Reversible Quantum Ripple Carry Adders

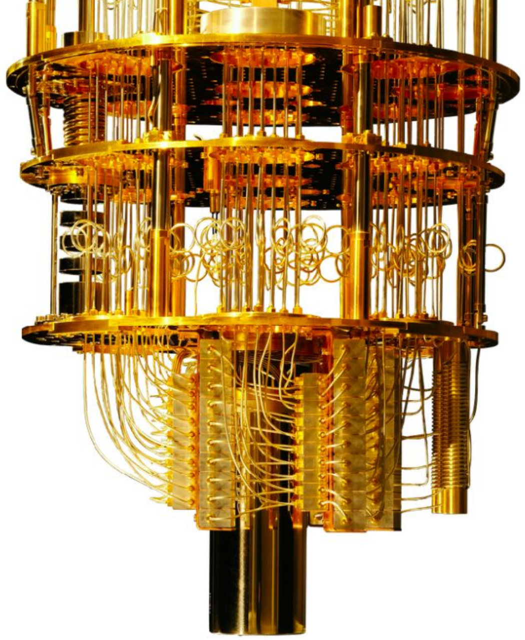
Advanced Architectures

SAPIENZA
UNIVERSITÀ DI ROMA

Andrea Di Marco (1835169)
Jemuel Espiritu Fuentes (1803530)

Introduction

SAPIENZA
UNIVERSITÀ DI ROMA



Introduction

- We implemented four quantum ripple-carry adders.
  - And some variations.
- In this presentation we consider 5-bit adders.
  - We made a generalized generator on Python.
- We used both Qiskit and QASM for the implementation.
- We are first going to explain the adders individually.
- At the end we will show a comparison of all the adders.
  - Using standard metrics that allow to highlight the strengths and weaknesses of each adder.

Efficient Addition

- Reversible adders are essential circuits in quantum computing systems.

- Shor's factoring algorithm is one of the most prominent examples in which reversible arithmetic is needed.
  - The most intensive part of the algorithm is exponentiation.
  - The usual approach uses multiplier circuits, constructed using adders.

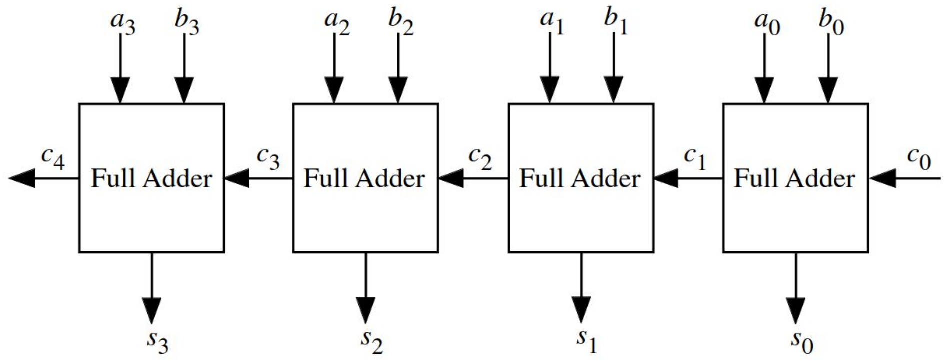
- For this reason there is a great interest in the creation of new designs and the perfection of the existing ones.

- Some methodologies focus on minimizing the necessary resources, others on optimizing computing time, etc.
  - Compromises are required.

Ripple-Carry Addition

- A one-bit full adder is a combinational circuit that forms the arithmetic sum of three bits.
  - It consists of three inputs and and two outputs

- A *n*-bit ripple-carry adder is built chaining *n* full adders, connecting the c_{out} output of every full adder with the c_{in} input of the next full adder.

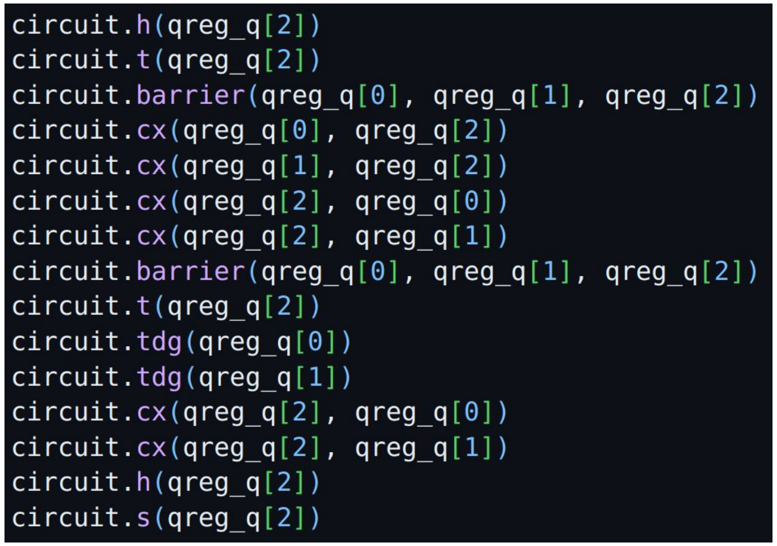


- Carry-lookahead adders precompute the carry in advance but the carry logic block gets very complicated for more than 4 bits

Qiskit

- Qiskit is an open-source software development kit (*SDK*) for working with quantum computers at the level of circuits, pulses, and algorithms.

- Provides tools for creating and manipulating quantum programs and running them on real quantum devices on IBM Quantum Platform or on simulators on a local computer.



```
circuit.h(qreg_q[2])
circuit.t(qreg_q[2])
circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2])
circuit.cx(qreg_q[0], qreg_q[2])
circuit.cx(qreg_q[1], qreg_q[2])
circuit.cx(qreg_q[2], qreg_q[0])
circuit.cx(qreg_q[2], qreg_q[1])
circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2])
circuit.t(qreg_q[2])
circuit.tdg(qreg_q[0])
circuit.tdg(qreg_q[1])
circuit.cx(qreg_q[2], qreg_q[0])
circuit.cx(qreg_q[2], qreg_q[1])
circuit.h(qreg_q[2])
circuit.s(qreg_q[2])
```

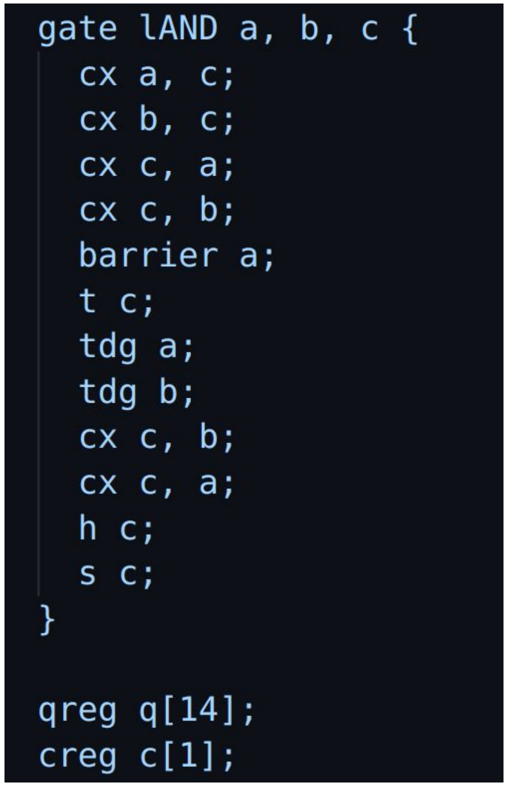
SAPIENZA
UNIVERSITÀ DI ROMA

QASM

- Open Quantum Assembly Language (*OpenQASM*) is a programming language designed for describing quantum circuits and algorithms for execution on quantum computers.

- Allows to define custom gates.

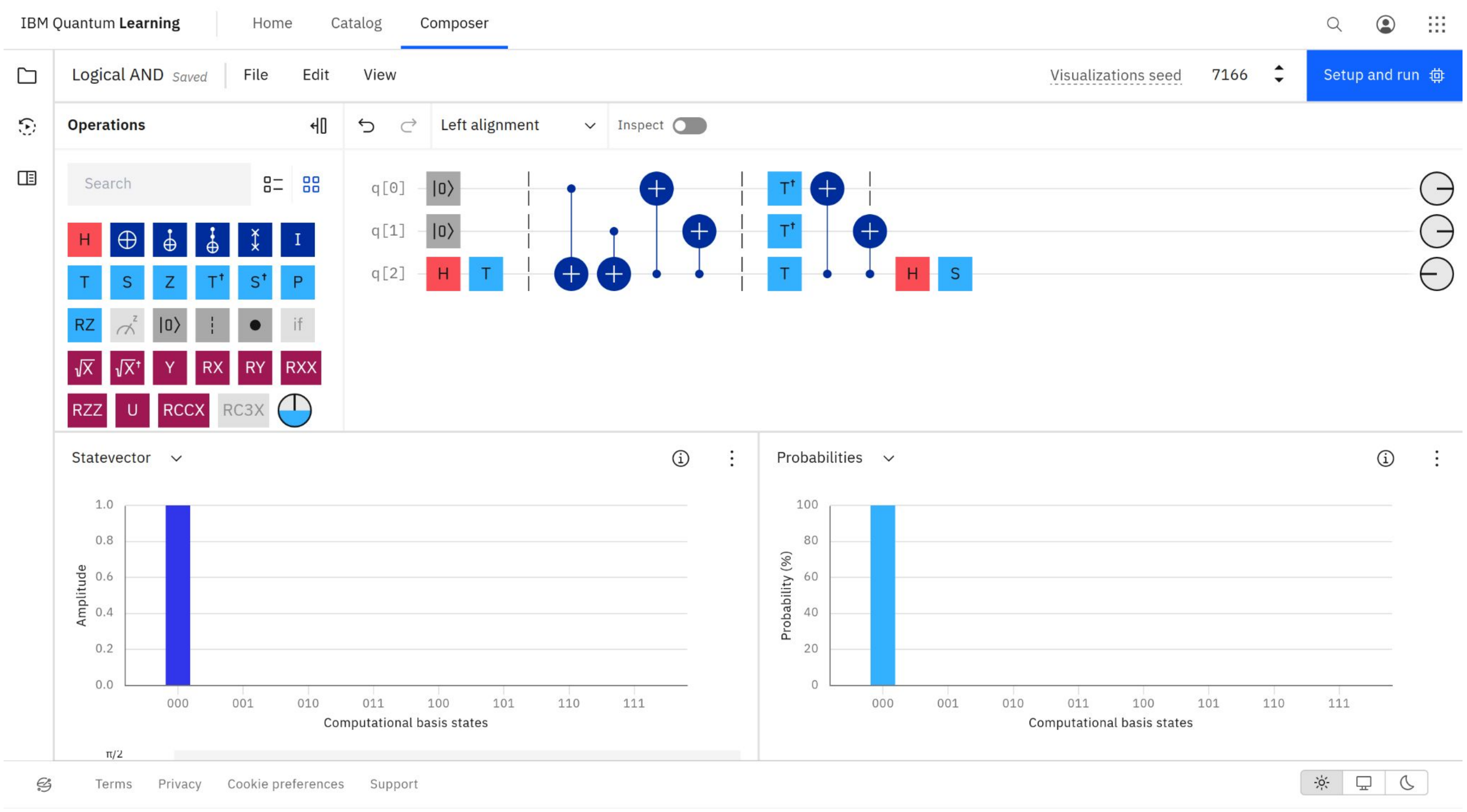
- Can also be simulated with the Qiskit libraries.



```
gate lAND a, b, c {
  cx a, c;
  cx b, c;
  cx c, a;
  cx c, b;
  barrier a;
  t c;
  tdg a;
  tdg b;
  cx c, b;
  cx c, a;
  h c;
  s c;
}

qreg q[14];
creg c[1];
```

IBM Quantum Learning

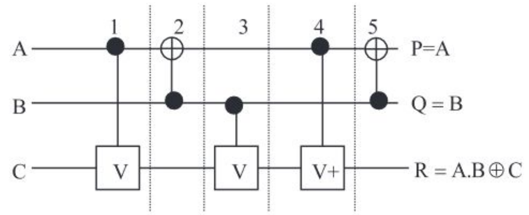


SAPIENZA
UNIVERSITÀ DI ROMA

Nagamani

2014

SAPIENZA
UNIVERSITÀ DI ROMA



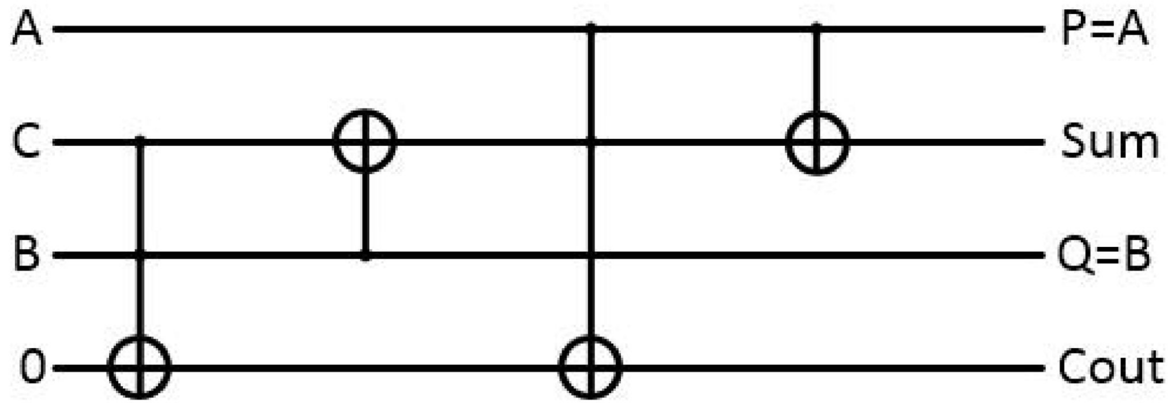
(b) quantum implementation of Toffoli gate

Introduction

- This work proposes a reversible quantum ripple-carry adder, optimized for quantum cost and delay and with zero garbage outputs.
- An operation is said to be *physically* reversible if it converts no energy to heat and produces no entropy.
- A reversible gate has the same number of input and output lines *i.e.*, for each output pattern there must be a unique input pattern.
- The Quantum Cost of a *Positive* controlled Toffoli gate is 5 and the delay associated with it is 5Δ.
- The Quantum Cost of a *Negative* controlled Toffoli gate is 6 with a delay of 6Δ.

1-bit Reversible Full Adder

- The base module for this ripple-carry adder is the following *1-bit adder*:



- This block computes the sum and the carry out as

  Sum = $C \oplus B \oplus A$ Cout = $0 \oplus (B \wedge C) \oplus (A \wedge (C \oplus B))$

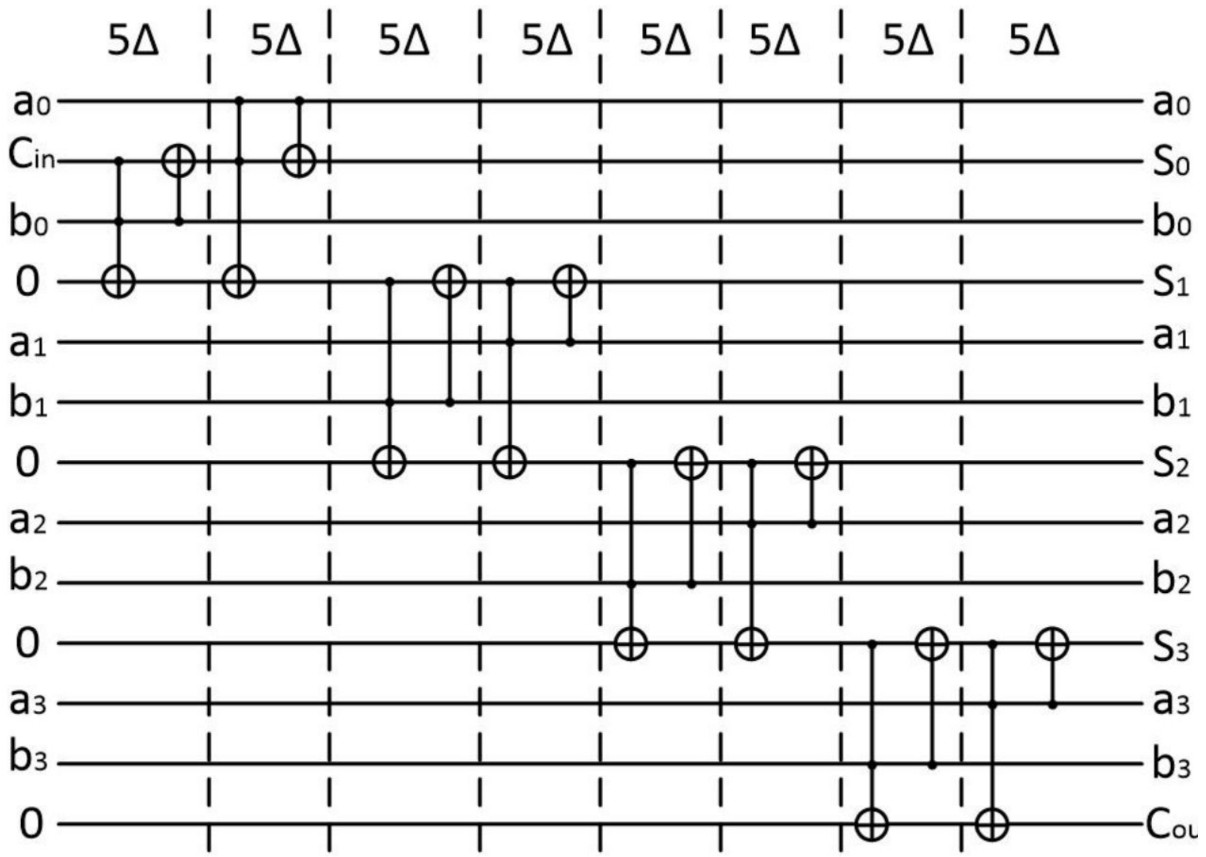
- The proposed design is simple and has
  - 4 gate count
  - 0 garbage outputs.

n-bit Reversible Full Adder

- The methodology adapted is to ripple the carry through each stage of full adder circuits.
- The proposed 1-bit full adder circuit has been cascaded to work as a 4-bit reversible full adder.
- For each stage of input, 1 ancilla input is required producing no garbage output.
- The quantum cost of the proposed design is **12*n*** with a delay of **10*n*Δ**.

SAPIENZA
UNIVERSITÀ DI ROMA

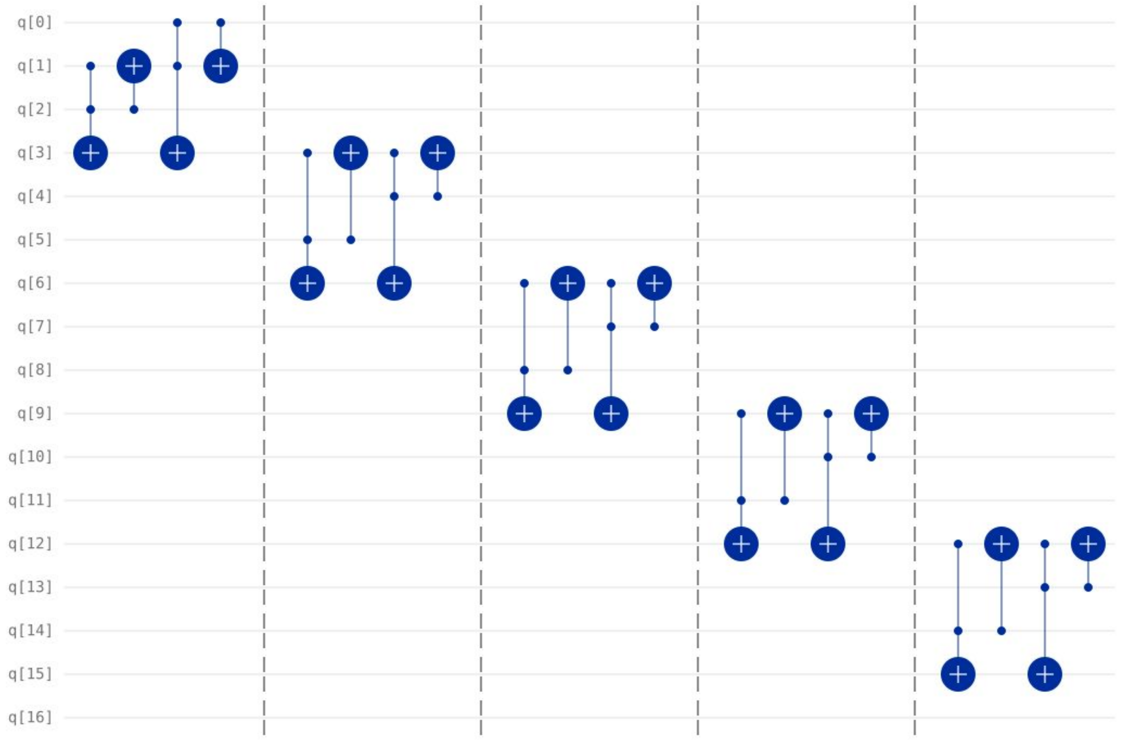
4-bit Reversible Full Adder (2)



SAPIENZA
UNIVERSITÀ DI ROMA

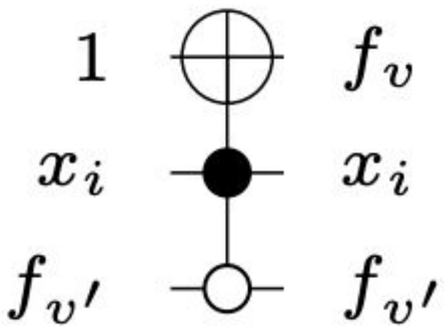
5-bit Reversible Full Adder (3)

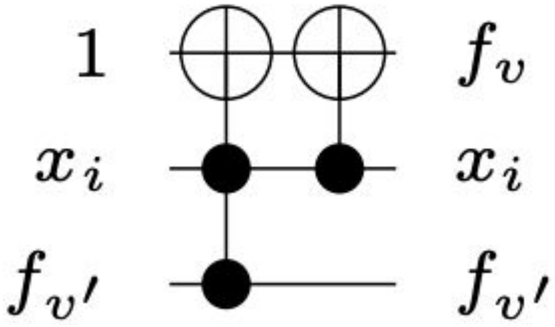
- **QC:** 12n = 60
- **Delay:** 10$n\Delta$ = 50Δ



Negative Control Line Toffoli

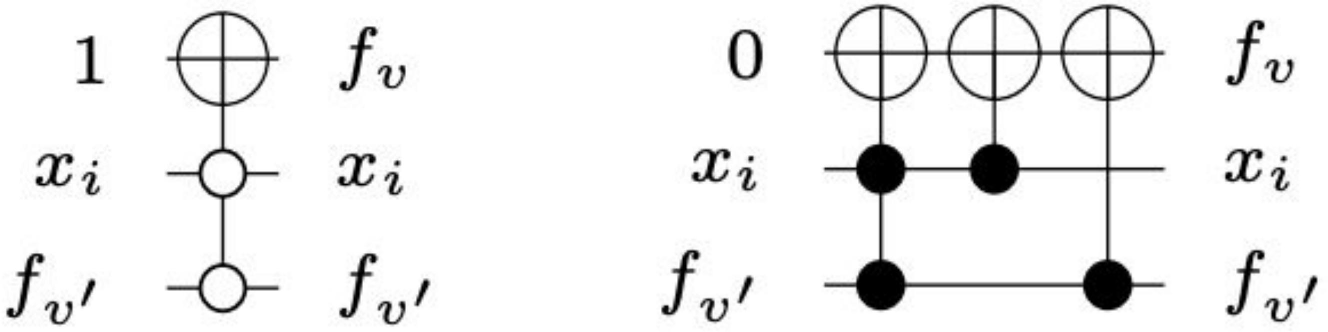
- From Schönborn et al. (2014)





Negative Control Line Toffoli (2)

- From Schönborn et al. (2014)

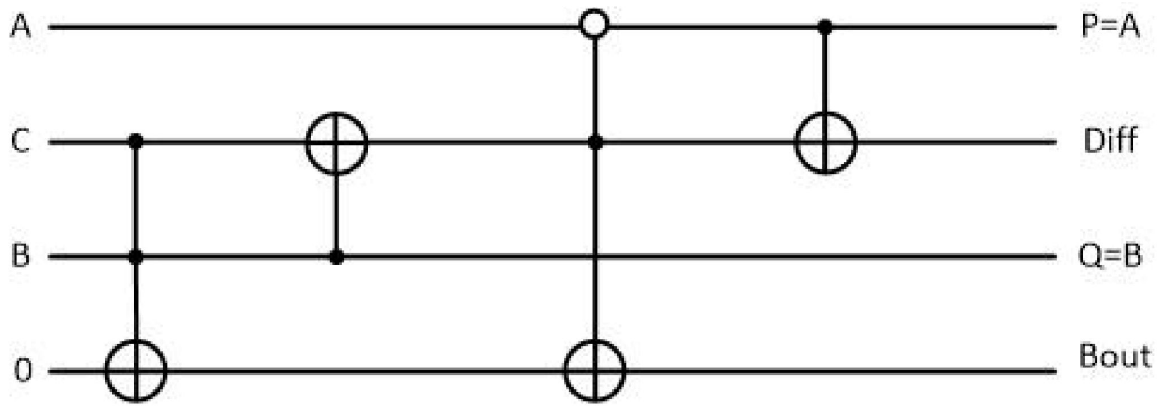


SAPIENZA
UNIVERSITÀ DI ROMA

n-bit Reversible Full (unsigned) Subtractor

- Designed using the concept of negative control lines. This allows replacing series of gates and thereby reducing the area, gate count and as well as power consumption.

- All parameters such as ancilla inputs an garbage outputs remain the same as the full adder.



