# Barrel Shifter Permutation Circuit, 2LAL and SCRL

Technical report ZF003 v1.0, May 12, 2020

Erik P. DeBenedictis
Albuquerque, NM 87112
erikdebenedictis@zettaflops.org

## Abstract

To illustrate the concept of FET-only adiabatic permutation stages as described in ref. 1, these test circuits demonstrate both 2LAL and SCRL 4-bit barrel shifters implemented with FET-only permutation stages. The circuits are in a test harness comprising a four- or five-stage shift register preloaded with data patterns 1, 2, 4, 8 and a shift count pattern of 3, 3, 3, 3 places. The files should be available wherever the reader obtained this document, but the main files have been pasted into the appendix of this document as a failsafe. The circuits are designed for ngspice version 30 and use the built-in BSIM 3.3.0 transistor model, with voltage sources to alter gate bias (which is not realistic).

## Permutation stages

Permutations stages[1] apply generally to both SCRL and 2LAL. They will be summarized here in SCRL notation, although both 2LAL and SCRL test circuits are provided.

The diagram in fig. 1, reproduced from Frank's thesis[2] shows how arbitrary functions $f_n$ can be realized in a reversible SCRL pipeline, provided that the pipeline also includes $f_n^{-1}$. However, this structure requires separate logic networks to compute $f_n$ and $f_n^{-1}$, doubling transistor count. A permutation stage accomplishes the same result with less resource consumption.

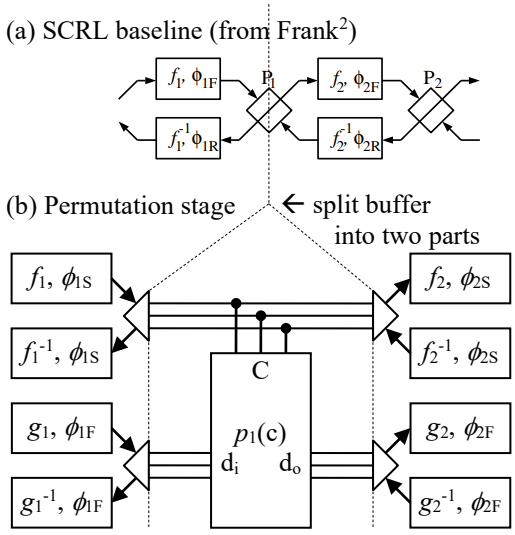



Fig. 1. Permutation stage. Requires fast and slow clocks, yet occurs independently of the invertible stages in the literature.

Like CMOS, 2LAL and SCRL gates can be composed into higher-level logic blocks with wires. Based on the terminology in ref. 1, the bidirectional latches are split through the middle with the two halves connected by single wires (SCRL) or dual-rail signals (2LAL). The splitting of the latches is illustrated in fig. 1b, which shows 3-bit signals between stages.

However, permutation stages can implement some important standard logic functions quite efficiently. For example, ref. 1 illustrates a Fredkin gate, which can be viewed as a control signal that either wires two signals straight through or swaps them.

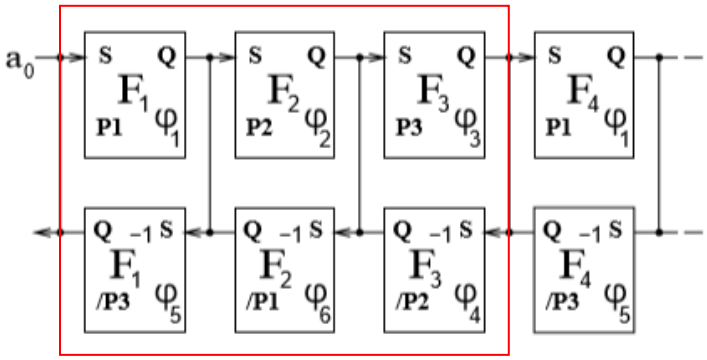
## nFET-only permutation networks in 2LAL

Standard 2LAL is entirely comprised of two-transistor pass gates, but ref. 1 shows how to adjust clock voltages and transistor thresholds so that the pFETs can be removed from alternate stages (leaving a stage with single nFETs in place of pass gates, resulting in half the transistor count for the stage). The test circuit includes two versions of the pass gate subcircuit. One version (PASS) is a standard two-transistor pass gate while the other version (NoPASS – for "nFET-only pass") has the pFET "commented out." An experimenter can uncomment the pFETs and observe that the circuit continues to work.

## SCRL circuit structure

Saad Younis's thesis[3] is the defining document for 2LAL, with fig. 2a reproducing the 3-phrase variant from his thesis. Unfortunately, at time $t$=0, Younis's 3-phase block holds state in the rightmost of the three buffers, leading to the block driving its initialization value to its output. For convenience, this work defines the basic block starting one phase to the right compared to Younis. As illustrated in fig. 2b, this means the state at $t$=0 is entirely contained in the central buffer shown in red and a module's outputs are all in a high impedance state.

(a) SCRL baseline (figure 4.7 in Younis' thesis[3])




(b) This work shifts by a phase but does not rename the clocks

(c) Permutation network here

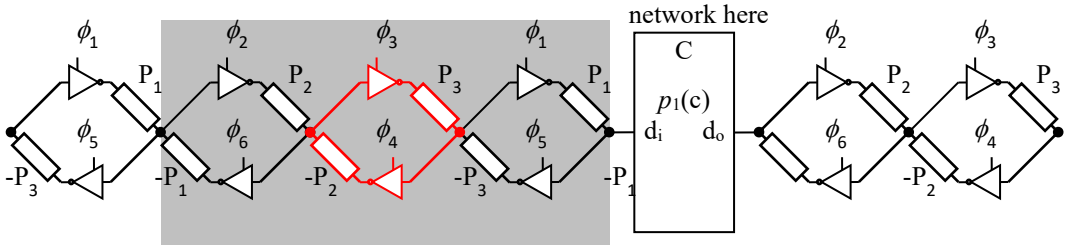
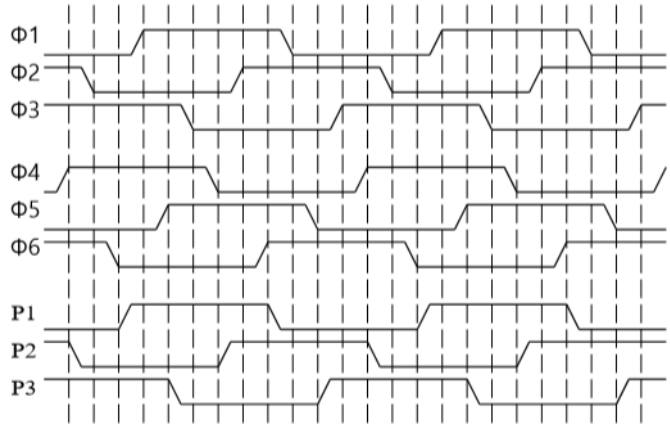



Fig. 2. Connection between this circuit and the SCRL circuit in the literature. Essentially, we shift right by one phase compared to the circuit in Younis's thesis,[3] but we do not change the names of the clocks. This causes initialization value to be entirely contained in the three-buffer module. If the permutation network is to go between modules, there must be fast clocks for $\phi_1$ and $\phi_6$ (but not the other phases).

If the permutation network is imagined as rewiring the circuit, as in an FPGA, the obvious place to put the permutation network is between the modules. This is illustrated in fig. 2c. This implies fast clocks for $\phi_1$ and $\phi_6$ (but not the other phases).

Fig. 3 shows the clocks from Younis's thesis and the equivalent plot from the SCRL test circuit. Younis used the same transition time for both the $\phi$s and Ps (one half the spacing between the vertical lines in fig. 3a), but the simulation program allows these two transition times to be set independently. The simulation also allows an independently settable gap between transitions.

(a) SCRL baseline (figure 4.8 in Younis' thesis[3])




(b) This work

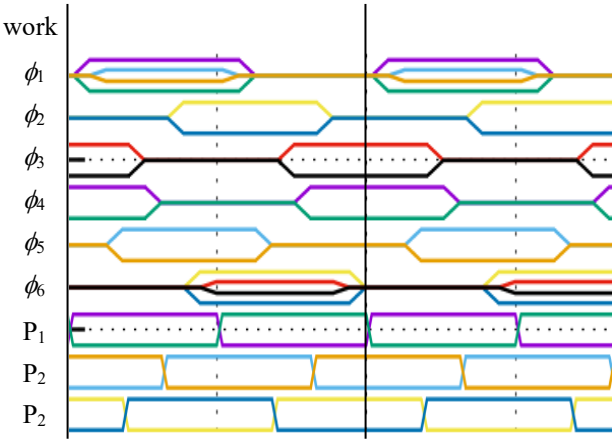



Fig. 3. Approximate translation of clocks. This work has unequal transition times for $\phi$s and Ps and fast $\phi_1$ and $\phi_6$. Just a fast $\phi_1$ and $\phi_6$ is not sufficient for inversion per Younis, but it can be accomplished in the permutation network.

