UNIVERSIDAD NACIONAL DE SAN JUAN
FACULTAD DE INGENIERIA
Departamento de Electrónica y Automática

Universidad Nacional de San Juan

Trabajo Final

COMUNICACIÓN USB 2.0 PARA SISTEMAS CIENTÍFICOS IMPLEMENTADOS EN FPGA

Informe

Edwin Barragán
Autór

Ing. Cristian Sisterna **Mg. Martín Pérez** **Dr. Marcelo Segura**
Asesores

2018

Agradecimientos

Acá le agradezco a todos los miembros de la prestigiosa y gloriosa Comisión de Trabajo Final por sus incontables aportes a la causa. Si pongo punto y meto enter no se vé en el documento.
Si escribo barra barra hago un salto de linea pero no cambio de párrafo.

Si doy doble enter, coloca sangría, pero no hace el salto de línea para el párrafo.

Este último sí que es un párrafo decente!

Índice general

Capítulo 1

Introducción

1.1. Motivación

Un carpintero desea medir la distancia de una barra de madera que luego será, tal vez, la altura de las patas de una futura mesa. Para ello, utiliza una cinta métrica, compuesta de una cinta metálica que posee una escala graduada. Sabe entonces que la barra mide la distancia que coincide con la distancia de la cinta graduada.

Un panadero desea medir cuanto pesa la harina que debe para poder amasar. Entonces, la coloca en una balanza y observa cuanto marca su indicador. Así conoce que la masa de la harina es equivalente a la fracción de medida que indica la balanza.

Un atleta desea conocer cuanto demora en correr un trayecto que posee 1.00 km. Por esto, registra el valor que indica su reloj al principio del recorrido y cuando alcanza el final observa nuevamente el artefacto. Luego de esto, calcula la diferencia entre el valor final y el inicial, conociendo cuanto tiempo le tomó realizar su travesía.

En los tres casos anteriores, tanto el carpintero, como el panadero y el atleta desconocen algo y necesitan cambiar su estado con respecto a esa incertidumbre. Por ello recurren a diferentes objetos, a fin de obtener conocimiento a partir de ellos. Sin embargo, estos objetos, por si mismos, no otorgan información, sino más bien otorgan un dato, que comparado y contrastado con otros datos, se traducen en conocimiento.

La información es el resultado de ordenar y procesar un conjunto de datos, de forma tal que permitan cambiar el estado de conocimiento sobre un asunto determinado. En el caso del carpintero, compara el tamaño de las patas de la mesa con una cinta metlálica, que a su vez, posee registrada su distancia en función de algún patrón de metrología, establecido por convención. Esto quiere decir que el dato 1, la longitud del patron, junto al dato 2, escala graduada de la cinta, más el dato 3, la longitud de la cinta métrica, permiten al carpintero cambiar su estado de desconocido a conocido, con respecto a la longitud del trozo de madera, a través de la información proporcionada por el conjunto de datos.

Se puede realizar el mismo análisis con respecto a la balanza del panadero, considerando un peso patrón, un desplazamiento y una escala graduada o una señal eléctrica emitida por una celda de carga deformada un porcentaje de su capacidad, registrada previamente por su fabricante conforme a pesos patrones, y un circuito adaptador que transforma esa señal electrica en un valor numérico mostrado en un indicador.

El atleta compara las posiciones y los desplazamientos de las agujas de su reloj, previamente calibrado para que dé una vuelta por cada minuto en una aguja, otra aguja que dé una vuelta por hora y la tercera una vez cada 12 horas. Además, es probable que él haya ajustado la hora que indica el reloj para que otorgue un horario idéntico al de referencia, establecido por convención.

En todos los casos, se posee una gran cantidad de datos que, ordenados, procesados y comparados otorgan al usuario un valor útil, ya sea una longitud, una masa, un tiempo o cualquiera sea la variable física que se desee conocer.

