GF(2^m) 상에서 새로운 디지트 시리얼 AB^2 시스톨릭 어레이 설계 및 분석

[Design and Analysis of a Digit-Serial AB^2 Systolic Arrays in GF(2^m)]

김 남 연 † 유 기 영 ††
(Nam-Yeun Kim) (Kee-Young Yoo)

요 약 GF(2^m) 상의 공개키 암호 시스템에서 나눗셈/역원은 기본이 되는 연산으로 내부적으로 AB^2 연산을 반복적으로 수행함으로써 계산이 된다. 본 논문에서는 유한 필드 GF(2^m) 상에서 AB^2 연산을 수행하는 디지트 시리얼(digit-serial) 시스톨릭 구조를 제안하였다. L(디지트 크기)×L 크기의 디지트 시리얼 구조로 유도하기 위하여 새로운 AB^2 알고리즘을 제안하고, 그 알고리즘에서 유도된 구조의 각 셀을 분리, 인덱스 변환 시킨 후 병합하는 방법을 사용하였다. 제안된 구조는 공간-시간 복잡도를 비교할 때, 디지트 크기가 m보다 적을 때 비트 패러럴 구조에 비해 효율적이고, $(1/5)\log_2(m+1)$ 보다 적을 때 비트 시리얼(bit-serial) 구조에 비해 효율적이다. 또한, 제안된 디지트 시리얼 구조에 파이프라인 기법을 적용하면 그렇지 않은 구조에 비해 m=160, L=8 일 때 공간-시간 복잡도가 10.9% 적다. 제안된 구조는 암호 프로세서 칩 디자인의 기본 구조로 이용될 수 있고, 또한 단순성, 규칙성과 병렬성으로 인해 VLSI 구현에 적합하다.

키워드 : 공개키 암호 시스템, AB^2 알고리즘 디지트 시리얼, 시스톨릭 어레이

Abstract Among finite filed arithmetic operations, division/inverse is known as a basic operation for public-key cryptosystems over GF(2^m) and it is computed by performing the repetitive AB^2 multiplication. This paper presents a digit-serial-in-serial-out systolic architecture for performing the AB^2 operation in GF(2^m). To obtain $L\times L$ digit-serial-in-serial-out architecture, new AB^2 algorithm is proposed and partitioning, index transformation and merging the cell of the architecture, which is derived from the algorithm, are proposed. Based on the area-time product, when the digit-size of digit-serial architecture, L, is selected to be less than about m, the proposed digit-serial architecture is efficient than bit-parallel architecture, and L is selected to be less than about $(1/5)\log_2(m+1)$, the proposed is efficient than bit-serial. In addition, the area-time product complexity of pipelined digit-serial AB^2 systolic architecture is approximately 10.9% lower than that of nonpipelined one, when it is assumed that m = 160 and L = 8. Additionally, since the proposed architecture can be utilized for the basic architecture of crypto-processor and it is well suited to VLSI implementation because of its simplicity, regularity and pipelinability.

Key words : Public-key cryptosystem, Power multiplier algorithm, Digit-serial, Systolic array

1. 서 론

최근 유한 필드상의 연산은 에러-교정 코드[1], 암호학[1-3], 디지 시그널 프로세싱[5] 등의 분야에서 주목을 받고 있다. 그리고 공개키 암호 시스템[3,4,6]에서의 기본 연산인 GF(2^m)상에서의 많은 구조들은 기저들을 달리하여 개발되어져 왔는데, 그 예로 정규(normal), 듀얼(dual), 표준(standard) 기저(basis) 타입이 있다. 그 중 정규기저와 듀얼 타입의 구조들은 각각의 장점을 가진 반면에 기저 변환을 해야 한다는 단점을 지닌다. 따라서, 본 논문에서는 기저 변환이 필요 없는 표준 기저 구조를 사용한다.

AB^2 연산은 GF(2^m)상에서 공개키 암호 시스템을 위한 효율적인 연산이다. 예를 들면, 고속 회로를 디자인

† 비 회 원 : 경북대학교 컴퓨터공학과
knyeun@hanmail.net
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
yook@knu.ac.kr
논문접수 : 2003년 9월 15일
심사완료 : 2004년 12월 28일

할 때 AB^2 연산은 곱셈과 곱셈의 역원($A/B = AB^{-1}$)을 이용한 나눗셈 연산을 수행하는데 효과적으로 쓰인다. 이 때 사용된 곱셈의 역원은 $B^{-1} = B^{2^m-2} = (B(B(B\cdots(B(B)^2)^2\cdots)^2)^2)^2$과 같이 지수의 반복을 통해 얻어질 수 있다. 다음은 역원 연산의 과정을 Fermat의 이론[15]에 따라 알고리즘으로 나타낸 것이다.

1단계 : $R = B$;
2단계 : For $i = m-2$ downto 1
3단계 : $R = B \cdot R^2$;
4단계 : $R = R^2$;

이 때, 결과는 $R = B^{-1}$이고, AB^2 연산이 단계 3과 4에서 사용되고 있음을 알 수 있다.