- The borrow-out formula is:

  Bout = 0 $\oplus$ (C $\wedge$ B) $\oplus$ ($\neg$A $\wedge$ (C $\oplus$ B))

SAPIENZA
UNIVERSITÀ DI ROMA

Practical Example

- The borrow-out formula is:

$$o_i = 0 \oplus (b_i \land o_{i-1}) \oplus (\neg a_i \land (b_i \oplus o_{i-1}))$$

- *Example:*

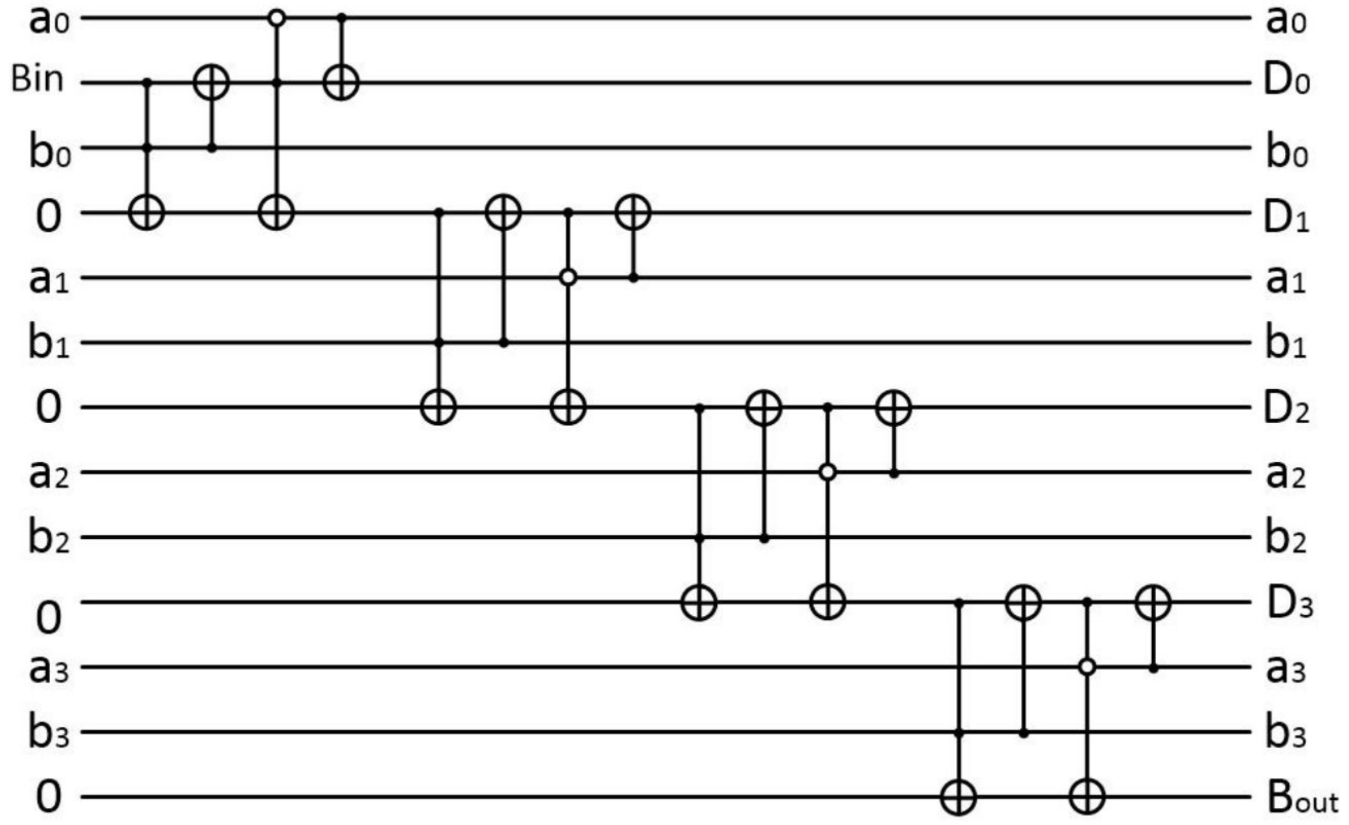
$$A = 1\ 0\ 1$$
$$B = 0\ 1\ 0$$

$a_0 - b_0 = a_0 \oplus b_0 = 1$ $\qquad o_0 = 0$

$a_1 - b_1 = a_1 \oplus b_1 \oplus o_0 = 1$ $\qquad o_1 = 1$

$a_2 - b_2 = a_2 \oplus b_2 \oplus o_1 = 0$ $\qquad o_2 = 0$

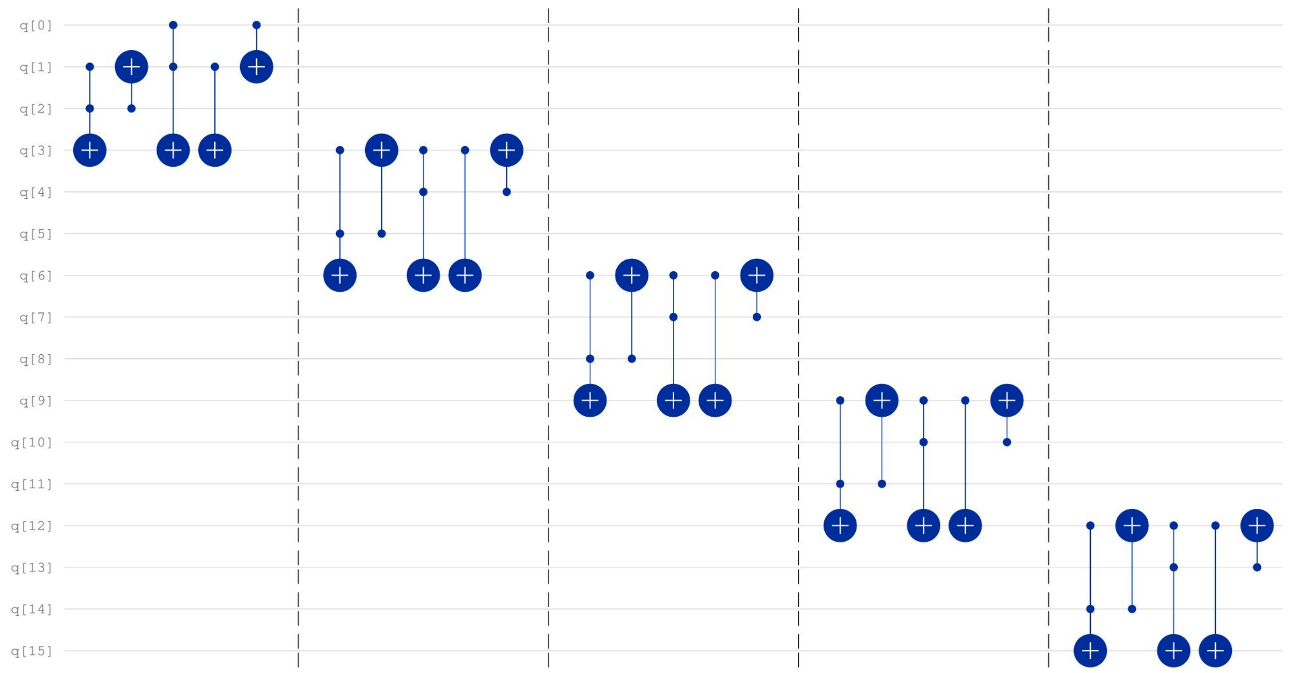
4-bit Subtractor



SAPIENZA
UNIVERSITÀ DI ROMA

5-bit Subtractor (2)

- **QC:** $13n = 65$
- **Delay:** $11n\Delta = 55\Delta$



SAPIENZA
UNIVERSITÀ DI ROMA

Gidney

2018

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

- This adder makes use of *temporary logical-AND* blocks.
- The goal is to minimize the overall T-count for the addition, since this reduces the T-count of any construction based on addition.
- This adder requires a high number of ancilla bits which might hinder its implementation in small systems.
- This adder is fault tolerant.

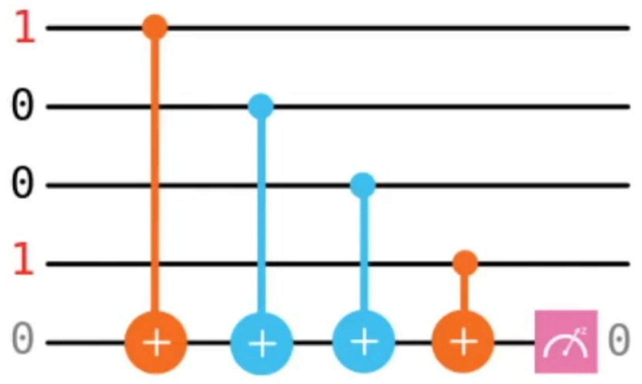
SAPIENZA
UNIVERSITÀ DI ROMA

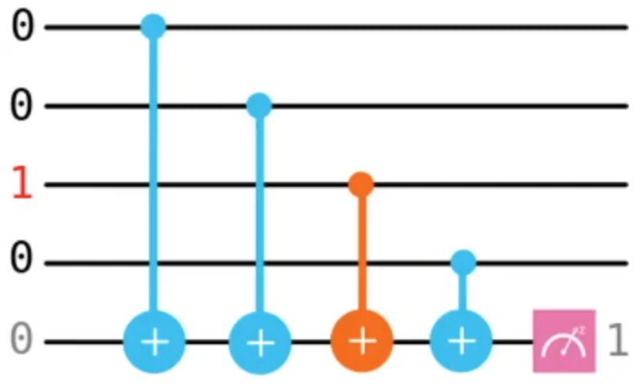
Surface Codes

- Surface codes are a likely necessity in future quantum computers.
- They involve the arrangement of qubits into two-dimensional lattices.
- There are different surface code families with distinct properties and applications.
- Surface Codes allow for quantum error correction.
- T gates dominate the cost of quantum computation based on the surface code
- For this reason temporary logical-ANDs can represents a significant reduction in projected costs of quantum computation.

Surface Codes (2)

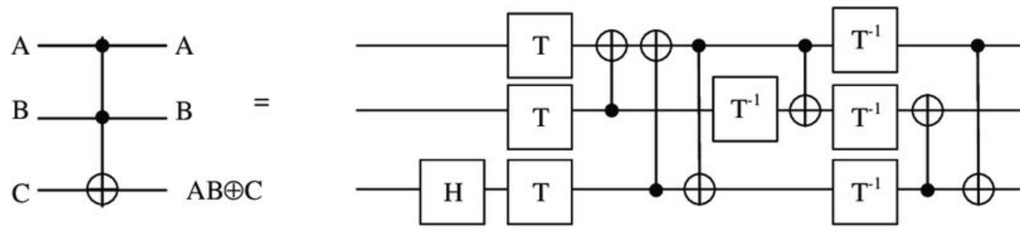
- A simple example is grouping the qubits in plaquettes.
- And perform a parity check for each plaquette to detect errors:





- The subsequent error correction phase requires many Toffoli gates.
  - Beyond the scope of this presentation.

SAPIENZA
UNIVERSITÀ DI ROMA



Phase Errors

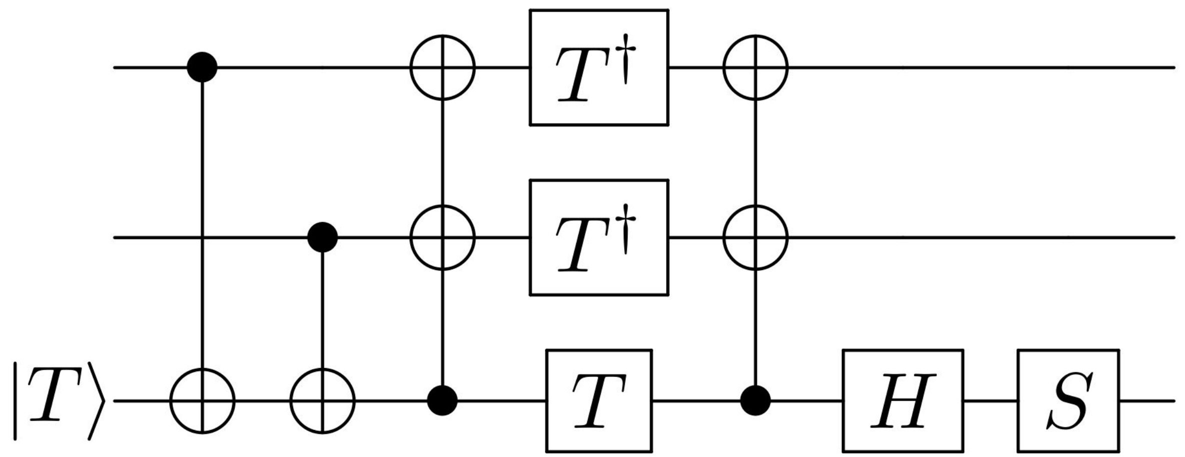
- The textbook construction of a Toffoli gate uses seven T gates.
- When a Toffoli operation is later un-computed by a second Toffoli operation, each gate in the pair can omit three of the T gates from the textbook construction.
- This introduces phase errors but the second Toffoli gate can uncompute the errors.
- Assuming intermediate operations aren’t sensitive to the phase errors
- It is proven that a Toffoli gate requires at least 4 T gates, but it might be possible that *n* Toffolis could be implemented with fewer than *4n* T gates for some *n*.

SAPIENZA
UNIVERSITÀ DI ROMA

logical-AND construction

- The computation circuit has a T-count of 4.

- The $|T\rangle$ state input contributes to the T-count.

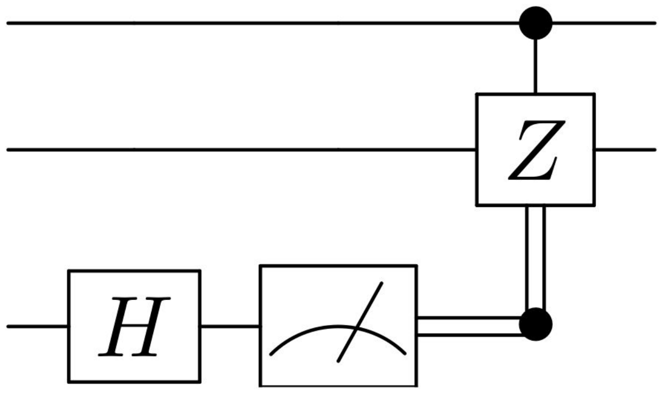


- *Practical example...*

SAPIENZA
UNIVERSITÀ DI ROMA

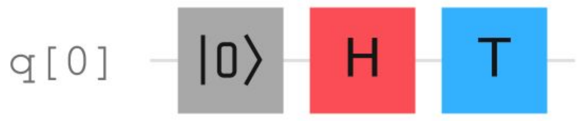
logical-AND uncomputation

- The uncomputation requires only Clifford gates.
- Has a T-count of zero.
- Faster than the logical-AND computation.
- Can also be done with the same gates as before.



$|T\rangle$

- Distilling this state can be an issue as it is a potential bottleneck since there is no cheap mechanism to apply non-Clifford operations such as T gates.
- $|T\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$
- On *Qiskit* it can be generated with:

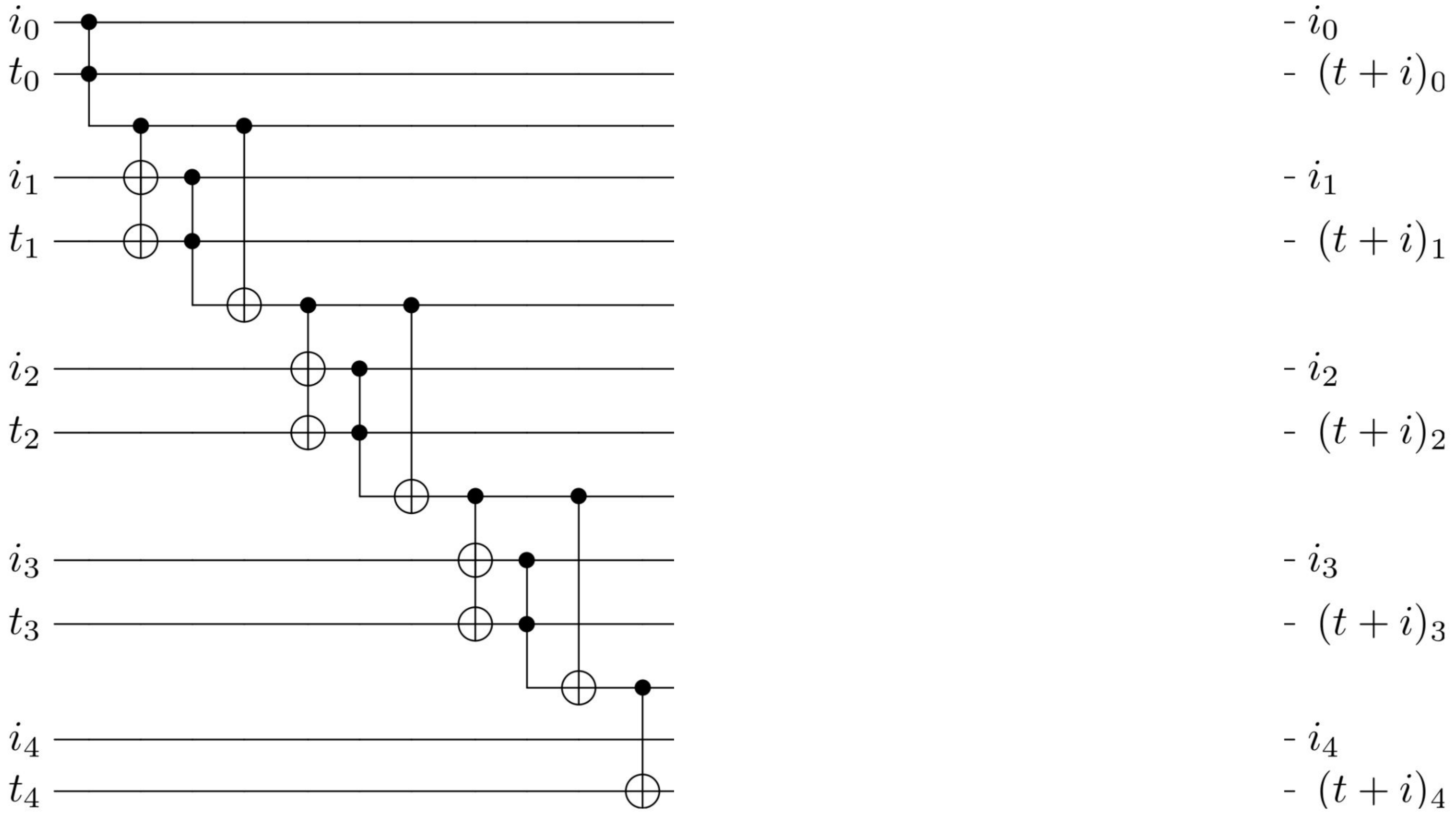


5-bit Adder Construction

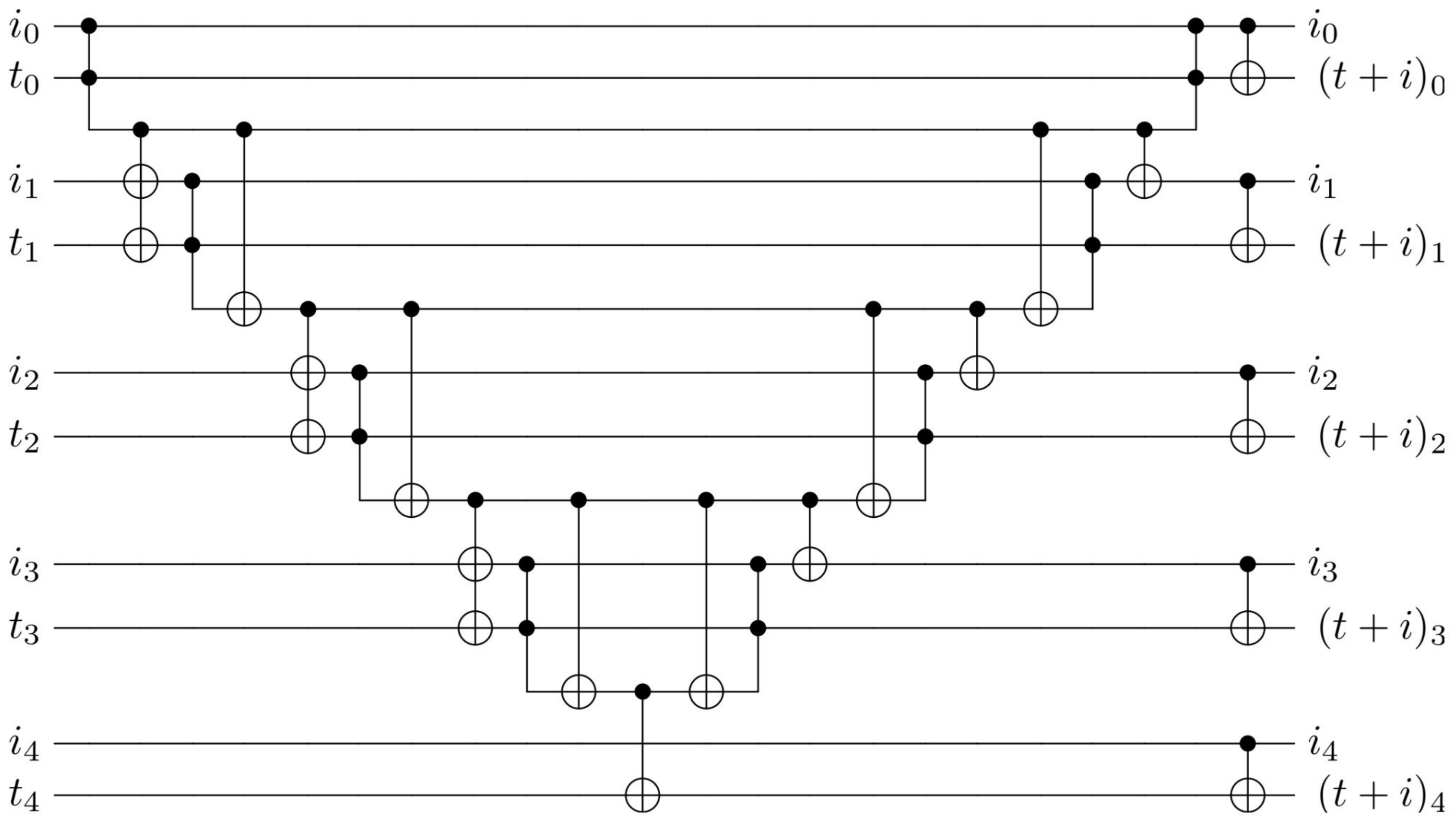
i_0 -	- i_0
t_0 -	- $(t+i)_0$
i_1 -	- i_1
t_1 -	- $(t+i)_1$
i_2 -	- i_2
t_2 -	- $(t+i)_2$
i_3 -	- i_3
t_3 -	- $(t+i)_3$
i_4 -	- i_4
t_4 -	- $(t+i)_4$

SAPIENZA
UNIVERSITÀ DI ROMA

5-bit Adder Construction (2)



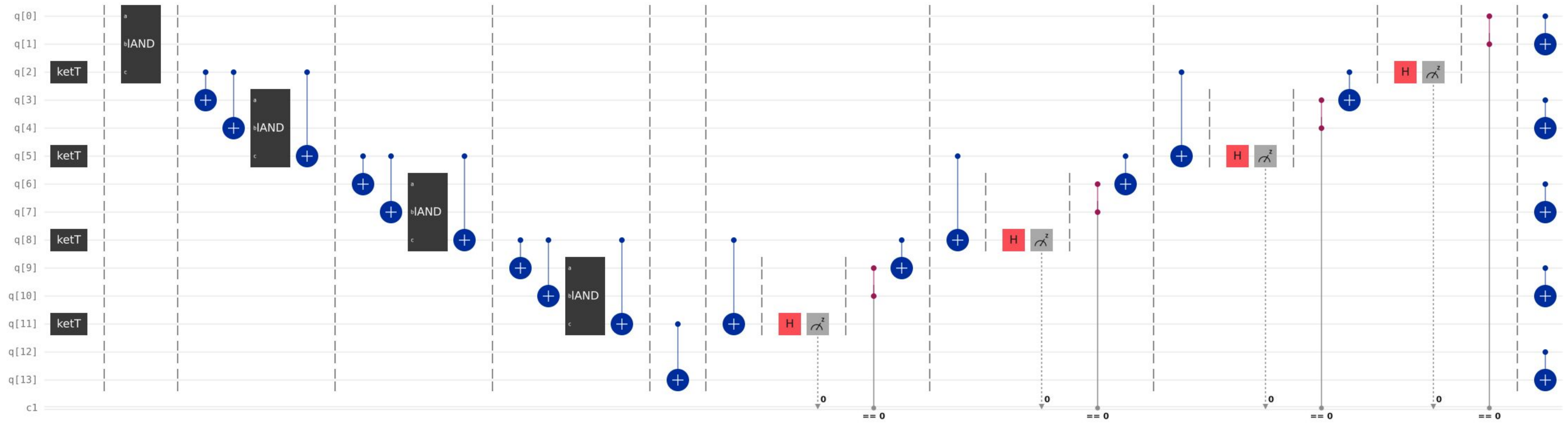
5-bit Adder Construction (3)



SAPIENZA
UNIVERSITÀ DI ROMA

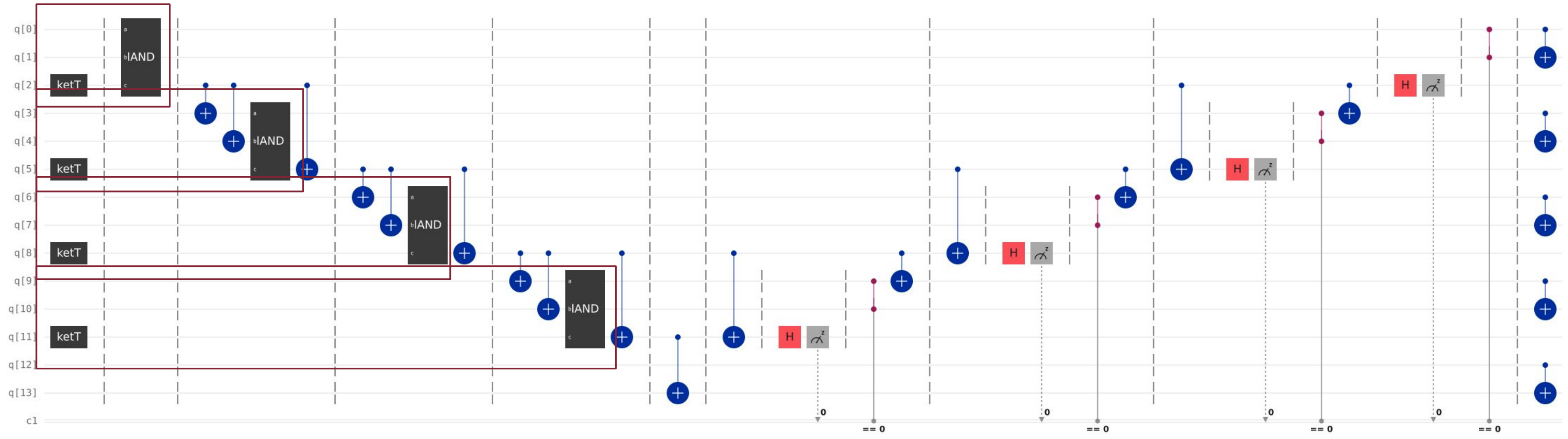
5-bit Addition

- ***T count***:
- ***QC***:
- ***Delay***:



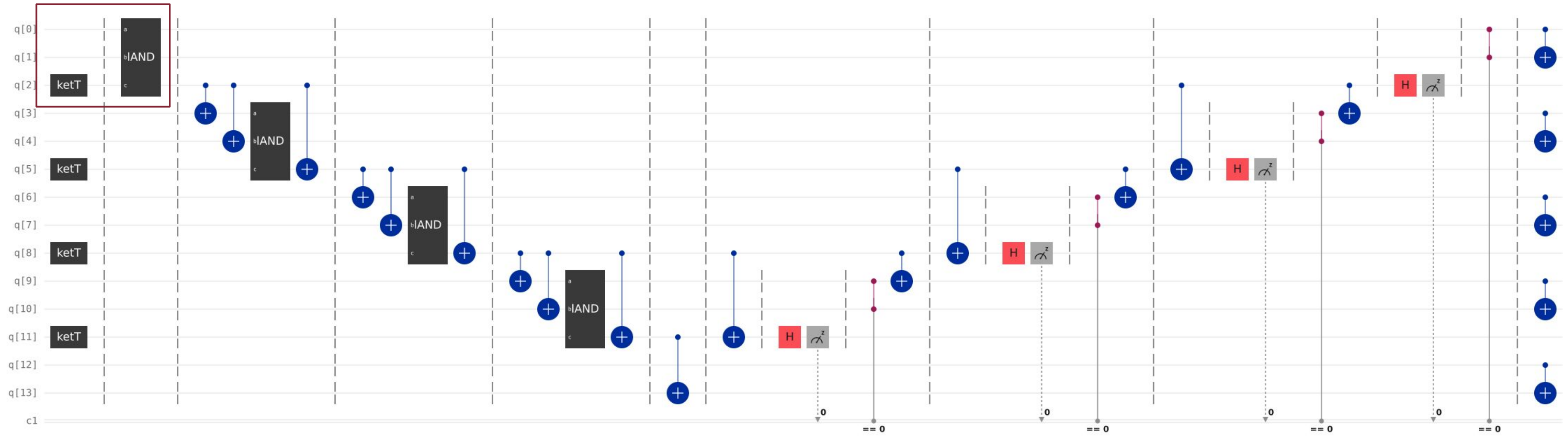
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***:
- ***Delay***:



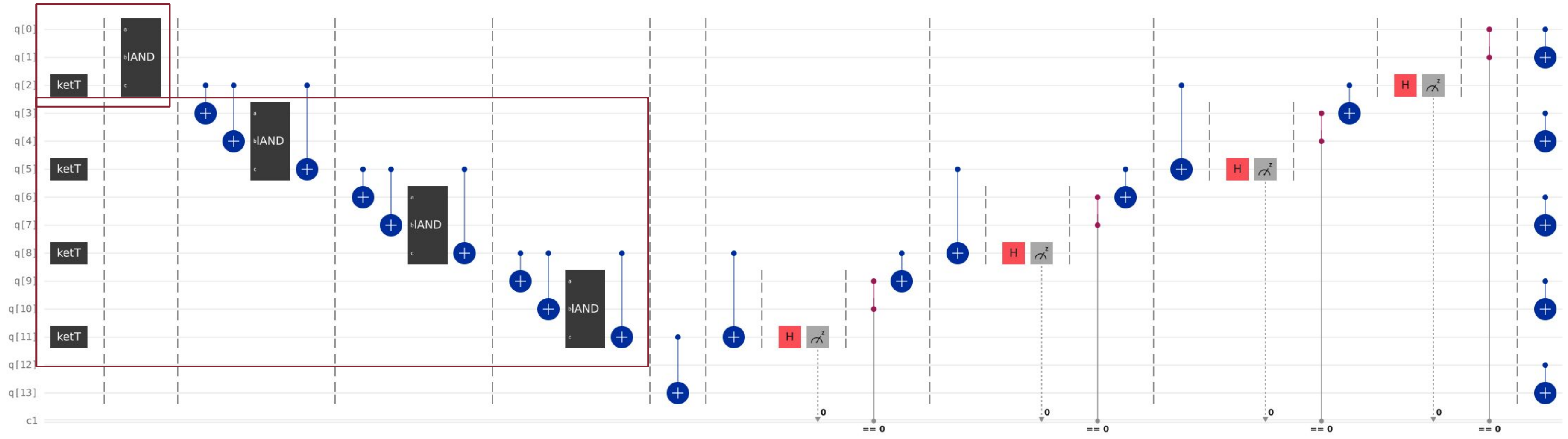
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 13
- ***Delay***:



