JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 24, No. 6: 736~743, Jun. 2020

듀얼 필드 모듈러 곱셈을 지원하는 몽고메리 곱셈기

김동성[1] · 신경욱[2*]

Montgomery Multiplier Supporting Dual-Field Modular Multiplication

Dong-Seong Kim[1] · Kyung-Wook Shin[2*]

[1]Graduate Student, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk, 39177, Korea
[2*]Professor, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

요 약

모듈러 곱셈은 타원곡선 암호 (elliptic curve cryptography; ECC), RSA 등의 공개키 암호에서 중요하게 사용되는 산술연산 중 하나이며, 모듈러 곱셈기의 성능은 공개키 암호 하드웨어의 성능에 큰 영향을 미치는 핵심 요소가 된다. 본 논문에서는 워드기반 몽고메리 모듈러 곱셈 알고리듬의 효율적인 하드웨어 구현에 대해 기술한다. 본 논문의 모듈러 곱셈기는 SEC2 ECC 표준에 정의된 소수체 $GF(p)$와 이진체 $GF(2^k)$ 상의 11가지 필드 크기를 지원하여 타원곡선 암호 프로세서의 경량 하드웨어 구현에 적합하도록 설계되었다. 제안된 곱셈기 구조는 부분곱 생성 및 가산 연산과 모듈러 축약 연산이 파이프라인 방식으로 처리하며, 곱셈 연산에 소요되는 클록 사이클 수를 약 50% 줄였다. 설계된 모듈러 곱셈기를 FPGA 디바이스에 구현하여 하드웨어 동작을 검증하였으며, 65-nm CMOS 표준셀로 합성한 결과 33,635개의 등가 게이트로 구현되었고, 최대 동작 클록 주파수는 147 MHz로 추정되었다.

ABSTRACT

Modular multiplication is one of the most important arithmetic operations in public-key cryptography such as elliptic curve cryptography (ECC) and RSA, and the performance of modular multiplier is a key factor influencing the performance of public-key cryptographic hardware. An efficient hardware implementation of word-based Montgomery modular multiplication algorithm is described in this paper. Our modular multiplier was designed to support eleven field sizes for prime field $GF(p)$ and binary field $GF(2^k)$ as defined by SEC2 standard for ECC, making it suitable for lightweight hardware implementations of ECC processors. The proposed architecture employs pipeline scheme between the partial product generation and addition operation and the modular reduction operation to reduce the clock cycles required to compute modular multiplication by 50%. The hardware operation of our modular multiplier was demonstrated by FPGA verification. When synthesized with a 65-nm CMOS cell library, it was realized with 33,635 gate equivalents, and the maximum operating clock frequency was estimated at 147 MHz.

키워드 : 듀얼 필드, 타원곡선 암호, 갈루이 체, 모듈러 곱셈, 몽고메리 곱셈기

Key word : Dual-field, Elliptic curve cryptography, Galois field, Modular multiplication, Montgomery multiplier

Received 25 March 2020, **Revised** 16 April 2020, **Accepted** 21 April 2020

*** Corresponding Author** Kyung-Wook Shin(E-mail:kwshin@kumoh.ac.kr, Tel:+82-54-478-7427)
Professor, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

Open Access http://doi.org/10.6109/jkiice.2020.24.6.736 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

전자서명, 인증, 키교환 등 대표적인 정보보안 프로토콜들은 타원곡선 암호 (elliptic curve cryptography; ECC) 또는 RSA (Rivest, Shamir, Adleman) 공개키 암호 방식을 기반으로 한다. ECC, RSA, EC-DSA 등을 포함하는 대부분의 공개키 암호 방식은 유한체 상의 덧셈, 곱셈, 나눗셈, 역원 (inversion) 등 모듈러 (modular) 연산을 기반으로 구현된다. RSA 공개키 암호에서는 암호화와 복호화를 위한 모듈러 멱승 (exponentiation)이 모듈러 곱셈의 반복 연산으로 구현된다. ECC의 스칼라 곱셈은 유한체 상의 모듈러 곱셈, 모듈러 나눗셈 및 모듈러 역원의 반복 연산으로 계산된다. 타원곡선 디지털서명 알고리듬 (EC-DSA)에서 서명 생성과 검증을 위한 연산에도 모듈러 연산이 사용된다 [1-4].

공개키 암호에 사용되는 모듈러 연산은 수 백 ~ 수 천 비트의 매우 큰 소수 (prime number)를 법 (modulo)으로 갖는 모듈러 곱셈, 나눗셈, 역원 연산이며, 연산 복잡도가 크고 요구되는 연산량이 매우 많다. 따라서 공개키 암호 시스템의 처리성능은 모듈러 연산의 성능에 크게 영향을 받으며, 공개키 암호의 하드웨어 구현을 위해서는 효율적인 모듈러 연산 알고리듬과 하드웨어 구조에 대한 연구가 필요하다.

