Quantum Carry-save Adder

장경배

https://youtu.be/9fpshCktcRM

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Project

Dear Anubhab,

Here are few discussion basis for tomorrow. You may share these with the Korean team.

Arithmetic for Finance

- f_k(s) = max (s – k, 0) → this is basically adder with 2's complement, and then checking the sign-bit. If negative then 0 is ans.
- g(x) = max (S_0 exp (\sigma x + c) – k, 0) → this can be compared with the basic modular exponentiation circuits used for Shor's. Quite a few references are there - https://arxiv.org/abs/1207.0511 , https://arxiv.org/abs/1202.6614

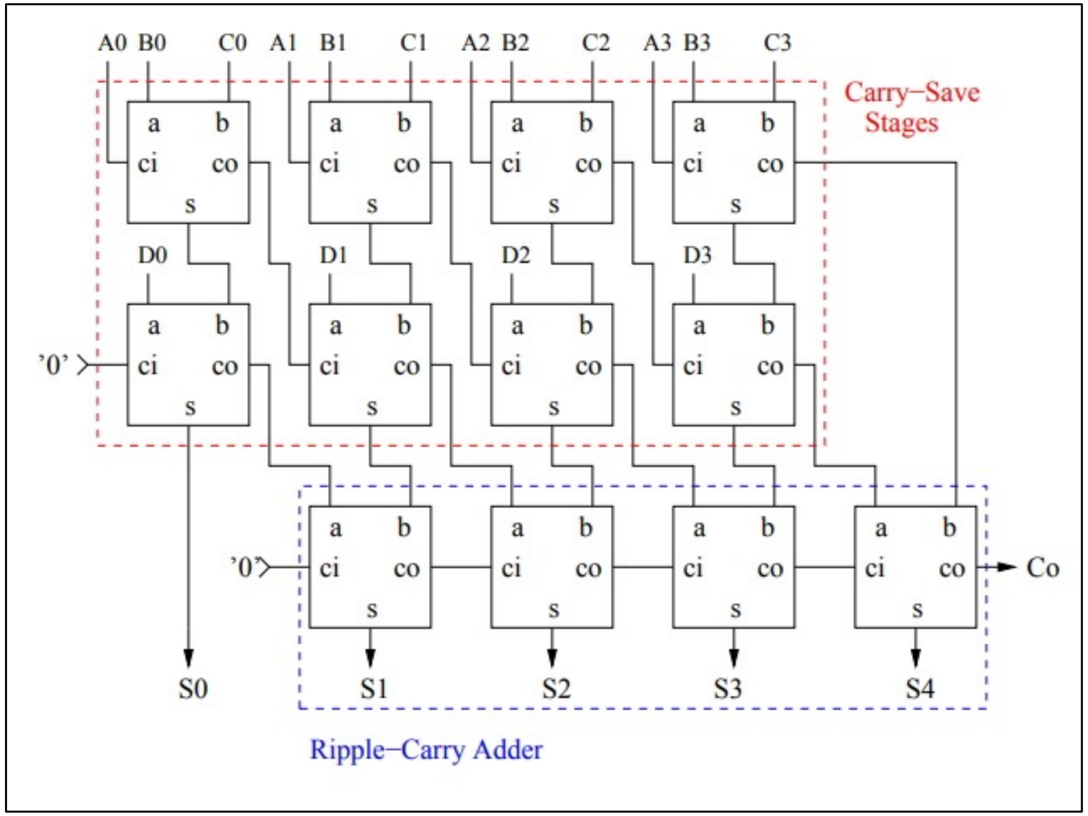
Carry-save Adder: Generic arithmetic

Here are a few ideas starting with Quantum carry-save adder, which can be also utilized for Quantum multiplier implementation. Carry-save adders are extension of 2-operand adders, which we typically need for multiplications.

1. This is a work from 1998 - https://arxiv.org/abs/quant-ph/9808061 - the most cited one in Quantum carry save adder. This uses majority gates. For carry save adders, majority logic can be implemented in a chained form - https://ieeexplore.ieee.org/abstract/document/7909019 - resulting in significant delay reduction. We can use the chained majority logic technique to reduce the overall Quantum carry-save adder delay (T-depth).
2. Use this improved quantum carry save adder for designing efficient quantum multiplier. We did a prior work on toom-cook multiplication (https://journals.aps.org/pra/abstract/10.1103/PhysRevA.98.012311) , which includes some benchmarked results. That can be used for a basis of comparison.
3. Use Gidney's logical-and technique for reducing Qubit count for (1) and (2).
4. Use large Toffoli gates (higher radix idea from Siyi) to reduce T-depth for (1) and (2).

