Exceptional service in the national interest

Asynchronous Ballistic Reversible Computing

Michael P. Frank
Center for Computing Research
Sandia National Laboratories

2nd IEEE International Conference on
Rebooting Computing (ICRC 2017), Washington, DC
Thursday, November 9th, 2017

Approved for Public Release
SAND2017-12534 C

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.

Abstract of Paper

***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.**

Talk Outline

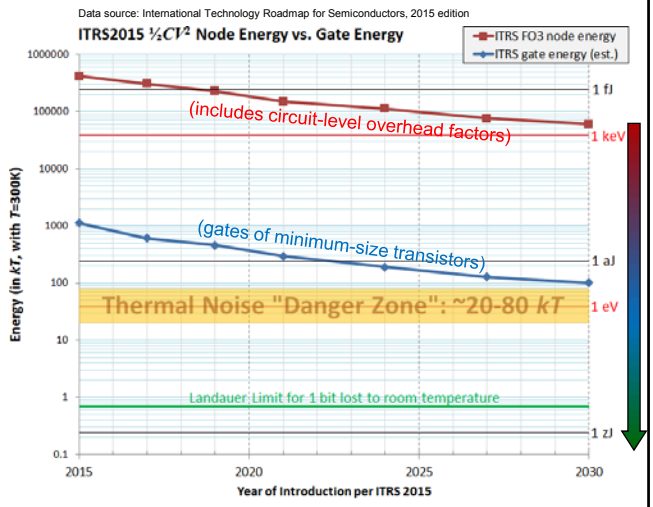
- Motivations:
  - End of the semiconductor roadmap & the power-performance wall
  - Reversible computing required for long-term sustainable growth
- Existing implementation paradigms for reversible computing:
  - Adiabatic, & (synchronous) ballistic. Problems and limitations
- Introducing a new theoretical implementation paradigm: *Asynchronous Ballistic Reversible Computing* (ABRC):
  - Summary/potential advantages
  - General network model – Initial & derived requirements
  - Examples of some primitive ABRC devices
  - Universality construction
  - Superconducting implementations – Simulations from LPS/JQI
- Conclusion

3

Semiconductor Roadmap is Ending…

- Thermal noise on gates of minimum-width segments of FET gates leads to channel PES fluctuations when $E_g \lesssim$ 1-2 eV
  - Increases leakage, impairs practical device performance
    - Thus, ITRS has minimum gate energy asymptoting to ~2 eV
- Also, real logic circuits incur many further overhead factors:
  - Transistor width 10-20× min.
  - Parasitic (junction, etc.) transistor capacitances (~2×)
  - Multiple (~2) transistors fed by each input to a given logic gate
  - Fan-out to a few (~3) logic gates
  - Parasitic wire capacitance (~2×)
- Due to all these overheads, the energy of each bit in real logic circuits is many times larger than the min.-width gate energy
  - 375-600× (!) larger in ITRS'15
    - ∴ Practical bit energy for irreversible logic asymptotes to ~1 keV!
- Practical, real-world logic circuit designs can't just magically cross this ~500× architectural gap!
  - ∴ Thermodynamic limits imply much *larger* practical limits!
    - The end is near!