하드웨어 자원의 사용을 최소화하면서 최대의 연산 성능을 얻기 위해서는 효율적인 모듈러 곱셈기 설계가 필수적이며, 모듈러 곱셈을 효율적으로 연산하기 위한 다양한 알고리듬들이 제안되고 있다. 공개키 암호의 하드웨어 구현에 널리 사용되고 있는 방법 중 하나는 몽고메리 (Montgomery) 모듈러 곱셈 알고리듬 [5]이다. 몽고메리 모듈러 곱셈 알고리듬은 나눗셈 연산을 사용하지 않기 때문에 하드웨어 구현에 적합하다는 특징이 있다. 효율적인 몽고메리 곱셈기의 설계 사례들이 발표되고 있으며 [6-7], Koc는 5가지 몽고메리 곱셈기의 구조를 제안하였다 [8].

본 논문에서는 워드기반 몽고메리 모듈러 곱셈의 효율적인 구현 방법과 하드웨어 구조를 제안한다. 워드기반 몽고메리 곱셈에 소요되는 클록 사이클 수를 줄일 수 있도록 부분곱 생성 및 가산과 모듈러 축약 (modular reduction) 연산이 파이프라인으로 처리되는 방식을 제안하였다. 본 논문의 몽고메리 곱셈기는 타원곡선 암호의 하드웨어 구현에 적용될 수 있도록 소수체 (prime field) $GF(p)$와 이진체 (binary field) $GF(2^k)$ 상의 유한체 연산을 지원하도록 설계되었으며, FPGA 구현을 통해 하드웨어 동작을 검증하였다. II장에서는 워드기반 몽고메리 모듈러 곱셈 알고리듬과 본 논문에서 제안하는 연산방식에 대해 기술하고, III장에서는 제안된 방법을 적용한 몽고메리 모듈러 곱셈기 설계에 대해 설명한다. IV장에서는 설계된 모듈러 곱셈기의 검증과 성능분석에 대해 기술하고, V장에서 결론을 맺는다.

Ⅱ. 워드기반 몽고메리 모듈러 곱셈

2.1. 듀얼 필드 워드기반 몽고메리 곱셈 알고리듬

모듈러 곱셈을 연산하는 직접적인 방법은 n-비트의 승수와 피승수를 곱해서 $2n$-비트의 결과를 얻고, 이를 다시 n-비트의 법 p로 나누어 n-비트의 모듈러 곱셈 결과를 얻는 것이다. 이와 같은 직접적인 방법은 나눗셈 연산을 필요로 하므로, 연산에 소요되는 시간이 길어지는 단점이 있다. 몽고메리 모듈러 곱셈 알고리듬은 나눗셈 연산을 사용하지 않고 단순한 시프트와 가산 연산만으로 모듈러 곱셈을 연산하는 효율적인 방법이다. 법 p가 n-비트의 소수이고, p 보다 작은 n-비트의 정수 A, B를 몽고메리 곱셈 알고리듬으로 연산하면, 곱셈 결과는 $S = A \times B \times R^{-1} \bmod p$가 되며, $R = 2^n$이다 [5].

워드기반 몽고메리 곱셈 알고리듬은 부분곱 생성 및 가산 연산과 모듈러 축약 연산이 w-비트의 워드단위로 처리되는 방식이다. 소수체 $GF(p)$와 이진체 $GF(2^k)$를 지원하는 워드기반 몽고메리 모듈러 곱셈 알고리듬은 그림 1의 슈도코드로 표현된다. 승수 A, 피승수 B 그리고 법 p가 모두 n-비트이고 워드 크기가 w-비트이면, 이들은 $m = \lceil n/w \rceil$ 워드로 구성된다. 승수와 피승수의 아래 첨자 i, j는 각각 i-번째, j-번째 워드를 나타낸다. 승수 A와 피승수 B는 $GF(p)$ 또는 $GF(2^k)$의 원소이다. 법 p는 $GF(p)$ 상의 연산에서는 소수이고, $GF(2^k)$에서는 기약 다항식 (irreducible polynomial)의 계수로 표현되는 값이다. 그림 1의 슈도코드에서 $p_0^* = -p_0^{-1} \bmod 2^w$는 법 p의 최하위 워드 p_0로부터 계산되며, 모듈러 연산의 합동 (congruence) 특성을 이용하여 부분곱 가산결과의 최하위 워드를 0으로 만드는 축약연산에 사용된다.

