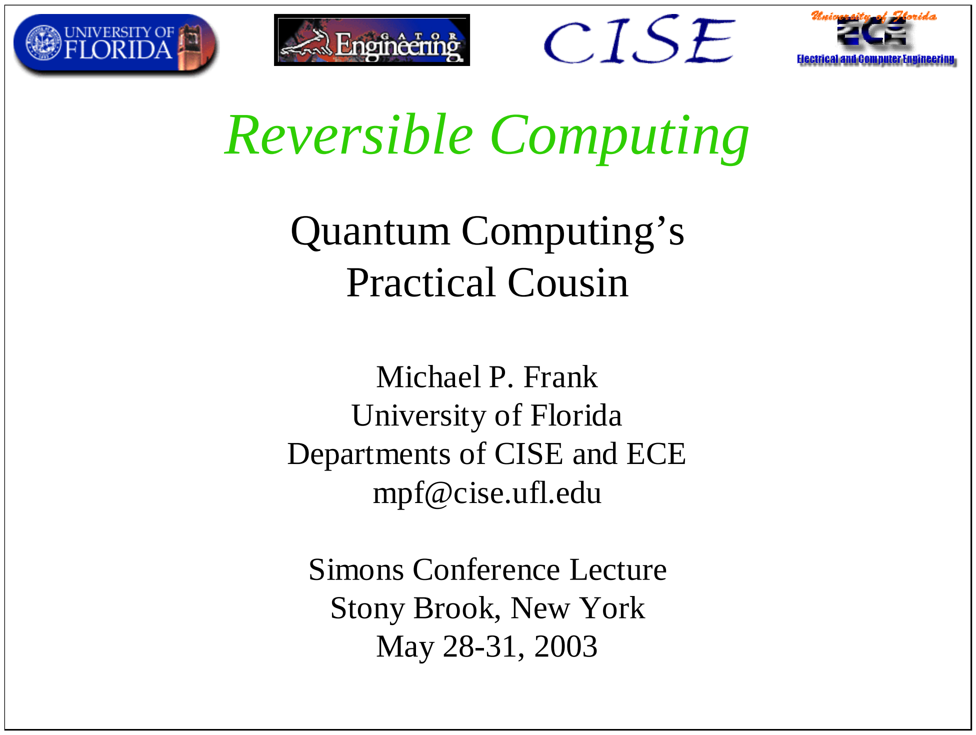


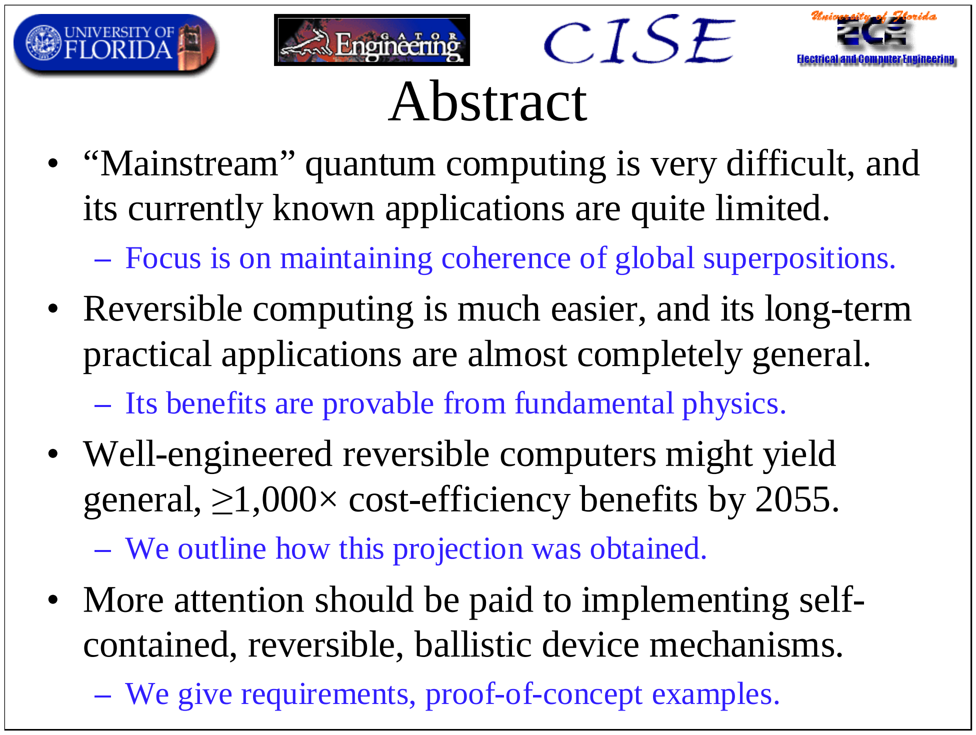
Reversible Computing

Quantum Computing's
Practical Cousin

Michael P. Frank
University of Florida
Departments of CISE and ECE
mpf@cise.ufl.edu

Simons Conference Lecture
Stony Brook, New York
May 28-31, 2003

The title of this talk is meant to be a little bit provocative, given that this group mostly consists of quantum computing people... But, don't get me wrong; I like quantum computing as much as the rest of you; I have been following the field with interest for about 8 years now. I do not mean to apply that quantum computing will never be practical for *some* applications. However, what I hope to convince you is that even without the quantum speedups, an area that is closely related to (but distinct from) QC, called Reversible Computing, will actually be even *more* useful than QC for the majority of real-world practical computing applications.



Abstract

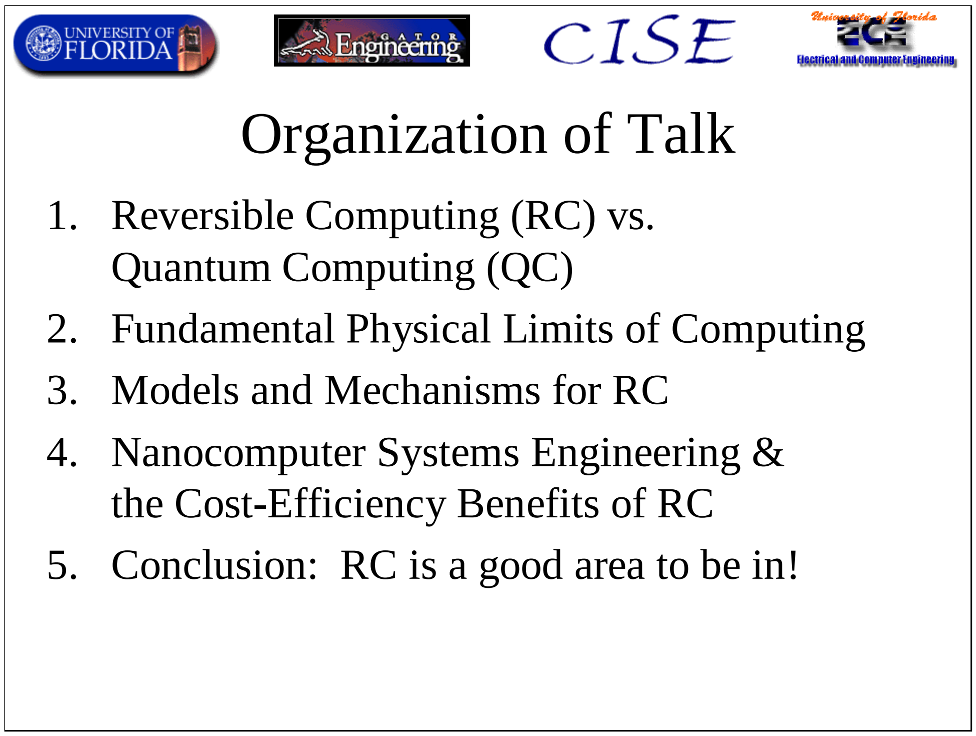
- "Mainstream" quantum computing is very difficult, and its currently known applications are quite limited.
  - Focus is on maintaining coherence of global superpositions.
- Reversible computing is much easier, and its long-term practical applications are almost completely general.
  - Its benefits are provable from fundamental physics.
- Well-engineered reversible computers might yield general, ≥1,000× cost-efficiency benefits by 2055.
  - We outline how this projection was obtained.
- More attention should be paid to implementing self-contained, reversible, ballistic device mechanisms.
  - We give requirements, proof-of-concept examples.

This is just a summary of the main points made in the already-announced abstract for the talk. The first observation is that mainstream quantum computing, which relies on maintaining coherence of global superposition states, is for this reason very difficult to achieve technologically. Moreover, its applications are quite limited.

If we avoid the requirement for global coherence and instead just require a sort of approximate local coherence, the resulting picture—classical reversible computing—is much easier to achieve technologically. Nevertheless, it still has significant advantages (ranging from large constant-factors to polynomial advantages, depending on the model) for, it turns out, the majority of general-purpose computing applications, in the long term. We can prove these advantages exist using thermodynamical and other considerations from fundamental physics. We can show advantages in terms of both cost-efficiency and raw performance (operations per second).

I will show a projection based on existing technological trends and known fundamental limits which suggests that a well-engineered reversible computer should be able to give us a cost-efficiency boost for most applications of at least a factor of 1,000 by about the middle of this century.

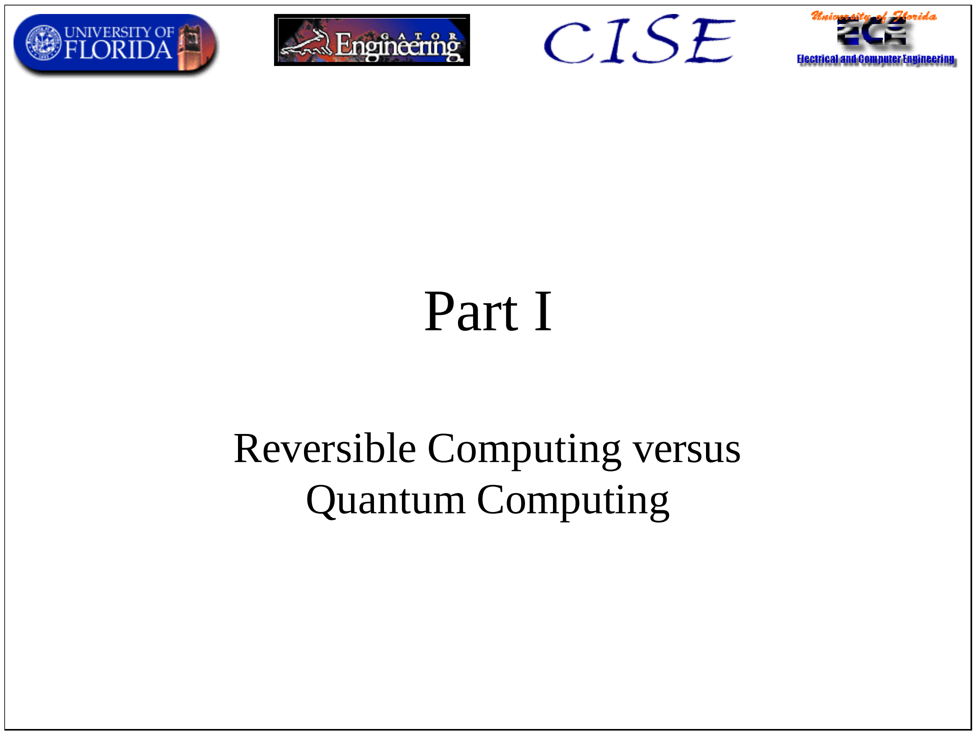
The conclusion is that more people need to join this field (reversible computing) and help work on the detailed engineering design of reversible devices. It is maybe not quite as interesting and deep of a subject, mathematically or theoretically speaking, as is quantum computing, but nevertheless it is still viewed as really far-out, blue-sky research from the perspective of industry at this time. It occupies this sort of "in between" status, between theory and practice, and not so many people are interested in both worlds. But, I believe it will be extremely important for the future of computing.



Organization of Talk

1. Reversible Computing (RC) vs.
   Quantum Computing (QC)
2. Fundamental Physical Limits of Computing
3. Models and Mechanisms for RC
4. Nanocomputer Systems Engineering &
   the Cost-Efficiency Benefits of RC
5. Conclusion: RC is a good area to be in!

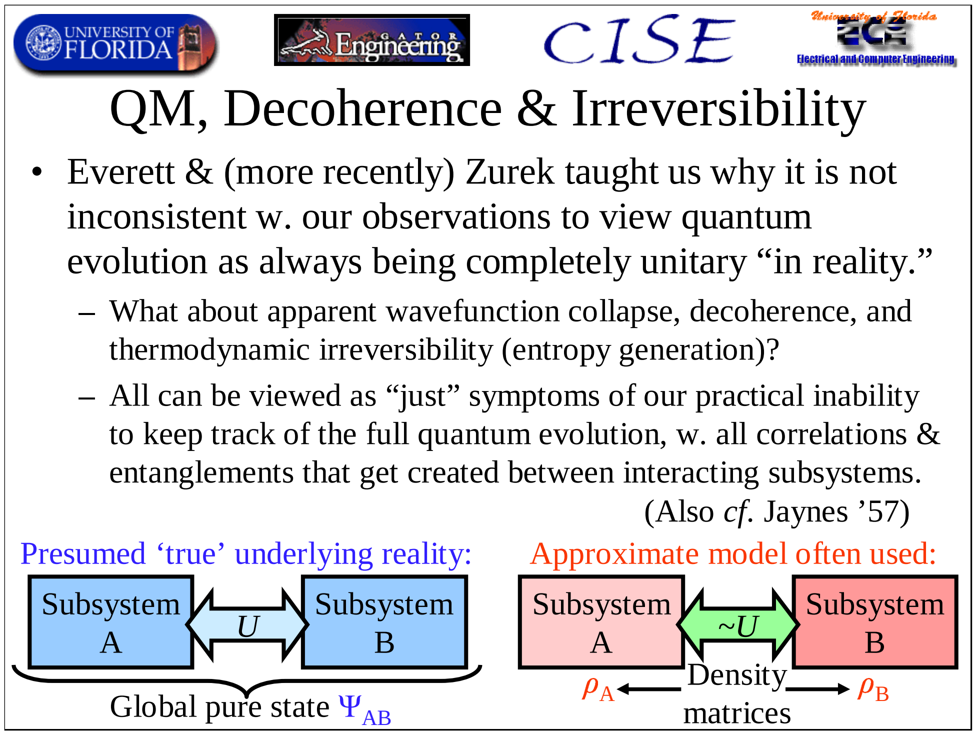
So, here's the structure of the talk. First I will compare and contrast reversible and quantum computing in terms of their aims, requirements, and benefits. Then, I'll spend a little time overviewing some of the fundamental physical limits of computing which form the basis for my models of computing that I use to analyze reversible computing. Then, I'll talk about these models and illustrate some RC mechanisms. Next, I'll talk about how I analyze the cost-efficiency benefits of reversibility. And my conclusion will be that RC is a good area to be in, at least for someone like me who wants to achieve practical engineering results.



Part I

Reversible Computing versus
Quantum Computing

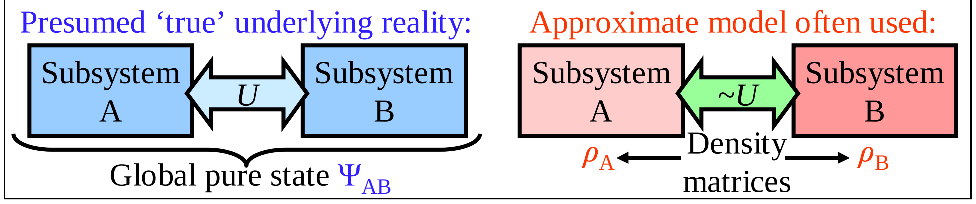
In this first part of the talk, I compare reversible computing to quantum computing.



QM, Decoherence & Irreversibility

- Everett & (more recently) Zurek taught us why it is not inconsistent w. our observations to view quantum evolution as always being completely unitary "in reality."
  - What about apparent wavefunction collapse, decoherence, and thermodynamic irreversibility (entropy generation)?
  - All can be viewed as "just" symptoms of our practical inability to keep track of the full quantum evolution, w. all correlations & entanglements that get created between interacting subsystems.

(Also *cf.* Jaynes '57)

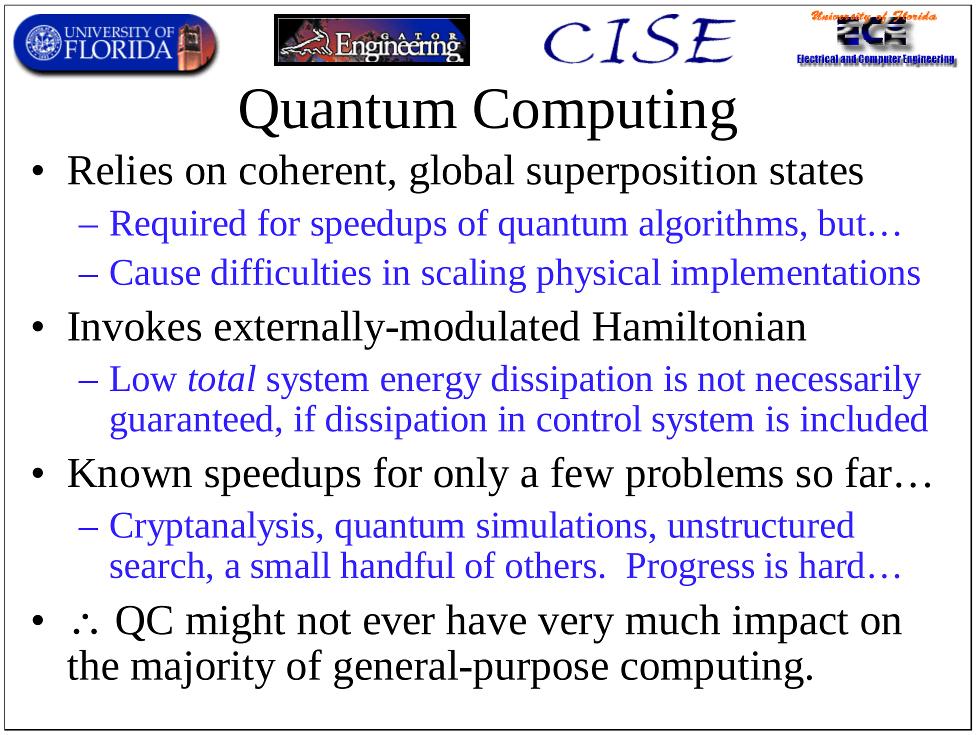

Presumed 'true' underlying reality:
Subsystem A
U
Subsystem B
Global pure state Ψ_{AB}
Approximate model often used:
Subsystem A
$\sim U$
Subsystem B
ρ_A
Density matrices
ρ_B

Segue: First, a little basic background for those who don't know. (But I don't want to spend too much time on this slide because this topic was already covered by previous speakers.) The best modern understanding from particle physics, etc. is that, as far as we know, it is perfectly consistent to view the time-evolution of the quantum state of any closed system (including the whole universe) as always being completely unitary "in reality." As Levitin was saying in the last talk, phase space is conserved and "true" entropy does not increase over time. The empirical phenomena of "wavefunction collapse," decoherence of quantum states, and thermodynamic irreversibility (which are all basically equivalent) can apparently be satisfactorily explained as simply consequences of our inability to track the full quantum evolution of a system.

For example, suppose the full quantum state is unknown and we factor the system into subsystems and model our ignorance about the system using separate density matrices describing a distribution over pure states for the subsystems separately. In doing this, we ignore the possible correlations and entanglement that might actually exist between the subsystems. It is a consequence of this style of modeling (as was first shown by Jaynes in 1957) that the entropy of the systems, so modeled, will in general increase when you allow them to interact, even if perfectly unitarily. Zurek described in detail this process of decoherence, in which the off-diagonal coherence terms in the density matrix decrease towards zero. Another reason entropy might increase is if the unitary interaction U between subsystems is not known exactly but only statistically.

An audience member asked if our inability to track the quantum state was a matter of practice or principle. My view is that it is a matter of practice. In practical terms we can never perfectly isolate any real physical system from some level of unwanted parasitic interactions with its outside environment. However, if you considered a system that was truly closed, if you knew its exact quantum state at one time, and the exact form of its unitary time evolution, then the evolution of a definite state could always be tracked (either analytically or by a computer simulation with any desired accuracy) with no increase in entropy.

After the talk, Warren Smith objected to the Everett universal-wavefunction view because of the "equivalence of all bases" in quantum mechanics, and it seems that he doesn't trust Zurek's approach of deriving a preferred basis for decoherence (e.g. position bases) by looking at the form of the interaction Hamiltonian. However, it is my understanding that Everett augmented with Zurek really does work, and I have never seen a convincing and logically rigorous demonstration that it does not. Perhaps Warren is correct that the consistency of the approach has not really been absolutely thoroughly, rigorously proven, but in fact we almost never do have real consistency proofs, even in pure mathematics. (In particular, the very foundations of analysis, e.g. ZF set theory, have never been proven consistent.) I believe that as a matter of methodology, the most pragmatic way to make progress, as rapidly as possible, in both mathematics and physics is to stick with the simplest available theories that seem to work (since we might expect that by Occam's Razor these are most likely to be valid) until there is some absolutely clear and definite proof of their inconsistency (either internally, in the mathematics, or a direct contradiction between the theory's predictions and experience) that forces us to move to an alternative model. In my view there has never been a logically valid proof that plain quantum theory with only pure states and global unitary evolution (and no "true" wavefunction collapse) is actually inconsistent with any aspect of our experience, and this theory is the conceptually simplest one that works, so I think that we must trust that theory's predictions until they are empirically (or logically) disproven.



Quantum Computing

- Relies on coherent, global superposition states
  - Required for speedups of quantum algorithms, but…
  - Cause difficulties in scaling physical implementations
- Invokes externally-modulated Hamiltonian
  - Low *total* system energy dissipation is not necessarily guaranteed, if dissipation in control system is included
- Known speedups for only a few problems so far…
  - Cryptanalysis, quantum simulations, unstructured search, a small handful of others. Progress is hard…
- ∴ QC might not ever have very much impact on the majority of general-purpose computing.

