# SHA256 알고리즘 및 양자회로 구현

**임세진**

https://youtu.be/kd_fZW16lRc

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

# Contents

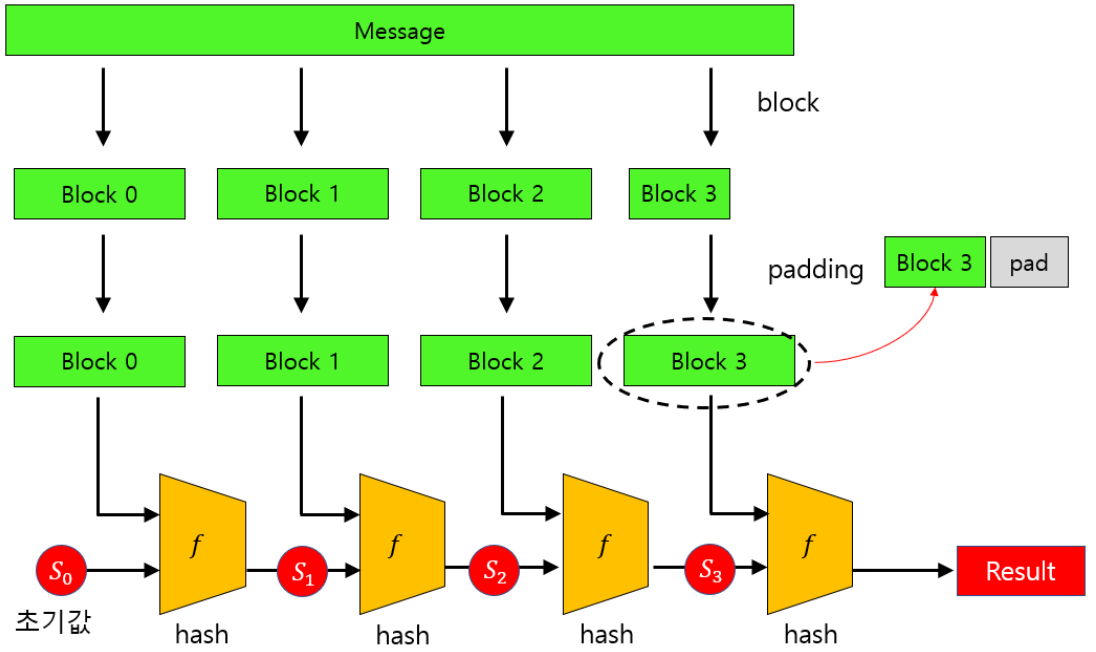


CryptoCraft LAB

# 01. SHA256 알고리즘

## SHA256 (Secure Hash Algorithm)

- 해시 값의 길이에 따라 6가지로 구성 (**일부 상수 및 라운드 수를 제외하고 구조적으로 동일**)
  - SHA224, SHA256, SHA384, SHA512, SHA512/224, SHA512/256
- 크게 전처리 단계와 해시 연산 단계로 나눌 수 있음
  - 전처리 단계 : 메세지 M → padding + parsing → 512 bit의 블록
  - 블록 개수만큼 해시 연산 반복 수행 (**양자 구현 시 블록 개수 1로 가정**)
  - 해시 연산 라운드 수 : 64 라운드



