SQUARE: Strategic Quantum Ancilla Reuse for Modular Quantum Programs via Cost-Effective Uncomputation

Yongshan Ding, Xin-Chuan Wu, Adam Holmes, Ash Wiseth, Diana Franklin, Margaret Martonosi, and Frederic T. Chong

June 3, 2020

Presentation by Vahagn Tovmasian

UC RIVERSIDE

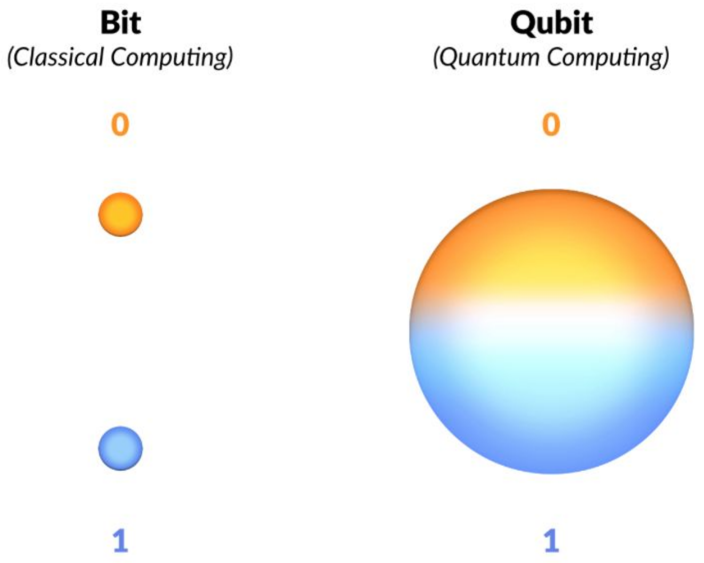
Overview

Introduction

Introduction

Basic Principles

- **Superposition**
  - Quantum State → Linear combination of discrete states.
  - Amplitude of states adds to 1
- **Measurement**
  - Collapses quantum state
  - Irreversible & Probabilistic
- **Transformation**
  - Unitary gates
  - Deterministic & Reversible**



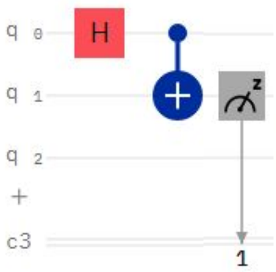
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

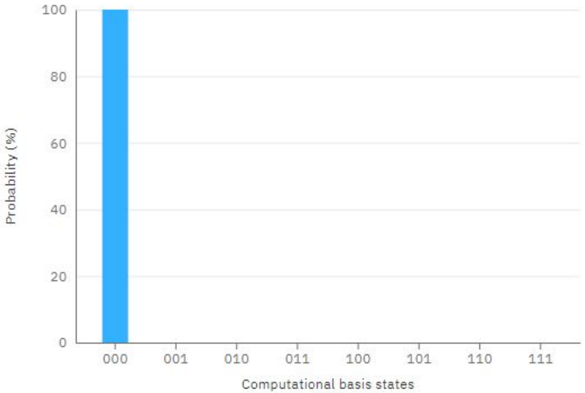
$$|\alpha|^2 + |\beta|^2 = 1$$

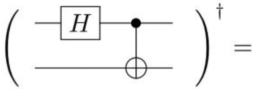
Introduction

Basic Principles

- Circuits basis of computation
- Use **gates** to manipulate qubits
- All gates ***must*** be **reversible**.