그림 1의 슈도코드에서 단계-2의 i-루프 반복문은 승수의 각 워드 A_i에 대한 반복 연산을 나타내며, 단계-5와 단계-16의 j-루프 반복문은 피승수의 각 워드 B_j에 대한 반복 연산을 나타낸다. 또한, $PF=1$인 경우의 단계-4 ~ 단계-14는 소수체 $GF(p)$ 상의 모듈러 곱셈 연산을 나타내고, $PF=0$인 경우의 단계-15 ~ 단계-26은 이진체 $GF(2^k)$ 상의 모듈러 곱셈 연산을 나타낸다. j-루프 반복문 내부의 연산은 부분곱 생성 및 가산 그리고 모듈러 축약의 두 부분으로 구성된다. $PF=1$인 경우의 단계-6 ~ 단계-7은 승수의 워드 A_i와 피승수의 워드 B_j가 곱해져 부분곱이 생성되고, 가산되는 연산을 나타내며, 단계-8 ~ 단계-12는 모듈러 축약 연산을 나타낸다. 마찬가지로, $PF=0$인 경우의 단계-17 ~ 단계-18은 승수의 워드 A_i와 피승수의 워드 B_j가 곱해져 부분곱이 생성되고, 가산되는 연산을 나타내며, 단계-19 ~ 단계-23은 모듈러 축약 연산을 나타낸다. $PF=0$인 이진체의 경우에는 가산 연산이 XOR로 구현된다. 단계-28 ~ 단계-32는 연산된 최종 곱셈 결과 S에 대해 마지막 모듈러 축약 연산을 수행하는 과정이다. $GF(p)$ 상의 연산에서는 $S \geq p$인 경우에 모듈러 축약이 수행된다. $GF(2^k)$ 상의 연산에서는 S 값이 $(k+1)$-비트가 되어 오버플로우 (overflow)가 발생되는 경우에 모듈러 축약이 수행된다.

$Input$; $Prime\ Field$: $A=\{A_{m-1},\cdots,A_1,A_0\}_{2^w},\ A\in GF(p)$
($PF=1$) $B=\{B_{m-1},\cdots,B_1,B_0\}_{2^w},\ B\in GF(p)$,
$p=\{p_{m-1},\cdots,p_1,p_0\}_{2^w},\ p\in\mathbb{P}$,
$Binary\ Field$: $A=\{A_{m-1},\cdots,A_1,A_0\}_{2^w},\ A\in GF(2^{wm})$,
($PF=0$) $B=\{B_{m-1},\cdots,B_1,B_0\}_{2^w},\ B\in GF(2^{wm})$,
$p=\{p_{m-1},\cdots,p_1,p_0\}_{2^w}$,
p is irreducible polynomial
$Pre\_computed: p_0^* = -p_0^{-1} \bmod 2^w$
$Output:\ S=\{S_{m-1},\cdots,S_1,S_0\}_{2^w} = AB(2^{wm})^{-1} \bmod p$

01 : $S \leftarrow 0$
02 : $for\ i=0\ to\ m-1\ do$
03 : $C1_{-1}\leftarrow 0,\ C2_{-1}\leftarrow 0,\ cy1\leftarrow 0,\ cy2\leftarrow 0$
04 : $if\ PF=1\ then$
05 : $for\ j=0\ to\ m-1\ do$
06 : $\{C1_j,P1\}\leftarrow A_i\times B_j$
07 : $\{\mathrm{cy1},H_j\}\leftarrow P1+C1_{j-1}+S_j+cy1$
08 : $if\ j=0\ then$
09 : $q_i\leftarrow H_j\times p_0^* \bmod 2^w$
10 : end if
11 : $\{C2_j,P2\}\leftarrow q_i\times p_j$
12 : $\{\mathrm{cy2},s_{j-1}\}\leftarrow P2+C2_{j-1}+H_j+cy2$
13 : $end\ for$
14 : $s_{m-1}\leftarrow C1_{m-1}+C2_{m-1}+cy1+cy2$
15 : $else\ if\ PF=0\ then$
16 : $for\ j=0\ to\ m-1\ do$
17 : $\{C1_j,P1\}\leftarrow A_i\times B_j$
18 : $H_j\leftarrow P1\oplus C1_{j-1}\oplus S_j$
19 : $if\ j=0\ then$
20 : $q_i\leftarrow H_j\times p_0^* \bmod 2^w$
21 : end if
22 : $\{C2_j,P2\}\leftarrow q_i\times p_j$
23 : $S_{j-1}\leftarrow P2\oplus C2_{j-1}\oplus H_j$
24 : $end\ for$
25 : $s_{m-1}\leftarrow C1_{m-1}\oplus C2_{m-1}$
26 : $end\ if$
27 : $end\ for$
28 : $if\ PF=1\ and\ S>p\ then$
29 : $S\leftarrow S-p$
30 : $else\ if\ PF=1\ and\ OVF=1\ then$
31 : $S\leftarrow S\oplus p$
32 : $end\ if$
33 : $return\ S$

Fig. 1 Pseudocode of dual-field word-based Montgomery modular multiplication algorithm.