## Test circuit

A 4-bit barrel shifter in fig. 4 can be viewed as two sequential permutations, a controlled rotation by one place followed by a controlled rotation of two places. The four combinations of the two controls can be viewed as a 2-bit binary number representing the number of places to rotate. Each of the small dashed blocks in fig. 2 is a two-input multiplexer controlled by one of the two bits of shift count c, where a zero on the control bit enables the straight-through dashed-line path and a one on the control bit enables the dotted-line diagonal path. The first stage of the wiring pattern shifts each bit down one position (with wrap around) and the wiring pattern in the second stage shifts two places. The rest of the structure comprises a test harness with initial data that rotates downward by three places (with wrap-around, so it is equivalent to upward rotation by one place).

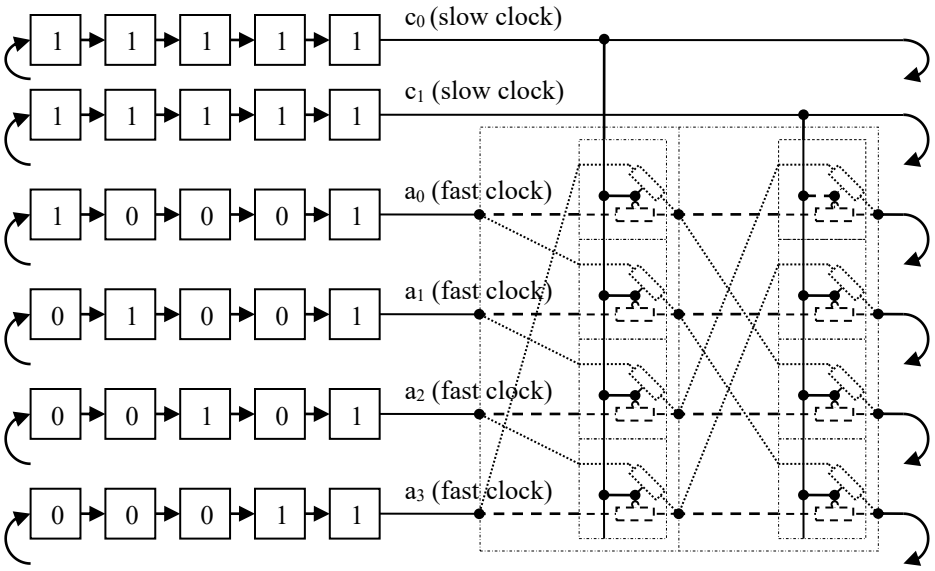



Fig. 4. 2LAL 4-bit barrel shifter and test harness. The smaller outlined structures are 2-input multiplexers and the larger ones are shifters. The result is a complex reversible module. Test harness and initial data for the simulation in the appendix appear on the left. The SCRL circuit has four shift register stages instead of five.

The 2LAL circuit in fig. 4 is essentially a five stage shift register. The fifth stage allows "alignment marks" on the plot, which is an advantage. However, each stages of an SCRL shift register inverts its data, so a five-stage shift register would invert the data every time the data cycles through the entire register, which is inconvenient. So, the SCRL circuit only has four stages.

## Expected output

Fig. 5 shows the output from running the 2LAL test circuit in ngspice. The six-bit word in each shift register stage comprises a two-bit rotation count and a four-bit data word. Four of the five stages are loaded with the binary codes 0001, 0010, 0100, and 1000. The fifth state is all 1's, and corresponds to the dashed overlay markings in fig. 5. The binary codes rotate one position every five cycles. This is apparent by the dotted overlay arrows in fig. 5: the first bit after each dashed overlay marking moves up one position, wrapping around to the bottom.

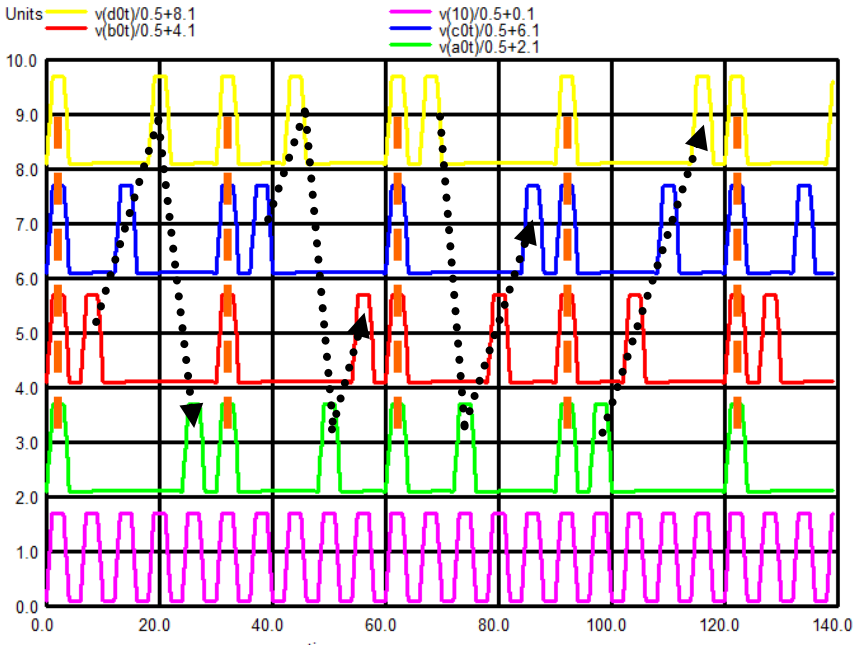



Fig. 5. Spice simulation of the 2LAL barrel rotator. Each dashed orange line represents the start of a cycle. The dotted orange line shows pulses that start each cycle moving down three traces (mod 4) from the previous, which is actually up one trace. Bottom trace is a clock.

Fig. 6 is the equivalent for SCRL. There are no alignment stages, but the plots have been deliberately constructed to be similar.

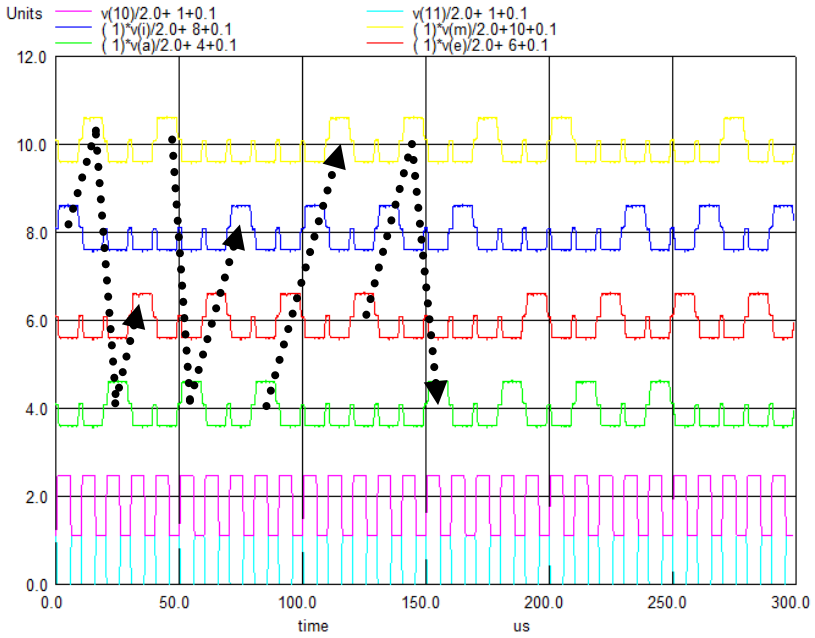



Fig. 6. Spice simulation of the SCRL barrel rotator. The dotted line shows pulses that starts each cycle moving down three traces (mod 4) from the previous, which is actually up one trace. Bottom trace is a clock.

## Instructions

Download and install ngspice version 30. The files b2lal.cir and bscrl.cir implements the circuit in described in this document.

There are several additional files that may be helpful in making changes to voltage ranges and transistor parameters, such as thresholds.

If there is a problem finding the files, they have been cut-and-pasted in the appendix of this document.

## Reference


[1] E. DeBenedictis, "Enhancements to Adiabatic Logic for Quantum Computer Control Electronics," Technical report ZF002 v1.02, February 20, 2020, http://www.zettaflops.org/CATC/2LAL_Inv_P5.pdf.

