Revised-CHAM 어셈블리 최적 구현

정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

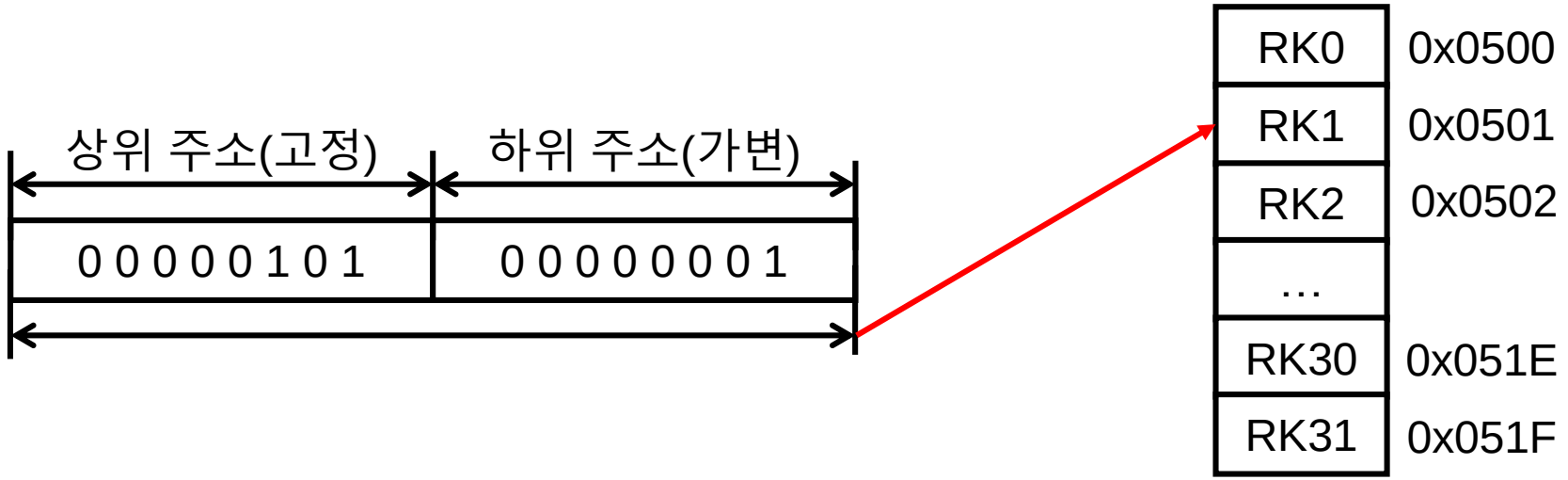
Revised CHAM

- ICISC'19에서 발표된 CHAM의 개량형
- 기존 CHAM과 구조 동일
- 라운드 수를 제외한 모든 파라미터 동일

규 격	n	k	w	r (구버전)
64-128	64	128	16	88 (80)
128-128	128	128	32	112 (80)
128-256	128	256	32	120 (96)

기존 구현 기법

- 라운드 키 접근 최적화
  - AVR 상에서 **메모리 접근은 2개의 레지스터**를 합쳐서 접근
  - CHAM의 RK 크기: 32바이트 또는 64바이트
  - **256바이트 내에서 모든 메모리 접근 가능**
  - 상위 레지스터 고정
  - 하위 레지스터 0x00부터 시작하여 하나로 모든 메모리 접근 가능



기존 구현 기법

- 카운터 최적화
  - 라운드 시작 시 X[0] 블록은 라운드 카운터와 XOR
  - CHAM의 라운드 수는 각각 **80, 80, 96**
  - 레지스터 하나의 최대 표현 범위는 **256**
  - **1바이트 하나로 카운터 관리가 가능**
  - Revised CHAM은 88, 112, 120이므로 **동일 기법 적용 가능**