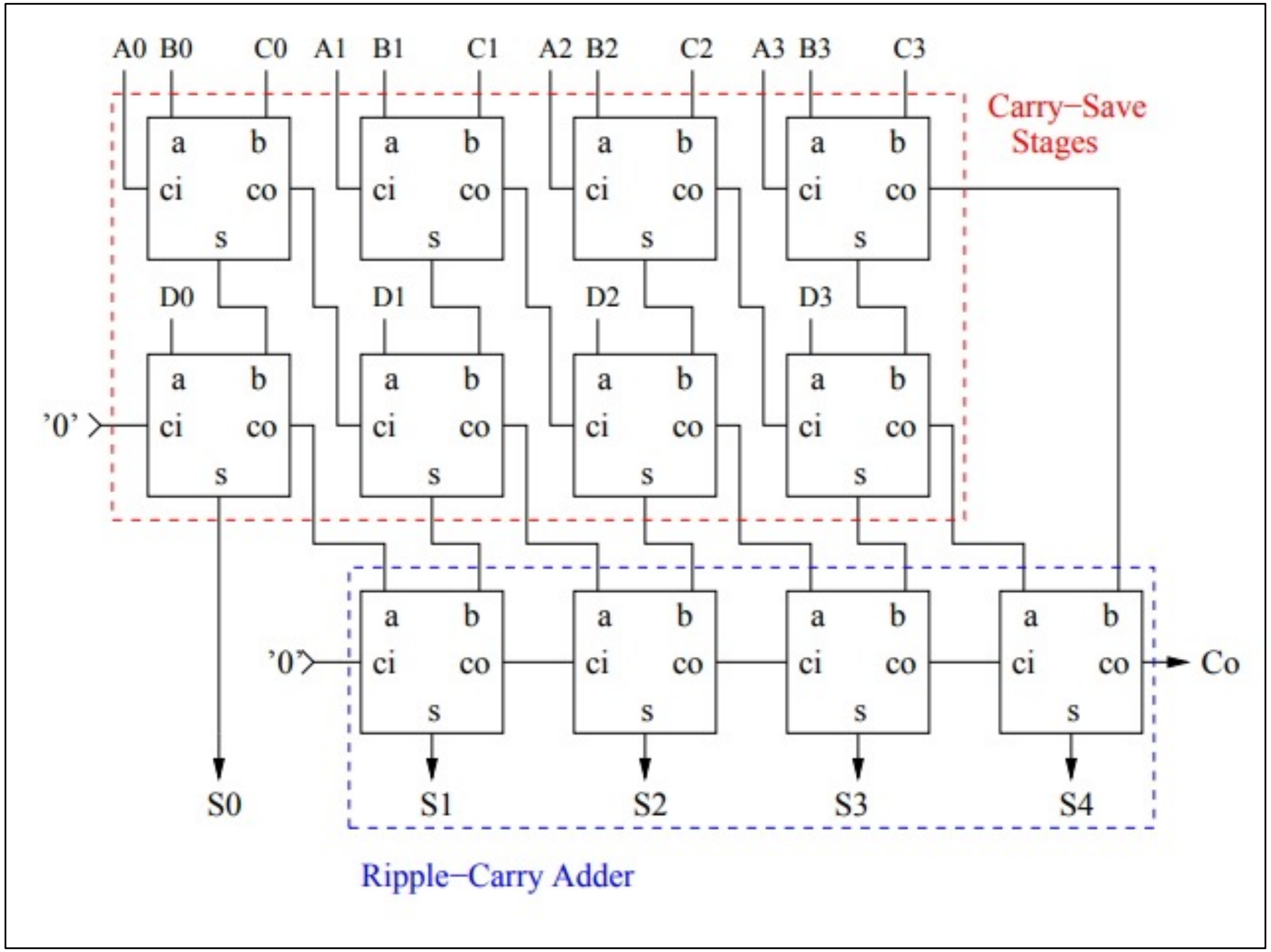
Carry-save adder

- 3개 이상의 덧셈을 효율적으로 수행하기 위한 덧셈기
  - 3 operands version: **a + b + c**
  - 4 operands version: **a + b + c + d**



