Software and Architectures for Large-Scale Quantum Computing

Symposium in honor of Paul Benioff's fundamental contributions in quantum information

Fred Chong
Seymour Goodman Professor of Computer Architecture
Department of Computer Science
University of Chicago

IARPA

with Ken Brown, Margaret Martonosi, Diana Franklin, Isaac Chuang,
John Reppy, Ali Javadi Abhari, Jeff Heckey, Daniel Kudrow,
Shruti Patil, Adam Holmes, Alexey Lvov, Sergey Bravyi
(GATech, Princeton, UChicago, MIT, IBM)

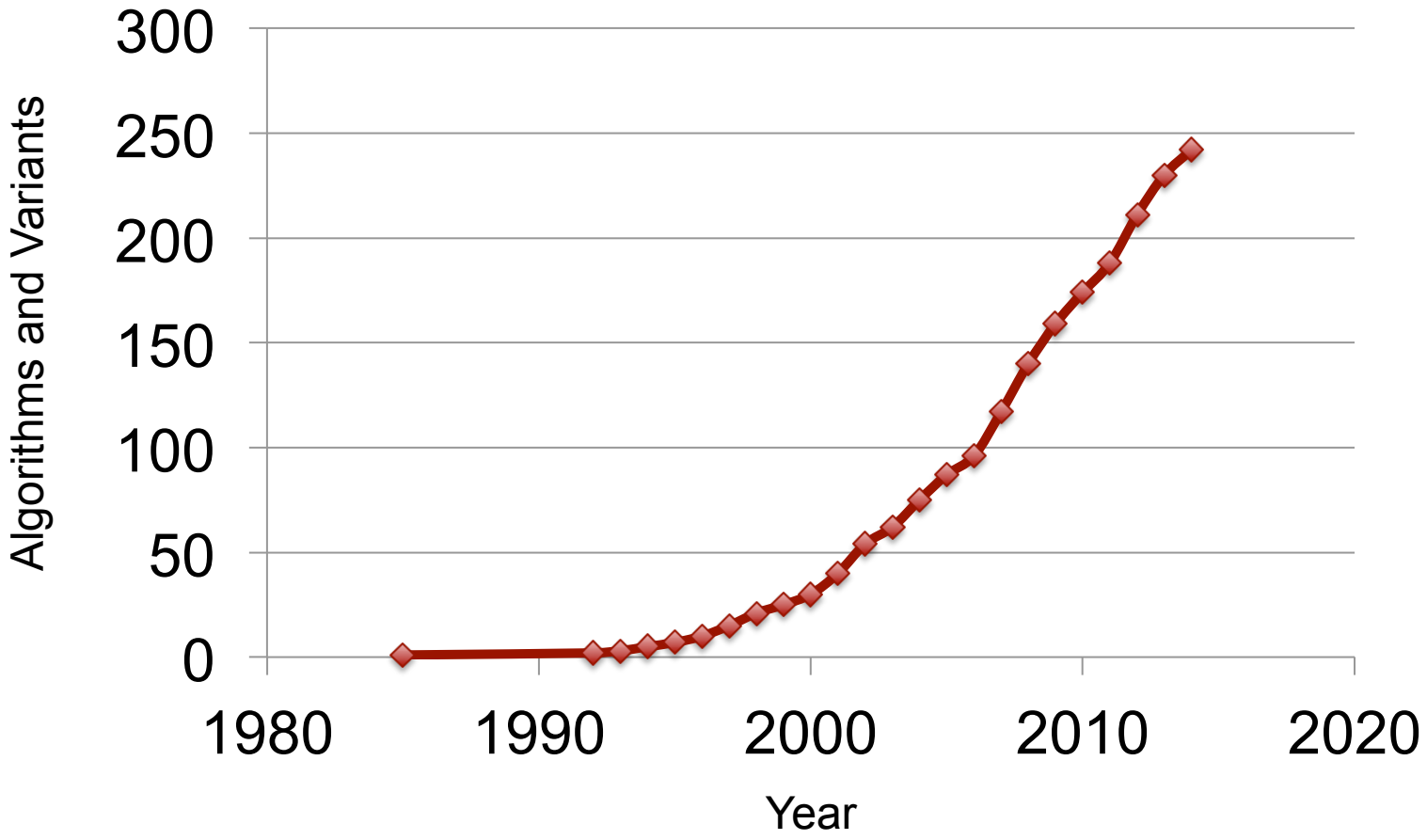
This Talk

- A futuristic systems perspective
  - Scalable architectures to guide device development
  - Software systems to enable pre-machine, large-scale applications work
    - Eg. 10^6 increase in efficiency

      in quantum chemistry (Microsoft)

      [arXiv:1403.1539v2]
  - Apply tools and ideas to 100-qubit machine

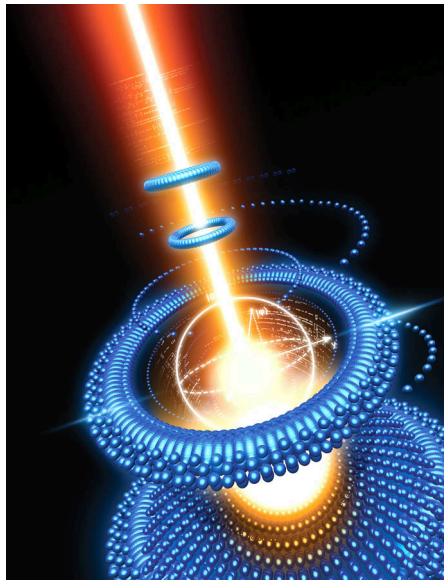
Progress in QC Algorithms



http://math.nist.gov/quantum/zoo/

Outline

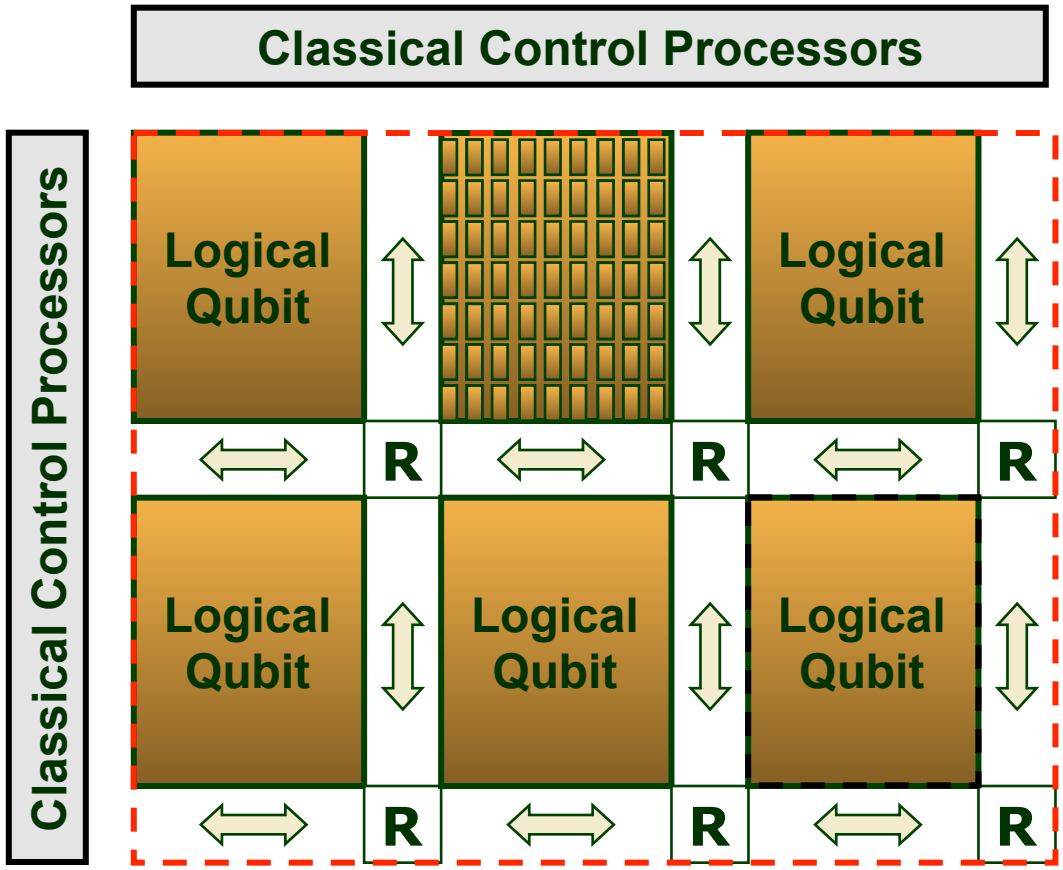
- Lessons Learned
  - Specialization for reliability, parallelism, and performance
  - Managing compiler resources for deep optimization
  - Dynamic code generation for arbitrary rotations
- Future research
  - Retarget SW tools for surface codes
  - Validation of quantum programs
  - What can we do with a 100-qubit machine?



