$GF(2^m)$상의 효율적인 비트-시리얼 시스톨릭 곱셈기
(An Efficient Bit-serial Systolic Multiplier over $GF(2^m)$)

이 원 호 † 유 기 영 ††
(Won-Ho Lee) (Kee-Young Yoo)

요 약 현대 통신 분야에서 많이 응용되고 있는 유한 필드상의 중요한 연산은 곱셈과 지수승 연산 등이 있다. 유한 필드에서 지수 연산은 이진 방법을 이용하여 곱셈과 제곱을 반복 함으로서 구현될 수 있다. 그래서 이러한 연산들을 위한 빠른 알고리즘과 효율적인 하드웨어 구조 개발이 중요하다. 본 논문에서는 $GF(2^m)$상의 MSB-우선 곱셈 연산을 위한 효율적인 비트-시리얼 시스톨릭 곱셈기를 구현하였다. 제안된 곱셈기는 지수 연산기의 핵심 회로로 사용될 수 있으며 기존의 곱셈기들과 비교하여 보다 적은 입력-단자의 수와 공간-시간 복잡도를 가진다. 그리고 제안된 구조는 정규성과 모듈성, 단 방향 자료 흐름을 가지기 때문에 VLSI 칩과 같은 하드웨어로 보다 쉽게 구현할 수 있다.

키워드 : 암호학, 유한 필드, 곱셈, 지수 연산

Abstract The important arithmetic operations over finite fields include multiplication and exponentiation. An exponentiation operation can be implemented using a series of squaring and multiplication operations over $GF(2^m)$ using the binary method. Hence, it is important to develop a fast algorithm and efficient hardware for multiplication. This paper presents an efficient bit-serial systolic array for MSB-first multiplication in $GF(2^m)$ based on the polynomial representation. As compared to the related multipliers, the proposed systolic multiplier gains advantages in terms of input-pin and area-time complexity. Furthermore, it has regularity, modularity, and unidirectional data flow, and thus is well suited to VLSI implementation.

Key words : Cryptography, Finite Fields, Multiplication, Exponentiation

1. 서 론

현대 통신 분야에서 유한 필드(finite fields) $GF(2^m)$는 에러 교정 코드(error correcting code)나 암호학(cryptography), 디지털 신호 처리(digital signal processing) 같은 곳에 많이 활용되고 있다. 특히, $GF(2^m)$상의 타원 곡선 암호시스템(elliptic curve cryptosystem: ECC)은 기존의 공개키 암호시스템인 RSA나 DSA(digital signature algorithm)의 키 사이즈(m = 1204)보다 작은 키 사이즈(m = 160)로 동등한 보안(security)을 제공하기 때문에 최근 많은 연구가 되고 있다. 이러한 유한 필드에서 중요한 연산은 덧셈(addition), 곱셈(multiplication), 지수승(exponentiation) 연산 등이 있다. 이들 연산들 중에서 덧셈은 필드 원소들이 다항식 표현(polynomial representation) 방식으로 표현된다면 배타적 논리합(eXclusive OR: XOR)으로 매우 간단한 연산이다. 그러나 다른 연산들은 매우 복잡한 연산들이다. 지수 연산은 유한 필드 $GF(2^m)$상에서 이진 방법을 사용하여 제곱과 곱셈을 반복 함으로서 구현할 수 있다. 그래서 곱셈 연산을 위한 효율적이고 빠른 알고리즘과 하드웨어 개발이 매우 중요하므로 본 논문에서는 유한 필드 $GF(2^m)$상의 곱셈 연산을 위해 적은 공간-시간 복잡도를 가지는 곱셈기를 설계한다.

유한 필드 $GF(2^m)$상의 효율적인 연산기 구현을 위해 다양한 구조(architecture)가 많이 연구 되었다[1-9]. 그러한 구조들은 서로 다른 기저를 사용하였는데 그 기저에는 정규 기저(normal basis)와 이중 기저(dual basis), 표준 기저(standard basis)가 있으며 각기 다른 특징들을 가진다. 이중에서 표준 기저의 구조는 적은 복잡도와 모듈성을 가지므로 본 논문에서는 표준 기저를 사용한다[4]. 그리고 이러한 곱셈기들은 승수의 계산 순서에 따라 LSB(least significant bit)-우선 곱셈(LSB-

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 · 지원사업의 연구결과로 수행되었음
† 정 회 원 : 위덕대학교 컴퓨터멀티미디어공학부 교수
whlee@uu.ac.kr
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
yook@knu.ac.kr
논문접수 : 2004년 10월 28일
심사완료 : 2005년 9월 28일

first multiplication)과 MSB(most significant bit)-우선 곱셈(MSB-first multiplication)으로 나누어진다[6]. 본 논문에서는 유한 필드 GF(2^m)상의 MSB-우선 곱셈 연산을 위해 표준 기저를 사용하여 보다 효율적인 비트-시리얼 곱셈기를 설계한다. 제안된 비트-시리얼 곱셈기는 이전의 비트-시리얼 곱셈기와 비교했을 때 공간-시간 복잡도가 적으며, 정규성과 모듈성, 단 방향 자료 흐름을 가지기 때문에 VLSI 칩과 같은 하드웨어로 보다 쉽게 구현할 수 있다.

