

## 了解同步 FIFO

相关项目：无

相关组件系列：CY7C42x5 / CY7C42x1

### 摘要

AN1042 简单介绍了同步 FIFO 的特性和功能。该应用手册还提供了同步 FIFO 的带宽和深度扩展方面的信息。

## 简介

由于同步 FIFO 的操作速度非常快，因此能够成为高性能系统的理想选择。除此之外，同步 FIFO 还提供了多项其他优势，能够提高系统性能并降低系统的复杂性。具体包括各个状态标志：同步标志、半满、可编程近空和近满标志。这些 FIFO 还具有带宽扩展、深度扩展和重新发送等特性。异步 FIFO 需要在不存在外部时钟参考的情况下生成读和写脉冲，所以，与其相比，同步 FIFO 更适用于高速操作，因为同步 FIFO 通过使用自由运行的时钟对内部操作进行定时。

## 范围

该应用手册对同步 FIFO 的架构进行了概述，并讨论了它的主要特性、使用指导和典型应用。

该应用手册只提供了大概内容，并不是赛普拉斯某一特定同步 FIFO 器件的特性信息。有关特定器件的详细信息，请参考赛普拉斯网站（[www.cypress.com](http://www.cypress.com)）上相关器件的数据手册。

## 同步 FIFO 架构

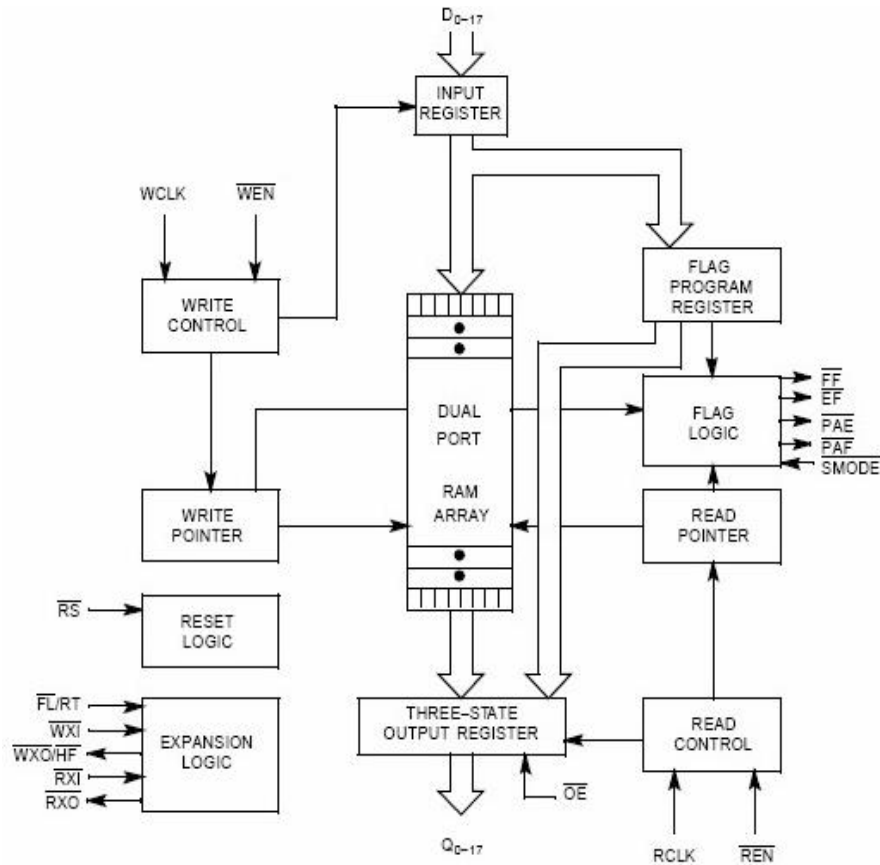
一个同步 FIFO 由寄存器阵列、标志逻辑和扩展逻辑等三个基本模块组成。图 1 显示的是一个同步 FIFO 的逻辑框图。寄存器阵列由各双端口寄存器单元组成。这些单元支持同时对写端口和读端口进行访问。该特性提供了 FIFO 内部同步化的能力。两端口之间的访问不存在任何时序或相位限制。因此在一个端口上以某一速度进行写入操作时，剩下的端口可以按另一速度进行读取，两者相互独立。这样有益于选择最佳速度进行读取和写入寄存器阵列中的数据。赛普拉斯提供了同步 FIFO CY7C42x5（x9 位宽）和 CY7C42x5（x18 位宽）。它们分别提供了 66 MHz 和 100 MHz 的高速度操作。

读地址指针和写地址指针控制着数据进入和退出寄存器阵列。每次操作完成后，相关指针都会增加，这样能够对阵列中下一个地址进行访问。有关详细信息，请参考同步 FIFO 中的引导。