GF(2^m)상에서 파워썸(AB^2+C) 연산을 수행하기 위한 표준 기저 시스톨릭 어레이에 대한 연구가 이미 이루어져 왔다[1]. Wei[7]는 파워썸 시스톨릭 구조에 MUX와 DEMUX를 하나씩 덧붙여 여덟 가지 다른 타입의 연산을 할 수 있도록 하였고, Wei[6]에서는 역원과 나눗셈을 위한 구조를 제안하고 있다. Wang[8]은 GF(2^m)에서 파워썸 연산을 수행하기 위한 단방향 구조를 제안하였다. 그러나 이러한 시스톨릭 어레이 파워썸 구조들은 하드웨어 복잡도가 높고 지연 시간이 길어 암호 시스템의 응용에는 적합하지 못하다. 그러므로 효율적인 연산에 대한 연구가 필요하다.

디지트 시리얼 구조는 전체 데이타 비트를 각각 몇 비트들의 디지트들로 나눈다. 데이타들은 디지트 단위로 처리되고 전송된다. 만약 데이타의 크기가 m 이고, 디지트 크기가 L 비트이면, 디지트 수는 $N=\lceil m/L \rceil$이다. 디지트 시리얼 구조는 N 시간 스텝마다 하나의 결과를 출력한다.

본 논문에서는 표준 기저를 사용한 GF(2^m)상에서의 AB^2 연산에 대한 새로운 알고리즘을 제안하고, 이 알고리즘을 통해 디지트 시리얼 시스톨릭 구조를 유도한다. 제안한 알고리즘은 병렬성을 제공하기 위해 MSB-first 구조를 사용하였고 하드웨어 복잡도와 지연 시간은 전통적인 구조들에 비해 효율적이다. 덧붙여 이 구조는 VLSI 구현에 적합하고 역원 구조에 쉽게 적용이 가능하다.

2. 알고리즘

유한필드 GF(2^m)은 2^m 개의 원소를 가진다. 본 논문에서는 GF(2^m) 상의 모든 (2^m-1)개의 0이 아닌 원소들을 표준 기저 방식으로 표현한다. GF(2^m) 상의 두 원소 A, B가 있다고 가정하고 이를 다항식 x로 나타내면 다음과 같다.

$$A(x)=\sum_{i=0}^{m-1} a_i x^i \quad , \quad B(x)=\sum_{i=0}^{m-1} b_i x^i$$

이 때 a_i, $b_i \in$ GF(2) ($0 \le i \le m-1$). GF(2^m)상의 유한 필드 원소들은 GF(2)상의 m 차수의 원시 다항식으로 표현된다. $F(x)$를 기약 다항식으로 표현하면

$$F(x)=x^m+\sum_{i=0}^{m-1} f_i x^i$$

와 같다. 이 때, $f_i \in$ GF(2) ($0 \le i \le m-1$)이다.

2.1 AB^2 알고리즘

AB^2 연산은 다음과 같은 순환식으로 유도할 수 있다.

$$R(x) = A(x)B^2(x) \bmod F(x) = \sum_{j=0}^{m-1} A(x)b_j x^{2j}$$
$$=\{(\cdots((\cdots((A(x)b_{m-1})x^2+A(x)b_{m-2})x^2+\cdots+A(x)b_{m-i})x^2+\cdots$$
$$+Ab_1)x^2+Ab_0\} \bmod F(x) \quad (1)$$

위 순환식에서 비트 패러럴 구조로 유도를 하면 양방향의 흐름이 존재하여 $L\times L$ 크기의 디지트 시리얼 구조로 유도할 수가 없다. 본 논문에서는 이 문제를 해결하기 위하여 AB^2 연산부분과 모듈러 리덕션 부분으로 분리하여 알고리즘을 유도하였다.

[**알고리즘 1**] $A(x)B^2(x) \bmod F(x)$의 제안된 알고리즘

입력 : $A(x)$, $B(x)$ and $F(x)$
출력 : $R(x) = A(x)B^2(x) \bmod F(x)$
$P_0(x) = 0$
Step 1: for $i = 1$ to $m-1$
Step 2: $D_i(x) = (P_{i-1}(x) + A(x)b_{m-i})\, x^2$
Step 3: $P_i(x) = D_i(x) \bmod F(x)$
Step 4: $R(x) = D_m(x) = P_{m-1}(x) + A(x)b_0$

알고리즘 1은 비트 레벨로 다시 쓰여질 수 있는데, 그 중간 결과값을 GF(2) 상의 계수를 가진 m+1 과 m-1차의 다항식 $D_i(x)$와 $P_i(x)$로 각각 정의하였다.

$$D_i(x) = d^i_{m-1}x^{m+1} + d^i_{m-2}x^m + \cdots + d^i_1 x^3 + d^i_0 x^2$$
$$(1 \le i \le m-1) \quad (2)$$

$$D_m(x) = d^m_{m-1}x^{m-1} + d^m_{m-2}x^{m-2} + \cdots + d^m_1 x^1 + d^m_0$$
$$(i = m) \quad (3)$$

$$P_i(x) = p^i_{m-1}x^{m-1} + p^i_{m-2}x^{m-2} + \cdots + p^i_1 x^1 + p^i_0$$
$$(1 \le i \le m) \quad (4)$$

