Inverter for Dual-Rail 2LAL

Technical report ZF014, v2, May 18, 2025

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

Overview

This technical report describes a circuit for inverting dual-rail 2LAL signals, thus making dual-rail 2LAL a universal logic family.

The dual-rail version of 2LAL cannot invert a signal with just the circuit design techniques disclosed in the literature. Therefore, 2LAL advocates create the effect of inversion by creating a copy of a dual-rail circuit with all the signals inverted, effectively creating a quad-rail system. Swapping a signal with its inverse in the copy creates the effect of inversion, but with the disadvantages of doubling the size of the circuit.

The design technique in the literature uses signals in tick i (e. g. d_iT, d_iC), to compute data in the next tick (e. g. $d_{i+1}T$, $d_{i+1}C$). With the previous discussion as background, the circuit in this technical report uses signals in both tick i (d_iT, d_iC) and tick i+2 ($d_{i+2}T$, $d_{i+2}C$) to compute the inverse during tick i+1 ($-d_{i+1}T$, $-d_{i+1}C$). This apparently new circuit design method mixes data in different phases.

The inverter presented here has some other interesting properties, for example some outputs "auto decompute."

Background

Athas published the earliest paper with a 2LAL buffer stage [1], although without giving the circuit a name. The author finds no evidence that Athas had identified gates. Frank claims to have reinvented and named 2LAL [2] [3], including AND and OR gates. For inversion, Frank "upgraded" the circuit to quad rail and used signal swapping.

In an iMessage, Frank says he "realized when working on S2LAL in 2020 that in fact 2LAL was almost exactly equivalent to the original CRL logic of Younis & Knight 1993."

Fig. 1a illustrates the 2LAL clocking, comprising four clocks φ_0-φ_3 that are essentially ramped waves in quadrature with a range from GND to V_p.

The 2LAL power-clocks look like sine waves in quadrature, leading one to wonder if the sine waves as shown in Fig. 1b would work. The author analyzed sine wave power-clocks and found that they work about as well. However, the sine function in mathematics has a universal starting point of $\sin(0) = 0$, but $\varphi_i \neq \sin(t + i\pi/2)$ (with voltage scaling) the clock will have a 45˚ phase difference from Frank's 2LAL power-clocks (Fig. 1b). This document applies to both the power-clocks in Fig. 1a and b.

2LAL data signals are denoted d_iT and d_iC, T for True and C for Complement, where $d_iC = V_p - d_iT$ (assuming GND = 0 V). Waveform d_iT has the geometric shape of φ_i if it is a logical 1 and GND if logical zero. Note that d_iT is a return to zero (RZ) signal, so a sequence of ones looks like square wave. Thus, d_iC is a "return to V_p" signal. Swapping the true and complement rails does not invert the signal, but instead breaks the electrical protocol.

Fig. 1c shows two basic 2LAL buffer stages, where $j = i$+1 and $k = i$+2. In Frank's notation, the rectangles are back-to-back MOSFETs that form a transmission gate. The transistor sources and drains are on the short ends and the gates are on the long ends. A wire attaching to a long side of the rectangle attaches to the nFET, so the signal on the wire logically enables the nFET gate. An inversion bubble on a control signal swaps the connections to the pFET and nFET gates.

Frank's 2LAL is implicitly dual rail, so he duplicates each diagram, replacing each φ_i with φ_{i+2} and swapping the T and C suffixes on data signals.

The description above is terse and does not include logic gates; the reader can see ref. [2] [3] for more information.

Dual-rail inverter

The objective is to produce both the logical and electrical inverse of the signal stream on d_jT, which will be $-d_jC$. We will worry about $-d_jT$ later. Each one-bit on signal d_jT has the same waveform as clock φ_j, and comprises DC GND otherwise. By symmetry d_jC and $-d_jC$ have the same waveform as clock φ_l, but emit a DC value in different places. The strategy is to manufacture $-d_jC$ starting with a continuous stream of one-bits – which is just power-clock φ_l – and block the bits that should be DC (but the DC value will be V_p not GND). Just like the 2LAL circuitry in Fig. 1b, we may be able to leave the output floating for the DC part (we will have to see).

Fig. 2a shows a "half inverter" called INV with power-clocks φ_i, φ_j, φ_k, and φ_l, where $j = i+1$, $k = i+2$, and $l = i+3$. The circuit's input is d_iT on the left and its output is the inverse $-d_{i+1}C = -d_jC$ with one tick delay on the right with one tick delay. The INV circuit does not destroy the input signal, but instead transmits that signal as d_kT with a two tick delay.