[2] Frank, Michael Patrick, and Thomas F. Knight Jr. *Reversibility for efficient computing*. Diss. Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1999.

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

# Appendix: ngspice files

## B2lal.cir

```
  NoBARREL
* 4-bit Barrel Shifter as an NFET-only Permutation Circuit; previously called 2LALINVERTER
* 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
* Barrel shifter is NFET-only, i. e .the pass gates in the shifter circuit do no have PFETs

*** SUBCIRCUIT DEFINITIONS
.SUBCKT PASS D GT GC S psub nsub                          // Pass gate. Args: Drain GateT/C Source psub nsub
voffn GT tn -1.1                                          // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp .7
M1 i1 tn v2 nsub n1
M2 i1 tp v2 psub p1
R1 D v1 0                                                 // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                                 // v1 and v2 are voltages on both sides of the channel
.ENDS PASS

.SUBCKT NoPASS D GT GC S psub nsub                        // NFET-only pass gate (i. e. just an NFET): Args: Drain GateT/C Source psub nsub
voffn GT tn -1.1                                          // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp .7
M1 i1 tn v2 nsub n1
//M2 i1 tp v2 psub p1
R1 D v1 0                                                 // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                                 // v1 and v2 are voltages on both sides of the channel
.ENDS NoPASS

.SUBCKT PHASE iiT iiC ooT ooC p0T p0C p1T p1C psub nsub   // One phase of the 2LAL shift register. Args: iiT/C ooT/C clock0T/C clock1T/C
X1 iiT ooT ooC p0T psub nsub PASS
X2 iiC ooT ooC p0C psub nsub PASS
X3 ooT iiT iiC p1T psub nsub PASS
X4 ooC iiT iiC p1C psub nsub PASS
C1 iiT 0 5e-12
C2 iiC 0 5e-12
.ENDS PHASE

.SUBCKT DELAY d0T d0C d4T d4C                             // Four phases that just delay. Args: 2*{ data<n>T/C }
+ p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C psub nsub     // 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 P0T P0C P1Tf P1Cf psub nsub PHASE
X2 d1T d1C d2T d2C P1Tf P1Cf p2T p2C psub nsub PHASE
X3 d2T d2C d3T d3C p2T p2C p3T p3C psub nsub PHASE
X4 d3T d3C d4T d4C p3T p3C P0T P0C psub nsub PHASE
.ENDS DELAY

.SUBCKT DELAY1 d0T d0C d1T d1C x1T x1C d4T d4C            // Four phases that delay, but tapping phase 1. Args: 4*{ data<n>T/C }
+ p0T p0C p1T p1C p2T p2C p3T p3C psub nsub               // clocks/power supplies. Includes either phase 1 slow or fast, but not both
+ ini=0
.ic V(d0T)={gg} V(x1T) = {gg} V(d2T) = {   ini} V(d3T) = {   ini}
.ic V(d0C)={vv} V(x1C) = {vv} V(d2C) = {vv-ini} V(d3C) = {vv-ini}
X1 d0T d0C d1T d1C P0T P0C P1T P1C psub nsub PHASE
X2 x1T x1C d2T d2C P1T P1C p2T p2C psub nsub PHASE
X3 d2T d2C d3T d3C p2T p2C p3T p3C psub nsub PHASE
X4 d3T d3C d4T d4C p3T p3C P0T P0C psub nsub PHASE
.ENDS DELAY1

.SUBCKT MUX2 a1T a1C b1T b1C c1T c1C x1T x1C psub nsub    // One input, three bidirectional, T/C. Args: addrT/C in<0..1>T/C outT/C
X1  c1T a1T a1C x1T psub nsub PASS
X2  c1C a1T a1C x1C psub nsub PASS
X3  b1T a1C a1T x1T psub nsub PASS
X4  b1C a1C a1T x1C psub nsub PASS
.ENDS

.SUBCKT NoMUX2 a1T a1C b1T b1C c1T c1C x1T x1C psub nsub  // NFET-only MUX: one input, three bidirectional, T/C. Args: addrT/C in<0..1>T/C outT/C
X1  c1T a1T a1C x1T psub nsub NoPASS
X2  c1C a1T a1C x1C psub nsub NoPASS
X3  b1T a1C a1T x1T psub nsub NoPASS
X4  b1C a1C a1T x1C psub nsub NoPASS
.ENDS NoMUX2

.SUBCKT ROT2x4 i0T i0C j0T j0C                            // Args: Two address inputs, i0, j0, T/C.
+ a0T a0C b0T b0C c0T c0C d0T d0C                         // Four inputs, a0, b0, c0, d0, T/C.
+ i4T i4C j4T j4C                                        // Two address outputs, i4, j4, T/C, copies of inputs
+ a4T a4C b4T b4C c4T c4C d4T d4C                         // Four outputs, a4, b4, c4, d4, T/C; circular rotation of inputs
+ p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C psub nsub     // clocks/power supplies
+ inii=0 inij=0 inia=0 inib=0 inic=0 inid=0
X1 i0T i0C i1T i1C i1T i1C i4T i4C p0T p0C p1T  p1C  p2T p2C p3T p3C psub nsub DELAY1 ini=inii
X2 j0T j0C j1T j1C j1T j1C j4T j4C p0T p0C p1T  p1C  p2T p2C p3T p3C psub nsub DELAY1 ini=inij
X3 a0T a0C a1T a1C w1T w1C a4T a4C p0T p0C p1Tf p1Cf p2T p2C p3T p3C psub nsub DELAY1 ini=inia
X4 b0T b0C b1T b1C x1T x1C b4T b4C p0T p0C p1Tf p1Cf p2T p2C p3T p3C psub nsub DELAY1 ini=inib
X5 c0T c0C c1T c1C y1T y1C c4T c4C p0T p0C p1Tf p1Cf p2T p2C p3T p3C psub nsub DELAY1 ini=inic
X6 d0T d0C d1T d1C z1T z1C d4T d4C p0T p0C p1Tf p1Cf p2T p2C p3T p3C psub nsub DELAY1 ini=inid
X7 i1T i1C a1T a1C b1T b1C q1T q1C psub nsub NoMUX2
X8 i1T i1C b1T b1C c1T c1C r1T r1C psub nsub NoMUX2
X9 i1T i1C c1T c1C d1T d1C s1T s1C psub nsub NoMUX2
X10 i1T i1C d1T d1C a1T a1C t1T t1C psub nsub NoMUX2
X11 j1T j1C q1T q1C s1T s1C w1T w1C psub nsub NoMUX2
X12 j1T j1C r1T r1C t1T t1C x1T x1C psub nsub NoMUX2
X13 j1T j1C s1T s1C q1T q1C y1T y1C psub nsub NoMUX2
X14 j1T j1C t1T t1C r1T r1C z1T z1C psub nsub NoMUX2
.ENDS ROT2x4

.SUBCKT LINEARSHIFT d0T d0C                               // Four full cycle delay. Args: data in/out T/C
+ e0T e0C                                                // stuff
+ p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C psub nsub     // clocks/power supplies
+ ini0=0 ini1=0 ini2=0 ini3=0
X1 d0T d0C d4T d4C     p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C     psub nsub DELAY ini=ini3
X2 d4T d4C d8T d8C     p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C     psub nsub DELAY ini=ini2
X3 d8T d8C dCT dCC     p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C     psub nsub DELAY ini=ini1
X4 dCT dCC e0T e0C     p0T p0C p1T p1C p1Tf p1Cf p2T p2C p3T p3C     psub nsub DELAY ini=ini0
.ENDS LINEARSHIFT

*** POWER-CLOCKS
```

```
VCC   99  0   DC {vv}
VGG   98  0   DC {gg}

VPS   79  0   DC { 8.0 }                                  // Substrate bias; overhangs by 2 V
VNS   78  0   DC { -2 }

*** ALL INPUTS
.param gg=0V                                              // range allowed from a full pass gate stage
.param vv=6V

.param gf=0V                                              // range allowed from an NFET-only stage
.param vf=4.6V

.param ticks=139                                          // number of ticks in the simulation
.param tick=2000NS/1.2                                    // time of a tick
.param tstep=100NS/1.2                                    // time of a simulation step, so number of steps is tick*ticks/tstep
.param ttn=36000ns/1.2                                    // integration time for energy

*** CLOCKS -- Original 4 clock phases and inverses (total four unique signal), but with slow and fast phase 1's (total six unique signals)
vph0  10  0   DC {gg} PWL({0*tick} {gg} {1*tick} {vv} {2*tick} {vv} {3*tick} {vv} {4*tick} {gg} {5*tick} {gg} {6*tick} {gg}) r={0*tick}
vp1s  11  0   DC {gg} PWL({0*tick} {gg} {1*tick} {gg} {2*tick} {vv} {3*tick} {vv} {4*tick} {vv} {5*tick} {vv} {6*tick} {gg}) r={0*tick}
vp1f  12  0   DC {gf} PWL({0*tick} {gf} {1*tick} {gf} {2*tick} {gf} {3*tick} {vf} {4*tick} {vf} {5*tick} {gf} {6*tick} {gf}) r={0*tick}
vph2  13  0   DC {vv} PWL({0*tick} {vv} {1*tick} {gg} {2*tick} {gg} {3*tick} {gg} {4*tick} {vv} {5*tick} {vv} {6*tick} {vv}) r={0*tick}
vph3  14  0   DC {vv} PWL({0*tick} {vv} {1*tick} {vv} {2*tick} {gg} {3*tick} {gg} {4*tick} {gg} {5*tick} {gg} {6*tick} {vv}) r={0*tick}

vqh0  20  0   DC {vv} PWL({0*tick} {vv} {1*tick} {gg} {2*tick} {gg} {3*tick} {gg} {4*tick} {vv} {5*tick} {vv} {6*tick} {vv}) r={0*tick}
vq1s  21  0   DC {vv} PWL({0*tick} {vv} {1*tick} {vv} {2*tick} {gg} {3*tick} {gg} {4*tick} {gg} {5*tick} {gg} {6*tick} {vv}) r={0*tick}
vq1f  22  0   DC {vf} PWL({0*tick} {vf} {1*tick} {vf} {2*tick} {vf} {3*tick} {gf} {4*tick} {gf} {5*tick} {vf} {6*tick} {vf}) r={0*tick}
vqh2  23  0   DC {gg} PWL({0*tick} {gg} {1*tick} {vv} {2*tick} {vv} {3*tick} {vv} {4*tick} {gg} {5*tick} {gg} {6*tick} {gg}) r={0*tick}
vqh3  24  0   DC {gg} PWL({0*tick} {gg} {1*tick} {gg} {2*tick} {vv} {3*tick} {vv} {4*tick} {vv} {5*tick} {vv} {6*tick} {gg}) r={0*tick}

*** TOP-LEVEL CIRCUIT
X1 i0T i0C i1T i1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=vv ini1=vv ini2=vv ini3=vv // 1's bit
X2 j0T j0C j1T j1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=vv ini1=vv ini2=vv ini3=vv // 2's bit
X3 a0T a0C a1T a1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=vv ini1=gg ini2=gg ini3=gg
X4 b0T b0C b1T b1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=gg ini1=vv ini2=gg ini3=gg
X5 c0T c0C c1T c1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=gg ini1=gg ini2=vv ini3=gg
X6 d0T d0C d1T d1C  10 20 11 21 12 22 13 23 14 24 79 78 LINEARSHIFT ini0=gg ini1=gg ini2=gg ini3=vv
X7 i1T i1C j1T j1C a1T a1C b1T b1C c1T c1C d1T d1C
+ i0T i0C j0T j0C a0T a0C b0T b0C c0T c0C d0T d0C
+                  10 20 11 21 12 22 13 23 14 24 79 78 ROT2x4      inii=gg inij=gg inia=vv inib=vv inic=vv inid=vv

* power and energy calculation
B4 0 16 V=I(vph0)*v(10)+I(vp1s)*v(11)+I(vp1f)*v(12)+I(vph2)*v(13)+I(vph3)*v(14)+I(vqh0)*v(20)+I(vq1s)*v(21)+I(vq1f)*v(22)+I(vqh2)*v(23)+I(vqh3)*v(24)
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
.TRAN {tstep} {ticks*tick}

* use BSIM3 model with default parameters
.model n1 nmos level=49 version=3.3.0
.model p1 pmos level=49 version=3.3.0

.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'

**********************************************
plot v(16)                                                // plot instantaneous energy consumption
plot v(17)                                                // plot accumulated energy dissipation

* 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=3

plot title "2LAL clock and data traces" ylimit 0 12 xlimit 0 300u
+ v(a0T)/3.5+3.1 v(b0T)/3.5+5.1 v(c0T)/3.5+7.1 v(d0T)/3.5+9.1// non-inverted output
+ v(X1.d4T)/3.5+0.1                                       // X[1-6] is Linearshift

plot title "current through shift register stage"
+ (v(X1.X1.X1.X1.v1)-v(X1.X1.X1.X1.i1))                   // X[1-6].X[1-4].X[1-4].X[1-4].[v1, i1, v2] is a voltage or current tap
+ (v(X1.X1.X1.X2.v1)-v(X1.X1.X1.X2.i1))
+ (v(X1.X1.X1.X3.v1)-v(X1.X1.X1.X3.i1))
+ (v(X1.X1.X1.X4.v1)-v(X1.X1.X1.X4.i1))

plot title "voltage across pass gate in shift register stage"
+ (v(X1.X1.X1.X1.v1)-v(X1.X1.X1.X1.v2))/2.0+0.1           // X[1-6].X[1-4].X[1-4].X[1-4].[v1, i1, v2] is a voltage or current tap
+ (v(X1.X1.X1.X2.v1)-v(X1.X1.X1.X2.v2))/2.0+2.1
+ (v(X1.X1.X1.X3.v1)-v(X1.X1.X1.X3.v2))/2.0+4.1
+ (v(X1.X1.X1.X4.v1)-v(X1.X1.X1.X4.v2))/2.0+6.1

plot title "current through mux"
+ (v(X7.X7.X1.v1)-v(X7.X7.X1.i1))                         // X[7].X[7-14].X[1-4].[v1, i1, v2] is a voltage or current tap
+ (v(X7.X7.X2.v1)-v(X7.X7.X2.i1))
+ (v(X7.X7.X3.v1)-v(X7.X7.X3.i1))
+ (v(X7.X7.X4.v1)-v(X7.X7.X4.i1))

plot title "voltage across pass gate in mux"
+ (v(X7.X7.X1.v1)-v(X7.X7.X1.v2))/2.0+0.1               // X[7].X[7-14].X[1-4].[v1, i1, v2] is a voltage or current tap
+ (v(X7.X7.X2.v1)-v(X7.X7.X2.v2))/2.0+2.1
+ (v(X7.X7.X3.v1)-v(X7.X7.X3.v2))/2.0+4.1
+ (v(X7.X7.X4.v1)-v(X7.X7.X4.v2))/2.0+6.1

.endc

.END
```

