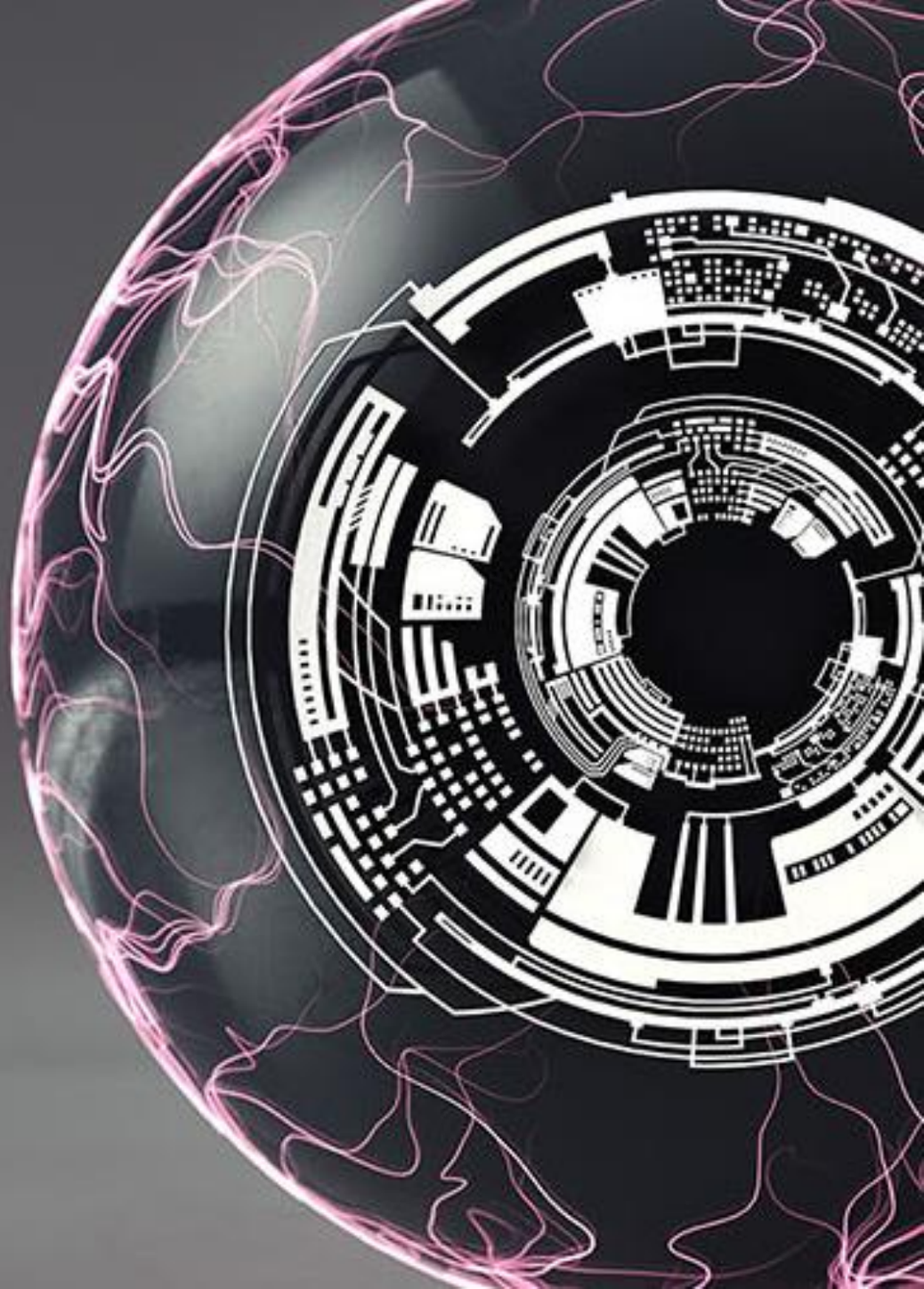

Microsoft
Microsoft Research
Faculty
Summit
2016

Microsoft

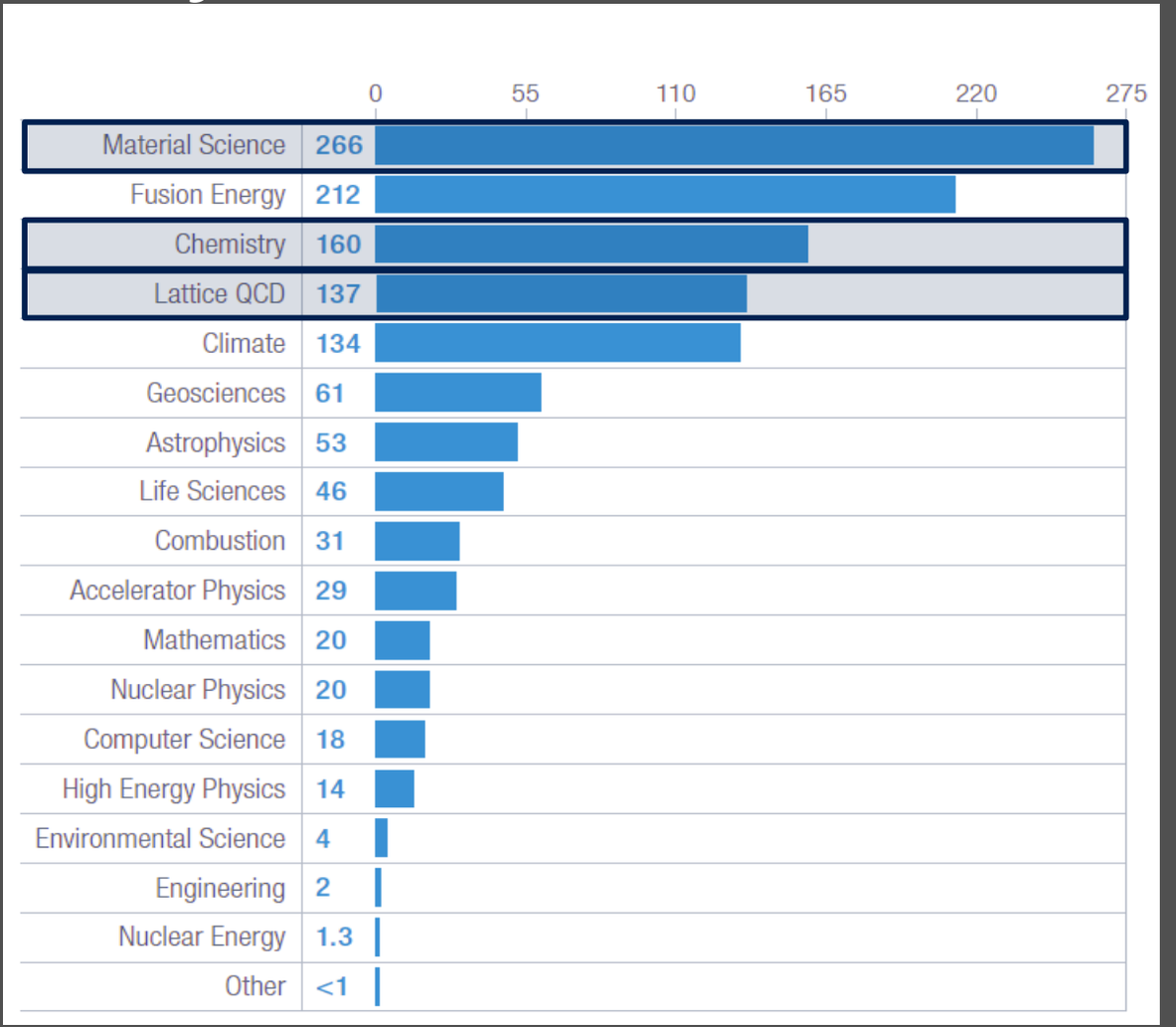
Quantum Algorithms: Today and Tomorrow

Dave Wecker
Microsoft Research, StationQ



Simulating physical systems

Hours (on Cray XE6)

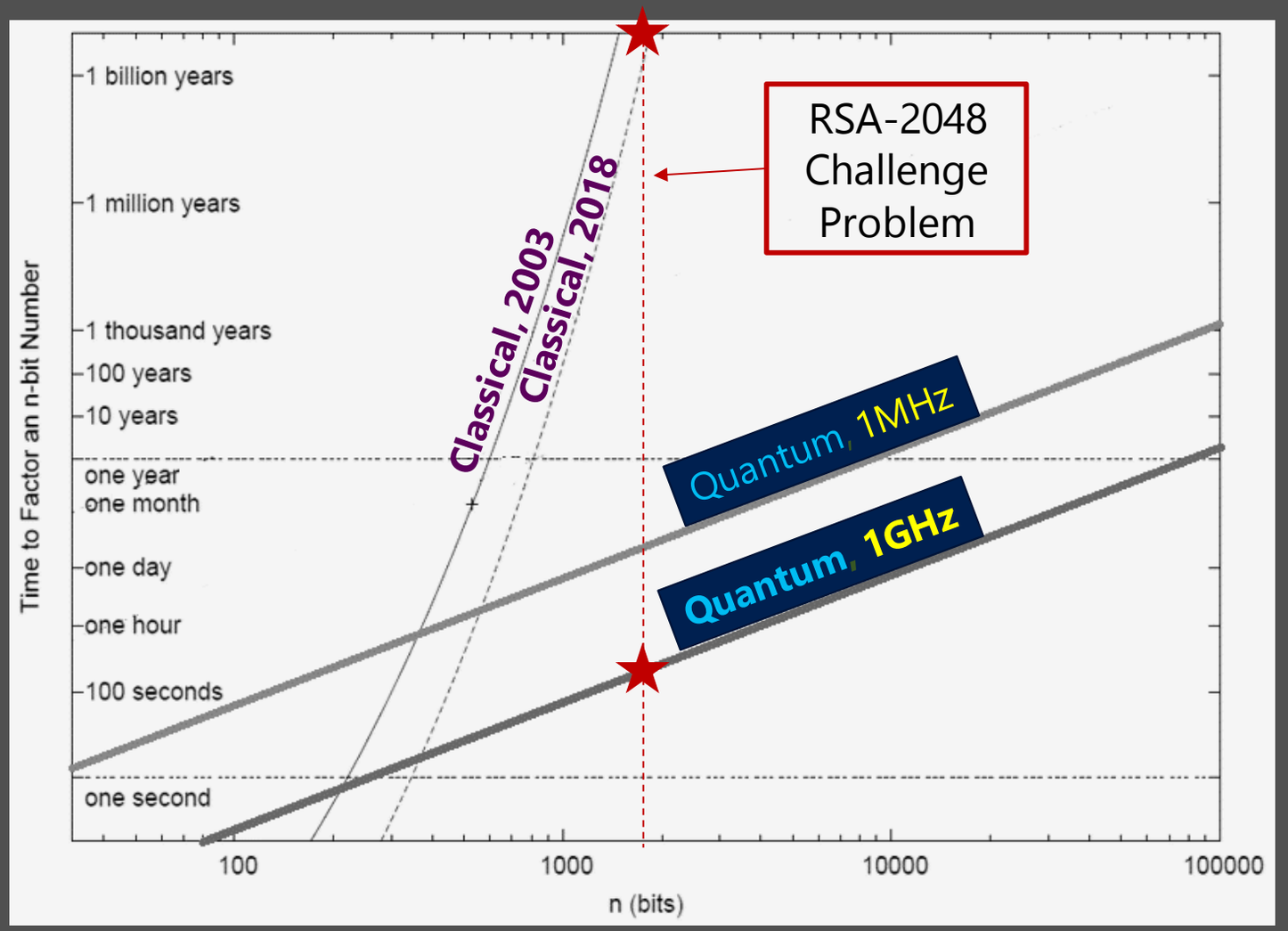


The promise…

Simulating 250 interacting qubits requires ~$\mathbf{10^{80}}$ classical bits!