La ciencia es un conjunto de técnicas y procedimientos que, a través del método científico, busca adquirir, descubrir y/o desarrollar nuevo conocimiento. Se desprende entonces, que la ciencia produce, de forma fundamental, información que luego es transformada en conocimiento. Cuando hablamos de ciencia, hablamos de una gran gama de objetos de estudio, sujeto a través del cuál se clasifican, en la mayoría de los casos, las ciencias: las Ciencias Sociales estudian las relaciones humanas, las Ciencias Naturales estudian objetos que se encuentran en la naturaleza, las Ciencias de la Tierra se enfocan en una rama más particular de la naturaleza, como lo son los minerales, la superficie terrestre, etc; y siguiendo así se puede encontrar un sinnuméro se ciencias. Sin embargo, toda ciencia necesita, para su correcta producción científica, adquirir una gran cantidad de datos que luego será nordenados, procesados y transformados en información y conocimiento.

La incorporación de una herramienta especialmente diseñada para el procesamiento de datos, como lo es la computadora, permite manejar un numero cada vez creciente de información. Es por eso que se encuentra en desarrollo un gran número de sensores y dispositivos que permitan obtener cada vez más datos.

En este sentido, uno de los desarrollo que se encuentran en boga es el sensores que adquieran imágenes. Como ejemplos podemos encontrar, entre muchos otros, el desarrollo de sensores de radiación[1], ultrasonografía[2], telescopía de objetos cercanos[3], imagenes de distancia[4].

La captura de imágenes, fundamentalmente en el desarrollo de sensores nuevos, requiere de sistemas digitales de alta velocidad que tengan la capacidad de acarrear los datos desde el lugar físico en donde se obtienen los datos, es decir, en el transductor mismo, hasta el circuito o el sistema destinado al proceso de los mismos. De esta forma, toma particular interés la utilización de FPGA's, circuitos integrados diseñados para que un diseñador pueda sintetizar un circuito digital de alta velocidad reprogramable en el cual se puede implementar, con ciertas restricciones, circuitos desarrollados para una tarea muy especifica que resuelva la tarea que el diseñador necesite.

A diferencia de un microprocesador o un microcontrolador, también muy usados en la industria electrónica, en el cual una unidad logica algoritmica ejecuta un programa cargado secuencial, es decir, linea por línea, un FPGA puede ser programado de forma tal que cada proceso se ejecute en forma independiente y paralela, dotando al sistema de una mayor velocidad en el procesamiento.

De esta forma, un diseñador puede manipular un volumen mucho mayor de datos, que a los efectos de la adquisición y medición de imágenes, resulta más adecuado.

Pero como ya se mencionó con anterioridad, la obtención de datos por si misma no le otorga al científico la información, y por ende, el conocimiento nuevo que desea. Para ello, es probable que este gran flujo de datos requiera de un procesamiento y análisis más exhaustivo de los mismos.

Es por esto que la PC *Personal Computer* se ha transformado en la herramienta indispensable en

cualquier ámbito, pero en especial en los entornos en donde se requiere el manejo, calculo, procesamiento y análisis de grandes cantidades de informacion de diferentes índoles.

Desde la inclusión de la norma USB, en el año 1996 a la fecha, se ha convertido en el elemento que no falta en ningun equipo, al punto tal que ha desplazado a cualquier otro conector. Al punto tal es esto, que para requerir algún puerto adicional que no sea de esta norma, cualquier compardor debe especificar que así sea, mas o es necesario especificar que tiene USB como norma de conexión.

Este trabajo, pretende elaborar una interfaz entre los dos extremos, es decir, entre la PFGA y la PC, de forma tal que permita a un desarrollador, investigador o usuario en general, obtener una comunicación confiable y con un ancho de banda que permita mover el flujo de datos que genera una sensor que adquiera imágenes.

Es cierto que el protocólo puede ser totalmente implementado en una FPGA, sin embargo, esto requeriría un muy alto costo tanto económico como en recursos disponibles del chip programable para una tarea genérica que es mejor elaborar con un circuito integrado diseñado especialmente para tal fin. Es por esto que se utiliza como lazo de interfaz un chip comercial elaborado por Cypress Semiconductor.