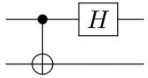




$$= (\mathrm{CNOT}(H \otimes I))^\dagger =$$

$$= (H^\dagger \otimes I^\dagger)\, \mathrm{CNOT}^\dagger =$$

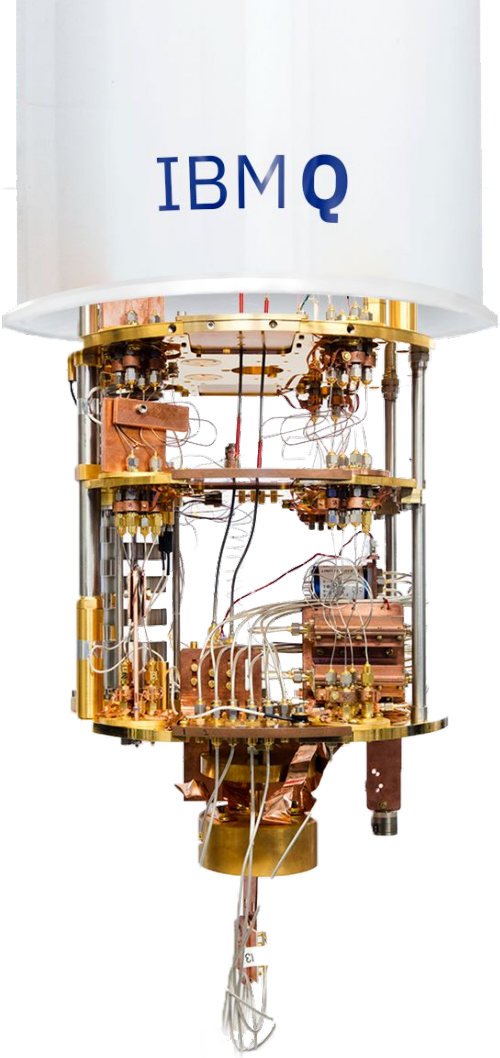
$$= (H \otimes I)\, \mathrm{CNOT} =$$

$=$ 

Introduction

Modern Quantum Computing

- We are in the **Noisy Intermediate-Scale Quantum (NISQ) Computing Era**
  - Heavily Space and Time constrained
    - Space → # of qubits
    - Time → more operations → more noise & decoherence
- Eventually we'll be in **Fault Tolerant (FT) Quantum Computing Era**
  - Less Space and Time constrained
  - Research focused on getting to this stage of QC



UC RIVERSIDE

Introduction

What is SQUARE?

Compilation Framework For Resource Constrained QCs

Heuristic Greedy Allocation and Reclamation Algorithms

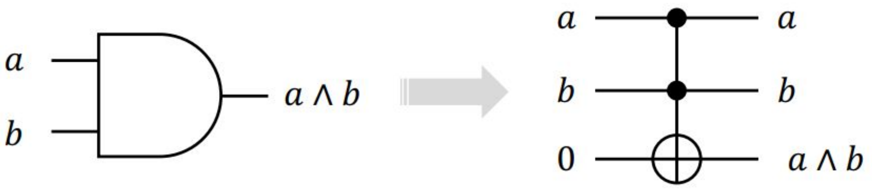
Uses Uncomputation to reset Ancilla Qubits

SQUARE Background

SQUARE Background

Reversible Arithmetic

- **Reversibility**
  - Use Ancilla Qubits for “scratch” work
- **Modularity**
  - Complex algorithms are not easily reversed
  - Solution?
    - Use modular subroutines w/ determined reversibility
- **Applications**
  - Classical parts of quantum algos
- **Constraints**
  - More complex circuits → more ancilla bits