Data source: International Technology Roadmap for Semiconductors, 2015 edition

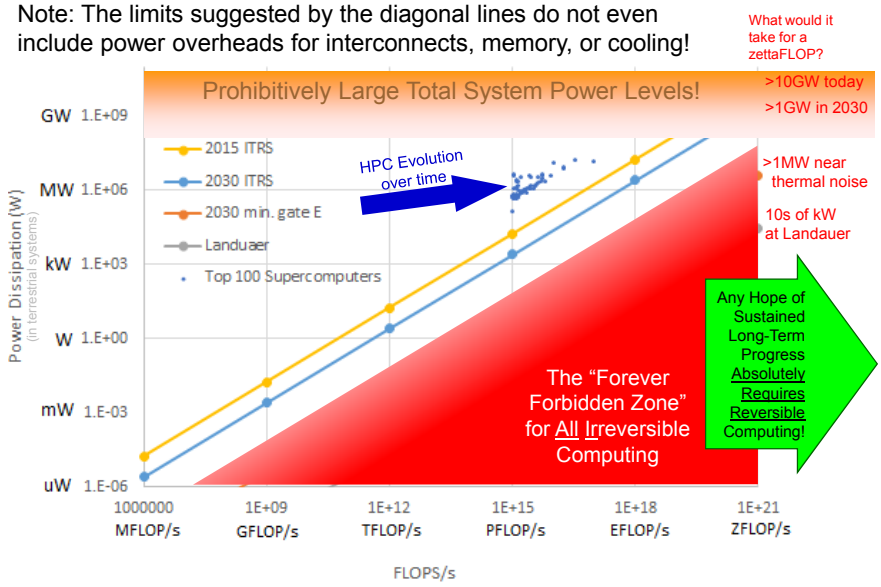


Only reversible computing can take us from ~1 keV at the end of the CMOS roadmap, all the way down to $\ll kT$.

4

Implications for FLOPS & power

Note: The limits suggested by the diagonal lines do not even include power overheads for interconnects, memory, or cooling!



5

Reversible Computing – Why? What?

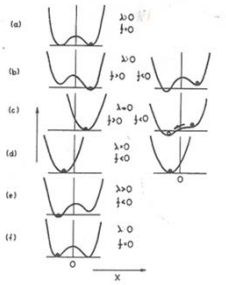
- Fundamental microphysics is *reversible*—it conserves information!
  - Therefore, losing information from a digital system (by erasing/overwriting it) necessarily implies ejecting that information into the system's environment
    - Once thermalized by the environment, information that was previously known (correlated) becomes entropy (unknown/uncorrelated information)
      - …and this implies dissipation of $kT \ln 2$ of organized energy (work) to heat at temperature T per bit of information lost (Landauer's Principle)
- Unfortunately, in the conventional (irreversible) computing paradigm, we discard computational information all the time…
  - Every active conventional logic gate destructively overwrites its output node on every clock cycle, losing the information embodied in the previous output
    - Similarly for line drivers, on every bus cycle for every interconnect wire
    - And for memory cells/lines, every time a cell is written, read out or refreshed
- How can we compute without losing information? (And please note that "computing" <u>includes</u> driving interconnects, accessing memory, etc. as needed!)
  - *Reversibly transform* states, instead of destructively overwriting them!
    - This then allows avoiding the Landauer principle's limit on energy efficiency

There is <u>no</u> known fundamental (technology-independent) limit on computational energy efficiency, but <u>only</u> if the reversible computing principle is used!

6

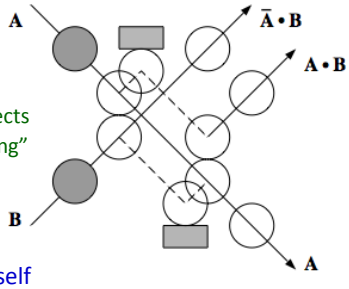
Adiabatic Reversible Computing

A general class of implementation techniques for reversible computing that relies on controlled adiabatic transformations of the information-bearing degrees of freedom.



- Has been explored in various physical systems:
  - Superconducting electronics (Likharev '77, etc.)
  - LC switching circuits (Fredkin & Toffoli '78)
  - Adiabatic CMOS (Seitz '85, etc.)
  - Molecular nanomechanical logic (Drexler '91, etc.)
  - Single-electron quantum dots (Lent '92, etc.)
- Some drawbacks of this class of approaches:
  - Every logic transition must be explicitly driven by a power-clock
    - Numerous clocks are required in combinational and sequential designs
    - Substantial design complexity overhead to distribute clocks to every gate
  - Challenging to design finely-tuned, high-*Q* power-clock resonators
    - Problems with load balancing in long-range global clock distribution networks with large parasitics, avoiding data-dependent back-action

