Energy Management for Adiabatic Circuits

Zettaflops, LLC Technical Report ZF008, v1.2, April 15, 2021

Erik P. DeBenedictis
Zettaflops, LLC
Albuquerque, New Mexico, USA

***Abstract*—Adiabatic and reversible computing have a previously unappreciated benefit that may make them important for supercomputing and quantum computing applications. This document also extends the concept of universal adiabatic logic to something like sequential logic and automata. While adiabatic and reversible methods did not catch on initially, they demonstrated ways to manage the location where waste energy is turned into heat. This document shows how to exploit this degree of freedom, explaining it with introduction of a new adiabatic logic family called Quiet 2-Level Adiabatic Logic (Q2LAL). Moving energy out of high-performance and quantum chips before turning the energy into heat could allow more capable chips within cooling limits. These extensions increase the range of applications suitable for adiabatic circuits.**

Keywords—adiabatic computing; reversible computing; quantum computer; supercomputer; CMOS; cryo CMOS; automata; Quiet 2-Level Adiabatic Logic; Q2LAL; CATC

I. Introduction

Supercomputers and quantum computers would both benefit by better control of where heat is dissipated, sometimes even at the price of more total heat.

A. Supercomputers

Amdahl's law [1] calls for supercomputers to have some fast processors and other processors that are energy efficient. Amdahl's law says that increasing the amount of parallelism will speed up a computation until an inevitable non-parallelizable serial portion limits further scaling. This document describes an improved technology for the serial processors.

The speed of a chip or module in a supercomputer is limited by heat dissipation in its small area or volume, such as the uppermost module in Fig. 1a. This document presents a new interpretation of adiabatic principles that can move waste energy away from a computational module to remotely located

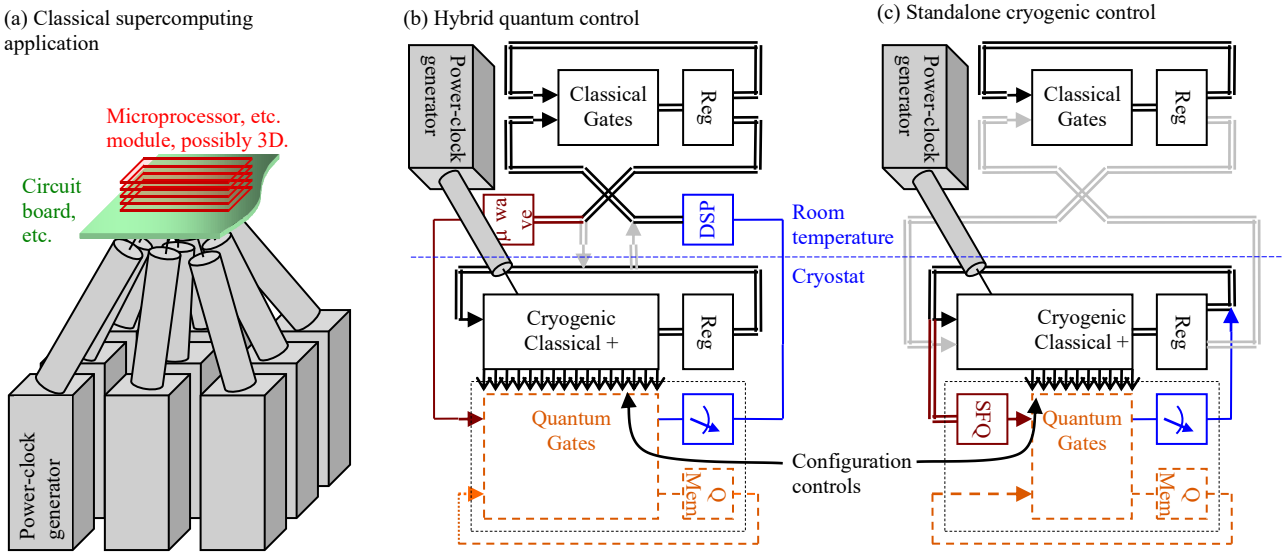


Fig. 1. (a) Adiabatic supercomputing system. The power-clock generation units at the bottom allow the (red) microprocessor module to dissipate less heat at high speed. (b) Coupled classical and quantum automata, where qubits must be in the cryostat but measurement and signal generation must be at room temperature. Reducing power to the cryogenic classical logic is a medium-term goal. Double- and single-lines are classical and quantum data. Black rectangles are classical and dashed orange are quantum automata. Blue is qubit measurement. Dark red is signal generation. (c) Integrated classical-quantum automata communicating with an external classical control system for higher level functions. This is a long-term goal and is even more demanding of energy efficiency in the cryogenic classical logic. DSP = Digital Signal Processing.

power-clock generators where there is more physical space for handling the heat. This will allow more computing within the small volume.

Architecturally, supercomputers support the key concepts in this document already and this document shows how to improve their implementation. In many cases, supercomputer "nodes" comprise a standard microprocessor and a coprocessor based on a graphics display chip. Both processor types mix operations on data with control functions such as conditional branches. The difference is that a microprocessor operates on single- or few-cycle scalar data that make other aspects of the architecture time critical, such as branches and irregular memory access.

In many cases, the microprocessor is responsible for Amdahl's serial code and the coprocessor for the parallel code. The energy management described in this document could speed up the microprocessor.

B. Quantum computers

Quantum computer scale up is currently limited, in part, by heat dissipation and noise in the cryostat. Heat must be removed by a refrigeration system that imposes, for example, a 1,000× energy overhead at 4 K. Electrical and other noise will also interact with qubits and create errors. Due to both heat and noise, many quantum computers put qubits and cryogenic classical devices on different physical structures. This separation permits today's quantum computers to function but creates an impediment to future scale up.

Fig. 1b shows the medium-term goal of a mixed classical-quantum system where classical automata are collocated with qubits in the cryostat. The automata in Fig. 1b and c are shown as classical-quantum finite (Moore) automata to capture the fact that they contain logic and state and capture the direction of communications paths. Each automaton is dawn in the conventional form of asynchronous gates and a state register, although neither quantum nor adiabatic systems have exact analogies to classical gates and registers. A natural implementation of an adiabatic automaton will be presented toward the end of this document.

The classical electronics can be used as configuration controls, but cryogenic implementations are not currently available for signal generation (microwave) and qubit measurement (DSP). The diagram shows the data for these functions being moved back and forth to room temperature, impeding scale up.

Fig. 1c shows the vision of a fully integrated quantum computer as communicating automata. Cryogenic classical gates support a natural set of low-level classical-quantum primitives, effectively the atomic operations or inner loops of a quantum computer, while higher level functions are performed by room temperature electronics [2, 3]. Single-Flux Quantum (SFQ) is a signal type that can be generated and processed in cryogenic classical circuits as well as interact with qubits for control and measurement.

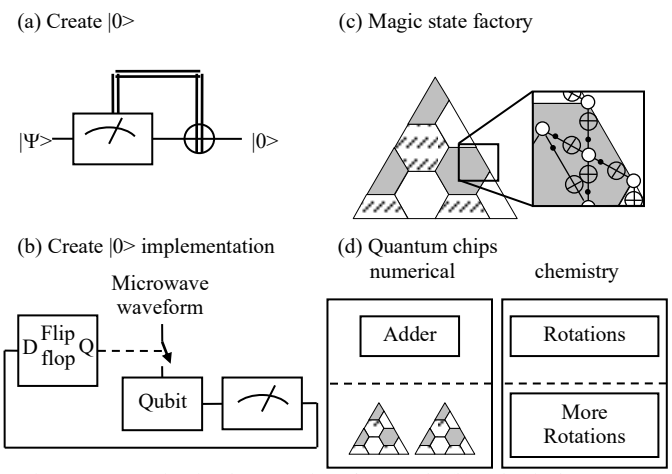


Fig. 2. (a) Circuit for creating |0> and (b) classical-quantum implementation with integrated microwave blocking. (c) Magic state factory [4] and (d) reconfiguration of a quantum chip for numerical and chemistry problems (simplified example).

C. Hybrid Classical-quantum primitives

Qubit initialization, quantum error correction, and magic state preparation can be used as a representative set of low-level primitives for cryogenic qubit types.

Qubit reset, illustrated in Fig. 2a, is the simplest primitive. An effective way to set a qubit in an unknown state to |0> is to measure the qubit. The measurement not only yields a classical 0 or 1 probabilistically, but also forces the qubit into the corresponding |0> or |1> state. So, the process is to latch the classical result into a flip flop, as shown in Fig 2b. If the result is 1, the qubit's quantum state is inverted with a classically controlled CNOT gate. Inverting a qubit's state is performed by exposing the qubit to a microwave signal produced at room temperature. Selectively inverting the qubit's state could be accomplished by a microwave switch collocated with the qubit and controlled by the flip flop.

Magic states are one or more qubits in a specific quantum state, which may encode key parts of certain quantum gate operations, such as the Toffoli gate. Creating a quantum error-corrected magic state, as illustrated in Fig. 2c [4], requires around 100 operations on a dozen qubits or more. The structure in Fig. 2c is like classical logic in that its gates have been physically placed to minimize wire length.

The higher-level use case is illustrated in Fig. 2c, which shows a quantum computer chip in two configurations. To solve an integer math problem, the configuration controls in Fig. 1b and c could set switches in the data path and classical logic so the top half of the quantum computer chip has a quantum adder and the bottom half has two Toffoli magic state factories. Toffoli gates are the main resource used by quantum adders. After the math problem completes, the chip would be configured for the next user, who might want to run a chemistry problem. For chemistry problems, the configuration controls might create gates for precise rotations.

The discussion above motivates the development of innovative technology for implementing classical automata (computers) that minimize heat dissipation and noise in the cryostat—although not necessarily room-temperature

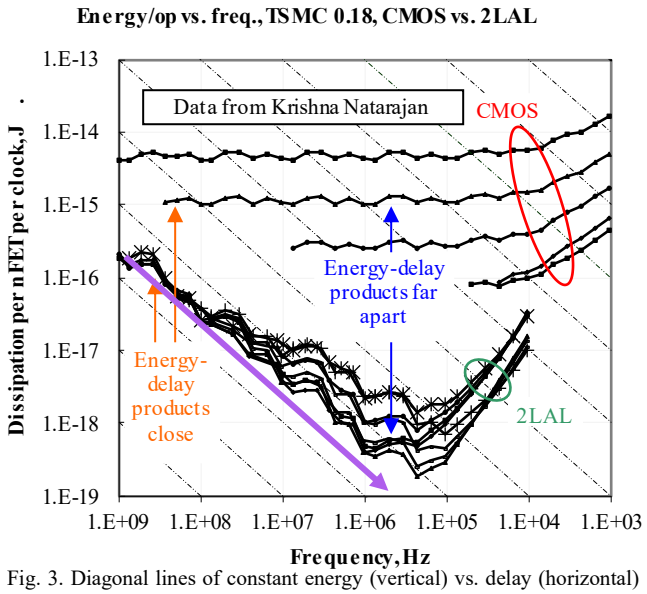


Fig. 3. Diagonal lines of constant energy (vertical) vs. delay (horizontal) product become straight lines sloping downward at –45° on a log-log scale (parallel to the purple arrow). CMOS (orange) and adiabatic circuits (green) are fairly close in energy-delay product at high speeds. However, 2LAL and other adiabatic circuits maintain the energy-delay product along the purple line, giving adiabatic circuits an advantage (blue). This gives greater design flexibility to adiabatic circuits, specifically using high speed logic only where necessary and saving energy elsewhere.

dissipation. The idea is to subdivide a computer into an energy preprocessing subsystem, the power-clock generators in Fig. 1, that enable computational elements located in a special environment some distance away to perform their function more effectively, such as with less dissipation or noise.

D. Existing technology overview

Adiabatic and reversible transistor circuit families emerged in the mid 1990s as a more energy efficient way to use transistors than CMOS's complementary pull-up/pull-down networks and a DC power supply. These circuits have been studied broadly and implemented commercially a few times. Their energy efficiency advantage is shown in Fig. 3, which is a well-known series of simulations comparing CMOS with the same transistors in an adiabatic circuit called 2LAL [5]. CMOS dissipates $\frac{1}{2}CV^2$ energy per operation, which is captured by the horizontal plots in Fig. 3 (which vary in energy and maximum speed as a function of supply voltage). However, the energy per operation of adiabatic circuits decreases linearly with clock period, with Fig. 3 showing up to a 1,000× energy efficiency increase over CMOS. Fig. 3 is based on simulations of 2LAL, but the graphs for other adiabatic and reversible logic families have the same linear drop-off indicated by the arrow.

There are design rules for adiabatic and reversible circuit families, which have names such as 2-level, fully reversible, 2N-2N2P, etc. Prior to this document, the adiabatic family with the largest number of relevant "features" was Static 2-Level Adiabatic Logic (S2LAL) [6], which was touted as "perfectly adiabatic logic." The Q2LAL circuit introduced in this document [7] makes some improvements to S2LAL, but also has a new feature of constant power supply load, i. e. the load is independent of data. Q2LAL uses S2LAL's [6] notation and its circuit as a starting point for a range of new features.

The highest performance chips in supercomputers include "dark silicon," which are areas of the chip that are filled with low energy density circuits, such as memory. This "choice" is dictated by the reality that filling a chip entirely with logic would cause it to overheat. This presents the possibility of an adiabatic processor with an operating point, say, 1/3 of the way down the purple arrow in Fig. 3. Such a processor would need much less silicon to be dark to avoid overheating and could thereby pack more performance into a compact module, yet the fact that it is a single module would allow on-module interconnects to reach more gates within a clock cycle, which is important in top-performing architectures. The memory could be relocated to the third dimension, i. e. stacked.

Fig. 3 also raises the possibility of quantum computer control electronics with an operating point at the tip of the arrow in Fig. 3, yielding a 1,000× energy efficiency increase and supporting more qubits.

The hybrid technology supports a reconfigurable architecture [8] where many slowly functioning transistors are on a base layer, used for memory and other complex logic functions. SFQ logic is layered on top of the transistor layer and can be configured by data in the transistor layer like a Field Programmable Gate Array (FPGA). This structure combines the high speed of the SFQ with the high device density available from transistors. This will be further explained later.

II. New Ideas on Adiabatic Transistor Circuits

This document uses adiabatic circuits in an environment where disruptive effects of electrical noise and the heat differ by location.

A. $\frac{1}{2}CV^2$ energy per operation

To maintain the growth rate of Moore's law [9], industry spent a lot of money on research to reduce CMOS's $\frac{1}{2}CV^2$ energy per operation. While these programs were not successful, the thought experiment in Fig. 4 will, in fact, reduce wall plug energy by 2×—not by changing the $\frac{1}{2}CV^2$ dissipation, but by changing the location where heat is dissipated.

Since the scenarios in Fig. 4 include a cryogenic refrigerator, the 10 KΩ charging resistor can be either part of the room temperature power supply or part of the cryogenic capacitor. The difference is whether the heat dissipated requires an additional 1,000× energy to power a cryogenic refrigerator.

In scenario (a), a capacitor charged from a fixed voltage dissipates $\frac{1}{2}CV^2$. If the heat must be removed from 4 K to room temperature with a 1,000× overhead, the total energy from the wall plug will be $1{,}000\times\frac{1}{2}\ CV^2 = 500\ CV^2$. In scenario (b), dividing the 10 KΩ resistor into two 5 KΩ resistors in series has no effect. One of the 5 KΩ resistors is moved out of the cryostat in scenario (c), with the result that only half the heat flows through the cryo cooler and incurs the 1,000× overhead. The worksheet on the right shows overall power consumption and heat generated declines from 500 CV^2 to

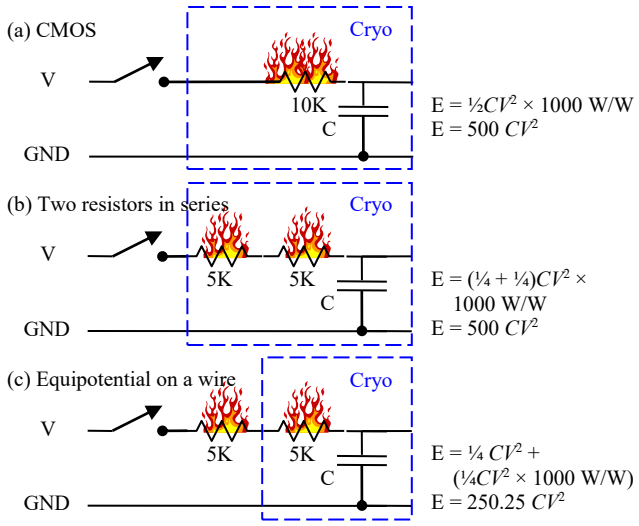


Fig. 4. Refrigerator bypass principle. (a) Charging a capacitor from a fixed voltage dissipates $\frac{1}{2}CV^2$. If the heat has to be removed from 4K with a 1,000× overhead (1,000 W/W), the total energy from the wall plug will be 500 CV^2. (b) Dividing the resistor into two equal parts has no effect. However, (c) if we move one of the resistors to room temperature, only half the heat will flow through the cryo cooler and incur the 1,000× overhead. This reduces overall power consumption and heat generated from 500 CV^2 to 250.25 CV^2 simply due to a circuit properties. The 2× savings can be increased by varying the room temperature resistor during charging.

