AN65974

# 使用 EZ-USB® FX3™从设备 FIFO 接口进行设计

作者：Rama Sai Krishna V
软件版本：EZ-USB FX3 SDK1.3.3
相关应用笔记：AN75705 、AN68829

AN65974 介绍了 EZ-USB® FX3™的同步从设备 FIFO 接口。它详尽说明了硬件接口和标志的配置设置，并提供了相关的示例。它还提供了有关 GPIF™ II Designer 的参考信息，以便能轻松地使用从设备 FIFO 接口进行设计。另外，本笔记还提供了两个完整的设计示例，说明如何使用同步从设备 FIFO 将 FPGA 连接至 FX3。

## 目录



## 1. 简介

赛普拉斯的 EZ-USB FX3 是新一代 USB 3.0 外设控制器，可让开发者将 USB 3.0 功能添加到任何系统内。该控制器非常适合图像和视频设备、打印机、扫描仪等应用。

EZ-USB FX3 设有一个完全可配置的并行、通用可编程 GPIF II 接口，可将其连接至外部处理器、ASIC 或 FPGA。GPIF II 是赛普拉斯旗舰 USB 2.0 产品 FX2LP 中的 GPIF 的加强版本。GPIF II 通过各接口（如同步地址数据复用接口）为常用设备（如 FPGA，图像传感器和处理器）提供无缝连接。

同步从设备 FIFO 接口是 GPIF II 的普遍应用。该接口适用于连接至 EZ-USB FX3 的外部器件通过访问 FX3 FIFO 对其进行数据读取/写入操作的应用。不可通过从设备 FIFO 接口执行直接寄存器访问。

本应用笔记先简单介绍 GPIF II，然后详细讨论同步从设备 FIFO 接口。此外，还提供了两个完整的设计示例，用来说明如何在 FPGA 上执行一个与同步从设备 FIFO 相兼容的主控接口。同时也提供了 Xilinx Spartan 6 FPGA 和 Altera Cyclone III FPGA 的 Verilog 和 VHDL 文件。示例中包含了同步从设备 FIFO 的相应 FX3 固件项目。这些示例使用 Xilinx SP601 评估套件（用于 Spartan 6 FPGA）、Altera Cyclone III Starter 电路板（用于 Cyclone III FPGA）、FX3 开发套件（DVK）和 FX3 软件开发套件（SDK）来开发。

## 2. GPIF II

GPIF II 是一个可编程状态机，可以灵活地实现一个工业标准或专用接口。它能够作为主设备，也可以作为从设备运行。

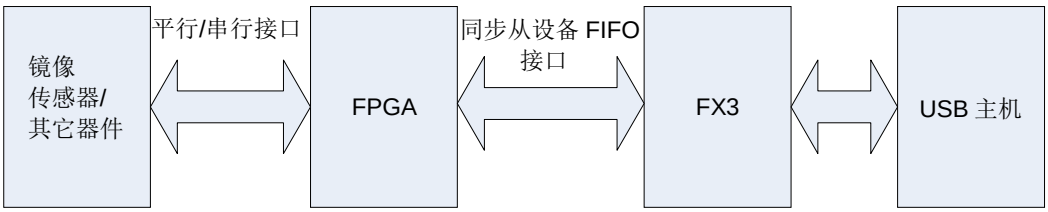
GPIF II 具有以下特性：

- 可用作主设备或从设备
- 提供 256 种固件可编程状态
- 支持 8 位、16 位和 32 位并行数据总线
- 接口频率可高达 100 MHz
- 使用 32 位数据总线时，可支持 14 个可配置的控制引脚；所有的控制引脚可作为输入/输出或双向引脚使用
- 使用 16 位或 8 位数据总线时，可支持 16 个可配置的控制引脚；所有的控制引脚可作为输入/输出或双向引脚使用

GPIF II 状态切换根据控制输入信号发生，而控制输出信号则由 GPIF II 状态转换所驱动。状态机的运行方式由一个描述符定义。该描述符为满足所需接口规格而设计。本质上，GPIF II 描述符是一组可编程寄存器配置。EZ-USB FX3 寄存器空间中的 8 KB 专门作为 GPIF II 波形存储空间，用于存储 GPIF II 描述符。

GPIF II 的常见实现是同步从设备 FIFO 接口。下面章节将详尽介绍该内容。图 1 显示了使用同步从设备 FIFO 接口的示例应用框图。

图 1. 示例应用框图




## 3. 同步从设备 FIFO 接口

同步从设备 FIFO 接口非常合适外部处理器或器件需要对 EZ-USB FX3 的内部 FIFO 缓冲区进行数据读/写访问的应用。寄存器访问不可通过从设备 FIFO 接口执行。同步从设备 FIFO 接口通常是 USB 应用的首选接口，可以满足高吞吐量的要求。

图 2 显示了同步从设备 FIFO 接口的接口框图。表 1 说明了图 2 中所显示的信号。

图 2. 同步从设备 FIFO 接口框图

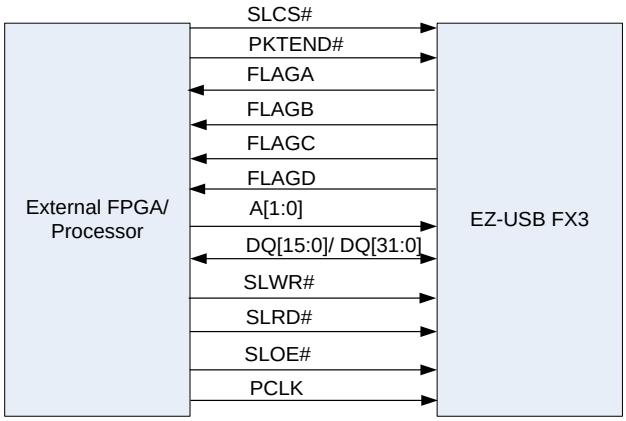



表 1. 同步从设备 FIFO 接口信号

| 信号名称 | 信号说明 |
|---|---|
| SLCS# | 这是从设备 FIFO 接口的芯片选择信号。必须激活它，才可以访问从设备 FIFO。 |
| SLWR# | 这是从设备 FIFO 接口的写入选通信号。必须激活它，才可以对从设备 FIFO 执行写入操作。 |
| SLRD# | 这是从设备 FIFO 接口的读取选通信号。必须激活它，才可以对从设备 FIFO 执行读取操作。 |
| SLOE# | 这是输出使能信号。该信号激活后，FX3 就可驱动从设备 FIFO 接口的数据总线。必须激活它，才可以对从设备 FIFO 执行读取操作。 |
| FLAGA/FLAGB/ FLAGC/ FLAGD | 这些信号均是 FX3 的标志输出。它们表示 FX3 套接字是否可用。[1] 在随附的示例项目中，FLAGA、FLAGB 使用于从设备 FIFO 写入操作，而 FLAGC、FLAGD 使用于从设备 FIFO 读取操作。 |
| A[1:0] | 这是从设备 FIFO 的 2 位地址总线。 |
| DQ[15:0]/ DQ[31:0] | 这是从设备 FIFO 的 16 位或 32 位数据总线。 |
| PKTEND# | 激活该信号后，可将短数据包或零长度数据包写入从设备 FIFO 内。 |
| PCLK | 这是从设备 FIFO 接口的时钟。 |

## 3.1 带两个地址线和带五个地址线的从设备 FIFO 之间的差别

带两个地址线的同步从设备 FIFO 接口支持多达四个套接字。如要访问四个套接字以上，请使用带五个地址线的同步从设备 FIFO 接口。除了额外的地址线之外，该接口还配有一个称为 EPSWITCH#的信号。引脚数量的增加导致用作标志的引脚数量降低，因此，需要将标志配置为一个 current_thread 标志。

使用带五个地址线的同步从设备 FIFO 接口时会产生额外的延迟：

- 每次开始传输时，在从地址得以读取到标志有效的期间将发生两个周期的延迟。
- 每当套接字地址切换时，将要经过多个周期的延迟来完成套接字的切换。

由于额外的延迟和其他接口协议要求，建议您只在应用需要访问四个 GPIF II 套接字以上时才使用带五个地址线的同步从设备 FIFO 接口。更多有关该接口的信息，请参考应用笔记 AN68829 — 用于 EZ-USB FX3 的从设备 FIFO 接口：5 位地址模式。

[1] 线程和套接字一节介绍了用于数据传输的套接字的概念。标志配置一节详细描述了各标志。

本应用笔记中的下面部分介绍了带两个地址线的同步从设备 FIFO 接口。

## 3.2 从设备 FIFO 接口的引脚映射情况

表 2 显示了从设备 FIFO 接口的默认引脚映射情况。该表还介绍了在 GPIF II 被配置为从设备 FIFO 接口情况下各个可用的 GPIO 引脚和其他串行接口（UART/SPI/I²S）。

可按需修改引脚映射情况，并且可通过 GPIF II Designer 工具添加或重新配置各个标志。更多信息，请查看标志配置一节。

表 2. 从设备 FIFO 接口的引脚映射情况

| EZ-USB FX3 引脚 | 使用 16 位数据总线的同步从设备 FIFO 接口 | 使用 32 位数据总线的同步从设备 FIFO 接口 |
|---|---|---|
| GPIO[17] | SLCS# | SLCS# |
| GPIO[18] | SLWR# | SLWR# |
| GPIO[19] | SLOE# | SLOE# |
| GPIO[20] | SLRD# | SLRD# |
| GPIO[21] | FLAGA | FLAGA |
| GPIO[22] | FLAGB | FLAGB |
| GPIO[23] | FLAGC | FLAGC |
| GPIO[24] | PKTEND# | PKTEND# |
| GPIO[25] | FLAGD | FLAGD |
| GPIO[28] | A1 | A1 |
| GPIO[29] | A0 | A0 |
| GPIO[0:15] | DQ[0:15] | DQ[0:15] |
| GPIO[16] | PCLK | PCLK |
| GPIO[33:44] | GPIO | DQ[16:27] |
| GPIO[45] | GPIO | GPIO |
| GPIO[46] | GPIO/UART_RTS | DQ28 |
| GPIO[47] | GPIO/UART_CTS | DQ29 |
| GPIO[48] | GPIO/UART_TX | DQ30 |
| GPIO[49] | GPIO/UART_RX | DQ31 |
| GPIO[50] | GPIO/I2S_CLK | GPIO/I2S_CLK |
| GPIO[51] | GPIO/I2S_SD | GPIO/I2S_SD |
| GPIO[52] | GPIO/I2S_WS | GPIO/I2S_WS |
| GPIO[53] | GPIO/SPI_SCK /UART_RTS | GPIO/UART_RTS |
| GPIO[54] | GPIO/SPI_SSN/UART_CTS | GPIO/UART_CTS |
| GPIO[55] | GPIO/SPI_MISO/UART_TX | GPIO/UART_TX |
| GPIO[56] | GPIO/SPI_MOSI/UART_RX | GPIO/UART_RX |
| GPIO[57] | GPIO/I2S_MCLK | GPIO/I2S_MCLK |

**注意**：如需 EZ-USB FX3 的完整引脚映射，请参见 EZ-USB FX3 SuperSpeed USB 控制器数据手册。

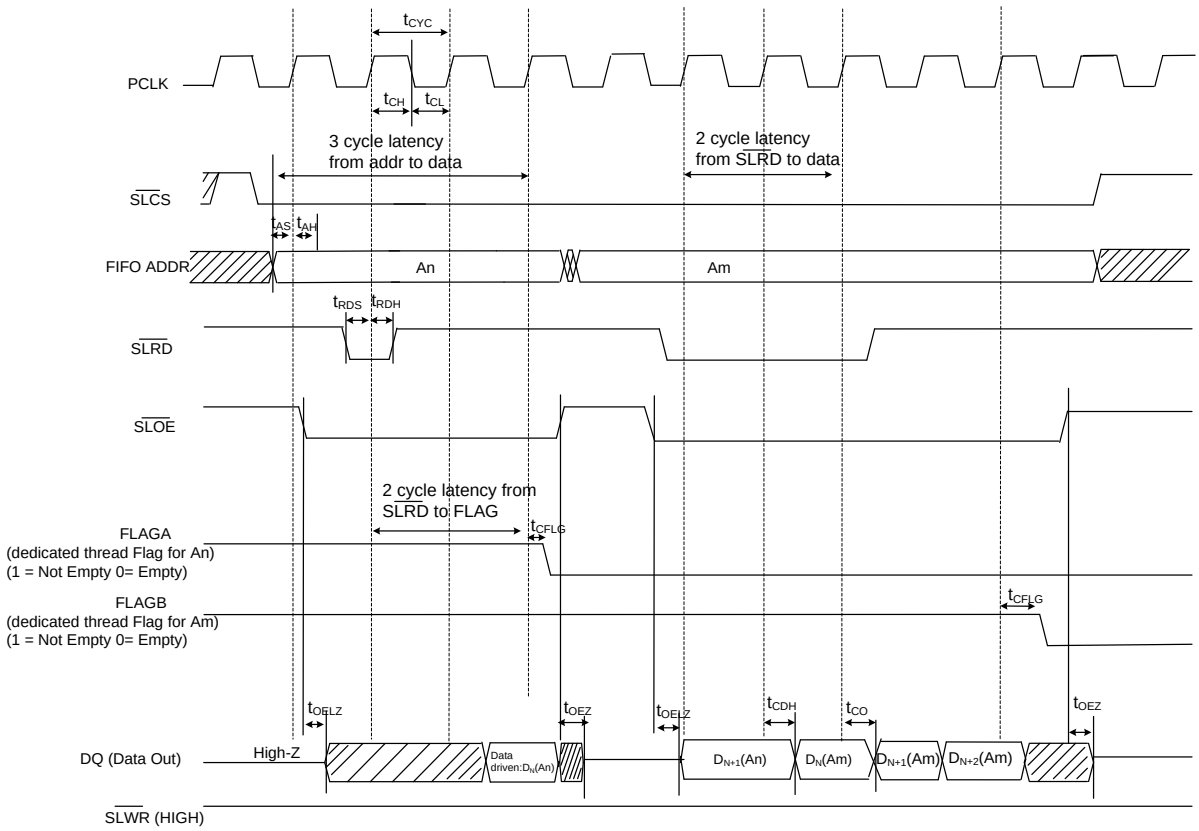
# 4. 从设备 FIFO 访问序列和接口时序

本节介绍了同步从设备 FIFO 接口的访问序列和时序。

作为接口主设备的外部处理器或器件可对 EZ-USB FX3 的内部 FIFO 缓冲区进行单周期或突发数据访问。外部主设备在 ADDR 线上输出 2 位地址，并激活读选通或写选通信号。EZ-USB FX3 激活标志信号，以表示缓冲区的空/满状态。

## 4.1 同步从设备 FIFO 接口时序

图 3. 同步从设备 FIFO 读序列



## 4.2 同步从设备 FIFO 读序列

从同步从设备 FIFO 接口进行读取的序列如下：

1. FIFO 地址稳定且 SLCS#信号被激活。
2. 激活 SLOE#。SLOE#仅是一个输出使能信号，其唯一功能是驱动数据总线。
3. 激活 SLRD#。

FIFO 指针在 PCLK 的上升沿上更新，同时，SLRD#被激活。这样会启动从新寻址的 FIFO 至数据总线的数据传输。经过 $t_{CO}$ 的传输延迟时间（从 PCLK 的上升沿算起）后即可提供新的数值。N 是从 FIFO 读取的第一个数值。要想驱动数据总线，还必须激活 SLOE#。

突发读取时将发生相同的事件序列。

**注意：**对于突发模式，SLRD#和 SLOE#在整个读取过程中保持激活状态。 当 SLOE#被激活时，将（使用之前已寻址的 FIFO 的数据）驱动数据总线。SLRD#被激活时，在每一个 PCLK 的连续升沿上，FIFO 指针会递增，且下一个数据值会被置于数据总线上。

**标志使用情况：**外部处理器监控标志信号来控制数据流量。标志信号是 EZ-USB FX3 的输出，可对其进行配置，以显示专用线程或正在寻址的线程的空/满/局部状态。

图 4. 同步从设备 FIFO 写序列

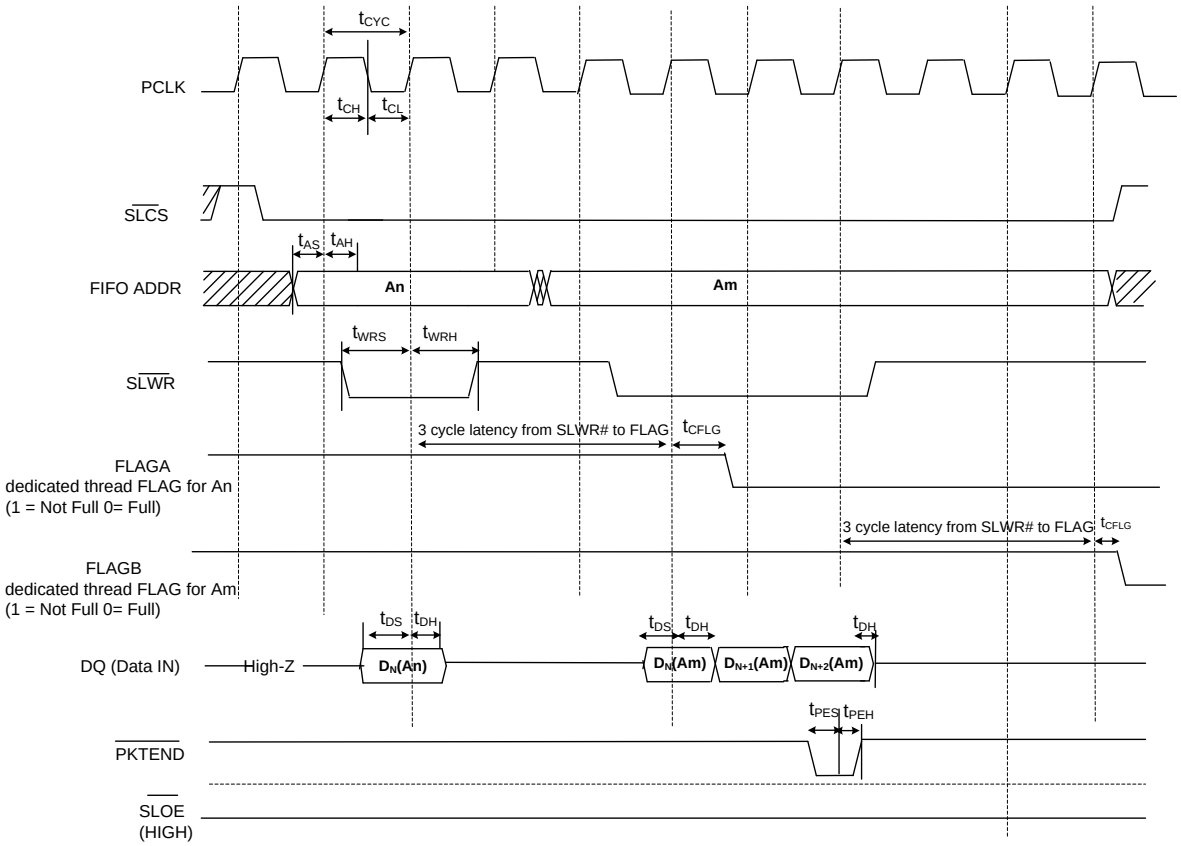
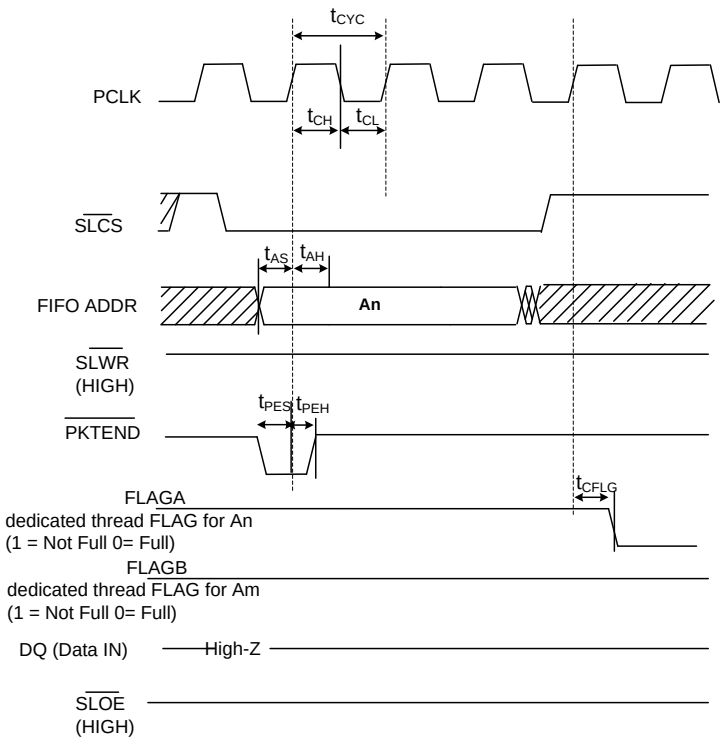


图 5. 同步 ZLP 写周期时序




## 4.3 同步从设备 FIFO 写序列

对同步从设备 FIFO 接口进行写操作的序列如下：

1. FIFO 地址稳定且 SLCS#信号被激活。
2. 外部主设备/外设将数据输出到数据总线上。
3. SLWR#被激活。
4. 当 SLWR#被激活时，数据将被写入到 FIFO 内，并且，FIFO 指针将在 PCLK 的上升沿上递增。
5. 从时钟的上升沿起，经过 $t_{CFLG}$ 的延迟后，FIFO 标志将被更新。

突发写入时将发生相同的事件序列。

**注意：**在突发模式下，SLWR#和 SLCS#在整个突发写入过程中保持激活状态。 在突发写入模式下，SLWR#被激活后，每当 PCLK 的上升沿到来时，都会将数据总线上的值写入到 FIFO 内。此外，FIFO 指针也在 PCLK 的每个上升沿上得到更新。

**短数据包：**通过 PKTEND#信号可将某个短数据包发送到 USB 主机。需要设计外部器件/处理器，使之在传输最后数据字时同时激活与该字相应的 SLWR#脉冲和 PKTEND#。PKTEND#激活期间，FIFOADDR 需要保持不变。同时激活 PKTEND#和 SLWR#时，GPIF II 状态机会将数据包视为短数据包，并将其发送到 USB 接口。如果协议不要求传输任何短数据包，则 PKTEND#信号可被置高。

请注意，执行读操作时，没有任何具体信号表示已从 USB 获取了短数据包。空标志必须由外部主设备监控，以确定读取完所有数据的时间。

**零长度数据包：**外部器件/处理器可仅通过激活 PKTEND#，而没有激活 SLWR#来传输一个零长度数据包（ZLP）。必须驱动 SLCS#和地址，如图 5 所示。

**标志使用情况：**外部处理器监控标志信号来控制流量。标志信号由 EZ-USB FX3 器件输出。通过配置各标志，可显示专用线程或当前寻址线程的空/满/局部状态。

表 3. 同步从设备 FIFO 时序参数

| 参数 | 说明 | 最小值 | 最大值 | 单位 |
|---|---|---|---|---|
| FREQ | 接口时钟频率 | – | 100 | MHz |
| $t_{CYC}$ | 时钟周期 | 10 | – | ns |
| $t_{CH}$ | 时钟为高电平的时间 | 4 | – | ns |
| $t_{CL}$ | 时钟为低电平的时间 | 4 | – | ns |
| $t_{RDS}$ | SLRD#至 CLK 的建立时间 | 2 | – | ns |
| $t_{RDH}$ | SLRD#至 CLK 的数据保持时间 | 0.5 | – | ns |
| $t_{WRS}$ | SLWR#至 CLK 的建立时间 | 2 | – | ns |
| $t_{WRH}$ | SLWR#至 CLK 的数据保持时间 | 0.5 | – | ns |
| $t_{CO}$ | 时钟至数据有效的时间 | – | 7 | ns |
| $t_{DS}$ | 数据输入的建立时间 | 2 | – | ns |
| $t_{DH}$ | 时钟上升沿后数据输入保持的时间 | 0 | – | ns |
| $t_{AS}$ | 地址到时钟的建立时间 | 2 | – | ns |
| $t_{AH}$ | CLK 至地址的地址保持时间 | 0.5 | – | ns |
| $t_{OELZ}$ | SLOE#到数据为低阻态的时间 | 0 | – | ns |
| $t_{CFLG}$ | 时钟到标志输出的传输延迟 | – | 8 | ns |
| $t_{OEZ}$ | SLOE#取消激活到数据为高阻态的时间 | – | 8 | ns |
| $t_{PES}$ | PKTEND#到 CLK 的建立时间 | 2 | – | ns |
| $t_{PEH}$ | CLK 上升沿后 PKTEND#保持的时间 | 0.5 | – | ns |
| $t_{CDH}$ | CLK 上升沿后数据输出保持的时间 | 2 | – | ns |
| **注意：**从 ADDR 到 DATA 的三个周期延迟 | | | | |

下面各部分介绍了使用 GPIF II Designer 和 EZ-USB FX3 SDK 对标志信号进行的配置。描述各种标志配置之前，首先需要理解线程、套接字以及 DMA 通道的概念。

## 5. 线程和套接字

本节简要说明了对 FX3 进行数据传入/传出所需的概念。

- 套接字
- DMA 描述符
- DMA 缓冲区
- GPIF 线程

套接字是外设硬件模块和 FX3 RAM 之间的连接点。FX3 上的每个外设硬件模块（如 USB、GPIF、UART 和 SPI）具有与其相关的固定套接字数量。流出外设的独立数据数量等于此外设上的套接字数量。使用一组寄存器来实现套接字，这些寄存器指向有效的 DMA 描述符，并使能或标记与套接字相关的中断。

DMA 描述符是一组位于 FX3 RAM 中的寄存器。它存储了 DMA 缓冲区的地址和大小数据，以及指向下一个 DMA 描述符的指针。这些指针构建成 DMA 描述符链。

DMA 缓冲区是 RAM 的一部，用于暂时存储通过 FX3 器件传输的数据。通过 FX3 固件，可将部分 RAM 空间作为 DMA 缓冲区使用。这些缓冲区的地址被存储为 DMA 描述符的一部分。

GPIF 线程是 GPIF II 模块内的专用数据路径，用来将外部数据引脚连接至套接字。套接字可通过某些事件直接互相发出信号，或者通过中断向 FX3 CPU 发出信号。该操作由固件配置的。例如，将数据流从 GPIF II 模块传输给 USB 模块。GPIF 套接字可以通知 USB 套接字它已经向 DMA 缓冲区写满了数据，或者 USB 套接字可以通知 GPIF 套接字该 DMA 缓冲区目前为空。该操作被称为自动 DMA 通道。当 FX3 CPU 不用修改数据流中的任何数据时，将实现自动 DMA 通道。

另外，GPIF 套接字可向 FX3 CPU 发送一个中断，来通知 GPIF 套接字已经写满了 DMA 缓冲区。FX3 CPU 可将该信息传递给 USB 套接字。USB 套接字可向 FX3 CPU 发送一个中断，来通知 USB 套接字已经读空了 DMA 缓冲区。此时，FX3 CPU 可将该信息反馈给 GPIF 套接字。该操作被称为手动 DMA 通道实现。如果 FX3 CPU 需要添加、删除或修改数据流中的数据，将需要执行该操作。

发送套接字是指用于将数据写入 DMA 缓冲区内的套接字。接收套接字是指从 DMA 缓冲区内读取数据的套接字。套接字使用存储于 DMA 描述符上的 DMA 缓冲区地址、DMA 缓冲区大小和 DMA 描述符链的值来管理数据。套接字写满或读空 DMA 缓冲区后，需要经过一段时间（几微秒）从一个 DMA 描述符转移到另一个描述符。转换过程中，套接字不能传输数据。

EZ-USB FX3 可提供多达四个物理硬件线程，以用于通过 GPIF II 传输数据。每次可将任何一个套接字映射到一个物理线程上。默认情况下，PIB 套接字 0 被映射到线程 0，PIB 套接字 1 被映射到线程 1，PIB 套接字 2 被映射到线程 2 以及 PIB 套接字 3 被映射到线程 3。

