Capítulo 1

Introducción

El presente informe busca dar a conocer al lector las tareas y actividades desarrolladas por el autor, a fin de dar cuenta sobre lo realizado en el marco del Trabajo Final de la carrera Ingeniería Electrónica, dictada en la Facultad de Ingeniería de la Universidad Nacional de San Juan. A lo largo de él, se busca que el lector comprenda la problemática que se pretende resolver y la configuración, fundamentos y modo de uso del sistema propuesto. El objetivo es diseñar e implementar una interfaz USB para la transmisión de datos a alta velocidad, adquiridos por sistemas desarrollados en FPGA para aplicaciones científicas.

En este Capítulo, se explica, en primer lugar, la motivación, es decir, se busca aclarar la problemática que se busca resolver. Luego, se detallan los objetivos que persigue este trabajo. Seguido a esto, se otorga un esquema que describe la solución planteada y se justifica el protocolo elegido. A continuación, el lector puede conocer la organización del presente informe. Finalmente, se repasan algunos conceptos importantes de la norma USB que luego se utilizan en el trabajo desarrollado.

1.1. Motivación

El grado de avance que ha experimentado la tecnología en general, y la electrónica en particular, gracias a la industria de los semiconductores, permite que la producción científica pueda adquirir una gran cantidad de datos. Esto, requiere el relevamiento y registro de diferentes tipos de magnitudes físicas y/o químicas sobre el objeto o proceso sobre el que se está investigando. Sin embargo, esta no es una tarea simple ya que, en muchos casos, estas magnitudes son difíciles de observar y cuantificar con los sentidos de un operador. Entonces, se vuelve conveniente transformar las variables a medir en otras que resulten más simples.

El dispositivo que recibe estímulos energéticos de una condición, situación o fenómeno físico y/o químico y los convierte en una señal asociada y definida de otra forma de energía, se lo denomina transductor[1][2]. En otras palabras, este tipo de máquinas son conversores de energías[2][1][3]. Un sensor es un tipo especial de transductores que posee como variable de salida una señal eléctrica que está adaptada para ser ingresada en un circuito electrónico, o adecuada al sistema de medida que se utilice [4][5][6].

Las altas escalas de integración de circuitos alcanzadas (aún en incremento) posibilitan el diseño de sistemas sensoriales cada vez más complejos, en los cuales se logra agrupar miles de sensores en áreas reducidas, obteniendo medidas simultáneas y flujos crecientes de datos. Uno de los desarrollos que se encuentra en boga es el de sensores y sistemas que adquieran imágenes de diferentes tipos. Desde el punto de vista digital, una imagen es un arreglo bidimensional de números, los cuales pueden ser exhibidos en una pantalla en forma de intensidad y colores de luz. Por esto, un sensor de imagen puede estar compuesto, bien por un arreglo bidimensional de sensores lumínicos, por un transductor que es simultáneamente desplazado y medido, o por una combinación de ambos métodos. En todo caso, es de suma utilidad que la lectura de imágenes sea realizada en el menor tiempo posible ya que cada imagen conlleva una cantidad no menor de datos.

Uno de los trabajos más importantes en este sentido, fue el desarrollo de los CMOS APS (*Active Pixel Array*, o matriz de pixeles activos) [7]. La Figura 1.1 muestra un dibujo esquemático del funcionamiento de cada uno de los pixeles activos que componen la matriz. Este posee un fotodiodo, que es el elemento transductor. En un primer momento, este diodo es conectado en forma inversa, generando una región de vaciamiento entre sus terminales. Luego, cuando un fotón incide sobre él, se forman pares electrón-hueco y ellos circulan a través del diodo a las terminales, generando una pequeña caída de tensión. Finalmente, se mide la tensión del diodo, cuya intensidad será inversamente proporcional a la cantidad de fotones que incidieron sobre el pixel en cuestión.

Figura 1.1: Esquema de un pixel activo

A partir del desarrollo de los APS, se fue perfeccionando el método hasta obtener integrados con mayor cantidad de pixeles y para lograr aplicaciones específicas basadas en este fenómeno. Por ejemplo, en los trabajos [8] y [9] se presentan sensores CMOS basados en la arquitectura MIMOSA (de *Minimum Ionizing particule MOS Active pixel Sensor*, Sensor con Pixel activo MOS de particulas ionizantes mínimas). Estos sensores se desarrollaron con el objetivo específico de detección de radiación. En otro trabajo, Hizawa y otros [10] fabricaron un sensor que adquiere imágenes midiendo el pH de en cada uno de los pixeles, obteniendo imágenes de fenómenos químicos en tiempo real.

Desde un punto de vista electrónico, para poder transmitir grandes volúmenes de datos en forma digital, se requiere de circuitos que sean capaces de operar a altas frecuencias de conmutación. Esto no es trivial, ya que cuando las longitudes de onda que se mueven a través de un circuito son comparables con sus dimensiones físicas, se evidencian capacidades e inductancias parásitas no contempladas que perjudican el desempeño de los sistemas. Por ejemplo, si se desea transmitir datos en serie con una variación entre sus datos de 300 MHz, se habla de ondas cuya longitud de onda es de 1 m. Se dice que las dimensiones son comparables cuando el largo del conductor posee al menos un cuarto de la longitud de onda, es decir, para este caso, 25 cm. Si bien esto puede parecer grande, se debe notar que a mayores frecuencias, este efecto comienza a ser aún más perjudicial.