State of $\boldsymbol{N}$ interacting qubits: ~ $\mathbf{2^N}$ bits of info!

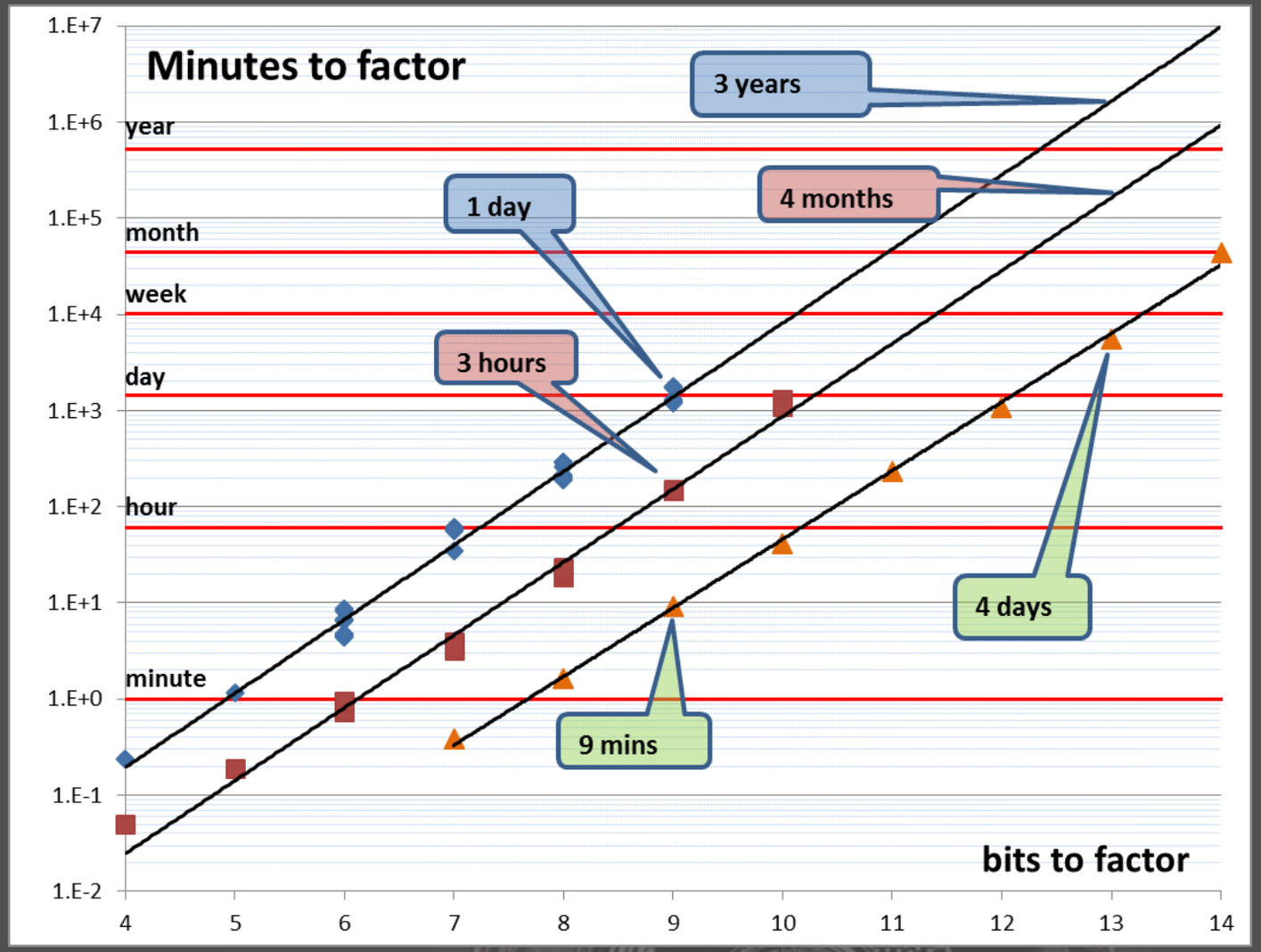
Breaking RSA



LIQ$Ui|\rangle$ Efficient Classical Simulation of Quantum Algorithms

- Only materialize entangled parts of state vector (Skyline Vectors)
- Re-order tensor products of gates to keep gate matrix dense
- Permute state vector to block-diagonalize generated sparse matrix
- Permute gate matrix to keep from re-ordering state vector
- Grow gate matrix to keep from re-ordering state vector
- Keep a running permutation until measurement
- Use measurement to shrink memory usage of state vectors
- Coalesce gates into more efficient larger arrays
- For Hamiltonians only maintain physically achievable states
- Exponentiate system for Trotter and Phase Estimation calculations
- Re-unitarize exponentiated matrices to maintain accuracy

Shor's algorithm results



Microsoft

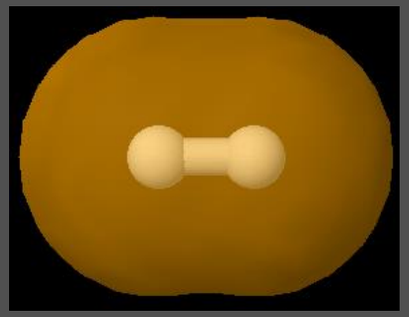
Quantum Chemistry

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs}\, a_p^\dagger a_q^\dagger a_r a_s$$

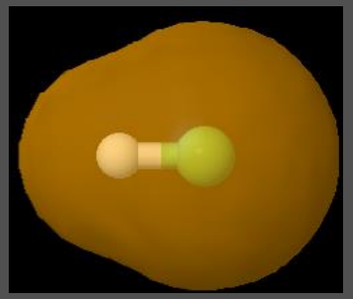
Ferredoxin (Fe_2S_2) used in many metabolic reactions including energy transport in photosynthesis

- *Intractable on a classical computer*
- *Assumed quantum scaling: ~24 billion years (N^{11} scaling)*
- *First paper: ~850 thousand years to solve (N^9 scaling)*
- *Second paper: ~30 years to solve (N^7 scaling)*
- *Third paper: ~5 days to solve ($N^{5.5}$ scaling)*
- *Fourth paper: ~1 hour to solve ($N^3, Z^{2.5}$ scaling)*

