

多个AD9122 TxDAC+转换器的同步

作者: Yi Zhang

简介

AD9122是一款双通道、16位、高动态范围数模转换器(DAC),提供1230 MSPS采样速率。在某些应用中,例如需要波束导引的应用,用户必须同步系统中的多个DAC。AD9122具有多芯片同步功能,多个AD9122器件的DAC输出可以在一个DAC时钟周期内同步。AD9122有两种同步模式。本应用笔记将说明这两种模式的差异,以及何时和如何使用AD9122的多芯片同步功能。本应用笔记的内容同样适用于多个AD9125和AD9148 TxDAC+转换器的同步。

差异来源

DAC会给系统带来流水线延迟差异,进而导致不同DAC的输出不对齐,并且每次上电的偏斜不一致。在需要固定延迟的应用中,必须消除这种差异。在本应用笔记中,固定延迟是指DAC每次上电后从数字输入到模拟输出的时间延迟是固定的。它假设时钟条件相同,即数据时钟输入(DCI)、帧时钟、DAC时钟和同步时钟均相同。利用固定延迟可以实现多个DAC的同步。

在AD9122中,引起延迟差异的原因有两方面:FIFO和插值

滤波器。FIFO产生最多一个数据时钟周期的延迟差异。插值滤波器产生的最大差异为:

$$\left(1 - \frac{1}{\text{Interpolation Rate}}\right) \times \text{Data Clock Period}$$

因此,不开启AD9122同步功能时的最大延迟差异为:

$$\left(2 - \frac{1}{\text{Interpolation Rate}}\right) \times \text{Data Clock Period}$$

例如,假设数据时钟速率 $f_{\text{DATA}} = 300 \text{ MHz}$,系统采用4倍插值,则多个AD9122器件的DAC输出之间的最大延迟差异或最大偏斜为 $(2 - \frac{1}{4}) \times 3.3 \text{ ns} = 5.8 \text{ ns}$ 。

根据上述计算,设计时的第一个问题是DAC是否需要同步。为使多个DAC同步,需要进行额外的设计工作并且开启AD9122的同步状态机,从而增加设计的复杂度(详见“同步的系统设计考虑”部分)。强烈建议用户首先定义同步要求并指定时序预算,然后决定是否要实现同步。如果最大DAC延迟差异在预算范围内,则不需要实现同步。否则,需要开启同步状态机以减小延迟差异。

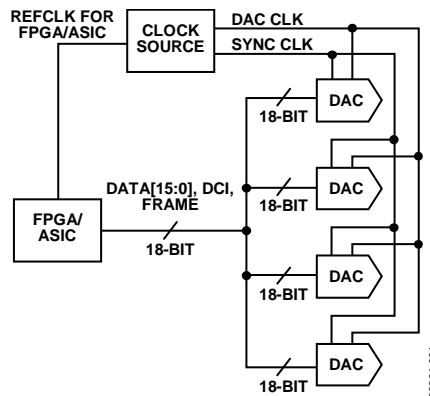


图1. 同步多个AD9122 TxDAC+转换器的框图

目录

简介.....	1	同步方案.....	4
修订历史.....	2	开启同步时的数据速率模式和FIFO速率模式.....	6
差异来源.....	1	同步的系统设计考虑.....	8
FIFO工作原理.....	3	同步设计示例.....	9
数据速率模式与FIFO速率模式.....	3	一次性同步实现.....	9
FIFO复位方法.....	3	总结.....	9
关闭同步功能应用的推荐工作模式.....	4		

修订历史

2010年9月—修订版0：初始版

FIFO工作原理

AD9122中的FIFO是一个多数据槽缓冲器，有助于将DCI时钟域的数据转交到DAC时钟域。在AD9122中，FIFO有8个双字(8×32位)槽。每个槽存储一对I和Q数据。针对FIFO输入和输出数据有两个指针：FIFO写指针和FIFO读指针。这两个指针循环移动，从槽0移动到槽7，然后回到槽0。在特定时间，它们指示输入数据进入哪个FIFO槽，输出数据从哪个FIFO槽出来。这种操作类似于水库。

水库储存一定量的水，同时有水流入和流出。为使水库的水位保持恒定，水流入的速度必须与水流出的速度相同。如果这两个速度略有波动，也不会有问题，因为水库能够消化二者之间的差异，最多可累积到其容量的一半。只要平均流入的水量和平均流出的水量相同，水库就永远不会干涸或溢流。

在AD9122中，当读指针和写指针指向同一FIFO槽时，即达到FIFO空置或溢出状态。这样会破坏FIFO中的数据传输，使得DAC输出有误。当FIFO正常工作时，输入数据以某一数据速率进入FIFO，输出数据以平均的相同速率离开FIFO。FIFO写指针由DCI时钟衍生的内部时钟控制。FIFO读指针由DAC时钟衍生(分频)的内部时钟控制。FIFO复位操作将这两个指针分开，二者之间的偏移由FIFO相位偏移(寄存器0x17)决定。其典型值为4，此时消解读取和写入速率波动的能力最强。