알고리즘1에서 일반항(i = 1 to $m-1$)은 (5)식과 같이 비트 레벨로 나타낼 수 있고 이를 모듈러 리덕션 하기 위해 (6)과 (7)식을 이용하였다.

$$D_i(x) = (P_{i-1}(x) + A(x)b_{m-i})x^2$$
$$= [\sum_{k=0}^{m-1}(p_k^{i-1} + a_k b_{m-i})x^k x^2] \quad (5)$$

$$x^m \bmod F(x)\ (f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \cdots + f_1 x + f_0 \quad (6)$$

$$x^{m+1} \bmod F(x) = x^m\, x \bmod F(x)$$
$$= (f_{m-1}f_{m-1} + f_{m-2})\, x^{m-1} + (f_{m-1}f_{m-2} + f_{m-3})\, x^{m-2} + \cdots +$$

$$(f_{m-1}f_1 + f_0)\,x + f_{m-1}f_0$$
$$\equiv f'_{m-1}x^{m-1} + f'_{m-2}x^{m-2} + \cdots + f'_1x + f'_0$$

where, $f'_i \in GF(2)$

즉, (5)식에 (6)과 (7)식을 대입하면 다음과 같이 모듈러 리덕션 연산을 할 수 있다.

$$P_i(x) = (d^i_{m-1}x^{m+1} + d^i_{m-2}x^m + d^i_{m-3}x^{m-1} + \cdots + d^i_1x^3 + d^i_0x^2) \bmod F(x)$$
$$= d^i_{m-1}(f'_{m-1}x^{m-1} + \cdots + f'_1x + f'_0) + d^i_{m-2}(f_{m-1}x^{m-1} + \cdots + f_1x + f_0) + \cdots + d^i_1x^3 + d^i_0x^2$$
$$= (d^i_{m-1}f'_{m-1} + d^i_{m-2}f_{m-1} + d^i_{m-3})x^{m-1} + (d^i_{m-1}f'_{m-2} + d^i_{m-2}f_{m-2} + d^i_{m-4})x^{m-2} + \cdots + (d^i_{m-1}f'_1 + d^i_{m-2}f_1)x + (d^i_{m-1}f'_0 + d^i_{m-2}f_0)$$
$$= \sum_{k=0}^{m-1} p^i_k x^k \qquad (8)$$

따라서, 비트 레벨 일반항(i=1 to m-1)을 다음과 같이 유도할 수 있다.

$$\begin{cases} d^i_k = p^{i-1}_k + a_k b_{m-i} \\ p^i_k = d^i_{m-1}f'_k + d^i_{m-2}f_k + d^i_{k-2} \end{cases}$$

(i = 1, 2, ···, m-1, k = m-1, m-2, ···, 0) (9)

이 때, p^0_k = 0, (k = m-1, m-2, ···, 0)이고, d^i_{-1}=0 (i = 1, 2, ···, m-1)이다.

위와 같은 방법으로 마지막항(i = m)을 비트 레벨로 나타내면 다음과 같다.

$$D_m(x) = P_{m-1}(x) + A(x)b_0 = \sum_{k=0}^{m-1} p^{m-1}_k x^k + \sum_{k=0}^{m-1} a_k b_0 x^k$$
$$= \sum_{k=0}^{m-1} (p^{m-1}_k + a_k b_0)x^k \qquad (10)$$

(3)과 (10)식을 비교하면 다음과 같은 비트 레벨의 식을 얻을 수 있다.

$$d^m_k = p^{m-1}_k + a_k b_0 \ , \ (k = m\text{-}1,\ m\text{-}2,\ \cdots,\ 0) \qquad (11)$$

따라서, $[A(x)B^2(x) \bmod F(x)]$ 연산은 (9)와 (11)식을 사용하여 효율적으로 이루어질 수 있다.

