design ideas

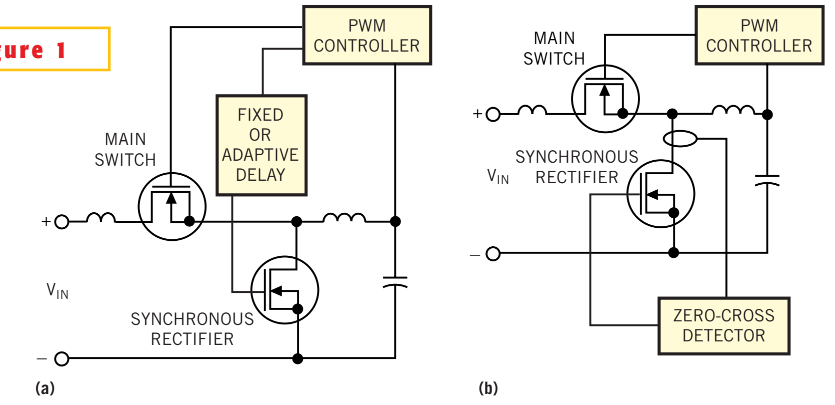
Edited by Bill Travis

Method provides self-timing for synchronous rectifiers

Giampaolo Carli, SAE Power Co, Saint-Lazare, Quebec, PQ, Canada

Synchronous rectifiers are MOSFETs, driven in such a way as to perform a rectifying function. They often take the place of diodes in the output-rectification stage of switching power converters, because of their lower on-state power loss. In power circuits, synchronous rectifiers are often complicated to use because of timing issues. Some techniques attempt to predict the correct timing by following the same drive signal that controls the main switching element of the circuit. Other techniques sense the current in the FET in various ways and then act on that information. **Figures 1a** and **1b** show simplified representations of these alternative techniques, applied to the forward/buck topology. During the off-time of the main switch, the rectifier conducts from source to drain. At the beginning of the switching cycle, the main switch turns on and begins driving current into the rectifier. Eventually, the current in the rectifier falls to zero and begins to reverse, flowing from drain to source. This instant is the optimum time to turn off the rectifier.

Figure 1



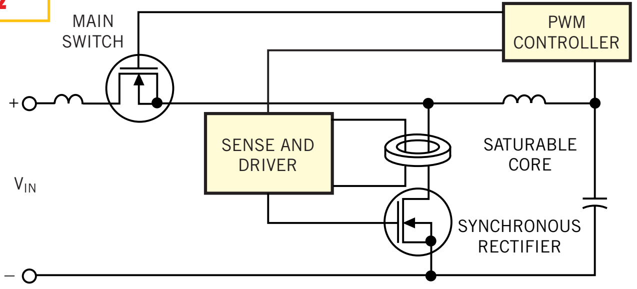
These topologies often exhibit a delayed turn-off of the synchronous rectifier, resulting in considerable reverse current. The current-reversal timing depends on loading conditions (a). The zero-cross current causes the rectifier to turn off too late (b).

Unfortunately, if the signal from the control circuit appears at this time, the synchronous rectifier turns off only after various delays (especially, the MOSFET's turn-off delay). Because a large di/dt is involved in the turn-off, the undesirable consequence is that the rectifier turns off only when considerable reverse current is flowing. If you use the concept in **Figure 1a** with a fixed delay, the turn-off of the synchronous rectifier rarely occurs at the optimum time, because the current-reversal timing depends on loading conditions. Adaptive-delay techniques that compensate for changes in delay with load are complex. The concept in **Figure 1b** has similar complications. The zero-cross current detector is often relatively slow, so, in addition to the cited FET delays, it causes the synchronous rectifier to turn off too late. **Figure 2** shows a simple way to modify the concept in **Figure 1b**. In this case, you introduce a saturable core with an additional sense winding in the drain connection of the rectifier. With some minor additional circuitry, this single component accomplishes by itself the two main functions necessary to eliminate the turn-off-delay problem.

The first function is to determine the instant that the current falls close to zero. At that time, the core comes out of saturation and blocks voltage. This voltage also appears on the additional winding, flagging to the sense circuit that it must immediately turn off the MOSFET. The second function is to significantly slow the di/dt of the current during this crucial turn-off time. The saturable core's operation allows wider tolerances in timing and more flexibility in the design of the synchronous-rectifier driver, resulting in far fewer and less expensive components. The core can be small, even for high-power applications, and should be of nonsquare-loop ferrite material. Regular power ferrite is much less expensive and lossy than its square-loop counterparts. The nonsquare-loop material allows the core to come out of saturation when the current is still slightly positive in the rectifier, thus giving a slight advance warning.

