Optimizing the layout and error properties of quantum circuits

Professor John Kubiatowicz
University of California at Berkeley

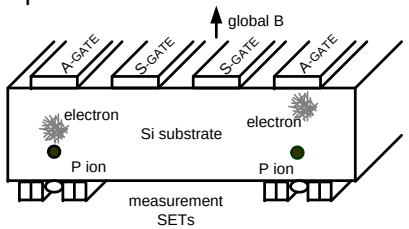
September 28th, 2012
kubitron@cs.berkeley.edu
http://qarc.cs.berkeley.edu/

Quantum Circuits are Big!

- Some recent (naïve?) estimates for Ground-State Estimation (Level 3 Steane code):
  - 209 logical qubits × 343 (EC) = 71687 data qubits
  - Total operations: 10^{11} to 10^{17} (depending on type)
  - 10^{17} T gates × 117,000 ancillas/T gate = 10^{22} ancillas
  - 5×10^{26} Operations for SWAP (communication)
  - And on…
- Shor's Algorithm for factoring?
  - 5×10^{5} or more data qubits
  - 1.5×10^{15} operations (or more)
- How can you possibly investigate such circuits?
  - This is the realm of *Computer Architecture* and *Computer Aided Design (CAD)*

Simple example of Why Architecture Studies are Important (2003)

- Consider Kane-style Quantum Computing Datapath
  - Qubits are embedded P^+ impurities in silicon substrate
  - Manipulate Qubit state by manipulating hyperfine interaction with electrodes above embedded impurities
- Obviously, important to have an efficient *wire*
  - For Kane-style technology need sequence of SWAPs to communicate quantum state
  - So – our group tried to figure out what involved in providing wire



- Results:
  - Swapping control circuit involves complex pulse sequence between every pair of embedded Ions
  - We designed a local circuit that could swap two Qubits (at < 4°K)
  - Area taken up by *control* was > 150 x area taken by bits!
- Conclusion: must at least have a practical WIRE!
  - Not clear that this technology meets basic constraint

Pushing Limits

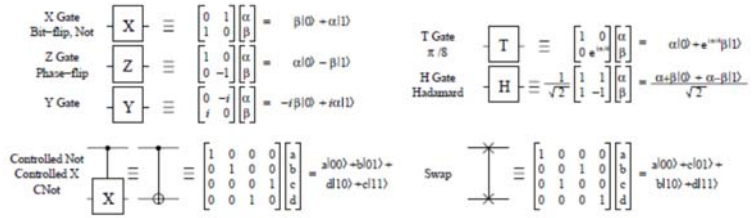
- Very interesting problems happen at scale!
  - Small circuits become Computer Architecture
    - Modular design
    - Pipelining
    - Communication Infrastructure
  - Direct analogies to classical chip design apply
    - The physical organization of components matters
    - "Wires are expensive, adders are not"?
- Important Focus Areas for the future:
  - Languages for Describing Quantum Algorithms
  - Optimal partitioning and layout
  - Global communication scheduling
  - Layout-driven error correction

Expressing Quantum Algorithms

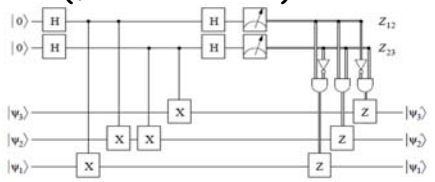
How to express Circuits/Algorithms?

- Graphically: Schematic Capture Systems
  - Several of these have been built
- QASM: the quantum assembly language
  - Primitives for defining single Qubits, Gates
- C-like languages
  - Scaffold: some abstraction, modules, fixed loops
- Embedded languages
  - Use languages such as Scala or Ruby to build Domain Specific Language (DSL) for quantum circuits
  - Can build up circuit by overriding basic operators
  - Can introduce a "Reverse" operator to turn classical circuits into reversible quantum ones

Quantum Circuit Model



- Quantum Circuit model – graphical representation
  - Time Flows from left to right
  - Single Wires: persistent Qubits, Double Wires: classical bits
    - Qubit – coherent combination of 0 and 1: $\psi = \alpha|0\rangle + \beta|1\rangle$
  - Universal gate set: Sufficient to form all unitary transformations
- Example: Syndrome Measurement (for 3-bit code)
  - Measurement (meter symbol) produces classical bits
- Quantum CAD
  - Circuit expressed as netlist
  - Computer manpulated circuits and implementations