2.2. 워드기반 몽고메리 곱셈 방법 제안

본 논문에서는 워드기반 몽고메리 곱셈의 효율적인 하드웨어 구현을 위해 그림 1의 슈도코드로 표현된 알고리듬을 그림 2의 연산 의존성 그래프로 매핑시켜 구현하는 방법을 제안한다. 그림 2에서 워드기반 연산을 나타내기 위해 승수 A와 피승수 B, 그리고 법 p를 m개의 워드 (워드는 w-비트)로 나누어 표현하였다. y축은 슈도코드의 j-루프 반복 연산을 나타내고, x축은 i-루프의 반복 연산을 나타낸다. 승수의 i-번째 워드 A_i와 피승수의 j-번째 워드 B_j에 대한 부분곱 생성 및 가산 (PPG and Add) 연산과 모듈러 축약 (Reduction) 연산이 파이프라인 방식으로 처리된다. 첫 번째 j-루프에서는 부분곱 생성 및 가산 연산만 수행되고, $(m+1)$-번째 j-루프에서는 모듈러 축약 연산이 수행되며, 나머지 j-루프에서는 부분곱 생성 및 가산 연산과 모듈러 축약 연산이 파이프라인 방식으로 수행된다.

$j=0$인 첫 번째 루프의 부분곱 생성 및 가산 연산에서는 승수의 i-번째 워드 A_i와 피승수의 최하위 워드 B_0가 곱해져 부분곱이 생성된다. 상위 32-비트는 $C1_0$에 저장되어 다음번 j-루프의 부분곱 생성 및 가산에 입력으로 사용된다. 하위 32-비트는 이전 i-루프 연산 결과인 S_0와 더해져 p_0^*와 함께 q_i 계산에 사용되고, 다음번 j-루프에서 모듈러 축약 연산에 입력으로 사용된다.

$j=1$에서 $j=m-1$까지 j-루프의 부분곱 생성 및 가산 연산에서는 승수의 i-번째 워드 A_i와 피승수의

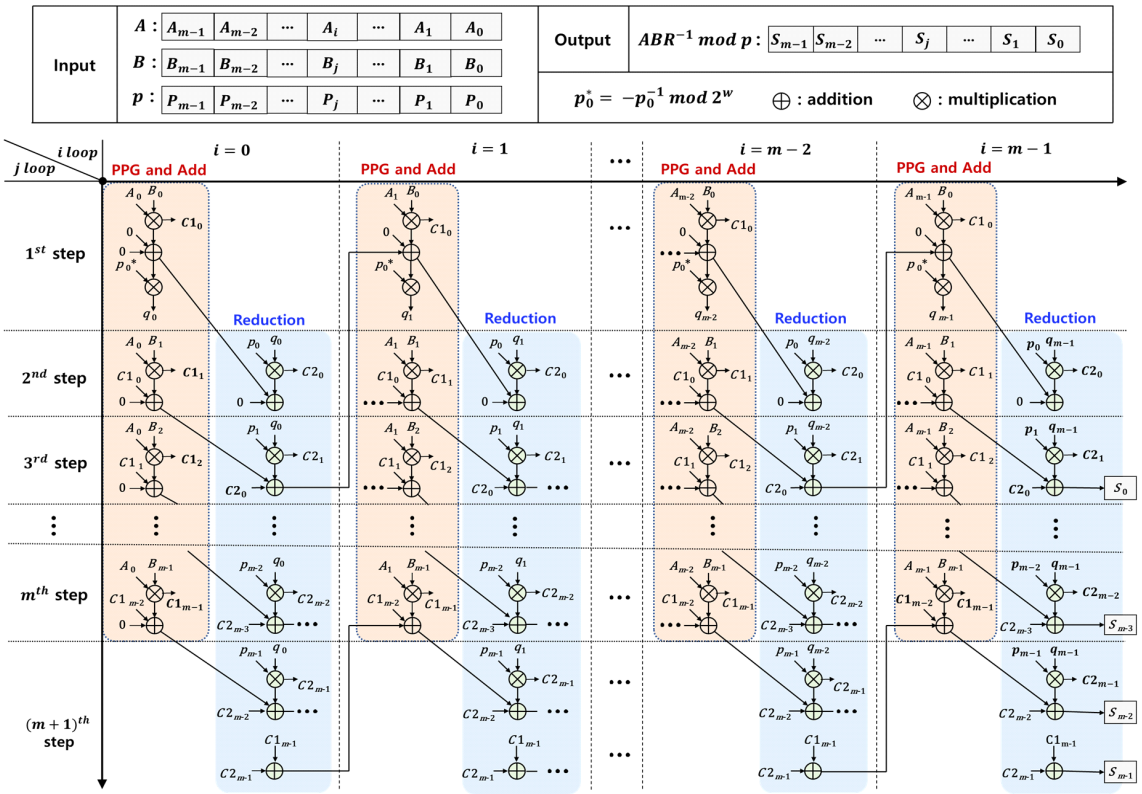


Fig. 2 Dependency graph for word-based Montgomery modular multiplication.