(CACM 2010)

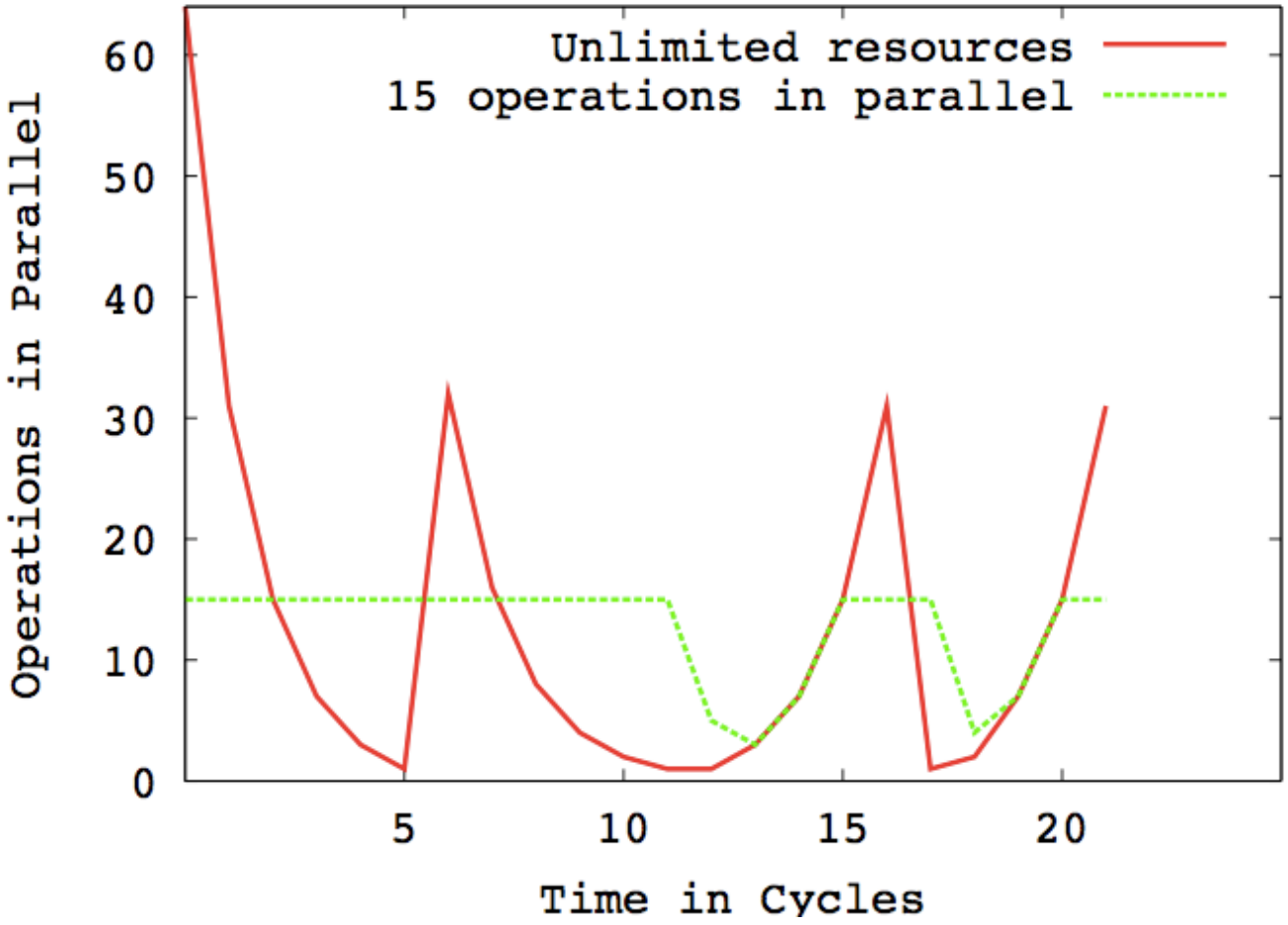
LESSON 1: SPECIALIZATION

"Quantum FPGA"



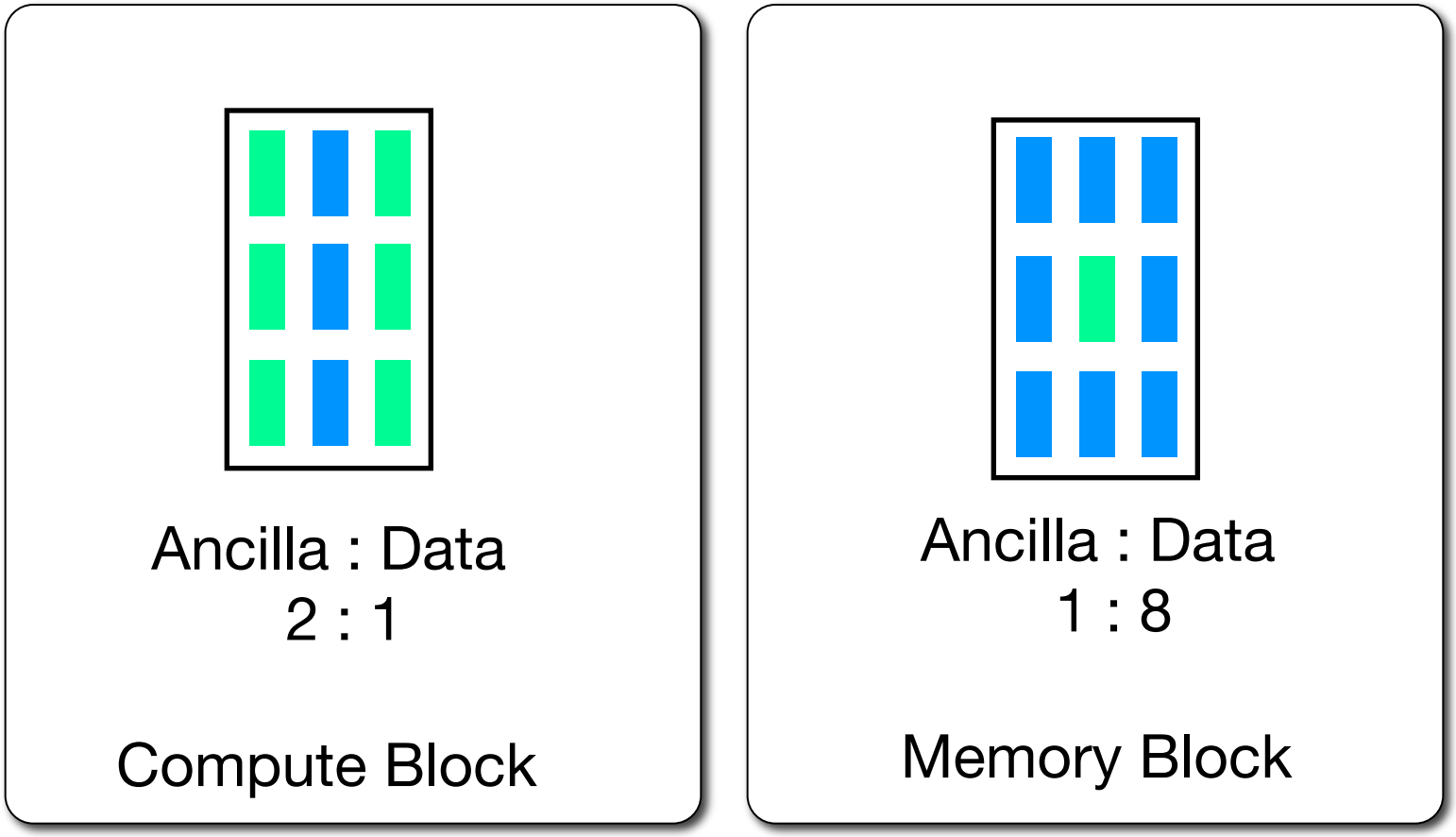
[Metodi et al, Micro05]

