Inversion for S2LAL

Technical note (report) ZF004 v1.02, December 15, 2020
Erik P. DeBenedictis
Zettaflops, LLC, Albuquerque, NM 87112
erikdebenedictis@zettaflops.org

Abstract

This technical note extends the recently introduced S2LAL[1] reversible logic family, which is a static version of the 2LAL family.[2] I created an inversion capability for 2LAL in a previous report,[3] and I do the same here for S2LAL. For S2LAL, inversion allows dual rail instead of quad rail, cutting the number of transistors and wires in half for a given function. This note includes two dual-rail circuits. One variant can co-exist in a S2LAL circuit. The second variant is not compatible with S2LAL because it changes the voltage polarity on one of the rails, but is more effective in other ways. Simulation source code is included in this document.

S2LAL Inversion, Variant 1

For 2LAL, I created a 2-level cascade by modifying the clock waveforms. This enabling inversion but extending the cycle from 4 to 6 ramps reduced throughput significantly. However, S2LAL already contains an adequate 2-level cascade in the sense that ϕ_2 fits entirely within the flat top of S_1, so there is no loss of throughput. In Fig. 1, I repeat the method of using data signal S_1 in the 0 state to gate clock ϕ_2, which starts out with the same shape as a 1 signal.

During forward clocking, each stage is expected to drive its output signal (S_2 or T_2) when ϕ_1 is high, making the transition from 0 to 1 (if there is to be a transition) at the same time as the ϕ_2 clock. The stage is expected to be tri-stated when ϕ_1 is low. The circuit in Fig. 1 complies, thus creating $T_2 = -S_2$. As in [3], this type of circuit creates a new data stream, which runs backwards naturally and can be mirrored for decomputation.

The intermediate signal Q_2 is created by using S_1 to select between ϕ_2 and ground. The timing diagram shows that the ϕ_2 clock transition occurs when the S_1 signal is stable, so Q_2 is a low-impedance voltage source and can deliver and recover energy.

The red circuitry in Fig. 1 is shorthand for a pair of circuits with complementary voltages. All indices may be shifted, mod 8, allowing inversion to occur in any phase. The pass gate to a fixed voltage can sometimes be replaced by a single transistor, as in [1].

The circuit naturally extends to support for arbitrary inverted inputs. The extension is simply to allow Q_2 and T_2 to replace F_2 in Fig. 1a, yet using a mirrored version of the red circuit for R_2 to decompute the function containing inversion.

In fact, the red circuitry in Fig. 1a can be replaced by the two-input gates from [1, Figs. 8-9] as shown in Fig. 1c, yet also allowing complemented inputs.

The enhancement in Fig. 1 is not exactly an inverter; it takes a stream of bits and creates a second stream with the logical complement of the bits. The circuit can likewise create a new stream with the AND or OR of two input streams, including any combination of input inversions. Extension to XOR and XNOR is left as a future project.