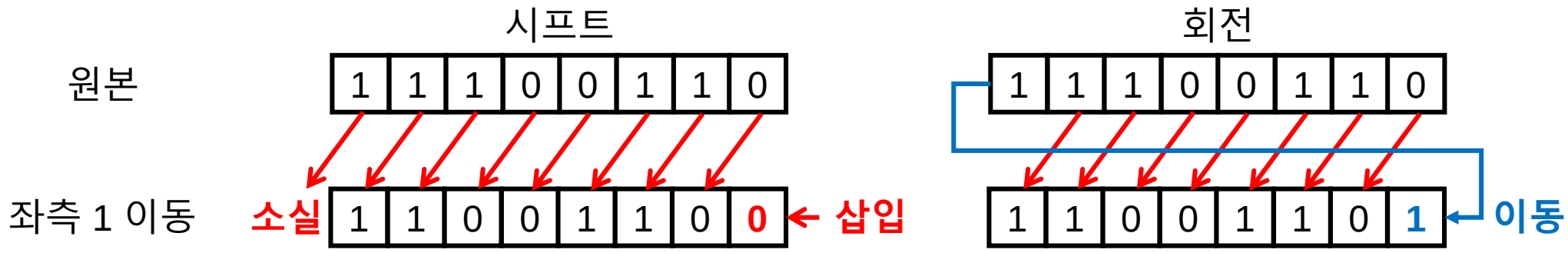
- 메모리 접근 최적화
  - Post incremental 명령어를 통해 메모리 접근 이후 다음 주소로 자동 이동

제안 기법

- 기존 구현 기법에 추가로 다음 기법들을 적용
- **회전 연산 최적화**
- **회전 연산 횟수 최적화**
- **블록 이동 최적화**

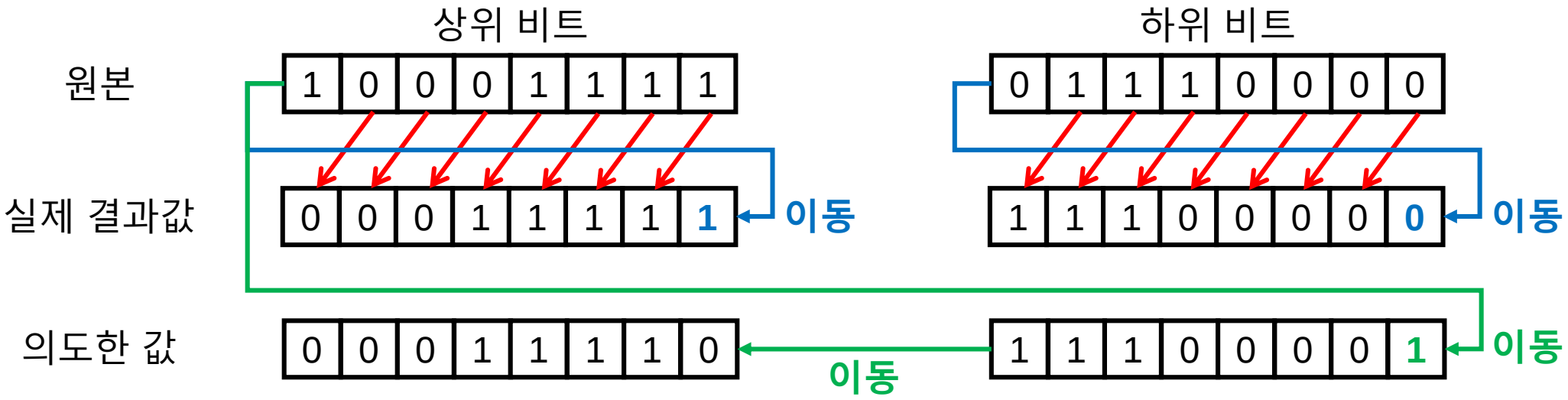
제안 기법

- 회전 연산 최적화
  - 시프트(shift) 연산과 회전(rotation) 연산은 서로 다른 연산
  - 시프트: 한쪽 방향으로 이동, **한쪽 끝은 값이 소실, 반대쪽은 0 삽입**
    - AVR의 LSL 명령어는 한쪽 끝 값이 소실 되지만 **캐리 플래그에 저장**됨
  - 회전: 한쪽 방향으로 이동, **한쪽 끝의 값은 반대편으로** 이동
    - AVR의 ROL 명령어는 캐리 플래그에 저장 밑 호출 과정이 포함



