



쉽게 이해할 수 있는

국가보안기술연구소 

PQC 알고리즘 HW 구현 기술

2026.02.26.

이봉수

CONTENTS

- | 01 암호 하드웨어 구현의 이해
- | 02 양자내성 암호의 하드웨어 구현 - NIST PQC
- | 03 양자내성 암호의 하드웨어 구현 - KpqC
- | 04 정리 및 질의 응답

| 01 |

암호 하드웨어 구현의 이해

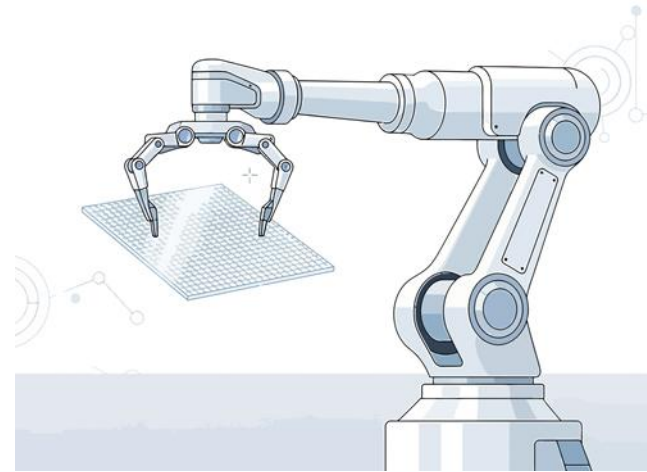


| 암호 구현의 두 가지 방식, 소프트웨어 V.S 하드웨어



소프트웨어

- 다양한 기능을 수행
- CPU가 명령어를 순차적으로 처리
- 기능 수정이 유연함
- Ex. AES연산에 수백 클록 이상*



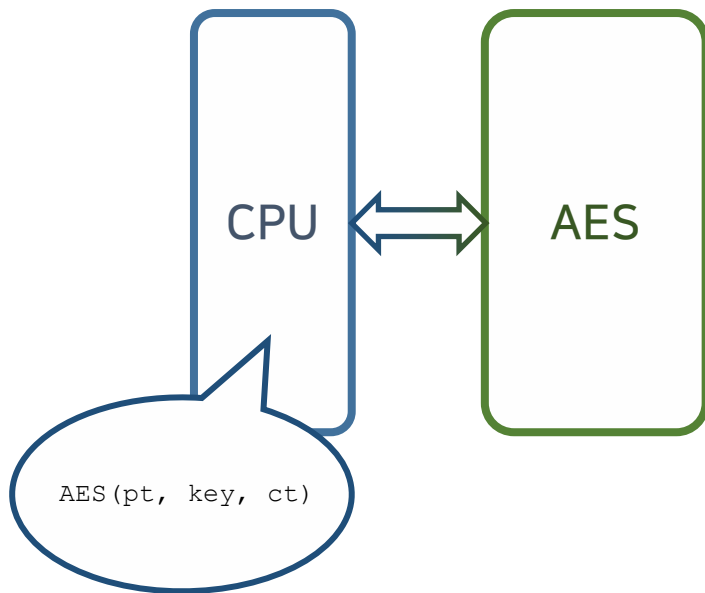
하드웨어

- 지정된 기능을 수행
- 반복 작업에 특화
- 병렬처리 가능
- 기능 수정이 어려움 (ASIC)
- Ex. AES연산에 10클록

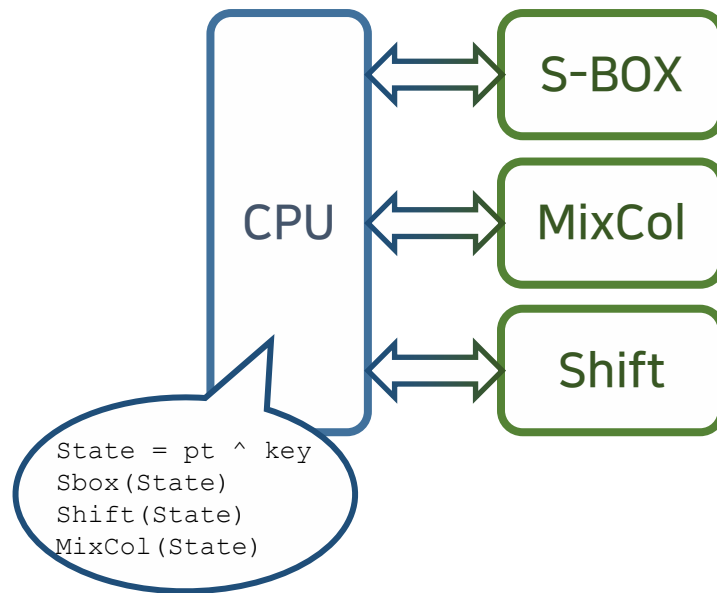
하드웨어 설계는 전용 HW 또는 HW/SW 공동설계(Co-design) 구현

HW/SW 공동설계는 하드웨어와 소프트웨어가 조합된 시스템에서
시스템 성능을 향상시키면서 기능의 유연성을 유지하기 위해
SW 기능의 일부를 HW로 대체하여 성능과 유연성을 모두 만족시키는 설계 방법론

전용 HW



HW/SW Co-design



연산 성능은 전용 HW보다 낮지만 효율성이 높은 장점

ML-KEM 전용 HW 구현 사례

알고리즘	Encap. Cycle(k)	Freq. MHz	AREA		
			LUT	FF	Slice
KYBER512	6.3	161	7,412	4,644	2,126
KYBER768	7.9				
KYBER1024	10.0				

Y. Xing and S. Li, "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPAG", TCHES, Vol. 2021, No. 2, pp. 328-356.

ML-KEM의 HW/SW 공동설계 사례

알고리즘	Encap. Cycle(k)	Freq. MHz	AREA	
			LUT	FF
KYBER512 Software-only	623.3	600	3,087	1,050
KYBER512 HW/SW co-design	151.6			

T. Wang, C. Zhang, X. Zhang, D. Gu and P. Cao, "Optimized Hardware-Software Co-Design for Kyber and Dilithium on RISC-V SoC FPGA", TCHES, Vol. 2024, No. 3, pp. 99-135.

HW/SW 공동설계와 전용 HW 구현 중
어떤 방법이 좋을까 ?



성능

동시에 처리하는 데이터가 많을수록,
클록 주기가 빠를수록,
성능이 향상됨

면적

동시에 처리하는 데이터가 많을수록,
클록 주기가 빠를수록,
(ASIC) 오래된 공정을 사용할수록,
하드웨어의 면적이 커짐

소모전력

클록 주기가 빠를수록,
동시에 처리하는 데이터가 많을수록,
하드웨어의 면적이 클수록,
소모되는 전력이 커짐

$$P = \alpha CV^2 f$$

비용

하드웨어의 면적이 작을수록
수율이 향상됨 (개별 단가 낮아짐)

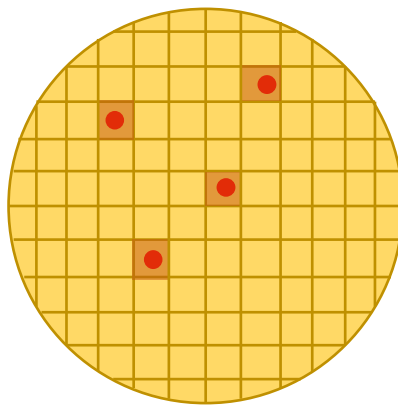
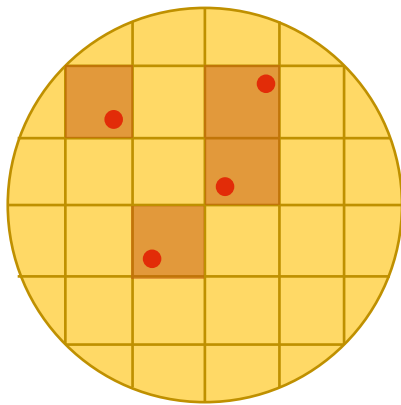


[참고] 수율이란 ?