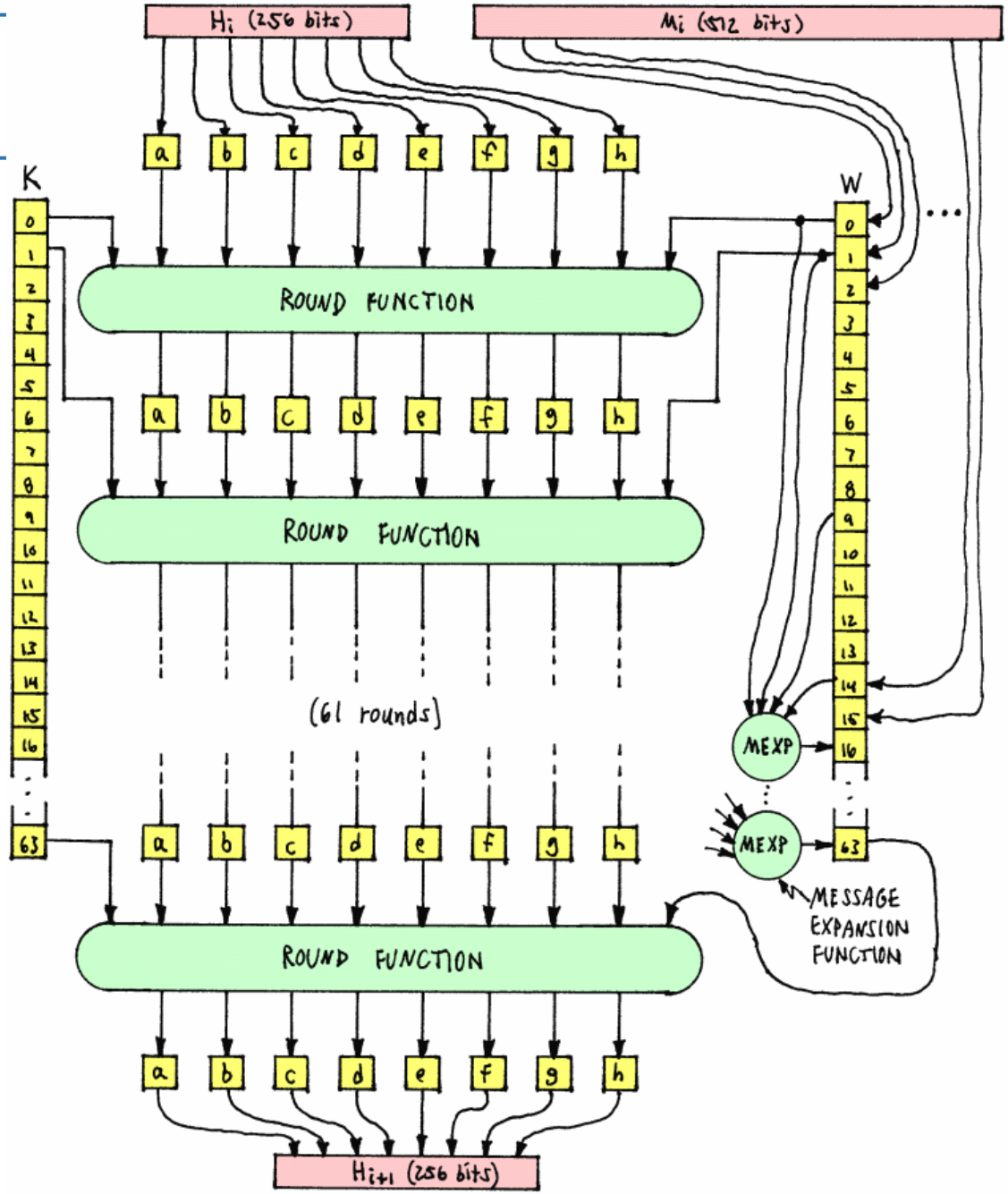
# 01. SHA256 알고리즘

$H_0^{(0)}$ = 6a09e667
$H_1^{(0)}$ = bb67ae85
$H_2^{(0)}$ = 3c6ef372
$H_3^{(0)}$ = a54ff53a
$H_4^{(0)}$ = 510e527f
$H_5^{(0)}$ = 9b05688c
$H_6^{(0)}$ = 1f83d9ab
$H_7^{(0)}$ = 5be0cd19

- $SHA(H, W, K)$ : K는 상수, H의 초기값은 상수
- $H$ $(a, b, c, d, e, f, g, h)$ : 각각 32bit, 총 256bit
  - 맨 처음 $H_0$만 상수
  - 이후에는 매 라운드에 걸쳐 업데이트 됨
- $K$ $(K_0 \sim K_{63})$ : 32bit씩 64개
  - 매 라운드마다 하나의 $K$가 적용됨
  - 전부 주어지는 상수 값

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

- $W$ $(W_0 \sim W_{63})$ : 32bit씩 64개
  - 매 라운드마다 하나의 $W$가 적용됨
  - 메세지 블록을 통해 512bit (32bit씩 16개) 만큼 생성
  - 16개의 값을 4배로 불려서 (by 메세지 확장 함수) 64개로 만듦

