Asynchronous Ballistic Reversible Computing

Michael P. Frank
Center for Computing Research
Sandia National Laboratories
Albuquerque, New Mexico USA
mpfrank@sandia.gov

***Abstract*— Most existing concepts for hardware implementation of reversible computing invoke an *adiabatic* computing paradigm, in which individual degrees of freedom (*e.g.*, node voltages) are synchronously transformed under the influence of externally-supplied driving signals. But distributing these "power/clock" signals to all gates within a design while efficiently recovering their energy is difficult. Can we reduce clocking overhead using a *ballistic* approach, wherein data signals self-propagating between devices drive most state transitions? Traditional concepts of ballistic computing, such as the classic Billiard-Ball Model, typically rely on a precise synchronization of interacting signals, which can fail due to exponential amplification of timing differences when signals interact. In this paper, we develop a general model of *Asynchronous* Ballistic Reversible Computing (ABRC) that aims to address these problems by eliminating the requirement for precise synchronization between signals. Asynchronous reversible devices in this model are isomorphic to a restricted set of Mealy finite-state machines. We explore ABRC devices having up to 3 bidirectional I/O terminals and up to 2 internal states, identifying a simple pair of such devices that comprises a computationally universal set of primitives. We also briefly discuss how ABRC might be implemented using single flux quanta in superconducting circuits.**

Keywords—Thermodynamics of computation, energy-efficient computing, reversible computing, ballistic transport, asynchronous computing, clockless logic, superconducting circuits.

I. Introduction

The overall economic impact of computing technology has increased by many orders of magnitude over the last 70 years. For this trend to continue, the energy efficiency of computation must continue increasing commensurately, so more computation can be carried out within any given energy budget.

How energy-efficient can computers possibly become? We have known since the seminal work of Landauer in 1961 [1] that there is an absolute thermodynamic upper limit to the energy efficiency of conventional *irreversible* computing processes, ones that continually discard unwanted information in the course of their operation. In brief, this limit arises because a loss of information implies increased uncertainty regarding the detailed physical state of the system—*i.e.*, increased *entropy*—which is exactly the difference between useful energy and waste heat. The less information we have about the detailed structure of any given piece of energy in a system at any given temperature, the less useful that energy becomes. We would have to move the entropy to an even lower-temperature heat sink to extract additional useful work out of its associated energy. But in practice, the temperature of terrestrial heat sinks is no less than roughly on the order of room temperature (300 K).

Whatever temperature T the coolest available heat sink has, Landauer's limit implies that, for each bit's worth of information discarded in a computer, at least an amount kT ln 2 of useful energy, where $k \approx 1.38 \times 10^{-23}$ J/K is Boltzmann's constant, must end up degraded to the form of waste heat at temperature T. In conventional computer designs, 1 bit's worth of computational information is discarded (destructively overwritten) by every active logic gate on each clock cycle, so that, given a clock frequency of (say) 3.5 GHz, even operating at Landauer's limit, we could operate at most 100 billion active conventional logic gates per Watt of system power consumption.

Moreover, in practice, our devices today, and even at the far reaches of the semiconductor roadmap (circa 2030), are far less energy-efficient than this, since the total information they erase on each clock cycle consists of not just the *logical* bits being erased, but rather, the much *larger* number of associated *physical* bits (*e.g.*, the occupancy numbers, 0 or 1, of numerous distinct electron states between logic high and low Fermi levels) that redundantly encode each digital bit. The *International Technology Roadmap for Semiconductors* [2] projects that typical logic gate operations in the year 2030 will dissipate 0.25 fJ > 60,000 kT of energy (Fig. 1), implying that >10 KB worth of such raw *physical* information must be lost (ejected as randomized entropy) per gate operation. At this level of inefficiency, a billion-gate, 3.5 GHz processor chip would use about 875 W of power. Conventional chips in 2030 (let alone today) can only require much less power than this by actively cycling fewer logic gates. We can build billion-gate chips today, but we can't cool single chips at kW power levels. Thus, most of the potential processing capacity of modern chips ends up as wasted *dark silicon* [3], sitting idle most of the time.

But, if we can reduce the physical information lost (and energy dissipated) in each computational operation to ever-smaller amounts, asymptotically approaching zero, then the energy efficiency and power-limited performance of computing technology can grow indefinitely large, and this can potentially enable correspondingly enormous economic benefits.

This idea motivates the study of *reversible computing*, which aims to minimize the information lost and energy dissipated by computational processes. We have known since Bennett [4] that the logical concept of computation does not inherently require information loss—*i.e.*, we can always restructure a computation to avoid discarding digital information. For any intermediate result that is no longer needed, we can always *decompute* it

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. Approved for public release SAND2017-8730 C

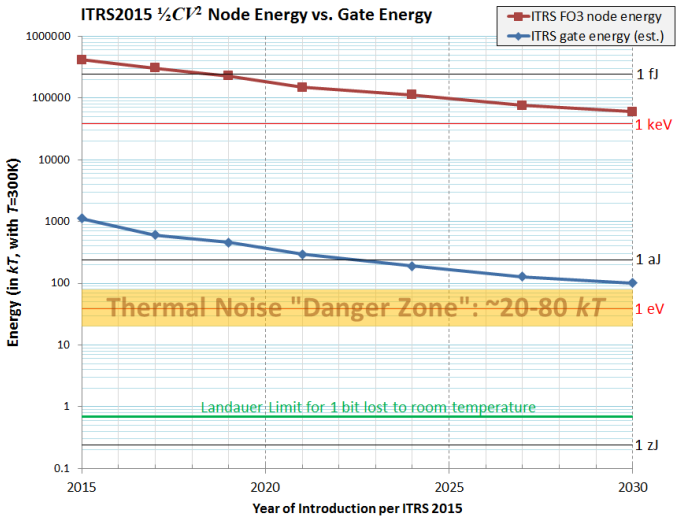

Fig. 1. Targets from the 2015 ITRS roadmap [2] for typical logic node energies (red curve, top) and minimum-size transistor gate energies (blue curve, center), the latter estimated from figures for device width, gate capacitance C_{gate} per unit width, and logic swing voltage V_{dd}. Note that although the total $\frac{1}{2}CV^2$ energy per logic node is still ~60,000 kT (in room-temperature environments) at roadmap's end in 2030, the $\frac{1}{2}CV^2$ energy of a minimum-width segment of a transistor's gate terminal is estimated to reach the 100 kT level, below which thermal noise begins to become a limiting factor on scaling. Beyond this point, significantly improving energy efficiency will require recovering and reusing increasingly-large fractions of the logic signal energy, but, achieving *thorough* energy recovery is only possible in a reversible computing paradigm.

instead of discarding it, via a time-reversed version of a process that could have computed it in the first place. If we ensure that all the *physical* information that redundantly encodes our digital bits is similarly transformed using a nearly thermodynamically-reversible *physical* process, then we can approach (as closely as desired, perhaps) the ideal of *adiabatic* operation (from Greek ἀδιάβατος, "impassable")—*i.e.*, operating in a way that holds back the useful energy invested in the system from passing out of the computing mechanism, as dissipated heat.