Wafer 직경이 12인치, 칩 크기가 3mm x 3mm 일 때

$$DPW = \left(\frac{\pi \cdot D^2}{4 \cdot A_{\text{die}}} \right) - \left(\frac{\pi \cdot D}{\sqrt{2} \cdot a} \right) = \left(\frac{\pi \cdot 300^2}{4 \cdot 9} \right) - \left(\frac{\pi \cdot 300}{\sqrt{2} \cdot 3} \right) \approx 7,632$$

수율 100%일 때, 7,632 개의 칩을 수율이 90%일 때, 6,869개의 칩을 생산



칩(die) 크기가 작아지면
제조상에 발생한 오류에도
사용할 수 있는 칩의 수가 늘어남

칩 제작 시 공정이 커지고 하드웨어 면적이 증가함에 따라 die 면적 증가 = 수율 감소

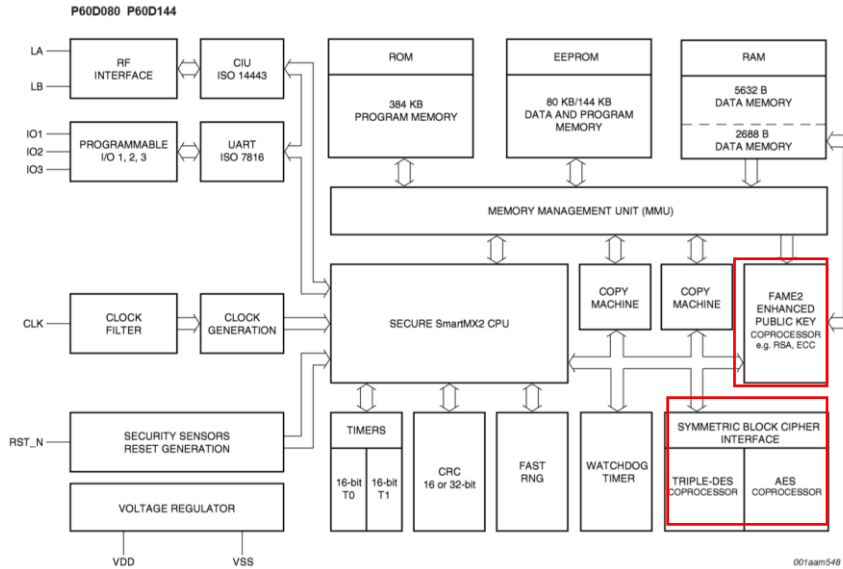
면적
비용
전력

L
O
S
S

G
A
I
N

성능

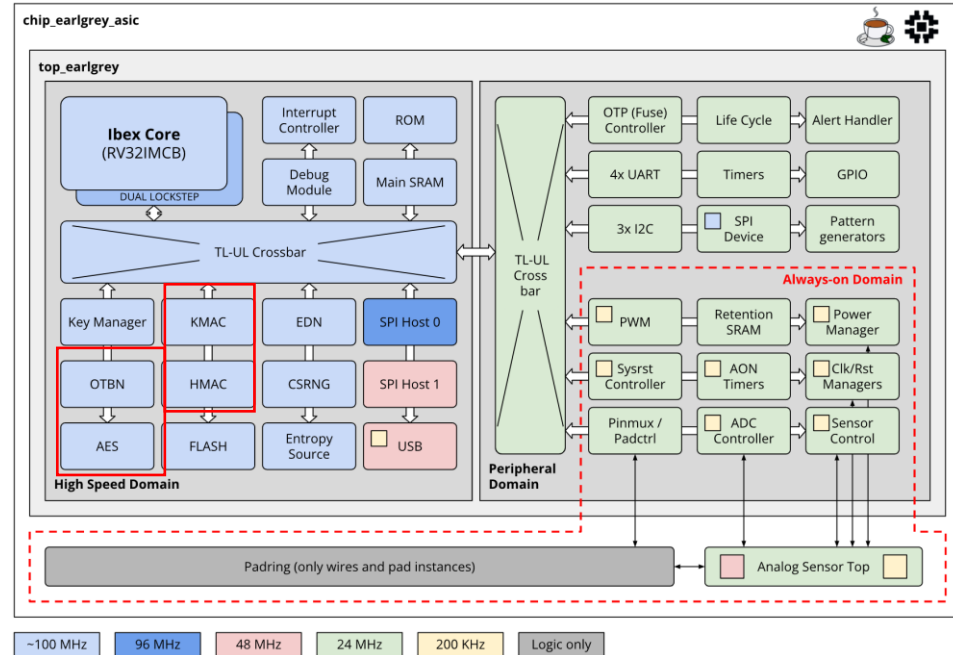




@NXP Secure Smart Card Controller P60D080/052/040yVC(Z/A)/yVG SecurityTarget Lite (Rev. 2.0), 21 August 2014

스마트카드 IC

- 블록암호 AES, T-DES 전용 HW
- 공개키 연산을 위한 부분 가속기



@OpenTitan

OpenTitan

- 블록암호, 해시 전용 HW
- 공개키 연산을 위한 OTBN 블록 지원

Open Titan Big Number 연산기

HW/SW 공동설계와 전용 HW 비교

Point

같은 알고리즘이라도 사용 환경이 다르면 설계 방법도 다르다

HW/SW Co-design

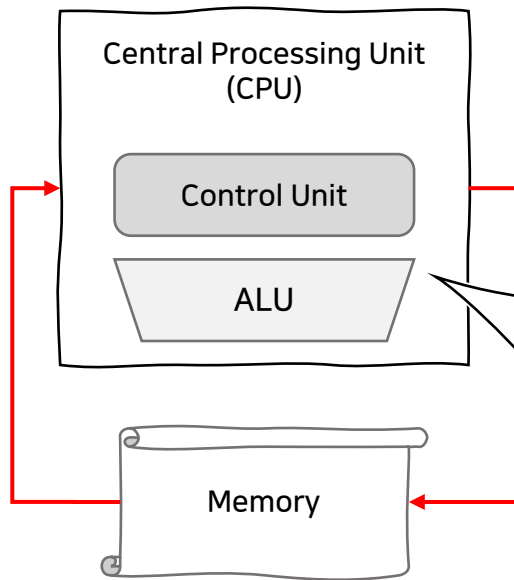
- 전력 요구 사항과 같이 자원이 제한적인 **IoT 환경**에서 제한된 성능으로 연산 양이 많은 계산을 수행해야할 때, **최소한의 HW 사양**으로 **효율적인** 연산이 가능
- 반도체 칩 제조사는 면적을 최소화하여 비용을 줄이고 수율을 높임과 동시에 범용으로 사용할 수 있도록 Co-design을 선택