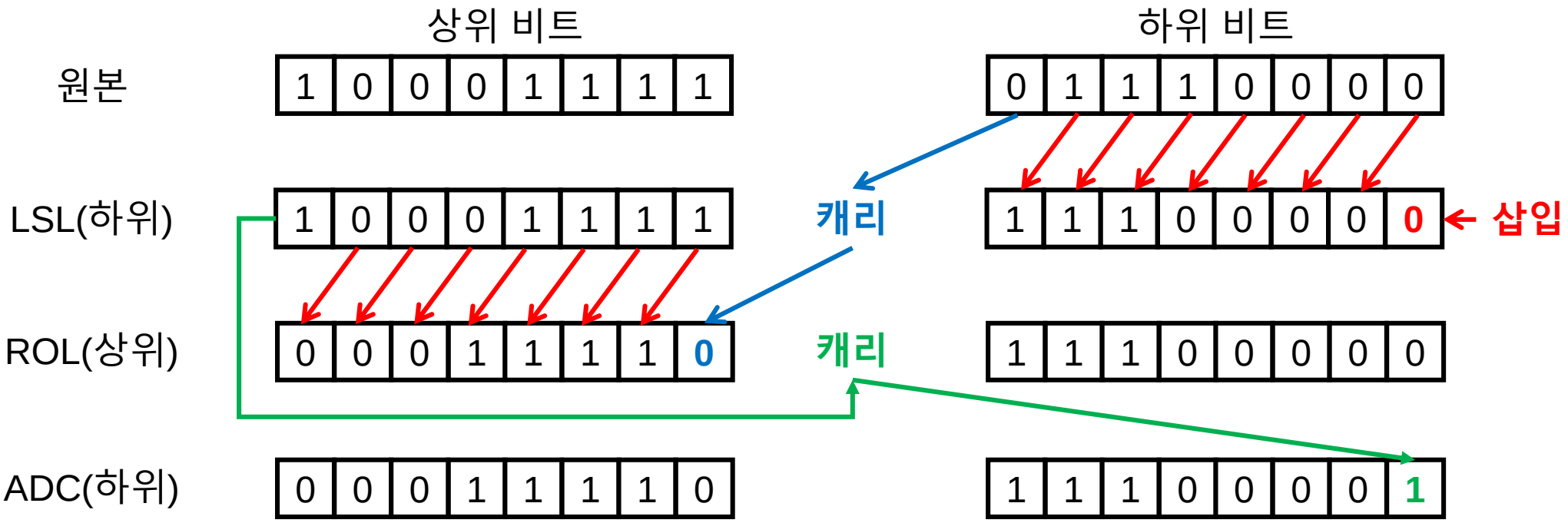
제안 기법

- Revised CHAM-64/128의 블록 하나는 16비트
- 8비트 AVR 프로세서이므로 **블록 하나를 레지스터 두 개에 저장**
- **ROL 명령어 하나로는 정상적인 회전 연산이 되지 않음**



