br_m	$\delta \geq 0$	$BR(x), GS(x)$	$PU(x), AV(x)$
0	*don't care*	$BR(x) := x \times BR(x);$	$PU(x) := x \times PU(x) \bmod F(x);$
1	*false*	$\begin{bmatrix} BR(x) \\ GS(x) \end{bmatrix} := \begin{bmatrix} x \times (BR(x) + \overline{mode} \times GS(x)) \\ mode \times GS(x) + \overline{mode} \times BR(x) \end{bmatrix};$	$\begin{bmatrix} PU(x) \\ AV(x) \end{bmatrix} := \begin{bmatrix} x \times (PU(x)+AV(x)) \bmod F(x) \\ PU(x) \end{bmatrix};$
	true	$BR(x) := x \times (BR(x) + \overline{mode} \times GS(x));$	$PU(x) := x \times (PU(x)+AV(x)) \bmod F(x);$

$$F(x) = \begin{cases} x^{m+1}+1 & mode = 0 \\ GS(x) \times x \,(= G(x) \times x) & mode = 1 \end{cases}$$

With Tables I and II, we also merge the polynomials $P(x)$ and $x^{-1} \times A(x)$ in algorithm MSB with the polynomials $U(x)$ and $V(x)$ in algorithm YS, respectively. We represent the obtained polynomials as $PU(x)$ and $AV(x)$, respectively. Note that, in the case of multiplication, we must modify the algorithm so that it satisfies $\delta \geq 0$ at all times.

Next, we consider merging the reduction polynomials of the two algorithms. The reduction polynomial of algorithm MSB is the irreducible polynomial with degree m that defines the field. On the other hand, the reduction polynomial of algorithm YS implemented as a circuit is the polynomial $x^{m+1}+1$ whose degree is $m+1$, because we employ the cyclic left shift instead of the logical left shift in an $(m+1)$-bit register, as described earlier. Since it is desirable for very large scale integration (VLSI) implementation that the degrees of the two reduction polynomials are identical, we employ $x \times G(x)$ instead of $G(x)$ as the reduction polynomial in algorithm MSB. Along with this modification, we also modify the other polynomials, $PU(x)$, $BR(x)$, and $AV(x)$, as multiplied by x. This modification is simply performed by changing the initial values of them with the values multiplied by x.

Finally, we apply two efficient modifications. One is that we employ $(m-1)$ and $x \times B(x) + 1$ as the initial values of δ and $BR(x)$, respectively, in the case of multiplication. By this modification, $\delta \geq 0$ is always true in the execution of the algorithm, and after m iterations, the values of $BR(x), PU(x)$, and δ will be 1, $(A(x) \cdot B(x)) \times x$, and -1, respectively. Therefore, in the $(m+1)$th iteration, the result $(A(x) \cdot B(x)) \times x$ will be transferred to the register for $AV(x)$ that stores the result of inversion. At the same time, we employ x^3 as the initial value of $PU(x)$ in the case of inversion so that the result $B^{-1}(x) \times x$ will be stored in the register for $AV(x)$, namely both the product and the inverse will appear in the same position in the register.

The other is that we also merge the two polynomials, $G(x)$ in algorithm MSB and $S(x)$ in algorithm YS, so that they can share the register. It is because the polynomial $G(x)$ in algorithm MSB is not employed in algorithm YS, and the polynomial $S(x)$ in algorithm YS is not employed in algorithm MSB. We represent the obtained polynomial as $GS(x)$.

A combined algorithm for multiplication and inversion in $\mathrm{GF}(2^m)$ is as follows, where br_j, pu_j, and gs_j denote the jth coefficient of the polynomials $BR(x), PU(x)$, and $GS(x)$, respectively. *mode* is a control signal for selection of multiplication or inversion.

[Algorithm MULINV]

(*A Combined Algorithm for Multiplication and Inversion in* $\mathrm{GF}(2^m)$)

Input: $A(x), B(x)$: $\deg(A(x)), \deg(B(x)) \leq m-1$

$G(x)$: $\deg(G(x)) = m > 3$ and irreducible

$$mode = \begin{cases} 0, & (inversion\ mode) \\ 1, & (multiplication\ mode) \end{cases}$$

