ISCA '25

RUTGERS

NC STATE UNIVERSITY

Genesis: A Hybrid CV-DV Compiler for Hamiltonian Simulation

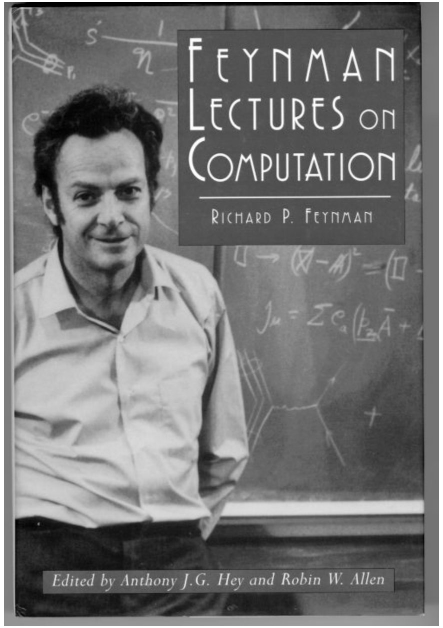
Zihan Chen[*,1], Jiakang Li[*,1], Minghao Guo[*,1]

Henry Chen[1], Zirui Li[1], Joel Bierman[2], Yipeng Huang[1], Huiyang Zhou[2], Yuan Liu[2], Eddy Z. Zhang[1]

[1]Rutgers University [2]North Carolina State University

[*]Denotes Equal Contribution

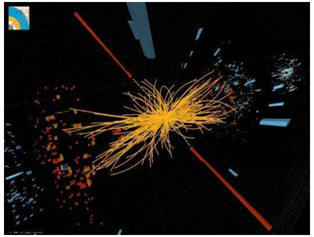
Introduction - Simulation of Nature

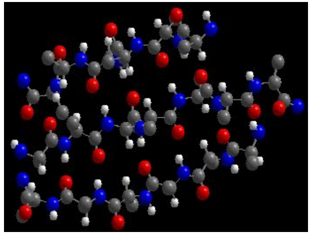


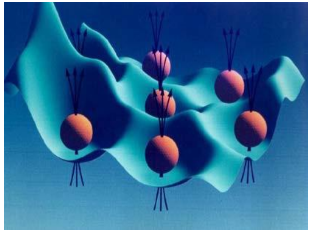
"Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem because it doesn't look so easy."

R. P. Feynman, 1981

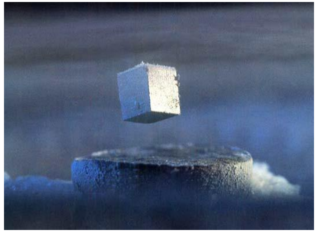
Introduction- Capability of Quantum Computer

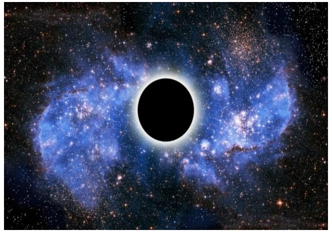

particle collision


molecular chemistry


entangled electrons

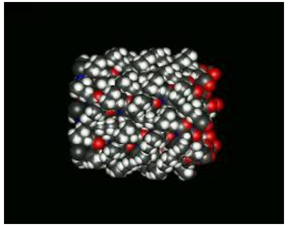
A quantum computer can simulate efficiently any physical process that occurs in Nature.
(Maybe. We don’t actually know for sure.) - John Preskill