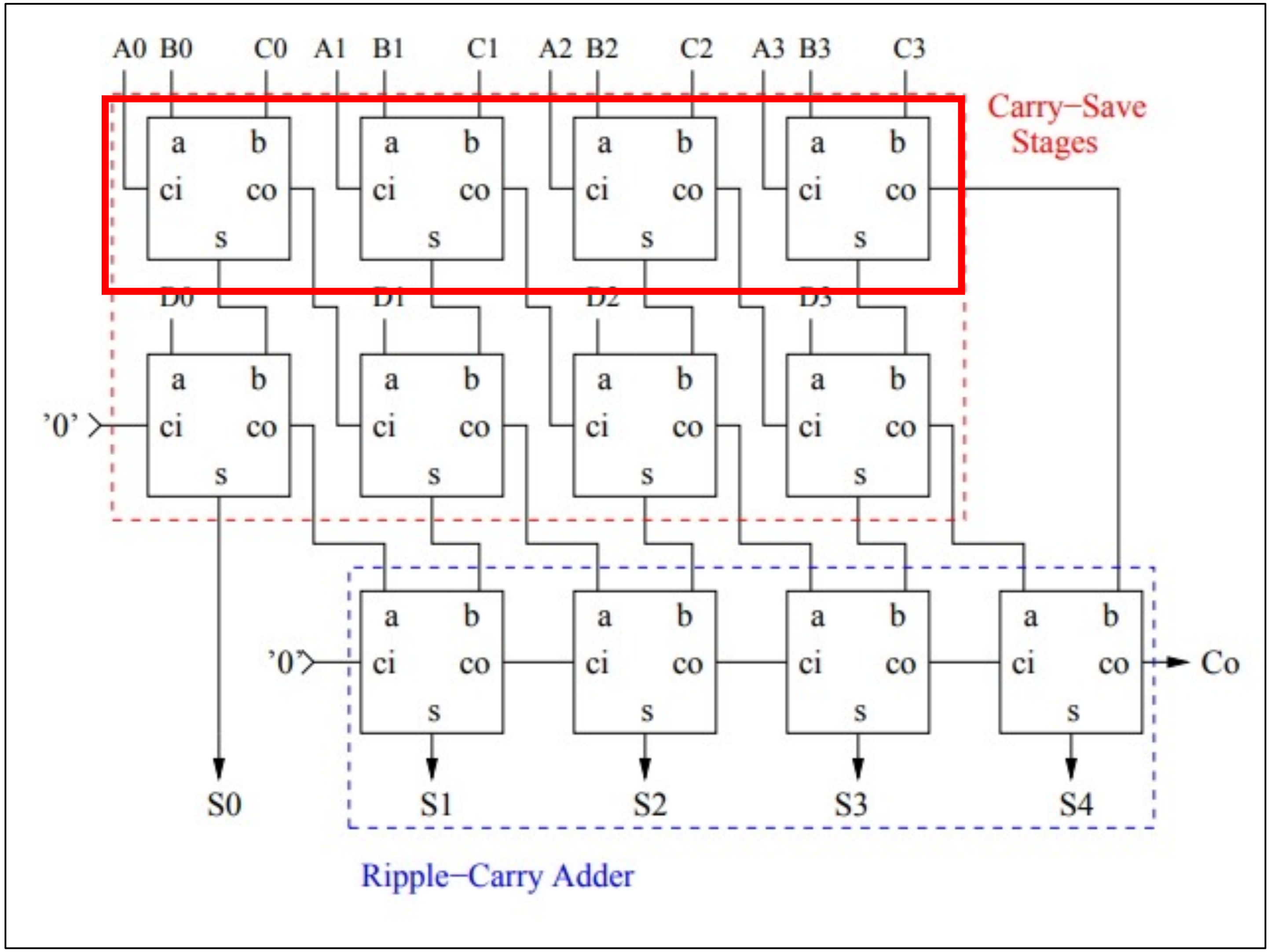
Carry-save Adder

- 3 operands version: **a + b + c**
- 4 operands version: **a + b + c + d**



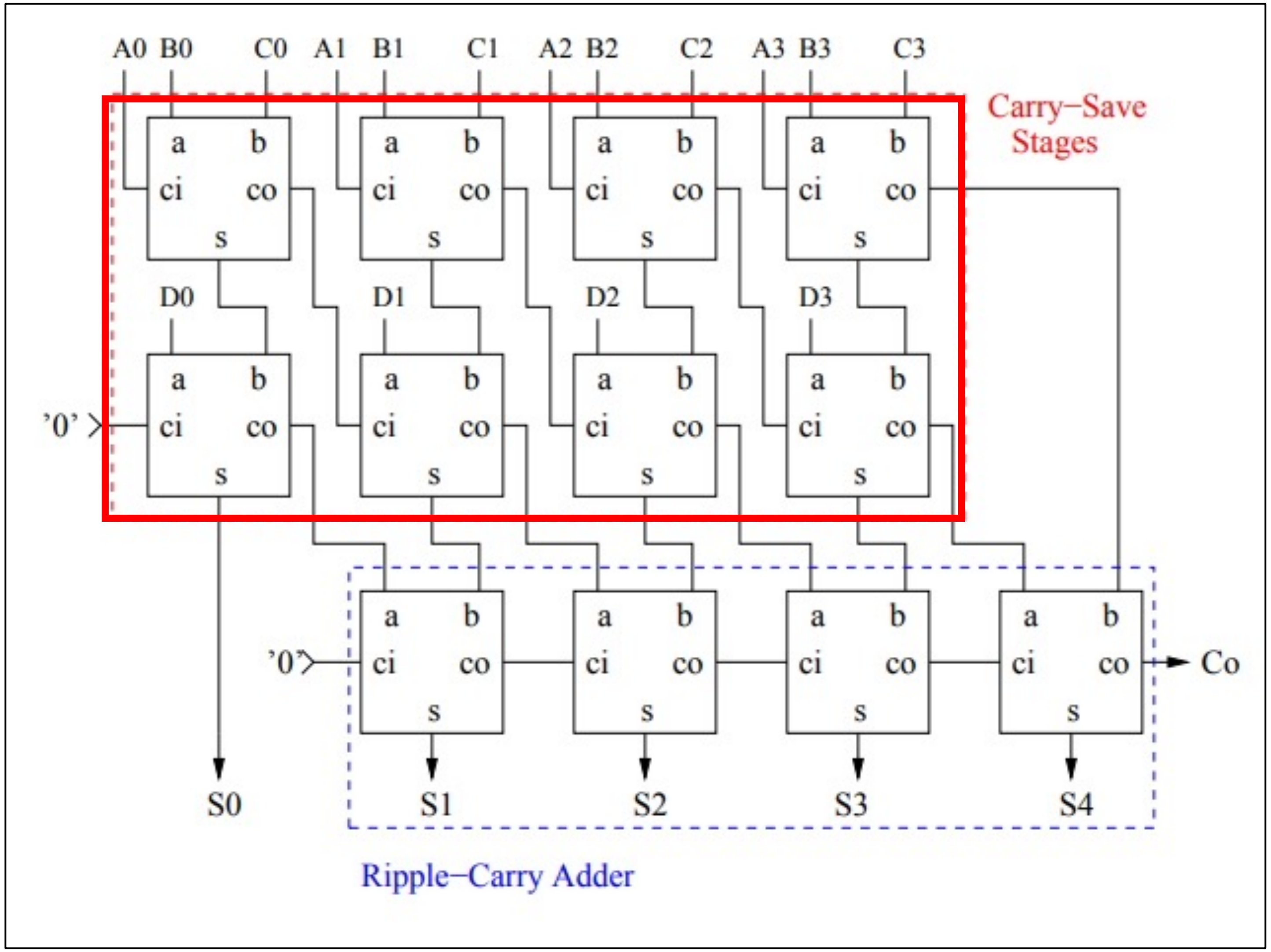
Carry-save Adder

- 3 operands version: **a + b + c**
- 4 operands version: **a + b + c + d**



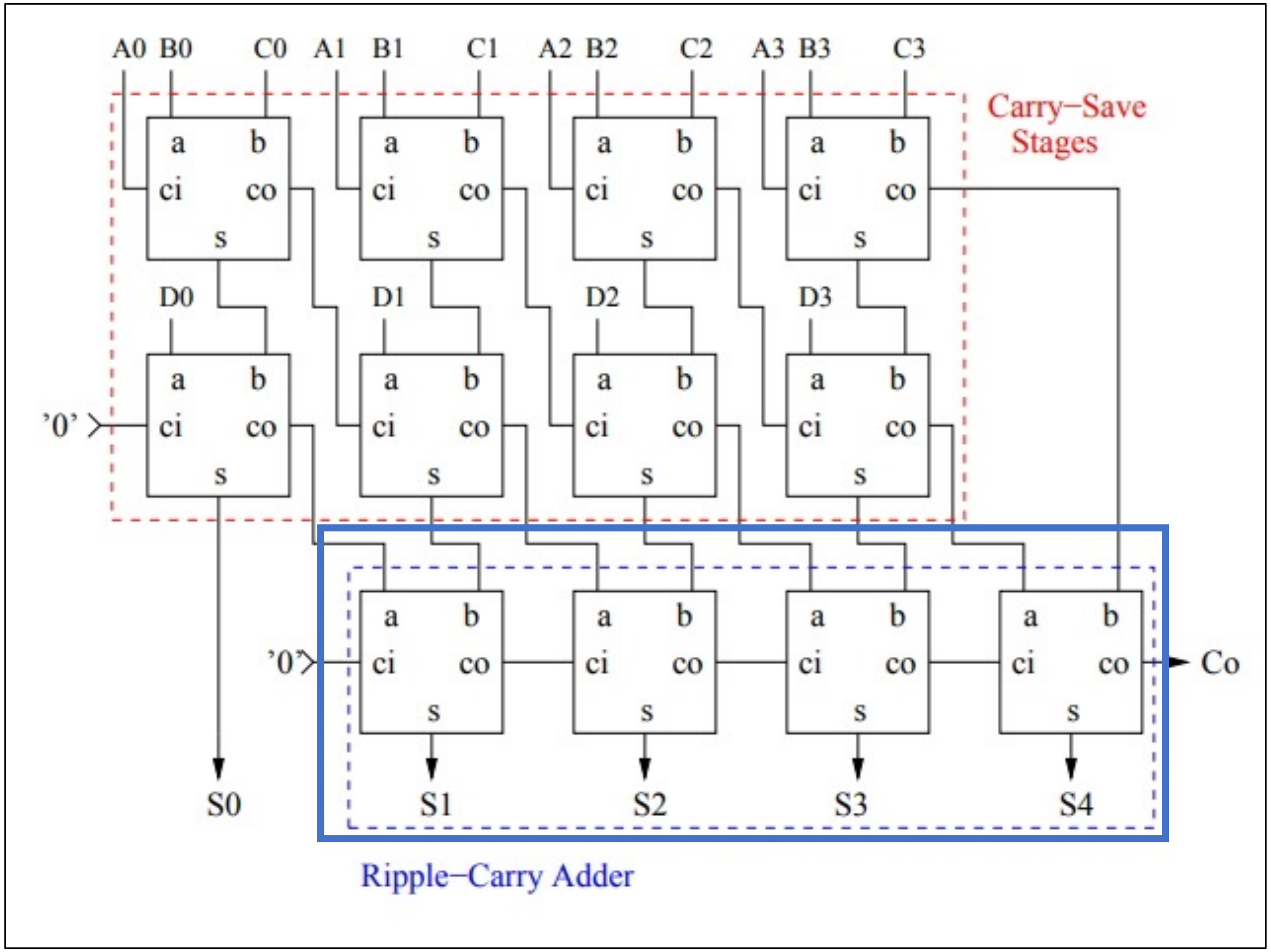
Carry-save Adder

- 3 operands version: **a + b + c**
- 4 operands version: **a + b + c + d**



