Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Máster Universitario en

Ingeniería Industrial

Trabajo Fin de Máster

Sistema de transmisión VOOK basado en lámparas LED de iluminación.

Autor: Airam Casañas Hernández

Tutor: Silvestre Rodríguez Pérez

7 de septiembre de 2020

La publicación de este Trabajo Fin de Máster solo implica que el estudiante ha obtenido al menos la nota mínima exigida en superar la asignatura correspondiente, no presupone que su contenido sea correcto, aunque si aplicable. En este sentido, la ULL no posee ningún tipo de responsabilidad hacia terceros por la aplicación total o parcial de los resultados obtenidos en este trabajo. También pone en conocimiento del lector que, según la ley de protección intelectual, los resultados son propiedad intelectual del alumno, siempre y cuando se haya procedido a los registros de propiedad intelectual o solicitud de patentes correspondientes con fecha anterior a su publicación.

Resumen

En el marco de este trabajo se implementa un sistema de transmisión digital que utiliza como medio de transmisión la luz visible (VLC – *Visible Light Communications*). A lo largo de este documento se exponen los elementos que están involucrados en el sistema, así como las características especiales de este. Utilizando lámparas LED de iluminación comerciales como dispositivo emisor y empleando un circuito receptor basado en fotodiodos, se crea el enlace óptico necesario para llevar a cabo la transmisión. Para poner en funcionamiento el sistema, también se diseña e implementa un modulador y demodulador VOOK, con codificación del mensaje 4B6B, en una placa de desarrollo modelo Nexys A7.

Es imprescindible que, durante la transmisión de información, las lámparas LED utilizadas como fuentes ópticas, mantengan su principal funcionalidad de fuente de iluminación. Además, esta transmisión no debe afectar al usuario, se debe modular la señal de tal manera que la transmisión del mensaje sea independiente de la intensidad de luz percibida. Todas estas cuestiones se abordan a lo largo de este documento, ofreciendo los resultados obtenidos de las pruebas del sistema.

Abstract

The main subject of this work is the implementation of a digital transmission system, which uses visible light to transfer information (VLC - Visible Light Communications). Throughout this document, the elements that are involved in the system will be exposed, as well as its special characteristics. By using commercial LED lighting lamps as an emitting device and using a photodiode-based receiver circuit, we have the necessary optical link to carry out data transmission. The design and development of a VOOK modulator and demodulator is capital to the system operation, including the message encode (4B6B). The modulator and demodulator are implemented on a Nexys A7 model development board.

It is of great importance, during the data transmission, that the LED lamps, used as optical emitters, maintain their main functionality as a light source. Furthermore, this system must not affect the user, the signal must be modulated in such a way that the data transmission will be independent of the perceived light intensity. All of these issues are addressed throughout this document, providing also the results of the performed system tests.

Índice general

Capítulo 1. Introducción

Índice Introducción

Este trabajo se desarrolla en el ámbito de la especialidad en Electrónica del Máster Universitario en Ingeniería Industrial de la Universidad de La Laguna. A lo largo de este documento se expone el desarrollo realizado para la elaboración de un sistema de transmisión digital punto a punto que utiliza como medio de transmisión la luz visible. Este sistema se enmarca en el campo de las Comunicaciones por Luz Visible o VLC (Visible Light Communication), esto quiere decir que se transmite información a través de radiación electromagnética de longitud de onda entre 375 – 780 nm (espectro visible por el ojo humano).

Históricamente, la transmisión mediante señales ópticas ha sido bastante utilizada, inicialmente con mensajes simples como las reflexiones de la luz en superficies brillantes, hasta mensajes más complejos actualmente como a través de la fibra óptica, pasando por faros, señales entre barcos, semáforos… En definitiva, están presentes en la vida diaria de gran parte de la población. Este tipo de tecnología puede aprovechar elementos que se encuentran bastante extendidos (faros de vehículos, semáforos, lámparas de interior, televisores, carteles luminosos comerciales…) y utilizarse para transmitir mensajes a unidades móviles (telefonía, vehículos…).

En el mundo actualmente, gran parte de la energía eléctrica producida se emplea en la iluminación. Se puede asociar el fuerte desarrollo del LED a la necesidad por reducir este consumo, y hoy en día cada vez se utiliza más en semáforos, alumbrado público, carteles publicitarios, luces de vehículos… Gracias en gran parte a la irrupción del LED, las investigaciones con VLC también han avanzado en los últimos años. Conseguir que la lámpara ilumine y a su vez transmita información, supone un gran avance en cuanto a eficiencia energética y está llamado a revolucionar el concepto de la utilización de lámparas o bombillas LED. En definitiva, con la introducción del LED, las velocidades de transmisión se han incrementado y, dada su polivalencia, el interés en la tecnología VLC se ha intensificado en los últimos años. Tanto es así, que diferentes grupos de investigación han demostrado que es posible conseguir velocidades de transmisión de hasta Gigabits por segundo con la tecnología LED.

1.1. Objetivos

El presente trabajo consiste en el desarrollo de un sistema de transmisión que utiliza como medio de transmisión la luz visible, caracterizado por emplear una modulación VOOK (Variable On-Off Keying), y las actuales lámparas LED de iluminación comerciales como dispositivo emisor. En la Figura 1.1 se puede observar el diagrama de bloques del sistema de

comunicación desarrollado, en el que se pueden distinguir, como en cualquier sistema de comunicación, el transmisor, el canal o medio de transmisión y el receptor.

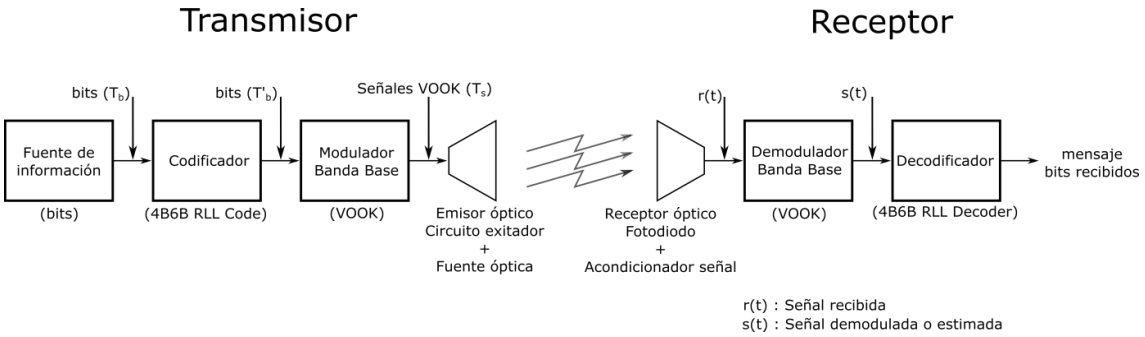


Figura 1.1. Diagrama de bloques del sistema de comunicación desarrollado

En el sistema desarrollado, el transmisor está compuesto por un codificador 4B6B, el modulador en banda base y el emisor óptico, compuesto por el circuito excitador y la fuente óptica. La función del codificador 4B6B es la de realizar una aleatorización de los datos suministrados por la fuente binaria con la finalidad de evitar la aparición de largas cadenas de un mismo símbolo binario. La función del modulador en banda base reside en transformar los bits que se desean transmitir en señales en tiempo continuo con el formato adecuado para ser transmitidas por el canal de comunicaciones, en este caso siguiendo un esquema de modulación VOOK. Por último, la del emisor óptico, basado en la utilización de una lámpara LED como fuente óptica, es la de llevar a cabo la conversión electro-óptica necesaria para adaptar la señal a transmitir a las características del canal visible. En lo que se refiere al receptor, éste está formado por el receptor óptico, que incluye el sensor de radiación visible y una etapa de amplificación/acondicionamiento de señal, el demodulador en banda base y el decodificador 4B6B. Como sensor de radiación, encargado de llevar a cabo la conversión óptico-eléctrica, se han empleado dos fotodiodos S7510 del fabricante Hamamatsu. La función del demodulador en banda base es realizar el proceso inverso del modulador, es decir, a partir de la señal VOOK resultante de la conversión óptico-eléctrica realizada por el receptor óptico, y en base a un criterio de decisión, estimar los bits o secuencia de bits que en formato 4B6B han sido recibidos desde el sistema transmisor. Por último, el decodificador 4B6B tiene como misión recuperar los bits de información transmitidos a partir de la secuencia de datos binarios estimados por el demodulador.

Para implementar el demodulador se ha recurrido a una estructura basada en la utilización de un correlador y decisor, que utiliza como criterio de decisión el de mínima distancia euclídea entre la señal recibida y las posibles recibidas desde el transmisor.

Por último, hay que indicar que, para implementar el codificador y decodificador 4B6B, así como el modulador y el demodulador VOOK, se ha hecho uso de una placa de desarrollo FPGA Nexys A7 con núcleo Artix-7 de Digilent.

1.2. Estructura de la memoria

En los siguientes capítulos se desarrolla el contenido del trabajo, iniciando con conceptos teóricos para pasar al desarrollo del sistema, exponiendo las dificultades encontradas y las decisiones adoptadas. Este documento se divide en 6 capítulos, nombrados a continuación:

- Capítulo 1. Introducción. Donde se exponen los objetivos de este Trabajo de Fin de Máster.
- Capítulo 2. Conceptos teóricos. Se exponen varios aspectos a tener en cuenta previamente al desarrollo del trabajo.
- Capítulo 3. Desarrollo del sistema. En él se describe, primero, de manera general el sistema y luego, detallando cada uno de los elementos involucrados.
- Capítulo 4. Simulaciones y resultados obtenidos. En este capítulo se muestran los resultados de las simulaciones realizadas, así como los resultados obtenidos del sistema completo.
- Capítulo 5. Presupuesto. Se muestra la valoración general del desarrollo del trabajo.
- Capítulo 6. Anexos. Donde se adjuntan documentos relacionados con los componentes utilizados (hojas de características), planos del diseño en PCB del circuito y el código implementado con comentarios.

Capítulo 2. Conceptos teóricos

Índice Conceptos teóricos

Para poder profundizar en el desarrollo del trabajo es necesario comentar algunos aspectos teóricos. En concreto, en este capítulo, se describen conceptos sobre los sistemas de comunicación por luz visible, la aleatorización de datos y la modulación VOOK.

2.1. Sistemas de comunicación mediante luz visible (VLC)

La Comunicación por Luz Visible o Visible Light Communication (VLC) es una tecnología que permite la transmisión de información en el espectro visible mediante la modulación de la intensidad de luz transmitida. Gracias al desarrollo del LED, con unas características adecuadas, favorecen que esta tecnología haya sido investigada en mayor profundidad en los últimos años. En VLC se transmite información desglosada en el envío de datos binarios ("unos" y "ceros"). Las características principales son las siguientes:

- Se encuentra en el espectro visible de la luz (375 – 780 nm). Ver Figura 2.1.
- Es inofensiva para el ojo humano.
- Se puede utilizar en áreas donde la radiofrecuencia esté regulada, o en espacios restringidos (aviones, hospitales…)
- Es bastante segura, la transmisión de datos está limitada al recinto donde se emite la luz en interiores y en la dirección de la emisión en exteriores (además de su corto alcance).

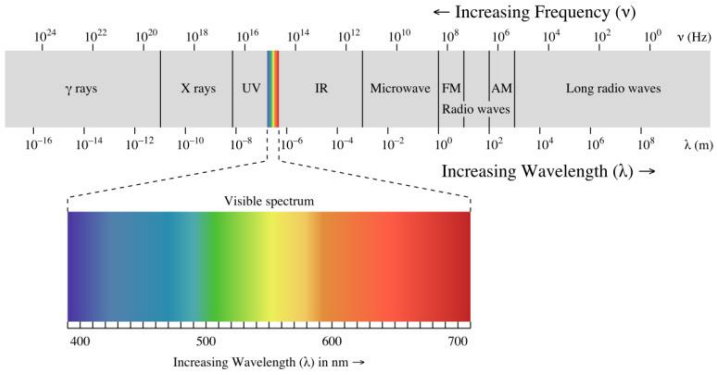


Figura 2.1. Espectro visible de longitud de onda

Por otro lado, otro de los objetivos del VLC, independiente de la transmisión de información, es que este mismo proceso no afecte a las personas. Los principales problemas en este sentido que se pueden encontrar son: el parpadeo (*flicker*) y el control de la atenuación o regulación de la intensidad de iluminación de la lámpara (*dimming*). Las altas frecuencias utilizadas durante la transmisión permiten que el usuario no perciba el parpadeo del LED y, por lo tanto, tampoco terminan influyendo en la principal función, la de iluminación, que debe cumplir lámpara que

se utilice como fuente óptica de emisión. Por otro lado, un aspecto también importante, es que el mensaje se pueda transmitir independientemente de la intensidad lumínica que prefiera establecer el usuario.

Al tratarse de un sistema que utiliza la luz para transmitir información, la velocidad puede llegar a ser bastante alta, apoyada por tecnología LED. Las velocidades alcanzadas serían suficientes para transmitir datos multimedia (audio y vídeo) Sin embargo, existen otras limitaciones que atenúan esa velocidad, como puede ser la electrónica del receptor, el medio y la distancia. Por ello, en este trabajo, se utilizan tasas de símbolos del orden de 100 kbs, ya que se ha comprobado que la bombilla utilizada y el receptor diseñado no soportan bien frecuencias por encima de 200 kHz.

Por otro lado, a la hora de diseñar un sistema de transmisión de información basado en VLC, se puede optar por realizar una transmisión basada en un enlace con línea de visión directa (*Line Of Sight* - LOS) o en un enlace en difusión. En este trabajo se ha optado por implementar un sistema VLC del tipo LOS, es decir, con visión directa entre el emisor y el receptor.

2.1.1. Ventajas y desventajas de los sistemas VLC.

Los sistemas VLC se pueden utilizar en entornos donde existen otro tipo de redes de transmisión, ya que no interfiere con los sistemas de radiofrecuencia y no se ve afectado por interferencias electromagnéticas. También se puede usar en entornos donde la radiofrecuencia estuviera limitada o fuera, en cierto modo, nociva (ejemplo: en hospitales). Es importante denotar como ventaja que la luz no es nociva para el ser humano. Otras redes inalámbricas como el WiFi, pueden ser sustituidas por VLC, ya que serían capaces de abarcar toda la superficie de la vivienda u oficina (con las múltiples lámparas). Un inconveniente que se presentaría sería el solapamiento entre señales de distintas lámparas que se podría corregir, por ejemplo, actuando sobre el receptor.

Si se realiza una comparación entre VLC y los sistemas por radiofrecuencia, se pueden encontrar varias ventajas en la transmisión de información mediante luz visible. A lo largo del tiempo, el espectro de radiofrecuencia ha sido utilizado ampliamente en numerosos sectores para las comunicaciones inalámbricas. Sin embargo, debido precisamente al crecimiento exponencial en el uso de redes inalámbricas, se ha alcanzado un espectro de radiofrecuencia cada vez más saturado y con mayor dificultad para el acceso. Esta limitación hay que tenerla en cuenta y, pese a los intentos para reducir la congestión, es necesario buscar alternativas. Es entonces cuando empieza a resultar atractiva la idea de usar luz visible para las comunicaciones.

El espectro de la luz visible (400 THz – 800 THz) es mucho más amplio que el espectro de radiofrecuencia (3 kHz – 300 GHz), lo cual permite un mayor margen para las comunicaciones inalámbricas. Es importante tener en cuenta la eficiencia energética que se deriva del uso de sistemas VLC basados en la utilización de las lámparas LED existentes, cada vez más implantadas. Por otro lado, como se utiliza luz para transmitir la información, ésta normalmente no atraviesa objetos y se puede redirigir fácilmente para evitar interferencias con otras transmisiones. Esto permite la coexistencia de varias transmisiones independientes en un área reducida, permitiendo mayor densidad de transmisión de información. Como conclusión, la idea no es sustituir todas las redes inalámbricas basadas en tecnología de radiofrecuencia, sino apoyar las mismas con los sistemas VLC para una mayor versatilidad.

Complementando lo expuesto anteriormente, el número de dispositivos conectados a las redes está en continuo crecimiento (lavadores, neveras, Smart-tv, horno, robots de limpieza…), lo que provoca que las redes se saturen en mayor medida y pierdan velocidad. Teniendo en cuenta esto, es necesario cubrir esta demanda y, dado que muchos de estos dispositivos (electrodomésticos) están situados en un emplazamiento fijo, el punto de conexión podría ser único. Una conexión individual mediante VLC sería una buena solución. Es de destacar también que cada vez se va reduciendo más el espectro de radiofrecuencia disponible, como se comentó anteriormente, y con este tipo de conexiones se puede reducir el uso de la radiofrecuencia (puede funcionar como alternativa).

Por otro lado, al utilizar tanto LED como fotodetectores se obtiene un sistema de comunicación con un consumo bajo, lo cual produce un impacto menos negativo para el medioambiente. Además, la luz no produce interferencias electromagnéticas (EMI). Estas ventajas, entre otras (ejemplo: reducción de costes), está haciendo que el LED esté cada vez más implantado en la vida diaria en todos los ámbitos, por lo que si se pretende implantar un sistema VLC ya se cuenta con parte de la infraestructura necesaria instalada (transmisor) y originaría menores costes.

En cuanto a seguridad, trabajar en entornos cerrados con VLC permite hacer una transmisión segura de la información, ya que no podrán acceder a la señal emitida desde fuera del recinto (no atravesará los cerramientos). Uno de los inconvenientes del VLC es que su rango de actuación es pequeño (unos cuantos metros), pero se puede ver como un punto a favor en la seguridad de la transmisión.

Como se ha mencionado en este punto, al tratarse de una transmisión por el aire, los sistemas VLC presentan algunas ventajas e inconvenientes, que se resumen en la Tabla 2.1.

Ventajas	Inconvenientes
Bajo consumo, eficiencia energética	Parpadeo
Altas frecuencias de transmisión (LED)	Control de la atenuación
LED inmune a interferencias electromagnéticas	Obstáculos en el medio
LED no interfiere con otros sistemas de radiofrecuencia	Ruido (armónicos, luz diurna)
Modulación del mensaje independiente de la intensidad lumínica	Rango de transmisión
Alta implantación del LED en todos los ámbitos	

Tabla 2.1. Resumen ventajas y desventajas de los sistemas de comunicación mediante luz visible.

Además, para disponer de conexión mediante VLC en los equipos actuales sería necesaria realizar una adaptación teniendo en cuenta los siguientes aspectos:

- Si se trata de un punto de conexión fijo (infraestructuras) o móvil (vehículos, móviles, tablets…)
- Si disponen de alimentación eléctrica suficiente, donde en infraestructuras no habrá problema y en dispositivos móviles está limitada por la batería.
- La potencia de la fuente de luz. Una lámpara de interior será más potente que el LED de un teléfono móvil.
- El rango de comunicación, generalmente corto o medio alcance.

Para finalizar, es necesario destacar que dos de los principales inconvenientes que se pueden dar en una transmisión VLC, el parpadeo y el control de la intensidad de iluminación de la lámpara, han sido objeto de estudios a lo largo de los años. Estos dos aspectos se abordan, entre otros asuntos, en el estándar IEEE 802.15.7 que trata sobre las comunicaciones inalámbricas de corto alcance que utilizan VLC, y donde se recogen métodos para solventar estos inconvenientes y permitir un buen funcionamiento del sistema de transmisión.

2.1.2. Estándar IEEE 802.15.7

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE – *Institute of Electrical and Electronics Engineers*) introdujo el estándar IEEE 802.15.7, aprobado en 2011, una vez reconocido el potencial que podía tener la tecnología VLC. En este estándar se establece, entre otros aspectos, especificaciones de diferentes capas físicas (PHY) para establecer este tipo de sistemas, ver Tabla 2.2. El uso de una u otra de las diferentes capas físicas que establece el enlace depende de la velocidad de transmisión que se desea obtener.

Capa física	Velocidad de transmisión
PHY I	11,67 kb/s hasta 266,6 kb/s
PHY II	1,25 Mb/s hasta 96 Mb/s
PHY III	12 Mb/s hasta 96 Mb/s (Para múltiples fuentes de transmisión, usando *color shift keying* o CSK)

Tabla 2.2. Clasificación de sistemas dependiente de la velocidad de transmisión.

Respecto al parpadeo o *flikering*, que consiste en la fluctuación de la intensidad lumínica de la fuente, en el estándar se establece la existencia de un periodo máximo de parpadeo (MFTP – *Maximum flicker time period*) donde el cambio de intensidad ya deja de ser perceptible por el ojo humano. Por lo tanto, las variaciones de intensidad de la transmisión deben tener períodos más pequeños que el valor del MFTP (5 ms). El parpadeo se puede clasificar en dos tipos: "*Intraframe flicker*" e "*interframe flicker*".

- *Intraframe flicker* (parpadeo en el envío de datos). Es el parpadeo que se produce durante la transmisión de un paquete de datos debido al envío de varios "ceros" seguidos, modulación errónea… Esto se puede reducir con la utilización de codificación *run-length limited* (RLL). En el caso que atañe a este trabajo, se utiliza el código RLL 4B6B y una modulación VOOK, que, además, permite regular la intensidad lumínica de la lámpara LED de emisión.
- *Interframe flicker* (parpadeo entre envío de mensajes). Es el parpadeo que se puede producir entre el envío de diferentes paquetes de datos. Se puede solucionar con la inclusión de un patrón de símbolos preestablecido con el valor del brillo promedio igual al del paquete de datos.

Por otro lado, para regular la intensidad de iluminación de la lámpara (*dimming control*) hay que tener en cuenta varios aspectos. Uno de ellos es el control de la intensidad lumínica independientemente de si se está transmitiendo mensaje o no (una de las ventajas del sistema

es que se pueda controlar). Es necesario que las personas no vean un parpadeo cuando el sistema deja de transmitir o comienza a transmitir un mensaje. Y por otro lado evitar el oscurecimiento de la lámpara debido al envío de múltiples "ceros" seguidos. En definitiva, controlar la intensidad lumínica percibida por el usuario, acorde a los requerimientos de este. Para lograrlo se pueden utilizar varios métodos que se resumen a continuación:

- Añadir símbolos compensatorios en el mensaje. Hay modulaciones que envían siempre una intensidad promedio de símbolo constante, como puede ser OOK, sin posibilidad de variación (ejemplo: OOK con codificación Manchester). Dadas estas circunstancias, si se pretende regular la intensidad de la lámpara, es necesario insertar tiempos de compensación, ver Figura 2.2. En estos tiempos se envían uno o varios símbolos en el paquete de datos para ajustar la intensidad promedio de la lámpara a la intensidad deseada.

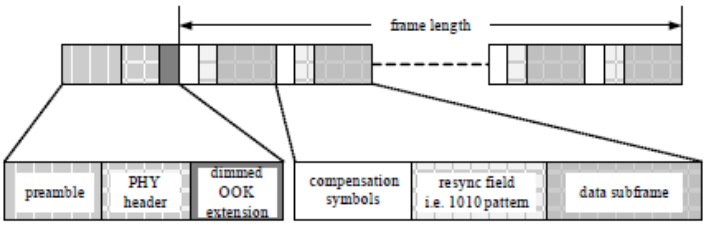


Figura 2.2. Estructura de mensaje de un sistema OOK.

A la vez que se realiza esta acción, se puede dividir el paquete de datos en varios más pequeños, a los que se les puede añadir un campo de resincronización de datos después de cada tiempo de compensación.

- Controlar el ancho del pulso de símbolo. El objetivo de este método es controlar el ancho de pulso de cada símbolo enviado, para aumentar o disminuir el ciclo de trabajo de la transmisión, ver Figura 2.3. Con esta regulación se puede adaptar el nivel de iluminación al deseado por el usuario. Los métodos de modulación VPPM (*Variable Pulse Position Modulation*) y VOOK (*Variable On-Off Keying*) se basan en este método. Al usar VOOK en este trabajo, el control de la intensidad se basa en el control del ancho de pulso del símbolo transmitido. Sólo se modifica el ancho del pulso, dejando la amplitud constante.

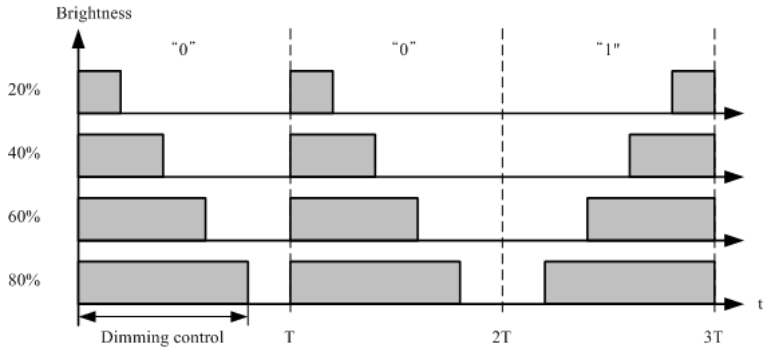


Figura 2.3. Ciclo de trabajo en función de la intensidad de la lámpara.

- Controlar la amplitud de la señal del mensaje. Se basa única y exclusivamente en modificar la amplitud de los pulsos para la transmisión de "ceros" y de "unos" en cada mensaje.

Como se ha mencionado anteriormente, el sistema de comunicación implementado en este trabajo se basa en la modulación VOOK, la cual permite modificar la intensidad lumínica de la lámpara independientemente del mensaje a transmitir. Por otro lado, se utiliza una codificación RLL 4B6B, en la que todos los mensajes codificados tienen igual número de "unos" que de "ceros" (probabilidad de bit del 50%). Con esta selección se consigue transmitir un mensaje codificado y modulado que permite variar la intensidad lumínica del transmisor.

2.1.3. Variable On-Off Keying (VOOK)

La modulación VOOK se basa en la modulación *On-Off Keying* (OOK) donde para controlar la intensidad lumínica de la lámpara se modifica el ciclo de trabajo de la señal OOK a transmitir. La principal característica de la modulación OOK es que al transmitir un "cero lógico" se envía el símbolo o la señal $p_0(t)$ y al transmitir un "uno lógico" se envía el símbolo o señal $p_1(t)$, ver Figura 2.4.

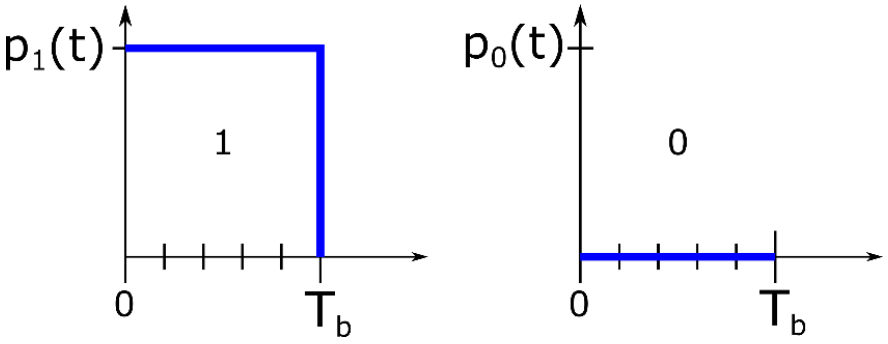


Figura 2.4. Símbolos en una modulación OOK.

Teniendo en cuenta lo anterior, este tipo de modulación no permite la regulación de la intensidad. Por ello, se necesita utilizar VOOK, que sí permite esto último. Es importante destacar, que este trabajo se realiza con una modulación VOOK donde los pulsos de las señales que se transmiten son del tipo NRZ (*Non-Return to Zero*). Para el correcto funcionamiento de esta modulación es necesario que la probabilidad a priori de bit sea del 50%, lo que se consigue a través de la codificación 4B6B. En la Tabla 2.3 se muestran los símbolos a transmitir en una modulación de tipo VOOK que permite regular la intensidad lumínica de la lámpara desde el 10% hasta el 90%, en incrementos del 10%. Una iluminación del 0% corresponde a la lámpara apagada (no se transmite información) y una iluminación del 100% equivale a utilizar la lámpara iluminando al máximo (no se transmite información). Esto se traduce en que la modulación VOOK no es capaz de transmitir datos con un ciclo de trabajo de $\delta = 0$ *o* $\delta = 1$.

δ	δ_b	Símbolo
1.0	0.0	1111111111
0.9	0.2	dd11111111
0.8	0.4	dddd111111
0.7	0.6	dddddd1111
0.6	0.8	dddddddd11
0.5	1.0	dddddddddd
0.4	0.8	dddddddd00
0.3	0.6	dddddd0000
0.2	0.4	dddd000000
0.1	0.2	dd00000000
0.0	0.0	0000000000

Tabla 2.3. Símbolos para transmitir en una modulación VOOK

La intensidad lumínica (γ_{vook}) de la lámpara coincide con el ciclo de trabajo de la señal a transmitir (δ), el cual está relacionado bidireccionalmente con la influencia de los datos (d) en cada símbolo o ciclo de trabajo del símbolo(δ_b).

$$\gamma_{vook} = \begin{cases} \frac{1}{2}\delta_b & 0 < \gamma \leq 0{,}5 \\ 1 - \frac{1}{2}\delta_b & 0{,}5 \leq \gamma < 1 \end{cases}$$

A continuación, ver Figura 2.5, se puede ver un ejemplo de modulación VOOK para diferentes porcentajes.

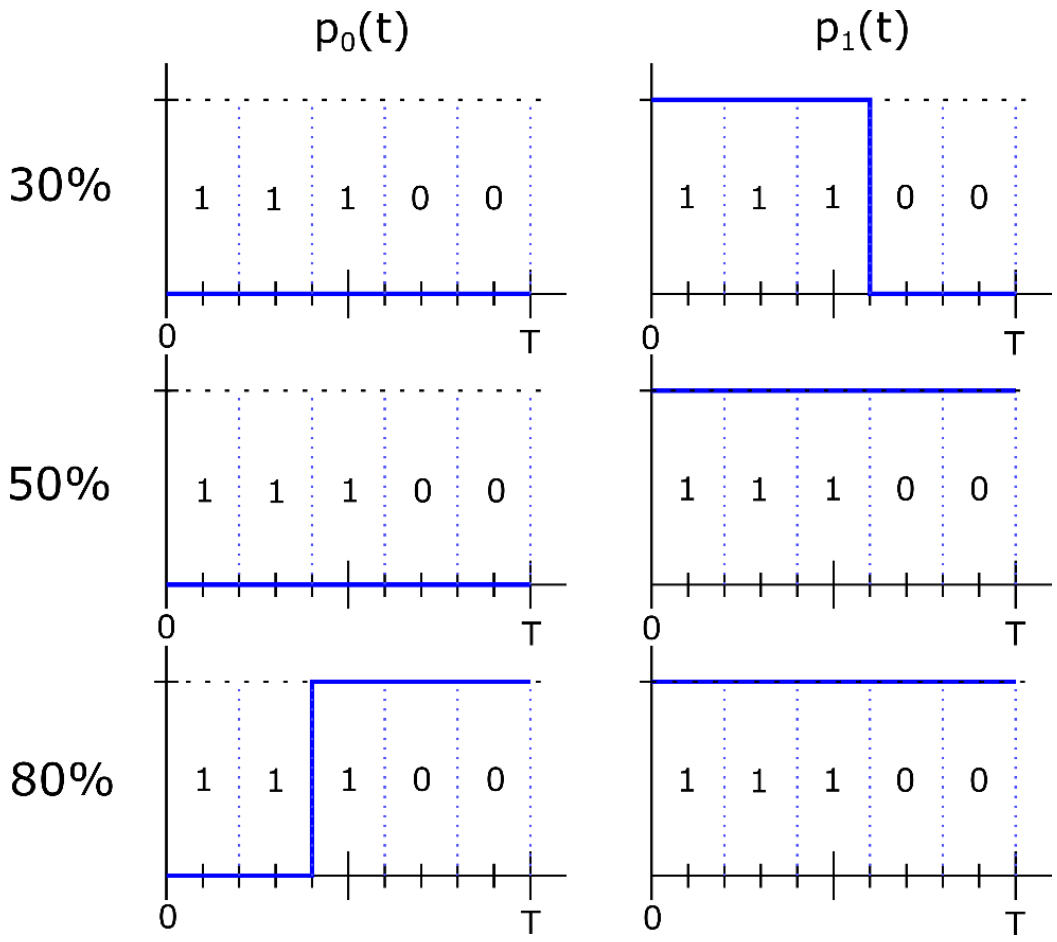


Figura 2.5. Símbolos a transmitir dependientes de la intensidad de la lámpara

2.1.4. Codificación 4B6B

Para evitar el parpadeo y utilizar el sistema VOOK para la regulación de la intensidad de la lámpara, una de las principales premisas es que la probabilidad de bit ("ceros" y "unos") sea $P_0 = P_1 = 1/2$, lo cual también evita cadena "ceros" o "unos" seguidos. Pues bien, esto se va a conseguir mediante una codificación RLL 4B6B del mensaje a transmitir. Los códigos RLL se usan para, precisamente, evitar largas cadenas de "ceros" o "unos" que pueden causar parpadeo, entre otros problemas de lectura. Por ello, la codificación 4B6B garantiza ese balance de "ceros" y "unos" mencionado anteriormente para cada símbolo.

La codificación 4B6B consiste en introducir redundancia, ampliando el número de bits del símbolo de 4 bits a 6 bits. Con esto se garantiza que todos los símbolos tengan el mismo número de "ceros" que de "unos, lo que implica la ausencia de parpadeos. En la Tabla 2.4 se muestra la asignación de cuatro bits a seis bits propuesta por el código RLL 4B6B.

4b	6b	Símbolo
0000	001110	0
0001	001101	1
0010	010011	2
0011	010110	3
0100	010101	4
0101	100011	5
0110	100110	6
0111	100101	7
1000	011001	8
1001	011010	9
1010	011100	A
1011	110001	B
1100	110010	C
1101	101001	D
1110	101010	E
1111	101100	F

Tabla 2.4. Valores correspondientes a la codificación 4B6B

Algunas ventajas de la utilización del 4B6B se listan a continuación:

- Evita el parpadeo de la lámpara LED durante el envío de datos.
- Permite la detección de si se ha producido un error en el símbolo.
- Permite regular fácilmente la intensidad lumínica mediante la modulación VOOK.
- Código sencillo por cada símbolo.

2.1.5. Limitaciones e interferencias en la señal VLC

Si bien se ha mencionado que las interferencias electromagnéticas no van a afectar a las señales de los sistemas VLC, existen otro tipo de interferencias o limitaciones que afectarían a la transmisión. Hay que tener en cuenta que las investigaciones y el desarrollo de la tecnología VLC es relativamente joven comparada con la radiofrecuencia.

En primer lugar, la interferencia entre lámparas contiguas puede provocar errores en la lectura al mezclar señales de dos fuentes diferentes. Se debería clasificar sobre qué tipo de dispositivo se trabaja (móvil o fijo) para determinar posibles soluciones. Desde un elemento físico a modo de barrera, hasta filtros en el receptor, serían eficaces para evitar este tipo de interferencias. También se puede implementar en el receptor la electrónica necesaria para discernir entre señales en base a la intensidad lumínica que alcanza el dispositivo.

Otro elemento importante es la interferencia debida a la luz ambiente, independientemente de que estos sistemas se localicen en el interior o exterior. La luz diurna generaría un valor continuo en el receptor, sobre el que habría que aplicar un filtro para eliminar esta componente. Por otro lado, a la hora utilizar un sistema VLC, es importante tener en cuenta también que otro tipo de lámparas (fluorescentes, incandescentes…) van a introducir ruido en forma de armónicos en la señal que llega al receptor, pudiendo provocar errores en la transmisión. Un método de codificación que resulta robusto para evitar este tipo de interferencias es la codificación Manchester (recomendada para sistemas OOK), cuando se tienen frecuencias por debajo de los MHz.

En sistemas VLC para interiores, normalmente se construye la estructura para tener una comunicación basada en un enlace directo entre el emisor y receptor. En ausencia de obstáculos, este tipo de enlaces es mucho más eficiente en potencia que un enlace no directo o en difusión. Pero si, por el contrario, el canal no es directo y el nivel de iluminación que llega al detector es bajo, limitando consecuentemente la velocidad de transmisión.

Si se pretende utilizar dispositivos móviles, es importante tener en cuenta que en algún movimiento (rotación) se puede perder la señal y cortar la transmisión de información. Por lo tanto, es importante que se implemente un elemento que recupere el enlace, o algún mecanismo que evite la pérdida de la señal debido al movimiento. Este inconveniente es uno de los más importantes para poder aplicar la tecnología a dispositivos móviles y debe ser donde se centren gran parte de los esfuerzos de investigación.

Para finalizar, todas estas limitaciones se encuentran en investigación y se han alcanzado ciertos avances. Muchas de las interferencias en la señal se pueden mitigar mediante el filtrado óptico. Pero también el uso de filtros analógicos o digitales después del fotodetector ayudarían a mejorar la transmisión.

2.2. Aplicación de la tecnología VLC

Principalmente, los entornos ideales para implementar sistemas VLC, son entornos donde las lámparas siempre vayan a estar encendidas (industria, transporte de pasajeros, hospitales, administración pública…). Sin embargo, se podría acoplar un emisor infrarrojo que complemente al sistema VLC y permita la transmisión cuando las lámparas se encuentren apagadas. Con esto último, el rango de aplicación se podría ampliar a todos los hogares y lugares donde no siempre estarían las lámparas encendidas.

Una de las principales ventajas de los sistemas VLC, es que no interfieren con las señales de radiofrecuencia y no generan tampoco interferencias electromagnéticas, la información se transmite a través de impulsos en la señal lumínica. Teniendo en cuenta esto, lugares sensibles en cuanto a interferencias (cabinas de aviones, industrias, hospitales…) podrían hacer uso de esta tecnología. En esta línea, es importante destacar que, en algunas zonas de los hospitales, la radiofrecuencia no es bien recibida como cerca de escáneres, o zonas quirúrgicas. VLC es una tecnología cuyo espectro no interfiere para nada con el espectro de radiofrecuencia y eso la hace una tecnología interesante de aplicación en dichos entornos.

Además de las zonas mencionadas en el párrafo anterior, dentro de hospitales existen más emplazamientos donde se podría utilizar la luz visible como medio de transmisión. Existen muchos pacientes conectados a monitores de signos vitales, desde UVI, URPA, hasta en planta, y muchos de estos monitores se conectan actualmente a una estación central de monitorización (bien mediante cable de Ethernet, o bien mediante WiFi). Una alternativa es proponer el envío de constantes vitales mediante VLC y comprobar su viabilidad. Por otro lado, también existen ubicaciones que presentan características para el aislamiento de pacientes que así lo necesiten. En estos casos, el protocolo para la interacción con el paciente es bastante restrictivo, y la transmisión de la información del paciente mediante VLC facilitaría el trabajo de los enfermeros y médicos.

A la hora de viajar en avión, siempre emiten el mensaje de apagar o poner los dispositivos electrónicos en modo avión. Esto es para reducir el riesgo de interferencias, pero si se utiliza la luz para transmitir información tanto en cabina, como a los pasajeros para que puedan acceder a los sistemas de comunicación, sería una solución interesante. El inconveniente es el coste que introduciría en la fabricación de aeroplanos o en su adaptación.

Existen emplazamientos donde, dada su intrincada arquitectura, es fácil desorientarse o no es posible acceder a un mapa del recinto. Pues bien, alguna de las aplicaciones VLC que han ido surgiendo van relacionadas con este tipo de edificaciones (hospitales, museos, aeropuertos, estaciones de metro/tren, centros comerciales…) y la localización en el interior. Como se ha mencionado anteriormente, una característica de este tipo de edificios es que las luces normalmente están encendidas. Una mayor o menor precisión se consigue mediante el uso de múltiples lámparas transmitiendo balizas individuales a teléfonos móviles, carteles luminosos, etc. Haciendo inciso en esto, la multinacional Phillips, junto a la cadena de supermercados Carrefour, ha llevado a cabo un proyecto de iluminación en una gran superficie donde también

han implementado este sistema de localización basado en el uso de la tecnología VLC. De este sistema dicen que actúa como un GPS de interior, transmitiendo información sobre la localización a una aplicación en el teléfono móvil. La propia empresa indica que es un sistema fácil de escalar y con una precisión bastante alta (margen de error de menos de 1 metro). Además, le ha supuesto a Carrefour un ahorro del casi 50% en iluminación. La compañía de supermercados prevé un aumento de las ventas y una mayor satisfacción de los consumidores gracias a este sistema de localización.

El uso de la luz para transmisiones bajo el agua resulta más eficiente que utilizar radiofrecuencia, las ondas de radio se atenúan mucho más rápido que la luz. Por lo que el uso de VLC aumenta el rendimiento de transmisiones inalámbricas submarinas, colocándola como una alternativa interesante para la comunicación en este medio. Varios factores entran en juego: la limpieza del agua, densidad… Pero, en definitiva, ofrecería un mayor rendimiento que las ondas de radiofrecuencia.

En cuanto a la utilización de VLC en exteriores, se podría utilizar en vehículos para transmitir información del entorno o mediante el alumbrado público a transeúntes que lleven consigo un dispositivo móvil. La comunicación entre vehículos y su entorno está enmarcada como un aspecto clave en los sistemas de seguridad de estos, y la velocidad de transmisión necesaria no sería muy alta. La comunicación no tendría que estar dirigida a un receptor en particular, y sería unidireccional a modo de información, sin necesidad de respuesta. Esta información sería de gran ayuda en el desarrollo de vehículos autónomos, ya que tendrían varias fuentes de información (sus sensores, el entorno, otros vehículos…). Desde el punto de vista de los elementos implicados en la comunicación, se podría clasificar la transmisión en: comunicación entre vehículos o comunicación vehículo-entorno.

- La comunicación entre vehículos. Muchos accidentes se basan en las distancias de seguridad y la lenta reacción de las personas. Con una comunicación mediante las luces diurnas, las luces de freno o intermitentes, se puede transmitir a los vehículos adyacentes mucha información: velocidad, aceleración, giro, etc.
- La comunicación entorno-vehículo. En entornos de tráfico congestionado, ciudad y sus alrededores, existen muchos indicadores luminosos a los que hay que atender durante la conducción, y sería atractivo que toda la información que percibe el ojo humano se transmita también a los vehículos. Empezando por los semáforos, los cuales aportarían a los vehículos la información que transmiten a los conductores. O también los vehículos

de servicios públicos (ambulancias, bomberos, policía…) que utilizan señales luminosas y a la vez acústicas para hacer notar a los conductores su urgencia. Por último, la transmisión de información sobre el tráfico en los alrededores a través del alumbrado público.

Otra posible aplicación es la transmisión de publicidad en difusión, carteles luminosos que transmitan además cierta información no presente en el mismo para acceder a una página web, en cierto modo como los códigos QR en la actualidad.

Existe gran variedad de dispositivos susceptibles de poder usar tecnología VLC: teléfonos inteligentes, PDA, señales luminosas, carteles publicitarios, luces de tráfico, alumbrado público, focos de vehículos… Para ello es necesaria la adaptación de los dispositivos, a nivel de hardware y de software.

Capítulo 3. Desarrollo del sistema

Índice Desarrollo del sistema

El objetivo de este trabajo es conseguir un sistema capaz de transmitir información mediante luz visible. Para ello se han construido un transmisor para generar y enviar un mensaje y un receptor capaz de tratar la señal recibida para recuperar el mensaje original. En este capítulo se describen tanto hardware como software implementado.

3.1. Descripción general del sistema.

Como ya se ha mencionado anteriormente, la finalidad del sistema desarrollado es la transmisión de un mensaje, una palabra de cuatro bits, usando la comunicación mediante luz visible. Para ello, como en todo sistema de comunicaciones, se necesitan dos elementos principales, un transmisor y un receptor.

El transmisor es el encargado de codificar, modular y enviar la señal. En este caso ya se dispone de una lámpara LED, ver Figura 3.1, para enviar el mensaje a través de una señal lumínica. Esta lámpara LED consta de 24 diodos emisores y se alimenta a 5 V. Para la conmutación de encendido a apagado, dispone de un driver que habilita esta conmutación dependiendo de la entrada de datos. Es importante destacar que la intensidad de la lámpara LED es inversa a la entrada de datos, por ello la entrada de datos se va a invertir antes de enviarla a la lámpara.

Figura 3.1. Lámpara LED utilizada

El mensaje que se desea enviar pasa a través de una etapa de codificación (4B6B) y luego por otra de modulación (VOOK). Sin embargo, la señal que le llegará a la lámpara no vendrá directa de la FPGA, sino que tendrá que pasar por una etapa de adecuación de la señal. Esta adecuación consiste en elevar el valor de la señal 0-3,3 V a 0-5 V. El circuito utilizado se basa en un buffer y una resistencia pull-up.

La lámpara se ha colocado a varias distancias de los fotodetectores (desde 20 hasta 80 centímetros), aunque finalmente se ha trabajado en torno a los 60 cm donde se obtenía una buena relación de distancia y fiabilidad de la señal.

Por último, el receptor será el encargado de recibir la señal transmitida para adecuarla al sistema, demodularla y descodificarla. Este elemento se compone principalmente de un receptor óptico (2 fotodetectores Hamamatsu S3071), acondicionamiento de la señal y la decodificación y demodulación del mensaje. Estos dos elementos principales, la lámpara LED y el circuito receptor con fotodiodos, forman el enlace óptico utilizado para el desarrollo del sistema.

El software desarrollado para el transmisor y el receptor se ha implementado en un kit de desarrollo FPGA Nexys A7 cuyo chip es el Artix-7.

3.2. Descripción de la electrónica.

Tanto el transmisor como el receptor han necesitado de la implementación de una parte electrónica para el correcto funcionamiento del sistema. El transmisor ha necesitado de un pequeño circuito elevador de tensión, para pasar los valores de salida de la FPGA (0-3,3 V) a valores requeridos por la lámpara LED (0-5 V). Por otro lado, el receptor ha necesitado la implementación de una electrónica que permita el acondicionamiento de la señal recibida por los fotorreceptores. El elemento principal del receptor es un amplificador transimpedancia, también conocido como un convertidor de corriente a voltaje. La implementación de la electrónica del receptor, ver Figura 3.2, es indispensable para que desarrollo del sistema VLC.

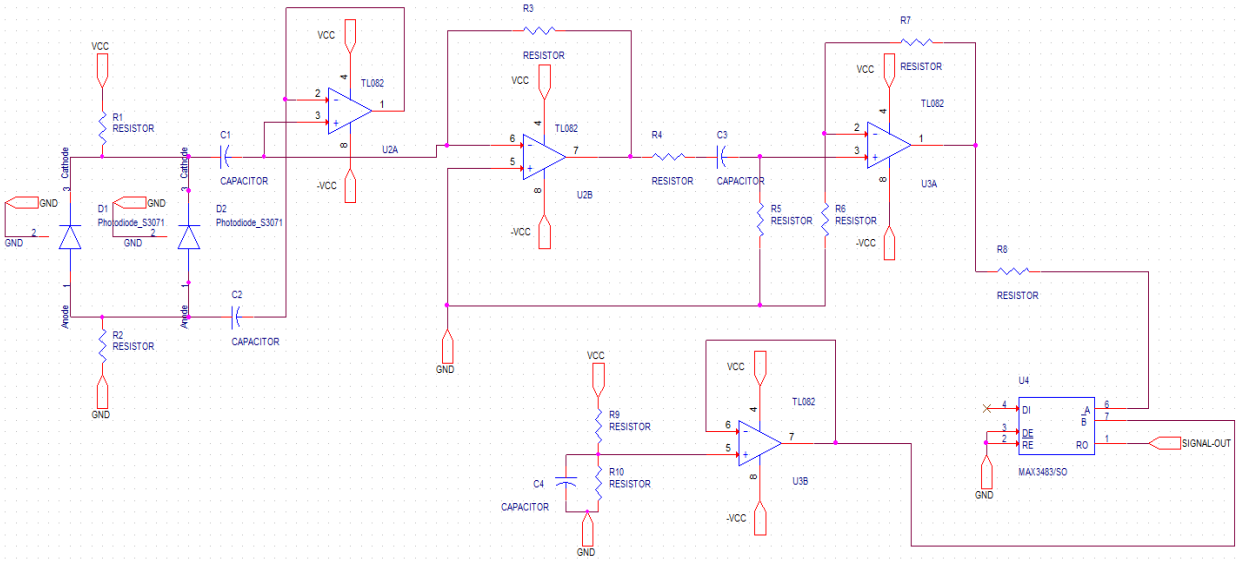


Figura 3.2. Circuito electrónico receptor de la señal lumínica.

3.2.1. Componentes utilizados

Es necesario introducir primero los componentes utilizados para el desarrollo y pruebas del receptor.

- Como fotodiodo se ha utilizado el modelo S3071 de la marca Hamamatsu.
- El circuito operacional utilizado ha sido el TL082.
- Un circuito integrado 74LS07 para la adaptación de los niveles de tensión entre FPGA y lámpara LED.
- Un circuito integrado MAX3485 para la comunicación con la FPGA.
- Componentes pasivos como resistencias y condensadores.

3.2.2. Descripción del esquema

En primer lugar, es necesario indicar que todos los circuitos integrados (CI) están alimentados a ±5 V, salvo el MAX3485 que se alimenta a 3,3V-0V. Para la elevación de tensión desde la salida de la señal a transmitir desde la FPGA hasta los 5 V admitidos por la lámpara LED, se ha utilizado un CI 74LS07, un *buffer*, y una resistencia de 1 kΩ en configuración "*pull-up*" para elevar la tensión de salida, ver Figura 3.3. Esta configuración es inversora, ya que la radiación o intensidad de la lámpara está invertida con respecto a la entrada.

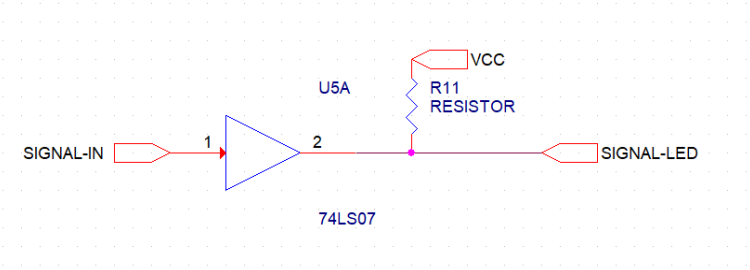


Figura 3.3. Circuito del transmisor para elevar voltaje de la señal

El receptor se compone de varios elementos para la adquisición de la señal lumínica y su conversión a valores interpretables por la FPGA. El elemento principal para la recepción de la señal lumínica es el fotodiodo. En este caso, el fotodiodo elegido es de la casa HAMAMATSU, modelo S3071, ver Figura 3.4. Lo más importante del fotodiodo va a ser su área activa (19.6 mm^2) y la corriente que es capaz de generar en función de la luz recibida. El fotodiodo viene acompañado de una lente y un filtro óptico, que ayuda a concentrar el haz en el área activa del fotodiodo.