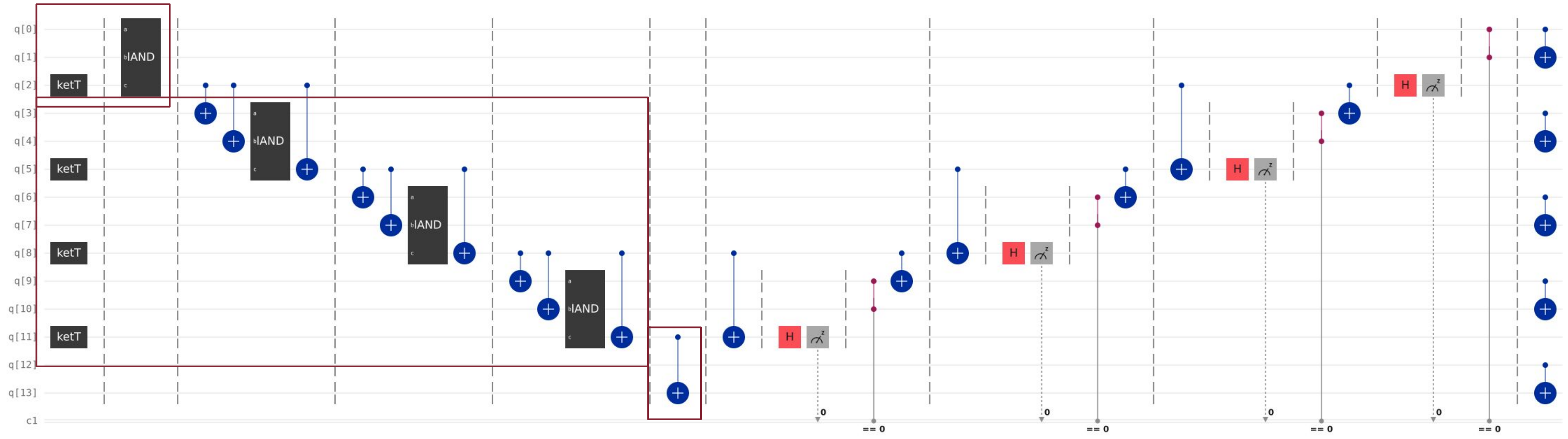
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 13 + 48
- ***Delay***:



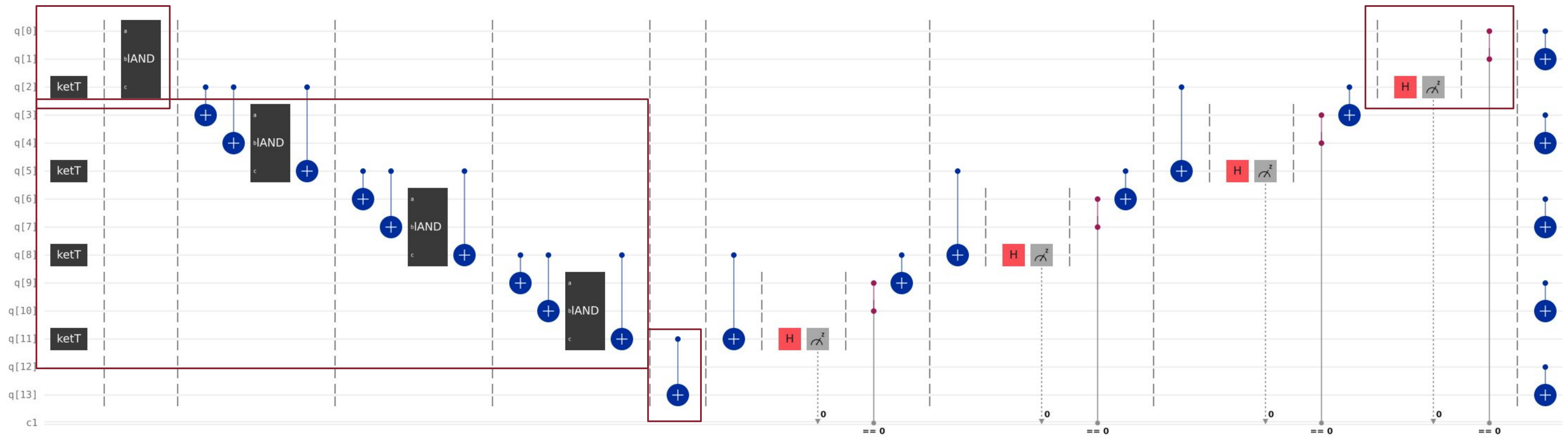
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 13 + 48 + 1
- ***Delay***:



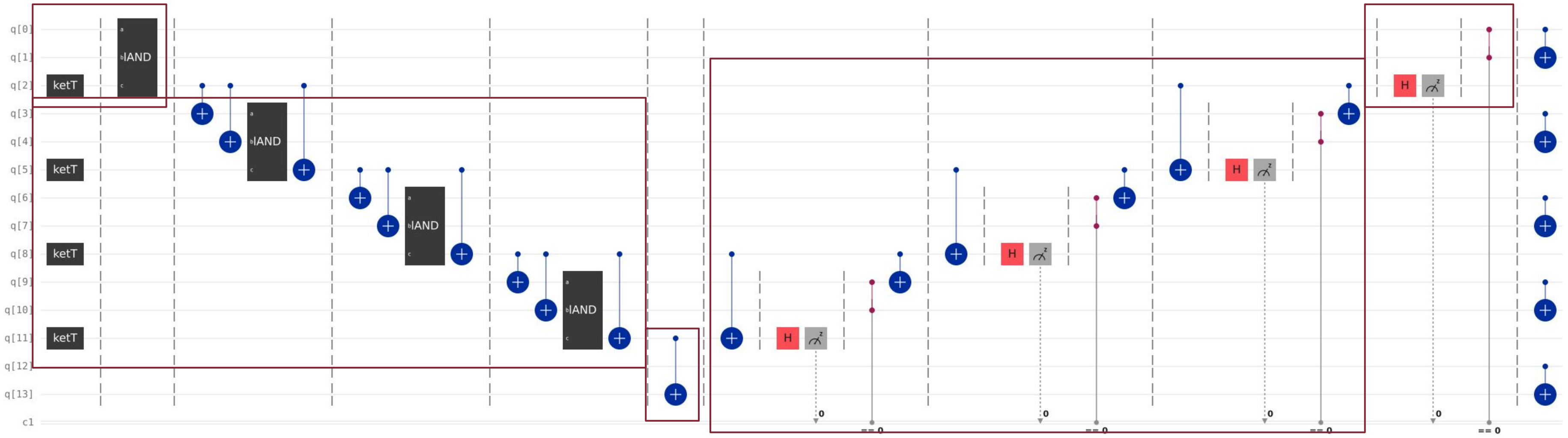
5-bit Addition

- ***T count***: 4(n-1) = 16
- ***QC***: 13 + 48 + 1 + 3
- ***Delay***:



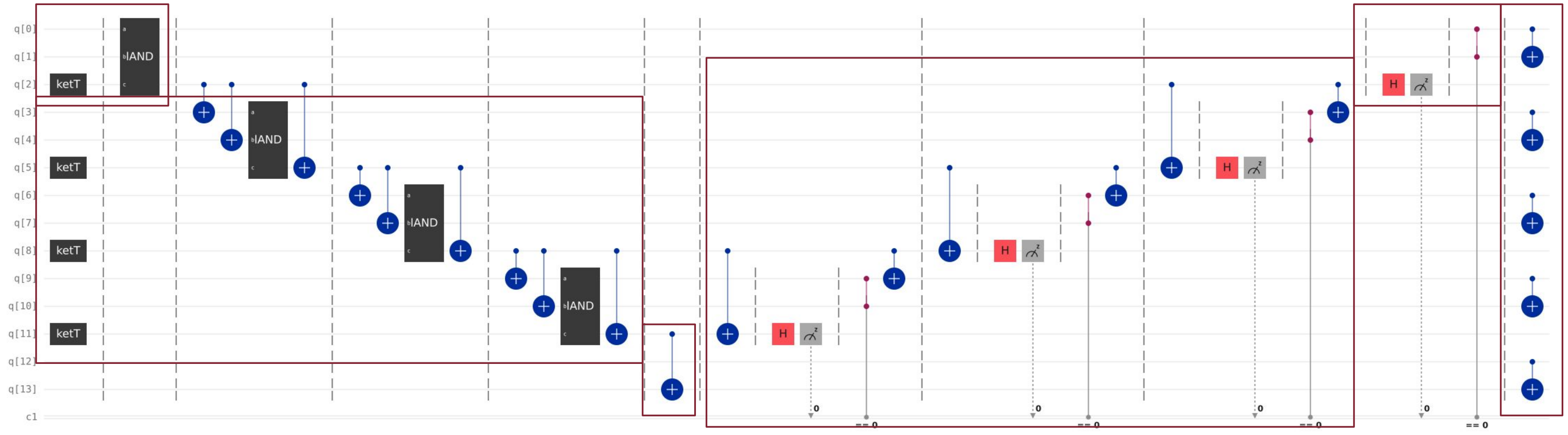
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 13 + 48 + 1 + 3 + 15
- ***Delay***:



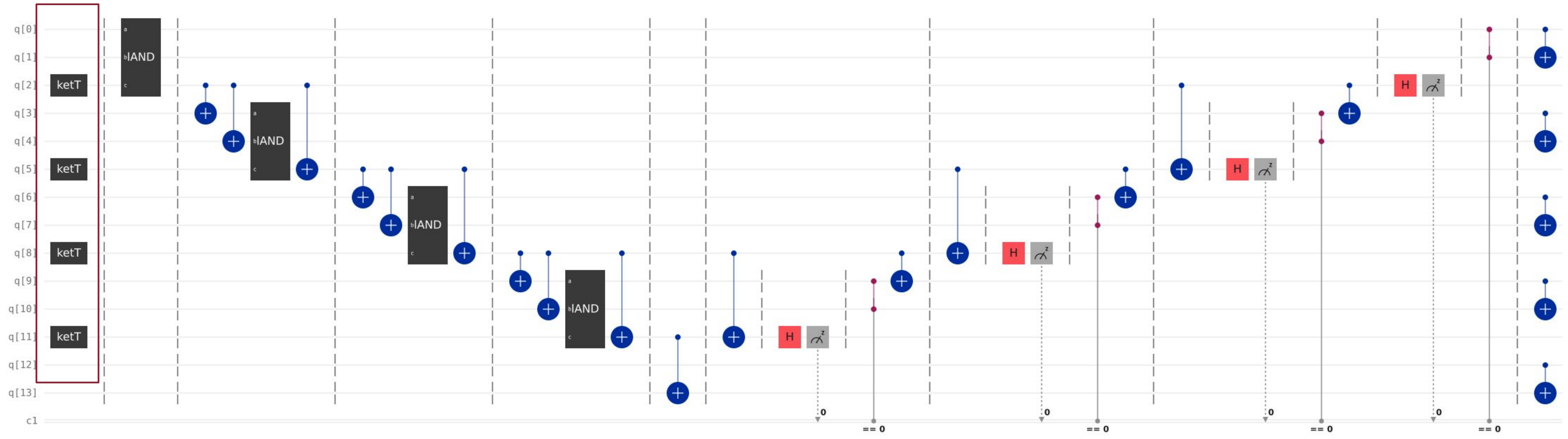
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 13 + 48 + 1 + 3 + 15 + 5 = 85
- ***Delay***:



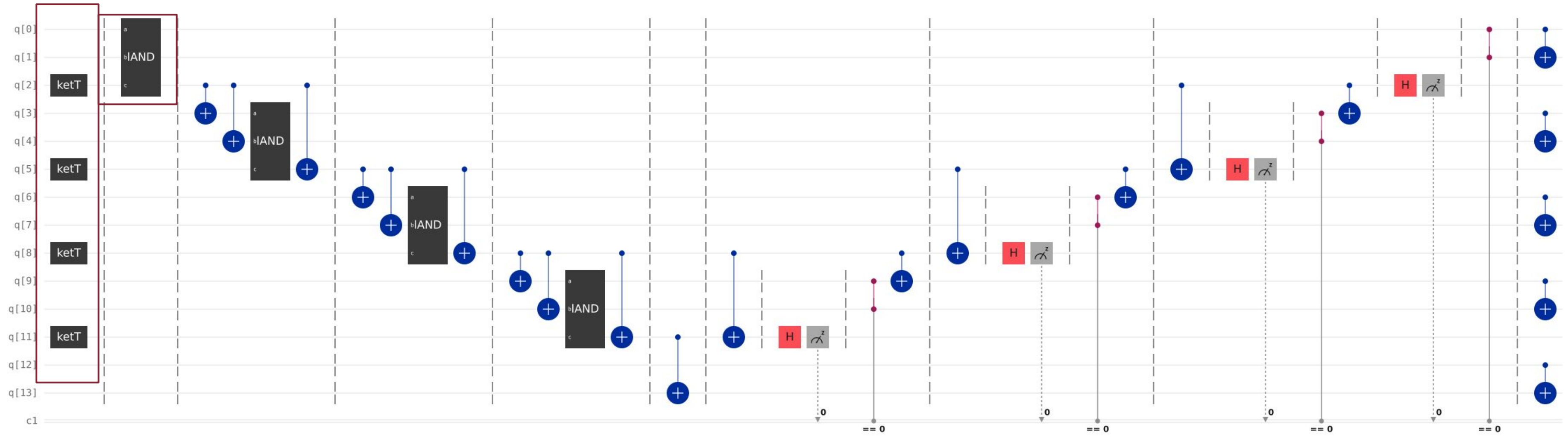
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 85
- ***Delay***: 2



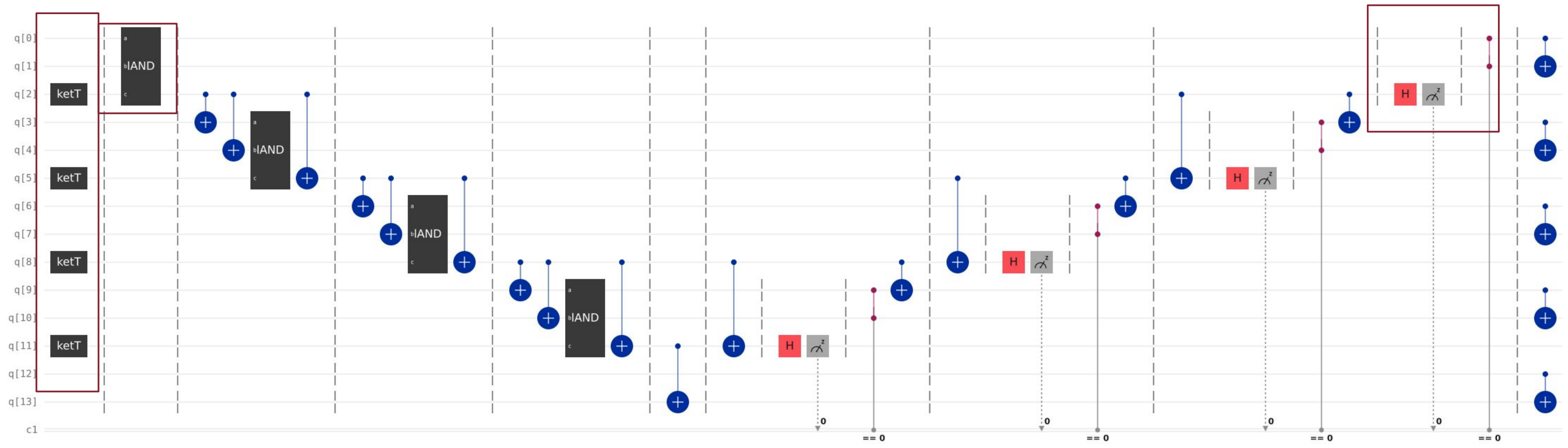
5-bit Addition

- ***T count***: $4(n\text{-}1) = 16$
- ***QC***: 85
- ***Delay***: 2 + 7



5-bit Addition

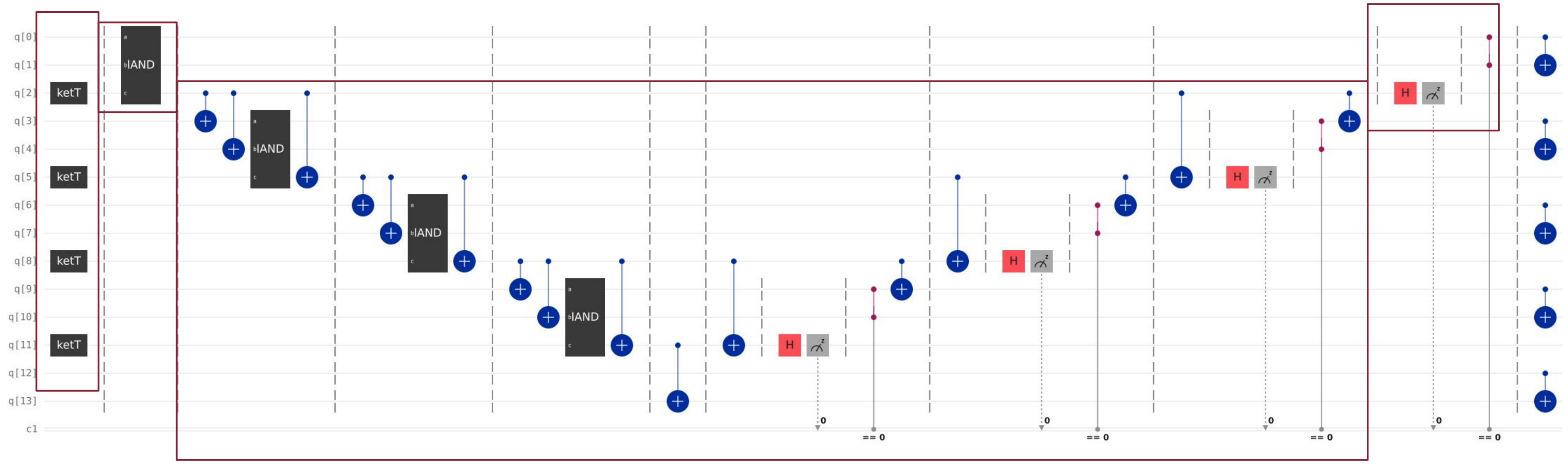
- ***T count***: $4(n\text{-}1) = 16$
- ***QC***: 85
- ***Delay***: 2 + 7 + 3



SAPIENZA
UNIVERSITÀ DI ROMA

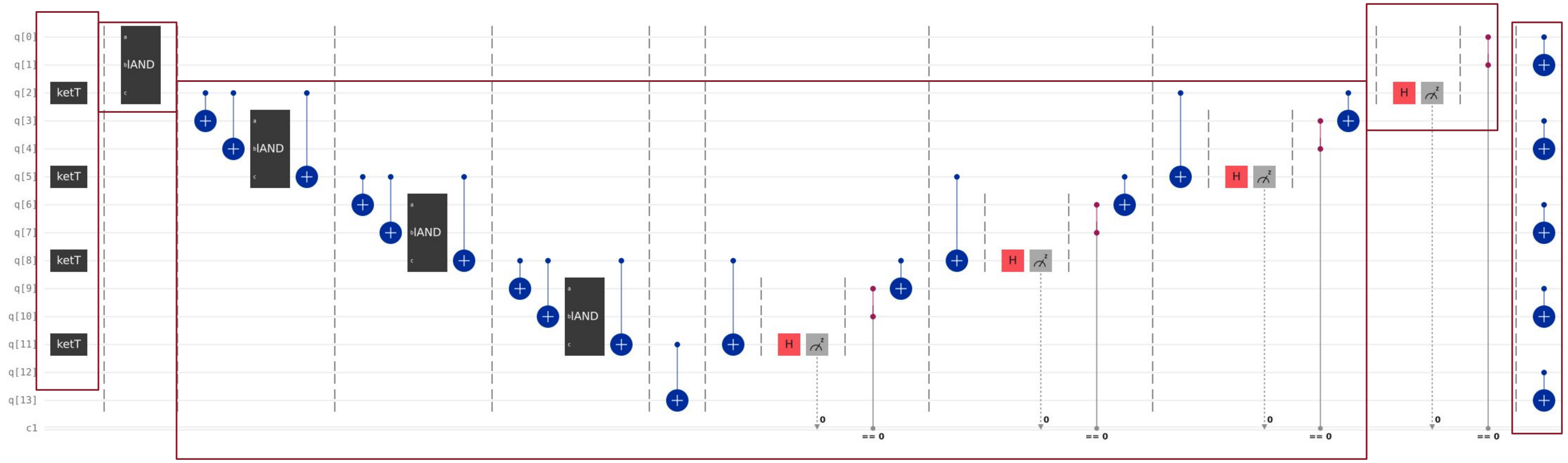
5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 85
- ***Delay***: 2 + 7 + 3 + 46



5-bit Addition

- ***T count***: 4(*n*-1) = 16
- ***QC***: 85
- ***Delay***: 2 + 7 + 3 + 46 + 1 = 59



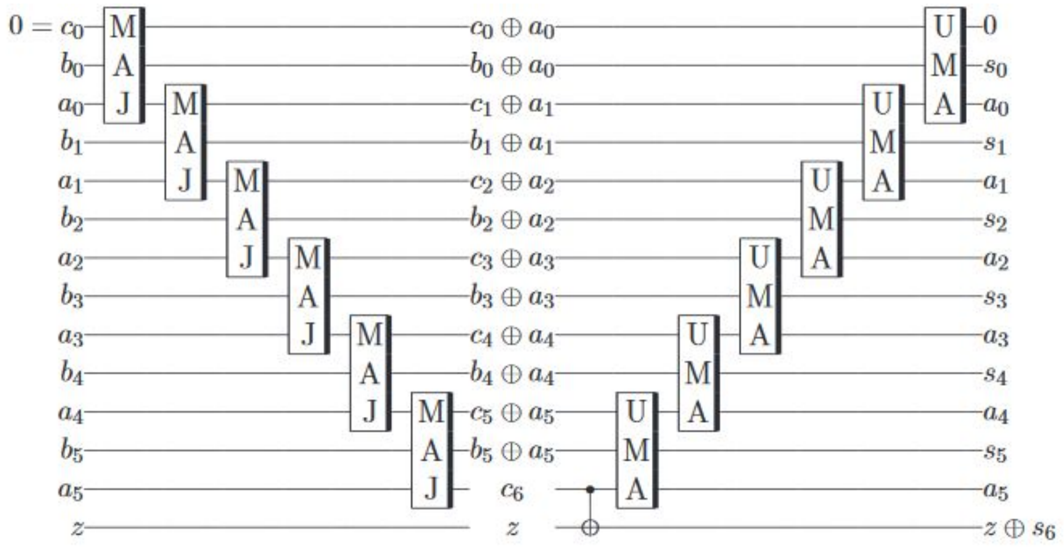
Sapienza
Università di Roma

Takahashi

2010

Introduction

- The work implements a quantum circuit for the addition of two binary numbers.
- The circuit follows a ripple carry approach and has O(n) depth and O(n) size
- It requires 2N+1 qubits with N being the number of bits of the addends
  - No ancilla or garbage bits!
- It is partially based on the following adder by Cuccaro et al. (2004)



SAPIENZA
UNIVERSITÀ DI ROMA

The Main Idea

- The problem of constructing quantum adder can be expressed as

$$\left(\bigotimes_{i=0}^{n-1} |b_i\rangle|a_i\rangle\right)|z\rangle \rightarrow \left(\bigotimes_{i=0}^{n-1} |s_i\rangle|a_i\rangle\right)|z \oplus s_n\rangle$$

- The Ripple Carry Approach requires the computation of the carry bits c_i

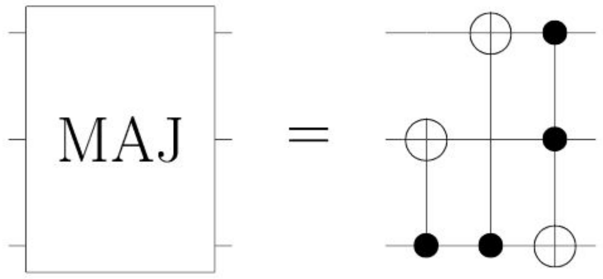
$$c_i = \begin{cases} 0 & i = 0, \\ \mathrm{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) & 1 \le i \le n \end{cases}$$ (MAJ(a, b, c) = ab ⊕ bc ⊕ ca)

- And by having the carry we can compute s_i

$$s_i = \begin{cases} a_i \oplus b_i \oplus c_i & 0 \le i \le n-1, \\ c_n & i = n. \end{cases}$$

The Main Idea

- Cuccaro uses an ancillary bit to store c_0 and uses the following gate in the circuit to produce the carry:



- In order to avoid using the ancilla, the Takahashi divides the MAJ gates in 2 parts:
  - The 2 CNOT's
  - The Toffoli gate

The Main Idea

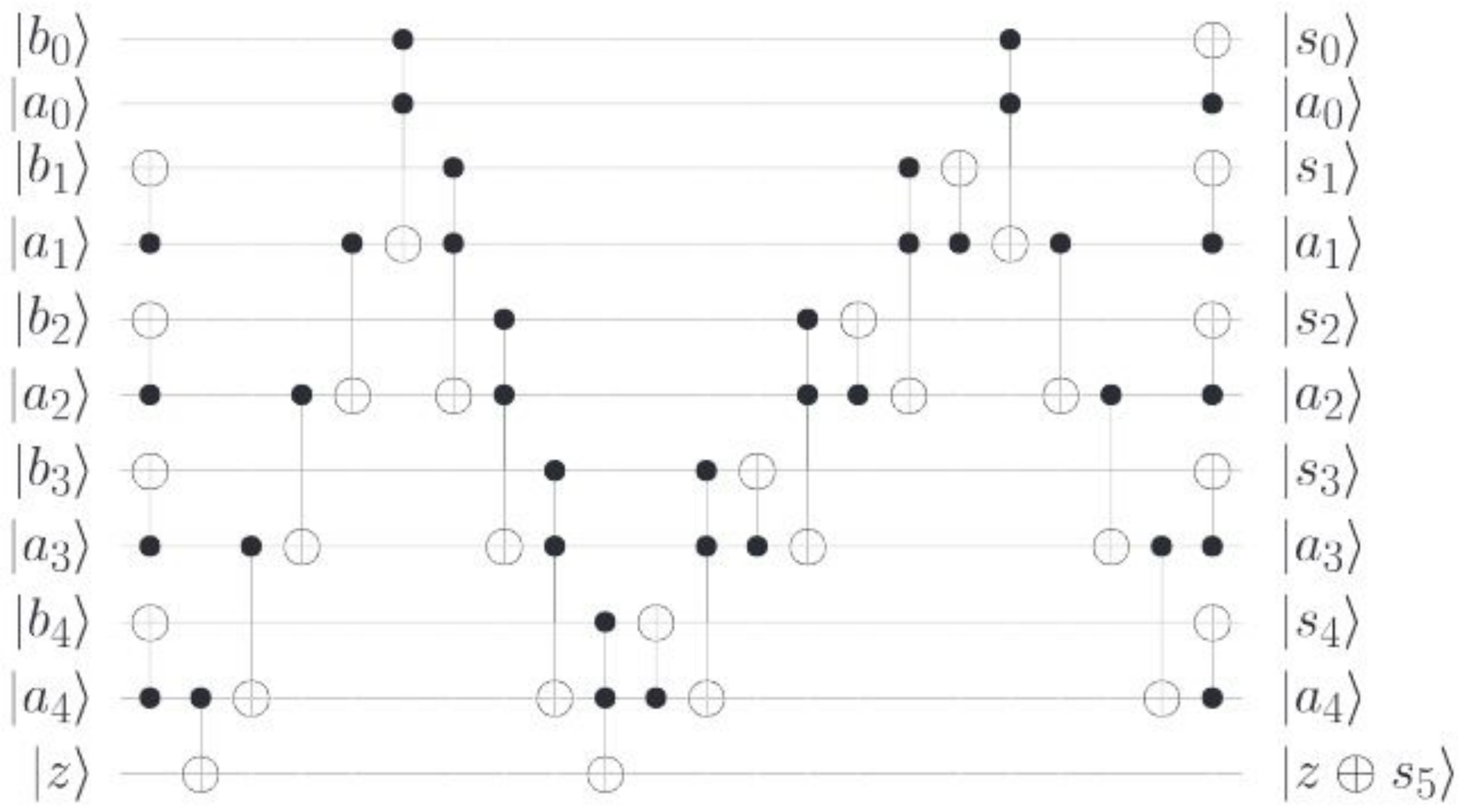
- using CNOT gates (the first parts of the MAJ gate) and a Toffoli gate, we first prepare the following state

$$|b_1 \oplus a_1\rangle|a_1 \oplus c_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus a_i\rangle|a_i \oplus a_{i-1}\rangle\right) |z \oplus a_{n-1}\rangle.$$

- By applying CCNOT gates, we can compute c_i and store it in the qubit initially storing a_i. The final Toffoli gate computes the carry out c_n and stores it in the qubit for z.
- The CCNOT maps
  - $|b_i \oplus a_i\rangle|a_i \oplus c_i\rangle|a_{i+1} \oplus a_i\rangle$ to $|b_i \oplus a_i\rangle|a_i \oplus c_i\rangle|a_{i+1} \oplus c_{i+1}\rangle$
- Then we first compute the $b_i \oplus c_i$ (with c_i stored in the qubit of a_i)
- Finally we bring back a_i and do the last sum to obtain the various s_i

SAPIENZA
UNIVERSITÀ DI ROMA

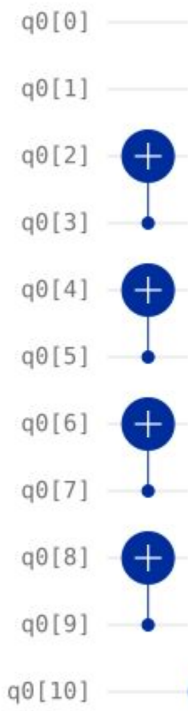
Takahashi - Circuit (N = 5)



SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 1:
  - apply CNOT gates to each B_i and A_i for i=n-1,...,1 with A_i as control