2.2 자료의존 그래프

위 알고리즘의 수행을 2차원 평면에 표현한 그래프는 그림 1과 같다[15].. 이 때, m=4이고, 각 인덱스 점

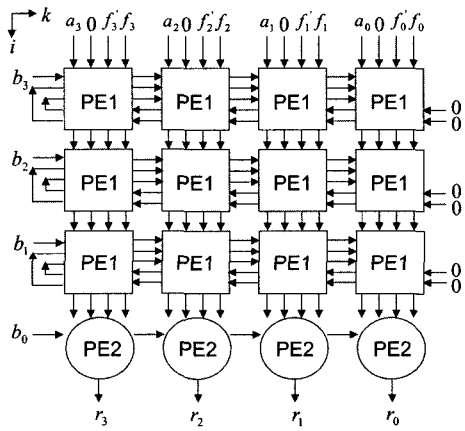
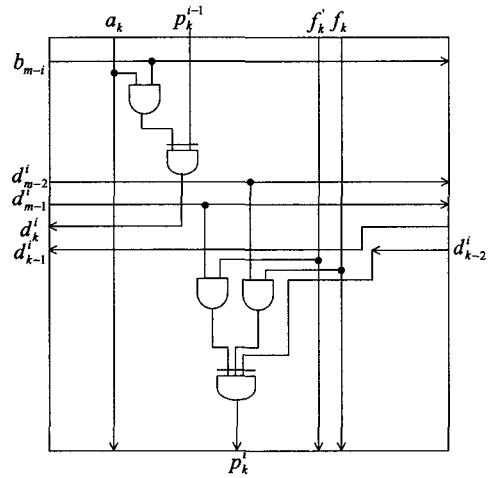
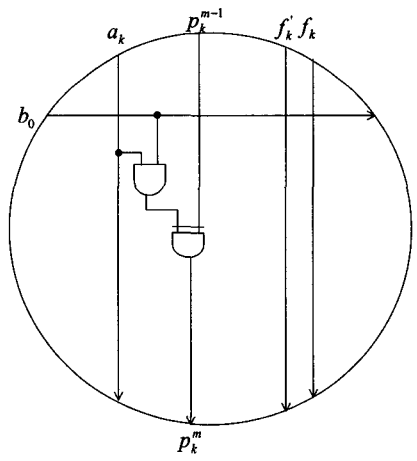


그림 1 GF(2^4)상의 AB^2 연산을 위한 자료 의존 그래프



(a) PE1



(b) PE2

그림 2

(index point) (i, k)은 i = 1, 2, ···, m and k = m-1, m-2, ···, 1, 0와 같다. 또한, 그림 2(a)의 PE1(Processing Element1)은 기본적인 셀들의 논리 회로를 표현하고, 그림 2(b)의 PE2는 마지막 행에 위치한 셀들의 논리 회로를 보여준다.

마지막 행의 셀들은 모듈러 리덕션 과정없이 오직 d_k^m만 계산된다는 사실은 주목할 만하다. 그림 1에서 보는 것처럼 마지막 행에 위치한 셀들의 회로가 간단하기 때문에 이전의 구조들과 비교해 볼 때 전체 셀 복잡도를 줄일 수 있다.

3. 디지트 시리얼 시스톨릭 곱셈기

본 논문에서는 디지트 개수 $N=\lceil m/L \rceil$ 을 정수라고 가정한다. 디지트 크기가 L 인 디지트 시리얼 시스톨릭 구조를 만들기 위해, 일반적으로 계산점을 $L\times L$로 묶는 방법을 사용한다. 이러한 방법을 그림1의 자료의존 그래프에 적용할 경우 수평 방향으로 양방향 데이타 흐름이 있기 때문에, 오른쪽으로 투영시킬 수 없다. 이러한 문제를 해결하기 위해 먼저, 마지막 행의 셀들을 제외한 나머지 셀들의 연산을 d_k^i의 계산부분과 p_k^i 계산부분으로 분리하였다. 분리된 자료 의존 그래프의 인덱스를 변환한 후, 인덱스 변환된 자료의존 그래프를 디지트 크기로 다시 분할하였다.

즉, 그림 1의 자료의존 그래프에서 수평 방향으로의 양방향 데이타 흐름을 피하기 위해 d_k^i의 계산부분과 p_k^i 계산부분으로 셀을 분할 후 d_k^i을 계산하는 셀들은 셀 인덱스 (i, k)를 $(i, -2i+k+2)$으로, 그리고 p_k^i을 계산하는 셀들은 (i, k)를 $(i, -2i+k)$으로 인덱스 변환을 한다. 인덱스 변환된 자료의존 그래프는 그림 1의 자료의존 그래프와 동일한 기능을 수행한다. 그림 1을 분할하여 인덱스 변환을 한 결과는 그림 3과 같다.

그림 3은 $(m^2+2m-2)/2$개의 셀들로 구성되는데, 그것은 $(3m-2)/2$개의 PE_A cells, $(m^2-3m+2)/2$개의 PE_B 셀들 그리고 m-1개의 PE_C 셀들로 구성된 것이며 그림에서 PE_A, PE_B and PE_C 셀들은 각각 원, 사각형, 삼각형으로 표현된다.

PE_A, PE_B, PE_C 셀들의 연산은 다음과 같은 식으로 나타낼 수 있다 :

PE_A	$\begin{cases} d_k^i = p_k^{i-1} \oplus a_k b_{m-i} \\ d_{k-1}^i = p_{k-1}^{i-1} \oplus a_{k-1} b_{m-i} \end{cases}$
PE_B	$\begin{cases} d_{k-2}^i = p_{k-2}^{i-1} \oplus a_{k-2} b_{m-i} \\ p_k^i = d_{m-1}^i f_k' \oplus d_{m-2}^i f_k \oplus d_{k-2}^i \\ d_{k-3}^i = p_{k-3}^{i-1} \oplus a_{k-3} b_{m-i} \\ p_{k-1}^i = d_{m-1}^i f_{k-1}' \oplus d_{m-2}^i f_{k-1} \oplus d_{k-3}^i \\ \vdots \end{cases}$
PE_C	$\begin{cases} p_1^i = d_{m-1}^i f_1' \oplus d_{m-2}^i f_1 \\ p_0^i = d_{m-1}^i f_0' \oplus d_{m-2}^i f_0 \end{cases}$