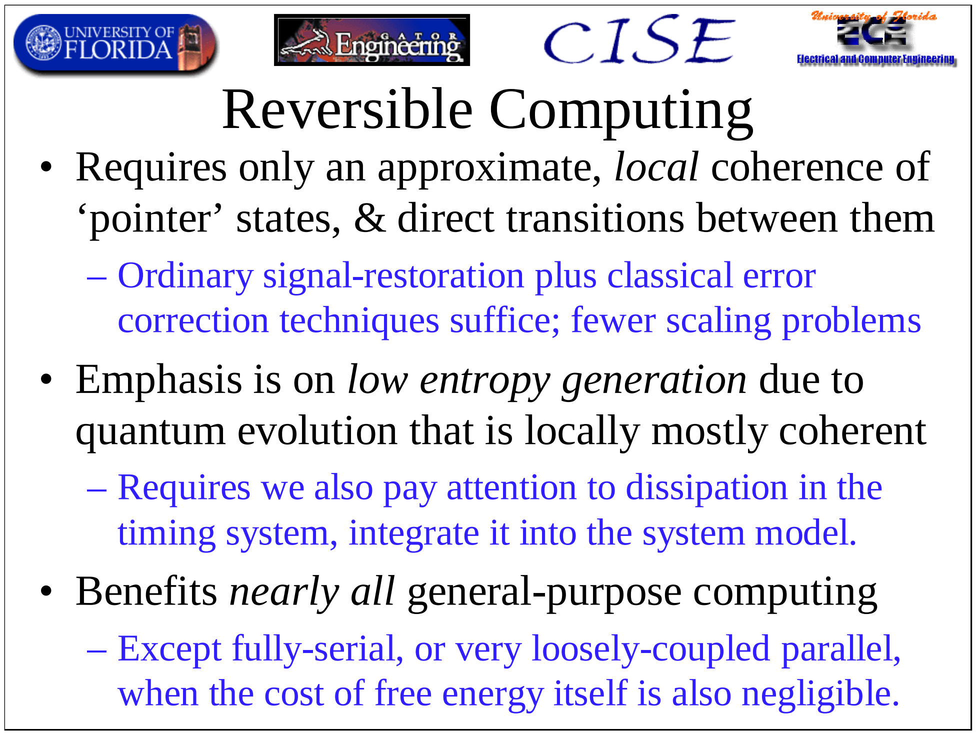
Now let's compare reversible computing and quantum computing. As we all know (at this conference), quantum computing crucially relies on maintaining coherent, global superposition states. These are needed to obtain the polynomial to exponential speedups enabled by quantum algorithms. But this requirement causes difficulties in scaling up physical implementations. As long as the local decoherence rate is below a certain threshold, fault-tolerant error correction techniques can be applied to maintain global coherence, but these introduce a significant amount of additional asymptotic overhead in the complexity of quantum logic networks and algorithms.

A second aspect of most QC work that I want to emphasize is that virtually all of the existing theoretical and experimental frameworks for quantum computing invoke an external system (which is not usually well modeled) that is responsible for controlling the computation and driving it along its trajectory in configuration space. (Although we had one previous speaker who discussed how to treat the controller as a quantum system as well.) Having an external control is fine for quantum algorithms, but later in the talk I will emphasize the advantages of low total system-wide entropy generation. I want to emphasize that the entropy generation of a complete computation (one that does not benefit from quantum speedups) is not necessarily minimized by the QC approach if the entropy generation in the external control system is included. Later I will talk about models that are totally self-contained.

A third aspect of quantum computing I want to point out is that we only know how to obtain asymptotic speedups so far for a very narrow class of problems, factoring and simulating QM and unstructured search and a few other number-theory problems. All of these problems added together only comprise a miniscule portion of the entire world market for computing. (Once RSA is broken and everyone moves to quantum cryptography, the biggest market for QC will probably be quantum physical simulations, but that's still a tiny part of the world market.) In the future, quantum algorithms with broader applications may yet be discovered. But progress in this area has proven so far to be very difficult, and so we cannot count on this assumption.

As a result of primarily this third point, even if the decoherence problems can be solved, and we can scale to millions of qubits, QC itself might still not ever have a very significant impact on the majority of general-purpose computing.

Let me now contrast these points with Reversible Computing.



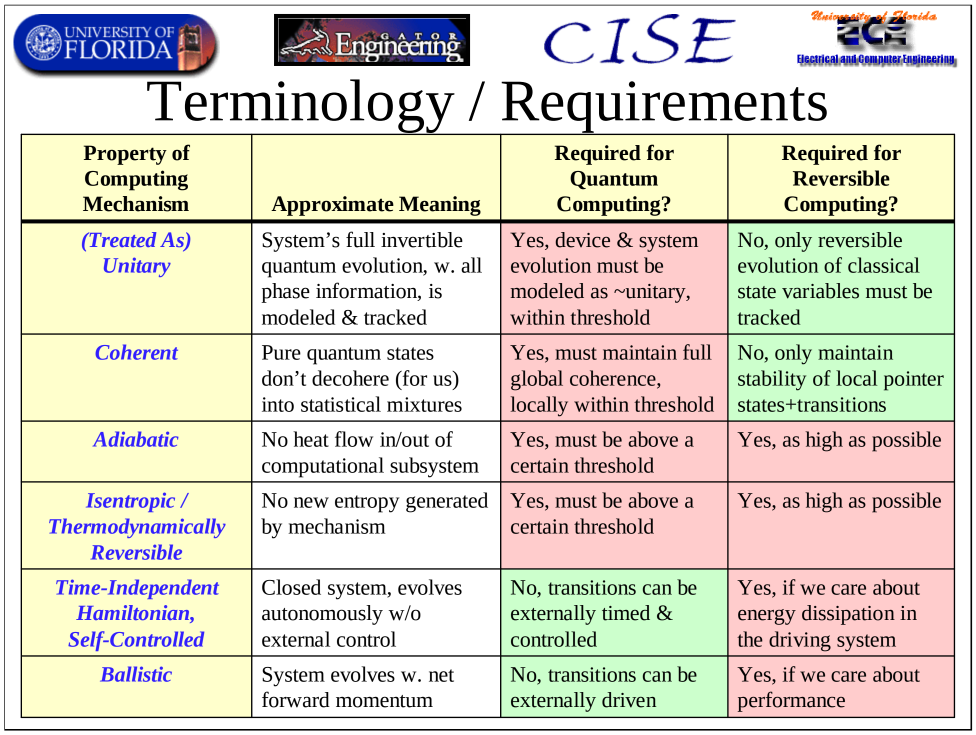
Reversible Computing

- Requires only an approximate, *local* coherence of ‘pointer’ states, & direct transitions between them
  - Ordinary signal-restoration plus classical error correction techniques suffice; fewer scaling problems
- Emphasis is on *low entropy generation* due to quantum evolution that is locally mostly coherent
  - Requires we also pay attention to dissipation in the timing system, integrate it into the system model.
- Benefits *nearly all* general-purpose computing
  - Except fully-serial, or very loosely-coupled parallel, when the cost of free energy itself is also negligible.

First, reversible computing does not require global coherence of superposition states. It only requires approximate local coherence for stable storage of naturally occuring stable ‘pointer’ states (eigenstates of the parasitic interaction Hamiltonian between the system and its environment) which are chosen as computational basis states, and temporary, local, approximate coherence of superpositions occurring along a direct transition between one pointer state and the next as the computer evolves through its configuration space. This approximate coherence is needed only to maintain a low rate of local entropy generation; global coherence is not needed. Therefore, we avoid the scaling problems associated with the fault-tolerant error-correction algorithms. We can use engineering of decoherence-free subspaces (isolation of computational subsystems from their environment) to minimize local decoherence as far as possible, while using ordinary “classical” techniques for signal restoration and error correction (as needed) to prevent the global trajectory of the computation from going awry. This avoids many of the scaling problems of the full-quantum approach.

But if we can’t implement quantum algorithms, what then is the point? The point is on the low rate of entropy generation of a quantum evolution that is mostly coherent even if just in this limited, local fashion. We will see that in the long term this low rate of entropy generation leads to speedups and thus improvements in cost-efficiency since entropy generation will be the primary limiting factor on the performance of future nanocomputing technologies. Since our concern is minimizing entropy generation, we also have to pay attention to how the system is controlled, timed, and driven, and model the entropy generation due to this process, and take it into account in the full system design.

Finally, we will see that this approach gives a practical benefit for nearly all general-purpose computing. The only exceptions are cases where the economic cost of energy is negligible (this is not true today) AND the application is one where a fully-serial or totally loosely-coupled (distributed) parallel algorithm gives optimal performance. This is because in these cases processors can be spread out over a large area and heat removal (cooling) is not a limiting factor on performance. However, the most general case is one in which the optimal algorithm for a given problem both benefits from parallelism AND requires frequent communication between distributed processors, so that there is a benefit from clumping processors together. In such cases, by reducing power requirements, reversible computing can allow increased compactness of the design and thus higher performance. Also, even in fully-serial or loosely-coupled parallel applications, total energy dissipation is a concern simply because of the economic cost of energy. Irreversible computing leads to a certain maximum performance per Watt of power consumption. Only reversible computing can circumvent this limit.



Terminology / Requirements

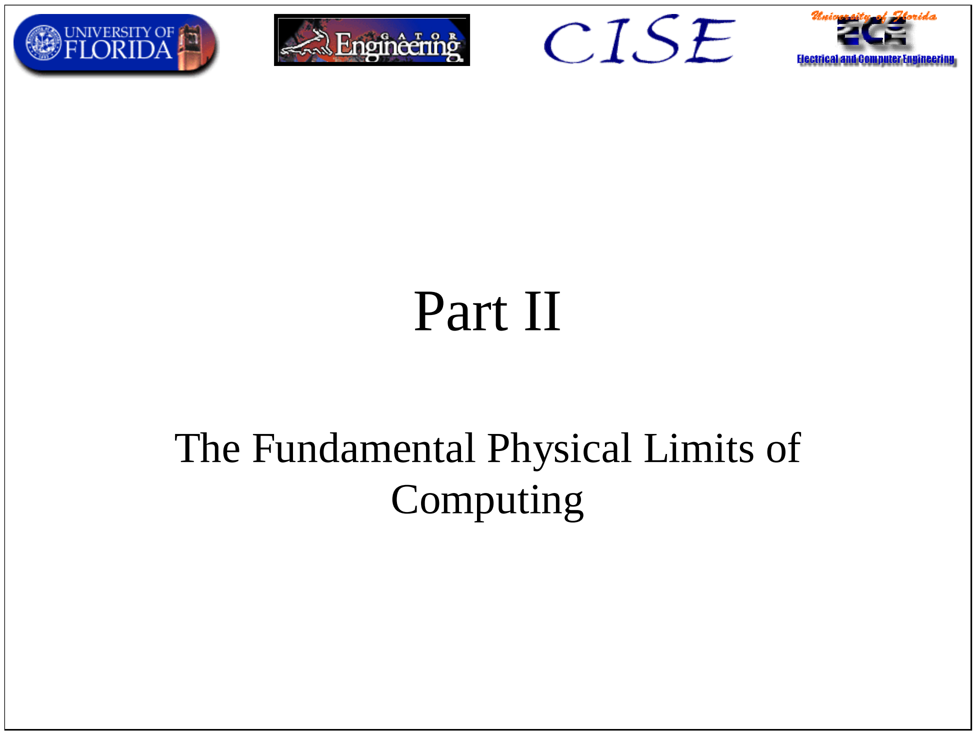
Property of Computing Mechanism	Approximate Meaning	Required for Quantum Computing?	Required for Reversible Computing?
(Treated As) Unitary	System's full invertible quantum evolution, w. all phase information, is modeled & tracked	Yes, device & system evolution must be modeled as ~unitary, within threshold	No, only reversible evolution of classical state variables must be tracked
Coherent	Pure quantum states don't decohere (for us) into statistical mixtures	Yes, must maintain full global coherence, locally within threshold	No, only maintain stability of local pointer states+transitions
Adiabatic	No heat flow in/out of computational subsystem	Yes, must be above a certain threshold	Yes, as high as possible
Isentropic / Thermodynamically Reversible	No new entropy generated by mechanism	Yes, must be above a certain threshold	Yes, as high as possible
Time-Independent Hamiltonian, Self-Controlled	Closed system, evolves autonomously w/o external control	No, transitions can be externally timed & controlled	Yes, if we care about energy dissipation in the driving system
Ballistic	System evolves w. net forward momentum	No, transitions can be externally driven	Yes, if we care about performance

This chart compares side-by-side the requirements for quantum and reversible computing. A quantum computer must be treated as evolving unitarily and coherently (which mean almost the same thing), whereas a reversible computer only needs to be locally approximately coherent (which is easier) and we do not need to track the unitary evolution of a full quantum state, only the reversible evolution of classical state variables. So it is much easier to model and simulate a reversible computer. (No exponential explosion in size of state description.)

Both quantum and reversible computing need to be adiabatic, as nearly as possible. Adiabatic literally means, "no heat flow"; if there is no heat flow between subsystems at different temperatures, this means there is no entropy generation. Essentially it means the system is isentropic or thermodynamically reversible.

Next, a reversible system needs to be closed in the sense of being self-controlled, evolving autonomously under a time-independent Hamiltonian, with no external control, whereas a quantum computer can be externally controlled. There are two reasons why the reversible computer needs to be self-controlled. First, since we care about total system energy dissipation, we must model the dissipation in any timing/control/driving system, and in its interaction with our reversible computational evolution. If we model it we may as well integrate it and consider the system as a whole to be autonomous. Second, if the control system is external, there are some questions about the scalability of the distribution of the control signals throughout a large parallel computer scaled up in 3-D. With an autonomous model, we will see we can locally synchronize the processors.

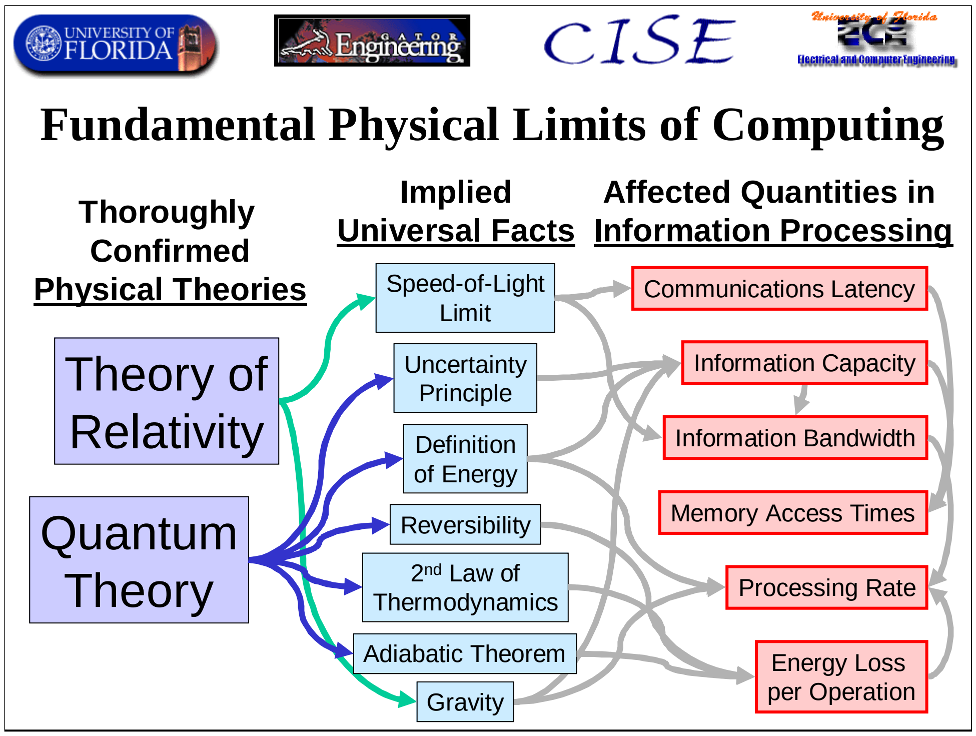
Finally, a reversible system needs to be "ballistic", which I take as meaning that the system (like a ballistic projectile) has a net forward momentum along its trajectory through configuration space, instead of just for example doing a diffusive random walk through its configuration space. This is needed for performance. A quantum computer, on the other hand, can sit statically storing its state whenever it is not being actively driven.



Part II

The Fundamental Physical Limits of Computing

In this next part of the talk, I give a whirlwind tour through the different fundamental physical limits on computing that I know about and account for in my analyses.



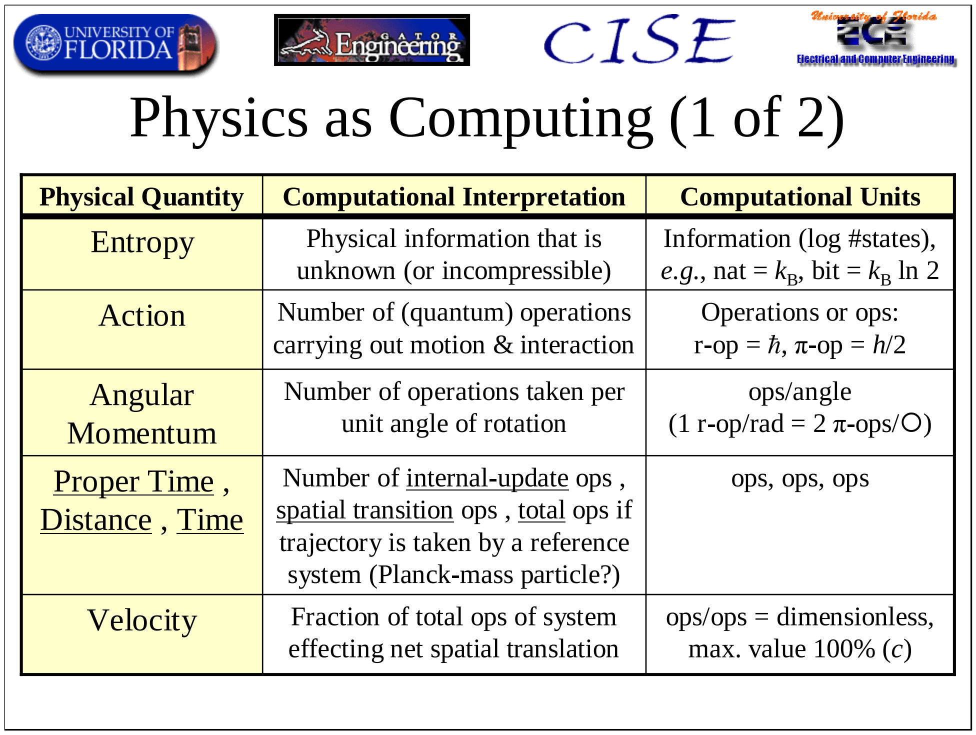
Fundamental Physical Limits of Computing

In our modeling efforts, we are not free to choose the mathematical structure of our computer model on a whim. If our model is to be physically realistic, it must respect the fundamental physical constraints on information processing that logically follow from our well-established theoretical models of physics.

Einstein's Special and General Theories of Relativity, together with Quantum Theory as expressed in the Standard Model of particle physics, together constitute an absolutely correct and precise description of our universe, as far as any currently experimentally-accessible phenomena are concerned. Both theories have been confirmed and verified many times to an accuracy of many decimal places. It is not likely that any further refinements of known physics will affect the limits of nanocomputing this century.

Relativity implies the speed-of-light limit on information propagation, as well as gravity. Quantum theory implies many things, including Heisenberg's Uncertainty principle, the relation between energy and frequency, the fundamental time-reversibility of physical law, the laws of thermodynamics, and the behavior of adiabatic (nearly perfectly ballistic) processes. These lead to the following limits on computing:

(1) The speed-of-light limit implies a lower bound on communications latency across a machine of given diameter. (2) The uncertainty principle together with the quantum definition of energy, imply an upper bound on information capacity for a system of given diameter and total energy. (3) Together with the speed-of-light limit, this implies an upper limit on information flow rate or bandwidth density per unit area for a flow of given power. (4) These principles also yield a lower bound on average random-access times for a memory of given size and energy density. (5) The quantum definition of energy also implies a limit on the rate at which useful bit-operations may be performed in a system of given free energy (due to the Margolus-Levitin theorem), as well as a limit on step-rate (clock frequency) for a logic system of given temperature. (6) The reversibility of quantum mechanics implies lower bounds on the entropy generated by bit erasure, and the free energy loss given the environment's temperature. (7) Therefore, irreversible computers in a given environment have a limited rate of operation per Watt of power consumption. (8) These facts also imply a maximum rate of irreversible bit-operations within a fixed-area enclosure in a fixed-temperature surroundings. (9) Meanwhile, the 2nd law of thermodynamics guarantees us that all systems (even reversible, adiabatic processes, as well as bits that are just statically storing data), still generate entropy at some rate, however small, and this limits the total capacity of a machine as a function of the entropy generation rate per device and the machine's total power. (10) Meanwhile, the adiabatic theorem of quantum mechanics teaches us that entropy generation of reversible operations is proportional to speed, which lets us put an upper limit on the rate of reversible operations per unit area. (11) Finally, general relativity gives loose upper bounds on the internal entropy and energy of computers of given diameter, which in turn limits their maximum speed and capacity.



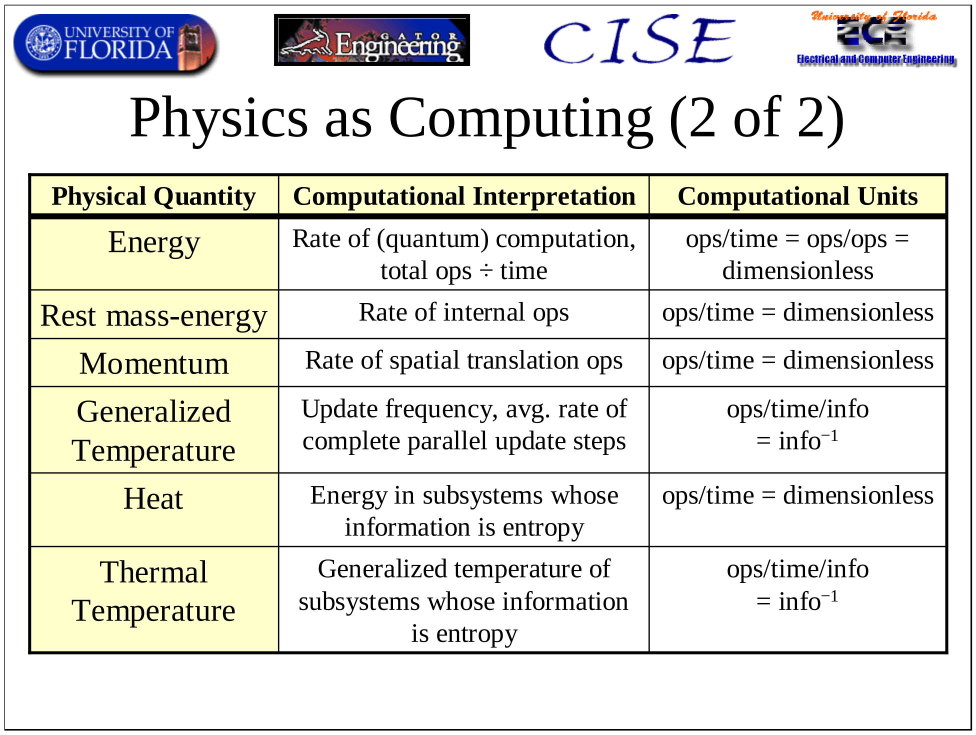
Physics as Computing (1 of 2)

Physical Quantity	Computational Interpretation	Computational Units
Entropy	Physical information that is unknown (or incompressible)	Information (log #states), *e.g.*, nat = k_B, bit = k_B ln 2
Action	Number of (quantum) operations carrying out motion & interaction	Operations or ops: r-op = $\hbar$, π-op = $h/2$
Angular Momentum	Number of operations taken per unit angle of rotation	ops/angle (1 r-op/rad = 2 π-ops/○)
Proper Time , Distance , Time	Number of internal-update ops , spatial transition ops , total ops if trajectory is taken by a reference system (Planck-mass particle?)	ops, ops, ops
Velocity	Fraction of total ops of system effecting net spatial translation	ops/ops = dimensionless, max. value 100% (c)

Incidentally, as a bit of a tangent to the talk, learning about the physical limits of computing leads one to think about ways to reinterpret physical quantities themselves in terms of computation. This is somewhat of a hobby of mine. Let me go through this briefly. (I am hoping to find time to prepare a complete paper on this soon.)

