2LAL Gates for Support of Quantum Computer Control Electronics

Technical report EPD002 v1.01, November 3, 2019

Erik P. DeBenedictis
Albuquerque, NM 87112
erikdebenedictis@gmail.com

Abstract

This document defines a variant of Michael Frank's fully-pipelined 2-Level Adiabatic Logic (2LAL)[1, 2] that supports inverters and arbitrary 2- or 3-input gates. The variant alters clocking but is otherwise compatible with existing fully-pipelined (shift register) 2LAL gates.

Frank invented 2LAL as an energy efficient alternative to room-temperature CMOS, a goal that required 2LAL to have high throughput and high speed. However, the quantum computer control electronics approach in Ref. 3 uses Josephson junctions for logic and 2LAL for memory. The fact that fast Josephson junction logic is available to the designer reduces the need for the 2LAL variant to have high speed, while the desire for large memories increases the importance of low-transistor-count circuits.

To better serve quantum computer control applications, this document defines a 2LAL variant that has a somewhat more sophisticated clocking structure needed to support of data inversion—thus eliminating the need for the entire system to be quad rail, as in Frank's 2LAL, with the consequential doubling of transistor count. In addition to inversion, the proposed variant can compute or uncompute any logic function of two or three variables.

Keywords—2LAL, quantum computer control electronics, CRL

Preliminaries

Let us first discuss changing the way 2LAL is described in published papers[1, 2] to a form more applicable to the ideas in this document.

Clocks

Frank's fully-pipelined (shift register) 2LAL is defined for four non-overlapping clocks, as illustrated in Fig. 1a. Each clock must be available to the circuit in true and complement form. However, the clocks have the property that the complement of each clock is another clock, so only four distinct signals are needed.

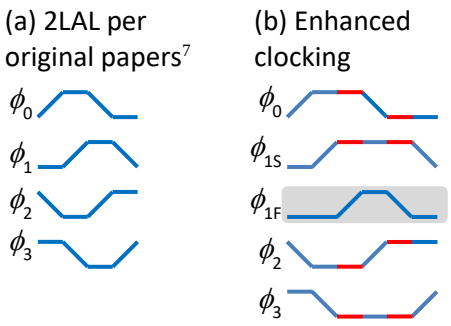

Fig. 1. 2LAL requires four non-overlapping clocks, but they don't need to be perfectly symmetric. For this document, the flat top of ϕ_1 is extended on either end, yielding ϕ_{1S} (slow) and ϕ_{1F} (fast). The resulting clock will drive all existing 2LAL circuits, albeit somewhat more slowly.

We expand the clock waveforms by adding a gap on either side of the flat top of ϕ_1, which is equivalent to stopping all four clocks in two places. This is illustrated by the red segments in Fig. 1b. Clock ϕ_1 is renamed ϕ_{1S}, S for slow, consistent with the terminology in Ref. 4. We also add a variant of ϕ_{1S} called ϕ_{1F}, F for fast, with a pulse that fits entirely within the flat top of ϕ_{1S}.

Using either ϕ_{1S} or ϕ_{1F} as ϕ_1, the clocks in Fig. 1b have the non-overlapping property required by fully-pipelined 2LAL and can drive existing circuits. However, the new clocking comes with some cost:

- There are six instead of four transitions per cycle, so there will be less throughput for a given transition time—and energy efficiency is proportional to transition time for adiabatic circuits.
- Six instead of four clock signals will be required. The complements of each of the clocks ϕ_0, ϕ_{1S}, ϕ_2, and ϕ_3 is another clock in the group, so only four distinct signals are needed for these four clocks and their complements, yet two new clock signals will be needed for ϕ_{1F} and its complement.
- The implementation in this document only permits inversion during one of the four clock phases. If the number of phases that could support inversion were increased, the system would slow further and require even more clocks.

Diagrammatic layout

Each stage in the description of 2LAL in published papers[1, 2] comprises two pass gates in a diagonal configuration, as shown in Fig. 2a. To better represent the enhancements in this document, Fig. 2b rearranges the circuit so pass gates driven by the same clock phase are vertically stacked.

