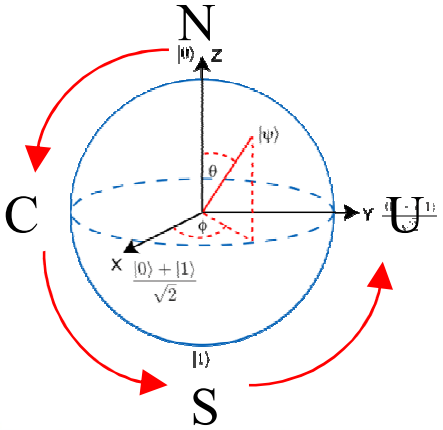
An Open-Source Framework to Integrate Quantum Computing with HPC

Frank Mueller

NC STATE UNIVERSITY
Department of Computer Science



Institute for Robust Quantum Simulation

STAQ

U.S. DEPARTMENT OF ENERGY

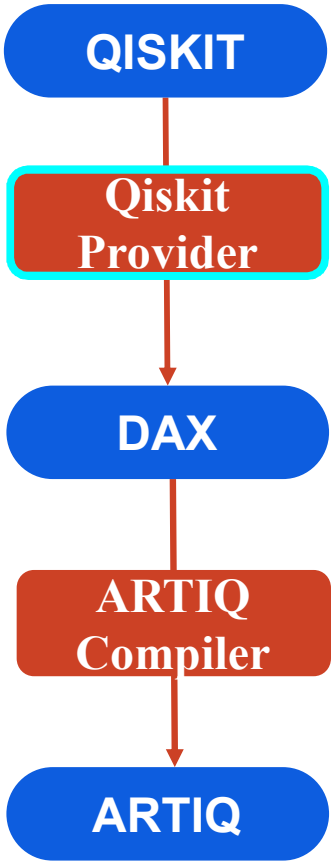
Contributions in Quantum (not just HPC)

- part of
  - NSF QLCI RQS (UMD lead)
  - NSF STAQ (Duke lead)
  - NSF PPoSS (NCSU lead)
  - DOE Bosonic (NCSU lead)

Institute for
Robust Quantum Simulation

STAQ

- Domain-specific quantum abstractions:
  SAT-problems, Physics, Chemistry [SC'22, QCE'24]
- Tensor networks for quantum simulation
- Circuit opts.: at gate+pulse levels
- Algo- & Noise-aware problem solving [SC'21]
- Semantic tracking during pgm translation
- HW/SW stack for ion traps
- Noise mitigation [QCE'20]
- FTQC development + verification



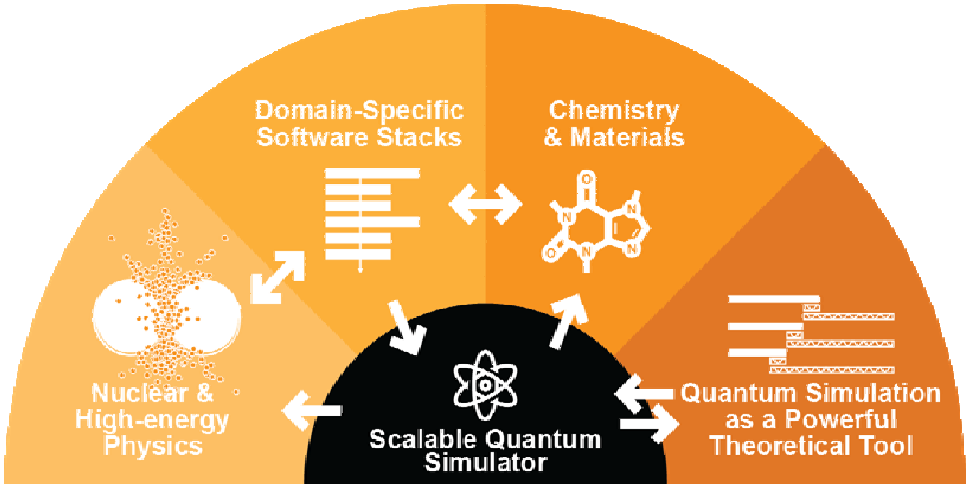
(1) Non-Expert Quantum Software Stacks

- Today's software stacks
  - Explicit gate-centric programming
  - Algorithms require intricate quantum understanding
- Can we find higher level abstractions?
  - ➢ Domain-specific languages (DSLs)
  - Example: molecular chemistry

```
#qiskit
qc.h(q[0])
qc.cx(q[0],q[1])
```

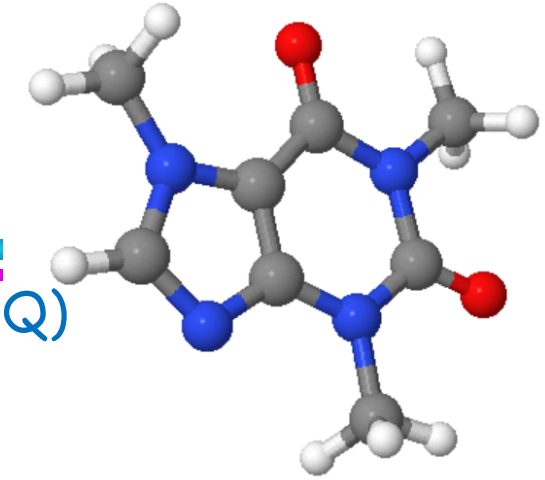
```
# Q#
H(q[0]);
    CNOT(q[0], q[1]);
    CNOT(q[0], q[2]);
```

```
#Quipper
q0 <- Hadamard q0
  q1 <- qnot q1 `controlled` q0
  q2 <- qnot q2 `controlled` q0
```



