design ideas

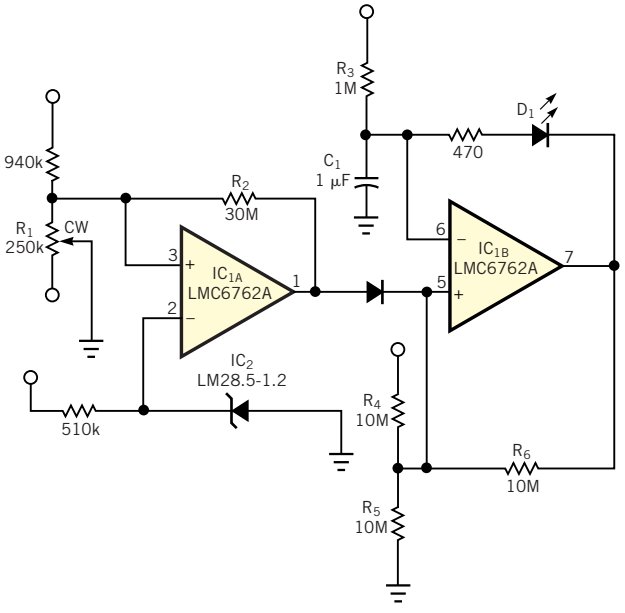
Edited by Bill Travis

Fleapower flasher draws less than 50 µA

Gary Butterfield, IEC Electronics Technology Center, Newark, NY

Some applications require a circuit to indicate that a battery's voltage has fallen below a certain value. However, if you don't frequently check the indicator, the low-battery indicator itself can easily discharge the battery. The circuit in **Figure 1** indicates when the battery voltage has dropped below a preset value. The circuit draws less than 50 µA, regardless of whether the indicator flashes. IC_{1A} operates as a simple comparator. IC_2, a low-current voltage reference, supplies the reference voltage to Pin 2 of comparator IC_{1A}. Resistor R_1 provides an adjustable trip point. A potentiometer setting of 124 kΩ yields a trip point of approximately 10.3V. When the power-supply voltage is above the trip point, IC_{1A}'s Pin 1 is high, forward-biasing diode D_1 and holding the flasher in the off state. R_2 provides approximately 300 mV of hysteresis for the detector. IC_{1B} provides the flashing of the LED. C_1 charges through R_3, and, when its voltage exceeds the voltage at IC_{1B}'s Pin 5, Pin 7 pulls low, discharging the capacitor through the LED. Resistors R_4 and R_5 provide a voltage reference, and resistor R_6 provides hysteresis of approximately one-third of the supply voltage.

Figure 1



This fleapower flasher indicates a low-battery condition with a flashing LED.

The circuit's current consumption is 45 µA at 10V, climbing to 48 µA at 12V. The state of the flasher does not affect current consumption, because the LED receives its power via the capacitor. You have to make a number of trade-offs to achieve this low current consumption:

For example, discharging the capacitor through the LED allows the circuit to reuse the capacitor's charge, instead of dumping it to ground. It also eliminates the current spikes that occur when driving the LED from the supply rail. However, the charge on the capacitor limits the resulting LED brightness, so you should use high-efficiency LEDs when possible.

Another trade-off is that the LED affects the minimum operating voltage because of the forward voltage drop of the diode. In the prototype, a red LED works down to a lower operating voltage of 4.3V. A yellow LED in the same circuit operates down to 6.4V. Additionally, the high resistances on the board, most notably the resistors attached to IC_{1B}'s Pin 5, require careful attention to board cleanliness. Small leakage currents can significantly affect circuit operation and current drain. You can reduce the values of these resistances to improve manufacturability at the expense of higher current consumption.

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

Publish your Design Idea in *EDN*. See the What's Up section at www.edn.com.