Because of the nonlinear response of the saturable core, the secondary sense winding has far fewer turns than the corresponding linear sensor in **Figure 1b** and virtually no loss in the sensing and clamping circuits for its secondary current.

Figure 2



A low-cost, saturable core in the drain circuit of the synchronous rectifier introduces a "self-timing" feature in the circuit.

These considerations improve response speed and reduce losses in the high-current outputs. Note that this simple circuit is versatile; you can apply it to various switches and rectifiers in most power-circuit topologies. The concept can even improve on well-known soft-switching techniques.

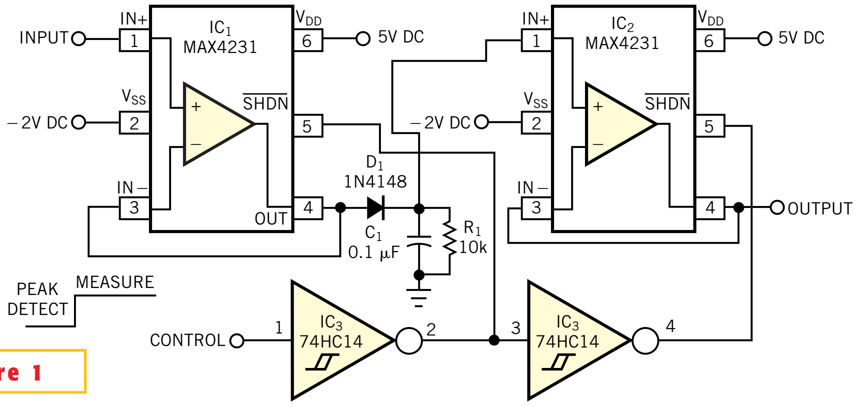
Is this the best Design Idea in this issue? Select at www.ednmag.com.

Sampling peak detector has shutdown feature

Shyam Tiwari, Sensors Technology Private Ltd, Gwalior, India

You face a serious problem in using a slow ADC with a fast peak detector. The circuit in **Figure 1** allows a slow ADC to measure a fast, sampled signal peak. The 100-MHz peak detector for ultrasonic-pulse sampling uses a fast MAX4231 amplifier from Maxim (www.maxim-ic.com). This amplifier has a shutdown feature that facilitates power savings without losing the sampled information. When the circuit samples a peak with a low-TTL-control input, the output of the peak-detector amplifier shuts off, and the output amplifier switches on to measure the output signal. This technique reduces power consumption by nearly 50%, because only one of the amplifiers is active at any given time. **Table 1** shows the circuit functions and the amplifier modes as a function of the control input's status.

Figure 1



This circuit allows you to use a slow ADC to measure fast peaks and saves power to boot.

TABLE 1—CIRCUIT FUNCTIONS AND AMPLIFIER MODES

Function	Amplifier 1	Amplifier 2	Control input
Signal-peak sampling	On	Off (shutdown)	TTL 0 (0V)
Peak measurement	Off (shutdown)	On	TTL 1 (2.4 to 5V)

The most important advantage of the circuit is that it prevents sampling of another input peak before a measurement takes place. The first peak-detector amplifier—in shutdown mode—does not permit the reading to change. This feature helps a slow ADC to monitor a high-speed sampled peak in a desired sampling interval, an impossible operation with a conventional peak-detecting circuit. You select the R_1 and C_1 values for the desired time constant to hold the peak and then let it decay according to the RC time constant. If peak decay is undesirable, then you can use a transistor switch (not shown) without R_1 to discharge the capacitor to ground before sampling a new input-signal peak.

Is this the best Design Idea in this issue? Select at www.ednmag.com.

