HIGH PERFORMANCE QUANTUM MODULAR MULTIPLIERS 논문리뷰

https://youtu.be/4EWO0-UFO-g

정보컴퓨터공학과 송경주

Basic multiplication: In-place, Out-of-place

- Out-of-place (일반적)

$$|y\rangle|0\rangle \rightarrow |y\rangle|Xy \bmod N\rangle$$

- In-place

$$|y\rangle\,|0\rangle \xrightarrow{(*X)_{fwd}} |y\rangle\,|Xy \bmod N\rangle \xrightarrow{\text{SWAP}} |Xy \bmod N\rangle\,|y\rangle \xrightarrow{(*X^{-1})_{rev}} |Xy \bmod N\rangle\,|0\rangle\,.$$

Rines, Rich, and Isaac Chuang. "High performance quantum modular multipliers." *arXiv preprint arXiv:1801.01081* (2018).

Basic multiplication: In-place, Out-of-place

- Out-of-place (일반적)

$$|y\rangle|0\rangle \rightarrow |y\rangle|Xy \bmod N\rangle$$

- In-place

$$|y\rangle\,|0\rangle \xrightarrow{(*X)_{fwd}} |y\rangle\,|Xy \bmod N\rangle \xrightarrow{\text{SWAP}} |Xy \bmod N\rangle\,|y\rangle \xrightarrow{(*X^{-1})_{rev}} |Xy \bmod N\rangle\,|0\rangle\,.$$

  - 입력 y를 결과값으로 덮어 써야함 → 연산과정에서 남은 garbage(y) 삭제

Division-based Modular Multiplication

- 일반 곱셈 결과 t = Xy을 연산 → Quantum Division (Q-DIV)으로 t mod N 계산

1. Multiplication

*Q-MAC : 누산 곱셈

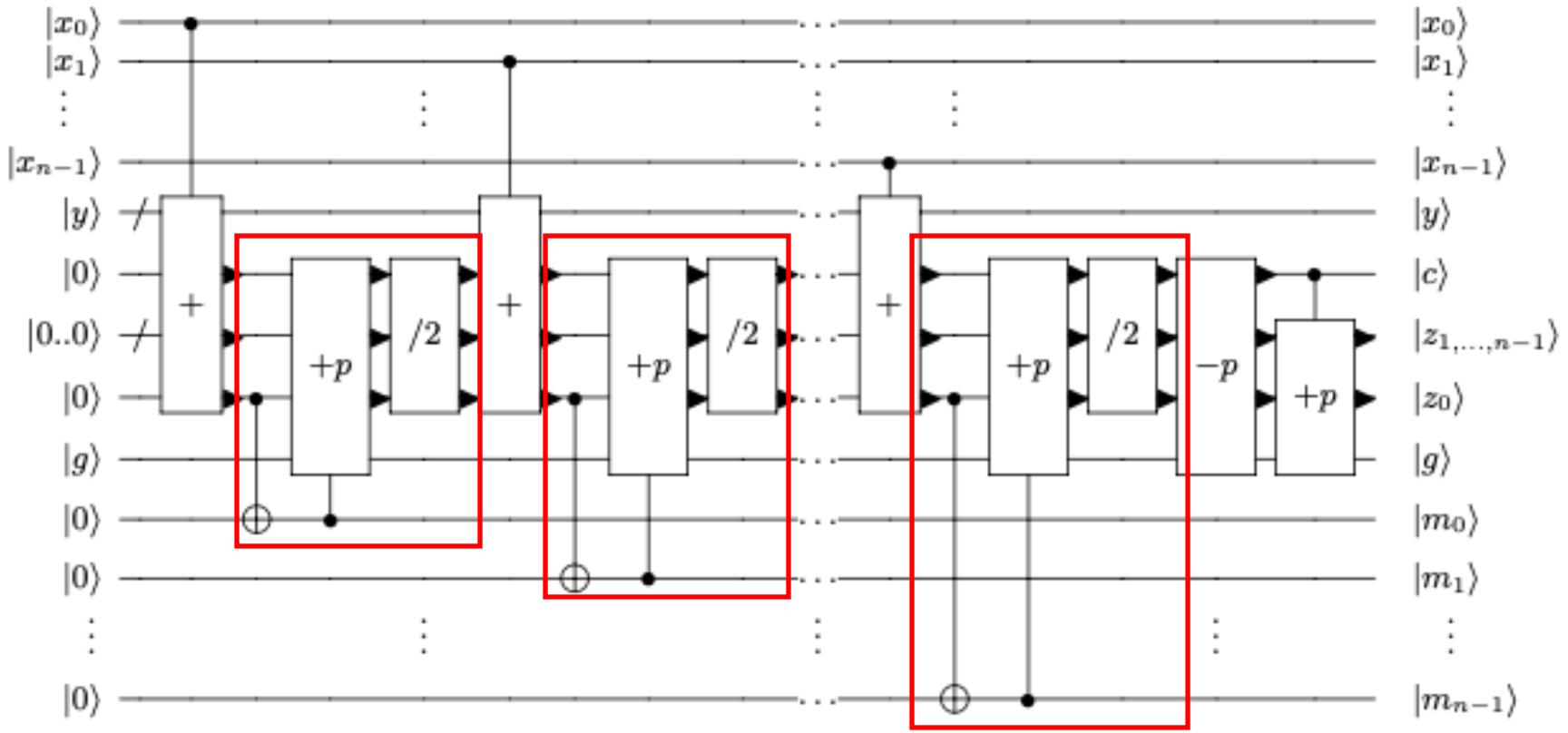
$$|0\rangle_{n+m}|y\rangle \xrightarrow{\text{Q-MAC}(X|N)} |t\rangle_{n+m}|y\rangle$$ (t = qN+r)

