AN61345

使用 EZ-USB® FX2LP™从设备 FIFO 接口进行设计

作者：Rama Sai Krishna V
相关工程：有
关联器件系列：CY7C68013A/CY7C68014A/CY7C68015A
软件版本：无
相关应用笔记：AN65209、 AN63620

想要更多例程吗？

获取更多完整的程序样例,请访问 here.

AN61345 提供了一个示例项目，用以通过从设备 FIFO 接口将 FX2LP™连接至 FPGA。示例实现中描述的接口为各个应用执行高速度的 USB 连接事项，如数据采集、工业控制和监控以及图像处理。本应用笔记提供的项目通过使用 Xilinx® Spartan® 6 FPGA 来实现和测试。

目录

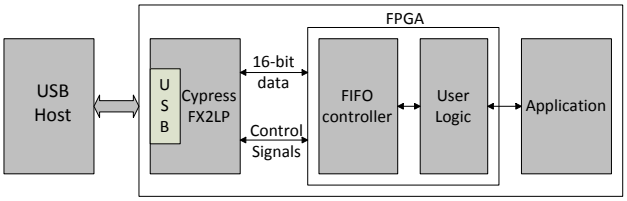
简介 1
硬件连接 2
固件实现 2
FX2LP 代码架构 2
FPGA 代码架构 3
数据回送 3
Stream IN（流入）传输 4
Stream OUT（流出）传输 4
仿真波形 5
设计示例 6
ZTEX 硬件设置 6
固件和软件组件 8
操作流程 10
吞吐量测量 16
关联项目文件 17
如何导入该设计，以便与 Altera® FPGA 一起工作 17
总结 17
全球销售和设计支持 19

简介

赛普拉斯 EZ-USB FX2LP 是一个灵活的 USB 2.0 外设控制器，用于处理 USB 2.0 的最大宽度。为充分利用 USB 2.0 中 480 每秒兆位的通信速率，FX2LP 包含了一个专用的硬件，用来缓冲 USB 数据，并与各种高带宽的外部设备（如 MCU、ASIC 和 FPGA）无缝连接。

FX2LP-FPGA 接口为基于 FPGA 的应用执行高速度的 USB 连接事项，如数据采集、工业控制和监控以及图像处理。FX2LP 在同步从设备 FIFO 模式下运行，FPGA 作为主设备使用。本应用笔记还为从设备 FIFO 实现提供了示例 FX2LP 固件，并为 FPGA 实现提供了示例 VHDL 和 Verilog 项目。

图 1. FX2LP-FPGA 系统



可以通过两个不同的模式将 FX2LP 连接至 FPGA。这两个模式分别为通用可编程接口（GPIF）模式和从设备 FIFO 模式。

- **GPIF 模式：**在该模式下，FX2LP 作为外部系统的主设备使用，它所生成的所有控制信号用于对外部系统进行读和写操作。当外部系统不能作为 FX2LP 的主设备（例如，图像传感器与 FX2LP 相连接的 USB 摄像机应

用）时，通常优先使用 GPIF 模式。在这种情况下，接口实现的复杂操作将由 FX2LP 执行。

- **从设备 FIFO 模式：**在该模式下，连接至 FX2LP 的外部系统能够生成读和写控制信号，因此，它能作为 FX2LP 的主设备使用。在本应用中，FX2LP 被配置为从设备 FIFO 模式。

本应用笔记描述了在 FX2LP 上 16 位从设备 FIFO 的实现，同时还包含了 Verilog 和 VHDL 示例项目，以介绍如何将一个外部 FPGA 连接至 FX2LP 的从设备 FIFO 接口。

注意：各个示例项目是在 Xilinx Spartan 6 FPGA 中实现并测试的。本应用笔记所提供的代码是标准的 Verilog/VHDL 代码。因此，在任何 FPGA 上实现项目时，可以将这些文件作为参考源使用。在合成和实现项目时，需要选择正确的 FPGA 器件。

在这里，假设您已经熟悉从设备 FIFO 接口、Verilog/VHDL 编码、FPGA 合成和实现工具。请参阅 EZ-USB 技术参考手册的第 9 章（从设备 FIFO）。

硬件连接

下表显示的是将 FX2LP 连接至 FPGA 时所需要的硬件连接。

图 2. 硬件连接框图

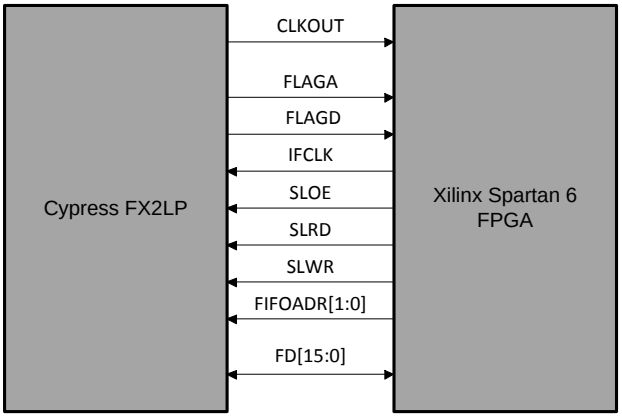


表 1 描述了图 2 中所显示的从设备 FIFO 接口信号。

表 1. FX2LP 和 FPGA 间的接口信号

引脚名称	说明
SLRD	SLRD 引脚应由主设备激活，用以从 FIFO 读取数据。
SLWR	SLWR 引脚应该由主设备激活，以将数据写入到 FIFO 内。
SLOE	是指 FIFO 输出驱动器的使能信号。
FIFOADR[1:0]	这些信号用于选择有效的端点。
FD[15:0]	16 位数据总线
FLAGA/FLAGB/ FLAGC/FLAGD	FIFO 使用这些标志来表示各种状态（满、空、可编程）。
IFCLK	是指与从设备 FIFO 接口同步的时钟。在本应用笔记所提供的设计中，该时钟的频率被配置为 48 MHz，并由连接至 FX2LP 的 FPGA 生成。
CLKOUT	FX2LP 的 CLKOUT 引脚可以提供的时钟频率分别为 12、24 或 48 MHz。

固件实现

通过使用 Keil uVision 2.0 集成开发环境来开发 FX2LP 固件。在 FX2LP DVK（CY3684）内容中提供了集成开发环境的评估版本。本节所介绍的是在从设备（FX2LP）和主设备端（FPGA）上实现从设备 FIFO 接口时所需要的配置。

FX2LP 代码架构

固件配置了 IN 和 OUT 端点 FIFO 的自动模式。这样，对于 IN 传输，数据包将从外部设备自动被发送到 USB 端；对于 OUT 传输，则反向传送。传送数据包时，无需 8051 CPU 干预。更多有关在自动或手动模式下端点 FIFO 配置的细节，请参考 EZ-USB 技术参考手册中的从设备 FIFO 章节。由于本应用中使用的批量（bulk）传输，您需要将端点配置为“Bulk”。但在 USB 描述符文件中，根据终端应用，您可以将端点类型配置为 Interrupt（中断）、Control（控制）或 Isochronous（同步）。

由于从设备运行在自动模式下，因此除了以下寄存器的初始化外，将数据传入/传出主设备时，无需任何代码（表 2）。

表 2. 从设备 FIFO 配置寄存器

寄存器名称	寄存器说明
IFCONFIG	配置 IFCONFIG 寄存器，以使 FX2LP 进入从设备 FIFO 模式
PINFLAGSAB/ PINFLAGSCD	配置 FIFO 标志。可以将 FIFO 标志配置为固定模式或索引模式。在索引模式下，FLAGA、FLAGB 和 FLAGC 被自动分别配置为表示空、满和可编程的标志。在特定时间点，这些标志用于 FIFOADR [1:0]行所指向的端点。在固定模式中，通过对可编程标志寄存器进行写操作，可将四个标志中的任何一个配置为空、满或可编程标志。在该设计中，FLAGA 被配置为 EP2 OUT FIFO 的空标志，FLAGD 被配置为 EP6 IN FIFO 的满标志。
EP2CFG/ EP6CFG	将 EP2 配置为 OUT、512 字节的四倍缓存端点，并将 EP6 配置为 IN、512 字节的四倍缓存端点。
EP2FIFOCFG/ EP6FIFOCFG	配置 FIFO，使之以字节格式传输数据，并运行于自动模式。
EP6AUTOINLENH/ EP6AUTOINLENL	当 EP6 FIFO 中的字节数值等于这些寄存器指定的数值时，将自动传送数据包。要求指定的数值不大于端点描述符所指定的最大数据包大小。