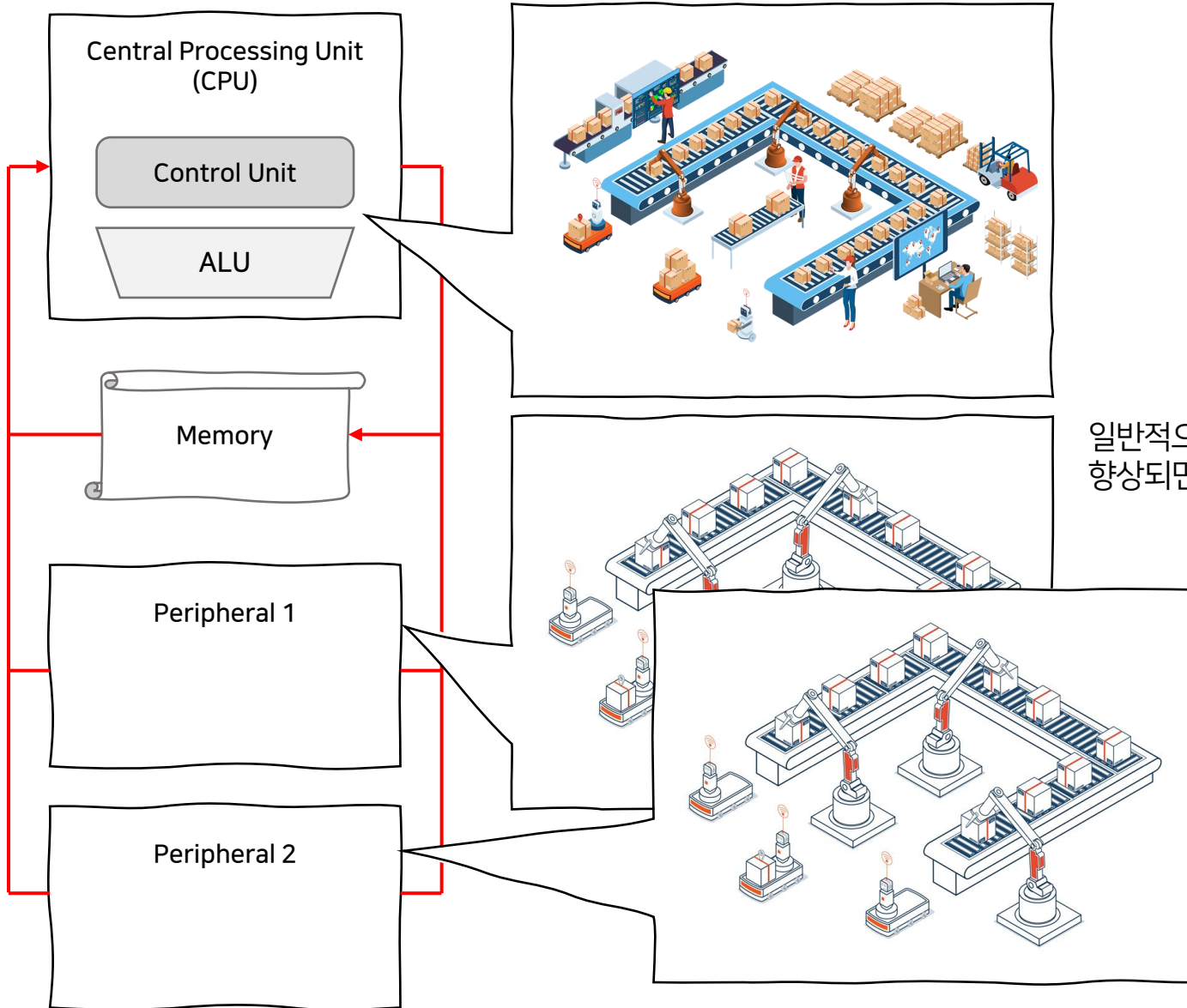
전용 HW

- 블록암호와 같이 크기가 비교적 크지 않고, **사용 빈도가 많을수록** 유리
- 공개키암호의 경우, 서버 환경과 같이 자원이 충분하고 **높은 성능**을 요구하는 환경에서 전용 HW 구현이 유리
- 면적 대비 성능 요구치가 높은 임베디드 환경에서도 충분히 고려할 수 있음

HW/SW 공동설계(Co-design)에서
효율이 생각보다 높지 않은데
연산기를 추가해서 더 빠르게 만들 수 있을까 ?







메모리의 대역폭은 무한하지 않고, 양자내성 암호의 성능은 데이터 이동이 결정함

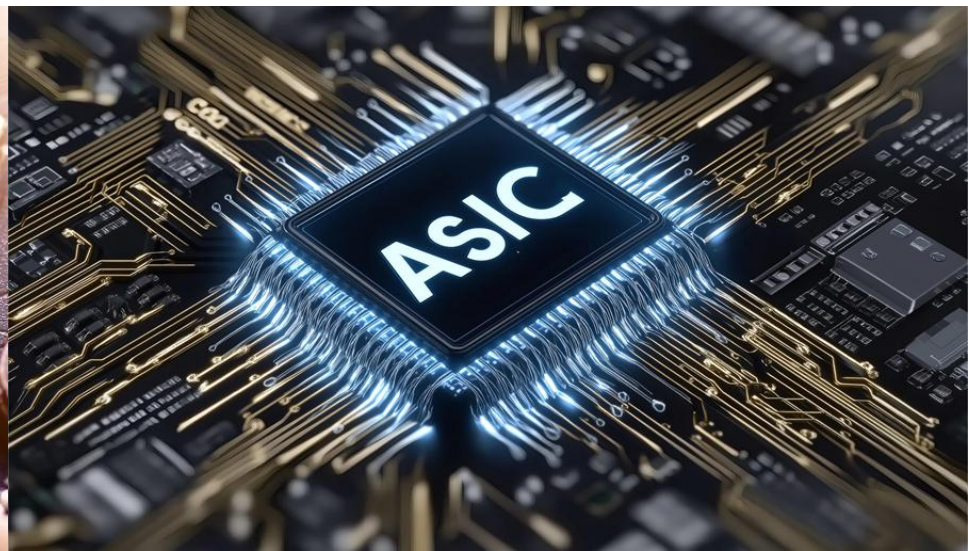


하드웨어 연산기를 빠르게 만드는 것 보다 데이터 이동을 효율적으로 처리하는 것이 중요

ML-KEM의 빠른 NTT는 40클록 사이클*로 구현 가능,
32비트 버스를 사용하는 시스템에서 ML-KEM의 16비트 256개의 데이터를 옮기기 위해
이상적인 경우 최소 256클록(입력 128클록, 출력 128클록)이 소요됨

하드웨어 설계 및 구현 대상은 ?





Field-Programmable Gate Array

- 내부에 수많은 기본 논리 블록을 탑재한 “조립식” 하드웨어로, 사용자가 연결방법을 변경하여 원하는 회로를 구성
- 하드웨어를 구성한 후에도 변경이 가능한 장점
- 지정된 설계 블록만 사용가능, 지정된 용량 이상의 설계는 불가능함
- 성능에 한계가 있음, 개당 단가 높음

Application Specific Integrated Circuit

- 사용자가 원하는 목적을 위해 만들어진 맞춤형 하드웨어
- 반도체 제작 공장을 통해서만 제작이 가능, 제작비용이 비싸고, 한번 제작된 칩은 수정할 수 없음
- 대량생산이 가능하며, 원하는 모든 기능을 탑재할 수 있음

Point

모든 하드웨어는 FPGA에서 태어나고, ASIC으로 완성된다

FPGA는 빨리 만들 수 있지만 효율이 떨어지고,
ASIC은 한번 만들기 어렵지만 만들면 매우 효율적

하드웨어 프로토타입 개발, 연구, 시험, 검증에 **FPGA**가 유리
스마트폰, 자동차 등 전력, 성능, 보안이 중요한 제품에서 **ASIC**이 필요