A variety of concepts for the asymptotically adiabatic implementation of reversible computing have been explored since the late 1970s. Device technologies that have been considered for this purpose so far include superconducting Josephson junctions [5], nanomechanical interlocks [6][7], MOS transistors [8], and quantum dots [9], among others.

However, those devices require the adiabatic transformations of their digital states to be controlled by externally-supplied driving signals. This poses significant challenges for the design of practical systems, since to simply distribute these driving "power-clock" signals throughout the system adds significant overhead (in terms of circuit complexity) to the design. Also, efficiently recovering and reusing almost all the energy contained in these large, widely-distributed signals itself poses a difficult engineering problem. We require a *resonant* clock distribution circuit with a high *quality factor* (Q), careful load balancing, and custom wave shapes (typically trapezoidal) to ensure adiabatic device operation [10]. Concepts for resonantly driving adiabatic logic designs have not yet been developed to the point of being practical enough to motivate large-scale adoption of adiabatic technology for computing. Another problem is that the currently most well-developed technology base for adiabatic logic (namely CMOS) is slower than we wish when we operate it in the adiabatic regime.

Can we find a methodology for reversible computing that avoids the overheads associated with power-clock distribution, while also facilitating fast operation? An alternative to the usual externally-controlled adiabatic approach would be a *ballistic* computing scheme, in which information is conveyed by spatially- and temporally-localized signals (*e.g.*, pulses, particles, or quasiparticles) that propagate *ballistically* (that is, freely, under their own inertia) through the system, carrying with them a signal energy that is reused repeatedly to carry out multiple transformations of the logical state of the machine.

The first specific ballistic computing concept was the *Billiard Ball Model* of Fredkin and Toffoli [11], which envisioned idealized, perfectly-elastic spheres traveling on precise trajectories; these could perform reversible logic operations via their collisions. But this model was unrealistic in the sense that, even if we obtained perfectly elastic "balls" (*e.g.*, individual subatomic particles), it also required them to travel along perfectly-precise trajectories. But realistically, the balls' positions and timing could not be prepared to infinite precision, due to the uncertainty principle. Worse, any initial uncertainties, however small, would tend to become exponentially amplified over the course of subsequent collisions.

Can we devise a *new* concept for ballistic computing that avoids the chaotic instabilities of the Billard-Ball Model? The attainment of stability in positioning is facilitated if the "balls" (or more generally, localized, ballistically-propagating signals) are confined to traveling along constrained pathways (*e.g.*, potential energy troughs, or high-quality transmission lines, in the case of electrical pulses), and the need for temporal precision can be avoided if the devices are *asynchronous* by design, that is, if we arrange, by construction, that their functional behavior is not significantly affected by the exact time of arrival of different input pulses, relative to each other.

In this paper, we start systematically exploring, at an abstract level, what the resulting concept of Asynchronous Ballistic Reversible Computing (ABRC) must look like. The requirements of asynchrony, logical reversibility, and ballistic propagation of localized, digitally-classifiable signals along wires, along with a few others, fully determine the overall structure of the theoretical model. Both classical and quantum variants of the model can be described, although in this paper, we restrict our attention to exploring the classical version of the model. We prove that the model is computation-universal, and we briefly discuss a potential implementation strategy using single flux quanta in superconducting circuits. We conclude with an outline of some directions for future work.

II. Basic Properties of the Model

To clarify, the initial set of requirements that will guide our development of the ABRC model will be as follows:

1) Universality: The model should be capable of universal computation (including reversible, and embedded irreversible).

2) Network model: The hardware model (Fig. 2a) consists of a fixed (albeit arbitrarily large) number of discrete, finite-sized primitive elements called *devices*, each with at most some small constant number of *terminals* which convey information

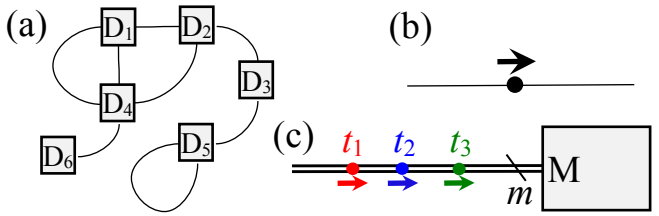

Fig. 2. (a) A network of 6 devices D_1, D_2, …, D_6, interconnected by wires. Devices D_1, D_2, D_5 each have 3 terminals; D_3 has 2 terminals; D_4, 4 terminals; and D_6, 1 terminal. (b) Graphic notation representing a spatially- and temporally-localized pulse on a wire; the pulse is propagating along to the right. (c) This device terminal labeled M, and the wire to which it is connected, have *multiplicity* (arity) $m = 3$, meaning they can handle pulses of 3 different types, here labeled t_1, t_2, t_3, and distinguished by color. However, this paper focuses primarily on the case of unary pulses, *i.e.*, $m = 1$, with unlabeled pulse type.

into and out of devices; information flows along one-dimensional signal paths ("*wires*") interconnecting device terminals. We generally assume that any given terminal is bidirectional—*i.e.*, may convey information both in and out of the device. Similarly, wires are assumed to be able to convey information in either direction along their length. Wires may neither branch nor merge, and there is a one-to-one correspondence between wire ends and device terminals.

