3-way Toom-Cook 곱셈 알고리듬과 고속 축약 알고리듬을 이용한 256-비트 모듈러 곱셈기 설계

양현준* · 신경욱

금오공과대학교

A Design of 256-bit Modular Multiplier using 3-way Toom-Cook Multiplication Algorithm and Fast Reduction Algorithm

Hyeon-Jun Yang* · Kyung-Wook Shin

Kumoh National Institute of Technology

E-mail : 20206038@kumoh.ac.kr / kwshin@@kumoh.ac.kr

요 약

모듈러 곱셈은 ECC의 점 스칼라 곱셈을 위한 핵심 연산이며, ECC 프로세서의 성능에 영향을 미치는 가장 중요한 요소이다. 본 논문에서는 3-way Toom-Cook 곱셈 알고리듬과 수정된 고속 축약 알고리듬을 적용한 256-비트 모듈러 곱셈기 설계에 대해 기술한다. 90-비트 곱셈기 1개와 264-비트 가산기 3개가 사용되었으며, 하드웨어 크기와 소요 클록 사이클 수 사이의 최적화를 이루었다. Zynq UltraScale+ MPSoC 디바이스에 구현하여 모듈러 곱셈기를 검증하였으며, 모듈러 곱셈 연산에 15 클록 사이클이 소요된다.

ABSTRACT

Modular multiplication is a key operation for point scalar multiplication of ECC, and is the most important factor affecting the performance of ECC processor. This paper describes a design of a 256-bit modular multiplier that adopts 3-way Toom-Cook multiplication algorithm and modified fast reduction algorithm. One 90-bit multiplier and three 264-bit adders were used to optimize the hardware size and the number of clock cycles required. The modular multiplier was verified by implementing it using Zynq UltraScale+ MPSoC device and the modular multiplication operation takes 15 clock cycles.

키워드

Modular multiplication, Toom-Cook multiplication, modular reduction, integer multiplication

Ⅰ. 서 론

모듈러 곱셈은 타원곡선 암호(elliptic curve cryptography; ECC)의 점 스칼라 곱셈을 위한 핵심 연산이며, ECC 프로세서의 성능에 영향을 미치는 가장 중요한 요소이다. 모듈러 곱셈을 구현하는 방법에는 몽고메리 곱셈 알고리듬 [1]과 같이 곱셈과 축약 연산을 하나의 알고리듬으로 구현하는 방법과 정수 곱셈 후 축약 연산으로 모듈러 곱셈 결과값을 얻는 방법 등이 있다. 본 논문에서는 3-way Toom-Cook 정수 곱셈 알고리듬과 고속 축약 알고리듬을 사용하여 NIST의 P-256 곡선에 적합한 256-비트 모듈러 곱셈기를 설계하고, Zynq UltraScale+ MPSoC 디바이스에 검증하였다. II장은 모듈러 곱셈에 사용된 정수 곱셈과 축약 연산 알고리듬을 소개한다. III장은 모듈러 곱셈기의 구조와 RTL 시뮬레이션 결과를 보이며, IV장에서 결론을 맺는다.

* speaker

II. Toom-Cook 알고리듬과 고속 축약 알고리듬

Toom-Cook 알고리듬은 매우 큰 정수의 곱셈을 고속으로 계산하기 위해 고안되었다 [2,3]. 피승수 A와 승수 B를 각각 d개의 부분(d>1)으로 나누어 연산하며, 이를 d-way Toom-Cook이라고 한다. d가 클수록 내부 연산량이 증가하지만, 연산 시간 복잡도 $O(N^{\log_d(2d-1)})$가 감소하는 특징을 갖는다 [4]. N은 피승수와 승수의 크기를 나타내며, Toom-Cook 알고리듬은 분할(splitting), 평가(evaluation), 재귀적 곱셈(recursive multiplication), 보간(interpolation), 재구성(recomposition)의 5단계로 구성된다. 본 논문에서는 하드웨어 크기와 소요 사이클의 최적화를 위해 d=3인 3-way Toom-Cook 알고리듬을 기반으로 하는 TC3W 알고리듬을 사용하였다. TC3W 알고리듬은 3-way Toom-Cook 알고리듬의 단점인 하드웨어 구현 시 많은 자원이 필요한 (1/3, 1/6) 연산을 제거하기 위해 문헌 [5]에 소개된 $d_3 = 3$을 곱하는 연산이 추가된 알고리듬이다. 피승수 A와 승수 B의 곱셈 결과값 C는 $C = 3 \times A \times B$이 얻어진다.