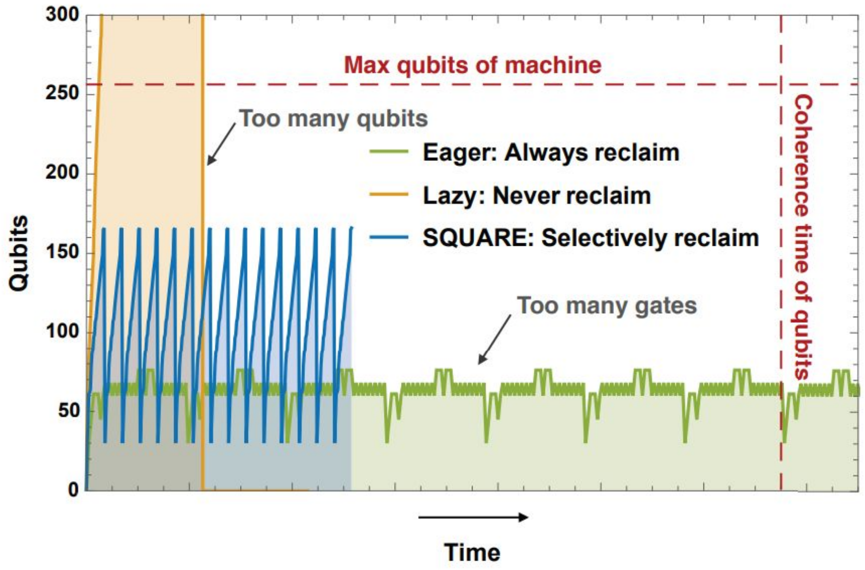
AND gate   Toffoli gate

UC RIVERSIDE

SQUARE Background

Limitations & Constraints

- **Must satisfy architectural constraints**
  - Instruction Set Architecture (ISA)
  - Qubit Communication
- **Must intelligently reclaim qubits**
- **Qubit Communication**
  - NISQ vs FT QCs
- **Resetting will affect entangled qubits**
  - How do you reset then?
  - When should you reset them?



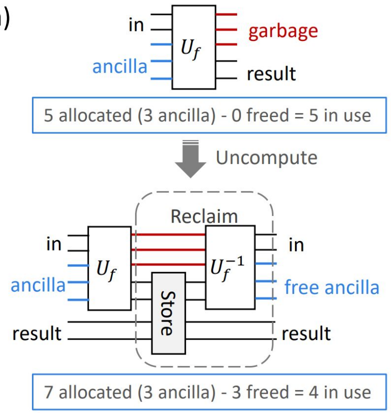
UC RIVERSIDE

SQUARE Background

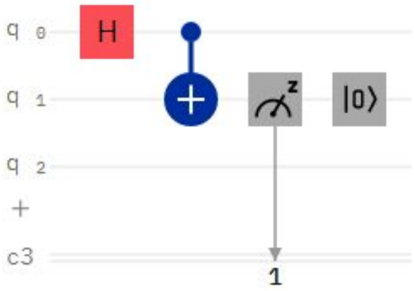
Reclamation Techniques

- **Uncomputation**
  - Reverse the logic until you get original input
  - Must be classically reversible gates
  - Potentially increases time complexity

(a)



- **Measure & Reset**
  - Collapse superposition then reset to zero
  - Decoupled Ancilla Bits only
  - Requires efficient reset
    - Most QCs just wait till decoherence occurs (order of ms)

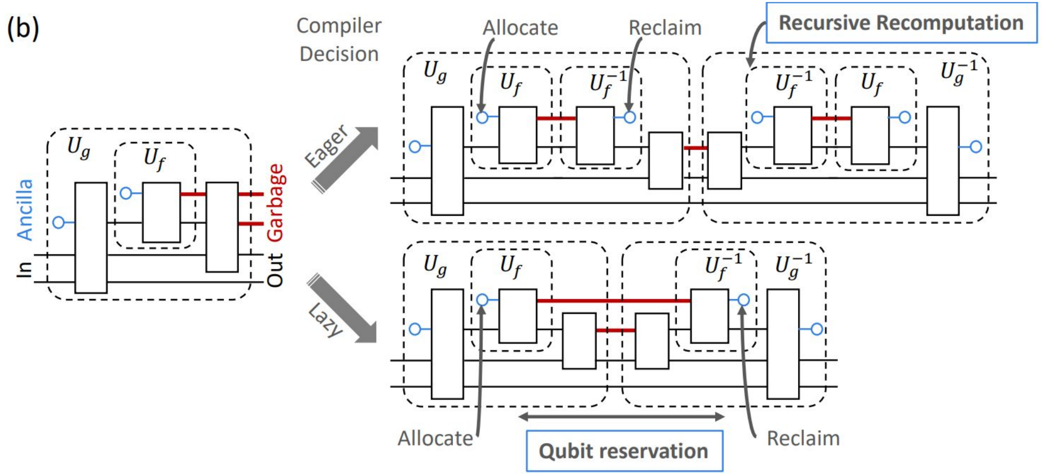


Key Ideas/Methodology

SQUARE Methodology

Qubit Reuse Strategies

- **Eager Strategy**
  - Reclaim after every function call
  - More ancilla available
  - **Recursive Recomputation** → exponential gate increase
- **Lazy Strategy**
  - Reclaim only at top of function
  - Avoids wasted recomputation
  - **Qubit Reservation ➔** less qubits to use