Circuit provides bidirectional, variable-speed motor control

Jean-Bernard Guiot, DCS AG, Allschwil, Switzerland

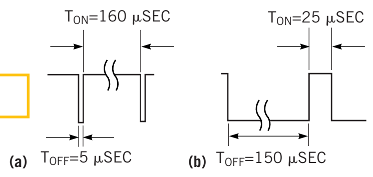
During the development of systems that include small motors, a simple, bidirectional motor controller with speed adjustment may be helpful. **Figure 1** shows such a controller. The circuit uses everyday components whose tolerances and ratings are unimportant as long as they sustain the required voltage, current, and power. The circuit's advantages are low cost, small size, flexibility, and ready availability. You can assemble it in less than an hour on a board measuring approximately 75×100 mm; its height is less than 12 mm. A transistor-based H-bridge allows two directions of rotation. A chopper controls the upper arms of the H-bridge, thereby enabling the speed adjustment. To start the rotation in one direction, you must connect one of the inputs (In CW or In CCW) to 0V. You can do this through switches, transistors, or open-collector TTL circuits, for example. If both inputs are high (no command), transistors Q_2 and Q_4 do not conduct, and the motor stops. The motor receives a slight braking action from the pulsing Q_1 and Q_3 transistors.

If one input is low (connected to 0V)—for example, In CW or In CCW—the corresponding transistor, Q_2 or Q_4, conducts, with base current limited by R_1 and R_4. The pulse signal to the base of Q_1 or Q_3 short-circuits to 0V, thus shutting off Q_1 or Q_3. On the opposite side, Q_4 or Q_2 does not conduct, but Q_3 or Q_1 receives pulses from the chopper through D_2 or D_1 and R_6 or R_5. Thus, Q_3 or Q_1 conducts each time transistor Q_5 is on. The chopper uses a 555 timer circuit, IC_1, connected as an astable multivibrator. The zener diode, D_7, and R_{10} limit IC_1's power-supply voltage to the maximum allowable: 15V. The timing capacitor, C_2, charges through R_{11}, the upper part of the potentiometer R_{12}, and zener diode D_7. The discharge takes place through the lower part of R_{12}. Using β for the position of R_{12}'s wiper (middle: β=0.5; down: β=0; up: β=1), the charging time is $T_{ON}=0.693C[R_{11}+R_{12}(1-\beta)]$, and the discharge time is $T_{OFF}=0.693\beta CR_{12}$.

The total time of one period of the chopper is thus $T_{ON}+T_{OFF}=0.693C[R_{11}+R_{12}(1-\beta+\beta)]=0.693C(R_{11}+R_{12})$. The output signal on Pin 3 is a square wave with nearly fixed frequency and adjustable duty cycle. In **Figure 2a**, the potentiometer's wiper is fully down (β=0). In **Figure 2b**, the wiper is fully up (β=1). Q_5 and Q_6 adapt the voltage level to drive the bases of Q_1 and Q_3, which conduct only in the time when the output (Pin 3) of the 555 is high (T_{ON}). This conduction adjusts the rotational speed. Diodes D_3 to D_6 protect the transistors Q_1 to Q_4 against inductive voltage peaks. The fuse, F_1, protects the whole circuit against overcurrent conditions. Capacitor C_3 between V_{CC} and ground acts as a kind of energy tank that filters out the current peaks. The circuit was a help in determining speeds or gear ratios to use during test and adjustment of prototypes on small machine tools. The transistors should preferably be Darlington types, adapted to the power-supply voltage and the motor current. (Don't forget the high inductance of the motor.) Select resistor R_{10} and the zener diode according to the power-supply voltage, V_{CC}.

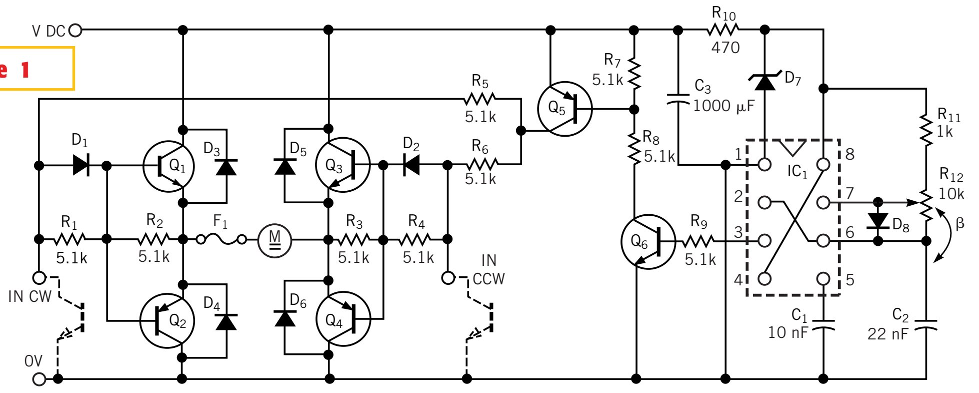
Is this the best Design Idea in this issue? Select at www.ednmag.com.

