SBM과 KOA 구현 방식에 따른 GF(2^m) 곱셈기 하드웨어 성능 분석

이성은, 최소연, 신예린, *유호영
충남대학교 전자공학과
e-mail : selee.*cas@gmail.com, sychoi.cas@gmail.com, yrshin.cas@gmail.com, hyyoo@cnu.ac.kr*

Comparison on GF(2^m) multipliers based on SBM and KOA

Sungeun Lee, Soyeon Choi, Yerin Shin, and *Hoyoung Yoo
Department of Electronics Engineering
Chungnam National University

Abstract

Polynomial multiplication on Galois Field (GF) is widely used for various error correction coding and encryption algorithms. In this paper, we introduce and compare two GF(2^m) structures based on conventional School Book Multiplication (SBM) and Karatsuba-Ofman Algorithm (KOA). SBM-based GF multiplier provides simple hardware implementation and KOA-based multiplier improves the performance of SBM-based ones. According to experimental results, SBM is desirable when m is small and KOA is desirable when m is large.

I. 서론

Galois field GF(p)는 p개의 원소로 구성된 유한체(Finite Field)이고, GF(p^m)은 p^m개로 구성되는 GF(p)의 확장체(Extension Field)이다 [1]. 이때 p와 m은 각각 소수(prime number)와 양의 정수를 나타낸다. 예를 들어 p가 2인 경우, Galois field는 GF(2)로 표현되고, 확장체는 GF(2^m)으로 표현 가능하다. 특히 GF(2^m)은 0 또는 1로 이루어진 m 비트의 원소 2^m개로 구성된다. GF(p)와 GF(p^m) 모두 유한체 특성에 의하여 각각 GF덧셈과 곱셈이 정의가 되어 있고 또한 동시에 닫혀 있다. 이러한 GF 산술 연산은 암호화 알고리즘, 오류정정 부호 등 다양한 분야에 널리 사용된다 [1]. GF덧셈은 XOR 연산으로 간단하게 구현되지만, GF곱셈은 비교적 복잡한 연산 과정을 거친다. 따라서 본 논문에서는 GF 곱셈을 위한 가장 교과서적인 방식인 SBM (School Book Multiplication)[2]과 SBM의 성능을 개선하기 위해 제안된 KOA (Karatsuba-Ofman Algorithm)[3]을 이용하여 GF 곱셈기를 구현하고, 면적과 연산 속도 측면에서 두 곱셈기의 성능을 분석한다. 이를 통해 다양한 제약조건에 따른 효율적인 구조를 제안하고자 한다.

II. 본론

일반적으로 GF(2^m) 상의 원소는 m 비트로 구성되기 때문에 식 (1)과 같이 (m-1)차 다항식으로 표현할 수 있다.

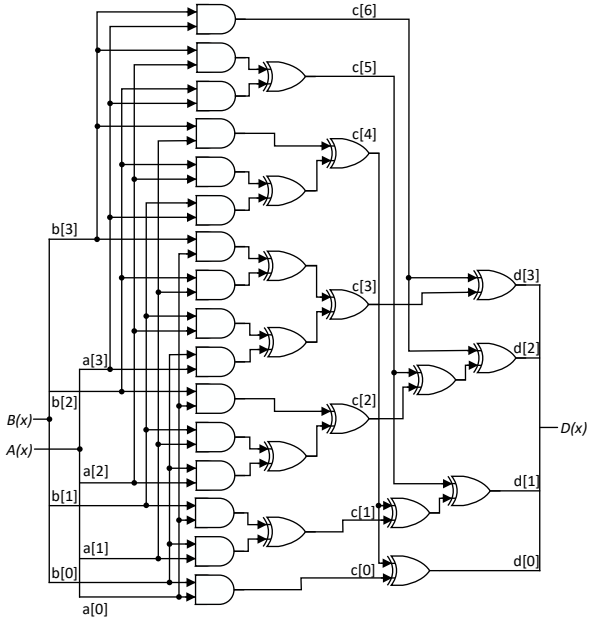


그림 1 GF(2^4)의 Galois field 곱셈을 위한 SBM 하드웨어 구조

$$A(x)=a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x^1+a_0x^0$$
$$B(x)=b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\cdots+b_1x^1+b_0x^0 \quad ,(1)$$