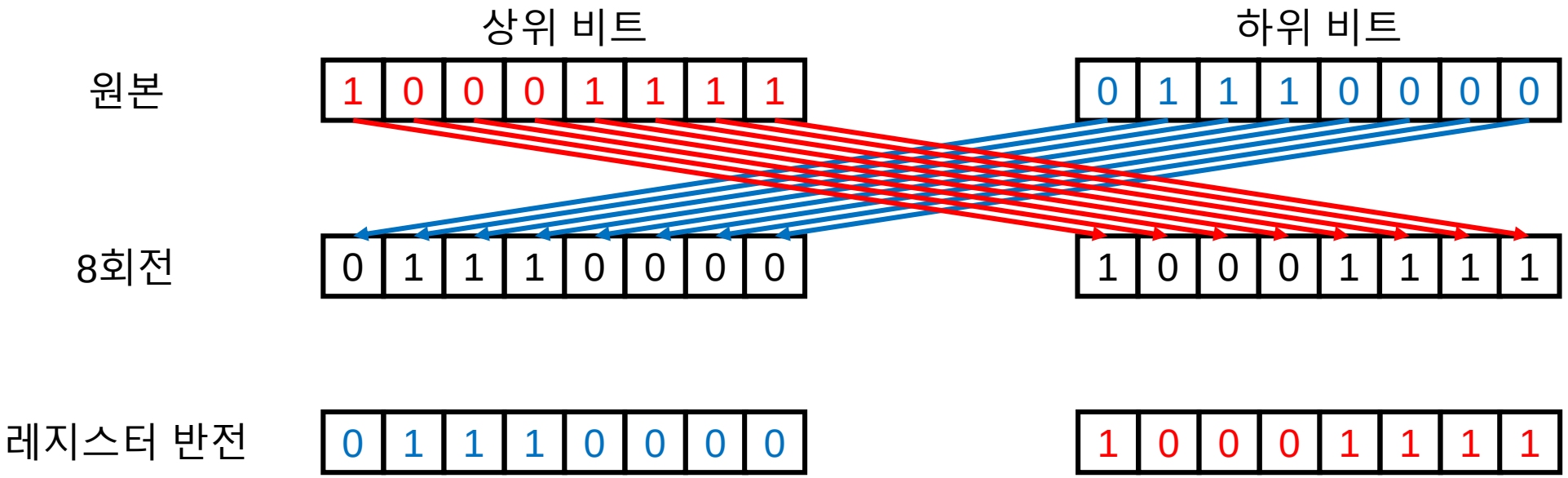
제안 기법

- 실제 연산을 위해 **LSL, ROL, ADC 명령어를 조합**하여 사용



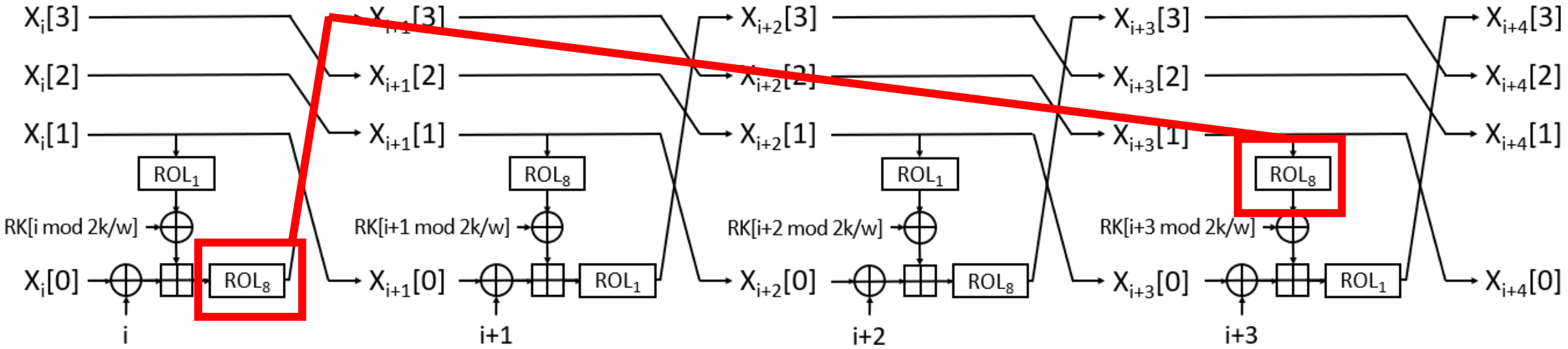
제안 기법

- 8회전 연산의 경우, 레지스터 상,하위 값을 뒤집는 것으로 가능
  - **8비트 레지스터이므로 가능한 구현 기법**
  - MOV 명령어 3개로 구현 가능