(1) Let us define "physical information" as the log of the total number of distinguishable states of a physical system. As some previous speakers have discussed, physical entropy can be interpreted as just a measure of the amount of the portion of that physical information whose specific informational content (identity, value) is completely unknown. Zurek has suggested extending this definition to also include physical information that is incompressible for any reason, not just because it is unknown. This is because information that is incompressible is effectively entropy because you cannot do anything sensible with it (cannot get rid of it so the space can be reused for other purposes) except by expelling it from the machine and turning it to actual unknown information. Another way to look at this extension of the definition is that it allows us to appropriately describe the entropy of correlated systems even "from the inside," from the perspective of one of the correlated systems, which "knows" the state of the other. E.g., if we perform a controlled-NOT between qubit A which contains 1 bit of entropy and qubit B which is in a pure state (0 entropy), qubit B becomes entangled (essentially, correlated) with qubit A. From the "outside", the total entropy is unchanged since the joint state is something like |00> mixed with |11> which still has 1 bit of entropy. However, if you think of qubit B as an "observer" who has made a "measurement" (and indeed, real observers are presumably just large physical quantum systems) then we can say that as a result of the correlation, qubit B "knows" the state of qubit A. That is subjectively (from B's perspective) it is in a definite state (0 or 1) and "knows" that qubit A contains the same value. Since B has two redundant copies of this information – the copy in A and the copy inside itself – this 2 bits of information (the 00 or 11) can be "compressed" into 1 bit, e.g., by simply undoing the coherent measurement operation. But, it cannot be compressed any further than that, since there is nothing else that that bit is correlated with. Actually Zurek's definition is restricted to algorithmically incompressible information, which is an uncomputable quantity in general, but I would go even broader than his definition and say that any information that is practically incompressible (infeasible to compress) is effectively entropy for all practical purposes. Anyway, since entropy is just a special case of information, we measure it in information units, which are logarithms (of the count of distinguishable states). If we use the natural (base e) logarithm, we get the information unit which I call the "nat", which is just equal to Boltzmann's constant k; if we use the logarithm base 2 we get the bit.

(2) Toffoli has previously suggestion that the physical concept of Action can be quantified as "amount of computation." I would like to propose a more specific interpretation along these lines. Let us define an "operation" or op (specifically, what I call a pi-operation, equal to Planck's constant h) as any local unitary transform that rotates at least one quantum state onto an orthogonal one. For example, rotating a spin in space by an angle of pi or 180 degrees, or a quantum controlled-not operation. Note that not all states need be transformed into orthogonal states for the transform to be considered an op. Anyway, we can define fractions of operations (for example, the radian-op or h-bar) and thereby describe any trajectory of a quantum evolution as a composition of local operations of some magnitude, and quantify the total number of operations (in units of pi-ops or r-ops) involved in following that trajectory. Action is traditionally defined as the time integral of the Lagrangian, which is kinetic minus potential energy. Since potential energy is usually defined as negative, this is effectively potential energy plus the absolute value of potential energy; this absolute value becomes greater as the strength of interaction between two particles increases. It is reasonable to suppose that the potential energy represents the average rate of quantum operations involved in carrying out the exchange of virtual particles which implements the interaction force in quantum field theory. Kinetic energy on the other hand we will see represents the rate of operations involved in translational motion. The only kind of energy not included in the usual Lagrangian is the rest mass-energy, but it can be easily generalized to include it. We will see that rest mass-energy represents the rate of operations in a system's updating of its internal state information (as opposed to its motion relative to or interactions with other external systems). (3) Angular momentum can be interpreted as counting the number of ops involved in carrying out a rotation by a certain angle. (4) Relativistic proper time, distance, and normal time along a trajectory through spacetime between two given points can seemingly all be defined by (respectively) the number of internal-update ops, spatial-translation ops, and total ops that would be involved if a reference particle (Planck-mass particle?) were to traverse that same trajectory. These can all therefore be measured in ops. (5) Velocity can be defined as spatial translation ops (momentum times time) as a fraction of total ops (energy times time). It is thus dimensionless and its maximum value is 1 or c.



Physics as Computing (2 of 2)

Physical Quantity	Computational Interpretation	Computational Units
Energy	Rate of (quantum) computation, total ops ÷ time	ops/time = ops/ops = dimensionless
Rest mass-energy	Rate of internal ops	ops/time = dimensionless
Momentum	Rate of spatial translation ops	ops/time = dimensionless
Generalized Temperature	Update frequency, avg. rate of complete parallel update steps	ops/time/info = info^{-1}
Heat	Energy in subsystems whose information is entropy	ops/time = dimensionless
Thermal Temperature	Generalized temperature of subsystems whose information is entropy	ops/time/info = info^{-1}

(6) The Margolus-Levitin theorem tells us that the rate of operations is limited to at most 2 times the average energy (1x for operations along long dynamic orbits) divided by h/2. Even though this is given as only a tight upper bound, we can consider it actually equal to the rate of operation if we note that we have defined operations in terms of orthogonalizing some states – not necessarily all. (This was pointed out to me by Lloyd in personal discussion.) So, we can say that physical energy *is* just a measurement of the rate of carrying out of operations.

(7) Rest mass-energy includes all energy associated with the internal motions and interactions within a system. One can presume that all of mass is ultimately accounted for by some low-level transformations that are taking place. Anyway, to say that some energy is tied up in rest mass is only to identify a particular state of that energy. Since other states of that energy are possible (e.g., two photons of that energy on a collision course just before colliding to create the particle and its antiparticle), we can say that an operation as we have defined it is being carried out, even if the particle state itself does not change.

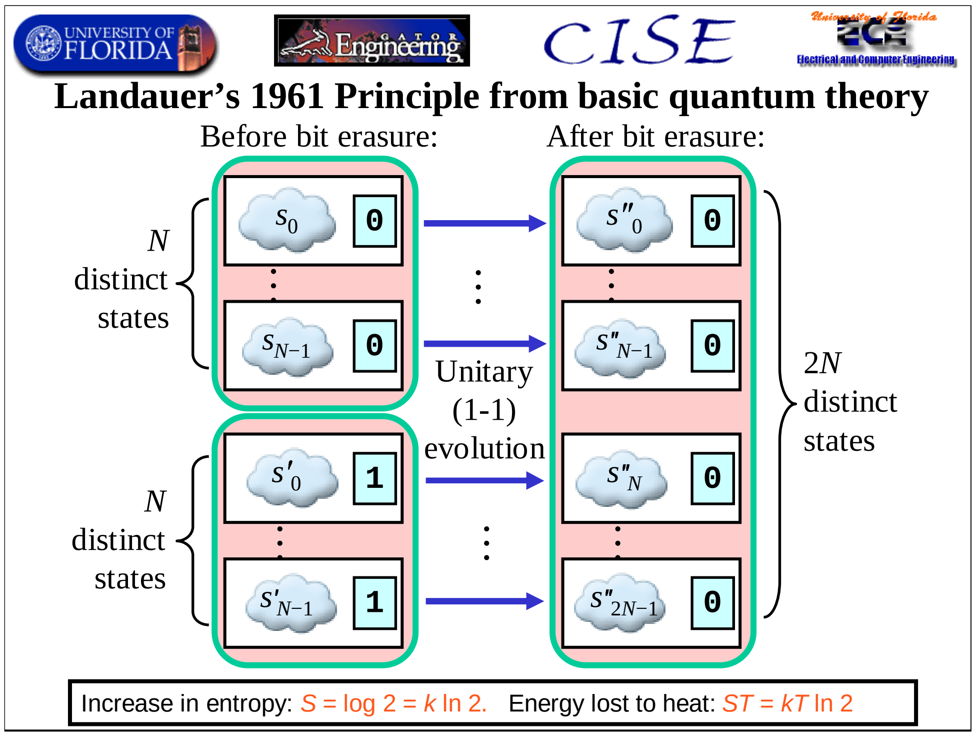
(8) Momentum (relativistic) can be presumed to measure the rate of ops that are carrying out translation of the system through space in the frame of reference of the observer. One motivation for this definition, together with the one for rest mass-energy, is that if we further assume that internal transitions of a system can be considered “orthogonal” to overall translations of the entire system, then it follows from the Pythagorean theorem that for a trajectory proceeding simultaneously in both of these “directions,” $E^2 = m^2 + p^2$. Of course, this is exactly the special-relativistic relation between energy, rest mass, and momentum (in normalized units, where c=1), and in fact all of special relativity (length contraction, time dilation,) follows from this. So, in a sense we have proved special relativity from the computational interpretation of physics (except of course we “cheated” in this by choosing our computational interpretation specifically so that it would achieve this goal).

(9) Given the definition of energy, we can define a generalized concept of temperature by dividing energy by physical information, thus the average rate of operations per unit of information, i.e. the rate of complete parallel update steps, or update frequency (like “clock frequency”). This is not the thermodynamic temperature since it applies even to systems of zero entropy whose thermodynamic temperature would be absolute zero.

(10) We can define heat as the portion of energy that is internal to (causing transitions between states of) subsystems whose state information is entropy (unknown, incompressible).

(11) Proper thermal temperature can then be defined as the generalized temperature of those subsystems whose information is entropy (whose energy is heat).

This is all very interesting, but of course it is still somewhat speculative. However, I have not found any definite inconsistencies between these definitions and known physical laws. In future work I hope to actively verify consistency with known physics and thus make these identities between physics and computation more rigorous and certain.

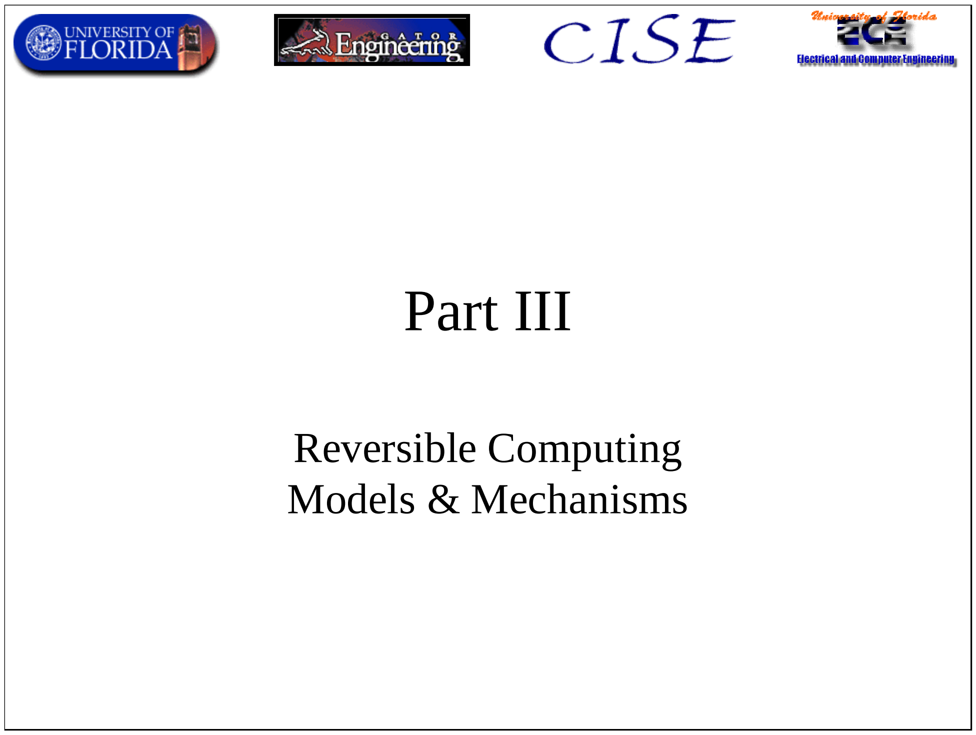


Landauer's 1961 Principle from basic quantum theory

Increase in entropy: $S = \log 2 = k \ln 2$. Energy lost to heat: $ST = kT \ln 2$

Back to the physical limits on computing now.

Let's give an example of the reasoning behind one of these relationships between fundamental physics and information processing. This one is the motivation for reversible computing and was discovered in 1961 by Rolf Landauer of IBM research. Landauer considered the minimum possible physical entropy generation that might result from the erasure of 1 bit of known information. Although Landauer used a more involved argument, the drawing here suffices to prove his point. There are 2 possible logical states of the bit in question, together with some number *N* of distinguishable physical states of the rest of the computer, for a total of 2*N* distict states of the entire machine. The unitary, one-to-one nature of time evolution in quantum mechanics guarantees that the number of distinct states of a closed system is exactly conserved. Therefore, after the logical bit is erased, there are still 2*N* states of the machine. There is the same total amount of variability, but it now must all reside in the rest of the machine. If the information is not logically present, it must be in the form of unknown physical information, or entropy. Since the number of states of the rest of the machine was multiplied by 2, the amount of entropy (which is the logarithm of the state count) is increased by an addition of log 2. A log 2 amount of entropy is (ln 2) times as large as Boltzmann's constant *k*. To release *k*(ln 2) entropy into an environment at temperature *T* requires committing *kT*(ln 2) energy to the environment, by the very definition of temperature. Thus, information erasure ultimately implies a minimum energy expenditure proportional to the temperature of the environment. Note that cooling the system to a low temperature T can not help since the entropy must still eventually get out to the environment, incurring a dissipation of k times the temperature of the environment.

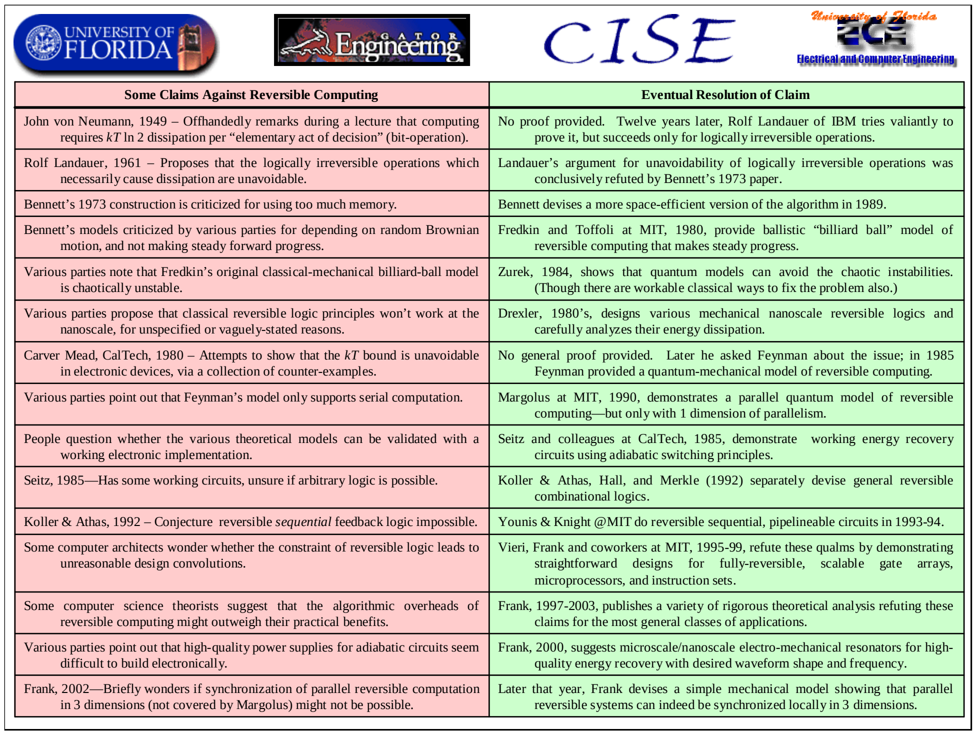


Part III

Reversible Computing
Models & Mechanisms

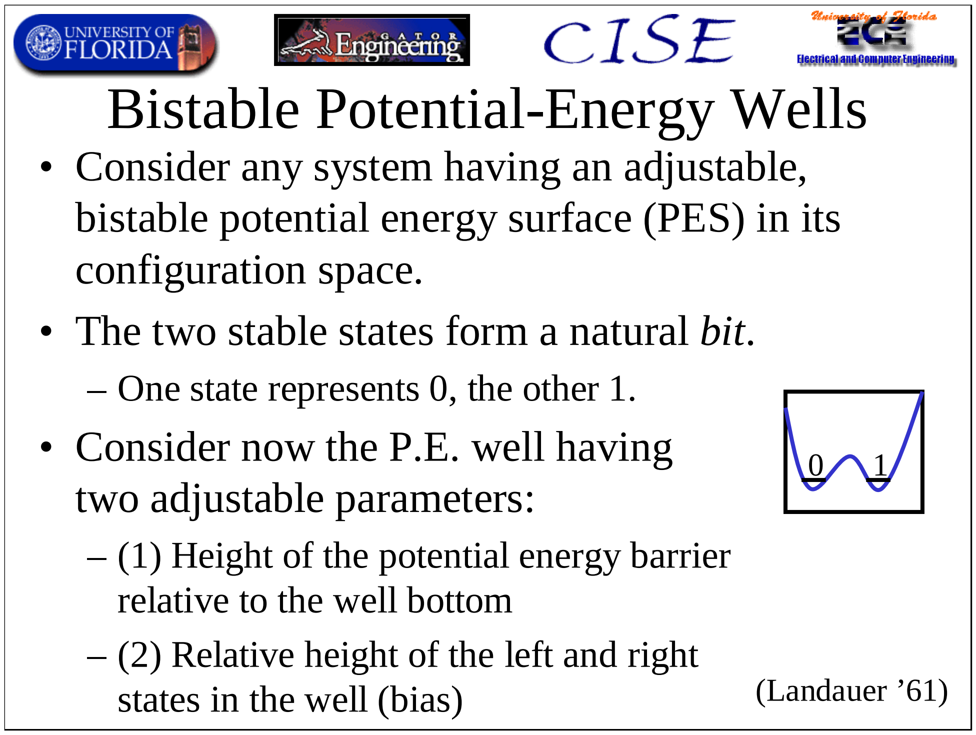
I wish I could go into the rest of the physical limits of computing in detail, but I don't have time here. I review some of the other limits in my paper "Physical Limits of Computing" which you can find on my website http://www.cise.ufl.edu/research/revcomp.

In the next section of the talk I present some theoretical models and more concrete, proof-of-concept physical mechanisms for reversible computing.



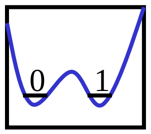
Some Claims Against Reversible Computing	Eventual Resolution of Claim
John von Neumann, 1949 – Offhandedly remarks during a lecture that computing requires $kT \ln 2$ dissipation per "elementary act of decision" (bit-operation).	No proof provided. Twelve years later, Rolf Landauer of IBM tries valiantly to prove it, but succeeds only for logically irreversible operations.
Rolf Landauer, 1961 – Proposes that the logically irreversible operations which necessarily cause dissipation are unavoidable.	Landauer's argument for unavoidability of logically irreversible operations was conclusively refuted by Bennett's 1973 paper.
Bennett's 1973 construction is criticized for using too much memory.	Bennett devises a more space-efficient version of the algorithm in 1989.
Bennett's models criticized by various parties for depending on random Brownian motion, and not making steady forward progress.	Fredkin and Toffoli at MIT, 1980, provide ballistic "billiard ball" model of reversible computing that makes steady progress.
Various parties note that Fredkin's original classical-mechanical billiard-ball model is chaotically unstable.	Zurek, 1984, shows that quantum models can avoid the chaotic instabilities. (Though there are workable classical ways to fix the problem also.)
Various parties propose that classical reversible logic principles won't work at the nanoscale, for unspecified or vaguely-stated reasons.	Drexler, 1980's, designs various mechanical nanoscale reversible logics and carefully analyzes their energy dissipation.
Carver Mead, CalTech, 1980 – Attempts to show that the kT bound is unavoidable in electronic devices, via a collection of counter-examples.	No general proof provided. Later he asked Feynman about the issue; in 1985 Feynman provided a quantum-mechanical model of reversible computing.
Various parties point out that Feynman's model only supports serial computation.	Margolus at MIT, 1990, demonstrates a parallel quantum model of reversible computing—but only with 1 dimension of parallelism.
People question whether the various theoretical models can be validated with a working electronic implementation.	Seitz and colleagues at CalTech, 1985, demonstrate working energy recovery circuits using adiabatic switching principles.
Seitz, 1985—Has some working circuits, unsure if arbitrary logic is possible.	Koller & Athas, Hall, and Merkle (1992) separately devise general reversible combinational logics.
Koller & Athas, 1992 – Conjecture reversible *sequential* feedback logic impossible.	Younis & Knight @MIT do reversible sequential, pipelineable circuits in 1993-94.
Some computer architects wonder whether the constraint of reversible logic leads to unreasonable design convolutions.	Vieri, Frank and coworkers at MIT, 1995-99, refute these qualms by demonstrating straightforward designs for fully-reversible, scalable gate arrays, microprocessors, and instruction sets.
Some computer science theorists suggest that the algorithmic overheads of reversible computing might outweigh their practical benefits.	Frank, 1997-2003, publishes a variety of rigorous theoretical analysis refuting these claims for the most general classes of applications.
Various parties point out that high-quality power supplies for adiabatic circuits seem difficult to build electronically.	Frank, 2000, suggests microscale/nanoscale electro-mechanical resonators for high-quality energy recovery with desired waveform shape and frequency.
Frank, 2002—Briefly wonders if synchronization of parallel reversible computation in 3 dimensions (not covered by Margolus) might not be possible.	Later that year, Frank devises a simple mechanical model showing that parallel reversible systems can indeed be synchronized locally in 3 dimensions.

This chart is unreadable on the projector screen, but that's OK. The point is just to show you that there have been many objections to reversible computing over the years from skeptics who thought that it was just too good to be true, and so it must be impossible for some reason. However, none of the many objections raised was ever proved rigorously, and in contrast, concrete counter-example constructions were found that clearly contradicted each of the objections. You would think that if reversible computing really were impossible, then at least some of the attempts to prove this would have borne fruit. In contrast, I believe that the best available concrete physical models of reversible computing that we have today (one of which I will show later) really do have *no* fundamental problems, and can indeed achieve significantly less than k ln 2 entropy generation per operation in practice. (Though perhaps not less by more than a technology-dependent constant factor.)