250.25 CV^2 simply due to the circuit. The remainder of this document shows how varying the room temperature resistor during charging can increase the energy savings beyond 2×.

There is no cooling overhead in the supercomputer scenario. There is instead a cooling limit for the compact microprocessor module motivating the same removal of energy from the module before it is turned to heat.

B. Adiabatic logic circuits and Q2LAL

At least four logic families, SCRL [10], 2LAL [5], S2LAL [6] and now Q2LAL use a common circuit framework and the same notation. Q2LAL can be most effectively explained by symbolic manipulation of S2LAL's circuit equations, which requires knowledge of both S2LAL and the common notation.

Fig. 5 illustrates the signal waveforms for S2LAL and Q2LAL. The primary signal waveforms are in the upper left of Fig. 5a and b and called $\hat{S}$ and $\hat{Q}$ depending on the logic family. The primary waveform in both cases represents a 1 as a positive-going pulse. The circumflex (hat) diacritical mark was chosen because it looks like the waveform of a positive-going pulse.

S2LAL's second waveform in Fig. 5a is carried on a second wire, or rail, that always carries the electrical complement of the first waveform. This means the second wire rests at supply voltage V_{dd} and has negative-going pulses

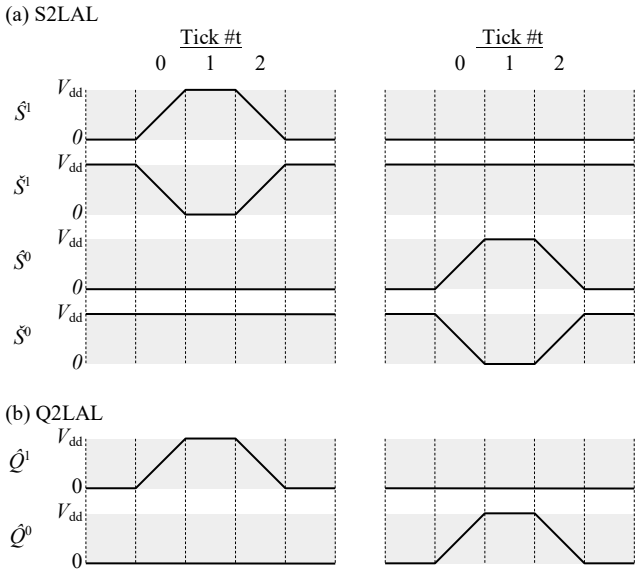


Fig. 5. Signal waveforms. (a) Predecessor S2LAL is dual or quad rail. (b) Q2LAL is dual rail. Notation from [6].

denoted $\check{S}$. The caron (cup) diacritical mark was chosen because it looks like the waveform of a negative going pulse. The absence of a pulse represents a 0 in S2LAL.

S2LAL can store data in a shift register with the dual-rail signaling just described, but it requires two more rails to implement universal logic. The other waveforms are found below the first pair in Fig. 5a. These rails hold the logical complement of the data on the first pair of rails.

Each rail requires a copy of the circuit. S2LAL requires four rails to create universal adiabatic logic, but it would be preferable to have fewer rails.

Q2LAL is dual rail as illustrated in Fig. 5b. Q2LAL signal has two rails and two wires, both of which have a resting state of 0. A positive-going pulse on one wire is called $\hat{Q}^1$ and indicates the transmission of 1, and likewise $\hat{Q}^0$ on the other wire represents a 0.

S2LAL and Q2LAL use the same 8-phase power-clocks illustrated in Fig. 6a, where the 8 phases are called ticks. If the waveforms are considered positive pulses, they are denoted ϕ_i, i=0…7, but they can also be considered negative pulses due to symmetry. So, $\check{\phi}_i = \phi_{i+4 \bmod 8}$, i=0…7. A waveform should follow a linear ramp for the entire duration of the tick. The consistency of the ramp's slope is critical to energy efficiency in all adiabatic circuits, so the unfamiliar reader should not see the ramps as just an artistic convenience.

Both S2LAL and Q2LAL have the same data timing. The data waveforms in Fig. 6b are pulses that are stable for 5 of 8 ticks. The other three ticks include one tick in the resting state and two transition ticks.

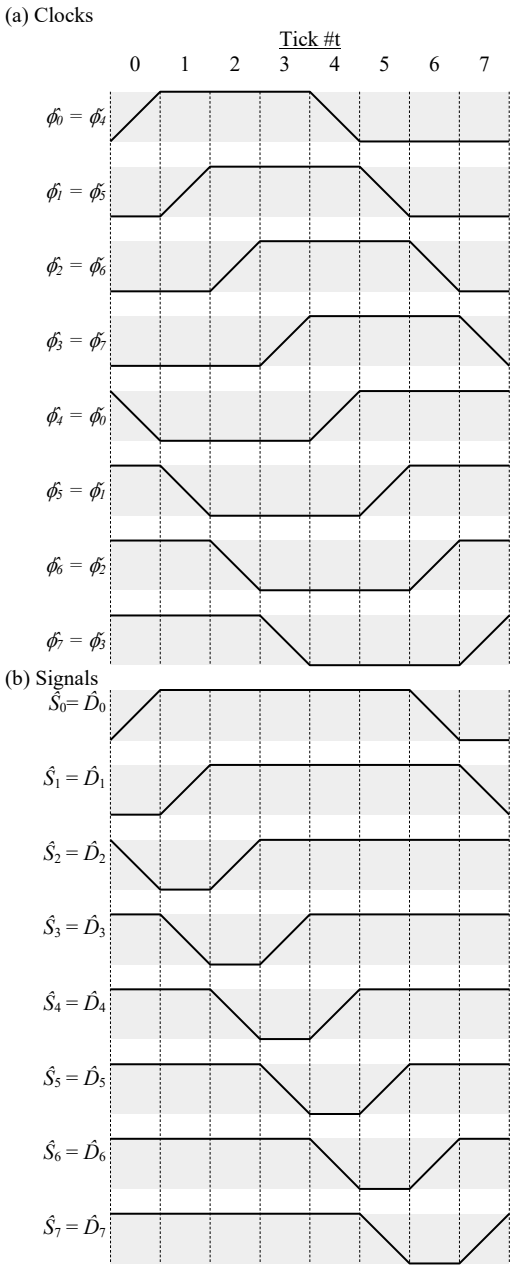


Fig 6. (a) Clocks and (b) data signaling formats are the same between S2LAL and Q2LAL. $\phi_i = \phi_{i+4 \bmod 8}$, but this is not true for the S's. Notation from [6].

All the logic families involve transmission gates. A pair of back-to-back p- and n-channel field effect transistors (FETs) appear as a rectangle with a line connected to the long side, as shown in Fig. 7a. The line represents two electrically complementary signals that go to the transistors' gates. If the lines on the short side of the rectangle are a quad-rail signal, the rectangle implicitly represents two transmission gates or four transistors, also illustrated in Fig. 7a.

The framework involves the coupled cycles in Fig. 7b, where 8 cycles form a complete logic stage in both S2LAL and Q2LAL.

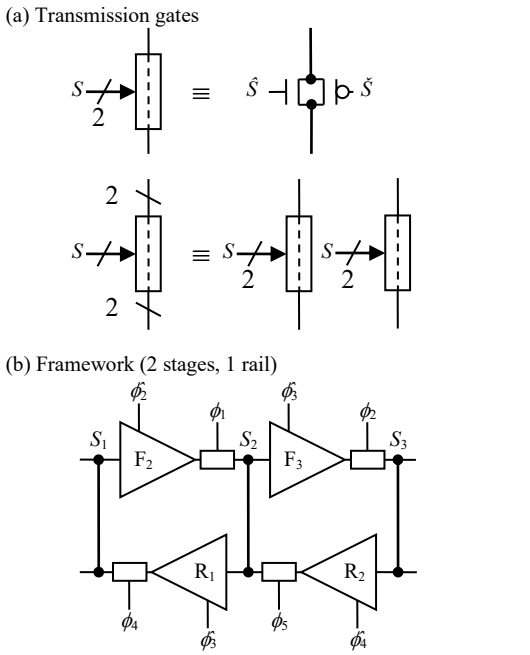


Fig 7. (a) Transmission gate notation, notably including multi-rail. (b) Emerging framework for a SCRL, 2LAL, S2LAL, and Q2LAL. Notation from [6].

The various families differ in the data representation on the lines, the clock sequencing, and the circuitry in the functional units. The lines in SCRL are trits with three signal levels of $V_{dd}/2$, 0, and $-V_{dd}/2$, as opposed to more familiar bits with two signal levels of V_{dd} and 0 in the other families.

Signals are defined by the clock phase or tick where the level becomes valid, so a signal A could be denoted by $\hat{A}_i$, where i identifies the tick.

Prior to Q2LAL, circuits using the framework could perform AND and OR logic functions but could not invert a signal without doubling the number of rails. While there are useful circuits that do not need inversion, such as memory, inversion is needed for universal logic.

The effect of inversion in non-Q2LAL families can be created by duplicating a circuit, except all ANDs are replaced by ORs and vice-versa. If all input data provided to the original circuit is also provided to the copy in a logically inverted form, the two circuits will proceed in lockstep with corresponding signals in the two circuits being logical inverses of each other. With the setup just described, a signal can be inverted by swapping it with the corresponding signal in the other circuit. SCRL stages unavoidably invert data, leading to a similar problem whose solution also requires a copy of the circuit.

However, a Q2LAL signal can be logically inverted by swapping the two wires. There is no need to copy the circuit.

Let us derive Q2LAL by symbolic manipulation of S2LAL's circuit diagram. By replacing negative-going pulses (cups) with functionally equivalent positive-going pulses (hats) representing logically inverted data, we will create circuitry that does not depend on negative-going pulses. This allows us

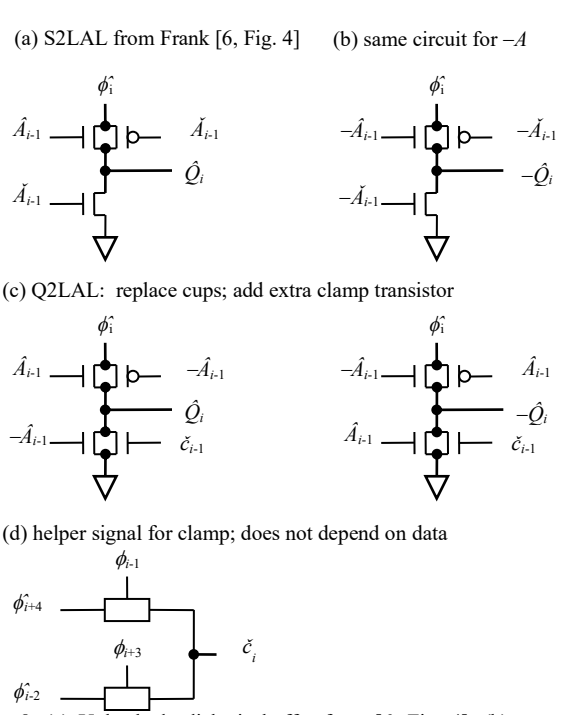


Fig. 8. (a) Unlatched adiabatic buffer from [6, Fig. 4], (b) same buffer for the negated signal, (c) however, the incoming cup signals can be generated from the negated signals in the previous stage, provided that a helper signal $\check{c}_i$ is available. (d) The helper signal can be generated once in an entire circuit from available clocks.

to delete the electrically complemented rail altogether, simplifying the circuit.

Fig. 8 illustrates the circuitry within the triangular structures of the framework called adiabatic amplifiers [11]. Fig. 8a is from S2LAL [6, Fig. 5], but expanding the transmission gate into its two transistors and labeling the input with the applicable phase. Fig. 8b is the same circuitry processing the logically inverted signal $-A$.

The upper symbol $\check{A}_{i-1}$ in Fig. 8a enables one transistor of the transmission gate that connects clock $\hat{\phi}_i$ to the output. In Fig. 8c, we can replace this signal with $-\hat{A}_{i-1}$ because the alternative signal is stable at the correct level when needed to gate the clock and is simply creating a redundant path to ground at other times.

Likewise, the lower symbol $\check{A}_{i-1}$ in Fig. 8a enables the transistor that clamps the output to ground. Replacing that signal with $-\hat{A}_{i-1}$ in Fig. 8c helps if the desired output is a 0 but will leave the output floating between output pulses. This leads us add a transistor gated by the signal $\check{c}_{i-1}$. Waveform $\check{c}_k$ is the electrical inverse of $\hat{D}_k$ in Fig. 6b. Thus, the signal $\check{c}_{i-1}$ goes high during the period where the output needs to be clamped to ground, irrespective of whether the output is a 0 or 1.

Fig. 8d shows how to create the $\check{c}_k$ signal for stage k from four available clocks and four transistors. There would need to be 8 variants of this circuit to create $\check{c}_k$ for $k = 0\ldots7$. However, the $\check{c}_k$'s are independent of data, so each signal can be shared across multiple gates.

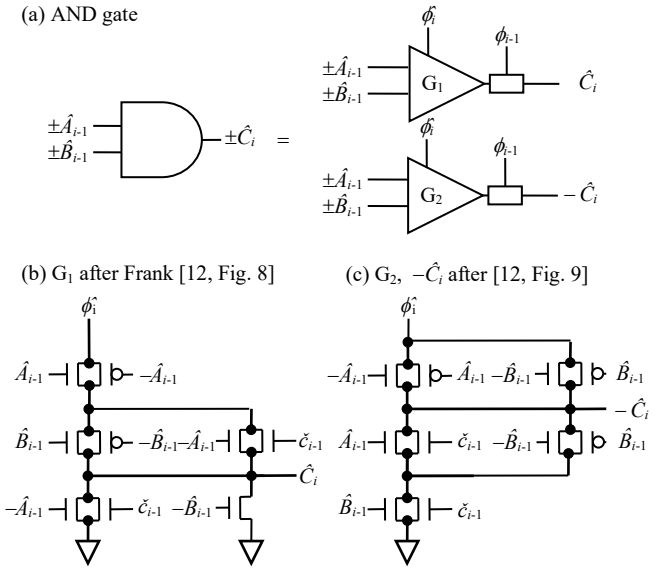


Fig. 9. (a) Definition of AND gate. (b) AND circuit based on S2LAL [6, Fig. 8], but modified for Q2LAL. (c) NAND circuit based on the S2LAL similar to OR gate [6, Fig. 9]. Becomes NAND, OR, NOR with input/output inversions.

Now notice that Fig. 8c and d, the three circuits that become the Q2LAL implementation, contain only $\hat{A}$ and $-\hat{A}$, there is no $\check{A}$, so we define Q2LAL as S2LAL with the second rail logically instead of electrically inverted.

Since Q2LAL signals can be inverted by swapping their wires, AND, OR, NAND and NOR are equivalent up to the labeling of inputs and outputs. Fig. 9 describes a 2-input AND gate and hence demonstrates universality.

The AND gate symbol shown in Fig. 9a defines inputs $\pm\hat{A}$ and $\pm\hat{B}$ and output $\pm\hat{C}$, all as dual rail signals with positive-going pulses. The $+\hat{C}$ pulse will appear when there are pulses on both inputs, which corresponds to a logical AND function. The $-\hat{C}$ pulse would appear in other circumstances, which are readily identified as the result of a logical NAND. Q2LAL uses significantly different circuitry for AND and NAND.

Q2LAL's AND-gate circuitry is the result of the same type of symbolic manipulation used in Fig. 8 to create the Q2LAL buffer. The AND circuit is the result of applying the symbolic manipulation to the S2LAL AND circuit. However, the NAND circuit starts out as a S2LAL OR gate with both inputs having their wires swapped and hence inverted.

The AND circuitry makes use of the clamp signal $\check{c}_k$ described previously.