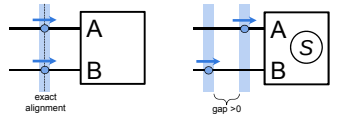
Ballistic Reversible Computing

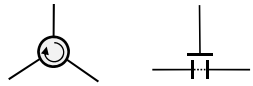


- Original concept:
  - Fredkin & Toffoli's *Billard Ball Model* of computation ("Conservative Logic," 1982)
    - Based on elastic collisions between moving objects
    - Spawned a subfield of "collision-based computing"
      - Localized pulses/solitons in various media
- No power-clock signals needed!
  - Devices operate when data signals arrive
  - The operation energy is carried by the signal itself
    - Most of the signal energy is preserved in outgoing signals
- However, existing design concepts for ballistic computing invoke implicitly *synchronized* arrivals of ballistically-propagating signals…
  - Making this work in reality presents some serious difficulties, however:
    - Unrealistic in practice to assume precise alignment of signal arrival times
      - Thermal fluctuations & quantum uncertainty, at minimum, are always present
    - Any relative timing uncertainty leads to chaotic dynamics when signals interact
      - Exponentially-increasing uncertainties in the dynamical trajectory
- Can we come up with a ballistic model that avoids these problems?

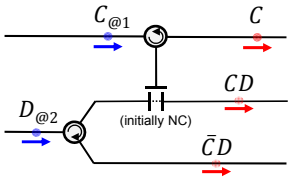
Asynchronous Ballistic Reversible Computing

- To avoid the problems with dynamical chaos that are inherent to collision-based computing,
  - We must avoid any direct interaction between ballistically-propagating signals
- Instead, require temporally-localized pulses to arrive at *distinct*, non-overlapping times
  - Device's dynamical trajectory then becomes *independent* of the precise pulse arrival time
    - Timing uncertainty per logic stage now accumulates only *linearly*, not exponentially
      - Only occasional re-synchronization will be needed
  - To do logic, devices now must have internal state
- No power-clock signals, unlike adiabatic designs
  - Devices simply operate whenever data pulses arrive
  - The operation energy is carried by the pulse itself
    - Most of the energy is preserved in outgoing pulses
      - Signal restoration can be carried out incrementally
- A new project has started at Sandia which aims to implement ABRC in superconducting circuits
  - 3-year, $1.5M internally-funded project





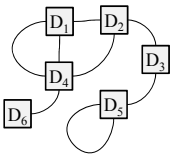
Example ABR device functions

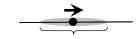


Example logic construction

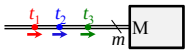
ABRC Model: Starting Requirements

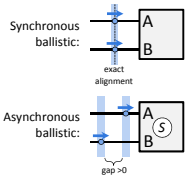
1. Universality – for reversible, and embedded irreversible
2. Network model – devices, bidirectional terminals, links
3. Localized signals ("pulses")
   a. Spatial confinement – Along 1-D signal paths (wires)
   b. Temporal localization – Pulse width specified as bounded
4. Ballistic propagation – along sufficiently large scales
5. Digital interpretation – *m* distinguished signal types
6. Asynchrony – exact pulse arrival times not important
7. Determinism – future depends non-randomly on past
   - Quantum version can generalize this in the usual way
8. Reversibility – over the assumed set of initial states
9. Quiescence – devices don't change in between pulses





Some specified maximum allowable pulse width and/or arrival-time uncertainty



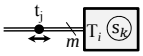


ABRC Model: Derived Requirements

These can be seen to follow from the starting requirements:

10. Non-overlap of arriving pulses – Required for determinism
11. Non-overlap of departing pulses – Required for reversibility
12. One-to-one correspondence between incoming and outgoing pulses
    – Necessary to reversibly carry away incoming pulse energy/timing information
13. Statefulness – To do useful logic, devices must have a stable internal state.
14. The possible ABRC device behaviors are exactly characterized by (isomorphic to) a restricted set of Mealy machines:
    - I/O symbol alphabet consists of $N = n \cdot m$ *compound signal characters*: $c_i^j = \binom{t_j}{T_i}$

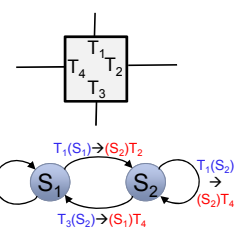
      $$\Sigma = \left\{c_i^j\right\} = \left\{\binom{t_j}{T_i}\right\}$$

