# MASTER EN INGENIERÍA INDUSTRIAL

**ESCUELA DE INGENIERÍAS INDUSTRIALES**
**UNIVERSIDAD DE VALLADOLID**

## TRABAJO FIN DE MÁSTER

# Implantación de circuitos detectores de secuencia en dispositivos FPGA

Autor: D. Rubén de Juan Hernández
Tutor: D. Francisco José de Andrés Rodríguez Trelles

Valladolid, Mayo, 2019

Implantación de circuitos detectores de secuencia en dispositivos FPGA

# Resumen

El estudio de investigación realizado y descrito en este Trabajo Fin de Master consiste en realizar el desarrollo y la implantación de varios circuitos lógicos que permitan la detección de secuencias suministradas en diferentes formatos.

Estos circuitos se encargan de obtener una secuencia de información y tratarla para conseguir el objetivo especificado para ella.

Para ello se crearan varios circuitos que extraigan información en forma secuencial, suministradas por diferentes fuentes. Como caso concreto se tratará la suministrada por un sensor de temperatura y humedad.

Como herramienta de diseño se ha empleado *Lattice Diamond* de *Lattice Semiconductors*, realizando la implantación en la FPGA Machx02 1200-ZE.

# Palabras clave

FPGA, VHDL, Secuencia, PWM, Temperatura, Humedad, Contraseña

# Abstract

The research developed in this project consists in the implantation of several logic circuits that can detect information given in different sequential formats.

The purpose of these circuits is to obtain information and transform it to be able to work with it and get the objective set for that circuit.

Several circuits that can get, analyze and work with this transformed information will be created. An example of his will be reading and working with the data given by a humidity and temperature sensor.

The main design tool is *Lattice Diamond* of *Lattice Semiconductors*, implementing it in the FPGA Machx02 1200-ZE.

# Keywords

FPGA, VHDL, Sequence, PWM, Temperature, Humidity, Password

# Agradecimientos

Quiero aprovechar este espacio para agradecer a todas aquellas personas que me han ayudado y apoyado a lo largo de mi vida.

Me gustaría empezar agradeciéndoselo especialmente a mis padres Alfonso y Consuelo, quienes han sido mi principal apoyo, todo el esfuerzo que han hecho para que yo haya llegado hasta aquí, así como el ánimo que me han dado siempre.

También quiero destacar mi hermana Laura por su buen carácter conmigo. He de agradecer también a todos mis compañeros durante el master porque sin su ayuda habría sido mucho más difícil mi paso por la Universidad.

Finalmente, me gustaría dar las gracias al tutor que he tenido durante este Trabajo Fin de Master Francisco, por haberme dado la oportunidad de llevar a cabo esta tarea y por toda la ayuda y consejos que me han dado durante la realización de todo este trabajo.

# ÍNDICE

# ÍNDICE DE FIGURAS

# 1. INTRODUCCIÓN

## 1.1 Motivación

En este Trabajo Fin de Master se ha llevado a cabo una investigación de cómo tratar información aportada en formato secuencial de diferentes formas y como aprovechar esta información para diferentes aplicaciones.

Para ello se van a realizar varios circuitos lógicos que se encarguen de tras recibir información bien sea mediante interruptores, pulsadores o mediante otras formas como puede ser una línea conectada a un sensor sea capaz de almacenar esta información, convertirla en un conjunto de bits que se puedan entender y realizar operaciones sobre ellos, como puede ser compararlos entre sí o con otros conjuntos de bits predefinidos, o cambiarlos a base decimal para extraer información fácil de interpretar a una persona.

Dentro de estos casos el principal será el que trata con un sensor dado que requiere más tratamiento y control de la información tanto enviada como recibida del mismo.

## 1.2 Objetivos

El principal objetivo de este proyecto es obtener la capacidad de entender diferentes formas de tratar de información de forma secuencial.

Un objetivo secundario debido a las herramientas utilizadas es ampliar el conocimiento de herramientas de diseño, simulación y síntesis lógico que son las encargadas de configurar el hardware empleado. Para la puesta a punto se emplea un analizador lógico.

Para la consecución de estos objetivos se han divido las tareas en apartados más pequeños que permitan un seguimiento más sencillo del avance. Estas se muestran a continuación.

- Análisis de los elementos empleados para realizar las operaciones lógicas y posibles alternativas.
- Implantación de la lógica empleada en los aparatos y comprobar su funcionamiento.
- Análisis del sensor utilizado para extraer información y sus características.
- Aplicación de máquina de estados finitos para simplificar la programación mediante un generador de código a partir de las mismas.
- Estudio del lenguaje VHDL para poder realizar de forma correcta y sencilla los diferentes bloques lógicos que forman parte de las diferentes implantaciones.

- Realizar simulaciones de los bloques lógicos para determinar si su funcionamiento es correcto o si existen posibles problemas que no se habían planteado durante la programación.
- Aplicación de analizadores lógicos y herramientas como *Saleae Logic* que permiten extraer en tiempo real el estado de variables lógicas.
- Capacidad de subdividir proyectos de programación de forma que sea más sencillo entender la lógica aplicada y encontrar posibles errores.

Como generalidad de este proyecto cabe destacar la facilidad de adaptar el último diseño a otras fuentes de información que utilizan un protocolo similar como podría ser un emisor de infrarrojos.

## 1.3 Estructura de la Memoria

A continuación se describe las diferentes partes de las que consta este documento.

En el capítulo 2 se describe todo el hardware empleado y se discuten diferentes alternativas posibles para la realización de este proyecto, señalando tanto ventajas e inconvenientes principales.

En el capítulo 3 se explica cómo se ha realizado la implantación de los diferentes diseños realizados señalando tanto características generales como particulares de cada diseño. Incluye toda la información necesaria para reproducir los diseños sobre los que trata este documento.

En el capítulo 4 se hace un estudio de los diferentes recursos empleados en el último diseño señalando los parámetros relevantes ante la posibilidad de utilizar componentes con otras características.

En el capítulo 5 se realiza un estudio del coste económico que tendría el último diseño explicado. Realizando dicho análisis tanto para la fase de prototipo como la de producción en masa.

En el capítulo 6 se detallan las conclusiones obtenidas en este trabajo y posibles adicciones al mismo.

En el capítulo 7 se incluye la bibliografía utilizada en la realización del trabajo fin de master.

En el primer anexo se incluye documentación adicional sobre la FPGA utilizada, el modelo MachXO2 1200-ZE.

En el segundo anexo se añade información adicional sobre el sensor utilizado el DHT22.

En el tercer anexo se añade información sobre cómo están configuradas las PCB y a que elemento está unido cada pin de la FPGA.

# 2. HARDWARE DE CONTROL

En este apartado se describen varas posibles alternativas para la realización de los proyectos realizados y una descripción de las características generales del hardware sobre el que se ha trabajado.

## 2.1 Posibles alternativas de implantación

Dentro de las diferentes alternativas posibles para la realización de los proyectos desarrollados, se van a analizar principalmente tres dispositivos lógicos capaces de realizar las operaciones necesarias, a saber, microcontroladores, ASIC y FPGA.

Una de las principales diferencias entre estos dispositivos son si permiten la reprogramación o si están formados por componentes estándar siendo el conjunto de estos los que conforman el dispositivo.

### 2.1.1 Microcontroladores

Estos dispositivos están formados por diferentes bloques que se encargan de diferentes funcionalidades concretas formando en su conjunto un dispositivo de características adaptables.

Estos dispositivos se caracterizan por tener una memoria en la que se programan las órdenes a ejecutar. En estos dispositivos a diferencia de un ordenador la memoria es escasa por lo que deberá tenerse en cuenta su capacidad durante la programación y el diseño de las aplicaciones.

Además de la memoria existe una unidad de procesamiento que determina en gran medida las características del microcontrolador en cuanto a velocidad de ejecución, control de las interrupciones y tipo de buses de comunicación a utilizar.

Además de estos componentes también suelen contar con periféricos que permiten comunicarse con el exterior, además normalmente estos dispositivos cuentan con un conversor analógico-digital para poder tratar con ambos tipos de señales.

Una ventaja de este aspecto es la opción de tener acceso a variedad de fabricantes obteniendo un precio más competitivo. Además existen herramientas accesibles que permiten su fácil configuración y manejo.

Las principales desventajas de estos dispositivos suelen ser la limitada resolución de los conversores analógico-digitales y la escasa memoria con la cuentan.

### 2.2.2 Dispositivos ASIC

Los ASIC o circuitos integrados para aplicaciones específicas consisten en un circuito integrado desarrollado para un diseño en concreto.

La principal ventaja de estos dispositivos es su ajuste al diseño obteniendo una mejor optimización de los recursos y por tanto mejor precio. Otra ventaja es su menor consumo al no incluir componentes innecesarios.

La principal ventaja es el requerimiento de personal más especializado en estos componentes y que requieren más tiempo para el desarrollo de aplicaciones. Además es necesario de programas específicos para estos dispositivos siendo esto por lo general más complejos que el de otras opciones.

### 2.2.3 Dispositivos FPGA

Una FPGA o *Field Programable Gate Array* consiste en un circuito integrado formado por bloques lógicos cuyas conexiones pueden ser configuradas mediante un programa.

Estos dispositivos son un punto intermedio entre los dos anteriores permitiendo un dispositivo reprogramable pero sin estar completamente diseñado y adaptado a la aplicación que va a realizar.

En general estos dispositivos tienen peores características que un dispositivo ASIC, más lento, mayor consumo… Sin embargo al ser reprogramables cuentan con una mayor flexibilidad en el diseño permitiendo reducir el tiempo de desarrollo y costes para lotes pequeños.

Estos dispositivos cuentan con funciones como contadores y sumadores ya integrados que simplifican el diseño, pudiendo realizar aplicaciones optimizadas mediante estas funciones.

Además estos dispositivos incluyen herramientas auxiliares de complejidad moderada que permiten su configuración de forma sencilla.

Este dispositivo será el utilizado en los siguientes proyectos, principalmente debido a su flexibilidad y a que es posible configurar sus conexiones para comunicarse con dispositivos que usen protocolos de información no estándar.

Dentro de las FPGAs existen varios proveedores que ofrecen diferentes alternativas entre las que elegir. En este caso la decisión estaba predefinida por los componentes con los que ya contaba el departamento.

La FPGA utilizada es de la empresa *Lattice Semiconductor* aunque existen otros fabricantes a resaltar *Altera,* que cuenta con tres niveles incrementando sus características en función de las necesidades de los usuarios y *Xilinx,* siendo la mayor empresa del sector.

El modelo utilizado es de la familia MachXO2 que son dispositivos de fácil implementación y precio asequible.

Además se cuenta con el software aportado por esta empresa *Lattice Diamond* que mediante la configuración de diseños usando VHDL permite la creación, compilación e instalación del diseño, teniendo acceso además de otras herramientas auxiliares que permiten la simulación.

## 2.2 Descripción del hardware empleado

En este apartado se describe los componentes utilizados para la realización de los diseños. Para la consecución de estos proyectos es necesario una serie de componentes auxiliares a la FPGA.

Los principales componentes que forman parte del dispositivo se muestran en la imagen 2.2.1.

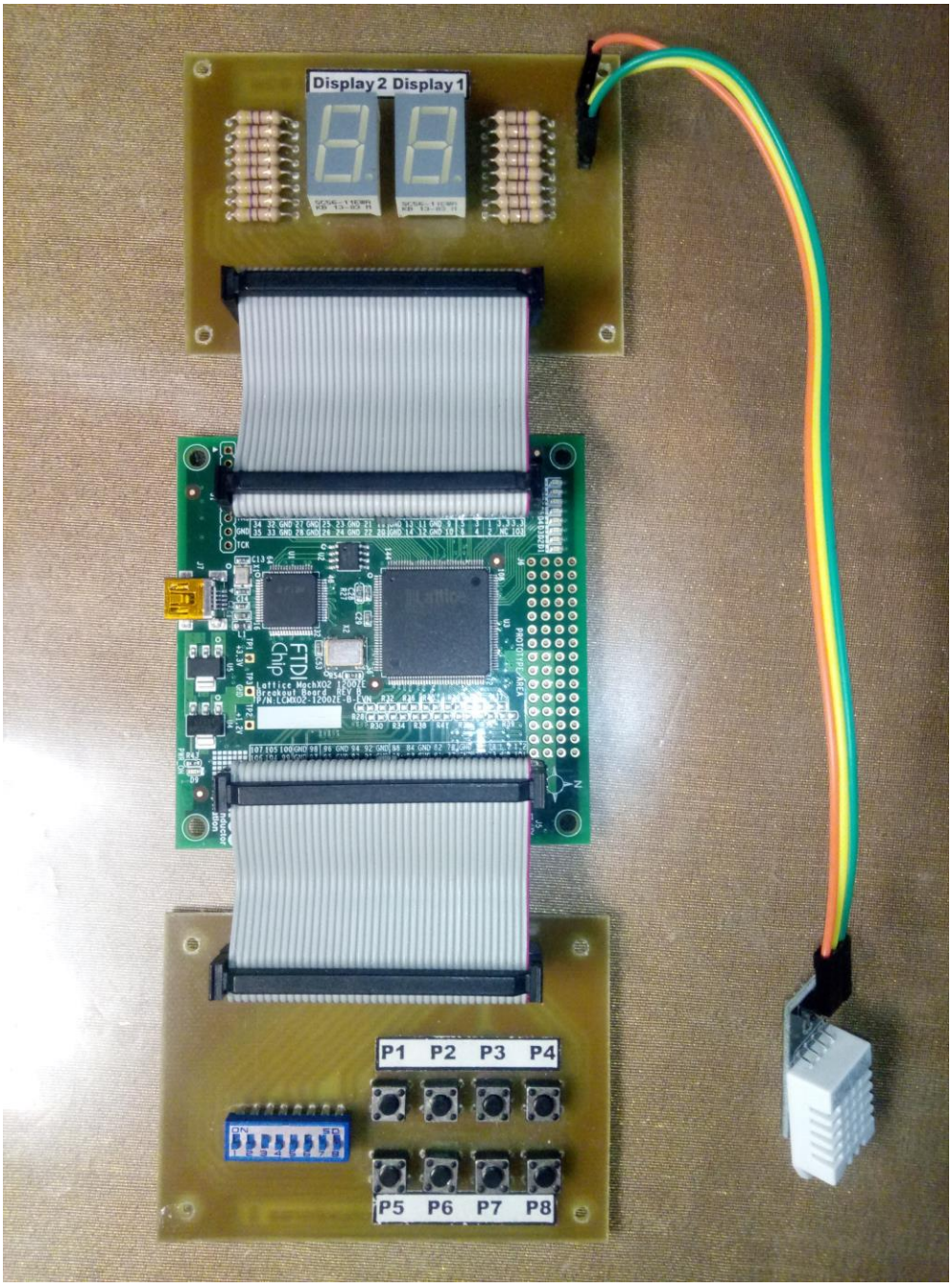



Figura 2.2.1. Componentes hardware

Los componentes están montados sobre tres placas con circuitos impresos que se encargan de transmitir las señales entre las diferentes partes.

La placa superior está formada por los displays que se encargan de mostrar la información pertinente al usuario y por pines para conectar módulos exteriores, como en este caso un sensor de temperatura y humedad.

La placa inferior cuenta con elementos de entrada de datos que como se observa en la imagen 2.2.1, consisten en ocho interruptores y ocho pulsadores.

La placa intermedia cuenta con un puerto USB mini-B que permite la conexión con un ordenador para configurarla y alimentarla, también cuenta con un reloj externo de alta precisión.

Todas estas partes están conectadas mediante buses de datos y dentro de las placas mediante el circuito impreso en las mismas.

Se incluye información más específica sobre la placa y como están configuradas sus conexiones en los anexos primero y tercero.

Para la comprobación de los diseños realizados se ha empleado un analizador lógico que se conecta a un USB de un ordenador, que mediante el software *Saleae Logic* permite extraer en tiempo real los valores de diferentes conexiones y los muestra en imagenes.

Los elementos utilizados para realizar estas comprobaciones con el analizador lógico se muestran en la figura 2.2.2.


Figura 2.2.2. Componentes analizador lógico

Los componentes utilizados son los siguientes:

- Cable adaptador USB a USB mini-B.
- Analizado lógico de ocho canales.
- Cables conectores entre el analizador y los cabezales.
- Lectores de punta conductora.

Un ejemplo de cómo se mostraría una señal analizada mediante estos componentes y el software indicado sería la siguiente.


Figura 2.2.3. Ejemplo señal

# 3. IMPLANTACIÓN

En este capítulo se presenta el desarrollo de los diferentes diseños realizados en este proyecto, describiendo todos sus bloques, acompañados cuando sea interesante por simulaciones.

Al realizar el diseño de los sistemas existen varios modelos según el estilo empleado pudiendo ser realizado exclusivamente en VHDL, exclusivamente mediante esquema o de forma mixta. En este caso se utilizara un procedimiento mixto.

Para la implantación el principal programa utilizado será Lattice Diamond Design Software, el principal motivo para el uso de este programa es la experiencia adquirida anteriormente, además de su fácil disponibilidad.

Otro factor para su empleo es el aporte de herramientas auxiliares que permiten la configuración del equipo empleado de forma más sencilla, ya que añade elementos de creación de bloques comúnmente usados mediante la aplicación IPexpress o herramientas de simulación del software creado como puede ser mediante Active-HDL.

Además también permite la implementación mediante bloques que se interrelacionan entre ellos y con el exterior mediante buffers, este método permite realizar un sistema modular de todo el sistema de forma que sea posible crear módulos más sencillos donde es más fácil encontrar posibles errores y son más fáciles de interpretar.

Además de Lattice Diamond Design Software también se han empleado otras herramientas, a destacar dos: Qfsm y Saleae Logic, los motivos de esto se explican a continuación.

El programa Qfsm se ha utilizado cuando ha sido necesario crear una máquina de estados finitos, normalmente como un bloque dentro de la implantación. El principal motivo de su uso es su sencillez para generar código VHDL que es con el que trabaja Lattice Diamond.

En cambio la utilización de Saleae Logic ha sido de cara a la comprobación de las implantaciones realizadas debido a que permite mediante unos conectores unidos a un puerto USB de un ordenador conocer el estado lógico de las diferentes conexiones accesibles, permitiendo por tanto una manera de ir controlando en formato físico los valores en tiempo real de diferentes bits.

## 3.1 Detector de secuencias binarias

La primera implantación consistirá en un dispositivo capaz de detectar cuando se ha introducido unas secuencias predeterminadas, activando o desactivando una salida en función de esto.

El principal objetivo de este proyecto es realizar un pequeño test para comprobar la viabilidad y ampliarlo posteriormente en las implantaciones.

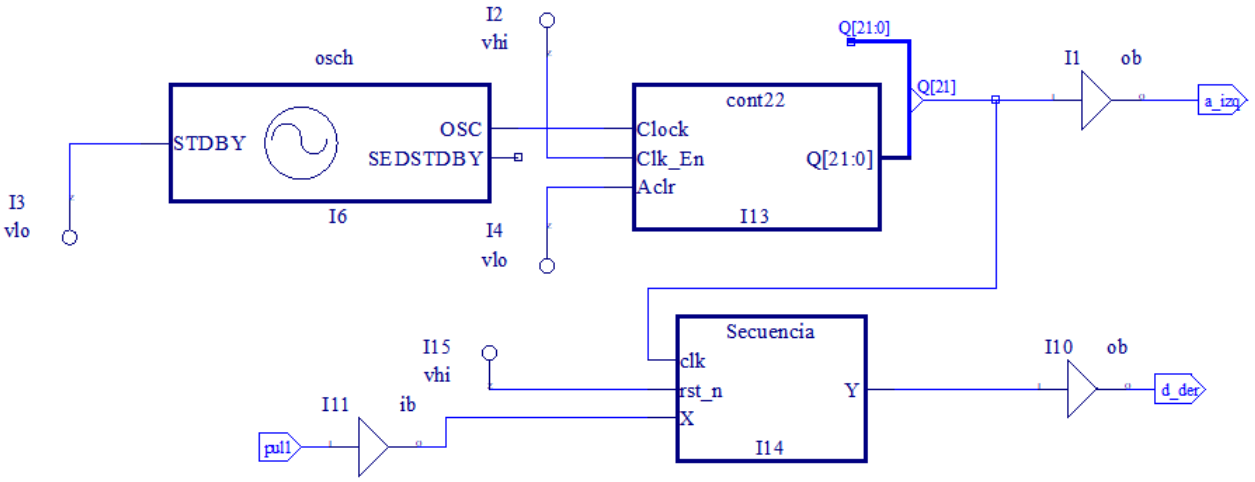



Figura 3.1.1. Esquema General Detector Secuencia

Como se puede observar en la figura 3.1.1 el esquema consta de 2 salidas y 1 entrada.

La entrada es la variable que se encarga de introducir la secuencia con la que trabaja el sistema. Las salidas en cambio se encargan de mostrar el estado del reloj que rige la lectura de información y la otra salida es la afectada si la secuencia es correcta.

El funcionamiento deseado es el siguiente:

- El reloj de 2 MHz se ralentiza mediante un contador de 22 bits de forma que se obtenga un reloj de 2 s de periodo.
- Este reloj se encargará de indicar al sistema cuando realizar la lectura de información, en este caso se realizará cada flanco de subida del mismo.
- Por último el bloque secuencia se encarga de obtener la información y operar con ella de forma que se controla la salida de la forma esperada.

Cabe destacar que en esta implantación el usuario debe adaptarse al ritmo del reloj para introducir adecuadamente la secuencia de su elección.

### 3.1.1 Generador reloj

En este apartado se explicará cómo se ha generado el reloj de la frecuencia deseada a partir del incluido en la FPGA.

La manera en lo que se ha conseguido esto es mediante un contador de pulsos, el funcionamiento consiste en que cada vez que el reloj rápido realiza un pulso de subida el contador aumenta su valor, debido a la forma de contar bits y a que el ciclo del contador se repite una vez alcanzado el máximo lo que se obtiene son relojes de diferentes frecuencias en cada bit de su salida.

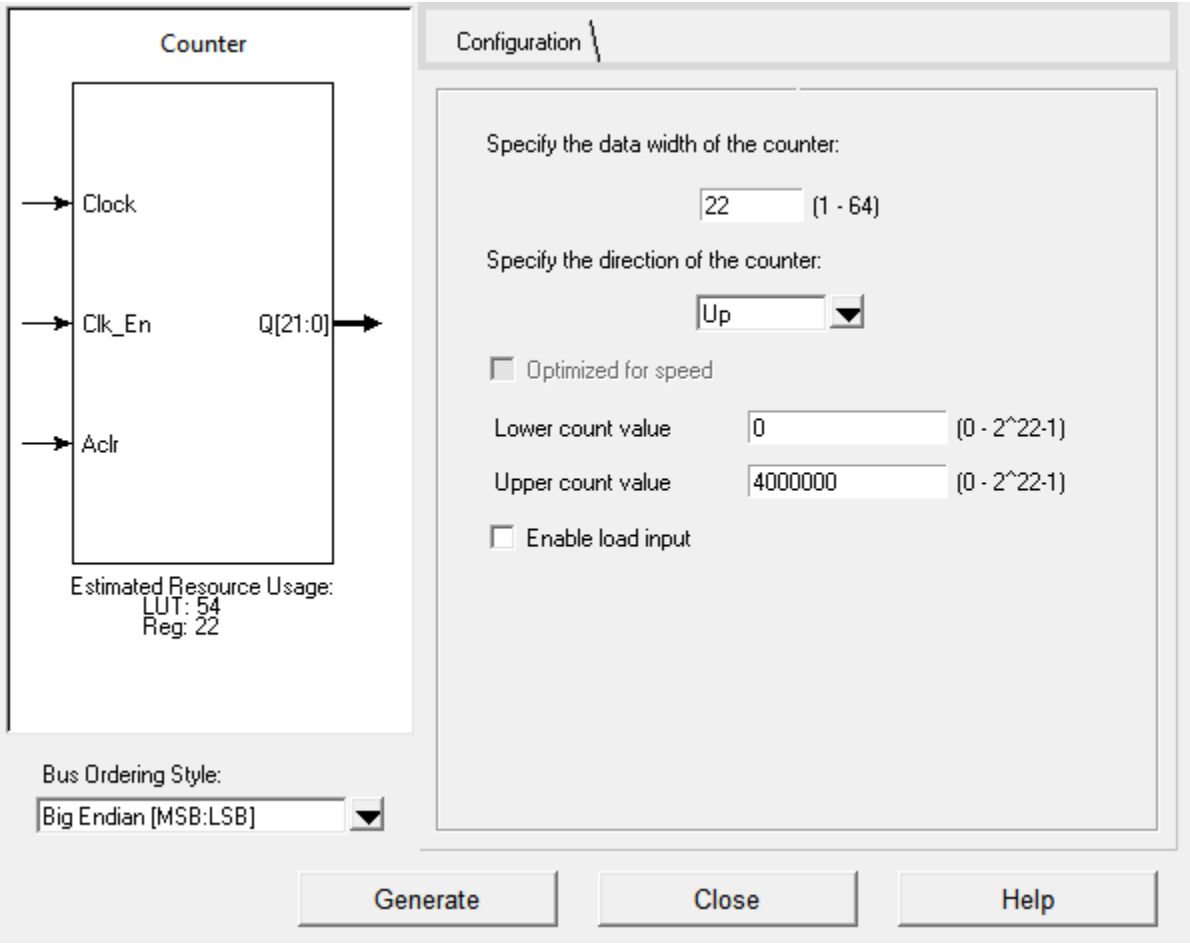



Figura 3.1.2. Contador 22 bits

En este caso como se observa la figura 3.1.2 debido al valor máximo del contador se divide la frecuencia hasta alcanzar los 0,5 Hz.

Una vez obtenida, se utilizara como señal de sincronismo para el siguiente bloque. Además de indicarse como salida para facilitar la interacción con el usuario.

### 3.1.2 Analizador secuencias

En este apartado se explica la forma en la que se analizan las secuencias introducidas y como se actúa en consecuencia con estas.

Para generar el bloque que se encargue de esta lógica se ha optado por apoyarse en una máquina de estados finitos mediante el programa Qfsm como se ha indicado anteriormente.