Higher-Level Language: Chisel

- Scala-based language for digital circuit design
  - High-level functional descriptions of circuits as input
  - Many outputs: for instance direct production on Verilog
  - Used in design of new advanced RISC pipeline
- Features
  - High-level abstraction
  - Hierarchical design
  - Abstractions build up circuit (netlist)
- Inner-Product FIR Digital Filter: $y[t] = \sum_j w_j * x_j[t-j]$

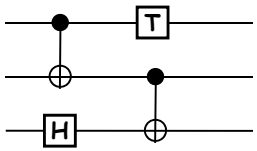
```
def innerProductFIR[T <: Num] (w: Array[Int], x: T) =
  foldR(Range(0, w.length).map(i => Num(w(i))
                                * delay(x, i)), _ + _)

def delay[T <: Bits](x: T, n: Int): T =
  if (n == 0) x else Reg(delay(x, n - 1))

def foldR[T <: Bits] (x: Seq[T], f: (T, T) => T): T =
  if (x.length == 1) x(0) else f(x(0),
                                 foldR(x.slice(1, x.length), f))
```

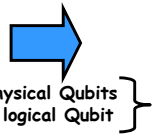
Quantum Chisel

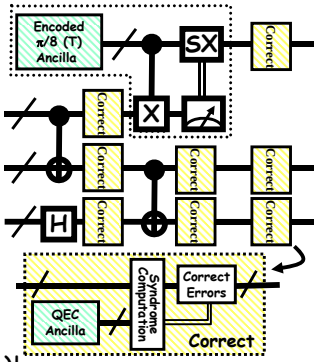
- Simple additions to Chisel Code base
  - Addition of Classical ⇒ Quantum translation
    - Produce Ancilla, UseToffoli Gates, CNots, etc
    - Reverse Logic to automagically reverse netlists and produce reversible output
    - State machine transformation (using "shift registers" to keep extra state when needed)
  - Because of the way Chisel constructed, can be *below the level of syntax (DSL) seen by programmer*
    - With possible exception of explicit REVERSE operator
- Goal? Take classical circuits designed in Chisel and produce quantum equivalents
  - Adders, Multipliers
  - Floating-Point processors
- Output: Quantum Assembly (QASM)
  - Input to other tools!

One Sticky Issue: Error Correction

Quantum ECC (Concatenated Codes)

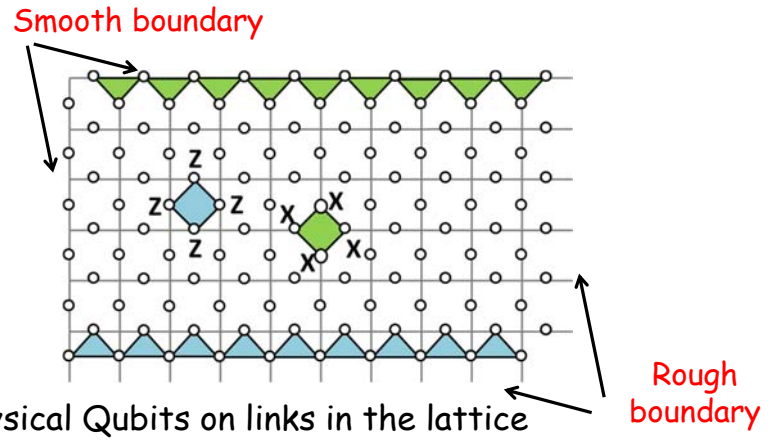






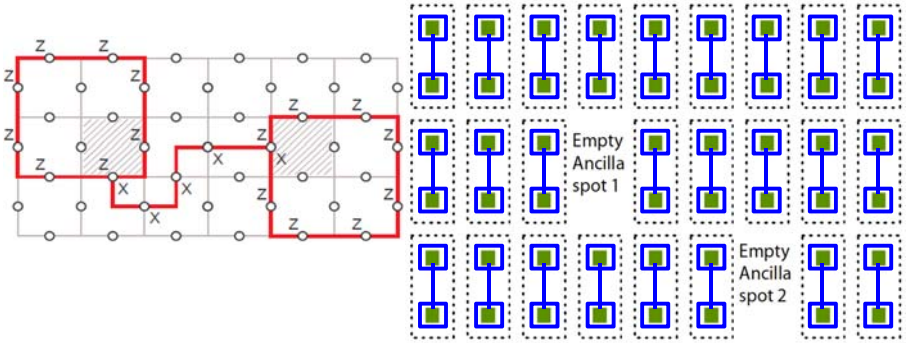
- Quantum State Fragile ⇒ encode all Qubits
  - Uses many resources: e.g. 343 physical Qubits/logical Qubit)!
- Need to handle operations (fault-tolerantly)
  - Some set of gates are simply "transversal": identical operation on each bit
  - Others (like T gate) much more complex (non-transversal)
- Finally, need to perform periodic error correction
  - Correct after every(?): Gate, Movement, Long Idle Period
  - Correction reducing entropy ⇒ Consumes Ancilla bits