数据速率模式与FIFO速率模式

数据速率模式与FIFO速率模式的主要区别在于FIFO的复位方式不同。在数据速率模式下，当读指针到达槽0时，FIFO的写指针复位。发生触发事件时(有关触发事件的详情，参见“FIFO复位方法”部分)，写指针不复位，除非读指针到达槽0。

如果FIFO相位偏移设置为4，则在数据速率模式下发生FIFO复位时，写指针复位到槽4。因为是这种FIFO复位机制，所以只要FIFO不是频繁复位，则DCI与DAC时钟之间的相位关系无关紧要。无论系统上电时这两个时钟之间有何种相位关系，数据速率模式下的FIFO复位都会确保读指针与写指针始终相距大约4个(有可能是3个、4个或5个)FIFO槽。工作期间，DAC要求锁定DCI与DAC时钟之间的相位关系。如果相位不锁定，则读指针和写指针会在FIFO中随机跳跃，从而破坏传输数据。正常工作时，FIFO要求这两个指针按照从槽0到槽7再回到槽0的顺序移动。

在FIFO速率模式下，发生触发事件后，FIFO的写指针立即复位到槽4(当寄存器0x17设置为4时)。器件不关心写指

针复位时读指针位于何处。读指针可以位于槽0、槽1或任何可能的FIFO槽。因此，每次FIFO复位时，读指针与写指针之间的相位偏移是任意值。

如上所述，FIFO的最佳设置是读写指针之间的相位偏移等于FIFO深度的一半，对于AD9122而言即等于4。为实现这一最佳偏移，在FIFO速率模式下，如果FIFO复位后的偏移不是最佳值，则用户必须在寄存器0x17中手动增加偏移。例如，上电复位时寄存器0x17设置为4；FIFO复位后，FIFO温度计回读值(寄存器0x19)为0x03，意味着读写指针之间的偏移为2，距离最佳设置还差2个槽。为获得最佳设置，用户必须将寄存器0x17的值改为6，即原始值加上2。调整后，FIFO温度计回读值应为0xF。

FIFO复位方法

FIFO有两种复位方法(触发)。第一种方法是使用SPI命令。用户可以将0x02写入寄存器0x18实现FIFO复位，并通过回读该寄存器的值来验证复位是否完成。如果回读值为0x07，则表明复位已完成。FIFO复位的第二种方法是使用帧信号。帧信号的周期和正脉冲长度必须满足表1所列的要求，才能被视为一个FIFO复位触发事件。

表1. AD9122 FIFO复位的帧时钟要求

同步模式	帧时钟最大速率	正脉冲长度
FIFO速率模式	$f_{DATA}/8$	$\geq 1/f_{DATA}$
数据速率模式	$f_{DATA}/2$	$\geq 1/f_{DATA}$

表1中的要求适用于字模式。在字节模式下，脉冲长度应加倍；在半字节模式下，脉冲长度应为所列值的4倍。第一种FIFO复位方法称为软件复位，第二种方法称为硬件复位，因为第一种方法不需要外部硬件产生帧信号。在软件复位中，当器件接收到有效的SPI命令时，FIFO仅复位一次。在硬件复位中，每次器件接收到有效的帧脉冲时，FIFO都会复位。

关于硬件复位，需要注意的一点是：如果帧信号为周期信号，则在数据速率模式下，对DCI与DAC时钟之间的时序有一定的要求。当DAC时钟恰好位于DCI的建立和保持时间窗口内时，FIFO可以复位到预定相位偏移或预定偏移±1。这种不确定性会导致DAC丢失数据点，从而破坏DAC输出。用户需要确保DCI和DAC时钟满足AD9122数据手册中就这种情况规定的时序要求。注意，当FIFO周期性复位时，FIFO速率模式没有这种时序约束。它仅存在于FIFO周期性硬件复位的数据速率模式中。

关闭同步功能应用的推荐工作模式

对于不需要同步的应用，推荐工作模式是数据速率模式加一次性FIFO软件复位。这种模式是获得最佳FIFO设置的最简单方式，而且对DCI和DAC时钟没有时序约束。为了防止FIFO因为噪声而意外复位，帧输入必须设置为逻辑0，即在字接口模式下，FrameP设为0，FrameN设为1。图2所示为AD9122同步功能关闭时的推荐SPI命令序列。

同步方案

根据“差异来源”部分的计算，如果要求减小差异，AD9122的同步方案可以实现1个DAC时钟周期的同步精度。该方案要求用户产生两个外部时钟：帧时钟和同步时钟。这两个时钟必须满足一定的时序要求，具体取决于同步模式。这些时钟有助于消除“差异来源”部分所述的两类差异。