Institute for Robust Quantum Simulation

Quantum Molecular Chemistry



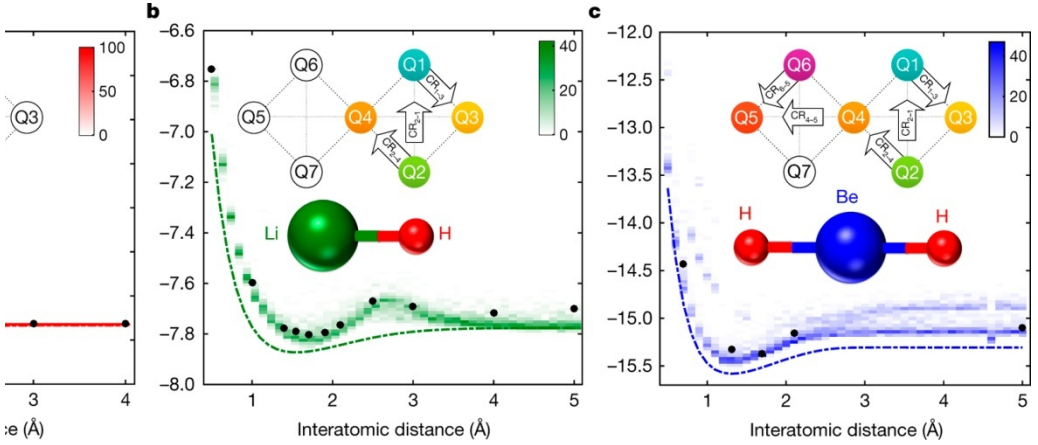
Caffeine molecule

- Near-term quantum app → tolerates noise (NISQ)
- Python libraries
  - IBM quiskit.nature (formerly aqua)
  - Microsoft.Quantum.Chemistry.Trotterization
- Quantum still exposed
  - Gates, Parameters
- ➢ Not suitable for non-quantum scientists

```
#quiskit.nature
#ground energy of mulecule
ansatz = TwoLocal(num_spin_orbitals, ['ry', 'rz'], 'cz')
algorithm = VQE(ansatz, optimizer=optimizer,
  quantum_instance=backend)
```

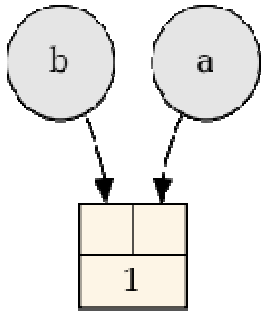
```
# phase estimation in Q#
GetEnergyRPE.simulate(
  JWEncodedData=encoded_data_caffeine,
  nBitsPrecision=10,
  trotterStepSize=0.2,
  trotterOrder=1 )
```

```
# VQE ground energy state
def VQE(initial_var_params, num_samples):
  opt_result = minimize( run_program,
    initial_var_params, args=(num_samples,),
    method="COBYLA", tol=0.000001,
    options={'disp': True, 'maxiter': 200,'rhobeg' : 0.05}
```

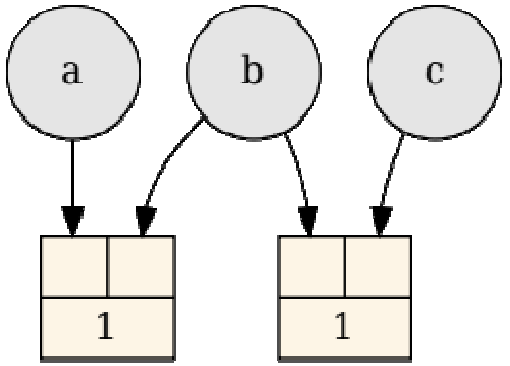


Our Answer: NchooseK [SC'22] w/ LANL

- Domain: constraint-programming system
  - NP-hard/NP-complete problem
- *Given*
  - *n* Boolean values
  - *k* of which constrained to be TRUE
- Notation: *nck*(*N*, *K*)
  - *N*: multiset of variables
  - *K*: set of allowable TRUE counts
  - Can combine multiple ones
- Express, xform: Hamiltonian → QUBO



***nck*({*a*, *b*}, {1})**
Exactly one of *a* or *b* must be TRUE



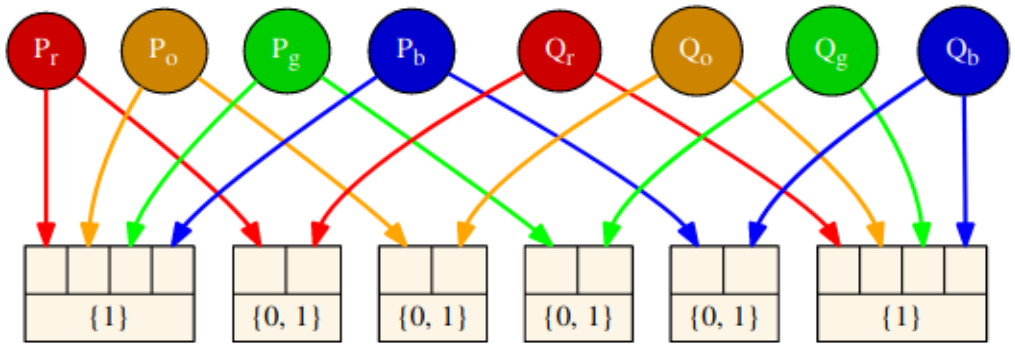
***nck*({*a*, *b*}, {1}) ∧ *nck*({*b*, *c*}, {1})**
***b*: same value in constraints**

$$\mathcal{H}_P = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i^z \sigma_j^z + \sum_{i=0}^{N-1} h_i \sigma_i^z$$

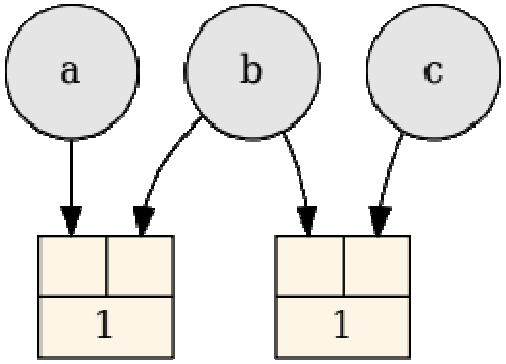
$$Obj(a_i, b_{ij}{:}q_i) = \Sigma_i a_i q_i + \Sigma_{ij} b_{ij} q_i q_j$$

NchooseK Translation

- Exact cover:
  a: *nck*({*v2*, *v3*, *v4*}, {1}) ∧ b:…
- Map coloring:
  nck({P_r, P_o, P_g, P_b}, {1}) ∧ …
  nck({P_r, Q_r}, {0, 1}) ∧ …



- Add soft constraints → min cover, min vertex cover, clique cover, k-SAT, max cut, …
- Expressed as DSL
  - Easily embedded in pgm




```
import nchoosek

env = nchoosek.Environment()
a = env.register_port('a')
b = env.register_port('b')
c = env.register_port('c')
env.nck([a, b], {1})
env.nck([b, c], {1})
result = env.solve()
print(result)
```