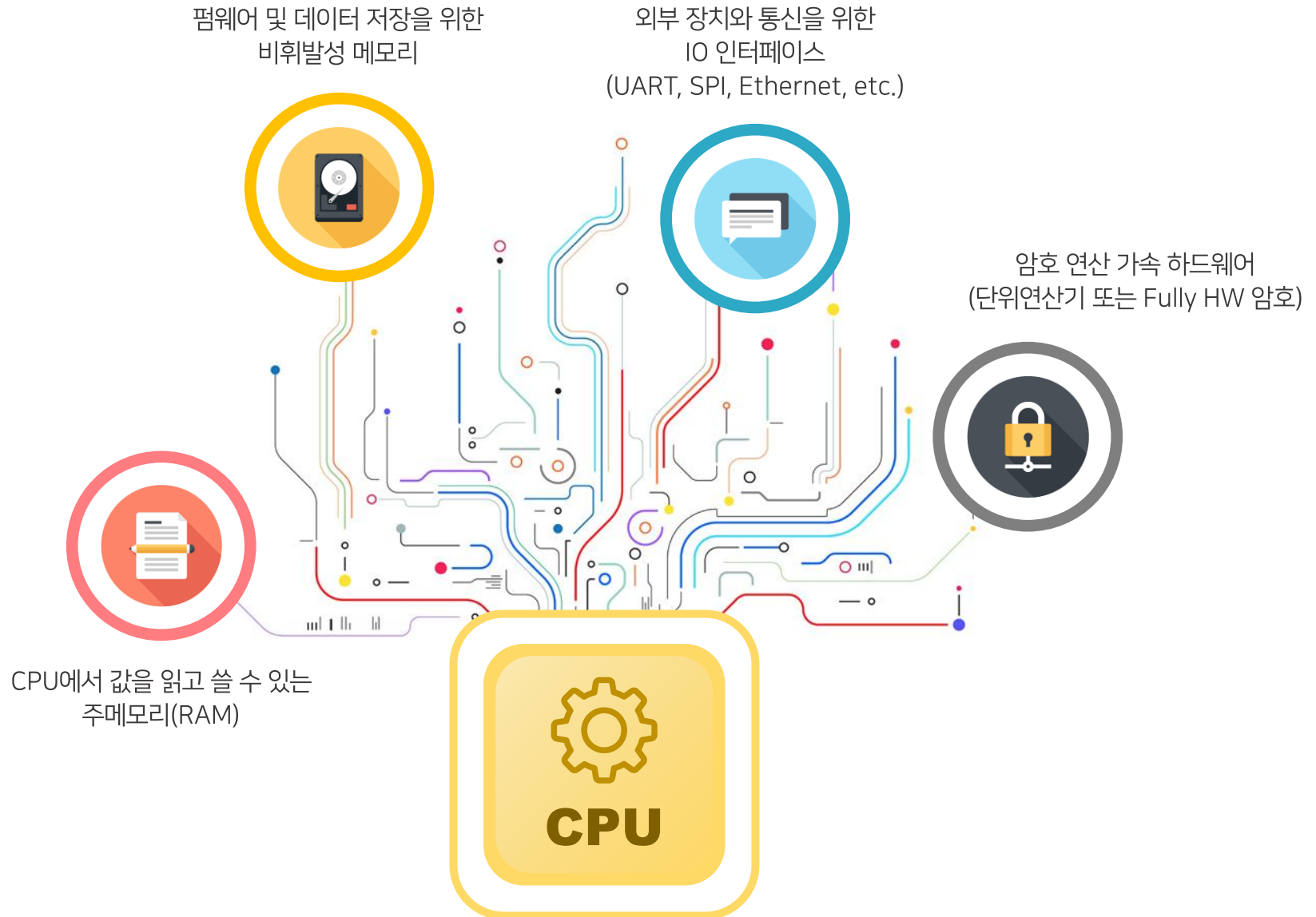


| 02 |

양자내성 암호의 하드웨어 구현

- NIST PQC 사례 -

| 암호(칩) 하드웨어의 구성

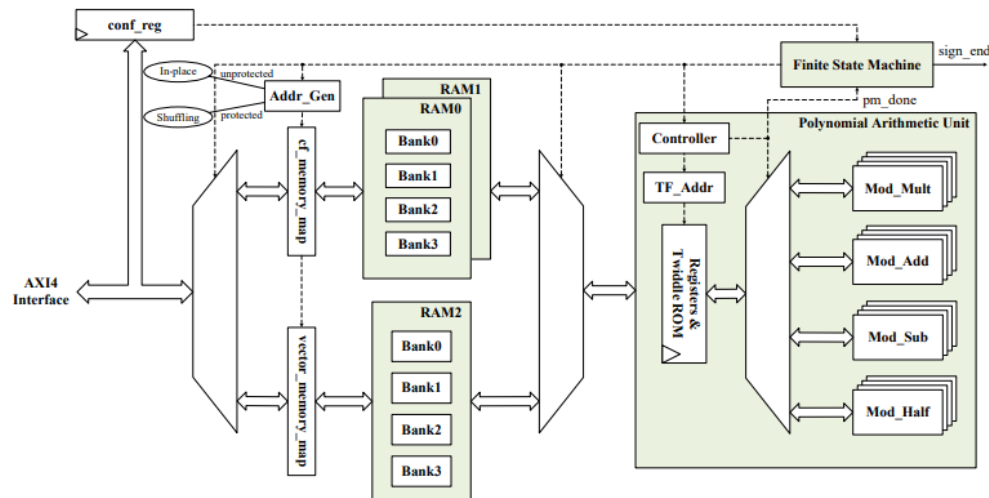


ML-KEM의 HW/SW 공동설계 연구 분석

알고리즘	Encap. Cycle(k)	Freq. MHz	AREA	
			LUT	FF
KYBER512 Software-only	623.3	600	3,087	1,050
KYBER512 HW/SW co-design	151.6			

T. Wang, C. Zhang, X. Zhang, D. Gu and P. Cao, "Optimized Hardware-Software Co-Design for Kyber and Dilithium on RISC-V SoC FPGA", TCHES, Vol. 2024, No. 3, pp. 99-135.

SHA3는 SW 구현, NTT 및 다항식 계산은 하드웨어 연산기 활용



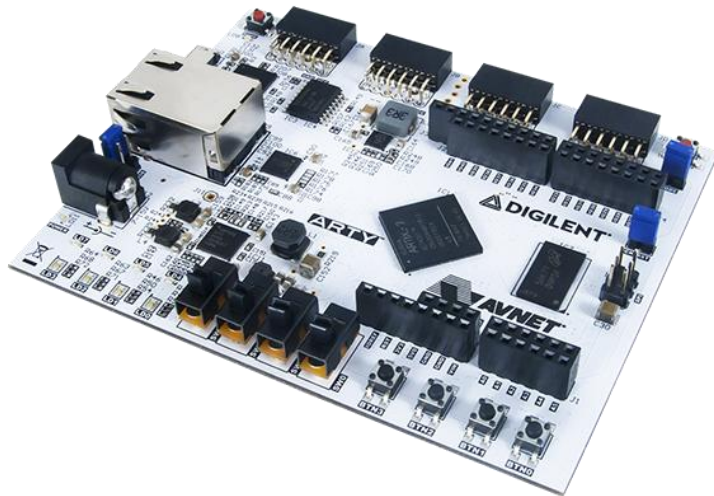
ML-KEM 알고리즘 및 참조구현 분석

연산	Cycles	비율(%)
SHA3-256(random, 32B)	209,616	1.129
SHA3-256(Public-key, 1184B)	1,858,072	10.008
SHA3-512({tmp1, tmp2}, 64B)	214,396	1.155
Unpack public-key, Poly_from_msg	68,924	0.372
Generate Matrix	6,221,942	33.512
Polynomial get noise	1,739,794	9.371
NTT	1,395,532	7.517
Polynomial Multiplication	2,355,000	12.684
INTT	1,958,736	10.55
Polynomial Addition & Reduction	294,341	1.586
Pack Ciphertext	186,266	1.003
SHA3-256(Ciphertext, 1088B)	1,847,560	9.951
SS:=SHAKE256	215,967	1.163
RISC-V 32bit Core, Encapsulation 기준	18,566,146	100

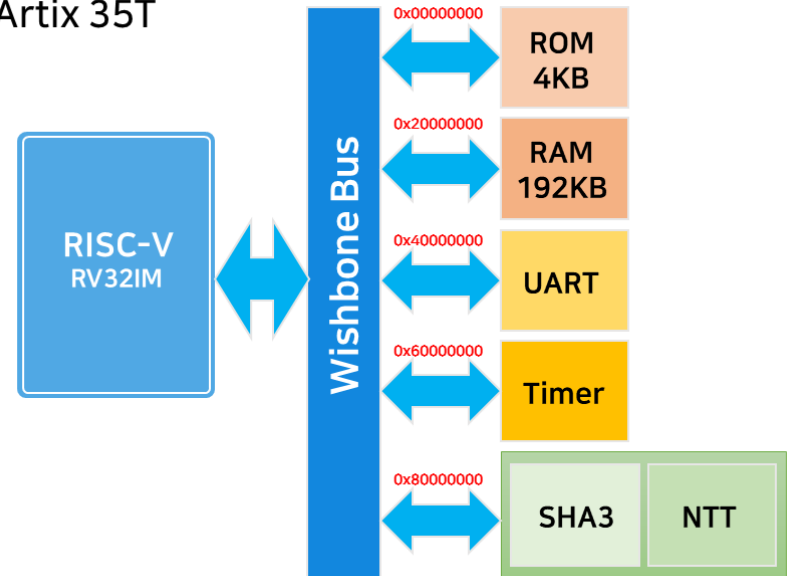
약 56.9%가
SHA3 관련 연산

약 18%가
NTT 관련 연산

벤치마크 플랫폼



Artix 35T



- 32-bit RISC-V CPU, 곱셈/나눗셈기 포함
- 4KB ROM 및 192KB RAM 사용,
알고리즘을 컴파일 하여 펌웨어 업로드 가능하도록 설계
- Timer를 사용하여 알고리즘 동작시간 측정 가능
- UART를 통해 펌웨어 업로드 및 사용자 입력 가능
- 하드웨어 SHA3와 NTT를 탑재하여 HW/SW 시험

Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth
(1.0V, 0.95V, 0.9V)

