해시기반 양자내성암호의 효과적인 하드웨어 설계를 위한 모듈화 가속 기법 연구

이용석 [1], 이윤지 [1], 백윤흥 [1]
[1] 서울대학교 전기정보공학부, 서울대학교 반도체 공동연구소
yslee@sor.snu.ac.kr, yjlee@sor.snu.ac.kr, ypaek@snu.ac.kr

A Study on Modular Acceleration of Hash-based Post Quantum Cryptography for Efficient Hardware Design

Yongseok Lee[1], Yunji Lee[1], Yunheung Paek[1]
[1]Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center(ISRC), Seoul National University

요 약

양자 컴퓨터 시스템에서도 기존 컴퓨터 시스템처럼 보안성을 보장하는 암호체계를 양자내성암호(PQC, Post Quantum Cryptography)라 한다. NIST 에서 표준화로 선정된 네 가지 알고리즘 중 SPHINCS+ 알고리즘은 해시 연산을 반복적으로 수행하는 해시기반 알고리즘이다. 이에 따라 해시기반 알고리즘에서 반복적으로 사용되며 많은 연산 시간을 구성하는 해시 연산을 효과적으로 구현하려는 연구들이 진행되고 있으며, 그 중에는 RISC-V 를 통한 하드웨어/소프트웨어 통합 설계 연구 그리고 FPGA 혹은 ASIC 을 이용한 하드웨어 통합 모듈 가속 연구들이 있다. 본 논문에서는 해시기반 양자내성암호 알고리즘의 효과적인 하드웨어 설계를 위해 해시 연산 모듈러 가속 방법을 분석하고, 이를 하드웨어에서 효과적으로 구현한 기존 연구 사례를 분석한다. 이를 통해 해시 연산의 모듈러 가속 기법 연구를 위해 고려되는 설계면적 그리고 연산 성능 등의 수치를 높이는 과정에서 발생하는 트레이드 오프와 이를 효과적으로 극복하는 방법들을 소개한다.

1. 서론

최근 양자컴퓨터가 개발됨에 따라, 양자 컴퓨터 시스템에서도 안전한 데이터 전송 및 인증을 위해 양자내성암호(PQC, Post Quantum Cryptography) 알고리즘은 해시기반, 다변수기반, 코드기반, 격자기반 등 다각도로 개발되고 있다[1]. 이러한 양자내성암호 알고리즘은 양자컴퓨터에서 사용되는 것은 물론 양자내성 기준을 요구하는 기존의 컴퓨터 시스템에서도 사용될 수 있다. 이러한 흐름에 맞춰 미국 NIST(National Institute of Standard and Technology)에서는 2017 년부터 현재까지 양자내성암호에 대한 표준화를 진행하고, 네 가지 기반을 바탕으로 알고리즘들을 분류하고 있다[2]. 양자내성암호 알고리즘 타입은 크게 두 분야로 표준화를 진행하였다. 그것은 KEA(Key Encapsulation Algorithm) 그리고 DSA(Digital Signature Algorithm) 분야이다. 두 분야에서 표준화 알고리즘으로 선정된 네 가지 알고리즘은 KEA 분야에서 Crystal-Kyber[3], 그리고 DSA 분야에서 Crystal-Dilithium[4], FALCON[5], SPHINCS+[6]를 선정하였다. 이 중 SPHINCS+ 알고리즘은 해시(Hash)기반으로 보안 파라미터에 따라서 SHA-3(SHAKE256) 혹은 SHA-2(SHA256, SHA512) 해시 함수를 사용한다. 또한 알고리즘을 통한 키 생성, 서명 생성 그리고 검증 과정 대부분의 연산 시간을 해시 함수를 반복해서 연산하는데 사용한다.

이런 해시기반 알고리즘의 특징으로 해시 연산을 하드웨어 모듈화하여 가속하려는 연구들이 수행되고 있다. 해시 연산은 비트단위 연산의 특징으로 소프트웨어의 워드 단위로 연산을 수행하는 것보다 하드웨어로 비트 연산을 구현하는 것이 유리한 특징이 있으며, 그 연산들도 병렬화와 파이프라인 설계가 가능한 하드웨어 친화적인 연산 구조를 가지고 있다.