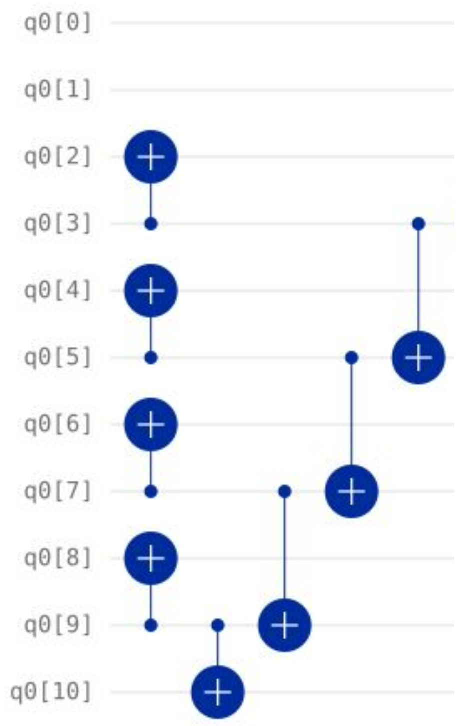


The system state is now represented by this model:

$$|b_0\rangle|a_0\rangle\left(\bigotimes_{i=1}^{n-1}|b_i \oplus a_i\rangle|a_i\rangle\right)|z\rangle$$

Step By Step Creation

- Step 2:
  - apply CNOT gates to each A_{i+1} and A_i for n=1,...,n-1 with A_i as control



SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 2:
  - apply CNOT gates to each A_{i+1} and A_i for n=1,...,n-1 with A_i as control

- The CNOT is in fact not applied to the first two qubits A_0 and A_1 and the state evolves to the following:

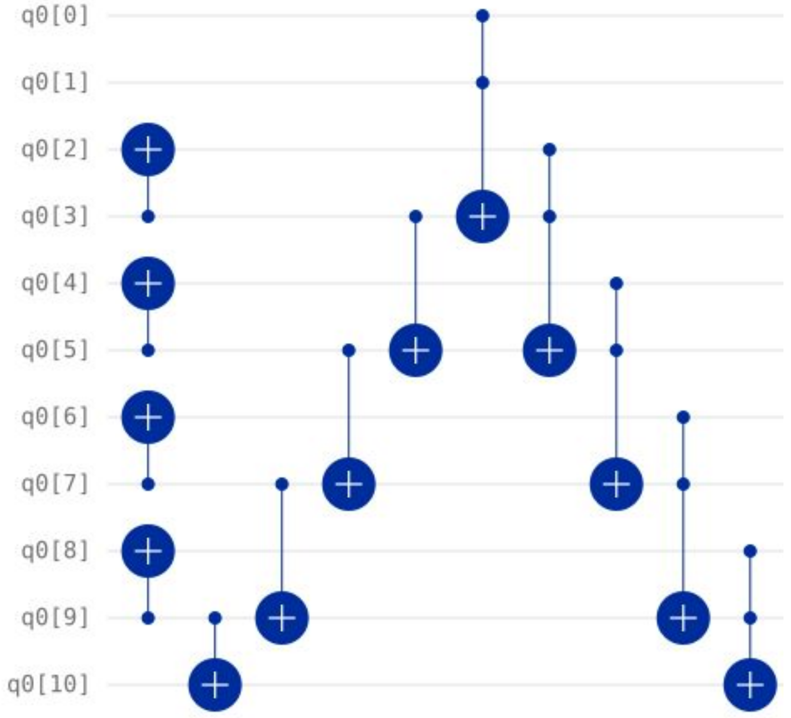
$$|b_0\rangle|a_0\rangle|b_1 \oplus a_1\rangle|a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus a_i\rangle|a_i \oplus a_{i-1}\rangle \right) |z \oplus a_{n-1}\rangle$$

- Thus we have completed the first portion of the decomposed MAJ gate

SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 3:
  - apply CCNOT gates to each B_i, A_i and A_{i+1} for n=0,...,n-1 with A_{i+1} as target



SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 3:
  - apply CCNOT gates to each B_i, A_i and A_{i+1} for n=0,...,n-1 with A_{i+1} as target

$$|b_0\rangle|a_0\rangle|b_1 \oplus a_1\rangle|a_1 \oplus c_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus a_i\rangle|a_i \oplus a_{i-1}\rangle\right) |z \oplus a_{n-1}\rangle$$

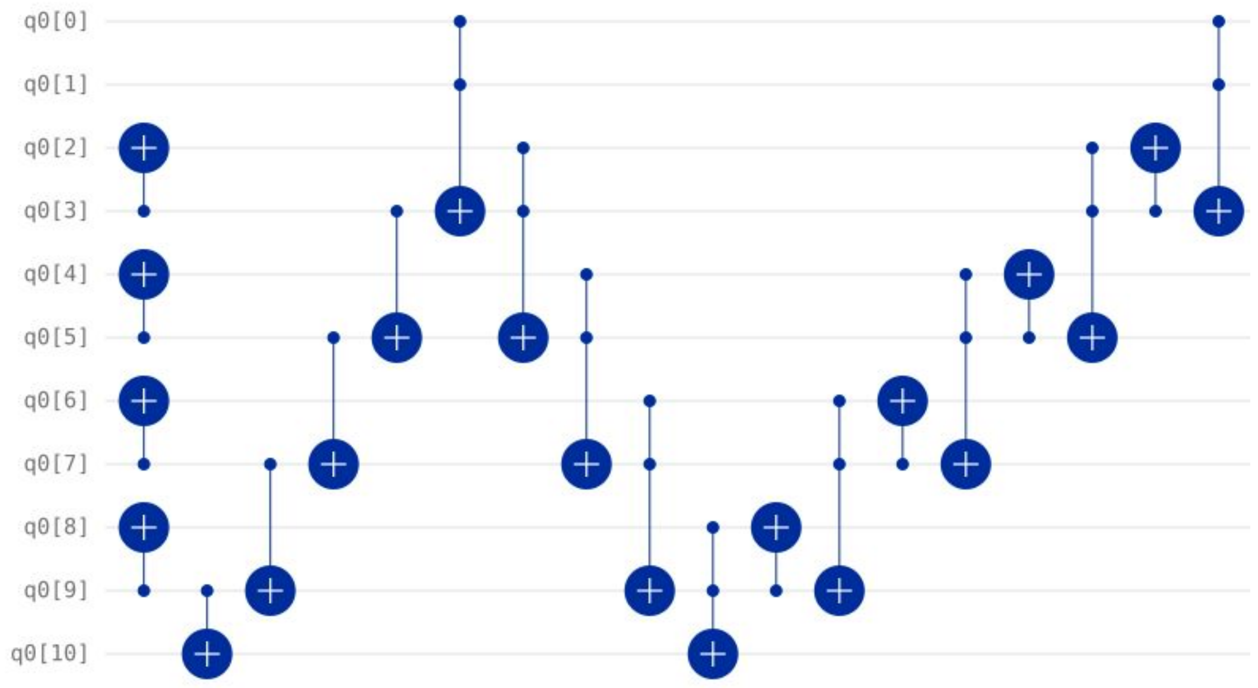
- Then we apply Toffoli to the rest of the A_i qubits:

$$|b_0\rangle|a_0\rangle \left(\bigotimes_{i=1}^{n-1} |b_i \oplus a_i\rangle|a_i \oplus c_i\rangle\right) |z \oplus s_n\rangle$$

- In short we "transfer" the carry c_i to each A_i qubit

Step By Step Creation

- Step 4:
    - Apply CNOT gates to each B_i, A_i for i=n-1,..,1 with A_{i+1} as control
    - Apply CCNOT gates to B_{i-1}, A_{i-1}, A_i for i=n-1,..,1 with A_i as target



SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 4:
  - apply CNOT gates to each B_i, A_i for i=n-1,..,1 with A_{i+1} as control
  - Apply CCNOT gates to B_{i-1}, A_{i-1}, A_i for i=n-1,..,1 with A_i as target

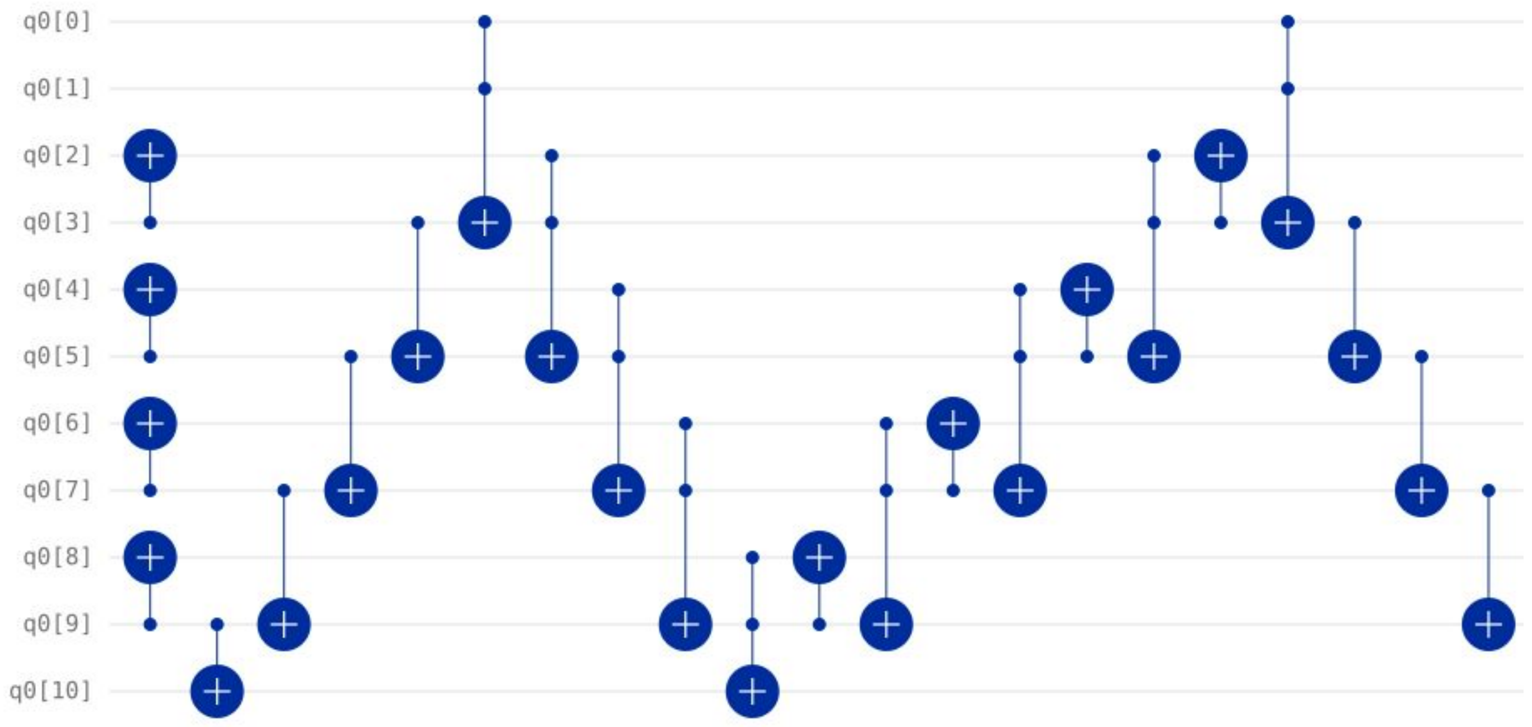
- The CNOT acts on the B_i qubit:
  - $b_i \oplus a_i \to (b_i \oplus a_i) \oplus (a_i \oplus c_i) \to b_i \oplus c_i$
- Then we reverse the application of the first group of Toffoli gates
- Thus we end up with:

$$|b_0\rangle|a_0\rangle|b_1 \oplus c_1\rangle|a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus c_i\rangle|a_i \oplus a_{i-1}\rangle \right) |z \oplus s_n\rangle$$

SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 5:
  - apply CNOT gates to each A_i . A_{i+1} for i=1,...,n-2 with A_{i+1} as target



SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 5: apply CNOT gates to each A_i . A_{i+1} for i=1,...,n-2 with A_{i+1} as target
- With this transformation we bring the state of each A_i to the original value

$$|b_0\rangle|a_0\rangle\left(\bigotimes_{i=1}^{n-1}|b_i\oplus c_i\rangle|a_i\rangle\right)|z\oplus s_n\rangle$$

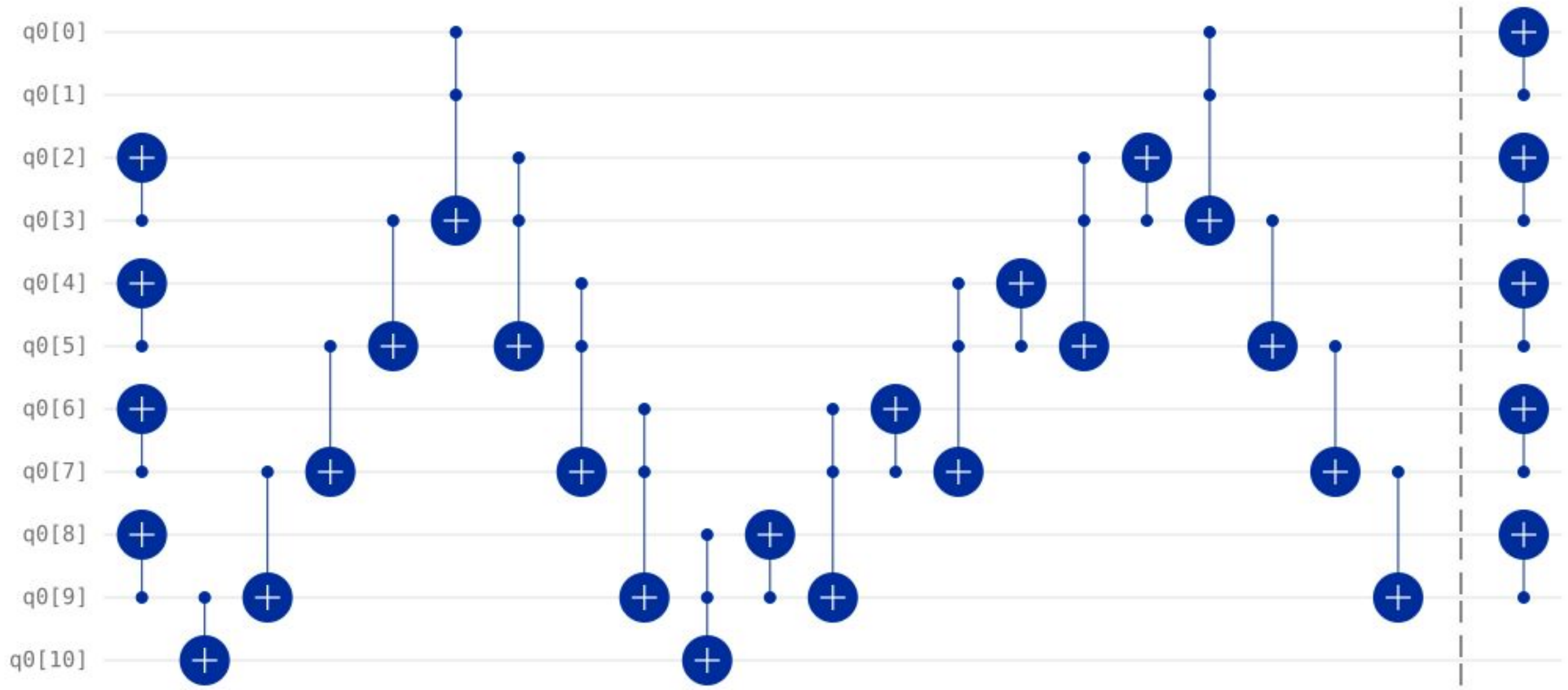
- The remaining step will finally proceed with the sum of the two components

SAPIENZA
UNIVERSITÀ DI ROMA

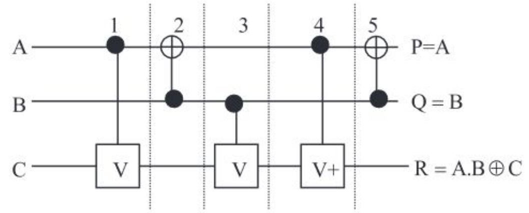
Step By Step Creation

$$\left(\bigotimes_{i=0}^{n-1} |b_i\rangle|a_i\rangle\right)|z\rangle \to \left(\bigotimes_{i=0}^{n-1} |s_i\rangle|a_i\rangle\right)|z \oplus s_n\rangle$$

- Step 6:
    - apply CNOT gates to B_i, A_i for i = 0,...,n-1 with B_i as target



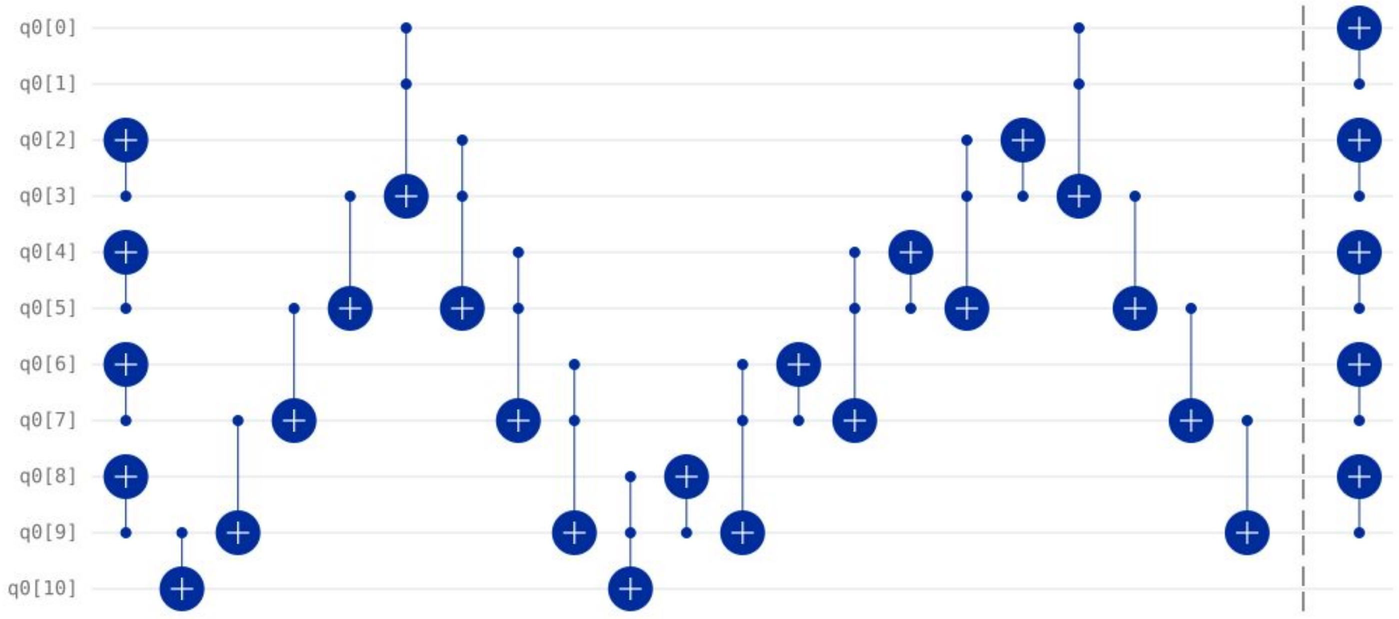
SAPIENZA
UNIVERSITÀ DI ROMA



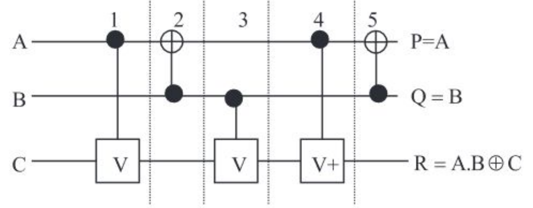
(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7



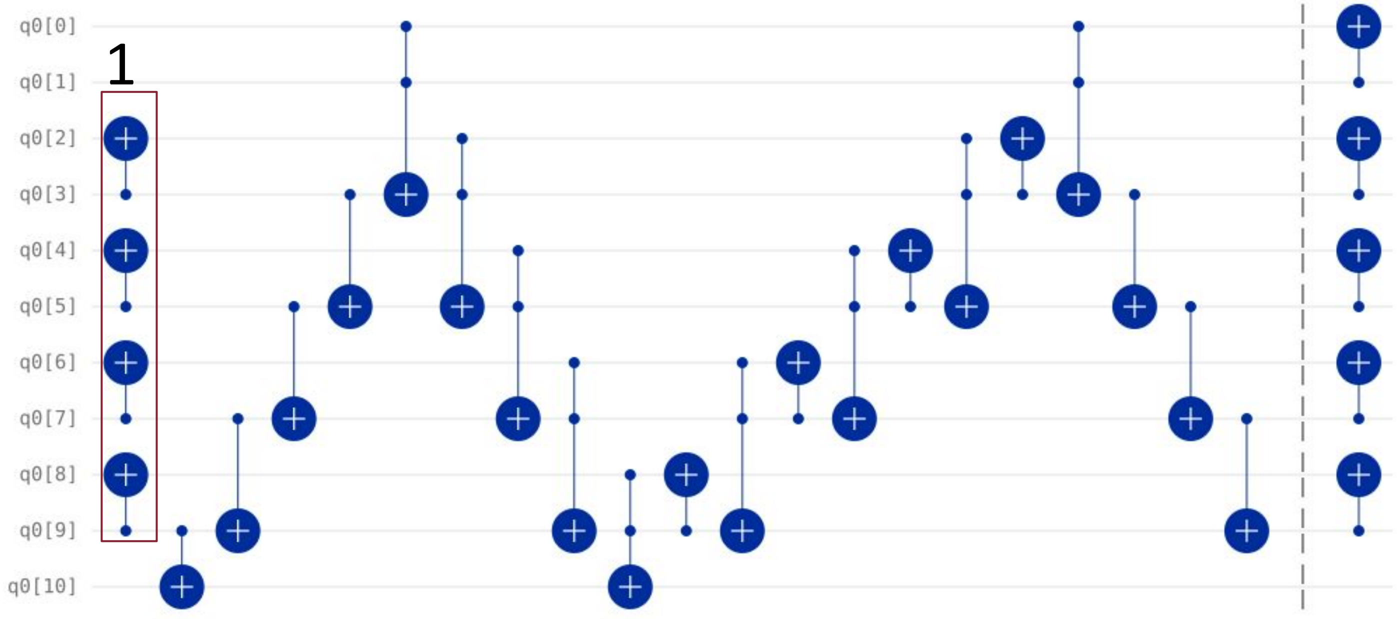
SAPIENZA
UNIVERSITÀ DI ROMA



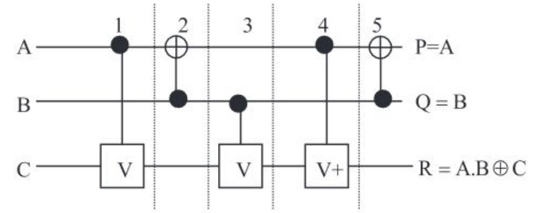
(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
  - In this example we have 1



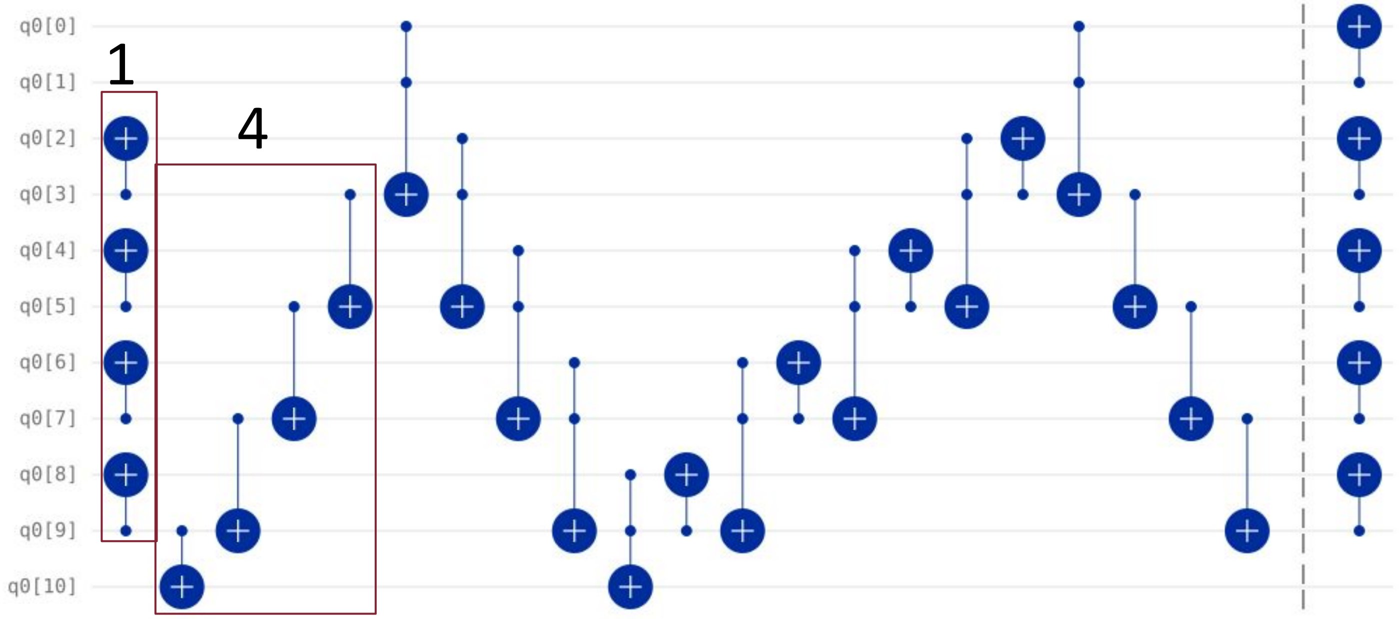
SAPIENZA
UNIVERSITÀ DI ROMA



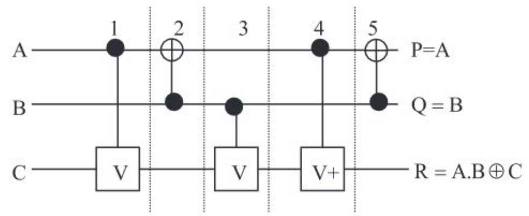
(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
  - In this example we have 1 + 4



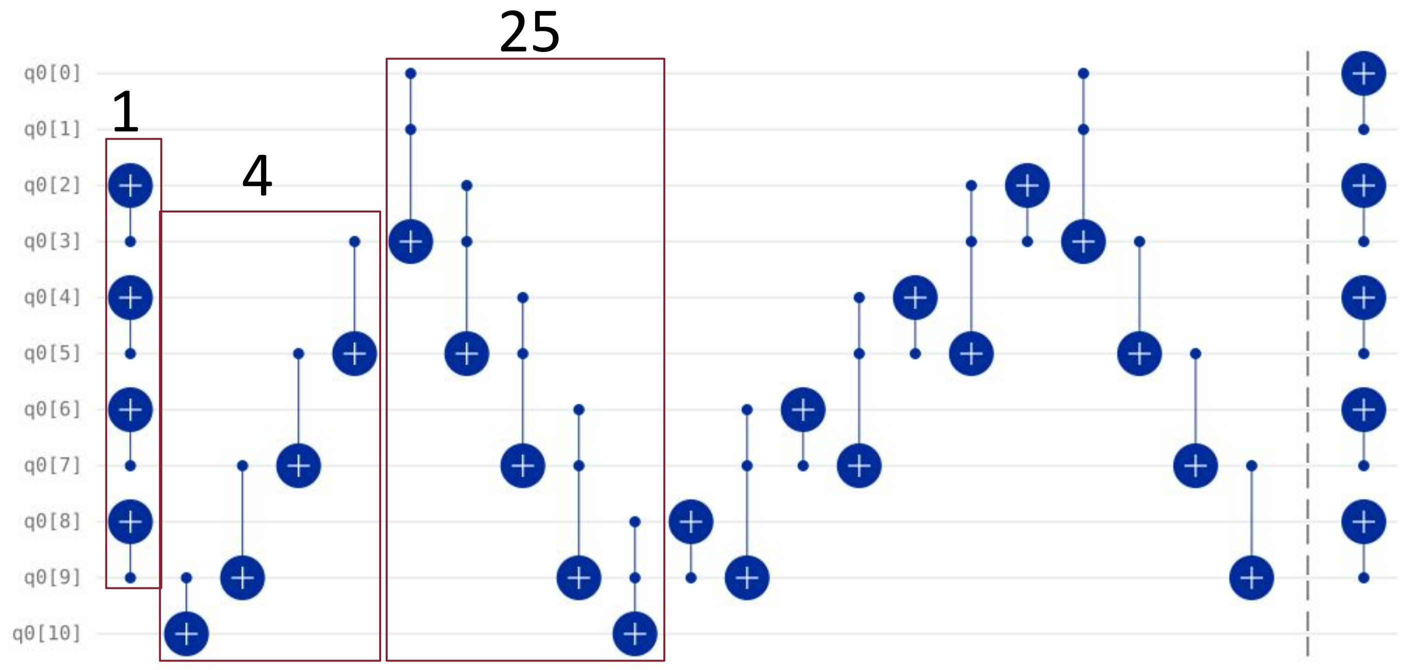
SAPIENZA
UNIVERSITÀ DI ROMA



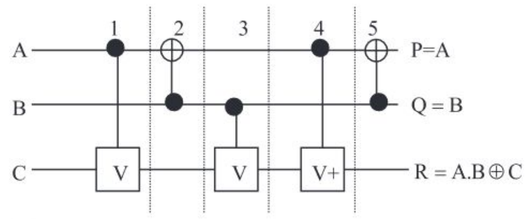
(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
    - In this example we have 1 + 4 + 25

