Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica y Automática

Monitorización y control de la intensidad luminosa mediante FPGA

Autor:

García Gallego, Gonzalo Agustín

Tutor:

Andrés Rodríguez-Trelles, Francisco José de
Departamento de Tecnología Electrónica

Valladolid, julio de 2014.

Resumen:

El objetivo de este proyecto es diseñar y construir un dispositivo capaz de monitorizar y controlar la intensidad luminosa de su entorno. Se podrá introducir una señal de referencia de dicha intensidad que el sistema tratará de alcanzar modificando el nivel de iluminación de una lámpara LED. El programa de control se implementará en una FPGA Lattice, modelo MachX02-1200ZE. Para monitorizar la intensidad luminosa se usará un convertidor luz-frecuencia TCS3210. La lámpara LED será controlada mediante modulación por ancho de pulso (PWM).

Palabras clave:

Control, monitorización, iluminación, FPGA, PWM.

INDICE

Capítulo 1. Introducción y Objetivos

En este capítulo se realizará una introducción al Trabajo Fin de Grado que se presenta. En primer lugar, se hará una breve descripción del proyecto. A continuación se plantearán los objetivos que se pretenden alcanzar con el mismo.

1.1 Descripción del proyecto

Este proyecto se lleva a cabo con objetivo de superar la asignatura "Trabajo Fin de Grado" de la titulación Grado en Ingeniería Electrónica y Automática de la Escuela de Ingenierías Industriales de la Universidad de Valladolid. Se ha realizado en el Departamento de Tecnología Electrónica de dicha escuela.

Se pretende construir un dispositivo capaz de monitorizar el nivel de intensidad luminosa de una sala. Además, en función de este valor se controlará una bombilla.

Como controlador del sistema se usará una FPGA Lattice, concretamente el modelo *MachX02-1200ZE*. Este dispositivo estará montado sobre una placa de prototipo *Breakout Board*.

Figura 1: FPGA Lattice MachX02-1200ZE montada en una Breakout Board

El software que se empleará en este proyecto es el *Lattice Diamond*, que proporciona el mismo fabricante. Esta herramienta nos permitirá tanto programar el dispositivo como realizar simulaciones para comprobar su correcto funcionamiento.

En cuanto al hardware, añadiremos dos placas PCB que proporcionarán al montaje ocho pulsadores, ocho micro interruptores y dos displays de 7 segmentos. También conectaremos un sensor convertidor de luz-frecuencia TCS3210 en los pines auxiliares para placa de expansión.

Figura 2: Convertidor luz- frecuencia TCS3210

Se añadirá también al montaje una lámpara LED modificada para funcionar a 12V y su fuente de alimentación correspondiente. El control de ésta lámpara se llevará a cabo mediante modulación por ancho de pulso (PWM) y estará gobernado por la lógica interna de la FPGA a través de un transistor.

Se calibrará el sistema con la ayuda de un luxómetro para obtener la relación entre el valor mostrado en los displays y la intensidad luminosa real en luxes.

1.2 Objetivos

Los objetivos que se pretenden alcanzar con el desarrollo de este proyecto son los siguientes:

- ✓ Analizar las características y posibilidades de los dispositivos tipo FPGA.
- ✓ Emplear herramientas de diseño, simulación y síntesis modernas, diseñando de forma estructurada y jerárquica.
- ✓ Emplear técnicas de diseño mixtas, utilizando lenguajes de descripción de hardware y esquemas. La descripción global se hará en base a esquemas y los diferentes bloques funcionales en VHDL.
- ✓ Utilizar herramientas automáticas para la generación de descripciones VHDL.

- ✓ Realizar y poner a punto un sistema de monitorización de la intensidad luminosa, así como el control de una lámpara LED en función del valor de dicha intensidad.
- ✓ Analizar los resultados y sacar conclusiones del trabajo realizado.

Capítulo 2. Alternativas de Diseño

En este capítulo se analizarán otras posibles alternativas que podrían haberse usado para el control de nuestro sistema. Se señalarán en cada caso sus principales ventajas e inconvenientes.

2.1 Circuito integrado de aplicación específica (ASIC)

Se conoce como ASIC a un circuito integrado personalizado que ha sido diseñado a medida para un propósito u aplicación específica dentro de un producto electrónico concreto. Al contrario que otros dispositivos, pueden contener funciones analógicas, digitales, y combinaciones de ambas.

En la actualidad, gracias a los avances en las tecnologías de miniaturización y de diseño, podemos encontrar en un ASIC más de 100 millones de puertas lógicas.

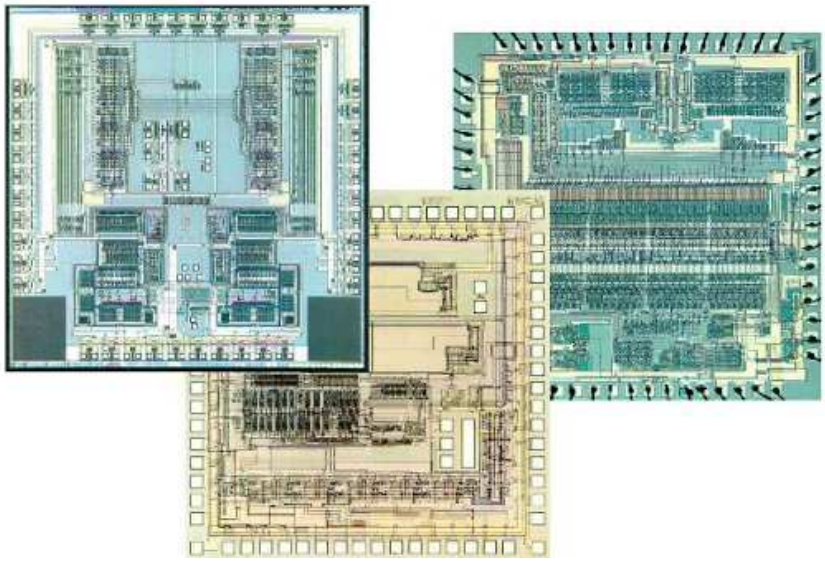


Figura 3: Vista interna de unos ASIC's

La gran ventaja de este tipo de circuitos diseñados a medida es su reducido tamaño, lo que conlleva menor coste por unidad. Además, en algunos casos pueden conseguirse mejoras en el rendimiento. Sus desventajas son el aumento de coste y tiempo de desarrollo y la necesidad de software CAD más complejo y equipo de diseño más cualificado.

Concluyendo, se puede afirmar que los ASIC son una alternativa interesante para dispositivos electrónicos que vayan a ser fabricados en serie, ya que la inversión en el diseño del circuito será amortizada con la reducción del área de cada unidad. Este método de diseño es rentable únicamente para volúmenes de producción elevados.

2.2 Microcontroladores

Un microcontrolador es un circuito integrado programable, capaz de ejecutar órdenes grabadas en su memoria. Funcionan como pequeños motores de cálculo para cualquier aplicación que requiera la toma de decisiones o la supervisión de un sistema. Normalmente, cuenta con los siguientes componentes:

- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM para contener los datos.
- Memoria tipo ROM/PROM/EPROM para el programa.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (Temporizadores, Puertas serie y paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).
- Generador de reloj que sincroniza el funcionamiento de todo el sistema.

Actualmente existe una gran variedad de microcontroladores disponibles para su integración en productos. Su programación puede llevarse a cabo mediante distintos lenguajes de programación tales como *Assembly, C y C ++.*

La clasificación más importante que podemos hacer se basa en el número de bits (4, 8, 16 ó 32). Si bien los microcontroladores de 16 y 32 bits ofrecen prestaciones mayores a los de 4 y 8 bits, la realidad es que éstos últimos son válidos para la mayoría de aplicaciones y hoy en día dominan el mercado.

A continuación podemos ver una lista con algunos de los microcontroladores más populares en la actualidad:

8048 (Intel): Es el padre de los microcontroladores actuales. Su popularidad se debe principalmente a su precio, disponibilidad y herramientas de desarrollo.

8051 (Intel y otros): Sin duda alguna el microcontrolador más popular en su día. Potente y fácil de programar. Está bien documentado y posee cientos de variantes e incontables herramientas de desarrollo.

80186, 80188 y 80386 EX (Intel): Versiones en microcontrolador de los populares microprocesadores 8086 y 8088. Su principal ventaja es que permiten aprovechar las herramientas de desarrollo para PC.

68HC11 (Motorola y Toshiba): Es un microcontrolador de 8 bits potente y popular con gran cantidad de variantes.

683xx (Motorola): Surgido a partir de la popular familia 68k, a la que se incorporan algunos periféricos. Son microcontroladores de altísimas prestaciones.

PIC (MicroChip): Familia de microcontroladores más usados en la actualidad. Su nombre proviene de las siglas: *Peripheral Interface Controller* (Controlador de interfaz periférico). Algunas de sus características más significativas son:

- Núcleos de CPU de 8/16 bits con arquitectura Harvard modificada.
- Memoria Flash y ROM disponible desde 256 bytes a 256 kbytes.
- Puertos de E/S (típicamente de 0 a 5,5V).
- Temporizadores de 8/16/32 bits.
- Tecnología *Nanowatt* para modos de control de energía.
- Periféricos serie síncronos y asíncronos: USART, AUSART, EUSART.
- Conversores analógico/digital de 8-10-12 bits.
- Comparadores de tensión.
- Módulos de captura y comparación PWM.
- Controladores LCD.
- Periférico MSSP para comunicaciones I²C, SPI, y I²S.
- Memoria EEPROM interna con duración de hasta un millón de ciclos de lectura/escritura.
- Periféricos de control de motores.
- Soporte de interfaz USB.
- Soporte de controlador Ethernet.
- Soporte de controlador CAN.
- Soporte de controlador LIN.
- Soporte de controlador Irda.

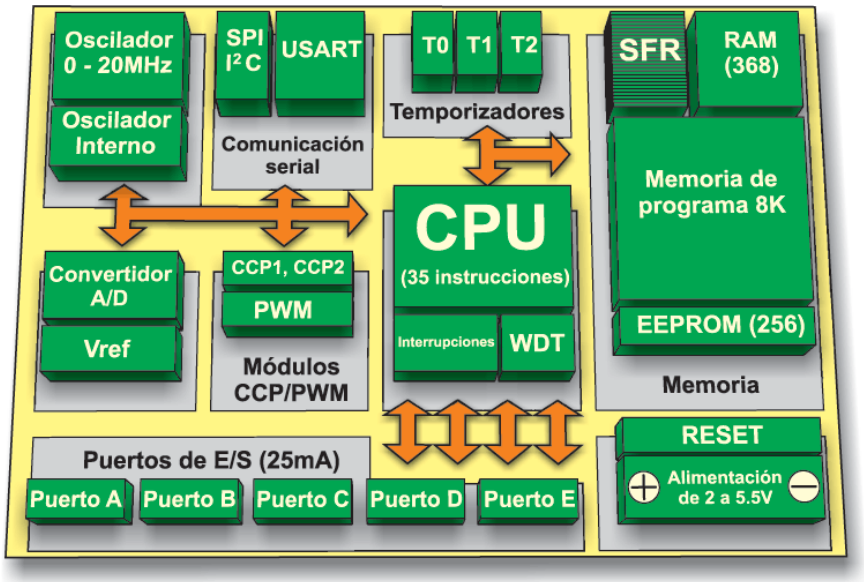


Figura 4: Arquitectura de un PIC16F887

Cabe destacar también la plataforma de hardware libre **Arduino**. Se trata de una placa que incorpora un microcontrolador ATmega junto los puertos de entrada/salida (analógicos y digitales) y el resto de componentes hardware necesarios. Gracias a su bajo coste y a su sencillez, se está convirtiendo en una alternativa muy interesante a la hora de automatizar pequeños montajes. Además, cuenta con un entorno de desarrollo muy sencillo para programar en lenguaje C.

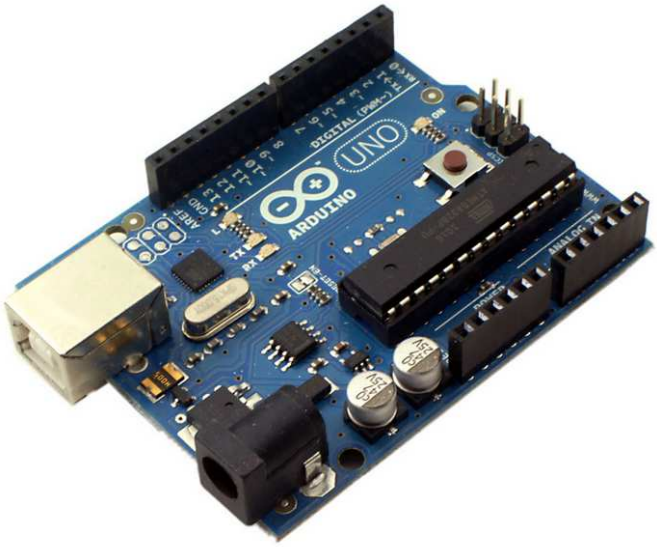


Figura 5: Arduino UNO (microcontrolador ATmega-328)

Implementar el control de este proyecto mediante un microcontrolador habría sido una alternativa perfectamente válida. Su precio reducido y la existencia de entornos que facilitan mucho su programación los convierten en una propuesta a tener en cuenta en cualquier proyecto que requiera algún tipo de automatización. Sin embargo, uno de los objetivos de este proyecto es el estudio de los dispositivos tipo FPGA, por lo que se descarta inmediatamente el uso de cualquier microcontrolador.

2.3 Dispositivos lógicos programables

Un Dispositivo Lógico Programable (PLD) es un componente electrónico estándar usado para construir circuitos digitales. Contienen una arquitectura general predefinida en la que el usuario puede programar la configuración final del dispositivo empleando herramientas de desarrollo. En muchos casos, estos dispositivos cuentan con la posibilidad de ser reprogramados y reconfigurados, lo que permite su reutilización en otros montajes.

Podemos distinguir dos tipos principales de PLDs dependiendo de la manera en que son programados: Dispositivos programables en fábrica y dispositivos programables en campo.

Los **dispositivos programables en fábrica** han de ser programados en la misma fábrica donde se construyen, de acuerdo a los requerimientos del cliente. Esta programación se lleva a cabo mediante procesos irreversibles por lo que no serán posibles futuras modificaciones en el circuito.

En cambio, los **dispositivos programables en campo**, están diseñados para poder ser programados por el usuario. Generalmente también son reprogramables. A continuación podemos ver un esquema con los distintos tipos de dispositivos de lógica programable que podemos encontrar en el mercado:

Figura 6: Clasificación de PLDs.

ROMs: Además de unidades de memoria, pueden funcionar como lógica programable para implementar circuitería combinacional.

SPLD: *(Simple Programmable Logic Devices)* Son los dispositivos más simples, pequeños y baratos que podemos encontrar para implementar circuitos lógicos. A esta familia de dispositivos pertenecen:

- PROM: Permiten implementar cualquier circuito combinacional. En una memoria PROM el valor de cada bit depende del estado de un fusible (o antifusible), que puede ser quemado tan sólo una vez. Esto quiere decir que una vez programada no podrá reprogramarse. Se requiere de un dispositivo especial para programarlas (programador PROM), que aplica pulsos de voltaje superiores a los que se dan en condiciones normales en los fusibles seleccionados por el diseñador.

  La arquitectura de las PROM se basa en un plano fijo de puertas AND (encargado de generar todos los términos producto) que alimenta una matriz programable de puertas OR.

- PAL: Es un dispositivo de matriz programable. Su arquitectura interna se compone de un conjunto de puertas AND con conexiones programables que alimentan un conjunto de puertas OR. Es la estructura usada en SPLD y CPLD por su rapidez y flexibilidad.

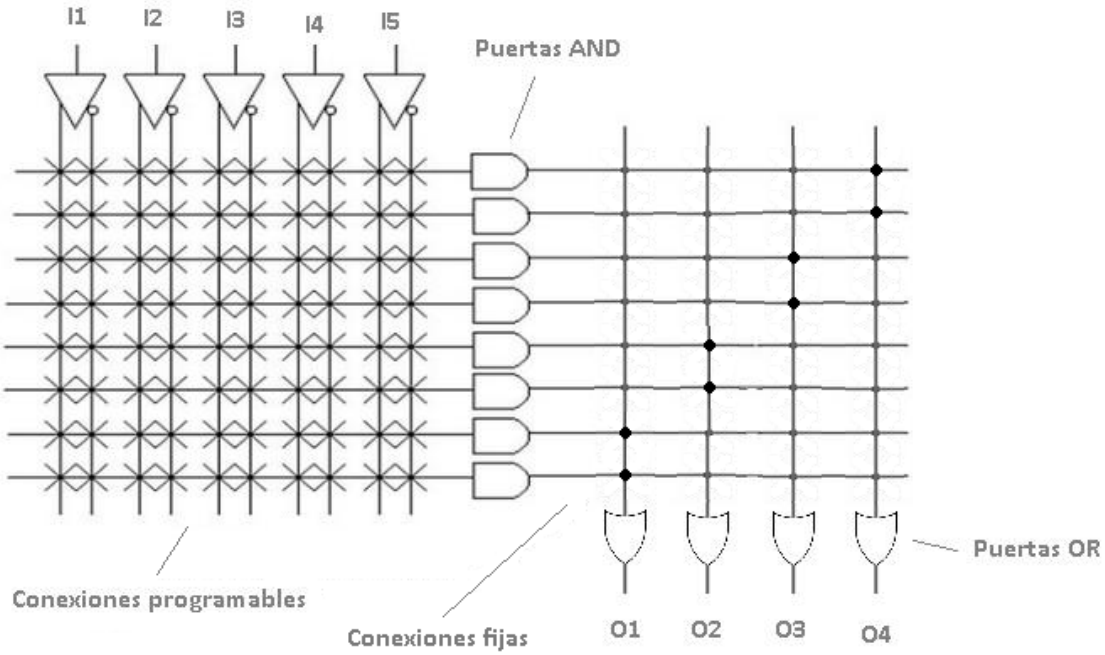


Figura 7: Esquema de conexiones de una PAL.

Existe otro tipo de estructura de programación SPLD que no se usa en dispositivos comerciales, pero sí como bloque funcional en ASIC: la matriz lógica programable (o PLA).

- PLA: Se compone de dos matrices de conexiones que alimentan un conjunto de puertas AND y de puertas OR. En este caso ambas matrices son programables. Es probablemente la estructura que permite mayor flexibilidad a la hora de implementar lógica.

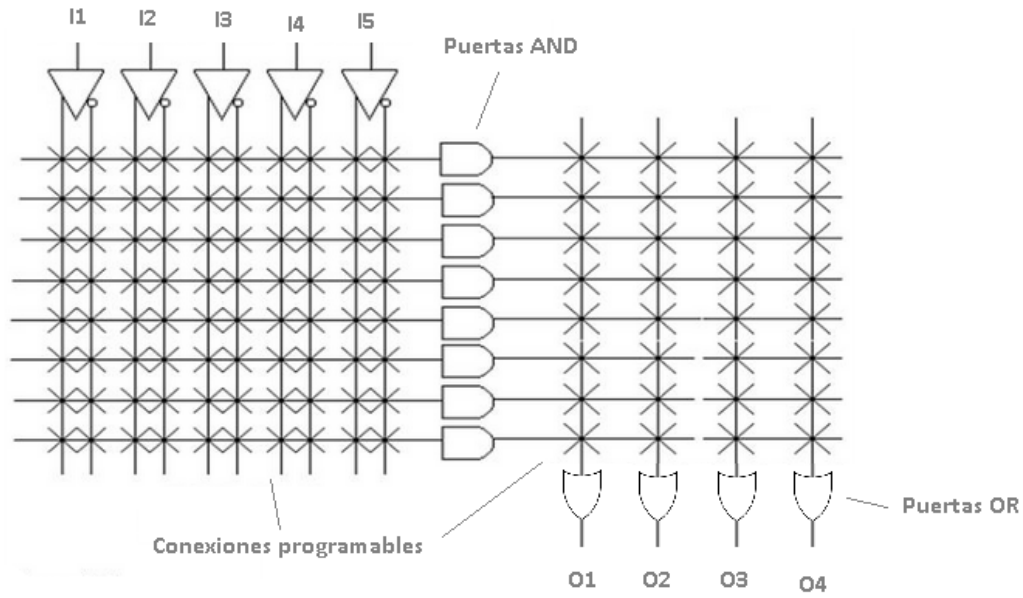


Figura 8: Esquema de conexiones de una PLA.