标志逻辑会对两个地址指针中的值进行比较。如果两个地址指针中的值相差为零，那么 FIFO 为空，并且空标志被激活。如果该差别等于器件深度，那么 FIFO 已满，并且满标志被激活。其他标志（如半满、可编程近空和可编程近满标志）将以相似的方式被激活。可编程标志是通过对偏移寄存器中编程的值与 FIFO 的字的数量进行比较后生成的。

最后，扩展逻辑用于创建更深逻辑的 FIFO，可通过级联各个器件实现（深度扩展）。在正常的“无深度级联”模式下，每个地址指针在达到最大值之后会返回到零。而在深度扩展模式下，当地址指针达到最大值后，会有一个脉冲被驱动到一个扩展引脚，进而传输一个令牌给另一个 FIFO。该令牌被传递后，地址指针不再继续增加，直到令牌返回为止。实际上，执行写入或读取操作的责任已经被传给了另一个器件。在深度扩展配置下，始终只有一个 FIFO 负责执行读取操作，并且只剩下一个负责执行写入操作。令牌返回时，地址指针复位为零，操作恢复。

图 1. 逻辑框图 — 同步 FIFO 架构 (CY7C42x5)



## 复位

加电后，FIFO 必须被复位。复位器件会将读和写地址指针设置为零、清空输出数据寄存器并将各个状态标志设置为空器件的状态。通过将  $\overline{RS}$  引脚设置为低电平可复位器件。同步 FIFO 需要在  $\overline{RS}$  上生成一个上升沿。这样便允许各器件（如处理器监控芯片）直接驱动  $\overline{RS}$ 。这些器件在  $V_{CC}$  下降后会将复位激活，并在最短的时间内将其保持为低电平，以便保证  $V_{CC}$  和所有时钟进入稳定状态。

在  $\overline{RS}$  被激活的过程中，不能对器件进行读或写操作。可通过取消激活读取和写入使能 ( $\overline{REN}$ ,  $\overline{WEN}$ )，或者通过将  $RCLK$  和  $WCLK$  设置为低状态实现禁止读写操作。在  $\overline{RS}$  的取消激活（上升）沿之后，需要保持写入和读取操作为禁用状态，直到复位恢复时间  $t_{RSR}$  结束为止。

**注意：**复位是一个异步操作，在不存在  $WCLK$  和  $RCLK$  的切换情况下仍能完成。

## 状态标志

状态标志，如空标志、可编程近空标志、半满标志、可编程近满标志和满标志 ( $\overline{EF}$ 、 $\overline{PAE}$ 、 $\overline{HF}$ 、 $\overline{PAF}$ 、 $\overline{FF}$ ) 均用于确定 FIFO 状态。这些标志是通过将读和写地址指针中的值进行比较而生成的。外部控制逻辑需要使用这些标志来确定是否可以在 FIFO 上进行读或写操作。FIFO 的标志逻辑禁止读取空 FIFO，并禁止对满 FIFO 进行写入操作。当对空 FIFO 进行读取操作时，输出始终显示最后被读取的有效数据。对满 FIFO 进行写入操作是无效的。

空标志 ( $\overline{EF}$ ) 和满标志 ( $\overline{FF}$ ) 是同步标志，即它们与各自的相对时钟同步。空标志 ( $\overline{EF}$ ) 与读取时钟 ( $RCLK$ ) 同步，满标志 ( $\overline{FF}$ ) 与写入时钟 ( $WCLK$ ) 同步。将标志与相对应的时钟进行同步便不再需要进行外部同步。在大多数情况下，写入逻辑要保证 FIFO 在被写入前未空。同样，读取控制逻辑会在读取 FIFO 之前进行检查空标志 ( $\overline{EF}$ )。CY7C42x1 FIFO 的可编程近空 ( $\overline{PAE}$ ) 和可编程近满 ( $\overline{PAF}$ ) 标志是同步的。 $\overline{PAE}$  标志与  $RCLK$  同步， $\overline{PAF}$  标志与  $WCLK$  同步。其他 FIFO，如 CY7C42x5，通过使用

$\overline{\text{SMODE}}$  控制信号允许可编程标志的同步和异步操作。有关编程  $\overline{\text{PAE}}$  和  $\overline{\text{PAF}}$  标志以及标志操作的详细信息，请参考赛普拉斯网站 ([www.cypress.com](http://www.cypress.com)) 上的器件数据手册。

