논문 2007-44SD-3-6

GF(2^m) 상에서의 나눗셈연산을 위한 효율적인 시스톨릭 VLSI 구조

(Efficient systolic VLSI architecture for division in GF(2^m))

김 주 영*, 박 태 근**

(Juyoung Kim and Taegeun Park)

요 약

타원곡선 암호 시스템에서 유한체 연산은 핵심적인 부분을 차지하고 있지만 나눗셈 연산의 경우 연산 과정이 복잡하여 이를 위한 효율적인 알고리즘 및 하드웨어 설계가 필요하다. 본 논문에서는 매우 큰 소수 m을 가지는 $GF(2^m)$상에서 효율적인 면적과 연산시간을 갖는 Radix-4 시스톨릭 나눗셈기를 제안한다. 제안된 유한체 나눗셈기는 유클리드 알고리즘과 표준기저 방식을 사용하였다. 수학적 정리를 통한 효율적인 알고리즘과 Radix-4에 맞는 새로운 카운터 구조를 제안하였고 이를 VLSI 설계에 적합하도록 시스톨릭 구조를 이용하여 설계하였다. 제안된 구조는 기존의 병렬 및 직렬 나눗셈기, Digit-serial 시스톨릭 나눗셈기와 비교해서 효율적인 면적과 연산 시간을 갖는다. 본 연구에서는 $GF(2^{193})$에서 동작하는 유한체 나눗셈기를 설계하였으며, 동부아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 400㎒이다.

Abstract

The finite-field division can be applied to the elliptic curve cryptosystems. However, an efficient algorithm and the hardware design are required since the finite-field division takes much time to compute. In this paper, we propose a radix-4 systolic divider on $GF(2^m)$ with comparative area and performance. The algorithm of the proposed divider is mathematically developed and new counter structure is proposed to map on low-cost systolic cells, so that the proposed systolic architecture is suitable for VLSI design. Compared to the bit-parallel, bit-serial and digit-serial dividers, the proposed divider has relatively effective high performance and low cost. We design and synthesis $GF(2^{193})$ finite-field divider using Dongbuanam 0.18㎛ standard cell library and the maximum clock frequency is 400㎒.

Keywords: finite field divider, systolic, radix-4, VLSI

I. 서 론

유한체(finite-field)는 보통 Galois Field(GF)라고 하는데, 에러 정정 코드와 암호 시스템에서 핵심적인 역할을 수행한다[1]. $GF(2^m)$의 유한체 연산은 가감승제 연산 모두 가능한데 가감 연산이 XOR만으로 쉽게 구현이 가능한 반면 승제 연산의 경우 일반적인 이진 연산과 달리 연산 과정이 복잡하고 긴 연산 시간을 요구한다. 유한체 나눗셈 연산은 유한체 곱셈을 반복 수행함으로써 구현할 수 있으나 최근 m의 크기가 커지고 면적과 소비 전력에 제한이 따르는 경우가 많으며 실시간 처리를 요구하는 응용 분야가 증가함에 따라 전용 유한체 나눗셈기의 필요성이 대두되고 있다.

유한체 나눗셈 연산에서 가장 먼저 해야 할 일은 승수에 대한 역원을 찾는 일이다. 가장 간단하게 역원을 찾는 방법은 미리 역원을 계산하여 테이블에 저장시켜 놓고 필요할 때 찾아서 사용하는 방법이지만 m의 크기가 8을 넘어가는 경우 테이블의 크기가 m^2에 비례하여 증가하기 때문에 비효율적이다. 따라서 유한체 나눗셈기를 하드웨어로 구현하는 경우 특정한 알고리즘을 사

* 학생회원, ** 정회원, 가톨릭대학교 정보통신전자공학부
(School of Information, Communications and Electronics Engineering, The Catholic University of Korea)

※ 본 연구는 2006년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.

접수일자: 2006년11월27일, 수정완료일: 2007년2월21일

용하게 되는데 가장 잘 알려진 알고리즘이 페르마(Fermat) 알고리즘과 유클리드(Euclid) 알고리즘이다.

페르마 알고리즘은 $A/B = AB^{-1} = AB^{2m-2} = A(B(B(B\cdots B(B(B)^2)^2\cdots)^2)^2)^2$과 같이 제곱과 곱셈을 반복적으로 사용함으로서 유한체 나눗셈 연산을 수행한다. 하지만 이 방법은 $m-1$번의 유한체 제곱 연산과 $m-2$번의 유한체 곱셈 연산을 수행해야 하기 때문에 타원곡선 암호 시스템과 같이 매우 큰 소수 m을 사용하는 경우 수행시간이 m^2에 비례하므로 적용하기 어렵다[2]. 유클리드 알고리즘을 이용하여 $GF(2^m)$ 상의 임의의 두 원소를 반복적으로 서로 나눔으로써 최종적으로 얻어진 최대 공약 다항식(Greatest Common Polynomial, GCP)을 통해 유한체 나눗셈 연산을 수행할 수 있다. 이 방법은 표준 기저(standard basis) 방식을 사용하므로 VLSI 구현 시 더 효과적이다. [3]에서 제안된 유한체 나눗셈 알고리즘은 유클리드 알고리즘을 개선하여 나눗셈 연산 없이 쉬프트와 덧셈 연산만으로 초기 $5m-4$ 사이클 이후에 $2m-2$ 사이클마다 나눗셈 결과를 얻을 수 있다.