Otro problema que presentan los circuitos de alta velocidad tiene que ver con los tiempos

de propagación de las corrientes y tensiones que circulan a través de ellos. Cuando se aplica un impulso en un conductor, la onda viaja a la velocidad de la luz. Esto quiere decir que la tensión no llega al mismo tiempo a todos los puntos del circuito, sino que, mientras más alejada está la fuente, más lejos está se demora en responder. Puede suceder, entonces, que la lógica del circuito se demore más que el pulso de reloj que indica un cambio de estado, obteniendo así un comportamiento no deseado, si no está correctamente diseñado y contemplado este aspecto.

Una solución para los circuitos electrónicos que necesitan mover grandes volúmenes de datos, puede ser la incorporación varios conductores a través de los cuales puede fluir la información. Retornando al ejemplo de los datos transmitidos con variaciones de hasta 300 MHz, si son enviados a través de dos conductores, la frecuencia necesaria cae a 150 MHz, con 3, se obtiene una reducción de la frecuencia a 100 MHz, con 10, es suficiente con 30 MHz, etc. La cantidad de conductores a través de los cuales circula la información, se denomina ancho de bus.

En gran medida, la incorporación y evolución de microcontroladores permite obtener una captura y obtención cada vez superior de datos. Sin embargo, este tipo de dispositivos poseen una estructura rígida, capacidad de procesamiento limitada a una instrucción por vez y ancho de bus definido, la única opción para aumentar los volúmenes de datos que circulan a través de ellos, es un aumento de la frecuencia de funcionamiento, generando los problemas anteriormente detallados. Una solución óptima, sin considerar los costos asociados a esto, sería el desarrollo de un circuito integrado de aplicación específica (ASIC del inglés*Application Specific Integrated Circuit*). En este tipo de circuitos, el diseñador elabora un circuito que puede correr a altas velocidades y, a su vez, obtener un ancho de bus sin restricciones, más que las dimensiones físicas del área donde será realizado el circuito. Sin embargo, cuando sí se considera el costo asociado a este enfoque, se vuelve una solución ineficiente en bajas cantidades. La manufactura de este tipo de dispositivos cuesta desde €2500, en los procesos más económicos, hasta cientos de miles de dólares. Gran parte de estos costos son no recurrentes, es decir, solo se pagan una vez por proyecto. En grandes volumenes, este tipo de soluciones se vuelven más convenientes.

Otro enfoque, es la utilización de Arreglos de Compuertas Programables por Campo (FPGA, acrónimo del inglés *Field-Programmable Gate Array*)para realizar el desarrollo. Un FPGA es un dispositivo electrónico que posee la capacidad de sintetizar casi cualquier circuito digital. En esencia, es una matriz de bloques lógicos (también llamadas *slices* o celdas lógicas, dependiendo del fabricante), que contienen Tablas de Verdad(LUTs o *Look-Up-Table*) y flip-flops (ff), entre otras cosas, y pueden ser interconectadas entre sí, según el criterio del usuario. Así, permite implementar una solución digital en un circuito físico, a diferencia del microcontrolador que lo realiza a través de un algoritmo. Esto, incorpora la ventaja de definir el ancho de bus necesario para relevar una gran cantidad de datos y transmitirlos a frecuencias de trabajo menores, además de ejecutar tareas en paralelo, disminuyendo los tiempos de procesamiento. A su vez, al ser implementado en un área muy pequeña, debido a la integración del sistema, puede trabajar a frecuencias muy elevadas, lo que implica una mayor tasa de datos aún. Los costos de un FPGA, dependiendo de las prestaciones, van desde U$S 1 a U$S 50 000. A pesar de la gran diversidad de precios existentes en el mercado, una FPGA de bajo costo suele tener muy buenas prestaciones para la mayor parte de las aplicaciones.

Existen diversas publicaciones en donde se observa el uso de FPGAs para la implementación de sistemas que producen imágenes. Por ejemplo, el desarrollo de un sensor de radiación utilizando una cámara CMOS comercial. Para ello, los autores utilizaron un FPGA para configurar y transmitir imagenes a un grabador de video a través de puerto UART. Esto permitió adquirir una imagen accionando un disparador realizado con un pulsador[11].

Se denomina ultrasonografía a la técnica de adquirir imágenes basandose en reflexiones de ultrasonido. Sus aplicaciones son múltiple, en las que se destaca el diagnóstico médico, ya que es una técnica no invasiva y sin riesgos de radiación ionizante sobre el paciente. Un trabajo reciente desarrolló un sistema de ecografía médica con bajo costo utilizando un FPGA[12]. El autor también presentó un algoritmo realizado y probado en PC. Luego se implementó e en una FPGA.

Yanagisawa, entre otros, desarrolló un sistema con telescopios pequeños para explorar objetos de campo cercano con la finalidad de monitorear cuerpos celestes que puedan colisionar con el planeta[13]. Los autores utilizan la velocidad de los circuitos implementados en FPGA para minimizar el tiempo de adquisición.

La obtención de datos por si misma no otorga información. Para ello, es probable que un gran flujo de datos requiera de un procesamiento y análisis exhaustivo de los mismos. La invención y evolución de las computadoras, como así también el desarrollo de nuevos algoritmos, dan lugar a procesamiento de datos cada vez más complejos en tiempos mucho menores.