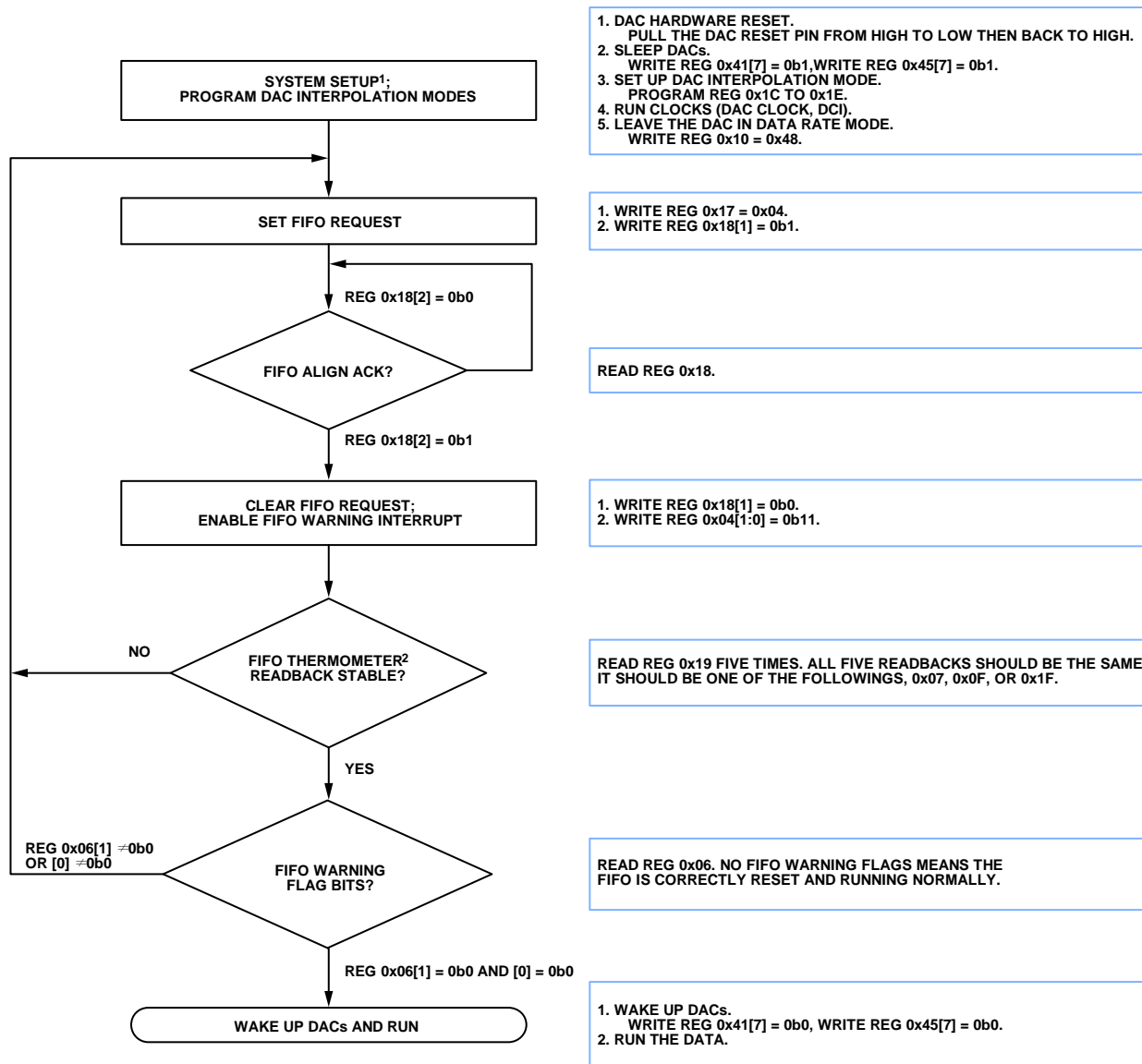
同步时钟用作系统中的参考时钟，使多个AD9122器件的内部时钟对齐，并且消除插值滤波器带来的差异。为此，同步时钟的运行速度必须等于或慢于AD9122中的最慢时

钟。在数据速率同步模式下，同步时钟速率(f_{SYNC})应等于数据速率(f_{DATA})或慢2n倍(n为整数)。在FIFO速率同步模式下， f_{SYNC} 应等于 $1/8 \times f_{\text{DATA}}$ 或慢2n倍(n为整数)。表2给出了同步时钟的最大速率。注意，由于同步时钟接收器具有交流耦合特性，同步时钟的最慢速率存在一个限制。应当正确选择交流耦合电容的值，确保信号摆幅达到数据手册中的要求。下面是 f_{SYNC} 最大速度的示例：假设 $f_{\text{DATA}} = 200\text{MHz}$ ，则FIFO速率模式下最大同步时钟速率为25 MHz，数据速率模式下为200 MHz。

表2. AD9122同步时钟最大速度

同步模式	同步时钟最大速度
FIFO速率模式	$f_{\text{DATA}}/8$
数据速率模式	f_{DATA}

帧时钟有助于消除FIFO引起的差异。FIFO由帧时钟复位，如“FIFO复位方法”部分所述。



¹IN WORD MODE, THE FRAME INPUT MUST BE DRIVEN AT LOGIC 0.