## Bscrl.cir

```
  SCRLCATC
* SCRL 3 Phase for CATC
* 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
* Implements a barrel shifter implemented as a permutation network, and the permutation networks uses only single transistor switches, not pass gates
.MODEL pch pmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
.MODEL nch nmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)

*** SUBCIRCUIT DEFINITIONS
```

```
.SUBCKT MUX2 in0 in1 adr out psub nsub                    // inputs in0 in1 adr out; connect in[adr] to out
V1 adr tp 0.0                                             // adr must have larger swing than data
M1 in0 tp out psub psub pch                               // adr = 0 --> in0 connects to out
V2 adr tn -0.0
M2 in1 tn out nsub nsub nch                               // adr = 1 --> in1 connects to out
r1 in0 out 1e8
r2 in1 out 1e8
.ENDS

.SUBCKT ROT2x4 c2 c1                                      // Args: c2 c1 binary shift count
+ in0 in1 in2 in3                                         // Four inputs, in0 in1 in2 in3
+ out0 out1 out2 out3                                     // Four outputs, out0 out1 out2 out3; circular rotation of inputs
+ psub nsub                                               // transistor substrate bias -- be careful, circuit depends of thresholds and unequal voltage levels
X1 in0 in1 c1 t0 psub nsub MUX2
X2 in1 in2 c1 t1 psub nsub MUX2
X3 in2 in3 c1 t2 psub nsub MUX2
X4 in3 in0 c1 t3 psub nsub MUX2
X5 t0 t2 c2 out0 psub nsub MUX2
X6 t1 t3 c2 out1 psub nsub MUX2
X7 t2 t0 c2 out2 psub nsub MUX2
X8 t3 t1 c2 out3 psub nsub MUX2
.ENDS ROT2x4

.SUBCKT INVPASS S Q phiP phiN pP pN psub nsub
* Page 44 Fig 4.1 Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic by Saed G. Younis
*   Args:  S Q phiP/N P1P/N substrate_for_PFETS substrate_for_NFETS
M1 phiP S node psub p1
M2 phiN S node nsub n1
M3 node pN Q psub p1
M4 node pP Q nsub n1
.ENDS INVPASS

.SUBCKT STAGE a a1 a2 b phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N  phi5P phi5N  phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub
+ ini=0
* Page 49 Fig 4.7 Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic by Saed G. Younis
*   Args: a 2InternalNodes b 6DualINVPASSClocks 3Ps substrate_for_PFETS substrate_for_NFETS
.ic V(a)={gg} V(a2)=ini
X1 a a1 phi2P phi2N p2P p2N psub nsub INVPASS
X2 a1 a phi6P phi6N p1N p1P psub nsub INVPASS
X3 a1 a2 phi3P phi3N p3P p3N psub nsub INVPASS
X4 a2 a1 phi4P phi4N p2N p2P psub nsub INVPASS
X5 a2 b phi1P phi1N p1P p1N psub nsub INVPASS
X6 b a2 phi5P phi5N p3N p3P psub nsub INVPASS
.ENDS STAGE

.SUBCKT SHIFTREG a a1 a2 b b1 b2 c c1 c2 d d1 d2 e phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N phi5P phi5N phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub
+ ini0=0 ini1=0 ini2=0 ini3=0
*   Args: a 5InternalNodes b 6DualRailClocks 3DualRailPs substrate_for_PFETS substrate_for_NFETS
X10 a a1 a2 b phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N phi5P phi5N phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub STAGE ini=ini3
X11 b b1 b2 c phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N phi5P phi5N phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub STAGE ini=ini2
X12 c c1 c2 d phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N phi5P phi5N phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub STAGE ini=ini1
X13 d d1 d2 e phi1P phi1N phi2P phi2N phi3P phi3N phi4P phi4N phi5P phi5N phi6P phi6N p1P p1N p2P p2N p3P p3N psub nsub STAGE ini=ini0
.ENDS SHIFTREG

*** POWER
VCC   99  0   DC {vv}
VGG   98  0   DC {gg}
VNN   97  0   DC {vn}
VFP   96  0   DC {vpf}                                   // the fast clock has a lower voltage
VFN   95  0   DC {vnf}
VBP   94  0   DC {vpf-Back}                               // Back gate voltages
VBN   93  0   DC {vnf+Back}

*** VOLTAGE SCALING
.param Back=1.75                                           // for nFET(pFET), this is the back gate voltage referenced to the smallest(largest) source voltage during
operation
.param vv= 2.5V
.param vp= 2.5V
.param vpf= 1.25V                                         // the fast clock has a lower voltage
.param gg= 0V
.param vnf=-1.25V                                         // the fast clock has a lower voltage
.param vn=-2.5V

*** TIME SCALING
.param ticks=299                                          // 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

.param RmC=0.539*tick                                     // ramp length of clock (used to be .49 but was ajusted to match 2 pulses per 20 us and barrel.cir)
.param RmP=0.19*tick                                      // ramp length of P signal
.param Gap=0.005*tick                                     // one Gap at beginning and end of sequence, two of these gaps between ramps

// The clocks comprise a series transistions (separated by gaps). 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 gap.
.param plup=Gap                                           // the gap at either end of the sequence is half the size of gaps in the middle
.param clup=plup+RmP+2*Gap                                // phase 1 slow clock
.param c1fu=c1up+RmC+2*Gap                                // phase 1 fast clock
.param c5up=c1fu+RmC+2*Gap
.param p3dn=c5up+RmC+2*Gap
.param c3dn=p3dn+RmP+2*Gap
.param c4dn=c3dn+RmC+2*Gap
.param p2up=c4dn+RmC+2*Gap
.param c2up=p2up+RmP+2*Gap
.param c6up=c2up+RmC+2*Gap                                // phase 6 slow clock
.param c6fu=c6up+RmC+2*Gap                                // phase 6 fast clock
.param p1dn=c6fu+RmC+2*Gap
.param c1fd=p1dn+RmP+2*Gap                                // phase 1 fast clock
.param c1dn=c1fd+RmC+2*Gap                                // phase 1 slow clock
.param c5dn=c1dn+RmC+2*Gap
.param p3up=c5dn+RmC+2*Gap
.param c3up=p3up+RmP+2*Gap
.param c4up=c3up+RmC+2*Gap
.param p2dn=c4up+RmC+2*Gap
.param c2dn=p2dn+RmP+2*Gap
.param c6fd=c2dn+RmC+2*Gap                                // phase 6 fast clock
.param c6dn=c6fd+RmC+2*Gap                                // phase 6 slow clock
.param epoc=c6dn+RmC+2*Gap

* Page 49 Fig 4.8 Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic by Saed G. Younis
* However, we define the clocking sequence to start at the third vertical dashed line in figure 4.8
* For these clocks, initialize the nodes as output of PREVIOUS stage, gg, inverse of stage output, and (actually initialized in the next gate) stage output
Vphi1P 10 0 DC {gg} PWL({0} {gg}                          {clup} {gg} {clup+RmC} {vp}     {c1dn} {vp} {c1dn+RmC} {gg}     {epoc} {gg}) r={0}
Vphi1N 11 0 DC {gg} PWL({0} {gg}                          {clup} {gg} {clup+RmC} {vn}     {c1dn} {vn} {c1dn+RmC} {gg}     {epoc} {gg}) r={0}
Vph1fP 28 0 DC {gg} PWL({0} {gg}                          {c1fu} {gg} {c1fu+RmC} {vpf}    {c1fd} {vpf} {c1fd+RmC} {gg}    {epoc} {gg}) r={0}
Vph1fN 29 0 DC {gg} PWL({0} {gg}                          {c1fu} {gg} {c1fu+RmC} {vnf}    {c1fd} {vnf} {c1fd+RmC} {gg}    {epoc} {gg}) r={0}
Vphi2P 12 0 DC {gg} PWL({0} {gg}                          {c2up} {gg} {c2up+RmC} {vp}     {c2dn} {vp} {c2dn+RmC} {gg}     {epoc} {gg}) r={0}
```

```
Vphi2N 13 0 DC {gg} PWL({0} {gg}                                    {c2up} {gg} {c2up+RmC} {vn}     {c2dn} {vn} {c2dn+RmC} {gg}     {epoc} {gg}) r={0}
Vphi3P 14 0 DC {vp} PWL({0} {vp} {c3dn} {vp} {c3dn+RmC} {gg}        {c3up} {gg} {c3up+RmC} {vp}                                     {epoc} {vp}) r={0}
Vphi3N 15 0 DC {vn} PWL({0} {vn} {c3dn} {vn} {c3dn+RmC} {gg}        {c3up} {gg} {c3up+RmC} {vn}                                     {epoc} {vn}) r={0}

Vphi4P 16 0 DC {vp} PWL({0} {vp} {c4dn} {vp} {c4dn+RmC} {gg}        {c4up} {gg} {c4up+RmC} {vp}                                     {epoc} {vp}) r={0}
Vphi4N 17 0 DC {vn} PWL({0} {vn} {c4dn} {vn} {c4dn+RmC} {gg}        {c4up} {gg} {c4up+RmC} {vn}                                     {epoc} {vn}) r={0}
Vphi5P 18 0 DC {gg} PWL({0} {gg}                                    {c5up} {gg} {c5up+RmC} {vp}     {c5dn} {vp} {c5dn+RmC} {gg}     {epoc} {gg}) r={0}
Vphi5N 19 0 DC {gg} PWL({0} {gg}                                    {c5up} {gg} {c5up+RmC} {vn}     {c5dn} {vn} {c5dn+RmC} {gg}     {epoc} {gg}) r={0}
Vphi6P 26 0 DC {gg} PWL({0} {gg}                                    {c6up} {gg} {c6up+RmC} {vp}     {c6dn} {vp} {c6dn+RmC} {gg}     {epoc} {gg}) r={0}
Vphi6N 27 0 DC {gg} PWL({0} {gg}                                    {c6up} {gg} {c6up+RmC} {vn}     {c6dn} {vn} {c6dn+RmC} {gg}     {epoc} {gg}) r={0}
Vph6fP 30 0 DC {gg} PWL({0} {gg}                                    {c6fu} {gg} {c6fu+RmC} {vpf}    {c6fd} {vpf} {c6fd+RmC} {gg}    {epoc} {gg}) r={0}
Vph6fN 31 0 DC {gg} PWL({0} {gg}                                    {c6fu} {gg} {c6fu+RmC} {vnf}    {c6fd} {vnf} {c6fd+RmC} {gg}    {epoc} {gg}) r={0}

Vp1P   20 0 DC {vn} PWL({0} {vn}                                    {p1up} {vn} {p1up+RmP} {vp}     {p1dn} {vp} {p1dn+RmP} {vn}     {epoc} {vn}) r={0}
Vp1N   21 0 DC {vp} PWL({0} {vp}                                    {p1up} {vp} {p1up+RmP} {vn}     {p1dn} {vn} {p1dn+RmP} {vp}     {epoc} {vp}) r={0}
Vp2P   22 0 DC {vn} PWL({0} {vn}                                    {p2up} {vn} {p2up+RmP} {vp}     {p2dn} {vp} {p2dn+RmP} {vn}     {epoc} {vn}) r={0}
Vp2N   23 0 DC {vp} PWL({0} {vp}                                    {p2up} {vp} {p2up+RmP} {vn}     {p2dn} {vn} {p2dn+RmP} {vp}     {epoc} {vp}) r={0}
Vp3P   24 0 DC {vp} PWL({0} {vp} {p3dn} {vp} {p3dn+RmP} {vn}        {p3up} {vn} {p3up+RmP} {vp}                                     {epoc} {vp}) r={0}
Vp3N   25 0 DC {vn} PWL({0} {vn} {p3dn} {vn} {p3dn+RmP} {vp}        {p3up} {vp} {p3up+RmP} {vn}                                     {epoc} {vn}) r={0}

*** DEFINE NOMINAL CIRCUIT
X1 q q1 q2 r r1 r2 s s1 s2 t t1 t2 q 10 11 12 13 14 15 16 17 18 19 26 27 20 21 22 23 24 25 99 97 SHIFTREG ini0=vn ini1=vp ini2=vn ini3=vp
X2 u u1 u2 v v1 v2 w w1 w2 x x1 x2 u 10 11 12 13 14 15 16 17 18 19 26 27 20 21 22 23 24 25 99 97 SHIFTREG ini0=vn ini1=vp ini2=vn ini3=vp

X3 a a1 a2 b b1 b2 c c1 c2 d d1 d2 aq 28 29 12 13 14 15 16 17 18 19 30 31 20 21 22 23 24 25 99 97 SHIFTREG ini0=vp ini1=vn ini2=vp ini3=vp
X4 e e1 e2 f f1 f2 g g1 g2 h h1 h2 ex 28 29 12 13 14 15 16 17 18 19 30 31 20 21 22 23 24 25 99 97 SHIFTREG ini0=vn ini1=vn ini2=vp ini3=vn
X5 i i1 i2 j j1 j2 k k1 k2 l l1 l2 iq 28 29 12 13 14 15 16 17 18 19 30 31 20 21 22 23 24 25 99 97 SHIFTREG ini0=vp ini1=vp ini2=vp ini3=vn
X6 m m1 m2 n n1 n2 o o1 o2 p p1 p2 mq 28 29 12 13 14 15 16 17 18 19 30 31 20 21 22 23 24 25 99 97 SHIFTREG ini0=vp ini1=vn ini2=vn ini3=vn

x7 q u                                                  // Args: Two address inputs, q, u
// 0. vn vn --> starting point does not shift
// 1. vn vp --> starting point shifts up one
// 2. vp vn --> starting point shifts up two
// 3. vp vp --> starting point shifts up three (down one)
//vn vp --> skip 2; vp vn --> -1
+ aq ex iq mq                                          // Four inputs, a b c d
+ a e i m                                              // Four outputs, w x y z; circular rotation of inputs
+ 94 93 ROT2x4

* power and energy calculation
B4 0 38 V=I(Vphi1P)*v(10)+I(Vphi1N)*v(11)+
+ I(Vph1fP)*v(28)+I(Vph1fN)*v(29)+
+ I(Vphi2P)*v(12)+I(Vphi2N)*v(13)+I(Vphi3P)*v(14)+I(Vphi3N)*v(15)+I(Vphi4P)*v(16)+I(Vphi4N)*v(17)+I(Vphi5P)*v(18)+I(Vphi5N)*v(19)+I(Vphi6P)*v(26)+I(Vphi6N)*v(27)
A1 38 39 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
.TRAN {tstep} {ticks*tick}

* use BSIM3 model with default parameters
.model n1 nmos level=49 version=3.3.0
.model p1 pmos level=49 version=3.3.0

.control
pre_set strict_errorhandling
unset ngdebug
run

* measure power consumption
meas tran Energylus INTEG v(38) from=0 to=5us
meas tran EnergyLev INTEG v(38) 'from=0us to=ttn'
echo results lus=$&Energylus all=$&EnergyLev

* clocks positive then negative, same order as figure 4.6
*plot v(26) v(27) v(24)+1 v(25)+1 v(22)+2 v(23)+2 v(20)+3 v(21)+3 v(16)+4 v(17)+4 v(14)+5 v(15)+5 v(12)+6 v(13)+6 v(10)+7 v(11)+7
*plot v(26)/3 v(27)/3 v(24)/3+.5 v(25)/3+.5 v(22)/3+1 v(23)/3+1 v(20)/3+1.5 v(21)/3+1.5 v(16)/3+2 v(17)/3+2 v(14)/3+2.5 v(15)/3+2.5 v(12)/3+3 v(13)/3+3 v(10)/3+3.5
v(11)/3+3.5 v(28)+5 v(a1)+6 v(a2)+7 v(a3)+8 v(b)+9 v(b1)+10 v(b2)+11 v(B3)+12

* 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=3

set gnuplot_terminal=png
//echo test filename file0
//set fn file0
gnuplot 0 //v(a)-1 v(24)/15 v(25)/15 v(26)/15+.5 v(27)/15+.5 v(b2)+1.5 v(22)/15+2.5 v(23)/15+2.5 v(18)/15+3 v(19)/15+3 v(b1)+4 v(20)/15+5 v(21)/15+5 v(16)/15+5.5 v(17)/15+5.5
v(b)+6.5 v(24)/15+7.5 v(25)/15+7.5 v(14)/15+8 v(15)/15+8 v(a2)+9 v(22)/15+10 v(23)/15+10 v(12)/15+10.5 v(13)/15+10.5 v(a1)+11.5 v(20)/15+12.5 v(21)/15+12.5 v(10)/15+13
v(11)/15+13 v(a)+14
//plot                                                  // plot clocks like Younis's thesis Fig 4.8
+ v(10)/15+13.0 v(11)/15+13.0
+ v(28)/15+13.0 v(29)/15+13.0
+ v(12)/15+12.5 v(13)/15+12.5
+ v(14)/15+12.0 v(15)/15+12.0
+ v(16)/15+11.5 v(17)/15+11.5
+ v(18)/15+11.0 v(19)/15+11.0
+ v(26)/15+10.5 v(27)/15+10.5
+ v(30)/15+10.5 v(31)/15+10.5
+ v(20)/15+10.0
+ v(21)/15+10.0
+ v(22)/15+ 9.5
+ v(23)/15+ 9.5
+ v(24)/15+ 9.0
+ v(25)/15+ 9.0

+ (-1)*v(a )/15+ 8.0
+ ( 1)*v(a1)/15+ 7.5
+ (-1)*v(a2)/15+ 7.0

+ ( 1)*v(b )/15+ 6.0
+ (-1)*v(b1)/15+ 5.5
+ ( 1)*v(b2)/15+ 5.0

+ (-1)*v(c )/15+ 4.0
+ ( 1)*v(c1)/15+ 3.5
+ (-1)*v(c2)/15+ 3.0

+ ( 1)*v(d )/15+ 2.0
+ (-1)*v(d1)/15+ 1.5
+ ( 1)*v(d2)/15+ 1.0

plot title "SCRL clock and data traces" ylimit 0 12
+ xlimit 0 300u
+ ( 1)*v(a)/.66+ 4+0.1                                 // non-inverted output
+ ( 1)*v(e)/.66+ 6+0.1
```

```
+ ( 1)*v(i)/.66+ 8+0.1
+ ( 1)*v(m)/.66+10+0.1
+     v(10)/2.0+ 1+0.1                                  // X[1-6] is Linearshift
+     v(11)/2.0+ 1+0.1                                  // X[1-6] is Linearshift

* plot instantaneous energy consumption
*plot v(38)
* plot accumulated energy dissipation
*plot v(39)

.endc

.END
```