UC RIVERSIDE

SQUARE Methodology

SQUARE Heuristics

- **Locality Aware Allocation (LAA)**
  - Prioritizes qubits by location
  - Balances communication, serialization, expansion
- **Active Quantum Volume (AQV)**
  - A heuristic about how long the qubit stays alive
  - A set AQV actually improves performance
- **Cost Effective Reclamation (CER)**
  - Cost vs Benefit analysis
  - Compare C_1 & C_0

$$V_A = \sum_{q \in Q} \sum_{(t_i, t_f) \in T_q} (t_f - t_i)$$

$$C_1 = N_{active} \times G_{uncomp} \times S \times 2^{\ell}$$

$$C_0 = N_{anc} \times G_p \times S \times \sqrt{(N_{active} + N_{anc})/N_{active}}$$

UC RIVERSIDE

SQUARE Methodology

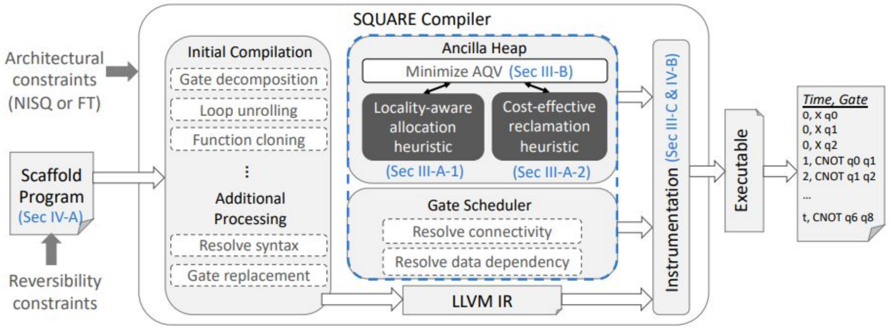
SQUARE Compilation Flow

- **Input Scaffold Program**

- **Replace `Allocate` and `Free`'s**
  - Classical heuristic functions
  - Establishes ancilla heap

- **Reclaim or Use new**
  - Determined based on AQV, LAA, CER

- **Output Classical Control Flow → dictates usage**



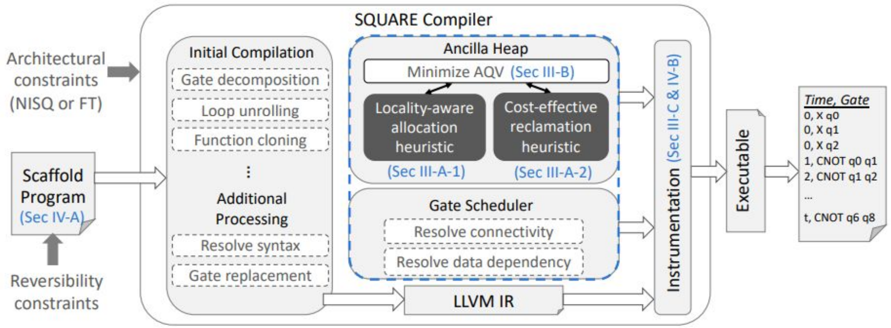

```
#include "qalloc.h"

void fun1(qbit* in, qbit* out) {
  qbit anc[1];
  Allocate(anc, 1);
  Compute {
    Toffoli(in[0], in[1], in[2]);
    CNOT(in[2], anc[0]);
    Toffoli(in[1], in[0], anc[0]);
  }
  Store {
    CNOT(anc[0], out[0]);
  }
  Uncompute{
    // Invoke Inverse() to populate
    // Or write out explicitly:
    Toffoli(in[1], in[0], anc[0]);
    CNOT(in[2], anc[0]);
    Toffoli(in[0], in[1], in[2]);
  }
  Free(anc, 1);
}
```