Figure 2



The position of the wiper on the speed-control potentiometer determines the duty cycle of the chopper circuit. When it is fully down, β=0 (a); when it is fully up, β=1 (b).

Figure 1



NOTES:
Q_1, Q_3=TIP142.
Q_2, Q_4=TIP147.
Q_5=BC161, BC556, 2N3906.
Q_6=BC160, BC546, 2N3904.
IC_1=555 TIMER.
D_3 TO D_6=BYV26E.
D_1, D_2, D_8=1N4148, 1N4007.
D_7=15V, 0.4W ZENER.

You can set a motor's rotational direction and speed using this simple circuit.

Software reset uses I²C I/O port

Bob Marshall, Philips Semiconductors, Sunnyvale, CA

You can use the circuit in **Figure 1** to allow the I²C or SMBus to control device resets in a system by using the PCA9554 I²C I/O-port IC. Normally, a reset function takes an active-low signal. On power-up of the PCA9554, the IC sets all the I/O pins as inputs. The 4.7-kΩ pull-down resistor on each I/O ensures that all the active-low reset pins are initially in a low state during power-up. You can now program the I²C controller to bring all or some of the external devices out of the reset condition. **Figure 2** shows additional details of the I/O internal structure. The PCA-9554 has four internal registers. Register 0 is the input-port register. Register 1 controls the output-port register. Register 2 is a polarity-inversion register, and Register 3 is the configuration register. All the registers, except Register 2, are initially set to all logic ones (high).

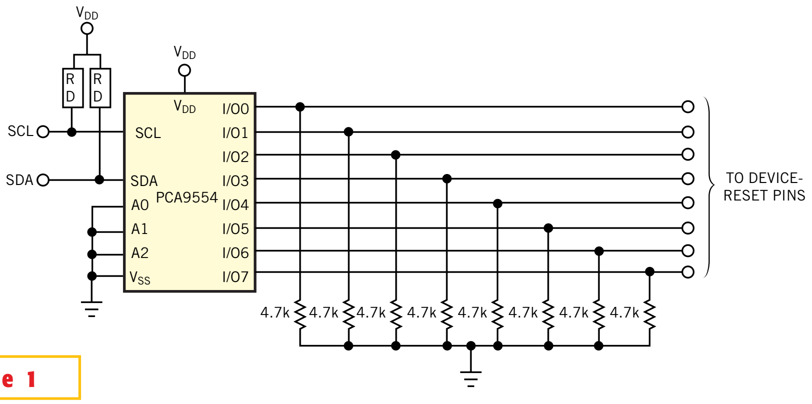
When you apply power to V_{DD}, an internal power-on reset holds the PCA9554 in its initial state until V_{DD} reaches approximately 1.5V. The power-on condition sets all the I/O pins as inputs, so Q_1 and Q_2 are off. The IC has a 100-kΩ internal pull-up resistor on each I/O pin. The 4.7-kΩ external pull-down resistors hold all the attached devices in a reset state. The I²C bus can now control which devices can come out of the reset state. The I²C bus uses unique addresses for slave devices on the bus. The PCA9554 uses an address, 0100xxx R/Wn, where xxx is the level of A2, A1, and A0. To communicate with the PCA9554, the bus master must first send the device address, a command byte that addresses one of the four internal registers, and then the data. After each byte sent from the I²C master, the slave device automatically generates an acknowledge signal on the I²C bus. The master then sends the next byte. The command sequence to bring all the devices out of reset at once is:

ST: Start bit generated by the master;
40: Write to slave address (A0 to A2 are all low in this example);
03: Write the next byte to the configuration register;
00: Write 00 to the configuration register, which sets all I/O as outputs;
SP: Stop bit generated by the bus master.