Las primeras ENIAC, computadora de propósito general desarrollada en el año 1946 para el cálculo de tablas balísticas de las fuerzas armadas estadounidenses, podía ejecutar 20 operaciones cada 10 µs [14], es decir, ejecutaba instrucciones con una frecuencia máxima de 200 kHz. A su vez, tuvo un costo aproximado de U$S 500.000, pesaba 5 t y consumía 175 kW.

En contraste con aquello, es posible conseguir en el mercado actual, computadoras cuyas dimensiones y peso hacen que puedan ser levantadas con las manos, ejecutan instrucciones en cuenstión de nanosegundos, (5 ordenes de magnitud menos), consumen menos de 1 kW y cuestan algunos cientos de U$S. A tal punto ha evolucionado esta tecnología que se encuentran presente computadoras muy potentes en casi cualquier laboratorio, oficina u hogar. Esta potencia de cálculo, ayudado por el desarrollo de nuevos métodos y algoritmos de cálculo, permiten a los investigadores procesar miles de datos en tiempos muy reducidos, ayudando al análisis de los mismos y la obtención de información.

La generación de datos y el procesamiento de lo mismos se da en sistemas diferentes. Ellos requieren, por tanto, de una conexión a través de la cual los datos puedan ser transferidos de un sistema al otro. Se torna de suma utilidad, entonces, proveer una comunicación efectiva y robusta que permita transmitir grandes volúmenes de datos en poco tiempo, y de esta forma facilitar los tiempos de desarrollo, las pruebas y depuración.

Implementar un sistema de comunicación en una FPGA, si bien no es trivial, puede ser resuelto de muchas maneras, quedando a criterio del desarrollador utilizar algún protocolo de comunicación estándar, o bien, diseñar uno propio. Sin embargo, desde el punto de vista de una

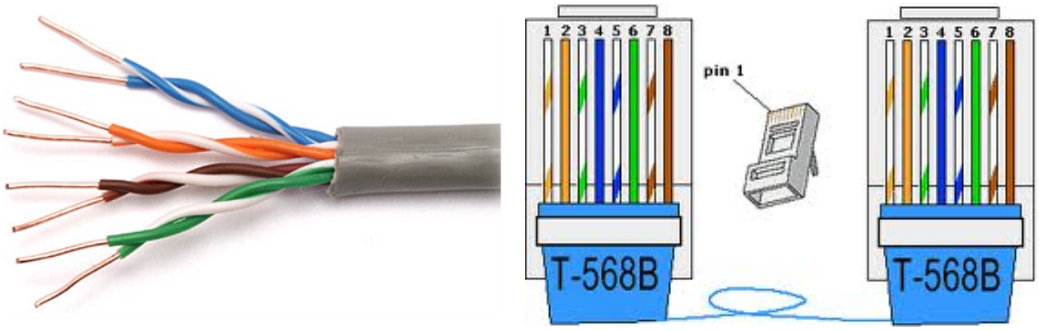

Figura 1.2: Par Trenzado y un dibujo de su ficha de conexión.

computadora, las comunicaciones se vuelven un poco más estrictas y acotadas a los puertos y señales que puede manejar el equipo, conforme el fabricante haya establecido.

Es por eso que este trabajo busca implementar una comunicación entre una computadora personal y una FPGA, utilizando un protocolo estándar, que esté disponible en cualquier computadora comercial y que posea una tasa de bit suficiente para poder transmitir imagenes.

Pero, ¿Cuántos datos son suficientes para este propósito? Se toma como base de diseño el sensor que utiliza Pérez en su Tesis de Maestría [15], una cámara para adquirir imágenes monocromáticas de código MT9M001C12STM, comercializado por Actina Imaging [16] que transmite datos a una tasa de $48\,\mathrm{Mbit\,s^{-1}}$.

Además la implementación debe ser compatible con equipos convencionales que se consigan fácilmente en el mercado y no posean especificaciones fuera de las de uso doméstico. Esto se cumple hoy en día con tres tipos de puertos: Ethernet, dedicado principalmente a conexión de redes mediante cables; Wi-Fi, utilizado para el accesos a la red de forma inalámbrica; y USB, especializado en periféricos.

Al hablar de Ethernet o Wi-Fi, se hace referencia a dos formas diferentes de conectarse a una misma cosa: una red de computadoras. En otras palabras, se habla de dos o más nodos, compuestos por PC's o cualquier aparato electrónico con capacidad de realizar cálculo binario, que pueden intercambiar datos a través de una trama bastante compleja de componentes diferentes. Ambos protocolos hacen referencia solo a la conexión física de los dispositivos y el control de acceso de cada uno de ellos a la conexión. Quedando a cargo de otro tipo de sistemas, con sus protocolos, que los datos enviados puedan ser correctamente recibidos por el usuario de la PC. La gran diferencia entre ellos radica en el medio físico que utilizan: Wi-Fi emplea ondas electromagnéticas emitidas mediante radiofrecuencia, mientras que en Ethernet, estas ondas son acarreadas por uno o más conductores, como ser cable coaxial, cables de par trenzado o fibra óptica.