The graphics in Fig. 2a collectively define the INV circuit, but the blue graphics are identical to the buffer in Fig. 1c. The green graphics are an optional clamp (which will be needed for output signals that draw static current)

The reader will see φ_l driving a sequence of three transmission gates enabled respectively by (taking the inversion bubbles into account) d_iC, d_jC, and d_kC, Simplistically, d_jC will block φ_l on a one-bit, as desired. While inspiring, just using d_jC would lead to half-on transmission gates transmitting weak signals at the onset of φ_l and again after the bit. So, we need to "widen" the block time on both sides. This can be accomplished by additionally gating φ_l with d_iC and d_kC (in the sense of an AND, where any of the inputs can block the signal).

The reader will see φ_l driving a sequence of three transmission gates enabled respectively by (taking the inversion bubbles into account) d_iC, d_jC, and d_kC, Simplistically, d_jC will block φ_l on a one-bit, as desired. While inspiring, just using d_jC would lead to half-on transmission gates transmitting weak signals at the onset of φ_l and again after the bit. So, we need to "widen" the block time on both sides. This can be accomplished by additionally gating φ_l with d_iC and d_kC (in the sense of an AND, where any of the inputs can block the signal).

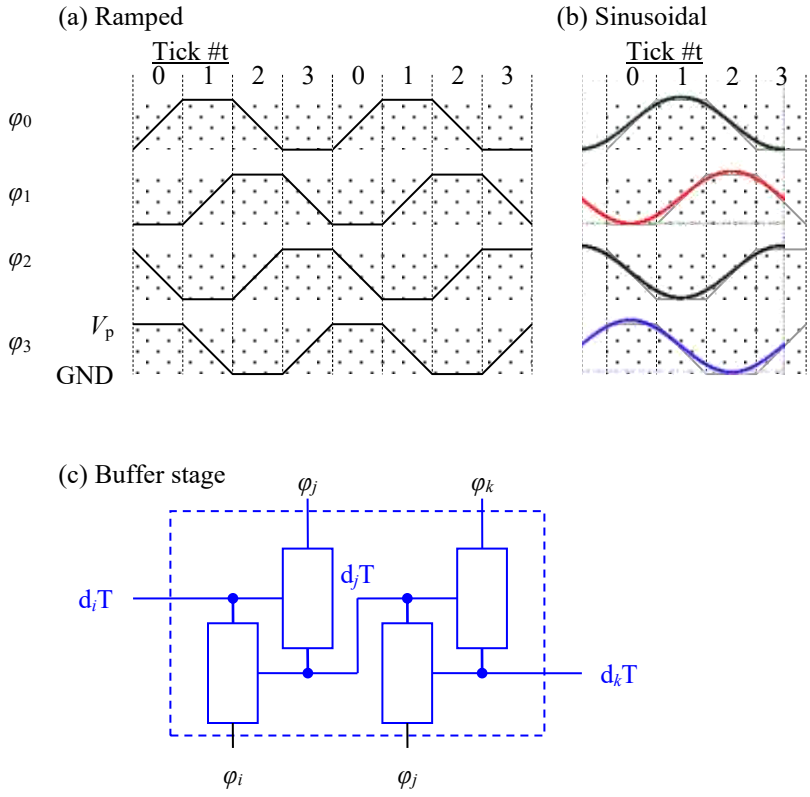

Fig. 1. (a) Ramped power-clocks, (b) sinusoidal power-clocks. (c) Buffer stage.

