Dual-polarity peak detector operates from single supply

Chester Simpson, National Semiconductor, Santa Clara, CA

The circuit in **Fig 1** can detect the peak of either positive or negative inputs, yet it operates from a single supply. The circuit uses two unity-gain amplifiers to monitor the input signal, one an inverting amplifier and the other a noninverting amplifier. A typical application is driving a VU meter.

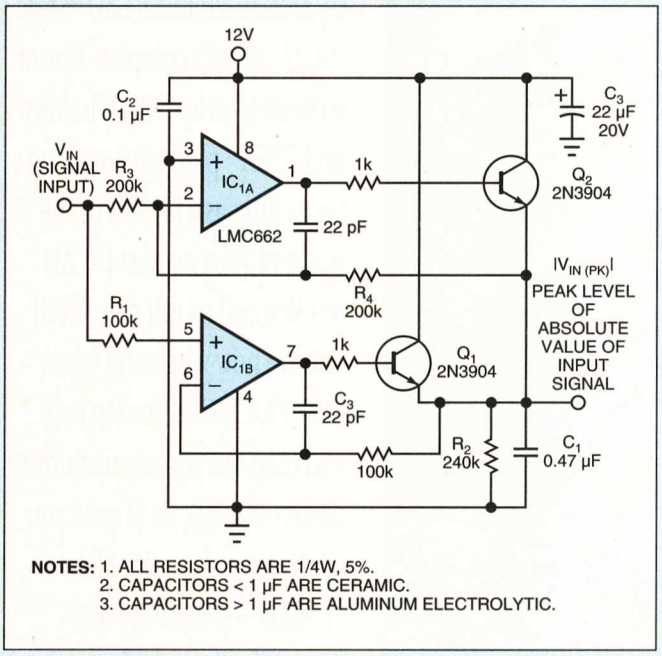

Fig 1—This circuit can detect the peak of positive or negative inputs, yet it operates from a single supply. The transistors boost the outputs of the op amps, ensuring a fast slew rate for the circuit's output.

The design takes advantage of a characteristic of the LMC662, IC_1. Although the op amp's data sheet warns against pulling inputs below ground, the op amp can withstand negative inputs if you limit the input current. In this circuit, R_1 limits input current to a very small amount. The op amp does nothing uncivilized, such as latching up or slamming against both rails, in this mode. It just waits patiently with its output at ground potential until the input signal goes positive and then starts following the input again. If you wish to substitute another op amp, check its performance first.

The outputs of the two amplifiers go through a "diode-OR" connection to C_1. Resistor R_2 bleeds off the voltage on C_1, determining the decay time of the output. The series path through R_1 and R_4 bleeds some current from C_1 but to negligible effect.

Q_1 and Q_2 ensure that current starving does not degrade the slew rate of the circuit. The transistors "turbocharge" the output of the op amps. Instead of the 20 to 30 mA available from the typical op amps, the 2N3904s can supply a peak current of 200 to 300 mA.

The 22-pF capacitors on the output of the op amps are necessary for stability because CMOS op amps do not like to drive capacitive loads. Locate capacitor C_2 near IC_1 because it bypasses the supply lead. C_3 provides the peak currents that Q_1 and Q_2 use to charge capacitor C_1. (DI #1415) EDN

To Vote For This Design, Circle No. 359

Verilog program models metastable flip-flop

Swapnajit Mittra, Wipro Infotech Ltd, Bangalore, Karnataka, India

The Verilog HDL program in **Listing 1** allows you to simulate the behavior of a set-reset (SR) flip-flop that has both its set and reset inputs high simultaneously. The outputs of a physical SR flip-flop become indeterminate in this condition. This property is the basis of circuits such as random-number and pseudo-noise-sequence (PSN) generators.

Simulating this simple circuit, however, poses problems. Software generates random numbers either by a predetermined sequence, such as linear congruential method, or by using a random parameter, such as the system time, as a seed. These methods require the software to map a range of numbers (probably floating point) to 1 or 0. These methods also need two software modules: one for the indeterminate case and another for normal behavior.

Verilog HDL provides system calls for generating random numbers whose statistical properties (distribution, mean, mode, etc) you can set very accurately. In particular, the system call *$random* generates a random number from the set

Listing 1—Behavioral model of set-reset flip-flop

```
module sr_flipflop(Q, QB, R, S, CP);
        input S, R, CP;
        output Q, QB;
        reg Q, QB;

        initial
        begin

                Q = $random;
                QB = $random;
                   // At $time = 0, Q and QB are both random
        end

        always @(posedge CP)
        begin
                if (S & R == 1'b1)
                        begin
                        Q =  $random ;
                        QB = $random ;
                     // Q and QB are random values
                        end
                 else
                        begin
                        fork
                        Q =  #1 S|(~R & Q);
                        QB = #1 ~(S)|(~R & Q@-{1});
                        join
                        end
          edn
endmodule
```