1.2. El protocolo USB

El protocolo USB es un sistema de comunicación diseñado durante los años 90 por los seis fabricantes de la industria de las computadoras, Compaq, Intel, Microsoft, Hewlett-Packard, Lucent, NEC y Philips, con la idea de proveer a su industria de un sistema que permita la conexión entre las PC's y los periféricos con un formato estandart, de forma tal que permita la compatibilidad entre los distintos fabricantes.

Hasta ese momento, el gran ecosistema de periféricos, sumado a los nuevos avances y desarrollos, hacia muy compleja la conectividad de todos ellos. Cada uno de los fabricantes desarrollaba componmentes con fichas, niveles de tensión, velocidades, drivers y un sinnumero de etc diferentes, lo cuál dificultaba al usuario estar al día y poder utilizar cada componente que compraba. Lo más probable era encontrarque cada vez que uno comparaba una PC, debía cambiar el teclado, el mouse o algún periferico específico. Esto también complicaba a las mismas empresas productoras, por que la introducción de un nuevo sistema requería un mucho soporte extra para poder conectar todo lo ya existente.

Todo esto, quedó saldado con el standar USB, que debido a la gran cuota de mercado de sus desarrolladores, rápidamente fue introducido y se transformó en la norma a la hora de seleccionar un protocolo. Al punto tal esto se cumplió que hoy, más de 20 años después, es muy dificil encontrar PC's con otro tipo de puertos, salvo que en el momento de su compra uno requiera un puerto específicamente. Sin embargo, por norma, cualquier PC nueva dispónible en le mercado debe poseer puertos USB para la conexión de los periféricos.

Desde el punto de vista técnico, el protocolo USB es un sistema del tipo maestro-esclavo, donde el maestro, denominado HOST, debe ser necesariamente una PC y cualquier periférico a ella acoplada será un esclavo.

Para describirlo es conveniente tal vez separar el protocolo en tres partes. Una parte física, en donde se definen los componentes que intervienen, una capa de protocolo, en donde se define el formato

y el marco en el que son enviados los paquetes, como se direccionana y como se comunican entre sí, y una parte lógica, en donde cada componente es visto solamente como un extremo y define como fluyen los datos desde un extremo hacia la PC y viceversa.

1.2.1. Capa física

En esta sección no se describirán los detalles de las conexiones eléctricas ni mecánicas a las que se refieren las especificaciones de la norma USB debido a dos cuestiones fundamentales. Una de ellas es que toda esta sección de la norma está resuelta ya por los fabricantes del chip de Cypress. Este chip maneja todas las señales, arma y desarma los paquetes que salen hacia la PC y que llegan de ella respectivamente. Por otro lado, no es el objetivo de este trabajo adentrarse en esos detalles. Gracias a la extensión de este tipo de comunicación existen una gran cantidad de fabricantes en el mercado que fabrican cada uno de los componentes, ya sean, cables, conectores en todas sus versiones, adaptadores de un tipo de enchufe a otro, su costo es despreciable con respecto a cualquier tipo de desarrollo en ese sentido y son de una muy buena calidad, en el sentido que todos cumplen con lo que la norma establece.

Si se hará a continuación una descripción de los dispositivos físicos y su categoría, degun la norma, en función del rol que cumplen.

La comunicación USB posee una topología maestro-esclavo. Es decir, Existe un dispositivo que dirige todas las transferencias de datos y otros dispositivos que responden luego de que el maestro a emitido una directiva. Por esto, el elemento central de cualquier comunicación USB es el HOST (director o anfitrión, por su traducción de la voz inglesa). Este dispositivo que en la mayoria de los casos es la PC, aunquetambién puede ser algún dispositivo inteligente como un smartphone, es el que posee en Host USB Controller. El HOST se encarga de enviar los tokens a todos los periféricos, con la dirección del dispositivo, el sentido de la comunicación, el tipo de transferencia que se espera y todas las acciones de control que el sistema requiera.

En el otro extremo de la comunicación, se encuentran los dispositivos. Los dispositivos son todos los periféricos que actúan como fuente o sumidero de información. Es decir, en caso de periféricos de entrada, serán una fuente de información hacia el Host. Si los periféricos son de salida, serán un sumidero de la información que proporciona la PC. Los casos de perifericos de entrada/salida, se denominarán periféricos compuestos.

