AN-822
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Synchronization of Multiple AD9779 TxDAC®s

by Steve Reine and Gina Colangelo

INTRODUCTION

The AD9779 TxDAC has the capability for DAC output sample rates of up to 1 GSPS. In some applications, such as those requiring beam steering, the user can synchronize multiple AD9779s. Therefore, the TxDAC timing specifications become critical when running the AD9779 near top speed.

This application note does not cover all of the details involved in the operation of the AD9779. The user should read the AD9779 data sheet for a complete understanding of the internal digital engine. This application note extends the use of SYNC_I to achieve the same REFCLK/DATACLK synchronization to multiple AD9779 devices.

In traditional interpolating TxDACs, two problems arise when the DAC is driven from the DAC output sample rate clock. First, it can be difficult to determine which of the DACCLK edges the input data is being latched on. Most DACs solve this problem by providing a DATACLK signal out that indicates the location of the input register latching edge. The second problem arises when the user tries to synchronize multiple TxDACs. This is the main topic of this application note. There is no guarantee that the DATACLK outputs from the multiple devices are synchronized; it is unlikely that they would be synchronized by themselves on power up. The AD9779 solves this problem by providing a second clock for data synchronization. This clock, called SYNC_I, is an input to the AD9779 and can be used to synchronize the latching of input data across multiple AD9779s.

This application note describes, in detail, methods for synchronizing the digital data inputs on multiple AD9779 devices. Phase alignment of the DAC output is guaranteed by design to be less than one DACCLK output cycle. However, due to unmatched delays on the outputs (at ambient temperature and at hot/cold temperatures), there can be a slight mismatch of phase alignment at multiple DAC outputs. This issue is not covered in this application note.

SYNCHRONIZATION OPTIONS

There are two options for synchronizing multiple AD9779 DACs. For the first option, one device is used as a master and the rest of the devices are used as slaves. For the second option, all devices operate as slaves. Both options have the same timing restrictions and there are no performance trade-offs. Block diagrams of the master/slave and slave modes are shown in Figure 1 and Figure 2.

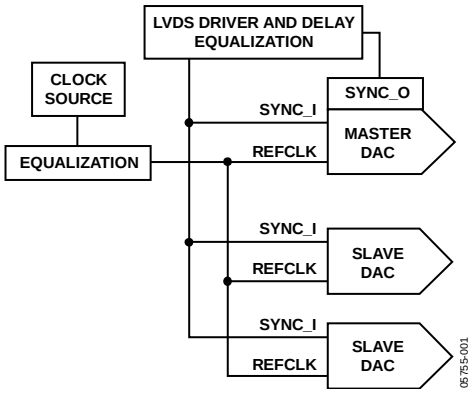

Figure 1. Master/Slave SYNC_I/O Distribution

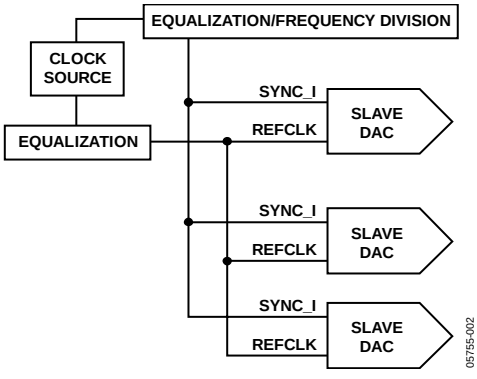

Figure 2. Slave SYNC_I Distribution

TABLE OF CONTENTS

REVISION HISTORY

9/06—Rev. 0 to Rev. A

Replaced Introduction Section 1
Replaced Synchronization Details Section 3
Replaced Timing Specifications Section 7

2/06—Revision 0: Initial Version

SYNCHRONIZATION DETAILS

In operation, a differential clock signal drives the AD9779 REFCLK inputs on all master and slave devices. The REFCLK input receiver is a high gain differential amplifier that requires a common-mode input level near 400 mV and a swing of at least 400 mV p-p on each differential input.

If a master device is chosen, a differential LVDS output signal can be enabled on the master device. This signal is referred to as SYNC_O+ and SYNC_O−. SYNC_O can be set to trigger on the rising or falling edge of DACCLK via Register 0x07, Bit 5. There is also a programmable delay to SYNC_O that can be set via Register 0x04, Bit 0 (MSB), and Register 0x05, Bits[7:4] (LSBs). SYNC_O is enabled by setting the sync driver enable bit (Register 0x07, Bit 6). The SYNC_O signal speed can be an integer divisor of the REFCLK speed, according to Register 0x04, Bits[3:1]. The possible timing scenarios of the REFCLK input and the SYNC_O signals on the master device are shown in Figure 3.