2. Division (Q-DIV)
   - t에 대해 N 빼기 반복 (반복 수: $2^k N$)
   - 결과의 부호(MSB)가 1이면 uncompute
   - 결과의 부호(MSB)가 0이면 유지
   - t= $|r\rangle|q\rangle$ ($|r\rangle = m$ 비트, $|q\rangle = n$비트)

3. Uncomputation
   - $|r\rangle|q\rangle|y\rangle$ → 여기서 q 는 garbage
   - qN+r = t 로 복구한 뒤, Q-MAC 역연산 실행 :$|t\rangle|q\rangle|y\rangle$ → $|0\rangle|q\rangle|y\rangle$
   - q 정리(clean) : $|0\rangle|q\rangle|y\rangle$ → $|r\rangle|0\rangle|y\rangle$

Montgomery Modular Multiplication

- 몽고메리 기법은 나눗셈을 직접 하지 않고, shift 연산으로 대체
- 각 단계에서 “LSB 확인” → “LSB가 1일 때, p를 더함” → “/2 (shift)”



Montgomery multiplication [1]

[1] Roetteler, Martin, et al. "Quantum resource estimates for computing elliptic curve discrete logarithms." International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer International Publishing, 2017.

Controlled Modular Multiplication (In-place)

- Controlled SWAP (Fredkin) – 일반적으로 in-place 에서 사용
  1. 제어 큐비트 q = 1 일 때, 0 상태의 ancilla qubit과 피연산자 y SWAP (Controlled SWAP)
  2. X 곱하기 → 기존: 0 + y*X
     → SWAP 후: y + 0*X

  3. 결과 → 기존: yX
     → SWAP 후 : y

$$|y\rangle\,|0\rangle \xrightarrow{\text{SWAP}} |0\rangle\,|y\rangle \xrightarrow{*X} |0\rangle\,|y+0*X\rangle = |0\rangle\,|y\rangle$$

Controlled Modular Multiplication

- 제어 큐비트 기준으로 Control
  - 제어 큐비트 q를 기준으로 Controlled arithmetic을 진행함
  - 즉, 게이트 승격? 발생함 ex) CNOT(x,y) → Toffoli(q, x, y)

- Swap 기반 (Fredkin)
  - 연산을 그대로 유지해서 자원량이 전체적으로 줆
  - Control 큐비트 q에 따라 입력을 clean ancilla 0 와 SWAP 함.
    - q = 0 : SWAP(x, 0), CNOT (x,y) :: x가 0 상태이므로 q=0일 때 CNOT (x, y)가 동작하지 않음
    - q = 1 : CNOT (x, y)

Controlled SWAP

- CNOT (b, a)
- Toffoli (c, a, b)
- CNOT (b, a)

(c, a, b)	CNOT(b, a)	Toffoli(c, a, b)	CNOT(b, a)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	**(0, 0, 0)**
(0, 0, 1)	(0, 1, 1)	(0, 1, 1)	**(0, 0, 1)**
(0, 1, 1)	(0, 0, 1)	(0, 0, 1)	**(0, 1, 1)**
(1, 0, 0)	(1, 0, 0)	(1, 0, 0)	**(1, 0, 0)**
(1, 0, 1)	(1, 1, 1)	(1, 1, 0)	**(1, 1, 0)**
(1, 1, 1)	(1, 0, 1)	(1, 0, 1)	**(1, 1, 1)**

Q & A