Quantum Arithmetic for Finance

https://youtu.be/h6pH4GYs-s8

Arithmetic for Finance

- f_k(s) = max (s – k, 0)
- g(x) = max (S_0 exp (\sigma x + c) – k, 0)

$$\left.\begin{array}{rl} (S - K) & \text{if } S > K \\ 0 & \text{if } S \leq K \end{array}\right\} = \mathrm{Max}\ (S\text{-}K, 0)$$

MAX(S-K,0)

- Step 1 : S-K
- Step 2 : MAX(S-K, 0)

- Ripple Borrow Subtractor 필요
- 어떻게 Ripple Borrow Subtractor 만드는가?

1. Parallel full-subtractor

2. Use Ripple Carry Adder

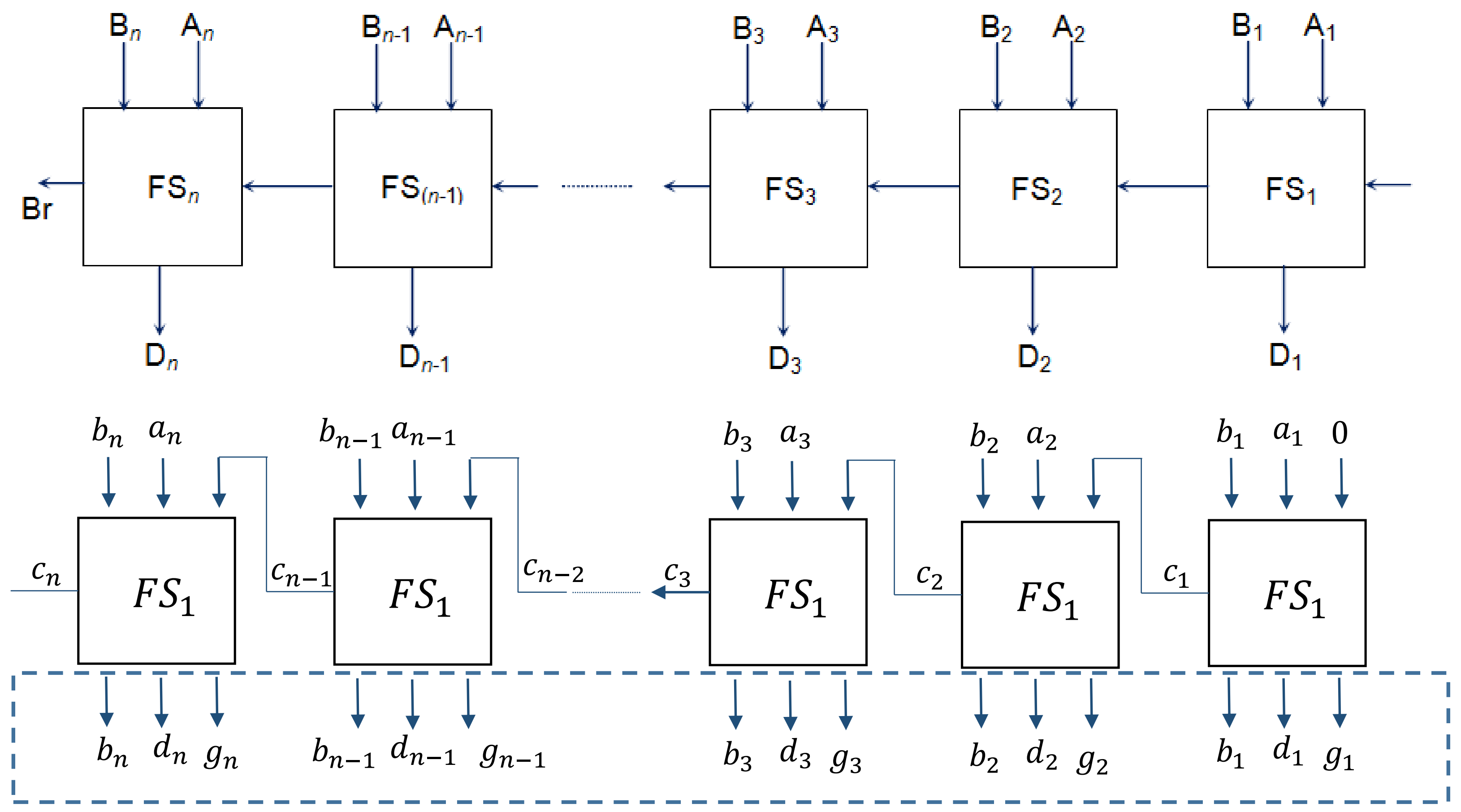
   2-1) $S - \mathrm{K} = S + \overline{K} + 1$