PC-based configurable filter uses no digital potentiometers

Saurav Gupta and Tejinder Singh, New Delhi, India

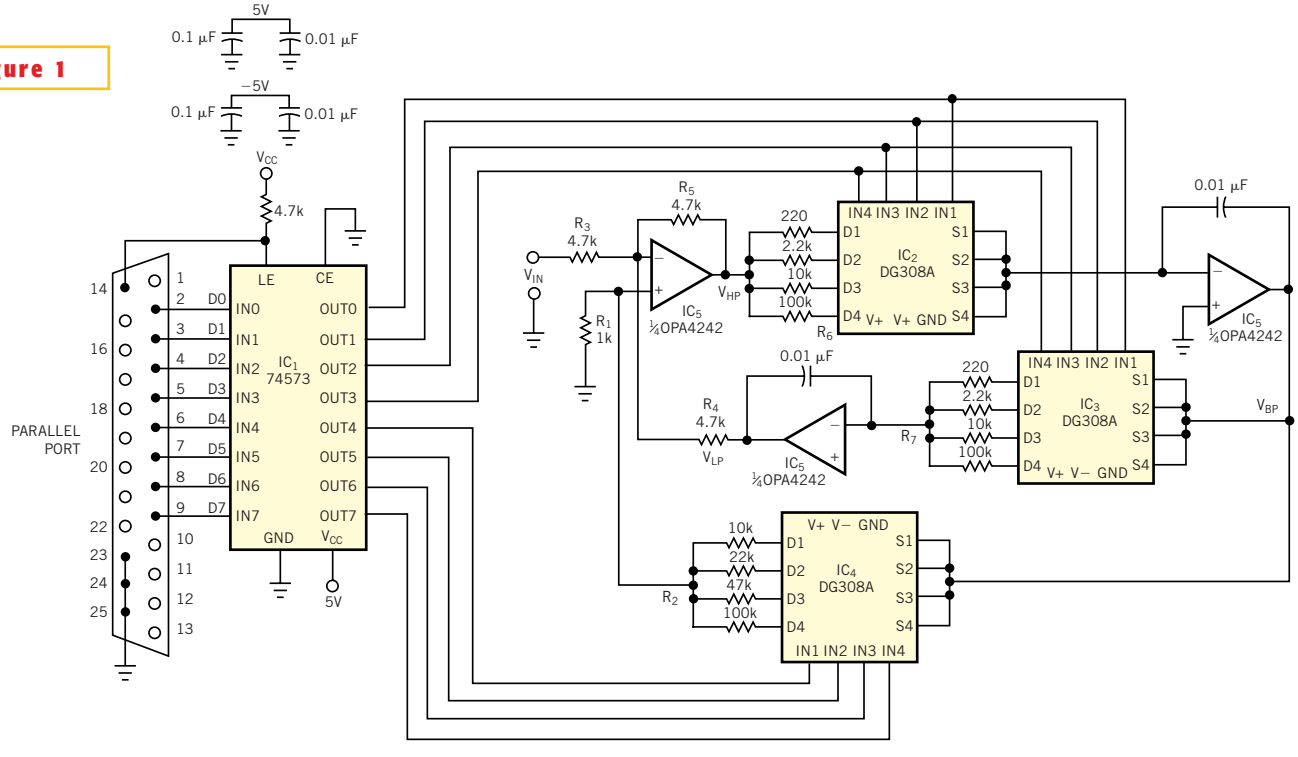
Modern instrumentation requires digital control signals. These signals come from a central microprocessor or, in modern context, the popular parallel or serial PC ports. In recent times, digital potentiometers have eliminated the hassles from this interface for the analog section. Designers can replace the resistors of the analog design with digital potentiometers, thus providing the necessary digital control. However, digital potentiometers suffer more severely from temperature-sensitive performance drifts than their manual counterparts, and they exhibit finite wiper-resistance effects. The design in **Figure 1** represents a multifunction, analog biquadrature design for automated mixed instrumentation. You can configure the design for both Q factor and center frequency via a PC's parallel port. The circuit requires no DACs or digital potentiometers. The circuit,

```
 Select Frequency(in Hz/sec)

        (1) 159          (6)  8110.0
        (2) 1591.5       (7)  30.68 K
        (3) 1750.7       (8)  32.20 K
        (4) 6366.0       (9)  37.00 K
        (5) 7957.0       (10) 38.70 K

Your Choice :6
Filter with Reconfigurable Characteristics

   Select Q (The Quality Factor)

        (1) 2.24         (9)  4.70
        (2) 2.50         (10)  5.36
        (3) 2.35         (11)  6.00
        (4) 2.65         (12)  6.38
        (5) 2.90         (13) 7.72
        (6) 3.12         (14) 11.01
        (7) 3.40         (15) 33.72
        (8) 3.72

Your Choice : 5
You Have Selected the Filter with
Frequency : 8110.00 Hz/sec
Quality Factor : 2.90
Bit Pattern : de (Hex),          1101 1110 (Binary)

Press
Enter : To Set the Switches
Esc   : To Exit

Chip Programmed Successfully!!
Press any key to exit
```