      State set: $S = \{s_1, \ldots, s_K\}$
      - where $T_i \in \{T_1, T_2, \ldots, T_n\}$ is any of n I/O terminals, each multiplicity m,
      - and $t_i \in \{t_1, t_2, \ldots, t_m\}$ is any of the m signal types.
        – Can easily generalize this to cases where not all terminals have the same arity
    - Transition function $f: \Sigma \times S \to S \times \Sigma$ is (at least conditionally) reversible
      - Injective at least over some assumed subset $A \subseteq \Sigma \times S$ of possible input syndromes
    - Machine implements an injective transformation of at least the subset of all input strings for which its assumed precondition for reversibility is met at each step

More on FSM correspondence

- As mentioned, ABRC devices correspond exactly to reversible Finite-State Machines (more specifically, Mealy machines), graphed with 1 input symbol and 1 output symbol per directed edge
  - Each input/output symbol $T \in \{T_1, \ldots, T_n\}$ labels the terminal on which the next pulse arrives/leaves
    - The device both (potentially) transforms the symbol passing through, and changes its own internal state.
  - Each edge is an arrow, here labeled: $T_i(S_k) \to (S_\ell)T_j$
    - Says that if an input symbol T_i (i.e., an incoming pulse on terminal T_i) comes in to the device when it is in internal state S_k, the internal state becomes S_ℓ and the device emits output symbol T_j (outgoing pulse on terminal T_j).
  - Transition table forms an injective map between subsets of possible initial and final pairs $\{(T_i, S_j)\}$.
    - A particular terminal-state pair is called a "syndrome."
    - Map is time-symmetric if time-reversal symmetry holds when the device state is held constant
      – However, that is not required for reversibility.
- The device's action transforms strings of input symbols to output strings, reversibly…
  - E.g., "$T_1T_2T_2T_3T_1$"$(S_1) \to (S_2)$"$T_2T_3T_3T_4T_2$"
    - In this notation, read the strings right to left…
      – Visualize as pulse trains moving from left to right.



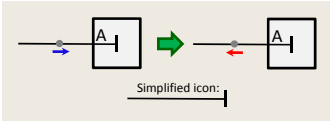
Initial Syndrome | Final Syndrome

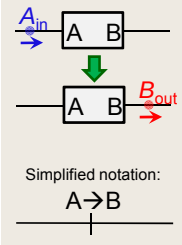
Input Symbol	Initial State	Final State	Output Symbol
T_1	S_1	S_2	T_2
T_2	S_1	S_1	T_3
T_1	S_2	S_2	T_4
T_3	S_2	S_1	T_4

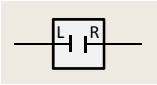
T_1 in: "Go to state 2." T_2 out: "Just got to state 2."
T_2 in: "Stay in state 1." T_3 out: "Staying in state 1."
T_3 in: "Go from 2 to 1." T_4 out: "I was just in state 2."

ABRC Primitives

- Here, we enumerate some simple unary ABRC primitives:
  - One-terminal unary primitives:
    - *Pulse Reflector* (PR)
  - Two-terminal unary primitives:
    - The one-state, two-terminal primitives:
      - *Wire* (W) a.k.a. signal renamer
        - » Functionally identical to a section of wire
      - *Barrier* (B)
        - » Two pulse reflectors back-to-back
    - *(Continued on next slide…)*



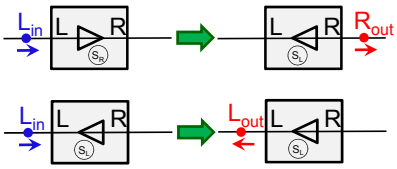




ABRC Primitives, cont.