Existe también, a los fines de la norma,un elemento intermedio, denominado HUB (concentrador o distribuidor, según la traducción del inglés). Este dispositivo se encarga de conectar dos o más dispositivos, ya sea de entrada o salida, de recibir y enviar las direcciones y de regenerar las señales que el host envía y deben ser recibidas por los dispositivos, o bien, los datos que fluyen por el sistema.

1.2.2. Capa de protocolo

En la capa de protocolo, las especificaciones detallan como se compone el marco y como los paquetes deben ser armados para que sean efectivamente enviados a través del sistema.

Cada mensaje que se intercambia por el bus se denomina paquete. Cada paquete posee en su inicio lo que se denomina PID. el PID (del inglés packet ID o identificador del paquete) puede ser de 4 tipos, definidos por cada uno de los tipos de paquetes que puede haber:

- en prime lugar se encuentran los paqutes token, a través del cual el host envía las directivas a los distintos periféricos. Estas directivas pueden ser IN, cuando solicita datos a un periferico; OUT, cuando va a enviar datos a un dispositivo; SOF, que indica el inicio de cada cuadro, para que cada dispositivo se sincronice y SETUP, cuando va a enviar un paquete de configuración a algún dispositivo.
- el segundo tipo de paquetes es el paquete de datos. Este tipo de PID puede ser emitido por un dispositivo, si es que envía datos al host o bien por el mismo host cuando el flujo de datos es a la inversa.
- el tercer tipo de paquets es un paquete de reconocimiento, denominado ACK, (por acknowledge o reconocimiento). Este paquete es enviado por los periféricos y le da idea al host de cual es el estado del dispositivo, es decir, si se encuentra operativo o no, si se encuientra ocupado o si recibió la trasnferencia de forma correcta.
- finalmente existen paquetes especiales, a través de los cuales el protocolo se comunica con los hubs, emite directivas intermedias, envia señales de polling para conocer el estado del bus, entre otras.

Como se menciono anteriormente, el host envía un toquen SOF que sirve para sincronizar los dispositivos al bus. En un sistema USB, el host provee la base de tiempo y envía cada 1 ms un SOF (Start of frame, o su traducción, inicio de cuadro) seguido de un numero de 11 bits que sirve para contar cada uno de los marcos. Además, en sistemas de alta velocidad, cada cuadro se divide en ocho microcuadros de 125 µs, que también son marcados por un SOF, sin embargo, el contador no se actualiza por cada microcuadro.

Luego de esto, el sistema puede comenzar con la transferencia de datos. USB dispone 4 tipos de posibles transferencias, que se detallan un poco más adelante, y que pueden ser usadas conforme a los diferentes requerimientos del sistema.

Cada transferencia de datos está compuesta por un primer paquete de token, emitido por el host, que posee el tipo de transferencia que se espera, sea de entrada, de salida, de control o especial; la dirección del dispositivo que debe responder o escuchar el mensaje enviado por el bus y un código de detección de errores del tipo CRC (código de checkeo redundante cíclico).

Luego, el siguiente paquete posee los datos transferir, precedido por un PID de datos, y otro código CRC para detectar errores. Este paquete es transmitido por el host o el dispositivo dependiendo del sentido de la transferencia.

Finalmente, el dispositivo devuelve un paquete de reconocimiento, indicandole al Host si el transferencia fue efectiva o no y por qué esta no fu efectiva, siendo ese el caso.

Transferencias por paquetes (Bulk transfers)

Este tipo de transferencias puede ser dispuesta para trasmitir un gran flujo de datos. No posee perdida de datos gracias a un sistema de chequeo y retransmisión de datos. El inconveniente que presenta este tipo de transferencias es que en un nivel de prioridades se presenta en el final del sistema. Es decir, el bus solo va a ser usado para transferir este tipo de datos siempre que se encuentre desocupado, o bien, se le asignará una proporción ínfima de ancho de banda para poder trasnmitir con este modo. Es comunmente usado para trasmitir datos que no son críticos en tiempo, por ejemplo para scanners

e impresoras.

