Classical Reversible Logic Circuits for Quantum Computer Control

Zettaflops, LLC Technical Report ZF010

Erik P. DeBenedictis
Zettaflops, LLC
1415 Canyon Rim Dr., NE
Albuquerque, NM 87112, USA
erikdebenedictis@gmail.com

March 28, 2022

***Abstract*—This document presents an approach for scaling up quantum computers that makes use of novel principles for both the quantum and classical portions. For qubits requiring a cryogenic environment, the heat dissipation of the collocated classical portion is expected to limit scale up. This document adapts and applies classical reversible logic, which is more energy efficient than CMOS of equivalent function when operated in a cryogenic environment. While the advantages of reversible logic have been demonstrated in other contexts, this document makes novel adaptations for cryogenic operation at relevant speeds and describes a method of turning quantum algorithms into the cryogenic processor component of a quantum computer architecture. The simulation file with the key circuits has been released as open source, which can be used to generate plots that illustrate the key advances. These ideas can enable more capable quantum computers in addition to answering the question of whether reversible logic would enable more capable computers in general. The difference from past work being that the result is achieved for quantum computers where the original expectation had been microprocessors.**

Keywords— adiabatic computing; reversible computing; reversible logic; Adiabatic CMOS; PL/AL architecture; quantum computer; CMOS; cryo CMOS

I. Introduction

This document shows how to implement the cryogenic portion of a quantum computer's classical control system using an adaptation of reversible logic, potentially reducing dissipation in the cryostat and allowing greater quantum computer scale up.

Quantum and reversible computing were invented in the 1980s and 1990s as the first two discoveries in a new field called the physics of computation [1, 2]. After 35 years of physics research, quantum computers demonstrated quantum supremacy [3] and created a societal expectation that quantum computers would become practical.

Reversible computing, also called reversible logic, offers an energy efficiency improvement over CMOS due to energy recycling, as illustrated by the circuit simulations in Fig. 1. The curves show cumulative electrical energy flow in and out of chips of comparable behaviors implemented with (a) Complementary Metal Oxide Semiconductor (CMOS) and (b) reversible logic built from the same transistors. Since all energy entering a CMOS chip leaves as heat, the CMOS curve rises monotonically. However, the reversible circuit uses energy for a short time and then transfers most of it back to the power supply for recycling, dissipating only a second order portion in the cryostat.

Starting in the 1990s, several research groups designed and tested chips fabricated using CMOS fabrication processes but using reversible logic circuits [4, 5, 6, 7, 8, 9, 10]. The new circuits were applied to the microprocessor architecture, creating reversible microprocessors [5, 7, 11]. Progress slowed

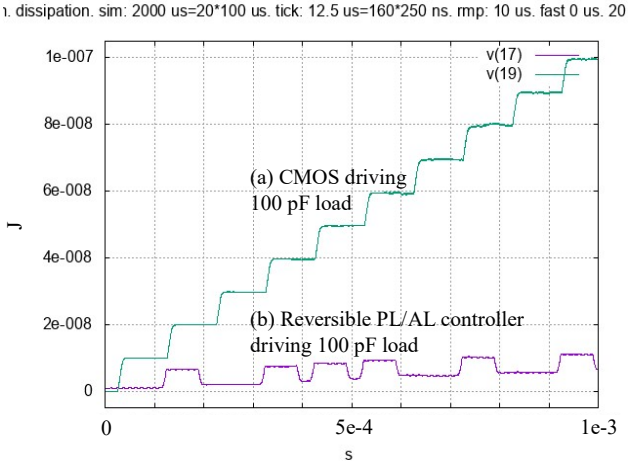


Fig. 1. Summary of the energy efficiency advance. (a) Cumulative energy drawn from a power supply for a CMOS circuit driving a 10 V square wave to a 100 pF capacitor. The curve rises by CV^2 each time the signal transitions from 0 to 1. (b) A reversible circuit driving the same voltage to the same size capacitor—plus a pattern generator representative of a quantum circuit (note the pattern causes pulses to be present or absent). Each transition from 0 to 1 draws $\frac{1}{2}CV^2$ (plus a small overhead) from the power supply, but the transition from 1 to 0 gives back $\frac{1}{2}CV^2$ (minus a small overhead). Curve (b) rises, but more slowly, representing higher energy efficiency.

when it became apparent that the resulting microprocessors would have to run slowly to achieve high energy efficiency and an essential component, the energy recycling power supply, would not be efficient enough without a technology breakthrough.

Quantum computers are distinct from classical computers, yet now well enough understood to be scaled up. Quantum computer scale up is limited in part by excessive dissipation in the cryostat, at least for some qubit types [12, Fig. 4e]. This poses the question of whether the energy efficiency approach in reversible logic could adapted to quantum computers to facilitate scale up. If reversible logic is to be a solution, it would have to avoid the obstacles that thwarted reversible microprocessors.

Qubit operations are slow, providing headroom on the speed issue. It has now become apparent that large-scale quantum computers will need to use quantum error correction [13]. Quantum error correction requires frequent quantum measurements, which are about 1,000× slower than CMOS gate operations and become the rate-limiting operation of a quantum computer. While slowing down a microprocessor clock by 1,000× cuts system throughput by the same factor, a classical control system running 1,000× slower (1 MHz) than a microprocessor (1 GHz) would naturally bridge the speed differential between quantum and classical operations and could be acceptable or even ideal.

While the elusive energy recycling power supply is essential for room temperature reversible logic, its role in the powertrain can be filled by a cryogenic refrigerator [14, 15], given that one already present for another reason.

A physical demonstration would be an appropriate initial step towards validating reversible logic for the role just discussed—yet these demonstrations occurred inadvertently in the 1990s [9]. While the reversible logic test chips mentioned above did not make a compelling argument for further development of a reversible microprocessor, application to quantum computers would require a reversible logic circuit in conjunction with a cryogenic refrigerator, not an energy recycling power supply. It turns out that the test chips were exactly what was needed to validate the quantum computer use case, even though the people involved did not realize it at the time.

The next step, which is the purpose of this document, is to devise a strategy for building a quantum computer that allows the energy efficiency of cryogenic reversible logic to carry through to the quantum computer as a whole and allow greater scale up.

This document will show a system design that includes a standard computer at room temperature plus a cryogenic processor based on reversible logic. These components would replace similarly named components in the PL/AL architecture [16], a known quantum architecture.

This document will show how to replace the function of the standard computer and the cryogenic processor using a new approach that puts the complex and otherwise energy consuming functions into the standard computer at room temperature so the remaining functions performed in the cryogenic processor can use known or readily demonstrable reversible logic circuits with low dissipation.

Using quantum error detection and correction as an example, this document shows how the ideas described above can be applied to any quantum algorithm. In summary, the quantum algorithm is first expressed as a flowchart. The boxes and diamonds of the flowchart are then replaced by reversible logic circuit schematics, with the arcs in the flowchart forming connections between the circuits. This yields a netlist that could be fed into a Computer Aided Design (CAD) tool and fabricated to create an integrated circuit.

The circuit simulated in Fig. 1b is actually the central portion of a classical control system of a quantum computer, so reducing dissipation in the cryostat could permit more qubits and hence scale up. Shor's quantum factoring algorithm [17] can factor a number in fewer steps than a conventional computer. This document builds on Shor's algorithm, or any quantum algorithm, by devising a classical reversible logic circuit, which is essentially an algorithm, for controlling the quantum computer with less dissipation than CMOS. Combining these two ideas would allow a quantum computer to run algorithms with both fewer steps and less dissipation.

At a higher level, this document extends Landauer's concept of a "minimum heat dissipation typically on the order of kT for each irreversible operation" [18] to a system with multiple regions at different temperatures. This document ties Landauer's physical concept to computer architecture by showing how to allocate calculations of different physical complexities, such as reversible and irreversible, to regions at different temperatures, and where those temperatures dictate different hardware options, in order to optimize the design.

To foreshadow the result, the cryogenic processor is a data decompressor, similar to image decompressors in web browsers. The data decompressor takes on a role similar to a cache in a classical computing system. Neither a cache nor a data decompresser "compute," but the roles of both are to alleviate an obstacle to scale up, specifically latency and heat. The phrase "reversible logic" also turns out to be a bit of an overstatement because the structure is almost entirely comprised of shift registers. Non-trivial reversible logic gates only appear in one place in this document, and it is just an optimization.

II. The Subcircuit Sequence Graph or Flowchart

The synthesis process will be illustrated for a simple algorithm, yet can be applied to any algorithm that expands into a quantum gate sequence.

Fig. 2 is an error detection and correction algorithm for the 5-bit error correcting code designated [[5, 1, 3]]. In rough equivalence to a 5-bit byte in a classical computer, a quantum computer supporting the [[5, 1, 3]] code would perform operations on groups of 5-7 qubits at once.

The top-level structure of the algorithm for checking a 5-bit codeword for an error and correcting it if found [13, p. 2 Error-correction procedure] is:

$$\text{if } (S_{XZZXI}) \text{ fix\_error} \quad (1)$$

else if (S_{IXZZX}) fix_error

else if (S_{XIXZZ}) fix_error

else if (S_{ZXIXZ}) fix_error,

where S_{XZZXI} performs verification or a syndrome check corresponding to the pattern XZZXI, returning true if there is error. The three other patterns like XZZXI are circular rotations of the symbols.

For completeness, other algorithms require various forms of a looping construct, which would have the textual representation:

while (M_{continue}) iteration, (2)

which repeats iteration while M_{continue} is true.

This section will show how to create the graph in Fig. 2a that represents all the qubit gate sequences that algorithm (1) could possibly generate. The graph will be translated into a hardware schematic later in this document.

Fig. 2a illustrates the range of primitive operations required for quantum error correction. The double-outlined block in the upper left represents the circuit in Fig. 2b that verifies the first of four check qubits, but does so in a way that also detects errors in the gates that perform the verification. This circuit has a typical sequence of state preparation, quantum gates, and quantum measurements.

The process then undergoes what a classical computer would call a conditional branch using the module in Fig. 2d. If qubit measurements (M) report no error, the process continues by verifying the second check pattern, and so on. If all four checks along the top row succeed, the error check completes having found no errors.

When one of the qubit measurements reveals an error, all four check qubits are verified by four single-outlined blocks shown in Fig. 2c.