j-번째 워드 B_j를 곱하여 부분곱을 생성한다. 상위 32-비트는 첫 번째 j-루프가 실행될 때와 동일하며, 하위 32-비트는 이전 i-루프 연산 결과인 S_j, 이전 j-루프에서 계산된 부분곱 생성 및 가산 연산의 캐리 출력 $C1_{j-1}$과 더해진다. 계산된 결과는 다음 j-루프가 실행될 때 부분곱 생성 및 가산 연산에 입력으로 사용된다. 모듈러 축약 연산에서는 p_{j-1}과 q_i가 곱해진다. 상위 32-비트는 다음 j-루프 연산에서 모듈러 축약 연산의 입력으로 사용되기 위해 $C2_{j-1}$에 저장되며, 하위 32-비트는 이전 j-루프에서 부분곱 생성 및 가산 연산을 통해 계산된 결과와 모듈러 축약 연산에서 계산된 모듈러 축약 연산의 캐리출력 $C2_{j-2}$와 더해져 S_{j-2}에 저장된다.

$(m+1)$-번째 j-루프의 모듈러 축약 연산에서 p_{m-1}과 q_i의 곱셈 결과 중, 상위 32-비트는 S_{m-1} 계산을 위해 $C2_{m-1}$에 저장된다. 하위 32-비트는 이전 j-루프의 부분곱 생성 및 가산 연산 결과와 모듈러 축약 연산으로 계산된 $C2_{m-2}$와 더해져 S_{m-2}가 계산된다. 이후 $C1_{m-1}$과 $C2_{m-1}$이 더해져 S_{m-1}이 생성된다.

위 과정을 통해 i-루프가 한번 실행되면 S_{-1}부터 S_{m-1}까지 계산되며, 이 값들은 다음 번 i-루프에서 부분곱 생성 및 가산 연산의 입력으로 사용된다. i-루프가 한 번 실행되면 하위 32-비트가 0이면서 유한체 상에서 합동인 값이 계산되며, 이 값이 오른쪽으로 32-비트 시프트된다. 따라서 워드기반 몽고메리 모듈러 곱셈의 결과는 $ABR^{-1} \bmod p$가 되며, 여기서 $R = 2^{mw}$ 이다.

이진체 $GF(2^k)$ 상의 모듈러 곱셈에서 덧셈 연산은 XOR 연산으로 구현되어 덧셈에 의한 캐리가 발생하지 않는다.

본 논문에서 제안하는 그림 2의 워드기반 몽고메리 모듈러 곱셈 방식은 부분곱 생성 및 가산과 모듈러 축약 연산이 파이프라인 방식으로 처리되며, 이들 두 연산을 순차적으로 처리하는 방식 [9]에 비해 연산에 소요되는 클록 사이클 수가 약 절반으로 감소되어 고속 모듈러 곱셈 연산이 가능하다 [10].

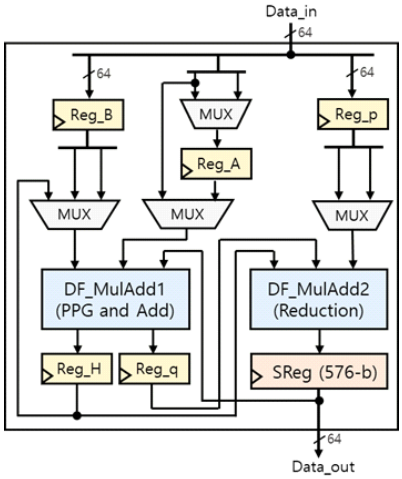


Fig. 3 Architecture of word-based Montgomery multiplier.

Ⅲ. 워드기반 몽고메리 곱셈기 설계

그림 2의 연산 의존성 그래프를 기반으로 그림 3의 구조를 갖는 워드기반 몽고메리 곱셈기를 설계하였다. 워드 단위로 연산이 처리되도록 내부 데이터 패스는 32-비트로 설계되었으며, 메모리 액세스에 소요되는 클록 사이클을 줄이기 위해 입출력 포트를 64-비트로 하였다. 소수체와 이진체 상의 곱셈과 덧셈 연산을 수행하는 두 개의 DF_MulAdd 블록과 입력 데이터 및 중간 연산결과를 저장하는 레지스터들로 구성된다. 64-비트 레지스터인 Reg_B와 Reg_p에는 각각 피승수 B와 법 p의 32-비트 워드 두 개씩 저장된다. 또한, 32-비트 레지스터 Reg_A에는 승수 A 값이 워드 단위로 저장되며, 32-비트 레지스터 Reg_q에는 q_i 값이 저장된다. 레지스터 SReg는 SEC 2 [11] 표준 문서에 정의된 타원곡선 중 키 길이가 가장 큰 타원곡선인 B571R, B571K을 지원할 수 있도록 576-비트 크기로 구현되었다.