Carry-save Adder

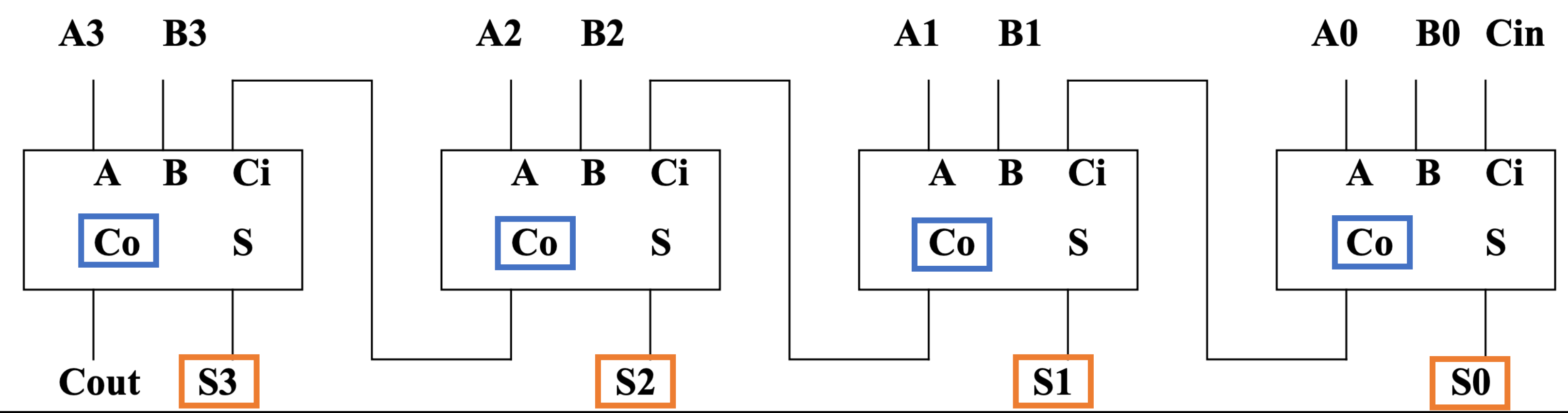
- carry-save 단계 이후, **단순 덧셈** 수행



Quantum Carry-save Adder

- 양자 버전의 carry-save adder는 1988년 Gosset의 논문이 유일
  - Classical carry-save adder 구조와 동일
    - **Carry-save 단계에서는, S와 C 두 값을 output으로 출력**
      - S는 Sum, C는 Carry 값을 저장, Ex) 1+1+1 → S = 1, C = 1

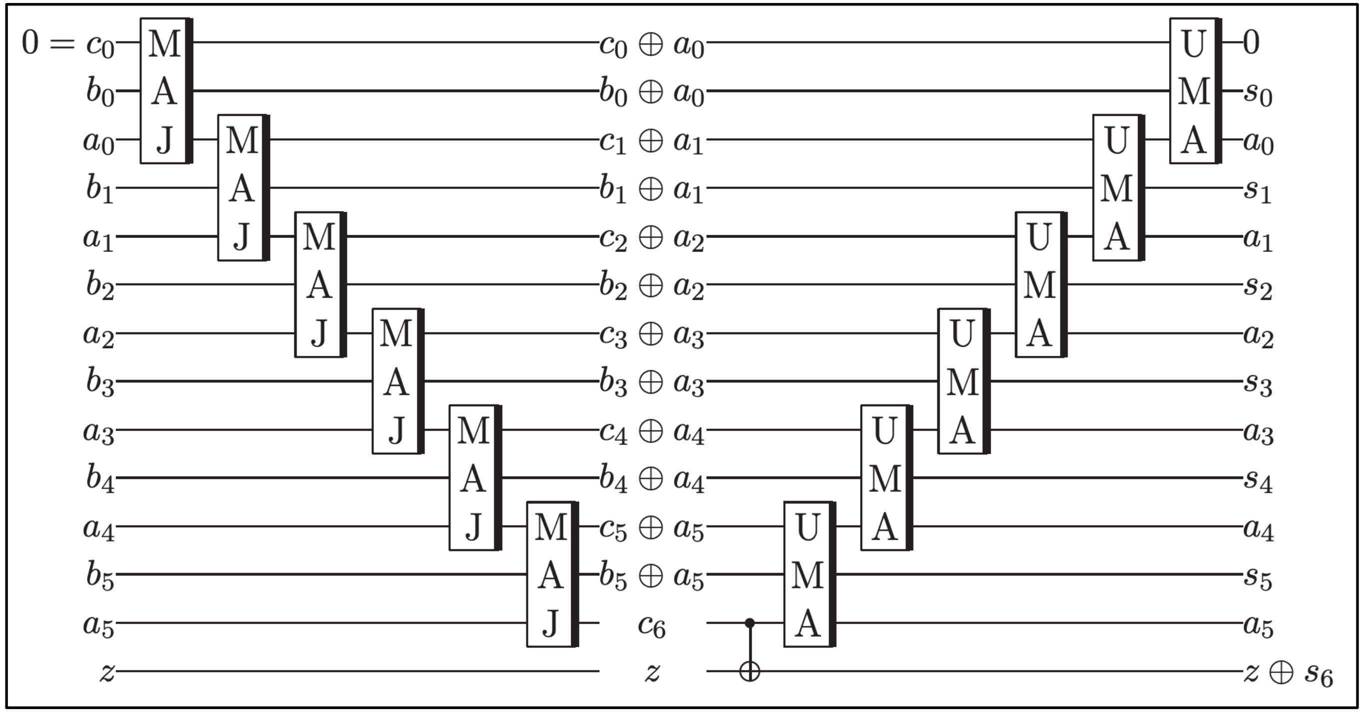
FIGURE 1. Classical Ripple Carry Adder.