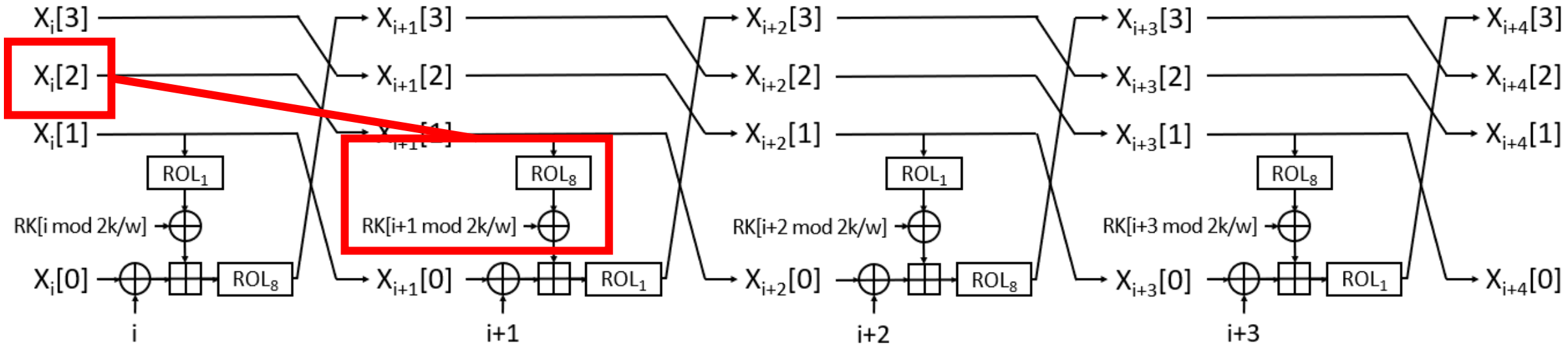
제안 기법

- 회전 연산 횟수 최적화
  - $X_i[0]$는 1라운드 종료시에 8회전 연산을 취함
  - $X_i[0]$가 입력 값으로 사용되는 4라운드에 8회전 연산을 취함
  - **8회전 연산이 두 번 적용되면, 원래 값으로 돌아옴**
  - 본 특성을 고려하면, **1라운드 종료시엔 8회전 연산을 생략** 가능