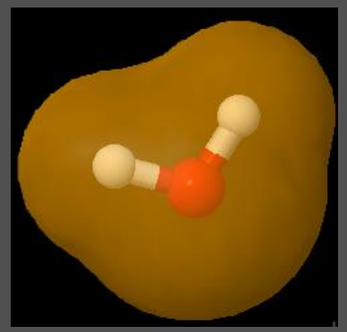
Molecules simulated in LIQ$Ui|\rangle$



H_2



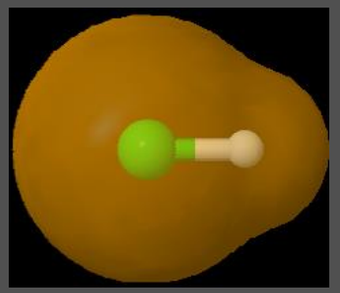
HF



H_2O

NH_3

CH_4



HCl

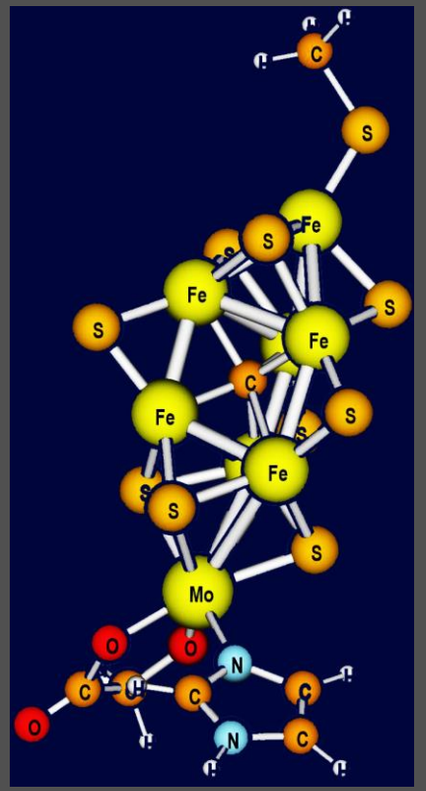
F_2

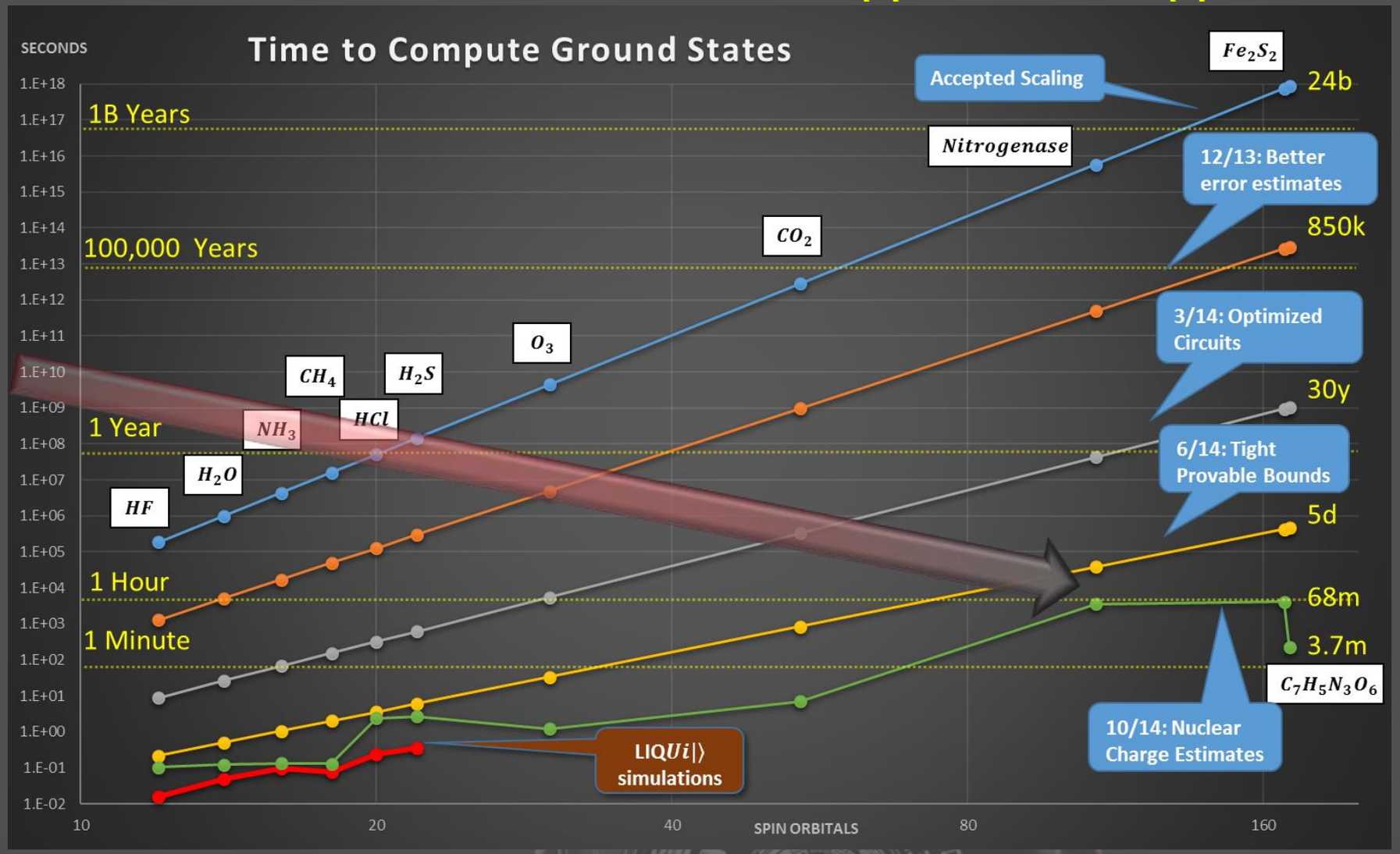
H_2S

Geometries and molecular models from http://www.colby.edu/chemistry/webmo/

Simulation Evidence

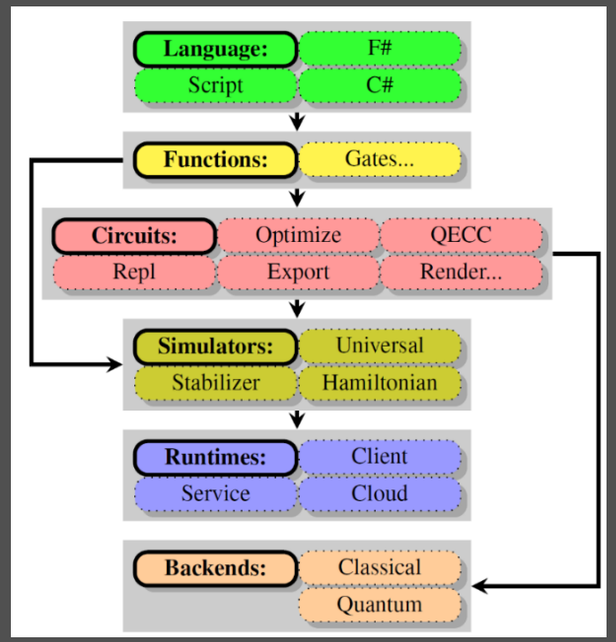
$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs}\, a_p^\dagger a_q^\dagger a_r a_s$$



