Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automatica

Estación Meteorológica basada en FPGAs

Autor:

Sánchez Marcos, Diego

Tutor:

ANDRES RODRIGUEZ TRELLES, FRANCISCO JOSE DE

Dpto. Tecnología Electrónica

Valladolid, Julio 2018.

Parte I

Resumen y Palabras Clave

RESUMEN

Este TFG presenta el desarrollo e implantación de una estación meteorológica en el Departamento de Tecnología Electrónica de la Universidad de Valladolid, y se complementa con otro TFG que implante la comunicacion inalámbrica entre varias estaciones con sensores remotos.
La estación tendrá la capacidad de medir la temperatura y humedad ambiente y permitirá la transmisión de los datos.
La lógica del sistema se ha implantado en una FPGA de la familia MachXO2, con capacidad de 1200-LUTs-4 y 144 pines. La configuración de la FPGA se realiza mediante descripciones VHDL y diagramas esquemáticos de Active-CAD. La medición de las variables meteorológicas la realiza un sensor DHT22. La comunicación entre dispositivos se realizará mediante bluetooth. La visualización de la información se realiza con dos displays de siete segmentos y leds. La selección de dicha información se selecciona a través de los interruptores situados en la placa.

PALABRAS CLAVE

FPGA, ESTACIÓN METEOROLÓGICA, VHDL, TEMPERATURA, HUMEDAD.

Parte II

Indices

Índice general

Índice de figuras

GLOSARIO

- **Amplificador Operacional:** es un dispositivo amplificador electrónico de ganancia acoplado en corriente continua que tiene dos entradas y una salida. En esta configuración, la salida del dispositivo, es generalmente de cientos de miles de veces mayor que la diferencia de potencial entre sus entradas.

- **barrel-Shifter:** se trata de un conjunto de registros en los que la salida de los registros se encuentra unida a la entrada del siguiente registro. Los datos en este tipo de registros van circulando por todos los registros, pueden estar en forma de bus; con un principio o un final o en forma de anillo con la entrada de datos del primer registro unido a la salida de datos del último.

- **Bit** es el acrónimo de «Binary digit» es un dígito del sistema de numeración binaria. La capacidad de almacenamiento de una memoria digital también se mide en bits.

- **Byte:** es la unidad de información de base utilizada en computación y en telecomunicaciones, y que resulta equivalente a un conjunto ordenado de 8 bits.

- **Buffer-Bidireccional:** se trata de un circuito lógico que permite la controlar la conexión entre dos dispositivos de forma que permite controlar el envío o recepción de información.

- **C:** es un lenguaje de programación basado en B. Es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. El lenguaje C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear aplicaciones.

- **CPLD:** acrónimo en inglés de (Complex Programmable Logic Device), se trata de un dispositivo electrónico programable que permite la implementación de sistemas eficaces en un menor espacio mejorando la

fiabilidad del diseño y reduciendo costos.

- **Display:** se denomina así a un dispositivo con el que cuentan ciertos dispositivos electrónicos que permite mostrar información a los usuarios de manera visual o táctil.

- **Domótica:** se denomina a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas.

- **FIFO:** es un concepto utilizado en estructuras de datos, contabilidad de costes y teoría de colas. Guarda analogía con las personas que esperan en una cola y van siendo atendidas en el orden de llegada «primera en entrar primera en salir».

- **Flip-Flop:** es un elemento electrónico que puede permanecer en dos estados por un tiempo indefinido en ausencia de perturbaciones. Esta característica lo convierte en muy utilizada dentro de la electrónica digital para memorizar la información.

- **I2C:** es un bus serie de datos desarrollado por Philips Semiconductors. Se utiliza principalmente internamente para la comunicación ente diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

- **Hardware:** en informática se refiere a las partes físicas tangibles de cualquier sistema informático.

- **Lattice:** es una empresa americana creadora de dispositivos lógicos programables como (FPGAs, CPLDs y SPLDs). Además de producir hardware se encarga de generar software para programar sus dispositivos.

- **LIFO:** es un termino de estructuras de datos y teoría de colas. Guarda analogía con una pila de platos «primero en salir ultimo en entrar».

- **LUT:** Unidad Lógica de programación en FPGAs, se utiliza para medir la capacidad de almacenar funciones de una FPGA. Normalmente se encuentran LUTs para realizar funciones con 4 o 5 variables.

- **Comunicación Maestro-Esclavo:** el termino se aplica a redes en las que existen nodos con papeles diferenciados de la forma que el maestro se trata de un nodo activo que pide información a nodos esclavos que son pasivos y necesitan que se les indique cuando enviar información a través de la red.

- **Microcontroladores:** es un circuito integrado programable, capaz de ejecutar ordenes grabadas en su memoria. Esta compuesto por varios bloques funcionales, los cuales desempeñan una función específica.

- **Microprocesadores:** es un circuito integrado encargado de ejecutar los programas, utilizando instrucciones programadas en lenguaje de bajo nivel. Para su correcto funcionamiento necesita de más unidades para que lleven a cabo las tareas que no puede desarrollar.

- **Multiplexor:** circuito combinacional con varias entradas y una única salida de datos. Están dotados de entradas de control que permiten realizar una selección en las entradas que se dirige hacia la salida del dispositivo.

- **PLL:** se trata de una utilidad para sistemas electrónicos en los que la frecuencia y la fase son realimentadas dentro del propio bloque.

- **Pseudoalatorio:** aplicado a los números se trata de un número generado en un proceso que parece producir números al azar, pero no lo hace realmente. Las secuencias de números no presentan ningún patrón o regularidad desde el punto de vista estadístico,a pesar de haber sido

generadas por un algoritmo determinista, en el cual las condiciones iniciales generan siempre el mismo resultado.

- **RAM:** acrónimo en inglés (Random Access Memory) es la memoria de trabajo de las computadoras y otros sistemas operativos, los programas y la mayor parte del software. En la RAM se cargan todas las instrucciones que ejecuta la unidad central de procesamiento y otras unidades del ordenador además de contener los datos que manipulan los distintos programas.

- **ROM:** también conocida como memoria de solo lectura, es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía.

- **Slice:** Unidad Lógica de programación en FPGAs, esta puede contener diferentes bloques internamente como LUTs y funciones específicas implementadas y prediseñadas.

- **Smart-sensor:** es un dispositivo que toma una entrada física del entorno y usa sus propios recursos computacionales para generar una salida específica procesando la información antes de enviarla.

- **Software:** se conoce al soporte lógico de un sistema informático, que comprende el conjunto de componentes lógicos necesarios que hacen posible la realización de tareas específicas.

- **Transistor:** es un dispositivo electrónico semiconductor utilizado para entregar una señal de salida en respuesta a una señal de entrada. Cumple funciones de amplificador, oscilador, conmutador o rectificador.

- **Verilog:** es un lenguaje de descripción de hardware usado para modelar sistemas electrónicos. El lenguaje soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes

niveles de abstracción.

- **VHDL:** es un lenguaje de especificación defindo por el IEEE utilizado para describir circuitos digitales y para la automatización del diseño electrónico. VHDL es el acrónimo de (Very High Speed Integrated Circuit y Hardware Description Language).

- **Wishbone-bus:** es un hardware de codigo abierto utilizado para permitir la comunicación entre las partes de un circuito integrado entre ellas y poder comprobar que los procesos de fabricación se han fabricado de forma correcta.

- **Xilinx:** compañía tecnológica americana, principalmente distribuidora de dispositivos lógicos programables, es reconocida por inventar los FPGA.

Parte III

Objetivos y Especificaciones

Capítulo 1

Objetivos

El objetivo principal la realización de una estación meteorológica, con capacidad para medir la temperatura y humedad, y visualizar las medidas realizadas, de una forma precisa y económica. El sistema deberá tener en cuenta, además, la posibilidad de incorporar, en proyectos posteriores, la comunicación de la estación con sensores remotos y otras estaciones de forma inalámbrica, a la que se le hace mención en el Anejo 1.

Para alcanzar el objetivo principal se han planteado los siguientes subaparatados, como medio de realizar el dispositivo.

- Estudio de las alternativas existentes para la implantación de la lógica, y la selección del dispositivo concreto, dentro de la alternativa seleccionada.
- Análisis y elección de las alternativas disponibles a nivel de Sensores más adecuados para la aplicación.
- Estudio de los protocolos de comunicación, analizando si se trata de buses estándar (I2C) o protocolos específicos.
- Profundización en el conocimiento del CAD electrónico, empleado y selección de herramientas externas que faciliten el diseño, como son un editor gráfico de diagramas de estado y un generador de código VHDL, de forma automática, a partir de diagramas de estado.
- Profundización en el conocimiento del lenguaje de descripción de hardware VHDL, al tener que realizar con él, algoritmos sintetizables complejos.
- Empleo de herramientas externas que facilitan la puesta a punto, como son los analizadores lógicos.

- Utilizar técnicas de trabajo en equipo y división de tareas de forma jerárquica dividiéndolo en bloques más simple.

Capítulo 2

Especificaciones

Las especificaciones que se deben cumplir son las siguientes

- Al dar tensión, el sistema comenzará con el proceso de lectura de datos, realizando antes todas las acciones previas necesarias para su funcionamiento.
- Dispondrá de un reset interno que llevará el sistema al estado inicial.
- La visualización de los mostrará tres dígitos, en visualizadores de 7 segmentos.
- Se dispondrá de un interruptor que permita seleccionar que parámetro se visualiza.
- La actualización de la información se realizará en un intervalo de aproximadamente 2 segundos.
- Realizar un registro con los valores máximos y mínimos, medidos por el dispositivo a lo largo de un tiempo indefinido.
- Optimizar las funciones de forma que ocupen los menores recursos en el dispositivo.

Capítulo 3

Organización de la memoria

La memoria se ha organizado de la siguiente forma:

En los Capítulos 1 y 2 se plantean los objetivos y especificaciones del TFG respectivamente.

El Capítulo 3 se presenta la organización global de la memoria.

El Capítulo 4 se centra en la selección del dispositivo a utilizar para la parte de control.

El Capítulo 5 se realiza una búsqueda en el mercado de los sensores disponibles.

El Capítulo 6 se presenta la generación de la configuración del dispositivo como tal, centrándose en la división por bloques y la elaboración de los mismos.

El Capítulo 7 se comenta el funcionamiento Final de la FPGA y posibles vías de comercialización.

El Capítulo 8 se comentan las conclusiones obtenidas del desarrollo del dispositivo.

Los Capítulos del 9 al 14 son los Anejos en los que se ven los temas no tratados en los anteriores capitulos.

El Capítulo 9 se realiza el estudio Económico y de utilización de Recursos.

El Capítulo 10 se centra en el TFG de comunicación de FPGA con Bluetooth, referido durante toda la memoria, presentando un breve resumen del mismo.

Los Capítulos 11-14: son datasheet proporcionados por los fabricantes externos al dispositivos de los módulos utilizados.

Parte IV

DESARROLLO DEL TFG

Capítulo 4

Hardware de Control

En este capitulo se seleccionará el Hardware que llevará a cabo la parte de control del dispositivo de forma que realizará una búsqueda del dispositivo que mejor se adapte a las necesidades de este TFG.

La elección del hardware de control se realizado siguiendo criterios como el conocimiento del lenguaje de programación, haber trabajado anteriormente con dispositivos similares y mayores posibilidades de aprendizaje. Como posibles soluciones se propusieron:

Microprocesadores: dispositivos con grandes capacidades debido a su arquitectura, que les permite tener una gran capacidad de cálculo, para poder utilizar un microprocesador sería necesario integrarlo dentro de todo un sistema informático. El coste económico de utilizar esto sería excesivo para la aplicación y se estarían desaprovechando tanto poder de calculo como recursos.

Microcontroladores: Son dispositivos más versátiles y con capacidades de calculo inferiores a los microprocesadores, estos sistemas tienen una arquitectura centralizada lo cual obliga a tener en un solo chip toda la arquitectura necesaria para su correcto funcionamiento.

FPGAs: Son dispositivos muy flexibles sin una arquitectura definida, permitiendo que el desarrollador genere su propia arquitectura de forma que variando la configuración el dispositivo se podrá adaptar a las diferentes necesidades a la hora de la implementación.

Para el desarrollo de este TFG se ha optado por la utilización de FPGAs debido a que estas permiten obtener gran flexibilidad a la hora del desarrollo de aplicaciones, para generar el control de los sensores será necesario un mayor conocimiento de modo de trasmisión de los datos por parte del sensor y el procesamiento de estos. La búsqueda de FPGAs tendrá que tener en cuenta varios factores, ya que los fabricantes presentan gran variedad de alternativas dentro de una misma clase de dispositivos, pudiendo variar el empaquetado de estos y así el número de entradas / salidas disponibles para el usuario.

Otro factor a tener en cuenta a la hora de seleccionar un dispositivo es el hardware interno, ya que para un mismo fabricante y un mismo modelo existen diferentes capacidades Hardware, normalmente para las FPGAs esta capacidad se mide en celdas lógicas programables capaces de utilizar hasta 4 variables. La última característica a tener en cuenta es el consumo ya que existen diferentes tecnologías, dependiendo de si interesa bajo consumo o mejores prestaciones, como velocidad de las puertas lógicas.

4.0 Alternativas de Diseño FPGA

Las FPGAs se basan en dispositivos reprogramables de forma que permiten ser configuradas para desempeñar ciertas tareas. A lo largo de los años han ido evolucionando hasta el día de hoy, en el que se consideran como una posible alternativa para la implantación de sistemas digitales complejos.

Las FPGAs fueron inventadas en el año 1984 por Ross Freeman y Bernard Vonderschmitt cofundadores de Xilinx, estos dispositivos aparecen como una evolución de los CPLDs. Siendo muy superior la densidad de puertas lógicas presentes en una FPGA con respecto a una CPLD.

Hoy en día las FPGAs están presentes en campos tan diversos como la automoción, la electrónica de consumo, o la investigación espacial. La tecnología FPGA tiene una aplicación horizontal en todas las industrias que requieren computación a alta velocidad. Tiene cabida en empresas que realizan las actividades indicadas en un gran tipo de actividades electrónicas.

Para esta tecnología también es muy importante el soporte académico. En el caso español, la incorporación fue también rápida y es habitual que el grupo de españoles sea el mayoritario en cualquier edición del FPL, el congreso europeo sobre FPGAs. En 2001 se fundó el JCRA (Jornadas de Computación Reconfigurable) como punto de encuentro de mundo español de la FPGA. Sus sedes han sido Alicante, Granada, Madrid (en la Universidad Autónoma) y Barcelona.

Aunque las FPGAs también se utilizan en grandes compañías, una gran parte de estas se encuentra en equipos de I+D.

Desde la aparición del circuito integrado, existen dos alternativas para realizar un hardware digital: codificación del algoritmo en un microprocesador o un mapeo directo del algoritmo en hardware.

Los microprocesadores, junto con los microcontroladores y DSPs permiten resolver eficazmente la mayor parte de los problemas electrónicos. En el mundo digital existen infinidad de problemas que se pueden resolver con un procesador sencillo. Esto lleva a que aún hoy día los micros de 8 bits son los más vendidos. Sin embargo, existe una serie de problemas donde los micros no son suficientes. Tal es el caso cuando la E/S de datos combina gran cantidad y gran velocidad, o cuando el número de operaciones por muestra es elevado. En

tal caso, la única opción era la realización de un ASICs (Circuito Integrado de Aplicación Específica), una tecnología que aparece en la década de los 80.

Los principales factores a tener en cuenta a la hora de barajar esta opción es 1º el costo que suele descartar esta opción tecnológica y 2º el consumo de potencia, que suele ser el factor principal que puede obligar a utilizar esta tecnología.

Dentro de los ASICs destacan los Gate Arrays, de los cuales deriva el nombre de FPGA. Están formados por filas de transistores sin interconexión junto con canales intermedios para rutado. Permiten una utilización de hasta un 90 % del área de silicio y se fabrica en diferentes tamaños. La idea central de los Gate Arrays es que con 4 transistores e interconexiones ente ellos, puede construirse cualquier puerta lógica. Los transistores son estándares y las interconexiones personalizables por el desarrollador.

Aunque los Gate Arrays y opciones tecnológicas posteriores como los Sea-of-Gates y los Standard Cells permiten realizar circuitos con retardo de área y consumo mínimo, el costo de fabricación y la vulnerabilidad a errores de diseño han limitado en la actualidad su utilización a grandes productos de consumo.

La FPGA es un componente estándar (re)programable por el usuario. Esto Implica que la interconexión debe ser (re)programable y las funciones lógicas y la E/S también deben ser (re)programables.

Hoy día los principales proveedores de FPGAs son Xilinx, ocupa desde hace unos años el primer puesto, con una cuota de mercado que se acerca al 50 %. Actualmente, esta compañía es el 3er fabricante de ASICs (FPGAs) del mundo, por detrás de IBM y NEC.

Dependiendo de los fabricantes las FPGAs ofrecen más un tipo de prestaciones que otros, de esta forma dependiendo de la aplicación puede interesar más la selección de un fabricante de FPGAs u otro con prestaciones diferentes, más adecuadas a las necesidades del proyecto.

Hoy día las FPGAs han formado parte de la electrónica moderna a modo de complemento de procesadores e incluso sustituyéndolos en ciertas aplicaciones, cada día que pasa se suman nuevos fabricantes de tradicionales procesadores como Intel y Arduino que comienzan a apostar por incorporar las FPGAs a sus dispositivos.

4.1. FPGA XILINX SPARTAN-3E

Se trata de una FPGA de la marca XILINX, la cual cuenta con diferentes gama de FPGA, debido a que la aplicación no precisa de una gran cantidad de recursos dentro de la FPGA y teniendo en cuenta que el precio asciende de una forma exponencial, se han optado por las placas FPGAs más sencillas. Los chips seleccionados cumplen perfectamente con las condiciones tanto de necesidades de células lógicas como de presupuesto.

Además de la FPGA se precisa una placa que cuente con los elementos necesarios tanto para programarla como para el prototipado necesario, como la de la Figura 4.1.1.

Figura 4.1.1: Spartan3 + Placa prototipado

La Spartan-3E con el empaquetado más grande puede contar con hasta 158 pines dedicados para entradas salidas. Para el empaquetado se tendría que seleccionar ademas la capacidad, en este caso existen dos opciones XC3S250E y XC3S500E.

La variación de los números viene dada por el valor de las puertas lógicas disponibles, que para estos casos serán 250 mil y 500 mil. Las puertas no equivalen a unidades de memoria pero aportan una idea aproximada de la capacidad con respecto a placas similares en LUTs la primera placa cuenta con 4.896 LUTs y la segunda con 9.312 LUTs, ademas de contar con RAM

distribuida de 39.168 Bits y 74.496 Bits respectivamente.

Las celdas de entradas/salidas se componen de una tecnología que permite seleccionar el valor de la salida, pudiendo variar éstas en varios niveles dentro de una misma celda y de una misma tensión de alimentación como se muestra en la Figura 4.1.2.