3) Localization: Information is conveyed between device terminals via localized signals (Fig. 2b), which we will generically refer to as *pulses*, regardless of whether they are signal pulses, discrete objects, quantum wavepackets for individual particles or quasiparticles, or soliton-like disturbances.

a) Spatial confinement: When traveling between device terminals, pulses are spatially restricted in two dimensions by being confined to propagate along the wires connecting the terminals; we will generically still call these paths "wires" regardless of whether they are composed of condensed material or are hollow conduits, or have some more complex structure, such as in a coaxial transmission line. We also assume that pulses are spatially localized in the third dimension (along the wire length) in the sense that the pulse width is limited by the system specifications. We assume that distinct pulses traveling on the same wire (or neighboring wires) do not overlap, in terms of their region of influence, and do not affect each other at all.

b) Temporal localization: Pulses are assumed to be temporally localized, that is, with a time-distribution of pulse energy at any given location that is characterized by a time-limited function that decays exponentially outside of some relatively narrow timespan. There may be some dispersion that lengthens pulse duration as the pulse propagates, but it should be gradual.

4) Ballistic propagation: Pulses are assumed to propagate along wires in an approximately ballistic fashion, coasting forwards in one direction while substantially conserving their own (generalized) kinetic energy and inertia in a localized form. There may be some dissipation of pulse energy and momentum as the pulse travels, but it should be gradual.

5) Digital interpretation: We assume that pulses traveling on any given wire (in either direction) can be classified into a finite number $m \geq 1$ of distinguishable categories (Fig. 2c). We call m the *multiplicity* or *arity* of the pulses on that wire, and we distinguish the cases $m = 1$ (which we refer to as *unary*, *single-valued*, *unipolar*) and $m > 1$ (*m-ary*, *multivalued*, *polarized*). For example, in a superconducting implementation based on quantized current pulses, discussed further in Sec. V, there would naturally be $m = 2$ distinct pulse polarizations, depending on the sign of the current. Each device terminal also has an associated multiplicity, which must match that of the wire to which it is connected. In this paper, we focus on networks in which all pulses are unary, but this turns out not to reduce the asymptotic power of the model, compared to the case with a constant maximum $m > 1$, since an m-ary wire can always be represented, with constant overhead, by m parallel unary wires.

6) Asynchrony: The functional behavior of devices should not depend on the exact absolute or relative time of arrival of pulses arriving at the device, but (at most) on their relative order of arrival. If we want the model to exhibit time-reversal symmetry, this requirement should also hold when considering device behavior in the reverse time direction as well; we can call this property *reverse asynchrony*.

7) Determinism: The logical behavior of devices should be *deterministic* (non-random). This particular constraint could be relaxed in future stochastic or quantum versions of the model.

8) Reversibility: The functional behavior of devices should be both logically reversible, and almost physically reversible; *i.e.*, the model should impose no fixed lower bound greater than zero on the entropy increase resulting from device operations. Thus, it must obey the principles of Generalized Reversible Computing (GRC), as described in [12].

9) Quiescence: In between subsequent pulse arrivals, devices should remain *quiescent* (that is, in a stationary state, with a negligible rate of background energy leakage and entropy increase). Any internal state information retained by the device must be preserved in a substantially nonvolatile way.

Starting from the above fundamental requirements, we can infer additional properties emergently required of our model:

10) Non-overlap of pulse arrivals. Due to the assumptions of asynchrony and determinism, we require that pulses arriving at a device must not overlap each other, in terms of their time of arrival, since otherwise their relative order of arrival might not be reliably determinable. We can state this requirement as a minimum relative time of arrival $\Delta t > \Delta t_{min}$ between pulse centers that is large enough to ensure that pulses are non-overlapping, *i.e.*, with some positive "gap" $g > 0$ between their temporal extents, given a specified maximum pulse duration that can be tolerated in a given design (Fig. 3a). Also, if pulses do not overlap, and devices are quiescent between pulses, then pulses do not interact with each other directly in ways that are affected by their relative time of arrival, and therefore, uncertainties in the relative time of arrival of pulses cannot be directly amplified, which reduces entropy increase, and improves the physical reversibility of the devices.

11) One-to-one correspondence between input and output pulses. Pulses necessarily carry with them a certain amount of physical entropy, due to inevitable uncertainties in their time of arrival and other detailed physical properties (*e.g.*, pulse width, energy, exact shape). This entropy cannot disappear, and cannot build up indefinitely within the device, so it must be carried out of the device shortly after each pulse arrives. How can be it carried out? Since the devices are required to be reversible, we can't allow the entropy to be emitted thermally; so, it can only be carried out in other, similar pulses—and there must be ex-

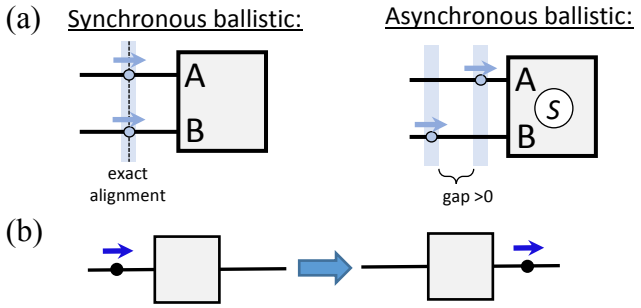

Fig. 3. (a) Left: In a synchronous ballistic device, an unphysical exact synchronization between the times of arrival of different inputs would be required; in reality, each pulse interaction would chaotically amplify timing uncertainties. Right: To ensure deterministic behavior, and avoid amplifying timing uncertainties, the asynchronous model requires that pulses arrive at a device in distinct, non-overlapping time periods, separated by a gap that is sufficient to ensure that there is no direct interaction between pulses. To enable indirect interaction between pulses to do computation, devices must have a variable internal state *S*. (b) To conserve the energy and timing information present in pulses coming into a device, while respecting reverse asynchrony and the finite nature of the devices, each incoming pulse must result in exactly one outgoing pulse that emerges from the device "immediately" (*i.e.*, after some limited delay).

actly one such pulse (Fig. 3b), since two or more pulses emerging simultaneously (*i.e.*, in an overlapping way) would violate the assumption of determinism, since the relative order of pulse emergence would be ambiguous, and for multiple pulses to emerge with a delay between them would violate the assumption of device quiescence between pulse arrivals (since the device would have to be counting down time internally until emission of the second pulse). Also, looking in the reverse time direction, multiple output pulses would violate the asynchrony assumption, since the reverse behavior of the device would depend on the relative timing of the output pulses.