The SYNC_O driver and SYNC_I receivers are specified for LVDS levels (see the AD9779 data sheet).

The parallel digital input buses driving the CMOS digital data inputs on multiple AD9779 devices should be time equalized. If the multiple data buses are not equalized, the AD9779 has the programmability, via DATA_CLOCK_DELAY (Register 0x04, Bits[7:4]), to offset the latching instant of each AD9779 in increments of roughly 180 ps. The AD9779 has no capability to compensate for bit skews contained within a single data bus.

On all AD9779 devices, there is a setup-and-hold relationship between SYNC_I, the REFCLK input, and the CMOS digital input data. These timing relationships are given in the Timing Information section of the AD9779 data sheet.

The recommended application of SYNC_O and SYNC_O_DELAY is to use SYNC_O_DELAY to equalize the timing of SYNC_I and REFCLK to ensure their timing relationship is valid.

SYNC_I has its own programmable delay that can be set via Register 0x05, Bit 0 (MSB) and Register 0x06, Bits[7:4] (LSBs). SYNC_I_DELAY can be used in applications where the equalization is not perfect, or where the circuitry of Figure 2 is chosen. SYNC_I is enabled by setting the sync receiver enable bit (Register 0x07, Bit 7).

Table 1 shows the incremental SYNC_O_DELAY and SYNC_I_DELAY, which can be set via the SPI registers.

Table 1.

Temperature	SYNC_I/O_DELAY (Approximate Delay per Increment)
−40°C	72 ps
+25°C	78 ps
+85°C	83 ps