NchooseK Quantum Mapping

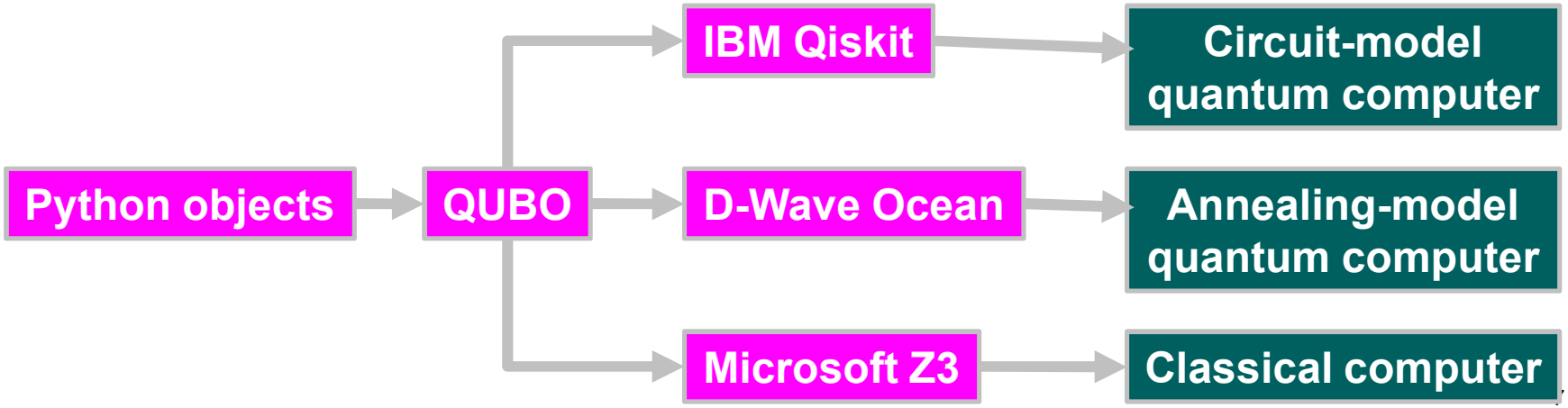
- NchooseK DSL →
- Hamiltonian →
- QUBO → QPU
  - Gate-based → QAOA
  - annealer → QUBO
  - Solver → constraints

```
import nchoosek

env = nchoosek.Environment()
a = env.register_port('a')
b = env.register_port('b')
c = env.register_port('c')
env.nck([a, b], {1})
env.nck([b, c], {1})
result = env.solve()
print(result)
```

$$\mathcal{H}_P = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i^z \sigma_j^z + \sum_{i=0}^{N-1} h_i \sigma_i^z$$

$$Obj(a_i, b_{ij}: q_i) = \Sigma_i a_i q_i + \Sigma_{ij} b_{ij} q_i q_j$$

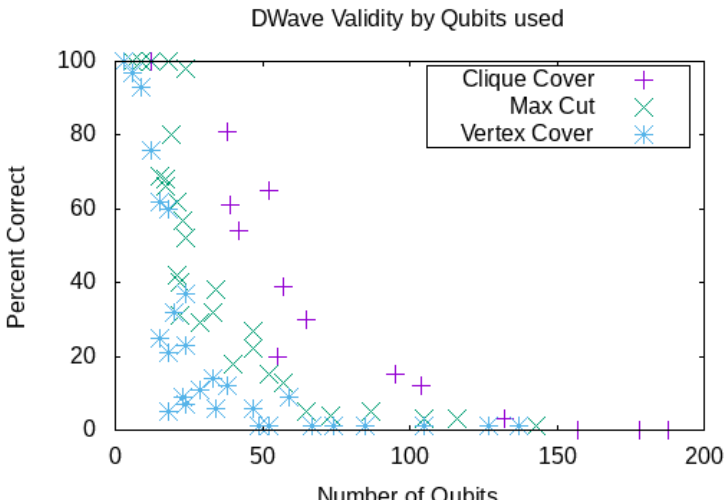


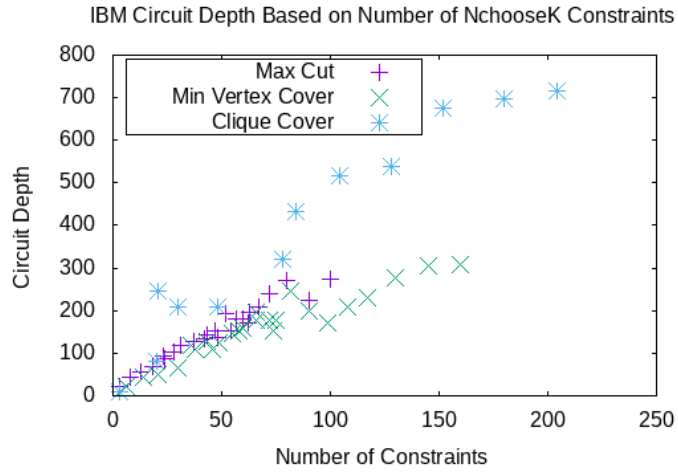
NchooseK Results

- Barrier of correct results on NISQ devices
- Limits on circuit depths → NISQ
- Non-expert programming:
  - —
  - — vs. linear/quadratic
    - -Changing coefficients for scaling

https://github.com/lanl/NchooseK

Problem	Class	# non-symm. constraints	NchooseK constraints	QUBO terms
Exact Cover	NP-C	n	n	nN^2
Min. Cover	NP-H	n	nN	nN^2
Min. Vert. Cover	NP-H	2	$\|V\|+\|E\|$	$\|V\|+\|E\|$
Map Color	NP-C	2	$\|V\|+\|E\|n$	$\|V\|n^2+\|E\|n$
Clique Cover	NP-C	2	$n\|V\|^2-\|E\|$	$n\|V\|^2-\|E\|$
k-SAT	NP-H	2	$n+m$	nm^2+n^2m
Max. Cut	NP-H	1	$\|E\|$	$\|E\|+\|V\|$





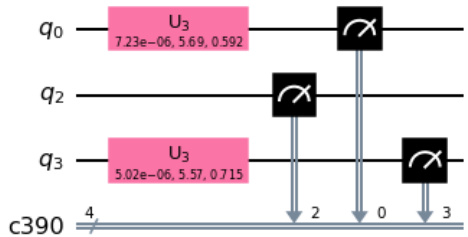
(2) Circuit Approximations on NISQ Devices

[SC'21] (Ellis Wilson, LANL)