Single-Ended IOSTANDARD	V_{CCO} Supply/Compatibility				
	1.2V	1.5V	1.8V	2.5V	3.3V
LVTTL	-	-	-	-	Input/ Output
LVCMOS33	-	-	-	-	Input/ Output
LVCMOS25	-	-	-	Input/ Output	Input
LVCMOS18	-	-	Input/ Output	Input	Input
LVCMOS15	-	Input/ Output	Input	Input	Input
LVCMOS12	Input/ Output	Input	Input	Input	Input

Figura 4.1.2: Tensión I/Os XILINX

Las entradas tienen funciones de retardo de forma que se puede introducir de forma diferencial utilizando pines adyacentes mediante configuración, esto no es posible para todos los pines. Las entradas también permiten el uso de entradas retrasadas pudiendo desfasar una entrada con respecto a la real hasta un máximo de 12 pasos.

Las FPGAs de XILINX están basada en CLBS (celdas lógicas configurables) y permiten interconexionar funciones lógicas con funciones vecinas o de otros puntos de la FPGA. Las CLBs se componen de 4 Slices existen dos tipos de slices el SLICEL y el SLICEM. El primer tipo incorpora 2 LUT4s, dos generadores de acareo, dos multiplexores y dos Registros. El segundo tipo, además de los bloques del primero, añade dos bloques de RAM16 y dos de SRL16. El bloque de RAM16 cuenta con 16X1 registros tipo D y por lo tanto cuenta con 8 entradas para seleccionar entre los diferentes registros, ya que pueden realizar hasta 2 lecturas simultáneas además tiene las entradas de lectura-escritura, solo dispone de una escritura. El bloque SRL16 es un registro de desplazamiento de 16 elementos, que también permite la lectura de un término mediante direccionamiento, este bloque muestra de forma constante el ultimo y el direccionado

por las entradas.

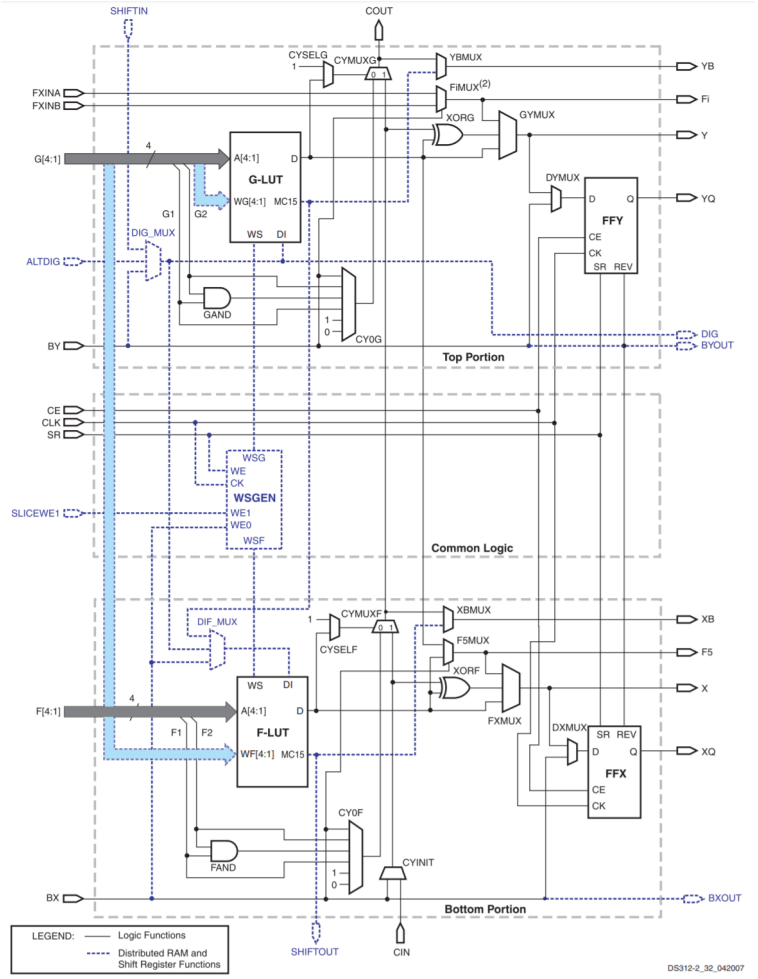


Figura 4.1.3: SLICEL

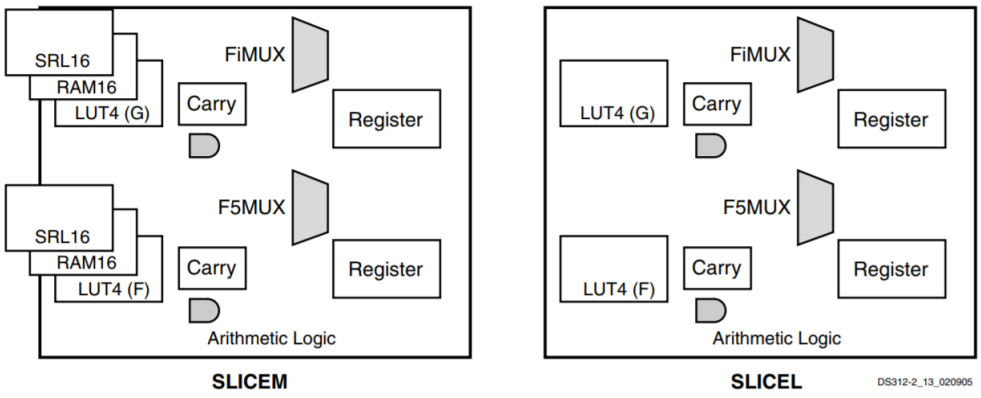


Figura 4.1.4: SLICEL y SLICEM

La FPGA cuenta con bloques para funciones especiales como el DLL (Delay-Lock Loop), que permite generar un reloj desfasado 0º, 90º, 180º, 270º con la

misma frecuencia que la entrada. También es posible generar un reloj con el doble de la frecuencia en fase con el reloj de entrada y el mismo reloj desfasado 180º. El DLL también puede dividir el reloj en frecuencias menores.

Los voltajes de alimentación son de como máximo 1.32V y de 1.20V de forma normal. El voltaje de alimentación para las salidas deberá de ser como máximo de 3.75V, la tensión máxima que soporta por pin de entrada y salida es 100mA.

Una vez comentado brevemente de las características hardware del dispositivo será necesaria una pequeña revisión del software que proporciona el fabricante para la programación de sus FPGAs

El fabricante Xilinx cuenta además con herramientas software como:

SDAccel para el desarrollo permitiendo programar procesadores en C, C++ y OpenCL, también permite la programación de FPGAs mediante lenguajes RTL (VHDL y Verilog).

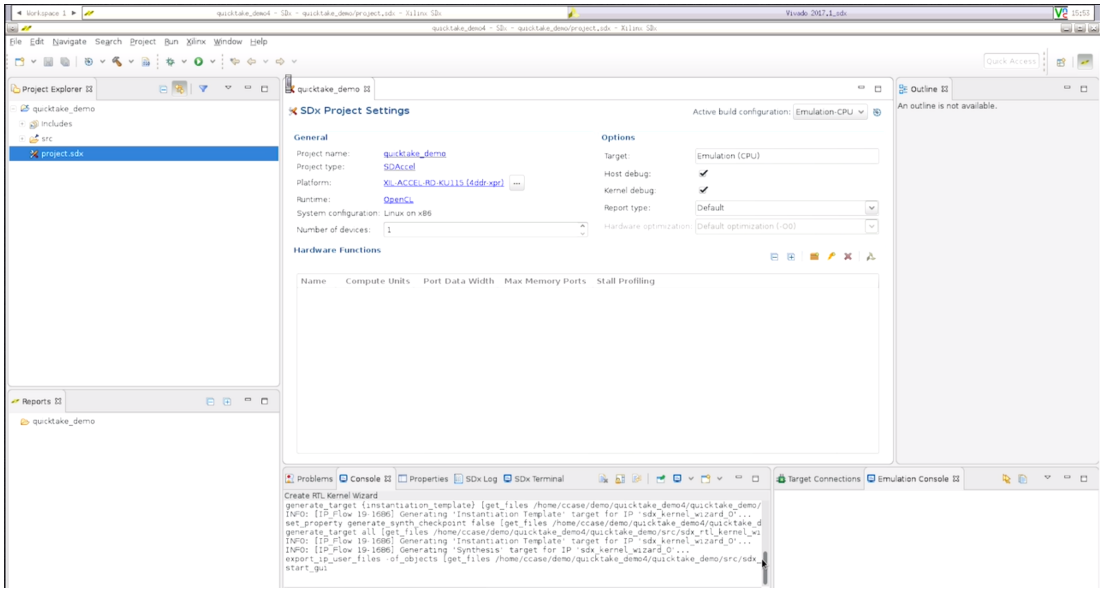


Figura 4.1.5: Entorno SDAccel

SDSoC se trata de un entorno de desarrollo que proporciona un entorno de desarrollo para aplicaciones en C, C++, OpenCL con Eclipse IDE fácil de usar y un entorno de diseño integral para MPSoC. Las bibliotecas de Xilinx OpenCV permiten integrar funciones de OpenCV optimizadas para hardware.

SDNet es un entorno de desarrollo para la comunicación con FPGAs y dispositivos SoC, permitiendo la creación redes hardware aceleradas mediante descripciones software. Incorpora una herramienta software que soporta el paquetede procesamiento del lenguaje moderno P4.

Aunque la familia de FPGA de Xilinx es una alternativa válida para la realización del TFG, no es la que se ha seleccionado, ya que en el Departamento de Tenconogía Electrónica se dispone de las herramientas y placas de prototipos de Lattice, que se han considerado más adecuadas para esta aplicación.

4.2. FPGA Lattice MACHXO2

La MACHX02 elegida será una familia de FPGAs muy flexible permitiendo tener muy bajos consumos en ellas y un relativo gran rango de LUTs, las cuales cuentan con memoria RAM distribuida, memoria Flash para el usuario (UFM), un generador de frecuencias en lazo (PLL), Entradas salidas síncronas predefinidas y así como protocolos permiten generar un controlador SPI, controlador I2C y contadores y temporizadores.

La FPGA seleccionada de la familia es la X02-1200ZE con 1280 LUTs disponibles para ser programadas y una memoria RAM distribuida de hasta 10Kbits. El empaquetado será el más grande disponible 20x20 mm, este empaquetado presenta 144 pines, de los cuales 107 son utilizables por el usuario para el desarrollo.

Lattice también utiliza Slices, aunque estos están compuestos por 2 LUTs con generadores de acarreo, Multiplexores para funciones de 5 variables y dos Registros, que pueden funcionar como Latch o como Flip-flops.

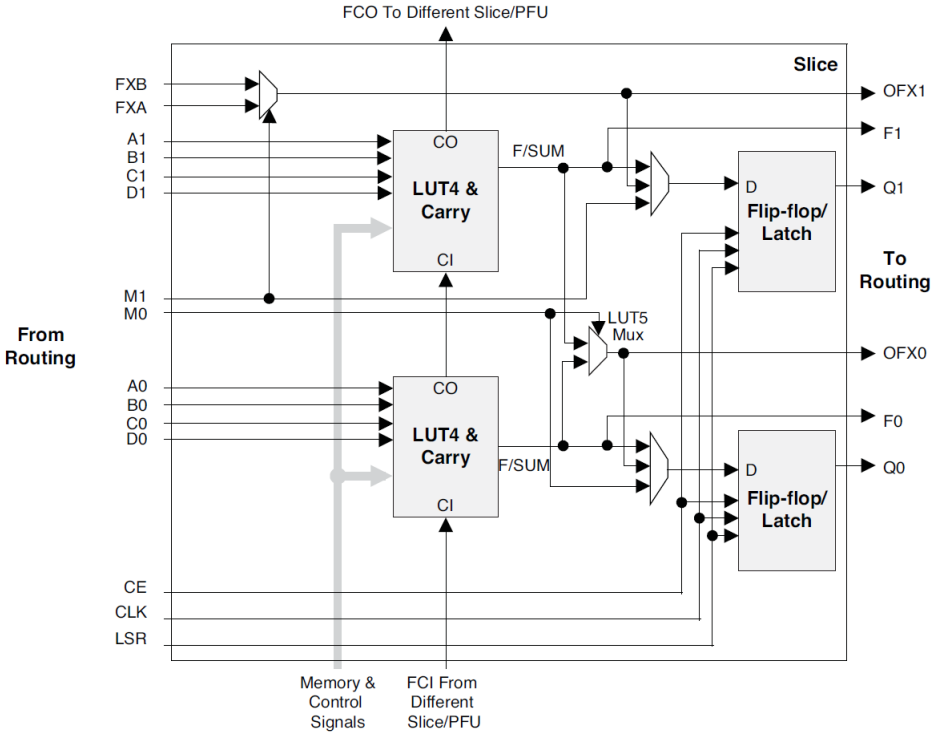


Figura 4.2.1: Slice Lattice

Lattice incorpora en sus FPGAs un PLL que permite generar un gran número de frecuencias a partir de una frecuencia de entrada, pudiendo generar hasta 4

Relojes de frecuencias siendo múltiplos unas de otras, ya que para generar una frecuencia muy baja necesita utilizar los relojes intermedios como recirculación.

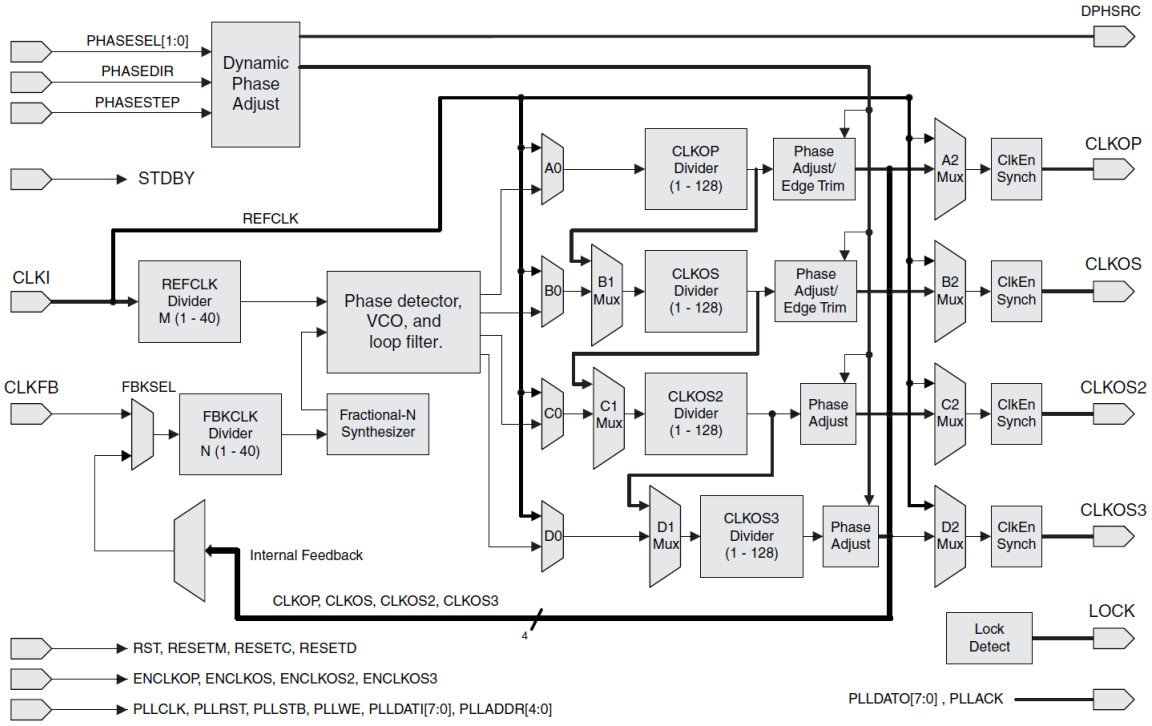


Figura 4.2.2: PLL Lattice

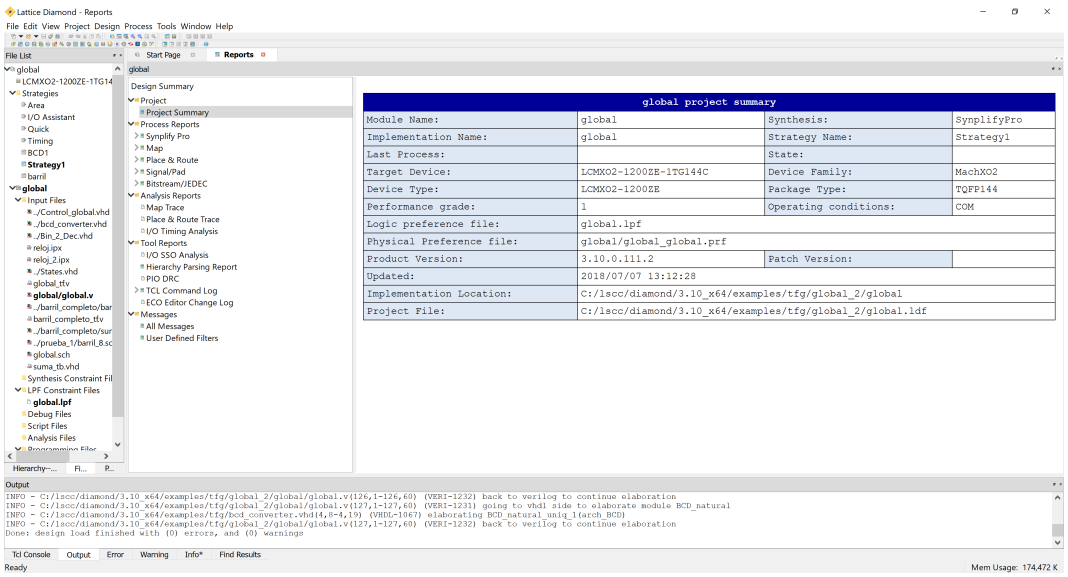
Las FPGAS de Lattice cuentan con varios tipos de memoria RAM interna. Single permite una lectura y una escritura. True Dual permite lecturas y escrituras simultaneas. Pseudo Dual que permite una lectura y una escritura de forma registrada. FIFO asemeja el comportamiento de una cola. ROM solo permite la lectura.

Los Pines de entrada salida pueden ser configurados con diferentes niveles de tensión de forma similar a XILINX, con valores que pueden ir desde los 3.3V hasta los 1.2V. La FPGA cuenta con un Oscilador interno de 2.08MHz este se puede configurar en varias frecuencias hasta llegar a 133MHz.

El fabricante incorpora dos núcleos I2C IP el primero se encuentra preasignado a unos pines específicos y el segundo se puede utilizar en cualquier pin que se designe para ello. También cuenta con un puerto SPI, contadores/Timer y el interfaz Wishbone para la programación mediante el interfaz de 8-bits.