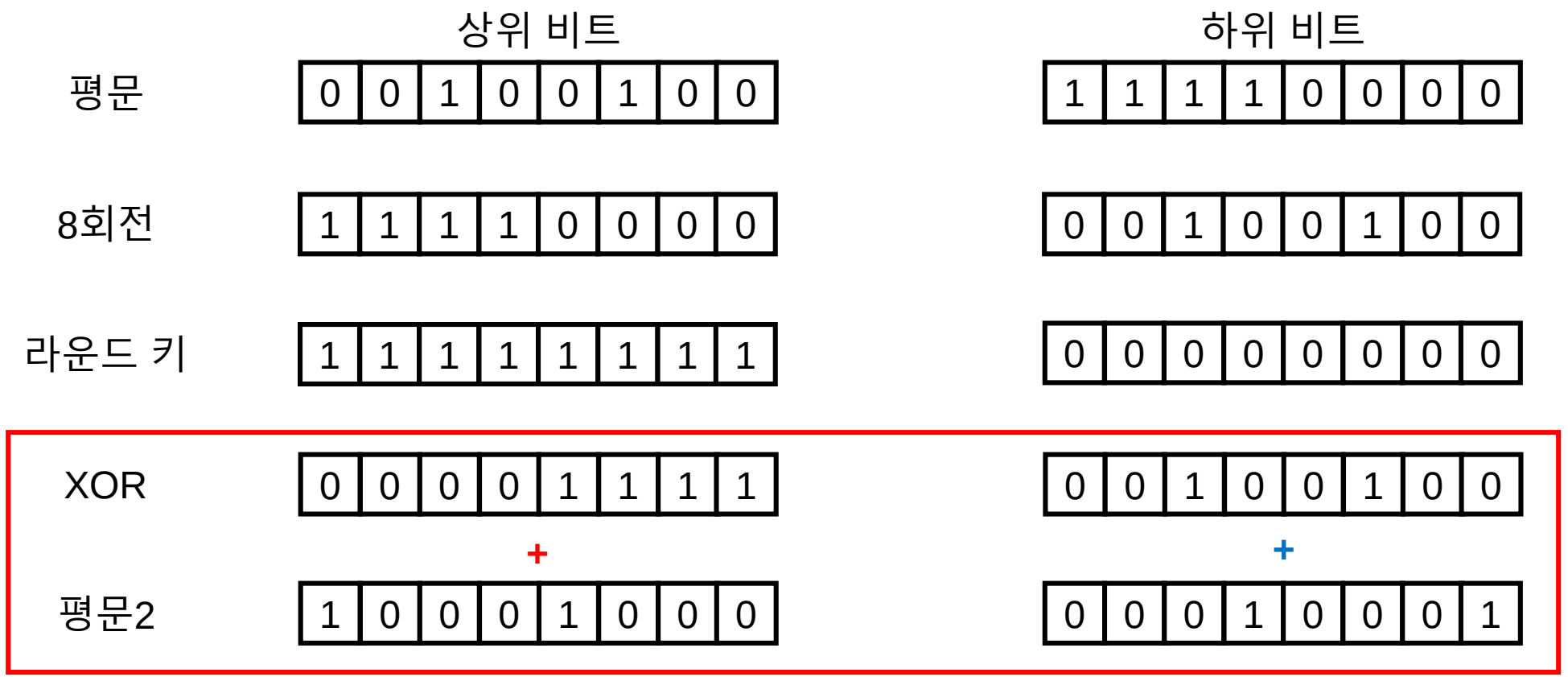


제안 기법

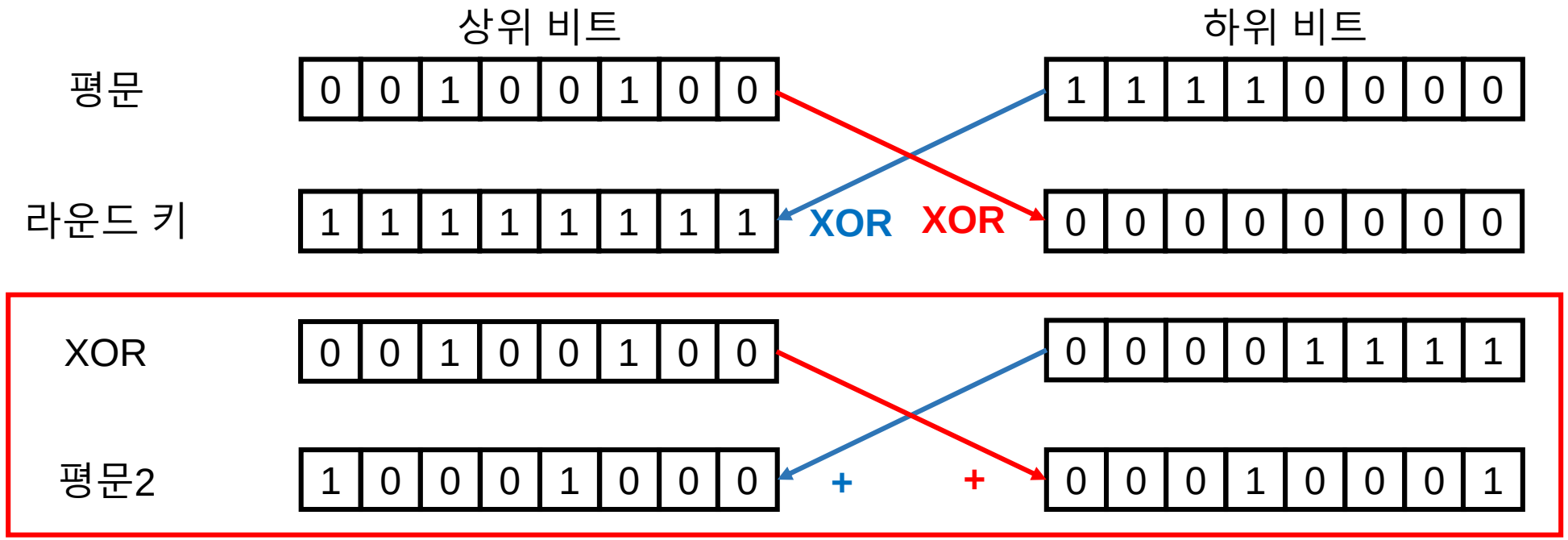
- 회전 연산 횟수 최적화
  - $X_i[2]$는 8회전 연산 후, RK와 XOR 연산을 취함
  - 상,하위 **레지스터를 서로 교차하는 것으로 8회전 연산을 포함**
  - 따라서 4라운드마다 2회의 ROL_8을 생략 가능
    - 전체 88라운드 중 44회 생략 가능