## B2tst.cir

```
  Pass Gate Tester
* Tests the dual voltage system for NFET-only stages
//.temp -269.15

*** SUBCIRCUIT DEFINITIONS
.SUBCKT PASS D GT GC S psub nsub                       // Pass gate. Args: Drain GateT/C Source psub nsub
voffn GT tn -1.1                                       // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp .7
M1 i1 tn v2 nsub n1
M2 i1 tp v2 psub p1
R1 D v1 0                                              // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                              // v1 and v2 are voltages on both sides of the channel
.ENDS PASS

.SUBCKT NPASS D GT GC S psub nsub                      // NFET-only reduced pass gate. Args: Drain GateT/C Source psub nsub
voffn GT tn -1.8                                       // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp 0
M1 i1 tn v2 nsub n1
//M2 i1 tp v2 psub p1
R1 D v1 0                                              // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                              // v1 and v2 are voltages on both sides of the channel
.ENDS NPASS

*** POWER
VCC   99  0   DC {vv}                                  // Full range that can only get through a full pass stage
VGG   98  0   DC {gg}

VCCf  89  0   DC {vf}                                  // Reduced range that can get through an NFET stage
VGGf  88  0   DC {gf}

VPS   79  0   DC { 8.0 }                               // Substrate bias; overhangs by .75 V
VNS   78  0   DC {-2 }

*** ALL INPUTS
.param gg=0V                                           // range allowed from a full pass gate stage
.param vv=6V

.param gf=0V                                           // range allowed from an NFET-only stage
.param vf=4.6V

.param g2=0V//-.375V
.param v2=6V

.param tick=2000NS                                     // time of a tick

*** CLOCKS -- Original 4 clock phases and inverses (total four unique signal), but with slow and fast phase 1's (total six unique signals)
vph0  10  0   DC {gf} PWL({0*tick} {gf} {1*tick} {vf} {2*tick} {gf}) r={0*tick}
vph1  11  0   DC {g2} PWL({0*tick} {g2} {1*tick} {v2} {2*tick} {g2}) r={0*tick}

*** TOP-LEVEL CIRCUIT
X1 10 99 98 50 79 78 NPASS                             // NFET on
c1 50 60 .2p
r1 60 0 1
X2 10 98 99 51 79 78 NPASS                             // NFET off
c2 51 61 .2p
r2 61 0 1

X3 11 89 88 52 79 78 PASS                              // pass gate on
c3 52 62 .2p
r3 62 0 1
X4 11 88 89 53 79 78 PASS                              // pass gate off
c4 53 63 .2p
r4 63 0 1

.option noinit acct
.TRAN {10ns} {4*tick}

* use BSIM3 model with default parameters
.model n1 nmos level=49 version=3.3.0
.model p1 pmos level=49 version=3.3.0

.control
pre_set strict_errorhandling
unset ngdebug
run


**********************************************
// NOTE: plots of V(10) and V(11) are moved down by .05 V so the curves don't sit exactly on top of each other

plot title "reduced range" v(10)-.05 v(50) v(51)        // plot drive and on/off output
plot title "reduced range current" v(60) v(61) I(vph0)  // currents

plot title "restored range" v(11)-.05 v(52) v(53)       // plot drive and on/off output
plot title "restored range current" v(62) v(63) I(vph1) // currents

plot title "substrate current" I(vns) I(vps)           // currents

.endc

.EN
```