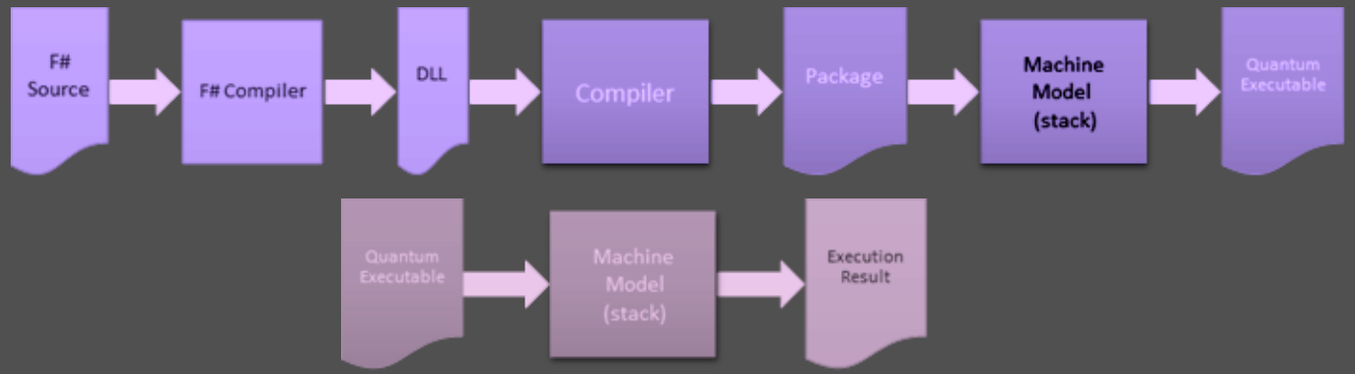


Microsoft

LIQ$Ui|\rangle$



SoL$i|\rangle$ and QCoDeS



Initial applications

Nitrogen fixation

100-200
qubits

Carbon capture

100-200
qubits

Materials science

100s-1000s
qubits

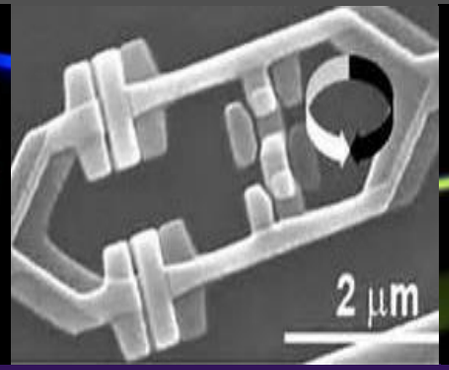
Machine learning

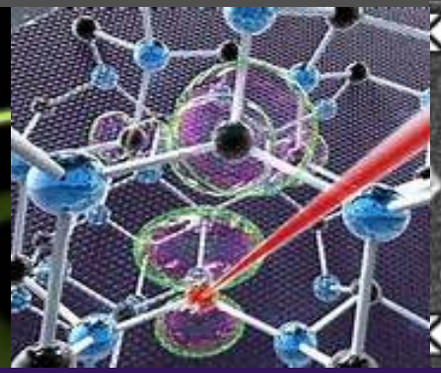
skype

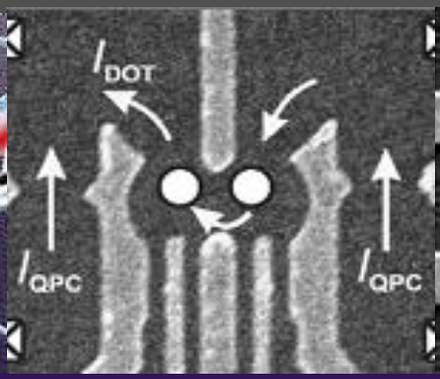
bing

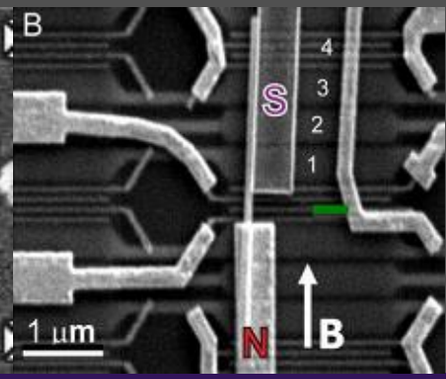
100s-1000s
qubits

Quantum hardware technologies









Ion traps

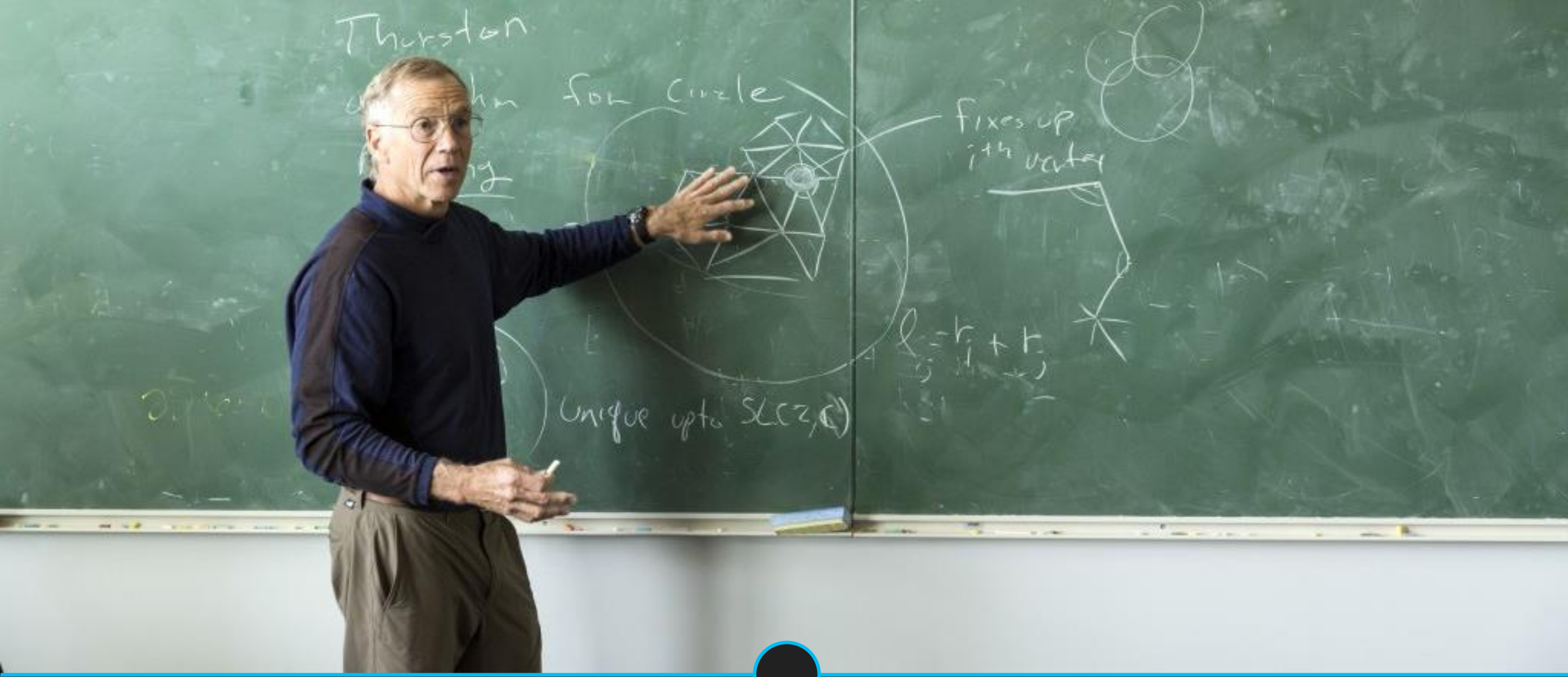
Super-conductors

Linear optics

NV centers

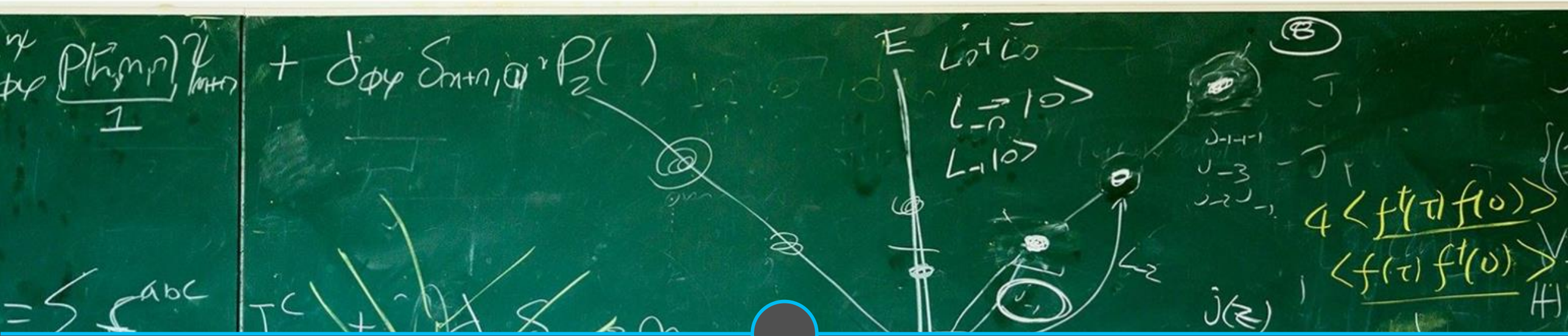
Quantum dots

Topological



2000

STATION Q



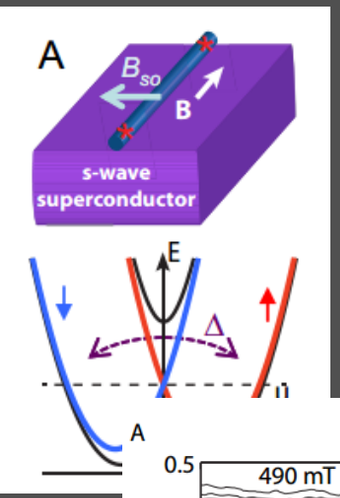
2006

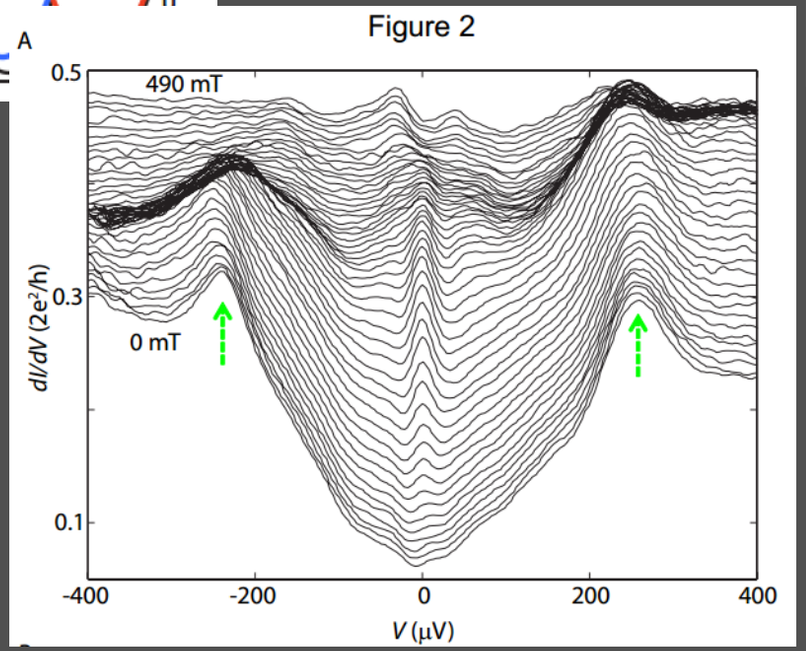
"MAJORANA PARTICLE GLIMPSED IN LAB."

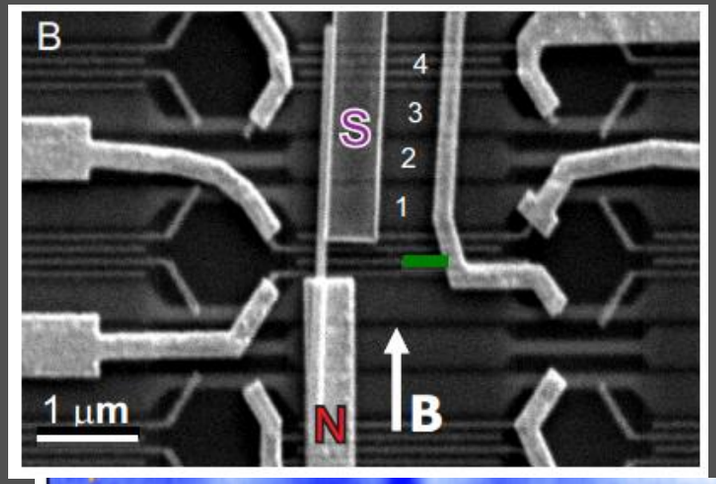
BBC NEWS

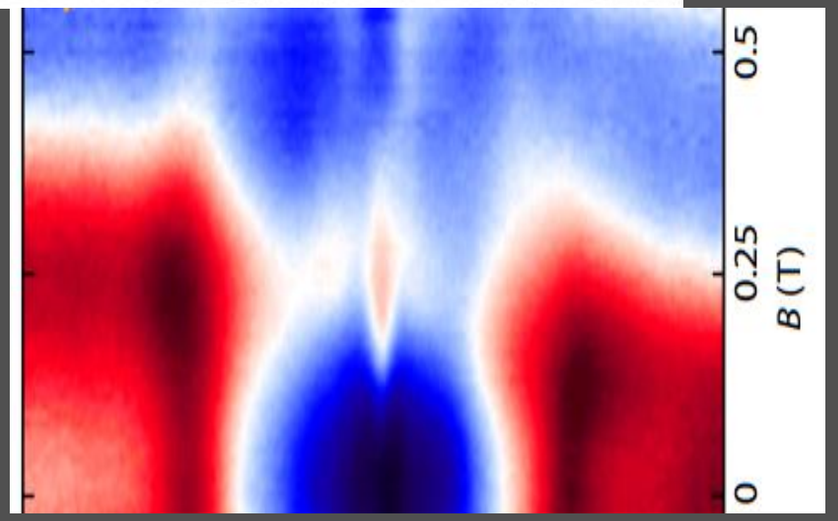
2012

Experimental Results









- Build experiment as described
- Measure zero-bias peak
- See that it goes away if we remove any of the necessary components

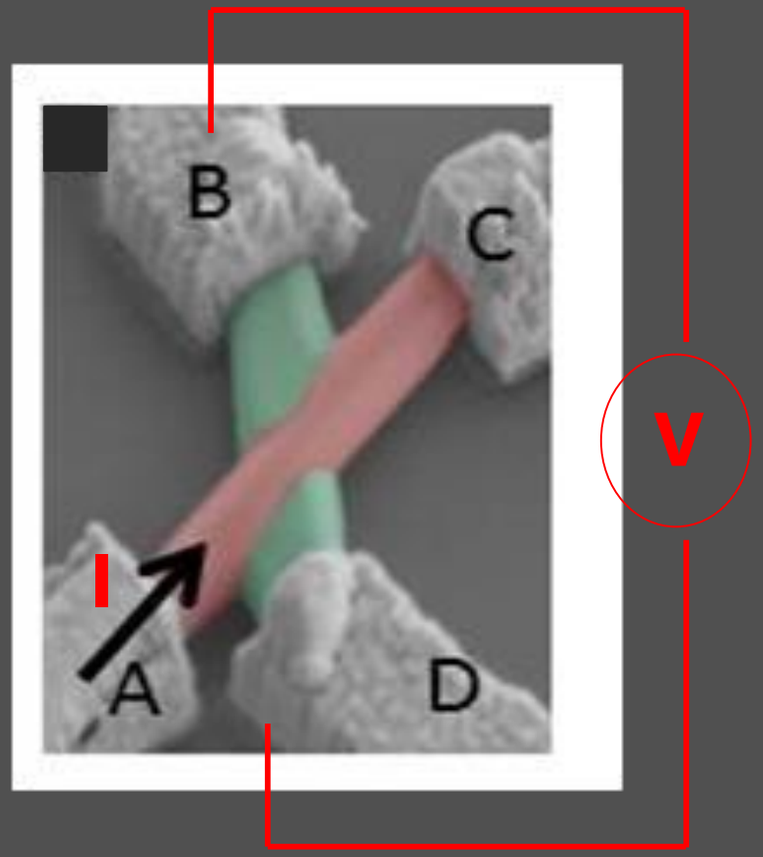
Mourik, … Kouwenhoven 2012
http://arxiv.org/abs/1204.2792

Charlie Marcus' lab in Copenhagen, Denmark

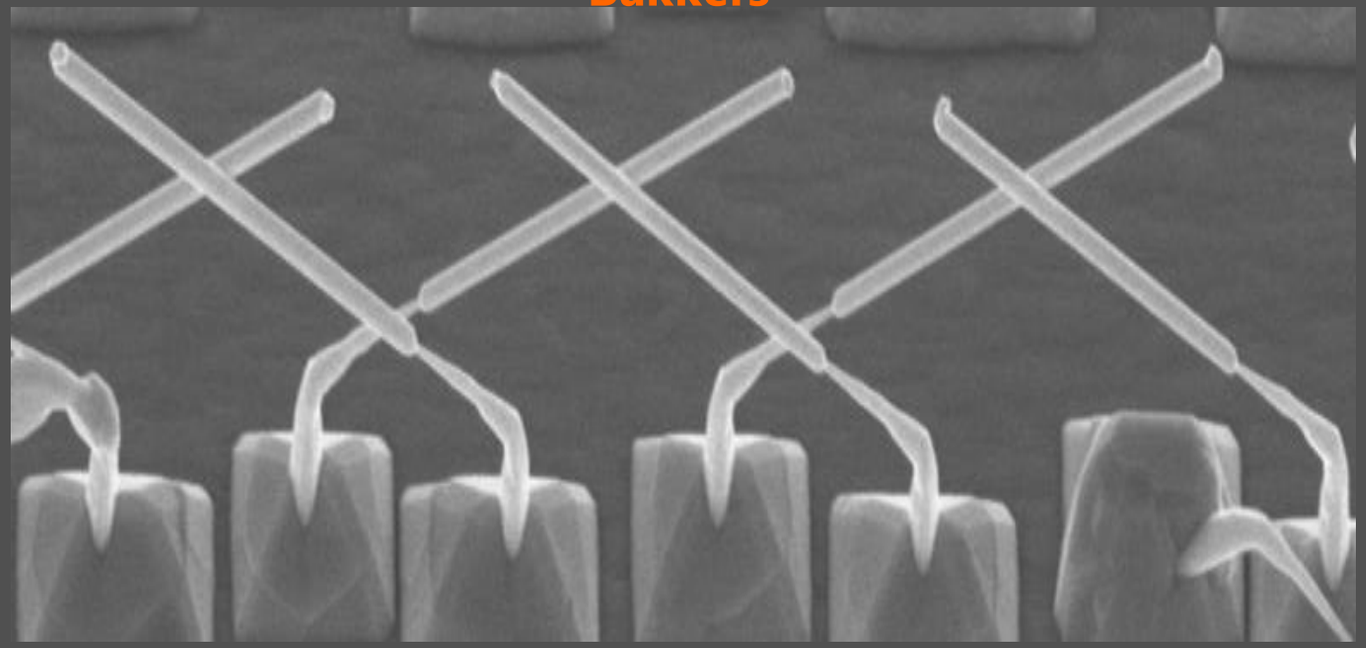
One of our primary experimental collaborators