서론에 이어 본 논문의 구성은 다음과 같다. 2장에서는 유한 필드 GF(2^m)상의 곱셈 연산을 분석하여 변경된 MSB-우선 곱셈 알고리즘을 제안한다. 3장에서는 변경된 알고리즘을 바탕으로 효율적인 비트-시리얼 시스톨릭 곱셈기의 설계하고, 제안된 곱셈기의 검증과 기존의 곱셈기들과 비교분석을 4장에서 한다. 마지막으로 5장에서 결론을 맺는다.

2. GF(2^m)상의 MSB-우선 곱셈

유한 필드 GF(2^m)상의 원소는 여러 가지 방식으로 표기될 수 있다. 그 표기법들 중에서 다항식 표기법은 일반적으로 가장 많이 쓰이고 하드웨어와 소프트웨어 구현에 적당하다. 그래서 본 논문에서는 이 다항식 표기법을 사용하기로 한다.

$A(x)$와 $B(x)$를 유한 필드 GF(2^m)상의 원소이고 기약(irreducible) 다항식 $G(x)$를 가진다고 하자. 그러면 $A(x)$와 $B(x)$, $G(x)$는 아래의 식과 같이 표현될 수 있다.

$$A(x)=\sum_{i=0}^{m-1}a_i x^i = a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x+a_0 \quad (1)$$

$$B(x)=\sum_{i=0}^{m-1}b_i x^i = b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\cdots+b_1x+b_0 \quad (2)$$

$$G(x)=x^m+\sum_{i=0}^{m-1}g_i x^i = x^m+g_{m-1}x^{m-1}+\cdots+g_1x+g_0 \quad (3)$$

식 (1)과 식 (2), 식 (3)의 계수 a_i와 b_i, g_i는 GF(2)의 원소로 0 또는 1이다. 그리고 유한 필드 GF(2^m)상의 원소는 길이가 m인 비트-스트링(bit-string)으로 표현이 가능하다. 예로 $A(x)$는 비트-스트링 A = ($a_{m-1}a_{m-2}$ ... a_1a_0)로 표현될 수 있다.

두 원소 $A(x)$와 $B(x)$의 곱을 $P(x)$라고 정의하면 식 (4)와 같이 나타낼 수 있다.

$$P(x)=A(x)B(x)\bmod G(x)=p_{m-1}x^{m-1}+p_{m-2}x^{m-2}+\cdots+p_1x+p_0 \quad (4)$$

그러면 식 (4)는 아래의 식 (5)와 같이 유도할 수 있다.

$$\begin{aligned}
P(x) &= A(x)B(x)\bmod G(x)\\
&= \sum_{i=0}^{m-1}A(x)b_i x^i \bmod G(x)\\
&= \left[\sum_{i=1}^{m-1}A(x)b_i x^i\right]+A(x)b_0 \pmod{G(x)}\\
&= \left[\sum_{i=1}^{m-1}A(x)b_i x^{i-1}\right]x+A(x)b_0 \pmod{G(x)}\\
&= \left[\left[\sum_{i=2}^{m-1}A(x)b_i x^{i-1}\right]+A(x)b_1\right]x+A(x)b_0 \pmod{G(x)}\\
&= \left[\left[\sum_{i=2}^{m-1}A(x)b_i x^{i-2}\right]x+A(x)b_1\right]x+A(x)b_0 \pmod{G(x)}\\
&\ \ \vdots\\
&= [\cdots[[A(x)b_{m-1}]x+A(x)b_{m-2}]x+\cdots+A(x)b_1]x+A(x)b_0 \pmod{G(x)}
\end{aligned} \quad (5)$$

식 (5)에서 Wang 등 [4]은 아래의 곱셈 알고리즘을 유도하였다.