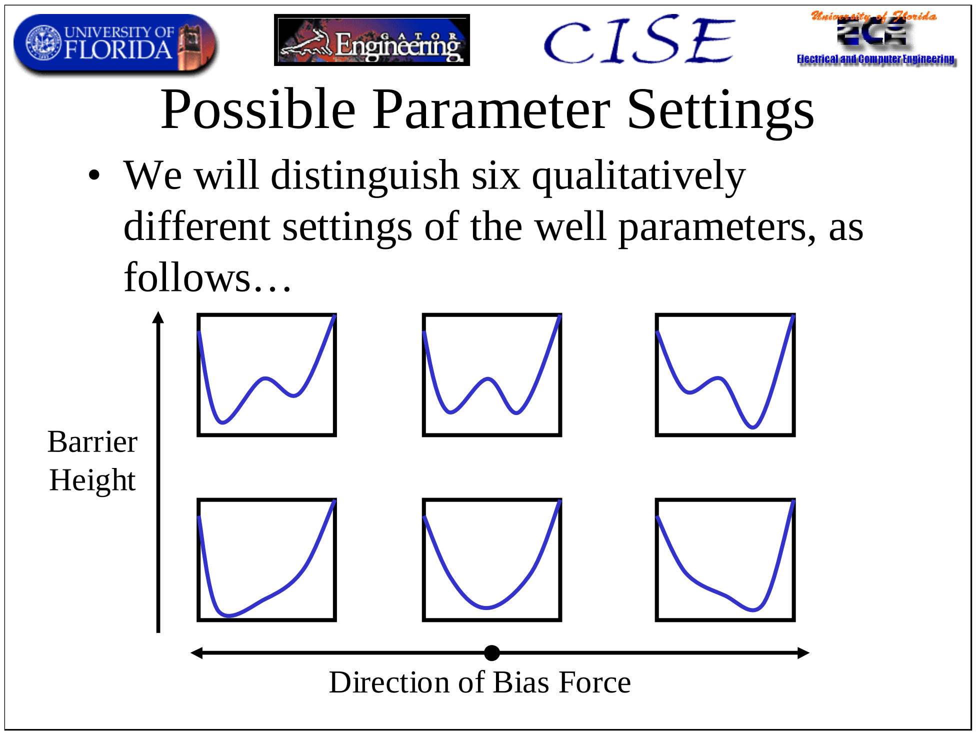
Bistable Potential-Energy Wells

- Consider any system having an adjustable, bistable potential energy surface (PES) in its configuration space.
- The two stable states form a natural *bit*.
  - One state represents 0, the other 1.
- Consider now the P.E. well having two adjustable parameters:
  - (1) Height of the potential energy barrier relative to the well bottom
  - (2) Relative height of the left and right states in the well (bias)



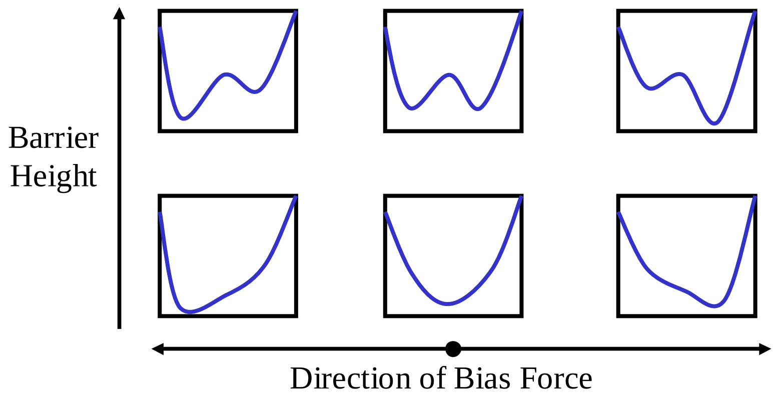
(Landauer ’61)

Let’s now develop a simple model of reversible computing. This one was discussed in Landauer’s original paper in 1961. Consider any system where there is a generalized “position” coordinate (configuration variable) and a potential energy surface as a function of that variable. Suppose this surface has multiple wells (local minima) separated by potential energy barriers. Then, the ground states of the wells form natural stable or at least metastable distinguishable states for the system. If there are two states, we get a natural bit. Now, suppose that the well is adjustable by externally-applied influences. We will suppose the following two characteristics of the well are separately controllable (next slide).



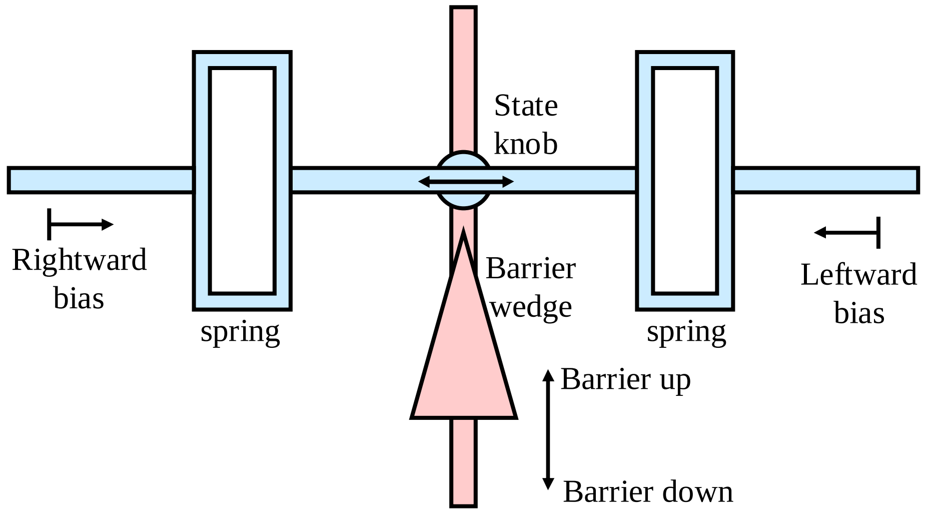
Possible Parameter Settings

- We will distinguish six qualitatively different settings of the well parameters, as follows…

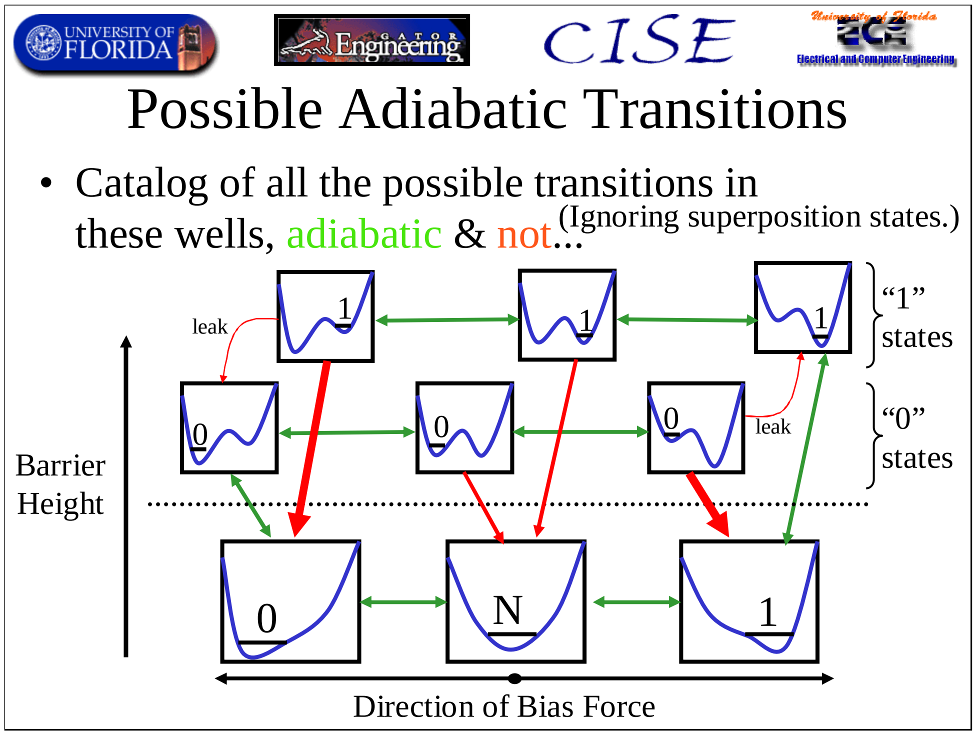


In the following, we will discuss 6 qualitatively different settings of the well parameters: the height of the barrier may be lowered or raised, and the direction of the bias force can be neutral (to there is equal ground state energy in both wells), biased to the left, or biased to the right. Of course one can continuously interpolate between these, but we will focus just on these 6 states.

One Mechanical Implementation



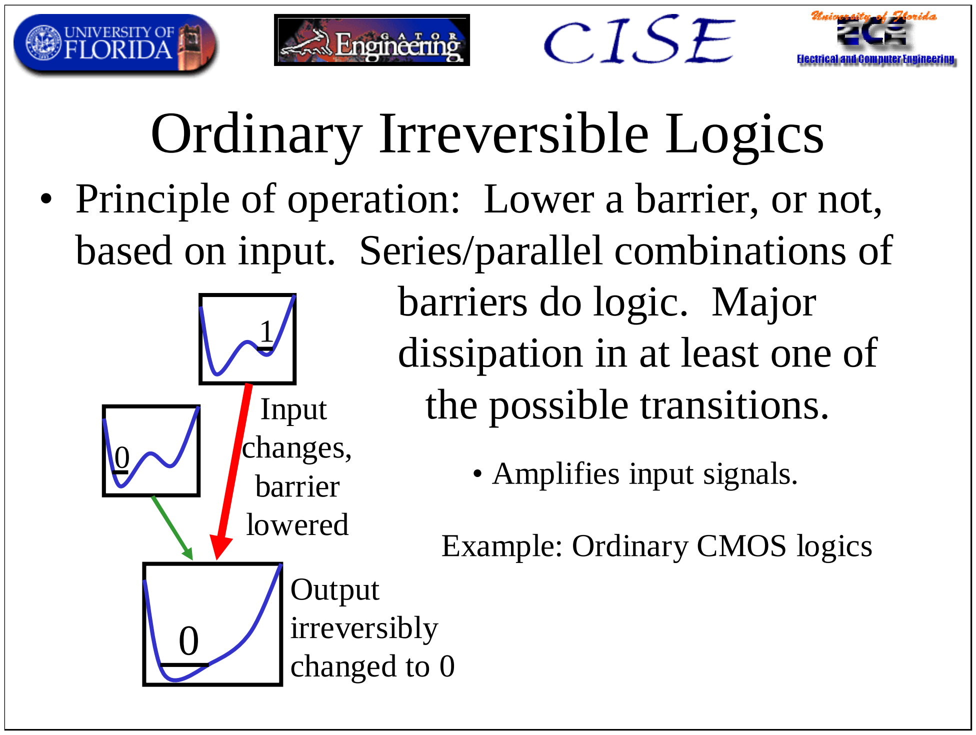
To illustrate the concept, here is a simple mechanical implementation of the bistable well. We have a knob on a rod which can be pushed to the right or the left, and a barrier “wedge” which we can push up to block the motion of the knob between left and right positions. So for instance, if you push on the left, the knob moves a little to the right, and then if you push the barrier up, you push the knob farther to the right and block its motion back to the left. Then, if the force on the left is released, the knob stays where it is since it is blocked by the barrier wedge.



Possible Adiabatic Transitions

- Catalog of all the possible transitions in these wells, adiabatic & not... (Ignoring superposition states.)

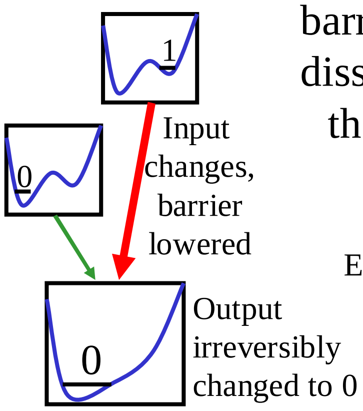
Now, given this potential well paradigm, it is easy to characterize what are all the possible adiabatic vs. non-adiabatic transitions between these qualitatively different states. For configurations with the barrier raised, we distinguish between configurations where the system is in the 0 (left) state – indicated by the lower-left graph in each pair of graphs in the top half of this diagram - and those where it is the 1 (right) state – indicated by the upper-right graph in each pair. In this diagram, the green arrows indicate possible adiabatic transitions. The red arrows indicate non-adiabatic transitions. The thick red arrows in which the amount of dissipation is determined by the energy difference between the two states, which would need to be several kT in order to guarantee that when a barrier is raised, the system will be on the biased side of the barrier with high probability. Note that in the middle column the energy difference between the states is zero, and as a result when the barrier is lowered sufficiently slowly the entropy generation can be made as low as k ln 2 (medium red arrows). In states where the barrier is raised and the system is in the metastable state, there will be a certain rate of leakage to the undesired state due to thermally-activated excitation over the barrier, and/or tunneling. This leakage however can be made arbitrarily small by making the barrier sufficiently high (and thick as well if desired). Thus we indicate leakage with a thin red arrow.



Ordinary Irreversible Logics

- Principle of operation: Lower a barrier, or not, based on input. Series/parallel combinations of barriers do logic. Major dissipation in at least one of the possible transitions.
  - Amplifies input signals.

Example: Ordinary CMOS logics

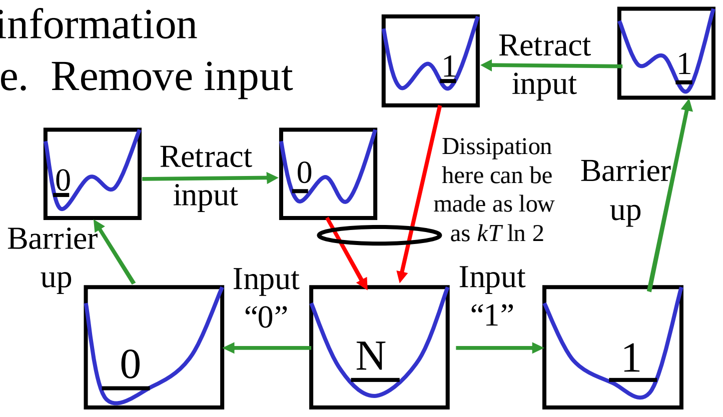

1
0
Input changes, barrier lowered
0
Output irreversibly changed to 0

Now, given this picture, it is straightforward to design all the irreversible and reversible classical logics and memory mechanisms that have been proposed to date in terms of how what kinds of transitions they make in this diagram of possible transitions, and how the change in well parameters is controlled.

First, in ordinary irreversible logics, such an ordinary irreversible CMOS, the procedure for doing logic is as follows. A potential energy barrier between two configurations (e.g. charge distributions in a circuit) is lowered conditionally by an input signal. Different series/parallel combinations of barriers controlled by different inputs implement Boolean logic (AND/OR combinations). The output state of the circuit unconditionally changes to a given state regardless of its previous value. Because this operation is logically irreversible, it must incur significant dissipation along at least one of the possible transitions. (In fact much more than kT if the energy difference between states is high, as it needs to be for data storage with high reliability.)

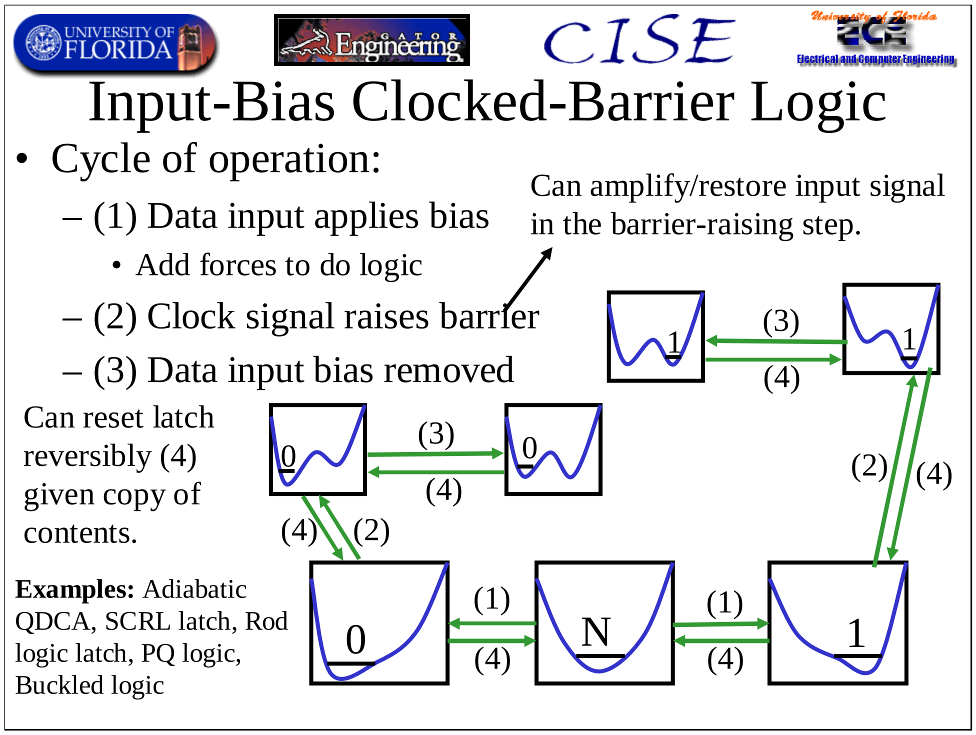
Ordinary Irreversible Memory

- Lower a barrier, dissipating stored information. Apply an input bias. Raise the barrier to latch the new information into place. Remove input bias.



Example: DRAM

Ordinary irreversible memory (e.g. a DRAM cell) works as follows. A potential energy barrier between states is lowered, dissipating stored information to entropy. (The entropy generation can be as low as k ln 2 if the energy of the two states is equalized and the barrier is lowered sufficiently slowly, though this is not usually done.) Then, an input signal applies a bias to the system towards the high or low direction. Then, the barrier is raised, latching the system into its new state, thus latching the information into the storage cell. Finally, the conditional input bias can be removed and the bias restored to a standard state. On the next cycle the storage cell can be overwritten with a new input.



Input-Bias Clocked-Barrier Logic

- Cycle of operation:
  - (1) Data input applies bias
    - Add forces to do logic
  - (2) Clock signal raises barrier
  - (3) Data input bias removed

Can amplify/restore input signal in the barrier-raising step.

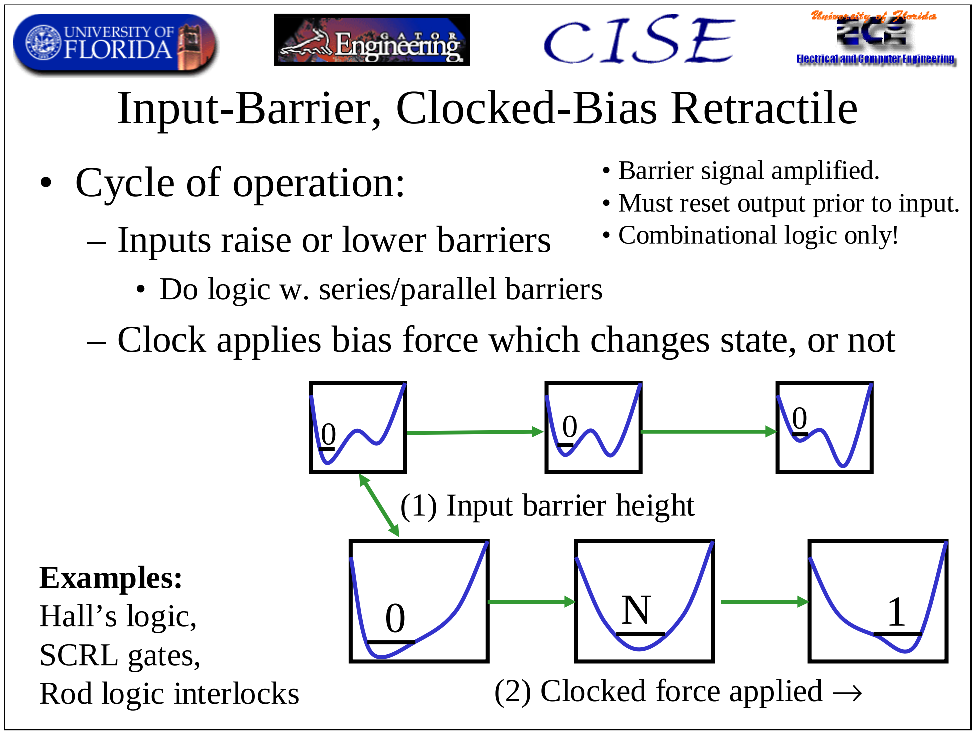
Can reset latch reversibly (4) given copy of contents.

Examples: Adiabatic QDCA, SCRL latch, Rod logic latch, PQ logic, Buckled logic

There is a very similar way to implement memory that is reversible. The only difference is that the memory is "erased" (actually, reversibly unwritten) by the reverse of the sequence that wrote it in the first place. It can also be used for logic, by adding together biases in step (1) to perform majority logic (which together with negation is universal). Also, when the barrier is raised in step (2) this can help amplify the strength of the stored signal.

There are a number of different physical implementations of adiabatic logic that have been proposed that use this scheme. These include the adiabatic variants of the quantum-dot "cellular automata" that have been studied by Lent, Tougaw and others mostly at Notre Dame; the SCRL ("Split-Level Charge Recovery Logic") technique for adiabatic CMOS logic, by Younis, Knight at MIT (later modified by myself and Margolus to fix a bug), Drexler's nanomechanical Rod Logic latch, Likharev's superconducting Parametric Quantron, and another mechanical logic called Buckled Logic by Merkle.

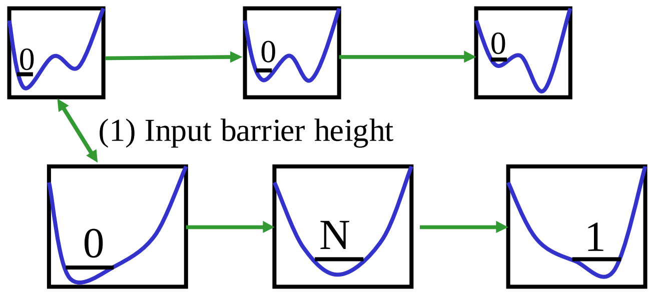
The scheme is logically complete for simulating efficient reversible circuits, given an appropriate set of timing signals. However, using majority logic is somewhat cumbersome, and slightly simpler techniques are possible.



Input-Barrier, Clocked-Bias Retractile

- Cycle of operation:
  - Inputs raise or lower barriers
    - Do logic w. series/parallel barriers
  - Clock applies bias force which changes state, or not

- Barrier signal amplified.
- Must reset output prior to input.
- Combinational logic only!



(1) Input barrier height

(2) Clocked force applied →

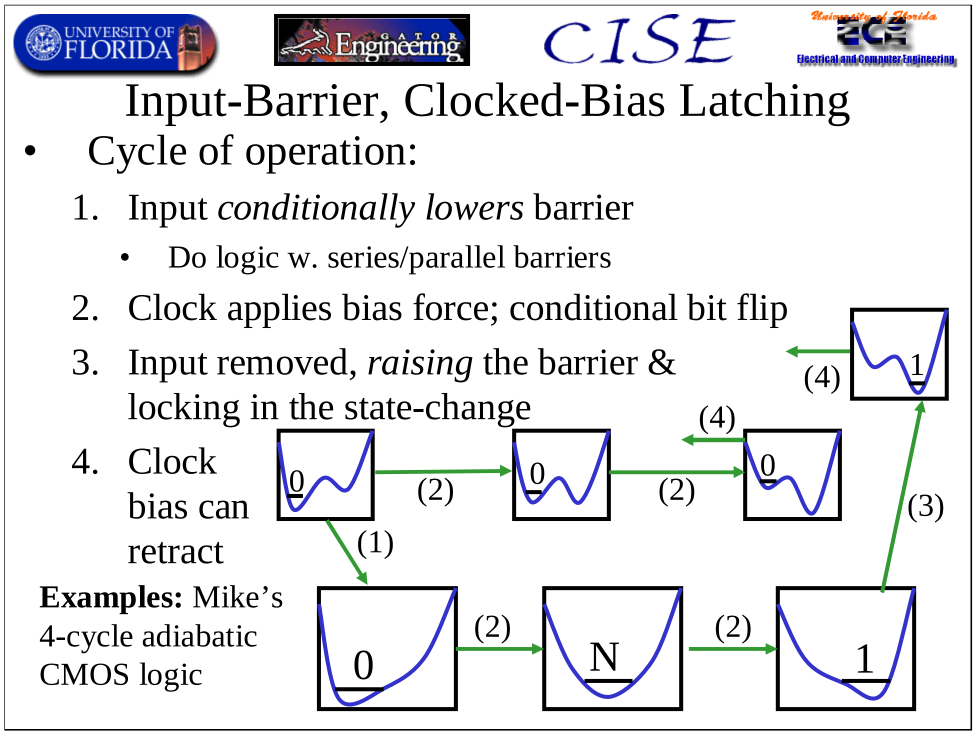
Examples:
Hall's logic,
SCRL gates,
Rod logic interlocks

Here is another scheme. This one has the disadvantage that by itself it can do combinational logic only.