이때 a_i $(0\le i<m)$와 b_j $(0\le j<m)$이 0 또는 1인 값을 가진다. GF(2^m)에서의 두 원소의 곱셈은 $A(x)$ 와 $B(x)$ 의 곱셈으로 표현 가능 하며, 이 때 곱셈 과정은 다항식 곱셈 (Polynomial Multiplication)과 모듈로 축소 (Modulo Reduction) 과정을 통해 이루어진다. 다항식 곱셈 과정은 식 (2)와 같다.

$$\begin{aligned}D(x)&=A(x)\times B(x)\\&=(a_{m-1}b_{m-1})x^{2m-2}+(a_{m-1}b_{m-2}+a_{m-2}b_{m-1})x^{2m-1}\\&\quad+\cdots+(a_1b_0+a_0b_1)x^1+(a_0b_0)x^0\\&=\sum_{i=0}^{2m-2}x^i\cdot\left(\sum_{s+t=i;s,t\ge0}a_sb_t\right)=\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}a_ib_jx^{i+j}\end{aligned} \quad ,(2)$$

식 (2)를 통해 계산된 다항식 결과 값을 GF(2^m) 상의 원소로 변환하기 위하여 식 (3)과 같은 모듈로 축소 과정을 수행하게 된다. $D(x)$ 의 차수가 m보다 작은 경우, $D(x)$ 가 최종 곱셈 결과 $C(x)$ 로 할당되며, $D(x)$ 의 차수가 m보다 클 경우 모듈로 축소 과정을 수행한다. 모듈로 축소를 위해 m차의 원시다항식(Primitive Polynomial) $P(x)$를 적용하며 결과적으로 m 차 이상의 $D(x)$를 GF(2^m)의 원소에 해당하는 m−1차 다항식 $C(x)$로 변환한다. 이는 식 (3)을 통해 나타내었다.

$$C(x)=\begin{cases}D(x) & \text{degree of } D(x)<m\\ D(x)\bmod P(x) & \text{degree of } D(x)\ge m\end{cases} \quad ,(3)$$

2.1 School Book Multiplication (SBM)

SBM은 가장 일반적으로 GF(2^m)의 다항식 곱셈을 수행하는 방식으로, 그 과정은 식 (2)와 (3)과 동일하다 [4]. 예를 들어 GF(2^4)에서의 $A(x), B(x)$ 에 대한 GF곱셈인 경우, 식 (2)의 다항식 곱셈 과정을 식 (4)으로 표현할 수 있다

$$\begin{aligned}D(x)=A(x)\times B(x)&=(a_3b_3)x^6+(a_3b_2+a_2b_3)x^5\\&\quad+(a_3b_1+a_2b_2+a_1b_3)x^4+(a_3b_0+a_2b_1+a_1b_2+a_0b_3)x^3\\&\quad+(a_2b_0+a_1b_1+a_0b_2)x^2+(a_1b_0+a_0b_1)x^1+(a_0b_0)x^0\end{aligned} \quad ,(4)$$

식 (4)의 최고차항이 m인 4보다 크기 때문에 모듈로 축소가 필요하며, 이때 사용되는 $P(x)$는 x^4+x+1 이다. 최종적인 곱셈 결과 $C(x)$ 를 나타내면 식 (5)와 같다.

$$\begin{aligned}C(x)&=(a_3b_0+a_2b_1+a_1b_2+a_0b_3+a_3b_3)x^3\\&\quad+(a_2b_0+a_1b_1+a_0b_2+a_3b_3+a_3b_2+a_2b_3)x^2\\&\quad+(a_1b_0+a_0b_1+a_3b_2+a_2b_3+a_3b_1+a_2b_2+a_1b_3)x^1\\&\quad+(a_0b_0+a_3b_1+a_2b_2+a_1b_3)x^0\end{aligned} \quad ,(5)$$

식 (4)와 (5)를 기반으로 GF(2^4)의 Galois filed 곱셈을 위한 SBM 하드웨어 구조를 나타내면 그림 1과 같다.