En particular, el estandar Ethernet, también conocido como IEEE 802.3, es un protocolo que define cómo se deben conectar nodos a través de conductores para conformar red de área local (LAN o *Local Area Network*), es decir, un red pequeña, como ser doméstica, oficina o de una empresa definida, de forma que puedan transmitir información a velocidades a elegir entre

1 Mbit/s y 400 Gbit/s [17]. Utiliza una tecnología denominada Acceso Múltiple Sensando la Portadora con Detección de Colisiones (CSMA/CD del inglés *Carrier Sense Multiple Access with Collision Detection*), la cual antes de transmitir, corrobora que no exista una señal portadora y, si ésta está presente, espera para retransmitir. Dependiendo de la frecuencia de trabajo y la tasa de transferencia a la que transporta el mensaje, la norma especifica el conector y la distancia máxima a la que debe conectarse una repetidora, es decir, un aparato que reconstruya y emita la señal. Estos conectores pueden ser cable coaxial, fibra optica o cable de par trenzado. Este último es el más usual en las PCs comerciales y se muestra, junto a su ficha característica en la Figura 1.2.

La norma IEEE 802.3 también especifica cómo debe ser el formato de la información que se transmite, de forma tal que todas las velocidades, conectores y conductores sean compatibles entre si. Indica así que la información debe estar contenida en paquetes. Estos, a su vez, deben tener una estructura que permita la comunicación entre muchos nodos de la red. Así, deben contener un encabezado, con información sobre el emisor y el receptor, los datos de interés del usuario, y una trama de verificación.

Por su parte Wi-Fi, perteneciente a la asociación de compañías denominada Wi-Fi Aliance, se rige por la norma que estableció esta última. Existe una norma equivalente, encuadrada en la especificación IEEE 802.11, referida a las redes de area local inalámbrica, o WLAN (siglas del ingles *Wireless Local Area Network*). Wi-Fi en las que se refieren a las comunicaciones de radiofrecuencia con portadora de 2.4 GHz.

Si bien se puede implementar la comunicación vía Ethernet, cumpliendo con las especificaciones propuestas, es muy probable que el único puerto que posea la PC se encuentre conectado a la red y se necesitará una infraestructura mayor para lograr una efectiva comunicación. Por tanto, USB se observa como una solución óptima.

A su vez, USB presenta, al día de la fecha, con tres versiones de su norma, con diferentes velocidades:

- La version 1 posee dos revisiones, 1.0 fue lanzada al mercado en el año 1996 y 1.1 que se presentó en Agosto de 1998. Ambas revisiones alcanzan una tasa máxima de $12\,\mathrm{Mbit\,s^{-1}}$.
- USB 2.0 fue presentado en Septiembre del 2000 y es capaz de transmitir $480\,\mathrm{Mbit\,s^{-1}}$.
- Dependiendo de la revisión, desde USB 3.0, lanzada al mercado en 2011, que transmite $5\,\mathrm{Gbit\,s^{-1}}$, hasta $20\,\mathrm{Gbit\,s^{-1}}$ de la revisión USB 3.2, que fue presentada en 2017.

Con estas velocidades, es suficiente para el propósito del siguiente trabajo, la implementación una comunicación USB 2.0, con $480\,\mathrm{Mbit\,s^{-1}}$.

El alumno sabe que es posible implementar una comunicación USB completa a través de una FPGA. Sin embargo, esto sería muy costoso en términos de tiempos de desarrollo y de recursos de FPGA disponibles para la implementación de otros sistemas, los cuales son el objetivo de la comunicación.

Se plantea, entonces, un esquema como el que se observa en la Figura 1.4 en la cual se utiliza una interfaz externa al FPGA. La comunicación USB propiamente dicha será efectuada entre

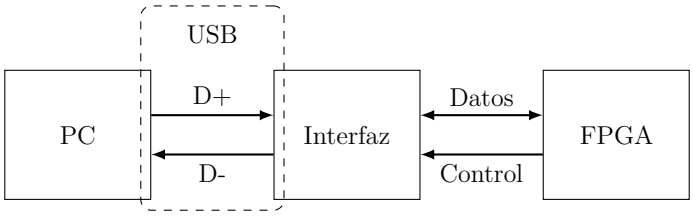

Figura 1.3: Esquema propuesto para implementar la comunicación

la interfaz y la PC, mientras que se plantea una comunicación diferente entre la interfaz y el FPGA. Este último, por su parte, tendrá la tarea de realizar el control de esta comunicación.

1.2. Transmisión de datos entre PC y FPGA

Dada la motivación, se puede decir a priori que el objetivo del presente trabajo es la implementación de una efectiva comunicación entre una Computadora Personal (PC) y una FPGA. Sin embargo, este objetivo tan amplio se plasmará de forma más concreta y detallada en la Sección 1.3.

El dato de diseño más relevante es el poder transmitir imágenes por la comunicación a implementar. Pero, ¿Cuántos datos son suficientes para este propósito? Se toma como base de diseño el sensor que utiliza Pérez en su Tesis de Maestría [15], una cámara para adquirir imágenes monocromáticas de código MT9M001C12STM, comercializado por Actina Imaging [16] que transmite datos a una tasa de $48\,\mathrm{Mbit\,s^{-1}}$.

