基于 cy7c68013 的 USB 通信（FPGA 端开发资料）

FPGA 基础和应用

USB 基本知识

概念：USB 是一种通用串行总线。

USB 系统：一定是由一个 PC 和外围设备组成。外设称为 USB 设备，PC 为主机。主机里有 USB 主控制器，负责完成主机和 USB 设备之间的物理数据传输。USB 主控制器分为两种，具体哪两种这与 FPGA 开发无关，在此不细说。USB 设备分为集线器和功能设备两种，集线器可以让一个 USB 端口连接多个设备。功能设备指的是功能性的设备，例如鼠标键盘。

USB 信号线：一共四根线，两根差分 NRZI 编码线。两根电源线，电压 5V,最大电流 100mA，500mA。

USB 事务处理：主机和 USB 设备之间的数据传输最基本的单位。它是有特定格式的信息包。主机和 USB 设备之间的通讯需要多个事务处理包才能完成。

USB 的传输类型：控制传输、批量传输、中断传输、同步传输。具体如下表。

表 1.1 USB 四种传输类型的比较

传输类型	端点类型	传输方向	所传输数据的特点
批量传输	块端点	IN 或 OUT	大量、无传输时间和传输速率要求
中断传输	中断端点	IN 或 OUT	少量或中量、有周期性
同步传输	同步端点	IN 或 OUT	大量、速率恒定、有周期性
控制传输	控制端点	IN 和 OUT	少量、无传输时间要求、传输有保证

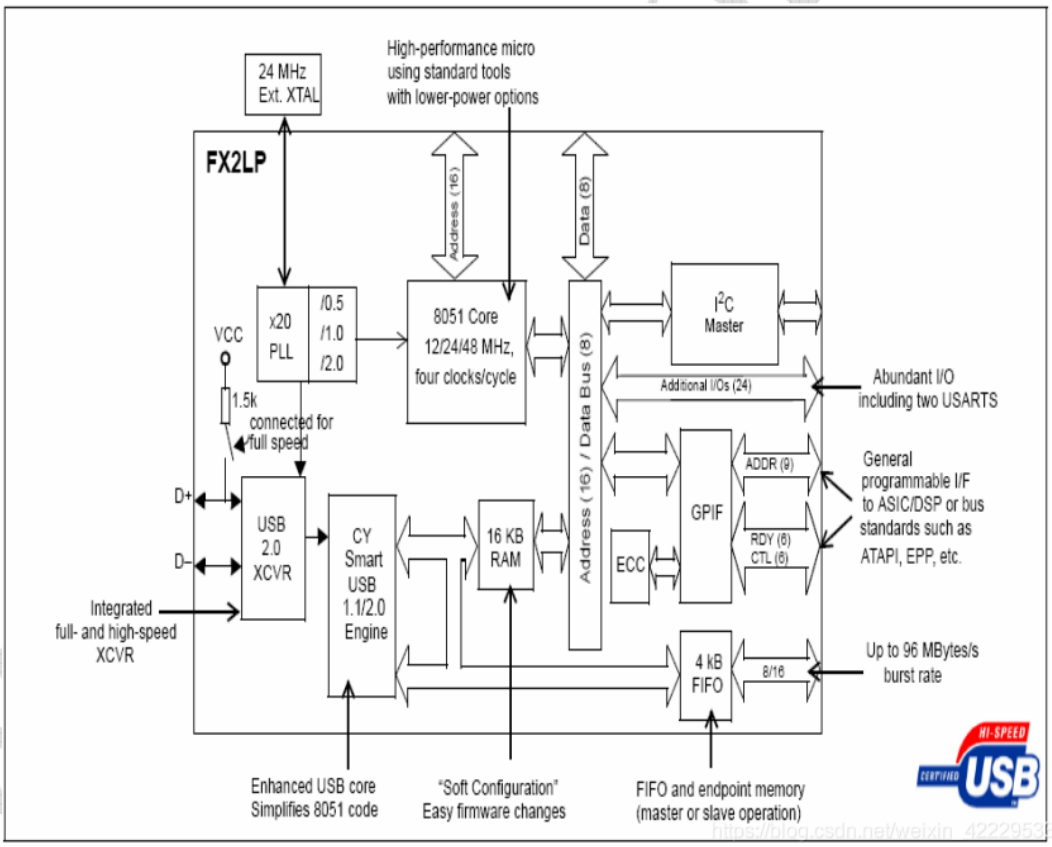
同时，USB 接口和 FPGA 的通讯大都采用的是 68013 芯片的 FIFO 模式。根据 FIFO 的传输特性，满足大量、速度恒定、有周期性的要求，所以应该是同步传输模式。这种通讯类型等特性的选择可以通过固件配置。

USB 的描述符：这个东西的定义晦涩难懂，对于 FPGAer 来说，把它看成了用来定义 USB 各个性能参数的寄存器。

USB 的设备请求：大概定义了 USB 各类请求格式。对于 FPGAer 而言，依旧不知道它干啥的，应该也不用知道。

CY7C68013 芯片

简介：他是第一款支持 USB2.0 协议的微处理器。支持 12Mb/s 全速传输(即 USB0)和 480Mb/s(即 USB2.0)高速传输。上文提到的四个传输类型该芯片都支持，但是和 FPGA 通讯所采用的 FIFO 模式，是其中的同步传输模式。具体框架如下图：