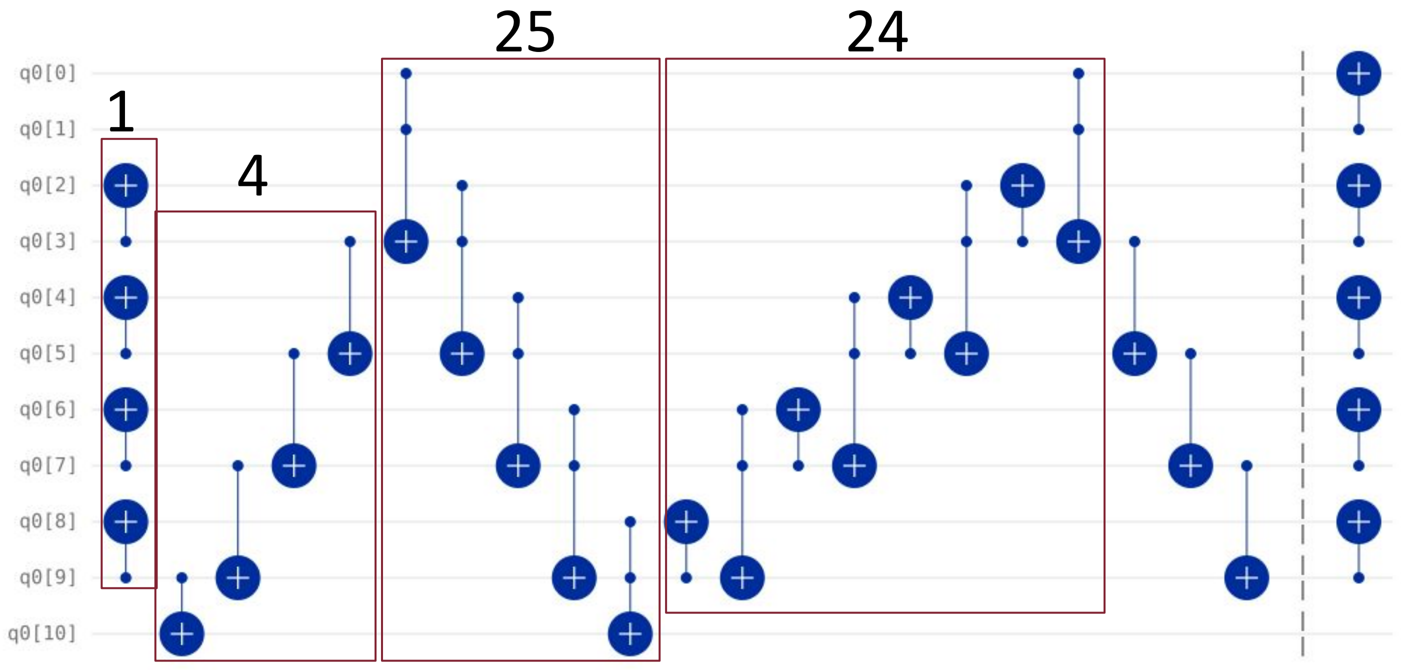


Cost Analysis - Depth and Delay

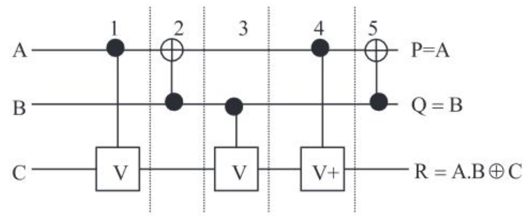


(b) quantum implementation of Toffoli gate

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
  - In this example we have 1 + 4 + 25 + 24



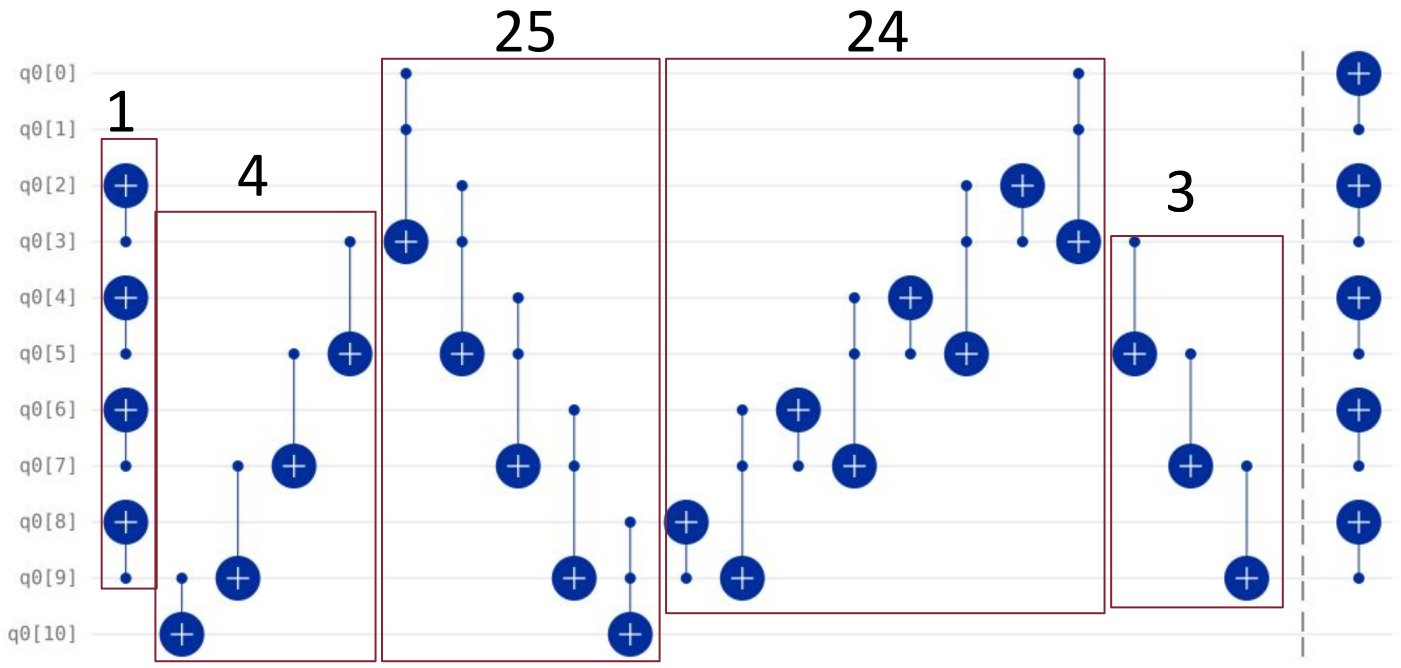
SAPIENZA
UNIVERSITÀ DI ROMA

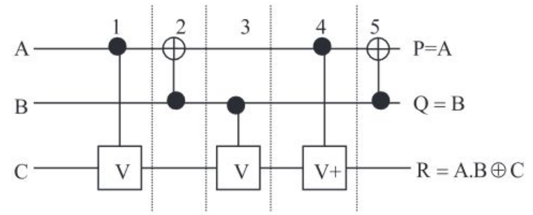


(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
  - In this example we have 1 + 4 + 25 + 24 + 3

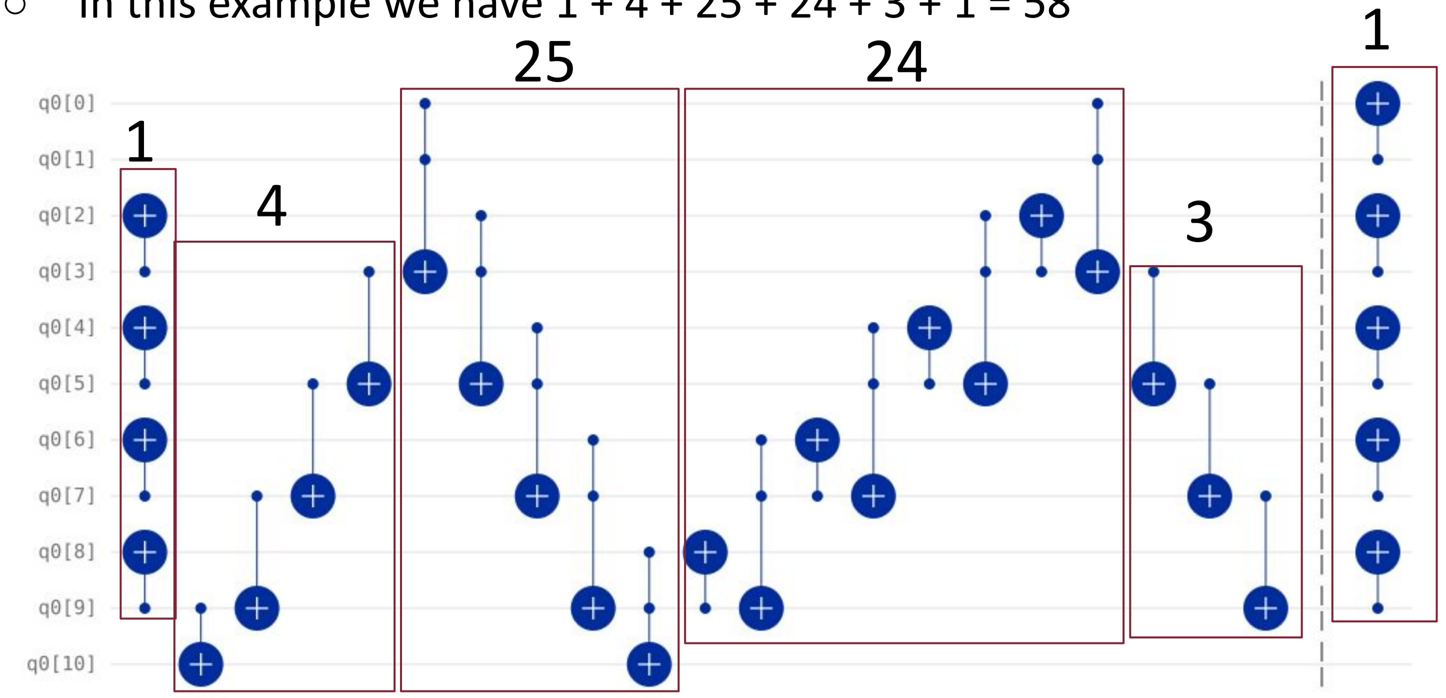


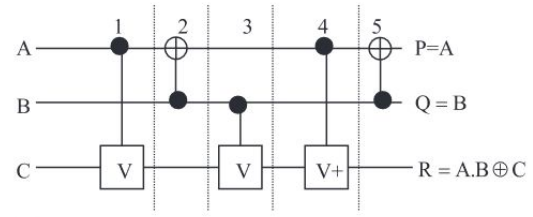


(b) quantum implementation of Toffoli gate

Cost Analysis - Depth and Delay

- The Depth of the entire circuit is 5N − 3
- The Delay is 13 N - 7 (for N = 5 we have 58)
  - In this example we have 1 + 4 + 25 + 24 + 3 + 1 = 58

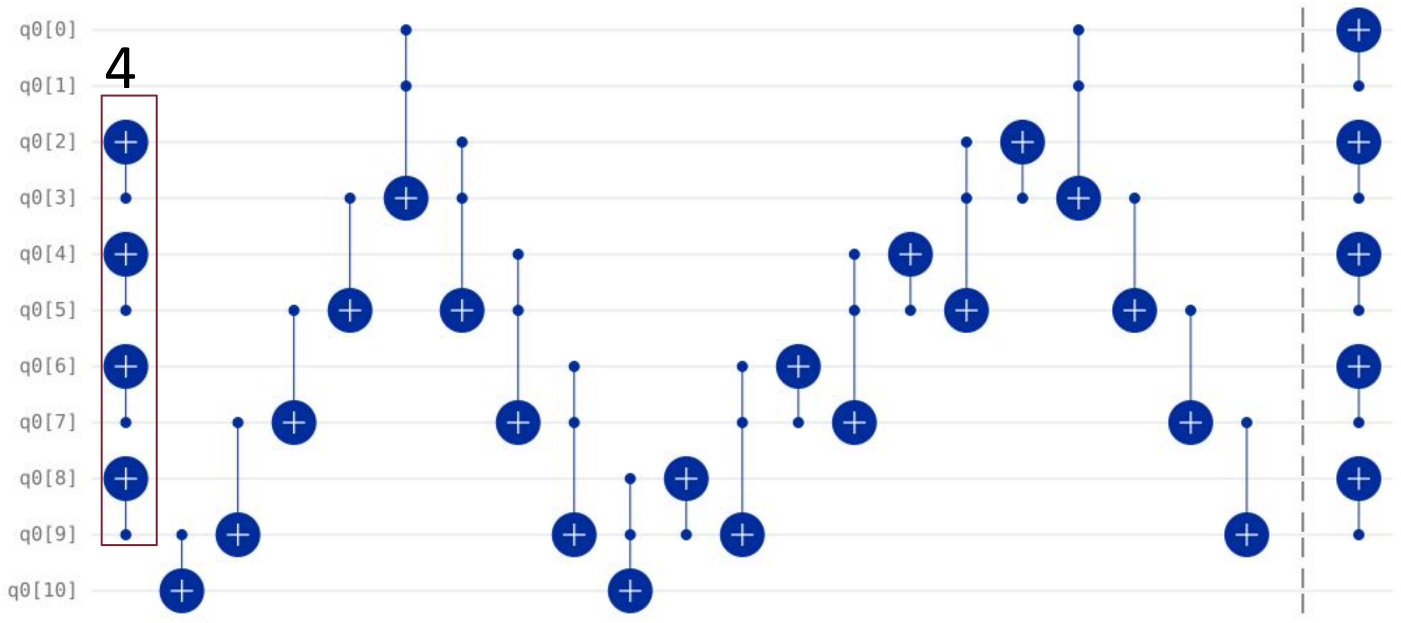




(b) quantum implementation of Toffoli gate

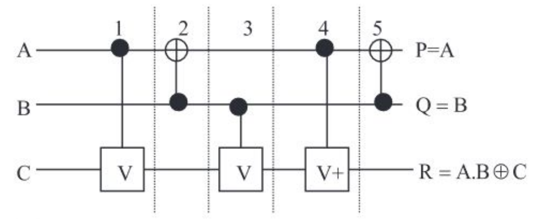
Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N − 6
- The Quantum Cost is 15N - 9
    - But here we have 4

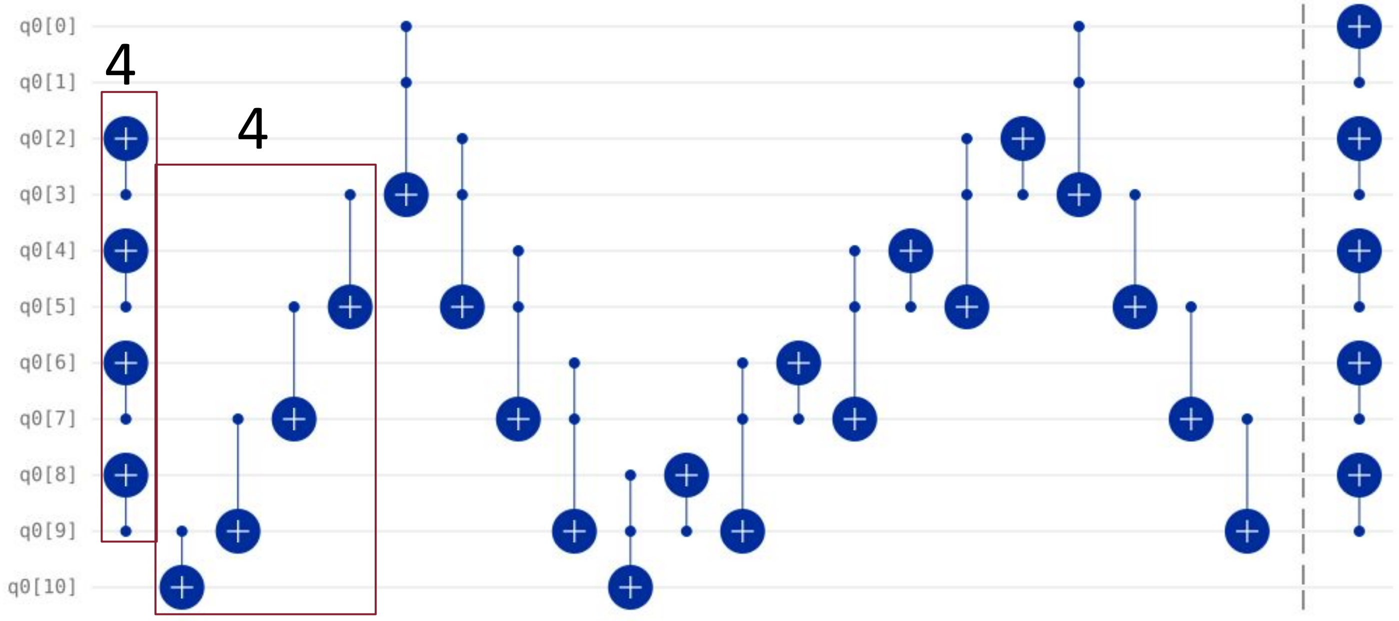


Cost Analysis - Size and Quantum Cost

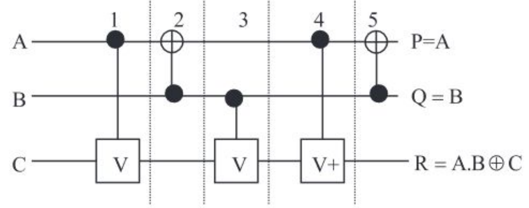
- The Size of the circuit is 7N − 6
- The Quantum Cost is 15N - 9
    - But here we have 4 + 4



(b) quantum implementation of Toffoli gate



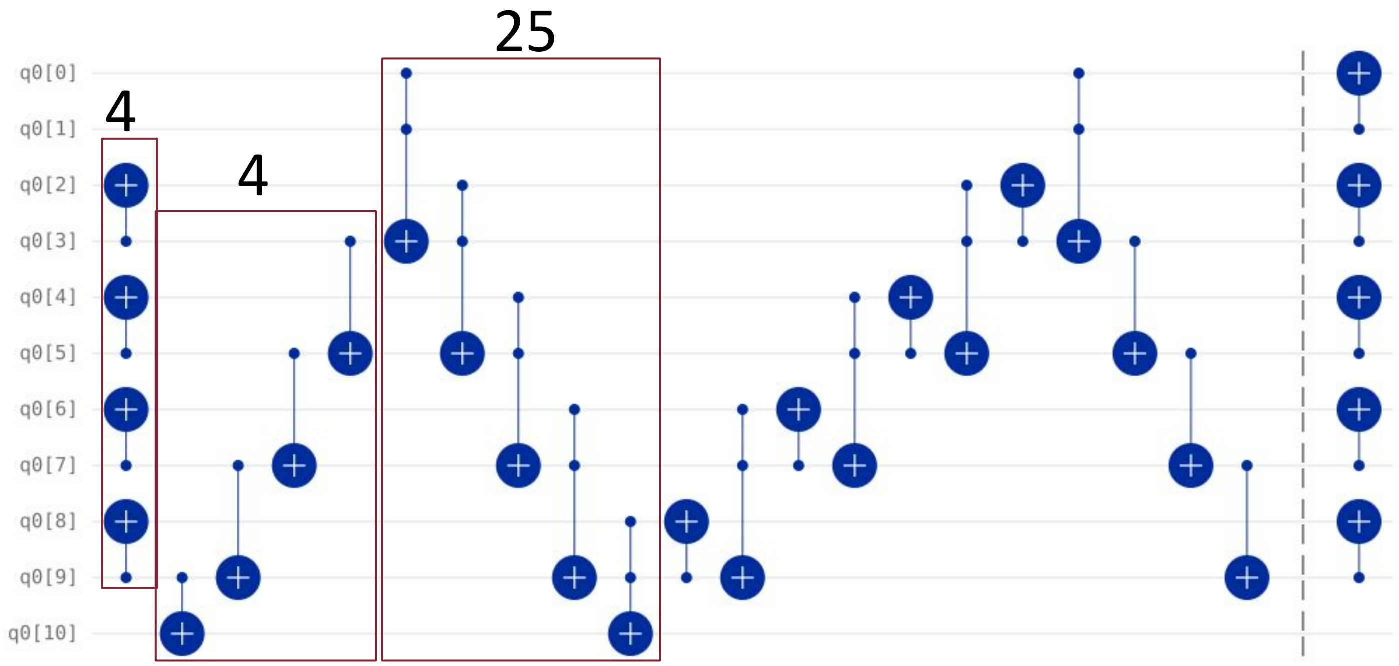
SAPIENZA
UNIVERSITÀ DI ROMA

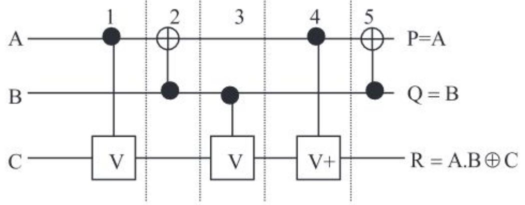


(b) quantum implementation of Toffoli gate

Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N – 6
- The Quantum Cost is 15N - 9
  - But here we have 4 + 4 + 25

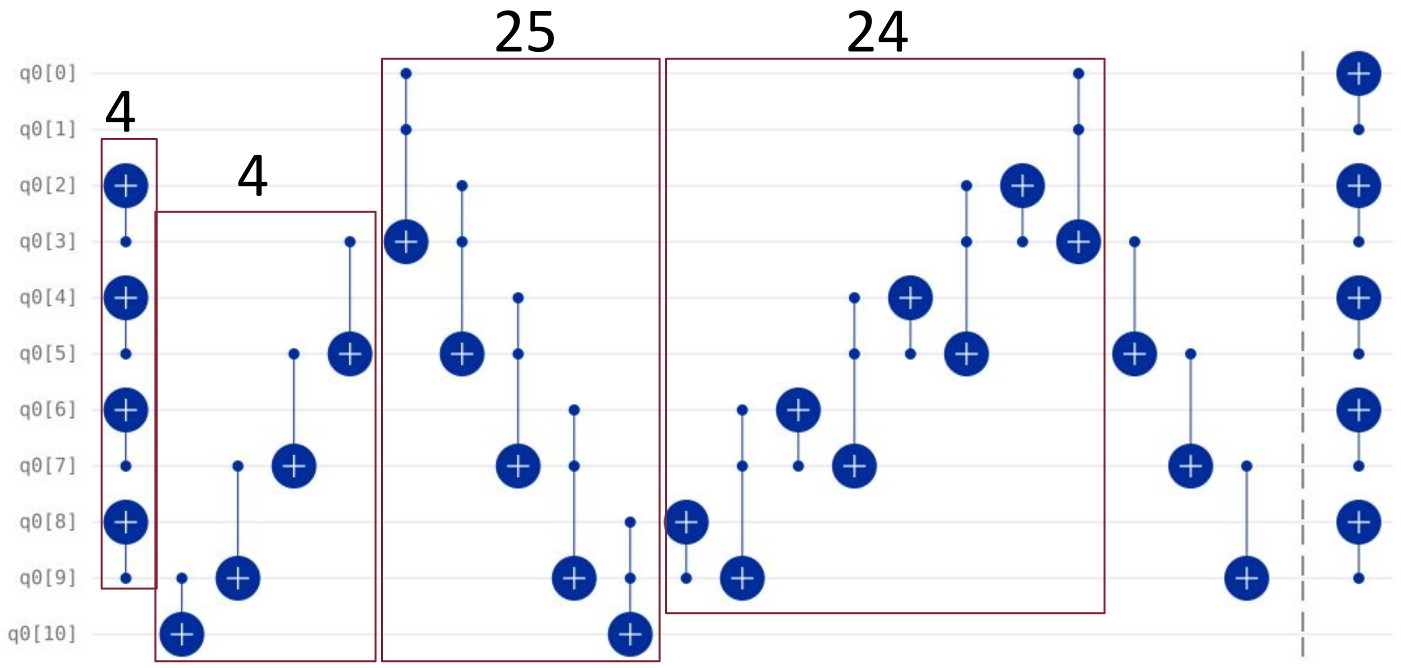




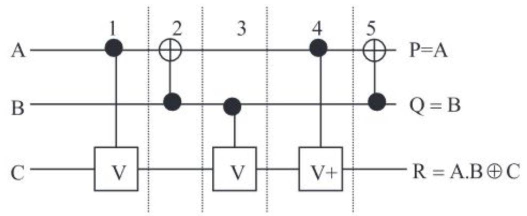
(b) quantum implementation of Toffoli gate

Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N – 6
- The Quantum Cost is 15N - 9
  - But here we have 4 + 4 + 25 + 24



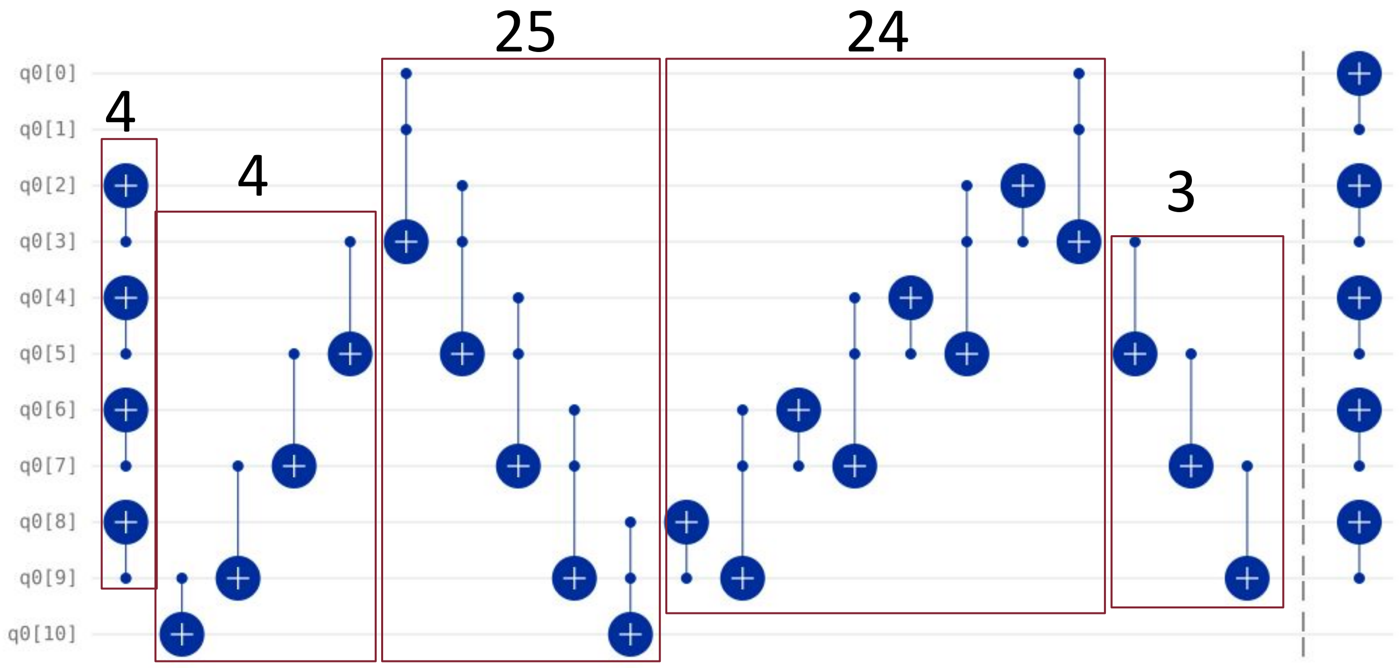
SAPIENZA
UNIVERSITÀ DI ROMA

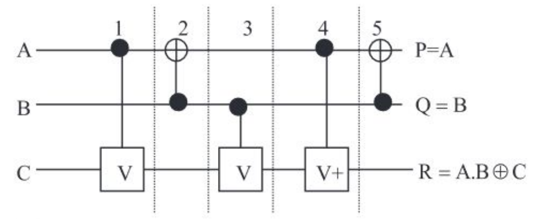


(b) quantum implementation of Toffoli gate

Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N − 6
- The Quantum Cost is 15N - 9
    - But here we have 4 + 4 + 25 + 24 + 3