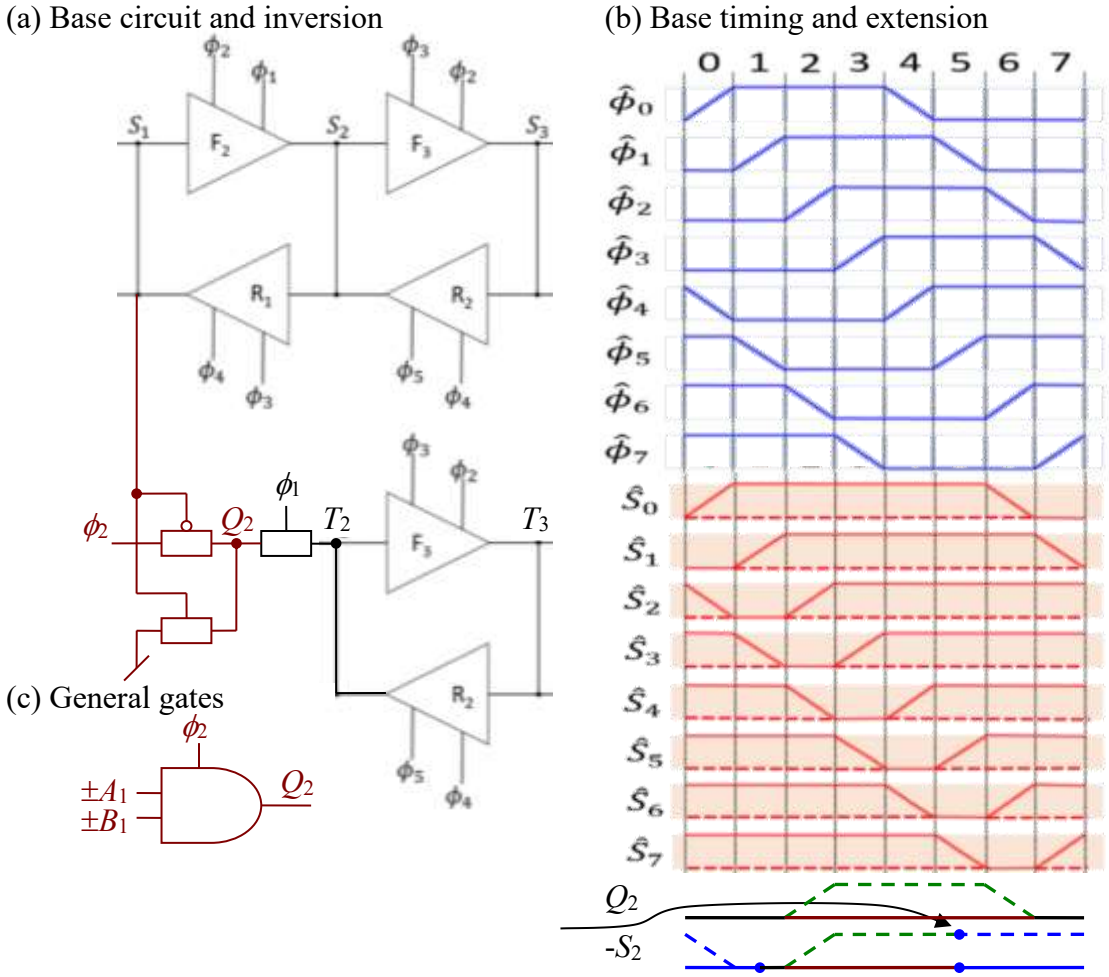


Fig. 1. S2LAL inversion. The (a) base circuit and (b) timing diagrams are copied from ref. 1, but I add three pass gates and the timing to invert the S_2 signal. The effect is not an inverter, but to launch an inverted stream $T_n = -S_n$. (c) Furthermore, the gates from [1, Figs. 8-9] will work even with inverted inputs.

S2LAL Inversion, Variant 2

Adiabatic circuits have been developed for computer security purposes that place a few very even load on the power supply, such as EE-SPFAL.[4] For background, there is a type of computer hardware attack called differential power analysis that attempts to figure out secret information by measuring changes in power supply current. If 0's consume less power than 1's, measuring the power supply current at a particular instant in time may yield information about a bit in a password. Repeating the measurement many times could yield an entire password.

The following situation with S2LAL's electrical signal format makes it vulnerable to differential power analysis.

S2LAL is naturally dual-rail, meaning each logical signal is represented by a signal on each of two wires. One signal represents a 0 by a V-to-ground pulse and the other signal represents 1 with a ground-to-V pulse. The S2LAL trace in Fig. 2a shows a sequence of three 1s followed by three 0s. The orange trace in Fig. 5a has a resting state of ground and is called "Q hat" ($\hat{Q}$, where the accent symbol is also called a circumflex) because the positive pulse representing 1 is peaked in the middle like a "hat." The same signal with a resting state of V is in black and called "Q cup" ($\check{Q}$, where the accent symbol is a caron or an inverted hat) because the voltage waveform is low in the middle. The logical inverse of Q is $-Q$, but in S2LAL it must be designated as $-\hat{Q}$ and $-\check{Q}$.

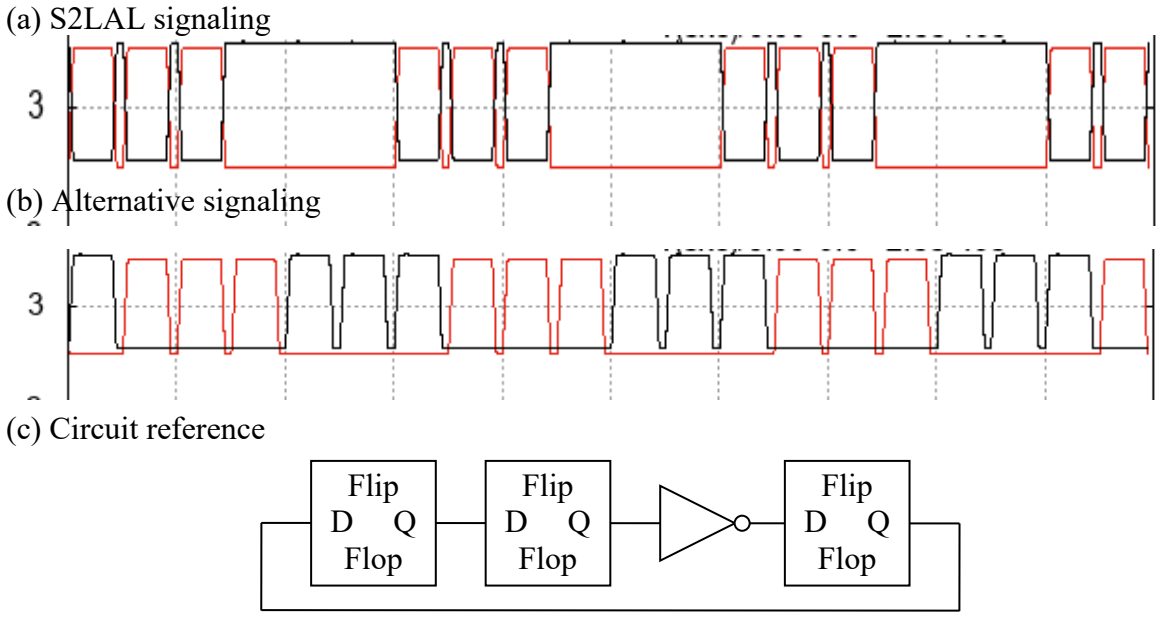


Fig. 2. (a) S2LAL signaling repeating sequence 111000… with $\hat{Q}$ and $\check{Q}$ (b) signaling 0111000… with Q and $-Q$. The latter may lead to a more even load. (c) Circuit reference.