고속 축약 알고리듬 (fast reduction algorithm)은 NIST에서 제공하는 소수체 (P-192, 224, 256, 384, 521)에 적합한 축약 알고리듬이다 [6]. 본 논문에서는 P-256 곡선에 적합한 고속 축약 알고리듬을 TC3W 정수 곱셈 알고리듬에 맞도록 수정하여 사용하였다.

설계된 모듈러 곱셈기의 결과값은 3이 곱해진 값이므로 결과값에 1/3 연산을 해주어야 한다. 하지만 모듈러 곱셈을 반복적으로 사용하는 ECC 연산에서 모든 모듈러 곱셈 결과마다 1/3 연산을 하는 것은 소요 사이클 측면에서 비효율적이다. 이를 개선하기 위해 Toom-Cook 도메인을 적용하였다 [5]. Toom-Cook 도메인은 피승수 A와 승수 B가 각각 $A \times d_3^{-1} \bmod p_{256}$, $B \times d_3^{-1} \bmod p_{256}$와 같이 표현된 데이터들을 의미하며, d_3^2의 모듈러 역원값과 매핑하고자 하는 데이터를 이용해 생성된다. 매핑된 데이터를 피승수와 승수로 사용할 경우 곱셈의 결과값 또한 Toom-Cook 도메인에 존재하므로 모듈러 곱셈을 반복적으로 수행하는 경우, 소요 사이클 측면에서 효율적이다. 모든 연산이 종료되면 리매핑 과정이 필요하며, 리매핑은 매핑된 데이터와 1을 모듈러 곱셈하여 얻어진다. 그림 1은 TC3W와 수정된 고속 축약(mFRed) 알고리듬을 기반으로 하는 모듈러 곱셈 알고리듬의 슈도코드이다. 단계-1은 Toom-Cook 도메인 매핑과정이며, 매핑된 데이터는 단계-2의 정수 곱셈과 단계-3의 축약 연산을 통해 모듈러 곱셈 결과값을 출력한다. 출력된 결과값은 Toom-Cook 도메인에 존재하므로 리매핑 과정을 통해 최종 결과값을 얻는다. 본 논문의 모듈러 곱셈기는 그림 1의 단계-2와 단계-3의 과정을 수행한다.

Input: $A = \{p_{3m-1}, p_{3m-2}, \cdots, p_1, p_0\}, (0 < A, B < 2^{256})$,
$B = \{q_{3m-1}, q_{3m-2}, \cdots, q_1, q_0\}, m = 86,\ p_i, q_i, r_i \in \{0,1\}$,
p_{256}
Output: $D = A \times B \bmod p_{256}$

01: $A^* \leftarrow$ ***Domain mapping*** (A, d_3^{-2}, p_{256});
$B^* \leftarrow$ ***Domain mapping*** (B, d_3^{-2}, p_{256});
02: $C^* \leftarrow$ ***TCM*** (A^*, B^*);
03: $D^* \leftarrow$ ***MR*** (C^*, p_{256});
04: $D \leftarrow$ ***Domain remapping*** $(D^*, 1, p_{256})$;
Return D

그림 1. TC3W와 mFRed 기반 모듈러 곱셈의 슈도코드

III. 256-비트 모듈러 곱셈기의 구조와 시뮬레이션 검증 결과

설계된 256-비트 모듈러 곱셈기의 블록도는 그림 2와 같이 FFAu 블록, FSM_FFAu 블록으로 구성된다. FFAu 블록은 유한체 연산인 모듈러 곱셈을 수행하는 블록으로 90-비트 곱셈기 1개와 264-비트 가산기 3개를 사용하며, 여러 피연산자를 한 번에 연산하는 알고리듬 특성상 메모리가 아닌 레지스터 파일을 사용하였다. FSM_FFAu 블록은 계수기와 유한상태머신을 이용해 FFAu 블록에 필요한 제어 신호를 생성한다.

그림 3은 설계된 모듈러 곱셈기의 RTL 기능검증 결과이며, 피승수 iA가 "0x6c0b5a78ac0e476f0b_7f885de4845b6abbe352f3ba05b4a946969ed6d5255556"이고, 승수 iB가 "0x888129fba9a95abd513a35231b_39146292c4465aa4723e89bd1d342df6d71187"일 때 모듈러 곱셈 결과값 oData가 "0xc4899a73119f45ee_ef570ad961d76185d90e5538b008a8c1a1ee0358ce751594"으로 얻어지며, 파이썬 모델과 일치하는 것을 확인하였다. 피승수와 승수는 모두 Toom-Cook 도메인의 데이터가 사용되었으며, 결과값 또한 Toom-Cook 도메인에 존재한다.