C. Circuit complexity

There are more transistors in Q2LAL's circuit than S2LAL's, but the two families are closer in complexity than one might think—and Q2LAL pulls ahead when one considers S2LAL's need for a second copy of a circuit for inversion. As described here, Q2LAL adds clamp transistors to what had been an adiabatic amplifier and a circuit to compute $\check{c}_k$. However, these extra costs are offset by some simplifications:

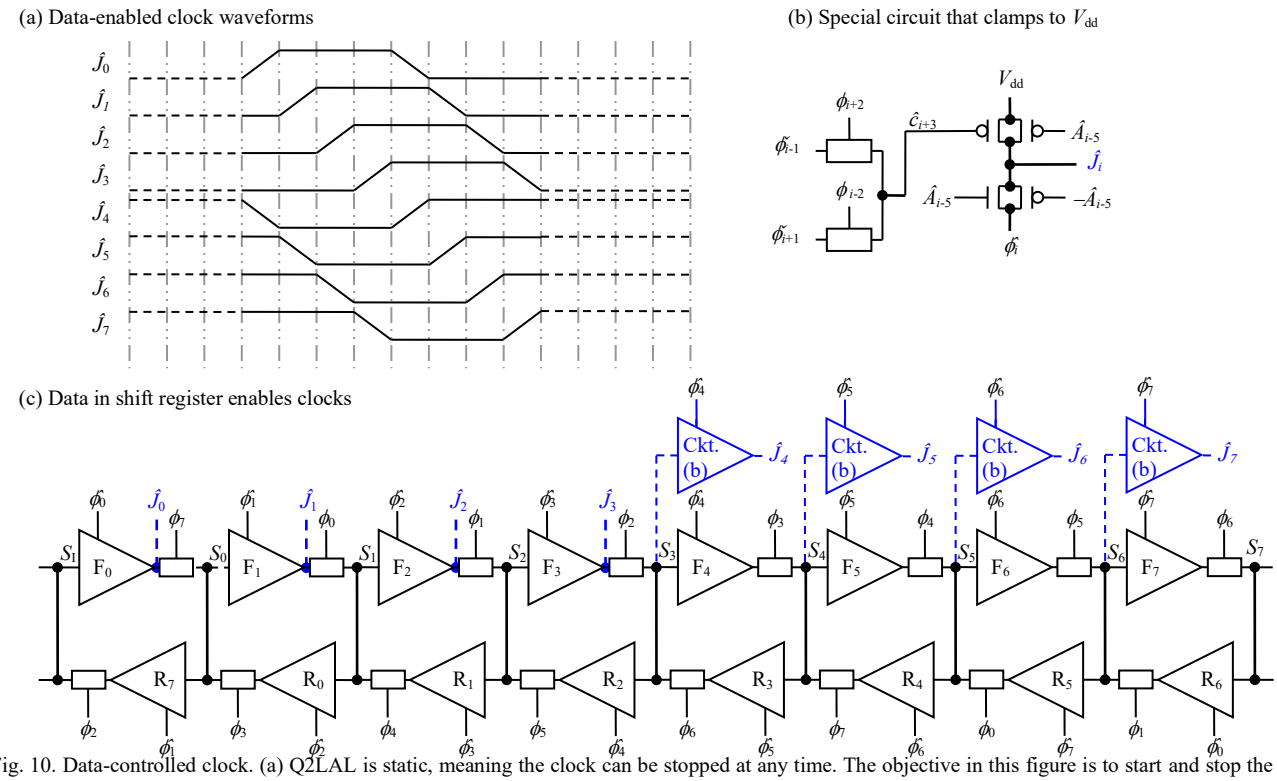


Fig. 10. Data-controlled clock. (a) Q2LAL is static, meaning the clock can be stopped at any time. The objective in this figure is to start and stop the clock at the beginning of the first phase or tick. However, this means the first four clocks will rest at 0 and the second four at Vdd. (b) A circuit for gating the clock that are at V_{dd} when turned off. (c) An 8-phase shift register where a clock enable bit flows left-to-right. The first four positive pulse clocks $\hat{J}$ are just taps of existing signals. However, the other $\hat{J}$ clocks require the special circuit.

Signal $\check{c}_k$ does not depend on data and can be generated once and serve up to, say, 10 gates before the electrical loading becomes excessive. Fig. 8 shows $\check{c}_k$ generated by four transistors, but this could be taken as a 0.4 transistor share of a circuit that generates a standard signal.

Only waveform $\phi\hat{}_k$ is used in Fig. 8, i. e. $\phi\check{}_k$ does not appear. However, both $\phi\hat{}_k$ and $\phi\check{}_k$ are required for the equivalent transmission gate latch in S2LAL [6, Fig. 6]. This permits a circuit simplification called "nFET-only stages" [12]. A person familiar with the literature will realize that an nFET-only stage results from deleting the pFETs from transmission gates. This cuts transistor count but means that voltage swings will decrease slightly. However, a stage with full transmission gates can restore the voltage swings. Thus, the literature shows how to delete the pFETs in even-numbered stages, restoring full signal swing in odd-numbered stages.

D. Data-controlled clocks

A universal logic circuit can compute any function. However, features that go outside the domain of logic can make implementations more efficient. This section describes how one part of a Q2LAL system can turn the clock on and off in another part of a Q2LAL system. Because Q2LAL is static, there is no risk of losing data, as there might be in a dynamic logic family. We will see that turning a clock off not only reduces power but can be a control mechanism like subroutines that can make circuit behaviors easier to create.

While Q2LAL clocks may be stopped at any time, Fig. 10a shows the clocks in Fig. 6c stopped at the beginning of tick 0. The dashed traces show that at this point, the first four clocks are resting at the 0 level and the latter four at the V_{dd} level. The new circuitry in Fig. 10b will be needed to clamp the waveform to the V_{dd} level when a data-controlled clock is stopped.

Shifting a 1 into the shift register in Fig. 10c enables the clock for one 8-tick cycle. This bit will flow across the circuit, creating 8 data-controlled clocks by passing through one of the $\phi\hat{}_i$'s, or a stopped clock by clamping the clock wires to 0 or V_{dd}. A continuous sequence of 1s will keep the clock running continuously.

The first four clocks are called $\hat{J}_i$, i=0…3 and are just taps from existing signals in the framework. The second four clocks need to be clamped to the V_{dd} level when turned off, so the $\hat{J}_i$, i=4…7 clocks are generated by the new circuit in Fig. 10b with the V_{dd} clamp. Data-controlled clocks have the familiar property that $\hat{J}_i$, = $\check{J}_{i+4 \bmod 8}$, i=0…7.

Data-controlled clocks can be designed to stop at the beginning of any tick with one additional consideration. No matter what tick is chosen for the beginning of the stopped clock, there will be four data-controlled clocks resting at the V_{dd} level that must be generated by special circuitry. This circuit references signals $\hat{A}_{i-5}$ and $-\hat{A}_{i-5}$. These are data signals, not clock signals, so the subscript i–5 is not computed with mod 8 arithmetic. It is clear in Fig. 10c that these signals come from shift register stages to the left of special circuit. Thus, there must be four shift register stages to the left of the first

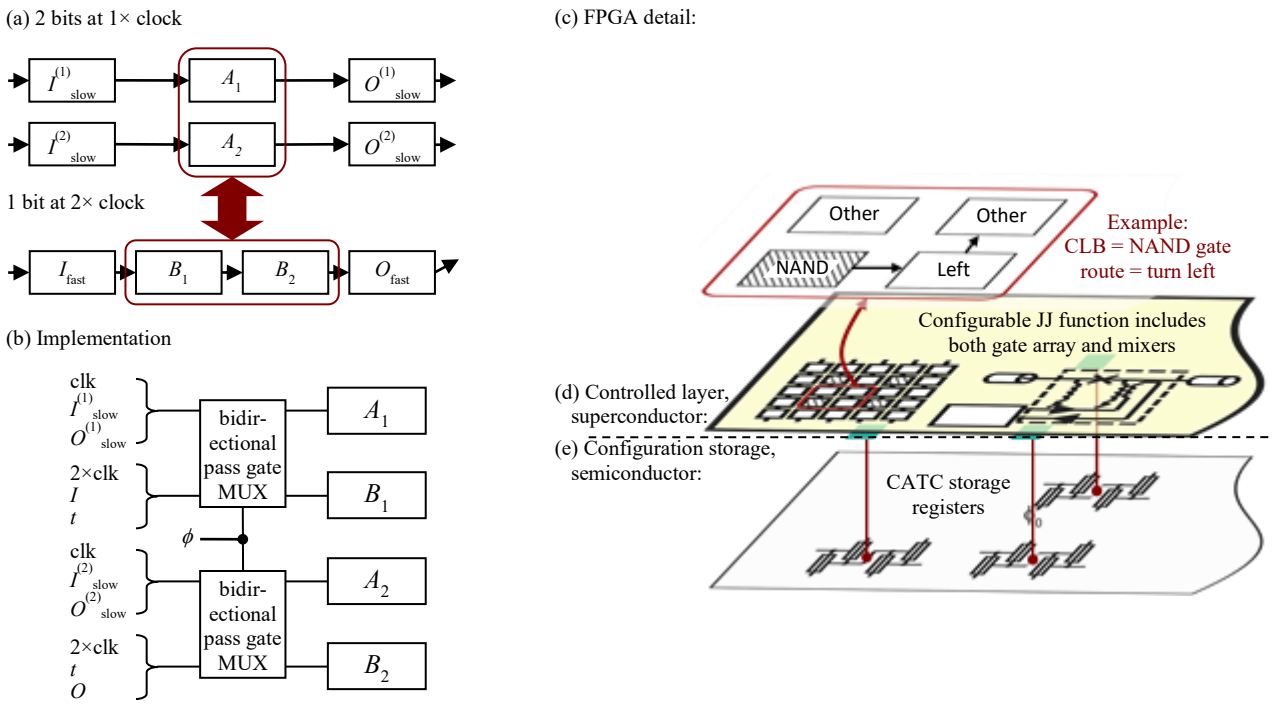


Fig.11. (a) Two domains swap data, where the bottom domain's clock rate is twice as fast. Essentially, (b) two stored bits and their clocks are virtually swapped. The circuit also functions if one of the domains is completely stopped. (c) FPGA-type structure based on DC configuration voltages in a semiconductor base layer feeding data to a superconductor (SFQ) layer or qubits

phase that clamps to V_{dd}, which could require adding stages in some cases.

E. Data transfer between clock domains

The circuit illustrated in Fig. 11 can transfer data between Q2LAL domains with different clock rates, including when one domain's clock is stopped.

Between ticks 4 and 5, a shift register comprised of 8 Q2LAL stages will store a data bit in its inner stages and the inputs and outputs will be in the resting state of 0. The strategy is to switch the overall circuit wiring so the groups of 8 Q2LAL stages swap positions. The circuit schematics for these stages are identical, the only difference being the voltages on internal nodes.

Say we have two Q2LAL domains running at frequencies f and $2f$, as illustrated in Fig. 11a. Say the clocks in the two domains are synchronized so that the points between ticks 4 and 5 align periodically. At these points, the two 8-stage shift registers can be "virtually" swapped using bidirectional (transmission gate) multiplexers addressed by a signal ϕ. The multiplexers would rewire data, clock, and the connection t between the two serial bits from each shift register into the other domain, with the effect that pairs of bits are swapped between the domains. This discussion uses clocks at rates f and $2f$ as an example, but a different ratio simply leads to swapping a different number of bits.

One 8-stage shift register in each domain is swapped when using a data-controlled clock. If both clocks are running, the circuit swaps the data streams between the circuits. If one clock is stopped, the data stream in the running circuit is simply delayed.

This method obeys the design rules for perfectly adiabatic circuits, recovering energy as expected.

F. Configurable logic

Turning the clock off will not only save energy, but the signals in the lower-level components will become DC voltages and potentially useful as configurable DC output voltages, as illustrated Fig. 11c and as the configuration controls in Fig. 1b and c. The DC voltages could turn a JJ FET [13] on or off, which would be able to pass an SFQ pulse or a qubit. In either case, the configuration voltage could configure either classical digital [8], classical analog, or quantum [13] signals in an FPGA-like structure.

An FPGA-like structure is described in Fig. 11c. Many transistors in adiabatic circuits are on a base layer and used for memory and other complex logic functions. SFQ logic is layered on top of the transistors and can be configured by data in the transistor layer similarly to the way a Field Programmable Gate Array (FPGA) is configured by a data string in a serial shift register. In lieu of the transistor layer configuring other transistors, the configuration voltages would be propagated to the superconductor layer through a JJ FET or some other electrical interface to JJ-based SFQ or other JJ circuits. These circuits could be SFQ configurable elements such as gates and routing elements. Analog elements like a microwave switch are another option.

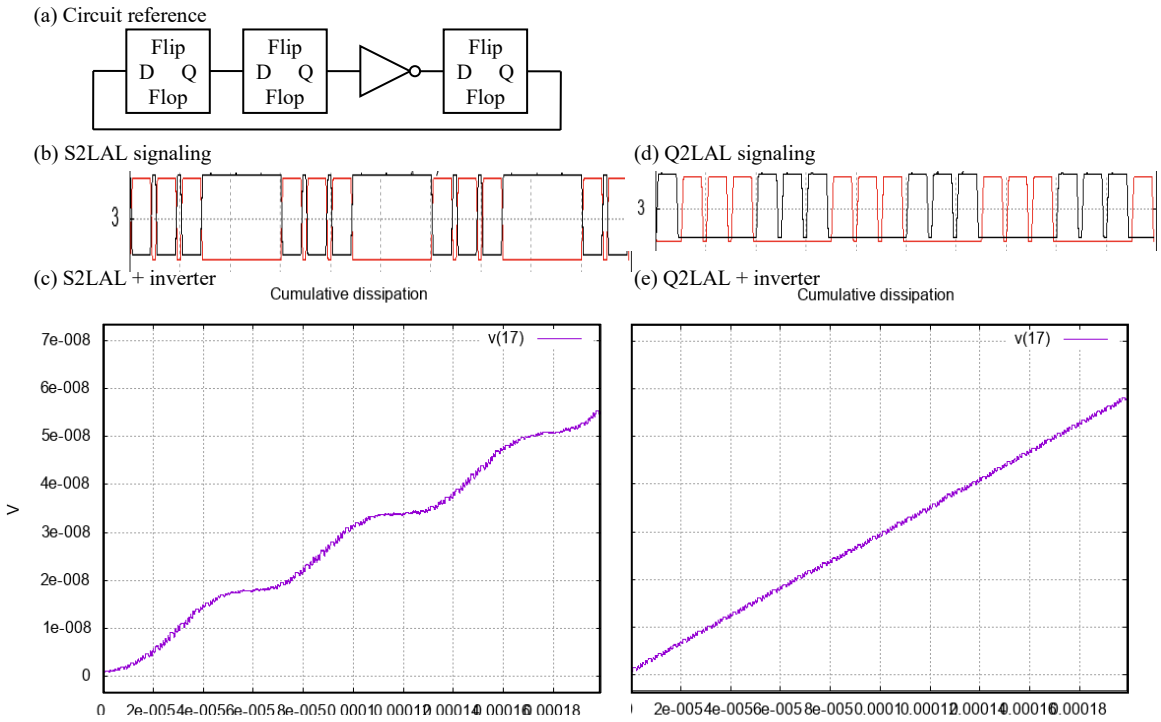


Fig. 12. (a) Circuit reference, generating repeating sequence 000 100 110 111 011 001. (b) S2LAL output $\hat{Q}$ and $\check{Q}$ (red and black) showing one bit position in the circuit (c) S2LAL cumulative dissipation, showing variance as the number of 1s changes. (d) Q2LAL signaling, where either $\hat{Q}$ or $-\hat{Q}$ is a 1 on each clock (e) Q2LAL dissipation, where the total number of 0s and 1s does not change and so the dissipation is constant.

The physical structure in Fig. 11c has precedent. SFQ superconductor chips are fabricated by evaporating the superconductors onto a blank silicon wafer; the hybrid in Fig. 11c would be constructed by using a completed silicon wafer instead.

G. *Even load*

Adiabatic circuits have been developed for computer security purposes that place an even load on the power supply, such as EE-SPFAL [14]. Q2LAL has this even-load property, but not when data-controlled clocks are used. This document will show how the even-load property can facilitate energy management and then generalize the energy management approach to include data-controlled clocks and other non-logic features.

For background, a differential power analysis (DPA) attack attempts to figure out secret information in a chip by measuring changes in power supply current. Fig. 12a is an exemplary circuit that cycles back and forth between being filled with 0s and 1s. If processing a 0 consumes a different amount of power than a 1, measuring the power supply current at a particular point in time may reveal the value of a certain data bit. While the analysis requires knowledge of the circuit and many trials, attackers find it worthwhile for obtaining high-value information such as passwords. There is literature on DPA, but further discussion of computer security is beyond the scope of this document. See ref. [15].

Fig. 12b and c show an ngspice simulation of cumulative energy dissipation of an S2LAL implementation of Fig. 12a and its signaling pattern in Fig. 12b. The curve is horizontal when the circuit is filled with 0s, indicating low dissipation, but rises steeply when filled with 1s, leading to the wavy appearance.

The two circuits in Fig. 8c differ only by swapping A's with $-A$'s. If the circuits are laid out near each other and have similar geometry, the combined electrical characteristics will be the same irrespective of the data.