Además la implementación debe ser compatible con equipos convencionales que se consigan fácilmente en el mercado y no posean especificaciones fuera de las de uso doméstico. Esto se cumple hoy en día con tres tipos de puertos: Ethernet, dedicado principalmente a conexión de redes; Wi-Fi, utilizado para el accesos a la red desde computadoras portátiles y teléfonos celulares; y USB, especializado en periféricos.

La comunicación Ethernet, o
Si bien se puede implementar la comunicación vía Ethernet, cumpliendo con las especificaciones propuestas, es muy probable que el único puerto que posea la PC se encuentre conectado a la red y se necesitará una infraestructura mayor para lograr una efectiva comunicación. Por tanto, USB se observa como una solución óptima.

A su vez, USB presenta, al día de la fecha, con tres versiones de su norma, con diferentes velocidades:

- La version 1 posee dos revisiones, 1.0 fue lanzada al mercado en el año 1996 y 1.1 que se presentó en Agosto de 1998. Ambas revisiones alcanzan una tasa máxima de $12\,\mathrm{Mbit\,s^{-1}}$.
- USB 2.0 fue presentado en Septiembre del 2000 y es capaz de transmitir $480\,\mathrm{Mbit\,s^{-1}}$.

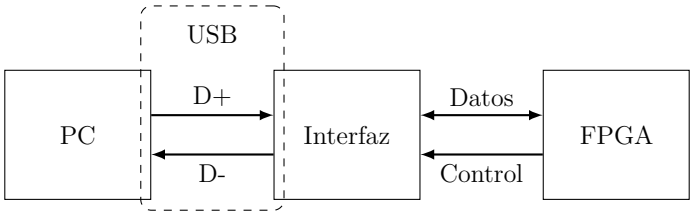

Figura 1.4: Esquema propuesto para implementar la comunicación

- Dependiendo de la revisión, desde USB 3.0, lanzada al mercado en 2011, que transmite $5\,\mathrm{Gbit\,s^{-1}}$, hasta $20\,\mathrm{Gbit\,s^{-1}}$ de la revisión USB 3.2, que fue presentada en 2017.

Con estas velocidades, es suficiente para el propósito del siguiente trabajo, la implementación una comunicación USB 2.0, con $480\,\mathrm{Mbit\,s^{-1}}$.

El alumno sabe que es posible implementar una comunicación USB completa a través de una FPGA. Sin embargo, esto sería muy costoso en términos de tiempos de desarrollo y de recursos de FPGA disponibles para la implementación de otros sistemas, los cuales son el objetivo de la comunicación.

Se plantea, entonces, un esquema como el que se observa en la Figura 1.4 en la cual se utiliza una interfaz externa al FPGA. La comunicación USB propiamente dicha será efectuada entre la interfaz y la PC, mientras que se plantea una comunicación diferente entre la interfaz y el FPGA. Este último, por su parte, tendrá la tarea de realizar el control de esta comunicación.

1.3. Objetivos

1.3.1. Objetivo Principal

El objetivo del presente trabajo es obtener una comunicación USB 2.0 de alta velocidad entre una PC y un FPGA.

Esta comunicación debe realizarse y documentarse de forma tal que pueda ser usado posteriormente en aplicaciones científicas desarrolladas con FPGA's.

1.3.2. Objetivos Particulares

Para la consecución del objetivo general, se deben cumplir los siguientes objetivos particulares:

- Comprender el funcionamiento del protocolo USB.
- Seleccionar los componentes a utilizar.
- Configurar los componentes seleccionados.
- Desarrollar un núcleo en VHDL que sirva de interfaz.

- Diseñar e implementar la interconexión de los componentes seleccionados.
- Verificar el sistema desarrollado.
- Desarrollar un documento que explique el modo de uso del código VHDL utilizado.

1.4. Estructura del Informe

El presente informe se divide en 2 bloques principales: uno referido al desarrollo del sistema y el siguiente a su forma de uso y verificación.

Dentro del bloque referido al desarrollo del sistema, se encuentran los primeros 5 capítulos:

1. **Introducción:** En este capítulo se intenta exponer lo que motiva el presente trabajo, la propuesta que da solución a la motivación, el objetivo y alcance que el trabajo busca y la estructura del mismo. Se brindan, además, conceptos importantes de la norma USB que son significativos para los objetivos de este trabajo.
2. **??:** Se describe aquí todas las herramientas de las que se vale este trabajo para cumplir con os objetivos propuestos.
3. **??:** Se presenta la arquitectura, configuración y código desarrollado para el presente trabajo, como así también las herramientas específicas provistas por el fabricante, que facilitan el desarrollo.
4. **??:** Este capítulo detalla lo desarrollado para implementar la comunicación entre la FPGA y la interfaz. Se expone una maquina de estados descrita en VHDL y sintetizada en FPGA. También se describe un circuito impreso realizado para conectar ambas partes.
5. **??:** Se desarrolla las tareas desarrolladas a fin de realizar las depuraciones del sistema y la verificación del cumplimiento de las especificaciones.