## Bstst.cir

```
  Pass Gate Tester
* Tested with ngspice-30 (creation date Dec 28, 2018 10:51:19, from ngspice-30_64.zip 8,687,648 bytes)
* For tutorial docs: no tabs; comments start column 61; 169 character maximum line length
* Uses built in BSIM ver 4.4 SOI model, but with parameters k1=1.5 k2=0
```

```
.MODEL pch pmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
.MODEL nch nmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
//.temp 25                                              // Room temperature
//.temp -196.15                                         // 77 K - 273.15 K
// The next line can produce the following error message: "New prediction returns to 0.0! Alberto says: YOU TURKEY! The limiting function received NaN"
//.temp -238.15                                         // 35 K - 273.15 K
.temp -180

*** SUBCIRCUIT DEFINITIONS
.SUBCKT PPASS D GT GC S pBak nBak                       // Pass gate. Args: Drain GateT/C Source p/nBackgate. Backgate also AC ground for substrate
voffn GT tn 0                                           // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp 0
//M1 i1 tn v2 nBak nBak nch
M2 i1 tp v2 pBak pBak pch
R1 D v1 0                                               // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                               // v1 and v2 are voltages on both sides of the channel
R4 i1 v2 1e8                                            // this resistor protects against ngspice failing to converge
.ENDS PPASS

.SUBCKT NPASS D GT GC S pBak nBak                       // NFET-only reduced pass gate. Args: Drain GateT/C Source p/nBackgate. Backgate also AC ground for substrate
voffn GT tn 0                                           // Gate differential; shifts threshold voltage by equivalent of a battery in series with the gate
voffp GC tp 0
M1 i1 tn v2 nBak nBak nch
//M2 i1 tp v2 pBak pBak pch
R1 D v1 0                                               // v1 and i1 are a one ohm current sensor on one side of the pass gate
R2 v1 i1 1
R3 S v2 0                                               // v1 and v2 are voltages on both sides of the channel
R4 i1 v2 1e8                                            // this resistor protects against ngspice failing to converge
.ENDS NPASS

*** POWER
VCC   99  0   DC {vv}                                   // Full range that can only get through a full pass stage
VGG   98  0   DC {gg}
VNN   97  0   DC {vn}
VBP   94  0   DC {vpf-Back}                             // Back gate voltages
VBN   93  0   DC {vnf+Back}

*** ALL INPUTS
// high voltage swing: -2.5 V to +2.5 V or 5 V
// reduced voltage swing -1.25 V to +1.25 V or 2.5 V
// pFET backgate -2.25 V or 3.5 V below the maximum reduced swing of +1.25 V
// nFET backgate +2.25 V or 3.5 V above the minimum reduced swing of -1.25
.param Back=3.5                                         // for nFET(pFET), this is the back gate voltage referenced to the smallest(largest) source voltage during
operation

.param vv= 2.5                                          // positive gating signal voltage -- largest swing
.param vpf= 1.25                                        // positive extent of smaller signal swing
.param gg= 0
.param vnf=-1.25                                        // negative extent of smaller signal swing
.param vn=-2.5                                          // negative gating signal voltage -- largest swing

.param tick=2000NS                                      // time of a tick

*** CLOCKS -- Original 4 clock phases and inverses (total four unique signal), but with slow and fast phase 1's (total six unique signals)
vph0  10  0   DC {vnf} PWL({0*tick} {vnf} {1*tick} {vpf} {2*tick} {vnf}) r={0*tick}

*** TOP-LEVEL CIRCUIT
X1 10 99 97 50 94 93 NPASS                              // NFET on
c1 50 60 .2p
r1 60 0 1
X2 10 97 99 51 94 93 NPASS                              // NFET off
c2 51 61 .2p
r2 61 0 1

.if (1)

X3 10 99 97 52 94 93 PPASS                              // pFET on
X4 52 99 97 53 94 93 NPASS                              // nFET on
X5 53 99 97 54 94 93 PPASS                              // pFET on
X6 54 99 97 55 94 93 NPASS                              // nFET on
X7 55 99 97 56 94 93 PPASS                              // pFET on
X8 56 99 97 57 94 93 NPASS                              // nFET on
//c4 57 62 200p                                           // this is a huge load designed to create an interesting plot
c4 57 62 .2p                                            // this is the nominal load
r4 62 0 1
r5 63 62 1

.else

X3 10 99 97 52 94 93 PPASS                              // PFET on
c3 52 62 .2p
r3 62 0 1
X4 10 97 99 53 94 93 PPASS                              // PFET off
c4 53 63 .2p
r4 63 0 1

r5 53 54 1                                              // these resistors are just to make the plot work when the .if changes
r6 53 55 1
r7 53 56 1
r8 53 57 1

.endif

.option noinit acct
.TRAN {50ns} {5*tick}

.control
save all @m.x3.m2[vth] @m.x4.m1[vth] @m.x5.m2[vth] @m.x5.m1[vth] @m.x7.m2[vth] @m.x8.m1[vth] // see ngspice manual page 614
pre_set strict_errorhandling
unset ngdebug
run
//set color0=white                                        // white background
//set color1=black                                        // black grid and text (only needed with X11, automatic with MS Win)
//set xbrushwidth=3                                       // wider grid and plot lines
plot title "reduced range nFET on/off" ylimit -3 3 v(10) v(50) v(51) // plot drive and on/off output
plot title "reduced range nFET current" v(60) v(61) I(vph0) // currents
plot title "nFET-pFET chain on" ylimit -3 3 v(10) v(52) v(53) v(54) v(55) v(56) v(57)     // plot drive and on/off output
plot title "nFET and pFET current" v(62) v(63) I(vph0)      // currents
//plot title "reduced range pFET on/off" ylimit -3 3 v(10) v(52) v(53) // plot drive and on/off output
//plot title "reduced range pFET current" v(62) v(63) I(vph0) // currents
//plot title "substrate current" I(VBP) I(VBN)              // currents
plot title "Thresholds in chain" @m.x3.m2[vth] @m.x4.m1[vth] @m.x5.m2[vth] @m.x6.m1[vth] @m.x7.m2[vth] @m.x8.m1[vth]
.endc
.END
```