12) Statefulness. In order to perform universal computation, there must be some way that different pieces of information (different pulses) can interact. Since pulses arrive at devices one at a time, with a gap between them, and since devices are quiescent in between pulse arrivals, the only way that there can be an interaction between pulses is for devices to carry internal state information (*S* in Fig. 3a, right), which (by quiescence) must be maintained stably in between pulse arrivals. Since each device is finite, it can only have some finite number *K* of reliably distinguishable internal states.

13) The set of possible ABRC devices is isomorphic to a restricted set of finite-state Mealy machines. To explain:

A finite-state Mealy machine can be thought of as a discrete device with a finite set $S = \{s_1, \ldots, s_K\}$ of internal states that takes as input an unbounded-length stream of symbols σ taken from some *input alphabet* Σ, which is a finite set of distinct symbols, and produces as output a corresponding stream of symbols taken from some *output alphabet*, which, for our present purposes, we will assume is just the same set Σ. We assume that there is exactly one output symbol produced for each input symbol that is consumed. The function of the Mealy machine is then described by a *transition function f*: $\Sigma \times S \rightarrow S \times \Sigma$ which maps any given ordered pair (σ_I, s_I) of an input symbol $\sigma_I \in \Sigma$ and an initial state $s_I \in S$ (which together we may call the *input syndrome*) to a resulting pair (s_F, σ_O) of a final state $s_F \in S$ and an output symbol $\sigma_O \in \Sigma$ (together, the *output syndrome*). Mealy machines are convenient, familiar abstractions for defining the functional behavior of standard synchronous digital circuits. The correspondence, now, between that conventional state-machine framework and our ABRC model goes as follows:

Let a given ABRC device D have *n* terminals, which we label $T_1, \ldots, T_n$. Each terminal T_i is connected to a wire that supports propagation of pulses with some multiplicity m_i. The total number of computationally-distinct cases for the *next incoming pulse* to device *D*, and similarly for the *next outgoing pulse from* device *D*, is therefore $N = m_1 + m_2 + \ldots + m_n$, so we can consider *D* to be operating on a stream of I/O "symbols" σ chosen from an alphabet of *N* distinct *signal characters* $\Sigma = \{c_i^j\}$, where $1 \le i \le n$ and $1 \le j \le m_i$; each signal character consists of an identification of the specific I/O terminal T_i that the pulse is entering/leaving, and the specific type of pulse $t_j \in \{t_1, \ldots, t_m\}$ that is arriving/departing, where $m = m_i$:

$$c_i^j = \begin{pmatrix} t_j \\ T_i \end{pmatrix}, \tag{1}$$

where the stacked variables here denote a *compound signal character* meaning "a pulse of type t_j traveling over the wire connected to terminal T_i". So, for example, a device with $n = 3$ terminals A, B, C, each having $m = 2$ pulse types labeled $\{\mathbf{0}, \mathbf{1}\}$, would operate on the $N = n \cdot m = 6$-symbol alphabet

$$\Sigma = \left\{ \begin{pmatrix} \mathbf{0} \\ A \end{pmatrix}, \begin{pmatrix} \mathbf{1} \\ A \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ B \end{pmatrix}, \begin{pmatrix} \mathbf{1} \\ B \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ C \end{pmatrix}, \begin{pmatrix} \mathbf{1} \\ C \end{pmatrix} \right\}, \tag{2}$$

where each of these compound characters designates which of the two types of pulse is arriving (or leaving) next, and on which of the three I/O terminals.

The only difference, then, between an ABRC device, and a standard Mealy machine whose input and output alphabets are identical, is that an ABRC device must be logically reversible; meaning that the transition function *f* must be an injective (one-to-one) function over its entire domain $\Sigma \times S$; or, using the theory of Generalized Reversible Computing (GRC) [12] it is actually a sufficient condition for reversibility if *f* is only injective over some *assumed* subset $a \subseteq \Sigma \times S$ of its domain, if we arrange, in the context of a given overall system design, that the precondition that the initial syndrome $(\sigma_I, s_I) \in a$ is always satisfied for that device D. In such a case, we can say that D is *conditionally* reversible, and more specifically that it implements a certain *conditioned* reversible transition function f_a whose assumed precondition for reversibility, *a*, is satisfied.

The general picture, then, is that the set of all possible ABRC devices is isomorphic to the set of Mealy machines with identical input and output alphabets and conditioned reversible transition functions. The functional behavior of any such device is to reversibly transform indefinitely-long streams of non-overlapping incoming pulses, each of which has one of *N* different characters (specifying its terminal and type), one-to-one (for cases where the device's precondition for reversibility is satisfied at each step) into streams of non-overlapping outgoing pulses, each of which also has one of *N* different characters.

A complete ABRC system design then simply consists of an assemblage of such devices, wired together into a network. We can inject input pulses into the network, one at a time, at some

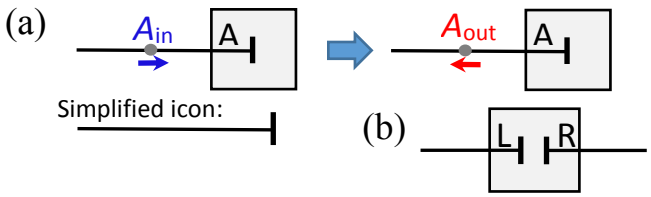

Fig. 4. (a) The only one-terminal unary device is the *pulse reflector (PR)*. Here, color is being used to distinguish incoming and outgoing unary pulses. (b) The *barrier (B)* is a trivial two-terminal device that is simply an assemblage of two reflectors, one connected to each terminal.

entry terminal(s); the pulses then shuttle through the network, updating its state as they go, and eventually emerge at some exit terminal(s)—the network's reversibility and finiteness prevents pulses from getting trapped. Note that multiple pulses can be in-flight in the network at the same time, as long as the wire delays are adjusted as needed to avoid race conditions—possible overlapping pulse arrivals at a device.