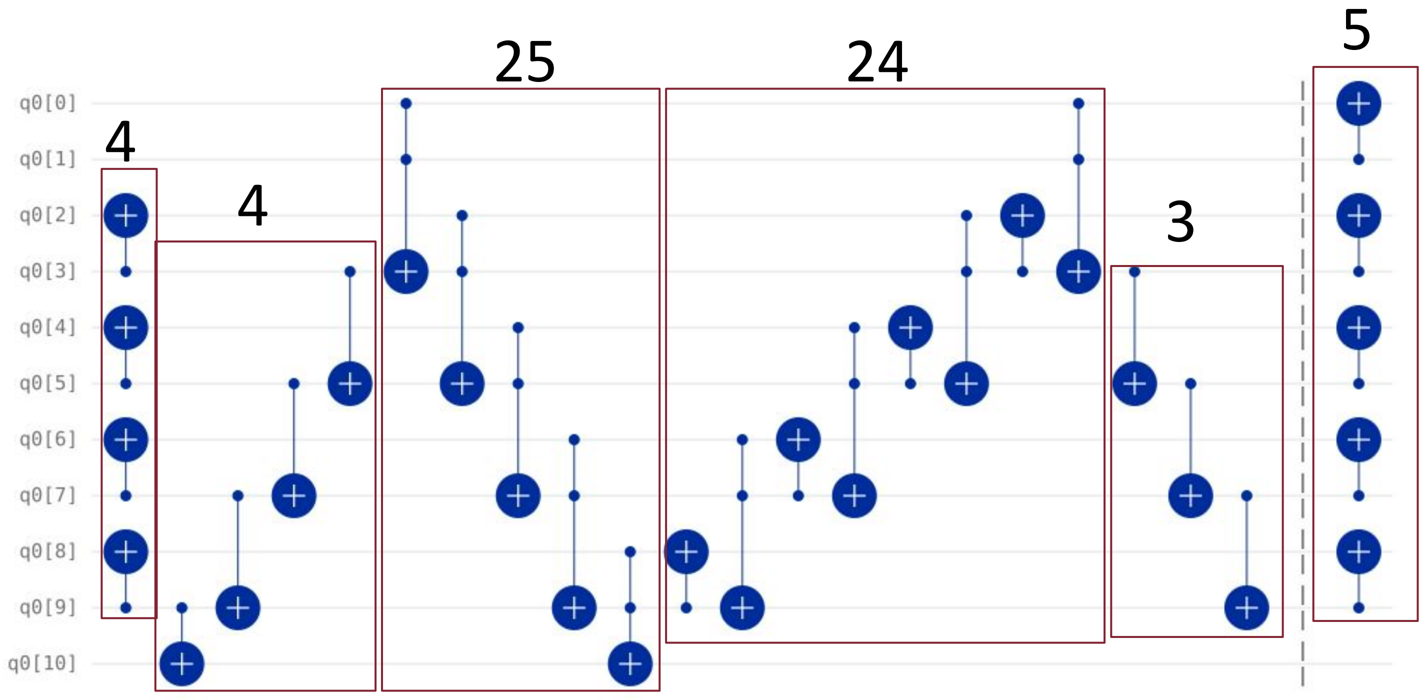




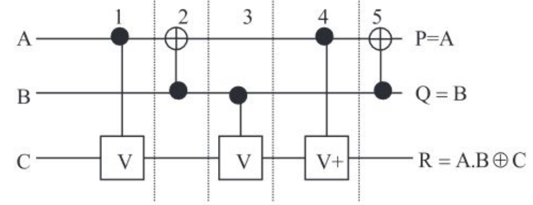
(b) quantum implementation of Toffoli gate

Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N − 6
- The Quantum Cost is 15N - 9
  - But here we have 4 + 4 + 25 + 24 + 3 + 5 = 65



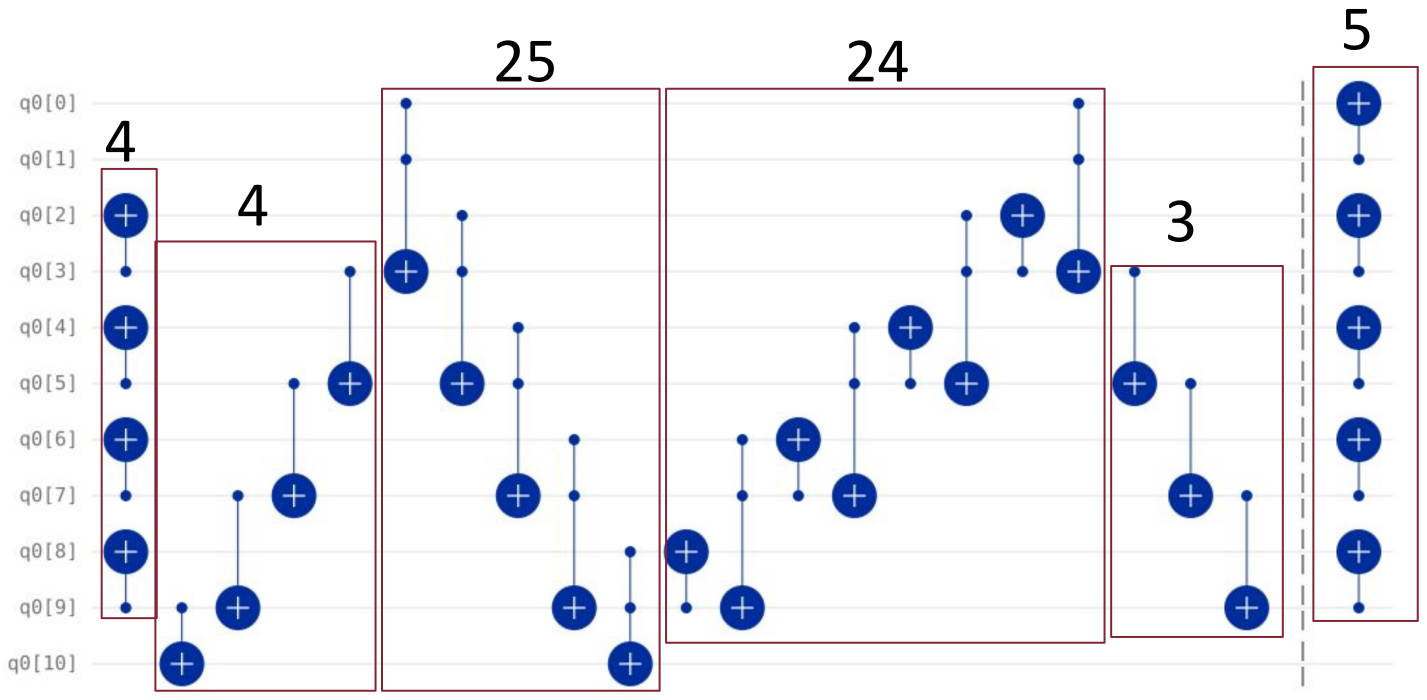
SAPIENZA
UNIVERSITÀ DI ROMA



(b) quantum implementation of Toffoli gate

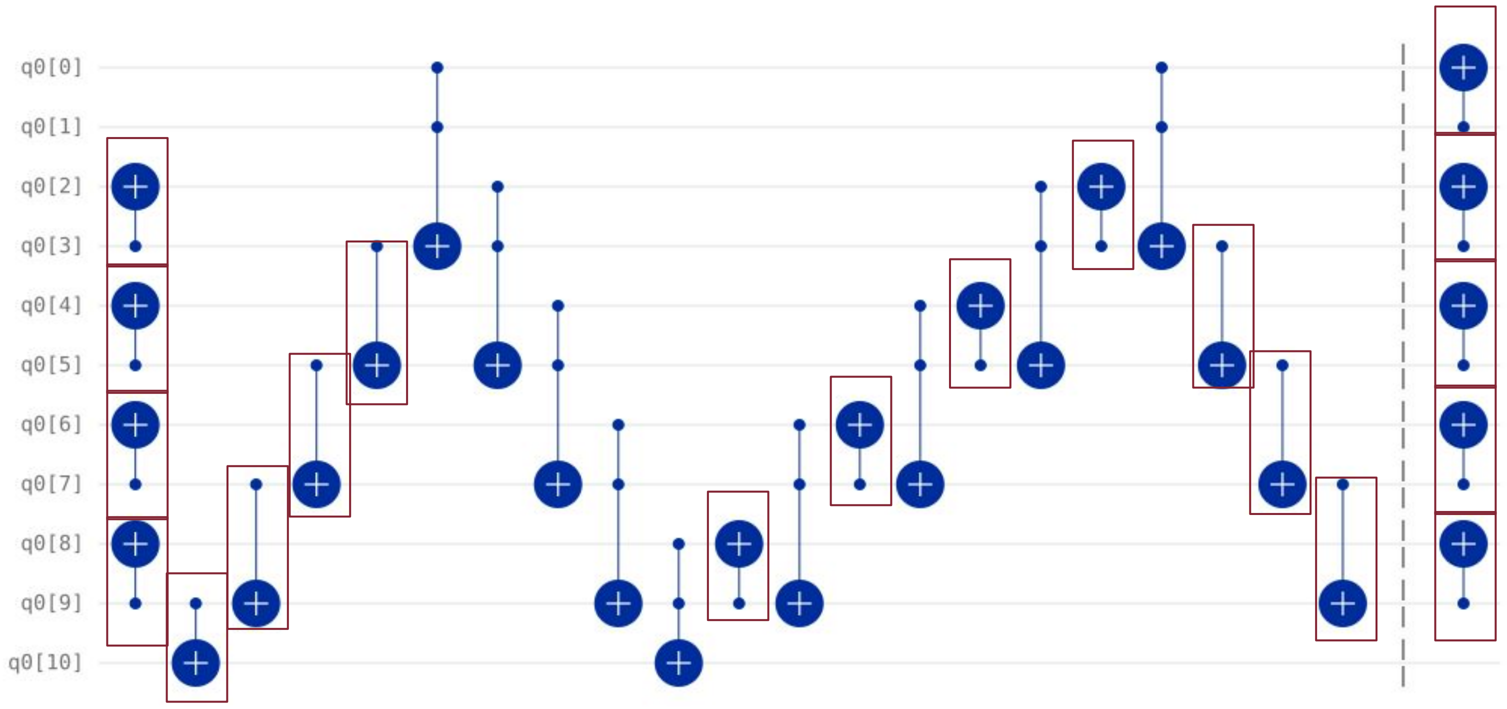
Cost Analysis - Size and Quantum Cost

- The Size of the circuit is 7N − 6
- The Quantum Cost is 15N - 9
  - But here we have 4 + 4 + 25 + 24 + 3 + 5 = 65 (and 15 N - 9 is 66)



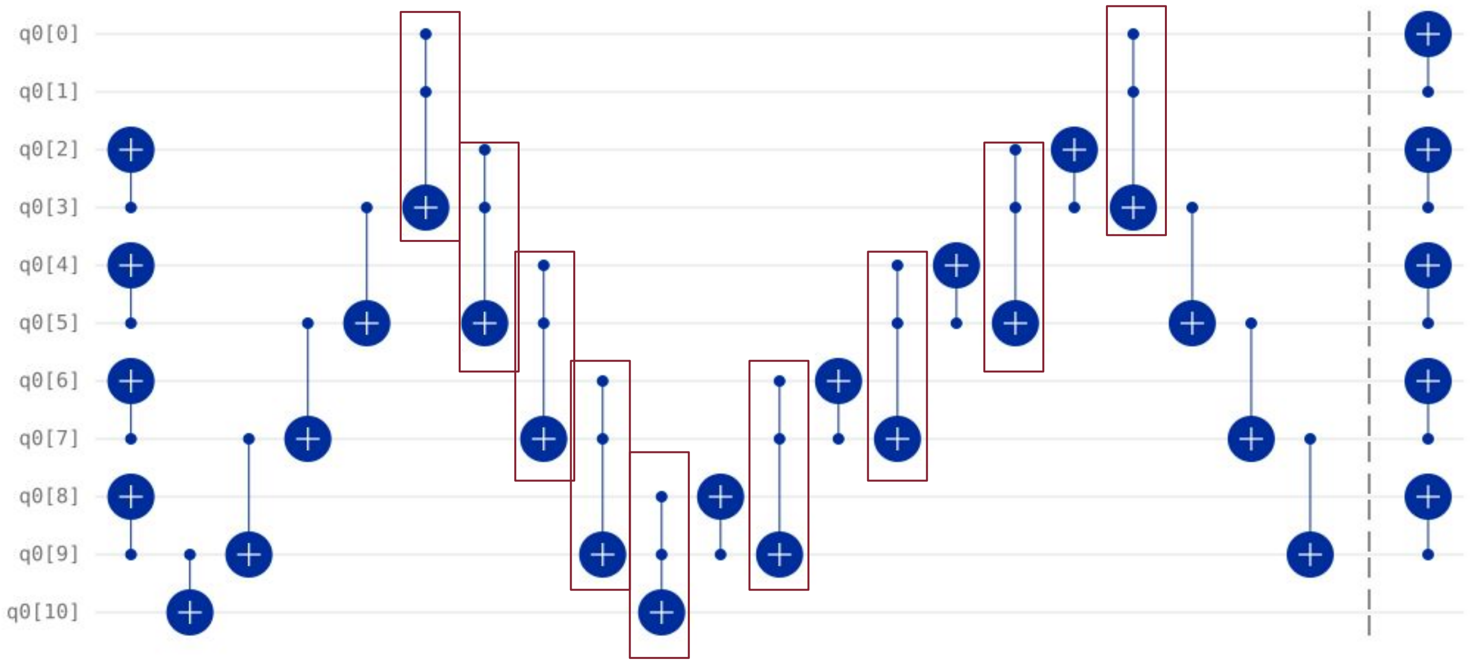
Cost Analysis - CNOT

- The number of Feynman gates is 5N − 5
- Here we have 20 CNOT gates



Cost Analysis - CCNOT

- The number of Toffoli gates is 2N - 1
- For N = 5, we have 9 CCNOT gates



SAPIENZA
UNIVERSITÀ DI ROMA

Thapliyal

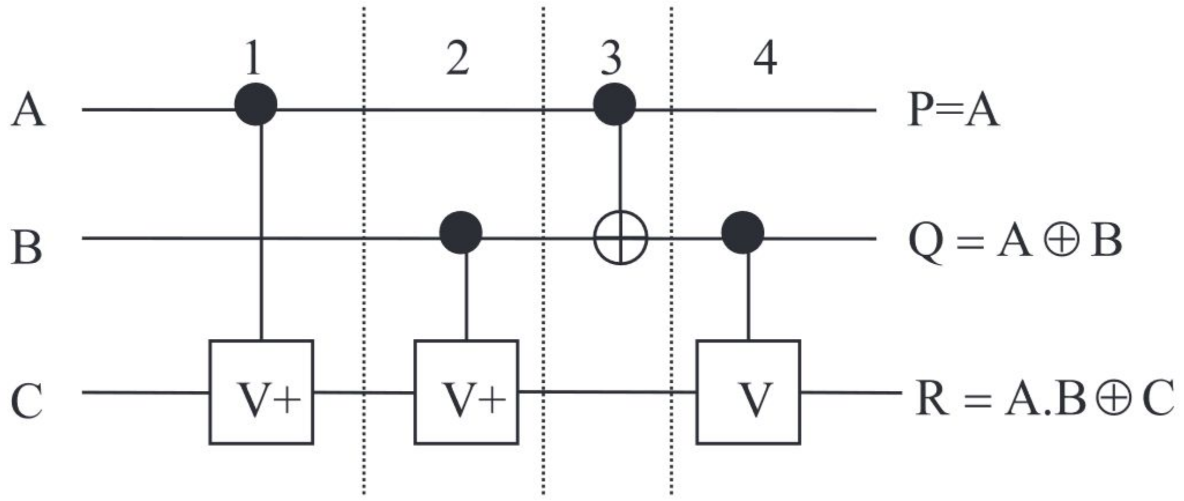
2013

Introduction

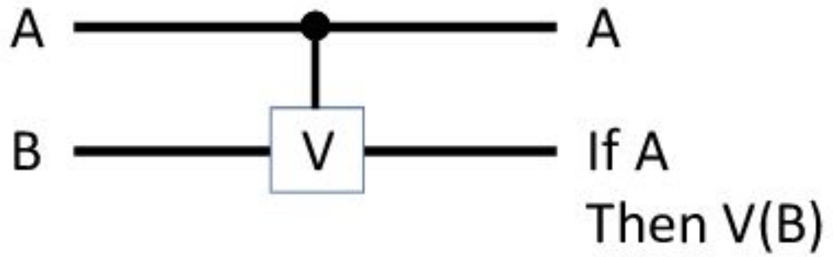
- The Thapliyal circuit we present is a reversible ripple carry adder
- The design has
  - no input carry
  - no ancilla bits
  - no garbage outputs
- The number of qubits is 2N +1
- The carry is computed in the first stage similarly to the Takahashi adder
- A key component is the Peres Gate

Peres Gate

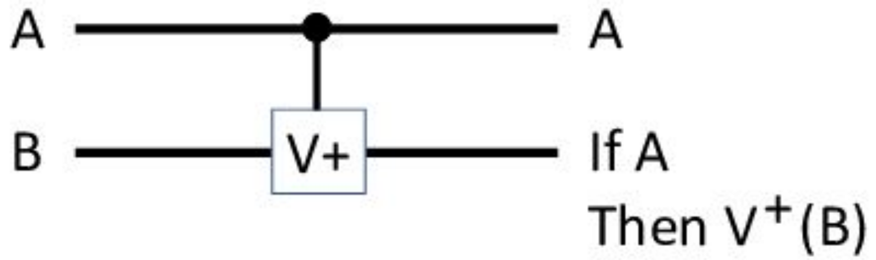
- The Peres gate has quantum cost 4 and delay 4Δ



Controlled V and V⁺ Gates



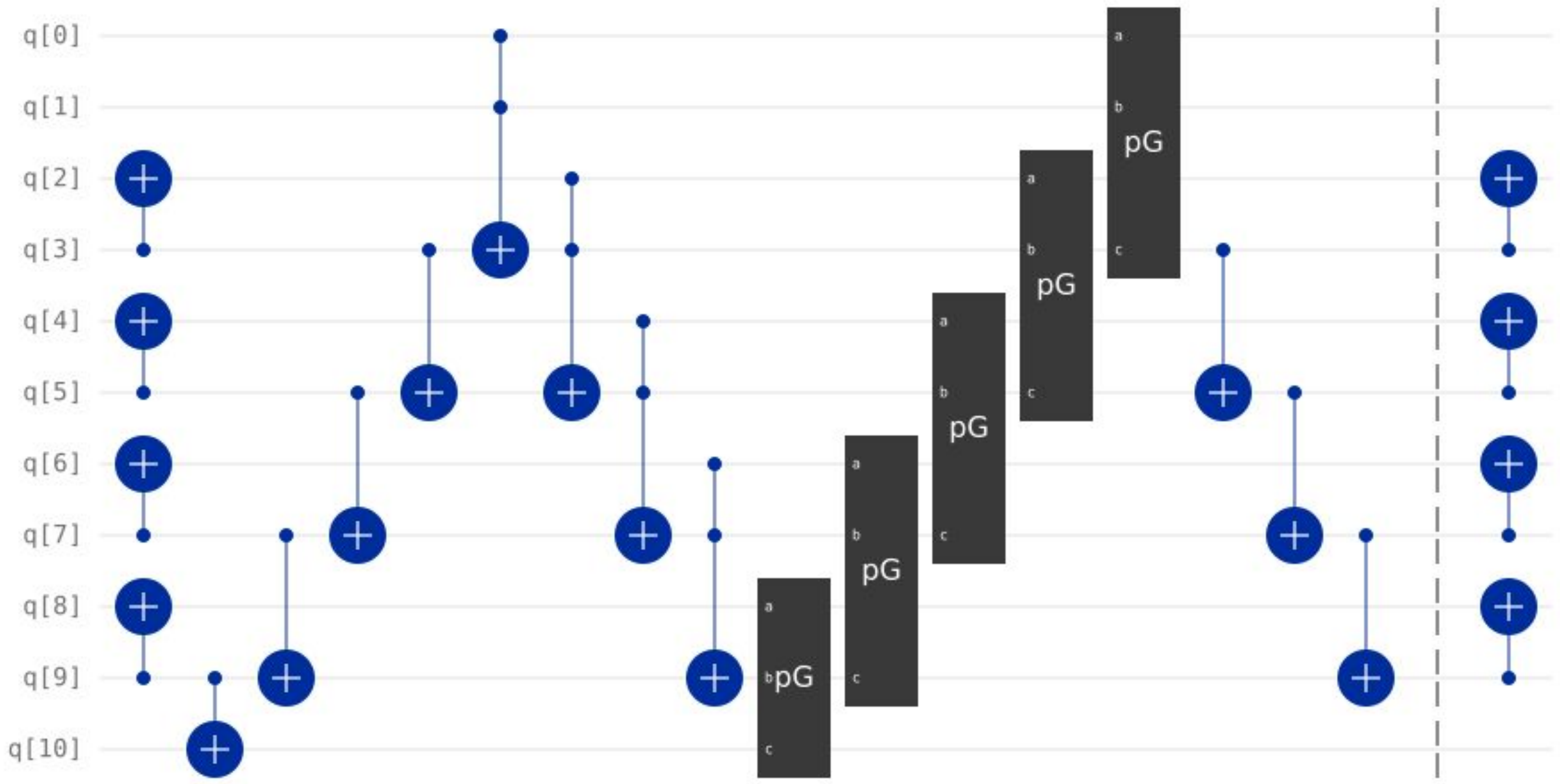
$$V = \frac{1+i}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -i \\ 0 & 0 & -i & 1 \end{pmatrix}$$



$$V^{+} = \frac{1-i}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & i \\ 0 & 0 & i & 1 \end{pmatrix}$$

SAPIENZA
UNIVERSITÀ DI ROMA

Thapliyal - Circuit (N = 5)



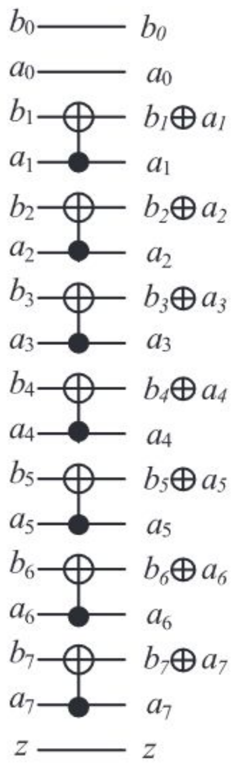
SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 1:
  - apply CNOT gates to each B_i, A_i for i=1,..,n-1 with A_{i+1} as control

- The system state becomes:

$$|b_0\rangle\,|a_0\rangle\left(\bigotimes_{i=1}^{n-1}|b_i\oplus a_i\rangle\,|a_i\rangle\right)|z\rangle$$



(a) after step 1

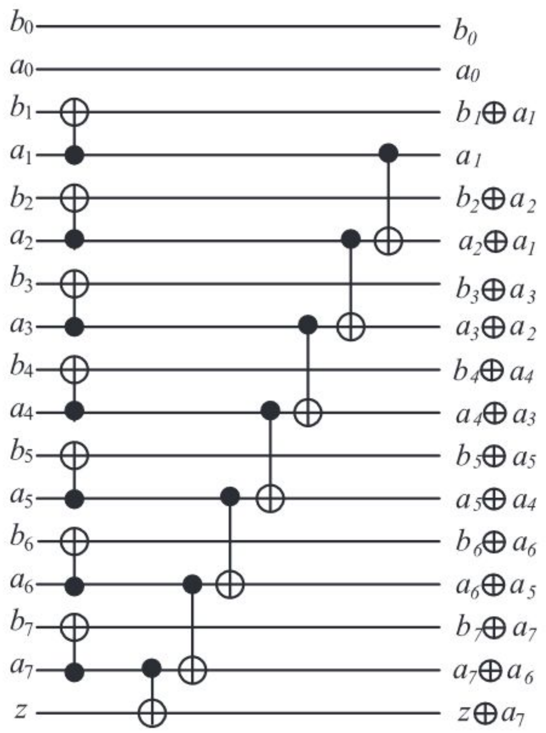
SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 2:
  - apply CNOT gates to each A_i, A_{i+1} for i=n-1,...,1 with A_i as control
- The system state becomes:

$$|b_0\rangle\,|a_0\rangle\,|b_1 \oplus a_1\rangle\,|a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus a_i\rangle\,|a_i \oplus a_{i-1}\rangle \right) |z \oplus a_{n-1}\rangle$$

- Step 1 and 2 are the same as the previous adder



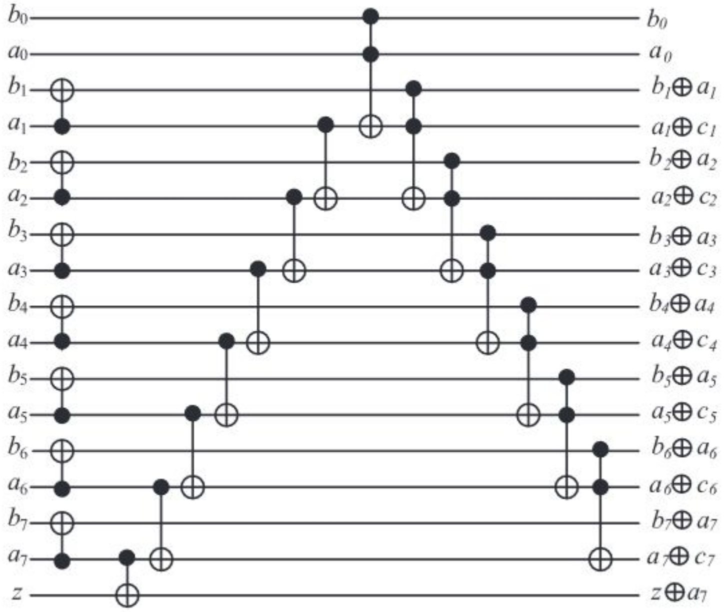
(b) after step 2

Step By Step Creation

- Step 3:
  - apply CCNOT gates to each A_i,B_i, A_{i+1} for i=0,...,n-2 with A_{i+1} as target
- The system state becomes:

$$|b_0\rangle\,|a_0\rangle\left(\bigotimes_{i=1}^{n-1}|b_i\oplus a_i\rangle\,|a_i\oplus c_i\rangle\right)|z\oplus a_{n-1}\rangle$$

- By applying Toffoli gates we obtain the carry bits

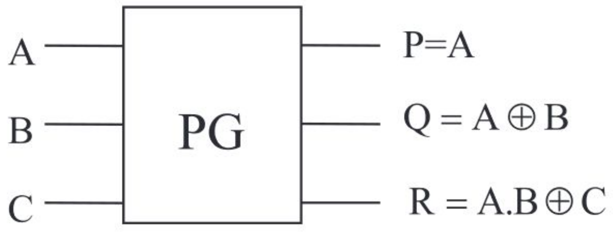


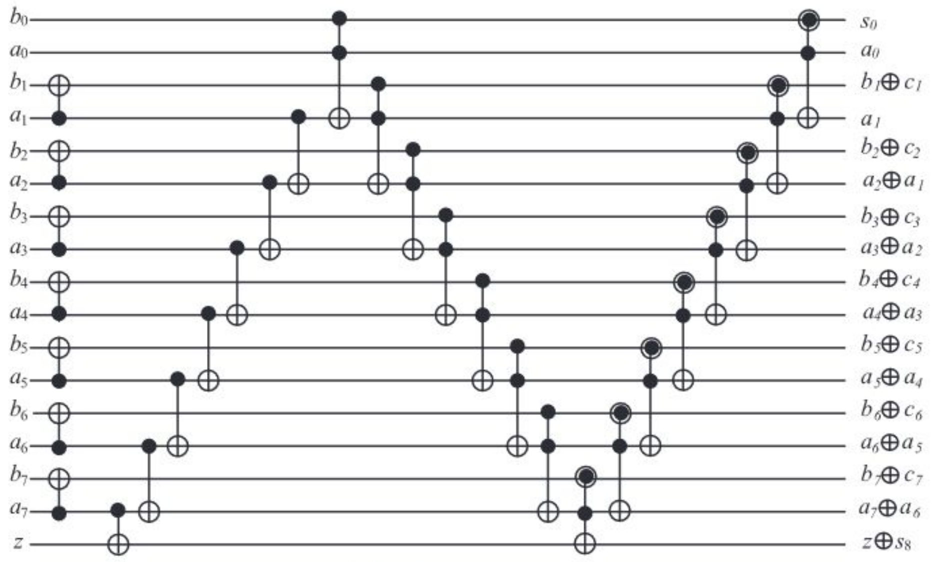
(c) after step 3

SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 4:
  - apply Peres gates to each A_i,B_i, A_{i+1} on gate inputs A,B and C for i=n-1,...,0
- The system state becomes: $|s_0\rangle\,|a_0\rangle\,|b_1 \oplus c_1\rangle\,|a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus c_i\rangle\,|a_i \oplus a_{i-1}\rangle\right) |z \oplus s_n\rangle$



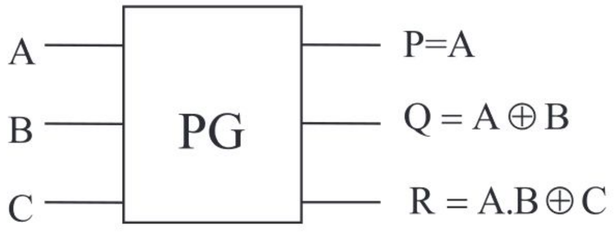


(a) after step 4

SAPIENZA
UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 4:
  - apply Peres gates to each A_i,B_i, A_{i+1} on gate inputs A,B and C for i=n-1,...,0
- The system state becomes: $$|s_0\rangle\,|a_0\rangle\,|b_1 \oplus c_1\rangle\,|a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus c_i\rangle\,|a_i \oplus a_{i-1}\rangle\right) |z \oplus s_n\rangle$$
- For example with respect to the Peres Gate for A_0, B_0, A_1 we have:
  - A_0: a_0 => a_0
  - B_0: b_0 => $b_0 \oplus a_0$
  - A_1: $a_1 \oplus c_1$ => $a_1 \oplus c_1 \oplus a_0b_0$ => a_1