Transferencias de interrupción

Este tipo de trasnferencias sirve para enviar y recibir paquetes de datos que requieren un buen sistema de control de errores, pero que, son más restrictivos en tiempos. El sistema siempre destinará un intervalo fijo de tiempo para transmitir los datos que estén pendientes para trasnferencias de interrupción.

Transferencias Isocrónicas

Este tipo de trasnferencias está destinado a datos que son realmente críticos en tiempo. Es usado, principalmente para enviar datos a chorro, como ser el caso de streaming de audio o video, en donde los datos producidos deben ser rápidamente llevados al usuario.

No posee un control de errores muy sofisticado, más que un simple código CRC, pero no existe mecanismo de retrasmisión de datos ni handshaking entre los dispositivos y el host.

Como el tiempo es el requerimiento críticoen este tipoo de datos, el controlador le asigna una determinada cantidad de tiempo de bus, o en otras palabras, una determinada cuota de ancho de banda.

Transferncias de control

Este tipo de trasnferencias solo las emite el host y el sistema las utiliza para configurar cada dispositivo. Debido a su criticidad, el controlador dispondra encada cuadro de una fracción de ancho de banda para las trasnmisiones de control. Es el tipo de trasnferencias que posee el sistema de detección de errores más sofisticado, de forma tal de asegurar la integridad de los datos de control.

A cambio de esto, solo muy poca información puede ser trasmitida por cada cuadro, de hasta 64 bytes en sistemas de alta velocidad.

1.2.3. Capa lógica

Desde el punto de vista lógico, cada dispositivo es visto por el host como un extremo independiente, que posee un módo de comunicación, es decir, con ese dispositivo el protocolo se comunicara solo por un tipo de trasnferencia;y un solo sentido. En otras palabras, USB notará como separado un dipositivo de entrada y otro de salida, independientemente de si físicamente el dispositivo es un perifericode entrada y salida.

Esta independencia brinda la posibilidad de configurar cada extremo de forma diferente y obtener el ancho de banda necesario para la subida y bajada de datos, los tiempos de acceso al bus, la dirección y todo lo relacionado a los modos de comunicación conforme a los requerimientos.

El protocolo entiende que entre le host y cada uno de los extremos existe un tubo (la norma en ingles habla de *pipes*) en donde la información es colocada y transferida. Luego, cada tubo posee la

configuración establecida por el controlador del host y se comunica con cada extremo por medio de estos tubos. A los fines del usuario, esto es lo importa, por cuanto uno solicita acceso al bus y define en que buffer va a contener los datos a enviar o transmitir y el protocolo se encarga de el empaquetado, el armado de los cuadros, el acceso el bus y el posterior envío de datos.

Capítulo 2

Elección de las herramientas para la realización de la interfaz

2.1. Elección de la FPGA

El principal objetivo de el presente Trabajo Integrador es el de proveer una comunicación USB para desarrollos basados en FPGA. Por esto mismo, es fundamental sintetizar un circuito en el FPGA que sirva de nexo entre el desarrollo y la placa de interfaz.

Es por esto que se utilizó la placa MOJO v3. Dicha placa posee un FPGA Spartan-VI de Xilinx. Al poseer un Spartan-VI, se tiene la posibilidad de elaborar sistemas de muy alta velocidad que permite al desarrollador de sensores y sistemas de adquisición de datos la implementación de circuitos que resuelvan problemas a la medida de los requerimientos.Dispone también de 84 puertos digitales configurables como entrada y/o salida, 8 puertos de entrada analógicos, 8 LED's de propósito general, un botón de tipo pulsador.

La placa MOJO es una placa inspirada en el concepto de prototipado rápido. Para ello los puertos se disponene en un arreglo de pines a través de los cuales es posible acoplar cualquier dispositivo que uno necesite. Se dispone en el mercado otros circuitos, que los fabricantes denominan shields (*escudo* traducido al castellano), que encajan a la perfección en todos los pines y que contiene algún set de perfiéricos útiles. Estos shields, también, con el fin de satisfacer requerimientos especiales, pueden ser diseñados por uno mismo, o bien es posible adaptar con algunos cables las entradas a un dispositivo particular.