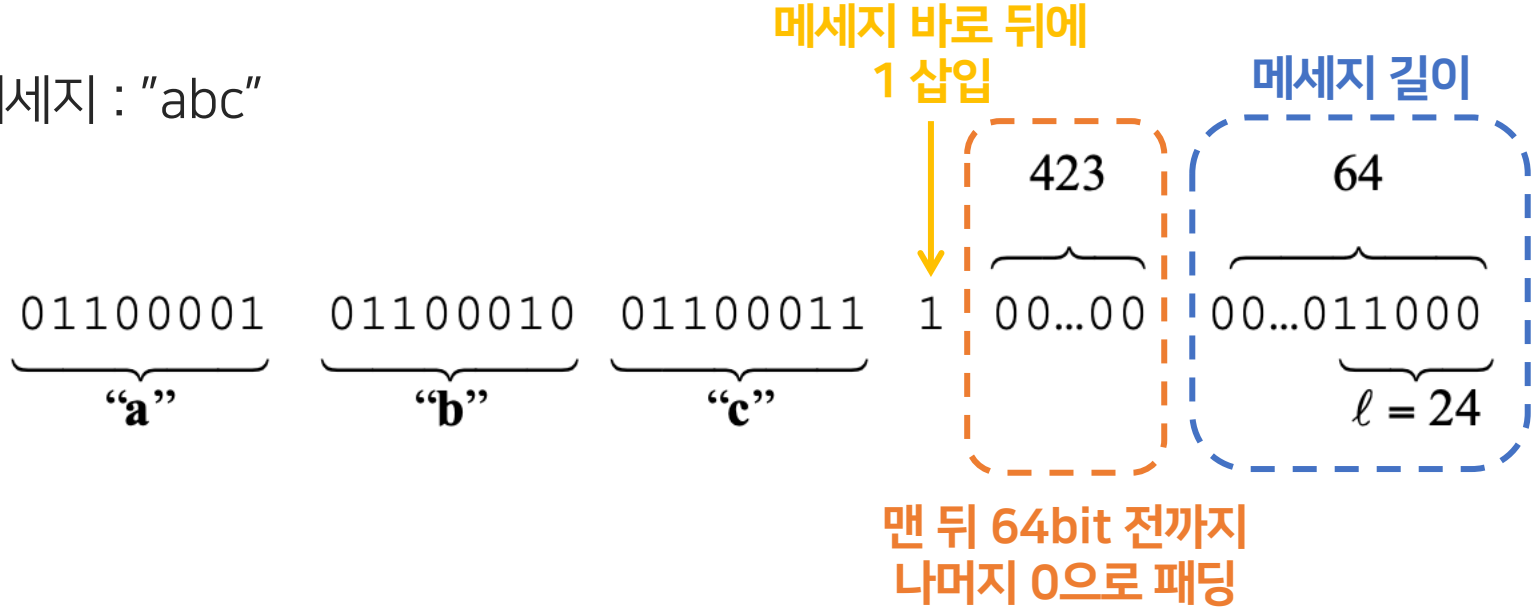



<SHA256 해시 연산 알고리즘>

# 01. SHA256 알고리즘

## <전처리 단계>

- Padding : 입력된 메세지 : "abc"