The information from all the measurements together will yield up to three qubit corrections designated U_i, U_j, and U_k, where each U is one of the four single-qubit gate operations X, Y, Z, or I (I means no correction) and i, j, and k identify which qubit receives the correction.

The discussion above motivates what is called a subcircuit sequence graph in this document, which is an intermediate form in the synthesis process:

- The graph is based on, at minimum, a set of universal quantum gates plus qubit reset and measurement. Beyond the minimum, the set may include separate operations for each input of a two-qubit gate (such as "CNOT control" and "CNOT target") and composite operations (such as "Hadamard, one input of control-Z, Hadamard).
- Boxes of functional type, using the terminology in [19] containing quantum subcircuits of the above.
- Boxes of functional type containing parameterized single-qubit gate operations U_i where the operation type U and qubit identity i are determined by measurements. While not discussed further in this document, the approach is for the subcircuit graph in Fig. 2a, which was described as applying to 5-7 qubits at a time, to split into graphs for each qubit, yielding separate decisions for rotations by X and Z for each qubit.
- Diamonds of predictive type, using the terminology in [19], that control which of several operations is to be carried out next based on the results of measurements.
- A subcircuit graph comprises boxes and diamonds connected by arcs, which is commonly called a flowchart. The flowchart is equivalent to a textual representation containing if and while decision statements.

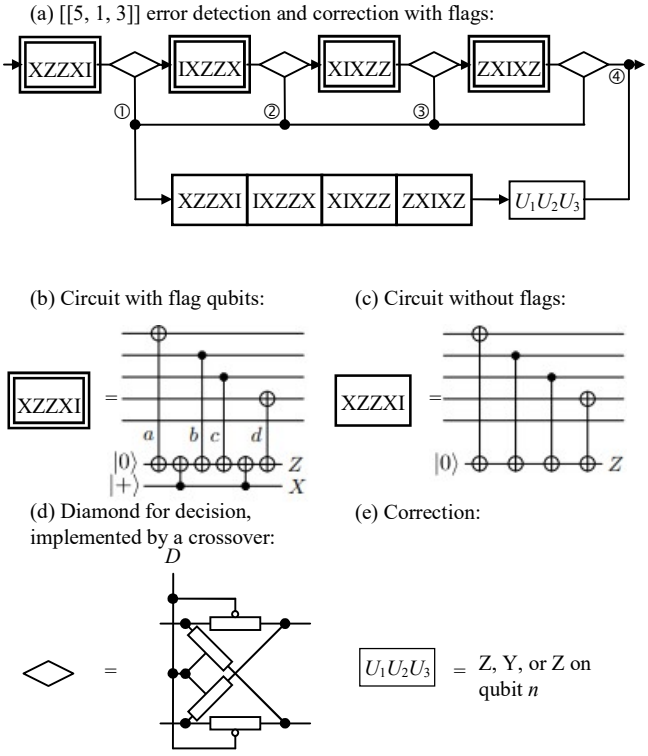


Fig. 2. Overall [[5, 1, 3]] correction algorithm is composed of (a) a subcircuit graph, (b) (c) syndrome extraction subcircuits, (d) diamond or crossover, and (e) correction module.

Ref. [19] demonstrates equivalence between a flowchart and a Turing machine, yet this equivalence requires a stack. The subcircuit sequence graph in Fig. 2 deliberately excludes the decisions that control the diamonds. This is so the room-temperature computer can provide the functionality of a stack, hence making the subcircuit sequence graph extremely general.

A. The Prime-line/address-line architecture

Fig. 3 illustrates the "Prime-Line/Address-Line architecture" or "PL/AL architecture" proposed for large-scale quantum computing [16, 20], where the cryogenic processor in Fig. 3a is the main topic of this document. The architecture is structured

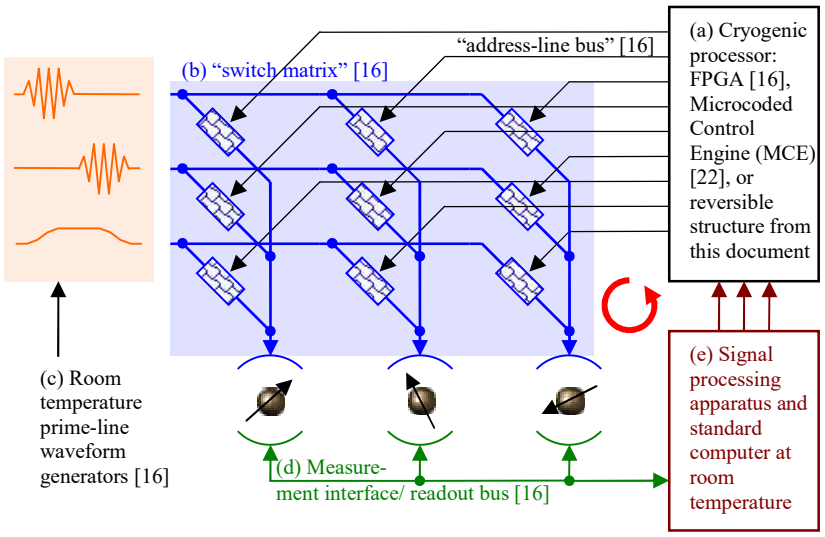


Fig. 3. "PL/AL architecture" [16] as redrawn for this document. (a) A cryogenic processor chooses which gate to apply to each qubit on each time step, the nature of this processor is the main topic of this document. (b) A switch matrix gates one prime line waveform to each qubit on each time step, where (c) the waveforms are generated at room temperature. (d) Measurement is performed by exposing qubits to waveforms generated at room temperature and routing the reflected signal back to room temperature. (e) At room temperature, signal processing apparatus processes the reflected signals, passing the digital measurement results to a standard room-temperature computer for management of high-level activities. As shown by the circular arrow, the overall information flow is counterclockwise.

like a player piano plus a semi-independent qubit readout capability.

The task of Fig. 3 is to expose each qubit to an analog waveform. The switch matrix in Fig. 3b fills the role of a piano keyboard for each qubit, combining analog prime-line waveforms in Fig. 3c, each equivalent to a piano note, into a time sequence defined by the cryogenic processor in Fig. 3a, equivalent to the moving paper roll controlling a player piano.

Just as integers are composed of prime factors, Fig. 3c shows three representative prime-line waveforms, collectively called a prime-line bus [16]. The analog waveforms define operations, such as X, H, and control-Z, which are able to specify all possible behaviors of the qubits if combined in an appropriate time sequence.

Currently, qubit readout involves complex signal processing that has only been demonstrated at room temperature, represented in Fig. 3d and e [21]. Similarly to [22], this document incorporates qubit readout through a subsystem that interfaces to room temperature equipment independently, returning information to the cryogenic processor in Fig. 3a.

In analogy to a jukebox, the standard computer in Fig. 3e observes the readout of qubit state and selects which roll of music to play at a given time. The standard computer also executes the classical portion of an overall quantum application.

The ideas in this document anticipate that designers will choose the prime-line waveforms in Fig. 3c in part to reduce dissipation in the cryostat. The prime-line waveforms should comprise a universal set of quantum gates, but there are many universal gate sets. Some quantum computer research groups consider many universal quantum gate sets and choose the one that has the best speed or accuracy for a given set of applications. This document goes a step further by allowing the designer to choose prime-line waveforms to reduce the speed of the cryogenic processor in Fig. 3a, thus reducing dissipation through the classical adiabatic principle.

III. Reversible Circuit Synthesis

This section describes reversible transistor circuits to the minimum level of detail required to understand the body of this document. Appendix A offers more detail.

Reversible transistor circuits are classically adiabatic, meaning a circuit's dissipation is proportional to clock rate – until the dissipation is so low that static dissipation dominates. The adiabatic circuits in this document use AC power-clocks with linear voltage ramps of duration τ, so adiabatic behavior is equivalent to the presence of a $1/\tau$ term in the circuit equations for dissipation.

Circuit simulations in Fig. 4 compare energy per operation of a CMOS shift register with a reversible logic shift register created with the same transistors.

The energy per operation of the CMOS circuit ① is $\frac{1}{2}CV^2$

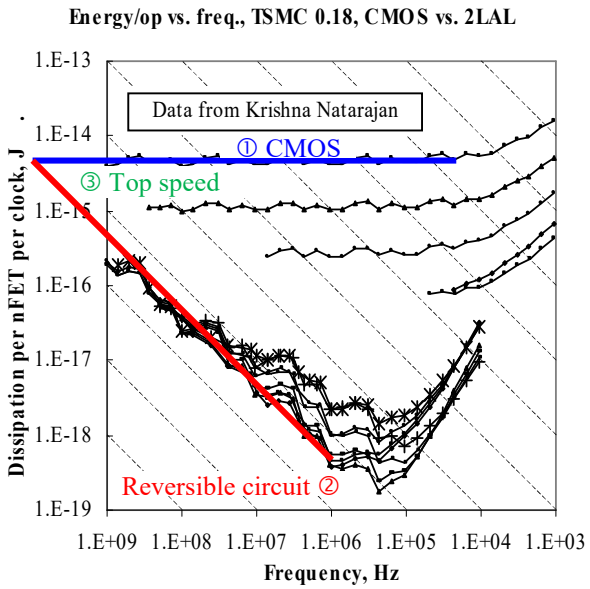


Fig. 4. Reversible transistor shift register. CMOS is ① is flat at $\frac{1}{2}CV^2$, reversible circuit ② has slope −1 due to $1/\tau$ adiabatic scaling. The lines intersect at ③, about the top speed of CMOS. Both CMOS and reversible circuits slope upwards at low frequencies due to leakage.

irrespective of clock rate.

However, the energy per operation of the reversible circuit ② includes a downward line of slope -1 on a log-log scale.

The straight lines above have slope 0 and −1, so they intersect. If the circuits have similar functions, the transistors are the same, and the supply voltages are the same, the two lines will intersect at about the top speed of CMOS. This implies that the energy advantage of the reversible circuit over CMOS is equal to the amount of slowdown from the intersection point ③, or about 1,000× for this application.

The discussion above supports the conclusions of this document because the cryogenic processor in this document is composed entirely of shift registers operating at about position ②. However, Appendix A provides additional detail on implementation, references to material on logic circuits, operating points other than position ②, and theory that applies to other circuits that work similarly.