The cycle of operation is that, starting from a biased state at the lowest energy level, the potential energy barrier is raised or lowered, depending on the input data. Then an unconditional bias force is applied, which either changes the state, or not, depending on whether the barrier was raised. Series/parallel combinations of barriers can determine whether a path to the other state exists or not. Note that the input signal (height of barrier) is amplified by the (possibly stronger) bias force.

A problem with this technique is that if one conceives of removing the input as having the effect of unconditionally lowering the barrier (which would cause dissipation), then one is not allowed to remove the input until downstream circuits have finished using the output and the output is retracted. Thus the technique is called "retractile cascade" (by Hall) and it has the unfortunate property that input circuits cannot be reused until downstream computations are finished. (It is thus equivalent to Bennett's 1973 embedding of irreversible Turing machines into reversible ones that temporarily take a unit of space for each operation performed by the original machine.)

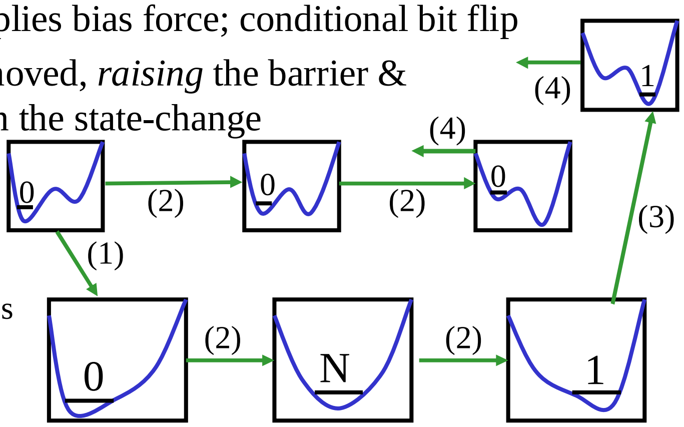
Several proposed adiabatic logic schemes used this method. J. Storrs Hall proposed an electronic logic using this technique together with (somewhat idealized) switches. Younis & Knight's SCRL technique did logic in this way (rather than by the majority logic approach mentioned earlier) in combination with input-bias, clocked-barrier latching used for reversible storage only. Drexler's Rod logic interlocks also worked in this fashion.



Input-Barrier, Clocked-Bias Latching

- Cycle of operation:
  1. Input *conditionally lowers* barrier
     - Do logic w. series/parallel barriers
  2. Clock applies bias force; conditional bit flip
  3. Input removed, *raising* the barrier & locking in the state-change
  4. Clock bias can retract

Examples: Mike's 4-cycle adiabatic CMOS logic



Finally, after drawing the other pictures (in Spring 2000) I realized that there is a very simple way to modify the previous technique to achieve the best of both worlds: Simple series/parallel logic together with a latching capability in a single mechanism. I call this Input-Barrier, Clocked-Bias Latching.

It is the same as the last slide, except that the standard default state when no input data is present is to have the barriers *raised*. Then, when the input comes in, it conditionally lowers the barrier. Logic is performed with series/parallel barriers as before. The clock applies the bias force, conditionally changing the state. But now, the input signal can be immediately removed (unlike the retractile cascade case) because this will not lower any barriers (which might have caused dissipation). Instead, the barriers are unconditionally raised (or kept raised if they were raised already), which locks in the new state. Now, the bias force can be retracted if desired. The storage element can be reset by the reverse procedure, possibly operating via a different path controlled by different inputs.

I had never seen a previous concrete reversible logic mechanism that used this approach, so I invented one. I will show you my original electronic implementation shortly. First, for clarity let me discuss the basic components of a simpler mechanical model.

Full Classical-Mechanical Model

The following components are sufficient for a complete, scalable, parallel, pipelinable, linear-time, stable, classical reversible computing system:

(a) Ballistically rotating flywheel driving linear motion.

(b) Scalable mesh to synchronize local flywheel phases in 3-D.

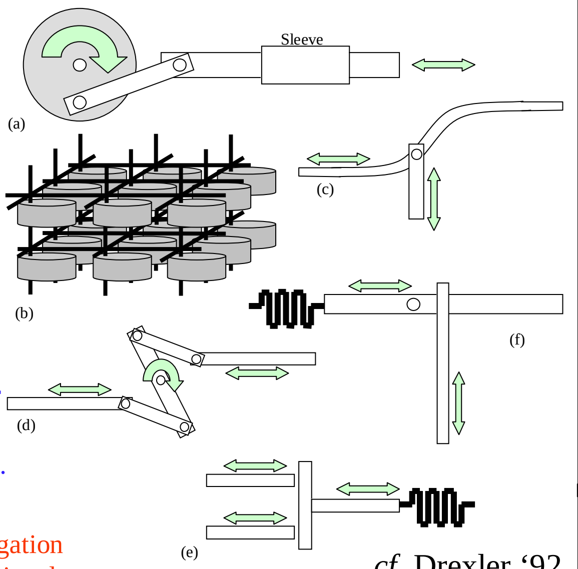
(c) Sinusoidal to flat-topped waveform shape converter.

(d) Non-amplifying signal inverter (NOT gate).

(e) Non-amplifying OR/AND gate.

(f) Signal amplifier/latch.

Primary drawback: Slow propagation speed of mechanical (phonon) signals.



cf. Drexler ‘92

Here is a full classical-mechanical model of reversible computing that avoids all the objections raised by various parties about previous approaches. Roughly the same mechanisms were described earlier by Merkle and Drexler in their various papers on reversible mechanical logics. The logic has the following properties which are desirable: (1) Complete – By this I mean universal for self-contained universal reversible logic, with no critical parts missing (such as resonators, or inverting logic). (2) Scalable – By this I mean that devices can be built out in 3 dimensional arrays and operated in parallel (up to the limit of leakage) (3) Parallel – Oh, I just said this. (4) Pipelinable – By this I mean that pipelined, sequential logic (with feedback) can be implemented (so long as it is reversible) (5) Linear time – By this I mean actually that it imposes no asymptotic spacetime overheads compared with any idealized reversible circuit scheme also based on local communications. (6) Stable – By this I mean there is no chaotic instability (accumulation of errors) in the configuration over time. (7) Classical – Oh, by the way it does not require quantum coherence. (8) Reversible – Asymptotically zero entropy generation in the adiabatic (low speed) limit.

Here are the components. Rods are implicitly constrained by rigid supports to move only in desired directions.

(a) Flywheel, rotating nearly ballistically (however we can reversibly replenish any adiabatic losses from a resonant power source). The wheel rotation couples to a roughly sinusoidal linear motion of a connected rod (like the connection between a steam engine piston and a locomotive wheel).

(b) Arbitrarily many such flywheels can be reversibly synchronized with each other in 3D via a simple mesh of rigid interconnecting rods. However I still need to investigate whether there might be scaling problems due to the following: The longer the rod, the larger the wavelength and lower the frequency and energy of undesired excited modes of vibration (e.g. compression/expansion modes) that the mesh can support. As these energies get smaller relative to the state where these modes are not excited (the desired rotation mode), the energy gap for purposes of the adiabatic theorem may get narrower and slower speeds may be required in order to keep the leakage of energy from the desired mode to the excited mode low.

(c) Custom-shaped track to convert roughly sinusoidal rod motions into flat-topped waveforms of translation of another rod, needed for purposes of some adiabatic logic schemes.

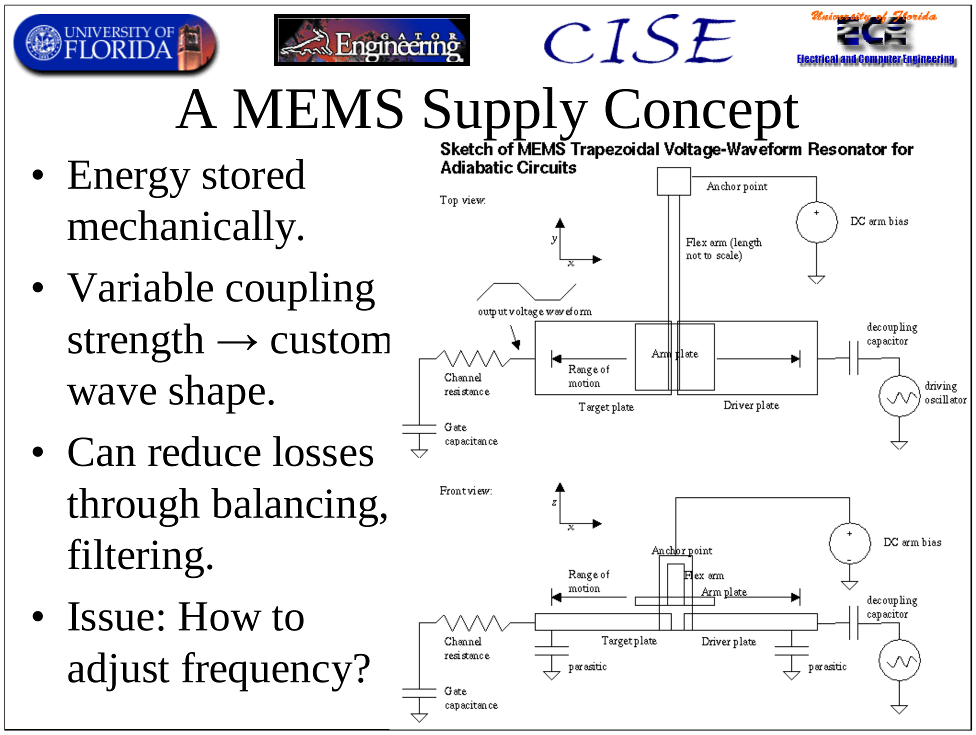
(d) Simple non-amplifying signal inverter (NOT gate).

(e) Simple non-amplifying OR/AND gate. If either of the rods on the left moves to the right, the rod on the right is pushed to the right, against the spring (squiggly line). Alternatively, if *both* of the rods move to the left, the rod on the right is pushed to the left by the spring.

(f) Reversible “interlock” (signal amplifier/latch, Drexler’s term), a simple variant of the bistable well mechanism on slide 18. Amplification mode: Input data pulls vertical rod down, or not. Then an unconditional signal from the left pushes the rod right, with protruding knob either being blocked by the vertical rod, or proceeding to the right. Latching mode: vertical rod is unconditionally lowered. Input data on horizontal rod moves the protruding knob either to the right past the vertical rod, or not. Vertical rod is raised, blocking the return path. Input force is removed but rod remains displaced if it was pushed past the vertical rod. Several protruding knobs with corresponding blocking rods can implement logic.

Not shown is a simple bistable well mechanism (like the one in slide 18) that can be used to perform bit erasure with only ~k ln 2 entropy generation if desired.

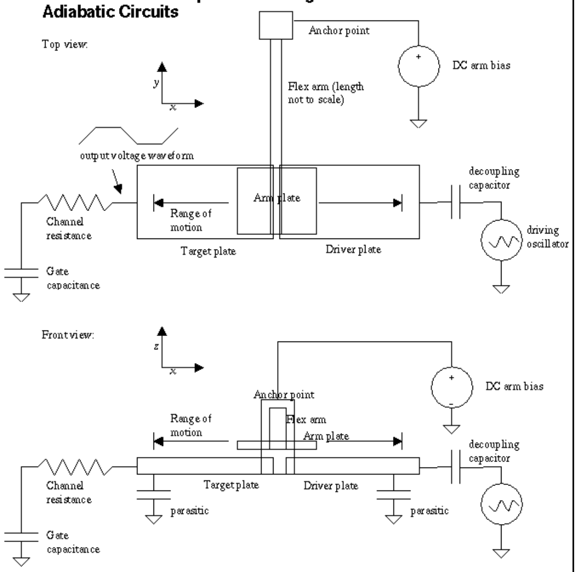
A drawback of the all-mechanical approach is the slow propagation speed of mechanical (phonon) signals, essentially the speed of sound in the material used. However, if we need faster signal propagation than this, we can always transduce a given mechanical signal to an electrical one, which can be propagated nearly losslessly along a shielded transmission line. (Hm, although at low frequencies it may be difficult to block the resulting RF signals with nanoscale-thickness shields!!)



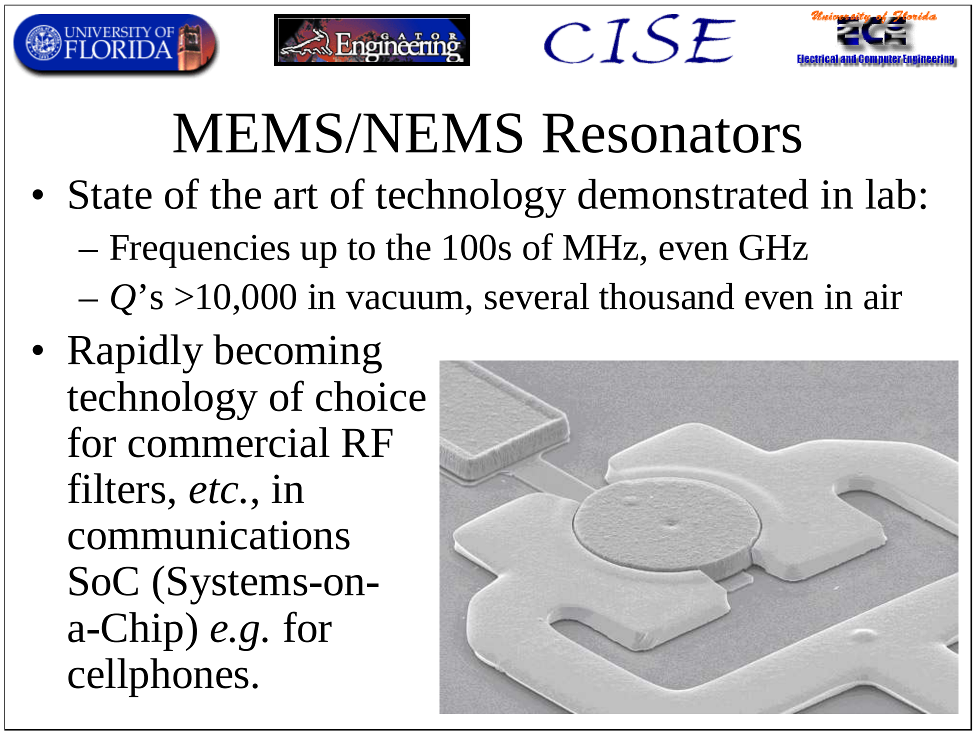
A MEMS Supply Concept

- Energy stored mechanically.
- Variable coupling strength → custom wave shape.
- Can reduce losses through balancing, filtering.
- Issue: How to adjust frequency?

Sketch of MEMS Trapezoidal Voltage-Waveform Resonator for Adiabatic Circuits

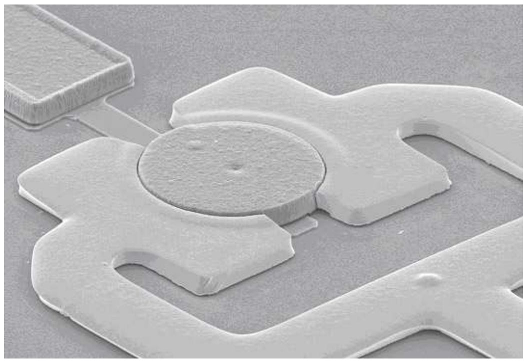


Here is another scheme in which the resonator is still mechanical (this is nice because some mechanical oscillators have very high Q) while the adiabatic logic can be electronic. We have an oscillating flexion-based spring (constrained to flex primarily in one dimension only) connected to one half of a parallel-plate capacitor. The resulting variable capacitance induces a variable voltage waveform on the output electrode, given a DC bias on the mobile plate.



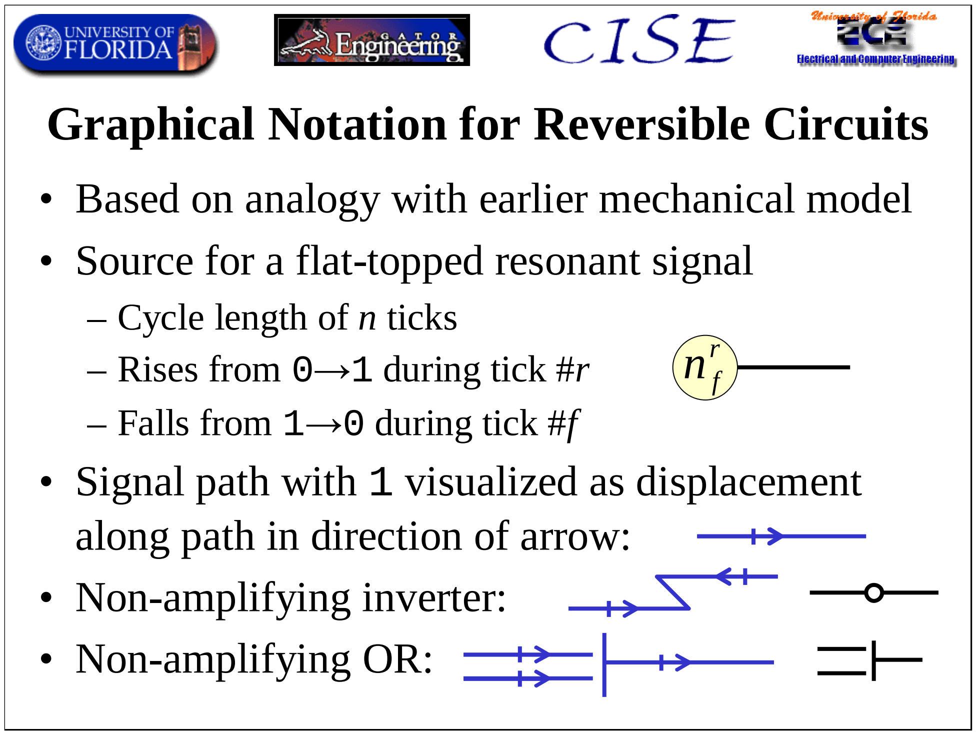
MEMS/NEMS Resonators

- State of the art of technology demonstrated in lab:
  - Frequencies up to the 100s of MHz, even GHz
  - *Q*’s >10,000 in vacuum, several thousand even in air
- Rapidly becoming technology of choice for commercial RF filters, *etc.*, in communications SoC (Systems-on-a-Chip) *e.g.* for cellphones.



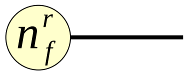
Here is a photo I stole off the web, of a MEMS disc resonator (operates in an expansion/contraction vibrational mode, in which there is a node of motion at the center support point, for low losses). Some resonator structures have been experimentally validated at frequencies up to hundreds of MHz and even GHz, with Q’s of up to and over 10,000 in vacuum, and several thousand even in air. This is today emerging as a real-world commercial technology for embedded systems-on-a-chip for wireless communications, e.g., chips in cellphones, which need high-Q resonators to build good filters and amplifiers.

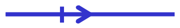
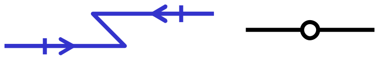
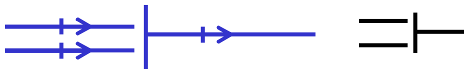
Perhaps we could do even better all-electronic resonators using superconducting technology, based on some of the superconducting quantum computing talks on Friday (multi-GHz systems with Q’s up to 100,000!!)



Graphical Notation for Reversible Circuits

- Based on analogy with earlier mechanical model
- Source for a flat-topped resonant signal
  - Cycle length of *n* ticks
  - Rises from `0`→`1` during tick #*r*
  - Falls from `1`→`0` during tick #*f*

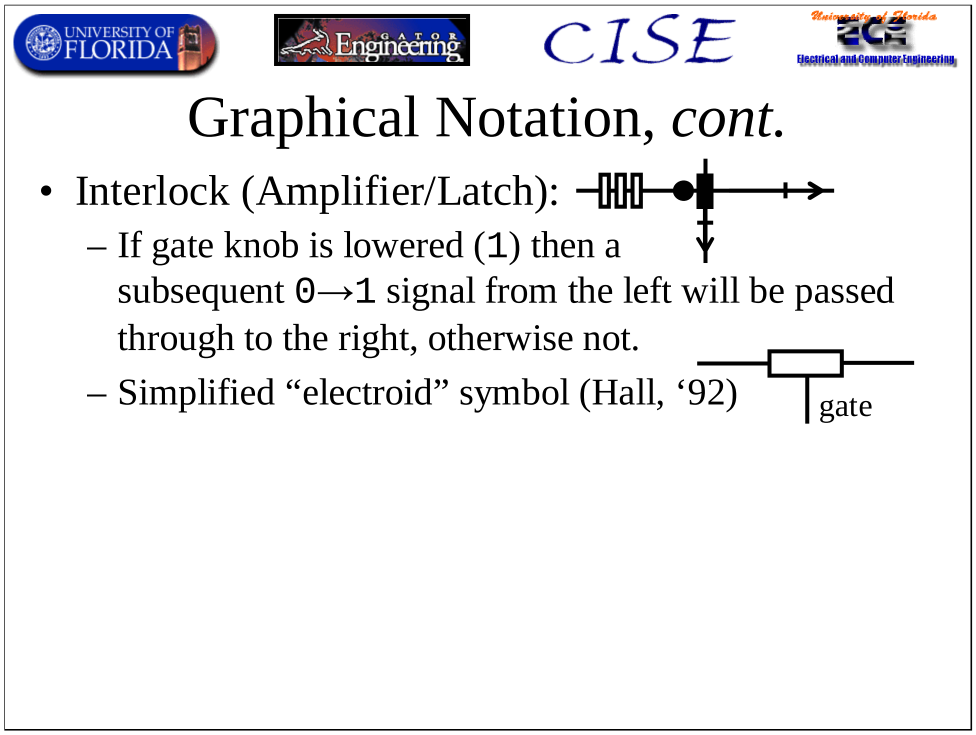


- Signal path with `1` visualized as displacement along path in direction of arrow: 
- Non-amplifying inverter: 
- Non-amplifying OR: 

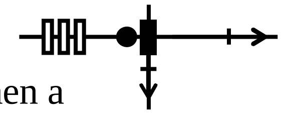
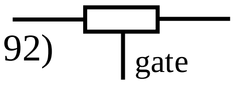
This slide is a sketch of an (unfinished) new graphical notation I am currently working on for describing reversible circuits. The lollipop symbol represents a source for a flat-topped resonant signal with a cycle length of n ticks (time units), which rises during the superscript tick (mod n) and falls during the subscript tick.