2.2 Karatsuba−Ofman Algorithm

KOA는 곱셈기의 SBM의 성능을 개선하기 위해 제안된 알고리즘으로, 1962년 Karatsuba와 Ofman에 의해 제안되었다 [3]. 긴 수의 곱셈을 더 빠르게 할 수 있는 방식 중 하나로, 다항식 곱셈 과정에 필요한 계수의 덧셈연산을 추가하는 대신 계수의 곱셈연산을 줄여 연산의 복잡도를 줄이는 방식이다 [4]. KOA에서는 복잡도를 줄이기 위하여 비트를 그룹화하여 그룹별 연산을 수행한다. 예를 들어 GF(2^4)에서의 다항식 $A(x)$ 와 $B(x)$ 의 계수를 [a_3 a_2 a_1 a_0], [b_3 b_2 b_1 b_0]의 이진표현으로 나타낼 수 있고, 각 표현의 MSB 2비트는 A_H, B_H로, 나머지 하위 LSB 2비트는 A_L, B_L로 표현할 수 있다. KOA 곱셈기에서는 그림 2(a)와 같이 이들을 조합하여 KOA_MUL의 입력으로 {A_H, B_H}, {(A_H+A_L), (B_H+B_L)}, {A_L, B_L}이 들어가고, 각 KOA_MUL의 출력은 D_H, D_M, D_L이 된다.

KOA_MUL은 2비트의 곱셈연산으로 보조변수 D_d

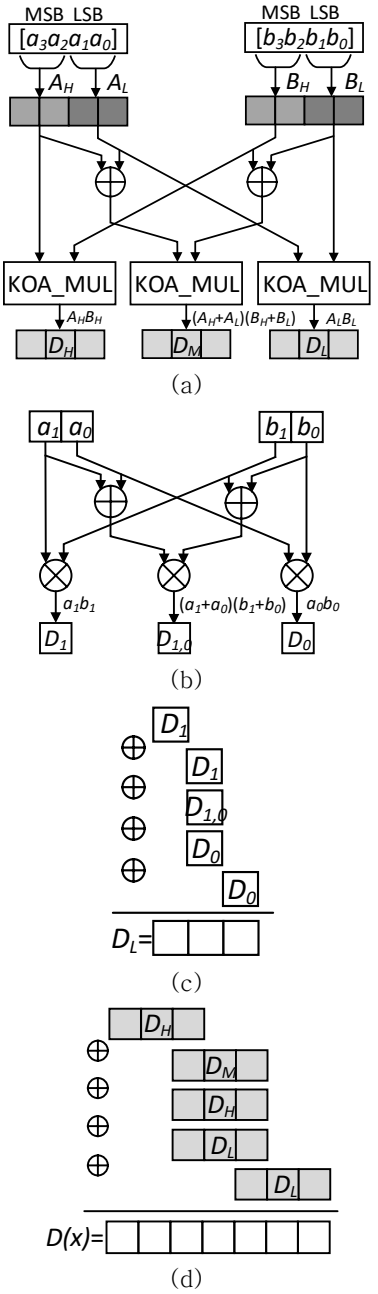


그림 2 (a) KOA의 입력 비트 분리 및 KOA_MUL의 출력 (b) KOA_MUL 보조변수 곱셈 (c)KOA_MUL 보조변수 GF덧셈 (d)KOA_MUL출력 GF덧셈

$(0 \le d < m-1)$로 표현할 수 있다. 예를 들어 A_L, B_L이 입력인 KOA_MUL은 MSB인 a_1, b_1의 곱을 보조변수 D_1, LSB인 a_0, b_0의 곱을 보조변수 D_0, 각 MSB와 LSB의 합의 곱인 $(a_1+a_0)(b_1+b_0)$는 $D_{1,0}$로 나타낼 수 있다. 이는 그림2의 (b)처럼 표현된다. KOA_MUL에서 생성된 보조변수는 1비트로, 3비트의 출력을 만들기 위해 보조변수를 더하는 과정을