Limited Parallelism

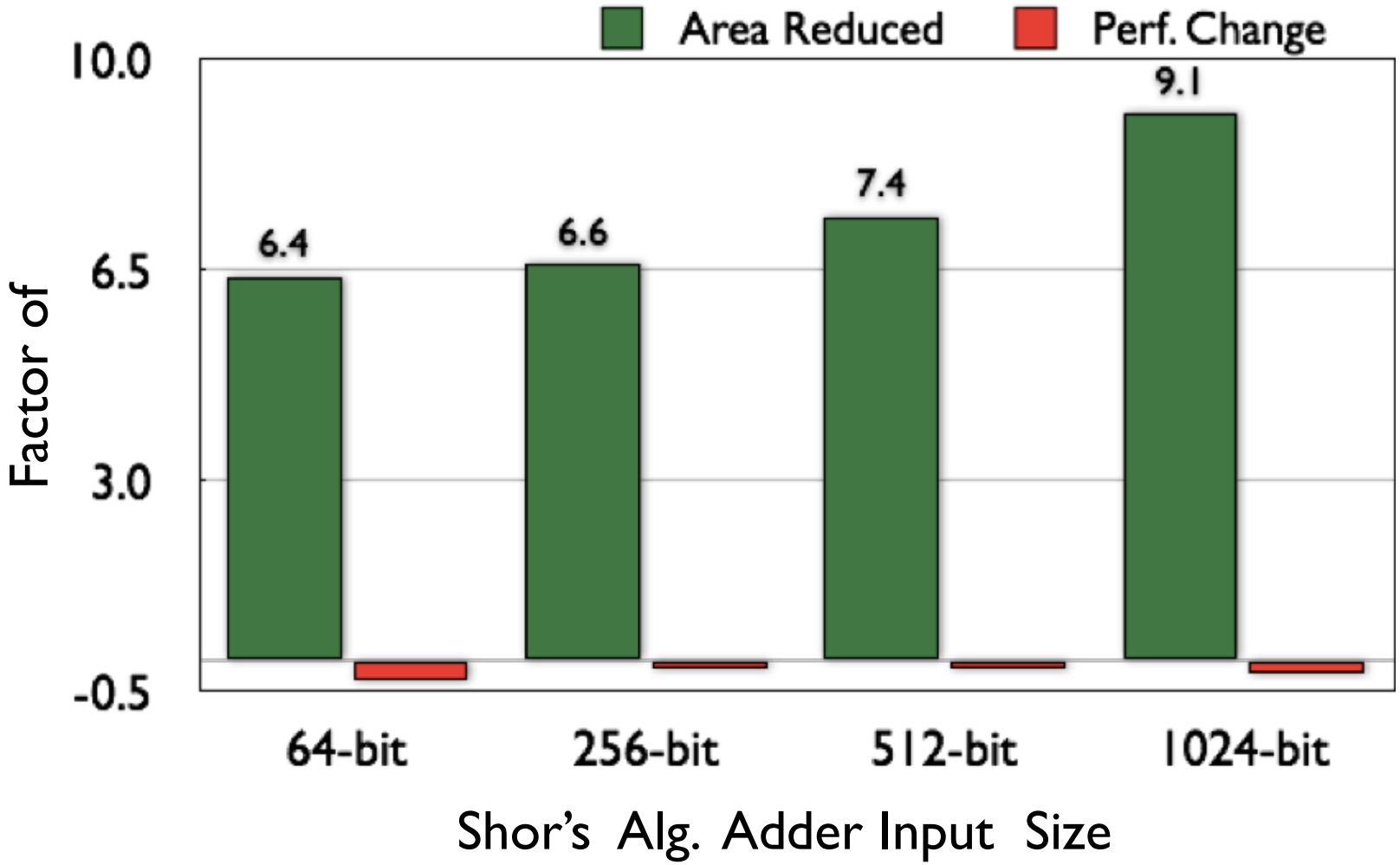


- **Modular Exponentiation Component: The Draper Carry-Lookahead Adder (64-qubit Adder)**

Specialization



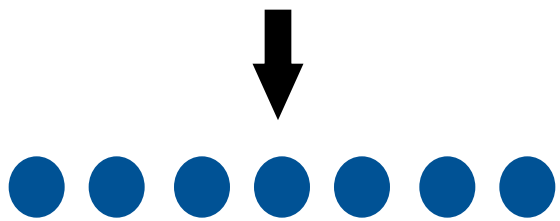
Area Reduced



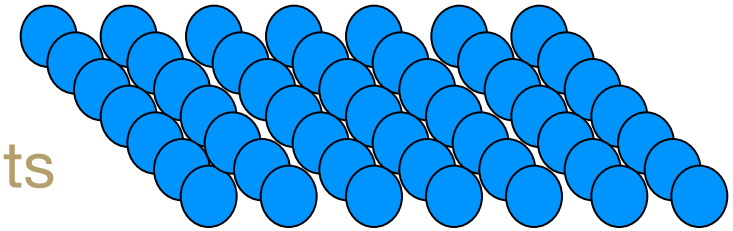
Faster Computation

1 logical qubit

Level 1:
7 physical qubits



Level 2:
49 physical qubits



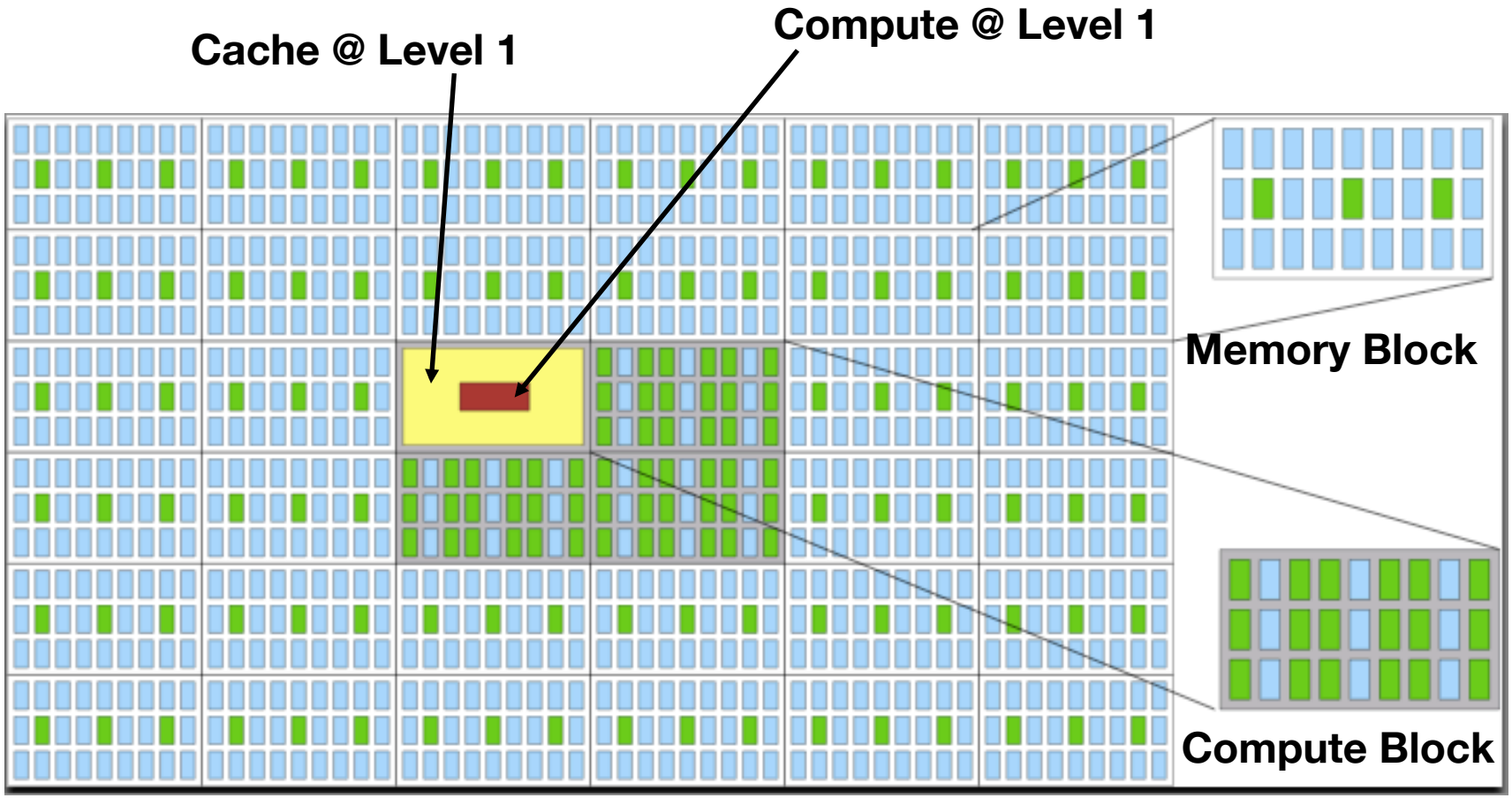
Reliability increases doubly exponentially.

Exponentially slower.

Exponentially greater resources.