²THERMOMETER CODE IS BASE ONE CODE. FOR EXAMPLE, THERMOMETER CODE 00001111 IS 3 IN DECIMAL CODE.

图2. SPI命令流程图(同步关闭)

开启同步时的数据速率模式和FIFO速率模式

当同步开启时，数据速率模式和FIFO速率模式均能实现同样的同步精度。如“开启同步时的数据速率模式和FIFO速率模式”部分所述，这两种模式的区别在于FIFO的复位方式不同。图3显示了两种模式下的FIFO复位操作。同步开启时，FIFO的读取侧始终与同步时钟同步。

在数据速率模式下，FIFO复位后，FIFO的写入侧与读取侧同步。如果多个AD9122器件共享DCI、帧时钟、DAC时钟和同步时钟，则同一输入数据在同一时间写入和读出FIFO。因此，这些DAC同步，其输出对齐。

FIFO速率模式允许不同器件的FIFO写入侧在不同时间复位。这种模式下，FIFO的读取侧仍然与同步时钟同步，因此数据在同一时间从FIFO的同一槽中输出。FIFO的写入侧可以在不同时间复位，这意味着可以将同一输入数据写入不同器件中的不同槽。

因此，FIFO速率模式要求对FIFO相位偏移(寄存器0x17)进行手动调整。虽然多了这一步骤，但FIFO速率模式对时钟时序的要求相对不严格，因此推荐使用这种同步模式。图4所示为推荐的FIFO速率同步序列。

