Quantum AND gate

임세진

https://youtu.be/lQvWw6-l47g

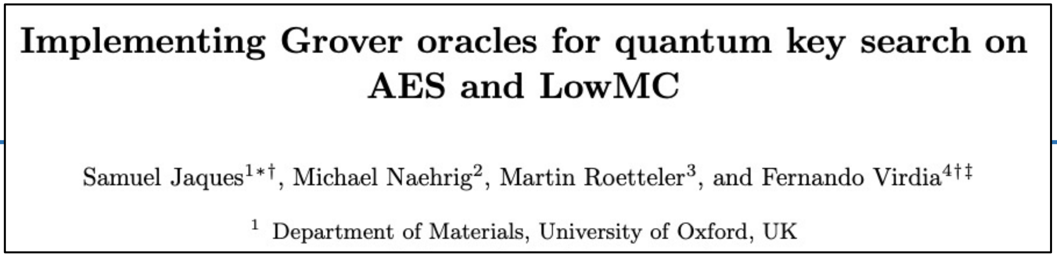
HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Contents

CryptoCraft LAB

01. Quantum AND gate


Implementing Grover oracles for quantum key search on AES and LowMC

Samuel Jaques[1*†], Michael Naehrig[2], Martin Roetteler[3], and Fernando Virdia[4†‡]

[1] Department of Materials, University of Oxford, UK

- Eurocrypt'20에서 제안된 Quantum AND gate
- [Toffoli Decomposition] + [Gidney logical AND] → T-depth = 1
- AND dagger 연산 시 T gate가 사용되지 않아 T-count, T-depth 비용 절감

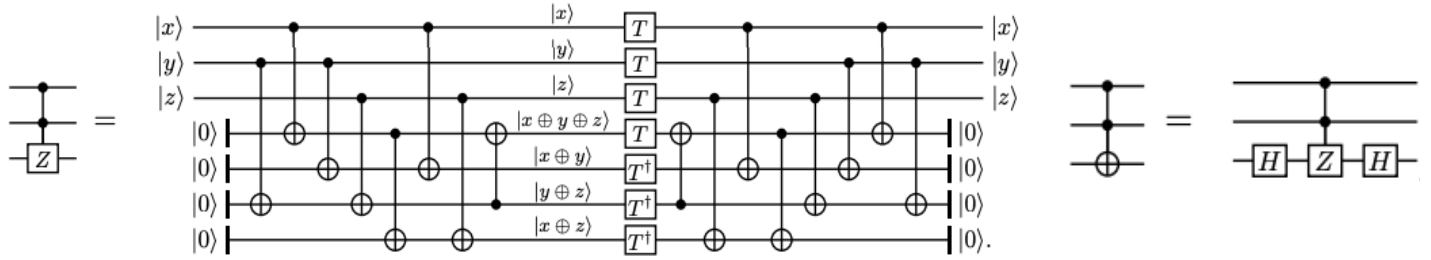
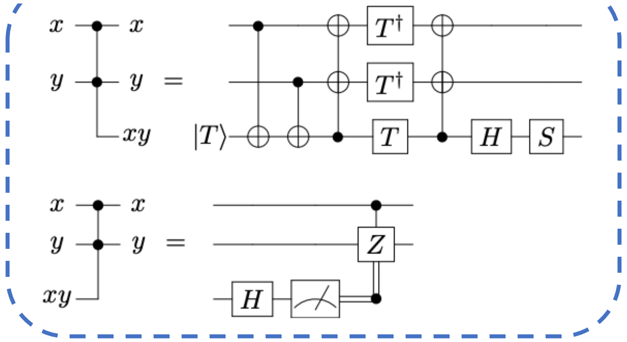
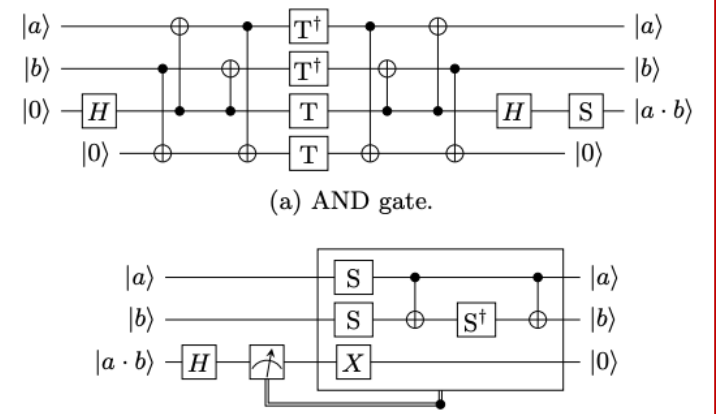