## Soi.cir

```
// SOI pass transistor simulation
// The nFET control gate's range is 0..Vmax and pFET Vmax..0
// There are curves for different values of the pass gate's back bias, Vbak.
// The signal input is applied to the pass gate's source; the source and drain are biased at 50 mV and current is reported

.MODEL pch pmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
.MODEL nch nmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
//.temp 25                                                  // Room temperature
//.temp -196.15                                             // 77 K - 273.15 K
// The next line can produce the following error message: "New prediction returns to 0.0! Alberto says: YOU TURKEY! The limiting function received NaN"
//.temp -238.15                                             // 35 K - 273.15 K
.temp -180

.param Vmax=5                                               // gating signal range 0..Vmax
.param Vsd=.05                                              // source-drain voltage, say 50 mV for small signal
.para lQ=0.2u wn=1.0u dn=2.0u

Vddn Dr_n S_np DC {Vsd}                                     // drain voltage flips depending on whether the device is an nFET or pFET
Vddp S_np Dr_p DC {Vsd}
Vsbn Sb_n 0 DC {-1}                                         // substrate goes to a pure capacitor in the SOI device model (so shouldn't matter)
Vsbp 0 Sb_p DC {-1}
Vctn Gt_n 0 DC {VMax}                                       // gate voltages for pass gate on; reverse _n and _p for pass gate off
Vctp Gt_p 0 DC {0}
Vdrn S_np 0 DC {0}                                          // overridden by the parameter sweep
Vbkn Bk_n 0 DC {0}                                          // overridden by the parameter sweep; creates back bias for nFET
E1 Bk_x 0 Bk_n 0 -1                                         // flip polarity
Vbkp Bk_p Bk_x DC {Vmax}                                    // shift to Vmax rail for pFET back bias

m1 S_np Gt_n Dr_n Sb_n Bk_n nch //w=wn l=lQ ad='wn*dn' pd= '(wn+dn)*2'
m2 S_np Gt_p Dr_p Sb_p Bk_p pch //w=wn l=lQ ad='wn*dn' pd= '(wn+dn)*2'

// Create 11 back bias curves ranging from -3 V to 3 V; this is the range described in the Web document
// soiconsortium.eu/wp-content/uploads/2017/08/AnalogRF_28FDSOI_ACathelin_19092017.pdf.
// The range is defined relative to the source, which is deemed to be 0 V and Vmax here.
// But since ngspice cannot generate two voltages for one sweep, we need a voltage offset.
// Also, this circuit will never cause sources to reach the boundary of 0..Vmax, so perhaps the back bias could range further.
//.dc Vdrn 0 {Vmax} {Vmax/100} Vbkn {-3} {3} {.6}           // 10+1 curves of different back bias
.dc Vdrn 0 {Vmax} {Vmax/100} Vbkn {4} {6} {.2}              // 10+1 curves of different back bias

.control
save all @m1[vth] @m2[vth]                                  // see ngspice manual page 614
run
//set color0=white                                          // white background
//set color1=black                                          // black grid and text (only needed with X11, automatic with MS Win)
//set xbrushwidth=3                                         // wider grid and plot lines
plot title "nFET current" xlimit 0 5 ylimit 1e-18 1e-3 ylog 'abs(Vddn#branch)+1e-25'
plot title "nFET environment" xlimit 0 5 V(S_np) V(Gt_n) V(Dr_n) V(Bk_n) '10000*(abs(Vddn#branch)+1e-25)'
plot title "pFET current" xlimit 0 5 ylimit 1e-18 1e-3 ylog 'abs(Vddp#branch)+1e-25'
plot title "pFET environment" xlimit 0 5 V(S_np) V(Gt_p) V(Dr_p) V(Bk_p) '10000*(abs(Vddp#branch)+1e-25)'
plot @m1[vth] @m2[vth]
.endc
.end
```