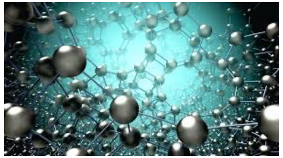

superconductor


black hole

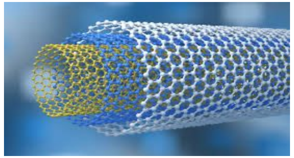

early universe

Hamiltonian Simulation

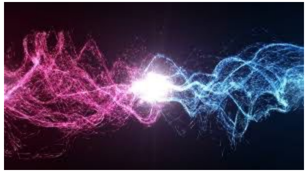




Quantum Chemistry



Material Science



High-Energy Physics

Hamiltonian Simulation

Molecular Biology

Drug Discovery

$$e^{-iHt}$$

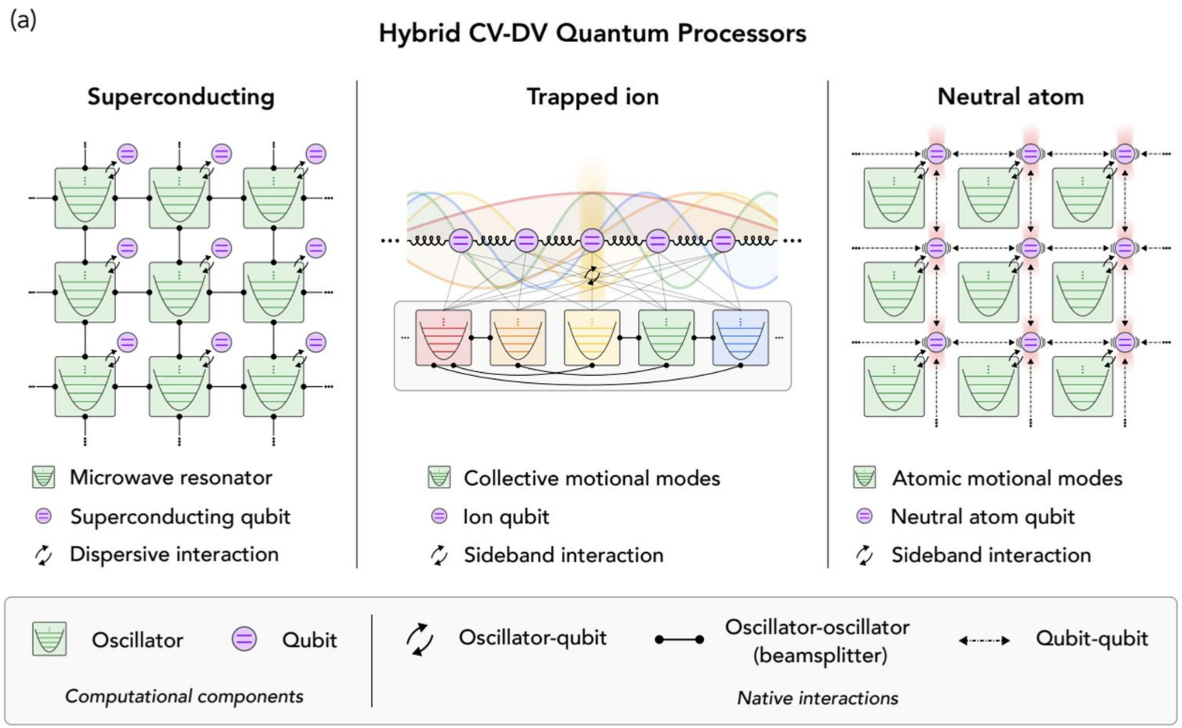
- Hamiltonian Simulation is a “killer” quantum computing application currently

Hybrid CVDV Background

- Most current quantum machines use qubits that are **discrete-variable (DV)** systems – which is quite fit to simulate the Fermions with Discrete states .
- In contrast, Bosons have continuous/infite states, and a **continuous-variable (CV)** quantum system (qumode) which has a spectrum of many possible states, is fit to simulate the Bosonic system. It can retain more robust quantum states and has the potential to achieve excellent quantum error correction.

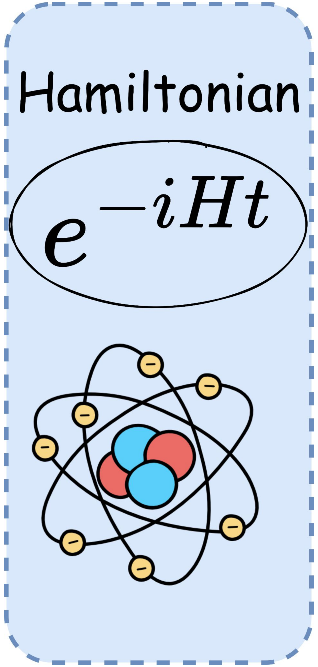
- **CV-only** hardware is challenging to have non-Gaussian resources.
- **DV-only** hardware needs truncation for simulating CV states, also it is difficult to simulate native bosonic operators.
- **Hybrid CV-DV** hardware takes the best of both system and is well-suited for the physical simulation with fermion-boson mixtures

Hybrid CVDV Background



Source: **“Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications.”** Liu et al. arXiv 2407.10381v2

Hybrid CVDV Background



- Hybrid CV-DV hardware takes the best of both system and is well-suited for the physical simulation with fermion-boson mixtures
- However, compiler and programming systems are largely undeveloped for hybrid CV-DV systems.
- Fermion-Boson mixtures interactions have not been thoroughly investigated, Genesis tries to bridge this gap and offers a complete end-to-end hamiltonian simulation compilation support!

Challenges and Motivation

1. Complex Cross-Domain Problem
   - Domain Specific Language (DSL) Hamiltonian Grammar and Multi-level Compilation
2. Qumode-centric Gate Synthesis
   - Rule-Based Recursive Template Matching
3. Multi-qubit Pauli-string Synthesis
   - Traveling Ancilla Qumode
4. Limited Connectivity Constraints
   - Hybrid CVDV Hardware Mapping and Routing

Hamiltonian Grammar DSL Representation

Hamiltonian Model Example:

$$H = -t\sum_{i,j,\sigma} c_{i\sigma}^{\dagger} c_{j\sigma} + U\sum_{i} b_i^{\dagger} b_i + g\sum_{i,\sigma} \hat{n}_{i\sigma}(b_i^{\dagger} + b_i)$$

Corresponding Example Hamiltonian Grammar Representation:

```
Const t = 1;
Const U = 1;
Const g = 1;

Range i = [0, 10, 1];
Range j = [0, 10, 1];
Range sigma = [0, 2, 1];

Result = - t *Sum_over(i, j, sigma){FC[i][sigma]* FA[j][sigma]}
        + U *Sum_over(i){BC[i]* BA[i]}
        + g * Sum_over(i, sigma){TensorProd(FN[i][sigma], BC[i] + BA[i])};
```