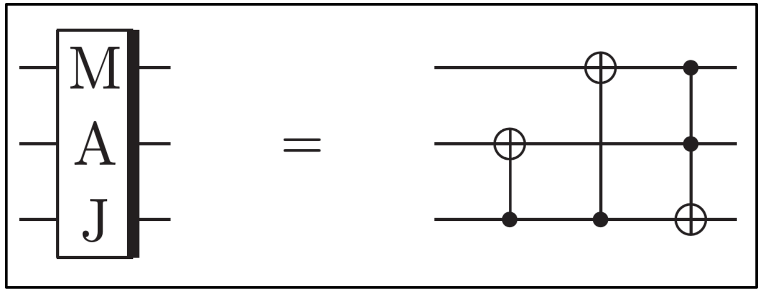


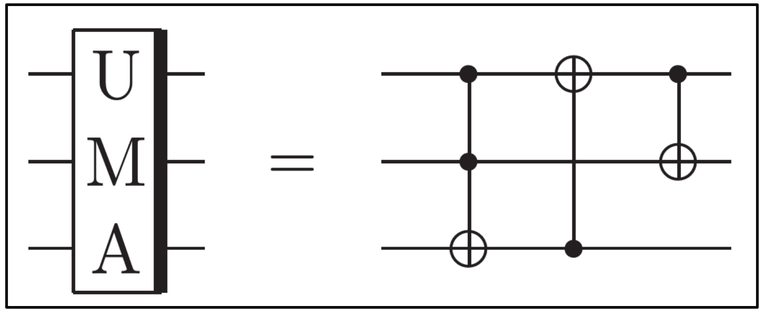
Phil Gossett, "Quantum Carry-Save Arithmetic", 1998.

Ripple-Carry Adder

- Ripple-carry adder에서의 **MAJ**oirty, **U**n**MA**jority 게이트

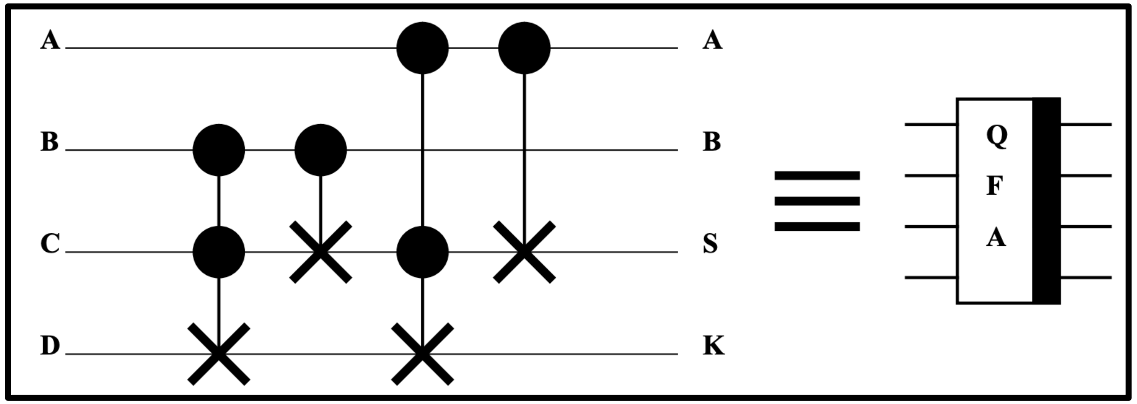






Quantum Carry-save Adder

- Quantum carry save adder에서의 MAJ와 UMAJ



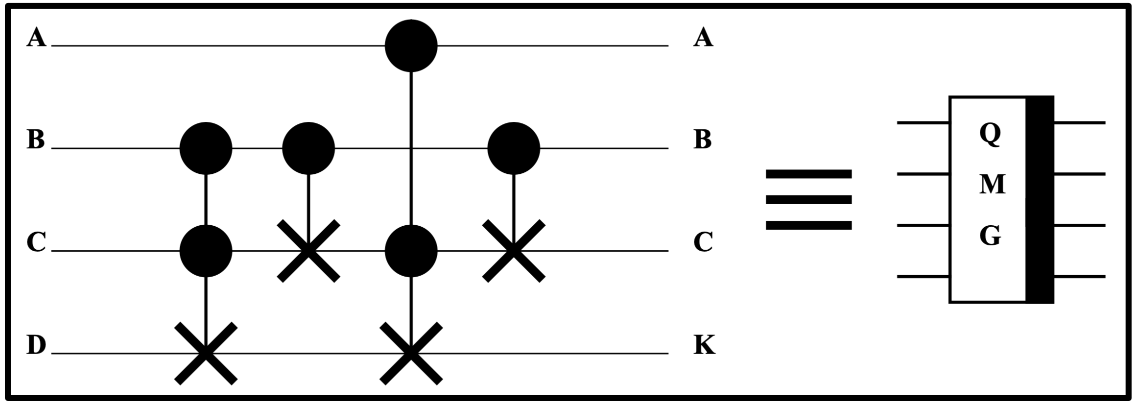


TABLE 2. Quantum Full Adder.