Of course, real ABRC devices would not be perfectly reversible, and pulse transport on real wires would not be perfectly ballistic, and so pulses would tend to increase in width and to decrease in energy as they pass through the network. So, a practical system design would alternate blocks of ABRC logic with regularization stages, which would re-synchronize signal timings, and restore pulse energy and sharpness. However, this does not prevent ABRC from still obtaining a substantial energy efficiency boost compared to conventional logic, and reduced clocking overhead compared to staged adiabatic logic. As the reversibility of the device technology and the ballistic character of the pulse conduits are improved, the frequency of signal regularization can be reduced, and the overall advantages of this design methodology will increase.

For practicality, we would like to be able to construct ABRC networks from a finite variety of simple *primitive* devices, with few terminals and states, operating on signals of low arities; such simple devices could potentially admit simple technological implementations. If we can identify a computationally universal set of such primitives, we could then implement arbitrary computational structures, such as assemblages of useful processing elements, including general-purpose CPUs.

However, the reader may wonder if regular networks of finite ABRC primitives, as described above, are computation-universal, given ABRC's restriction to reversible transition functions. In the next two sections, we show that they are. We begin by exploring some simple ABRC primitives, and then show that a certain pair of primitives comprises a universal set for Boolean logic, and thus can be used to construct arbitrary digital systems, including arrays of general-purpose processors.

III. Primitive Devices

In this section, we explore the possible ABRC device types with up to two internal states, up to three I/O terminals, and unary pulse arities. With all pulses unary, the I/O symbols σ_i reduce to terminal labels T_i. We will ignore device-type differences that do not affect behavior (such as internal states that are ignored, or state relabelings). For simplicity, we also restrict our attention here to devices whose transition functions are unconditionally reversible, *i.e.*, injective over all input syndromes.

TABLE I. Two-State, Two-Terminal Unary ABRC Devices

Input Syndrome	Output Syndrome (by device type)				
	FD	***AFD***	***TB***	***DFD***	***FC***
$L(S_0)$ [a]	$(S_1)R$	$(S_0)R$	$(S_1)L$	$(S_1)R$	$(S_1)L$
$L(S_1)$	$(S_1)L$	$(S_0)L$	$(S_0)R$	$(S_0)L$	$(S_1)R$
$R(S_0)$	$(S_0)R$	$(S_1)R$	$(S_1)R$	$(S_0)R$	$(S_0)R$
$R(S_1)$	$(S_0)L$	$(S_1)L$	$(S_0)L$	$(S_1)L$	$(S_0)L$

[a] In this table, $\sigma(s)$ denotes an input syndrome (σ, s), and an output syndrome (s, σ) is written $(s)\sigma$.

A. One-Terminal Primitives

The only one-terminal unary ABRC device is a *pulse reflector (PR)* (Fig. 4a). A pulse entering its single terminal *A* immediately re-emerges. Since there is only one place that the pulse can go anyway, any internal state that the device may or may not have has no effect, and can be ignored.

B. Two-Terminal Primitives

We will separately consider the "stateless" (one-state) two-terminal primitives, and the two-state two-terminal primitives that use their state in some nontrivial way.

1) One-state, two-terminal primitives.

a) Wire (W). This trivial "device" is really just functionally identical to a section of wire. Any pulse that enters one terminal immediately emerges from the other terminal. This element finds schematic use as a symbol renamer.

b) Barrier (B). This device can be implemented internally as two reflectors, one attached to each terminal (Fig. 4b). Any input pulse immediately reflects right back out of the same terminal that it went into.

2) Two-state, two-terminal primitives.

We can classify these devices according to the set of basic symmetries that they respect, which can be denoted as follows:

- **T** – *Time-reversal symmetry.* The transition function from input to output syndromes is self-inverse; $f = f^{-1}$ (ignoring pair order). The device behaves identically in both time directions with no other changes.
- **D** – *Data-terminal reversal symmetry.* The transition function stays the same if the I/O terminals are exchanged.
- **TS** – *Time/state reversal symmetry.* The inverse of the transition function is the same as the forward function if the roles of the two states are also exchanged.

The nontrivial 2-state, 2-terminal devices (Table I) can be categorized into the following 3 symmetry-based classes:

a) Devices with both **T** *and* **D** *symmetries.* It turns out that the only such device is the *flipping diode (FD)*. It conducts pulses passing in its (state-dependent) forward direction, and reflects pulses coming from the other direction, It reverses its directional state when a pulse passes through it, but not when a pulse bounces off of it. It may be used as a memory cell.

b) Devices with both **D** *and* **TS** *symmetries.* There are only two of these devices: The *anti-flipping diode (AFD)*, and the *toggling barrier (TB)*. (Discussion omitted to save space.)

c) Other devices. There are just two: The *directional flipping diode (DFD)* and the *flipping comparator (FC)*; they are each other's time-reversals. (Discussion omitted to save space.)

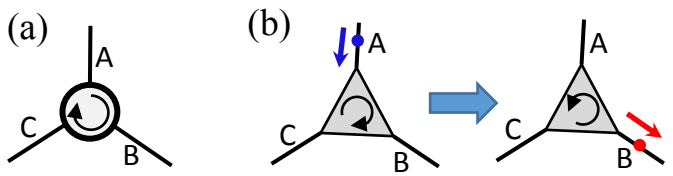

Fig. 5. (a) The only nontrivial stateless three-terminal unary device is the *rotary (R)*, which routes pulses circularly between terminals in a fixed rotational direction (clockwise or counter-clockwise). (b) The only nontrivial two-state, three-terminal, **T**- and **D3**-symmetric unary device is the *flipping rotary (FR)*, which reverses its orientation each time a pulse passes through it.

The above categorization theorem was proven by hand via detailed inspection of all 4! = 24 of the possible bijective transformations (permutations) of the 4 different (T_i, s_k) syndromes.

C. Three-Terminal Primitives

1) One-state, three-terminal primitives.

There is only one type of nontrivial, stateless, three-terminal unary device: the *rotary (R)* (Fig. 5a), which routes each terminal to the next in a cyclic fashion. It comes in two dual varieties, which circulate signals in opposite directions; each of them is the time-reverse of the other. Among other purposes, the rotary is useful for converting device interfaces between a single bidirectional I/O terminal and two unidirectional terminals.

2) Two-state, three-terminal primitives.

We will not attempt here to categorize all 3-terminal, 2-state ABRC devices; there are 2 × 3 = 6 syndromes in this case, thus 6! = 720 different possible unconditionally-reversible transition functions (syndrome permutations) to consider.