大概的接口图如下：

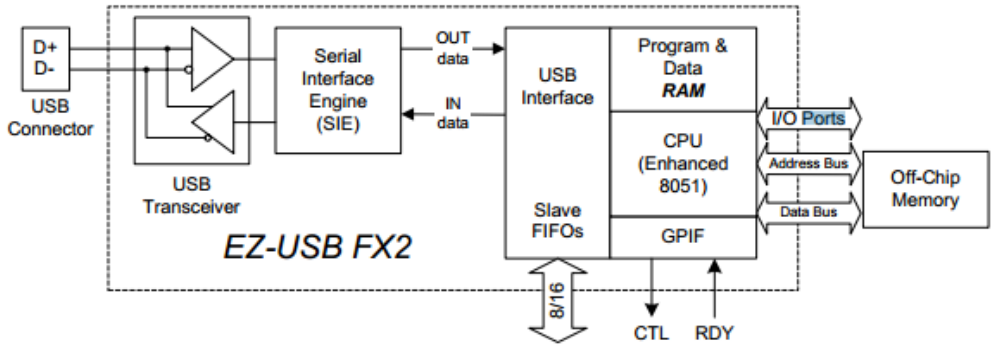


Figure 1-8. FX2 128-pin Package Simplified Block Diagram

显著特点：

USB 单芯片解决方案，集成了 USB2.0，串行接口引擎，和增强型 51 内核。

可以软件配置 RAM（即配置固件），方式多种：USB 口下载、E2PROM（有专门的烧写软件）、或者外围的存储（只限于部分封装）。

可编程 GPIF。

集成的 81 内核。

智能串行接口引擎。

中断矢量

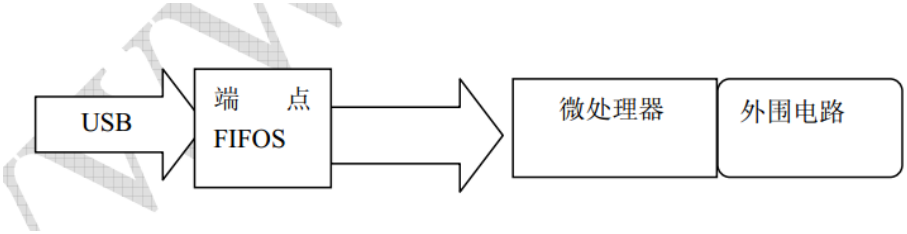
I2C 接口

4 个集成的 FIFO 轻松和外部 FPGA/DSP 等连接

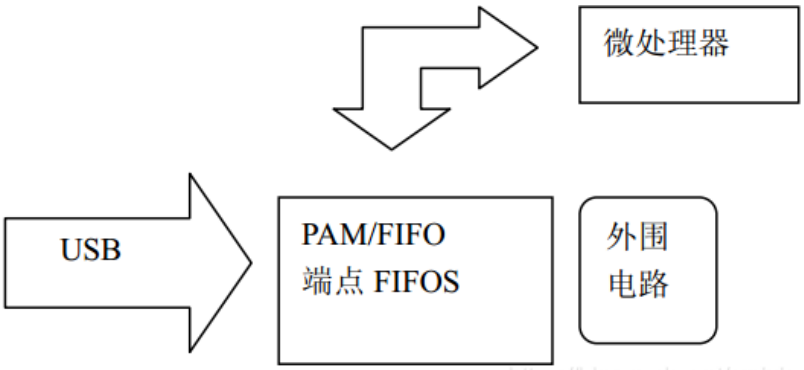
，

FIFO 模式

该模式是与 FPGA/DSP 等芯片连接的一种常见模式。之前版本的 USB 芯片，微处理器（51 内核的）都要参与数据的处理。在速率不高的时候，12Mb/s 全速传输模式下没有问题，但是到了 480Mb/s 的时候，微处理器就成为了瓶颈。



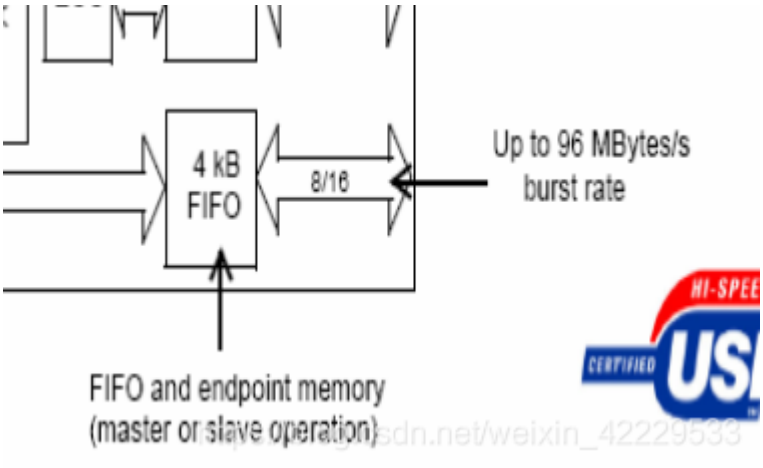
而此版本的 68013 直接解除了微处理器对数据处理的干预。



68013 的通讯类型等特性可以通过固件配置。根据 FIFO 的传输特性 ,满足大量、速度恒定、有周期性的要求，所以应该是同步传输模式。

最高是 96MByte/s。以 8bit 为例 ,则传输速率是 96Mhz ,一共四个 FIFO ,可以同时工作，所以 fifo 最高速率是 24Mhz。（ FPGA 和 FIFO 的通讯同时只能和一个 fifo ）

