Implementing Repeat-Until-Success Circuits on *SliQSim*, a BDD-based quantum circuit simulator

Chang, Ho-Mu and Chiu, Yen-Sheng
Department of Electrical Engineering,
National Taiwan University, Taipei, Taiwan
b109010{08, 16}@ntu.edu.tw

***Abstract*—Quantum circuit simulation plays a crucial role in advancing quantum computing technology. This work introduces the integration of the Repeat-Until-Success (RUS) circuit into the *SliQSim* quantum simulation engine. The RUS circuit, a non-deterministic decomposition method for single-qubit unitaries, exhibits significant advantages in terms of gate count and scalability. *SliQSim*, leveraging Binary Decision Diagrams (BDDs) for simulation, offers a novel approach to quantum circuit simulation with high accuracy and scalability.**

The integration involves several steps, including gate implementation, BDD construction, qubit rearrangement, and measurement optimization. The combined approach aims to provide a comprehensive tool for efficient quantum design automation. While the implementation enhances the capabilities of *SliQSim* in handling non-Clifford + T gates, challenges arise due to the high cost of measurements, making the rus circuit resource-intensive compared to other circuits. The study concludes with insights into potential refinements for achieving better results in *SliQSim*.

I. Introduction

As quantum computing rapidly advances, the prospect of transforming computation looms on the horizon. Within the realms of quantum hardware and software development, the significance of quantum circuit simulation becomes paramount, playing a pivotal role in testing and optimizing quantum algorithms and providing essential insights for the reliable advancement of technology.

A. RUS Circuit: Non-deterministic Decomposition for Single-Qubit Unitaries

A noteworthy contribution to quantum design automation comes in the form of the "Repeat-Until-Success" (RUS) circuit—a decomposition technique utilizing non-deterministic circuits to approximate single-qubit unitaries [3]. Achieving high precision within a defined distance (ε), RUS circuits demand significantly fewer non-Clifford gates than existing methods . Operating by conditioning on specific measurement outcomes with minimal non-Clifford gates and ancilla qubits, the RUS algorithm excels in approximating Z-axis rotations. Extensions of the algorithm exhibit scaling improvements, outperforming deterministic decomposition methods.

B. SliQSim: BDD-Based Quantum Circuit Simulation

In response to the expansive Hilbert space of quantum states, a paradigm shift in quantum circuit simulation is introduced through *SliQSim*—an engine meticulously designed for accuracy and scalability [1] [4]. This innovative approach replaces classical matrix-vector multiplication with Binary Decision Diagrams (BDDs), offering a novel method for implementing quantum circuit simulation. *SliQSim* leverages an algebraic representation for precision and employs a bit-slicing strategy to enhance scalability. Through symbolic Boolean function manipulation, the engine demonstrates superior performance in simulating a variety of quantum circuits. Experimental results highlight its exceptional capabilities, particularly in handling benchmark families with dimensions extending to tens of thousands of qubits.

C. Integration: Extending SliQSim with RUS Circuit

This work seamlessly integrates RUS circuits into the *SliQSim* engine by introducing a dedicated quantum gate design. Enhancing *SliQSim*'s capabilities with non-deterministic decomposition for single-qubit unitaries, this integration aims to provide a comprehensive tool for efficient quantum design automation. By synergizing RUS circuit-based decomposition with *SliQSim*'s advanced quantum circuit simulation, this novel approach contributes significantly to the ongoing progress in quantum computing research.

II. Implementation

A. Overview

Implementing the Repeat-Until-Success (RUS) circuit in a simulation tool differs slightly from its physical counterpart. In simulation, the entire state vector is known, eliminating the need for literal repetition. Instead, the state can be directed to collapse deliberately. The key distinction lies in the sampling process.

Our implementation comprises several steps:

(1) Integrating the rus gate into the parsable gate list within *SliQSim*.
(2) Constructing a *monolithic* Binary Decision Diagram (BDD) for measurement.
(3) Rearranging the to-be-measured qubits to the top of the BDD.
(4) Measuring along the desired BDD branch.
(5) Collapsing the states of the to-be-measured qubits and resetting the measured qubits.