FPGA 代码架构

FPGA 代码的主要功能是监控从设备 FIFO 的满标志和空标志，然后分别对 FIFO 进行读和写操作。

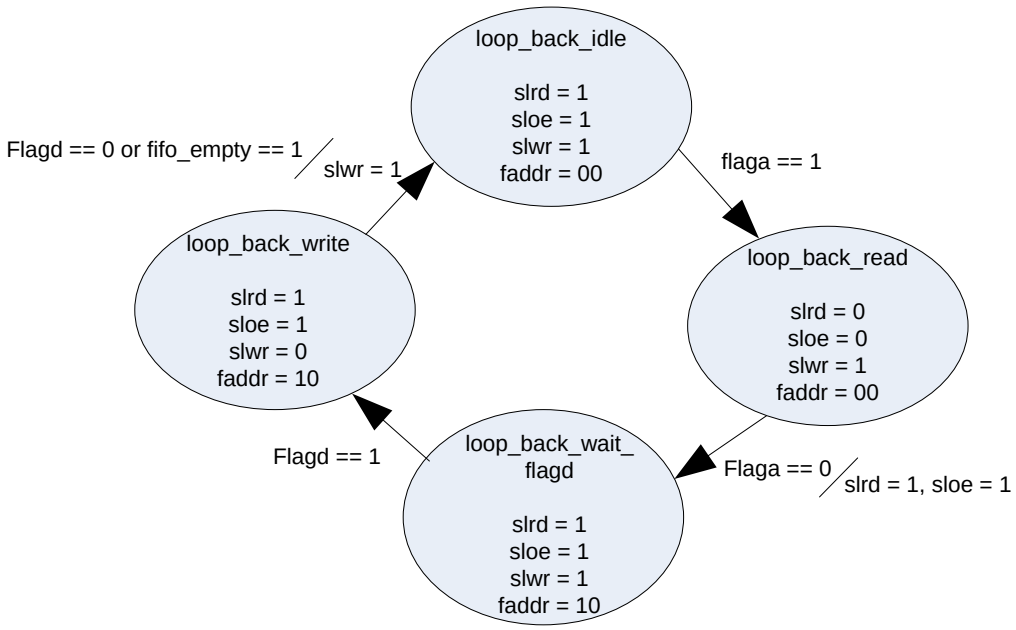
来自 FPGA 的接口时钟（IFCLK）需要转动 180 度，以满足 FX2LP 的从设备 FIFO 接口建立时间的要求。

实现 Verilog 和 VHDL 项目，以说明如何将 FPGA 配置为 FX2LP 从设备 FIFO 的主设备。Xilinx ISE 设计套件用于代码开发。

数据回送

FPGA 对写入到 FX2LP 的 EP2 FIFO 中的所有数据进行读取，然后将这些数据写入到 EP6 FIFO 内。（本应用笔记附上的）关联项目文件夹 *Loopback* 中包含了该项目。

图 3. 回送状态框图



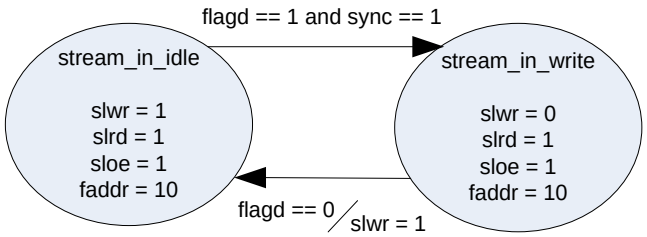
loop_back_idle 状态指的是 SLRD、SLOE 和 SLWR 信号均为低电平（取消激活）时的闲置状态。当 EP2 FIFO 空标志（FLAGA）变为高电平时，状态机将变为 loop_back_read 状态。在 loop_back_read 状态中，FPGA 通过激活 SLRD 和 SLOE 来读取数据。当 EP2 FIFO 空标志变为低电平时，状态机将进入 loop_back_wait_flagd 状态。在转换过程中，会取消激活 SLRD 和 SLOE 信号。在 loop_back_wait_flagd 状态中，FIFO 地址行被驱动到地址

EP6。它会一直处于该状态，直到 EP6 满标志（FLAGD）变为低电平为止。当 FLAGD 变为高电平时，状态机将转到 loop_back_write 状态。在该状态中，通过激活 SLWR 信号，FPGA 将相同的数据写入到 EP6 FIFO 内。

Stream IN（流入）传输

FPGA 监控 EP6 的满标志（FLAGD）和同步（FX2LP 的 PC0）信号。当 FLAGD 和同步信号均为高电平时，FPGA 会连续将递增数据写入到 FIFO 内。当写入到 EP6 FIFO 时，如果激活满标志，FPGA 将暂停写操作。如果取消激活该标志，则 FPGA 将恢复该写操作。关联项目文件夹（*Stream IN*）中包含了该项目。

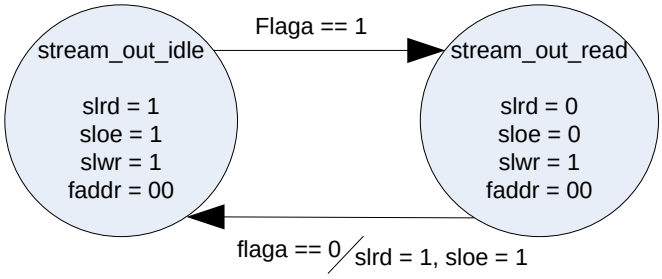
图 4. 数据 Stream IN 状态框图



该状态机中包括两个不同的状态：stream_in_idle 和 stream_in_write。stream_in_idle 是指 SLWR 处在高电平时的闲置状态。只要 EP6 满标志（FLAGD）为低电平（被激活），则状态机会处于 stream_in_idle 状态。在满标志和同步（FX2LP 的 PC0）信号变为低电平时，状态机将从 stream_in_idle 状态进入 stream_in_write 状态。在 stream_in_write 状态中，FPGA 连续将递增数据写入到 EP6 FIFO 内。当 FLAGD 变为低电平时，状态机将返回到 stream_in_idle 状态。在该转换过程中，会取消激活 SLWR 信号。

Stream OUT（流出）传输

图 5. Stream OUT 状态图



该状态机中包括两个不同的状态：stream_out_idle 和 stream_out_read。stream_out_idle 是指 SLRD 和 SLOE 均为高电平时的闲置状态。只要 EP2 空标志（FLAGA）为低电平（被激活）状态机将处在 stream_out_idle 状态。EP2 空标志变为高电平后，状态机将从 stream_out_idle 状态转到 stream_out_read 状态。在 stream_out_read 状态中，FPGA 将连续从 EP2 FIFO 中读取数据。当 FLAGA 变为低电平时，状态机将返回 stream_in_idle 状态。在转换过程中，将取消激活 SLRD 和 SLOE 信号。关联项目文件夹（*Stream OUT*）中包含了该项目。

仿真波形

本节介绍的是在不同模式下（Stream IN 和 Stream OUT）从设备 FIFO 接口信号的仿真波形。通过使用 Xilinx ISim 工具捕获这些波形。

图 6. 数据 Stream IN — 开始连续写入时的波形

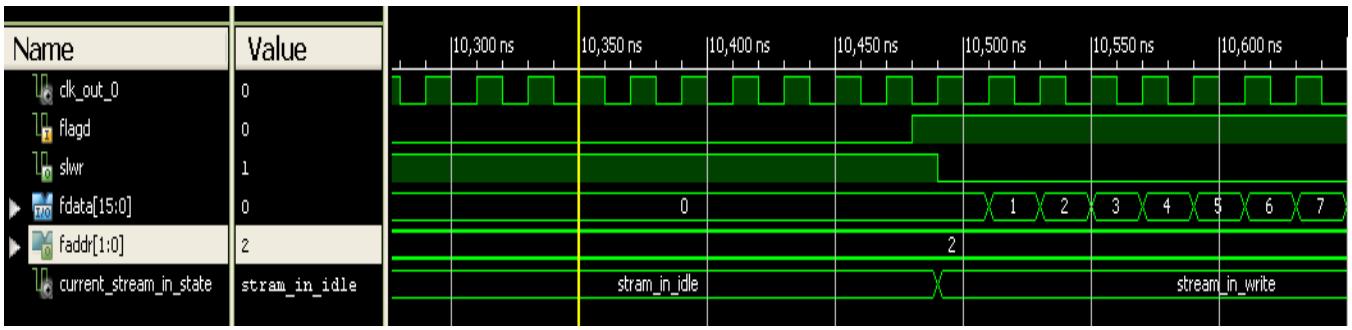
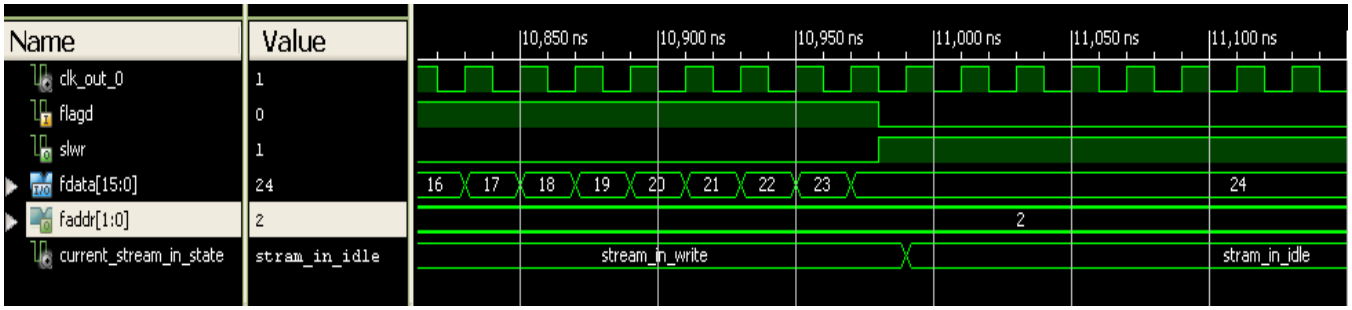