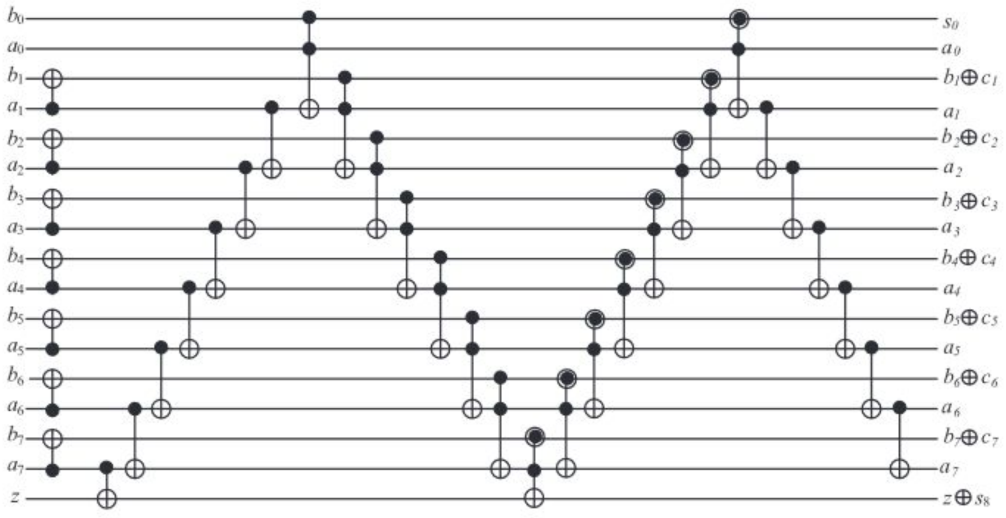
SAPIENZA UNIVERSITÀ DI ROMA

Step By Step Creation

- Step 5:
  - apply CNOT gates to each A_i, A_{i+1} for i=1,n-2 with A_{i+1} as target
- The system state becomes:

$$|s_0\rangle\,|a_0\rangle\left(\bigotimes_{i=1}^{n-1}|b_i\oplus c_i\rangle\,|a_i\rangle\right)|z\oplus s_n\rangle$$

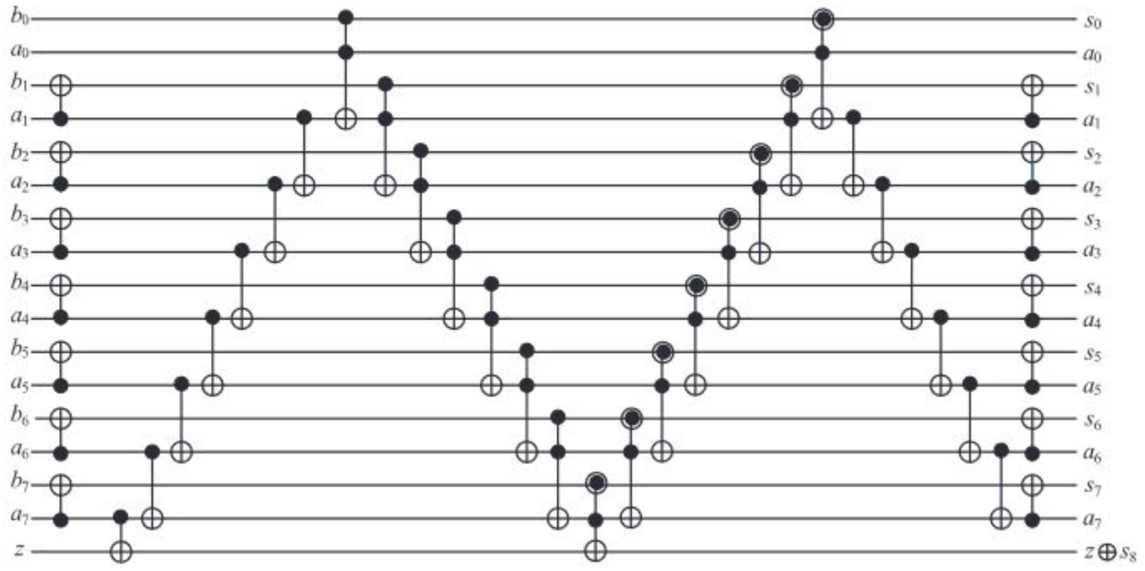
- We reverse the transformation on A_{i+1} qubit
- From $|a_i \oplus a_{i-1}\rangle$ to $|a_i\rangle$



(b) after step 5

Step By Step Creation

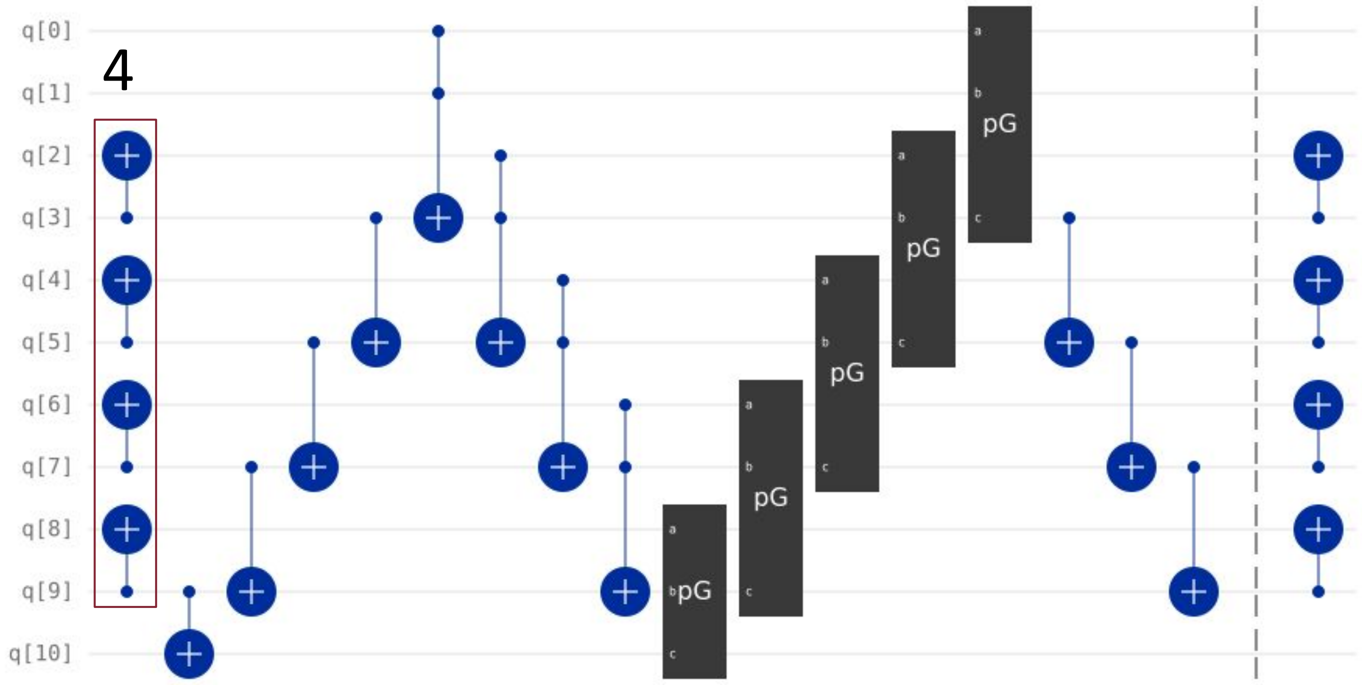
- Step 6:
  - apply CNOT gates to each B_i, A_i for i=1,...,n-1 with A_i as control
- The circuit is finished and computes the sum $\left(\bigotimes_{i=0}^{n-1} |s_i\rangle\,|a_i\rangle\right) |z \oplus s_n\rangle$



(c) after step 6

Cost Analysis

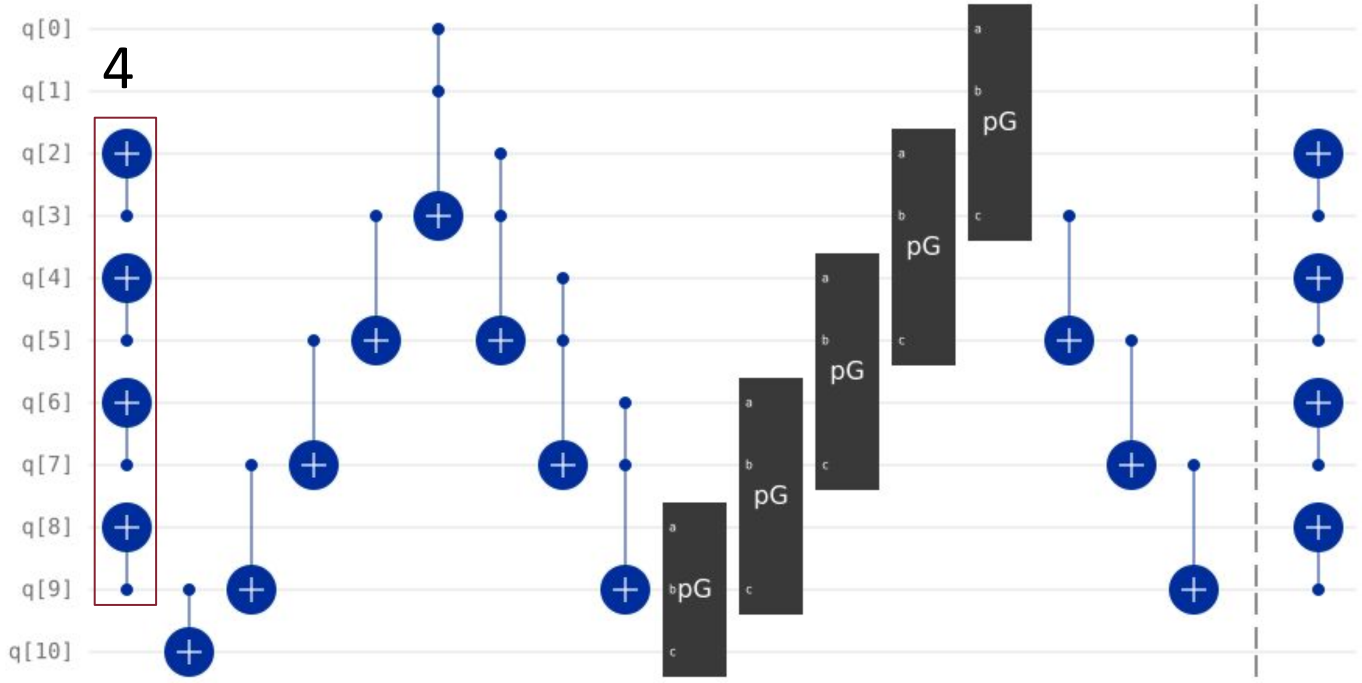
- Quantum Cost is 13N − 8 (with N = 5, the value is 57)



SAPIENZA
UNIVERSITÀ DI ROMA

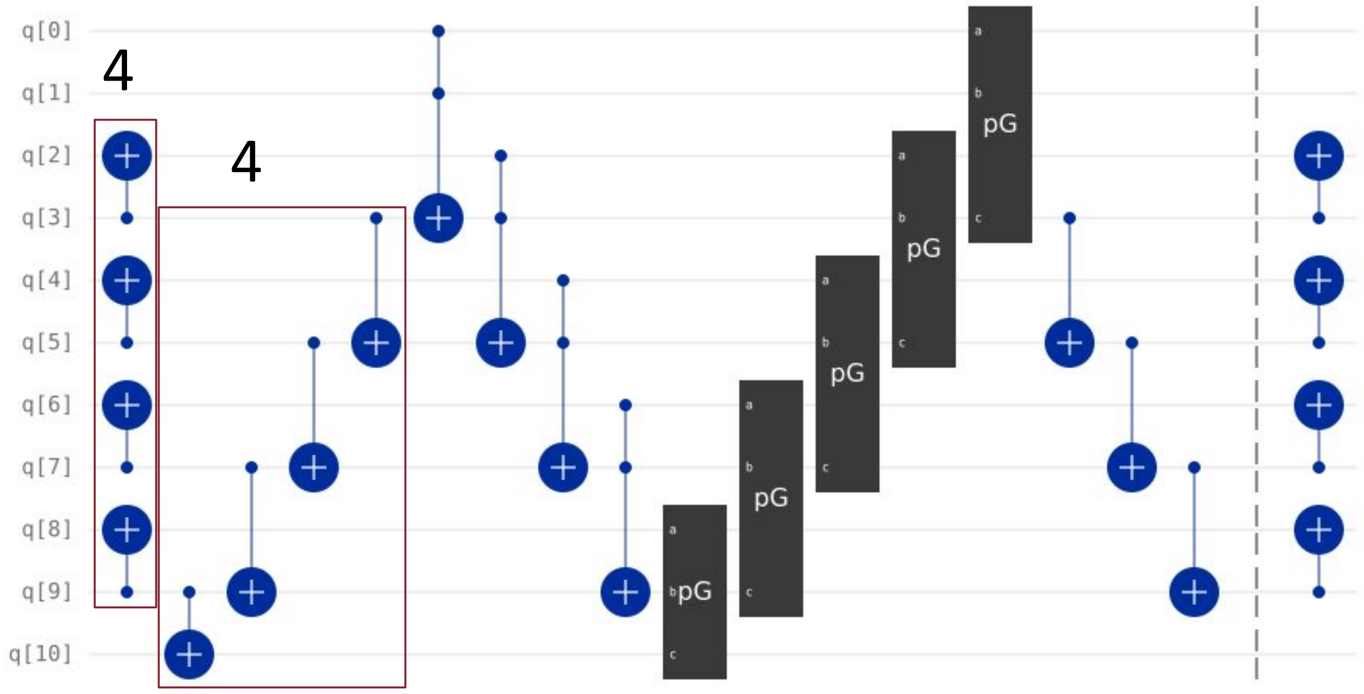
Cost Analysis

- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
    - But in the example: 4



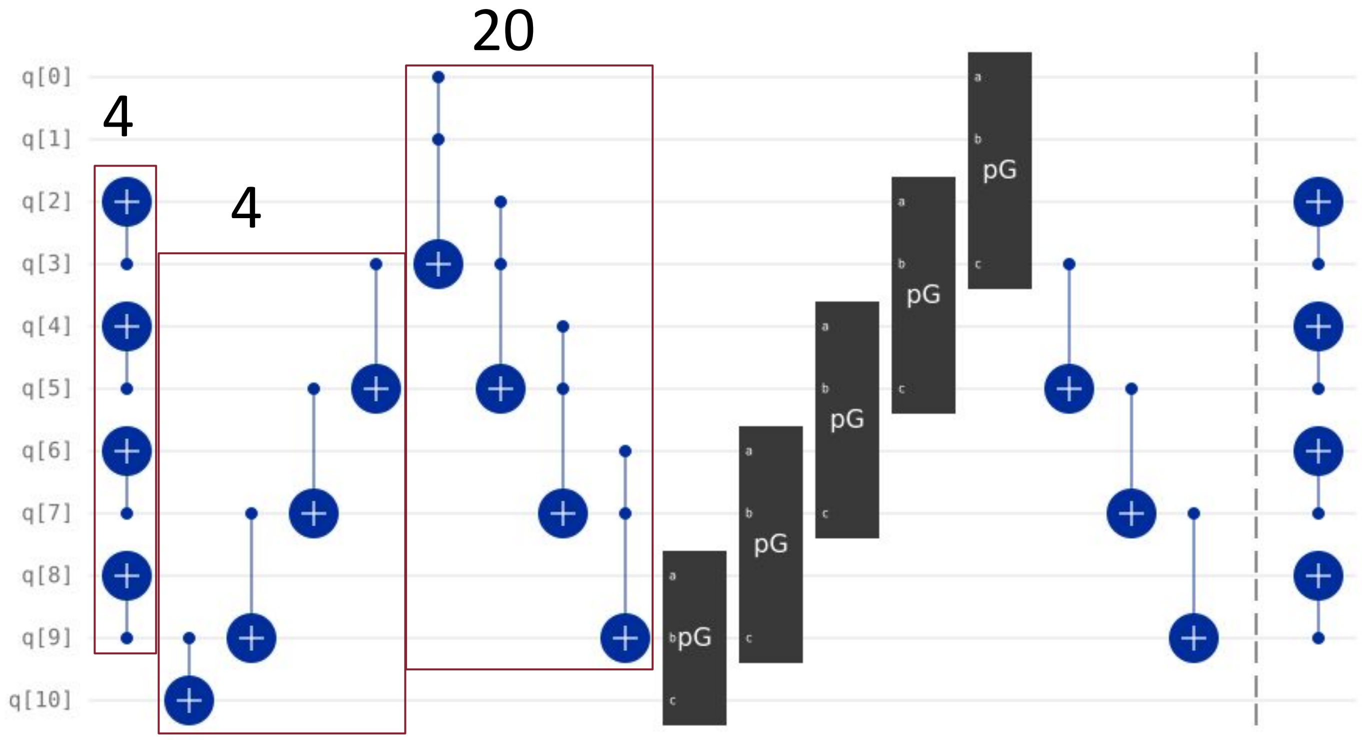
Cost Analysis

- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
    - But in the example: 4 + 4



Cost Analysis

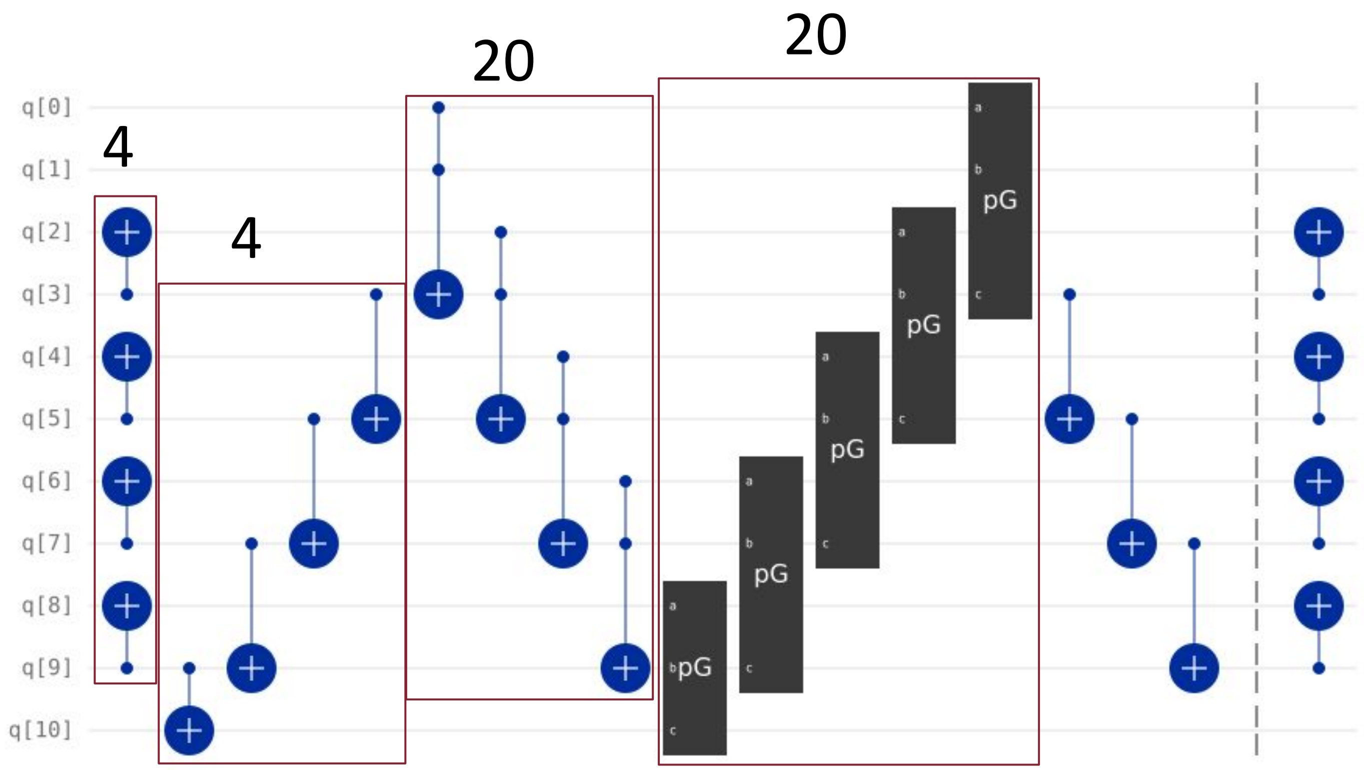
- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
  - But in the example 4 + 4 + 20



SAPIENZA
UNIVERSITÀ DI ROMA

Cost Analysis

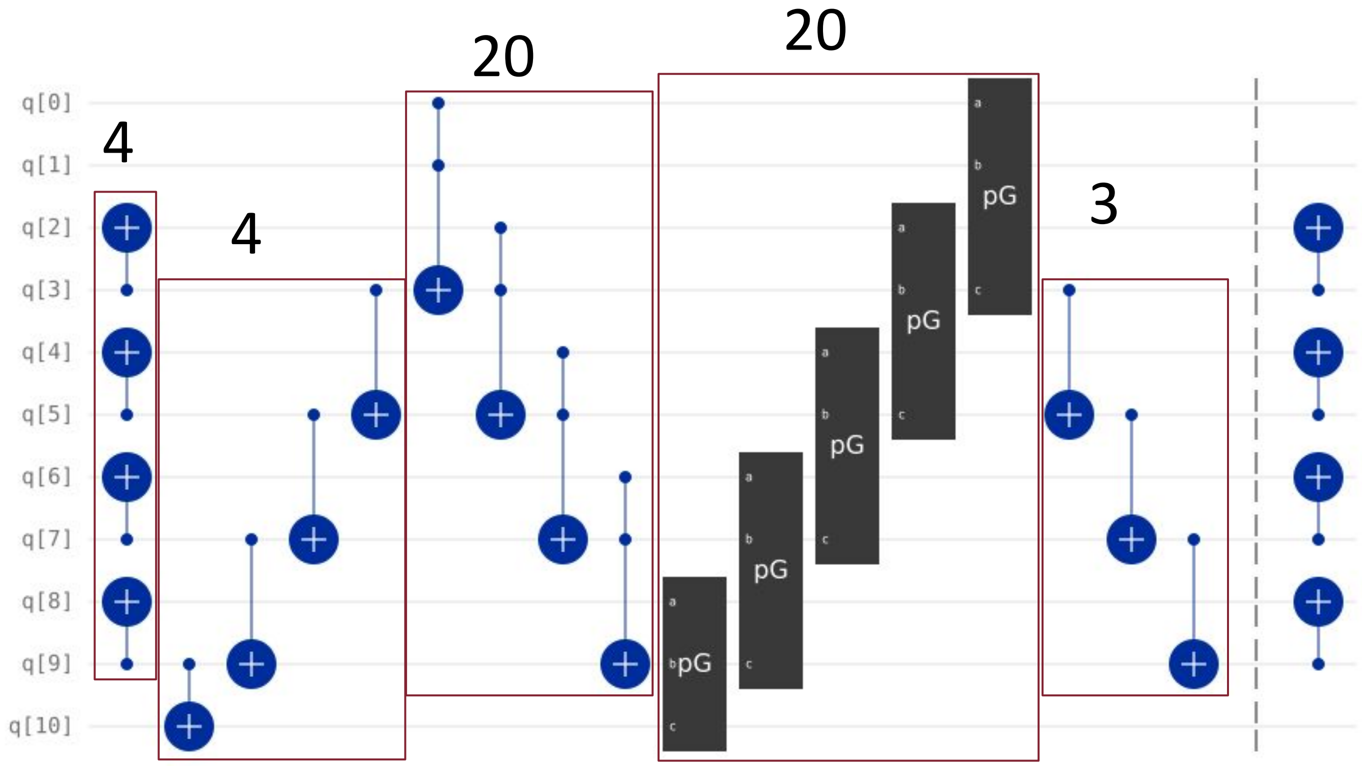
- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
    - But in the example 4 + 4 + 20 + 20



SAPIENZA
UNIVERSITÀ DI ROMA

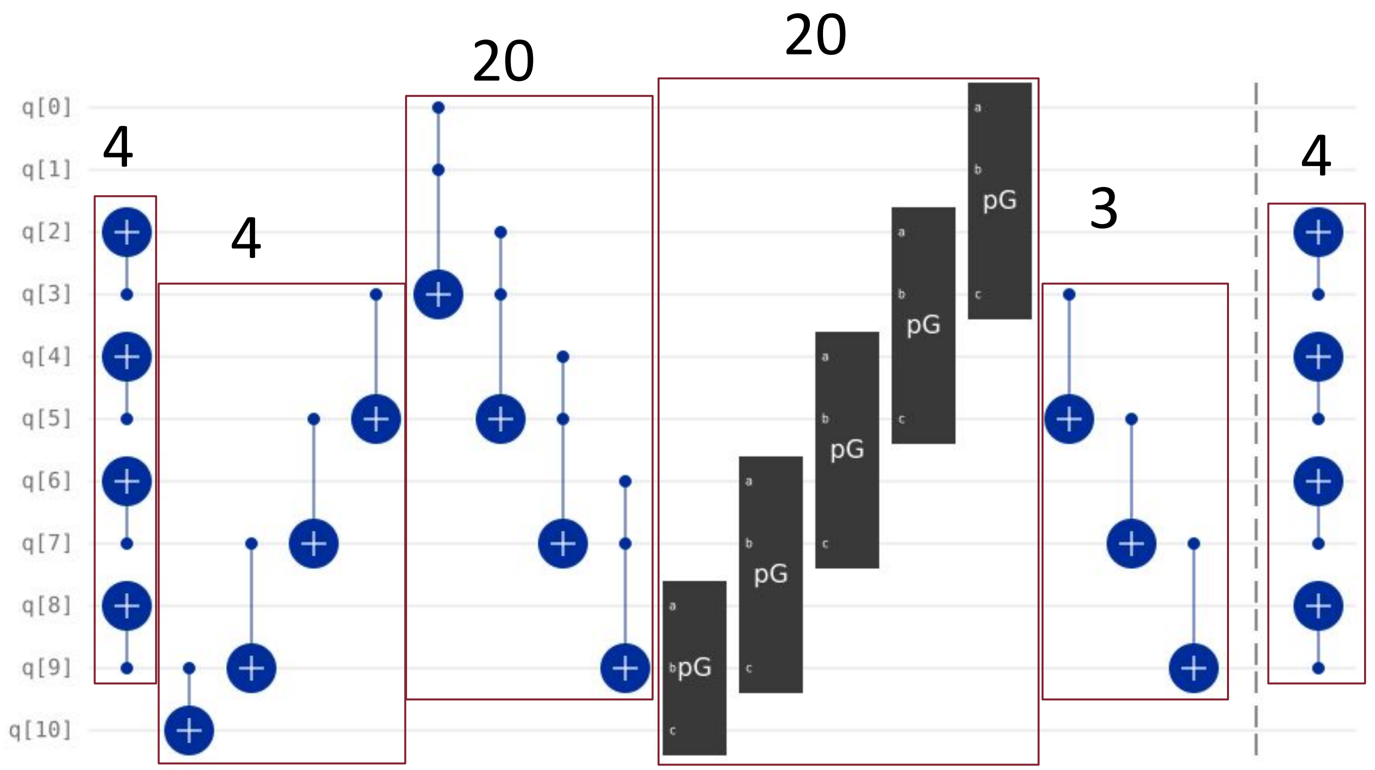
Cost Analysis

- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
  - But in the example 4 + 4 + 20 + 20 + 3



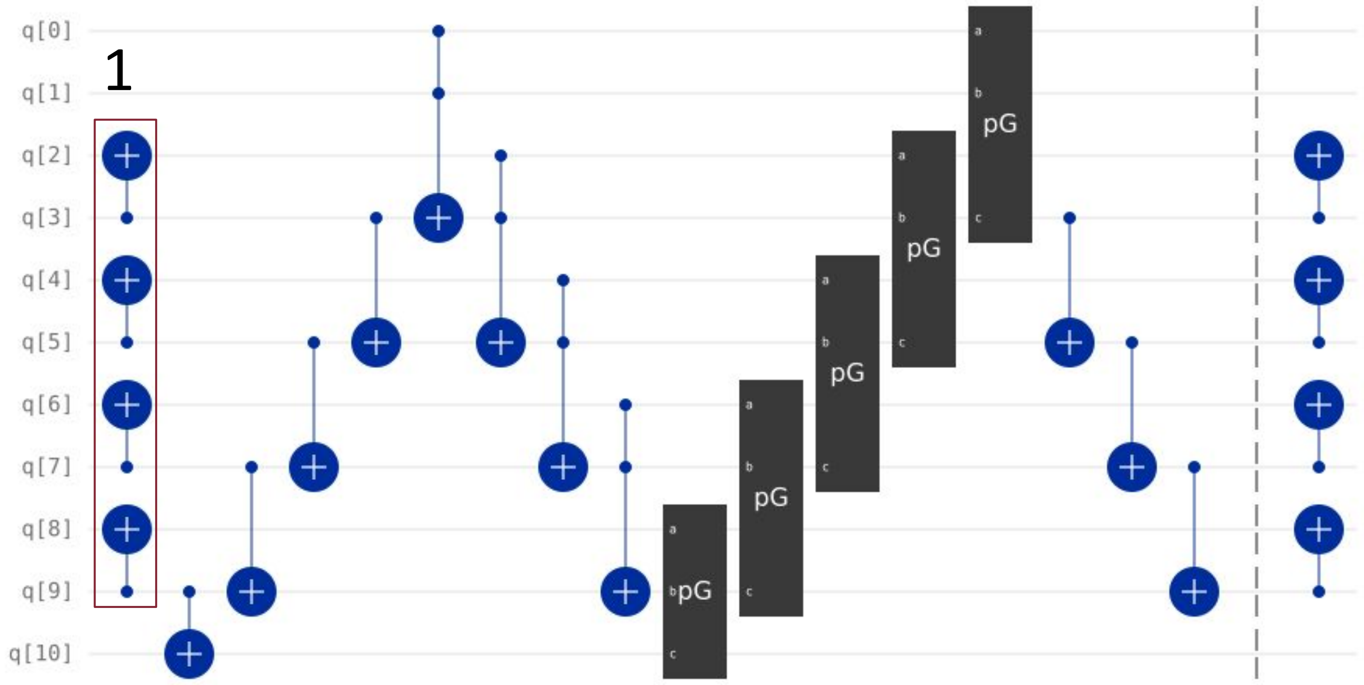
Cost Analysis

- Quantum Cost is 13N − 8 (with N = 5, the value is 57)
  - But in the example 4 + 4 + 20 + 20 + 3 + 4 = 55



