Draper Adder

임세진

https://youtu.be/NUP6vqugMjU

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Contents

CryptoCraft LAB

01. Draper Adder

A Logarithmic-Depth Quantum Carry-Lookahead Adder

Thomas G. Draper* Samuel A. Kutin† Eric M. Rains‡

Carry Lookahead Adder

1. P-rounds: $\boxed{\text{Compute } p[i,j]}$ values into the ancillary space.
2. G-rounds: $\boxed{\text{Set } G[j] = g[i,j];}$ for each j, we choose a particular i value.[1]
3. C-rounds: $\boxed{\text{Set } G[j] = c_j.}$
4. P^{-1}-rounds: $\boxed{\text{Erase the work done in the } P\text{-rounds.}}$

주요 연산

1. P-rounds. For $t = 1$ to $\lfloor \log n \rfloor - 1$: for $1 \le m < \lfloor n/2^t \rfloor$:

$$P_t[m] \oplus= P_{t-1}[2m]P_{t-1}[2m+1].$$ 모든 ancilla 사용

2. G-rounds. For $t = 1$ to $\lfloor \log n \rfloor$: for $0 \le m < \lfloor n/2^t \rfloor$:

$$G[2^t m + 2^t] \oplus= G[2^t m + 2^{t-1}]P_{t-1}[2m+1].$$

3. C-rounds. For $t = \lfloor \log \frac{2n}{3} \rfloor$ down to 1: for $1 \le m \le \lfloor (n - 2^{t-1})/2^t \rfloor$:

$$G[2^t m + 2^{t-1}] \oplus= G[2^t m]P_{t-1}[2m].$$

4. P^{-1}-rounds. For $t = \lfloor \log n \rfloor - 1$ down to 1: for $1 \le m < \lfloor n/2^t \rfloor$:

$$P_t[m] \oplus= P_{t-1}[2m]P_{t-1}[2m+1].$$ 사용한 ancilla $|0\rangle$으로 복구

01. Draper Adder

4.1 Addition out of place

We would like to add two n-bit numbers, a and b, stored in arrays A and B. We need $n+1$ bits for the output, denoted by Z, and $n - w(n) - \lfloor \log n \rfloor$ ancillary bits, denoted by X. We assume that Z and X are initialized to zero. In the end, we want Z to contain the quantity $s = a + b$.

The key relation is that the sum s is equal to $a \oplus b \oplus c$, where c is the carry string. Hence, the key step in our algorithm is to compute c, using the technique of the previous section. We compute the carry string c_1 through c_n into the bits $Z[1]$ through $Z[n]$.

The out-of-place QCLA adder proceeds as follows:

1. For $0 \le i < n$, $Z[i+1] \mathrel{\oplus}= A[i]B[i]$. This sets $z_{i+1} = g[i, i+1]$. Step 1) $Z[1] = g[0, 1]$
2. For $1 \le i < n$, $B[i] \mathrel{\oplus}= A[i]$. This sets $B[i] = p[i, i+1]$ for $i > 0$, which is what we need to run our addition circuit. Step 2) $P_0[i] = P[i, i+1] = b[i]$
3. Run the circuit of Section 3, using X as ancillary space. Upon completion, $Z[i] = c_i$ for $i \ge 1$.
4. For $0 \le i < n$, $Z[i] \mathrel{\oplus}= B[i]$. Now, for $i > 0$, $Z[i] = a_i \oplus b_i \oplus c_i = s_i$. For $i = 0$, we have $Z[i] = b_i$.
5. Set $Z[0] \mathrel{\oplus}= A[0]$. For $1 \le i < n$, $B[i] \mathrel{\oplus}= A[i]$. This fixes $Z[0]$, and resets B to its initial value.