따라서 다양한 해시 연산 모듈과 인터페이스로 연결되어 설계 유연성을 가질 수 있는 것을 목표로 하는 연구들이 진행되고 있으며, [7] 연구는 RISC-V 라는 임베디드 환경에서 하드웨어/소프트웨어 통합 설계로 컨트롤 파트는 RISC-V 코어가 수행하고, 연산 시간이

오래 걸리는 해시 연산을 하드웨어 모듈화하여 SPHINCS+ 알고리즘을 수행할 수 있는 방법을 제안하였다. 이는 해시기반 알고리즘의 다양한 해시 연산을 효과적으로 가속하는 방법으로 기존 소프트웨어(Cortex M4) 단독 수행 대비 SHAKE256 버전에서는 약 100~300 배 빠른 연산 속도를 보여주었고, SHA-2 버전에서는 약 10~40 배 빠른 연산 속도를 보여주었다. 여기서 더 심층적인 방식으로 [8] 연구는 RISC-V 코어와 해시 연산 하드웨어 모듈이 연결될 수 있는 여러가지 인터페이스 환경을 고려한 설계 방법을 제안하였다. 이는 해시 연산 RISC-V 코어와 어떻게 연결되는지에 따라서 SPHINCS+ 알고리즘의 성능에서 성능 차이가 발생할 수 있음을 보여주었다.

이처럼 해시기반 알고리즘들은 서로 다른 보안 파라미터를 사용하더라도 공통되는 해시 연산을 반복적으로 수행하는 특징이 존재하며, 이 해시 연산 모듈을 최대한 가속하려는 연구와 설계면적 대비 효율적으로 설계하여 최적의 성능을 보이려는 연구들이 진행되었다. 본 논문에서는 해시 연산 모듈화를 통해 효과적인 하드웨어 설계를 위한 가속 기법들을 중심으로 분석하려 한다. 먼저 해시기반 양자내성암호 알고리즘이 가지는 해시연산들의 특징을 분석하고, 이를 효과적으로 모듈화하는 과정에서 발생하는 문제점 그리고 기존 연구에서 극복한 기법들을 알고리즘이 사용되는 목적에 맞게 다양한 성능 요구사항 측면에서 분석한다. 이를 통해 해시기반 양자내성암호 알고리즘들에 대해 효과적인 모듈화 가속 설계하는 방법을 제안한다.

2. 해시(Hash)기반 양자내성암호 알고리즘 특징

양자내성암호 중 해시기반 알고리즘은 그림 1 과 같이 반복적인 해시 함수를 사용하며, 해시 함수는 단방향성 특징을 가지고 있다. 이러한 해시 함수는 표준화에 따라 SHA-3(Secure Hash Algorithm 3) 그리고 SHA-2(Secure Hash Algorithm 2)로 그 방식을 나눌 수 있다. 하지만 표 1 에서 볼 수 있듯이, 해시 기반 양자내성암호 알고리즘에서는 보안 파라미터에 따라서 해시 함수 표준화 레벨을 분류하는 것이 아니라, 어떤 해시 함수 표준화 방식을 사용하더라도, 다양한 보안 파라미터를 만족할 수 있도록 양자내성암호 알고리즘을 구성하였다. 즉, 사용자가 사용할 수 있는 해시 함수 표준화 기법에 따라서 그에 맞는 해시기반 암호 알고리즘을 적용할 수 있는 것이다. 이는 해시 연산을 모듈러하여 알고리즘의 컨트롤 플로우와 분리하여 구현할 수 있는 특징을 만들어 준다.

해시 연산을 모듈러하는 방법은 암호 알고리즘에서 해시 연산이 사용되는 부분을 특정 해시 모듈을 호출하고, 그 연산 결과를 돌려받는 식으로 진행 될 수 있다. 이러한 방식은 그 해시 모듈이 어떠한 연산 성능을 가지고 있는지 혹은 어떤 해시 모듈인지에 상관없이 암호 알고리즘의 컨트롤 플로우를 유지할 수 있다는 특징이 있다. 물론 현재 해시기반 양자내성암호 알고리즘에서 해시 함수 표준화 기법이 바뀐다면 입력 데이터를 패딩하는 방식 등이 약간 다르게 적용되지만, 큰 컨트롤 플로우 관전에서는 동일한 방식을 유지한다.

모듈화된 해시 연산은 그 표준화 기법의 차이뿐만 아니라, 연산 성능에 있어서도 차이를 가질 수 있다. SHA-3 기반의 SHAKE256 해시 함수의 경우 내부적인 연산으로 슬라이스 연산이 존재하며, 해시 모듈을 설계하는 방법에 있어서 슬라이스 연산 병렬화 정도에 따라 연산 사이클 감소와 하드웨어 자원과의 trade-off 가 가능하며, 중간 데이터를 저장하는 레지스터를 할당하는 개수에 따라서 연산 스텝들의 파이프라인 유무를 설정할 수 있다.