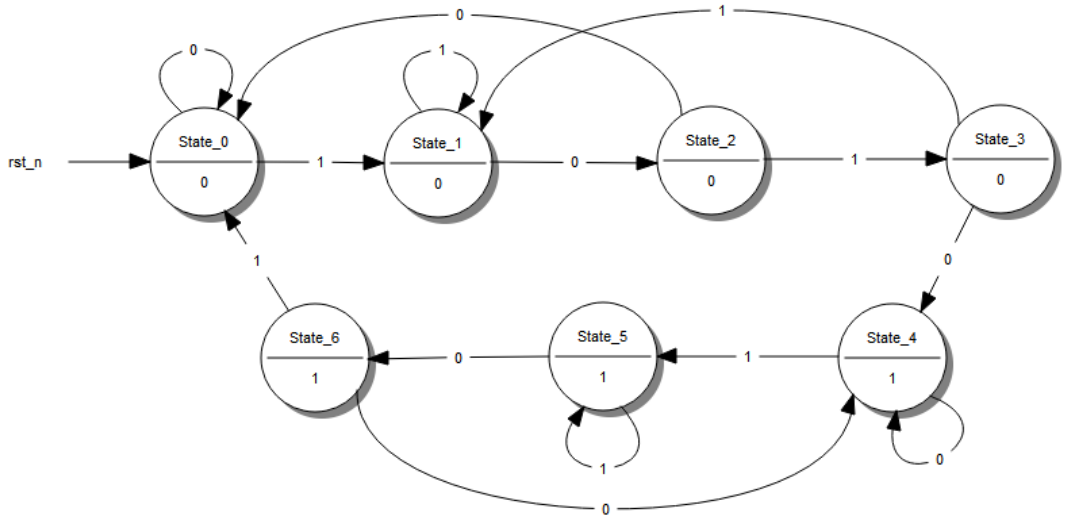



Figura 3.1.3. Maquina Estados Finitos de Secuencia

Como se observa en la imagen 3.1.3 la maquina consiste en un bucle cerrado que activa o desactiva la salida una vez introducida la secuencia correcta. En este caso la secuencia de activación sería 1010 y la de desactivación seria 101.

Este bloque realiza la lógica principal del proyecto, y sirve de ejemplo que será ampliado en la última implantación.

### 3.1.3 Asignación de pines

Para realizar la asignación de pines se utilizará la herramienta *spreedsheet view.* Esta herramienta permite alterar la configuración para soportar diferentes tensiones o adaptar la selección de pines a su configuración física con el resto de elementos

Por último se muestra como se han asignados los pines de la FPGA para conectarse al pulsador y los displays de los que consta el dispositivo.

| Name | Group By | Pin | BANK | BANK_VCC | VREF | IO_TYPE | PULLMODE |
|---|---|---|---|---|---|---|---|
| All Ports | N/A | N/A | N/A | N/A | N/A | | |
| Input | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| pul1 | N/A | 76 | 1 | Auto | N/A | LVCMOS33 | DOWN |
| Output | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| a_izq | N/A | 23 | 3 | Auto | N/A | LVCMOS33 | DOWN |
| d_der | N/A | 10 | 3 | Auto | N/A | LVCMOS33 | DOWN |

Figura 3.1.4. Asignación de pines

Algo a tener en cuenta para la asignación tanto en esta como en futuras implantaciones es que todos los puertos del dispositivo usa el mismo tipo, LVCMOS33, como se muestra en la figura.

## 3.2 Contraseña reconfigurable de cuatro dígitos

El siguiente proyecto que se va a realizar consiste en la detección de una contraseña de cuatro dígitos, siendo posible cambiar esta siempre y cuando no se esté en el medio del proceso de introducir una contraseña.

Este proyecto es un avance respecto a la versión anterior permitiendo tanto el cambio de la secuencia, como cambiando el formato de esta a decimal y adaptándose al ritmo de introducción de datos del usuario.

La forma de funcionar será la siguiente, se marcaran en los interruptores el digito a introducir en binario y cuando se quiera introducir se activará el pulsador. Cuando se desee cambiar la contraseña solo será necesario apretar el pulsador correspondiente, siempre y cuando no se esté resolviendo en ese momento.

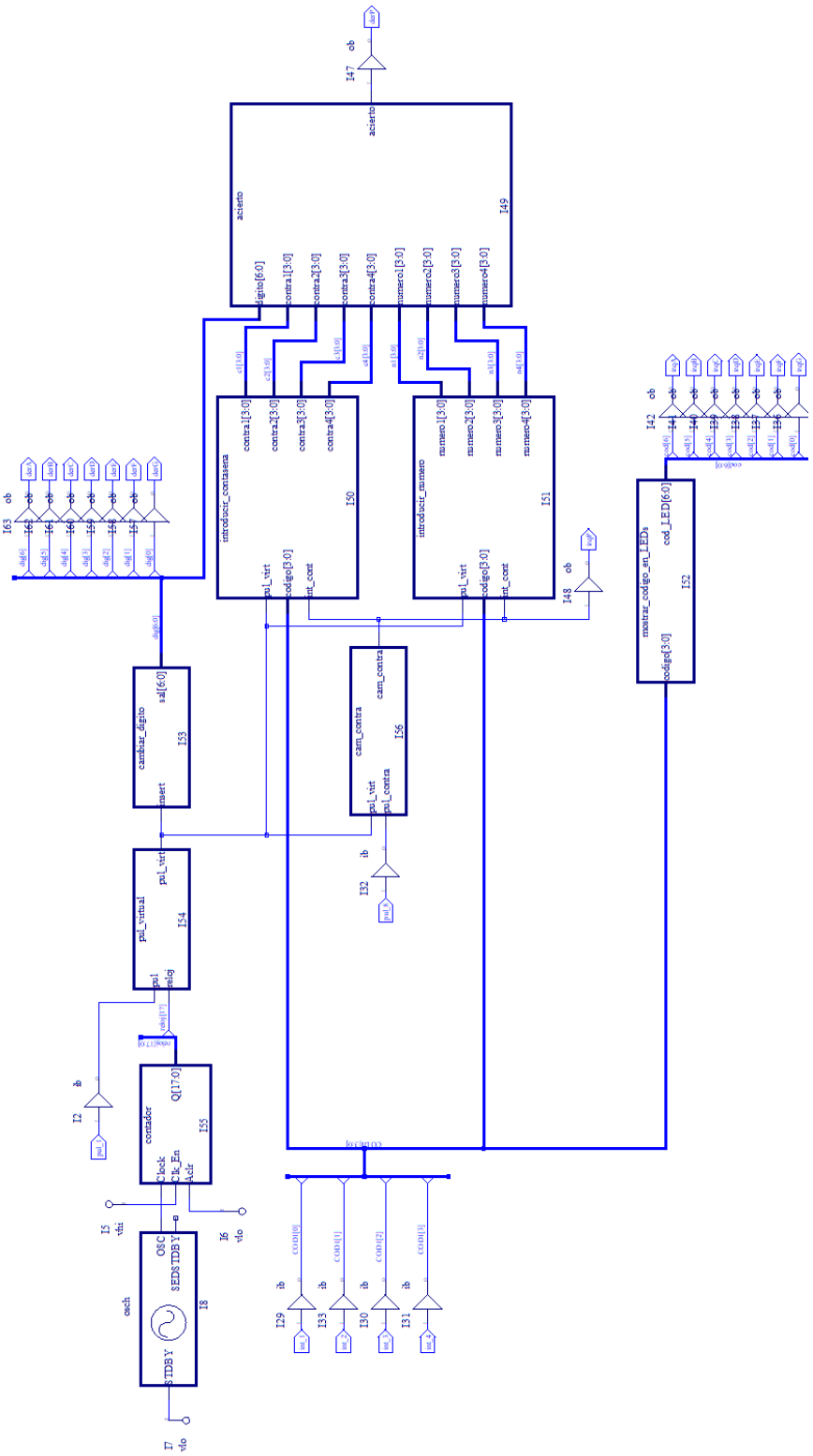


Figura 3.2.1. Esquema General Contraseña

La figura 3.2.1 muestra el esquema global del proyecto. Como se puede observar cuenta con 6 entradas y 16 salidas.

Las 4 entradas agrupadas indican los interruptores que mediante código binario se encargan de señalar el número a introducir en el instante en el que el pulsador que se encarga de indicar cuando son insertados sea activado.

El pulsador central se encarga de indicar al sistema cuando queremos modificar la contraseña, entrara en el estado de cambiar la contraseña cuando este pulsador sea apretado y no se esté en medio de resolver la contraseña introduciendo dígitos.

Debido a problemas observados mientras se comprobaba este sistema se observaron rebotes que ocurrían al introducir los números, por tanto se optó por poner un reloj seguido de un contador para ralentizarlo de forma que permita controlar cuando se introducen números de forma que solo se introduzca un digito por cada vez que el pulsador se activa. Otra opción hubiera sido introducir condensadores en los pulsadores, pero la solución adoptada no requiere de modificar el hardware.

A continuación se explica el funcionamiento deseado de los diferentes conjuntos de bloques

- Los bloques en la parte superior se encargan de controlar cuando se inserta un número de la forma ya indicada.
- El bloque central se encarga de controlar cuando se está resolviendo la contraseña o cuando se está cambiando.
- Los siguientes módulos se encargan de almacenar la contraseña como el número introducido.
- El bloque final se encarga de comprobarlos y se encarga de indicarlo mediante el LED indicado.
- Por último los bloques junto a las salidas se encargan de controlar los LEDs para mostrar tanto el número que se va a introducir como que digito de los cuatro se está introduciendo.

### 3.2.1 Generador del pulsador virtual

A continuación se describe como se genera el pulsador. Que se observa en la figura 3.2.2.

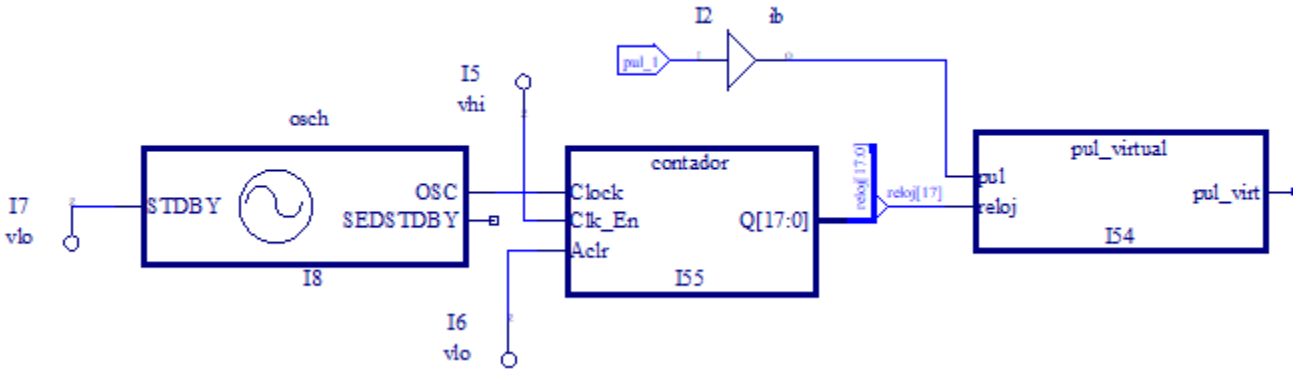



Figura 3.2.2.Esquema generador pulsador virtual

En este caso el único uso del reloj es para evitar que se introduzcan varios números por pulsación debido a rebotes en el pulsador.

Para ello se genera el reloj a 2MHz y mediante el contador configurado como se muestra a continuación se disminuye dicha frecuencia hasta un valor que evite los rebotes del pulsador pero que no sea tan baja que interfiera con la velocidad de introducir datos.

Para decidir cuál es la frecuencia adecuada se partió de una frecuencia de 2Hz y se fue aumentando, disminuyendo los bits del contador, hasta llegar a una frecuencia cómoda que no interfería con el usuario.

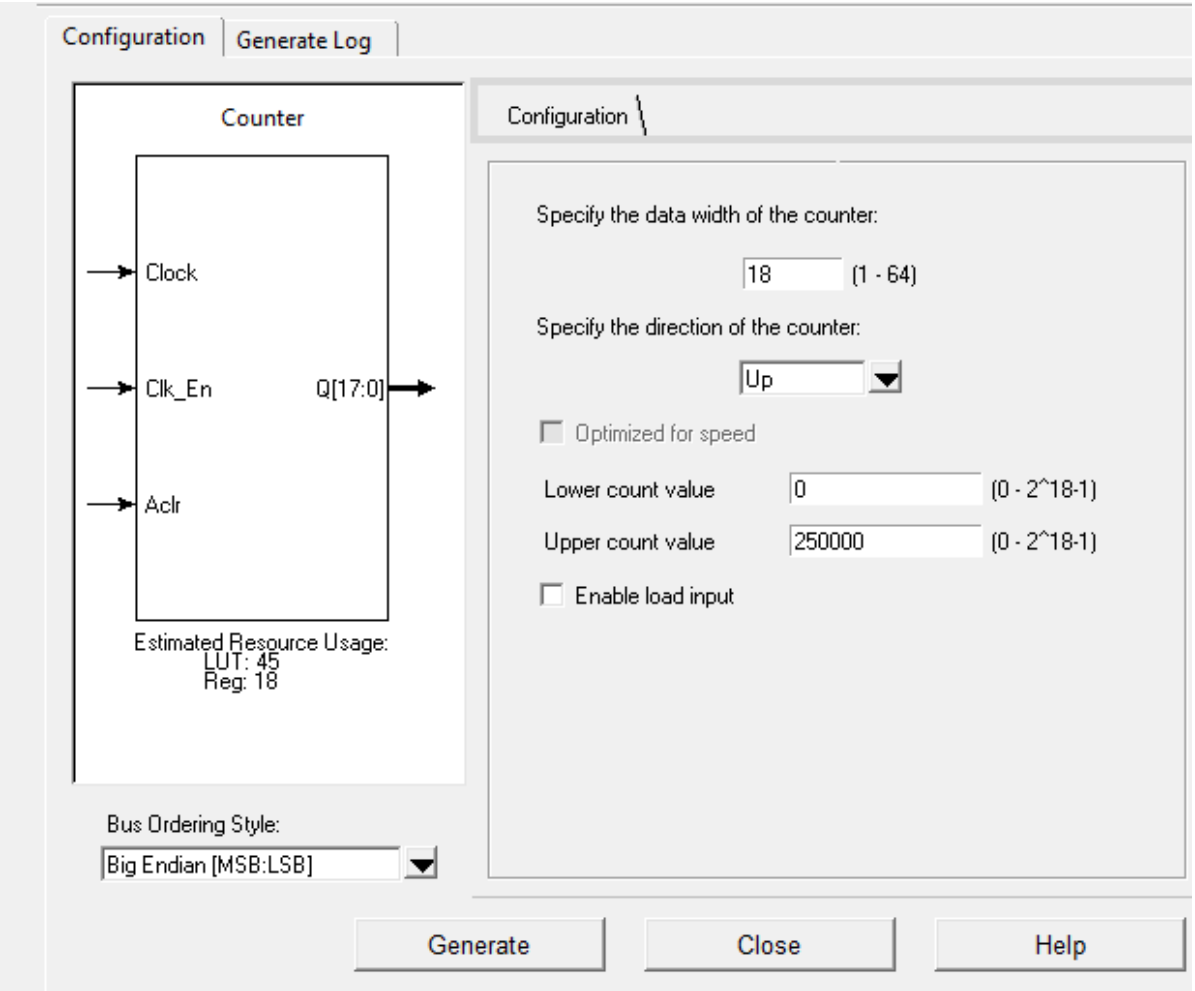



Figura 3.2.3. Contador 18 bits

Posteriormente con la salida del contador y el pulsador se generará el pulsador virtual. La forma que eso se hace se muestra en la figura 3.2.3.

Básicamente lo que este módulo hace es que solo se lea la pulsación en el instante en el que el reloj tiene un flanco de subida, de esa forma cualquier rebote que pudiera haber al pulsar será ignorado por el sistema.

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

entity pul_virtual is
port(
    pul: in std_logic;
    reloj: in std_logic;
    pul_virt: out std_logic
    );
end pul_virtual;

architecture arch of pul_virtual is
begin
    process(reloj)
    begin
        if (reloj'event and reloj='1') then
            pul_virt<=pul;
        end if;
    end process;
end arch;
```

Figura 3.2.4. Código pulsador virtual

### 3.2.2 Indicador cambio contraseña

A continuación se explica cómo funciona la lógica para cambiar la contraseña. De esto se encarga el bloque con el código de la figura 3.2.5.

El funcionamiento es el siguiente, el proceso del código se encarga de controlar que no se esté en medio de la introducción de dígitos, además cada vez que se ha introducido los 4 dígitos se vuelve a valor nulo permitiendo que una vez introducida la contraseña se pueda resolver.

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

entity cam_contra is
port(
    pul_virt: in std_logic;
    pul_contra: in std_logic;
    cam_contra: out std_logic
    );
end cam_contra ;

architecture arch of cam_contra  is
signal aux : std_logic_vector(1 downto 0);
begin
    process(pul_virt)
    begin
        if(pul_virt'event and pul_virt='1') then
            if (aux="00") then
                aux<="01";
            elsif (aux="01") then
                aux<="10";
            elsif (aux="10") then
                aux<="11";
            elsif (aux="11") then
                aux<="00";
            end if;
        end if;
    end process;
    aux<="00" when (aux="UU") else
        aux;
    cam_contra<='1' when (pul_contra='1' and aux="00") else
            '0' when (aux="00" and pul_virt'event and pul_virt='0');
end arch;
```

Figura 3.2.5. Código cambio contraseña

### 3.2.3 Registro contraseña y solución

En este apartado se describe como se realiza el almacenamiento tanto de la contraseña como de la solución propuesta, de forma que posteriormente se puedan comparar ambos valores.

Esto se hace la forma que se muestra a continuación, en este caso se ha optado por un registro sencillo de desplazamiento de los cuatro dígitos.

Este sistema no tiene en cuenta en qué posición de digito se encuentra el que se va a introducir a continuación por lo que será necesario que otros módulos se encarguen de esto.

Ambos registros son iguales, la única diferencia reside en que uno está activo mientras se está cambiando la contraseña mientras que el otro funciona al revés.

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

entity introducir_contasena is
port(
    pul_virt: in std_logic;
    codigo: in std_logic_vector (3 downto 0);
    int_cont: in std_logic;
    contra1: out std_logic_vector (3 downto 0);
    contra2: out std_logic_vector (3 downto 0);
    contra3: out std_logic_vector (3 downto 0);
    contra4: out std_logic_vector (3 downto 0)
    );
end introducir_contasena;

architecture arch of introducir_contasena is
signal con1_aux: std_logic_vector (3 downto 0);
signal con2_aux: std_logic_vector (3 downto 0);
signal con3_aux: std_logic_vector (3 downto 0);
signal con4_aux: std_logic_vector (3 downto 0);
begin
    process(pul_virt)
    begin
        if (int_cont='1') then
            if(pul_virt'event and pul_virt='1') then
                con1_aux<=con2_aux;
                con2_aux<=con3_aux;
                con3_aux<=con4_aux;
                con4_aux<=codigo;
            end if;
        end if;
    end process;
    contra1<=con1_aux;
    contra2<=con2_aux;
    contra3<=con3_aux;
    contra4<=con4_aux;
end arch;
```

Figura 3.2.6. Código registro contraseña

### 3.2.4 Salidas formato decimal

La principal función de estos bloques es la de permitir al usuario una mayor capacidad y conocimiento de la situación del dispositivo.

En el caso de este proyecto destacan dos partes, indicar cuál es el digito que tiene que introducir y por otro lado mostrar cual sería el siguiente número que introduciría en caso de activar el pulsador de insertar, esto se hace para evitar problemas en los que interruptores se encuentren en posiciones intermedias y no se esté seguro del valor que se va a introducir.

Ambos bloques, que se muestran a continuación se comunican directamente al exterior mediante los displays de siete segmentos que contiene el dispositivo.

En el caso del display que muestra los dígitos esta información será utilizada en el apartado siguiente.

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

entity cambiar_digito is
port(
    sal: out std_logic_vector (6 downto 0);
    insert: in std_logic
    );
end cambiar_digito;

architecture arch of cambiar_digito is
signal digito_aux: std_logic_vector (1 downto 0);
begin
    process(insert)
    begin
        if(insert'event and insert='1') then
            if (digito_aux="00") then
                digito_aux<="01";
            elsif (digito_aux="01") then
                digito_aux<="10";
            elsif (digito_aux="10") then
                digito_aux<="11";
            elsif (digito_aux="11") then
                digito_aux<="00";
            end if;
        end if;
    end process;
    digito_aux<="00" when (digito_aux="UU");
    sal<="1101101" when digito_aux="01" else
        "1111001" when digito_aux="10" else
        "0110011" when digito_aux="11" else
        "0110000" when digito_aux="00" else
        "1001111";
end arch;
```

Figura 3.2.7. Código displays dígito

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

entity mostrar_codigo_en_LEDs is
port(
    codigo: in std_logic_vector (3 downto 0);
    cod_LED: out std_logic_vector (6 downto 0)
    );
end mostrar_codigo_en_LEDs;

architecture arch of mostrar_codigo_en_LEDs is
begin
    cod_LED <="0110000" when codigo="0001" else
        "1101101" when codigo="0010" else
        "1111001" when codigo="0011" else
        "0110011" when codigo="0100" else
        "1011011" when codigo="0101" else
        "1011111" when codigo="0110" else
        "1110000" when codigo="0111" else
        "1111111" when codigo="1000" else
        "1111011" when codigo="1001" else
        "1111110" when codigo="0000" else
        "1001111";

end arch;
```

Figura 3.2.8. Código displays número

Hay que señalar que es importante el orden en el que se realiza la asignación de los pines de salida para este caso, ya que sino la información mostrada no coincidirán con los números previstos.

### 3.2.5 Comprobación solución

Después de almacenar toda la información introducida de forma secuencial se trabaja con ella en este caso la operación a realizar, al ser una contraseña, es comprobar si ambas secuencias son iguales.

Hay que tener en cuenta que, debido a la sencillez de los registros, este bloque ha de tener en cuenta que solo debe comprobar si la solución es correcta una vez introducidos los cuatro dígitos y no mientras aún falta información por aportar.

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

entity acierto is
port(
    digito: in std_logic_vector (6 downto 0);
    contra1: in std_logic_vector (3 downto 0);
    contra2: in std_logic_vector (3 downto 0);
    contra3: in std_logic_vector (3 downto 0);
    contra4: in std_logic_vector (3 downto 0);
    numero1: in std_logic_vector (3 downto 0);
    numero2: in std_logic_vector (3 downto 0);
    numero3: in std_logic_vector (3 downto 0);
    numero4: in std_logic_vector (3 downto 0);
    acierto: out std_logic
    );
end acierto;

architecture arch of acierto is
begin
    acierto<='1' when (numero1=contra1 and numero2=contra2 and numero3=contra3 and numero4=contra4 and digito="0110000") else
            '0';
end arch;
```

Figura 3.2.9. Código comprobación

Como se observa solo realiza la comprobación cuando el display está requiriendo el primer digito, por lo tanto todavía no ha empezado recogida de nueva información.

### 3.2.6 Asignación de pines

A continuación se muestran cómo se han configurado todos los pines de acuerdo a los nombres mostrados en la figura 3.2.1.

| Name | Group By | Pin | BANK | BANK_VCC | VREF | IO_TYPE | PULLMODE |
|---|---|---|---|---|---|---|---|
| All Ports | N/A | N/A | N/A | N/A | N/A | | |
| Input | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| int_1 | N/A | 86(86) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| int_2 | N/A | 85(85) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| int_3 | N/A | 84(84) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| int_4 | N/A | 83(83) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| pul_1 | N/A | 76(76) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| pul_8 | N/A | 93(93) | 1(1) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| Output | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| derA | N/A | 21(21) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derB | N/A | 22(22) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derC | N/A | 11(11) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derD | N/A | 10(10) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derE | N/A | 13(13) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derF | N/A | 12(12) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derG | N/A | 14(14) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derP | N/A | 9(9) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqA | N/A | 23(23) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqB | N/A | 25(25) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqC | N/A | 33(33) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqD | N/A | 34(34) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqE | N/A | 35(35) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqF | N/A | 26(26) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqG | N/A | 24(24) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqP | N/A | 32(32) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |

Figura 3.2.10. Asignación de pines

## 3.3 Extracción de información de un sensor

En este proyecto, se extrae la secuencia a partir de la información suministrada por un sensor.

La información está codificada como se muestra en la figura 3.3.10 y se trata de un protocolo específico, no pudiéndose emplear los buses estándar, como el I2C, contenidos en el dispositivo, por lo que se debe desarrollar una circuitería  específica para extraer la información.

En el proyecto desarrollado, se extrae la secuencia proporcionada por el sensor y se generan señales de aviso en función de si los valores proporcionados por este están dentro o fuera de un rango predeterminado.

### 3.3.1 Sensor

En este apartado se van a explicar las características principales y los motivos por lo que se ha optado por el sensor utilizado.

Como sensor se emplea el DHT22, del que se muestra una imagen en la figura 3.3.1.

Figura 3.3.1. Sensor DHT22

El sistema de comunicación es similar a otros sistemas inalámbricos como el infrarrojo de un mando a distancia. Una ventaja de este sistema es que reduce el cableado.

Al ser la comunicación por un único cable el sensor no envía información continuamente sino que está en modo reposo hasta que la FPGA le mande una señal de control, a partir de ese momento el sensor envía la información de forma secuencial mediante 40 bits que incluyen tanto la información de temperatura como humedad así como bits de redundancia para comprobar posibles errores en la transmisión.

Para este sensor la señal de mando consiste en mantener en nivel bajo el canal, que está por defecto a nivel alto, durante un periodo de entre 0,8 y 20 ms, siendo recomendado 1 ms.

Una vez recibida la señal de control el sensor tomará el control de la línea de comunicación realizando la transmisión, en un formato propio, de los bits que posteriormente se traducirán en la información a tratar.

Una vez realizada la transmisión el sensor liberará la línea de comunicación. Posteriormente el sensor requiere la realización de un reset a lo largo 2 s, durante este proceso el sensor no mandará información útil.

Al final de este documento se incluye en el segundo anexo más documentación con información de este sensor que no se ha considerado necesaria incluirla en el documento en sí.

