# SPECK 양자 회로 최적화

장경배

https://youtu.be/SMbLBHuFbJ8

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

# Quantum Ripple-Carry Adder (기존)

- **2개의 carry 큐비트를** 사용하며, **높은 회로 Depth**

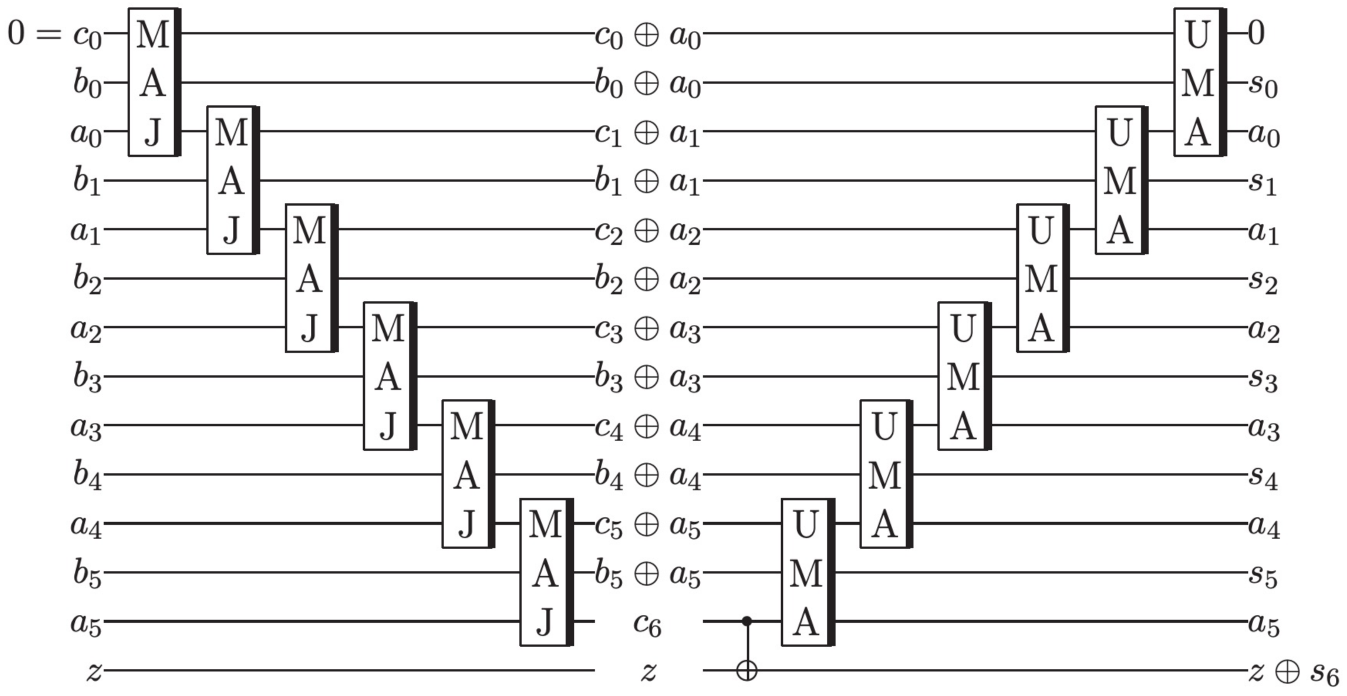
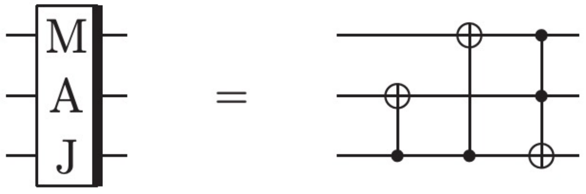
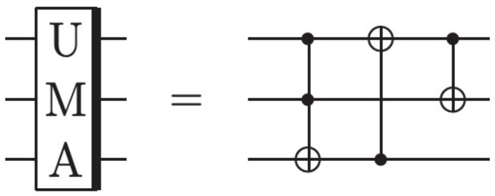