   2-2). $S - \mathrm{K} = \overline{\overline{S} + K}$

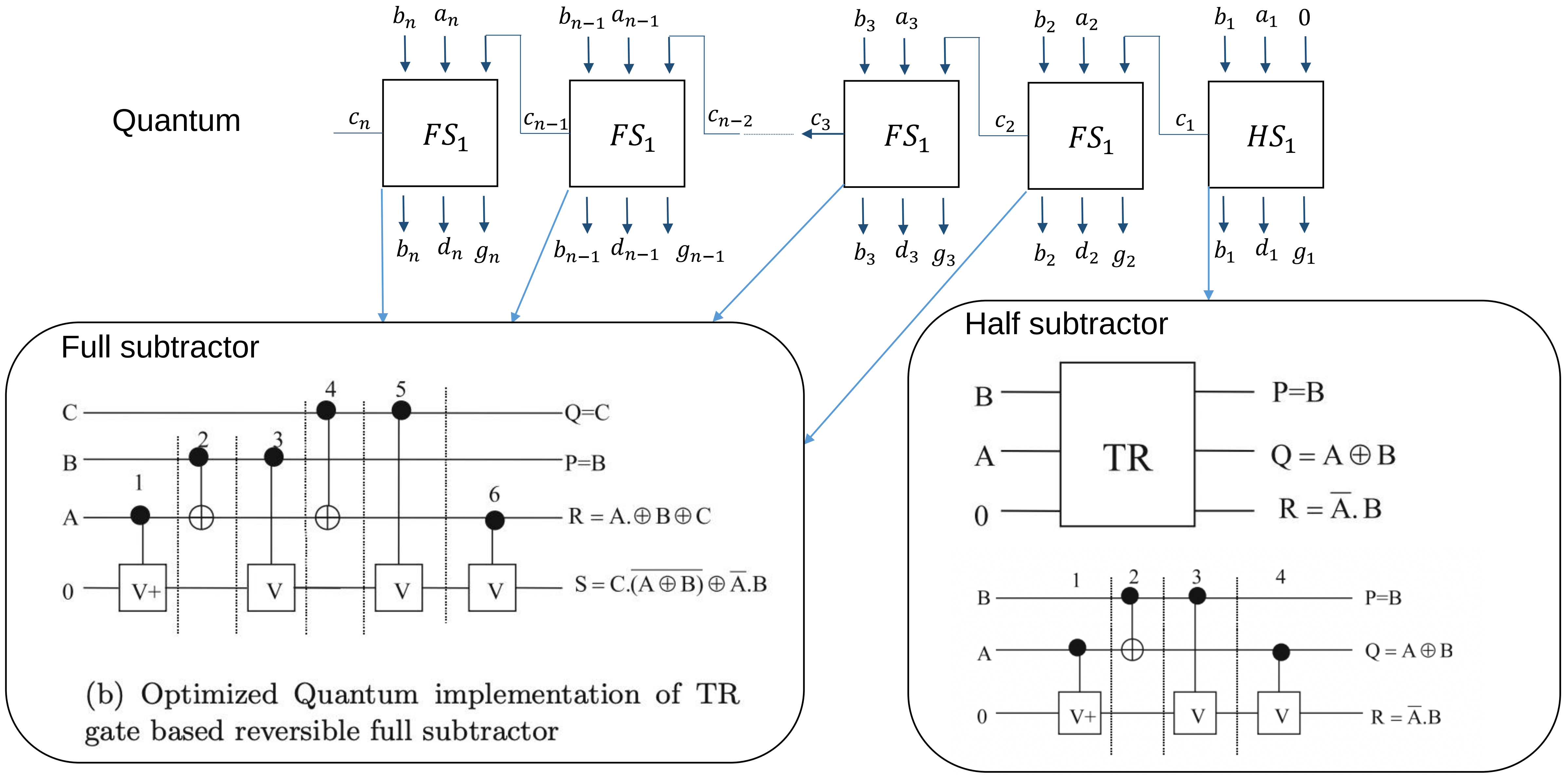
Parallel full-subtractor

Classic

Quantum



Use of extra bits to achieve reversible computation.