### 3.3.2 Esquema general

A continuación se muestra el esquema global del proyecto donde se observan los diferentes bloques que lo forman y como se relacionan entre sí.

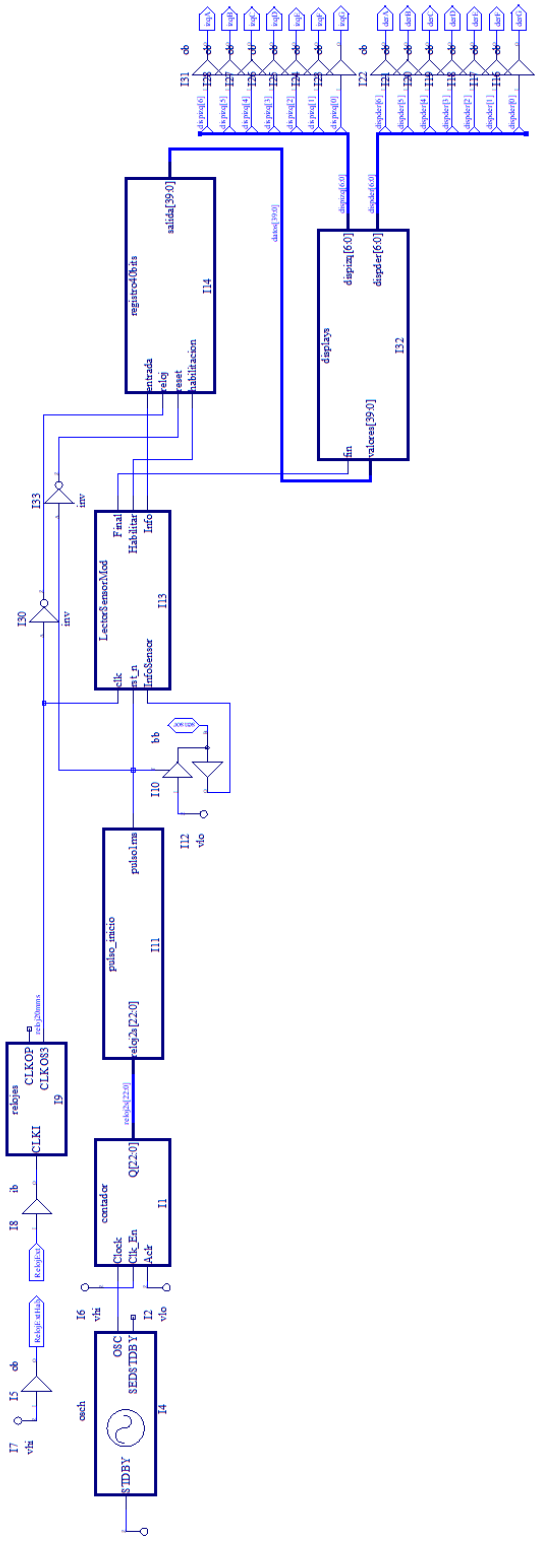


Figura3.3.2. Esquema General Control Temperatura-Humedad

La figura 3.3.2 muestra el esquema global del proyecto. Como se puede observar cuenta con 1 entradas, 15 salidas y 1 puerto bidireccional.

El puerto bidireccional corresponde al sensor de temperatura y humedad conectado.

La salida situada en la parte superior izquierda es la habilitación del reloj externo incluido en el dispositivo, mientras que la entrada cercana es el reloj.

El resto de salidas están ligadas a los displays por los que se muestra la información al usuario.

A continuación se explica el funcionamiento de los diferentes conjuntos de bloques

- Los bloques en la parte superior se encargan de generar un reloj de 20 µs de periodo a partir del reloj externo de 50 MHz.
- Los bloques de la izquierda y centro se encargan de generar un pulso de 1 ms cada 3,8 s que se utilizara para requerir información.
- El bloque siguiente está conectado al sensor y traduce la información en forma de pulsos a bits de información.
- Por último los bloques finales se encargan de almacenar los bits generados y tratarlos para obtener una salida útil.

Los inversores situados están para evitar que el registro tome datos mientras se está produciendo la lectura y evitar que realice el reset cuando va a comenzar la toma de datos.

Por ultimo debido a que la línea entre sensor y FPGA está por defecto a nivel alto para mandar un nivel bajo lo único que se hace es activar el triestado del buffer bidireccional, teniendo la entrada del mismo siempre nivel bajo.

### 3.3.2 Generador pulso

Para que el sensor mande información es necesario generar un pulso de 1 ms cada 2s o más para la consecución de este objetivo se recurre a los bloques mostrados en la figura 3.3.3.

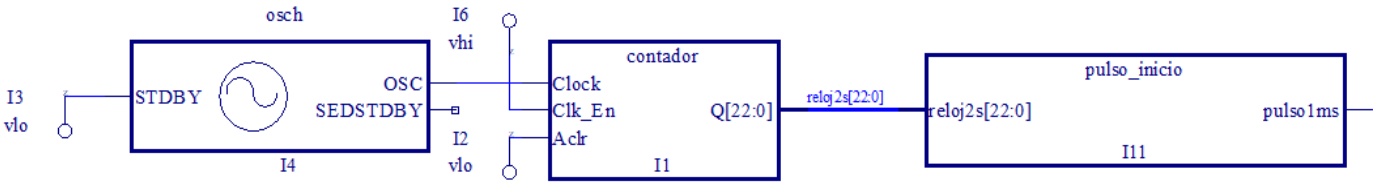



Figura 3.3.3. Esquema generador de pulso

Este esquema funciona de la siguiente forma, al igual que en proyectos anteriores se parte del reloj interno y se conecta a un contador para obtener relojes a diferentes frecuencias.

En este caso el contador es de 23 bits como observa ve en la figura 3.3.3, obteniendo una división de la frecuencia hasta llegar a un periodo de 4 s aproximadamente, se podría reducir el periodo, pero manteniéndolo superior a 2 s, debido a las restricciones impuestas por el propio sensor.

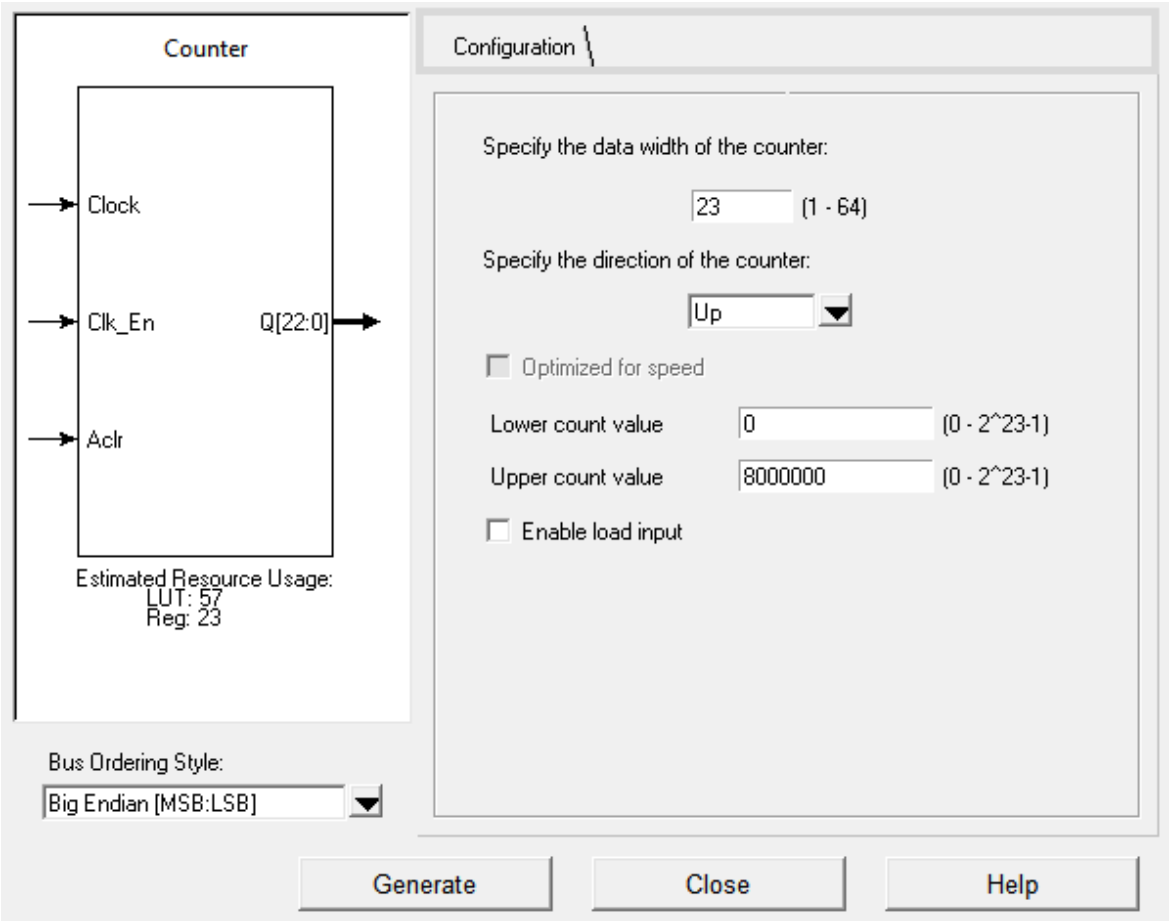



Figura 3.3.4. Contador 23 bits

Posteriormente a partir de los diferentes relojes obtenidos a partir del contador estos se pasan por un bloque que genere un pulso de 1 ms cada más de 2 s, este bloque se muestra en la siguiente imagen.

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

entity pulso_inicio is
port(
    pulsolms: out std_logic;
    reloj2s: in std_logic_vector (22 downto 0)
    );
end pulso_inicio;

architecture arch of pulso_inicio is
begin
    pulsolms<=(reloj2s(22) or reloj2s(21) or reloj2s(20) or reloj2s(19) or // or reloj2s(12) or reloj2s(11));
end arch;
```

Figura 3.3.5. Generador de pulso

En este caso debido a que el triestado se activa a nivel bajo el pulso a generar tiene que estar a nivel bajo durante 1 ms y el resto del tiempo debe estar a nivel alto.

En la siguiente imagen se muestra el resultado de una simulación similar solo que tiene un funcionamiento inverso. Cabe resaltar que el eje superior se encuentra en milisegundos.

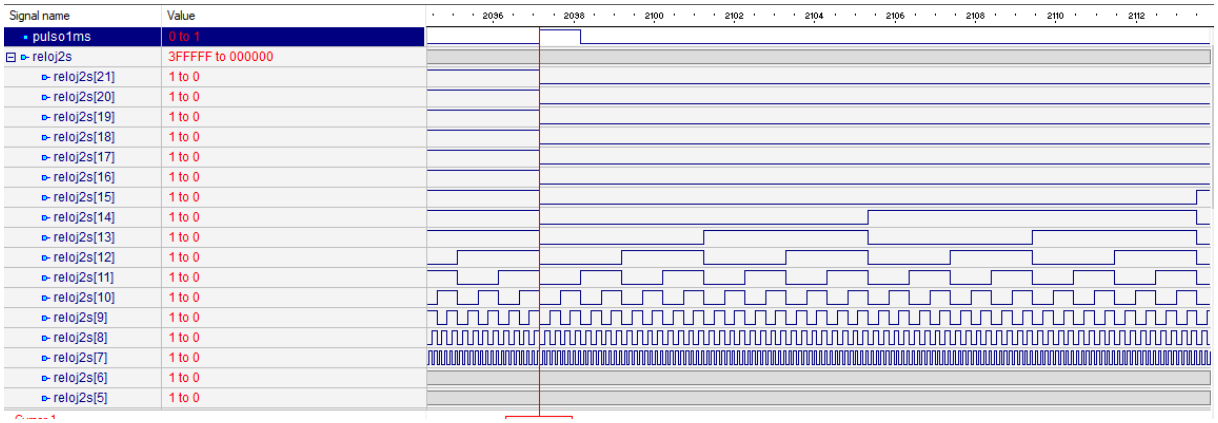



Figura 3.3.6. Simulación generador de pulsos

Este pulso se conecta directamente al triestado del buffer bidireccional y se encargará de mandar la señal que activa el sensor.

### 3.3.3 Generador reloj preciso

Para la lectura de la información en formato PWM se utilizará un reloj de 20 µs que indique al sistema los instantes en los que captar información del sensor.

El motivo por lo que se ha optado por este reloj se explicara en el siguiente apartado, está determinado en función de la máquina de estados finitos utilizada para decodificar la información y los tiempos en los que esta se envía.

La forma de generar el reloj se observa en la figura 3.3.7. Se utiliza el sintetizador de frecuencia contenido en el dispositivo, alimentado por el reloj de 50 MHz y configurado como se muestra en la figura 3.3.8.

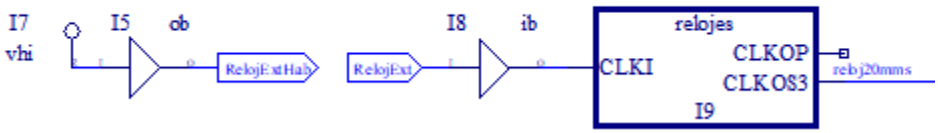



Figura 3.3.7. Generador reloj 20 µs

El motivo para utilizar este reloj y no el ya incluido en la FPGA y utilizado anteriormente es la mayor precisión del exterior permitiendo obtener información con menor probabilidad de errores.

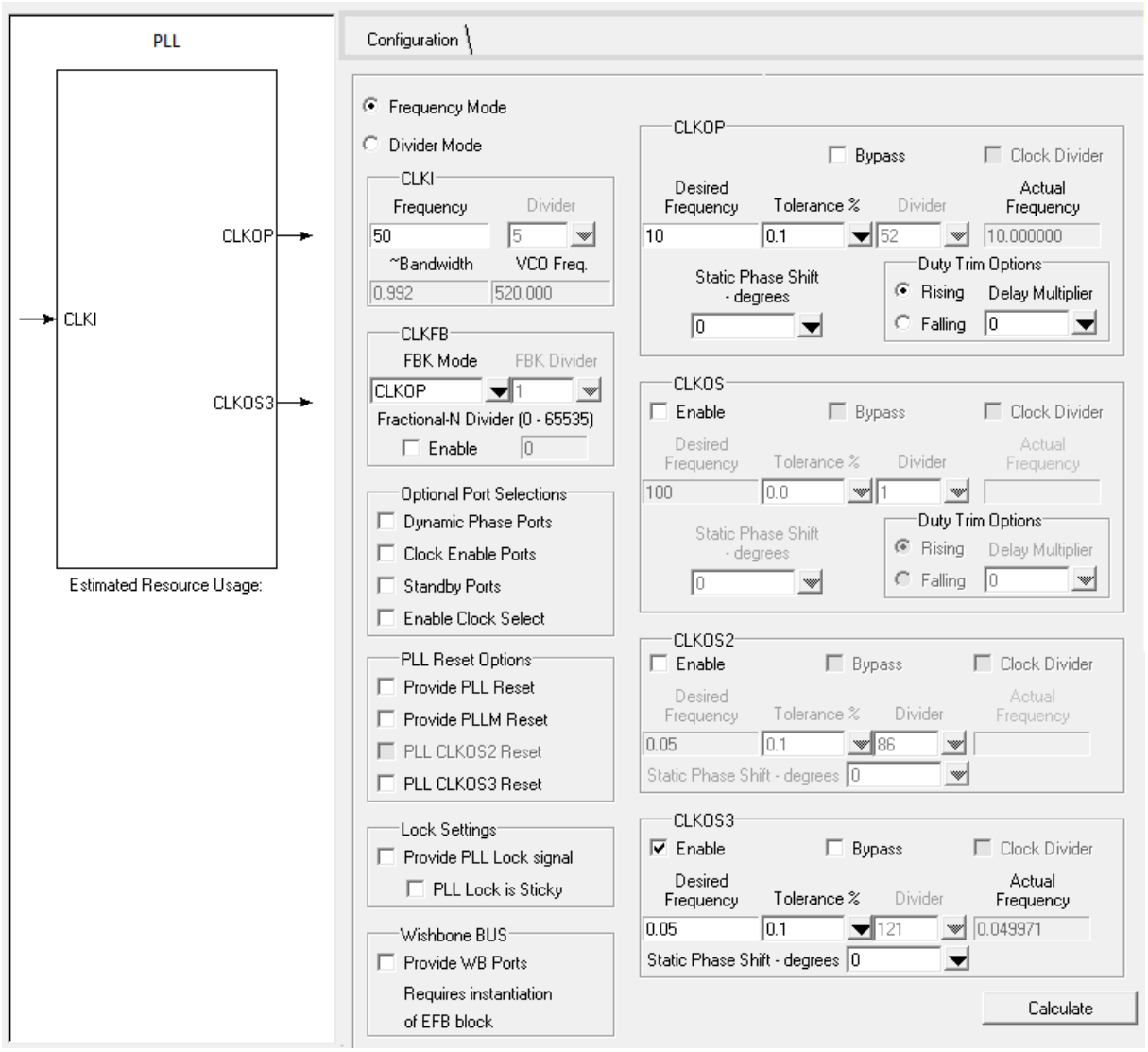



Figura 3.3.8. Configuración PLL

Como se observa en la configuración de la figura 3.3.8 a partir del reloj de 50 MHz se genera uno de 50 kHz, no se ha utilizado un contador debido a que el módulo usado permite también una gran precisión.

### 3.3.4 Obtención y tratamiento de datos

Una vez generadas todas las señales indicadas anteriormente ya es posible el control del sensor mediante la FPGA y la lectura de los datos obtenidos por el mismo.

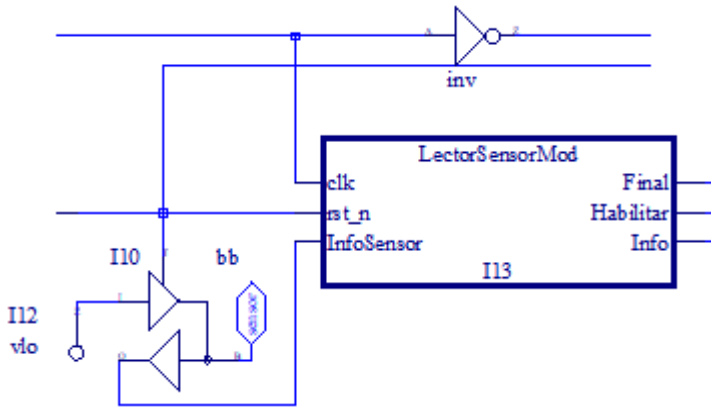



Figura 3.3.9. Lector de información

Como se observa en la figura 3.3.9 el sensor está conectado a un buffer bidireccional que le permite interactuar con la FPGA, la información tanto enviada al sensor como la recibida por él se envía a una máquina de estados finitos que la transforme en una secuencia de bits, que sea interpretable más fácilmente por el sistema.

La forma de las señales enviadas por el sensor tiene la forma y tiempos indicados en las siguientes imágenes. Debido a esto se optó por el reloj ya comentado anteriormente.

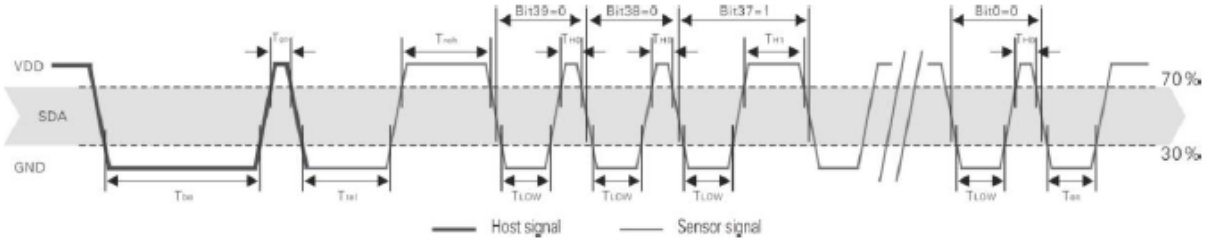



Figura 3.3.10. Forma de la onda del sensor

| Symbol | Parameter | min | typ | max | Unit |
|---|---|---|---|---|---|
| $T_{be}$ | Host the start signal down time | 0.8 | 1 | 20 | mS |
| $T_{go}$ | Bus master has released time | 20 | 30 | 200 | μs |
| $T_{rel}$ | Response to low time | 75 | 80 | 85 | μs |
| $T_{reh}$ | In response to high time | 75 | 80 | 85 | μs |
| $T_{LOW}$ | Signal "0", "1" low time | 48 | 50 | 55 | μs |
| $T_{H0}$ | Signal "0" high time | 22 | 26 | 30 | μs |
| $T_{H1}$ | Signal "1" high time | 68 | 70 | 75 | μs |
| $T_{en}$ | Sensor to release the bus time | 45 | 50 | 55 | μs |

Figura 3.3.11. Tiempos de la onda del sensor

La comunicación entre sensor y FPGA ocurre de la forma siguiente:

- Primero la FPGA debe mandar un pulso a nivel bajo de entre 0,8 y 20 ms.
- Después de ello el sensor responde con una onda cuadrada de unos 80 ms aproximadamente.
- A continuación empieza la transmisión de los bits, lo que marca la diferencia ente 0s y 1s es la duración del nivel de alta siendo entre 22 y 30 ms para un 0 y entre 68 y 75 ms para un 1.
- Por último una vez terminada la transmisión de datos la línea vuelve a su estado por defecto, el nivel alto.

Un ejemplo de la secuencia de información que se intercambia entre el sensor y la FPGA extraída con el analizador lógico se muestra en la figura 3.3.12.


Figura 3.3.12. Ejemplo señal sensor

Conocida toda esta información la máquina de estados finitos que se ha diseñado para interpretar esta información es la que se muestra en la siguiente imagen.

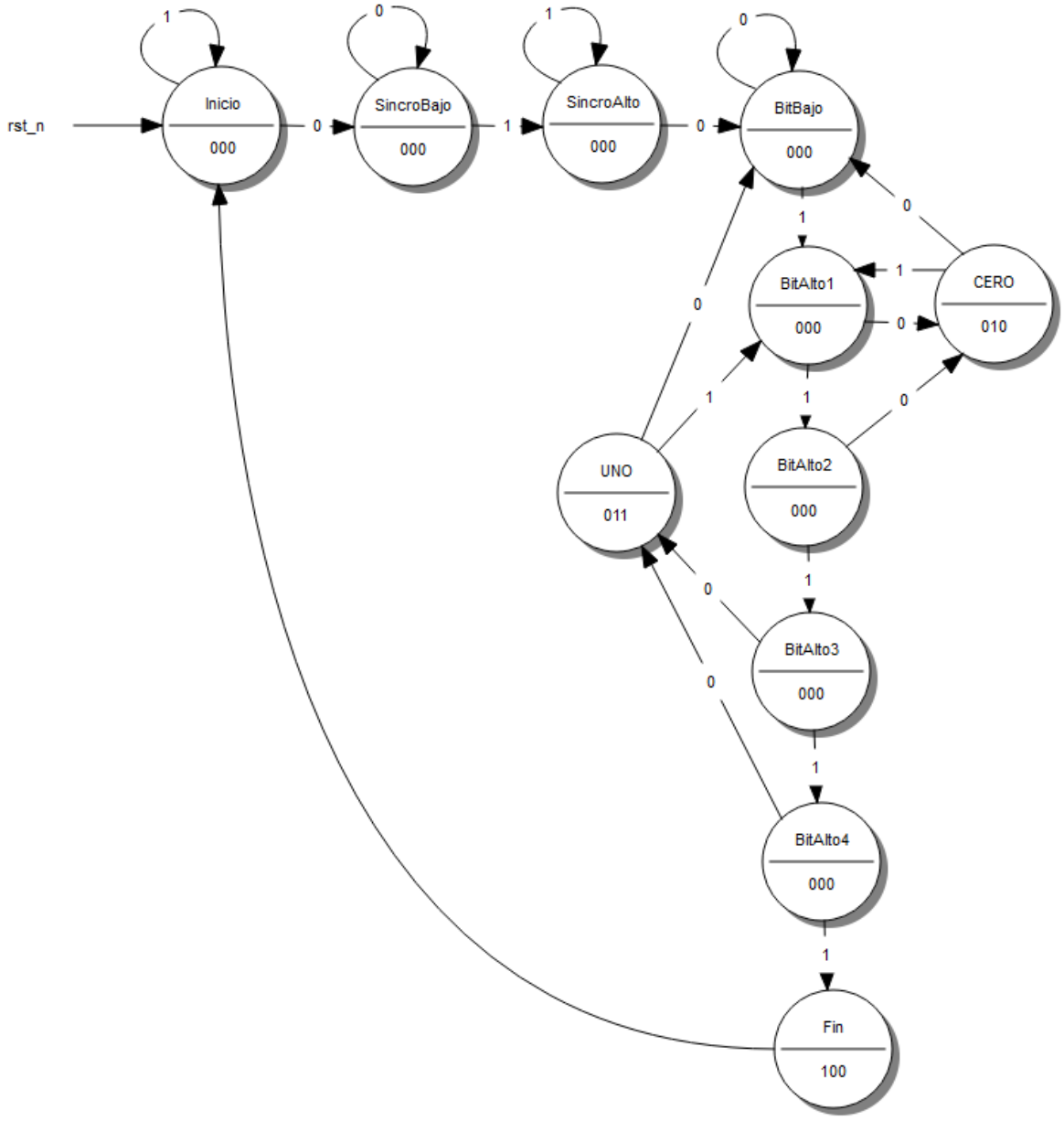



Figura 3.3.13. Máquina estado finitos sensor

Como se observa, una vez empieza la transmisión de información, la máquina de estados tiene 3 opciones:

- Recibir 2 señales o menos a nivel alto lo que indicaría un 0.

  Recibir 4 o 3 señales a nivel alto indicando un 1.

- Recibir 5 o más señales a nivel alto lo que indicaría que se ha terminado la transmisión de información.

La máquina de estados finitos genera tres señales de salida, estas se van a emplear para controlar el registro de desplazamiento en el que se va a almacenar la información, a saber, la información propiamente dicha, una señal de habilitación y otra señal que indica cuando se ha transmitido toda la información.

Cabe destacar que como reset de la máquina de estados finitos se ha utilizado la señal de 1 ms que controla al sensor, de esta forma se garantiza que cuando vaya a comenzar la transmisión de información el bloque este en situación de recibir la información.

### 3.3.5 Registro de información y comprobación de valores

La información extraída, se almacena en un registro de desplazamiento de 40 bits. La descripción VHDL del registro se presenta en la figura 3.3.14.

Una vez realizados los pasos anteriores se cuenta con una información procesada que permite un fácil almacenaje y tratamiento para poder obtener información útil de ella.

En este proyecto la información que se desea obtener es si la temperatura y humedad de un local está dentro de unos parámetros aceptables o avisar de las desviaciones y en qué sentido es para actuar en consecuencia.

Una vez obtenida esta información se ha optado por mostrarla mediante los displays, pero también sería posible mandarlo mediante conexión a otro dispositivo o enviar alertas sonoras en caso de añadir dispositivos auxiliares.

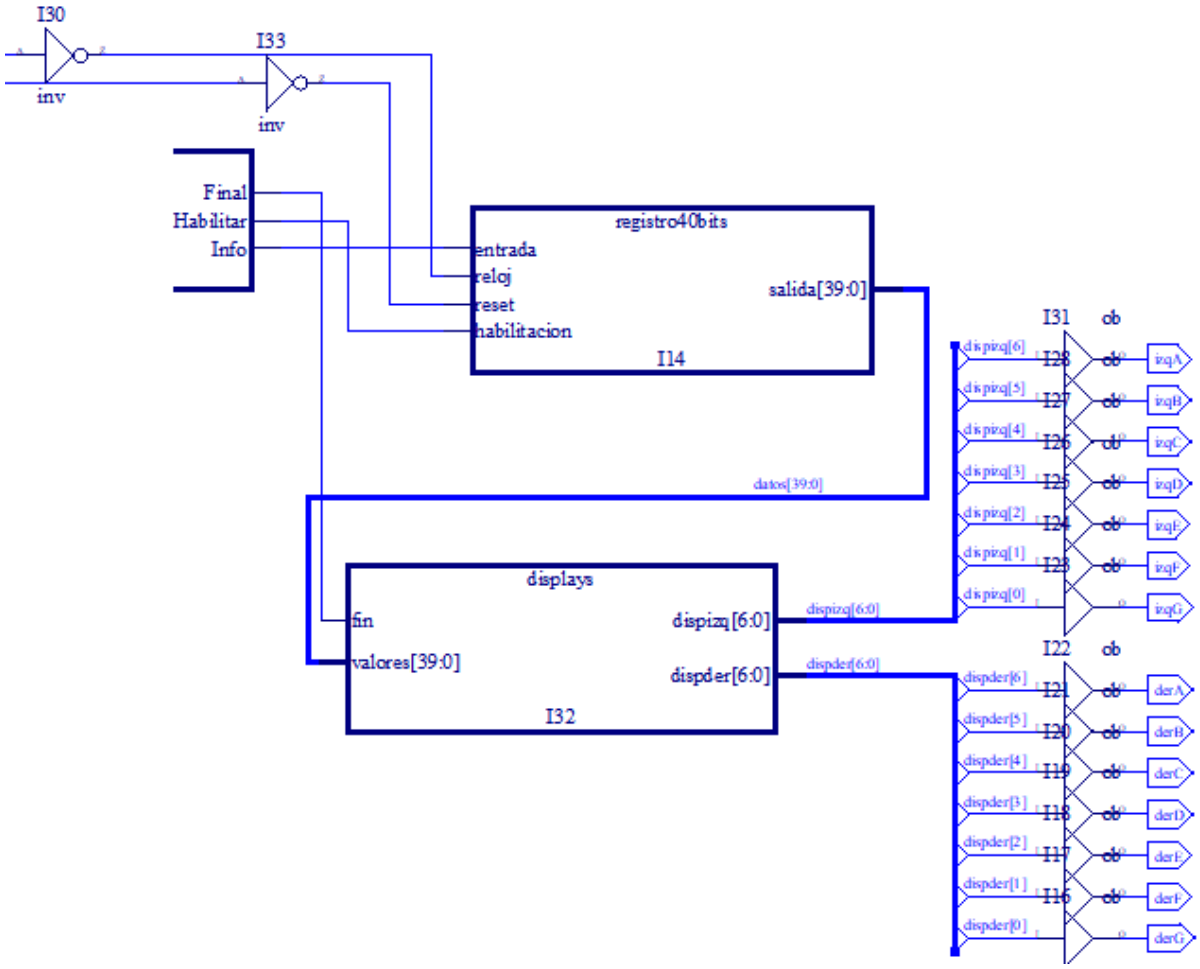



Figura 3.3.14. Esquema registro y muestra de datos

La forma en el que el conjunto funciona es la siguiente. Primero el registro almacena la información según sale de la máquina de estados finitos y la agrupa para crear un vector con ella. Después el siguiente bloque la lee y según esté dentro o fuera de unos rangos predefinidos decide que muestran los displays.

Hay que resaltar los inversores situados en la parte superior para evitar que el registro lea información del módulo anterior mientras está en el proceso de cambiar de valor, el otro inversor es debido a que el registro tiene un reset a nivel alto. De esta forma, la máquina de estados finitos actualiza su estado en los flacos de subida del reloj, mientras que el registro almacena la información en los flancos de bajada.

El registro se muestra en la figura 3.3.15, consiste en un registro de 40 bits que incluye reset y habilitación a nivel alto.

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

entity registro40bits is
port(
    entrada: in std_logic;
    reloj: in std_logic;
    reset: in std_logic;
    habilitacion: in std_logic;
    salida: out std_logic_vector (39 downto 0)
    );
end registro40bits;

architecture arch of registro40bits is
signal salida_aux: std_logic_vector (39 downto 0);
begin
    process(reloj,reset)
    begin
        if (reset='1') then
            salida_aux<="0000000000000000000000000000000000000000";
        elsif (reloj'event and reloj ='1') then
            if (habilitacion ='1') then
                for I in 0 to 38 loop
                    salida_aux(I+1)<=salida_aux(I);
                end loop;
                salida_aux(0)<=entrada;
            end if;
        end if;
    end process;
    salida<=salida_aux;
end arch;
```