[알고리즘 1] Wang의 곱셈 알고리즘
입력: $A(x)$, $B(x)$, $G(x)$
출력: $P(x)$ = $T_m(x)$ = $A(x)B(x)$ mod $G(x)$
1. $T_0(x)$ = 0;
2. **for** i = 1 **to** m **do**
3. $\quad T_i(x)$ = $T_{i-1}(x)x$ mod $G(x)$ + $A(x)b_{m-i}$;
4. **return** $T_m(x)$;

알고리즘 1에서 보는 것과 같이 Wang은 모듈러 리덕션(modular reduction) 부분과 모듈러 곱셈(modular multiplication) 부분을 같이 수행한다. 그래서 이 알고리즘은 일정한 규칙을 가지고 반복하는 규칙성 즉 정규성을 많이 가지는 반면에 첫 번째 항목(i = 1)에서 필요 없는 모듈러 리덕션($T_0(x)x$ mod $G(x)$)을 수행하고 있다. 이는 하드웨어 구현에서 공간-시간 복잡도를 증가시킨다. 그래서 본 논문에서는 이러한 단점을 없애기 위해 다음과 같이 변경된 곱셈 알고리즘을 제안한다.

[알고리즘 2] 변경된 MSB-우선 곱셈 알고리즘
입력: $A(x)$, $B(x)$, $G(x)$
출력: $P(x)$ = $T_m(x)$ = $A(x)B(x)$ mod $G(x)$
1. $T_1(x)$ = $A(x)b_{m-1}$;
2. **for** i = 2 **to** m **do**
3. $\quad T_i(x)$ = $T_{i-1}(x)x$ mod $G(x)$ + $A(x)b_{m-i}$;
4. **return** $T_m(x)$;

변경된 곱셈 알고리즘은 기존 Wang의 알고리즘의 첫 번째 항목(i = 1)에서 필요 없는 모듈러 리덕션($T_0(x)x$ mod $G(x)$)을 소거하였다. 그래서 곱셈하는데 필요한 시

간 즉 곱셈 지연시간을 줄일 수 있다. 또한 초기화 ($T_0(x)$ = 0) 부분이 없어져 입력-단자 수를 줄일 수 있고, 필요 없는 연산에 사용되는 게이트들이 제거 됨으로써 하드웨어 복잡도도 줄일 수 있다. 따라서 기존의 곱셈기보다 공간-시간 복잡도를 줄일 수 있다.

3. 시스톨릭 곱셈기 설계

이 장에서는 유한 필드 GF(2^m)상의 곱셈을 위한 시스톨릭 어레이(systolic array)를 설계하기 위해 참고문헌 [10,11]에 있는 절차를 따른다. 먼저 주어진 알고리즘으로부터 비트-레벨 형태의 정규 순환 알고리즘으로 표현한 후 이를 자료 의존 그래프로 나타낸다. 자료 의존 그래프에서 변수들의 초기 위치를 추출한 후에 계산 공간의 프로세서 공간을 명시하는 공간 변환 행렬(S)과 연산 순서를 결정하는 시간벡터(Π)를 구한다. 그리고 시간-공간 변환 행렬(T)을 구한 후 이 행렬을 적용하여 비트-시리얼 시스톨릭 어레이를 설계한다.

3.1 자료 의존 그래프

시스톨릭 어레이 설계를 위한 비트-레벨 순환 알고리즘을 유도하기 위해 먼저 식 (5)에서 다음 식 (6)과 같이 순환식을 얻는다.

$$T_i(x) = T_{i-1}(x)x \bmod G(x) + A(x)b_{m-i}, \quad (2 \le i \le m) \tag{6}$$

여기서 $T_1(x) = A(x)b_{m-1}$이고 $P(x) = T_m(x)$이다. 그리고 $T_i(x)$는 식 (7)과 같이 나타낼 수 있다.

$$T_i(x) = t_{i,m-1}x^{m-1} + t_{i,m-2}x^{m-2} + \cdots + t_{i,1}x + t_{i,0} \tag{7}$$

식 (6)에서 $T_{i-1}(x)$값 대신에 식 (7)을 변경대입하면 식 (8)과 같이 나타낼 수 있다.

$$T_i(x) = t_{i-1,m-1}x^m \bmod G(x) + t_{i-1,m-2}x^{m-1} + \cdots + t_{i-1,1}x^2 + t_{i-1,0}x + A(x)b_{m-i}, \quad (2 \le i \le m). \tag{8}$$

식 (8)에서 $x^m \bmod G(x)$는 식 (3)을 이용하여 식 (9)와 같이 쉽게 유도할 수 있다.

$$x^m \bmod G(x) = g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1x + g_0. \tag{9}$$

만약 $x^m \bmod G(x)$를 $g(x)$로 둔다면 식 (8)은 다음 식 (10)과 같이 나타낼 수 있다.

$$T_i(x) = t_{i-1,m-1}g(x) + t_{i-1,m-2}x^{m-1} + \cdots + t_{i-1,1}x^2 + t_{i-1,0}x + A(x)b_{m-i}, \quad (2 \le i \le m). \tag{10}$$

식 (1)과 식 (9), 식 (10)을 이용하여 알고리즘 2를 다음의 알고리즘 3과 같이 비트-레벨 순환 알고리즘으로 나타낼 수 있다.