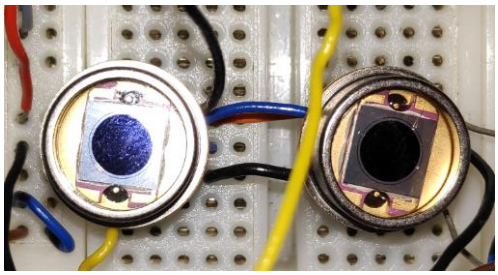


Figura 3.4. Fotorreceptores utilizados. Modelo S3071 del fabricante Hamamatsu.

Debido a que el área activa no es muy grande, se han utilizado dos fotodiodos en paralelo, como se ve en la figura anterior, para aumentar la misma hasta casi 40 mm² y conseguir así un mejor comportamiento del circuito. Gracias a un área activa mayor, permite alejar un poco más la bombilla LED frente al circuito con un solo fotodiodo.

Los fotodiodos trabajan típicamente polarizados inversamente para altas frecuencias y tener así tener un ancho de banda mayor, donde la fotocorriente generada es proporcional a la iluminancia. La respuesta es prácticamente lineal, por lo que resulta ideal para este tipo de aplicaciones. En este caso los fotodiodos en paralelo se conectan en un circuito de entrada para dar estabilidad a la señal, ver Figura 3.5.

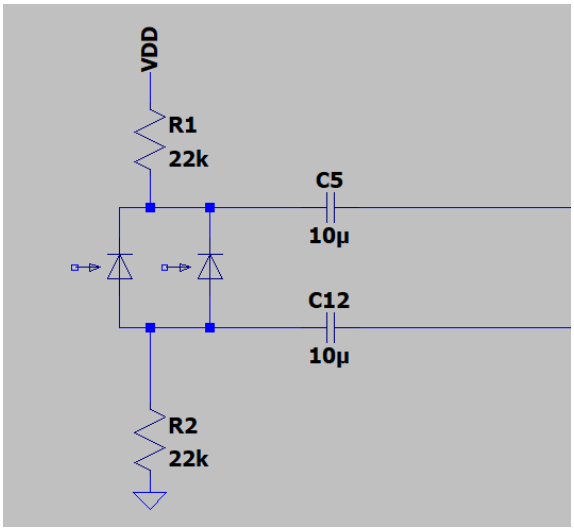


Figura 3.5. Circuito receptor de señal, con dos fotodiodos en paralelo.

A continuación del fotorreceptor, es necesario un circuito para convertir esa señal lumínica en una señal eléctrica que pueda ser interpretada y con la que se pueda trabajar para recuperar el mensaje recibido. Es aquí donde se introduce el amplificador transimpedancia. El principio fundamental de un amplificador transimpedancia es que convierte una corriente de entrada en una tensión proporcional, por eso también se puede llamar convertidor corriente-voltaje. Basado en un amplificador operacional (OpAmp), idealmente, la resistencia de entrada del

amplificador transimpedancia es cero y tiene baja resistencia de salida, ver Figura 3.6, por ello tiene muchas aplicaciones posibles en la electrónica. En este caso, se utiliza como parte de un circuito acondicionador de señal del receptor del sistema de transmisión con luz visible.

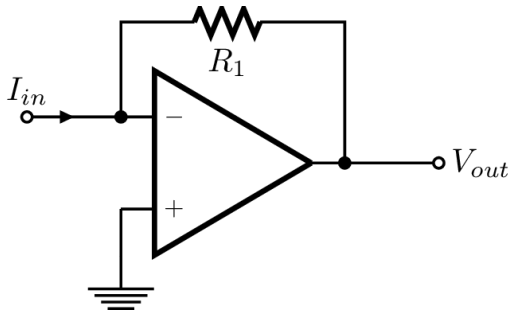


Figura 3.6. Circuito estándar de un amplificador transimpedancia

En primer lugar, se realizaron pruebas con operacionales LM308, pero dado que las frecuencias de trabajo no iban a ser muy altas, se optó por utilizar el operacional TL082. Este operacional tiene menor ancho de banda, pero ofrece altas velocidades y no necesitan mucha corriente de alimentación para trabajar bien. También presentan niveles de ruido bajos, ideal para la aplicación. Tanto este amplificador operacional, como el resto de los utilizados en el circuito, se alimenta a ±5 V directamente desde una fuente de alimentación del laboratorio de comunicaciones.

En una etapa previa al amplificador de transimpedancia, y para limitar los efectos del ruido, la resistencia equivalente del sensor tiene que ser mayor que la resistencia del amplificador. Por ello, se ha utilizado un amplificador operacional realimentado negativamente (buffer de ganancia 1) para generar alta impedancia en el circuito de entrada previo al amplificador de transimpedancia. Además, si la resistencia del amplificador es muy grande, se limita el ancho de banda. En las primeras pruebas se comenzó utilizando una de 300 kΩ, que posteriormente se fue reduciendo hasta el valor de 56 kΩ, ver Figura 3.7, para ganar ancho de banda.

Por otro lado, el circuito puede verse limitado por la capacidad parásita de la resistencia R_1 de la Figura 3.6, por ello a la salida del amplificador se incluye un condensador de mayor capacidad (100 nF) para anular esta limitación en frecuencia. Si se tratara de un amplificador ideal, se obtendría que la salida en tensión sería proporcional a corriente de entrada $(v = iR)$. Sin embargo, teniendo en cuenta que la ganancia varía con la frecuencia (disminuye al aumentar esta última), el amplificador operacional tiene a su vez limitaciones en este aspecto. Este mismo efecto, puede producir que el sistema se vuelva inestable.

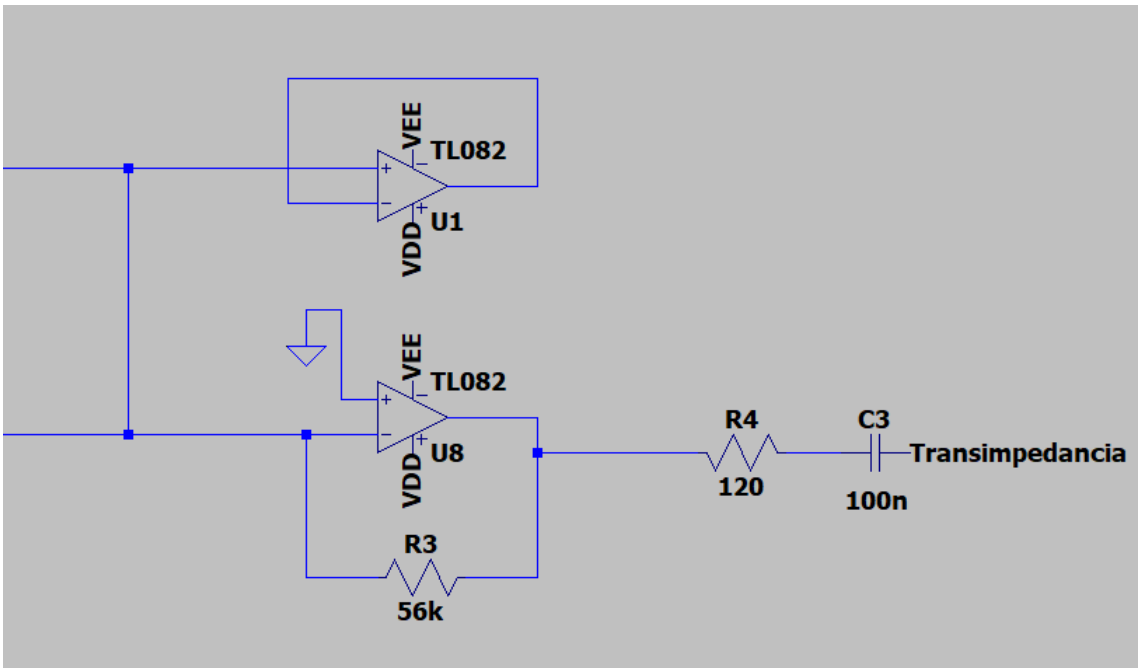


Figura 3.7. Amplificador transimpedancia implementado

La señal que ofrece la salida del amplificador transimpedancia no es suficiente para ser transmitida a la FPGA directamente si baja mucho la intensidad lumínica, por lo que se ha añadido una etapa amplificadora. En este caso se trata de un amplificador no inversor, ver Figura 3.8.

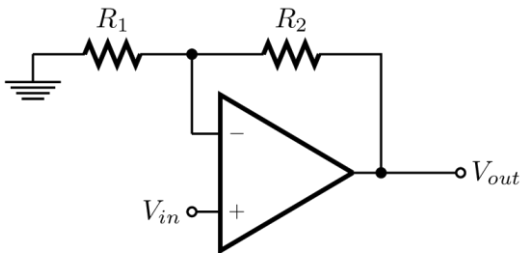


Figura 3.8. Amplificador no inversor

Teniendo en cuenta lo expuesto anteriormente sobre el amplificador transimpedancia, se ha elegido un amplificador no inversor, ver Figura 3.9, con ganancia suficiente para continuar la señal hacia el *transceiver* que va a emitir la señal a la FPGA, ver Ecuación 3.1.

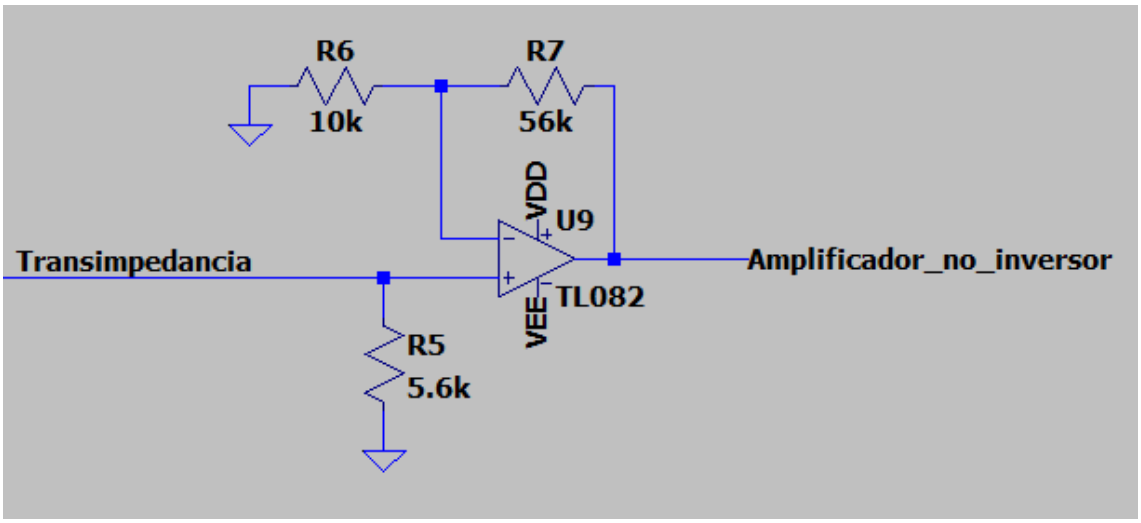


Figura 3.9. Segunda etapa amplificadora.

$$R_1 = 10\ k\Omega$$

$$R_2 = 56\ k\Omega$$

$$A_v = \frac{v_{out}}{v_{in}} = \frac{R_2}{R_1} + 1 = 6{,}6 \qquad (3.1)$$

Posteriormente a estas etapas de amplificación, se obtiene una señal que está entre -5 V y +5 V. Dado que la señal de entrada en la FPGA deberá estar entre 0-3.3 V, se ha utilizado un *transceiver* (MAX3485), ver Figura 3.10, que recoge la señal de salida de circuito acondicionador y la compara con un valor prefijado para generar la señal de salida adecuada hacia la FPGA. El uso del *transceiver* evita que el ruido del circuito afecte sobremanera a la señal que se le va a transmitir a la FPGA. Se han probado otros métodos como un buffer reductor, disparador de Schmitt básico… pero se ha optado finalmente por este circuito integrado dada la sencillez del circuito y la buena respuesta que se obtiene.

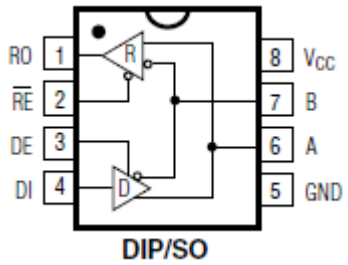


Figura 3.10. Circuito integrado MAX3485

Las características principales del circuito integrado MAX3485 son:

- El circuito está alimentado a 3.3 V (directamente desde la FPGA) y GND.
- Las entradas A y B para las señales de entrada.

- Los pines $\overline{RE}$ y DE irán directamente a tierra. $\overline{RE}$ para habilitar la salida RO. DE está en baja ya que no aplica su uso.
- El pin DI es para manejar salidas, permanece no conectado ya que no aplica su uso tampoco.
- El pin RO es el que tiene la señal de salida hacia la FPGA.

Este circuito se basa en la comparación de dos valores y si bien uno se encuentra por arriba o por debajo del otro. En este caso, la comparación de una señal (salida del circuito acondicionador) en la entrada A y a un valor de referencia conectado a la entrada B. Se ha utilizado un valor de tensión mayor que 0 V, en torno a 1 V, como referencia, mediante un divisor de tensión en el circuito. Esta decisión se ha adoptado pese a que la señal recibida tendrá una media de continua de valor 0 V (que se podría usar como umbral), pero este valor genera un mal comportamiento del integrado, el ruido en la señal de entrada puede provocar cambios no deseados a la salida del *transceiver*. El uso de este circuito integrado favorece al sistema en general. Se ha probado incluso primero con un MAX485 (salida a 5 V), pero dado que existe el MAX3485 que ofrece una salida de 3.3 V hacia la FPGA, ha resultado más atractivo usar este último. Para finalizar con este circuito, es importante destacar que con el MAX3485 se atenúa el pequeño ruido que pueda existir en la señal leída.

Para el valor de referencia del circuito MAX3485 se ha utilizado un divisor de tensión sencillo. Tras lo comentado en el párrafo anterior, la inestabilidad con un valor umbral de 0 V, se ha decidido colocar un divisor de tensión regulable con potenciómetro, ver Figura 3.11, trabajando con valores por encima de un voltio. El principio fundamental es que se pueda establecer el nivel umbral con este potenciómetro fácilmente, idealmente por encima de 0V.

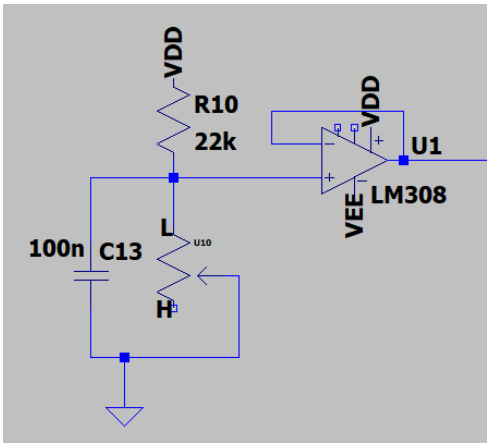


Figura 3.11. Circuito divisor de tensión

3.2.3. Diseño de placa de circuito impreso

El esquema anteriormente descrito ha sido trasladado al software OrCAD Lite 17.2 para la realización del diseño de una placa de circuito impreso (PCB). En la misma placa se han dispuesto el circuito acondicionador de la salida de la FPGA hacia la lámpara LED, así como la electrónica del receptor. Para la implementación del circuito se ha utilizado la herramienta *Capture* del software y para el diseño de la placa se ha utilizado la herramienta *PCB Editor*.

A la hora de implementar el circuito en la herramienta capture se ha tenido en cuenta la necesidad de incluir un conector para dar entrada/salida a todas las señales necesarias para el funcionamiento del receptor. El conector dispone en total de ocho pines, de los cuales se utilizan siete, ver Figura 3.12.

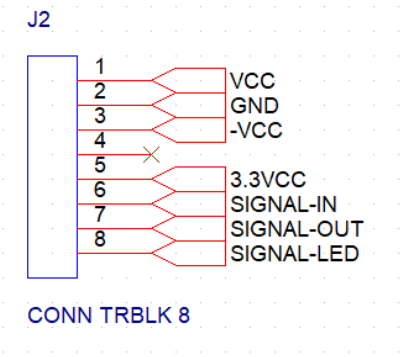


Figura 3.12. Conector de señales en el diseño PCB

Las distintas señales se describen en la siguiente Tabla 3.1:

Señal	Descripción
VCC	Entrada de alimentación +5V
GND	Entrada de alimentación 0V/Tierra
-VCC	Entrada de alimentación -5V
3,3VCC	Entrada de alimentación 3.3V
SIGNAL-IN	Señal para transmitir desde FPGA
SIGNAL-OUT	Señal adquirida
SIGNAL-LED	Señal para transmitir hacia la lámpara LED

Tabla 3.1. Descripción de las señales del conector de la placa diseñada.

Por último, para el diseño de la PCB se ha seleccionado una forma rectangular con unas medidas de 10x5 cm, incluyendo agujeros para tornillería de montaje, ver Figura 3.13. En los anexos se pueden encontrar unos planos con más detalles sobre el diseño de la PCB.

Figura 3.13. Dimensiones del diseño de la PCB del circuito.

3.3. Descripción del software

Como ya se ha mencionado, para el desarrollo del trabajo se ha utilizado un kit de desarrollo basado en FPGA (Nexys A7) cuyo chip es el Artix-7. Para el desarrollo del código se ha utilizado el lenguaje de programación VHDL, que es un acrónimo que proviene de la combinación de otros dos: VHSIC (*Very High Speed Integrated Circuit*) y HDL (*Hardware Description Language*). Además, es uno de los lenguajes que se suele usar para la programación de FPGA (*Field Programmable Gate Array*). El código se anexa al final de esta memoria para su lectura.

La idea principal de funcionamiento del sistema es que, a través de los interruptores que tiene la placa de desarrollo, se introduzcan dos comandos. El primero, con cuatro interruptores (uno por bit), el mensaje a transmitir y el segundo, con otros cuatro interruptores, el porcentaje de intensidad lumínica de la lámpara (10% hasta 90%, tramos de 10%). Estos parámetros servirán para generar una señal acorde a las indicaciones del usuario. Es necesario indicar que estos parámetros solo van a ser leídos por el transmisor, el módulo del receptor no va a tener información de ninguno de estos parámetros, ni el mensaje ni el porcentaje de intensidad lumínica que se ha indicado.

La señal de salida principal será el mensaje ya codificado y modulado. Sin embargo, se utilizarán además señales de control en pasos intermedios para comprobaciones. A la entrada, se encuentra la señal recibida y tratada por el receptor para la FPGA. También existirán señales adicionales entre la demodulación y la decodificación, y el resultado final (mensaje demodulado y decodificado) en la salida para comprobaciones.

En líneas generales, el código se ha dividido en tres grandes bloques: generador de relojes, transmisor y receptor, ver Figura 3.14. Estos tres bloques serán expuestos a lo largo de este punto. A continuación, se puede ver un esquema sencillo de los tres bloques y sus interconexiones.

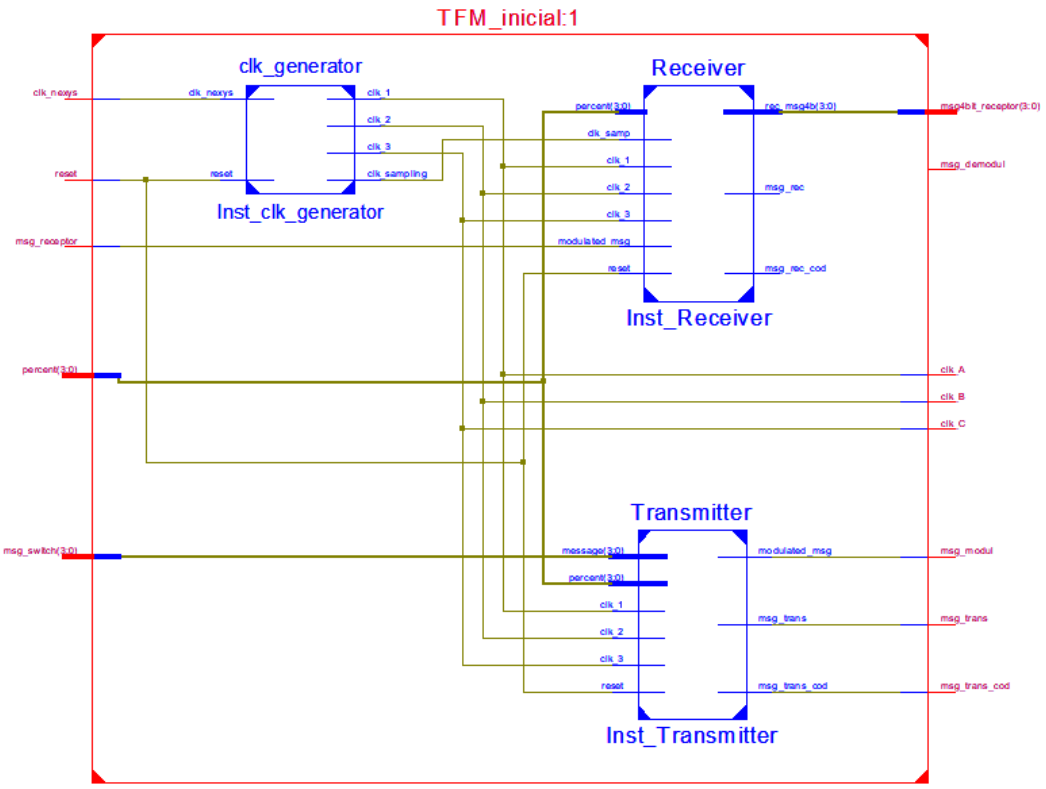


Figura 3.14. Esquema global del sistema

Tanto el desarrollo del sistema, como las simulaciones de este, se han llevado a cabo con una herramienta de Xilinx (ISE Design Suite 14.7). Desde este software se crea el archivo de programación necesario para la placa de desarrollo FPGA. En los siguientes puntos se mencionarán las características principales tanto del kit de desarrollo, como de los diferentes bloques que componen el sistema.

3.3.1. Características del kit de desarrollo Nexys A7

Como se ha mencionado anteriormente, se dispone de un kit de desarrollo en FPGA (Nexys A7) basado en el chip Artix-7. La placa Nexys A7 dispone de una FPGA de alta capacidad y numerosos puertos de entrada-salida. También dispone de un acelerómetro, un sensor de temperatura, salida VGA, Ethernet, micrófono digital… En definitiva, es una placa de desarrollo bastante completa. Se alimenta mediante USB o desde una fuente externa entre 4.5-

5,5 V. Del manual se ha obtenido la relación de I/O de la placa, ver Figura 3.15, y la descripción de estos, ver Tabla 3.2.

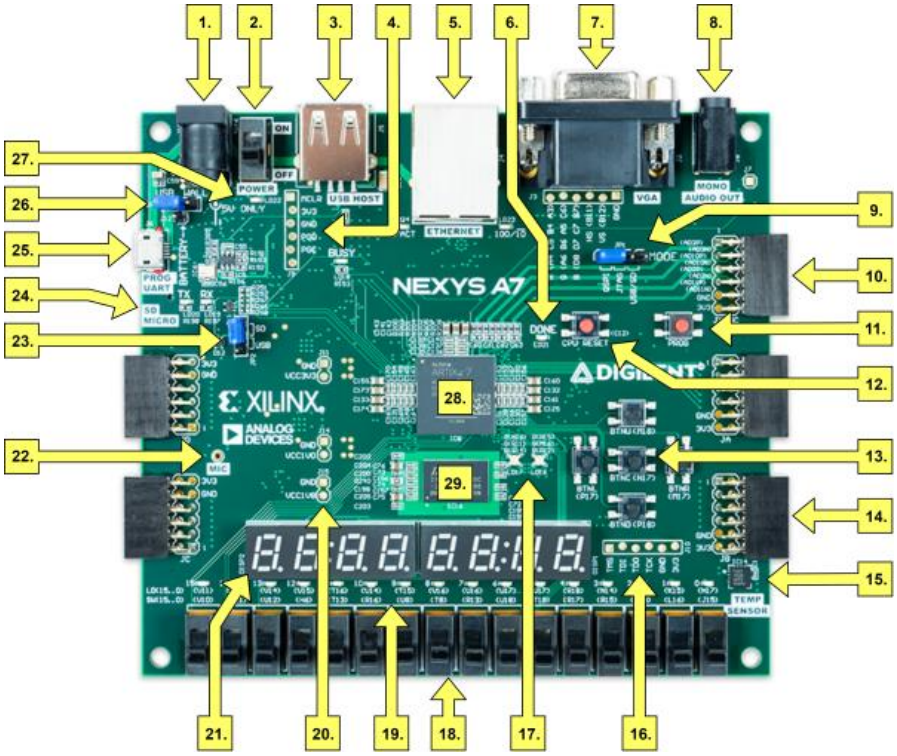


Figura 3.15. Placa de desarrollo Nexys A7

Pin	Denominación del puerto	Pin	Denominación del puerto
1	Powerjack	16	JTAG port for (optional) external cable
2	Power switch	17	Tri-color (RGB) LEDs
3	USB host connector	18	Slides witches (16)
4	PIC24 programming port (factory use)	19	LEDs (16)
5	Ethernet connector	20	Power supply test point(s)
6	FPGA programming done LED	21	Eight digit 7-seg display
7	VGA connector	22	Microphone
8	Audio connector	23	External configuration jumper (SD / USB)
9	Programming mode jumper	24	MicroSD card slot
10	Analog signal Pmod port (XADC)	25	Shared UART/ JTAG USB port
11	FPGA configuration reset button	26	Power select jumper and battery header
12	CPU reset button (for soft cores)	27	Power-good LED
13	Five push buttons	28	Xilinx Artix-7 FPGA
14	Pmod port(s)	29	DDR2 memory
15	Temperature sensor		

Tabla 3.2. Descripción de puertos de la placa Nexys A7

Para el desarrollo del sistema en VHDL se ha tenido en cuenta que el reloj principal de la tarjeta es de 50 MHz, con el cual se ha trabajado para generar las frecuencias requeridas para el funcionamiento del sistema. La placa de desarrollo tiene un total de 16 interruptores, los cuales se usarán para la selección de la intensidad de la lámpara y el mensaje de 4 bits a enviar.

3.4. Bloque del generador de frecuencias.

Es fundamental el uso de distintas frecuencias para poder realizar todas las labores del sistema correctamente. Por ello se ha dedicado un bloque independiente a la generación de hasta cuatro relojes diferentes para la generación de la señal de salida (transmisor) y el muestreo de la señal de entrada (receptor). Como ya se ha mencionado anteriormente, la placa dispone de un reloj de 50 MHz, desde el cual se van a conseguir el resto de las frecuencias con unas cuentas de ciclos de reloj que estarán parametrizadas para facilitar la manipulación de estos relojes.

Previamente a la realización del código, se han realizados pruebas con la lámpara LED y la electrónica, a distintas frecuencias, para establecer el valor más adecuado para trabajar. Luego de diversas pruebas (frecuencias entre 50 kHz – 200 kHz), se ha decidido trabajar con una frecuencia para el mensaje ya modulado de 100 kHz o de 100 kbps.

Serán necesarios cuatro relojes diferentes que se van a enumerar a continuación. La selección de frecuencias se ha hecho tal que facilite el valor de cuentas que hay que realizar sobre el reloj interno.

a) Reloj para el mensaje original de cuatro bits (*clk_1*). Se utilizará un reloj de 66.67 kHz, con un número de cuentas de 1500, ver ecuación 3.2.

$$f_{clk1} = \frac{f_{syst}}{nº\ cuentas\ clk1} \rightarrow nº\ cuentas\ clk1 = \frac{f_{syst}}{f_{clk1}} = 1500 \quad (3.2)$$

b) Reloj para el mensaje codificado (*clk_2*). Se utilizará un reloj de 100 kHz, con un número de cuentas de 1000, ver Ecuación 3.3. El número de cuentas depende del tipo de codificación RLL utilizada, en este caso 4B6B.

$$f_{clk2} = \frac{f_{syst}}{nº\ cuentasclk2} \rightarrow nº\ cuentasclk2 = \frac{f_{syst}}{f_{clk2}} = 1000 \quad (3.3)$$

c) Reloj para la modulación (*clk_3*). Se utilizará un reloj de 1MHz, con un número de cuentas de 100, ver Ecuación 3.4. Este reloj es diez veces más rápido que *clk_2*, ya que se utilizará para la modulación VOOK de la señal en tramos de 10%. Se ha

elegido así para poder dividir la señal de salida en diez partes, como se pudo ver en la introducción del método de modulación VOOK.

$$f_{clk3} = \frac{f_{syst}}{nº\ cuentasclk3} \rightarrow nº\ cuentasclk3 = \frac{f_{syst}}{f_{clk3}} = 100 \quad (3.4)$$

d) Reloj para el muestreo de la señal recibida (*clk_sampling*). Se utilizará un reloj de 5 MHz, con un número de cuentas de 20, ver Ecuación 3.5. Este reloj se utilizará para poder recoger al menos 9-10 muestras de la señal recibida en cada tramo.

$$f_{clk_{samp}} = \frac{f_{syst}}{nº\ cuentasclk\_samp} \rightarrow nº\ cuentasclk\_samp = \frac{f_{syst}}{f_{clk_{samp}}} = 20 \quad (3.5)$$

A continuación, ver Figura 3.16, se muestran las señales de reloj simuladas, donde se puede ver la magnitud de cada una de ellas.

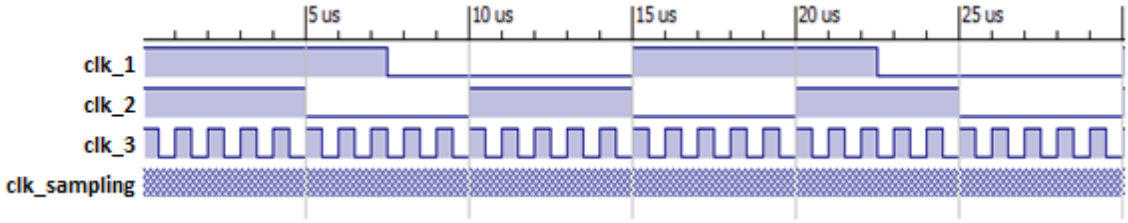


Figura 3.16. Señales creadas a diferentes frecuencias en el generador de relojes.

Por último, se utilizará una señal de reset para reiniciar los relojes manualmente mediante un pulsador en la placa. Por lo tanto, este bloque necesitará de dos entradas (reloj de 50 MHz y reset) y de cuatro salidas, las cuatro señales implementadas, ver Figura 3.17.

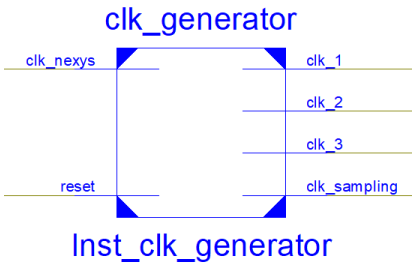


Figura 3.17. Módulo generador de relojes

3.5. Bloque del transmisor

El bloque del transmisor, ver Figura 3.18, engloba todos los procesos para transformar un vector de 4 bits en una señal codificada en 4B6B y modulada en VOOK. En primer lugar, se realiza la codificación basándose en la Tabla 2.4, anteriormente expuesta en este trabajo. Utilizando las señales de reloj descritas en el apartado anterior y el porcentaje de iluminación

(definido como entrada al bloque) se elabora una señal modulada en VOOK a transmitir con la lámpara LED. Las señales de reloj implementadas, salvo la del reloj de muestro, una señal de reset, el mensaje de cuatro bits y el nivel de iluminación, los dos últimos valores determinados en la placa Nexys A7 mediante los interruptores, son las entradas al bloque del transmisor. Las salidas de este bloque son tres señales, una salida hacia la lámpara LED y otras dos para comprobaciones de funcionamiento. La primera es la señal con el mensaje codificado en 4B6B y modulado en VOOK (*modulated_msg*), y las otras dos (*msg_trans* y *msg_trans_cod*) son señales para comprobación del mensaje de cuatro bits a transmitir y del mensaje codificado a 4B6B respectivamente.

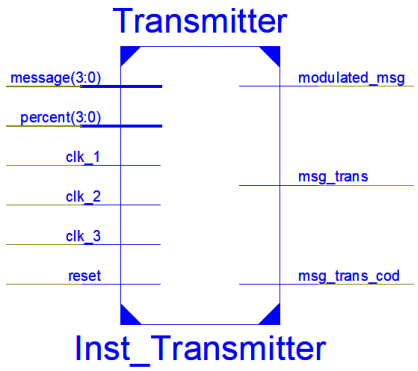


Figura 3.18. Módulo transmisor del mensaje

A su vez, el bloque del transmisor se ha dividido en tres módulos para generar las tres señales de salida mencionadas anteriormente, ver Figura 3.19. Todos los bloques, a su vez, incorporan una señal de reset para reiniciar todos los procesos y las variables de estos.

- Un generador de señal del mensaje de cuatro bits sin codificar ni modular.
- Un generador de la señal del mensaje codificado 4B6B y del vector del mensaje de 6 bits.
- La modulación VOOK del mensaje de 6 bits dependiente del porcentaje de iluminación.

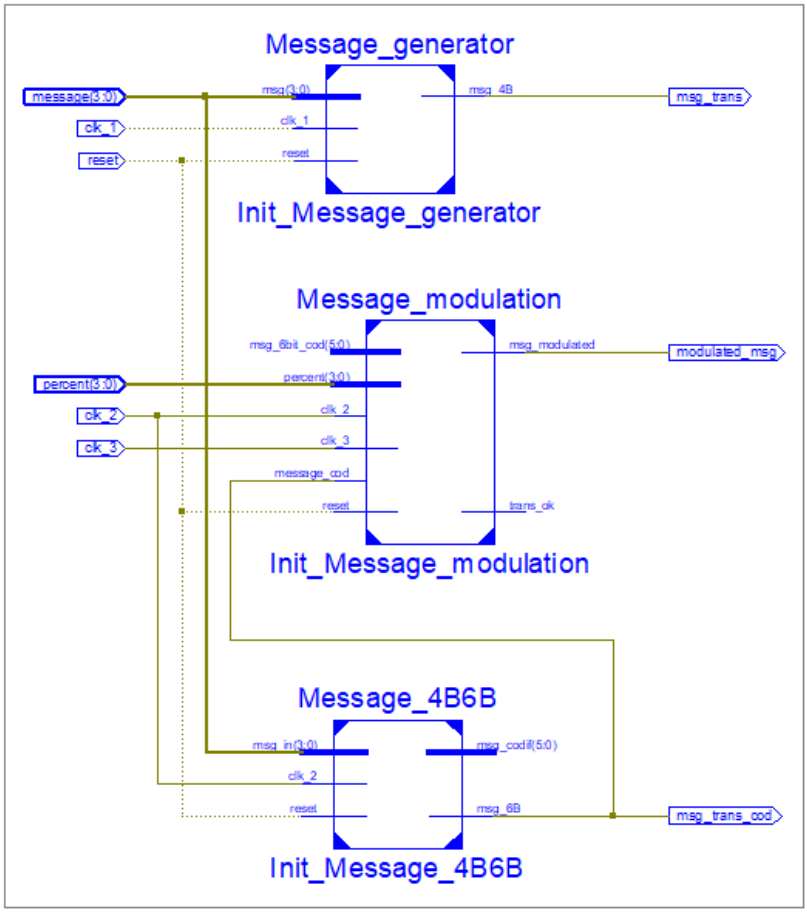


Figura 3.19. Módulo transmisor en mayor detalle

Dentro del programa se ha implementado un proceso que impide el cambio del mensaje de 6 bits durante la transmisión de este. Únicamente se enviará un mensaje diferente al finalizar la transmisión del mensaje en curso. Por ello, se puede apreciar en la Figura 3.19 que la señal de salida de la instancia "*Message_4b6b*" con el vector de 6 bits con el mensaje codificado no va directamente al módulo para la modulación VOOK del mensaje de salida.

Cada uno de los bloques del transmisor generan una señal de salida hacia los puertos de la tarjeta de desarrollo. Los bloques *Message_4B6B* y *Message_generator* simplemente generan señales para comprobación de funcionamiento utilizando el primero el reloj de 100 kHz y el segundo el reloj de 66,67 kHz.

Por último, el bloque *Message_modulation* comprende la modulación de la señal a transmitir a través del LED. Por esto, se necesita como entrada el mensaje de 6 bits, el porcentaje de iluminación elegido (10% hasta 90%) y los relojes de 100 kHz y 1 MHz. Como se ha mencionado anteriormente, para implementar la modulación VOOK con intensidad lumínica variable, es necesario establecer los valores de unos y ceros dependientes del porcentaje de

iluminancia elegido. A continuación, ver Tabla 2.1, se muestra gráficamente la composición de cada bit del símbolo dependiente del porcentaje elegido. Este bloque también dispone de

Porcentaje	Cero											Uno									
90%	0	0	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1
80%	0	0	0	0	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1
70%	0	0	0	0	0	0	1	1	1	1		1	1	1	1	1	1	1	1	1	1
60%	0	0	0	0	0	0	0	0	1	1		1	1	1	1	1	1	1	1	1	1
50%	0	0	0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1	1	1
40%	0	0	0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1	0	0
30%	0	0	0	0	0	0	0	0	0	0		1	1	1	1	1	1	0	0	0	0
20%	0	0	0	0	0	0	0	0	0	0		1	1	1	1	0	0	0	0	0	0
10%	0	0	0	0	0	0	0	0	0	0		1	1	0	0	0	0	0	0	0	0

Tabla 3.3. Detalle de los diferentes símbolos dependiente del porcentaje de intensidad lumínica.

Se puede observar en la tabla anterior, que se dispone de diez posiciones a modificar dependiente de los valores del porcentaje de intensidad lumínica seleccionada para cada bit. Sin embargo, para simplificar el código han agrupado las posiciones por parejas, teniendo que modificar únicamente la mitad de las posiciones, cinco, para variar la intensidad de la lámpara. Esto significa que la señal de salida se puede dividir en cinco secciones o "*slots*" por cada bit.

Además, dentro del bloque se han creado dos procesos, ver Figura 3.20, uno para la modulación VOOK del mensaje que se ha mencionado en párrafos anteriores y el otro para seleccionar el valor que deben tener los unos y los ceros en función del porcentaje de intensidad lumínica seleccionado con los interruptores de la tarjeta Nexys A7. Esos valores se representan como dos vectores de cuatro bits.

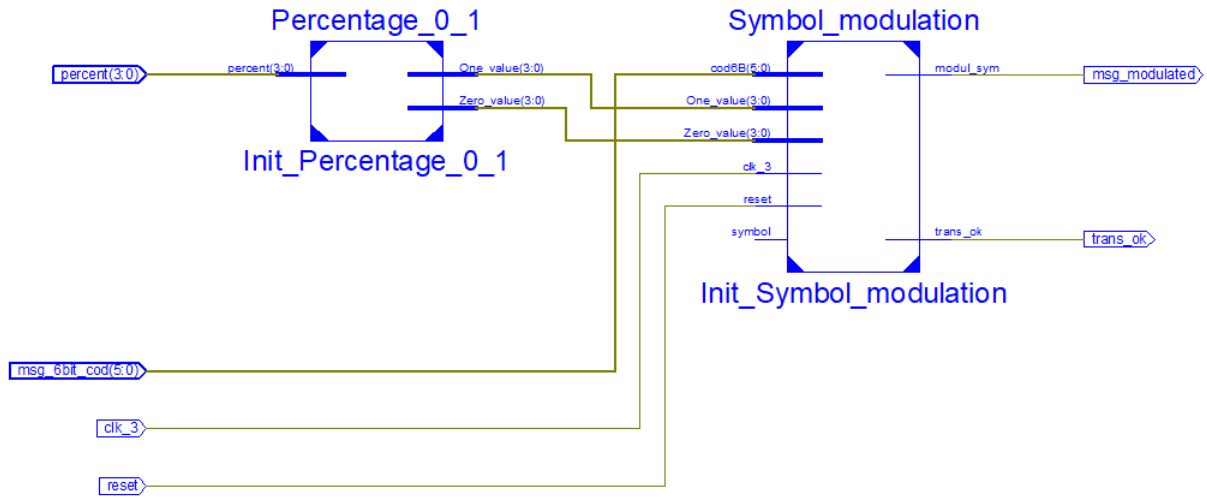


Figura 3.20. Detalle del bloque Message_modulation

Estas señales de cuatro bits se envían directamente a la instancia que realiza la modulación VOOK. Ambos valores indican el número de ciclos del reloj de 1 MHz (10 ciclos necesarios para modular cada bit) que debe estar la señal en alta o en baja dependiendo del valor del bit a transmitir. Por otro lado, cómo se había indicado, los porcentajes de intensidad 0% y 100% no están permitidos en la modulación VOOK y en el código estos valores no son seleccionables.

El proceso encargado de realizar la modulación del mensaje recibe los valores mencionados en el párrafo anterior y además el mensaje de 6 bits a modular. Además, se introduce un retardo para sincronizar la señal con los pulsos de reloj, y un pulso inicial para que el receptor entienda que se comienza a enviar el mensaje. Se envía primero el LSB (*Less Significant Bit*) para ir cambiando de posición en el vector del mensaje (6 bits) hasta llegar al MSB (*Most Significant Bit*). A la hora de generar la señal de salida, no se tendrá en cuenta ningún cambio en el mensaje que se introduzca mientras se está realizando la transmisión, se procederá al cambio una vez enviado el mensaje.

3.6. **Bloque del receptor**

Una vez finalizado el transmisor, se va a describir funcionalmente el bloque que contiene el receptor del sistema. Este bloque se dividió en dos procesos simples y uno más complejo para desarrollar las tareas que tiene encargadas esta etapa del sistema de transmisión, ver Figura 3.21. Los dos módulos más sencillos se encargan, uno, de generar una señal de habilitación para activar la lectura, y otro, de generar una señal dependiente de la entrada de modo que pequeñas fluctuaciones por ruido que pudieran generarse, no sean apreciables por el resto del sistema.

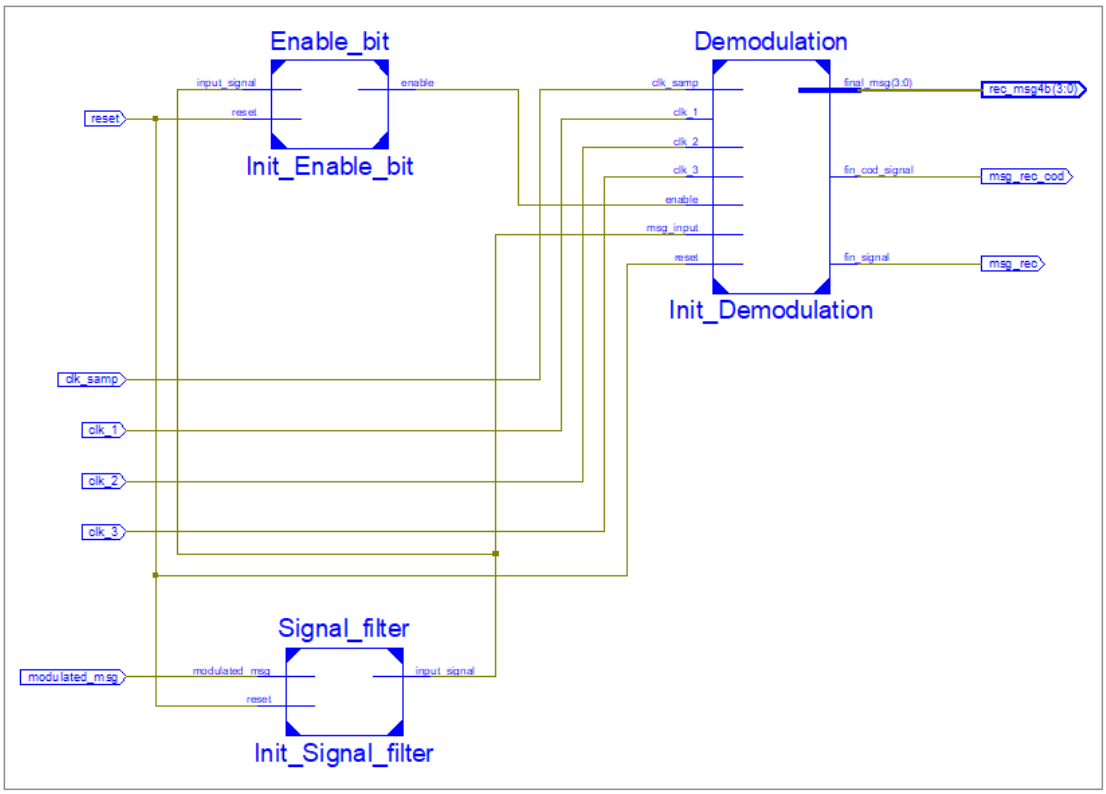


Figura 3.21. Procesos del módulo receptor

Hay que destacar que el receptor no tiene implementado un módulo de resincronización, por lo que, si se pierde la señal, se pierde el mensaje y pueden aparecer errores de lectura hasta que vuelva a recuperar la señal.

En primer lugar, para lograr la demodulación y decodificación completa de la señal se ha dividido el demodulador en tres bloques (*correlation, decision, message_decod*), ver Figura 3.22, que engloban los procesos de demodulación y decodificación del mensaje. Sin duda el aspecto más importante va a ser la correlación de la señal, que va evaluando la señal a medida que se va recibiendo.

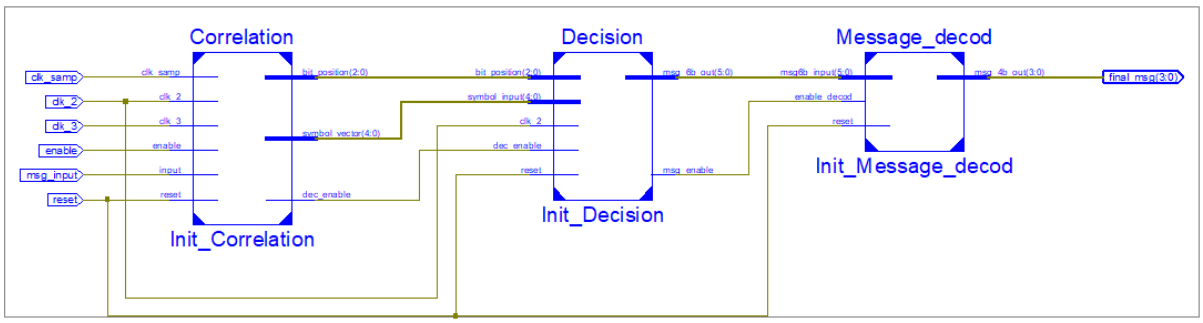


Figura 3.22. Etapa de demodulación y decodificación

La señal de entrada pasa por un proceso de muestreo donde se estudia en el tiempo de bit (velocidad de 100 kbps) para decidir si se trata de un "cero" o un "uno". Una vez obtenido el vector de 6 bits del mensaje se realiza la decodificación de este para obtener el mensaje de 4 bits que se pretendía transmitir en el sistema. A continuación, se detalla cada proceso elaborado en estos tres módulos.

3.6.1. Muestreo de la señal y decisión

A la hora de decodificar la señal es necesario saber qué valor tiene la misma, por ello existe esta etapa de muestreo y decisión de bit. Como ya se había mencionado anteriormente, existe una señal de habilitación, la cual hace saber al módulo que puede empezar a estudiar la señal de entrada. Una vez habilitada la lectura, se recogen muestras durante dos ciclos de reloj de 1 MHz, un slot. Se divide, por tanto, la lectura del símbolo en cinco partes o *slots*, dando como resultado una variable de cinco componentes. Una vez formado este vector, que corresponde a un bit de la señal, se pasa este a una etapa de decisión. Con el veredicto se va completando progresivamente otra variable de seis componentes, conteniendo el mensaje aún modulado 4B6B.

Para poder llegar a decidir si el valor recibido es un cero o un uno hay que estudiar previamente las condiciones de las señales que se pueden recibir. El receptor desconoce el porcentaje de regulación de la intensidad lumínica, porque lo va a tener que comparar lo recibido con las señales posibles que se pueden recibir. Sólo existen diez señales posibles, sin ruido, que llegan al receptor, las cuales dependen directamente de cinco funciones base $\varphi_i(t)$, ver Ecuación 3.6.

$$s_i(t) = a_1\varphi_1(t) + a_2\varphi_2(t) + a_3\varphi_3(t) + a_4\varphi_4(t) + a_5\varphi_5(t) \qquad (3.6)$$

Donde:

- s_i: representa las distintas señales de cada símbolo del mensaje
- a_i: representan las proyecciones sobre las funciones base de las posibles señales transmitidas
- φ_i: representan a las funciones base utilizadas.

Las funciones base utilizadas son cinco en total, cinco pulsos unitarios con $\frac{T_S}{5}$ de longitud, o lo que es lo mismo, un pulso unitario en cada uno de los cinco *slots* en los que se ha dividido

cada símbolo, ver Tabla 3.4. Como se ha mencionado sólo existen diez señales posibles, siempre que no haya ruido, que se pueden recibir.

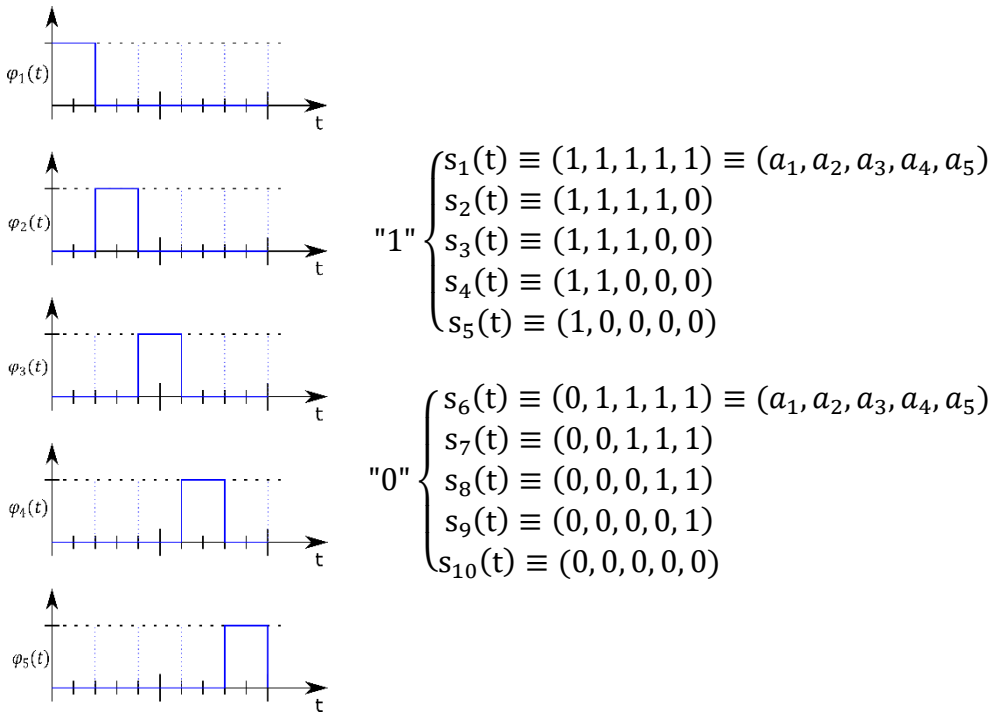


Tabla 3.4. Funciones base y símbolos posibles.