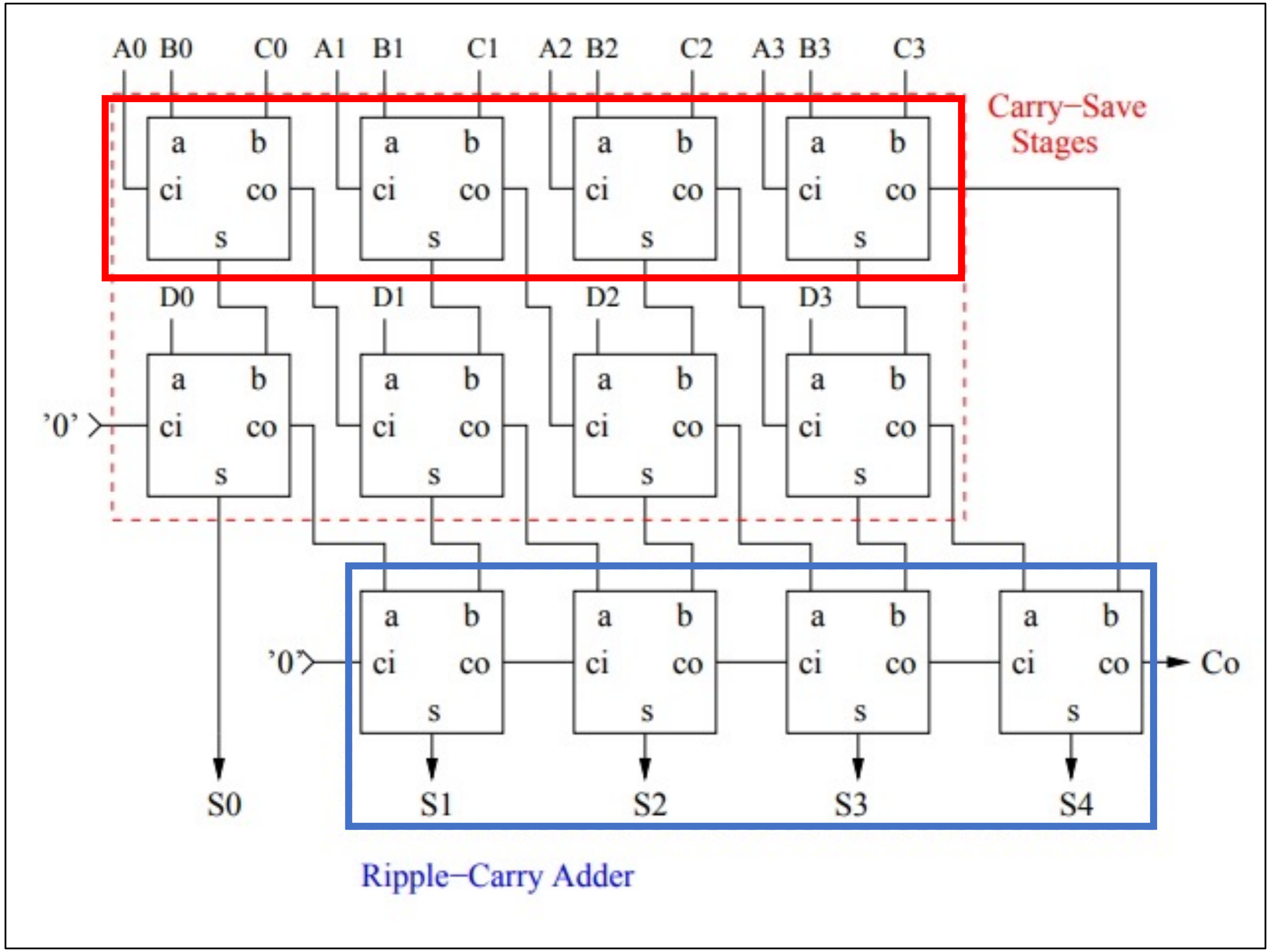
Input				Output			
D	C	B	A	K	S	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1
0	0	1	0	0	1	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	1
0	1	1	0	1	0	1	0
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	0
1	0	1	1	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	0	0	0	1
1	1	1	0	0	0	1	0
1	1	1	1	0	1	1	1

Quantum Carry-save Adder

- 3 operands version: **a + b + c**

```
for i in range(n):
    CSA(eng, a[i], b[i], c[i], ancilla[i]) # c is sum, ancilla is carry
```

```
new_b = eng.allocate_qubit()
c.append(new_b)
CDKM_no_modular(eng, ancilla, c[1:n+1], input_c, output_c, n)
c.append(output_c)
```

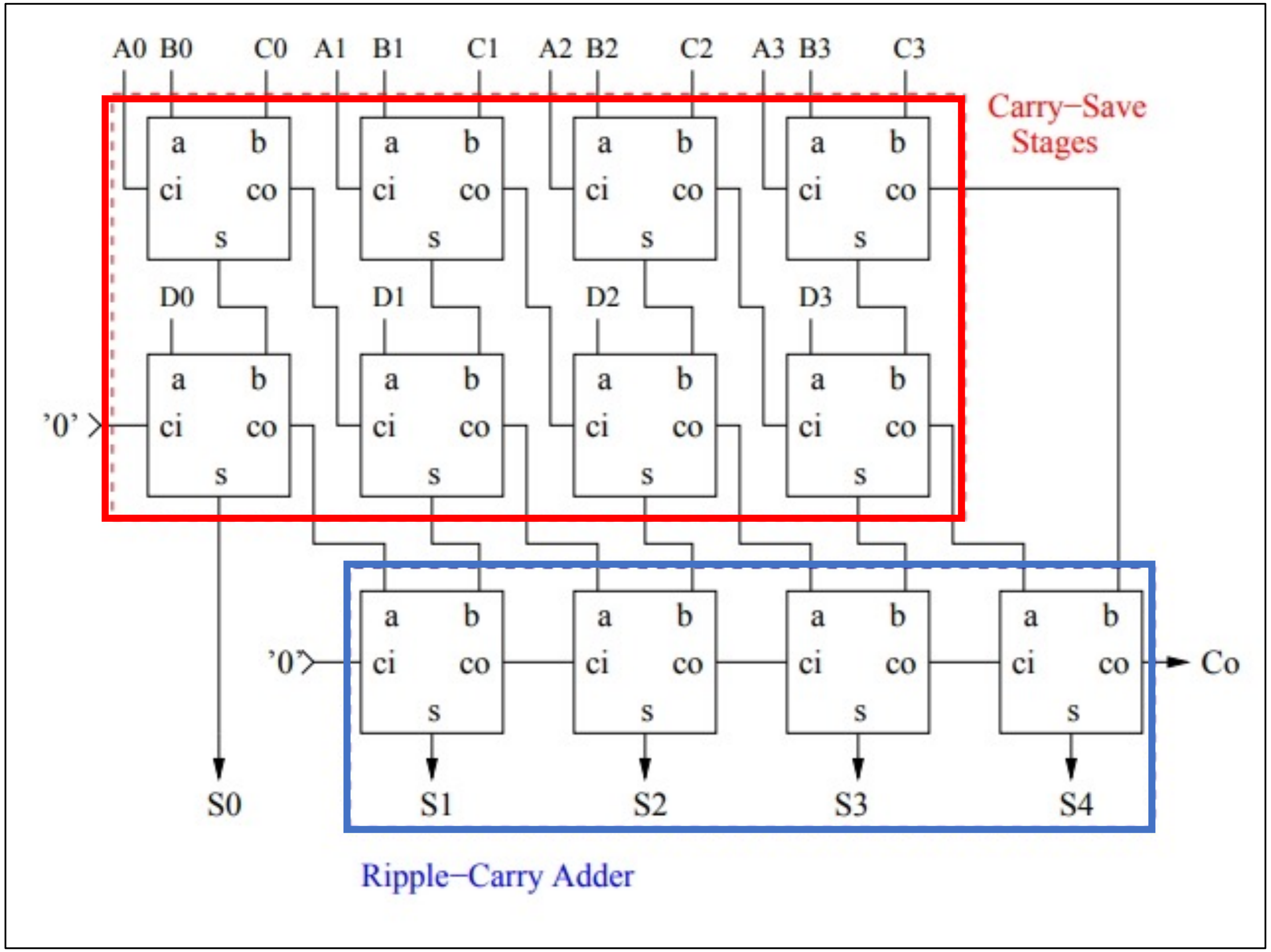


Quantum Carry-save Adder

- 4 operands version: **a + b + c + d**

```
for i in range(n):
    CSA(eng, a[i], b[i], c[i], ancilla_0[i]) # c is sum, ancilla is carry

new_a = eng.allocate_qubit()
for i in range(n-1):
    new_a.append(ancilla_0[i]) # ancilla[n] = first highest carry

for i in range(n):
    CSA(eng, new_a[i], d[i], c[i], ancilla_1[i]) # c is sum, ancilla is carry
```