半满标志 ( $\overline{\text{HF}}$ ) 是异步标志，因为该标志是否被读取或写入控制逻辑使用是不确定的。

## 作为异步 FIFO 使用

当数据即将进入或从器件退出时，用户通常不希望将一个自由运行的时钟连接到同步 FIFO 的 RCLK 和 WCLK 引脚上，他们更愿意为这些时钟提供脉冲。对于同步 FIFO，该操作模式完全可行，但需要考虑一些问题。想要将同步 FIFO 作为异步 FIFO 使用，只要给同步 FIFO 的 RCLK 和 WCLK 引脚提供脉冲。WCLK 的每一个上升沿会执行数据写入操作，而 RCLK 的每一个上升沿会执行数据读取操作。在操作过程所有阶段内（复位和重新传输除外），读取和写入使能 ( $\overline{\text{REN}}$ ,  $\overline{\text{WEN}}$ ) 可被设置为低电平。需要保证在复位和重新传输的过程中，不会进行任何读或写操作。可通过将  $\overline{\text{REN}}$  和  $\overline{\text{WEN}}$  驱动为高电平，或者将 RCLK 或 WCLK 设置为低电平实现。将 RCLK 和 WCLK 设置为高电平会生成内部时钟脉冲（如果其相对应的使能已被激活），这是不允许的。

标志更新周期是有关该操作模式的另一个问题。因为器件不使用任何自由运行时钟，所以同步标志不会总是被自动更新。例如，对于一个空 FIFO，将对其进行一些写入操作。此时，FIFO 不再是空的，但是因为不存在“标志更新周期”， $\overline{\text{EF}}$  仍然被激活。从用户角度来看，在这种情况下，需要两个读取周期才能对 FIFO 中的第一字进行读取：第一个周期是标志更新周期，第二个将执行第一次读取操作。

标志得到更新后，在每一个 RCLK 的上升沿，数据会被读取。若想在每次  $\overline{\text{EF}}$  被激活都能添加这个附加的读取周期，需要特别小心。同样， $\overline{\text{FF}}$  在满边界时也需要一个标志更新周期。每次 FIFO 已满，并且完成了一次读取操作，便需要两个 WCLK 上升沿用于写入下一个数据。

## 标志更新周期

因为空标志和满标志是同步的，因此它们需要相应时钟的一个上升沿，用于更新为当前的值。

在满边界或空边界中存在一个被称为“标志更新周期”的死周期。这个死周期能够保证满标志和空标志能够在至少一个完整的时钟周期内被激活，进而可被相对应控制逻辑观察到。

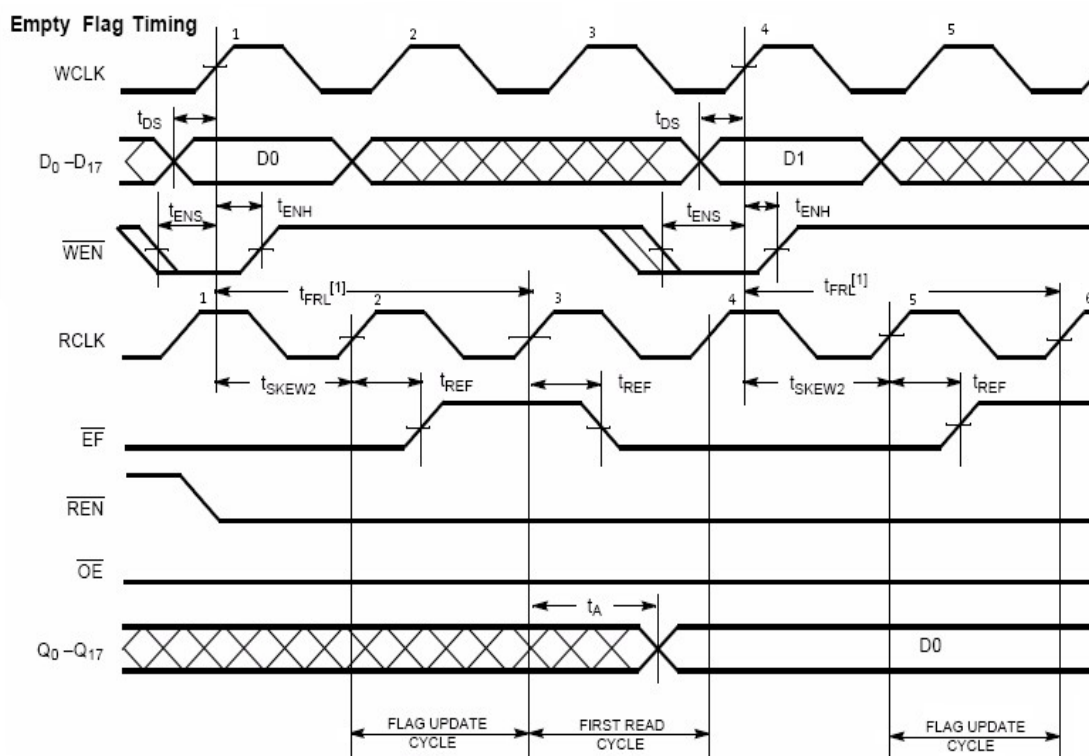
例如，我们有一个空 FIFO。如图 2 所示，在 WCLK 的时钟周期 1 内将对器件进行一次写入操作。此时，该器件将不再是空的。因为空标志 ( $\overline{\text{EF}}$ ) 与读取时钟同步，所以该标志不会被取消激活状态，直到一个 RCLK 上升沿到来为止。读取操作被禁止，直到  $\overline{\text{EF}}$  被取消激活状态为止。第一个 RCLK 上升沿（第一次写入完成后 — RCLK 的时钟周期 2）会更新标志，并且第二个 RCLK 上升沿（时钟周期 3）会读取第一个字。第一个 RCLK 周期作为一个死周期或标志更新周期。该附加周期能够保证空标志 ( $\overline{\text{EF}}$ ) 的激活状态或取消激活状态始终被延长至少一个周期。