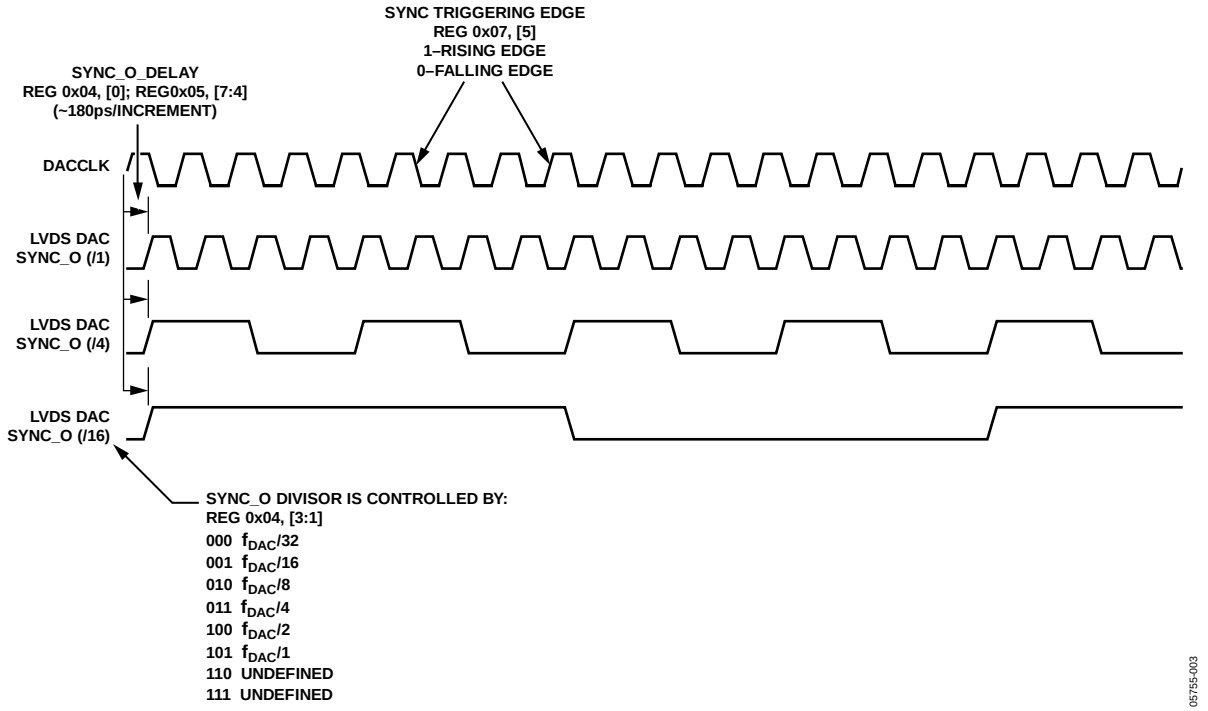

Figure 3. DACCLK SYNC_O Timing

Figure 4 shows a block diagram of the internal circuitry used to synchronize multiple AD9779s. The SYNC_I signal, after the programmable delay, is processed so that only one pulse of length DACCLK period remains for every rising edge of SYNC_I. Note that in this case, DACCLK represents the internal sample rate clock of the AD9779 DAC; this can be the same as REFCLK, depending on how the AD9779 is programmed. This single pulse of length DACCLK period drives the load signal on the 5-bit divider in Figure 4. The five signals output from the divider delay logic represent the possible DATACLK signals for all of the interpolation rates, including the possibility that zero stuffing is enabled. By programming the DACCLK offset register, Bit 1 through Bit 4 in Figure 4 can be delayed in increments of DACCLK cycles. The internal timing of the 5-bit divider, the effect of the load signal, and the DACCLK offset value are given in Figure 6.

The edge detector also drives the error detect circuitry shown in more detail in Figure 5. The programmable error detect circuitry can be used to measure the timing margin, generating an interrupt if a timing margin is exceeded.

The circuitry represented within the dotted line in Figure 4 is shown in more detail in Figure 5. Internally, the signals at the input to FF5 must meet setup-and-hold requirements with respect to each other. Invalid timing at the inputs to FF5 can cause loss of synchronization between REFCLK and the digital input data. Timing failure at this point is typically indicated by an increase in the DAC output noise floor. Extracting the timing requirements at the input to FF5 to the DACCLK and SYNC_I inputs, they become the setup-and-hold requirement for these two inputs.

