Software for Advancing Quantum Computing

Yufei Ding

Assistant Professor
Department of Computer Science
University of California, Santa Barbara

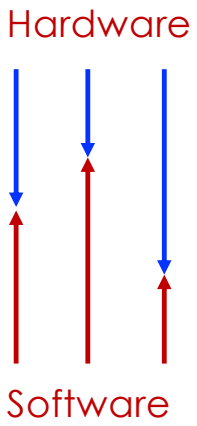
Software vs. Hardware

- How to divide the job between software and hardware optimizations?
  - Example: practical application, say Shor's algorithm.
  - Example: QEC with 2 logic qubits with gate error rate at the level of $10^{-6} \sim 10^{-9}$.

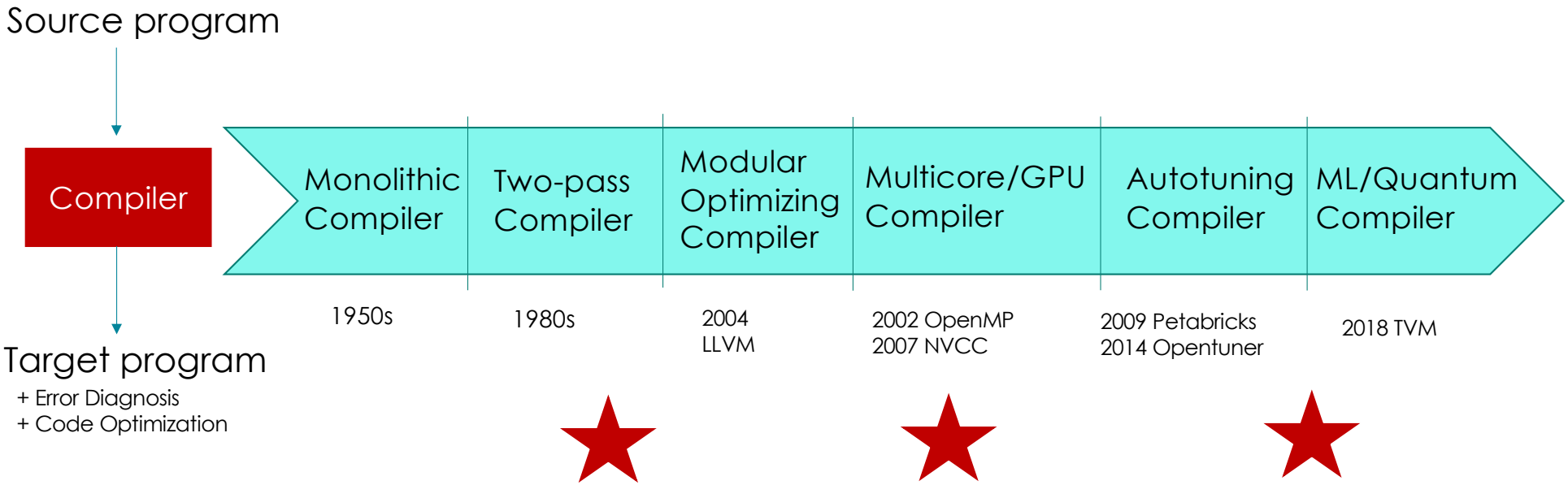
- Be realistic.
  - More powerful **hardware** makes **software** easier.
  - More powerful **software** makes **hardware** easier.

- To find the **boundary** for software and hardware optimization is like an **art** and the boundary will **change over time due to:**
  - Overall Cost budget
  - State-of-the-art Hardware technology.
  - Specific Application.
  - Our understanding of the physics and system designs.

- A good software will
  - help resolve hardware constraints.
  - quantitively tell us how good the hardware need to be.



Lessons Learned from Classical World: Classical Compiler



Our observation: From **Monolithic** to **Modular** to **Architecture(Input)-aware** to **Domain-Specific.**

- Modular with a good set of abstractions (intermediate representation) allows easy adaption.
- Get used to specialization towards whatever is needed.