2.2 增强型 8051 核



增强型 8051 核

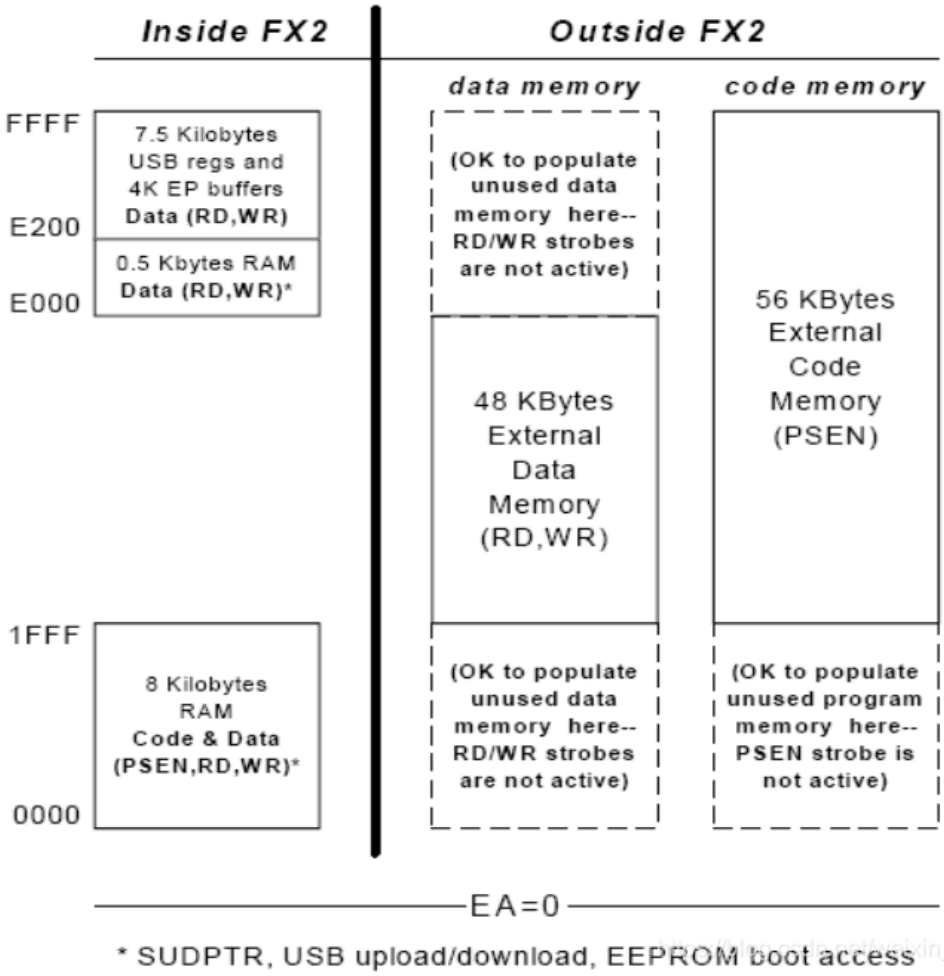
2.2 增强型 8051 核

EZ-USB FX2 中内嵌的增强型 8051 微处理器带有 256B 的数据存储器、扩展的中断系统、3 个定时/计数器和 2 个串口 UART。FX2 需外接 24MHz 的晶振，并匹配 20pF 的电容接地，经过内部振荡电路和索相环(PLL)倍频电路，产生 48MHz 的增强型 8051 默认工作频率。根据需要，用户也可设定 8051 工作于 24MHz 或 12MHz 的频率下。FX2 还需产生 480MHz 的时钟脉冲以供 USB2.0 串行收发使用。

画重点：外接 24Mhz,内部分频，产生 480MHZ 供 USB2.0 使用。（疑问：这里的 480MH 的时钟就是为上文中 480Mb/s 高速传输模式提供的源吗）

8051 的内存：由下图而知，从下至上主要分为了 8K 的主 RAM(地址 0000·1FFF)，用于缓存代码和数据，同时这部分的内存可以通过 EA 角进行外扩内存。（地址 E000·E1FF)作为临时 RAM,不能跑程序。（地址 E400·E47F)用于存储 GPIF 的波形。（地址 E600·E6FF)用于保存控制状态寄存器。（地址 E740·E7FF)和（F000-FFFF）都是用于数据缓冲的内存。分别称之为端点 0/1/2/4/6/8。 其中端点 0 和端点 1 仅能由固件程序访问。2/4/6/8 端点是 FIFO 的空间。