A. Storing subcircuits in reversible shift registers

The first step in the synthesis procedure is to create a place to store the subcircuits. It was noted by another author addressing this task that "the QECC microcode memory can be designed as FIFO" [22], which translates to the terminology in this document as "the PL/AL architecture's quantum circuit memory can be implemented with a sequential-access memory."

A shift register is a sequential memory and is also the most basic reversible circuit in the literature, frequently used for defining logic families [4, 5, 6, 7, 8, 9] and as a test circuit for measuring [9] or simulating [10] dissipation.

This document takes the straightforward approach of storing each subcircuit definition in a circular shift register as shown in Fig. 5. Since the switch matrix in Fig. 3b has 9 switches, the shift registers holding the subcircuits in Fig. 2a will be 9 bits wide. Each circular shift register will have the exact length l needed to store the circuit specified in Fig. 2b or c, given the chosen prime-line encoding.

If a shift register has length l, clocking the shift register l times will cause its entire contents to shift out its right side in the correct sequence to drive the switch matrix. Since the shift register is circular with period l, the data will be returned to its original position and can be reused.

While reversible logic recycles energy, we need to consider whether or not the energy delivered to the switch matrix is also recycled. The creators of the PL/AL architecture used a matrix of High Electron Mobility Transistors (HEMTs) [16, Fig. 1] as switches. Each switch is controlled by a 300 mV signal in [16, Fig. 3], which is about the same as the operating voltage V used for the circuits in this document. However, the argument here applies to any voltage-controlled switch.

As explained in Appendix A, reversible transistor circuits recycle energy stored in the capacitance C of electrical nodes carrying signals. In this case, the HEMT's gate capacitance C_L is in parallel with the circuit's electrical node capacitance C, so the reversible circuit will naturally try to recycle the energy delivered to the switches.

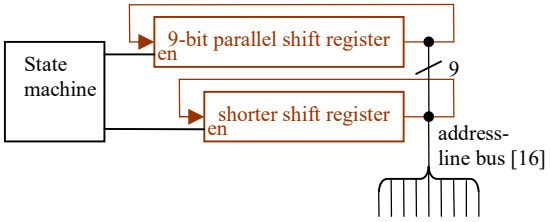


Fig. 5. Reversible PL/AL based on shift register storage. The two 9-bit wide shift registers can be imagined to hold the circuit descriptions for Fig. 1b and c. To execute Fig. 1b, the state machine enables the clock for a full rotation of the longer cyclic shift register. Likewise for the circuit in Fig. 1c and the shorter register.

If the switch presents a large capacitance C_L—such as a large HEMT or a remotely located switch with large interconnect capacitance—the load may cause I^2R power dissipation in the transistors creating the signal. The dissipation will be quadratic in $C+C_L$ and may be undesirable. The remedy is to increase the transistor widths on either side of the bus. In the simulation associated with this document, the transistors are 4× wider on the path between the power-clock and the address line bus the circuit generating Fig. 1b.

The design process described above is distinctly different from the one used in CMOS—and the difference is critical to energy efficiency. A CMOS designer confronted with Fig. 5 would typically "improve" the design by inserting buffers and format converters into the address line bus. This would defeat energy recycling if applied to reversible logic, but CMOS does not recycle energy anyway.

B. Clock enables

The reversible circuit in Fig. 5 looks like a tantalizing start of a reversible classical control system, but when both shift registers are considered, one discovers there are no reversible circuits in the literature with either clock enables or bus interfaces. Fully adiabatic static circuits have been devised for these purposes and are summarized below to the extent necessary to support the remainder of this document. The circuits are also described, with circuit diagrams, in Appendix B [14] and reference implementations appear in the ngspice simulator input in Appendix C.

The fully adiabatic clock enables in this document work differently than CMOS clock enables. CMOS clock enables leave the clock running but alter circuit's logic so the circuit does not do anything.

In contrast, a data-controlled clock for reversible logic halts at DC levels when not enabled and is identical to the main clock when enabled, as illustrated in Fig. 6a. The topmost trace is ϕ_0, the first phase of the main clock. The following 8 traces are data-enabled clocks (except for the bottommost green trace, which will be explained later).

A data-controlled clock in Q2LAL, the reversible logic family developed for the purposes of this document, is created by augmenting a one-bit or 8-stage shift register. A set of 8 data-controlled power-clocks are generated from four internal nodes of the shift register and four additional simple circuits connected to other internal nodes of the shift register. When a string of k 1 bits enter the shift register, all 8 power-clocks go

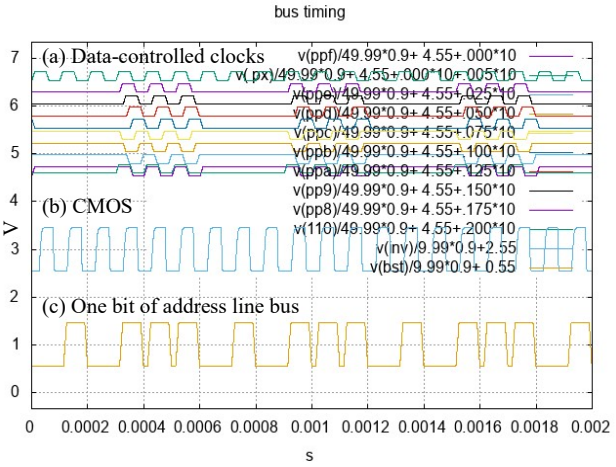


Fig. 6. Simulated traces. See text for explanation, but essentially from top: ϕ_0, $\pi_{0\text{-}6}$, and then the superimposed traces of π_7 and π_x. This is followed by the CMOS reference and on the bottom, 3+ repetitions of the pattern 010 111 being transmitted to the address line bus.

through k cycles and then halt at DC levels. Fig. 6a shows the waveforms for $k = 3$.

C. Fully adiabatic busses

A data-controlled clock can be enhanced to drive a bus by adding one additional clock and its inverse (two signals). The bus interface is created by rewiring one internal node in the shift register to use this new clock in lieu of one of the original 8 phases. The bottommost trace in Fig. 6a shows two traces that are the same when the clock is running, but the green bus control trace starts and stops one tick, or time τ, sooner than the other.

The bus's use in a system can be explained with reference to Fig. 5. A bus-interface shift register drives its output when the clock is enabled, going into a high-impedance state when the clock is disabled. The transition between driving and not driving is designed such that a series of shift registers can take turns driving the bus if exactly one shift register is enabled at all times.

The sub circuit powered by a data-controlled clock has all the benefits of energy recycling when the clock is running. When the clock is stopped, the sub circuit essentially turns into a static memory, holding state while dissipating only leakage power.

D. Advances

While shift registers are ubiquitous in the reversible logic literature, Fig. 5 makes advances over standard reversible logic.

- The reversible logic literature does not contain data-controlled clocks, meaning clocks run all the time. While the classical adiabatic principle can reduce the dissipation of logic, turning memories into low-dissipation logic still dissipates more than storing data in static circuits. The data-controlled clock provides a consistent way to implement both logic and memory with high energy efficiency.
- If the function Fig. 5 were to be implemented with the reversible logic in the literature, a multiplexer would be required to select the output of one of the registers to become the address-line bus—and then demultiplex the data to preserve reversibility. Each multiplexer would include a tree of AND-OR gates that would be vastly more complex than the bus interface circuit in Appendix B.
- The reversible logic in the literature recovers energy from its own operation, yet the shift registers in Fig. 5 recover energy from the output load, which is not described in the literature.

E. State machine and PL/AL architectural issues

The next step in the synthesis process is to create the fully reversible state machine in Fig 5. This should be impossible under the theory of reversible logic because the state transition diagram is irreversible, but this obstacle is overcome by placing irreversible functions on the standard computer in Fig. 3e.

Fig. 2a shows a quantum computer performing error correction and then forgetting that it did so, which is irreversible. Specifically, symbols ①-④ identify points where the state transition function maps multiple states to the same successor state.

Landauer wrote a seminal paper [18] on how a classical bit produces a minimum dissipation of the order of kT when it is "erased" or deleted. Taken at face value, Fig. 3 shows a continuous stream of bits flowing into the cryostat as it moves from Fig. 3e to a. While qubit measurement sends information out, there is no way to erase a bit by turning it into a qubit, leading to a buildup of bits in the cryostat that will have to be erased.

While kT is very small, the minimum dissipation for transistor circuit to erase a bit is about $\frac{1}{2}C_{min}V_t^2$, where C_{min} is minimum node capacitance to hold a signal and V_t is the transistor's threshold voltage [6, p. 91]—which is thousands of kT in today's technologies.

F. Externally controlled Fredkin gates

The theoretical discussion of how to avoid dissipation due to bit erasure is beyond the scope of this document, but the argument is summarized below.

We achieve the effect of sending data into the cryostat without dissipation through an externally controlled Fredkin gate, shown in Fig. 7a and b. For reference, a Fredkin gate has one control signal and two data signals. The data signals are swapped if the control is a 1, but the gate has no effect if the control is a 0.

The standard design in Fig. 7a shows data being sent into the cryostat as voltage $v(t)$ and back later as voltage $v'(t)$ to be erased by the external system. This is perfectly legitimate under reversible logic theory, but would require two sets of wires and the cryogenic electronics would need to drive the

long cable out of the cryostat. The alternative design in Fig. 7b applies a single voltage $v(t) = v'(t)$ to the cryogenic circuit. While more efficient, it does not follow the standard reversible logic electrical protocol and must be engineered from first principles.

The Fredkin gate in Fig. 7b is implemented as the crossover in Fig. 2d, where the control signal *D* is allowed to transition only at the reset point of the reversible protocol. This synchronization is straightforward because the standard computer Fig. 3e generates both the reversible clocks and the control signals.

The practical consequence is to route a 1 bit to one of two destinations, but the circuit does not add classical bits to the cryostat that must be erased with some minimum dissipation. While the system in the cryostat remains adiabatic, it is only reversible if the external system applies the time-reversed sequence of values to the Fredkin gate's control.

G. A one-hot state machine

For purposes of exposition, Fig. 7c shows two reversible shift registers. Depending on whether the diamond structure sends signals straight through or crosses them over, the overall structure will behave as either two independent reversible shift registers or a single long one. The crossovers could be the same as Fig. 2d as long as the signal *D* only makes transitions when the electrical protocol is in the reset state.

The state machine in Fig. 5 will be a shift register initialized with a single 1 bit. To explain the terminology, there is "one hot" bit, i. e. a 1, and the rest of the bits are 0. The position of the 1 bit defines the state.