구조적 측면에서 유한체 나눗셈을 구현하는 대표적인 방법은 병렬 구조(bit-parallel)와 직렬(bit-serial) 구조가 있다. 병렬 구조는 $O(1)$의 시간 복잡도를 갖기 때문에 고속의 나눗셈을 요구할 때는 유리하지만 $O(m^2)$의 면적 복잡도를 갖기 때문에 매우 큰 m을 사용하는 경우 실제로 구현이 어렵다[4]. 반면 직렬 구조는 시간과 면적의 복잡도 모두 $O(m)$을 갖는다. 하지만 최근 저면적, 저 전력 요구 및 실시간 요구를 맞추기에는 병렬 구조의 경우 면적과 전력의 소비가 크고 직렬 구조의 경우 연산 시간이 오래 걸린다[3]. 따라서 성능과 면적의 장단점을 응용 분야에 맞게 혼합한 Digit-serial 구조가 대안으로 주목받고 있다.

본 연구에서는 타원곡선 암호 시스템에 적용이 가능한 유한체 나눗셈기를 구현하기 위하여 매우 큰 소수 m을 갖는 $GF(2^m)$에서 동작하는 유한체 나눗셈기에 대한 VLSI 구조를 제안하였다. [3]에서 제안한 유클리드 알고리즘을 사용하되 효율적인 면적과 연산 시간을 얻기 위하여 수학적 전개를 통하여 두 비트씩 연산하는 시스톨릭 Radix-4 유한체 나눗셈기를 구현하였다. 본 연구에서 제안된 구조는 HDL로 모델링되어 검증되었고 Synopsys사의 Design Compiler를 사용하여 합성하였다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 유한체 나눗셈기의 이론적 배경을 소개하고 Ⅲ장에서는 제안된 알고리즘과 그에 대한 하드웨어 구조를 설명하였다. 그리고 Ⅳ장에서는 성능을 분석하였고 마지막으로 Ⅴ장은 본 논문의 결론이다.

Ⅱ. 배 경 이 론

1. 유한체 나눗셈 알고리즘

$GF(2^m)$상에 존재하는 임의의 두 원소 $A(x)$와 $B(x)$가 있고 $G(x)$를 유한체를 생성하는 기약다항식이라고 하자. 유한체 나눗셈의 결과를 $C(x)$라고 할 때 표준기저 방식으로 각각 아래의 식과 같이 나타낼 수 있다. 이 때, 다항식의 각각의 계수 a_i, b_i, g_i, c_i는 {0,1}의 값을 갖는다.

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_0$$
$$B(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_0$$
$$G(x) = x^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_0$$
$$C(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_0$$

$C(x) = A(x)/B(x) \bmod G(x)$를 계산하기 위해서 알고리즘 1과 같이 유클리드 알고리즘을 사용할 수 있다.

알고리즘 1. [3]의 유한체 나눗셈 알고리즘
Algorithm 1. $GF(2^m)$ division algorithm in [3].

$R_0 = B(x); S_0 = G(x); U_0 = A(x); V_0 = T_0 = 0;$
$state = 0; count = 0;$
for i=1 to 2m-2 do
  $R_i = xR_{i-1}; T_i = xT_{i-1} \bmod G(x);$
  if $state = 0$ then
    $count = count + 1;$
    if $r_{m-1} = 0$ then
      $R_i = R_{i-1} + S_{i-1}; \quad S_i = R_{i-1}; \quad T_i = U_{i-1};$
$state = 1;$
    end
  else
    $count = count - 1;$
    if r_{m-1} then
      $R_i = R_{i-1} + S_{i-1}; \; T_i = T_{i-1} + U_i;$
    end
    if $count = 0$ then
      $V_i = T_i + V_{i-1}; \quad U_i = V_i; \quad V_i = U_{i-1};$
$state = 0;$
    end
  end
end (U has the result $C(x)$)

2. Digit-serial 구조

기존 병렬 구조의 하드웨어 복잡도가 크다는 문제점과 직렬 구조의 속도가 느리다는 단점을 극복하기 위한 대안으로 최근 Digit-serial 구조가 주목받고 있다.

Digit-serial 시스톨릭 어레이 구조는 병렬 구조의 인접한 PE를 L^2개 씩 묶어서 하나의 셀로 재구성하고 시스톨릭 구조를 적용함으로써 얻을 수 있다. 이 방법을 사용하는 경우 하드웨어 복잡도는 다소 증가하지만 성능은 L배 증가한다[5].

Ⅲ. 제안된 Radix-4 시스톨릭 유한체 나눗셈기

1. 알고리즘

본 연구에서는 타원곡선 암호 시스템에 사용이 가능한 유한체 나눗셈기를 구현하기 위하여 매우 큰 소수 m상의 $GF(2^m)$을 가정하였으며 $g_{m-1}=0$인 모든 기약 다항식에 대해서 적용이 가능하다. VLSI 구현과 고성능의 시스톨릭 유한체 나눗셈기를 구현에 적합하도록 유클리드 알고리즘과 표준기저를 사용한다.