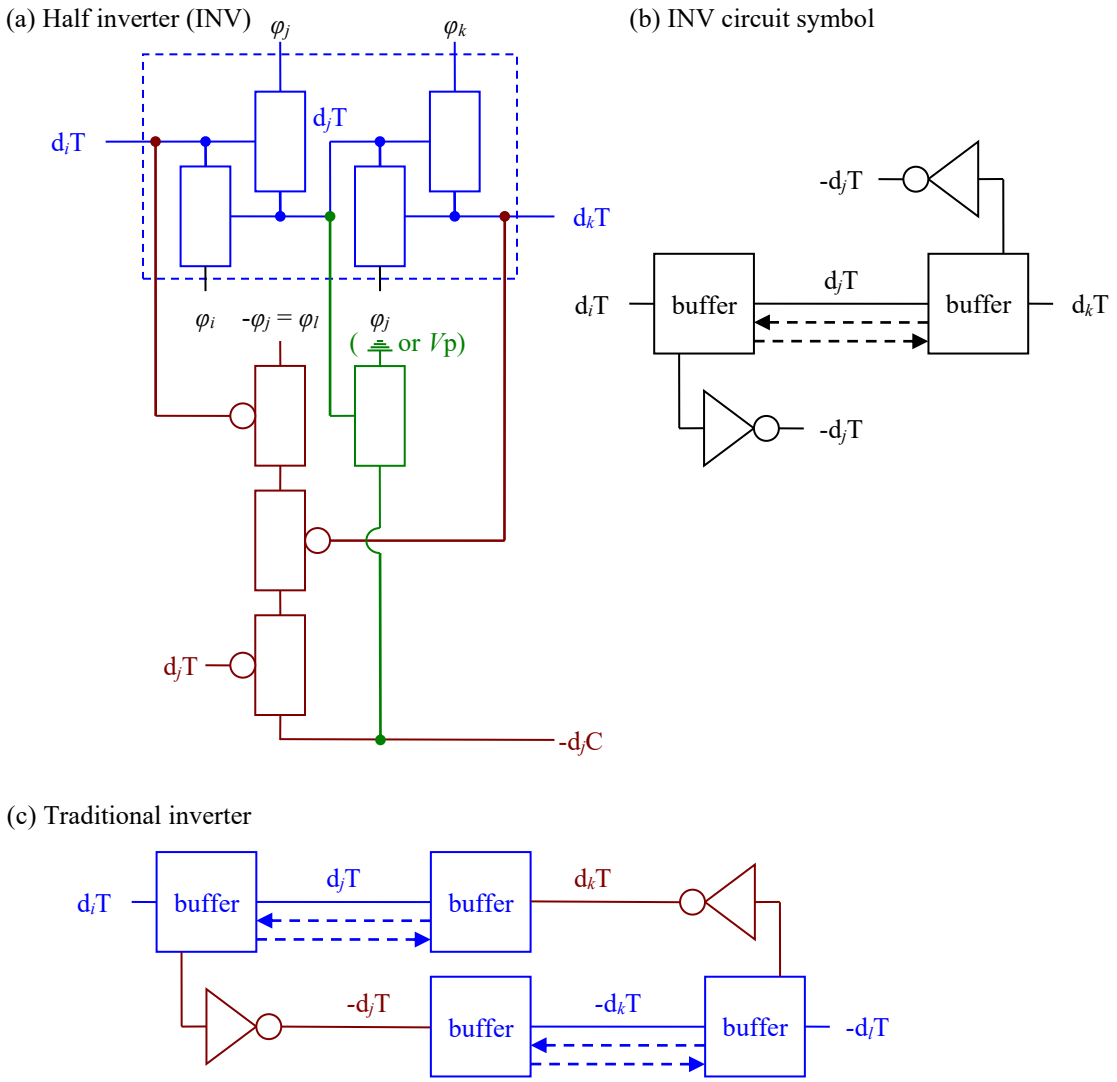

Fig. 2. 2LAL inverter. (a) Circuit diagram comprising two standard 2LAL buffer stages (blue) plus basic inversion circuitry above (red) and an optional clamp (green). Label $-d_jT$ is the inverted output, but note that it occurs earlier (tick j) than the rightmost output in the diagram (tick k). (b) Two copies of the reversible gate diagram, including backwards inverters that recover energy. Note that the circuit does not change when reversed. Note also that the inverted output does not require energy recovery unless connected to something (i.e. it can be ignored). (c) A traditional inverter, which computes the inverse and then decomputes the original signal. Clock phases are (in sequential order) i, j, k, and l; d_iT and d_iC denote true and complement signals in phase i; diagram is for one of two rails.

At this point, we have constructed $-d_jC$; just flip φ_l to φ_j to construct $-d_jT$.

Up to this point, we have been relying on the fact that driving an output to GND or V_p and then just leaving it there will hold the DC value. If there is a static current load, the claim shown in green will hold it in position.

Fig. 2b illustrates the INV circuit symbol. The circuit in Fig. 2a has the layout geometry of the bottom 2/3rd of Fig. 2b, namely it receives a signal from the left and transmits it to the right (a) inverted and delayed by one tick and (b) delayed by two ticks. As typical for reversible logic symbols, vertically mirroring a circuit causes it to decompute (although the reader should note that the INV circuit is symmetric around the

centerline). So, mirroring the bottom 2/3rd of Fig. 2b produces the top 2/3rd of Fig. 2b – and the buffer stages merge. Note that the outputs of the two inverter symbols are shorted together.