RUTGERS NC STATE UNIVERSITY

CVDVQASM and Multi-level Compilation

1. Hamiltonian Formula

$$H = -t\sum_{i,j,\sigma} c^{\dagger}_{i\sigma}c_{j\sigma} + U\sum_{i} b^{\dagger}_{i}b_{i} + g\sum_{i,\sigma}\hat{n}_{i\sigma}(b^{\dagger}_{i} + b_{i})$$

2. Hamiltonian Grammar

```
- t *Sum_over(i, j, sigma){FC[i][sigma]* FA[j][sigma]}
+ U *Sum_over(i){BC[i]* BA[i]}
+ g * Sum_over(i, sigma){TensorProd(FN[i][sigma], BC[i] + BA[i])};
```

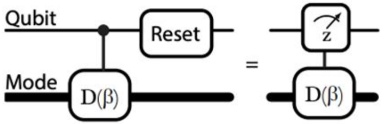
3. Intermediate Representation

```
pauli(0.392699075j): IXIZIYII;
pauli(0.392699075j): IIYIXIII;
bosonic: exp(prod((-1j),dagger(b(0)),b(0)));
bosonic: exp(prod((-1j),dagger(b(1)),b(1)));
hybrid: exp(prod((-0.78539815j),sigma(0, 0),sum(dagger(b(0)),b(0))));
hybrid: exp(prod((0.78539815j),sigma(3, 0),sum(dagger(b(0)),b(0))));
```

4. Logical CVDVQASM file

```
// Pauli String with Parameter
pauli(pi/4) YIYZXXIIIIIIII;
pauli(pi/4) XZYZXYIIIIIIII;
...
// Phase Space Rotation Gate
R(pi/4) qm[1];
R(pi/4) qm[2];
...
// Control Displacement Gate
CD(pi/4) q[2], qm[1];
CD(pi/4) q[3], qm[1];
...
// Displacement Gate
D(pi/4) qm[2];
D(pi/4) qm[2];
...
```

5. Physical CVDVQASM file



Direct Qumode-centric Gate Synthesis

Type	Gate Name	Definition
Qubit	x, y Rotation	$r_\varphi(\theta) = \exp\left[-i\frac{\theta}{2}(\cos\varphi\sigma_x + \sin\varphi\sigma_y)\right]$
	z Rotation	$r_z(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_z\right)$
Qumode	Phase-Space Rotation	$\mathrm{R}(\theta) = \exp\left[-i\theta a^\dagger a\right]$
	Displacement	$\mathrm{D}(\alpha) = \exp\left[\left(\alpha a^\dagger - \alpha^* a\right)\right]$
	Beam-Splitter	$\mathrm{BS}(\theta,\varphi) = \exp\left[-i\frac{\theta}{2}\left(e^{i\varphi}a^\dagger b + e^{-i\varphi}ab^\dagger\right)\right]$
Hybrid	Conditional Phase-Space Rotation	$\mathrm{CR}(\theta) = \exp\left[-i\frac{\theta}{2}\sigma_z a^\dagger a\right]$
	Conditional Parity	$\mathrm{CP} = \exp\left[-i\frac{\pi}{2}\sigma_z a^\dagger a\right]$
	Conditional Displacement	$\mathrm{CD}(\alpha) = \exp\left[\sigma_z\left(\alpha a^\dagger - \alpha^* a\right)\right]$
	Conditional Beam-Splitter	$\mathrm{CBS}(\theta,\varphi) = \exp\left[-i\frac{\theta}{2}\sigma_z\left(e^{i\varphi}a^\dagger b + e^{-i\varphi}ab^\dagger\right)\right]$
	Rabi Interaction	$\mathrm{RB}(\theta) = \exp\left[-i\sigma_x\left(\theta a^\dagger - \theta^* a\right)\right]$

Quantum Algorithms Define Compiler Rules: Operator Decomposition in CV-DV Systems

- Trotterization(Trotter-Suzuki formula)

$$e^{(M+N)t} \approx \left(e^{Mt'} e^{Nt'}\right)^n$$

- BCH(Baker Campbell Hausdorff formula)

$$e^{[M,N]t^2} \approx e^{Mt} e^{Nt} e^{-Mt} e^{-Nt}.$$

- Block Encoding

$$O = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$O = |0\rangle\langle 0| \otimes A + |0\rangle\langle 1| \otimes B + |1\rangle\langle 0| \otimes C + |1\rangle\langle 1| \otimes D.$$

- Commutator $\sigma_z[A, B]$ and anticommutator $\sigma_z\{A, B\}$ implementation in CVDV architecture

Source: **"Leveraging Hamiltonian Simulation Techniques to Compile Operations on Bosonic Devices."** Kang, Christopher, et al. arXiv:2303.15542