1.5. El protocolo USB

El protocolo USB es un sistema de comunicación diseñado durante los años 90 por seis fabricantes vinculados a la industria informáticas, Compaq, Intel, Microsoft, Hewlett-Packard, Lucent, NEC y Philips, con la idea de proveer a su negocio de un sistema que permita la conexión entre las PC's y los periféricos con un formato estándar, 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 interoperatividad de todos ellos. Cada uno de los fabricantes desarrollaba componentes con fichas, niveles de tensión, velocidades, drivers y un sinnúmero de etc diferentes, lo cuál dificultaba al usuario estar al día y poder utilizar cada componente que compraba. Lo más probable era encontrar que cuando se comparaba una PC, se requería

cambiar el teclado, el mouse y/o algún periférico específico. Esto también complicaba a las mismas empresas productoras, por que la introducción de un nuevo sistema requería de mucho soporte extra para poder conectar todo lo ya existente.

Todo esto, quedó saldado con el aparición de la norma USB, que debido a la gran cuota de mercado de sus desarrolladores, fue adoptado en forma rápida y se transformó en la especificación por defecto a la hora de seleccionar un protocolo. Al punto tal esto se cumplió que hoy, más de 20 años después, es muy difícil encontrar PC's con otro tipo de puertos, salvo que en el momento de su compra uno solicite especialmente un puerto determinado. Así, cualquier PC nueva disponible en el 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 (o un dispositivo con software y hardware capaces de incorporar los drivers necesarios) y cualquier periférico a ella acoplada será un esclavo[18].

Para describirlo es conveniente diferenciar tres partes. Una capa física, en donde se definen los componentes que intervienen, una capa de protocolo, en donde se define el formato, el marco en el que son enviados los paquetes, como se direccionan 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.

1.5.1. Capa física

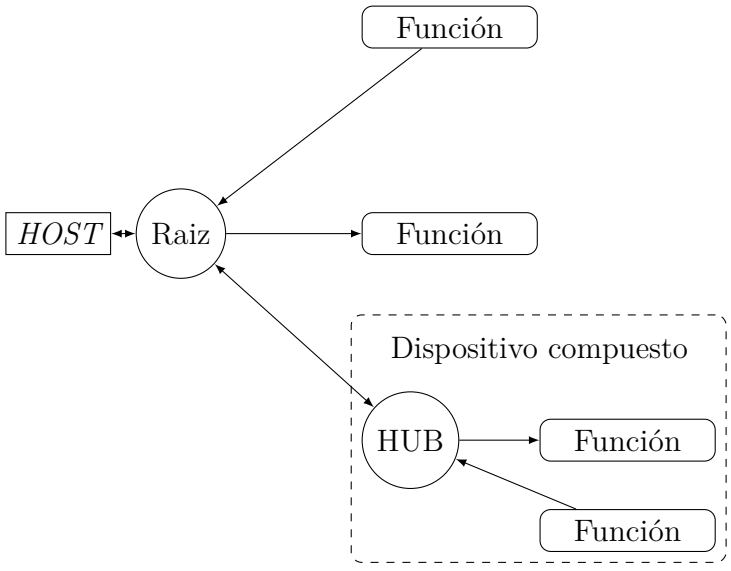

Figura 1.5: Topología de un sistema USB

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 de la

interfaz que se utiliza en este trabajo. A su vez, 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 estos, su costo es despreciable con respecto a cualquier tipo de desarrollo en ese sentido y son de una muy buena calidad, es decir que todos cumplen con lo que la norma establece. Sí, se describen los dispositivos físicos y su categoría, según 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 que responden a sus directivas. Por esto, el elemento central de cualquier comunicación USB es el *HOST* (director o anfitrión, por su traducción de la voz inglesa). Él es el que posee el *Host USB Controller*[18]. Esto quiere decir que tiene la capacidad de registrar los dispositivos acoplados, asignarles dirección, colocar los paquetes de salida y/o llegada en sus respectivas listas y servilos a los procesos que utilizan esta comunicación. Además, 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 la mayoría de los casos, el *HOST* es una PC, auqnue también puede ser cualquier dispositivos "inteligente" como un smartphone.

En el otro extremo de la comunicación, se encuentran lo que la norma denomina *funciones*[18]. Las *funciones* son todos los periféricos que actúan como fuente o sumidero de información. Es decir, en caso de periféricos de entrada, serán una fuente de datos hacia el *HOST*. Si los periféricos son de salida, serán un sumidero de la información que proporciona la PC. Los casos de periféricos de entrada/salida, se denominan *dispositivos 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 *funciones*, 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 las *funciones*.

La Figura 1.5 muestra la topología típica de un sistema USB. En ella, se observa el *HOST* como un rectángulo, las *funciones* como rectángulos con los bordes redondeados y los distribuidores como círculos. Se puede notar que el *HOST* posee un distribuidor propio llamado *Raiz* en el cual se conectan todos las *funciones* y distribuidores. Cada *Función*posee una única dirección. Pueden existir dispositivos que posean funciones diversas con un mismo encapsulado, como por ejemplo un auricular que tenga micrófono incorporado. Este dispositivo, tendrá un *HUB* que concatena dos *funiones* diferentes.

1.5.2. Capa lógica

Desde el punto de vista lógico, cada dispositivo es visto por el *HOST* como un extremo (EP, del inglés, *endpoint*) independiente, que posee solo un modo de comunicación, es decir, el protocolo se comunicara solo por un tipo de transferencia y en un único sentido con cada *EP*. En otras palabras, USB registra un periférico de entrada/salida como un *EP* de entrada y otro

de salida en forma independiente.

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 *EP* por medio de estos tubos. A los fines del usuario, esto es lo importante, por cuanto se 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.