Lattice también dispone de una herramienta software para la programación de las FPGAs, esta herramienta se denomina Lattice Diamond y es una herra-

mienta para la programación en lenguajes VHDL y Verilog además de ofrecer la posibilidad de realizar esquemáticos en los que se genera de forma automática el código para la compilación posterior.



global project summary			
Module Name:	global	Synthesis:	SynplifyPro
Implementation Name:	global	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMXO2-1200ZE-1TG144C	Device Family:	MachXO2
Device Type:	LCMXO2-1200ZE	Package Type:	TQFP144
Performance grade:	1	Operating conditions:	COM
Logic preference file:	global.lpf		
Physical Preference file:	global/global_global.prf		
Product Version:	3.10.0.111.2	Patch Version:	
Updated:	2018/07/07 13:12:28		
Implementation Location:	C:/lscc/diamond/3.10_x64/examples/tfg/global_2/global		
Project File:	C:/lscc/diamond/3.10_x64/examples/tfg/global_2/global.ldf		

Figura 4.2.3: Entorno Lattice

El software cuenta con programas auxiliares que permiten ejecutar funciones complementarias para la programación de las FPGAs de esta forma se completan las funciones que no lleva a cabo diamond, que básicamente se encarga de organizar todos los ficheros, comprobar errores y hacer de enlace para los demás programas.

Aunque no se puede considerar un programa como tal, ya que se ejecuta dentro de Diamond sin necesidad de salir del entorno de diseño, hay una herramienta de gran importancia a la hora de la programación en Lattice, se trata del IPexpress. Mediante esta herramienta se pueden generar gran cantidad de bloques funcionales ya preconfigurados por el fabricante de forma que son bloques ya optimizados y permiten la configuración de estos en base a las necesidades de programación. Dentro de esta herramienta se puede configurar tanto el generador de frecuencias interno (PLL) como contadores, temporizadores, multiplexores, sumadores... y más bloques de uso común a la hora de desarrollar un código descriptivo hardware.

Aldec es la herramienta que se utiliza para realizar las simulaciones del código descrito de forma que permite comprobar el código descrito en Diamond

y chequear que cumple con las especificaciones fijadas en un principio o ver que cambios son necesarios para cumplirlas. Como parte de la simulación se pueden simular las entradas de forma que pueden ser valores fijos, valores generados por un reloj, valores que se reciben de un contador o incluso valores pseudoaleatorios.

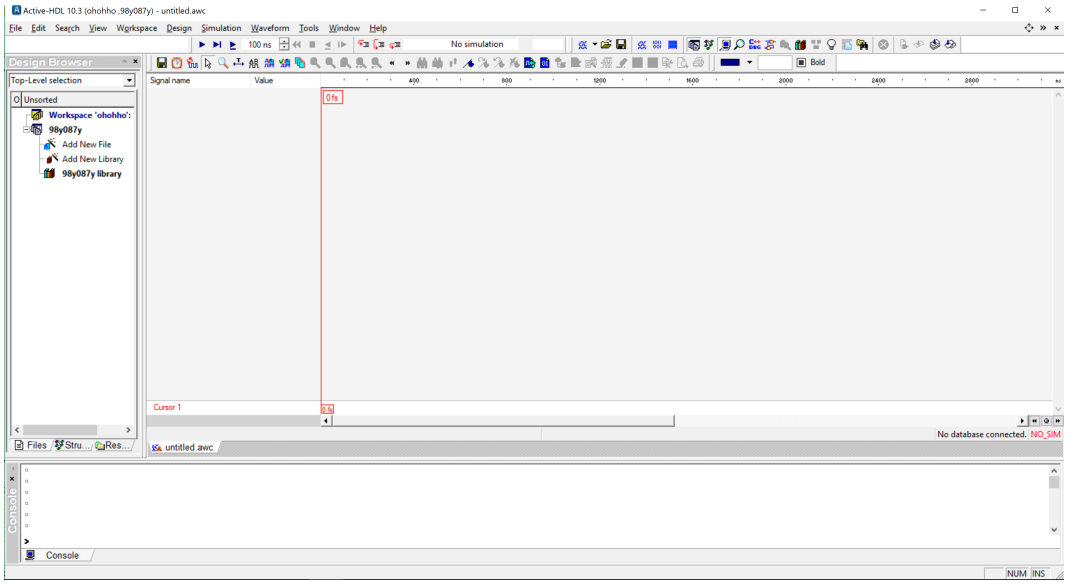


Figura 4.2.4: Entorno Aldec

Simplify-pro es una herramienta de sintesis que se ejecuta cuando se pretende generar un fichero exportable a otro formato ya sea código para simulación o para exportarlo a una FPGA real como tal, el programa no se ejecuta directamente, aunque se puede correr de forma manual, pudiendo modificar parámetros para una mejor optimización, si no se conoce lo suficiente es conveniente no ejecutarla en modo manual y utilizar la configuración que le proporciona Diamond.

Una vez generado el archivo «.jedec», se tendrán que configurar los pines que corresponderán con las salidas y entradas de acuerdo al modelo teórico y al modelo real, para ello se utiliza otra herramienta dentro de Diamond el Spreadsheet View que permite configurar los pines de salida, los pines pueden soportar multiples configuraciones, no unicamente como entrada/salidas sino también pueden variar la tensión de salida que estos presentan así como la configuración de las mismas pudiendo tener propiedades distintas, siendo los mismos valores de tensión.

Para realizar la conexión de la FPGA con el ordenador es necesaria utilizar la aplicación Programmer para iniciar el traspaso de información.

Como este tipo de FPGA ha sido el que ha sido utilizado durante la asignatura Sistemas Reconfigurables se ha seleccionado para llevar a cabo este TFG, ya que se conocía la tecnología previamente y el entorno de trabajo facilitando el desarrollo del mismo al evitar tener que aprender a manejar un nuevo entorno gráfico y configuraciones que supondría utilizar un entorno nuevo, reduciendo el tiempo necesario para realizar el TFG.

4.3. Placas Auxiliares

El proyecto como tal necesita una forma para la visualización de los datos ya que estos deben ser interpretados por una persona física, aunque bien podrían llevarse a una base de datos, pero se trataría de un proyecto a parte. La visualización de los datos se realiza mediante dos displays de siete segmentos, con sus respectivas resistencias para limitar la corriente de los leds poder conectarlos mediante lógica negativa y los bits que cuenta la placa base de la FPGA. En la parte superior cuenta con 5 pines para conectar módulos si fuera necesario, contando con dos pines para alimentación y común y otros 3 pines adicionales para enviar y recibir información en la FPGA.

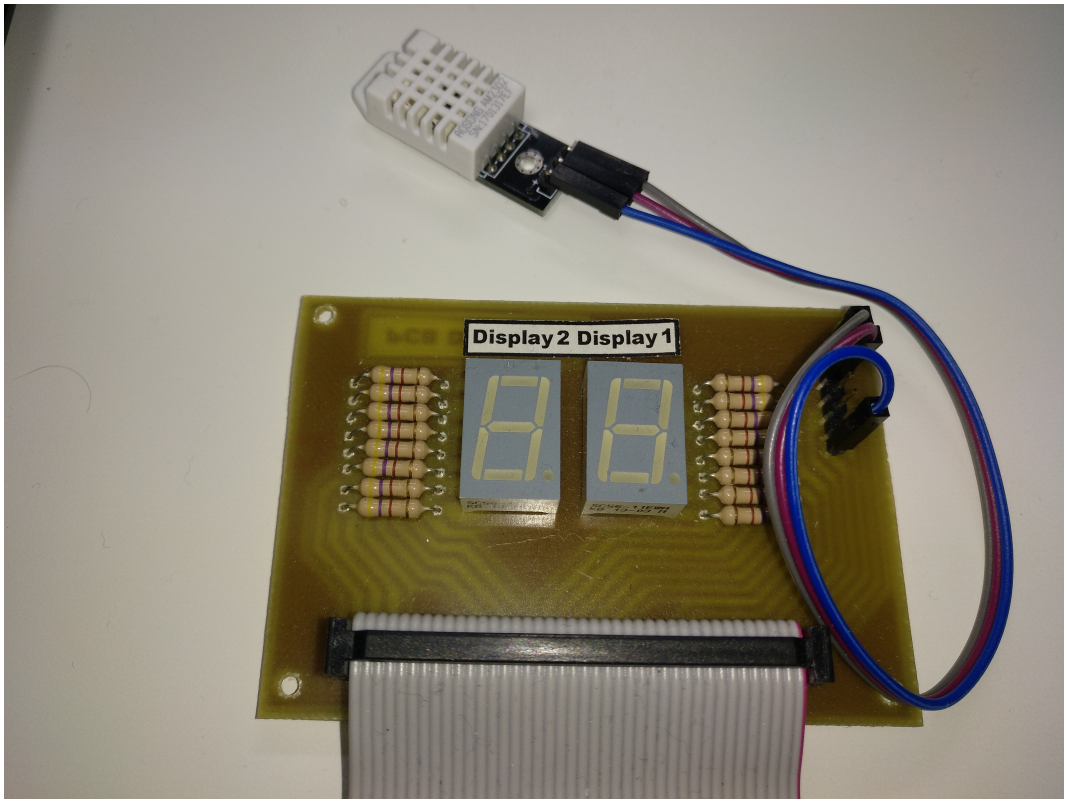
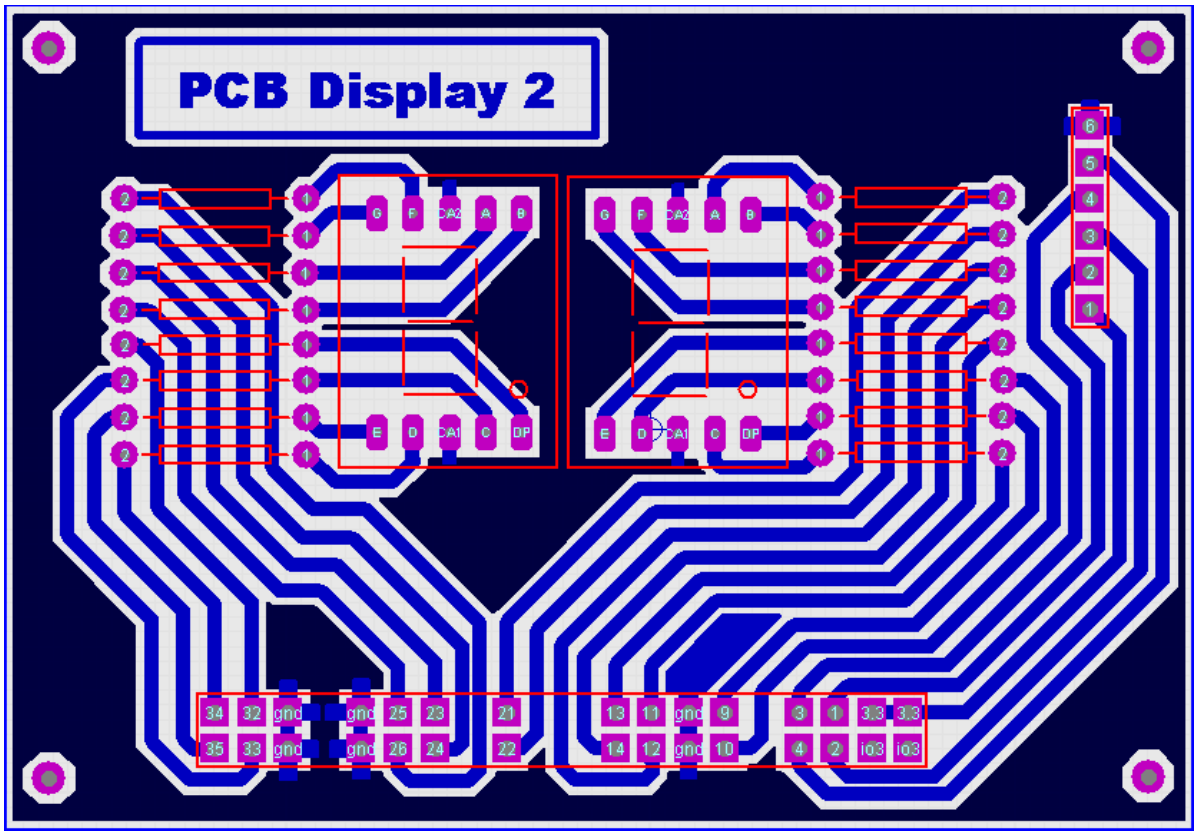


Figura 4.3.1: Placa Superior

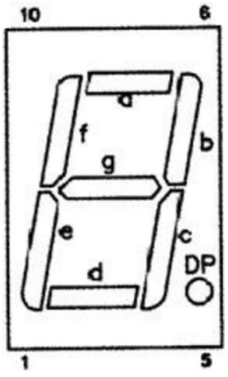
PLACA PCB DISPLAY 2



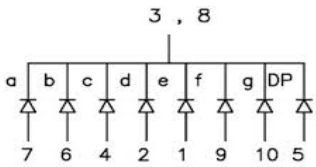
PINOUT DE LA FPGA ASIGNADO EN LA PLACA PCB DISPLAY 2:

DISPLAY 2 (A LA IZQDA)

A2	PIN7	23
B2	PIN6	25
C2	PIN4	33
D2	PIN2	34
E2	PIN1	35
F2	PIN9	26
G2	PIN10	24
DP2	PIN5	32



SC56-11



DISPLAY1 (A LA DRCHA)

A1	PIN7	21
B1	PIN6	22
C1	PIN4	11
D1	PIN2	10
E1	PIN1	13
F1	PIN9	12
G1	PIN10	14
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

En la selección de los datos a mostrar se utilizará otra placa auxiliar, esta placa contiene 8 pulsadores y un Switch con 9 interruptores, estos se utilizarán para seleccionar la información a mostrar por parte de la FPGA. Para este proyecto no se utilizan los botones ya que para cambiar la información que se muestra se ha preferido utilizar el switch, ya que este mantiene la información un tiempo indefinido. Si se usaran los botones para desempeñar la misma función habría que mantener pulsados los botones o bien recordar el estado de estos mediante memorias internas. El utilizar los botones impide conocer en que configuración se encuentra el dispositivo y por lo tanto se descartó su uso.

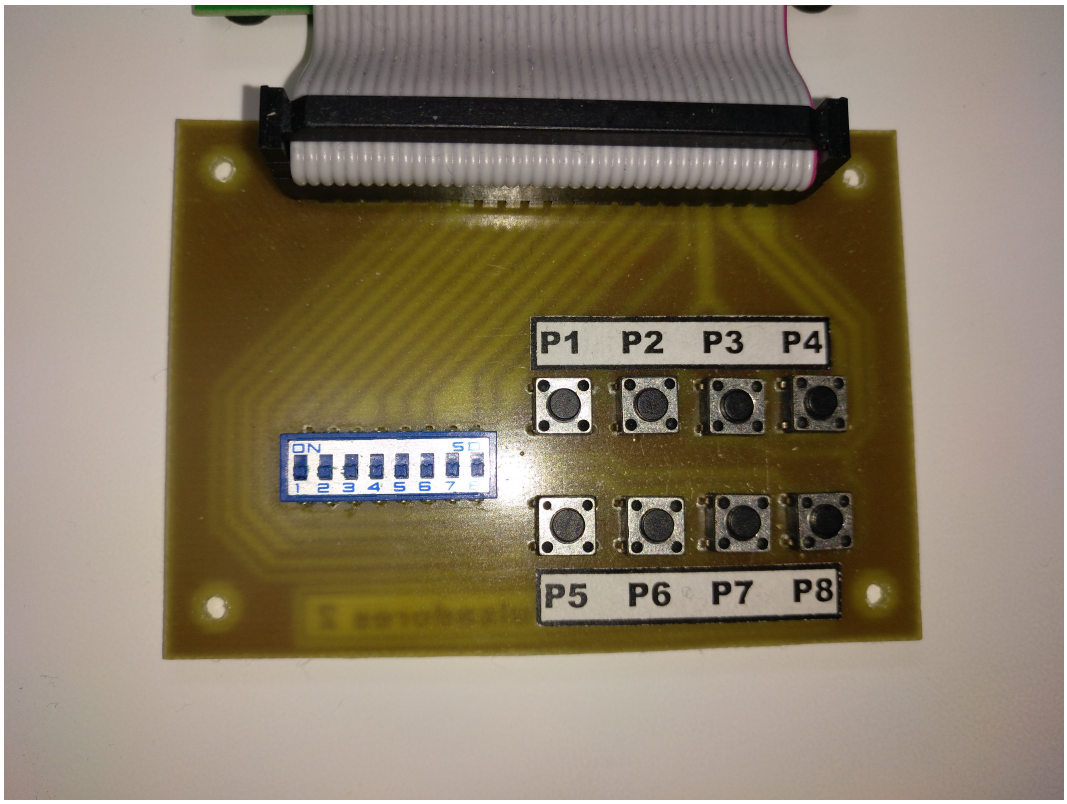
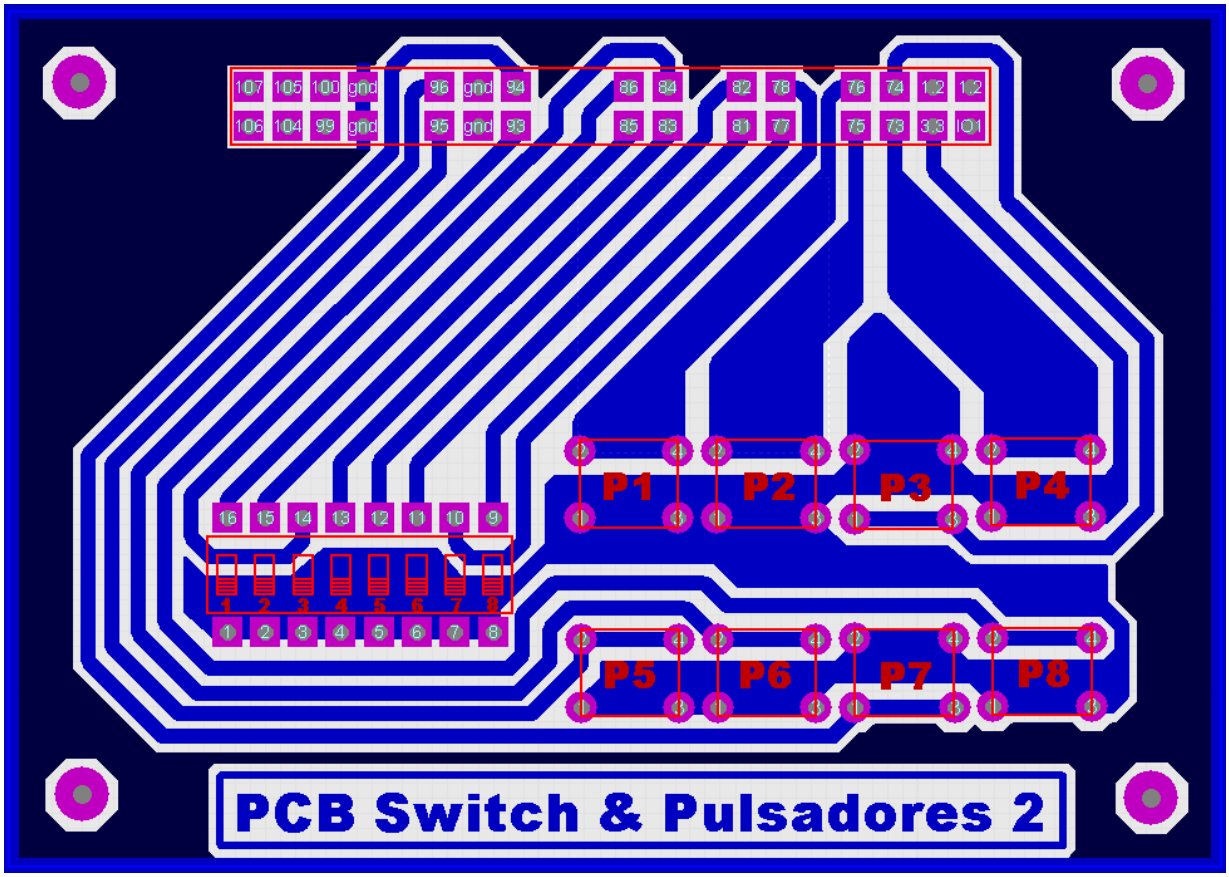


Figura 4.3.2: Placa Inferior

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).