---

1. P-rounds. For $t = 1$ to $\lfloor \log n \rfloor - 1$: for $1 \le m < \lfloor n/2^t \rfloor$:

$$P_t[m] \mathrel{\oplus}= P_{t-1}[2m]P_{t-1}[2m+1].$$ $\boldsymbol{P_1[1] = P_0[2]P_0[3] = b[2]b[3]}$

2. G-rounds. For $t = 1$ to $\lfloor \log n \rfloor$: for $0 \le m < \lfloor n/2^t \rfloor$: $\boldsymbol{G[i] = g[i-1, i] = Z[i]}$

$$G[2^t m + 2^t] \mathrel{\oplus}= G[2^t m + 2^{t-1}]P_{t-1}[2m+1].$$ $\boldsymbol{G[2] = G[1]P_0[1] = G[1]B[1]}$

3. C-rounds. For $t = \lfloor \log \frac{2n}{3} \rfloor$ down to 1: for $1 \le m \le \lfloor (n - 2^{t-1})/2^t \rfloor$:

$$G[2^t m + 2^{t-1}] \mathrel{\oplus}= G[2^t m]P_{t-1}[2m].$$

P-round

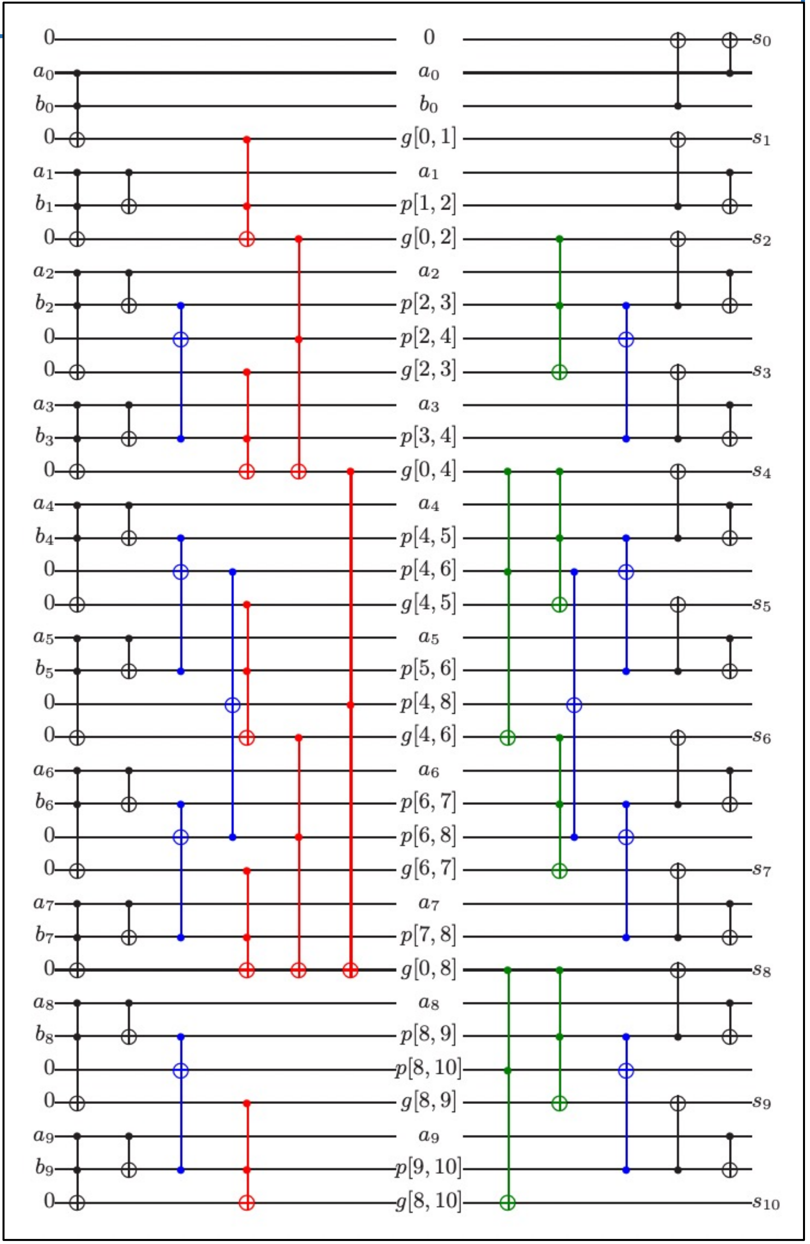
t=1 $P_1[m] \oplus= P_0[2m]P_0[2m+1]$
$= b[2m]b[2m+1]$