Rule-Based Recursive Template Matching

Rules	Operator Template	Conditions	Decomposition Output	Reference	Precision
1	$\exp(Mt + Nt) \approx \text{Trotter}(Mt, Nt)$		$(\exp(Mt/k)\exp(Nt/k))^k$	Trotterization	Approx
2	$\exp([Mt, Nt]) \approx \text{BCH}(Mt, Nt)$		$\exp(Mt)\exp(Nt)\exp(-Mt)\exp(-Nt)$	BCH	Approx
3	$\exp(t^2[M, N])$	M, N Hermitian	$\exp([it\sigma_i N, it\sigma_i M])$	[20]	Exact
4	$\exp(-it^2\sigma_i\{M, N\})$	M, N Hermitian	$\exp([it\sigma_j M, it\sigma_k N])$	[20]	Exact
5	$\exp(-it^2\sigma_z[M, N])$		$\exp([itN, it\sigma_z M])$	This paper	Exact
6	$\exp(t^2\sigma_z((MN - (MN)^\dagger)))$	$[M, N] = 0$	$\exp([X \cdot it\mathcal{B}_N \cdot X, it\mathcal{B}_M])$	[20]	Exact
7	$\exp(it^2\sigma_z((MN + (MN)^\dagger)))$	$[M, N] = 0$	$\exp([S \cdot it\mathcal{B}_M \cdot S^\dagger, X \cdot it\mathcal{B}_N \cdot X])$	[20]	Exact
8	$\exp\left(-2it\begin{pmatrix} MN & 0 \\ 0 & -MN \end{pmatrix}\right)$	M, N Hermitian	$\exp(-it\sigma_z[M, N] - it\sigma_z\{M, N\})$	This paper	Exact
9	$\exp\left(2it^2\begin{pmatrix} MN & 0 \\ 0 & -MN \end{pmatrix}\right)$	$[M, N] = 0$ $MN = (MN)^\dagger$	$\exp([S \cdot it\mathcal{B}_M \cdot S^\dagger, X \cdot it\mathcal{B}_N \cdot X])$	[20]	Exact
10	$\exp(2it\mathcal{B}_{MN})$	$[M, N] = 0$	$X \cdot \exp(t\sigma_y(MN - (MN)^\dagger) + it\sigma_x(MN + (MN)^\dagger)) \cdot X$	[20]	Exact
11	$\exp\left(it\begin{pmatrix} 2MN & 0 \\ 0 & -NM - (NM)^\dagger \end{pmatrix}\right)$	$MN = (MN)^\dagger$	$\exp([S \cdot it\mathcal{B}_M \cdot S^\dagger, X \cdot it\mathcal{B}_N \cdot X])$	[20]	Exact
12	$\mathcal{B}_a = \exp\left(2i\alpha\begin{pmatrix} 0 & a \\ a^\dagger & 0 \end{pmatrix}\right)$	$\alpha = \alpha^*$	$\exp(i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger + a)) \otimes \sigma_y)\exp(-i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger + a)) \otimes \sigma_x)$	[20]	Approx
13	$\mathcal{B}_{a^\dagger} = \exp\left(2i\alpha\begin{pmatrix} 0 & a^\dagger \\ a & 0 \end{pmatrix}\right)$	$\alpha = \alpha^*$	$\exp(i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger + a)) \otimes \sigma_y)\exp(-i(\pi/2)a^\dagger a)\exp(-i(\alpha(a^\dagger + a)) \otimes \sigma_x)$	This paper	Approx
14	$e^{(P_1P_2\cdots P_n)(\alpha a_k^\dagger - \alpha^* a_k)}$		Multi-qubit-controlled displacement: Right hand side (RHS) of Equation (11) first line	[28]	Exact
15	$e^{2i\alpha^2 P_1P_2\cdots P_n}$		Multi-Pauli Exponential: Right hand side (RHS) of Equation (9) first line	This Paper	Exact
16	All Native Gates RHS in Table 2		All Native Gates Left Hand Side (LHS) Table 2	[28]	Exact

[20] **"Leveraging Hamiltonian Simulation Techniques to Compile Operations on Bosonic Devices."** Kang, Christopher, et al. arXiv:2303.15542