4.4. Analizador Lógico

La programación siempre necesita un medio por el cual probar el funcionamiento de los bloques realizados para esta Tarea el conjunto global es demasiado grande como para probarlo en el Aldec de forma que la depuración de errores en la programación tiene problemas a la hora de probarlo en el sistema real, ya que no es posible visualizar las variables.

La solución al problema es la utilización de un módulo analizador lógico de señales digitales, que permite visualizar hasta 8 canales digitales unicamente conectando los pines, por lo que se pueden utilizar pines para mostrar las comunicaciones entre etapas intermedias y señales que quedarán ocultas en el desarrollo final.

Mediante el analizador lógico se pueden muestrear señales con una frecuencia máxima de 24MHz, esto limitará algunas aplicaciones, aunque si es posible siempre se puede disminuir la frecuencia y poder conectar el analizador lógico.

Figura 4.4.1: Analizador Lógico

El módulo genera las señales una frecuencia variable pero siempre es inferior a 20mS por lo tanto se puede utilizar el analizador lógico de la señales que genere el módulo y las etapas posteriores ya que estas no tendrán un periodo superior al del módulo.

Capítulo 5

Hardware Sensor y Comunicación

La toma de valores necesita un hardware específico que pueda ser controlado por la FPGA de forma que nos permita obtener medidas del entorno. Existen gran cantidad de variables que determinan la meteorología y por lo tanto una gran cantidad de sensores. Por lo tanto será necesario ver las diferentes opciones que presenta el mercado. El objetivo de este trabajo es realizar una pequeña estación meteorológica, para la determinación de un par de variables. El incremento del número de sensores a controlar, podría ser objeto de trabajos posteriores.

Será necesario destacar que por temas hardware la FPGA elegida carece de módulos convertidores de señales analógicos a digitales por lo que a la hora de seleccionar el sensor éste tendrá que tratarse de un sensor preferiblemente digital, o en caso de tratarse de uno analógico contar con un módulo que proporcione la conversión.

Al seleccionar un sensor digital es posible encontrarlo de dos tipos: Digitales que trasmiten de forma ininterrumpida y los llamados «Smart Sensor». Los primeros envían la información sin procesar de forma que es necesaria procesarla en la FPGA, aumentando la complejidad del proyecto. Los segundos pre-procesan la información para que sea más sencilla su utilización en posteriores aplicaciones.

En los siguientes subapartados se procederá a analizar las características de

una serie de sensores que miden variables ambientales y que son apropiados para la estación meteorológica.

5.1. Sensores Analógicos

5.1.1. LM35

Se trata de un sensor de temperatura Analógico basado en Amplificadores operacionales y transistores para generar una fuente de Tensión dependiente de la temperatura, esta fuente se podrá considerar prácticamente lineal durante el rango de temperaturas de -55ºC a 150ºC con una precisión de un grado durante prácticamente todo el rango.

El sensor como tal presenta consumos bajos,inferiores a 1mA , para los niveles más altos de temperatura.

La precisión del sensor se situaría entorno a ±1ºC.

La principal desventaja de este sensor es que se trata de un sensor Analógico y por lo tanto sería necesario añadir un convertidor analógico-digital para poder utilizarlo con la FPGA.

Para poder medir temperaturas por debajo del 0ºC el voltaje debería ser negativo de forma que habría que añadir un circuito auxiliar para poder invertir la tensión que aplicamos al dispositivo.

También, como todos los sensores analógicos, son sensibles al ruido.

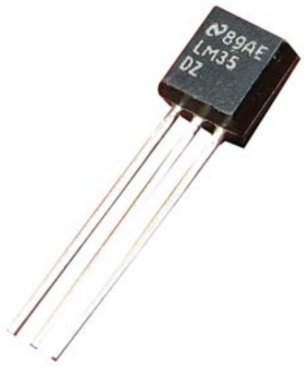


Figura 5.1.1: LM35

5.1.2. LM36

Es similar al LM35 solo que no precisa de un circuito auxiliar cuando queramos medir temperaturas bajo cero, ya que su ventaja principal es que puede medir rangos negativos con una tensión positiva de alimentación, por lo que no será

necesario ningún circuito inversor. Al utilizar una sola fuente en un sentido la precisión del sensor disminuye hasta tener un rango de los ±2ºC.

Figura 5.1.2: LM36

5.1.3. YL-83

Se trata de una placa sensible a la lluvia de forma que, cuando se moja esta placa provoca cortocircuito entre las pistas, utilizando un módulo específico permite generar dos tipos de señales, una digital que se activa cuando detecta una gota sobre la placa y otra analógica que cambia en función de las gotas que caen sobre el dispositivo. La desventaja principal del dispositivo es su alto precio.

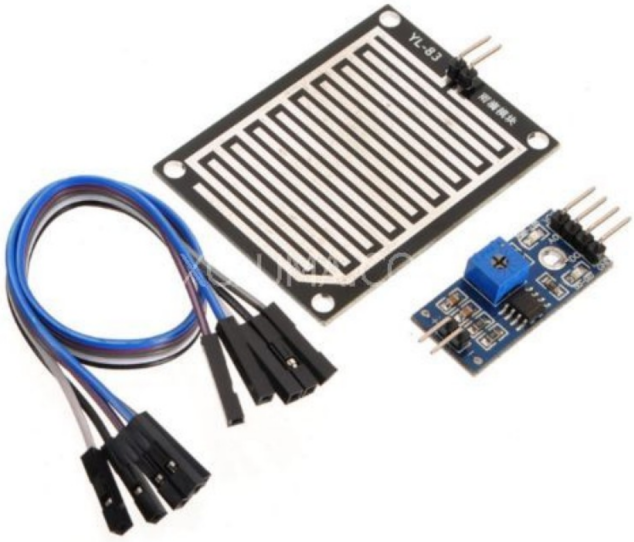


Figura 5.1.3: YL-83

5.2. Smart Sensors

En este apartado se comienza con los sensores, que realizan un pretratamiento de la información, antes de enviarla.

5.2.1. TC74

Se trata de un sensor de temperatura digital este tiene ventajas con respecto a los analógicos como que son mucho menos sensibles al ruido. El sensor proporciona un valor de temperatura de -40ºC a 125ºC con una resolución de 8bits por muestra proporcionando unas 8 muestras por segundo. La precisión del sensor es alrededor de ±2ºC, para valores centrados del rango de medidas cuando el rango esta más alejado del centro la precisión disminuye.

La comunicación con el dispositivo se realiza mediante el puerto I2C de forma que permite conectar viarios sensores a la misma linea de datos seleccionando desde el maestro el sensor que va a leer.

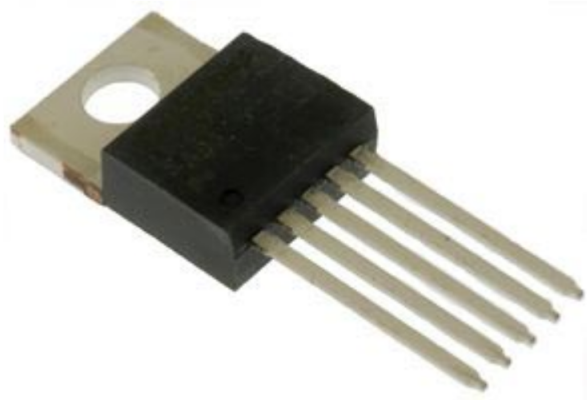


Figura 5.2.1: Sensor TC74

5.2.2. DHT11

El sensor proporciona medidas tanto de temperatura como de humedad. El sensor presenta ventajas con respecto al TC74 comentado anteriormente ya que este no se controla mediante I2C sino a través de una señal especial que proporciona el maestro para que comience a trasmitir. También presenta la desventaja de que en una línea solo será posible conectar un sensor de forma que se necesita una línea por cada sensor, aunque menor complejidad a la hora de comenzar la trasmisión.

El sensor cuenta con una precisión para la temperatura de ±2ºC y ±5 % para la humedad relativa.

El consumo es de como máximo 2.5mA, produciéndose este cuando inicia la trasmisión mientras que cuando permanece sin ser llamado el consumo es de 100µA. El sensor necesita cada vez que produce una lectura al menos un tiempo de un segundo para poder realizar una nueva lectura.

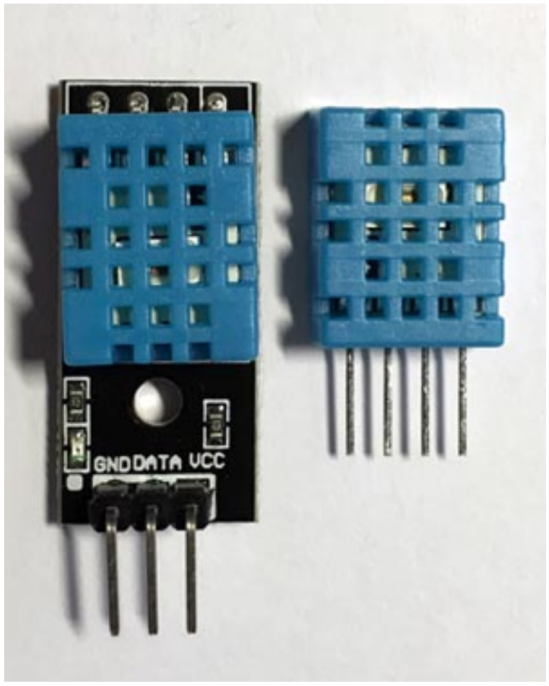


Figura 5.2.2: DHT 11

5.2.3. DHT22

Se trata de un sensor muy parecido en el funcionamiento al DHT 11, esto es debido a que uno es la versión superior al otro y el DHT22 lo es. El sensor proporciona 16 bits de información para los valores de temperatura y humedad relativa, lo que destina el doble de bits que el sensor más pequeño para las dos medidas. A diferencia del otro sensor necesitará 2s para refrescar las medidas. Las medias tienen una precisión de ±0.5ºC y un ±3 % de humedad relativa. El rango de medidas para humedad relativa es completo aunque a costa de reducir la precisión y de temperaturas de -40ºC a 80ºC.

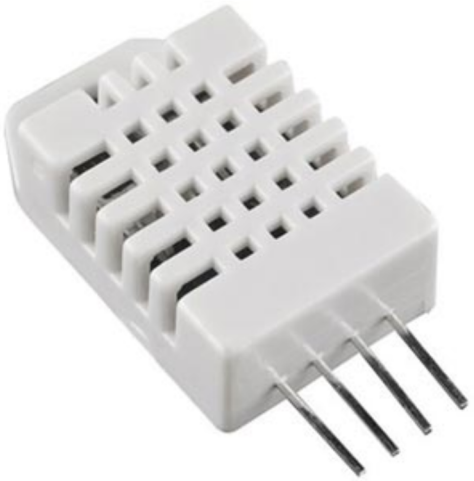


Figura 5.2.3: DHT 22

5.2.4. SHT 15

Es un sensor humedad y temperatura que envía 14 bits por cada variable medida, la comunicación se produce mediante I2C. La precisión de este sensor es muy alta ofreciendo ±0.3ºC y ±2 % de humedad relativa para los valores centrales del rango de medidas. El rango de temperaturas es desde -40ºC a 123,8ºC, y la humedad relativa se puede medir durante todo el rango. Es un sensor especializado para variaciones altas en las medidas. Los consumos son de 2µW mientras este esperando y 3mW cuando se produzca la medición. La principal desventaja de este sensor es el precio ya que es muy elevado.

Figura 5.2.4: SHT 15

5.2.5. BMP 180

El sensor mide temperaturas y presiones atmosféricas proporcionando de 16 a 19 bits dedicados a las presiones y 16 bits dedicados a la temperatura. La comunicación con el sensor se produce mediante I2C y además permite diferentes modos de funcionamiento, dependiendo si se desea un bajo consumo o una gran precisión, esto se realiza de forma interna aumentando el número de muestras por segundo que se producen para la medición. Los rangos de medida para presiones desde 300 hasta 1100 hPa, para la temperatura sera de -20ºC a 65ºC. Los consumos en el modo de bajo consumos son de 3µA y de 12µA.

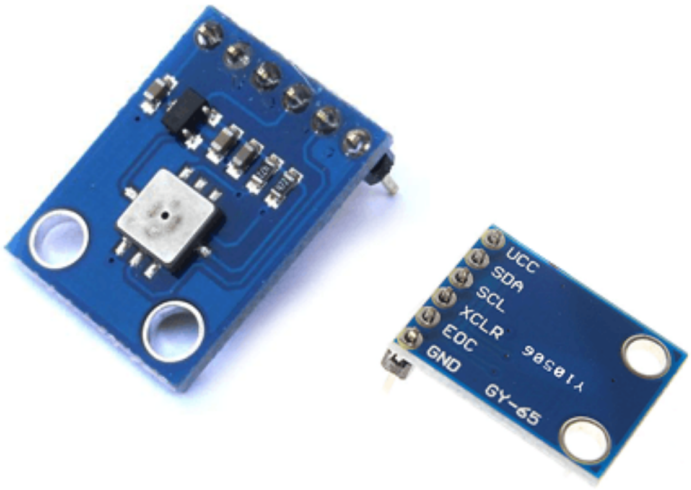


Figura 5.2.5: BMP 180

5.3. Comunicación Bluetooth

A continuación se comentarán los módulos de comunicación vía Bluetooth.

5.3.1. HC-12

Se trata de un módulo de comunicación de nueva generación multicanal embebido. La capacidad de transmisión es de aproximadamente de 5000bps hasta una distancia de 500m. La comunicación se realiza según el estándar: IPEX20279-001E-03 que utiliza 433MHz para el envío de información .El módulo cuenta con dos entradas de antena una con cable coaxial y otra para ir soldada, pudiendo cambiar entre estas dependiendo de las necesidades. El módulo cuenta con cuatro modos de transmisión UART con diferentes corriente de operación.

Dependiendo del modo de comunicación cuanto mayor sea la velocidad de trasmisión de la información que la distancia de recepción disminuye a la mitad de un modo a otro y la sensibilidad al ruido aumenta ligeramente.

El primer modo de comunicación se denomina FU1 modo de reserva de energía, con este modo de funcionamiento consume alrededor de 3.6mA, soportando diferentes modos de funcionamiento de hasta 250000bps, en una comunicación relativamente cercana.

El segundo modo de funcionamiento FU2 es modo de más ahorro de energía con una corriente de 80µA, con velocidades de 1200bps, 2400bps y 4800bps, solo permite la transmisión de pequeños paquetes de datos de 20bytes y los intervalos de trasmisión deberían de ser de más de 2 segundos preferiblemente.

El tercer modo FU3, se trata del modo de fabrica donde la comunicación se realiza a 9600bps 8 bits sin paridad y un bit de parada. En este modo los consumos rondan los 16mA y con un retardo de 4-80mS en la comunicación, siempre dependiendo de la distancia de los dispositivos.

El cuarto modo FU4 es el modo de trasmisión de larga distancia, este modo está fijado para 1200bps. Puede transmitir un conjunto reducido de datos de alrededor de 60bytes y en intervalos de tiempo preferiblemente superiores a 2 segundos y consumos de al rededor de 16mA y un retardo de aproximadamente 1s dependiendo de la distancia

La configuración del módulo se realiza mediante comandos AT . Enviando AT+Bxxxx permite definir la velocidad de trasmisión en bps. El canal de comunicación con el comando AT+Cxxx existiendo un rango de 001 hasta 127 canales diferentes para la comunicación, si se trata de una comunicación

superior a 9600bps el rango se escalona en canales de 5 en 5. Enviando AT+FUx cambiamos de modos definidos por el fabricante. La potencia de trasmisión puede ser ajustada mediante AT+Px con 8 niveles disponibles. Existe un modo para conocer la configuración del módulo enviando AT+Ry (sustituyendo la y por B, C, F, P) para conocerla en cada caso. El módulo puede devolver todos los parámetros, con una instrucción AT+RX devuelve el modo del fabricante, la velocidad, el canal y la potencia, todas las especificaciones van precedidas de un OK.

Existen más comandos para modificar el formato de trasmisión de los bits con AT+Uxxx. AT +V devuelve la versión del dispositivo. AT+SLEEP entra en suspensión y la corriente se reduce hasta 22µA, enviando un carácter AT se reactiva el módulo. AT+DEFAULT devuelve el módulo a los parámetros de fabrica.

Todos los comandos AT generan una respuesta de forma OK más el comando enviado, si simplemente se ha enviado AT como forma de comprobación de errores el módulo reenviará un OK.

Como comunicación entre dispositivos este módulo permite grandes ventajas en cuanto a posibilidades y configuraciones, además de una gran versatilidad, teniendo en cuenta que no necesita corrientes elevadas para funcionar, es el seleccionado para realizar la comunicación entre las dos FPGAs.

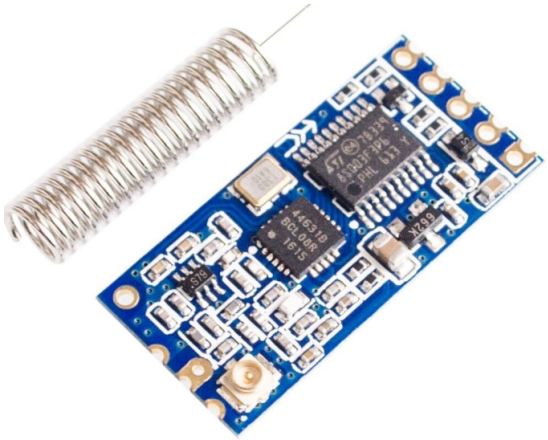


Figura 5.3.1: HC-12

El HC-12 a pesar de ser perfecto para la comunicación con dispositivos no es compatible con dispositivos móviles algo muy deseable hoy día por lo que también habrá que tener en cuenta el siguiente módulo.

5.3.2. HC-05

Es un módulo que utiliza bluetooth para la comunicación por lo tanto es compatible con los dispositivos móviles. El módulo soporta velocidades de trasmisión de 9600 a 460800, pasando por frecuencia intermedias, siendo cada una de estas el doble de la anterior.

El módulo puede ser configurado como Master o como Slave,

Al igual que el módulo HC-12 comentado anteriormente este módulo se puede configurar mediante comandos AT.

AT+RESET resetea el módulo. AT+VERSION devuelve la versión de firmware. AT+ORGL devuelve a estado de fabrica el módulo con contraseña 1234, nombre de dispositivo HC-05 y velocidad de trasmisión 38400bps. AT+ADDR devuelve la dirección Bluetooth del dispositivo. AT-NAME= cambia el nombre del dispositivo, añadiendo en lugar del signo «=» un signo de «?» el dispositivo devolverá el nombre actual. AT+RNAME? devuelve la dirección de Bluetooth del dispositivo. Para configurar el dispositivo AT+ROLE= siendo posible los diferentes modos de funcionamiento: 0 Slave, 1 Master, 2 Slave-loop. También es posible programar el dispositivo para que lea entradas o escriba en las salidas mediante AT+PIO= «pin», «estado» pudiendo ser el estado 0 ó 1.