In the following sections, we delve into the specifics of each implementation step. It is noteworthy that we have made certain simplifications to the quantum simulation methodology, deviating from real-world physical implementations. To streamline the process without significantly affecting simulation results, we manually determine the state to which ancillary bits collapse. Since the outcome predominantly depends on the *success* stage (measured qubits turning out to be in the desired states) of the RUS circuit, we guide the qubits to collapse into this state, yielding the final output. Additionally, we choose to disregard restrictions on Clifford + T gates, allowing for the inclusion of gates beyond Clifford + T to diversify our experimental approach.

B. Detailed introduction

1) Integrating the rus *Gate into the Parsable Gate List within SliQSim: SliQSim* supports reading Open Quantum Assembly Language (OPENQASM) [4], and we've designed a gate rus in a format similar to other gates. In the RUS circuit, we design m to-be-measured ancillae bits, assigned with a vector $\mathbf{q}_{\mathrm{m}} = (q_{\mathrm{m},\ i})_{i=1}^{m}$, and other n not-to-be-measured qubits that carry the information of the original state vector [2].

In the rus gate, since we need to mark the *success* or *failure* in every measurement, we must also assign the corresponding desired state vector $\mathbf{s}_{\mathrm{d}} = (s_{\mathrm{d},\ i})_{i=1}^{m}$. That is, when it's measured that $\mathbf{q}_{\mathrm{m}} = \mathbf{s}_{\mathrm{d}}$, the rus gate reports *success*.

We design the syntax of the rus gate as follows:

```
rus qm1, qm2, ..., [s1], [s2], ...
```

where qm1, qm2, ... are the qubits $\mathbf{q}_{\mathrm{m}}$, and [s1], [s2], ... are the desired state specifications in $\mathbf{s}_{\mathrm{d}}$.

For example:

```
OPENQASM 2.0;
include "qelib1.inc";
h q[0];
h q[1];
ccx q[0], q[1], q[2];
s q[2];
ccx q[0], q[1], q[2];
z q[2];
h q[0];
h q[1];
rus q[0], q[1], [0], [0];
```

In the quantum circuit, we denote the rus as the circuit in Figure 1 below:

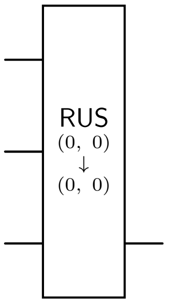

Fig. 1. The symbol of the rus gate, where $(m,\ n) = (2,\ 1)$.

2) Constructing a Monolithic Binary Decision Diagram (BDD) for Measurement: In the paper proposing BDD-based Quantum Circuit Simulation [1], methods for measuring BDD qubits are introduced. It has been established that any complex value α required for the Clifford + T gate set can be represented in the following format:

$$\alpha = \frac{1}{\sqrt{2}^{k}} \sum_{t=0}^{3} a_t \mathrm{e}^{\frac{\mathrm{i}t\pi}{4}}, \tag{1}$$

where $(a_3, a_2, a_1, a_0, k) \in \mathbb{Z}^5$. Now, consider a state vector $|\psi\rangle$ of size 2^n representing an n-qubit state. We need corresponding integer vectors for a_t, $\mathbf{a}_t$, to denote $|\psi\rangle$:

$$|\psi\rangle = \frac{1}{\sqrt{2}^{k}} \sum_{t=0}^{3} \mathbf{a}_t \mathrm{e}^{\frac{\mathrm{i}t\pi}{4}}, \tag{2}$$

where $\forall\, t \in \{0,1,2,3\}\, \mathbf{a}_t \in \mathbb{Z}^{2^n \times 1}$. Moreover, for an integer n, we need a bit string of size $r = \lfloor \log_2 n \rfloor + 1$ to store it (especially, $n = 0$ doesn't apply for this formula, but we know we need 1 bit to store it). We can introduce corresponding bit vectors for $\mathbf{a}_t$, $\mathbf{a}_{tu}$, to denote $|\psi\rangle$:

$$|\psi\rangle = \frac{1}{\sqrt{2}^{k}} \sum_{t=0}^{3} \left(\sum_{u=0}^{r} \mathbf{a}_{tu} 2^u \right) \mathrm{e}^{\frac{\mathrm{i}t\pi}{4}}, \tag{3}$$

where r is the maximum bit string length required to store every integer in $\mathbf{a}_t$, and $\forall\, (t,u) \in \{0,1,2,3\} \times \{0,1,\cdots,r\}\, \mathbf{a}_{tu} \in \mathbb{B}^{2^n \times 1} = \{0,1\}^{2^n \times 1}$. As a result, a state vector $|\psi\rangle \in \mathbb{C}^{2^n \times 1}$ is sliced into $4r$ bit vectors $\mathbf{a}_{tu} \in \mathbb{B}^{2^n \times 1}$.

It's worth noting that the 2^n entries of the bit vectors correspond to the 2^n possibilities of the value of the n qubits, which effectively works as a truth table for a boolean function and can be transformed into a BDD.

The idea is that, with the introduction of more boolean variables, we can build larger and larger BDDs, eventually constructing the *Monolithic* BDD $F^{|\psi\rangle}$. Note that it has already been shown that we need $4r$ BDDs to represent a state vector. Correspondingly, we may assume we need $s = \lceil \log_2 4r \rceil = \lceil \log_2 r \rceil + 2$ more variables to encode the *Monolithic* BDD $F^{|\psi\rangle}$. Assume the variables $\{x_i\}_{i=0}^{s-1}$, the BDD of $\mathbf{a}_{ut}$ is $F^{\mathbf{a}_{ut}}$, and the BDD of $\mathbf{a}_u$ is $F^{\mathbf{a}_u}$, and the BDD of $|\psi\rangle$ is $F^{|\psi\rangle}$. Note our goal is to obtain $F^{|\psi\rangle}$. The paper gives the incremental build of the BDD:

$$\forall\, t \in \{0,1,2,3\}\, F^{\mathbf{a}_u} = \bigvee_{t=0}^{r-1} g_t F^{\mathbf{a}_{ut}}, \tag{4}$$

$$F^{|\psi\rangle} = \bigvee_{u=0}^{3} h_u F^{\mathbf{a}_u}, \tag{5}$$

Initial State	*qiskit* Result q_q	*SliQSim* Result q_S	
$\|0\rangle$	$0.577\|0\rangle + 0.816\mathrm{i}\|1\rangle$	$(0.408-0.408\mathrm{i})\|0\rangle + (0.577+0.577\mathrm{i})\|1\rangle$	$\approx e^{0.785\mathrm{i}}q_\mathrm{q}$
$\|+\rangle$	$(0.408+0.577\mathrm{i})\|0\rangle + (0.408+0.577\mathrm{i})\|1\rangle$	$(0.697+0.120\mathrm{i})\|0\rangle + (0.697+0.120\mathrm{i})\|1\rangle$	$\approx e^{0.785\mathrm{i}}q_\mathrm{q}$
$\|\mathrm{i}\rangle$	$-0.816\|0\rangle + 0.577\mathrm{i}\|1\rangle$	$(-0.577+0.577\mathrm{i})\|0\rangle + (0.408+0.408\mathrm{i})\|1\rangle$	$\approx e^{0.785\mathrm{i}}q_\mathrm{q}$
$\|\psi\rangle$	$(-0.247+0.475\mathrm{i})\|0\rangle, (-0.187+0.824\mathrm{i})\|1\rangle$	$(0.162+0.510\mathrm{i})\|0\rangle + (0.451+0.714\mathrm{i})\|1\rangle$	$\approx e^{0.785\mathrm{i}}q_\mathrm{q}$

TABLE I
CORRECTNESS VERIFICATION RESULTS. $|\psi\rangle = (0.53+0.427\mathrm{i})|0\rangle + (0.28+0.677\mathrm{i})|1\rangle$.