Figure 2

This user-friendly screen helps you configure the desired filter.

Figure 1



You can use a PC-configurable filter design to select both Q and center frequency.

based on a two-integrator configuration, provides simultaneous highpass, lowpass, and bandpass outputs.

By running simple code on the PC, you can choose from more than 150 programmable combinations of Q factor and center frequency (**Listing 1**). You can thus build a filter of desired parameters on the fly. The design uses quad analog switches DG308 from Maxim (www.maxim-ic.com) together with octal latches for programmability. A micropower precision op amp, OPA4242, from Burr-Brown (www.ti.com) makes up the analog-filter section. The software provides the data bits on ports pin 2 through 9, stored in the latch that controls the analog switches and, hence, selects the appropriate resistance combination to select the desired Q and f_0 values. The center frequency and Q values are: $f_0=[1/C_1C_2R_{P6}R_{P7}]^{1/2}$ and $Q=(1+R_{P2}/R_1)/3$, where R_{P2}, R_{P6}, and R_{P7} are PC-programmable resistances.

Data nibbles from port pins D2 through D5 provide ganged-switch settings for R_{P6} and R_{P7}, ensuring that they are always equal. Data nibbles correspon-

TABLE 1–DATA BITS FOR Q SELECTION

D9 (100k)	D8 (47k)	D7 (22k)	D6 (j10k)	Q
1	1	1	1	2.24
0	1	1	1	2.35
1	0	1	1	2.5
0	0	1	1	2.65
1	1	0	1	2.9
0	1	0	1	3.12
1	0	0	1	3.4
0	0	0	1	3.72
1	1	1	0	4.7
0	1	1	0	5.36
0	1	0	0	6
1	0	1	0	6.38
0	0	1	0	7.72
1	1	0	0	11.01
1	0	0	0	33.72