Figura 3.3.15. Código registro 40 bits

El bloque que comprueba que los valores de temperatura y humedad están dentro de rango se muestra en la figura 3.3.16.

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

entity displays is
port(
    fin: in std_logic;
    valores: in std_logic_vector (39 downto 0);
    dispizq: out std_logic_vector (6 downto 0);
    dispder: out std_logic_vector (6 downto 0)
    );
end displays;

architecture arch of displays is
signal temp: std_logic_vector (15 downto 0);
signal hume: std_logic_vector (15 downto 0);
begin
    process(fin)
    begin
        if (fin'event and fin ='1') then
            temp<=valores(23)&valores(22)&valores(21)&//&valores(11)&valores(10)&valores(9)&valores(8);
            hume<=valores(39)&valores(38)&valores(37)&//&valores(27)&valores(26)&valores(25)&valores(24);
        end if;
    end process;
    dispizq<="1000111" when (temp>"0100000000000000") else -- F temp<0
            "1001110" when (temp>"0000000011111001") else  -- C temp> 25
            "1111110" when (temp>"0000000011010001") else  -- 0
            "1000111" when (temp<"0000000011010010") else  -- F temp<21
            "1001111";                                     -- E error
    dispder<="1011011" when (hume<"0000000100101100") else -- S hume<30%
            "1111110" when (hume<"0000001010111100") else  -- 0
            "0110111" when (hume>"0000001010111011") else  -- H hume>70%
            "1001111";                                     -- E error
end arch;
```

Figura 3.3.16. Código comprobación temperatura y humedad

Este bloque la primera función que realiza es una vez terminada la transmisión de información lee del registro y agrupa los datos según pertenezcan a la temperatura o humedad.

Posteriormente compara estos valores con unos rangos predefinidos para decidir qué valor muestra por pantalla.

Los valores tomados son una estimación de valores dentro de los cuales se consideran que las condiciones interiores de un local son aceptables para la estancia un tiempo considerable en su interior. Estos valores son entre 21 y 25 ºC y entre 30 y 70 % de humedad relativa. Estos valores se pueden alterar en función de las características del local en cuestión.

La forma en que el sensor manda la información de ambos valores es en un vector de 16 bits que contienen el valor hasta la décima. Este valor lo envía multiplicado por 10 por lo que será necesario tenerlo en cuenta a la hora de marcar los límites.

Además para valores negativos de la temperatura el bit más significativo de la temperatura toma el valor de 1 mientras que el resto del tiempo toma el valor de 0.

La forma en la que el dispositivo advierte del valor es mediante letras en los displays utilizando el siguiente código:

- C en el display de la izquierda indica que hace más calor de lo recomendable.
- F en el display de la izquierda indica que hace más frio de lo recomendable.
- H en el display derecho indica que el ambiente está muy húmedo.
- S en el display derecho indica que el ambiente está muy seco.
- O en los displays indica que el ambiente está en “Ok” condiciones.
- E en los displays indica un error en la medida obtenida.

### 3.3.6 Asignación de pines

En la siguiente imagen se muestran los puertos y como se han configurado las diferentes entradas y salidas del dispositivo.

| Name | Group By | Pin | BANK | BANK_VCC | VREF | IO_TYPE | PULLMODE |
|---|---|---|---|---|---|---|---|
| All Ports | N/A | N/A | N/A | N/A | N/A | | |
| Input | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Clock | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| RelojExt | N/A | 27(27) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| Output | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| RelojExtHab | N/A | 32(32) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derA | N/A | 21(21) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derB | N/A | 22(22) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derC | N/A | 11(11) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derD | N/A | 10(10) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derE | N/A | 13(13) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derF | N/A | 12(12) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| derG | N/A | 14(14) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqA | N/A | 23(23) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqB | N/A | 25(25) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqC | N/A | 33(33) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqD | N/A | 34(34) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqE | N/A | 35(35) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqF | N/A | 26(26) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| izqG | N/A | 24(24) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | DOWN(DOWN) |
| Bidir | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| sensor | N/A | 1(1) | 3(3) | Auto | N/A | LVCMOS33(LVCMOS33) | NONE(NONE) |

Figura 3.3.17. Asignación de pines

# 4. RECURSOS EMPLEADOS

En este capítulo se procede a realizar un estudio de los recursos utilizados por la FPGA una vez los proyectos han sido insertados en el dispositivo.

A continuación se muestran las partes principales del último diseño, debido a que es el que más recursos requiere y que es el principal que ha sido desarrollado en mayor medida.

## 4.1. Map Report

Este apartado presenta las principales características del proyecto, como puede ser el modelo de la FPGA la versión del software o la ocupación del diseño.

```
Design Information

Command line:   map -a MachXO2 -p LCMXO2-1200ZE -t TQFP144 -s 1 -oc Commercial
     intento16_impl1.ngd -o intento16_impl1_map.ncd -pr intento16_impl1.prf -mp
     intento16_impl1.mrp -lpf C:/RUBEN/UNI/TFM/INTENtOS
     VHDL/intento16(relojes)/impl1/intento16_impl1_synplify.lpf -lpf
     C:/RUBEN/UNI/TFM/INTENtOS VHDL/intento16(relojes)/intento16.lpf -c 0 -gui
     -msgset C:/RUBEN/UNI/TFM/INTENtOS VHDL/intento16(relojes)/promote.xml
Target Vendor:  LATTICE
Target Device:  LCMXO2-1200ZETQFP144
Target Performance:   1
Mapper:  xo2c00,  version:  Diamond (64-bit) 3.10.2.115
Mapped on:  04/29/19  12:28:12
```

Figura 4.1.1. Informe Map Report Design Information

En el apartado mostrado en la figura 4.1.1 se pueden observar las características generales como el fabricante y modelo de la FPGA o el software utilizado para crear el proyecto entre otros.

```
Design Summary
   Number of registers:    105 out of  1604 (7%)
      PFU registers:          105 out of  1280 (8%)
      PIO registers:            0 out of   324 (0%)
   Number of SLICEs:        54 out of   640 (8%)
      SLICEs as Logic/ROM:     54 out of   640 (8%)
      SLICEs as RAM:            0 out of   480 (0%)
      SLICEs as Carry:         13 out of   640 (2%)
   Number of LUT4s:        101 out of  1280 (8%)
      Number used as logic LUTs:         75
      Number used as distributed RAM:     0
      Number used as ripple logic:       26
      Number used as shift registers:     0
   Number of PIO sites used: 18 + 4(JTAG) out of 108 (20%)
   Number of block RAMs:  0 out of 7 (0%)
   Number of GSRs:  1 out of 1 (100%)
   EFB used :       No
   JTAG used :      No
   Readback used :  No
   Oscillator used :  Yes
   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:  1 out of 1 (100%)
   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 N_7: 12 loads, 12 rising, 0 falling (Driver: I4 )
     Net N_8: 1 loads, 1 rising, 0 falling (Driver: PIO RelojExt )

     Net N_25: 15 loads, 15 rising, 0 falling (Driver: I13/current_state[8] )
     Net reloj20mms: 27 loads, 7 rising, 20 falling (Driver: I9/PLLInst_0 )
   Number of Clock Enables:  1
     Net I13.N_107_1_i: 20 loads, 20 LSLICEs
   Number of local set/reset loads for net N_26 merged into GSR:  40
   Number of LSRs:  1
     Net N_26_i: 7 loads, 7 LSLICEs
   Number of nets driven by tri-state buffers:  0
   Top 10 highest fanout non-clock nets:
     Net I1/func_and_inet_4: 23 loads
     Net I1/func_and_inet_5: 23 loads
     Net I1/func_and_inet_6: 23 loads
     Net I13.N_107_1_i: 20 loads
     Net N_28: 12 loads
     Net N_26_i: 7 loads
     Net VCC: 6 loads
     Net I13.current_state[9]: 4 loads
     Net I32/dispizq_1_0_a2_0_o4[3]: 4 loads
     Net dispder[3]: 3 loads



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

Figura 4.1.2. Informe Map Report Design Sumary

En la figura 4.1.2 se observa los diferentes recursos empleados en el diseño, a continuación se señalan algunos.

- 7% de registros empleados.
- 8% de SLICEs empleados
- 8% de LUT4s usados.
- 1/1 PLL utilizado.
- 20% de PIO sites empleados

A la vista de esta información sería posible utilizar un componente con menos potencia lógica siempre y cuando mantenga los mínimos utilizados, incluyendo al menos un PLL.

## 4.2 Signal / PAD

A continuación se muestra los resultados más relevantes del informe que lleva tiene como título el de este subapartado, a destacar la configuración de los pines de entradas y salidas.

| Port Name | Pin/Bank | Buffer Type | Site | PG Enable | BC Enable | Properties |
|---|---|---|---|---|---|---|
| RelojExt | 27/3 | LVCMOS33_IN | PL9A | | | PULL:DOWN CLAMP:ON HYSTERESIS:SMALL |
| RelojExtHab | 32/3 | LVCMOS33_OUT | PL10A | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derA | 21/3 | LVCMOS33_OUT | PL5C | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derB | 22/3 | LVCMOS33_OUT | PL5D | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derC | 11/3 | LVCMOS33_OUT | PL4A | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derD | 10/3 | LVCMOS33_OUT | PL3D | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derE | 13/3 | LVCMOS33_OUT | PL4C | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derF | 12/3 | LVCMOS33_OUT | PL4B | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derG | 14/3 | LVCMOS33_OUT | PL4D | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| derP | 9/3 | LVCMOS33_OUT | PL3C | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqA | 23/3 | LVCMOS33_OUT | PL8A | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqB | 25/3 | LVCMOS33_OUT | PL8C | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqC | 33/3 | LVCMOS33_OUT | PL10B | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqD | 34/3 | LVCMOS33_OUT | PL10C | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqE | 35/3 | LVCMOS33_OUT | PL10D | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqF | 26/3 | LVCMOS33_OUT | PL8D | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| izqG | 24/3 | LVCMOS33_OUT | PL8B | | | DRIVE:8mA PULL:DOWN SLEW:SLOW |
| sensor | 1/3 | LVCMOS33_BIDI | PL2A | | | DRIVE:8mA HYSTERESIS:SMALL OPENDRAIN:ON SLEW:SLOW |

Figura 4.2.1 Informe Signal / PAD puertos

En este el diseño todos los pines traban a 3,3 V. Los pines son los siguientes:

- 1 pin de entrada, que consiste en el reloj externo incluido en la placa.
- 1 pin de salida que habilita el reloj externo.
- 14 pines de salidas conectados a los displays de la placa superior.
- 1 pin bidireccional que se comunica con el sensor, cabe resaltar que no consta de pull mode y que tiene el opendrain activado.

Todos los pines excepto el puerto bidireccional están conectados a nivel bajo por defecto.

# 5. ESTUDIO ECONÓMICO

En este apartado se describen los recursos económicos necesarios para la realización del último diseño. Solo se realiza el de este diseño debido a que es el único que tiene sentido una implantación en grandes cantidades y no como es un prototipo sin uso más allá del académico.

Primero se consideraran los costes de la realización del prototipo y posteriormente el coste incurrido al realizar una producción elevada, disminuyendo por tanto el coste individual del dispositivo.

En cuanto al desarrollo del prototipo los costes incurridos en materiales se muestran en la figura 5.1.1.

| Dispositivo | Coste unitario (€) | Unidades | Precio total (€) |
|---|---|---|---|
| MachXO2 1200-ZE | 26,20 | 1 | 26,20 |
| PCB auxiliares | 5,00 | 2 | 10,00 |
| DHT22 | 2,25 | 1 | 2,25 |
| Analizador lógico | 9,00 | 1 | 9,00 |
| Total | | | 47,45 |

Figura 5.1.1. Costes materiales prototipo

Debido a la descatalogación del producto MachXO2 1200-ZE se ha considerado el precio de un dispositivo similar de la misma familia.

No se ha considerado el gasto en herramientas empleadas debido a que eran de uso libre o con acceso a licencia académica. En el caso del transporte de los bienes y su amortización se ha considerado un sobrecoste del 5% en ambos casos.

| Motivo | Coste (€) |
|---|---|
| Materiales | 47,45 |
| Transporte | 2,37 |
| Amortización | 2,37 |
| Total | 52,20 |

Figura 5.1.2. Costes totales prototipo

También es necesario considerar la mano de obra necesaria para este proyecto, en este caso solo existe la del ingeniero que lo ha realizado. Se estima en unas 225 horas el tiempo de trabajo necesario para el desarrollo del diseño.

En cuanto a los costes de producción en grandes cantidades para la venta de este producto, siendo el principal los materiales, se muestran en la figura 5.1.3.

| Dispositivo | Coste unitario (€) | Unidades | Precio total (€) |
|---|---|---|---|
| MachXO2 1200-ZE | 3,15 | 1 | 3,15 |
| PCB auxiliar | 0,95 | 2 | 1,90 |
| DHT22 | 0,90 | 1 | 0,90 |
| Total | | | 5,95 |

Figura 5.1.3. Costes unitarios producción

Por tanto se consigue un producto completo por 5,95 € cada unidad.

# 6. CONCLUSIONES

Se ha estudiado la implantación en dispositivos tipo FPGA de diseños capaces de la detección de distintos tipos de secuencias, desde secuencias binarias sencillas, secuencias complejas, de gran número de bits, y proporcionadas por sensores mediante protocolos específicos. Para alcanzar este objetivo se han tenido que cumplir una serie de objetivos parciales, como son:

- Se han desarrollados diseños de forma jerárquica mediante descripciones VHDL, de forma que una persona ajena a la realización de este proyecto pueda entender, modificar y acceder a la información incluida en este trabajo.
- Se han comprobado diferentes alternativas en cuanto dispositivos lógicos, analizando varios tipos de elementos y dentro de las FPGAs diferentes fabricantes para encontrar un elemento adecuado a las necesidades planteadas.
- Se ha realizado un estudio de las diferentes características que tiene el sensor utilizado, teniendo en cuenta el modo de comunicación del mismo y cómo afecta esto al diseño realizado.
- Se han realizado simulaciones de diferentes bloques para comprobar su funcionamiento y ajustarlo en caso de ser necesario antes de realiza la implantación final.
- Mediante el uso de máquinas de estados finitos se ha simplificado la programación en VHDL, al apoyarse en herramientas externas que permitan generar el código correspondiente a estas.
- Una vez realizada la implantación física, se ha utilizado herramientas que permitan tomar el valor de diferentes señales para garantizar el correcto funcionamiento de los diseños realizados.
- Se ha realizado un estudio técnico de los recursos empleados para conocer en que nivel se están utilizando los recursos incluidos en el dispositivo utilizado, y si sería posible realizar los diseños en otro dispositivo de diferentes características.

En cuanto a las posibles líneas futuras de desarrollo a mejoras o adicciones al proyecto actual cabe destacar las siguientes.

Realizar una optimización de la programación tanto a nivel de VHDL como a nivel de los esquemas realizados, obteniendo los mismos resultados con menor poder computacional.

Incorporar el diseño en sistema de control complejos, que podrían incorporar microprocesadores, dentro del mismo dispositivo como podrían ser:

- Instalar una conexión entre el dispositivo y un equipo de acondicionamiento de interiores, para así ponerlo en marcha cuando sea necesario y disminuir el posible consumo energético del local.

- Adaptar la configuración para poder tener un control manual sobre los rangos de confort configurados sin que sea necesario cambiarlos en el código, requiriendo por tanto realizar otra vez la implantación.
- Añadir configuraciones para diferentes tipos de locales según sea el público afluente y la actividad realizada en el interior.
- Añadir sensores de parámetros como la presión para determinar completamente las condiciones de un local.
- Incluir sensores de partículas o contaminantes que permitan no tener solo un control sobre la climatización del aire de un local, sino también sobe la calidad del aire en el interior del mismo.

# 7. BIBLIOGRAFÍA

https://en.wikipedia.org/wiki/Microcontroller

Consultada por última vez: 1/5/2019

https://www.sciencedirect.com/topics/engineering/application-specific-integrated-circuits

Consultada por última vez: 1/5/2019

https://www.xilinx.com

Consultada por última vez: 1/5/2019

http://www.latticesemi.com

Consultada por última vez: 1/5/2019

https://www.latticesemi.com/en/Products/DevelopmentBoardsAndKits/MachXO2BreakoutBoard

Consultada por última vez: 25/4/2019

https://akizukidenshi.com/download/ds/aosong/AM2302.pdf

Consultada por última vez: 25/4/2019

http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/i2c-bus-protocol.html

Consultada por última vez: 17/4/2019

https://www.ics.uci.edu/~jmoorkan/vhdlref/for_loop.html

Consultada por última vez: 20/3/2019

# ANEXO 1. DOCUMENTACIÓN MACHXO2

# MachXO2 Breakout Board Evaluation Kit

## User's Guide

January 2014 Revision: EB68_02.2

## Introduction

Thank you for choosing the Lattice Semiconductor MachXO2™ Breakout Board Evaluation Kit!

This user's guide describes how to start using the MachXO2 Breakout Board, an easy-to-use platform for evaluating and designing with the MachXO2 ultra-low density FPGA. Along with the board and accessories, this kit includes a pre-loaded demonstration design. You may also reprogram the on-board MachXO2 device to review your own custom designs.

The MachXO2 Breakout Board currently features the MachXO2-7000HE device. A previous version of this board featured the MachXO2-1200ZE. The board design and features have not changed, and consequently, this document can be used as a guide for either version of the board. If you require a board featuring the MachXO2-1200ZE, Lattice recommends the MachXO2 Pico Development Kit.

See "Ordering Information" on page 16 for more information.

*Note: Static electricity can severely shorten the lifespan of electronic components. See the Storage and Handling section of this document for handling and storage tips.*

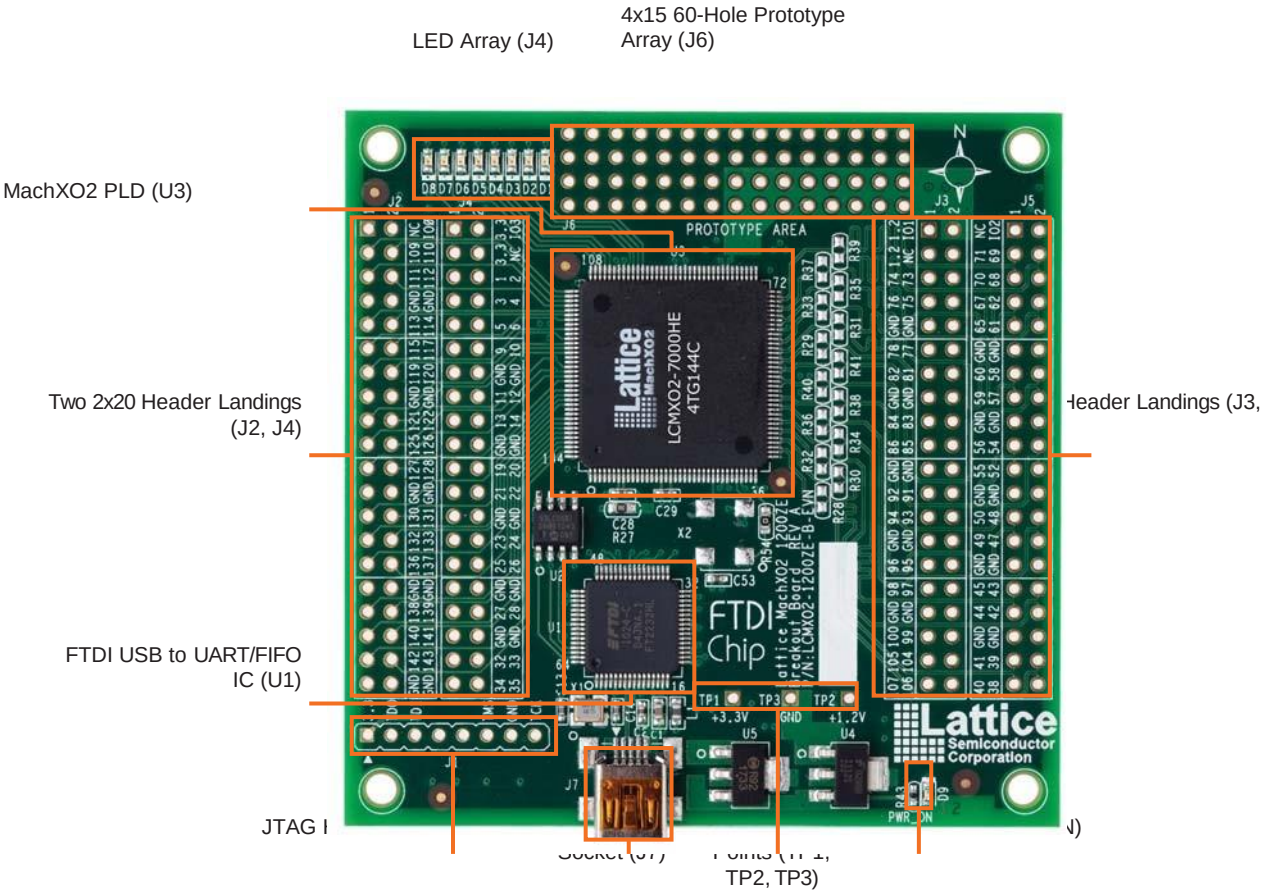
## Features

The MachXO2 Breakout Board Evaluation Kit includes:

- **MachXO2 Breakout Board** – The board is a 3" x 3" form factor that features the following on-board components and circuits:
  - MachXO2 FPGA – Current board version: LCMXO2-7000HE-4TG144C
    (Previous board version no longer available: LCMXO2-1200ZE-1TG144C)
  - USB mini-B connector for power and programming
  - Eight LEDs
  - 60-hole prototype area
  - Four 2x20 expansion header landings for general I/O, JTAG, and external power
  - 1x8 expansion header landing for JTAG
  - 3.3V and 1.2V supply rails
- **Pre-loaded Demo** – The kit includes a pre-loaded counter design that highlights use of the embedded MachXO2 oscillator and programmable I/Os configured for LED drive.
- **USB Connector Cable** – The board is powered from the USB mini-B socket when connected to a host PC. The USB channel also provides a programming interface to the MachXO2 JTAG port.
- **Lattice Breakout Board Evaluation Kits Web Page** – Visit www.latticesemi.com/breakoutboards for the latest documentation (including this guide) and drivers for the kit.