[알고리즘 3] 변경된 비트-레벨 MSB-우선 곱셈 알고리즘

입력: $A(x)$, $B(x)$, $G(x)$
출력: $P(x) = T_m(x) = A(x)B(x) \bmod G(x)$

```
1.  for i = 1 to m do
2.      for j = 1 to m do
3.          if i = 1 then
4.              t_{1,m-j} = a_{m-j}·b_{m-1};
5.          else
6.              if j = m then
7.                  t_{i,0} = (a_0·b_{m-i}) ⊕ (t_{i-1,m-1}·g_0);
8.              else
9.                  t_{i,m-j} = t_{i-1,m-j-1} ⊕ (a_{m-j}·b_{m-i})
                    ⊕ (t_{i-1,m-1}·g_{m-j});
10. return T_m(x);
```

유한 필드 GF(2^m)상의 MSB-우선 곱셈을 위한 자료 의존 그래프는 알고리즘 3의 비트-레벨 순환 알고리즘을 이용하여 그림 1과 같이 나타낼 수 있다. 그림 1에서 보는 것과 같이 $m \times m$ 개의 기본 노드(node)로 구성되어 있다. 그림 1의 자료 의존 그래프에서 노드는 계산이 일어나는 곳을 의미하고 에지(edge)는 자료의 흐름을 나타낸다. 그림 1에서 보면 총 3가지 타입의 노드 즉 처리 요소(processing element: PE)가 존재하는데 m개의 타입-1 PE와 $(m-1)^2$개의 타입-2 PE와 $(m-1)$개의 타입-3 PE가 존재한다. 각 $PE_{i,j}$의 구조는 그림 2, 3, 4와 같다. 그림 2에서 보는 것과 같이 타입-1의 PE는 1개의 2-입력 AND 게이트(gate)로 구성되어 있고, 그림 3

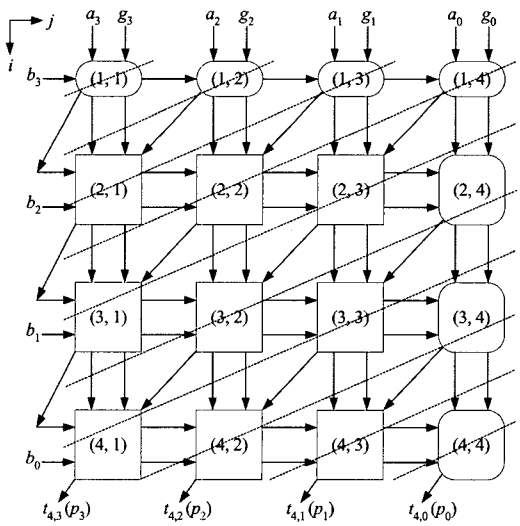

그림 1 유한 필드 GF(2^m)상의 자료의존 그래프(m = 4)

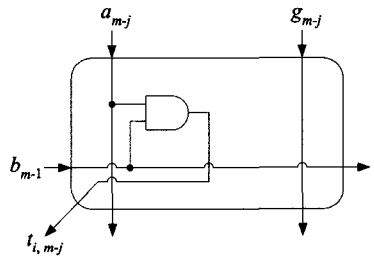

그림 2 타입-1의 PE 구조

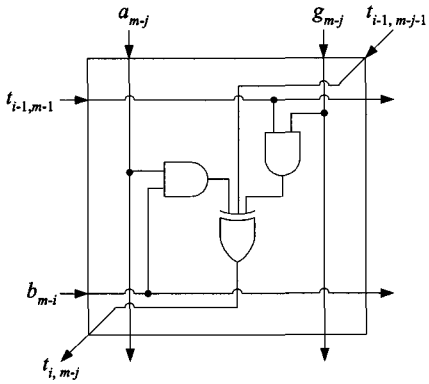

그림 3 타입-2의 PE 구조

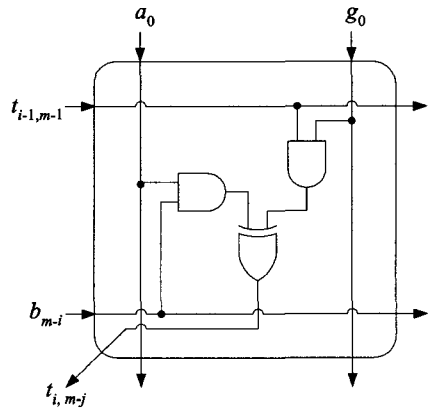

그림 4 타입-3의 PE 구조

에서 보는 것과 같이 타입-2의 PE는 1개의 2-입력 AND 게이트와 3-입력 XOR 게이트로 구성되어 있다. 그리고 그림 4에서 보는 것과 같이 타입-3의 PE는 1개의 2-입력 AND 게이트와 2-입력 XOR 게이트로 구성된다.