제안 기법

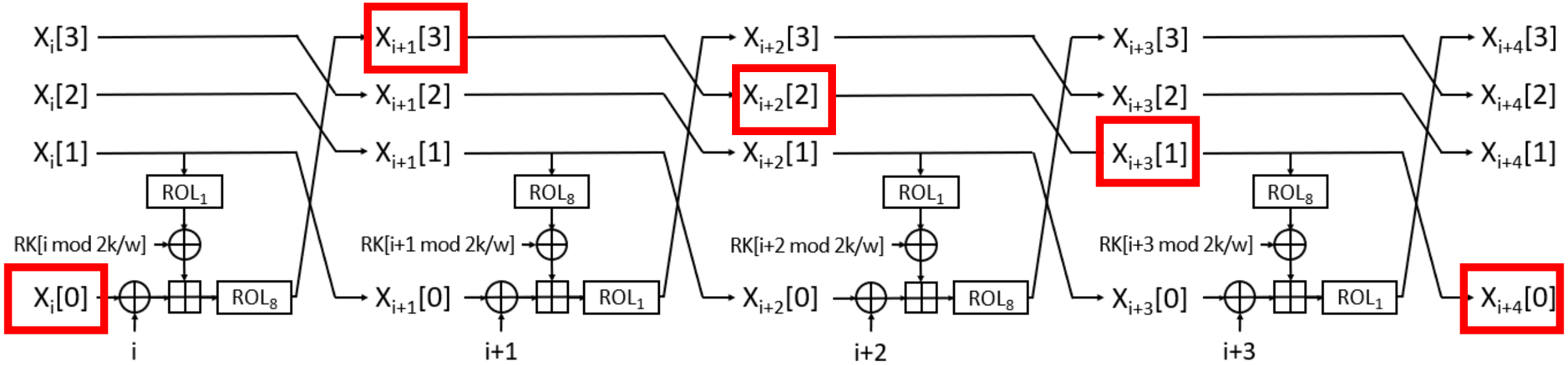

상위 비트
하위 비트
평문
0 0 1 0 0 1 0 0
1 1 1 1 0 0 0 0
8회전
1 1 1 1 0 0 0 0
0 0 1 0 0 1 0 0
라운드 키
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
XOR
0 0 0 0 1 1 1 1
0 0 1 0 0 1 0 0
+
+
평문2
1 0 0 0 1 0 0 0
0 0 0 1 0 0 0 1

제안 기법


상위 비트
하위 비트
평문
0 0 1 0 0 1 0 0
1 1 1 1 0 0 0 0
라운드 키
1 1 1 1 1 1 1 1
XOR
XOR
0 0 0 0 0 0 0 0
XOR
0 0 1 0 0 1 0 0
0 0 0 0 1 1 1 1
평문2
1 0 0 0 1 0 0 0
+
+
0 0 0 1 0 0 0 1