(b) Optimized Quantum implementation of TR gate based reversible full subtractor

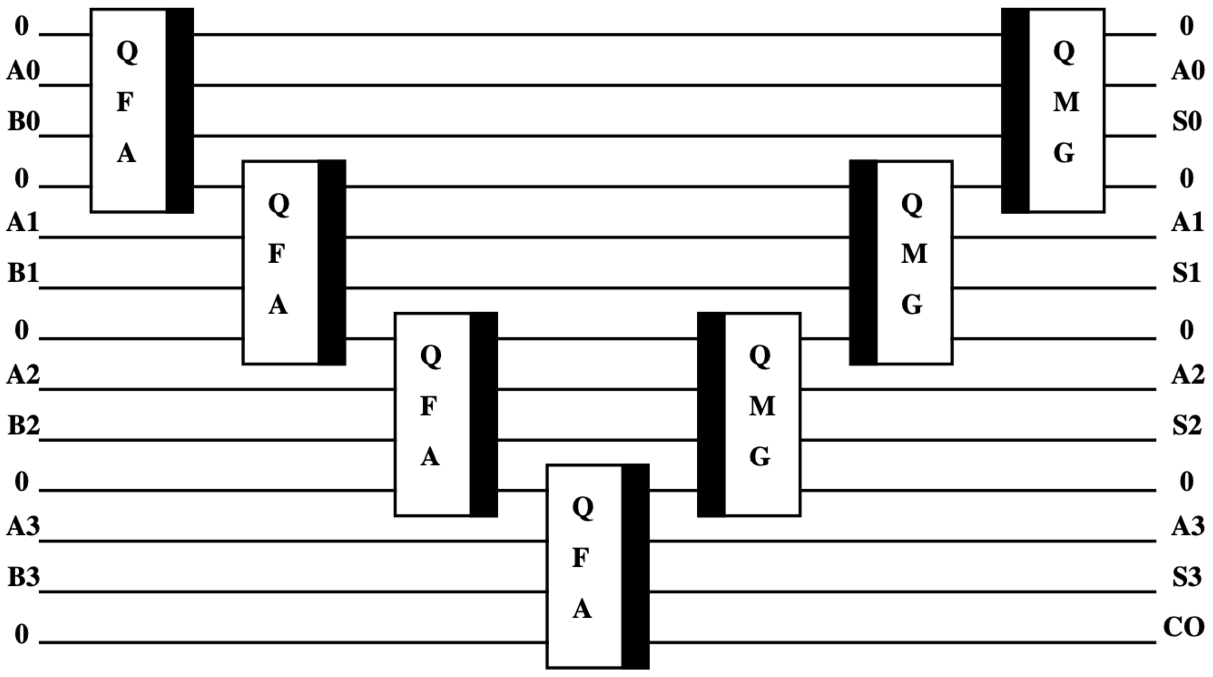
Step 1 : $S - \mathrm{K}$

- $S - \mathrm{K} = \mathrm{S} + \overline{K}$ + 1
  - 2의 보수 뺄셈
  - 2의 보수는 어떤 수를 2진법으로 표현했을 때,
    그 수의 비트 값을 뒤집고 1을 더한 값
    ex) $-\mathrm{K}$ = $\overline{K}$ + 1

- $S - \mathrm{K} = \overline{\overline{\mathrm{S}} + K}$
  - $-\mathrm{K}$ = $\overline{K}$ + 1 → $\overline{K}$ = $-\mathrm{K} - 1$
  - $\overline{\overline{\mathrm{S}} + K}$ = $-(\overline{\mathrm{S}} + K) - 1$ = $-((-S - 1) + K) - 1$ = $(S + 1 - \mathrm{K}) - 1$ = $\mathrm{S} - \mathrm{K}$
  - + 1을 하지 않아도됨

Quantum Ripple Carry Adder

FIGURE 7. Quantum Ripple Carry Adder.