LISTING 1–C PROGRAM FOR PC-CONFIGURABLE FILTER

```
#include<dos.h>
#include<conio.h>
#include<stdio.h>
#include<process.h>
#define LPT 0x378        /* Data Port Address     */
#define CONT 0x37A       /* Control Port Address */
#define ESC 27
typedef unsigned char u08;
enum boolean {false,true};

/* Funtion Declarations */
void Chip_Program(u08);  /* To set the analog switches */
void showbits(u08);      /* To see bit pattern */
void DrLine(void);       /* Draw a Line */
void init(void);         /* Initializes the Parallel Port */

void main()
{
 int fch,ctr,qch;
 u08 pdat,tdat;
 float freq,qlty;
 init();
 do{
 clrscr();
 printf("RECONFIGURABLE UNIVERSAL FILTER  By Saurav Gupta & Tejinder Singh\n");
 DrLine();
 printf("   Select Frequency(in Hz/sec)\n");
 DrLine();
 printf("\t(1) 159 \t(6)  8110.0\n");
 printf("\t(2) 1591.5\t(7)  30.68 K\n");
 printf("\t(3) 1750.7\t(8)  32.20 K\n");
 printf("\t(4) 6366.0\t(9)  37.00 K\n");
 printf("\t(5) 7957.0\t(10) 38.70 K\n");
 DrLine();
 printf("Your Choice :");
 scanf("%d",&fch);
 }while(fch<1 || fch>10); /* Loop until a valid choice is entered*/

  switch(fch)
 {
  case 1 : freq=159; pdat=0x08; break;
  case 2 : freq=1591.5; pdat=0x04; break;
  case 3 : freq=1750.7; pdat=0x0C; break;
  case 4 : freq=6366; pdat=0x02; break;
  case 5 : freq=7957; pdat=0x06; break;
  case 6 : freq=8110; pdat=0x0E; break;
  case 7 : freq=30680; pdat=0x01; break;
  case 8 : freq=32200; pdat=0x05; break;
  case 9 : freq=37000; pdat=0x03; break;
  case 10: freq=38000; pdat=0x0F; break;
 }


 do{
 clrscr();
 printf("Filter with Reconfigurable Characteristics\n");
                                  C_code.txt
 DrLine();
 printf("   Select Q (The Quality Factor)\n");
 DrLine();
 printf("\t(1) 2.24\t(9)  4.70\n");
 printf("\t(2) 2.50\t(10)  5.36\n");
 printf("\t(3) 2.35\t(11)  6.00\n");
 printf("\t(4) 2.65\t(12)  6.38\n");
 printf("\t(5) 2.90\t(13) 7.72\n");
 printf("\t(6) 3.12\t(14) 11.01\n");
 printf("\t(7) 3.40\t(15) 33.72\n");
 printf("\t(8) 3.72\n");
 DrLine();
 printf("Your Choice :");
 scanf("%d",&qch);

 }while(qch<1 || qch>15);
  switch(qch)
 {
  case 1 : qlty=2.24; tdat=0xF0; break;
  case 2 : qlty=2.50; tdat=0xB0; break;
  case 3 : qlty=2.35; tdat=0x70; break;
  case 4 : qlty=2.65; tdat=0x30; break;
  case 5 : qlty=2.90; tdat=0xD0; break;
  case 6 : qlty=3.12; tdat=0x50; break;
  case 7 : qlty=3.40; tdat=0x90; break;
  case 8 : qlty=3.72; tdat=0x10; break;
  case 9 : qlty=4.70; tdat=0xE0; break;
  case 10: qlty=5.36; tdat=0x60; break;
  case 11: qlty=6.00; tdat=0x40; break;
  case 12: qlty=6.38; tdat=0xA0; break;
  case 13: qlty=7.72; tdat=0x20; break;
  case 14: qlty=11.01; tdat=0xC0; break;
  case 15: qlty=33.72; tdat=0x80; break;
 }
 pdat=pdat|tdat;
 printf("\nYou Have Selected the Filter with\n");
 printf("Frequency : %6.2f Hz/sec\nQuality Factor : %3.2f\n",freq,qlty);
 printf("Bit Pattern : ");
 showbits(pdat);
 printf("\n");
 DrLine();
 printf("\nPress \nEnter : To Set the Switches\nESC   : To Exit");
 if(getch()==ESC)
 exit(0);
 Chip_Program(pdat);
 printf("\n");
 DrLine();
 printf("Chip Programmed Successfully!!\nPress any key to exit");
 getch();
}

void DrLine(void)
{
 int ctr;
 for(ctr=0;ctr<50;ctr++)
  printf("%c",205);
  printf("\n");
}

void init(void)
{
 outport(CONT,0xFF);
 delay(100);
                                  C_code.txt
 outport(LPT,0x00);
 delay(100);
}

void Chip_Program(u08 pdat)
{
 outport(CONT,0x00);
 delay(200);
 outport(LPT,pdat);
 delay(200);
 outport(CONT,0xFF);
 delay(200);
}

void showbits(u08 num)
{
 int i,k,mask;
 printf("%x (Hex),\t\t",num);
 for(i=7;i>=0;i--)
 {
  if(i==3 || i==7 || i==11)
  printf(" ");
  mask=0x0001<<i;
  k=num & mask;
  k==0 ? printf("0"):printf("1");
 }
 printf(" (Binary)");

}
```