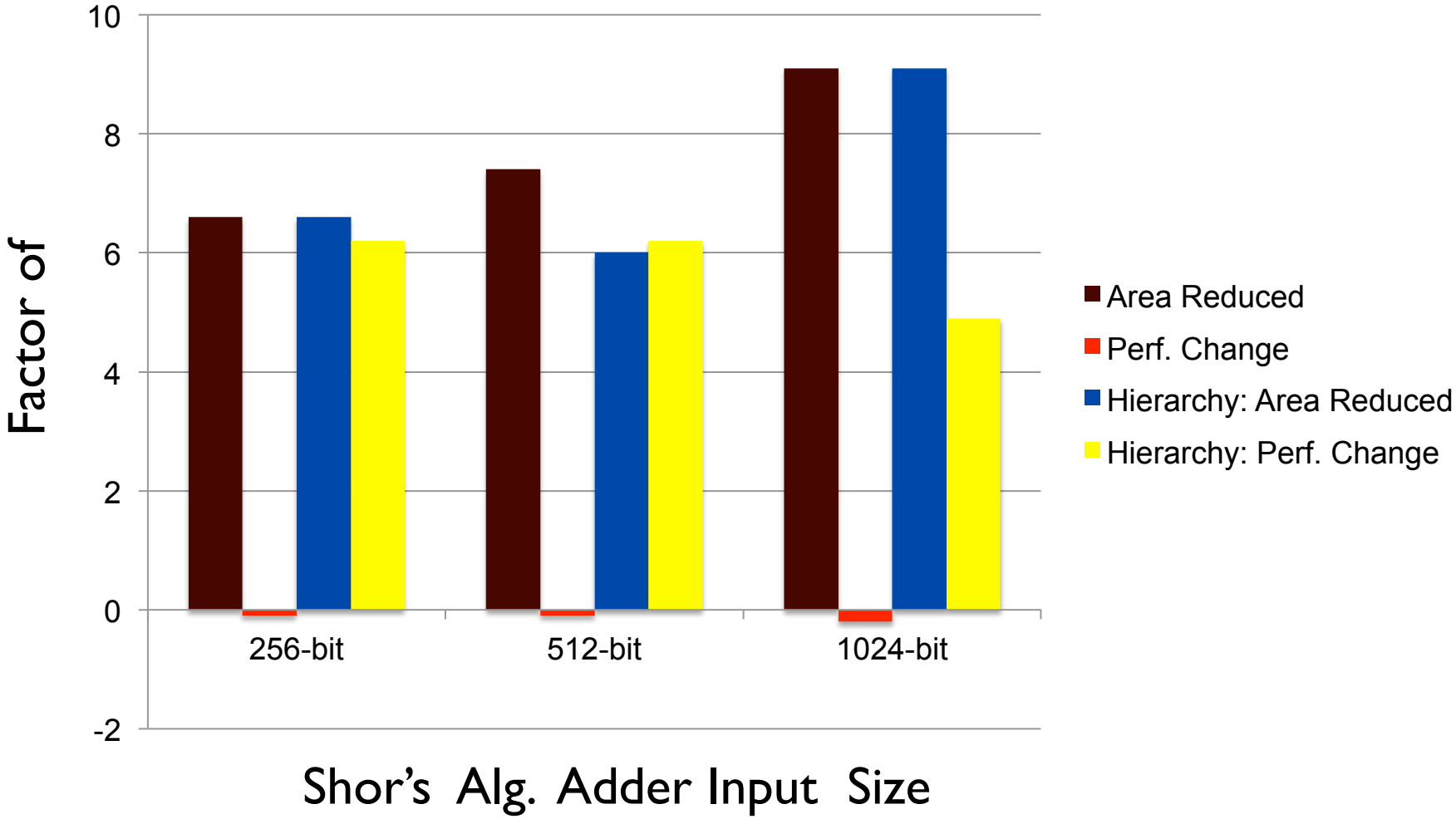
Concatenated Steane Code

Error-Correction Hierarchy



[Thaker et al, ISCA 2006]

Performance Benefits

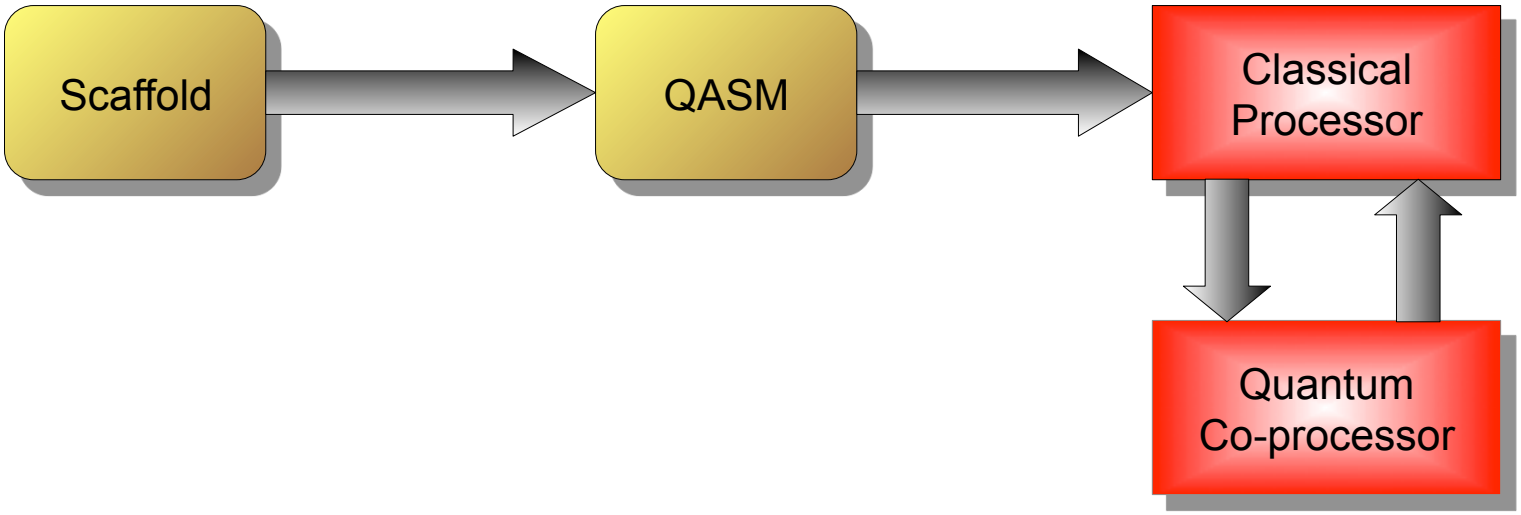


LESSION 2: MANAGING COMPILER RESOURCES

Deep Optimization

- QC similar to circuit synthesis for ASICs
- Program inputs known at compile time
  - Enables compiler optimizations
    - Constant propagation
    - Loop unrolling
- Scarce resources
  - Every qubit and gate is important

Execution Model



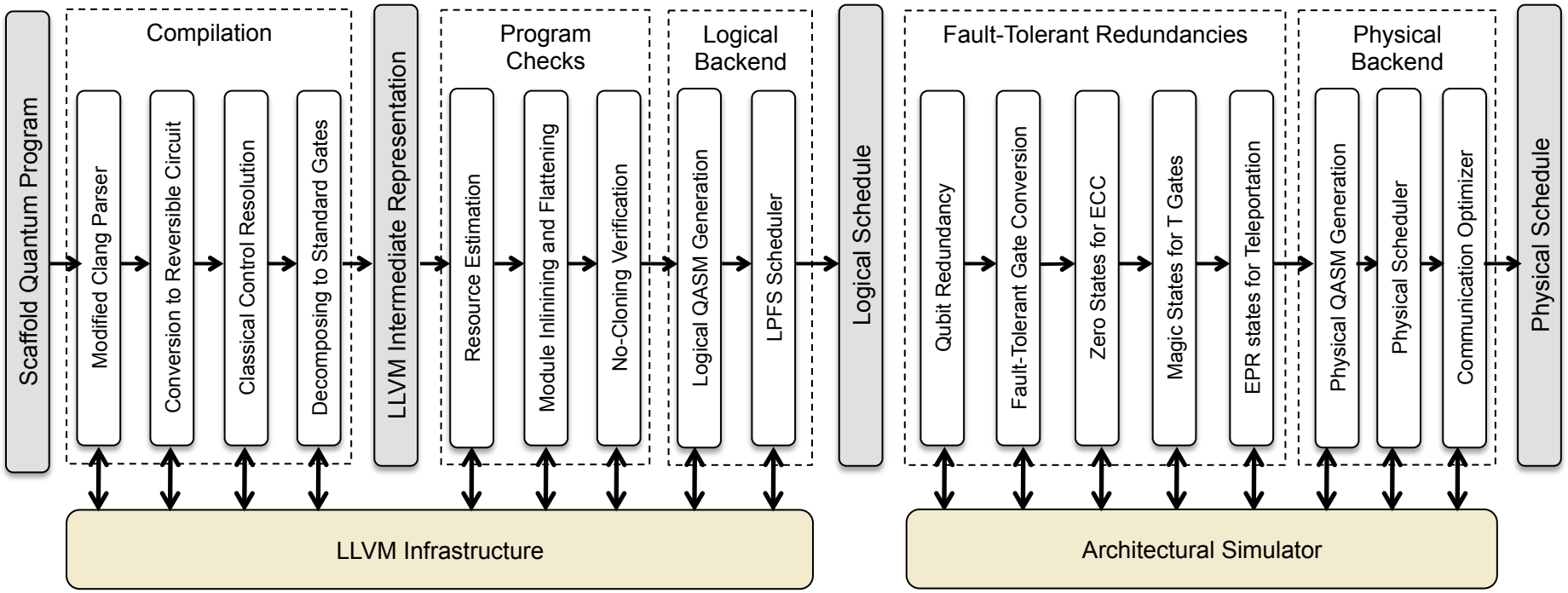
The Scaffold Language and Compiler

- Extended C
  - No pointers
  - Quantum datatypes
  - Extensible gates
  - Parallel loops
  - Reversible logic synthesis for classical functions (includes fixed point arithmetic)

```
1 #include "gates.h"
2 module main ( ) {
3   int i=0;
4   qreg extarget[4];
5   qreg excontrol[4];
6   forall(i=0; i<4; i++) {
7     CNOT(extarget[i],excontrol[i]);
8   }
9 }
```

[Heckey et al, ASPLOS 2015]