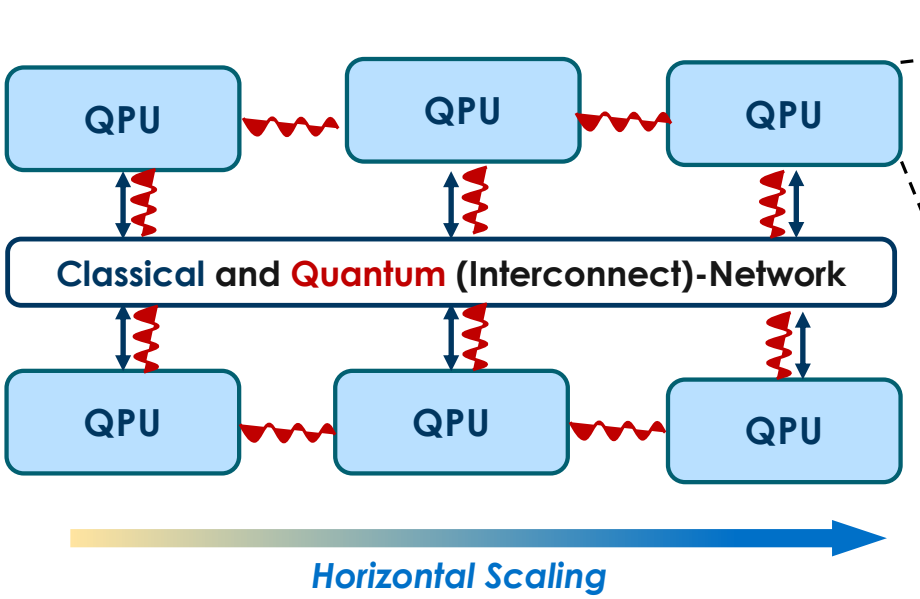
My View of Three Important Set of Software Toolchains

My focus is more at top level:

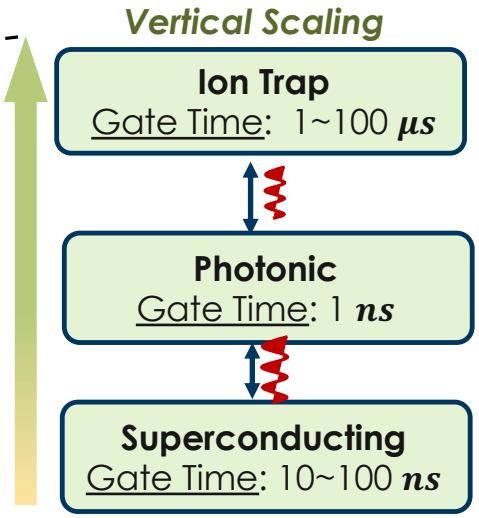
- Toward **Hardware Scaling Up.**
- Toward **Quantum Error Correction.**

Other interesting toolchains for optimization at lower level:
e.g., EDA for qubit engineering.

Quantum Scaling-Up Strategies



Chiplet Quantum Computing
Distributed Quantum Computing



Physics

- **Quantum processor unit**
- **Quantum interconnect**
- **Quantum memory**
- **Quantum register**

CS

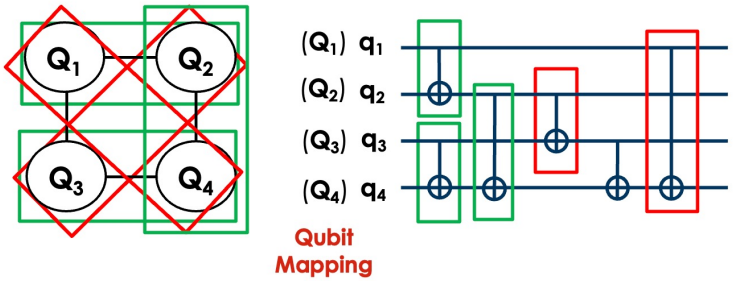
Software ecosystem would be different for different hardware architectures.

A Synthesis Framework for Stitching Surface Code with Superconducting Quantum Devices

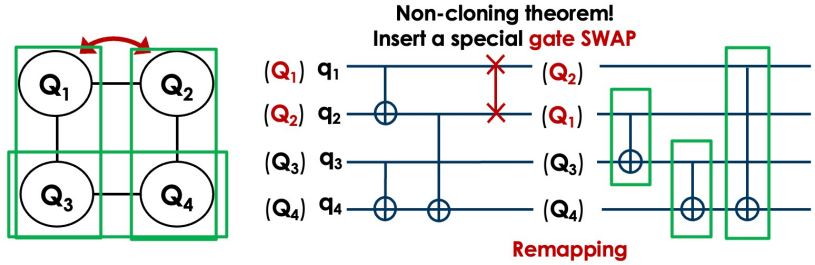
Anbang Wu, Gushu LI, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie.