对于 FPGA 开发者而言，唯一需要了解的是，FIFO 模式下读写的数据（端点 2/4/6/8）缓存的地方其实就是 8051 的内存区 F000-FFFF。



* SUDPTR, USB upload/download, EEPROM boot access

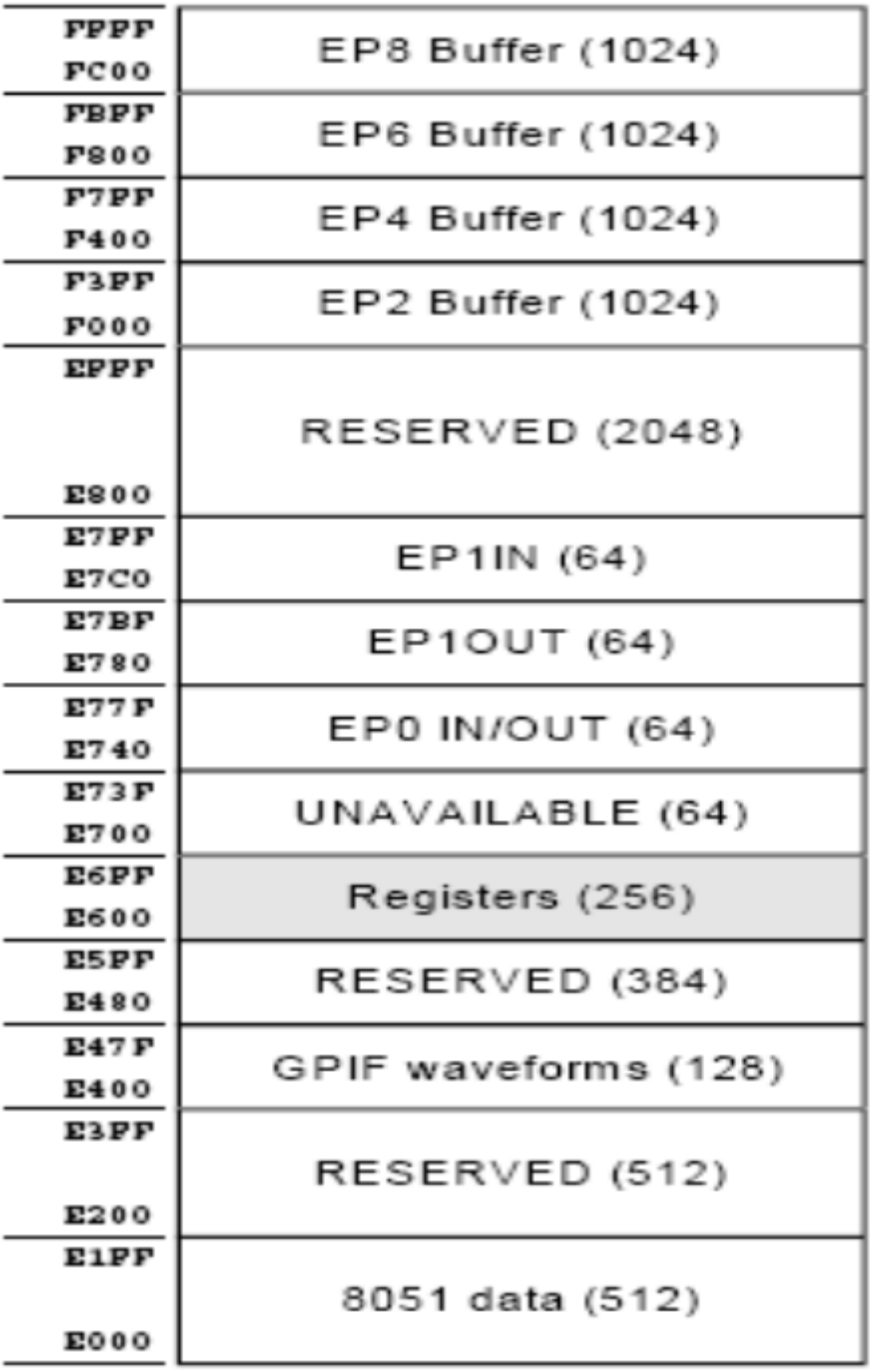


图 2.9 片内数据存储区 0xE000~0xFFFF 的结构

IO 系统：IO 功能分为了三类，第一类就是普通的 IO 口，第二类就是 FIFO 模式下的数据读写口，第三类是 GPIF 接口。

Slave FIFO 模式调试（USB 设备为 FPGA）

1.基本要素：虽然基于 FX2（68013）芯片的设备可以直接使用 FX2 的内核 8051 传输数据，但是大部分环境下，还是将 FX2 用做 USB 和外部逻辑芯片（DSP 或者 ASIC 等）的一种连接通道。