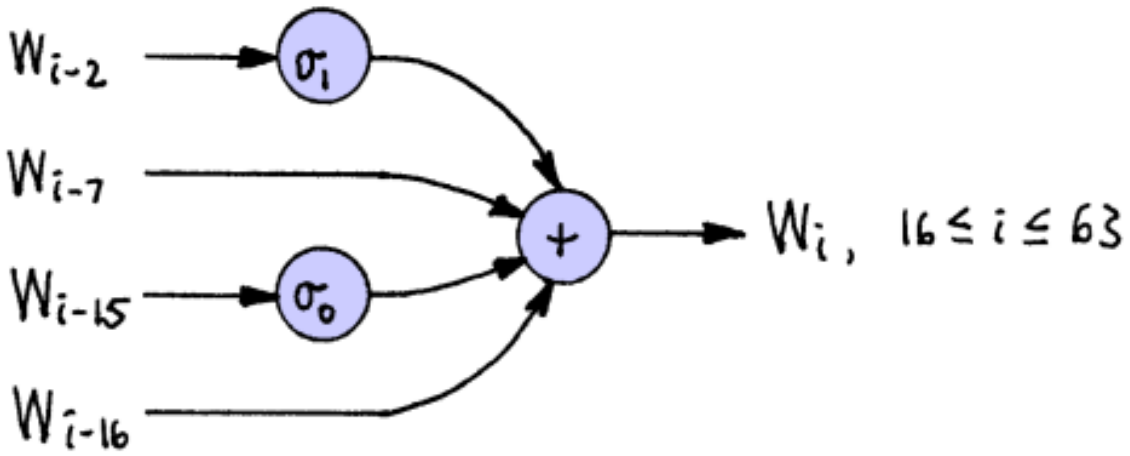



- Parsing : padding을 통해 생성된 512bit의 메세지를 32bit씩 16개로 나눔 → $W_0 \sim W_{15}$

# 01. SHA256 알고리즘

## <해시 연산 단계>

- Message Expansion Function : $W_0 \sim W_{15}$로 부터 $W_{16} \sim W_{63}$ 생성




$$\sigma_0(x) = (x\ right-ratate\ 7)\ xor (x\ ROTR\ 18)\ xor\ (x \gg 3)$$
$$\sigma_1(x) = (x\ ROTR\ 17)\ xor (x\ ROTR\ 19)\ xor\ (x \gg 10)$$