- Motivation: Current machines → too much noise
- Especially bad on long circuits
  - Each gate applied adds noise (especially multi-qubit gates)
  - Qubits holding excited states decohere
- Idea: Shorter, more imprecise circuits may perform better
  - Similar to levels of precision in classical computing
- ➢ Given unitary, search compiler finds short circuits
- ➢ Short approximate circuits can outperform long precise circuits

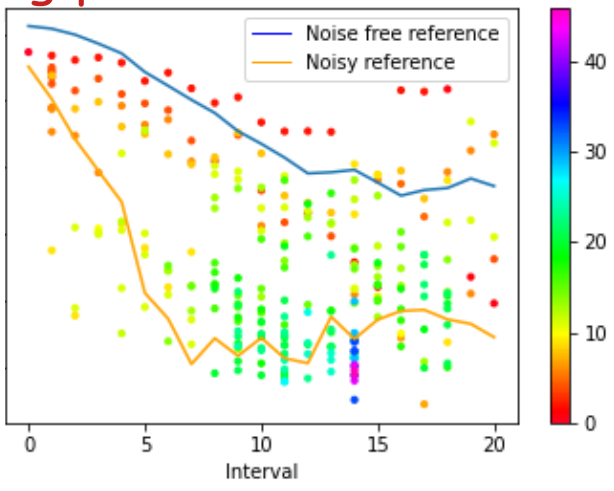
Obtain a unitary matrix

Use synthesis software to generate approximate circuits



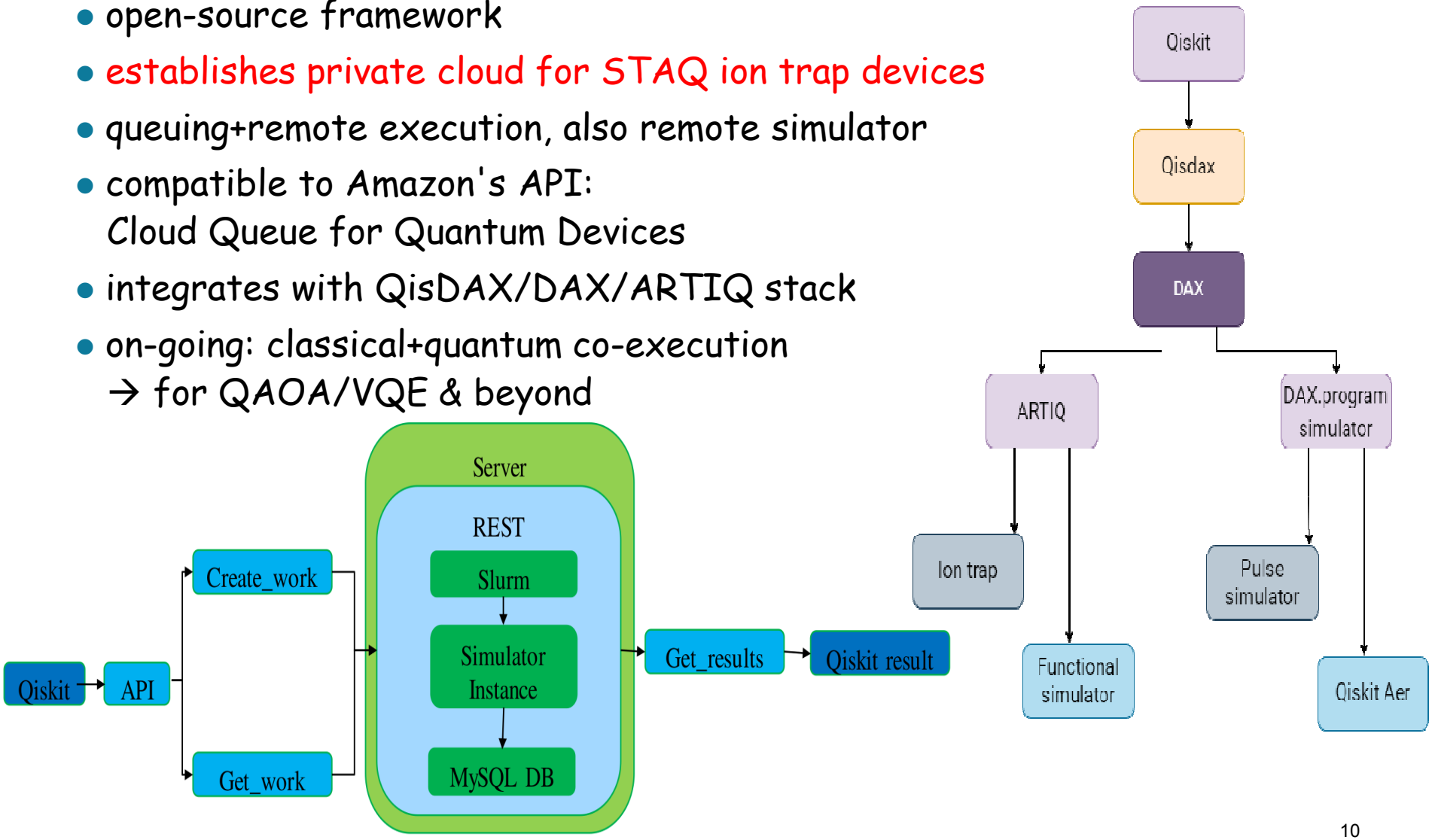
Select appropriate approximate circuits

Run approximate circuits on quantum simulator or machine



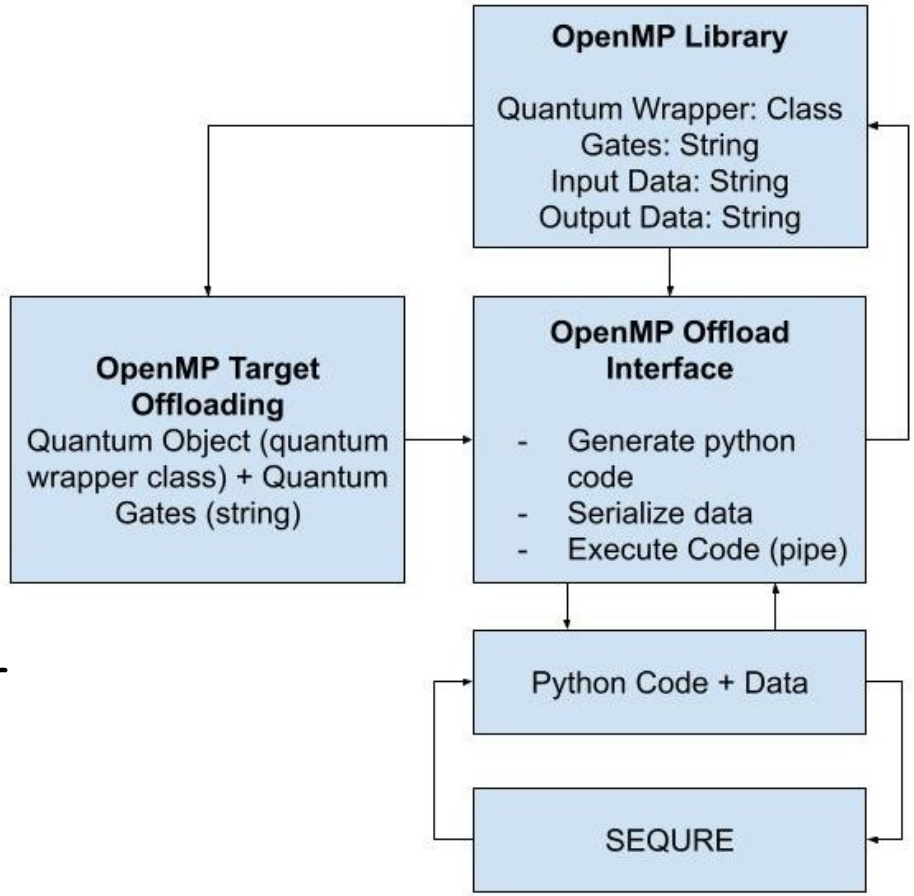
(3) SEQURE: Submission Environment for Quantum Resources w/ Duke

- open-source framework
- establishes private cloud for STAQ ion trap devices
- queuing+remote execution, also remote simulator
- compatible to Amazon's API:
  Cloud Queue for Quantum Devices
- integrates with QisDAX/DAX/ARTIQ stack