Let us see if we can derive an alternative to S2LAL[1] that puts a more consistent load on the power supply. This will be a circuit that signals via two wires containing $\hat{Q}$ and $-\hat{Q}$. However, the circuit will not have any cups, so the hats would not be needed to distinguish the pulse polarity. Thus, we could describe the alternative logic family with no hats and no cups. The circuit's terminology could be simplified to use just Q and $-Q$.

The timing is illustrated in Fig. 2b. With the orange trace being Q and the black one being $-Q$, the trace shows the sequence 0111000111…

The alternative will put S2LAL into the category of circuits that are resistant to some side channel attacks, so the second variant circuit can be interpreted in the context of papers on that topic.[4]

Fig. 3 illustrates the alternative circuit. Fig. 3a is derived from [1, Fig. 4], but expanding the pass gate into its two transistors and labeling the input with the applicable phase. Without loss of generality, Fig. 3b is the same circuit processing the signal $-A$.

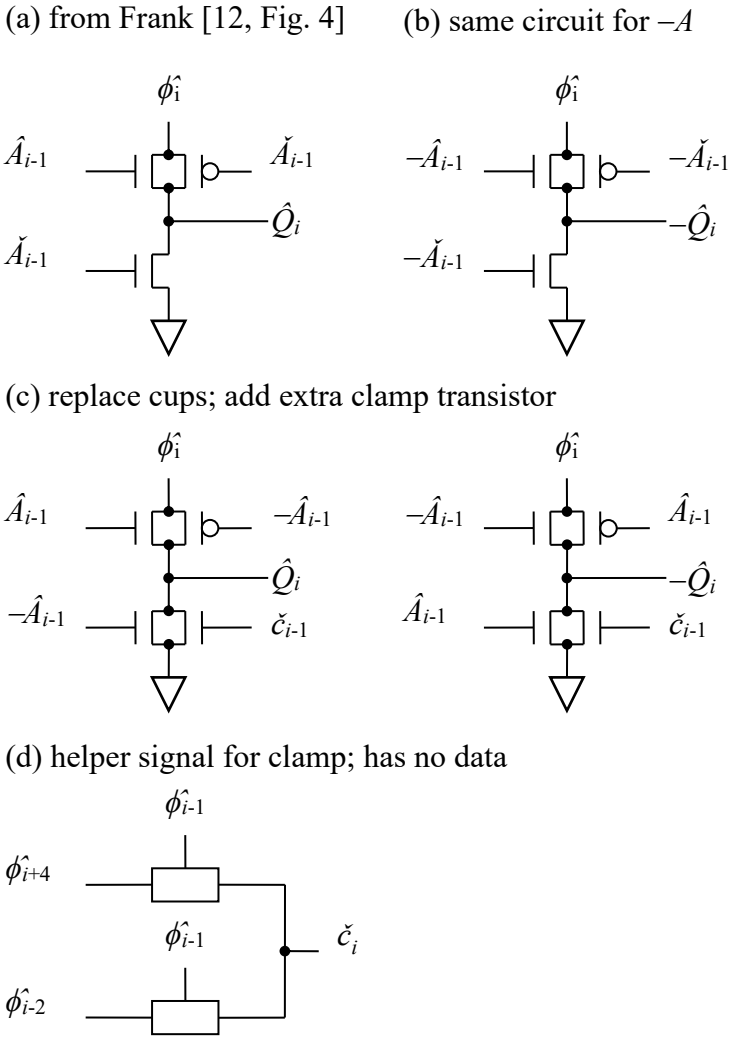


Fig. 3. (a) Unlatched adiabatic buffer from [1, Fig. 4], (b) same buffer for the negated signal (but not the cup symbol), (c) however, the incoming cup signals can be generated from the negated signals in the previous stage, provided that a helper signal $\check{c}_i$ is available. (d) The helper signal can be generated once in an entire circuit from available clocks.

The upper symbol $\hat{A}_{i-1}$ in Fig. 3a enables one transistor of the pass gate that gates the clock $\hat{\phi}_i$ to the output. We can replace this signal with $-\hat{A}_{i-1}$ because the alternative signal is stable at the correct level when needed to gate the clock, and is simply creating a redundant path to ground at other times.

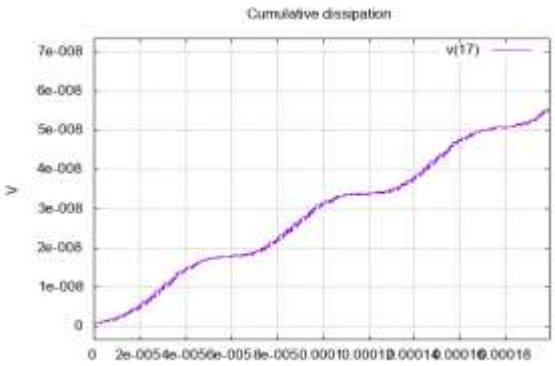
Likewise, the lower symbol $\check{A}_{i-1}$ in Fig. 3a enables the transistor that clamps the output to ground. Replacing that signal with $\hat{A}_{i-1}$ helps if the desired output is a 0, but will leave the output floating between output pulses. This leads us add a transistor gated by the signal $\check{c}_{i-1}$. This signal is the electrical inverse of the other signals for $\hat{A}_{i-1}$ where $A_{i-1} = 1$. Thus, the signal $\check{c}_{i-1}$ goes high during the period where the output needs to be clamped to ground, irrespective of whether the output is a 0 or 1.