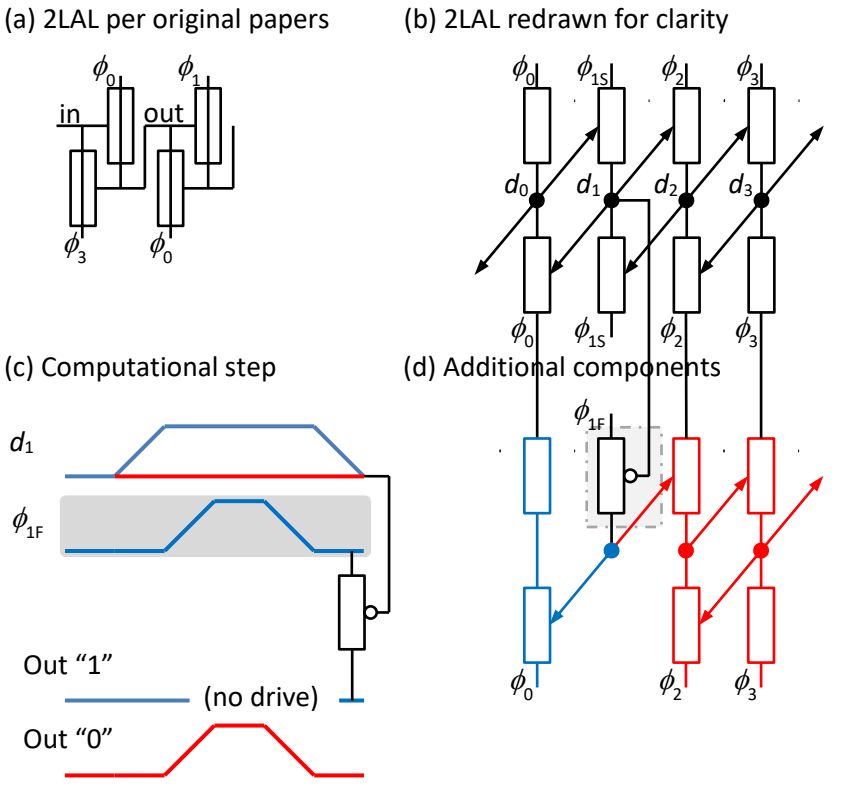

Fig. 2. (a) The orignal publications for 2LAL use an offset geometry, which would confuse this exposition. (b) 2LAL redrawn so transmission gates with the same clock phases are stacked on top of each other. This leads to a geometry where each stage’s interaction with others is readily apparent. (c) An additional stage computes the logical complement of the original signal. (d) The red pass gates create an additional data stream with the logical complement while the blue pass gates “uncompute” a stream containing the logical complement, recovering signal energy. The red and blue structures would not appear on the same gate (see text).

Inverter

Data signal d_1 in Fig. 2c shows a red DC voltage for a 0 value and a blue pulse for a 1 value. The inversion of d_1 should be a pulse if input is the red DC signal and a low DC output value if the input is a pulse. This can be accomplished with the inverting pass gate and the ϕ_{1F} clock in Fig. 2c, as follows:

If the pass gate’s inverting input is ground, the pass gate drives the ϕ_{1F} clock as a data signal, which is the desired behavior.

If the pass gate’s inverting input is the d_1 signal, which is the same shape as ϕ_{1S}, the pass gate’s output drive will weaken when d_1 leaves ground, becoming a reliably high impedance when the d_1 signal reaches the positive supply voltage. Inspection of the graph shows that the pass gate’s upper input will stay at ground for the entire time the drive weakens. As a result, the pass gate blocks d_1 entirely, leaving the natural circuit capacitance to hold the output at ground, representing a 0 value. Likewise for the strengthening at the end of the ϕ_{1S} pulse. This is the desired behavior.