	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160
	Slices	2,000	2,600	3,650	5,200	8,150
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5
	Maximum Single-Ended I/O	150	250	150	250	250
I/O Resources	Maximum Differential I/O Pairs	72	120	72	120	120
	DSP Slices	40	45	80	90	120
Embedded Hard IP Resources	PCIe® Gen2 ⁽¹⁾	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate) ⁽²⁾	2	4	4	4	4
	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
Speed Grades	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

ML-KEM 알고리즘 HW/SW 공동설계 결과 (HW SHA3 & NTT/INTT)

연산	SW Only	HW SHA3	HW SHA3&NTT
SHA3-256(random, 32B)	209,616	9,738	9,738
SHA3-256(Public-key, 1184B)	1,858,072	59,521	59,521
SHA3-512({tmp1, tmp2}, 64B)	214,396	14,557	14,557
Unpack public-key, Poly_from_msg	68,924	68,924	68,924
Generate Matrix	6,221,942	626,450	626,450
Polynomial get noise	1,739,794	1,739,794	1,739,794
NTT	1,395,532	1,395,532	210,369
Polynomial Multiplication	2,355,000	2,355,000	2,355,000
INTT	1,958,736	1,958,736	81,200
Polynomial Addition & Reduction	294,341	294,341	294,341
Pack Ciphertext	186,266	186,266	186,266
SHA3-256(Ciphertext, 1088B)	1,847,560	49,009	49,009
SS:=SHAKE256	215,967	16,128	16,128
RISC-V 32bit Core, Encapsulation 기준	18,566,146	8,773,996	5,711,297

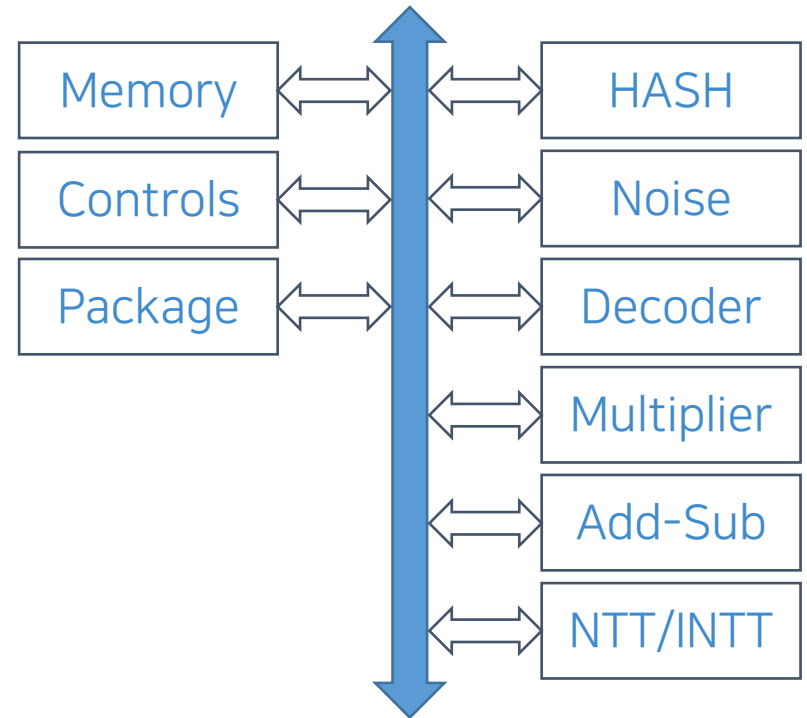
ML-KEM의 전용 하드웨어 설계 연구 분석

Kyber768 Encapsulation, 19,586 clocks

Data load	: ~307 clocks
HASH	: ~801 clocks
Noise for vector	: ~2,333 clocks
Generate matrix	: ~6,009 clocks
Decode	: ~6,777 clocks
NTT	: ~11,478 clocks
Multiply	: ~11,864 clocks
INTT	: ~18,374 clocks

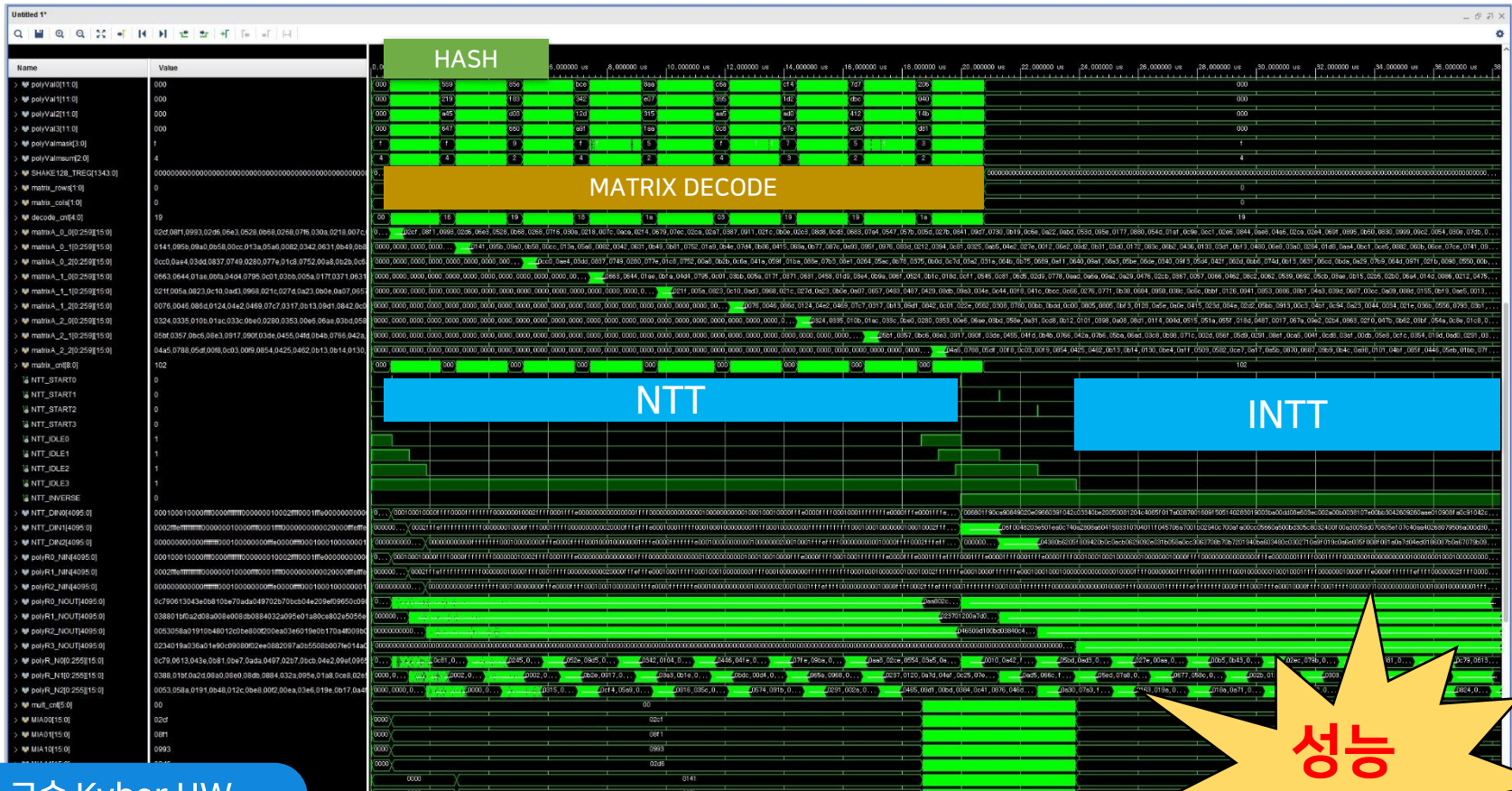
FPGA Utilization (Virtex-7)

LUT	: 17,702
LUTRAM	: 2,962
FF	: 7,029
BRAM	: 8.5
DSP	: 43





ML-KEM의 전용 하드웨어 설계 연구 분석



서버용 고속 Kyber HW

4개의 NTT/INTT 병렬화

곱셈연산 6코어 병렬처리/덧셈연산 4코어 병렬처리

성능
x8

| 03 |

양자내성 암호의 하드웨어 구현

- KpqC 사례 -

KpqC (NTRU+) 알고리즘 분석

NTRU+ 768 알고리즘 및 참조구현 분석

연산	Cycles	비율(%)
hash_h_Kem	1,251,552	10.748
poly_cbd	54,770	0.470
poly_ntt	1,574,692	13.524
poly_to_bytes	53,042	0.456
hash_g	6,270,749	53.854
poly_sotp_encode	66,839	0.574
poly_ntt	1,574,692	13.524
poly_from_bytes	48,050	0.413
poly_basemul_add	696,624	5.983
poly_to_bytes	53,042	0.456
RISC-V 32bit Core, KEM Enc 기준	11,591,010	100