The content of this user's guide includes demo operation, programming instructions, top-level functional descriptions of the Breakout Board, descriptions of the on-board connectors, and a complete set of schematics.

***Figure 1. MachXO2 Breakout Board, Top Side***




## Storage and Handling

Static electricity can shorten the lifespan of electronic components. Please observe these tips to prevent damage that could occur from electro-static discharge:

- Use anti-static precautions such as operating on an anti-static mat and wearing an anti-static wrist-band.
- Store the evaluation board in the packaging provided.
- Touch a metal USB housing to equalize voltage potential between you and the board.

## Software Requirements

You should install the following software before you begin developing new designs for the Breakout board:

- Lattice Diamond® design software
- FTDI Chip USB hardware drivers (installed as an option within the Diamond installation program)

## MachXO2 Device

This board currently features the MachXO2-7000HE FPGA which offers embedded Flash technology for instant- on, non-volatile operation in a single chip. Numerous system functions are included, such as two PLLs and 256 Kbits of embedded RAM plus hardened implementations of $I^2C$, SPI, timer/counter, and user Flash memory. Flexi- ble, high performance I/Os support numerous single-ended and differential standards including LVDS, and also source synchronous interfaces to DDR/DDR2/LPDDR DRAM memory. The 144-pin TQFP package provides up to

114 user I/Os in a 20mm x 20mm form factor. Previous versions of this board featured the MachXO2-1200ZE PLD in the same package. This version of the board is no longer available. A complete description of this device can be found in the MachXO2 Family Data Sheet.

## Demonstration Design

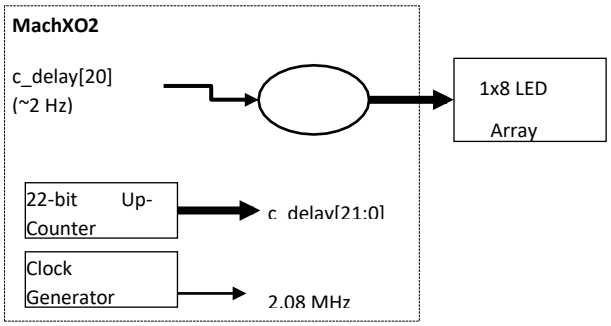
Lattice provides a simple, pre-programmed demo to illustrate basic operation of the MachXO2 device. The design integrates an up-counter with the on-chip oscillator.

*Note: You may obtain your Breakout Board after it has been reprogrammed. To restore the factory default demo and program it with other Lattice-supplied examples see the Download Demo Designs section of this document.*

### Run the Demonstration Design

Upon power-up, the preprogrammed demonstration design automatically loads and drives the LED array in an alternating pattern. The program shows a clock generator based on the MachXO2 on-chip oscillator. The counter module is clocked at the oscillator default frequency of 2.08MHz to illustrate how low speed timer functions can be implemented with a FPGA. The 22-bit up-counter further divides the clock to advance the LED display approxi- mately every 500ms. The resulting light pattern will appear as an alternating pair of lit LEDs per row.

***Figure 2. Demonstration Design Block Diagram***




*WARNING: Do not connect the Breakout Board to your PC before you follow the driver installation procedure of this section.*

Communication with the Breakout Board with a PC via the USB connection cable requires installation of the FTDI chip USB hardware drivers. Loading these drivers enables the computer to recognize and program the Breakout Board. Drivers can be loaded as part of the installation of Lattice Diamond design software or Diamond Programmer, or as a stand-alone package.

To load the FTDI Chip USB hardware drivers as part of the Lattice Diamond installation:

1. Select **Programmer Drivers** in the Product Options of Lattice Diamond Setup.
2. Select **FTDI Windows USB Driver** or **All Drivers** in the LSC Drivers Install/Uninstall dialog box.
3. Click **Finish** to install the USB driver.
4. After the driver installation is complete, connect the USB cable from a USB port on your PC to the board's USB mini-B socket (J2). After the connection is made, a green Power LED (D9) will light indicating the board is powered on.
5. The demonstration design will automatically load and drive the LED array in an alternating pattern.

To load the FTDI chip USB hardware drivers via the stand-alone package on a Windows system:

1. Browse to www.latticesemi.com/breakoutboards and download the FTDI Chip USB Hardware Drivers package.
2. Extract the FTDI chip USB Hardware driver package to your PC hard drive.
3. Connect the USB cable from a USB port on your PC to the board's USB mini-B socket (J7). After the connection is made, a green Power LED (D9) will light indicating the board is powered on.
4. If you are prompted, "Windows may connect to Windows Update" select **No, not this time** from available options and click **Next** to proceed with the installation. Choose the **Install from specific location (Advanced)** option and click **Next**.
5. Search for the best driver in these locations and click the **Browse** button to browse to the Windows driver folder created in the Download Windows USB Hardware Drivers section. Select the **CDM 2.04.06 WHQL Certified** folder and click **OK**.
6. Click **Next**. A screen will display as Windows copies the required driver files. Windows will display a message indicating that the installation was successful.
7. Click **Finish** to install the USB driver.
8. The demonstration design will automatically load and drive the LED array in an alternating pattern.

See the Troubleshooting section of this guide if the board does not function as expected.

## Download Demo Designs

The counter demo is preprogrammed into the Breakout Board, however over time it is likely your board will be modified. Lattice distributes source and programming files for demonstration designs compatible with the Breakout Board. The demo design for the 1200ZE version of the board is available on the web. Use the same design files for MachXO2-7000HE. Change the device in the Diamond Software tool and re-run the process flow to generate the JEDEC for MachXO2-7000HE. The description below references the 7000HE version.

To download demo designs:

1. Browse to the Lattice Breakout Board Evaluation Kits web page (www.latticesemi.com/breakoutboards) of the Lattice web site. Select **MachXO2 Breakout Board Demo Source** and save the file.
2. Extract the contents of **MachXO21200ZEBreakoutBoardDemoDesignSource.zip** to an accessible location on your hard drive.
3. Open the Project in the Diamond Design Software and change the device to MachXO2-7000HE-4TG144C.
4. Run the Process Flow and regenerate the JEDEC file.

Continue to Programming a Demo Design with Lattice Diamond Design Software.

## Programming a Demo Design with the Lattice Diamond Programmer

The demonstration design is pre-programmed into the MachXO2 Breakout Board by Lattice. If you have changed the design but now want to restore the Breakout Board to factory settings, use the procedure described below.

To program the MachXO2 device:

1. Install, license and run Lattice Diamond software. See www.latticesemi.com/latticediamond for download and licensing information.
2. Connect the USB cable to the host PC and the MachXO2 Breakout Board.

3. From Diamond, open the **Default_pattern_w_standby.ldf** project file.
4. Click the **Programmer** icon.
5. Click **Detect Cable**. The Programmer will detect the cable (Cable: USB2, Port: FTUSB-0). If the cable is not detected, see the Troubleshooting section.
6. Click the **Program** icon. When complete, **PASS** is displayed in the Status column.

# MachXO2 Breakout Board

This section describes the features of the MachXO2 Breakout Board in detail.

## Overview

The Breakout Board is a complete development platform for the MachXO2 FPGA. The board includes a prototyping area, a USB program/power port, an LED array, and header landings with electrical connections to most of the FPGA's programmable I/O, power, and JTAG pins. The board is powered by the PC's USB port or optionally with external power. You may create or modify the program files and reprogram the board using Lattice Diamond software.

***Figure 3. MachXO2 Breakout Board Block Diagram***

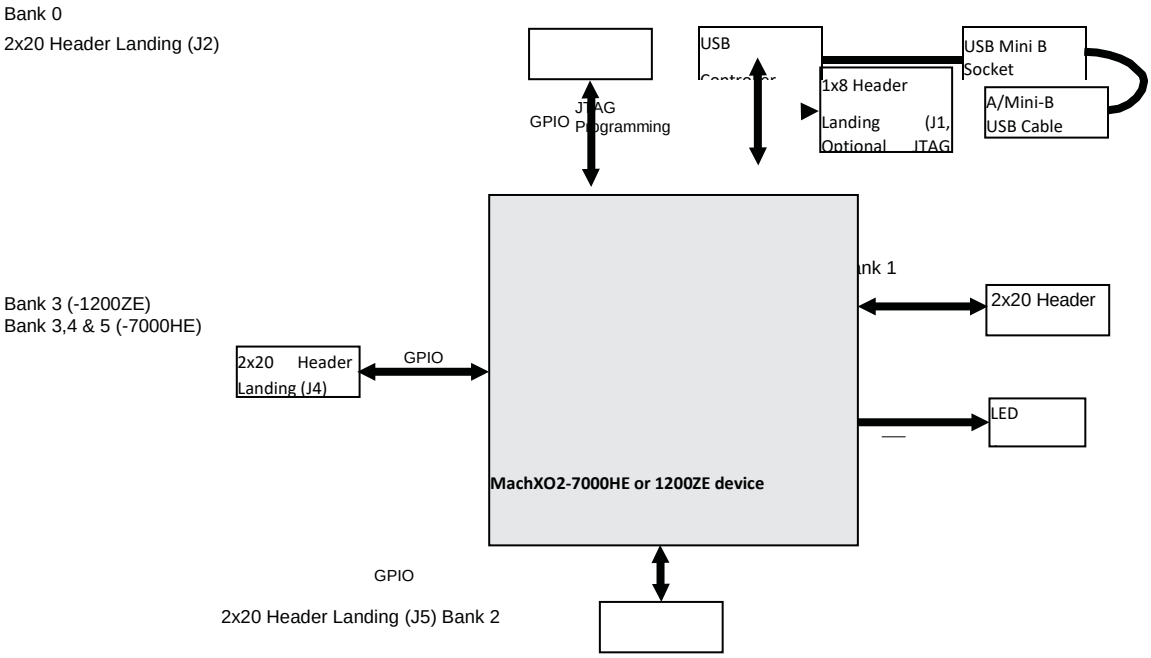


Table 1 describes the components on the board and the interfaces it supports.

***Table 1. Breakout Board Components and Interfaces***

| Component/Interface | Type | Schematic Reference | Description |
|---|---|---|---|
| **Circuits** | | | |
| USB Controller | Circuit | U2: FT2232H | USB-to-JTAG interface and dual USB UART/FIFO IC |
| USB Mini-B Socket | I/O | J7:USB_MINI_B | Programming and debug interface |
| **Components** | | | |
| LCMXO2 | FPGA | U3: LCMXO2-7000HE-4TG144C | 7000-LUT device packaged in a 20 x 20mm, 144-pin TQFP |
| **Interfaces** | | | |
| LED Array | Output | D8-D1 | Red LEDs |
| Four 2x20 Header Landings | I/O | J2: header_2x20<br>J3: header_2x20<br>J4: header_2x20<br>J5: header_2x20 | User-definable I/O |
| 1x8 Header Landing | I/O | J1: header_1x8 | Optional JTAG interface |
| 4x15 60-Hole Prototype Area | | | Prototype area 100mil centered holes. |
| Test Points | Power | TP1: +3.3V<br>TP2: +1.2V<br>TP3: GND | Power and ground reference points |

## Subsystems

This section describes the principle sub systems for the Breakout Board in alphabetical order.

### Clock Sources

All clocks for the counter demonstration designs originate from the MachXO2 on-chip oscillator. You may use an expansion header landing to drive a FPGA input with an external clock source.

### Expansion Header Landings

The expansion header landings provide access to user GPIOs, primary inputs, clocks, and VCCO pins of the MachXO2. The remaining pins serve as power supplies for external connections. Each landing is configured as one 2x20 100 mil.

***Table 2. Expansion Connector Reference***

| Item | Description |
|---|---|
| Reference Designators | J2, J3, J4, J5 |
| Part Number | header_2x20 |

*Table 3. Expansion Header Pin Information (J2)*

| Header Pin Number | -1200ZE Function | -7000HE Function | MachXO2 Pin |
|---|---|---|---|
| 1 | NC | NC | - |
| 2 | VCCIO0 | VCCIO0 | 118, 123, 135 |
| 3 | PT17D / DONE | PT36D / DONE | 109 |
| 4 | PT17C / INITn | PT36C / INITn | 110 |
| 5 | PT17B | PT36B | 111 |
| 6 | PT17A | PT36A | 112 |
| 7 | GND | GND | - |
| 8 | GND | GND | - |
| 9 | PT16D | PT33B | 113 |
| 10 | PT16C | PT33A | 114 |
| 11 | PT16B | PT28B | 115 |
| 12 | PT16A | PT28A | 117 |
| 13 | PT15D / PROGn | PT27D / PROGn | 119 |
| 14 | PT15C / JTAGen | PT27C / JTAGen | 120 |
| 15 | GND | GND | - |
| 16 | GND | GND | - |
| 17 | PT15B | PT25B | 121 |
| 18 | PT15A | PT25A | 122 |
| 19 | PT12D / SDA / PCLKC0_0 | PT22D / SDA / PCLKC0_0 | 125 |
| 20 | PT12C / SCL / PCLKT0_0 | PT22C / SCL / PCLKT0_0 | 126 |
| 21 | PT12B / PCLKC0_1 | PT18B / PCLKC0_1 | 127 |
| 22 | PT12A / PCLKT0_1 | PT18A / PCLKT0_1 | 128 |
| 23 | GND | GND | - |
| 24 | GND | GND | - |
| 25 | PT11D / TMS | PT17D / TMS | 130 |
| 26 | PT11C / TCK | PT17C / TCK | 131 |
| 27 | PT11B | PT15B | 132 |
| 28 | PT11A | PT15A | 133 |
| 29 | PT10D / TDI | PT14D / TDI | 136 |
| 30 | PT10C / TDO | PT14C / TDO | 137 |
| 31 | GND | GND | - |
| 32 | GND | GND | - |
| 33 | PT10B | PT11B | 138 |
| 34 | PT10A | PT11A | 139 |
| 35 | PT9D | PT10B | 140 |
| 36 | PT9C | PT10A | 141 |
| 37 | PT9B | PT9B | 142 |
| 38 | PT9A | PT9A | 143 |
| 39 | GND | GND | - |
| 40 | GND | GND | - |

*Table 4. Expansion Header Pin Information (J3)*

| Header Pin Number | -1200ZE Function | -7000HE Function | MachXO2 Pin |
|---|---|---|---|
| 1 | VCC_1.2V | VCC_1.2V | 36, 72, 108, 144 |
| 2 | VCCIO1 | VCCIO1 | 79, 88, 102 |
| 3 | VCC_1.2V | VCC_1.2V | 36, 72, 108, 144 |
| 4 | NC | NC | - |
| 5 | PR10C | PR24A | 74 |
| 6 | PR10D | PR24B | 73 |
| 7 | PR10A | PR23A | 76 |
| 8 | PR10B | PR23B | 75 |
| 9 | GND | GND | - |
| 10 | GND | GND | - |
| 11 | PR9C | PR21A | 78 |
| 12 | PR9D | PR21B | 77 |
| 13 | PR9A | PR18A | 82 |
| 14 | PR9B | PR18B | 81 |
| 15 | GND | GND | - |
| 16 | GND | GND | - |
| 17 | PR8C | PR17A | 84 |
| 18 | PR8D | PR17B | 83 |
| 19 | PR8A | PR16A | 86 |
| 20 | PR8B | PR16B | 85 |
| 21 | GND | GND | - |
| 22 | GND | GND | - |
| 23 | PR5C / PCLKT1_0 | PR12A / PCLKT1_0 | 92 |
| 24 | PR5D / PCLKC1_0 | PR12B / PCLKC1_0 | 91 |
| 25 | PR5A | PR11A | 94 |
| 26 | PR5B | PR11B | 93 |
| 27 | GND | GND | - |
| 28 | GND | GND | - |
| 29 | PR4C | PR9A | 96 |
| 30 | PR4D | PR9B | 95 |
| 31 | PR4A | PR7A | 98 |
| 32 | PR4B | PR7B | 97 |
| 33 | GND | GND | - |
| 34 | GND | GND | - |
| 35 | PR3A | PR5A | 100 |
| 36 | PR3B | PR5B | 99 |
| 37 | PR2C | PR3A | 105 |
| 38 | PR2D | PR3B | 104 |
| 39 | PR2A | PR2A | 107 |
| 40 | PR2B | PR2B | 106 |

*Table 5. Expansion Header Pin Information (J4)*

| Header Pin Number | -1200ZE Function | -7000HE Function | MachXO2 Pin |
|---|---|---|---|
| 1 | VCC_3.3V | VCC_3.3V | - |
| 2 | VCCIO3 | VCCIO3/4/5 | 30, 16, 7 |
| 3 | VCC_3.3V | VCC_3.3V | - |
| 4 | NC | NC | - |
| 5 | PL2A / L_GPLLT_FB | PL3A / L_GPLLT_FB | 1 |
| 6 | PL2B / L_GPPLC_FB | PL3B / L_GPPLC_FB | 2 |
| 7 | PL2C / L_GPLLT_IN | PL4A / L_GPLLT_IN | 3 |
| 8 | PL2D / L_GPLLC_IN | PL4B / L_GPLLC_IN | 4 |
| 9 | PL3A / PCLKT3_2 | PL6A / PCLKT5_0 | 5 |
| 10 | PL3B / PCLKC3_2 | PL6B / PCLKC5_0 | 6 |
| 11 | PL3C | PL8A | 9 |
| 12 | PL3D | PL8B | 10 |
| 13 | GND | GND | - |
| 14 | GND | GND | - |
| 15 | PL4A | PL9A | 11 |
| 16 | PL4B | PL9B | 12 |
| 17 | PL4C | PL10A | 13 |
| 18 | PL4D | PL10B | 14 |
| 19 | GND | GND | - |
| 20 | GND | GND | - |
| 21 | PL5A / PCLKT3_1 | PL12A / PCLKT4_0 | 19 |
| 22 | PL5B / PCLKC3_1 | PL12B / PCLKC4_0 | 20 |
| 23 | PL5C | PL15A | 21 |
| 24 | PL5D | PL15B | 22 |
| 25 | GND | GND | - |
| 26 | GND | GND | - |
| 27 | PL8A | PL17A | 23 |
| 28 | PL8B | PL17B | 24 |
| 29 | PL8C | PL19A | 25 |
| 30 | PL8D | PL19B | 26 |
| 31 | GND | GND | - |
| 32 | GND | GND | - |
| 33 | PL9A / PCLKT3_0 | PL22A / PCLKT3_0 | 27 |
| 34 | PL9B / PCLKC3_0 | PL22B / PCLKC3_0 | 28 |
| 35 | GND | GND | - |
| 36 | GND | GND | - |
| 37 | PL10A | PL24A | 32 |
| 38 | PL10B | PL24B | 33 |
| 39 | PL10C | PL25A | 34 |
| 40 | PL10D | PL25B | 35 |

*Table 6. Expansion Header Pin Information (J5)*

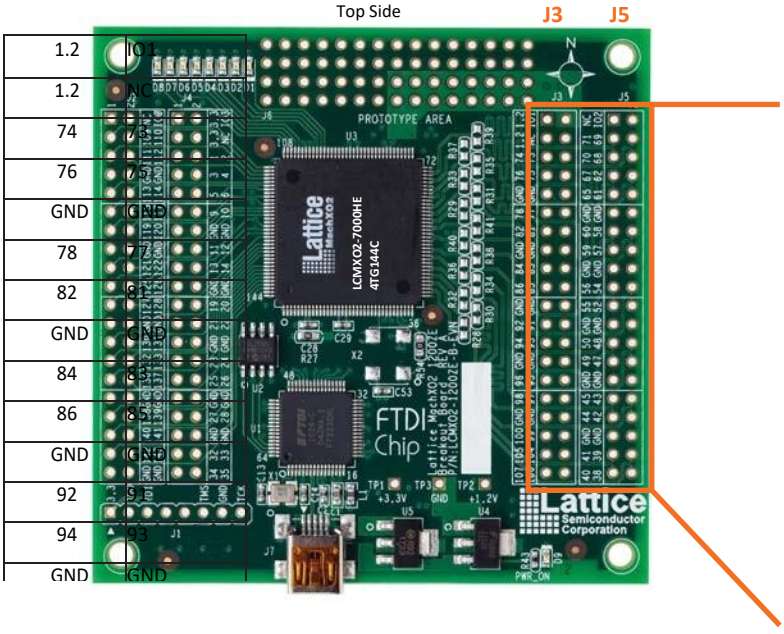
| Header Pin Number | -1200ZE Function | -7000HE Function | MachXO2 Pin |
|---|---|---|---|
| 1 | NC | NC | - |
| 2 | VCCIO2 | VCCIO2 | 37, 51, 66 |
| 3 | PB20D / SI / SISPI | PB38B / SI / SISPI | 71 |
| 4 | PB20B | PB37B | 69 |
| 5 | PB20C / SN | PB38A / SN | 70 |
| 6 | PB20A | PB37A | 68 |
| 7 | PB18D | PB35B | 67 |
| 8 | PB18B | PB31B | 62 |
| 9 | PB18C | PB35A | 65 |
| 10 | PB18A | PB31A | 61 |
| 11 | GND | GND | - |
| 12 | GND | GND | - |
| 13 | PB15D | PB29B | 60 |
| 14 | PB15B | PB26B | 58 |
| 15 | PB15C | PB29A | 59 |
| 16 | PB15A | PB26A | 57 |
| 17 | GND | GND | - |
| 18 | GND | GND | - |
| 19 | PB11B / PCLKC2_1 | PB23B / PCLKC2_1 | 56 |
| 20 | PB11D | PB18B | 54 |
| 21 | PB11A / PCLKT2_1 | PB23A / PCLKT2_1 | 55 |
| 22 | PB11C | PB18A | 52 |
| 23 | GND | GND | - |
| 24 | GND | GND | - |
| 25 | PB9B / PCLKC2_0 | PB16B / PCLKC2_0 | 50 |
| 26 | PB9D | PB13B | 48 |
| 27 | PB9A / PCLKT2_0 | PB16A / PCLKT2_0 | 49 |
| 28 | PB9C | PB13A | 47 |
| 29 | GND | GND | - |
| 30 | GND | GND | - |
| 31 | PB6D / S0 / SPISO | PB12B / S0 / SPISO | 45 |
| 32 | PB6B | PB9B | 43 |
| 33 | PB6C / MCLK / CCLK | PB12A / MCLK / CCLK | 44 |
| 34 | PB6A | PB9A | 42 |
| 35 | GND | GND | - |
| 36 | GND | GND | - |
| 37 | PB4D | PB6B | 41 |
| 38 | PB4B | PB4B | 39 |
| 39 | PB4C / CSSPIN | PB6A / CSSPIN | 40 |
| 40 | PB4A | PB4A | 38 |

*Figure 4. J2/J4 Header Landing Callout*

| J2 | |
|---|---|
| 1 | 2 |
| NC | IO0 |
| 109 | 110 |
| 111 | 112 |
| GND | GND |
| 113 | 114 |
| 115 | 117 |
| 119 | 120 |
| GND | GND |
| 121 | 122 |
| 125 | 126 |
| 127 | 128 |
| GND | GND |
| 130 | 131 |
| 132 | 133 |

| J4 | |
|---|---|
| 1 | 2 |
| 3.3 | IO3 |
| 3.3 | NC |
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
| 9 | 10 |
| GND | GND |
| 11 | 12 |
| 13 | 14 |
| GND | GND |
| 19 | 20 |
| 21 | 22 |
| GND | GND |
| 23 | 24 |

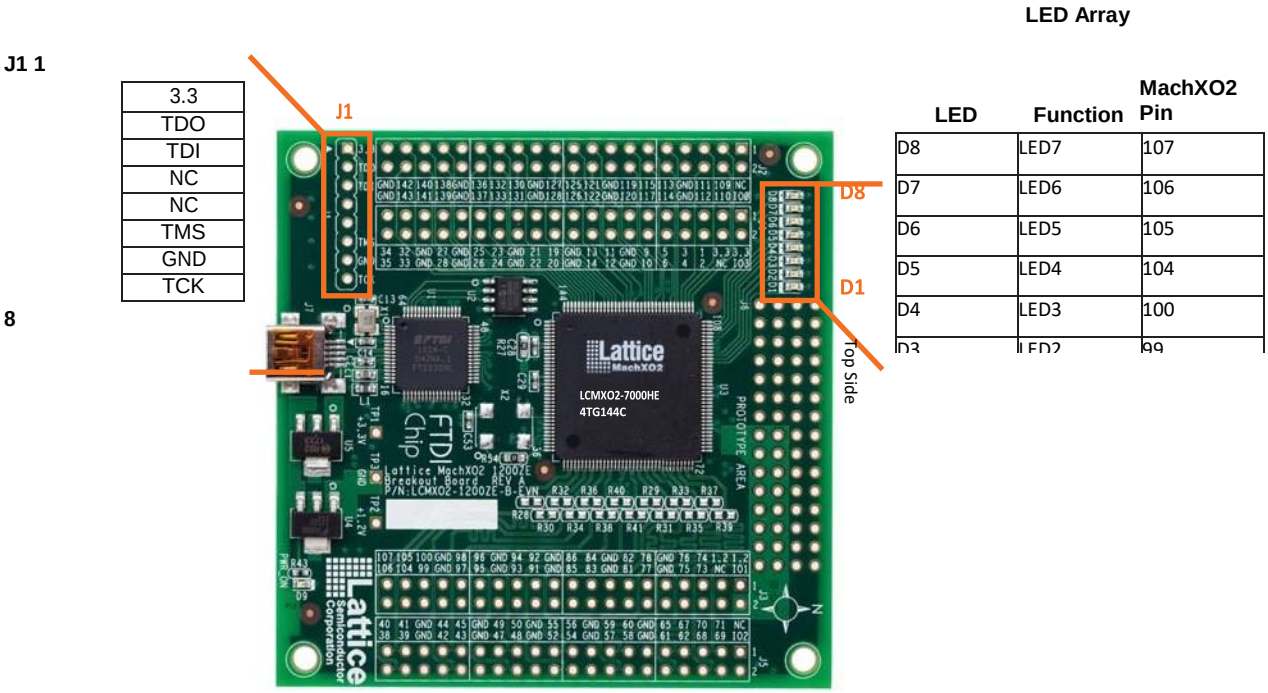

*Figure 5. J3/J5 Header Landing Callout*




| J3 | |
|---|---|
| 1 | 2 |
| 1.2 | IO1 |
| 1.2 | NC |
| 74 | 73 |
| 76 | 75 |
| GND | GND |
| 78 | 77 |
| 82 | 81 |
| GND | GND |
| 84 | 83 |
| 86 | 85 |
| GND | GND |
| 92 | 91 |
| 94 | 93 |
| GND | GND |

| J5 | |
|---|---|
| 1 | 2 |
| NC | IO2 |
| 71 | 69 |
| 70 | 68 |
| 67 | 62 |
| 65 | 61 |
| GND | GND |
| 60 | 58 |
| 59 | 57 |
| GND | GND |
| 56 | 54 |
| 55 | 52 |
| GND | GND |
| 50 | 48 |
| 49 | 47 |

*Figure 6. J1 Header Landing and LED Array Callout*




### MachXO2 FPGA

The MachXO2-7000HE-4TG144C is a 144-pin TQFP package FPGA device which provides up to 114 usable I/Os in a 20 x 20mm package. 108 I/Os are accessible from the breakout board headers.