H. The synthesis step

The logic synthesis process is based on shifting between two views of Fig. 2a. These views are the (1) subcircuit sequence graph, flowchart, or state machine description where the rectangles represent a state and (2) circuit diagram where rectangles represent shift register stages. The connection between these views is that when a shift register stage holds a 1, the state machine is in the corresponding state.

The rectangles and diamonds in Fig. 2a become the shift registers and crossovers in Fig. 7d, which is simply a reorientation of Fig. 7c to the form of an "if" statement in the error correction algorithm in Fig. 2a [13].

I. The function of the state machine

The function of the state machine in Fig. 3a and Fig. 5 is to enable shift registers containing quantum gate patterns by turning on their clock.

Say a gate pattern has *n* time steps, corresponding to an *n*-stage shift register.

If $n = 1$, the state machine's shift register could be used as the shift register portion of a data controlled clock, so the shift register built into the state machine would drive the subcircuit definition onto the switching matrix directly.

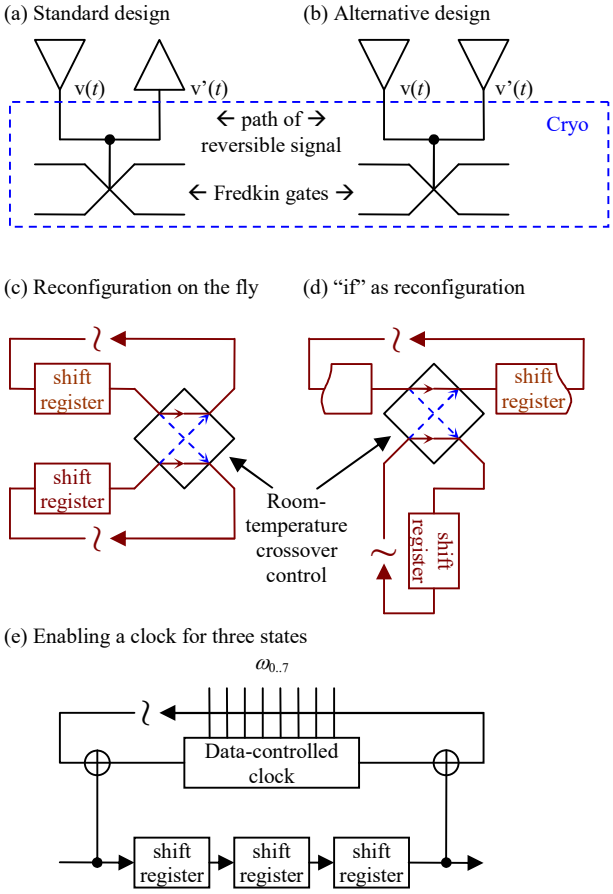


Fig. 7. It is legal to send data into a reversible logic region and later send it out. (b) However, an externally controlled Fredkin gate is equivalent to immediately sending the data back, requiring only one wire. (c) Depending on whether the diamond-shaped modules connect wires straight through or cross over, the circuit comprises either one or two cyclic shift registers. (d) An algorithmic "if" statement invoking states in the lower shift register conditioned on qubit measurement data coming from room temperature. (e) A module that turns on the clock when the system is in any of the three states in the bottom row.

The circuit in Fig. 7e could be used where $n > 1$. The *n* single-stage shift registers create a *n*-cycle delay. The two CNOT gates turn a data controlled clock on for *n* cycles.

In fact, a gate pattern of *n* time steps can be reduced to *n* gate patterns of one time step each, which avoids the need for CNOT gates.

This document has been discussing "reversible logic," but this CNOT gate is the only reference in this document to any reversible gate other than a non-inverting buffer—which is not normally considered logic.

J. Advances

The structure in Fig. 5 is recognizable as a data decompressor, which may represent an advance in a field that has repeatedly attempted to build a viable microprocessor.

In the terminology of data compression, symbols are loaded into the circular shift registers for repeated use. The output is

sequence of symbols, representing quantum subcircuits, specified by just the decisions in the flowchart. The decisions comprise a lot less information than, for example, storing the symbols in an array and specifying a sequence of array indices.

A GIF image in a Web browser or a Zip file on a computer can represent the output of an arbitrarily complex calculation even though the GIF or Zip decompressor is simpler than a microprocessor. This is obvious because the complex calculation occurs someplace else and it is just the answer that gets put in the GIF or Zip file.

What has been illustrated above is an architectural extension to Landauer's concept of minimum energy. Practical refrigeration systems are always below 100% Carnot efficiency and practical logic devices always operate above Landauer's minimum energy, so moving a calculation from one environment to another causes a change in the energy of the calculation and the cost to transmit data between environmentss—the latter including digital costs of compression and decompression, signal energy in data transmission, and heat leakage in cables.

The subsystem just presented translates information between environments that differ in they way they treat energy and information. This is a new issue because all computers had been of the same type prior to the advent of quantum computers.

This is analogous to a cache in a conventional computer that translates information between environments that differ in the way they address device count and latency.

K. *Adiabatic multiplexing*

Crossovers using the circuit in Fig. 2d would require one or two wires into the cryostat per signal, but a multiplexing scheme [23] with adiabatic dissipation levels (i. e. RC/τ) could reduce the number of wires quadratically.

Fig. 8a shows the baseline dissipation model for on-chip reversible transistor circuits. One side of a transistor is connected to a power-clock originating off chip. The transistor charges wiring capacitance C_W plus load capacitance C_L through a channel resistance R_{on} that is likely to be around 50 kΩ. Taking this as the baseline, the multiplexing circuit should have dissipation of the same order or lower.

The DRAM-like circuit in Fig. 6b [23] can send data from a room-temperature system to a cryogenic reversible subsystem. The DRAM-like circuit will be driven by r analog or digital row wires and c digital column wires coming from room temperature, one of each shown. Since these wires are driven externally, the only dissipation would be due to the on-chip wire resistance, which will be low compared to 50 kΩ and its dissipation can be neglected. The access transistor, equivalent to R_{on} when selected, could be located physically close to C_L so the wire capacitance downstream of the access transistor will be small. This will permit C_L to be charged with the same level of dissipation as an internal reversible logic signal.

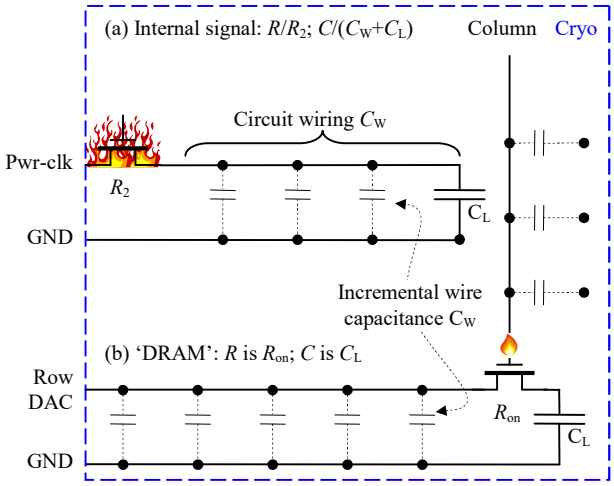


Fig. 8. DRAM-type external drive. (a) The baseline on-chip logic is cryogenic transistors (R_{on}) driving circuit wiring plus a load capacitance C_L. (b) For off-chip DRAM-type signals, the cryogenic transistors (R_{on}) are deliberately close to the load capacitor C_L; the $\frac{1}{2}CV_P^2$ energy on the long wires is dissipated at room temperature.

L. *Interface to standard computer*

A complete interface to the standard computer is beyond the scope of this document, but can be summarized with reference to Fig. 9.

Fig 9 shows the interface between the synthesized reversible logic and software on the standard computer. A state array and an AL-bus array in the memory of standard computer appear at the top. Below, the subcircuit graph in Fig. 2a has been linearized to show vertical alignment with entries in the arrays and the state machine's initial state is indicated by a star.

The functional connection starts with software maintaining the data in the arrays so they are a shadow copy of the state of the reversible logic. If the state machine in Fig. 5 has n states, state array will be of length n, where each entry represents a bit of the one-hot state register. Each bit could be represented as a 0, 1, or X, where X represents the unpredictable value that appears at power up.

The AL-bus array will also have length n, representing the AL-bus output when the state machine is the state with the same index, such as at points C and D. Each entry in the AL-bus array could be an X if the register contents are uninitialized. The subcircuit definitions could be stored in the AL-bus array as well.

Since the standard computer generates both the power-clocks and decisions, the two arrays can be updated to maintain a shadow copy of the state on the hardware in the cryostat.

While details are not included in this document, an algorithm would be created that resets the internal state after power-up through use of uninitialized X state symbols, shortest path algorithms to navigate the graph, and a limited set of multiplexed voltage drivers that set shift register contents like a SRAM cells.

Power-up initialization completes when the one-hot state register is correctly set to the starred location in Fig. 9 and all

the shift register memory has been initialized to the proper subcircuits.

With one exception, this document has been organized such that the clock could be turned on at this point and the quantum algorithm executed by via guidance from the standard computer at decision points.

The exception is that the DRAM external drive could be called upon to simultaneously drive decision values that interfere due to a use of the same row or column or different data values. This situation can be alleviated by judicious allocation of decision values to positions in the crossbar. If conflicts still exist, the crossbar can set decision values early through use of a scheduling algorithm on the standard computer.