What about "decomputing" the values on inverter signals if unused? The answer is "ignore decomputing" because the outputs to the inverters "auto decompute." Of course, after passing inverter output through a 2LAL buffer, the data needs to be decomputed (due to the rules for standard buffers).

So, INV converts dual rail to quad rail with twelve or sixteen transistors (depending on the presence or absence of a clamp). Mirroring INV converts quad rail back to dual rail – of course noting that the mirroring the INV circuit has no effect.

Fig. 2c is closer to a traditional inverter, or through an extension, a CNOT or Toffoli gate. Basically, Fig. 2c comprises to INV circuits pointed at each other – with one flipped up side down. The standard interpretation of the symbols applies: The circuit turns the input signal into quad rail and then decomputes the original signal, leaving the inverse.

Fig. 3 is a more sophisticated inverter, one that adds three ticks of delay instead of two. So, what is the advantage? If the central block connects the wires straight through, the circuit is an inverter, but if the central block swaps the wires a shown by the diagonal dotted lines, the circuit is a non-inverting buffer. As drawn, Fig. 3 is a CNOT with a manual control.

(a) Alternative inverter, extendible to CNOT or Toffoli

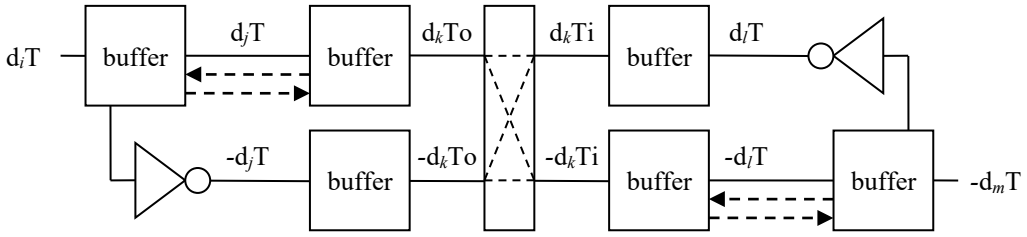

Fig. 3. Alternative inverter. (a) Two mirrored copies of the circuit in Fig. 2 with the inverted signal buffered. If the central rectangle can swap the output (suffix o) and input (suffix i) signals controllably, the circuit would be a CNOT or Toffoli based on the control.

We can replace the central block with transmission gates that either connect the inputs straight through or swap them based on a voltage input. This would make Fig. 3 a CNOT controlled by a voltage. We leave the reader to figure out how to generate the control voltage from a 2LAL signal.

Conclusions and Future Work

As far as the author knows, Athas invented the (unnamed) shift register [1], Frank coined the name 2LAL and expanded it to a quad-rail universal logic family [2], and this document demonstrates that 2LAL can be a two-rail logic family as well.

The inverters will help most for data storage. Without inverters, a system requiring universal logic and a lot of data storage would need to store the data in the quad-rail representation. With inverters, data could be:

(a) computed in quad-rail representation

(b) converted to dual-rail representation for storage, which cuts the component count in half

(c) converted back to quad-rail representation when needed for logic.

As far as the author knows, the circuit obeys 2LAL design rules, but is not within the "template" for gate design in the literature [2]. As such, it represents a new design method. If review of this document reveals interest in the method, the author might write it up in a later version of this document. The method has a lot of other uses.

There is a loose end for future work. The three transmission gates in Fig. 2a do not work as well as desired. The transmission gate turn on completely and off completely. However, two transmission gates in series have an intermediate node that can contain change. The illustrated circuit has three transmission gates in series, which has to intermediate nodes. If the circuit blocks a large number of bits in sequence, this small amount of charge will move non-adiabatically between the terminal, eventually eroding signal integrity. Some people might see this as a bug in the circuit inverter circuit. On the other hand, the standard 2LAL buffer stage (Fig. 1c) has the same problem with a long sequence of zero-bits. So, this undesirable anomaly does not seem any worse than anomalies more even more deeply baked in the circuit family.

However, the author has two mitigations:

The three transmission gates can be three nFETs in series and three pFETs in series, with no bridges between the intermediate points. This helps.