약 64.6%가
SHA3 관련 연산

약 27%가
NTT 관련 연산

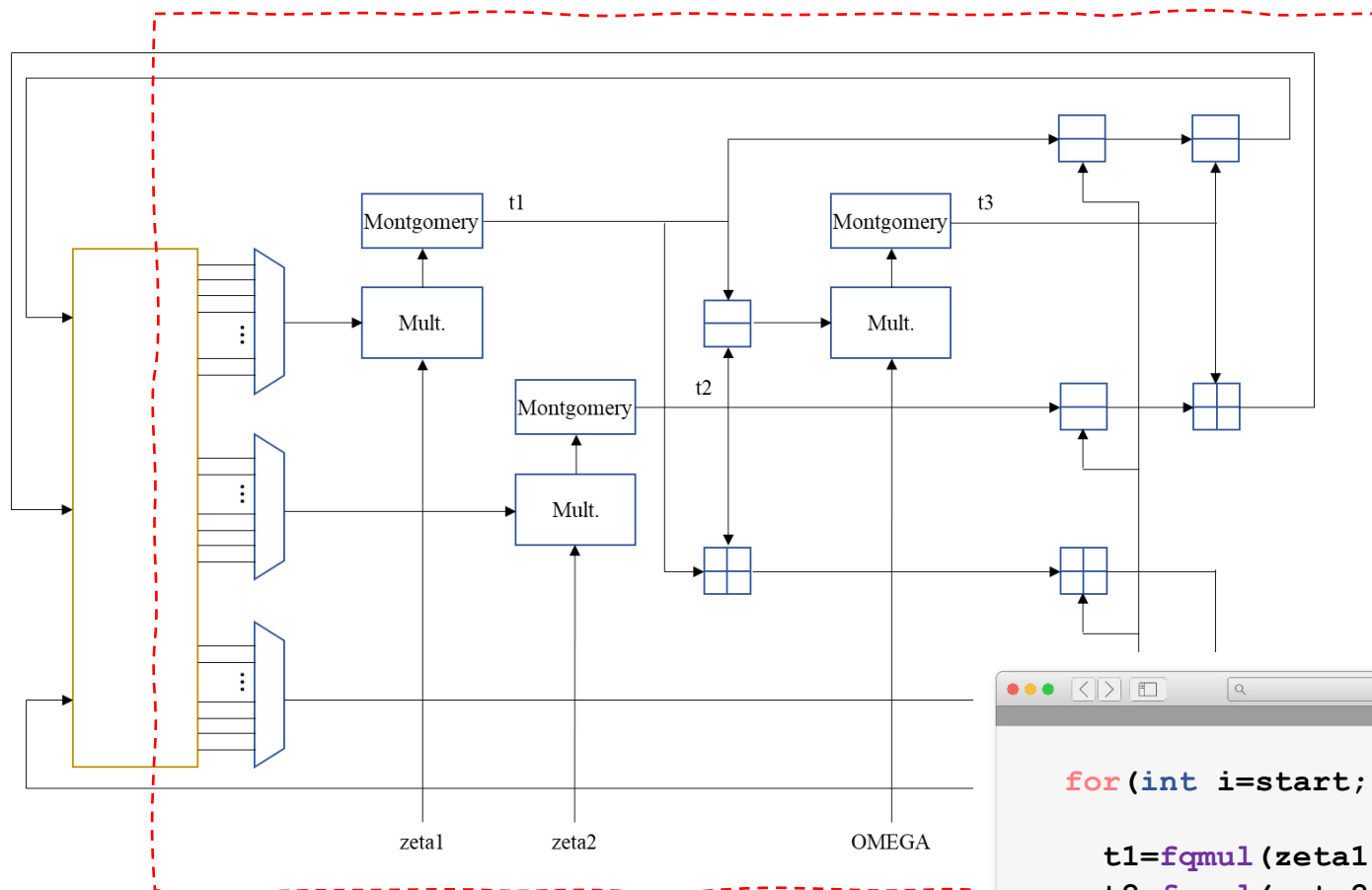
SHA3 하드웨어 연산기를 추가하여 성능을 개선

NTRU+ 768 HW/SW 공동설계 (SHA3 HW 추가)

연산	Cycles	비율(%)
hash_h_Kem	54,582	1.223
poly_cbd	54,770	1.227
poly_ntt	1,574,692	35.289
poly_to_bytes	53,042	1.189
hash_g	285,899	6.407
poly_sotp_encode	66,839	1.498
poly_ntt	1,574,692	35.289
poly_from_bytes	48,050	1.077
poly_basemul_add	696,624	15.612
poly_to_bytes	53,042	1.189
RISC-V 32bit Core, KEM Enc 기준	4,462,232	100

NTT 연산에
소요되는 시간이
여전히 큼

NTT 하드웨어 연산기를 추가하여 성능을 추가 개선



1 Cycle ?

**하드웨어는 복잡한 작업을
빠르게, 병렬로 수행할 수 있음**

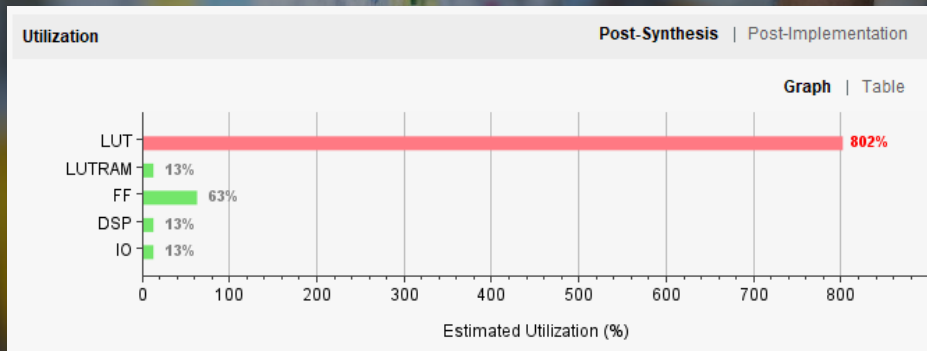
```
for(int i=start; i<start+128; i++) {

    t1=fqmul(zeta1, r[i+128]);
    t2=fqmul(zeta2, r[i+256]);
    t3=fqmul(NTRUPLUS_OMEGA, t1-t2);
    r[i+128]=r[i]-t1-t3;
    r[i+256]=r[i]-t2+t3;
    r[i]=r[i]+t1+t2;

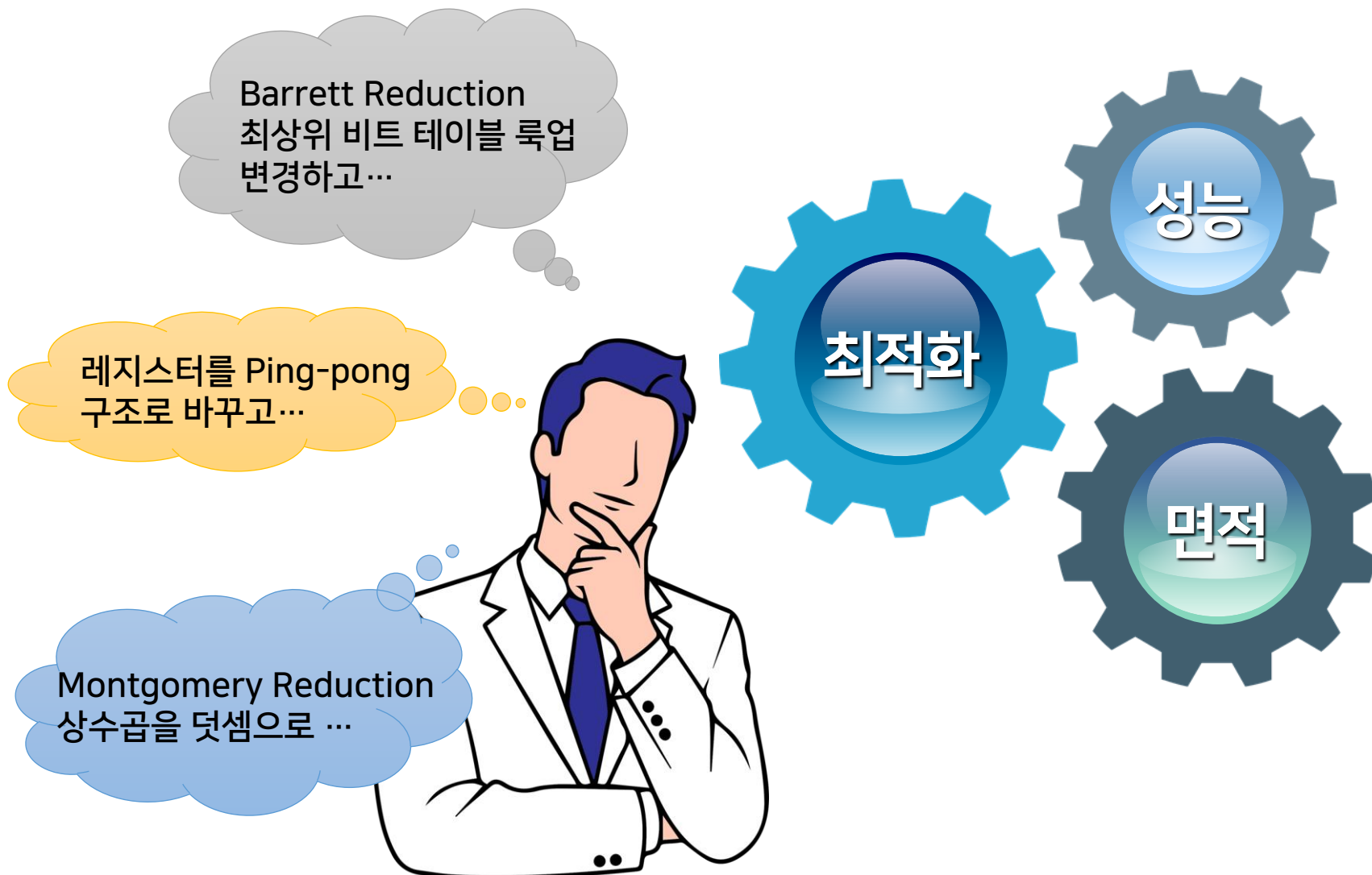
}
```