*Table 7. MachXO FPGA Interface Reference*

| Item | Description |
|---|---|
| Reference Designators | U3 |
| Part Number | LCMXO2-7000HE-4TG144C |
| Manufacturer | Lattice Semiconductor |
| Web Site | www.latticesemi.com |

### JTAG Interface Circuits

For power and programming an FTDI USB UART/FIFO IC converter provides a communication interface between a PC host and the JTAG programming chain of the Breakout Board. The USB 5V supply is also used as a source for the 3.3V supply rail. A USB mini-B socket is provided for the USB connector cable.

*Table 8. JTAG Interface Reference*

| Item | Description |
|---|---|
| Reference Designators | U1 |
| Part Number | FT2232HL |
| Manufacturer | Future Technology Devices International (FTDI) |
| Web Site | www.ftdichip.com |

*Table 9. JTAG Programming Pin Information*

| Description | MachXO2 Pin |
|---|---|
| Test Data Output | 137:TDO |
| Test Data Input | 136:TDI |
| Test Mode Select | 130:TMS |
| Test Clock | 131:TCK |

**LEDs**
A green LED (D9) is used to indicate USB 5V power. Eight red LEDs are driven by I/O pins of the MachXO2 device.

*Table 10. Power and User LEDs Reference*

| Item | Description |
|---|---|
| Reference Designators | Red LEDs (D1, D2, D3, D4, D5, D6, D7, D8)<br>Green LEDs (D9) |
| Part Number | LTST-C190KRKT (D1-D8)<br>LTST-C190KGKT (D9) |
| Manufacturer | Lite-On It Corporation |
| Web Site | www.liteonit.com |

**Power Supply**
3.3V and 1.2V power supply rails are converted from the USB 5V interface when the board is connected to a host PC.

**Test Points**
In order to check the various voltage levels used, test points are provided:

- TP1: +3.3V
- TP2: +1.2V
- TP3: GND

**USB Programming and Debug Interface**
The USB mini-B socket of the Breakout Board serves as the programming and debug interface.

JTAG Programming: For JTAG programming, a preprogrammed USB PHY peripheral controller is provided on the Breakout Board to serve as the programming interface to the MachXO2 FPGA.

Programming requires the Lattice Diamond or ispVM System software.

*Table 11. USB Interface Reference*

| Item | Description |
|---|---|
| Reference Designators | U1 |
| Part Number | FT2232HL |
| Manufacturer | Future Technology Devices International (FTDI) |
| Web Site | www.ftdichip.com |

## Board Modifications

This section describes modifications to the board to change or add functionality.

### Bypassing the USB Programming Interface

The USB programming interface circuit (USB Programming and Debug Interface section) may be optionally bypassed by removing the 0 ohm resistors: R5, R6, R7, R8 (See Appendix A. Schematics, Sheet 2 of 5). Header landing J1 provides JTAG signal access for jumper wires or a 1x8 pin header.

### Applying External Power

The Breakout Board is powered by the circuit of Schematic Sheet 5 of 5 based on the 5V USB power source. You may disconnect this power source by removing the 0 ohm resistors: R42 (VCC_1.2V) and R44 (VCC_3.3V). Power connections are available from the expansion header landings, J3 (+1.2V, pins 1 and 3, schematic sheet 3 of 5) and J4 (+3.3V, pins 1 and 3, schematic sheet 4 of 5).

### Measuring Bank and Core Power

In addition to the expansion headers, test points (TP1, TP2) provide access to power supplies of the MachXO2 FPGA. Inline 1 ohm resistors: R24 (VCCIO0, +3.3V, Bank 0), R25 (VCCIO1, +3.3V, Bank 1), R26 (VCCIO2, +3.3V, Bank 2), R27 (VCCIO3, +3.3V, Bank 3), R56 (VCC core, +1.2V) can be used to measure current for the power supplies.

## Mechanical Specifications

Dimensions: 3 in. [L] x 3 in. [W] x 1/2 in. [H]

## Environmental Requirements

The evaluation board must be stored between -40° C and 100° C. The recommended operating temperature is between 0° C and 90° C.

The board can be damaged without proper anti-static handling.

# Glossary

**FPGA**: Field Programmable Gate Array

**DIP**: Dual in-line package **LED**:

Light Emitting Diode. **LUT**:

Look Up Table

**PCB**: Printed Circuit Board

**RoHS**: Restriction of Hazardous Substances Directive

**USB**: Universal Serial Bus

**WDT**: Watchdog Timer

# Troubleshooting

Use the tips in this section to diagnose problems with the Breakout Board.

### LEDs Do Not Flash

If power is applied but the board does not flash according to the preprogrammed counter demonstration then it is likely the board has been reprogrammed with a new design. Follow the directions in the Demonstration Design section to restore the factory default.

**USB Cable Not Detected**

If Lattice Diamond Programmer or ispVM System does not recognize the USB cable after installing the Lattice USB port drivers and rebooting, the incorrect USB driver may have been installed. This usually occurs if you attach the board to your PC prior to installing the Lattice-supplied USB driver.

**To access the *Troubleshooting the USB Driver Installation Guide*:**

For Diamond software and standalone Diamond Programmer:

1. Start Diamond or Diamond Programmer and choose **Help**.
2. Search for **USB driver** or **Troubleshooting**, then select the **Troubleshooting the USB Driver** topic.
3. Follow the directions to install the Lattice USB driver.

For ispVM:

1. Start ispVM System and choose **Options > Cable and I/O Port Setup**.
   The Cable and I/O Port Setup Dialog appears.
2. Click the **Troubleshooting the USB Driver Installation Guide** link.

The *Troubleshooting the USB Driver Installation Guide* document appears in your system's PDF file reader.

3. Follow the directions to install the Lattice USB driver.

**Determine the Source of a Pre-Programmed Device**

If the Breakout Board has been reprogrammed, the original demo design can be restored. To restore the board to the factory default, see the Download Demo Designs section for details on downloading and reprogramming the device.

# Ordering Information

| Description | Ordering Part Number | China RoHS Environment-Friendly Use Period (EFUP) |
|---|---|---|
| MachXO2-7000HE Breakout Board Evaluation Kit | LCMXO2-7000HE-B-EVN |  |
| MachXO2 Breakout Board Evaluation Kit | LCMXO2-1200ZE-B-EVN[1] |  |

1.For reference only. This version of the board is no longer available for sale.

# Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America) e-mail:
techsupport@latticesemi.com
Internet: www.latticesemi.com

## Revision History

| Date | Version | Change Summary |
|---|---|---|
| December 2011 | 01.0 | Initial release. |
| January 2012 | 01.1 | Figure "MachXO2-1200ZE Breakout Board, Top Side" updated with revision B board photo. |
| December 2012 | 01.2 | Updated document to describe new version of the board featuring the MachXO2-7000HE. Indicated that the MachXO2-1200ZE version of the board is no longer available. |
| February 2013 | 02.0 | Updated Tables 3-6 to include -7000HE information. Added -7000HE notes to Figure 3 and Appendix A. |
| September 2013 | 02.1 | Updated procedure in Programming a Demo Design with the Lattice Diamond Programmer section. |
| | | Added information to the procedure on loading the FTDI chip USB hardware drivers via the standalone package: |
| | | Updated description of Reference Designators in the Power and User LEDs Reference table. |
| January 2014 | 02.2 | Updated description and procedure for downloading demo designs in Download Demo Designs section. |
| | | Updated project file name in Programming a Demo Design with the Lattice Diamond Programmer section. |

## Appendix A. Schematics

Note: The schematics are drawn using the MachXO2-1200ZE device. Please consult Tables 3 through 6 for -1200 and -7000HE pin name and bank synonyms. Pin numbers are correct for either device.

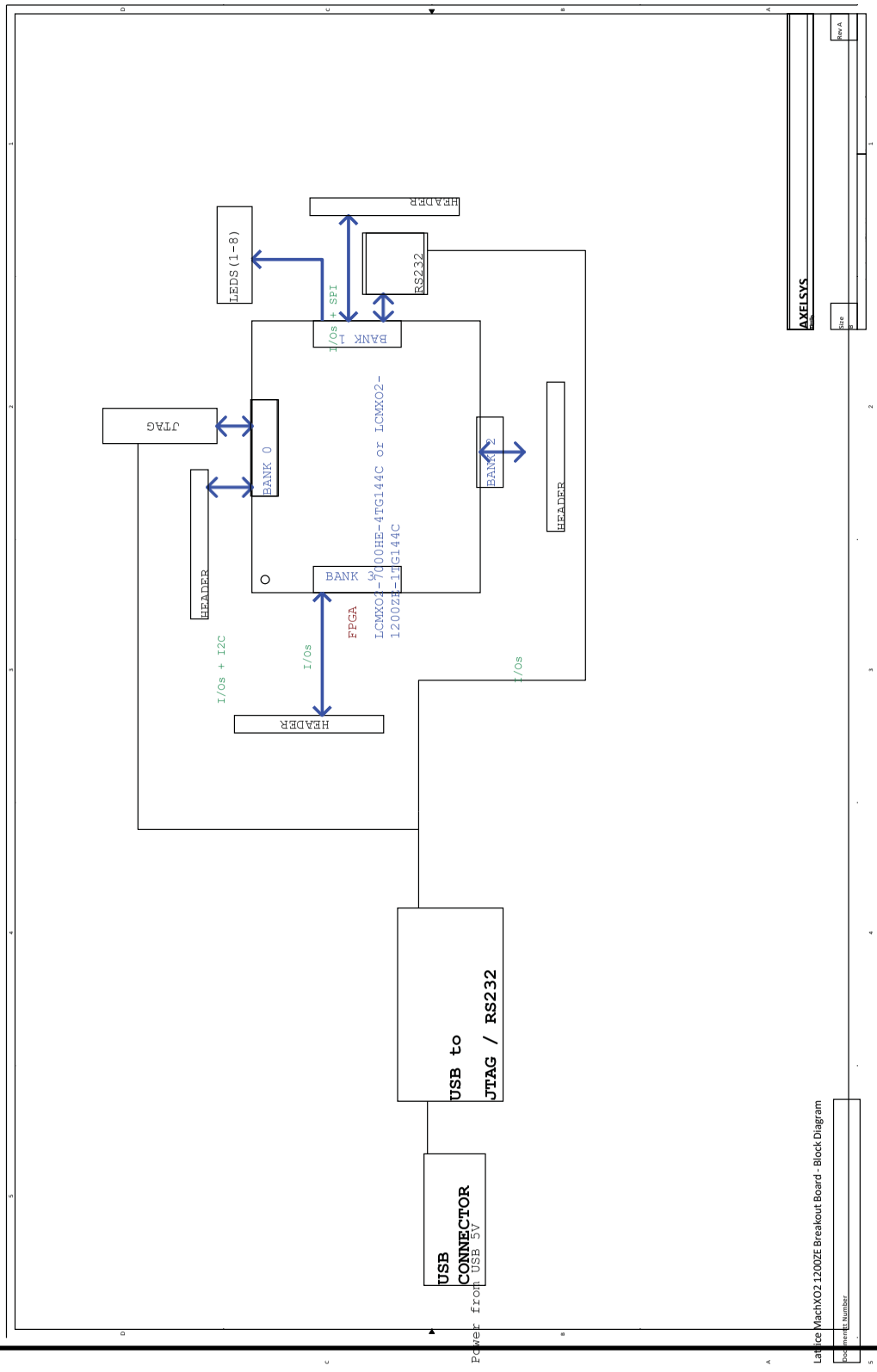


*Figure 8. USB Interface to JTAG*

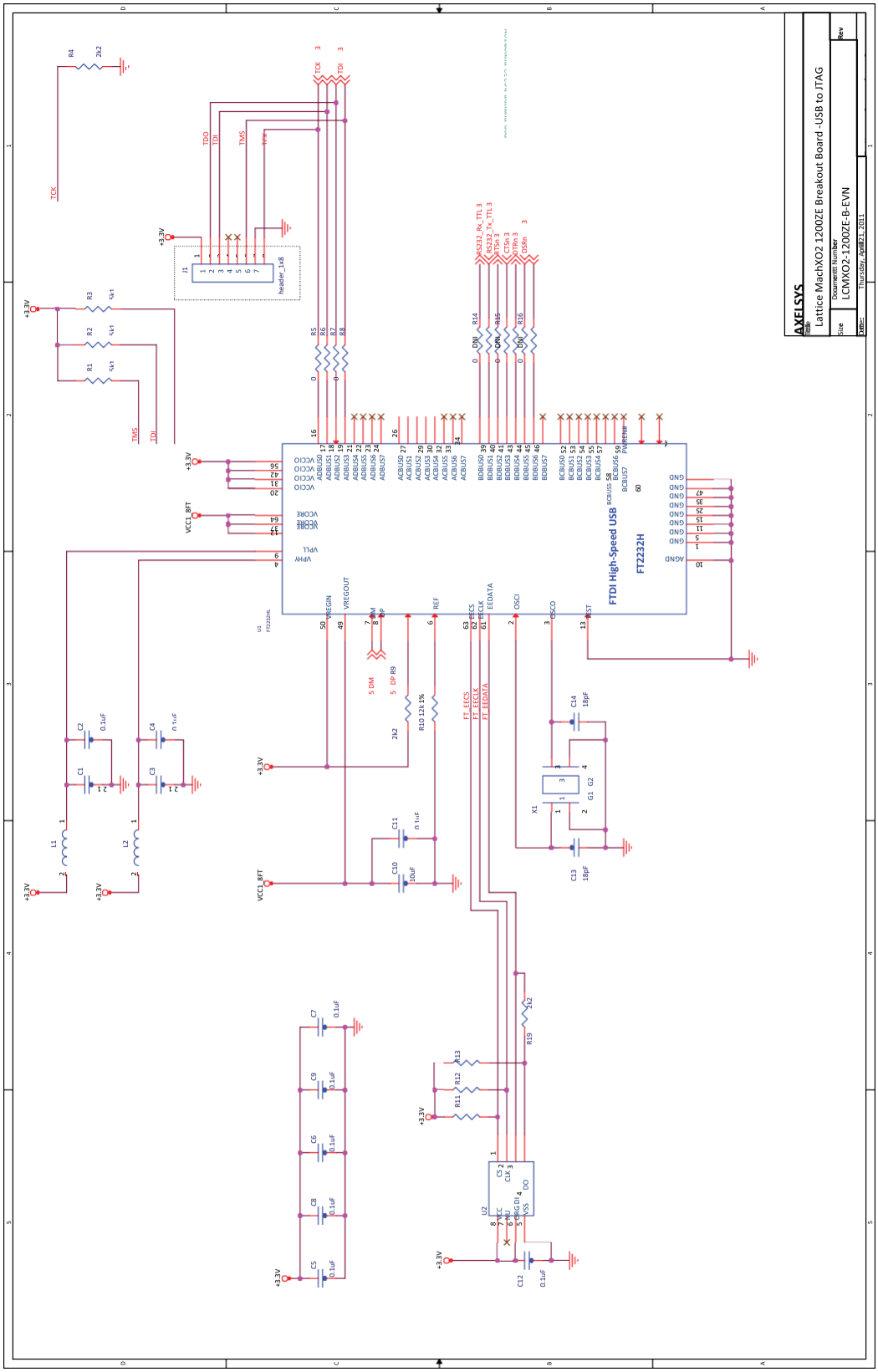


*Figure 9. FPGA*

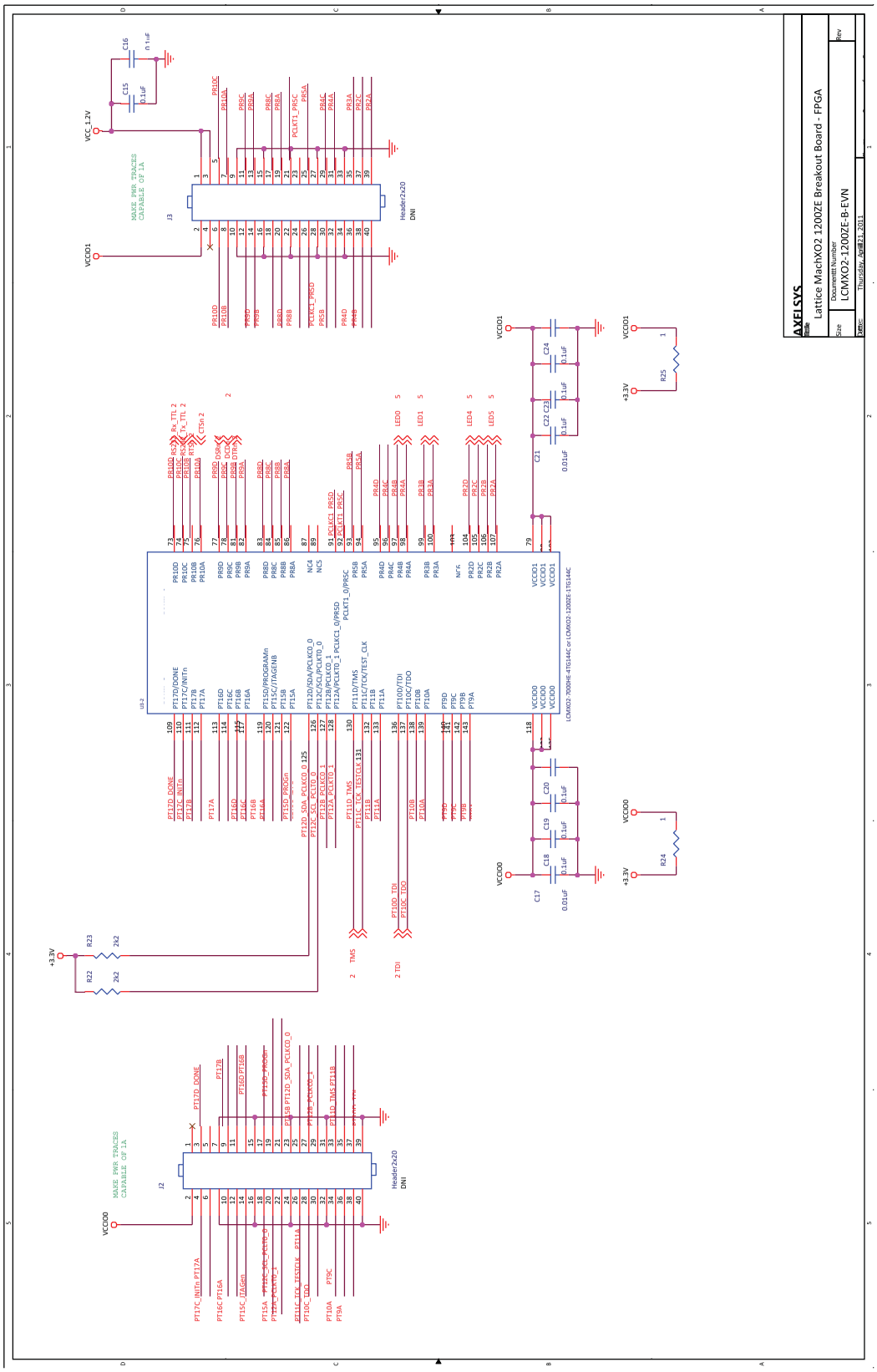


*Figure 10. FPGA*

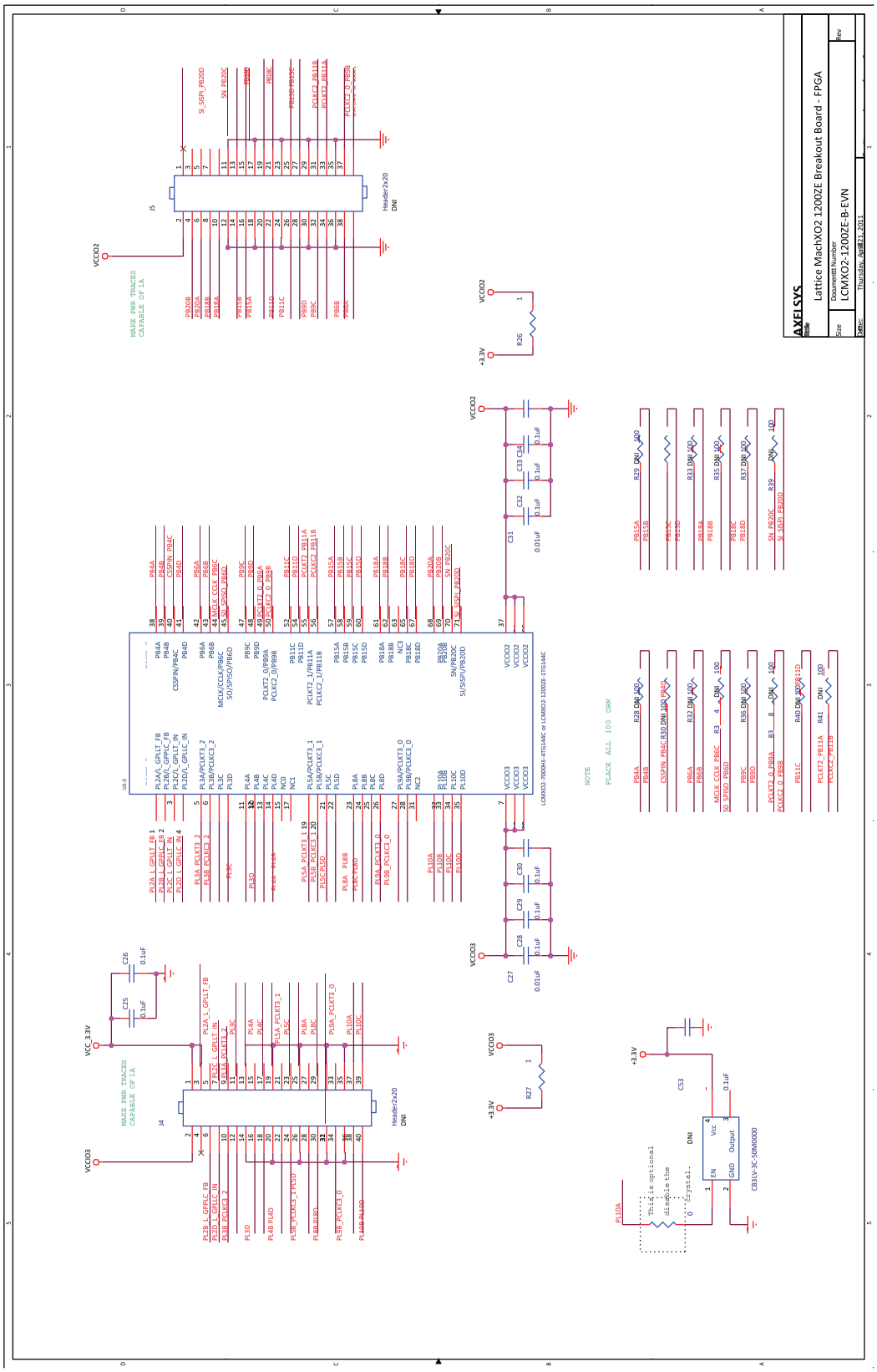
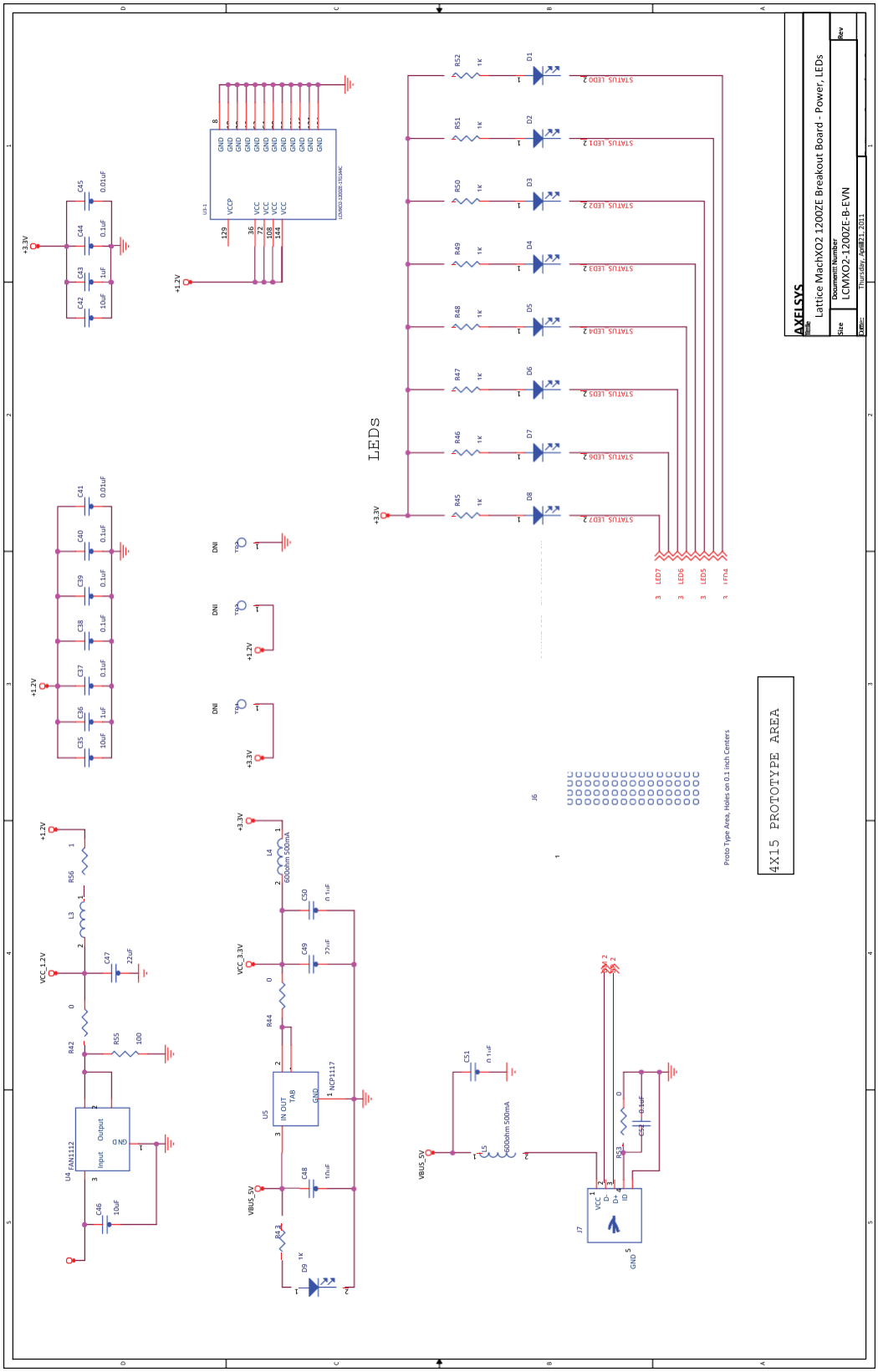