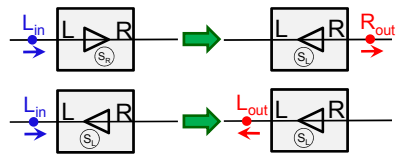
- Unary primitives, cont.
  - Two-terminal unary primitives, cont.
    - Two-state, two-terminal unary primitives:
      - We can categorize them using these symmetry groups:
        - » **T** – Time-reversal symmetry (operation is the same in either time direction)
        - » **D** – Data-terminal reversal symmetry (operation unchanged if terminals swapped)
        - » **TS** – Time/state reversal symmetry (unchanged if time reversed & states swapped)
      - All nontrivial 2-state, 2-terminal unary devices can then be classified as follows:
        - » Devices exhibiting both **T** and **D** symmetries
          - ❖ *Flipping Diode* (FD) – Can use it as a memory!
        - » Devices exhibiting both **D** and **TS** symmetries
          - ❖ *Anti-Flipping Diode* (AFD)
          - ❖ *Toggling Barrier* (TB)
        - » Devices exhibiting none of these symmetries
          - ❖ *Directional Flipping Diode* (DFD)
          - ❖ *Flipping Comparator* (FC)

Flipping Diode Behavior

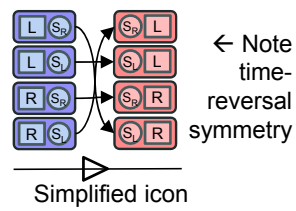


Flipping Diode: More Discussion

- The only nontrivial two-state, two-terminal, time-reversal-symmetric (TRS) A.R. device
  - The only other TRS two-state, two-terminal AR devices are just barriers or renamers with redundant states
- Equivalent to a reversible 1-bit temporary memory cell (or delay element) with bidirectional I/O…
  - With some signal routing/renaming, this can also act as a reversible SR flip-flop (reversible SRAM cell) useable in pipelined logic
    - And if we also add a simple sequencing protocol, we can even make it into an asynchronous reversible AND gate!

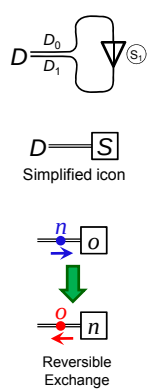


Input Symbol	Initial State	Final State	Output Symbol
L	S_R	S_L	R
L	S_L	S_L	L
R	S_R	S_R	R
R	S_L	S_R	L



Flipping Diode as Memory/Delay Cell

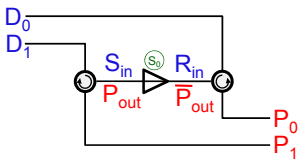
- Bundle the two terminals of the flipping diode into one dual-rail signal,
  - And we can see its function as a reversible memory/delay element…
- Let the dual-rail bidirectional I/O signal be called "*D*" (for data bit), with values 0, 1
  - Encoded by pulses on the D_0 and D_1 lines respectively
- Let the internal state variable of the flipping diode be called *S*, with values 0, 1
  - Encoded by states S_0 and S_1 for the up/down orientations of the diode in this diagram, respectively
- Then it's easy to see that the function of this element can be described as follows:
  - $D_{out} = S_{old}$; $S_{new} = D_{in}$. *I.e.*, exchange $D \leftrightarrow S$.
    - (Output old value, store new value.)
- Its operation on bit-strings is to delay their data by 1 pulse.




```
In:  0 0 1 1 1 0 0 0
Out:   0 0 1 1 1 0 0 0
```
Time

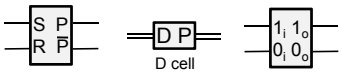
Reversible Flip-Flop from Flipping Diode

- Rename states/terminals of flipping diode like so:
  - S_R = "0" state; S_L = "1" state
  - L_{in} = "S(et)" (to 1) input.
  - R_{in} = "R(eset)" (to 0) input.
  - R_{out} = "not(P)" = "Previous state was 0."
  - L_{out} = "P" = "Previous state was 1."
- Use constant rotaries to split bidirectional terminals into separate input/output terminals
  - Now we have a dual-rail D(ata) input and a dual-rail P(revious state) output



(3 primitives)

Input Symbol	Initial State	Final State	Output Symbol
D_0	S_0	S_0	P_0
D_1	S_0	S_1	P_0
D_0	S_1	S_0	P_1
D_1	S_1	S_1	P_1



SR cell notation | D cell notation | Binary notation

All 2-state, 2-terminal unary devices

- Flipping Diode (FD), Anti-Flipping Diode (AFD), Toggle Barrier (TB), Directed Flipping Diode (DFD), Flipping Comparator (FC)