Output: $B^{-1}(x) \qquad (if\ mode = 0)$

$A(x) \cdot B(x) \qquad (if\ mode = 1)$

```
1: if $mode = 0$ then/*inversion*/
2:   $BR(x) := x \times B(x); GS(x) := G(x);$
3:   $PU(x) := x^3; AV(x) := 0;$
4:   $\delta := -1; n := 2m-1;$
5: else/*multiplication*/
6:   $BR(x) := x \times B(x) + 1; GS(x) := G(x);$
7:   $PU(x) := 0; AV(x) := A(x);$
8:   $\delta := m-1; n := m+1;$
9: end if
10: for $i := 1$ to $n$ do
11:   if $br_m = 0$ then
12:     $BR(x) := BR(x) \times x;$
13:     $PU(x) := PU(x) \times x;$
14:   else
15:     if $\delta \geq 0$ then
16:       $BR(x) := (BR(x) + \overline{mode} \times GS(x)) \times x;$
```

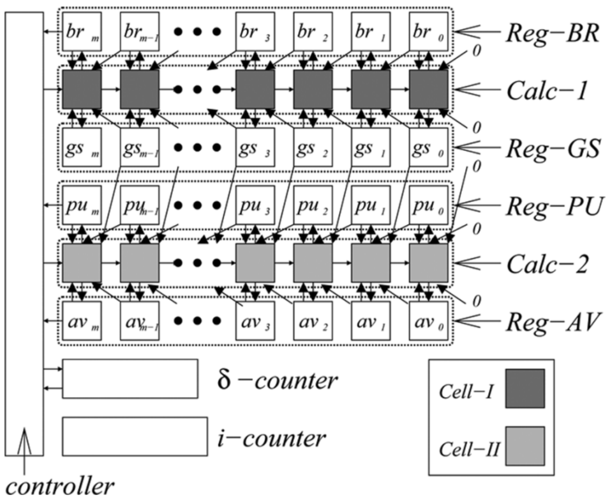

Fig. 1. Block diagram of the proposed circuit.

```
17:         $PU(x) := (PU(x) + AV(x)) \times x;$
18:     else
19:         $\begin{bmatrix} BR(x) \\ GS(x) \end{bmatrix} := \begin{bmatrix} x \times (BR(x) + GS(x)) \\ mode \times GS(x) + \overline{mode} \times BR(x) \end{bmatrix};$
20:         $\begin{bmatrix} PU(x) \\ AV(x) \end{bmatrix} := \begin{bmatrix} x \times (PU(x) + AV(x)) \\ PU(x) \end{bmatrix};$
21:         $\delta := -\delta;$
22:     end if
23:   end if
24:   if $pu_{m+1} = 1$ then
25:     $PU(x) := PU(x) + x^{m+1} + \sum_{k=1}^{m-1} (mode \wedge gs_k) x^{k+1} + \overline{mode};$
26:   end if
27:   $\delta := \delta - 1;$
28: end for
29: output $AV(x)/x$ as the result.
```

□

IV. Combined Circuit

We have designed a sequential circuit based on algorithm MULINV, which calculates operations of an iteration of the algorithm in a cycle. Table III shows the operations in an iteration in algorithm MULINV.

Fig. 1 shows a block diagram of the proposed circuit. *Calc-1* and *Calc-2* consist of $(m+1)$ *Cell-I*s and $(m+1)$ *Cell-II*s, respectively. *Calc-1* updates $BR(x)$ and $GS(x)$, and *Calc-2* updates $BR(x)$ and $AV(x)$ in accordance with Table III. *Reg-BR*, *Reg-GS*, *Reg-PU*, and *Reg-AV* are the registers for storing $BR(x)$, $GS(x)$, $PU(x)$, and $AV(x)$, which have $(m+1)$-bit width, respectively. Fig. 2(a)–(c) shows design examples of basic cells (*Cell-I, Cell-II*), and the controller in Fig. 1, respectively. The control signals are calculated by the controller as follows:

$$\begin{aligned}
reduce\_A &= \mathrm{br}_m \wedge \overline{mode} \\
reduce\_B &= \mathrm{br}_m \\
swap\_B &= \begin{cases} 1, & \mathrm{br}_m = 1 \text{ and } \delta < 0 \\ 0, & \text{otherwise} \end{cases} \\
swap\_A &= swap\_B \wedge \overline{mode} \\
reduction &= \begin{cases} (pu_m \oplus (av_m \wedge br_m)) \wedge \overline{mode}, & \text{for } pu_0 \text{ and } av_0 \\ (pu_m \oplus (av_m \wedge br_m)) \wedge mode, & \text{for the others.} \end{cases}
\end{aligned}$$

Table IV shows the comparison of the combined circuits that have $O(m)$ area complexity. The proposed circuit can be implemented with smaller area, although it needs more m clock cycles for division than the other circuits. Note that gate count in the table includes only gates for basic cells because the area of the whole circuit is almost occupied by basic cells and registers when m is large. Also note that we added m multiplexers (MUXs) to the circuits proposed in [8] and[9] to stop the operations after the circuit finished the multiplication. Otherwise, the result of multiplication will change after m clock cycles.