IV. 결 론

설계된 256-비트 모듈러 곱셈기는 NIST의 P-256 곡선에 적합하게 설계되었으며, Zynq UltraScale+

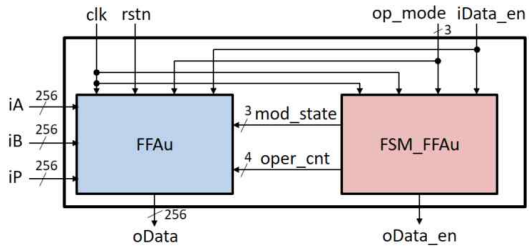

그림 2. 모듈러 곱셈기의 블록도

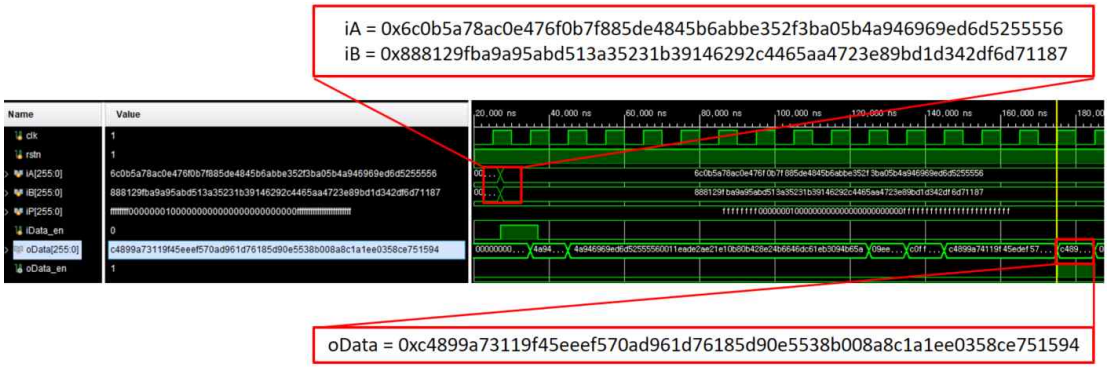

그림 3. 설계된 모듈러 곱셈기의 RTL 기능 검증 결과

MPSoC 디바이스에 구현하여 모듈러 곱셈기를 검증하였다. xczu7ev-ffvc1156 디바이스에서 27,552개의 LUT와 2,564개의 플립플롭 그리고 25개의 DSP 블록이 사용되었으며, 최대 동작 주파수는 140 MHz로 평가되었다. 140 MHz로 동작하는 경우, 초당 약 930만 번의 256-비트 모듈러 곱셈을 연산할 수 있다. 향후, 설계된 모듈러 곱셈기를 이용한 타원곡선 암호 프로세서 설계가 진행될 예정이다.

Acknowledgement

- This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1I1A3A04038083)
- The authors are thankful to IDEC for EDA tool support.

References

[1] P. L. Montgomery, “Modular multiplication without trial division,” Mathematics of Computation, vol. 44, no. 170, pp.519-521, May 1985.

[2] A. L. Toom, “The complexity of a scheme of functional elements realizing the multiplication of integers,” Soviet Math. Doklady, vol. 3, no. 4, pp. 714–716, 1963.

[3] S. A. Cook and S. O. Aanderaa, “On the minimum computation time of functions,” Trans. Amer. Math. Soc., vol. 142, pp. 291–314, Aug. 1969.

[4] M. J. Kronenburg. (Feb. 2016). “Toom-cook multiplication: Some theoretical and practical aspects.” [Online]. Available: https://arxiv.org/abs/1602.02740.

[5] J. Ding, S. Li and Z. Gu, “High-speed ECC processor over NIST prime fields applied with Toom–Cook multiplication,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 3, pp. 1003-1016, Mar. 2019.

[6] L. Chen (NIST), D. Moody (NIST), A. Regenscheid (NIST) and K. Randall (Randall Consulting), “Recommendations for discrete logarithm-based cryptography: elliptic curve domain parameters,” SP 800-186 (draft), Oct. 2019.