ding to pins D6 through D9 control the value of R_{P2}; hence, you use them for Q-value selection. For the given resistance values, **Table 1** shows the data bits for different values of Q (ranging from 2.24 to 33.72). **Table 2** shows the data bits for various center frequencies (159 Hz to 38.70 kHz). **Figure 2** shows the software front-end screen to make the selection. The design uses an 8.11-kHz filter with Q-factor 2.9 for demonstration and hardware validation. **Listing 1** gives the necessary back-end C code. The switches employed have a finite on-resistance, R_{DS}, of approximately 150Ω, which the design takes into account. For higher precision, you can use better switches having on-resistances of approximately 35Ω. Note that more switches provide a wider range from which to select. You can choose the resistances to suit the application's bandwidth range. You can download the filter software from the Web version of this Design Idea at www.edn.com.

TABLE 2–DATA BITS FOR CENTER-FREQUENCY SELECTION

D5	D4	D3	D2	f0 (kHz)
1	0	0	0	0.159
0	1	0	0	1.519
1	1	0	0	1.75
0	0	1	0	6.37
0	1	1	0	7.96
1	1	1	0	8.11
0	0	0	1	30.68
0	1	0	1	32.2
0	0	1	1	37
1	1	1	1	38.7

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

Frequency source feeds entire lab

Mitchell Lee, Linear Technology Corp, Milpitas, CA

Plumbing a laboratory with a standard frequency makes a lot of sense if the lab uses multiple frequency counters, spectrum analyzers, and other frequency-dependent test equipment. Rather than spending time keeping all of the instruments' oscillators in calibration or buying expensive, high-precision oscillators, you can use the circuit in **Figure 1** to distribute a single calibrated frequency source to the external-reference input of each instrument. The circuit represents a simple, 10-MHz source and distribution amplifier. The output comes not from the emitter or collector of the Colpitts-oscillator transistor, Q_1, but rather from the current flowing in the 10-MHz crystal. The common-base stage, Q_2, converts this current into a voltage and establishes the correct dc level for the output amplifier, IC_1. This IC contains four gain-of-two buffers with 110-MHz, 3-dB bandwidth and can drive double-terminated 50 or 75Ω loads.

As **Figure 1** shows, the outputs use 75Ω impedance levels to take advantage of inexpensive F-type connector hardware and low-cost video coaxial cable. IC_1 also provides good isolation between its outputs, so that changes in loading on one output do not affect the other outputs. The circuit delivers more than 6 dBm to each termination. If high accuracy and low drift are critical needs, you can substitute Hewlett-Packard's (www.hp.com) HP10811A component oscillator for the Colpitts oscillator. Connect the HP10811A's output through a 510Ω resistor and a 10-nF coupling capacitor, directly to the emitter of Q_2. If you need more than four outputs, you can duplicate the IC_1 stage as many times as necessary.