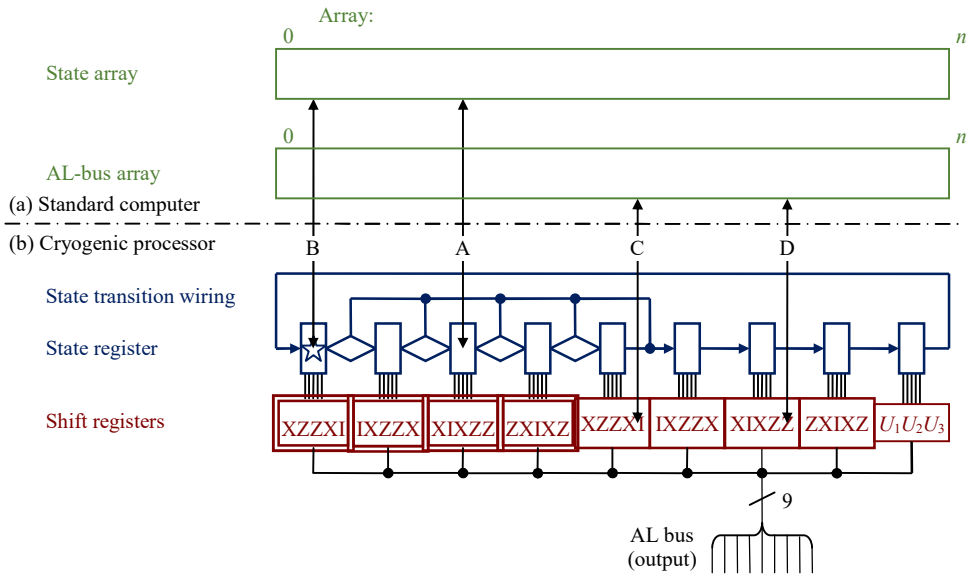


Fig. 9. Functional interface between the (a) standard computer and (b) cryogenic processor. Software in the standard computer includes two arrays that shadow the one-hot state machine and the subcircuit storage registers. The reversible hardware located in two rows, the top comprising the one-hot state register and the bottom comprising the subcircuit storage registers. A state transition region includes wiring that implements the arcs in the subcircuit graph. The state machine's initial state is indicated by a star. The AL is the output of the system.

IV. Results

The circuits in Fig. 5 have been simulated with Spice (ngspice) using input available as open source (details later), with the simulation output in Fig. 1 and Fig. 6.

A. Functionality

The trace in Fig. 6b is the comparison waveform generated by a CMOS inverter driving a 100 pF capacitor. The power consumption of the CMOS inverter creates Fig. 1a.

The trace in Fig. 6c is one line of the address line bus, which drives the 100 pF capacitor. The circuit being simulated has two shift registers containing patterns 010 and 111. They are transmitted in an alternating sequence, so the bottom trace outputs to the address line bus three and a third repetitions of each pattern: 010 111 010 111 010 111 01 (with spaces inserted for visual convenience).

B. Scaling

The hypothesis put forth in the discussion related to Fig. 1 is that CMOS circuits would have constant energy per operation while reversible circuits would scale as $O(1/\tau)$.

The curves at Fig. 4 point ② should have a slope of −1 to the extent the hypothesis is correct. There are many curves near point ②, representing simulations with, for example, different substrate bias, and some have slope pretty close to −1. As far as the author knows, the irregularity near point ② is due to the transistor operating points being outside normal CMOS operating conditions where Spice and the transistor models are not well tested. Debugging Spice at unusual operating points is currently designated as future work.

Transistors can be described by equations and the equations analyzed for asymptotic dependence as though they were operation counts in an algorithm. This makes the hypothesis equivalent to saying that the transistor equations for CMOS and reversible circuits should have energy per operation O(1) and $O(1/\tau)$.

Combining algorithms in computational complexity theory is equivalent to combining circuits. Since the entire controller in Fig. 5 is comprised of subcircuits where energy per operation scales as $O(1/\tau)$, the behavior of the whole circuit should scale as $O(1/\tau)$.

Circuit simulation should reveal the asymptotic behavior of the circuit equations. The difference in slopes between Fig. 1a and b is about 15×, which is non-trivial although less than expected. This is in part due to the curve in Fig 1b including a simple controller whereas the curve in Fig. 1a is just an inverter.

The open source ngspice simulator file includes that compares a more sophisticated reversible logic quantum computer controller with a CMOS work-alike, yielding a 131× dissipation reduction. Appendix C has more detail.

V. Conclusions

Reversible microprocessors demonstrated a reduction in dissipation equal to the reduction in clock rate, but the reduction in throughput detracted from the increased energy efficiency and the overall approach was thwarted by the lack of an adequate energy recycling power supply.

This document showed how to transform a quantum algorithm into a reversible transistor circuit for driving qubits. The reduction in the transistors' clock rate is about the same as the speed difference between CMOS gates and qubit measurements, so it is benign, and the elusive energy-recycling power supply is not necessary due to cryogenic operation.

Thus, this document show how to apply reversible logic to an important application without further breakthroughs.

The open source Spice (ngspice) simulator input described in Appendix C rigorously defines the circuits and shows a 131× dissipation improvement at 1 MHz and $1/\tau$ dissipation.

Thus, disaggregating the concept of "CMOS" into transistors and circuits, and switching to reversible circuits may reduce dissipation in the cryostat substantially. This will cut the user's power bill and also reduce congestion in the cryostat, perhaps leading to larger quantum computers.

The generalization of the concept above is a computing system with multiple subsystems at different temperatures, built from technologies with different computational properties, and connected by refrigerators with sub-Carnot efficiencies. To implement this type of computing system optimally, the designer would allocate computations and memory to different subsystems to minimize dissipation per unit of throughput. This document provides an example.

This document showed that the structure in Fig. 3a, called a cryogenic processor [16] due to its location, is performing new information heat management function unique to quantum computers. The structure is in an environment where heat generation should be minimized and this document showed that it can be implemented with essentially no logic gates. The structure in this document is similar to a data compression system, yet it could be given an name related to its function – similarly to the way classical computing community named its memory interface a "cache."

Appendix A Cryogenic Reversible Logic Using Transistors

This section compares CMOS, cryo CMOS, and cryogenic reversible logic all based on the same transistors.

Cryogenic reversible logic has been considered previously [5, p. 93], but no refrigerated computing technology, reversible or not, has successfully competed with room temperature CMOS.

However, qubits requiring cryogenic operation present a different decision tree. A quantum computer requiring cryogenic qubits will get its computational power from quantum speedup, not the classical control electronics, and the classical control system will need to accommodate. This creates a competition between cryogenic classical technologies. Cryo CMOS and classical Josephson junction logic are the incumbents, yet this document considers only cryo CMOS.

A. Capacitor charging

Most computing technologies use voltage-based signaling where dissipation is dominated by the energy required to charge the capacitance of the signal nodes.

Fig. 10 compares the dissipation of two circuits charging an node in a multi-temperature system, extending the result to the wall-plug energy of the logic families using the circuits.

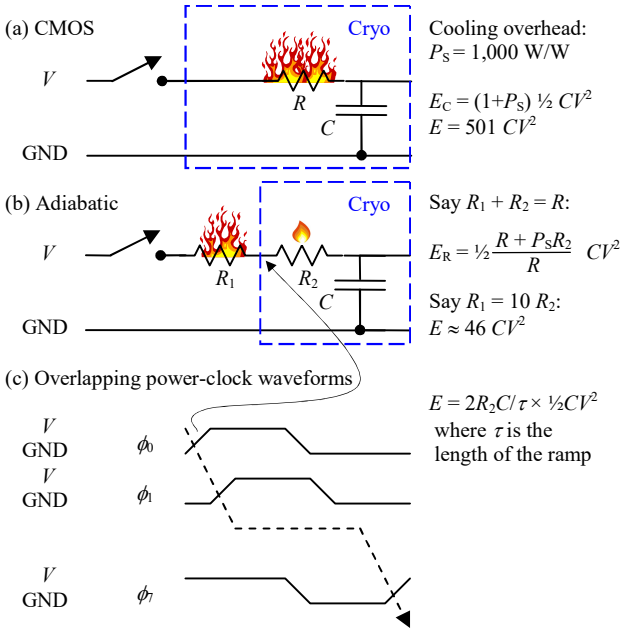


Fig. 10. (a) Charging a capacitor from a fixed voltage dissipates ½CV^2. If the heat has to be removed from 4 K with P_S = 1,000× overhead (1,000 W/W), the total energy from the wall plug will be 501CV^2. (b) If we move R_1 to room temperature, only the portion $R_2/(R_1+R_2)$ of the heat will flow through the refrigerator and incur the 1,000× overhead. If R_1 = 10 R_2, for example, this reduces overall power consumption and heat generated from 501 CV^2 to about 46 CV^2. Resistor R_1 could be the output transistor of a waveform generator, where it would vary with time (see text). (c) To establish context, reversible circuit families effectively vary R_1 so the voltage entering the chip is a linear ramp.

As illustrated in Fig. 10a, CMOS dissipates E = ½CV^2 every time a voltage node switches, where C is the wire or node capacitance and V is the supply voltage.

For cryo CMOS, the dissipation in the cryostat, indicated by the flame in the figure, must be multiplied by the refrigerator's specific power P_S and added to the dissipation. Specific power is the number of watts of wall-plug energy required to remove one watt from the chip. A heat sink has P_S = 0 and a refrigerator cooling to 4 K has $P_S \approx$ 1,000.

Fig. 10b shows resistance R being divided into two series resistances R_1 and R_2, $R_1 + R_2 = R$, where R_1 is outside the cryostat. The total energy drawn from the power supply must be the same because two resistors in series is just another resistor, but only R_2 is in the cryostat and contributes to cooling overhead.

The right side of Fig. 10 tallies the wall-plug energy consumption. Note that room temperature CMOS would win if it could compete.

B. Ramped power-clocks leading to a logic family

While the circuit in Fig. 10b is powered from the fixed voltage on the left, voltage enters the cryostat at the center point of the divider formed by R_1 and R_2. For a given charging time τ, the smallest dissipation in R_2 occurs when the capacitor is charged at a constant current [5]. This requires R_1 to be a variable

resistance that creates a linear ramp at the center point between R_1 and R_2.

If the voltage entering the cryostat is a fixed waveform, the voltage can be generated once and connected in parallel to many instances of R_2 and C. Such a waveform is called an AC power-clock, illustrated in Fig. 10c.

Power-clocks for reversible logic families have upward and downward sloping ramps as indicated in Fig. 10c. These logic families typically have 4-8 clocks that contain flat tops and bottoms. The multiple overlapping clocks allow signals to be processed by electrical protocols, such as reset-charge-compute-reset, ultimately yielding a family of reversible gates.

C. *Quantifying the dissipation reduction*

The in-cryostat dissipation of the circuit in Fig. 10b driven by a ramp is $E = 2R_2C/\tau \times \frac{1}{2}CV^2$, for large τ, where τ is the length of the ramp [5]. The reader will notice $\frac{1}{2}CV^2$ is common to the dissipation expressions for both CMOS and the adiabatic circuit, but the adiabatic circuit has the additional factor $2R_2C/\tau$, which can be considered an energy factor and is the reason for the $1/\tau$ dependence of energy on clock period.