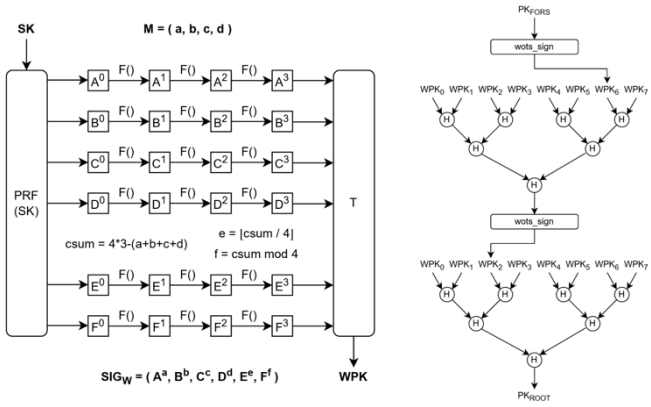

그림 1. 해시기반 암호알고리즘(SPHINCS+)에서 WOTS+ 알고리즘 및 XMSS 알고리즘 구조 예시[7]. (M: Message Digest, SK: Secret Key, WPK: WOTS+ Public Key, SIG_W: WOTS+ Signature, PK_{FORS}: FORS Public Key, PK_{ROOT}: XMSS Public Key)

표 1. 해시기반 양자내성암호 알고리즘 SPHINCS+의 해시 타입, 파라미터별 보안 레벨, 해시 출력 데이터 길이(**n**)[6].

Algorithm	Hash Type	Security Level	Parameters	n (byte)
SPHINCS+	SHAKE256	Level 1	128s, 128f	16
		Level 2	192s, 192f	24
		Level 3	256s, 256f	32
	SHA256	Level 1	128s, 128f	16
	SHA256 & SHA512	Level 2	192s, 192f	24
		Level 3	256s, 256f	32

그림 1 에서 볼 수 있듯이, SPHINCS+ 알고리즘은 대부분의 연산이 해시 연산을 반복적으로 수행하면서 진행된다. 예를 들어 WOTS+과정의 **PRF** 연산은 입력된 서명의 SK(Secret Key)데이터를 해시 연산을 통해 WOTS+의 SK 를 생성하는 과정이다. SPHINCS+ 디지털 서명 알고리즘은 WOTS+ 서명 알고리즘과 XMSS 서명 알고리즘 그리고 FORS 알고리즘이 합쳐져서 만들어진 구조를 가지고 있기 때문에, 서명 생성을 위한 SK, PK(Public Key)가 반복적으로 등장한다. 또한 WOTS+의 체인 과정이라 불리는 **F** 연산과 생성된 WOTS+의 PK 를 압축하는 과정인 **T** 연산도 모두 해시 연산을 수행함으로써 진행된다. 오른쪽 그림의 XMSS 과정 또한 **H** 연산은 자식 노드의 데이터를 해시 연산하여 부모 노드의 데이터로 만드는 과정으로, 역시 해시 연산이 반복되는 특징을 보이는 것을 알 수 있다.

3. 해시 연산 하드웨어 모듈화 가속 연구

해시기반 양자내성암호 SPHINCS+ 알고리즘의 특징들은 2 장에서 살펴본 것처럼 **PRF**, **F**, **T**, **H** 연산 같은 이러한 해시 연산이 반복적으로 사용되는 알고리즘의 연산 특징들로 인해 해시 연산 모듈을 하드웨어로써 별도로 구성하고, 이를 가속하는 최적화 연구들[7,8]이 주로 진행되고 있으며, 각각 연구들의 특징을 분석하려 한다.

먼저, 그림 2 에서 볼 수 있듯이 [7] 연구에서는 RISC-V 코어와 해시 연산 하드웨어 모듈을 구성하여 소프트웨어/하드웨어 통합 환경에서 연구를 수행했다. 이는 RISC-V 코어에서 SPHINCS+ 알고리즘에 대한 컨트롤 플로우를 모두 수행하면서, 해시 연산이 필요한 순간마다 해시 연산 하드웨어 모듈을 호출하여 연산을 수행하는 방식이다. 이러한 방식은 2 장에서 살펴본 것처럼 두 가지 이유가 있는데, 첫 번째는 해시 기반 양자내성암호 알고리즘의 특성상 해시 연산이 계속적으로 반복 호출되는 특성이다. 그리고 두 번째는 하드웨어 친화적인 연산 구조를 가지고 있는 해시 연산의 특징으로 소프트웨어에서 연산할 때 보다 더 효과적인 성능 향상을 보일 수 있기 때문이다.