图 6 显示的是取消激活 FLAGD（即满标志）时 SLWR 的激活情况。

图 7. 数据 Stream IN — 结束连续写入



激活满标志后，将取消激活 SLWR，如图 7 所示。

图 8. 数据 Stream OUT — 开始连续读取

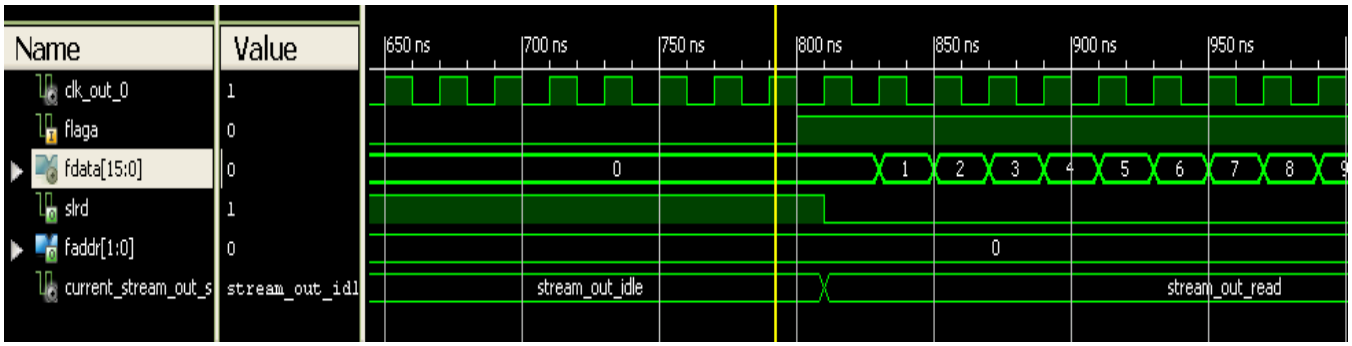
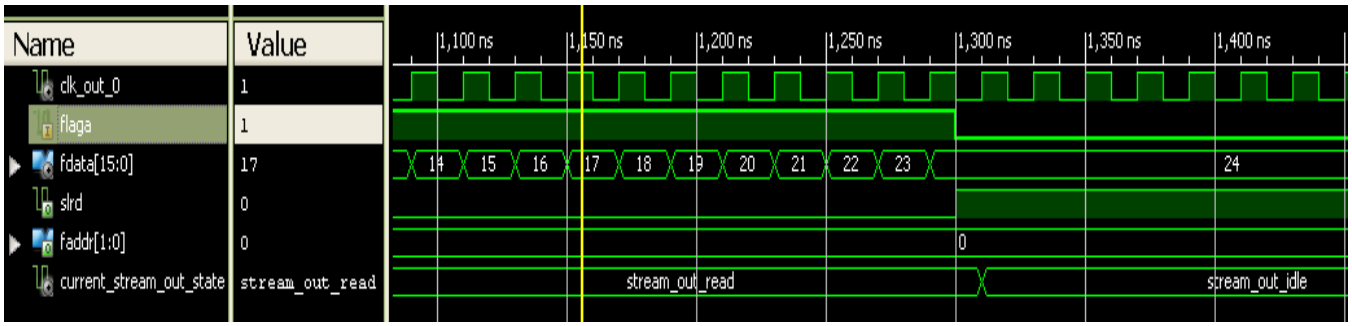


图 8 显示的是取消激活 FLAGA（EP2 EF）时 SLRD 的激活情况。

图 9. 数据 Stream OUT — 结束连续读取



在激活 FLAGA 时，将取消激活 SLRD 信号，如图 9 所示。

设计示例

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 FX2LP。

本应用笔记中提供了一个与 Spartan-6 相兼容的示例。该示例实现了 stream IN、stream OUT 和回送传输。FPGA 代码架构一节解释了 FPGA 位字段的行为。

以下各节还描述了用于实现该设计的硬件、固件及软件。

ZTEX 硬件设置

结合使用 ZTEX FX2LP - FPGA 模块 1.11（如图 11 所示）和 实验电路板 1.3（如图 12 所示）。实验电路板需要一个电源供应和一个 JTAG 线缆（用于配置 FPGA）。CON1（位于实验电路板 1.3 上）是一个标准的直流电源插头。其中心引脚(+)直径为 2.1 mm，柱形(-)直径为 5.5 mm，用于提供范围为 4.5 V 到 16 V 的电源电压。CON9（位于实验电路板 1.3 上）是一个由 Xilinx 标准化的 14 引脚、2.0 mm 间距的 JTAG 连接器。模块中和实验电路板上都有一个名称为‘1’的极化键（孔）。需要在实验电路板 1.3 上安装 ZTEX FX2LP - FPGA 模块 1.11，以便使两者的极化孔位于同一个角。如要确定这两个电路板上的极化孔，请参考上述链接的布局图。

在实验电路板 1.3 上安装 FPGA 模块 1.11，如图 13 所示。使用 5V 或 12V 的电源为电路板供电，并使用微型 USB 线缆将该板连接至主机 PC。

平台 USB 线缆 II（JTAG 适配器）用于配置 ZTEX FX2LP - FPGA 模块 1.11 中的 Xilinx Spartan-6 FPGA。‘Chipscope Pro’（下面步骤中进行介绍）是与该 JTAG 适配器相兼容的软件。