Topological (Surface) Quantum ECC



- Physical Qubits on links in the lattice
- Continuous Measurement and Correction
  - Measuring stabilizers (groups of 4) yields error syndromes
  - Optimizations around the decoding algorithm and frequency of measurement

Computation with Topological Codes



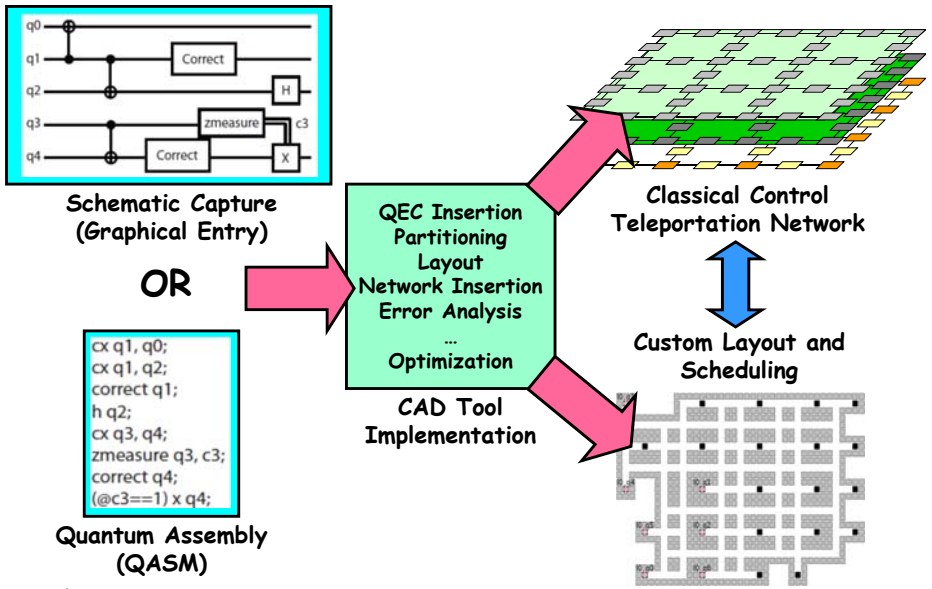
- Each logical Qubit represented by a pair of holes
- Layout for Large Algorithm: Tile Lattice with paired holes
- CNOT: move a smooth hole around a rough one
  - Complications: may need to transform a smooth hole into a rough one before performing CNOT
  - Rules for how to move holes (grow and shrink them)
- Again: Some gates easy, some not (Once again, T is messy)

Moving to the Realm of Quantum Computer Aided Design

Need for CAD: More than just Size

- Data locality:
  - Where qubits "live" and how they move can make or break the ability of a quantum circuit to function:
    - Movement carries risk and consumes time
    - Ancilla must be created close to where used
    - Communication must be minimized through routing optimization
- Customized (optimal?) data movement ⇒ customized channel structure/quantum data path
  - One-size fits all topology not necessarily the best
- Parallelism:
  - How to exploit parallelism in dataflow graph
    - Partitioning and scheduling algorithms
  - Area-Time tradeoff in Ancilla generation
  - Customized circuits for pre-computing non-transversal Ancilla reuse?
- Error Correction:
  - One-size fits all probably not desirable
    - Adapt level of encoding in circuit-dependent way
  - Corrections after every operation may not be necessary