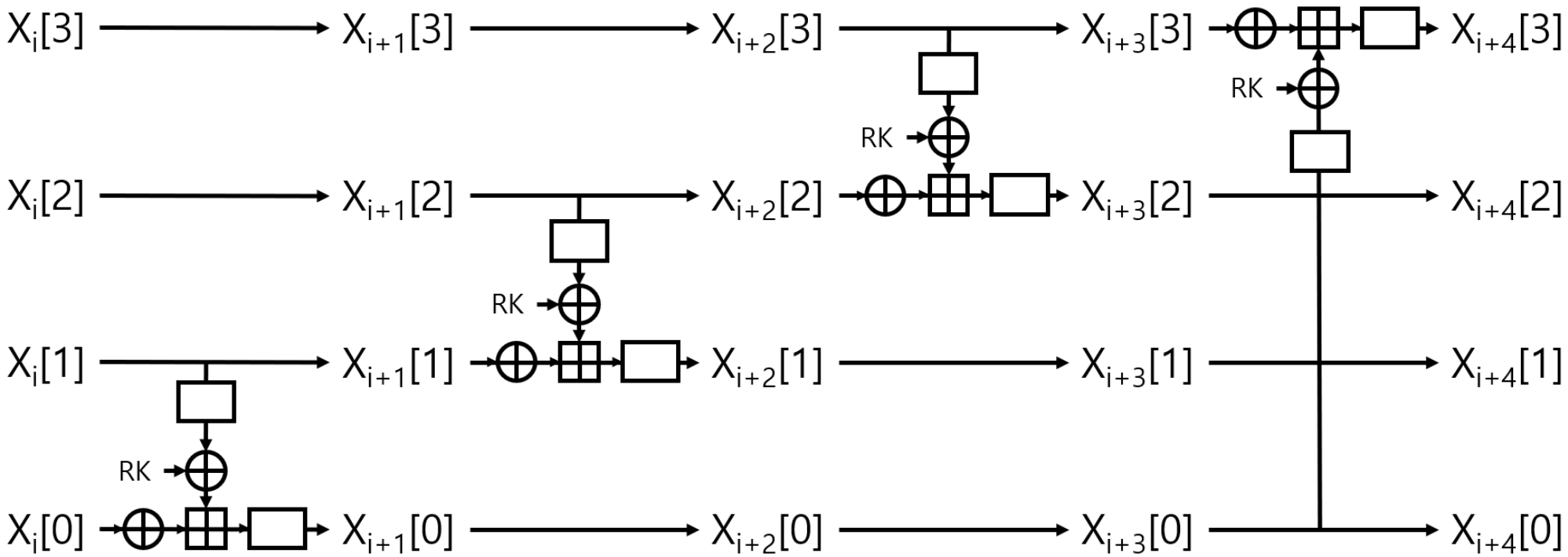
제안 기법

- 블록 이동 최적화
  - Revised CHAM의 블록 이동은 **4라운드 반복 시, 제자리**로 돌아옴
  - 64/128 규격 기준, 88라운드는 4의 배수
  - **모든 블록은 암호화 종료 후 제자리**로 돌아옴



제안 기법

- 모든 라운드에서 **블록 이동을 제거**



성능 평가

- 8bit AVR 프로세서인 Atmega128을 대상으로 구현
- 기존 구현물이 CHAM 기준이므로 이것을 Revised로 이식
  - 약 10% 빠른 동작

구현물	Cpb
기존 구현 + Revised	232
제안 기법	209

결론

- Revised CHAM을 대상으로 구현
  - 64/128 규격에 대해서만 최적, 다른 규격 최적화도 필요
- 코드 길이가 상대적으로 김
  - 코드 사이즈 부분에서 손해인 만큼, 속도를 더 개선할 필요