The line with a dot represents the non-amplifying reversible inverter.

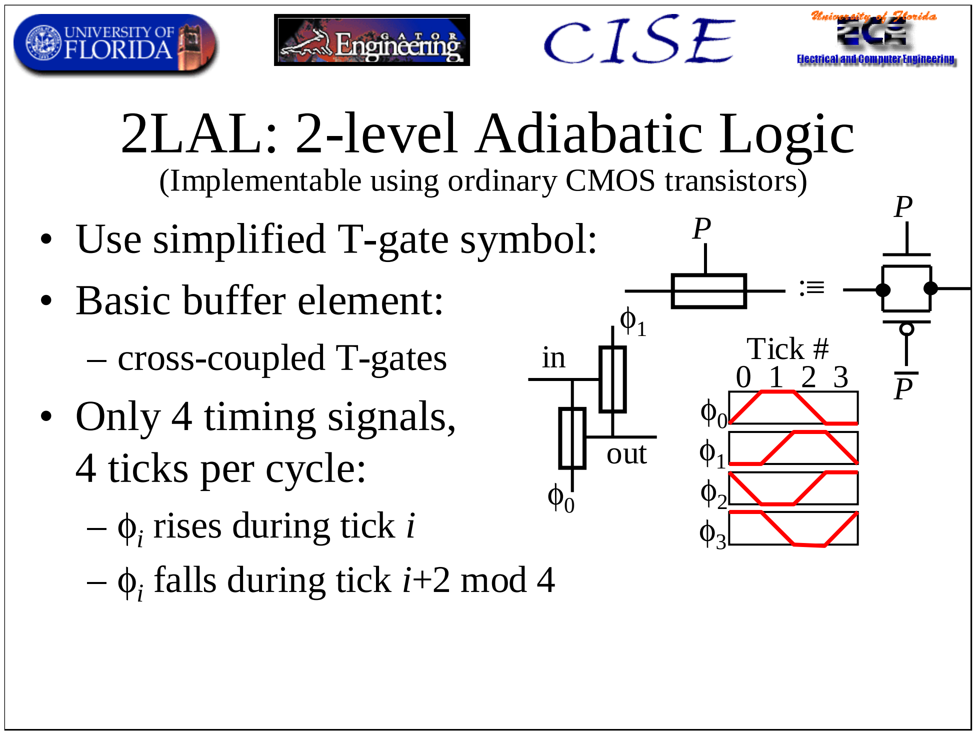
The piston-looking thing represents the non-amplifying OR/AND (however there is still ambiguity in the notation as to which it is, guess you could have it always be OR and add dots on the input and output to turn it into AND).



Graphical Notation, *cont.*

- Interlock (Amplifier/Latch): 
  - If gate knob is lowered (`1`) then a subsequent `0`→`1` signal from the left will be passed through to the right, otherwise not.
  - Simplified "electroid" symbol (Hall, '92) 

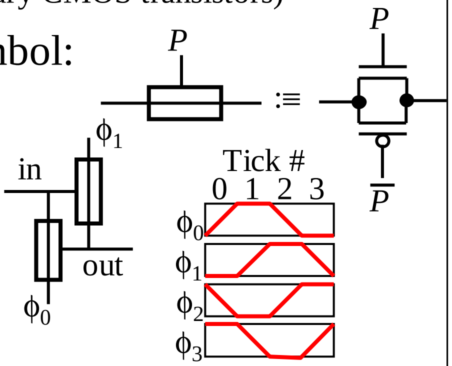
Here is a complicated symbol for the interlock. A simpler notation was suggested by Hall in his 1992 paper, a representation of an ideal switch in which if the gate signal is 1, the input is passed through to the output. This may be missing some information about the initial state, biasing, etc. that is implicit in the interlock, but I think this can be fixed. I am not yet quite sure if this set of symbols is sufficient, but I think that it is.



2LAL: 2-level Adiabatic Logic

(Implementable using ordinary CMOS transistors)

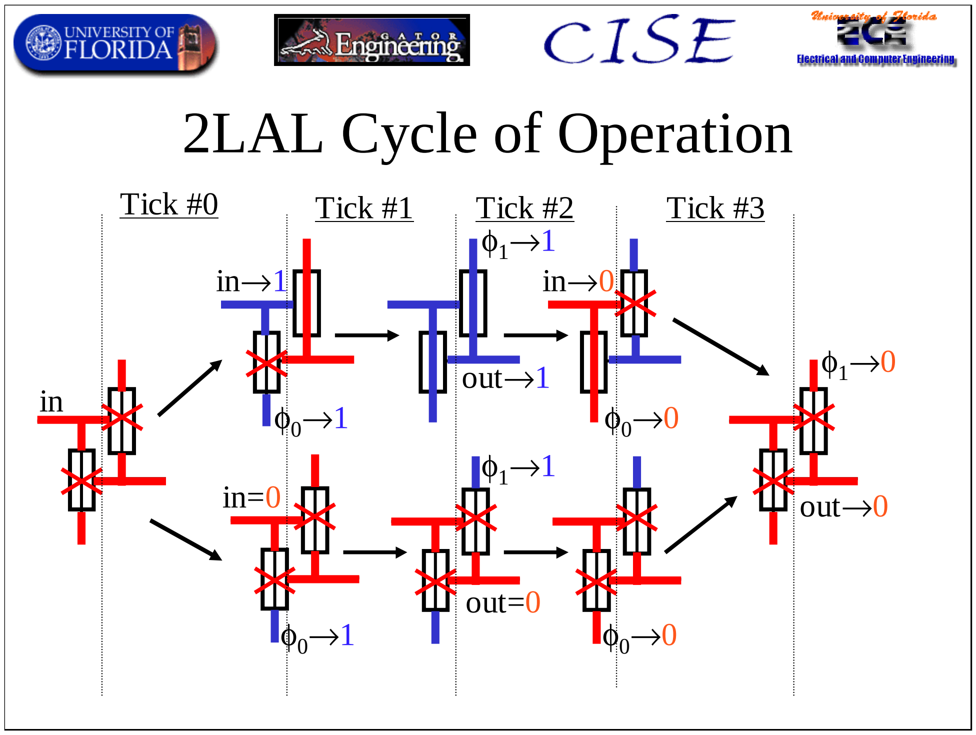
- Use simplified T-gate symbol:
- Basic buffer element:
  - – cross-coupled T-gates
- Only 4 timing signals, 4 ticks per cycle:
  - – ϕ_i rises during tick i
  - – ϕ_i falls during tick i+2 mod 4



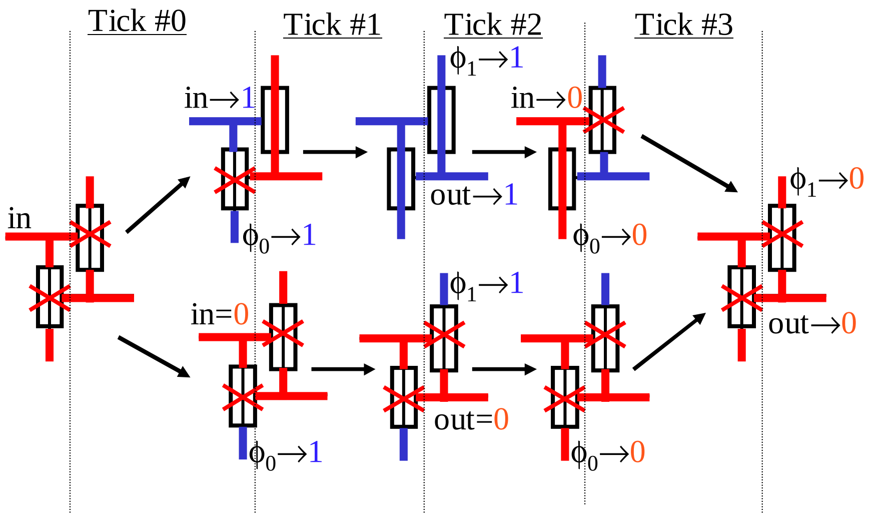
Anyway here is a simple adiabatic logic scheme, here demonstrated using ordinary CMOS transistors, that is based on the new operation paradigm discovered on slide 24 earlier (input-barrier, clocked-bias latching). For convenience we use Hall’s electroid (switch) symbol, which can be implemented in CMOS with a parallel nFET/pFET pair (transmission gate).

There is a basic clocked buffer element consisting of a pair of cross-coupled switches.

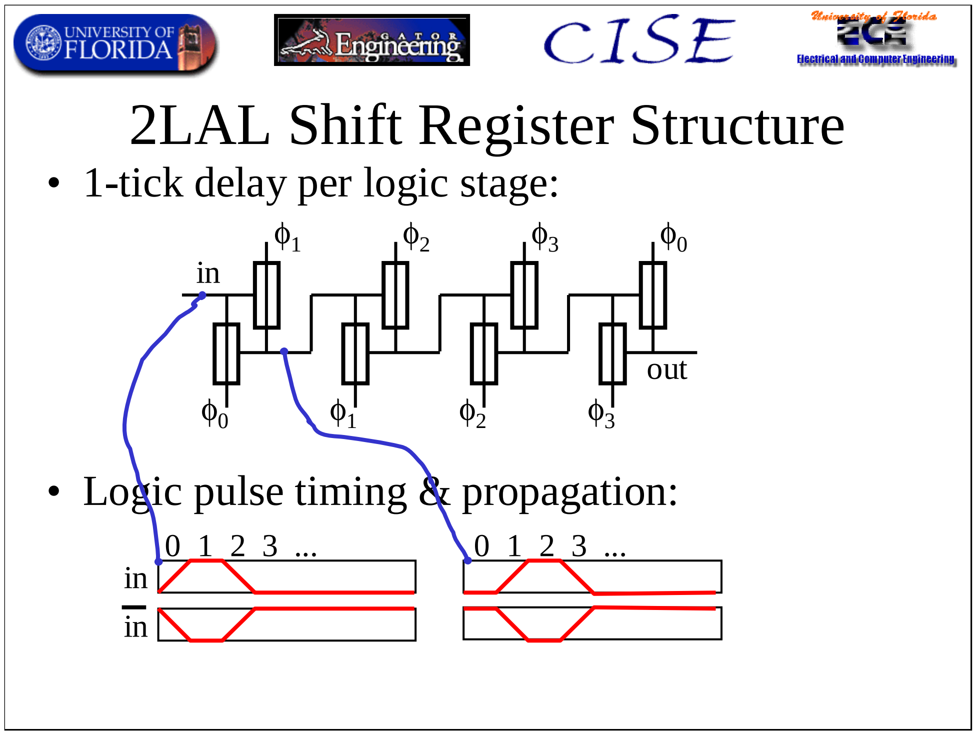
This logic scheme is more economic than many previous ones because it requires only 4 global timing signals, really just 4 different phases of a single waveform. These are shown in the timing diagram. The top and bottom portions must be flat for at least a full tick. The shape of the transitions is arbitrary (though the slope should be finite everywhere and scale down with increasing tick length).



2LAL Cycle of Operation

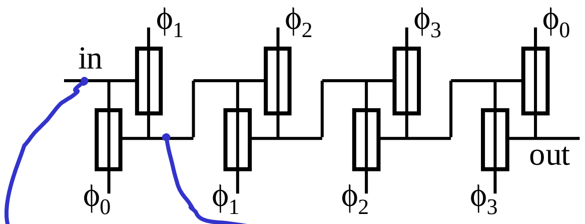


Here is the cycle of operation of the buffer gate in the 2LAL scheme. Initially, all signals are low (red, 0) and the switches are off. Then in tick 0, the input transitions to 1 (at the same time as phi_0), and the output switch turns on, or not (input conditionally lowers barrier). Now in tick 1, phi_1 goes high (unconditional bias) taking the output with it, or not. This turns on the reverse switch, or not. (If so there is no dissipation since the input is at the same level as phi_0.) In tick 2, the input is retracted from its source (and also simultaneously by phi_0 in the upper case), turning off the output switch (unconditional barrier raising). Now the output information is latched into place. Finally in tick 3 phi_0 reverts to its low state which does not affect anything inside the circuit but prepares us to be able to turn on the forward switch again in the next cycle. Meanwhile, the next gate in the chain restores the output to the zero level. (This particular gate is intended to be used as part of a pipeline of similar gates.)

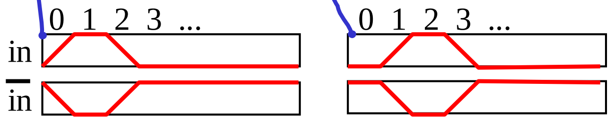


2LAL Shift Register Structure

- 1-tick delay per logic stage:



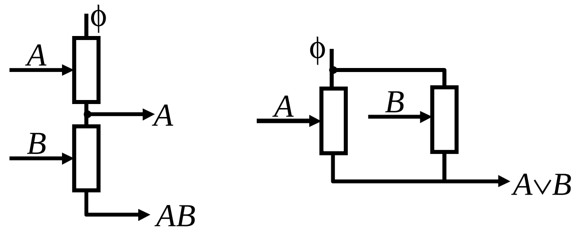
- Logic pulse timing & propagation:



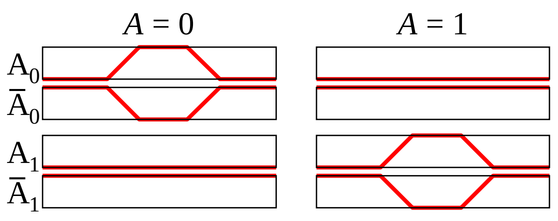
Here is how to build a shift register of 2LAL buffers: Just connect them together with incrementing phases on successive clock signals. A pulse introduced at the input will propagate down the chain, 1 stage per tick. If CMOS transmission gates are used for switches, then dual-rail logic must be used.

More complex logic functions

- Non-inverting Boolean functions:



- For inverting functions, must use quad-rail logic encoding:
  - To invert, just swap the rails!
    - Zero-transistor “inverters.”

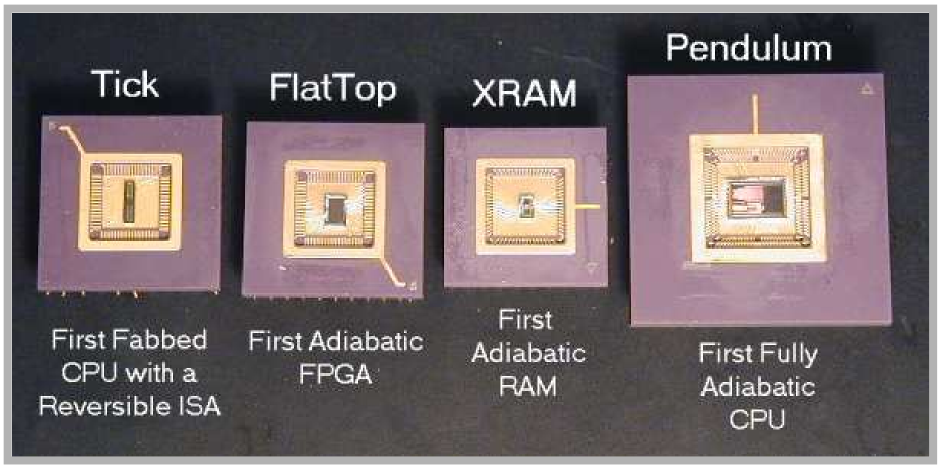


How about more complex functions? Again, series/parallel combinations of input-controlled switches will do the job. (Forward parts shown.) However, one must remember that information on internal nodes (such as the A output of the left circuit) must also be retracted by subsequent gates. Inputs that are not echoed in the output (e.g. B in both these examples) must be retracted seperately by some other circuit.

The easiest way to do inverting functions is to use a dual-rail encoding: a pulse on one wire represents 0, while a pulse on another represents 1. (Quad-rail encoding is shown since this is needed if switches are implemented using CMOS transmission gates.) Then a NOT gate is just a renaming of rails. Dual-rail has the further advantage of allowing the total magnitude of back-reactions on the resonant driver to be data-independent.

Reversible / Adiabatic Chips Designed @ MIT, 1996-1999

By the author and other then-students in the MIT Reversible Computing group, under AI/LCS lab members Tom Knight and Norm Margolus.

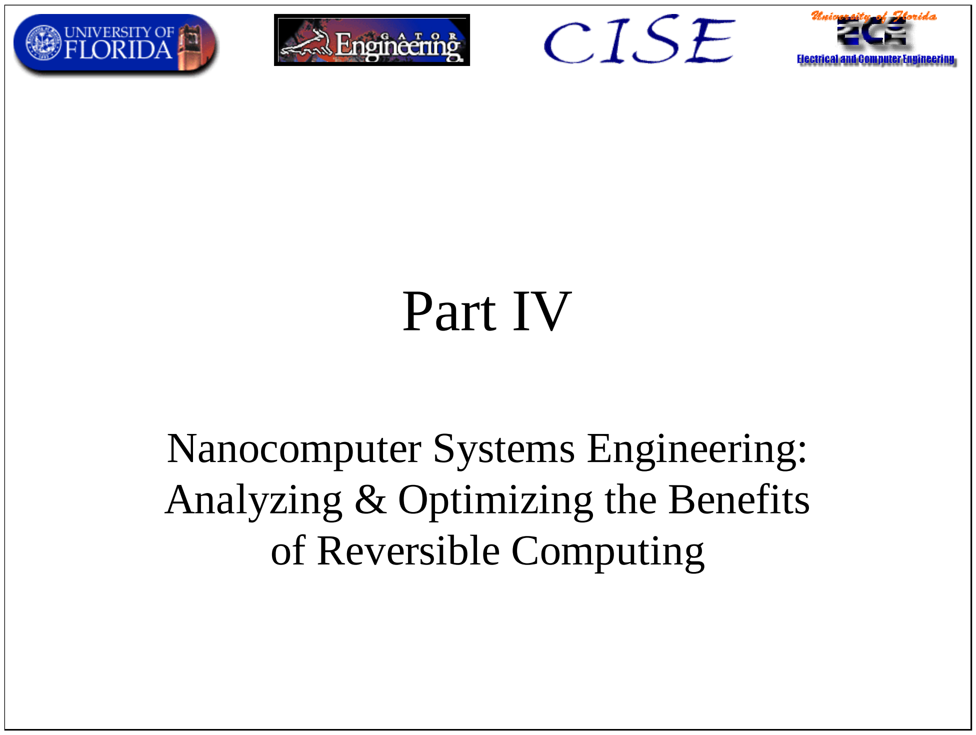

Tick
First Fabbed CPU with a Reversible ISA
FlatTop
First Adiabatic FPGA
XRAM
First Adiabatic RAM
Pendulum
First Fully Adiabatic CPU

So anyway, using another (more complicated, and buggy) adiabatic logic style, my collaborators and I built these reversible chips at MIT (under Tom Knight and Norm Margolus) to demonstrate that the architectural problems of reversible logic are straightforward to solve. Tick (by myself and Scott Rixner) was a simple non-adiabatic microprocessor implemented in standard CMOS that nevertheless implemented a logically reversible machine-language instruction set, as a proof-of-concept and basis for comparison with fully-adiabatic circuits. FlatTop (me, Nicole Love, Carlin Vieri, Josie Ammer) was the first scalable, universal, fully-adiabatic reconfigurable processor, capable of efficiently simulating 2D and 3D reversible circuits of arbitrary complexity when tiled in large arrays. (FlatTop works by simulating in SCRL the Margolus BBMCA cellular automaton which itself simulates Fredkin's BBM billiard ball model of reversible computing which itself simulates reversible logic networks composed of Fredkin gates, which themselves can simulate arbitrary reversible CAs – the simulation is so indirect that it is not very efficient, but it is universal for reversible CAs with "only" constant-factor overhead. Anyway it is just a proof of concept.) XRAM (Carlin, Josie) was an adiabatic memory with a reversible interface (though I have since invented far more efficient ones), and Pendulum (Vieri, with ISA mods from me) was a complete, fully-adiabatic, MIPS-style microprocessor.

This chip-design work (the Pendulum project, DARPA-funded under the Scalable Computing Systems Program) demonstrated that reversible logic is by no means incompatible with traditional styles of computer organization. It only requires a fairly minor translation of traditional architectural methods.

However, this work begged the question: Can reversible computing ever really be cost-effective? Can the overheads of reversible and adiabatic operation ever be outweighed by their energy savings?

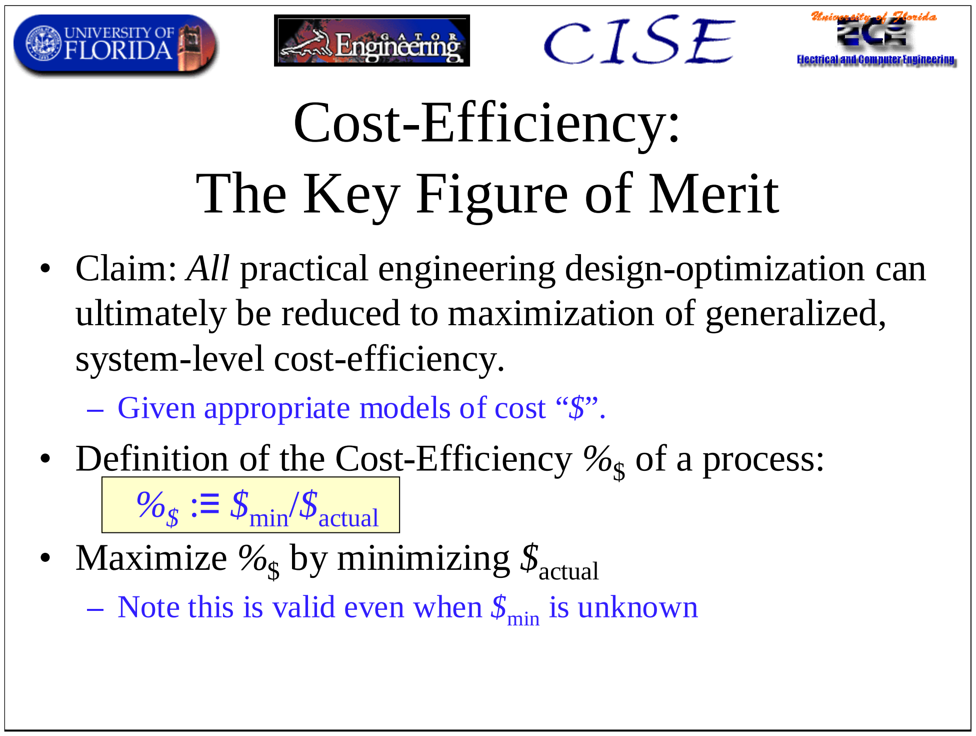
It was the goal of my subsequent research to answer this question, using a principled systems-engineering methodology which I will describe.



Part IV

Nanocomputer Systems Engineering:
Analyzing & Optimizing the Benefits
of Reversible Computing

In this part of the talk I discuss my recent work on what I call “Nanocomputer Systems Engineering” – my buzzword for analyzing and optimization the system-level cost-efficiency benefits to be gained for future nanocomputing from doing reversible computing.