[ASPLOS'20]

Key Novelty:
We can use a compiler to efficiently mitigate the constraints of limiting sparse 2-qubit connection on hardware.



Software Infrastructure:
1. Initial mapping
2. efficient routing



Paulihedral: A Generalized Block-Wise Compiler Optimization Framework For Quantum Simulation Kernels

Gushu Li, Anbang Wu, Yunong Shii, Ali Javadi-Abhari, Yufei Ding, Yuan Xie.

[ASPLOS'22]

❖ **Intermediate Representation (IR)**:
- Between **source code** and **machine code**
- Right level of abstraction for efficient analysis and optimization.

```
x = b;
for(i = 1; i ≤ N; i++){
    for(j = 1; j ≤ M; j++){
        x = x + a[i];}}
```
A nested loop

$\longleftrightarrow$ $b + \sum_i^{1,N} \sum_j^{1,M} a[i] \Rightarrow x$

IR for the loop

Key here is a **higher-level, formula-like IR** to concisely encode (for/while) loops.

Key Novelty:
What is a Good Quantum Intermediate representation? Principles for Guidance

Abstraction: Multi-Qubit Operation, i.e., Multi-Qubit Gate.

❖ Expressiveness: Efficiently + Universal

❖ Beneficial

❖ Easy Lowering

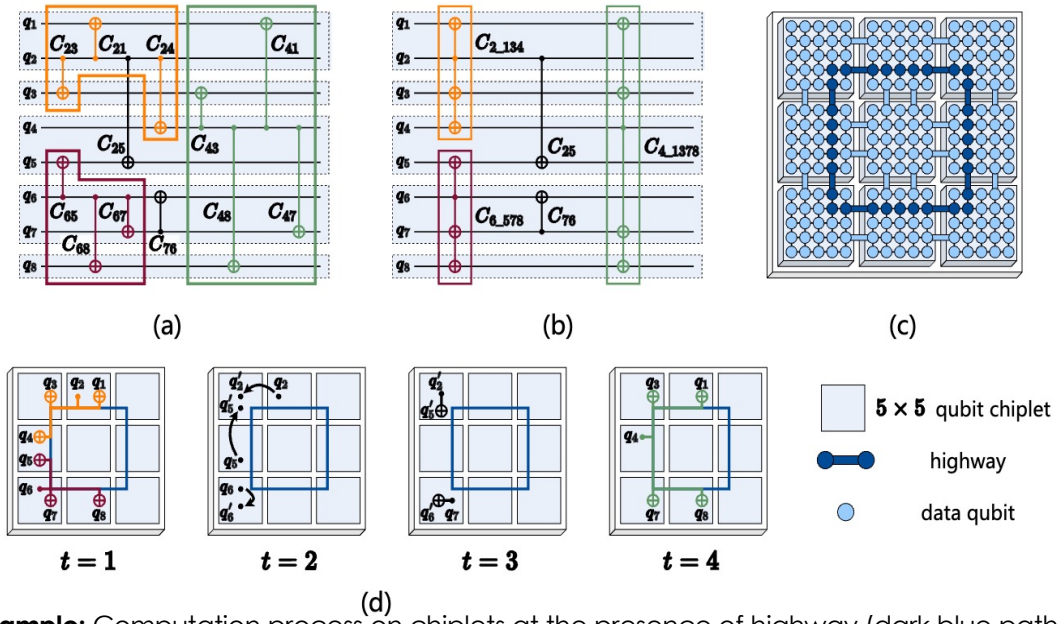
Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices

https://arxiv.org/abs/2305.05149

Gushu Li, Yufei Ding, Yuan Xie

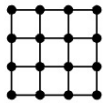
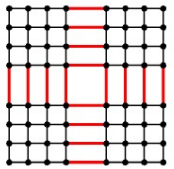
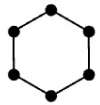
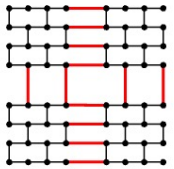
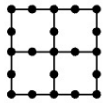
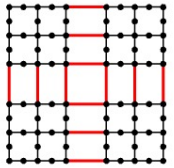
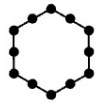
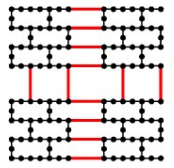
Key Novelty:

We propose to build a special compiler for optimizing quantum computing on large-scale chiplet quantum hardware. A new **highway** model is propped to boost computation by enabling more concurrency in gate execution regardless of the distances among the involved qubits.



Example: Computation process on chiplets at the presence of highway (dark blue paths in (c)). Commutable gates in circuit (a) are aggregated to multi-target control gates (b) and executed simultaneously on the highway (d).

Preliminary Results:

Name	Building blocks	Single chiplet	Connected chiplets
Square			
Hexagon			
Heavy Square			
Heavy Hexagon			

On average (geomean), the circuit depth is reduced by **69.05%**, and the effective number of CNOTs is reduced by **27.25%**.

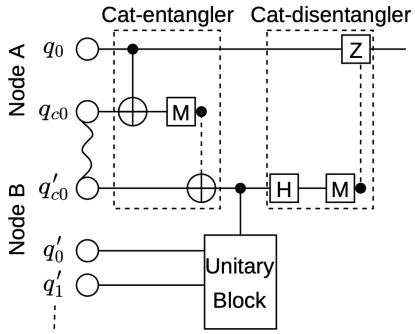
AutoComm: A Framework for Enabling Efficient Communication in Distributed Quantum Programs

Anbang Wu, Hezi Zhang, Gushu Li, Alireza Shabani, Yuan Xie, Yufei Ding

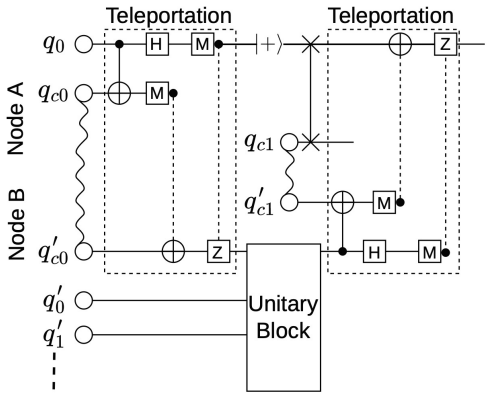
[MICRO'22]

Key Novelty:

First Compiler that enables optimization for burst communication.



(1) The optimized implementation of one controlled unitary block by one call of Cat-Comm



(2) The optimized implementation of one Unitary block by two calls of TP-Comm

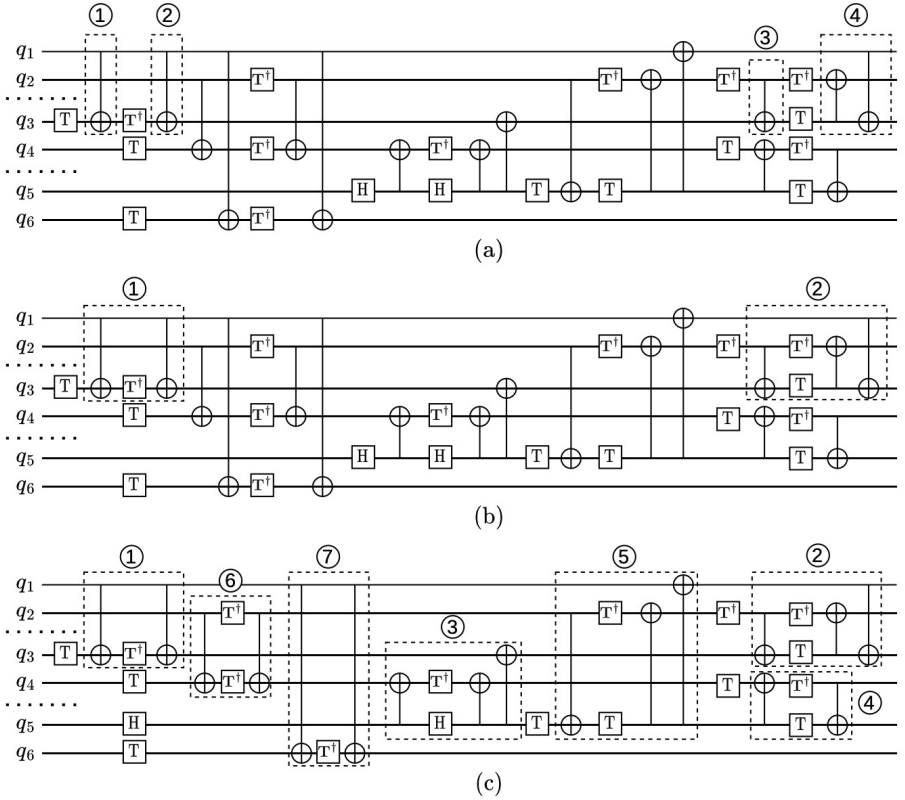
Insights: Burst communications enable more remote gates and thus reduce the ERP pair consumption, but they are often hidden in the original quantum programs.