在 RCLK 和 WCLK 的异步条件下（常见于 FIFO 应用），标志激活可以无限地小，并且不带这个死周期。

对于使用了自由运行时钟的应用，该“死”周期或“标志更新”周期都是透明的，并且无需关注它们。读取控制逻辑会忽略读取使能 ( $\overline{\text{REN}}$ )，直到空标志 ( $\overline{\text{EF}}$ ) 被取消激活为止。因此，虽然已经为 RCLK 周期 2 和 4 的读取操作设定了控制信号，但数据仍然不会出现在数据线上。在  $\overline{\text{EF}}$  被取消激活后，RCLK 的第一个上升沿会读取 FIFO 的第一字（如果  $\overline{\text{REN}}$  被激活 — RCLK 时钟周期 6）。

对于不使用自由运行时钟的应用，RCLK 需要从低电平切换为高电平两次才可以读取第一个字 — 第一次用于“标志更新周期”，第二次用于读取第一个字。

**注意：**标志更新周期发生在空边界和满边界。对于  $\overline{\text{PAE}}$  和  $\overline{\text{PAF}}$  标志，不存在“标志更新周期”。

图 2. 标志更新周期<sup>[1]</sup>


<sup>1</sup> 当  $t_{SKEW2} >$  最短规格,  $t_{FRL}$  (最长) =  $t_{CLK} + t_{SKEW2}$ 。当  $t_{SKEW2} <$  最短规格,  $t_{FRL}$  (最长) = 或  $(2 \times t_{CLK} + t_{SKEW2})$  或  $(t_{CLK} + t_{SKEW2})$ 。延迟时序仅应用于空边界 ( $\overline{EF} = \text{LOW}$ )。

## 重新传输

重新传输特性用于重新对已经读取的 FIFO 数据模块执行读取操作。该特性通常用于串行通信接口。如果在数据传输过程中发生了错误，能够重新从 FIFO 传输数据包，然后再通过串行媒介重新发送。

通过给 FIFO 重新传输 ( $\overline{RT}$ ) 引脚提供脉冲，可以执行该特性。通过将  $\overline{RT}$  引脚驱动为低电平，可将 FIFO 的读地址指针设置到物理位置（零）。图 3 显示的是重新传输操作。请注意，为了准确执行重新传输，必须在将需要重新传输的数据写入 FIFO 前，复位 FIFO。

例如：如果您想将一个 1 K 深的数据包发送给其他电路板。该数据能被写入到一个 FIFO 内，然后被传送给一个串行收发器，该器件通过一个串行媒介会继续发送数据。首先，FIFO 被复位，FIFO 的读和写地址指针被设置到位置零。1 K 的数据包被写入 FIFO 内。 $\overline{EF}$  被取消激活，并且串行收发器器件开始读取 FIFO。