The default condition of the output register is all logic ones (high). If you want to bring devices out of the reset state one at a time, simply change the pattern written in the configuration register. Any bit left at logic one in the configuration register keeps the corresponding output low (in reset). To place any device into reset, the I²C bus simply writes a logic one into the corresponding bit in the configuration register. Another feature of the PCA9554 is that it remembers the last command byte. Subsequent writes to the configuration register require only a 2-byte operation, provided that no other command register is addressed. For example, the following command sequence brings four devices out of reset (devices attached to I/O0 through I/O3) and then, on the subsequent write, brings the rest of the four devices out of reset:

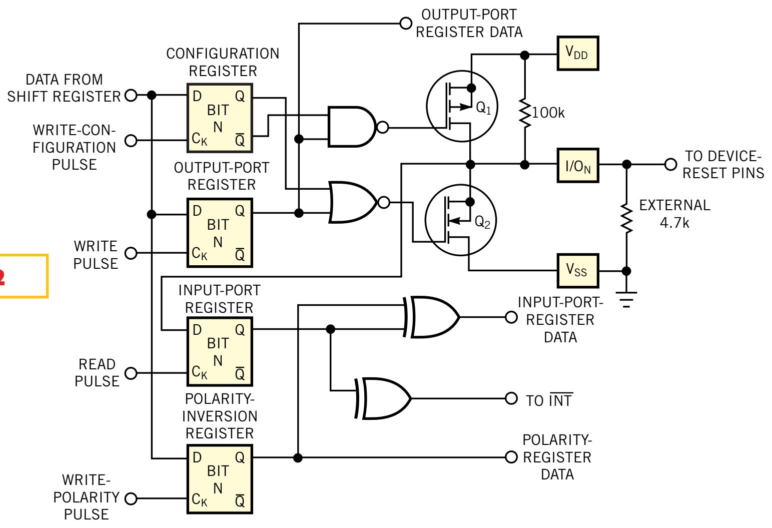
ST: Start bit generated by the master;
40: Write to slave address (A0 to A2 are all low in this example);

Figure 1



This circuit allows the I²C or SMBus to control device resets in a system.

Figure 2



The details of the I/O structure of the circuit in Figure 1 show four internal registers.

03: Write the next byte to the configuration register;
F0: Set I/O0 through I/O3 as outputs;
SP: Stop bit generated by the master;
40: Write to slave address;
00: Set all I/O as outputs;
SP: Stop bit generated by the master.

If you want to control more than eight devices, you can use the 16-bit I/O-port PCA9555 IC. Using the A0 to A2 address pins and the PCA9555, you can control as many as 128 devices, using the I²C bus or the SMBus. You can find additional information about the I²C bus, including the bus specification, at www.semiconductors.philips.com/buses/i2c/support/. For information about the SMBus, you can go to www.smbus.org/specs/.

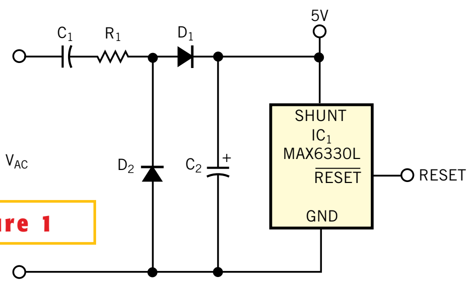
Is this the best Design Idea in this issue? Select at www.ednmag.com.

Supply uses ac to generate 5V, power-on reset

Greg Sutterlin, Maxim Integrated Products, Sunnyvale, CA

The need often arises for a low-cost logic supply for powering microcontrollers and related circuitry in "white-goods" products, such as industrial controllers and sensors. These applications usually include 24 or 115V ac or higher levels of ac voltage for conversion to 3.3 or 5V dc. The simplest approach to generating low-current logic-supply levels is to apply the rectified and filtered ac input to a high-input-voltage linear regulator. However, power dissipation in the regulator can be considerable, even for modest load currents. A standard shunt regulator also dissipates notable amounts of power in the limiting resistor. A switching regulator minimizes power dissipation, but that type may be impractical for cost-sensitive designs. As an alternative, consider the ac-coupled approach in **Figure 1**. The circuit suits applications in which 24V ac is available. With proper safety precautions, you can apply the circuit to double-insulated white goods and other products that require a logic supply for control or monitoring functions.

Figure 1



This SOT-23 IC with low-cost external components provides a combination of power-on reset and an efficient logic supply.