where

$$g_t = \begin{cases} x_2x_3\cdots x_s, & t = 11\cdots 1_{(2)} = (r-1)_{(10)} \\ x_2x_3\cdots \overline{x_s}, & t = 11\cdots 0_{(2)} = (r-2)_{(10)} \\ \vdots & \vdots \\ \overline{x_2x_3}\cdots x_s, & t = 00\cdots 1_{(2)} = 1_{(10)} \\ \overline{x_2x_3}\cdots \overline{x_s}, & t = 00\cdots 0_{(2)} = 0_{(10)}, \end{cases} \tag{6}$$

$$h_u = \begin{cases} x_0x_1, & u = 11_{(2)} = 3_{(10)}, \\ x_0\overline{x_1}, & u = 10_{(2)} = 2_{(10)}, \\ \overline{x_0}x_1, & u = 01_{(2)} = 1_{(10)}, \\ \overline{x_0x_1}, & u = 00_{(2)} = 0_{(10)}. \end{cases} \tag{7}$$

In the process of constructing $F^{|\psi\rangle}$, we can employ functions provided by the `cudd` package to build the BDDs of the bit vectors $F^{\mathbf{a}_{ut}}$ first. Subsequently, we can use the AND and OR operations of BDDs, implemented by the `Cudd_bddAnd` and `Cudd_bddOr` functions repeatedly to construct the *Monolithic* BDD $F^{|\psi\rangle}$. After successfully building $F^{|\psi\rangle}$, we can proceed with the measurement on this *Monolithic* BDD [4].

3) Rearranging the To-be-measured Qubits to the Top of the BDD: Note that qubits essentially function as boolean variables in the fundamental BDD $F^{\mathbf{a}_{ut}}$, so they must be variables of the $F^{|\psi\rangle}$ as well. To perform the measurement, we must alter the order of these variables, ensuring that the to-be-measured qubits, or the qubits in the vector $\mathbf{q}_\mathrm{m}$, move closer to the root node or, essentially, to the "top" of the BDD, as shown in the Figure 2 below. This can be accomplished effortlessly by utilizing the `Cudd_ShuffleHeap` function [4].

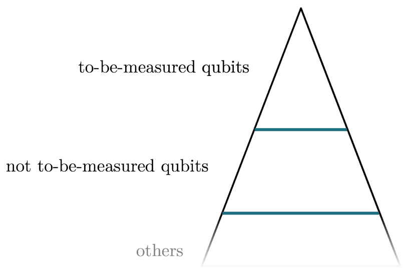

Fig. 2. Schematic diagram of the rearrangement of the qubits. The inverted V represents the BDD, with the apex indicating the root node. The to-be-measured qubits are assigned higher priority orders.

4) Measuring Along the Desired BDD Branch: Following the variable rearrangement, we can now compute the probability of the *success* state in the RUS circuit. By iteratively querying the `then` or `else` branch in the BDD (depending on whether the desired value is 1 or 0), we can determine the probability of different state vectors and evaluate their likelihood [4]. Let $\mathbf{q}$ represent the vector of all $n+m$ qubits, $\mathbf{q}_\mathrm{m}$ denote the to-be-measured ancillary qubits with m qubits, $\mathbf{s}_\mathrm{d}$ indicate the corresponding desired state, and $\mathbf{q}_\mathrm{others}$ encompass the not-to-be-measured qubits with n qubits. The probability of *success*, denoted as p_success, is given by:

$$p_\mathrm{success} = \Pr[\mathbf{q}_\mathrm{m} = \mathbf{s}_\mathrm{d}] \tag{8}$$

$$= \sum_{\mathbf{q}_\mathrm{others}\in\mathbb{B}^n} \Pr(\mathbf{q} = \{\mathbf{s}_\mathrm{d}, \mathbf{q}_\mathrm{others}\}), \tag{9}$$

where the notation $\{\cdot,\cdot\}$ signifies the concatenation of two vectors.

This involves computing the summation over all possible states. Although sampling the qubits is not necessary in this context, the probability is used to represent the normalization factor for state recovery in the final step—resetting the measured qubits.