또한, 효율적인 면적 대비 연산 시간을 갖는 유한체 나눗셈기를 구현하기 위해 Radix-2 유한체 나눗셈기의 PE를 단순히 두 개씩 묶는 기존의 Digit-serial 방식 (L=2)[5]을 사용하지 않고 수학적 정리를 통하여 Radix-4 유한체 나눗셈기를 위한 새로운 알고리즘을 제안한다.

알고리즘 1을 Radix-4에 맞게 정리하면 알고리즘 2와 같다. 이 때 $\cdot$와 $\oplus$는 각각 AND와 XOR 연산을 의미하며 $\bar{a}$는 a에 대한 NOT 연산을 의미한다.

2. 제안된 Radix-4 시스톨릭 어레이 구조

그림 1은 $GF(2^5)$일 경우의 DG(Dependency graph)로 Radix-4이고 유클리드알고리즘이 m-1번 반복되므로 모두 12개의 PE로 구성된다. 따라서 $GF(2^m)$일 경우 모두 $(m^2-1)/2$개의 PE로 구성된다.

PE의 입출력 신호는 그림 2와 같다. 이때, $r_{k,l}$은 R_i의 $(m-1)-2j-k$와 $(m-1)-2j-l$번째 비트를 의미하며 $s_{k,l}$, $t_{k,l}$, $u_{k,l}$, $v_{k,l}$, $g_{k,l}$도 동일한 방법으로 표현하였다.

그림 3(a)(b)(c)는 알고리즘 2에 의해서 R_i, S_i, T_i의 값을 계산하는 모듈의 회로도이다. 이때 i와 j의 범위는 각각 $0 \le i \le m-2$와 $0 \le j \le (m-1)/2$이다.

알고리즘 2. 제안된 Radix-4 유한체 나눗셈 알고리즘
Algorithm 2. Algorithm for Radix-4 finite-field divider on $GF(2^m)$.

$R_0 = B(x); S_0 = G(x); U_0 = A(x); V_0 = T_0 = 0;$
$state_0 = 0; count_0 = 0;$

for i=1 to m-1 do
$R_i = x^2 R_{i-1}; S_i = S_{i-1} \cdot state_{i-1};$
$T_i = x^2 T_{i-1} \bmod G(x);$ (1)
if $r_{m-1} = 0$ then
if $r_{m-2} = 0$ then
$S_i = S_i \oplus (S_{i-1} \cdot \overline{state_{i-1}});$
else
$S_i = S_i \oplus (R_i \cdot \overline{state_{i-1}});$
$T_i = (T_i \cdot state_{i-1}) \oplus U_{i-1};$
$R_i = R_i \oplus S_{i-1};$
end
else
$R_i = R_i \oplus x S_{i-1};$
$S_i = S_i \oplus (x R_{i-1} \cdot \overline{state_{i-1}});$
$T_i = (T_i \cdot state_{i-1}) \oplus (x U_{i-1} \bmod G(x));$ (2)
if $(r_{m-2} \oplus s_{m-1})$ then
$T_i = T_i \oplus U_{i-1};$
$R_i = R_i \oplus (x R_{i-1} \cdot \overline{state_{i-1}}) \oplus (S_{i-1} \cdot state_{i-1});$
end
end
if $state_{i-1}$ then
if $count = 1$ then
$state_i = 0; U_i = T_i \oplus V_{i-1}; V_i = U_{i-1};$
else
$U_i = U_{i-1}; V_i = V_{i-1};$
end
$count = count - 1;$
else
if $r_{m-1} = 0$ then
$count = count + 1;$
$U_i = U_{i-1}; V_i = V_{i-1};$
if r_{m-2} then
$state_{i-1} = 1;$
end
else
if $count = 0$ then
$U_i = T_i \oplus V_{i-1}; V_i = U_{i-1};$
else
$state_{i-1} = 1; U_i = U_{i-1}; V_i = V_{i-1};$
end
end
end
end (U has the result $C(x)$)

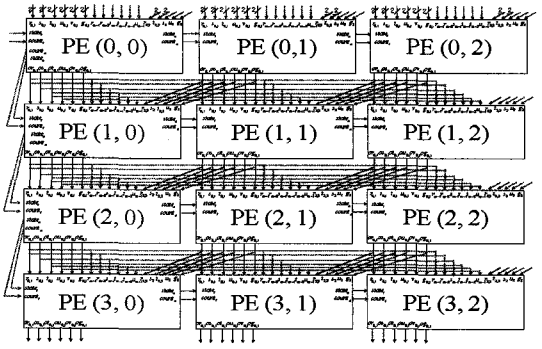


그림 1. $GF(2^5)$ 유한체 나눗셈기의 DG
Fig. 1. Two Dimensional DG of Radix-4 finite-field divider on $GF(2^5)$.

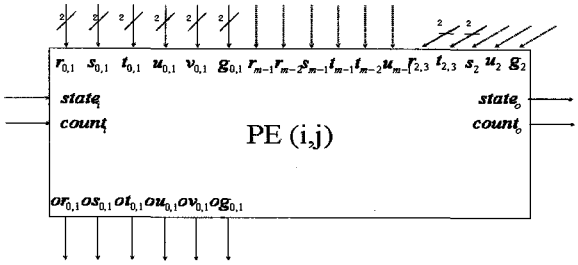


그림 2. PE의 입출력
Fig. 2. IO format of PE.

