

# Optimizing Quantum Oracle for Standard Hash Algorithms

Korea University  
Sangheon Lee, Mingyu Cho

December 13, 2020

# Table of Contents

- 1 Introduction
  - Introduction
- 2 SHA-2 Quantum Circuit Experiment
  - Background
- 3 Clifford + T gate
- 4 Ripple-Carry Adder
- 5 References

# Introduction

# Previous Work

- Studied the basics of quantum computing and Grover's Algorithm
- Analysis of reversible quantum S-DES oracle and its construction
- Implementation of quantum S-DES oracle with Grover's Algorithm in Microsoft Q#

# Motivation

- Targeting SHA-2/3
  - Current de facto standard hash algorithm
- Goal is to perform a preimage attack.
  - The most straightforward to apply Grover's Algorithm.
  - It will be assumed that Grover's Algorithm (or some variation of it) will be applied
  - Therefore target is to optimize the quantum oracle for SHA-2/3.

# Recap: Grover's Algorithm

- Searching in an unordered database among  $2^n$  elements takes  $\tilde{O}(2^{n/2})$  time complexity and  $\tilde{O}(n)$  quantum space complexity
- Proved to be optimal in general
- Specifically, improved Grover's Algorithm for collision search yields  $\tilde{O}(2^{n/3})$  time complexity and (quantum) space complexity [BHT98]
  - However, quantum queries are costly in this algorithm

# Chailloux's Algorithm [CNPS17]

- Use poly-time qubits and reduce time complexity to  $\tilde{O}(2^{2n/5})$  for collision search and  $\tilde{O}(2^{3n/7})$  for multi-target preimage attacks with additional classic memory

# SHA-2/3 Pre-image Attack Cost Estimation [ADMG<sup>+</sup>16]

- Suggest cost metric for quantum computation based on surface code
- Theoretically implement reversible SHA-2/3 quantum circuits
- Estimate required physical resources and its scale



# Expectations

- Focus on micro-scale improvement(regarding quantum oracle)
- May be able to reduce the cost in at least one of the following metrics:
  - The cost metric from before
  - The raw number of gates
  - The number of levels after achieving parallelization
- Main target will be the cost metric, not the gates or the levels

# In-depth Topics

- Surface code(Toric code) : topological quantum error correcting code
- Cost metric based on surface code
- T-par [AMM14] : an quantum circuit optimization tool
- Advanced quantum circuit (in-place adder, *etc.*)

# SHA-2 Quantum Circuit Experiment

# SHA-256 Algorithm

---

**Algorithm 1** SHA-256. All variables are 32-bit words.

---

```

1: for i=0 to 63 do
2:    $\Sigma_1 \leftarrow (\mathbf{E} \ggg 6) \oplus (\mathbf{E} \ggg 11) \oplus (\mathbf{E} \ggg 25)$ 
3:    $\mathbf{Ch} \leftarrow (\mathbf{E} \wedge \mathbf{F}) \oplus (\neg \mathbf{E} \wedge \mathbf{G})$ 
4:    $\mathbf{t}_1 \leftarrow \mathbf{H} + \Sigma_1 + \mathbf{Ch} + \mathbf{K}[i] + \mathbf{W}[i]$ 
5:    $\Sigma_0 \leftarrow (\mathbf{A} \ggg 2) \oplus (\mathbf{A} \ggg 13) \oplus (\mathbf{A} \ggg 22)$ 
6:    $\mathbf{Maj} \leftarrow (\mathbf{A} \wedge \mathbf{B}) \oplus (\mathbf{A} \wedge \mathbf{C}) \oplus (\mathbf{B} \wedge \mathbf{C})$ 
7:    $\mathbf{t}_2 \leftarrow \Sigma_0 + \mathbf{Maj}$ 
8:    $\mathbf{H} \leftarrow \mathbf{G}$ 
9:    $\mathbf{G} \leftarrow \mathbf{F}$ 
10:   $\mathbf{F} \leftarrow \mathbf{E}$ 
11:   $\mathbf{E} \leftarrow \mathbf{D} + \mathbf{t}_1$ 
12:   $\mathbf{D} \leftarrow \mathbf{C}$ 
13:   $\mathbf{C} \leftarrow \mathbf{B}$ 
14:   $\mathbf{B} \leftarrow \mathbf{A}$ 
15:   $\mathbf{A} \leftarrow \mathbf{t}_1 + \mathbf{t}_2$ 
16: end for

```

---

Embedded images and charts in the section are from [ADMG<sup>+</sup>16].