Note that this analysis only yields an approximate comparison because CMOS and reversible circuits are different. The reversible circuit is usually more complex, resulting in more gates and more dissipation than is predicted by this analysis. Irrespective of the number of gates, wiring capacitance depends on layout details, causing C to vary from one circuit to another.

Fig. 1 plots the cumulative energy delivered to each circuit in Fig. 10, where the second circuit is driven by a power-clock similar to Fig. 10c.

D. *Adiabatic powertrain*

Sending power-clock waveforms from room temperature into the cryogenic environment without little distortion, noise, or heat leakage in the transmission lines requires a structure called a cryo-adiabatic powertrain [14].

E. *Historical context of cryogenic reversible logic*

The programmatic impact of the cryo cooler can be explained with the help of Fig. 11.

Of the external energy entering a reversible circuit, a portion $2R_2C/\tau = (1 - G_L)$ is turned into heat and the remaining portion G_L passes through and is available for recycling. G_L is a sub unity power gain that could be 99.9%. The energy recycling power supply reorganizes energy into a the properly shaped waveform, but turns a portion $(1 - G_P)$ into heat and passes portion G_P to the circuit to augment the external energy. Thus, the external energy is recycled in amounts G_LG_P, $G_L^2G_P^2$, ..., amplifying the external energy by the factor $1/(1 - G_LG_P)$. Let us say the objective is to deliver $\frac{1}{2}CV^2$ to the circuit. This will require external energy in the amount of

$$E_1 = (1 - G_LG_P)\ \tfrac{1}{2}CV^2.$$

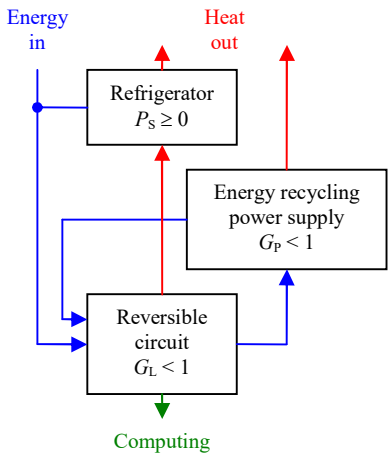


Fig. 11. Power flow for reversible circuits. P_S is the specific power of the refrigerator, $P_S = 0$ for a no refrigeration. G_L and G_P are the sub-unity power gain of the logic and energy recycling power supply, with guideline values of 99.9% and 95%.

However, the portion $(1 - G_L)$ of the $\frac{1}{2}CV^2$ delivered to the chip will be dissipated with overhead P_S. This leads to additional external cooling power

$$E_2 = (1 - G_L)\ \tfrac{1}{2}CV^2\ P_S.$$

Thus, the total external power is $E_R = E_1 + E_2$ for the reversible circuit and $E_C = (1 + P_S)\ \frac{1}{2}CV^2$ for CMOS.

This leads to a lower-is-better figure of merit

$$E_R/E_C = 1 - G_L(G_P + P_S)/(1 + P_S).$$

The equation above is remarkable because varying P_S transforms it into recognizable and important forms.

The original conceptualization of reversible logic had just one temperature, implying a passive cooler with a $P_S = 0$. In this case the previous equation becomes

$$E_R/E_C = 1 - G_LG_P \text{ at room temperature.}$$

Physical demonstrations in the 1990s measured G_L values up to 99.9%, but G_P lagged so the research projects concluded that higher efficiency energy recycling power supplies were a long-term research direction.

Operating the circuit at progressively lower cryogenic temperature is equivalent to raising P_S. In the limit as $P_S \rightarrow \infty$,

$$E_R/E_C = 1 - G_L \text{ at a cryogenic temperature,}$$

essentially a high P_S makes the problematic term G_P term disappear.

The physical demonstrations in the 1990s successfully demonstrated a G_L values as high as 99.9%, albeit at room temperature. However, we now know that cooling CMOS has little effect on this type of circuit. Thus, the physical demonstrations of reversible logic in the 1990s yielded high enough values the G_L parameter to validate the quantum use case, even though the experimenters did not know it at the time.

APPENDIX B Q2LAL AND ENHANCEMENTS

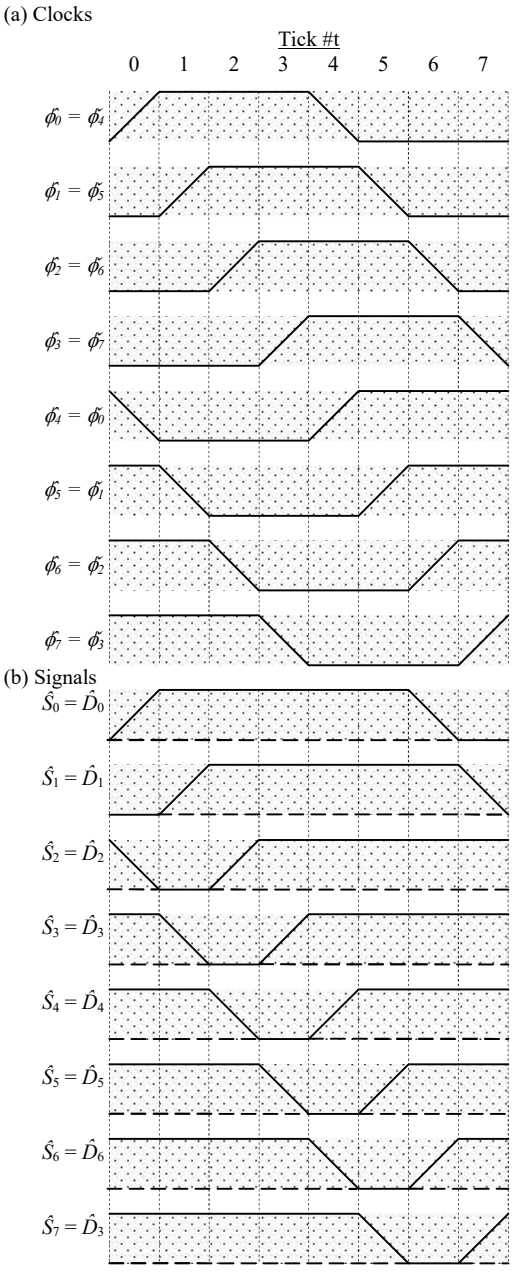


Fig. 12. (a) Power-clocks and (b) data signaling formats for a representative 8-phase circuit. Each positive-going pulse, $\hat{\phi}_i$ is equivalent to a negative-going pulse $\check{\phi}_{i+4 \bmod 8}$.

This appendix explains data-controlled clocks and circuit enhancements for support of busses, both of which are necessary to the cryogenic processor in Fig. 5. These enhancements are explained in the context of the Quiet 2-Level Adiabatic Logic (Q2LAL) family, developed by the author for more stable operation in a cryogenic environment [14].

Fig. 12 shows Q2LAL's 8-phase waveforms. The power-clocks are divided into ticks of duration τ, the ramp time. The power-clocks can be labeled $\hat{\phi}_{0\text{-}7}$, with the circumflex (hat) accent indicating that the power-clock is a positive-going pulse. However, flipping a power-clock upside down yields the same waveform as the pulse four ticks ahead or behind. Thus, the power-clocks have the property that $\hat{\phi}_i = \check{\phi}_{i+4}$ mod 8, with the caron (cup) accent indicating the waveform is a negative-going pulse.

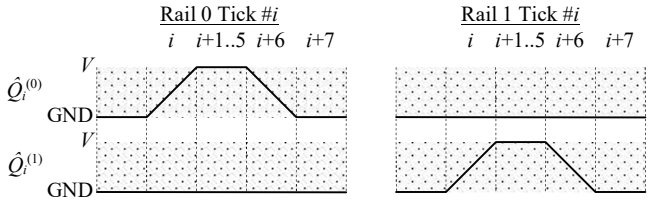


Fig. 13. Dual-rail signal waveforms. The first rail pulses for 1 and is a flat line for 0; the second rail is the opposite. Arithmetic on i is mod 8.

Data signals follow a reversible electrical protocol. The signal starts at a resting or reset state of 0 V for one tick of duration τ. If the signal is a 1, it rises in one tick of duration τ, stays at V for five ticks and then ramps back to GND in the sixth tick.

Fig. 13 illustrates dual-rail signaling. Each data connection driven by $\hat{\phi}_i$ constitutes two wires designated $\hat{Q}_i^{(0)}$ and $\hat{Q}_i^{(1)}$. The base wire $\hat{Q}_i^{(0)}$ has a pulse to V for a 1 as shown and a DC value of GND for a 0. The second rail is the opposite with a DC value of GND for a 1 and a pulse to V for a 0. Alternatively stated, the first rail carries signal $\hat{Q}$ and the second rail carries $-\hat{Q}$.

Fig. 14a illustrates the basic circuit building blocks. The transmission gate symbol represents a pFET and an nFET connected as shown and driven by electrically complementary signals $\hat{S}$ and $\check{S}$. The dual-rail transmission gate uses the same symbol but 2-conductor busses and a replication of the circuit.

The circuit framework is illustrated in Fig. 14b as a sequence of cycles comprising triangular adiabatic amplifiers and transmission gates. The relative phase numbers around a loop S_1,

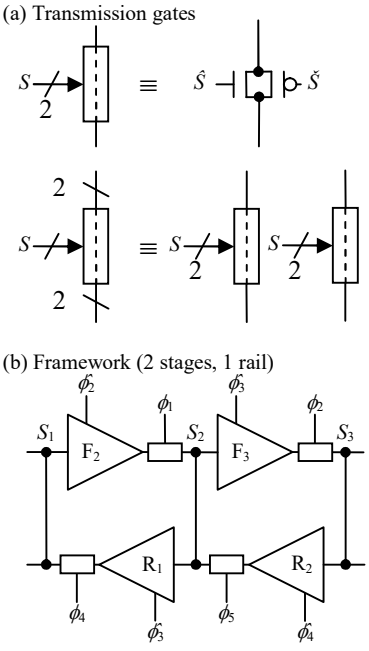


Fig. 14. (a) Transmission gate notation, notably including multi-rail. (b) Framework for multiple reversible logic families. F is forward and R is reverse computation.

ϕ'_2, ϕ_1, S_2, ϕ'_3, and ϕ_4 is always the same, yet subsequent loops repeat the pattern with the indices incrementing each time, mod 8. The F (forward) and R (reverse) functions, such as F_2 and R_2, are can be used to implement reversible gates.