1.5.3. Capa de protocolo

En la capa de protocolo, la especificación de la norma detalla cómo se compone un cuadro y cómo deben ser estructurados los paquetes para que sean efectivamente enviados a través del sistema. Cada mensaje que se intercambia por el bus se denomina paquete. Cada paquete puede poseer hasta cinco campos, dependiendo del tipo de paquete que sea enviado a través del sistema y de quien sea el remitente. A su vez, cada paquete comienza con una señal de sincronismo (*SYNC*) y un Comienzo de Paquete (SOP de *Start-of-Packet*), y terminan con una señal de Fin de Paquete (EOP por *End-of-Packet*).

Por otra parte, los paquetes está temporalmente encapsulados en cuadros. Cada cuadro posee un Comienzo de Cuadro (SOF, *Start-of-Frame*) y posee una duración de 1 ms, hasta el próximo SOF. En las comunicaciones de alta velocidad, es decir, aquellas que poseen una tasa de bit de 480 $\mathrm{Mbit\,s^{-1}}$. Se subdivide un cuadro en 8 micro-cuadros de 125 µs cada uno.

Campos de Paquetes

Cada paquete contiene un campo denominado identificador de paquete (PID del inglés *Packet Identifier*). El PID indica el tipo de paquete que se está enviando y, como consecuencia, el formato de cada uno, es decir, que campos acarrea y que control de datos utiliza.

A su vez, cuando el host solicita algo al sistema, lo realiza a través del denominado campo de dirección. Este campo, se compone de dos partes, la primera es el campo de dirección de la función y el segundo es la dirección de extremo.

Los mensajes de datos, poseen un campo dedicado de forma específica a los datos. Puede poseer un numero entero de bytes, desde 0 a 1024.

Para corroborar el envío de datos, USB utiliza verificación de redundancia cíclica (CRC o *Cyclic Redundancy Checks*). Los paquetes especiales y los de token poseen un verificador CRC5, es decir, de 5 bits, cuyo polinomio generador es:

$$G(X) = X^5 + X^2 + 1$$

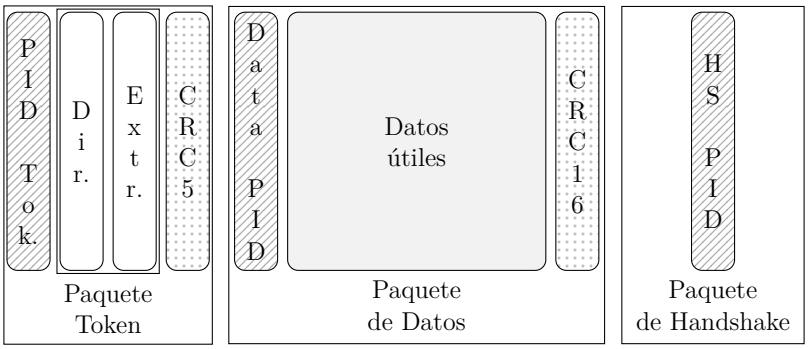

Figura 1.6: Distintos tipos de paquetes USB

Por su parte, los paquetes de datos, poseen CRC16, ya que pueden llegar a ser bastante extensos. En su caso, el polinomio generador está dado por:

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

Existe un campo relativo a los cuadros temporales, que se denomina campo de número de cuadro. Este es enviado por el *HOST* en cada SOF y es incrementado a cada cuadro. Los micro-cuadros también poseen un número de cuadro, sin embargo, este es aumentado solamente cada 8 micro-cuadros, es decir, el número se incrementa cada 1 ms y se repite durante los 7 micro-cuadros de 125 µs, en comunicaciones de alta velocidad.

Formato de paquetes

- **Paquetes Token:** A través de este tipo de paquetes el host envía las directivas a los distintos periféricos. Estas directivas pueden ser IN, cuando solicita datos de un periférico; OUT, cuando se dispone a enviar datos hacia una *función*; SOF, que indica el inicio de cada cuadro, para que cada función se sincronice y SETUP, cuando va a enviar un paquete de configuración a algún extremo.

- **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.

- **Paquete de *Handshake*:** Es enviado por el receptor del mensaje y le informa al emisor el estado de la transferencia. ACK significa que el paquete fue recibido sin errores; NAK, los datos poseen error o el emisor no puede enviar; la señal STALL quiere decir que la solicitud no es soportada o que el extremo está detenido; NYET implica que no hay respuesta aún por parte del receptor.

- **Paquetes Especiales:** Son paquetes con propositos específicos. Con ellos se señalan preambulos emitidos por el *HOST*, se informan errores, se solicitan mensajes divididos en diferentes paquetes y se intercambia señales de ping para conocer el estado de los componentes del sistema.

Cada uno de los tipos de paquetes posee un formato específico, tales como se muestran en la Figura 1.6. En ella se observa que los paquetes de token envían un PID, una dirección y un

CRC5; los paquetes de datos, se componen de un PID, los datos transmitidos y un CRC16; en el caso de los paquetes de Hanshake, solo el PID indica que tipo de mensaje se envía. Los paquetes especiales no se detallan ya que el formato es muy variable, en función del paquete.

1.5.4. Flujo de datos

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 de la función que debe responder o recibir el mensaje enviado por el bus y los verificadores CRC5.