값 b_i는 $[i,\ 0]^T$에서 공급되고 $[0,\ 1]^T$ 방향으로 흐른다. 여기서 $[\pm i,\ \pm j]^T$는 $[\pm i,\ \pm j]$의 전치행렬로 행렬의 각 값은 방향을 나타낸다. $+i$는 i축의 아래쪽 방향을 의미하고 $-i$는 위쪽 방향을 나타낸다. $+j$는 오른쪽 방향을 나타내고 $-j$는 왼쪽 방향을 나타낸다. 값 a_j와 g_j는 $[0,\ j]^T$에서 공급되고 $[1,\ 0]^T$ 방향으로 흐른다. 각 $t_{i,j}$ 값은 모든 노드에서 계산되어지고 $[1,\ -1]^T$ 방향으로 흐른다. 그림 1에서 방향 벡터 $[1,\ -2]^T$의 모든 노드들 즉 같은 점선상에 있는 노드들은 동시에 수행된다. 결과 값 $t_{m,j}(=p_j)$는 $[m,\ j]^T$에서 나타난다.

3.2 비트-시리얼 시스톨릭 곱셈기

그림 1에서 주어진 자료 의존 그래프에서 비트-시리얼 시스톨릭 어레이를 설계하기 위해 먼저 자료 의존 행렬(D)을 구하면 식 (11)과 같다.

$$D=\left[\overline{d_{b_i}},\overline{d_{t_{i,m-1}}},\overline{d_{t_{i,j}}},\overline{d_{a_j}},\overline{d_{g_j}}\right]=\begin{bmatrix}0 & 0 & 1 & 1 & 1\\ 1 & 1 & -1 & 0 & 0\end{bmatrix}. \quad (11)$$

선형 시스톨릭 어레이를 설계하기 위해서는 투영벡터(projection vector)가 필요한데 이 벡터에는 크게 4가지가 있다. 즉 $[0,\ 1]^T$과 $[1,\ 0]^T$, $[1,\ -1]^T$, $[1,\ 1]^T$의 투영벡터 들이 있다. 이 벡터들 중에서 $[0,\ 1]^T$의 투영벡터가 가장 좋은 성능을 가진다[8]. 그래서 본 논문에서는 이 투영벡터를 사용한다. 이 경우 공간변환 행렬(S)은 $[1,\ 0]$이고, 시간벡터(Π)는 $[2,\ 1]$이다[8]. 그러므로 공간 시간 변환 행렬은 식 (12)와 같이 나타낼 수 있다.

$$T=\begin{bmatrix}\Pi\\ S\end{bmatrix}=\begin{bmatrix}2 & 1\\ 1 & 0\end{bmatrix}. \quad (12)$$

자료 흐름 링크들에서 지연시간과 흐름의 방향은 식 (11)과 식 (12)를 이용하여 아래의 식 (13)과 같이 구할 수 있다.

$$TD=\begin{bmatrix}2 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}0 & 0 & 1 & 1 & 1\\ 1 & 1 & -1 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 2 & 2\\ 0 & 0 & 1 & 1 & 1\end{bmatrix}. \quad (13)$$

식 (13)에서 첫 번째 행은 자료 흐름 링크들의 지연시간을 나타내고 두 번째 행은 자료 흐림 링크들의 흐름 방향을 나타낸다. 그러므로 자료 의존 그래프를 $[0,\ 1]^T$의 투영벡터로 투영시켜 새로운 비트-시리얼 시스톨릭 어레이를 쉽게 유도할 수 있다. 그림 5는 그림 1에서 유도된 비트-시리얼 시스톨릭 곱셈기를 나타낸다. 유도된 곱셈기는 한 개의 PE_1과 $(m-1)$개의 PE_2로 구성되고, PE_1의 구조는 그림 6에 PE_2의 구조는 그림 7에 각각 나타나 있다.