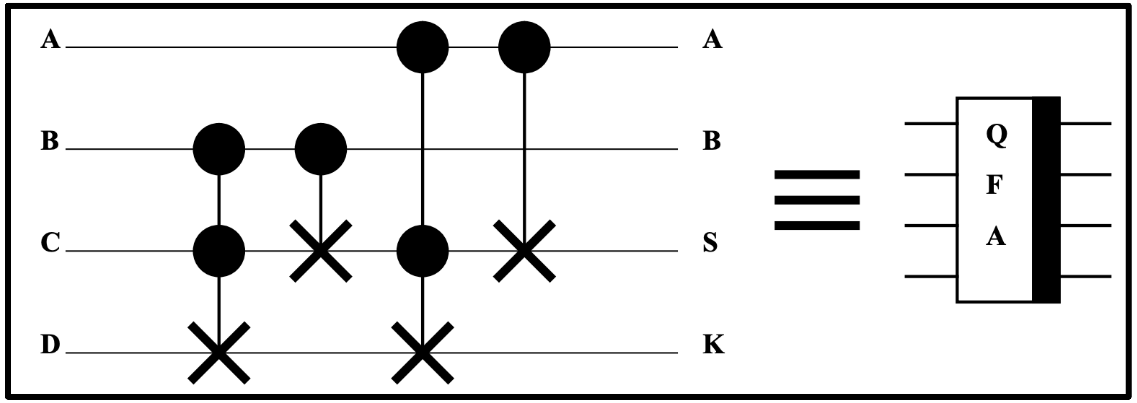
```
c.append(ancilla_0[n-1])

CDKM_no_modular(eng, ancilla_1, c[1:n+1], input_c, output_c, n)

c.append(output_c)
```