Implementation

Implementation

Instrument Driven Compilation

- **Modified version of Scaffold**
  - Compute Store Uncompute Syntax Block
  - Very similar to C/Verilog
- **Original Compilation Flow**
  - Separately optimize for parameters & each level
  - Produces conflicting results & expensive to perform
- **Instrumentation Driven Flow**
  - Better fit for dynamic allocation
  - Compilation time scalability



UC RIVERSIDE

Implementation

Allocation and Reclamation Details

- **Allocation Policy**
  - Concerned w/ qubit interaction
  - For NISQ Locality → # of swaps
  - FT Locality is complicated
    - Logical qbits arranged in braids
  - Allocate only when able
    - Marked as pending
    - Schedule nondependent, parallel computation & reclamation
- **Reclamation Policy**
  - Qubit savings, communication overhead uncomputation gate #
    - NISQ
      - Avg chain length per gate
    - FT
      - Avg braid crosses per gate

Algorithm 1 Allocate: *Locality-Aware Allocation*

Input: Number of qubits n
Output: Set of qubits $\mathcal{S}$

$\mathcal{I} \leftarrow$ LLVM::get_interact_qubits()
$\mathcal{S} \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** n **do**
  $q_1, score_1 \leftarrow$ closest_qubit_in_heap($\mathcal{I}$)
  $q_2, score_2 \leftarrow$ closest_qubit_new($\mathcal{I}$)
  if $score_1 \leq score_2$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q_1\}$
  else
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q_2\}$
  end if
end for

Algorithm 2 Free: *Cost-Effective Reclamation*

Input: Number of qubits n, Set of qubits $\mathcal{S}$

$C_1 \leftarrow$ cost of uncomputation
$C_0 \leftarrow$ cost of no uncomputation
if $C_1 \leq C_0$ **then**
  LLVM::gen_uncompute_block()
  heap_push($n, \mathcal{S}$)
else
  LLVM::rm_uncompute_block()
  LLVM::transfer_to_parent($n, \mathcal{S}$)
end if