USB 数据通过可以通过 FX2 的内置 FIFO 让数据在主设备（PC）和逻辑设备(FPGA)间传输。

GPIF 和 slave fifo 模式其实都是属于 FX2 芯片的 fifo 模式。、

对于那些逻辑芯片里没有标准 fifo 接口的，FX2 里面内置的 GPIF 可以作为主控，在这种模式下（GPIF 模式下）我们称之为主 fifo 模式，FX2 本身就是 FIFO 的主控设备。当 FIFO 被外部的设备控制时（外部设备主控），例如，和 FPGA 相连，就是处在 Slave FIFO 模式。

Slave FIFO 模式下，包括了四个 FIFO。他们能够工作在 16 位和 8 位模式。大概的系统框图如下：

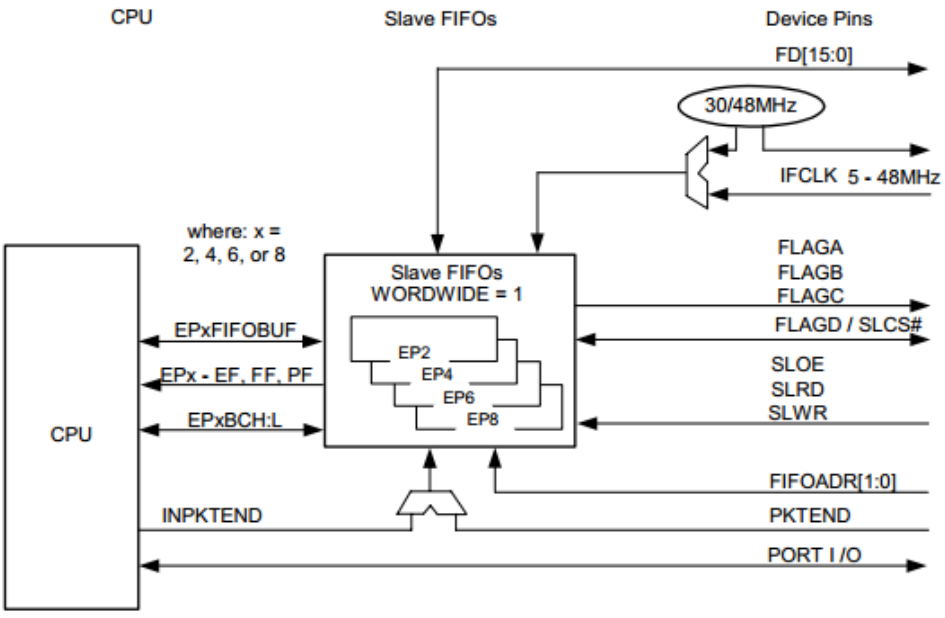


Figure 9-1. Slave FIFOs' Role in the FX2 System

其中硬件配置涉及的部分寄存器如下：

Table 9-1. Registers Associated with Slave FIFO Hardware

IFCONFIG	EPxFIFOPFH/L
PINFLAGAB	PORTACFG
PINFLAGCD	INPKTEND
FIFORESET	EPxFLAGIE
FIFOPINPOLAR	EPxFLAGIRQ
EPxCFG	EPxFIFOBCH:L
EPxFIFOCFG	EPxFLAGS
EPxAUTOINLENH:L	EPxBUF

2.引脚：

FX2 的复位只能在普通的 IO 模式下进行,在 fifo 模式下是不可以进行复位的，所以你看在 slave 模式下是没有复位信号的。

如果想把芯片配置到 SLAVE fifo 模式下，需要对 IFCONFIG 这个寄存器进行设置，fifo 和外部主控的信号连线如下图：

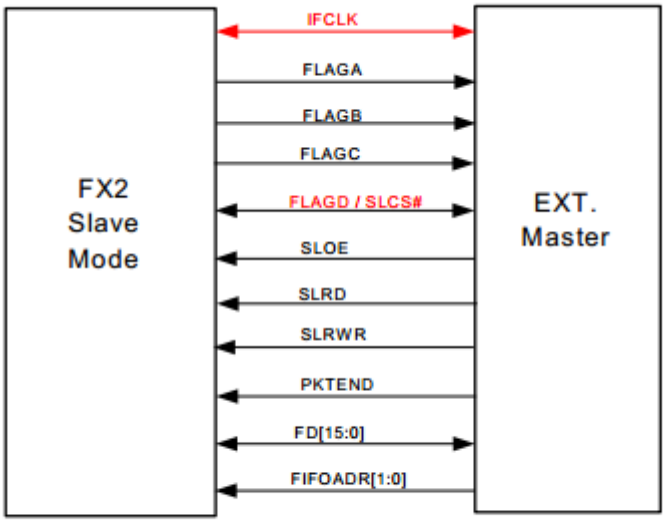


Figure 9-2. FX2 Slave Mode Full-Featured Interface Pins

SLOE:FD 数据信号是双向的，可以通过 SLOE 信号线进行选择。低有效。

FIFOADR:选择哪一个 FIFO 挂到 FD 总线上。低有效。

SLRD、SLWR:读写信号。分为了同步和异步两种模式。低有效。

FD:总线。可以配置为 8bit 和 16bit 模式。8 模式下，用的是 PORT B。16 模式下，用的是 PORT B 和 PORT D。(PORT 的概念可以去 FX2 手册中找)。上电默认是 16 位模式。其中，如果改成 8 位模式的时候，PORT D 可以作为普通 IO 使用。