请注意，接口上的地址信号 A1:A0 表示需要访问的线程。FX3 的 DMA 运行机制将数据传输到映射到该线程上的套接字。因此，当 A1:A0 = 0 时，将访问线程 0，并且，通过线程 0 传输的所有数据将被传输到套接字 0。同样地，当 A1:A0 = 1 时，将数据传入/传出套接字 1。

**注意**：从设备 FIFO 接口仅包含两个地址线，所以最多只能访问四个套接字。如要访问四个套接字以上，必须使用带五个地址线的从设备 FIFO 接口。更多信息，请参阅应用笔记 AN68829 — 用于 EZ-USB FX3 的从设备 FIFO 接口：5 位地址模式。

必须通过配置 DMA 通道指定需要访问的套接字。

**注意**：有关 FX3 线程和套接字的更多信息，请参考 EZ-USB FX3 技术参考手册 中的第 7.4.6 节。

## 6. DMA 通道配置

固件必须使用所需的发送套接字和接收套接字来配置 DMA 通道。

如果将数据从从设备 FIFO 接口传输到 USB 接口，那么“P”端口将为发送套接字，USB 为接收套接字，反之亦然。

因此，若要通过从设备 FIFO 接口实现双向数据传输，则需配置两个 DMA 通道，其中一个带有作为发送套接字的“P”端口，另一个带有作为接收套接字的“P”端口。

作为发送套接字的“P”端口是外部器件通过从设备 FIFO 接口对其进行写入的套接字；作为接收套接字的“P”端口是外部器件通过从设备 FIFO 接口对其进行读取的套接字。

DMA 通道上的“P”端口套接字编号要与 A1:A0 上所寻址的套接字编号相同。

配置 DMA 通道时，可将多个缓冲区分配给一个特定的 DMA 通道。请注意，标志将分别表示每个缓冲区的满/空状态（单个缓冲区的最大尺寸为 64 KB-16）。

例如，如果已将两个大小均为 1024 字节的缓冲区分配给 DMA 通道，则当将 1024 字节写入到第一个缓冲区后，满标志将表示“满”状态。它持续显示满状态，直至 DMA 通道转到第二个缓冲区为止。虽然 DMA 通道转到下一个缓冲区所需的时长通常为几微秒，但未能确定该时长。外部主设备必须监控标志来确定完成切换操作以及可用下一个缓冲区来访问数据的时间。

下一节将说明如何配置标志，以表示不同线程的状态。

# 7. 标志配置

标志可被配置为空、满、一部分空或一部分满信号。这些信号并非由 GPIF II 状态机控制，而是由 EZ-USB FX3 内部的 DMA 硬件引擎控制。标志与特定线程或当前寻址的线程相关联，用于表示映射到该线程的套接字状态。

标志将根据（初始化套接字过程中所配置的）套接字方向来指示空或满状态。因此，如果从套接字读取数据，标志将表示空/未空状态；如果对套接字写入数据，标志将表示满/未满状态。

可用的标志类型分别为：

- 专用线程标志（空/满或一部分空/一部分满）
- 当前线程标志（空/满或一部分空/一部分满）

下面各节将介绍不同的标志类型。不同的标志配置会导致不同的延迟，如表 4 中所述。

## 7.1 专用线程标志

可对标志进行配置，以指示特定线程的状态。在这种情况下，无论在地址总线上寻址哪个线程，该标志将专门用于该线程，且始终只表示映射到该特定线程的套接字的状态。

此时，外部处理器/器件必须跟踪专用于各线程的标志，并且每当寻址到不同线程时需要保持监控正确标志。

例如，如果 FLAGA 专用于线程 0，FLAGB 专用于线程 1，当外部处理器对线程 0 进行访问时，必须监控 FLAGA。当外部处理器访问线程 1 时，则必须监控 FLAGB。

标志可专用于将要访问的某个线程。如果应用要求访问四个线程，将会使用四个相应的标志。

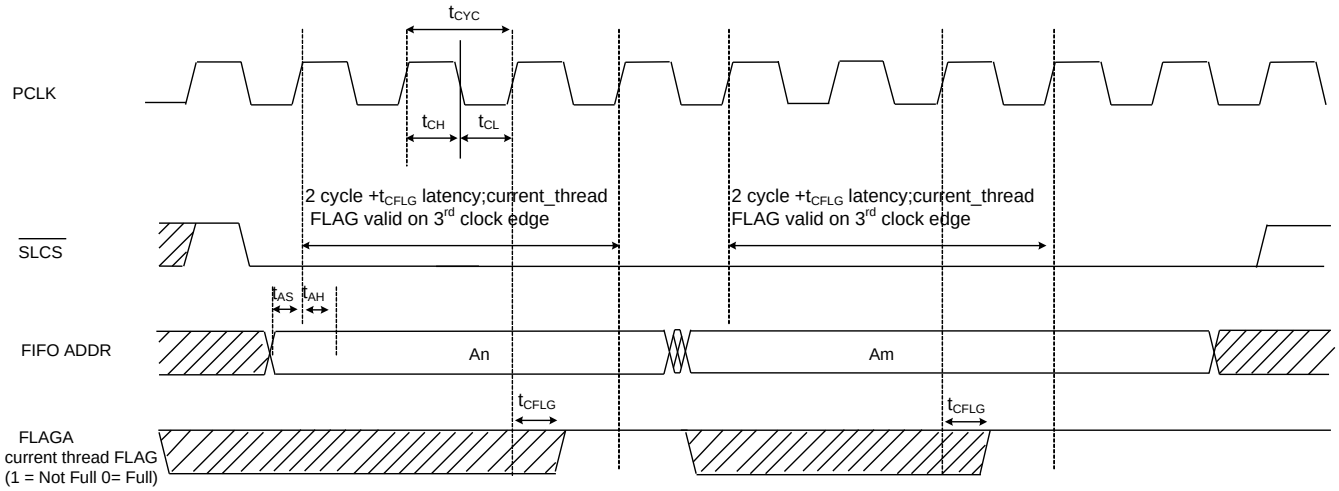
请注意，写传输过程结束时，标志始终经过三个周期的延迟。这三个周期的延迟是从使缓冲区变为满的写周期到标志被置低的时间。在第四个时钟沿上，外部主设备可以对处于低电平的标志进行采样，如图 4 所示。

读传输过程结束时，标志始终经过两个周期的延迟。两个周期的延迟是从使缓冲区变为空的读周期（ 最后一次激活 SLRD#）到标志被置低的时间。在第三个时钟沿上，外部主设备可以对处于低电平的标志进行采样，如图 3 所示。

## 7.2 当前线程标志

可对标志进行配置，来指示当前寻址的线程状态。在这种情况下，GPIF II 状态机对地址总线上的地址进行采样，然后更新相应标志，用于指示该线程的状态。这种配置要求的引脚数量更少，因为单个“current_thread” 标志可用来表示四个线程的状态。但当同步从设备 FIFO 接口使用 current_thread 标志时，将发生两个周期的延迟，因为 GPIF II 先要对地址进行采样，然后再更新该标志。两个周期的延迟是从有效地址被传输到接口算起。该事件发生后，在第三个时钟沿上，可以对刚寻址线程的标志的有效状态进行采样（请注意，SDK 中的从设备 FIFO 描述符使用 “current_thread” 标志配置）。

图 6. 使用当前线程标志情况下开始传输时所发生的额外延迟




**注意**：写传输过程结束时，标志始终经过三个周期的延迟。这三个周期的延迟是从使缓冲区变为满的写周期到标志被置低的时间。在第四个时钟沿上，外部主设备可以对处于低电平的标志进行采样，如图 4 所示。

读传输过程结束时，标志始终经过两个周期的延迟。两个周期的延迟是从使缓冲区变为空的读周期（最后一次激活 SLRD#）到标志被置低的时间。在第三个时钟沿上，外部主设备可以对处于低电平的标志进行采样，如图 3 所示。

### 7.2.1 局部标志

可对标志进行配置，以指示套接字的一部分空/一部分满状态。必须选择满足下面条件的水印值：当将要读/写的 32 位字数低于或等于该水印值时，将激活标志。

**注意** ：局部标志的延迟取决于该标志的特定水印值。

表 4 汇总了各种标志配置的相应延迟。该表还显示了 GPIF II Designer 中必须为特定标志选择的设置。有关标志的 GPIFII Designer 设置的示例和截图，请查看标志配置一节。

表 4. 不同标志配置的相关延迟

| 标志配置 | GPIF II Designer 标志设置选择 | 传输开始时从地址传输到标志的延迟 | 传输结束时的标志延迟 | | 所需的附加 API 调用 |
|---|---|---|---|---|---|
| | | | 对从设备 FIFO 进行的写操作（从最后一次激活 SLWR#到激活满标志的延迟） | 对从设备 FIFO 进行的读操作（从最后一次激活 SLRD#到激活空标志的延迟） | |
| 专用于特定线程“n”的满/空标志 | Thread_n_DMA_Ready | 0 周期 | 3 个周期 + $t_{CFLG}$（外部器件可在第 4 个时钟沿上对有效标志进行采样） | 2 个周期 + $t_{CFLG}$（外部器件可在第 3 个时钟沿上对有效标志进行采样） | N/A |
| 用于当前寻址线程的满/空标志 | Current_thread_DMA_Ready | 2 个周期 +$t_{CFLG}$（外部器件可在第 3 个时钟沿上对有效标志进行采样） | 3 个周期 + $t_{CFLG}$（外部器件可在第 4 个时钟沿上对有效标志进行采样） | 2 个周期 + $t_{CFLG}$（外部器件可在第 3 个时钟沿上对有效标志进行采样） | N/A |
| 专用于特定线程“n”的一部分满/一部分空标志 | Thread_n_DMA_Watermark | 0 周期 | 取决于水印等级 | 取决于水印等级 | 通过调用 CyU3PGpifSocketConfigure() API 来设置水印等级。<br>**注意：**水印的单位为 32 位数据字。<br>示例：<br>CyU3PGpifSocketConfigure (0,PIB_SOCKET_0,4,CyFalse,1) 设置了线程 0 到 4 的水印值<br>CyU3PGpifSocketConfigure (3,PIB_SOCKET_3,4,CyFalse,1) 设置了线程 3 到 4 的水印值 |
| 用于当前寻址线程的一部分满/一部分空标志 | Current_thread_DMA_Watermark | 2 个周期 +$t_{CFLG}$（外部器件可在第 3 个时钟沿上对有效标志进行采样） | 取决于水印等级 | 取决于水印等级 | 通过调用 CyU3PGpifSocketConfigure() API 来设置水印等级。<br>**注意：**水印的单位为 32 位数据字。<br>示例：<br>CyU3PGpifSocketConfigure (0,PIB_SOCKET_0,4,CyFalse,1) 设置了线程 0 到 4 的水印值<br>CyU3PGpifSocketConfigure (3,PIB_SOCKET_3,4,CyFalse,1) 设置了线程 3 到 4 的水印值 |

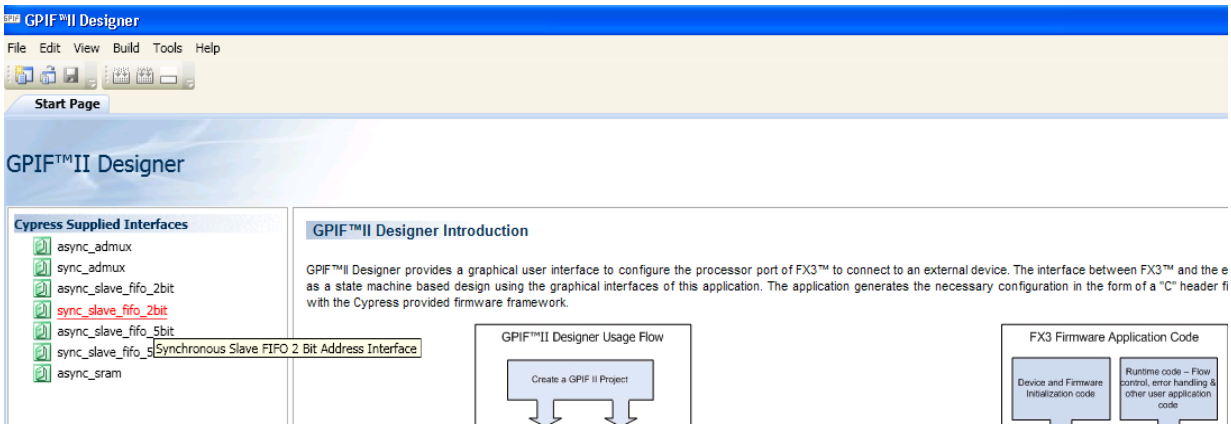
下一节将介绍如何使用 GPIF II Designer 工具和 EZ-USB FX3 SDK 来配置标志。

# 8. GPIF II Designer

## 8.1 实现同步从设备 FIFO 接口

通过安装 GPIF II Designer 工具，您可以看到从设备 FIFO 接口的 GPIF II 实现指导。请从赛普拉斯网站 www.cypress.com 上安装 GPIF II Designer 工具。当启动 GPIF II Designer 时，您可以在起始页上寻找 **Cypress Supplied Interfaces**（赛普拉斯提供的接口）项。

图 7. GPIF II Designer 工具内的从设备 FIFO 项目 — Cypress Supplied Interfaces




sync_slave_fifo_2bit 项目是使用 2 位地址进行同步从设备 FIFO 接口的 GPIF II 实现。下一节将说明如何使用 GPIF II Designer 配置局部标志。

## 8.2 配置局部标志

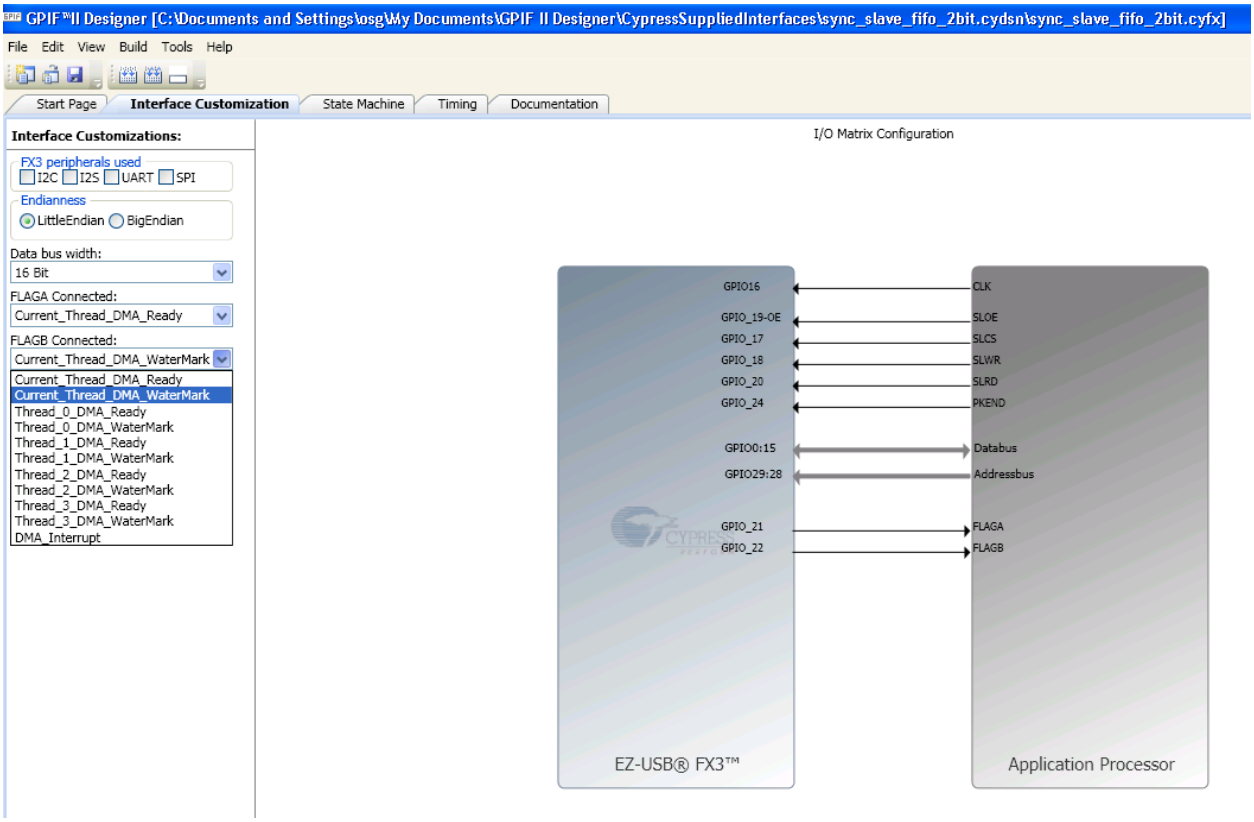
按照下面各个步骤对局部标志进行配置：

- 必须在 GPIF II Designer 工具中选中局部标志设置。
- 必须使用固件中的 CyU3PGpifSocketConfigure() API 设置局部标志的水印等级

在 GPIF II Designer 工具中，打开 Cypress Supplied Interfaces 下方的 **sync_slave_fifo_2bit** 项目，然后选中“FLAGA Connected”或“FLAGB Connected”下的 **Current_Thread_DMA_Watermark**，以将标志配置为当前线程的局部标志。

或者，选中 **Thread_n_DMA_Watermark**，以将标志配置为线程“n”的专用局部标志。

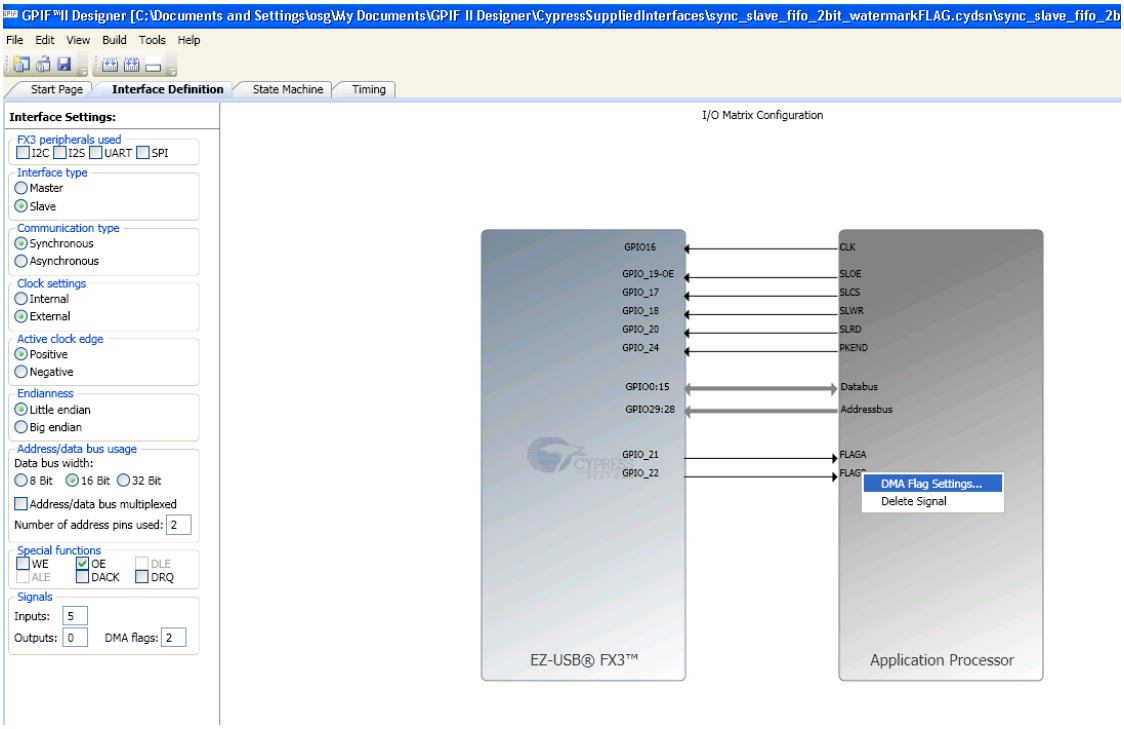
图 8. GPIF II Designer 中的标志设置 — Cypress Supplied Interfaces 项下的 sync_slave_fifo_2bit




除了 *sync_slave_fifo_2bit.cyfx* 项目中所提供的设置之外，若想要添加更多标志或进行更多修改，请依次选择 **File** > **Save Project as Editable**。这样，可以将该项目保存为新一个名称，从而对新保存的项目进行更改。

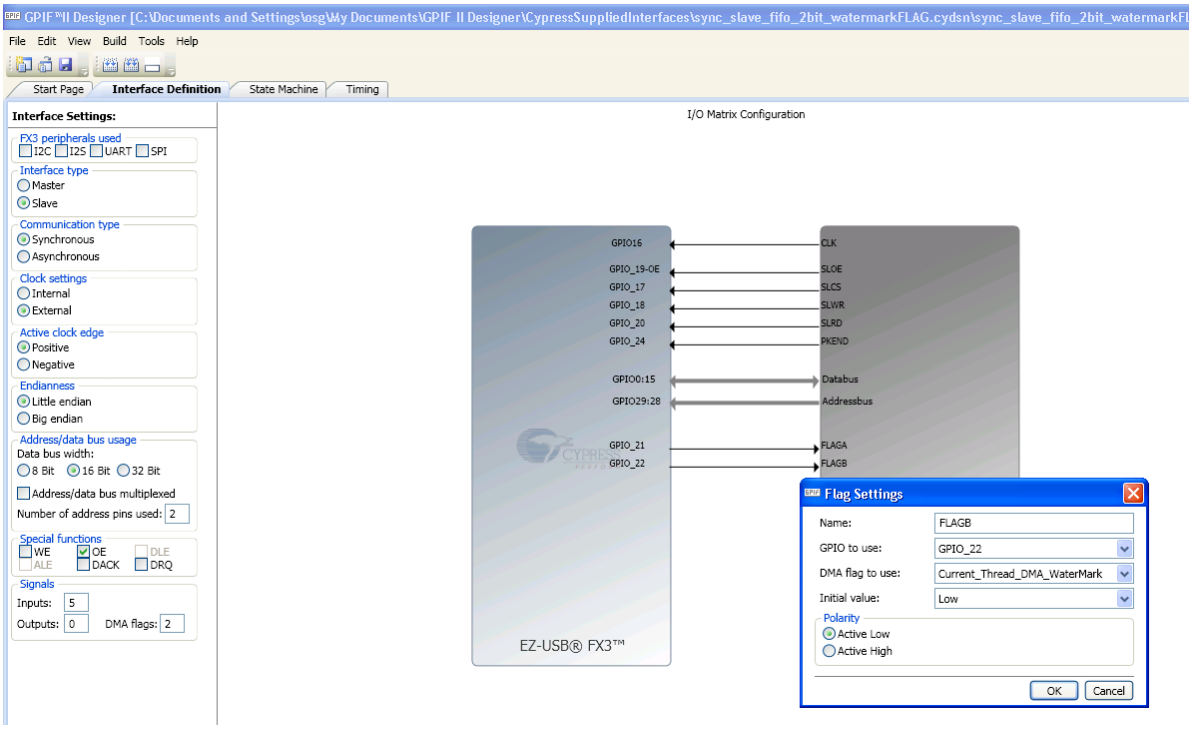
在这种情况下，如要将某个标志配置为局部标志，请右击 I/O 矩阵配置框图中的标志。点击 **DMA Flag Settings**，如下图所示。

图 9. 通过默认 sync_slave_fifo_2bit 中的“Save Project as Editable”功能，设置新项目中的标志




下图显示了如何选择标志配置：

图 10. 选择特定的标志设置



配置局部标志的第二步是指定固件项目中的标志水印值。在 *cyfxslfifo.c* 文件中，添加一个 CyU3PGpifSocketConfigure() API 调用，以指定水印值。在调用 CyU3PGPIFLoad() API 后可立即调用该 API。有关 CyU3PGpifSocketConfigure() API 的完整说明，请参阅 EZ-USB FX3 SDK API 指南。水印值是输入该 API 的参数之一。

水印值决定何时激活局部标志。下面一节介绍了用于计算激活局部标志后将要读/写的数据字数量的公式。

## 8.3 使用局部标志情况下的通用公式

前一节描述了标志的可能配置以及配置局部标志的各步骤。本节说明如何确定某个局部标志的水印值。

激活局部标志后，应使用下面公式来计算将要读/写的数据字数量。

**注意：**在 CyU3PGpifSocketConfigure() API 中指定的水印值的单位为 32 位数据字。

1. 当外部主设备对同步从设备 FIFO 进行写入时：

   (a) 时钟沿到来（这时，对处于低电平的局部标志进行采样）后将要写入的数据字的数量 = 水印值 x （32/总线宽度） – 4

2. 当外部主设备对同步从设备 FIFO 进行读取时:

   (a) 时钟沿到来（这时，对处于激活状态的局部标志进行采样）后可用于读取的数据字的数量（SLOE#维持激活时） = 水印值 x （32/总线宽度） – 1

   (b) 已包含从 SLRD#置低到数据在总线上有效的两个周期的延迟。 因此，时钟沿到来（这时，对处于激活状态的局部标志进行采样）后维持激活 SLRD#的周期数 = 水印值 x （32/总线宽度）– 3。

## 8.4 CyU3PGpifSocketConfigure() API 使用示例

本节介绍了使用 CyU3PGpifSocketConfigure() API 指定的水印值的影响示例。还提供了截图，以明确显示使用不同水印值时的局部标志性能。

**注意 ：**在这些示例中，标志的极性被设为低电平有效。因此，当标志为低电平时，它们指示满/空或一部分满/一部分空状态。

### 8.4.1 示例 1

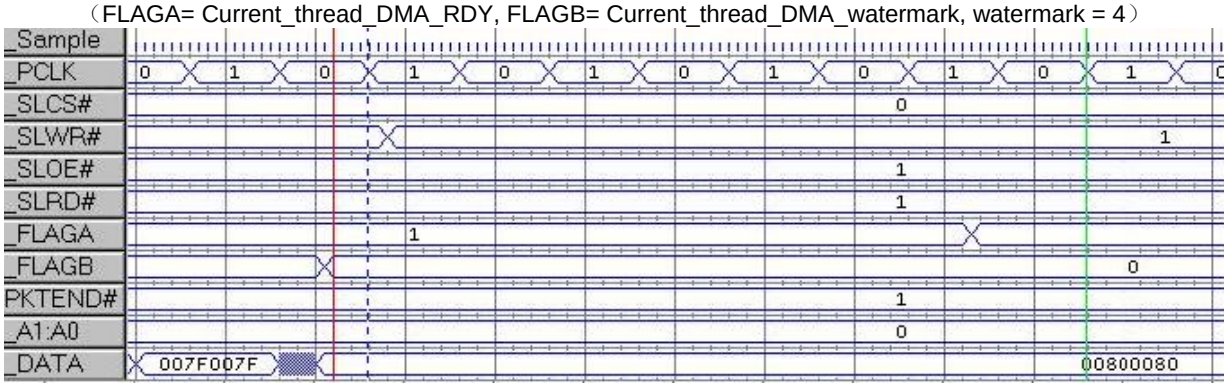
使用 32 位数据总线的从设备 FIFO：

- 在 GPIF II Designer 中，FLAGA 被配置为 Current_thread_DMA_RDY，FLAGB 被配置为 Current_thread_DMA_watermark。
- CyU3PGpifSocketConfigure (0,PIB_SOCKET_0,4,CyFalse,1)
- 外部 FPGA 通过从设备 FIFO 对 EZ-USB FX3 进行突发写入传输（最后写入的数据为 0x00800080）。

下面显示了标志在传输结束时变为 0 的逻辑分析器截图。可见，FLAGB（局部标志）在写入最后数据字的同一个周期内转为低电平。

图 11. 使用 32 位数据总线宽度的突发写入传输

（FLAGA= Current_thread_DMA_RDY, FLAGB= Current_thread_DMA_watermark, watermark = 4）



