Circle 521

Phase-Locked Digital Synthesizer

SAMUEL KEREM
Patton Electronics, 7622 Rickenbacker Dr.,
Gaithersburg, MD 20879; (301) 975-1000; e-mail: samuelkerem@juno.com.

The circuitry shown in Figure 1 demonstrates the technical features from both phase-locked-loop (PLL) and direct-digital-synthesis (DDS) designs. The design represents a phase-locked oscillator with practically zero lock-in and recovery time, and jitter less than 4 ns. The device is significantly more tolerant to the absence of synchronous pulses, and it stays in phase much longer than PLL-based devices. There's no loop in the design, and the absence of a feedback makes the design quite simple.

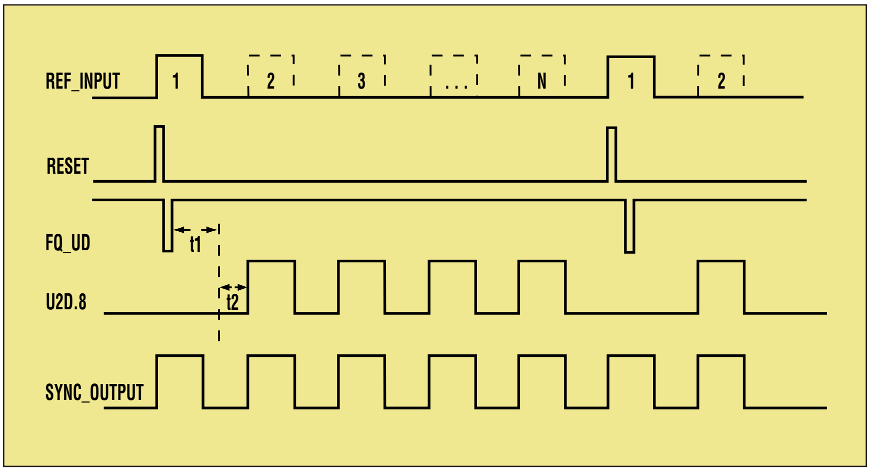
Designs requiring this performance include applications such as clock recovery and synchronous frequency multiplication. This circuit also can also be used whenever any two frequencies need to be synchronized at a particular moment in time.

The maximum input (reference) frequency range is 5 MHz and can be doubled, tripled, and so on with a simple modification. The output frequency can reach 50 MHz. There's one condition, though—the frequency of the input (reference) signal should be known in advance. This situation is common in telecommunications, when synchronization of two remotely located clocks (transmitter and receiver) is required, and a number of sync-pulses are absent in the transmitted signal. Such is the case in Figure 2, in which the reference pulses following reference pulse #1 are absent.

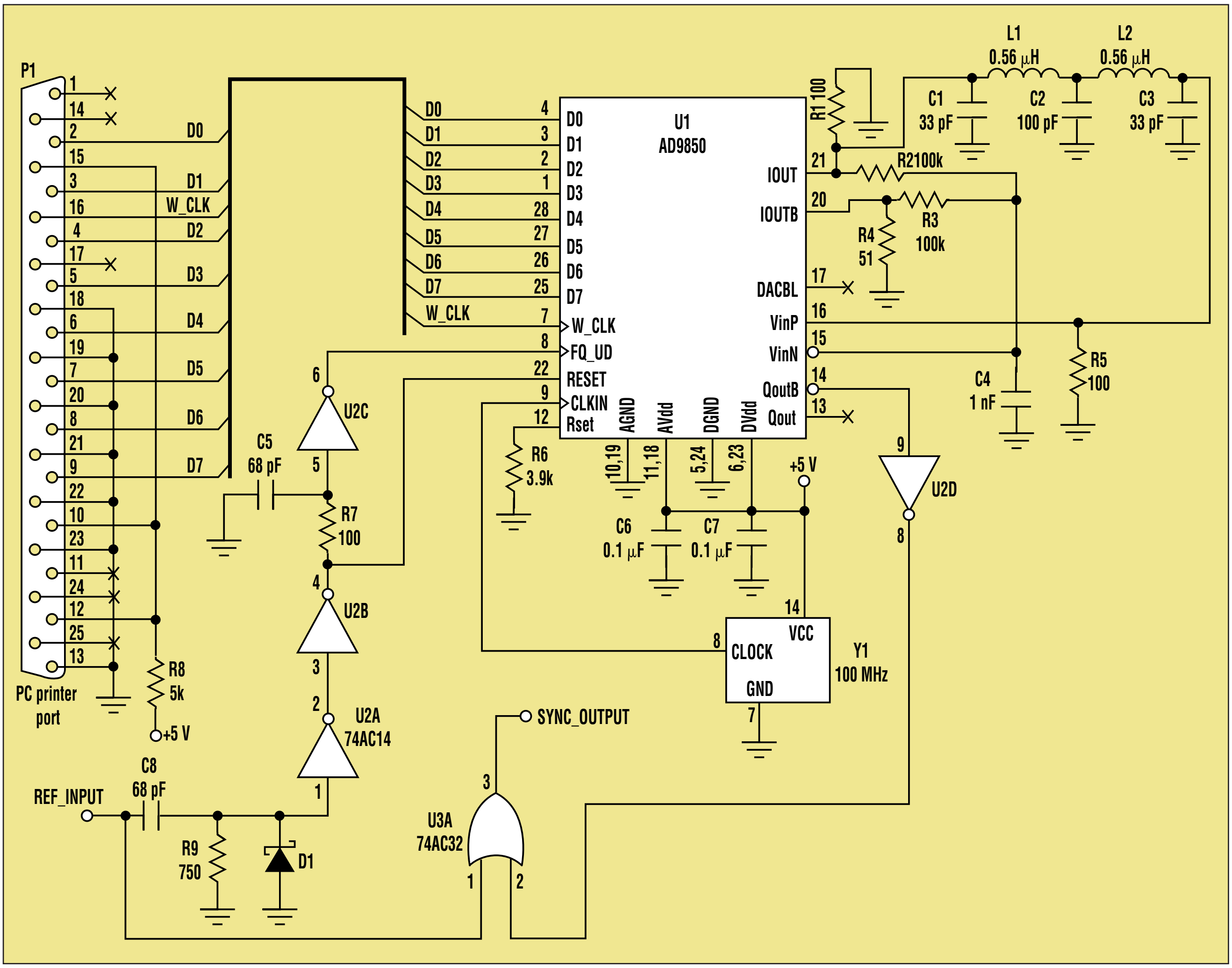
At the heart of the circuit is Analog Device's AD9850 DDS chip. The chip produces a digitally synthesized sinewave using a frequency value programmed from a master device (in our case, the PC printer port). The frequency value is latched through the data bus into the U1 input registers by W_CLK, coming from PC.