## Bp.cir

```
  SOI pFET tester
* 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
* Uses built in BSIM ver 4.4 SOI model, but with parameters k1=1.5 k2=0

.MODEL pch pmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
.MODEL nch nmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
//.temp 25                                                  // Room temperature
//.temp -196.15                                             // 77 K - 273.15 K
// The next line can produce the following error message: "New prediction returns to 0.0! Alberto says: YOU TURKEY! The limiting function received NaN"
//.temp -238.15                                             // 35 K - 273.15 K
.temp -180

* Floating-body SOI nMOSFET
vg1 g1 g 0                                                  // gate offset to be in series with sweep
vg2 g2 g 0

vb1 b b1  1                                                 // back bias offset
vb2 b b2 -1

* Floating-body SOI nMOSFET
mn_bf1   d1   g1   s   e          pch   w=10u l=5u         // nFETs are default size; pFETs are double width
rf1 d d1 1

* Grounded -body SOI nMOSFET
mn_bc2   d2   g1   s   e    b1    pch   w=10u l=5u
rg2 d d2 1

* Floating-body SOI nMOSFET
mn_bf3   d3   g2   s   e          pch   w=10u l=5u
rf3 d d3 1

* Grounded -body SOI nMOSFET
mn_bc4   d4   g2   s   e    b2    pch   w=10u l=5u
rg4 d d4 1

vg   g   0     dc  0.0
vd   d   0     dc  0.0
vs   s   0     dc  0.0
ve   e   0     dc  0.0
vb   b   0     dc  0.0
.dc   vd   0   -3.6   -0.01                                 // sweep drain voltage
//+ vb -1 1 .2                                              // sweep back bias
+ vg   -2.4   -4   -0.4

.control
run
//set color0=white                                          // white background
//set color1=black                                          // black grid and text (only needed with X11, automatic with MS Win)
//set xbrushwidth=3                                         // wider grid and plot lines
plot title "pFET floating and 1 V back bias" '(abs(d-d1)+1e-25)' '(abs(d-d2)+1e-25)' ylimit 0 .002
plot title "pFET floating and -1 V back bias" '(abs(d-d3)+1e-25)' '(abs(d-d4)+1e-25)' ylimit 0 .002
.endc

.end
```

## Bn.cir

```
  SOI nFET tester
* 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
* Uses built in BSIM ver 4.4 SOI model, but with parameters k1=1.5 k2=0

.MODEL pch pmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
.MODEL nch nmos(LEVEL=58 VERSION=4.4 k1=1.5 k2=0)
//.temp 25                                                  // Room temperature
//.temp -196.15                                             // 77 K - 273.15 K
// The next line can produce the following error message: "New prediction returns to 0.0! Alberto says: YOU TURKEY! The limiting function received NaN"
//.temp -238.15                                             // 35 K - 273.15 K
.temp -180

* Floating-body SOI nMOSFET
vg1 g1 g 0                                                 // gate offset to be in series with sweep
vg2 g2 g 0

vb1 b b1 -1                                                // back bias offset
vb2 b b2 1

* Floating-body SOI nMOSFET
mn_bf1   d1   g1   s   e           nch   w=5u l=5u          // nFETs are default size; pFETs are double width
rf1 d d1 1

* Grounded -body SOI nMOSFET
mn_bc2   d2   g1   s   e    b1     nch   w=5u l=5u
rg2 d d2 1

* Floating-body SOI nMOSFET
mn_bf3   d3   g2   s   e           nch   w=5u l=5u
rf3 d d3 1

* Grounded -body SOI nMOSFET
mn_bc4   d4   g2   s   e    b2     nch   w=5u l=5u
rg4 d d4 1

vg   g   0     dc  0.0
vd   d   0     dc  0.0
vs   s   0     dc  0.0
ve   e   0     dc  0.0
vb   b   0     dc  0.0
.dc   vd   0    3.6    0.01                                // sweep drain voltage
//+ vb -1 1 .2                                             // sweep back bias
+ vg    2.4    4    0.4

.control
run
//set color0=white                                         // white background
//set color1=black                                         // black grid and text (only needed with X11, automatic with MS Win)
//set xbrushwidth=3                                        // wider grid and plot lines
plot title "nFET floating and -1 V back bias" '(abs(d-d1)+1e-25)' '(abs(d-d2)+1e-25)' ylimit 0 .002
plot title "nFET floating and 1 V back bias" '(abs(d-d3)+1e-25)' '(abs(d-d4)+1e-25)' ylimit 0 .002
.endc

.end
```