Tool Flow



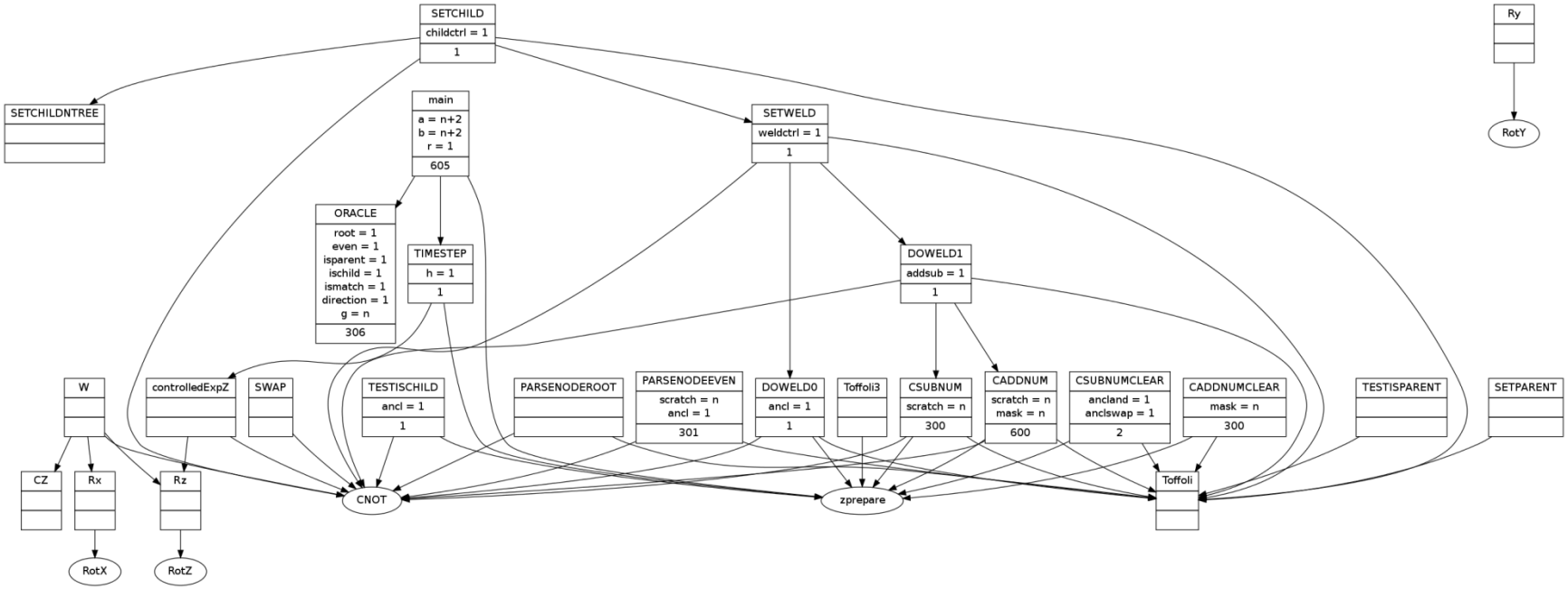
https://github.com/epiqc/ScaffCC

Algorithms in Scaffold

Algorithm	Lines of Code
Boolean Formula	479
Linear Systems	1741
Binary Welded Tree	608
Class Number	226
Triangle Finding	1231
Shortest Vector Problem	539
Ground State Estimation	554
Shor's Algorithm	1055
Grover's Algorithm (invert SHA-1)	388
Ising Model	113

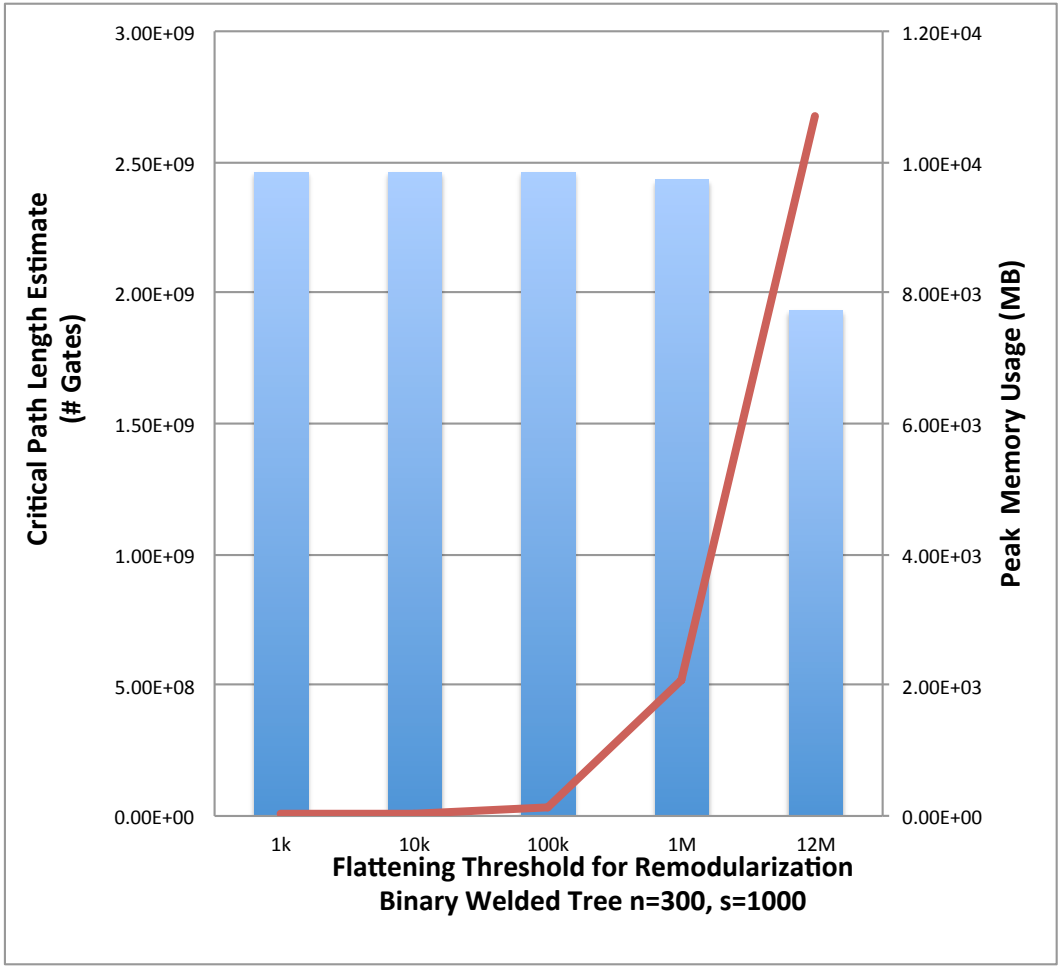
Tool Output: Resource Estimation

- Binary Welded Tree Call Graph
  - Shows quantum resources used at each module
  - Maximum qubits used: 911 (for n=300)

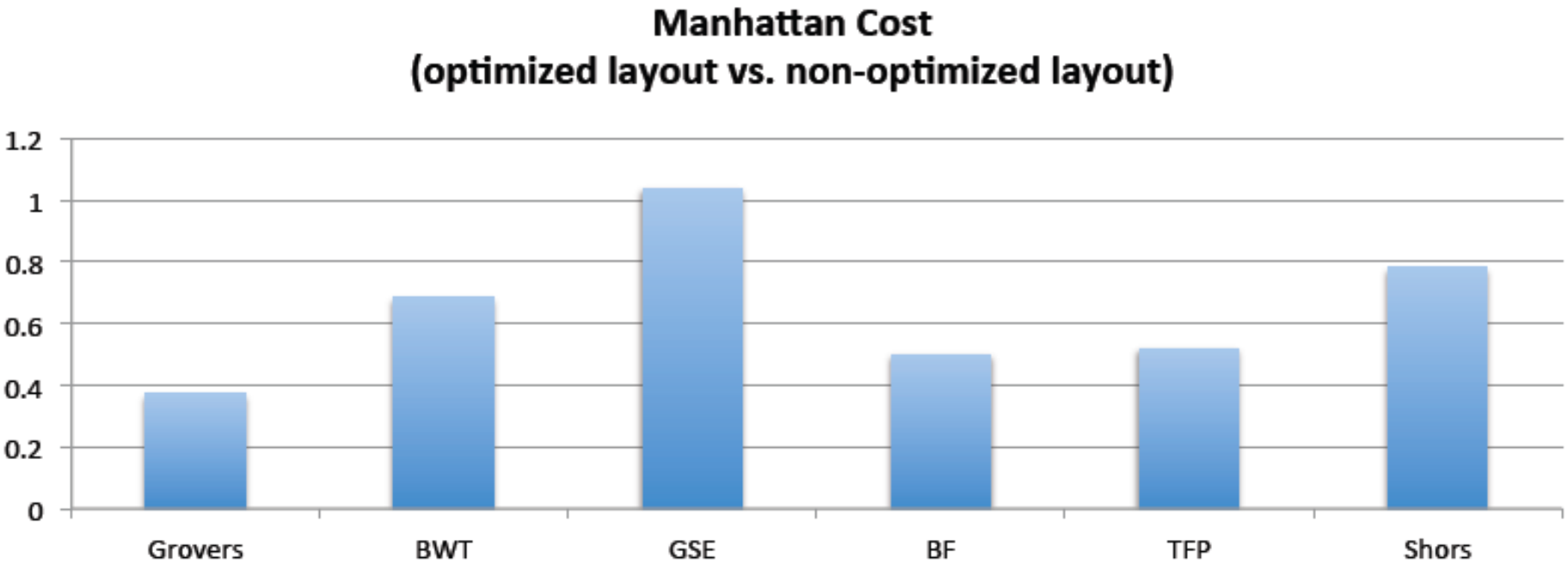


Effect of Remodularization

- Based on resource analysis, flatten modules with size less than a threshold
- Limited by memory on compilation machine