Además de estos shields, los desarrolladores pensaron en que no sea necesario ninguna herramienta extra a la hora de programar la FPGA. Para ello, dotaron al sistema de un microcontrolador ATmega32U4 de Atmel y cargaron un programa bootloader, que se encarga de transferir la configuración del FPGA cargada desde una memoria flash incorporada al sistema con ese propósito particular, o trasmitida por el usuario desde una PC, a través de un transceptor USB que contiene el microcontrolador. Luego, este último es colocado en modo esclavo y se configura de forma tal que dota al sistema de una comunicación con entre la FPGA y una PC, vía USB y se utiliza su ADC para leer los puertos analógicos.

Una vez llegado a este punto, el lector podría preguntar con toda razón ¿por qué es necesario realizar un sistema de comunicación USB extra, si ya cuenta con un microcontrolador que se encarga de dicho asunto? La respuesta a esta pregunta es simple y se basa en la capacidad del ancho de banda

del sistema de comunicación que dispone la placa.

La linea de controladores ATmega posee USB 2.0 full-speed. Esto quiere decir que puede enviar datos a una tasa de 12 Mbps. Además, la comunicación entre ambos chips se realiza via SPI (Serial Peripherical Interface, o en español Interfaz Serie de Periféricos), comandada por un cristal de cuarzo de 50 MHz, ofreciendo una velocidad de salida que puede resultar muy lenta a los fines de este trabajo. Se pretende dar el mayor ancho de banda posible utilizando la capacidad de del USB 2.0 High-Speed, de hasta 480 Mbps.

2.2. Arquitectura y Funciones del CY7C68013A

Como interfaz entre la FPGA y la PC se utilizó la placa de desarrollo CY3684 FX2LP EZ-USB Development Kit de Cypress Semiconductor. Esta placa posee como núcleo un CY7C68013A, circuito integrado que posee todas las herramientas necesarias para realizar la interfaz, como así también un buen número de periféricos que permiten al desarrollador realizar pruebas y depuración.

Entre estas, se pueden mencionar 6 pulsadores, de los cuales cuatro se utilizan para proposito general, uno para reestablecer los valores por defecto de la placa y uno para enviar señales de suspensión y reestablecimiento del programa actualmente cargado en el microcontrolador. A su vez, posee dos memorias EEPROM que sirven para cargar firmware y archivos de configuración del sistema, un display de 8 segmentos, 4 leds de multiple propósito, dos puertos UART, una salida de pines compatible con puertos ATA y 6 puertos de 20 pines que se utilizan para la conexión hacia el chip núcleo. Como soporte para el firmware, posee también un bloque de 64 kB de memoria SRAM.

Se seleccionó este controlador como interfaz con el objetivo de utilizar la m9enor cantidad de los recursos configurables de la FPGA, de forma tal que estos queden disponibles para el desarrollo de los sistemas que necesiten los potenciales sensores que se desee leer a posteriori.

A continuación se describe en datalle la arquitectura y las funciones del CI CY7C68103A, la configuración de funcinamiento escogida, y el desarrollo del firmaware en base al framwork provisto por Cypress que facilita la implementación de periféricos.

El núcleo del Kit de Desarrollo FX2LP EZ-USB es un CY7C68013A. Dicho circuito integrado, cuya arquitectura se presenta en la Figura 2.1. Los chips de la familia FX2LP integran un transceptor USB, un SIE *Serial Interfaz Engine*, buffers de datos, un microcontrolador 8051 mejorado y una interfaz programable hacia los periféricos. Además posee un un PLL y un divisor configurable a través de los cuales provee al sistema de las señales de reloj adecuadas para el correcto funcionamiento del sistema.