DF_MulAdd1 블록은 그림 2에서 부분곱 생성과 가산 (PPG and Add) 연산을 수행하며, 그림 4와 같이 32-비트 곱셈기 DF_Mul32b와 32-비트 캐리선택 가산기 DF_CSA_32b로 구성된다. 승수의 i-번째 워드 A_i, 피승수의 j-번째 워드 B_j의 부분곱 연산이 DF_Mul32b를 통해 수행되며, 생성된 64-비트 부분곱의 상위 32-비트는 내부 레지스터에 저장되어 부분곱 가산에 사용된다.

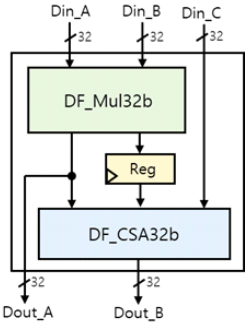


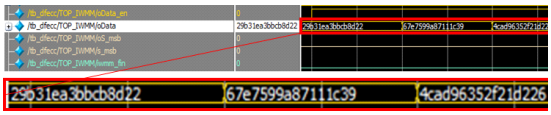
Fig. 4 Dual-field multiplier and adder block.

DF_CSA32b는 32-비트 캐리선택 가산기 두 개로 구성되며, DF_Mul32b에서 생성된 부분곱, 이전 j-루프에서 생성된 $C0_{j-1}$ 그리고 이전 i-루프의 DF_MulAdd2 블록에서 계산된 S_j의 가산이 수행된다. DF_CSA32b를 통해 계산된 결과는 레지스터 Reg-H에 저장된다. 첫 번째 j-루프의 연산에서는 가산기 DF_CSA32b에서 연산된 결과와 p_0^*의 곱셈이 DF_Mul32b를 통해 수행되고, 연산 결과의 하위 32-비트만 출력되어 다음 j-루프에서 DF_MulAdd2 블록으로 전달된다.

DF_MulAdd2 블록은 그림 2에서 Reduction으로 표시된 모듈러 축약 연산을 수행하며, 그림 4와 동일한 구조를 갖는다. p_{j-1}과 q_i의 곱셈 연산은 DF_Mul32b를 통해 수행되고, 곱셈 연산 결과의 상위 32-비트는 내부 레지스터에 저장되어 다음 j-루프에서 사용된다. 이전 j-루프에서 DF_Mul32b의 곱셈 연산 결과의 상위 32-비트와 현재 j-루프에서 DF_Mul32b의 곱셈 연산 결과의 하위 32-비트 그리고 이전 j-루프에서 DF_MulAdd1 블록의 연산 결과가 DF_CSA32b를 통해 가산된다.

본 논문에서 설계된 그림 3의 워드기반 몽고메리 곱셈기는 모듈러 곱셈 연산에 $m^2+3.5m+4$ (m이 짝수인 경우) 클록 사이클 또는 m^2+4m+5 (m이 홀수인 경우) 클록 사이클이 소요된다. 여기서 m은 승수와 피승수를 구성하는 워드의 개수를 나타낸다. 한편, DF_MulAdd 블록 내부의 가산기를 통해 모듈러 덧셈과 모듈러 뺄셈 연산이 가능하도록 설계되었다. 소수체상의 모듈러 덧셈/뺄셈 연산은 $2.5m+5$ (m이 짝수인 경우) 또는 $2.5(m+1)+4$ (m이 홀수인 경우) 클록 사이클이 소요된다. 이진체상의 모듈러 덧셈 연산은 $m+3$ (m이 짝수인 경우) 클록 사이클 또는 $m+4$ (m이 홀수인 경우) 클록 사이클이 소요된다.

Ⅳ. 하드웨어 동작 검증 및 성능 평가



(a)

```
########################################################################################
####### [ The result of a Montgomery modular multiplication when w is 32  ] ######
########################################################################################

Multiplier A :  0xdb4ff10ec057e9ae26b07d0280b7f4341da5d1b1eae06c7d
Multiplicator B :  0x9b2f2f6d9c5628a7844163d015be86344082aa88d95e2f9d
Prime number P :  0xffffffffffffffffffffffffffffffffffffffffffffffffffffffee37

The result of a WMM :  0x4cad96352f21d22667e7599a87111c3929b31ea3bbcb8d22
AB(R^(-1)) mod p :  0x4cad96352f21d22667e7599a87111c3929b31ea3bbcb8d22

*******************************************
 The calculation results are matched!
*******************************************
```

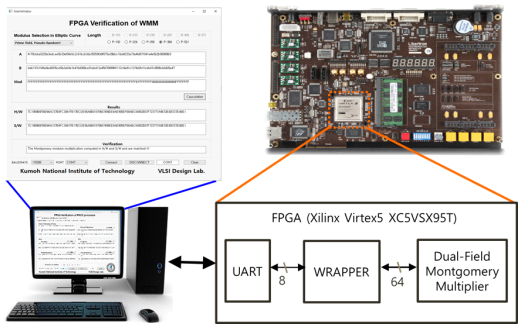
(b)

Fig. 5 (a) RTL simulation result of the designed modular multiplier, (b) modular multiplication result obtained by Python software.