- on-going: classical+quantum co-execution
  → for QAOA/VQE & beyond



OpenMP Extension

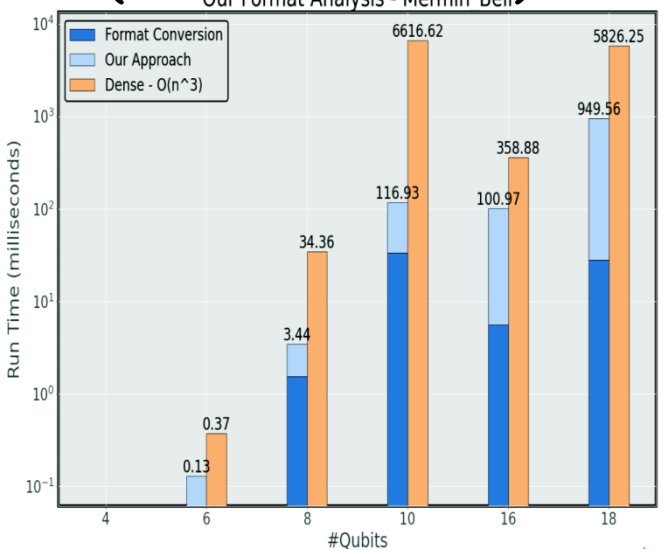
- Quantum in OpenMP
- implemented in LLVM
- auto-generates python script
  → submitted to Sequre (slurm)
- pulls results back into C/C++
  - post-processing options: py/C/C++




```
#pragma omp requires reverse_offload
#pragma omp target map(to: qubits[0:N]) // state preparation
{
  for (i = 0; i < iterations; i++) { // repeat VQA circuit
    #pragma omp target map(to: angles[0:M]) // just-in-time transpilation
    #pragma omp metadirective when( device={arch(quantum),edsl(qiskit)}: )
    VQA circuit; // some Quantum eDSL, Qiskit or Pulse kernel;
    #pragma omp target device(ancestor: 1) map(from: qubits[0:N]) // read-out
    MPI_Broadcast(... qubits ...); // optionally distribute over nodes
    angles = classical_solver(qubits);
    MPI_Allreduce(0, ... angles ...); // receive all results, pick "best" angles
  }
}
```

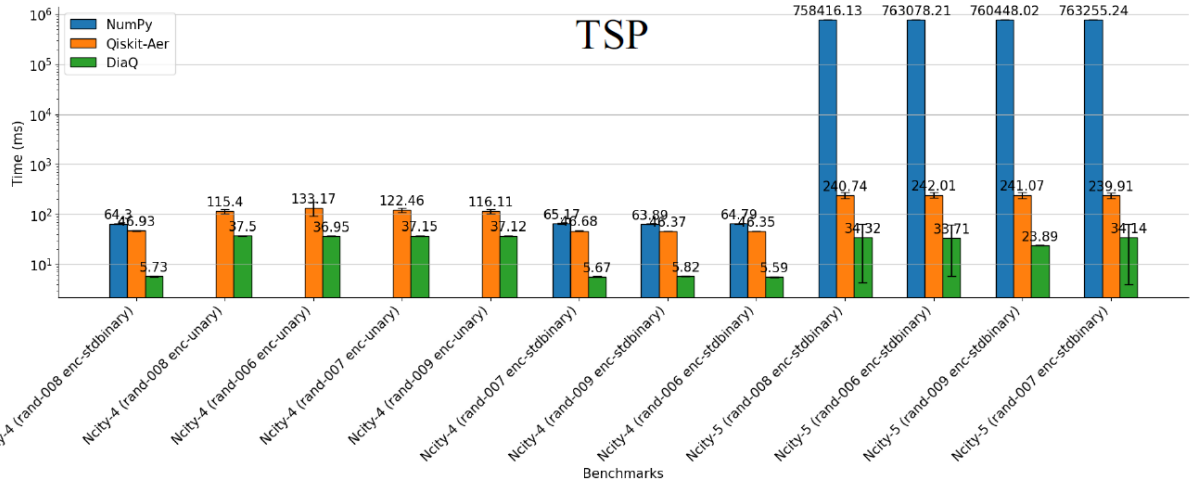
(4) Sparse Quantum Simulation via DiaQ w/ ORNL