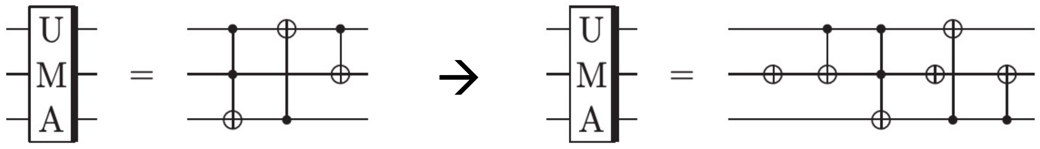

Figure 4: A simple ripple-carry adder for $n = 6$.



# Improved Quantum Ripple-Carry Adder

- **3-CNOT의 UMA를 사용**하면 덧셈기 성능을 **1차적으로 향상**시킬 수 있음 ($n \geq 4$)




(a) 2-CNOT version (b) 3-CNOT version

**input:** $A_i = a_i$ $B_i = b_i$ $Z = z$ $X = 0$
**output:** $A_i = a_i$ $B_i = s_i$ $Z = z \oplus s_n$ $X = 0$
**circuit:**
  **for** $i = 1$ **to** $n-1$: $B_i \oplus= A_i$
  $X \oplus= A_1$
  $X \oplus= A_0 B_0$ ; $A_1 \oplus= A_2$
  $A_1 \oplus= X B_1$ ; $A_2 \oplus= A_3$
  **for** $i = 2$ **to** $n-3$:
    $A_i \oplus= A_{i-1} B_i$ ; $A_{i+1} \oplus= A_{i+2}$
  $A_{n-2} \oplus= A_{n-3} B_{n-2}$ ; $Z \oplus= A_{n-1}$
  $Z \oplus= A_{n-2} B_{n-1}$ ; **for** $i = 1$ **to** $n-2$: Negate $B_i$
  $B_1 \oplus= X$ ; **for** $i = 2$ **to** $n-1$: $B_i \oplus= A_{i-1}$
  $A_{n-2} \oplus= A_{n-3} B_{n-2}$
  **for** $i = n-3$ **down to** 2:
    $A_i \oplus= A_{i-1} B_i$ ; $A_{i+1} \oplus= A_{i+2}$ ; Negate $B_{i+1}$
  $A_1 \oplus= X B_1$ ; $A_2 \oplus= A_3$ ; Negate $B_2$
  $X \oplus= A_0 B_0$ ; $A_1 \oplus= A_2$ ; Negate $B_1$
  $X \oplus= A_1$
  **for** $i = 0$ **to** $n-1$: $B_i \oplus= A_i$