5) Collapsing the States of the To-be-Measured Qubits and Resetting the Measured Qubits: In this concluding step, we collapse the state of the to-be-measured ancillary qubits $\mathbf{q}_\mathrm{m}$ to their corresponding desired state $\mathbf{s}_\mathrm{d}$ and apply corresponding modifications to the BDD. The state information of the remaining qubits can be efficiently gathered through cofactoring with respect to $\mathbf{q}_\mathrm{m}$. Specifically, if we denote the original and new BDDs as F and F_collapse, respectively:

$$F_\mathrm{collapse} = F_{\mathbf{q}_\mathrm{m}=\mathbf{s}_\mathrm{d}} \wedge (\mathbf{q}_\mathrm{m} = \mathbf{0}), \tag{10}$$

which universally holds for all BDDs. The cofactor process can be facilitated by the `Cudd_Cofactor` function [4]. Furthermore, it's crucial to note that while in a physical device, the measured bits would persist in the success state, for simulation purposes, we can repurpose the ancillary bits, initializing these bits to $|0\rangle$ by default. A image explanation can be elucidated with reference to Figure 3.

III. EXPERIMENTAL RESULTS

A. Correctness Verification

In the first experiment, we execute the circuit depicted in Figure 4, representing a Repeat-Until-Success (RUS) circuit that implements the unitary transformation:

$$\frac{I + \mathrm{i}\sqrt{2}X}{\sqrt{3}} \approx R_X(-54.74°) \tag{11}$$

Running Times	*qiskit* Result q_{q}	*SliQSim* Result q_{S}
1	$(0.258 + 0.516\mathrm{i})\ket{0} + (0.730 + 0.365\mathrm{i})\ket{1}$	$(0.516 - 0.258\mathrm{i})\ket{0} + (0.365 - 0.730\mathrm{i})\ket{1}$
256	$(-0.635 + 0.413\mathrm{i})\ket{0} + (0.584 + 0.292\mathrm{i})\ket{1}$	$(-0.635 + 0.413\mathrm{i})\ket{0} + (0.584 + 0.292\mathrm{i})\ket{1}$
512	$(-0.194 - 0.524\mathrm{i})\ket{0} + (-0.742 - 0.371\mathrm{i})\ket{1}$	$(-0.194 - 0.524\mathrm{i})\ket{0} + (-0.742 - 0.371\mathrm{i})\ket{1}$

TABLE II
PRECISION TEST RESULTS COMPARISON

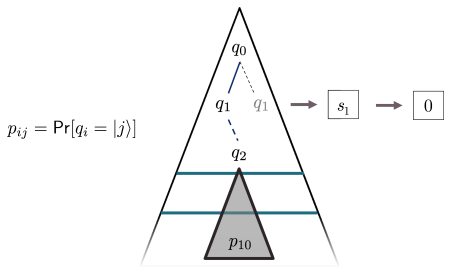

Fig. 3. Schematic diagram illustrating the collapsing and resetting of qubit states. Assuming q_1 is a to-be-measured qubit, we measure along the branch from the root node q_0 to it (the solid line represents the `then` branch, while the dotted line represents the `else` branch). The probability $p_{10} = \Pr[q_1 = \ket{0}]$ can be calculated. After collapsing to its desired state, the qubit is reset to $\ket{0}$.

(up to a global phase). The simulation is performed using the extended *SliQSim* with the `rus` gate and *qiskit*, an open-source toolkit for quantum computing, executing any desired unitary gate. We then compare the results obtained from both simulations.

The circuit diagram is presented in Figure 4, and the corresponding results are outlined in Table I. The observed outcomes closely align with the expected values.

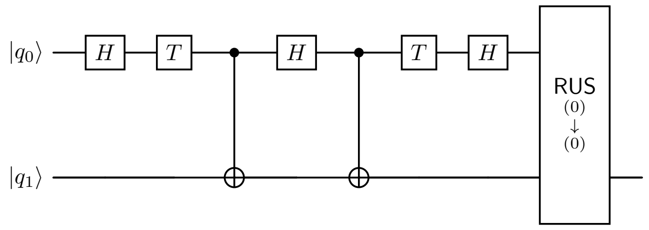

Fig. 4. Circuit for $R_X(-54.74°)$.

B. Precision Loss Test

As both the RUS circuit and floating-point calculations (for probability) introduce precision loss, we conduct a test on a long circuit containing multiple RUS circuits to evaluate the impact.