Figure 1: *T*-depth 1 representation of the Toffoli gate

Ancilla를 사용해서 T-depth를 1로 만듦
Cost : ancillas < T-gate

Ancilla 1개를 사용해서 T-depth를 2로 만듦





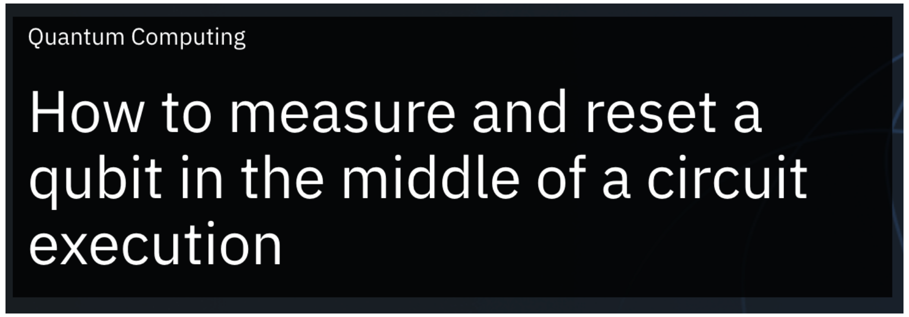
(a) AND gate.

(b) AND† gate.

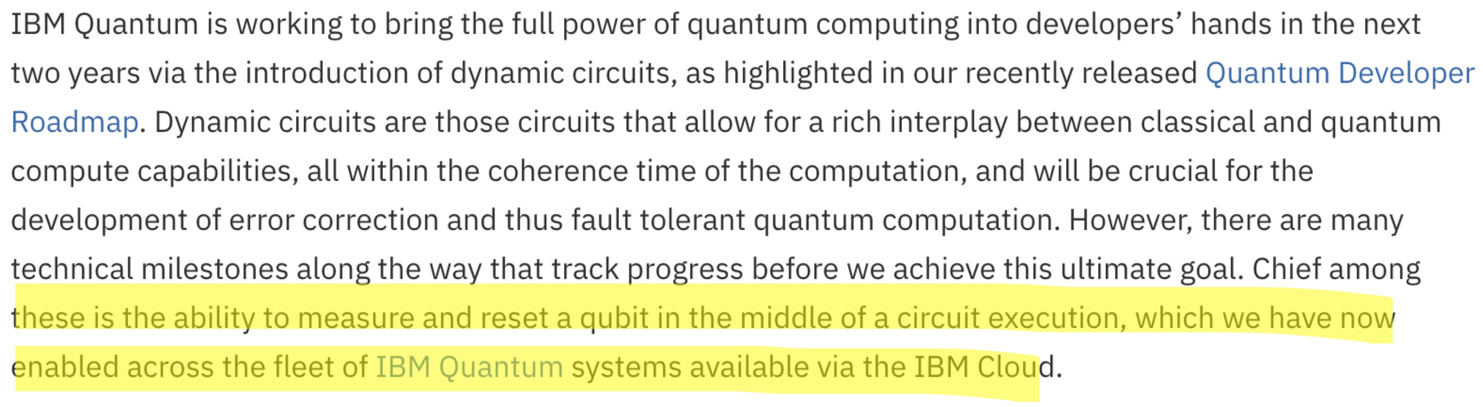
Ancilla 1개 (다시 0으로 초기화됨 → 재활용 가능)
➔ T-depth를 1로 만듦