Fig. 3d shows how to create the $\check{c}_k$ signal for stage k from four available clocks and four transistors. There would need to be eight variants of this circuit to create $\check{c}_k$ for $k = 0\ldots7$. However, the $\check{c}_k$'s are independent of data, so each such signal can be shared across multiple gates.

The alternative circuit has several advantages.

The two circuits in Fig. 3c differ only by swapping A's with $-A$'s. If the circuits are lain out near each other and with a similar interconnect pattern, the overall electrical characteristics will be the same irrespective of the data. This will smooth the load on the power supply. The code at the end of this document simulates the shift register in Fig. 2c for both S2LAL and the second variant circuit. The simulation output in Fig. 4 shows cumulative dissipation over time.

(a) S2LAL+inverter signaling



(b) Alternative signaling

Fig.4. Both plots represent cumulative power dissipation of a 3-bit cyclic shift register with a single inverter in the data stream. This will generate the pattern 000 100 110 111 011 001, going from all zeros to all ones and back. (a) S2LAL dissipation, showing variance as the number of 1s changes, and (b) the circuit in Fig. 3 where the total number of 0s and 1s does not change, so the dissipation is constant.

In variant 1, the circuit creates a new data stream with the inverted data. Strictly speaking, an inverter would require a second, mirrored copy of the circuit to decompute the original stream. However, inversion in variant 2 can be created just by swapping the wires. Variant 1 is less efficient due to requiring an extra stage, which increases transistor count and delay.

Variant 2 is not simpler than variant 1, but they are closer than first appears. As described here, each gate requires two more clamp transistors and the symbol $\hat{c}_k$. These extra costs are offset by

some simplifications: Only the clock $\hat{\phi_k}$ is required as a clock, although both $\hat{\phi_k}$ and $\tilde{\phi_k}$ are required for a pass gate. If $\tilde{\phi_k}$ can be driven to a higher voltage without causing transistor breakdown, the pass gates can be replaced by single pFET and nFET transistors driven by $\tilde{\phi_k}$. With appropriate design, neither additional clocks nor additional transistors would be required. Specifically, $\hat{\phi_k}$ would be unchanged but $\tilde{\phi_k}$ would be driven to a higher voltage. The two extra clamp transistors would be offset by removing a transistor from each of the pass gates. The signal $\hat{c_k}$ must be generated, but $\hat{c_k}$ is not dependent on data. It is like a standard clock that only goes to transistor gates. Thus, multiple $\hat{c_k}$'s may need to be generated, perhaps one per every 10 loads.

Conclusions

This note presents two variants on S2LAL that eliminate the need for quad-rail logic. Going from dual to quad rail cuts the complexity of the circuitry that will fit on a chip in half and increases power consumption. The second variant may also be of interest to the community that is researching the use of reversible logic for resistance to side channel attacks.

References

[1] Frank, Michael P., et al. "Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS," *2020 IEEE International Conference on Rebooting Computing*, online. At the time of this writing, the conference is over but the paper is not in IEEE Xplore, but see *arXiv preprint arXiv:2009.00448* (2020).

[2] 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.

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

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

Appendix: ngspice file

The file below includes both S2LAL with an inverter and the alternative implementation.

There are two implementation of a three-bit cyclic shift register with an inverter in the data stream. This will cause the three bits to go through the sequence 000 100 110 111 011 001 repeatedly. In other words, the register will go from all 0s to all 1s. This would be expected to cause a change in dissipation, which is plotted (Fig. 4).

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

The code includes an ".if (_)" control line. Changing the condition from 0 to 1 controls whether the code simulates variant 1 or 2.