Esta arquitectura permite al usuario trasmitir datos desde y hacia la PC desde el mimso puerto USB, o bien via RS-232, desde la PC. A la hora de comunicarse con sistemas periféricos se puede aprovecar el puerto I2C, la interfaz de propósito general o una interfaz esclavo que puede ser conectada a un sistema maestro. Esto brinda muchas alternativas, desde la conexión a puertos estandar, como ser ATA, PCMCIA o EPP, o también la conexión de dispositivos tales como DSP's y FPGA's.

La comunicación USB es llevada a cabo a través del transceptor, unido al SIE. Como se observa en la Figura 2.2. El usuario, a fin de intercambiar datos, solo debe colocar o extraer los datos de registros destinados a tal fin y modificar las banderas de handshaking, que en la figura se observan como ACK

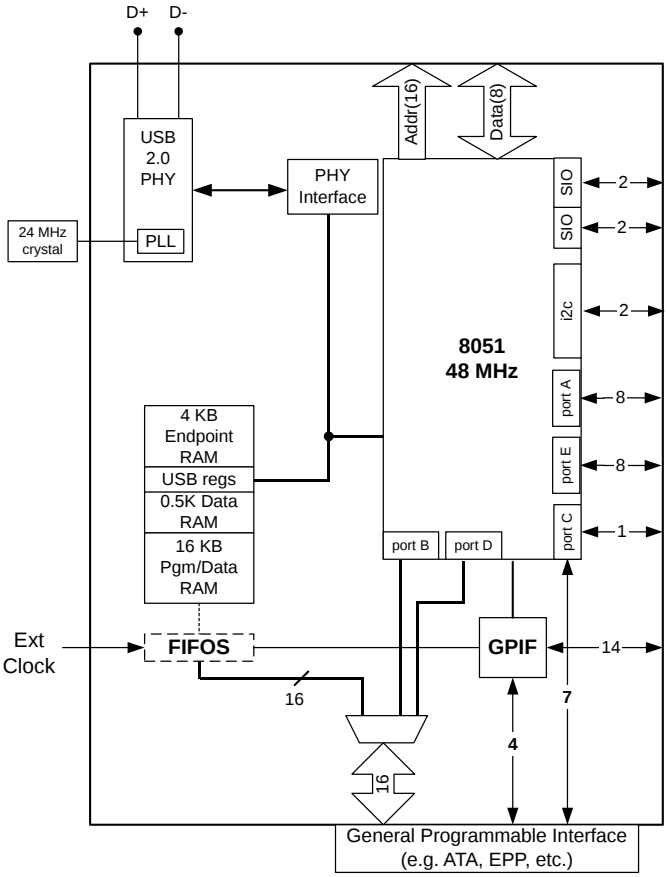

Figura 2.1: Arquitectura FX2LP

(abreviación del ingles *acknowledge*, que significa reconocer, aceptar o agradecer), que indican si el sistema está disponible, si los datos fueron colocados o leídos, dependiendo el caso tratado. El SIE y el transceptor USB se encargan de empaquetar, enviar, recibir y desempaquetar toda la información, así como leer los tokens que emite el host, calcular y corroborar los códigos cíclicos de detección de errores y todo lo relacionado al protocolo en sí.

El esquema de bus permite utilizar el microcontrolador 8051 para procesar datos, hacer control de errores, empaquetar datos de una forma particular, generar datos nuevos, entre otras, o bien, simplemente enviar datos desde un periférico de forma directa al SIE y luego transmitirlos a la PC por la tubería USB.

Para este trabajo final, se configuró el funcionamiento del EZ-USB en modo esclavo, conectando al maestro, la FPGA, a la memoria FIFO destinada para tal fin. Por lo que se describirá en detalle a continuación.

Al poseer el sistema un SIE, que es un serializador de datos, la memoria FIFO de 4kB es usada por el sistema como buffer. Se conecta de forma directa a los periféricos y es configurable, lo que permite al usuario disponer del espacio conforme requiera las necesidades de ancho de banda de los sistemas diseñados, evitando así las congestiones en casos de mucho flujo de datos. En el otro extremo, puede ser conectada al tubo USB o al microntolador, dirigiendo los datos directamente a la PC o realizando algúna acción sobre ellos antes de enviarlos, respectivamente.