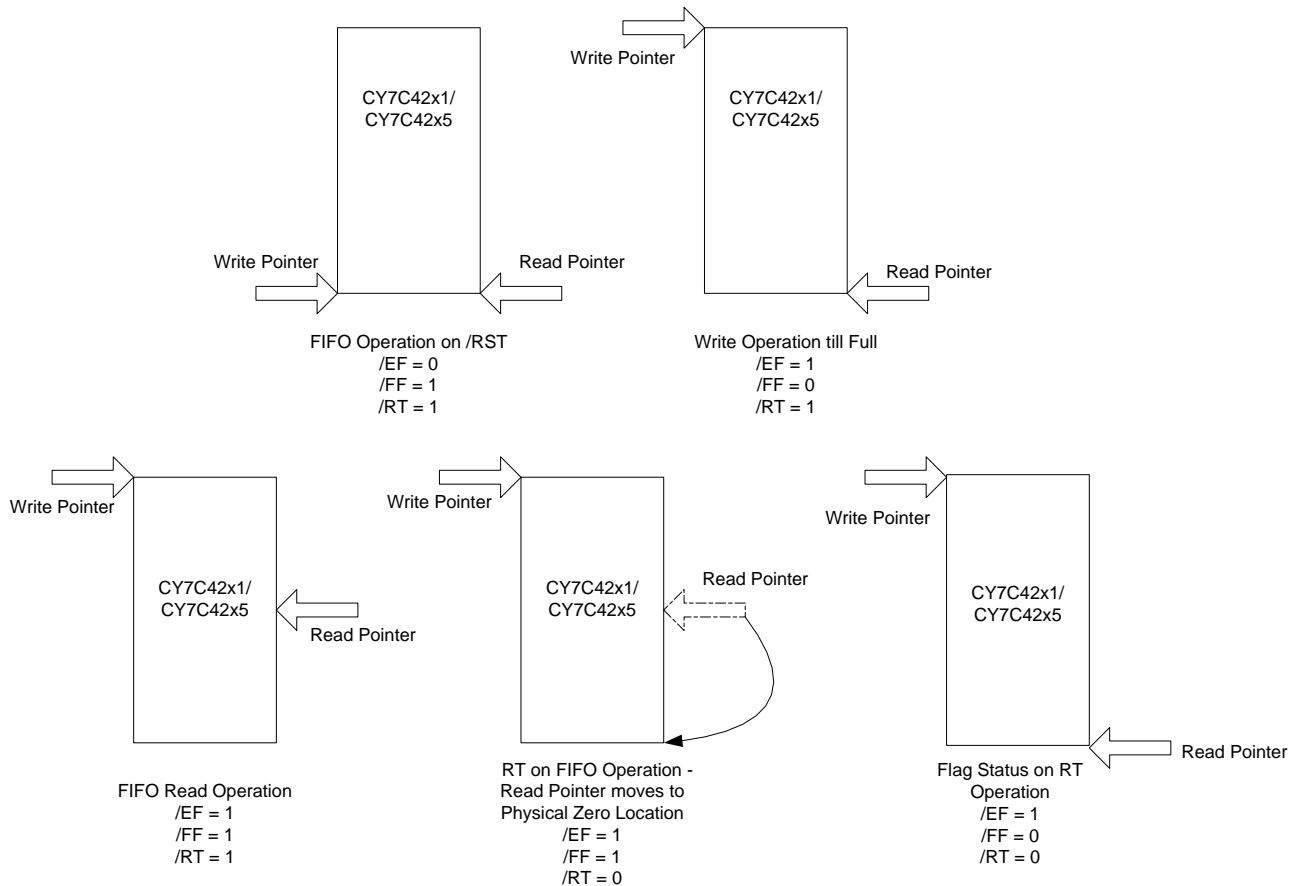
当数据被读取时，读地址指针将依次增加，直到达到位置 1024 为止，这时候 FIFO 将变为空。请注意，虽然 FIFO 中的数据已被读取，但仍未擦出 FIFO 中的这些数据。如果在读取过程中出现了错误，则  $\overline{RT}$  引脚可被提供脉冲，使读地址指针返回到位置零，并且数据包被重新发送到目的位置。该过程可以重复无限次。

请注意， $\overline{RT}$  引脚只会将读地址指针复位到位置零。FIFO 不知道特定的数据包被存储到器件中的什么地方。需要重新传输的数据包被写入之前，您必须复位器件。

在正常的 FIFO 操作过程中，或在  $\overline{RT}$  引脚已被取消激活后，FIFO 可随时执行正常的读/写操作。在  $\overline{RT}$  被激活的过程中，不能进行任何读或写操作。

**注意：**重新传输是一个异步操作，即便不存在 WCLK 和 RCLK 的切换情况，它仍能执行。

图 3. 重新传输中的 FIFO 操作



## 扩展配置

### 带宽扩展

带宽扩展用于创建数据带宽更大的 FIFO。两个 x18 FIFO 可用于扩大带宽，以便创建一个 x36 FIFO 等。与正常模式相比，带宽扩展模式下进行的读取、写入和重新传输操作均是相同的。实际上，FIFO 对其使用的模式是随机的。

为了进行 FIFO 带宽扩展，各标志需要组成“复合标志”。通过对每个 FIFO 中的各个标志进行外部的 AND 运算，可以执行该操作。

必须将复合标志应用于空标志和满标志。标志的组合可以保证 FIFO 的同步性（各 FIFO 的字数量是相等的）。请参考图 4。

我们需要了解经过带宽扩展的 FIFO 是如何失去它的同步性的。问题的根源在于 RCLK 和 WCLK 的异步关系。例如，我们有两个经过带宽扩展的 FIFO，每一个都有一字。由于读取和写入时钟之间的相位存在差异，FIFO 可能在接近同一时间接收一个读操作和写操作。如果先执行读取操作，那