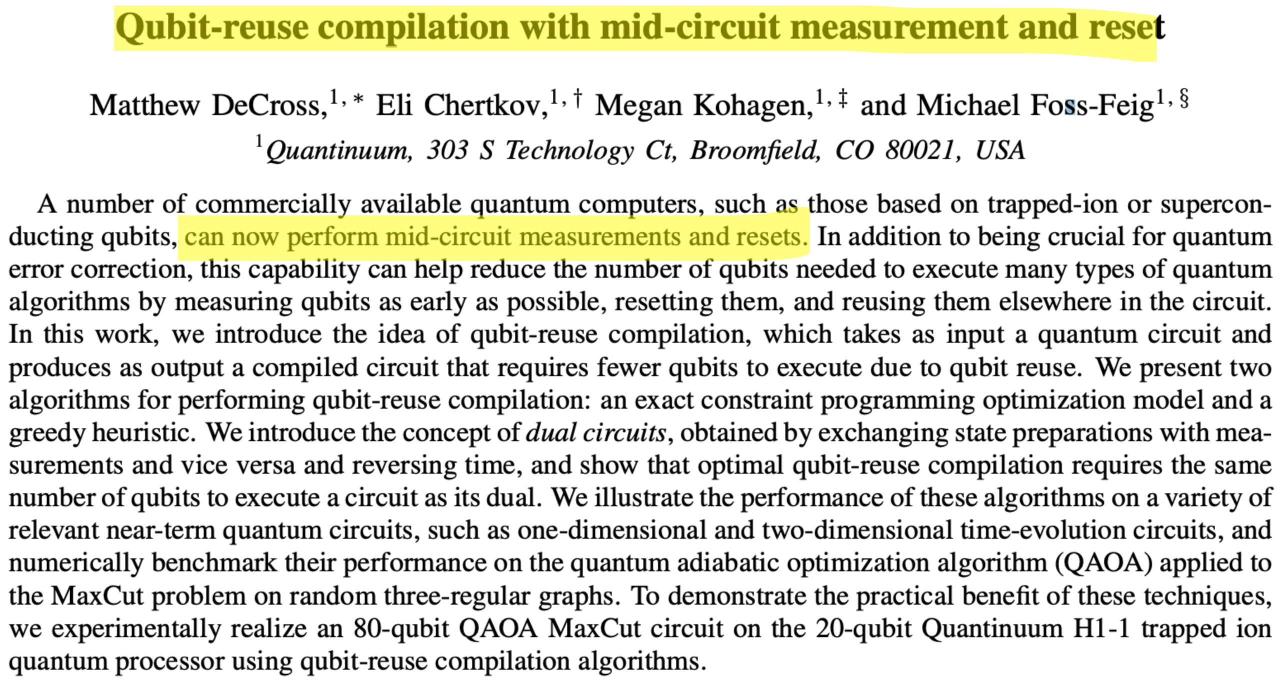
[Toffoli Decomposition] https://arxiv.org/pdf/1210.0974.pdf
[Gidney logical AND] https://quantum-journal.org/papers/q-2018-06-18-74/pdf/

01. Quantum AND gate


Quantum Computing

How to measure and reset a qubit in the middle of a circuit execution


IBM Quantum is working to bring the full power of quantum computing into developers' hands in the next two years via the introduction of dynamic circuits, as highlighted in our recently released Quantum Developer Roadmap. Dynamic circuits are those circuits that allow for a rich interplay between classical and quantum compute capabilities, all within the coherence time of the computation, and will be crucial for the development of error correction and thus fault tolerant quantum computation. However, there are many technical milestones along the way that track progress before we achieve this ultimate goal. Chief among these is the ability to measure and reset a qubit in the middle of a circuit execution, which we have now enabled across the fleet of IBM Quantum systems available via the IBM Cloud.


Qubit-reuse compilation with mid-circuit measurement and reset

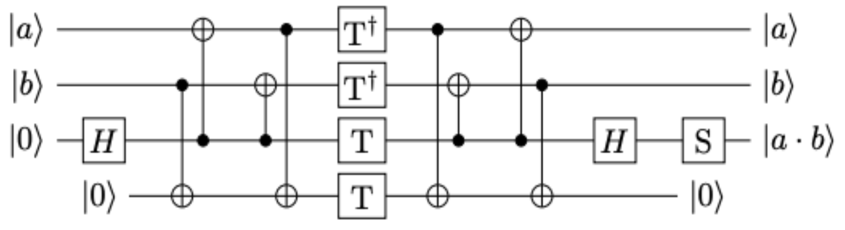
Matthew DeCross,[1,*] Eli Chertkov,[1,†] Megan Kohagen,[1,‡] and Michael Foss-Feig[1,§]

[1] *Quantinuum, 303 S Technology Ct, Broomfield, CO 80021, USA*

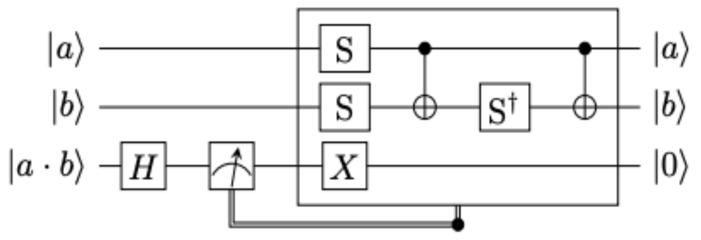
A number of commercially available quantum computers, such as those based on trapped-ion or superconducting qubits, can now perform mid-circuit measurements and resets. In addition to being crucial for quantum error correction, this capability can help reduce the number of qubits needed to execute many types of quantum algorithms by measuring qubits as early as possible, resetting them, and reusing them elsewhere in the circuit. In this work, we introduce the idea of qubit-reuse compilation, which takes as input a quantum circuit and produces as output a compiled circuit that requires fewer qubits to execute due to qubit reuse. We present two algorithms for performing qubit-reuse compilation: an exact constraint programming optimization model and a greedy heuristic. We introduce the concept of *dual circuits*, obtained by exchanging state preparations with measurements and vice versa and reversing time, and show that optimal qubit-reuse compilation requires the same number of qubits to execute a circuit as its dual. We illustrate the performance of these algorithms on a variety of relevant near-term quantum circuits, such as one-dimensional and two-dimensional time-evolution circuits, and numerically benchmark their performance on the quantum adiabatic optimization algorithm (QAOA) applied to the MaxCut problem on random three-regular graphs. To demonstrate the practical benefit of these techniques, we experimentally realize an 80-qubit QAOA MaxCut circuit on the 20-qubit Quantinuum H1-1 trapped ion quantum processor using qubit-reuse compilation algorithms.