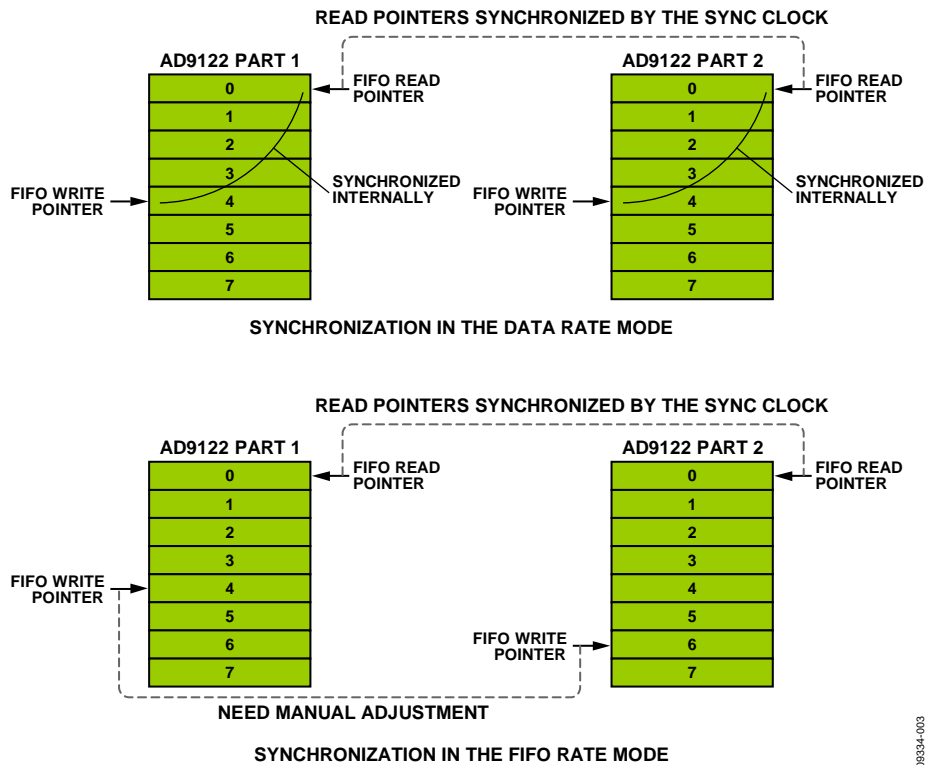
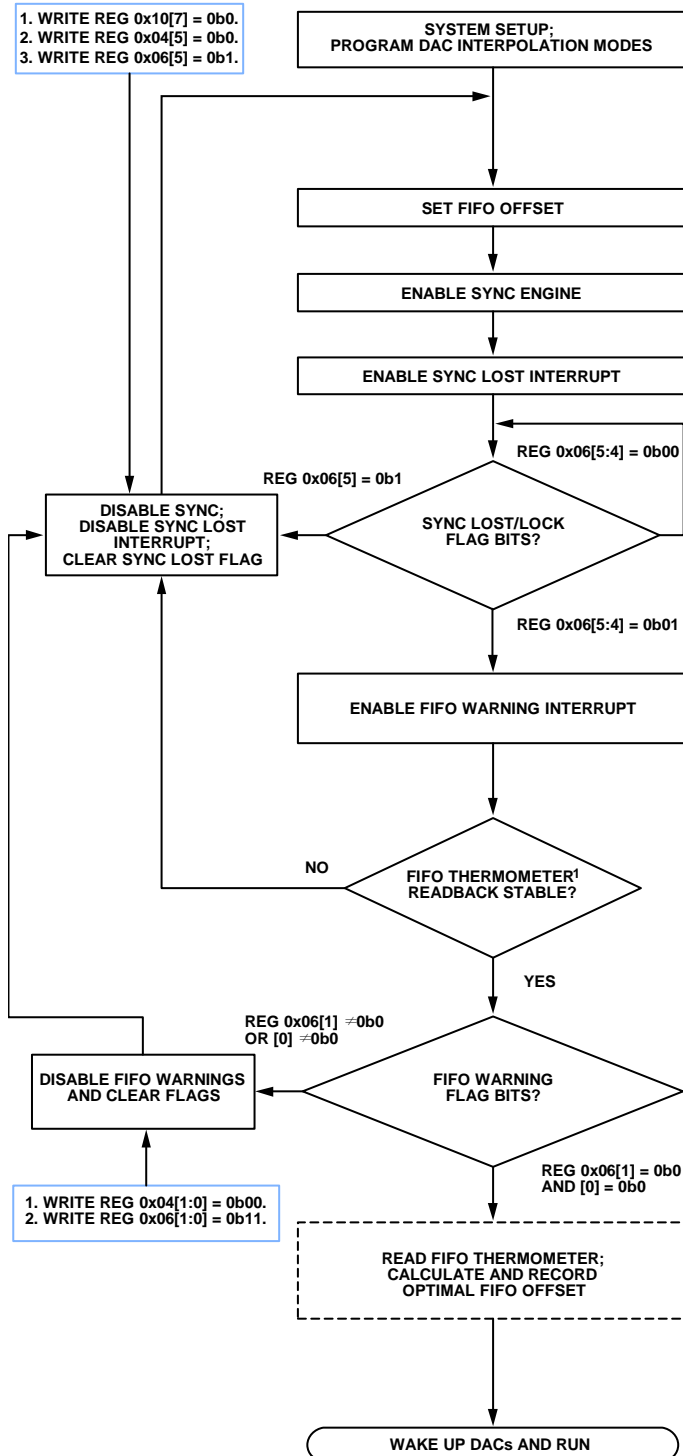


图3. FIFO复位操作：数据速率模式与FIFO速率模式

09334-003



- DAC HARDWARE RESET. PULL THE DAC RESET PIN FROM HIGH TO LOW THEN BACK TO HIGH.
- SLEEP DACs. WRITE REG 0x41[7] = 0b1, WRITE REG 0x45[7] = 0b1.
- SET UP DAC INTERPOLATION MODE. PROGRAM REG 0x1C TO 0x1E.
- RUN CLOCKS (DAC CLOCK, SYNC CLOCK, DCI, AND FRAME CLOCK).
- SET UP SYNC MODE. IN THIS CASE, IT IS FIFO RATE. WRITE REG 0x10[6] = 0b0.

WRITE REG 0x17 = OFFSET.
OFFSET IS DETERMINED BY THE STEP IN THE DOTTED BOX BELOW.
OFFSET = 0 WHEN THE DOTTED BOX IS IN THE FLOW.
OFFSET = (SAVED VALUE) WHEN THE DOTTED BOX IS NOT IN THE FLOW.

WRITE REG 0x10 = 0x88, IF RISING EDGE SYNC.
OR
= 0x80, IF FALLING EDGE SYNC.

WRITE REG 0x04[5] = 0b1.

READ REG 0x06[5:4] ((SYNC SIGNAL LOST; SYNC SIGNAL LOCKED)).
IF THE SYNC-DAC SETUP/HOLD TIMES ARE NOT MET, THE SYNC MAY NOT LOCK. CHANGE THE SYNC EDGE WHEN REENABLING THE SYNC NEXT ROUND.

WRITE REG 0x04[1:0] = 0b11.

READ REG 0x19 FIVE TIMES.
ALL FIVE READBACKS SHOULD BE THE SAME.
THE VALUE CAN BE ANY LEGAL THERMOMETER VALUE.

READ REG 0x06. IF NO FIFO WARNING FLAGS COME ON,
THE SYSTEM IS SYNCHRONIZED AND OPERATING NORMALLY.