Q2LAL.cir

```
  Q2LAL
* Q2LAL initial test setup. S2LAL with "quiet 2LAL."
* Q2LAL is a signficant conceptual modification to S2LAL, albeit one that differs only in one transistor.
* Q2LAL transmits bits in straightforward dual-rail, which means a 1 is a pulse from ground to V. In S2LAL terminology, this is a "hat" pulse, meaning it has
* the most positive voltage in the middle. A Q2LAL 0 is a "hat" pulse on a different wire. In contrast, S2LAL sends a 1 on two wires, a hat pulse like Q2LAL but also
* an electrically inverted pulse on a different wire, i. e. a pulse from the idle V state to ground. S2LAL sends a 0 with both wires in the idle state.
*
* S2LAL references:
* Frank, Michael P., et al. "Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS." arXiv preprint arXiv:2009.00448 (2020).
* Contains Athas's adiabatic amplifier from:
* Athas, W. C., et al. "Low-power digital systems based on adiabatic-switching principles." IEEE Transactions on VLSI Systems 2.4 (1994): 398-407
*
* Tested with ngspice-30 (creation date Dec 28, 2018, from ngspice-30_64.zip 8,687,648 bytes)
*
* For tutorial docs: no tabs; comments start column 61; 169 character maximum line length
*
* Instructions:
* There is an .if statement on line 268. Set this to 0 or 1 to simulate S2LAL or Q2LAL
* There are three sets of plot commands at the end. Comment out either "plot or "gnuplot"
* If you want to change the length of a plot, ticks on line 210 and the time duration of the plot on line 334 and 335

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

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

*** SUBCIRCUIT DEFINITIONS
* Figure 4 in arXiv:2009.00448, Athas's adiabatic amplifier but with complementary voltages on the two halves
.SUBCKT AAMP AT AC T C piT piC GND PWR nsub psub ini='gg'   $ Athas's adiabatic amplifier. Args: AT/C T/C clockT/C substrate supplies
.ic V(T)='ini' V(C)='vv-ini'                            $ .ic V(a)={gg} V(a2)=ini
M0 piT AT T nsub n1                                     $ pass gate
M1 piT AC T psub p1
M2 piC AT C nsub n1                                     $ pass gate
M3 piC AC C psub p1
.if (CLAMP=1)
M4 GND AC T nsub n1                                     $ clamp
M5 PWR AT C psub p1
.endif
.ENDS AAMP

* Figure 5 in arXiv:2009.00448
.SUBCKT LATCH AT AC QT QC piT piC pjT pjC GND PWR           $ One phase of the 2LAL shift register. Args: AT/C QT/C clock0T/C clock1T/C
+ nsub psub tap0 tap1 tap2 tap3 ini='gg'                    $ substrate supplies
R0 tap5 QT 1                                            $ circuit taps for debugging
X1 AT AC T C piT piC GND PWR nsub psub AAMP ini='ini'
M1 T pjT QT nsub n1                                     $ Frank's latch
M2 T pjC QT psub p1
M3 C pjT QC nsub n1                                     $ Frank's latch
M4 C pjC QC psub p1
C1 AT 0 ACAP
C2 AC 0 ACAP
C3 T 0 QQCAP
C4 C 0 QQCAP
.ENDS LATCH

* Figure 6 in arXiv:2009.00448, except this is just the first stage; shift clocks for subsequent stages
.SUBCKT PHASE S0T S0C S1T S1C                           $ One stage of the 2LAL shift register. Args: AT/C QT/C
+ p0T p0C p1T p1C p2T p2C p3T p3C GND PWR nsub psub      $ 4x{ phi<n>T/C } DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 ini='gg'
X0  S0T S0C S1T S1C p1T p1C p0T p0C GND PWR nsub psub tap0 tap1 tap2 tap3 LATCH ini=ini
X10 S1T S1C S0T S0C p2T p2C p3T p3C GND PWR nsub psub tap4 tap5 tap6 tap7 LATCH ini=ini
.ends PHASE

* Figure 6 in arXiv:2009.00448, except this is all 8 stages
.SUBCKT SDELAY S0T S0C S8T S8C                          $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p0C p1T p1C p2T p2C p3T p3C                       $ clocks/power supplies
+ p4T p4C p5T p5C p6T p6C p7T p7C
+ GND PWR nsub psub                                    $ DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 tap8 tap9 tapA tapB ini='gg'
R0 tap0 S0T 1                                           $ circuit taps for debugging
R1 tap1 S0C 1
R2 tap2 S1T 1
R3 tap3 S1C 1
R4 tap4 S2T 1
R5 tap5 S2C 1
```

```
R6 tap6 S3T 1
R7 tap7 S3C 1
R8 tap8 S4T 1
R9 tap9 S4C 1
RA tapA S5T 1
RB tapB S5C 1
RC tapC S6T 1
RD tapD S6C 1
RE tapE S7T 1
RF tapF S7C 1
X0  S0T S0C S1T S1C p0T p0C p1T p1C p2T p2C p3T p3C GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 PHASE ini=gg
X1  S1T S1C S2T S2C p1T p1C p2T p2C p3T p3C P4T P4C GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 PHASE ini=ini
X2  S2T S2C S3T S3C p2T p2C p3T p3C P4T P4C P5T P5C GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 PHASE ini=ini
X3  S3T S3C S4T S4C p3T p3C P4T P4C P5T P5C P6T P6C GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 PHASE ini=ini