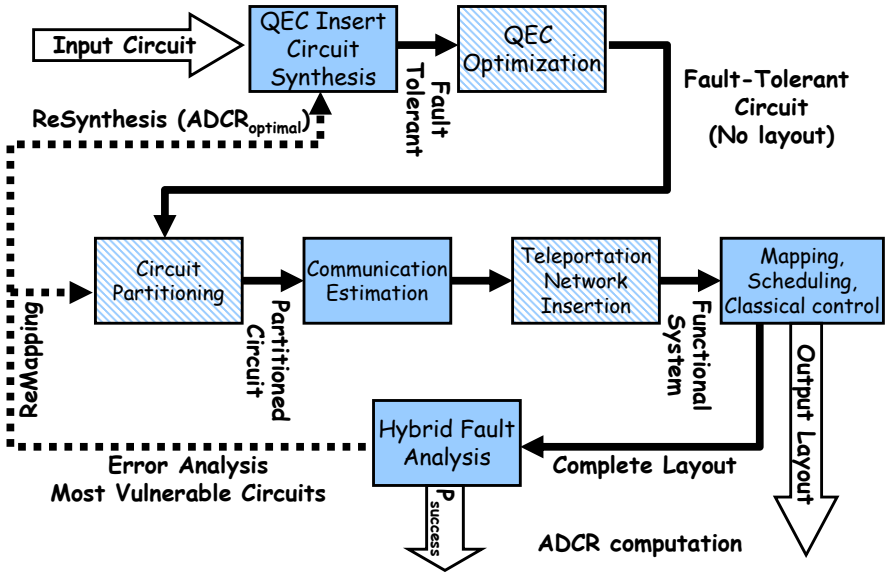
Quadence Design Tool



Important Measurement Metrics

- Traditional CAD Metrics:
  - Area
    - What is the total area of a circuit?
    - Measured in macroblocks (ultimately μm^2 or similar)
  - Latency ($Latency_{single}$)
    - What is the total latency to compute circuit *once*
    - Measured in seconds (or μs)
  - Probability of Success ($P_{success}$)
    - Not common metric for classical circuits
    - Account for occurrence of errors and error correction
- Quantum Circuit Metric: ADCR
  - Area-Delay to Correct Result: Probabilistic Area-Delay metric
  - $ADCR = Area \times E(Latency) = \frac{Area \times Latency_{single}}{P_{success}}$
  - $ADCR_{optimal}$: Best ADCR over all configurations
- Optimization potential: Equipotential designs
  - Trade Area for lower latency
  - Trade lower probability of success for lower latency

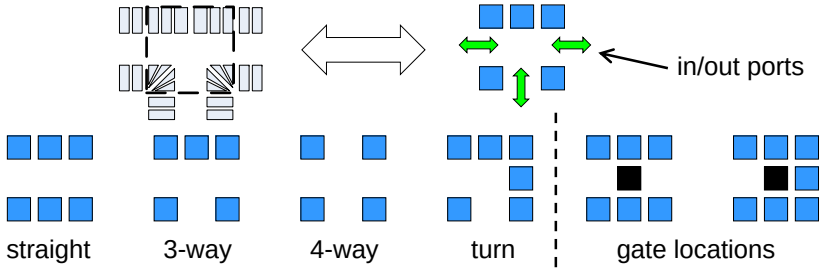
Quantum CAD flow



Optimizing Ancilla and Layout

An Abstraction of Ion Traps

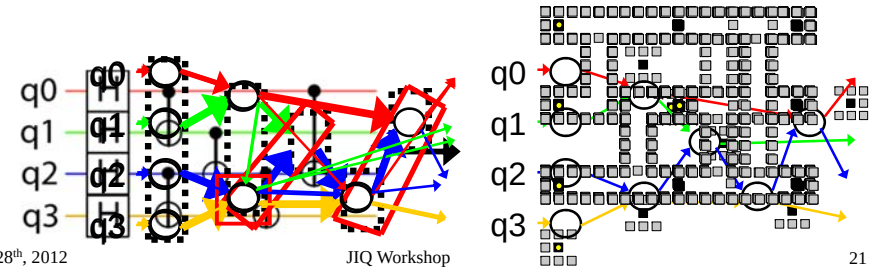
- *Basic block* abstraction: Simplify Layout



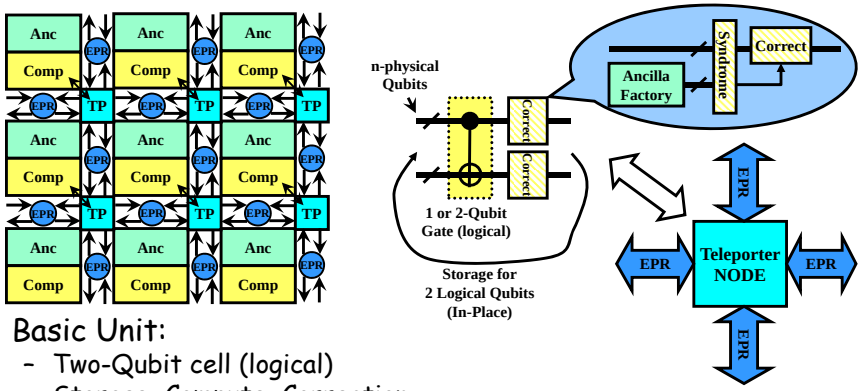
- Evaluation of layout through simulation
  - Movement of ions can be done classically
  - Yields Computation Time and Probability of Success
- Simple Error Model: Depolarizing Errors
  - Errors for every Gate Operation and Unit of Waiting
  - Ballistic Movement Error: Two error Models
    1. Every Hop/Turn has probability of error
    2. Only Accelerations cause error