### 8.4.2 示例 2

使用 32 位数据总线的从设备 FIFO：

- 在 GPIF II Designer 中，FLAGA 被配置为 Current_thread_DMA_RDY，FLAGB 被配置为 Current_thread_DMA_watermark。
- CyU3PGpifSocketConfigure (3, PIB_SOCKET_3, 4, CyFalse, 1)
- 外部 FPGA 通过从设备 FIFO 对 EZ-USB FX3 进行突发读取传输（将要读取的最后数据为 0x00000080）。

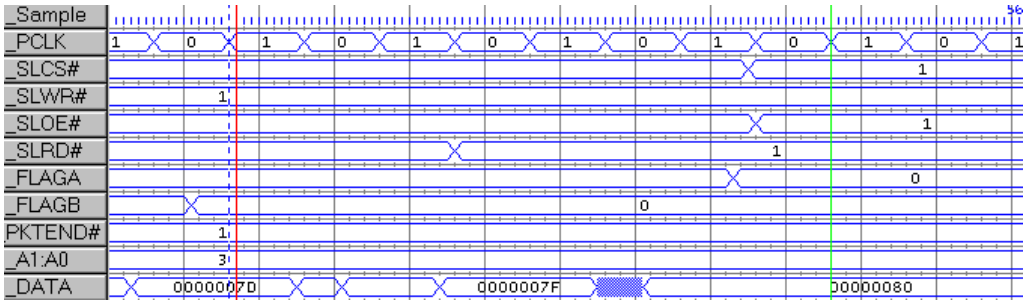
下面显示了标志在传输结束时变为 0 的逻辑分析器截图。可见，在读取最后数据前四个周期，FLAGB（局部标志）转为低电平。这样，在 FLAGB 转为低电平的周期后，可以读取三个数据字。

在该示例中，可使用使用局部标志情况下的通用公式部分中的公式 2(a)，具体如下：

水印值 = 4，总线宽度 = 32

因此，时钟沿到来（这时，对处于激活状态的局部标志进行采样）后可用于读取的 32 位数据字的数量 = 4 x (32/32) – 1 = 3

图 12. 使用 32 位数据总线宽度的突发读取传输
（FLAGA= Current_thread_DMA_RDY，FLAGB= Current_thread_DMA_watermark，watermark = 4）




### 8.4.3 示例 3

使用 16 位数据总线的从设备 FIFO：

- 在 GPIF II Designer 中，FLAGA 被配置为 Current_thread_DMA_RDY，FLAGB 被配置为 Current_thread_DMA_watermark。
- CyU3PGpifSocketConfigure (0, PIB_SOCKET_3, 3, CyFalse, 1)
- 外部 FPGA 通过从设备 FIFO 对 EZ-USB FX3 进行突发写入传输（最后写入的数据为 0x0200）。

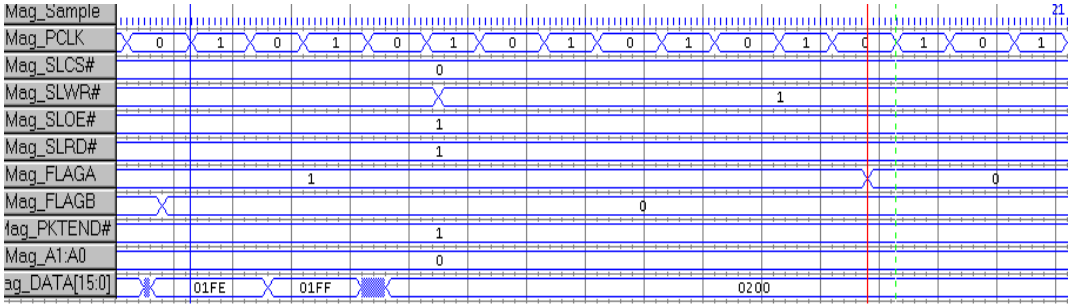
下面显示了标志在传输结束时变为 0 的逻辑分析器截图。可见，在写入最后数据前的三个周期内，FLAGB（局部标志）处于低电平。这样，在 FLAGB 转为低电平的周期后，可以读取两个数据字。

在该示例中，可以使用使用局部标志情况下的通用公式部分中的公式 1，具体如下：

水印值 = 3，总线宽度 = 16

因此，时钟沿（所采样的局部标志为低电平时）后可写入的 16 位数据字数量 = 3 x (32/16) – 4 = 2

图 13. 使用 16 位数据总线宽度的突发写入传输
（FLAGA= Current_thread_DMA_RDY，FLAGB= Current_thread_DMA_watermark，watermark = 3）



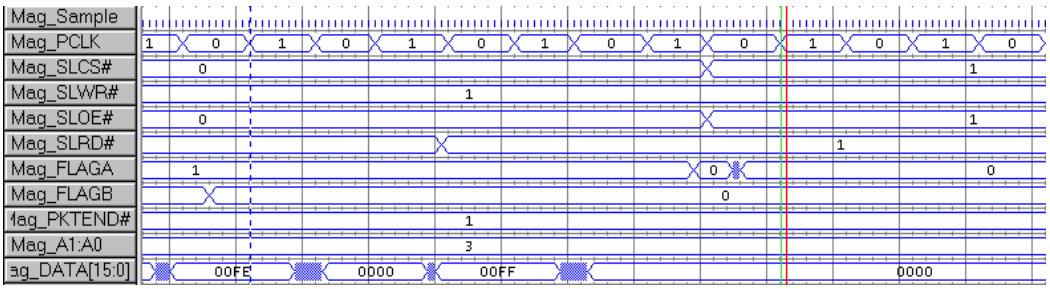

### 8.4.4 示例 4

使用 16 位数据总线的从设备 FIFO：

- 在 GPIF II Designer 中，FLAGA 被配置为 Current_thread_DMA_RDY，FLAGB 被配置为 Current_thread_DMA_watermark。
- CyU3PGpifSocketConfigure (3, PIB_SOCKET_3, 2, CyFalse, 1)
- 外部 FPGA 通过从设备 FIFO 对 EZ-USB FX3 进行突发读取传输（最后写入的数据为 0x0000）。

下面显示了标志在传输结束时变为 0 的逻辑分析器截图。可见，在读取最后数据前的四个周期内，FLAGB（局部标志）处于低电平。这样，在 FLAGB 转为低电平的周期后，可以读取三个数据字。

图 14. 使用 16 位数据总线宽度的突发读取传输
（FLAGA= Current_thread_DMA_RDY，FLAGB= Current_thread_DMA_watermark，watermark = 2）




## 8.5 使用局部标志时的其他注意事项：

- 局部标志仅用于决定传输结束的时间。必须在开始传输时监控满/空标志，以确保套接字的可用性。这意味着，不能单独使用局部标志，而必须和满/空标志结合使用。
- 如果外部主设备能够执行一个计数机制，并始终写入与 EZ-USB FX3 的 DMA 缓冲区大小相同的数据量，就能完全避免使用局部标志。外部主设备应计算将要写入或读取的数据量，并保证该计数值不超过创建 DMA 通道时所设置的缓冲区大小。在这种情况下，应通过监控满/空标志来决定开始传输的时间。
- 如果尚未执行上面介绍的计数机制，并且使用了一个局部标志，请按照下面各步骤的其中一个进行操作：
  - 如果外部主设备始终使用固定的数据量进行突发传输，那么，必须在选择水印值时考虑到该突发的大小。在 CyU3PGpifSocketConfigure() API 中设置突发值，该值是此 API 中的最后参数。突发值被设置后，DMA 硬件将在每个突发传输后（而不是每个字传输后）检测水印值。例如，如果外部主设备始终进行 8 个字的突发写入，则必须设置水印值，使之满足下面条件：当 EZ-USB FX3 的 DMA 缓冲区中有 8 字突发的空间，标志将转为低电平。在局部标志转为低电平后，外部主设备可写入一个完整的 8 字突发。对于在 16 位模式下（对从设备 FIFO 进行）的写操作，请根据使用局部标志情况下的通用公式一节所介绍的公式，将 CyU3PGpifSocketConfigure() API 中的水印值设为‘6’。

- 前一步骤的备用步骤是，在局部标志变为零后，外部主设备不会执行突发访问，而切换到单周期访问模式。然后，在每个周期，外部主设备将在写入前先检查满/空标志，以确保缓冲区是可用的。

## 8.6 由标志违规引起的错误条件

当局部标志或满/空标志指示缓冲区尚未可用时，不能对从设备 FIFO 接口进行数据读/写访问。

如果读取空白的缓冲区，将发生缓冲区欠载错误。如果写入到已满的缓冲区，则发生缓冲区过载错误。这些错误可导致缓冲区边界的数据被损坏。由于从设备 FIFO 接口位于 FX3 的 PIB 模块域，因此，与接口相关的所有错误将被表示为 PIB 错误。

如果发生某个 PIB 错误，将触发 PIB 中断。FX3 SDK 允许寄存一个用于 PIB 中断的回调函数。该回调函数用于检查 PIB 错误代码。下面代码示例介绍了如何寄存一个回调函数以及用于检查欠载/过载错误并弹出一个调试信息的实际回调函数。

错误代码指示发生错误的线程，如表 5 所示。如果发生这些错误中的任意一个，赛普拉斯建议您使用一个逻辑分析器谨慎地分析接口时序，并特别注意局部标志变为零后所执行的读/写周期数量。有关的示例，请参考图 11 到图 14。

```
/* Register a callback for notification of PIB interrupts*/
CyU3PpibRegisterCallback(gpif_error_cb,intMask);

/* Callback function to check for PIB ERROR*/
void gpif_error_cb(CyU3PpibIntrType cbType, uint16_t cbArg)
{
       if(cbType==CYU3P_PIB_INTR_ERROR)
       {
              switch(CYU3P_GET_PIB_ERROR_TYPE(cbArg))
               {
                      case CYU3P_PIB_ERR_THR0_WR_OVERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR0_WR_OVERRUN");
                     break;
                     case CYU3P_PIB_ERR_THR1_WR_OVERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR1_WR_OVERRUN");
                     break;
                      case CYU3P_PIB_ERR_THR2_WR_OVERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR2_WR_OVERRUN");
                     break;
                     case CYU3P_PIB_ERR_THR3_WR_OVERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR3_WR_OVERRUN");
                     break;

                     case CYU3P_PIB_ERR_THR0_RD_UNDERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR0_RD_UNDERRUN");
                     break;
                     case CYU3P_PIB_ERR_THR1_RD_UNDERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR1_RD_UNDERRUN");
                     break;
                     case CYU3P_PIB_ERR_THR2_RD_UNDERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR2_RD_UNDERRUN");
                     break;
                     case CYU3P_PIB_ERR_THR3_RD_UNDERRUN:
                            CyU3PdebugPrint (4, "CYU3P_PIB_ERR_THR3_RD_UNDERRUN");
                     break;

                     default:
                            CyU3PdebugPrint (4, "No Underrun/Overrun Error");
                     break;
              }
        }
}
```

表 5. 过载/欠载条件的 PIB 错误代码

| PIB 错误代码 | 说明 |
|---|---|
| CYU3P_PIB_ERR_THR0_WR_OVERRUN | 线程 0 缓冲区中的写入过载 |
| CYU3P_PIB_ERR_THR1_WR_OVERRUN | 线程 1 缓冲区中的写入过载 |
| CYU3P_PIB_ERR_THR2_WR_OVERRUN | 线程 2 缓冲区中的写入过载 |
| CYU3P_PIB_ERR_THR3_WR_OVERRUN | 线程 3 缓冲区中的写入过载 |
| CYU3P_PIB_ERR_THR0_RD_UNDERRUN | 线程 0 缓冲区中的读取欠载 |
| CYU3P_PIB_ERR_THR1_RD_UNDERRUN | 线程 1 缓冲区中的读取欠载 |
| CYU3P_PIB_ERR_THR2_RD_UNDERRUN | 线程 2 缓冲区中的读取欠载 |
| CYU3P_PIB_ERR_THR3_RD_UNDERRUN | 线程 3 缓冲区中的读取欠载 |

# 9. SDK 内的从设备 FIFO 固件示例

本应用笔记介绍了同步从设备 FIFO 接口以及配置标志的方法。实现 GPIF II Designer 工具中的所需配置后，需要将已更新的配置集成到固件内。完成编译 GPIFII Designer 中的项目后，将生成一个头文件 *cyfxgpifconfig.h*。该头文件必须包含在固件项目中。EZ-USB FX3 SDK 包含一个集成了从设备 FIFO 接口的固件示例。

安装 EZ-USB FX3 SDK 后，可在下面目录中找到集成了同步从设备 FIFO 接口的固件示例：

*[FX3 SDK Install Path]\EZ-USB FX3 SDK\1.3\firmware\slavefifo_examples\slfifosync*

该固件示例支持 16 位和 32 位数据总线宽度。头文件 *cyfxslfifosync.h* 定义了常量 CY_FX_SLFIFO_GPIF_16_32BIT_CONF_SELECT。想要选择 32 位数据总线宽度，请将该常量设为 1；若想选择 16 位数据总线宽度，则将该常量设为 0。

**注意**：如果从设备 FIFO 在 100 MHz 的工作频率下使用 32 位数据总线运行，那么，必须将 PLL 频率配置为 400 MHz。
为实现该操作，要将 setSysClk400 参数设置为 CyU3PDeviceInit()函数的输入。更多信息，请参考 FX3 SDK 中的 API 指南。

# 10. 设计示例 1：将 Xilinx FPGA 连接至 FX3 同步从设备 FIFO 接口

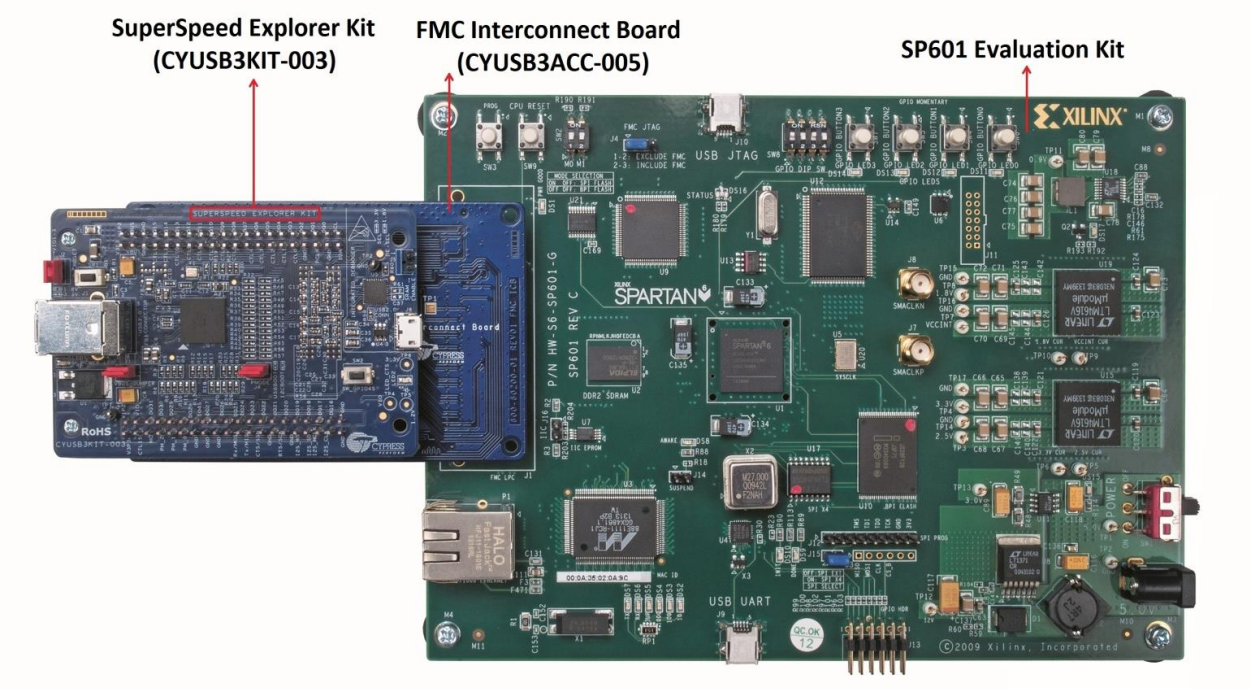
本节提供一个完整的设计示例，其中通过同步从设备 FIFO 接口将 Xilinx Spartan 6 FPGA 连接至 FX3。本节还描述了用于实现该设计的硬件、固件及软件组件。

## 10.1 硬件设置

可使用具有一个与 Xilinx Spartan 6 SP601 评估套件互联的赛普拉斯 FX3 开发套件（CYUSB3KIT-001）或 SuperSpeed Explorer 套件（CYUSB3KIT-003）的硬件设置来运行示例中的项目。FX3 电路板和 Xilinx 电路板通过 Samtec 至 FMC 互联电路板相连。CYUSB3ACC-002 互联电路板通过 Samtec 连接器与赛普拉斯 FX3 开发套件（CYUSB3KIT-001）相连，并通过 FMC 连接器与 Xilinx 电路板相连。

CYUSB3ACC-005 互联电路板通过多个插座与 SuperSpeed Explorer 套件（CYUSB3KIT-003）相连，并通过 FMC 连接器与 Xilinx 电路板相连。图 15 显示了使用 SuperSpeed Explorer 套件的硬件设置。更多信息，请参阅附录 B 用 FX3 开发套件（CYUSB3KIT-001）的硬件设置。除了硬件设置，以下各步骤在使用任何 FX3 电路的情况下均相同。

图 15. 赛普拉斯 SuperSpeed Explorer 套件通过 FMC 互联电路板与 Xilinx SP601 电路板相连



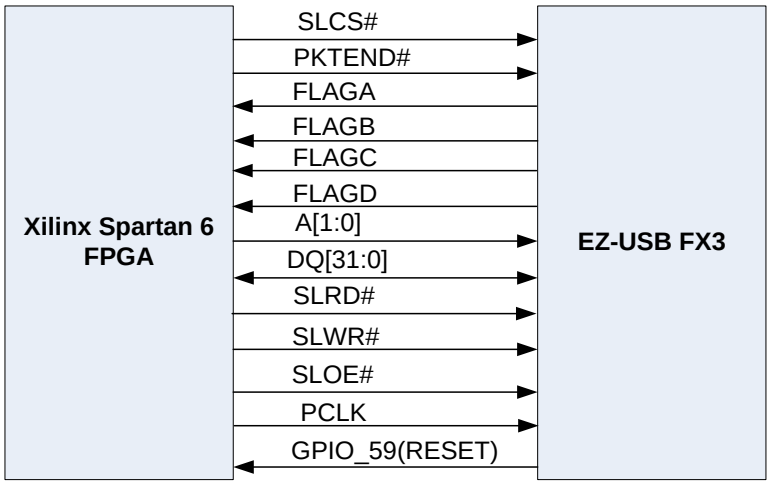

**注意：**要确保在 FPGA 通过从设备 FIFO 接口连接至 FX3 的所有应用中，SuperSpeed Explorer 套件上的跳线器 J5 均处于开路状态。

## 10.2 固件和软件组件

- FX3 同步从设备 FIFO 固件项目可从 FX3 SDK 获取
- Control Center（控制中心）和 Streamer（串流）软件工具可从 FX3 SDK 获取

下图显示了 FPGA 和 FX3 之间的互连图。

图 16. FPGA 和 FX3 之间的互连图




该示例包括以下组件：

- 回送传输：对于该组件，FPGA 先从 FX3 读取整个缓冲区的内容，然后将其回写到 FX3 内。USB 主机应发送 OUT/IN 令牌数据包，用于发送和接收该数据。您可使用 EZ-USB FX3 SDK 中提供的 Control Center 工具实现该操作。
- 短数据包：在该组件中，FPGA 先将一个完整的数据包传输到 FX3，然后再发送一个短数据包。USB 主机应发送 IN 令牌数据包，从而接收该数据。
- 零长度数据包（ZLP）传输：在该组件中，FPGA 先将一个完整的数据包传输到 FX3，然后再发送一个零长度数据包。USB 主机应发送 IN 令牌数据包，从而接收该数据。
- 串流输入（IN）数据传输：在该组件中，FPGA 实现单向传输，即是通过同步从设备 FIFO 连续将数据写入 FX3。USB 主机应发送 IN 令牌数据包，从而接收该数据。您可以使用 EZ-USB FX3 SDK 中提供的 Control Center 或 Streamer 工具实现该操作。
- 串流输出（OUT）数据传输：在该组件中，FPGA 实现单向传输，即是通过同步从设备 FIFO 从 FX3 连续读取数据。USB 主机应发送 OUT 令牌数据包，从而发送该数据。您可使用 EZ-USB FX3 SDK 中提供的 Control Center 或 Streamer 工具实现该操作。

## 10.3 FX3 固件的详细信息

FX3 固件是基于 FX3 SDK 中的示例项目。

该固件的主要性能为：

- 启用 USB 3.0 和 USB 2.0。
- 使用赛普拉斯 VID/PID（即 0x04B4/0x00F1）进行枚举。这样，可以使用赛普拉斯的 Control Center 和 Streamer 工具启动 USB 传输。
- 集成具有下面特性的同步从设备 FIFO 描述符：
  - 支持访问多达四个套接字
  - 将数据总线的宽度配置为 32 位
  - 在 100 MHz 的 PCLK 输入时钟频率下工作
  - 能够配置四个标志：

i. FLAGA：线程 0 专用的满标志

ii. FLAGB：线程 0 水印值为 6 的专用局部标志

iii. FLAGC：线程 3 专用的空标志

iv. FLAGD：线程 3 水印值为 6 的专用局部标志

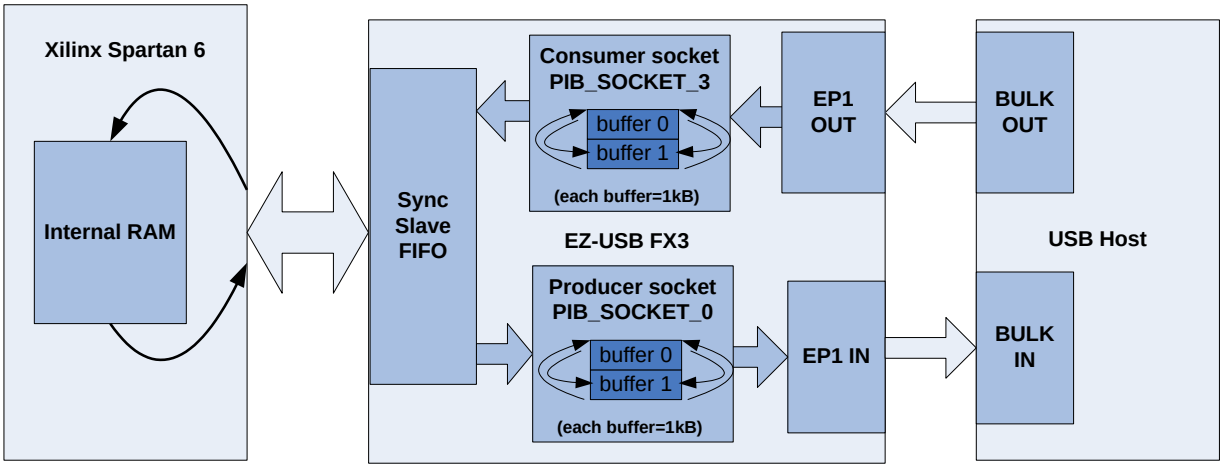
**注意：** 本应用笔记中提供的 GPIF II Designer 项目介绍了这些设置。此外，该固件项目还介绍了如何使用 CyU3PGpifSocketConfigure() API 来配置水印值。

- 将 PLL 频率配置为 400 MHz。为了实现该操作，要将 setSysClk400 参数设置为 CyU3PDeviceInit()函数的输入。

**注意：** 在工作频率为 100 MHz 且数据总线宽度为 32 位的条件下，该设置对从设备 FIFO 的运行起着关键作用。

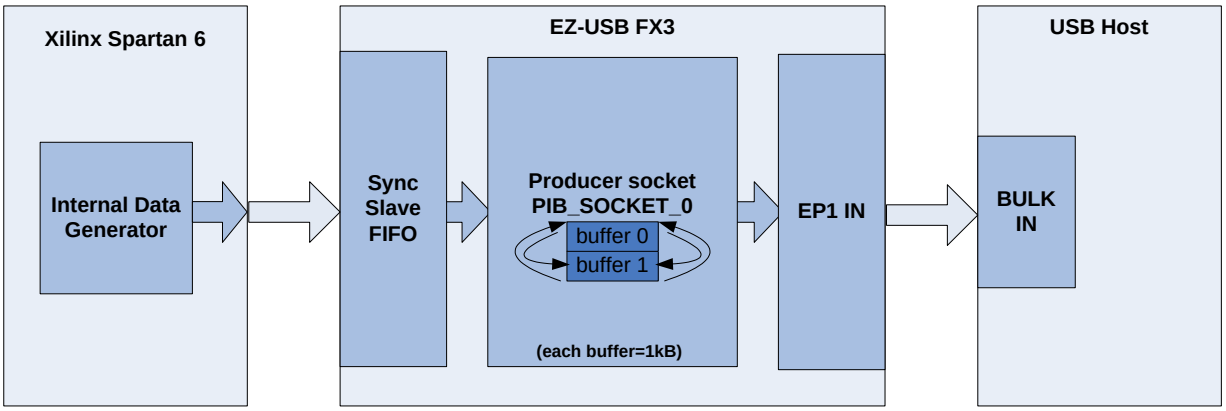
- 设置 DMA 通道：
  - 对于回送传输、短数据包和 ZLP 传输，需要创建两个 DMA 通道：
    - 一个是 P2U 通道，其中 PIB_SOCKET_0 作为发送套接字，UIB_SOCKET_1 作为接收套接字。DMA 缓冲区大小为 512 还是 1024 取决于 USB 连接是 USB 2.0 还是 USB 3.0。DMA 缓冲区的数量为 2。
    - 另一个是 U2P 通道，其中 PIB_SOCKET_3 作为接收套接字，UIB_SOCKET_1 作为发送套接字。DMA 缓冲区大小为 512 还是 1024 取决于 USB 连接是 USB 2.0 还是 USB 3.0。DMA 缓冲区数量为 2。

图 17. 回送传输的设置



**注意：** 只有 P2U 通道用于传输短数据包和 ZLP 数据包。

图 18. 短数据包和 ZLP 数据包传输的设置

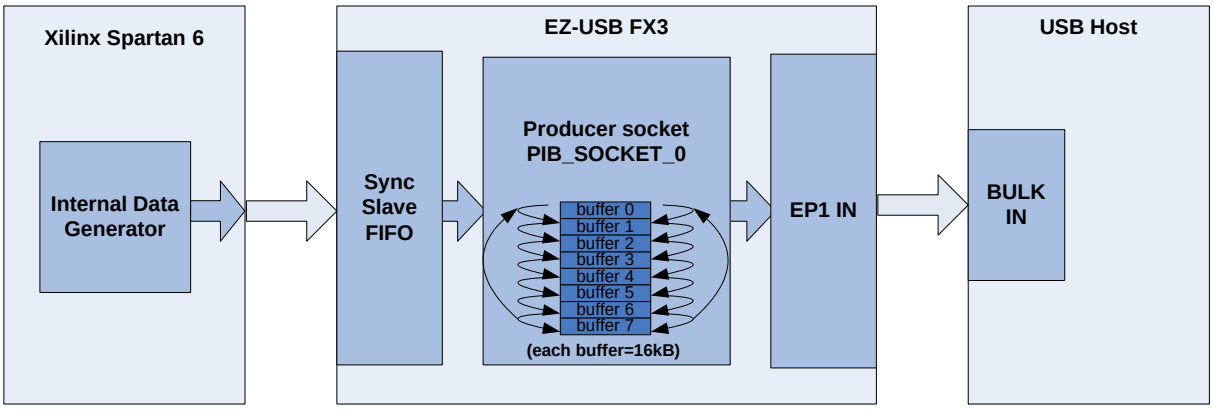



本节所描述的 DMA 通道仅在本应用笔记中提供的 FX3 固件项目内 *cyfxslfifosync.h* 文件中的下列定义被使能时设置。

```
/* set up DMA channel for loopback/short packet/ZLP transfers */
#define LOOPBACK_SHRT_ZLP
```