Al utilizar bluetooth para la comunicación permite utilizar una aplicación usar la pantalla del móvil como visualizador de información.

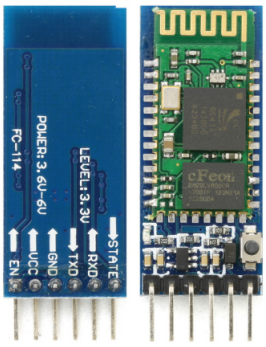


Figura 5.3.2: HC-05

5.4. Conclusión

Al existir un gran número de posibilidades en lo que respecta a los sensores se ha optado por el sensor DHT-22 ya que este proporciona buenas prestaciones a un precio muy asequible. Además para controlarlo será necesario implementar bloques específicos de forma que transformen los datos que nos proporciona el sensor a datos legibles por la FPGA. A continuación se comentará más en detalle el sensor seleccionado.

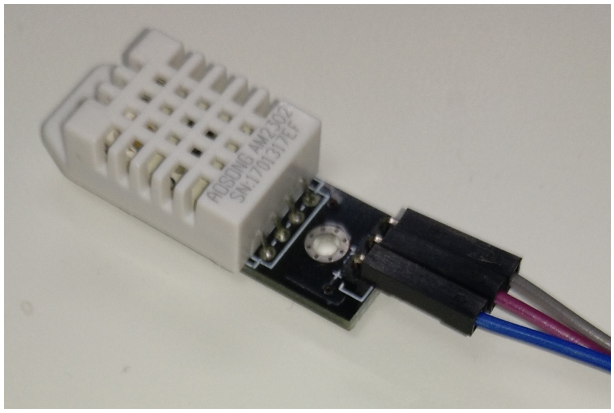


Figura 5.4.1: Sensor Utilizado

El sensor comenzará a emitir cuando reciba una señal procedente del maestro, indicándole que debe comenzar la comunicación de los datos.

La comunicación empezará cuando el maestro pone la línea que comparte con el sensor a nivel bajo, durante aproximadamente 1mS, una vez trascurrido ese tiempo el sensor comenzará a generar los valores para que los interprete el Maestro que permanecerá escuchando hasta que finalice la trasmisión.

El sensor comienza enviando señales previas de sincronización para los dos,. Una vez terminado este periodo comenzará a enviar la información válida. La lectura de los bits dependerá del tiempo que permanezca el sensor a nivel alto, ya que para comenzar a enviar un bit la línea permanecerá a nivel bajo durante unos 50µS, si se trata de un bit 0 la línea permanecerá a nivel alto durante 26µS aproximadamente. Si por el contrario se tratase de un bit de valor 1 la línea tendría que permanecer a nivel alto durante 70µS.

El sensor mandará un total de 40 bits estos están agrupados en 5 grupos de 8 bits cada uno. El primer grupo representa la parte alta del sensor de humedad, el segundo la parte baja para formar una palabra de 16 bits; esta será la lectura

de humedad relativa expresada en entera, a la que habrá que desplazar la coma hacia la izquierda para obtener la lectura real en %. El tercer grupo proporciona la parte alta de la temperatura y el cuarto la parte baja de ésta que también al igual que la humedad forman una palabra de 16 bits a la que habrá que mover la coma para poder obtener la lectura real en ºC de la misma. El último grupo es un conjunto de comprobación de errores, este grupo representa la suma de los 4 anteriores grupos de forma individual, esto se usa a modo de corrección de errores ya que es posible que se haya malinterpretado la información y esta no sea válida, por lo que es responsabilidad del maestro comprobar que la suma de los términos corresponde con el valor de este último grupo.

Teniendo consumos reducidos de 8mA durante el arranque y 20µA mientras permanece a la espera, además el precio de este sensor no es excesivamente alto.

Este sensor ha sido el elegido para la realización de este proyecto ya que tiene una forma fiable y no muy compleja de comunicación y ofrece buenas prestaciones en las medidas a un precio asequible.

Al utilizar un protocolo de comunicación propio proporciona ventajas e inconvenientes al compararlo con otros modos de trasmisión de información, como protocolos I2C.

Ventajas:

- Al utilizar una línea de conexión por dispositivo impide que se produzcan interferencias entre estos al enviar más de uno la información.
- Al disminuir las interferencias la posibilidad de errores disminuye al solo poder producirse estos por ruido.
- El protocolo propio permite implementar las mejoras necesarias por parte del desarrollador de dicho protocolo que más se adapten a su dispositivo y a las necesidades de este.

Inconvenientes:

- Utilizar una línea de interconexión por dispositivo el número de cables a utilizar aumenta el precio y también si los sensores están alejados de la fuente de procesamiento.

- Utilizar un protocolo propio obliga a desarrollar funciones específicas para ese dispositivo y por lo tanto implementar una función por cada dispositivo nuevo que no tenga ese protocolo. Utilizando un protocolo común para todos los dispositivos permite con una función conectar todos los dispositivos.

Capítulo 6

Elaboración del Sistema

A continuación se procederá a comentar la generación de la configuración del dispositivo en la parte de bloques, describiendo el funcionamiento de estos y como se han elaborado.

Como visión global del funcionamiento del dispositivo, la FPGA recibirá los datos procedentes del sensor con el protocolo propio del sensor que se comentará en una sección específicamente dedicada a ello. En la FPGA Habrá que implementar bloques específicos para procesar y almacenar la información procedente del sensor.
Una vez la FPGA tenga los datos en un formato más cómodo para su manipulación tendrá que mostrar la información en un formato para el usuario. El usuario verá la información en formato BCD, ya que es el formato natural para expresar los números.

Para la generar la configuración se ha utilizado el programa Lattice Diamond, ya que el fabricante proporciona una versión gratuita del mismo para estudiantes a través de la pagina web oficial. La versión utilizada es la 3.10 del mismo. El software permite la programación en diferentes lenguajes, VHDL, verilog, además de un modo esquemático donde se genera de forma autónoma el código. El propio programa cuenta con programas auxiliares de cara a facilitar el chequeo de los posibles defectos en la programación.

El *Simplify Pro* permite generar los ficheros exportables a las FPGAs además de ficheros auxiliares de cara a simulaciones y también permite generar exportables a otros formatos de programación similares.

El *IPexpress* permite generar de una forma rápida e intuitiva bloques con funciones bastante comunes ya desarrollados por el propio fabricante, permitiendo generar contadores con múltiples funciones, sumadores, multiplicadores y permite configurar la herramienta de PLL interna para la generación de relojes con distintas frecuencias dentro de la propia FPGA.

El *Active-HDL* permite realizar simulaciones de los archivos de código generado en Lattice, pudiendo manipular las entradas necesarias a cada bloque obteniendo así una comprobación de que las funciones implementadas son correctas al funcionamiento de la aplicación.

El programa ofrece más soluciones pera la implementación en los dispositivos, de cara a solución de errores más específicos, como no han sido utilizados de cara al desarrollo de este trabajo, no serán comentados.

También se han utilizado programas externos para generar el diagrama de estados ya que Lattice Diamond no ofrece un programa para ello. El software utilizado es el Qfsm que permite generar diagramas de estados en diferentes formato Melay y Moore, además de generar un archivo exportable a otros formatos.

El chequeo de las señales generadas hacia el exterior, por parte de la FPGA se ha realizado con el analizador lógico Saleae Logic, que permite analizar los valores lógicos que toma una salida durante un periodo de tiempo determinado.

6.1. Esquema General

La descripción del sistema que se construye es la siguiente, la estación meteorológica se ha realizado mediante el uso de un conjunto de bloques, donde cada uno de los cuales realiza funciones simples, de esta forma que simplifica la resolución del problema. El esquema general se ha dividido en diferentes partes, de forma que sea más visual su comprensión al reducir los elementos que se presenta en cada imagen. A continuación se comentará cada conjunto de bloques y las funciones que realiza cada uno de forma global para ofrecer una descripción del conjunto.

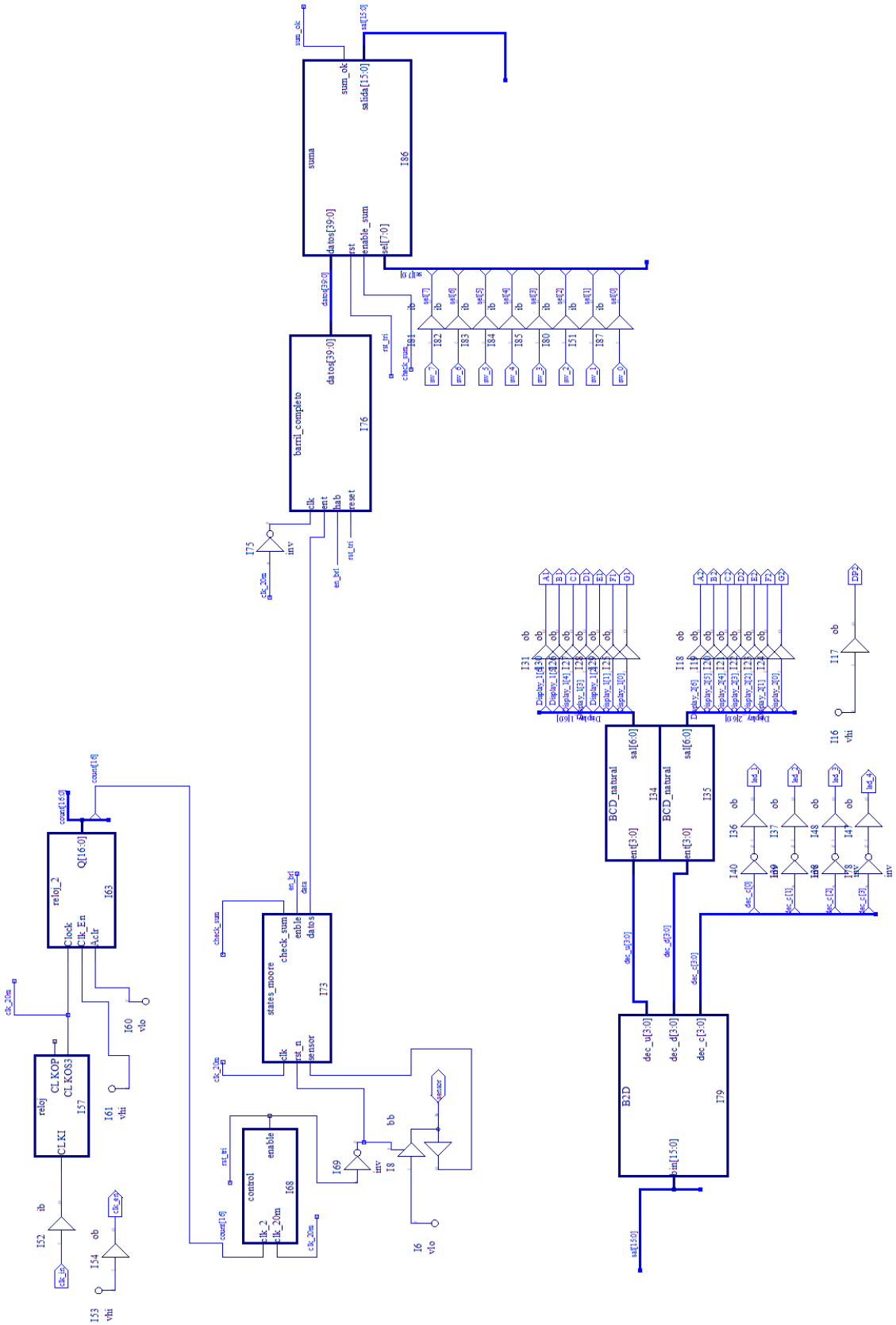


Figura 6.1.1: Esquema General vista Global

La Figura 6.1.1 representa el conjunto general de la estación meteorológica con todas las entradas y salidas que se utilizan tanto para la representación de los valores deseados como para la selección de los mismos y la lectura del sensor de temperatura y humedad.

Se pueden apreciar las entradas que será una destinada al sensor en la parte izquierda a media altura, en la parte sueprior izquierda se generan los relojes que que se utilizarán por todo el circuito, utilizando una salida y una entrada para generar el reloj de 50MHz que alimenta los circuitos generadores de frecuencias.

La parte del sensor se utiliza una estada/salida con un buffer bidireccional que estará conectado al pin común que utiliza el sensor para la comunicación de los datos con el dispositivo de control.

Una vez se comiencen a recibir los datos estos pasarán por una máquina de estados para ser interpretados y después almacenados en un barrel-shifter de 40 bits.

Cuando los datos ya estén transformados y almacenados entonces se tendrá que comenzar el procesado de la información para esto existe un bloque que la procesa y en función de 8 entradas de selección que se conectarán a la placa para que la selección de la información a mostrar por los displays.

Para mostrar la información por los displays será necesario volver a tratarla, por lo que la información pasará por un convertidor de binario a BCD.

Ya para mostrarse por el display necesita una transformación más de BCD a 7 segmentos, como la placa no dispone de 3 displays se ha optado por mostrar los bits más significativos por los leds auxiliares de la placa, utilizando cada display 8 pines de salida 7 para la representación de los números y uno más para el punto que ilumina para mostrar donde está la coma, (el otro display también se ilumina porque esta conectado con la habilitación del oscilador de 50MHz. Los leds utilizados para los bits más significativos son cuatro, mostrar el resultado en BCD directamente.

6.1.1. General I

La Figura 6.1.2 contiene la primera parte del esquema general donde se generarán los relojes que posteriormente serán necesarios en los bloques posteriores,

ya que será necesario un reloj con un periodo de 20µs y otro con un periodo de 2s, para completar el correcto funcionamiento.

El reloj de frecuencia menor se generará mediante un contador a partir del primer reloj, ya que el divisor de frecuencias solo permite generar la frecuencia del primer reloj. El reloj que se genera con el contador no será simétrico ya que como se pretende que realice todo el conjunto de operaciones cada dos segundos se utilizará el flanco de subida.

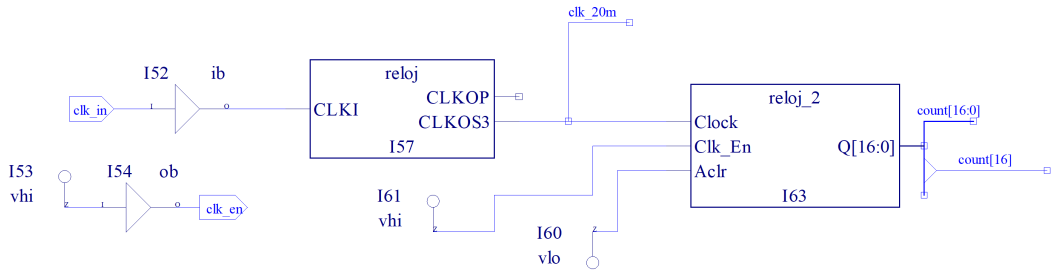


Figura 6.1.2: Esquema General 1

6.1.2. General II

La Figura 6.1.3 contiene los bloques necesarios para la lectura de los datos ya que el formato en el que presenta la información el sensor, no es apropiado para una lectura directa por lo que precisa un tratamiento. También para iniciar la trasmisión necesitará una señal específica por parte del circuito maestro, de esta función se encargará el bloque denominado control. Cuando se la proporcione el pin destinado a la trasmisión de la propia señal tendrá que modificar su función, leer los datos y enviarlos al diagrama de estados, para ello se ha de declarar como bidireccional.

Los datos que se reciban del sensor, se tendrán que los transformar desde el formato en que los envía el sensor, a uno en serie, bit a bit, para su tratamiento posterior. Como ya se ha comentado el trabajo esta ideado como un conjunto con la trasmisión de datos mediante bluetooth entre FPGAS, dicho trabajo se explica en el Anejo 2.

La trasmisión de estos datos se podría producir desde este bloque, ya que además de proporcionar los datos en serie proporciona una señal de habilitación, cuando se produce un cambio de bit en la entrada. El siguiente bloque también

posibilita la trasmisión de los datos, ya que es cuando se produce el final del envío de datos.

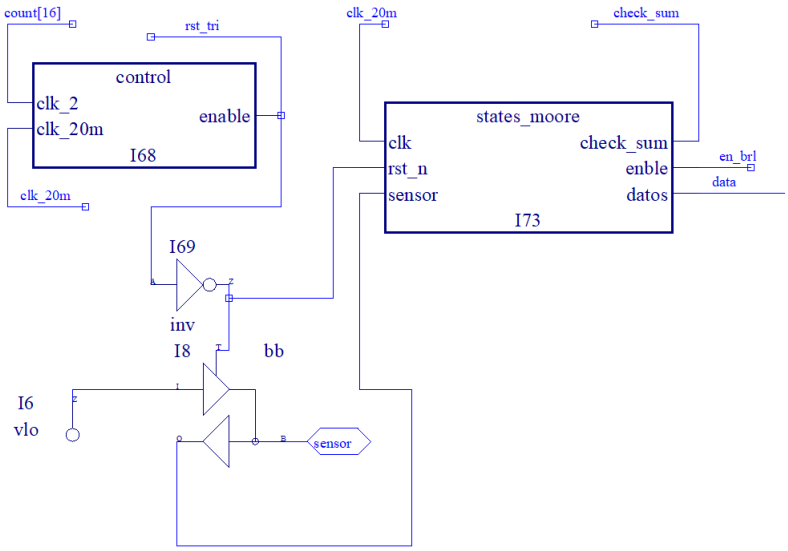


Figura 6.1.3: Esquema General 2

6.1.3. General III

La Figura 6.1.4 muestra la parte encargada del almacenado y procesamiento de los datos, el bloque denominado «barrel_shifter» almacenará todos los datos que los produzca el bloque anterior, una vez almacenados, se podrán utilizar de forma conjunta en el siguiente bloque «suma», que tomando todos los datos hará el chequeo de los datos para determinar si los datos son válidos o no si éstos lo son, entonces en función de las entradas de selección dará una salida u otra en función de la seleccionada. Estas entradas serán las que lleva la placa de prácticas utilizada como interruptores, para seleccionar lo que se mostrará por los displays.

Además el bloque «Suma» producirá una señal cuando sea, o no, correcta la suma, esta señal no se utiliza en bloques posteriores pero puede utilizarse para comenzar la trasmisión de los datos, unicamente cuando se reciban datos

válidos, de esta forma se evitaría enviar datos erróneos, gastando recursos y el tiempo de procesamiento necesario para enviar la información.