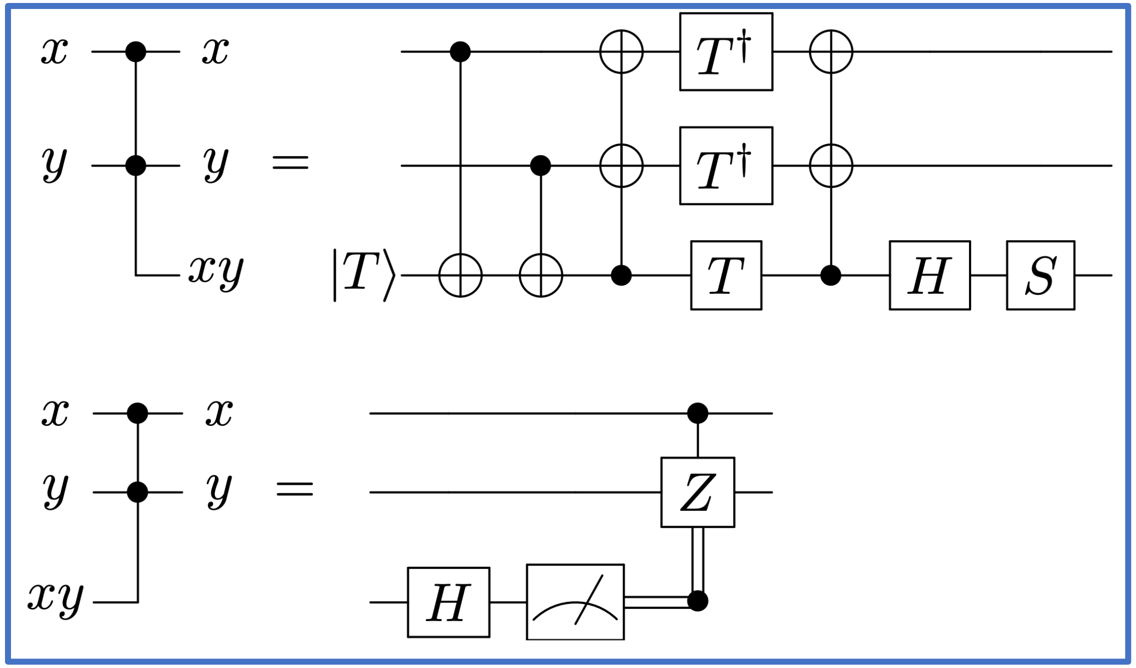


Demo

Future Work

- **Majority gate의 Toffoli gate를** **logical AND gate로 교체**



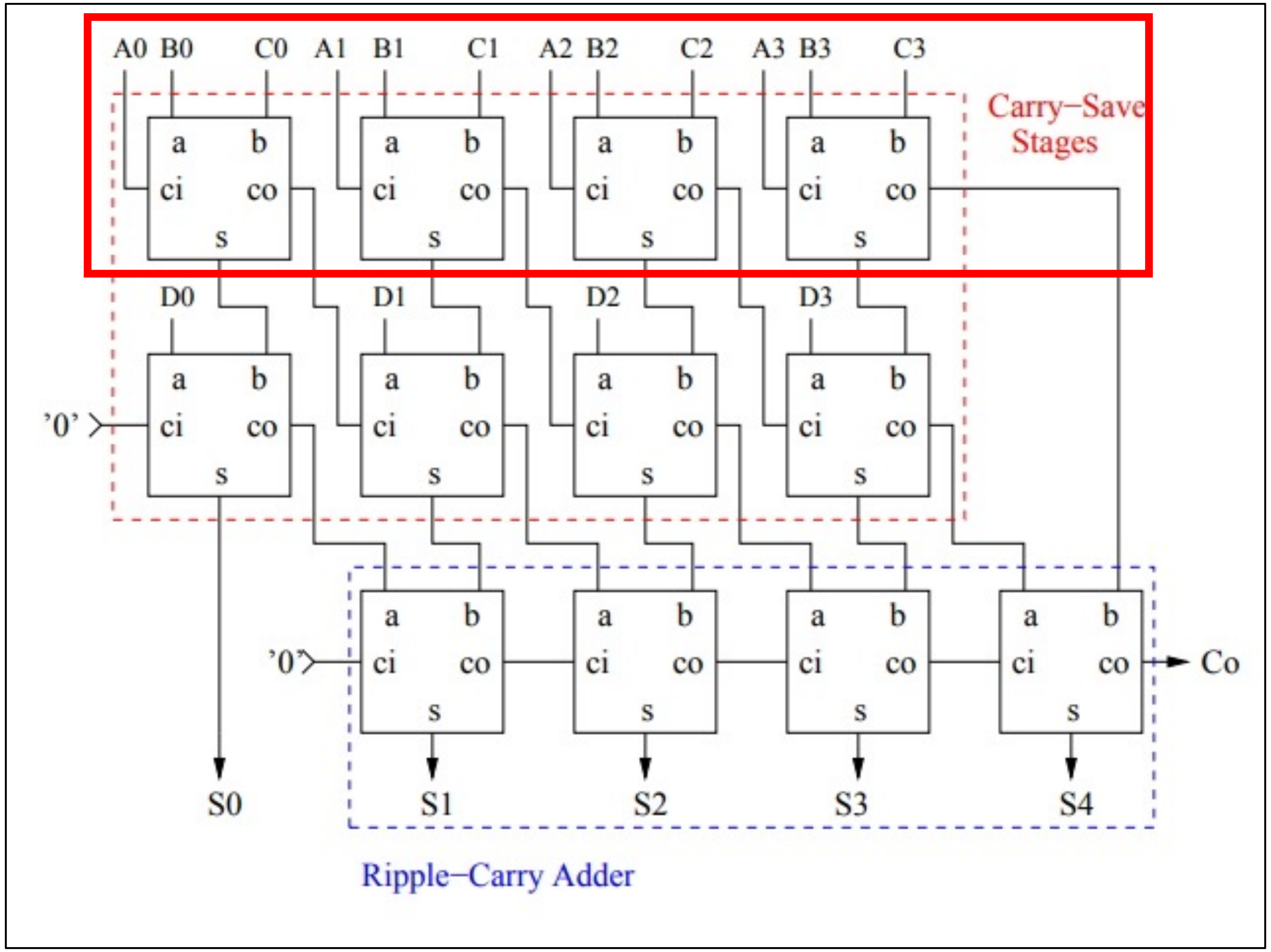


Future Work

Majority Logic Formulations
for Parallel Adder Designs at Reduced Delay
and Circuit Complexity

Vikramkumar Pudi, K. Sridharan, *Senior Member, IEEE* and F. Lombardi, *Fellow, IEEE*

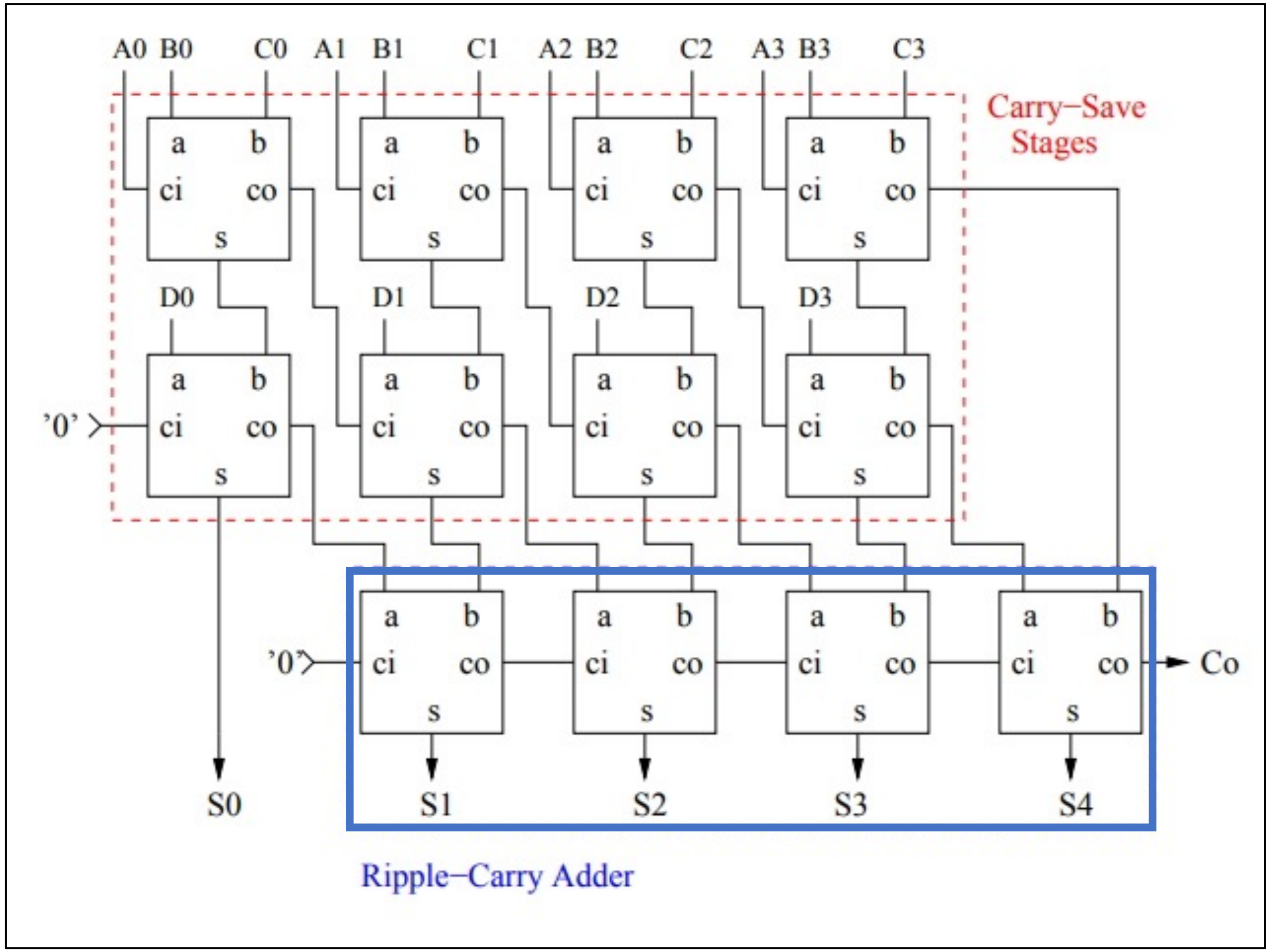
- **위 논문 (Classical) 기법을 적용 (Quantum)**
  - Majority 게이트들이 체인 형식으로 구현되어 복잡도를 크게 감소시킴



Vikramkumar Pudi; K. Sridharan; Fabrizio Lombardi, "Majority Logic Formulations for Parallel Adder Designs at Reduced Delay and Circuit Complexity", 2017.

Future Work

- 마지막, **단일 덧셈에 대해** 다양한 양자 덧셈기 적용 & 자원 비교
  - CDKM
  - Takahashi
  - Gidney
  - …



Vikramkumar Pudi; K. Sridharan; Fabrizio Lombardi, "Majority Logic Formulations for Parallel Adder Designs at Reduced Delay and Circuit Complexity", 2017.

Future Work

- 더 많은 operand로의 확장 (4개의 operand 단위로 구분 시키는 것 같음)
  - a + b + c + d + e + f …

- 구현 플랫폼 Cirq (Qiskit, Q# …)로 포팅

감사합니다