We described circuits based on algorithm MULINV with Verilog-HDL for several m's. For comparison, we also described circuits based on the MSB-first multiplication and the Yan and Sarwate's algorithm, and the previously proposed combined multiplication/division circuits in [8] and [9]. Note that although the circuit in [9] needs to reverse the order of the coefficients of inputs and output polynomials, we did not include the cost of such pre- and postcomputation in the estimation because they do not describe how to implement such computation.

We synthesized them with Synopsys Design Compiler using Rohm 0.18 μm CMOS standard cell library provided by VLSI Design and Education Center (VDEC), the University of Tokyo. Table V shows the synthesis results. We set 0 as area constraint and various values as critical path delay constraints, and synthesized them with Design Compiler's synthesis option "`-incremental _mapping`." The figures in the table are the best area–time product ones obtained with the synthesis. The area of the proposed circuit is approximately over 15% smaller than that of the combined multiplication/division circuits proposed in [8] and [9].

V. Concluding Remarks

We have proposed a combined circuit for multiplication and inversion in GF(2^m). In order to develop the combined circuit, we have combined the MSB-first multiplication algorithm and the Yan and Sarwate's inversion algorithm by focusing on the

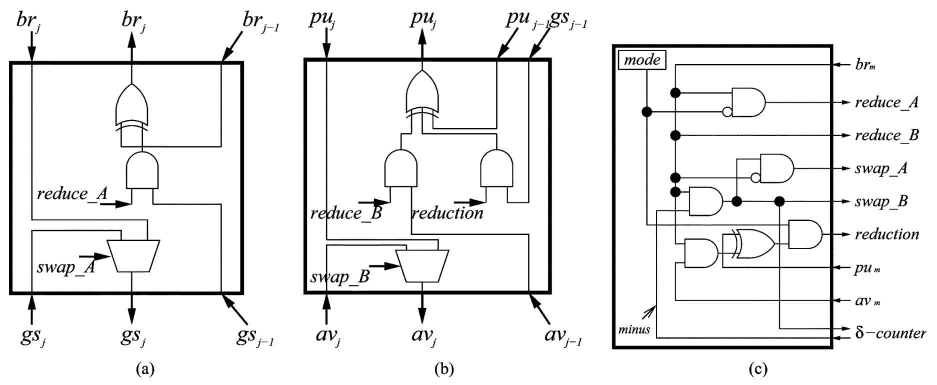

Fig. 2. Basic cells and the controller of the proposed circuit. (a) Cell-I. (b) Cell-II. (c) Controller.

TABLE IV
COMPARISON OF COMBINED CIRCUITS

	Kim et al. [9]	Lin et al. [8]	Proposed
# of Cycles per Multiplication	m	m	$m+1$
# of Cycles per Inversion	$2m-1$	$2m-1$	$2m-1$
# of Cycles per Division	$2m-1$	$2m-1$	$(3m)$
Register Width (bit)	$6m+2$	$5m+\lceil\log_2(m+1)\rceil$	$4m+7+\lceil\log_2(m+1)\rceil$
Gate Count for Basic Cells:			
2-input AND	$4m$	$6m+4$	$3m+3$
2-input XOR	$4m$	$3m$	$3m+3$
2:1 MUX	$4m$	$5m$	$2m+2$
Critical Path Delay	$2T_{\mathrm{AND}}+2T_{\mathrm{XOR}}+2T_{\mathrm{MUX}}$	$2T_{\mathrm{AND}}+2T_{\mathrm{XOR}}+T_{\mathrm{MUX}}$	$3T_{\mathrm{AND}}+2T_{\mathrm{XOR}}$
Pre- and Post-Computation	necessity	necessity	unnecessity
Positions of Results in Register	different	different	same

T_{AND}, T_{XOR}, and T_{MUX} denote the propagation delays of 2-input AND gate, 2-input XOR gate, and 2:1 MUX, respectively.