A continuación, se muestra la representación gráfica, ver Figura 3.23, de las diez posibles señales a recibir, separando los unos y los ceros.

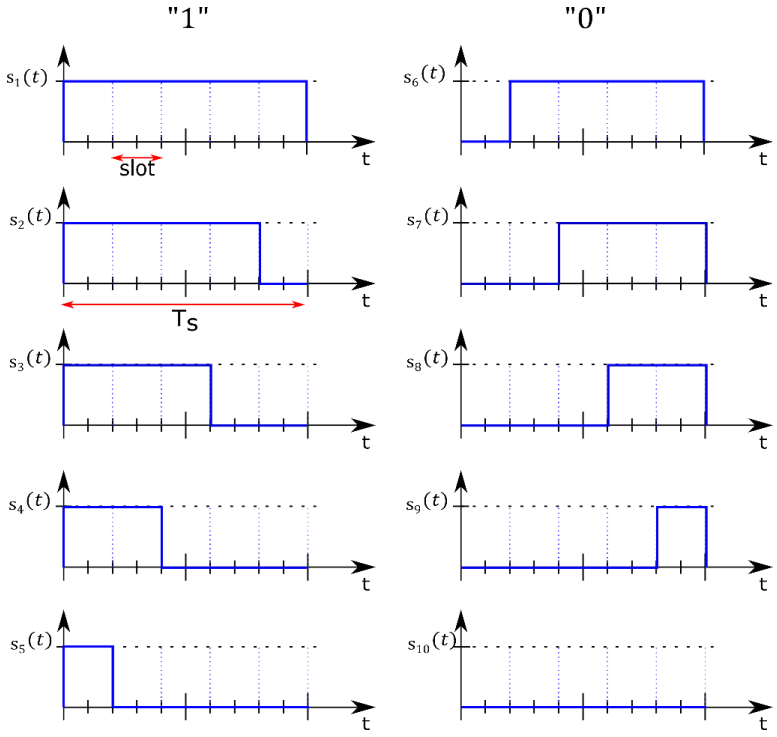


Figura 3.23. Símbolos posibles para transmitir

Una vez que la señal recibida ha sido muestreada, se trabaja para conseguir una señal demodulada o estimada de la primera. Para ello, como se ha ido describiendo en este punto, se necesita conocer el valor de cinco componentes por símbolo. La manera de sacar esta información es mediante el uso de las funciones base y las proyecciones de la señal sobre estas. A continuación, ver Figura 3.24, se puede ver una representación gráfica de esto. El vector de cinco componentes, ver Ecuación 3.7, estaría formado por estas cinco proyecciones sobre las funciones base (unitarias) dando lugar a una de las diez posibilidades mencionadas anteriormente.

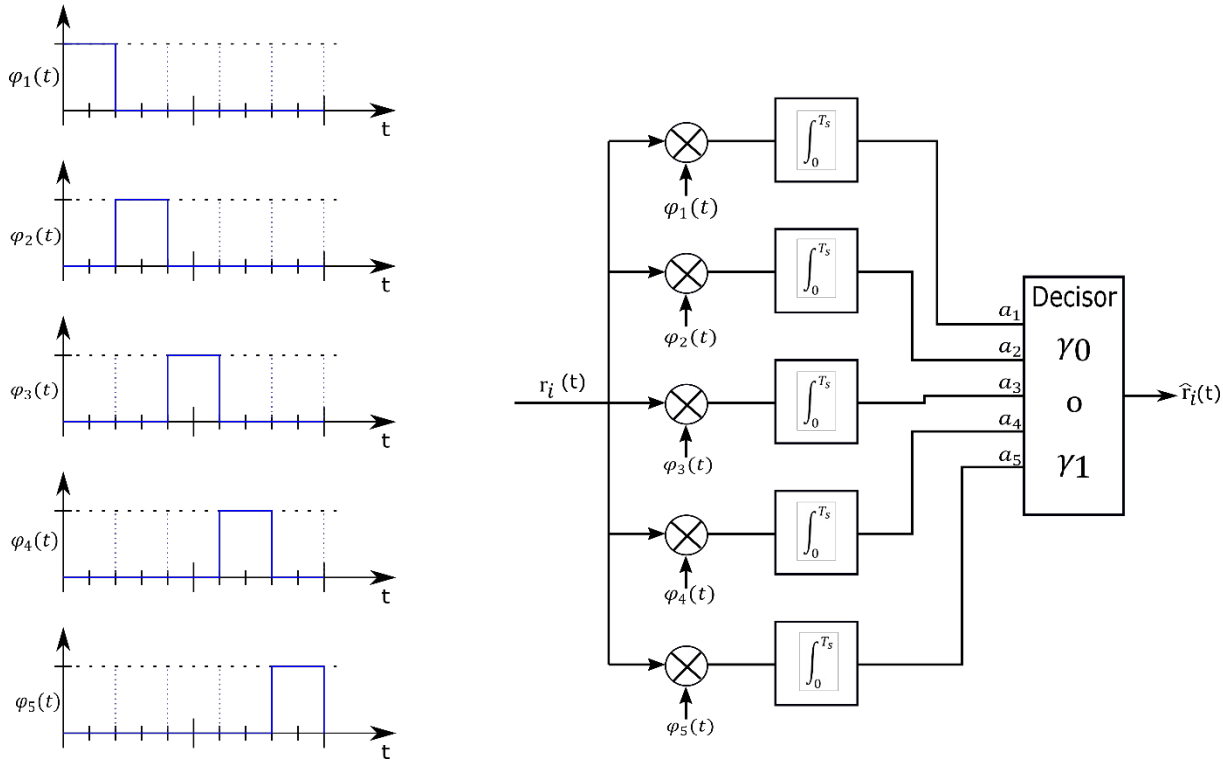


Figura 3.24. Señales base y proceso de correlación de la señal.

$$\frac{1}{T_s}\int_0^{T_s} r_i(t)\cdot\varphi_j(t)dt = a_{ij} \qquad (3.7)$$

El decisor, dependiendo del vector recibido, elige el valor del símbolo recibido y cumplimenta una variable que, cuando se complete, será el mensaje codificado en 4B6B. Es importante recordar que no se dispone de información sobre el porcentaje de intensidad lumínica de la lámpara. Por lo tanto, a la hora de seleccionar si se trata de un uno o un cero, tendrá que comparar el vector recibido con los posibles valores. El decisor utiliza un criterio de mínima distancia euclídea para decidir el valor de la componente del mensaje. Sin embargo, existen casos en los que la distancia euclídea mínima no es única, y se puede identificar tanto

como un cero o como un uno. En estos casos, se recurre al valor de la componente de la proyección sobre $\varphi_1(t)$, de forma que:

- Si $a_1 = 1$. Se decide por un valor de uno, interpretando que la señal $s_i(t)$ corresponde a un uno ($i = 1, 2, 3, 4, 5$), dependiendo de las demás componentes de a_i ($i \neq 1$)
- Si $a_1 = 0$. Se decide por un valor de uno, interpretando que la señal $s_i(t)$ corresponde a un uno ($i = 6, 7, 8, 9, 10$), dependiendo de las demás componentes de a_i ($i \neq 1$)

De esta forma se reduce potencialmente el número de errores que puedan deberse a ruido en la señal.

Una vez el decisor haya cumplimentado la variable de seis componentes que representa al mensaje codificado, se habilita el decodificador para recuperar el mensaje original de cuatro bits. Haciendo el mismo proceso que el codificador, mediante una tabla se recupera este mensaje original. El resultado de salida de este módulo es el mensaje transmitido por el sistema inicialmente. Con esta etapa se cierra el proceso, donde se consigue comunicar un mensaje a través de luz visible mediante una lámpara LED. Este mensaje decodificado se muestra en la placa de desarrollo mediante 4 pequeños LED encima de los interruptores usados para elegir el mensaje.

Todos los procesos dentro del demodulador están relacionados entre sí mediante señales de habilitación y de cuentas para identificar las posiciones de los vectores intermedios utilizados en el código. Con ello se logra facilitar la sincronización de procesos y evitar un desfase en alguna variable.

Capítulo 4. Simulaciones y resultados obtenidos

Índice simulaciones y resultados obtenidos

La parte final de este trabajo ha consistido en diferentes pruebas de laboratorio para comprobar el correcto funcionamiento del sistema. Sin embargo, previamente se ha tenido que testear por separado los distintos elementos del sistema. Inicialmente se han realizado pruebas con el circuito electrónico, usando un generador de señales como fuente de información. Los siguientes pasos han consistido en el desarrollo del código, haciendo uso de simulaciones para depurar el mismo. Finalmente, cuando se han completado todos estos pasos, se ha procedido a las pruebas finales del sistema completo.

A lo largo de este capítulo, se expondrán los resultados obtenidos con el sistema de comunicación mediante luz visible. Como ya se ha mencionado, el enlace óptico se realizará con una lámpara LED, ver Figura 4.1, y un circuito receptor con fotodiodos. El mensaje a transmitir se indica en la placa de desarrollo mediante interruptores, al igual que el nivel de intensidad lumínica, y consta de cuatro bits. Este mensaje posteriormente es codificado 4B6B para evitar los problemas de oscurecimientos y parpadeos de la lámpara LED, que ya se han mencionado en esta memoria.

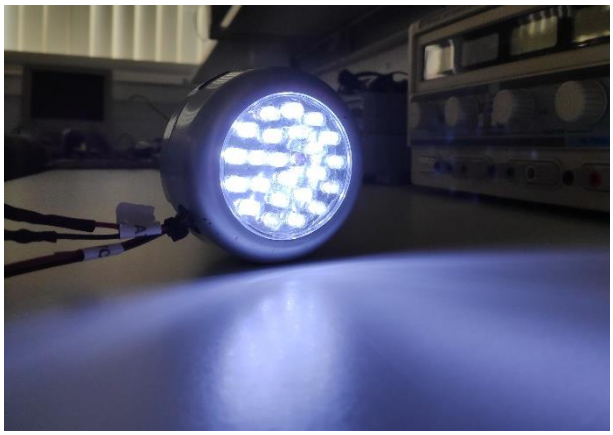


Figura 4.1. Lámpara LED utilizada en el sistema VLC

A continuación, se muestran los resultados de las simulaciones de los distintos bloques del sistema y, posteriormente, las pruebas con el sistema completo.

4.1. Pruebas del circuito receptor

En primer lugar, las pruebas de laboratorio se han comenzado con únicamente el amplificador transimpedancia, aunque se han ido implementando secuencialmente el resto de los componentes del circuito y realizando pruebas de funcionamiento durante el avance. Todo el circuito se ha montado sobre una protoboard alimentada con ±5 V y GND. Previamente ya se habían realizado algunas simulaciones para comprobar el circuito.

Inicialmente se trabajó con un solo fotoreceptor de Hamamatsu, modelo S3071, obteniendo resultados poco favorables. Debido a un área activa pequeña, la respuesta era débil e inestable desde que la lámpara era separada a más de 20-30 cm del fotodiodo. La solución ha sido la de incluir un segundo fotodiodo en paralelo para aumentar así el área activa, consiguiendo trabajar bien a distancias entre 60-70 cm, ver Figura 4.2.

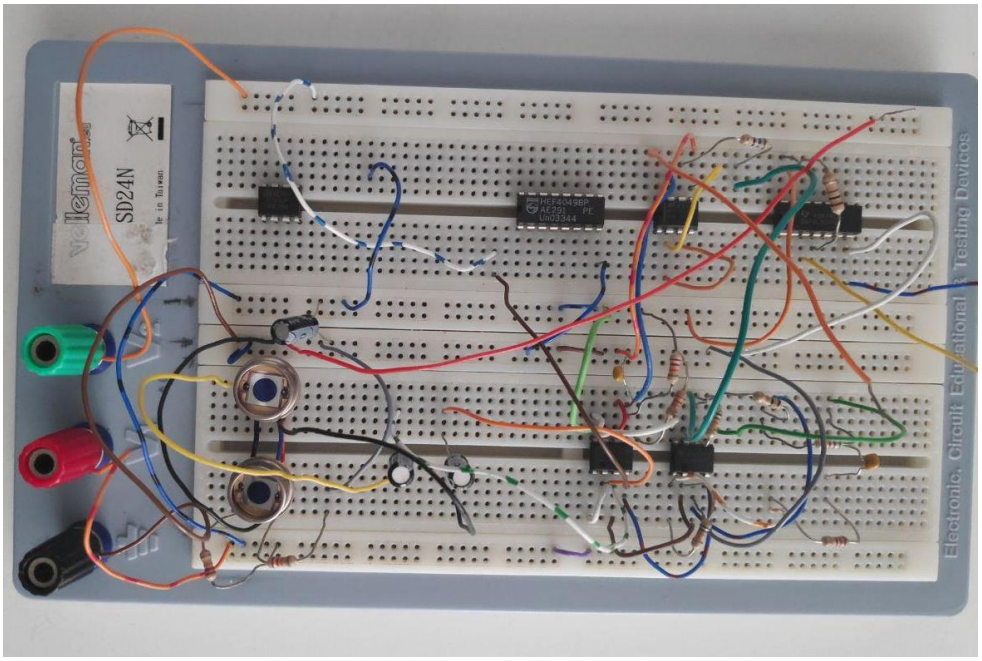


Figura 4.2. Circuito dispuesto en una protoboard

Para realizar las pruebas del circuito se ha alimentado la protoboard con ±5 V y tierra. Se ha utilizado como fuente de información simulada, una señal cuadrada de frecuencia de 100 kHz. Inicialmente se probaron frecuencias mayores, pero no se obtuvieron los resultados esperados y se optó por la frecuencia mencionada anteriormente. Las pruebas finalizan con la inclusión de un MAX3485 para enviar la señal a la placa con la FPGA, pero previamente se probaron otras configuraciones que no aportaban los resultados esperados. Una vez todos los componentes dispuestos se realizan las pruebas con el circuito completo, ver Figura 4.3. Es importante destacar que en la ubicación del laboratorio se han tenido que apagar las lámparas fluorescentes, ya que ocasionaban mucho ruido en la señal, dificultando las pruebas.

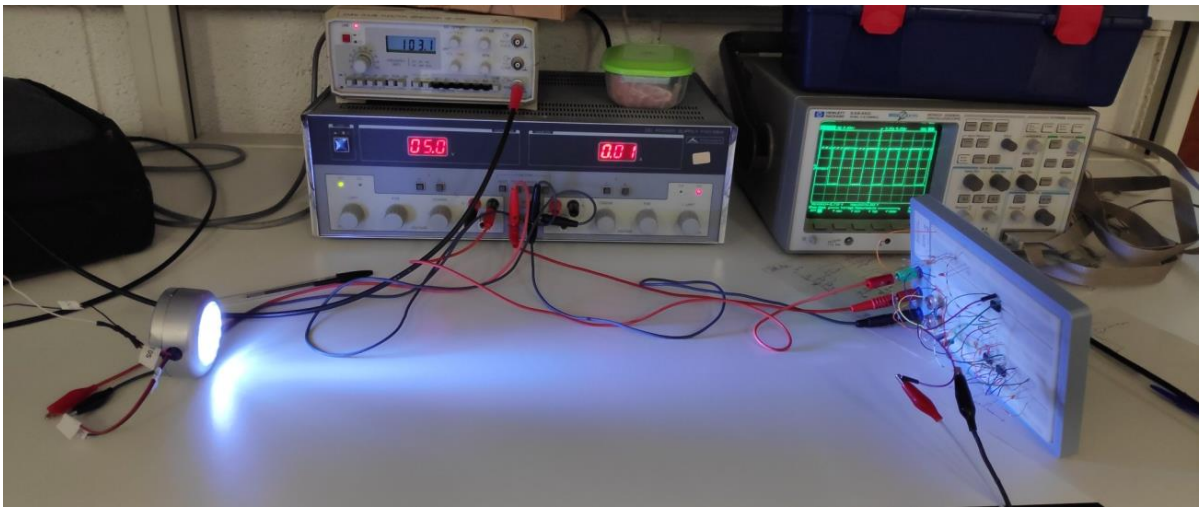


Figura 4.3. Disposición de los componentes para las pruebas de circuito.

Una vez comienzan las pruebas y, tras el ajuste de algunos parámetros, se consiguen realizar pruebas satisfactorias con el circuito, ver Figura 4.4. La señal superior es la entrada al circuito MAX3485 y la inferior es la salida, adecuada a los valores interpretables por la FPGA (0 – 3,3 V). Con estas pruebas se establece el correcto funcionamiento del acondicionamiento de la señal y se pasa a las simulaciones del código.

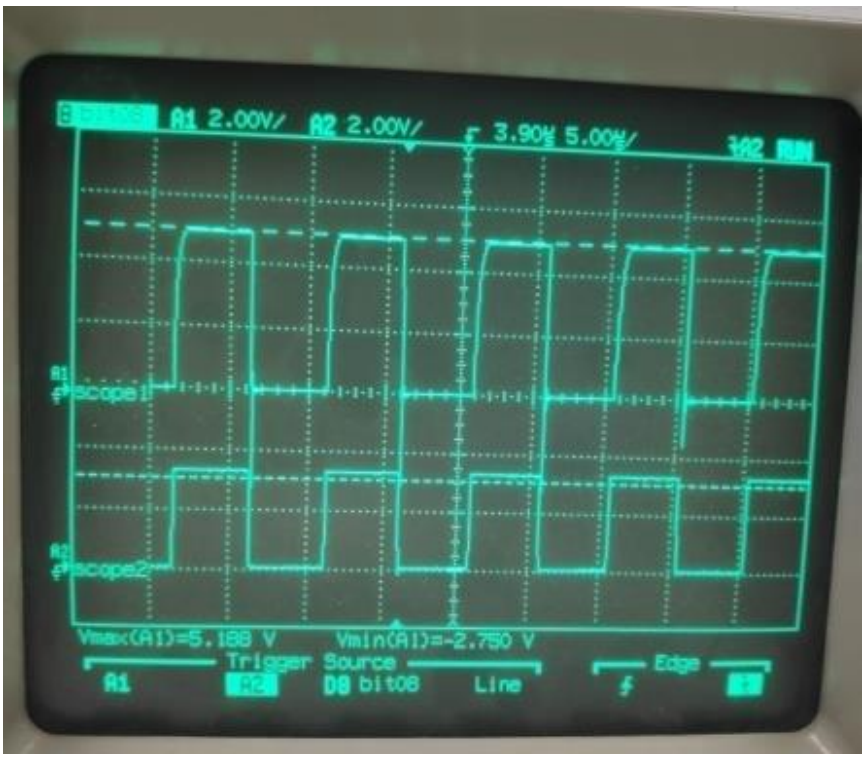


Figura 4.4. Resultados a la salida del circuito con una señal cuadrada de 100 kHz

4.2. Simulaciones del código

Es de gran importancia la coordinación y el sincronismo entre las diferentes etapas del sistema, es indispensable para el correcto funcionamiento de este. Por ello, todas las etapas tienen que trabajar bien en equipo, pero primero tienen que funcionar por separado. Para lograr que las simulaciones sean fácilmente interpretables, se han elaborado varias señales de comprobación y se han sincronizado con los relojes creados en el sistema. En primer lugar, se ha comprobado el funcionamiento de estos relojes, para poder comenzar las simulaciones de la etapa o bloque de transmisión del mensaje. Posteriormente se ha comprobado la correcta transmisión del mensaje, se han realizado simulaciones con el receptor. En este punto se van a mostrar y describir los resultados de las simulaciones del transmisor y el receptor.

4.2.1. Simulaciones del transmisor

La finalidad principal del módulo transmisor es generar una señal que va a ser enviada a través de una lámpara LED. Sin embargo, para conseguir esta señal, se ha pasado por un proceso donde se han creado varias señales y variables intermedias que van evolucionando hasta conseguir la señal codificada y modulada que se va a transmitir. Por ello, en las diferentes ilustraciones que se muestran en este punto se ven varias señales que se definen a continuación:

- Msg_switch: Es el mensaje de cuatro bits de entrada (vector) que se indica en los interruptores de la placa de desarrollo.
- Msg_trans: Es el mensaje de cuatro bits transformado en una señal continua (se muestra de bit más significativo a menos significativo).
- Msg_trans_cod: Es el mensaje ya codificado (4B6B), de 6 bits, en forma de señal (se muestra de bit más significativo a menos significativo).
- Percent: el porcentaje elegido para la modulación desde la placa.
- Reset: La señal que reinicia todos los procesos del código.
- Msg_modul: el mensaje final, ya codificado y modulado (VOOK).

En primer lugar, ver Figura 4.5, se muestra la simulación del inicio del código, donde se puede apreciar que el mensaje se envía en $60\ \mu s$. El porcentaje de intensidad lumínica está fijado en un 10% (“0000” el mínimo posible) y todas están sincronizadas para empezar a la vez.

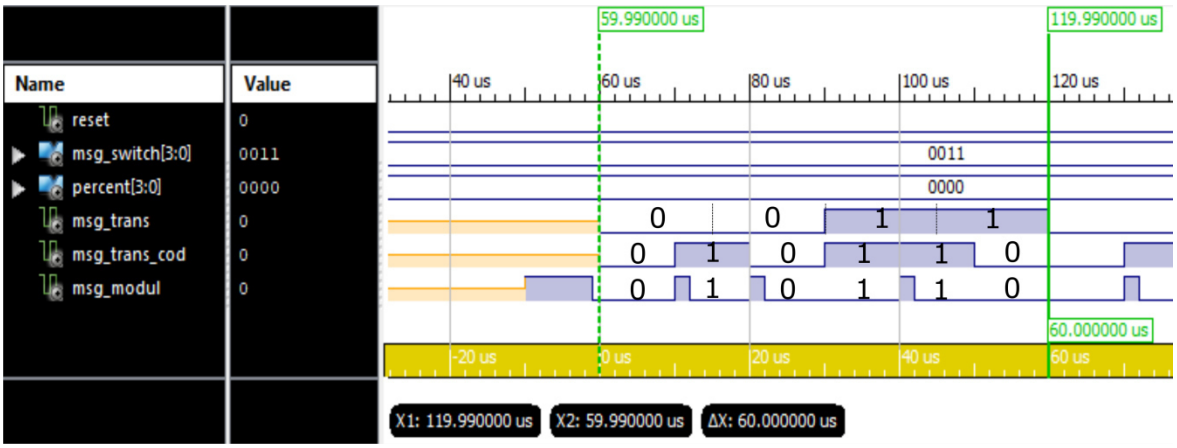


Figura 4.5. Simulación de la etapa de transmisión. Mensaje: "0011"

Por otro lado, para mostrar el funcionamiento del reset, ver Figura 4.6, se puede ver a continuación como se ponen a cero todas las señales. Se hace visible este efecto directamente en la lámpara LED, que se apagaría al pulsarlo. Se puede observar que mientras la señal del reset está activa, el mensaje de salida es cero.

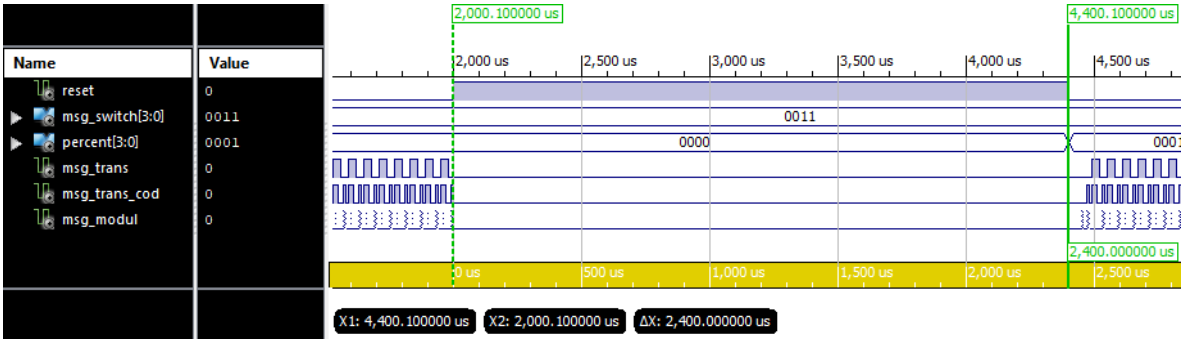


Figura 4.6. Simulación del reset del sistema

Por último, ha sido necesario introducir un pulso en la señal de salida, ver Figura 4.7, que el receptor interpreta para el comienzo de la lectura. Este pulso termina justo un ciclo de reloj de 1 MHz (1 µs) antes del comienzo del mensaje. Existe un retardo para el inicio de la transmisión del mensaje de 60 µs, y el pulso comienza en 50 µs para anticiparse a la transmisión del mensaje.

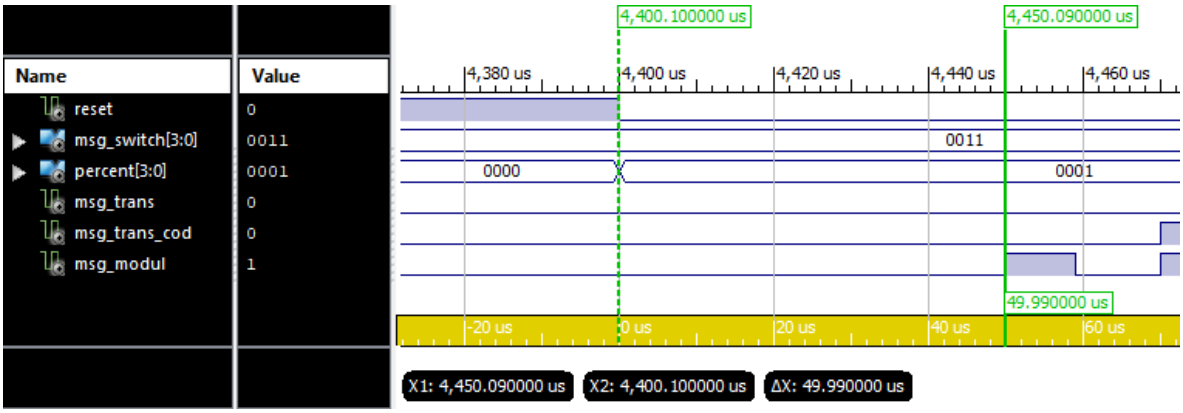


Figura 4.7. Pulso de inicio de lectura

4.2.2. Simulaciones receptor

Para hacer posibles las simulaciones del receptor se realiza un bypass de la salida del transmisor hacia la entrada del receptor. En este caso se lleva a cabo una demodulación y decodificación para acceder al mensaje original. Al igual que en el transmisor, se van a observar unas señales que necesitan ser definidas:

- Msg_switch: Es el mensaje de cuatro bits de entrada (vector) que se indica en los interruptores de la placa de desarrollo.
- Msg_trans: Es el mensaje de cuatro bits transformado en una señal continua (se muestra de bit más significativo a menos significativo).
- Percent: el porcentaje elegido para la modulación.
- Reset: La señal que reinicia las señales y las variables del código.
- Msg_modul: el mensaje final, ya codificado y modulado (VOOK).
- Msg4bit_receptor: Es el mensaje de cuatro bits, una vez demodulado y decodificado.
- Msg_demodul: Es la señal demodulada, se usa únicamente para comprobaciones.
- Msg_rec: Es la señal equivalente al mensaje recibido.
- Msg_6b_out: Es el mensaje demodulado de seis bits, aún codificado.

Al igual que en el transmisor, el módulo de receptor se ve alterado por la presencia del reset, ver Figura 4.8, y las señales se ponen a cero.

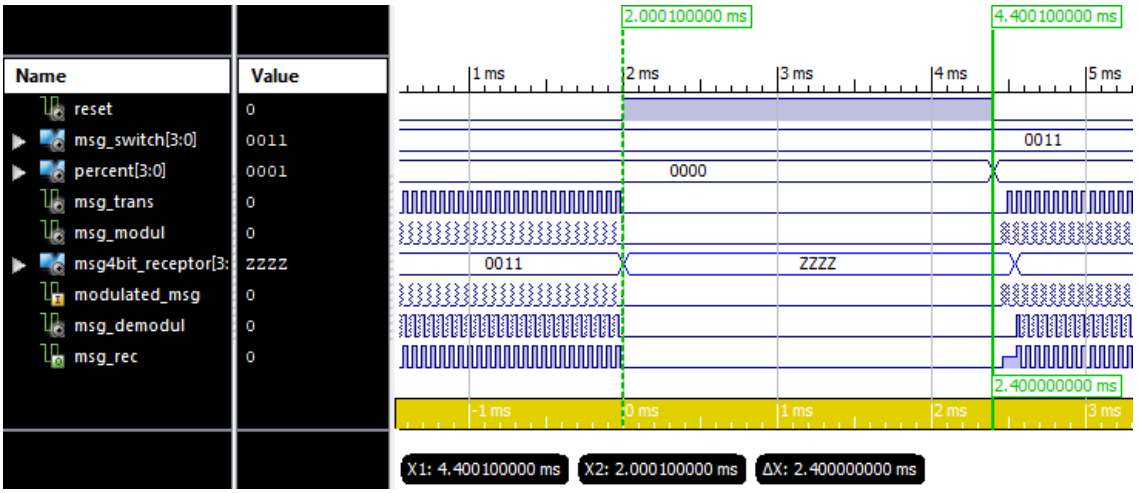


Figura 4.8. Simulación del reset con el módulo del receptor

A medida que se va recibiendo señal, ésta se pasa por el proceso de muestreo y decisión para determinar el valor de bit correspondiente. Una vez se haya rellenado el vector temporal de seis posiciones equivalente al mensaje de seis bits, este se vuelca a la variable final del mensaje codificado (vector del mismo tamaño). Como se puede observar (referenciar figura), la variable del mensaje recibido cambia 15 µs después de la finalización de transmisión del mensaje, ver Figura 4.9. Este retardo se ha introducido deliberadamente para evitar conflictos entre variables. El mensaje recibido se muestra una vez ha finalizado la transmisión, la generación de la señal está sincronizada con el mismo reloj que las señales del transmisor.

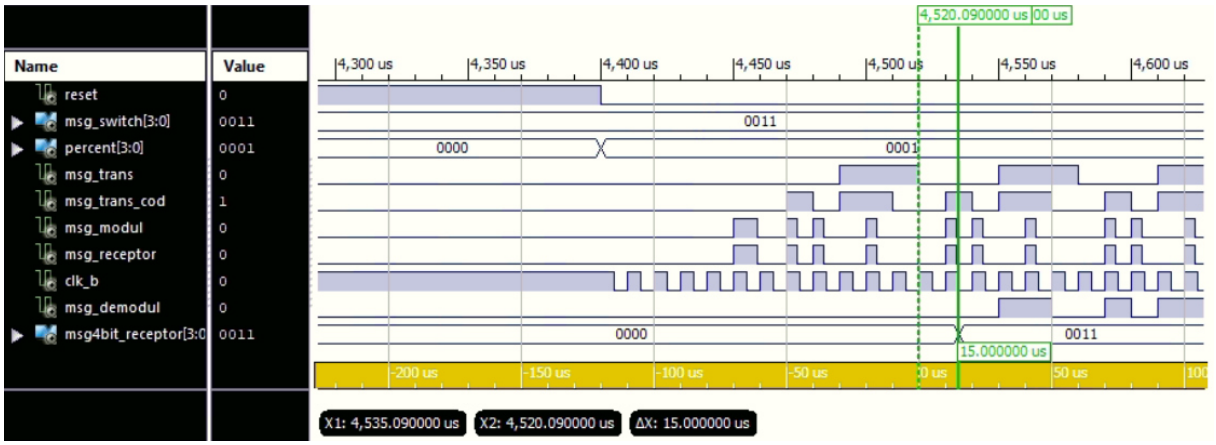


Figura 4.9. Retardo en la modificación de la variable que indica el mensaje decodificado.

4.3. Pruebas sistema completo

Para finalizar con los ensayos, se han realizado pruebas de laboratorio con el sistema completo. Esto ha dado lugar a modificaciones del código para adaptar el sistema, ver Figura 4.10, y conseguir un correcto funcionamiento de este.

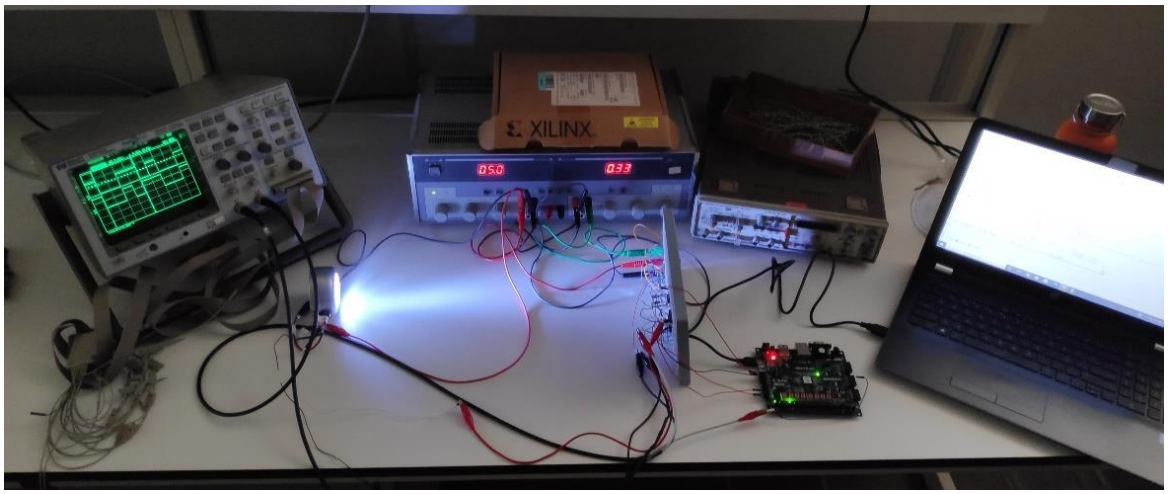


Figura 4.10. Montaje del sistema completo

Desde un primer momento la sincronización de señales ha funcionado adecuadamente y la transmisión del mensaje se realiza sin inconvenientes. Sin embargo, el receptor es el que ha tenido que sufrir cambios con respecto a la estructura del código original. Para poder presentar en este trabajo los resultados obtenidos, se ha utilizado un analizador lógico RIGOL disponible en el laboratorio destinado a las pruebas. En este punto se muestran varias imágenes con señales del transmisor y receptor, se han utilizado distintos mensajes y distintas intensidades para poder observar las variaciones en las señales. Por otro lado, la placa de desarrollo es alimentada directamente por USB desde el PC. En la Figura 4.11 se muestra esto y se pueden observar también los LEDs encendidos que corresponden al mensaje recibido.

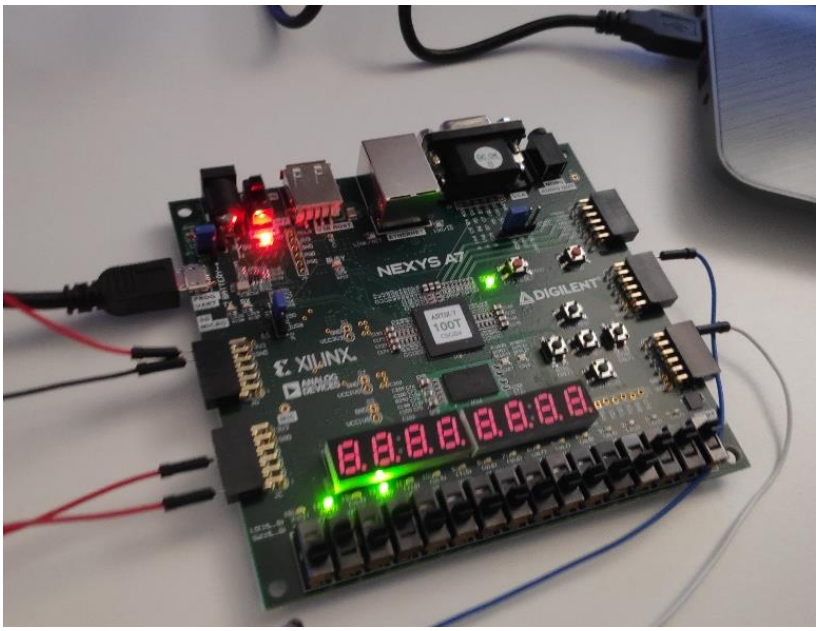
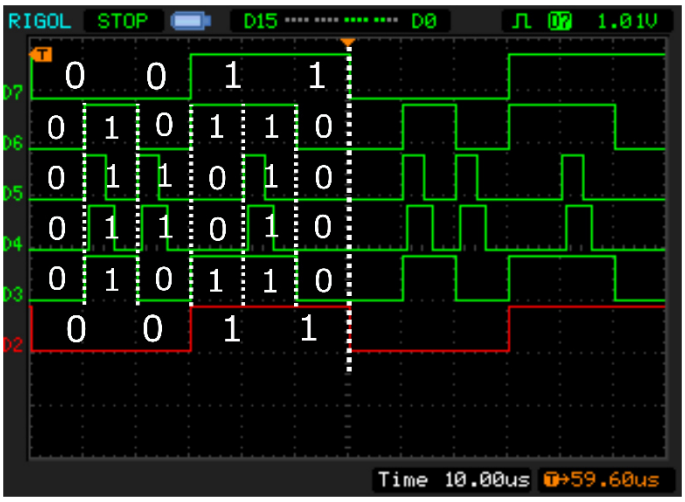


Figura 4.11. Placa de desarrollo Nexys A7 en funcionamiento con el sistema

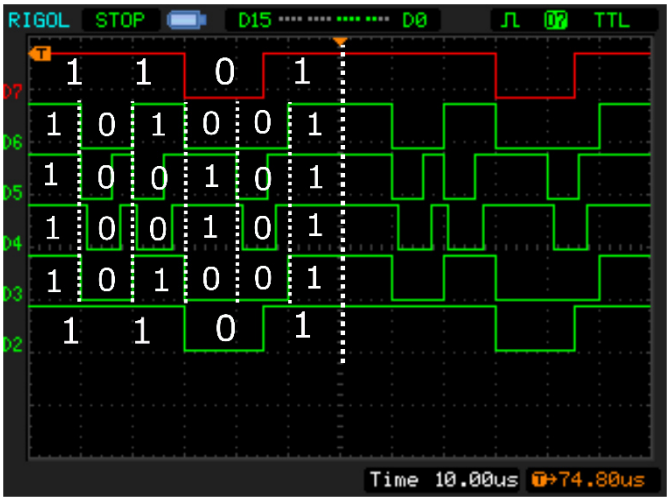
En la siguiente imagen, ver Figura 4.12, se muestra la evolución del mensaje en su transmisión completa. El mensaje a transmitir es "0011" con una intensidad del 20%. En primer lugar, la señal D7 representa al mensaje original, el indicado en los interruptores de la placa Nexys A7. El mensaje ya codificado en 4B6B se muestra en la señal D6 y, una vez modulado en VOOK, se muestra también en la señal D5. La señal D4 representa la transmisión que recibe la placa después de la adecuación. Una vez demodulado el mensaje se muestra en la señal D3, todavía codificado. Y, por último, en la señal D2 se muestra el mensaje recuperado.



D7: Mensaje original
D6: Mensaje codificado 4B6B
D5: Mensaje modulado VOOK
D4: Mensaje recibido
D3: Mensaje demodulado
D2: Mensaje recuperado

Figura 4.12. Transmisión de un mensaje ("0011") con un porcentaje de intensidad lumínica del 20%

También se puede observar el resultad, ver Figura 4.13, con un mensaje distinto, "1101", y una intensidad mayor, 70%.



D7: Mensaje original
D6: Mensaje codificado 4B6B
D5: Mensaje modulado VOOK
D4: Mensaje recibido
D3: Mensaje demodulado
D2: Mensaje recuperado

Figura 4.13. Transmisión de un mensaje ("1101") con un porcentaje de intensidad lumínica del 70%

Para finalizar, se muestra a continuación, ver las siguientes tres figuras, la evolución del ancho de los pulsos del mensaje a medida que se aumenta la intensidad de la lámpara.

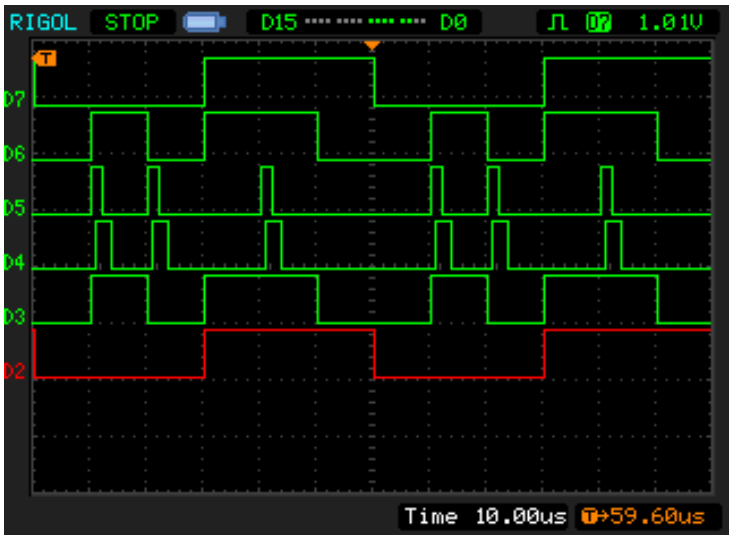


Figura 4.14. Evolución del ancho de pulso (10%) del mismo mensaje.

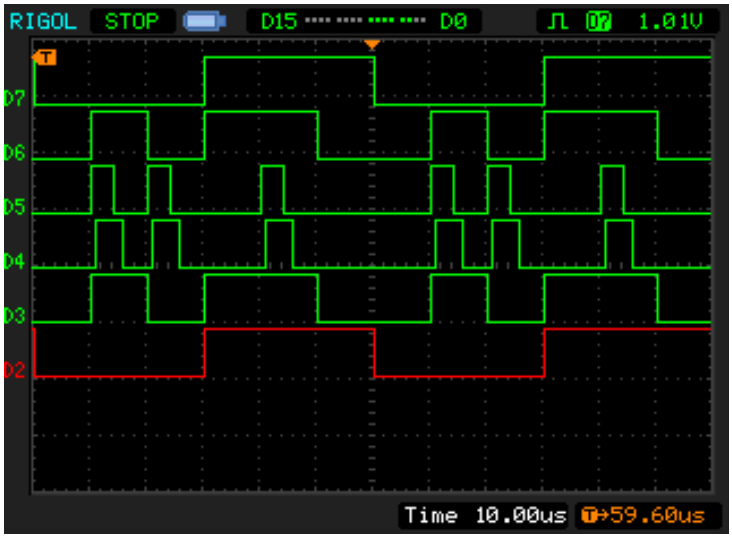


Figura 4.15. Evolución del ancho de pulso (20%) del mismo mensaje.

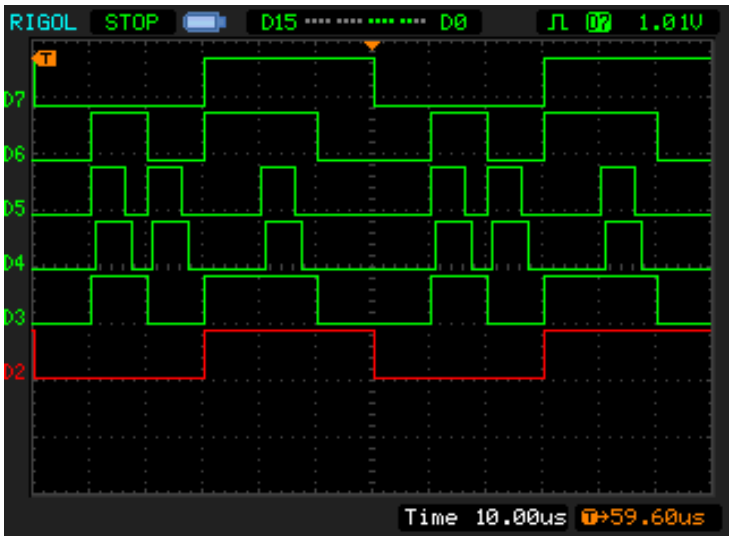


Figura 4.16. Evolución del ancho de pulso (30%) del mismo mensaje.

Capítulo 5. Presupuesto

A continuación, en este capítulo se muestra el presupuesto correspondiente a este trabajo. Se divide en coste de materiales y coste de mano de obra. Además, se incluyen gastos generales y beneficio industrial.

Concepto	Coste unitario (€)	Cantidad (u)	Coste total (€)
Tarjeta de desarrollo Nexys A7	244,57 €	1	244,57 €
Lámpara LED	8,50 €	1	8,50 €
Fotodiodo S3071 de Hamamatsu	12,63 €	2	25,26 €
Circuito integrado MAX3485	3,61 €	1	3,61 €
Circuito integrado TL082	0,55 €	2	1,10 €
Circuito integrado 74LS07	1,31 €	1	1,31 €
Resistencias (diferentes valores)	0,08 €	10	0,80 €
Condensador 10µF	0,18 €	2	0,36 €
Condensador 100nF	0,15 €	2	0,30 €
	Total de costes materiales (MT)		285,81 €

Concepto	Coste unitario (€/h)	Cantidad (h)	Coste total (€)
Tiempo de análisis	90,00 €	100	9.000,00 €
Tiempo de codificación	50,00 €	400	20.000,00 €
Tiempo de implementación	20,00 €	20	400,00 €
Tiempo de documentación	30,00 €	80	2.400,00 €
	Total de costes mano de obra (MO)		31.800,00 €

Total costes materiales (MT)	285,81 €
Total costes mano de obra (MO)	31.800,00 €
Gastos generales: 6% (MT+MO)	1.925,15 €
Beneficio Industrial: 13% (MT+MO)	4.171,16 €
Coste Total del Proyecto:	38.182,11 €

Capítulo 6. Conclusiones y bibliografía

Conclusiones

Durante el desarrollo de este Trabajo de Fin de Master se han alcanzado los objetivos propuestos al inicio del mismo. Se ha conseguido implementar un sistema de transmisión de información usando luz visible. Además, se han desarrollado un receptor óptico viable para diferentes tipos de modulación y un sistema implementado en FPGA con modulación VOOK. Todo esto precedido de correcciones y adaptaciones a los recursos disponibles, lo cual ha supuesto un reto interesante. El sistema implementado es capaz de transmitir un mensaje a través del enlace óptico, demodularlo y decodificarlo para recuperar el mensaje original.

En primer lugar, se ha conseguido diseñar un circuito acondicionador basado en un fotorreceptor como elemento principal. Éste es capaz de convertir la señal lumínica recibida en una señal en tensión interpretable por la FPGA.

En segundo lugar, se ha diseñado e implementado un sistema capaz de modular primero una señal en VOOK para la transmisión y luego de demodular la señal recibida para recuperar el mensaje original. Este sistema se ha desarrollado en VHDL para poder implementarlo en FPGA.

Para finalizar con el desarrollo del trabajo, se han conseguido pruebas de laboratorio satisfactorias en las que se demuestra el correcto funcionamiento del sistema completo. Se han realizado simulaciones con diferentes mensajes e intensidades lumínicas para corroborar el funcionamiento del mismo. Se han encontrado algunas dificultades a la hora de sincronizar el sistema completo, pero se han superado con cambios en los distintos módulos.

Por último, quiero destacar que creo que el uso de la luz como medio de transmisión tendrá, no solo que complementar, sino dar un paso adelante en las comunicaciones, sobre todo en interiores. Las ya saturadas redes de radiofrecuencia tienen que reducirse, así como las interferencias electromagnéticas. Es necesario desarrollar aún más los sistemas, sobre todo para sortear algunas limitaciones que vienen implícitas en el uso de lámparas de iluminación. Sin embargo, la necesidad de dar una alternativa a las redes de radiofrecuencia empujará el desarrollo de los sistemas VLC.

Conclusions

Throughout the development of this master’s thesis, the objectives proposed have been achieved. The information transmission system using light has been implemented. In addition,

the optical receiver has been developed for different kinds of modulation and the VOOK modulation system has been implemented in the FPGA board. During the procedures performed, there have been some corrections and modifications, resulting in a more interesting challenge. In conclusion, the implemented system is capable of the transmission of a message through the optical channel, demodulate it and decodify it to recover the original message.

First, the circuit based on a photoreceptor as the main element has been designed and tested successfully. It can convert the received light signal into an interpretable voltage signal for the FPGA.

Secondly, the designed system is capable of first modulate a VOOK signal for transmission, and then demodulate the received signal to recover the original message. This system has been developed in VHDL programming language to be able to work with the FPGA board.

The last steps of this assignment have been the system tests. These laboratory tests have been satisfactory performed, demonstrating the system correct operation. The system simulations have been carried out at different light intensities and with different messages. During the tests, some difficulties have been encountered regarding the whole system synchronizing. However, working on the modules code has been the best way to overcome these.

Finally, I want to emphasize that I consider that the use of light for data transmission will not only complement, but also need to take a step forward in communications, especially indoors. Radio frequency networks already saturated must be relieved to reduce the electromagnetic interferences. VLC systems need to be further developed, especially to overcome some limitations that are implicit in the use of lighting lamps. However, the need to provide an alternative to radio frequency networks will push the development of VLC systems.

Bibliografía.

[1] Areny, Ramón Pallás (2004). Sensores y acondicionadores de señal. Marcombo. 4ª Edición.

[2] Curbelo Meneses, Alejandra; Eiroa Mateo, Laura. Diseño de un sistema de comunicación VPPM para enlaces ópticos inalámbricos en el visible. 2016. Universidad de La Laguna

[3] GROBE, Liane, et al. High-speed visible light communication systems. *IEEE communications magazine*, 2013, vol. 51, no 12, p. 60-66.

[4] IEEE Computer Society, IEEE Standard for Local and metropolitan area networks, Part 15.7: ShortRange Wireless Optical Communication Using Visible Light, 2011

[5] K. Kim, J. Kim and T. Kwon, "Variable intensity-on-off keying modulation and its application to wearable visual-MIMO," *2016 22nd Asia-Pacific Conference on Communications (APCC)*, Yogyakarta, 2016, pp. 400-403

[6] LEE, Kwonhyung; PARK, Hyuncheol. Modulations for visible light communications with dimming control. *IEEE photonics technology letters*, 2011, vol. 23, no 16, p. 1136-1138.

[7] MARIN-GARCIA, Ignacio; GUERRA, Victor; PEREZ-JIMENEZ, Rafael. Study and validation of eavesdropping scenarios over a visible light communication channel. *Sensors*, 2017, vol. 17, no 11, p. 2687.

[8] O'BRIEN, Dominic C., et al. Visible light communications: Challenges and possibilities. En *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2008. p. 1-5.