Example Place and Route Heuristic: Collapsed Dataflow

- Gate locations placed in dataflow order
  - Qubits flow left to right
  - Initial dataflow geometry folded and sorted
  - Channels routed to reflect dataflow edges
- Too many gate locations, collapse dataflow
  - Using scheduler feedback, identify latency critical edges
  - Merge critical node pairs
  - Reroute channels
- Dataflow mapping allows pipelining of computation!



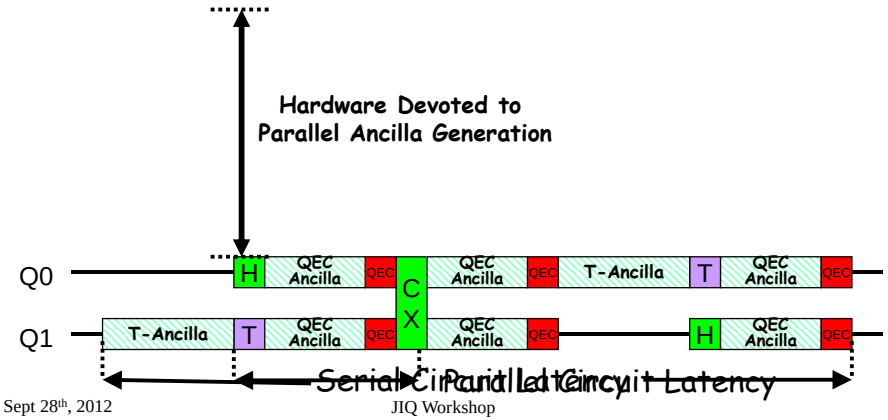
Quantum Logic Array (QLA)



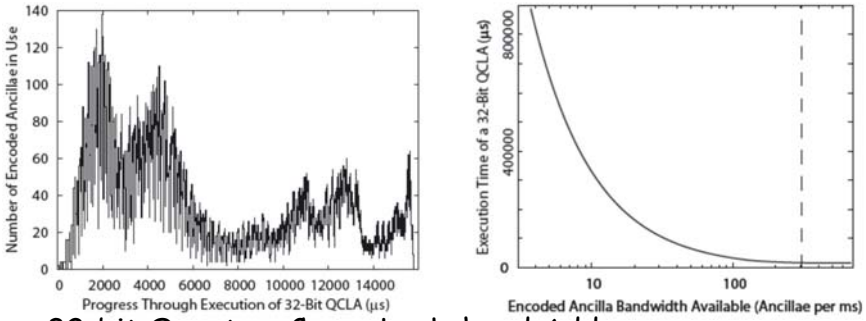
- Basic Unit:
  - Two-Qubit cell (logical)
  - Storage, Compute, Correction
- Connect Units with Teleporters
  - Probably in mesh topology, but details never entirely clear from original papers
- First Serious (Large-scale) Organization (2005)
  - Tzvetan S. Metodi, Darshan Thaker, Andrew W. Cross, Frederic T. Chong, and Isaac L. Chuang

Running Circuit at "Speed of Data"

- Often, Ancilla qubits are independent of data
  - Preparation may be pulled offline
  - Very clear Area/Delay tradeoff:
    - Suggests Automatic Tradeoffs (CAD Tool)
- Ancilla qubits should be ready "just in time" to avoid ancilla decoherence from idleness

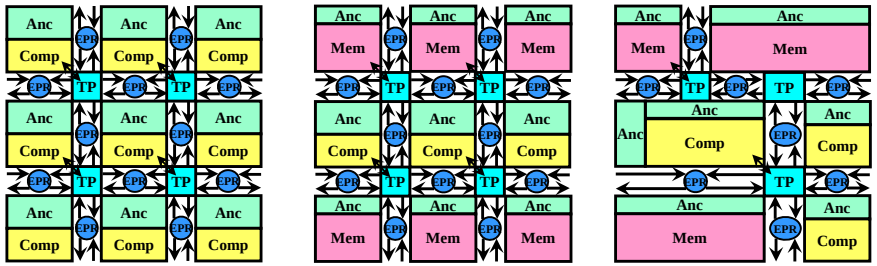


How much Ancilla Bandwidth Needed?