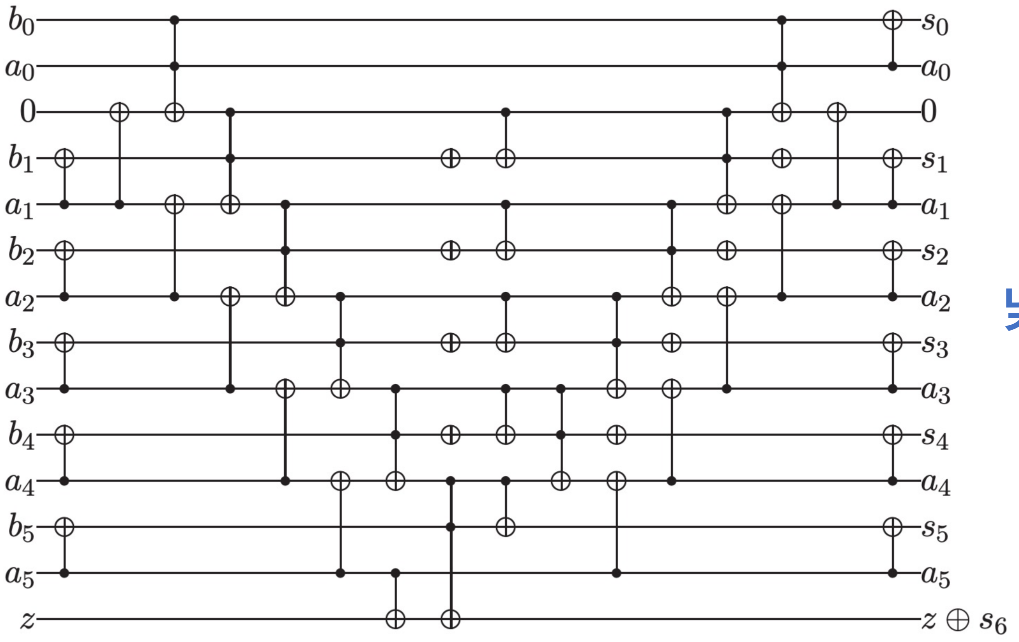



Figure 6: The ripple-carry adder for $n = 6$.

낮은 Depth

# Quantum Ripple-Carry Adder (기존)

## 4.1 Addition Modulo $2^n$

Suppose that we wish to compute $a + b \pmod{2^n}$; that is, we do not want to compute the high bit $c_n$. One approach is the following:

1. Add the low-order $n-1$ bits of $a$ and $b$, using the circuit of Section 3. Use $B_{n-1}$ as the output bit.

2. Set $B_{n-1} \oplus= A_{n-1}$.

After step 1, we have correctly computed $s_0$ through $s_{n-2}$, and we have written $b_{n-1} \oplus c_{n-1}$ into $B_{n-1}$. Then, in step 2, we complete the calculation of $s_{n-1}$. Note that step 2 occurs in parallel with the final time-slice of step 1.

For $n \geq 3$, this circuit contains $2n-3$ Toffolis, $5n-7$ CNOTs, and $2n-6$ negations. The depth is $2n+2$: $2n-3$ Toffoli time-slices and 5 CNOT time-slices.

# 덧셈기 성능

**Evaluation of Quantum Cryptanalysis on SPECK**

Ravi Anand[1], Arpita Maitra[2], Sourav Mukhopadhyay[1]

In [11], K. Jang, S. Choi, H. Kwon and H. Seo analyzed Grover search on SPECK. They followed the same approach which has been followed in [4]. For addition modulo $2^n$, they exploited Cuccaro et al.'s ripple-carry addition circuit [6]. We follow the addition circuit presented in [23]. This is because of the fact that the circuit presented in [23] for adding two $n$-bit numbers exploits $(2n-2)$ Toffoli and $(5n-6)$ CNOT gates and requires no ancilla whereas the circuit described in [6] requires $(2n-2)$ Toffoli , $(5n-4)$ CNOT , and $(2n-4)$ NOT gates and one ancilla. Hence, while the number of Toffoli gates remain the same, the number of Clifford, i.e., (CNOT +NOT ), gates are relatively low in our circuit. Moreover, we need one less qubit compared to [11].

해당 논문에서 사용하는 Adder

기존 SPECK에서 사용한 Adder

# 덧셈기 성능

- 3가지 모두 구현하여 비교

```
def Adder(eng):

    b = eng.allocate_qureg(16)
    a = eng.allocate_qureg(16)
    c = eng.allocate_qubit()

    #value_xor(eng, b, 0xffff, 8)
    #value_xor(eng, a, 0xffff, 8)

    ADD16(eng, a, b, c) #33, 30, 62, 77
    #new_adder(eng, a, b, 16) #32, 30, 74, 75
    #improved_adder(eng, a, b, c, 16) #33, 29, 73, 26, 35

    #All(Measure) | b
    #Measure | c
    #for i in range(16):
    #    print(int(b[15-i]), end=' ')
```

```
Gate counts:
    Allocate : 33
    CCX : 30
    CX : 62
    Deallocate : 33

Depth : 77.
```

<기존>

```
Gate counts:
    Allocate : 32
    CCX : 30
    CX : 74
    Deallocate : 32

Depth : 75.
```