CPLD: *(Complex Programmable Logic Devices)* Estos dispositivos amplían el concepto de PLD a un mayor nivel de integración. Permiten la creación de circuitos integrados equivalentes a varios PLDs en un único chip. Además, utilizan menor espacio, mejoran la fiabilidad del diseño y reducen costes.

En cuanto a su arquitectura, un CPLD es la unión de una PAL, una macrocelda y un asignador lógico. Los bloques lógicos se comunican entre sí utilizando una matriz programable de interconexiones. Casi cualquier diseño puede ser implementado dentro de estos dispositivos.

FPGA: Son matrices de puertas programables con varios niveles de lógica. Están formadas por bloques lógicos, bloques de entrada/salida e interconexiones. Realizan la lógica en base a memorias, no en base a suma de productos como otros dispositivos. El usuario será capaz de programar tanto la función realizada por cada bloque, como las interconexiones entre ellos.

Se caracterizan por altas densidades de puerta, alto rendimiento, un gran número de entradas y salidas definibles por el usuario, un esquema de interconexión flexible y un entorno de diseño similar al de matriz de puertas.

Se denominan dispositivos de "grano fino" debido a que contienen un gran número de elementos de almacenamiento con relativamente poca lógica asociada a cada uno.

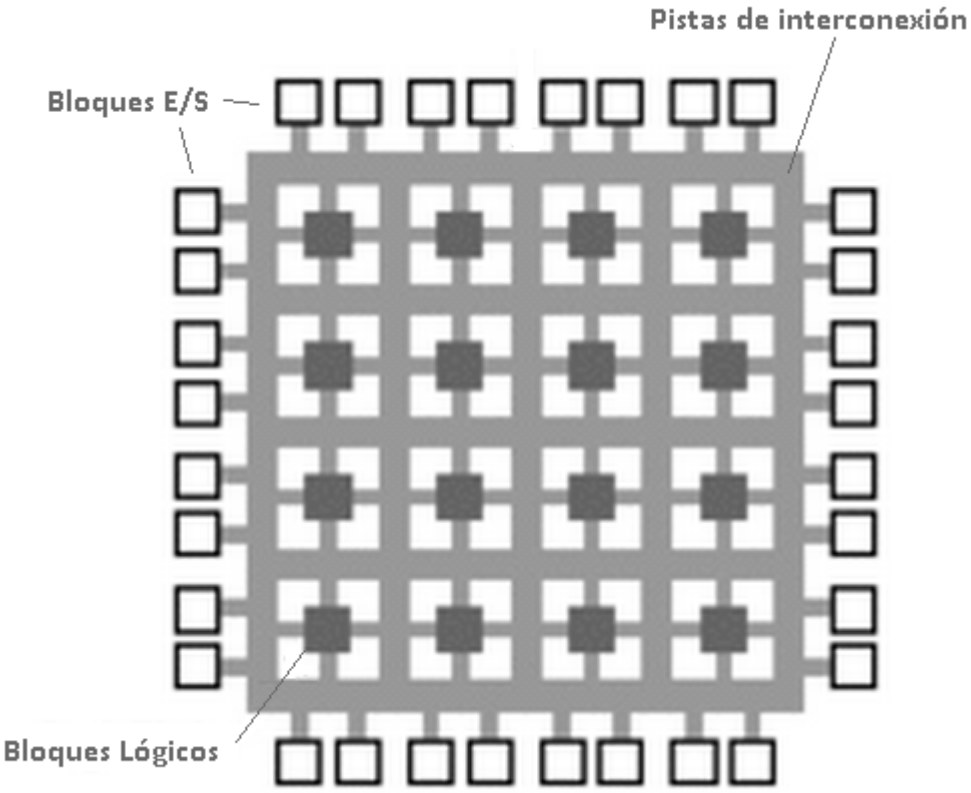


Figura 7: Arquitectura básica de una FPGA.

En comparación con los ASICs, las FPGAs son más lentas, tienen mayor consumo de potencia y no pueden abarcar sistemas tan complejos. Sin embargo, presentan como ventajas sus menores costes de desarrollo y adquisición, su mayor sencillez a la hora de diseñar circuitos y la posibilidad de ser reprogramadas.

Para este proyecto en concreto, debido a las características que se han expuesto, se ha decidido que el control del sistema se lleve a cabo mediante una FPGA.

2.4 Sensores de luz

Existen también varias alternativas para monitorizar el nivel de intensidad luminosa. A continuación se presentan varios dispositivos capaces de realizar esa tarea, indicando sus principales ventajas e inconvenientes:

Fotorresistencia (LDR):

Se trata simplemente de una resistencia cuyo valor resistivo se ve modificado en función de la cantidad de luz que la ilumine. Concretamente, el valor resistivo de este elemento disminuye cuando aumenta la luz incidente, y viceversa. Con el suficiente nivel de luz su valor puede oscilar entre los 50 y 1000 ohms, mientras que en penumbra puede alcanzar los 50k ohms, aproximadamente.

Para monitorizar la intensidad luminosa ambiental bastaría con diseñar un circuito sensible a los cambios de resistencia de este dispositivo y, conociendo su relación, realizar la conversión correspondiente.

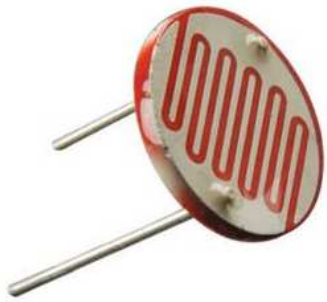


Figura 8: Fotorresistencia o LDR

Sus principales ventajas son su bajo precio y su sencillez de uso.

Fototransistor:

Su apariencia es similar a la de un transistor convencional. Posee las mismas patillas de conexión: Emisor, base y colector. Puede funcionar como un transistor normal o como un elemento fotosensible, siendo la corriente de base generada por la intensidad luminosa incidente. A mayor nivel de iluminación se generará una mayor corriente de base.

Figura 9: Fototransistor

Su principal ventaja es que, debido a la ganancia propia del transistor, es un elemento muy sensible a la luz. Su principal inconveniente es que normalmente es sensible únicamente a luz infrarroja.

Convertidor Luz-Frecuencia:

Este elemento proporciona una señal de onda cuadrada cuya frecuencia varía dependiendo de la cantidad de luz que incide sobre él.

Un ejemplo de este tipo de convertidores es el chip TCS3210. Este sensor posee una matriz de 24 fotodiodos, de los cuales 6 son sensibles a la luz roja, 6 a la luz azul, 6 a la luz verde y 6 a cualquier tipo de luz. Esta matriz genera una corriente u otra dependiendo de la intensidad luminosa incidente sobre ella. Esta corriente pasará por un bloque convertidor corriente-frecuencia que generará la salida final del sensor, como puede verse en el siguiente esquema:

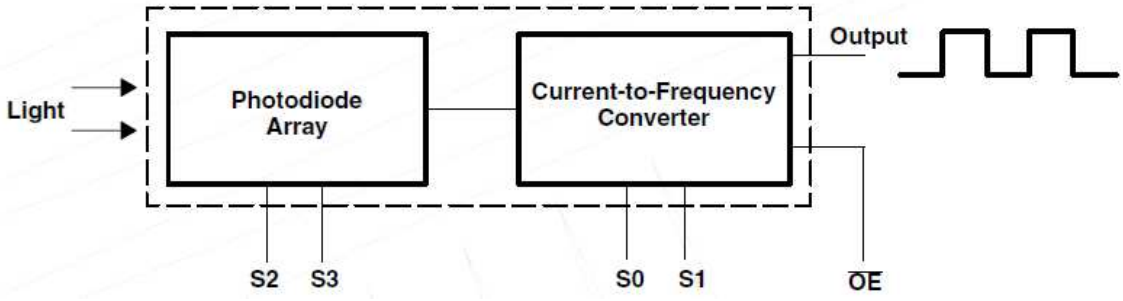


Figura 10: Esquema de funcionamiento del convertidor luz-frecuencia TCS3210

También se puede ver en el esquema que este chip cuenta con cuatro entradas de control: S3, S2, S1 y S0, y una señal de habilitación: OE. Mediante estas señales será posible controlar el tipo de color que se desea percibir y el nivel de frecuencia de la señal de salida.

Este dispositivo tiene, frente a los que se acaban de comentar, la importante ventaja de generar, directamente, una señal digital. Es decir, la señal generada por este sensor puede ser manipulada por un sistema digital, sin necesidad de ninguna circuitería de adaptación adicional.

Como ya se adelantó en el primer capítulo, para este proyecto se ha optado por utilizar este sensor. La posibilidad de elegir el tipo de luz que se desea monitorizar dotará al sistema de una versatilidad que no se lograría con cualquiera de los otros elementos fotosensibles.

Capítulo 3. Estado del arte, selección y breve descripción de los dispositivos

Como ya se indicó en el primer capítulo, una de las metas de este proyecto es el estudio de las posibilidades de los dispositivos de tipo FPGA. En este capítulo se analizarán las características de estos dispositivos, y en concreto del modelo elegido para este proyecto.

3.1 Fabricantes

Dentro del mundo de los dispositivos FPGA, los fabricantes más importantes que podemos encontrar son *Altera, Xilinx y Lattice Semiconductor*. A continuación se analizarán las características que ofrece cada uno de ellos.

Altera:

Una de las empresas pioneras en el desarrollo de dispositivos de lógica programable. Dispone de tres grandes familias de FPGAs. El diseñador podrá elegir una de ellas en función de las exigencias del proyecto que vaya a llevar a cabo:

- *Cyclone Serie:* El modelo más básico de FPGA de Altera. Precio reducido y poder de cálculo moderado. Ideal para aplicaciones en las que el precio es un factor determinante. Dentro de esta familia encontramos un amplio catalogo de modelos que se amoldarán a nuestras necesidades de potencia, memoria, encapsulado, etc.
- *Arria Serie:* Modelo intermedio. Proporciona un balance óptimo entre precio y prestaciones.
- *Stratix Serie:* Modelo de gama alta de Altera. FPGAs de densidad muy alta. Diseñadas para satisfacer las necesidades de los proyectos más exigentes.

Xilinx:

Empresa líder en la investigación y desarrollo de FPGAs. Fue fundada en 1984 por Ross Freeman (inventor de las FPGA), Bernie Vonderscmitt y Jim Barnett.

En cuanto a su oferta, podemos encontrar seis familias ordenadas de menor a mayor por sus prestaciones: *Spartan, Artix, Kintex, Virtex, Kintex UltraSCALE y Virtex UltraSCALE*. A continuación se puede ver una tabla en la que aparecen ordenadas por su densidad de puertas:

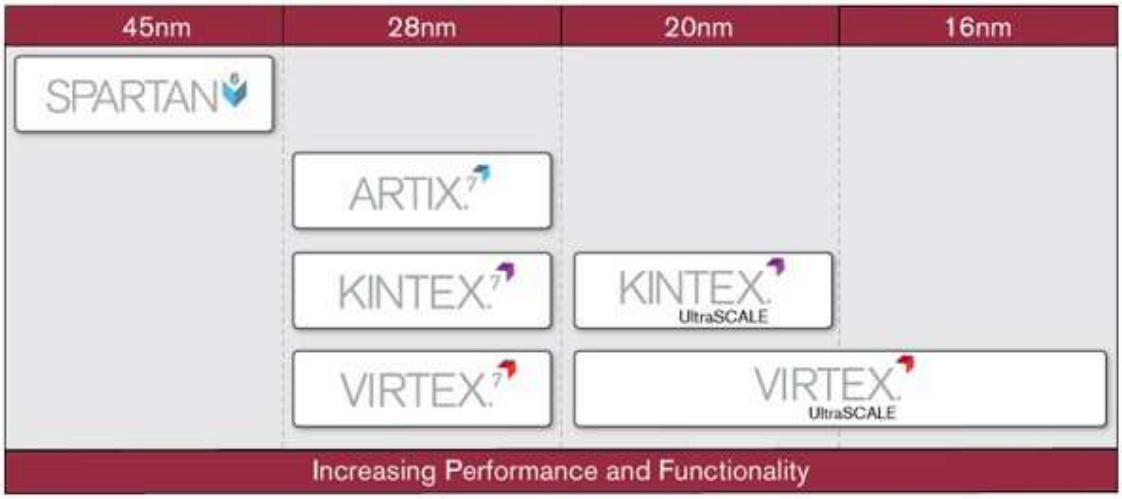


Figura 11: Clasificación de familias de FPGAs Xilinx

Cabe destacar dentro de esta clasificación los modelos *UltraSCALE* de las familias *Kintex* y *Virtex*. Con esta arquitectura se obtiene una considerable mejora de las prestaciones de estos dispositivos. A continuación puede verse una tabla con las principales características de cada una de estas familias:

Range	Kintex UltraScale	Virtex UltraScale
Logic Cells (K)	355–1,160	627–4,407
Block Memory (Mb)	19.0-75.9	44.3–132.9
DSP (Slices)	1,700–5,520	600–2,880
DSP Performance (GMAC/s)	8,180	4,268
Transceivers	16–64	36–120
Peak Transceiver Speed (Gb/s)	16	33
Peak Serial Bandwidth (full duplex) (Gb/s)	2,086	5,886
PCIe® Interface	2–6	2–6
Memory Interface Performance (Mb/s)	2,400	2,400
I/O Pins	312–832	364–1,456
I/O Voltage	1.0–3.3V	1.0–3.3V

Figura 12: Especificaciones de Kintex UltraScale y Virtex UltraScale

Lattice Semiconductor:

Empresa estadounidense fabricante de dispositivos lógicos programables de alto rendimiento (FPGAs, CPLDs y SPLDs). Fue fundada en 1983 en Oregon. Se autodefine como líder mundial en el diseño de circuitos integrados programables para aplicaciones de ultra-bajo consumo de energía como teléfonos inteligentes, control industrial, electrónica de automóvil, etc.

En cuanto a FPGAs, dispone de un amplio catalogo de familias que engloban una inmensidad de modelos distintos. A continuación se presentan las principales características de cada una de estas familias:

- *ECP5 FPGA Family:* Dispositivo de baja densidad, poco consumo y precio asequible. Ideal para aplicaciones con alto volumen de producción en las que el coste es un factor determinante.
- *iCE40 FPGA Family:* Otro dispositivo con rendimiento y precio equilibrados. Disponible en tres series: Bajo consumo (LP), bajo consumo con IP embebido (LM) y alto rendimiento (HX).
- *MachX03 FPGA Family:* Dispositivos no volátiles. Poseen la mayor densidad de puertos E/S de los dispositivos de bajo coste. Encapsulados de hasta 2,5mm x 2,5mm. Ofrecen un rendimiento mejorado de los interfaces MIPI DSI/CSI-2.
- *MachX02 FPGA Family:* Dispositivo perfecto para la implementación rápida de sistemas de control. Ofrece fiabilidad a bajo precio.
- *LatticeECP3 FPGA Family:* Solución eficiente para aplicaciones con requisitos muy altos. Incluye PCIe, HDMI, CPRI, JESD204, GbE y XAUI.
- *ispMACH 4000ZE FPGA Family:* Diseñada para aplicaciones móviles de ultra-bajo consumo. Corriente de standby típica de 10µA.

Como conclusión, se puede afirmar que cualquiera de estas marcas dispone de dispositivos que podrían satisfacer con creces las necesidades de este proyecto. Sin embargo, cabe destacar que las FPGAs de *Lattice Semiconductor* cuentan con una ventaja frente a sus competidores: la no volatilidad de su configuración. Esta es una característica importante, ya que el sistema que utilizan otras marcas para cargar la información en la memoria del dispositivo en el momento del arranque, es relativamente accesible y por lo tanto susceptible de ser copiado.

3.2 Modelo elegido y software utilizado

Como ya se adelantó en el primer capítulo, el modelo elegido para el control de este proyecto es una FPGA **Lattice modelo MachX02-1200ZE**, ya que es el dispositivo del que se dispone en el Departamento de Tecnología Electrónica. A continuación se explica el significado de su nomenclatura:

- *MachX02:* Familia del dispositivo.
- *1200:* Capacidad lógica de 1280 LUTs.
- *Z:* Dispositivo de bajo consumo.
- *E:* Voltaje de alimentación de 1,2V.

Se encuentra integrado en una placa *Breakout Board* que facilita mucho la elaboración de prototipos sencillos.

Otra ventaja que presenta este dispositivo, es que Lattice proporciona una un software gratuito muy completo para programar sus dispositivos. Se trata del programa *Laticce Diamond*, que permite entre otras cosas el diseño, síntesis, análisis, programación, debugger, simulación e implementación de esquemas y circuitos electrónicos. Concretamente, para este proyecto se ha usado la versión 3.1.0.96.

A continuación se desarrollan más detalladamente las características del dispositivo FPGA, de la placa *Breakout Board* y del software de diseño.

Características del dispositivo:

En cuanto a la arquitectura, la familia MachX02 se compone de una matriz de bloques lógicos rodeados por E/S programables (PIO). Contienen también bloques sysCLOCK PLLs y bloques sysMEM RAM embebidos (EBRs).

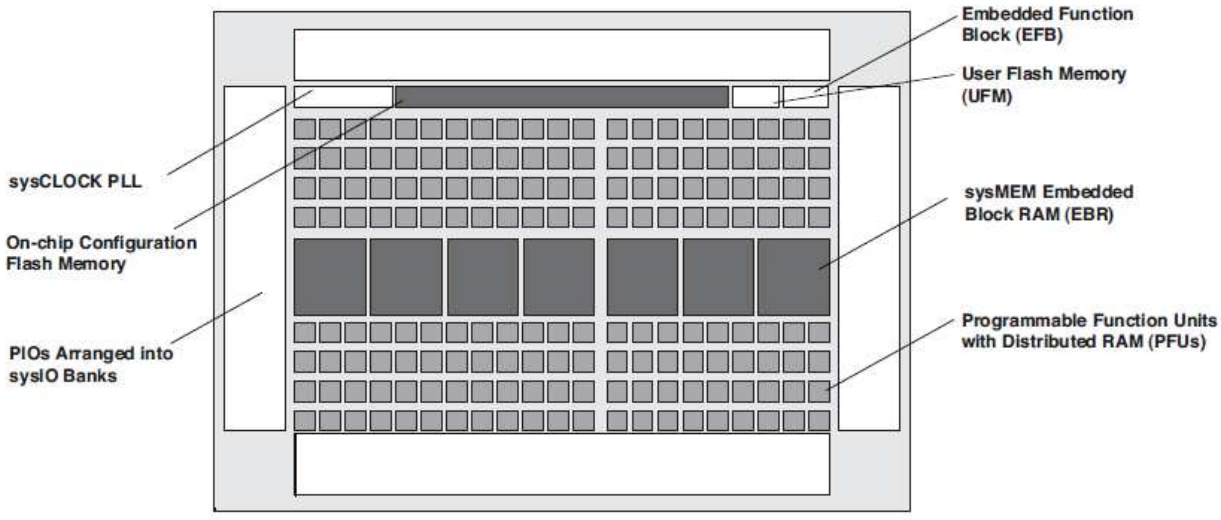


Figura 13: Arquitectura básica de la familia de dispositivos MachX02-1200

Los bloques lógicos, Unidad Funcional Programable (UFP) y sysMEM EBR, están situados en una cuadricula de dos dimensiones (filas y columnas). Cada fila puede tener bloques lógicos o bloques de EBR. Las células PIO están situadas en la periferia del dispositivo. La UFP contiene los bloques correspondientes a la lógica, aritmética, RAM, ROM y registros. Las PIO utilizan un buffer de E/S flexible compatible con varios interfaces estándar. Los bloques se interconectan entre sí mediante pistas horizontales y verticales. La herramienta de diseño software se encarga de asignar automáticamente estos recursos de enrutamiento.