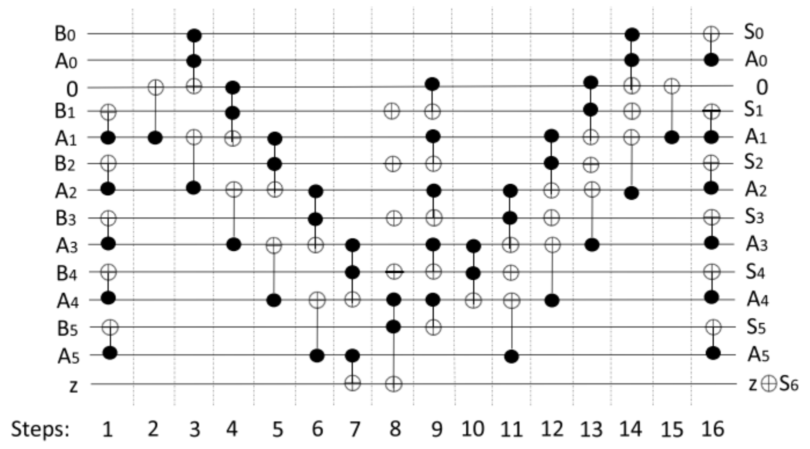
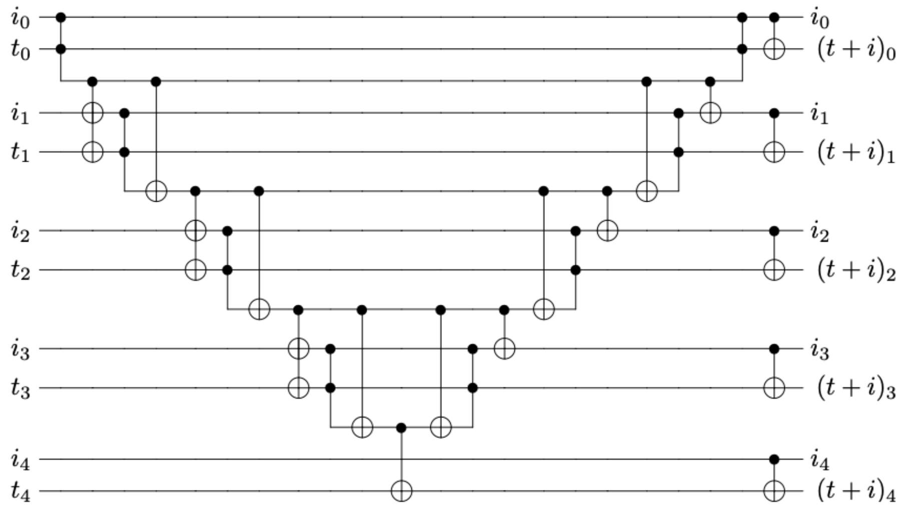
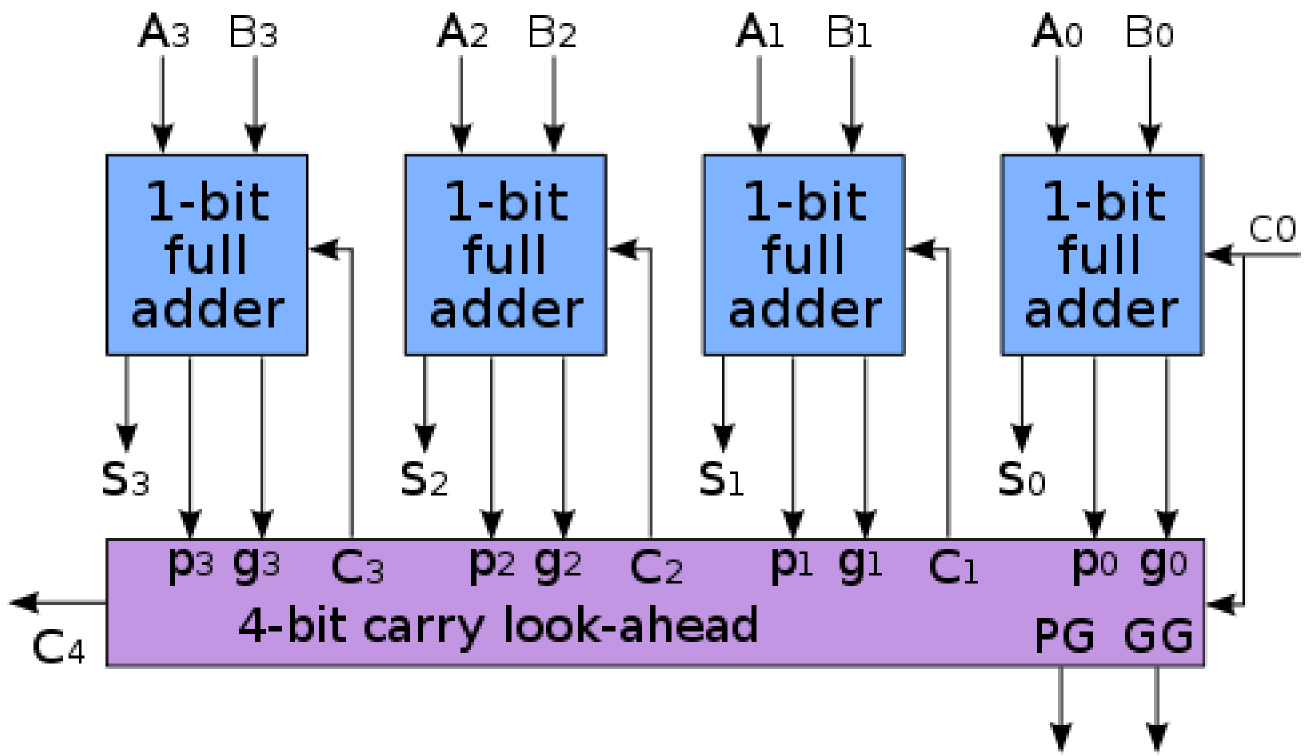
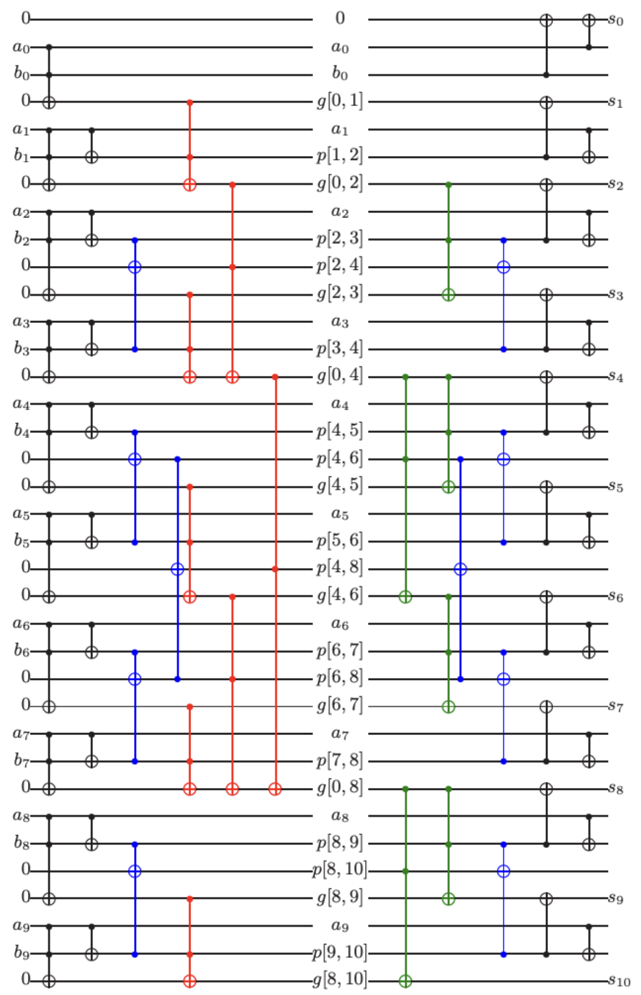


Fig. 18. Ripple-carry adder for $N = 6$ (assuming $C_{in} = 0$) proposed in Cuccaro et al. (2004).