<Indocrypt>

```
Gate counts:
    Allocate : 33
    CCX : 29
    CX : 73
    Deallocate : 33
    X : 26

Depth : 35.
```

<이번 덧셈기>

# SPECK 병렬 덧셈 구현

- SPECK 라운드 함수

$$R_k(x,y) = (\boxed{(x \lll \alpha) + y} \oplus k, (y \ggg \beta) \oplus ((x \lll \alpha) + y) \oplus k) \qquad (4)$$

- SPECK 키 스케줄

$$\begin{aligned} l_{i+m-1} &= (\boxed{k_i + (l_i \lll \alpha)}) \oplus i \\ k_{i+1} &= (k_i \ggg \beta) \oplus l_{i+m-1} \end{aligned}$$

- 병렬 덧셈을 위해 라운드 키와 키 스케줄을 병행하는 **on-the-fly 방식** 사용
- $k_i$는 라운드 키로 사용되기 때문에 $\boxed{k_i + (l_i \lll \alpha)}$ 의 **덧셈 결과는 $l_i$에 저장**
  - $k_i$에 저장되면 라운드 함수의 덧셈을 병렬로 수행할 수 없음
- **라운드 함수(1/2) → 키 스케줄(1/2) → 라운드 함수(2/2) → 키 스케줄(2/2)** 의 순서로 하나의 라운드를 구성

# SPECK 병렬 덧셈 구현

**Algorithm 1:** Quantum circuit implementation for SPECK-32/64.

**Input:** 32-qubit block $(x, y)$, 64-qubit keywords $(k_0, l_0, l_1, l_2)$, Carry qubits $\boxed{c_0, c_1}$

2개의 캐리carry 큐비트 사용

**Output:** 32-qubit ciphertext $(x, y)$

**for** $i = 0$ to $r - 2$ **do**
  **Round function(1/2) :**
    $x \leftarrow x \lll 7$
    $\boxed{x \leftarrow \mathrm{ADD}(y, x, c_0)}$
  **Key schedule(1/2) :**
    $l_{i\%3} \leftarrow l_{i\%3} \lll 7$
    $\boxed{l_{i\%3} \leftarrow \mathrm{ADD}(k_0, l_{i\%3}, c_1)}$
  **Round function(2/2) :**
    $x \leftarrow \mathrm{CNOT16}(k_0, x)$
    $y \leftarrow y \ggg 2$
    $y \leftarrow \mathrm{CNOT16}(x, y)$
  **Key schedule(2/2) :**
    **for** $j = 0$ to $5$ **do** //Constant XOR
      **if** $(i \gg j)\&1$ **then**
        $l_{i\%3}[j] \leftarrow \mathrm{X}(l_{i\%3}[j])$
    $k_0 \leftarrow k_0 \ggg 2$
    $k_0 \leftarrow \mathrm{CNOT16}(l_{i\%3}, k_0)$
//Last round
**Round function(1/2) :**
  $x \leftarrow x \lll 7$
  $x \leftarrow \mathrm{ADD}(y, x, c_0)$
**Round function(2/2) :**
  $x \leftarrow \mathrm{CNOT16}(k_0, x)$
  $y \leftarrow y \ggg 2$
  $y \leftarrow \mathrm{CNOT16}(x, y)$
**return** $(x, y)$

$$R_k(x, y) = (\boxed{(x \lll \alpha) + y} \oplus k, (y \ggg \beta) \oplus ((x \lll \alpha) + y) \oplus k)$$

$$l_{i+m-1} = (\boxed{k_i + (l_i \lll \alpha)}) \oplus i$$
$$k_{i+1} = (k_i \ggg \beta) \oplus l_{i+m-1}$$