[9] P. Namonta, P. Cherntanomwong,. Real Time Vital Sign Transmission Using IEEE 802.15.7 VLC PHY-I Transceiver, 2017.

[10] SARBAZI, Elham; UYSAL, Murat. PHY layer performance evaluation of the IEEE 802.15. 7 visible light communication standard. En *2013 2nd International workshop on optical wireless communications (IWOW)*. IEEE, 2013. p. 35-39.

Capítulo 7. Anexos

Índice anexos

6.1. Código del sistema

6.1.1. Estructura principal y generador de frecuencias.

```
----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    TFM_inicial - Behavioral
-- Description:    Sistema de comunicación mediante luz visible, implementado sobre la
--                 placa de desarrollo Nexys A7. El código se divide principalmente en
--                 dos módulos: transmisión y recepción.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity TFM_inicial is
        Port (  percent : in  STD_LOGIC_VECTOR (3 downto 0);   -- Porcentaje iluminación.
                reset : in  STD_LOGIC;     -- Reset del programa
                msg_receptor : in  STD_LOGIC;   -- Mensaje codificado que viene del receptor
                msg_switch : in  STD_LOGIC_VECTOR (3 downto 0); -- Mensaje a enviarde 4 bits
                clk_nexys : in STD_LOGIC;
                 -- Señales para comprobaciones transmisor:
                msg_trans : out  STD_LOGIC;    -- Mensaje que se va a enviar (sin modular)
                msg_trans_cod : out  STD_LOGIC;  -- Mensaje a transmitir, ya codificado
                msg_modul : out STD_LOGIC;     -- Señal hacia la lámpara LED. Mensaje modulado.
                 -- Señales receptor :
                msg4bit_receptor : out STD_LOGIC_VECTOR (3 downto 0);
                msg_demodul : out STD_LOGIC;
                 -- Relojes por pantalla:
                clk_A : out  STD_LOGIC;  -- clk_1 : Reloj principal 66.66 KHz del mensaje
                clk_B : out  STD_LOGIC;  -- clk_2 : Reloj para el mensaje codificado 100 KHz
                clk_C : out  STD_LOGIC); -- clk_3 : Reloj de 1 MHz para la modulación
end TFM_inicial;

architecture Behavioral of TFM_inicial is

-- Instancia para generar los relojes.
        COMPONENT clk_generator PORT(
```

```
            clk_1 : out STD_LOGIC;   -- Reloj principal 66.66 KHz del mensaje.
            clk_2 : out STD_LOGIC;   -- Reloj para el mensaje codificado 100 KHz
            clk_3 : out STD_LOGIC;   -- Reloj de 1 MHz para la modulación
            clk_nexys : in STD_LOGIC; -- Reloj propio de la placa de desarrollo.
            clk_sampling : out STD_LOGIC;  -- Reloj para el muestreo de la señal. 5 MHz.
            reset : in STD_LOGIC
        );
        END COMPONENT;

-- Instancia del transmisor. Se codifica y modula el mensaje a enviar.
        COMPONENT Transmitter PORT(
            clk_1 : in STD_LOGIC;   -- Reloj principal 66.66 KHz del mensaje.
            clk_2 : in STD_LOGIC;   -- Reloj para el mensaje codificado 100 KHz.
            clk_3 : in STD_LOGIC;   -- Reloj de 1 MHz para la modulación.
            percent : in  STD_LOGIC_VECTOR (3 downto 0);  -- Porcentaje de iluminación elegido.
            msg_trans_cod : out STD_LOGIC;   -- Mensaje a transmitir codificado (6 bits).
            msg_trans : out STD_LOGIC;       -- Mensaje a transmitir (4 bits).
            modulated_msg : out STD_LOGIC;   -- Mensaje codificado y señal modulada.
            message : in STD_LOGIC_VECTOR (3 downto 0); -- Mensaje introducido mediante switches.
            reset : in STD_LOGIC
        );
        END COMPONENT;

-- Instancia del receptor. Se recibe, muestrea, demodula y decodifica la señal para obtener el mensaje.
        COMPONENT Receiver PORT(
            clk_1 : in STD_LOGIC;   -- Reloj principal 66.66 KHz del mensaje.
            clk_2 : in STD_LOGIC;   -- Reloj para el mensaje codificado 100 KHz.
            clk_3 : in STD_LOGIC;   -- Reloj de 1 MHz para la modulación.
            clk_samp : in STD_LOGIC;  -- Reloj para muestreo. (50 muestras por bit)
            percent : in  STD_LOGIC_VECTOR (3 downto 0);  -- Porcentaje de iluminación elegido.
            msg_rec_cod : out STD_LOGIC;   -- Mensaje recibido codificado (6 bits).
            msg_rec : out STD_LOGIC;       -- Mensaje recibido (4 bits).
            modulated_msg : in STD_LOGIC;   -- Mensaje a la entrada (codificado y modulado).
            rec_msg4b : out STD_LOGIC_VECTOR (3 downto 0); -- Mensaje demodulado y
descodificado.
            reset : in STD_LOGIC
        );
        END COMPONENT;

        signal clk_msg : STD_LOGIC;                -- clk_1
```

```
    signal clk_cod : STD_LOGIC;                 -- clk_2
    signal clk_perc_mod : STD_LOGIC;            -- clk_3
    signal sampling_clk : STD_LOGIC;                    -- clk_sampling
    signal message_RCod : STD_LOGIC;
    signal message_R : STD_LOGIC;
-------------------------------------------------------------------------------------------------------------------------------------
begin
---------- Instancia para generar los relojes.
    Inst_clk_generator: clk_generator port map(
            clk_1 => clk_msg,
            clk_2 => clk_cod,
            clk_3 => clk_perc_mod,
            clk_nexys => clk_nexys,
            clk_sampling => sampling_clk,
            reset => reset
    );
-------------------------------------------------------------------------------------------------------------------------------------
---------- Para mostrar los relojes en las simulaciones y en osciloscopio para comprobaciones.
    clk_A <= clk_msg;
    clk_B <= clk_cod;
    clk_c <= clk_perc_mod;
-------------------------------------------------------------------------------------------------------------------------------------
---------- Instancia del transmisor. Se codifica y modula el mensaje a enviar.
    Inst_Transmitter : Transmitter port map(
            clk_1 => clk_msg,
            clk_2 => clk_cod,
            clk_3 => clk_perc_mod,
            percent => percent,
            msg_trans_cod => msg_trans_cod,
            msg_trans => msg_trans,
            modulated_msg => msg_modul,
            message => msg_switch,
            reset => reset
    );
-------------------------------------------------------------------------------------------------------------------------------------
---------- Instancia del receptor. Se recibe, muestrea, demodula y decodifica la señal para obtener el mensaje.
    Inst_Receiver : Receiver port map(
            clk_1 => clk_msg,
            clk_2 => clk_cod,
            clk_3 => clk_perc_mod,
            clk_samp => sampling_clk,
```

```
                percent => percent,
                msg_rec_cod => message_RCod,
                msg_rec => message_R,
                modulated_msg => msg_receptor,
                rec_msg4b => msg4bit_receptor,    -- Mensaje recibido, demodulado y descodificado (4 bits)
                reset => reset
        );
        msg_demodul <= message_R;        -- Mensaje final (en forma de señal para ver en pantalla)

end Behavioral;


----------------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    clk_generator - Behavioral
-- Description:     Generación de diferentes frecuencias utilizadas en el sistema.
----------------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clk_generator is
        Port (  clk_1 : out STD_LOGIC;          -- Reloj principal 66.66 KHz del mensaje.
                clk_2 : out STD_LOGIC;       -- Reloj para el mensaje codificado 100 KHz
                clk_3 : out STD_LOGIC;       -- Reloj para generar la señal VOOK - 1 MHz.
                clk_nexys : in  STD_LOGIC;
                clk_sampling :out STD_LOGIC;     -- Reloj para el muestreo de la señal de entrada. 5 MHz
                reset : in  STD_LOGIC);
end clk_generator;

architecture Behavioral of clk_generator is
   constant period1 : integer := 750;     -- Cuenta para reloj de 66.66 KHz
   constant period2 : integer := 500;     -- Cuenta para reloj de 100 KHz
   constant period3 : integer := 50;      -- Cuenta para reloj de 1 MHz
   constant period_samp : integer := 10;  -- Cuenta para reloj de 5 MHz

   signal clk_1_temp : STD_LOGIC := '1';      -- Variable temporal reloj 1
   signal clk_2_temp : STD_LOGIC := '1';      -- Variable temporal reloj 2
   signal clk_3_temp : STD_LOGIC := '1';      -- Variable temporal reloj 3
   signal clk_samp_temp : STD_LOGIC := '1';   -- Variable temporal reloj muestreo
```

```
    signal clk_1_count : integer range 0 to period1 := 0;
    signal clk_2_count : integer range 0 to period2 := 0;
    signal clk_3_count : integer range 0 to period3 := 0;
    signal clk_samp_count : integer range 0 to period_samp := 0;


begin
-- Proceso de generador de frecuencias mediante cuentas.
frequency_generator: process (reset, clk_nexys)
        begin
        if reset = '1' then     -- con el pulso del reset, se reinician las señales a valores iniciales.
                clk_1_count <= 0;
                clk_1_temp <= '1';
                clk_2_count <= 0;
                clk_2_temp <= '1';
                clk_3_count <= 0;
                clk_3_temp <= '1';
                clk_samp_count <= 0;
                clk_samp_temp <= '1';
        elsif rising_edge(clk_nexys) then   -- Con los flancos de subidas del reloj se suma a las cuentas.
-- A medida que se alcancen objetivos se invierte la señal del reloj, resultando en las diferentes frecuencias       --
elegidas (66.66 kHz, 100 kHz, 1 MHz, 5 MHz)
                if (clk_1_count = period1/2 - 1) then
                        clk_1_temp <= NOT(clk_1_temp);
                        clk_1_count <= 0;
                else
                        clk_1_count <= clk_1_count + 1;
                end if;
                if (clk_2_count = period2/2 - 1) then
                        clk_2_temp <= NOT(clk_2_temp);
                        clk_2_count <= 0;
                else
                        clk_2_count <= clk_2_count + 1;
                end if;
                if (clk_3_count = period3/2 - 1) then
                        clk_3_temp <= NOT(clk_3_temp);
                        clk_3_count <= 0;
                else
                        clk_3_count <= clk_3_count + 1;
                end if;
```

```
                if (clk_samp_count = period_samp/2 - 1) then
                        clk_samp_temp <= NOT(clk_samp_temp);
                        clk_samp_count <= 0;
                else
                        clk_samp_count <= clk_samp_count + 1;
                end if;
        end if;
        end process;
    -- Traspaso de variables temporales a la salida.
    clk_1 <= clk_1_temp;
    clk_2 <= clk_2_temp;
    clk_3 <= clk_3_temp;
    clk_sampling <= clk_samp_temp;
end Behavioral;
```

6.1.2. Código del modulo transmisor

```
----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Transmitter - Behavioral
-- Description:    Instancia para la transmisión del mensaje modulado en VOOK.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Transmitter is
        Port (  clk_1 : in  STD_LOGIC;
                clk_2 : in  STD_LOGIC;
                clk_3 : in  STD_LOGIC;
                percent : in  STD_LOGIC_VECTOR (3 downto 0);
                msg_trans_cod : out  STD_LOGIC;          -- Mensaje codificado en 4b6b
                msg_trans : out  STD_LOGIC;      -- Mensaje sin codificar (señal)
                modulated_msg : out  STD_LOGIC;          -- Mensaje modulado VOOK para transmitir.
                message : in STD_LOGIC_VECTOR (3 downto 0); -- Mensaje a transmitir
                reset : in  STD_LOGIC);
end Transmitter;

architecture Behavioral of Transmitter is
```

```
        Signal msg_4bits : STD_LOGIC := '0';   -- Señal temporal del mensaje de 4 bits.
        Signal msg_6bits : STD_LOGIC := '0';   -- Señal temporal del mensaje de 6 bits.
        Signal message_cod : STD_LOGIC_VECTOR (5 downto 0) := "ZZZZZZ"; -- Mensaje codificado 6bits.
        signal send_msg : STD_LOGIC := '0';   -- Flag inicial para la transmisión.
        Signal msg_sent : STD_LOGIC := '0';   -- Señal para indicar que ha terminado de enviar un mensaje.
        Signal Flag : STD_LOGIC := '0';       -- Flag para indicar que se ha completado la transmisión.
        Signal msg_hold : STD_LOGIC_VECTOR (5 downto 0) := "ZZZZZZ"; -- Mensaje codificado.


-- Instancia para generar una señal con el mensaje de 4 bits.
        COMPONENT Message_generator PORT(
                clk_1 : in STD_LOGIC;                   -- Reloj 66.66 kHz.
                msg : in STD_LOGIC_VECTOR (3 downto 0); -- Mensaje de 4 bits de entrada.
                msg_4B : out STD_LOGIC;          -- Señal representando el mensaje para comprobaciones.
                reset : in STD_LOGIC
        );
        END COMPONENT;


-- Instancia para conversión del mensaje de 4 bits a 6 bits y generación de señal para comprobaciones.
        COMPONENT Message_4B6B PORT(
                clk_2 : in STD_LOGIC;                   -- Reloj 100 kHz.
                msg_in : in STD_LOGIC_VECTOR (3 downto 0);  -- Mensaje de 4 bits de entrada.
                msg_6B : out STD_LOGIC;                 -- Mensaje de 6 bits (codificado).
                msg_codif : out STD_LOGIC_VECTOR (5 downto 0);       -- Señal representa mensaje
codificado.
                reset : in STD_LOGIC
        );
        END COMPONENT;
-- Instancia para la modulación del mensaje a transmitir VOOK.
        COMPONENT Message_modulation PORT(
                msg_6bit_cod : in STD_LOGIC_VECTOR (5 downto 0); -- Mensaje de 6 bits (codificado).
                message_cod : in STD_LOGIC;      -- Señal representa mensaje codificado.
                reset : in STD_LOGIC;
                clk_3 : in STD_LOGIC;            -- Reloj de 1 MHz.
                clk_2 : in STD_LOGIC;            -- Reloj de 100 kHz.
                percent : in STD_LOGIC_VECTOR (3 downto 0); -- Porcentaje intensidad lumínica.
                trans_ok : out STD_LOGIC;        -- Flag indicando transmisión del mensaje finalizada.
                msg_modulated : out STD_LOGIC            -- Mensaje modulado hacia el LED.
        );
        END COMPONENT;
```

```
begin

------------------------------------------------------------------------------------------------------------------------
---------- Instancia para generar una señal con el mensaje de 4 bits.
        Init_Message_generator: Message_generator port map(
                clk_1 => clk_1,     -- 66.66KHz
                msg => message,
                msg_4B => msg_4bits,
                reset => reset
        );
------------------------------------------------------------------------------------------------------------------------
---------- Instancia para conversión del mensaje de 4 bits a 6 bits y generación de señal para comprobaciones.
        Init_Message_4B6B : Message_4B6B port map(
                clk_2 => clk_2,     -- 100KHz
                msg_in => message,
                msg_6B => msg_6bits,
                msg_codif => message_cod,
                reset => reset
        );
------------------------------------------------------------------------------------------------------------------------
---------- Asignación de señales
        msg_trans <= msg_4bits;
        msg_trans_cod <= msg_6bits;
------------------------------------------------------------------------------------------------------------------------
---------- Instancia para la modulación del mensaje a transmitir VOOK.
        Init_Message_modulation : Message_modulation port map(
                msg_6bit_cod => msg_hold,
                message_cod => msg_6bits,
                reset => reset,
                clk_3 => clk_3,    -- 1 MHz
                clk_2 => clk_2,    -- 100 KHz
                percent => percent,
                trans_ok => msg_sent,  -- señal de que se ha enviado el mensaje
                msg_modulated => modulated_msg
        );
------------------------------------------------------------------------------------------------------------------------
-- En este proceso se asegura que el valor del mensaje de 6 bits, que se pasa a la instancia de modulación, no -- se modifica hasta que se realiza la modulación completa del mensaje.
        process (msg_sent, clk_1, reset)
        begin
                if falling_edge(msg_sent) then
                        msg_hold <= message_cod;
                elsif msg_sent = 'Z' then
```

```
                msg_hold <= message_cod;
        end if;
        if reset = '1' then
                Flag <= '1';
                msg_hold <= "ZZZZZZ";
        elsif reset = '0' and Flag = '1'  and clk_1 = '0' then
                msg_hold <= message_cod;
                Flag <= '0';
        end if;
        if falling_edge(clk_1) then
                        if send_msg ='0' then  -- Señal inicialmente a 0 para el primer mensaje.
                                msg_hold <= message_cod;
-- Una vez realizada su función, send_msg se deja a '1' hasta el reinicio de la FPGA.
                                send_msg <= '1';
                        end if;
        end if;
    end process;

end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Message_generator - Behavioral
-- Description:    Instancia para generar una señal con el mensaje de 4 bits a transmitir.
--                 Señal para comprobaciones.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Message_generator is
    Port (  clk_1 : in  STD_LOGIC;
            msg : in  STD_LOGIC_VECTOR (3 downto 0);
            msg_4B : out  STD_LOGIC;
            reset : in  STD_LOGIC);
end Message_generator;

architecture Behavioral of Message_generator is
begin
```

```
-- Proceso para generar la señal del mensaje 4bits para comprobaciones.
-- Se utiliza el reloj de 66.66 kHz para la señal.
	process(clk_1,reset)
		variable i : integer := 0;
		variable clk_count : integer := 0;
	begin
		if clk_count = 3 then
			if reset = '1' then
				i := 0;
				clk_count := 0;
				msg_4B <= '0';
			elsif rising_edge(clk_1) then
				msg_4B <= msg(3-i);
				i := i + 1;
			end if;
			if i = 4 then
				i := 0;
			end if;
		elsif reset = '1' then
			i := 0;
			clk_count := 0;
		elsif rising_edge(clk_1) then
			clk_count := clk_count + 1;
		end if;
	end process;

end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Message_4B6B - Behavioral
-- Description:    Instancia para generar una señal con el mensaje de 4 bits a transmitir.
--                 Señal para comprobaciones.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Message_4B6B is
```

```
        Port (    clk_2 : in  STD_LOGIC;
                  msg_in : in  STD_LOGIC_VECTOR (3 downto 0);
                  msg_6B : out  STD_LOGIC;
                  msg_codif : out  STD_LOGIC_VECTOR (5 downto 0);
                  reset : in  STD_LOGIC);
end Message_4B6B;

architecture Behavioral of Message_4B6B is
-- Señal temporal para trabajar en los procesos.
        Signal signal_6Bits : STD_LOGIC_VECTOR (5 downto 0) := "000000";
-- Tabla para seleecionar el vector de 6 bits en base al mensaje de 4 bits.
        COMPONENT Table_4B6B PORT(
                  msg_in : in STD_LOGIC_VECTOR (3 downto 0);   -- Mensaje de 4 bits de entrada
                  msg_out : out STD_LOGIC_VECTOR (5 downto 0); -- Mensaje de 6 bits de salida
                  reset : in STD_LOGIC
        );
        END COMPONENT;

begin
--------------------------------------------------------------------------------
-- Tabla para seleecionar el vector de 6 bits en base al mensaje de 4 bits.
        Init_Table_4B6B: Table_4B6B port map(
                  msg_in => msg_in,
                  msg_out => signal_6Bits,
                  reset => reset
        );

        msg_codif <= signal_6Bits;
--------------------------------------------------------------------------------
-- Proceso que genera la señal del mensaje codificado para mostrar por pantalla.
-- Se genera un retardo para el envío del mensaje.
-- Se utiliza el reloj de 100 kHz.
        process(clk_2,reset)
                  variable i : integer := 0;
                  variable clk_count : integer := 0;
        begin
                  if clk_count = 5 then        -- Retardo en la generación y codificación del mensaje 4b6b.
                            if reset = '1' then
                                      i := 0;
                                      clk_count := 0;
```

```
                msg_6B <= '0';
            elsif rising_edge(clk_2) then
                msg_6B <= signal_6Bits(5-i);
                i := i + 1;
            end if;
            if i = 6 then
                i := 0;
            end if;
        elsif reset = '1' then
            i := 0;
            clk_count := 0;
        elsif rising_edge(clk_2) then
            clk_count := clk_count + 1;
        end if;
    end process;

end Behavioral;

----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Table_4B6B - Behavioral
-- Description:    Instancia para seleccionar el mensaje codificado 4b6b en función del
--                 mensaje de 4 bits de entrada.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Table_4B6B is
   Port ( msg_in : in  STD_LOGIC_VECTOR (3 downto 0);
          msg_out : out  STD_LOGIC_VECTOR (5 downto 0);
          reset : in  STD_LOGIC);
end Table_4B6B;

architecture Behavioral of Table_4B6B is
begin
    process (msg_in, reset)
        begin
        if reset = '1' then
```

```
                msg_out <= "ZZZZZZ";
        else
                case msg_in is
                        when "0000" =>  -- 0
                                msg_out <= "001110";
                        when "0001" =>  -- 1
                                msg_out <= "001101";
                        when "0010" =>  -- 2
                                msg_out <= "010011";
                        when "0011" =>  -- 3
                                msg_out <= "010110";
                        when "0100" =>  -- 4
                                msg_out <= "010101";
                        when "0101" =>  -- 5
                                msg_out <= "100011";
                        when "0110" =>  -- 6
                                msg_out <= "100110";
                        when "0111" =>  -- 7
                                msg_out <= "100101";
                        when "1000" =>  -- 8
                                msg_out <= "011001";
                        when "1001" =>  -- 9
                                msg_out <= "011010";
                        when "1010" =>  -- A
                                msg_out <= "011100";
                        when "1011" =>  -- B
                                msg_out <= "110001";
                        when "1100" =>  -- C
                                msg_out <= "110010";
                        when "1101" =>  -- D
                                msg_out <= "101001";
                        when "1110" =>  -- E
                                msg_out <= "101010";
                        when "1111" =>  --F
                                msg_out <= "101100";
                        when others =>  -- Casos no identificados
                                msg_out <= "ZZZZZZ";
                end case;
        end if;
end process;
```

```
end Behavioral;

----------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Message_modulation - Behavioral
-- Description:     Instancia para modular el mensaje con VOOK.
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Message_modulation is
        Port (    msg_6bit_cod : in  STD_LOGIC_VECTOR (5 downto 0);  -- Vector de 6 bits con el mensaje.
                  message_cod : in STD_LOGIC;
                  reset : in  STD_LOGIC;
                  clk_3 : in  STD_LOGIC;
                  clk_2 : in STD_LOGIC;
                  percent : in  STD_LOGIC_VECTOR (3 downto 0);
                  trans_ok : out STD_LOGIC;
                  msg_modulated : out  STD_LOGIC);
end Message_modulation;

architecture Behavioral of Message_modulation is
-- Instancia para generar unos valores para ceros y unos dependiendo de la intensidad elegida.
-- En la tabla de los valores para ceros y unos depende de la señal de entrada percent.
        COMPONENT Percentage_0_1 PORT(
                  percent : in STD_LOGIC_VECTOR (3 downto 0);
                  Zero_value : out STD_LOGIC_VECTOR (3 downto 0);       -- Cuentas de ceros.
                  One_value : out STD_LOGIC_VECTOR (3 downto 0)         -- Cuentas de unos.
        );
        END COMPONENT;
-- Instancia para la modulación de la señal a transmitir con VOOK.
        COMPONENT Symbol_modulation PORT(
                  Zero_value : in STD_LOGIC_VECTOR (3 downto 0); -- Vector que alberga las cuentas del
cero.
                  One_value : in STD_LOGIC_VECTOR (3 downto 0);  -- Vector que alberga las cuentas del
uno.
                  reset : in STD_LOGIC;
                  symbol : in STD_LOGIC;                                -- bit que va a modular el proceso.
```

```
            cod6B : in STD_LOGIC_VECTOR (5 downto 0); -- Vector con el mensaje de 6 bits.
            clk_3 : in STD_LOGIC;                     -- Reloj de 1 MHz.
            trans_ok : out STD_LOGIC;                 -- Flag que indica el fin de la transmisión.
            modul_sym : out STD_LOGIC                 -- Señal del mensaje modulado.
        );
        END COMPONENT;
-- Señales auxiliares para la modulación.
        signal Zero : STD_LOGIC_VECTOR (3 downto 0) := "ZZZZ";
        signal One : STD_LOGIC_VECTOR (3 downto 0) := "ZZZZ";
        signal msg_symbol : STD_LOGIC := '0';

begin
-- En este proceso se modifica el bit que va a modular este módulo.
-- Se modifica con el reloj de 100 kHz. Para comprobaciones.
        process (clk_2, reset, msg_6bit_cod)
                variable i : integer := 0;
        begin
            if reset = '1' then
                msg_symbol <= '0';
                i := 0;
            elsif msg_6bit_cod = "000000" then
                msg_symbol <= '0';
                i := 0;
            elsif i = 5 then
                i := 0;
            elsif rising_edge(clk_2) then
                i := i+1;
            end if;

            msg_symbol <= msg_6bit_cod(5-i);

        end process;
-------------------------------------------------------------------------------------------------------------------
---------- Instancia para generar unos valores para ceros y unos dependiendo de la intensidad elegida.
-- En la tabla de los valores para ceros y unos depende de la señal de entrada percent.
        Init_Percentage_0_1 : Percentage_0_1 Port map(
            percent => percent,
            Zero_value => Zero,
            One_value => One
        );
```

```
----------------------------------------------------------------------------------------------------------------------
---------- Instancia para la modulación de la señal a transmitir con VOOK.
        Init_Symbol_modulation: Symbol_modulation port map(
                Zero_value => Zero,
                One_value => One,
                reset => reset,
                symbol => msg_symbol,
                cod6B => msg_6bit_cod,
                clk_3 => clk_3,
                trans_ok => trans_ok,
                modul_sym => msg_modulated
        );

end Behavioral;


----------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Percentage_0_1 - Behavioral
-- Description:    Tabla para seleccionar unos valores de cuentas para la modulación
--                 tanto de ceros, como de unos en el mensaje.
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Percentage_0_1 is
   Port ( percent : in  STD_LOGIC_VECTOR (3 downto 0);
          Zero_value : out  STD_LOGIC_VECTOR (3 downto 0);
          One_value : out  STD_LOGIC_VECTOR (3 downto 0));
end Percentage_0_1;

architecture Behavioral of Percentage_0_1 is
begin
        process (percent)
        begin
                case percent is
                        when "0000" =>          --      10%
                                Zero_value <= "1010";  -- 10/10 porcentaje de apagado.
                                One_value <= "0010";  -- 2/10  porcentaje de encendido.
                        when "0001" =>          --      20%
```

```
                    Zero_value <= "1010"; -- 10/10
                    One_value <= "0100";  -- 4/10
                when "0010" =>              --          30%
                    Zero_value <= "1010"; -- 10/10
                    One_value <= "0110";  -- 6/10
                when "0011" =>              --          40%
                    Zero_value <= "1010"; -- 10/10
                    One_value <= "1000";  -- 8/10
                when "0100" =>              --          50%
                    Zero_value <= "1010"; -- 10/10
                    One_value <= "1010";  -- 10/10
                when "0101" =>              --          60%
                    Zero_value <= "1000"; -- 8/10
                    One_value <= "1010";  -- 10/10
                when "0110" =>              --          70%
                    Zero_value <= "0110"; -- 6/10
                    One_value <= "1010"; -- 10/10
                when "0111" =>              --          80%
                    Zero_value <= "0100";  -- 4/10
                    One_value <= "1010";  -- 10/10
                when "1000" =>              --          90%
                    Zero_value <= "0010"; -- 2/10
                    One_value <= "1010";  -- 10/10
                when others =>  -- Cualquier otro valor por encima de 90% se satura al 90%
                    Zero_value <= "0010";  -- 2/10
                    One_value <= "1010";  -- 10/10
                end case;
        end process;
end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Symbol_modulation - Behavioral
-- Description:     Instancia para modular el mensaje con VOOK a transmitir.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;

entity Symbol_modulation is
        Port (  Zero_value : in  STD_LOGIC_VECTOR (3 downto 0);
                One_value : in  STD_LOGIC_VECTOR (3 downto 0);
                reset : in  STD_LOGIC;
                symbol : in  STD_LOGIC;
                cod6B : in STD_LOGIC_VECTOR (5 downto 0);
                clk_3 : in  STD_LOGIC;
                trans_ok : out STD_LOGIC;
                modul_sym : out  STD_LOGIC);
end Symbol_modulation;

architecture Behavioral of Symbol_modulation is

-- Señales para trabajar en el proceso de modulación
        signal position : STD_LOGIC_VECTOR (3 downto 0) := "0000";
        signal bit_out : STD_LOGIC := '0';
        signal confirm : STD_LOGIC := '0';

begin
-- En este proceso único se realiza la modulación de la señal a transmitir.
-- Se utiliza el vector de 6 bits del mensaje codificado para modular los diferentes bits.
-- La modulación se realiza mediante cuentas tanto para ceros como para unos.
-- Se trabaja con el reloj de 1 MHz.
        process (reset, clk_3, cod6B)
                variable i : integer := 0;
                variable clk_count : integer := 0;
                variable pos : std_logic_vector (3 downto 0) := "0000";

        begin
                if clk_count = 59 then   -- Cuenta para sincronizar señales (aspecto visual)
                        if reset = '1' then
                                modul_sym <= '0';
                                i := 0;
                                clk_count := 0;
                                pos := "0000";
                        elsif cod6B = "000000" then
                                modul_sym <= '0';
                                i := 0;
```

```
                pos := "0000";
        elsif rising_edge(clk_3) then
                if cod6B(i) = '0' then
                        if (pos < Zero_value) then
                                modul_sym <= '0';
                        elsif (pos >= Zero_value) then
                                modul_sym <= '1';
                        end if;
                elsif cod6B(i) = '1' then
                        if (pos < One_value) then
                                modul_sym <= '1';
                        elsif (pos >= One_value) then
                                modul_sym <= '0';
                        end if;
                end if;
                if pos = "1001" then
                        if i = 5 then
                                i := 0;
                                trans_ok <= '0';
                        elsif i < 5 then
                                i := i + 1;
                                if i = 4 then
                                        trans_ok <= '1'; -- Señal de mensaje transmitido.
                                end if;
                        end if;
                        pos := "0000";
                elsif pos < "1001" then
                        pos := pos + '1';
                end if;
        end if;
elsif reset = '1' then
        i := 0;
        pos := "0000";
        clk_count := 0;
        modul_sym <= '0';
elsif rising_edge(clk_3) then
        clk_count := clk_count + 1;
        if clk_count = 50 then
                modul_sym <= '1';
        end if;
```

```
                    if clk_count = 59 then
                        modul_sym <= '0';
                    end if;
                end if;
                position <= pos;
                bit_out <= cod6B(5-i);   -- Señal de comprobación
        end process;
end Behavioral;
```

6.1.3. Código del modulo receptor

```
----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Receiver - Behavioral
-- Description:    Instancia del receptor del sistema de transmisión.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Receiver is
        Port (  clk_1 : in  STD_LOGIC;
                clk_2 : in  STD_LOGIC;
                clk_3 : in  STD_LOGIC;
                clk_samp : in  STD_LOGIC;
                percent : in  STD_LOGIC_VECTOR (3 downto 0);
                msg_rec_cod : out  STD_LOGIC;           -- Mensaje demodulado (comprobación)
                msg_rec : out  STD_LOGIC;               -- Mensaje descodificado (comprobación)
                modulated_msg : in  STD_LOGIC;                  -- Entrada de la señal recibida.
                rec_msg4b : out  STD_LOGIC_VECTOR (3 downto 0); -- Mensaje de 4 bits final
                reset : in  STD_LOGIC);
end Receiver;
architecture Behavioral of Receiver is
-- Instancia para generar una señal igual a la señal de entrada.
        COMPONENT Signal_filter PORT(
                reset : in STD_LOGIC;
                modulated_msg : in STD_LOGIC; -- Señal de entrada desde el receptor físico.
                input_signal : out STD_LOGIC    -- Señal generada para el demodulador.
        );
```

```
        END COMPONENT;
-- Instancia para generar una habilitación que permita comenzar con la lectura/muestreo.
        COMPONENT Enable_bit PORT(
                input_signal : in STD_LOGIC;      -- Señal de entrada, del módulo Signal_filter.
                enable : out STD_LOGIC;                   -- Variable habilitadora para el muestreo de la señal.
                reset : in STD_LOGIC
        );
        END COMPONENT;
-- Instancia para la demodulación de la señal de entrada.
        COMPONENT Demodulation PORT(
                msg_input : in STD_LOGIC;         -- Señal de entrada, del módulo Signal_filter.
                reset : in STD_LOGIC;
                enable : in STD_LOGIC;            -- Variable habilitadora para el muestreo de la señal.
                clk_3 : in STD_LOGIC;             -- 1 MHz
                clk_2 : in STD_LOGIC;             -- 100 kHz
                clk_1 : in STD_LOGIC;             -- 66.66 kHz
                clk_samp : in STD_LOGIC;                  -- 5 MHz
                fin_signal : out STD_LOGIC;       -- Señal de salida, mensaje demodulado y descodificado
                fin_cod_signal : out STD_LOGIC;-- Señal de salida con el mensaje demodulado.
                final_msg : out STD_LOGIC_VECTOR (3 downto 0) -- Mensaje demodulado y descodificado.
         );
        END COMPONENT;
-- Señales auxiliares para la recepción.
        signal b_enable : STD_LOGIC := '0';
        signal rec_input : STD_LOGIC := '0';

begin
-------------------------------------------------------------------------------------------------------------------------------
-- Instancia para generar una señal igual a la señal de entrada.
        Init_Signal_filter: Signal_filter port map(
                reset => reset,
                modulated_msg => modulated_msg,
                input_signal => rec_input
        );
-------------------------------------------------------------------------------------------------------------------------------
-- Instancia para generar una habilitación que permita comenzar con la lectura/muestreo.
         Init_Enable_bit : Enable_bit port map(
                input_signal => rec_input,
                enable => b_enable,
                reset => reset
```

```
        );
-------------------------------------------------------------------------------------------------------------------
--------
-- Instancia para la demodulación de la señal de entrada.
        Init_Demodulation : Demodulation port map(
                msg_input => rec_input,
                reset => reset,
                enable => b_enable,
                clk_3 => clk_3,
                clk_2 => clk_2,
                clk_1 => clk_1,
                clk_samp => clk_samp,
                fin_signal => msg_rec,
                fin_cod_signal => msg_rec_cod,
                final_msg => rec_msg4b
        );

end Behavioral;


-------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Signal_filter - Behavioral
-- Description:     Recepción de la señal y generación de una señal interna que no se vea
--                  afectada por pequeños armónicos de ruido en la señal recibida.
-------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Signal_filter is
        Port (  reset : in  STD_LOGIC;
                modulated_msg : in  STD_LOGIC;
                input_signal : out  STD_LOGIC);
end Signal_filter;
architecture Behavioral of Signal_filter is
begin
-- En este proceso se genera una señal que se pone a uno o a cero en función
-- de la entrada desde el receptor.
        process (modulated_msg, reset)
```

```
        begin
                if reset = '1'then
                        input_signal <= '0';
                elsif modulated_msg ='1' then
                        input_signal <= '1';
                elsif modulated_msg = '0' then
                        input_signal <= '0';
                end if;
        end process;
end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Enable_bit - Behavioral
-- Description:     Generación de una variable habilitadora que permita el comienzo del
--                  muestreo de la señal de entrada.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Enable_bit is
        Port (  input_signal : in  STD_LOGIC;
                enable : out  STD_LOGIC;
                reset : in  STD_LOGIC);
end Enable_bit;
architecture Behavioral of Enable_bit is
-- Señales auxiliares
        signal inter_enable : STD_LOGIC := '1';
        signal en : STD_LOGIC := '0';
begin
-- El bit de enable se pone a uno un ciclo de reloj (1 MHz) antes de que comience a llegar el mensaje.
-- Se presenta una sincronización entre esta variable y el demodulador para comenzar el muestreo en el momento
-- adecuado.
        process (input_signal, reset)
        begin
                if reset = '1' then
                        en <= '0';
                elsif en = '0' and inter_enable = '0' then
```

```
                if rising_edge(input_signal) then
                        inter_enable <= '1';
                end if;
        elsif en = '0' and inter_enable = '1' then
                if falling_edge(input_signal) then
                        en <= '1';
                end if;
        elsif en = '1' then
                -- do nothing
        end if;
    end process;
    enable <= en;
end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Demodulation - Behavioral
-- Description:    Instancia para la demodulación de la señal recibida.
---------------------------------------------------------------------------------- ----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity Demodulation is
    Port (  msg_input : in  STD_LOGIC;
            reset : in  STD_LOGIC;
            enable : in  STD_LOGIC;
            clk_3 : in  STD_LOGIC;
            clk_2 : in  STD_LOGIC;
            clk_1 : in  STD_LOGIC;
            clk_samp : in  STD_LOGIC;
            ---------- Señales para comprobaciones.
            fin_signal : out  STD_LOGIC;
            fin_cod_signal : out  STD_LOGIC;
            ---------- Mensaje final demodulado y descodificado.
            final_msg : out  STD_LOGIC_VECTOR (3 downto 0));
end Demodulation;
```

```
architecture Behavioral of Demodulation is
-- Instancia para realizar el muestreo de la señal y realizar la correlación para establecer el
-- valor de la señal en curso.
	COMPONENT Correlation PORT(
		reset : in STD_LOGIC;
		enable : in STD_LOGIC;		-- Señal habilitadora para comenzar el muestreo
		clk_3 : in STD_LOGIC;		-- 1 MHz
		clk_2 : in STD_LOGIC;		-- 100 kHz
		clk_samp : in STD_LOGIC;			-- 5 MHz
		input : in STD_LOGIC;		-- Señal de entrada
		dec_enable : out STD_LOGIC;	-- Señal para habilitar la selección del bit.
		bit_position : out STD_LOGIC_VECTOR (2 downto 0); -- Posición del bit actual en el mensaje.
		test_signal : out STD_LOGIC_VECTOR (5 downto 0); -- Variable para pruebas
		symbol_vector : out STD_LOGIC_VECTOR  (4 downto 0)  -- Vector_5  valor del bit.
	);
	END COMPONENT;
-- Instancia donde se compara el vector de cinco componentes que representan el bit muestreado
-- del mensaje para obtener el valor del bit actual y rellenar el vector del mensaje codificado.
	COMPONENT Decision PORT(
		reset : in STD_LOGIC;
		clk_2 : in STD_LOGIC;
		symbol_input : in STD_LOGIC_VECTOR (4 downto 0);
		dec_enable : in STD_LOGIC;
		bit_position : in STD_LOGIC_VECTOR (2 downto 0);
		msg_6b_out : out STD_LOGIC_VECTOR (5 downto 0);
		msg_enable : out STD_LOGIC
	);
	END COMPONENT;
-- Instancia para descodificar el mensaje usando una tabla.
	COMPONENT Message_decod PORT(
		reset : in STD_LOGIC;
		msg6b_input : in STD_LOGIC_VECTOR (5 downto 0);
		enable_decod : in STD_LOGIC;
		msg_4b_out : out STD_LOGIC_VECTOR (3 downto 0)
	);
	END COMPONENT;
-- Señales auxiliares durante la demodulación
	 Signal symbol_v : STD_LOGIC_VECTOR (4 downto 0);
	 Signal msg_codif : STD_LOGIC_VECTOR (5 downto 0);
```

```
    Signal enable_dec : STD_LOGIC;
    Signal enable_msg : STD_LOGIC;
    Signal Position : STD_LOGIC_VECTOR (2 downto 0) := "000";
    signal final_msg_temp : STD_LOGIC_VECTOR (3 downto 0) := "0000";
   Signal enable_msg : STD_LOGIC := '0';

begin
-------------------------------------------------------------------------------------------------------------------
-- Instancia para realizar el muestreo de la señal y realizar la correlación para establecer el
-- valor de la señal en curso.
    Init_Correlation : Correlation port map(
            reset => reset,
            enable => enable,
            clk_3 => clk_3,
            clk_2 => clk_2,
            clk_samp => clk_samp,
            input => msg_input,
            dec_enable => enable_dec,
            bit_position => position,
            test_signal => msg_prueba,
            symbol_vector => symbol_v
    );
-------------------------------------------------------------------------------------------------------------------
-- Instancia donde se compara el vector de cinco componentes que representan el bit muestreado
-- del mensaje para obtener el valor del bit actual y rellenar el vector del mensaje codificado.
    Init_Decision : Decision port map(
            reset => reset,
            clk_2 => clk_2,
            symbol_input => symbol_v,
            dec_enable => enable_dec,
            bit_position => position,
            msg_6b_out => msg_codif,
            msg_enable => enable_msg
    );

-------------------------------------------------------------------------------------------------------------------
-- Instancia para descodificar el mensaje usando una tabla.
    Init_Message_decod : Message_decod port map(
            reset => reset,
            msg6b_input => msg_codif,
```

```
            enable_decod => enable_msg,
            msg_4b_out => final_msg_temp
        );
        final_msg <= final_msg_temp;
-------------------------------------------------------------------------------------------------------------------
-- Proceso para mostrar en la salida la señal de 6 bits.
        process(clk_2,reset)

            variable i : integer := 0;
            variable clk_count : integer := 0;
        begin
            if clk_count = 5 then
                if reset = '1' then
                    i := 0;
                    clk_count := 0;
                    fin_cod_signal <= '0';
                elsif rising_edge(clk_2) then
                    fin_cod_signal <= msg_codif(5-i);
                    i := i + 1;
                end if;
                if i = 6 then
                    i := 0;
                end if;
            elsif reset = '1' then
                i := 0;
                clk_count := 0;
            elsif rising_edge(clk_2) then
                clk_count := clk_count + 1;
            end if;
        end process;
-------------------------------------------------------------------------------------------------------------------
-- Proceso para mostrar en la salida la señal de 4 bits final
        process(clk_1,reset)

            variable i : integer := 0;
            variable clk_count : integer := 0;

        begin
            if clk_count = 3 then
                if reset = '1' then
```

```
                    i := 0;
                    clk_count := 0;
                    fin_signal <= '0';
                elsif rising_edge(clk_1) then
                    fin_signal <= final_msg_temp(3-i);
                    i := i + 1;
                end if;
                if i = 4 then
                    i := 0;
                end if;
            elsif reset = '1' then
                i := 0;
                clk_count := 0;
            elsif rising_edge(clk_1) then
                clk_count := clk_count + 1;
            end if;
        end process;

end Behavioral;


----------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Correlation - Behavioral
-- Description:     Instancia donde se realiza el muestreo de la señal.
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity Correlation is
        Port (  reset : in  STD_LOGIC;
                enable : in  STD_LOGIC;
                clk_3 : in  STD_LOGIC;
                clk_2 : in  STD_LOGIC;
                clk_samp : in  STD_LOGIC;
                input : in  STD_LOGIC;
                dec_enable : out  STD_LOGIC;
```

```
                bit_position : out STD_LOGIC_VECTOR (2 downto 0);
                test_signal : out STD_LOGIC_VECTOR (5 downto 0);
                symbol_vector : out  STD_LOGIC_VECTOR (4 downto 0));
end Correlation;

architecture Behavioral of Correlation is

    COMPONENT sampling PORT(
            reset : in STD_LOGIC;
            enable_samp : in STD_LOGIC;
            clk_3 : in STD_LOGIC;          -- 1 MHz
            clk_samp : in STD_LOGIC;               -- 5 MHz
            zero : in STD_LOGIC_VECTOR (3 downto 0);  -- cuenta de ceros.
            uno : in STD_LOGIC_VECTOR (3 downto 0);  -- cuenta de unos.
            enable_deco : out STD_LOGIC;  -- Señal para habilitar la selección del bit.
            borra : out STD_LOGIC;  -- Señal para comprobaciones.
            prueba_s : out STD_LOGIC_VECTOR (5 downto 0);  -- variable para pruebas.
            symbol_vector_temp : out STD_LOGIC_VECTOR (4 downto 0);  -- Símbolo recuperado.
    );
    END COMPONENT;

-- Proceso para volcar el valor del símbolo recuperado en una variable que no sufra cambios durante la
-- lectura del siguiente símbolo.

    COMPONENT vector PORT(
            reset : in STD_LOGIC;
            enable_sample : in STD_LOGIC;
            clk_2 : in STD_LOGIC;          -- 100 kHz
            clk_samp : in STD_LOGIC;               -- 5 MHz
            s_vector_temp : in STD_LOGIC_VECTOR (4 downto 0);  -- símbolo recibido.
            b_position : out STD_LOGIC_VECTOR (2 downto 0);  -- posición del símbolo.
            test_signal : out STD_LOGIC_VECTOR (5 downto 0);  -- Variable para pruebas.
            s_vector : out STD_LOGIC_VECTOR (4 downto 0);  -- Variable para pruebas.

    );
    END COMPONENT;

    COMPONENT countingPORT(
            reset : in STD_LOGIC;
            c_enable : in STD_LOGIC;
```

```
            clk_4 : in STD_LOGIC;          -- 1 MHz
            clk_samp : in STD_LOGIC;               -- 5 MHz
            signal_in : in STD_LOGIC;
            prueba_signal : out STD_LOGIC_VECTOR (5 downto 0);  -- símbolo recibido.
            zero_out : out STD_LOGIC_VECTOR (3 downto 0);
            one_out : out STD_LOGIC_VECTOR (3 downto 0);

        );
        END COMPONENT;

        -- Señales auxiliares durante el muestreo.
        -- Vector temporal de 5 componentes para el valor del bit en curso.
        signal Symbol_bit_temp : STD_LOGIC_VECTOR (4 downto 0) := "00000";
        -- Señal para cuenta de las muestras que son cero.
        signal Zero_count : STD_LOGIC_VECTOR (3 downto 0) := "0000";
        -- Señal para cuenta de las muestras que son uno.
        signal One_count : STD_LOGIC_VECTOR (3 downto 0) := "0000";
        -- Señal para habilitar el muestreo en esta instancia.
        signal enable_corr : STD_LOGIC := '0';
        -- Señal para informar que se puede transmitir el vector del bit.
        signal flag : STD_LOGIC := '0';
        -- Señal para representar la posición del bit estudiado.
        signal position : STD_LOGIC_VECTOR (2 downto 0) := "000";
        -- Señal intermedias y auxiliares para comprobaciones.
        signal clk_enable_a : STD_LOGIC := '0';
        signal borrar : STD_LOGIC := '0';
        signal aux : STD_LOGIC := '0';
        signal clk_4 : STD_LOGIC := '1';
        signal auxiliar : STD_LOGIC_VECTOR (5 downto 0) := "000000";

begin
-- En esta instancia se activa una señal de habilitación.
        process (clk_samp, clk_3, reset, enable)
            variable m : integer := 0;          -- Cuenta para muestreo
        begin
            if enable = '0' or reset = '1' then
                enable_corr <= '0';
                aux <= '0';
            elsif enable = '1' and rising_edge(clk_3) and enable_corr = '0' then
```

```
                    enable_corr <= '1';
    end process;

-- En esta instancia se una señal inversa al reloj de 1 MHz, para uno de los subprocesos.
    process (clk_3, reset)
        variable m : integer := 0;          -- Cuenta para muestreo
    begin
        if enable = '0' or reset = '1' then
            enable_corr <= '0';
            aux <= '0';
        elsif enable = '1' and rising_edge(clk_3) and enable_corr = '0' then
            enable_corr <= '1';
    end process;

    Init_sampling : sampling port map(
        reset => reset,
        enable_samp => enable_corr,
        clk_3 => clk_4,
        clk_samp => clk_samp,
        uno => One_count,
        zero => Zero_count,
        enable_deco => dec_enable,
        prueba_s => test_signal,
        borra => aux,
        symbol_vector_temp => Symbol_bit_temp
    );

    Init_vector : vector port map(
        reset => reset,
        enable_sample => enable_corr,
        clk_2 => clk_2,
        clk_samp => clk_samp,
        s_vector_temp => Symbol_bit_temp,
        b_position => bit_position,
        s_vector => symbol_vector
    );

    Init_counting : counting port map(
        reset => reset,
        c_enable => enable_corr,
        clk_4 => clk_4,
```

```
                clk_samp => clk_samp,
                signal_in => input,
                prueba_signal => auxiliar,
                zero_out => Zero_count,
                one_out => One_count
        );
end Behavioral;


----------------------------------------------------------------------------------------------------------------------
-- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
--
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    sampling - Behavioral
-- Description:    Instancia donde se construye el vector con el valor del símbolo.
--
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;

    entity sampling is
       Port ( reset : in  STD_LOGIC;
           clk_3 : in  STD_LOGIC;
           clk_samp : in STD_LOGIC;
           enable_samp : in STD_LOGIC;
           zero : in STD_LOGIC_VECTOR (3 DOWNTO 0);
           uno : in STD_LOGIC_VECTOR (3 DOWNTO 0);
           enable_deco : out STD_LOGIC;
           borra : out STD_LOGIC;
           prueba_s : out STD_LOGIC_VECTOR (5 downto 0);
           symbol_vector_temp : out STD_LOGIC_VECTOR (4 DOWNTO 0));
    end sampling;
    architecture Behavioral of sampling is
           signal clk_enable_a : STD_LOGIC := '0';
           signal aux : STD_LOGIC := '0';
           signal borrar : STD_LOGIC := '0';
    begin
    -- En este proceso se crea un vector de 5 componentes que representa al símbolo leído, se va
    -- cumplimentando a medida que se muestrea la señal, cada dos ciclos del reloj de 1 MHz.
```

```
    process (clk_3, reset, enable_samp, clk_samp)
        variable m : integer := 0;    -- Cuenta para desplazarse por el vector
        begin
                if enable_samp = '0' or reset = '1' then
                        m := 0;
                        symbol_vector_temp <= "00000";
                        enable_deco <= '0';
                        clk_enable_a <= '1';
                        borrar <= '0';
                        aux <= '0';
                elsif enable_samp = '1' and rising_edge(clk_3) then
                                if zero > uno then
                                        symbol_vector_temp(4-m) <= '0';
                                        if m = 4 then
                                                m := 0;
                                                enable_deco <= '1';
--                                              prueba_s <= "101100";   -- "1111"
                                        elsif m < 5 then
                                                enable_deco <= '0';
                                                m := m + 1;
                                        end if;
                                elsif zero < uno then
                                        symbol_vector_temp(4-m) <= '1';
                                        if m = 4 then
                                                m := 0;
                                                enable_deco <= '1';
--                                              prueba_s <= "011010";   -- "1001"
                                        elsif m < 5 then
                                                enable_deco <= '0';
                                                m := m + 1;
                                        end if;
                                end if;
        end if;
        borra <= borrar;
    end process;
    end Behavioral;
```

```
----------------------------------------------------------------------------------
-- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
--
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    vector - Behavioral
-- Description:    Instancia donde se vuelca el valor temporal del simbolo en un vector
--                 para procesar.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;

entity vector is
   Port ( reset : in  STD_LOGIC;
          enable_sample : in  STD_LOGIC;
          clk_2 : in STD_LOGIC;                    -- 100 kHz
          clk_samp : in STD_LOGIC;                 -- 5 MHz
          s_vector_temp : in STD_LOGIC_VECTOR (4 downto 0);
          b_position : out STD_LOGIC_VECTOR (2 downto 0); -- Posición del bit actual en el mensaje.
          s_vector : out STD_LOGIC_VECTOR  (4 downto 0));  -- Valor del símbolo a falta de decisión.
end vector;
architecture Behavioral of vector is
          -- Señal para informar que se puede transmitir el vector del bit.
          signal flag : STD_LOGIC := '0';
          -- Señal para representar la posición del bit estudiado.
          signal position : STD_LOGIC_VECTOR (2 downto 0) := "000";
          signal s_vector_2 : STD_LOGIC_VECTOR (4 downto 0) := "00000";
begin
          -- En este proceso se vuelva el valor temporal del símbolo en un vector para su procesamiento.
          process (reset, enable_sample, clk_2, clk_samp)
          begin
                    if reset = '1' then
                              s_vector_2 <= "00000";
                              b_position <= "000";
                              position <= "000";
                              flag <= '0';
                    elsif enable_sample = '1' then
                              if rising_edge (clk_2) then
                                        flag <= '1';
                                        if position < "110" then
```

```
                                        position <= position + '1';
                                elsif position = "110" then
                                        position <= "001";
                                end if;
                        end if;
                        if flag = '1' then
                                s_vector_2 <= s_vector_temp;
                                flag <= '0';
                        end if;
                end if;
                b_position <= position;
                s_vector <= s_vector_2;
        end process;
end Behavioral;


----------------------------------------------------------------------------------------------------------------------------
-- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
--
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    counting - Behavioral
-- Description:    Instancia donde muestrea la señal y se registran las cuentas de ceros y de unos
--
----------------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;


entity counting is
   Port ( reset : in  STD_LOGIC;
          c_enable : STD_LOGIC;
          clk_4 : in  STD_LOGIC;
          clk_samp : in  STD_LOGIC;
          signal_in : in STD_LOGIC;
          prueba_signal : out STD_LOGIC_VECTOR (5 downto 0);
          zero_out : out STD_LOGIC_VECTOR (3 DOWNTO 0);
          one_out : out STD_LOGIC_VECTOR (3 DOWNTO 0));
end counting;
architecture Behavioral of counting is
          -- Señal para cuenta de las muestras que son cero.
```

```
        signal zero: STD_LOGIC_VECTOR (3 downto 0) := "0000";
        -- Señal para cuenta de las muestras que son uno.
        signal uno : STD_LOGIC_VECTOR (3 downto 0) := "0000";
begin
        process (clk_samp, reset, clk_4)
                        variable k : integer := 0;              -- Cuenta para muestreo
        begin
                if reset = '1' then

                        zero <= "0000";
                        uno <= "0000";
                        k := 0;
                elsif k >= 10 then
                        zero <= "0000";
                        uno <= "0000";
                        k := 0;
                elsif c_enable = '1' and rising_edge (clk_samp) then
                                if signal_in = '0' then
                                        zero <= zero + '1';
                                        prueba_signal <= "101100";   -- "1111"
                                elsif signal_in = '1' then
                                        prueba_signal <= "011010";   -- "1001"
                                        uno <= uno + '1';
                                end if;
                                k := k+1;
                end if;
                one_out <= uno;
                zero_out <= zero;
        end process;
end Behavioral;


----------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Decision - Behavioral
-- Description:        Instancia donde se decide el valor del bit muestreado.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use ieee.numeric_std.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity Decision is
        Port (  reset : in  STD_LOGIC;
                clk_2 : in STD_LOGIC;
                symbol_input : in  STD_LOGIC_VECTOR (4 downto 0);
                dec_enable : in  STD_LOGIC;
                bit_position : in  STD_LOGIC_VECTOR (2 downto 0);
                msg_6b_out : out  STD_LOGIC_VECTOR (5 downto 0);
                msg_enable : out  STD_LOGIC);
end Decision;

architecture Behavioral of Decision is
-- Señales auxiliares.
        signal msg_6b_temp : STD_LOGIC_VECTOR (5 downto 0) := "000000";
        signal flag : STD_LOGIC := '0';
        signal flag_msg : STD_LOGIC := '0';
        signal actual_bit : STD_LOGIC := '0';
-- Instancia donde se compara el valor obtenido en Correlation, para decidir el valor del bit.
        COMPONENT Bit_table PORT(
                reset : in STD_LOGIC;
                clk_2 : in STD_LOGIC;          -- 100 kHz
                dec_enable : in STD_LOGIC;     -- Señal habilitadora para realizar la decisión.
                symbol : in STD_LOGIC_VECTOR (4 downto 0);  -- Vector entrada desde el correlador.
                flag_enable : out STD_LOGIC;  -- Señal indicadora del comienzo de la conversión.
                current_bit : out STD_LOGIC    -- Bit final de la decisión.
        );
        END COMPONENT;

begin
--------------------------------------------------------------------------------
-- Instancia donde se compara el valor obtenido en Correlation, para decidir el valor del bit.
        Init_Bit_table : Bit_table port map(
                reset => reset,
                clk_2 => clk_2,
                dec_enable => dec_enable,
                symbol => symbol_input,
                flag_enable => flag,
```

```
                current_bit => actual_bit
        );
-- En este proceso se va cumplimentando el vector de 6 bits con el mensaje codificado a medida que
-- se va demodulando la señal y decidiendo el valor de cada uno de los bits.
-- Al finalizar con la cumplimentación del vector del mensaje codificado, se envía el vector hacia
-- la siguiente etapa.
        process (clk_2, reset)
                variable position : integer := 0;
        begin
                if reset = '1' then
                        msg_6b_out <= "000000";
                        msg_enable <= '0';
                elsif falling_edge(clk_2) then
                        if bit_position > "000" then
                                if bit_position = "001" then
                                        position := 1;
                                elsif bit_position = "010" then
                                        position := 2;
                                elsif bit_position = "011" then
                                        position := 3;
                                elsif bit_position = "100" then
                                        position := 4;
                                elsif bit_position = "101" then
                                        position := 5;
                                elsif bit_position = "110" then
                                        position := 6;
                                elsif bit_position = "000" then
                                        position := 0;
                                end if;
                                if position = 0 then
                                        msg_6b_temp(0) <= 'Z';
                                elsif position > 0 then
                                        msg_6b_temp(position-1) <= actual_bit;
                                end if;
                        end if;
                        if bit_position = "110" then
                                msg_enable <= '1';
                                flag_msg <= '1';
                        elsif bit_position < "110" then
                                msg_enable <= '0';
```

```
                        end if;
                        if flag_msg = '1' then
                                msg_6b_out <= msg_6b_temp;
                                flag_msg <= '0';
                        end if;
                end if;
        end process;
end Behavioral;

----------------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Bit_table - Behavioral
-- Description:     Instancia donde se elige entre los valores cero y uno dependiendo de la variable symbol.
----------------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Bit_table is
   Port ( reset : in  STD_LOGIC;
          clk_2 : in  STD_LOGIC;
          dec_enable : in  STD_LOGIC;
          symbol : in  STD_LOGIC_VECTOR (4 downto 0);
          flag_enable : out  STD_LOGIC;
          current_bit : out  STD_LOGIC);
end Bit_table;

architecture Behavioral of Bit_table is
begin
-- En este proceso se decide el valor del bit con respecto a la señal muestreada.
        process(clk_2, dec_enable, reset)
        begin
                if reset = '1' then
                        flag_enable <= '0';
                        current_bit <= 'Z';
                elsif rising_edge(dec_enable) then
                        flag_enable <= '1';
                elsif falling_edge(dec_enable) then
                        case symbol is
```

```
                when "11111" =>
                        current_bit <= '1';
                when "11110" =>
                        current_bit <= '1';
                when "11100" =>
                        current_bit <= '1';
                when "11000" =>
                        current_bit <= '1';
                when "10000" =>
                        current_bit <= '1';
                when "01111" =>
                        current_bit <= '0';
                when "00111" =>
                        current_bit <= '0';
                when "00011" =>
                        current_bit <= '0';
                when "00001" =>
                        current_bit <= '0';
                when "00000" =>
                        current_bit <= '0';
                when others =>  -- Casos no identificados
                        if symbol <= "01111" then
                                current_bit <= '0';
                        elsif symbol > "01111" then
                                current_bit <= '1';
                        else
                                current_bit <= 'Z';
                        end if;
            end case;
        end if;
    end process;
end Behavioral;
```