图 10 显示的是硬件连接框图：

图 10. 硬件连接（Ztex 电路板）

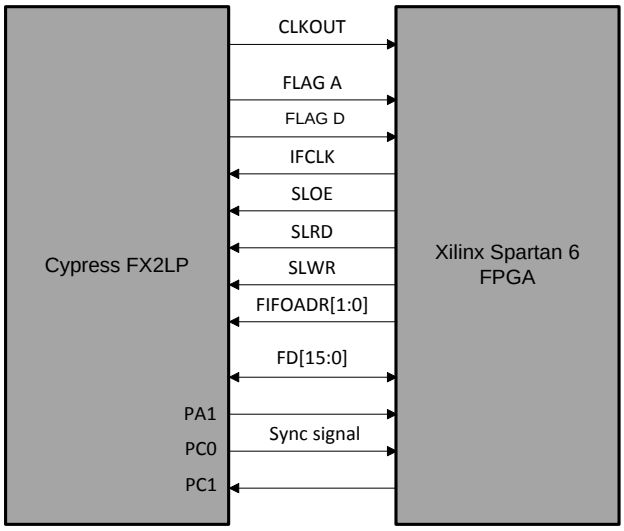


图 11. Ztex FX2LP – FPGA 模块 1.11



图 12. Ztex 实验电路板 1.3

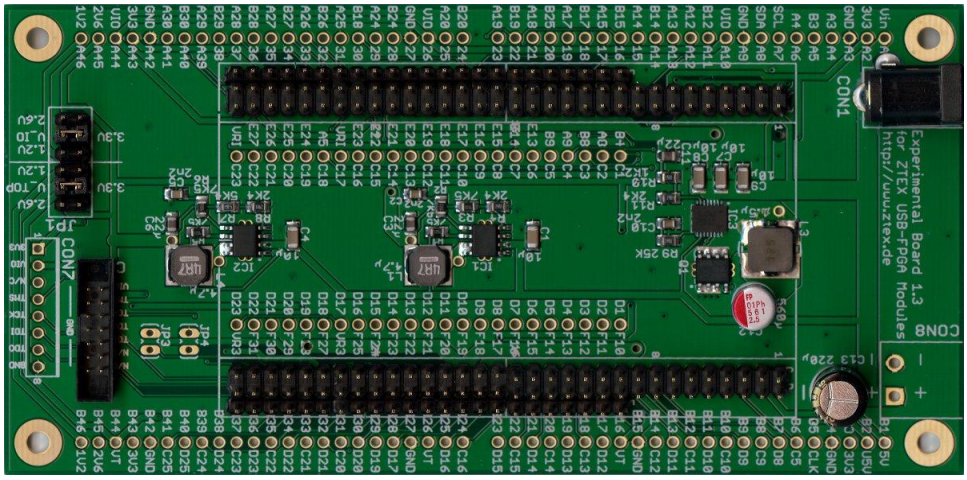
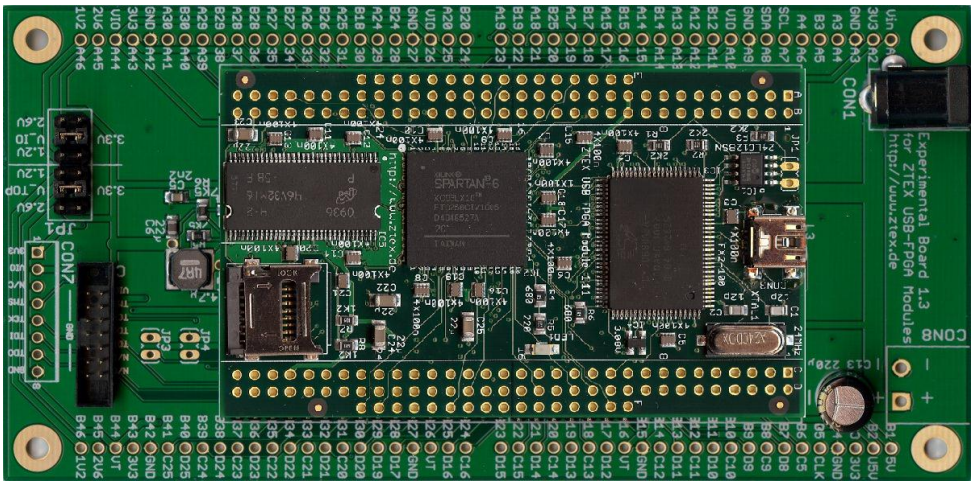


图 13. 位于实验电路板顶层上的 Ztex FX2LP-FPGA 模块

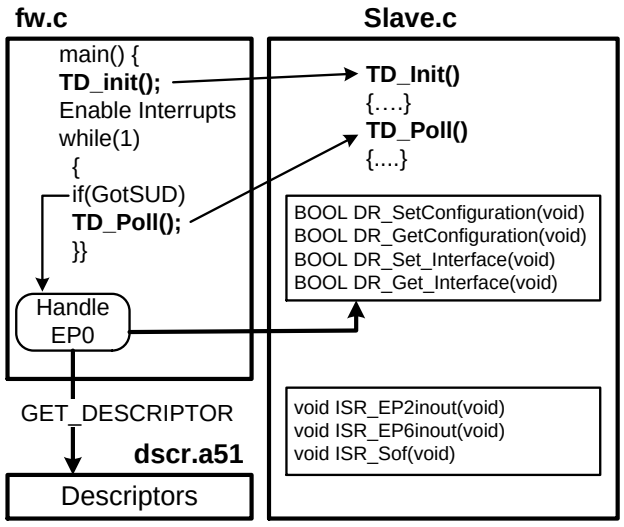


固件和软件组件

1. FX2LP 固件

图 14 显示的是各个 FX2LP 固件模块的组合情况。

图 14. 从设备 FIFO 接口的 FX2LP 固件



Fw.c 文件包含 **main** 函数。它执行了 USB 维持的大部分操作（如进行枚举），并且每当需要自定义时，它将调用应用代码（*Slave.c*）中特定名称的外部函数。一般情况下，您不需要修改 *Fw.c* 文件。执行各个日常操作的步骤后，该函数将调用 *Slave.c* 所提供的外部函数，即 TD_init。（前缀 TD 表示“任务调度”。然后，它进入一个无限循环，以通过 CONTROL 端点 0 检查 SETUP 数据包的到来。该循环还会检查 USB 暂停事件，但从设备 FIFO 应用不会使用该循环。每次进入该循环时，该函数都将调用 *Slave.c* 文件中提供的外部函数 TD_Poll。 在本应用中，TD_Poll 函数用于同步化 FPGA 和 FX2LP 间所传输的数据。开始传输数据时，由于 FIFO 被配置为自动模式，因此该函数不会进行任何操作。

每个 USB 外设通过它们的 CONTROL 端点接收两个请求类型：枚举和操作。

枚举

当与 USB 器件连接时，主机 PC 将发送多个 GET_DESCRIPTOR 请求以确定器件类型及其要求。这些操作属于枚举过程的一部分。*fw.c* 代码截取这些请求，并通过使用 *dscr.a51* 文件中所存储的数值处理请求。

使用 USB 框架的优势是已经测试和验证过代码，并通过 USB“第 9 章”中的要求。第 9 章是指 USB 规范中用于处理器件请求（通过 EP0）及其正确响应的章节。

操作

需要用户代码时，fw.c 将调用一个带有特定名称前缀为 DR（器件请求）的外部函数（存储在 *Slave.c* 文件中）。对于从设备 FIFO 这种简单的应用，只会使用一个配置和一个借口。因此，图 14 中所显示的两对 DR_Set-Get 函数只存储由主机发送的“Set”值，并在主机发出“Get”请求时对该值进行随路。对于更加复杂的配置，您可以使用这些 DR 调用（“hooks”）更改摄像机的分辨率或将请求路由到两个不同的接口等。

本节剩下的内容描述了该文件的三个部分，这三个部分要求用户代码实现从设备 FIFO 应用。

TD_Init

该函数执行以下操作：

- 将 8051 时钟频率设置为 48 MHz。
- 对从设备 FIFO 接口进行配置，使之使用 48 MHz 大小的内部时钟。

```
IFCONFIG = 0xE3; //Internal clock, 48
MHz, Slave FIFO interface
SYNCDELAY;
```

- 将 EP2 配置为 BULK-OUT 端点，并将 EP6 配置为 BULK-IN 端点。该两个端点均为四倍缓冲，并使用 512 字节的 FIFO。由于本设计中没有使用 EP4 和 EP8，所以它们均被取消激活。

```
EP2CFG = 0xA0; //out 512 bytes, 4x,
bulk
SYNCDELAY;
EP6CFG = 0xE0; //in 512 bytes, 4x,
bulk
SYNCDELAY;
EP4CFG = 0x02; //clear valid bit
SYNCDELAY;
EP8CFG = 0x02; //clear valid bit
SYNCDELAY;
```

- 复位 FIFO。

- 分别将端点 2 FIFO 和端点 6 配置为自动输出模式和自动输入模式，同时使用 16 位接口。

```
EP2FIFOCFG = 0x00;// AUTOOUT=0,
WORDWIDE=1
// core needs to see AUTOOUT=0 to
AUTOOUT=1 switch to arm endpoints
SYNCDELAY;
EP2FIFOCFG = 0x11;// AUTOOUT=1,
WORDWIDE=1
SYNCDELAY;
EP6FIFOCFG = 0x0D;// AUTOIN=1,
ZEROLENIN=1, WORDWIDE=1
SYNCDELAY;
```

- 配置 FIFO 标志输出。FLAGA 被配置为 EP2 OUT FIFO 的空标志，FLAGD 被配置为 EP6 IN FIFO 的满标志。

```
PINFLAGSAB = 0x08; // FLAGA - EP2EF
SYNCDELAY;
PINFLAGSCD = 0xE0; // FLAGD - EP6FF
SYNCDELAY;
```

- 将 PA1 引脚（连接至 FPGA 的 PROG_B 引脚）置为高电平。为使能 FPGA 的 JTAG 配置，需要进行上述设置。如果结合使用了 FX2LP 固件和 ZTEX 硬件设置，请确保将 FX2LP 的 PA1 引脚置于高电平。

```
OEA|=0x02; //Declare PA.1 as output
SYNCDELAY;
IOA|=0x02; //output 1 on PA.1
SYNCDELAY;
```

- 将 PC0 引脚配置为输出，并将 PC1 引脚配置为输出。PC0 用于同步化 FPGA 和 FX2LP 之间传输的数据。PC1 用于确定 FPGA 是否准备好提供从设备 FIFO 接口时钟（IFCLK）。

```
OEC|=0x01; //PC.0 as output (SYNC
signal)
SYNCDELAY;
IOC|=0x00; //output 0 on PC.0...SYNC
signal is LOW
SYNCDELAY;
OEC&=0xFD; //PC.1 as input (Clock
changing signal)
SYNCDELAY;
```