To transfer energy to the regulator with negligible power loss, the circuit uses a coupling capacitor in conjunction with an IC containing a shunt regulator and power-on-reset circuitry. Available with 50-mA maximum output-current capability in 3, 3,3, and 5V shunt-voltage versions, IC_1 also includes a power-on-reset function. Because IC_1 is an active shunt versus a passive zener diode, you must rectify the ac voltage before applying it. Typically, a capacitor follows the rectifier to hold the charge during off cycles. The design in **Figure 1** uses a simple half-wave rectifier to save cost. C_1 is the transfer capacitor, and C_2 stores energy. D_1 acts as a half-wave rectifier, and D_2 discharges the transfer capacitor during negative cycles. R_1 limits surge current during the discharge of C_1 and, if applicable, during high-voltage transient testing.

Several simplifications help to approximate the available output current. Assume 0V forward drops in the diodes and 0V shunt voltage in the IC regulator. With, for example, a 60-Hz sinusoidal input of 24V rms amplitude (V_{PEAK}=33.94V), you calculate as follows: Peak current in C_1 is $I_{PEAK}(C_1)=C_1(dV_S/dt)=C_1[V_{PEAK}(d\sin(\omega t)/dt)]=C_1[\omega V_{PEAK}(\cos\omega t))]=C_1\omega V_{PEAK}$.

Thus, with C_1=3 μF, ω=377.1 radians/sec, and V_{PEAK}=33.9V, I_{PEAK}=38.4 mA. The rms charge current (I_{RMS}) in C_1 is

$$I_{RMS}(C_1)=\sqrt{1/T\int_0^{T/2}\left[(C_1(\omega)V_{PEAK}\cos(\omega t))^2dt\right]};$$

$$I_{PEAK}(C_1)=C_1(V_{PEAK})d\sin(\omega t)/dt=C_1(\omega)V_{PEAK}.$$

Thus, for T=16.7 msec, I_{RMS}=19.1 mA. By adjusting the value of C_1, you can limit peak-current levels to the value of maximum MAX6330 shunt current, 50 mA, and achieve an output of 20 mA or so. The voltage rating of C_1 must be able to withstand the maximum input voltage. Because peak currents are limited to I_{PEAK}, practically any small-signal diode can serve as the half-wave rectifier, D_1. D_2 discharges C_1 during the negative portion of the cycle. The current rating of D_2 depends on the value of V_{PEAK} and the selected value, 50Ω, of surge-limiting resistor R_1. The maximum reverse voltage on D_1 and D_2 is ($V_{SHUNT}+V_{DIODE}$). C_2 acts as a storage capacitor that maintains load current during the negative portion of the cycle. To calculate its value, use the following approximation based on the allowable level of ripple voltage (V_{RIPPLE}): $C_2=(I_{LOAD}\times T/2)/V_{RIPPLE}$. With V_{RIPPLE}= 150 mV, T/2=8.3 msec, and I_{LOAD}=10 mA, C_2=550 μF.

Is this the best Design Idea in this issue? Select at www.ednmag.com.

Routine automates pattern/sequence detection

K Venkatachalam, Murugesh Rajiah, and Vijayakrishnan Rousseau, Tata Elxsi Limited, KR Puram, Bangalore, India

Sequence detection is a common operation in many communication and security systems. Some good examples are HDLC-flag identification and signature analysis. As the complexity of the system increases, designing circuitry for sequence detection becomes tedious and laborious. Using the software tool in this Design Idea, you can generate HDL code in VHDL or Verilog formats for both Mealy and Moore machines for any sequence of arbitrary length. The tool additionally presents options for inferring the sequence-detector state machine in one of the popular encoding styles, such as one-hot, binary, and Gray. Let S0 to Sn−1 be the states of a Mealy machine for n-bit sequence detection. The key to any state-machine design is to find the next state transition, which is a function of the input and the current state. Any state Sm (0≤m≤n−1) indicates that the m bit of the n-bit sequence has been detected so far.

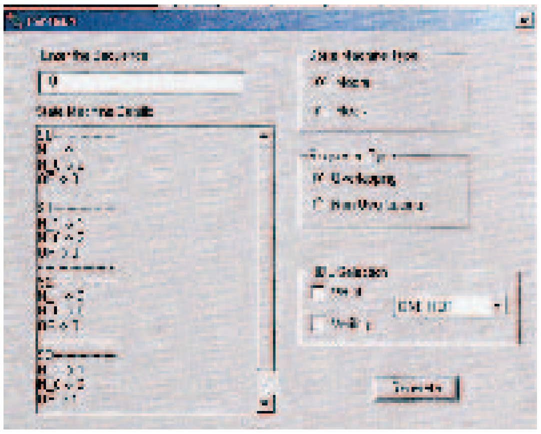


Figure 1 These are the options available for the sequence-detection method.