Todos los dispositivos MachX02 incluyen un módulo EFB *(Embedded Function Block).* Este bloque permite implementar funciones de control que facilitan el diseño y permiten ahorrar recursos de uso general como LUTs, registros, señales de reloj y rutado. Contiene los siguientes elementos:

- Dos núcleos I^2C
- Un núcleo SPI
- Un *timer/counter* de 16 bits
- Interfaz para configurar las características dinámicas PLL
- Interfaz para configurar la lógica
- Interfaz para controlar la potencia del chip
- Memoria flash de usuario (UFM)

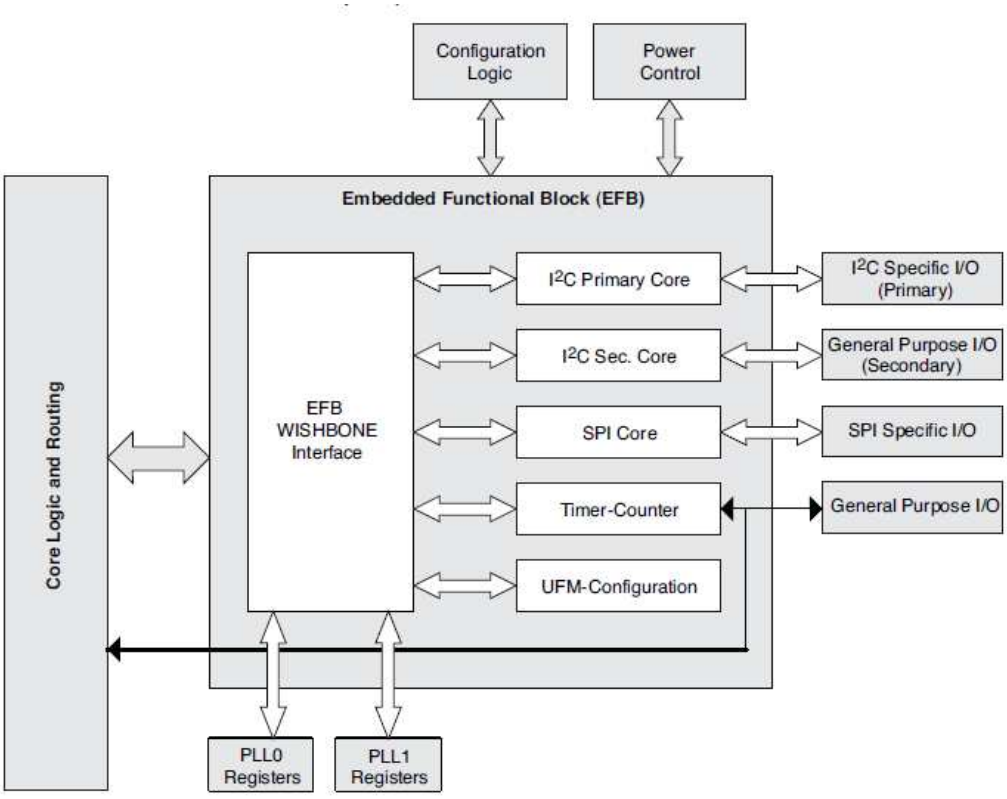


Figura 14: Arquitectura del bloque EFB

La conectividad entre el núcleo lógico de nuestro dispositivo y el bloque EFB, así como la comunicación entre las distintas funciones individuales del mismo, se lleva a cabo mediante el bus WISHBONE, como puede verse en el esquema de la arquitectura de este bloque (Figura 14).

Mediante la herramienta *IPexpress*, el usuario será capaz de crear y configurar un bloque EFB que implemente las funciones que necesite. Se generará un modulo Verilog o VHDL que podrá ser usado en el diseño de proyectos.

La estructura de este interfaz sigue el modelo maestro-esclavo. El usuario deberá diseñar su propio maestro para que interactúe con el esclavo.

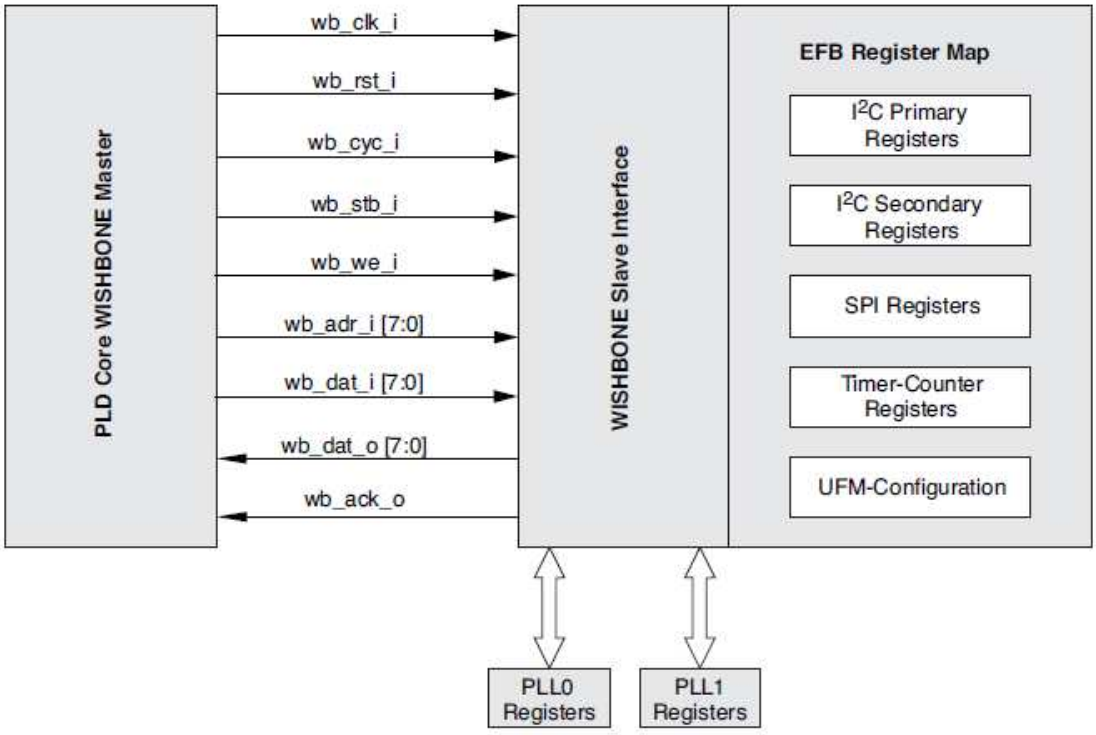


Figura 15: Estructura maestro-esclavo del interfaz WISHBONE

Siendo las señales de comunicación entre el maestro y el esclavo:

- *wb_clk_i:* Señal de reloj. Soporta velocidades de hasta 133MHz.
- *wb_rst_i:* Señal síncrona de reset, activa a nivel alto.
- *wb_cyc_i:* Se activa para indicar el inicio de un ciclo de lectura/escritura.
- *wb_stb_i:* Se activa para indicar el inicio de un ciclo de lectura/escritura.

- *wb_we_i:* Nivel bajo indica lectura, nivel alto escritura.
- *wb_adr_i[7:0]:* Señal de 8 bits que indica la dirección del registro.
- *wb_dat_i[7:0]:* Señal de 8 bits que contiene el byte a escribir.
- *wb_dat_o[7:0]:* Señal de 8 bits que recibe el byte leído.
- *wb_ack_o:* Indica que el ciclo lectura/escritura a finalizado.

Ciclo de escritura en el WISHBONE bus:

En un ciclo de escritura el WISHBONE maestro escribe un byte de datos en el bloque EFB. Se requieren al menos tres ciclos de reloj para completar esta operación. La forma de onda de cada una de las señales para llevar a cabo esta operación es la siguiente:

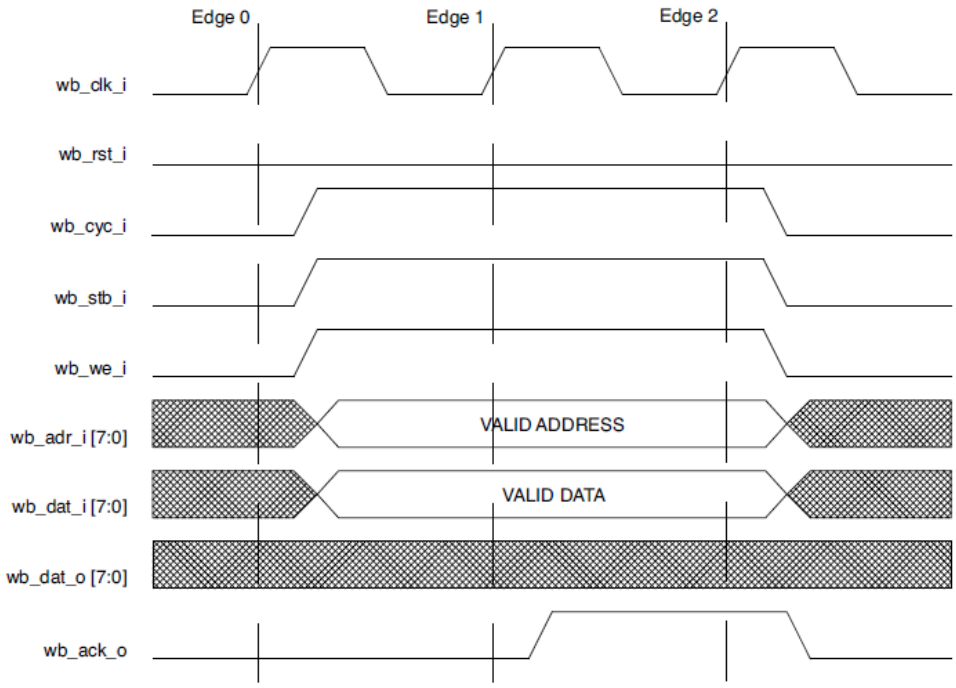


Figura 16: Operación de escritura en el WISHBONE bus

Ciclo de lectura en el WISHBONE bus:

De manera análoga, en un ciclo de lectura se lee un byte de datos del bloque EFB. Se requieren al menos tres ciclos de reloj para completar esta operación. La forma de onda de cada una de las señales para llevar a cabo esta operación es la siguiente:

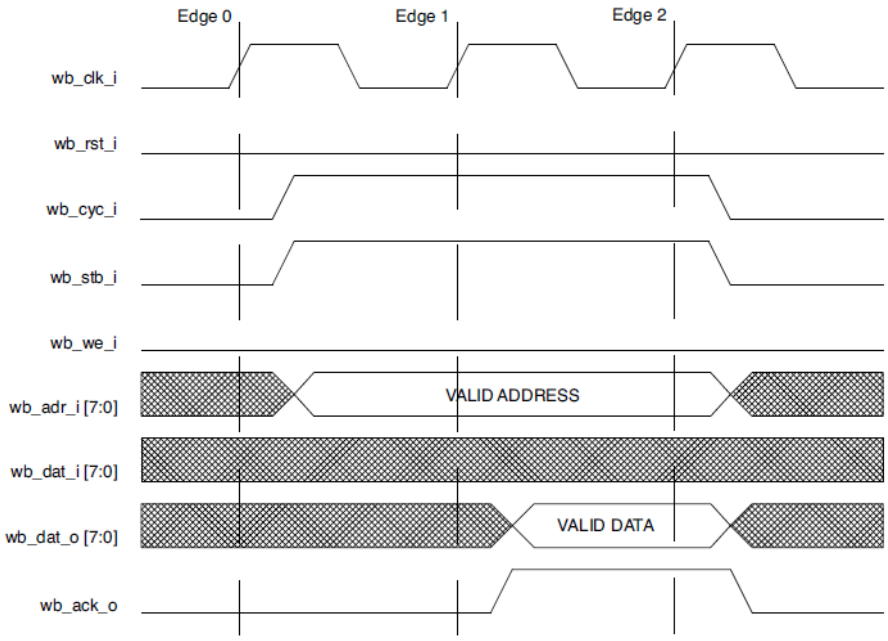


Figura 17: Operación de lectura del WISHBONE bus

De todas las funciones que ofrece el modulo EFB, la única que se usará en este proyecto es la del *Timer/Counter PWM*. A continuación se analizaran detalladamente sus características.

Timer/Counter:

Este modulo proporciona un temporizador/contador de 16 bits de propósito general. Es bidireccional y cuenta con salida independiente. Permite además implementar control por modulación de ancho de pulso (PWM).

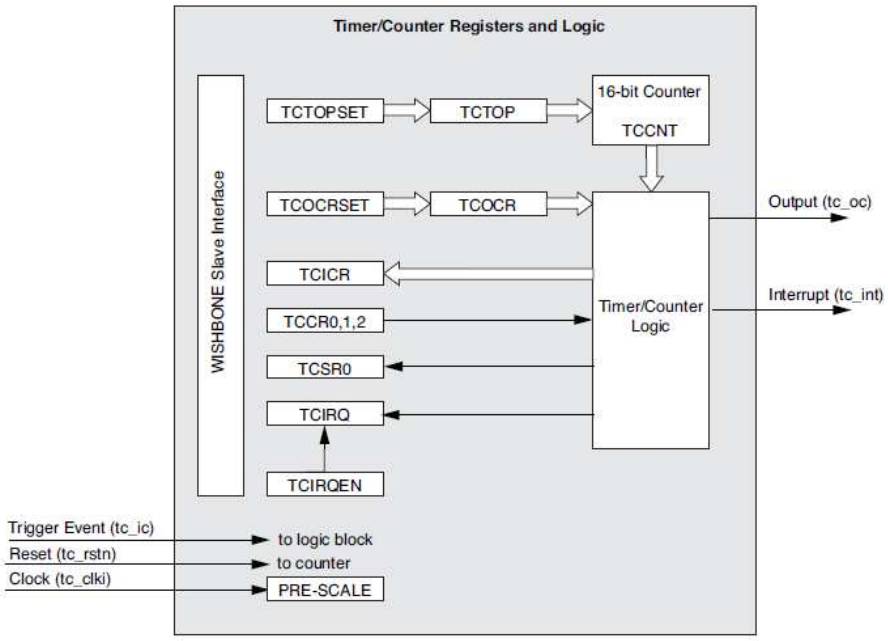


Figura 18: Arquitectura del Timer/Counter

Dispone de señales especificas para comunicarse con el núcleo lógico de nuestro dispositivo a través del WISHBONE bus, aunque existe también la posibilidad de incluirlo en el diseño sin necesidad de usar este interfaz. Tiene cuatro modos de funcionamiento:

- ***Clear Time on Compare Match:*** El contador se reinicia a 0x0000 cuando el valor del registro TCCNT alcanza el valor cargado en el registro TCTOP. El valor del registro TCTOP puede ser constante o actualizado dinámicamente a través del WISHBONE bus. Por defecto, su valor es de 0xFFFF.
- ***Watchdog timer:*** Los contadores Watchdog se usan para controlar posibles bloqueos en los procesos de sistemas operativos o microcontroladores. Si se detecta que el tiempo de ejecución de una tarea supera el tiempo estimado, el *Watchdog timer* ejecuta una interrupción.
- ***Fast PWM:*** La salida del Timer/Counter se pone a nivel bajo cuando el contador alcanza el valor cargado en el registro TCTOP (valor máximo). Se pone a nivel alto cuando el contador alcanza el valor cargado en el registro TCOCR (valor de comparación). De esta manera se obtiene una señal en la que es posible controlar el tiempo que está a nivel alto y a nivel bajo (control PWM).

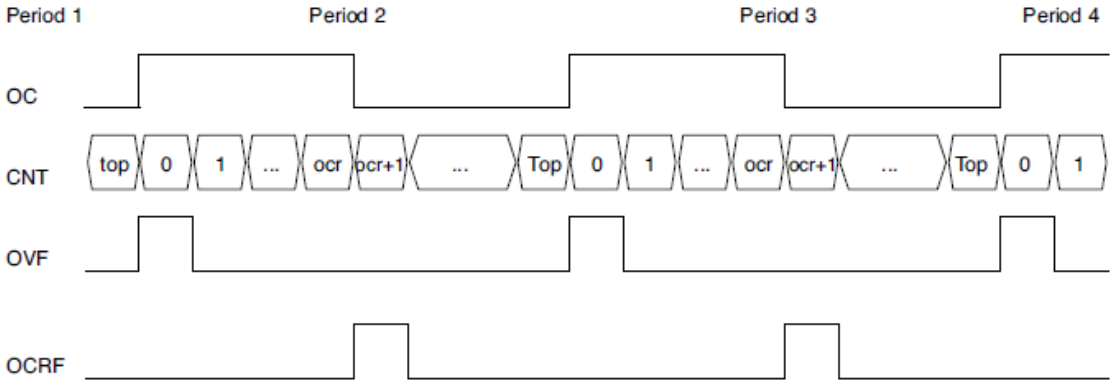


Figura 19: Generación de la forma de onda en modo Fast PWM

(Esta forma de onda se puede invertir, de manera que el valor alto pase a ser bajo y viceversa).

- ***Phase and Frequency Correct PWM:*** Proporciona otra manera de conseguir una señal de ancho de pulso variable. En este caso el contador funciona de manera ascendente y descendente, lo que permite ajustar también la frecuencia y la fase de la onda.

Toda esta información ampliada puede encontrarse en manual del dispositivo: *MachX02 Family Handbook*, que puede ser descargado de manera gratuita en la página web del fabricante.

Características de la placa:

Como ya se dijo en la introducción del capítulo, nuestro dispositivo FPGA está montado en una placa de prototipo *Breakout Board.* Se trata de una placa simple y de bajo coste que simplifica considerablemente el trabajo con FPGAs. Cada uno de los pines de E/S del dispositivo está conectado a un orificio de 100 milésimas de pulgada, lo que permite al usuario acceder a éstas de una manera sencilla.

El tamaño de esta placa es de 3x3 pulgadas, aproximadamente 7,5x7,5 cm. Cuenta además con una matriz de LEDs, una parte libre para el desarrollo de prototipos y un puerto USB mini-B, que sirve para programar el dispositivo y como toma de alimentación. En la siguiente imagen puede verse la localización de estos elementos dentro de la placa.

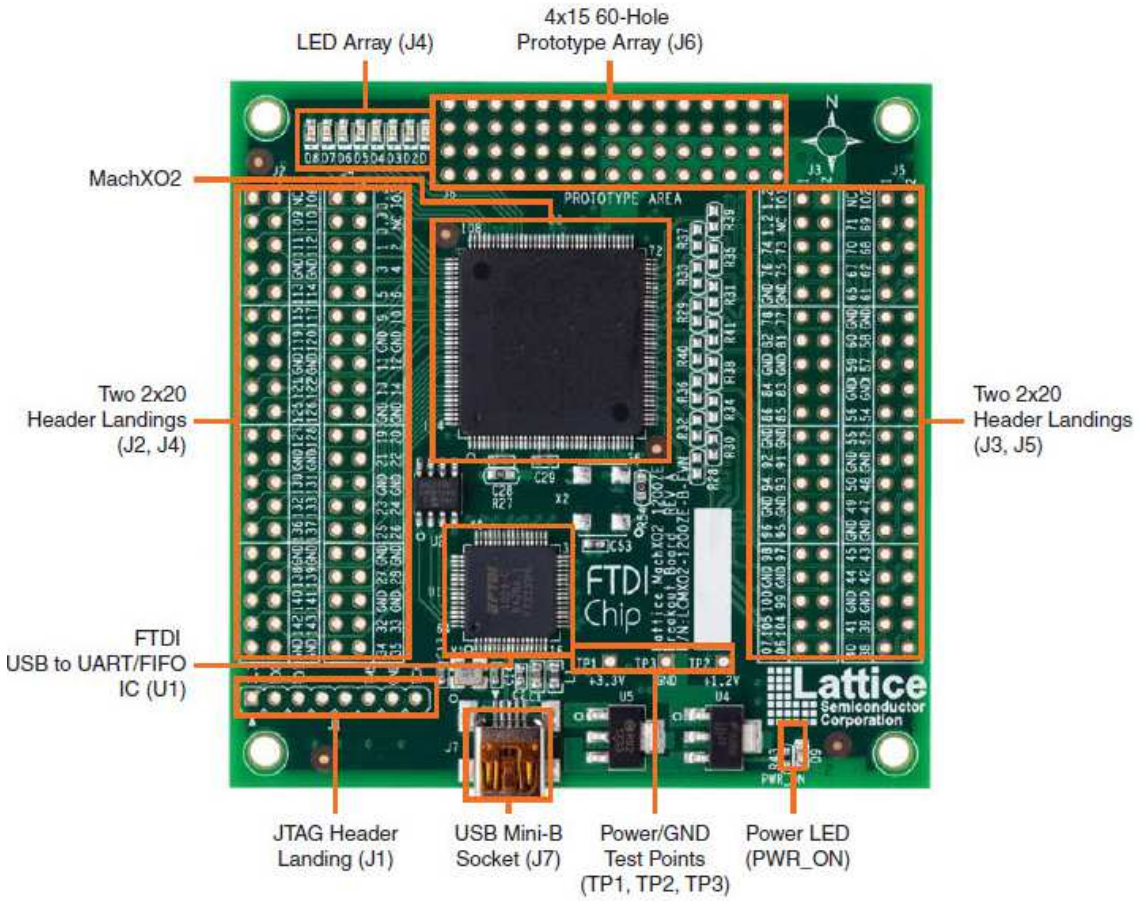


Figura 20: Placa Breakout Board MachX02 de Lattice

Para poder usar esta placa es necesario tener instalados los drivers del *FTDI Chip,* que se pueden descargar de manera gratuita en la página web de Lattice.