@Github, NTRUPLUS Reference Implementation

제한된 자원으로
원하는 성능을 내도록
만드는 것은 어렵다



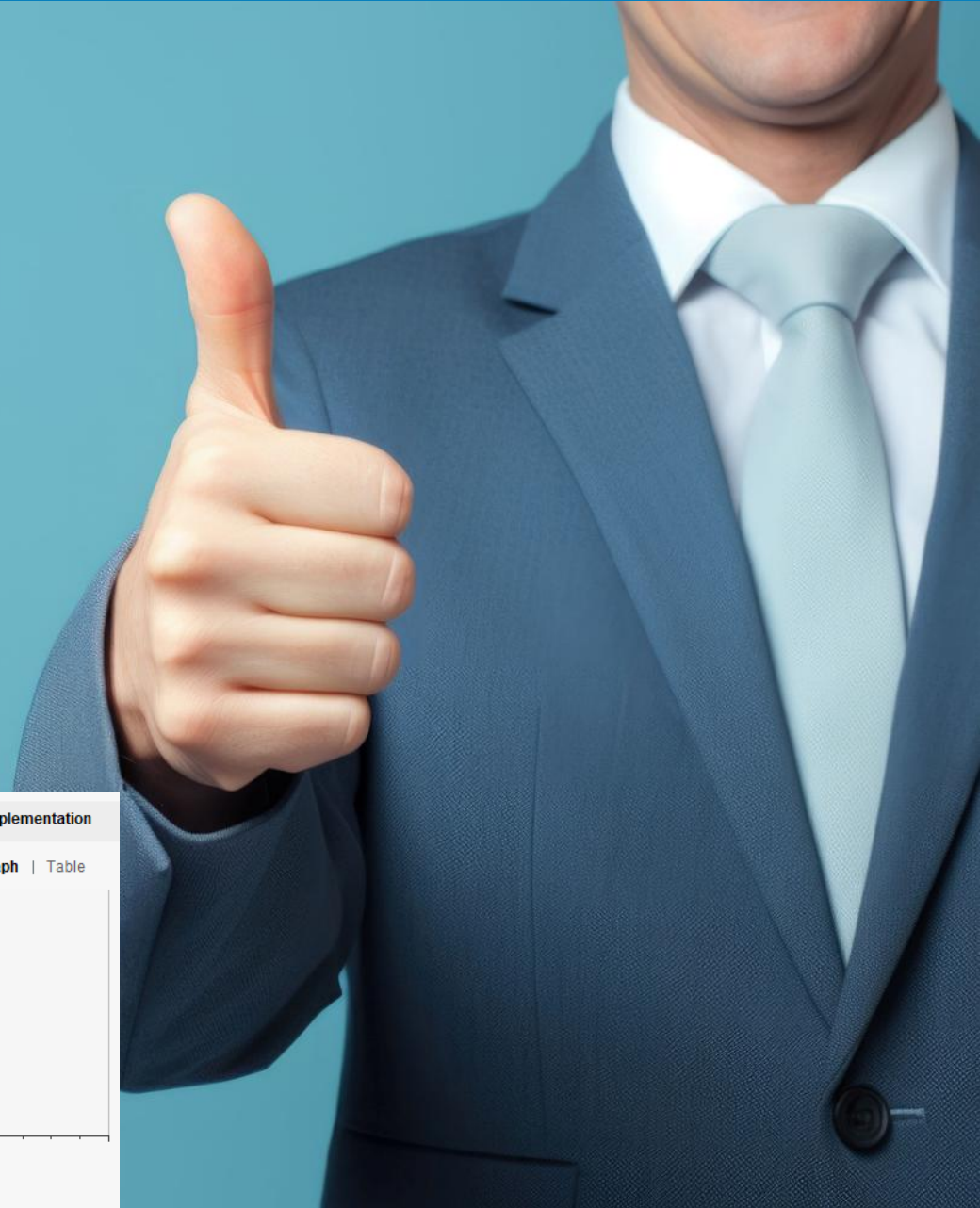
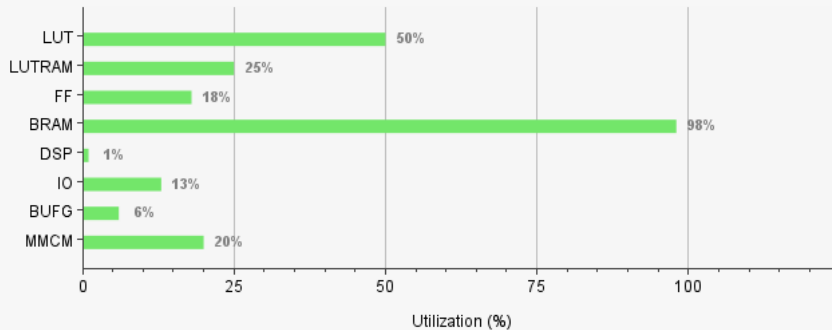
하드웨어 면적(크기)을 고려하여 목표하는 성능을 위한 최적화 구현이 필요