THIS OPERATION IS A CALIBRATION STEP. IT NEEDS TO BE DONE ONLY ONCE IN THE SYSTEM DEVELOPMENT ON ANY OF THE DACs THAT NEED TO BE SYNCHRONIZED IN THE SYSTEM. IT SHOULD BE BYPASSED IN THE FLOW LATER ON. THE VALUE IS SAVED IN THE MEMORY FOR STEP 2 (SET FIFO OFFSET).

READ REG 0x19 AND CALCULATE ITS OFFSET FROM THE OPTIMAL VALUE ACCORDING TO THE EQUATIONS BELOW.
IF REG 0x19 READBACK (CONVERTED TO DECIMAL NUMBER) ≤ 4 ,
OFFSET = 4 - (REG 0x19 READBACK);
IF REG 0x19 READBACK > 4 ,
OFFSET = 12 - (REG 0x19 READBACK);

- WAKE UP DACs. WRITE REG 0x41[7] = 0b0, WRITE REG 0x45[7] = 0b0.
- RUN THE DATA.

¹THERMOMETER CODE IS BASE ONE CODE.
FOR EXAMPLE, THERMOMETER CODE
00001111 IS 3 IN DECIMAL CODE.

图4. SPI命令流程图(FIFO速率模式、同步开启)

同步的系统设计考虑

为了实现一个DAC时钟周期的同步精度，用户必须为AD9122提供两个时钟信号：帧时钟和同步时钟。根据用户选择的同步模式，这些时钟必须满足一定的建立和保持时序要求(见表3)。

在同步状态机中，同步时钟由DAC时钟采样，产生一个用于对齐内部时钟的参考点，因此同步时钟沿与DAC时钟沿之间存在一个建立和保持时序。AD9122允许用户选择在DAC时钟的上升沿或下降沿对同步时钟进行采样，从而更容易满足时序要求。

如果FIFO周期性复位，则在数据速率模式下，还有另一个建立和保持时序要求。在此模式中，DCI与DAC时钟之间的关系变得至关重要。这一时序约束以数据速率出现，即建立和保持时序窗口位于DCI上升沿附近。DAC时钟的上升沿或下降沿(取决于用户选择在哪一个边沿对同步时钟采样)必须根据这一时序要求进行放置。

为了实现目标同步精度并满足时序要求，必须做出一些系统性的决策：

- 是否有速度为数据速率1/8的同步时钟可用？
- 同步时钟由时钟芯片还是FPGA/ASIC产生？
- 系统涉及到的时钟(DAC时钟、同步时钟和DCI)是否锁相？
- 时钟走线的匹配程度如何？

对第一个问题的回答决定使用何种同步模式。如果有这样一个时钟可用，则推荐使用FIFO速率模式。该模式对DCI和DAC时钟没有时序约束，有助于简化系统设计。

此外，建议利用时钟芯片产生同步时钟，该时钟芯片最好

就是分配DAC时钟的芯片。该时钟芯片提供与DAC时钟锁相的低抖动同步时钟，从而很容易满足同步时钟与DAC时钟的时序要求。如果由于系统限制原因，同步时钟必须从FPGA或ASIC产生，则必须将其抖动降至最低。大抖动会导致同步逻辑不稳定，从而破坏DAC输出。

如果FPGA/ASIC与DAC时钟处于不同的时钟域，为了满足同步时钟和DAC时钟的时序要求，FPGA/ASIC和DAC时钟必须锁定系统中的同一参考时钟。在数据速率模式下，类似的锁相要求也适用于DCI和DAC时钟。通常而言，DCI由FPGA/ASIC产生，这要求FPGA/ASIC锁定DAC时钟域。

走线不匹配会影响同步精度以及满足时钟时序要求的能力。建议让所有DAC时钟走线都保持良好的匹配。同样的要求也适用于同步时钟走线和DCI(帧时钟)走线的匹配。这些时钟域之间没有匹配要求。例如，假设系统中有4个AD9122器件，DAC时钟走线长度、同步时钟走线长度、DCI走线长度分别为6 cm (2.4 in.)、8 cm (3.1 in.)、10 cm (3.8 in.)。将第一个AD9122的这三条走线分别表示为 $L1_{DAC}$ 、 $L1_{SYNC}$ 、 $L1_{DCI}$ ，则器件之间的走线匹配应为：

$$L1_{DAC} = L2_{DAC} = L3_{DAC} = L4_{DAC} = 6 \text{ cm}$$

$$L1_{SYNC} = L2_{SYNC} = L3_{SYNC} = L4_{SYNC} = 8 \text{ cm}$$

$$L1_{DCI} = L2_{DCI} = L3_{DCI} = L4_{DCI} = 10 \text{ cm}$$

同步时钟之间的任何偏斜都会直接转变为DAC输出之间的偏斜。DAC时钟之间的偏斜对DAC输出具有同样的影响。如果走线匹配不佳，多个DAC器件之间将难以满足时序要求。这是因为，多个器件的等效建立和保持窗口等于各器件的各窗口的并集，不匹配会导致各窗口彼此偏移。DAC速率越高，可用采样窗口越小，匹配就变得越发重要。建议将走线匹配控制在 $\pm 0.5 \text{ mm} (\pm 20 \text{ 密耳})$ 范围内。