- 32-bit Quantum Carry-Lookahead Adder
  - Ancilla use very uneven (e.g. zero and T ancilla)
  - Performance is flat at high end of ancilla generation bandwidth
    - Can back off 10% and save orders of magnitude in area
- Many bits idle at any one time
  - Need only enough ancilla to maintain state for these bits
  - Many not need to frequently correct idle errors
- Conclusion: makes sense to compute ancilla requirements and share area devoted to ancilla generation
- Can precompute ancilla for non-transverse gates!

Tiled Quantum Datapaths

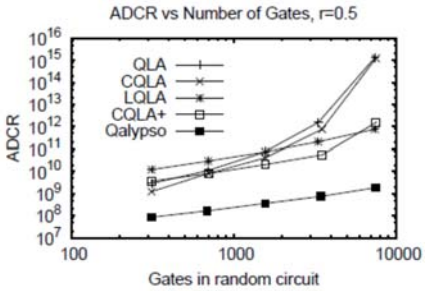


Previous: QLA, LQLA Previous: CQLA, CQLA+ Our Group: Qalypso

- Several Different Datapaths mappable by our CAD flow
  - Variations include hand-tuned Ancilla generators/factories
- Memory: storage for state that doesn't move much
  - Less/different requirements for Ancilla
  - Original CQLA paper used different QEC encoding
- Automatic mapping must:
  - Partition circuit among compute and memory regions
  - Allocate Ancilla resources to match demand (at knee of curve)
  - Configure and insert teleportation network

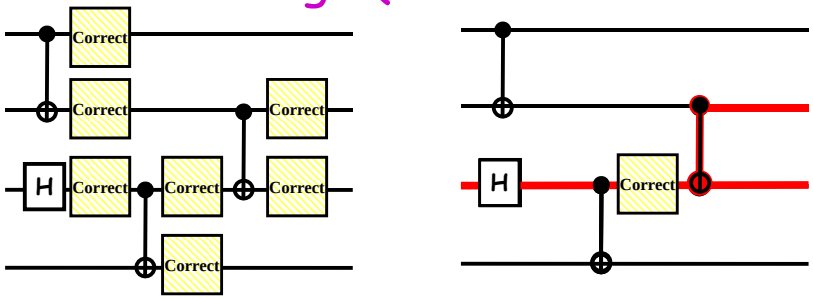
Which Datapath is Best?

- Random Circuit Generation
  - Splitting factor (r): measures connectivity of the circuit
    - Related to Rent's factor
- Qalypso clear winner
  - 4x lower latency than LQLA
  - 2x smaller area than CQLA+
- Why Qalypso does well:
  - Shared, matched ancilla factories
  - Automatic network sizing (rather than fixed teleportation)
  - Automatic Identification of Idle Qubits (memory)
- LQLA and CQLA+ perform close second
  - Original supplemented with better ancilla generators, automatic network sizing, and Idle Qubit identification
  - Original QLA and CQLA do very poorly for large circuits



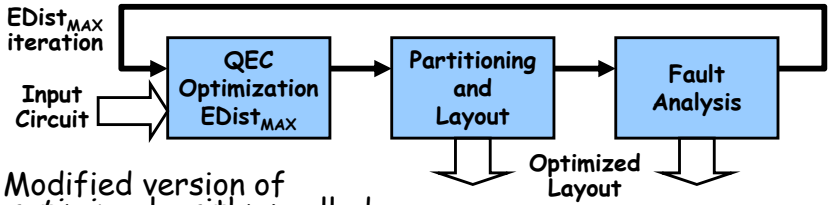
Optimizing Error Correction

Reducing QEC Overhead



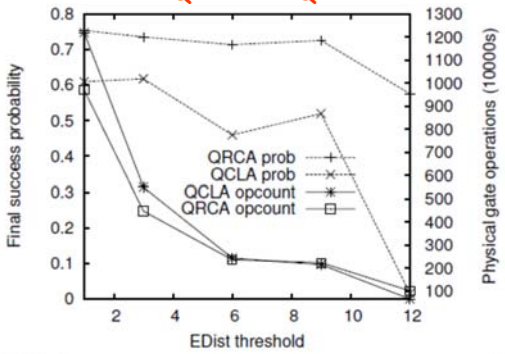
- Standard idea: correct after every gate, and long communication, and long idle time
  - This is the easiest for people to analyze
- This technique is suboptimal
  - Not every bit has same noise level!
- Different idea: identify critical Qubits
  - Try to identify paths that feed into noisiest output bits
  - Place correction along these paths to reduce maximum noise