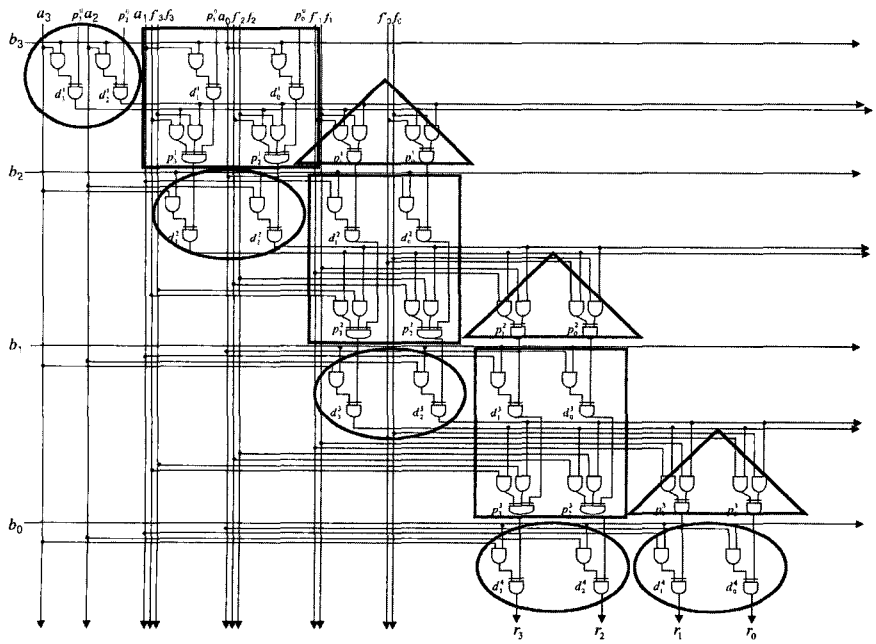


그림 3 인덱스 변환된 자료의존 그래프

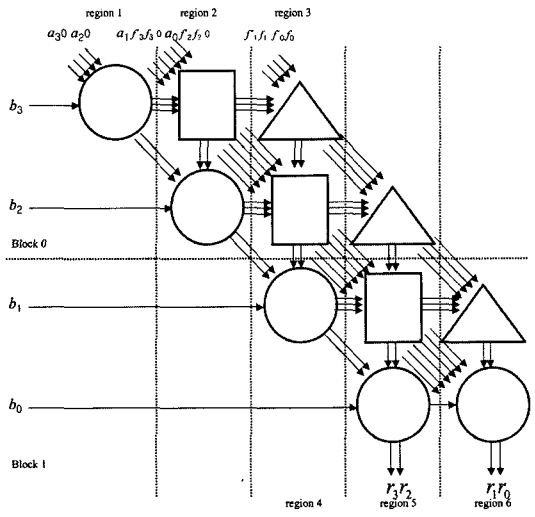


그림 4 그림 3을 L=2로 분할하는 과정

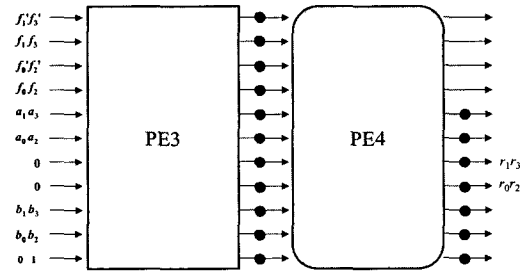


그림 5 GF(2^4) 상에서 L=2인 AB^2 연산을 위한 디지트-시리얼 시스톨릭 구조

그림 4는 m=4이고 L=2로 분할하는 과정을 보여준다. 디지트 크기가 L인 GF(2^m) 상의 디지트 시리얼 시스톨릭 곱셈기를 설계하기 위해, 먼저 그림 3을 수평방향으로 m/L 부분으로 분할한다. 이 때, 각 부분은 L 개의 열과 (m+2L)개의 행으로 구성된다. 그러나, 마지막 블록은 L 개의 열과 (m+2(L-1))개의 열로 구성된다. 다음으로 수직방향으로 $\lceil (m+2L)/L \rceil$ 개의 영역으로 분할을 하면, 첫번째 영역은 L/2개의 PE$_A$ 셀과 $\sum_{i=1}^{2i<L}(L/2-i)$ PE$_B$ 셀로, 두번째 영역은 L/2개의 PE$_A$ 셀과 ((L/2)^2+ $\sum_{i=1}^{2i<L}(L/2-i)$)개의 PE$_B$ 셀로, 마지막에서 두번째 영역은 L/2개의 PE$_C$ 셀과 ((L/2)^2 + $\sum_{i=1}^{2i<L}(L/2-i)$)개의 PE$_B$ 셀로, 그리고 마지막 영역은 L/2개의 PE$_C$ 셀과 $\sum_{i=1}^{2i<L}(L/2-i)$개의 PE$_B$ 셀로, 나머지 모든 셀들은 L^2/2 개의 PE$_B$ 셀로 구성된다.