However, there is only one 2-state, 3-terminal device that both exhibits **T** symmetry and treats all 3 data terminals symmetrically with each other (call this **D3** symmetry): the *flipping rotary (FR)* (Fig. 5b). It is like the ordinary rotary R, except that each pulse passing through it reverses its orientation.

Also, only two 2-state, 3-terminal devices both have **T** symmetry and treat 2 of their 3 data terminals symmetrically (**D2** symmetry), while not exhibiting full **D3** symmetry: The *controlled flipping diode (CFD)* and the *toggling controlled barrier (TCB)*. The CFD is an FD with a state-toggling control terminal. The TCB (Fig. 6) alternates between wire and barrier behavior in the path between its two data terminals whenever a pulse reflects off its control terminal.

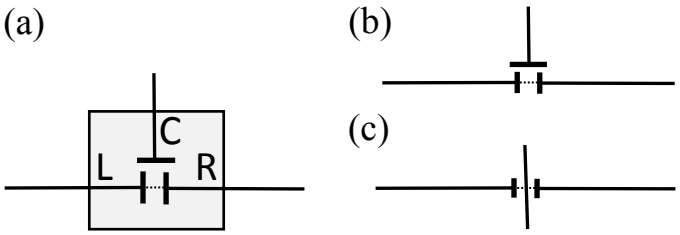

Fig. 6. (a) The *toggling controlled barrier (TCB)* is one of the two types of 2-state, 3-terminal devices that have **T** and **D2** symmetry but not **D3** symmetry. It toggles between having wire and barrier behavior in effect between its left (L) and right (R) terminals whenever a pulse reflects off its control (C) terminal. (b) Simplified icon for TCB. (c) A four-terminal variant of the TCB in which the control pulse passes over the channel instead of reflecting back may be simpler to implement physically in some technologies.

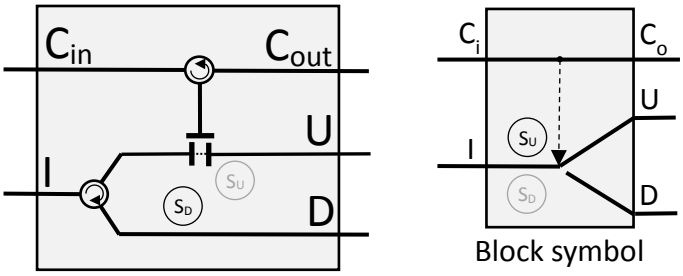

Fig. 7. *Toggling Switch Gate (TSG)* built from a TCB and two rotaries. In the "up state" S_U, inputs on I are routed to the output U, and in the "down state" S_D they are routed to output D. Pulses on the control input C_{in} toggle the state.

At this point, we can conclude our exploration of primitives, because those above are already sufficient to achieve our goal: In particular, we can now prove that the set of device types {R, TCB}, at least, comprises a computationally-universal set.

IV. Higher-Level Constructions and Universality

Here, we give examples of higher-level ABRC components, composed from the primitive devices already discussed, that are sufficient to demonstrate ABRC's computation-universality.

A. Toggling Switch Gate from Toggling Controlled Barrier

The first construct we will examine is the *Toggling Switch Gate (TSG)* (Fig. 7). This routes an incoming data signal I to either the upwards (U) or downwards (D) output terminal, depending on the state, which is toggled by the control. Ordinary (non-toggling) switch gates were previously studied by Feynman in [13], where he develops a simple quantum-mechanical model of reversible computing, and they were shown by him to be universal. Later we will see how to make one.

B. Asynchronous Multiplexer/Demultiplexer

Given a time-series of constant control pulses that are deliberately ordered with respect to the data pulses, a toggling switch gate can implement an asynchronous signal multiplexer/demultiplexer, as shown in Fig. 8. The source of constant control pulses may seem like a clock of sorts; however, since the exact time of arrival of pulses does not matter as long as the order is right, the constraints on the "clock" distribution are looser than

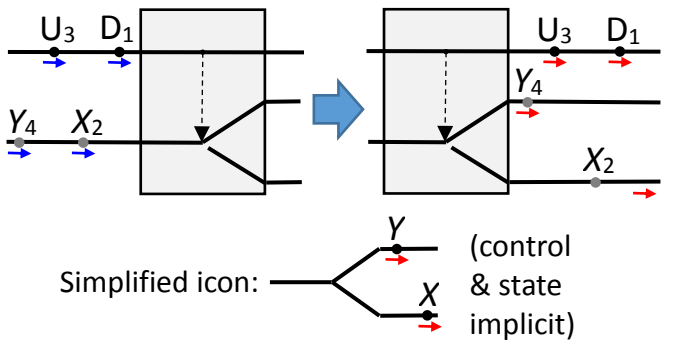

Fig. 8. Asynchronous multiplexer/demultiplexer from a switch gate. Here, we label pulses, and the subscripts denote the order of arrival/departure. Gray dots denote optional (data-dependent) pulses. First, control pulse D_1 toggles the switch from up to down. Then the (optional) data pulse X_2 arrives and takes the downwards path. Then control pulse U_3 arrives and toggles the switch back up. Then data pulse Y_4 arrives and takes the upwards path. In reverse, this demux acts as a mux. When we draw branching or merging wires (bottom), really this represents a switch gate with the associated control pulses.