t=2 $P_2[m] = P_1[2m]P_1[2m+1]$
(2이상)
ancilla 접근★

G-round / C-round

t=1 → b에 접근

t≥2 → ancilla에 접근



01. Draper Adder

4.2 Addition in place

For the in-place circuit, we begin the same way as above: we compute the carry string c into $n-1$ ancillary bits (plus one output bit for the high bit). The total ancillary space required is $2n - w(n) - \lfloor \log n \rfloor - 1$. We then write the low n bits of the sum on top of b. The key new step is the erasure of the low $n-1$ bits of the carry string c.

The in-place QCLA adder proceeds as follows. We denote the $n-1$ ancillae which store the carry string as $Z[1], \ldots, Z[n-1]$, and the remaining ancillae as X. The output bit is labeled $Z[n]$.

1. For $0 \le i < n$, $Z[i+1] \oplus= A[i]B[i]$. This sets $Z[i+1] = g[i, i+1]$.
2. For $0 \le i < n$, $B[i] \oplus= A[i]$. This sets $B[i] = p[i, i+1]$ for $i > 0$. Also, $B[0] = s_0$.
3. Run the circuit of Section 3, using X as ancillary space. Upon completion, $Z[i] = c_i$ for $i \ge 1$.
4. For $1 \le i < n$, $B[i] \oplus= Z[i]$. Now $B[i] = s_i$.
5. For $0 \le i < n-1$, negate $B[i]$. Now B contains s'.
6. For $1 \le i < n-1$, $B[i] \oplus= A[i]$.
7. Run the circuit of Section 3 in reverse.[2] Upon completion, $Z[i+1] = a_i s'_i$ for $0 \le i < n-1$, and $B[i] = a_i \oplus s'_i$ for $1 \le i < n$.
8. For $1 \le i < n-1$, $B[i] \oplus= A[i]$.
9. For $0 \le i < n-1$, $Z[i+1] \oplus= A[i]B[i]$.
10. For $0 \le i < n-1$, negate $B[i]$.

[2] In Step 7, we actually reverse the $(n-1)$-bit adder.