X4  S4T S4C S5T S5C P4T P4C P5T P5C P6T P6C P7T P7C GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 PHASE ini=ini
X5  S5T S5C S6T S6C P5T P5C P6T P6C P7T P7C P0T P0C GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 PHASE ini=ini
X6  S6T S6C S7T S7C P6T P6C P7T P7C P0T P0C P1T P1C GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 PHASE ini=gg
X7  S7T S7C S8T S8C P7T P7C P0T P0C P1T P1C P2T P2C GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 PHASE ini=gg
.ENDS SDELAY

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

* This is an inverting version of the delay circuit. It simply calls PHASEv at a point that doesn't interfere with initialization.
.SUBCKT SDELAYv S0T S0C S8T S8C                         $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p0C p1T p1C p2T p2C p3T p3C                       $ clocks/power supplies
+ p4T p4C p5T p5C p6T p6C p7T p7C
+ GND PWR nsub psub                                    $ DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 tap8 tap9 tapA tapB ini='gg'
R0 tap0 S0T 1                                          $ circuit taps for debugging
R1 tap1 S0C 1
R2 tap2 S1T 1
R3 tap3 S1C 1
R4 tap4 S2T 1
R5 tap5 S2C 1
R6 tap6 S3T 1
R7 tap7 S3C 1
R8 tap8 S4T 1
R9 tap9 S4C 1
RA tapA S5T 1
RB tapB S5C 1
RC tapC S6T 1
RD tapD S6C 1
RE tapE S7T 1
RF tapF S7C 1
X0  S0T S0C S1T S1C p0T p0C p1T p1C p2T p2C p3T p3C GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 PHASE ini=gg
X1  S1T S1C S2T S2C p1T p1C p2T p2C p3T p3C P4T P4C GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 PHASE ini=ini
X2  S2T S2C S3T S3C p2T p2C p3T p3C P4T P4C P5T P5C GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 PHASE ini=ini
X3  S3T S3C S4T S4C p3T p3C P4T P4C P5T P5C P6T P6C GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 PHASE ini=ini
X4  S4T S4C S5T S5C P4T P4C P5T P5C P6T P6C P7T P7C GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 PHASE ini=ini
X5  S5T S5C S6T S6C P5T P5C P6T P6C P7T P7C P0T P0C GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 PHASE ini=ini
X6  S6T S6C S7T S7C P6T P6C P7T P7C P0T P0C P1T P1C GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 PHASEv ini=gg
X7  S7T S7C S8T S8C P7T P7C P0T P0C P1T P1C P2T P2C GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 PHASE ini=gg
.ENDS SDELAYv

* Erik's "two hat" adiabatic amplifier. In S2LAL notation, it expects data input as A-hat and -A-hat. Given this, it produces the correct output.
.SUBCKT QAAmp AT AC T C pT C1 GND nsub psub ini='gg'    $ Erik's adiabatic amplifier. Args: AT/C T/C clock&clamp substrate supplies
.ic V(T)='ini' V(C)='vv-ini'                           $ .ic V(a)={gg} V(a2)=ini
M0 pT AT T nsub n1                                     $ pass gate
M1 pT AC T psub p1
M2 pT AC C nsub n1                                     $ pass gate
M3 pT AT C psub p1
.if (CLAMP=1)
M4 GND AC T nsub n1                                    $ clamp
M5 GND AT C nsub n1
M6 GND C1 T nsub n1                                    $ clamp
M7 GND C1 C nsub n1
.endif
.ENDS QAAmp

* This is the latched version; it is just a QAAmp followed by a pass gate.
.SUBCKT qLatch AT AC QT QC piT C1i pjT pjC GND PWR      $ One phase of the 2LAL shift register. Args: AT/C QT/C clockiT&clamp clockjT/C
+ nsub psub tap0 tap1 tap2 tap3 ini='gg'                $ substrate supplies
r0 tap0 piT 1                                          $ green
r1 tap1 T 1                                            $ red
r2 tap2 AC 1                                           $ blue
r3 tap3 C1i 1                                          $ yellow
X1 AT AC T C piT C1i GND nsub psub QAAmp ini='ini'
M1 T pjT QT nsub n1                                    $ Frank's latch
M2 T pjC QT psub p1
M3 C pjT QC nsub n1                                    $ Frank's latch
M4 C pjC QC psub p1
C1 AT 0 ACAP
C2 AC 0 ACAP
C3 T 0 QQCAP
C4 C 0 QQCAP
.ENDS qLatch

* One phase of a Q2LAL shift register.
.SUBCKT qPhase S0T S0C S1T S1C                          $ One stage of the 2LAL shift register. Args: AT/C QT/C
+ p0T p0C p1T C11 p2T C12 p3T p3C GND PWR nsub psub     $ two clocks T/C and two clocks T&clamp DC Supply substrate supplies
+ tap0 tap1 tap2 tap3 tap4 tap5 tap6 tap7 ini='gg'
r0 tap0 t0 1
r1 tap1 t1 1
r2 tap2 t2 1
r3 tap3 t3 1
X0  S0T S0C S1T S1C p1T C11 p0T p0C GND PWR nsub psub t0 t1 t2 t3 qLatch ini=ini
X10 S1T S1C S0T S0C p2T C12 p3T p3C GND PWR nsub psub tap4 tap5 tap6 tap7 qLatch ini=ini
.ends qPhase

* 8 phases of a Q2LAL shift register.
.SUBCKT qDelay S0T S0C S8T S8C                          $ Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p0C p1T p1C p2T p2C p3T p3C                       $ clocks/power supplies
+ p4T p4C p5T p5C p6T p6C p7T p7C