Fig. 12d is the signaling pattern for the same circuit implemented in Q2LAL, with its dissipation in Fig. 12e. One would expect a linear increase in dissipation over time, which is true to the resolution of the eye.

Q2LAL would thus be suitable for computer security applications, but its even-load feature will be used for energy management later in this document.

H. *Noise issues*

Fig. 13 is a simulation of the circuit in Fig. 12c at different frequencies, plotting the current from the 8 Q2LAL clocks. For comparison, the supply current of a CMOS implementation of the circuit in Fig. 12a would show delta functions of supply current whenever a signal makes a transition. The height of the delta functions would not depend on the clock rate and the bandwidth could be as high as the frequency response f_t of the transistors.

CMOS f_t's can be as high as hundreds of GHz, which is higher than qubit control frequencies. So, qubits exposed to CMOS noise would rotate randomly, leading to errors. However, Q2LAL has less noise, and the noise is at lower frequencies.

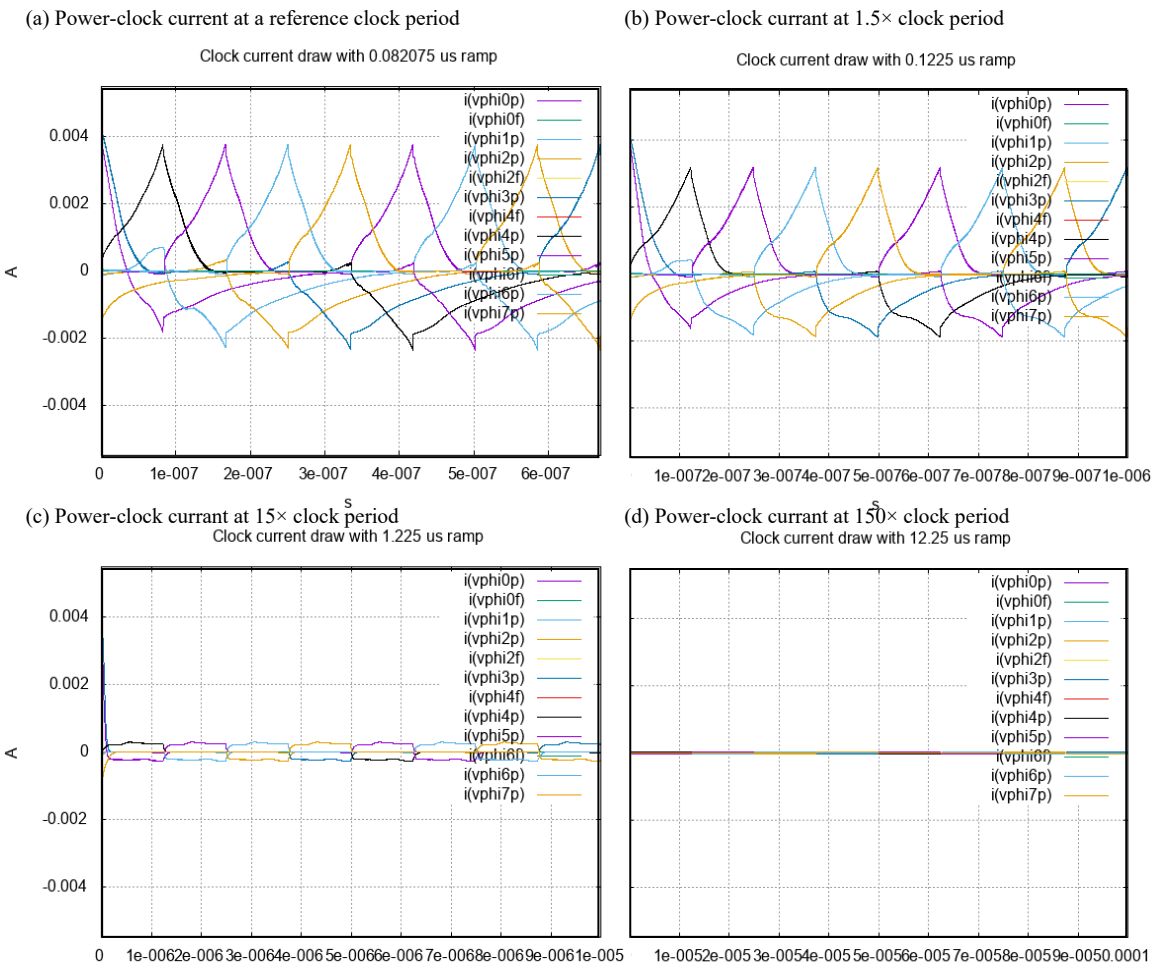


Fig. 13. Q2LAL power supply noise. (a)-(d) Ngspice simulation of circuit in Fig 6c, showing the 8 power-clocks a an arbitrary clock period and 1.5, 15, and 150× the clock period. The length of the simulation increases at the same rate as the clock period, so the data is the same. As speed decreases, clock current becomes a constant charge/discharge current that switches from one clock phase to another.

For Q2LAL, Fig. 13a shows the supply current at a clock period appropriate for the default transistor model built in to ngspice (i. e. absolute speed references are irrelevant) and Fig. 13b shows the same plot at 1.5× the clock period and with a 1.5× horizontal scale. In other words, Fig. 13a and b are logically the same, but time expands. The reader will note that the curves have similar shapes but Fig. 13b has lower amplitude. Adiabatic behavior does not manifest itself at high speeds, so the wave shape and noise are dependent on device characteristics just as they are in CMOS.

Fig. 13c and d show noise decreasing in both amplitude and frequency as the clock period increases further. The circuits and vertical scale are the same as Fig. 13a and b, but the clock periods are 15× and 150× longer with corresponding increases in horizontal scale. Just as with Fig. 13b, the plots are logically the same, but time has been expanded much more. Fig. 13c shows that the jagged curves in Fig. 13b were current trying to rise to a certain level. Fig. 13c shows the currents reaching that level and staying there. Fig. 13d looks like a flat line but expanding the vertical scale (not shown) reveals the same waveform as Fig. 13c, but at lower amplitude and frequency.

I. The adiabatic power train

There can be many implementations of the ideas discussed in this document, but Fig. 14 illustrates the big picture.

The objective is to minimize heat dissipation in computational chips or modules, such as supercomputer or quantum computer control chips.

The cryogenic adiabatic energy management approach requires room-temperature power-clock generators, which are illustrated by the large rectangular structures above the cryostat. The power-clocks will cross the temperature gradient in transmission lines, encountering an improper termination near the computational chip. The transmission line could include filters for frequencies that are not part of the power-clock waveform, as shown in Fig. 14b.

The engineer must assume the waveform generators are launching (predistorted) waveforms into transmission lines intending that they end up as the power-clocks in Fig. 6a. The distance between a clock generator and a CMOS chip today, such as a microprocessor in a server, is a few centimeters compared with about a meter for a system in a cryostat.

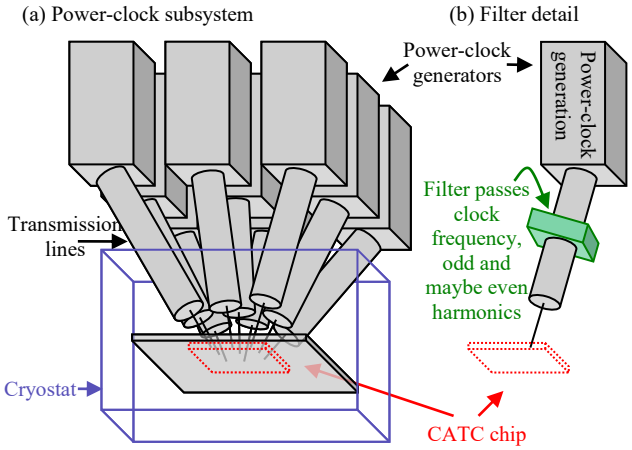


Fig. 14. Power-clock subsystem. (a) Overview comprising signal generators, transmission lines to cold space, and CATC chip. (b) Filter on transmission line.

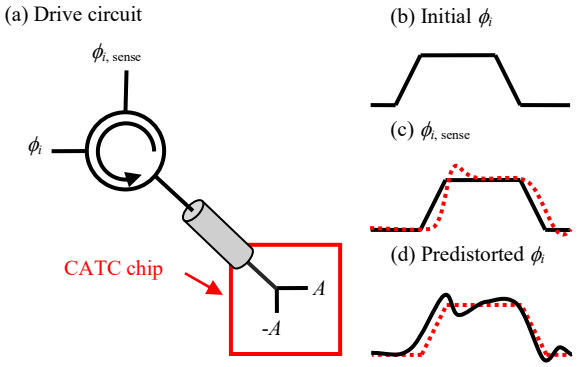


Fig. 15. Driving a ramped clock through a transmission line. (a) Circuit with circulator component; (b) Initial guess at clock ϕ_i, (c) sensing $\phi_{i,\,sense}$ reveals undershoot and overshoot (red) but (d) predistortion ϕ_i can causes signal at chip to be the shape of ϕ_i although delayed.

However, quantum computer control electronics should operate well below qubit control signals of a few GHz, so the transmission line's length will be close to a wavelength of relevant frequencies in either case.

Fig. 15 shows the predistortion strategy. A transmission line has two transmission modes that carry waveforms independently in each direction.

A circulator separates the modes. A circulator is a circuit element with a clean definition and hence convenient for this explanation, yet other methods of separating transmission modes may be more appropriate for implementations.

One mode carries the signal $\phi_{i,\,sense}$, which, if properly terminated and with some mathematics, will reveal the actual waveform that was applied to the chip. A power-clock generator would use knowledge of the signal it transmitted, $\phi_{i,\,sense}$, and other data, such as the length of the transmission line to compute the load and the waveform applied to the chip.

A waveform ϕ_i inserted into the transmission line by the circulator will propagate to the chip using the second propagation mode. After a delay, the waveform will reach the end of the transmission line and encounter the load presented by Q2LAL circuitry, which will not be the characteristic impedance of the transmission line. Improper transmission line termination leads to a reflection back up the transmission line.

Q2LAL power-clocks may go through resistive transistor channels but always end up on the gate of a transistor, i. e. Q2LAL does not include conductive paths between the power-clocks or between power-clocks and ground. Transistor gates are capacitors, so the transmission line will have a capacitive termination.

Since all Q2LAL power-clocks end at capacitors, all the charge that comes from the transmission line goes back into it. In fact, for clock periods much longer than the RC time constant of the circuitry, the reflected waveform will have as much energy as the incoming waveform.

The waveform will also be periodic. The discussion around Fig. 12 shows constant overall dissipation because all logic bits are transmitted in dual rail form with a pulse on wire A or wire $-A$ but not both. Q2LAL implementations should attempt to match the load created by complementary signals by giving them wires of equal length or other methods of matching loading. Given this matching, the waveform will be the same every clock cycle even if the 1s and 0s in the circuit are different.

In the idealized case just described, the waveform appearing at the chip will rise more slowly than the applied voltage and then overshoot. Longer transmission lines, larger transmission line impedance, larger load capacitance, losses in the transmission line, and potential filtering of the signal make the distortion more difficult to conceptualize, but its effect can be predicted with circuit simulation or sensed on $\phi_{i,\,sense}$.

The goal is to compute a predistorted waveform, such as the solid curve in Fig. 15, which has the property that the predistortion plus the distortion yields the desired waveform

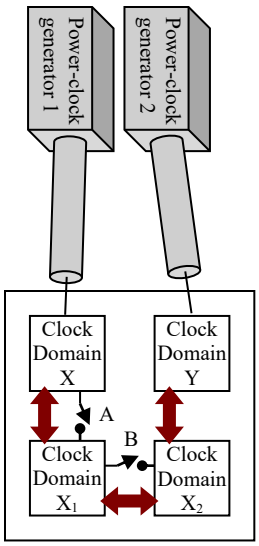
In less challenging situations, it may be possible to compute the predistorted waveform by simulating the design before it is built.

In more challenging situations, the signal generator could monitor $\phi_{i,\,sense}$, to compute the predistortion. The computation could be performed during system startup or even through feedback during operation.

Data-controlled clocks are the most challenging but open new opportunities, as will be discussed below.

J. *Multiple clock domains*

Switching a data-controlled clock, disclosed in Fig. 10, will change the load and make the predistorted waveform incorrect. So, let us define a Q2LAL system as having a "system power mode" for each combination of clock frequencies. For example, Fig. 16 shows an adiabatic chip with four clock domains. In general, each power-clock generator can produce any frequency f within some range. The domains are in a hierarchy, so a domain can pass its clock at frequency f to a domain below it in the hierarchy. A domain can also use the



Generator 1	Switch A	Switch B	Predistort
500 MHz	open	n/a	Wave 1
500 MHz	closed	open	Wave 2
500 MHz	closed	closed	Wave 3
100 MHz	open	n/a	Wave 4
100 MHz	closed	open	Wave 5
100 MHz	closed	closed	Wave 6

Generator 2	Predistort
100 MHz	Wave 7
99 MHz	Wave 8
98 MHz	Wave 9
97 MHz	Wave 10
96 MHz	Wave 11

Fig. 16. Hardware to support predicted loads. Power-clock generators include a table of predistorted waveforms and coordinate with the circuit to apply the correct predistortion for the load at a given instant. This enables a series of new capabilities that are described in the text. The double-ended arrow represents data transfer between domains, Fig. 11a and b.

data-controlled clock as a switch, such as A or B in Fig. 16, and effectively pass a clock at frequency 0.

The power-clock generators store a predistorted waveform for each system power mode in tables. By synchronizing the bits entering the register of Fig. 10c with similar bits in the power-clock generator, the power-clock generator can switch to the correct predistorted waveform when the load changes.

Domains can exchange data as illustrated by Fig. 11a and Fig. 16 when their clocks are at compatible frequencies, as Fig. 11a.

The system power modes and data exchange can work together to create new behaviors for energy efficiency and flexibility of control.

For example, the system modes in Fig. 16 include the common frequency of 100 MHz. This would permit data transfer between domains at that frequency. For example, information about quantum errors detected in one domain could flow to another domain that would configure the hardware to correct the error.

Generator 2 has waveforms for 96-100 MHz. Let us say for sake of argument that the waveform for each frequency is similar enough to the waveform one megahertz higher that the waveform generator could create an adequate predistortion for any frequency in the range 96-100 MHz. Thus, generator 2 would be able to generate an externally defined frequency, such as 98,314,159 Hz. This would give the power-clock domain the ability to generate or process control signals tuned to externally defined frequencies. For example, the adiabatic power-clock domain could be tuned to match a qubit resonance for precise control—or could be tuned so Q2LAL noise such as in Fig. 13a-d would avoid qubit control frequencies that could lead to quantum errors.

The power-clock generators could also change frequency based on computational load, similarly to the microprocessor in a laptop changing clock frequency based on software load and die temperature. The advantage is that the system could slow down calculations that are not on a critical path, saving energy or reducing noise.

K. Automata

Fig. 16 can also represent coupled automata that follow adiabatic design rules. The clock domain X_2 at the bottom of the hierarchy could represent an automaton that performs the control process in Fig. 2a and b that creates a |0> qubit. This automaton is active only when switches A and B are closed.

Clock domain X_1 in the middle of the hierarchy could be an automaton that creates a magic state via the complex sequence of operations, but process is active only when switch A is closed. Magic state production needs many |0> states, which can be obtained by closing switch B repeatedly to activate domain X_2.

The hierarchy of domains shown in Fig. 16, with the switches and data transfer between domains, create a mechanism like subroutine calls. While the circuit for subroutine X_2 always exists as transistors and wires on the surface of a chip, its "calling program" X_1 can choose when the "subroutine" runs and has a mechanism for sending data, like subroutine arguments, and receiving data, like a return value. This control concept is also like threads and coroutines in computer programming, so it is not the intent of this section to be overly specific in its analogies.

L. Filtering

Let us next consider thermal noise entering the cryostat through the power-clocks. As stated earlier, the preferred solution is to filter frequencies that are not essential to the power-clock waveforms. Attenuation is also possible but would dissipate power into the cryostat and attenuate the reflected signal so it would be more difficult to compute the predistortions.

If the load is constant, the required waveform will be periodic in both voltage and current so the predistortion can be fixed. The Fourier decomposition of any periodic waveform contains only multiples of the base frequency, such as the 2^{nd}, 3^{rd}, 4^{th} harmonic. A trapezoidal ramped waveforms in Fig. 6a are anti-symmetric, so they only contains odd harmonics, such as the 3^{rd}, 5^{th}, 7^{th}, and 9^{th} harmonics. However, predistortion is not anti-symmetric and may create even harmonics.

To minimize noise under constant load, each harmonic could in principle be filtered separately with a very narrow pass band. With a 1 Hz pass band and the harmonics in the paragraph above, the Johnson-Nyquist noise power would be 7 kT.