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]	$(S_1)R$	$(S_0)R$	$(S_1)L$	$(S_1)R$	$(S_1)L$
$L(S_1)$	$(S_1)L$	$(S_0)L$	$(S_0)R$	$(S_0)L$	$(S_1)R$
$R(S_0)$	$(S_0)R$	$(S_1)R$	$(S_1)R$	$(S_0)R$	$(S_0)R$
$R(S_1)$	$(S_0)L$	$(S_1)L$	$(S_0)L$	$(S_1)L$	$(S_0)L$

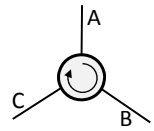
[a] In this table, $\sigma(s)$ denotes an input syndrome (σ, s), and an output syndrome (s, σ) is written $(s)\sigma$.

18

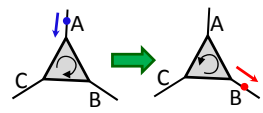
ABRC Primitives, cont.

- Unary primitives, cont.
  - Three-terminal unary primitives:
    - One-state, three-terminal primitives:
      - Only one: ***Rotary*** **(R)**
    - Two-state, three-terminal primitives:
      - Some important symmetries:
        - » **D3** – All 3 data terminals treated symmetrically
        - » **D2** – A specific 2 out of the 3 data terminals are interchangeable with each other
      - Some interesting cases:
        - » Devices with both **T** and **D3** symmetry:
          - ❖ Only one: *Flipping Rotary* (FR)
        - » Devices with **T** and **D2**, but not **D3** symmetry:
          - ❖ *Controlled Flipping Diode* (CFD)
          - ❖ ***Toggling Controlled Barrier*** **(TCB)**

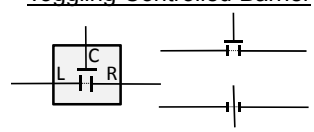
Rotary (CW)



Flipping Rotary



Toggling Controlled Barrier



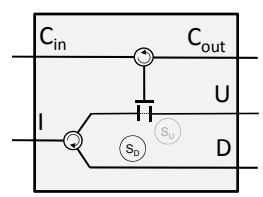
Universality Construction

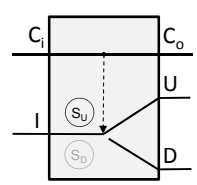
(slide 1 of 6)

- **Theorem:** {R, TCB} comprises a universal set of primitives for reversible (and embedded irreversible) computing
  - Constructive proof proceeds as follows:
    1. Using two rotaries and a toggling controlled barrier,
       - We can construct a toggling version of the reversible "switch gate" studied by Feynman and others
         - » We can then also build up a non-toggling version of it…

(Continued on following slides…)

Toggling Switch Gate:





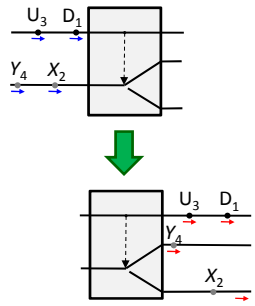
Block symbol

Universality Construction
(slide 2 of 6)

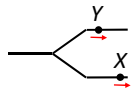
- Universality theorem, cont.
  - Constructive proof, cont.

    2. A toggling switch gate can be used as an asynchronous pulse (de)multiplexer
       - Requires pre-prepared supply of control pulses tho… ☹
         - » Still may be easier than fully-clocked adiabatic logic
         - » Also, we may discover other universality constructions later that reduce the number of pre-prepared control streams that are needed

Asynchronous (De)Mux:



Simplified icon:



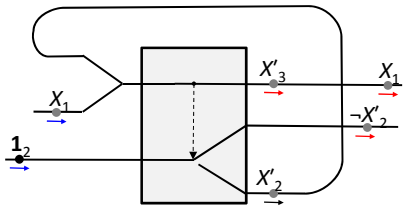
(control & state implicit)

Universality Construction
(slide 3 of 6)

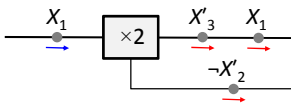
- Universality theorem, cont.
  - Constructive proof, cont.