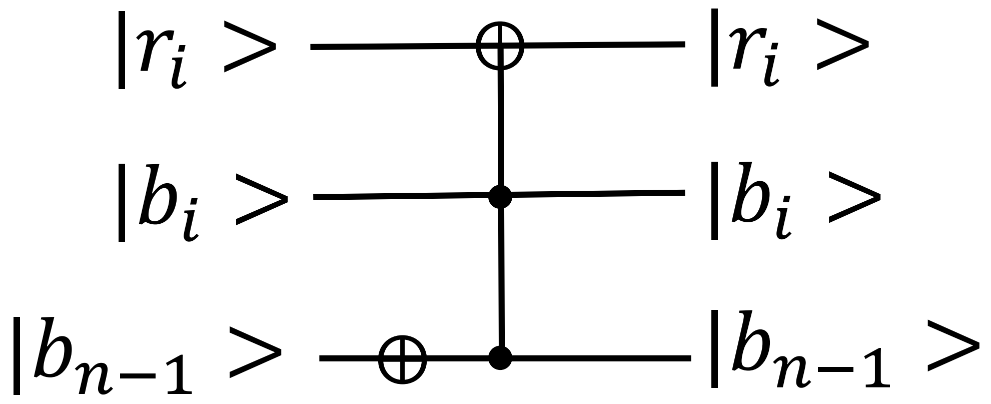


Carry-lookahead adder





Step 2 : $Max(S - K, 0)$

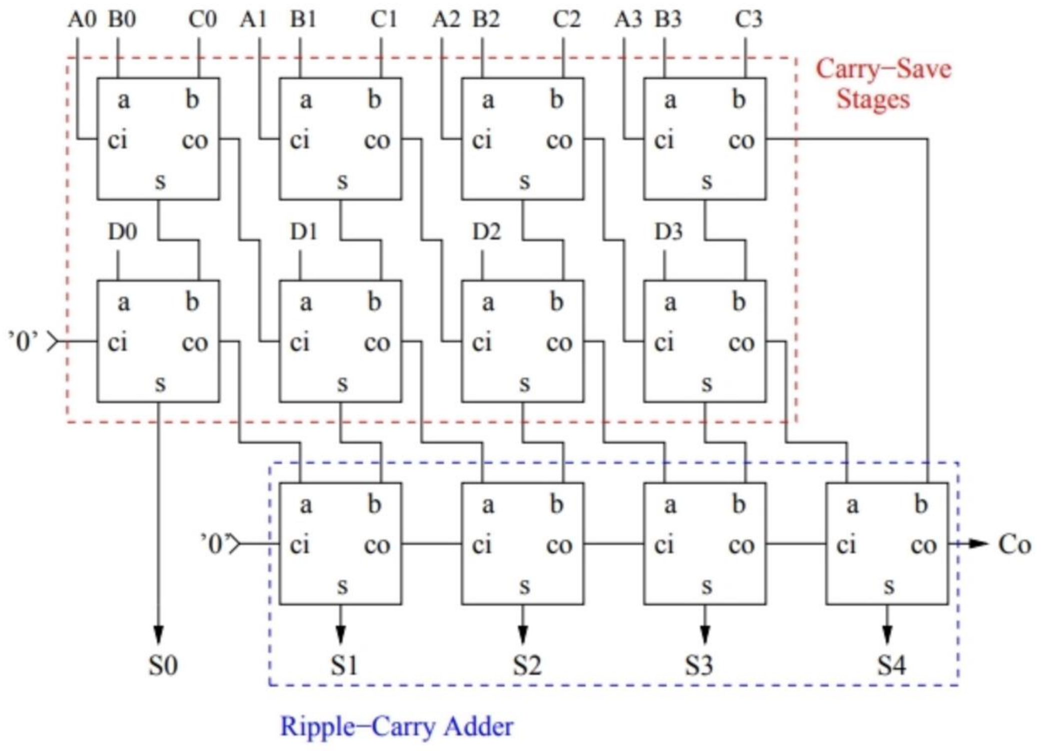


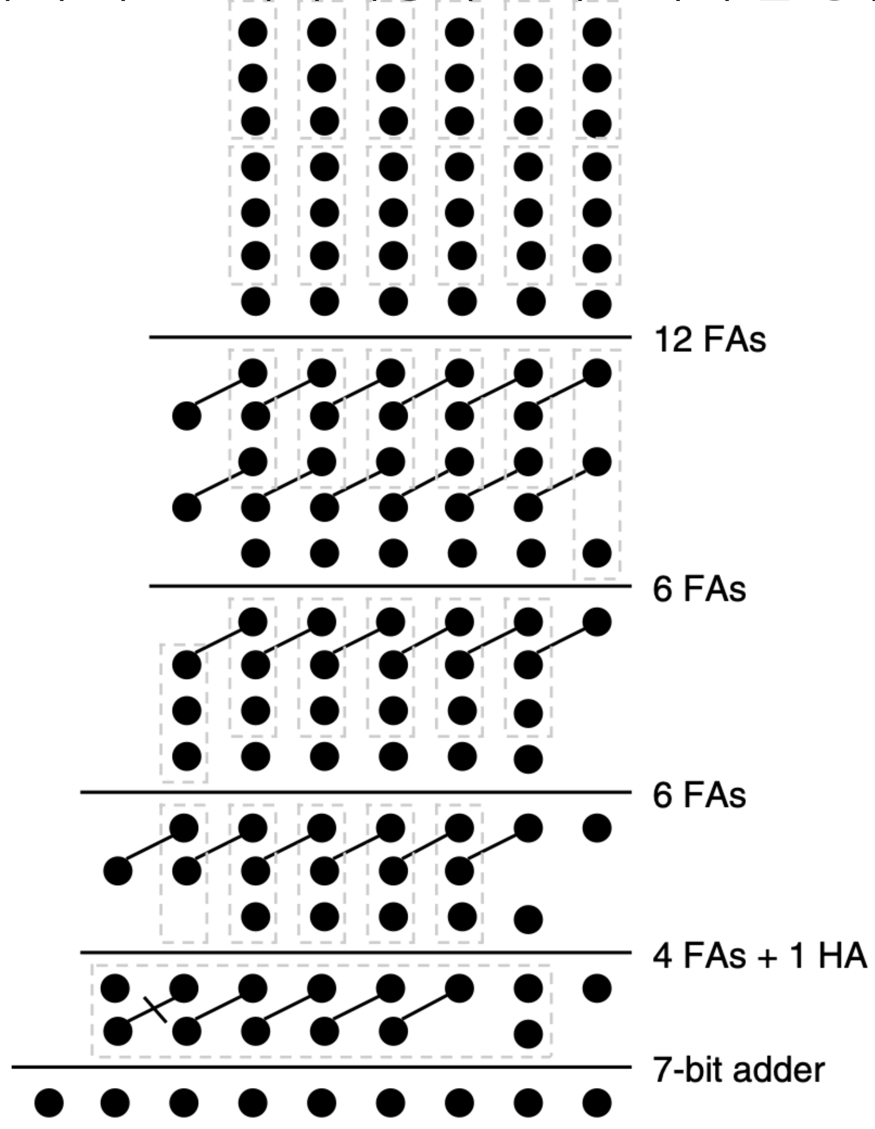
IF (S-K is negative)
    b_{n-1} is 1
    $r = 0$
else
    b_{n-1} is 0
    $r = b$ = S-K

Carry-save Adder: Generic arithmetic

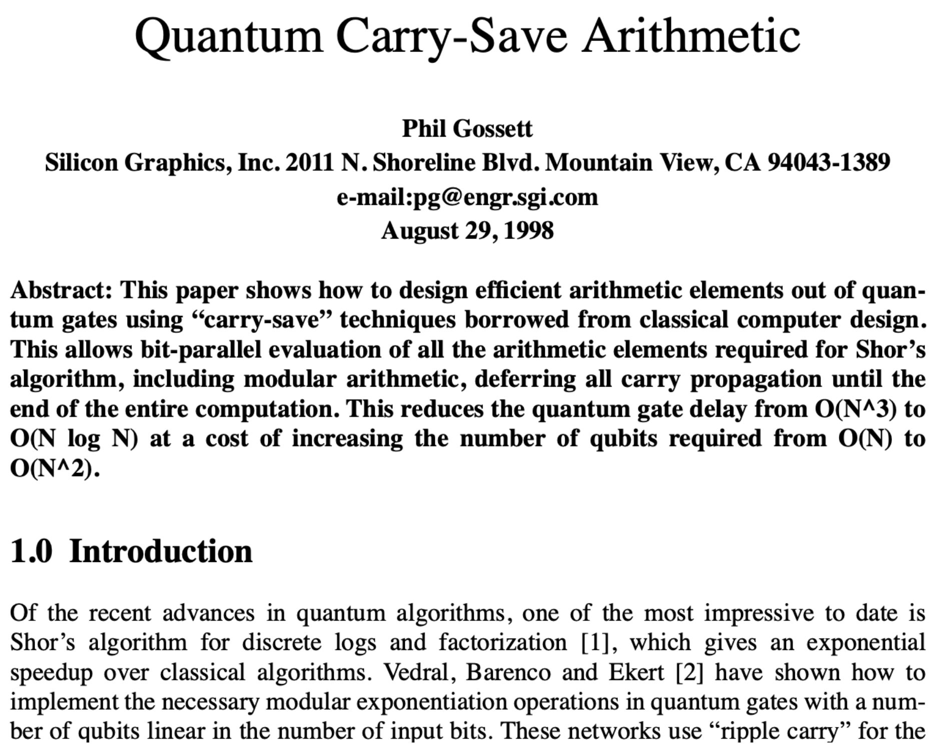
이진법에서 3개 그이상의 n 비트수 덧셈을 계산하기위한 컴퓨터 기본설계에 사용하는 가산기의 한 종류

A+B+C+D





Carry-save Adder: Generic arithmetic



Quantum Carry-Save Arithmetic

Phil Gossett
Silicon Graphics, Inc. 2011 N. Shoreline Blvd. Mountain View, CA 94043-1389
e-mail:pg@engr.sgi.com
August 29, 1998

Abstract: This paper shows how to design efficient arithmetic elements out of quantum gates using "carry-save" techniques borrowed from classical computer design. This allows bit-parallel evaluation of all the arithmetic elements required for Shor's algorithm, including modular arithmetic, deferring all carry propagation until the end of the entire computation. This reduces the quantum gate delay from O(N^3) to O(N log N) at a cost of increasing the number of qubits required from O(N) to O(N^2).