그림 1에서 보는 것과 같이 각 행의 b_i 값과 $t_{i,m-1}$ 값이 옆의 노드들에게 전달이 되어야 하기 때문에 2-to-1 멀티플렉서(multiplexer: MUX)들과 1-비트-래치(latch)들을 그림 6, 7과 같이 추가하였다. 그리고 추가한 회로는 제어신호(ctl)에 의해 제어된다. 제어신호의 순서는 하나의 1과 $(m-1)$개의 0으로 구성되어 있다. 만약 제어신호가 1이면 b_i 값과 $t_{i,m-1}$ 값은 각 PE의 계산을 위해 PE의 래치에 저장이 되고, 0이면 이웃 PE로 값이 전송되어 진다. 제안된 곱셈기는 단 방향 자료 흐름을 가지

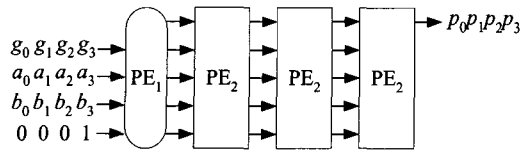

그림 5 유한 필드 GF(2^m)상의 비트-시리얼 시스톨릭 곱셈기(m = 4)

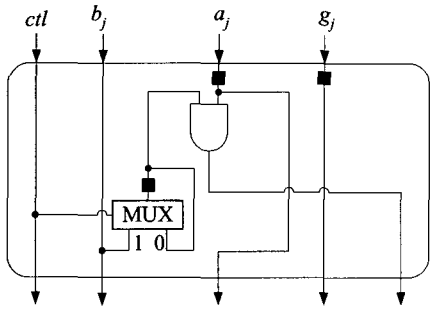

그림 6 제안된 곱셈기의 PE_1 구조

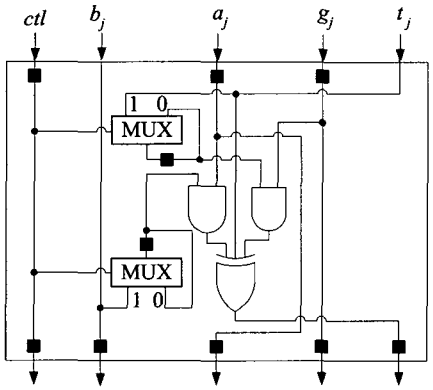

그림 7 제안된 곱셈기의 PE_2 구조

고, (3m-1)의 클럭 후에 결과 값을 얻을 수 있다. 클럭 주기를 결정하는 임계경로(critical-path)는 하나의 2-입력 AND 게이트와 하나의 3-입력 XOR 게이트, 그리고 하나의 1-비트-래치를 통과하는 신호로 구성된다.

4. 검증 및 비교분석

제안된 비트-시리얼 곱셈기를 검증하기 위하여 ALTERA의 MAX+PLUS(multiple array matrix programmable logic user system) II 툴을 사용해서 회로를 설계하고, 클럭 주기(clock period)를 20ns로 하여 시뮬레이션을 하였다. 시뮬레이션의 입력 값은 GF(2^4)상의 원소인 $A(x) = x^3 + x^2 + 1$ (1101)과 $B(x) = x^2 + 1$ (0101)을 사용하였고, 기약 다항식은 $G(x) = x^4 + x + 1$을 사용하였다. 그래서 $g(x) = x + 1$ (0011)이다. 그림 8은 제안된 비트-시리얼 곱셈기를 시뮬레이션 한 결과를 보여준다. 그림 8에서 clk는 클럭을 나타내고, A_in은 피연산자 $A(x)$를 나타내고, B_in은 피연산자 $B(x)$를 나타내고, G_in은 $g(x)$의 값을 나타내고, 마지막으로 T_out는 곱셈의 결과 값을 나타낸다. 그림 8에서 보는 것과 같이 곱셈의 결과 값인 $P(x) = A(x)B(x) \bmod G(x) = x^3 + x^2$ (1100)이 11번째 클럭(3m-1) 즉 220ns에서 모두 나온다.

유한 필드 GF(2^m)상의 효율적인 연산기 구현은 참고문헌 [1-9]에서 보는 것과 같이 많이 연구 되어왔다. Jain 등 [6]이 제안한 곱셈기는 수행속도 면에서 아주 효율적이지만 브로드케스팅 (broadcasting) 데이타 입력을 가지고 있어서 시스톨릭 구조와 좀 다르다. 그리고, 고속 시스템 구현할 때는 브로트케스팅을 피해서 구현을 한다. 참고문헌 [7]과 [8], [9]에서 제안한 GF(2^m)상의 연산기는 시스톨릭 구조로 설계되었지만 AB곱셈을 위한 구조가 아닌 AB^2연산을 위한 구조로 이러한 구조는 나눗셈연산에 자주 사용된다. Yeh 등[1]이 제안한 시스톨릭 곱셈기는 2개의 제어 신호가 필요하지만 제안한 GF(2^m)상의 곱셈기는 제어 신호가 한 개만 있으면 충분하다. 그리고, Scott 등[2]과 Bandyopadhyay 등[3]이 제안한 GF(2^m)상의 곱셈기는 데이타 흐름이 한 방