The discussion above illustrates the connection between even load, as illustrated in Fig. 12b, and the engineering of filters for a cryogenic environment. If the load were to vary as indicated by the wavy curve in Fig. 12a, a single predistorted waveform would not be sufficient to create the proper adiabatic waveforms. Instead, the best predistorted waveform would vary across the duration of a wave, having higher amplitude

during the crest of the wave to compensate for the higher load. This change in amplitude is equivalent to modulation, such as Amplitude Modulation (AM) of the signal being sent through the transmission line. Modulation produces additional frequencies, which would be sidebands at plus or minus the frequency of the modulating wave in Fig. 12e. The filters in Fig. 14b would have to be less selective to include these additional frequencies, which would also allow more thermal noise to enter the cryostat. If the clock domains changed at a rate of, say, 1 kHz, the Johnson-Nyquist noise would be 7,000 kT—a lot more than 7 kT but a lot less than the noise from a microwave signal modulated into sub-microsecond pulses used for qubit control.

The criteria above lead to tradeoffs related to filtering. The detrimental effect of room temperature noise entering the cryostat will vary by application. This cost will have to be weighed against the complexity of filtering.

M. Scaling issues

Let us use the simulation in Fig. 17 to illustrate scaling issues. The simulation is of four clock generators like Fig. 15 (although without the circulator) driving the circuit in Fig. 12a. However, Fig. 17 is effectively a simulation of multiple (500) copies of the circuit driven by the same clocks.

The simulation in Fig. 17 has "weakened" transmission lines driving one copy of the circuit. The transmission line parameters were obtained from an SPF-250 datasheet but with parameters L and R increased and C decreased, all by 500×. The resulting voltage waveforms will be correct for multiple copies of the circuit, but current waveforms would have to be increased by 500×.

The reader should first locate the predistorted clocks in Fig. 17. Examine the four colored traces near the "①" symbols and observe that they are ramped waveforms like Fig. 6a but with the ramps elevated or depressed by 10 V. The displacement is substantial, so the wave looks quite different.

Beware that the left hand quarter of Fig. 17 is influenced by simulation start up effects.

The predistorted ramped clocks effectively put a voltage across the characteristic impedance of the transmission line and cause the capacitive load to be charged at constant current. The reader will see the waveforms at the chip, identified with "②," have recognizable ramps but are visibly imperfect.

Adiabatic clock ramps should not overlap to avoid excessive current during the time of overlap. To help, the simulator has a parameterized gap, set to 2%, between ramps to give voltages time to settle. The curve are placed on the chart so the reader can see that one wave settles before the next wave starts it transition—more or less.

The objective is to control a voltage at a distance, but this task that becomes more difficult as the length of the transmission line increases, which is set to 3 ns or 2.5 feet in Fig. 17. The effective length is also directly related to the clock period, set to 3 µs. Control sensitivity is apparent if one changes the transmission line's length or the clock rate because the other parameters will need adjustment to maintain linear but non-overlapping ramps.

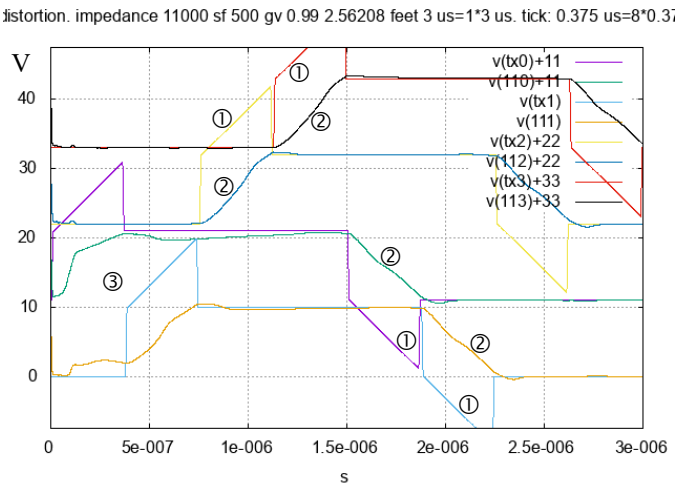


Fig. 17. Limits of predistortion. ① Predistorted clock waveforms, with overdrive on the ramp and a gap between adjacent clocks. ② The resulting waveform is recognizable as a ramp but with visible imperfections. The waveforms are aligned to show the difficulty of one clock having stabilized before the next starts its transition. ③ The left part of the plots have simulation start up effects.

The number of copies of the simulated circuit is also important. The three stages in Fig. 12a put negligible load on a typical transmission line with a characteristic impedance of 50-100 Ω. However, the circuit "borrows" energy from the propagating wave. As the circuit gets larger, the borrowed energy will deplete the wave, lowering the voltage in the wave and to the circuit. The wave must be created with a higher voltage to compensate.

For a high-power circuit, such as for the supercomputer chip in Fig. 1a, the required waveform could have impractically high voltages. The remedy would be to lower the impedance of the transmission line. This could be accomplished by, for example, using multiple transmission lines in parallel.

N. Systems

The technology disclosed in the document leads to a more powerful system model than universal logic, but this will require some additional background to explain.

CMOS is an asynchronous universal logic, but it can do more. Acyclic networks of CMOS gates can compute anything, so CMOS logic is universal. However, it is well known that a pair of cross-coupled CMOS NAND gates can form a flip flop or a memory cell. A flip flop can be used as a frequency divider, creating a clock of frequency $f/2$ from a clock of frequency f. So, in addition to CMOS being able to compute anything, it can perform the non-computational activity of storing data—and hence create sequential logic and finite state machines (also called automata). The state of an automata advances every clock tick based on a state transition function. CMOS can also create the clock, which is not a computational activity.

Clocks derived from other clocks are important architecturally. For example, many microprocessors have a 4 GHz clock for the processor core, but the memory interface

runs on a, say, 1/5th speed clock that has been generated locally. Memory chips will use the slower clock.

However, adiabatic gates are clocked. Cross-coupled adiabatic NAND gates become a 2-bit shift register rather than a memory cell. Fig. 10 shows how to generate a slower clock, but it requires the additional circuitry in Fig. 10b that is not an adiabatic gate. Two different clocks can be used, but without the ability to move data between clock domains, the two clocks would control independent systems and could not be called a system with multiple clock rates. The circuitry in Fig. 11b is not an adiabatic gate, so adiabatic gates alone are not as capable as CMOS gates.

However, this document has used Q2LAL to extend the original concept of an adiabatic logic family. Q2LAL plus the enhancements in Fig. 10 and 11a and b can implement finite state automata. These automata operate with reduced power consumption based on the principle in Fig. 4 and have lower noise based on Fig. 13. Paralleling S2LAL being touted as "perfectly adiabatic logic" [6], Q2LAL could be touted as "perfectly adiabatic sequential logic with predictable loads."

III. Conclusions

Adiabatic and reversible transistor circuits were introduced starting in the mid 1990s as a way of reducing computational energy. The circuits worked as intended but translating their advantage to a power savings at the wall plug would have required an energy recycling power supply. No such supply has appeared to date. This document introduces a change to the original plan that may lead to practical use.

The change is not to try and save wall-plug power directly, but to reduce the amount of heat dissipated in certain environments, such as key portions of supercomputers and quantum computers.

The advance is to explicitly divide what has been considered a logic family into a computational circuit and an energy preprocessing subsystem, with the two parts located some distance from each other. The design rules for adiabatic circuits will be met within the environment, but not everywhere. However, we find important applications that will benefit from this more limited advantage.

The new Q2LAL circuit combines ideas from traditional adiabatic logic and a branch developing around computer security. People investigating adiabatic logic for computer security have found circuit families that place a very even load on the power supply, yet computer security does not specifically require high energy efficiency. Q2LAL is fully adiabatic and has an even load, meaning that the purple arrow in Fig. 3 would extend forever if transistors had zero gate and source-drain leakage.

This document goes beyond features that are normally associated with logic. A Q2LAL system not only tolerates unusual environments but thrives in them—such as utilizing a temperature difference to improve energy efficiency, adjusting clock rates to match critical paths in the algorithm, noise requirements, or I/O. The domains created by clock rate adjustments can communicate, leading to adiabatic sequential logic, automata, and an subroutine capability.

This document also increases the scope of adiabatic designs and their design rules. Adiabatic principles used to apply to (universal) logic and separately to computer security. This document describes logic in a multi-temperature environment and considers systems with multiple clock rates. The flexibility to mix clock rates allows finer control of energy efficiency and a new ability to control computational noise.

A supercomputer is comprised of serial and parallel components, with most of the expense being in the parallel components. However, inherently serial portions of many algorithms cause large and expensive parallel resources to lie idle much of the time. The idea is to improve the supercomputer's overall efficiency by making an exceptionally high-speed serial processor that would raise the utilization of the more expensive parallel resource. The enhancement would not be in terms of energy efficiency per operation, but rather in the ability to pack a lot of computing in a small volume without overheating.

The quantum computer use case is architecturally similar to current cryo CMOS architectures and demonstrations, but the technology in this document essentially causes 99% or more of the cryo CMOS dissipation to bypass the refrigeration system and hence avoid its 1,000× overhead (at 4 K). Of course, the goal is not to save energy, but to make a quantum computer with more qubits.

In addition, the agility of adiabatic logic to move between clock rates enables interfacing to exotic physical systems—including but not limited to qubits.

Notes on Notation

This document was written with the intent of following Mike Frank's notation in [6], but writing this document revealed some points for reconciliation. Future versions of this document may address these points and this section may be deleted:

The notation in [6, Fig. 3] is uses a slash through a wire with a "2" to indicate dual rail. However, this terminology is not used consistently, such as [6, Fig. 6]: Does S_0 refer both the hat and cup? If the absence of a hat and a cup refer to both, then how do you refer to a single wire? Could a simultaneous hat and cup be designated by boldface or perhaps yet another diacritical mark ($\tilde{}$ or $\bar{}$)?

The "adiabatic amplifier" in [6, Fig. 4] should have tick numbers consistently, such as on the A_{i-1}'s.

For support of higher level structures, like the data-controlled clocks and shift registers, it would be more reasonable to start the clocks on what is phase 4 in [6, Fig. 2] rather than phase 0. At this alternate starting point, data is entirely contained within an 8-stage shift register such as in Fig. 10c].

Acknowledgment

Michael P. Frank has made many contributions to reversible computing over the years. Mike championed the framework in Fig. 7b that contains at least four circuit families so far—plus versions within each family containing different numbers of cycles. Mike also developed S2LAL and a

consistent terminology [6], both of which became a starting point for this work. This document uses Mike's terminology, including diagrams, with his permission.

REFERENCES

[1] Amdahl, Gene M. "Validity of the single processor approach to achieving large scale computing capabilities." *Proceedings of the April 18-20, 1967, spring joint computer conference*. 1967.

[2] R. McDermott, et al. "Quantum–classical interface based on single flux quantum digital logic." *Quantum science and technology* vol. 3, no. 2, 2018: 024004

[3] Wang, Albert. "Clock generation for a photonic quantum computer to convert electrical pulses into a plurality of clock signals." U.S. Patent No. 10,379,420. 13 Aug. 2019.

[4] Chamberland, Christopher, and Kyungjoo Noh. "Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits." *npj Quantum Information* 6.1 (2020): 1-12.

[5] V. Anantharam, M. He, K. Natarajan, H. Xie, and M. P. Frank. "Driving fully-adiabatic logic circuits using custom high-Q MEMS resonators," in *Proc. Int. Conf. Embedded Systems and Applications and Proc. Int. Conf VLSI (ESA/VLSI)*. Las Vegas, NV, pp. 5-11.

[6] 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).

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

[8] DeBenedictis, Erik P. "Quantum Computer Control using Novel, Hybrid Semiconductor-Superconductor Electronics." *arXiv preprint* arXiv:1912.11532 (2019).

[9] Moore, Gordon E. "Cramming more components onto integrated circuits, Electronics, 38: 8 (1965)." *URL: ftp://download. intel. com/research/silicon/moorespaper. pdf* 16 (2005).

[10] Saed G. Younis. *Asymptotically Zero Energy Computing Using SplitLevel Charge Recovery Logic*. No. AI-TR-1500. Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1994.

[11] 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.

[12] E. DeBenedictis, *Enhancements to Adiabatic Logic for Quantum Computer Control Electronics*, technical report ZF002, http://www.zettaflops.org/CATC.

[13] Sardashti, Kasra, et al. "Voltage-tunable superconducting resonators: a platform for random access quantum memory." *IEEE Transactions on Quantum Engineering* 1 (2020): 1-7.]

[14] Kumar, S. Dinesh, Himanshu Thapliyal, and Azhar Mohammad. "EE-SPFAL: A Novel Energy-Efficient Secure Positive Feedback Adiabatic Logic for DPA Resistant RFID and Smart Card," in *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 281-293, 1 April-June 2019, doi: 10.1109/TETC.2016.2645128.

[15] Moradi, Amir, and Axel Poschmann. "Lightweight cryptography and DPA countermeasures: A survey." *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, 2010.

APPENDIX: NGSPICE FILE

The file below includes

- S2LAL basic circuits
- Q2LAL basic circuits
- The even load comparison between the two, generating the graphs in Fig. 12.
- The noise graphs in Fig. 13.
- Testing for the AND gate circuits in Fig. 9.
- Simulating the power train, including predistortion and transmission lines. Generates Fig. 17.
- There is also code for testing an extended clock phase and modulating the onset of the ramp in the Q/S2LAL clocks.

The code uses built-in transistor models, which are based on obsolete transistors. Therefore, no absolute performance is revealed.