The state machine switches from Sm to Sm+1 on detecting an input that matches the next bit of the sequence. Otherwise, the state machine stays in the same state or switches one of the previous states. The state machine's status depends on the currently received bit and the previously received bits. If this pattern of bits matches the pattern of bits successfully detected in any of the previous states, then the state machine switches to that state. **Listing 1** is the pseudocode of the algorithm for determining the next state transition from any state Sm (where m can assume values of 0 to n−1). Let the n-bit sequence be represented as an array of bits called SEQ. The index of the array ranges from 0 to n−1. In the pseudocode, the state machine generally switches from Sm to Sm+1 on detecting an input that matches the next bit in the sequence. But this is not the case for the last state. For nonoverlapping sequence detection, the state machine switches from the last state to the initial state (S0) upon detecting the last bit of the sequence. On the other hand, for an overlapping sequence detection, the state machine either stays in the last state or switches to one of the previous states upon detecting the last bit of the sequence. The routine determines the state-machine behavior in the same way as it determines the next state transition for an input that doesn't match the next bit of the sequence. Refer to the "while" loop of the "if" and "else" blocks of the pseudocode.

For both overlapping and nonoverlapping sequence detection, the output switches high from the last state when the

More on two-transistor circuit

In a recently published Design Idea, Jim Hagerman proposes a two-transistor circuit that claims to replace the LTC4300 hot-swappable, two-wire LTC4300 bus buffer ("Two-transistor circuit replaces IC," *EDN*, Feb 7, 2002, pg 104). I feel compelled to point out that the LTC4300 offers numerous advantages over this circuit for both buffering and hot-swapping the I^{2}C bus.

The NPNs in Hagerman's circuit provide no capacitive buffering between the card and the backplane, which allows card capacitance to add directly to backplane capacitance. The LTC4300 isolates card capacitance from the backplane capacitance, making it possible to meet the 400-pF specification.

The LTC4300 further aids in meeting rise-time requirements by providing boost pull-up current circuits on all four SDA and SCL pins. This feature allows users to choose weaker pull-up resistors on the bus that reduce power consumption and increase noise margin while still meeting system rise-time requirements.

When hot-swapping the card, the Hagerman solution immediately connects the card bus to the backplane bus with no regard for the condition of either node. The LTC4300 monitors the backplane side for a Stop Bit or Bus Idle and the card side for logic-high states before connecting the buffers. This approach ensures that no data transaction is occurring during the instant of connection.

Although I do not doubt the attractiveness of Hagerman's idea, I felt it was important for your readers to appreciate the additional benefits of the LTC4300.

Todd Nelson
Product Marketing Manager
Mixed Signal Business Unit
Linear Technology Corp

method detects the last bit of the sequence. You could generate a Moore machine by following the same concepts. The only difference would be that the number of states is one more than that of a Mealy machine. Moreover, the output depends only on the states and not on the input, and it goes high only in the last state where the sequence has been successfully detected. This tool can no doubt help a designer to focus on high-level designs, rather than intricate design details. You can extend the idea to detecting double sequences, such as start-of-frame and end-of-frame sequences, and sequences having more than one bit as inputs. You can download **Listing 1** and an executable file that creates a GUI (**Figure 1**) for the sequence detection from the Web version of this Design Idea at www.ednmag.com.

Is this the best Design Idea in this issue? Select at www.ednmag.com.

LISTING 1—ROUTINE FOR DETECTING SEQUENCES

```
if (SEQ[m] == 1)
{
        Next state transition on input 1 = Sm+1.
        Let p = m.    //p is a variable.
        while (p! = 0)
        {
                Form a p-bit sequence comprising of the last p-1 bits received and 0.
                if(the p bit sequence obtained == the first p bits of the sequence)
                        break;
                p--;
        }
        Next state transition on input 0 = Sp.
}
else if(SEQ[m] == 0)
{
        Next state transition on input 0 = Sm+1.
        Let p = m.
        while (p! = 0)
        {
                Form a p-bit sequence comprising of the last p-1 bits received and 1.
                if(the p bit sequence obtained == the first p bits of the sequence)
                        break;
                p--;
        }
        Next state transition on input 1 = Sp.
}
```