There are multiple independent gate transistors (MIGFETS) that provide AND functionality in two gates without the intermediate charge storage location. These devices had been hypothetical, but the can not be manufactured with some FINFET processes. Currently manufacturable devices do not have a high enough on-off ratio to be helpful so far.

References

[1] Athas, William C. "Energy-recovery CMOS." *Low Power Design Methodologies*. Boston, MA: Springer US, 1996.

[2] Frank, Michael P. "*Adiabatic Circuits: A Tutorial Introduction*." https://www.osti.gov/servlets/purl/1459779.

[3] Anantharam, Venkiteswaran, et al. "Driving Fully-Adiabatic Logic Circuits Using Custom High-Q MEMS Resonators." *ESA/VLSI*. 2004. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7c8d654ce333d5b032af809c4c7770a61fb3add9

Appendix: Ngspice code

The following ngspice code defines the circuits in this document precisely. The code is incomplete in the sense of lacking clock generation and plotting.

```
.param cwire=1e-13 cand=.5e-13

.SUBCKT PASS D GT GC S nsub psub                          $ Pass gate. Args: Drain GateT/C Source nsub psub
xM10 D GT S nsub nFET n=1 m=1                             $ pass gate
xM11 D GC S psub pFET n=1 m=1
.ENDS PASS

.SUBCKT PASS2 D GT GC HT HC S I J nsub psub               $ Double pass gate. Args: Drain GateT/C Source nsub psub
xM10 D GT I nsub nFET n=1 m=1                             $ pass gate
xM20 I HT S nsub nFET n=1 m=1                             $ pass gate
xM11 D GC J psub pFET n=1 m=1
xM21 J HC S psub pFET n=1 m=1
C1 I nsub 'cand'                                          $ load capacitors on inputs, to simulate wiring
C2 J nsub 'cand'                                          $ load capacitors on inputs, to simulate wiring
R1 I J 5e7                                                $ can either have the transistor chains shorted or 5e7
resistor
.ENDS PASS2

.SUBCKT PASS3 D GT GC HT HC IT IC S I J K L nsub psub      $ Triple pass gate. Args: Drain GateT/C Source nsub psub
xM10 D GT I nsub nFET n=1 m=1                             $ pass gate
xM20 I HT J nsub nFET n=1 m=1                             $ pass gate
xM30 J IT S nsub nFET n=1 m=1                             $ pass gate
xM11 D GC K psub pFET n=1 m=1
xM21 K HC L psub pFET n=1 m=1
xM31 L IC S psub pFET n=1 m=1
C1 I nsub 'cand/2'                                        $ load capacitors on inputs, to simulate wiring
C2 J nsub 'cand/2'                                        $ load capacitors on inputs, to simulate wiring
C3 K nsub 'cand/2'                                        $ load capacitors on inputs, to simulate wiring
C4 L nsub 'cand/2'                                        $ load capacitors on inputs, to simulate wiring
R1 I J 5e7                                                $ can have the transistor chains shorted or 5e7 resistor
R2 K L 5e7                                                $ can have the transistor chains shorted or 5e7 resistor
.ENDS PASS3

.SUBCKT PHAS2 iiT iiC ooT ooC L0 L1 L2 L3 nsub psub       $ One phase of the 2LAL shift register. Args: iiT/C ooT/C
* clocks/power supplies
X1 iiT ooT ooC L0 nsub psub PASS
X2 iiC ooT ooC L2 nsub psub PASS
X3 ooT iiT iiC L1 nsub psub PASS
```

```
X4 ooC iiT iiC L3 nsub psub PASS
C1 iiT nsub 1e-13                                       $ load capacitors on inputs, to simulate wiring
C2 iiC nsub 1e-13
.ENDS PHAS2

.SUBCKT DELA2 d0T d0C d4T d4C                           $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ L0 L1 L2 L3 nsub psub                                 $ clocks/power supplies
+ ini=0
.ic V(d0T)={gg} V(d1T) = {gg} V(d2T) = {   ini} V(d3T) = {   ini}