Características del software de diseño:

El programa que se usará es el *Laticce Diamond*, en concreto la versión 3.1.0.96, que podemos descargar de manera gratuita de la pagina web de *Laticce Semiconductor.* Además del programa podemos encontrar su manual de usuario: *"Lattice Diamond User Guide",* así como una amplia colección de ejemplos y tutoriales.

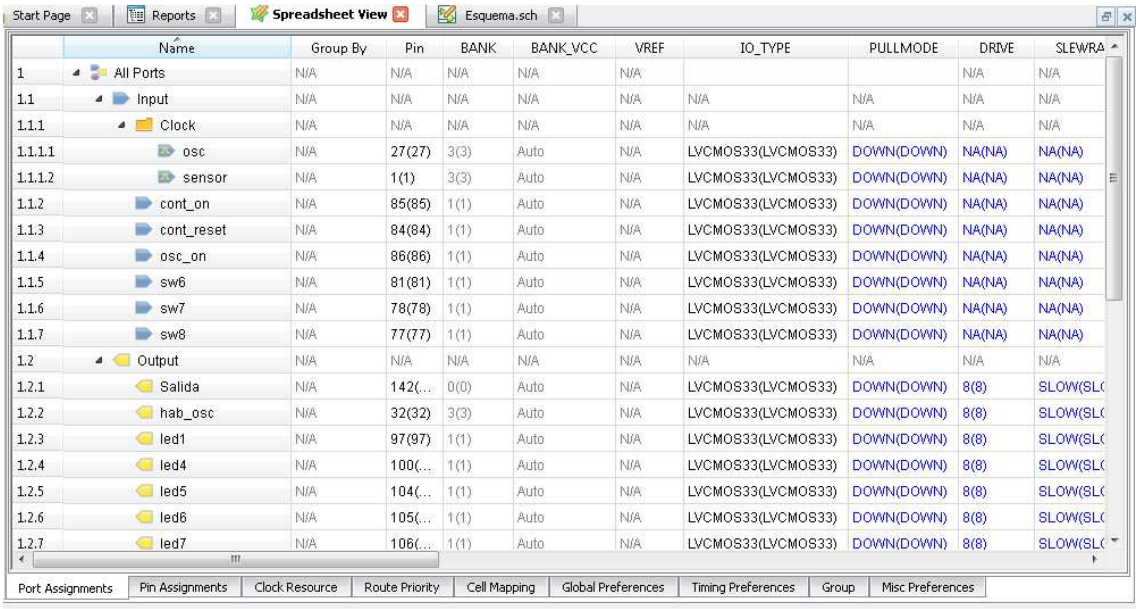
Lattice Diamond ofrece un entorno de desarrollo basado en proyectos, incorporando además herramientas de diseño, simulación y síntesis. Permite trabajar de una manera estructurada y jerárquica.

Una característica interesante de este programa es que pueden usarse técnicas de diseño mixtas. Es decir, podemos generar bloques mediante descripción VHDL que posteriormente podrán ser usados en el esquema de cualquier proyecto.

A continuación, se hará un análisis más detallado de las principales herramientas usadas en este proyecto. En cada caso, se indicarán las páginas del manual de usuario del programa donde se puede ampliar esta información:

Spreadsheet View:

Proporciona un interfaz desde el que visualizar y modificar distintas restricciones de diseño. En el caso de este proyecto, se ha utilizado principalmente para la asignación de puertos en la placa. Esta función muestra una lista con todas las señales del circuito permitiendo modificar los parámetros de cada una. *[Lattice Diamond User Guide, pág. 81]*



	Name	Group By	Pin	BANK	BANK_VCC	VREF	IO_TYPE	PULLMODE	DRIVE	SLEWRA
1	All Ports	N/A	N/A	N/A	N/A	N/A			N/A	N/A
1.1	Input	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1	Clock	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1.1	osc	N/A	27(27)	3(3)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.1.2	sensor	N/A	1(1)	3(3)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.2	cont_on	N/A	85(85)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.3	cont_reset	N/A	84(84)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.4	osc_on	N/A	86(86)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.5	sw6	N/A	81(81)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.6	sw7	N/A	78(78)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.1.7	sw8	N/A	77(77)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	NA(NA)	NA(NA)
1.2	Output	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.2.1	Salida	N/A	142(...	0(0)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.2	hab_osc	N/A	32(32)	3(3)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.3	led1	N/A	97(97)	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.4	led4	N/A	100(...	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.5	led5	N/A	104(...	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.6	led6	N/A	105(...	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL
1.2.7	led7	N/A	106(...	1(1)	Auto	N/A	LVCMOS33(LVCMOS33)	DOWN(DOWN)	8(8)	SLOW(SL

Figura 21: Vista de la herramienta "Spreadsheet View"

IPexpress:

Esta herramienta recoge una amplia colección de modelos funcionales que pueden ser usados para generar bloques Verilog o VHDL personalizados, listos para incluir en nuestro diseño. Están optimizados para la arquitectura de los dispositivos Lattice, por lo que su uso conlleva una mejora en el rendimiento y la velocidad de nuestro circuito. Una vez creado un módulo, basta con incluir en nuestro proyecto el fichero IPX que genera la herramienta para poder trabajar con él. *[Lattice Diamond User Guide, pág. 89]*

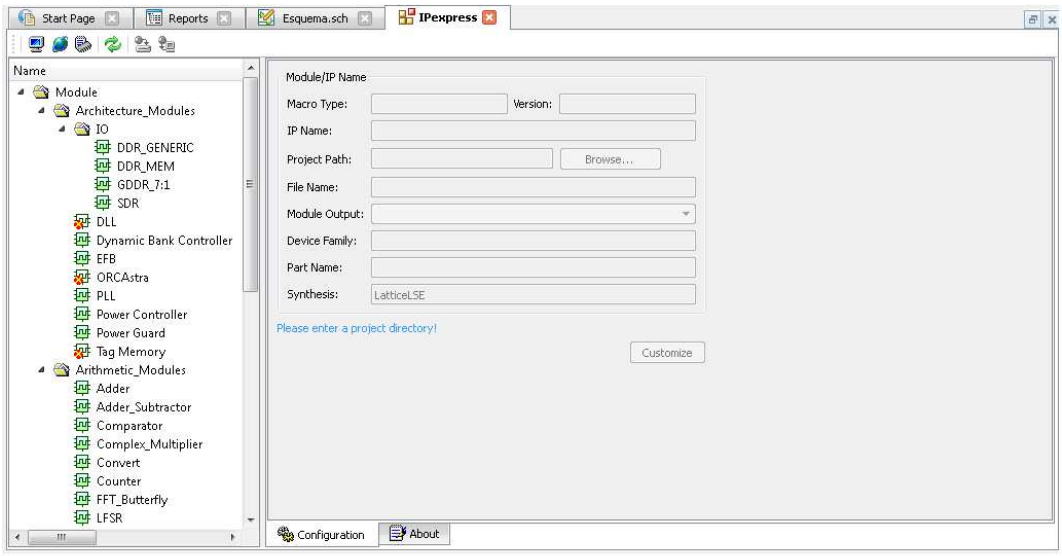


Figura 22: Vista de la herramienta "IPexpress"

Programmer:

Una vez finalizado el diseño y generado el fichero JEDEC, esta herramienta permite transferir la información del proyecto a nuestro dispositivo Lattice. *[Lattice Diamond User Guide, pág. 108]*

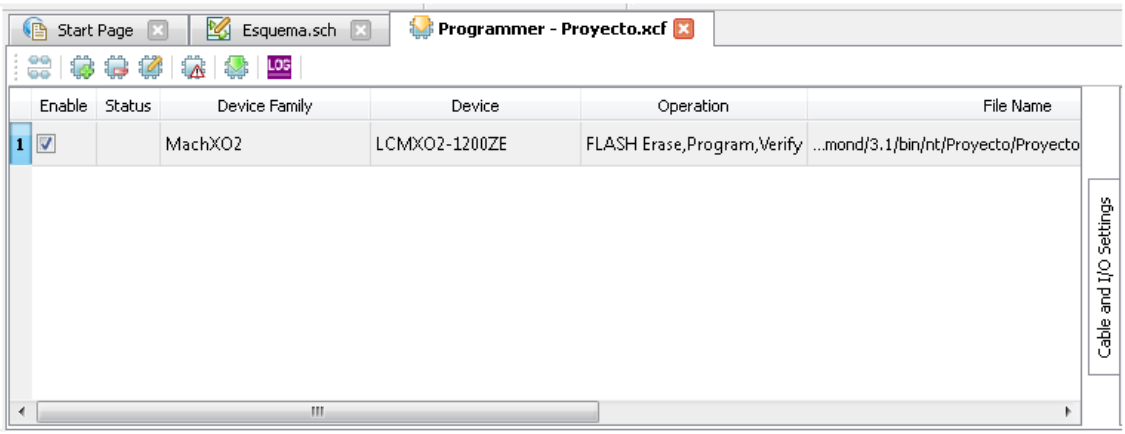


Figura 23: Vista de la herramienta "Programmer"

Active-HDL Lattice Edition:

Esta herramienta no pertenece al entorno de desarrollo del proyecto como tal. Se trata de un simulador OEM que permite comprobar el correcto funcionamiento de cada módulo por separado o del conjunto completo. *[Lattice Diamond User Guide, pág. 111]*

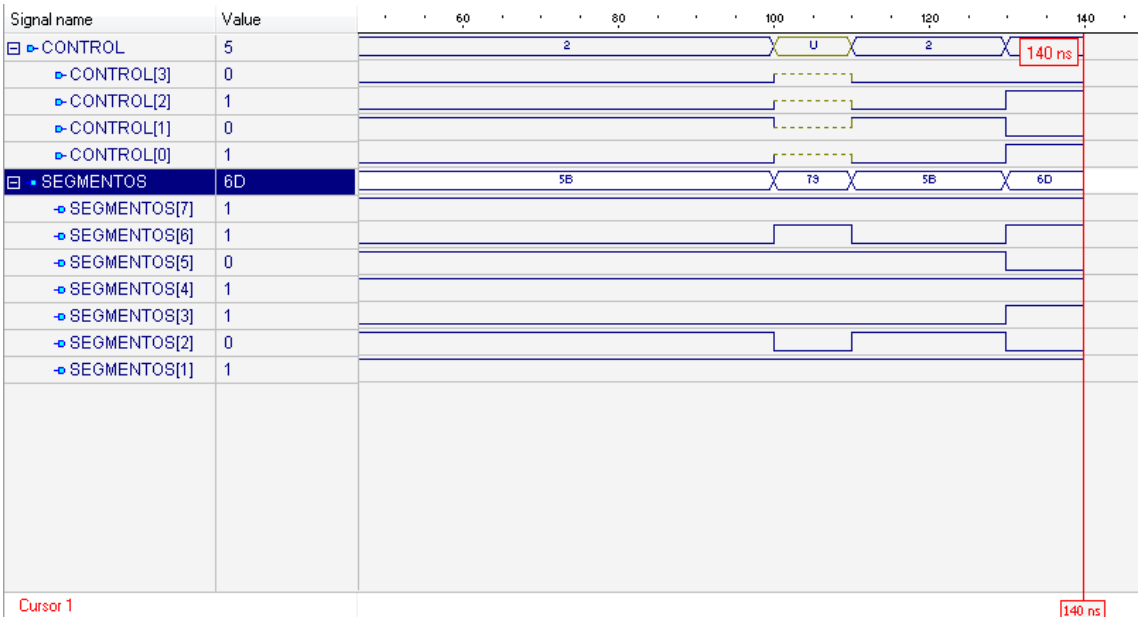


Figura 24: Simulación del módulo de control del display de 7 segmentos

Capítulo 4. Implementación

En este capítulo se aborda paso por paso el procedimiento llevado a cabo para la implementación de la aplicación en el dispositivo FPGA. Se explica detalladamente cada una de las etapas necesarias para el desarrollo del sistema, tanto en la parte de hardware como en la de software.

4.1 Desarrollo de las placas de ampliación

Como se adelantó en el capítulo 1, se diseñaran y fabricarán tres placas PCB de ampliación para añadir al sistema ocho micro interruptores, ocho pulsadores, dos displays de 7 segmentos y el sensor convertidor luz-frecuencia.

Placa PCB Display:

Esta placa contiene los dos displays de 7 segmentos, así como las 16 resistencias necesarias para limitar la intensidad en sus entradas. También se encargara de dar soporte a los pines auxiliares para la placa de ampliación del convertidor luz-frecuencia. A continuación puede verse el esquema de conexiones de esta placa:

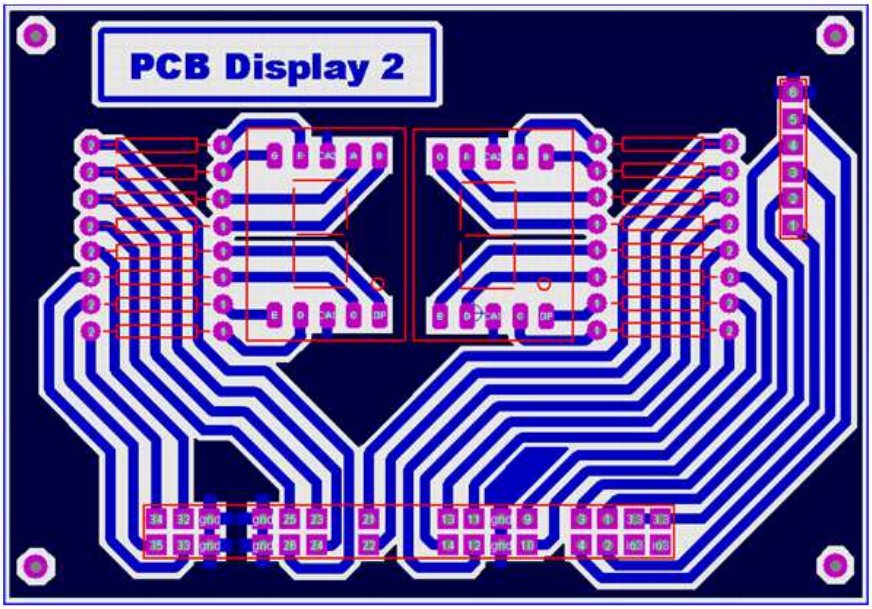


Figura 25: Esquema de conexiones de la placa PCB Display

En el modelo de displays utilizados, la conexión entre los ocho diodos responde a un esquema de conexión en ánodo común:

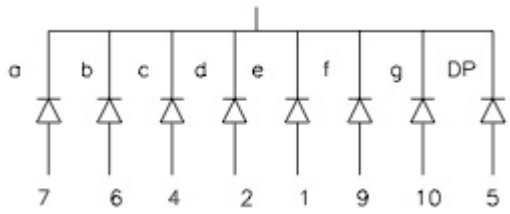


Figura 26: Conexión en ánodo común

Cada pin de entrada de los displays estará relacionado con una señal de salida de la FPGA, siendo su relación la que puede verse en la siguiente tabla:

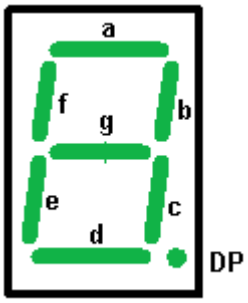


Figura 27: Esquema de segmentos de un display

Segmento	Salidas de la FPGA	
	Display 1	*Display 2*
a	*21*	*23*
b	*22*	*25*
c	*11*	*33*
d	*10*	*34*
e	*13*	*35*
f	*12*	*26*
g	*14*	*24*
dp	*9*	*32*

Una vez fabricada y montada, el aspecto de esta placa será el que puede verse en la siguiente imagen:

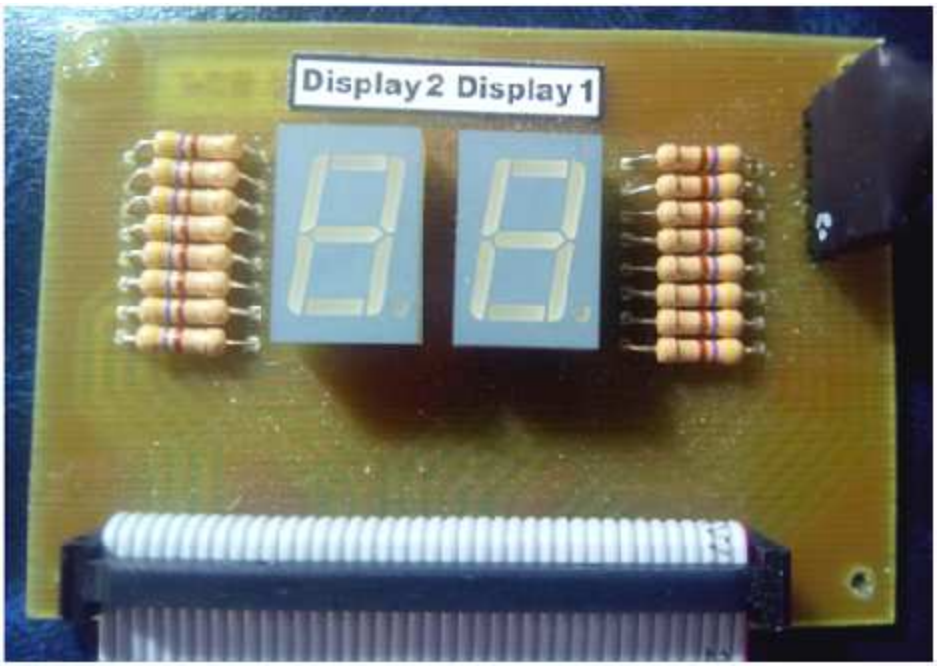


Figura 28: Placa PCB Display 2

Placa PCB Switch y Pulsadores:

Esta placa contiene ocho micro interruptores y ocho pulsadores, que proporcionarán una manera simple de controlar los programas implementados en la FPGA. Su esquema de conexiones puede verse a continuación:

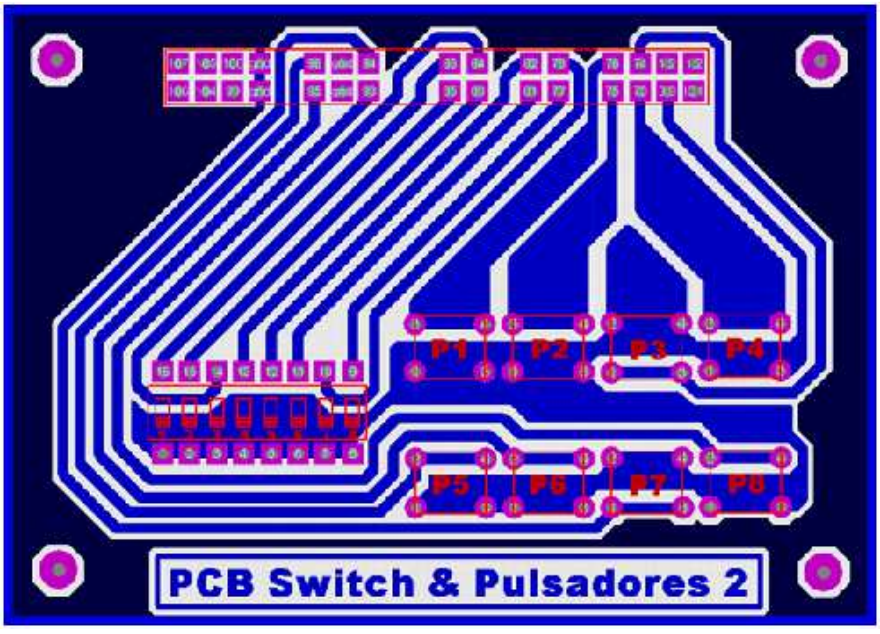


Figura 29: Esquema de conexiones de la placa PCB Switch y Pulsadores

Al subir a 'ON' un micro interruptor o presionar un pulsador, el estado lógico de la entrada correspondiente en la FPGA se pondrá a nivel alto (3.3 V). En caso contrario (interruptor en 'OFF' o pulsador sin pulsar), el valor de la línea se pondrá a nivel bajo mediante un *pull-down* en el interior del dispositivo.