# 01. SHA256 알고리즘

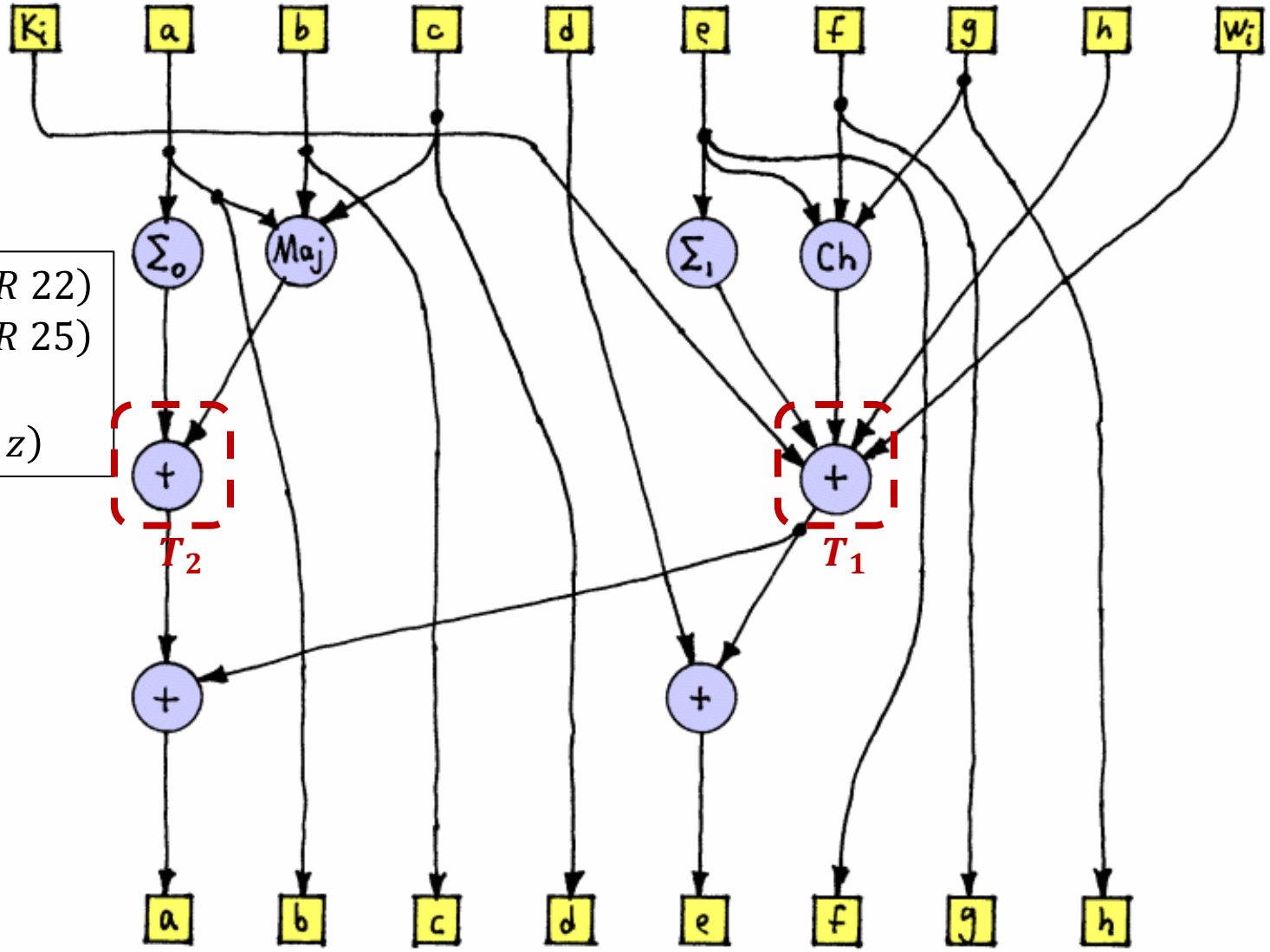
## <해시 연산 단계>

- 라운드 함수

$$\sum_0(x) = (x\ ROTR\ 2)\ xor\ (x\ ROTR\ 13)\ xor\ (x\ ROTR\ 22)$$
$$\sum_1(x) = (x\ ROTR\ 6)\ xor\ (x\ ROTR\ 11)\ xor\ (x\ ROTR\ 25)$$
$$Ch(x,y,z) = (x\ and\ y)\ xor\ \big((not\ x)\ and\ z\big)$$
$$Maj(x,y,z) = (x\ and\ y)\ xor\ (x\ and\ z)\ xor\ (y\ and\ z)$$

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$
$$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$