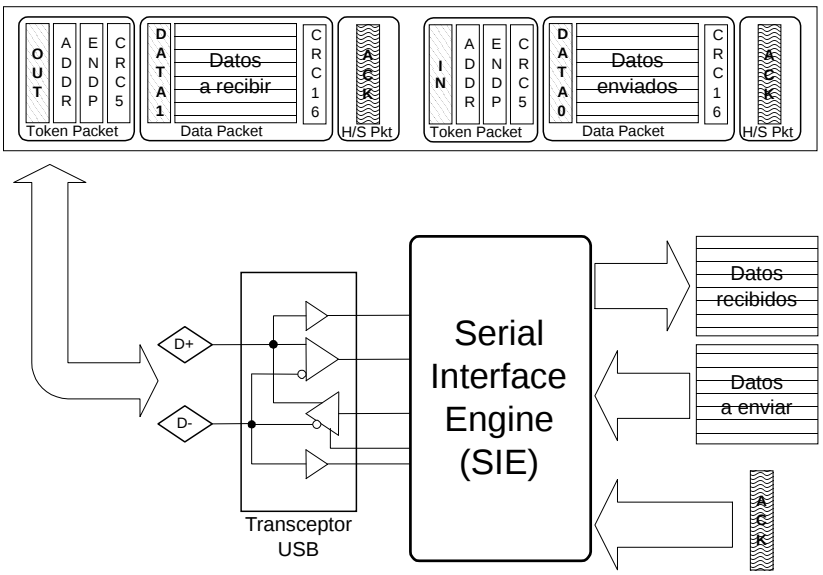

Figura 2.2: Implementación del enlace USB realizado por el EZ-USB

El sistema FX2LP permite configurar los buffers conforme la se ve en la Figura 2.3. Es de destacar que en cualquiera de las configuraciones posibles, se tiene al menos dos buffers. Los diseñadores del integrado pensaron esto como una solución a la congestión. Para ello, los buffers se pueden configurar duplicados, triplicados o cuadruplicados, dependiendo de las necesidades. Luego, el sistema de forma automática se encarga de permutar los buffers de forma tal que no queden datos retenidos en el dispositivo maestro. Cómo se detallará más adelante, para este trabajo se configuraron dos enpoints como el modo 11 de la fFigura 2.3, es decir, un extremo con 3 buffers de 1024 bytes y otro con 2 buffers de 512 bytes.

Los buffers pueden ser conectados directa hacia el SIE, de forma que realice la comunicación entre la PC y los periféricos de forma automática, incluso programando un umbral de cantidad de datos que cunado sea revasado envíe los datos hacia la PC; o bien se puede acceder a ellos desde el microcontrolador, de forma de poder realizar alguna acción específica sobre los datos antes de mandarlos hacia la PC o los periféricos.

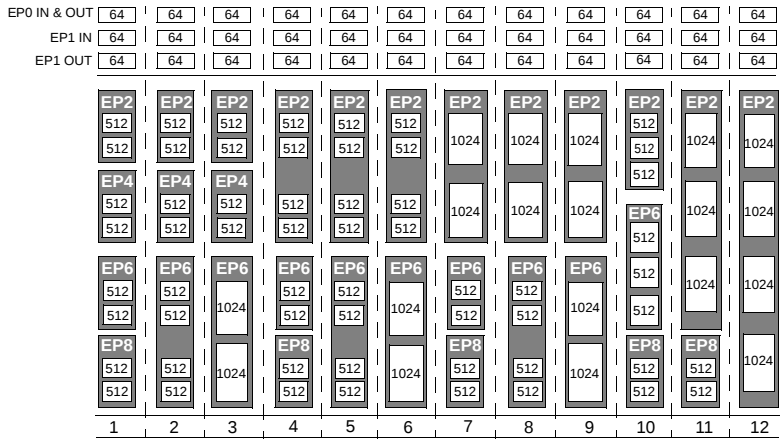

Figura 2.3: Configuraciones admitidas para los buffers de los diferentes periféricos