[8]의 투영 절차에 따라 오른쪽으로 그림 4의 자료의 존 그래프를 투영시키고 컷-셋 시스톨릭화 기법(cut-set systolization techniques)[8]을 이용하여, 그림 5와 같은 디지트 시리얼 시스톨릭 어레이를 유도할 수 있다. 이 때, '·'은 1 비트 클럭 사이클 지연 요소이다. 이 구조는 그림 6과 같이 N-1 개의 기본셀과 그림 7과 같은 하나의 셀로 구성된다.

이것은 길이 N 인 10000(((0의 컨트롤 시퀀스에 의해 컨트롤된다. 결과값의 계수인 r_iS은 클럭당 L-비트의 비율로 구조의 오른쪽방향으로 출력된다. L개의 임시 결과값 p_iS와 b_iS 값은 그림 5의 모든 셀들에 broadcast 되어야 하므로, 그림 6에 3L 개의 멀티플렉서와 3L 개의 한 비트 래치가 추가 되고, 그림 7에 1+3(L-1)개의 멀티플렉서와 1+3(L-1)개의 한 비트 래치가 추가 되어야 한다. 컨트롤 시그널이 1일 때, L개의 임시 결과값 p_iS와 b_iS 값이 래치된다. 또한, $p_k^i = d_{m-1}^i f_k' + d_{m-2}^i f_k + \overline{CS} \cdot x = d_{m-1}^i f_k' + d_{m-2}^i f_k + 0 = d_{m-1}^i f_k' + d_{m-2}^i f_k$ 연산에서 모르는 값 x의 입력에 대해 0값을 유지하기 위해 그림 6에 L^2 개의 2-입력 AND 게이트와 L^2 개의 NOT 게이트가 추가되고, 그림 7에 L(L-1)개의 2-입력 AND 게이트와 L(L-1) NOT 게이트가 추가되어야 한다.

그림 5의 디지트 시리얼 시스톨릭 구조에 대해, 최대 임계경로는 T_{max} = L (T_{AND2} + T_{NOT} + T_{XOR2} + T_{XOR3} + T_{MUX})인데, 이 때, T_{ANDi}, T_{XORi}, T_{NOT} and T_{MUX}는 각각 i-입력 AND 게이트, i-입력 XOR 게이트, NOT 게이트 그리고 2-to-1 멀티플렉서를 각각 나타낸다. 그러나, 디지트 크기 L이 커지면, 임계경로도 길어진다는 단점이 있으므로, 이를 극복하기 위해 각 셀에 파이프라인 기법을 적용하였다. 따라서, 최대 임계경로를 줄여, L이 커질 때도 높은 클럭 비율을 유지하도록 하였다.

[9]의 컷 이론을 적용하여, 그림 6과 그림 7의 점선 라인에 각각 5L+1개의 한 비트 래치를 추가하여 쉽게 파이프라인 시켰다. 그 결과, 지연시간은 (5m-4)/2가 되었고, 최대 지연시간은 T'_{max} = T_{AND2} + T_{NOT} + T_{XOR2} + T_{XOR3} + T_{MUX} 이므로, Daniel[12]의 가정에 따라, 파이프라인된 디지트 시리얼 시스톨릭 구조는 그렇지 않은 구조에 비해 공간-시간 복잡도가 m=160이고 L=8인 경우, 10.9% 낮다.

4. 분 석

본 논문은 비트 레벨 구조를 디지트 시리얼 구조로 변환하여 구현하였다. 디지트 시리얼 구조의 장점은 성능과 하드웨어 복잡도 사이의 trade-off 관계를 향상시킨다[1].