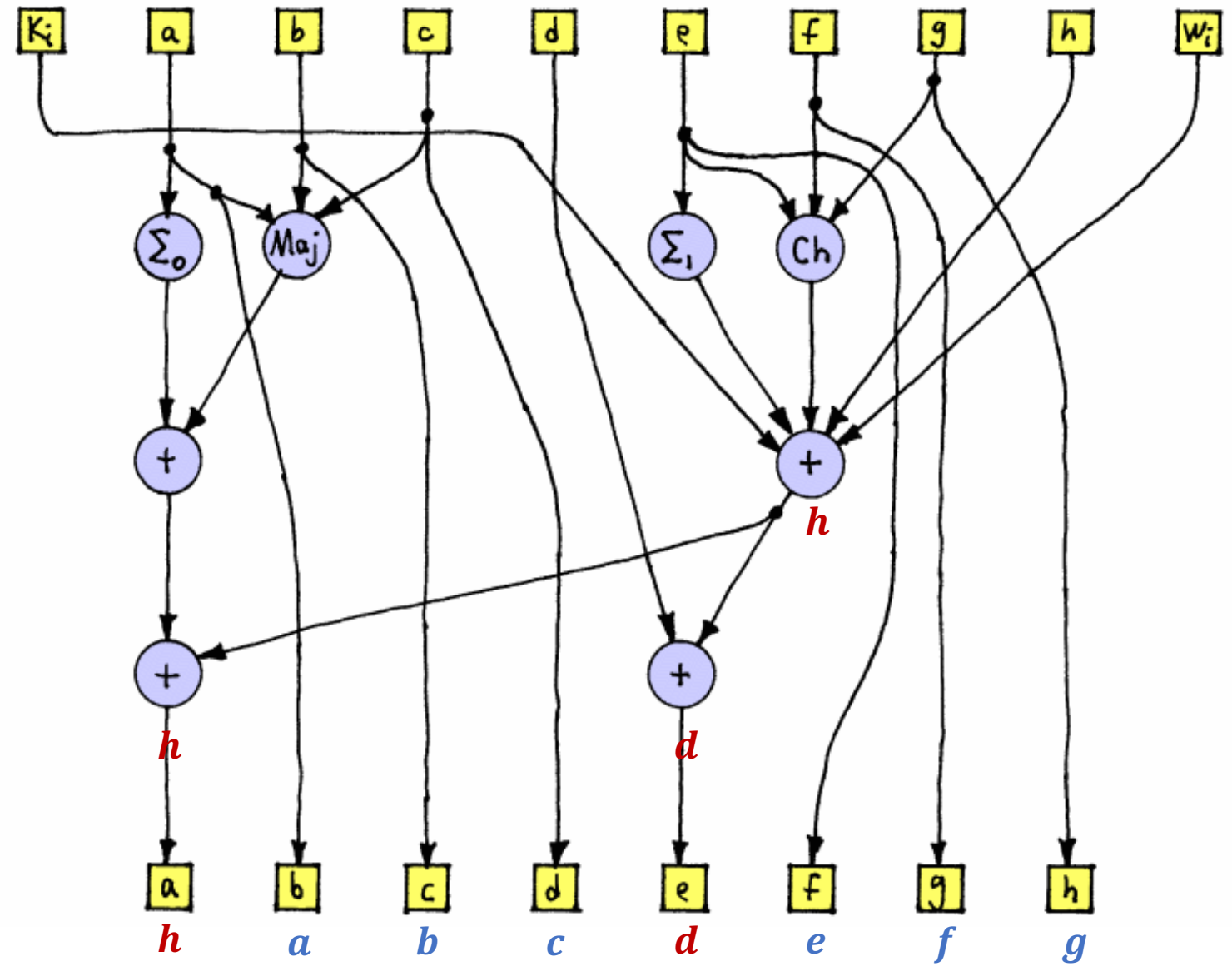
# 02. 양자회로 구현

1. Toffoli-depth를 줄이는 것이 최적화의 핵심

2. Toffoli gate가 사용되는 연산 : $Ch$, $Maj$, 덧셈 연산
   - $Ch$와 $Maj$는 ancilla qubit을 사용하여 Toffoli-depth를 1로 만듦