To produce the output frequency (U1.21), the chip must be clocked by the FQ_UD signal, usually supplied by the same master device. In our case, the FQ_UD comes from the reference signal. The reset signal, formed from the same reference signal (C8, R9, U2A, U2B), will bring all of the AD9850's internal registers into the predefined state, but will not affect the data stored. After a short delay following the reset pulse (R7, C5, U2C), the FQ_UD signal will be created and a digitized sine wave is produced at the output (U1.21). The sine wave is applied through a low-pass filter (C1, C2, C3, L1, L2) to the internal comparator input, resulting in a squarewave output (U1.14) with the programmed frequency.

The time interval t1 *(Fig .2, again)*, between the rising edge of the reference signal and the moment when updated data appears at the U1 output is predetermined and remains constant. For different frequencies, the initial phase shift t2 is program-controlled (0xd0 in our case) and will place pulse #2



2. Missing reference pulses are recovered and resynchronized with the reception of each detected reference pulse.



1. Shown here is the simplified schematic of a phase-locked oscillator with practically zero lock-in and recovery time. Jitter is less than 4 ns.

from U1 after the period equal to the reference signal period.

U3 will logically add reference pulse #1 and the following DDS pulses. Therefore, the first pulse and every subsequent reference pulse will synchronize the pulse sequence coming from DDS. These pulses are quartz-generated and their frequency variation will be significantly less than the VCO. The squarewave signal is produced from the spectrally pure DDS sinewave, with negligible phase noise. There is no feedback signal, which usually oscillates, even when the PLL is in the lock-in state. This also degrades the phase noise performance.

The supplied source code sample *(see the listing)* is for a 1.544-MHz frequency case.

```
/* Phase Locked Synthesizer.  Samuel Kerem
This program sends data to the AD9850.
Frequency = 1.544000 MHz +/- lHz
Phase = 146.25° */

#include <dos.h>

const PRN_DATA = 0x378;
const PRN_CNTR = 0x37A;
const W_CLK_1 = 0x04;
const W_CLK_0 = 0x00;

main()
{
     // 2^32(9850 resolution)/100*10^6(Yl clock)
     const float CLOCK=42.9497;
    // frequency and phase values
    unsigned long fr = 1544000;
    int ph = 0xd0;
    void syn(unsigned long fr, int ph);
    fr *=CLOCK;
    syn(fr, ph);
    return(0);
}

void syn( unsigned long freq, int phase )
{
    int byte_1, byte_2, byte_3, byte_4;
byte_l = freq/0xffffff;
    byte_2 = (freq%0xffffff)/0xffff;
    byte_3 = (freq%0xffff)/0xff;
    byte_4 = (freq%0xff);

   //control word + phase value
   outp(PRN_CNTR, W_CLK_0);
   outp(PRN_DATA, phase);
   outp(PRN_CNTR, W_CLK_1);
   outp(PRN_CNTR, W_CLK_0);
   // first byte loading
   outp(PRN_DATA, byte_1);
   outp(PRN_CNTR, W_CLK_1);
   outp(PRN_CNTR, W_CLK_0);
   // second byte
   outp(PRN_DATA, byte_2);
   outp(PRN_CNTR, W_CLK_1);
   outp(PRN_CNTR, W_CLK_0);
   // third byte
   outp(PRN_DATA, byte_3);
   outp(PRN_CNTR, W_CLK_1);
   outp(PRN_CNTR, W_CLK_0);
   // fourth byte
   outp(PRN_DATA, byte_4);
   outp(PRN_CNTR, W_CLK_1);
   outp(PRN_CNTR, W_CLK_0);
}
```