QEC Optimization

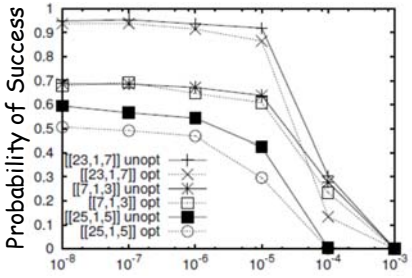


- Modified version of *retiming* algorithm: called "recorrection:"
  - Find minimal placement of correction operations that meets specified MAX(EDist) ≤ EDist$_{MAX}$
- Probably of success *not* always reduced for EDist$_{MAX}$ > 1
  - But, operation count and area drastically reduced
- Use Actual Layouts and Fault Analysis
  - Optimization *pre-layout*, evaluated *post-layout*

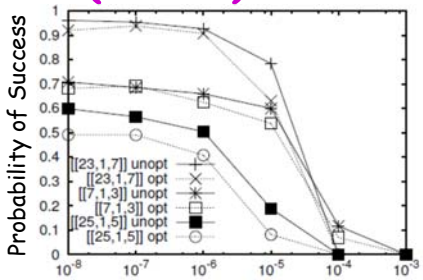
1024-bit QRCA and QCLA adders



Recorrection of 500-gate Random Circuit (r=0.5)



Move Error Rate per Macroblock
EDist$_{MAX}$=3



Idle Error Rate per CNOT Time
EDist$_{MAX}$=3

- Not all codes do equally well with Recorrection
  - Both [[23,1,7]] and [[7,1,3]] reasonable candidates
  - [[25,1,5]] doesn't seem to do as well
- Cost of communication and Idle errors is clear here!
- However – real optimization situation would vary EDist to find optimal point

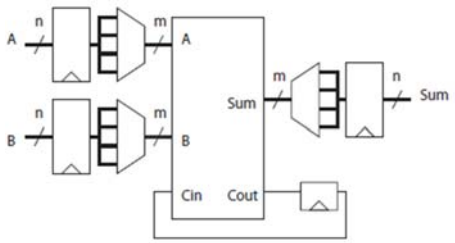
Investigating Larger Circuits

What does Quadence do?

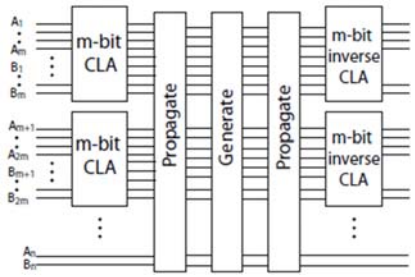
- ECC Insertion and Optimization
  - Logical ⇒ Physical circuits
    - Includes encoding, and correction
  - ECC Recorrection optimization (more later)
- Circuit partitioning
  - Find minimum places to cut large circuit
  - Compute ancilla needs
  - Place physical qubits in proper regions of grid
- Communication Estimation and insertion
  - Generate Custom Teleportation network
- Schedule movement of bits
  - Movement within Ancilla generators (Macros)
  - Movement within compute and memory regions
  - Movement two and from teleportation stations
- Simulation of result to get timing for full circuit
- MonteCarlo simulation to get error analysis

Possible 1024-bit adders

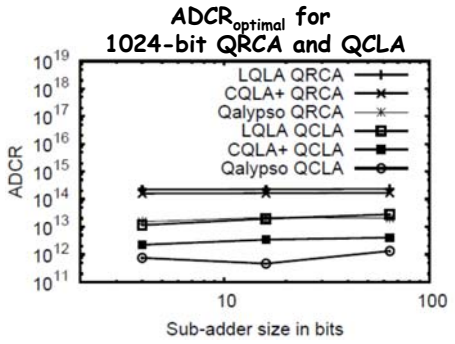
- Quantum Ripple-Carry adder (QRCA)
  - Tradeoffs between area and parallelism
  - Or – between speed and circuit reuse
  - Subadder: m-bit QRCA

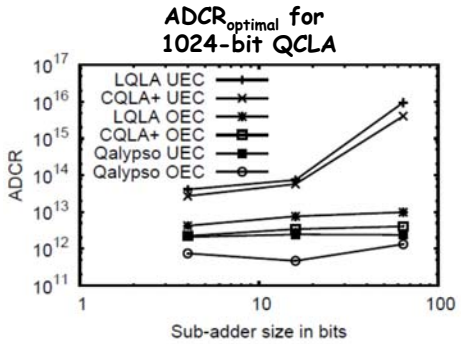


- Quantum Carry-Lookahead adder (QCLA)
  - Stronger tradeoff between area and parallelism
  - Arity of carry-lookahead
  - Subadder: m-bit QCLA



