격자 기반 암호를 위한 파이프라인 방식의 NTT 하드웨어 구조

최소연, 김민수, 황지우, *유호영
충남대학교 전자공학과
e-mail : *sychoi.cas@gmail.com, mskim.cas@gmail.com, jwhwang.cas@gmail.com, hyyoo@cnu.ac.kr*

Pipelined NTT Hardware Architecture for Lattice-based Cryptography

Soyeon Choi, Minsu Kim, Jiwoo Hwang, and *Hoyoung Yoo
Department of Electronics Engineering
Chungnam National University

Abstract

The NTT processor is essential to implement lattice-based cryptography. The previous memory-based NTT hardware architectures are proposed to decrease hardware complexity, it takes long time and requires additional memories. In this paper, we propose a pipelined NTT hardware architecture that compensates for the problems of the memory-based architectures. The proposed structure continuously computes real-time data input. As a result, the proposed structure is more efficient than memory-based and fully parallel architecture in terms of latency and hardware complexity, respectively.

I. 서론

최근, 양자컴퓨터의 개발로 기존의 암호 시스템의 보안이 위협받고 있으며, 이에 대응하기 위하여 양자 내성 암호 (post quantum cryptography)가 제안되고 있다 [1]. 양자 내성 암호 가운데 격자 기반 (lattice-based)의 암호가 각광받고 있으며 [2], 격자 기반 암호의 연산 과정에는 finite ring에서 다항식 곱셈을 위한 NTT (Number Theoretic Transform)이 필요하다.

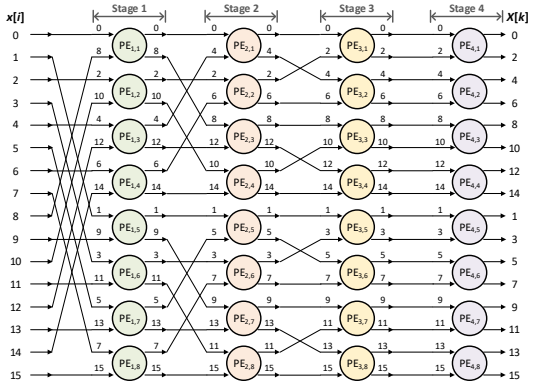

그림 1. N = 16인 경우 NTT의 data flow graph

길이가 N인 다항식 곱셈을 수행하는 NTT의 기본 연산 구조는 FFT (Fast Fourier Transform)와 유사하여 NTT의 data flow graph는 그림 1과 같이 나타낼 수 있으며, N개의 데이터를 NTT 연산 할 때 필요한 스테이지의 수는 $n = \log_2 N$이고, 각 스테이지에 필요한 PE (processing element)의 수는 $N/2$이다. NTT 하드웨어의 완전 병렬 구조는

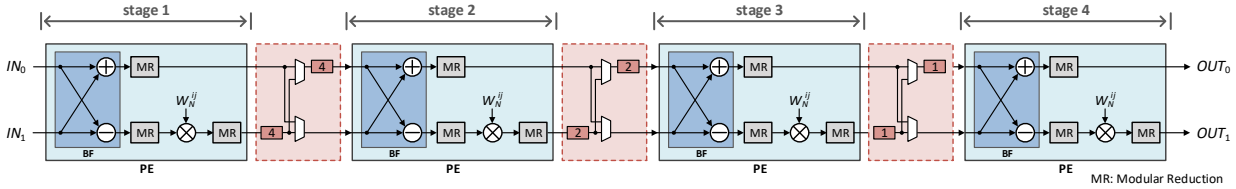

그림 2. N = 16, P = 4인 경우 제안하는 파이프라인 방식의 NTT 하드웨어 구조

latency가 짧으나 하드웨어 복잡도가 매우 높아 비효율적이다. 완전 병렬 구조의 단점을 보완하기 위해 부분 병렬화를 적용한 메모리 기반의 NTT 하드웨어 구조가 제안되었다 [3],[4].

메모리 기반의 NTT 하드웨어 구조 [3],[4]는 NTT 연산에 필요한 PE를 일부만 구현하여 하나의 스테이지에 필요한 연산을 모두 수행하고, 그 과정을 스테이지 별로 반복한다. 이때, 구현한 PE의 수 P ($1 \le P \le N/2$)는 하나의 스테이지에서 동시에 처리하고자 하는 데이터의 수에 따라 결정된다. 이 경우, 입력데이터와 그 다음 연산을 위해 현재의 연산 결과를 저장할 메모리가 추가적으로 필요하다.