    3. A toggling switch gate plus a mux can make a pulse duplicator
       - Produces incidental output ("garbage")
         - » This can be cleaned up using the usual approaches (Bennett reversal)

Pulse Duplicator:



Simplified icon:

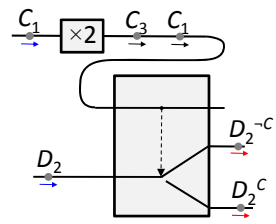


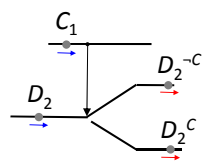
Universality Construction
(slide 4 of 6)

- Universality theorem, cont.
  - Constructive proof, cont.

    4. With a pulse duplicator plus a toggling switch gate, we can build a non-toggling switch gate
       - This gate was previously described by Feynman and shown to be universal
         - » We'll go ahead and show why…

Non-toggling Switch Gate:





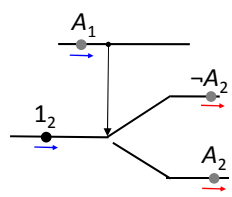
Simplified Icon

Universality Construction
(slide 5 of 6)

- Universality theorem, cont.
  - Constructive proof, cont.

    5. E.g., the (non-toggling) switch gate can be used to build a single-rail to dual-rail converter…
       - This can also be considered as a NOT gate that also produces an extra (garbage) copy of its input
         - » Note we need the constant "1" pulse to be supplied…

Single-rail to Dual-rail Converter:
(Includes NOT function)

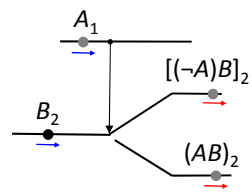


Universality Construction

(slide 6 of 6)

- Universality theorem, cont.
  - Constructive proof, cont.
    6. …and the switch gate can also be used to produce a reversible AND function
       - Also produces $\overline{A}B$ as a garbage output
    7. Standard techniques like Lecerf reversal and the Bennett trick can be applied to decompute all garbage,
       - while leaving us with just the desired result, and a copy of the input.
    - Thus, we can compute any Boolean function using an ABRC circuit made from {R, TCB} devices only. **Q.E.D.**

Asynchronous reversible AND gate:



Remarks on Universality Construction

- The above construction is sufficient for formally proving the computation universality of the model…
  - But, considered as a logic synthesis method, it clearly has significant practical drawbacks…
    - In particular, this construction requires a great many streams of constants / control signals (effectively clocks)
      - Some of these streams may be reused, but the overhead is still high
  - Open research problem:
    - Find much simpler constructions for general functions, ideally ones requiring no (or fewer) periodic clocking streams
      - Considering primitives other than {R, TCB} could be helpful for this
        - » Including primitives based on *m*-ary pulses

Physical realizations of ABRC?

- Of course, to be useful, this model needs to be realized in a specific physical implementation technology that actually provides (nearly) thermodynamically-reversible operation.
  - Need some kind of soliton-like, near-ballistically-propagating pulse,
    - or some sort of particle or quasiparticle (or a larger bound object).
  - Need some physical state variable that can stably maintain at least binary state within the devices
    - for the toggling devices
  - Need a means of physically interacting the pulses with the device states…
    - in ways that can reliably, and almost physically-reversibly, implement at least a universal subset of (probably 2- and 3-terminal) primitive devices.

27

ABRC in superconducting circuits

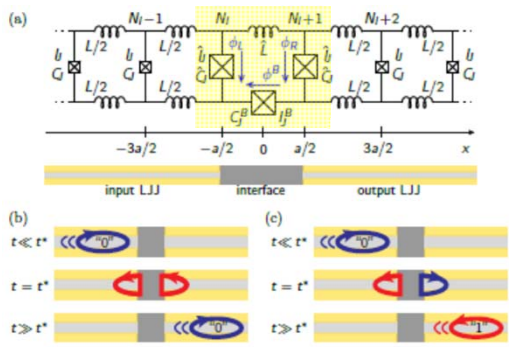
- One intriguing possible candidate implementation technology is to use superconducting circuits…
  - SFQ (single flux quantum, or fluxon) pulses on appropriately constructed superconducting transmission lines can carry info. with relatively low dispersion and high propagation velocity (e.g. 2/3 c)
    - Fluxons are naturally quantized by the SQUID-like circuits that produce them, and are naturally polarized (carry 1 bit's worth of +/– polarization state information per pulse)
      - Need to select suitable ABRC primitives operating on arity-2 signals
  - Fluxons trapped in loops (SQUID-like structures) can hold data quiescently
    - Generally, loops hold integer numbers of fluxons in some small range: …, –2, –1, 0, +1, +2, …
  - How exactly to implement the reversible interactions?
    - A 3-year, internally-funded project is just starting at Sandia to investigate this…

28

A Very Recent Advance!