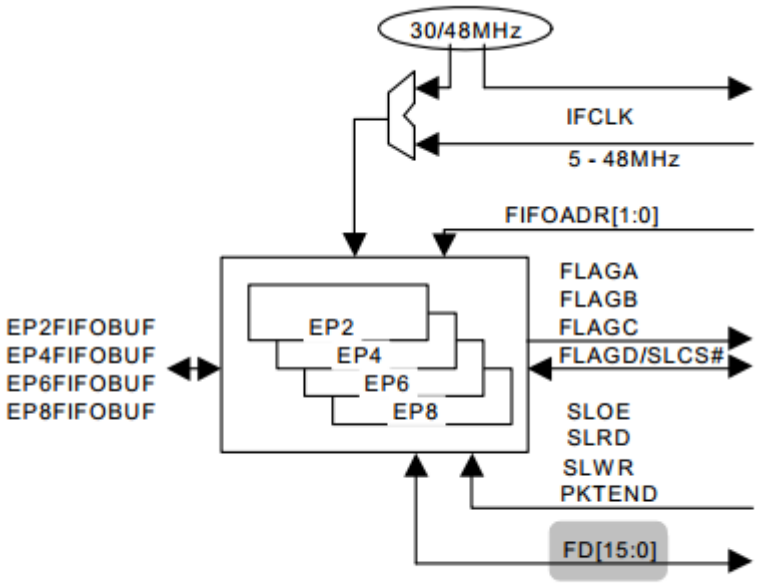


Figure 9-5. 16-bit Mode Slave FIFOs, WORDWIDE=1

IFCLK:可以使用内部时候和外部时钟。内部时钟的时候，时钟频率 30/48 可以选择。外部时钟的时候，5 到 48MHZ 时钟可以调整。上电默认是内部时钟 48MHZ。

FLAG(FLAGA, FLAGB, FLAGC, FLAGD):这些信号主要用于监测 FIFO 状态，具体功能可以配置。FLAGA, FLAGB, and FLAGC 这些都可以工作在 indexed 和 fixed 两个模式。FLAGD 只能工作在 FIXED 模式。

Indexed 模式下，flag 只能显示所选择的 fifo 的状态，FLAGA/B/C 分别代表“programmable-level”，full,empty。Fixed 模式下，flag 显示特定 fifo 的状态，而无论是否选择了该 FIFO。full,empty 默认是低有效，也可以用寄存器进行配置。

上电默认是 Indexed 模式。

FIFOADR:选择哪一个 FIFO。

PKTEND:无论设置的数据长度是多少，有效该信号则代表数据传输的结束。一般用于长度比较短的数据。例如，如果通过寄存器设置的数据长度是 512byte.当一次传输的数据长度短于 512 时，由于数据低于设置的长度，一般主控有两个方法让数据发送。一个就是发送虚拟的数据满足最终的长度。或者有效 PKTEND 脚，这样就能直接写短长度的数据。

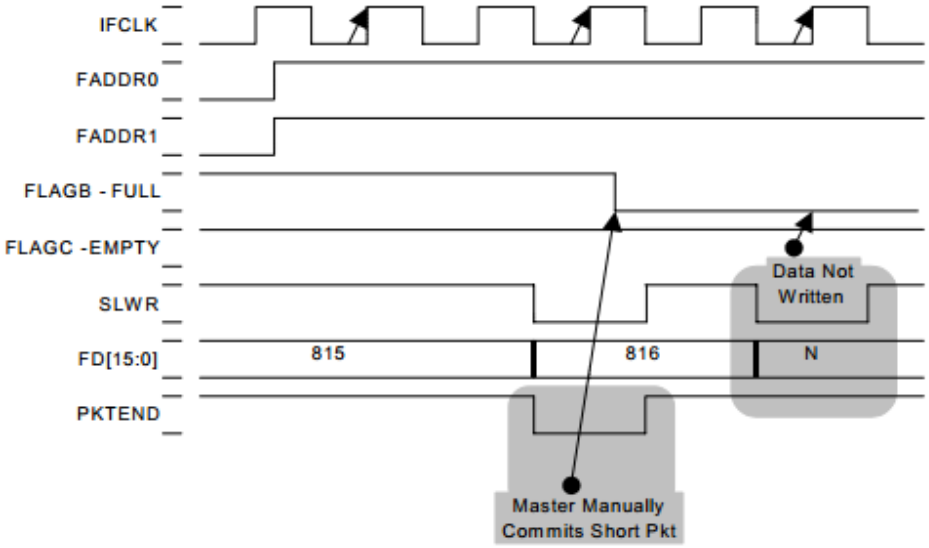
SLCS:FIFO 选通信号。该信号有效，则所有控制才有效。

3.时序

引脚的控制时序：控制管脚主要是 SLOE(输出使能)，SLED(读使能)，SLWR(写使能)，PKTEND(包尾)，FIFOADR(FIFO 选择)。这里的读写都是以外部的主设备角度讲的。