+ GND PWR nsub psub                                    $ DC Supply substrate supplies
+ C10 C11 C12 C13 C14 C15 C16 C17                       $ clamps
+ tap8 tap9 tapA tapB ini='gg'                         $ debugging taps and initialization
R8 tap8 t120 1
R9 tap9 t121 1
RA tapA t122 1
RB tapB t123 1
RC tapC S6T 1
```

```
RD tapD S6C 1
RE tapE S7T 1
RF tapF S7C 1
X0  S0T S0C S1T S1C p0T p0C p1T C10 p2T C11 p3T p3C GND PWR nsub psub t100 t101 t102 t103 t200 t201 t202 t203 qPhase ini=gg
X1  S1T S1C S2T S2C p1T p1C p2T C11 p3T C12 P4T P4C GND PWR nsub psub t110 t111 t112 t113 t210 t211 t212 t213 qPhase ini=ini
X2  S2T S2C S3T S3C p2T p2C p3T C12 P4T C13 P5T P5C GND PWR nsub psub t120 t121 t122 t123 t220 t221 t222 t223 qPhase ini=ini
X3  S3T S3C S4T S4C p3T p3C P4T C13 P5T C14 P6T P6C GND PWR nsub psub t130 t131 t132 t133 t230 t231 t232 t233 qPhase ini=ini
X4  S4T S4C S5T S5C P4T P4C P5T C14 P6T C15 P7T P7C GND PWR nsub psub t140 t141 t142 t143 t240 t241 t242 t243 qPhase ini=ini
X5  S5T S5C S6T S6C P5T P5C P6T C15 P7T C16 P0T P0C GND PWR nsub psub t150 t151 t152 t153 t250 t251 t252 t253 qPhase ini=ini
X6  S6T S6C S7T S7C P6T P6C P7T C16 P0T C17 P1T P1C GND PWR nsub psub t160 t161 t162 t163 t260 t261 t262 t263 qPhase ini=gg
X7  S7T S7C S8T S8C P7T P7C P0T C17 P1T C10 P2T P2C GND PWR nsub psub t170 t171 t172 t173 t270 t271 t272 t273 qPhase ini=gg
.ENDS qDelay

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

.param ticks=199                                        $ number of ticks in the simulation
$ .param ticks=499                                        $ number of ticks in the simulation
.param tick=1000NS                                      $ time of a tick
.param tstep=25NS                                       $ time of a simulation step, so number of steps is tick*ticks/tstep
.param ttn=18000ns                                      $ integration time for energy

*** CLOCKS -- Original 8 clock phases and inverses (total eight unique signals), but with slow and fast phase 1's (total 12 unique signals)
.param Ramp=0.80*tick
.param PPT=0.10*tick                                   $ one PPT at beginning and end of sequence, two of these PPTs between ramps
$ Extra delay to split phi0 into a fast and slow clock; if Fast=0, the clocks become the same
$ See Saed G. Younis. Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic. No. AI-TR-1500. MIT AI Laboratory, 1994.
$ .param Fast=0                                          $ PPT+Ramp+PPT
.param Fast=PPT+Ramp+PPT

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

* These are clamp waveforms. They go high for one tick to clamp signals to ground. These are not clocks.
* Each can be generated with four transistors from existing clocks. They only connect to transistor gates, so they do not need a lot of drive capability.
Vc0    710 0 DC 'vv' PWL('0' 'vv'                               'f0uS' 'vv' 'f0uS+Ramp' 'gg'     'f2dS' 'gg' 'f2dS+Ramp' 'vv'     'epoc' 'vv' r='0')
Vc1    711 0 DC 'vv' PWL('0' 'vv'                               'f1up' 'vv' 'f1up+Ramp' 'gg'     'f3dn' 'gg' 'f3dn+Ramp' 'vv'     'epoc' 'vv' r='0')
Vc2    712 0 DC 'gg' PWL('0' 'gg'                               'f0uS' 'gg' 'f0uS+Ramp' 'vv'     'f2up' 'vv' 'f2up+Ramp' 'gg'     'epoc' 'gg' r='0')
Vc3    713 0 DC 'gg' PWL('0' 'gg'                               'f1up' 'gg' 'f1up+Ramp' 'vv'     'f3up' 'vv' 'f3up+Ramp' 'gg'     'epoc' 'gg' r='0')
Vc4    714 0 DC 'gg' PWL('0' 'gg'                               'f2up' 'gg' 'f2up+Ramp' 'vv'     'f0dn' 'vv' 'f0dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vc5    715 0 DC 'gg' PWL('0' 'gg'                               'f3up' 'gg' 'f3up+Ramp' 'vv'     'f1dn' 'vv' 'f1dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vc6    716 0 DC 'gg' PWL('0' 'gg'                               'f0dn' 'gg' 'f0dn+Ramp' 'vv'     'f2dS' 'vv' 'f2dS+Ramp' 'gg'     'epoc' 'gg' r='0')
Vc7    717 0 DC 'gg' PWL('0' 'gg'                               'f1dn' 'gg' 'f1dn+Ramp' 'vv'     'f3dn' 'vv' 'f3dn+Ramp' 'gg'     'epoc' 'gg' r='0')

* These are the power clocks, including separate fast and slow clocks
Vphi0P 110 0 DC 'gg' PWL('0' 'gg'                               'f0uS' 'gg' 'f0uS+Ramp' 'vv'     'f0dn' 'vv' 'f0dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi0f 510 0 DC 'gg' PWL('0' 'gg'                               'f0uF' 'gg' 'f0uF+Ramp' 'vv'     'f0dn' 'vv' 'f0dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi1P 111 0 DC 'gg' PWL('0' 'gg'                               'f1up' 'gg' 'f1up+Ramp' 'vv'     'f1dn' 'vv' 'f1dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi2P 112 0 DC 'gg' PWL('0' 'gg'                               'f2up' 'gg' 'f2up+Ramp' 'vv'     'f2dS' 'vv' 'f2dS+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi2f 512 0 DC 'gg' PWL('0' 'gg'                               'f2up' 'gg' 'f2up+Ramp' 'vv'     'f2dF' 'vv' 'f2dF+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi3P 113 0 DC 'gg' PWL('0' 'gg'                               'f3up' 'gg' 'f3up+Ramp' 'vv'     'f3dn' 'vv' 'f3dn+Ramp' 'gg'     'epoc' 'gg' r='0')