*Figure 11. Power LEDs*



# Appendix B. Bill of Materials

***Table 12. MachXO2 Breakout Board Bill of Materials***

| Item | Quantity | Reference | Manufacturer | Part Number |
|---|---|---|---|---|
| 1 | 2 | C1, C3 | Panasonic | ECJ-1VB0J475K |
| 2 | 34 | C2, C4, C5, C6, C7, C8, C9, C11, C12, C15, C16, C18, C19, C20, C22, C23, C24, C25, C26, C28, C29, C30, C32, C33, C34, C37, C38, C39, C40, C44, C50, C51, C52, C53 | Kemet | C0402C104K4RACTU |
| 3 | 5 | C10, C35, C42, C46, C48 | Taiyo Yuden | LMK107BJ106MALTD |
| 4 | 2 | C13, C14 | Kemet | C0402C180K3GACTU |
| 5 | 6 | C17, C21, C27, C31, C41, C45 | Kemet | C0402C103J4RACTU |
| 6 | 2 | C36, C43 | Kemet | C0402C105K9PACTU |
| 7 | 2 | C47, C49 | Taiyo Yuden | LMK212BJ226MG-T |
| 8 | 8 | D1, D2, D3, D4, D5, D6, D7, D8 | LITE-On, Inc. | LTST-C190KRKT |
| 9 | 1 | D9 | LITE-On, Inc. | LTST-C190KGKT |
| 10 | 1 | J1 | Molex | 22-28-4081 |
| 11 | 4 | J2, J3, J4, J5 | Samtec | |
| 12 | 1 | J6 | | |
| 13 | 1 | J7 | Neltron | 5075BMR-05-SM-CR |
| 14 | 5 | L1, L2, L3, L4, L5 | Murata | BLM18AG601SN1D |
| 15 | 3 | R1, R2, R3 | Yageo | RC0402FR-075K1L |
| 16 | 5 | R4, R9, R19, R22, R23 | Yageo | RC0402FR-072K2L |
| 17 | 8 | R5, R6, R7, R8, R42, R44, R53, R54 | Yageo | RC0603JR-070RL |
| 18 | 1 | R10 | Yageo | RC0402FR-0712KL |
| 19 | 3 | R11, R12, R13 | Yageo | RC0402FR-0710KL |
| 20 | 7 | R14, R15, R16, R17, R18, R20, R21 | Yageo | RC0603JR-070RL |
| 21 | 5 | R24, R25, R26, R27, R56 | Vishay/Dale | CRCW06031R00JNEAHP |
| 22 | 14 | R28, R29, R30, R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41 | Yageo | RC0603FR-07100RL |
| 23 | 9 | R43, R45, R46, R47, R48, R49, R50, R51, R52 | Yageo | RC0402FR-071KL |
| 24 | 1 | R55 | Yageo | RC0603FR-07100RL |
| 25 | 3 | TP1, TP2, TP3 | | |
| 26 | 1 | U1 | FTDI | FT2232HL |
| 27 | 1 | U2 | Microchip | 93LC56C-I/SN |
| 28 | 1 | U3 | Lattice | LCMXO2-7000HE-4TG144C or LCMXO2-1200ZE-1TG144C |
| 29 | 1 | U4 | Fairchild Semi | FAN1112SX |
| 30 | 1 | U5 | On Semi | NCP1117ST33T3G |
| 31 | 1 | X1 | TXC | 7M-12.000MAAJ-T |
| 32 | 1 | X2 | CTS | CB3LV-3C-50M0000 |

# ANEXO 2. DOCUMENTACIÓN DHT22

AOSONG

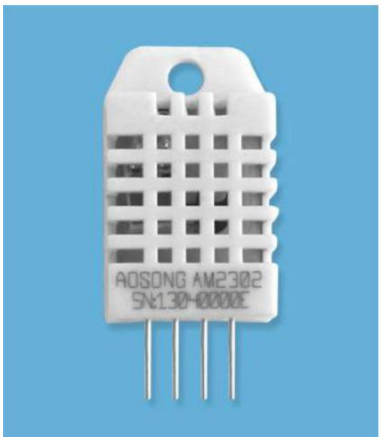
# Temperature and humidity module

## AM2302 Product Manual

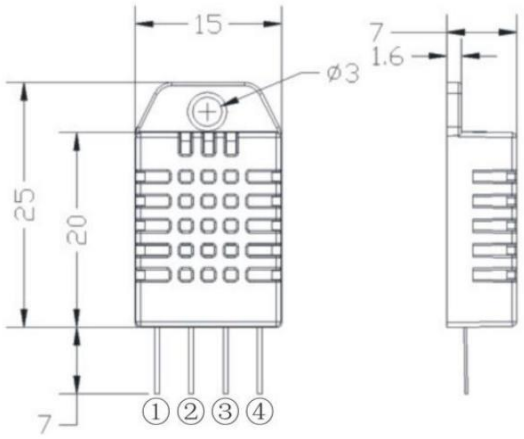

www.aosong.com

## 1、Product Overview

AM2302 capacitive humidity sensing digital temperature and humidity module is one that contains the compound has been calibrated digital signal output of the temperature and humidity sensors. Application of a dedicated digital modules collection technology and the temperature and humidity sensing technology, to ensure that the product has high reliability and excellent long-term stability. The sensor includes a capacitive sensor wet components and a high-precision temperature measurement devices, and connected with a high-performance 8-bit microcontroller. The product has excellent quality, fast response, strong anti-jamming capability, and high cost. Each sensor is extremely accurate humidity calibration chamber calibration. The form of procedures, the calibration coefficients stored in the microcontroller, the sensor within the processing of the heartbeat to call these calibration coefficients. Standard single-bus interface, system integration quick and easy. Small size, low power consumption, signal transmission distance up to 20 meters, making it the best choice of all kinds of applications and even the most demanding applications. Products for the 3-lead (single-bus interface) connection convenience. Special packages according to user needs.




Physical map




Dimensions (unit: mm)

## 2、Applications

HVAC, dehumidifier, testing and inspection equipment, consumer goods, automotive, automatic control, data loggers, home appliances, humidity regulator, medical, weather stations, and other humidity measurement and control and so on.

## 3、Features

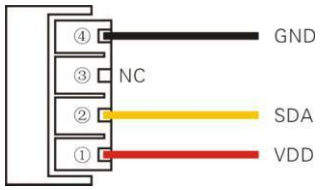
Ultra-low power, the transmission distance, fully automated calibration, the use of capacitive humidity sensor, completely interchangeable, standard digital single-bus output, excellent long-term stability, high accuracy temperature measurement devices.

## 4、The definition of single-bus interface

### 4.1 AM2302 Pin assignments

**Table 1：** AM2302 Pin assignments

| Pin | Name | Description |
|---|---|---|
| ① | VDD | Power (3.3V-5.5V) |
| ② | SDA | Serial data, bidirectional port |
| ③ | NC | Empty |
| ④ | GND | Ground |




**PIC1：** AM2302 Pin Assignment

### 4.2 Power supply pins（VDD GND）

AM2302 supply voltage range 3.3V - 5.5V, recommended supply voltage is 5V.

### 4.3 Serial data（SDA）

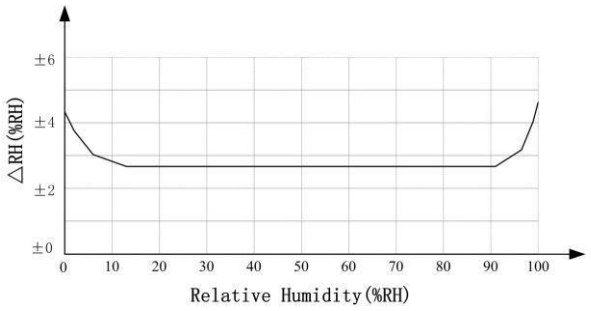
SDA pin is tri structure for reading, writing sensor data. Specific communication timing, see the detailed description of the communication protocol.

## 5、Sensor performance

### 5.1 Relative humidity

**Table 2：** AM2302 Relative humidity performance table

| Parameter | Condition | min | typ | max | Unit |
|---|---|---|---|---|---|
| Resolution | | | 0.1 | | %RH |
| Range | | 0 | | 99.9 | %RH |
| Accuracy [1] | 25℃ | | ±2 | | %RH |
| Repeatability | | | ±0.3 | | %RH |
| Exchange | | Completely interchangeable | | | |
| Response [2] | 1/e(63%) | | <5 | | S |
| Sluggish | | | <0.3 | | %RH |
| Drift [3] | Typical | | <0.5 | | %RH/yr |

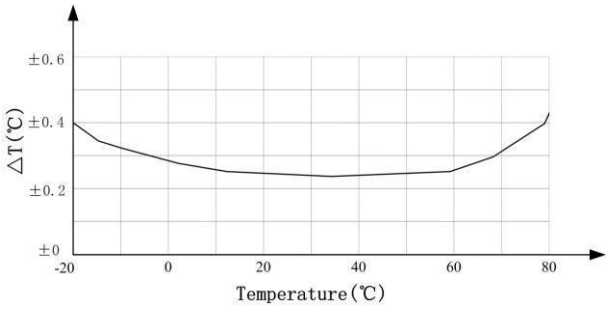



**Pic2:** At25℃ The error of relative humidity

### 5.2 Temperature

**Table 3：** AM2302 Relative temperature performance

| Parameter | Condition | min | typ | max | Unit |
|---|---|---|---|---|---|
| Resolution | | | 0.1 | | ℃ |
| | | | 16 | | bit |
| Accuracy | | | ±0.5 | ±1 | ℃ |
| Range | | -40 | | 80 | ℃ |
| Repeat | | | ±0.2 | | ℃ |
| Exchange | | Completely interchangeable | | | |
| Response | 1/e(63%) | | <10 | | S |
| Drift | | | ±0.3 | | ℃/yr |




**Pic3：** The maximum temperature error

## 6、Electrical Characteristics

Electrical characteristics, such as energy consumption, high, low, input, output voltage, depending on the power supply. Table 4 details the electrical characteristics of the AM2302, if not identified, said supply voltage of 5V. To get the best results with the sensor, please design strictly in accordance with the conditions of design in Table 4.

**Table** 4： AM2302 DC Characteristics

| Parameter | Condition | min | typ | max | Unit |
|---|---|---|---|---|---|
| Voltage | | 3.3 | 5 | 5.5 | V |
| Power consumption [4] | Dormancy | 10 | 15 | | μA |
| | Measuring | | 500 | | μA |
| | Average | | 300 | | μA |
| Low level output voltage | $I_{OL}$[5] | 0 | | 300 | mV |
| High output voltage | Rp<25 kΩ | 90% | | 100% | VDD |
| Low input voltage | Decline | 0 | | 30% | VDD |
| Input High Voltage | Rise | 70% | | 100% | VDD |
| Rpu[6] | VDD = 5V VIN = VSS | 30 | 45 | 60 | kΩ |
| Output current | turn on | | 8 | | mA |
| | turn off | 10 | 20 | | μA |
| Sampling period | | 2 | | | S |

[1] the accuracy of the factory inspection, the sensor 25°C and 5V, the accuracy specification of test conditions, it does not include hysteresis and nonlinearity, and is only suitable for non-condensing environment.

[2] to achieve an order of 63% of the time required under the conditions of 25℃ and 1m / s airflow.

[3] in the volatile organic compounds, the values may be higher. See the manual application to store information.

[4] this value at VDD = 5.0V when the temperature is 25℃, 2S / time, under the conditions of the average.

[5] low output current.

[6] that the pull-up resistor.

## 7、Single-bus communication（ONE-WIRE）

### 7.1 Typical circuits for single bus

Microprocessor and AM2302 connection typical application circuit is shown in Figure 4. Single bus communication mode, pull the SDA microprocessor I / O port is connected.

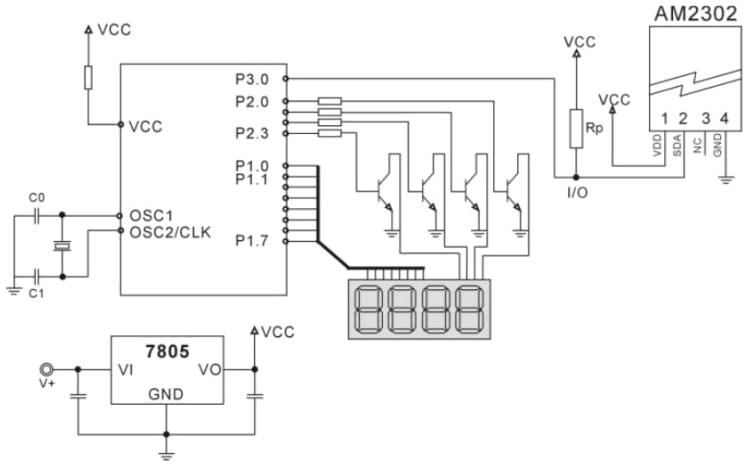
**Special instructions of the single-bus communication：**

1．Typical application circuit recommended in the short cable length of 30 meters on the 5.1K pull-up resistor pullup resistor according to the actual situation of lower than 30 m.

2．With 3.3V supply voltage, cable length shall not be greater than 100cm. Otherwise, the line voltage drop will lead to the sensor power supply, resulting in measurement error.

3．Read the sensor minimum time interval for the 2S; read interval is less than 2S, may cause the temperature and humidity are not allowed or communication is unsuccessful, etc..

4．Temperature and humidity values are each read out the results of the last measurement For real-time data that need continuous read twice, we recommend repeatedly to read sensors, and each read sensor interval is greater than 2 seconds to obtain accuratethe data.




**Pic4：** AM2302 Typical circuits for single bus

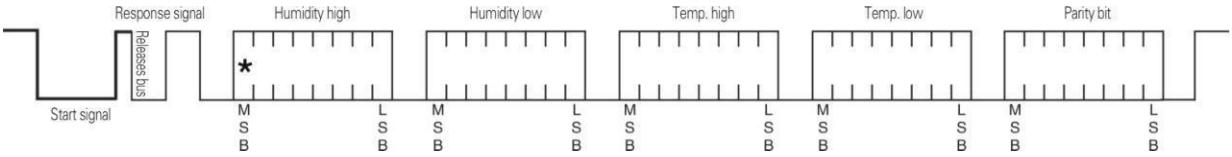
### 7.2、Single-bus communication protocol

#### ◎Single bus Description

AM2302 device uses a simplified single-bus communication. Single bus that only one data line, data exchange system, controlled by the data line to complete. Equipment (microprocessor) through an open-drain or tri-state port connected to the data line to allow the device does not send data to release the bus, while other devices use the bus; single bus usually require an external about 5.1kΩ pull-up resistor, so when the bus is idle, its status is high. Because they are the master-slave structure, only the host calls the sensor, the sensor will answer, so the hosts to access the sensor must strictly follow the sequence of single bus, if there is a sequence of confusion, the sensor will not respond to the host.

#### ◎Single bus to send data definition

SDA For communication and synchronization between the microprocessor and the AM2302, single-bus data format, a transmission of 40 data, the high first-out. Specific communication timing shown in Figure 5, the communication format is depicted in Table 5.




**Pic5：** AM2302 Single-bus communication protocol

**Table 5：** AM2302 Communication format specifier

| Name | Single-bus format definition |
|---|---|
| Start signal | Microprocessor data bus (SDA) to bring down a period of time (at least 800μ s) [1] notify the sensor to prepare the data. |
| Response signal | Sensor data bus (SDA) is pulled down to 80μ s, followed by high-80μ s response to host the start signal. |
| Data format | Host the start signal is received, the sensor one-time string from the data bus (SDA) 40 data, the high first-out. |
| Humidity | Humidity resolution of 16Bit, the previous high; humidity sensor string value is 10 times the actual humidity values. |
| Temp. | Temperature resolution of 16Bit, the previous high; temperature sensor string value is 10 times the actual temperature value;<br>The temperature is the highest bit (Bit15) is equal to 1 indicates a negative temperature, the temperature is the highest bit (Bit15) is equal to 0 indicates a positive temperature;<br>Temperature in addition to the most significant bit (Bit14 ~ bit 0) temperature values. |
| **Parity bit** | Parity bit = humidity high + humidity low + temperature high + temperature low |

**◎Single-bus data calculation example**

**Example 1：** 40 Data received：

| 0000 0010 | 1001 0010 | 0000 0001 | 0000 1101 | 1010 0010 |
|---|---|---|---|---|
| High humidity 8 | Low humidity 8 | High temp. 8 | Low temp. 8 | Parity bit |

**Calculate：**

0000 0010+1001 0010 +0000 0001+0000 1101= 1010 0010（Parity bit）

Received data is correct：

**humidity：** 0000 0010 1001 0010 = 0292H (Hexadecimal)= 2×256 + 9×16 + 2 = 658 => Humidity = 65.8%RH

**Temp.：** 0000 0001 0000 1101 = 10DH(Hexadecimal) = 1×256 + 0×16 + 13 = 269 => Temp.= 26.9℃

**◎Special Instructions：**

When the temperature is below 0 ℃, the highest position of the temperature data.

**Example：** -10.1 ℃ Expressed as 1 000 0000 0110 0101

**Temp.：** 0000 0000 0110 0101 = 0065H(Hexadecimal)＝ 6×16 +5 = 101

=> Temp. ＝ -10.1℃

**Example 2：** 40 received data：

| 0000 0010 | 1001 0010 | 0000 0001 | 0000 1101 | 1011 0010 |
|---|---|---|---|---|
| High humidity 8 | Low humidity 8 | High temp. 8 | Low temp. 8 | Parity bit |

**Calculate：**

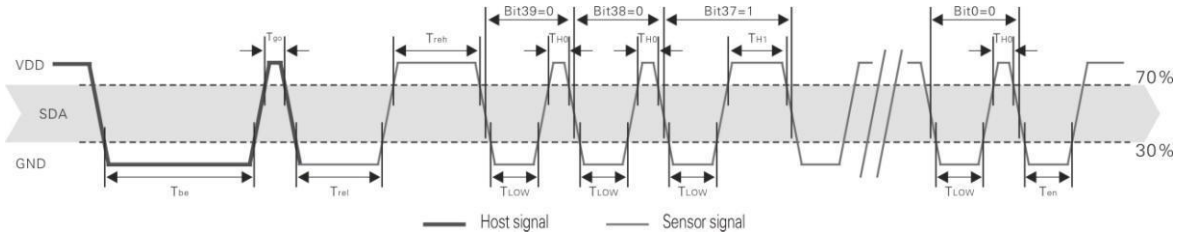
0000 0010+1001 0010 +0000 0001+0000 1101= 1010 0010 ≠ 1011 0010 （Validation error）

The received data is not correct, give up, to re-receive data.

### 7.3 Single-bus communication timing

User host (MCU) to send a start signal (data bus SDA line low for at least 800μ s) after AM2302 from Sleep mode conversion to high-speed mode.The host began to signal the end of the AM2302 send a response signal sent from the data bus SDA serial 40Bit's data, sends the byte high; data sent is followed by: Humidity high、 Humidity low、Temperature high、Temperature low、Parity bit，Send data to the end of trigger information collection, the collection end of the sensor is automatically transferred to the sleep mode, the advent until the next communication.

Detailed timing signal characteristics in Table 6，Single-bus communication timing diagram Pic 6：




**Pic 6：** AM2302 Single-bus communication timing

**Note:** the temperature and humidity data read by the host from the AM2302 is always the last measured value, such as the two measurement interval is very long, continuous read twice to the second value of real-time temperature and humidity values, while two readtake minimum time interval be 2S.

**Table 6:** Single bus signal characteristics

| Symbol | Parameter | min | typ | max | Unit |
|---|---|---|---|---|---|
| $T_{be}$ | Host the start signal down time | 0.8 | 1 | 20 | mS |
| $T_{go}$ | Bus master has released time | 20 | 30 | 200 | μS |
| $T_{rel}$ | Response to low time | 75 | 80 | 85 | μS |
| $T_{reh}$ | In response to high time | 75 | 80 | 85 | μS |
| $T_{LOW}$ | Signal "0", "1" low time | 48 | 50 | 55 | μS |
| $T_{H0}$ | Signal "0" high time | 22 | 26 | 30 | μS |
| $T_{H1}$ | Signal "1" high time | 68 | 70 | 75 | μS |
| $T_{en}$ | Sensor to release the bus time | 45 | 50 | 55 | μS |

**Note：** To ensure the accurate communication of the sensor, the read signal, in strict accordance with the design parameters and timing in Table 6 and Figure 6.

### 7.4 Peripherals read step example

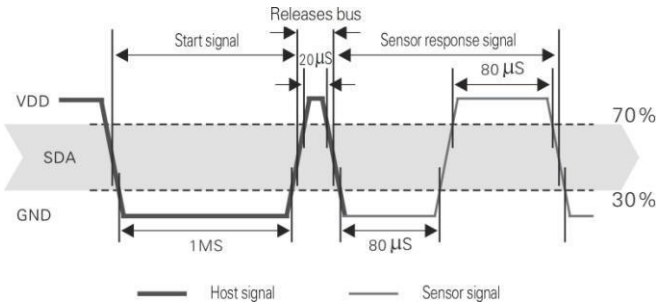
Communication between the host and the sensor can read data through the following three steps to complete.

**Step 1**

AM2302 have to wait for the power (on AM2302 power 2S crossed the unstable state, the device can not send any instructions to read during this period), the test environment temperature and humidity data, and record data, since the sensor into a sleep state automatically. AM2302 The SDA data line from the previous pull-up resistor pulled up is always high, the AM2302 the SDA pin is in input state, the time detection of external signal.

**Step 2**

Microprocessor I/O set to output, while output low, and low hold time can not be less than 800us, typical values are down 1MS, then the microprocessor I/O is set to input state, the release of the bus, due to the pull-up resistor, the microprocessor I/O AM2302 the SDA data line also will be high, the bus master has released the AM2302 send a response signal, that is, the output 80 microseconds low as the response signal, tightthen output high of 80 microseconds notice peripheral is ready to receive data signal transmission as shown to Pic7 :




**Pic7**：Single bus decomposition of the timing diagram

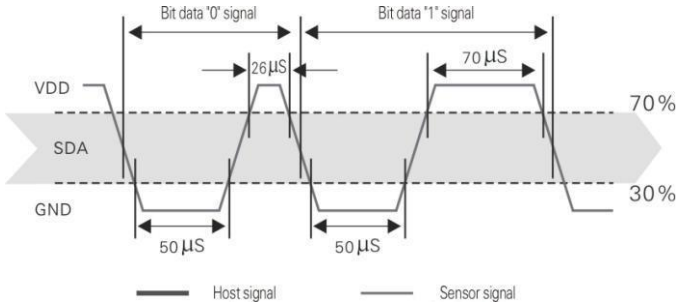
**Step 3**

AM2302 sending the response, followed by the data bus SDA continuous serial output 40 data, the microprocessor receives 40 data I/O level changes.