Fig. 15a details the two rails of the adiabatic amplifier, which contains circuits for each of the two rails. Each of the two rails for phase i, $\hat{Q}_i$, is controlled by data signals from the previous phase $\pm A_{i-1}$ and a signal $\check{c}_{i-1}$.

Clamp signal $\check{c}_i$ in Fig. 15b is generated by two transmission gates. The inputs to these transmission gates are just clock phases, making $\check{c}_i$ independent of data. So, $\check{c}_i$ can be generated once and used for more than one gate.

The explanation above is specific to the Q2LAL logic family, yet the subsequent discussion applies to other reversible logic families as well, in part because it is implemented by novel clocking, not the circuit.

F. Data-controlled clocks and bus interfaces

Clocks $\omega\hat{}_{0\text{-}7}$ in Fig. 16b are solid lines in the shaded center of the diagram to show the clocks when running, but dashed lines on the left and right represent the clock levels when stopped. Building on the data-controlled clock waveforms in [14], this article introduces an additional clock $\omega\hat{}_x$ and its electrical inverse $\omega\check{}_x$, which starts and stops one tick or time τ earlier than the others, as shown by the bent bottom of the shaded region.

Fig. 16a is the circuit for an 8-phase section of a Q2LAL shift register. The subsequent discussion around Fig. 16 applies more generally because other fully reversible circuit families differ only by the clock waveforms and the signaling convention for data.

G. The memory cell

The center of Fig. 16b shows the clocks running for one 8-tick clock cycle, but the clock is stopped in the outer regions of the graph.

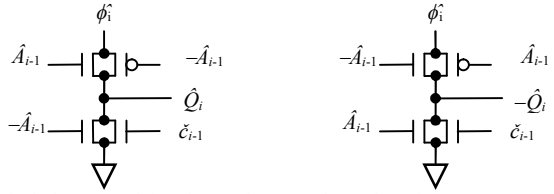
The red circular arrow in Fig. 16a identifies an amplified conductive cycle when the clock is stopped, creating a memory cell. At this point, $\omega\hat{}_{0\text{-}3}$ are low and $\omega\hat{}_{4\text{-}7}$ are high. The reader will see that the clocks around the cycle, $\omega\hat{}_5$, ω_4, $\omega\hat{}_6$, and ω_7, all have high values. Signals $\omega\hat{}_5$ and $\omega\hat{}_6$ supply power to two adiabatic amplifiers and ω_4 and ω_7 cause two transmission gates to be turned on.

H. The bus interface

The red circle with a cross through the middle, $\omega\hat{}_1$, ω_0, $\omega\hat{}_2$, and ω_3 has all low clocks indicating that there is no memory cell. However, $\omega\hat{}_7$ enables a transmission gate that drives A_0. A_0 is actively driven low when the clock is stopped because the drive voltage is determined by $\omega\hat{}_0$, which is low at that point.

As background information, a reversible shift registers of most any logic family cause the memory cell to migrate to the right as the clock phases proceed, see [7, 18].

(a) Q2LAL: Buffer stage i, both rails



(b) helper signal for clamp; does not depend on data

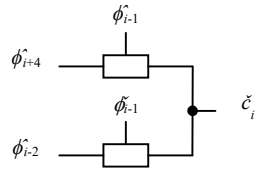


Fig. 15. (a) Adiabatic amplifiers for both rails, each based on data signals A_{i-1} from the previous stage. (b) The helper signal can be generated once in an entire circuit from available clocks.

I. Special clock ω_x for busses

However, the power-clock ω_x in Fig. 16b allows the shift register to drive a reversible bus. Signal ω_x is dual-rail comprising $\omega\hat{}_x$ and its electrical inverse $\omega\check{}_x$ (not shown). The ω_x waveform is identical to ω_7 when the clock is running but it turns on and off one tick earlier.

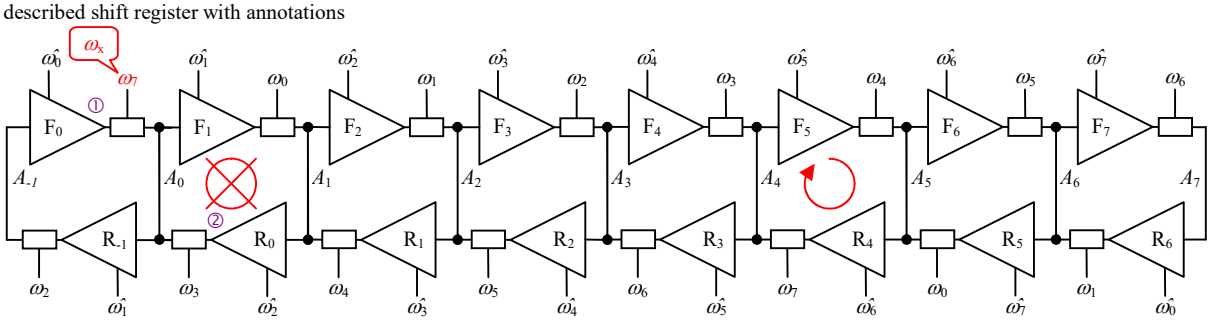
Thus, replacing ω_7 with ω_x at the location illustrated in Fig. 16a will change the circuit's behavior only when the clock is stopped. While ω_7 turns on the transmission gate that drives A_0 low when the clock is stopped, ω_x will cause the transmission gate to be an open circuit and A_0 will not be driven.

Since the reversible logic literature assumes all clocks run all the time, Fig. 16a is just a shift register in accordance with reversible logic theory prior to this article, even when ω_7 is switched to ω_x.

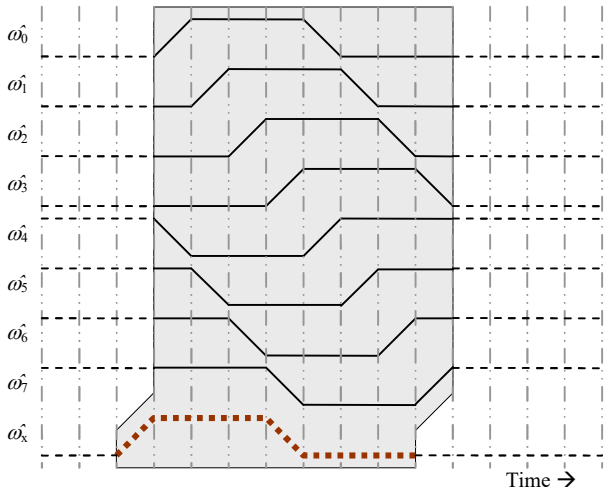
However, signal ω_x will put A_0 into a floating state when the clock is stopped, allowing us to extend reversible logic theory with a circuit that drives a bus only when the clock is running.

A CMOS tri-state [24] bus interface can drive a bus to the 0 and 1 states, but also has a third state that does not drive the bus at all. A CMOS bus also has a design constraint that exactly one of the interfaces drives the bus at a time—except during a handoff period where one circuit stops driving and another starts. The handoff must be designed to avoid electrical conflicts, such as short circuits, that could result if two circuits attempted to drive the bus to different values at the same time.

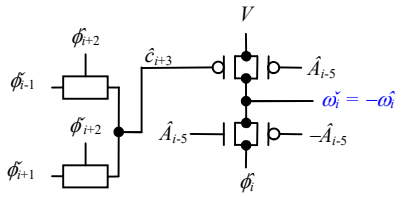
(a) Previously described shift register with annotations



(b) Data-enabled clock waveforms, with ω_x; waveforms range GND to V



(c) Special circuit that clamps to V



Power notes:
①②: 2 transistors in the adiabatic amplifier plus the pass gate must be wide
③④: 2 transistors in the adiabatic amplifier must be wide

(d) Data in shift register enables clocks

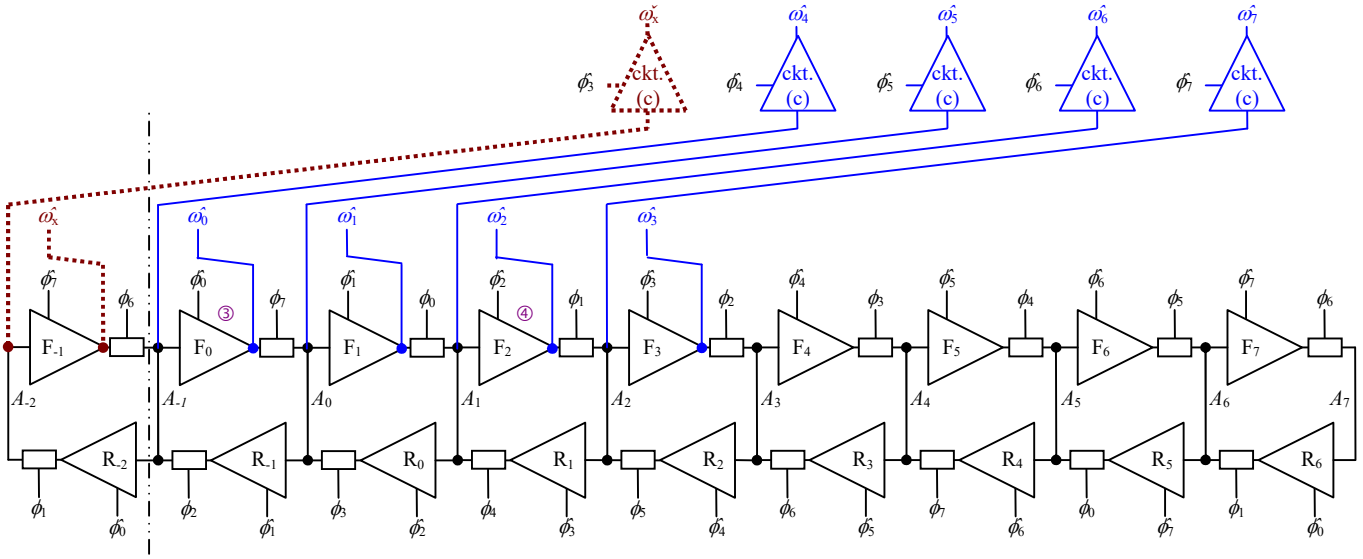


Fig. 16. Data-controlled clock. (a) Q2LAL is static, meaning the clock can be stopped at any time. The objective in this circuit is to start and stop the clock at the beginning of the first phase or tick. However, this means the first four clocks will rest at GND and the second four at V. (b) Clocks $\omega_{0\text{-}7}$ plus special clocks ω_x and its electrical inverse $\omega_{\breve{x}}$ for supporting busses. The first four data-controlled clocks $\omega_{0\text{-}3}$ are just taps of existing signals. However, the others $\omega_{4\text{-}7}$ require the special circuit. (d) Augmented circuit shows ω_x and $\omega_{\breve{x}}$ need data signals from beyond the left edge of the circuit, suggesting a renumbering of phases. However, the diagram highlights the fact that data is used from only five of the eight phases, making three phases available where the data may be altered by logic in time for another, back-to-back, data-controlled clock (such as in a state machine).