La relación de pines de esta placa con señales de entrada de la FPGA es la siguiente:

Micro Interruptores	Entrada FPGA	Pulsadores	Entrada FPGA
1	*86*	*1*	*76*
2	*85*	*2*	*75*
3	*84*	*3*	*74*
4	*83*	*4*	*73*
5	*82*	*5*	*96*
6	*81*	*6*	*95*
7	*78*	*7*	*94*
8	*77*	*8*	*93*

Una vez fabricada y montada, el aspecto de esta placa será el que puede verse en la siguiente imagen:

Figura 30: Placa PCB Switch y Pulsadores 2

Placa PCB Módulo Luz-Frecuencia:

Se diseñará y fabricará una placa donde irá montado el sensor convertidor luz-frecuencia TCS3210. Este chip permite programar el color y ajustar la frecuencia de salida mediante las señales de control S0, S1, S2 y S3. Se alimenta a 3,3V. Esta placa se conectará al resto del sistema mediante el conector de ampliación de la placa PCB Display. Su esquema de conexiones será el siguiente:

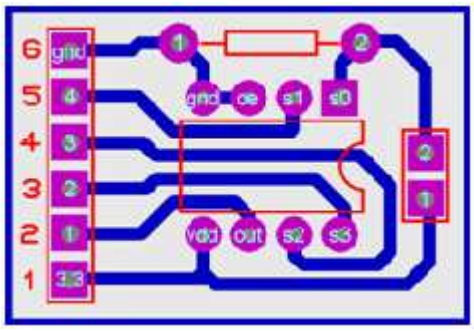


Figura 31: Esquema de conexiones de la placa PCB del convertidor luz-frecuencia

Exceptuando la señal "S0" (que será controlada mediante un jumper en la propia placa), la relación de entradas/salidas de esta placa con los pines de la FPGA es la siguiente:

Entrada/Salida	Señal en la placa	Pines FPGA
1	*Vdc (3,3V)*	*3,3V*
2	*OUT*	*1*
3	*S3*	*2*
4	*S2*	*3*
5	*S1*	*4*
6	*GND*	*GND*

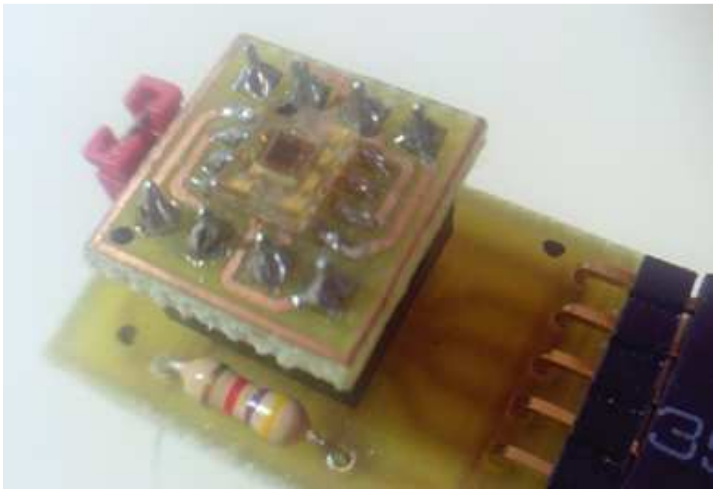


Figura 32: Placa PCB del módulo luz-frecuencia

Reloj externo:

Además de estas tres placas, se añadirá a *la Breakout Board* un oscilador externo de 50 MHz. Proporcionará una señal de reloj más precisa que la del reloj interno de 2,08 MHz del dispositivo. Se habilitará este oscilador mediante el pin 32 de la FPGA (activo a nivel alto). La salida del reloj estará conectada al pin 27 de la FPGA.

4.2 Desarrollo del programa

Como ya se ha comentado, la herramienta de diseño software utilizada en este proyecto permite diseñar de forma estructurada y jerárquica combinando esquemas con bloques funcionales en VHDL. Otra ventaja que presenta es la posibilidad de realizar simulaciones de cada bloque por separado, pudiendo así comprobar el correcto funcionamiento de cada parte del circuito.

A continuación, se explican con detalle cada una de las distintas fases de desarrollo del esquema general de la aplicación:

4.2.1 Convertidor BCD a 7 segmentos:

En primer lugar se generará un bloque con la descripción VHDL de un convertidor de código BCD a display de 7 segmentos. Este bloque se encargará de activar las entradas necesarias de los displays para que muestren el numero BCD correspondiente.El código generado es el siguiente:

```
----------------------------------------------------
------ TRABAJO FIN DE GRADO, curso 2013/14 ---------
---------- Gonzalo A. Garcia Gallego ---------------
-- Descripcion del convertidor BCD a 7 segmentos ---
----------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity bcd_7seg is
    port(
            CONTROL: in STD_LOGIC_VECTOR(3 downto 0);
            SEGMENTOS: out STD_LOGIC_VECTOR(7 downto 1)
            );
end bcd_7seg;

architecture bcd_7seg_arch of bcd_7seg is
    begin
        SEGMENTOS <= "0111111" when (CONTROL="0000") else
                     "0000110" when (CONTROL="0001") else
                     "1011011" when (CONTROL="0010") else
                     "1001111" when (CONTROL="0011") else
                     "1100110" when (CONTROL="0100") else
                     "1101101" when (CONTROL="0101") else
                     "1111101" when (CONTROL="0110") else
                     "0000111" when (CONTROL="0111") else
                     "1111111" when (CONTROL="1000") else
                     "1100111" when (CONTROL="1001") else
                     "1111001";
end bcd_7seg_arch;
```

Para estar seguros de que su funcionamiento coincide con el esperado, someteremos este bloque a una simulación mediante la herramienta *Active-HDL,* en la que comprobaremos cada uno de los posibles casos. En la siguiente imagen podemos ver un ejemplo de esta simulación para los casos 1, 5 y 9:

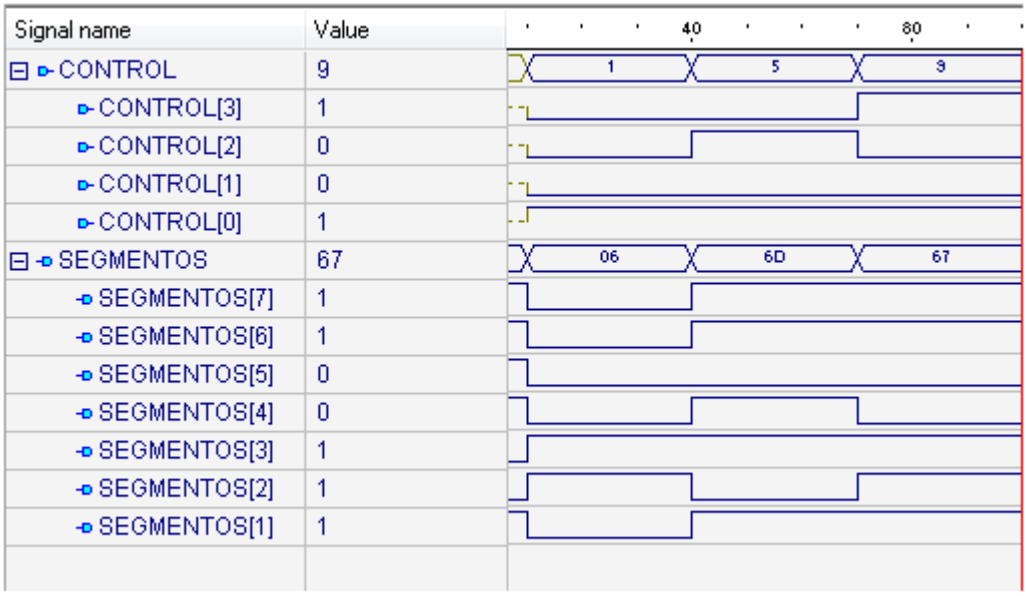


Figura 33: Simulación del bloque convertidor BCD a 7 segmentos

Una vez comprobado que su funcionamiento es correcto, se generará un símbolo que podrá ser usado en el esquema de nuestro circuito:

Figura 34: Bloque funcional del convertidor BCD a 7 segmentos

4.2.2 Oscilador a 1Hz de frecuencia:

El siguiente paso será diseñar un bloque que genere una señal de reloj de aproximadamente 1Hz. Se elige este valor porque permitirá visualizar los cambios producidos en el sistema de manera correcta, con una señal más rápida los cambios en la iluminación de la bombilla podrían producirse de

manera tan gradual que sería difícil percibirlos. Además, se considera que la actualización del valor del display una vez por segundo es más que suficiente.

Para obtener esta señal se utilizará el reloj externo de 50 MHz que se ha agregado la placa y un contador del tamaño adecuado que actúe como divisor de frecuencia.

Concretamente, mediante la herramienta *IPexpress* se generará un contador de 26 bits. El máximo valor en decimal que podrá alcanzar este contador es de 67108863. Siendo la señal de reloj que llega a este contador de 50 MHz, el bit de mayor valor de este contador tendrá una frecuencia de:

$$\frac{67108863}{50000000} = 1{,}34\ Hz$$

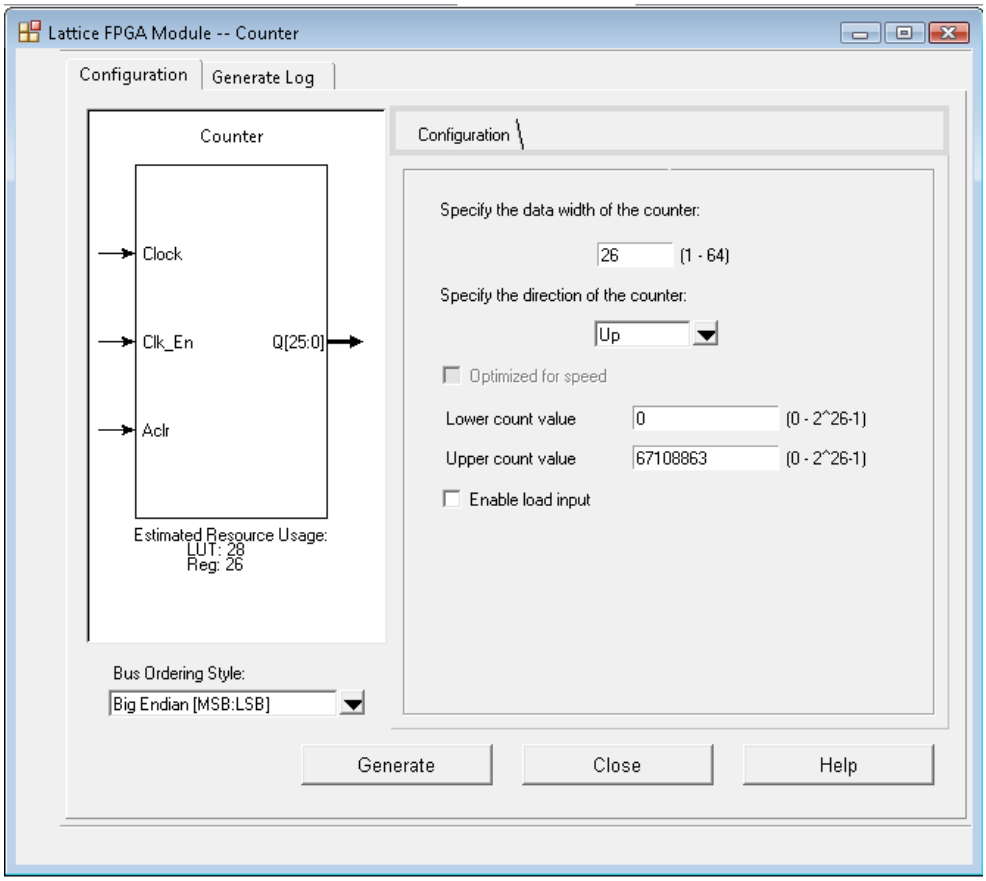


Figura 35: Creación del contador mediante la herramienta IPexpress

En este circuito, la precisión de la señal de reloj no es algo importante, ya que sólo influye en la frecuencia de actualización del valor de los displays y de la iluminación de la bombilla. Para obtener una señal con frecuencia de 1Hz exactamente se tendría que emplear mucha más lógica y el diseño final

resultaría más complejo. Es por ello que se aceptará la aproximación de 1,34 Hz ≃ 1 Hz, (se considera que un tiempo de 0,76 segundos entre actualizaciones es aceptable).

Como puede verse en la siguiente figura, se obtendrá esta señal de reloj tomando el bit de mayor valor de la salida del contador. Utilizaremos también el LED1 de la placa para visualizarla.

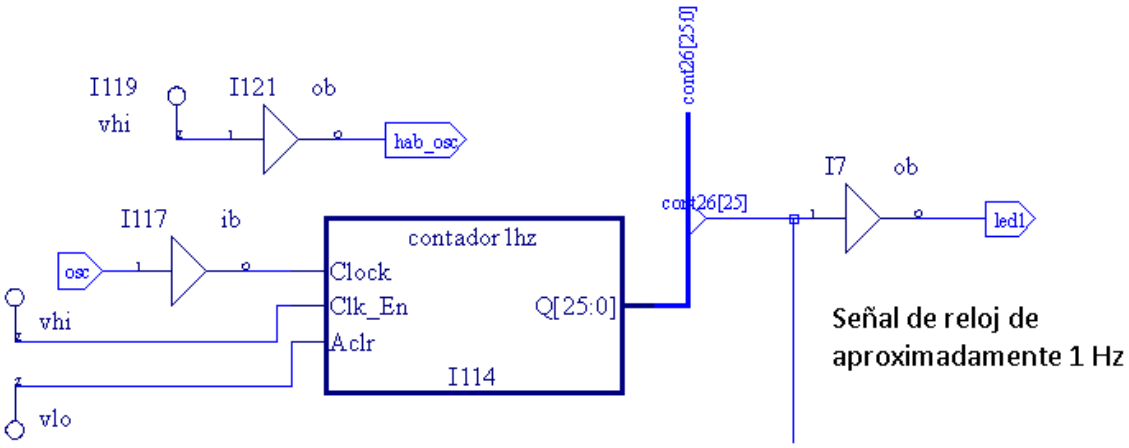


Figura 36: Obtención de la señal de reloj de 1 Hz

Como vemos en el esquema, las señales *Clk_En* (Habilitación de reloj) y *Aclr* (Reset) se conectan a valor lógico alto y bajo respectivamente. La entrada *Clock* recibe la señal del oscilador de 50 MHz.

4.2.3 Contador BCD:

A continuación, partiendo del esquema anterior, implementaremos un contador BCD de un digito que cuente a la frecuencia de aproximadamente 1 Hz que acabamos de generar. Se usará la herramienta *IPexpress* para crear un contador de 4 bits con límite superior de 9. Utilizaremos cuatro leds de la placa para visualizar los cuatro bits de salida de este contador.

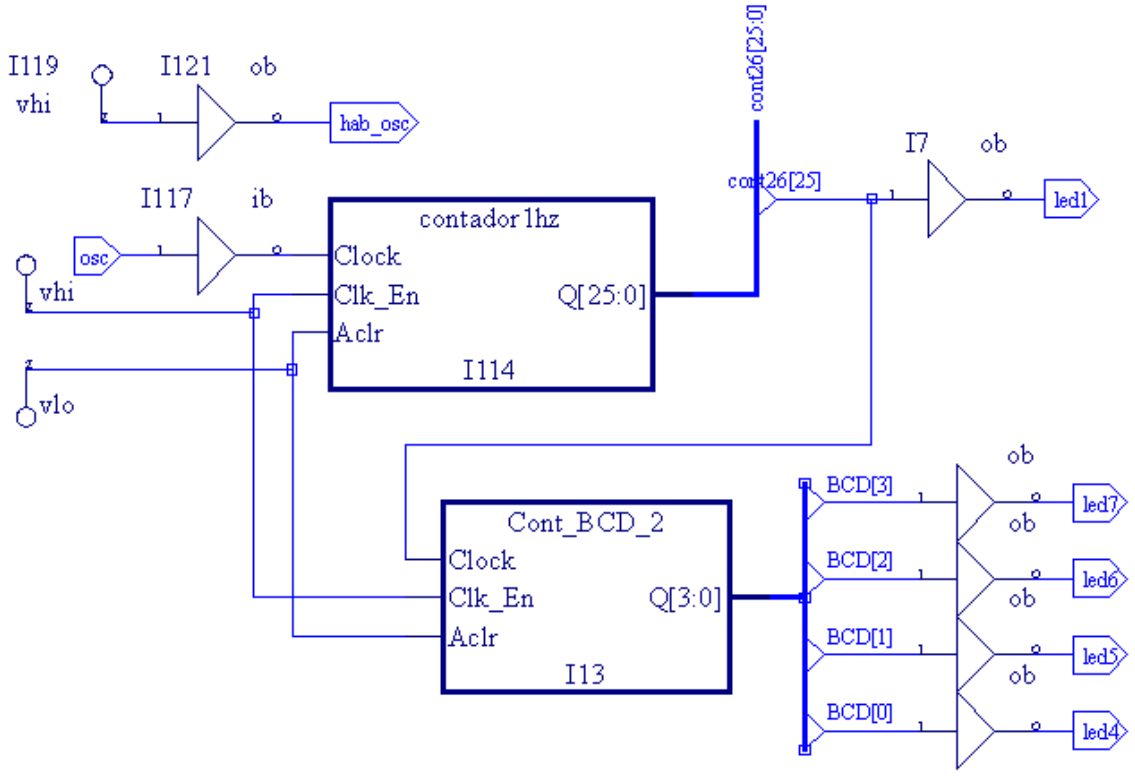


Figura 37: Esquema del contador BCD

En este momento ya es posible transferir el programa al dispositivo FPGA y comprobar que se comporta de la manera esperada. En caso afirmativo, el siguiente paso será visualizar el valor de este contador en uno de los displays de 7 segmentos de los que dispone la placa. Para ello, se usará el convertidor BCD desarrollado en el apartado 4.2.1 de este informe. Asignaremos cada una de sus señales de salida a los pines de entrada correspondientes de uno de los displays de la placa.