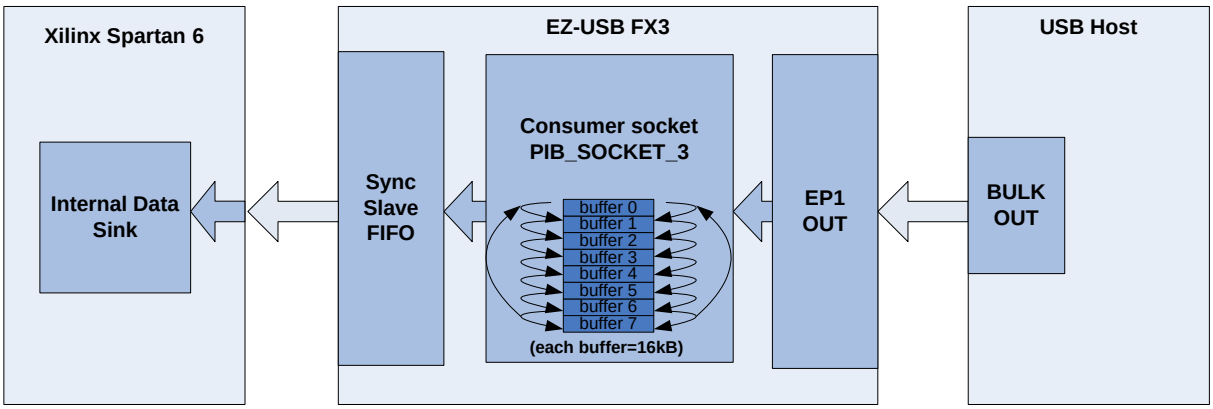
- 对于串流传输，需要创建两个 DMA 通道：
  - 一个是 P2U 通道，其中 PIB_SOCKET_0 作为发送套接字，UIB_SOCKET_1 作为接收套接字。根据所使用的 USB 连接类型，DMA 缓冲区大小将为 16×1024（USB 3.0 连接方式）或为 16×512（USB 2.0 连接方式）。DMA 缓冲区的数量为 8。选用该缓冲区大小和数量，以得到高吞吐量性能。

图 19. 串流 IN 传输设置 — 优化缓冲区数量和大小，以得到更好的性能




  - 另一个是 U2P 通道，其中 PIB_SOCKET_3 作为接收套接字，UIB_SOCKET_1 作为发送套接字。DMA 缓冲区大小为 16×1024（USB 3.0 连接方式）或为 16×512（USB 2.0 连接方式）。DMA 缓冲区的数量为 4。可通过增大缓冲区数量来增强性能，但需要降低 P2U 通道的缓冲区数量。其原因是 FX3 SDK 所提供的缓冲区存储空间不足以支持两个通道同时具有 8 个大小为 16×1024 的缓冲区。

图 20. 串流 OUT 传输设置 — 优化缓冲区数量和大小，以得到更好的性能




本节所描述的 DMA 通道仅在本应用笔记中提供的 FX3 固件项目内 *cyfxslfifosync.h* 文件中的下列定义被使能时设置。

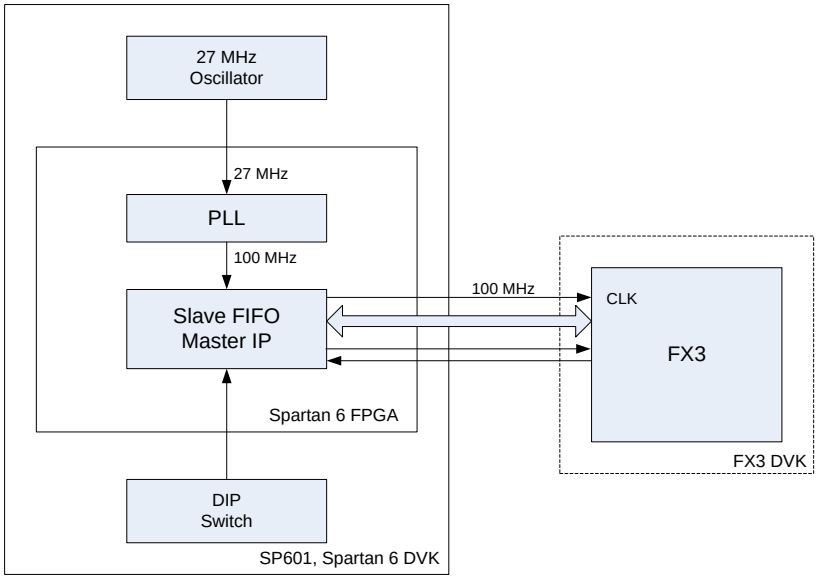
```
/* set up DMA channel for stream IN/OUT transfers */
#define STREAM_IN_OUT
```

通过使用下面各定义（*cyfxslfifosync.h* 文件中也包含这些定义），可以控制分配给 P2U 和 U2P DMA 通道的缓冲区数量。

```
/* Slave FIFO P_2_U channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_P_2_U      (4)
/* Slave FIFO U_2_P channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_U_2_P      (8)
```

## 10.4 FPGA 实现的详细信息

图 21. 使用 SP601 评估套件实现 Xilinx Spartan 6（XC6SLX16）FPGA



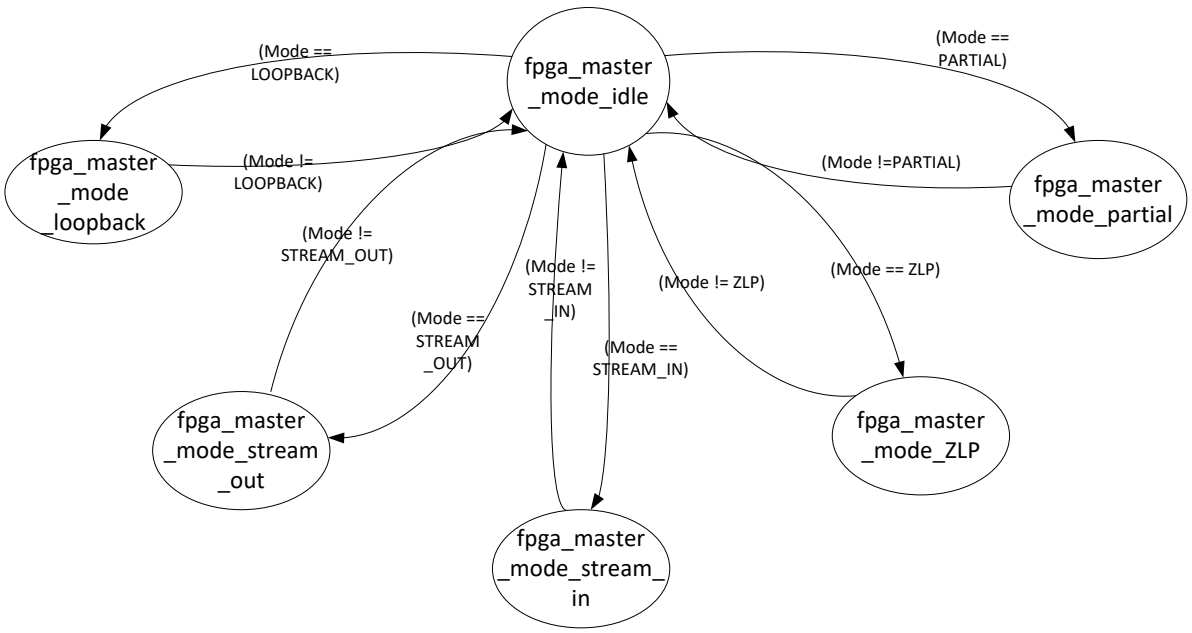

为了得到 FX3 的最大性能，GPIF 接口将以 100 MHz 的频率工作。SP601 配有一个 27 MHz 的板上单端振荡器。FPGA 使用一个 PLL，以从 27 MHz 的时钟生成一个 100 MHz 的时钟。

下面章节描述了不同传输类型的状态实现。

### 10.4.1 FPGA 主设备模式状态机

运行状态机，以选择 FPGA 主设备的传输模式。共有五种传输模式，包括回送、短数据包（局部）、零长度数据包、串流 IN 和串流 OUT 传输。

图 22. 用于选择模式的 FPGA 状态机



**fpga_master_mode_idle 状态：**

若尚未选中传输模式，FPGA 主设备将处于该状态。

**fpga_master_mode_partial 状态：**

如果 mode = PARTIAL，状态机将进入该状态。如果 mode != PARTIAL，状态机将从该模式切换到 fpga_master_mode_idle 状态。

**fpga_master_mode_zlp 状态：**

如果 mode = ZLP，状态机将进入该状态。如果 mode != ZLP，状态机将从该模式切换到 fpga_master_mode_idle 状态。

**fpga_master_mode_stream_in 状态：**

如果 mode = STREAM_IN，状态机将进入该状态。如果 mode != STREAM_IN，状态机将从该模式切换到 fpga_master_mode_idle 状态。

**fpga_master_mode_stream_out 状态：**

如果 mode = STREAM_OUT，状态机将进入该状态。如果 mode != STREAM_OUT，状态机将从该模式切换到 fpga_master_mode_idle 状态。

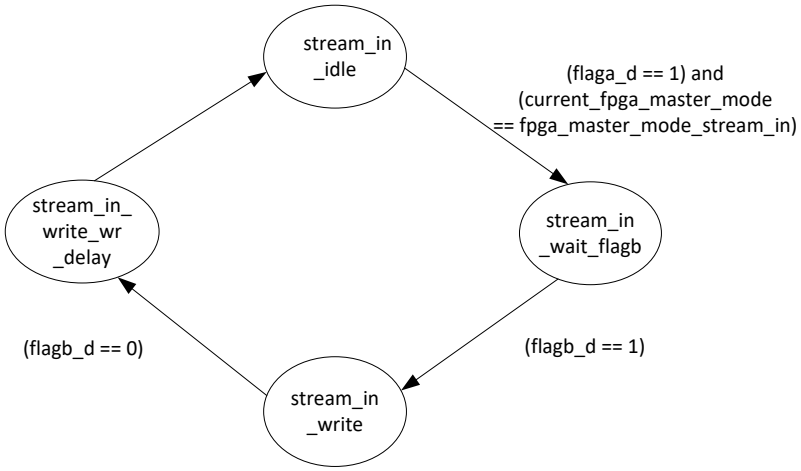
**fpga_master_mode_loop_back 状态：**

如果 mode = LOOPBACK，状态机将进入该状态。如果 mode != LOOPBACK，状态机将从该状态切换到 fpga_master_mode_idle 状态。

### 10.4.2 串流 IN 示例 [FPGA 对从设备 FIFO 进行写操作]

下图显示了 Verilog RTL 中针对串流 IN 传输执行的状态机。

图 23. 串流 IN 的 FPGA 状态机




**stream_in_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**stream_in_wait_flagb 状态：**

每当 flaga_d =1 和 FPGA 主设备模式为 stream_in 时，状态机将进入该状态，并等待 flagb_d。

**stream_in_write 状态：**

每当 flagb_d =1 时，状态机将进入该状态，并开始写入从设备 FIFO 接口。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

### 10.4.3 stream_in_write_wr_delay 状态：

每当 flagb_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

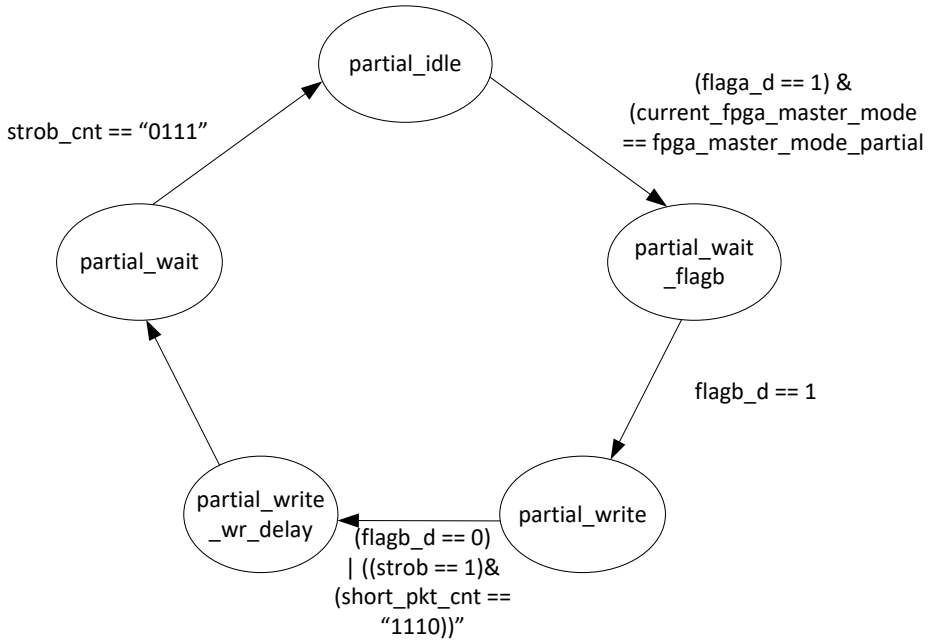
经过一个时钟周期后，状态机将进入 stream_in_idle 状态。

根据使用局部标志情况下的通用公式一节中的公式（1），在局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

### 10.4.4 短数据包示例[FPGA 分别将整个数据包和短数据包写入到从设备 FIFO]

该示例说明了如何使用 PKTEND#传输短数据包的程序。下图显示了在 Verilog RTL 中针对短数据包示例执行的状态机。

图 24. 短数据包传输的状态机




**partial_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**partial_wait_flagb 状态：**

每当 flaga_d=1 和 FPGA 主设备为局部模式时，状态机将进入该状态。

**partial_write 状态：**

每当 flagb_d = 1，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**partial_write_wr_delay 状态：**

每当 flagb_d =0 或（strob=1 和 short_pkt_cnt=“1110”）时，状态机将进入该状态。如果 strob =1，FPGA 主设备将传输一个短数据包。

从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

经过一个时钟周期后，状态机将进入 partial_wait 状态。

根据使用局部标志情况下的通用公式一节中的公式（1），在局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**partial_wait 状态：**

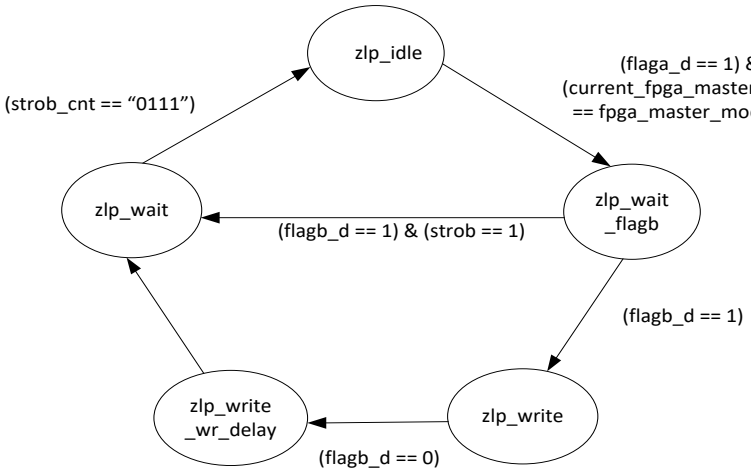
每当 strob_cnt = 0111 时，状态机将进入 partial_idle 状态。

由于水印值为 6，因此，只有经过局部标志（flagb）后的 6 个周期，flaga 才会变为 0。该状态会暂停执行多于 4 个时钟周期，以确保 flaga 上的有效状态。

### 10.4.5 零长度数据包的示例[FPGA 分别将整个数据包和 ZLP 写入到从设备 FIFO]

该示例说明了使用 PKTEND#传输 ZLP 的程序。下图显示了在 Verilog RTL 中针对 ZLP 示例执行的状态机。

图 25. ZLP 传输的状态机




**zlp_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**zlp_wait_flagb 状态：**

每当 flaga_d=1 和 FPGA 主设备为 zlp 模式时，状态机将进入该状态。

**zlp_write 状态：**

每当 flagb_d = 1，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**zlp_write_wr_delay 状态：**

每当 flagb_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

经过一个时钟周期后，状态机将进入 zlp_wait 状态。

根据使用局部标志情况下的通用公式一节中的公式（1），在局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**zlp_wait 状态：**

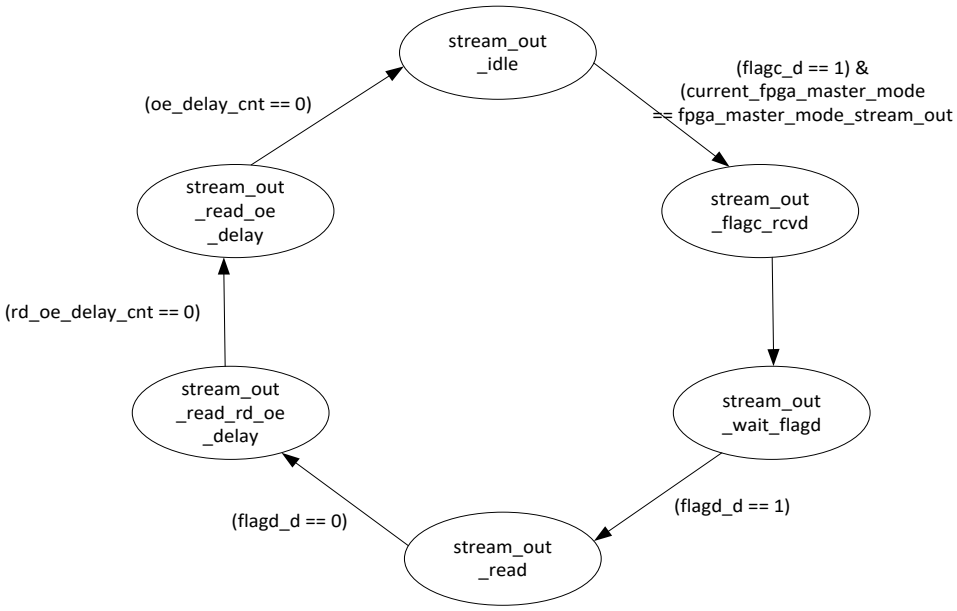
每当 flagb_d = 1 和 strob = 1 时，状态机将从 zlp_wait_flagb 状态切换到该状态，并将一个 zlp 数据包传输到 s lavefifo。

由于水印值为 6，因此，只有经过局部标志（flagb）后的 6 个周期，flaga 才会变为 0。该状态会暂停执行多于 4 个时钟周期，以确保 flaga 上的有效状态。

每当 strob_cnt = 0111 时，状态机将进入 zlp_idle 状态。

### 10.4.6 在 Verilog RTL 中针对串流 OUT 示例执行的状态机

图 26. 串流 OUT 传输的状态机




**stream_out_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**stream_out_flagc_rcvd 状态：**

每当 flagc_d = 1 和 FPGA 主设备为串流 OUT 模式时，状态机将进入该状态。

**stream_out_wait_flagd 状态：**

经过一个时钟周期后，状态机将进入该状态。

**stream_out_read 状态：**

每当 flagc_d = 1 时，状态机将进入该状态。状态机将激活读控制信号如下：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**stream_out_read_rd_oe_delay 状态：**

每当 flagc_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

根据使用局部标志情况下的通用公式一节中的公式（2b），在局部标志（flagd）变为 0 后，FX3 需要对处于激活状态的 SLRD#进行采样三个周期。由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 对被置为 0 的 flagd_d（flagd 的触发输出）进行采样后，激活 SLRD#一个周期。

**stream_out_read_oe_delay 状态：**

每当 rd_oe_delay_cnt = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

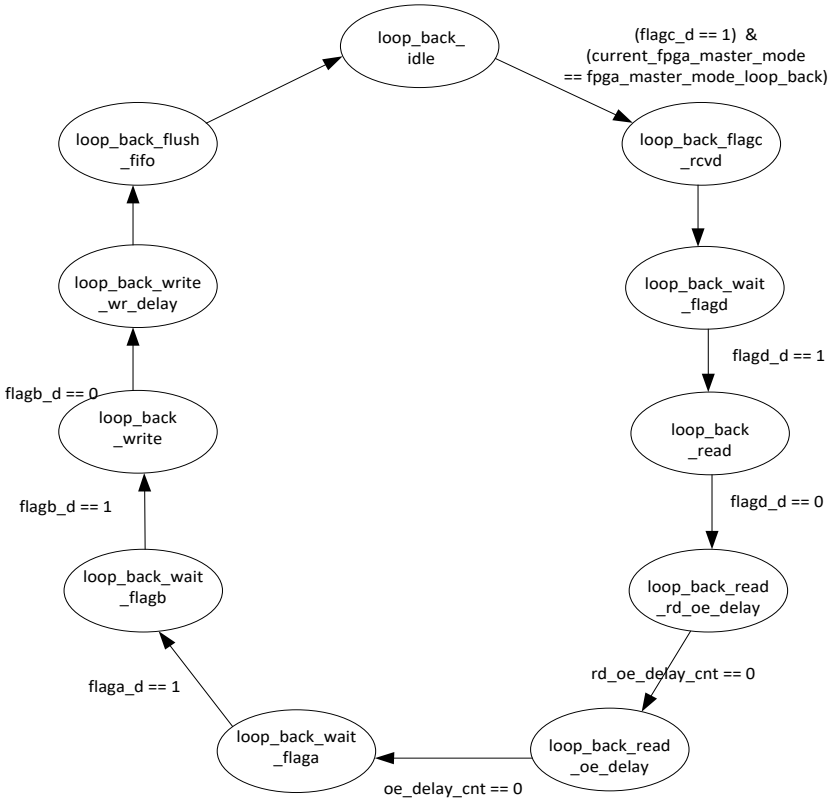
PKTEND# = 1；SLOE# = 0；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

如果 oe_delay_cnt = 0，状态机将从该状态切换到 stream_out_idle 状态。

### 10.4.7 回送示例[FPGA 从从设备 FIFO 读取数据并将该数据回写到从设备 FIFO]

状态机需要经过六个状态后才能完成一个回送周期。下图显示了状态机及其相应操作。

图 27. 回送传输的状态机




**loop_back_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_flagc_rcvd 状态：**

每当 flagc_d = 1 和 FPGA 主设备为回送模式时，状态机将进入该状态。

**loop_back_wait_flagd 状态：**

经过一个时钟周期后，状态机将进入该状态并等待 flagd。

**loop_back_read 状态：**

如果 flagd_d = 1，状态机将进入该状态。状态机将激活读控制信号如下：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_read_rd_oe_delay 状态：**

每当 flagc_d = 0，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

根据使用局部标志情况下的通用公式一节中的公式（2b），在局部标志（flagd）变为 0 后，FX3 需要对处于激活状态的 SLRD#进行采样三个周期。由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 对被置为 0 的 flagd_d（flagd 的触发输出）进行采样后，激活 SLRD#一个周期。

**loop_back_read_oe_delay 状态：**

每当 rd_oe_delay_cnt = 0，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_wait_flaga 状态：**

如果 oe_delay_cnt = 0，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**loop_back_wait_flagb 状态：**

如果 flaga_d = 1，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**loop_back_write 状态：**

如果 flagb_d = 1，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**loop_back_write_wr_delay 状态：**

如果 flagb_d = 0，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

根据使用局部标志情况下的通用公式一节中的公式（1），在局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**loop_back_flush_fifo 状态：**

经过一个时钟周期后，状态机将进入该状态，并清理内部 FIFO。从设备 FIFO 控制线的状态为：

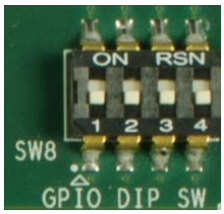
PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

## 10.5 项目操作

### 10.5.1 测试回送传输的步骤

1. 使用 SuperSpeed Explorer 套件时，要确保已将跳线器 J5 开路。使用 FX3 开发套件（CYUSB3KIT-001）时，请按照表 7 对各跳线器和开关进行设置。使用 FMC 连接器将 FX3 DVK 电路板连接至 Xilinx SP601 DVK 电路板，并在给 Xilinx SP601 DVK 供电之前，先给 FX3 DVK 电路板供电。
2. 在 FPGA 开始任何数据操作前，必须先对 FPGA 和 FX3 进行配置。FPGA 代码通过使用 GPIO 输入确定需要启动的模式。

图 28. Xilinx SP601 电路板上的 SW8 — 在 FPGA 和 FX3 配置前均处于关闭状态




3. 使用固件镜像文件（*SF_loopback.img*）对 FX3 器件进行编程。 也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。
4. 使用 *slavefifo2b_fpga_top.bit* 文件来编程 Xilinx Spartan 6 FPGA。可使用任何标准的编程器（如 Xilinx ISE 设计套件中包含的 iMPACT 应用）对 FPGA 进行编程。在编程 Xilinx FPGA 电路板之前，请先对 FX3 DVK 进行编程。如果先编程好了 FPGA，FX3 DVK 将不会在主机 PC 上进行枚举，因为该套件负责驱动 PMODE 线。

图 29. 使用 Control Center 编程 FX3 固件

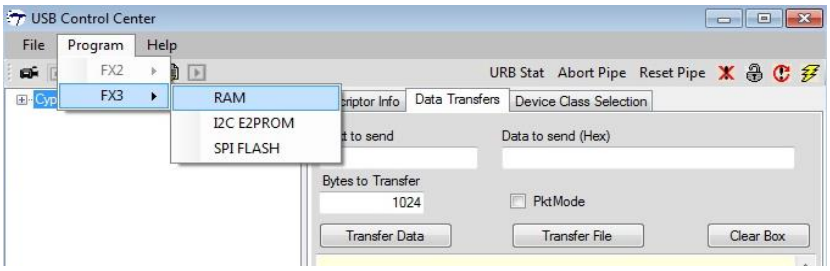
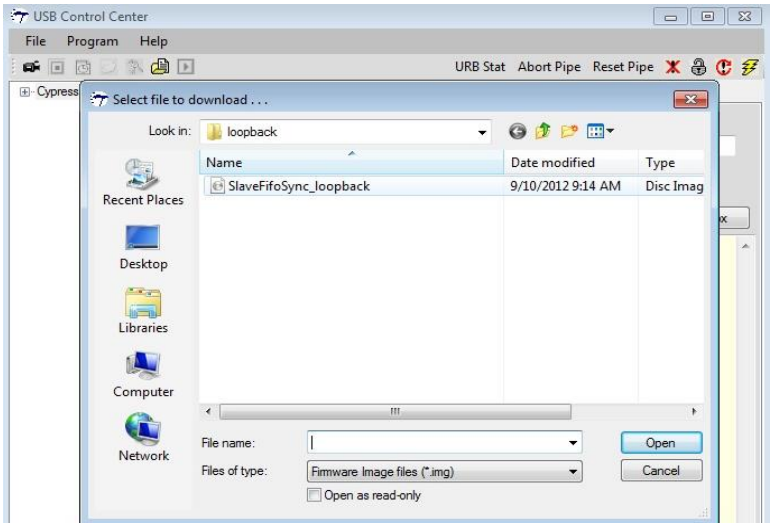



图 30. 使用 Control Center 编程 FX3 固件，用于进行*回送测试*



下载固件后，在启动数据传输前，FX3 器件将作为 SuperSpeed 器件进行枚举（如果已连接至一个 USB 3.0 端口）。FPGA 使用 GPIO 输入来确定需要执行的传输。SP601 DVK 电路板上的 SW8 开关用于该目的。表 6 显示了不同传输所需要的开关设置。

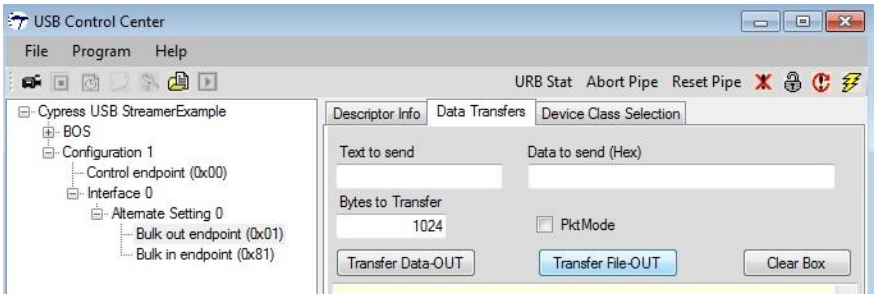
5. Xilinx SP601 电路板上 SW8 的第一个和第三个位置必须为打开，以执行回送传输。