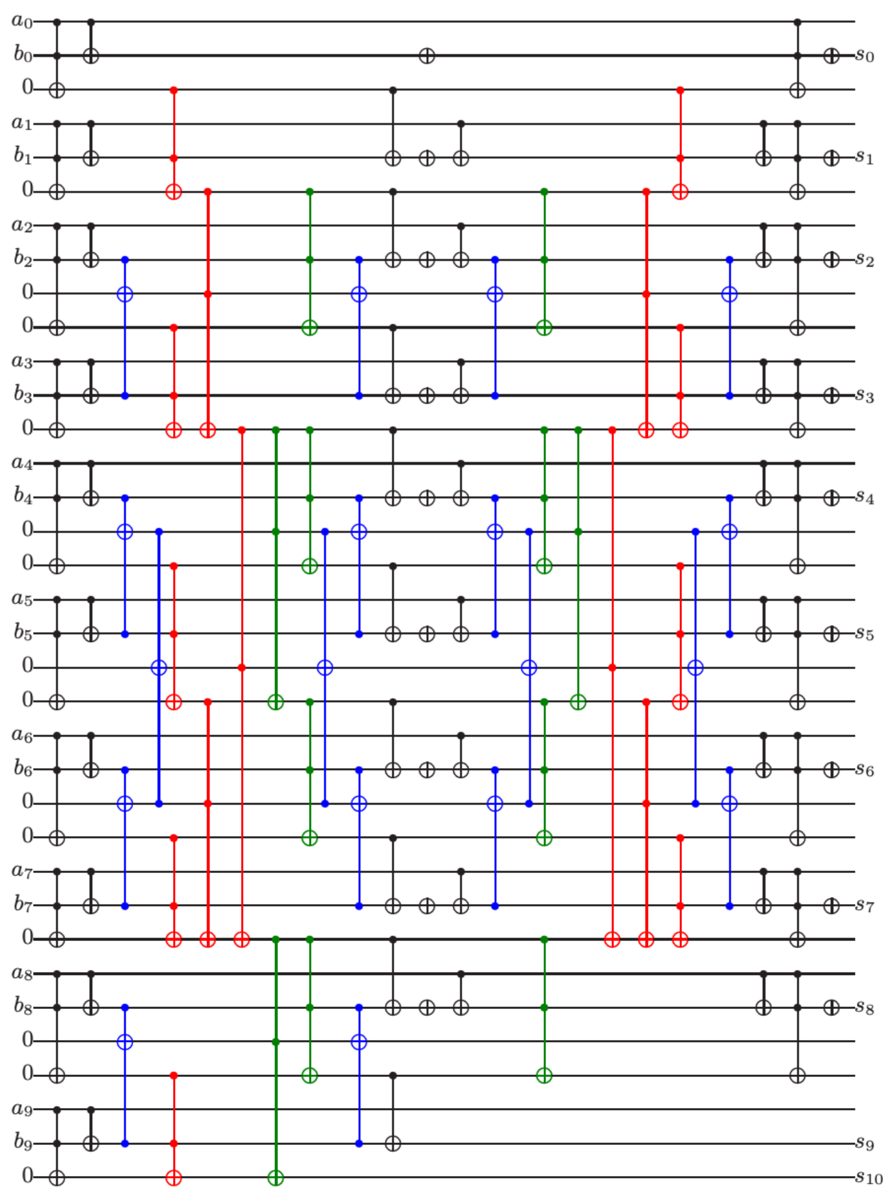
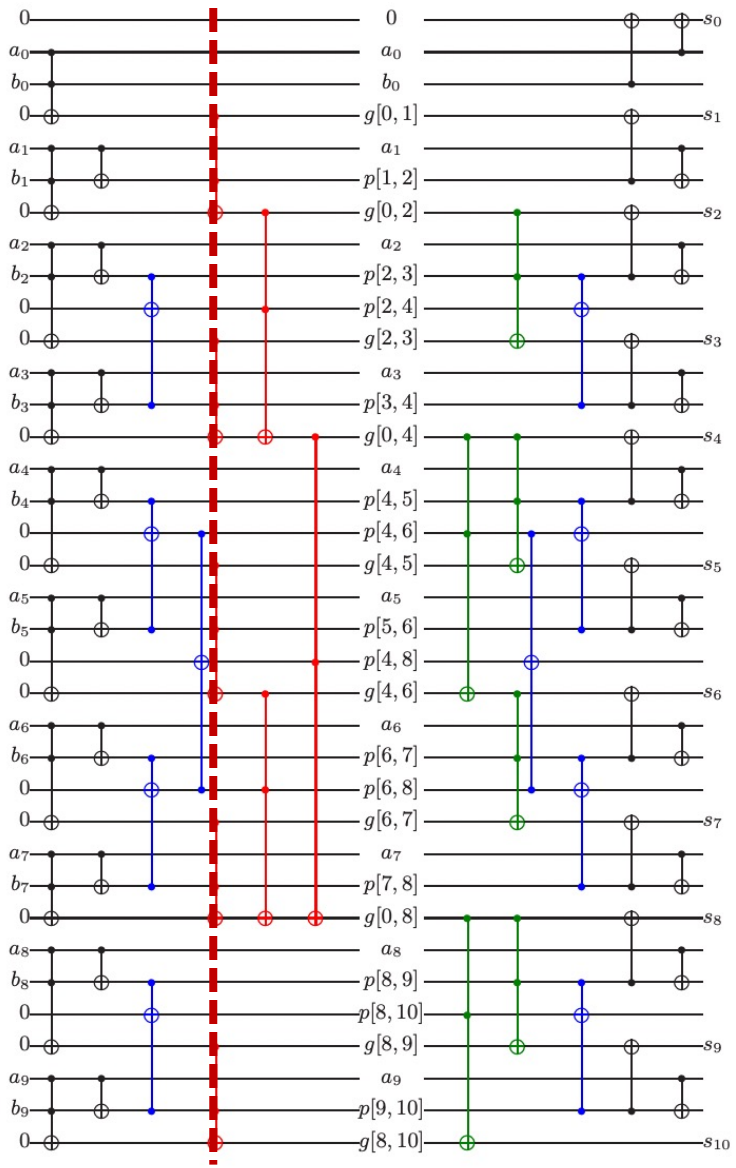


Figure 5: In-place QCLA adder for 10 bits. P-rounds and P^{-1}-rounds are shown in blue. G-rounds are red, and C-rounds are green.