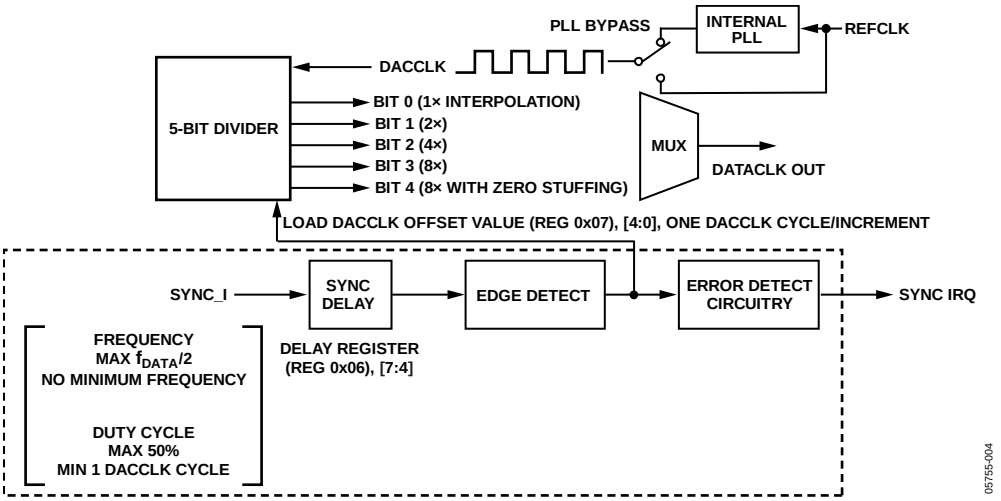

Figure 4. Block Diagram of AD9779 Multiple DAC Sync Circuitry

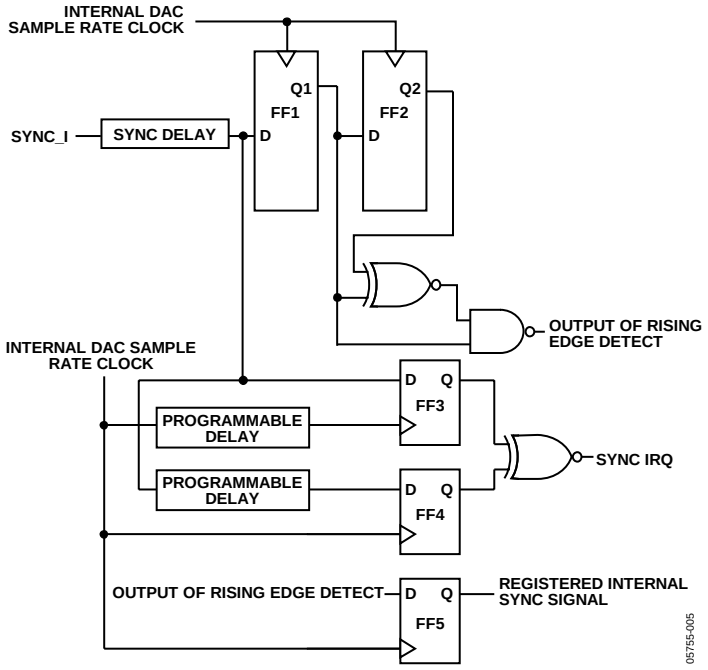

Figure 5. Detail of Programmable Timing Margin and Load Signal Generation

Varying the sync input delay can effectively shift the valid timing window of REFCLK/SYNC_I. In an actual application, with a given sync input delay, the result is a valid REFCLK/SYNC_I timing window with a given width. The timing margin values can be set to a value that the SYNC IRQ is set, if the timing margin is incremented by 1. Setting the timing margin to this value, in effect, sets the SYNC IRQ to 0 margin. The SYNC IRQ does not distinguish between timing errors caused by setup-and-hold violations. However, the SYNC IRQ, by design, is set when the programmable timing margin exceeds the smaller of the setup or hold margins. The user can increase the timing margin by increasing the value in Register 0x06, Bits[3:0]. With 0 margin, the SYNC IRQ is set if there is any drift toward the sensitive (setup or hold) specification.

In effect, the DACCLK is sampling the output of the edge detector. The output of the edge detector is a single pulse with a logic high width equal to one DACCLK cycle. For the load signal to be valid, the output of the edge detector must remain constant (high or low) during a given timing window around the rising edge of the internal DACCLK signal.

Assuming the programmable timing margin is set to 0 and the timing at the inputs to FF5 is valid, the Q outputs of FF3 and FF4 are the same and SYNC IRQ remains reset. Under these same conditions, if the timing at the inputs to FF5 is invalid, the outputs of FF3 and FF4 are different and the SYNC IRQ is set. If valid timing conditions exist at the inputs to FF5, the programmable timing margin has to be set to something greater than 0 to determine the timing margin.

In designing a system that uses the AD9779 in a master/slave synchronization configuration, the recommended course is to find the value of SYNC_O_DELAY (at which the programmable timing margin can be set to the largest value possible) before SYNC IRQ is set. This represents optimal timing, with the greatest timing margins. The user can then lower the value of the programmable timing margin. The amount by which the programmable timing margin is lowered represents the sensitivity of the SYNC IRQ to drift.

At the high DACCLK frequencies the AD9779 can receive, the valid timing window for DACCLK and SYNC_I can be a significant part of the DACCLK cycle. However, at slower DACCLK frequencies, it is likely that the range of the programmable timing margin does not allow the user to find an invalid timing window. In this situation, the user can be confident that under normal drift characteristics, the AD9779 does not drift over temperature into an invalid timing condition.

To ensure synchronization, SYNC_I has a maximum rate of DATACLK/2, where DATACLK is the input data rate (not DACCLK) to the AD9779. In Figure 6, two possible examples of applying SYNC_I are shown. In both examples, the AD9779 is in 4× interpolation mode, so that SYNC_I is running at a speed of DACCLK/8. Therefore, the 4× line is also the DATACLK out signal. In Figure 6 (a), the DACCLK offset value is set to 00000. On the rising edge of the internal SYNC_I delayed (a) signal, the rising edge of DACCLK causes all of the DATACLK output bits to reset to 0. Note that SYNC_I delayed has to occur in the window (Y) with respect to DACCLK in order for the 4× line to be set at time (X). If SYNC_I delayed (a) occurs slightly before

or after this window, the rising edge of the 4× line is advanced or delayed in time by one DACCLK cycle.

Note that with a DACCLK offset value of 00000, there is a delay of one DACCLK cycle between the application of SYNC_I delayed (a) and the rising edge of the 4× line.

In Figure 6 (b), the DACCLK offset value is set to 00010 at time (Z). Bits 8×, 4×, and 2× are thus set to 010 (which match the DACCLK offset bits). The next rising edge of the 4× line (DATACLK out) occurs three DACCLK cycles later.

If multiple DACs receive SYNC_I pulses within a certain time window and if they all have identical values of DACCLK offset, their DATACLK signals are synchronized. Therefore, data latching in multiple AD9779 devices occurs concurrently.