1.0 Introduction

Of the recent advances in quantum algorithms, one of the most impressive to date is Shor's algorithm for discrete logs and factorization [1], which gives an exponential speedup over classical algorithms. Vedral, Barenco and Ekert [2] have shown how to implement the necessary modular exponentiation operations in quantum gates with a number of qubits linear in the number of input bits. These networks use "ripple carry" for the

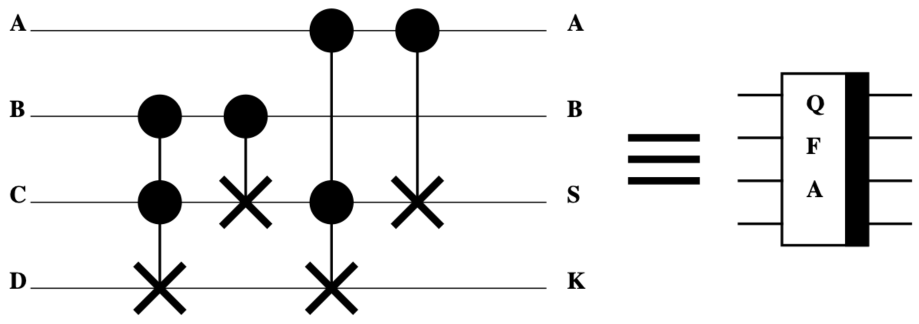


FIGURE 9. Quantum 4->2 Carry-Save Adder.