알고리즘 2에서 식(1)과 식(2)는 모듈러 연산(modular arithmetic)을 수행해야하는데 식(2)를 다음과 같이 수학적으로 정리한다. 우선 x^m을 $G(x)$를 이용하여 아래의 식(3)과 같이 구한다.

$$x^m = g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1 x + g_0 \quad (3)$$

식(3)을 이용하여 식(2)의 $xU_{i-1} \bmod G(x)$를 정리하면 식(4) 같다.

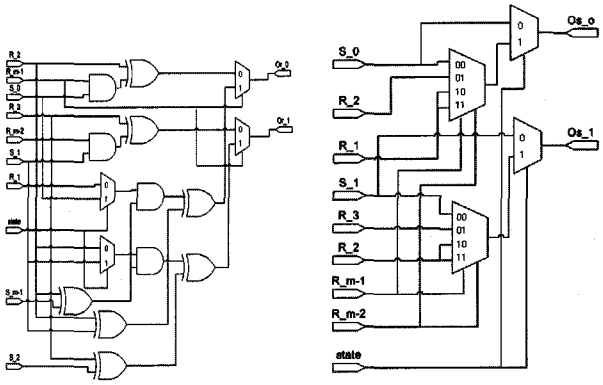
$$\begin{aligned} & xU \bmod G(x) \\ &= (u_{m-1}x^m + u_{m-2}x^{m-1} + \cdots + u_0 x) \bmod G(x) \\ &= u_{m-1}(g_{m-1}x^{m-1} + \cdots + g_0) + u_{m-2}x^{m-1} + \cdots + u_0 x \\ &= (u_{m-1}g_{m-1} + u_{m-2})x^{m-1} + \cdots + (u_{m-1}g_0) \end{aligned} \quad (4)$$

위와 유사한 방법으로 식(1)를 정리하면 아래와 같다.

$$x^{m+1} = g_{m-1}x^m + g_{m-2}x^{m-1} + \cdots + g_0 x \quad (5)$$

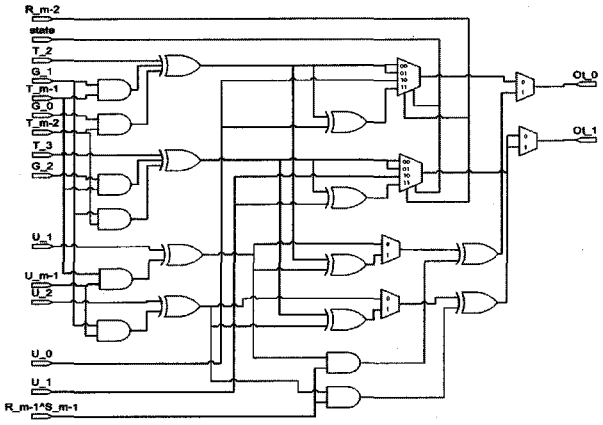
$$x^2 T \bmod G(x) = (t_{m-1}x^{m+1} + t_{m-2}x^m + \cdots + t_0 x^2) \bmod G(x) \quad (6)$$

식(6)에 식(3)과 식(5)를 대입하면 다음과 같은 식을 얻을 수 있다.



(a) 모듈 R 셀
(a) logic diagram of R module

(b) 모듈 S 셀
(b) logic diagram of S module



(c) 모듈 T 셀 회로도
(c) logic diagram in T module

그림 3. 모듈 R, S, T 셀 회로도
Fig. 3. logic diagram in R, S, and T modules.

$$\begin{aligned} & x^2 T \bmod G(x) = t_{m-1}(g_{m-1}x^m + g_{m-2}x^{m-1} + \cdots + g_0 x) + \\ & t_{m-2}(g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_0) + \cdots t_1 x^3 + t_0 x^2 \\ & = t_{m-1}g_{m-1}x^m + (t_{m-1}g_{m-2} + t_{m-2}g_{m-1} + t_{m-3})x^{m-1} + \\ & \cdots + (t_{m-1}g_0 + t_{m-2}g_1)x + t_{m-2}g_0 \end{aligned}$$

타원곡선 암호시스템에서 주로 사용하는 3항 다항식이나 5항 다항식의 공통된 특성은 계수 g_{m-1}이 항상 0을 갖는다는 것이다[1]. 따라서 위의 식을 아래의 식(7)과 같이 정리할 수 있다.

$$\begin{aligned} & x^2 T \bmod G(x) = (t_{m-1}g_{m-2} + t_{m-3})x^{m-1} \\ & + (t_{m-1}g_{m-3} + t_{m-2}g_{m-2} + t_{m-4})x^{m-2} \\ & + \cdots + (t_{m-1}g_2 + t_{m-2}g_3 + t_1)x^3 \\ & + (t_{m-1}g_1 + t_{m-2}g_2 + t_0)x^2 \\ & + (t_{m-1}g_0 + t_{m-2}g_1)x + t_{m-2}g_0 \end{aligned} \quad (7)$$

그림 3(c)의 T 모듈을 구성할 때 식(4)와 식(7)을 이용하여 효율적인 면적을 갖는 유한체 나눗셈기를 구현

할 수 있다.