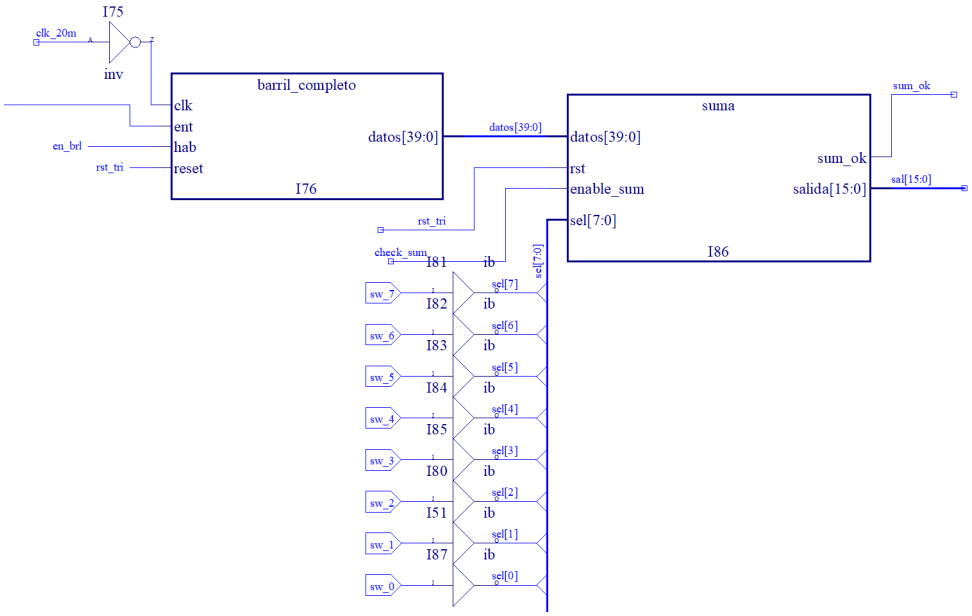


Figura 6.1.4: Esquema General 3

6.1.4. General IV

La Figura 6.1.5 representa la parte final del Esquema encargada de la representación de los valores recibidos en formato binario. Para llevar a cabo esa transformación será necesario, a través de un algoritmo especifico, realizar la división. Una vez realizada la división en potencias de diez, estos valores estarán en formato BCD Natural para su fácil interpretación en bloques posteriores. Cuando se haga referencia a las unidades decenas y centenas, son con respecto al número entero recibido, no con respecto al número a representar, ya que esté está desplazado un número la coma, es decir, las unidades son las décimas, las decenas son las unidades y las centenas son las decenas. Las unidades y decenas se mostrarán en dos displays de siete segmentos que, mediante un bloque, se transformarán para su correcta visualización, debido a que la placa solo cuenta con dos displays las centenas se mostrarán por los leds de la placa, en formato BCD Natural.

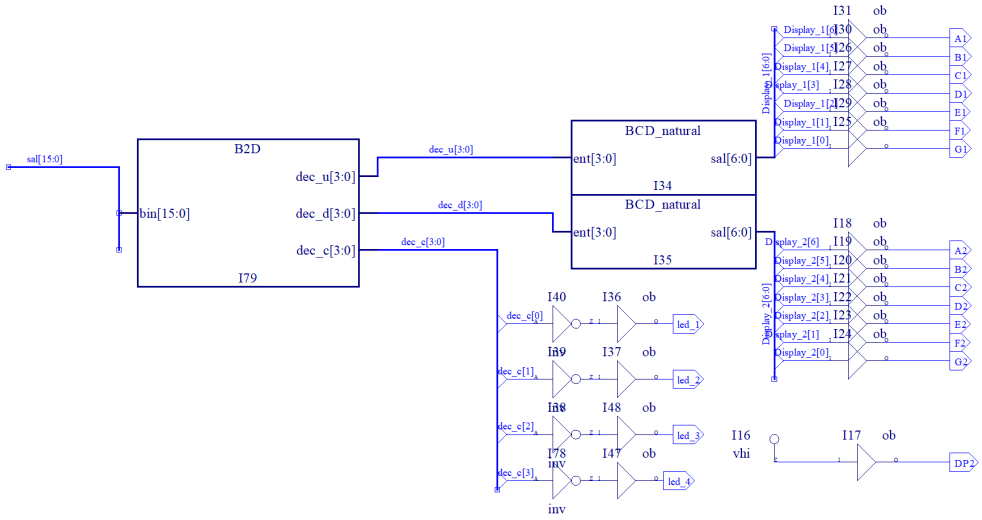


Figura 6.1.5: Esquema General 4

6.2. Generación de Relojes

A la hora de muestrear las señales será necesario hacerlo cada dos segundos, ya que esta es la frecuencia de actualización del sensor. El muestreo se ha de realizar a una frecuencia apropiada para poder leer de forma correcta las transiciones de nivel y la duración de los pulsos.

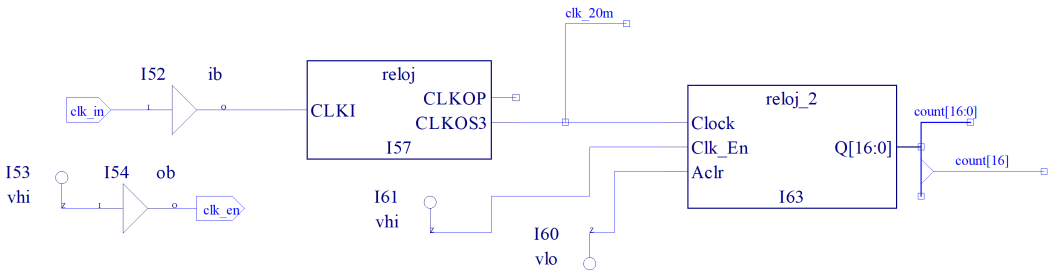


Figura 6.2.1: Esquema Relojes

El correcto funcionamiento del sistema se basa principalmente en la generación de dos relojes. La tarjeta diseñada para realizar prototipos incorpora un oscilador propio de 50MHz. Para la obtención de los datos necesitaremos un periodo de, al menos, $20\mu s$, por lo tanto se usará una frecuencia de 50KHz.

La generación del primer reloj se hará con el PLL contenido en el dispositivo que se configurará con la herramienta IPexpress, utilizado en el modo divisor de frecuencia. El modo se usa para obtener a partir un reloj de frecuencia alta otro o varios de frecuencias inferiores, de forma que se usará para obtener el primer reloj al dividir la frecuencia que genera el oscilador entre 1.000.

.

El segundo reloj lo generaremos mediante un contador, a partir del primer reloj, para ello tendremos que volver a utilizar la herramienta IPexpress pero en el apartado “counter”, generando un contador hasta 10.000. De esta manera y tomando el bit más significativo de este, se puede dividir el primer reloj que hemos generado entre 10.000, lo cual nos produce un reloj de 0,5Hz o un periodo de 2s, que es el reloj necesario para iniciar los mecanismos de lectura del sensor. El reloj no será simétrico como tal ya que al reducir el recorrido de este hasta 10.000 cuando lleguemos a ese valor, el tiempo que habrá permanecido a valor bajo el último bit será mayor que el tiempo a valor alto. Esta desigualdad se aprecia más claramente en la Figura 6.2.2. Esto no será un problema mayor debido a que en etapas posteriores se utiliza el flanco de subida del reloj que se produce exactamente cada 2s, así no se depende del tiempo a nivel alto o a nivel bajo.

Figura 6.2.2: Simulación Relojes

Para la simulación de la Figura 6.2.2 se ha introducido un reloj de 50MHz a través de la señal «clk_in» que será la misma frecuencia que nos proporciona el oscilador de la placa. La frecuencia se reduce a 50KHz para obtener el primer reloj y ésta a su vez se reduce hasta 0.5Hz ofreciendo así los dos relojes necesarios para el correcto funcionamiento.

6.3. Protocolo de Comunicación

A continuación se comentará el protocolo propio que utiliza el sensor para su comunicación con el dispositivo de control, en el caso de este dispositivo, una FPGA. Los dos dispositivos cumplen los roles correspondientes a una comunicación tipo maestro-esclavo, siendo la FPGA la que actúa como maestro y el sensor como esclavo.

Cuando el Sensor reciba la señal de inicio, está sera de 1ms de duración y poniendo a nivel bajo la línea, el sensor comenzará a entregar la información. Los datos vendrán precedidos por un grupo de señales de sincronismo, de tal forma que permita al maestro estar preparado para la comunicación.

Finalizada la trama de bis de sincronización comienza la emisión de datos, estos datos al presentar un protocolo propio, la interpretación de estos es también específica. La lectura de un «0» o un «1» vendrá dada por la duración del pulso a nivel alto de la línea, es decir, para todos los bits la línea comenzará a nivel bajo. La selección entre sí será si la línea permanece primero $\approx 50\mu s$ y después $\approx 26\mu s$ para un «0» o $\approx 70\mu s$ para un «1». El fin de los datos se producirá cuando se encuentre la línea a «1» más tiempo. La producción de los estados de la línea se pueden ver con un diagrama de tiempos en la Figura 6.3.1 y los valores de dichos tiempos en la Figura 6.3.2. La explicación más detallada del funcionamiento del sensor a la hora de trasmitir los datos se realiza por parte del fabricante en el Anejo 3 de este TFG.

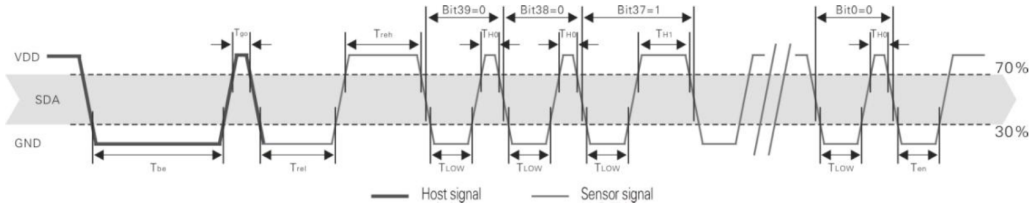


Figura 6.3.1: Salida sensor

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 6.3.2: Tabla tiempos Sensor

6.4. Señal de inicio

Este bloque comentará como se ha realiza internamente la señal de inicio necesaria para que el sensor comience la trasmisión de los datos.

La señal de inicio se generará a partir de los dos relojes. Desde el punto de vista del sensor será necesario imponer un valor lógico bajo por parte del maestro y mantenerlo durante un tiempo específico, para iniciar la comunicación. El tiempo que tendrá que permanecer la línea a nivel bajo deberá de ser aproximadamente de 1ms si deseamos hacer una lectura de temperatura en grados centígrados.

Luego tendremos que generar un pulso con una duración de 1ms y un periodo de 2s que es la frecuencia de activación del sensor. Para generar la temporización se creará un contador con limite superior de 50. El reloj de mayor frecuencia será utilizado para incrementar el contador, que mientras no se haya alcanzado dicho límite mantendrá una señal activa. La señal será utilizada como reset global y como señal de inicio de la comunicación del sensor.