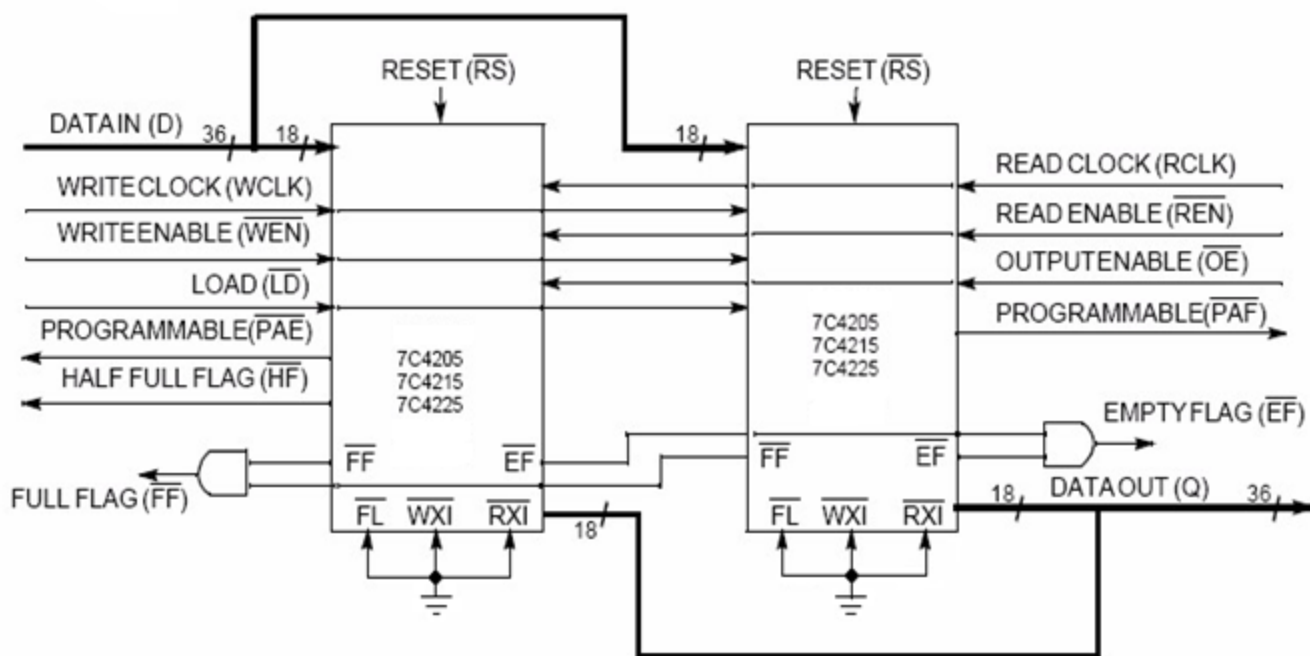
么将在写入一个字前，器件会变空。这时器件需要等待一个标志更新周期。实际上，将失去一个读取周期。在其他情况下，写入操作将发生在读取操作前，那么器件将瞬间拥有两个字，然后再返回到一个字。请注意，因为 FIFO 永远不会变为空，所以不存在标志更新周期。

经过带宽扩展的 FIFO 在这种情况下可有不同的响应，因而会失去同步性。短时钟滞，甚至器件处理过程存在的差异都可能导致一个 FIFO 先接收读取操作，而其他 FIFO 则会先执行一次写入操作。当一个或多个器件需要一个标志更新周期，FIFO 会失去同步性。

通过将各个 FIFO 的空标志组成一个复合空标志，任何 FIFO 为空时，可以取消激活读取使能（REN）。如果 REN 在至少一个时钟周期内被取消激活状态，则所有 FIFO 会得到所需的标志更新周期，并且 FIFO 会保持它的同步性。

请注意，这种情况也适合于满边界处的满标志。当某个复合标志被激活时，则驱动 REN 和 WEN 的控制逻辑需要取消这些使能的有效状态。

图 4. 同步 FIFO 的带宽扩展 (CY7C42x5)