TD_Poll

在 *fw.c* 文件的无限循环中调用了 **TD_Poll**（图 14）。在这个从设备 FIFO 设计中，TD_Poll 用于更改接口时钟（IFCLK）源。

本应用笔记中的示例项目将使用来自 FPGA 的接口时钟（IFCLK）。如果您使用了 ZTEX 硬件电路板，请先编程 FX2LP，用以将 FPGA 的 PROG_B 引脚（连接至 PA1）设置为高电平，并对该引脚进行配置。在对 FPGA 进行配置，使之提供内部时钟前，FX2LP 固件不能通过配置 IFCONFIG 寄存器来使用外部引脚（注意，在固件将 IFCONFIG.7 设置为 0 前，必须存在外部 IFCLK 源（IFCLK 由外部器件提供））。为满足该条件，应首先配置 IFCONFIG 寄存器，以使用内部 IFCLK。然后，在 FPGA 启动并以位流的方式运行时，才能通过更改 IFCONFIG.7 获取来自 FPGA 的 IFCLK。FX2LP 的 PC1 用于进行该更改。FX2LP 的 PC0 作为 FX2LP 和 FPGA 间的同步信号使用。使用以下代码，可以将时钟源从内部源改为外部源。

```
if(!(IOC & 0x02))
      {
      done_frm_fpga = 1;
      }
if((done_frm_fpga) && (IOC & 0x02))
      {
      IFCONFIG = 0x03; //external
clock input, Slave FIFO interface
      SYNCDELAY;

      IOC|=0x01; //output 1 on
PC.0...SYNC signal is HIGH
      SYNCDELAY;
      done_frm_fpga = 0;
      }
```

FX2LP 固件是本应用笔记的附录部分。在将位流下载到 FPGA 前，请通过使用 Control Center 工具编译该固件，并将其下载到 FX2LP 中。操作流程一节说明了这些步骤。

2. Control Center 工具

使用赛普拉斯 Control Center 工具下载固件，并对 FX2LP 进行 BULK 传输。 通过安装赛普拉斯的 SuiteUSB 开发工具，可以获取 Control Center。

3. Chipscope Pro

‘ChipScope Pro’软件（与 Xilinx ISE 设计套件 14.1 一同提供）用于配置 Xilinx Spartan-6 FPGA。在 30 天内，可以试用 Xilinx ISE 设计套件 14.1 的评估版本，无需许可证。您也可以采用配置 Xilinx Spartan-6 FPGA 的其他任何有效的方式。下一节将描述如何使用 Chipscope Pro 配置 Xilinx Spartan-6 FPGA 的各个步骤。

如果使用‘Chipscope Pro’软件的其他任何版本，请确保该版本支持 Xilinx Spartan-6 FPGA。

操作流程

1. 分配 CYUSB 驱动程序

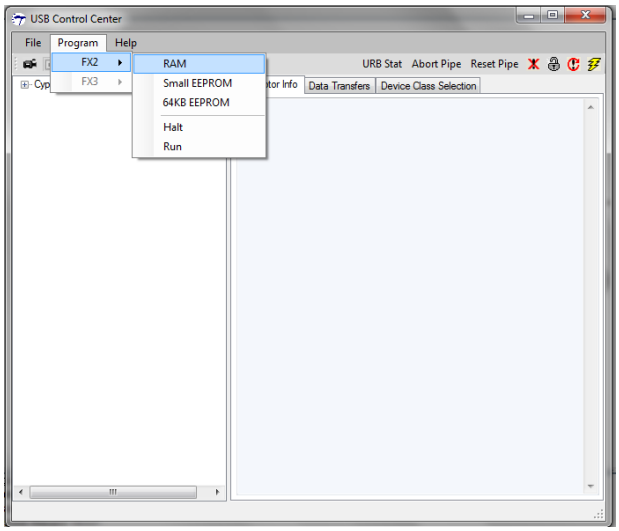
在通过 USB 将 ZTEX FPGA 模块连接至主机后，请检查器件的 VID 和 PID。如果电路板上的 EEPROM 中没有存储任何固件信息，则 VID 和 PID 分别为 0x04b4 和 0x8613。如果通过 ZTEX SDK 构建的固件被存储，则 VID 和 PID 分别为 0x2214 和 0x0100。请确保将这些 VID 和 PID 的值添加到 *CYUSB.inf* 文件内。将‘*CYUSB.sys*’驱动程序分配给该器件。如果器件需要显示 Control Center 工具，则需要实现该操作。

更多有关将 *CYUSB.sys* 分配给带有自定义 VID 和 PID 的器件的详细信息，请参考文档 *CyUSB.pdf* 中的内容。当您安装赛普拉斯的 SuiteUSB 开发工具时，可以在下面路径中查找 *CyUSB.pdf* ： C:\Cypress\Cypress Suite USB 3.4.7\Driver （根据安装路径有所变化）。

2. 下载 FX2LP 固件

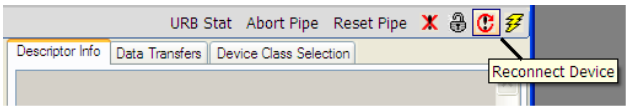
打开 Control Center 工具并下载 FX2LP 固件（依次选择 **Program>FX2>RAM**，然后导航到关联项目文件夹中的 *slave.hex* 文件）。

图 15. 将固件镜像下载到 FX2LP RAM



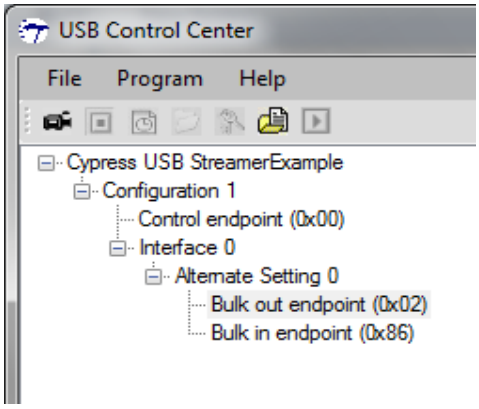
如果您的器件 VID 和 PID 分别为 0x2214 和 0x0100，则在下载 *slave.hex* 后，您需要按下下图所示的按键，以重新连接器件。

图 16. 通过 Control Center 重新连接按键



FX2LP 按照图 17 所示的内容进行枚举。端点 2 被配置为 Bulk OUT 端点，端点 6 被配置为 Bulk IN 端点。

图 17. 下载 *slave.hex* 后 Control Center 上的 FX2LP 器件视图



3. 下载 FPGA 位流

将 JTAG 线缆连接至 ZTEX 实验电路板 1.3 上的 JTAG 链接器。然后，FPGA 的 PROG_B 引脚（连接至 FX2LP 的 PA1）将变为高电平，以使能 FPGA 的 JTAG 配置。通过 Spartan-6 兼容位流（本应用笔记提供的 Stream IN、Stream OUT 或回送位流）并根据下面各个图表配置 FPGA。