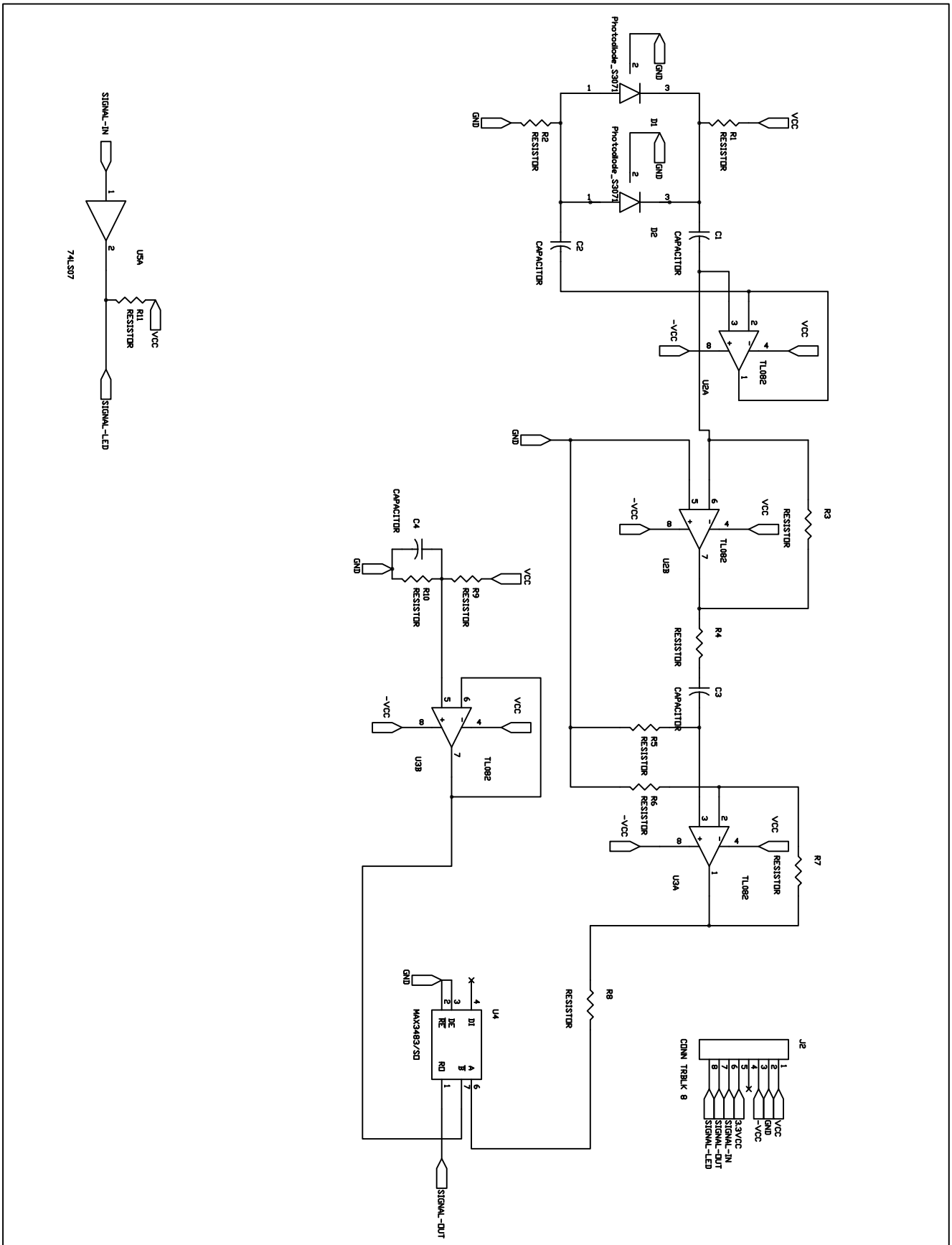
```
----------------------------------------------------------------------------------------------------------------------
---------- Company: University of La Laguna
-- Engineer: Airam Casañas Hernández
-- Project Name:   Sistema de transmisión VOOK basado en lámparas LED de iluminación.
-- Module Name:    Message_decod - Behavioral
-- Description:     Instancia donde realiza la descodificación del mensaje de 6 bits
--                  al mensaje final de 4 bits.
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Message_decod is
        Port (  reset : in  STD_LOGIC;
                msg6b_input : in  STD_LOGIC_VECTOR (5 downto 0);
                enable_decod : in  STD_LOGIC;
                msg_4b_out : out  STD_LOGIC_VECTOR (3 downto 0));
end Message_decod;

architecture Behavioral of Message_decod is
begin
-- El proceso se basa en un case dependiente del valor de entrada de 6 bits.
        process (msg6b_input, reset)
        begin
                if reset = '1' then
                        msg_4b_out <= "ZZZZ";
                else
                        case msg6b_input is
                                when "001110" =>  -- 0
                                        msg_4b_out <= "0000";
                                when "001101" =>  -- 1
                                        msg_4b_out <= "0001";
                                when "010011" =>  -- 2
                                        msg_4b_out <= "0010";
                                when "010110" =>  -- 3
                                        msg_4b_out <= "0011";
                                when "010101" =>  -- 4
                                        msg_4b_out <= "0100";
                                when "100011" =>  -- 5
                                        msg_4b_out <= "0101";
                                when "100110" =>  -- 6
```

```
                msg_4b_out <= "0110";
            when "100101" =>  -- 7
                msg_4b_out <= "0111";
            when "011001" =>  -- 8
                msg_4b_out <= "1000";
            when "011010" =>  -- 9
                msg_4b_out <= "1001";
            when "011100" =>  -- A
                msg_4b_out <= "1010";
            when "110001" =>  -- B
                msg_4b_out <= "1011";
            when "110010" =>  -- C
                msg_4b_out <= "1100";
            when "101001" =>  -- D
                msg_4b_out <= "1101";
            when "101010" =>  -- E
                msg_4b_out <= "1110";
            when "101100" =>  -- F
                msg_4b_out <= "1111";
            when others =>  -- Casos no identificados
                msg_4b_out <= "ZZZZ";
        end case;
    end if;
  end process;
end Behavioral;
```

6.2. **Planos del diseño en placa de circuito impreso.**

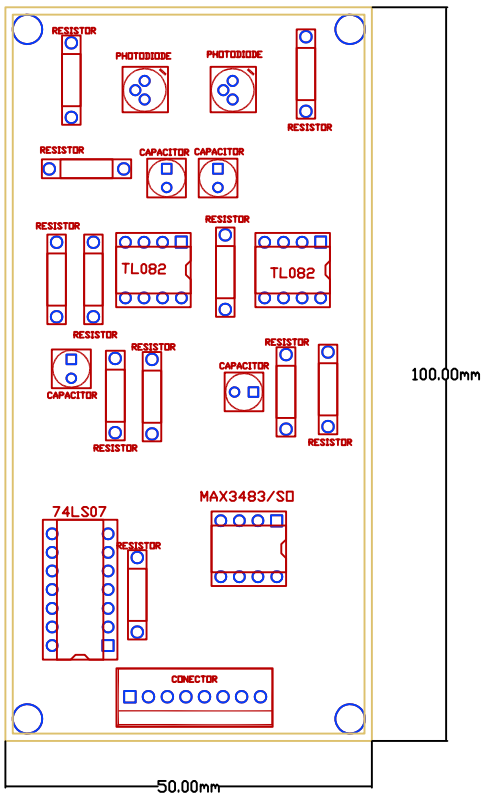


Sistema de transmisión VOOK basado en lámparas LED de iluminación

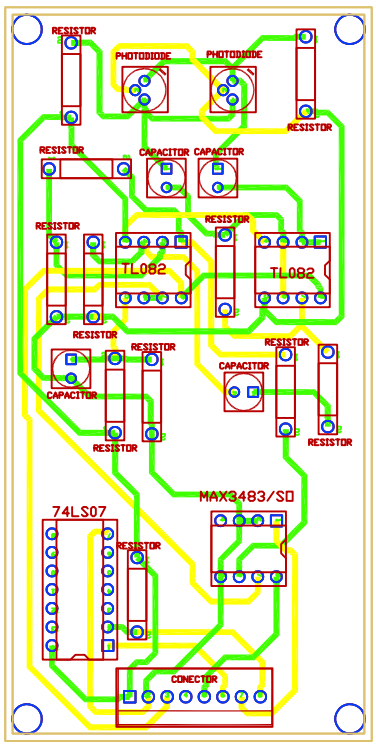
	Fecha	Autor		
Dibujado	*08–2020*	*Airam*	ULL Universidad de La Laguna	ESCUELA DE DOCTORADO Y ESTUDIOS DE POSGRADO *Master en Ingeniería Industrial – Esp. Electrónica* *Universidad de La Laguna*
Comprobado	*08–2020*	*Casañas Hernández*		
Id. s. normas	*UNE–EN–DIN*			

ESCALA:	Esquemático del driver del receptor	Nº P. : 1 Nom.Arch: Esquematico.dwg

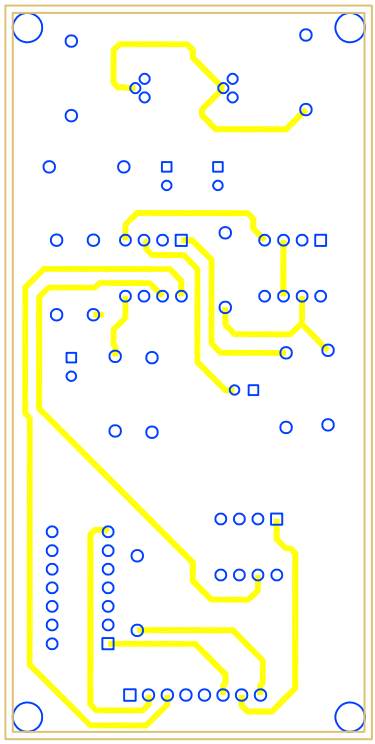
COMPONENTES



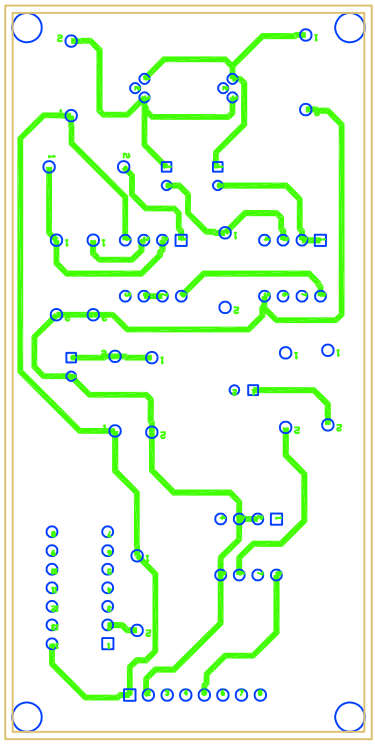
PCB COMPLETA



BOTTOM LAYER



TOP LAYER



Sistema de transmisión VOOK basado en lámparas LED de iluminación

	Fecha	Autor
Dibujado	*08–2020*	*Airam*
Comprobado	*08–2020*	*Casañas Hernández*
Id. s. normas	*UNE–EN–DIN*	

ULL Universidad de La Laguna

ESCUELA DE DOCTORADO Y ESTUDIOS DE POSGRADO
Master en Ingeniería Industrial – Esp. Electrónica
Universidad de La Laguna

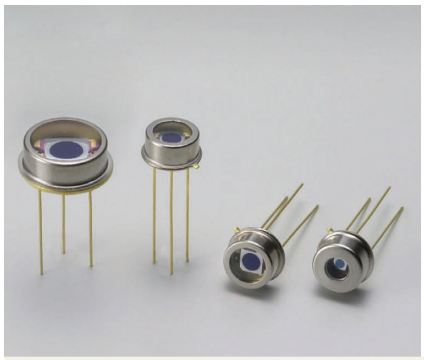
ESCALA: *1:1*	Placa de circuito impreso - capas TOP y BOTTOM	Nº P. : 2 Nom.Arch: pcb.dwg

6.3. Hojas de Características de componentes

- Fotodiodo S3071
- Circuito integrado MAX3485
- Placa de desarrollo Nexys A7

HAMAMATSU
PHOTON IS OUR BUSINESS

Si PIN photodiodes



S3071 S3072 S3399 S3883

Large area, high-speed Si PIN photodiodes

The S3071, S3072, S3399 and S3883 are Si PIN photodiodes having a relatively large photosensitive area from ϕ1.5 to ϕ5.0 mm yet they offer excellent frequency response from 40 to 300 MHz. These photodiodes are suitable for FSO (free space optics) and high-speed pulsed light detection.

Features

- **Photosensitive area size**
  S3071: ϕ5.0 mm
  S3072: ϕ3.0 mm
  S3399: ϕ3.0 mm
  S3883: ϕ1.5 mm
- **Cutoff frequency**
  S3071: 40 MHz (VR=24 V)
  S3072: 45 MHz (VR=24 V)
  S3399: 100 MHz (VR=10 V)
  S3883: 300 MHz (VR=20 V)
- **High reliability: TO-5/8 metal package**

Applications

- **FSO**
- **High-speed pulsed light detection**

Structure / Absolute maximum ratings

Type no.	Dimensional outline/ Window material*1	Package	Photosensitive area size (mm)	Effective photosensitive area (mm^2)	Absolute maximum ratings			
					Reverse voltage VR max. (V)	Power dissipation Pd (mW)	Operating temperature Topr (°C)	Storage temperature Tstg (°C)
S3071	(1)/K	TO-8	ϕ5.0	19.6	50	50	-40 to +100	-55 to +125
S3072	(2)/K	TO-5	ϕ3.0	7.0				
S3399	(3)/K		ϕ3.0	7.0	30			
S3883	(4)/K		ϕ1.5	1.7				

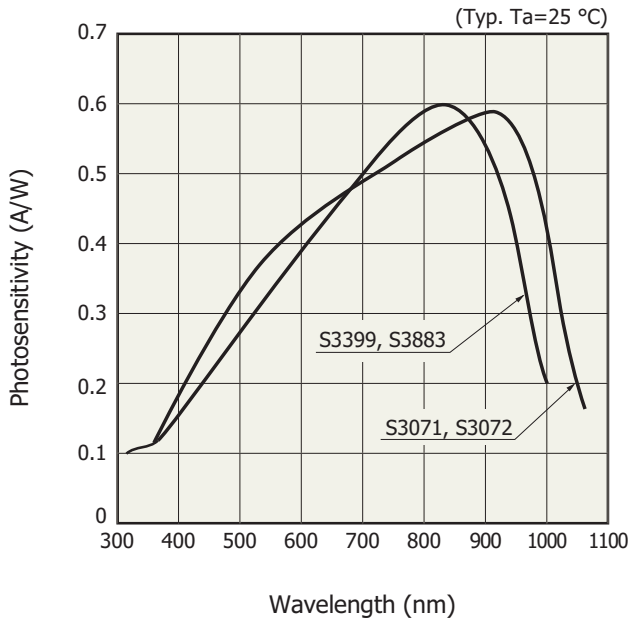
Note: Exceeding the absolute maximum ratings even momentarily may cause a drop in product quality. Always be sure to use the product within the absolute maximum ratings.
*1: Window material K=borosilicate glass

Electrical and optical characteristics (Typ. Ta=25 °C, unless otherwise noted)

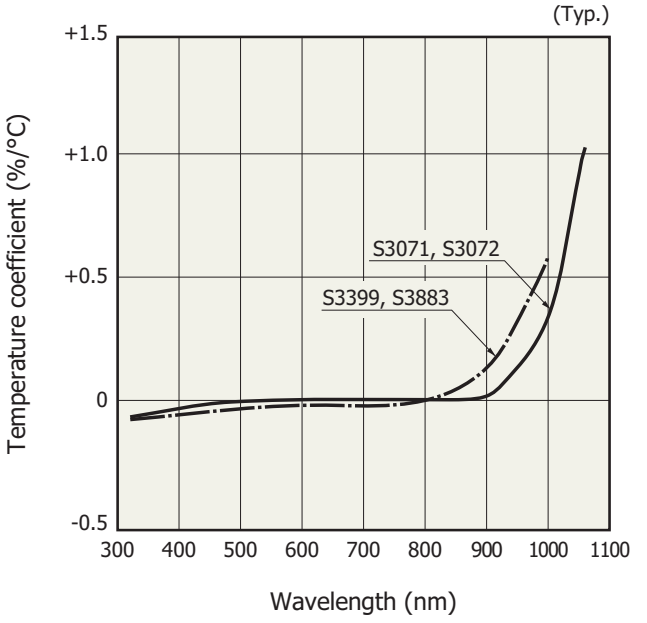
Type no.	Spectral response range λ (nm)	Peak sensitivity wavelength λp (nm)	Photosensitivity S (A/W)				Short circuit current Isc 100 lx (μA)	Dark current ID (nA)		Temp. coefficient of ID TCID (times/°C)	Cutoff frequency fc RL=50 Ω (MHz)	Terminal capacitance Ct f=1 MHz (pF)	Noize equivalent power NEP λ=λp ($W/Hz^{1/2}$)
			λp	660 nm	780 nm	830 nm		Typ.	Max.				
S3071	320 to 1060	920	0.6	0.47	0.54	0.56	17	0.5*2	10*2	1.15	40*2	18*2	2.1×10^{-14} *2
S3072							6.5	0.3*2	10*2		45*2	7*2	1.6×10^{-14} *2
S3399	320 to 1000	840	0.6	0.45	0.58	0.6	5.6	0.1*3	1.0*3	1.12	100*3	20*3	9.4×10^{-15} *3
S3883							1.4	0.05*4	1.0*4		300*4	6*4	6.7×10^{-15} *4

*2: VR=24 V
*3: VR=10 V
*4: VR=20 V

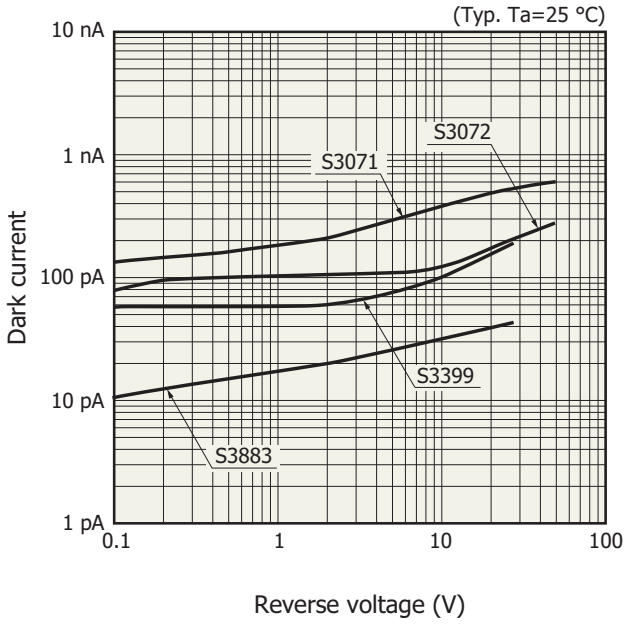
Spectral response



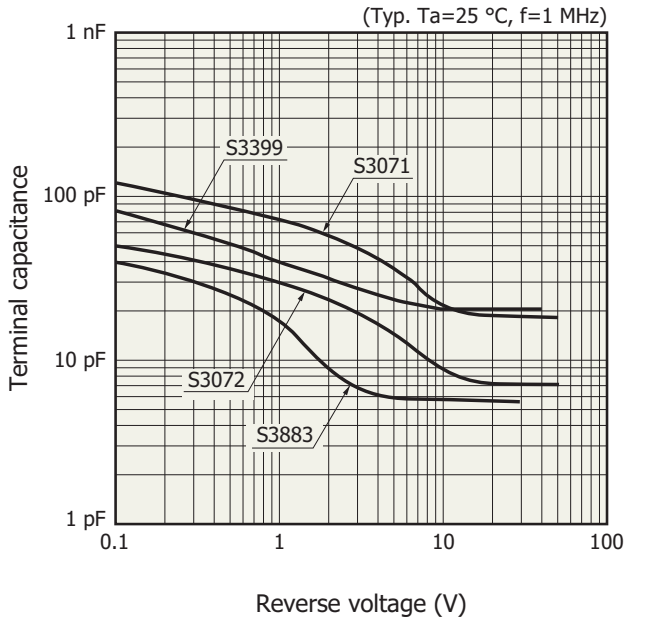
Photosensitivity temperature characteristics



Dark current vs. reverse voltage

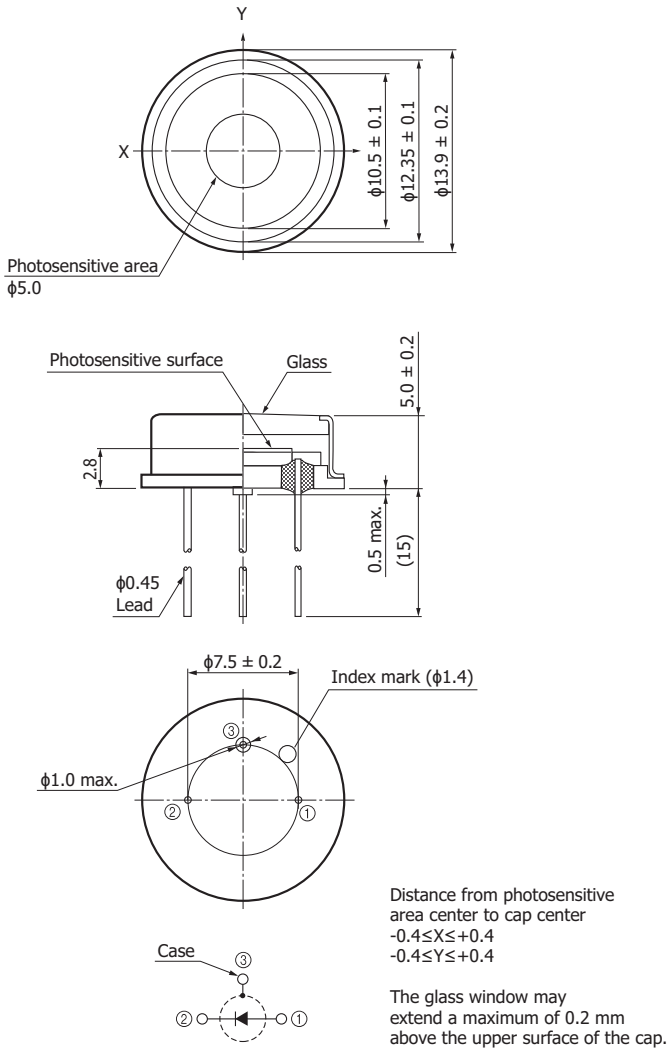


Terminal capacitance vs. reverse voltage



Dimensional outlines (unit: mm)

(1) S3071

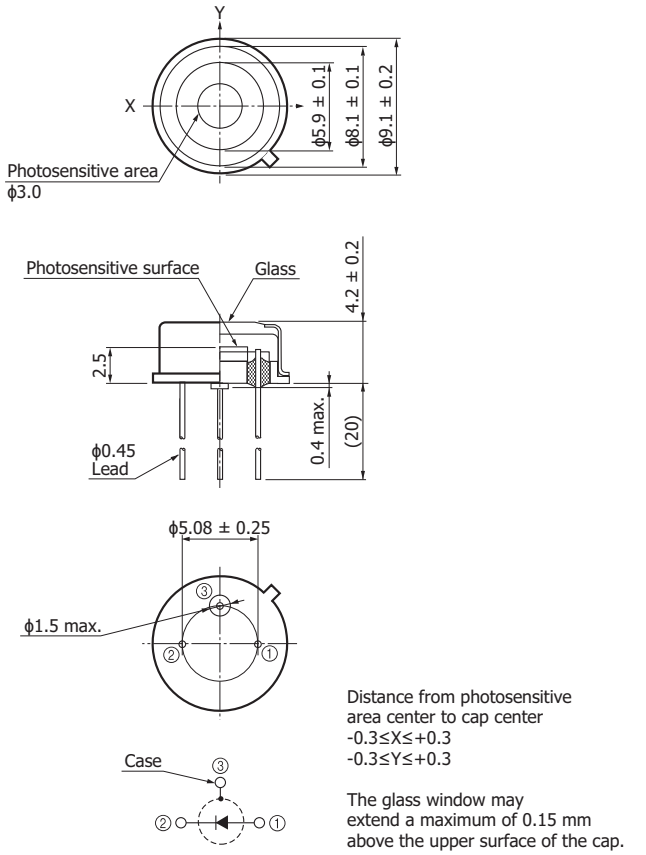


Distance from photosensitive area center to cap center
-0.4≤X≤+0.4
-0.4≤Y≤+0.4

The glass window may extend a maximum of 0.2 mm above the upper surface of the cap.

KPINA0027ED

(2) S3072

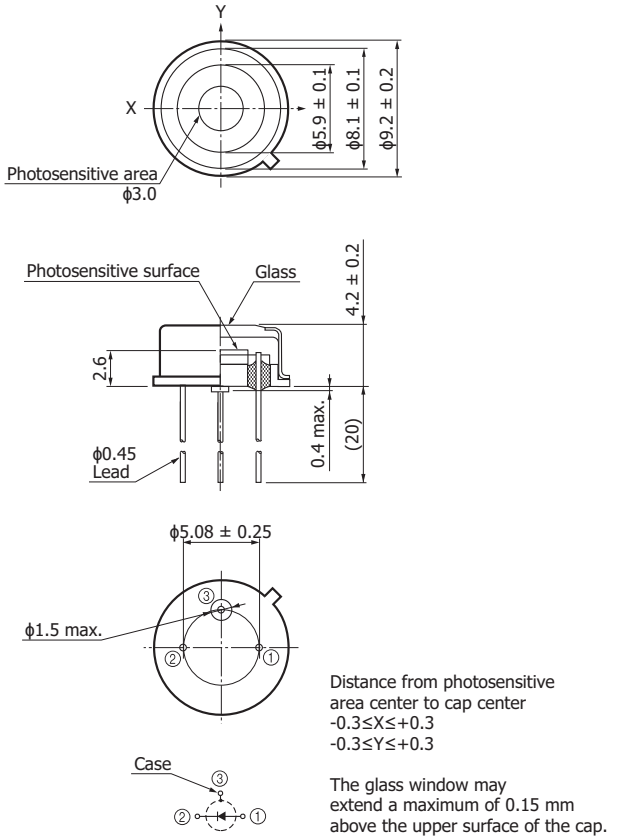


Distance from photosensitive area center to cap center
-0.3≤X≤+0.3
-0.3≤Y≤+0.3

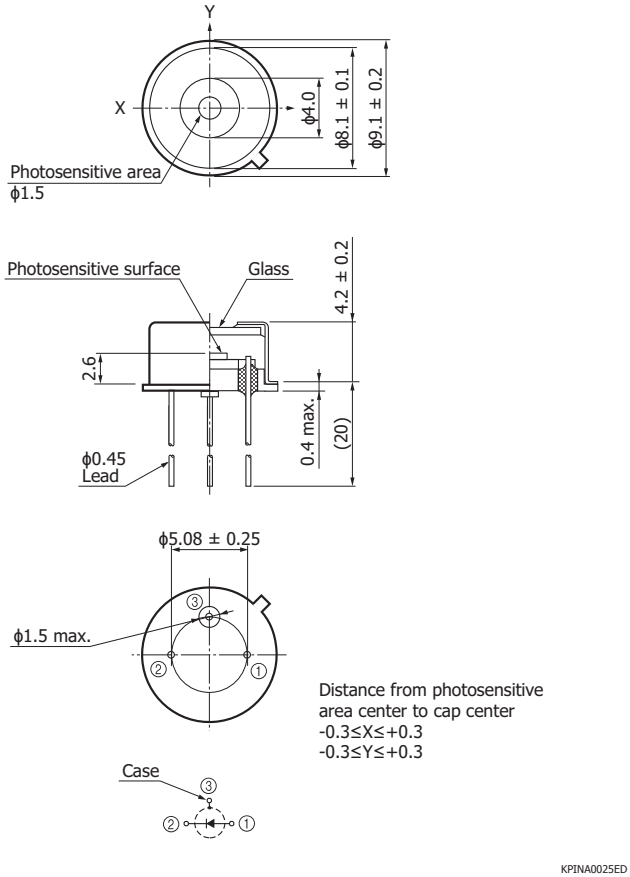
The glass window may extend a maximum of 0.15 mm above the upper surface of the cap.

KPINA0024EC

(3) S3399



(4) S3883



HAMAMATSU www.hamamatsu.com

HAMAMATSU PHOTONICS K.K., Solid State Division
1126-1 Ichino-cho, Higashi-ku, Hamamatsu City, 435-8558 Japan, Telephone: (81) 53-434-3311, Fax: (81) 53-434-5184
U.S.A.: Hamamatsu Corporation: 360 Foothill Road, Bridgewater, N.J. 08807, U.S.A., Telephone: (1) 908-231-0960, Fax: (1) 908-231-1218
Germany: Hamamatsu Photonics Deutschland GmbH: Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany, Telephone: (49) 8152-375-0, Fax: (49) 8152-265-8
France: Hamamatsu Photonics France S.A.R.L.: 19, Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France, Telephone: 33-(1) 69 53 71 00, Fax: 33-(1) 69 53 71 10
United Kingdom: Hamamatsu Photonics UK Limited: 2 Howard Court, 10 Tewin Road, Welwyn Garden City, Hertfordshire AL7 1BW, United Kingdom, Telephone: (44) 1707-294888, Fax: (44) 1707-325777
North Europe: Hamamatsu Photonics Norden AB: Torshamnsgatan 35 16440 Kista, Sweden, Telephone: (46) 8-509-031-00, Fax: (46) 8-509-031-01
Italy: Hamamatsu Photonics Italia S.r.l.: Strada della Moia, 1 int. 6, 20020 Arese (Milano), Italy, Telephone: (39) 02-93581733, Fax: (39) 02-93581741
China: Hamamatsu Photonics (China) Co., Ltd.: B1201, Jiaming Center, No.27 Dongsanhuan Beilu, Chaoyang District, Beijing 100020, China, Telephone: (86) 10-6586-6006, Fax: (86) 10-6586-2866

Cat. No. KPIN1044E06 Nov. 2014 DN

19-0122; Rev 8; 10/03

MAXIM

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

General Description

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120µA and 500µA of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1µA. All parts operate from a single 5V supply.

Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488–MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

Applications

Low-Power RS-485 Transceivers

Low-Power RS-422 Transceivers

Level Translators

Transceivers for EMI-Sensitive Applications

Industrial-Control Local Area Networks

Next Generation Device Features

- **For Fault-Tolerant Applications**
  MAX3430: ±80V Fault-Protected, Fail-Safe, 1/4 Unit Load, +3.3V, RS-485 Transceiver
  MAX3440E–MAX3444E: ±15kV ESD-Protected, ±60V Fault-Protected, 10Mbps, Fail-Safe, RS-485/J1708 Transceivers
- **For Space-Constrained Applications**
  MAX3460–MAX3464: +5V, Fail-Safe, 20Mbps, Profibus RS-485/RS-422 Transceivers
  MAX3362: +3.3V, High-Speed, RS-485/RS-422 Transceiver in a SOT23 Package
  MAX3280E–MAX3284E: ±15kV ESD-Protected, 52Mbps, +3V to +5.5V, SOT23, RS-485/RS-422, True Fail-Safe Receivers
  MAX3293/MAX3294/MAX3295: 20Mbps, +3.3V, SOT23, RS-855/RS-422 Transmitters
- **For Multiple Transceiver Applications**
  MAX3030E–MAX3033E: ±15kV ESD-Protected, +3.3V, Quad RS-422 Transmitters
- **For Fail-Safe Applications**
  MAX3080–MAX3089: Fail-Safe, High-Speed (10Mbps), Slew-Rate-Limited RS-485/RS-422 Transceivers
- **For Low-Voltage Applications**
  MAX3483E/MAX3485E/MAX3486E/MAX3488E/MAX3490E/MAX3491E: +3.3V Powered, ±15kV ESD-Protected, 12Mbps, Slew-Rate-Limited, True RS-485/RS-422 Transceivers

Ordering Information appears at end of data sheet.

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF TRANSMITTERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC}) 12V
Control Input Voltage ($\overline{RE}$, DE) ..... -0.5V to (V_{CC} + 0.5V)
Driver Input Voltage (DI) ..... -0.5V to (V_{CC} + 0.5V)
Driver Output Voltage (A, B) ..... -8V to +12.5V
Receiver Input Voltage (A, B) ..... -8V to +12.5V
Receiver Output Voltage (RO) ..... -0.5V to (V_{CC} +0.5V)
Continuous Power Dissipation (T_A = +70°C)
8-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 727mW
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) ..800mW
8-Pin SO (derate 5.88mW/°C above +70°C) 471mW
14-Pin SO (derate 8.33mW/°C above +70°C) 667mW
8-Pin µMAX (derate 4.1mW/°C above +70°C) 830mW
8-Pin CERDIP (derate 8.00mW/°C above +70°C) 640mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
Operating Temperature Ranges
MAX4_ _C_ _/MAX1487C_ A ..... 0°C to +70°C
MAX4_ _E_ _/MAX1487E_ A ..... -40°C to +85°C
MAX4_ _MJ_/MAX1487MJA ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Differential Driver Output (no load)	V_{OD1}					5	V
Differential Driver Output (with load)	V_{OD2}	R = 50Ω (RS-422)		2			V
		R = 27Ω (RS-485), Figure 4		1.5		5	
Change in Magnitude of Driver Differential Output Voltage for Complementary Output States	ΔV_{OD}	R = 27Ω or 50Ω, Figure 4				0.2	V
Driver Common-Mode Output Voltage	V_{OC}	R = 27Ω or 50Ω, Figure 4				3	V
Change in Magnitude of Driver Common-Mode Output Voltage for Complementary Output States	ΔV_{OD}	R = 27Ω or 50Ω, Figure 4				0.2	V
Input High Voltage	V_{IH}	DE, DI, $\overline{RE}$		2.0			V
Input Low Voltage	V_{IL}	DE, DI, $\overline{RE}$				0.8	V
Input Current	I_{IN1}	DE, DI, $\overline{RE}$				±2	µA
Input Current (A, B)	I_{IN2}	DE = 0V; V_{CC} = 0V or 5.25V, all devices except MAX487/MAX1487	V_{IN} = 12V			1.0	mA
			V_{IN} = -7V			-0.8	
		MAX487/MAX1487, DE = 0V, V_{CC} = 0V or 5.25V	V_{IN} = 12V			0.25	mA
			V_{IN} = -7V			-0.2	
Receiver Differential Threshold Voltage	V_{TH}	-7V ≤ V_{CM} ≤ 12V		-0.2		0.2	V
Receiver Input Hysteresis	ΔV_{TH}	V_{CM} = 0V			70		mV
Receiver Output High Voltage	V_{OH}	I_O = -4mA, V_{ID} = 200mV		3.5			V
Receiver Output Low Voltage	V_{OL}	I_O = 4mA, V_{ID} = -200mV				0.4	V
Three-State (high impedance) Output Current at Receiver	I_{OZR}	0.4V ≤ V_O ≤ 2.4V				±1	µA
Receiver Input Resistance	R_{IN}	-7V ≤ V_{CM} ≤ 12V, all devices except MAX487/MAX1487		12			kΩ
		-7V ≤ V_{CM} ≤ 12V, MAX487/MAX1487		48			kΩ

DC ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS			MIN	TYP	MAX	UNITS
No-Load Supply Current (Note 3)	I_{CC}	MAX488/MAX489, DE, DI, $\overline{RE}$ = 0V or V_{CC}				120	250	µA
		MAX490/MAX491, DE, DI, $\overline{RE}$ = 0V or V_{CC}				300	500	
		MAX481/MAX485, $\overline{RE}$ = 0V or V_{CC}	DE = V_{CC}			500	900	
			DE = 0V			300	500	
		MAX1487, $\overline{RE}$ = 0V or V_{CC}	DE = V_{CC}			300	500	
			DE = 0V			230	400	
		MAX483/MAX487, $\overline{RE}$ = 0V or V_{CC}	DE = 5V	MAX483		350	650	
				MAX487		250	400	
			DE = 0V			120	250	
Supply Current in Shutdown	I_{SHDN}	MAX481/483/487, DE = 0V, $\overline{RE}$ = V_{CC}				0.1	10	µA
Driver Short-Circuit Current, V_O = High	I_{OSD1}	-7V ≤ V_O ≤12V (Note 4)			35		250	mA
Driver Short-Circuit Current, V_O = Low	I_{OSD2}	-7V ≤ V_O ≤12V (Note 4)			35		250	mA
Receiver Short-Circuit Current	I_{OSR}	0V ≤ V_O ≤ V_{CC}			7		95	mA

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Driver Input to Output	t_{PLH}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		10	30	60	ns
	t_{PHL}			10	30	60	
Driver Output Skew to Output	t_{SKEW}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF			5	10	ns
Driver Rise or Fall Time	t_R, t_F	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	MAX481, MAX485, MAX1487	3	15	40	ns
			MAX490C/E, MAX491C/E	5	15	25	
			MAX490M, MAX491M	3	15	40	
Driver Enable to Output High	t_{ZH}	Figures 7 and 9, C_L = 100pF, S2 closed			40	70	ns
Driver Enable to Output Low	t_{ZL}	Figures 7 and 9, C_L = 100pF, S1 closed			40	70	ns
Driver Disable Time from Low	t_{LZ}	Figures 7 and 9, C_L = 15pF, S1 closed			40	70	ns
Driver Disable Time from High	t_{HZ}	Figures 7 and 9, C_L = 15pF, S2 closed			40	70	ns
Receiver Input to Output	t_{PLH}, t_{PHL}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	MAX481, MAX485, MAX1487	20	90	200	ns
			MAX490C/E, MAX491C/E	20	90	150	
			MAX490M, MAX491M	20	90	200	
$\lvert t_{PLH} - t_{PHL} \rvert$ Differential Receiver Skew	t_{SKD}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF			13		ns
Receiver Enable to Output Low	t_{ZL}	Figures 5 and 11, C_{RL} = 15pF, S1 closed			20	50	ns
Receiver Enable to Output High	t_{ZH}	Figures 5 and 11, C_{RL} = 15pF, S2 closed			20	50	ns
Receiver Disable Time from Low	t_{LZ}	Figures 5 and 11, C_{RL} = 15pF, S1 closed			20	50	ns
Receiver Disable Time from High	t_{HZ}	Figures 5 and 11, C_{RL} = 15pF, S2 closed			20	50	ns
Maximum Data Rate	f_{MAX}			2.5			Mbps
Time to Shutdown	t_{SHDN}	MAX481 (Note 5)		50	200	600	ns

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487 (continued)

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Enable from Shutdown to Output High (MAX481)	$t_{ZH(SHDN)}$	Figures 7 and 9, C_L = 100pF, S2 closed		40	100	ns
Driver Enable from Shutdown to Output Low (MAX481)	$t_{ZL(SHDN)}$	Figures 7 and 9, C_L = 100pF, S1 closed		40	100	ns
Receiver Enable from Shutdown to Output High (MAX481)	$t_{ZH(SHDN)}$	Figures 5 and 11, C_L = 15pF, S2 closed, A - B = 2V		300	1000	ns
Receiver Enable from Shutdown to Output Low (MAX481)	$t_{ZL(SHDN)}$	Figures 5 and 11, C_L = 15pF, S1 closed, B - A = 2V		300	1000	ns

SWITCHING CHARACTERISTICS—MAX483, MAX487/MAX488/MAX489

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Input to Output	t_{PLH}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250	800	2000	ns
	t_{PHL}		250	800	2000	
Driver Output Skew to Output	t_{SKEW}	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		100	800	ns
Driver Rise or Fall Time	t_R, t_F	Figures 6 and 8, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250		2000	ns
Driver Enable to Output High	t_{ZH}	Figures 7 and 9, C_L = 100pF, S2 closed	250		2000	ns
Driver Enable to Output Low	t_{ZL}	Figures 7 and 9, C_L = 100pF, S1 closed	250		2000	ns
Driver Disable Time from Low	t_{LZ}	Figures 7 and 9, C_L = 15pF, S1 closed	300		3000	ns
Driver Disable Time from High	t_{HZ}	Figures 7 and 9, C_L = 15pF, S2 closed	300		3000	ns
Receiver Input to Output	t_{PLH}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF	250		2000	ns
	t_{PHL}		250		2000	
| t_{PLH} - t_{PHL} | Differential Receiver Skew	t_{SKD}	Figures 6 and 10, R_{DIFF} = 54Ω, C_{L1} = C_{L2} = 100pF		100		ns
Receiver Enable to Output Low	t_{ZL}	Figures 5 and 11, C_{RL} = 15pF, S1 closed		20	50	ns
Receiver Enable to Output High	t_{ZH}	Figures 5 and 11, C_{RL} = 15pF, S2 closed		20	50	ns
Receiver Disable Time from Low	t_{LZ}	Figures 5 and 11, C_{RL} = 15pF, S1 closed		20	50	ns
Receiver Disable Time from High	t_{HZ}	Figures 5 and 11, C_{RL} = 15pF, S2 closed		20	50	ns
Maximum Data Rate	f_{MAX}	t_{PLH}, t_{PHL} < 50% of data period	250			kbps
Time to Shutdown	t_{SHDN}	MAX483/MAX487 (Note 5)	50	200	600	ns
Driver Enable from Shutdown to Output High	$t_{ZH(SHDN)}$	MAX483/MAX487, Figures 7 and 9, C_L = 100pF, S2 closed			2000	ns
Driver Enable from Shutdown to Output Low	$t_{ZL(SHDN)}$	MAX483/MAX487, Figures 7 and 9, C_L = 100pF, S1 closed			2000	ns
Receiver Enable from Shutdown to Output High	$t_{ZH(SHDN)}$	MAX483/MAX487, Figures 5 and 11, C_L = 15pF, S2 closed			2500	ns
Receiver Enable from Shutdown to Output Low	$t_{ZL(SHDN)}$	MAX483/MAX487, Figures 5 and 11, C_L = 15pF, S1 closed			2500	ns

NOTES FOR ELECTRICAL/SWITCHING CHARACTERISTICS

Note 1: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 2: All typical specifications are given for V_{CC} = 5V and T_A = +25°C.

Note 3: Supply current specification is valid for loaded transmitters when DE = 0V.