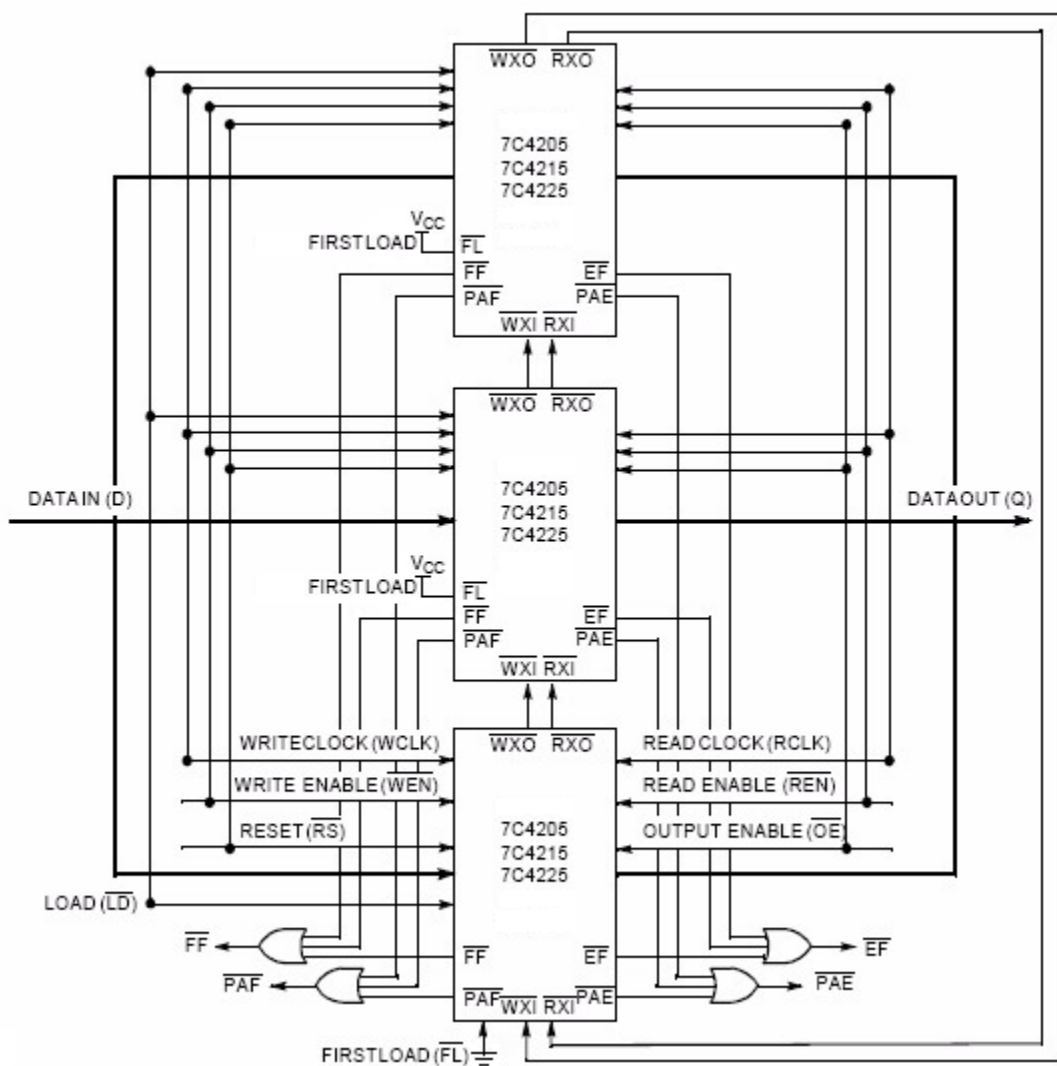


## 深度扩展

深度扩展用于级联多个 FIFO，以便创建一个逻辑更深的 FIFO 缓冲区。同步 FIFO 通过使用令牌传递方式进行深度扩展。每个 FIFO 被并行连接（使用共同的读取和写入数据总线），并且使用共同的控制线，如  $\overline{WEN}$ 、 $\overline{REN}$ 、 $\overline{RCLK}$  和  $\overline{WCLK}$  等。在令牌链路中，一个器件将其第一个负载（ $\overline{FL}$ ）引脚接地，作为该链路的“第一个”器件。所有器件需要将

$\overline{FL}$  连接到  $V_{CC}$ 。当第一个器件的写地址指针达到最大值后，将有一个扩展脉冲被驱动到  $\overline{WXO}$  引脚。第一个器件的  $\overline{WXO}$  引脚连接下一个器件的  $\overline{WXI}$  引脚，该引脚会观察到扩展脉冲，进而负责执行下一个写入操作。同样，当读地址指针达到最大值后，它会通过  $\overline{RXO}$  将令牌传递给下一个器件的  $\overline{RXI}$  引脚。图 5 是 CY7C42x5 FIFO 深度扩展配置框图。

图 5. 同步 FIFO (CY7C42x5) 的深度扩展



为了保证正常的操作，必须使用复合空标志和满标志。复合标志是通过对令牌传递链路中各个 FIFO 的标志进行外部 OR 运算生成的。该操作能够保证在复合满标志被激活前，所有 FIFO 已满。同样，在复合空标志被激活前，所有

FIFO 必须为空。在深度扩展模式下，不能使用  $\overline{PAE}$ 、 $\overline{PAF}$  和  $\overline{HF}$  标志。虽然这些标志在一个特定 FIFO 中可以作为字数量的准确指示，但它们的组合则能够提供有关经过深度扩展 FIFO 缓冲区的总体状态。

不具有片上深度扩展逻辑的 FIFO 仍能够执行深度扩展。使用乒乓方法时可对各 FIFO 的数据进行交换的读取和写入操作。执行写入操作的方法是对各器件中每一个数据操作进行反转。例如，如果对两个 FIFO 进行了深度扩展，那么第一次写入操作针对 FIFO#1 进行，第二次写入针对 FIFO#2 进行，第三次则对 FIFO#1 进行，并逐渐反转进行。读取操作的执行方法也是相似的。