01. Quantum AND gate

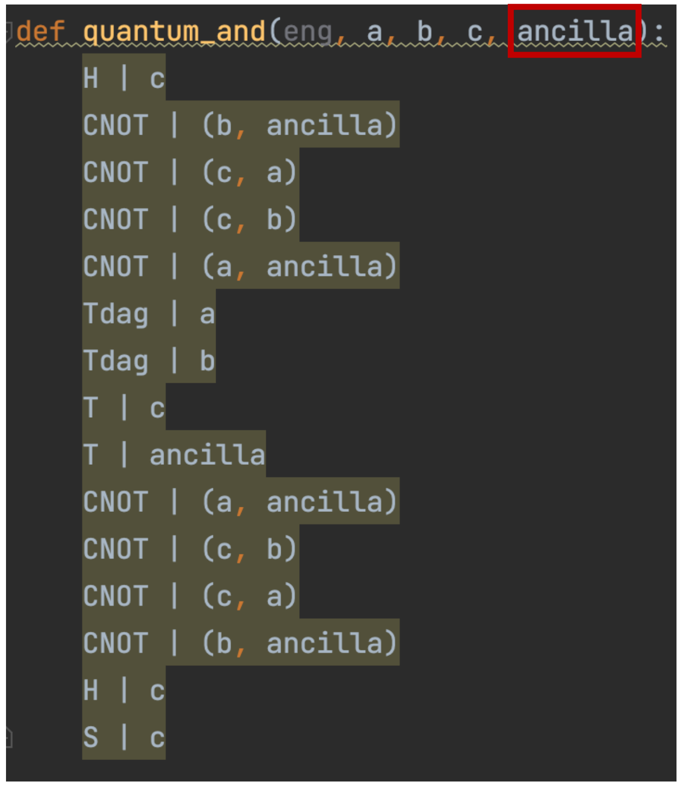
- Quantum AND gate 구현



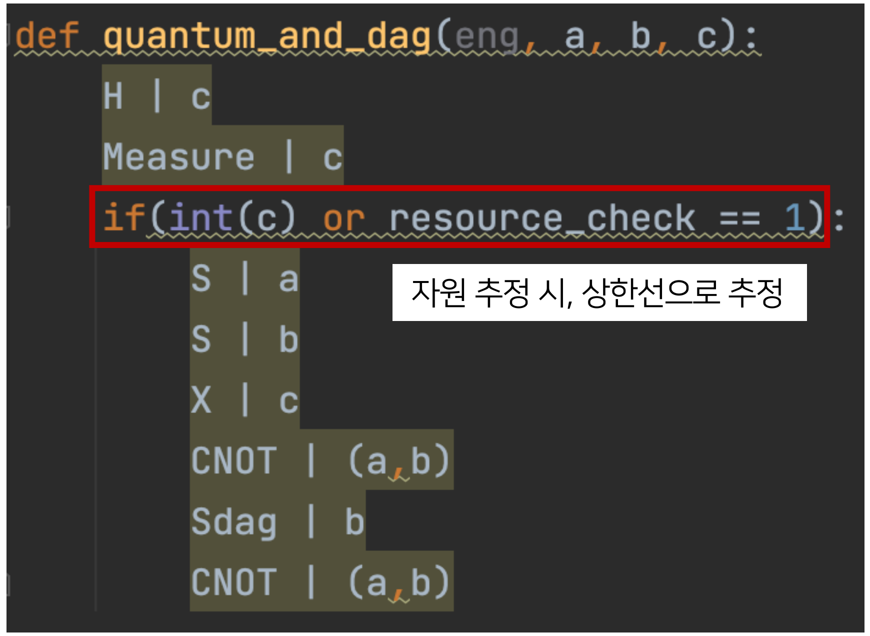
(a) AND gate.