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

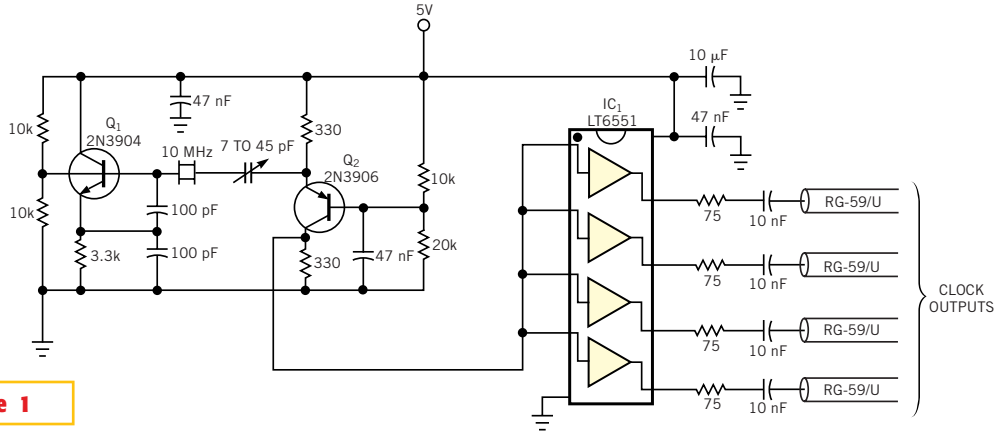


Figure 1

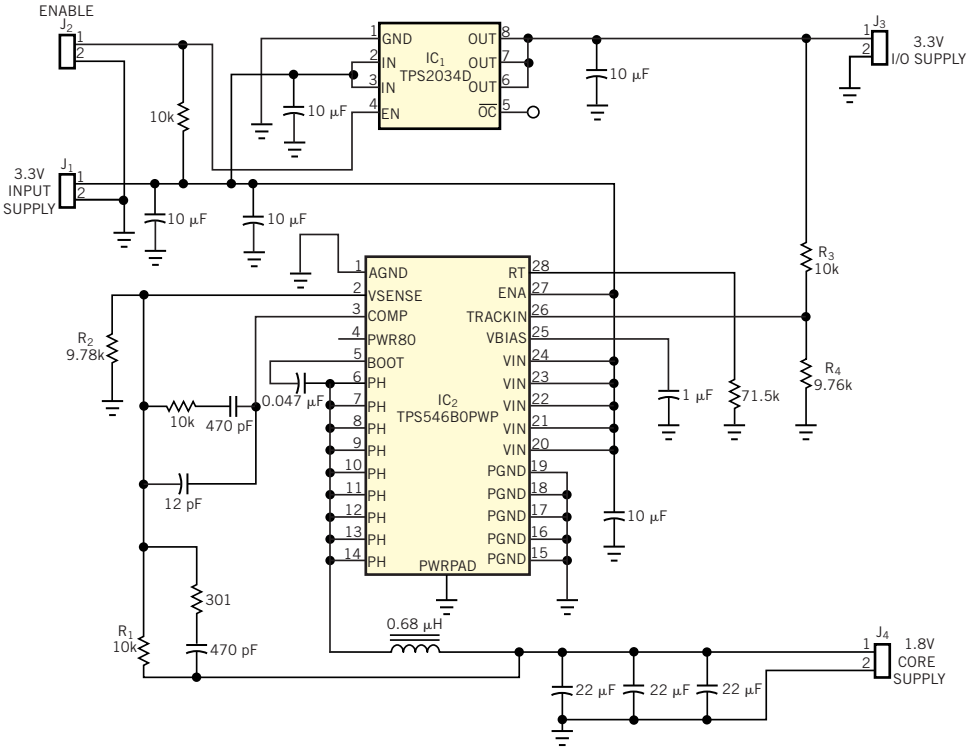
A laboratorywide distribution system is an alternative to multiple frequency sources.

Circuit sequences supplies for FPGAs

David Daniels, Texas Instruments, Dallas, TX

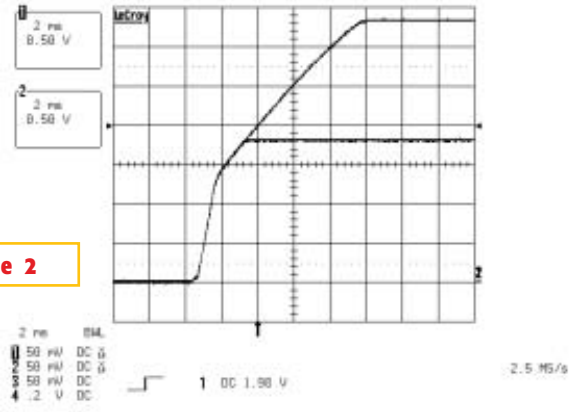
System designers must consider the timing and voltage differences between core and I/O power supplies (in other words, power-supply sequencing) during power-up and power-down. The possibility of a latch-up failure or excessive current draw exists when power-supply sequencing does not occur properly. The trigger for latch-up may occur if power supplies apply different potentials to the core and the I/O interfaces. FPGAs and other components with different sequencing requirements further complicate the power-system design. To eliminate the sequencing problem, you should min-