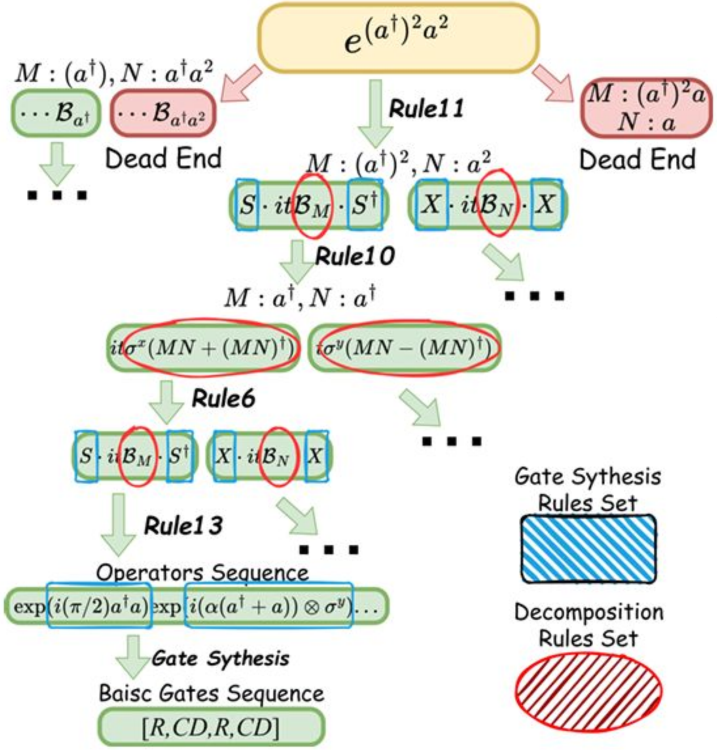
Rule-Based Recursive Template Matching

Basic Gates Set

Type	Gate Name	Definition
Qubit	x, y Rotation	$r_\varphi(\theta) = \exp\left[-i\frac{\theta}{2}(\cos\varphi\sigma_x + \sin\varphi\sigma_y)\right]$
	z Rotation	$r_z(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_z\right)$
Qumode	Phase-Space Rotation	$\mathrm{R}(\theta) = \exp\left[-i\theta a^\dagger a\right]$
	Displacement	$\mathrm{D}(\alpha) = \exp\left[\left(\alpha a^\dagger - \alpha^* a\right)\right]$
	Beam-Splitter	$\mathrm{BS}(\theta,\varphi) = \exp\left[-i\frac{\theta}{2}\left(e^{i\varphi}a^\dagger b + e^{-i\varphi}ab^\dagger\right)\right]$
Hybrid	Conditional Phase-Space Rotation	$\mathrm{CR}(\theta) = \exp\left[-i\frac{\theta}{2}\sigma_z a^\dagger a\right]$
	Conditional Parity	$\mathrm{CP} = \exp\left[-i\frac{\pi}{2}\sigma_z a^\dagger a\right]$
	Conditional Displacement	$\mathrm{CD}(\alpha) = \exp\left[\sigma_z\left(\alpha a^\dagger - \alpha^* a\right)\right]$
	Conditional Beam-Splitter	$\mathrm{CBS}(\theta,\varphi) = \exp\left[-i\frac{\theta}{2}\sigma_z\left(e^{i\varphi}a^\dagger b + e^{-i\varphi}ab^\dagger\right)\right]$
	Rabi Interaction	$\mathrm{RB}(\theta) = \exp\left[-i\sigma_x\left(\theta a^\dagger - \theta^* a\right)\right]$

Decomposition Rules Set

Rules	Operator Template	Conditions	Decomposition Output	Reference	Precision
1	$\exp(Mt+Nt) \approx \mathrm{Trotter}(Mt,Nt)$		$(\exp(Mt/k)\exp(Nt/k))^k$	Trotterization	Approx
2	$\exp([Mt,Nt]) \approx \mathrm{BCH}(Mt,Nt)$		$\exp(Mt)\exp(Nt)\exp(-Mt)\exp(-Nt)$	BCH	Approx
3	$\exp(t^2[M,N])$	M,N Hermitian	$\exp([it\sigma_i N, it\sigma_i M])$	[20]	Exact
4	$\exp(-it^2\sigma_i\{M,N\})$	M,N Hermitian	$\exp([it\sigma_j M, it\sigma_k N])$	[20]	Exact
5	$\exp(-it^2\sigma_z[M,N])$		$\exp([(itN, it\sigma_z M])$	This paper	Exact
6	$\exp(t^2\sigma_z((MN-(MN)^\dagger)))$	$[M,N]=0$	$\exp([X\cdot it\mathcal{B}_N\cdot X, it\mathcal{B}_M])$	[20]	Exact
7	$\exp(it^2\sigma_z((MN+(MN)^\dagger)))$	$[M,N]=0$	$\exp([S\cdot it\mathcal{B}_M\cdot S^\dagger, X\cdot it\mathcal{B}_N\cdot X])$	[20]	Exact
8	$\exp\left(-2it\begin{pmatrix} MN & 0 \\ 0 & -MN\end{pmatrix}\right)$	M,N Hermitian	$\exp(-it\sigma_z[M,N] - it\sigma_z\{M,N\})$	This paper	Exact
9	$\exp\left(2it^2\begin{pmatrix} MN & 0 \\ 0 & -MN\end{pmatrix}\right)$	$[M,N]=0$ $MN=(MN)^\dagger$	$\exp([(S\cdot it\mathcal{B}_M\cdot S^\dagger, X\cdot it\mathcal{B}_N\cdot X])$	[20]	Exact
10	$\exp(2it\mathcal{B}_{MN})$	$[M,N]=0$	$X\cdot\exp(t\sigma_y(MN-(MN)^\dagger) + it\sigma_x(MN+(MN)^\dagger))\cdot X$	[20]	Exact
11	$\exp\left(it\begin{pmatrix} 2MN & 0 \\ 0 & -NM-(NM)^\dagger\end{pmatrix}\right)$	$MN=(MN)^\dagger$	$\exp([S\cdot it\mathcal{B}_M\cdot S^\dagger, X\cdot it\mathcal{B}_N\cdot X])$	[20]	Exact
12	$\mathcal{B}_a = \exp\left(2i\alpha\begin{pmatrix} 0 & a \\ a^\dagger & 0\end{pmatrix}\right)$	$\alpha=\alpha^*$	$\exp(i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger+a))\otimes\sigma_y)\exp(-i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger+a))\otimes\sigma_x)$	[20]	Approx
13	$\mathcal{B}_{a^\dagger} = \exp\left(2i\alpha\begin{pmatrix} 0 & a^\dagger \\ a & 0\end{pmatrix}\right)$	$\alpha=\alpha^*$	$\exp(i(\pi/2)a^\dagger a)\exp(i(\alpha(a^\dagger+a))\otimes\sigma_y)\exp(-i(\pi/2)a^\dagger a)\exp(-i(\alpha(a^\dagger+a))\otimes\sigma_x)$	This paper	Approx
14	$e^{(P_1P_2\cdots P_n)(\alpha a_k^\dagger - \alpha^* a_k)}$		Multi-qubit-controlled displacement: Right hand side (RHS) of Equation (11) first line	[28]	Exact
15	$e^{2i\alpha^2 P_1P_2\cdots P_n}$		Multi-Pauli Exponential: Right hand side (RHS) of Equation (9) first line	This Paper	Exact
16	All Native Gates RHS in Table 2		All Native Gates Left Hand Side (LHS) Table 2	[28]	Exact



$$e^{(a^\dagger)^2a^2}$$

Basic Gates Sequence

(Logical CVDVQASM Circuit)