In this experiment, we utilize another RUS circuit shown in Figure 5. The circuit implements:

$$V_3 = \frac{I + 2\mathrm{i}Z}{\sqrt{5}} \approx R_Z(-1.1071487), \qquad (12)$$

if both the first two qubits are measured to be $\ket{0}$.

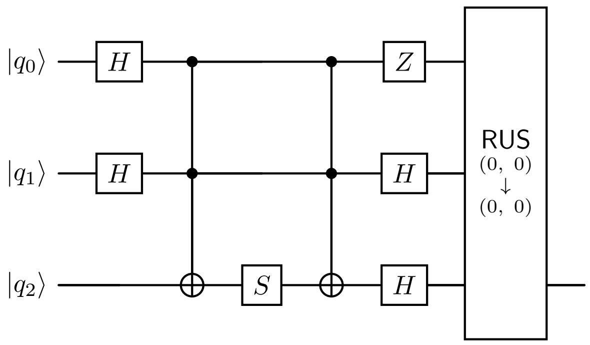

Fig. 5. Circuit for V_3.

We run $R_X(-54.74°)\,V_3$, $[R_X(-54.74°)\,V_3]^{256}$, and $[R_X(-54.74°)\,V_3]^{512}$ respectively. We then check if the circuit remains within an acceptable correctness range.

The results in Table II demonstrate that the states remain almost equivalent even for a long circuit.

However, it's important to note that running the experiment 1024 times may result in incorrect (`nan`) outcomes. This issue arises due to the precision limitations of floating-point calculations, specifically when the probability of success is extremely low and cannot be accurately represented in a `double` of IEEE 754 format. Potential solutions include choosing a circuit with a higher success probability, utilizing a higher-precision floating format, or rebuilding the BDD with the normalization factor added.

IV. CONCLUSIONS

We have successfully implemented the Repeat-Until-Success (RUS) circuit on *SliQSim*, and the code is available `here`. This achievement allows us to efficiently decompose non-Clifford + T gates, making circuits, such as VQE, QAOA, or those requiring small-angle rotations, more feasible on *SliQSim*. The RUS decomposition, coupled with the inherent advantages of *SliQSim*, including high time and memory efficiency and minimal precision loss with basic gates, enhances the execution of various quantum algorithms.

However, the substantial costs associated with both decomposition and measurement pose challenges, especially when inserting multiple RUS gates, leading to increased execution times. Therefore, optimizing the measurement procedure becomes crucial. Additionally, while the second experiment demonstrates acceptable precision loss, the accumulation of success probabilities may result in a total probability/normalization factor too small to be accurately recorded. Consequently, reforming the Binary Decision Diagram (BDD) with the normalized state might be necessary when utilizing numerous low success probability RUS circuits.

It is important to note that, despite the excellent performance of RUS circuits in physical implementations, the direct application to *SliQSim* reveals challenges. The relatively high cost of measurements in *SliQSim* diminishes some advantages of the RUS circuit, making it resource-intensive compared to circuits with the same gate count. Moreover, while the RUS circuit excels based on the T count, *SliQSim* handles T gates more efficiently than other gates. These observations suggest that refining the RUS decomposition algorithm may be necessary to achieve better results on *SliQSim*.

References

[1] Y. -H. Tsai, J. -H. R. Jiang and C. -S. Jhang, ”Bit-Slicing the Hilbert Space: Scaling Up Accurate Quantum Circuit Simulation,” 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2021, pp. 439-444, doi: 10.1109/DAC18074.2021.9586191.

[2] A. Bocharov, M. Roetteler, and K. M. Svore, “Efficient synthesis of universal repeat-until-success quantum circuits,” Phys. Rev. Lett., vol. 114, no. 8, p. 080502, Feb. 2015.

[3] A. Paetznick and K. M. Svore, “Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries,” in Proc. IEEE International Conference on Quantum Computing and Engineering (QCE), 2014, pp. 146–152.

[4] NTU-ALComLab. *SliQSim*. GitHub repository. Available at: https://github.com/NTU-ALComLab/SliQSim. Accessed: 2023.12.23.