Photos courtesy of: Professor Charlie Marcus

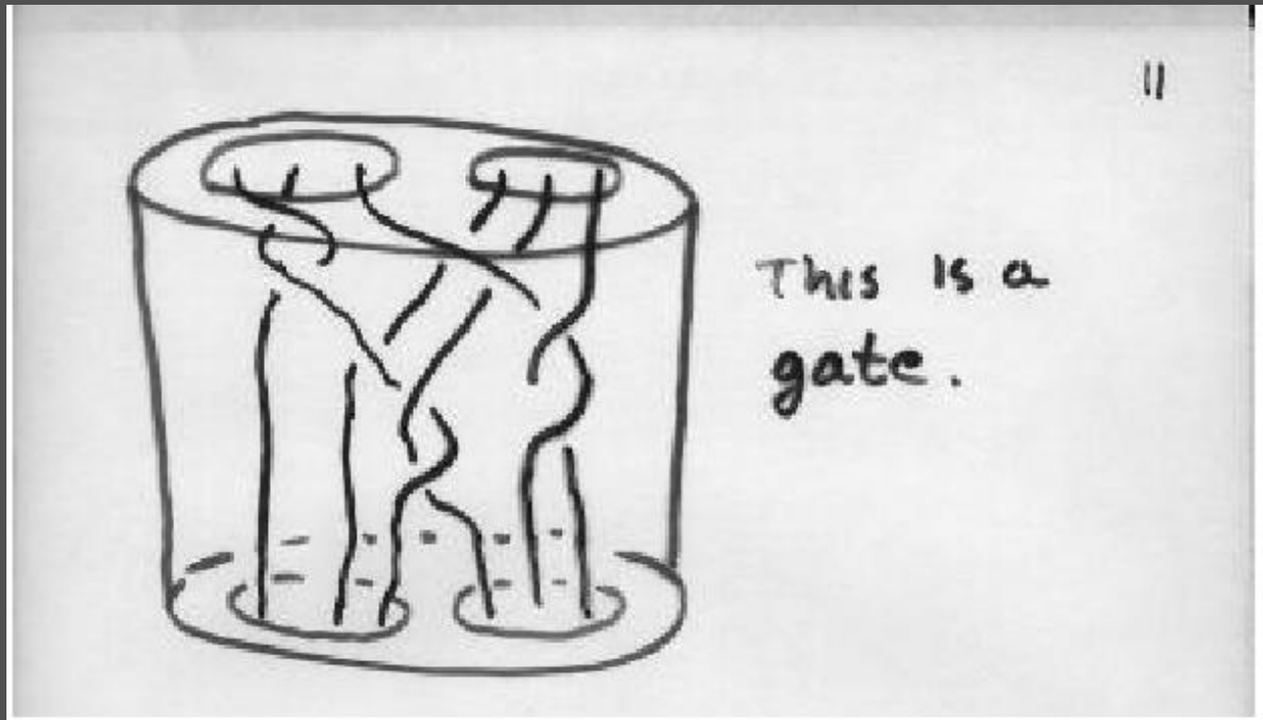
Junctions



Growth by Diana Car, Sébastien Plissard and Erik Bakkers



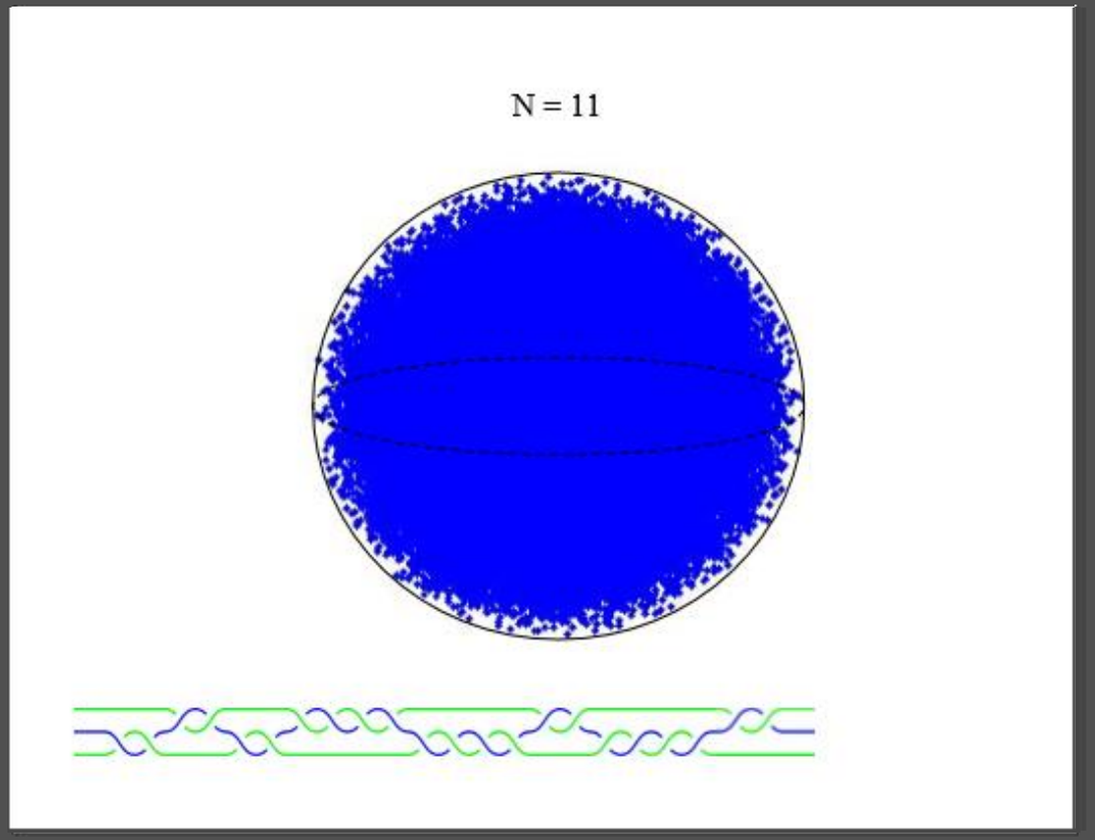
Inspiration



From "A topological modular functor which is universal for quantum computation"

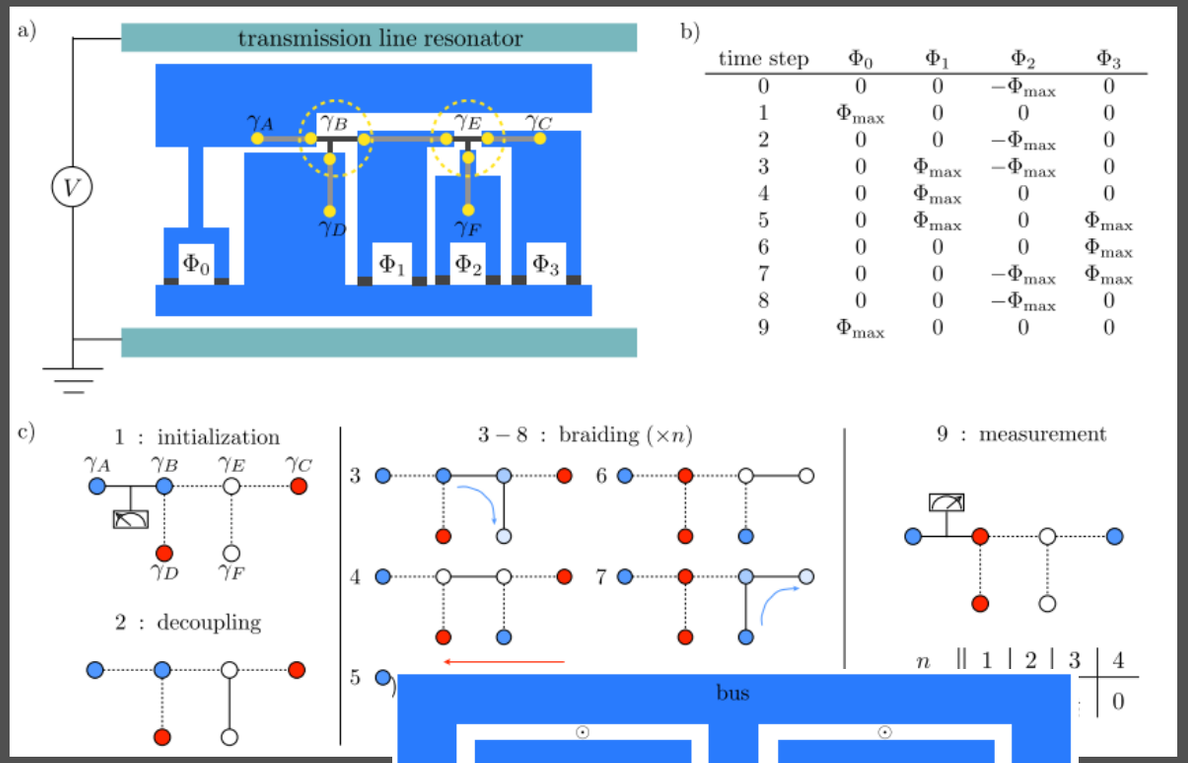
Talk given by Michael Freedman at "Mathematics of Quantum Computation", MSRI, Feb. 2000 (available online).

Braiding



From: Nick Bonesteel talk at KITP UCSB

Proposed Architecture

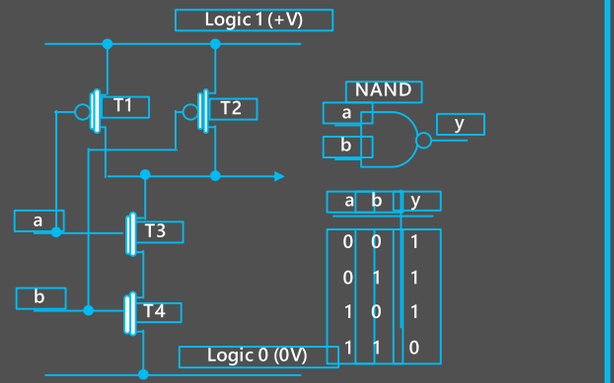
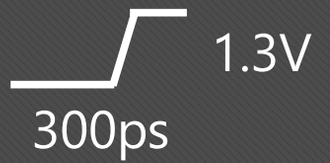


time step	Φ_0	Φ_1	Φ_2	Φ_3
0	0	0	$-\Phi_{\max}$	0
1	$\Phi_{\max}$	0	0	0
2	0	0	$-\Phi_{\max}$	0
3	0	$\Phi_{\max}$	$-\Phi_{\max}$	0
4	0	$\Phi_{\max}$	0	0
5	0	$\Phi_{\max}$	0	$\Phi_{\max}$
6	0	0	0	$\Phi_{\max}$
7	0	0	$-\Phi_{\max}$	$\Phi_{\max}$
8	0	0	$-\Phi_{\max}$	0
9	$\Phi_{\max}$	0	0	0

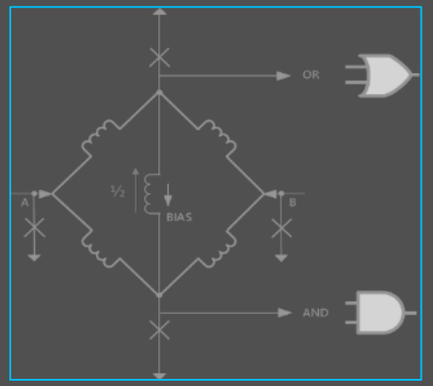
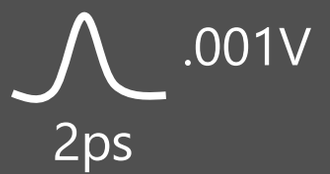
- Proposed experiment to braid Majoranas
- The same control structure is used for initialization, braiding and measurement
- Instead of moving Majoranas, they are coupled and de-coupled to de-localize them and then get them to appear at the desired location
- Must be done adiabatically
- May be scaled to much larger system