Repeat the rewrite process until it produces only basis gates

Multi-qubit Pauli-string Synthesis

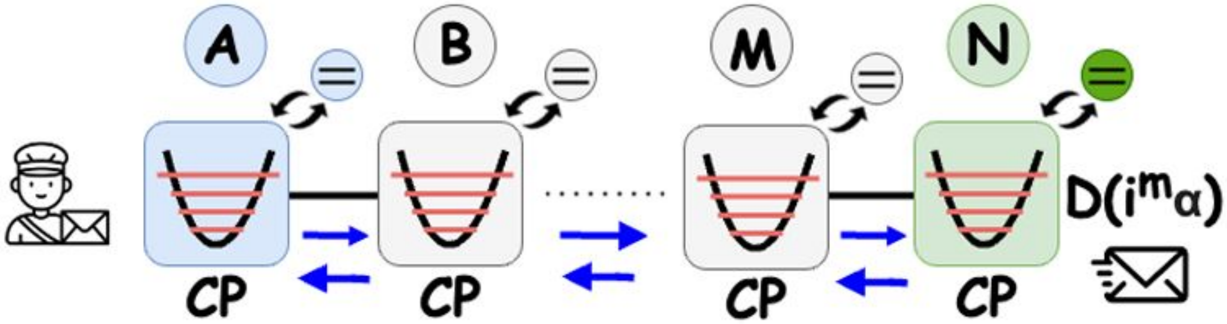
- Qubits are not directly connected with each other, we propose a scheme to synthesize an arbitrary multi-qubit Pauli-string on Hybrid CV-DV platforms.
- It is inspired by phase kickback in DV systems, where the phase of the control qubit is influenced by the operation on the target qubits.

$$U = D^k(i\alpha)\, CD^{(k,P_{1\cdots n})}(-\alpha)\, D^k(-i\alpha)\, CD^{(k,P_{1\cdots n})}(\alpha)$$
$$= e^{2i\alpha^2 P_1 P_2 \cdots P_n}$$

Multi-qubit Pauli-string Synthesis

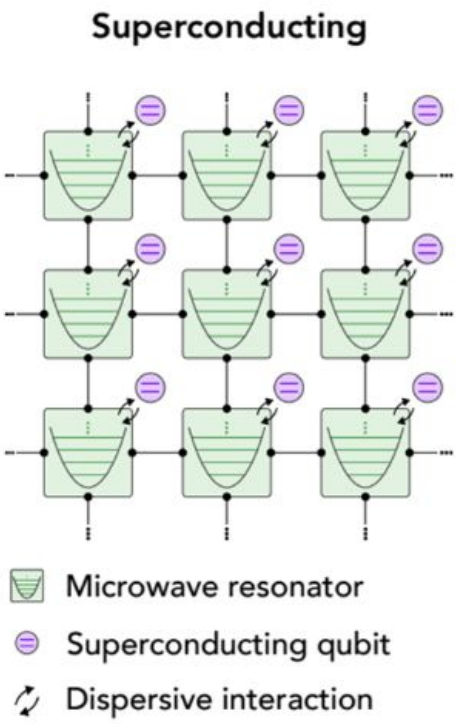
- Our final multi-Pauli exponential decomposition makes use of a multi-qubit controlled CD gate proposed by Liu et al., as below:

$$\mathrm{CD}^{(k,P_1P_2\cdots P_n)}(\alpha) = U_{\mathrm{seq}}^{\dagger} D(i^n\alpha) U_{\mathrm{seq}}$$
$$= e^{(P_1P_2\cdots P_n)(\alpha a_k^{\dagger} - \alpha^* a_k)}$$



Source: **"Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications."** Liu et al. arXiv 2407.10381v2

Limited Hardware Connectivity



- Qumode-qumode Mapping.

Interactions are limited to adjacent qumodes. For non-adjacent qumodes, qumode SWAP gates are used, with routing optimized to minimize such SWAPs.

- Qubit-qumode Mapping.