The red pass gates in Fig. 2d illustrate the construction of a new inverted signal and two additional buffer stages for that signal. To avoid excessive dissipation, the new signal must be uncomputed eventually, which is the purpose of the blue pass gates. The red and blue pass gates are driven by the same signal in Fig. 2d for convenience of illustration, but sensible circuits would not use both the red and blue pass gates on the same signal.

Generalization to Other Gates

Fig. 3a repeats the specific inversion logic in Fig. 2d for reference, with Fig. 3b extending it to an XOR gate. Since more than the single input d_1 is needed, the extension adds a second input δ_1 that would be created by a replica of the shift register in Fig. 2b. The generalization of Fig. 3b is an and-or tree based on a charge recovery logic (CRL)[4] circuit on inputs d_1, δ_1… that gates ϕ_{1F} to the output. This generalization will work for any number of inputs in principle, but scalability has not been analyzed.

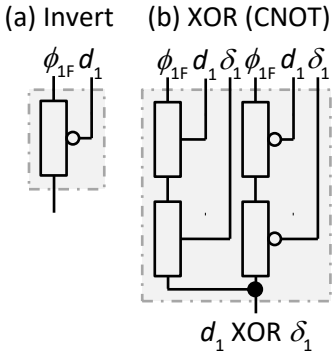

Fig. 3. General gates. (a) Inverter, passing the "0" pulse when d_1 is low. (b) XOR, or CNOT when in association with a saved signal. If d_1 and δ_1 are both 0, the two pass gates on the left will become conductive, passing the pulse. The pulse will also pass if they are both 1.

Conclusions

It would be interesting to extend the analysis of synthesized retractile cascade and fully-pipelined 2LAL circuits in Ref. 2 to include this 2LAL variant.

This 2LAL variant was created because the quantum computer control application needed memory. A retractile cascade is a logic circuit with no storage capability and a fully-pipelined 2LAL shift register requires four rails, which is tremendously inefficient in terms of device count. So, this work's initial goal was a circuit to generate the third and fourth rails from the first two, thus enabling logic on data stored in a shift register. Decomputing the third and fourth rails was implied.

The somewhat oblique conclusion from the combination of this work and Ref. 2 is that some 2LAL data formats impose a 2× overhead on memory size unless bidirectional 2-rail to 4-rail conversion is available. This document presents the converter, but it requires additional clocks—which are a global resource.

In further comparison to the analysis of fully-pipelined 2LAL in Ref. 2, this 2LAL variant requires six phases instead of four, affecting both clock signal count and speed. As compensation for clock issues, the need for the third or fourth rails goes away, which could cut circuit complexity in half. However, the authors of Ref. 2 employed some circuit reorganization that would seem reduce this 2× advantage of the 2LAL variant in some cases while never increasing the advantage.

This 2LAL variant can only apply its inversion feature to one of the four computational phases. This would require delaying gates that might be most optimally placed in one of the three excluded phases. However, there are effects in the other direction. While there has been no deep analysis of the following points, perhaps the ϕ_1 extension that enables inversion could be applied to ϕ_3 as well, reducing delay at the price of more clocks. Perhaps noninverting gates already covered in Ref. 2 could be productively inserted in some of the excluded phases. These effects will be circuit dependent, and may become interesting further work.

Finally, it appears that the best gate set will depend on the prevalence of memory in the overall system, or some other way of expressing the same concern.

Clarification of Terminology

This document raises a question about the definition of 2LAL. Frank[1, 2] stated via email "going forwards […] we should use 2LAL as a general term for any 2-level adiabatic logic, while CRL can perhaps be considered to refer to the more limited special case described by Younis and Knight back in '93." Thus, this document describes an unnamed variant, or special case, of 2LAL for quantum computer control.

References

[1] 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.

[2] Zulehner, Alwin, Michael P. Frank, and Robert Wille. "Design automation for adiabatic circuits." *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 2019.

[3] E. DeBenedictis, *New Design Principles for Cold, Scalable Electronics*. Technical report EPD001, http://www.zettaflops.org/CATC/.

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