.ic V(d0C)={vv} V(d1C) = {vv} V(d2C) = {vv-ini} V(d3C) = {vv-ini}
X1 d0T d0C d1T d1C L0 L1 L2 L3 nsub psub PHAS2
X2 d1T d1C d2T d2C L1 L2 L3 L0 nsub psub PHAS2
X3 d2T d2C d3T d3C L2 L3 L0 L1 nsub psub PHAS2
X4 d3T d3C d4T d4C L3 L0 L1 L2 nsub psub PHAS2
.ENDS DELA2

.SUBCKT INV diT diC djT djC ijT ijC dkT dkC x1 x2 x3 x4 Li Lj Lk Ll nsub psub
X0 diT diC djT djC Li Lj Lk Ll nsub psub PHAS2
X1 djT djC dkT dkC Lj Lk Ll Li nsub psub PHAS2
X2 Ll diC diT dkC dkT ijC nt1 nt2 nsub psub PASS2       $ create the inverted signal ^ pulse from this clock
X6 Lj diC diT dkC dkT ijT ny1 ny2 nsub psub PASS2       $ create the inverted signal v pulse from this clock

r101 x1 djT 1e6                                         $ connections needed for plots
r102 x2 djC 1e6
r103 x3 ijT 1e6
r104 x4 ijC 1e6
.ends INV

.SUBCKT INV3 diT diC djT djC ijT ijC dkT dkC x1 x2 x3 x4 Li Lj Lk Ll nsub psub
X0 diT diC djT djC Li Lj Lk Ll nsub psub PHAS2
X1 djT djC dkT dkC Lj Lk Ll Li nsub psub PHAS2
X2 Ll diC diT djC djT dkC dkT ijC nt1 nt2 xx xy nsub psub PASS3 $ create the inverted signal ^ pulse from this clock
X6 Lj diC diT djC djT dkC dkT ijT ny1 ny2 wx wy nsub psub PASS3 $ create the inverted signal v pulse from this clock

r101 x1 djT 1e6                                         $ connections needed for plots
r102 x2 djC 1e6
r103 x3 ijT 1e6
r104 x4 ijC 1e6
.ends INV3

* the following circuit is essentially a three-stage inverter with a buffer to round it up to a full cycle
.SUBCKT INVERT d0T d0C i1T i1C i2T i2C i3T i3C i4T i4C   $ Four phases that just delay. Args: 5*{ data<n>T/C }
+ x1 x2 x3 x4                                           $ inverted output
+ L0 L1 L2 L3 nsub psub                                 $ clocks/power supplies
+ ini=0
.ic V(d0T)={gg} V(d1T) = {gg} V(d2T) = { ini}
.ic V(d0C)={vv} V(d1C) = {vv} V(d2C) = {vv-ini}
.ic             V(i1T) = {vv} V(i2T) = {vv-ini} V(i3T) = {vv-ini}
.ic             V(i1C) = {gg} V(i2C) = {ini}    V(i3C) = {ini}
X1 d0T d0C d1T d1C i1T i1C         d2T d2C              x5 x6 x7 x8 L0 L1 L2 L3 nsub psub INV3
X2                i1T i1C i2T i2C d2T d2C i3T i3C      ww xx yy zz L1 L2 L3 L0 nsub psub INV3
*  0000000 111111111111111 222222222222222 3333333 4444444 ttttttttttt LLLLLLLLLL nsub psub
X3                                        i3T i3C i4T i4C             L3 L0 L1 L2 nsub psub PHAS2

.if (1)                                                 $ connections needed for plots
r101 x1 d0T 1e6
r102 x2 d0C 1e6
r103 x3 i4T 1e6
r104 x4 i4C 1e6
.endif
.ends INVERT

* the following circuit is a two-stage inverter followed by uncomputing so it inverts
* inversion is the normal mode, so the input is d0T d0C and output is i4T i4C
* it can be manually swapped to be a buffer -- like a CNOT with a manual control
.SUBCKT CNOT d0T d0C i1T i1C i2To i2Co i3T i3C i4T i4C   $ Four phases that just delay. Args: 5*{ data<n>T/C }
+ x1 x2 x3 x4                                           $ inverted output
+ L0 L1 L2 L3 nsub psub                                 $ clocks/power supplies
+ ini=0
.ic V(d0T)={gg} V(d1T) = {gg} V(d2Ti) = { ini}   V(d3T) = { ini}
.ic V(d0C)={vv} V(d1C) = {vv} V(d2Ci) = {vv-ini} V(d3C) = {vv-ini}
.ic             V(i1T) = {vv} V(i2Ti) = {vv-ini} V(i3T) = {vv-ini}
.ic             V(i1C) = {gg} V(i2Ci) = {ini}    V(i3C) = {ini}
X1 d0T d0C d1T d1C i1T i1C d2To d2Co                      x5 x6 x7 x8 L0 L1 L2 L3 nsub psub INV3
X2                i1T i1C i2To i2Co                          L1 L2 L3 L0 nsub psub PHAS2
*  0000000 111111111111111 222222222 333333333333333 4444444 ttttttttttt LLLLLLLLLL nsub psub
x3                        d2Ti d2Ci         d3T d3C             L2 L3 L0 L1 nsub psub PHAS2