Vphi4f 514 0 DC 'vv' PWL('0' 'vv'                               'f0uF' 'vv' 'f0uF+Ramp' 'gg'     'f0dn' 'gg' 'f0dn+Ramp' 'vv'     'epoc' 'vv' r='0')
Vphi4P 114 0 DC 'vv' PWL('0' 'vv'                               'f0uS' 'vv' 'f0uS+Ramp' 'gg'     'f0dn' 'gg' 'f0dn+Ramp' 'vv'     'epoc' 'vv' r='0')
Vphi5P 115 0 DC 'vv' PWL('0' 'vv'                               'f1up' 'vv' 'f1up+Ramp' 'gg'     'f1dn' 'gg' 'f1dn+Ramp' 'vv'     'epoc' 'vv' r='0')
Vphi6f 516 0 DC 'vv' PWL('0' 'vv'                               'f2up' 'vv' 'f2up+Ramp' 'gg'     'f2dF' 'gg' 'f2dF+Ramp' 'vv'     'epoc' 'vv' r='0')
Vphi6P 116 0 DC 'vv' PWL('0' 'vv'                               'f2up' 'vv' 'f2up+Ramp' 'gg'     'f2dS' 'gg' 'f2dS+Ramp' 'vv'     'epoc' 'vv' r='0')
Vphi7P 117 0 DC 'vv' PWL('0' 'vv'                               'f3up' 'vv' 'f3up+Ramp' 'gg'     'f3dn' 'gg' 'f3dn+Ramp' 'vv'     'epoc' 'vv' r='0')

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

*** TOP-LEVEL CIRCUIT
* Set the flat to 0 for a test of the quiet circuit and 1 for standard 2LAL
.if (1)
X0 SAT SAC SBT SBC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 pp8 pp9 ppA ppB qDelay ini=gg
X1 SBT SBC SCT SCC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 uu8 uu9 uuA uuB qDelay ini=gg
X5 SCT SCC SAC SAT 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 xx8 xx9 xxA xxB qDelay ini=vv

X2 SXT SXC SYT SYC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 qq8 qq9 qqA qqB qDelay ini=gg
X3 SYT SYC SZT SZC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 vv8 vv9 vvA vvB qDelay ini=gg
X4 SZT SZC SXC SXT 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 710 711 712 713 714 715 716 717 ww8 ww9 wwA wwB qDelay ini=vv
.else
X0 SAT SAC SBT SBC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 pp0 pp1 pp2 pp3 pp4 pp5 pp6 pp7 pp8 pp9 ppA ppB SDELAY ini=gg
X1 SBT SBC SCT SCC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 uu0 uu1 uu2 uu3 uu4 uu5 uu6 uu7 uu8 uu9 uuA uuB SDELAY ini=gg
X5 SCT SCC SAT SAC 110 114 111 115 112 116 113 117 114 110 115 111 116 112 117 113 200 201 200 201 xx0 xx1 xx2 xx3 xx4 xx5 xx6 xx7 xx8 xx9 xxA xxB SDELAYv ini=gg

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

* power and energy calculation
B4 0 16 V=0
+ +I(Vc0)*v(710)+I(Vc1)*v(711)+I(Vc2)*v(712)+I(Vc3)*v(713)+I(Vc4)*v(714)+I(Vc5)*v(715)+I(Vc6)*v(716)+I(Vc7)*v(717)
+ +I(vphi0P)*v(110)+I(vphi1P)*v(111)+I(vphi2P)*v(112)+I(vphi3P)*v(113)+I(vphi4P)*v(114)+I(vphi5P)*v(115)+I(vphi6P)*v(116)+I(vphi7P)*v(117)
+ +I(vphi0f)*v(510)+I(vphi2f)*v(512)+I(vphi4f)*v(514)+I(vphi6f)*v(116)
+ +I(VGND)*v(200)+I(VPWR)*v(201)
A1 16 17 power_tally
.model power_tally int(in_offset=0.0 gain=1.0 out_lower_limit=-1e12 out_upper_limit=1e12 limit_range=1e-9 out_ic=0.0)

.option noinit acct

*********************************************************************************************
$ NGSPICE CONTROL AREA
.TRAN 'tstep' 'ticks*tick'
.control
pre_set strict_errorhandling
unset ngdebug
run

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

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

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

plot                                                    $ plot instantaneous energy consumption
$ gnuplot gp/power title "Instantaneous dissipation"
+ ylimit -25m 25m
+ v(16)

plot  ylimit 0 70n                                      $ plot accumulated energy dissipation
$ gnuplot gp/energy title "Cumulative dissipation" ylimit 0 70n
+ v(17)

plot title "3-stage Q2LAL/S2LAL inverting shift register" ylimit 0 6 xlimit 0 200u
$ gnuplot gp/traces ylimit 0 6 xlimit 0 200u title "3-stage Q2LAL/S2LAL inverting shift register"
*+ v(710)/9.99*0.9+ 8.55+.000*3
*+ v(711)/9.99*0.9+ 8.55+.025*3
*+ v(712)/9.99*0.9+ 8.55+.050*3
*+ v(713)/9.99*0.9+ 8.55+.075*3
*+ v(714)/9.99*0.9+ 8.55+.100*3
*+ v(715)/9.99*0.9+ 8.55+.125*3
*+ v(716)/9.99*0.9+ 8.55+.150*3
*+ v(717)/9.99*0.9+ 8.55+.175*3
+
*+ v(110)/9.99*0.9+6.55
+
+ v(uu8)/9.99*0.9+ 4.675                                $ green
+ v(uu9)/9.99*0.9+ 4.625                                $ red
+ v(uuA)/9.99*0.9+ 4.575                                $ blue
+ v(uuB)/9.99*0.9+ 4.525                                $ yellow
+
+ v(SAT)/9.99*0.9+ 0.55
+ v(SAC)/9.99*0.9+ 0.55+.05
+ v(SXT)/9.99*0.9+ 2.55
+ v(SXC)/9.99*0.9+ 2.55+.05

.endc

.END
```