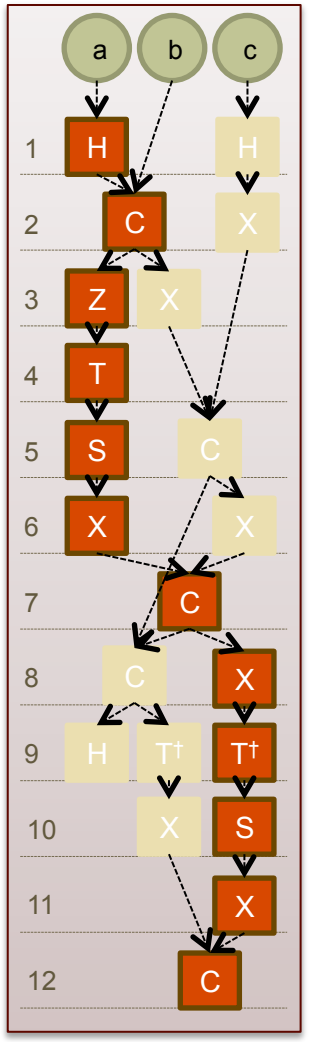
Mapping Qubits



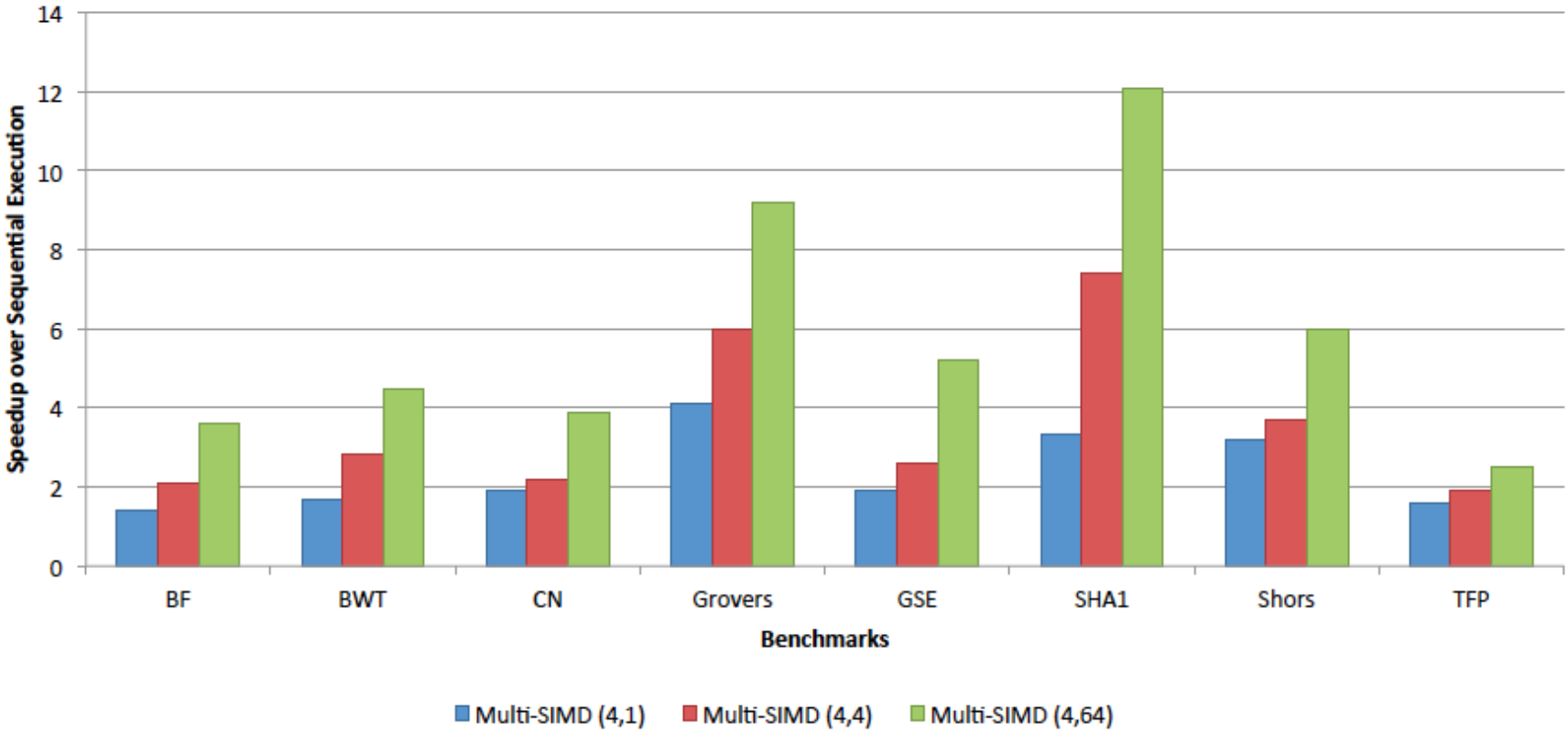
- Modified heuristic graph partitioner
  - based on Metis [Karypis and Kumar, 1995]

Longest Path First Scheduling

Strategy: Minimize qubit motion by assigning long dependence chains to a single compute region, where they can compute locally with little communication.



Tool Output: Speedup Estimates

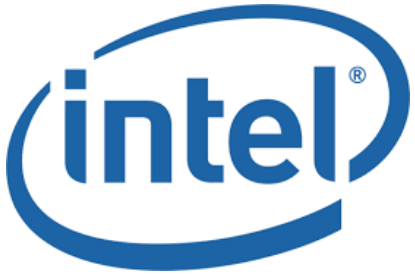


Small-Scale Simulation Path

- Simulation effort at TU Delft
- Takes Scaffold QASM output
- Optimized for Intel supercomputing resources

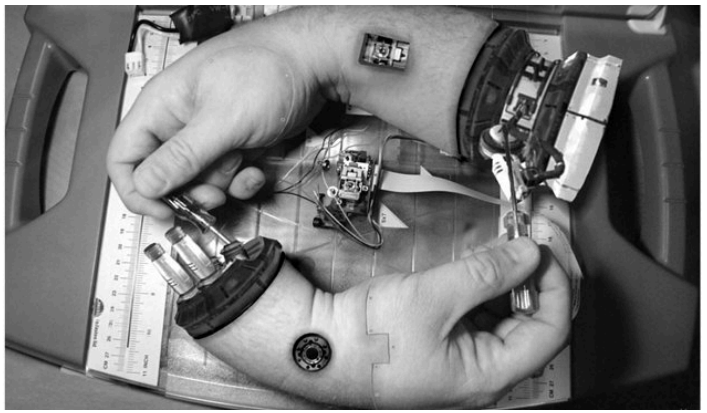
  http://www.xpu-project.net/qx/download.html
- Other closed-source tools: LIQU|d>

  http://github.com/msr-quarc/Liquid



https://github.com/epiqc/ScaffCC

LESSON 3: DYNAMIC CODE GENERATION

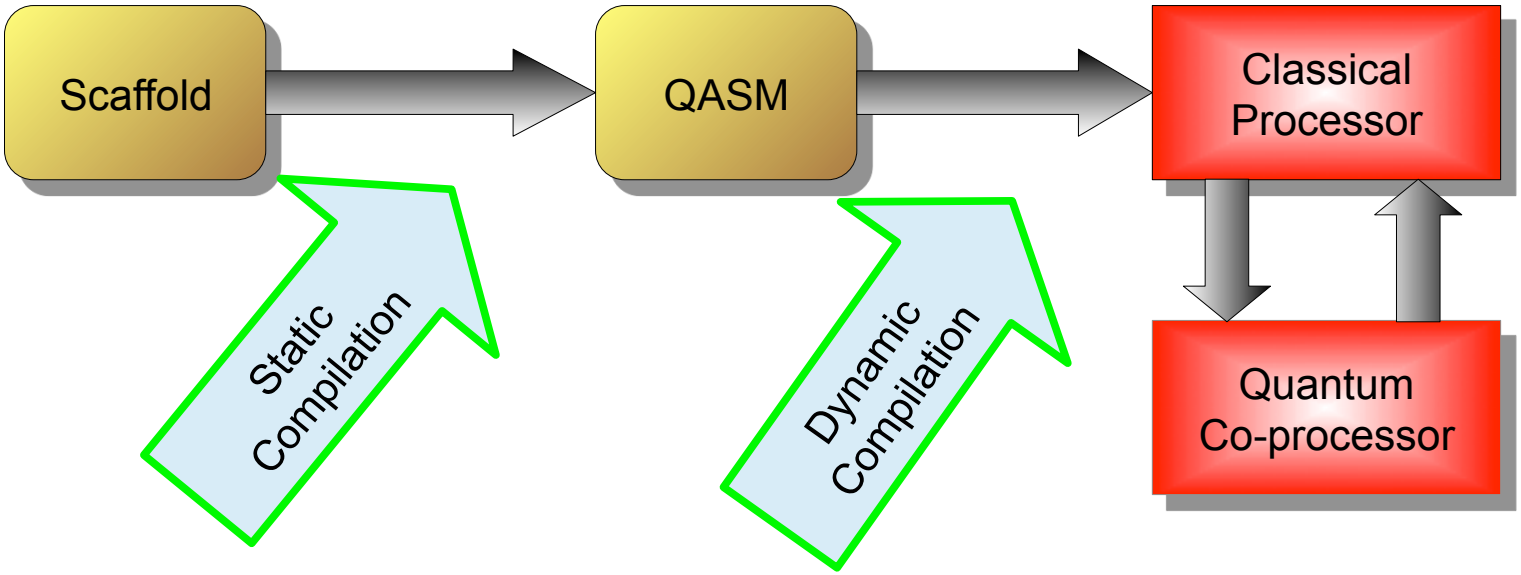


Quantum Code Generation for Arbitrary Rotations

- Arbitrary rotations are important, difficult to compile for, and expensive to execute
- Unique sequence for every distinct rotation
  - Can be 4 TB of code!
- Sometimes need dynamic code generation
  - Rotation angles determined at runtime
  - Large code size