After Hyart et al 2013
http://arxiv.org/abs/1303.4379

Reciprocal Quantum Logic (RQL)

CMOS	Data	Energy
		10^{-15}J

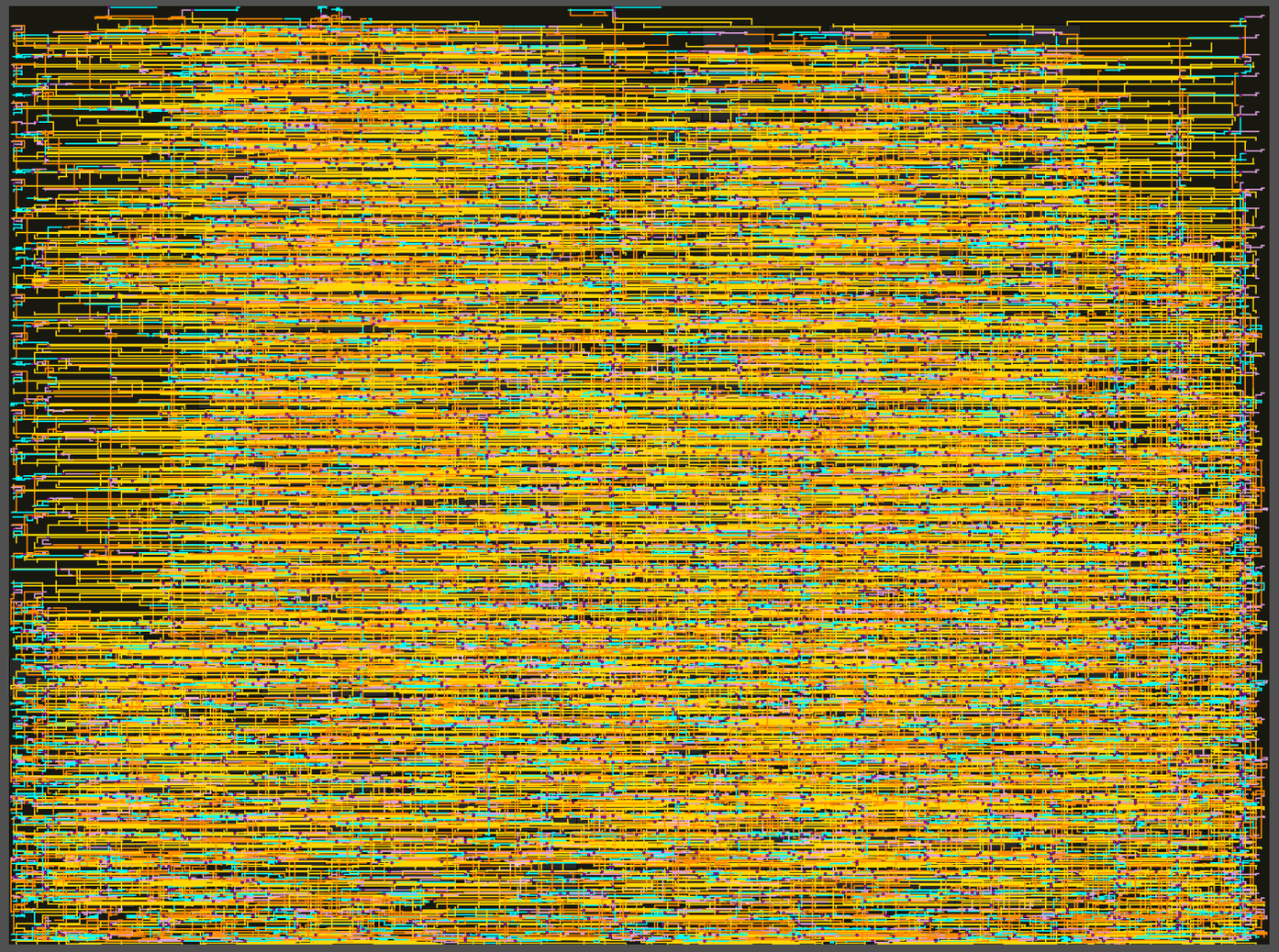
- Reached limits of technology scaling
- Significant energy in wires
- Frequency limited to 3-4GHz
- Not efficient at low temperatures

RQL	Data	Energy
		10^{-19}J

- Memory technology unproven
- CAD tools, methodology needed
- Relatively small scale demonstrations

Microsoft

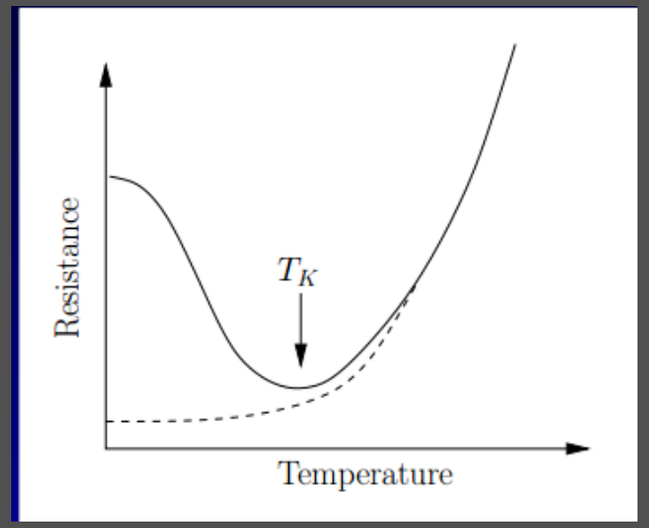
Reciprocal Quantum Logic (RQL)

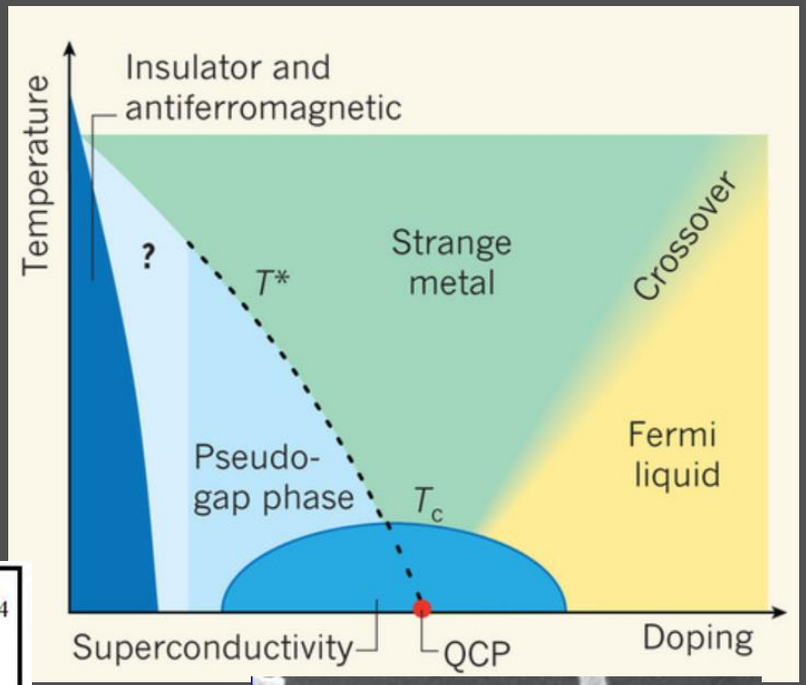


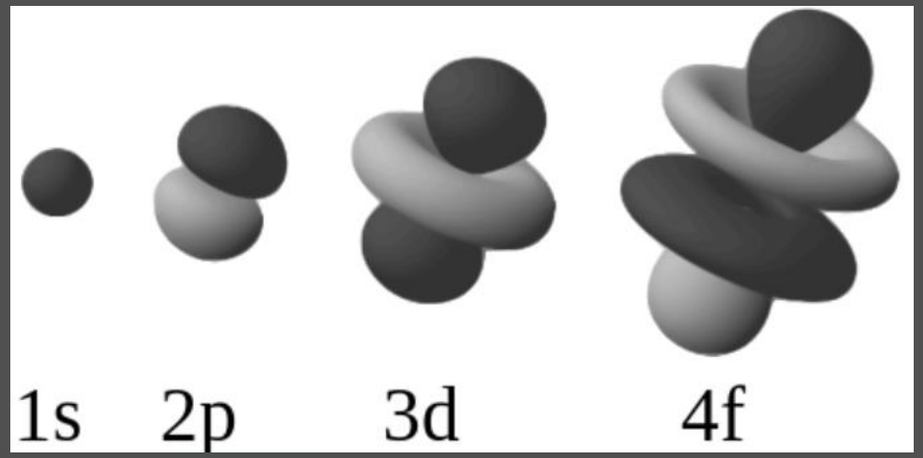
The Data Path Compiler (DPC)

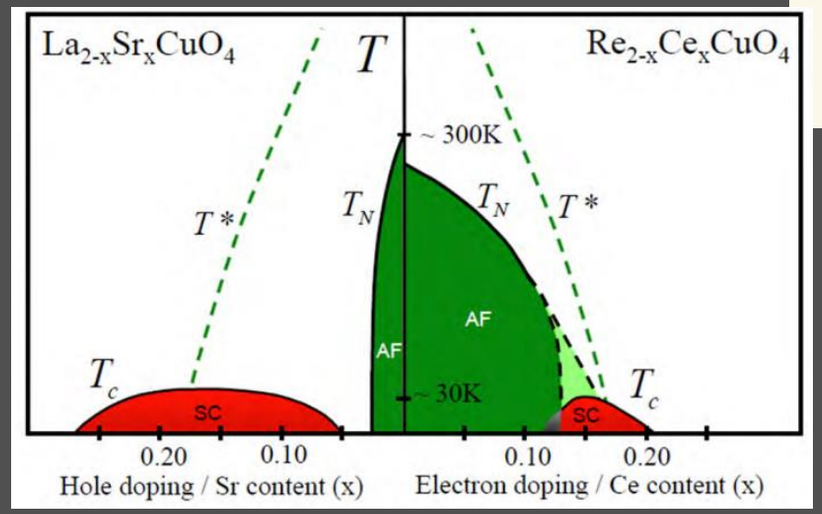
Quantum Algorithms for Quantum Impurity Problems

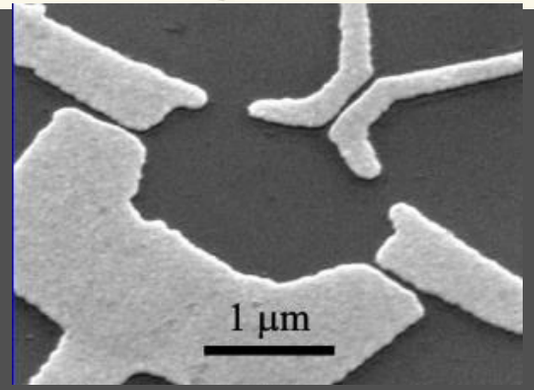
- Mott Insulators
- Transition Metal Compounds
  - Cuprates (e.g., High Tc SC)
- Lanthanides and Actinides
- Kondo Physics (Low temperature Resistance) from Magnetic Impurities
  - Quantum Dots