Results:

Operation	Variable Name	Latency
Single-qubit gates	t_{1q}	$\sim$ 0.1 CX
CX and CZ gates	t_{2q}	1 CX
Measure	t_{ms}	5 CX
EPR preparation	t_{ep}	$\sim$ 12 CX
One-bit classical comm	t_{cb}	$\sim$ 1 CX

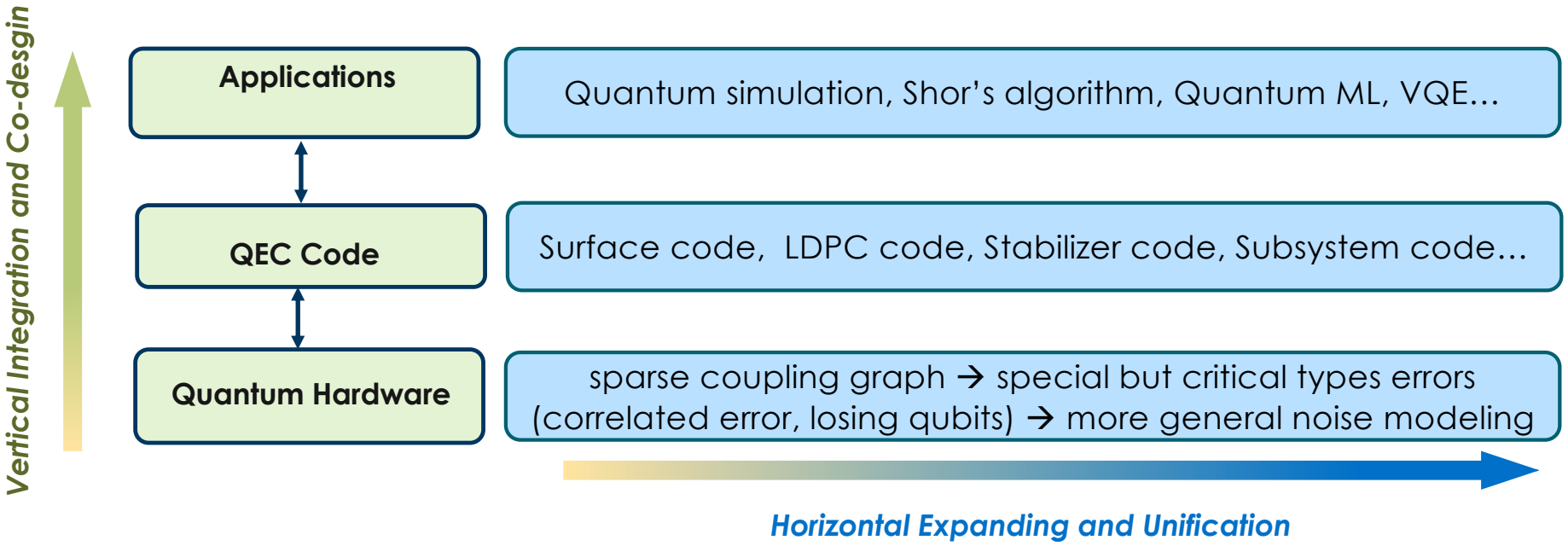
Compared to state-of-the-art DQC compilers, experimental results show that our proposed AutoComm can reduce the communication resource consumption and the program latency by **72.9%** and **69.2%** on average, respectively.