비교를 위하여 본 논문에서는 다음의 가정을 따른다 : 1) 3-입력 과 4-입력 XOR 게이트는 두개와 세개의 2-

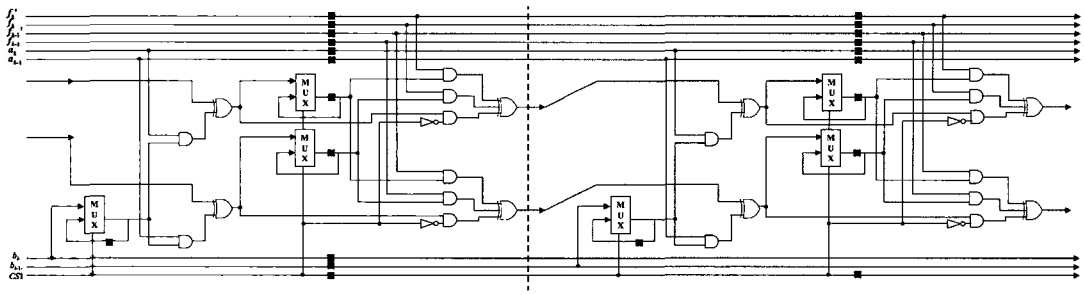


그림 6 그림 5의 PE3 회로

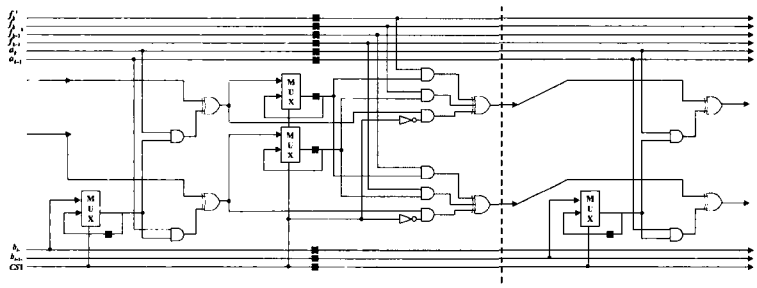


그림 7 그림 5의 PE4 회로

입력 XOR를 사용하여 구성되었다[11]. 2) 2-입력 AND 게이트, 2-입력 XOR 게이트, 2-to-1 MUX, 1 비트 래치는 6, 14, 20, 8 개의 레지스터로 각각 구성된다. 3) 2-입력 AND 게이트, 2-입력 XOR 게이트, 2-to-1 MUX, 1-비트 Latch는 각각 2.4ns, 4.2ns, 5.8ns, 1.4ns의 게이트 지연시간을 가진다[12].

표 1은 제안된 AB^2 디지트 시리얼 구조와 비트 패러럴 구조[13], 비트 시리얼[14] 구조를 비교하고 있다.

표 1 GF(2^m) 상의 구조 비교

Item \ Circuit	Wang *et al* [13] [Fig.3]	Kim [14] MPOM	DSPM	Pipelined DSPM
Architecture	Systolic	LFSR	Systolic	Systolic
Irreducible Polynomial	General	AOP	General	General
I/O format	Bit-parallel	Bit-serial	Digit-serial	Digit-serial
Number of cells	$m^2/2$	m	$\lceil m/L \rceil$	$\lceil m/L \rceil$
Function	$AB^2 + C$	AB^2	AB^2	AB^2
Throughput	1	$1/(m+1)$	L/m	L/m
Maximum cell delay	$T_{AND2}+3T_{XOR}$	$T_{AND}+log_2(m+1)T_{XOR}+T_{MUX}+T_{IFF}$	$L\ (T_{AND2} + T_{NOT} + T_{XOR2} + T_{XOR3} + T_{MUX})$	$T_{AND2} + T_{NOT} + T_{XOR2} + T_{XOR3} + T_{MUX}$
Latency	$2m+m/2$	$2m-1$	$\lceil (m+2(L-1))/L \rceil + 3(m/L-1)$	$(5m-4)/2$
Circuit Complexity				
AND gates	$3m^2$	$2m-1$	$\lceil m/L \rceil \cdot (4L^2-3$	$\lceil m/L \rceil_{(4L^2-3}$
XOR gates	$3m^2$	$2m-2$	$\lceil m/L \rceil \cdot (3L^2-2)$	$\lceil m/L \rceil_{(3L^2\ 2}$
Latches	$8.5m^2$	$3m+2$ (Registers)	$\lceil m/L \rceil_{(3L^2+5L)-2}$	$\lceil m/L \rceil_{(7L^2+5L-3)-4L-2}$
Mux		$m+3$	$\lceil m/L \rceil \cdot 3L-2$	$\lceil m/L \rceil_{(3L-2}$
NOT gates			$\lceil m/L \rceil_{L^2-L}$	$\lceil m/L \rceil_{L^2-L}$
No. of CS	-	-	1	1

Wang[13]이 제안한 구조의 지연시간은 $2m+m/2$ 이고, 반면에 제안된 구조의 지연시간은 $\lceil (m+2(L-1))/L \rceil + 3(m/L-1)$이다. 따라서, m=160이고, L=2일 때 제안된 구조의 지연시간은 Wang의 구조에 비해 20% 정도 낮다. 또한, Kim[14]의 LFSR 구조의 시간 복잡도는 $O(m \log_2 m)$로써 이 역시 제안된 디지트 시리얼 구조에 비해 높고, broadcast 문제가 있어 제안된 시스톨릭 구조에 비해 비효율적이다. 표 1에 의하면, Wang 구조의 전체 셀 복잡도는 $3m^2$ AND + $3m^2$ XOR + $8.5m^2$ Latches 임에 반해, 제안된 구조는 $(\lceil m/L \rceil 4L^2 - L)$AND + $(\lceil m/L \rceil 3L^2-2L)$XOR + $(\lceil m/L \rceil (3L^2+5L)-2)$ Latches + $(\lceil m/L \rceil (3L-2))$Mux + $(\lceil m/L \rceil L^2-L)$NOT이다. 이 때, 위 두번째 가정에 따르면, L이 약 $1.42m$ 보다 적을 때 제안한 구조는 Wang의 구조에 비해 전체 셀 복잡도가 적다.