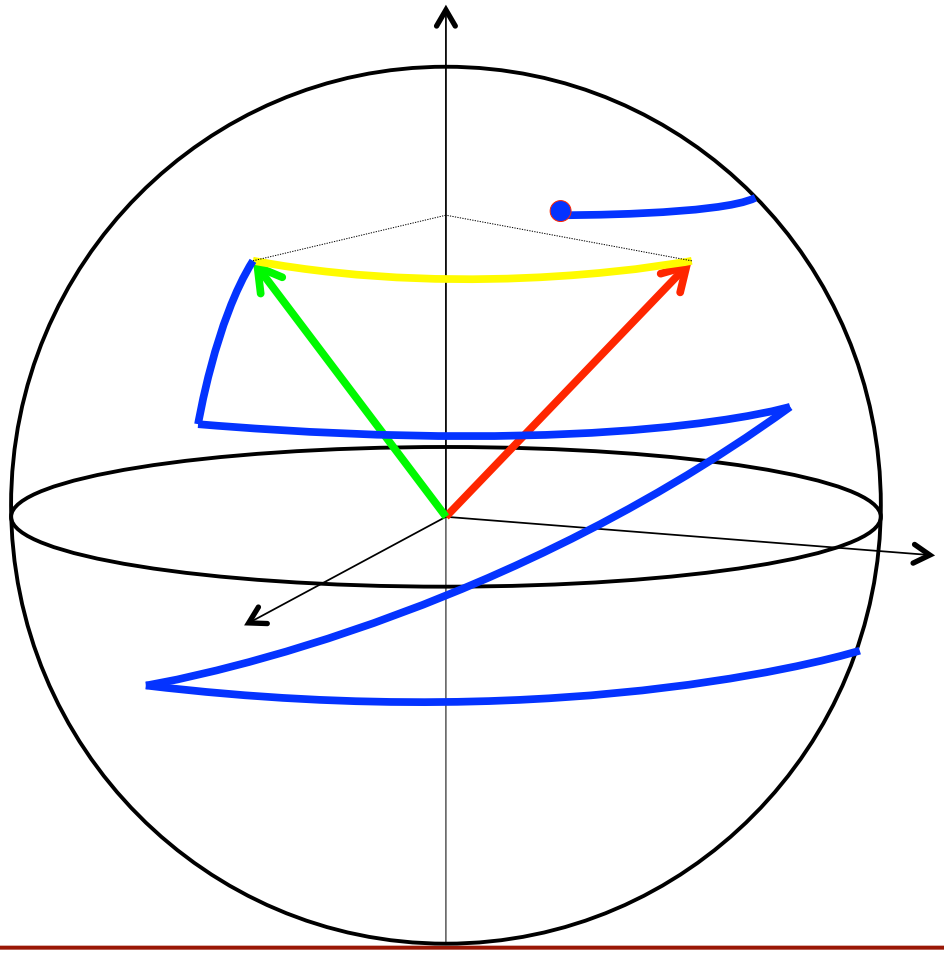
[Kudrow et al, ISCA 2013]

Dynamic Code Generation

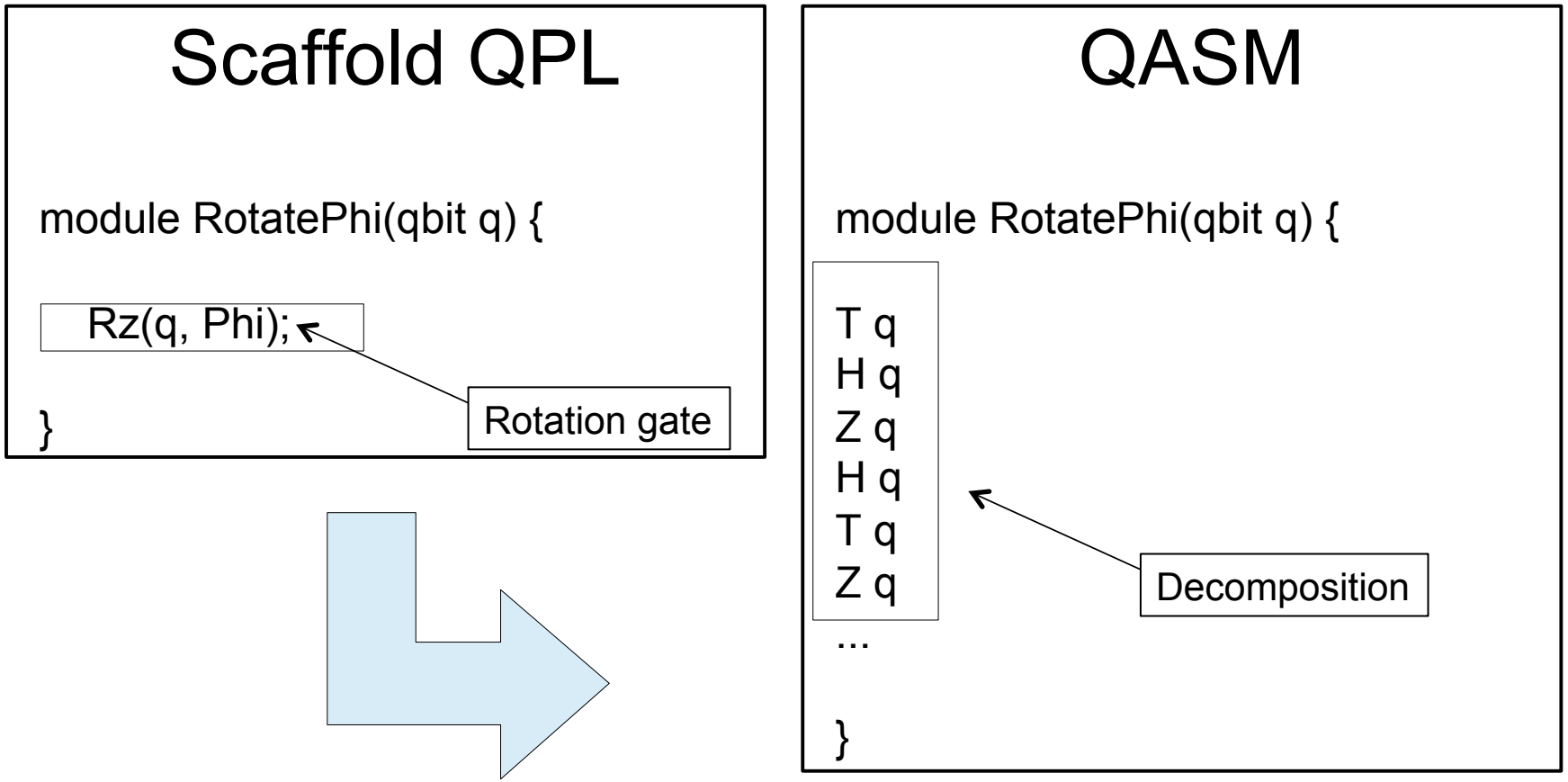


Rotation Decomposition

H gate
T gate
X gate
H gate
$T^{\dagger}$ gate
...



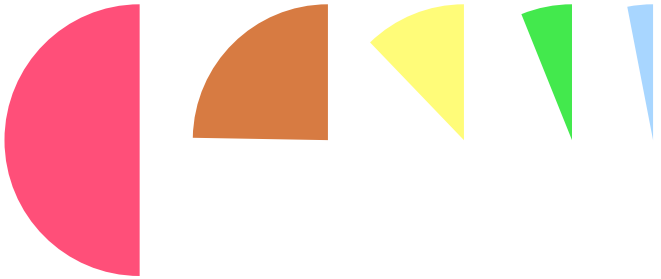
Rotation Decomposition



Precomputed Library

- Example: binary construction

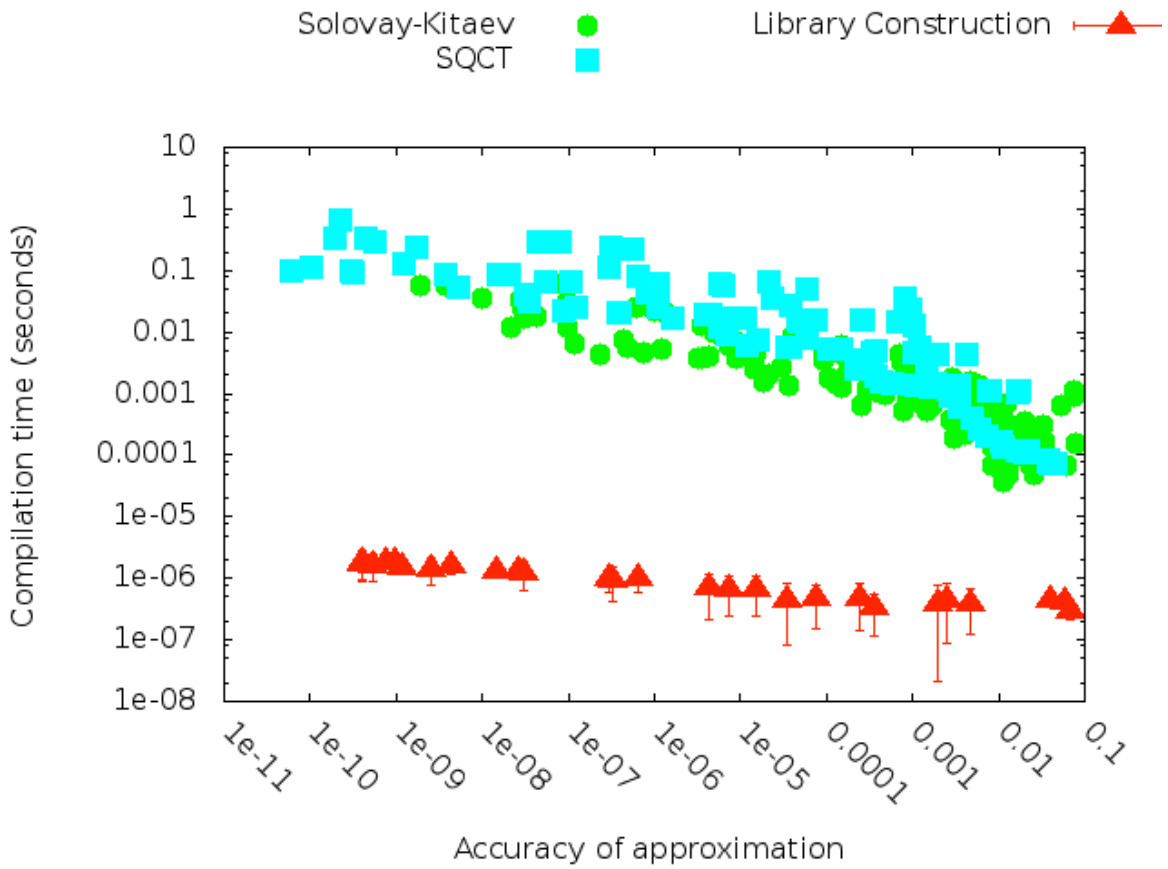
Generate library:



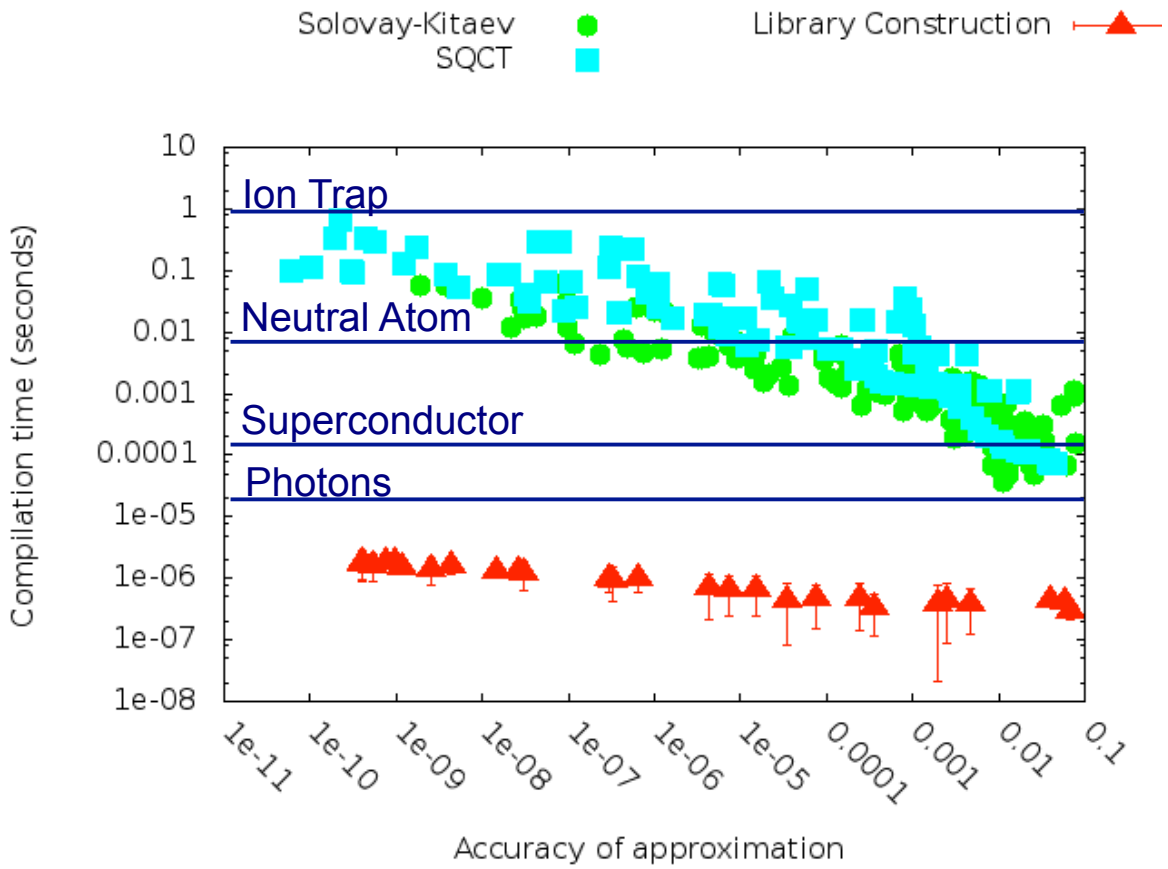
T, H, T, Z, T, Z, H, ...

Concatenate appropriate sequences to approximate desired angle:

Results – Compilation Time



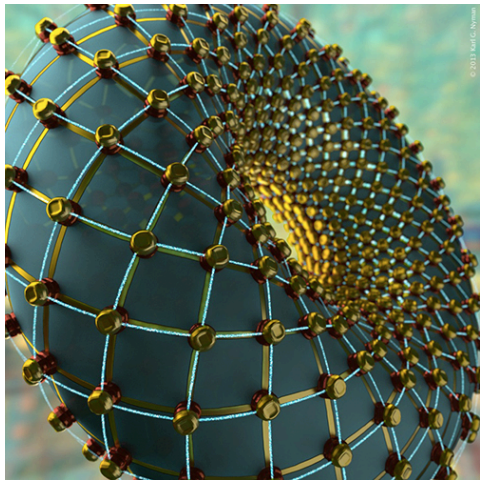
Results – Compilation Time



Dynamic Compilation Summary

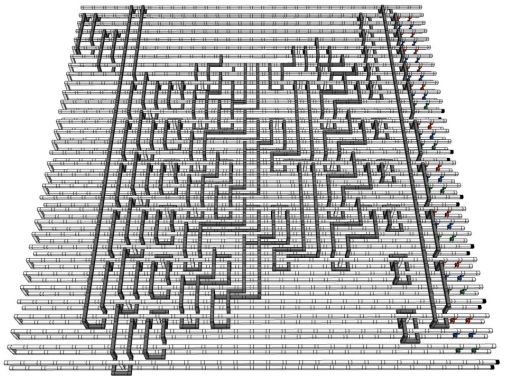
- Up to 100,000X speedup for dynamic compilation with 5X increase in sequence length (T-gate depth)

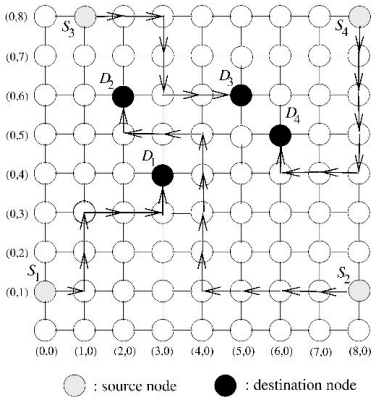
FUTURE WORK 1: TARGETING SURFACE CODES



Surface vs Concatenated Codes

- Less sensitive to communication distance
- Sensitive to braid crossings
  - Serializes communication
  - Qubit mapping for locality is important
- Network routing heuristics for scalable scheduling



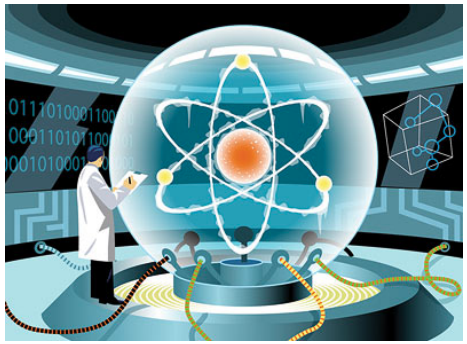


FUTURE WORK 2: PROGRAM CORRECTNESS

How do I know if my QC program is correct?

- Need: Specification language for QC algorithms
- Check implementation against the specification
  - Simulation for small problem sizes (~30 qubits)
  - Symbolic execution for larger problems
  - Type systems
  - Model checking
  - Certified compilation passes
- Compiler checks general quantum properties
  - No-cloning, entanglement, uncomputation
- Checks based on programmer assertions where possible

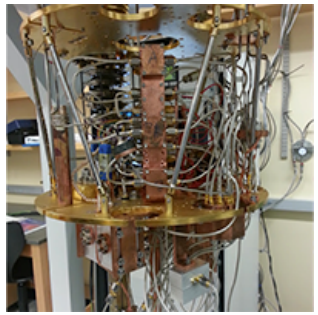
FUTURE WORK 3: ENABLING A PRACTICAL-SCALE QUANTUM COMPUTER (EPIQC)



"Practical" Quantum Computing

- Algorithms and Software for a 100-qubit quantum computer
  - Chong, Reppy, Franklin, Schuster (UChicago), Shor, Farhi, Harrow (MIT), Brown (GATech), Harlow (UCSB)
- Fill the gap between theory and experiment
  - Expose physical effects to software and algorithms
  - Exhaustive optimizations
  - Compiler analysis and partial simulation for correctness

(Schuster Lab)



Summary

- QC is at an exciting time
- Software and architecture can generate key insights and accelerate progress
- With the right models and abstractions, classical techniques can have significant impact

https://github.com/epiqc/ScaffCC

http://people.cs.uchicago.edu/~ftchong