- Custom sparse tensor format: DiaQ
- Exploit sparse patterns in unitaries
  - Simulation: state vector, unitary, Hamiltonian, tensor networks
- Lower memory footprint → simulation w/ more qubits feasible
- Lower simulation time → longer circuits
  - Demonstrated for benchmark (GHZ, …) and apps (VQE, …)
- Challenge: efficient active ops in sparse format (multi-D tensors)
  - Reshape, transpose, tensordot, also format conversion(!)
- C++ and pybind11 → integrated into
  - Quimb (tensor networks)
  - SV-Sim (PNNL)
  - QFw (ORNL)



DiaQ Results

- Orders of magnitude faster
  → need sparesity in problems
- Less memory required
  → scales beyond dense sim.
- Exploring
  - mixed dense/sparse
  - Sparse tensor sim.
    (MPS, PEPS, …)
- Bosonic simulation
  → higher encodings
  - Fock states
  - Qumodes
  - Positional, phase



(6) New qLPDC codes

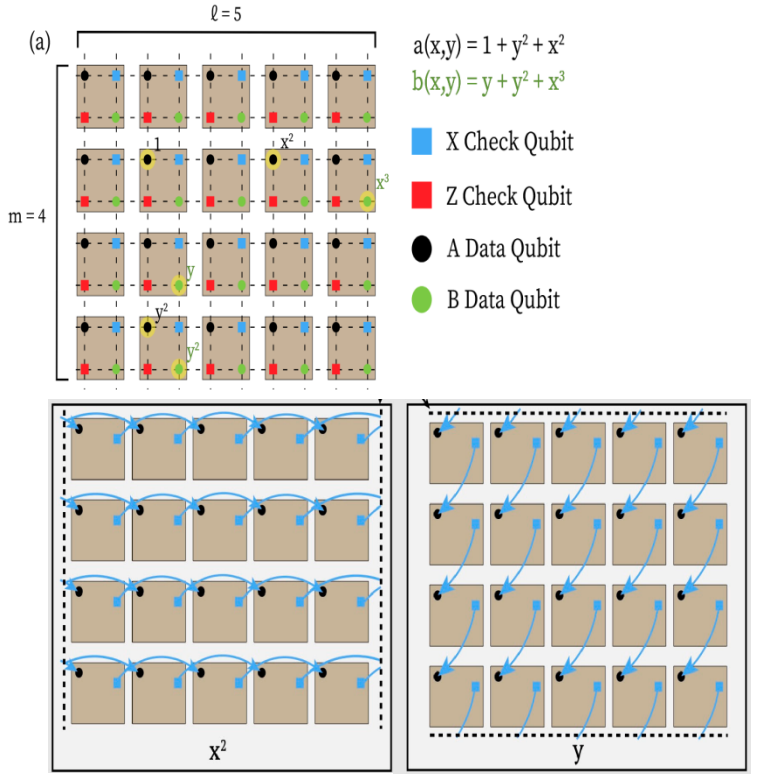
l	m	a	b	$[[n,k,d]]$
2	7	$1+xy+(xy)^3$	$1+xy+(xy)^{10}$	[[28,6,4]]
2	9	$1+(xy)^2+(xy)^{10}$	$1+(xy)^4+(xy)^8$	[[36,8,4]]
3	5	$1+xy+(xy)^2$	$xy+(xy)^3+(xy)^8$	[[30,4,6]]
3	7	$1+(xy)^2+(xy)^3$	$1+(xy)^3+(xy)^{11}$	[[42,6,6]]
3	8	$1+xy+(xy)^2$	$1+(xy)^2+(xy)^{10}$	[[48,4,8]]

1) Found new BB codes (algorithmic search)

- Determined by $l, m, a(x,y), b(x,y)$
- $H_X = [A|B], H_Z = [B^T|A^T]$
- $A = a(x,y), B = b(x,y)$
  - $x = S_l \otimes I_m, y = I_l \otimes S_m$
  - $S_i = I_i \gg 1$
- Usually use "pure" terms:
  - $a(x,y) = x^d + y^e + y^f$
  - $b(x,y) = y^g + x^h + x^i$

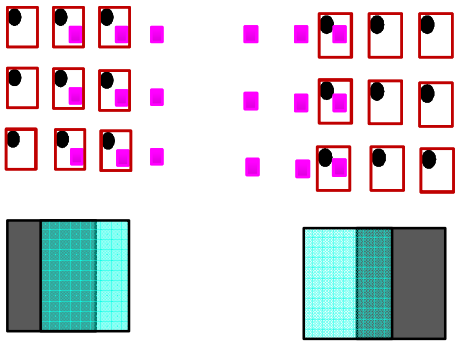
2) Found new class of Co-prime BB codes

- Same as BB codes, but l, m are coprime integers
- Usually use "mixed" terms:
  - $a(x,y) = (xy)^d + (xy)^e + (xy)^f$,
  - $b(x,y) = (xy)^g + (xy)^h + (xy)^i$
- $k = \gcd\left(a(x,y), b(x,y), (xy)^{lm} - 1\right)$