알고리즘 상의 카운터 동작을 위해 $log_2(m+1)$ 비트 크기의 카운터가 PE(i,0)마다 존재해야한다. 하지만 본 연구에서는 매우 큰 소수 m을 사용하므로 사실상 카운터의 사용은 불가능하다.

알고리즘 2를 보면 카운터의 증가, 유지 및 감소 여부는 *state*와 r_{m-1}만으로 제어가 가능하며 실제로 나눗셈 연산에 영향을 주는 것은 카운터가 0과 1이 되는 경우이다. 따라서 카운터가 0과 1이 되는 경우만 확인할 수 있다면 카운터 없이도 동일한 동작을 수행할 수 있다.

그림 4는 $GF(2^5)$일 때 본 논문에서 제안하는 새로운 카운터의 동작을 보이기 위한 DG이다.

*state*와 r_{m-1}이 모두 0인 경우 카운터는 증가해야한다. PE(0,0)의 *inc*를 1로 시작하고 *state*와 r_{m-1}이 0인 경우 *check* 값은 1이 되며 PE(1,0)과 PE(1,1)이 1 값을 각각 *keep*과 *inc* 값으로 받는다. 이 때, 첫 번째 행의 다른 PE의 *check*값은 0을 갖는다. i 가 1인 두 번째 행에서도 *state*와 r_{m-1}이 0인 경우 PE(1,1)의 *check* 값은 1이 되고 두 번째 행의 다른 PE들의 *check* 값은 0이 된다. 즉, PE(i,j) 번째 PE가 1의 값을 출력한다면 i 번째 행은 카운터의 값으로 j를 갖는다는 것을 의미한다.

만약 i+1 번째 행의 *state*와 r_{m-1}의 값이 각각 0과 1이고 PE(i,j)의 값이 1을 갖는다면 PE(i+1,j)의 *check* 값은 1을 그대로 전달받아 출력한다. 이것은 i+1번째 행의 카운터 값이 그대로 j를 유지한다는 것을 의미한다.

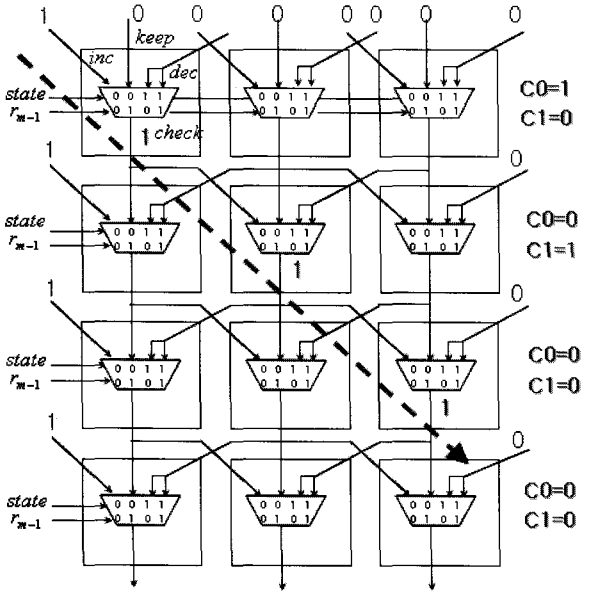


그림 4. 새로운 카운터의 DG (증가의 예)
Fig. 4. Two Dimensional DG of new counter. (in case of increasing)

또한, i+1번째 행의 state의 값이 1인 경우 PE(i,j)의 1값이 PE(i+1, j-1)의 check 값에 전달되는데 이것은 i+1번째 행의 카운터 값이 j-1로 감소되었다는 것을 의미한다.

본 논문에서 제안하는 새로운 카운터 구조는 PE(i,0)과 PE(i-1,1)의 *check* 값과 PE(i-1,0)의 *state* 값으로부터 카운터가 0인지(*check_zero* 신호, 그림4 상의 C0) 1인지(*check_one* 신호, 그림4 상의 C1)를 파악할 수 있으며 이 신호를 이용하여 아래와 같이 U, V와 *state* 값을 구하는 모듈을 구현할 수 있다.

카운터가 0이 되는 경우는 *state*가 0이고 r_{m-1}이 1일 때 카운터의 값이 0을 그대로 유지하는 경우와 *state*가 1일 때 카운터의 값 1에서 감소하여 0이 되는 경우이다. 따라서 그림 5와 같이 *check_zero* 신호를 위한 회로를 구현할 수 있다.

i 번째 행의 카운터가 1인 경우는 PE(i,0)에서 *check* 값이 1이 되는 경우이므로 이 값이 *check_one* 신호가 되며 i 행의 모든 PE에 이 값을 전달해주면 된다.