X4                        i2Ti i2Ci i3T i3C d3T d3C i4T i4C ww xx yy zz L2 L3 L0 L1 nsub psub INV3

.if (1)                                                 $ A CNOT with electrically controlled swap
r10 rcT nsub 0                                          $ electrical CNOT control rcT gg for non-inverting buffer
+                                                       $ and vv for CNOT invert
r11 rcC psub 0
X100 d2Ti rcT rcC d2To nsub psub PASS                   $ controlled swap network -- invert (straight through)
X101 d2Ci rcT rcC d2Co nsub psub PASS
X102 i2Ti rcT rcC i2To nsub psub PASS
X103 i2Ci rcT rcC i2Co nsub psub PASS
X104 d2Ti rcC rcT i2To nsub psub PASS                   $ non-invering buffer (swap)
X105 d2Ci rcC rcT i2Co nsub psub PASS
X106 i2Ti rcC rcT d2To nsub psub PASS
X107 i2Ci rcC rcT d2Co nsub psub PASS
.elseif (1)                                             $ A CNOT with manual swap set to straight through
+                                                       $(inverter)
r0 d2Ti d2To 0                                          $ inverter
r1 d2Ci d2Co 0
```

```
r2 i2Ti i2To 0
r3 i2Ci i2Co 0
.else                                                   $ A CNOT with manual swap set to interchange (non-inverting
+                                                       $ buffer)
r0 d2Ti i2To 0                                          $ non inverting buffer
r1 d2Ci i2Co 0
r2 i2Ti d2To 0
r3 i2Ci d2Co 0
.endif                                                  $ A
.if (0)                                                 $ B plot setup #1
r101 x1 d2To 1e6
r102 x2 d2Co 1e6
r103 x3 i2To 1e6
r104 x4 i2Co 1e6
.elseif (1)                                             $ B plot setup #2
r101 x1 d0T 1e6
r102 x2 d0C 1e6
r103 x3 i4T 1e6
r104 x4 i4C 1e6
.else                                                   $ B NRZ output, with a plot setup

.if (0)                                                 $ C historical version of NRZ circuit
X200 w d0T d0C ww nsub psub PASS                        $ pretty good solution for now
X201 ww i4C i4T L1 nsub psub PASS
X211 ww i4T i4C psub nsub psub PASS

X202 w i4T i4C wx nsub psub PASS
X203 wx d0C d0T L3 nsub psub PASS

X205 w d2To d2Co psub nsub psub PASS

r101 x1 d2To 1e6 $ d0T 1e6
r102 x2 L1 0e6 $ d2To 1e6
r103 x3 w 1e6 $i3T 1e6
r104 x4 i4T 1e6

.else                                                   $ C latest NRZ circuit here; ones above can be deleted

X200 w d0T d0C i4C i4T L1 I1 J1 nsub psub PASS2         $ both energies have suffix E-14 before adding cwire
X201 w i4T i4C d0C d0T L3 I2 J2 nsub psub PASS2
*C1 w nsub 'cwire'                                       $ load capacitors on wires, to simulate wiring

.if (1)                                                 $ D  baseline; seems to work; can delete the nFET
*X202 w d2To d2Co psub nsub psub PASS
xM202 w d2Co psub psub pFET n=1 m=1

.elseif (1)                                             $ D more intelligent, but does not seem to work as well
X202 w i4T i4C d0T d0C psub I3 J3 nsub psub PASS2       $ PASS2 module, but nfets unnecessary
.else
xM202 w i4C x psub pFET n=1 m=1                         $ D just the pFETs
xM203 x d0C psub psub pFET n=1 m=1
C0 x nsub 'cand'                                        $ for compatibility
.endif                                                  $ D

r1 x1 w 1e6 $ d0T 1e6                                   $ for plotting only
r2 x2 d2To 1e6 $ d2To 1e6
r3 x3 d2To 1e6 $i3T 1e6
r4 x4 d2Co 1e6


.endif                                                  $ C
.endif                                                  $ B
.ends CNOT
```