이러한 특징들로 인해 해시 연산 모듈은 하드웨어로 설계되어 인터페이스로 컨트롤 파트와 통신하는 구조를 가지고 있다. 또한, SPHINCS+ 알고리즘은 하나의 해시 연산을 사용하는 것이 아니라, 파라미터에 따라서 그리고 사용자의 선택에 따라서 SHAKE256 연산 혹은 SHA256 연산 아니면 SHA256 & SHA512 연산을 같이 사용하기도 한다. 다양한 해시 연산들을 지원하기 위해 세 가지 해시 연산을 모두 모듈화하여 인터페이스로 연결한 특징을 확인할 수 있다.

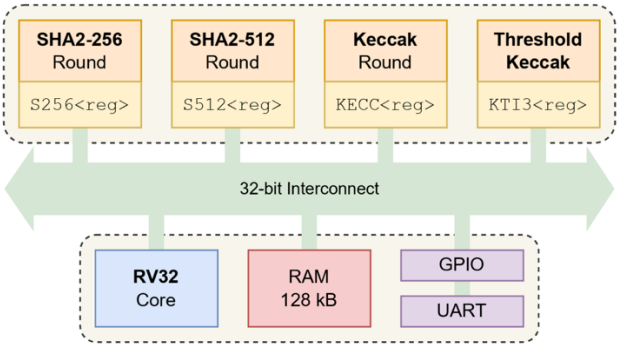

그림 2. RISC-V 코어와 해시 연산 하드웨어 모듈의 인터페이스 구조 예시[7]. (RV32: RISC-V Processor, Keccak: SHAKE256.)

그림 3 에서 인터페이스 구조 예시를 보여주고 있는 연구[8]에서는 RISC-V 코어와 해시 연산 하드웨어 모듈을 인터페이스로 연결하는 방식에서 좀 더 심층적인 연구를 수행하였는데, [7] 연구에서는 RISC-V 를 인터컨넥트를 통한 하드웨어/소프트웨어 통합 설계 방법을 적용하였고, [8] 연구는 하드웨어 모듈이 RISC-V 코어 내부에 연결된 방식의 설계 혹은 AXI4 인터페이스를 통해 연결되더라도, SRAM 이나 FIFO 를 통해 연결될 경우 성능이 어떻게 달라지는지 측정하는 연구를 수행하였다. 이에 [7] 연구에서는 RISC-V 에 SHA-3 해시 모듈을 통합할 경우 설계면적은 5,582 LUTs 가 RISC-V 와 별도로 추가된다는 하나의 경우를 보여주지만, 이와 다르게 다양한 방식의 모듈러 연결을 보여준 [8] 연구에서는 연결 방식에 따라서 같은 SHA-3 해시 연산 모듈을 추가할 경우라도 4,494 LUTs 에서부터 8,731 LUTs 가 추가되는 여러가지 설계 옵션을 제공하는 특징이 있다.

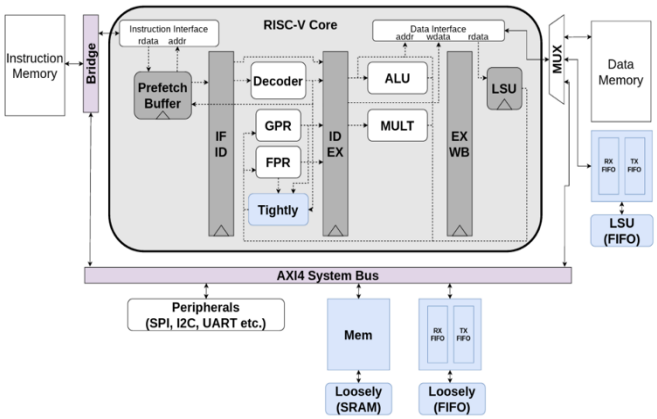

그림 3. 다양한 인터페이스 환경에서 RISC-V 코어와 해시 연산 모듈 가속 설계 기법 예시[8]. (파란색 블럭: 해시 연산 가속 모듈이 올 수 있는 네 가지 경우(Tightly, Loosely(SRAM or FIFO), LSU))

이처럼 해시기반 알고리즘의 특성으로 인해 해시 연산을 하드웨어 모듈화하여 가속하려는 연구들이 진행되고 있으며, 연결 방식에 있어서도 설계면적이 달라지는 등 성능 차이를 발생시키는 것을 알 수 있다.