02. Implementation



Out-of-place

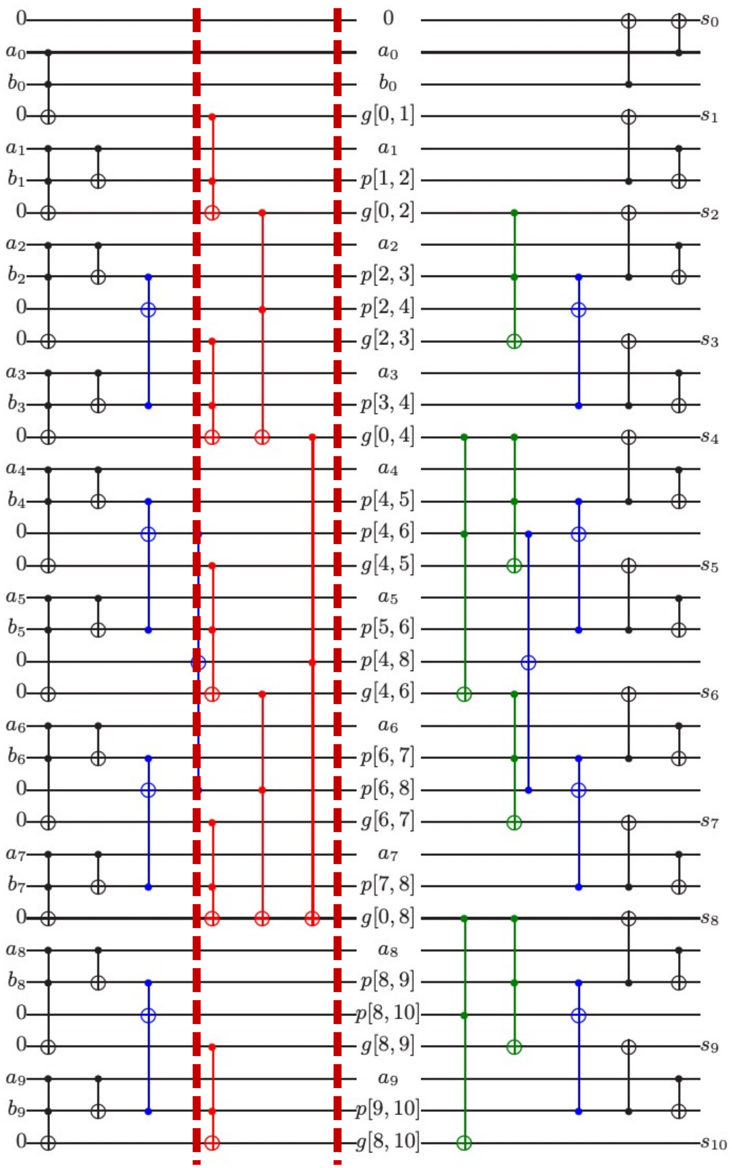
```
# Init round
for i in range(n):
    toffoli_gate(eng, a[i], b[i], z[i + 1])
for i in range(1,n):
    CNOT | (a[i], b[i])

# P-round
idx = 0 # ancilla idx
tmp = 0 # m=1일 때 idx 저장해두기
for t in range(1, int(log2(n))):
    pre = tmp  # (t-1)일 때의 첫번째 자리 저장
    for m in range(1, l(n, t)):
        if t == 1:  # B에 저장되어있는 애들로만 연산 가능
            toffoli_gate(eng, b[2 * m], b[2 * m + 1], ancilla[idx])
        else:  # t가 1보다 클 때는 ancilla에 저장된 애들도 이용해야함
            toffoli_gate(eng, ancilla[pre - 1 + 2 * m], ancilla[pre - 1 + 2 * m + 1], ancilla[idx])
        if m == 1:
            tmp = idx
        idx += 1
```