La señal se produce mediante una descripción en mediante código VHDL.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_arith.all;
4  use IEEE.std_logic_unsigned.all;
5
6  entity control is
7  port(
8      clk_2       :in std_logic;
9      clk_20m     :in std_logic;
10     enable      :out std_logic);
11 end control;
12
13 architecture arch_control of control is
14
15 signal s_enable: std_logic := '0';
16 begin
17
18     process(clk_2,clk_20m)
19     variable var_count: integer range 0 to 50;
20
21     begin
22
23
24         if (clk_20m'event AND clk_20m = '1' AND clk_2 = '1') then
25
26             if (var_count < 50 ) then
27                 s_enable    <= '1';
28                 var_count   := var_count + 1;
29
30             else
31                 s_enable <= '0';
32             end if;
33
34         elsif(clk_2 = '0')then
35             var_count:=0;
36
37         end if;
38
39     end process;
40     enable <= s_enable;
41
42 end arch_control;
```

Figura 6.4.1: Control_global

El pulso se realiza con la señal «s_enable» que, mientras se este produciendo la cuenta. Ésta tomará el valor de «1», mantenimiento mientras el valor de la cuenta sea inferior al limite fijado. Una vez alcanza dicho valor la salida tomará el valor de 0 manteniéndose hasta que se inicie de nuevo la cuenta, el valor de ésta no volverá a ser cero hasta que el reloj de dos segundos tome de nuevo dicho valor.

El resultado final será que la señal «enable» se mantiene activa durante 1ms, el tiempo necesario para que el sensor identifique la señal y comience la transmisión de la información. Ya que necesitamos borrar el contenido de los bloques posteriores, esta señal de impulso se utilizará además para resetearlos, dejándolos preparados para los eventos que producirá el sensor. La señal será

denominada «rst_tri», debido a su doble funcionalidad.

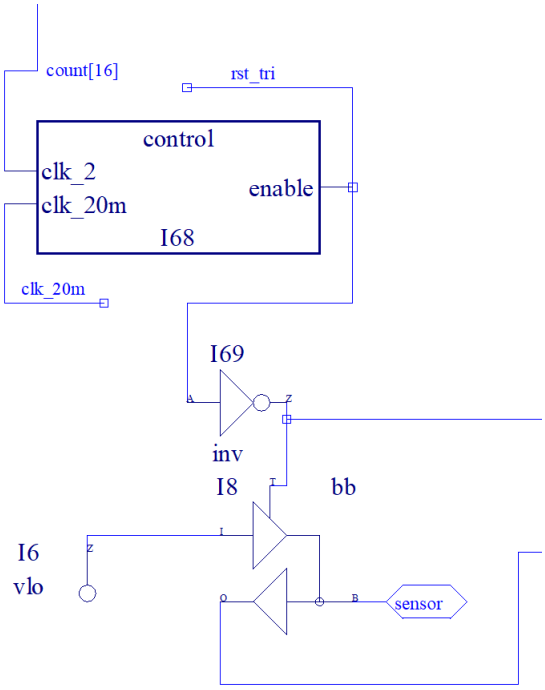


Figura 6.4.2: Control_Esquematico

La señal afectará al bidireccional de la Figura 6.4.2 este controla la funcionalidad Maestro-Esclavo entre la FPGA y el Sensor respectivamente. Cuando se activa la señal invertida de «rst_tri» el bi estable colocará la linea común a nivel bajo durante el tiempo que permanezca la señal «rst_tri» activada. Una vez haya terminado el pulso entonces el sensor comenzará a trasmitir su trama, para entonces el bidireccional deberá permitir la lectura del pin de forma que los valores que proporcione el sensor se trasmitan por todo el dispositivo y se traduzcan para interpretarlos en los bloques posteriores.

6.5. Máquina de estados

La sección actual comentará la obtención y transformación de los datos correspondientes al sensor, primero se ilustrará de forma teórica dicha obtención, para más adelante presentar el esquema real.

El modelo teórico comienza en Inicio, comprobando si se encuentra en la fase de sincronización o no. Si la respuesta es negativa continua en Inicio. Si por el contrario la respuesta es afirmativa pasa al estado de lectura de Datos. Este estado cuando se produce una lectura de dato no válido vuelve al origen. Si el dato es válido entonces se almacena y vuelve al estado de lectura de dato. También es posible que se interrumpa la comunicación en un momento dado en una fase intermedia, dicho evento se soluciona mediante un reset a Inicio, que se produce de forma automática y cada 2s para que la máquina se encuentre en el estado Inicial de forma inequívoca cuando se inicie la trasmisión.

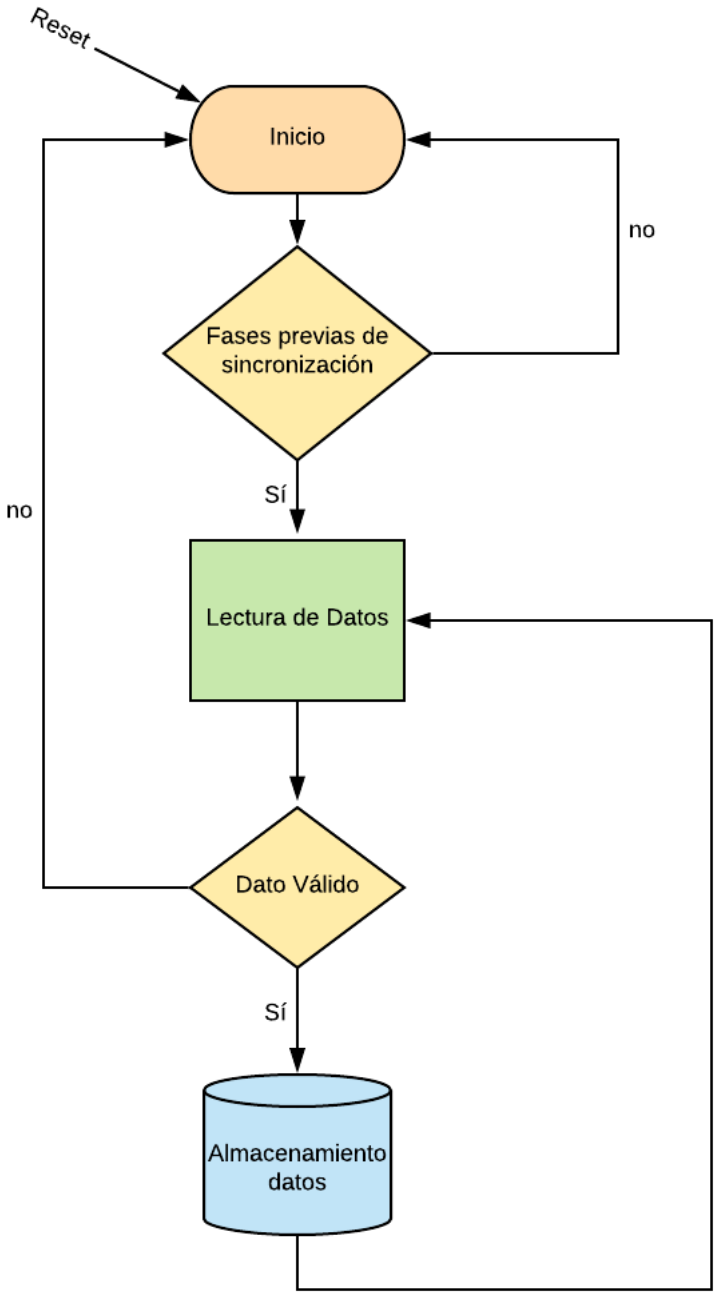


Figura 6.5.1: Diagrama de Flujo de la FSM

Una vez iniciada la trasmisión por parte del sensor la máquina de estados se encargará de recibir los valores que tomará el pin y este los interpretara para generar las señales necesarias para los bloques siguientes. Como se puede apreciar en la Figura 6.5.2 el sensor recibe la activación de la FPGA y después comienza la transmisión de la información, donde habrá un par de señales de

sincronización, para comenzar la transmisión de la información como tal cuando la linea permanezca a nivel alto indicará que la trasmisión ha finalizado.

La Figura 6.5.2 muestra la señal física del sensor cuando se produce la comunicación, capturada con el analizador lógico.

Figura 6.5.2: Señal de Inicio y respuesta Sensor

El bloque utiliza una máquina de estados generada a través del modelo de Moore para que sea más robusta. Esta lo es debido a que cuando se utiliza el modelo de Moore se asegura que las salidas de la máquina permanecen al menos un ciclo de reloj en cada estado, lo cual, evita pulsos demasiado cortos para que los interpreten los bloques posteriores. El bloque contará con una entrada de «reset», este se activará cuando se active la señal «rst_tri», que reiniciará la máquina de estados, ya que debe de encontrarse en dicha posición cuando se inicie el sensor. De esta forma se puede asegurar, que cuando comience a trasmitir el sensor la máquina de estados este preparada para recibir la correcta información y trasmitirla al siguiente bloque de forma adecuada.

La máquina de estados cuenta con una entrada «sensor» que será la que proporcione la salida del sensor y también contará con 3 salidas, «check_sum, enable y datos». La primera se utiliza unicamente cuando se detecta el fin de emisión de caracteres, poniéndose a «1» durante un ciclo de reloj. La segunda habilita la escritura de datos. La tercera representa el dato a escribir en el registro. Éstas salidas serán utilizadas por los bloques posteriores.

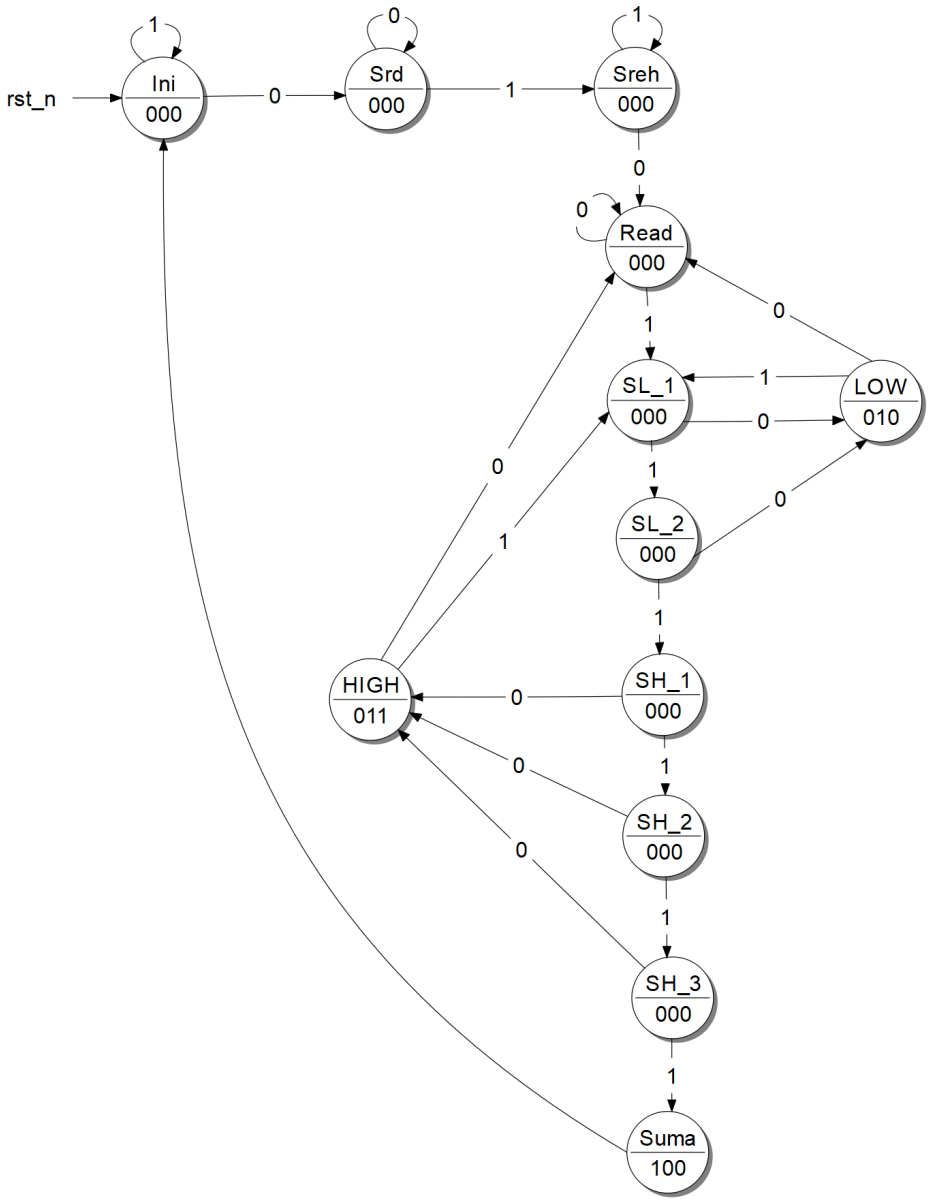


Figura 6.5.3: States_Moore

El sensor generará un conjunto de bits que como se ha explicado en el capitulo del sensor, una vez se salga del «Reset» el sensor generará un «0» y un «1» , que corresponderán con los estados «Srd» y «Sreh», serán estado preparatorios para la fase detección de datos. Una vez se reciba un «0» se pasará al estado «Read». Si después de pasar al estado «Read» detectara uno o varios «0», permaneceríamos en este. Cuando se detecten uno o dos «1», pasaremos a los estados «SL_1» y «SL_2» respectivamente, lo cual significará que si el siguiente bit es un «0», la máquina de estados pasará a «LOW» y tendrá que almacenar un «0». Si en este estado se vuelve a recibir un «0» se pasará a «Read» y si se recibe un «1» a «SL_1» puesto que ya contaría como que ha

recibido un cero.

Si por el contrario se detectan más unos, se pasará a los estados «SH_1», «SH_2» ó «SH_3», ya que se podrá recibir hasta 5 unos seguidos, si el siguiente es un «0» la máquina pasará a «HIGH» donde tendrá que almacenar un «1», y al igual que en «LOW». Si continúa recibiendo un «0», cuando se detecten más unos, la máquina de estados pasará a Suma y enviará una señal de terminación de datos, de forma que indicará a los siguientes bloques que pueden proceder con la información ya que esta es coherente.

La máquina permanecerá en el estado «Ini» por defecto, ya que la línea se queda a nivel alto cuando no es usada para transmitir información. La máquina, por lo tanto no abandonará ese estado hasta que se termine el «Reset». Cuando se produce el «Reset» se envía la máquina a «Ini», ya que el estado de la máquina puede no ser «Ini», debido a un incorrecto funcionamiento ya sea por parte de sensor o de un error al leer la entrada de datos. De esta forma siempre que comienza la trasmisión de datos, la máquina de estados deberá comenzar en el estado «Ini».

6.6. Barrel Shifter

A continuación se comentará el bloque encargado de almacenar la información procedente del sensor.

Este bloque es el encargado de almacenar de forma secuencial la información proporcionada por el sensor, ya que se basa en un registro con desplazamiento, el funcionamiento de este se basa en desplazar los bits a medida que se va generando nueva información de forma que cuando se llegue al final, se tendrá un conjunto de información válida. El barril completo estará compuesto por 5 bloques que realizarán las funciones de un desplazador de barril de 8 bits, como se muestra en la Figura 6.6.1.

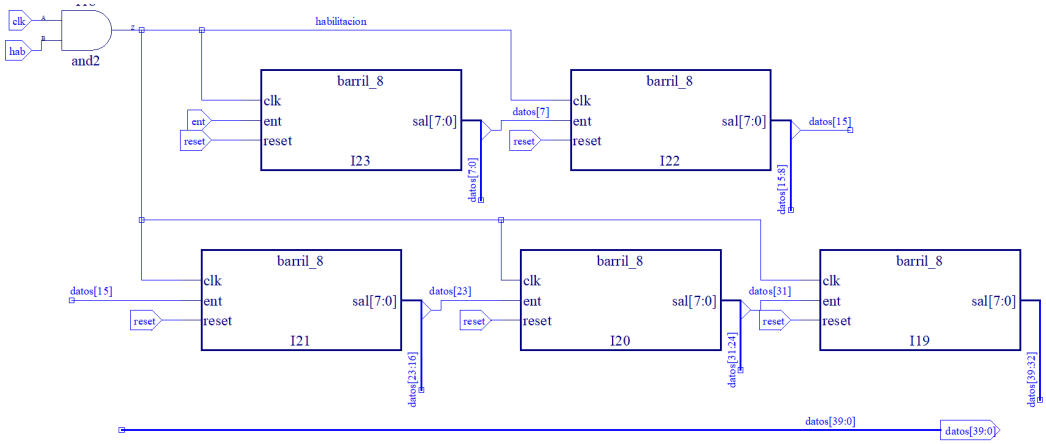


Figura 6.6.1: barrel Shifter

Para su correcto funcionamiento tendrá que contar con varias funciones como habilitación entrada de «datos», «clk», entrada de «datos» y «reset». Para «clk» se utilizará el reloj de 50kHz, ya que este es el reloj que se utiliza en la máquina de estados facilitando así la coordinación entre ambos bloques.

La lectura de los valores correctos por parte de los registros adquiere una gran importancia, debido a que cuando se usa el mismo flanco de reloj para actualizar el estado de la máquina y para habilitar la lectura del registro, existe un grado de incertidumbre sobre el valor que se guardará en dicho registro.

La forma propuesta para eliminar la incertidumbre se basa en utilizar el complemento del reloj en el bloque secundario, es decir, la habilitación del registro se produce en el flanco posterior a la actualización de los valores de la entrada, así no se presentan problemas a la hora de leer la entrada al registro ya que esta lleva activa medio ciclo de reloj. Por lo que será necesario introducir una

puerta inversora delante del reloj para tener la activación como se ha descrito anteriormente.

La habilitación se producirá por parte de la máquina de estados, cuando tenga un dato válido cambiará la señal de habilitación a «1» y cuando el «clk» de el flanco de bajada (de subida para el bloque en cuestión), leerá la entrada y almacenará dicho dato hasta que se produzca otro dato válido, desplazando el anterior sucesivamente hasta llenar el registro completo.

El registro necesitará colocar el bit más significativo como entrada del siguiente bloque de esta forma se generará el desplazamiento del registro a lo largo de los cinco bloques que lo componen.

El borrado del registro se produce de manera interna, cuando se recibe la señal de reset global entonces se borrará el contenido del registro y se irá introduciendo un nuevo valor de la forma antes comentada, el reset se producirá de forma repetida cada 2 segundos, ya que esta es la frecuencia de actualización del sensor.

6.6.1. Barril_8

Los módulos en los que se subdivide el barril completo a su vez estarán compuestos por 8 filp-flops tipo D, colocados en cascada de forma que permite almacenar cada vez un bit. La salida de un flip-flop será la entrada del siguiente flip-flop de forma que se encadenen hasta generar el registro de desplazamiento de 8 bits. Las entradas de «reset» y de «clk» serán comunes ya que todos los registros se tienen que actualizar de forma simultánea ya sea para leer datos o para borrarlos cuando se active la entrada de «reset».

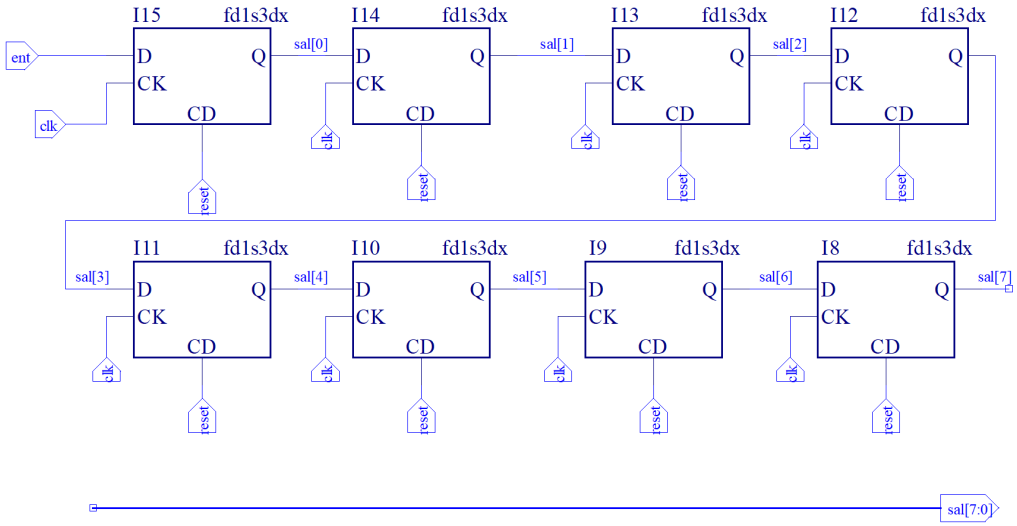


Figura 6.6.2: Barril Shifter 8

6.7. Chequeo de Errores

El bloque se encargará de tomar los valores del registro de desplazamiento y comprobar si se tratan de datos válidos o no ya que se tendrá que realizar la suma de los dos datos y su posterior comprobación.

La comprobación comenzará cuando reciba la señal por parte de la máquina de estados que se ha concluido la emisión de información. El sensor proporciona 3 conjuntos el primer conjunto pertenece a los 16 primeros bits y representan la lectura de humedad esta lectura será un número en hexadecimal. El segundo conjunto proporciona otros 16 bits que representarán la lectura del sensor en temperatura. El último conjunto de bits que representan 8 bits son destinados para la depuración de errores ya que es posible que al recibir los datos se haya introducido algún error, por lo tanto, este último conjunto es la suma de los 4 bytes que mandará el sensor que será también otro byte.

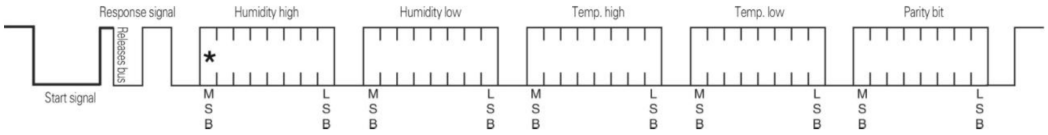


Figura 6.7.1: Formato de Datos

La comprobación de que los datos no contienen errores se realiza en el bloque suma, esta comprobación se realiza sumando los Bytes utilizados para datos unicamente y comparándolos con el Byte de Paridad. La comparación se realiza acuerdo a la Figura 6.7.2

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)= $2 \times 256 + 9 \times 16 + 2 = 658$

=> Humidity = 65.8%RH

Temp.: 0000 0001 0000 1101 = 10DH(Hexadecimal) = $1 \times 256 + 0 \times 16 + 13 = 269$

=> 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)= $6 \times 16 + 5 = 101$

=> 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.

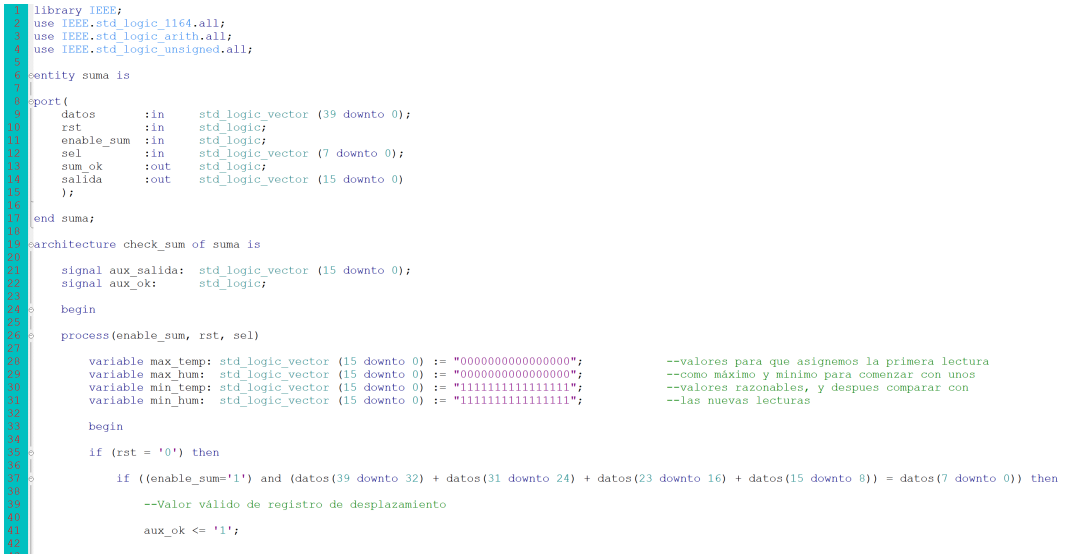
Figura 6.7.2: Chequeo Datos

Cuando compruebe que la suma es correcta se generará una señal para continuar con el procesamiento de los datos y mostrarlos por el display.

Una vez que ha comprobado los errores podrá, a través de las entradas digitales seleccionar los datos a mostrar por la pantalla, se ha configurado para que, por defecto y cuando se introduce un código no válido, se muestre el valor de la suma en la parte baja y todo unos en la parte alta.

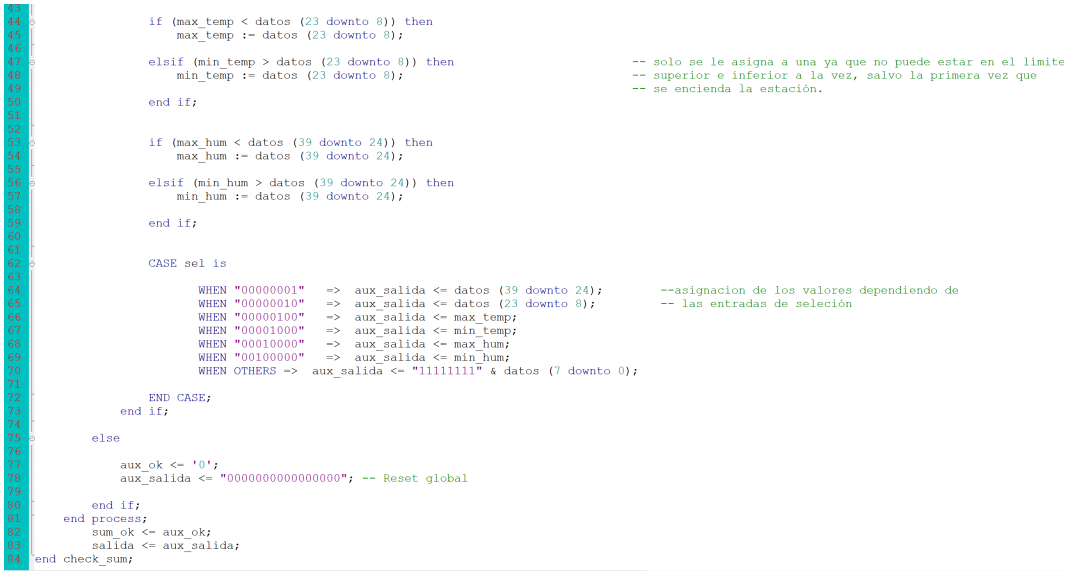
Los valores posibles de salidas serán una para el valor actual de la temperatura y otra para la humedad, así como sus valores máximos y mínimos registrados, desde que se encendió el dispositivo, dejando un total de 6 posibles salidas útiles ya que la salida de la suma no se ha considerado como válida.

La salida de este bloque tendrá que ser tratada por el siguiente bloque ya que si se mostrarán los valores por los dos módulos de 7 segmentos se debe dividir el número.