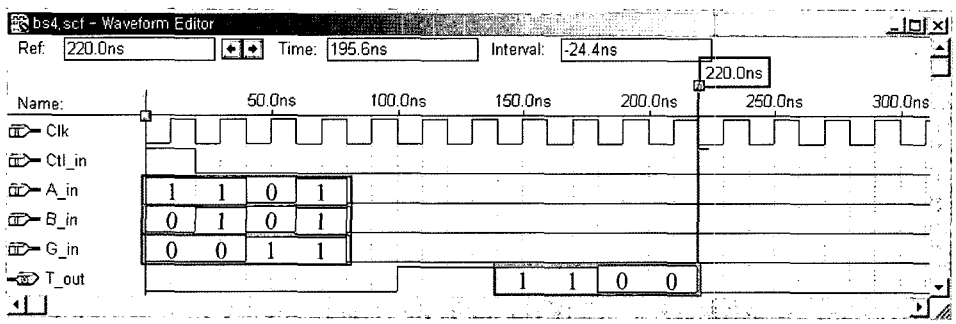

그림 8 비트-시리얼 곱셈기의 시뮬레이션 결과(m = 4)

향으로 흐르지 않지만, 제안한 시스톨릭 곱셈기는 한 방향으로 흘러서 고속 시스템 구현에 적합하다. Wang 등[4]과 Mekhallalati 등[5]의 GF(2^m)상의 비트-시리얼 시스톨릭 곱셈기와 제안된 비트-시리얼 곱셈기의 비교는 표 1과 같다. 모든 곱셈기들은 MSB-우선 곱셈 알고리즘으로 구현이 되었으며, 제어신호 수와 처리율(throughput)이 같은 것을 볼 수 있다. 그러나 제안된 곱셈기가 입력-단자의 수가 적음을 볼 수 있다. 이는 기존의 곱셈 알고리즘에서 초기화($T_0(x)$ = 0) 연산을 제거함으로써 가능하게 되었다. 또한 필요 없는 모듈러 리덕션 연산($T_0(x)x$ mod $G(x)$)을 소거를 함으로써 곱셈하는데 필요한 시간 즉 곱셈 지연시간을 기존 $3m$에서 $3m$-1로 단축한 것을 볼 수 있다. 또한 필요 없는 연산에 사용되는 게이트들이 제거됨으로써 하드웨어복잡도도 줄은 것을 확인할 수 있다.

제안된 곱셈기와 기존의 곱셈들의 공간-시간 복잡도를 비교하기 위해 참고문헌[12]에 있는 다음과 같은 값들을 사용한다. 1) 3-입력 XOR 게이트는 2개의 2-입력 XOR 게이트로 구성된다. 2) 각 게이트에 필요한 트랜지스터(transistor) 수는 A_{AND2} = 6, A_{XOR2} = 14, A_{MUX2} = 20, A_L = 8 이다. 3) 각 게이트의 지연시간(ns)은 T_{AND2} = 2.4, T_{XOR2} = 4.2, T_{MUX2} = 5.8, T_L = 1.4.이다.

그림 9는 위의 값들을 사용하여 표 1에 있는 곱셈기들의 공간-시간 복잡도를 m의 값에 따라 도식화한 것이다. 곱셈기들의 공간-시간 복잡도는 공간과 시간 복잡도의 곱으로 계산되었는데, 공간 복잡도는 각 게이트들의 트랜지스터(transistor) 수들의 합이고 시간 복잡도는 임계경로의 시간과 지연시간의 곱이다. 제안된 곱셈기가 기존의 곱셈기 보다 가장 적은 공간-시간 복잡도를 가지는 것을 그림 9에서 확인할 수 있다. 예로 m = 160일 경우, Wnag 등[4]과 Mekhallalati 등[5]의 곱셈기보다 각각 4.29%와 13.85%가 향상되었다.

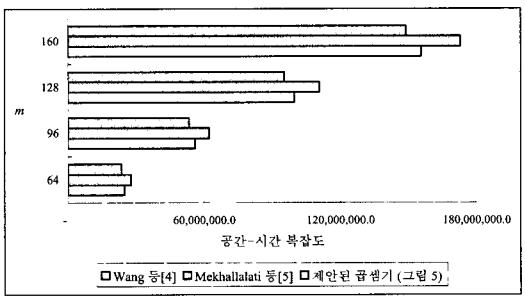

그림 9 비트-시리얼 곱셈기들의 공간-시간 복잡도

5. 결 론

본 논문에서는 효율적인 GF(2^m)상에서 표준 기저를 사용하여 비트-시리얼 시스톨릭 곱셈기를 제안하였다. 기존의 곱셈 알고리즘을 분석하여 공간-시간 복잡도를 줄일 수 있는 변경된 알고리즘을 유도하였다. 그리고 변경된 알고리즘으로부터 비트-레벨 순환 방정식 알고리즘을 유도하여 자료 의존 그래프를 얻은 다음 효율적인 비트-시리얼 시스톨릭 곱셈기를 설계하였다.