UC RIVERSIDE

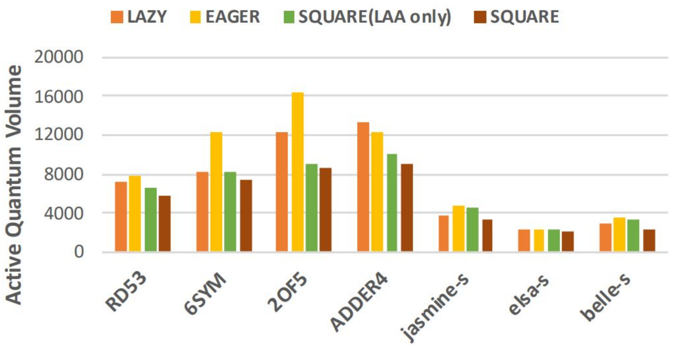
Results & Experimental Setup

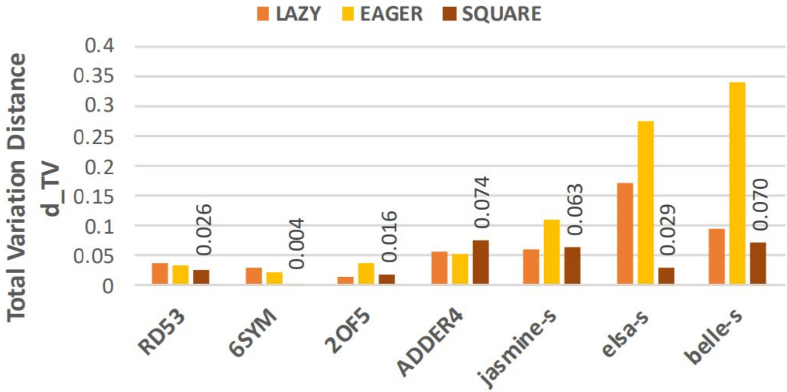
Results & Experimental Setup

NISQ Results

Benchmarks	Policy	# Gates[a]	# Qubits	Circuit Depth	# Swaps
RD53	Lazy	536	19	395	462
	Eager	1064	10	878	633
	SQUARE	932	11	635	370
6SYM	Lazy	648	19	456	654
	Eager	1293	11	1279	1247
	SQUARE	1078	12	731	520
2OF5	Lazy	708	18	723	759
	Eager	1410	8	2374	1728
	SQUARE	1176	10	952	385
ADDER4	Lazy	656	18	787	725
	Eager	1184	12	1139	748
	SQUARE	920	14	715	421
Jasmine-s	Lazy	275	16	232	73
	Eager	1226	5	1055	327
	SQUARE	510	8	427	128
Elsa-s	Lazy	163	15	787	725
	Eager	501	8	438	163
	SQUARE	254	13	223	85
Belle-s	Lazy	220	14	202	69
	Eager	712	6	574	113
	SQUARE	294	9	266	89

	# Qubits	ϵ_{single}	ϵ_{two}	T1 (μs)	T2 (μs)
IBM-Sup [3], [70]	20	$< 1\%$	$< 2\%$	55	60
IonQ-Trap [33]	79	$< 1\%$	$< 2\%$	$> 10^6$	$> 10^6$
Our Simulation	< 20	0.1%	1%	50	70