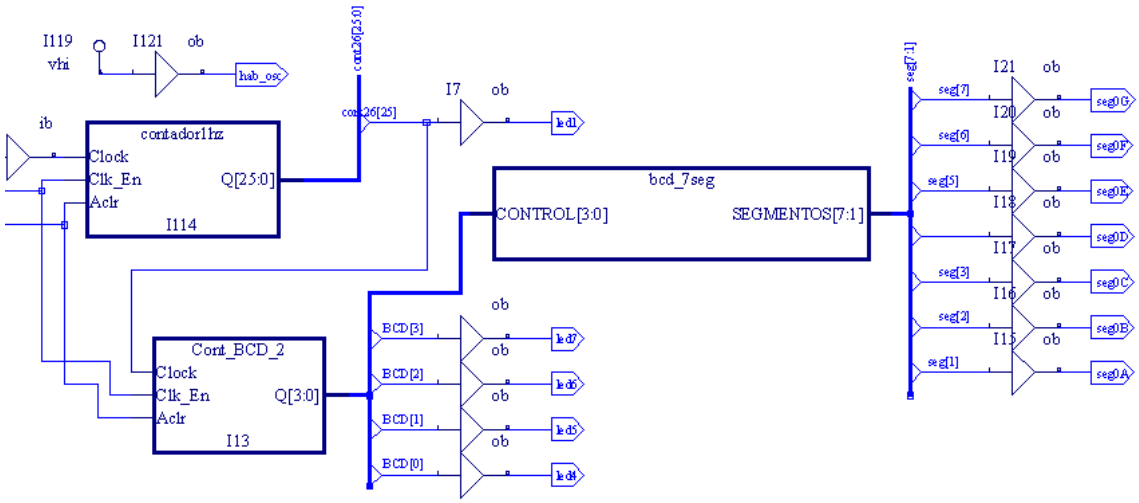


Figura 38: Esquema del contador BCD visualizado mediante display de 7 segmentos

4.2.4 Contador BCD de dos dígitos:

A continuación se modificará el diseño del apartado anterior para obtener un contador BCD de dos dígitos, que será visualizado usando los dos displays de la placa. Para ello, se introducirá otro contador BCD y otro convertidor BCD a siete segmentos como los del apartador anterior. Las señales de reloj y de reset del segundo contador coincidirán con las del primero. Para la entrada de habilitación se diseñará un circuito lógico que hará que la segunda cifra se modifique únicamente en el momento en que la primera cifra pase de 9 a 0, como puede verse en la siguiente figura:

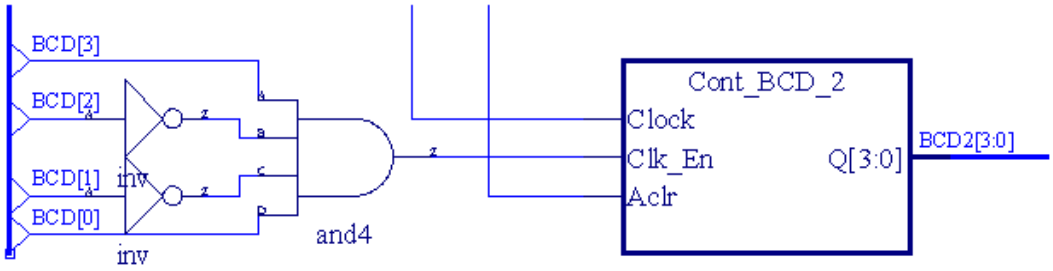


Figura 39: Circuito lógico que habilita el segundo contador BCD

Asignaremos las señales de salida del segundo convertidor BCD a las entradas correspondientes del segundo display. A continuación se puede ver el montaje final con las modificaciones necesarias para obtener un contador BCD de dos dígitos:

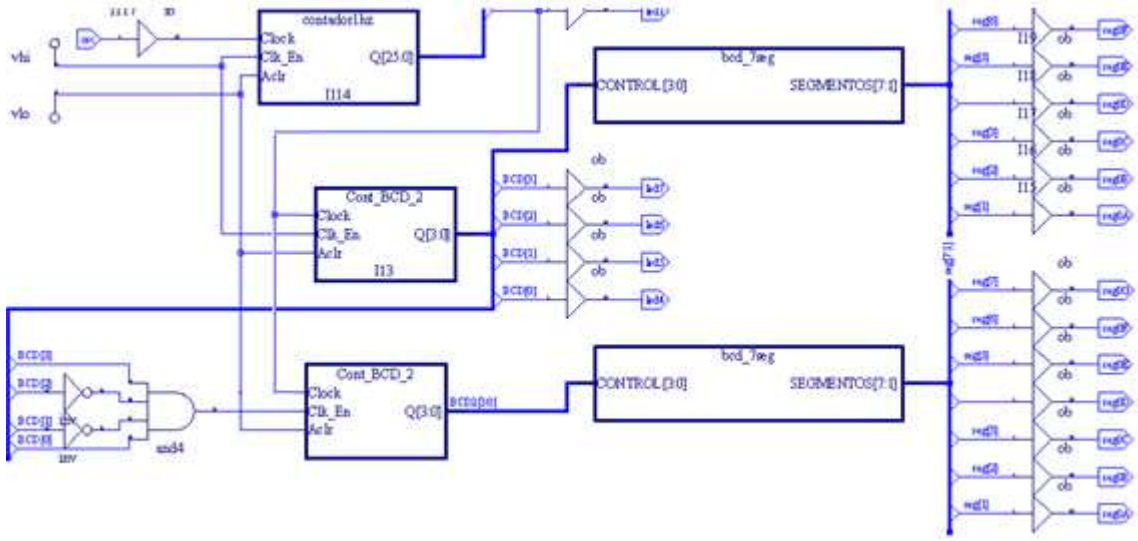


Figura 40: Esquema del contador BCD de dos dígitos

4.2.5 Lectura del sensor de luminosidad:

El objetivo de este proyecto es que los displays muestren el valor de la intensidad luminosa del ambiente, actualizándose cada segundo. Como ya se ha explicado, esta lectura se llevará a cabo mediante un convertidor luz-frecuencia *TCS3210*. Este chip cuenta con cuatro entradas de control: S0, S1, S2 y S3, que permiten elegir el color que se desea percibir y la frecuencia de salida, de acuerdo con las siguientes tablas:

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Figura 41: Comportamiento del sensor TCS3210 en función de sus señales de control

El valor de la señal de control S0 está definido por un jumper en la propia placa, siendo su valor S0='1' si el jumper está cerrado y S0='0' si se retira. Para controlar el resto de señales se usarán tres micro interruptores de la placa, que tendrán que ser definidos en el diseño de la siguiente manera:

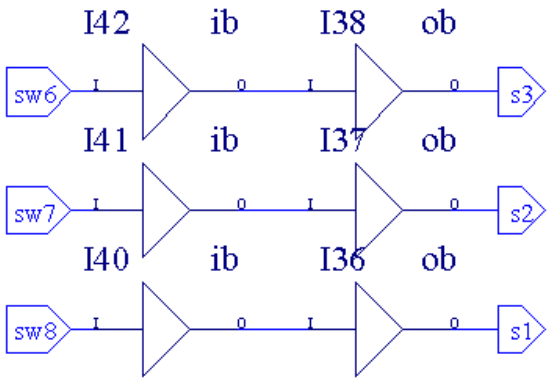


Figura 42: Asignación de micro-interruptores a las señales de control del sensor TCS3210

La línea de salida del sensor se conectará al contador BCD de dos dígitos del apartado anterior, de manera que cuente los pulsos generados. Se diseñará el circuito para que cuando la señal de 1Hz se encuentre a nivel bajo se habilite el contador BCD de dos dígitos, y cuando se produzca el flanco de subida el contador guarde su valor en ocho registros y se resetee.

De esta manera, se consigue que los registros contengan un valor de dos dígitos BCD proporcional al número de pulsos generados por el

convertidor luz-frecuencia durante el intervalo en el que ha estado habilitado el contador. Cuanto mayor sea la intensidad de luz incidente en el sensor, se generaran más pulsos y por lo tanto el valor almacenado en los registros será mayor.

Además se esto, se debe tener en cuenta que la frecuencia del convertidor luz-frecuencia suele oscilar entre los 0 y 60 kHz. Para evitar los desbordamientos que podrían producirse en el sistema, escalaremos la frecuencia antes de llevarla al contador. Para ello se dividirá la frecuencia por 1000 empleando un contador, como puede verse a continuación:

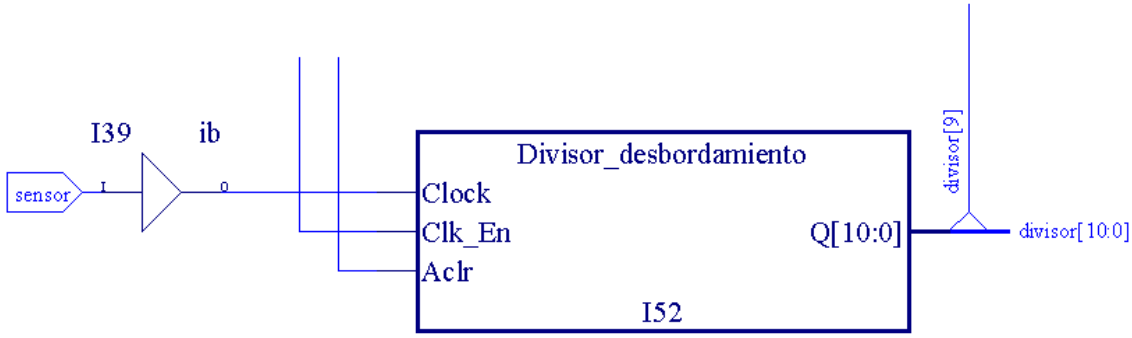


Figura 43: Escalado de la frecuencia del sensor mediante un contador

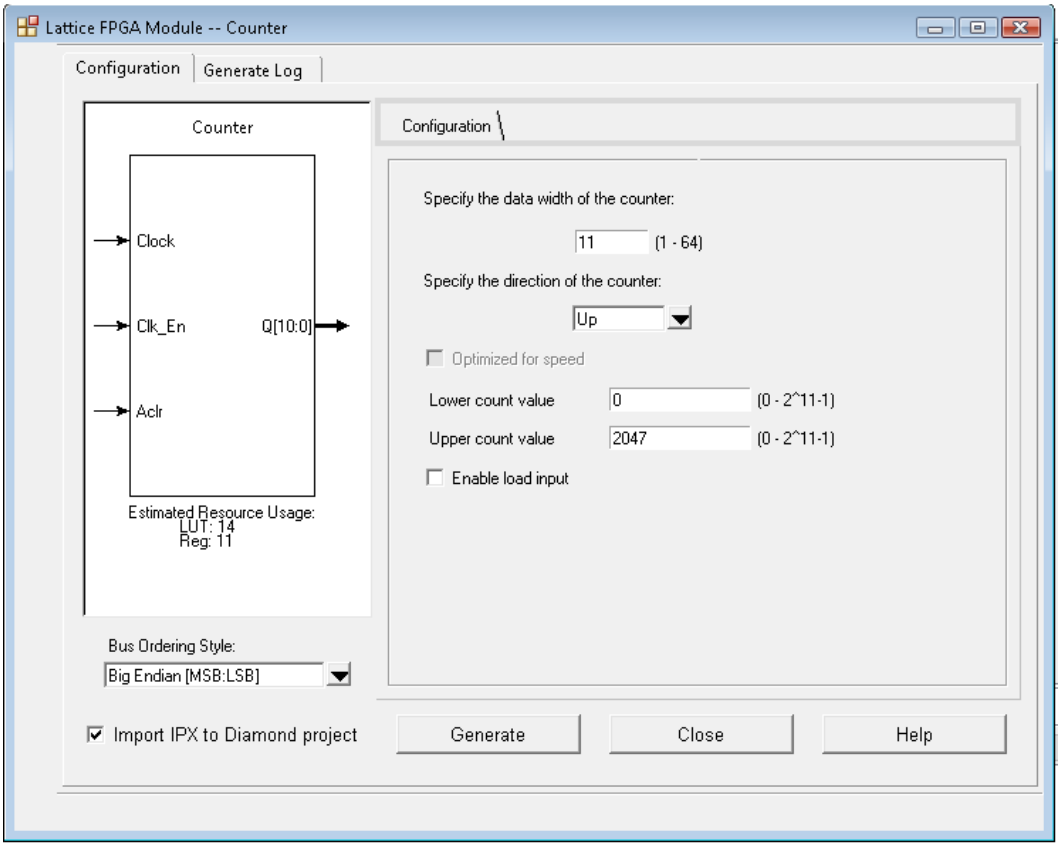


Figura 44: Creación del divisor de desbordamiento mediante IPexpress

El esquema final del diseño, una vez añadidos los registros, las señales de control del sensor y el divisor de desbordamiento, quedaría de la siguiente forma:

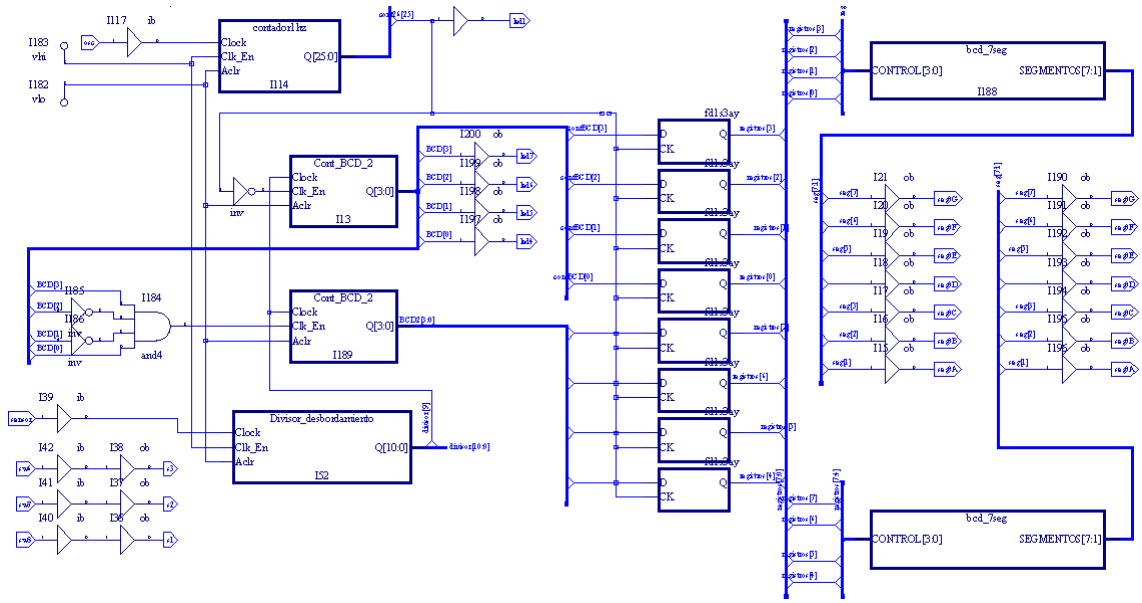


Figura 45: Esquema del diseño incluyendo la lectura del sensor

Una vez hecho esto, se comprueba que el valor de la intensidad luminosa mostrado en los displays varía dependiendo de la incidencia de la luz sobre el sensor. Mediante los interruptores que gobiernan las señales de control del convertidor, se puede configurar el sistema para visualizar la energía incidente en forma de luz roja, verde, azul o blanca.

4.2.6 Implementación del control por PWM:

El siguiente paso a dar en este proyecto es conseguir controlar la intensidad luminosa de un diodo LED en función del valor del convertidor luz-frecuencia, (posteriormente se sustituirá este diodo por una bombilla LED de 12V).

Como se explicó en el apartado 3.2, el dispositivo empleado incluye la posibilidad de implementar un *Fast PWM* configurando a través del *Wishbone bus* el Contador/Temporizador interno del módulo EFB.

Para ello, mediante la herramienta IPexpress se creará un bloque de control del módulo EFB como el que puede verse en la siguiente imagen:

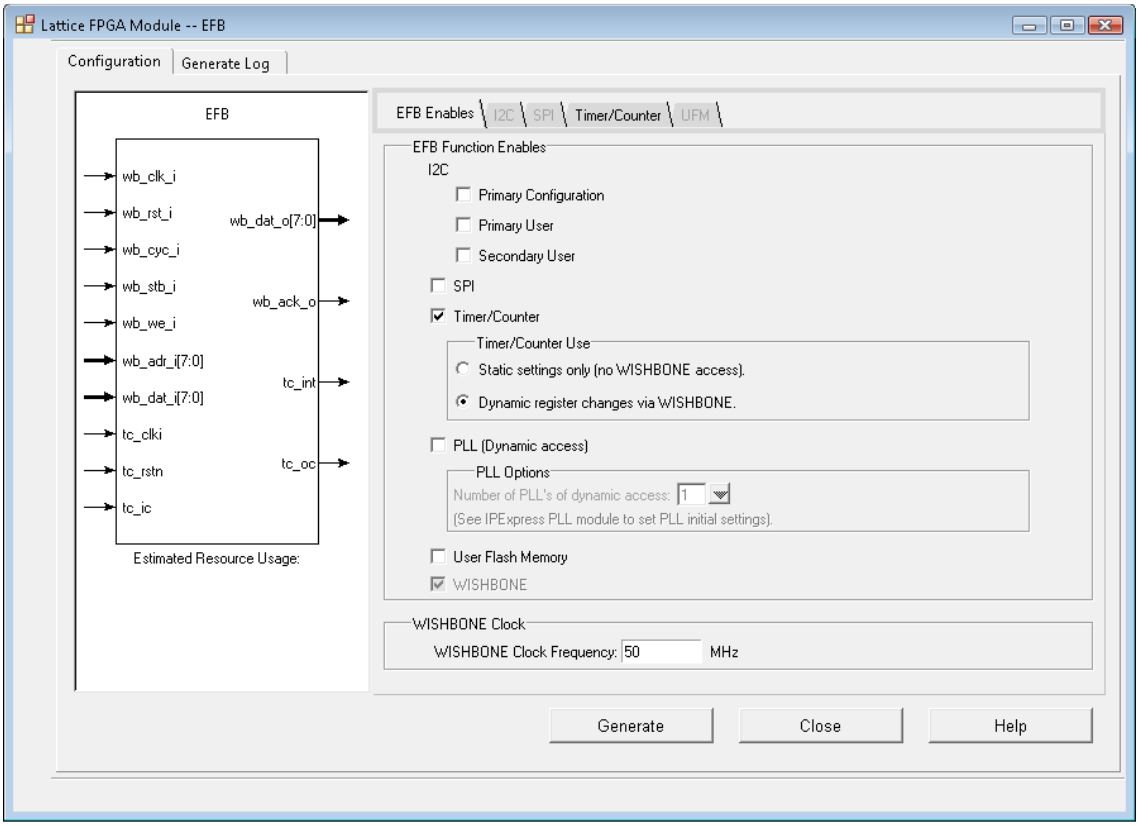


Figura 46: Creación de un bloque de control para el módulo EFB

En la pestaña de opciones del *Timer/Counter* se configurará este bloque de la siguiente manera:

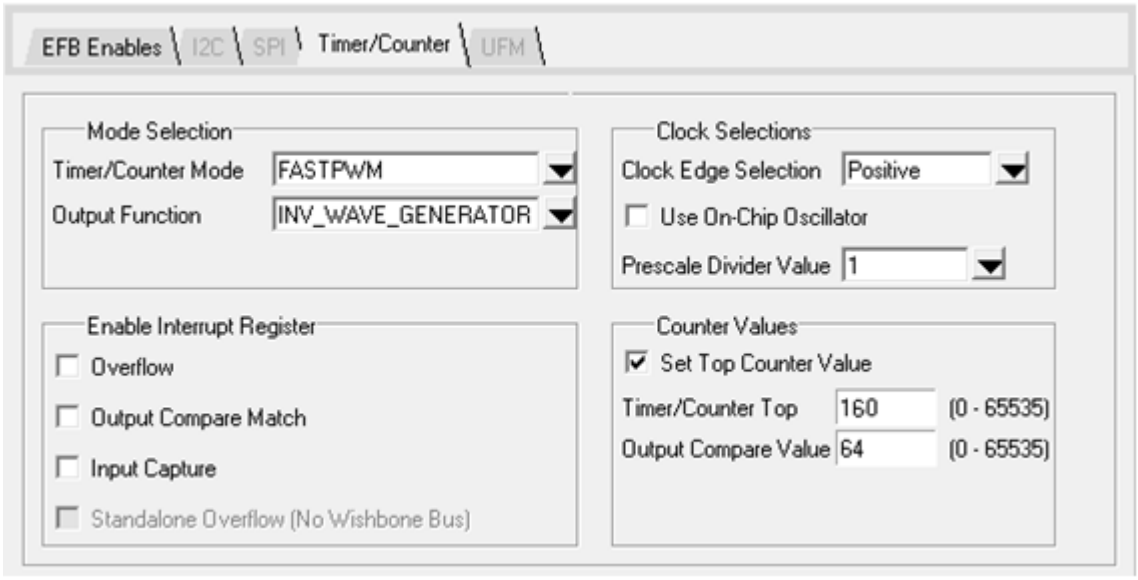


Figura 47: Configuración del Timer/Counter del módulo EFB

Funcionando en modo *Fast PWM* con esta configuración, la salida del Timer/Counter se pondrá a nivel bajo cuando el contador alcance el valor cargado en el registro TCTOP (valor máximo) y a nivel alto cuando el contador coincida con el valor cargado en el registro TCOCR (valor de comparación).