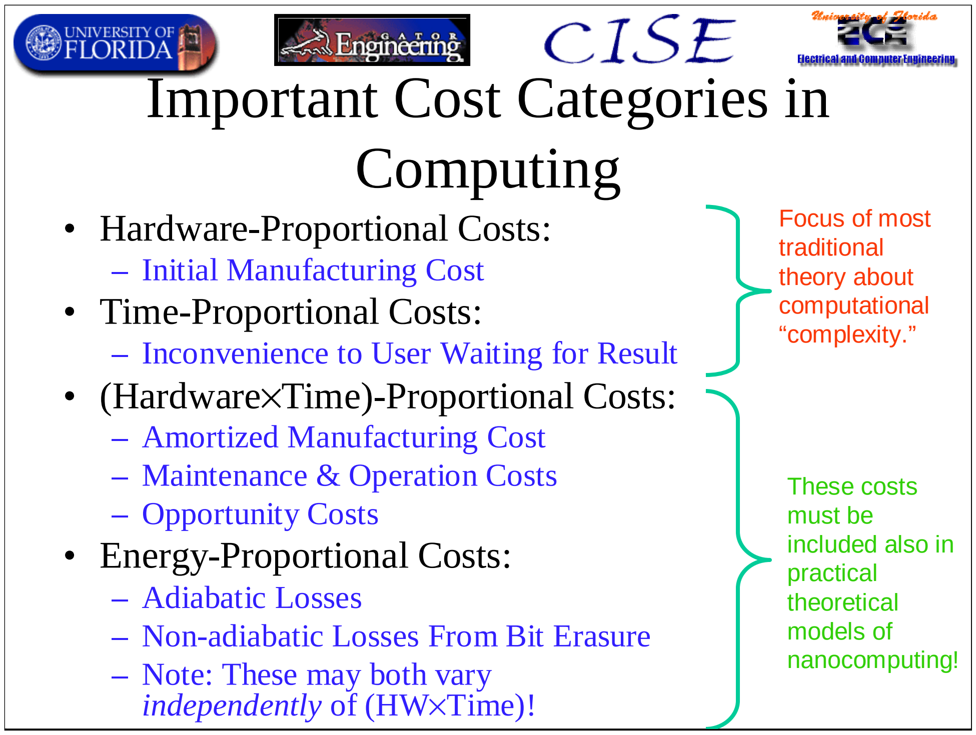
Cost-Efficiency: The Key Figure of Merit

- Claim: *All* practical engineering design-optimization can ultimately be reduced to maximization of generalized, system-level cost-efficiency.
  - Given appropriate models of cost "$".
- Definition of the Cost-Efficiency $\%_\$$ of a process:
  $\boxed{\%_\$:\equiv \$_{\text{min}}/\$_{\text{actual}}}$
- Maximize $\%_\$$ by minimizing $\$_{\text{actual}}$
  - Note this is valid even when $\$_{\text{min}}$ is unknown

Throughout this work, we rely on *cost-efficiency* as the ultimate figure of merit for any engineering design optimization problem. Even when it appears that we are optimizing something else, such as computer performance or energy efficiency, if we look closely at the reason *why* we are optimizing these things, we find that always there is an economic motive lurking in the background. Decision theory teaches us that we are always trying to maximize the utility (benefit minus cost) that we can obtain, given our available resources. Or, equivalently, we are always trying to minimize the *amount* of our available resources that must be used up – in other words, the cost – in order to achieve some given fixed amount of economic benefit as a result of achieving some goal.

Whenever we are minimizing cost to achieve a given result, we are maximizing something I call the *cost-efficiency* of the process used to obtain the result. *Efficiency* is in general the fraction of resources that are well-spent (given some definition of "well"). For example, the thermal efficiency of a heat engine is the fraction of heat that is transformed into useful work. *Cost-efficiency* is the fraction of the cost that we spend that actually leads directly to the desired result. We can say that this fraction is simply the *minimum* cost that could possibly yield the desired result. The ratio between minimum cost and actual cost is the fraction of cost that is well-spent.

Of course, the minimum cost may be difficult to calculate. However, we know that whatever value the minimum cost has, anything we can do to reduce the actual cost will help to improve our overall cost-efficiency.



Important Cost Categories in Computing

- Hardware-Proportional Costs:
  - Initial Manufacturing Cost
- Time-Proportional Costs:
  - Inconvenience to User Waiting for Result
- (Hardware×Time)-Proportional Costs:
  - Amortized Manufacturing Cost
  - Maintenance & Operation Costs
  - Opportunity Costs
- Energy-Proportional Costs:
  - Adiabatic Losses
  - Non-adiabatic Losses From Bit Erasure
  - Note: These may both vary *independently* of (HW×Time)!

Focus of most traditional theory about computational "complexity."

These costs must be included also in practical theoretical models of nanocomputing!

Now, in order to make progress in the cost minimization of any system, the first thing we need is a good model of what the primary sources of cost really are. In systems that perform computing tasks, in particular, we can identify a number of different sources of cost. In this slide, we categorize these costs according to the key parameters of the computation to which they are proportional.

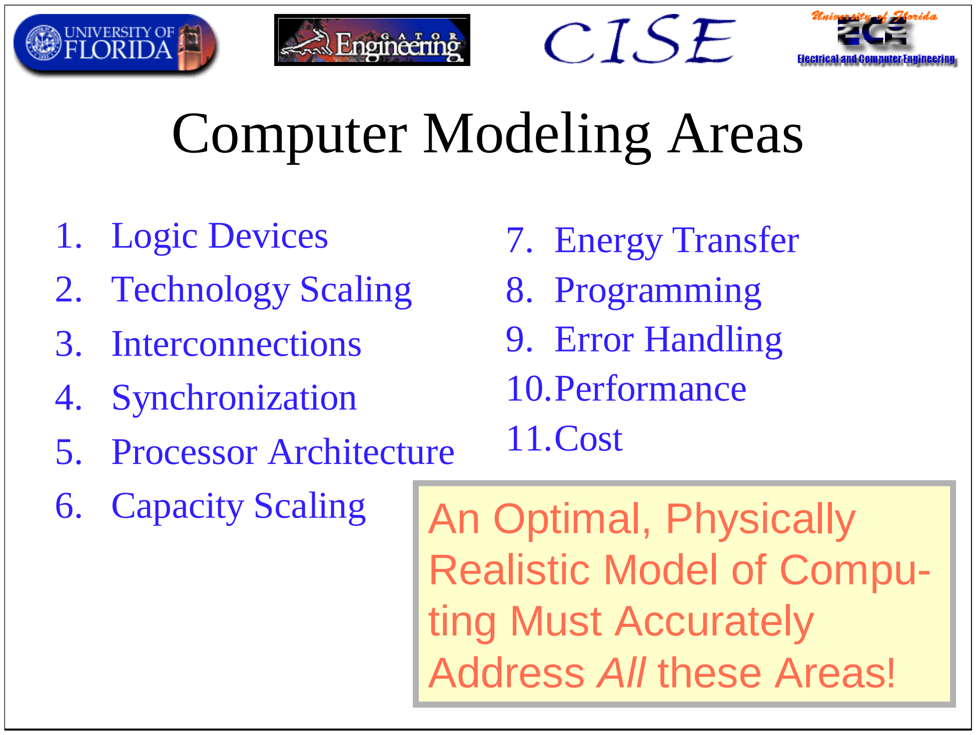
Of course, an obvious category of cost is computing is the cost to actually build a machine that has sufficient memory and storage capacity to carry out the desired computation. In a given technology, this cost is roughly proportional to the number of bits of capacity needed for the computation, and also to the amount of physical space (in cubic meters, say) occupied by the machine. In the traditional theory of so-called "computational complexity", they count the number of bits needed, and call this "space complexity."

Another obvious measure of cost in computation is the cost to the user, in terms of inconvenience, frustration, or lost profits or opportunities, of having to wait a larger amount of time before the result of the computation is available. Traditional complexity theory approximates this by the number of computational "steps" or machine clock cycles taken, and calls it "time complexity."

However, in the real world, both of these measures are inadequate. Space complexity by itself is misleading, because a machine can be reused to perform many computations, so not all of the manufacturing cost can be attributed to a single task that is performed. And, time complexity is inadequate, because it ignores the cost of using additional hardware. A more accurate measure of cost in computing is something I call "spacetime complexity." This is the *product* of the utilized capacity per unit time, multiplied by the amount of time used. This would be the cost to rent the needed capacity for the needed amount of time. Spacetime cost reflects the real manufacturing cost, amortized over the expected lifetime of the machine. It also reflects maintenance and operation costs, for example to repair or replace processors on a regular basis, or to provide a baseline level of standby power to all components in the machine. It also reflects the opportunity cost of *not* using the given capacity to perform some *other* computation.

Finally, another increasingly-significant component of costs are the energy-proportional costs. This includes the cost of free energy used (such as electrical energy) the inconvenience to the user of recharging or refueling for portable devices, and the costs associated with waste heat disposal. For example, in a major data center, safely releasing a certain number of megawatts (a not-unheard-of number today!) of waste heat into the atmosphere requires renting a proportional amount of land (planetary surface area), in order to provide sufficient room for the exhaust vents on the building's roof!

There is a contribution to energy cost that comes directly from spacetime cost, due to the standby power consumption of devices due to leakage, decay, or decoherence. However, other components of energy cost include adiabatic losses, which can scale down in proportion to the speed at which a computation is performed, and non-adiabatic losses, which result from the erasure of bits of information, and the discarding of their associated energy. Both of these sources of energy cost may scale independently of spacetime cost, and so they must be considered separately in any proper analysis of total system cost-efficiency.



Computer Modeling Areas

1. Logic Devices
2. Technology Scaling
3. Interconnections
4. Synchronization
5. Processor Architecture
6. Capacity Scaling
7. Energy Transfer
8. Programming
9. Error Handling
10. Performance
11. Cost

An Optimal, Physically Realistic Model of Computing Must Accurately Address *All* these Areas!

A good theoretical model of computing suitable for real-world systems-engineering should include model components for all of the key areas shown here.

• The logic device model addresses the very smallest functional components of the computer, which manipulate individual bits. These could be anything from conventional transistors, to nano-electromechanical switches, to spintronic transistors, to quantum logic gates.

• The technology scaling model tells us how the key characteristics of the logic devices change as technology progresses and fundamental physical characteristics are varied, for example as we scale them to smaller sizes, or operate them at higher or lower temperatures.

• The interconnect model deals with the characteristics of pathways for the flow of information through the machine. One valid way of handling interconnects is to treat them as a type of device. But interconnects cannot be ignored, they way that they were in most early models of computing.

• The synchronization or timing model tells us how device operations are synchronized with each other, so that they can exchange information. Current systems use global clock signals for synchronization, but local, “self-timed” designs are also feasible. The model must account for the means of disposal of any entropy in the timing sequence that is removed by the synchronization procedures.

• The processor architecture model describes how the machine’s circuits are organized so as to carry out complex computations.

• The capacity scaling model tells us how the architecture changes as the capacity of the machine is increased. Usually, we take a simple multiprocessor approach of simply adding more identical copies of the same kind of processor. (Preferably, connected to at most a finite number of immediate neighbors via bounded-length connections – e.g. a mesh.)

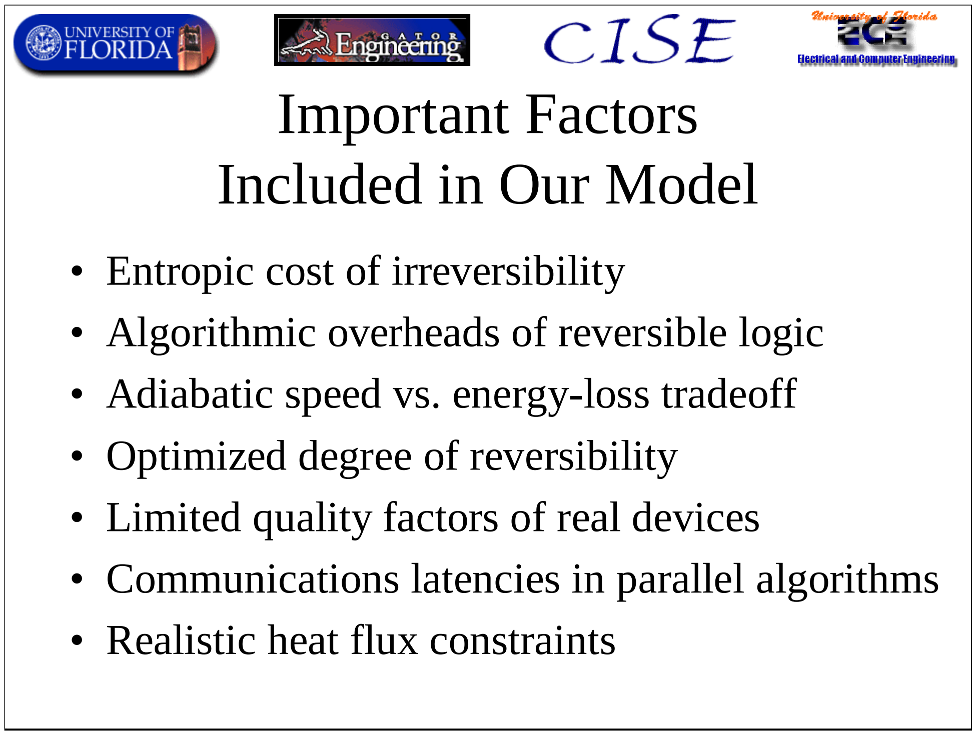
• The energy transfer model describes the distribution of temperatures, and the detailed flow of free energy and waste heat, or equivalently, known information and entropy, through the machine.

• The programming model describes how the architecture can be programmed to carry out arbitrary desired computations.

• The error handling model tells how dynamically-occurring errors or faults are detected and corrected, and how static defects are detected and worked around. The error handling model must also account for how the entropy of the corrected errors is disposed of.

• The performance model allows us to predict how the execution time of a well-defined algorithm will scale as we increase the size of the computational task to be performed, and accordingly the capacity of the machine needed to solve it.

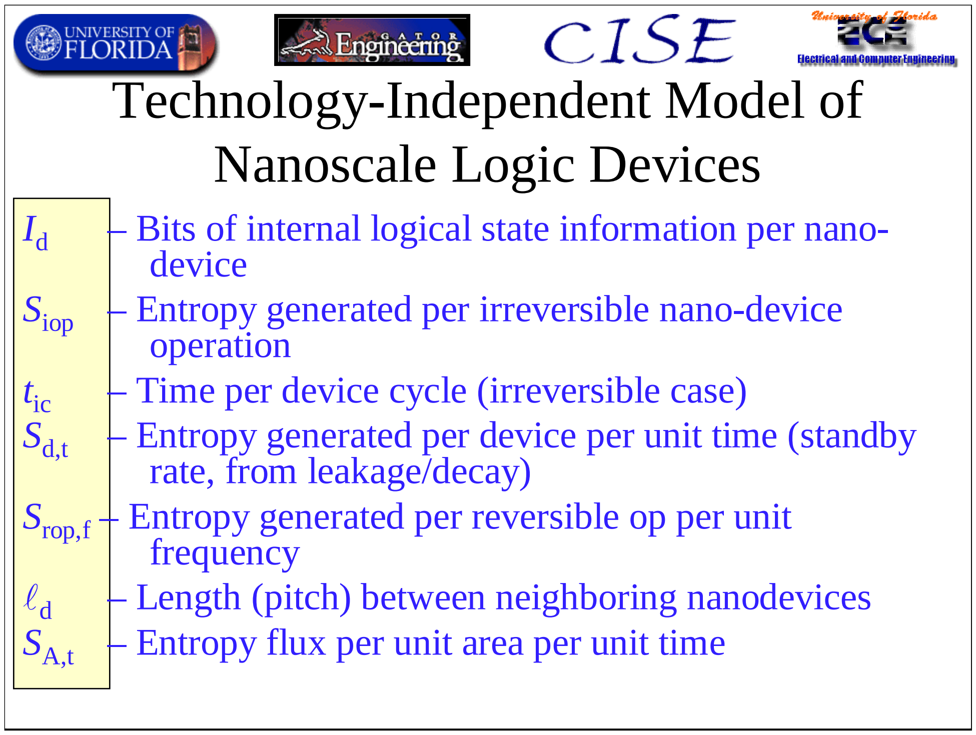
• Finally, the cost model integrates all the important components of cost described earlier to accurately assess the overall cost of a computation, including both spacetime-proportional costs and energy-proportional costs.



Important Factors Included in Our Model

- Entropic cost of irreversibility
- Algorithmic overheads of reversible logic
- Adiabatic speed vs. energy-loss tradeoff
- Optimized degree of reversibility
- Limited quality factors of real devices
- Communications latencies in parallel algorithms
- Realistic heat flux constraints

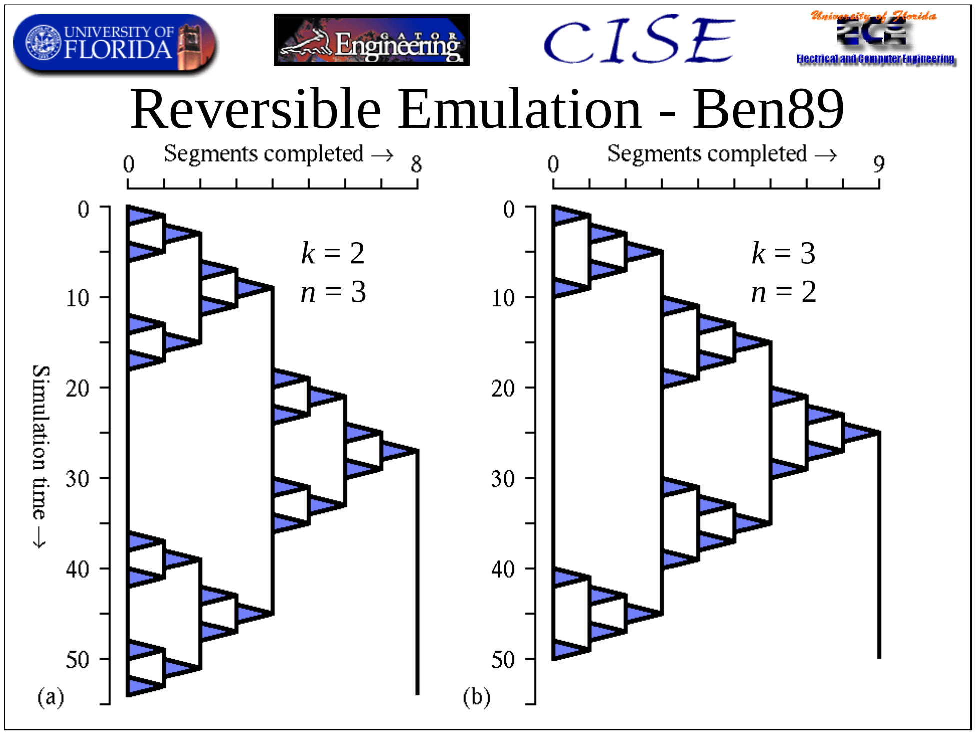
The model analyzed in this work incorporates all of the important factors listed here. This has never been done before. Most traditional models of computing, such as Turing machines, ignore the entropic cost of irreversibility. Some early discussions of reversible logic, such as those by Fredkin and Drexler, ignored the issues of the algorithmic overheads, as well as the adiabatic speed vs. energy-loss tradeoff. Work by Bennett addressed the algorithmic overheads, but did not reveal how to optimize the degree of reversibility, to trade off the algorithmic overheads against the energy savings, to maximize overall cost-efficiency. Finally, most work in reversible computing ignores the limited quality factors achievable in real devices, which always suffers some non-zero baseline rate of entropy generation due to leakage, decay, and error processes. Our model incorporates leakage. Finally, there is an important tension between the communication requirements of parallel algorithms, which want processors to be packed close together as compactly as possible (*i.e.* in 3 dimensions), versus the heat flux constraints of cooling systems, which want processors to be spread farther apart, over (at most) a two-dimensional surface. In this work, we optimized both reversible and irreversible computers in the face of all these constraints, in order to make the best possible case for both kinds of machines, so as to make a fair comparison between them.



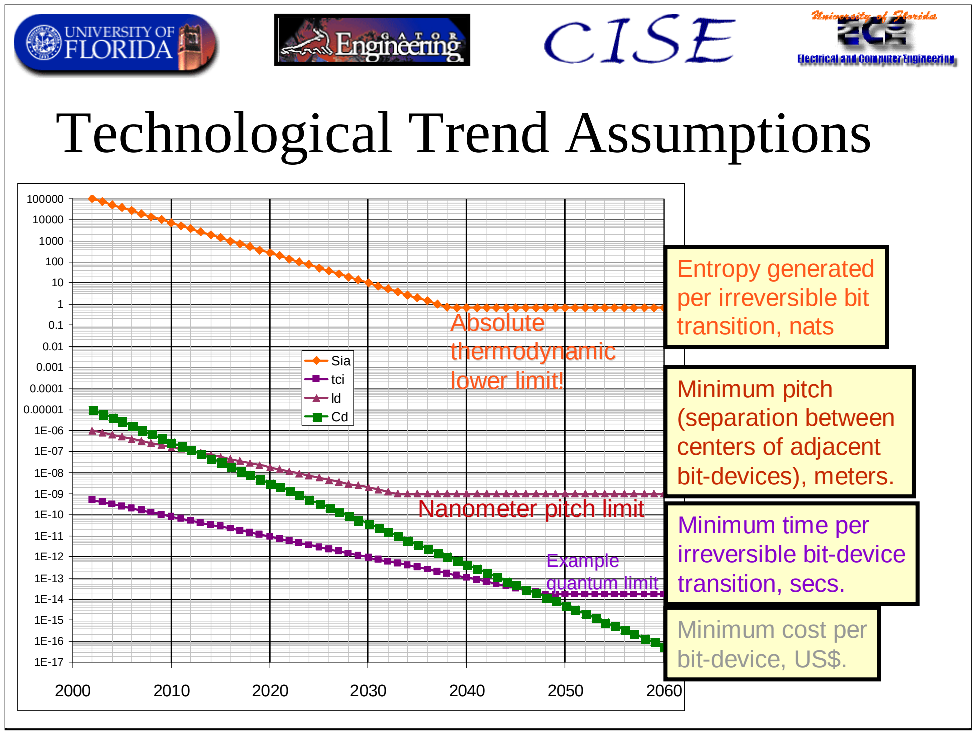
Technology-Independent Model of Nanoscale Logic Devices

I_d	– Bits of internal logical state information per nano-device
S_{iop}	– Entropy generated per irreversible nano-device operation
t_{ic}	– Time per device cycle (irreversible case)
$S_{d,t}$	– Entropy generated per device per unit time (standby rate, from leakage/decay)
$S_{rop,f}$	– Entropy generated per reversible op per unit frequency
ℓ_d	– Length (pitch) between neighboring nanodevices
$S_{A,t}$	– Entropy flux per unit area per unit time

Our model includes some treatment of all the modeling areas described earlier. Just to give an example of one of the model components, here are the key external characteristics of nanoscale logic devices in our model. The focus here is on size (ell$_d$), capacity (I_d), raw performance (reciprocal of t_{ic}), and energy consumption, expressed in entropy units. Notice that there are four different S parameters, relating to the thermodynamic characteristics of the device. From top to bottom, S_{iop} is used to model the non-adiabatic entropy generation from bit erasure. S_{dt} is used to model the non-adiabatic entropy generation due to energy leakage. $S_{rop,f}$ models the adiabatic entropy generation from reversible operations. And $S_{A,t}$ expresses the limited entropy-removal capabilities of the cooling system technology. All other important performance characteristics of our devices are derived from the parameters listed here.