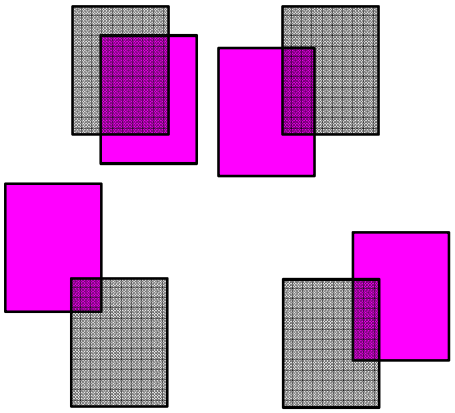


Explore Topology costs of Surface vs. BB

1. Cold Atoms: Find better layout for coprime BB codes

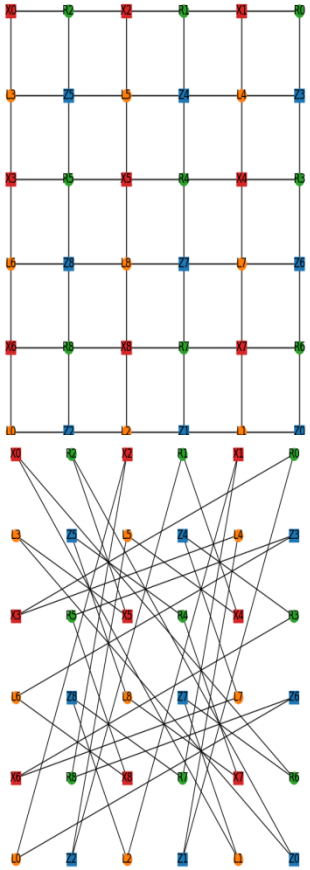


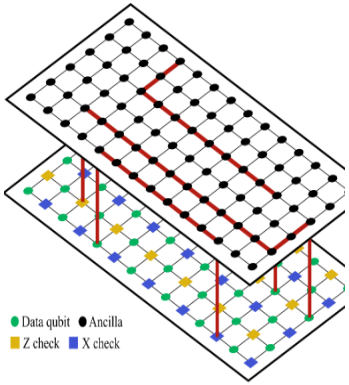
Pure terms



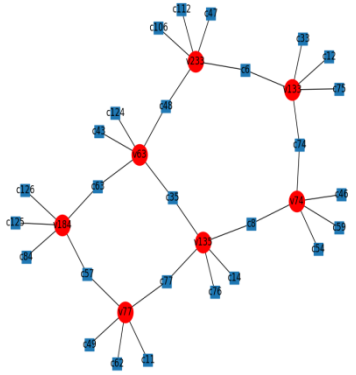
Mixed terms

2. Superconducting:
   - Reduce long-range connections
   - 3-layer model
   - Routing & scheduling





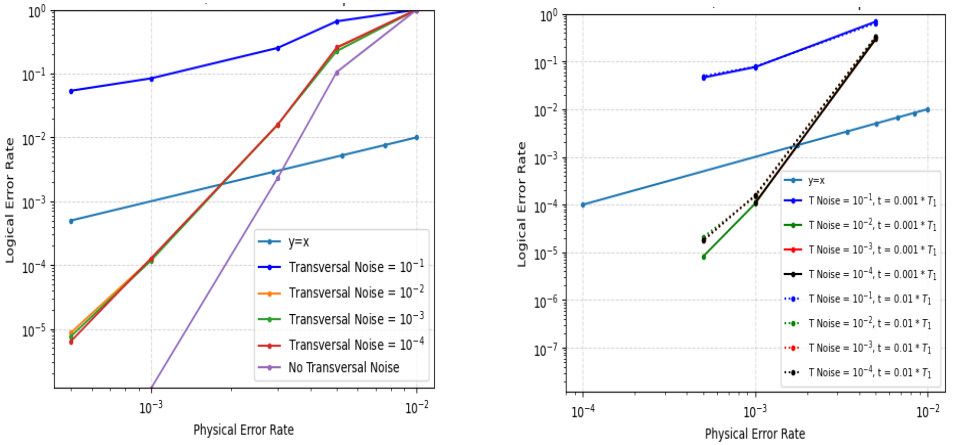
3. Decoders:
   - Trapping sets
   - BP-OSD opt.



Example error patterns in Panteleev's [[254, 28]] GB code

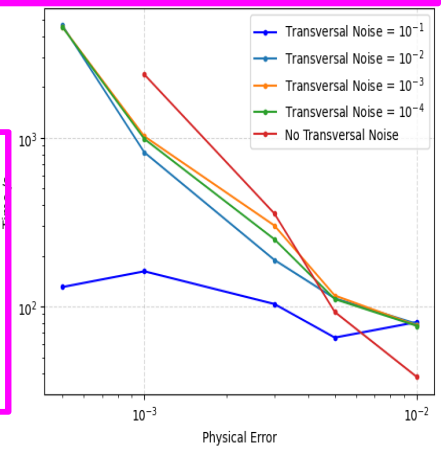
(7) HPC Simulation of Transversal CNOTs for arbitrary BB codes over Ebits

- Distributed Quantum: Multiple devices
  - Conntected via EPR pairs (ebits)
  - Requires teleportation
  - Alternative: physical shuttling (traps)
- HPC simulator for circuit-level simulations of transversal CNOTs between arbitrary "non-local" BB code blocks
- Heavily modified STIM circuit and code from [arXiv:2403.18901v1]
  - n code blocks, simulate noise
  - Syndrome decoding with BP-OSD
- explore impact of different DQC parameters on the logical error rate
  - E.g.: ebit noise, ebit production time
- Used MPI to parallelize pipeline (certifying logical error rates in the 10^{-7} region needs 10 million to 100+ million syndrome decodings!)
- dynamically adjusts amount of work per task dependent on early results → reduce MPI overheads



(left) Physical vs logical error rate for a transversal CNOT between two [144,12,12] BB codes under the same local physical noise but different transversal noise.

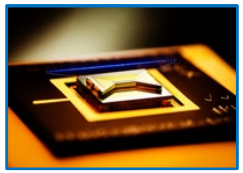
Time to achieve > 100 logical errors using 256 MPI tasks. As physical error rate decreases, time to identify 100> logical errors increases exponentially → due to the exponentially decreasing prob of logical error



Summary

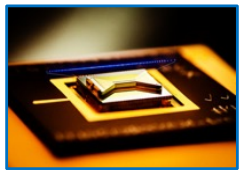
- NchooseK: DSL for SAT problems [SC'22]
- approximate circuit synthesis [SC'21, QCE'24]
- Sequre: Quantum accelerators within HPC
- DiaQ: sparse tensor simulation
- FTQC: error codes and distributed quantum

HPC view



Summary

- NchooseK: DSL for SAT problems [SC'22]
- approximate circuit synthesis [SC'21, QCE'24]
- Sequre: Quantum accelerators within HPC
- DiaQ: sparse tensor simulation
- FTQC: error codes and distributed quantum

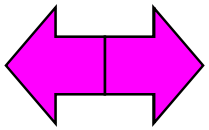


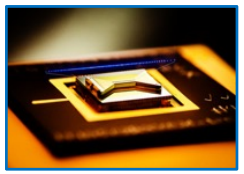
Physicist's view

Summary

- NchooseK: DSL for SAT problems [SC'22]
- approximate circuit synthesis [SC'21, QCE'24]
- Sequre: Quantum accelerators within HPC
- DiaQ: sparse tensor simulation
- FTQC: error codes and distributed quantum