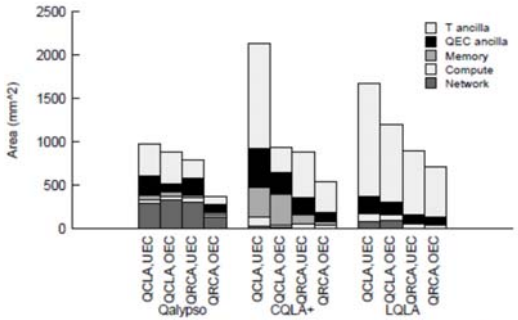
Comparison of 1024-bit adders





- Carry-Lookahead is better in all architectures
- QEC Optimization improves ADCR by order of magnitude in some circuit configurations

Area Breakdown for Adders

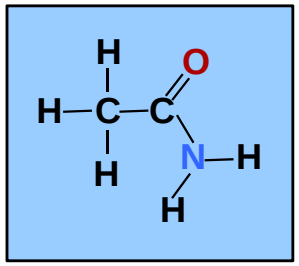


- Error Correction is *not* predominant use of area
  - Only 20-40% of area devoted to QEC ancilla
  - For Optimized Qalypso QCLA, 70% of operations for QEC ancilla generation, but only about 20% of area
- T-Ancilla generation is major component
  - Often overlooked
- Networking is significant portion of area when allowed to optimize for ADCR (30%)
  - CQLA and QLA variants didn't really allow for much flexibility

Direct Comparison: Concatenated and Topological QECC

Ground State Estimation

- Ground State Estimation
  - Find ground state of Glycine
- Problem Size:
  - 50 Basis Functions
  - Result Calculated with 5 Bits accuracy
  - 60 Qubits, 6.9 x 10^{12} gates, Parallelism: 2.5
- Conceptual Primitives
  - Quantum Simulation and Phase Estimation



Properties of Quantum Technologies: Gate Times and Errors

	Supercond. Qubits (Primitive)	Supercond. Qubits (Optimal)	Ion Traps (Primitive)	Ion Traps (Optimal)	Neutral Atoms (Primitive)	Neutral Atoms (Trotter)
Time (ns)	25	28	32,000	32,000	14,818	19,465
Gate Err	1.0×10^{-5}	6.6×10^{-4}	3.2×10^{-9}	2.9×10^{-7}	8.1×10^{-3}	1.5×10^{-3}
Mem Err	1.0×10^{-5}	1.0×10^{-5}	2.5×10^{-12}	2.5×10^{-12}	0.0	0.0

- Ion traps slower but more reliable than superconductors
- Neutral atoms unusable with concat. codes

Ground State Estimation, Multiple Technologies

	Neutral Atoms (Trotter)	Supercond. Qubits (Primitive)	Ion Trap (Primitive)	
	1×10^{-3} 19,000 ns	1×10^{-5} 25 ns	1×10^{-9} 32,000 ns	
Surface Code	10,883 years	4.5 years	5,588 years	Time
	2.0×10^{24}	3.5×10^{22}	3.9×10^{22}	Gates
	2.5×10^{8}	1.7×10^{7}	4.4×10^{7}	Qubits
Bacon Shor Code	-	4,229 years	128 years	Time
	-	9.5×10^{32}	1.5×10^{19}	Gates
	-	9.4×10^{11}	1.6×10^{5}	Qubits
	-	5	1	Concatenations

Conclusion

- How to express quantum algorithms?
  - Embedded DSLs in higher-order languages
- Size of Quantum Circuits ⇒ Must Optimize Locality
  - Presented Some details of a Full CAD flow (Partitioning, Layout, Simulation, Error Analysis)
  - New Evaluation Metric: ADCR = Area × E(Latency)
  - Full mapping and layout accounts for communication cost
- Ancilla Optimization Important
  - Ancilla bandwidth varies widely
  - Custom ancilla factories sized to meet needs of circuit
- "Recorrection" Optimization for QEC
  - Selective placement of error correction blocks
  - Validation with full layout to find optimal level of correction
- Analysis of 1024-bit adder architectures
  - Carry-Lookahead adders better than Ripple Carry adders
  - Error correction *not* the primary consumer of area!