   - 덧셈 연산은 CLA인 draper adder를 사용하여 Toffoli-depth를 줄임

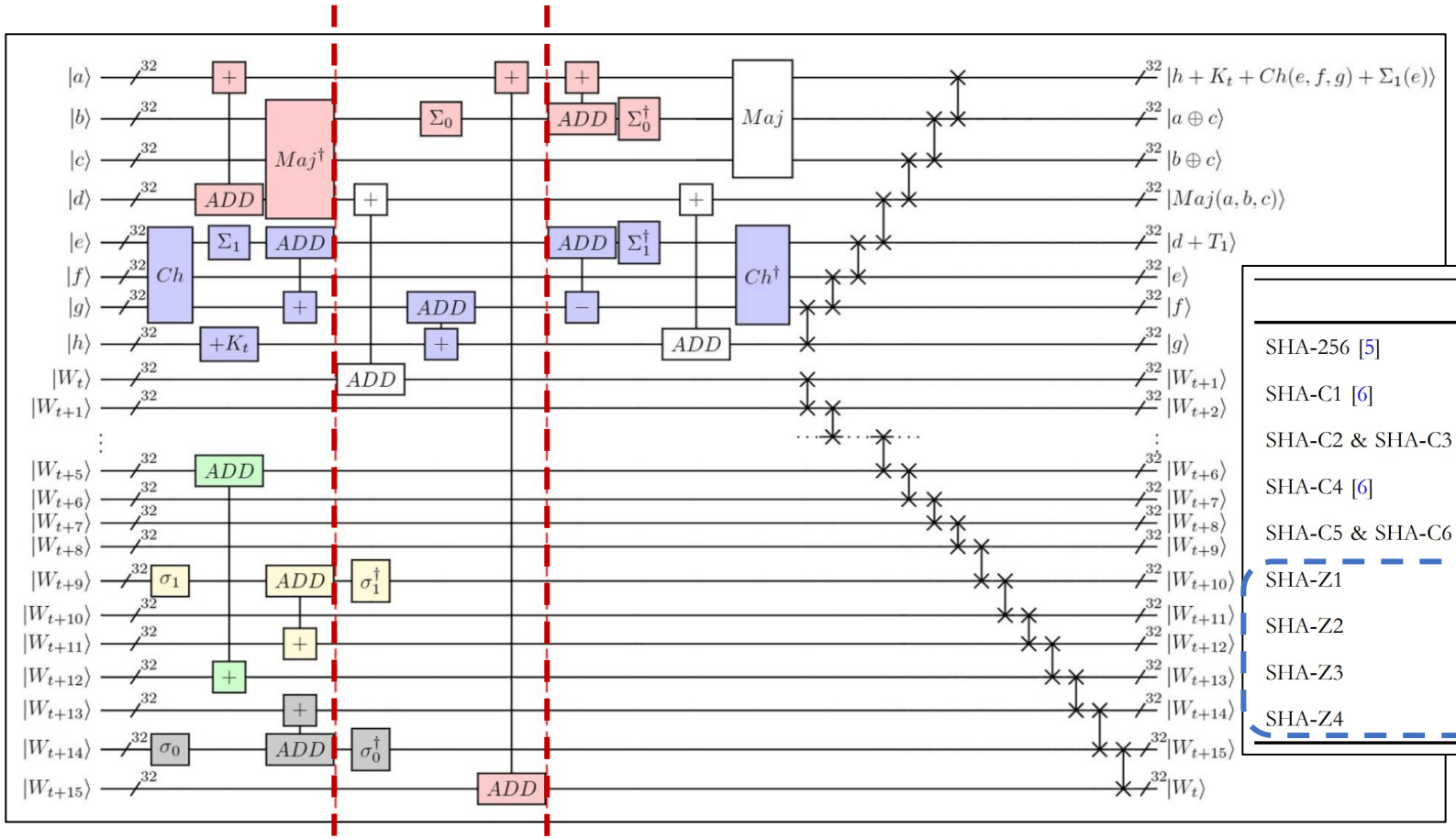
3. 덧셈 연산 : 각 라운드마다 7번, 메세지 확장 함수에서 $W_i$를 구할 때마다 3번
   - 라운드 함수 연산과 메세지 확장 함수 연산이 병렬로 수행된다고 할 때, 10번의 덧셈을 최대한 병렬로 계산하는 것이 핵심