A. Q2LAL.cir

```
  Q2LAL
* Proprietary information of Zettaflops LLC. Not for public distribution.
* Q2LAL initial test setup. Q2LAL is "quiet 2LAL" derived from Static 2-Level Adiabatic Logic (S2LAL). More information at the end of the file.
*
* Instructions for duplicate the figures in [ZF008] and several PowerPoints:
* Fig q2la1 periods period FastSlow Porch GentleT  GV GW   ylimit   dcc cwf  sf  Imped
* 12c   0     20      10u      1      .1    .15     .15  -200u 200u  0   0
* 12e   1     20      10u      1      .1    .15     .15  -200u 200u  0   0
* 13a   1      1     .67u      0     .01    .15     .15    -5m 5m    0   0
* 13b   1      1       1u      0     .01    .15     .15    -5m 5m    0   0
* 13c   1      1      10u      0     .01    .15     .15    -5m 5m    0   0
* 13d   1      1     100u      0     .01    .15     .15    -5m 5m    0   0
* 13+   1      1     100u      0     .01    .15     .15  -400m 400m  0   0
* 17    1      1       3u      0     .01    .02 .99 1.99   -5m 5m    0   1  500 '22*sf'
* xxx   1      1      10u      0     .01    .05     .15  -400m 400m  0   0
* xxx   1      1     100u      0     .01    .05     .15   -50m 50m   0   0

* To duplicate slides in [Q2LALv2.ppt]. Slide number:
*  5    1      1     .67u      0     .01    .15     .15    -5m 5m    0   0
*  6    1      1       1u      0     .01    .15     .15    -5m 5m    0   0
*  7    1      1      10u      0     .01    .15     .15    -5m 5m    0   0
*  8    1      1     100u      0     .01    .15     .15    -5m 5m    0   0
* 10    1      1     .67u      0     .01    .15     .15  -200u 200u  0   0
* 11    1      1       1u      0     .01    .15     .15  -200u 200u  0   0
* 12    1      1      10u      0     .01    .15     .15  -200u 200u  0   0
* 13    1      1     100u      0     .01    .15     .15  -200u 200u  0   0

* Slide deck [ZF007] is the same as above but Porch is .1

* Predistortion waveform
* ???   1      1     100u      0     .01    .02 .15 1.15 -200u 200u  0   1 2000 '25*sf'
* ???   1      1      10u      0     .01    .02 .33 1.33   -5m 5m    0   1  500 '25*sf'
* ???   1      1     100u      0     .01    .02 .03 1.03 -200u 200u  0   1  500 '25*sf'
* ???   1      1      10u      0     .01    .02 .05 1.05   -5m 5m    0   1  100 '25*sf'
* ???   1      1     100u      0     .01    .02 .02 .98  -200u 200u  0   1  100 '25*sf'

.param q2la1=1                                          $ nonzero for q2la1; otherwise s2la1
.param periods=20                                       $ number of repetitions of the basic waveform
.param period= 10u                                      $ period of the clock waveform, which comprises a number of ticks
.param FastSlow=1                                       $ 0 for regular clock 1 for several waveforms having fast and slow versions
.param Porch=.1                                         $ A tick as this proportion of 0 V gap at the start and end, so the spacing is twice this
.param GenT=.15                                         $ Gentle rise time as a proportion of the period
.param GenV=.15                                         $ Gentle rise voltage as a proportion of the voltage
.param GenW=.85                                         $ Voltage at start of third segment as a proportion of the voltage
* vertical scale must be set manually on lines identified BOOKMARK1
.param dcc=0 $ manage comments on lines identified BOOKMARK2$ demonstrate data controlled clock [ZF005 Fig. 10] (consumes power when on)
.param cwf=0                                            $ generate clamp waveform from (0) ngspice waveform generator or (1) [ZF008 Fig 8d] (consumes power when on)
.param sf=100                                           $ Number of copies of the circuit. Actually, a factor "weakening" the transmission line
.param Imped=0                                          $ Effective resistance in series with the clock
.param Gain=1                                           $ (Subunity) gain of the transmission line's simulation model, i. e. clock processing
.param ats=0                                            $ include code to test AND gates. Wire swap inversions turn AND into NAND, OR, and NOR
.param Delay=3e-9                                       $ Transmission line delay
* There are three sets of plot commands at the end. Comment out either "plot" or "gnuplot"

.MODEL p1 pmos(LEVEL=49 version=3.3.0)
.MODEL n1 nmos(LEVEL=49 version=3.3.0)

.param CLAMP=1                                          $ clamp transistor of Athas's adiabatic amplifier [Athas], set to 0 to disable
.param ACAP=2e-12                                       $ capacitive load on the data line
.param QQCAP=0e-12                                      $ capacitive load on the internal QQ node

*** SUBCIRCUIT DEFINITIONS
* [S2LAL Fig. 4], Athas's adiabatic amplifier but with complementary voltages on the two halves [Athas]
.SUBCKT AAMP AT AC T C piT piC GND PWR nsub psub ini='gg'   $ [Athas] adiabatic amplifier. Args: AT/C T/C clockT/C substrate supplies
.ic V(T)='ini' V(C)='vv-ini'                            $ .ic V(a)={gg} V(a2)=ini
M0 piT AT T nsub n1                                     $ pass gate
M1 piT AC T psub p1
M2 piC AT C nsub n1                                     $ pass gate
M3 piC AC C psub p1
.if (CLAMP=1)
M4 GND AC T nsub n1                                     $ clamp
M5 PWR AT C psub p1
.endif
.ENDS AAMP

* [S2LAL Fig. 5]
.SUBCKT LATCH AT AC QT QC piT piC pjT pjC GND PWR          $ One phase of the 2LAL shift register. Args: AT/C QT/C clock0T/C clock1T/C
+ nsub psub tap0 tap1 tap2 tap3 ini='gg'                 $ substrate supplies
R0 tap5 QT 1                                            $ circuit taps for debugging
X1 AT AC T C piT piC GND PWR nsub psub AAMP ini='ini'
M1 T pjT QT nsub n1                                     $ Frank's latch
M2 T pjC QT psub p1
```

```
M3 C pjT QC nsub n1                                     $ Frank's latch
M4 C pjC QC psub p1
C1 AT 0 ACAP
C2 AC 0 ACAP
C3 T 0 QQCAP
C4 C 0 QQCAP
.ENDS LATCH

* [S2LAL Fig. 6], except this is just the first stage; shift clocks for subsequent stages
.SUBCKT PHASE S0T S0C S1T S1C                           $ One stage of the 2LAL shift register. Args: AT/C QT/C
+ p0T p0C p1T p1C p2T p2C p3T p3C GND PWR nsub psub     $ 4x{ phi<n>T/C } DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 ini='gg'
X0  S0T S0C S1T S1C p1T p1C p0T p0C GND PWR nsub psub tap0 tap1 tap2 tap3 LATCH ini=ini
X10 S1T S1C S0T S0C p2T p2C p3T p3C GND PWR nsub psub tap4 tap5 tap6 tap7 LATCH ini=ini
.ends PHASE

* [S2LAL Fig. 6], except this is all 8 stages
.SUBCKT SDELAY S0T S0C S8T S8C                          $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p1T p2T p3T p4T p5T p6T p7T                       $ clocks/power supplies
+ GND PWR nsub psub                                    $ DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 tap8 tap9 tapA tapB ini='gg'
R0 tap0 S0T 1                                           $ circuit taps for debugging
R1 tap1 S0C 1
R2 tap2 S1T 1
R3 tap3 S1C 1
R4 tap4 S2T 1
R5 tap5 S2C 1
R6 tap6 S3T 1
R7 tap7 S3C 1
R8 tap8 S4T 1
R9 tap9 S4C 1
RA tapA S5T 1
RB tapB S5C 1
RC tapC S6T 1
RD tapD S6C 1
RE tapE S7T 1
RF tapF S7C 1
X0  S0T S0C S1T S1C p0T p4T p1T p5T p2T p6T p3T p7T GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 PHASE ini=gg
X1  S1T S1C S2T S2C p1T p5T p2T p6T p3T p7T P4T p0T GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 PHASE ini=ini
X2  S2T S2C S3T S3C p2T p6T p3T p7T P4T p0T P5T p1T GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 PHASE ini=ini
X3  S3T S3C S4T S4C p3T p7T P4T p0T P5T p1T P6T p2T GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 PHASE ini=ini
X4  S4T S4C S5T S5C P4T p0T P5T p1T P6T p2T P7T p3T GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 PHASE ini=ini
X5  S5T S5C S6T S6C P5T p1T P6T p2T P7T p3T P0T p4T GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 PHASE ini=ini
X6  S6T S6C S7T S7C P6T p2T P7T p3T P0T p4T P1T p5T GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 PHASE ini=gg
X7  S7T S7C S8T S8C P7T p3T P0T p4T P1T p5T P2T p6T GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 PHASE ini=gg
.ENDS SDELAY

* This is an inverting version of the phase circuit. It simply reverses the input wires.
.SUBCKT PHASEv S0T S0C S1T S1C                          $ One stage of the 2LAL shift register. Args: AT/C QT/C
+ p0T p0C p1T p1C p2T p2C p3T p3C GND PWR nsub psub     $ 4x{ phi<n>T/C } DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 ini='gg'
X0  S0C S0T S1T S1C p1T p1C p0T p0C GND PWR nsub psub tap0 tap1 tap2 tap3 LATCH ini=ini
X10 S1C S1T S0T S0C p2T p2C p3T p3C GND PWR nsub psub tap4 tap5 tap6 tap7 LATCH ini=ini
.ends PHASEv

* This is an inverting version of the delay circuit. It simply calls PHASEv at a point that doesn't interfere with initialization.
.SUBCKT SDELAYv S0T S0C S8T S8C                         $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p1T p2T p3T p4T p5T p6T p7T                       $ clocks/power supplies
+ GND PWR nsub psub                                    $ DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 tap8 tap9 tapA tapB ini='gg'
R0 tap0 S0T 1                                           $ circuit taps for debugging
R1 tap1 S0C 1
R2 tap2 S1T 1
R3 tap3 S1C 1
R4 tap4 S2T 1
R5 tap5 S2C 1
R6 tap6 S3T 1
R7 tap7 S3C 1
R8 tap8 S4T 1
R9 tap9 S4C 1
RA tapA S5T 1
RB tapB S5C 1
RC tapC S6T 1
RD tapD S6C 1
RE tapE S7T 1
RF tapF S7C 1
X0  S0T S0C S1T S1C p0T p4T p1T p5T p2T p6T p3T p7T GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 PHASE ini=gg
X1  S1T S1C S2T S2C p1T p5T p2T p6T p3T p7T P4T p0T GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 PHASE ini=ini
X2  S2T S2C S3T S3C p2T p6T p3T p7T P4T p0T P5T p1T GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 PHASE ini=ini
X3  S3T S3C S4T S4C p3T p7T P4T p0T P5T p1T P6T p2T GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 PHASE ini=ini
X4  S4T S4C S5T S5C P4T p0T P5T p1T P6T p2T P7T p3T GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 PHASE ini=ini
X5  S5T S5C S6T S6C P5T p1T P6T p2T P7T p3T P0T p4T GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 PHASE ini=ini
X6  S6T S6C S7T S7C P6T p2T P7T p3T P0T p4T P1T p5T GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 PHASEv ini=gg
X7  S7T S7C S8T S8C P7T p3T P0T p4T P1T p5T P2T p6T GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 PHASE ini=gg
.ENDS SDELAYv

* Erik's "two hat" adiabatic amplifier. In S2LAL notation, it expects data input as A^ and -A^. Given this, it produces the correct output [ZF008 Fig. 8c (both sides)].
* Same role in framework as [S2LAL Fig. 4], Athas's adiabatic amplifier but with complementary voltages on the two halves [Athas]
*  :           :           :           :           :           :           :           :           :
* 1: A(i-1)^   2: -A(i-1)^
* 3: Q(i)^     4: Q(i)^
* 5: phi(i)^   6: Clmp(i-1)v
* 7: GND
* 8: nsub      9: psub
.SUBCKT QAAmp AT AC T C pT C1 GND nsub psub ini='gg'    $ Erik's adiabatic amplifier. Args: AT/C T/C clock&clamp substrate supplies
.ic V(T)='ini' V(C)='vv-ini'                            $ .ic V(a)={gg} V(a2)=ini
M0 pT AT T nsub n1                                      $ pass gate
M1 pT AC T psub p1
M2 pT AC C nsub n1                                      $ pass gate
M3 pT AT C psub p1
.if (CLAMP=1)
M4 GND AC T nsub n1                                     $ clamp
M5 GND AT C nsub n1
M6 GND C1 T nsub n1                                     $ clamp
M7 GND C1 C nsub n1
.endif
.ENDS QAAmp

* This is the latched version; it is just a QAAmp followed by a pass gate.
* Erik's "two hat" adiabatic amplifier plus pass gate. In S2LAL notation, it expects data input as A^ and -A^. Given this, it produces the correct output
* re. (a) [ZF008 Fig. 8c] followed by two pass gates or (b) [ZF008 Fig. 9a, right side] but a non-inverting buffer instead of an AND gate.
* Same role in framework as [S2LAL Fig. 5 (left)].
*  :           :           :           :           :           :           :           :           :
* 1: A(i-1)^   2: -A(i-1)^
* 3: C(i)^     4: -C(i)^
* 5: phi(i)^   6: Clmp(i-1)v
* 7: phi(j)^   8: phi(j)v
```

```
* 9: GND      10: PWR
*11: nsub     12: psub
*13: Q(i)^    14: -Q(i)^   15: tap      16: tap
.SUBCKT qLatch AT AC QT QC piT Cli pjT pjC GND PWR          $ One phase of the 2LAL shift register. Args: AT/C QT/C clockiT&clamp clockjT/C
+ nsub psub tap0 tap1 tap2 tap3 ini='gg'                    $ substrate supplies
r0 tap0 T 1                                                 $ green
r1 tap1 C 1                                                 $ red
r2 tap2 piT 1e9                                             $ blue
r3 tap3 Cli 1e9                                             $ yellow
X1 AT AC T C piT Cli GND nsub psub QAAmp ini='ini'
M1 T pjT QT nsub n1                                         $ Frank's latch
M2 T pjC QT psub p1
M3 C pjT QC nsub n1                                         $ Frank's latch
M4 C pjC QC psub p1
C1 AT 0 ACAP
C2 AC 0 ACAP
C3 T 0 QQCAP
C4 C 0 QQCAP
.ENDS qLatch

* One phase of a Q2LAL shift register [ZF008 Fig. 7b].
* Same role in framework as one loop of [S2LAL Fig. 6].
*  :           :           :           :           :           :           :           :
* 1: S0        2: -S0
* 3: S1        4: -S1
* 5: phi(0)    6: -phi(0)   7: phi(1)    8: -phi(1)   9: phi(2)   10: -phi(2)
*11: phi(3)   12: -phi(3)
*13: GND      14: PWR
*15: nsub     16: psub
*17: Q(i)^    18: -Q(i)^   19: q(i)^    20: -q(i)^   21: tap      22: tap      23: tap      24: tap
.SUBCKT qPhase S0T S0C S1T S1C                              $ One stage of the 2LAL shift register. Args: AT/C QT/C
+ p0T p0C p1T C11 p2T C12 p3T p3C GND PWR nsub psub         $ two clocks T/C and two clocks T&clamp DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 ini='gg'
r0 tap0 t0 1
r1 tap1 t1 1
r2 tap2 t2 1
r3 tap3 t3 1
X0  S0T S0C S1T S1C p1T C11 p0T p0C GND PWR nsub psub t0 t1 tap4 tap5 qLatch ini=ini