表3. AD9122同步系统要求

同步模式	时序要求	同步时钟速率	帧时钟速率
FIFO速率模式	同步时钟—DAC时钟建立和保持时间	$f_{DATA}/8$ 或更慢	$f_{DATA}/8$ 或更慢
数据速率模式	同步时钟—DAC时钟建立和保持时间 DCI—DAC时钟建立和保持时间	f_{DATA} 或更慢	$f_{DATA}/2$ 或更慢

同步设计示例

以下部分为延迟差异计算、硬件设计和软件设计的示例，假设系统中有两个AD9122器件， $f_{DATA} = 200$ MSPS，采用4倍插值和字接口模式。

延迟差异计算

两个AD9122器件的DAC输出之间的最大偏斜为：

$$T_{MAXSKEW} = \left(2 - \frac{1}{\text{Interpolation Rate}} \right) \times \frac{1}{f_{DATA}} = \left(2 - \frac{1}{4} \right) \times 5 \text{ ns} = 8.75 \text{ ns}$$

如果DAC输出之间的最大偏斜在系统要求范围内，则无需同步这两个器件或者产生同步时钟和帧时钟。这种情况下，请参考图2中的流程图来初始化FIFO。如果系统要求更小的偏斜，则需要开启同步状态机，并且系统设计必须满足时序要求。

硬件设计

本例使用ADI公司的AD9516时钟芯片，它提供6路LVPECL输出和4路LVDS/CMOS输出。LVPECL输出具有一个分频比最高为32的可编程分频器。LVDS/CMOS输出具有两种这种分频器，提供更大的分频比。LVDS/CMOS输出还有一条精密延迟线，可用于调整特定输出相对于其它输出的偏斜。更多产品详情，请参阅AD9516数据手册。

本例将两路LVPECL输出指定为两个AD9122器件的DAC时钟。DAC需要极高质量和低抖动的时钟以实现额定性能。同步要求DAC时钟之间的偏斜非常低。LVPECL信号的出色抖动和偏斜性能能够很好地满足这一要求。使用一路LVDS输出作为两个AD9122器件的同步时钟。同步时钟同样需要低抖动的时钟源来实现最佳同步精度。LVDS信号具有非常好的抖动性能。更重要的是，AD9516 LVDS输出的精密延迟线可用于调整同步时钟与DAC时钟之间的时序，从而很容易满足建立和保持时间要求。使用FIFO速率同步

模式和fDATA/8的同步时钟频率(25 MHz)。另一路LVDS输出用来产生FPGA/ASIC的参考时钟，确保来自FPGA/ASIC的数据和DCI时钟能够相位锁定DAC时钟。

帧时钟由产生DCI和数据的FPGA产生，用于复位两个器件的FIFO。本例选择 $1/8 \times f_{DATA}$ 的帧信号速率。帧的上升沿应与DCI对齐，如AD9122数据手册所述。两个AD9122器件的数据、DCI和帧时钟走线长度应匹配，DAC时钟和同步时钟走线同样应匹配。

软件设计

通过SPI命令开启AD9122的同步状态机。使能同步功能之前，需要确保所有相关时钟都正常工作并且稳定。图4的流程图显示了使能各AD9122的同步引擎并确认FIFO对齐的步骤。完成这些步骤后，两个AD9122的DAC输出便已实现对齐。

一次性同步实现

对于无法实现“同步的系统设计考虑”部分所述连续同步方案的用户，可以使用另一种方案，在上电时或必要时执行一次性同步。在一次性同步方案中，同步引擎在DAC达到同步状态后关闭。因此，DAC不会跟踪系统的偏斜漂移情况，不会向用户报告同步锁定/解锁状态。

这种方案可以降低同步时钟与DAC时钟之间的时序要求，但同步精度略有损失，变为 ± 1 DAC周期，而不是1 DAC周期。实现一次性同步时，必须遵循图5所示的流程图步骤。

总结

AD9122具有多芯片同步功能，多个AD9122器件的DAC输出可以在一个DAC时钟周期内同步。同步模式有两种：数据速率模式和FIFO速率模式。两种模式能够实现相同的同步精度，但系统设计要求不同。一次性同步选项允许用户在实现同步后关闭同步引擎。与连续同步模式相比，其同步对齐精度较低，系统设计的时序要求也较低。

1. WRITE REG 0x10[7] = 0b0.
2. WRITE REG 0x04[5] = 0b0.
3. WRITE REG 0x06[5] = 0b1.

SYSTEM SETUP;
PROGRAM DAC INTERPOLATION MODES;

SET FIFO OFFSET

ENABLE SYNC ENGINE

ENABLE SYNC LOST INTERRUPT

REG 0x06[5] = 0b1

SYNC LOST/LOCK
FLAG BITS?