메모리 기반의 구조는 이전 데이터에 대한 연산이 모두 끝나야 새로운 데이터에 대한 연산이 가능하기 때문에 실시간 데이터에 대해서 연속적인 연산을 수행할 수 없다. 게다가 NTT의 특성 상 그림 1에 나타난 것과 같이 각 스테이지 별 PE에 입력되는 데이터 쌍이 다르기 때문에 메모리 접근 시 충돌이 발생하지 않도록 해야 한다. 그 결과, 메모리 기반의 구조는 메모리 접근 시 충돌을 방지하기 위한 추가적인 메모리 요소가 필요하고 그로 인해 latency가 길어진다. 본 논문에서는 메모리 기반 NTT 하드웨어 구조의 단점을 보완하기 위한 파이프라인 방식의 NTT 하드웨어 구조를 제안한다.

II. 파이프라인 방식의 NTT 하드웨어

제안하는 파이프라인 방식의 NTT 하드웨어 구조는 메모리 기반의 구조와 마찬가지로 부분 병렬을 적용한 NTT의 하드웨어 구조이다. 파이프라인 방식의 NTT 하드웨어는 각 스테이지 사이에 파이프라인 레지스터를 삽입하여 스테이지별 독립적인 동작이 가능한 구조이다.

제안하는 구조에서 파이프라인 레지스터는 파이프라인 구조를 구현하는 것 뿐만 아니라 각 스테이지별로 PE에 입력되어야 하는 데이터의 쌍의 조합을 재구성하는 역할도 수행한다. 그림 1의 스테이지 1과 스테이지 2의 데이터 쌍을 보면, 스테이지 1의 PE에 입력되는 데이터 쌍은 (0, 8), (1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15)인데, 스테이지 2의 PE에 입력되는 데이터 쌍은 (0, 4), (1, 5), (2, 6), (3, 7), (8, 12), (9, 13), (10, 14), (11, 15)로 입력 데이터 쌍이 달라진다. 이때, 파이프라인 레지스터와 MUX (multiplexer)를 이용해서 다음 스테이지에 필요한 데이터 쌍을 재조합 하는 것이 가능하다.

메모리 기반의 하드웨어 구조 [3],[4]와 비교하기 위해 파이프라인 기반의 하드웨어 구조에서도 P개의 PE를 사용했다고 가정했다. 이 경우 제안하는 구조의 스테이지 하나에 구성되는 PE의 수는 $P/\log_2 N$ 이다. 그림 2는 제안하는 파이프라인 방식의 NTT 하드웨어 구조를 N이 16이고, P가 4인 경우에 대해 나타낸 것이며, 붉은색 점선으로 표시된 부분은 파이프라인 레지스터와 MUX를 활용한 데이터 쌍 재조합 회로이다. 파이프라인 방식의 하드웨어 구조는 하나의 스테이지를 구성하는 PE의 수가 증가할 수록, 전체 구조에 필요한 파이프라인 레지스터의 수가 줄어들며, 각 스테이지에서 필요한 파이프라인 레지스터의 수는 각 스테이지의 PE 수에 따라 $(P/\log_2 N)2^{n-s+1}$로 결정된다. 다시 말해, 하나의 스테이지에 포함된 PE의 수 $P/\log_2 N$ 가 2 이상이라면, 각 스테이지에서 필요한 레지스터의 수는 2의 승수로 줄어들고, 후반부의 스테이지에서는 파이프라인 레지스터 없이 구현하는 것이 가능하다.

제안하는 하드웨어는 각 스테이지의 연산 결과를 파이프라인 레지스터에 저장하기 때문에 모든 스테이지는 독립적인 연산이 가능하다. 따라서 실시간 데이터 입력에 대해 연속적인 연산이 가능하다. 즉, 후반부의 스테이지에서 이전 입력에 대한 연산을 수행하고 있을 때, 전반부의 스테이지에서 새로운 입력에 대한 연산을 동시에 수행하는 것이 가능하다.

III. 구현 결과

메모리 기반 NTT 하드웨어 구조를 제안한 [3]과 [4]의 연구와 제안하는 파이프라인 기반 NTT 하드웨어 구조에 필요한 하드웨어 요소를 N과 PE의 수 P를 이용해서 표 1에 비교하였다.

표 1. 하드웨어 요소 및 latency 비교

Architecture	Memory-based		Proposed
	[3]	[4]	
PE	P	P	P
Memory	$2N+P$	$2N$	$N-2P/\log_2 N$
Latency	$(2N+P)(\log_2 N)$	$2N(\log_2 N)$	$N/(P\log_2 N)-1$

표 1의 메모리의 경우, 메모리 기반의 구조 [3],[4]은 각 스테이지별 연산 결과를 저장하기 위한 메모리와 입력 데이터를 저장하기 위한 메모리, 메모리 충돌 방지를 위한 추가 메모리를 요구하므로 [3]은 $2N+P$, [4]은 $2N$의 메모리가 필요하다. 제안하는 파이프라인 방식은 각 스테이지별 파이프 라인 레지스터만 요구하기 때문에 모든 스테이지의 레지스터의 합인 $N-2P/\log_2 N$ 메모리만 사용하여 구현이 가능하다.