表 6.*slavefifo2b_fpga_top.bit* 中的 FPGA 传输模式配置

| SW8[4] | SW8[3] | SW8[2] | SW8[1] | FPGA 传输模式 |
|---|---|---|---|---|
| 关闭 | 关闭 | 关闭 | 打开 | FPGA 连续写入一完整的数据包，后跟一个短数据包 |
| 关闭 | 关闭 | 打开 | 关闭 | FPGA 连续写入完整的数据包，后跟 ZLP |
| 关闭 | 关闭 | 打开 | 打开 | FPGA 连续写入完整的数据包 （来自主机的串流 BULK IN 数据包） |
| 关闭 | 打开 | 关闭 | 关闭 | FPGA 连续读取完整的数据包（来自主机的串流 BULK OUT 数据包） |
| 关闭 | 打开 | 关闭 | 打开 | 回送传输模式 |
| 关闭 | X | X | X | 无效 |

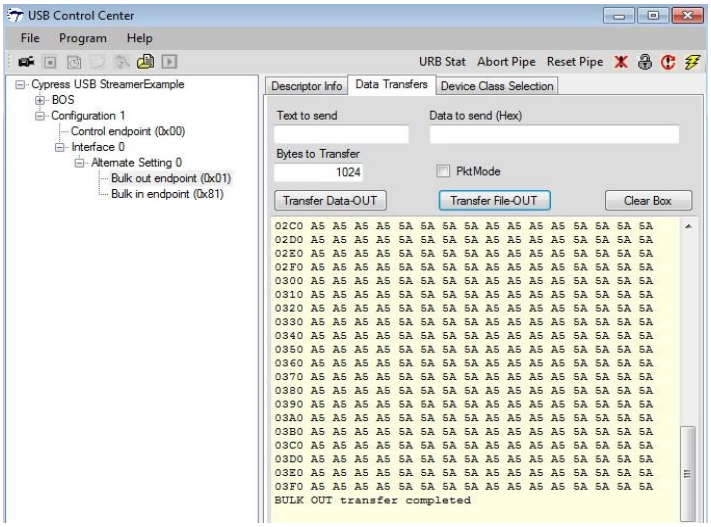
6. 现在，可使用 Control Center 工具启动传输过程。首先，从 USB 主机启动一个 BULK OUT 传输。在 Control Center 中选择 BULK OUT endpoint 项，然后点击 **Transfer File-OUT** 按键。

图 31. 使用 Transfer File-OUT 选项启动 BULK OUT 传输



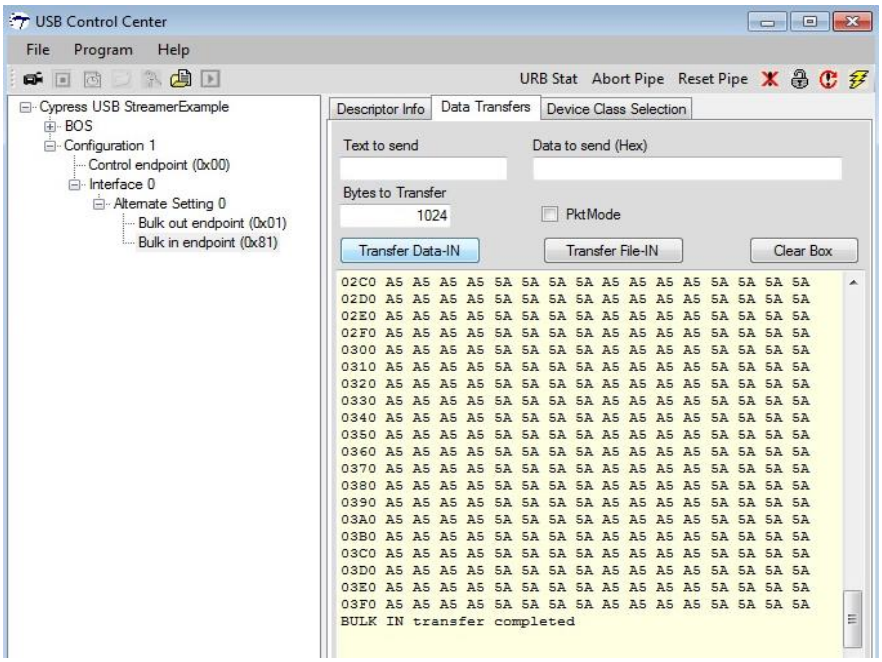

7. 这样，您可浏览并选择包含需要传输的数据的文件。在本应用笔记的附件中，您可在 Loopback 文件夹中找到 *TEST.txt* 文件。该文件包含一个以交替方式发送“0xA5A5A5A5 0x5A5A5A5A”的数据图案。双击选择该文件，然后发送数据。

图 32. 采用 Transfer File-OUT 选项时通过选择 TEST.txt 文件所实现的数据图案传输




8. FPGA 正在等待 FLAGA 变为 1。只要 PIB_SOCKET_0 的缓冲区中的数据可用，FPGA 会立即读取该数据。然后，FPGA 会回送同样的数据，并将其写入到 FX3 的 PIB_SOCKET_3。
9. 您可从 USB 主机发送一个 BULK IN 传输。在 Control Center 中选择 BULK IN endpoint 项，并点击 **Transfer Data-IN** 按键。先前写入的数据将被读取。

图 33. 通过使用 Transfer Data-IN 按键启动 BULK IN 传输，以完成回送测试




### 10.5.2 测试串流传输的步骤

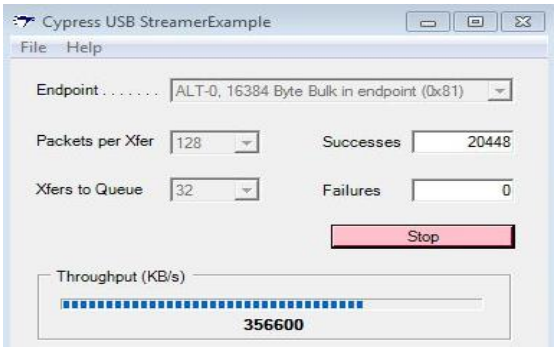
**注意**：始终使用下面路径中 FX3 SDK 所提供的 C++ Streamer 工具：

<FX3_SDK_installation_path>*EZ-USB FX3 SDK**1.3**application**cpp**streamer**x86*\

该路径中的 1.3 是 FX3 SDK 版本编号。FX3 SDK 将来版本的编号可高于本版本的。

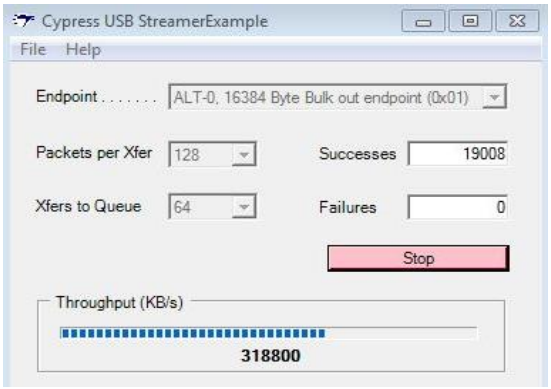
1. 对于串流传输，可以使用先前提及的同样比特文件来保持 FPGA 的编程状态。您仅需要配置开关设置，如表 6 所示。
2. 对于串流 IN 或 OUT，使用固件镜像文件 *SF_streamIN.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。
3. 下载固件后，FX3 器件会作为一个 SuperSpeed 器件进行枚举（如果已连接到 USB 3.0 端口）。这样，可以通过 Control Center 工具或 FX3 SDK 中的串流工具启动数据传输。
4. 使用 *slavefifo2b_fpga_top.bit* 文件来编程 Xilinx Spartan 6 FPGA。
5. 根据所需的传输对 SP601 电路板上的 SW8 开关进行设置，如表 6 所示。
6. 在串流 IN 情况下，FPGA 正在等待 FLAGA 变为 1。当缓冲区可用时，FPGA 立即将数据连续写入到 FX3 的 PIB_SOCKET_0。您可从 USB 主机发送连续的 BULK IN 传输。在赛普拉斯 Streamer 工具中选择 BULK IN 端点，然后点击 **Start**。这时，将显示性能数值。图 34 中显示了在配有 Intel Z77 Express Chipset 的 Win7 64 位电脑上观察到的性能。

图 34. 赛普拉斯 Streamer 工具中显示的串流 IN 性能




7. 在串流 OUT 情况下，FPGA 正在等待 FLAGC 变为 1。当数据可用时，FPGA 立即连续读取 FX3 的 PIB_SOCKET_3 中的数据。您可从 USB 主机发送连续的 BULK OUT 传输。在赛普拉斯 Streamer 工具中选择 BULK OUT 端点，然后点击 **Start**。这时，将显示性能数值。图 35 中显示了在配有 Intel Z77 Express Chipset 的 Win7 64 位电脑上观察到的性能。

图 35. 在赛普拉斯 Streamer 工具中显示的串流 OUT 性能




*SF_streamIN.img* 可使用于串流 IN 和串流 OUT。在 *SF_streamIN.img* 中，已将 8 个缓冲区和 4 个缓冲区分别分配给 P2U DMA 通道和 U2P 通道。在 *SF_streamOUT.img* 文件中，则将 8 个缓冲区和 4 个缓冲区分别分配给 U2P DMA 通道和 P2U 通道。因此，*SF_streamIN.img* 固件文件将显示更高的 P2U 性能；*SF_streamOUT.img* 固件文件将显示更高的 U2P 性能。

### 10.5.3 短数据包和 ZLP 数据包传输的测试步骤：

1. 对于串流传输，可以使用先前提及的同样比特文件来保持 FPGA 的编程状态。您仅需要配置开关设置，如表 6 所示。
2. 使用固件图像文件 *SF_shrt_ZLP.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。
3. 使用 *slavefifo2b_fpga_top.bit* 文件来编程 Xilinx Spartan 6 FPGA。
4. 完成编程 FX3 固件后，分配给 PIB_SOCKET_0 的缓冲区立即可用。通过监控 FLAGA，FPGA 就处于等待该条件发生的状态。一旦 FLAGA 等于 1，FPGA 会开始写入 FX3。
5. 如果将开关配置为短数据包传输，FPGA 将写入完整的数据包（1024 个字节），然后再写入一个短数据包。如果将开关配置为 ZLP 传输，FPGA 将写入一个完整的数据包（1024 个字节），然后再写入一个 ZLP*。
6. 现在，USB 主机可发送 BULK IN 令牌数据包。在 Control Center 工具中，选择 BULK IN 端点，然后点击 **Transfer Data-IN** 按键。首先，将收到完整的数据包。再次点击 **Transfer Data-IN**。现在，将收到短数据包或 ZLP。

   ***注意：**如果传输短数据包后不对 FX3 重新设置，就立即传输 ZLP 的话，那么第一次传输的数据包将是短数据包。ZLP 后立即传输短数据包的情况与此相同。这是因为 FX3 缓冲区不被删除，直到主机要求 Control Center 上的数据。

图 36.连续执行 Transfer Data-IN 操作分别接收到的完整数据包和短数据包

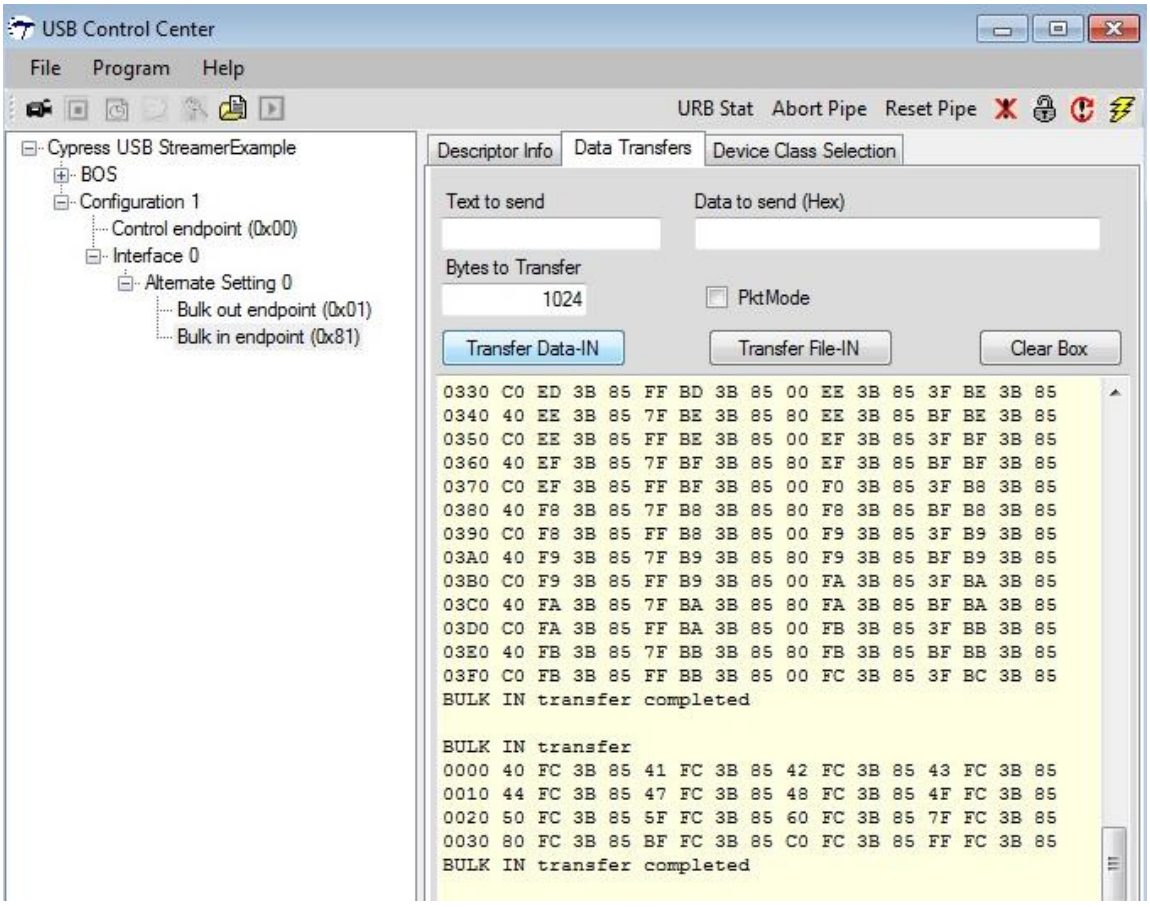
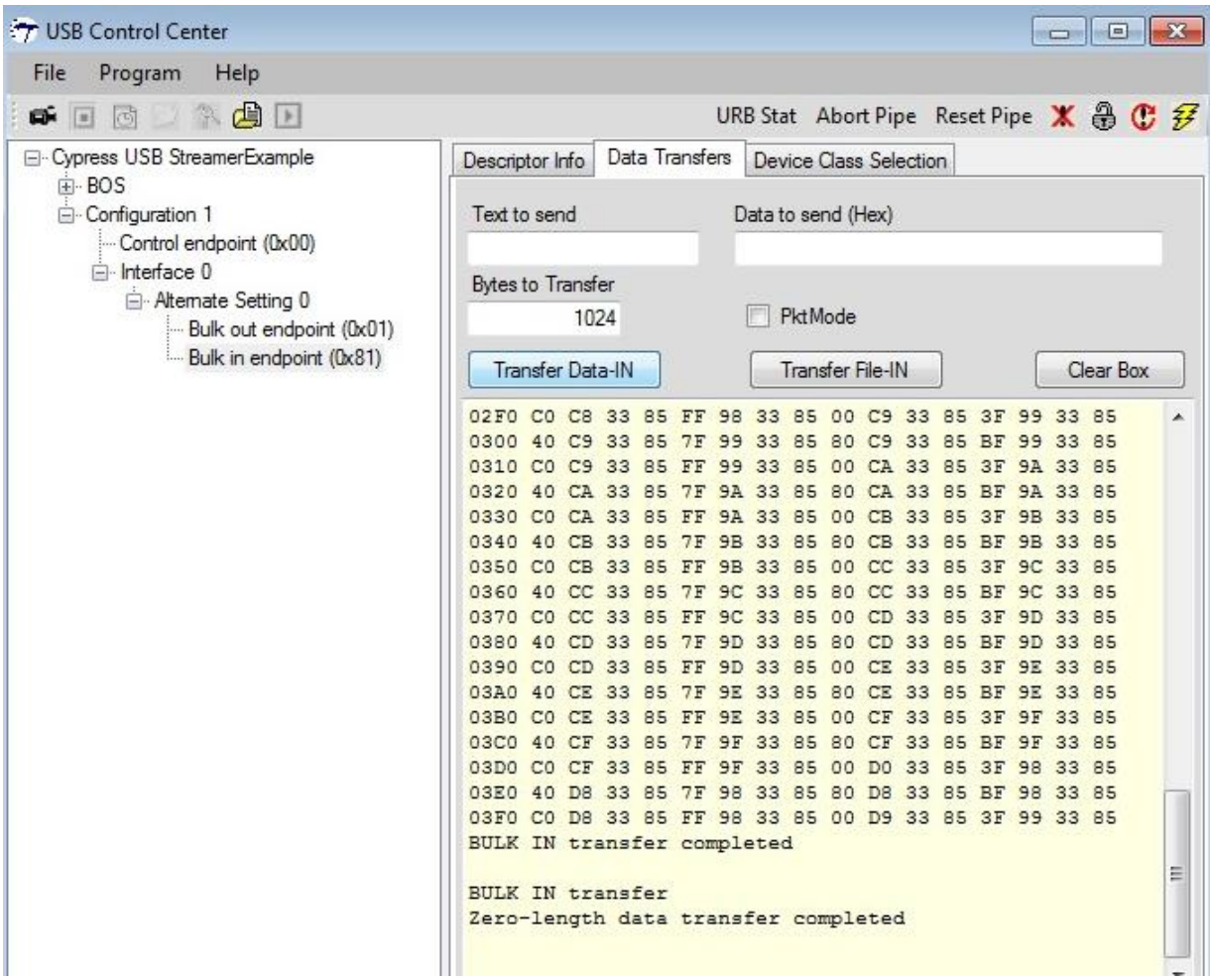


图 37. 连续执行 Transfer Data-IN 操作分别接收到的完整数据包和零长度数据包




7. 由于 FPGA 连续写入数据，所以可实现多次 BULK IN 传输。

# 11. 设计示例 2：将 Altera FPGA 连接至 FX3 的同步从设备 FIFO 接口

本节将提供一个完整的设计示例，其中的 Altera Cyclone 3 FPGA 通过同步从设备 FIFO 接口连接至 FX3。本节还描述用于实现该设计的硬件、固件和软件组件。

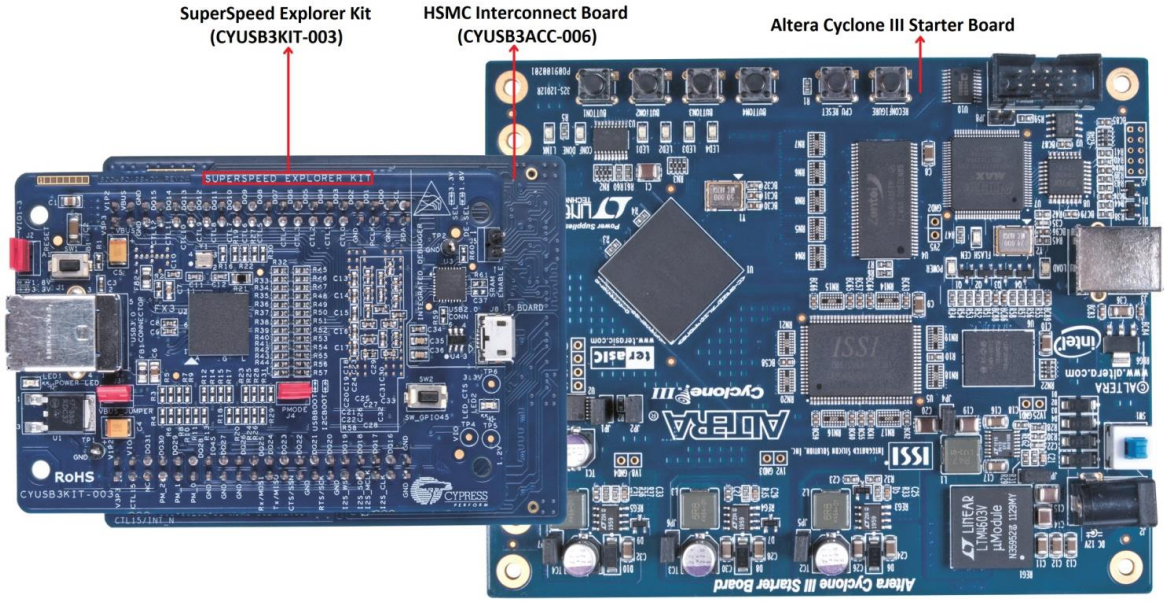
## 11.1 硬件设置

可按照硬件设置执行本示例中的项目。该硬件设置包括一个与 Cyclone III FPGA 入门电路板相连的赛普拉斯 FX3 开发套件（CYUSB3KIT-001）或 SuperSpeed Explorer 套件（CYUSB3KIT-003）。 使用 Samtec 至 HSMC 互联电路板将 FX3 电路板和 Altera 电路板连接起来。CYUSB3ACC-003 互联电路板通过 Samtec 连接器与赛普拉斯 FX3 开发套件（CYUSB3KIT-001）相连，并通过 HSMC 连接器与 Altera 电路板相连。

CYUSB3ACC-006 互联电路板通过多个插座与 SuperSpeed Explorer 套件（CYUSB3KIT-003）相连，并通过 HSMC 连接器与 Xilinx 电路板相连。

图 38 显示了使用 SuperSpeed Explorer 套件的硬件设置。更多信息，请参见附录 B 用 FX3 开发套件（CYUSB3KIT-001）的硬件设置。除了硬件设置，以下各步骤在使用任何 FX3 电路的情况下均相同。

图 38. 赛普拉斯 SuperSpeed Explorer 套件通过 HSMC 互联电路板与 Altera Cyclone III 入门电路板相连



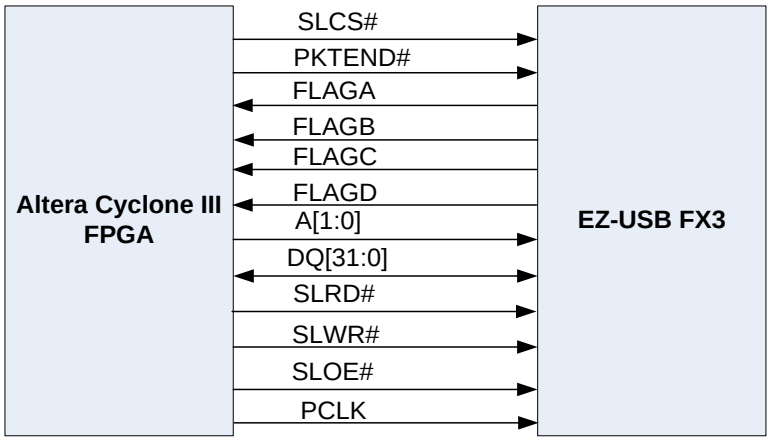

**注意：**要确保在 FPGA 通过从设备 FIFO 接口连接至 FX3 的所有应用中，SuperSpeed Explorer 套件上的跳线器 J5 均处于开路状态。

## 11.2 固件和软件组件

- FX3 同步从设备 FIFO 固件项目可从 FX3 SDK 获取。
- Control Center 和 Streamer 软件工具可从 FX3 SDK 获取。

下图显示了 FPGA 和 FX3 之间的互连图。

图 39. FPGA 和 FX3 之间的互连图




该示例包括以下成分：

- 回送传输：FPGA 先从 FX3 读取整个缓冲区的内容，然后将其回写到 FX3 内。USB 主机应发送 OUT/IN 令牌数据包，用于发送和接收该数据。您可使用 EZ-USB FX3 SDK 中提供的 Control Center 工具实现该操作。
- 短数据包：FPGA 先将一个完整的数据包传输到 FX3，然后再发送一个短数据包。USB 主机应发送 IN 令牌数据包，用于接收该数据。
- 零长度数据包（ZLP）传输：FPGA 先将一个完整的数据包传输到 FX3，然后再发送一个零长度数据包。USB 主机应发送 IN 令牌数据包，用于接收该数据。
- 串流（IN）数据传输：FPGA 实现单向传输，即是通过同步从设备 FIFO 连续将数据写入到 FX3。USB 主机应发送 IN 令牌数据包，用于接收该数据。可使用 EZ-USB FX3 SDK 中的 Control Center 或 Streamer 工具实现该操作。
- 串流（OUT）数据传输：FPGA 实现单向传输，即是通过同步从设备 FIFO 从 FX3 连续读取数据。USB 主机应发送 OUT 令牌数据包，用于发送该数据。可使用 EZ-USB FX3 SDK 中的 Control Center 或 Streamer 工具实现该操作。

## 11.3 FX3 固件的详细信息

FX3 固件是基于 FX3 SDK 中的示例项目。

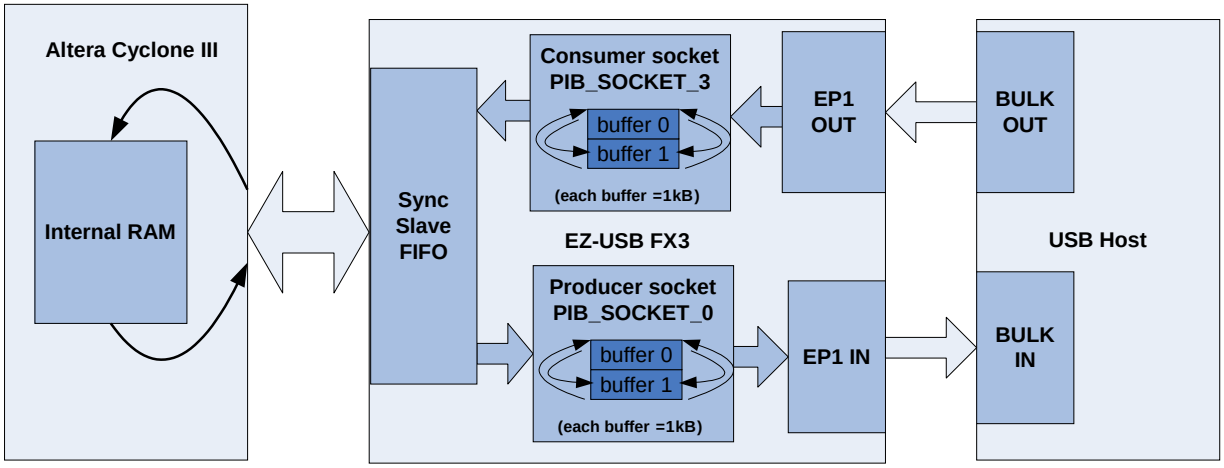
该固件的主要特性包括：

- 启用 USB 3.0 和 USB 2.0。
- 使用赛普拉斯 VID/PID（即 0x04B4/0x00F1）进行枚举。这样，可以使用赛普拉斯的 Control Center 和 Streamer 工具启动 USB 传输。
- 集成具有下面特性的同步从设备 FIFO 描述符：
  - 支持访问多达四个套接字
  - 将数据总线的宽度配置为 32 位
  - 在 100 MHz 的 PCLK 输入时钟频率下工作
  - 能够配置四个标志：
    i. FLAGA：线程 0 专用的满标志
    ii. FLAGB：线程 0 水印值为 6 的专用局部标志
    iii. FLAGC：线程 3 专用的空标志
    iv. FLAGD：线程 3 水印值为 6 的专用局部标志

  **注意：** 本应用笔记中提供的 GPIF II Designer 项目介绍了这些设置。此外，该固件项目还介绍了如何使用 CyU3PGpifSocketConfigure() API 来配置水印值。
- 将 PLL 频率配置为 400 MHz。为了实现该操作，要将 setSysClk400 参数设置为 CyU3PDeviceInit()函数的输入。