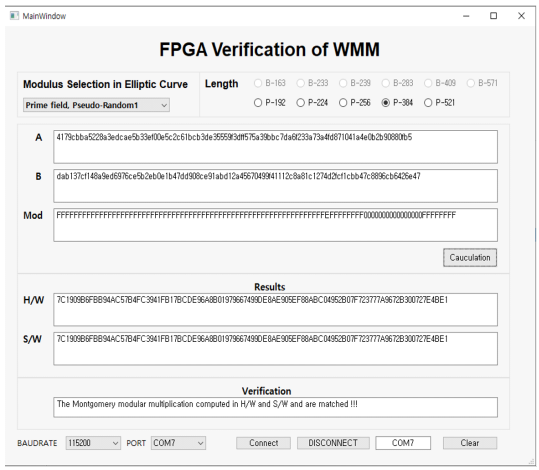
설계된 워드기반 몽고메리 모듈러 곱셈기는 RTL 시뮬레이션을 통해 기능검증을 수행하고, FPGA 구현을 통해 하드웨어 동작을 검증하였다. 그림 5는 설계된 몽고메리 곱셈기의 기능검증 결과 중 일부를 보인 것이다. 소수체 상의 연산에서 법 p의 값이 “0x ffff ffff ffff ffff ffff ffff ffff ffff ffff fffe ffff ee37”인 경우에, 승수 “0x db4f f10e c057 e9ae 26b0 7d02 80b7 f434 1da5 d1b1 eae0 6c7d”와 피승수 “0x 9b2f 2f6d 9c56 28a7 8441 63d0 15be 8634 4082 aa88 d95e 2f9d”의 모듈러 곱셈 결과가 “0x 4cad 9635 2f21 d226 67e7 599a 8711 1c39 29b3 1ea3 bbcb 8d22”이며, 그림 5-(a)의 RTL 시뮬레이션 결과와 그림 5-(b)의 Python 소프트웨어 연산 결과가 일치함을 확인할 수 있다.

설계된 모듈러 곱셈기를 Xilinx Virtex-5 FPGA 디바이스에 구현하여 하드웨어 동작을 검증하였다. FPGA 검증 시스템은 그림 6-(a)와 같으며, 그림 6-(b)는 법 p가 384-비트인 경우에 대한 검증결과 화면캡처를 보인 것이다. FPGA 디바이스에 구현된 몽고메리 모듈러 곱셈기 동작에 의한 연산 결과와 소프트웨어로 계산된 결과가 정확하게 일치하여 설계된 몽고메리 곱셈기가 올바로 동작함을 확인하였다.

표 1은 본 논문에서 설계된 모듈러 곱셈기를 문헌에 발표된 사례와 비교를 보인 것이다. 최대 동작주파수는 71.4 MHz로 문헌 [12-14]의 사례에 비해 작은 값으로



(a)



(b)

Fig. 6 (a) FPGA verification setup, (b) screen shot of verification result of modular multiplier

추정되었으며, 처리율도 비교적 낮다. 그러나 본 논문의 모듈러 곱셈기는 소수체와 이진체를 모두 지원하며, 192-비트 ~ 576-비트의 11가지 필드 크기를 지원하면서도 적은 하드웨어를 사용하는 장점을 갖는다.

Ⅴ. 결 론

소수체와 이진체 상의 모듈러 곱셈, 모듈러 덧셈, 모듈러 뺄셈 연산을 지원하는 워드기반 몽고메리 곱셈기의 효율적인 하드웨어 구현 방법과 설계 결과를 제시하였다. 부분곱 생성 및 가산 연산과 모듈러 축약 연산이 파이프라인 방식으로 처리되도록 하여 모듈러 곱셈 연산에 소요되는 사이클을 감소시켰다. Virtex-5 FPGA 디바이스에 구현하여 하드웨어 동작을 검증하였으며, 합

Table. 1 Comparison of modular multipliers.