One particularly interesting aspect of this optimization problem was optimizing the reversible algorithm to maximize cost-efficiency in the worst-case scenario when we don't know how to do any better than by using Bennett's general 1989 algorithm. The algorithm works by recurisvely doing and undoing subcomputations of different sizes and making checkpoints. The algorithm has two important parameters: *n*, the number of recursive levels, and *k*, the number of forward "recursive calls" to the next lower level from a given level. Here is an illustration of the algorithm for two example values of *n* and *k*. Note the different space/time tradeoff in the two cases. The choice of these parameters affects the frequency of dissipation from irreversible operations, the spacetime costs per op simulated (including leakage costs), and the average number of reversible operations per op simulated which affects adiabatic energy losses. In addition the speed of the adiabatic operations was simultaneously optimized together with the algorithm parameters for maximum overall cost-efficiency.



Technological Trend Assumptions

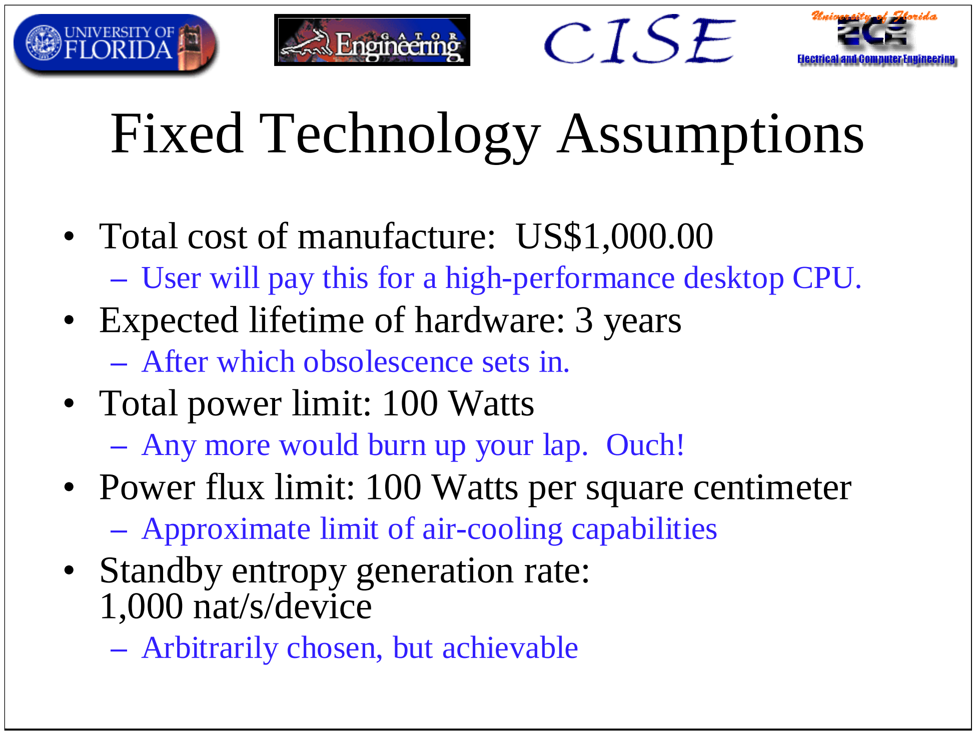
Now, in order to actually make any kind of statement about the timing of the emergence of the usefulness of reversibility, we had to make some assumptions about how various raw parameters of the device technology would change as a function of time over future decades. Although obviously it is difficult to forecast these developments exactly, there are some strong, steady historical trends, as well as some clear limits to these trends, that together allow us to sketch out a technology scaling model with some confidence in its approximate correctness.

The upper red line shows the entropy generated per irreversible bit erasure, in units of Boltzmann's constant k. Today in 0.1-micron CMOS technology, this is about 100 thousand. Calculations based on the International Technology Roadmap for Semiconductors show that the industry wants this to decline by 28% per year in the future (historically it has decreased somewhat faster). At this rate, it would reach the absolute thermodynamic minimum of about 0.7k by about the year 2038.

Next, the mahogany line shows average device pitch, or separation between the centers of neighboring devices. This is about 1 micron today if you include space for interconnects. The standard Moore's Law trend is for pitch to decrease by a factor of 2 every 3 years (so that density doubles every 18 months). We assumed that 1 nm (just ~3-4x larger than atomic diameters) is an absolute minimum. This will be reached by about 2033.

The purple line shows clock period, which is about half a nanosecond today and decreases at about the same rate as pitch. The quantum maximum frequency is about half a PetaHertz per electron volt, giving a minimum period of about 2 femtoseconds if we assume no more than 1 eV of energy per bit. The maximum voltages achievable across nanometer-pitch or smaller structures are on the order of a volt, because molecular structure breaks down at much higher voltages than this. (Molecular ionization energies are on the order of a few eV.)

Finally, the green line shows cost per bit-device. The cost per device is on the order of a thousandth of cent today. For example, an Intel Itanium 2 microprocessor with 220 million transistors probably costs on the order of 2200 dollars or less. Moore's Law has cost-per-device decreasing by about half every 18 months. We assume this trend can continue indefinitely, due to improvements in 3D nanomanufacturing technology (self-assembly, nanofabrication, etc.), even after the pitch limit is reached. We should note that even if cost per device does not continue decreasing after devices reach a minimum size, our results will still end up favoring reversible computing.



Fixed Technology Assumptions

- Total cost of manufacture: US$1,000.00
  - User will pay this for a high-performance desktop CPU.
- Expected lifetime of hardware: 3 years
  - After which obsolescence sets in.
- Total power limit: 100 Watts
  - Any more would burn up your lap. Ouch!
- Power flux limit: 100 Watts per square centimeter
  - Approximate limit of air-cooling capabilities
- Standby entropy generation rate:
  1,000 nat/s/device
  - Arbitrarily chosen, but achievable

In addition to these assumptions about changing technology parameters, we made the following assumptions about parameters which are held constant for one reason or another.

We held total manufacturing cost constant at $1,000, on the assumption that individuals will always be willing to pay about this much for a desktop computer or laptop. This figure has not been adjusted for inflation.

We hold the expected lifetime of the hardware to be about 3 years, since in this time the original machine would have lost most of its original value anyway (specifically, ¾ of it), due to the assumed cost trend.

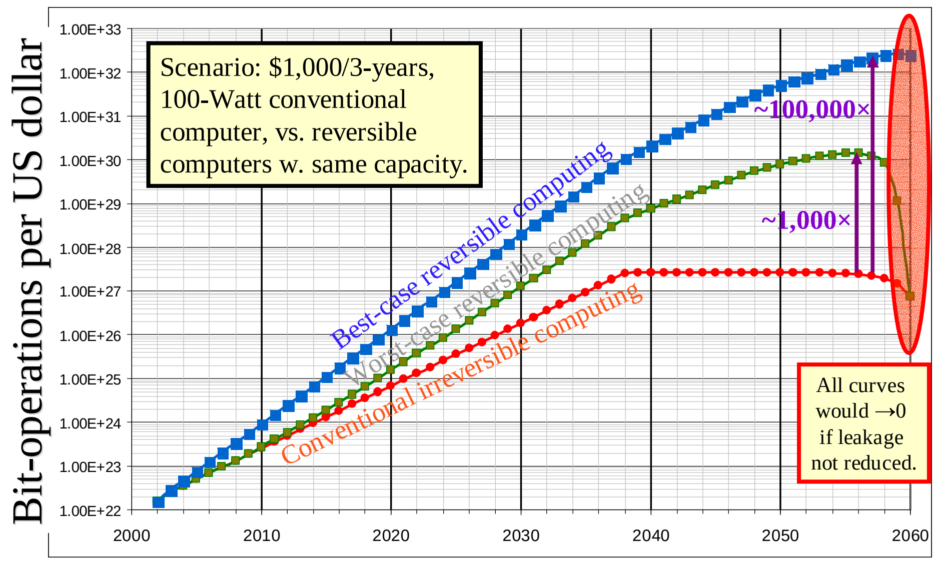
We set a total power limit of 100 Watts, to model the case of a machine that is held in the user's lap and thus cannot get rid of much more waste heat than this without the user experiencing some discomfort (or being annoyed by a noisy fan, think of a 1kW hairdryer).

We set a heat-flux density limit of 100 Watts per cubic centimeter, since this is roughly the most that can be achieved using ordinary air-cooling capabilities. (Actually probably the practical air-cooling limit is even less than this.)

Finally, we model a standby entropy generation rate of 1,000 nats/s/device. This fits the time constant for decay of a DRAM circuit node today which is about 1 millisecond. If a storage node were set at a low voltage level holding just a few nats of physical information, this would then yield the given rate. However, keeping this low of a rate as devices shrink to smaller sizes is a major challenge for nano-device technology. But we know it is possible, since for example Drexler's original mechanical rod-logic interlocks have essentially zero rate of standby entropy generation at room temperature, due to the high energy barriers presented by the steric intermolecular interactions between rigidly-bonded carbon-based structures. However, whether we can truly maintain this low rate in an all-electric or electromechanical technology at nanometer length scales is somewhat of an open research question. This may the most unrealistic assumption in our current model. I would like to invite other researchers to help me develop a more refined scaling model for this parameter, to see how it would affect the results.

However, I should point out that if the desired low leakage cannot be maintained at say a 1 nm length scale then the answer is obvious: don't go down to this scale. Scaling up the device exponentially reduces tunneling losses but only polynomially increases size, cost, and energy. Therefore there will be an advantage to not going to the scale where leakage is totally dominant.

Cost-Efficiency Benefits



Next I wrote a simple numerical optimization program in C that optimizes this model in each year's technology based on the scaling assumptions. This chart shows number of bit-operations that each technology can perform per US dollar, taking into account both time-amortized manufacturing cost and energy cost at present electric utility rates.

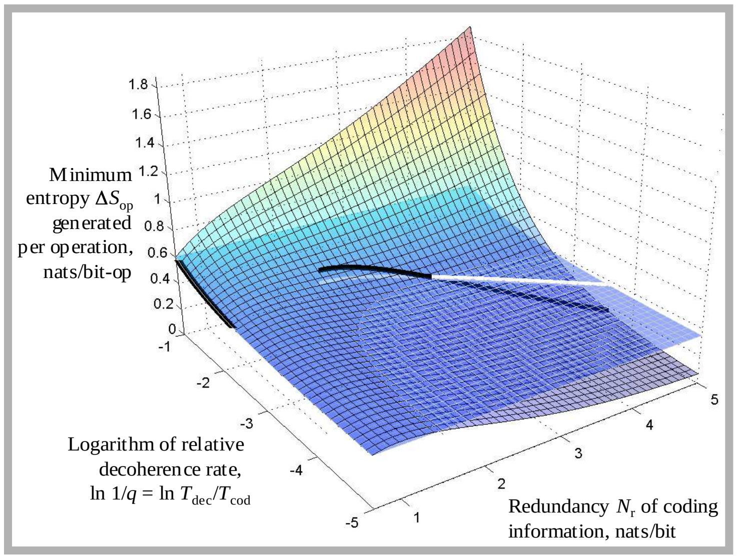
In the long run, energy concerns turn out to dominate the situation, but mostly through their affect on performance due to the cooling constraints, rather than because of the raw cost of energy itself. This reflects the fact that the total cost of the energy used by a 100-Watt computer operating continuously over its 3-year life is currently less than the cost of the computer itself.

The upper, blue line is the cost-efficiency of reversible computers on idealized problems for which the algorithmic overheads of reversibility are nil. The middle, green line is a more conservative model that assumes we find no better reversible algorithms for performing arbitrary computations than one that was discovered in 1989 by Bennett. Finally, the lower, red line shows the best that conventional irreversible computing can offer. Notice that its cost-efficiency hits the thermodynamic brick wall imposed by Landauer's principle by the year 2038, and cannot improve further. In constrast, reversible computing keeps improving. It starts to outperform irreversible computing between now and 2020, and becomes 1,000-100,000 x more cost-efficient by the 2050's.

After 2060, the cost-efficiencies of all technologies drop to 0 in this scenario because devices are so cheap that in order to spend as much as $1,000 on your computer (as the scenario requires), it has to contain so many devices that it dissipates more than 100 Watts due to leakage when it is powered up, even when it is sitting passively doing nothing at all! Obviously, in practice, the curves would not actually dip – either leakage rates would be further reduced, continuing the upward trend, or the pressure to further reduce device manufacturing cost would halt (due to the dominance of energy cost), and so cost-efficiency would stabilize at the peak level shown.

UNIVERSITY OF FLORIDA GATOR Engineering CISE University of Florida ECE Electrical and Computer Engineering

Minimizing Entropy Generation in Field-Effect Nano-devices



Earlier I mentioned the tradeoff of leakage with device size. In the following I attempt to quantify that tradeoff rigorously, in a simple model of field-effect devices that takes thermally activated leakage and physical information ("size") per digital bit into account. However it does not yet address tunneling and physical size, except indirectly insofar as this can be captured in the parameters that we do study.

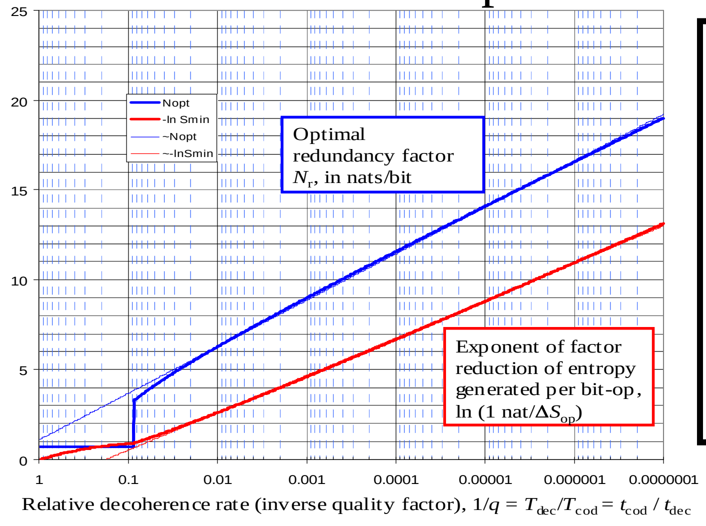
In this more recent model, the key independent parameter is a quantity that I call the "quantum quality factor" *q* (or Q) of a given device technology. *Q* is the ratio between the number of useful computational operations performed, and the number of entropy-generating quantum "decoherence" events. Equivalently, it is the ratio between the energy involved in carrying out a logical transition, and the energy lost to heat. Or, it is the ratio between the generalized "temperature" of the logic system itself, and the generalized temperature of the interaction between the logic system and its surroundings. It is a measure of the degree to which the computational system is insulated from its environment. It measures the extent to which the system can be treated as system that evolves ballistically, in a quantum-coherent fashion. The inverse of *q* is called the *relative decoherence rate*, and it is an important measure in quantum computing as well.

The axis at the lower left gives the natural logarithm of 1/*q*, that is, the quality increases exponentially as we move towards the lower right. The relative decoherence rate here ignores thermally-activated leakage, because this is treated separately in this analysis.

The axis at the lower right gives the number of natural-log units worth of physical information that is used to encode each bit of information. In field-effect devices, this controls the height of energy barriers as multiples of kT, which affects thermal leakage rates, as we mentioned earlier.

Finally, the vertical axis gives the total entropy generated per bit-operation in nats, including both the thermally activated leakage as well as the effect of *q*, when the speed of the adiabatic process is optimized so as to minimize the total entropy generation. When *q* is very small (less than about 11.6) the optimum (indicated by the black line) hugs the left edge of the surface, where only one physical bit (e.g. a spin) is used to represent each logical bit. But for higher *q*, there exists a local-optimal level of redundancy of the logic encoding, that depends on *q*. Let us look at this optimum behavior more closely.

Lower Limit to Entropy Generation Per Bit-Operation



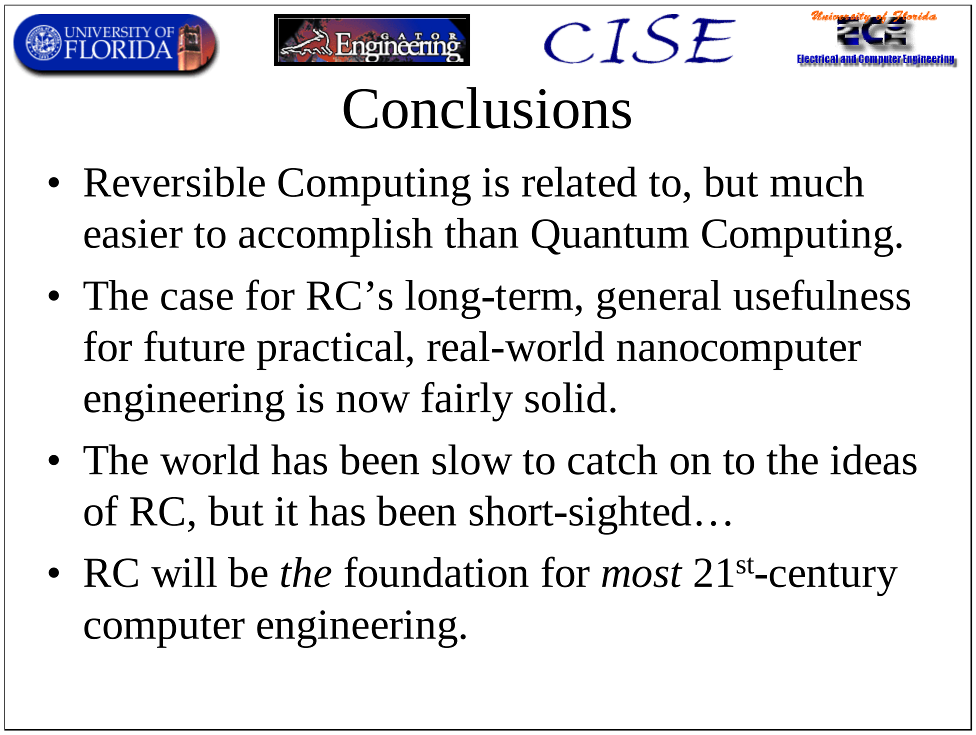
Scaling with device's quantum "quality" factor *q*.

- The optimal redundancy factor scales as: 1.1248(ln *q*)
- The minimum entropy generation scales as: $q^{-0.9039}$

In this graph, we plot 1/q along a logarithmic scale on the horizontal axis. The upper, blue line shows the optimal redundancy factor, which increases logarithmically with *q*. Note the discontinuity at (1/q = 1/11.6) where the local minimum on the surface becomes the global minimum. The lower, red line shows the logarithm of the factor by which the entropy generated per bit-op is reduced below a level of 1 nat. Note that it increases almost as quickly as log q itself. In other words, so long as we can improve the quantum quality of our devices *apart* from thermally activated leakage, we can reduce the total entropy per operation almost proportionally to this, even if we are still just using the field effect for switching.

The limitation of this new analysis is that this interesting aspect of device modeling has not yet been incorporated into the whole systems-engineering context presented earlier, of optimizing overall computational cost-efficiency. To do this, we first need to develop a more detailed technology scaling model, that tells us how *q* may interact with other device characteristics such as a device's size and its adiabatic entropy coefficient. This is made more difficult by the fact that *q* includes the losses in the resonant, oscillating circuit elements that are needed to power adiabatic circuits, which have not previously been well-modeled. We are currently making progress on this new modeling effort, but it has not yet been completed.

That concludes our discussion of our results to date.

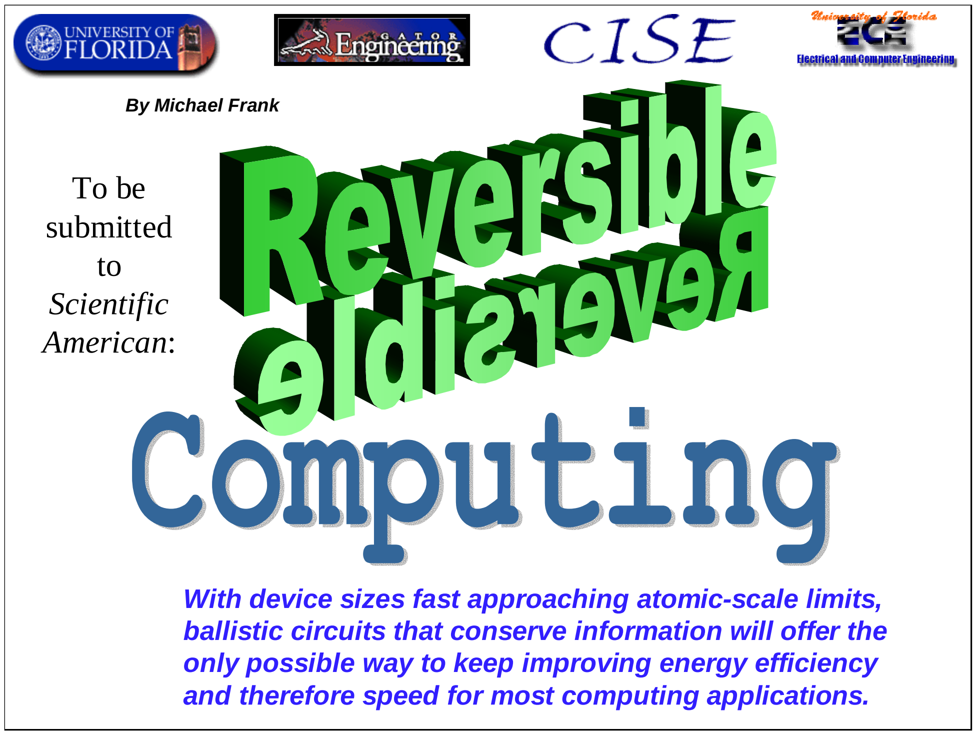


Conclusions

- Reversible Computing is related to, but much easier to accomplish than Quantum Computing.
- The case for RC's long-term, general usefulness for future practical, real-world nanocomputer engineering is now fairly solid.
- The world has been slow to catch on to the ideas of RC, but it has been short-sighted…
- RC will be *the* foundation for *most* 21st-century computer engineering.

This slide is self-explanatory. I want to emphasize that reversible computing looks like a better choice than quantum computing if we are interested in making the most practical impact on general-purpose computing in the long term, with the least effort.

In contrast, quantum computing can be expected to revolutionize cryptography and computational quantum physics, but not much else, and these fields only comprise a miniscule fraction of the present and anticipated future world market for computing. However, we cannot be sure that very generally applicable quantum algorithms that provide powerful speedups for more common practical algorithmic problems are not just around the corner. But, I suspect not, given the slow pace of progress on quantum algorithms to date, as well as a number of proofs of the impossibility of speeding up a variety of problems with quantum algorithms.



University of Florida

GATOR Engineering

CISE

University of Florida ECE Electrical and Computer Engineering

By Michael Frank

To be submitted to *Scientific American*:

Reversible Computing

With device sizes fast approaching atomic-scale limits, ballistic circuits that conserve information will offer the only possible way to keep improving energy efficiency and therefore speed for most computing applications.

Title splash from a working Scientific American article currently under development. I think reversible computing deserves wider exposure and the time is ripe to promote it more broadly.