Wustman (LPS) & Osborn (JQI) '17 (preprint), "Efficient reversible logic gates without adiabatic constraint: Fluxon resonant scattering with polarity changes"

- The circuit shown at right can be considered as a 2-terminal ABRC device for binary pulses (fluxons)
  - The specified function is to preserve or flip the polarity of a fluxon passing through, depending on device parameters
- Here, the "wires" are LJJ transmission lines
  - Major loss mechanism is resonant plasmon emission
    - With lattice spacing $0.4\lambda_J$, fluxon decay time is ~10^7 junction switching times given initial $v = 0.6c$.



- W&O's paper also describes some more complex (4-terminal) devices
  - Synchronous so far, but they are now starting to explore asynchronous

W&O's simulation of identity/NOT

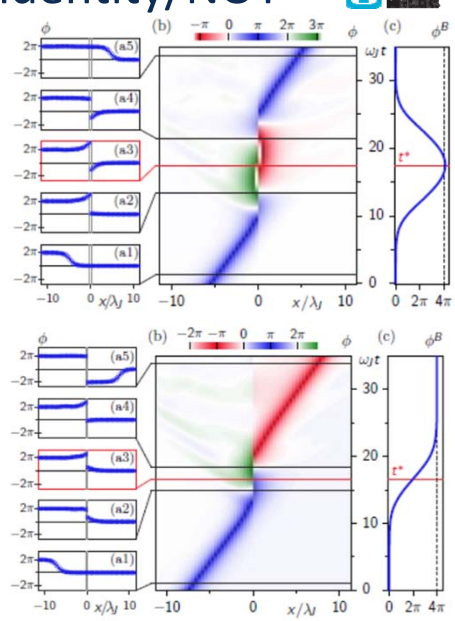
- Direct numerical integration of JJ circuit's equations of motion
  - Lagrangian:

$$\mathcal{L} = \left(\frac{\Phi_0}{2\pi}\right)^2 \left[\sum_n \frac{C_{J,n}}{2}(\dot{\phi}_n)^2 + \frac{C_J^B}{2}(\dot{\phi}^B)^2\right] - \left(\frac{\Phi_0}{2\pi}\right)\left[\sum_n I_{J,n}(1-\cos\phi_n) + I_J^B(1-\cos\phi^B)\right] - \frac{1}{2}\sum_n \left[L_n^A(I_n^A)^2 + L_n^B(I_n^B)^2\right]$$

  - Gives a discrete approximation to sine-Gordon equation:

$$\ddot{\phi} - c^2\phi'' + \omega_J^2 \sin\phi = 0$$

- Scattering interaction at interface is nearly elastic
  - Loss in fluxon velocity of only 4%
    - Loss in energy of 2.1-2.5%



Conclusion

- Reversible computing is absolutely required in the long term to sustain practical performance growth of digital systems
  - This is guaranteed by irrefutable facts of fundamental physics
- However, the existing implementation paradigms for reversible computing suffer from some problematic practical limitations:
  - Substantial clocking overheads in adiabatic reversible technologies
  - Chaotic instabilities in synchronous ballistic reversible technologies
- We introduced a new theoretical paradigm for ***Asynchronous Ballistic Reversible Computing*** that aims to solve these problems
  - Present logic constructions still use some clock-like constant streams, but with further exploration of the design space, these might be reduced
- We are currently investigating fluxon-based Josephson junction circuits as an implementation technology for ABRC at Sandia
  - Project goal: Experimentally demonstrate efficient ABRC circuits
- A new and better foundation for reversible computer design?