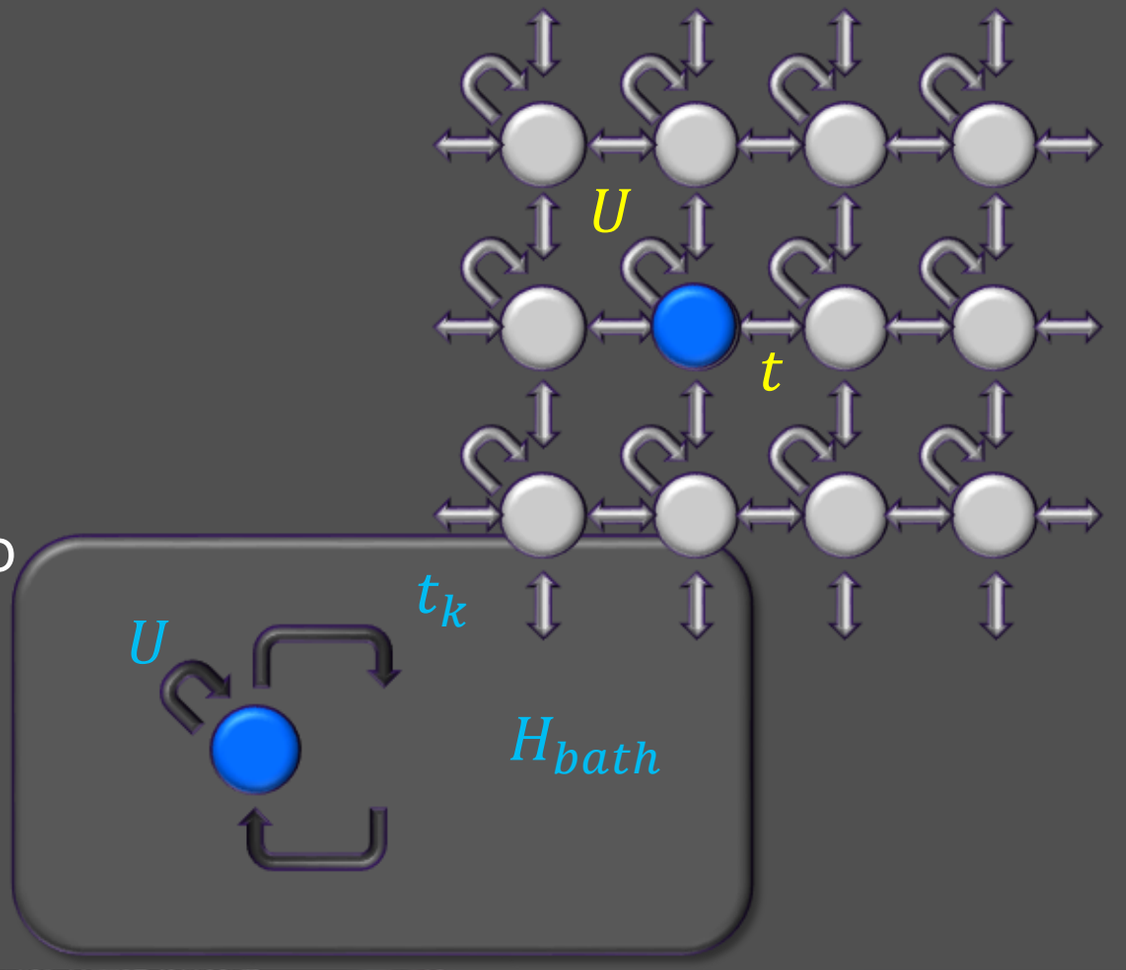


Materials Modeling

$$H_{hub} = U\Sigma_i n_{i\uparrow} n_{i\downarrow} - t\Sigma_{<i,j>,\sigma} c_{i\sigma}^{\dagger} c_{j\sigma}$$

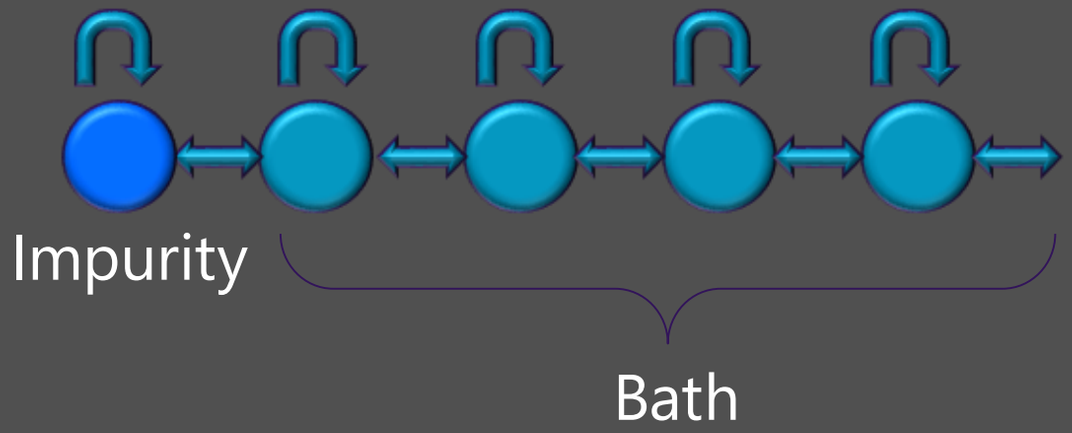
$$H_{imp} = U n_{\uparrow} n_{\downarrow} - \Sigma_{k,\sigma}\left(t_k c_{\sigma}^{\dagger} a_{k,\sigma}^{bath} + h.c.\right) + H_{bath}$$

- Solids have regular structure that can be modeled as lattices
- The Hubbard model only implements H_{pp} and H_{pqqp} terms
- This doesn't cover many of the materials we're interested in
- One can choose a single site in the lattice to model
- The effect of the rest of the lattice can be modeled in terms of its effect on this site



Anderson Impurity Model

- Choose a single place in the lattice to model (the impurity). This may contain a collection of local sites
- The impurity is typically a full two-body model
- The effect of the rest of the lattice can be modeled in terms of its effect on the impurity (the bath) via a Dynamic Green's function $G(\omega)$
- The bath may have many sites and interconnections

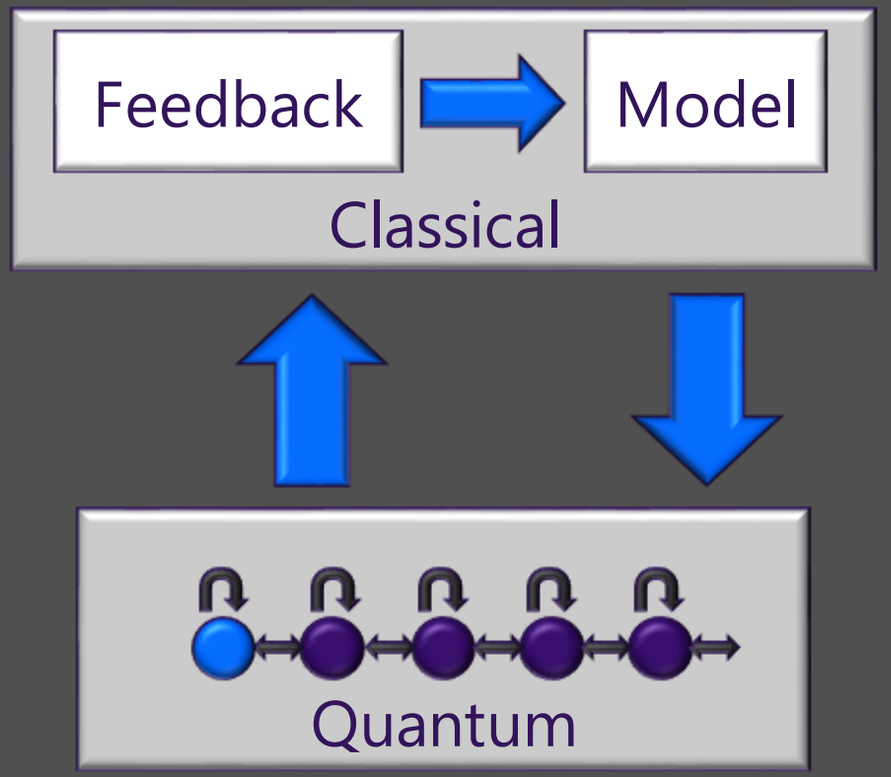


$$H = \sum_{ij} t_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{ijkl} w_{ijkl} a_i^\dagger a_j^\dagger a_k a_l + \sum_{ip} V_{ip} (a_i^\dagger a_p + a_p^\dagger a_i) + \sum_{pq} \epsilon_p a_p^\dagger a_q$$

Dynamical Mean Field Theory (DMFT)

$$G_{solver}(\omega) \rightarrow \sum(\omega) \rightarrow G(k,\omega) \rightarrow G_n(\omega) \rightarrow \Delta_n(\omega)$$

- We can posit an initial model for a material
- Measure quantum simulations at many sites and frequencies deriving a dynamical Green's function
- Use feedback to update model
- Repeat until converged
- The resulting model is defined classically and may be used to efficiently investigate many questions about the material



http://arxiv.org/abs/1012.3609

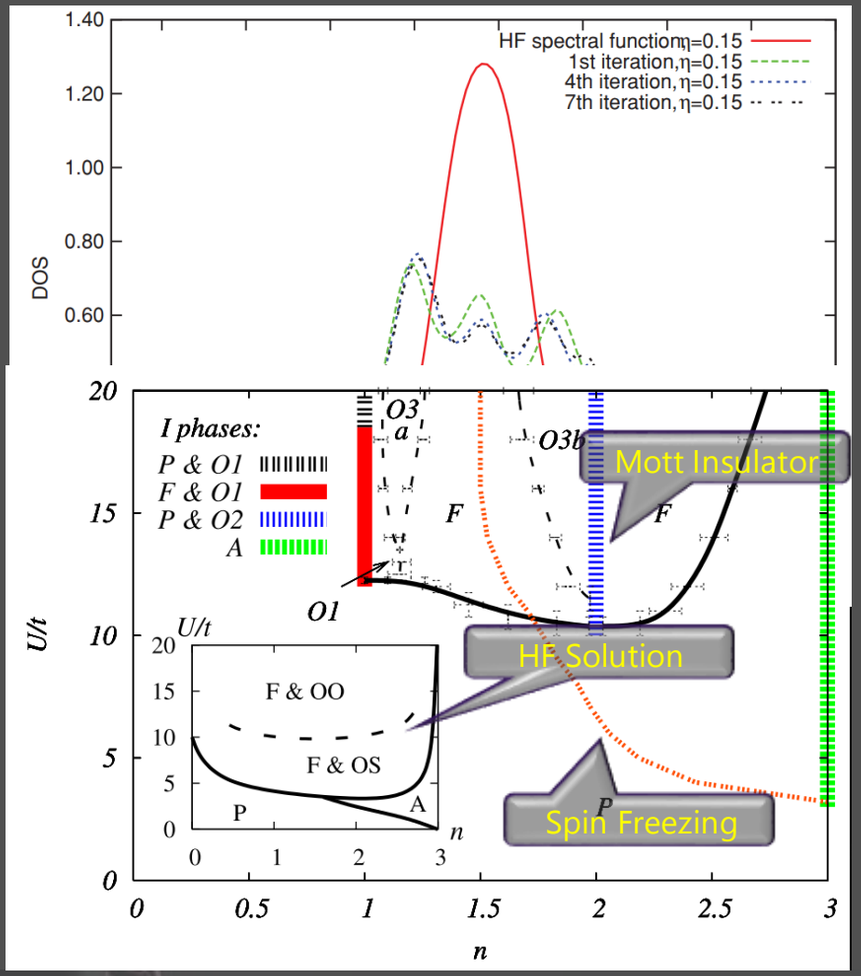
$$G_{solver}(\omega) = \langle c_i^{\dagger}(\omega)\, c_j(-\omega) \rangle$$