图 18. ChipScope Pro

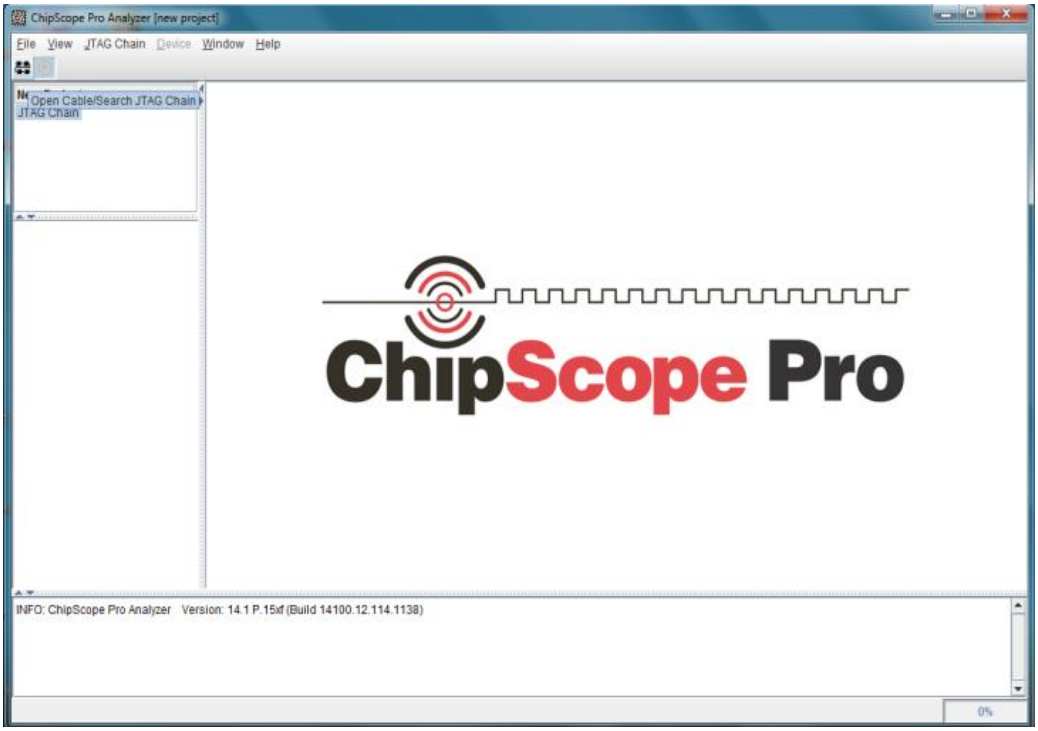


图 19. 使用 ChipScope Pro 配置 FPGA

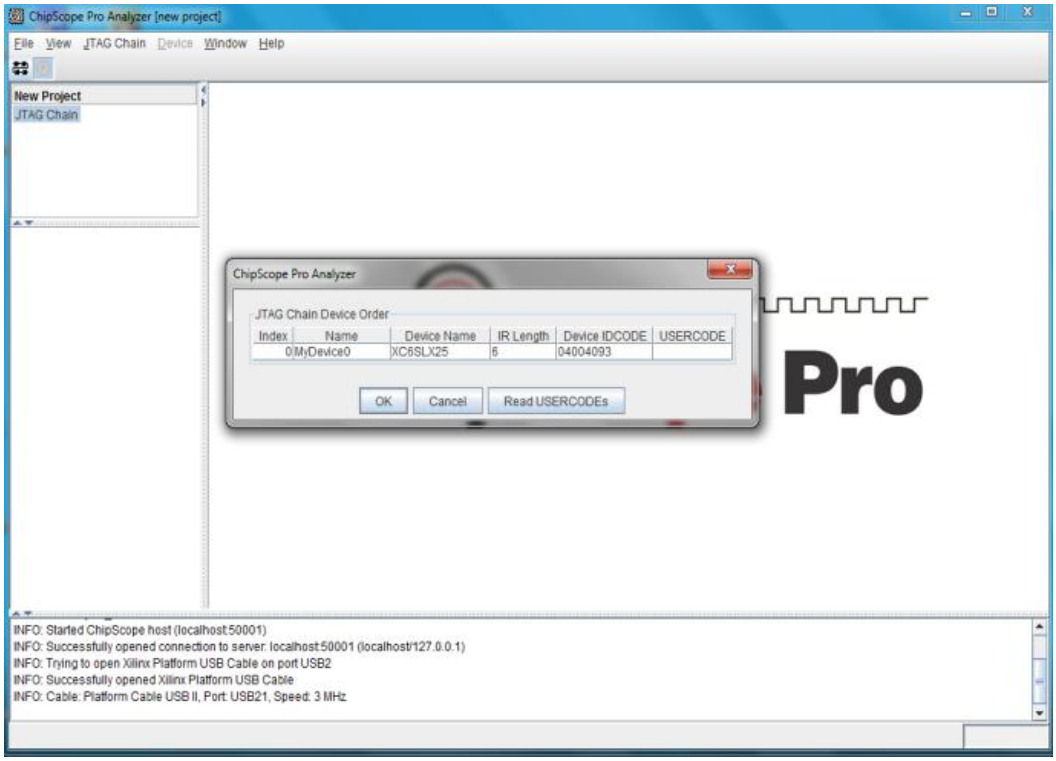


图 20. 使用 ChipScope Pro 配置 FPGA

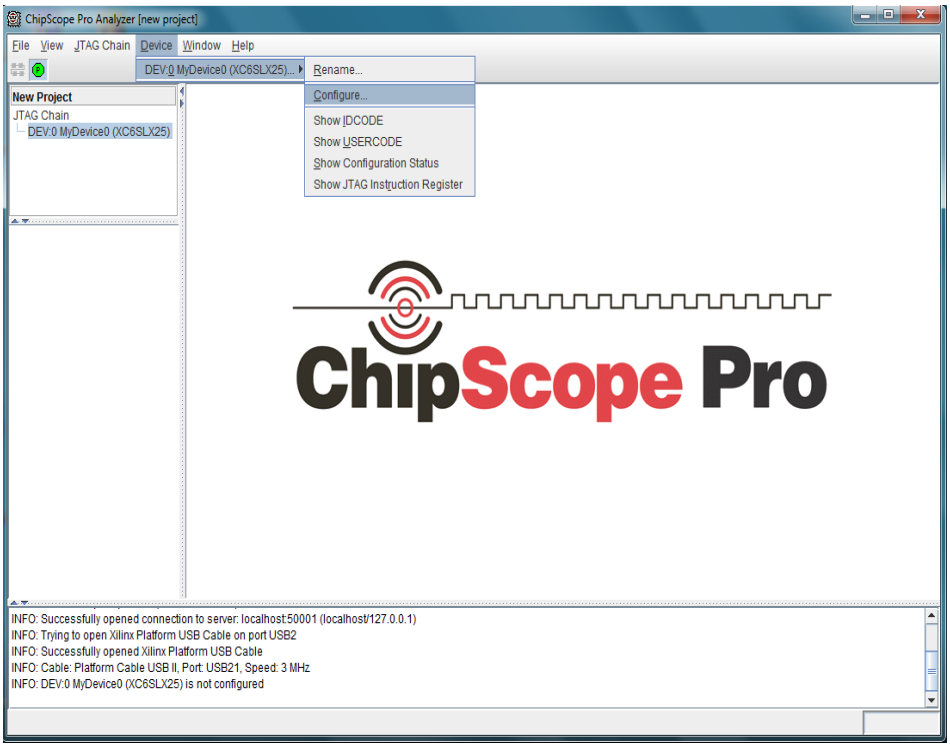
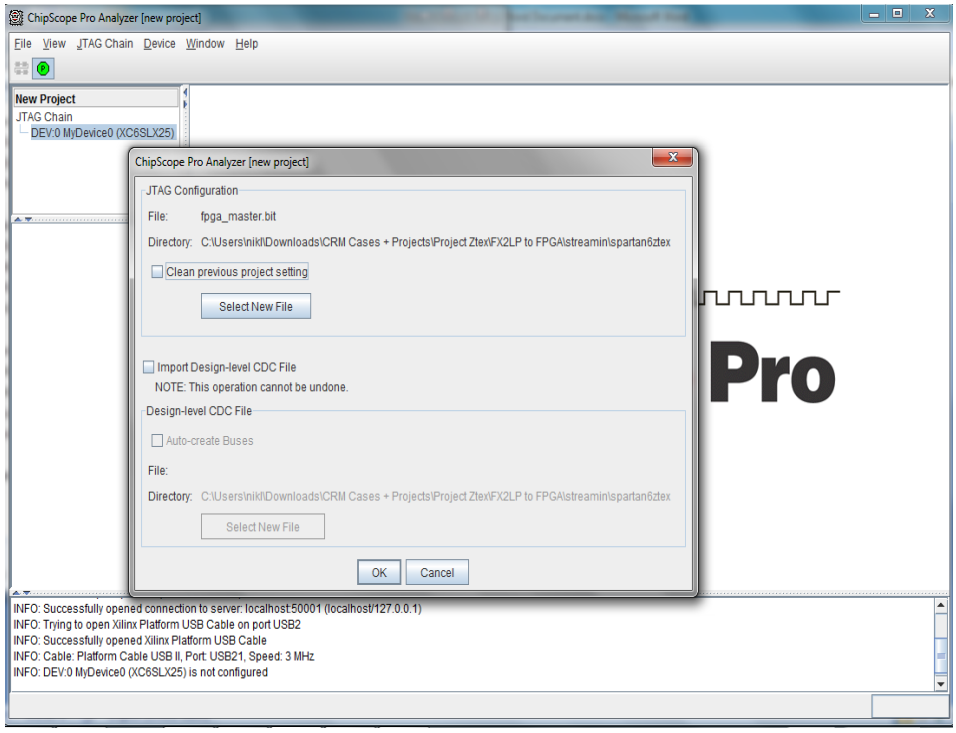


图 21. 使用 ChipScope Pro 配置 FPGA



4. 验证回送位流的结果

如果将 FPGA 配置为回送位流（第 3 步骤），它将在端点 EP2OUT 和 EP6IN 上实现一次回送操作。FPGA 将读取主机发送到 EP2 的数据，并将这些数据写入到 EP6 端点 FIFO 内。

为验证回送操作，在 Control Center 窗口中选择 ***Bulk Out Endpoint (0x02)***项，然后点击 **Transfer File-OUT** 按键，并浏览 *512_count.hex*（附件中提供）以执行数据传输。通过点击 **Transfer Data-IN** 按键读取 EP6IN 缓冲器，以验证写入到该缓冲器的数据。下面各图显示了这些步骤。