제안된 곱셈기는 기존의 비트-시리얼 곱셈기들 보다 최소 4.29% 향상된 공간-시간 복잡도를 가졌으며, 지수 연산기의 핵심 회로로 사용될 경우 곱셈을 반복적으로 하기 때문에 더 많은 성능 향상을 가져올 수 있다. 그리고 제안된 곱셈기는 모듈성과 정규성, 단 방향 자료 흐름을 가지고 있어서 VLSI 구현에 적합하고, 더 나아가 공개키 암호시스템의 연산기로 사용될 수 있다.

참 고 문 헌

[1] C. S. Yeh, I. S. Reed, T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," *IEEE Trans. Computer*, Vol.C-33, pp.357-360, 1984.

[2] P. A. Scott, S. E. Tavares, L. E. Peppard, "A fast VLSI multiplier for GF(2^m)," *IEEE J. Selected Areas in Communication*, Vol.4, pp.62-66, 1986.

[3] S. Bandyopadhyay, A. Sengupta, "Algorithms for multiplication in Galois field for implementing

표 1 유한 필드 GF(2^m)상의 비트-시리얼 시스톨릭 곱셈기들의 비교

		Wang 등[4]	Mekhallalati 등[5]	제안된 곱셈기(그림 5)
입출력 형태		비트-시리얼	비트-시리얼	비트-시리얼
알고리즘		MSB-우선	MSB-우선	MSB-우선
입력-단자 수		5	5	4
제어신호 수		1	1	1
하드웨어 복잡도	2-입력 AND	$3m$	$3m$	$2m$-1
	2-입력 AND	m	m	m-1
	2-to-1 MUX	$2m$	$2m$	$2m$-1
	1-비트 Latch	$10m$+2	$8m$+2	$10m$-7
처리율		$1/m$	$1/m$	$1/m$
곱셈 지연시간		$3m$	$3m$	$3m$-1
임계경로		T_{AND2} +T_{XOR3} + T_L	T_{AND2} + T_{XOR3} + T_{MUX2} + T_L	T_{AND2} + T_{XOR3} + T_L

using systolic arrays," *Proc. Inst. Elect. Eng.*, Vol.35, pp.336-339, 1988.
[4] C. L. Wang, J. L. Lin, "Systolic Array Implementation of Multipliers for finite fields GF(2^m)," *IEEE Trans. Circuits Systems*, Vol.38, pp.796-800, 1991.
[5] M. C. Mekhallalati and A. S. Ashur, "Novel Structures for Serial Multiplication over the Finite Field GF(2^m)," *Workshop on VLSI Signal Processing*, Vol.IX, pp.65-74, 1996.
[6] S. K. Jain, L. Song, K. K. Parhi, "Efficient Semisystolic Architectures for Finite-Field Arithmetic," *IEEE Trans. on VLSI Systems*, Vol.6. pp.101-113, 1998.
[7] C. L. Wang and J. H. Guo, "New Systolic Array for $C + AB^2$, Inversion and Division in GF(2^m)," *IEEE Trans. Computer*, vol.49, pp.1120-1125, 2000.
[8] N. Y. Kim, H. S. Kim, and K. Y. Yoo, "Computation of AB^2 multiplication in GF(2^m) using low-complexity systolic architecture," *IEE Proc. Circuits, Devices & Systems*, vol.150, No.2, pp.119-123, 2003.
[9] 이원호, 유기영 "공간 효율적인 비트-시리얼 제곱/곱셈기 및 AB^2-곱셈기", 정보과학회 논문지: 시스템 및 이론, 제31권, 제1 · 2호, pp.1-9, 2004.
[10] S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[11] K. Y. Yoo "A Systolic Array Design Methodology for Sequential Loop Algorithms," Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, 1992.
[12] D. D. Gajski, *Principles of Digital Design*, Prentice Hall, Upper Saddle River, NJ, 1997.

이 원 호
1998년 대구대학교 전자계산학과 공학사 2000년 경북대학교 컴퓨터공학과 공학석사. 2004년 경북대학교 컴퓨터공학과 공학박사. 2005년~현재 위덕대학교 컴퓨터멀티미디어공학부 전임강사. 관심분야는 정보보호, 암호학, 어레이 프로세서 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