위와 같은 방법으로 구한 *check_zero*와 *check_one*

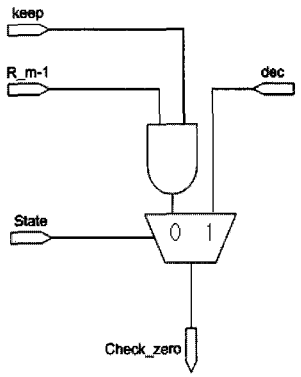
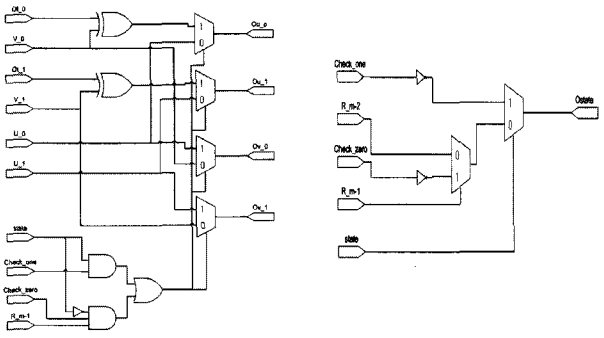


그림 5. *check_zero* 회로도
Fig. 5. logic diagram of *check_zero* module.



(a) 모듈 U와 V 셀
(a) logic diagram of U and V cell

(b) 모듈 *state* 셀
(b) logic diagram of *state* cell

그림 6. 모듈 U, V 및 state 셀의 회로도
Fig. 6. The logic diagram of U, V and state cell.

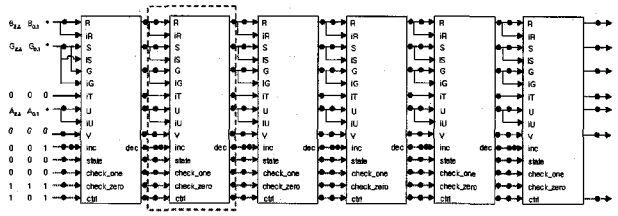


그림 7. $GF(2^7)$ Radix-4 유한체 나눗셈기의 시스톨릭 어레이 구조

Fig. 7. Systolic array architecture of Radix-4 finite-field divider in $GF(2^7)$.

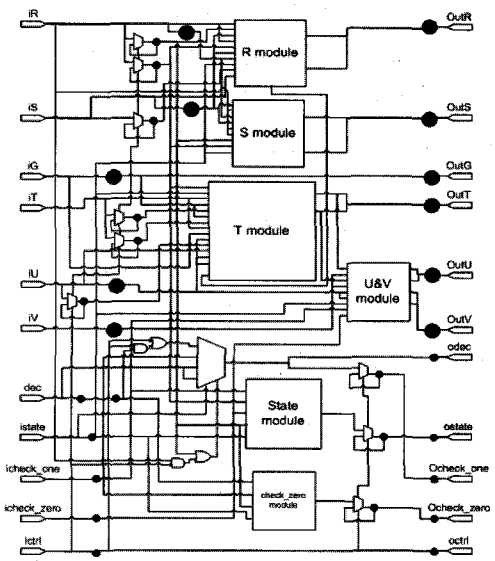


그림 8. 그림 7의 PE 회로도

Fig. 8. The logic diagram of PE.

신호를 이용하여 U, V 모듈과 *state* 모듈을 그림 6(a)(b)와 같이 구현할 수 있다.

그림 1의 DG를 바탕으로 시스톨릭 매핑하면 그림 7과 같은 시스톨릭 어레이 구조의 SFG(Signal Flow Graph)를 얻을 수 있다. 모두 m-1개의 PE셀로 구성되며 타이밍을 맞추기 위해서 각각의 PE 사이에 그림 7의 '●' 위치에 플리플롭을 삽입한다. 입출력 단의 실선은 2 비트 입출력이며 점선은 1 비트 입출력을 의미한다. 이 때 2차원 배열 상의 PE와 1차원 배열 상의 PE의 R, S, T, U, V, *state*, *check_zero* 및 새로운 카운터 모듈의 구조는 동일하다.

그림 7의 점선으로 표시한 하나의 PE 구조를 자세히 보면 그림 8과 같으며 1비트의 제어 신호 *ctrl*이 추가되었다. 또한, 각각의 PE는 m-1 사이클 동안 i 행의 값을 유지해야하기 때문에 r_{m-1}, r_{m-2}, s_{m-1}, t_{m-1}, t_{m-2}, u_{m-1}, *ostate*, *ocheck_one* 및 *ocheck_zero*의 값을 저장하기위한 멀티플렉서와 플립플롭이 추가되었다. 그림에서 큰 '●'은 2 비트 플립플롭이고 작은 '•'은 1비트 플립플롭이다.

Ⅳ. 결과 및 성능분석

제안한 알고리즘을 이용하여 나눗셈기의 동작을 검증하기 위해서 $GF(2^{193})$에서 동작하는 유한체 나눗셈기를 설계하였으며 기약다항식은 $G(x)=x^{193}+x^{15}+1$을 사용하였다. 193비트를 사용하는 타원곡선 암호 기반의 암호키는 향후 약 20년간 안전한 것으로 알려져 있기 때문에 구현 대상으로 선택하였다[6].