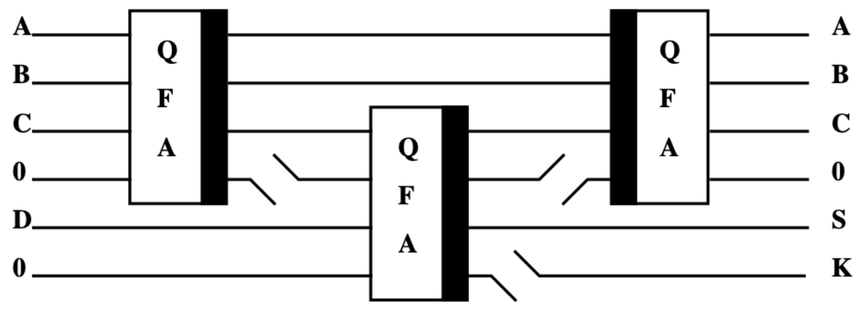
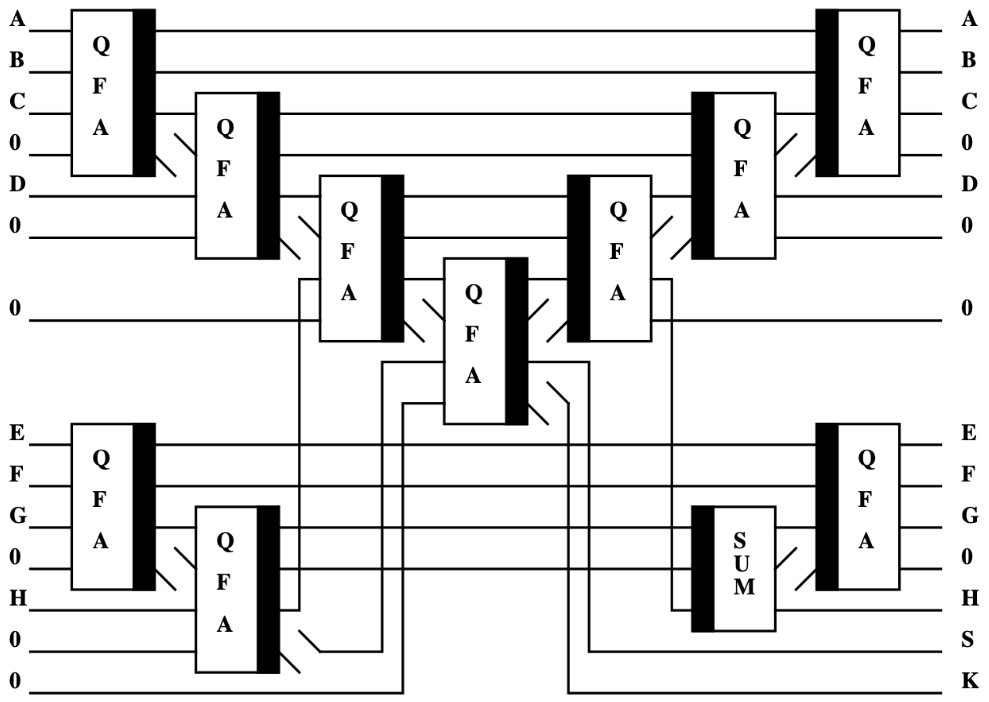
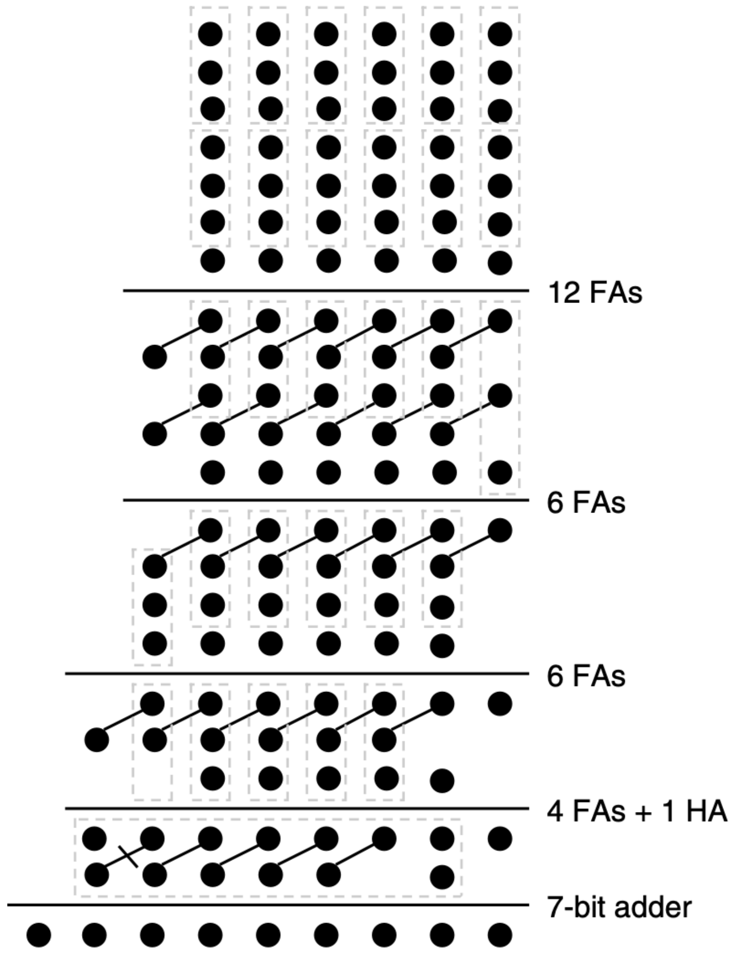
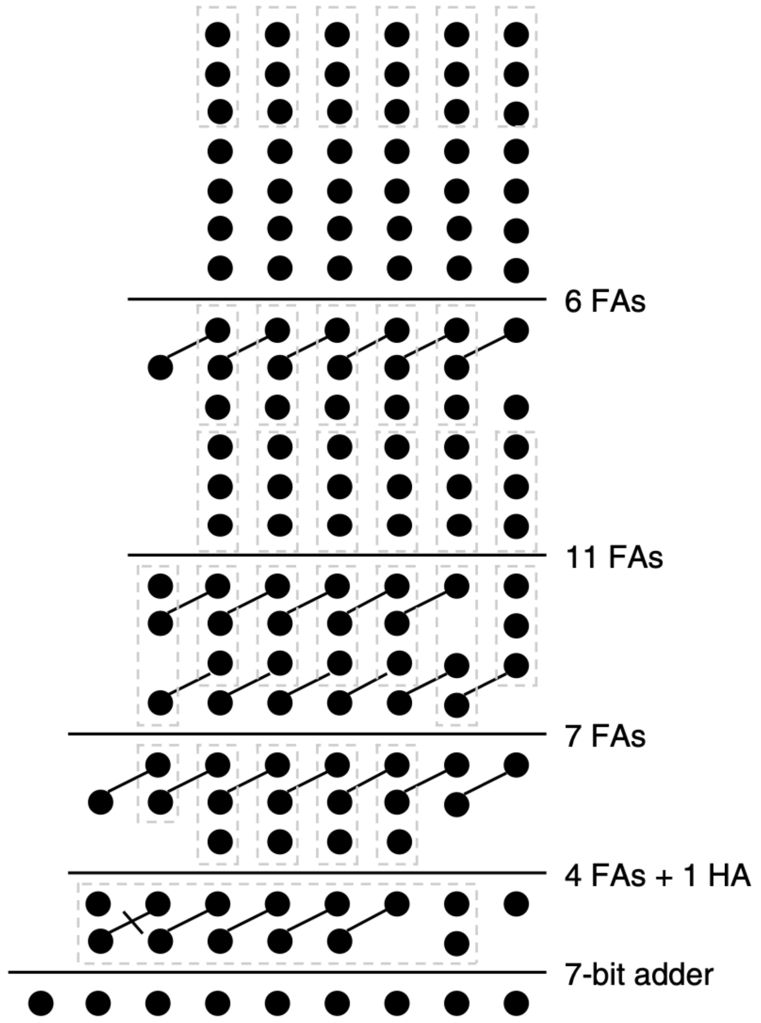


FIGURE 10. Quantum 8->2 Tree-Structured Carry-Save Adder.



WALLACE and DADDA TREES





Q & A