El valor máximo se define como 160 y no se modificará (más adelante se justificará el por qué de este valor y no otro). Para variar el ancho de pulso de la señal de salida modificaremos dinámicamente (vía *Wishbone bus)* el valor de comparación, almacenado en el registro TCOCR.

Como ya se explicó en el apartado 3.2 de este informe, el proceso de escritura de un byte en este bus se corresponde con el siguiente patrón de formas de onda:

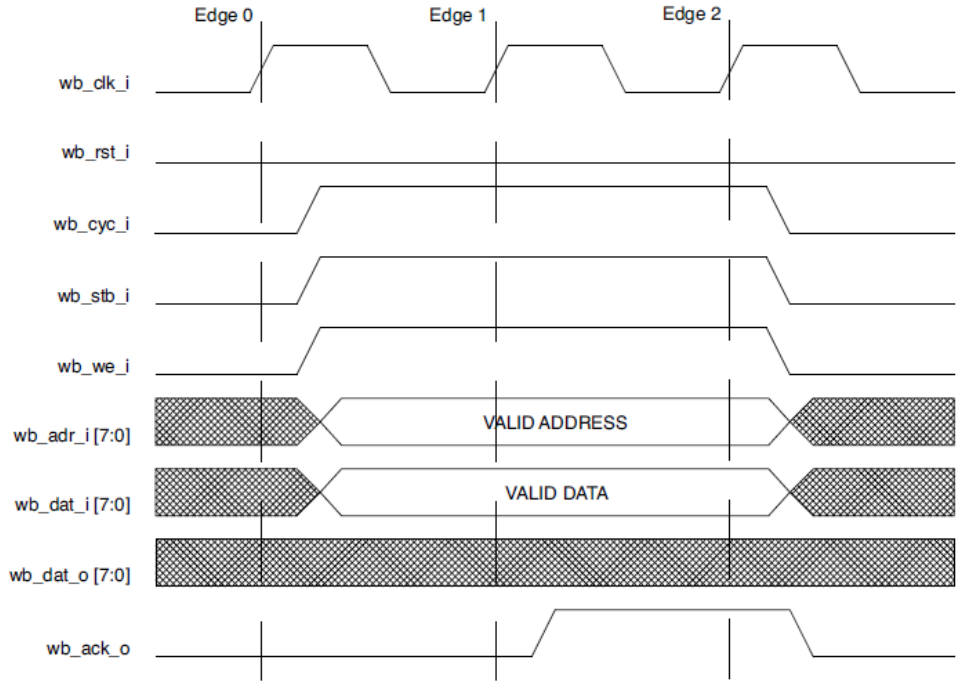


Figura 48: Operación de escritura de un byte en el Wishbone bus

Las señales *wb_cyc_i, wb_stb_i y wb_we_i* tendrán que permanecer en valor alto hasta que finalice la escritura, lo cual vendrá marcado por la puesta a nivel alto de la señal *wb_ack_o.*

La dirección del registro TCOCR en valor hexadecimal es 0x62. Su correspondiente valor binario, que habrá que introducir en la señal *wb_adr_i[7:0],* es 01100010.

La señal *wb_dat_i[7:0]* recibirá el dato en binario que se desea escribir en el registro.

Una vez comprendido el proceso que permite escribir un dato en un registro mediante el *Wishbone bus*, se diseñará un circuito que lleve a cabo esta tarea.

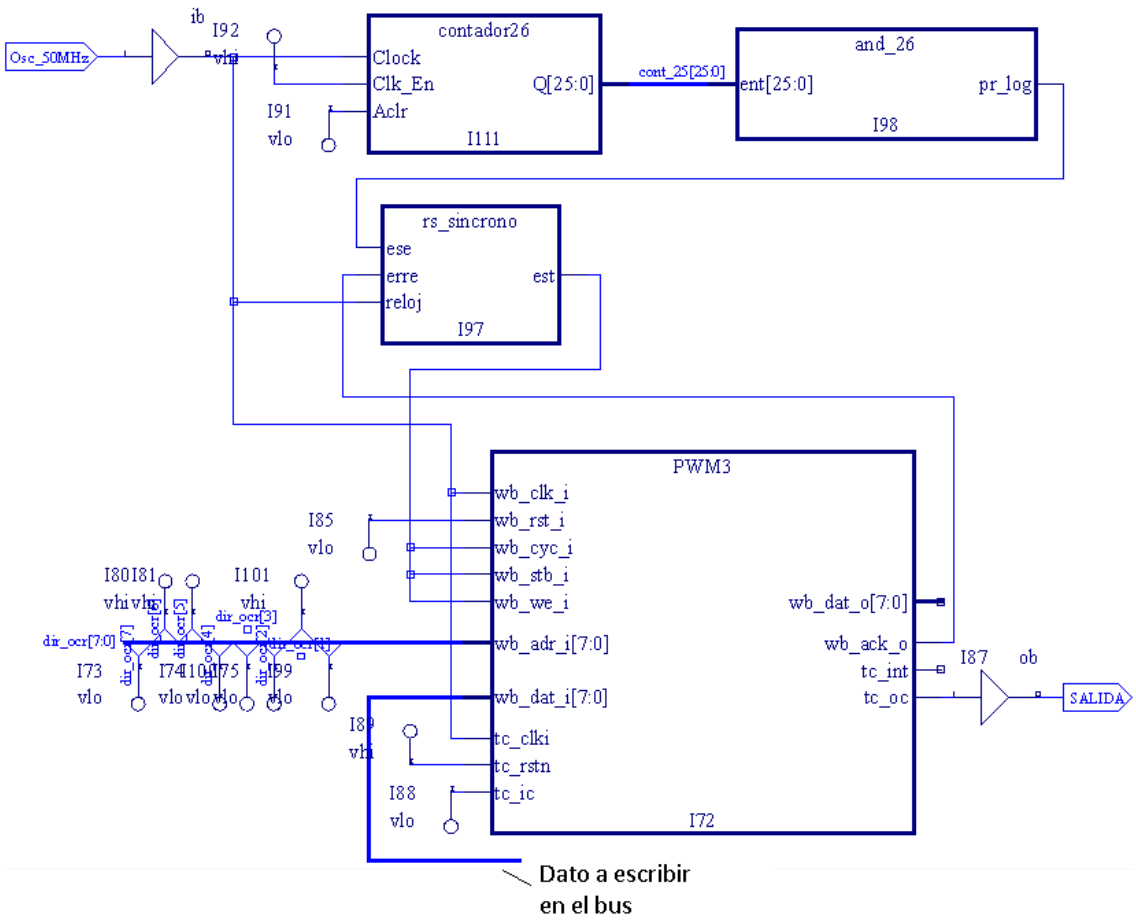


Figura 49: Circuito de escritura en el registro TCOCR, vía Wishbone bus

Cuando el contador de 26 bits alcanza su valor máximo, aproximadamente una vez cada segundo, la operación AND de todos sus dígitos llevada a cabo por el bloque “and_26” devuelve un uno lógico y se activa el bloque “rs_sincrono”.

Este bloque es el encargado de generar los ciclos de escritura. Mediante su salida “est” pone a nivel alto las señales *wb_cyc_i, wb_stb_i y wb_we_i,* que permanecen así hasta que se reciba de *wb_ack_o* un uno lógico indicando que la escritura del dato en el registro se llevo a cabo de manera correcta. El código VHDL de este bloque es el siguiente:

```
----------------------------------------------------
------ TRABAJO FIN DE GRADO, curso 2013/14 ---------
---------- Gonzalo A. Garcia Gallego ---------------
---Generador de ciclos de escritura en el WB bus ---
----------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity rs_sincrono is
    port(
        ese: in std_logic;
        erre: in std_logic;
        reloj: in std_logic;
        est: out std_logic
        );
end rs_sincrono;

architecture rs_sincrono_arch of rs_sincrono is
begin
    process(reloj)
        begin
            if(reloj'event and reloj='1') then
                if(ese='1' and erre='0') then est<='1';
                elsif(ese='0' and erre='1') then est<='0';
                elsif(ese='1' and erre='1') then est<='0';
                end if;
            end if;
    end process;
end rs_sincrono_arch;
```

La salida *tc_oc* del bloque PWM devuelve la señal con ancho de pulso variable que se buscaba. Mediante está señal se podrá controlar la intensidad luminosa de un diodo LED introduciendo en la señal *wb_dat_i[7:0]* un número binario equivalente a un valor entre 0 y 160.

Una manera sencilla de comprobar que este montaje funciona es introducir como dato a escribir en el registro TCOCR el valor almacenado en los registros de los displays del circuito del apartado anterior. De esta manera, la variación en la luz incidente en el sensor tendrá como consecuencia el aumento o disminución de la intensidad luminosa apreciable en el LED, pudiendo así visualizar que el circuito de escritura en el *Wishbone bus* funciona.

El número más alto que podrá escribirse en el registro de esta manera (cuando los displays muestren el valor “9 9”) será 1001 1001, que en decimal equivale a 153. Por este motivo se definió 160 como valor máximo del contador.

4.2.7 Control mediante micro interruptores:

El último paso del desarrollo del programa consistirá en añadir la posibilidad de introducir en el sistema una señal de referencia de la intensidad luminosa que se desea que haya en el ambiente. Este valor se introducirá mediante tres interruptores.

En función del valor de las dos entradas del sistema: el sensor convertidor luz-frecuencia y el valor de referencia en los micro interruptores, se generará una señal de salida PWM para controlar la intensidad luminosa de una lámpara LED. El objetivo es conseguir que la suma de la intensidad luminosa en el ambiente (almacenada en los registros) más la intensidad luminosa generada en la lámpara coincidan con la señal de referencia introducida por el usuario. O lo que es lo mismo:

$$I.L.\,de\ la\ lámpara = I.L.referencia - I.L.\,guardada\ en\ registros$$

Lo primero que se hará es un bloque que reciba como entradas el valor de los tres micro interruptores y devuelva un valor binario equivalente en ocho bits. Esta conversión se llevará a cabo de acuerdo a la siguiente tabla:

Valor Micro interruptores	Señal de referencia	Equivalente binario
000	*0*	*00000000*
001	*100*	*01100100*
010	*150*	*10010110*
011	*200*	*11001000*
100	*250*	*11111010*

El código VHDL desarrollado para implementar este bloque es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity control_sw is
    port(
        ENTRADA: in STD_LOGIC_VECTOR(2 downto 0);
        SALIDA: out STD_LOGIC_VECTOR(7 downto 0)
        );
end control_sw;

architecture control_sw_arch of control_sw is
begin
    process(ENTRADA)
        begin
            if(ENTRADA="000") then SALIDA<="00000000";
            elsif(ENTRADA="001") then SALIDA<="01100100";
            elsif(ENTRADA="010") then SALIDA<="10010110";
            elsif(ENTRADA="011") then SALIDA<="11001000";
            elsif(ENTRADA="100") then SALIDA<="11111010";
            else SALIDA<="00000000";
            end if;
    end process;
end control_sw_arch;
```

Se generará con la ayuda de la herramienta *IPexpress* un bloque restador de ocho bits. Se debe tener en cuenta que este bloque trabaja con números en complemento a 2, por lo que se realizará la transformación correspondiente de las señales de entrada y de salida. (Bastará con invertir cada uno de los ocho bits y al resultado sumarle 00000001).

El siguiente circuito devuelve la diferencia entre la señal de referencia introducida por el usuario y el valor almacenado en los registros.

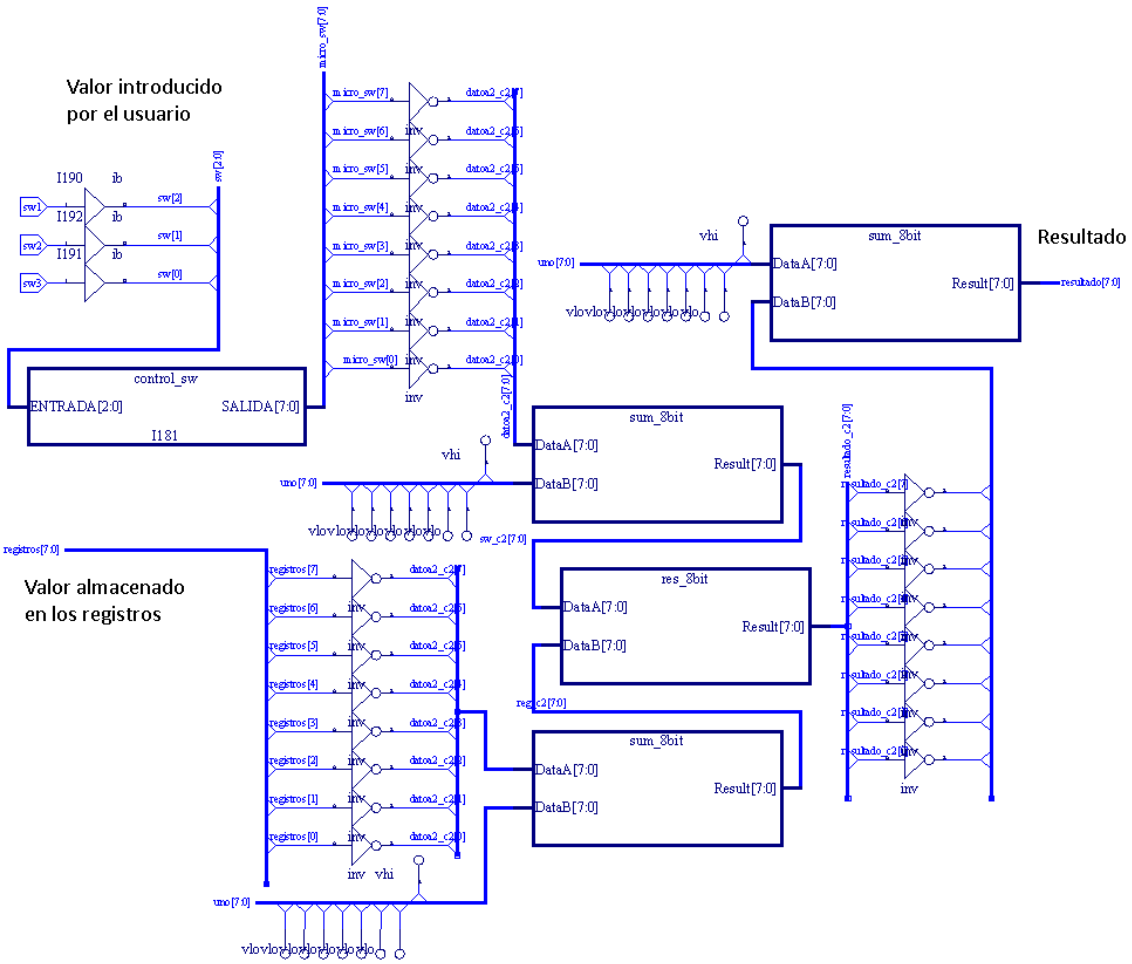


Figura 50: Restador de dos señales de 8 bits

Para estar seguros de que el valor devuelto por la resta es positivo, se usará un comparador de ocho bits y un bloque selector de salida. En caso de que el valor de la referencia sea mayor al de los registros, el selector de salida devolverá el resultado de la resta. En caso contrario, la salida será 0, ya que la intensidad luminosa en el ambiente será mayor a la de referencia y la lámpara deberá permanecer apagada.

El esquema modificado puede verse a continuación:

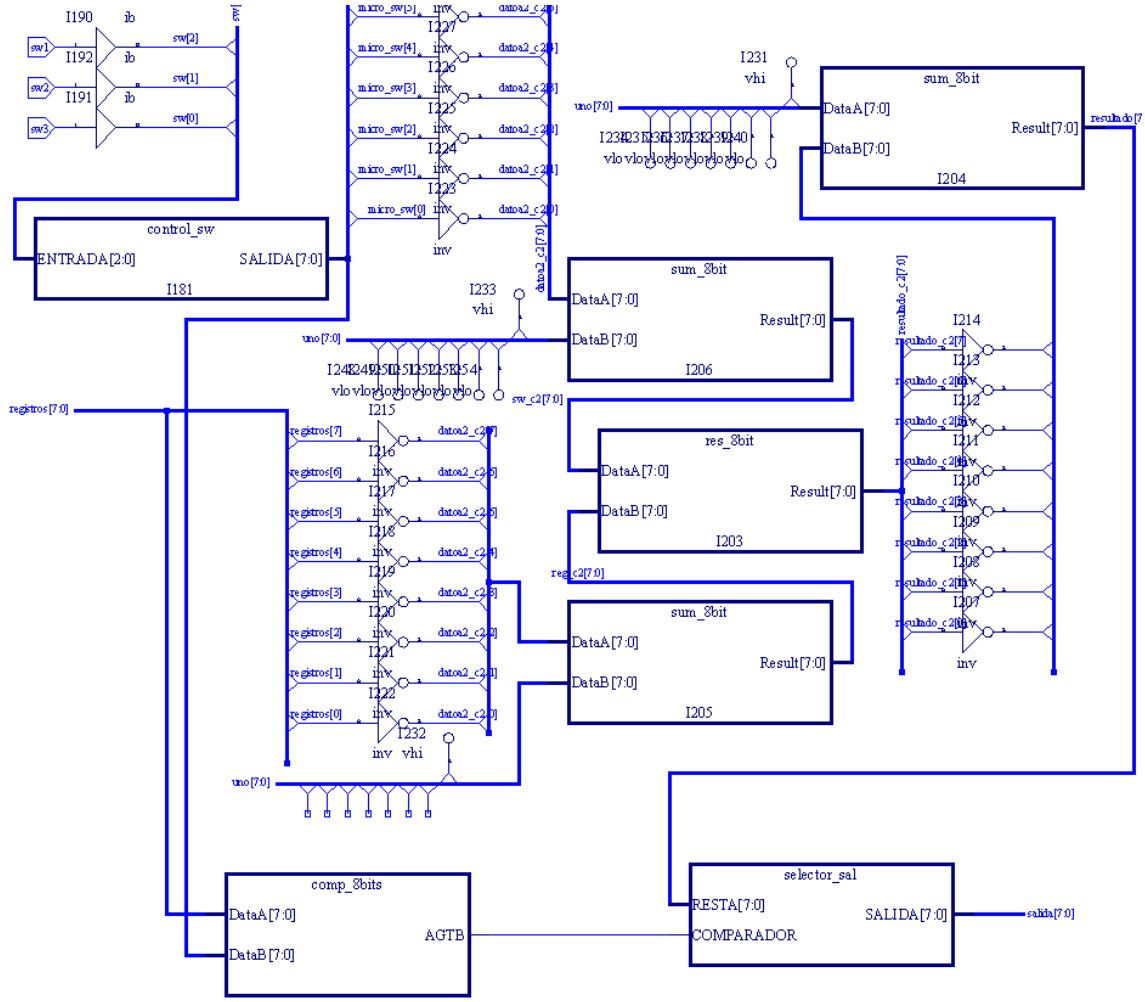


Figura 51: Restador de dos señalas de ocho bits con comparador y selector de salida

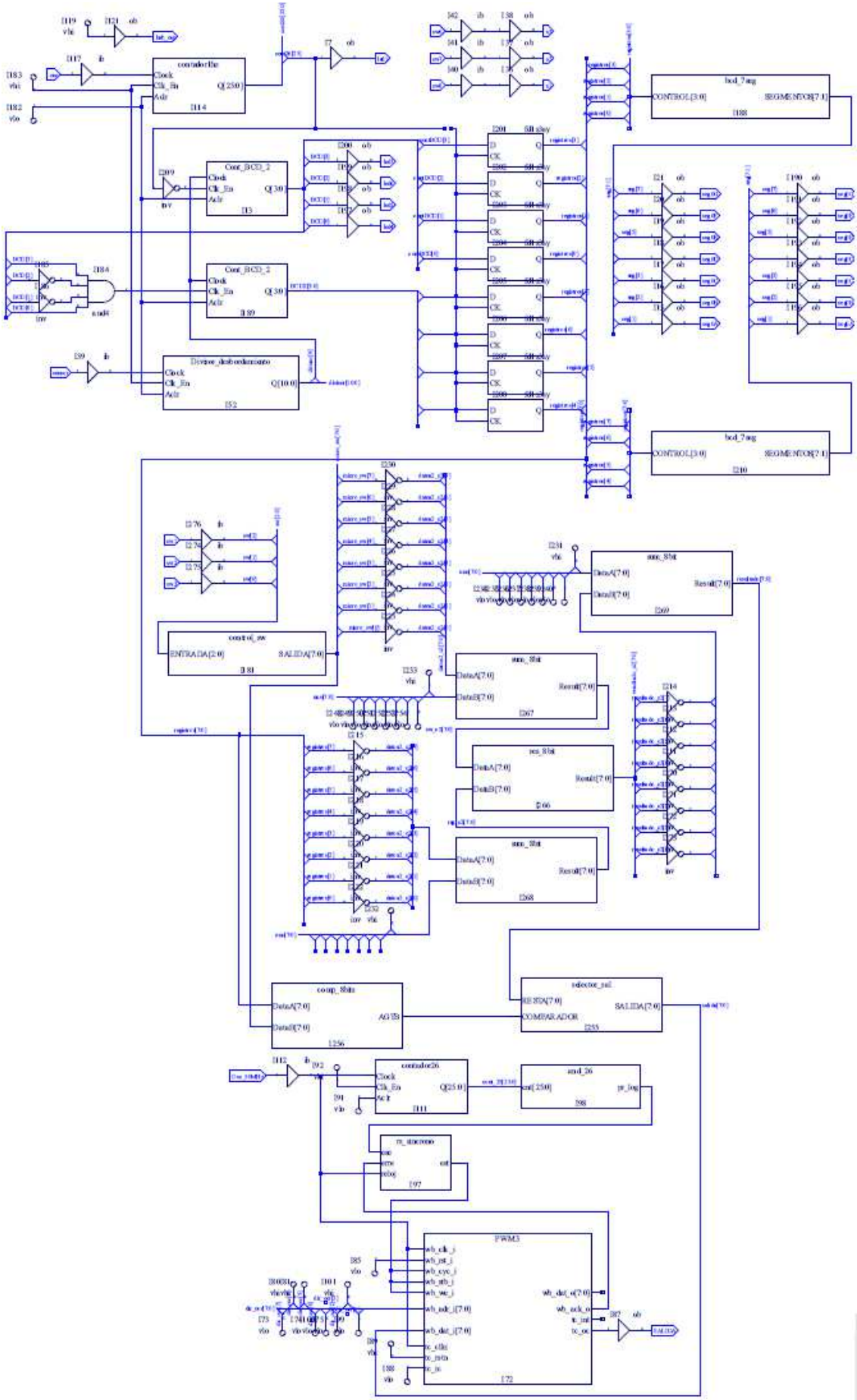
La salida del selector genera el dato que deseamos escribir en el registro TCOCR, por lo que se conectará directamente a la entrada *wb_dat_i[7:0]* del bloque controlador del modulo EFB.

4.2.8 Visión general del sistema completo:

La versión final del programa de control de este proyecto estará formada por la unión de los tres circuitos desarrollados en los apartados anteriores:

- Lectura del sensor de luminosidad y visualización a través del displays.
- Implementación del *Fast PWM* mediante escritura en el *WIshbone Bus.*
- Lectura de la señal de referencia a través de los micro interruptores y cálculo de la intensidad luminosa necesaria en la lámpara.

El esquema final de este programa puede verse a continuación:


contador10e
Conf_BCD_2
Divisor_disbordamiento
bcd_7seg
CONTROL[3:0]
SEGMENTOS[7:1]
control_sw
ENTRADA[2:0]
SALIDA[7:0]
suma_8bit
DataA[7:0]
DataB[7:0]
Result[7:0]
resta_8bit
comp_8bits
AGTB
selector_out
RESTA[7:0]
COMBARADOR
SALIDA[7:0]
contador26
mod_26
PWM3
wb_clk_i
wb_rst_i
wb_cyc_i
wb_stb_i
wb_we_i
wb_adr_i[7:0]
wb_dat_i[7:0]
wb_dat_o[7:0]

Recapitulando, este programa mostrará por los displays de 7 segmentos el valor de la intensidad luminosa incidente en el sensor convertidor luz-frecuencia. El usuario podrá introducir a través de tres micro interruptores una señal de referencia para la intensidad luminosa, que el sistema tratará de alcanzar modificando la intensidad luminosa de una lámpara LED mediante control PWM.

En la siguiente tabla se recuerda la función de cada micro interruptor y el significado de cada uno de los LEDs de la placa:

	Micro interruptor	LED
1	*Señal referencia I.L. [2]*	*Señal Reloj de 1 Hz*
2	*Señal referencia I.L. [1]*	*No usado*
3	*Señal referencia I.L. [0]*	*No usado*
4	*No usado*	*Contador BCD [0]*
5	*No usado*	*Contador BCD [1]*
6	*Señal control sensor S3*	*Contador BCD [2]*
7	*Señal control sensor S2*	*Contador BCD [3]*
8	*Señal control sensor S1*	*No usado*

A continuación se muestran los informes generados por el programa a la hora de compilar nuestro diseño, donde se pueden ver detalles de la implementación y del aprovechamiento de la capacidad del dispositivo:

Design Summary