During initial synchronization, there can be discontinuity of the 2×, 4×, and 8× counter bits. That is, on initial application of the SYNC_I rising edge, the counter can be in such a state that synchronization causes it to change by multiple values. After the initial synchronization, however, as long as the speed of SYNC_I is kept to DATACLK/2 or slower, synchronization pulses occur only at times where the 2×, 4×, and 8× bits are reset to 0. (Although this seems redundant, it is true that after synchronization is achieved, the SYNC_I pulse actually does not have to be applied). The main purpose for periodic SYNC_I pulses after the initial pulse is in the rare event of the AD9779 devices becoming unsynchronized. This can occur due to a power supply glitch or possibly a runt clock pulse that triggers some, but not all, of the AD9779 devices in the system.

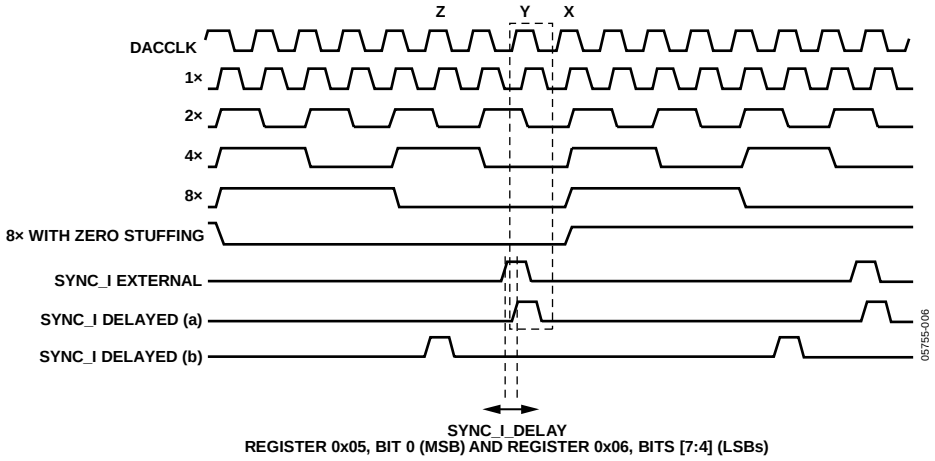

Figure 6. Internal Timing of SYNC_I, DACCLK, and DATACLK

TIMING SPECIFICATIONS

The first timing specification to note is the required relationship between SYNC_I and REFCLK, as shown in Figure 7. From the AD9779 data sheet, the required timing specifications are t_S = −0.2 ns and t_H = 1.0 ns.

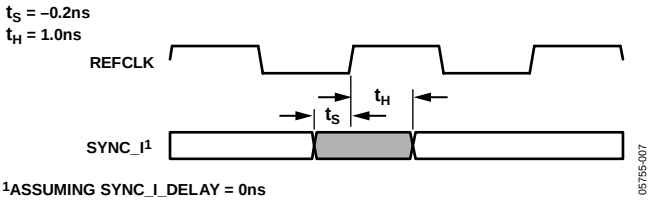

Figure 7. Timing Relationship of REFCLK and SYNC_I

If the DACCLK OFFSET value is programmed to a value other than 0, the DACCLK signal shown in Figure 7 effectively slides to the left by one DACCLK cycle. Likewise, if SYNC_I_DELAY is set to a value other than 0, with every increment of SYNC_I_DELAY, the SYNC_I signal in Figure 7 slides to the left by the incremental SYNC_I_DELAY given in the AD9779 data sheet.

The second important timing specification is the timing relationship between DATACLK out and the digital input data. This timing information is given in Figure 8. These values are valid for DATACLK_DELAY_ENABLE reset. If DATACLK_DELAY_ENABLE is set, then DATACLK is delayed (moved to the right in Figure 8) while the sampling point for the digital input data remains stationary. The keep out window of t_S and t_H therefore moves to the left with respect to DATACLK. The average delay per increment with DATACLK_DELAY_ENABLE set and for the incremental value of DATACLK_DELAY is given in the AD9779 data sheet.

See the AD9779 datasheet for setup-and-hold data vs. REFCLK, because this data can also be necessary in some applications.

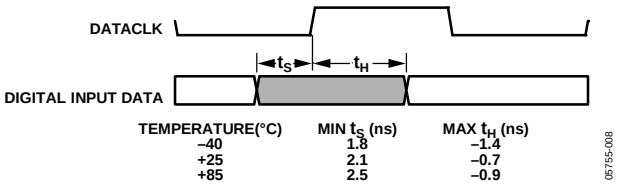

Figure 8. Setup-and-Hold, DATACLK to Input Data

NOTES

AN05755-0-9/06(A)