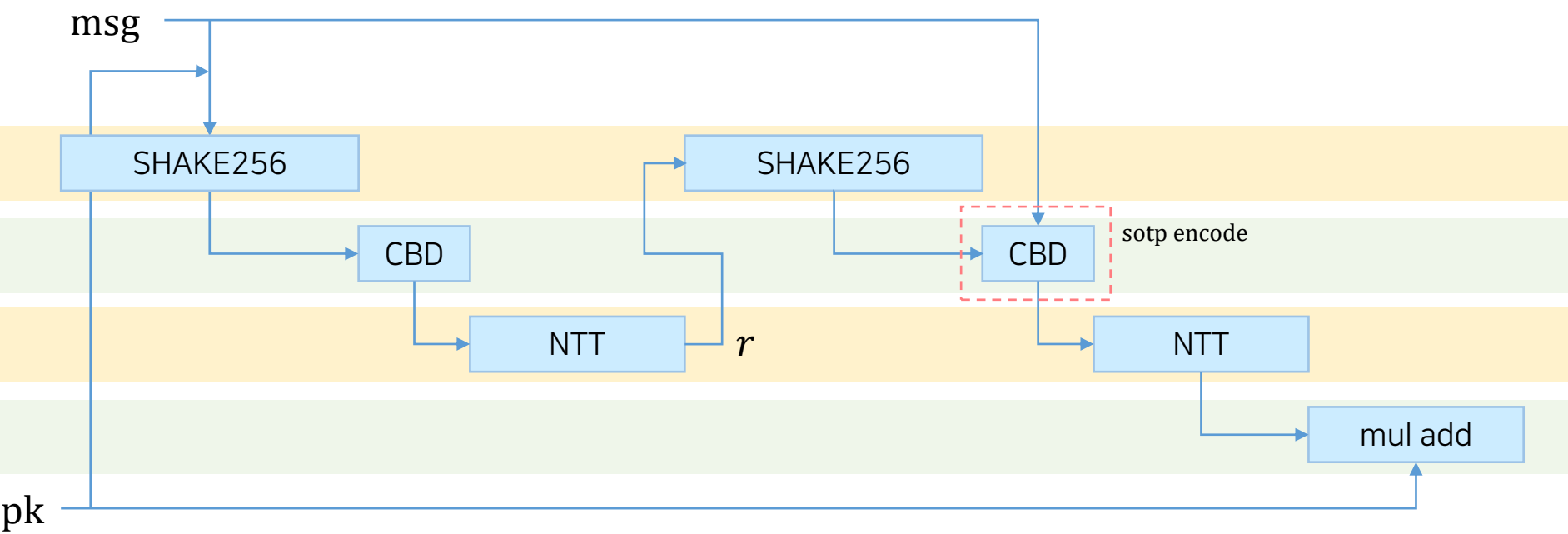
연산	Cycles
hash_h_Kem	54,582
poly_cbd	54,770
poly_ntt	51,434
poly_to_bytes	53,042
hash_g	285,899
poly_sotp_encode	66,839
poly_ntt	53,852
poly_from_bytes	48,050
poly_basemul_add	696,624
poly_to_bytes	53,042
RISC-V 32bit Core, KEM Enc 기준	1,418,134

Synthesis | Post-Implementation

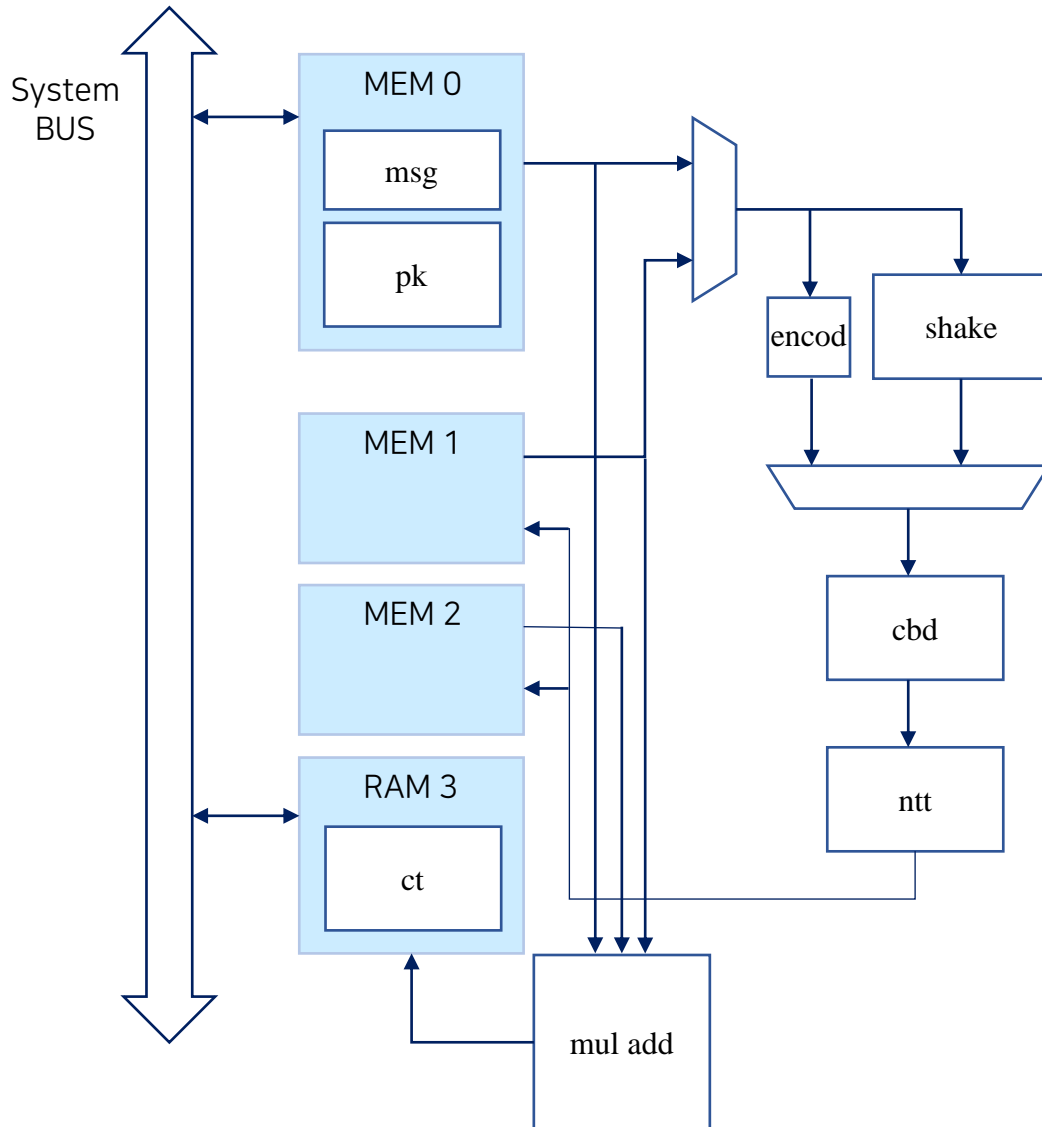
Graph | Table



NTRU+ 알고리즘 흐름 (KEM_ENC 예시)



NTRU+ 알고리즘 하드웨어 블록도 (KEM_ENC 예시)



SHAKE, CBD, NTT 반복 사용 구조
설계 예시

다항식을 바이트로, 바이트를 다항식으로
변환하는 연산은 자원소모 없이 구현 가능

AIMer 알고리즘 with HW SHA3

연산		Reference SW	HW SHA3
Phase 1	GF from byte	2,180	2,180
	Message pre-hashing	601,430	19,321
	Aim2 sbox out	1,574,692	1,574,692
	Generate Matrix LU	28,682,461	14,129,736
	Hash for tree	596,118	14,009
	Expand tree	296,354,464	6,464,182
	GF calculations & Aim2 MPC	517,231,314	143,517,336
Phase 2,3		106,636,357	42,022,258
Phase 4		596,556	14,447
Phase 5		214,920	214,920
RISC-V 32bit Core, Sign 기준		952,469,937	207,952,526

AIM2 알고리즘의
Inverse Mersenne
S-box 연산기,
행렬 생성 및 곱셈
연산기가 추가로 필요

GF 연산기 필요

| 04 |

정리 및 질의 응답



01

암호 구현 방법

암호를 구현하는 두 가지 방법
소프트웨어 vs 하드웨어

하드웨어 구현

전용 하드웨어 vs
하드웨어/소프트웨어
공동설계

하드웨어 구현 대상

FPGA vs ASIC
FPGA는 빨리 만들 수 있고
ASIC은 비싸지만 효율적

02

ML-KEM 구현

하드웨어/소프트웨어 공동설계
전용하드웨어 구현(효율구현/고속구현)

HW/SW 공동설계

SHA3 & NTT HW로
3배 이상 가속 가능

전용 하드웨어

약 2만 클럭 수준 동작
고속 구현 시 8배 가속

03

KpqC 구현

알고리즘 분석 및
하드웨어를 효율적으로 구현 필요

HW/SW 공동설계

NTRU+ 기준,
SHA3 & NTT HW로
8배 이상 가속 가능



국가보안기술연구소 
| 감사합니다 |