Software Infrastructure:



Example: Our compiler would enable communication aggregation with three key modules. (a) Identifying potential burst communication. (b) Linear merge. (c) Iterative refinement.

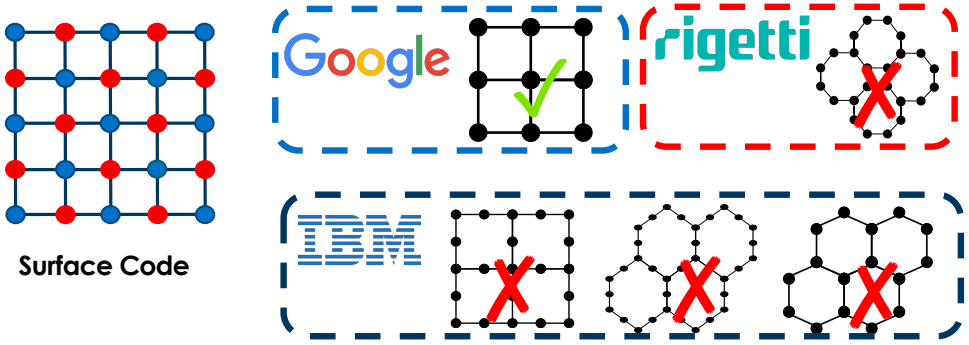
QEC is a Software Layer



A Synthesis Framework for Stitching Surface Code with Superconducting Quantum Devices

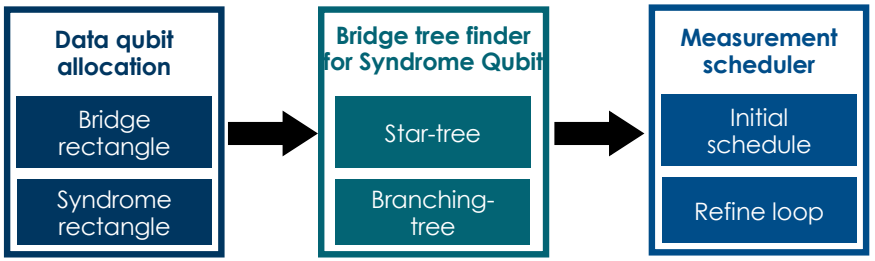
Anbang Wu, Gushu LI, Hezi Zhang, Gian Giacomo Guerreschi, Yufei Ding, and Yuan Xie. **[ISCA'22]**

Key Novelty:
We can use a compiler to mitigate such a structural-level mismatch.



Software Infrastructure:

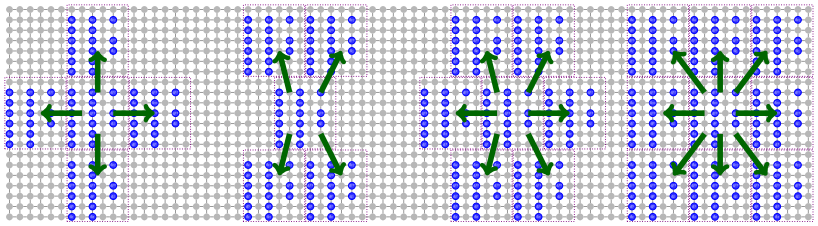
We can systematically solve the mismatch: 1) Good abstraction; 2) Knowledge of beneficial and legal transformations; 3) An efficient search scheme.



Architecture and Compiler Support for Fault-tolerant Quantum Computing based on Code Switching

Anbang Wu, Keyi Yin, Andrew Cross, Ang Li, and Yufei Ding **[In submission]**

Key Novelty:
We take (Steane code + RM code) as an example, to show the power of full-stack integration/codesign, from QEC-hardware codesign to compiler-application codesign.



1. Mapping of two code on the same area to facilitate code conversion.
2. When to apply the conversion?

Examples of placing multiple logical qubits. Green arrows denote logical CX directions. (a) connectivity 4. (b) connectivity 4 rotated. (c) connectivity 6. (d) connectivity 8

Thank you!