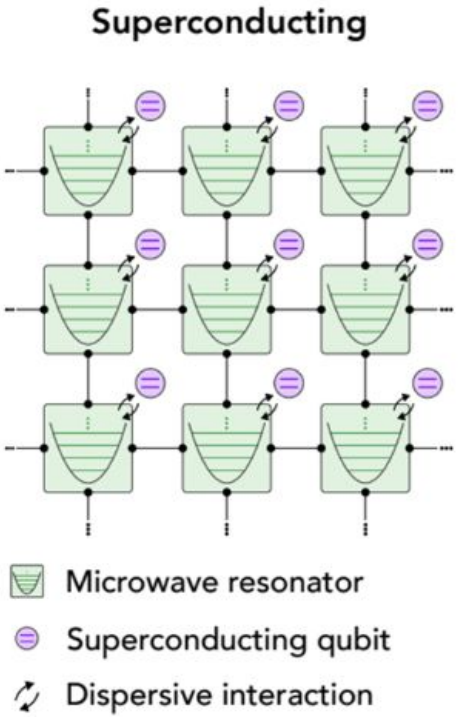
Each qumode interacts only with its associated qubit. For interactions with other qumodes, adjacency is established by moving qumodes, similar to qumode-qumode mapping.

- Qubit-qubit Mapping.

Qubits interact indirectly via an ancilla qumode, which is moved between qubits to mediate interactions and complete gate operations.

Source: **"Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications."** Liu et al. arXiv 2407.10381v2

Limited Hardware Connectivity



- ## Qumode-qumode Mapping.

Interactions are limited to adjacent qumodes. For non-adjacent qumodes, qumode SWAP gates are used, with routing optimized to minimize such SWAPs.

- ## Qubit-qumode Mapping.

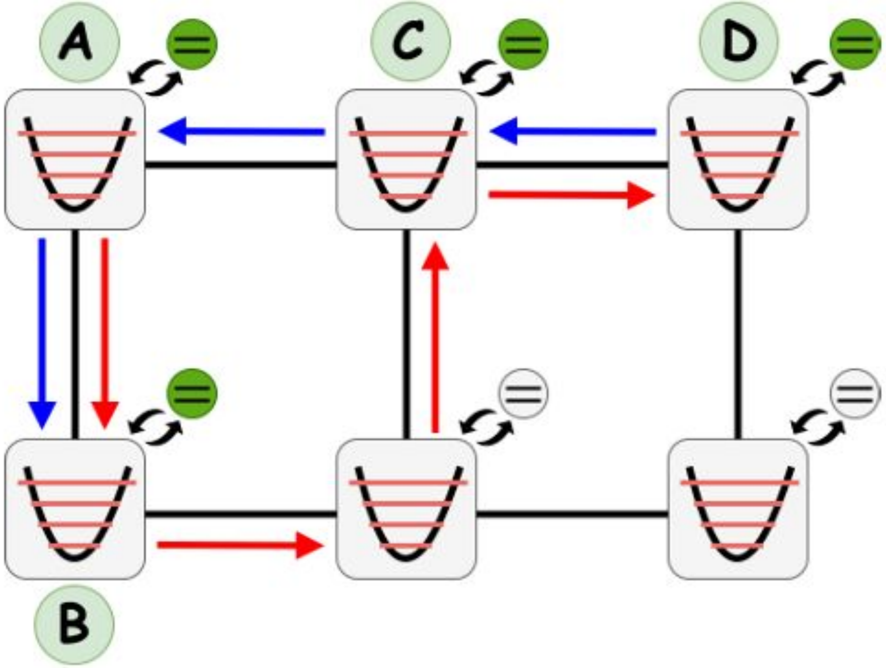
Each qumode interacts only with its associated qubit. For interactions with other qumodes, adjacency is established by moving qumodes, similar to qumode-qumode mapping.

Working Frontier: all unresolved gates whose dependence has been resolved

Using a Qiskit Sabre-like reward function to execute gate from the frontier and update it.

Source: **"Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications."** Liu et al. arXiv 2407.10381v2

Optimized Ancilla Qumode Routing



A → B → (A) → C → D: 4 BS gates

D → C → A → B: 3 BS gates

The Optimized Ancilla Qumode Routing Problem can be reformulated as a relaxed **Hamiltonian Path Problem**, similar to a modified Traveling Salesman Problem (TSP). Unlike the closed-path TSP, this problem allows revisiting vertices and does not require returning to the starting vertex.

RUTGERS NC STATE UNIVERSITY

Optimized Ancilla Qumode Routing

Qumode-SWAP

1 Beam-Splitter gate

(20x depth/duration)

Qubit-SWAP

(in CVDV System)

12 control displacement gates

12 Qumode-SWAPs

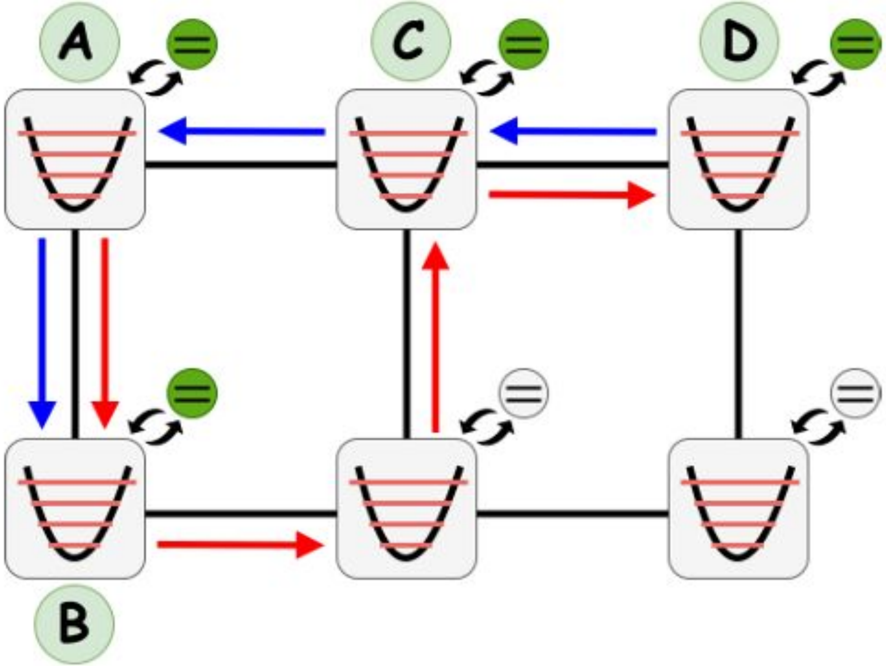
(480x depth/duration)