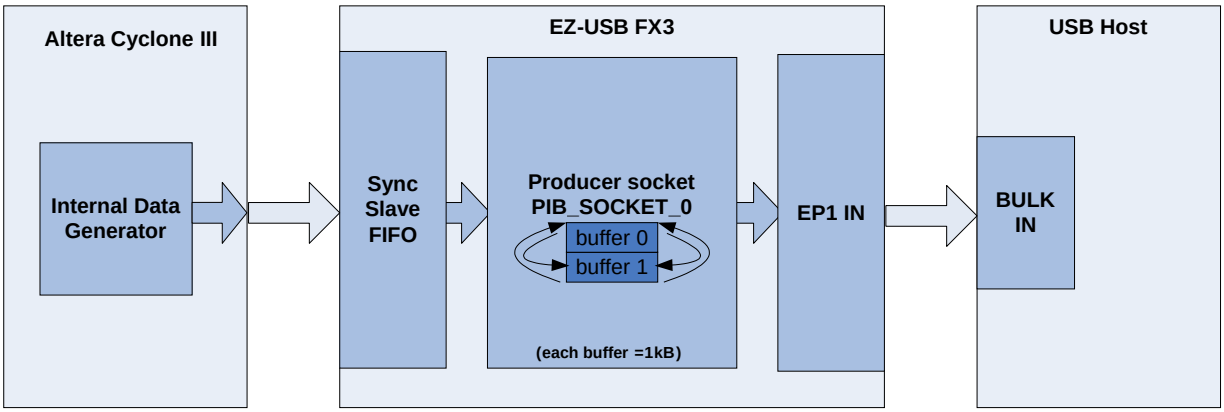
  **注意：** 在工作频率为 100 MHz 且数据总线宽度为 32 位的条件下，该设置对从设备 FIFO 的运行起着关键作用。
- 设置 DMA 通道：
  - 对于回送传输、短数据包和 ZLP 传输，需要创建两个 DMA 通道：
    - 一个是 P2U 通道，其中 PIB_SOCKET_0 作为发送套接字，UIB_SOCKET_1 作为接收套接字。DMA 缓冲区大小为 512 还是 1024 取决于 USB 连接是 USB 2.0 还是 USB 3.0。DMA 缓冲区的数量为 2。
    - 另一个是 U2P 通道，其中 PIB_SOCKET_3 作为接收套接字，UIB_SOCKET_1 作为发送套接字。DMA 缓冲区大小为 512 还是 1024 取决于 USB 连接是 USB 2.0 还是 USB 3.0。DMA 缓冲区的数量为 2。

图 40. 回送传输设置




**注意：**只有 P2U 通道用于传输短数据包和 ZLP 数据包。

图 41. 短数据包和 ZLP 传输的设置

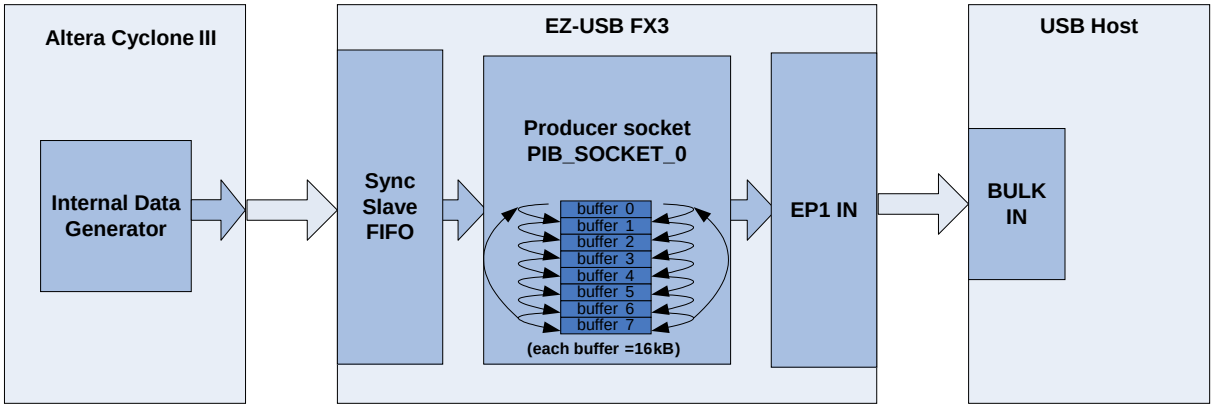



本节所描述的 DMA 通道仅在本应用笔记中提供的 FX3 固件项目内 *cyfxslfifosync.h* 文件中的下列定义被使能时设置。

```
/* set up DMA channel for loopback/short packet/ZLP transfers */
#define LOOPBACK_SHRT_ZLP
```

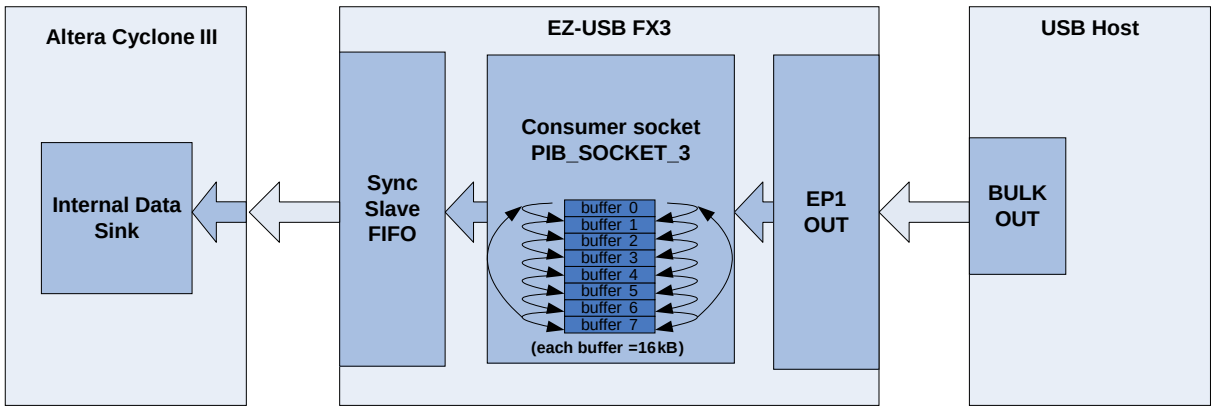
- 对于串流传输，需要创建两个 DMA 通道：
  - 一个是 P2U 通道，其中 PIB_SOCKET_0 作为发送套接字，UIB_SOCKET_1 作为接收套接字。根据所使用的 USB 连接类型，DMA 缓冲区大小将为 16×1024（USB 3.0 连接方式）或为 16×512（USB 2.0 连接方式）。DMA 缓冲区的数量为 8。选择该缓冲区大小和数量，以得到高吞吐量性能。

图 42. 串流 IN 传输设置 — 优化缓冲区数量和大小，以得到更好的性能




  - 另一个是 U2P 通道，其中 PIB_SOCKET_3 作为接收套接字，UIB_SOCKET_1 作为发送套接字。DMA 缓冲区大小为 16×1024（USB 3.0 连接方式）或为 16×512（USB 2.0 连接方式）。DMA 缓冲区的数量为 4。请注意，可通过增大缓冲区数量来增强性能，但需要降低 P2U 通道的缓冲区数量。其原因是 FX3 SDK 所提供的缓冲区存储空间不足以支持两个通道同时具有 8 个大小为 16×1024 的缓冲区。

图 43. 串流 OUT 传输设置 — 优化缓冲区数量和大小，以得到更好的性能




前面描述的 DMA 通道仅在本应用笔记中提供的 FX3 固件项目内 *cyfxslfifosync.h* 文件中的下列定义被使能时设置。

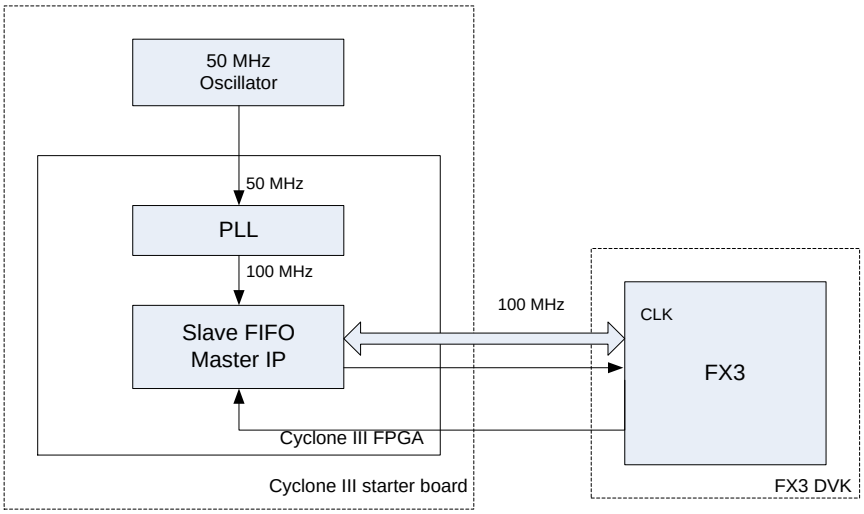
```
/* set up DMA channel for stream IN/OUT transfers */
#define STREAM_IN_OUT
```

通过使用下面各定义（*cyfxslfifosync.h* 文件中也包含这些定义），可以控制分配给 P2U 和 U2P DMA 通道的缓冲区数量。

```
/* Slave FIFO P_2_U channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_P_2_U      (4)
/* Slave FIFO U_2_P channel buffer count */
#define CY_FX_SLFIFO_DMA_BUF_COUNT_U_2_P      (8)
```

## 11.4 FPGA 实现的详细信息

图 44. 使用 SP601 评估套件实现的 Altera Cyclone III （EP3C25F324C6）FPGA



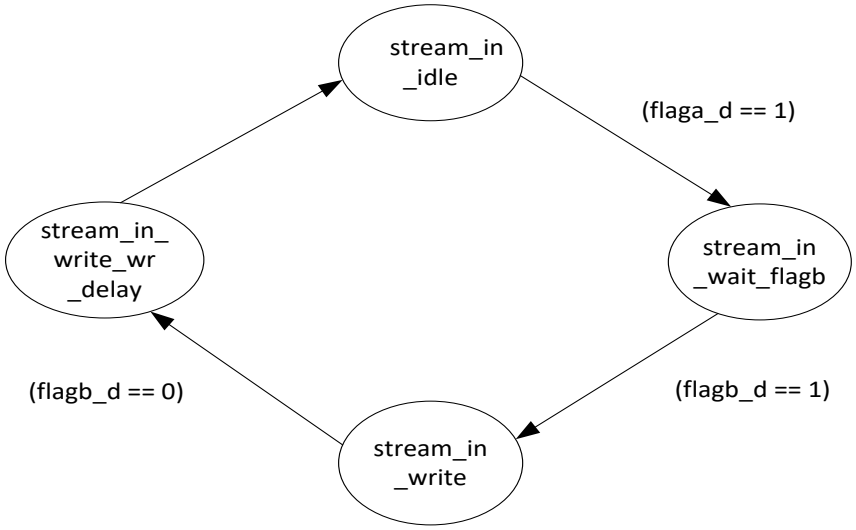

为了得到 FX3 的最大性能，GPIF 接口将以 100 MHz 的频率工作。Cyclone III 入门电路板配有一个 50 MHz 的板上单端振荡器。FPGA 使用一个 PLL，以从 50 MHz 的时钟生成一个 100 MHz 的时钟。

下面是不同传输类型的状态实现：

### 11.4.1 串流 IN 示例[FPGA 对从设备 FIFO 进行写操作]

下图显示了 Verilog RTL 中针对串流 IN 传输执行的状态机。

图 45. 串流 IN 的 FPGA 状态机




**stream_in_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**stream_in_wait_flagb 状态：**

每当 flaga_d = 1 时，状态机将进入该状态，并等待 flagb_d。

**stream_in_write 状态：**

每当 flagb_d = 1 时，状态机将进入该状态，并开始写入从设备 FIFO 接口。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

### 11.4.2 stream_in_write_wr_delay 状态：

每当 flagb_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

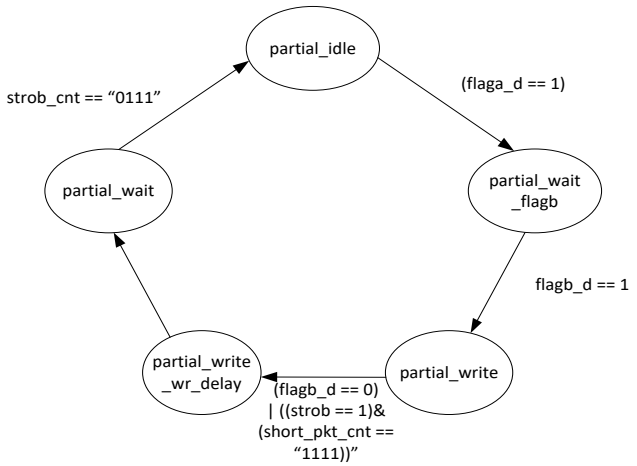
经过一个时钟周期后，状态机将进入 stream_in_idle 状态

根据使用局部标志情况下的通用公式一节中的公式（1），局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

### 11.4.3 短数据包示例[FPGA 分别将整个数据包和短数据包写入到从设备 FIFO]

该示例说明了使用 PKTEND#传输短数据包的程序。下图显示了在 Verilog RTL 中针对短数据包示例执行的状态机。

图 46. 短数据包传输的状态机




**partial_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**partial_wait_flagb 状态：**

每当 flaga_d = 1 时，状态机将进入该状态。

**partial_write 状态：**

每当 flagb_d = 1 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**partial_write_wr_delay 状态：**

每当 flagb_d = 0 或（strob = 1 和 short_pkt_cnt = “1111”）时，状态机将进入该状态。如果 strob = 1，FPGA 主设备将传输一个短数据包。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

经过一个时钟周期后，状态机将进入 partial_wait 状态。

根据使用局部标志情况下的通用公式一节中的公式（1），局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**partial_wait 状态：**

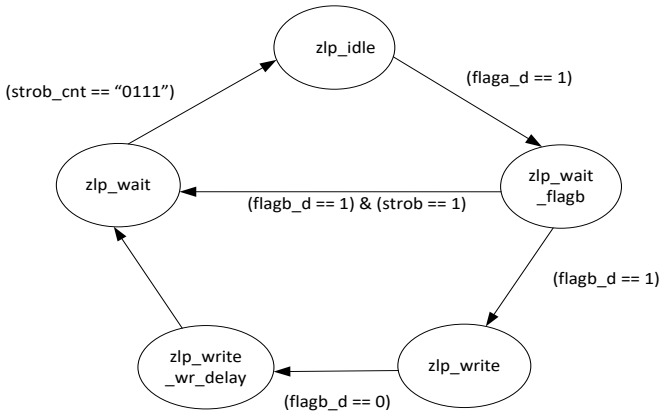
每当 strob_cnt = 0111 时，状态机将进入 partial_idle 状态。

由于水印值为 6，因此，只有经过局部标志（flagb）后的 6 个周期，flaga 才会变为 0。该状态会暂停执行多于 4 个时钟周期，以确保 flaga 上的有效状态。

### 11.4.4 零长度数据包的示例[FPGA 分别将整个数据包和 ZLP 写入到从设备 FIFO]

该示例说明了使用 PKTEND#传输 ZLP 的程序。下图显示了在 Verilog RTL 中针对 ZLP 示例执行的状态机。

图 47. ZLP 传输的状态机




**zlp_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**zlp_wait_flagb 状态：**

每当 flaga_d = 1 时，状态机将进入该状态。

**zlp_write 状态：**

每当 flagb_d = 1 时，状态机将进入该状态。从设备 FIFO 控制线的状态为

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**zlp_write_wr_delay 状态：**

每当 flagb_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：
PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

经过一个时钟周期后，状态机将进入 zlp_wait 状态。

根据使用局部标志情况下的通用公式一节中的公式（1），局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**zlp_wait 状态：**

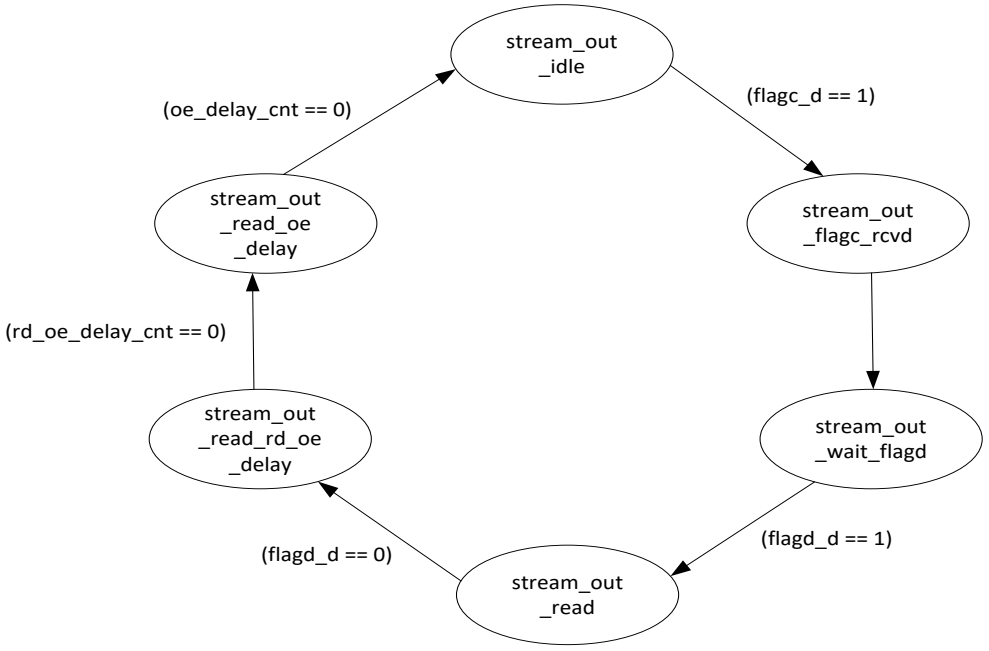
每当 flagb_d = 1 和 strob = 1 时，状态机将从 zlp_wait_flagb 状态切换到该状态，并将一个 zlp 数据包传输到 slavefifo。

由于水印值为 6，因此，只有经过局部标志（flagb）后的 6 个周期，flaga 才会变为 0。该状态会暂停执行多于 4 个时钟周期，以确保 flaga 上的有效状态。

每当 strob_cnt = 0111 时，状态机将进入 zlp_idle 状态。

### 11.4.5 在 Verilog RTL 中针对串流 OUT 示例执行的状态机

图 48. 串流 OUT 传输的状态机




**stream_out_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**stream_out_flagc_rcvd 状态：**

每当 flagc_d = 1 时，状态机将进入该状态。

**stream_out_wait_flagd 状态：**

经过一个时钟周期后，状态机将进入该状态。

**stream_out_read 状态：**

每当 flagc_d = 1 时，状态机将进入该状态。状态机将激活读控制信号如下：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**stream_out_read_rd_oe_delay 状态：**

每当 flagc_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

根据使用局部标志情况下的通用公式一节中的公式（2b），局部标志（flagd）变为 0 后，FX3 需要对处于激活状态的 SLRD#进行采样三个周期。由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 对被置为 0 的 flagd_d（flagd 的触发输出）进行采样后，激活 SLRD#一个周期。

**stream_out_read_oe_delay 状态：**

每当 rd_oe_delay_cnt = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

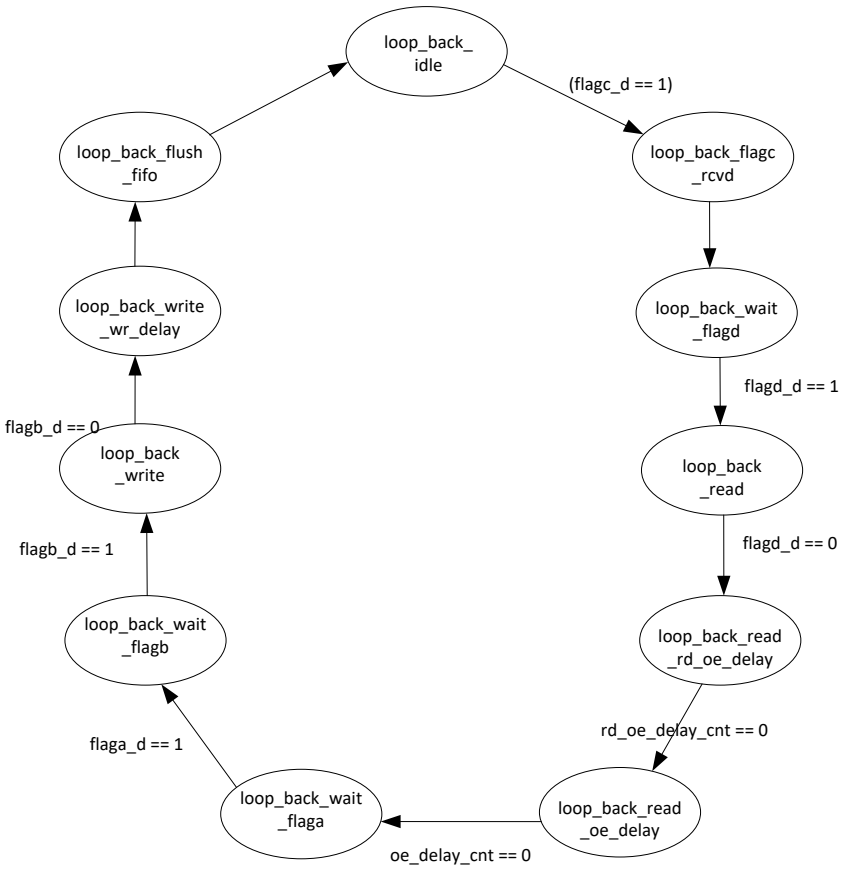
PKTEND# = 1；SLOE# = 0；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

如果 oe_delay_cnt = 0，状态机将从该状态切换到 stream_out_idle 状态。

### 11.4.6 回送示例[FPGA 从从设备 FIFO 读取数据，并将数据回写到从设备 FIFO]：

状态机需要经过六个状态后才能完成一个回送循环。下图显示了状态机及其相应操作。

图 49. 回送传输的状态机




**loop_back_idle 状态：**

该状态初始化状态机中所使用的所有寄存器和信号。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_flagc_rcvd 状态：**

每当 flagc_d = 1 时，状态机将进入该状态。

**loop_back_wait_flagd 状态：**

经过一个时钟周期后，状态机将进入该状态并等待 flagd。

**loop_back_read 状态：**

如果 flagd_d = 1，状态机将进入该状态。状态机将激活读控制信号如下：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_read_rd_oe_delay 状态：**

每当 flagc_d = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 0；SLCS# = 0；SLWR# = 1；A[1:0] = 3

根据使用局部标志情况下的通用公式一节中的公式（2b），局部标志（flagd）变为 0 后，FX3 需要对处于激活状态的 SLRD#进行采样三个周期。由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 对被置为 0 的 flagd_d（flagd 的触发输出）进行采样后，激活 SLRD#一个周期。

**loop_back_read_oe_delay 状态：**

每当 rd_oe_delay_cnt = 0 时，状态机将进入该状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 0；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 3

**loop_back_wait_flaga 状态：**

如果 oe_delay_cnt = 0，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**loop_back_wait_flagb 状态：**

如果 flaga_d = 1，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

**loop_back_write 状态：**

如果 flagb_d = 1，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 0；A[1:0] = 0

**loop_back_write_wr_delay 状态：**

如果 flagb_d = 0，状态机将进入 loop_back_wait_flaga 状态。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

根据使用局部标志情况下的通用公式一节中的公式（1），局部标志（flagb）变为 0 后，FX3 需要对处于激活状态的 SLWR#进行采样两个周期。 由于考虑到 FPGA 至接口的一个周期的传输延迟，FPGA 在对被置为 0 的局部 flagb_d（flagb 的触发输出）进行采样后激活 SLWR#一个周期。

**loop_back_flush_fifo 状态：**

经过一个时钟周期后，状态机将进入该状态，并清理内部 FIFO。从设备 FIFO 控制线的状态为：

PKTEND# = 1；SLOE# = 1；SLRD# = 1；SLCS# = 0；SLWR# = 1；A[1:0] = 0

## 11.5 项目操作

### 11.5.1 回送传输的测试步骤

1. 使用 SuperSpeed Explorer 套件时，要确保已将跳线器 J5 开路。使用 FX3 开发套件（CYUSB3KIT-001）时，请按照表 7 对各跳线器和开关进行设置。使用 HSMC 连接器将 FX3 DVK 电路板连接至 Altera Cyclone III FPGA 入门电路板，并在给 Altera Cyclone III FPGA 入门电路板供电之前，先给 FX3 DVK 电路板供电。
2. 使用固件镜像文件 *SF_loopback.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。编程 Altera Cyclone III FPGA 前，应先对 FX3 进行编程。下载固件后，FX3 器件会作为一个 SuperSpeed 器件进行枚举（如果已连接到 USB 3.0 端口）。

图 50. 使用 Control Center 对 FX3 固件进行编程

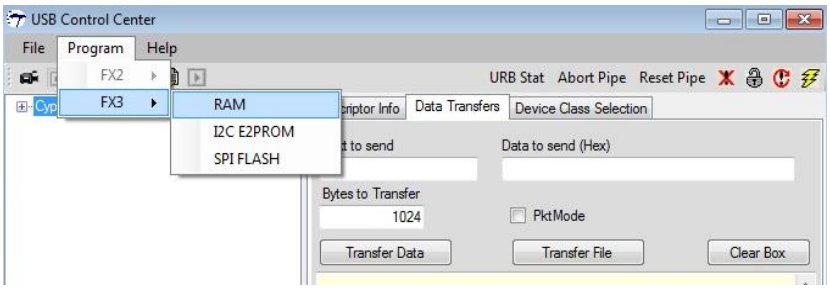
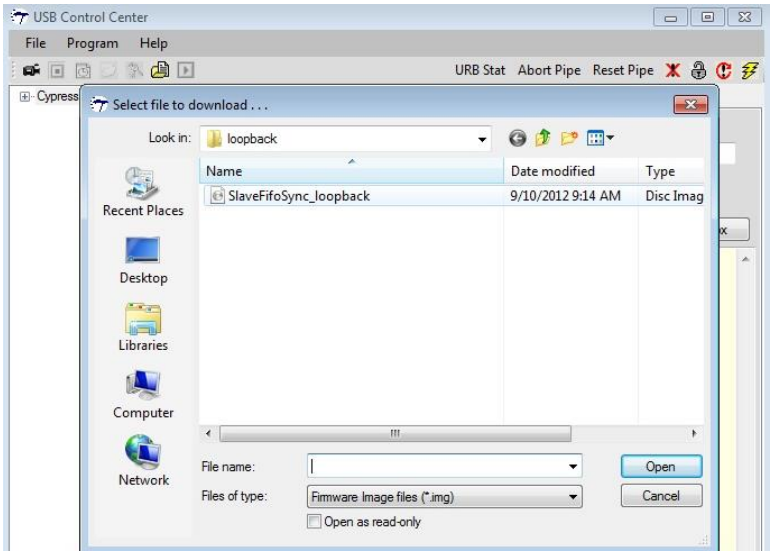
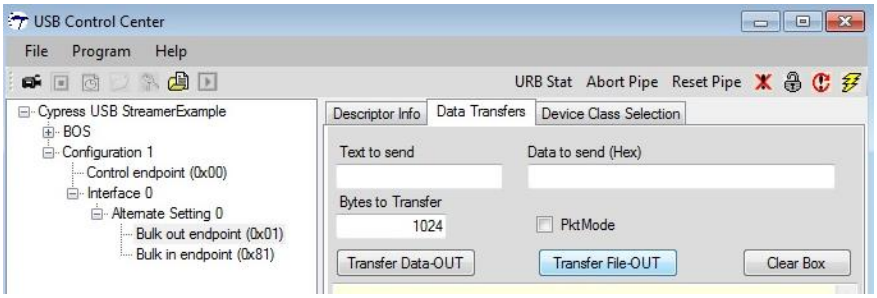