(b) AND† gate.



```
def quantum_and(eng, a, b, c, ancilla):
    H | c
    CNOT | (b, ancilla)
    CNOT | (c, a)
    CNOT | (c, b)
    CNOT | (a, ancilla)
    Tdag | a
    Tdag | b
    T | c
    T | ancilla
    CNOT | (a, ancilla)
    CNOT | (c, b)
    CNOT | (c, a)
    CNOT | (b, ancilla)
    H | c
    S | c
```



```
def quantum_and_dag(eng, a, b, c):
    H | c
    Measure | c
    if(int(c) or resource_check == 1):
        S | a
        S | b
        X | c
        CNOT | (a,b)
        Sdag | b
        CNOT | (a,b)
```

자원 추정 시, 상한선으로 추정

02. Draper Adder (Replace Toffoli → Quantum AND gate)

- **Quantum AND gate 적용 시 고려할 사항**

1) 모든 Toffli gate 대체 가능

→ Quantum AND gate는 대상 큐비트가 임의의 상태가 아닌 |0> 상태에 있다고 가정한다는 점을 제외하면 Toffoli 게이트와 동일한 기능

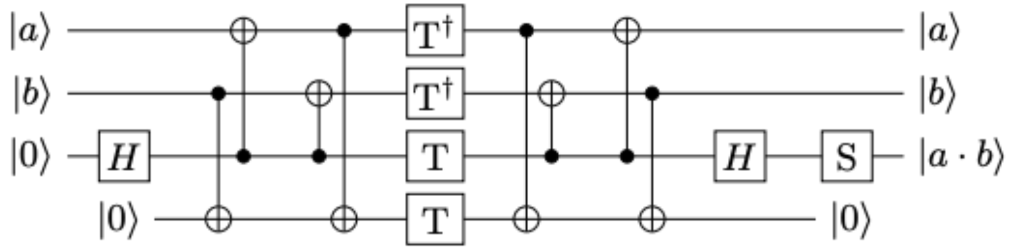
2) 특히 이점이 생기는 경우 : <토폴리 연산 + 토폴리 연산 해제> 쌍 형태일 때 Dagger 연산으로 대체 가능

→ 원래는 동일한 토폴리 연산이 두 번 적용됨 (2번의 자원 사용)

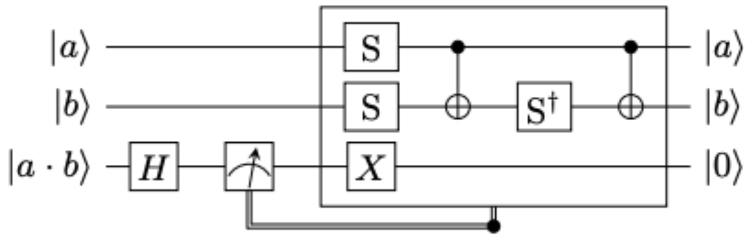
→ 쌍 형태이면, 해제 연산은 Dagger 연산으로 대체 가능 (T gate가 사용되지 않음)

3) AND 연산 후에 Ancilla를 |0⟩로 초기화해주므로 재활용 가능 → Toffli gate 병렬처리 고려해서 처리

→ 이때, AND dagger는 Ancilla 사용 X, AND 연산만 신경쓰면 됨



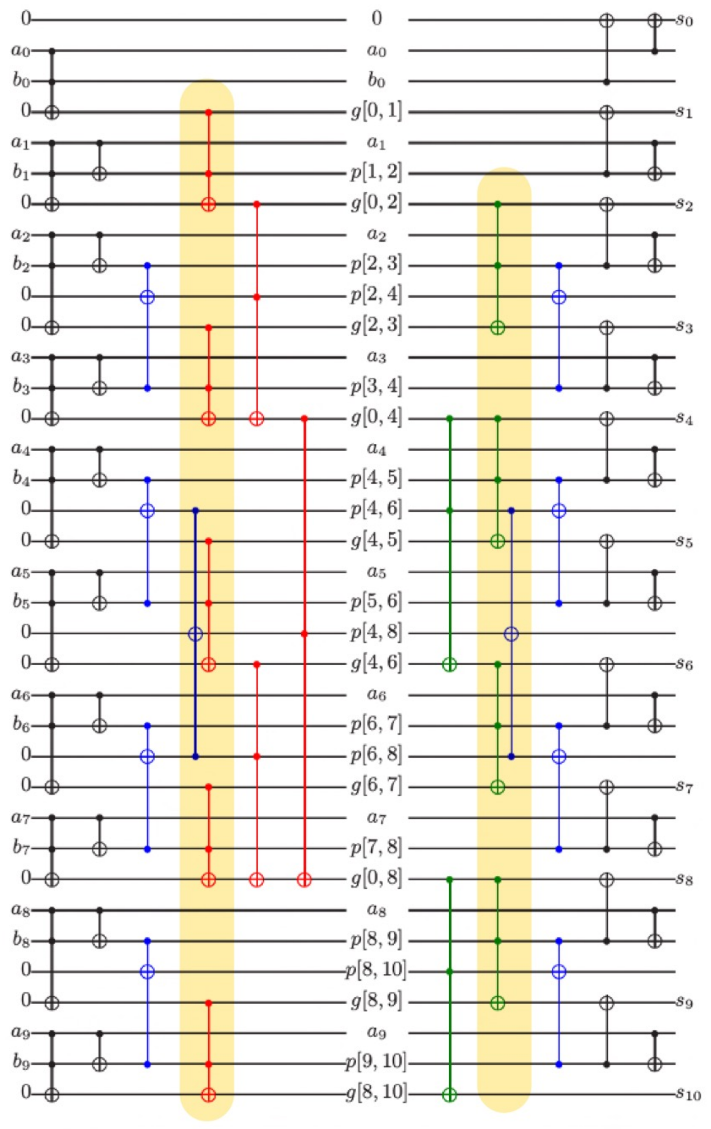
(a) AND gate.