# SHA-256 Stretching

---

**Algorithm 2** SHA-256 Stretch. All variables are 32-bit words.

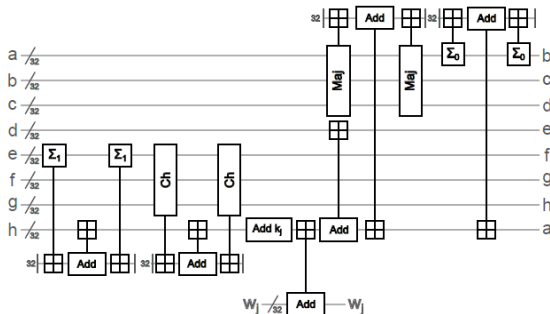
---

```
1: for  $i = 16$  to  $63$  do
2:    $\sigma_0 \leftarrow (\mathbf{W}_{i-15} \ggg 7) \oplus (\mathbf{W}_{i-15} \ggg 18) \oplus (\mathbf{W}_{i-15} \gg 3)$ 
3:    $\sigma_1 \leftarrow (\mathbf{W}_{i-2} \ggg 17) \oplus (\mathbf{W}_{i-2} \ggg 19) \oplus (\mathbf{W}_{i-2} \gg 10)$ 
4:    $w[i] \leftarrow \mathbf{W}_{i-16} + \sigma_0 + \mathbf{W}_{i-7} + \sigma_1$ 
5: end for
```

---

- Both requires bitwise operations, bit shifting and addition
- Addition can be done by Ripple-Carry Adder

# SHA-256 Quantum Circuit



- Ignore padding and precalculations
- Thesis used 2402 qubits
- But with our calculation, 801 explicit qubits were sufficient
  - 256 for hash,  $32 \times 16$  for  $w[i]$ ,  $32+1$  for ancilla

# SHA-256 Quantum Circuit Analysis

	$T/T^\dagger$	$P/P^\dagger$	$Z$	$H$	CNOT	$T$ -Depth	Depth
Round	5278	0	0	1508	6800	2262	8262
Round (Opt.)	3020	931	96	1192	63501	1100	12980
Stretch	1329	0	0	372	2064	558	2331
Stretch (Opt.)	744	279	0	372	3021	372	2907
SHA-256	401584	0	0	114368	534272	171552	528768
SHA-256 (Opt.)	228992	72976	6144	94144	4209072	70400	830720

The literature has calculated the number of required gates.

- How to calculate these?
  - Implement digital SHA-256 quantum circuit in OpenQASM[CBSG17]
- How to optimize these?
  - T-par algorithm [AMM14]
  - Implemented in Feynman toolkit<sup>1</sup>

<sup>1</sup><https://github.com/meamy/feynman>

# Implementing SHA-256 in OpenQASM 2.0

OpenQASM : *an imperative programming language for describing quantum circuits*

- Latest version : 3.0(Pre-release), 2.0
- Feynman accetps 2.0 format
- Problem: 2.0 lacks basic operations
  - Only qubits/bits available, no loops, no array slicing



# Implementing SHA-256 in OpenQASM 2.0

We made a code to generate SHA-256 OpenQASM code in Python 3.6.

- Excluding preambles like hardcoded gate definitions, 100 lines were sufficient.

```

190
191 ind = lambda i, r : (i + 8 - (r & 7)) & 7
192 wind = lambda x : '{:02d}'.format(x & 15) # x == 0
193 prepare()
194 for round_num in range(64):
195     print(f'// Round {round_num+1}')
196     print('// [+] Calculating s1')
197     for rsh in [6, 11, 25]:
198         q_xor(qregs[ind(4, round_num)], qregs[-1], rsh)
199     q_add(qregs[-1], qregs[ind(7, round_num)])
200     for rsh in [6, 11, 25]:
201         q_xor(qregs[ind(4, round_num)], qregs[-1], rsh)
202
203     print('// [+] Calculating Ch')
204     q_ch(qregs[ind(4, round_num)], qregs[ind(5, round_num)],
205         qregs[ind(6, round_num)], qregs[-1])
206
207     q_add(qregs[-1], qregs[ind(7, round_num)])
208
209     q_ch_a(qregs[ind(4, round_num)], qregs[ind(5, round_num)],
210         qregs[ind(6, round_num)], qregs[-1])

```

# Optimizing SHA-256

Generated OpenQASM code is quite long (36k lines)

- Gate calculation was successful
- However, optimizing with T-par as a whole failed due to memory limit
- Workaround: Split code into 32-64 parts

Our experiment has two criteria: optimization algorithm and round per code.