표 1의 latency는 N개의 데이터가 한번 입력될 때 필요한 latency이다. 메모리 기반의 구조로 실시간 NTT 연산을 하려면, NTT 연산에 필요한 latency에 입력데이터를 위한 시간이 추가로 필요하다. 따라서 메모리 기반의 구조 [3],[4]의 N개의 데이터가 M번 연속적으로 입력될 때의 latency는 표 1에 나타난 latency의 M배이다. 반면, 제안하는 알고리즘은 입력에 대해 연속적인 연산이 가능하기 때문에 총 M번 입력되는 데이터를 처리하기 위한 latency는 $(M-1)(N\log_2 N/2P)+2(N\log_2 N/2P-1)$이다. 따라서 실시간 입력을 처리하는 경우에 제안하는 파이프라인 방식의 구조가 메모리 기반의 구조 [3],[4]보다 전체 실시간 데이터를 빠르게 처리할 수 있다.

그림 3은 N이 256이고 N개의 데이터가 한 번만 입력된다고 가정했을 때, P에 따라 latency가 달라지는 정도에 대해서 메모리 기반의 하드웨어 구조 [3],[4]와 제안하는 구조의 latency를 비교한 그래프이다. 그림 3에서 보이는 것과 같이, 메모리 기반의 구조 [3],[4]는 실시간 입력에 대해 연속적인 연산이 불가능하기 때문에 매 스테이지마다 $2N+P$ [3] 혹은 $2N$[4]의 latency가 필요해 latency가 매우 길다. 그러나 제안하는 파이프라인 방식의 구조는 실시간 입력에 대해 연속적인 연산이 가능하므로 기존의 메모리 기반의 구조에 비해 매우 적은 latency만으로 동일한 연산을 수행할 수 있다.

Ⅳ. 결론

본 논문에서는 실시간 입력 데이터를 연속적으로 연산이 가능한 파이프라인 방식의 NTT 하드웨어 구조를 제안하였다. PE의 수가 동일한 경우, 제안하는

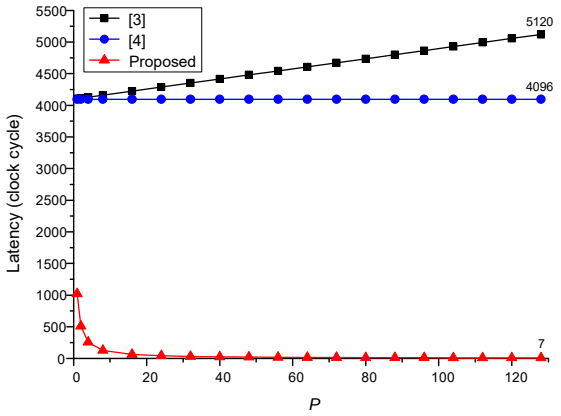

그림 3. $N=256$인 경우 latency 비교

구조가 메모리 기반 구조 대비 적은 메모리 수를 필요로 한다. 또한, 따라서 실시간으로 NTT 연산을 수행해야 하는 경우에는 제안하는 파이프라인 방식의 하드웨어 구조를 사용하는 것이 latency 측면에서 메모리 구조보다 효율적이며, 하드웨어 복잡도 측면에서 완전 병렬 구조보다 효율적이다.

참고문헌

[1] T. N. Tan and H. Lee, "High-Secure Low-Latency Ring-LWE Cryptography Scheme for Biomedical Images Storing and Transmitting," in Proc. 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1-4.

[2] T. Shimada and M. Ikeda, "High-Throughput Polynomial Multiplier Architecture for Lattice-Based Cryptography," in Proc. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5.

[3] A. C. Mert, E. Karabulut, E. Ozturk, E. Savaş, M. Becchi, and A. Aysu, "A flflexible and scalable ntt hardware: applications from homomorphically encrypted deep learning to post-quantum cryptography," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2020, pp. 346-351.

[4] W. Tan, B. M. Case, G. Hu, S. Gao, and Y. Lao, "An ultra-highly parallel polynomial multiplier for the bootstrapping algorithm in a fully homomorphic encryption scheme," *Journal of Signal Processing Systems*, vol. 93, no. 6, pp. 643-656, 2021.