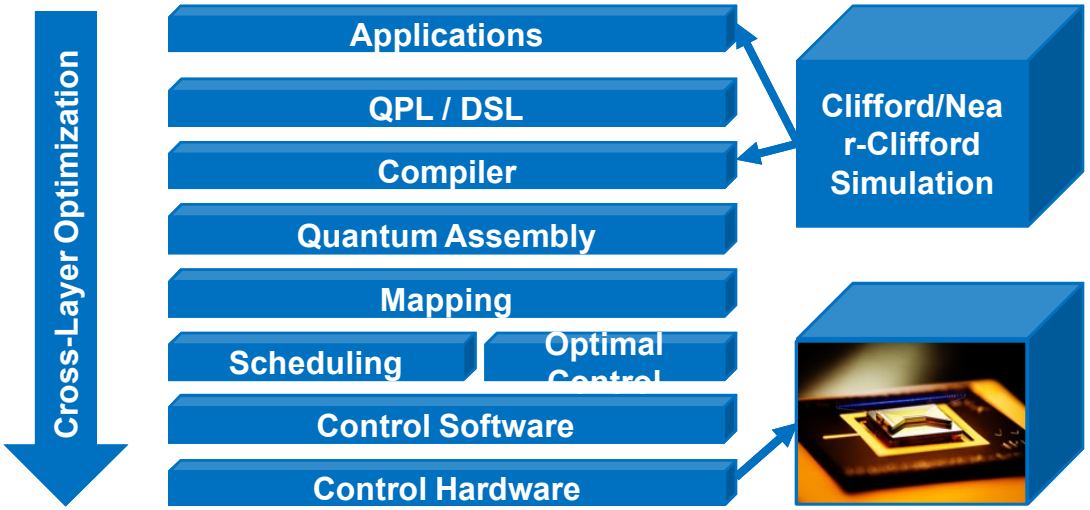
Vision: Equal partners





Summary

- NchooseK: DSL for SAT problems [SC'22]
- approximate circuit synthesis [SC'21, QCE'24]
- Sequre: Quantum accelerators within HPC
- DiaQ: sparse tensor simulation
- FTQC: error codes and distributed quantum
- QACTI: Quantum Advantage-Class Trapped Ion system



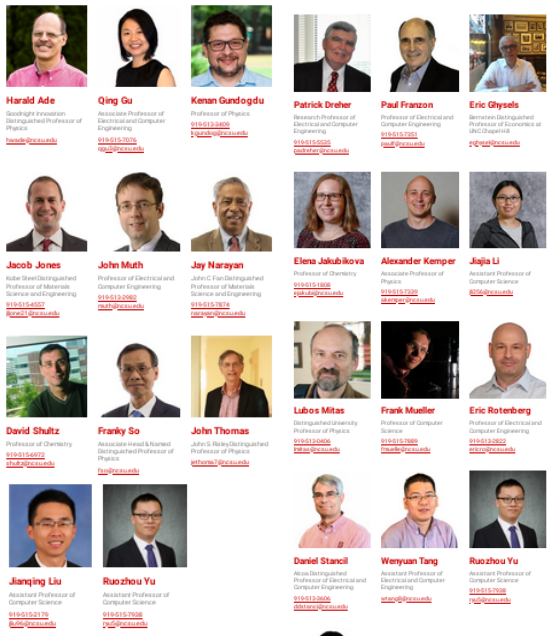
← Sandia Ion Trap @ Duke

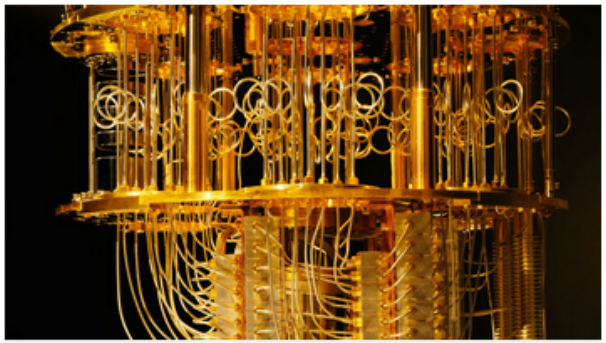
Quantum Innovation Center @ NCSU

- Chemistry
- Computer Science
- Electrical and Computer Eng.
- Mathematics
- Physics
- Materials Science and Eng.

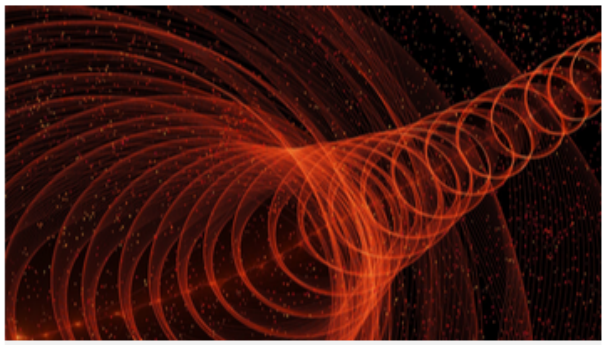
NC State Named First University-Based IBM Q Hub in North America

May 9, 2018 | University Communications | 2-min. read





Quantum Computing



Quantum Materials

Quantum Networking

Acknowledgements (Mueller Group)

- Ack. funding by NCSU, NSF STAQ+QLCI(RQS), DOE(LANL, LBNL)

- See http://moss.csc.ncsu.edu/~mueller for more papers, excerpt:

- **Synthesis of Approximate Parametric Circuits for Variational Quantum Algorithms by Blake Burgstahler, et al.in QCE'24**
- **DISQ: Dynamic Iteration Skipping for Variational Quantum Algorithms by Junyao Zhang, et al. in QCE 2023**
- **QisDAX: An Open Source Bridge from Qiskit to Ion Trap Quantum Devices by Kaustubh Badrike et al. QCE 2023.**
- **Combining Hard and Soft Constraints in Quantum Constraint-Satisfaction Systems by Ellis Wilson, Frank Mueller, Scott Pakin in SC'22**
- Just-in-time Quantum Circuit Transpilation Reduces Noise by Ellis Wilson, Sudhakar Singh, Frank Mueller in IEEE International Conference on Quantum Computing and Engineering (QCE), Oct 2020, Preprint arXiv:2005.12820
- Empirical Evaluation of Circuit Approximations on NISQ Devices by Ellis Wilson, Frank Mueller, Lindsay Bassman, Costin Iancu, Preprint arXiv:2107.06701, Supercomputing'21
- Quantum Annealing Stabils with Applications to Fuel Loading of a Nuclear by Joseph Fustero, Scott Pakmtag Frank Mueller, QCE
- Codes at github

Duke UNIVERSITY

Los Alamos NATIONAL LABORATORY

NC STATE UNIVERSITY
Department of Computer Science

UNIVERSITY OF MARYLAND

BERKELEY LAB