REG 0x06[5:4] = 0b00

DISABLE SYNC;
DISABLE SYNC LOST
INTERRUPT;
CLEAR SYNC LOST FLAG

REG 0x06[5:4] = 0b01

DISABLE SYNC ENGINE;
ENABLE FIFO WARNING INTERRUPT

FIFO THERMOMETER¹
READBACK STABLE?

TURN OFF SYNC CLOCK (OPTIONAL)

REG 0x06[5] ≠ 0b0 OR
[1] ≠ 0b0 OR [0] ≠ 0b0

SYNC LOST/FIFO WARNING
FLAG BITS?

DISABLE FIFO WARNINGS
AND CLEAR FLAGS

1. WRITE REG 0x04[1:0] = 0b00.
2. WRITE REG 0x06[1:0] = 0b11.

READ FIFO THERMOMETER;
CALCULATE AND RECORD
OPTIMAL FIFO OFFSET;

WAKE UP DACs AND RUN

1. DAC HARDWARE RESET.
PULL THE DAC RESET PIN FROM HIGH TO LOW THEN BACK TO HIGH.
2. SLEEP DACs.
WRITE REG 0x41[7] = 0b1, WRITE REG 0x45[7] = 0b1.
3. SET UP DAC INTERPOLATION MODE.
PROGRAM REG 0x1C TO 0x1E.
4. RUN CLOCKS (DAC CLOCK, SYNC CLOCK, DCI, AND FRAME CLOCK)
5. SET UP SYNC MODE. IN THIS CASE, IT IS FIFO RATE.
WRITE REG 0x10[6] = 0b0.

WRITE REG 0x17 = OFFSET
OFFSET IS DETERMINED BY THE STEP IN THE DOTTED BOX BELOW.
OFFSET = 0 WHEN THE DOTTED BOX IS IN THE FLOW.
OFFSET = (SAVED VALUE) WHEN THE DOTTED BOX IS NOT IN THE FLOW.

WRITE REG 0x10 = 0x88, IF RISING EDGE SYNC.
OR
= 0x80, IF FALLING EDGE SYNC.

WRITE REG 0x04[5] = 0b1.

READ REG 0x06[5:4] ((SYNC SIGNAL LOST; SYNC SIGNAL LOCKED)).
IF THE SYNC-DAC SETUP/HOLD TIMES ARE NOT MET, THE SYNC MAY NOT LOCK. CHANGE THE SYNC EDGE WHEN REENABLING THE SYNC NEXT ROUND.

1. WRITE REG 0x10[7] = 0b0.
2. WRITE REG 0x04[1:0] = 0b11.

READ REG 0x19 FIVE TIMES.
ALL FIVE READBACKS SHOULD BE THE SAME.
THE VALUE CAN BE ANY LEGAL THERMOMETER VALUE.

TURN OFF THE EXTERNAL SYNC CLOCK (OPTIONAL).

READ REG 0x06. IF NO SYNC AND FIFO FLAGS COME ON, THE SYSTEM IS SYNCHRONIZED AND OPERATING NORMALLY.

THIS OPERATION IS A CALIBRATION STEP. IT ONLY NEEDS TO BE DONE ONCE IN THE SYSTEM DEVELOPMENT ON ANY OF THE DACs THAT NEED TO BE SYNCHRONIZED IN THE SYSTEM. IT SHOULD BE BYPASSED IN THE FLOW LATER ON. THE VALUE IS SAVED IN THE MEMORY FOR STEP 2 (SET FIFO OFFSET).

READ REG 0x19 AND CALCULATE ITS OFFSET FROM THE OPTIMAL VALUE ACCORDING TO THE EQUATIONS BELOW.
IF REG 0x19 READBACK (CONVERTED TO DECIMAL NUMBER) ≤ 4 ,
OFFSET = 4 - (REG 0x19 READBACK);
IF REG 0x19 READBACK > 4 ,
OFFSET = 12 - (REG 0x19 READBACK);

1. WAKE UP DACs.
WRITE REG 0x41[7] = 0b0, WRITE REG 0x45[7] = 0b0.
2. RUN THE DATA.

¹THERMOMETER CODE IS BASE ONE CODE.
FOR EXAMPLE, THERMOMETER CODE
00001111 IS 3 IN DECIMAL CODE.

图5. SPI命令流程图(FIFO速率模式、一次性同步)

注释

注释