图 51. 使用 Control Center 编程 FX3 固件，用于回送测试



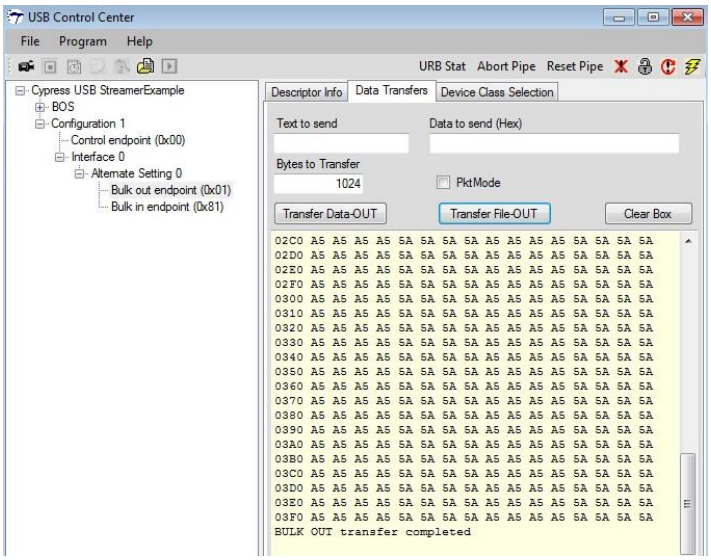

3. 使用 *slaveFIFO2b_loopback.sof* 文件编程 Altera Cyclone III FPGA。可使用任意标准的编程器（如 Quartus II 软件中的 USB-Blaster 应用）编程 FPGA。
4. 现在，可使用 Control Center 工具启动传输过程。首先，从 USB 主机启动一个 BULK OUT 传输。在 Control Center 中选择 BULK OUT endpoint 项，然后点击 **Transfer File-OUT** 按键。

图 52. 使用 Transfer File-OUT 选项启动 BULK OUT 传输



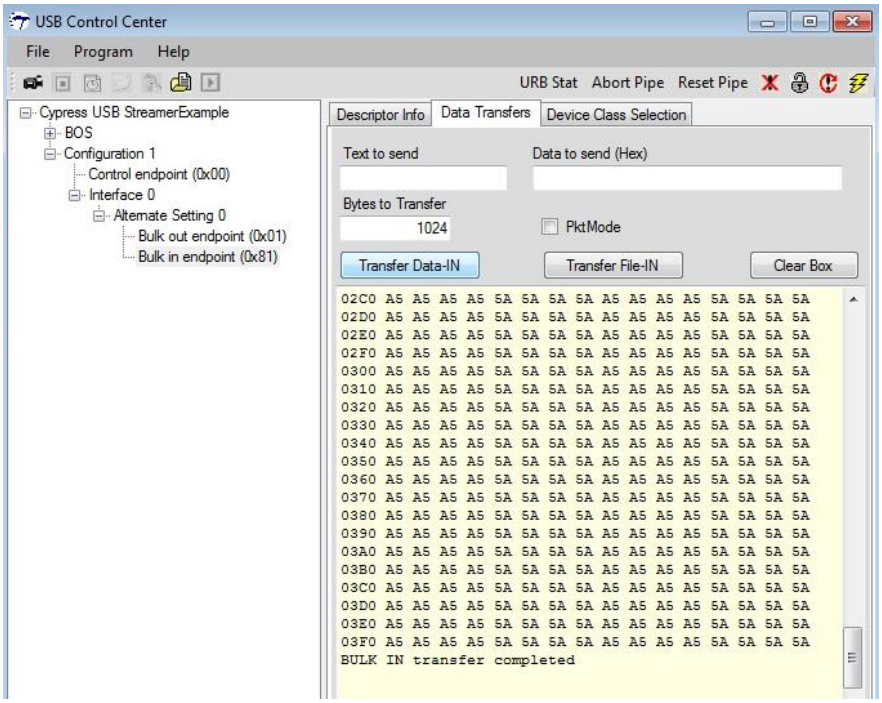

5. 这样，您可浏览并选择包含需要传输的数据的文件。在本应用笔记的附件中，您可在回送文件夹中找到 *TEST.txt* 文件。该文件包含一个以交替方式发送“0xA5A5A5A5 0x5A5A5A5A”的数据图案。双击选择该文件，然后发送数据。

图 53. 采用 Transfer File-OUT 选项时通过选择 TEST.txt 文件所实现的数据图案传输




6. FPGA 正在等待 FLAGA 变为 1。只要 PIB_SOCKET_0 的缓冲区中的数据可用，FPGA 会立即读取该数据。然后，FPGA 会回送同样的数据，并将其写入到 FX3 的 PIB_SOCKET_3。
7. 您可从 USB 主机发送一个 BULK IN 传输。在 Control Center 中选择 BULK IN endpoint 项，并点击 **Transfer Data-IN** 按键。这样，先前写入的数据将被读取。

图 54. 通过使用 Transfer Data-IN 按键启动 BULK IN 传输，以完成回送测试




### 11.5.2 串流传输的测试步骤

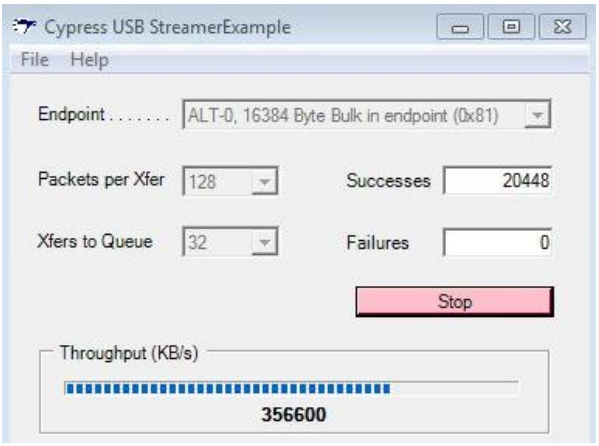
**注意：** 始终使用下面路径中 FX3 SDK 所提供的 C++ Streamer 工具：

*<FX3_SDK_installation_path>\EZ-USB FX3 SDK\1.3\application\cpp\streamer\x86*

该路径中的 1.3 是 FX3 SDK 版本编号。FX3 SDK 将来版本的编号可高于本版本的。

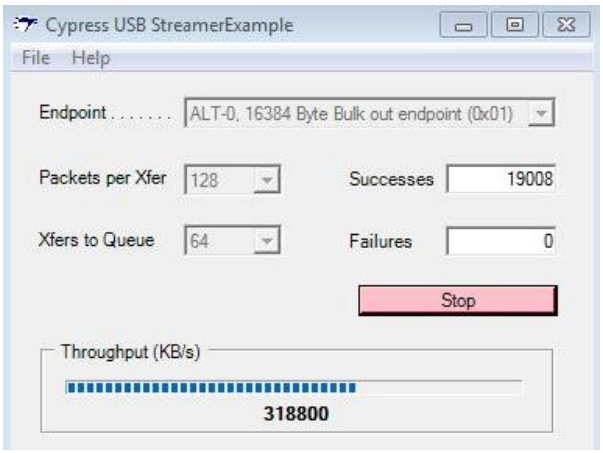
1. 使用 HSMC 连接器将 FX3 DVK 电路板连接至 Altera Cyclone III FPGA 入门电路板，并同时给 FX3 DVK 电路板和 Altera Cyclone III FPGA 入门电路板供电。
2. 对于串流 IN 或串流 OUT，使用固件镜像文件 *SF_streamIN.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。编程 Altera Cyclone III FPGA 前，应先对 FX3 进行编程。下载固件后，FX3 器件会作为一个超速器件进行枚举（如果已连接到 USB 3.0 端口）。
3. 对于串流 IN 传输和串流 OUT 传输，分别使用 *slaveFIFO2b_streamIN.sof* 文件和 *slaveFIFO2b_streamOUT.sof* 文件编程 Altera Cyclone III FPGA。可使用任意标准的编程器（如 Quartus II 软件中的 USB-Blaster 应用）编程 FPGA。
4. 在串流 IN 情况下，FPGA 正在等待 FLAGA 变为 1。当缓冲区可用时，FPGA 立即将数据连续写入到 FX3 的 PIB_SOCKET_0。您可从 USB 主机发送连续的 BULK IN 传输。在赛普拉斯 Streamer 工具中选择 BULK IN 端点，然后点击 **Start**。这时，将显示性能数值。图 55 中显示了在配有 Intel Z77 Express Chipset 的 Win7 64 位电脑上观察到的性能。

图 55. 赛普拉斯 Streamer 工具中显示的串流 IN 性能




5. 在串流 OUT 情况下，FPGA 正在等待 FLAGC 变为 1。当数据可用时，FPGA 立即连续读取 FX3 的 PIB_SOCKET_3 中的数据。您可从 USB 主机发送连续的 BULK OUT 传输。在赛普拉斯 Streamer 工具中选择 BULK OUT 端点，然后点击 **Start**。这时，将显示性能数值。图 56 中显示了在配有 Intel Z77 Express Chipset 的 Win7 64 位电脑上观察到的性能。

图 56. 赛普拉斯 Streamer 工具中显示的串流 OUT 性能

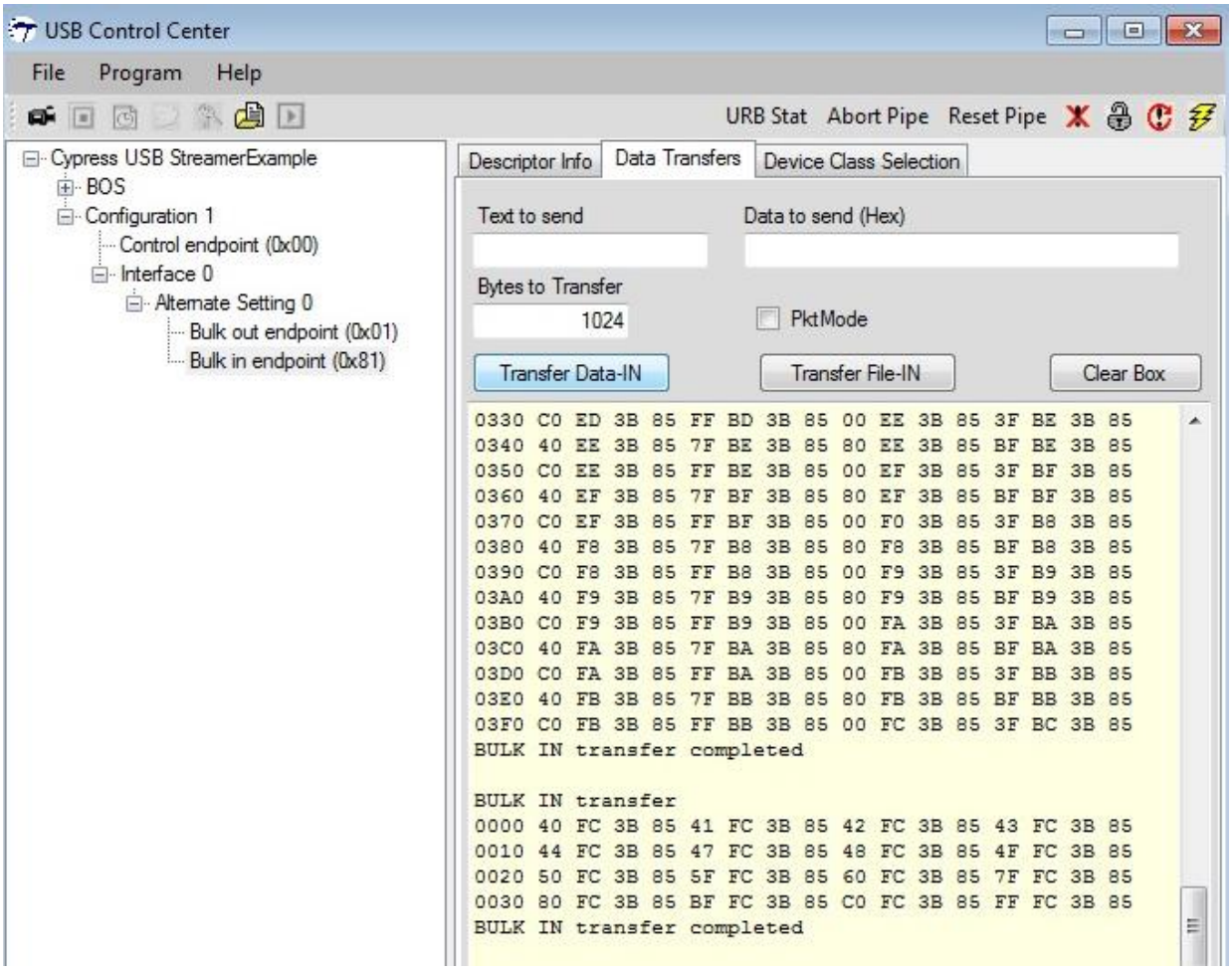



*SF_streamIN.img* 文件可使用于串流 IN 和串流 OUT。在 *SF_streamIN.img* 中，将 8 个缓冲区和 4 个缓冲区分别分配给 P2U DMA 通道和 U2P 通道。在 *SF_streamOUT.img* 中，则将 8 个缓冲区和 4 个缓冲区分别分配给 U2P DMA 通道和 P2U 通道。因此，*SF_streamIN.img* 固件文件将显示更高的 P2U 性能；*SF_streamOUT.img* 固件文件将显示更高的 U2P 性能。

### 11.5.3 短数据包传输的测试步骤

1. 使用 HSMC 连接器将 FX3 DVK 电路板连接至 Altera Cyclone III FPGA 入门电路板，并同时给 FX3 DVK 电路板和 Altera Cyclone III FPGA 入门电路板供电。
2. 对于串流 IN 或串流 OUT，使用固件镜像文件 *SF_shrt_ZLP.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。编程 Altera Cyclone III FPGA 前，应先对 FX3 进行编程。下载固件后，FX3 器件会作为一个 SuperSpeed 器件进行枚举（如果已连接到 USB 3.0 端口）。
3. 通过使用 *slaveFIFO2b_partial.sof* 文件对 Altera Cyclone III FPGA 进行编程。可使用任意标准的编程器（如 Quartus II 软件中的 USB-Blaster 应用）编程 FPGA。
4. 完成编程 FX3 固件后，分配给 PIB_SOCKET_0 的缓冲区立即可用。通过监控 FLAGA，FPGA 正在处于等待该条件发生的状态。一旦标志等于 1，FPGA 会开始对 FX3 进行写操作。
5. FPGA 写入一个完整的数据包（1024 个字节），然后再写入一个短数据包。
6. 现在，USB 主机可发送 BULK IN 令牌数据包。在 Control Center 工具中，选择 BULK IN 端点，然后点击 **Transfer Data-IN** 按键。首先，将收到完整的数据包。再次点击 **Transfer Data-IN** 按键。现在，将收到短数据包。

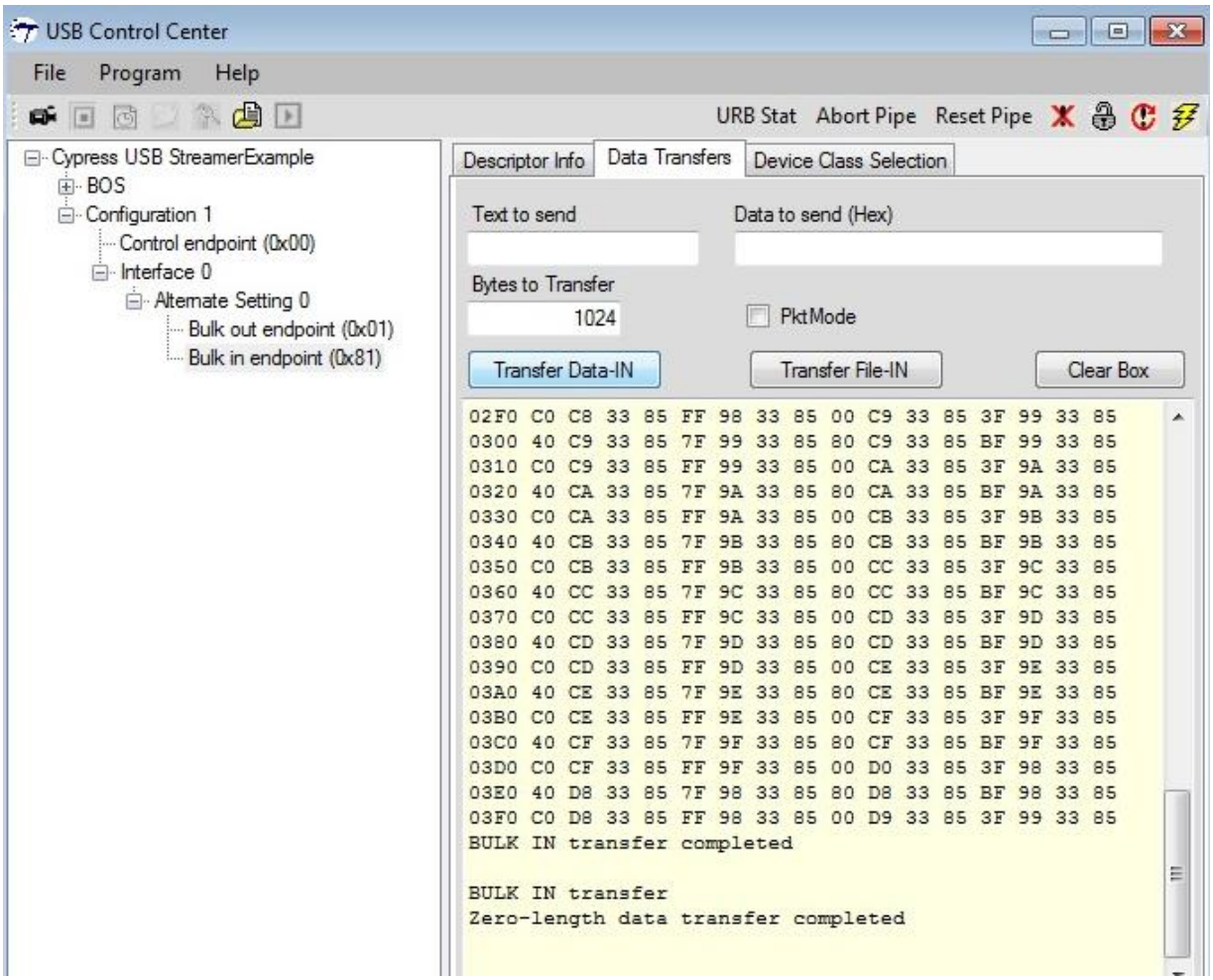
图 57. 连续执行 Transfer Data-IN 操作分别接收到的完整数据包和短数据包



### 11.5.4 ZLP 传输的测试步骤

1. 使用 HSMC 连接器将 FX3 DVK 电路板连接至 Altera Cyclone III FPGA 入门电路板，并同时给 FX3 DVK 电路板和 Altera Cyclone III FPGA 入门电路板供电。
2. 对于串流 IN 或串流 OUT，使用固件镜像文件 *SF_shrt_ZLP.img* 编程 FX3 器件。也可以使用 FX3 SDK 中的 Control Center 工具通过 USB 主机编程 FX3。编程 Altera Cyclone III FPGA 前，应先对 FX3 进行编程。下载固件后，FX3 器件会作为一个 SuperSpeed 器件进行枚举（如果已连接到 USB 3.0 端口）。
3. 通过使用 *slaveFIFO2b_ZLP.sof* 文件编程 Altera Cyclone III FPGA。可使用任意标准的编程器（如 Quartus II 软件中的 USB-Blaster 应用）编程 FPGA。
4. 完成编程 FX3 固件后，分配给 PIB_SOCKET_0 的缓冲区立即可用。通过监控 FLAGA，FPGA 正在处于等待该条件发生的状态。一旦标志等于 1，FPGA 会开始对 FX3 进行写操作。
5. FPGA 写入一个完整的数据包（1024 个字节），然后再写入一个 ZLP。
6. 现在，USB 主机可发送 BULK IN 令牌数据包。在 Control Center 工具中，选择 BULK IN 端点，然后点击 **Transfer Data-IN** 按键。首先，将收到完整的数据包。再次点击 **Transfer Data-IN** 按键。现在，将收到 ZLP。

图 58. 连续执行 Transfer Data-IN 操作分别接收到的完整数据包和零长度数据包




7. 请注意，由于 FPGA 连续写入数据，所以可实现多次 BULK IN 传输。

## 12. 相关的项目文件

| 文件/文件夹名称 | | 说明 |
|---|---|---|
| FX3 固件 | | FX3 固件的源 |
| FPGA 源文件 | fx3_slaveFIFO2b_xilinx | 该文件夹包括下面的子文件夹：<br>*fpga_slavefifo2b_verilog*<br>Xilinx FPGA 源代码（Verilog），可支持所有传输类型（串流 IN、串流 OUT、短数据包、零长度数据包和回送）。<br>*Fpga_slavefifo2b_vhdl*<br>Xilinx FPGA 源代码（VHDL），可支持所有的传输类型（串流 IN、串流 OUT、短数据包、零长度数据包和回送）。 |
| | Fx3_slaveFIFO2b_altera | Verilog 和 VHDL 中的 Altera FPGA 源代码。<br>该文件夹包含针对每一种传输的子文件夹。下面的每个文件夹中均包含 Verilog 和 VHDL 项目：<br>*fpga_loopback*，用于回送数据传输，<br>*fpga_partial*，用于短数据包数据传输，<br>*fpga_streamIN*，用于串流 IN 数据传输，<br>*fpga_streamOUT*，用于串流 OUT 数据传输，<br>*fpga_zlp*，用于 ZLP 传输。 |
| SF_loopback.img | | 该文件是一个 FX3 固件镜像，包含了从设备 FIFO 实现，并用来设置回送传输所需要的 DMA 通道。<br>**注意：** 各个 FX3 固件镜像之间的唯一区别在于 DMA 通道的设置方式，以便能轻松地实现不同类型的传输。因此，可以使用任一固件镜像来执行任何传输类型。 |
| SF_streamIN.img | | 该文件是一个 FX3 固件镜像，包含了从设备 FIFO 实现。当执行串流 IN 传输时，它可设置性能优化所要求的 DMA 通道。 |
| SF_streamOUT.img | | 该文件是一个 FX3 固件镜像，包含了从设备 FIFO 实现。当执行串流 OUT 传输时，它可设置性能优化所要求的 DMA 通道。 |
| SF_shrt_ZLP.img | | 该文件是一个 FX3 固件镜像，包含了从设备 FIFO 实现。当执行短数据包或零长度数据包（ZLP）传输时，它可设置性能优化所要求的 DMA 通道。<br>当使用该镜像时，将看到传输完整数据包后再传输短数据包或零长度数据包的连续串流。 |
| TEST.txt | | 该文件包含了通过使用 Control Center 工具中 **Transfer File-OUT** 按键发送的数据图案。 |

## 13. 汇总

从设备 FIFO 接口非常合适外部 FPGA，处理器或器件需要对 EZ-USB FX3 的内部 FIFO 缓冲区进行数据读/写访问的应用。

本应用笔记详细说明了同步从设备 FIFO 接口。此外，还介绍了不同的标志配置以及使用 GPIF II Designer 工具配置标志的方式。文档中已提供了两个完整的设计示例。

# 附录A. 故障排除

**症状 1**

BULK IN 和 BULK OUT 数据传输操作失败。

```
BULK IN transfer
BULK IN transfer failed with Error Code:997
```

```
BULK OUT transfer
BULK OUT transfer failed with Error Code:997
```

**原因及解决方案**

连接至从设备 FIFO 接口的 FPGA 不发送连续的数据，或者接口时钟（PCLK）在较低频率下运行。在这种情况下，有可能 USB 链接在低功耗模式下停滞。

使用 **CyU3PUsbLPMDisable** 函数停止进入低功耗模式，如此处所示。

```
CyFxSlFifoApplnUSBEventCB (
    CyU3PusbEventType_t evtype,
    uint16_t            evdata
    )
{
    switch (evtype)
    {
        case CY_U3P_USB_EVENT_SETCONF:
            /* Stop the application before re-starting. */
            if (glIsApplnActive)
            {
                CyFxSlFifoApplnStop ();
            }
            CyU3PUsbLPMDisable();
            /* Start the loop back function. */
            CyFxSlFifoApplnStart ();
```

**注意**：该解决方案可能不与 USB 兼容。您可以采取 SDK 提供的 **USBBULKSourceSink** 示例中（浏览至“**CyU3PusbSetLinkPowerState（CyU3PUsbLPM_U0）**”）的其他解决方案。在最终固件中使用该解决方案或者联系赛普拉斯技术支持以得到帮助。

实现该解决方案后，如果 FX3 DVK 的 BULK IN 传输操作还失败，此时，请确保短接跳线器 J100 的 1 和 2 引脚。短接这些引脚后，FLAGA 可用于连接至从设备 FIFO 接口的 FPGA。

**症状 2**

即使已激活 PKTEND#信号，而不激活 SLWR#，但读取数据后获取不到 ZLP（读取的数据是数据包大小（USB 3.0 为 1024 字节，USB 2.0 为 512 字节）的倍数，且小于 FX3 的 DMA 缓冲区大小）。

**原因及解决方案**

假设将 DMA 缓冲区大小配置为 2 KB。FPGA 将 1 KB 数据写入到 DMA 缓冲区内，并要求将该数据传输到 USB 主机。USB 主机要求 FX3 的 2 KB 数据。

在这种情况下，FX3 需要发送一个 ZLP 和 1 KB 数据，以停止 USB 主机要求的传输操作。

当 FPGA 将数据写入从设备 FIFO 时，从设备 FIFO 接口的 GPIF II 状态机将处于“Write”状态。写入数据后（该数据为数据包大小的倍数），如要立即发送 ZLP，FPGA 在取消激活 SLWR#信号后需要在至少两个时钟周期激活 PKTEND#信号。这是因为 GPIF II 状态机需要两个时钟周期，以便从“Write”状态切换到“ZLP”状态。

**症状 3**

将固件下载到 FX3 后，FLAGA 立即变为低电平。

**原因及解决方案**

检查是否已成功建立 DMA 通道。如果与 GPIF 线程 0 相连的 DMA 通道失败，那么，与该线程相连的标志会指示错误状态。您可以检查 **CyU3PdmaChannelCreate** 函数的返回值，以确认是否成功建立 DMA 通道。

**CyU3PdmaChannelCreate** 函数失败主要是由于以下原因。在这两种情况下，该函数返回 CY_U3P_ERROR_MEMORY_ERROR 代码。

- DMA 缓冲区分配失败。将所有可用的（SYS MEM – CODE MEM）存储空间分配给 DMA 缓冲区。系统存储器（SYS MEM）取决于您正在使用的器件型号，并且 CODE MEM 取决于开发的应用代码。请确保 DMA 缓冲区总大小（缓冲区总数 * 每个 DMA 缓冲区大小）小于可用缓冲区空间（SYS MEM – CODE MEM）。
- DMA 缓冲区描述符分配失败：缓冲区描述符是用于跟踪每个 DMA 缓冲区的大小和状态的结构。该系统具有 512 个描述符。AUTO 通道的每一个缓冲区需要使用一个描述符，而 MANUAL 通道的每一个缓冲区则需要两个描述符。

DMA 缓冲区数量应该满足这些条件。

**症状 4**

当 FPGA 在 100 MHz 频率下通过 32 位数据总线将数据写入到 FX3 的从设备 FIFO 接口时，发生许多 DMA 溢出错误。

**原因及解决方案**

当使用 19.2 MHz 晶振或时钟源时，如果 FX3 主时钟被设置为 384 MHz，并且如果 GPIF II 被配置为 32 位宽度，并在 100 MHz 频率下运行，则会导致 GPIF II 上的 DMA 溢出错误。