Dynamical Mean Field Theory (DMFT)

$$\boldsymbol{G_{imp}(\omega)^{-1} = (\omega + \mu + i0_{\pm})S - h_{imp} - \Sigma(\omega) - \Delta(\omega)}$$

- Cubic Hydrogen (H_2 crystal structure)
- Simulate at 2.5 $\mathbf{\AA}$ **with a** bath of 9 orbitals
- Density of States plot for different frequencies. Red curve is the Hartree-Fock solution (used as initial guess). DMFT Converges after 7 iterations
- 5-7 total orbitals for a *single* site is state-of-the-art.
- Example from Troyer et al the diagram shows a 3 shell degenerate solution (need 5 for D and 7 for F)
- A Quantum Computer could do a 4x larger impurity or 4x more orbitals than state-of-the-art with 200 qubits

http://arxiv.org/abs/1012.4474

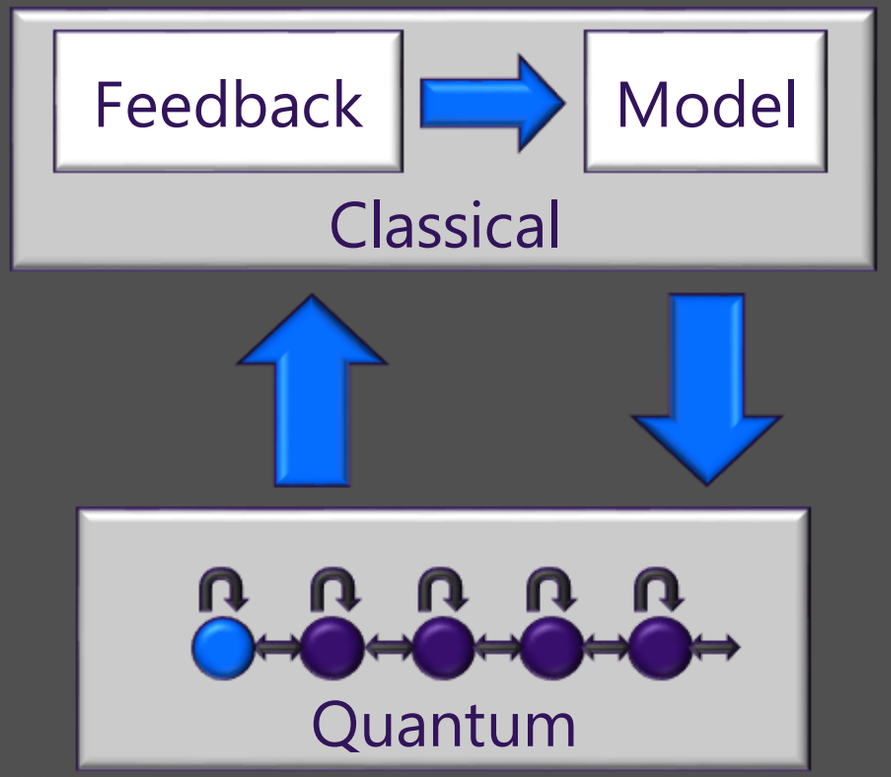


Microsoft

Variational Eigensolver

$$E_{gs} = \sum_k \left(H_{FF} + \sum_i \sum_j \theta_{ij} H_j + M(H_k) \right)$$

- Good: Only need to stay coherent for prep, evolution and measurement
- Bad: Parameters discovered by sampling (quadratically worse than PE)
- Useful for small machines with physical qubits that have relatively short coherence times.
- Only needs few thousand gate executions before losing coherence



http://arxiv.org/abs/1012.3609

Two Hubbard Plaquettes (16 qubits)

Gates	Samps/Pt	Samps	Evals	Energy	Overlap	Error	Hours
1000	91680	4.64E+07	506	-8.40204553	0.97616926	0.07625777	6.4
Measures	89395	4.52E+07	506	-8.44653127	0.98536880	0.03177203	6.3
5	86170	4.36E+07	506	-8.43120304	0.98722563	0.04710026	6.1
Speed	91652	4.64E+07	506	-8.45373504	0.98680546	0.02456826	6.4
1.00E-07	77749	3.93E+07	506	-8.32567973	0.97155555	0.15262357	5.5
eGS	92597	4.69E+07	506	-8.43120606	0.98206797	0.04709724	6.5
-8.4783	92404	4.68E+07	506	-8.42695124	0.98382985	0.05135206	6.5
	53207	2.69E+07	506	-8.38545966	0.96754805	0.09284364	3.7
	86289	4.37E+07	506	-8.44177206	0.99095627	0.03653124	6.1
	89696	4.54E+07	506	-8.43294365	0.98231876	0.04535965	6.3
Average	85084	4.31E+07	506	-8.41775273	0.98138456	0.06055057	6.0

SoL$i|\rangle$ - Son of LIQ$Ui|\rangle$

SoL$i|\rangle$

300 Kelvin - Room

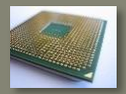
77K-Nitrogen

4K-Helium

.02K-He_3/He_4

CMOS

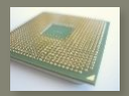
CPU



Memory

Superconducting

CPU

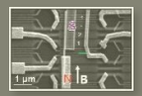


Memory

Quantum

Control

Qubits



STATION Q

Microsoft

Understanding of Classical + Quantum

```
let R (k:int) (q:Qubit) =
    Op.Unitary <-
        let phi     = (2.0*Math.PI)/(pown 2.0 k)
        let phiR    = Math.Cos phi
        let phiI    = Math.Sin phi
        CSMat(2,[(0,0,1.,0.);(1,1,phiR,phiI)])

let CR (k:int) (c:Qubit) (t:Qubit) = Control1 R c k t

let QFT (qs:Qubits) =
    let n          = qs.Length
    for aIdx in n-1..-1..0 do
        let a      = qs.[aIdx]
        H a
        for k in 2..aIdx+1 do
            let c = qs.[aIdx-(k-1)]
            CR k c a
```

- Unitary Operations are defined by their matrices
- Meta-operations (control, adjoint) understand how to re-write the AST (including the classical generators)
- Joint Measurement is a fundamental operation (depends on Machine Model for implementation)
- Target code will be unrolled at the discretion of the target Machine Model
- `Adjoint QFT` will reverse the code order, run the loops backwards and adjoint all the unitary operations

Compilation

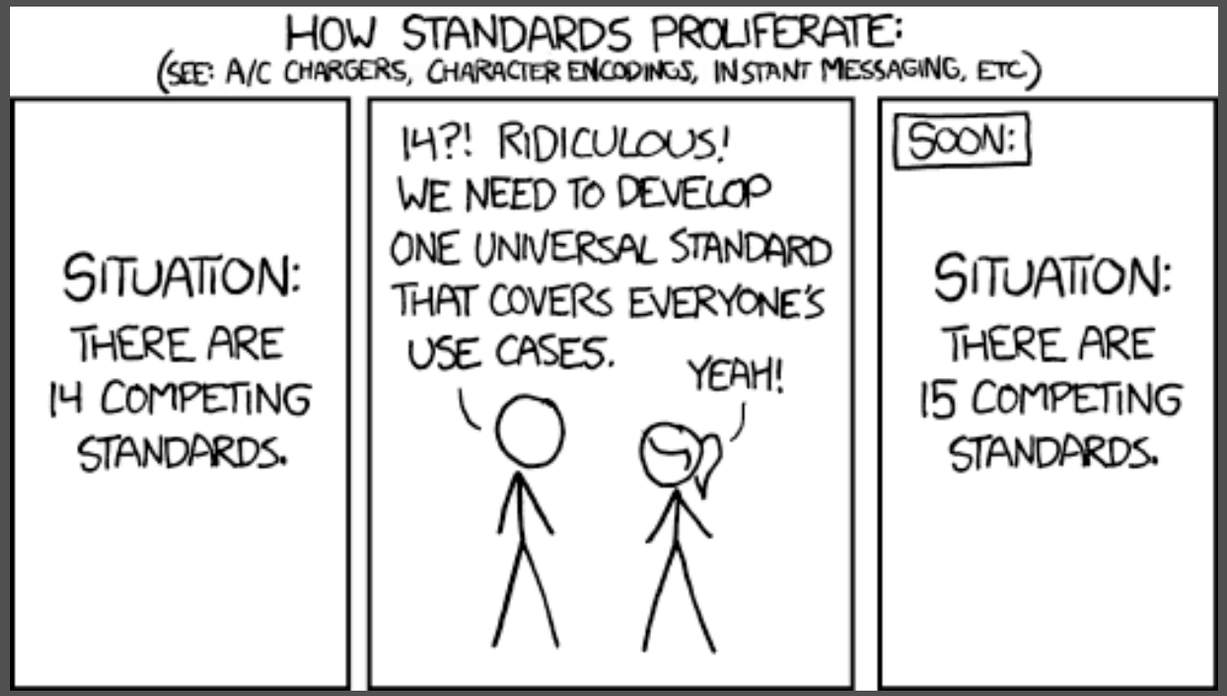
- The compiler front end maps the quantum algorithm to “quantum intermediate language” (QIL)
- The back end rewrites the QIL for execution on actual hardware
- Different quantum computers will require different rewrites of the original QIL
- We have designed and built a layered architecture to support flexible rewriting

```
Push symbol n =
        Property: qs.Length
Push symbol inputSequence =
        Call function Operators.op_RangeStep
          Args:
            Call function Operators.op_Subtraction
              Args:
                Var: n
                Int: 1
            Int: -1
            Int: 0
For each aIdx
        In: inputSequence
        Do:
          Push symbol a =
            Property: qs.Item[aIdx]
          Invoke operation H with args:
            Var: a
          For k
            From: 2
            To:
              Call function Operators.op_Addition
                Args:
                  Var: aIdx
                  Int: 1
            By: 1
            Do:
              Push symbol c =
                Property: qs.Item[(expr)]
              Invoke operation CR with args:
                …
```


QCoDeS

Where we are now:

- Igor (based on Alex Johnson code circa 2002)
- QTLab (Delft Python package circa 2008)
- LabVIEW + Mathematica
- MATLAB (various ad-hoc efforts)
- Some other lightweight Python code



QCoDeS Measurement Syntax

```
data = Loop(c0[-20:20:0.1], 0.1).run()

data2 = Loop(c1[-15:15:1], 0.1).each(
    Task(c0.set, -10),
    qubit1.t1,
    fridge.mc_temp,
    Loop(c0[-15:15:1], 0.01).each(meter.amplitude),
    Task(c0.set, -10),
    Wait(0.1),
    Loop(c2[-10:10:0.2], 0.01),
    Task(c2.set, 5)
).run()
```


More information…

http://StationQ.com

http://StationQ.github.io/Liquid