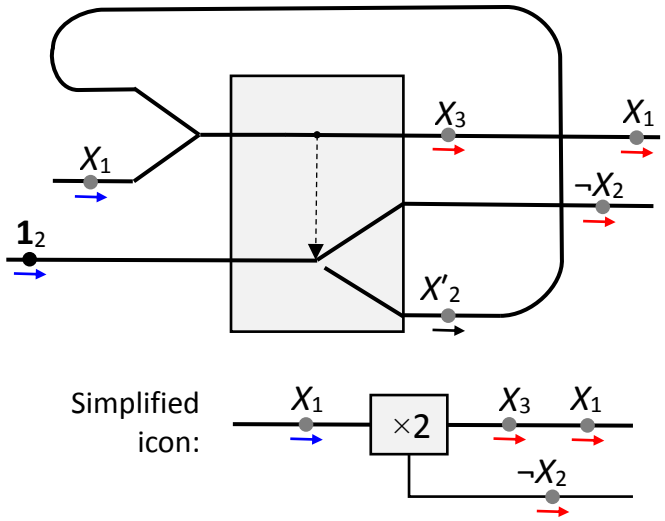

Fig. 9. Asynchronous pulse duplicator from a toggle switch and a mux. Subscripts show event order. First, if the data pulse X_1 at time 1 is present, the mux routes it into the control of the toggle switch, and it toggles the switch gate to the down state. The the constant pulse $\mathbf{1}_2$ comes in at time 2, and is switched down to produce X_2 (which is present if X_1 was present, so is logically equal to a delayed copy of X_1). Then X_2 is circulated around back through the mux and the switch control, restoring the switch to the up state, and emerges (after some delay for the loopback) at time 3 as X_3. If X_1 was not present, all that happens is that the constant pulse $\mathbf{1}_2$ emerges immediately on the upper path as $\neg X_2$, since it is logically equal to $\neg X_1$, but delayed until time 2.

they would be in a synchronous design, so, we still benefit from the asynchronous approach.

C. Asynchronous Pulse Duplicator

See Fig. 9. Using a switch gate and a pulse multiplexer (whose control is left implicit in this diagram) and an additional constant input pulse that is ordered with respect with the data, we can produce an asynchronous *pulse duplicator*—given a single pulse on its X input, it produces two pulses on its output, with a delay between them. If the pulse X is not present, it produces an incidental output pulse ¬X. As is usual in reversible computing, we can decompute any unneeded incidental signals (a.k.a. garbage) at a later time using a mirror circuit.

D. Non-Toggling Switch Gate

Using a pulse duplicator and a toggling switch gate, we can build a non-toggling switch gate (Fig. 10). We simply duplicate

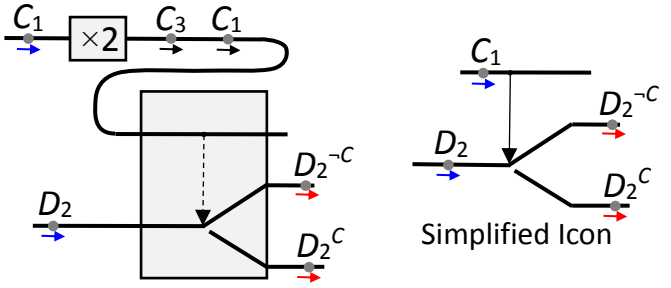

Fig. 10. Ordinary (non-toggling) *Switch Gate (SG)* from a pulse duplicator and a toggling switch gate. First the control pulse C optionally comes in at time 1, and (if it is present) it is duplicated by the pulse duplicator, to produce one copy immediately and, another delayed one at time 3; both are fed into the control of the toggle switch, temporarily toggling it down if the control was present. In between these events, the data pulse D comes in at time 2, and is routed down if the control pulse was present, and up otherwise. The delayed control pulse at time 3 restores the toggle switch back to the default "up" state.

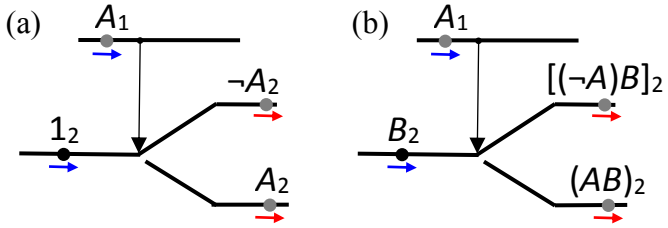

Fig. 11. (a) Single-rail to dual-rail converter using a switch gate. The input A comes in at time 1, and then a constant pulse (logic 1) comes in at time 2. If A was present, the constant pulse is routed down, producing a copy of A; else the constant pulse is routed up, producing $\neg A$, the logical complement of A (this gives us a NOT operation). The pair of outputs can be used as a dual-rail encoding of A. (b) AND operation using a switch gate. If pulse A is present at time 1, then when the optional data pulse B comes in at time 2, it is routed down, producing the logical AND of A and B, denoted AB. If A was not present, then B gets routed up, thus producing an output pulse there if the expression NOT(A) AND B (denoted $(\neg A)B$) is true. Since we can implement NOT and AND functions, it follows that ABRC is computation-universal.

the control pulse in such a way that one copy of it arrives before the data pulse, and the other one arrives after the data pulse. In this way, the data pulse is routed along one branch if the control pulse was present in a certain time window, and along the other branch otherwise.

Switch gates are already known to be computation-universal [13], so the preceding material already establishes that ABRC is computation-universal. However, for clarity, we will now show a couple of additional constructions that illustrate why switch gates (in particular) are computation-universal.

E. Single-Rail/Dual-Rail Converter

Using a (non-toggling) switch gate, we can convert single-rail signals to dual-rail (Fig. 11a). In a single-rail signal, the presence or absence of a pulse during a certain time window encodes a bit. In a dual-rail signal, a pulse is always present on exactly one of two wires; which one it is present on encodes the bit's value. One advantage of dual-rail signaling is that an in-place Boolean NOT operation on a dual-rail encoded signal can be "performed" trivially, by just relabeling the wires. However, single-rail encodings are more compact, so it is helpful to be able to convert between representations if needed.

F. AND Gate & Computation Universality

Using a (non-toggling) switch gate, we can implement a Boolean AND operation on single-rail inputs A,B (Fig. 11b). An incidental output of $\overline{A}B$ (not-A and B) is also produced. Meanwhile, using the complemented output from Fig. 11a yields a NOT operation. We can combine AND and NOT operations as needed, and thereby (via standard logic constructions) produce OR operations, and more generally any Boolean function. Standard Bennett reversals [4] can be used to decompute "garbage" data (undesired incidental outputs) as needed, to obtain the desired output and (at most) an incidental copy of the input.