1. P-rounds. For $t = 1$ to $\lfloor \log n \rfloor - 1$: for $1 \le m < \lfloor n/2^t \rfloor$:

$$P_t[m] \oplus= P_{t-1}[2m]P_{t-1}[2m+1].$$

02. Implementation



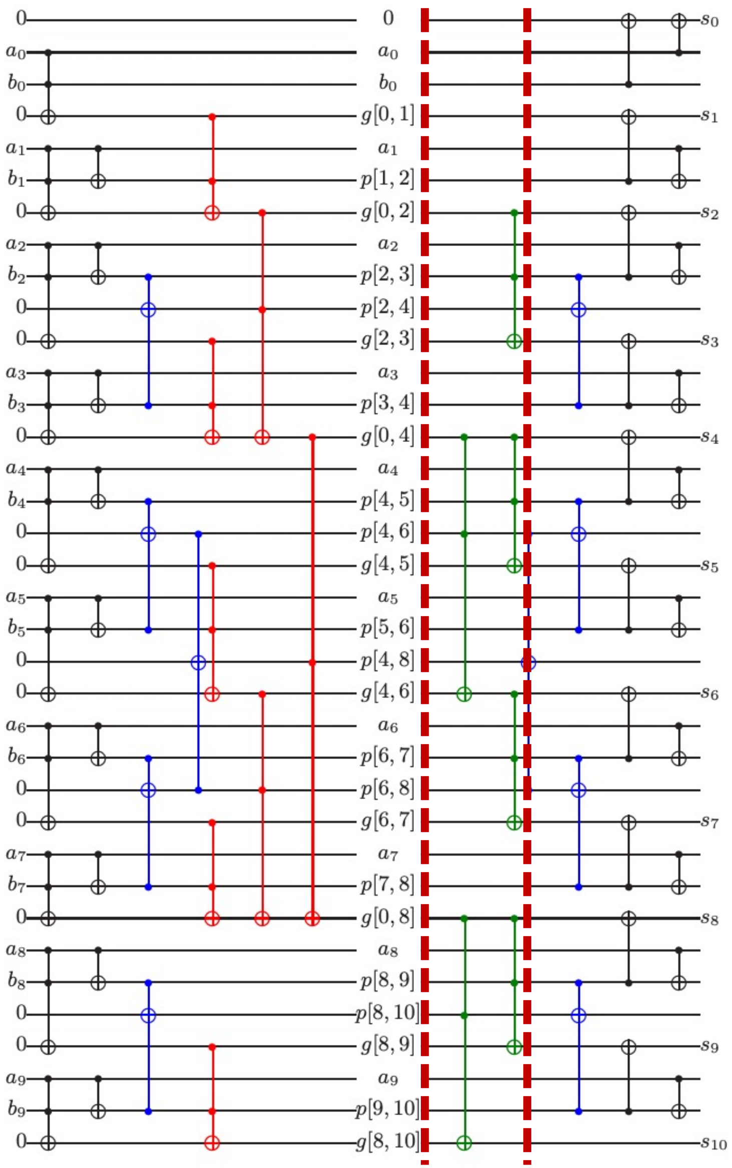
Out-of-place

```python
# G-round
pre = 0  # The number of cumulative p(t-1)
idx = 0  # ancilla idx
for t in range(1, int(log2(n)) + 1):
    for m in range(l(n, t)):
        if t == 1:  # B에 저장되어있는 애들로만 연산 가능
            toffoli_gate(eng, z[int(pow(2, t) * m + pow(2, t - 1))], b[2 * m + 1], z[int(pow(2, t) * (m + 1))])
        else:  # t가 1보다 클 때는 ancilla에 저장된 애들도 이용해야함
            toffoli_gate(eng, z[int(pow(2, t) * m + pow(2, t - 1))], ancilla[idx+2*m], z[int(pow(2, t) * (m + 1))])
    if t != 1:
        pre = pre + l(n, t-1) -1
        idx = pre
```