读时序 :SLOE 使能 FD 总线脚。 同步模式 ,SLRD 在时钟上升沿有效 ,SLRD 有效同时 FIFO 计数加一个。异步模式则在 SLRD 变化时有效。他们默认都是低有效。

写时序：同步模式，SLWR 上升沿有效。默认都是低有效。



ure 9-14. Timing Example: Synchronous FIFO Writes, Waveform 3, PKTEND Pin Illustrated

写案例：

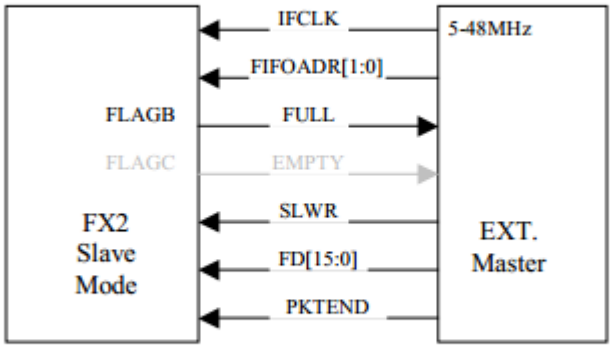
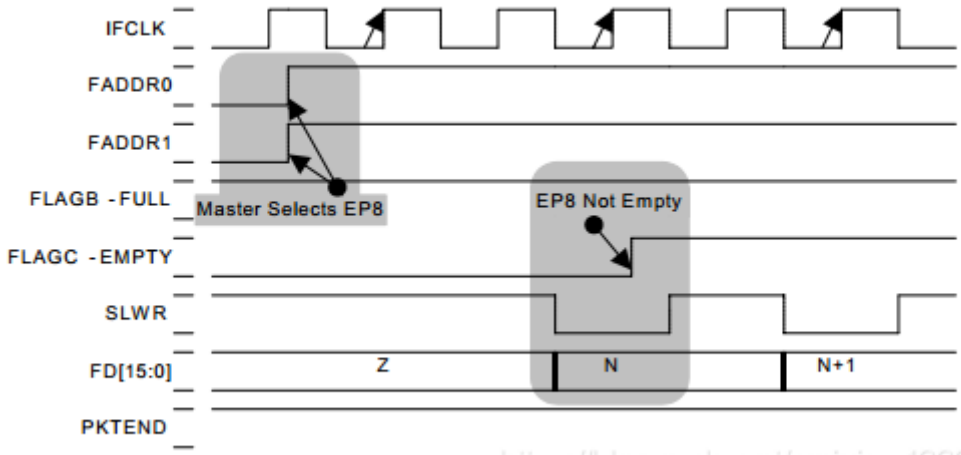


Figure 9-10. Interface Pins Example: Synchronous FIFO Writes

典型写过程（相对于主机）：



Typically, the sequence of events for the external master is:

IDLE: When write event occurs, transition to State 1.

STATE 1: Point to IN FIFO, assert FIFOADR[1:0], transition to State 2.

STATE 2: If FIFO-Full flag is false (FIFO not full), transition to State 3 else remain in State 2.

STATE 3: Drive data on the bus, assert SLWR for one IFCLK, transition to State 4.

STATE 4: If more data to write, transition to State 2 else transition to IDLE.

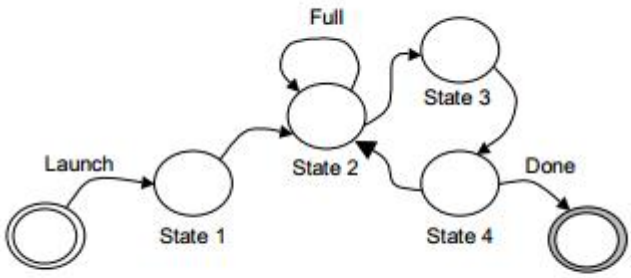


Figure 9-11. State Machine Example: Synchronous FIFO Writes

典型读过程（相对于主机）：

IDLE: When read event occurs, transition to State 1.

STATE 1: Point to OUT FIFO, assert FIFOADR[1:0], transition to State 2.

STATE 2: Assert SLOE. If FIFO-Empty flag is false (FIFO not empty), transition to State 3 else remain in State 2.

STATE 3: Sample data on the bus, increment pointer by asserting SLRD for one IFCLK, de-assert SLOE, transition to State 4.

STATE 4: If more data to read, transition to State 2 else transition to IDLE.