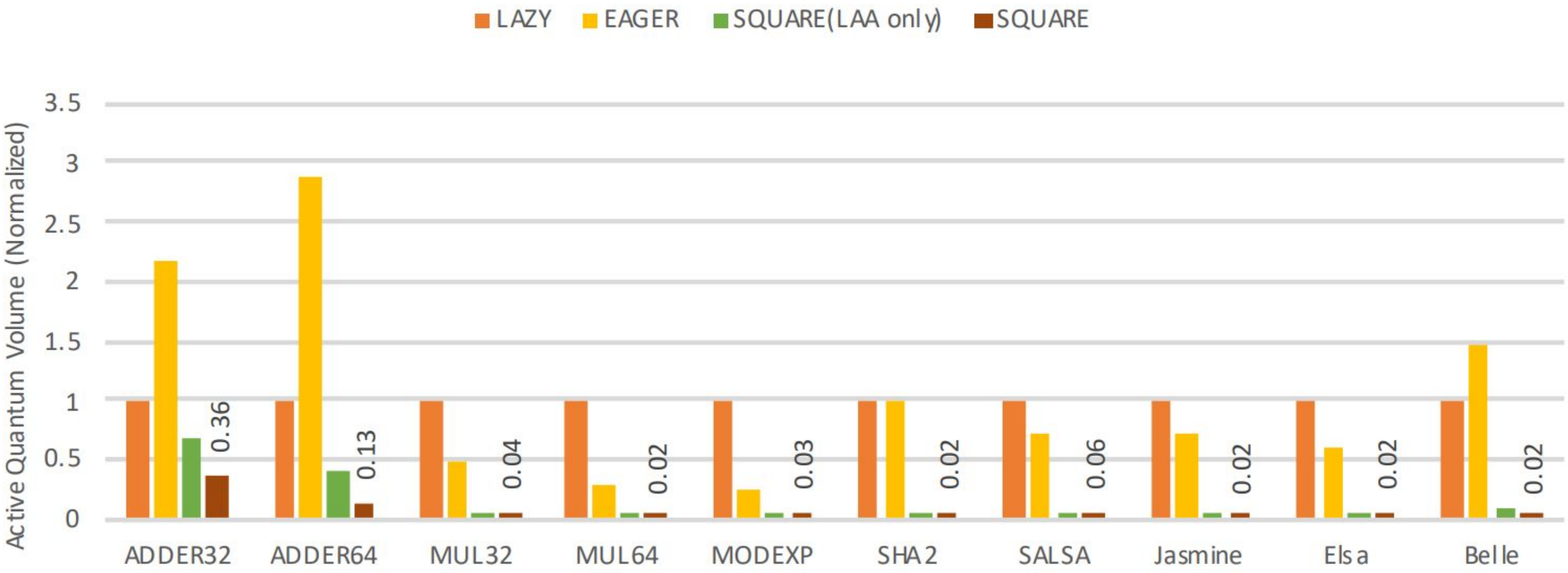




UC RIVERSIDE

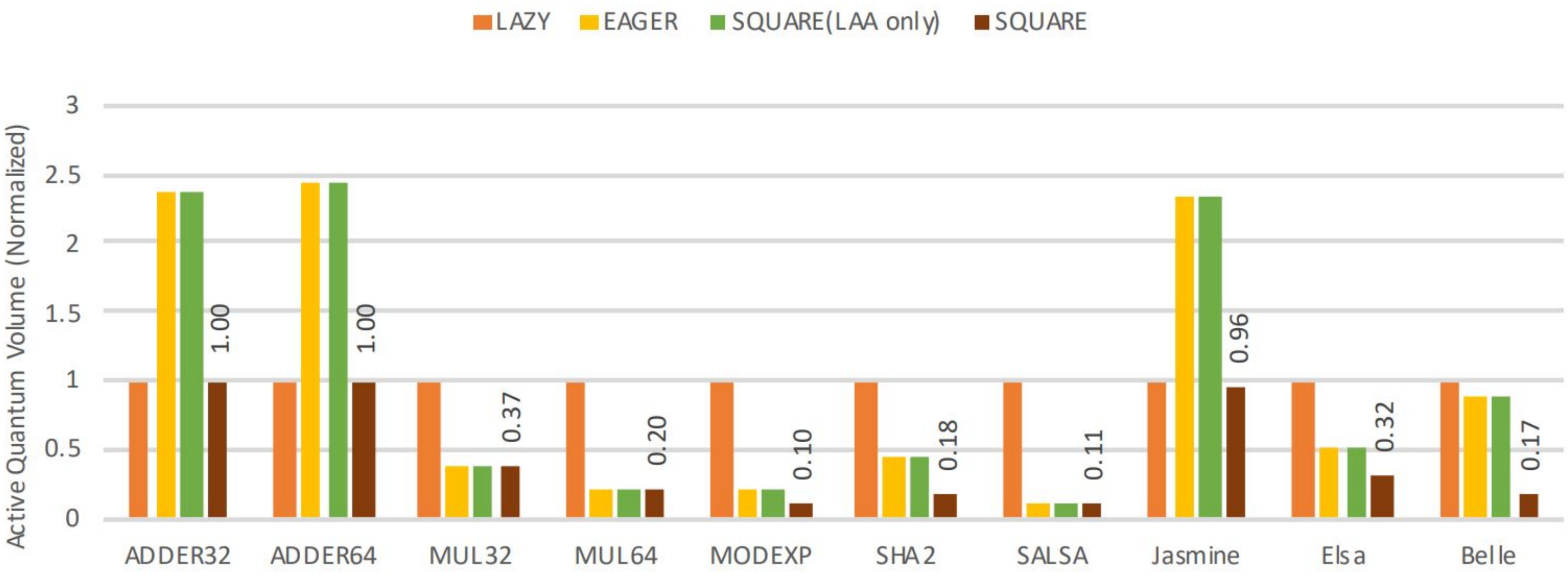
Results & Experimental Setup

Intermediate Scale QC Results



Results & Experimental Setup

FT Results



Conclusion

Conclusion

Acknowledgements

- **Diagrams, Research, Results**
  - SQUARE: Strategic Quantum Ancilla Reuse for Modular Quantum Programs via Cost-Effective Uncomputation[1]
  - Derivation of Reversibility for Hadamard & CNOT gate[2]
- **Quantum Circuits**
  - IBM Quantum Circuit Composer[3]
- **Presentation Theme**
  - UCR Communications[4]
- **Misc Images of Quantum Computers**
  -

UC RIVERSIDE

Thank you

Questions & Comments?

UC RIVERSIDE