2. G-rounds. For $t = 1$ to $\lfloor \log n \rfloor$: for $0 \le m < \lfloor n/2^t \rfloor$:

$$G[2^t m + 2^t] \oplus\!= G[2^t m + 2^{t-1}] P_{t-1}[2m+1].$$

02. Implementation



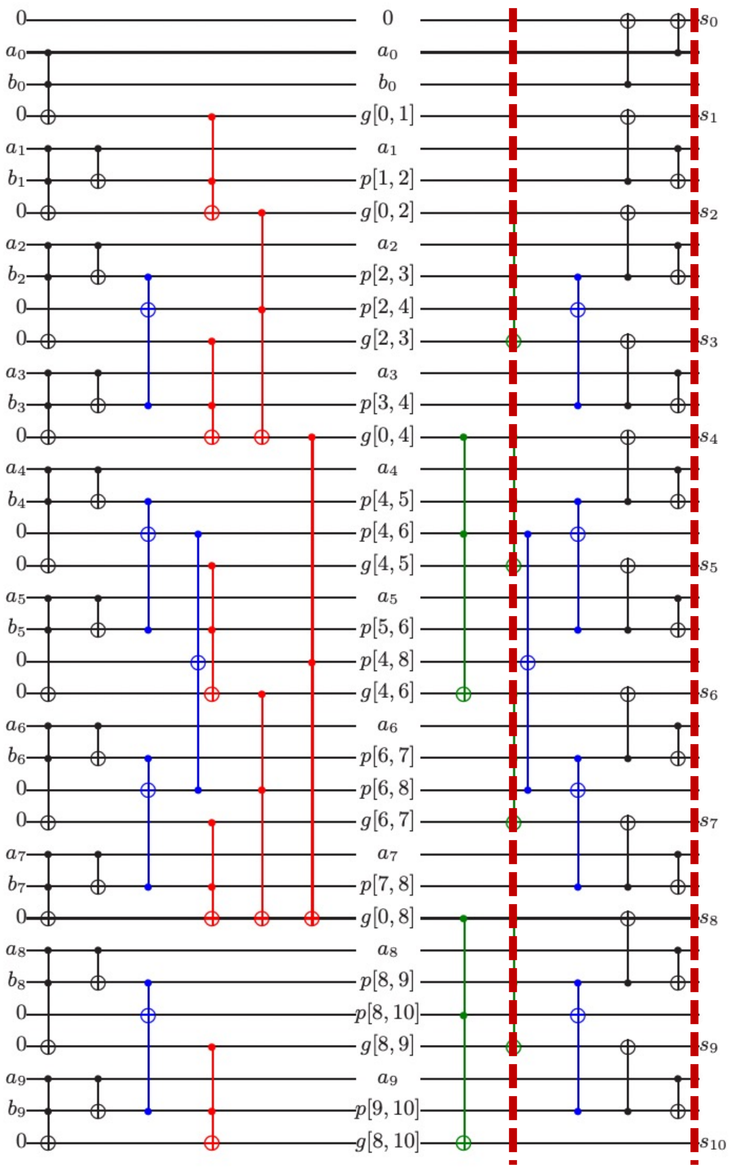
Out-of-place

```
# C-round
if int(log2(n)) - 1 == int(log2(2 * n / 3)): # p(t-1)까지 접근함
    iter = l(n, int(log2(n)) - 1) - 1 # 마지막 pt의 개수
else: # p(t)까지 접근함
    iter = 0
pre = 0  # (t-1)일 때의 첫번째 idx
for t in range(int(log2(2 * n / 3)), 0, -1):
    for m in range(1, l((n - pow(2, t-1)),t)+1):
        if t == 1:  # B에 저장되어있는 애들로만 연산 가능
            toffoli_gate(eng, z[int(pow(2, t) * m)], b[2 * m], z[int(pow(2, t) * m + pow(2, t - 1))])
        else:
            if m==1:
                iter += l(n, t - 1) - 1
                pre = length - 1 - iter
            toffoli_gate(eng, z[int(pow(2, t) * m)],
                         ancilla[pre + 2 * m], z[int(pow(2, t) * m + pow(2, t-1))])
```