(b) AND† gate.

02. Draper Adder (Replace Toffoli → Quantum AND gate)

Out of place



- 앞의 Toffoli만 Ancilla 병렬처리에 신경써주면 됨

  → C라운드는 AND gate, P-inv 라운드는 AND dagger gate를 사용하므로

- 병렬처리

  1. Init 라운드에서 n개의 Ancilla 필요

     → 라운드는 병렬로 수행되므로 총 n개의 Ancilla만 필요

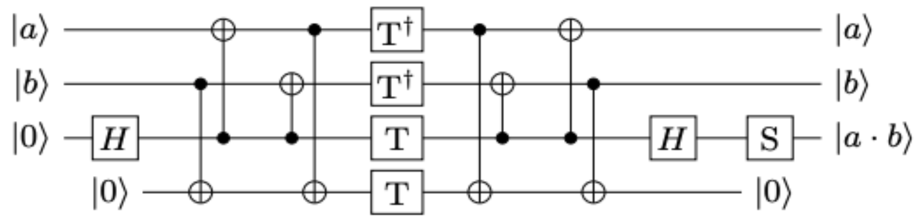
  3. P라운드와 G라운드 병렬처리 시, Ancilla 순서대로 할당

     → Ancilla 배열의 index++ 해서 계속 할당

     → 이때 index = (index + 1) % n 을 해주어 범위를 벗어나지 않게 조정

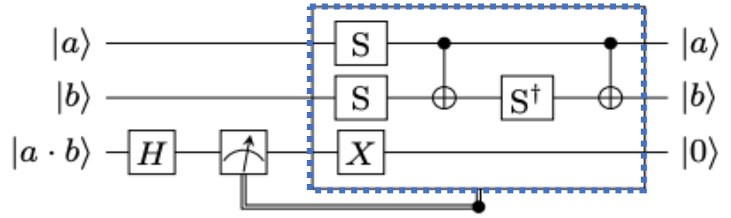
02. Draper Adder (Replace Toffoli → Quantum AND gate)

- Out of place



(a) AND gate.

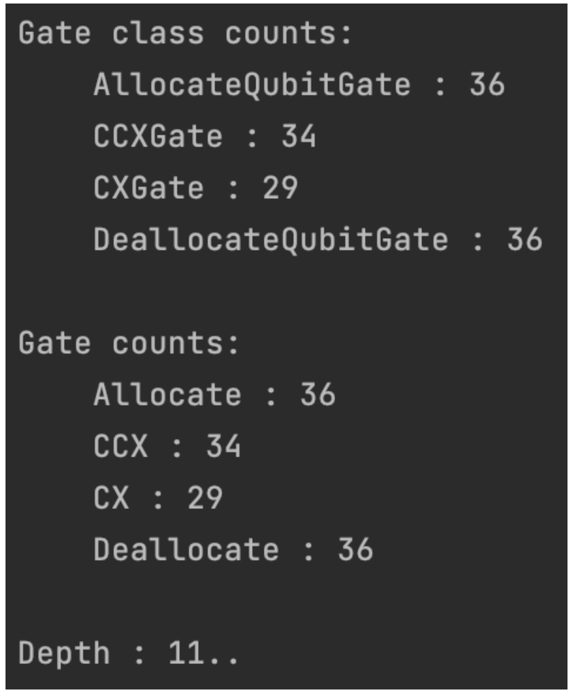
자원 추정 시, 상한선으로 추정



(b) AND† gate.