- Algorithms: `tpar`[AMM14] and `cnotmin`[AAM18]
- Code size : 1/2 round per code (64/32 parts)

# Analysis

SHA-256 Type	$T$	$H$	$X$	$CNOT$	Misc.
Literature, naïve	401584	114368	0	534272	
Ours, naïve	351232	100352	1986	465088	
Ours, T-Par, 1r	343040	100352	10730	661376	8192 <i>SWAP</i>
Ours, CNOTmin, 1r	343040	100352	10730	381824	
Ours, T-Par, 2r	343040	100352	10730	664832	8192 <i>SWAP</i>
Ours, CNOTmin, 2r	343040	100352	10730	385280	
Literature, T-Par	228992	94144	0	4209072	72976 $P$ , 6144 $Z$

'1r' and '2r' denotes 1 round per code and 2 rounds per code, respectively.

## Clifford + T gate

# Why Clifford + T?

- By Gottesman-Knill Theorem, Clifford gates can be simulated efficiently on a classical computer.
- T-gate is equal to  $R_{\pi/4} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{i} \end{pmatrix}$ , a  $\frac{\pi}{4}$  rotation.
- Clifford+T gate utilizes only one inefficient gate, while is universal.
- Since Clifford gates can be efficiently simulated, we need to minimize the number(or the depth once parallized) of the T-gates.

# Ripple-Carry Adder

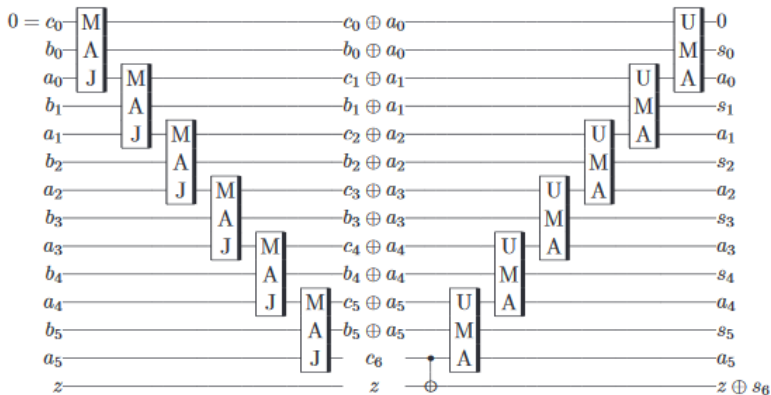





Figure 4: A simple ripple-carry adder for  $n = 6$ .




## References



# References I

-  Matthew Amy, Parsiad Azimzadeh, and Michele Mosca, *On the controlled-not complexity of controlled-not-phase circuits*, Quantum Science and Technology **4** (2018), no. 1, 015002.
-  Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck, *Estimating the cost of generic quantum pre-image attacks on sha-2 and sha-3*, International Conference on Selected Areas in Cryptography, Springer, 2016, pp. 317–337.
-  Matthew Amy, Dmitri Maslov, and Michele Mosca, *Polynomial-time  $t$ -depth optimization of clifford+  $t$  circuits via matroid partitioning*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **33** (2014), no. 10, 1476–1489.

# References II

-  Gilles Brassard, Peter Høyer, and Alain Tapp, *Quantum cryptanalysis of hash and claw-free functions*, Latin American Symposium on Theoretical Informatics, Springer, 1998, pp. 163–169.
-  Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta, *Open quantum assembly language*, 2017.
-  André Chailloux, María Naya-Plasencia, and André Schrottenloher, *An efficient quantum collision search algorithm and implications on symmetric cryptography*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2017, pp. 211–240.