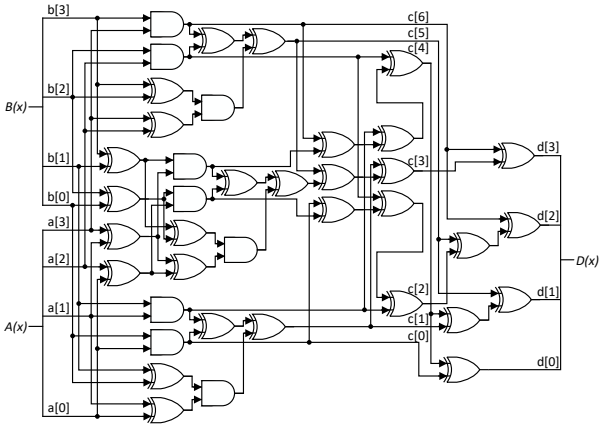


그림 3 GF(2^4)의 Galois field 곱셈을 위한 KOA 하드웨어 구조

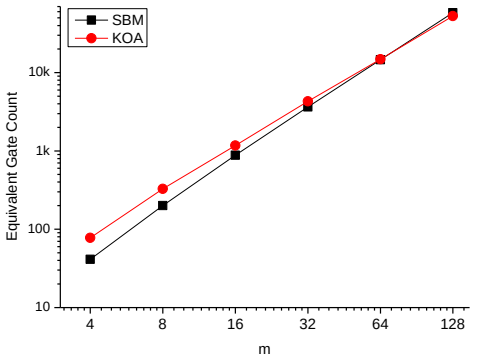
거친다. 2비트의 입력 중 MSB의 곱으로 생성된 보조변수 D_1은 2비트 시프트(shift)하여 출력의 MSB 1비트가 되고, 입력의 LSB의 곱인 D_0는 출력의 LSB 1비트가 된다. 나머지 MSB와 LSB의 합으로 얻어진 $D_{1,0}$은 D_1, D_0과 GF덧셈을 수행한 뒤 출력의 중간인 2번째 비트가 된다. 이는 그림 2의 (c)처럼 표현된다.

KOA_MUL의 3비트 출력인 D_H, D_M, D_L은 다시 GF덧셈을 거쳐 7비트 출력을 만든다. 이때 D_H는 4비트 시프트, D_M은 D_H, D_L과 GF덧셈한 뒤 2비트 시프트, D_L은 시프트 하지 않고 모두 GF덧셈하여 생성된 7비트 출력이 GF(2^4)에서의 $A(x), B(x)$ 의 곱인 $D(x)$ 가 된다. 이는 그림 2의 (d)처럼 표현된다. m의 크기가 늘어나면 입력 $A(x), B(x)$ 의 비트가 2비트가 될 때까지 MSB 절반과 나머지 LSB 절반을 나누는 과정을 반복한 뒤, 나눠진 개수만큼 KOA_MUL을 수행한다. 나누는 단계의 개수를 r이라고 하면, KOA_MUL의 개수는 3^r만큼 필요하며 m이 4인경우 r은 1이다 [5]. GF(2^4)에서 계산되는 보조변수 D_d는 식 (6)와 같이 표현되고, 다항식 곱셈과정은 식 (7)과 같다.

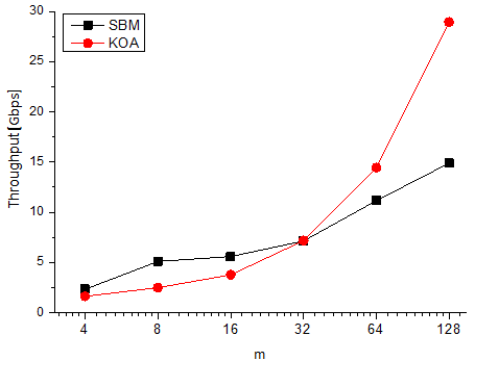
$$\begin{aligned} &D_0 = a_0b_0, D_1 = a_1b_1, D_2 = a_2b_2, D_3 = a_3b_3 \\ &D_{3,2} = (a_3+a_2)(b_3+b_2), D_{3,1} = (a_3+a_1)(b_3+b_1) \\ &D_{2,0} = (a_2+a_0)(b_2+b_0), D_{1,0} = (a_1+a_0)(b_1+b_0) \\ &D_{3,2,1,0} = (a_3+a_2+a_1+a_0)(b_3+b_2+b_1+b_0) \end{aligned} \quad ,(6)$$