图 22. 将某个文件传输到 OUT 端点 2

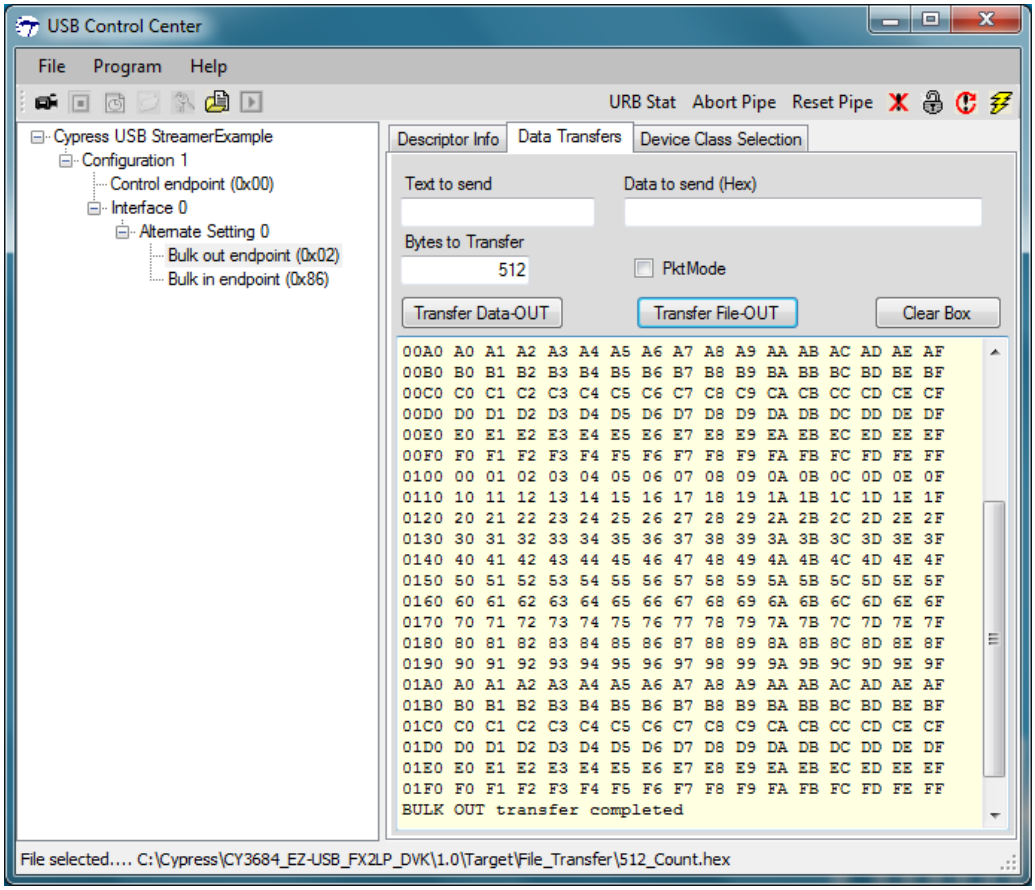
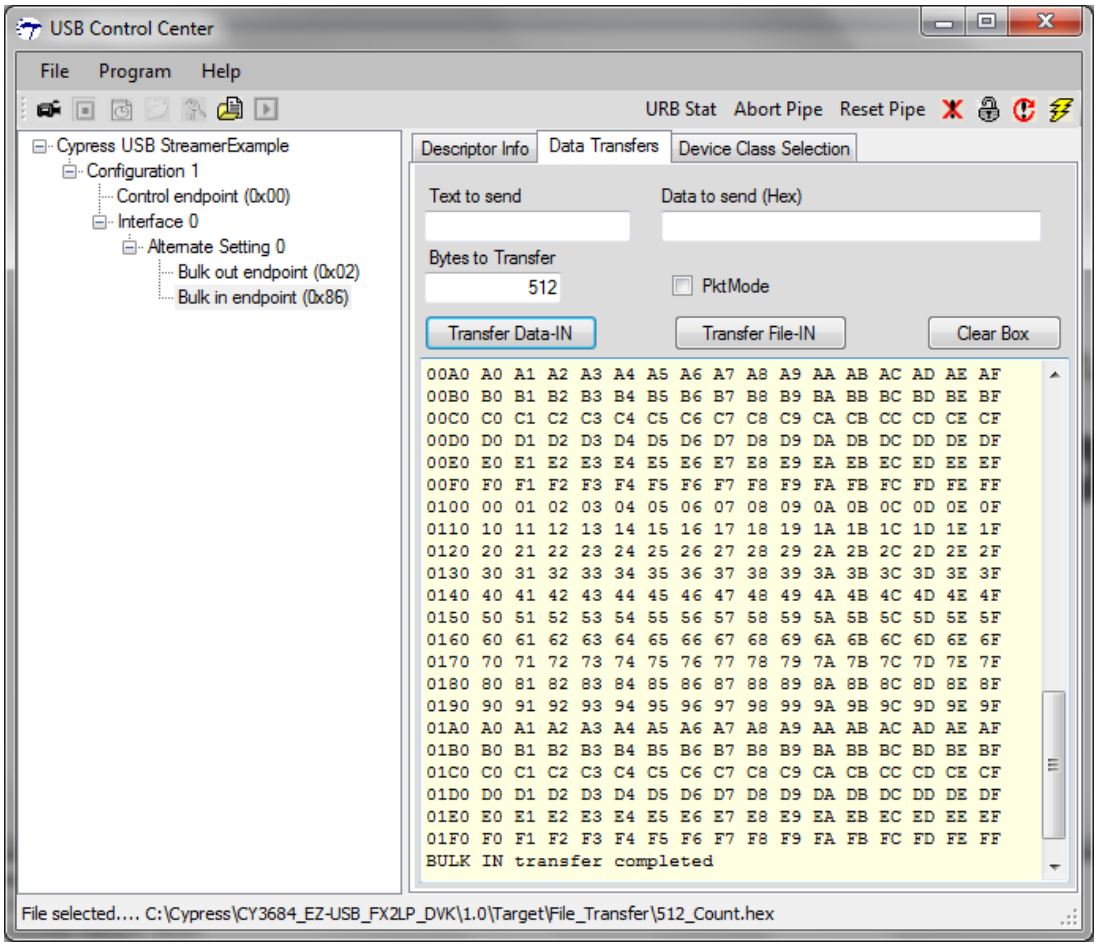