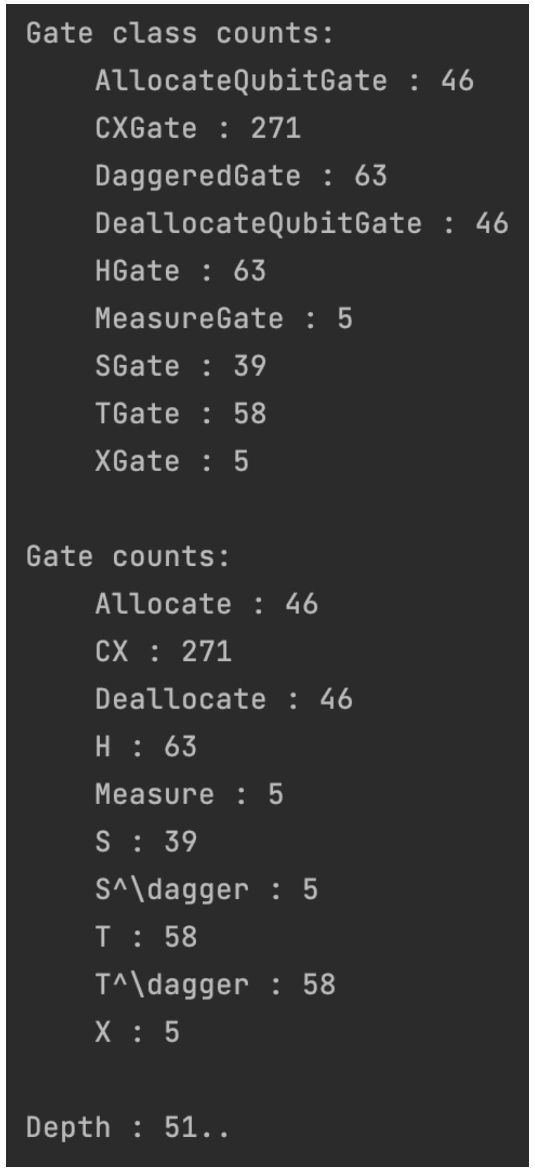
AND – 2T + 2T′ + 2H + S + 8CNOT

AND dag - H + M + 2S + X + S′ + 2CNOT



```
Gate class counts:
    AllocateQubitGate : 36
    CCXGate : 34
    CXGate : 29
    DeallocateQubitGate : 36

Gate counts:
    Allocate : 36
    CCX : 34
    CX : 29
    Deallocate : 36

Depth : 11..
```

적용 후



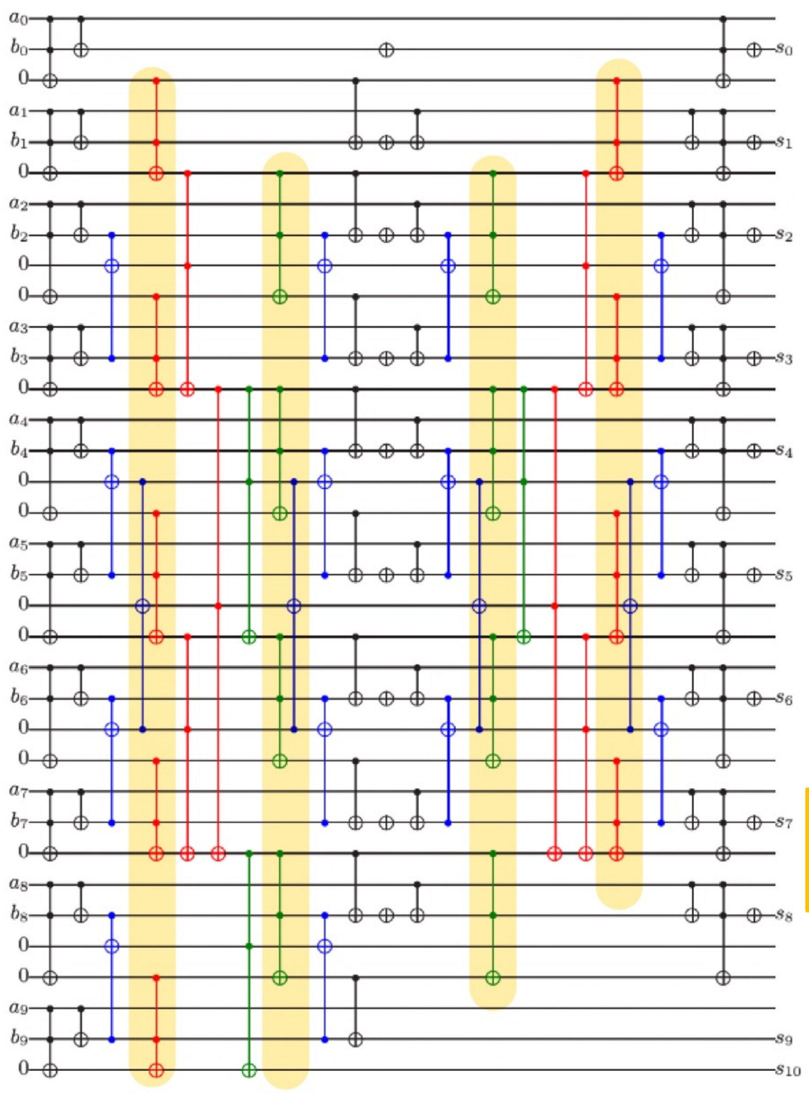
```
Gate class counts:
    AllocateQubitGate : 46
    CXGate : 271
    DaggeredGate : 63
    DeallocateQubitGate : 46
    HGate : 63
    MeasureGate : 5
    SGate : 39
    TGate : 58
    XGate : 5

Gate counts:
    Allocate : 46
    CX : 271
    Deallocate : 46
    H : 63
    Measure : 5
    S : 39
    S^\dagger : 5
    T : 58
    T^\dagger : 58
    X : 5

Depth : 51..
```