그림 9는 전체적인 유한체 나눗셈의 타이밍 다이어그램을 간단히 보여주기 위한 것으로 $GF(2^7)$일 경우의 동작을 보여준다. *reset* 신호가 1이 되면 나눗셈기 내의 모든 플립플롭이 리셋 되고 0이 되면 제어신호 *ctrl*은 1이 되어 새로운 나눗셈 연산이 시작됨을 알린다. *inc* 신호는 카운터의 증가를 의미하는 신호로 *reset* 신호가 0일 때 한 사이클 동안 1의 값을 갖는다. 각각의 PE는 모두 2 사이클 동안 동작하여 결과를 출력하므로 나눗셈기에 첫 번째 데이터가 입력된 후 2(m-1) 사이클 후에 가장 우측의 PE 모듈로부터 첫 번째 2비트 출력 값이 나오며 2(m+1) 사이클 동안 하나의 나눗셈 결과를 얻는다.

제안된 구조는 HDL로 모델링 되었으며 Synopsys Design Compiler를 이용하여 게이트 수준의 합성 결과를 얻었다. 동부아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 400㎒이며 PE 한 개당 약 4,856 게이트가 사용되었다.

본 논문은 타원곡선 암호시스템에 적용 가능한 기약다항식을 사용하는 다양한 구조의 기존 유한체 나눗셈기와 성능을 표 1과 같이 비교 및 평가하였다.

표 1의 하드웨어 복잡도에서 AND_2와 XOR_2는 각각 2-input AND 게이트와 2-input XOR 게이트를 의미하며 편의상 3-input XOR는 2개의 2-input XOR로 환산하여 복잡도를 계산했다. FF는 1비트 플립플롭을 의미

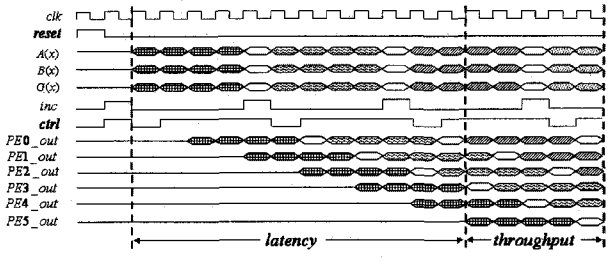


그림 9. 제안한 Radix-4 시스톨릭 어레이 구조의 $GF(2^7)$유한체 나눗셈기의 타이밍 다이어그램

Fig. 9. Timing diagram for the proposed Radix-4 systolic array divider on $GF(2^7)$.

표 1. 기존의 유한체 나눗셈기와 제안된 나눗셈기간의 성능비교
Table 1. Performance and area of the various finite-field dividers.

	[4]bit parallel	[3]bit serial	[7]digit power-sum(L=2)	[5]digit serial(L=2)	proposed
throughput rate	1	$1/m$	$2/m$	$2/m$	$2/(m+1)$
latency	$5m-4$	$5m-4$	$(3m^2-3m)/2$	$(5m/2)-1$	$2(m-1)$
# of PE	type1 : $2m-2$ type2 : $(2m-2)m$	$2m-2$	$m(m-1)/2$	m	$m-1$
HW complexity	AND_2 : $6m^2-2m-4$ XOR_2 : $8m^2-6m-2$ INV : $2m-2$ MUX_2 : $8m^2-6m-2$ FF : $28m^2-20m-8$ log_2(m+1)adder : $2m-2$ log_2(m+1) NOR : $2m-2$	AND_2 : $16(m-1)$ XOR_2 : $10(m-1)$ MUX_2 : $22(m-1)$ FF : $44(m-1)$ INV : $2m-2$	AND_2 : $6m^2-6$ XOR_2 : $6m^2-6$ MUX_2 : $3m(m-1)$ FF : $m(m-1)(16.5)$	AND_2 : $20m$ XOR_2 :$10m$ MUX_2 : $32m$ FF : $60m$ INV : $4m$	AND_2 : $18(m-1)$ XOR_2 : $19(m-1)$ OR : $3(m-1)$ INV : $5(m-1)$ MUX_2 : $40(m-1)$ FF : $39(m-1)$
critical path	$T_{AND2}+T_{XOR3}+T_{XOR2}+T_{MUX2}$	$T_{AND2}+T_{MUX2}+3T_{XOR2}$	$2T_{AND2}+4T_{XOR2}+T_{MUX2}$	$4T_{AND2}+3T_{XOR3}+3T_{XOR2}+3T_{MUX2}$	$T_{AND2}+T_{XOR3}+3T_{XOR2}+3T_{MUX2}$
IO format	parallel-in parallel-out	serial-in serial-out	2 digit-serial	2 digit-serial	2 digit-serial
control signal	0	1	1	1	1
Algorithm	Euclid	Euclid	Fermat	Euclid	Euclid

한다. 시간 복잡도에서 T_{AND2}, T_{XOR2}, T_{XOR3}, T_{MUX2}는 각각 2-input AND, 2-input XOR, 3-input XOR와 2-input MUX의 연산시간을 의미한다.

[4]에서 제안한 병렬 시스톨릭 나눗셈기는 $5m-4$사이클 후에 1사이클 동안 나눗셈 결과가 출력되지만 $O(m^2)$의 하드웨어 복잡도를 갖는다. 마찬가지로 페르마의 알고리즘을 이용하여 구현한 [7]의 Digit-serial 시스톨릭 구조도 Digit 크기 파라미터 L이 2인 경우 병렬 구조에 비해 다소 하드웨어 복잡도가 줄기는 하지만 여전히 $O(m^2)$의 복잡도를 가지며 레이턴시와 성능이 현격히 떨어진다. 때문에 매우 큰 소수 m을 사용하는 타원곡선 암호시스템에서는 사실상 사용이 불가능하다.