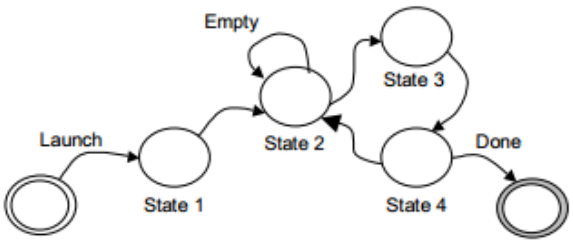


Figure 9-16. State Machine Example: Synchronous FIFO Reads

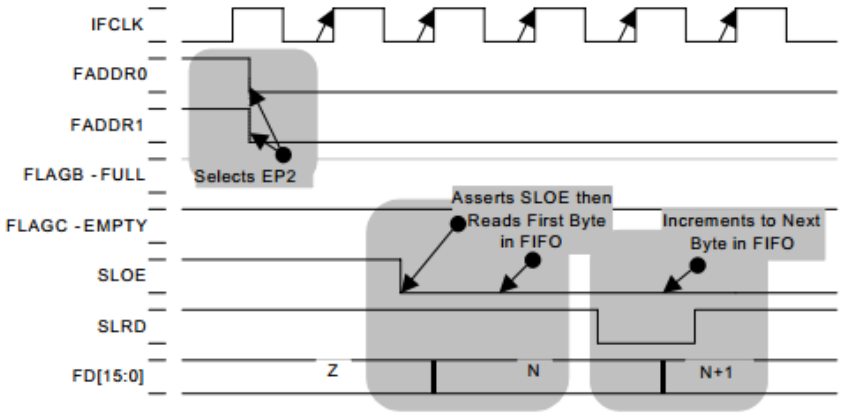


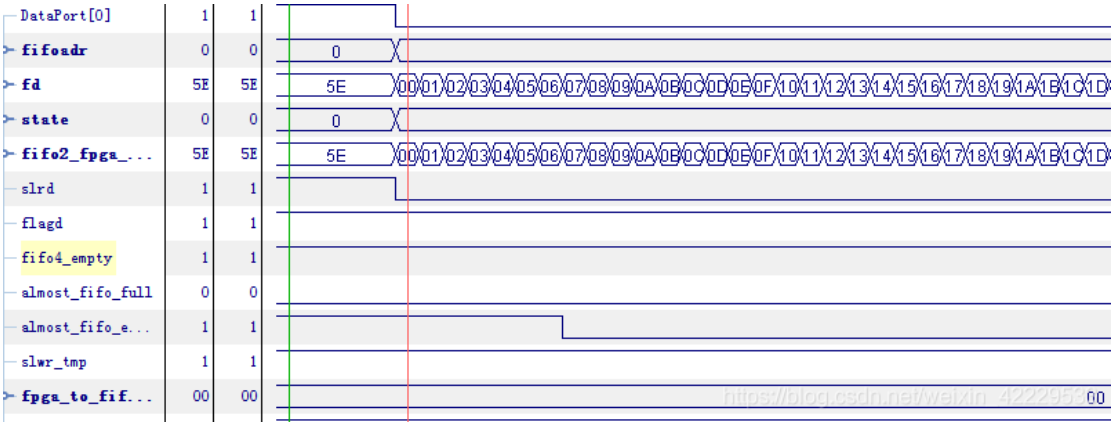
Figure 9-17. Timing Example: Synchronous FIFO Reads, Waveform 1

注意：读过程比写过程多了对 SLOE 信号的使能。

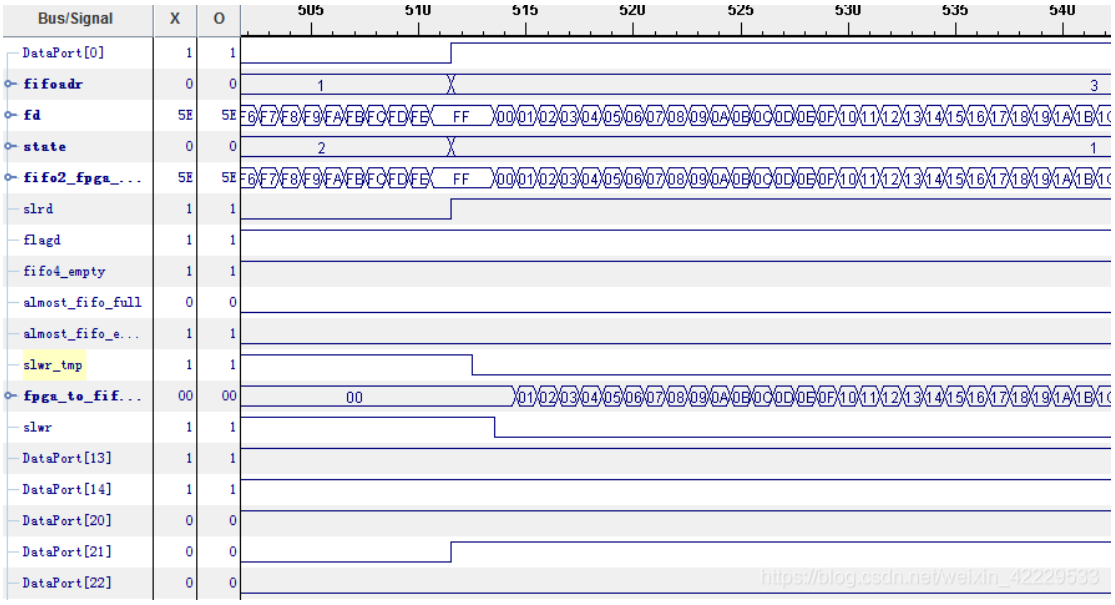
异步时序暂时不做讨论。可以参照官方文档。

调试记录

（读）：



（写）：



Rd 和 oe 的关系：按照官方推荐 sloe 是需要提前 slrd 有效的