## 同步 FIFO 的应用

若想运行于不同的速度，或使用异步时钟的处理器和外设器件的数据执行传送操作，FIFO 是一个理想的选择。例如，现代处理器比它连接的外设器件运行速度快。使用 FIFO 可以确保在处理器与外设器件进行数据交换时，处理速度不会降低。如果外设器件运行速度比处理器的快，则使用 FIFO 仍能够解决数据交换问题。

在电脑网络和数字通信交换中，数据被分成多个数据块，并且在数据线上传输。数据被分成各数据块后，会在数据线上高速传输，这样便需要使用 FIFO 来执行数据传送。

FIFO 在 HDTV/IPTV 的机顶盒中非常有用。它控制着主 CPU（执行 MPEG 编码/解码和音频处理）和 CPLD（通信处理器，用于捕获输入信号）之间的数据流。

数据获取设备是一种常见的总线速度不对称情况。在这种情况下，以不同的本地总线频率运行的各个电路板需要进行数

据交换，或一个系统总线速度更高的主控制器进行数据交换。FIFO 能够执行该操作，因为它们可以使用两个不同的输入时钟和输出时钟。这样能够使数据流与本地总线频率同步。该特性的根源由 FIFO 的双端口寄存器单元引起，这些单元允许同时对两个独立端口进行无约束的访问。

FIFO 被广泛用于处理器间的通信，用于对运行速度不同的各个处理器进行数据传输，并保证不会发生过长的等待状态延迟。FIFO 还可以用于缓冲串行数据，如通信系统中的视频/语音和数据块。

与双端口寄存器相比，FIFO 不具有寻址能力。FIFO 和它的名字一样是数据缓冲区，它们仅支持串行访问，因此取消了对外部寻址的需要。这样便降低了操作的复杂性和引脚数量，并且节省电路板空间。

## 总结

对于以不同速度运行或使用异步时钟源的系统而言，同步 FIFO 非常适用于数据传输。它们提供了高达 100 MHz 的高速度操作；通过将时钟输入连接到自由运行时钟，可将 FIFO 设置为同步模式，进而保证速度最好。此外，通过为时钟输入提供脉冲，可设置 FIFO 作为异步模式使用。重新传输和同步标志等特性提高了器件的灵活性和易用性。还可以将这些 FIFO 进行级联，实现深度扩展和带宽扩展（该操作在单一设备中是不可行的）。因此，赛普拉斯的同步 FIFO 非常适用于缓冲和同步化高性能应用中的数据。



## 文档修订记录

文档标题：了解同步 FIFO — AN1042

文档编号：001-95805

版本	ECN	变更者	提交日期	变更说明
**	4691563	YLIU	04/20/2015	本文档版本号为 Rev**，译自英文版 001-19979 Rev*D。

## 全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要找到离您最近的办事处，请参考[赛普拉斯所在地](#)。

### 产品

汽车级产品	<a href="http://cypress.com/go/automotive">cypress.com/go/automotive</a>
时钟与缓冲器	<a href="http://cypress.com/go/clocks">cypress.com/go/clocks</a>
接口	<a href="http://cypress.com/go/interface">cypress.com/go/interface</a>
照明和电源控制	<a href="http://cypress.com/go/powerpsoc">cypress.com/go/powerpsoc</a> <a href="http://cypress.com/go/plc">cypress.com/go/plc</a>
存储器	<a href="http://cypress.com/go/memory">cypress.com/go/memory</a>
光学导航传感器	<a href="http://cypress.com/go/ons">cypress.com/go/ons</a>
PSoC	<a href="http://cypress.com/go/psoc">cypress.com/go/psoc</a>
触摸感应	<a href="http://cypress.com/go/touch">cypress.com/go/touch</a>
USB 控制器	<a href="http://cypress.com/go/usb">cypress.com/go/usb</a>
无线/射频	<a href="http://cypress.com/go/wireless">cypress.com/go/wireless</a>

### PSoC®解决方案

[psoc.cypress.com/solutions](http://psoc.cypress.com/solutions)

PSoC 1 | PSoC 3 | PSoC 5

赛普拉斯开发者社区

社区 | 论坛 | 博客 | 视频 | 培训

此处引用的所有其他商标或注册商标都归其各自所有者所有。

	赛普拉斯半导体	电话	: 408-943-2600
	198 Champion Court	传真	: 408-943-4730
	San Jose, CA 95134-1709	网站地址	: <a href="http://www.cypress.com">www.cypress.com</a>

© 赛普拉斯半导体公司，2007-2015。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不会以明示或暗示的方式授予任何专利许可或其他权利。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯不保证产品能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

该源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途外，未经赛普拉斯明确的书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用于赛普拉斯软件许可证的限制。