# 성능 평가

- Depth 측면에서 56% 성능 향상

**Table 2:** Quantum resources required for variants of SPECK [2]

| SPECK | Qubits used | Toffoli gates | CNOT gates | X gates | Depth |
|---|---|---|---|---|---|
| 32/64 | 96 | 1,290 | 4,222 | 42 | 1,694 |
| 48/72 | 120 | 1,978 | 6,462 | 42 | 2,574 |
| 48/96 | 144 | 2,070 | 6,762 | 45 | 2,691 |
| 64/96 | 160 | 3,162 | 10,318 | 54 | 4,082 |
| 64/128 | 192 | 3,286 | 10,722 | 57 | 4,239 |
| 96/96 | 192 | 5,170 | 16,854 | 60 | 6,636 |
| 96/144 | 240 | 5,358 | 17,466 | 64 | 6,873 |
| 128/128 | 256 | 7,938 | 25,862 | 75 | 10,144 |
| 128/192 | 320 | 8,190 | 26,682 | 80 | 10,461 |
| 128/256 | 384 | 8,442 | 27,502 | 81 | 10,778 |

**Indocrypt**

**Table 3:** Ours

| SPECK(Ours) | Qubits used | Toffoli gates | CNOT gates | X gates | Depth |
|---|---|---|---|---|---|
| 32/64 | 98 | 1,247 | 4,179 | 1,160 | 814 |
| 48/72 | 122 | 1,935 | 6,419 | 1,848 | 1,166 |
| 48/96 | 146 | 2,025 | 6,717 | 1,935 | 1,219 |
| 64/96 | 162 | 3,111 | 10,267 | 3,012 | 1,794 |
| 64/128 | 194 | 3,233 | 10,669 | 3,131 | 1,863 |
| 96/96 | 194 | 5,115 | 16,799 | 5,010 | 2,828 |
| 96/144 | 242 | 5,301 | 17,409 | 5,194 | 2,929 |
| 128/128 | 256 | 7,875 | 25,799 | 7,761 | 4,256 |
| 128/192 | 322 | 8,125 | 26,617 | 8,010 | 4,389 |
| 128/256 | 386 | 8,375 | 27,435 | 8,255 | 4,522 |

**Ours**

# 양자 후 보안 강도 평가

**Table 5:** Quantum resources required for exhaustive key search.

| SPECK | $r$ | Qubits | Total gates | Total depth | Cost | NIST security |
|---|---|---|---|---|---|---|
| 32/64 | 2 | 132 | $1.797 \cdot 2^{47}$ | $1.249 \cdot 2^{42}$ | $1.122 \cdot 2^{90}$ | Not achieved |
| 48/72 | 2 | 245 | $1.805 \cdot 2^{52}$ | $1.789 \cdot 2^{46}$ | $1.615 \cdot 2^{99}$ | |
| 48/96 | 2 | 293 | $1.888 \cdot 2^{64}$ | $1.87 \cdot 2^{58}$ | $1.766 \cdot 2^{123}$ | |
| 64/96 | 2 | 325 | $1.449 \cdot 2^{65}$ | $1.376 \cdot 2^{59}$ | $1.993 \cdot 2^{124}$ | |
| 64/128 | 2 | 389 | $1.505 \cdot 2^{81}$ | $1.429 \cdot 2^{75}$ | $1.075 \cdot 2^{157}$ | |
| 96/96 | 1 | 195 | $1.199 \cdot 2^{65}$ | $1.084 \cdot 2^{60}$ | $1.3 \cdot 2^{125}$ | |
| 96/144 | 2 | 485 | $1.233 \cdot 2^{90}$ | $1.123 \cdot 2^{84}$ | $1.385 \cdot 2^{174}$ | Level-1 |
| 128/128 | 1 | 257 | $1.842 \cdot 2^{81}$ | $1.632 \cdot 2^{76}$ | $1.503 \cdot 2^{158}$ | Not achieved |
| 128/192 | 2 | 645 | $1.888 \cdot 2^{114}$ | $1.683 \cdot 2^{108}$ | $1.589 \cdot 2^{223}$ | Level-1 |
| 128/256 | 2 | 773 | $1.945 \cdot 2^{146}$ | $1.734 \cdot 2^{140}$ | $1.687 \cdot 2^{287}$ | Level-3 |

Level 1 : $2^{170}$, Level 3 : $2^{233}$, Level 5 : $2^{298}$

감사합니다