Figure 1



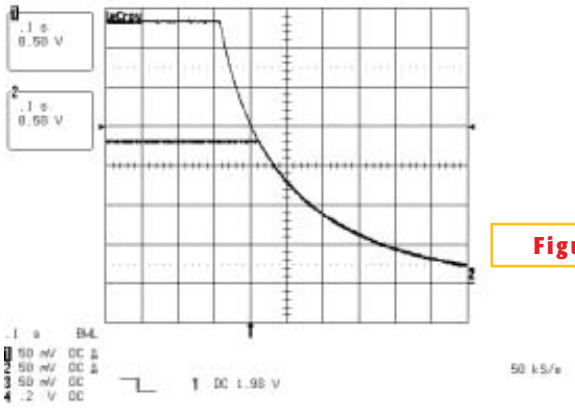
This power-supply-sequencing circuit eliminates latch-up problems and reduces FPGA start-up transient currents.

Figure 2



When the I/O-supply voltage decays, the core voltage gracefully follows suit.

Figure 3



As the I/O-supply voltage rises smoothly toward 3.3V, the core voltage clamps cleanly at 1.8V.

imize the voltage difference between the core and the I/O supplies during power-up and -down. The power supply in **Figure 1** regulates the 3.3V input voltage to the 1.8V core voltage and tracks the 3.3V I/O during power-up and -down to minimize the voltage differences between the supply rails.

The circuit in **Figure 1** comprises IC_1 and IC_2, a TPS2034 power switch and a TPS54680 step-down switching regulator, respectively. Component IC_1 is a high-side power switch that generates a slow ramp that IC_2 tracks during start-up. The ramp time of 6 msec minimizes the inrush currents to the bulk capacitors on the power-switch and supply outputs. The slow ramp minimizes the transient-current draw of the FPGA. The power switch ensures that the I/O voltage is not applied to the load before IC_2 has enough bias voltage to operate and generate the core voltage. Assuming that the input supply voltage is at 3.3V on J_1, floating the J_2 connector enables component IC_1. The I/O supply voltage, J_3, slowly rises until it reaches 3.3V. As the I/O voltage rises, the core voltage supply, J_4, rises accordingly until the voltage reaches 1.8V (**Figure 2**). The TPS54680 device incorporates an analog multiplexer on the TRACKIN pin to implement the tracking function.

During power-up and -down, when the voltage on the TRACKIN pin is lower than the internal reference of 0.891V, the voltage on the TRACKIN pin connects to the noninverting node of the error amplifier. When the TRACKIN pin is below 0.891V, the pin effectively functions as the switching regulator's reference. The resistor divider of R_3 and R_4 on the TRACKIN pin must equal the resistor divider of R_1 and R_2 in the feedback compensation to track with minimal voltage difference during power-up and -down. The TPS2034 has an on-resistance of 37 mΩ and can supply as much as 2A output current. The TPS54680 is a synchronous buck regulator that contains two 30-mΩ MOSFETs. Because the TPS54680 can source and sink as much as 6A load current at efficiencies greater than 90%, the output can track another power-supply rail during power-down. When the IC_1 device becomes disabled by shorting J_2 to ground, the I/O supply voltage decays, and the core supply voltage follows once the I/O voltage falls below the core voltage (**Figure 3**). Typically, Schottky diodes connect to the output of a dual power supply to clamp the voltage difference between the core and the I/O supplies during power-down, but most applications do not require the diodes with the power-supply circuit in **Figure 1**. Using this power-supply design reduces component count and increases reliability by eliminating the potential for latch-up and reducing FPGA start-up transient currents.

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