3. C-rounds. For $t = \lfloor \log \frac{2n}{3} \rfloor$ down to 1: for $1 \le m \le \lfloor (n-2^{t-1})/2^t \rfloor$:

$$G[2^t m + 2^{t-1}] \oplus= G[2^t m] P_{t-1}[2m].$$

02. Implementation



Out-of-place

```
# P-inverse round
pre = 0  # (t-1)일 때의 첫번째 idx
iter = l(n, int(log2(n)) - 1) - 1  # 마지막 pt의 개수
iter2 = 0 # for idx
idx = 0
for t in reversed(range(1, int(log2(n)))):
    for m in range(1, l(n, t)):
        if t == 1:  # B에 저장되어있는 애들로만 연산 가능
            toffoli_gate(eng, b[2 * m], b[2 * m + 1], ancilla[m - t])
        else:  # t가 1보다 클 때는 ancilla에 저장된 애들도 이용해야함
            if m == 1:
                iter += l(n, t - 1) - 1  # p(t-1) last idx
                pre = length - iter
                iter2 += (l(n, t) - 1)
                idx = length - iter2
            toffoli_gate(eng, ancilla[pre - 1 + 2 * m], ancilla[pre - 1 + 2 * m + 1], ancilla[idx-1+m])
```

4. P^{-1}-rounds. For $t = \lfloor \log n \rfloor - 1$ down to 1: for $1 \le m < \lfloor n/2^t \rfloor$:

$$P_t[m] \oplus= P_{t-1}[2m]P_{t-1}[2m+1].$$

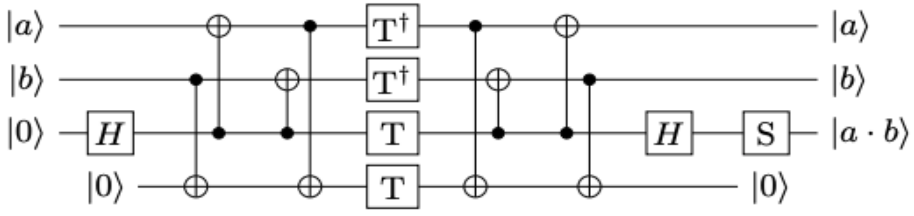
```
# Last round
for i in range(n):
    CNOT | (b[i], z[i])
CNOT | (a[0], z[0])
for i in range(1, n):
    CNOT | (a[i], b[i])
```

03. Future Work (1,2번 구현하여 다음주 세미나 진행 예정)

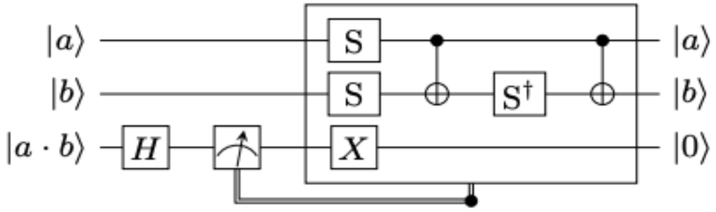
1. Generic Adder를 modular 2^{32} Adder로 구조 수정

2. Eurocrypt'20에서 제안된 **Quantum AND gate**를 Drapper Adder의 Toffoli gate 대신 적용하여 최적화
   - T-depth : 1
   - 한번의 AND 연산에 1개의 Ancilla 사용 (다시 $|0\rangle$으로 초기화 → 재활용 가능)

3. SHA2 구현 → 회로 최적화



(a) AND gate.



(b) AND† gate.

Quantum AND gate

04. Demo

- **올바르게 구현한 것이 맞는지 확인**해야하는 요소들

1) 덧셈 후 sum, ancilla, a, b 값 확인

2) 논문의 Overall depth와 비교

overall depth of the circuit is

$$\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 7,$$

Each step other than 3 and 7 has depth 1. By (5), the overall depth is

$$\lfloor \log n \rfloor + \lfloor \log(n-1) \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + \left\lfloor \log \frac{n-1}{3} \right\rfloor + 14,$$

3) N이 큰 수 일 때도 잘 동작하는지 확인

감사합니다 ☺