**CyU3PsysClockConfig_t** 结构的 setSysClk400 参数指定 FX3 器件的主时钟是否被设置为高于 400 MHz 的频率。**CyU3PdeviceInit** 调用期间，设置该参数，可以将 FX3 的主时钟频率设置为 403.2 MHz。当在 **main** 函数中调用 **CyU3PdeviceInit** 时，该结构被视为一个参数。

```
/* setSysClk400 clock configurations */
        clkCfg.setSysClk400 = CyTrue;     /* FX3 device's master clock is set to a
frequency > 400 MHz */
        clkCfg.cpuClkDiv = 2;           /* CPU clock divider */
        clkCfg.dmaClkDiv = 2;           /* DMA clock divider */
        clkCfg.mmioClkDiv = 2;          /* MMIO clock divider */
        clkCfg.useStandbyClk = CyFalse; /* device has no 32-KHz clock supplied */
        clkCfg.clkSrc = CY_U3P_SYS_CLK; /* Clock source for a peripheral block  */

    /* Initialize the device */
    status = CyU3PdeviceInit (&clkCfg);
    if (status != CY_U3P_SUCCESS)
    {
        goto handle_fatal_error;
    }
```

**注意**：如果还未能解决您应用中的问题，请联系赛普拉斯技术支持。

# 附录B. 附录 B 用 FX3 开发套件（CYUSB3KIT-001）的硬件设置

图 59 显示了使用 FX3 开发套件（CYUSB3KIT-001）和 Xilinx SP601 电路板的固件设置。

图 59. 赛普拉斯 FX3 开发套件通过 FMC 互联电路板与 Xilinx SP601 电路板相连

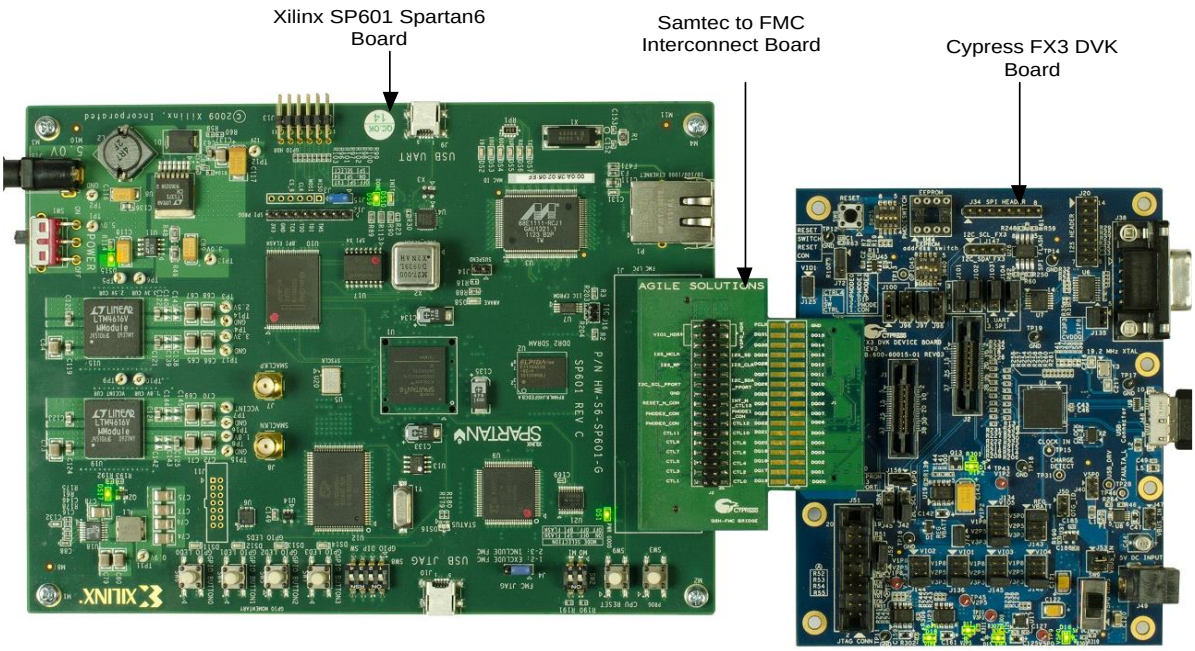
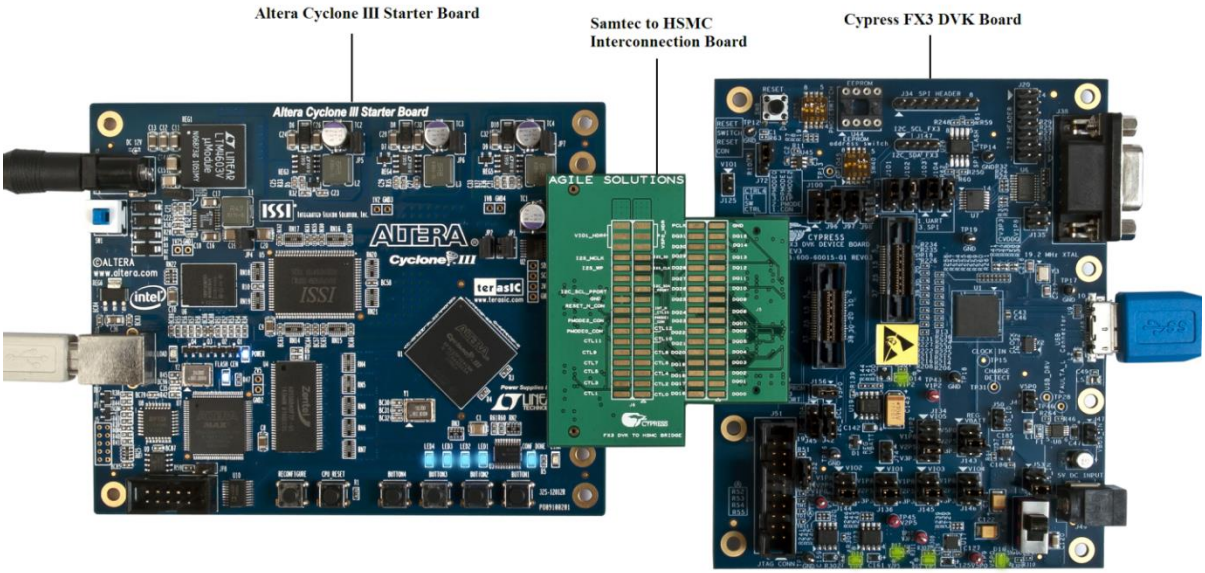



图 60 显示了使用 FX3 开发套件（CYUSB3KIT-001）和 Altera Cyclone III 电路板的固件设置。

图 60. 赛普拉斯 FX3 开发套件通过 HSMC 互联电路板与 Altera Cyclone III 电路板相连



## B.1 跳线器和开关设置

表 7 列出了执行演示的 FX3 DVK 板的跳线器和开关设置。这些设置用于连接到 FX3 DVK 的任何 FPGA 电路板。

表 7. FX3 DVK 跳线器和开关设置

| Sl. 序号 | 跳线器/开关 | 使用跳线器短接的引脚 | 功能 |
|---|---|---|---|
| 1 | J100 | 1 和 2 | GPIO[21]/CTL[4] — 配置为 FLAGA |
| 2 | J136 | 3 和 4 | VIO1（3.3 V） |
| 3 | J144 | 3 和 4 | VIO2（3.3 V） |
| 4 | J145 | 3 和 4 | VIO3（3.3 V） |
| 5 | J146 | 3 和 4 | VIO4（3.3 V） |
| 6 | J134 | 4 和 5 | VIO5（3.3 V） |
| 7 | J135 | 2 和 3 | CVDDQ（3.3 V） |
| 8 | J143 | 3 和 4 | VBATT（3.3 V） |
| 9 | J101 | 1 和 2 | GPIO[53] = UART_RTS |
| 10 | J102 | 1 和 2 | GPIO[54] = UART_CTS |
| 11 | J103 | 1 和 2 | GPIO[56] = UART_TX |
| 12 | J104 | 1 和 2 | GPIO[57] = UART_RX |
| 13 | J96 和 SW25 | 2 和 3 | 使用 SW25 选择 PMODE0 引脚状态（打开/关闭）。SW25.1 应关闭 |
| 14 | J97 和 SW25 | 2 和 3 | 使用 SW25 选择 PMODE1 引脚状态（打开/关闭）。SW25.1 应为关闭 |
| 15 | J98 | 1 和 2 | PMODE2 引脚处于悬空状态 |
| 16 | J72 | 1 和 2 | 复位 |
| 17 | J53 | 1–3 或 2–4 | 总线供电 |
| 18 | SW9 | 开关应转至标有 VBUS_IN 的方向。 | 总线供电 |
| 19 | J156 | 放置一个跳线器用于短接 | 为 Samtec 连接器供电 |
| 20 | J45 | 2 和 3 | GPIO[59] — 通过 FX3 将 FPGA 复位 |

**注意：** 设置 PMODE 引脚，以启动 USB。该表中不提及的跳线器可处于开路状态。

# 附录C. 附录 C

本部分介绍在 FX3 的 DMA 缓冲区大小和 USB 主机应用的传输缓冲区大小有所改变时从设备 FIFO 应用的两个数据传输模式（短数据包和 ZLP）运行的方式。

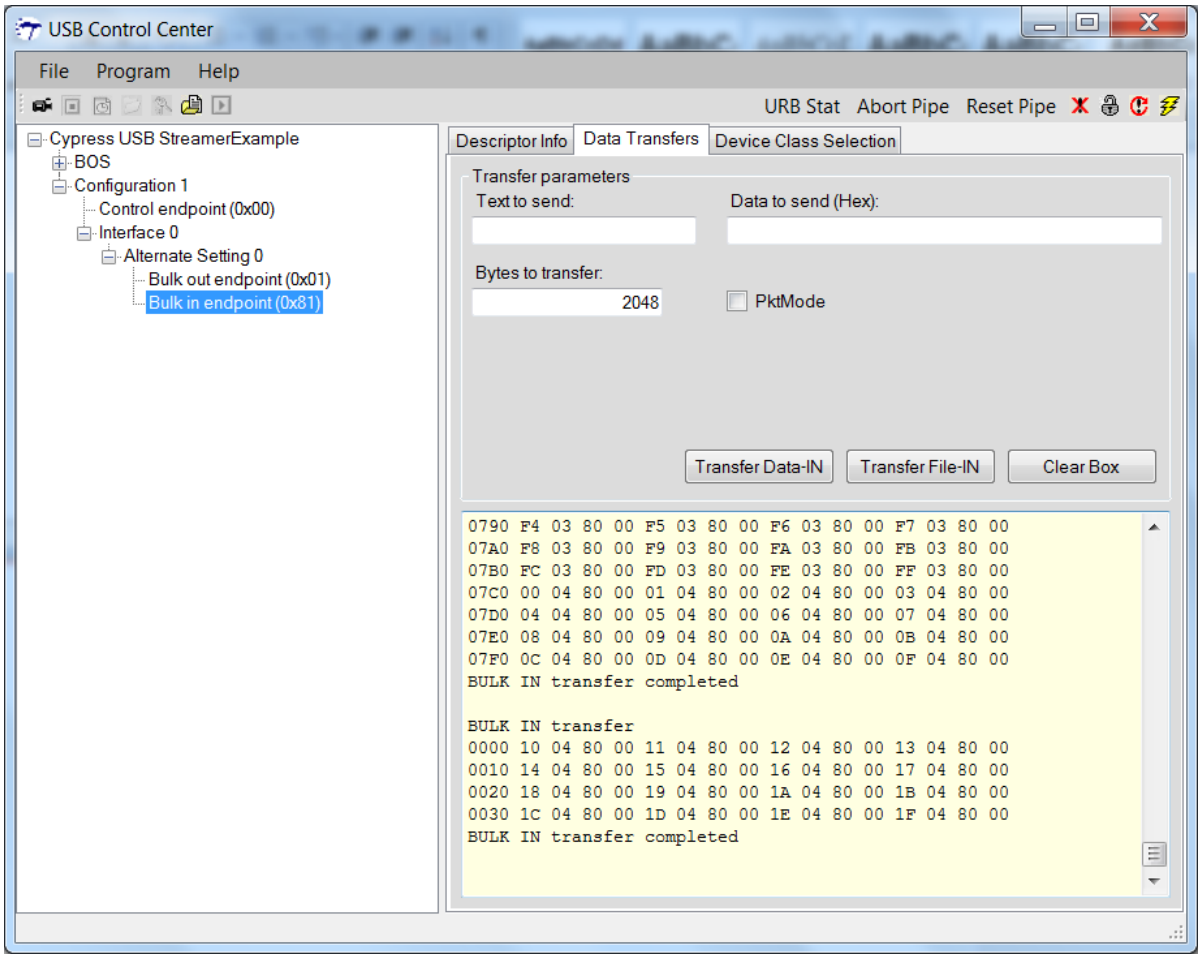
## C.1 短数据包示例

FX3 固件提供的 DMA 缓冲区大小为 1024 个字节，缓冲区数量为 2。在该示例中，FPGA 将 1024 字节数据写入到第一个 DMA 缓冲区内，然后，它将 64 字节数据写入到第二个 DMA 缓冲区内。

您可以在 USB Control Center 中将 1024 输入 **Bytes to transfer** 字段，然后点击 **Transfer Data-IN** 按键来读取该数据。将得到 1024 字节数据；如果重新点击同样的按键，您将得到由 FPGA 写入的下一个 64 字节数据。

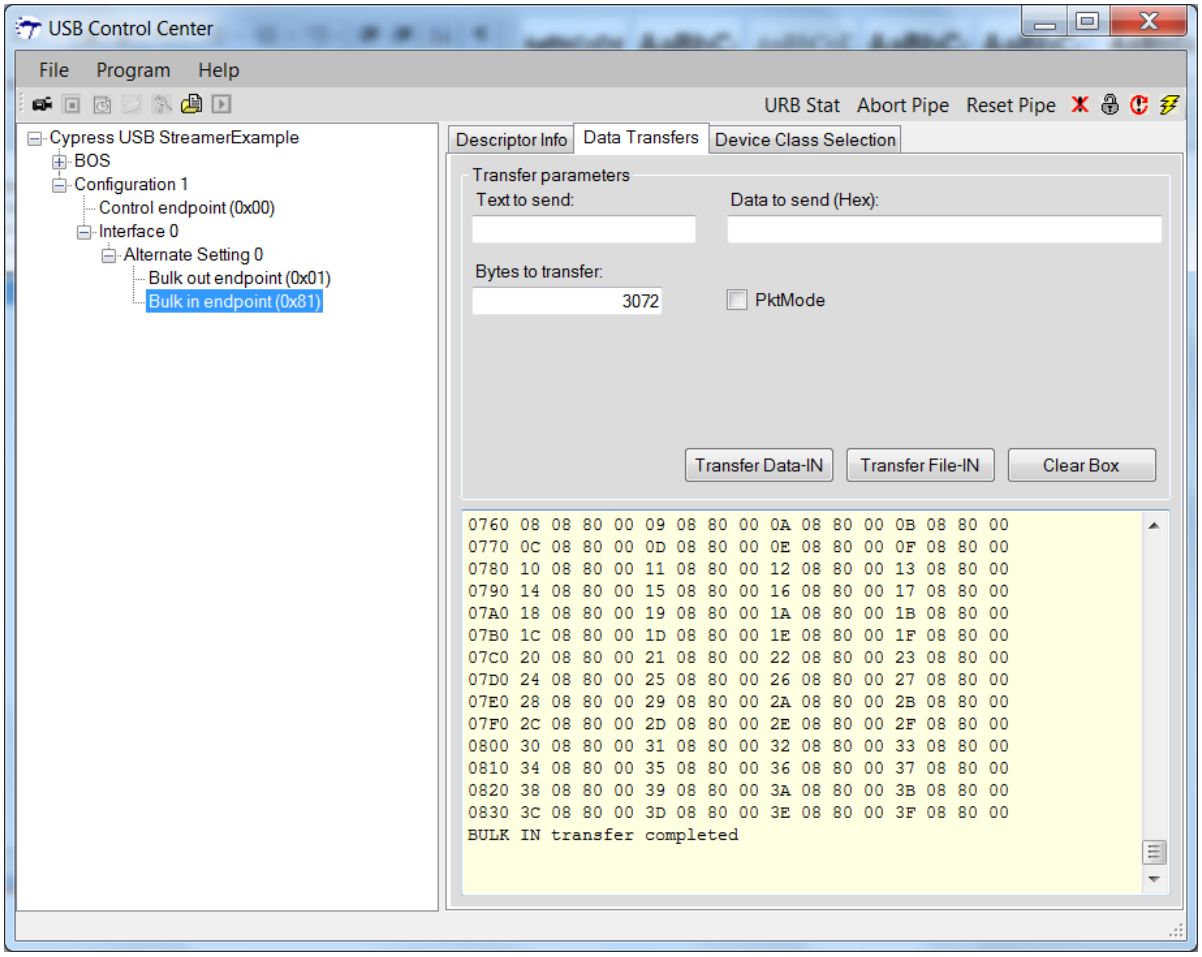
假设 DMA 缓冲区大小改为 2048 字节。FPGA 分别将 2048 字节数据以及 4 字节数据写入到第一个 DMA 缓冲区内和第二个 DMA 缓冲区内。FPGA 完全写满第一个 DMA 缓冲区后，就将一个短数据包写入到下一个可用的 DMA 缓冲区。如果分配多个 DMA 缓冲区，可重新实现该操作（写入一个满数据包和一个短数据包）。在 Control Center 中，可通过将 2048 输入 **Bytes to transfer** 字段，并点击 **Transfer Data-IN** 按键来读取数据；这样会提取 2048 字节数据。如果重新点击同样的按键，您将得到由 FPGA 写入的下一个 64 字节数据，如图 61 所示。

图 61. 使用 Control Center 读取短数据包




如果将传输缓冲区大小增到 3072 字节，每次您点击 **Transfer Data-IN** 按键会得到 2112（0x840）字节数据，如图 62 所示。

图 62. 需要的字节多于 DMA 缓冲区大小时的短数据包传输

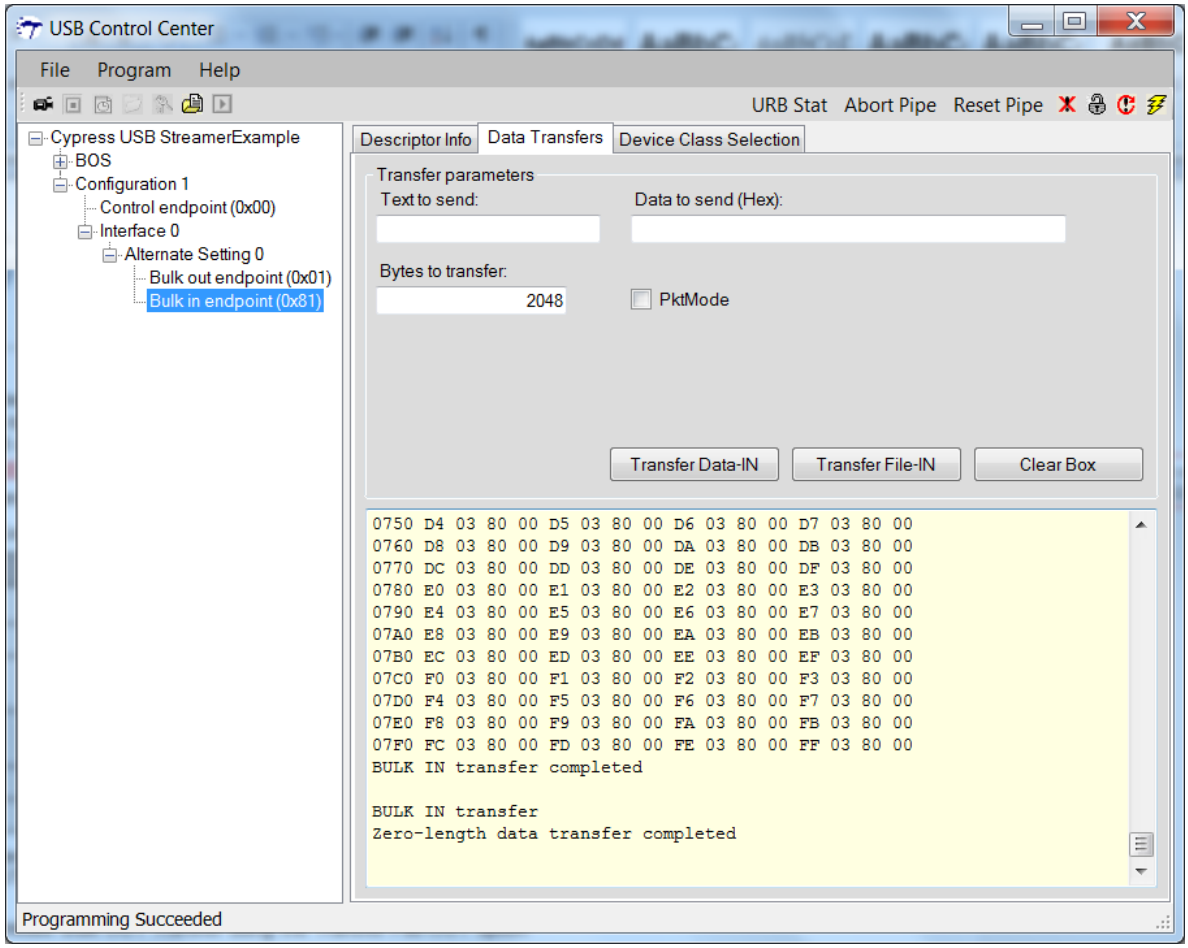



## C.2 零长度数据包（ZLP）示例

FX3 固件提供的 DMA 缓冲区大小为 1024 个字节，缓冲区的数量为 2。在该示例中，FPGA 写入 1024 字节数据，并激活用来生成 ZLP 的控制信号。您可以在 USB Control Center 中将 1024 输入 **Bytes to transfer** 字段，然后点击 **Transfer Data-IN** 按键来读取该数据。您将得到 1024 字节的数据，如果重新点击同样的按键，你会得到一个由 FPGA 写入的 ZLP。

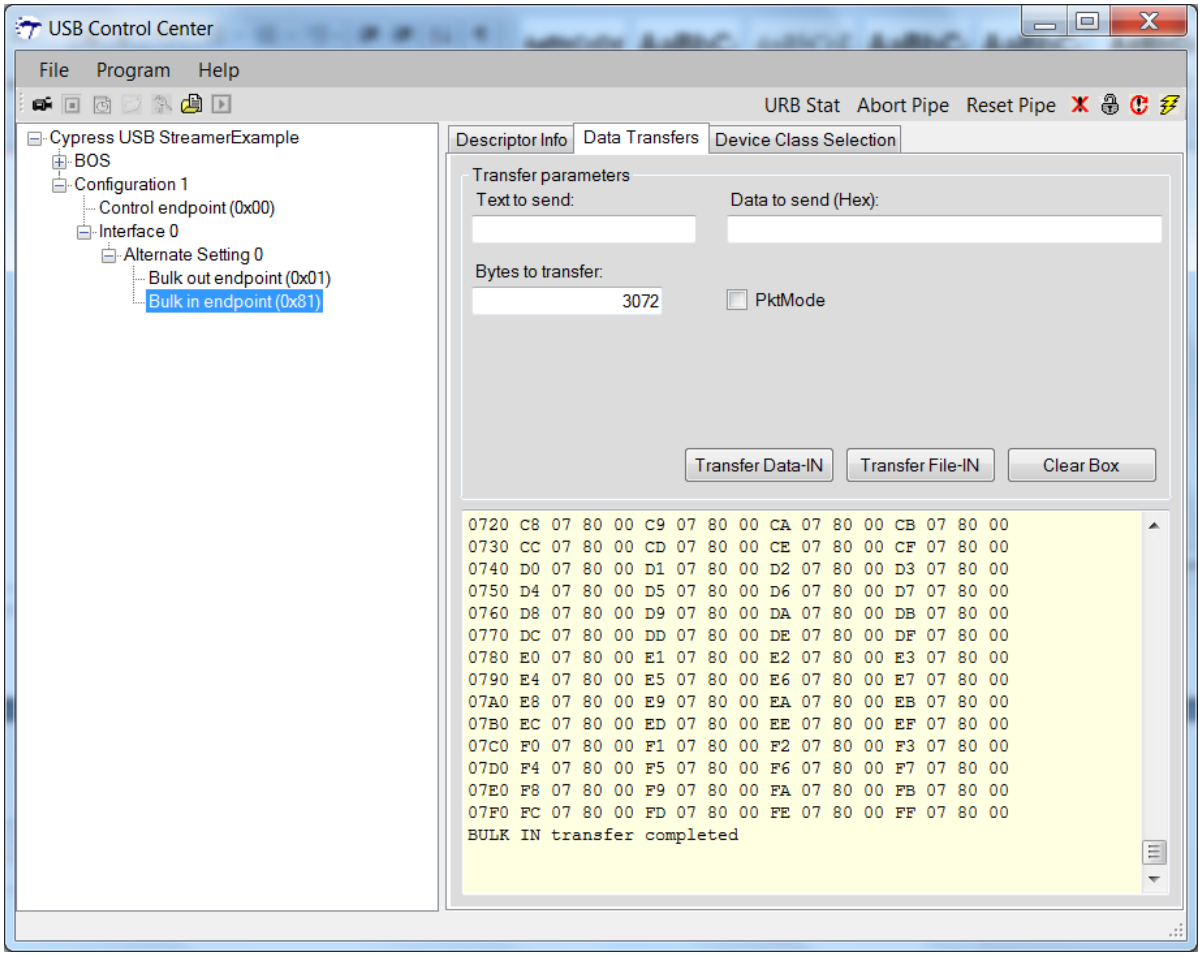
假设 DMA 缓冲区大小改为 2048 字节。FPGA 写入 2048 字节的数据，并激活用来生成一个 ZLP 的控制信号。FPGA 完全写满第一个 DMA 缓冲区，然后生成一个 ZLP。如果分配多个 DMA 缓冲区，可重新实现该操作（写入一个满数据包和一个 ZLP）。 在 Control Center 中，可通过将 2048 输入 **Bytes to transfer** 字段，并点击 **Transfer Data-IN** 按键来读取数据；这样会提取 2048 字节数据。如果重新点击同样的按键，您将得到一个由 FPGA 写入的 ZLP，如图 63 所示。

图 63. 使用 Control Center 读取 ZLP




如果将传输缓冲区大小增到 3072（大于 DMA 缓冲区大小），每次点击 **Transfer Data-IN** 按键，您将得到 2048（0x800）字节数据。发送 2048 字节数据后将发送一个 ZLP，但即使在物理 USB 总线上出现了一个 ZLP，这将不在 Control Center 中显示，如图 64 所示。

图 64. 需要的字节多于 DMA 缓冲区大小时的 ZLP 传输



# 文档修订记录

文档标题：AN65974 — 使用 EZ-USB® FX3™从设备 FIFO 接口进行设计
文档编号：001-92219

| 版本 | ECN | 变更者 | 提交日期 | 变更说明 |
|---|---|---|---|---|
| ** | 4354611 | GKL | 04/21/2014 | 本文档版本号为 Rev**，译自英文版 001-65974 Rev*I。 |
| *A | 4929869 | LIP | 09/28/2014 | 本文档版本号为 Rev*A，译自英文版 001-65974 Rev*L。 |
| *B | 5165296 | LIP | 02/29/2016 | 本文档版本号为 Rev*B，译自英文版 001-65974 Rev*M。 |
| *C | 5708841 | AESATP12 | 04/26/2017 | Updated logo and copyright |
| *D | 6502184 | SSAS | 03/06/2019 | 本文档版本号为 Rev*D，译自英文版 001-65974 Rev*N。 |

# 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、原厂代表和经销商组成的全球性网络。如欲查找离您最近的办事处，请访问赛普拉斯所在地。

## 产品

| | |
|---|---|
| Arm® Cortex®微控制器 | cypress.com/arm |
| 汽车级产品 | cypress.com/automotive |
| 时钟与缓冲器 | cypress.com/clocks |
| 接口 | cypress.com/interface |
| 物联网 | cypress.com/iot |
| 存储器 | cypress.com/memory |
| 微控制器 | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| 电源管理 IC | cypress.com/pmic |
| 触摸感应 | cypress.com/touch |
| USB 控制器 | cypress.com/usb |
| 无线连接 | cypress.com/wireless |

## PSoC®解决方案

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

## 赛普拉斯开发者社区

社区 | 项目 | 视频 | 博客 | 培训 | 组件

## 技术支持

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709