First, let us consider why turning the clock on and off at different times violates the design rules in existing reversible logic theory [7, 14]. Leaving the clock off for a long period of time would let A_0 float and device leakage could sometimes cause drift to a significant voltage. When the clock is subsequently turned on, the sudden discharge of this voltage could cause a current transient that could disrupt the circuit.

However, two copies of the circuit in Fig. 16a can create a bus if one interface is driven by the clocks $\hat{\omega}_{0\text{-}7}$, $\hat{\omega}_x$, and $\check{\omega}_x$, and another by clocks $\hat{\pi}_{0\text{-}7}$, $\hat{\pi}_x$, $\check{\pi}_x$, where exactly one of clocks is turned on at each point in time. This would result in one shift register leaving data signals, such as A_0, floating at exactly the times when another shift register drives them. As mentioned above, the voltage on A_0 is low on both sides of the handoff, so there is no short circuit even during the handoff. Thus, the circuits in Fig. 16 create a bus based on a naturally extended set of reversible design rules.

J. *Data-enabled clock for a bus interface*

The circuit to generate the clock for each shift register is shown in Fig. 16c and d, which is an enhancement of the circuit in [14]. Shifting a 1 bit into both Fig. 16d and [14, Fig. 10c] at the A_0 position will produce the solid waveforms in Fig. 16b, which are now called $\hat{\omega}_{0\text{-}7}$ and $\hat{\pi}_{0\text{-}7}$. Shifting in a 0 stops the clock, clamping the waveforms at GND and V.

The new signal $\hat{\omega}_x$ is already available in Fig. 16d, but its electrical inverse $\check{\omega}_x$ is dependent on a data signal and is thus not available as a copy of any existing signal $\hat{\omega}_{0\text{-}7}$. Thus, Fig. 16d includes a fifth instance of the circuit in Fig. 16c.

Fig. 16d has been drawn to make it visually evident that the clock relies on only data signals $A_{-2}…A_2$ and makes no connection to the other A_i signals. The functions F_i and the corresponding functions R_i that reverse the computation must be the identity function for $-2 < i < 2$, i. e. they are just buffers, otherwise the shift register will either produce incorrect clock signals or not recover energy properly. However, the functions F_i and R_i for $3 < i < 5$ are unconstrained and can be replaced by arbitrary reversible logic—such as the CNOT gates in Fig. 7e.

Busses are used in computer architecture to connect many subsystems or distantly separated subsystems, so busses frequently present a heavy load, typically a capacitive load, that requires greater current handling capability. For the PL/AL architecture, this means some transistors in both the bus interface in Fig. 16a and the gated clocks driving that interface in Fig. 16d should be made wider, or some other accommodation to increase their drive capability. Fig. 16 identifies these locations with ①②③④.

VI. Appendix C Open source simulator deck

The current version of this document has not been submitted for archival publication, so it may be updated. The following information was current when written.

With the exception of Fig. 4, this document has been backed up by an ngspice simulator input deck.

This deck comprises six files that each start with an Apache 2.0 open-source license statement. The main file is called aa.cir and includes instructions in comment fields.

The installation instructions suggest using ngspice version 36 (because it has been tested with only that version). Gnuplot is needed, otherwise plots will be generated by the more limited ngspice built-in plot functions. The author uses Windows 10 and 11; there has been no testing under any other operating system.

The installation instructions say the code will run with no additional files, but it defaults to a mode that generates only Fig. 1 and Fig. 6.

For other simulations, the instructions indicate that the user must install BSIM models from (a) other places in the ngspice distribution, (b) the Sky130 open source development kit, or (c) obtain models from a source that is not publically disclosed.

The distribution files contain a regression testing capability.

The author has not created a permanent home for these files, but an interested reader is encouraged to check the following URLs:

https://zettaflops.org/aa/

https://debenedictis.org/erik and search the page for a section on the “Adiabatic Analysis ngspice simulator.”

References

[1] Feynman, Richard. "Simulating Physics with Computers." *International Journal of theoretical physics* 21.6 (1982): 467-488. https://catonmat.net/ftp/simulating-physics-with-computers-richard-feynman.pdf.

[2] Fredkin, Edward, and Tommaso Toffoli. "Conservative logic." *International Journal of theoretical physics* 21.3 (1982): 219-253. https://apps.dtic.mil/sti/pdfs/ADA101383.pdf.

[3] Arute, Frank, et al. "Quantum supremacy using a programmable superconducting processor." *Nature* 574.7779 (2019): 505-510. https://www.nature.com/articles/s41586%20019%201666%205

[4] W. C. Athas, L. “J.” Svensson, J. G. Koller, N. Tzartzanis, and E. Y.-C. Chou, “Low-Power Digital Systems Based on Adiabatic-Switching Principles,” *IEEE Trans. VLSI Sys.*, vol. 2, no. 4, pp. 398–407, Dec. 1994. http://www.cisl.columbia.edu/courses/spring-2002/ee6930/papers/00335009.pdf.

[5] Saed G. Younis. *Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic*. No. AI-TR-1500. Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1994. https://apps.dtic.mil/sti/pdfs/ADA290054.pdf.

[6] Athas, William C. "Energy-recovery CMOS." *Low Power Design Methodologies.* Springer, Boston, MA, 1996. 65-100.

[7] Frank, Michael P., et al. "Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS," *2020 IEEE International Conference on Rebooting Computing, online.* At the time of this writing, the conference is over but the paper is not in IEEE Xplore, but see arXiv preprint arXiv:2009.00448 (2020). https://arxiv.org/ftp/arxiv/papers/2009/2009.00448.pdf.

[8] DeBenedictis, Erik P., *Inversion for S2LAL.* Zettaflops LLC Technical report ZF004. http://www.zettaflops.org/CATC/S2LAL_Inv_1.02.pdf.

[9] Lim, Joonho, Dong-Gyu Kim, and Soo-Ik Chae. "Reversible energy recovery logic circuits and its 8-phase clocked power generator for ultra-low-power applications." *IEICE transactions on electronics* 82.4 (1999): 646-653. https://s-space.snu.ac.kr/bitstream/10371/21101/1/Reversible%20Energy%20Rec

overy%20logic%20circuits%20and%20its%208-phase%20clocked%20power%20generator%20for%20ultra-low-power%20applications.pdf.

[10] Rengarajan, Krishnan S., Saroj Mondal, and Ravindra Kapre. "Challenges to adopting adiabatic circuits for systems-on-a-chip." *IET Circuits, Devices & Systems* (2021). https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/cds2.12053.

[11] Celis-Cordova, Rene, et al. "Design of a 16-bit adiabatic microprocessor." *2019 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2019. https://ieeexplore.ieee.org/document/8914699

[12] Pauka, S. J., et al. "A cryogenic CMOS chip for generating control signals for multiple qubits." *Nature Electronics* 4.1 (2021): 64-70. https://manfragroup.org/wp-content/uploads/2021/02/2021.01.25_Nature-Electronics_A-cryogenic-CMOS-chip-for-generating-control-signals-for-multiple-qubits.pdf.

[13] Chao, Rui, and Ben W. Reichardt. "Quantum error correction with only two extra qubits." *Physical review letters* 121.5 (2018): 050502. https://doi.org/10.1103/PhysRevLett.121.050502. https://arxiv.org/pdf/1705.02329.pdf.

[14] DeBenedictis, Erik P. *Energy Management for Adiabatic Circuits*. Zettaflops, LLC Technical Report ZF008, http://zettaflops.org/CATC.

[15] DeBenedictis, Erik P. "Cryogenic Adiabatic Transistor Circuits for Quantum Computer Control." *2021 IEEE 14th Workshop on Low Temperature Electronics (WOLTE)*. IEEE, 2021. https://ar.zettaflops.org/CATC/CATC4QCtl-WOLTE.pdf

[16] Hornibrook, J. M., et al. "Cryogenic control architecture for large-scale quantum computing." *Physical Review Applied* 3.2 (2015): 024010. https://doi.org/10.1103/PhysRevApplied.3.024010. https://arxiv.org/pdf/1409.2202.pdf.

[17] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." *SIAM review* 41.2 (1999): 303-332 https://arxiv.org/pdf/quant-ph/9508027.pdf.

[18] Landauer, Rolf. "Irreversibility and heat generation in the computing process." *IBM journal of research and development* 5.3 (1961): 183-191. https://www.informationphilosopher.com/solutions/scientists/landauer/Landauer-1961.pdf

[19] Böhm, Corrado, and Giuseppe Jacopini. "Flow diagrams, Turing machines and languages with only two formation rules." *Communications of the ACM* 9.5 (1966): 366-371. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.9119&rep=rep1&type=pdf

[20] Fu, Xiang, et al. "eQASM: An executable quantum instruction set architecture." 2019 *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019. https://arxiv.org/pdf/1808.02449.pdf

[21] Ruffino, Andrea, et al. "Integrated multiplexed microwave readout of silicon quantum dots in a cryogenic CMOS chip." *arXiv preprint arXiv*:2101.08295 (2021). https://arxiv.org/pdf/2101.08295.pdf.

[22] Tannu, Swamit S., et al. "Taming the Instruction Bandwidth of Quantum Computers via Hardware-Managed Error Correction." *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2017. http://memlab.ece.gatech.edu/papers/MICRO_2017_1.pdf.

[23] Vandersypen, L. M. K., et al. "Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent." *npj Quantum Information* 3.1 (2017): 1-10. https://juser.fz-juelich.de/record/861563/files/s41534-017-0038-y.pdf.

[24] Wikipedia. *Three state logic*. https://en.wikipedia.org/wiki/Three-state_logic.