[3]에서 제안한 직렬 시스톨릭 나눗셈기는 $5m$-4 사이클 이후 m 사이클 동안 1개의 나눗셈 결과를 출력하는 구조로 $O(m)$의 하드웨어 복잡도를 가지며 모두 $2m$-2개의 PE로 구성된다. [4]에서 제안한 논문은 Digit-serial 시스톨릭 나눗셈기로써 L이 2인 경우 $(5m/2)$-1 사이클 이후 m 사이클 동안 모두 2개의 나눗셈 출력하고 m 개의 PE로 구성된다.

본 논문에서 제안하는 구조는 시스톨릭 어레이 구조의 Radix-4 나눗셈기로써 한번에 2 비트씩 처리하므로 [5]에서 L이 2인 Digit-serial 시스톨릭 나눗셈기와 동일한 성능을 나타낸다. 하지만 수학적 정리를 통해서 PE의 내부구조를 최적화하였으므로 레이턴시는 $2(m$-1) 사이클에 불과하다. 이는 타원곡선 암호 시스템과 같이 큰 m 을 사용하는 경우 그 차이가 커지므로 본 논문에서 제안하는 구조의 이득이 증가한다. 또한 하드웨어 복잡도를 보면 [5]의 논문과 비교 했을 때 한 개의 PE 당 XOR와 MUX의 수가 각각 9개, 8개 증가했지만 대신 플립플롭의 수가 21개 감소했다는 것을 알 수 있다. 즉 동일한 성능을 내면서도 본 논문에서 제안한 구조의 하드웨어 복잡도가 다소 감소되었다는 것을 알 수 있다. 또한 시간복잡도 역시 [5]의 구조보다 T_{AND2} 3개와 T_{XOR3} 2개만큼 더 감소하였으므로 고속의 연산이 가능하다.

V. 결 론

본 논문은 타원곡선 암호시스템에서 핵심적인 연산을 수행하는 $GF(2^m)$ 유한체 나눗셈기를 제안하였다. 효율적인 면적을 요구하면서도 빠른 나눗셈 결과를 얻기 위하여 수학적 정리를 바탕으로 2 비트씩 처리하는 Radix-4 알고리즘과 새로운 카운터구조를 제안하고 이를 통해 PE 내부구조를 최적하여 구현하였다. 제안된 Radix-4 알고리즘은 $g_{m-1}=0$인 모든 기약다항식에 대해서 차수에 상관없이 적용이 가능하다. 따라서 본 논문에서 제안한 유한체 나눗셈기는 기존의 Digit-serial 시스톨릭 어레이 구조(L=2)와 동일한 성능을 내면서도 레이턴시와 하드웨어 복잡도 및 시간 복잡도가 감소되어 면적대비 성능이 향상되었다.

본 논문에서 제안한 Radix-4 시스톨릭 어레이 구조의 유한체 나눗셈기는 면적 대비 성능 면에서 효율적이며 모듈성과 규칙성을 가지므로 적은 면적에서 고속으로 동작하는 타원곡선 암호시스템의 유한체 나눗셈기를 VLSI로 구현하기에 적합하다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] IEEE P1363, Standard Specifications for Public key Cryptography, 2000.
[2] S. W. Wei, "VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in $GF(2^m)$," IEEE Trans. Circuits and Systems, vol.44, pp.847-855, 1997.
[3] J. H. Guo and C. L. Wang, "Bit-Serial Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in $GF(2^m)$," Proc. 1997 Int. Symp. on VLSI Technology, Systems, and Applications, pp. 113-117, Taipei, Taiwan, June 1997.
[4] J. H. Guo and C. L. Wang, "Hardware-efficient systolic architecture for inversion and division in $GF(2^m)$," IEE Proc. Comput. Digit. Tech., vol. 145, No. 4, pp.272-178, July 1998.
[5] J. H. Guo and C. L. Wang, "Novel digit-serial systolic array implementation of Euclid's algorithm for division in $GF(2^m)$," Proc. 1998 IEEE int. symp. on Circuits and Systems, vol. 2, pp.478-481, 1998.
[6] 이찬호, 이정호, "ECC 연산을 위한 가변 연산 구조를 갖는 정규기저 곱셈기와 역원기," 대한전자공학회논문지 제 40권 SD편 제 12호, 2003.
[7] W. H. Lee, K. J. Lee and K. Y. Yoo, "New Digit-Serial Systolic Arrays for Power-Sum and Division Operatin in $GF(2^m)$," ICCSA 2004, LNCS 3045, pp.638-647, 2004.

저 자 소 개

김 주 영(학생회원)
2005년 가톨릭대학교 정보통신공학과 졸업.
2005년~현재 가톨릭대학교 컴퓨터공학과 석사과정
<주관심분야 : VLSI 설계, 영상처리 및 암호시스템 등>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년~1993년 Coherent Research Inc. VLSI 설계 엔지니어.
1994년~1998년 현대전자 System IC 연구소 책임연구원.
1998년~현재 가톨릭대학교 정보통신전자공학부 부교수.
<주관심분야 : VLSI 설계, CAD, 병렬처리>