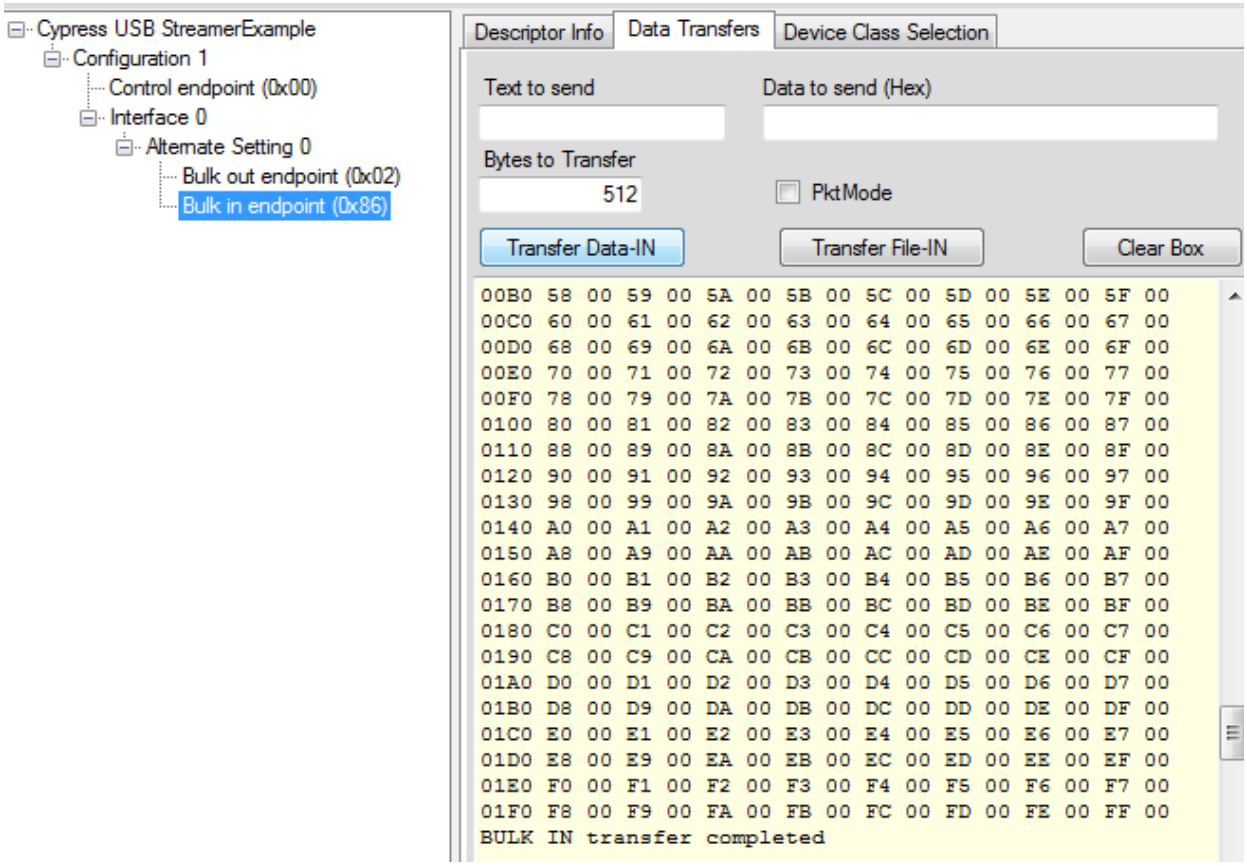
图 23. 从 IN 端点 6 读取同样的数据



5. 验证 Stream IN 位流的结果

如果通过 Stream IN 位流配置 FPGA（第 4 步骤），将使 FPGA 进入主设备模式。FPGA 会将递增数据发送到 FX2LP 的端点 6IN。为验证 Stream IN 操作，在 Control Center 窗口中选择 **Bulk in endpoint (0x86)**项，然后点击 **Transfer Data-IN** 按键。这时，您可以查看数据，如下图所示：

图 24. 从 IN 端点 6 读取数据



6. 验证 Stream OUT 位流的结果

如果通过 Stream OUT 位流配置 FPGA（第 4 步骤），将使 FPGA 进入主设备模式。FPGA 将读取来自 FX2LP 的 OUT 端点 2 的数据。通过使用 USB Control Center 工具，您可以对端点 2 执行无限的 OUT 传输数量。您还可以通过运行一个流式程序来验证端点 2 上的 OUT 传输。FPGA 根据来自 FX2LP 的标志从 OUT 端点 2 读取数据。FPGA 只会忽略所接收的数据，并等待更多来自 FX2LP 的 OUT 端点 2 的数据。

吞吐量测量

通过使用代码配置 FPGA，可以进行数据流传输。通过赛普拉斯 Streamer（流式）应用（套件 USB 3.4 中提供的），可以侧脸接口的吞吐量。如果器件与 *CyUSB.sys* 驱动程序（版本 3.4.7）绑定在一起，则在运行 Windows 7 的某个 64 位 Intel 7 系列/c216 芯片集系列中，可以测量到大小为 39 MB/s 左右的吞吐量。更多有关流式器件的吞吐量评估的详细信息，请参考应用笔记 AN4053 — 通过 EZ-USB FX2™和 EZ-USB FX2LP™上的同步/批量端点进行数据流传输。

图 25. 针对 Stream IN 传输测量的吞吐量

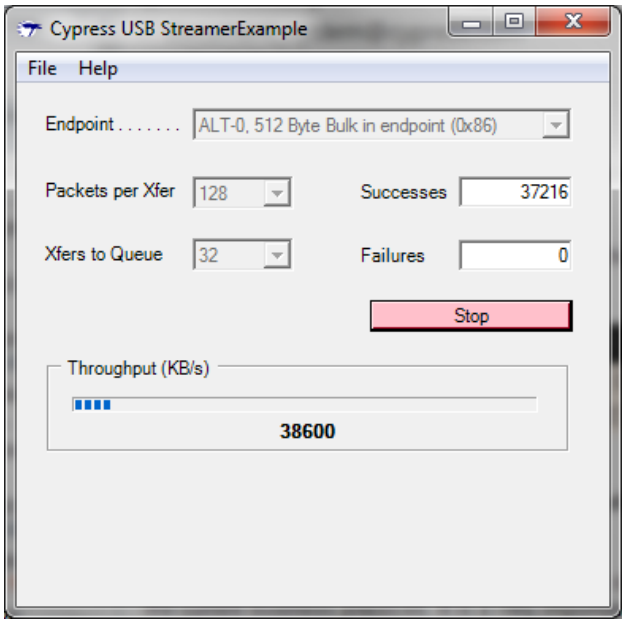
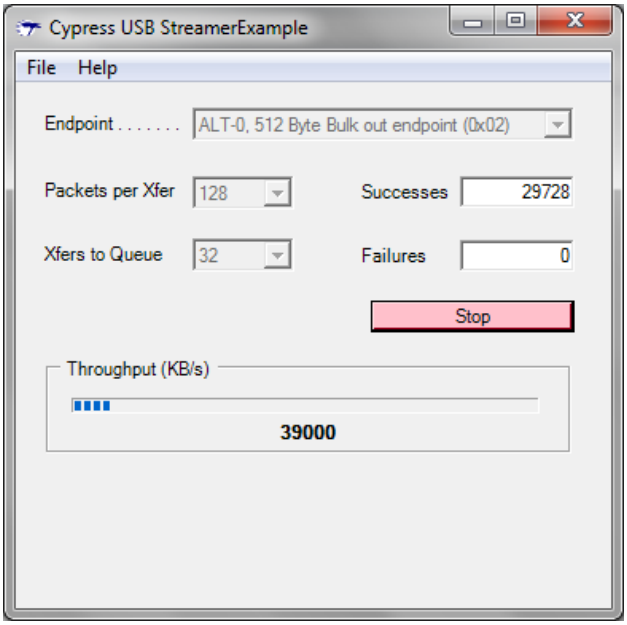


图 26. 针对 Stream OUT 传输测量的吞吐量



注意： 通过选择流式应用中的 **256 “Packets per Xfer”和 64 “Xfers to Queue”**，可以测量吞吐量。

关联项目文件

表3描述了本应用笔记附带的文件。

表 3. 应用笔记文件的说明

文件/文件夹名称		说明
FX2LP firmware		FX2LP 固件项目源文件
FPGA Source Code_Verilog	Loopback	用于执行数据回送的 FPGA Verilog 源代码。
	Stream IN	用于执行 Stream IN 数据传输的 FPGA Verilog 源代码。
	Stream OUT	用于执行 Stream OUT 数据传输的 FPGA Verilog 源代码。
FPGA Source Code_VHDL	Loopback	用于执行数据回送的 FPGA VHDL 源代码。
	Stream IN	用于执行 Stream IN 数据传输的 FPGA VHDL 源代码。
	Stream OUT	用于执行 Stream OUT 数据传输的 FPGA VHDL 源代码。
512_count.hex		用于执行文件传输的 512 个字节数据。
Firmware_SDCC		需要被编译的 FX2LP 固件项目源文件（通过使用 SDCC 编译器）
Firmware_SDCC/Release		使用 SDCC 编译器编译 FX2LP 固件项目后所生成的文件。
Readme_SDCC.pdf		用于解释如何使用 SDCC 编译器来编译某个项目的文档。
Sdccman.pdf		用于解释 SDCC 编译器的 SDCC 参考手册。

如何导入该设计，以便与 Altera® FPGA 一起工作

如要导入该设计，以便同 Altera FPGA 一起使用，您需要执行以下各步骤：

1. 使用 Altera 工具生成的基元。
   - 用户生成时钟的 PLL
   - 用于为 FX2LP 提供内部时钟的 DDR
2. FX2LP 从设备 FIFO 接口信号到 FPGA 的引脚映射情况

总结

本应用笔记描述了如何通过 FX2LP（配置为从设备 FIFO 模式）来设置 FPGA-FX2LP 接口。关联项目包含了用于在从设备 FIFO 模式下初始化 FX2LP 的固件，以及用于将 FPGA 配置为 FX2LP 的主设备的 Verilog 和 HDL 代码。

关于作者

姓名： Rama Sai Krishna V

职务： 应用工程师

文档修订记录

文档标题： AN61345 — 使用 EZ-USB® FX2LP™从设备 FIFO 接口进行设计

文档编号： 001-92463

修订版	ECN	原始变更	提交日期	变更说明
**	4376869	LISZ	08/29/2014	本文档版本号为 Rev**，译自英文版 001-61345 Rev*J。
*A	5715619	AESATP12	04/27/2017	Updated logo and copyright.
*B	5872510	ZHUB	09/04/2017	增加程序例子的链接. 本文档版本号为 Rev*B, 译自英文版 001-61345 Rev*L。

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

赛普拉斯开发者社区

论坛 | WICED IoT 论坛| 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

EZ-USB®和 FX3™是赛普拉斯半导体公司的注册商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

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

赛普拉斯半导体公司，2010-2017 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）：（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产