- Dynamic Qubit Floating

Relocation strategy, when a qubit-qubit pair distance in a specific multi-qubit exponential is too far, and this qubit-qubit pair appear often in the following multi-qubit exponential, we will try to relocate the qubit using Qubit-SWAP to cluster them.

Source: **"Hybrid Oscillator-Qubit Quantum Processors: Instruction Set Architectures, Abstract Machine Models, and Applications."** Liu et al. arXiv 2407.10381v2

Optimized Ancilla Qumode Routing



- Christofides Algorithm

  Baseline

- Threshold Accepting Algorithm

  3-7% better duration time, 4.8% in avg

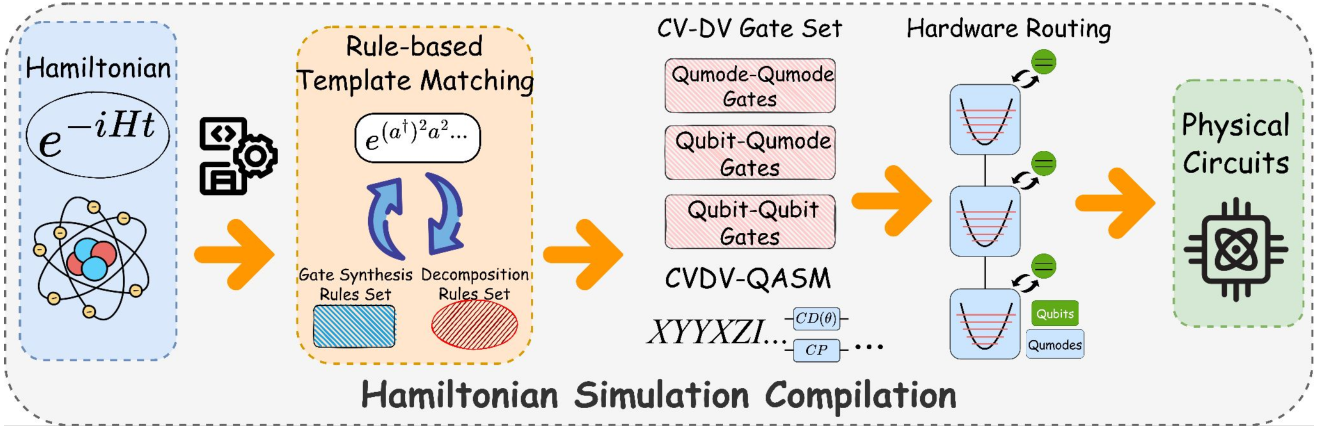
- Dynamic Qubit Floating

  6% worse duration time in avg

  4/20 better than baseline and 2/20 better than Threshold Accepting

End to End Implementation

1. **Hamiltonian Parsing**: Translates a Hamiltonian from mathematical form into a DSL-based representation.
2. **Intermediate Representation (IR)**: Converts the DSL into an IR consisting of Pauli strings and operator expressions(bosonic, hybrid).
3. **Pattern Matching and Gate Synthesis**: Matches fermionic and bosonic operator terms and synthesizes them into logical CV-DV circuits in CVDVQASM format.
4. **Physical Mapping**: Maps logical circuits and Pauli terms to hardware-compliant physical circuits, and outputs the final(physical) CVDVQASM program(s).



Software access: https://github.com/ruadapt/Genesis-CVDV-Compiler

GitHub QR Code:

End to End Implementation

- Evaluation 1. Multi Pauli-String Synthesis
  - 20 Qubit Hamiltonian such as LiH(4,12), BeH2(6,14) …
  - # Pauli Strings from 631 to 1884
  - JW and BK encoding
- Evaluation 2. General Hamiltonian Simulation Compilation
  - 6 Hamiltonian Models such as Hubbard-Holstein Model, Bose-Hubbard Model … At most 60 Qubits and 120 Qumodes

End to End Implementation-in the future

- Intermediate Tools 1. CVDV Mapping and Routing
  - Support more architecture(neutral atom)
  - Better relocate strategy when compile multi pauli-strings
- Intermediate Tools 2. Operator Pattern Matching
  - Flexible customize rules and multiple decomposition perspectives
  - Better compilation efficiency and robustness
  - Error analysis and unitary verification

Codes Available

Welcome to give it a try!

Software access:
https://github.com/ruadapt/Genesis-CVDV-Compiler

GitHub Link:

Thank You!