	[12]	[13]	[14]	This paper
Supported field	GF(p)	GF(p)	GF(p)	GF(p) and $GF(2^k)$
Operands size [bit]	512	256	512	192～576
FPGA device	Vertex-5	Vertex-6	Vertex-7	Vertex-5
Number of slice registers	4,121	2,061	3,739	1,059
Number of slice LUT	4,652	23,977	4,608	4,127
Max. frequency [MHz]	430.84	204	456.6	71.4
Number of clock cycles	-	50 @256-bit	266 @512-bit	316 @512-bit
Throughput [Mbps]	427.5	1,044	878.9	115.7

성 결과 1,712 슬라이스로 구현되었다. 최대 동작주파수는 71.4 MHz로 추정되었으며, 512-비트 모듈러 곱셈 연산의 처리율은 115.7 Mbps로 평가되었다. 65-nm CMOS 표준셀로 합성한 결과, 동작 주파수 100 MHz에서 33,635 등가 게이트로 구현되었으며, 최대 동작주파수는 147 MHz로 추정되었다. 최대 동작주파수와 처리율이 다소 낮은 편이나, SEC2에 정의된 11가지 필드 크기의 모듈러 곱셈 연산이 가능하고, 적은 하드웨어로 구현이 가능하기 때문에 경량 ECC 프로세서 설계에 적합한 것으로 평가된다.

ACKNOWLEDGEMENT

This research was supported by Kumoh National Institute of Technology (2019-104-072)

REFERENCES

[1] Z. Liu, D. Liu, and X. Zou, “An Efficient and Flexible Hardware Implementation of the Dual-Field Elliptic Curve Cryptographic Processor,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2353-2362, Mar. 2017.

[2] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales- Sandoval, “Elliptic Curve Lightweight Cryptography: A Survey,” in *IEEE Access*, vol. 6, pp. 72514-72550, 2018.

[3] N. Thampi, and M. E. Jose, “Montgomery Multiplier for Faster Cryptosystems,” *Procedia Technology*, vol. 25, pp. 392-398, 2016.

[4] D. B. Roy, and D. Mukhopadhyay, “High-Speed Implementation of ECC Scalar Multiplication in GF(p) for Generic Montgomery Curves,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1587-1600, July 2019.

[5] P. L. Montgomery, “Modular multiplication without trial division,” *Mathematics of computation*, vol. 44, no. 170, pp. 519-521, Apr. 1985.

[6] A. Nadjia, and A. Mohamed, “High Throughput Parallel Montgomery Modular Exponentiation on FPGA,” in *Proceeding of the 9th International Symposium on Design and Test*, Algiers, pp. 225-230, 2014.

[7] A. Rezai, and P. Keshavarzi, “High-throughput Modular Multiplication and Exponentiation Algorithms using Multibit-scan-multibit-shift Technique,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1710-1719, Sep. 2015.

[8] C. K. Koc, T. Acar, and B. S. Kaliski, “Analyzing and Comparing Montgomery Multiplication Algorithms,” *IEEE Micro*, vol. 16, no. 3, pp. 26-33, Jun. 1996.

[9] S. H. Lee, “A Lightweight ECC Processor Supporting Dual Field Elliptic Curves of *GF(p)* and *GF(2^m)*,” Master’s Thesis, Kumoh National Institute of Technology, Jun. 2019.

[10] D. S. Kim, and K. W. Shin, “A Design of Montgomery Modular Multiplier supporting Prime Field and Binary Field,” *Proceedings of 2019 1st Conference of the Institute of Electronics and Information Engineers*, vol. 42, no. 1, pp. 54-55, Jeju, 2019.

[11] SEC 2. “Elliptic Curve Cryptography,” *Standards for Efficient Cryptography Group*, Sep. 2000.

[12] R. Verna, M. Duttam, and R. Vig, “FPGA Implementation of Modified Montgomery for RSA Cryptosystem,” *International Journal of Computer Science and Telecommunication*, vol. 4, no. 1, pp. 42-46. Jan. 2013.

[13] Y. Yang, C. Wu, Z. Li, and J. Yang “Efficient FPGA Implementation of Modular Multiplication based on

Montgomery Algorithm," *Microprocessors and Microsystems*, vol. 47, pp. 209-215, Apr. 2016.

[14] S. Erdem, T. Yamk, and A. Celebi, "A General Digit-Serial Architecture for Montgomery Modular Multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1658-1668. May 2017.

김동성(Dong-Seong Kim)

2018 : BS degree in Electronic Engineering, Kumoh National Institute of Technology
2020 : MS degree in Electronic Engineering, Kumoh National Institute of Technology

신경욱(Kyung-Wook Shin)

1984 : BS degree in Electronic Engineering, Korea Aerospace University
1986 : MS degree in Electronic Engineering, Yonsei University
1990 : Ph.D. degree in Electronic Engineering, Yonsei University
1990~1991 : Senior Researcher, Semiconductor Research Center, Electronics and Telecommunications Research Institute (ETRI)
1991~ : Professor in School of Electronic Engineering, Kumoh National Institute of Technology
1995~1996 : University of Illinois at Urbana- Champaign (Visiting Professor)
2003~2004 : University of California at San Diego (Visiting Professor)
2013~2014 : Georgia Institute of Technology (Visiting Professor)