$$\begin{aligned} D(x) &= A(x) \times B(x) \\ &= (D_3)x^6 + (D_{3,2}+D_3+D_2)x^5 + (D_{3,1}+D_3+D_2+D_1)x^4 \\ &\quad + (D_{3,2,1,0}+D_{3,2}+D_{3,1}+D_{2,0}+D_{1,0}+D_3+D_2+D_1+D_0)x^3 \\ &\quad + (D_{2,0}+D_2+D_0)x^2 + (D_{1,0}+D_1+D_0)x^1 + (D_0)x^0 \end{aligned} \quad ,(7)$$

모듈로 축소는 SBM과 동일한 방법으로 수행된다.



(a)



(b)

그림 4 SBM과 KOA의 m에따른 (a) equivalent gate count 와 (b) 처리량 비교

KOA를 적용한 GF(2^4)의 곱셈기를 위한 하드웨어는 그림 3와 같다.

Ⅲ. 구현

SBM과 KOA를 적용한 GF(2^m) 곱셈기의 m에 따른 하드웨어 면적과 처리량(throughput)을 그림 4의 그래프로 표현하였다. 곱셈기의 동작 주파수는 100MHz로 가정하였고, CMOS 180nm 공정을 이용하여 합성을 진행하였다. 그림 4의 (a)는 SBM과 KOA의 하드웨어 면적을 비교한 결과이다. m이 4일때는 SBM에 필요한 XOR과 AND의 개수가 9개, 16개이고 KOA의 경우 24개와 9개로 SBM이 더 작은 형태를 보였으나, m이 커질수록 SBM과 KOA의 면적 차이가 줄어드는 것을 알 수 있다. 그림 4의 (b)는 처리량을 나타내며, m이 32인 경우를 기점으로 KOA의 처리량이 SBM보다 커지며, m이 128인 경우 KOA의 처리량이 SBM의 처리량보다 약 50% 크다. 따라서 m이 32보다 큰 경우에는 KOA가 면적대비 높은 처리량을 가져 SBM보다 유리한 형태를 보임을 확인할 수 있다.

Ⅳ. 결론 및 향후 연구 방향

본 논문에서는 GF(2^m)의 GF 곱셈기를 하드웨어로 구현하는 다양한 방법 가운데 SBM과 KOA를 적용한 하드웨어를 m을 다르게 하여 구현하고, 그 결과를 비교하였다. m이 커질수록 처리량의 차이가 커지며, GF(2^{128})에서 KOA의 처리량이 SBM에 비해 약 50% 크다. 따라서 본 논문의 실험 결과를 바탕으로 GF 곱셈이 필요한 경우에 제약조건에 따라 회로 구성에 효율적인 GF 곱셈기의 구조를 선택하여 시스템의 성능을 최적화할 수 있다.

참고문헌

[1] Jagannath Samanta, Razia Sultana, Jaydeb Bhaumik, "FPGA based modified Karatsuba multiplier", International Conference on VLSI and Signal Processing (ICVSP), vol. 10-12, January 2014.

[2] Wibowo FW 2018 Comparison of Multiplication Algorithms Based on FPGA Proc. of 2018 2nd Borneo International Conference on Applied Mathematics and Engineering (BICAME)

[3] A. Karatsuba and Y. Ofman, "Multiplication of Many-Digital Numbers by Automatic Computers," Doklady Akad. Nauk SSSR, vol. 145, 1962.

[4] A. Weimerskirch and C. Paar, Generalizations of the Karatsuba Algorithm for Efficient Implementations, Cryptology ePrint Archive, Report 2006/224.

[5] Y. Zhang, et al.: "High performance AES-GCM implementation based on efficient AES and FR-KOA multiplier," IEICE Electron. Express 15 (2018).