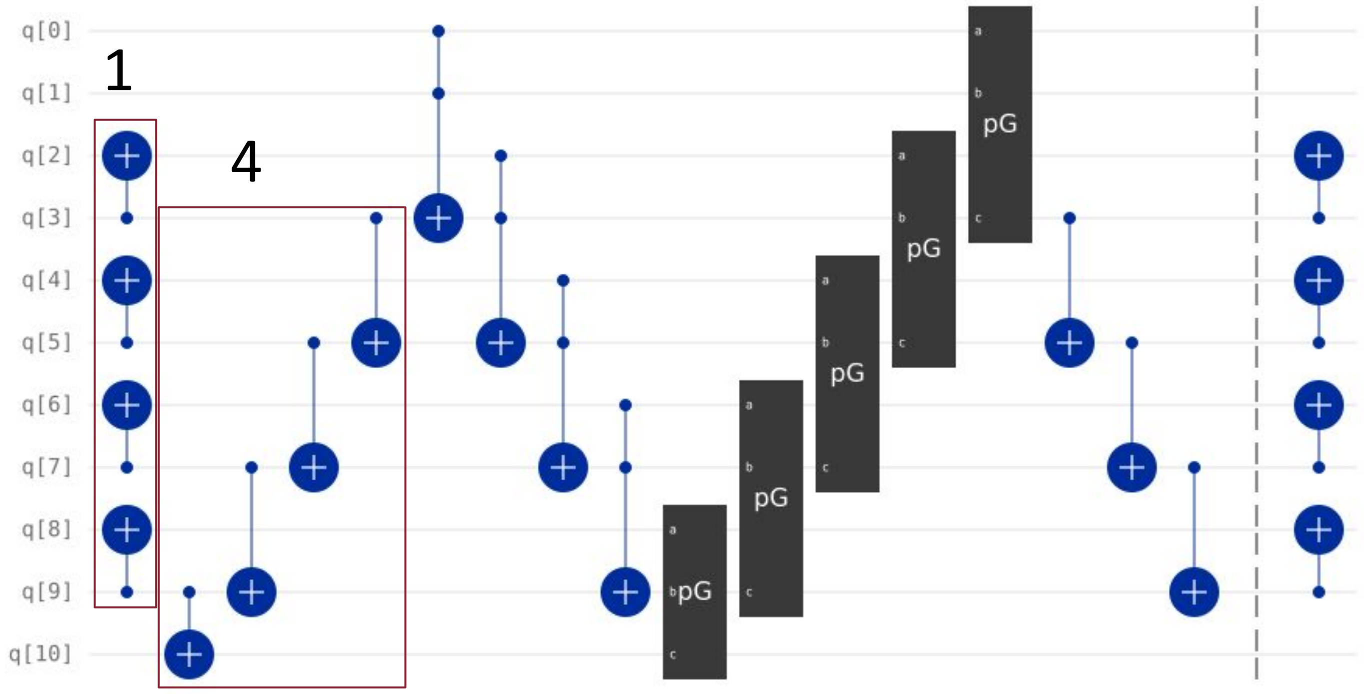
Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
    - But in this example: 1



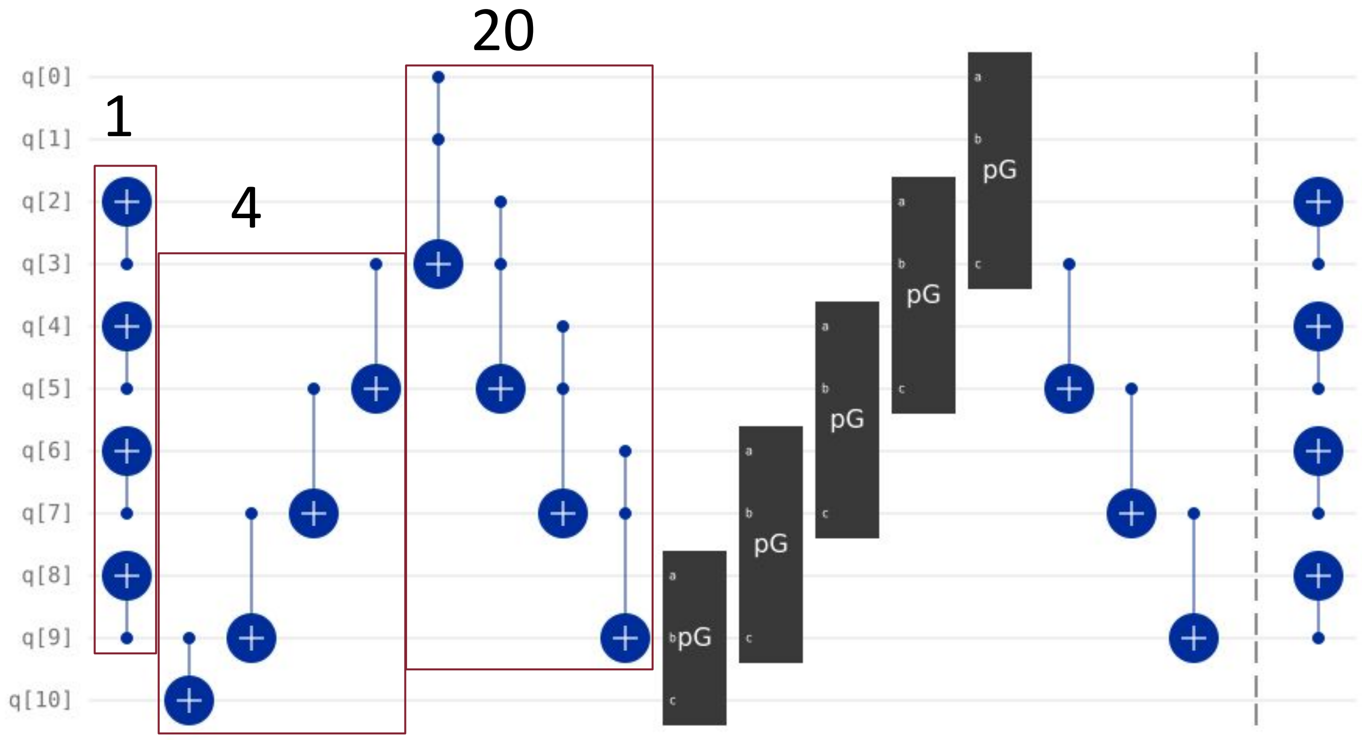
Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
  - But in this example: 1 + 4



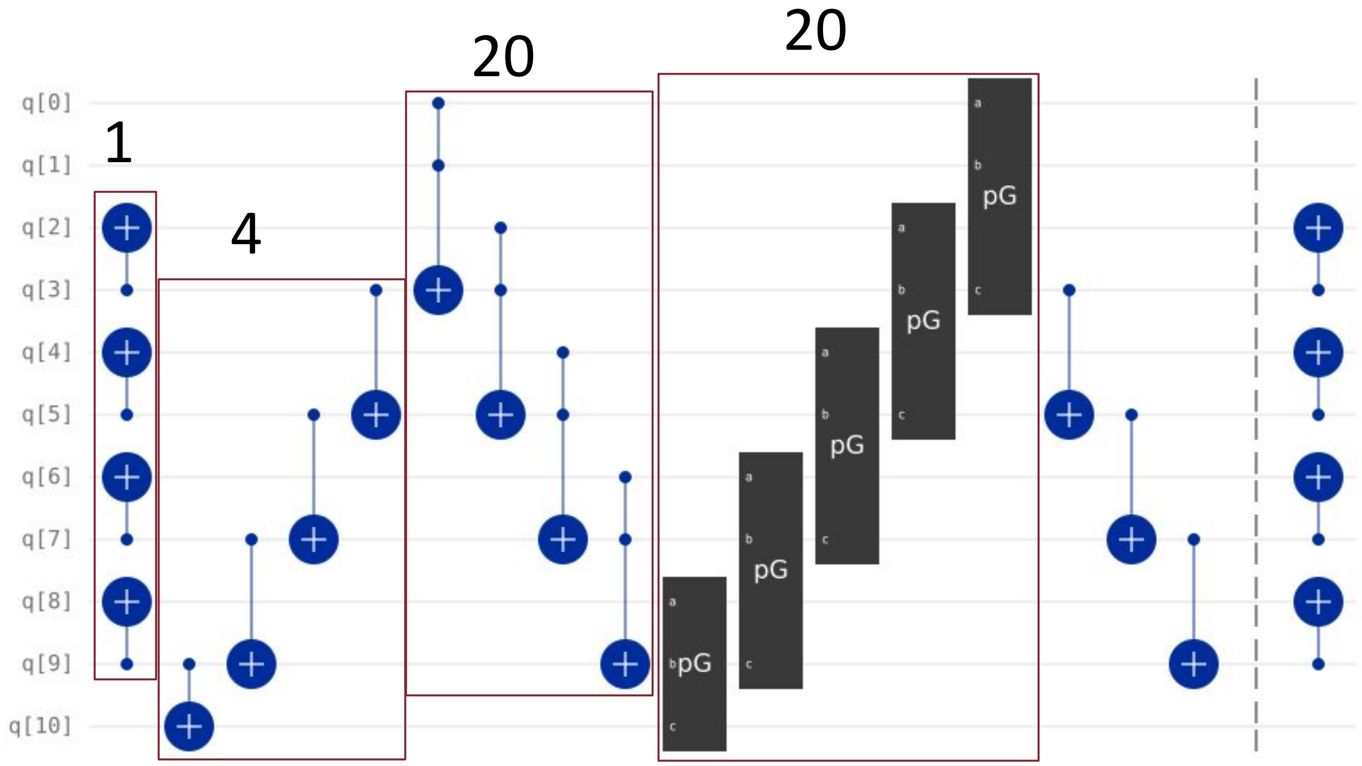
Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
    - But in this example: 1 + 4 + 4 * 5



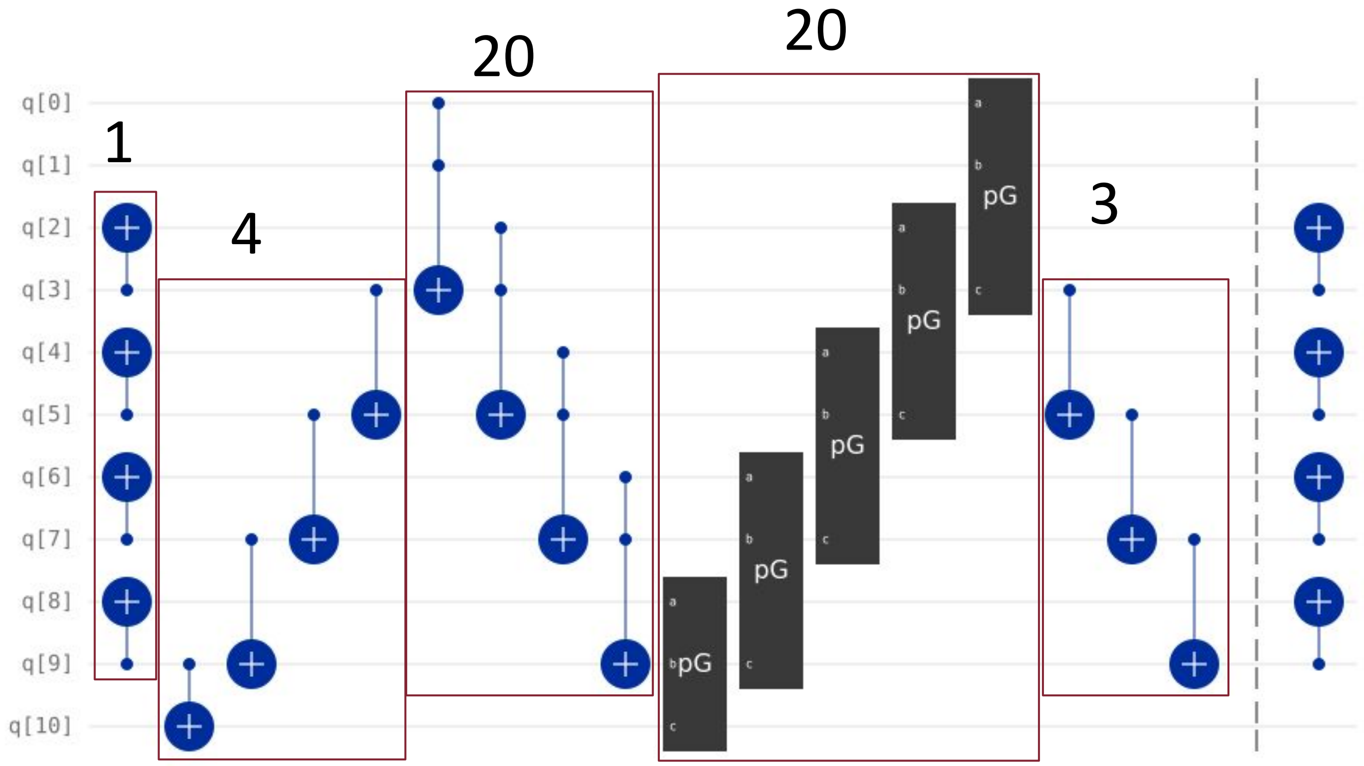
Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
  - But in this example: 1 + 4 + 4 * 5 + 5 * 4



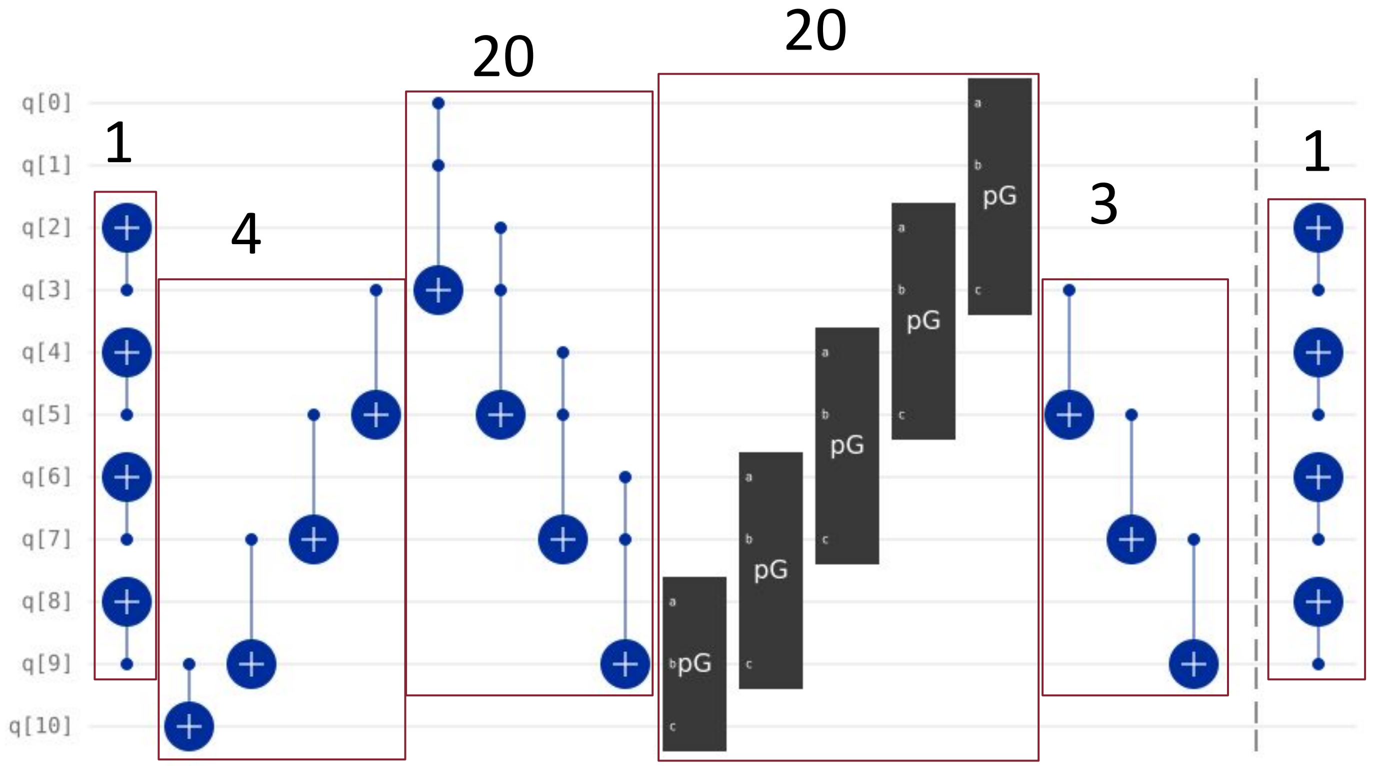
Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
  - But in this example: 1 + 4 + 4 * 5 + 5 * 4 + 3



Cost Analysis

- Propagation Delay is 11N - 4 (with N = 5, we have 51)
  - But in this example: 1 + 4 + 4 * 5 + 5 * 4 + 3 + 1 = 49



SAPIENZA
UNIVERSITÀ DI ROMA

Comparison

Comparison

- For this brief analysis we chose the following metrics
  - Quantum Cost: Number of 1x1 and 2x2 gates in the circuit.
    - 1x1 and 2x2 gates have cost 1 Unit
  - Delay: Number of 1x1 and 2x2 gates in the circuit to be executed sequentially.
    - 1x1 and 2x2 gates have 1 delay
  - Qubit Required: number of qubits needed
  - Auxiliary Inputs: inputs set to constant value
  - Garbage Outputs: outputs that cannot be used at the end
  - Fault Tolerance: capability of error suppression

SAPIENZA
UNIVERSITÀ DI ROMA

Comparison

Adder	Quantum Cost	Delay	Number of Qubits
Nagamani	12N	10N	3N + 1
Gidney	18N - 2	15N - 5	3N - 1
Takahashi	15N - 9	13N - 7	2N + 1
Thapliyal (1)	13N - 8	11N - 4	2N + 1

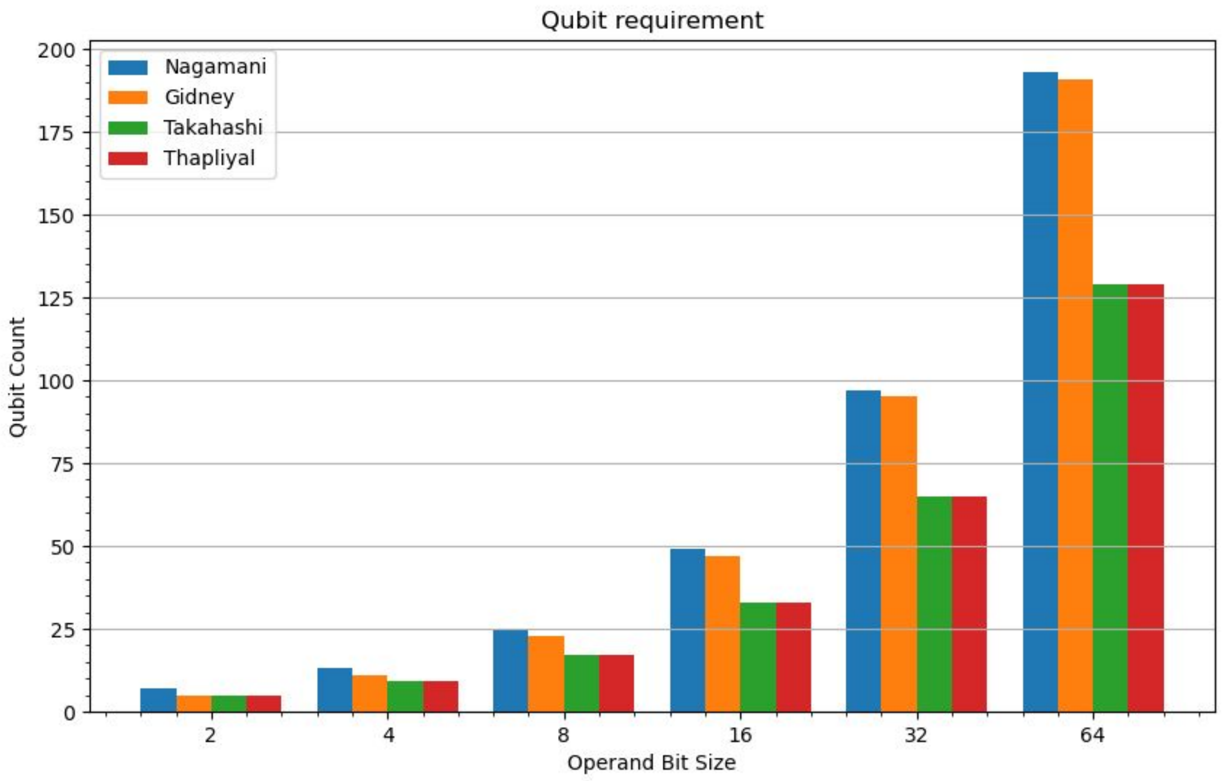
SAPIENZA
UNIVERSITÀ DI ROMA

Comparison

Adder	Ancilla Bits	Garbage Bits	Fault Tolerant	Carry In
Nagamani	4N	0	No	Yes
Gidney	N	0	Yes	No
Takahashi	0	0	No	No
Thapliyal (1)	0	0	No	No

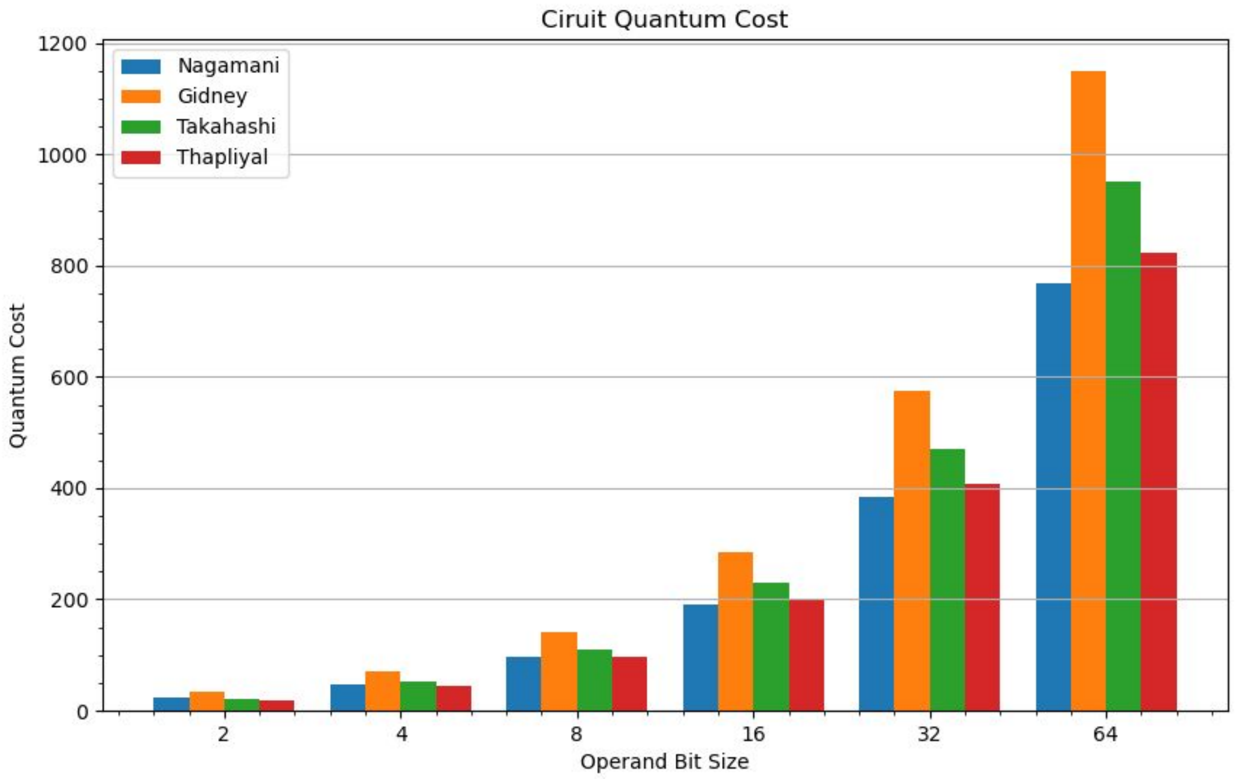
SAPIENZA
UNIVERSITÀ DI ROMA

Qubit Requirement Comparison



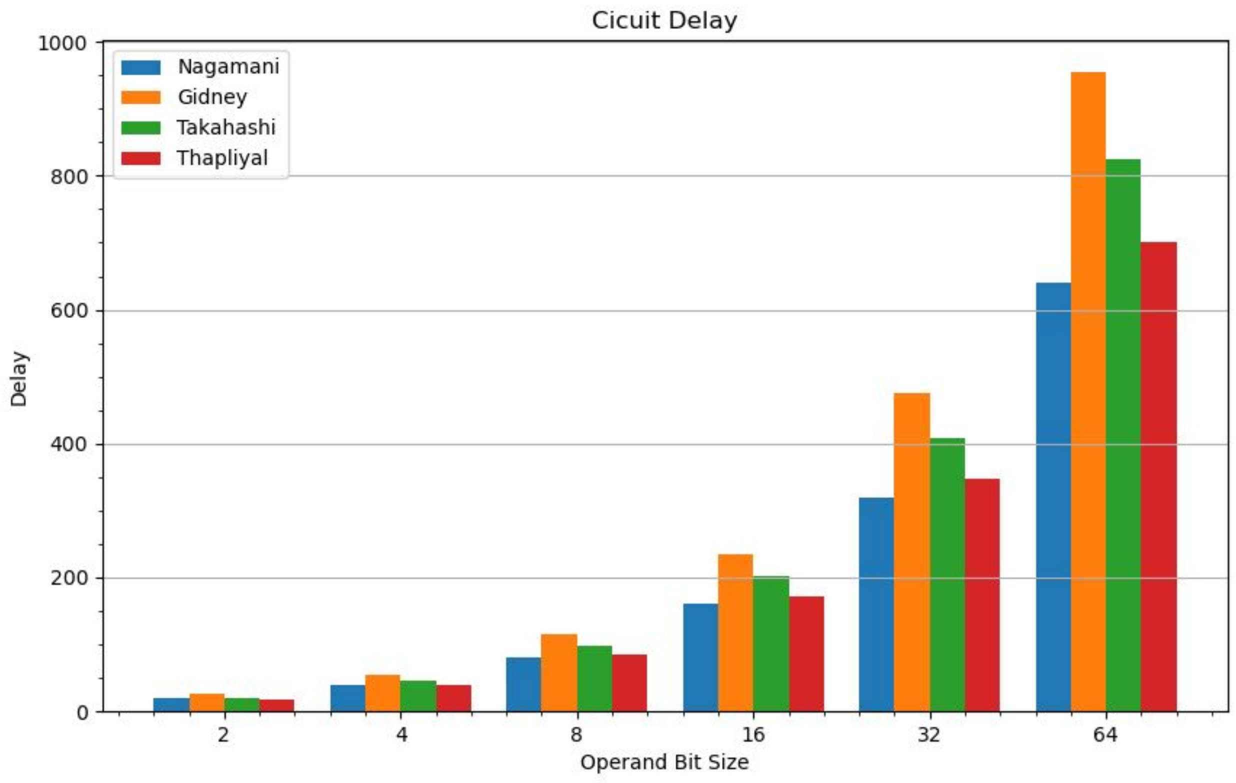
SAPIENZA
UNIVERSITÀ DI ROMA

Quantum Cost Comparison



SAPIENZA
UNIVERSITÀ DI ROMA

Delay Comparison



SAPIENZA
UNIVERSITÀ DI ROMA

References

- Nagamani, A., Ashwin, S., & Agrawal, V. (2014). Design of optimized reversible binary adder/subtractor and BCD adder. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)* (pp. 774-779).
- Gidney, C. (2018). Halving the cost of quantum addition. *Quantum, 2, 74.*
- Takahashi, Y., Tani, S., & Kunihiro, N. (2010). Quantum addition circuits and unbounded fan-out. *Quantum Info. Comput., 10(9), 872–890.*
- Thapliyal, H., & Ranganathan, N. (2013). Design of efficient reversible logic-based binary and BCD adder circuits. *J. Emerg. Technol. Comput. Syst., 9(3).*
- Orts, F., Ortega, G., Combarro, E. F., & Garzón, E. M. (2020). A review on reversible quantum adders. *Journal of Network and Computer Applications*, *170*, 102810. doi:10.1016/j.jnca.2020.102810

That’s all folks!