표 2는 제안된 구조와 Wang과 Kim 구조의 공간-시간 복잡도를 비교하고 있다. 만약 L을 m보다 적은 수를 사용했을 때, 제안된 구조는 Wang의 구조에 비해 효율적이고, L을 $(1/5)\log_2(m+1)$ 보다 적은 수를 사용했을 때, 제안된 구조는 Kim의 구조에 비해 효율적이다.

또한, 앞서 살펴 본 대로, 파이프라인된 디지트 시리얼 시스톨릭 구조는 그렇지 않은 구조에 비해 공간-시간 복잡도가 m=160이고 L=8인 경우, 10.9% 낮다.

표 2 시간-공간 복잡도 비교 (ϕ : 트랜지스터수 △ : 게이트 지연시간 (ns) by Daniel D. Gajski [12])

Item / Circuit	Area-Time Product Complexity
Wang [13]	$128m^2\phi * 16.5m\triangle = 2112m^3\phi\triangle$
Kim [14]	$(114m+62)\phi * m(32.4+8.4\ log_2\ (m+1))\triangle$ $= ((3693.6+957.6\ log_2\ (m+1))m^2 + (2008.8 + 520.8\ log_2\ (m+1))m)\phi\triangle$
DSPM	$(90mL+100m-46L-40)\phi * (65.6m)\triangle$ $= ((5904L+ 6560)\ m^2)\phi\triangle$
Pipelined -DSPM	$(122mL+100m-78L-40\lceil m/L-40)\phi * (41m-32.8)\triangle = ((5002L+4100)\ m^2)\phi\triangle$

5. 결 론

본 논문에서는 디지트 시리얼 시스톨릭 AB^2 구조를 제안하였다. 제안한 구조는 공간-시간 복잡도를 비교할 때, 디지트 크기가 m보다 적을 때 비트 패러럴 구조에 비해 효율적이고, $(1/5)\log2(m+1)$ 보다 적을 때 비트 시리얼(bit-serial) 구조에 비해 효율적이었다. 또한, 제안된 디지트 시리얼 구조에 파이프라인 기법을 적용하면 그렇지 않은 구조에 비해 m=160, L=8 일 때 공간-시간 복잡도가 10.9% 적었다. 즉, 최소의 하드웨어로 비트 패러럴 구조와 비트 시리얼 구조에 비해 효율성을 보였다. 따라서, 제안된 구조는 암호 프로세서 칩 디자인의 기본 구조로 이용될 수 있고, 또한 단순성, 규칙성과 병렬성으로 인해 VLSI 구현에 적합하다.

참 고 문 헌

[1] W. W. Peterson and E. J. Weldon, *Error-correcting codes*, MIT Press, MA, 1972.

[2] D. E. R. Denning, *Cryptography and data securit*, Addison-Wesley, MA, 1983.

[3] A. Menezes, *Elliptic Curve Public Key Cryptosystem*, Kluwer Ademic Publiers, Boston, 1993.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Sigtures and Public-key Cryptosystems," *Comm. AC.* Vol. 21, pp. 120-126, 1978.

[5] I. S. Reed and T. K. Truong, "The use of finite fields to compute cvolutions," *IEEE Trans. Inform. Theory*, 21, pp.208-213, 1975.

[6] S. W. Wei, VLSI architectures for computing exponentiations, multiplicative inverses, and divisions inGF(2^m)," *IEEE Trans. Circuits and System*, 44, pp.847-855, 1997.

[7] S. W. Wei, "A Systolic Power-Sum Circuit for GF(2^m)," *IEEE Trans. Computer*, 43: 226-229, 1994.

[8] S. Y. Kung, *VLSI array processors*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[9] Kung, H. T., and LAM, M., "Fault tolerant and two level pipelining in VLSI systolic arrays," Proceedings of MIT conference on *Advanced res. VLSI*, Cambridge, MA, January 1984, pp.74-83.

[10] J. H. Guo, C. L. Wang, "Digit-serial systolic multiplier for finite fields GF(2^m)," *IEE Proc. Comput. Digit. Tech.*, Vol.145, No.2, March 1998.

[11] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, A System Perspective. Reaking, Mass. Addison-Wesley, 1985.

[12] Daniel D. Gajski,Principles of Digital Desig, Prentice-hall international, INC, 1997.

김 남 연

1995년 대구교육대학교 졸업(교육학학사). 1999년 계명대학교 교육대학원 전산교육 졸업(교육학석사). 2004년 경북대학교 컴퓨터공학과 졸업(공학박사). 관심분야는 Arithmetic 알고리즘, 암호학, VLSI array processors 설계

유 기 영

1976년 경북대학교 이과대학 수학교육과(이학사). 1978년 한국과학기술원 컴퓨터공학과(공학석사). 1993년 New York Rensselaer Polytechnic Institute 컴퓨터과학과(이학박사). 1978년~현재 경북대학교 공과대학 컴퓨터공학과 교수. 관심분야는 암호연산, 병렬처리, 암호화 프로토콜, 정보보호