V. Possible Technological Realizations

How can we implement the above model physically? A possible implementation technology may exist in superconducting circuits using Josephson junctions. Magnetic flux threading a superconducting loop is naturally quantized to multiples of the *magnetic flux quantum* $\Phi_0 = h/(2e) \approx 2.07 \times 10^{-15}$ Wb. Due to

Meissner-effect trapping, these flux quanta are naturally conserved in a circuit, and they can propagate in the form of soliton-like current pulses near-ballistically over reasonable distances along certain one-dimensional structures, such as long Josephson junction (LJJ) transmission lines. These mobile *fluxons* (also known as single flux quanta, or SFQ), propagate at a substantial fraction of the speed of light, and typical initial pulse profiles in SFQ-based designs are ~2 mV pulse height and ~1 ps pulse duration (albeit with subsequent dispersion as the pulse travels). Pulse frequencies can thus attain many GHz even with reasonably wide spacing between pulses. Individual flux quanta can also be stored quiescently and controllably in SQUID-like structures (superconducting loops incorporating Josephson junctions), and pulse interactions can carry out logic, as is done in RSFQ [14] and related superconducting logic families.

Thus, it is reasonably plausible that appropriately designed, sufficiently finely-tuned superconducting circuit structures may support a sufficiently wide variety of basic ABRC primitives, interconnected by passive transmission lines, to allow ABRC logic blocks of useful sizes to be efficiently constructed. Since fluxons are naturally polarized by the direction of field winding, the most appropriate primitive devices for this case will likely operate on binary pulse type multiplicities ($m = 2$).

Of course, numerous details of this potential implementation concept still remain to be worked out. Could ABRC be the basis for designing superconducting circuits that operate near-reversibly, and thus dissipate much less than the typical fluxon energy per operation, while still operating at multi-GHz frequencies? If so, then it could potentially greatly increase the energy-efficiency and overall competitiveness of superconducting electronics as a technology for high-performance computing.

VI. Conclusion

In this paper, we presented and explored a new circuit-based theoretical model of reversible computing that, in contrast to almost all previous reversible computing models, is substantially *asynchronous*, rather than being fully synchronous. The existing literature addressing asynchronous reversible computing (*e.g.*, [15], [16]) focuses on cellular automata, and covers only a few specific kinds of asynchronous primitives. It does not attempt to develop a broad, general circuit-based model of asynchronous reversible computing, as we do here.

In the present paper, we have not yet begun exploring how to further generalize ABRC to stochastic or quantum versions, but it is likely that these kinds of generalizations would be fairly straightforward. For example, for a quantum version of ABRC, we could simply replace the injective Mealy-machine transition function with a unitary transformation over a basis set of possible symbol-state syndromes. The machine would then in general enter a superposition of configurations in which pulses are propagating along different sets of paths. This new model of quantum computation might inspire new ideas for the design of quantum algorithms, or new concepts for the physical implementation of quantum computing.

In the meantime, the ABRC model, as it stands, appears worthy of further development. Some useful next steps for this line of research include: (1) Classify the possible ABRC devices with somewhat larger numbers of states, terminals, and pulse polarities, looking for useful primitives, and also include devices whose transition functions are only conditionally reversible; (2) develop a wider variety of circuit constructions, including more efficient constructions of Boolean logic gates, as well as useful higher-level functions such as adders, *etc.*; (3) flesh out the ideas for a superconducting implementation of ABRC discussed in the previous section, and assess their feasibility as a possible path towards high-performance computing beyond the energy-efficiency limits of CMOS; (4) conduct a search for other possible implementation technologies that might lead to efficient realizations of the ABRC concept.

We hope this paper has illustrated that the range of possible ways in which to approach the goal of reversible computing is broader than previously realized. Holding in our mind's eye the clear vision, supported by the most fundamental laws of physics, that we can find ways to keep computer efficiency increasing indefinitely, a vast field of promising new directions to explore in pursuit of that goal opens up ahead of us. Let's get to work.

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," IBM J. Res. Dev., vol. 5, no. 3, pp. 183–191, July 1961.

[2] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors 2.0," 2015, https://www.semiconductors.org/main/2015_international_technology_roadmap_for_semiconductors_itrs.

[3] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam and D. Burger, "Dark silicon and the end of multicore scaling," IEEE Micro, vol. 32, no. 3, pp. 122–134, April 2012.

[4] C. Bennett, "Logical reversibility of computation," IBM J. Res. Dev. vol. 17, no. 6, pp. 525–532, November 1973.

[5] K. Likharev, "Dynamics of some single flux quantum devices: I. Parametric quantron," IEEE Trans. Magnetics, vol. 13, no. 1, pp. 242–244, January 1977.

[6] K.E. Drexler, Nanosystems, John Wiley & Sons, Inc., November 1992.

[7] T. Hogg, M.S. Moses, and D.G. Allis, "Evaluating the friction of rotary joints in molecular machines," Mol. Syst. Des. Eng., vol. 2, pp. 235–252, May 2017.

[8] S.G. Younis and T.F. Knight, Jr., "Practical implementation of charge recovering asymptotically zero power CMOS," in Proc. 1993 Symp. on Research in Integrated Systems, MIT Press, 1993, pp. 234–250.

[9] C.S. Lent, P.D. Tougaw, W. Porod, G.H. Bernstein, "Quantum cellular automata," Nanotechnology, vol. 4, no. 1, p. 49–57, January 1993.

[10] V. Anantharam, M. He, K. Natarajan, H. Xie and M.P. Frank, "Driving fully-adiabatic logic circuits using custom high-*Q* MEMS resonators," in ESA/VLSI, pp. 5-11, 2004.

[11] E. Fredkin and T. Toffoli, "Conservative logic," Int. J. Theor. Phys., vol. 21, no. 3, pp. 219–253, April 1982.

[12] M.P. Frank, "Foundations of generalized reversible computing," in Reversible Computation, I. Phillips and H. Rahaman, Eds., Lecture Notes in Computer Science, vol. 10301. Cham: Springer, 2017, pp. 19-34.

[13] R.P. Feynman, "Quantum mechanical computers," Found. Phys., vol. 16, no. 6, pp. 507–531, June 1986.

[14] K.K. Likharev, "Rapid single-flux-quantum logic," in The New Superconducting Electronics, Netherlands: Springer, 1993, pp. 423–452.

[15] T. Toffoli and N. Margolus, Cellular Automata Machines: A New Environment for Modeling, MIT Press, 1987.

[16] J. Lee, F. Peper, S. Adachi, K. Morita and S. Mashiko, "Reversible computation in asynchronous cellular automata," in Unconventional Models of Computation. Berlin, Heidelberg: Springer, 2002, pp. 220–229.