X10 S1T S1C S0T S0C p2T C12 p3T p3C GND PWR nsub psub t2 t3 tap6 tap7 qLatch ini=ini
.ends qPhase

* 8 phases of a Q2LAL shift register [ZF008 Fig. 7b].
* Same role in framework as one loop of [S2LAL Fig. 6].
*  :           :           :           :           :           :           :           :           :
* 1: S0        2: -S0
* 3: S8        4: -S8
* 5: phi(0)    6: -phi(0)   7: phi(1)    8: -phi(1)   9: phi(2)   10: -phi(2)  11: phi(3)   12: -phi(3)
*13: phi(4)   14: -phi(4)  15: phi(5)   16: -phi(5)  17: phi(6)   18: -phi(6)  19: phi(7)   20: -phi(7)
*21: Clmp(0)v 22: Clmp(1)v 23: Clmp(2)v 24: Clmp(3)v 25: Clmp(4)v 26: Clmp(5)v 27: Clmp(6)v 28: Clmp(7)
*29: GND      30: PWR      31: nsub     32: psub
*33: tap      34: tap      35: tap      36: tap
.SUBCKT qDelay SiT SiC S7T S7C                              $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p1T p2T p3T p4T p5T p6T p7T                           $ clocks/power supplies
+ C10 C11 C12 C13 C14 C15 C16 C17                          $ clamps
+ tap8 tap9 tapA tapB                                      $ debugging taps
+ tapC tapD tapE tapF                                      $ debugging taps and initialization
+ GND PWR nsub psub ini='gg'                               $ DC Supply substrate supplies
R8 tap8 t100 1
R9 tap9 t110 1
RA tapA t120 1
RB tapB t130 1
RC tapC t140 1
RD tapD t150 1
RE tapE t160 1
RF tapF t170 1
X0  S0T S0C S1T S1C p0T p4T p1T C10 p2T C11 p3T p7T GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 qPhase ini=gg
X1  S1T S1C S2T S2C p1T p5T p2T C11 p3T C12 P4T p0T GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 qPhase ini=ini
X2  S2T S2C S3T S3C p2T p6T p3T C12 P4T C13 P5T p1T GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 qPhase ini=ini
X3  S3T S3C S4T S4C p3T p7T P4T C13 P5T C14 P6T p2T GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 qPhase ini=ini
X4  S4T S4C S5T S5C P4T p0T P5T C14 P6T C15 P7T p3T GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 qPhase ini=ini
X5  S5T S5C S6T S6C P5T p1T P6T C15 P7T C16 P0T p4T GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 qPhase ini=ini
X6  S6T S6C S7T S7C P6T p2T P7T C16 P0T C17 P1T p5T GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 qPhase ini=gg
X7  SiT SiC S0T S0C P7T p3T P0T C17 P1T C10 P2T p6T GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 qPhase ini=gg
.ENDS qDelay

* Clamp waveform [ZF008 Fig. 8d]. Operates in two modes:
* Production, where the wave is generated from four clocks [ZF008 Fig. 8d].
* Testing, where the function is created from a hardcoded ngspice clock. This clock is included in the power computation.
* Choose by switch the condition between .if (1) and .if (0)
*  :           :           :           :           :           :           :           :           :
* 1: Phi(i+4)^ 2: Phi(i-2)^
* 3: Phi(i-1)^ 4: Phi(i-1)v
* 5: Test      6: Clmp(i)v
.SUBCKT Clp Pip4 Pim2 Pm1hat Pm1cup Test Clmp               $ Phi(i+4) Phi(i-2) Phi(i-1)hat Phi(i-1)cup Clamp(i)
+ nsub psub                                                $ Substrate supplies
.if (cwf!=0)
M0 Pip4  Pm1hat Clmp nsub n1                                $ pass gate
M1 Pip4  Pm1cup Clmp psub p1
M2 Pim2 Pm1cup Clmp nsub n1                                 $ pass gate
M3 Pim2 Pm1hat Clmp psub p1
* C1 Clmp 0 5p
.else
R1 Test Clmp 0 $1000                                           $ basically a direct connection
* C1 Clmp 0 10p
.endif
.ENDS Clp

* Special circuit waveform [ZF008 Fig. 10b].
*  :           :           :           :           :           :           :           :
* 1: Phi(i-1)v 2: Phi(i+1)v
* 3: Phi(i+2)^ 4: Phi(i+2)v
* 5: A(i-5)^   6: -A(i-5)^
* 7: Phi(i)^   8: J(i)^
.SUBCKT Spec Pip4 Pim2 Pm1cup Pm1hat AT AC Picup J          $ Phi(i+4) Phi(i-2) Phi(i-1)hat Phi(i-1)cup Clamp(i)
+ VDD nsub psub                                            $ Substrate supplies
M0 Pip4 Pm1cup c nsub n1                                    $ pass gate
M1 Pip4 Pm1hat c psub p1
M2 Pim2 Pm1hat c nsub n1                                    $ pass gate
M3 Pim2 Pm1cup c psub p1
M4 VDD c J psub p1                                          $ c is c(i+3)hat
M5 VDD AT J psub p1
M6 Picup AT J nsub n1
M7 Picup AC J psub p1
.ENDS Spec

* 8 phases of a Q2LAL shift register [ZF008 Fig. 7b].
* Same role in framework as one loop of [S2LAL Fig. 6].
```

```
*  :           :           :           :           :           :           :           :           :
* 1: S0        2: -S0
* 3: S8        4: -S8
* 5: phi(0)    6: -phi(0)   7: phi(1)    8: -phi(1)   9: phi(2)   10: -phi(2)  11: phi(3)   12: -phi(3)
*13: phi(4)   14: -phi(4)  15: phi(5)   16: -phi(5)  17: phi(6)   18: -phi(6)  19: phi(7)   20: -phi(7)
*21: Clmp(0)v 22: Clmp(1)v 23: Clmp(2)v 24: Clmp(3)v 25: Clmp(4)v 26: Clmp(5)v 27: Clmp(6)v 28: Clmp(7)v
*29: GND      30: PWR      31: nsub     32: psub
*33: tap      34: tap      35: tap      36: tap
.SUBCKT qDataClock SiT SiC S7T S7C                          $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p1T p2T p3T p4T p5T p6T p7T                           $ clocks/power supplies
+ C10 C11 C12 C13 C14 C15 C16 C17                           $ clamps
+ J0 J1 J2 J3 J4 J5 J6 J7                                   $ Generated clocks. These are the "hat" clocks; J(n)v = J(n + 4 mod 8)^
+ GND PWR nsub psub ini='gg'                                $ DC Supply substrate supplies

X0  S0T S0C S1T S1C p0T p4T p1T C10 p2T C11 p3T p7T GND PWR nsub psub J1   t101 t102 t103 t200 t201 t202 t203 qPhase ini=gg
X1  S1T S1C S2T S2C p1T p5T p2T C11 p3T C12 P4T p0T GND PWR nsub psub J2   t111 t112 t113 t210 t211 t212 t213 qPhase ini=ini
X2  S2T S2C S3T S3C p2T p6T p3T C12 P4T C13 P5T p1T GND PWR nsub psub J3   t121 t122 t123 t220 t221 t222 t223 qPhase ini=ini
X3  S3T S3C S4T S4C p3T p7T P4T C13 P5T C14 P6T p2T GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 qPhase ini=ini
X4  S4T S4C S5T S5C P4T p0T P5T C14 P6T C15 P7T p3T GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 qPhase ini=ini
X5  S5T S5C S6T S6C P5T p1T P6T C15 P7T C16 P0T p4T GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 qPhase ini=ini
X6  S6T S6C S7T S7C P6T p2T P7T C16 P0T C17 P1T p5T GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 qPhase ini=gg
X7  SiT SiC S0T S0C P7T p3T P0T C17 P1T C10 P2T p6T GND PWR nsub psub J0   t171 t172 t173 t270 t271 t272 t273 qPhase ini=gg
* Selectively turn on/off the data-controlled clock. Connect clocks to PWR when off to avoid an error when trying to plot a disconnected node.
.if (dcc!=0)
X8  p7T p1T p6T p2T SiT SiC p4T J4 PWR nsub psub Spec
X9  p0T p2T p7T p3T S0T S0C p5T J5 PWR nsub psub Spec
X10 p1T p3T p0T p4T S1T S1C p6T J6 PWR nsub psub Spec
X11 p2T p4T p1T p5T S2T S2C p7T J7 PWR nsub psub Spec
.else
R0 J4 PWR 1e6
R1 J5 PWR 1e6
R2 J6 PWR 1e6
R3 J7 PWR 1e6
.endif
.ENDS qDataClock

* Clock processing [ZF008 Fig. 15].
*  :           :           :           :           :           :           :           :           :
.SUBCKT Distort In Com Out                                  $ Clock drive, common (current not counted), output to circuit
.if (Imped = 0)                                             $ Set to 1 for a 2:1 voltage divider; set to 0 to add a transmission line
R1 In Out 'Imped'                                           $ resistor
.else
* Scaled transmission line. The intent of this project is to create systems that are larger than can be simulated with Spice by a factor we'll call sf. Our model for
* a single transmission line that will power sf copies of the simulated circuit is as follows: Velocity is 1/sqrt(LC), so the LC product does not change.
* Characteristic impedance is sqrt(L/C), so raise L and decrease C by sf. Increase resistances by sf. From datasheet:
* SPF-250 velocity 84% imped 50 ohms 24.2 pf/ft 79.4 pf/m .61 uH/ft .2 uH/m (center) 3 ohm/1000 ft 9.84 ohm/km (shield) 2 ohm/1000 ft 6.56 ohm/km
.MODEL ymod txl R='5e-3*sf' L='.61e-6*sf' G=0 C='24.2e-12/sf' length=1
r1 In t 'Imped'                                             $ matched impedance on drive
y1 t Com Out Com ymod LEN='Delay*1.0167e9*.84'              $ Length in feet, based on speed of light = c = 1.0167 ft/ns
.endif

.ENDS Distort

*** POWER-CLOCKS
.param gg= 0V
.param vv= 9.99V

*** CLOCKS -- Original 8 clock phases and inverses (total eight unique signals), but with slow and fast phase 1's (total 12 unique signals)
.param simlen=periods*period                                $ length of the plot in time

$ Extra delay to split phi0 into a fast and slow clock; if Fast=0, the clocks become the same
$ See Saed G. Younis. Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic. No. AI-TR-1500. MIT AI Laboratory, 1994.
.param tick=period/(8+FastSlow*2)                           $ regular: period/8; FastSlow: period/10
.param Fast=FastSlow*tick                                   $ regular: tick; FastSlow: tick

.param Ramp=(1-2*Porch)*tick                                $ waveform is parameterized so there is a "porch" on either side of a ramp
.param PPT=Porch*tick                                       $ one PPT at beginning and end of sequence, two of these PPTs between ramps

$ Parameters for three-segment ramp
.param Rx=GenT*Ramp                                         $ end time of initial gentle rise
.param Ry=(1-GenT)*Ramp                                     $ start time of final gentle rise
.param v2=GenV*vv/Gain                                      $ On rise: voltage at start of the second segment
.param v3=GenW*vv/Gain                                      $ On rise: voltage at start of the third segment
.param w2=(1-GenV)*vv/Gain                                  $ On fall: voltage at start of the second segment
.param w3=(1-GenW)*vv/Gain                                  $ On fall: voltage at start of the third segment
.param v4=vv/Gain                                           $ This is the effective Vdd level for AC signals that are transferred to the cryostat

.param Rp=Ramp                                              $ total length of ramp

.param ticks=simlen/tick                                    $ number of ticks in the simulation
.param ttn=18000ns                                          $ integration time for energy

.param tstep=25NS*period/10u*periods/20                             $ time of a simulation step, so number of steps is tick*ticks/tstep

$ The clocks comprise a series of transitions (separated by PPTs). Starting at the beginning of the three-phase cycle, the clock are computed by repeatedly
$ incrementing the time by the length of a transition and a PPT.
.param f0uS=PPT
.param f0uF=f0uS+Fast
.param f1up=f0uF+Ramp+2*PPT
.param f2up=f1up+Ramp+2*PPT
.param f3up=f2up+Ramp+2*PPT
.param f0dn=f3up+Ramp+2*PPT
.param f1dn=f0dn+Ramp+2*PPT
.param f2dF=f1dn+Ramp+2*PPT
.param f2dS=f2dF+Fast
.param f3dn=f2dS+Ramp+2*PPT
.param epoc=f3dn+Ramp+PPT

* Clamp waveforms that are high for one tick to clamp signals to ground. VCi is high on tick i-1. These are for testing only.
* Each can be generated with four transistors from existing clocks. They only connect to transistor gates, so they do not need a lot of drive capability.
Vc0    Tc0 0 DC 'vv' PWL('0' 'vv' 'f0uS' 'vv' 'f0uS+Rx' 'w2' 'f0uS+Ry' 'w3' 'f0uS+Rp' 'gg' 'f2dS' 'gg' 'f2dS+Rx' 'v2' 'f2dS+Ry' 'v3' 'f2dS+Rp' 'vv' 'epoc' 'vv' r='0')
Vd0    Tg0 0 DC 'gg'
Vc1    Tc1 0 DC 'vv' PWL('0' 'vv' 'f1up' 'vv' 'f1up+Rx' 'w2' 'f1up+Ry' 'w3' 'f1up+Rp' 'gg' 'f3dn' 'gg' 'f3dn+Rx' 'v2' 'f3dn+Ry' 'v3' 'f3dn+Rp' 'vv' 'epoc' 'vv' r='0')
Vd1    Tg1 0 DC 'gg'
Vc2    Tc2 0 DC 'gg' PWL('0' 'gg' 'f0uS' 'gg' 'f0uS+Rx' 'v2' 'f0uS+Ry' 'v3' 'f0uS+Rp' 'vv' 'f2up' 'vv' 'f2up+Rx' 'w2' 'f2up+Ry' 'w3' 'f2up+Rp' 'gg' 'epoc' 'gg' r='0')
Vd2    Tg2 0 DC 'gg'
Vc3    Tc3 0 DC 'gg' PWL('0' 'gg' 'f1up' 'gg' 'f1up+Rx' 'v2' 'f1up+Ry' 'v3' 'f1up+Rp' 'vv' 'f3up' 'vv' 'f3up+Rx' 'w2' 'f3up+Ry' 'w3' 'f3up+Rp' 'gg' 'epoc' 'gg' r='0')
Vd3    Tg3 0 DC 'gg'
Vc4    Tc4 0 DC 'gg' PWL('0' 'gg' 'f2up' 'gg' 'f2up+Rx' 'v2' 'f2up+Ry' 'v3' 'f2up+Rp' 'vv' 'f0dn' 'vv' 'f0dn+Rx' 'w2' 'f0dn+Ry' 'w3' 'f0dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vd4    Tg4 0 DC 'gg'
Vc5    Tc5 0 DC 'gg' PWL('0' 'gg' 'f3up' 'gg' 'f3up+Rx' 'v2' 'f3up+Ry' 'v3' 'f3up+Rp' 'vv' 'f1dn' 'vv' 'f1dn+Rx' 'w2' 'f1dn+Ry' 'w3' 'f1dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vd5    Tg5 0 DC 'gg'
Vc6    Tc6 0 DC 'gg' PWL('0' 'gg' 'f0dn' 'gg' 'f0dn+Rx' 'v2' 'f0dn+Ry' 'v3' 'f0dn+Rp' 'vv' 'f2dS' 'vv' 'f2dS+Rx' 'w2' 'f2dS+Ry' 'w3' 'f2dS+Rp' 'gg' 'epoc' 'gg' r='0')
Vd6    Tg6 0 DC 'gg'
Vc7    Tc7 0 DC 'gg' PWL('0' 'gg' 'f1dn' 'gg' 'f1dn+Rx' 'v2' 'f1dn+Ry' 'v3' 'f1dn+Rp' 'vv' 'f3dn' 'vv' 'f3dn+Rx' 'w2' 'f3dn+Ry' 'w3' 'f3dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vd7    Tg7 0 DC 'gg'
```

```
* These are the power clocks, including separate fast and slow clocks
Vphi0P Tx0 0 DC 'gg' PWL('0' 'gg' 'f0uS' 'gg' 'f0uS+Rx' 'v2' 'f0uS+Ry' 'v3' 'f0uS+Rp' 'v4' 'f0dn' 'v4' 'f0dn+Rx' 'w2' 'f0dn+Ry' 'w3' 'f0dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi0Q Ts0 0 DC 'gg'
Vphi0f TxA 0 DC 'gg' PWL('0' 'gg' 'f0uF' 'gg' 'f0uF+Rx' 'v2' 'f0uF+Ry' 'v3' 'f0uF+Rp' 'v4' 'f0dn' 'v4' 'f0dn+Rx' 'w2' 'f0dn+Ry' 'w3' 'f0dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi0g TsA 0 DC 'gg'
Vphi1P Tx1 0 DC 'gg' PWL('0' 'gg' 'f1up' 'gg' 'f1up+Rx' 'v2' 'f1up+Ry' 'v3' 'f1up+Rp' 'v4' 'f1dn' 'v4' 'f1dn+Rx' 'w2' 'f1dn+Ry' 'w3' 'f1dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi1Q Ts1 0 DC 'gg'
Vphi2P Tx2 0 DC 'gg' PWL('0' 'gg' 'f2up' 'gg' 'f2up+Rx' 'v2' 'f2up+Ry' 'v3' 'f2up+Rp' 'v4' 'f2dS' 'v4' 'f2dS+Rx' 'w2' 'f2dS+Ry' 'w3' 'f2dS+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi2Q Ts2 0 DC 'gg'
Vphi2f TxB 0 DC 'gg' PWL('0' 'gg' 'f2up' 'gg' 'f2up+Rx' 'v2' 'f2up+Ry' 'v3' 'f2up+Rp' 'v4' 'f2dF' 'v4' 'f2dF+Rx' 'w2' 'f2dF+Ry' 'w3' 'f2dF+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi2g TsB 0 DC 'gg'
Vphi3P Tx3 0 DC 'gg' PWL('0' 'gg' 'f3up' 'gg' 'f3up+Rx' 'v2' 'f3up+Ry' 'v3' 'f3up+Rp' 'v4' 'f3dn' 'v4' 'f3dn+Rx' 'w2' 'f3dn+Ry' 'w3' 'f3dn+Rp' 'gg' 'epoc' 'gg' r='0')
Vphi3Q Ts3 0 DC 'gg'