TABLE V
SYNTHESIS RESULTS (0.18 μM CMOS TECHNOLOGY)

m	Circuit	Critical Path Delay [ns]	Calc. Time [ns]			Area [mm^2]
			Mul.	Inv.	Div.	
163	Algorithm MSB+YS	1.034 / 1.172 (Mul. / Inv.)	168.5	308.9	—	0.2132
	Lin et al. [8]	1.282	209.0	416.8	416.8	0.1429
	Kim et al. [9]	1.314	214.2	427.1	427.1	0.1600
	Proposed	1.233	202.2	400.7	(602.7)	0.1238
233	Algorithm MSB+YS	1.099 / 1.189	256.1	552.9	—	0.2980
	[8]	1.318	307.0	612.7	612.7	0.2007
	[9]	1.303	303.6	605.9	605.9	0.2211
	Proposed	1.274	298.1	592.4	(890.4)	0.1710
283	Algorithm MSB+YS	1.046 / 1.197	296.0	676.3	—	0.3653
	[8]	1.324	374.6	747.9	747.9	0.2406
	[9]	1.373	388.6	775.7	775.7	0.2696
	Proposed	1.292	366.9	730.0	(1097)	0.2092
409	Algorithm MSB+YS	1.150 / 1.208	470.4	986.9	—	0.5234
	[8]	1.354	553.8	1106	1106	0.3485
	[9]	1.418	580.0	1159	1159	0.3880
	Proposed	1.326	543.7	1083	(1627)	0.2957
571	Algorithm MSB+YS	1.150 / 1.284	656.7	1464	—	0.7202
	[8]	1.341	765.6	1530	1530	0.4872
	[9]	1.433	818.2	1635	1635	0.5335
	Proposed	1.320	755.0	1506	(2261)	0.4169

similarities between them. Almost all hardware components of the proposed circuit are shared by multiplication and inversion.

The area of the proposed circuit has been estimated with logic synthesis using Rohm 0.18 μm CMOS standard cell library. The area of the circuit is significantly smaller than that necessary to implement both multiplication and inversion separately and that of the previously proposed combined multiplication/division circuits. Therefore, the proposed circuit is effective for area-restricted devices.

ACKNOWLEDGMENT

The authors would like to express sincere thanks to Prof. K. Takagi of Nagoya University for his valuable advice and technical support.

REFERENCES

[1] *Standard Specifications for Public Key Cryptography*, IEEE P1363/D13 (Draft Version 13), Nov. 1999.

[2] S. K. Jain, L. Song, and K. K. Parhi, "Efficient semisystolic architectures for finite-field arithmetic," *IEEE Trans. VLSI Syst.*, vol. 6, no. 1, pp. 101–113, Mar. 1998.

[3] S. Kumar, T. Wollinger, and C. Paar, "Optimum digit serial $\mathrm{GF}(2^m)$ multipliers for curve-based cryptography," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1306–1311, Oct. 2006.

[4] Z. Yan and D. V. Sarwate, "New systolic architectures for inversion and division in $\mathrm{GF}(2^m)$," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1514–1519, Nov. 2003.

[5] Y. Watanabe, N. Takagi, and K. Takagi, "A VLSI algorithm for division in $\mathrm{GF}(2^m)$ based on extended binary GCD algorithm," *IEICE Trans. Fund.*, vol. E85-A, no. 5, pp. 994–999, May 2002.

[6] *Digital Signature Standard (DSS)*, FIPS PUB 186-2, Jan. 2000.

[7] M. A. Hasan and V. K. Bhargava, "Bit-serial systolic divider and multiplier for finite fields $\mathrm{GF}(2^m)$," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 972–980, Aug. 1992.

[8] W.-C. Lin, J.-H. Chen, M.-D. Shieh, and C.-M. Wu, "A combined multiplication/division algorithm for efficient design of ECC over $\mathrm{GF}(2^m)$," in *Proc. 2007 IEEE Region 10 Conf. (TENCON 2007)*, Oct. 30–Nov. 2 , pp. 1–4.

[9] C. H. Kim, C. P. Hong, and S. Kwon, , L. T. Yang, O. F. Rana, B. D. Martino, and J. Dongarra, Eds., *A Novel Arithmetic Unit Over* $\mathrm{GF}(2^m)$ *for Low Cost Cryptographic Applications HPCC* Transl.:Lecture Notes in Computer Science.. New York: Springer-Verlag, 2005, vol. 3726., pp. 524–534.