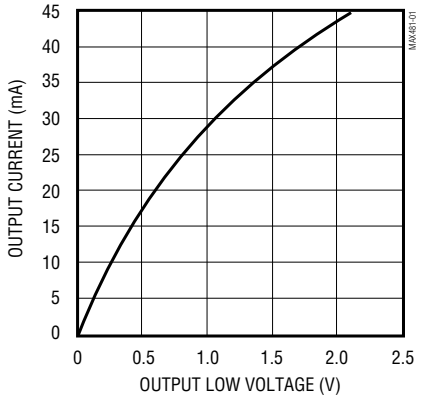
Note 4: Applies to peak current. See *Typical Operating Characteristics.*

Note 5: The MAX481/MAX483/MAX487 are put into shutdown by bringing $\overline{RE}$ high and DE low. If the inputs are in this state for less than 50ns, the parts are guaranteed not to enter shutdown. If the inputs are in this state for at least 600ns, the parts are guaranteed to have entered shutdown. See *Low-Power Shutdown Mode* section.

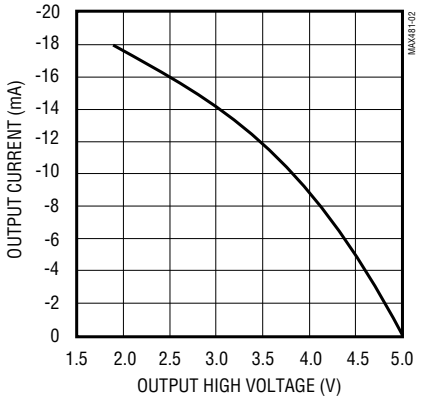
Typical Operating Characteristics

(V_{CC} = 5V, T_A = +25°C, unless otherwise noted.)

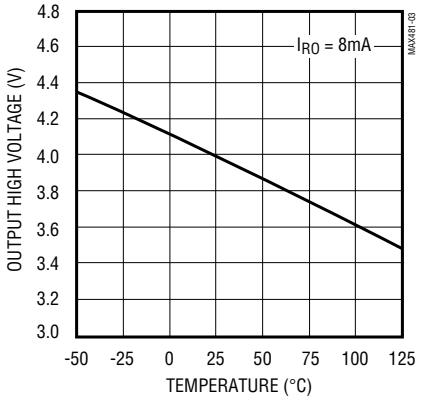
OUTPUT CURRENT vs. RECEIVER OUTPUT LOW VOLTAGE



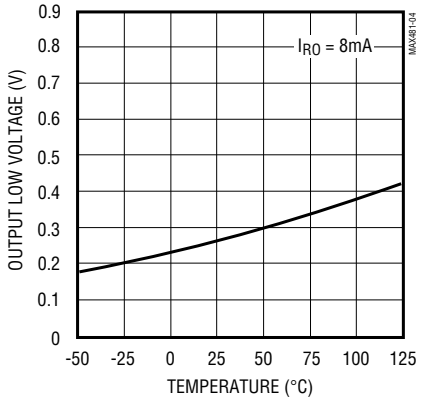
OUTPUT CURRENT vs. RECEIVER OUTPUT HIGH VOLTAGE



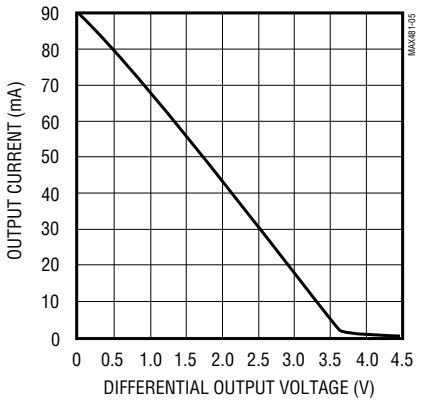
RECEIVER OUTPUT HIGH VOLTAGE vs. TEMPERATURE



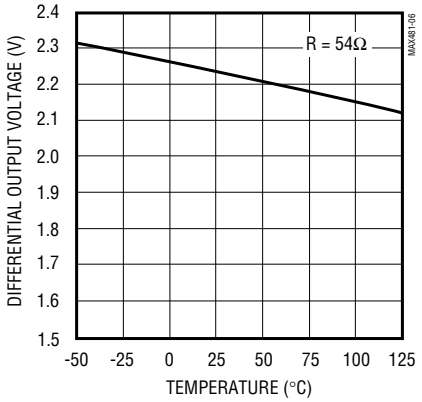
RECEIVER OUTPUT LOW VOLTAGE vs. TEMPERATURE



DRIVER OUTPUT CURRENT vs. DIFFERENTIAL OUTPUT VOLTAGE



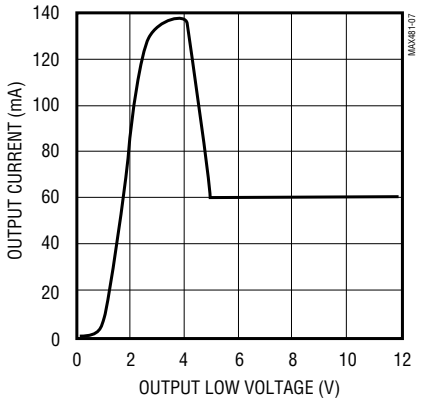
DRIVER DIFFERENTIAL OUTPUT VOLTAGE vs. TEMPERATURE



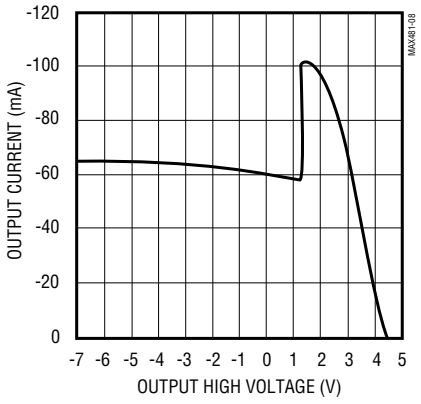
Typical Operating Characteristics (continued)

(V_{CC} = 5V, T_A = +25°C, unless otherwise noted.)

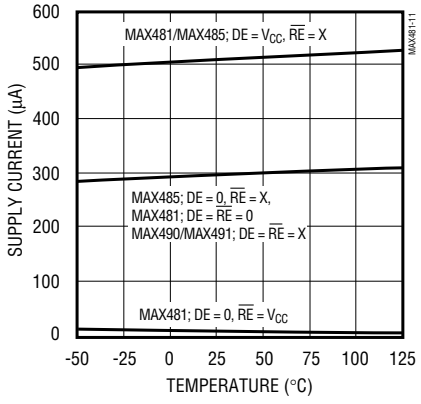
OUTPUT CURRENT vs. DRIVER OUTPUT LOW VOLTAGE



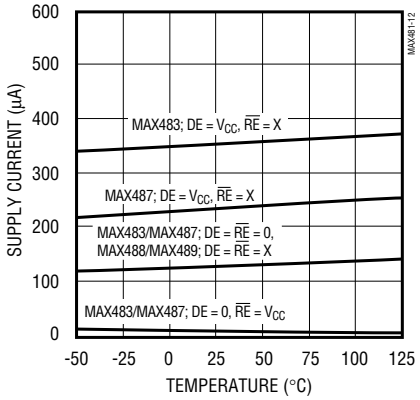
OUTPUT CURRENT vs. DRIVER OUTPUT HIGH VOLTAGE



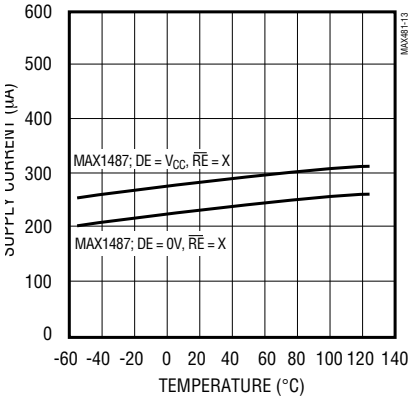
MAX481/MAX485/MAX490/MAX491 SUPPLY CURRENT vs. TEMPERATURE



MAX483/MAX487–MAX489 SUPPLY CURRENT vs. TEMPERATURE



MAX1487 SUPPLY CURRENT vs. TEMPERATURE



Pin Description

PIN					NAME	FUNCTION
MAX481/MAX483/ MAX485/MAX487/ MAX1487		MAX488/ MAX490		MAX489/ MAX491		
DIP/SO	µMAX	DIP/SO	µMAX	DIP/SO		
1	3	2	4	2	RO	Receiver Output: If A > B by 200mV, RO will be high; If A < B by 200mV, RO will be low.
2	4	—	—	3	$\overline{RE}$	Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when $\overline{RE}$ is high.
3	5	—	—	4	DE	Driver Output Enable. The driver outputs, Y and Z, are enabled by bringing DE high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low.
4	6	3	5	5	DI	Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low.
5	7	4	6	6, 7	GND	Ground
—	—	5	7	9	Y	Noninverting Driver Output
—	—	6	8	10	Z	Inverting Driver Output
6	8	—	—	—	A	Noninverting Receiver Input and Noninverting Driver Output
—	—	8	2	12	A	Noninverting Receiver Input
7	1	—	—	—	B	Inverting Receiver Input and Inverting Driver Output
—	—	7	1	11	B	Inverting Receiver Input
8	2	1	3	14	V_{CC}	Positive Supply: 4.75V ≤ V_{CC} ≤ 5.25V
—	—	—	—	1, 8, 13	N.C.	No Connect—not internally connected

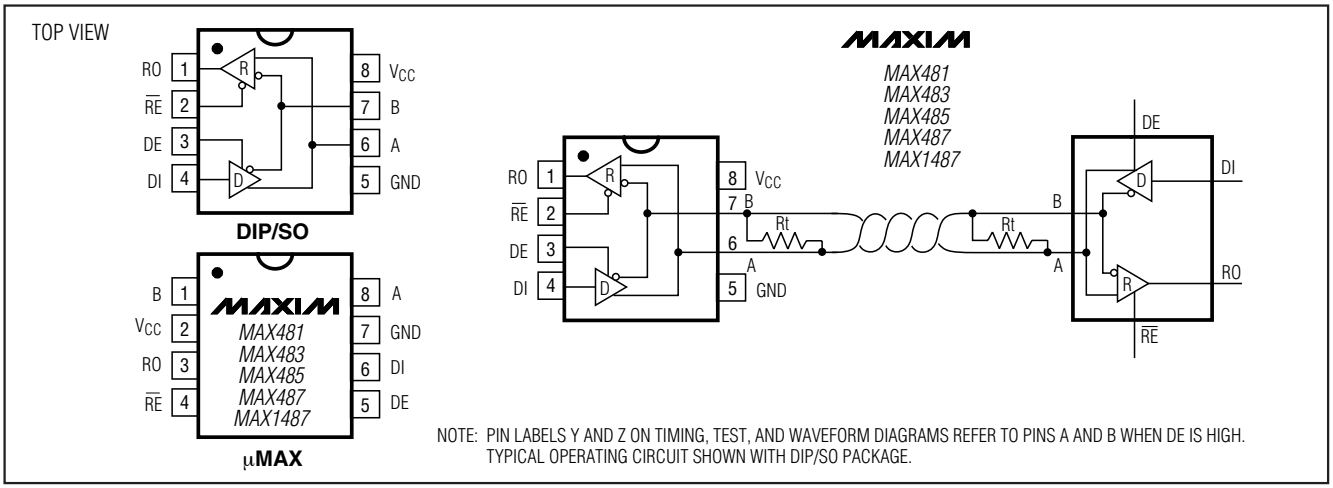


Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

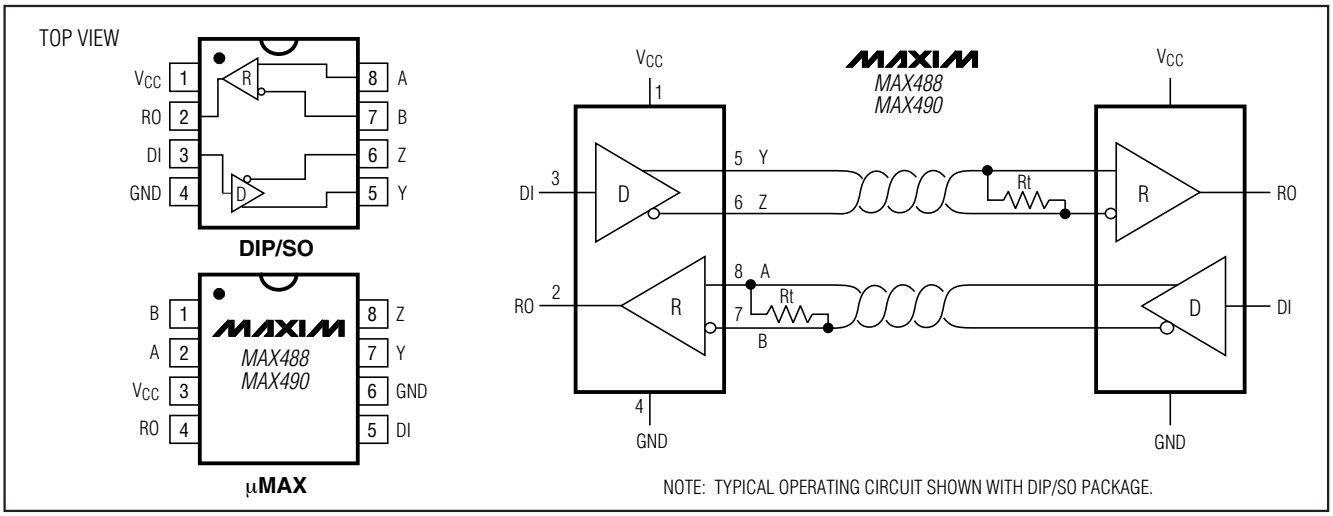


Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

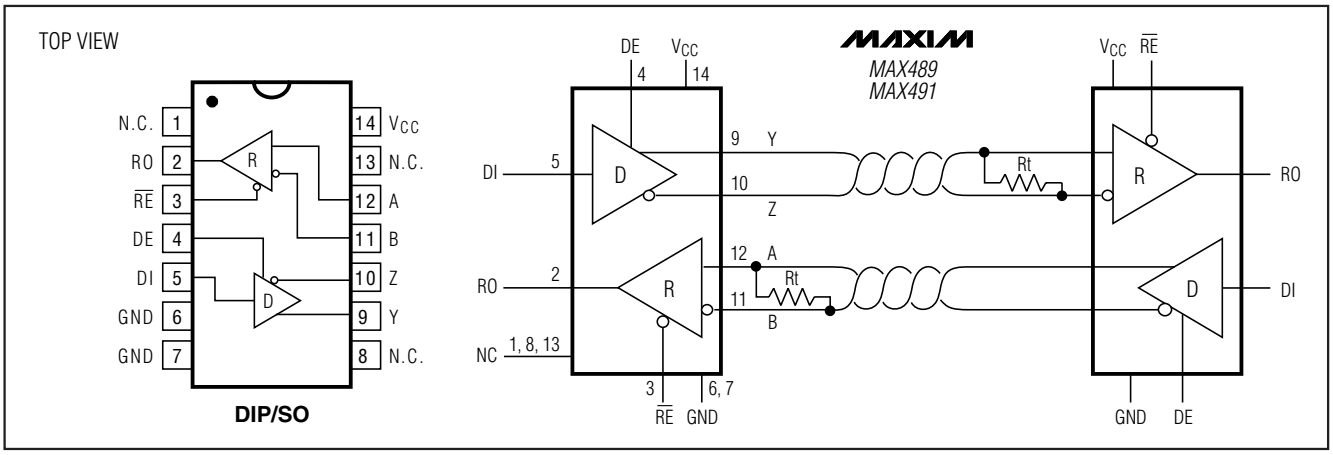


Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

Applications Information

The MAX481/MAX483/MAX485/MAX487–MAX491 and MAX1487 are low-power transceivers for RS-485 and RS-422 communications. The MAX481, MAX485, MAX490, MAX491, and MAX1487 can transmit and receive at data rates up to 2.5Mbps, while the MAX483, MAX487, MAX488, and MAX489 are specified for data rates up to 250kbps. The MAX488–MAX491 are full-duplex transceivers while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are half-duplex. In addition, Driver Enable (DE) and Receiver Enable ($\overline{RE}$) pins are included on the MAX481, MAX483, MAX485, MAX487, MAX489, MAX491, and MAX1487. When disabled, the driver and receiver outputs are high impedance.

MAX487/MAX1487: 128 Transceivers on the Bus

The 48kΩ, $^1/_4$-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12kΩ input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488–MAX491 have standard 12kΩ Receiver Input impedance.

Test Circuits

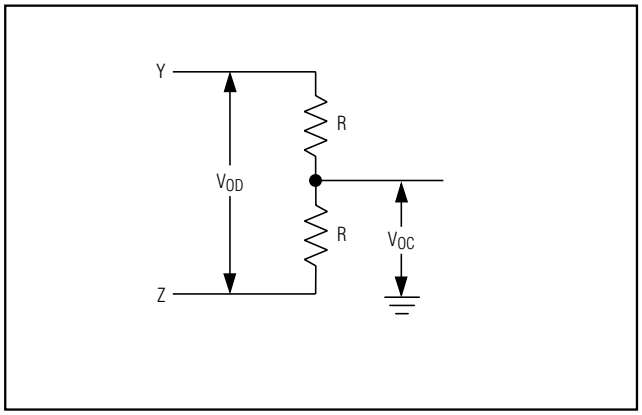


Figure 4. Driver DC Test Load

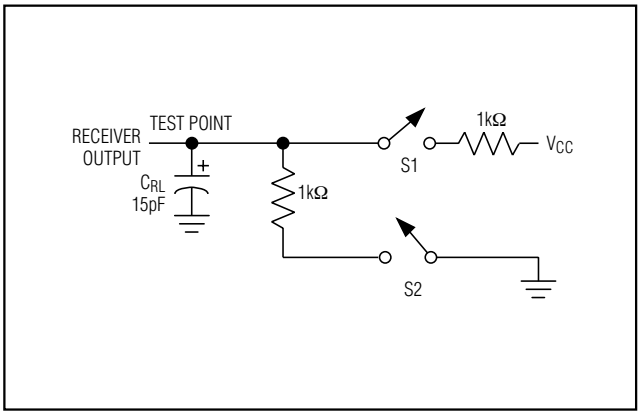


Figure 5. Receiver Timing Test Load

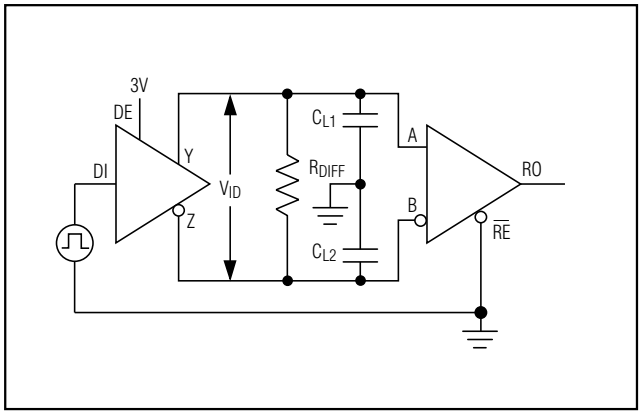


Figure 6. Driver/Receiver Timing Test Circuit

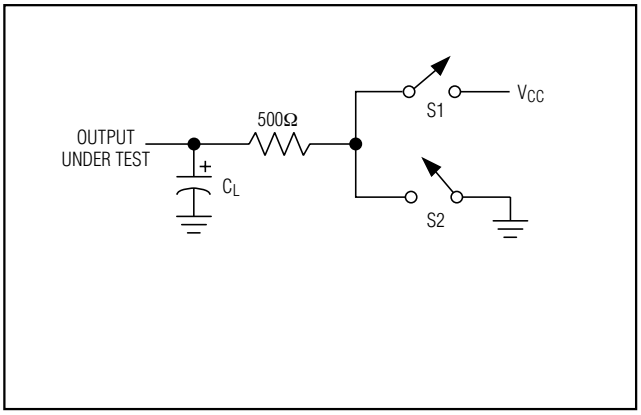


Figure 7. Driver Timing Test Load

MAX483/MAX487/MAX488/MAX489: Reduced EMI and Reflections

The MAX483 and MAX487–MAX489 are slew-rate limited, minimizing EMI and reducing reflections caused by improperly terminated cables. Figure 12 shows the driver output waveform and its Fourier analysis of a 150kHz signal transmitted by a MAX481, MAX485, MAX490, MAX491, or MAX1487. High-frequency harmonics with large amplitudes are evident. Figure 13 shows the same information displayed for a MAX483, MAX487, MAX488, or MAX489 transmitting under the same conditions. Figure 13's high-frequency harmonics have much lower amplitudes, and the potential for EMI is significantly reduced.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Switching Waveforms

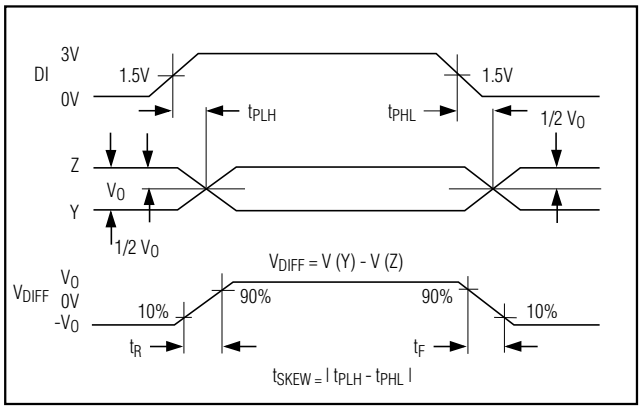


Figure 8. Driver Propagation Delays

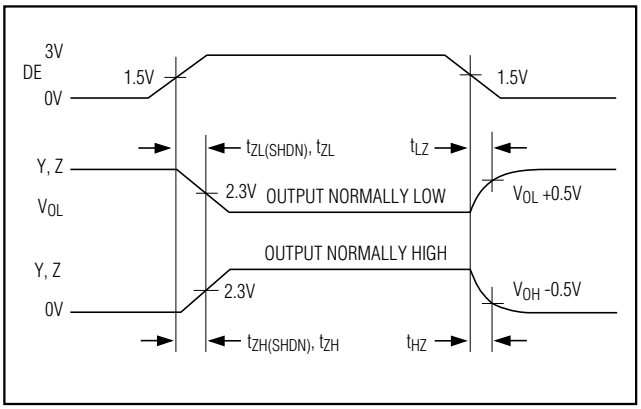


Figure 9. Driver Enable and Disable Times (except MAX488 and MAX490)

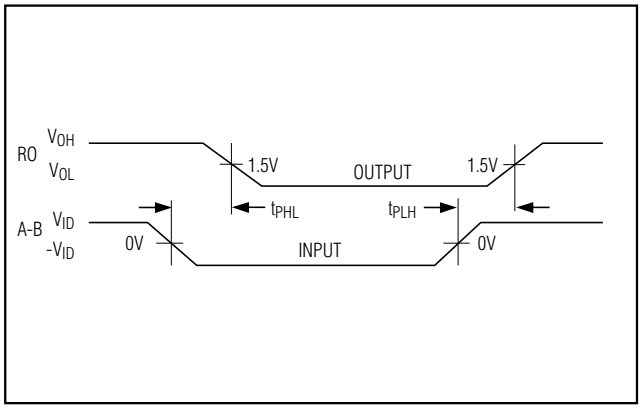


Figure 10. Receiver Propagation Delays

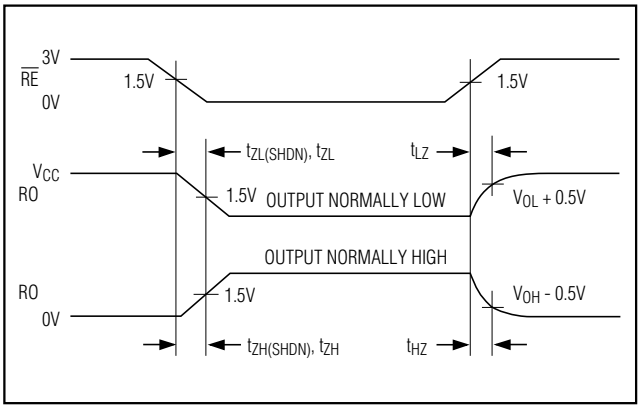


Figure 11. Receiver Enable and Disable Times (except MAX488 and MAX490)

Function Tables (MAX481/MAX483/MAX485/MAX487/MAX1487)

Table 1. Transmitting

INPUTS			OUTPUTS	
$\overline{RE}$	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	High-Z	High-Z
1	0	X	High-Z*	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

Table 2. Receiving

INPUTS			OUTPUT
$\overline{RE}$	DE	A-B	RO
0	0	≥ +0.2V	1
0	0	≤ -0.2V	0
0	0	Inputs open	1
1	0	X	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

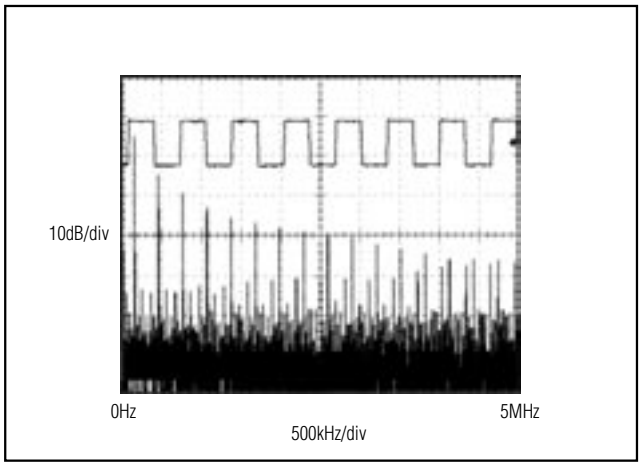


Figure 12. Driver Output Waveform and FFT Plot of MAX481/ MAX485/MAX490/MAX491/MAX1487 Transmitting a 150kHz Signal

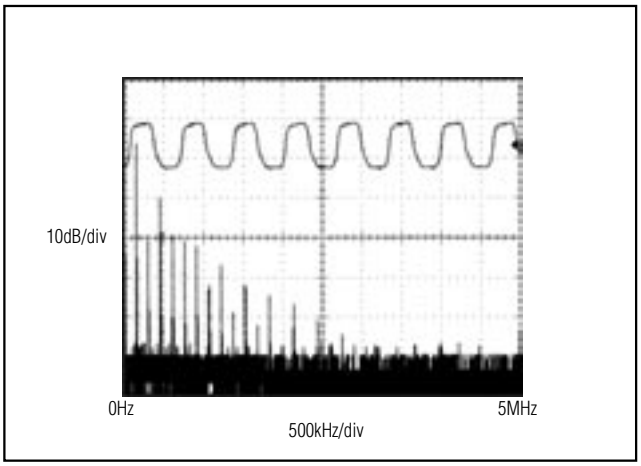


Figure 13. Driver Output Waveform and FFT Plot of MAX483/ MAX487–MAX489 Transmitting a 150kHz Signal

Low-Power Shutdown Mode (MAX481/MAX483/MAX487)

A low-power shutdown mode is initiated by bringing both $\overline{RE}$ high and DE low. The devices will not shut down unless both the driver and receiver are disabled. In shutdown, the devices typically draw only 0.1µA of supply current.

$\overline{RE}$ and DE may be driven simultaneously; the parts are guaranteed not to enter shutdown if $\overline{RE}$ is high and DE is low for less than 50ns. If the inputs are in this state for at least 600ns, the parts are guaranteed to enter shutdown.

For the MAX481, MAX483, and MAX487, the t_{ZH} and t_{ZL} enable times assume the part was not in the low-power shutdown state (the MAX485/MAX488–MAX491 and MAX1487 can not be shut down). The $t_{ZH(SHDN)}$ and $t_{ZL(SHDN)}$ enable times assume the parts were shut down (see *Electrical Characteristics*).

It takes the drivers and receivers longer to become enabled from the low-power shutdown state ($t_{ZH(SHDN)}$, $t_{ZL(SHDN)}$) than from the operating mode (t_{ZH}, t_{ZL}). (The parts are in operating mode if the $\overline{RE}$, DE inputs equal a logical 0,1 or 1,1 or 0, 0.)

Driver Output Protection

Excessive output current and power dissipation caused by faults or by bus contention are prevented by two mechanisms. A foldback current limit on the output stage provides immediate protection against short circuits over the whole common-mode voltage range (see *Typical Operating Characteristics*). In addition, a thermal shutdown circuit forces the driver outputs into a high-impedance state if the die temperature rises excessively.

Propagation Delay

Many digital encoding schemes depend on the difference between the driver and receiver propagation delay times. Typical propagation delays are shown in Figures 15–18 using Figure 14's test circuit.

The difference in receiver delay times, $|t_{PLH} - t_{PHL}|$, is typically under 13ns for the MAX481, MAX485, MAX490, MAX491, and MAX1487 and is typically less than 100ns for the MAX483 and MAX487–MAX489.

The driver skew times are typically 5ns (10ns max) for the MAX481, MAX485, MAX490, MAX491, and MAX1487, and are typically 100ns (800ns max) for the MAX483 and MAX487–MAX489.

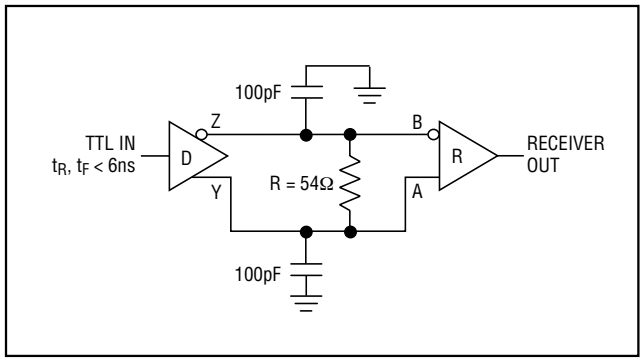


Figure 14. Receiver Propagation Delay Test Circuit

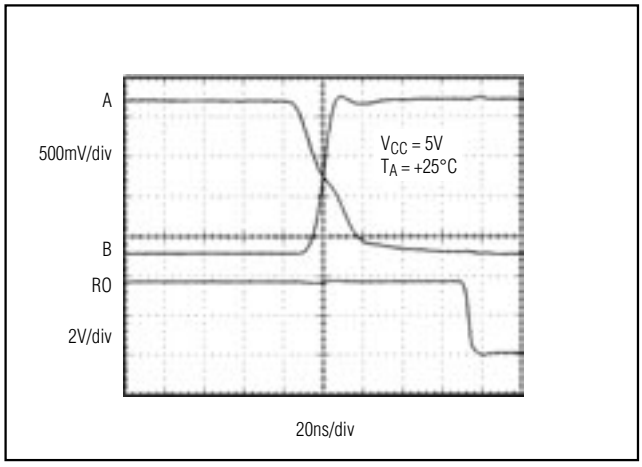


Figure 15. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PHL}

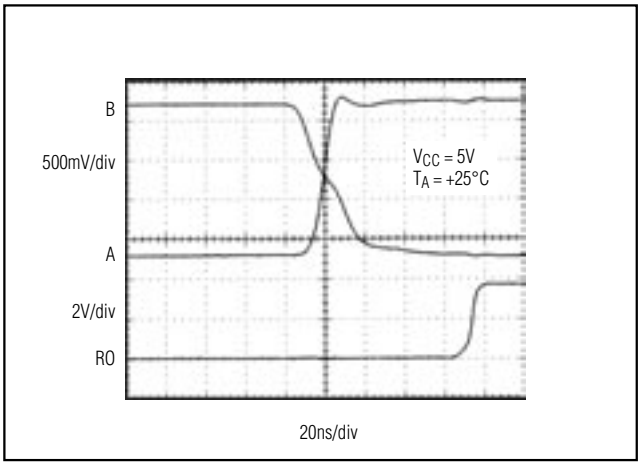


Figure 16. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver t_{PLH}

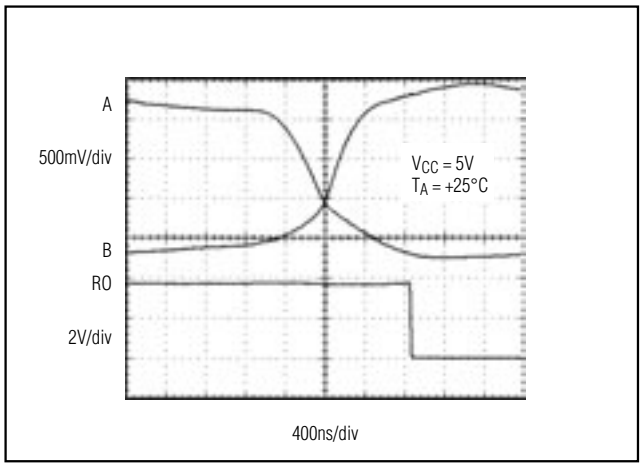


Figure 17. MAX483, MAX487–MAX489 Receiver t_{PHL}

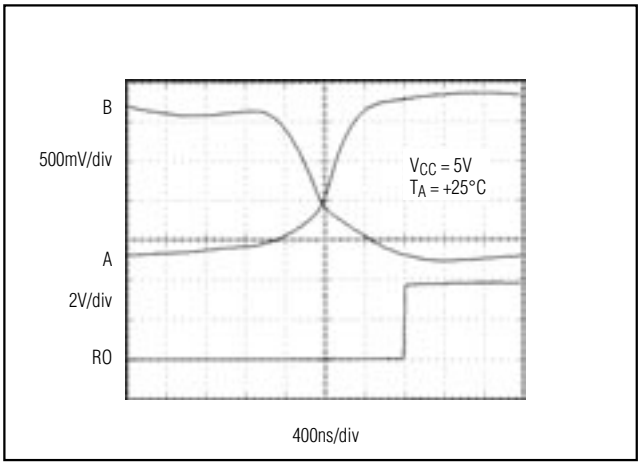


Figure 18. MAX483, MAX487–MAX489 Receiver t_{PLH}

Line Length vs. Data Rate

The RS-485/RS-422 standard covers line lengths up to 4000 feet. For line lengths greater than 4000 feet, see Figure 23.

Figures 19 and 20 show the system differential voltage for the parts driving 4000 feet of 26AWG twisted-pair wire at 110kHz into 120Ω loads.

Typical Applications

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 transceivers are designed for bidirectional data communications on multipoint bus transmission lines. Figures 21 and 22 show typical network applications circuits. These parts can also be used as line repeaters, with cable lengths longer than 4000 feet, as shown in Figure 23.

To minimize reflections, the line should be terminated at both ends in its characteristic impedance, and stub lengths off the main line should be kept as short as possible. The slew-rate-limited MAX483 and MAX487–MAX489 are more tolerant of imperfect termination.

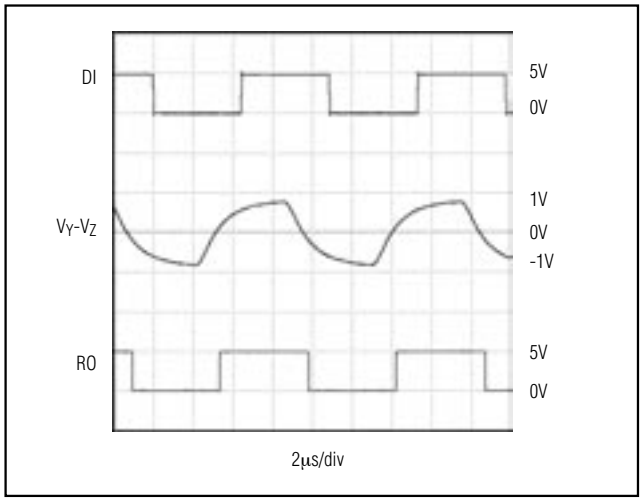


Figure 19. MAX481/MAX485/MAX490/MAX491/MAX1487 System Differential Voltage at 110kHz Driving 4000ft of Cable

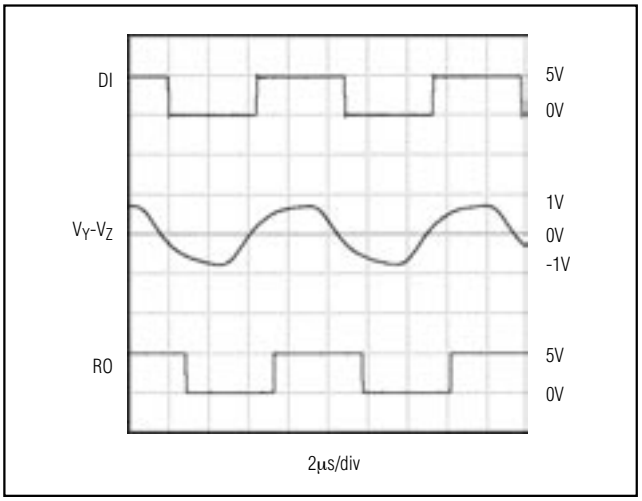


Figure 20. MAX483, MAX487–MAX489 System Differential Voltage at 110kHz Driving 4000ft of Cable

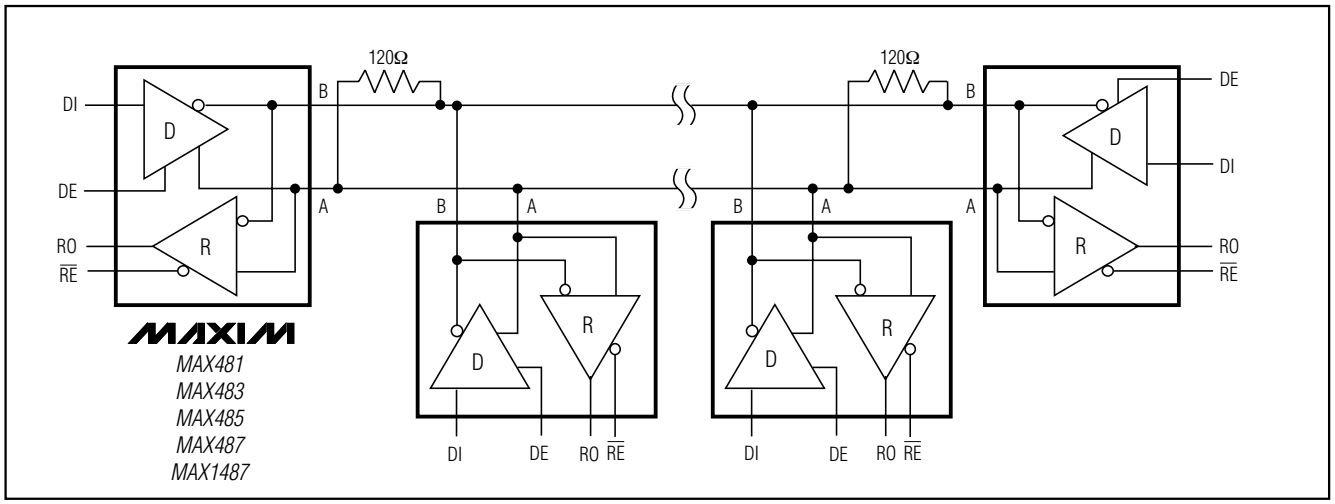


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

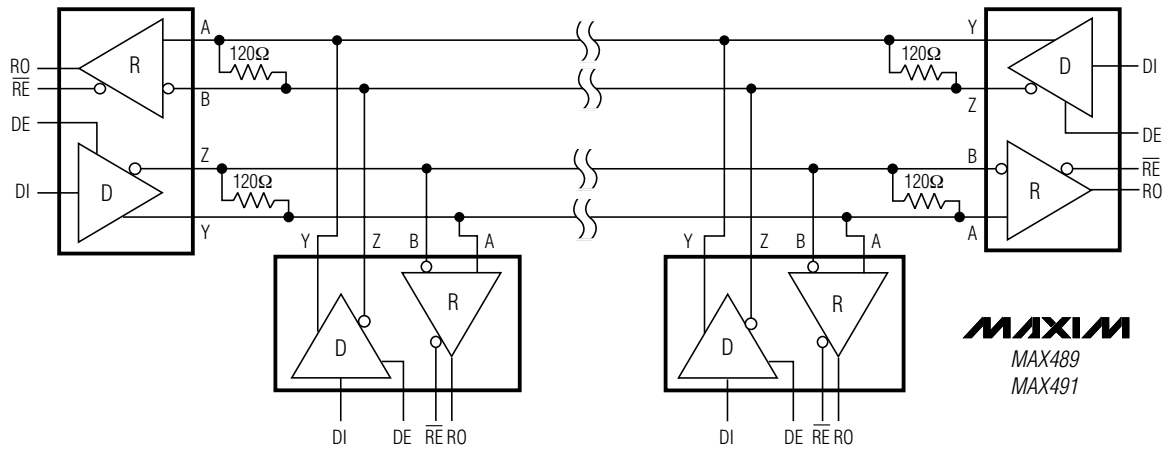


Figure 22. MAX488–MAX491 Full-Duplex RS-485 Network

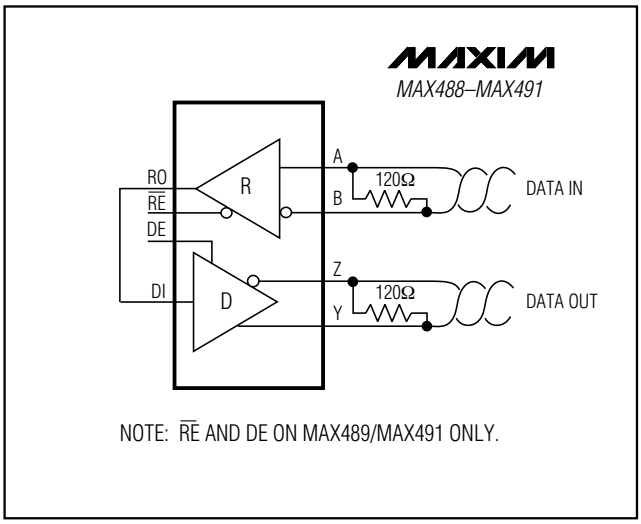


Figure 23. Line Repeater for MAX488–MAX491

Isolated RS-485

For isolated RS-485 applications, see the MAX253 and MAX1480 data sheets.

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX481CPA	0°C to +70°C	8 Plastic DIP
MAX481CSA	0°C to +70°C	8 SO
MAX481CUA	0°C to +70°C	8 µMAX
MAX481C/D	0°C to +70°C	Dice*
MAX481EPA	-40°C to +85°C	8 Plastic DIP
MAX481ESA	-40°C to +85°C	8 SO
MAX481MJA	-55°C to +125°C	8 CERDIP
MAX483CPA	0°C to +70°C	8 Plastic DIP
MAX483CSA	0°C to +70°C	8 SO
MAX483CUA	0°C to +70°C	8 µMAX
MAX483C/D	0°C to +70°C	Dice*
MAX483EPA	-40°C to +85°C	8 Plastic DIP
MAX483ESA	-40°C to +85°C	8 SO
MAX483MJA	-55°C to +125°C	8 CERDIP
MAX485CPA	0°C to +70°C	8 Plastic DIP
MAX485CSA	0°C to +70°C	8 SO
MAX485CUA	0°C to +70°C	8 µMAX
MAX485C/D	0°C to +70°C	Dice*
MAX485EPA	-40°C to +85°C	8 Plastic DIP
MAX485ESA	-40°C to +85°C	8 SO
MAX485MJA	-55°C to +125°C	8 CERDIP
MAX487CPA	0°C to +70°C	8 Plastic DIP
MAX487CSA	0°C to +70°C	8 SO
MAX487CUA	0°C to +70°C	8 µMAX
MAX487C/D	0°C to +70°C	Dice*
MAX487EPA	-40°C to +85°C	8 Plastic DIP
MAX487ESA	-40°C to +85°C	8 SO
MAX487MJA	-55°C to +125°C	8 CERDIP
MAX488CPA	0°C to +70°C	8 Plastic DIP
MAX488CSA	0°C to +70°C	8 SO
MAX488CUA	0°C to +70°C	8 µMAX
MAX488C/D	0°C to +70°C	Dice*
MAX488EPA	-40°C to +85°C	8 Plastic DIP
MAX488ESA	-40°C to +85°C	8 SO
MAX488MJA	-55°C to +125°C	8 CERDIP
MAX489CPD	0°C to +70°C	14 Plastic DIP
MAX489CSD	0°C to +70°C	14 SO
MAX489C/D	0°C to +70°C	Dice*
MAX489EPD	-40°C to +85°C	14 Plastic DIP
MAX489ESD	-40°C to +85°C	14 SO
MAX489MJD	-55°C to +125°C	14 CERDIP

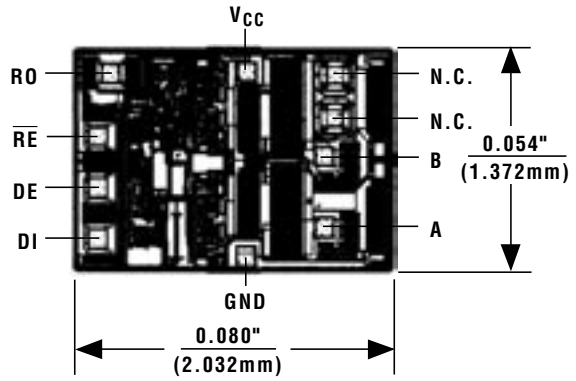
Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX490CPA	0°C to +70°C	8 Plastic DIP
MAX490CSA	0°C to +70°C	8 SO
MAX490CUA	0°C to +70°C	8 µMAX
MAX490C/D	0°C to +70°C	Dice*
MAX490EPA	-40°C to +85°C	8 Plastic DIP
MAX490ESA	-40°C to +85°C	8 SO
MAX490MJA	-55°C to +125°C	8 CERDIP
MAX491CPD	0°C to +70°C	14 Plastic DIP
MAX491CSD	0°C to +70°C	14 SO
MAX491C/D	0°C to +70°C	Dice*
MAX491EPD	-40°C to +85°C	14 Plastic DIP
MAX491ESD	-40°C to +85°C	14 SO
MAX491MJD	-55°C to +125°C	14 CERDIP
MAX1487CPA	0°C to +70°C	8 Plastic DIP
MAX1487CSA	0°C to +70°C	8 SO
MAX1487CUA	0°C to +70°C	8 µMAX
MAX1487C/D	0°C to +70°C	Dice*
MAX1487EPA	-40°C to +85°C	8 Plastic DIP
MAX1487ESA	-40°C to +85°C	8 SO
MAX1487MJA	-55°C to +125°C	8 CERDIP

** Contact factory for dice specifications.*

Chip Topographies

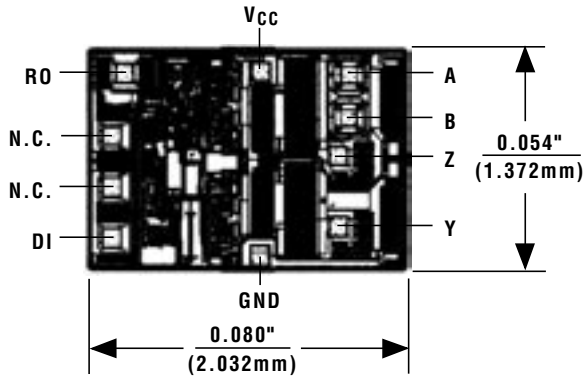
MAX481/MAX483/MAX485/MAX487/MAX1487



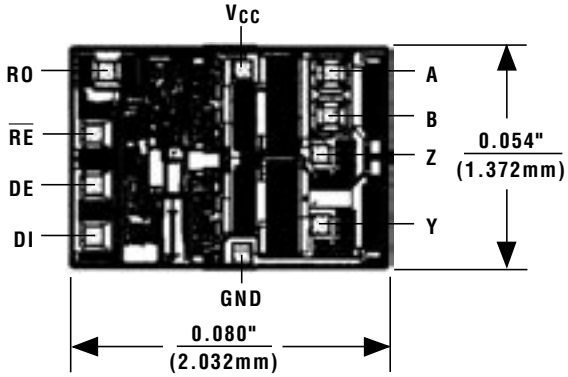
Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Chip Topographies (continued)

MAX488/MAX490



MAX489/MAX491



TRANSISTOR COUNT: 248

SUBSTRATE CONNECTED TO GND

Package Information

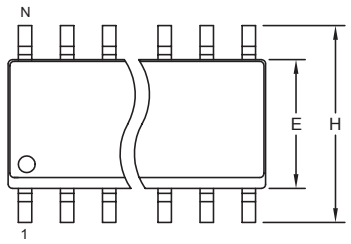
(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to **www.maxim-ic.com/packages**.)

SOICN .EPS

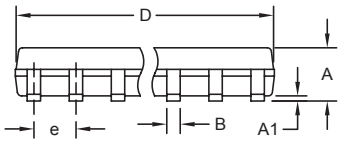
	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.053	0.069	1.35	1.75
A1	0.004	0.010	0.10	0.25
B	0.014	0.019	0.35	0.49
C	0.007	0.010	0.19	0.25
e	0.050 BSC		1.27 BSC	
E	0.150	0.157	3.80	4.00
H	0.228	0.244	5.80	6.20
L	0.016	0.050	0.40	1.27

VARIATIONS:

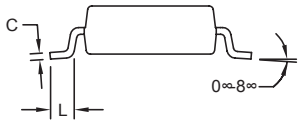
	INCHES		MILLIMETERS			
DIM	MIN	MAX	MIN	MAX	N	MS012
D	0.189	0.197	4.80	5.00	8	AA
D	0.337	0.344	8.55	8.75	14	AB
D	0.386	0.394	9.80	10.00	16	AC



TOP VIEW



FRONT VIEW



SIDE VIEW

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED 0.15mm (.006").
3. LEADS TO BE COPLANAR WITHIN 0.10mm (.004").
4. CONTROLLING DIMENSION: MILLIMETERS.
5. MEETS JEDEC MS012.
6. N = NUMBER OF PINS.