Vphi4f TxC 0 DC 'v4' PWL('0' 'v4' 'f0uF' 'v4' 'f0uF+Rx' 'w2' 'f0uF+Ry' 'w3' 'f0uF+Rp' 'gg' 'f0dn' 'gg' 'f0dn+Rx' 'v2' 'f0dn+Ry' 'v3' 'f0dn+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi4g TsC 0 DC 'gg'
Vphi4P Tx4 0 DC 'v4' PWL('0' 'v4' 'f0uS' 'v4' 'f0uS+Rx' 'w2' 'f0uS+Ry' 'w3' 'f0uS+Rp' 'gg' 'f0dn' 'gg' 'f0dn+Rx' 'v2' 'f0dn+Ry' 'v3' 'f0dn+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi4Q Ts4 0 DC 'gg'
Vphi5P Tx5 0 DC 'v4' PWL('0' 'v4' 'f1up' 'v4' 'f1up+Rx' 'w2' 'f1up+Ry' 'w3' 'f1up+Rp' 'gg' 'f1dn' 'gg' 'f1dn+Rx' 'v2' 'f1dn+Ry' 'v3' 'f1dn+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi5Q Ts5 0 DC 'gg'
Vphi6f TxD 0 DC 'v4' PWL('0' 'v4' 'f2up' 'v4' 'f2up+Rx' 'w2' 'f2up+Ry' 'w3' 'f2up+Rp' 'gg' 'f2dF' 'gg' 'f2dF+Rx' 'v2' 'f2dF+Ry' 'v3' 'f2dF+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi6g TsD 0 DC 'gg'
Vphi6P Tx6 0 DC 'v4' PWL('0' 'v4' 'f2up' 'v4' 'f2up+Rx' 'w2' 'f2up+Ry' 'w3' 'f2up+Rp' 'gg' 'f2dS' 'gg' 'f2dS+Rx' 'v2' 'f2dS+Ry' 'v3' 'f2dS+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi6Q Ts6 0 DC 'gg'
Vphi7P Tx7 0 DC 'v4' PWL('0' 'v4' 'f3up' 'v4' 'f3up+Rx' 'w2' 'f3up+Ry' 'w3' 'f3up+Rp' 'gg' 'f3dn' 'gg' 'f3dn+Rx' 'v2' 'f3dn+Ry' 'v3' 'f3dn+Rp' 'v4' 'epoc' 'v4' r='0')
Vphi7Q Ts7 0 DC 'gg'

VGND   200 0 DC 'gg'
VPWR   201 0 DC 'vv'

*** TOP-LEVEL CIRCUIT
* Initialization pattern gg gg vv results in 6-cycle 001 000 100 110 111 011; pattern vv gg vv results in 2-cycle 101 010
* Set the q2lal variable to 0 for a test of the quiet circuit and 1 for standard 2LAL
.if (q2lal!=0)
X0 SAT SAC SBT SBC 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 pp8 pp9 ppA ppB ppC ppD ppE ppF 200 201 200 201 qDataClock ini=gg $ flip for cycle...
X1 SBT SBC SCT SCC 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 uu8 uu9 uuA uuB uuC uuD uuE uuF 200 201 200 201 qDelay ini=gg
X5 SCT SCC SAC SAT 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 xx8 xx9 xxA xxB xxC xxD xxE xxF 200 201 200 201 qDelay ini=vv

X2 SXT SXC SYT SYC 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 qq8 qq9 qqA qqB qqC qqD qqE qqF 200 201 200 201 qDelay ini=gg
X3 SYT SYC SZT SZC 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 vv8 vv9 vvA vvB vvC vvD vvE vvF 200 201 200 201 qDelay ini=gg
X4 SZT SZC SXC SXT 110 111 112 113 114 115 116 117 710 711 712 713 714 715 716 717 ww8 ww9 wwA wwB wwC wwD wwE wwF 200 201 200 201 qDelay ini=vv

X5  114 116 117 113 720 710 nsub psub Clp                  $ VPhi4P VPhi6P VPhi7P VPhi3P C0
X6  115 117 110 114 721 711 nsub psub Clp                  $ VPhi5P VPhi7P VPhi0P VPhi4P C1
X7  116 110 111 115 722 712 nsub psub Clp                  $ VPhi6P VPhi0P VPhi1P VPhi5P C2
X8  117 111 112 116 723 713 nsub psub Clp                  $ VPhi7P VPhi1P VPhi2P VPhi6P C3
X9  110 112 113 117 724 714 nsub psub Clp                  $ VPhi0P VPhi2P VPhi3P VPhi7P C4
X10 111 113 114 110 725 715 nsub psub Clp                  $ VPhi1P VPhi3P VPhi4P VPhi0P C5
X11 112 114 115 111 726 716 nsub psub Clp                  $ VPhi2P VPhi4P VPhi5P VPhi1P C6
X12 113 115 116 112 727 717 nsub psub Clp                  $ VPhi3P VPhi5P VPhi6P VPhi2P C7

.if (ats)
* AND gate [ZF007 Fig. 9b and c] test process.
* First, create test inputs. Manually change the two q2lal registers so the initialization patterns are gg gg vv and vv gg vv. This will create period 6 and 2
* repetition patterns. These pattern will naturally creates all four binary combinations of two bits in (for example) SAT/SAC and SXT/SXC. Enable the code below and the
* "AND test code" plotting. This code below will then compute the AND and NAND function to wires aout and oout. Set FastSlow to 0 to avoid going bonkers.
M1   110 SAT t1 200 n1                                     $ 1. AND function. Two series transmission gates pass the clock when both inputs are asserted
M2   110 SAC t1 201 p1                                     $ 2.
M3    t1 SXT aout 200 n1                                   $ 3. Second transmission gate
M4    t1 SXC aout 201 p1                                   $ 4.
M5    t1 SAC aout 200 n1                                   $ 5. Internal node clamp
M6    t1 727 aout 200 n1                                   $ 6. Idle internal node clamp
M7     0 SAC aout 200 n1                                   $ 7. Output pull down
M8     0 727 aout 200 n1                                   $ 8. Idle output clamp
M9     0 SXC aout 200 n1                                   $ 9. Output pull down

M10  110 SAC oout 200 n1                                   $ 1. NAND function. Two parallel transmission gates pass the clock when both inputs are asserted
M11  110 SAT oout 201 p1                                   $ 2.
M12  110 SXC oout 200 n1                                   $ 3. Second transmission gate
M13  110 SXT oout 201 p1                                   $ 4
M14   t2 SAT oout 200 n1                                   $ 5. Output pull down
M15    0 727 oout 200 n1                                   $ 6. Idle clamp
M16   t2 SXC oout 200 n1                                   $ 7. Internal node clamp
M17   t2 SXT oout 201 p1                                   $ 8.
M18    0 SXT   t2 200 n1                                   $ 9. Output pull down
M19   t2 727 oout 200 n1                                   $ 10. Idle clamp
.endif

.else
X0 SAT SAC SBT SBC 110 111 112 113 114 115 116 117 200 201 200 201 pp4 pp5 pp6 pp7 pp8 pp9 ppA ppB ppC ppD ppE ppF SDELAY ini=gg $ flip for cycle...
X1 SBT SBC SCT SCC 110 111 112 113 114 115 116 117 200 201 200 201 uu0 uu1 uu2 uu3 uu4 uu5 uu6 uu7 uu8 uu9 uuA uuB SDELAY ini=gg
X5 SCT SCC SAT SAC 110 111 112 113 114 115 116 117 200 201 200 201 xx0 xx1 xx2 xx3 xx4 xx5 xx6 xx7 xx8 xx9 xxA xxB SDELAYv ini=vv

X2 SXT SXC SYT SYC 110 111 112 113 114 115 116 117 200 201 200 201 qq0 qq1 qq2 qq3 qq4 qq5 qq6 qq7 qq8 qq9 qqA qqB SDELAY ini=gg
X3 SYT SYC SZT SZC 110 111 112 113 114 115 116 117 200 201 200 201 vv0 vv1 vv2 vv3 vv4 vv5 vv6 vv7 vv8 vv9 vvA vvB SDELAY ini=gg
X4 SZT SZC SXT SXC 110 111 112 113 114 115 116 117 200 201 200 201 ww0 ww1 ww2 ww3 ww4 ww5 ww6 ww7 ww8 ww9 wwA wwB SDELAYv ini=vv
.endif

* These circuits model the transmission lines from the power-clock generators to the circuit
* Tx0...Tx7 are the 8 clock phases; TxA...TxD are the extended clocks; Ts* is the coorsponding shield
X20 Tx0 Ts0 110 Distort
X28 TxA TsA 510 Distort
X21 Tx1 Ts1 111 Distort
X22 Tx2 Ts2 112 Distort
X29 TxB TsB 512 Distort
X23 Tx3 Ts3 113 Distort
X30 TxC TsC 514 Distort
X24 Tx4 Ts4 114 Distort
X25 Tx5 Ts5 115 Distort
X31 TxD TsD 516 Distort
X26 Tx6 Ts6 116 Distort
X27 Tx7 Ts7 117 Distort

* Clamp signal. Signals need to be 0 to avoid a crash in S2LAL simulation
* Tc0...Tc7 are the clamp signals; Tg* is the shield (ground)
.if (q2lal!=0)
X32 Tc0 Tg0 720 Distort
X33 Tc1 Tg1 721 Distort
X34 Tc2 Tg2 722 Distort
X35 Tc3 Tg3 723 Distort
X36 Tc4 Tg4 724 Distort
X37 Tc5 Tg5 725 Distort
X38 Tc6 Tg6 726 Distort
X39 Tc7 Tg7 727 Distort
.else
R0 0 720 0
R1 0 721 0
R2 0 722 0
```

```
R3 0 723 0
R4 0 724 0
R5 0 725 0
R6 0 726 0
R7 0 727 0
.endif

* power and energy calculation
B4 0 16 V=0
+ +(I(Vc0)+I(Vd0))*v(720)+(I(Vc1)+I(Vd1))*v(721)+(I(Vc2)+I(Vd2))*v(722)+(I(Vc3)+I(Vd3))*v(723)
+ +(I(Vc4)+I(Vd4))*v(724)+(I(Vc5)+I(Vd5))*v(725)+(I(Vc6)+I(Vd6))*v(726)+(I(Vc7)+I(Vd7))*v(727)
+ +(I(vphi0P)+I(vphi0Q))*v(110)+(I(vphi1P)+I(vphi1Q))*v(111)+(I(vphi2P)+I(vphi2Q))*v(112)+(I(vphi3P)+I(vphi3Q))*v(113)
+ +(I(vphi4P)+I(vphi4Q))*v(114)+(I(vphi5P)+I(vphi5Q))*v(115)+(I(vphi6P)+I(vphi6Q))*v(116)+(I(vphi7P)+I(vphi7Q))*v(117)
+ +(I(vphi0f)+I(vphi0g))*v(510)+(I(vphi2f)+I(vphi2g))*v(512)+(I(vphi4f)+I(vphi4g))*v(514)+(I(vphi6f)+I(vphi6g))*v(116)
+ +I(VGND)*v(200)+I(VPWR)*v(201)
A1 16 17 power_tally
.model power_tally int(in_offset=0.0 gain=1.0 out_lower_limit=-1e12 out_upper_limit=1e12 limit_range=1e-9 out_ic=0.0)

.option noinit acct

*********************************************************************************************
$ NGSPICE CONTROL AREA
.TRAN 'tstep' 'ticks*tick'
.csparam slen = 'simlen*1e6'
.csparam prds = 'periods'
.csparam epch = 'epoc*1e6'
.csparam ticu = 'tick*1e6'
.csparam ntks = 'ticks'
.csparam tste = 'tstep*1e9'
.csparam fstp = 'Fast*1e6'
.csparam rmpu = Ramp*1e6
.csparam imp = 'Imped'
.csparam isc = 'sf'
.csparam igv = 'GenV'
.csparam igf = 'Delay*1.0167e9*.84'
.control
pre_set strict_errorhandling
unset ngdebug
echo "********************Sim: $&slen us=$&prds*$&epch us. Tick: $&ticu us=$&ntks*$&tste ns. Rmp: $&rmpu us. Fast $&fstp us."
run

* measure power consumption
meas tran Energylus INTEG v(16) from=0 to=5us
meas tran EnergyLev INTEG v(16) 'from=5us to=ttn'
echo --------------------Results $&Energylus , $&EnergyLev
echo Results , $&Energylus , $&EnergyLev >>Q2LAL.csv

* white background
set color0=white
* black grid and text (only needed with X11, automatic with MS Win)
set color1=black
* wider grid and plot lines
set xbrushwidth=1
set xgridwidth=1

set hcopypscolor=1
set hcopyscale=4
set hcopypstxcolor=2
set hcopyfontsize=3
set gnuplot_terminal=png

$ plot                                                  $ plot clock current
gnuplot gp/clkcur
+ title "Clock current. Sim: $&slen us=$&prds*$&epch us. Tick: $&ticu us=$&ntks*$&tste ns. Rmp: $&rmpu us. Fast $&fstp us."
$ + ylimit -5m 5m                                       $ BOOKMARK1
+ ylimit -200u 200u                                    $ BOOKMARK1
+ I(Vphi0P)+I(Vphi0Q) I(Vphi0f)+I(Vphi0g) I(Vphi1P)+I(Vphi1Q) I(Vphi2P)+I(Vphi2Q) I(Vphi2f)+I(Vphi2g) I(Vphi3P)+I(Vphi3Q)
+ I(Vphi4f)+I(Vphi4g) I(Vphi4P)+I(Vphi4Q) I(Vphi5P)+I(Vphi5Q) I(Vphi6f)+I(Vphi6g) I(Vphi6P)+I(Vphi6Q) I(Vphi7P)+I(Vphi7Q)

plot                                                   $ plot instantaneous energy consumption
$ gnuplot gp/power
+ title "Dissipation. Sim: $&slen us=$&prds*$&epch us. Tick: $&ticu us=$&ntks*$&tste ns. Rmp: $&rmpu us. Fast $&fstp us."
+ ylimit -25m 25m
+ v(16)

$ plot                                                  $ plot accumulated energy dissipation
gnuplot gp/energy
+ title "Cum. dissipation. Sim: $&slen us=$&prds*$&epch us. Tick: $&ticu us=$&ntks*$&tste ns. Rmp: $&rmpu us. Fast $&fstp us."
+ ylimit 0 70n
+ v(17)

$ plot                                                  $ plot predistorted waveform plus waveform at chip
gnuplot gp/line
+ title "Predistortion. Impedance $&imp sf $&isc gv $&igv $&igf feet $&slen us=$&prds*$&epch us. Tick: $&ticu us=$&ntks*$&tste ns."
+ ylimit -5 45
+ v(Tx0)+11 v(110)+11
+ v(Tx1)    v(111)
+ v(Tx2)+22 v(112)+22
+ v(Tx3)+33 v(113)+33

plot ylimit 0 7 xlimit 0 50u
$ gnuplot gp/traces ylimit 0 7 xlimit 0 200u
+ title "3-stage Q2LAL/S2LAL inverting shift register"
+ v(ppF)/49.99*0.9+ 4.55+.000*10
+ v(ppE)/49.99*0.9+ 4.55+.025*10
+ v(ppD)/49.99*0.9+ 4.55+.050*10
+ v(ppC)/49.99*0.9+ 4.55+.075*10
+ v(ppB)/49.99*0.9+ 4.55+.100*10
+ v(ppA)/49.99*0.9+ 4.55+.125*10
+ v(pp9)/49.99*0.9+ 4.55+.150*10
+ v(pp8)/49.99*0.9+ 4.55+.175*10
+
* These lines create a non-controlled set of waveforms to make [ZF008 Fig. 10a] more understandable BOOKMARK2
* + v(117)/49.99*0.9+ 1.55+.000*10
* + v(116)/49.99*0.9+ 1.55+.025*10
* + v(115)/49.99*0.9+ 1.55+.050*10
* + v(114)/49.99*0.9+ 1.55+.075*10
* + v(113)/49.99*0.9+ 1.55+.100*10
* + v(112)/49.99*0.9+ 1.55+.125*10
* + v(111)/49.99*0.9+ 1.55+.150*10
* + v(110)/49.99*0.9+ 1.55+.175*10
* end BOOKMARK2
+
* AND test code
* + v(oout)/49.99*0.9+2.55+.175*10                       $ NAND output, allows AND and NAND to be a two-rail signal (green)
* + v(aout)/49.99*0.9+2.55+.150*10                       $ AND output (red)
* + v(727)/49.99*0.9+2.55+.125*10                        $ clamp c(i-1)v in [ZF008 Fig. 9b and c] (blue)
```

```
* + v(117)/49.99*0.9+2.55+.100*10                          $ clock for the next phase, phi(i)^ in [ZF008 Fig. 9b and c] (yellow)
* + v(SXC)/49.99*0.9+2.55+.075*10                          $ B input complementary value, asserted when B is 0 (magenta)
* + v(SAC)/49.99*0.9+2.55+.050*10                          $ A input complementary value, asserted when A is 0 (turquoise)
* + v(SXT)/49.99*0.9+2.55+.025*10                          $ B input (orange)
* + v(SAT)/49.99*0.9+2.55+.000*10                          $ A input (brown)
+
+ v(uu8)/9.99*0.9+2.55
+
* These lines are the source of [ZF008 Fig. 12b and d]
+ v(SAT)/9.99*0.9+ 0.55
+ v(SAC)/9.99*0.9+ 0.55+.05

.endc

.END
* Notes:
* Q2LAL is a significant conceptual modification to S2LAL, albeit one that differs only in one transistor.
* Q2LAL transmits bits in straightforward dual-rail, which means a 1 is a pulse from 0 V to Vdd. Using S2LAL terminology, this is a "hat" pulse, meaning it has
* the most positive voltage in the middle. A Q2LAL 0 is a "hat" pulse on a second wire. In contrast, S2LAL sends a 1 on two wires, a hat pulse like Q2LAL but also
* an electrically inverted pulse on a different wire, i. e. a pulse from the idle Vdd state to 0 V. S2LAL sends a 0 by leaving both wires in the idle state.
*
* Tested with ngspice-30 (creation date Dec 28, 2018, from ngspice-30_64.zip 8,687,648 bytes)
*
* For tutorial docs: no tabs; comments start column 61; 169 character maximum line length
*
* Notation: A postive pulse A is designated in print with a circumflex (^) diacritical mark. It may be designated here as "A-hat" or "A^"; a negative pulse is
* designated in print with a caron (v) diacritical mark. It may be designated here as "A-cup" or Av. In this notation, -A^ does not mean -(A^) = Av but rather (-A)^,
* a positive-going pulse when A is 0
*
* References:
* [ZF008] DeBenedictis, Erik. "Energy Management with Adiabatic Circuits." Technical report ZF008 (publication pending)
* [ZF007] http://zettaflops.org/CATC/DPA-Q2LAL.pdf December 19, 2020 Document ZF007
* [Q2LALv2.ppt] Slide deck DPA-Q2LALv2.ppt January 2, 2021 Document ZF007, a non-public PowerPoint on Erik's computer
* [S2LAL] Frank, Michael P., et al. "Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS." arXiv preprint arXiv:2009.00448 (2020).
* [Athas] Athas, W. C., et al. "Low-power digital systems based on adiabatic-switching principles." IEEE Transactions on VLSI Systems 2.4 (1994): 398-407
```