Bit data "0" format: 26-28 microseconds 50 microseconds low plus high;

Bit data "1" format: the high level of low plus, 50 microseconds to 70 microseconds;

Bit data "0" bit data "1" format signal shown to pic 8:

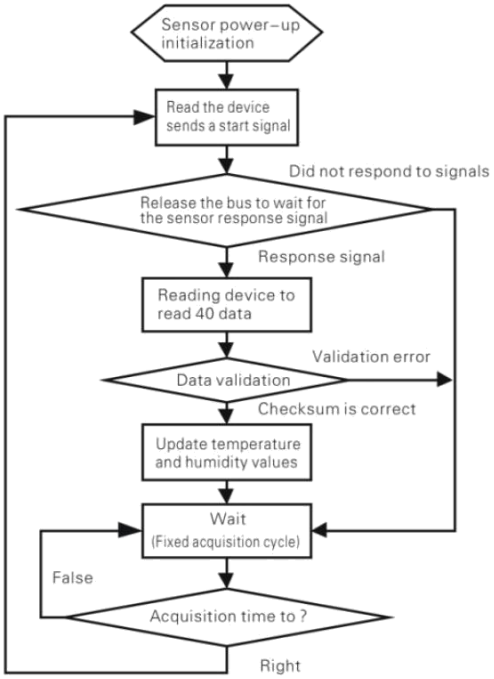



**Pic 8**：The single bus break down the timing diagram

AM2302 data bus SDA output 40 data continue to output the low 50 microseconds into the input state, followed by pull-up resistor goes high. AM2302 internal re-test environmental temperature and humidity data, and record the data, the end of the test records, the microcontroller automatically into hibernation. Microcontroller only after receipt of the start signal of the host wake-up sensor, into the working state.

**7.5 Peripheral to read flow chart**

AM2302 sensor read single bus flow chart diagram shown in Figure 9, we also provide the C51 read the code examples, customers need to download, please visit our website (www.aosong.com) related to downloadthis manual does not provide the code description.




**Pic9**：Single-bus to read the flow chart

## 8、Application of information

1. Work and storage conditions

Outside the sensor the proposed scope of work may lead to temporary drift of the signal up to 300% RH. Return to normal working conditions, sensor calibration status will slowly toward recovery. To speed up the recovery process may refer to "resume processing". Prolonged use of non-normal operating conditions, will accelerate the aging of the product.

Avoid placing the components on the long-term condensation and dry environment, as well as the following environment.

A, salt spray

B, acidic or oxidizing gases such as sulfur dioxide, hydrochloric acid

Recommended storage environment

Temperature: 10 ~ 40 ℃ Humidity: 60% RH or less

2. The impact of exposure to chemicals

The capacitive humidity sensor has a layer by chemical vapor interference, the proliferation of chemicals in the sensing layer may lead to drift and decreased sensitivity of the measured values. In a pure environment, contaminants will slowly be released. Resume processing as described below will accelerate this process. The high concentration of chemical pollution (such as ethanol) will lead to the complete damage of the sensitive layer of the sensor.

3. The temperature influence

Relative humidity of the gas to a large extent dependent on temperature. Therefore, in the measurement of humidity,

should be to ensure that the work of the humidity sensor at the same temperature. With the release of heat of electronic components share a printed circuit board, the installation should be as far as possible the sensor away from the electronic components and mounted below the heat source, while maintaining good ventilation of the enclosure. To reduce the thermal conductivity sensor and printed circuit board copper plating should be the smallest possible, and leaving a gap between the two.

4. Light impact

Prolonged exposure to sunlight or strong ultraviolet radiation, and degrade performance.

5. Resume processing

Placed under extreme working conditions or chemical vapor sensor, which allows it to return to the status of calibration by the following handler. Maintain two hours in the humidity conditions of 45℃ and <10% RH (dry); followed by 20-30℃ and> 70% RH humidity conditions to maintain more than five hours.

6. Wiring precautions

The quality of the signal wire will affect the quality of the voltage output, it is recommended to use high quality shielded cable.

7. Welding information

Manual welding, in the maximum temperature of 300℃ under the conditions of contact time shall be less than 3 seconds.

8. Product upgrades

Details, please the consultation Aosong electronics department.

**9、The license agreement**

Without the prior written permission of the copyright holder, shall not in any form or by any means, electronic or mechanical (including photocopying), copy any part of this manual, nor shall its contents be communicated to a third party. The contents are subject to change without notice.

The Company and third parties have ownership of the software, the user may use only signed a contract or software license.

**10、Warnings and personal injury**

This product is not applied to the safety or emergency stop devices, as well as the failure of the product may result in injury to any other application, unless a particular purpose or use authorized. Installation, handling, use or maintenance of the product refer to product data sheets and application notes. Failure to comply with this recommendation may result in death and serious personal injury. The Company will bear all damages resulting personal injury or death, and waive any claims that the resulting subsidiary company managers and employees and agents, distributors, etc. that may arise, including: a variety of costs, compensation costs, attorneys' fees, and so on.

**11、Quality Assurance**

The company and its direct purchaser of the product quality guarantee period of three months (from the date of delivery). Publishes the technical specifications of the product data sheet shall prevail. Within the warranty period, the product was confirmed that the quality is really defective, the company will provide free repair or replacement. The user must satisfy the following conditions:

① The product is found defective within 14 days written notice to the Company;

② The product shall be paid by mail back to the company;

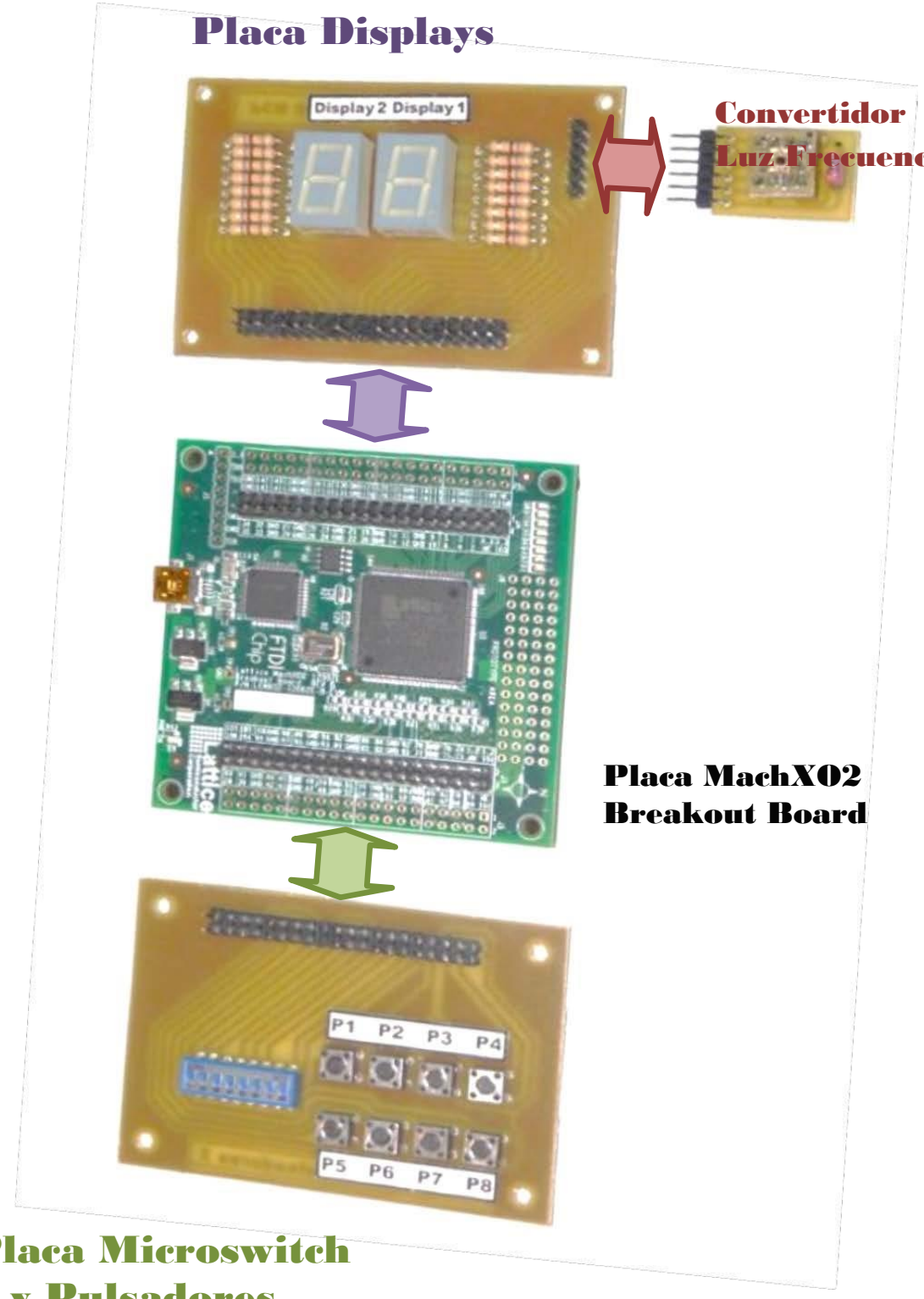
③ The product should be within the warranty period.

The Company is only responsible for those used in the occasion of the technical condition of the product defective product. Without any guarantee, warranty or written statement of its products used in special applications. Company or its products applied to the reliability of the product or circuit does not make any commitment.

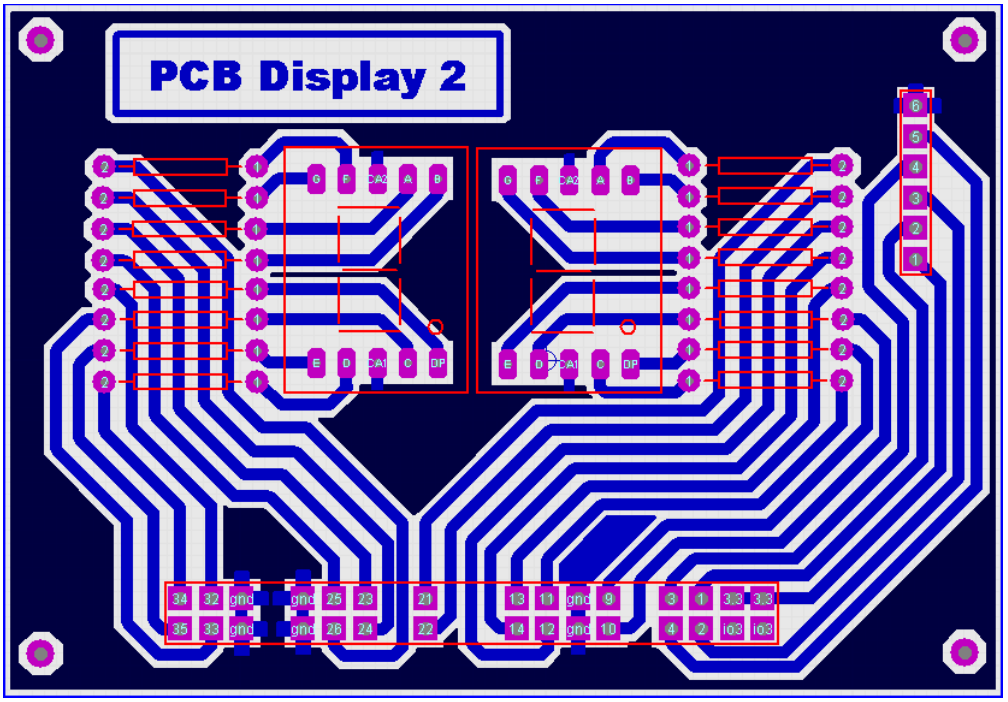
# ANEXO 3. DOCUMENTACIÓN PLACAS PCB

# MÓDULO DE PRÁCTICAS DE

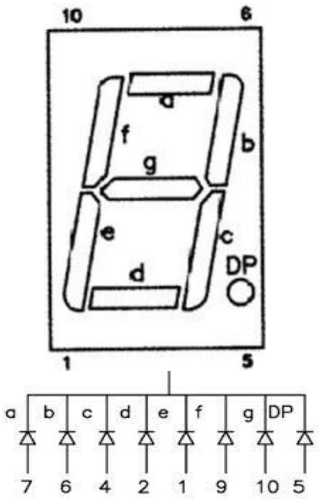
# SISTEMAS ELECTRÓNICOS RECONFIGURABLES



# PLACA PCB DISPLAY 2




## PINOUT DE LA FPGA ASIGNADO EN LA PLACA PCB DISPLAY 2:

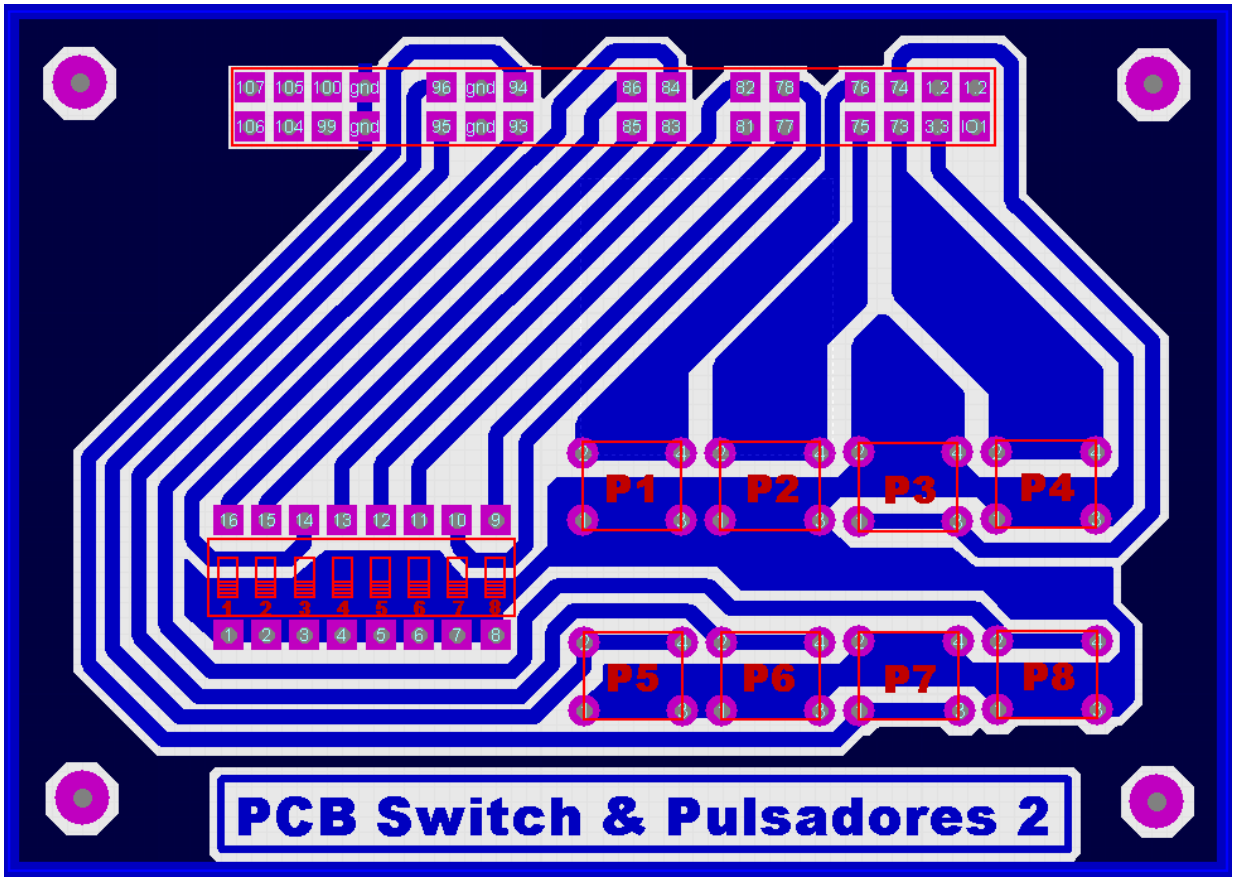



| *DISPLAY 2 (A LA IZQDA)* | | | *DISPLAY1 (A LA DRCHA)* | | |
|---|---|---|---|---|---|
| A2 | PIN7 | 23 | A1 | PIN7 | 21 |
| B2 | PIN6 | 25 | B1 | PIN6 | 22 |
| C2 | PIN4 | 33 | C1 | PIN4 | 11 |
| D2 | PIN2 | 34 | D1 | PIN2 | 10 |
| E2 | PIN1 | 35 | E1 | PIN1 | 13 |
| F2 | PIN9 | 26 | F1 | PIN9 | 12 |
| G2 | PIN10 | 24 | G1 | PIN10 | 14 |
| DP2 | PIN5 | 32 | DP1 | PIN5 | 9 |

*PINES AUXILIARES PARA OTRA PLACA DE AMPLIACIÓN (DE ABAJO A ARRIBA)*

| | | | |
|---|---|---|---|
| PIN1 | VCC (3.3v) | PIN4 | 3 |
| PIN2 | 1 | PIN5 | 4 |
| PIN3 | 2 | PIN6 | GND |

# PLACA PCB SWITCH & PULSADORES 2

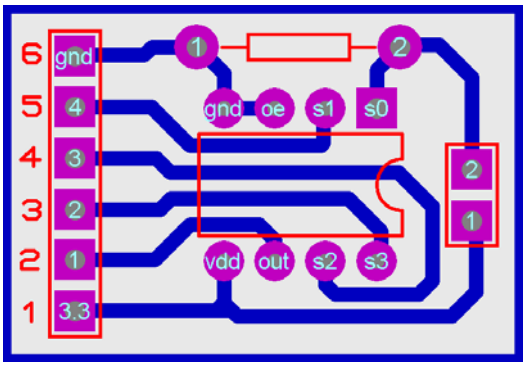


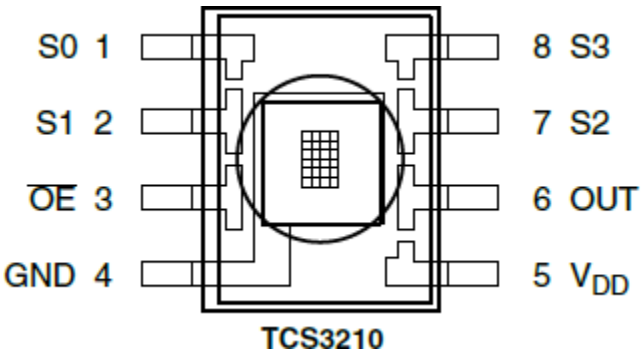

| PINES MICROSWITCH | PINES PULSADORES |
|---|---|
| SW1 86 | PULS1 76 |
| SW2 85 | PULS2 75 |
| SW3 84 | PULS3 74 |
| SW4 83 | PULS4 73 |
| SW5 82 | PULS5 96 |
| SW6 81 | PULS6 95 |
| SW7 78 | PULS7 94 |
| SW8 77 | PULS8 93 |

**NOTA:** Al subir a 'ON' un microswitch o presionar un pulsador el estado de la entrada

correspondiente en la FPGA se pone a nivel alto (3.3 V).

# PLACA PCB MÓDULO LUZ FRECUENCIA




TCS3210

**El Chip TCS3210 es un convertidor luz frecuencia con color programable y salida en frecuencia ajustable. Con los pines S0 y S1 se configura la frecuencia y con los pines S2 y S3 el color. A continuación se muestran las tablas que determinan el comportamiento del convertidor.**

| 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 |

**Dentro de nuestro módulo la asignación de pines es la siguiente:**

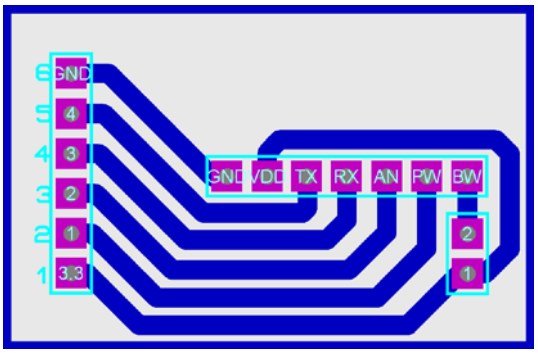
| | | |
|---|---|---|
| **PIN1** | **VDC (3.3v)** | |
| **PIN2** | **OUT** | **1** |
| **PIN3** | **S3** | **2** |
| **PIN4** | **S2** | **3** |
| **PIN5** | **S1** | **4** |
| **PIN6** | **GND** | |

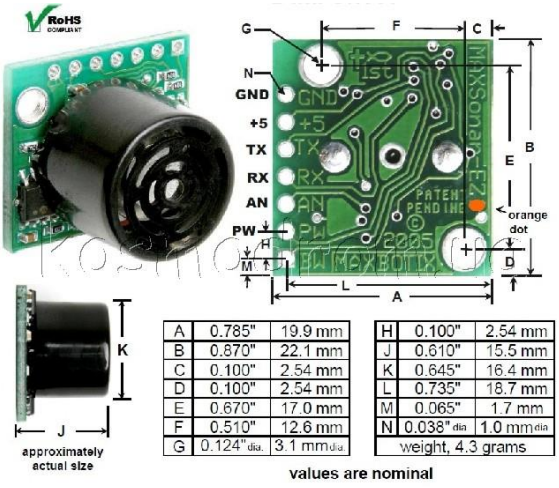
**La placa también dispone de un Jumper que define el valor del pin 'S0'.**

**Si JMP está cerrado S0='1' y si se retira el jumper entonces S0='0'.**

**Nota:** Esta placa solo se puede conectar mediante el conector de ampliación de la placa PCB Display.

# PLACA PCB MÓDULO ULTRASONIDOS




| | | |
|---|---|---|
| A | 0.785" | 19.9 mm |
| B | 0.870" | 22.1 mm |
| C | 0.100" | 2.54 mm |
| D | 0.100" | 2.54 mm |
| E | 0.670" | 17.0 mm |
| F | 0.510" | 12.6 mm |
| G | 0.124" dia. | 3.1 mm dia. |

| | | |
|---|---|---|
| H | 0.100" | 2.54 mm |
| J | 0.610" | 15.5 mm |
| K | 0.645" | 16.4 mm |
| L | 0.735" | 18.7 mm |
| M | 0.065" | 1.7 mm |
| N | 0.038" dia | 1.0 mm dia |
| weight, 4.3 grams | | |

values are nominal

**El LV-MaxSonar-EZ3 es un sensor ultrasónico que tiene integrado emisor y receptor en un solo módulo. Tiene un rango de detección de 0 cm a 6.45 m. Desde 15.24 cm hasta los 6.45 m tiene una resolución de una pulgada (2.54 cm). Dispone de tres formatos de salida:**

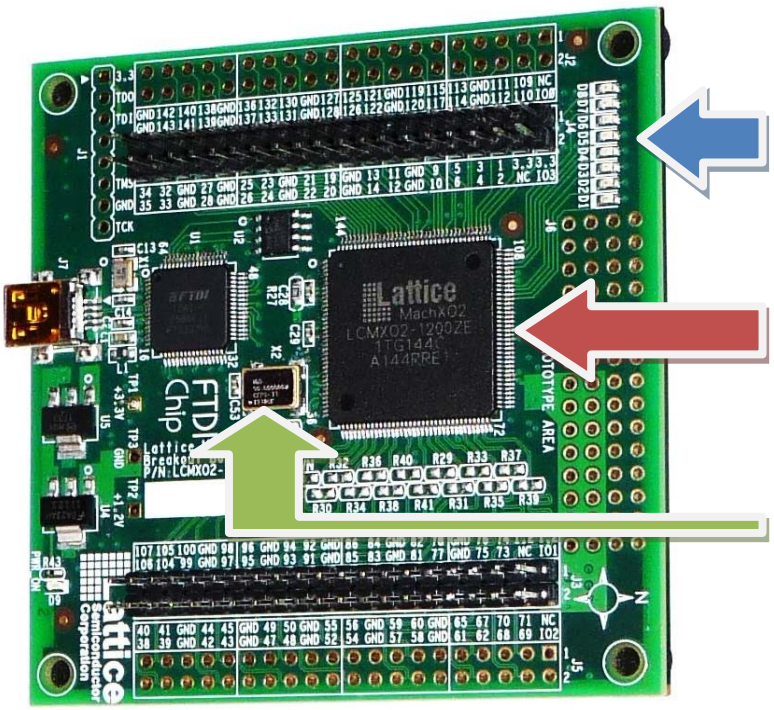
- **Ancho de pulso a través de la patilla PW (Pin 1 de la FPGA). La distancia se determina aplicando el factor de escala de 147uS por pulgada.**
- **Salida de tensión analógica AN (Pin 2 de la FPGA). La distancia se determina en nuestro caso para una Vcc de 3.3 voltios aplicando el factor de escala de 6.4mV/pulgada (El factor de escala es *Vcc/512* V/pulgada).**
- **Salida digital serie TX (Pin 4 de la FPGA). Para habilitar esta salida hay que poner a nivel bajo o sin conexión la patilla BW. Esta salida emplea comunicación RS232. Inicia la trama mandando el código ASCII 'R', después manda tres dígitos en ASCII indicando la distancia en pulgadas hasta un valor máximo de 255 y finalmente termina la trama mandando un retorno de carro (valor ASCII 13).**

**El Pin 3 de la FPGA va al pin 'RX' del módulo. La entrada RX a nivel alto durante un mínimo de 20 uS realiza una lectura. En caso de permanecer a nivel alto realizará lecturas continuadas y a nivel bajo detiene la medición.**

**La placa dispone de un Jumper que define el valor del pin 'BW'. La entrada BW como se ha comentado anteriormente solo sirve para habilitar (BW='0') o deshabilitar (BW='1') la salida digital serie.**
**Si JMP está cerrado BW='1' y si se retira el jumper entonces BW='0'.**

**Nota:** Esta placa solo se puede conectar mediante el conector de ampliación de la placa PCB Display.

# PLACA MachXO2 BREAKOUT BOARD




## RELOJ EXTERNO

Se ha agregado a la placa un reloj externo de 50 MHz. El reloj dispone de una patilla de habilitación CLK_EN que va al PIN 32 de la FPGA. A nivel alto el Cristal externo esta habilitado y a nivel bajo deshabilitado. La salida del reloj CLK_OUT va al PIN 27 de la FPGA.

**CLK_OUT PIN 27** **CLK EN PIN 32**

## LEDS ON BOARD

| | |
|---|---|
| **D1 -> PIN 97** | **D5 -> PIN 104** |
| **D2 -> PIN 98** | **D6 -> PIN 105** |
| **D3 -> PIN 99** | **D7 -> PIN 106** |
| **D4 -> PIN 100** | **D8 -> PIN 107** |