DALLAS SEMICONDUCTOR MAXIM

PROPRIETARY INFORMATION

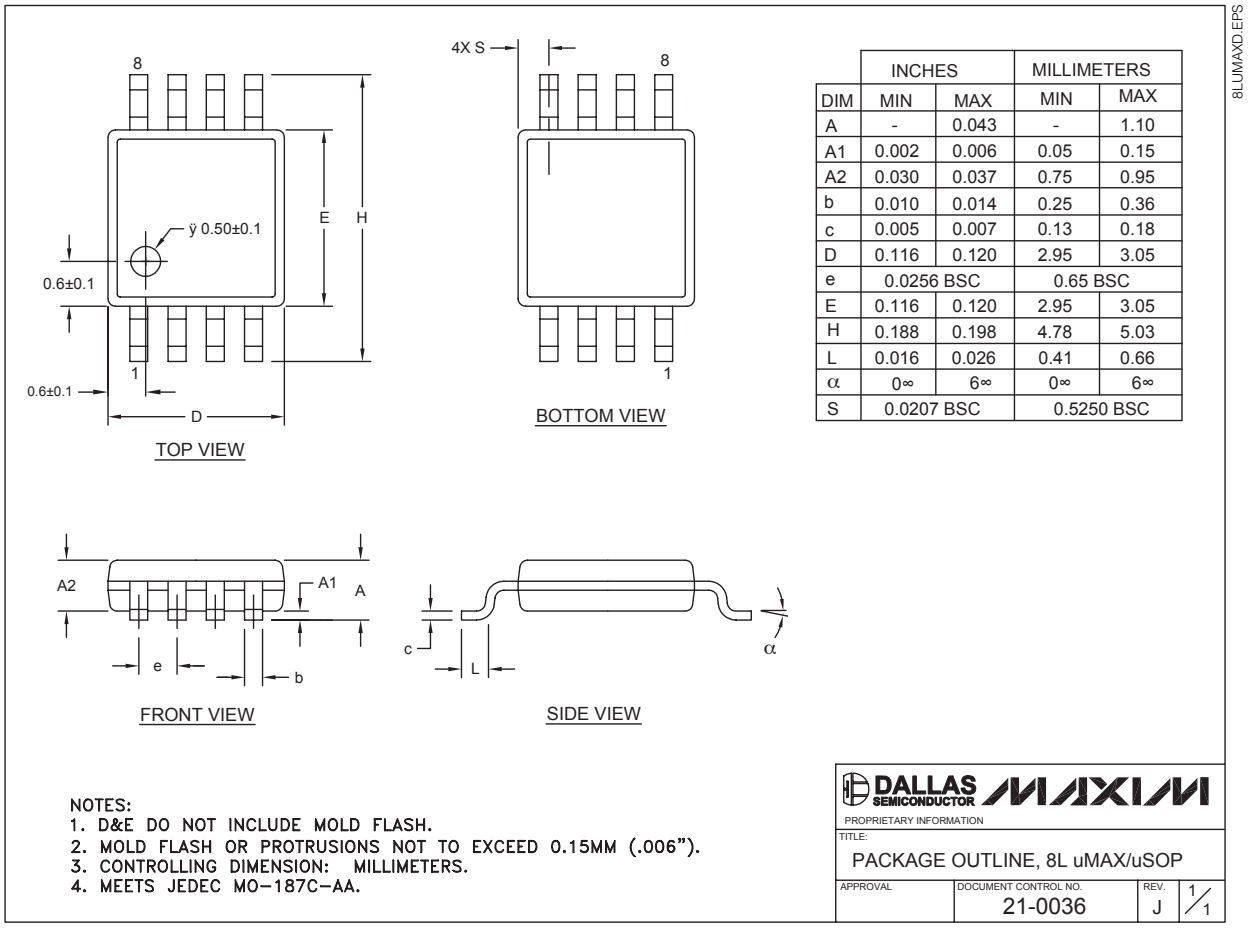
TITLE: PACKAGE OUTLINE, .150" SOIC

APPROVAL	DOCUMENT CONTROL NO.	REV.	
	21-0041	B	1/1

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to **www.maxim-ic.com/packages**.)

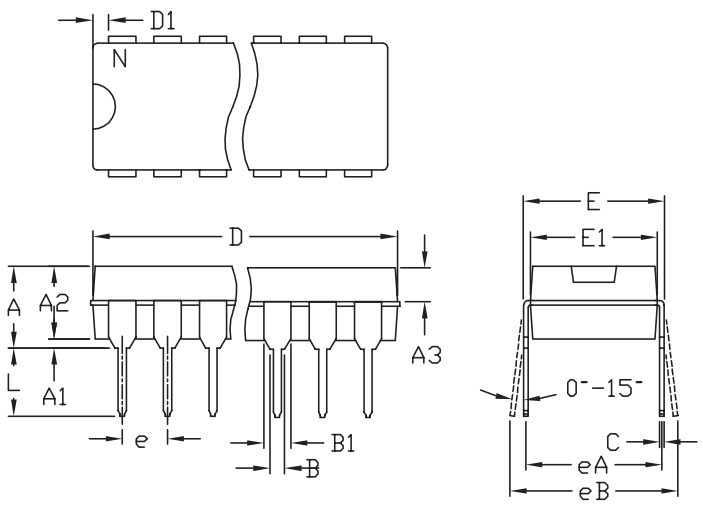


	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	-	0.043	-	1.10
A1	0.002	0.006	0.05	0.15
A2	0.030	0.037	0.75	0.95
b	0.010	0.014	0.25	0.36
c	0.005	0.007	0.13	0.18
D	0.116	0.120	2.95	3.05
e	0.0256 BSC		0.65 BSC	
E	0.116	0.120	2.95	3.05
H	0.188	0.198	4.78	5.03
L	0.016	0.026	0.41	0.66
α	0∞	6∞	0∞	6∞
S	0.0207 BSC		0.5250 BSC	

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to **www.maxim-ic.com/packages**.)

PDIPN.EPS



	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	---	0.180	---	4.572
A1	0.015	---	0.38	---
A2	0.125	0.175	3.18	4.45
A3	0.055	0.080	1.40	2.03
B	0.015	0.022	0.381	0.56
B1	0.045	0.065	1.14	1.65
C	0.008	0.014	0.2	0.355
D1	0.005	0.080	0.13	2.03
E	0.300	0.325	7.62	8.26
E1	0.240	0.310	6.10	7.87
e	0.100 BSC.		2.54 BSC.	
eA	0.300 BSC.		7.62 BSC.	
eB	0.400 BSC.		10.16 BSC.	
L	0.115	0.150	2.921	3.81

	INCHES		MILLIMETERS			
	MIN	MAX	MIN	MAX	N	MS001
D	0.348	0.390	8.84	9.91	8	AB
D	0.735	0.765	18.67	19.43	14	AC
D	0.745	0.765	18.92	19.43	16	AA
D	0.885	0.915	22.48	23.24	18	AD
D	1.015	1.045	25.78	26.54	20	AE
D	1.14	1.265	28.96	32.13	24	AF
D	1.360	1.380	34.54	35.05	28	*5

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED .15mm (.006")
3. CONTROLLING DIMENSION: MILLIMETER
4. MEETS JEDEC MS001-XX AS SHOWN IN ABOVE TABLE
5. SIMILIAR TO JEDEC MO-058AB
6. N = NUMBER OF PINS

MAXIM 120 SAN GABRIEL DR SUNNYVALE CA 94086 FAX (408) 737 7194
PROPRIETARY INFORMATION

PACKAGE FAMILY OUTLINE: PDIP .300"
TITLE

1/1

21-0043 D
DOCUMENT CONTROL NUMBER REV

1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

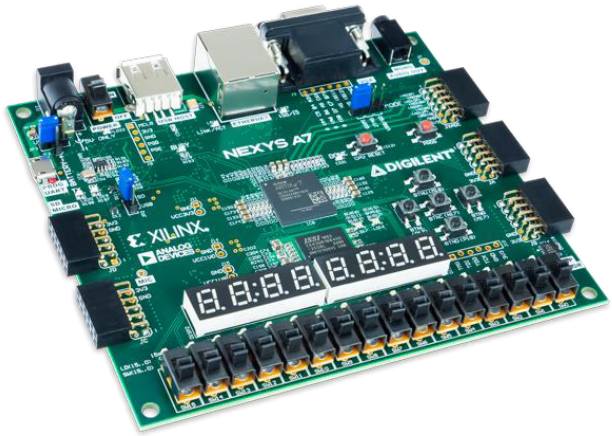
Nexys A7™ FPGA Board Reference Manual

Revised July 10, 2019

Table of Contents

Features

The Nexys A7 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx®. With its large, high-capacity FPGA, generous external memories, and collection of USB, Ethernet, and other ports, the Nexys A7 can host designs ranging from introductory combinational circuits to powerful embedded processors. Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and several I/O devices allow the Nexys A7 to be used for a wide range of designs without needing any other components.



The Nexys A7 FPGA.

- **Artux-7 FPGA**
  - 15,850 Programmable logic slices, each with four 6-input LUTs and 8 flip-flops (*8,150 slices)
  - 1,188 Kbits of fast block RAM (*600 Kbits)
  - Six clock management tiles, each with phase-locked loop (PLL)
  - 240 DSP slices (*120 DSPs)
  - Internal clock speeds exceeding 450 MHz
  - Dual-channel, 1 MSPS internal analog-digital converter (XADC)
- **Memory**
  - 128MiB DDR2
  - Serial Flash
  - microSD card slot
- **Power**
  - Powered from USB or any 4.5V-5.5V external power source
- **USB and Ethernet**
  - 10/100 Ethernet PHY
  - USB-JTAG programming circuitry
  - USB-UART bridge
  - USB HID Host for mice, keyboards and memory sticks
- **Simple User Input/**Output
  - 16 Switches
  - 16 LEDs
  - Two RGB LEDs
  - Two 4-digit 7-segment displays
- **Audio and Video**
  - 12-bit VGA output
  - PWM audio output
  - PDM microphone
- **Additional Sensors**
  - 3-axis accelerometer
  - Temperature sensor
- **Expansion Connectors**
  - Pmod connector for XADC signals
  - Four Pmod connectors providing 32 total FPGA I/O

The Nexys A7-100T is compatible with Xilinx's Vivado® Design Suite as well as the ISE® toolset, which includes ChipScope™ and EDK. Xilinx ISE has been discontinued in favor of Vivado® Design Suite.

The Nexys A7-50T variant is compatible only with Vivado® Design Suite.

Xilinx offers free WebPACK™ versions of these toolsets, so designs can be implemented at no additional cost.

The Nexys A7 is not supported by the Digilent Adept Utility.

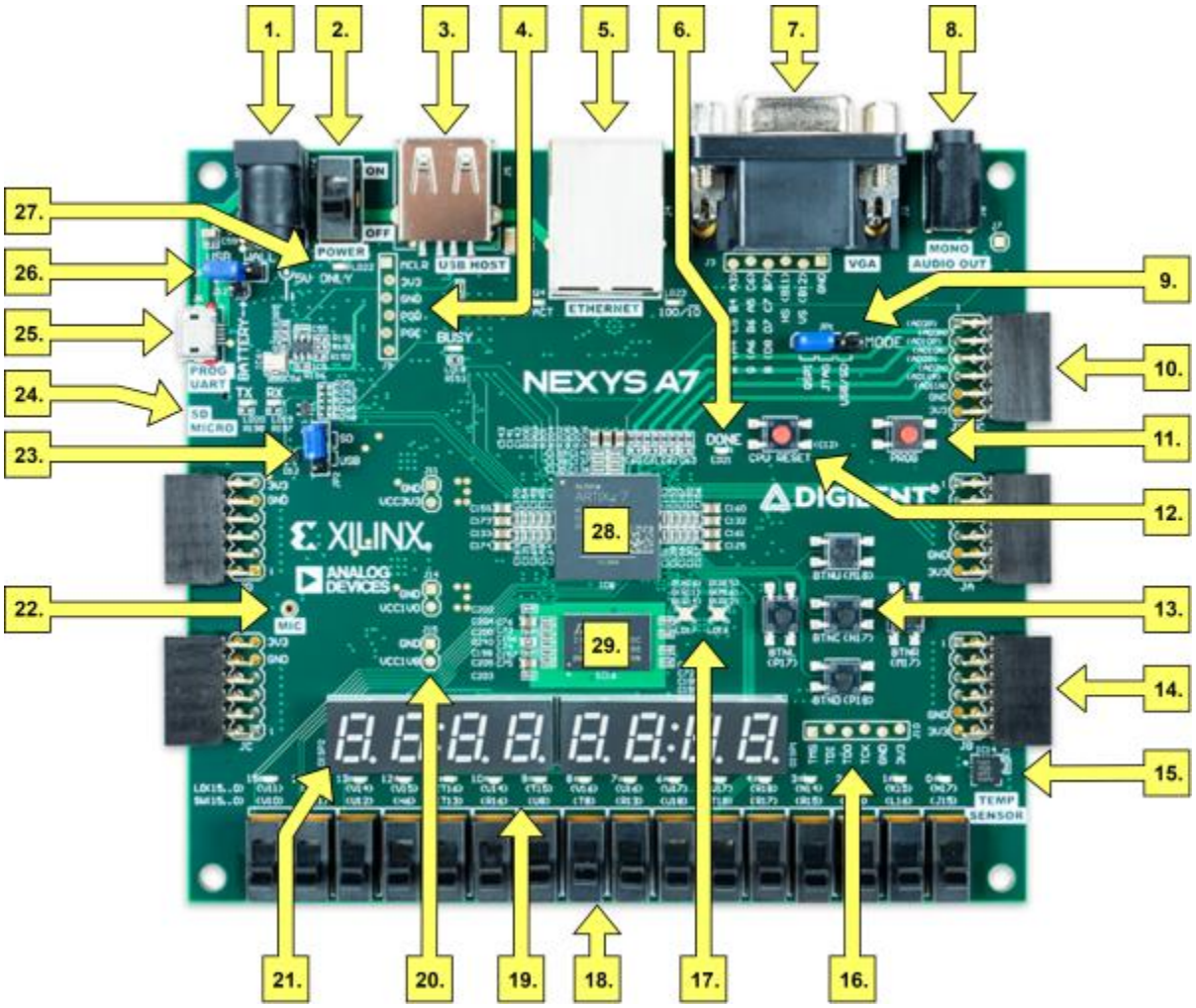


Figure 1. Nexys A7 Feature Callout.

Callout	Component Description	Callout	Component Description
1	Power jack	16	JTAG port for (optional) external cable
2	Power switch	17	Tri-color (RGB) LEDs
3	USB host connector	18	Slide switches (16)
4	PIC24 programming port (factory use)	19	LEDs (16)
5	Ethernet connector	20	Power supply test point(s)
6	FPGA programming done LED	21	Eight digit 7-seg display
7	VGA connector	22	Microphone
8	Audio connector	23	External configuration jumper (SD / USB)
9	Programming mode jumper	24	MicroSD card slot
10	Analog signal Pmod port (XADC)	25	Shared UART/ JTAG USB port

11	FPGA configuration reset button	**26**	Power select jumper and battery header
12	CPU reset button (for soft cores)	**27**	Power-good LED
13	Five pushbuttons	**28**	Xilinx Artix-7 FPGA
14	Pmod port(s)	**29**	DDR2 memory
15	Temperature sensor		

Purchasing Options

The Nexys A7 can be purchased with either a XC7A100T or XC7A50T FPGA loaded. These two Nexys A7 product variants are referred to as the Nexys A7-100T and Nexys A7-50T, respectively. When Digilent documentation describes functionality that is common to both of these variants, they are referred to collectively as the "Nexys A7". When describing something that is only common to a specific variant, the variant will be explicitly called out by its name.

The only difference between the Nexys A7-100T and Nexys A7-50T is the size of the Artix-7 part. The Artix-7 FPGAs both have the same capabilities, but the XC7100T has about a 2 times larger internal FPGA than the XC750T. The differences between the two variants are summarized below:

Product Variant	Nexys A7-100T	Nexys A7-50T
FPGA Part Number	XC7A100T-1CSG324C	XC7A50T-1CSG324I
Look-up Tables (LUTs)	63,400	32,600
Flip-Flops	126,800	65,200
Block RAM	1,188 Kb	600 Kb
DSP Slices	240	120
Clock Management Tiles	6	5

Board Revisions

The Nexys A7 is a rebrand of the Nexys 4 DDR board, which is an incremental update to the Nexys 4 board.

Migrating from Nexys 4 DDR

The only difference between the Nexys A7 and Nexys 4 DDR is the addition of the Nexys A7-50T variant of the Nexys A7, which has a smaller gate array. The Nexys A7-100T variant is functionally identical to the Nexys 4 DDR.

Users of the Nexys A7 may find resources produced for the Nexys 4 DDR helpful, which can be found at the Nexys 4 DDR's Resource Center.

Migrating from Nexys 4

The major improvement from the Nexys 4 to the Nexys 4 DDR is the replacement of the 16 MiB Cellular RAM with a 128 MiB DDR2 SDRAM memory. Furthermore, to accommodate the new memory, the pin-out of the FPGA banks changed as well.

The audio output (AUD_PWM) needs to be driven open-drain as opposed to push-pull on the Nexys 4.

1 Functional Description

1.1 Power Supplies

The Nexys A7 board can receive power from the Digilent USB-JTAG port (J6) or from an external power supply. Jumper JP3 (near the power jack) determines which source is used.

All Nexys A7 power supplies can be turned on and off by a single logic-level power switch (SW16). A power-good LED (LD22), driven by the "power good" output of the ADP2118 supply, indicates that the supplies are turned on and operating normally. An overview of the Nexys A7 power circuit is shown in Figure 1.1.

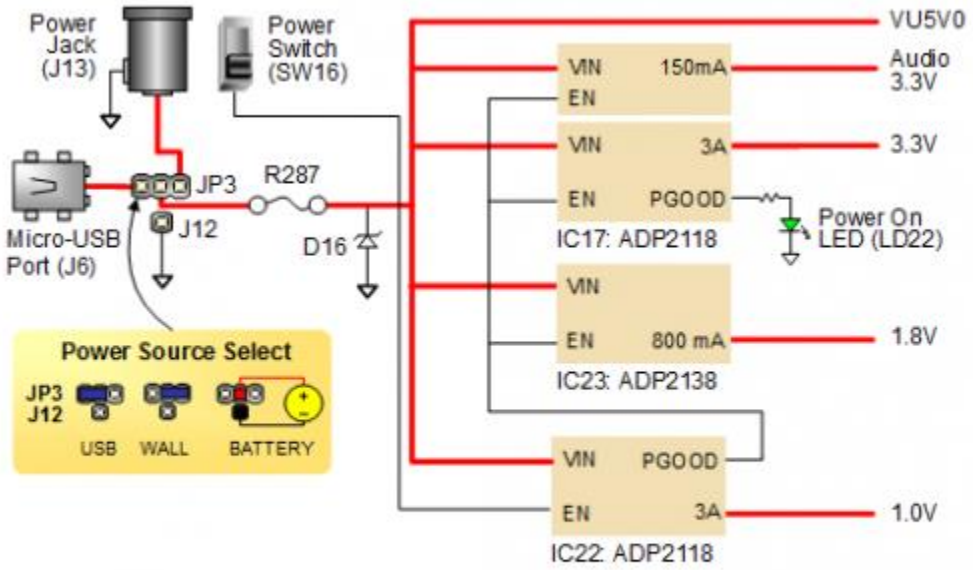


Figure 1.1 Nexys A7 Power Circuit

The USB port can deliver enough power for the vast majority of designs. In order to power the board from USB port set jumper JP3 to "USB". Our out-of-box demo draws ~400mA of current from the 5V input rail. A few demanding applications, including any that drive multiple peripheral boards, might require more power than the USB port can provide. Also, some applications may need to run without being connected to a PC's USB port. In these instances, an external power supply or battery pack can be used.

An external power supply can be used by plugging into to the power jack (J13) and setting jumper JP3 to "WALL". The supply must use a coax, center-positive 2.1mm internal-diameter plug, and deliver 4.5VDC to 5.5VDC and at least 1A of current (i.e., at least 5W of power). Many suitable supplies can be purchased from Digilent, through Digi-Key, or other catalog vendors.

An external battery pack can be used by connecting the battery's positive terminal to the center pin of JP3 and the negative terminal to the pin labeled J12, directly below JP3. Since the main regulator on the Nexys A7 cannot accommodate input voltages over 5.5VDC, an external battery pack must be limited to 5.5VDC. The minimum voltage of the battery pack depends on the application: if the USB Host function (J5) is used, at least 4.6V needs to be provided. In other cases, the minimum voltage is 3.6V.

Voltage regulator circuits from Analog Devices create the required 3.3V, 1.8V, and 1.0V supplies from the main power input. Table 1.1 provides additional information. Typical currents depend strongly on FPGA configuration and the values provided are typical of medium size/speed designs.

Supply	Circuits	Device	Current (Max/Typical)
3.3V	FPGA I/O, USB ports, Clocks, RAM I/O, Ethernet, SD slot, Sensors, Flash	IC17: ADP2118	3A/0.1 to 1.5A
1.0V	FPGA Core	IC22: ADP2118	3A/ 0.2 to 1.3A
1.8V	DDR2, FPGA Auxiliary and RAM	IC23: ADP2118	0.8A/ 0.5A

Table 1.1 Nexys A7 power supplies.

1.2 Protection

The Nexys A7 features overcurrent and overvoltage protection on the input power rail. A 3.5A fuse (R287) and a 5V Zener diode (D16) provide a non-resettable protection for other on-board integrated circuits, as displayed in Figure 2. Applying power outside of the specs outlined in this document is not covered by warranty. If this happens, either or both might get permanently damaged. The damaged parts are not user replaceable.

2 FPGA Configuration

After power-on, the Artix-7 FPGA must be configured (or programmed) before it can perform any functions. You can configure the FPGA in one of four ways:

1. A PC can use the Digilent USB-JTAG circuitry (portJ6, labeled "PROG") to program the FPGA any time the power is on.
2. A file stored in the nonvolatile serial (SPI) flash device can be transferred to the FPGA using the SPI port.
3. A programming file can be transferred to the FPGA from a micro SD card.
4. A programming file can be transferred from a USB memory stick attached to the USB HID port.

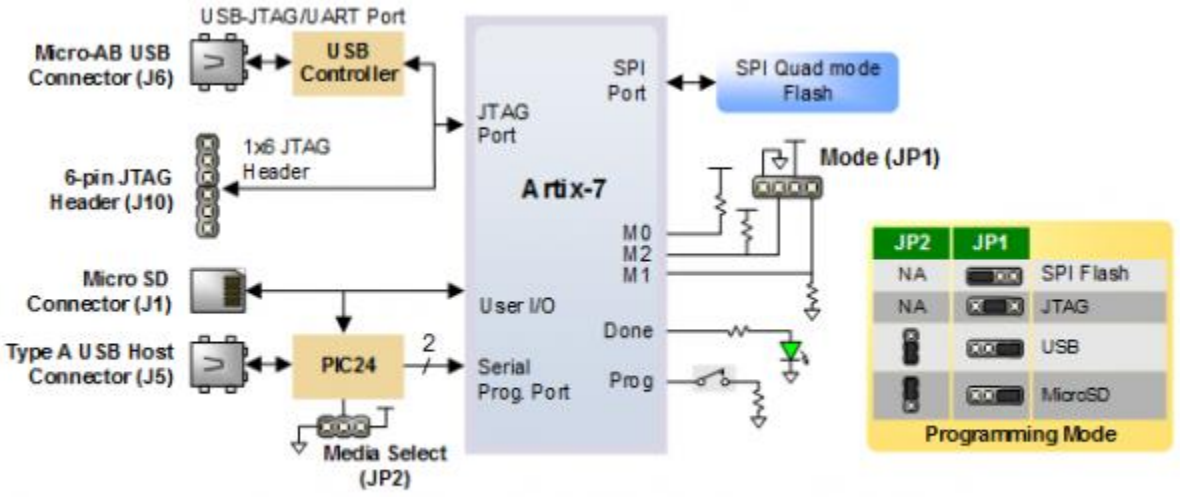


Figure 2.1 Nexys A7 DDR Configuration Options.

Figure 2.1 shows the different options available for configuring the FPGA. An on-board "mode" jumper (JP1) and a media selection jumper (JP2) select between the programming modes.

The FPGA configuration data is stored in files called bitstreams that have the .bit file extension. The ISE or Vivado software from Xilinx can create bitstreams from VHDL, Verilog®, or schematic-based source files (in the ISE toolset, EDK is used for MicroBlaze™ embedded processor-based designs).

Bitstreams are stored in SRAM-based memory cells within the FPGA. This data defines the FPGA's logic functions and circuit connections, and it remains valid until it is erased by removing board power, by pressing the reset button attached to the PROG input, or by writing a new configuration file using the JTAG port.

An Artix-7 100T bitstream is typically 30,606,304 bits and can take a long time to transfer. The time it takes to program the Nexys A7 can be decreased by compressing the bitstream before programming, and then allowing the FPGA to decompress the bitstream itself during configuration. Depending on design complexity, compression ratios of 10x can be achieved. Bitstream compression can be enabled within the Xilinx tools (ISE or Vivado) to occur during generation. For instructions on how to do this, consult the Xilinx documentation for the toolset being used. After being successfully programmed, the FPGA will cause the "DONE" LED to illuminate. Pressing the "PROG" button at any time will reset the configuration memory in the FPGA. After being reset, the FPGA will immediately attempt to reprogram itself from whatever method has been selected by the programming mode jumpers.

The following sections provide greater detail about programming the Nexys A7 using the different methods available.

2.1 JTAG Configuraiton

The Xilinx tools typically communicate with FPGAs using the Test Access Port and Boundary-Scan Architecture, commonly referred to as JTAG. During JTAG programming, a .bit file is transferred from the PC to the FPGA using the onboard Digilent USB-JTAG circuitry (port J6) or an external JTAG programmer, such as the Digilent JTAG-HS2, attached to port J10. You can perform JTAG programming any time after the Nexys A7 has been powered on, regardless of what the mode jumper (JP1) is set to. If the FPGA is already configured, then the existing configuration is overwritten with the bitstream being transmitted over JTAG. Setting the mode jumper to the JTAG setting (seen in Figure 3) is useful to prevent the FPGA from being configured from any other bitstream source until a JTAG programming occurs.

Programming the Nexys A7 with an uncompressed bitstream using the on-board USB-JTAG circuitry usually takes around five seconds. JTAG programming can be done using the hardware server in Vivado or the iMPACT tool included with ISE and the Lab Tools version of Vivado. The demonstration project available at http://www.digilentinc.com/ gives an in-depth tutorial on how to program your board.

2.2 Quad-SPI Configuration

Since the FPGA on the Nexys A7 is volatile, it relies on the Quad-SPI flash memory to store the configuration between power cycles. This configuration mode is called Master SPI. The blank FPGA takes the role of master and reads the configuration file out of the flash device upon power-up. To that effect, a configuration file needs to be downloaded first to the flash. When programming a nonvolatile flash device, a bitstream file is transferred to the flash in a two-step process. First, the FPGA is programmed with a circuit that can program flash devices, and then data is transferred to the flash device via the FPGA circuit (this complexity is hidden from the user by the Xilinx tools). This is called indirect programming. After the flash device has been programmed, it can automatically configure the FPGA at a subsequent power-on or reset event as determined by the mode jumper setting (see Figure 3). Programming files stored in the flash device will remain until they are overwritten, regardless of power-cycle events.

Programming the flash can take as long as four to five minutes, which is mostly due to the lengthy erase process inherent to the memory technology. Once written however, FPGA configuration can be very fast—less than a second. Bitstream compression, SPI bus width, and configuration rate are factors controlled by the Xilinx tools that can affect configuration speed. The Nexys A7 supports x1, x2, and x4 bus widths and data rates of up to 50 MHz for Quad-SPI programming.

Quad-SPI programming can be done using the iMPACT tool included with ISE or the Lab Tools version of Vivado.

2.3 USB Host and Micro SD Programming

You can program the FPGA from a pen drive attached to the USB Host port (J5) or a microSD card inserted into J1 by doing the following:

1. Format the storage device (Pen drive or microSD card) with a FAT32 file system.
2. Place a single .bit configuration file in the root directory of the storage device.
3. Attach the storage device to the Nexys A7.
4. Set the JP1 Programming Mode jumper on the Nexys A7 to “USB/SD”.
5. Select the desired storage device using JP2.
6. Push the PROG button or power-cycle the Nexys A7.

The FPGA will automatically configure with the .bit file on the selected storage device. Any .bit files that are not built for the proper Artix-7 device will be rejected by the FPGA.

The Auxiliary Function Status, or “BUSY” LED, gives visual feedback on the state of the configuration process when the FPGA is not yet programmed:

- When steadily lit, the auxiliary microcontroller is either booting up or currently reading the configuration medium (microSD or pen drive) and downloading a bitstream to the FPGA.
- A slow pulse means the microcontroller is waiting for a configuration medium to be plugged in.
- In case of an error during configuration, the LED will blink rapidly.

When the FPGA has been successfully configured, the behavior of the LED is application-specific. For example, if a USB keyboard is plugged in, a rapid blink will signal the receipt of an HID input report from the keyboard.

3 Memory

The Nexys A7 board contains two external memories: a 1Gib (128MiB) DDR2 SDRAM and a 128Mib (16MiB) non-volatile serial Flash device. The DDR2 modules are integrated on-board and connect to the FPGA using the industry standard interface. The serial Flash is on a dedicated quad-mode (x4) SPI bus. The connections and pin assignments between the FPGA and external memories are shown below.

3.1 DDR2

The Nexys A7 includes one Micron MT47H64M16HR-25:H DDR2 memory component, creating a single rank, 16-bit wide interface. It is routed to a 1.8V-powered HR (High Range) FPGA bank with 50 ohm controlled single-ended trace impedance. 50-ohm internal terminations in the FPGA are used to match the trace characteristics. Similarly, on the memory side, on-die terminations (ODT) are used for impedance matching.

For proper operation of the memory, a memory controller and physical layer (PHY) interface needs to be included in the FPGA design. There are two recommended ways to do that, which are outlined below and differ in complexity and design flexibility.

The straightforward way is to use the Digilent-provided DDR-to-SRAM adapter module which instantiates the memory controller and uses an asynchronous SRAM bus for interfacing with user logic. This module provides backward compatibility with projects written for older Nexys-line boards featuring a CellularRAM instead of DDR2. It trades memory bandwidth for simplicity.

More advanced users or those who wish to learn more about DDR SDRAM technology may want to use the Xilinx 7-series memory interface solutions core generated by the MIG (Memory Interface Generator) Wizard. Depending on the tool used (ISE, EDK or Vivado), the MIG Wizard can generate a native FIFO-style or an AXI4 interface to connect to user logic. This workflow allows the customization of several DDR parameters optimized for the particular application. Table 3.1 below lists the MIG Wizard settings optimized for the Nexys A7.

Setting	Value
Memory type	DDR2 SDRAM
Max. clock period	3000ps (667Mbps data rate)
Recommended clock period (for easy clock generation)	3077ps (650Mbps data rate)
Memory part	MT47H64M16HR-25E
Data width	16
Data mask	Enabled
Chip Select pin	Enabled
Rtt (nominal) – On-die termination	50ohms
Internal Vref	Enabled

Table 3.1.1 DDR2 settings for the Nexys A7.

Although the FPGA, memory IC, and the board itself are capable of the maximum data rate of 667Mbps, the limitations in the clock generation primitives restrict the clock frequencies that can be generated from the 100 MHz system clock. Thus, for simplicity, the next highest data rate of 650Mbps is recommended.

The MIG Wizard will require the fixed pin-out of the memory signals to be entered and validated before generating the IP core. For your convenience, an importable UCF file is provided on the Digilent website to speed up the process.

For more details on the Xilinx memory interface solutions, refer to the 7 Series FPGAs Memory Interface Solutions User Guide (ug586)[i].

3.2 Quad-SPI Flash

FPGA configuration files can be written to the Quad-SPI Flash (Spansion part number S25FL128S), and mode settings are available to cause the FPGA to automatically read a configuration from this device at power on. An Artix-7 100T configuration file requires just less than four MiB (mebibyte) of memory, leaving about 77% of the flash device available for user data. Or, if the FPGA is getting configured from another source, the whole memory can be used for custom data.

The contents of the memory can be manipulated by issuing certain commands on the SPI bus. The implementation of this protocol is outside the scope of this document. All signals in the SPI bus except SCK are general-purpose user I/O pins after FPGA configuration. SCK is an exception because it remains a dedicated pin even after configuration. Access to this pin is provided through a special FPGA primitive called STARTUPE2.

NOTE: Refer to the manufacturer's data sheets[ii] and Xilinx user guides[iii] for more information.

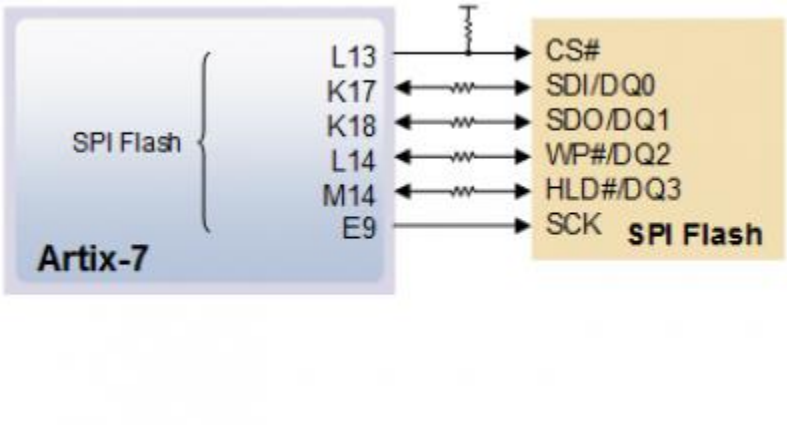


Figure 3.2.1 Nexys A7 DDR SPI Flash Pin-out

4 Ethernet PHY

The Nexys A7 board includes an SMSC 10/100 Ethernet PHY (SMSC part number LAN8720A) paired with an RJ-45 Ethernet jack with integrated magnetics. The SMSC PHY uses the RMII interface and supports 10/100 Mb/s. Figure 4.1 illustrates the pin connections between the Artix-7 and the Ethernet PHY. At power-on reset, the PHY is set to the following defaults:

- RMII mode interface
- Auto-negotiation enabled, advertising all 10/100 mode capable
- PHY address=00001

Two on-board LEDs (LD23 = LED2, LD24 = LED1) connected to the PHY provide link status and data activity feedback. See the PHY datasheet for details.

EDK-based designs can access the PHY using either the axi_ethernetlite (AXI EthernetLite) IP core or the axi_ethernet (Tri Mode Ethernet MAC) IP core. A mii_to_rmii core (Ethernet PHY MII to Reduced MII) needs to be inserted to convert the MAC interface from MII to RMII. Also, a 50 MHz clock needs to be generated for the

mii_to_rmii core and the CLKIN pin of the external PHY. To account for skew introduced by the mii_to_rmii core, generate each clock individually, with the external PHY clock having a 45 degree phase shift relative to the mii_to_rmii Ref_Clk. An EDK demonstration project that properly uses the Ethernet PHY can be found on the Nexys A7 product page at http://www.digilentinc.com/.

ISE designs can use the IP Core Generator wizard to create an Ethernet MAC controller IP core.

NOTE: Refer to the LAN8720A data sheet for further information[iv].

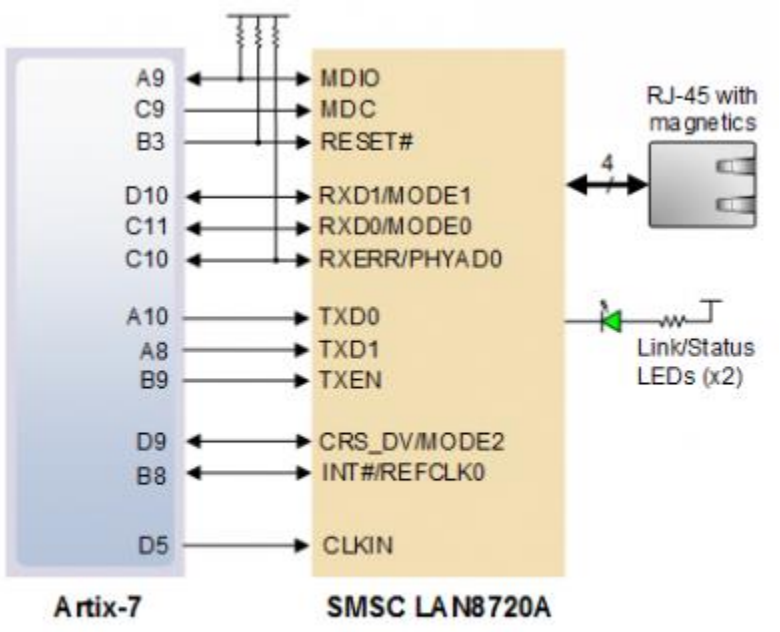


Figure 4.1 Pin Connections between the Artix-7 and the Ethernet PHY

5 Oscillators/Clocks

The Nexys A7 board includes a single 100 MHz crystal oscillator connected to pin E3 (E3 is a MRCC input on bank 35). The input clock can drive MMCMs or PLLs to generate clocks of various frequencies and with known phase relationships that may be needed throughout a design. Some rules restrict which MMCMs and PLLs may be driven by the 100 MHz input clock. For a full description of these rules and of the capabilities of the Artix-7 clocking resources, refer to the "7 Series FPGAs Clocking Resources User Guide" available from Xilinx.

Xilinx offers the Clocking Wizard IP core to help users generate the different clocks required for a specific design. This wizard will properly instantiate the needed MMCMs and PLLs based on the desired frequencies and phase relationships specified by the user. The wizard will then output an easy-to-use wrapper component around these clocking resources that can be inserted into the user's design. The clocking wizard can be accessed from within the Project Navigator or Core Generator tools.

6 USB-UART Bridge (Serial Port)

The Nexys A7 includes an FTDI FT2232HQ USB-UART bridge (attached to connector J6) that allows you use PC applications to communicate with the board using standard Windows COM port commands. Free USB-COM port drivers, available from http://www.ftdichip.com/ under the "Virtual Com Port" or VCP heading, convert USB packets to UART/serial port data. Serial port data is exchanged with the FPGA using a two-wire serial port (TXD/RXD) and optional hardware flow control (RTS/CTS). After the drivers are installed, I/O commands can be used from the PC directed to the COM port to produce serial data traffic on the C4 and D4 FPGA pins.

Two on-board status LEDs provide visual feedback on traffic flowing through the port: the transmit LED (LD20) and the receive LED (LD19). Signal names that imply direction are from the point-of-view of the DTE (Data Terminal Equipment), in this case the PC.

The FT2232HQ is also used as the controller for the Digilent USB-JTAG circuitry, but the USB-UART and USB-JTAG functions behave entirely independent of one another. Programmers interested in using the UART functionality of the FT2232 within their design do not need to worry about the JTAG circuitry interfering with the UART data transfers, and vice-versa. The combination of these two features into a single device allows the Nexys A7 to be programmed, communicated with via UART, and powered from a computer attached with a single Micro USB cable.

The connections between the FT2232HQ and the Artix-7 are shown in Figure 6.1.

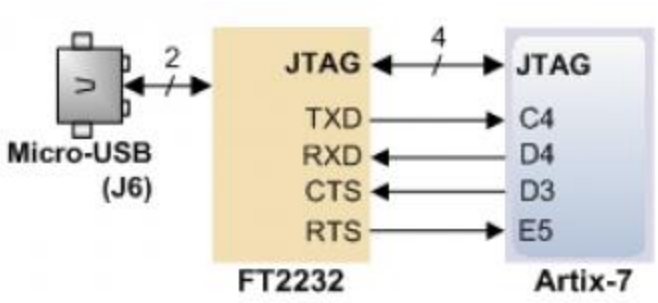


Figure 6.1 Nexys A7 FT2322HQ Connections

7 USB HID Host

The Auxiliary Function microcontroller (Microchip PIC24FJ128) provides the Nexys A7 with USB Embedded HID host capability. After power-up, the microcontroller is in configuration mode, either downloading a bitstream to the FPGA, or waiting to be programmed from other sources. Once the FPGA is programmed, the microcontroller switches to application mode, which is USB HID Host in this case. Firmware in the microcontroller can drive a mouse or a keyboard attached to the type A USB connector at J5 labeled "USB Host". Hub support is not currently available, so only a single mouse or a single keyboard can be used. Only keyboards and mice supporting the Boot HID interface are supported. The PIC24 drives several signals into the FPGA – two are used to implement a standard PS/2 interface for communication with a mouse or keyboard, and the others are connected to the FPGA's two-wire serial programming port, so the FPGA can be programmed from a file stored on a USB pen drive or microSD card.

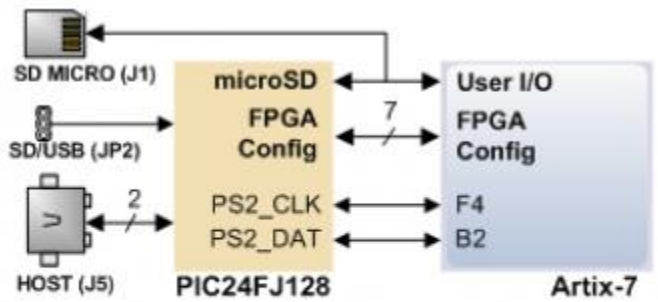
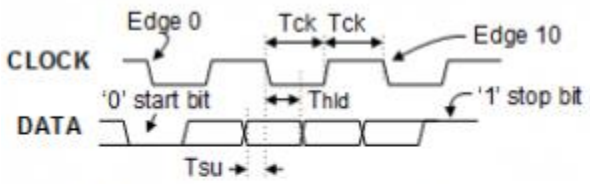


Figure 7.1 Nexys A7 PIC24 Connections

7.1 HID Controller

The Auxiliary Function microcontroller hides the USB HID protocol from the FPGA and emulates an old-style PS/2 bus. The microcontroller behaves just like a PS/2 keyboard or mouse would. This means new designs can re-use existing PS/2 IP cores. Mice and keyboards that use the PS/2 protocol use a two-wire serial bus (clock and data) to communicate with a host. On the Nexys A7, the microcontroller emulates a PS/2 device while the FPGA plays the role of the host. Both the mouse and the keyboard use 11-bit words that include a start bit, data byte (LSB first), odd parity, and stop bit, but the data packets are organized differently, and the keyboard interface allows bi-directional data transfers (so the host device can illuminate state LEDs on the keyboard). Bus timings are shown in Figure 7.1.1.



Symbol	Parameter	Min	Max
T_{CK}	Clock time	30us	50us
T_{SU}	Data-to-clock setup time	5us	25us
T_{HLD}	Clock-to-data hold time	5us	25us

Figure 7.1.1 PS/2 Device-to-Host Timing Diagram

The clock and data signals are only driven when data transfers occur; otherwise, they are held in the idle state at high-impedance (open-drain drivers). This requires that when the PS/2 signals are used in a design, internal pull-ups must be enabled in the FPGA on the data and clock pins. The clock signal is normally driven by the device, but may be held low by the host in special cases. The timings define signal requirements for mouse-to-host communications and bi-directional keyboard communications. A PS/2 interface circuit can be implemented in the FPGA to create a keyboard or mouse interface.

When a keyboard or mouse is connected to the Nexys A7, a “self-test passed” command (0xAA) is sent to the host. After this, commands may be issued to the device. Since both the keyboard and the mouse use the same PS/2 port, one can tell the type of device connected using the device ID. This ID can be read by issuing a Read ID

command (0xF2). Also, a mouse sends its ID (0x00) right after the “self-test passed” command, which distinguishes it from a keyboard.

7.2 Keyboard

PS/2-style keyboards use scan codes to communicate key press data. Each key is assigned a code that is sent whenever the key is pressed. If the key is held down, the scan code will be sent repeatedly about once every 100ms. When a key is released, an F0 key-up code is sent, followed by the scan code of the released key. If a key can be shifted to produce a new character (like a capital letter), then a shift character is sent in addition to the scan code and the host must determine which ASCII character to use. Some keys, called extended keys, send an E0 ahead of the scan code (and they may send more than one scan code). When an extended key is released, an E0 F0 key-up code is sent, followed by the scan code. Scan codes for most keys are shown in Figure 7.2.1.

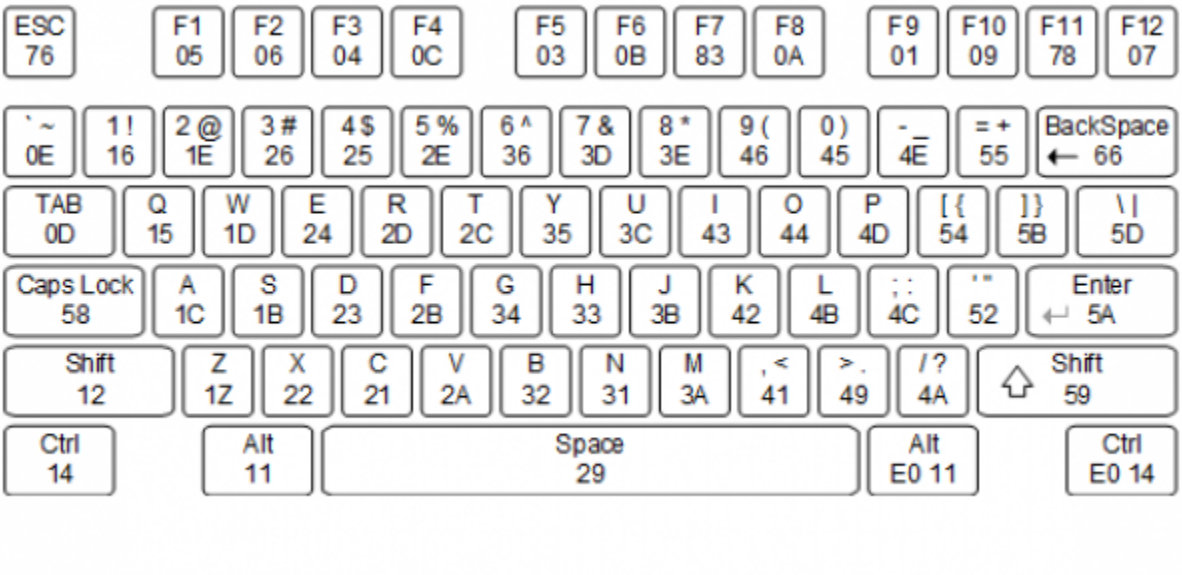


Figure 7.2.1 Keyboard Scan Codes

A host device can also send data to the keyboard. Table 7.2.1 shows a list of some common commands a host might send.

The keyboard can send data to the host only when both the data and clock lines are high (or idle). Because the host is the bus master, the keyboard must check to see whether the host is sending data before driving the bus. To facilitate this, the clock line is used as a “clear to send” signal. If the host drives the clock line low, the keyboard must not send any data until the clock is released. The keyboard sends data to the host in 11-bit words that contain a ‘0’ start bit, followed by 8-bits of scan code (LSB first), followed by an odd parity bit, and terminated with a ‘1’ stop bit. The keyboard generates 11 clock transitions (at 20 to 30 KHz) when the data is sent, and data is valid on the falling edge of the clock.

Command	Action
ED	Set Num Lock, Caps Lock, and Scroll Lock LEDs. Keyboard returns FA after receiving ED, then host sends a byte to set LED status: bit 0 sets Scroll Lock, bit 1 sets Num Lock, and bit 2 sets Caps lock. Bits 3 to 7 are ignored.
EE	Echo (test). Keyboard returns EE after receiving EE
F3	Set scan code repeat rate. Keyboard returns F3 on receiving FA, then host sends second byte to set the repeat rate.
FE	Resend. FE directs keyboard to re-send most recent scan code.

Table 7.2.1. Keyboard commands.

7.3 Mouse

Once entered in stream mode and data reporting is enabled, the mouse outputs a clock and data signal when it is moved; otherwise, these signals remain at logic '1.' Each time the mouse is moved, three 11-bit words are sent from the mouse to the host device, as shown in Figure 7.3.1. Each of the 11-bit words contains a '0' start bit, followed by 8 bits of data (LSB first), followed by an odd parity bit, and terminated with a '1' stop bit. Thus, each data transmission contains 33 bits, where bits 0, 11, and 22 are '0' start bits, and bits 11, 21, and 33 are '1' stop bits. The three 8-bit data fields contain movement data, as shown in Figure 7.3.1. Data is valid at the falling edge of the clock, and the clock period is 20 to 30 KHz.

The mouse assumes a relative coordinate system wherein moving the mouse to the right generates a positive number in the X field and moving to the left generates a negative number. Likewise, moving the mouse up generates a positive number in the Y field, and moving down represents a negative number (the XS and YS bits in the status byte are the sign bits – a '1' indicates a negative number). The magnitude of the X and Y numbers represent the rate of mouse movement; the larger the number, the faster the mouse is moving (the XV and YV bits in the status byte are movement overflow indicators. A '1' means overflow has occurred). If the mouse moves continuously, the 33-bit transmissions are repeated every 50ms or so. The L and R fields in the status byte indicate Left and Right button presses (a '1' indicates the button is being pressed).

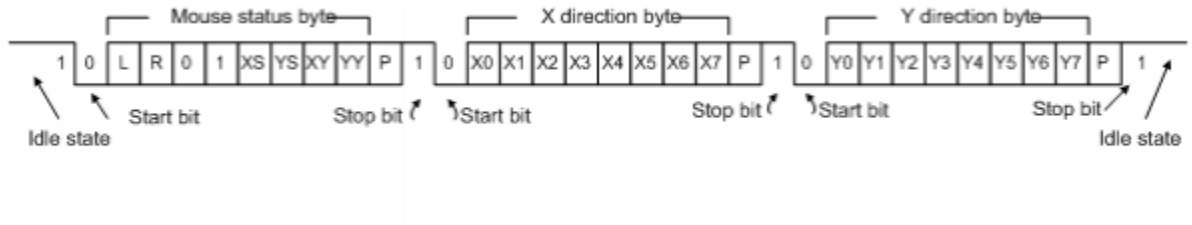


Figure 7.3.1 Mouse Data Format

The microcontroller also supports Microsoft® IntelliMouse®-type extensions for reporting back a third axis representing the mouse wheel, as shown in Table 7.3.1.

Command	Action
EA	Set stream mode. The mouse responds with "acknowledge" (0xFA) then resets its movement counters and enters stream mode.
F4	Enable data reporting. The mouse responds with "acknowledge" (0xFA) then enables data reporting and resets its movement counters. This command only affects behavior in stream mode. Once issued, mouse movement will automatically generate a data packet.
F5	Disable data reporting. The mouse responds with "acknowledge" (0xFA) then disables data reporting and resets its movement counters.
F3	Set mouse sample rate. The mouse responds with "acknowledge" (0xFA) then reads one more byte from the host. This byte is then saved as the new sample rate, and a new "acknowledge" packet is issued.
FE	Resend. FE directs mouse to re-send last packet.

Table 7.3.2. Microsoft IntelliMouse-Type extensions, commands, and actions.

8 VGA Port

The Nexys A7 board uses 14 FPGA signals to create a VGA port with 4 bits-per-color and the two standard sync signals (HS – Horizontal Sync, and VS – Vertical Sync). The color signals use resistor-divider circuits that work in conjunction with the 75-ohm termination resistance of the VGA display to create 16 signal levels each on the red, green, and blue VGA signals. This circuit, shown in Figure 8.1, produces video color signals that proceed in equal increments between 0V (fully off) and 0.7V (fully on). Using this circuit, 4096 different colors can be displayed, one for each unique 12-bit pattern. A video controller circuit must be created in the FPGA to drive the sync and color signals with the correct timing in order to produce a working display system.