Toffoli : 34 → AND : 10+5+8+6 = 29
AND dag : 5

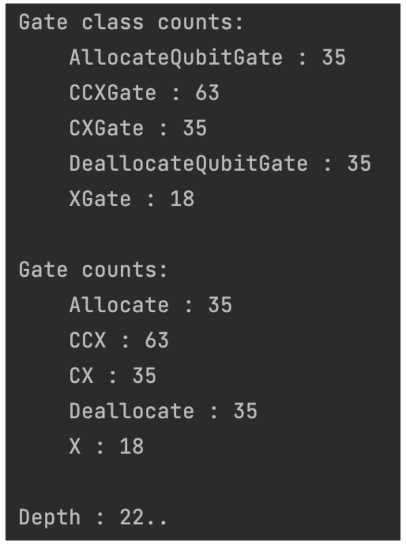
< n=10일 때 자원 추정 결과 >
1. Qubit 수 +10
2. T, T′, H, S, M 모두 추가된 AND 연산만큼 증가
3. CNOT + 242 (8x29 + 2x5)

02. Draper Adder (Replace Toffoli → Quantum AND gate)

In place

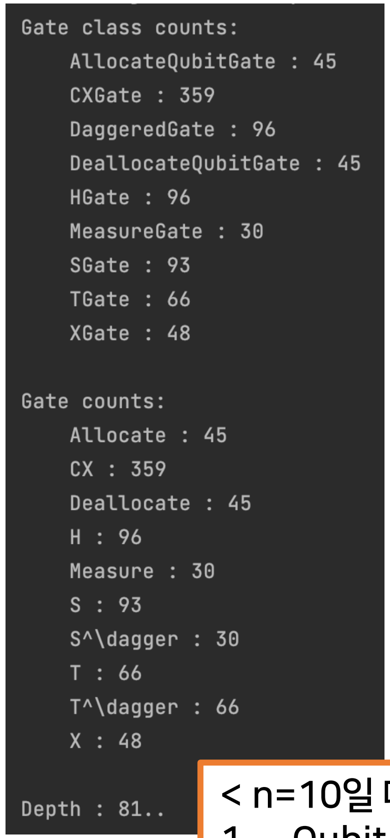


- 1, 3번째 Toffoli만 Ancilla 병렬처리에 신경써주면 됨
- 병렬처리는 Out of place와 같은 방식으로 구현함



```
Gate class counts:
    AllocateQubitGate : 35
    CCXGate : 63
    CXGate : 35
    DeallocateQubitGate : 35
    XGate : 18

Gate counts:
    Allocate : 35
    CCX : 63
    CX : 35
    Deallocate : 35
    X : 18

Depth : 22..
```

적용 후



```
Gate class counts:
    AllocateQubitGate : 45
    CXGate : 359
    DaggeredGate : 96
    DeallocateQubitGate : 45
    HGate : 96
    MeasureGate : 30
    SGate : 93
    TGate : 66
    XGate : 48

Gate counts:
    Allocate : 45
    CX : 359
    Deallocate : 45
    H : 96
    Measure : 30
    S : 93
    S^\dagger : 30
    T : 66
    T^\dagger : 66
    X : 48

Depth : 81..
```

Toffoli : 63 → AND : 10+5+8+6+4 = 33
AND dag : 5+5+7+4+9 = 30

AND – 2T + 2T′ + 2H + S + 8CNOT

AND dag - H + M + 2S + X + S′ + 2CNOT

< n=10일 때 자원 추정 결과 >
1. Qubit 수 +10
2. T, T′, H, S, M 모두 추가된 AND 연산만큼 증가
3. CNOT + 324 (8x33 + 2x30)

감사합니다