# 02. 양자회로 구현



# 02. 양자회로 구현

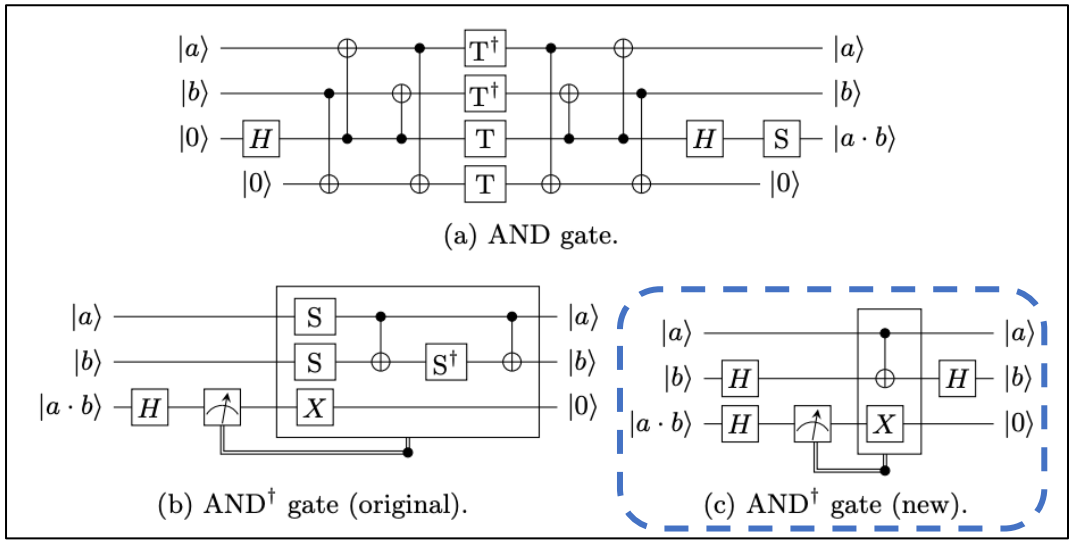
- Target 논문
  - 덧셈을 병렬로 **총 3번** 수행



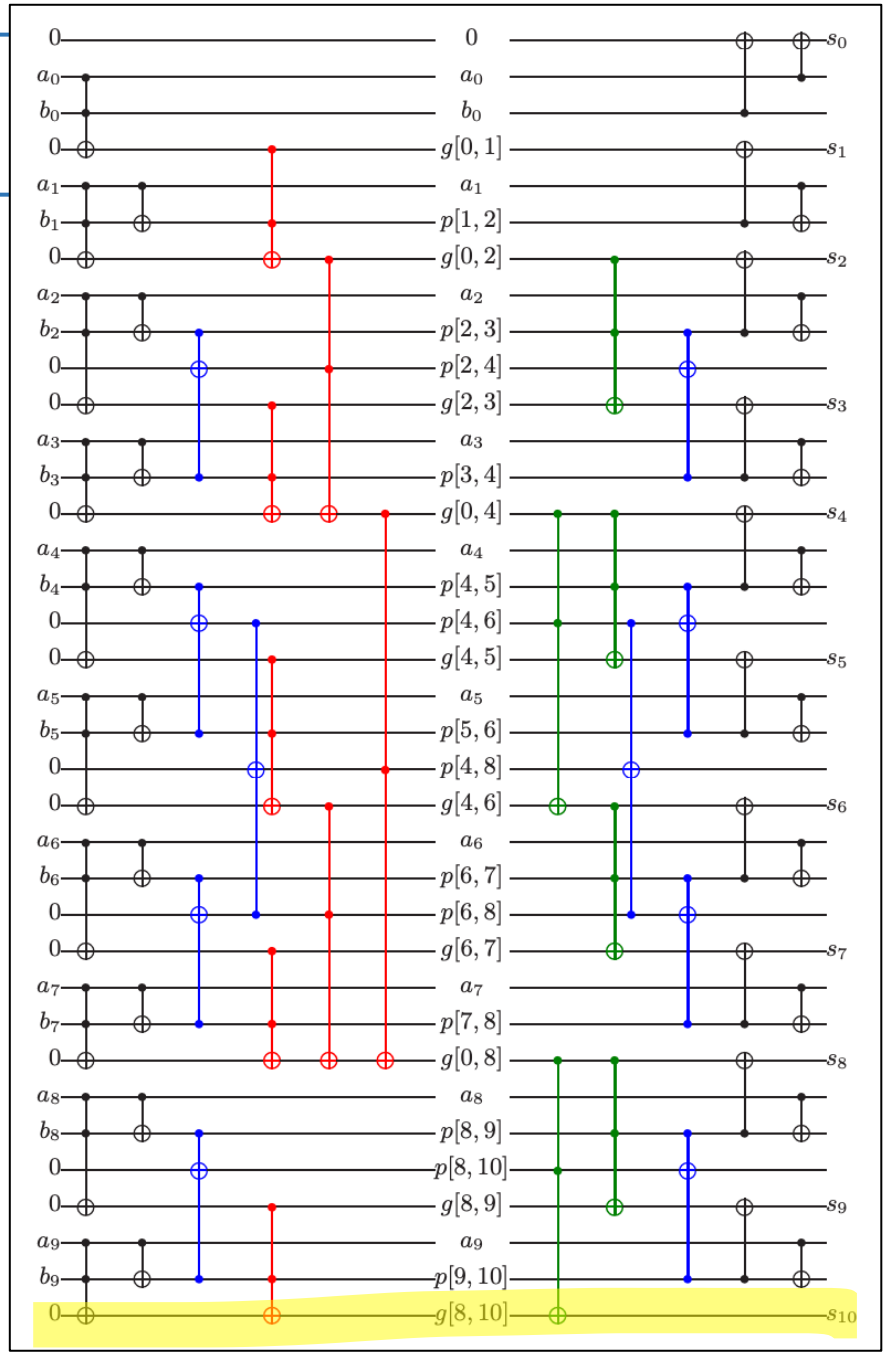

| | Width | T-depth | Toffoli-depth | Used quantum adder |
|---|---|---|---|---|
| SHA-256 [5] | 2402 | 70,400 | - | CDKM [7] |
| SHA-C1 [6] | 801 | - | 36,368 | CDKM [7] |
| SHA-C2 & SHA-C3 [6] | 853 | - | 13,280 | QCLA [12] |
| SHA-C4 [6] | 834 | - | 27,584 | CDKM [7] |
| SHA-C5 & SHA-C6 [6] | 938 | - | 10,112 | QCLA [12] |
| SHA-Z1 | 768 | 43,510 | 32,895 | HRS, TK-v1 |
| SHA-Z2 | 797 | 16,055 | 12,023 | VBE, TK-v1, TK-v3 |
| SHA-Z3 | 927 | 7304 | 6914 | VBE, TK-v3, QCLA |
| SHA-Z4 | 962 | 4936 | 4418 | QCLA, TK-v2 |

# 02. 양자회로 구현

1. Generic Adder → mod $2^n$ 덧셈기로 구조 변경 (최상위 캐리 연산 제거)
2. Draper adder + Quantum AND gate (update) 적용
   - Quantum AND gate 적용은 T-depth 최적화 용도
   - 이전 논문보다 성능을 개선하려면 무조건 Toffoli-depth가 낮아야 함




3. SHA256 양자회로 구현 완료 (최적화 필요)
   - Toffoli-depth 200정도 줄임. 대신 큐비트를 몇 배로 씀,,,
   - 현재는 라운드 1~16까지 덧셈 병렬 4번, 17~64까지 덧셈 병렬 5번 수행 중
   - qubit를 최대한 적게 쓰면서 덧셈 병렬을 3번 수행하면 최적화 가능할 것으로 보임



<draper adder – out of place>

# 감사합니다