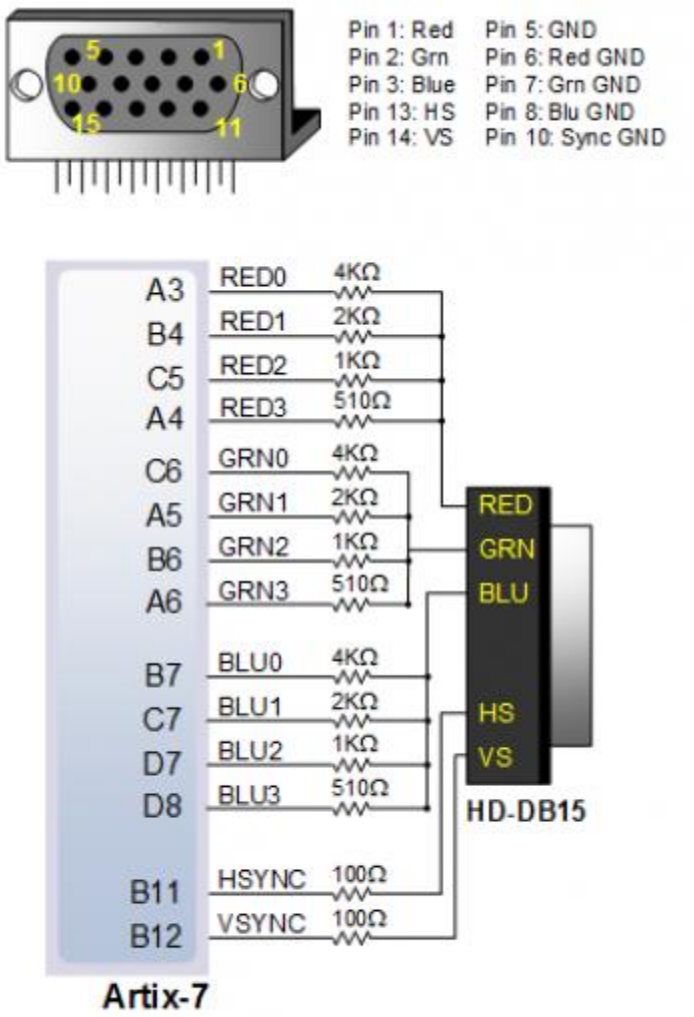


Figure 8.1 Nexys A7 VGA Interface

8.1 VGA System Timing

VGA signal timings are specified, published, copyrighted, and sold by the VESA® organization (http://www.vesa.org/). The following VGA system timing information is provided as an example of how a VGA monitor might be driven in 640 by 480 mode.

NOTE: For more precise information, or for information on other VGA frequencies, refer to documentation available at the VESA website.

CRT-based VGA displays use amplitude-modulated moving electron beams (or cathode rays) to display information on a phosphor-coated screen. LCD displays use an array of switches that can impose a voltage across a small amount of liquid crystal, thereby changing light permittivity through the crystal on a pixel-by-pixel basis. Although the following description is limited to CRT displays, LCD displays have evolved to use the same signal timings as CRT displays (so the "signals" discussion below pertains to both CRTs and LCDs). Color CRT displays use three electron beams (one for red, one for blue, and one for green) to energize the phosphor that coats the inner side of the display end of a cathode ray tube (see Figure 8.1.1).

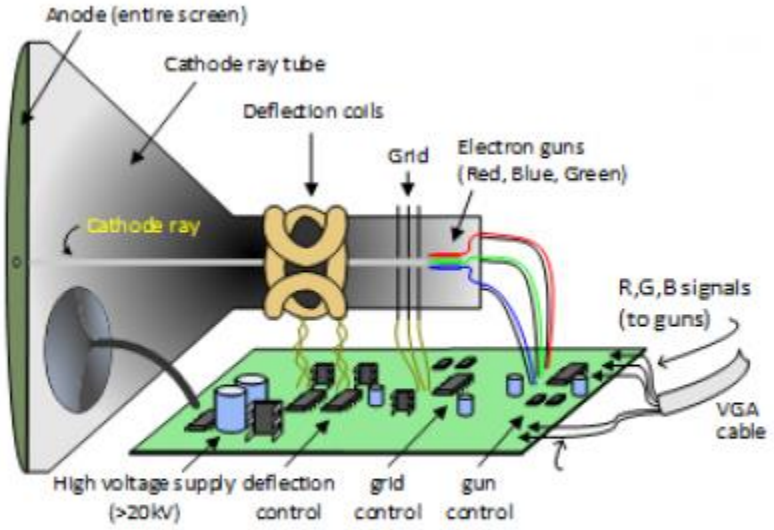


Figure 8.1.1 Color CRT Display

Electron beams emanate from "electron guns," which are finely-pointed heated cathodes placed in close proximity to a positively charged annular plate called a "grid." The electrostatic force imposed by the grid pulls rays of energized electrons from the cathodes, and those rays are fed by the current that flows into the cathodes. These particle rays are initially accelerated towards the grid, but they soon fall under the influence of the much larger electrostatic force that results from the entire phosphor-coated display surface of the CRT being charged to 20kV (or more). The rays are focused to a fine beam as they pass through the center of the grids, and then they accelerate to impact on the phosphor-coated display surface. The phosphor surface glows brightly at the impact point, and it continues to glow for several hundred microseconds after the beam is removed. The larger the current fed into the cathode, the brighter the phosphor will glow.

Between the grid and the display surface, the beam passes through the neck of the CRT where two coils of wire produce orthogonal electromagnetic fields. Because cathode rays are composed of charged particles (electrons), they can be deflected by these magnetic fields. Current waveforms are passed through the coils to produce magnetic fields that interact with the cathode rays and cause them to transverse the display surface in a "raster" pattern, horizontally from left to right and vertically from top to bottom, as shown in Figure 8.1.2. As the cathode ray moves over the surface of the display, the current sent to the electron guns can be increased or decreased to change the brightness of the display at the cathode ray impact point.

Information is only displayed when the beam is moving in the "forward" direction (left to right and top to bottom), and not during the time the beam is reset back to the left or top edge of the display. Much of the potential display time is therefore lost in "blanking" periods when the beam is reset and stabilized to begin a new horizontal or vertical display pass. The size of the beams, the frequency at which the beam can be traced across the display, and the frequency at which the electron beam can be modulated determine the display resolution.

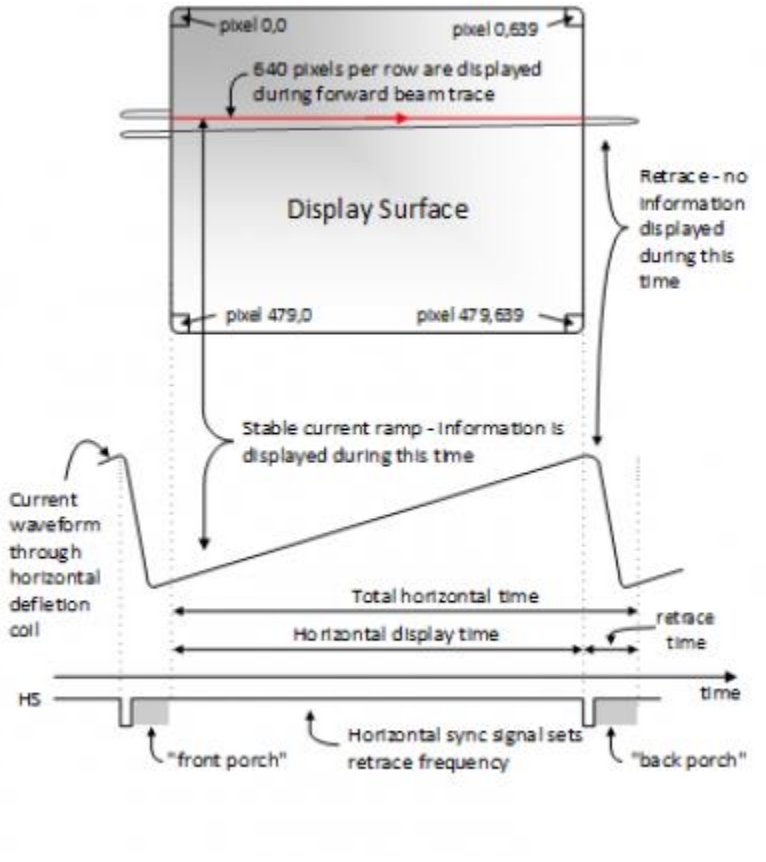


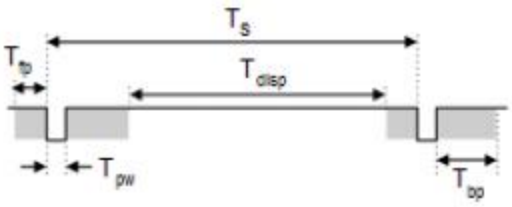
Figure 8.1.2 VGA Horizontal Synchronization

Modern VGA displays can accommodate different resolutions, and a VGA controller circuit dictates the resolution by producing timing signals to control the raster patterns. The controller must produce synchronizing pulses at 3.3V (or 5V) to set the frequency at which current flows through the deflection coils, and it must ensure that video data is applied to the electron guns at the correct time. Raster video displays define a number of "rows" that corresponds to the number of horizontal passes the cathode makes over the display area, and a number of "columns" that corresponds to an area on each row that is assigned to one "picture element," or pixel. Typical displays use from 240 to 1200 rows and from 320 to 1600 columns. The overall size of a display and the number of rows and columns determines the size of each pixel.

Video data typically comes from a video refresh memory; with one or more bytes assigned to each pixel location (the Nexys A7 uses 12 bits per pixel). The controller must index into video memory as the beams move across the display and retrieve and apply video data to the display at precisely the time the electron beam is moving across a given pixel.

A VGA controller circuit must generate the HS and VS timings signals and coordinate the delivery of video data based on the pixel clock. The pixel clock defines the time available to display one pixel of information. The VS signal defines the "refresh" frequency of the display, or the frequency at which all information on the display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity, with practical

refresh frequencies falling in the 50Hz to 120Hz range. The number of lines to be displayed at a given refresh frequency defines the horizontal “retrace” frequency. For a 640-pixel by 480-row display using a 25 MHz pixel clock and 60 +/-1Hz refresh, the signal timings shown in Figure 8.1.3 can be derived. Timings for sync pulse width and front and back porch intervals (porch intervals are the pre- and post-sync pulse times during which information cannot be displayed) are based on observations taken from actual VGA displays.



Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Figure 8.1.3 Signal Timings for a 640-Pixel by 480-Row Display Using a 25 MHz Pixel Clock and 60 Hz Vertical Refresh

A VGA controller circuit, such as the one diagrammed in Figure 8.1.4, decodes the output of a horizontal-sync counter driven by the pixel clock to generate HS signal timings. You can use this counter to locate any pixel location on a given row. Likewise, the output of a vertical-sync counter that increments with each HS pulse can be used to generate VS signal timings, and you can use this counter to locate any given row. These two continually running counters can be used to form an address into video RAM. No time relationship between the onset of the HS pulse and the onset of the VS pulse is specified, so you can arrange the counters to easily form video RAM addresses, or to minimize decoding logic for sync pulse generation.

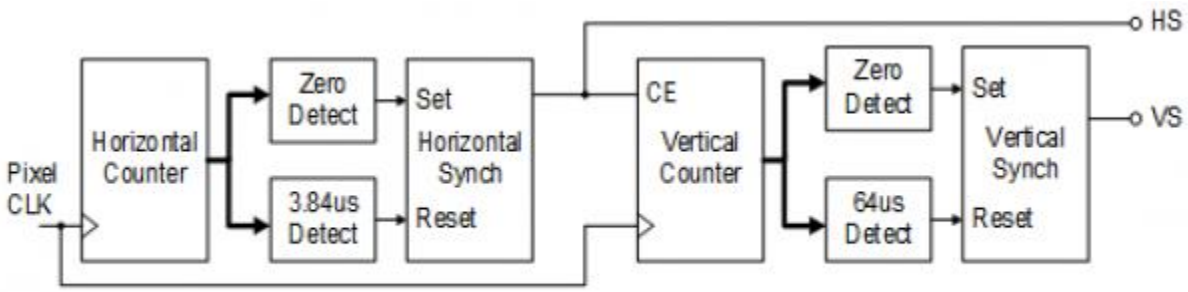


Figure 8.1.4 VGA Display Controller Block Diagram

9 Basic I/O

The Nexys A7 board includes two tri-color LEDs, sixteen slide switches, six push buttons, sixteen individual LEDs, and an eight-digit seven-segment display, as shown in Figure 9.1. The pushbuttons and slide switches are connected to the FPGA via series resistors to prevent damage from inadvertent short circuits (a short circuit could occur if an FPGA pin assigned to a pushbutton or slide switch was inadvertently defined as an output). The five pushbuttons arranged in a plus-sign configuration are "momentary" switches that normally generate a low output when they are at rest, and a high output only when they are pressed. The red pushbutton labeled "CPU RESET," on the other hand, generates a high output when at rest and a low output when pressed. The CPU RESET button is intended to be used in EDK designs to reset the processor, but you can also use it as a general-purpose pushbutton. Slide switches generate constant high or low inputs depending on their position.

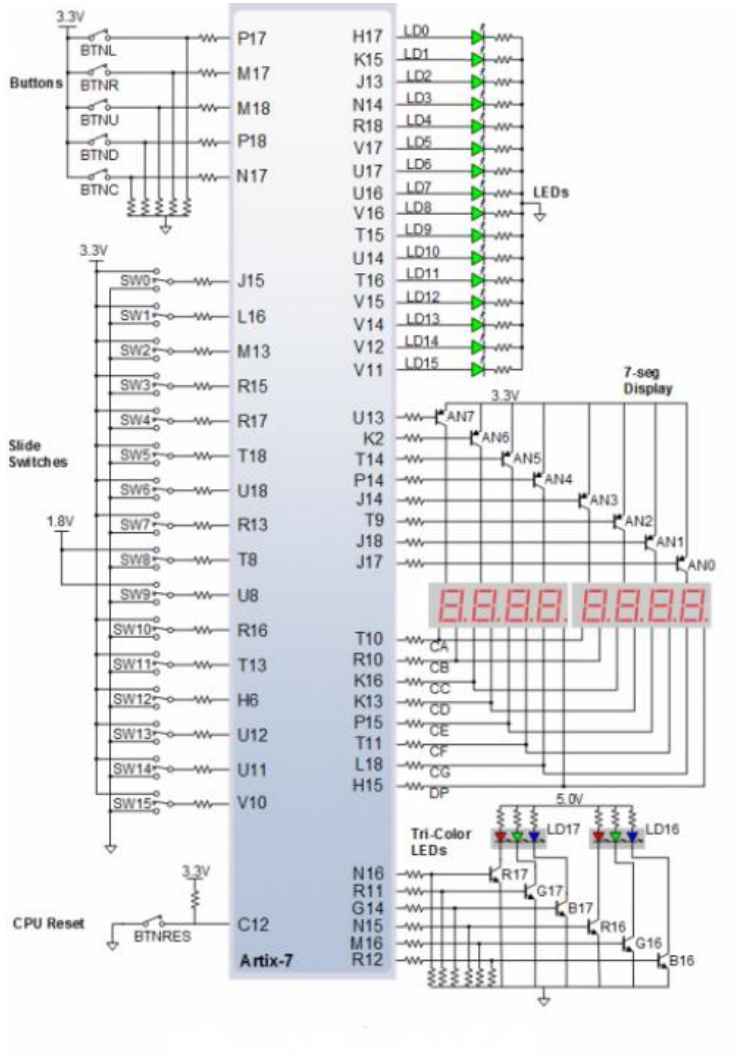


Figure 9.1 General Purpose I/O Devices on the Nexys A7

The sixteen-individual high-efficiency LEDs are anode-connected to the FPGA via 330-ohm resistors, so they will turn on when a logic high voltage is applied to their respective I/O pin. Additional LEDs that are not user-accessible indicate power-on, FPGA programming status, and USB and Ethernet port status.

9.1 Seven-Segment Display

The Nexys A7 board contains two four-digit common anode seven-segment LED displays, configured to behave like a single eight-digit display. Each of the eight digits is composed of seven segments arranged in a “figure 8” pattern, with an LED embedded in each segment. Segment LEDs can be individually illuminated, so any one of 128 patterns can be displayed on a digit by illuminating certain LED segments and leaving the others dark, as shown in Figure 9.1.1. Of these 128 possible patterns, the ten corresponding to the decimal digits are the most useful.

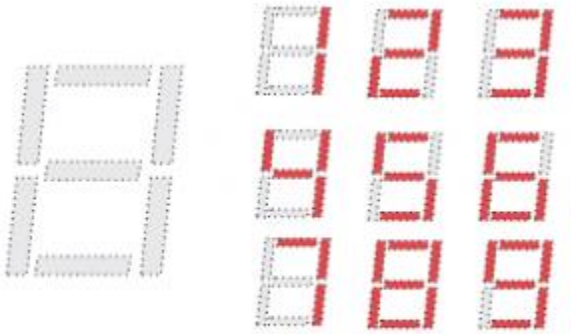


Figure 9.1.1 An Un-illuminated Seven-Segment Display and Nine Illumination Patterns Corresponding to Decimal Digits

The anodes of the seven LEDs forming each digit are tied together into one “common anode” circuit node, but the LED cathodes remain separate, as shown in Fig 18. The common anode signals are available as eight “digit enable” input signals to the 8-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG. For example, the eight “D” cathodes from the eight digits are grouped together into a single circuit node called “CD.” These seven cathode signals are available as inputs to the 8-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

To illuminate a segment, the anode should be driven high while the cathode is driven low. However, since the Nexys A7 uses transistors to drive enough current into the common anode point, the anode enables are inverted. Therefore, both the AN0..7 and the CA..G/DP signals are driven low when active.

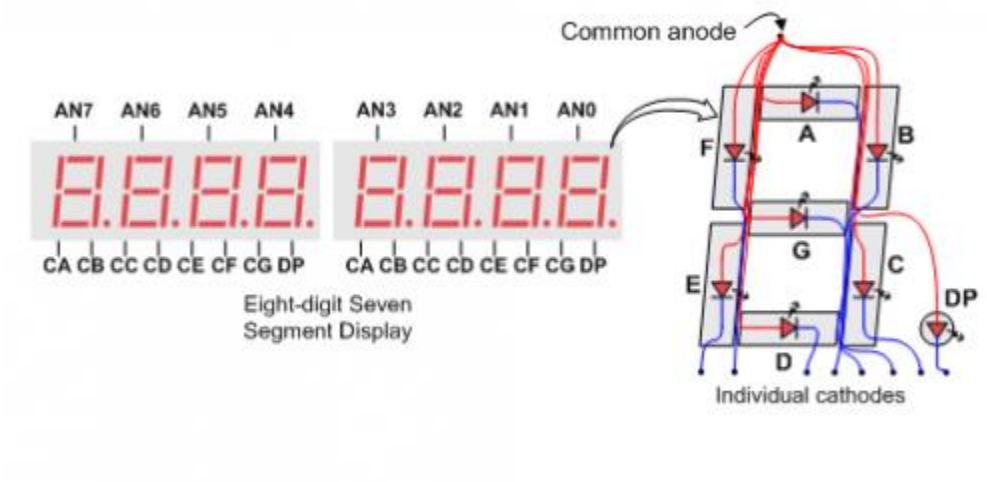


Figure 9.1.2 Common Anode Circuit Node

A scanning display controller circuit can be used to show an eight-digit number on this display. This circuit drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession at an update rate that is faster than the human eye can detect. Each digit is illuminated just one-eighth of the time, but because the eye cannot perceive the darkening of a digit before it is illuminated again, the digit appears continuously illuminated. If the update, or "refresh", rate is slowed to around 45Hz, a flicker can be noticed in the display.

For each of the four digits to appear bright and continuously illuminated, all eight digits should be driven once every 1 to 16ms, for a refresh frequency of about 1 KHz to 60Hz. For example, in a 62.5Hz refresh scheme, the entire display would be refreshed once every 16ms, and each digit would be illuminated for 1/8 of the refresh cycle, or 2ms. The controller must drive low the cathodes with the correct pattern when the corresponding anode signal is driven high. To illustrate the process, if AN0 is asserted while CB and CC are asserted, then a "1" will be displayed in digit position 1. Then, if AN1 is asserted while CA, CB, and CC are asserted, a "7" will be displayed in digit position 2. If AN0, CB, and CC are driven for 4ms, and then AN1, CA, CB, and CC are driven for 4ms in an endless succession, the display will show "71" in the first two digits. An example timing diagram for a four-digit controller is shown in Figure 9.1.3.

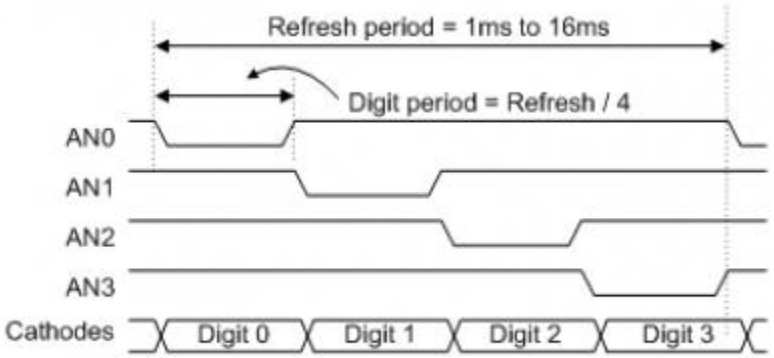


Figure 9.1.3 Four Digit Scanning Display Controller Timing Diagram

9.2 Tri-Color LED

The Nexys A7 board contains two tri-color LEDs. Each tri-color LED has three input signals that drive the cathodes of three smaller internal LEDs: one red, one blue, and one green. Driving the signal corresponding to one of these colors high will illuminate the internal LED. The input signals are driven by the FPGA through a transistor, which inverts the signals. Therefore, to light up the tri-color LED, the corresponding signals need to be driven high. The tri-color LED will emit a color dependent on the combination of internal LEDs that are currently being illuminated. For example, if the red and blue signals are driven high, and green is driven low, the tri-color LED will emit a purple color.

Note: Digilent strongly recommends the use of pulse-width modulation (PWM) when driving the tri-color LEDs (for information on PWM, see section 15.1 Pulse Density Modulation (PDM)). Driving any of the inputs to a steady logic '1' will result in the LED being illuminated at an uncomfortably bright level. You can avoid this by ensuring that none of the tri-color signals are driven with more than a 50% duty cycle. Using PWM also greatly expands the potential color palette of the tri-color led. Individually adjusting the duty cycle of each color between 50% and 0% causes the different colors to be illuminated at different intensities, allowing virtually any color to be displayed.

10 Pmod Ports

The Pmod ports are arranged in a 2×6 right-angle, and are 100-mil female connectors that mate with standard 2×6 pin headers. Each 12-pin Pmod port provides two 3.3V VCCsignals (pins 6 and 12), two Ground signals (pins 5 and 11), and eight logic signals, as shown in Figure 10.1. The VCC and Ground pins can deliver up to 1A of current. Pmod data signals are not matched pairs, and they are routed using best-available tracks without impedance control or delay matching. Pin assignments for the Pmod I/O connected to the FPGA are shown in Table 5.

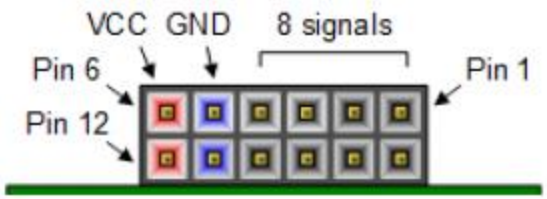


Figure 10.1 Pmod Connectors; Front View, as Loaded on PCB

Pmod JA	Pmod JB	Pmod JC	Pmod JD	Pmod XDAC
JA1: C17	JB1: D14	JC1: K1	JD1: H4	JXADC1: A13 (AD3P)
JA2: D18	JB2: F16	JC2: F6	JD2: H1	JXADC2: A15 (AD10P)
JA3: E18	JB3: G16	JC3: J2	JD3: G1	JXADC3: B16 (AD2P)
JA4: G17	JB4: H14	JC4: G6	JD4: G3	JXADC4: B18 (AD11P)
JA7: D17	JB7: E16	JC7: E7	JD7: H2	JXADC7: A14 (AD3N)
JA8: E17	JB8: F13	JC8: J3	JD8: G4	JXADC8: A16 (AD10N)
JA9: F18	JB9: G13	JC9: J4	JD9: G2	JXADC9: B17 (AD2N)

Table 10.1. Nexys A7 Pmod Pin Assignment

Digilent produces a large collection of Pmod accessory boards that can attach to the Pmod expansion connectors to add ready-made functions like A/D's, D/A's, motor drivers, sensors, as well as other functions.
See http://www.digilentinc.com/ for more information.

10.1 Dual Analog/Digital Pmod

The on-board Pmod expansion connector labeled "JXADC" is wired to the auxiliary analog input pins of the FPGA. Depending on the configuration, this connector can be used to input differential analog signals to the analog-to-digital converter inside of the Artix-7 (XADC). Any or all pairs in the connector can be configured either as analog input or digital input-output.

The Dual Analog/Digital Pmod on the Nexys A7 differs from the rest in the routing of its traces. The eight data signals are grouped into four pairs, with the pairs routed closely coupled for better analog noise immunity. Furthermore, each pair has a partially loaded anti-alias filter laid out on the PCB. The filter does not have capacitors C60-C63. In designs where such filters are desired, the capacitors can be manually loaded by the user.

NOTE: The coupled routing and the anti-alias filters might limit the data speeds when used for digital signals.

The XADC core within the Artix-7 is a dual channel 12-bit analog-to-digital converter capable of operating at 1 MSPS. Either channel can be driven by any of the auxiliary analog input pairs connected to the JXADC header. The XADC core is controlled and accessed from a user design via the Dynamic Reconfiguration Port (DRP). The DRP also provides access to voltage monitors that are present on each of the FPGA's power rails, and a temperature sensor

that is internal to the FPGA. For more information on using the XADC core, refer to the Xilinx document titled "7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter."

11 MicroSD Slot

The Nexys A7 provides a microSD slot for both FPGA configuration and user access. The on-board Auxiliary Function microcontroller shares the SD card bus with the FPGA. Before the FPGA is configured the microcontroller must have access to the SD card via SPI. Once a bit file is downloaded to the FPGA (from any source), the microcontroller power cycles the SD slot and relinquishes control of the bus. This enables any SD card in the slot to reset its internal state machines and boot up in SD native bus mode. All of the SD pins on the FPGA are wired to support full SD speeds in native interface mode, as shown in Figure 11.1. The SPI is also available, if needed. Once control over the SD bus is passed from the microcontroller to the FPGA, the SD_RESET signal needs to be actively driven low by the FPGA to power the microSD card slot. For information on implementing an SD card controller, refer to the SD card specification available at http://www.sdcard.org/

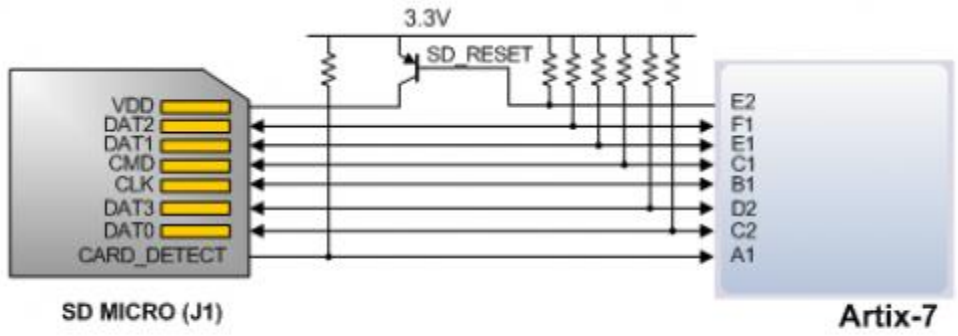


Figure 11.1 Artix-7 microSD Card Connector Interface (PIC24 Connections not Shown)

12 Temperature Sensor

The Nexys A7 includes an Analog Device ADT7420 temperature sensor. The sensor provides up to 16-bit resolution with a typical accuracy better than 0.25 degrees Celsius. The interface between the temperature sensor and FPGA is shown in Figure 12.1.

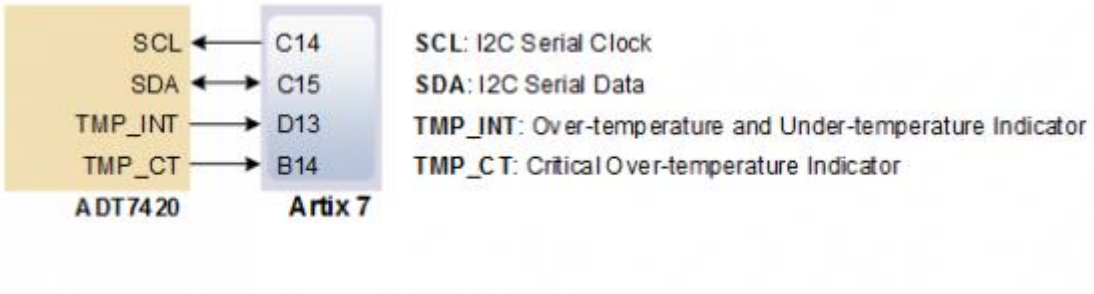


Figure 12.1 Temperature Sensor Interface

12.1 I²C Interface

The ADT7420 chip acts as a slave device using the industry standard I²C communication scheme. To communicate with ADT7420 chip, the I²C master must specify a slave address (0x4B) and a flag indicating whether the communication is a read (1) or a write (0). Once specifications are made for communication, a data transfer takes place. For ADT7420, the data transfer should consist of the address of the desired device register followed by the data to be written to the specified register. To read from a register, the master must write the desired register address to the ADT7420, then send an I²C restart condition, and send a new read request to the ADT7420. If the master does not generate a restart condition prior to attempting the read, the value written to the address register will be reset to 0x00. As some registers store 16-bit values as 8-bit register pairs, the ADT7420 will automatically increment the address register of the device when accessing certain registers, such as the temperature registers and the threshold registers. This allows for the master to use a single read or write request to access both the low and high bytes of these registers. A complete listing of registers and their behavior can be found in the ADT7420 datasheet available on the Analog Devices website.

12.2 Open Drain Outputs

The ADT7420 provides two open drain output signals to indicate when pre-set temperature thresholds are reached. If the temperature leaves a range defined by registers TLOW (0x06:0x07) and THIGH (0x04:0x05), the INT pin can be driven low or high based upon the configuration of the device. Similarly, the CT pin can be driven low or high if the temperature exceeds a critical threshold defined in TCRIT (0x08:0x09). Both of these pins need internal FPGA pull-ups when used.

For details on the electrical specifications and configuration of the INT and CT pins, refer to the ADT7420 datasheet.

12.3 Quick Start Operation

When the ADT7420 is powered up, it is in a mode that can be used as a simple temperature sensor without any initial configuration. By default, the device address register points to the temperature MSB register, so a two byte read without specifying a register will read the value of the temperature register from the device. The first byte read back will be the most significant byte (MSB) of the temperature data, and the second will be the least significant byte (LSB) of the data. These two bytes form a two's complement 16-bit integer. If the result is shifted to the right three bits and multiplied by 0.0625, the resulting signed floating point value will be a temperature reading in degrees Celsius.

For information on reading and writing to the other registers of the device, as well as notes on the accuracy of the temperature measurements, refer to the ADT7420 datasheet.

13 Accelerometer

The Nexys A7 includes an Analog Device ADXL362 accelerometer. The ADXL362 is a 3-axis MEMS accelerometer that consumes less than 2µA at a 100Hz output data rate and 270nA when in motion triggered wake-up mode. Unlike accelerometers that use power duty cycling to achieve low power consumption, the ADXL362 does not alias input signals by under-sampling; it samples the full bandwidth of the sensor at all data rates. The ADXL362 always provides 12-bit output resolution; 8-bit formatted data is also provided for more efficient single-byte transfers when a lower resolution is sufficient. Measurement ranges of ±2 g, ±4 g, and ±8 g are available with a resolution of 1 mg/LSB on the ±2 g range. The FPGA can talk with the ADXL362 via SPI interface. While the ADXL362 is in

Measurement Mode, it continuously measures and stores acceleration data in the X-data, Y-data, and Z-data registers. The interface between the FPGA and accelerometer can be seen in Figure 13.1.

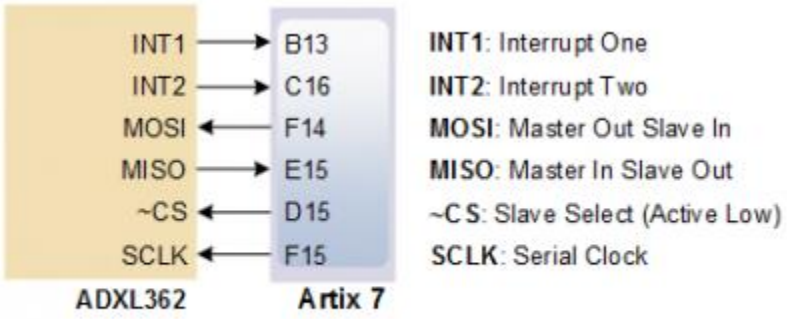


Figure 13.1 Accelerometer Interface

13.1 SPI Interface

The ADXL362 acts as a slave device using an SPI communication scheme. The recommended SPI clock frequency ranges from 1 MHz to 5 MHz. The SPI operates in SPI mode 0 with CPOL = 0 and CPHA = 0. All communications with the device must specify a register address and a flag that indicate whether the communication is a read or a write. Actual data transfer always follows the register address and communication flag. Device configuration can be performed by writing to the control registers within the accelerometer. Access accelerometer data by reading the device registers.

For a full list of registers, their functionality, and communication specifications, refer to the ADXL362 datasheet[v].

13.2 Interrupts

Several of the built-in functions of the ADXL362 can trigger interrupts that alert the host processor of certain status conditions. Interrupts can be mapped to either (or both) of two interrupt pins (INT1, INT2). Both of these pins require internal FPGA pull-ups when used. For more details about the interrupts, see the ADXL362 datasheet.

14 Microphone

The Nexys A7 board includes an omnidirectional MEMS microphone. The microphone uses an Analog Device ADMP421 chip which has a high signal to noise ratio (SNR) of 61dBA and high sensitivity of -26 dBFS. It also has a flat frequency response ranging from 100Hz to 15 kHz. The digitized audio is output in the pulse density modulated (PDM) format. The component architecture is shown in Figure 14.1.

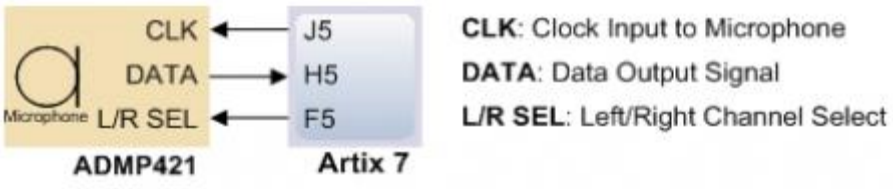


Figure 14.1 Microphone Block Diagram

14.1 Pulse Density Modulation (PDM)

PDM data connections are becoming more and more popular in portable audio applications, such as cellphones and tablets. With PDM, two channels can be transmitted with only two wires. The frequency of a PDM signal usually falls in the range of 1 MHz to 3 MHz. In a PDM bitstream, a 1 corresponds to a positive pulse and a 0 corresponds to a negative pulse. A run consisting of all '1's would correspond to the maximum positive value and a run of '0's would correspond to the minimum amplitude value. Figure 14.1.1 shows how a sine wave is represented in PDM signal.

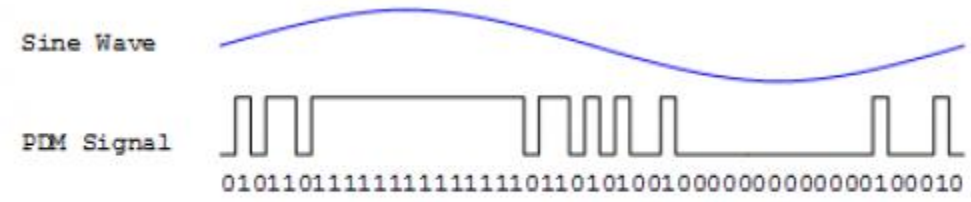


Figure 14.1.1 PDM Representation of a Sine Wave

A PDM signal is generated from an analog signal through a process called delta-sigma modulation. A simple idealized circuit of delta-sigma modulator is shown in Figure 14.1.2.

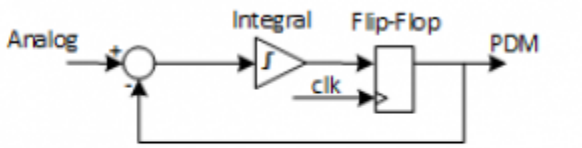


Figure 14.1.2 Simple Delta-Sigma Modulator Circuit

Sum	Integrator Out	Flip-flop Output
0.4-0=0.4	0+0.4=0.4	0
0.4-0=0.4	0.4+0.4=0.8	1
0.4-1=-0.6	0.8-0.6=0.2	0
0.4-0=0.4	0.2+0.4=0.6	1
0.4-1=-0.6	0.6-0.6=0	0
0.4-0=0.4	0+0.4=0.4	0
0.4-0=0.4	0.4+0.4=0.8	1

Table 14.1.2. Sigma delta modulator with a 0.4Vdd input.

To keep things simple, assume that the analog input and digital output have the same voltage range 0~Vdd. The input of the flip-flop acts like a comparator (any signal above Vdd/2 is considered as '1' and any input bellow Vdd/2 is considered '0'). The input of the integral circuit is the difference of the input analog signal and the PDM signal of the previous clock cycle. The integral circuit then integrates both of these inputs, and the output of the integral circuit is sampled by a D-Flip-flop. Table 6 shows the function of the delta-sigma modulator with an input of 0.4Vdd.

Note that the average of the flip-flop output equals the value of the input analog signal. So in order to get the value of analog input, all that is needed is a counter that counts the '1's for a certain period of time.

14.2 Microphone Digital Interface Timing

The clock input of the microphone can range from 1 MHz to 3.3 MHz based on the sampling rate and data precision requirement of the applications. The L/R Select signal must be set to a valid level, depending on which edge of the clock the data bit will be read. A low level on L/RSEL makes data available on the rising edge of the clock, while a high level corresponds to the falling edge of the clock, as shown in Figure 14.2.1.

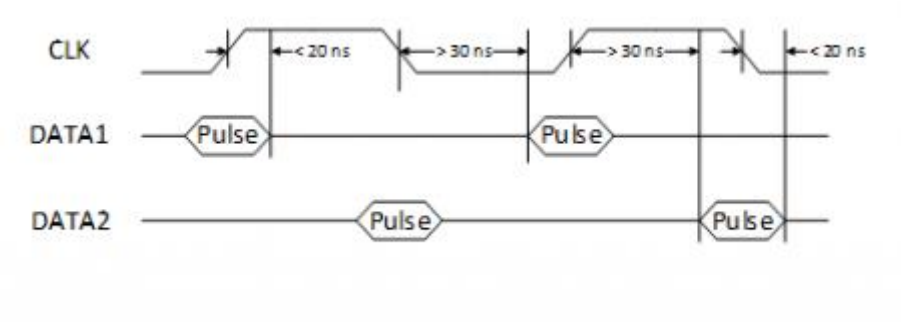


Figure 14.2.1 PDM Timing Diagram

The typical value of the clock frequency is 2.4 MHz. Assuming that the application requires 7-bit precision and 24 KHz, there can be two counters that count 128 samples at 12 KHz, as shown in Figure 14.2.2.

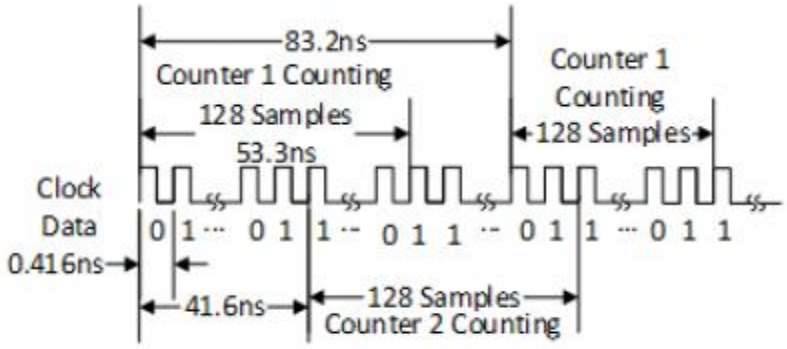


Figure 14.2.2 Sampling PDM with Two Counters

15 Mono Audio Output

The on-board audio jack (J8) is driven by a Sallen-Key Butterworth Low-pass 4th Order Filter that provides mono audio output. The circuit of the low-pass filter is shown in Figure 15.1. The input of the filter (AUD_PWM) is connected to the FPGA pin A11. A digital input will typically be a pulse-width modulated (PWM) or pulse density modulated (PDM) open-drain signal produced by the FPGA. The signal needs to be driven low for logic '0' and left in high-impedance for logic '1'. An on-board pull-up resistor to a clean analog 3.3V rail will establish the proper voltage for logic '1'. The low-pass filter on the input will act as a reconstruction filter to convert the pulse-width modulated digital signal into an analog voltage on the audio jack output.

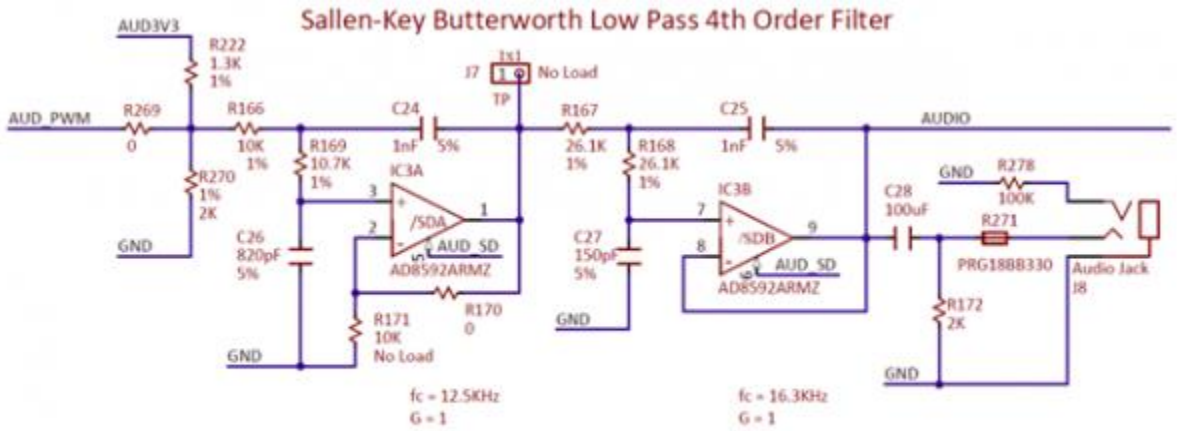


Figure 15.1 Sallen-Key Butterworth Low-Pass 4th Order Filter

The frequency response of SK Butterworth Low-Pass Filter is shown in Figure 15.2. The AC analysis of the circuit is done using NI Multisim 12.0.

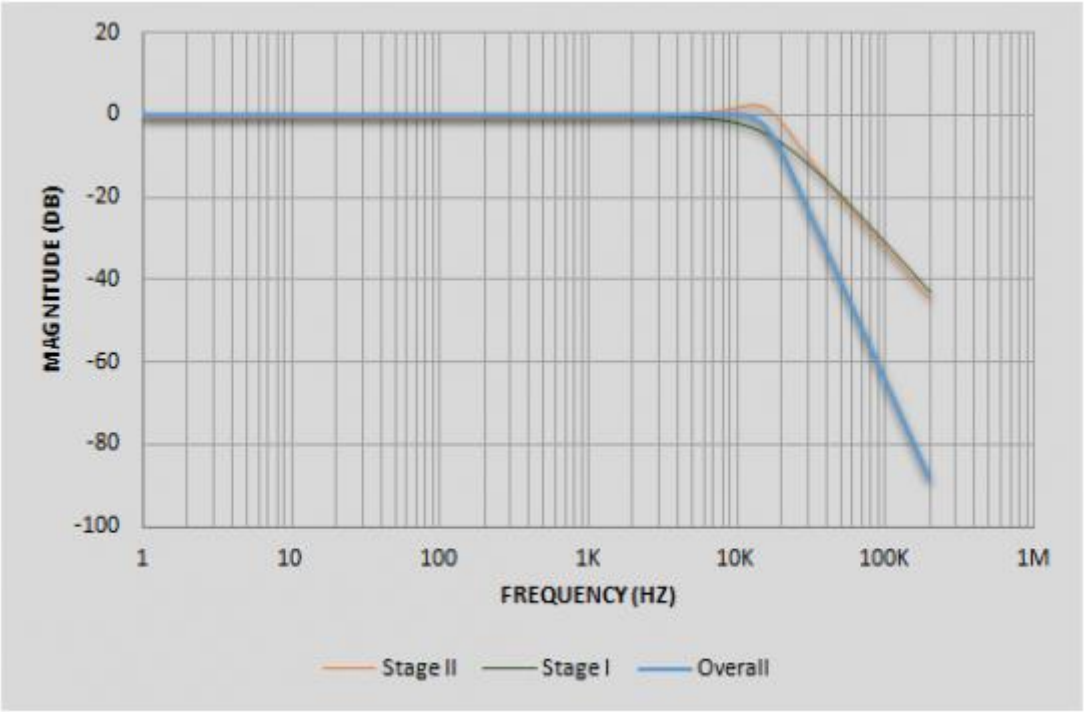


Figure 15.2 SK Butterworth Low-Pass Filter Frequency Response

15.2 Pulse-Width Modulation

A pulse-width modulated (PWM) signal is a chain of pulses at some fixed frequency, with each pulse potentially having a different width. This digital signal can be passed through a simple low-pass filter that integrates the digital waveform to produce an analog voltage proportional to the average pulse-width over some interval (the interval is

determined by the 3dB cut-off frequency of the low-pass filter and the pulse frequency). For example, if the pulses are high for an average of 10% of the available pulse period, then an integrator will produce an analog value that is 10% of the Vdd voltage. Figure 15.1.1 shows a waveform represented as a PWM signal.

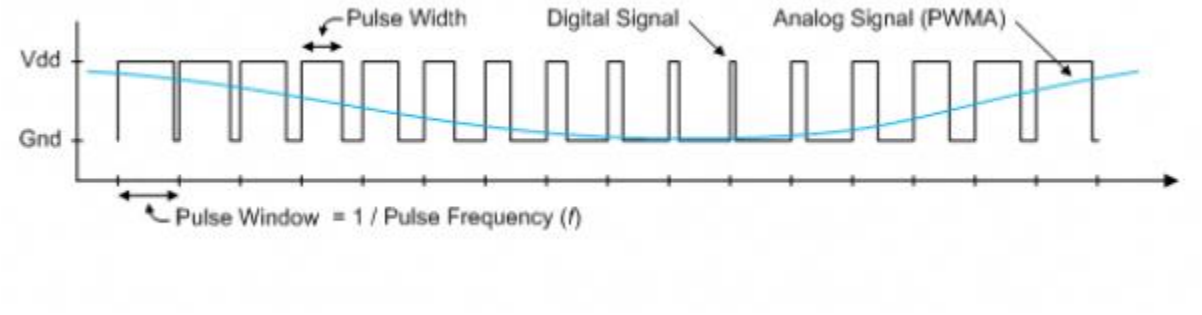


Figure 15.1.1 Simple Waveform Represented as PWM

The PWM signal must be integrated to define an analog voltage. The low-pass filter 3dB frequency should be an order of magnitude lower than the PWM frequency, so that signal energy at the PWM frequency is filtered from the signal. For example, if an audio signal must contain up to 5 KHz of frequency information, then the PWM frequency should be at least 50 KHz (and preferably even higher). In general, in terms of analog signal fidelity, the higher the PWM frequency, the better. Figure 15.1.2 shows a representation of a PWM integrator producing an output voltage by integrating the pulse train. Note the steady-state filter output signal amplitude ratio to Vdd is the same as the pulse-width duty cycle (duty cycle is defined as pulse-high time divided by pulse-window time).

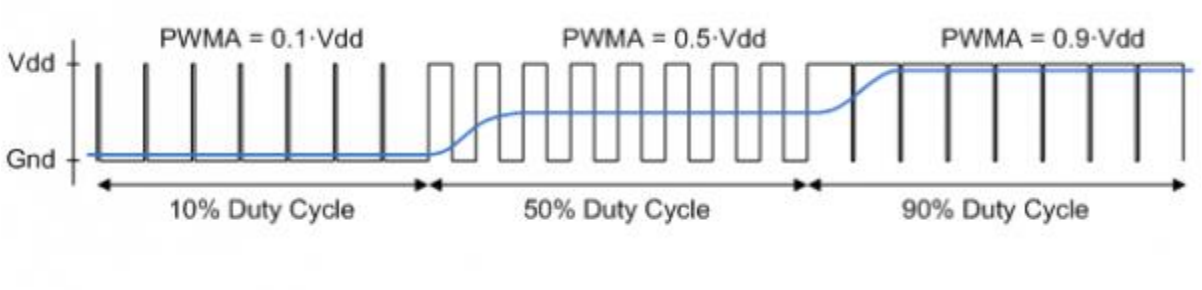


Figure 15.1.2 Representation of a PWM Integrator Producing an Output Voltage by Integrating the Pulse Train

Built-In Self-Test

A demonstration configuration is loaded into the Quad-SPI flash device on the Nexys A7 board during manufacturing. The source code and prebuilt bitstream for this design are available for download from the Digilent website. If the demo configuration is present in the flash and the Nexys A7 board is powered on in SPI mode, the demo project will allow basic hardware verification. Here is an overview of how this demo drives the different onboard components:

- The user LEDs are illuminated when the corresponding user switch is placed in the on position.
- The tri-color LEDs are controlled by some of the user buttons. Pressing BTNL, BTNC, or BTNR causes them to illuminate either red, green, or blue, respectively. Pressing BTND causes them to begin cycling through many colors. Repeatedly pressing BTND will turn the two LEDs on or off.
- Pressing BTNU will trigger a 5 second recording from the onboard PDM microphone. This recording is then immediately played back on the mono audio out port. The status of the recording and playback is displayed on the user LEDs. The recording is saved in the DDR2 memory.

- The VGA port displays feedback from the onboard microphone, temperature sensors, accelerometer, RGB LEDs, and USB Mouse.
- Connecting a mouse to the USB-HID Mouse port will allow the pointer on the VGA display to be controlled. Only mice compatible with the Boot Mouse HID interface are supported.
- The seven-segment display will display a moving snake pattern.

All Nexys A7 boards are 100% tested during the manufacturing process. If any device on the Nexys A7 board fails test or is not responding properly, it is likely that damage occurred during transport or during use. Typical damage includes stressed solder joints and contaminants in switches and buttons resulting in intermittent failures. Stressed solder joints can be repaired by reheating and reflowing solder and contaminants can be cleaned with off-the-shelf electronics cleaning products. If a board fails test within the warranty period, it will be replaced at no cost. Contact Digilent for more details.

[i] Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions from Xilinx
[ii] Spansion S25FL032P_00 Datasheet
[iii] 7-Series FPGAs Configuration User Guide from Xilinx
[iv] SMSC LAN8720A Datasheet from Microchip
[v] ADXL362 Product Page from Analog Devices