Luego, el siguiente paquete posee los datos que se transfieren, precedido por un PID de datos, y verificadores CRC16. Este paquete es transmitido por el emisor de los datos. Finalmente, el receptor devuelve un paquete de *handshake*, indicandole al emisor si el transferencia fue efectiva o no.

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 transferencias 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 transferencias está destinado a datos que son 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 retransmisión de datos ni handshake entre los *EP* y el *HOST*.

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

Transferencias de control

Este tipo de transferencias 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 transferencias 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.

1.6. Sumario del capítulo

En el presente capítulo se expuso la necesidad de la elaboración de un sistema de comunicación que permita la transferencia de datos entre una PC y un FPGA para ser utilizados por sistemas implementados con este último dispositivo. Se planteó una solución utilizando una interfaz comercial que sirve de intermediario entre estas herramientas y se brindó una justificación del empleo del protocolo USB 2.0 de alta velocidad como la implementación óptima del sistema. Se presentó también la estructura del presente informe y se dieron algunos detalles relevantes para este trabajo de la norma USB.

Bibliografía

[1] R. Pallàs-Areny and J. G. Webster, *Sensors and signal conditioning*. Wiley-Interscience, 2001.

[2] D. M. Considine, *Encyclopedia of instrumentation and control*. McGraw-Hill, Inc., 1971.

[3] A. Perez Garcia, "Curso de instrumentación," p. 261, 2008.

[4] J. Fraden, *Handbook of modern sensors: physics, designs, and applications*. New York, NY: Springer New York, 2010.

[5] E. Slawiñski and V. Mut, *Humanos y máquinas inteligentes: conocimiento educativo sobre el comportamiento interno de robots que actúan juno y para el hombre*. Saarbrücken, Alemania: Editorial Académica Española, 2011.

[6] K. Ogata, *Modern control engineering*. Aeeizh, 2002.

[7] S. Mendis, S. Kemeny, and E. Fossum, "CMOS active pixel image sensor," *IEEE Transactions on Electron Devices*, vol. 41, pp. 452–453, mar 1994.

[8] C. Hu-Guo, J. Baudot, G. Bertolone, A. Besson, A. S. Brogna, C. Colledani, G. Claus, R. D. Masi, Y. Degerli, A. Dorokhov, G. Doziere, W. Dulinski, X. Fang, M. Gelin, M. Goffe, F. Guilloux, A. Himmi, K. Jaaskelainen, M. Koziel, F. Morel, F. Orsini, M. Specht, Q. Sun, I. Valin, and M. Winter, "CMOS pixel sensor development: a fast read-out architecture with integrated zero suppression," *Journal of Instrumentation*, vol. 4, pp. P04012–P04012, apr 2009.

[9] J. Baudot, G. Bertolone, A. Brogna, G. Claus, C. Colledani, Y. Değerli, R. De Masi, A. Dorokhov, G. Dozière, W. Dulinski, M. Gelin, M. Goffe, A. Himmi, F. Guilloux, C. Hu-Guo, K. Jaaskelainen, M. Koziel, F. Morel, F. Orsini, M. Specht, I. Valin, G. Voutsinas, and M. Winter, "First test results of MIMOSA-26, a fast CMOS sensor with integrated zero suppression and digitized output," *IEEE Nuclear Science Symposium Conference Record*, pp. 1169–1173, 2009.

[10] T. Hizawa, J. Matsuo, T. Ishida, H. Takao, H. Abe, K. Sawada, and M. Ishida, "32 × 32 pH image sensors for real time observation of biochemical phenomena," *TRANSDUCERS and EUROSENSORS '07 - 4th International Conference on Solid-State Sensors, Actuators and Microsystems*, pp. 1311–1312, 2007.

[11] M. Perez, F. Alcalde, M. S. Haro, I. Sidelnik, J. J. Blostein, M. G. Berisso, and J. Lipovetzky, "Implementation of an ionizing radiation detector based on a FPGA-controlled COTS CMOS image sensor," in *2017 XVII Workshop on Information Processing and Control (RPIC)*, pp. 1–6, IEEE, sep 2017.

[12] R. Biswas, *An Embedded Solution for JPEG 2000 Image Compression Based Back-end for Ultrasonography System*. PhD thesis, IIT, Kharagpur, 2018.

[13] T. Yanagisawa, T. Ikenaga, Y. Sugimoto, K. Kawatsu, M. Yoshikawa, S.-i. Okumura, and T. Ito, "New NEO search technology using small telescopes and FPGA," in *2018 IEEE Aerospace Conference*, vol. 2018-March, pp. 1–7, IEEE, mar 2018.

[14] H. H. Goldstine and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," *Mathematical Tables and Other Aids to Computation*, vol. 2, p. 97, jul 1946.

[15] M. Perez, *DETECCIÓN DE RADIACIÓN IONIZANTE UTILIZANDO SENSORES DE IMAGEN CMOS COMERCIALES*. PhD thesis, 2018.

[16] I. Micron Technology, "1 / 2-Inch Megapixel CMOS Digital Image Sensor MT9M001C12STM (Monochrome)," pp. 1–35, 2004.

[17] IEEE Computer Society, *IEEE Standard for Ethernet*, vol. 2018. 2018.

[18] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips, *Universal Seral Bus Specification*, vol. Revision 2.0. 2000.