양자내성암호는 사용자 인증이 필요한 디지털 서명 분야에서 다양하게 사용될 수 있는 만큼, 그 플랫폼 또한 다양하다. 따라서 요구되는 설계면적과 성능이 기준이 서로 상이하여 어떠한 인터페이스 기법이 우수하다고 선택하기 어려울 것이다. 그러한 이유로 앞서 살펴본 두 연구의 방식인 RISC-V 와 해시 하드웨어 모듈의 가속화 방법 외에도 RISC-V 대신 하드웨어 컨트롤 모듈을 사용하는 방법 등 다양한 방법이 적용될 수 있다. 하지만 SPHINCS+ 알고리즘의 특성으로 해시 연산 모듈을 하드웨어 모듈화 가속하는 방식은 유지될 것으로 보여진다.

4. 결론

본 논문에서는 해시기반의 양자내성암호 알고리즘인 SPHINCS+의 연산 특징을 알아보고, 해시 연산을 하드웨어로 모듈화 가속하는 연구들에 대해 각각 분석하였다. 이는 기존 알고리즘의 해시 연산 특성을 분석하고 이에 맞는 하드웨어 가속 기법을 적용하였다. 또한 그러면서 발생하는 인터페이스가 사용 환경에 맞는 새로운 고려사항으로 작용하였고, 이를 분석하기 위해 모듈과의 인터페이스 경우들을 분석하여 연산 성능과 설계면적의 밸런스를 고려한 모듈러 설계 방법에 대해서도 분석하였다. 이러한 설계 방법은 RISC-V 와 통합 구현이 아니더라도 독립적으로 분리된 해시 하드웨어 모듈화로 인해 해시 기반 알고리즘을 사용할 경우 고려해야 하는 문제점들을 잘 보여주고 있으며, 향후 양자내성암호 알고리즘의 하드웨어 연구에 좋은 기여가 될 것으로 기대된다.

ACKNOWLEDGEMENT

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구이며 (IITP-2023-RS-2023-00256081), 2025 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구이며 (RS-2023-00277326), 2025 년도 정부(산업통상자원부)의 재원으로 한국산업기술기획평가원의 지원을 받아 수행된 연구이며(No. RS-2024-00406121, 자동차보안취약점기반 위협분석시스템개발(R&D)), 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구이며(No.RS-2024-00438729, 익명화된 기밀실행을 이용한 전주기적 데이터 프라이버시 보호 기술 개발), 2025 년도 BK21 FOUR 정보기술 미래인재 교육연구단에 의하여 지원되었음.

참고문헌

[1] Nejatollahi, Hamid, et al. "Post-quantum lattice-based cryptography implementations: A survey." *ACM Computing Surveys (CSUR)* 51.6 (2019): 1-41.

[2] Asif, Rameez. "Post-quantum cryptosystems for Internet-of-Things: A survey on lattice-based algorithms." *IoT* 2.1 (2021): 71-91.

[3] Avanzi, R. et al. CRYSTALSKyber: Algorithm Specifications and Supporting Documentation, Submission to the NIST Post-Quantum Project. 2021. Available online: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf

[4] Ducas, L. et al. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation, Submission to the NIST Post-Quantum Project. 2021. Available online: https: //pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[5] Fouque, P.A. et al. Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU, Specification v1.2. 2020. Available online: https: //falcon-sign.info/falcon.pdf

[6] Aumasson, J.P. et al. SPHINCS+ Specification. Submission to the NIST Post-Quantum Project. 2020. Available online: https://sphincs.org/data/sphincs+-r3.1-specification.pdf

[7] Saarinen, M. O. et al. "Accelerating SLH-DSA by Two Orders of Magnitude with a Single Hash Unit." In Annual International Cryptology Conference, pp. 276-304. Cham: Springer Nature Switzerland, 2024.

[8] Karl, P. et al. "The impact of hash primitives and communication overhead for hardware-accelerated SPHINCS+." In International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 221-239. Cham: Springer Nature Switzerland, 2024.