```
Number of registers:     79 out of  1604 (5%)
   PFU registers:           79 out of  1280 (6%)
   PIO registers:            0 out of   324 (0%)
Number of SLICEs:        94 out of   640 (15%)
   SLICEs as Logic/ROM:     94 out of   640 (15%)
   SLICEs as RAM:            0 out of   480 (0%)
   SLICEs as Carry:         67 out of   640 (10%)
Number of LUT4s:        187 out of  1280 (15%)
   Number of logic LUTs:       53
   Number of distributed RAM:   0 (0 LUT4s)
   Number of ripple logic:     67 (134 LUT4s)
   Number of shift registers:   0
Number of PIO sites used: 32 + 4(JTAG) out of 108 (33%)
Number of block RAMs:  0 out of 7 (0%)
Number of GSRs:  1 out of 1 (100%)
EFB used :       Yes
JTAG used :      No
Readback used :  No
Oscillator used :  No
Startup used :   No
POR :            On
Bandgap :        On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 4 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 1 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
```

```
   Number of CLKDIVC:  0 out of 4 (0%)
   Number of ECLKSYNCA:  0 out of 4 (0%)
   Number of ECLKBRIDGECS:  0 out of 2 (0%)
   Notes:-
      1. Total number of LUT4s = (Number of logic LUT4s) + 2*(Number of
     distributed RAMs) + 2*(Number of ripple logic)
      2. Number of logic LUT4s does not include count of distributed RAM
and
     ripple logic.
   Number of clocks:  4
     Net divisor_9: 4 loads, 4 rising, 0 falling (Driver: I52/FF_1 )
     Net cont26_25: 6 loads, 6 rising, 0 falling (Driver: I114/FF_0 )
     Net N_6: 5 loads, 5 rising, 0 falling (Driver: PIO sensor )
     Net N_14: 29 loads, 29 rising, 0 falling (Driver: PIO osc )

   Number of Clock Enables:  3
     Net cont26_25: 2 loads, 2 LSLICEs
     Net N_7: 2 loads, 2 LSLICEs
     Net I97/n398: 1 loads, 1 LSLICEs
   Number of LSRs:  2
     Net scuba_vhi: 1 loads, 0 LSLICEs
     Net cont26_25: 4 loads, 4 LSLICEs
   Number of nets driven by tri-state buffers:  0
   Top 10 highest fanout non-clock nets:
     Net registros_0: 9 loads
     Net registros_1: 9 loads
     Net registros_2: 9 loads
     Net registros_3: 9 loads
     Net registros_4: 9 loads
     Net registros_5: 9 loads
     Net registros_6: 9 loads
     Net registros_7: 9 loads
     Net co3: 8 loads
     Net scuba_vhi: 6 loads

   Number of warnings:  1
   Number of errors:    0
```

PAD Specification File

```
PART TYPE:        LCMXO2-1200ZE
Performance Grade:      1
PACKAGE:          TQFP144
Package Status:                     Final          Version 1.39

Sat Jun 21 12:02:22 2014

Pinout by Port Name:
+-----------+----------+--------------+--------------------------------------+
| Port Name | Pin/Bank | Buffer Type  | Properties                           |
+-----------+----------+--------------+--------------------------------------+
| Salida    | 142/0    | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| hab_osc   | 32/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| led1      | 97/1     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| led4      | 100/1    | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| led5      | 104/1    | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| led6      | 105/1    | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| led7      | 106/1    | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| osc       | 27/3     | LVCMOS33_IN  | PULL:DOWN CLAMP:ON HYSTERESIS:SMALL  |
| s1        | 4/3      | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| s2        | 3/3      | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| s3        | 2/3      | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0A     | 21/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0B     | 22/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0C     | 11/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0D     | 10/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0E     | 13/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
| seg0F     | 12/3     | LVCMOS33_OUT | DRIVE:8mA PULL:DOWN SLEW:SLOW        |
```

seg0G	14/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1A	23/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1B	25/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1C	33/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1D	34/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1E	35/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1F	26/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
seg1G	24/3	LVCMOS33_OUT	DRIVE:8mA PULL:DOWN SLEW:SLOW
sensor	1/3	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw1	86/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw2	85/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw3	84/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw6	81/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw7	78/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL
sw8	77/1	LVCMOS33_IN	PULL:DOWN CLAMP:ON HYSTERESIS:SMALL

4.3 Adaptación de lámpara LED y calibración

Para simplificar el montaje, se modificará una bombilla LED de 5W para que funcione con un voltaje de alimentación de 12V. De esta manera podremos implementar el control PWM usando únicamente un transistor y una resistencia. El esquema eléctrico será el siguiente:

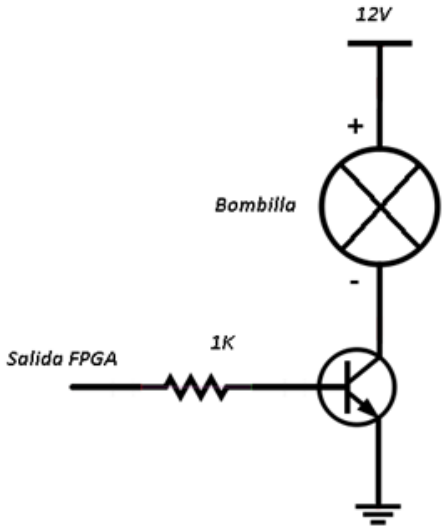


Figura 52: Esquema de conexiones para el control PWM de la bombilla

Se montará este circuito en el espacio libre destinado a prototipos de la *Breakout Board*. Se añadirán además dos conectores donde enchufar de manera sencilla la alimentación a 12V y los polos positivo y negativo de la bombilla.

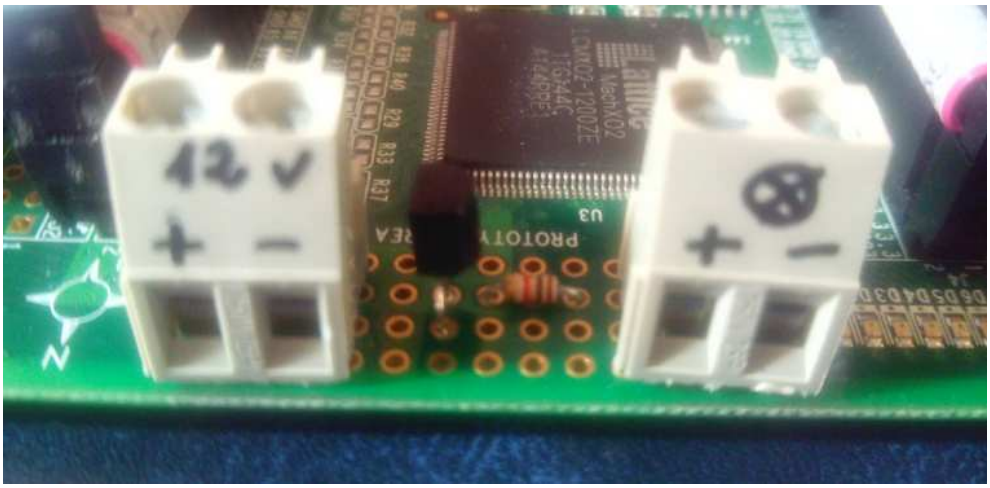


Figura 53: Conectores, transistor y resistencia en la zona de prototipos de la Breakout Board

Una vez hecho el montaje y comprobado su correcto funcionamiento, se realizará una tabla de equivalencias entre el valor de la intensidad luminosa mostrado en los displays y la intensidad luminosa real medida en luxes mediante un luxómetro. A continuación puede verse una tabla con los resultados obtenidos y su correspondiente representación gráfica:

Valor Display	Luxes
17	91
19	107
21	122
22	138
24	153
25	167
27	182
29	193
31	213
34	230
35	246
37	260
38	277
40	294
43	308
44	320
46	340

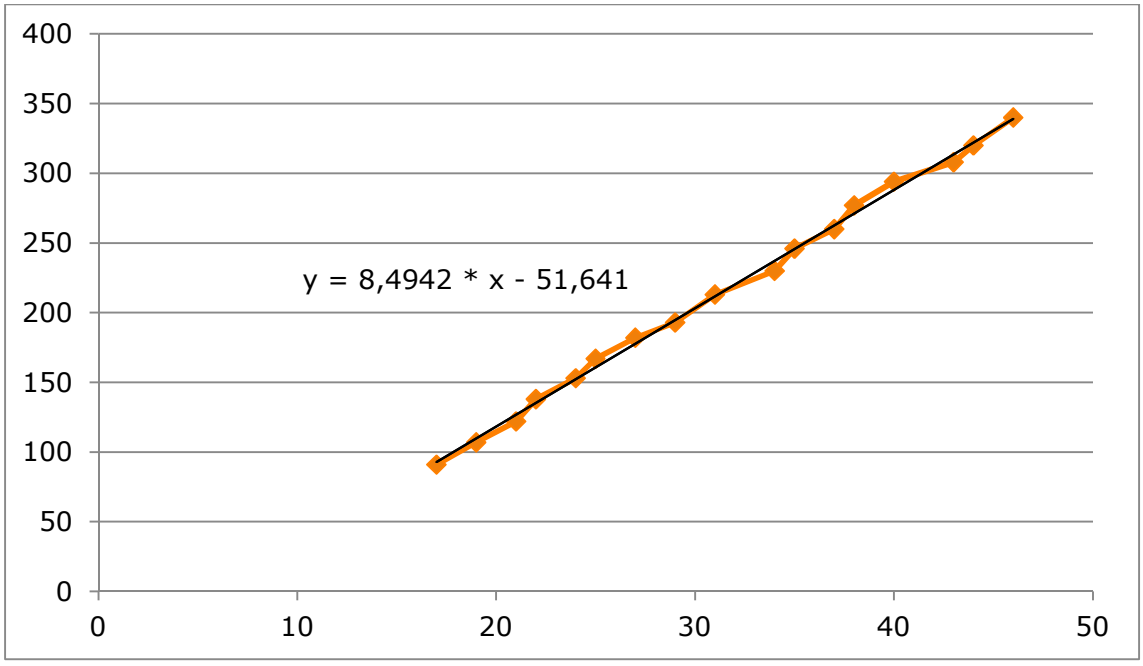


Figura 54: Eje X: Valor mostrado en el Display; Eje Y: Intensidad luminosa en luxes

Como puede verse en la gráfica, la relación lineal entre estos valores, que permitirá calcular la intensidad luminosa en luxes para cualquier valor del display dentro del rango de iluminación normal de una habitación es:

$$\boldsymbol{Intensidad\ luminosa\ en\ luxes = 8,49 * Valor\ display - 51,54}$$

Figura 55: Bombilla LED similar a la usada en el proyecto

Capítulo 5. Conclusiones

En este capítulo se presentan las conclusiones obtenidas después de la ejecución del proyecto:

- ✓ Se ha seleccionado el hardware de partida que se ha considerado más oportuno, después de analizar otras posibles alternativas.
- ✓ Se ha llevado a cabo un estudio de las características y posibilidades de los dispositivos tipo FPGA.
- ✓ Se ha dotado al hardware de partida de un interfaz a base de interruptores, pulsadores y displays de 7 segmentos.
- ✓ Se ha integrado en el sistema un sensor luminoso.
- ✓ Se han utilizado herramientas de diseño, simulación y síntesis modernas, obteniendo un diseño estructurado y jerárquico.
- ✓ Se han empleado técnicas de diseño mixtas, utilizando lenguajes de descripción de hardware y esquemas. La descripción global se ha hecho en base a esquemas y a diferentes bloques funcionales en VHDL.
- ✓ Se han utilizado herramientas automáticas para la generación de descripciones VHDL.
- ✓ Se ha realizado y puesto a punto un sistema de monitorización de la intensidad luminosa, así como el control de una lámpara LED en función del valor de dicha intensidad.
- ✓ Se ha calculado la relación lineal entre el valor mostrado en los displays y la intensidad luminosa real medida en luxes a través un luxómetro.

En cuanto a las líneas futuras de este proyecto, se podría estudiar el desarrollo de otras posibles aplicaciones de este dispositivo tales como:

- **Analizador de color:** En base a la posibilidad del sensor convertidor luz-frecuencia de detectar la energía lumínica incidente en forma de luz roja, verde o azul, podría diseñarse un sistema que indicara el porcentaje de cada uno de estos colores presente en una señal luminosa.
- **Exposímetro:** Útil para operaciones que requieren un tiempo de exposición a la luz muy exacto, como por ejemplo el rebelado de placas de circuito impreso. Este dispositivo podría calcular de

manera exacta el tiempo necesario de exposición a la luz dependiendo de la intensidad luminosa que se esté aplicando. Una vez cumplido el tiempo, se encendería/apagaría un led de la placa.

Capítulo 6. Bibliografía

- "MachX02 Family Handbook" (Manual de usuario de la familia de dispositivos *MachX02*)
- "Lattice Diamond User Guide" (Guía de usuario del programa *Lattice Diamond*)
- Datasheet del convertidor luz-frecuencia TCS3210
- K. C. Chang, "Digital Design and Modeling with VHDL and synthesis".
- S. A. Pérez, E. Soto, S. Fernández, "Diseño de sistemas digitales VHDL".
- http://pablin.com.ar/electron/cursos/intropld/index.htm, junio de 2014.
- http://icprgm-asic.blogspot.com.es/2007/12/qu-es-un-asics.html, junio de 2014.
- http://microcontroladores-e.galeon.com/, junio de 2014.
- http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/658/A8.pdf?sequence=8, junio de 2014.
- http://pablin.com.ar/electron/cursos/intropld/index.htm, junio de 2014.