```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity suma is

port(
    datos       :in     std_logic_vector (39 downto 0);
    rst         :in     std_logic;
    enable_sum  :in     std_logic;
    sel         :in     std_logic_vector (7 downto 0);
    sum_ok      :out    std_logic;
    salida      :out    std_logic_vector (15 downto 0)
    );

end suma;

architecture check_sum of suma is

    signal aux_salida:  std_logic_vector (15 downto 0);
    signal aux_ok:      std_logic;

    begin

    process(enable_sum, rst, sel)

        variable max_temp: std_logic_vector (15 downto 0) := "0000000000000000";            --valores para que asignemos la primera lectura
        variable max_hum:  std_logic_vector (15 downto 0) := "0000000000000000";            --como máximo y mínimo para comenzar con unos
        variable min_temp: std_logic_vector (15 downto 0) := "1111111111111111";            --valores razonables, y despues comparar con
        variable min_hum:  std_logic_vector (15 downto 0) := "1111111111111111";            --las nuevas lecturas

        begin

        if (rst = '0') then

            if ((enable_sum='1') and (datos(39 downto 32) + datos(31 downto 24) + datos(23 downto 16) + datos(15 downto 8)) = datos(7 downto 0)) then

                --Valor válido de registro de desplazamiento

                aux_ok <= '1';
```

Figura 6.7.3: Suma.vhdl 1



```vhdl
                if (max_temp < datos (23 downto 8)) then
                    max_temp := datos (23 downto 8);

                elsif (min_temp > datos (23 downto 8)) then                          -- solo se le asigna a una ya que no puede estar en el límite
                    min_temp := datos (23 downto 8);                                 -- superior e inferior a la vez, salvo la primera vez que
                                                                                     -- se encienda la estación.
                end if;


                if (max_hum < datos (39 downto 24)) then
                    max_hum := datos (39 downto 24);

                elsif (min_hum > datos (39 downto 24)) then
                    min_hum := datos (39 downto 24);

                end if;


                CASE sel is

                       WHEN "00000001"   =>  aux_salida <= datos (39 downto 24);        --asignacion de los valores dependiendo de
                       WHEN "00000010"   =>  aux_salida <= datos (23 downto 8);         -- las entradas de seleción
                       WHEN "00000100"   =>  aux_salida <= max_temp;
                       WHEN "00001000"   =>  aux_salida <= min_temp;
                       WHEN "00010000"   =>  aux_salida <= max_hum;
                       WHEN "00100000"   =>  aux_salida <= min_hum;
                       WHEN OTHERS =>  aux_salida <= "11111111" & datos (7 downto 0);

                END CASE;
            end if;

        else

            aux_ok <= '0';
            aux_salida <= "0000000000000000"; -- Reset global

        end if;
    end process;
        sum_ok <= aux_ok;
        salida <= aux_salida;
end check_sum;
```

Figura 6.7.4: Suma.vhdl 2

La descripción realizada en vhdl en la Figura 6.7.3 consiste en el código utilizado para la descripción del bloque suma para que realice todo el conjunto de funciones antes comentadas.

Las entradas al bloque son: «datos» que es un vector de 40 bits, «sel» vector de 8 bits y «rst», «en_sum» son entradas de control de un único bit. Las salidas del bloque son «salida» un vector de 16 bits que contiene la información, que se mostrará por el display y «sum_ok», que se trata de un bit de control para bloques posteriores.

Se han definido dos señales auxiliares para utilizarlos en lugar de los valores de salida del bloque «aux_salida» y «aux_ok» con 16 y 1 bits respectivamente. Dentro del «Process» se utilizan 4 variables que almacenarán los valores máximos y mínimos que reciba procedentes del sensor. Cuando no se reciba un valor negativo de «rst» entonces se comprobarán los datos obtenidos ya que estos pueden ser erróneos, en caso contrario se «resetearán» los valores de salida y «sum_ok».

Para realizar la comprobación se necesita la suma algebraica de los cuatro primeros Bytes de información, estos se comprueban con el ultimo Byte recibido. Si la comprobación es positiva entonces se procederá a comprobar los valores leídos con los almacenados si estos son menores o mayores se guardan en las variables correspondientes para cada una de las magnitudes. Se ha optado por la utilización de una única sentencia «if»y «elsif», en lugar de dos sentencias «if», esto es debido a que cuando se produzca una lectura no es posible que ésta se encuentre por encima del límite superior y por debajo del inferior.

A la situación anterior existe una excepción ya que, cuando se inicia el dispositivo, éste tendrá como valores iniciales para las variables «max» y «min» y por lo tanto se podrá encontrar en ambos límites Ésto no es preocupante ya que, en el primer ciclo se almacenará dicho valor leído en la variable «max» al estar ésta la primera condición. En el siguiente ciclo si el valor es el mismo no se cumplirá la primera condición pero si la segunda, provocando que se actualice el valor «min».

El funcionamiento de la estación esta destinado a leer las condiciones del entorno durante largos periodos de tiempo y por lo tanto las variaciones en las condiciones del entorno son mucho más lentas que la tasa de refresco del sensor, por lo tanto, durante un funcionamiento normal los valores máximos y mínimos se almacenarán de forma correcta.

El modo seleccionado para cuando se produce una entrada de selección diferente se realiza con la concatenación de un Byte «0xFF» (todo a unos) y el Byte de suma, como resultado de salida.

Una vez terminada la descripción del «Process» se asigna los valores temporales a las salidas del bloque.

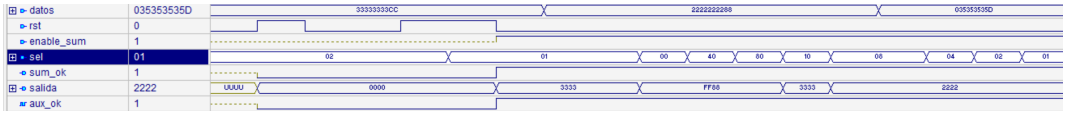


Figura 6.7.5: Simulación Suma

La Figura 6.7.4 muestra la simulación del bloque suma se aprecia, que hasta que no se active la señal de «enable_sum» el bloque no producirá salidas diferentes de «0» una vez esta este activada el bloque comenzará con su funcionamiento normal. Cambiando la función a mostrar, a la vez que cambian las entradas de selección ya que mediante las entradas de selección se pueden observar los valores inferiores y superiores, en distintos instantes, sin tener por que ser el valor que esta recibiendo actualmente.

Esto se demuestra ya que cuando se recibe la segunda lectura en la parte de salida se mostrará el valor leído anteriormente, ya que éste se trata del más alto registrado desde que se encendió el dispositivo. Lo mismo sucede cuando cambiamos la entrada a la tercera lectura de forma inversa, ya que esta vez se muestran los valores más bajos que coinciden con la lectura anterior.

Durante la ejecución de la simulación aparece como salida válida «0xFF88», esto es debido a que se han introducido valores no contemplados como válidos por lo tanto la salida será todo «unos» en la parte alta y el valor de la suma en la parte baja.

6.8. Binario a BCD

El Bloque B2D convierte a BCD el número recibido antes en binario con 16 bits

La señal que se reciba del bloque suma necesita ser convertida en BCD para que se pueda representar en los displays de 7 segmentos que cuenta el circuito, como solo se dispone de 2 displays y se necesitan 3 para la representar todos los dígitos, que nos proporciona el sensor en cuanto a humedad y a temperatura, se usaran los leds auxiliares par la cifra más significativa. Como ya se ha comentado la información generada por el sensor, precisa, para su representación, tres dígitos decimales, dos corresponden a la parte entera de la lectura y uno esta destinado a la parte fraccionaria. Para la representación de los 3 dígitos que son proporcionados, se ha optado por utilizar los leds auxiliares situados en la placa representando este último número en formato binario.

Ya que en descripciones vhdl no existe una función optimizada para realizar una división de un número entero sera necesario utilizar un algoritmo con cíclico. El algoritmo de división, es un algoritmo que consumirá muchos recursos del dispositivo, cuando se realice su implantación física.

El bloque devolverá los 3 dígitos de forma separada y en binario para poder luego llevarlos a los convertidores de siete segmentos.

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

entity B2D is
    port(
    bin     :in     std_logic_vector (15 downto 0);
    dec_u   :out    std_logic_vector (3 downto 0);
    dec_d   :out    std_logic_vector (3 downto 0);
    dec_c   :out    std_logic_vector (3 downto 0)
    );
end B2D;

architecture arch_B2D of B2D is

    begin
    Process (bin) is

        variable res, den_1,den_2, d_1,d_2, c: integer;
        begin

           c := conv_Integer(bin);
           den_1 := 10;
           d_1 := 0;
           den_2 := 100;
           d_2 := 0;


           for i in 1 to 10 loop                        --division centenas

                c := c - den_2;
            if (c>=den_2) then                          --Algorítmo división
                d_2 := d_2 +1;

            end if;
            end loop;

            c := (conv_Integer(bin)-d_2*100);           ---Se eliminan las centenas previamente para
                                                        --reducir el número de bucles a realizar
```

Figura 6.8.1: Binario a BCD 1

```
            for i in 1 to 10 loop                       --division decenas
             if (c>=den_1) then                         --Algorítmo división

               c := c - den_1;
               d_1 := d_1 +1;

            end if;
            end loop;

        dec_c <= conv_std_logic_vector(d_2,4);
        dec_d <= conv_std_logic_vector((d_1),4);
        dec_u <= conv_std_logic_vector((conv_Integer(bin)-d_1*10-d_2*100),4);

    end process;
end arch_B2D;
```

Figura 6.8.2: Binario a BCD 2

La descripción vhdl realizada en las Figura 6.8.1 y 6.8.2, y consiste en el código realizado para la descripción del bloque Binario a BCD, dentro de dicho bloque se tendrá que producir la transformación, de binario a decimal. Para facilitar la utilización posterior de la información, la salida tendrá que ser también tratada de forma que se obtendrán 3 grupos de 4 bits, en formato BCD natural, donde cada grupo corresponderá con el valor que representa el número natural en potencia de 10.

La división se realiza mediante un algoritmo basado en restas sucesivas, la forma natural de realizarlo sería mediante un «while» con condición de terminación, tuvo que ser descartada debido a que se producían problemas de

compilación. Debido a que si se declara como «while» se desconoce a priori el número de ciclos que tendrá que realizar para obtener un resultado y por lo tanto el intérprete presentaba un error de compilación.

La resolución del error se llevó a cabo de la siguiente manera: ya que el número máximo que se podrá mostrar por el display será siempre un número comprendido entre 0-9 por lo tanto se podrán realizar como máximo 9 iteraciones por cada número. Mediante la combinación de las sentencias «for» e «if», se puede sustituir la sentencia «while», dentro del primer bucle «for» se introducirá el número original transformado a entero y se producirán restas sucesivas hasta completar la iteraciones necesarias, dentro del bucle. Cuando se cumpla la condición impuesta por el «if», siendo ésta que cuando el número sea inferior a «den» incrementaremos en una unidad «d» siendo esta el resultado que se obtendrá de la operación de división.

El segundo bloque unicamente presenta una diferencia ya que en lugar de comenzar con el número original será necesario restarle el resultado obtenido en el bucle anterior multiplicado por 100 que es el peso de las centenas en base decimal. Esto es necesario para que obtener el valor unicamente de las decenas sin tener en cuenta de las centenas, ya que si estas se tienen en cuenta y el valor de estas es superior a cero, el resultado de las decenas sería 9 independientemente del valor de estas.

Al aplicar la división para las centenas y las decenas, será necesario por último calcular las unidades, estas se obtendrán como resultado de la resta de las centenas y las decenas por sus respectivas potencias. La salida a valores interpretables se realiza mediante una conversión a binario con 4 bits, para obtener así el número en formato BCD.

Figura 6.8.3: Simulación Binario a BCD

La Figura 6.8.3 representa la simulación del bloque conversor de binario a BCD, este bloque simplemente convierte de un formato a otro dividendo el número en las diferentes unidades en las correspondientes potencias de 10 de forma que el número se divide en los 3 conjuntos que representan las unidades, las decenas y las centenas

6.9. Convertidor BCD 7 segmentos

El bloque se encarga de transformar la información que recibe a siete segmentos, de forma que cuando se reciba un valor válido (0-9) en la pantalla tendrán que iluminarse los segmentos correspondientes a la representación de dicho número. Como forma de corrección de errores cuando se introduce un número que no corresponda con un valor decimal en la pantalla, aparecerá una «E» para representar el error. Este valor se podrá ver cuando se introduzca un código no válido como selección.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity BCD_natural is
5  port (
6       ent: in std_logic_vector (3 downto 0);
7       sal: out std_logic_vector (6 downto 0)
8       );
9  end BCD_natural;
10
11 architecture arch_BCD of BCD_natural is
12 begin
13      sal<= "0110000" when (ent="0001") else
14            "1101101" when (ent="0010") else
15            "1111001" when (ent="0011") else
16            "0110011" when (ent="0100") else
17            "1011011" when (ent="0101") else
18            "1011111" when (ent="0110") else
19            "1110000" when (ent="0111") else
20            "1111111" when (ent="1000") else
21            "1111011" when (ent="1001") else
22            "1111110" when (ent="0000") else
23            "1001111";
24
25 end arch_BCD;
26
```

Figura 6.9.1: BCD 7 segmentos

La figura 6.9.1 muestra el código que contiene el bloque denominado como BCD_Natural que se encarga de transformar el número recibido en BCD y transformarlo para que de cara a la visualización en un display de siete segmentos se muestre el número correspondiente con la configuración BCD, esto se realiza mediante la estructura «when» «else» de forma que sin necesidad de generar un «process» y con una ejecución concurrente se genera la salida de forma automática.

Las salidas tendrán que ser conectadas a los displays para visualizar la información de forma correcta ya que si no se realiza la asignación correcta, las barras que se encenderán no serán las que corresponden con el número.

Capítulo 7

Funcionamiento Final de la Estación Meteorológica

A continuación se realizará una descripción del funcionamiento final de la Estación Meteorológica.

El funcionamiento final de la FPGA es igual al descrito en fases anteriores utilizando el swicht para la selección de la información a mostrar y con los displays y los leds auxiliares de la placa para mostrar la información.

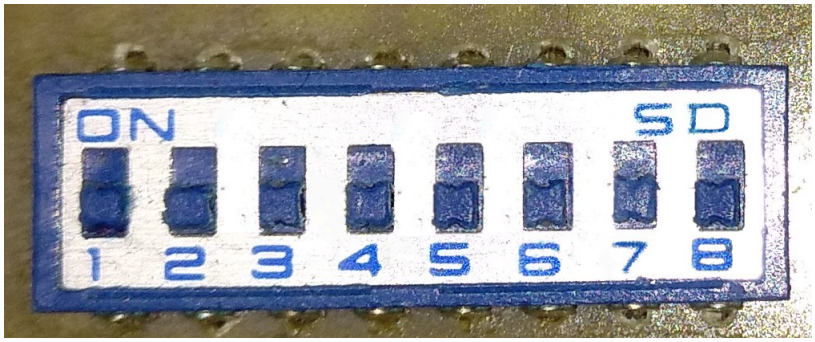


Figura 7.0.1: Selección bonotes

1. Valor actual humedad
2. Valor actual Temperatura
3. Valor Máximo Registrado de Temperatura
4. Valor Mínimo Registrado de Temperatura
5. Valor Máximo Registrado de Humedad

6. Valor Mínimo Registrado de Humedad

7. Cualquier Otra combinación de los botones mostrara un mensaje de error con el primer byte todo a unos y con el valor de la suma.

Un vez seleccionado el modo de funcionamiento por parte del usuario entonces se mostrará por los dos displays y por los leds el

valor de la lectura seleccionada, si se trata de el valor de la lectura actual, tendrá un tiempo de refresco de aproximadamente dos segundos, si se esta observando un valor de registro, este valor puede permanecer inalterado hasta alcanzar un nuevo valor que supere las condiciones del mismo, ya sea superior al máximo o inferior al mínimo.

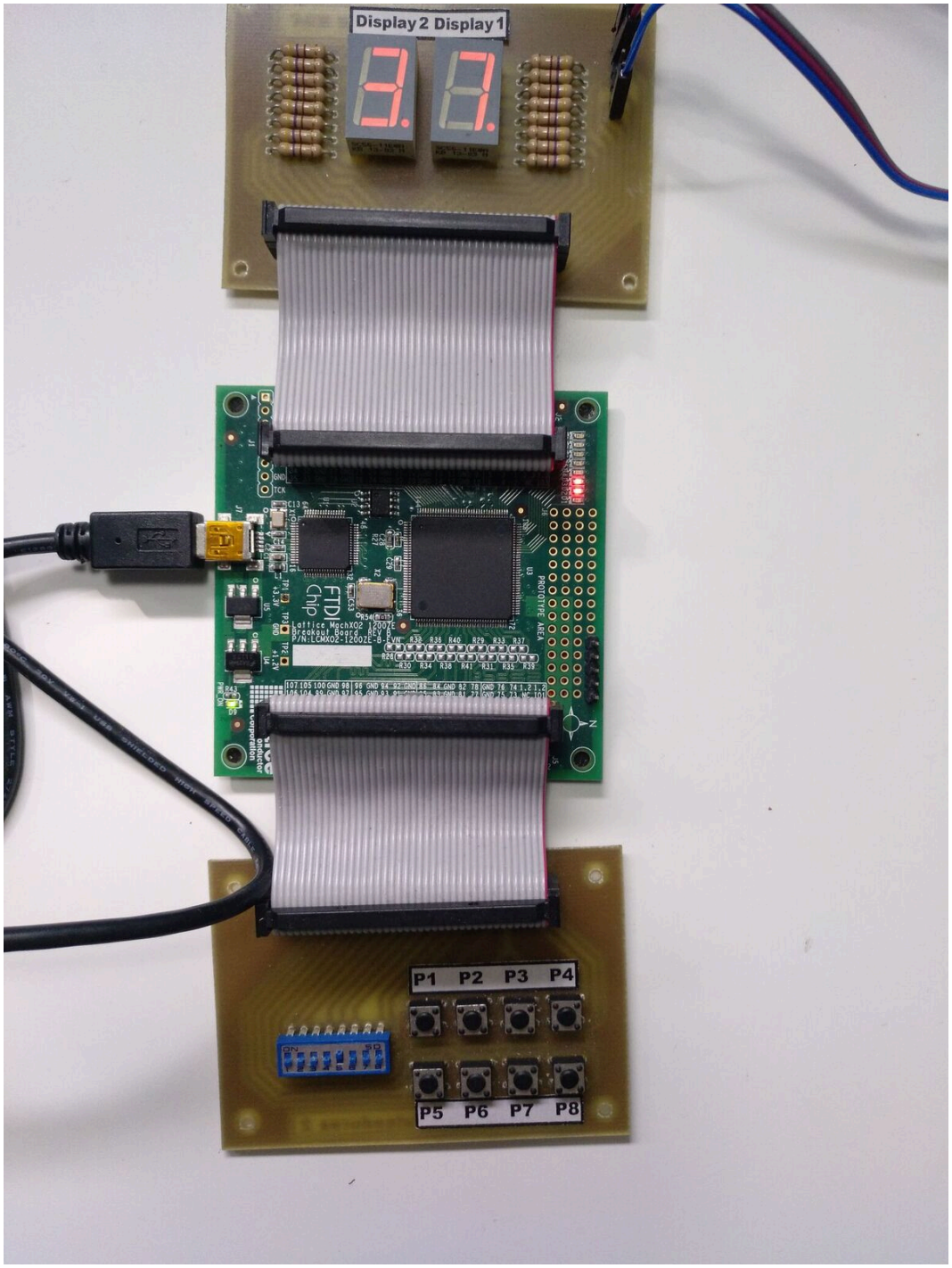


Figura 7.0.2: Funcionamiento Humedad

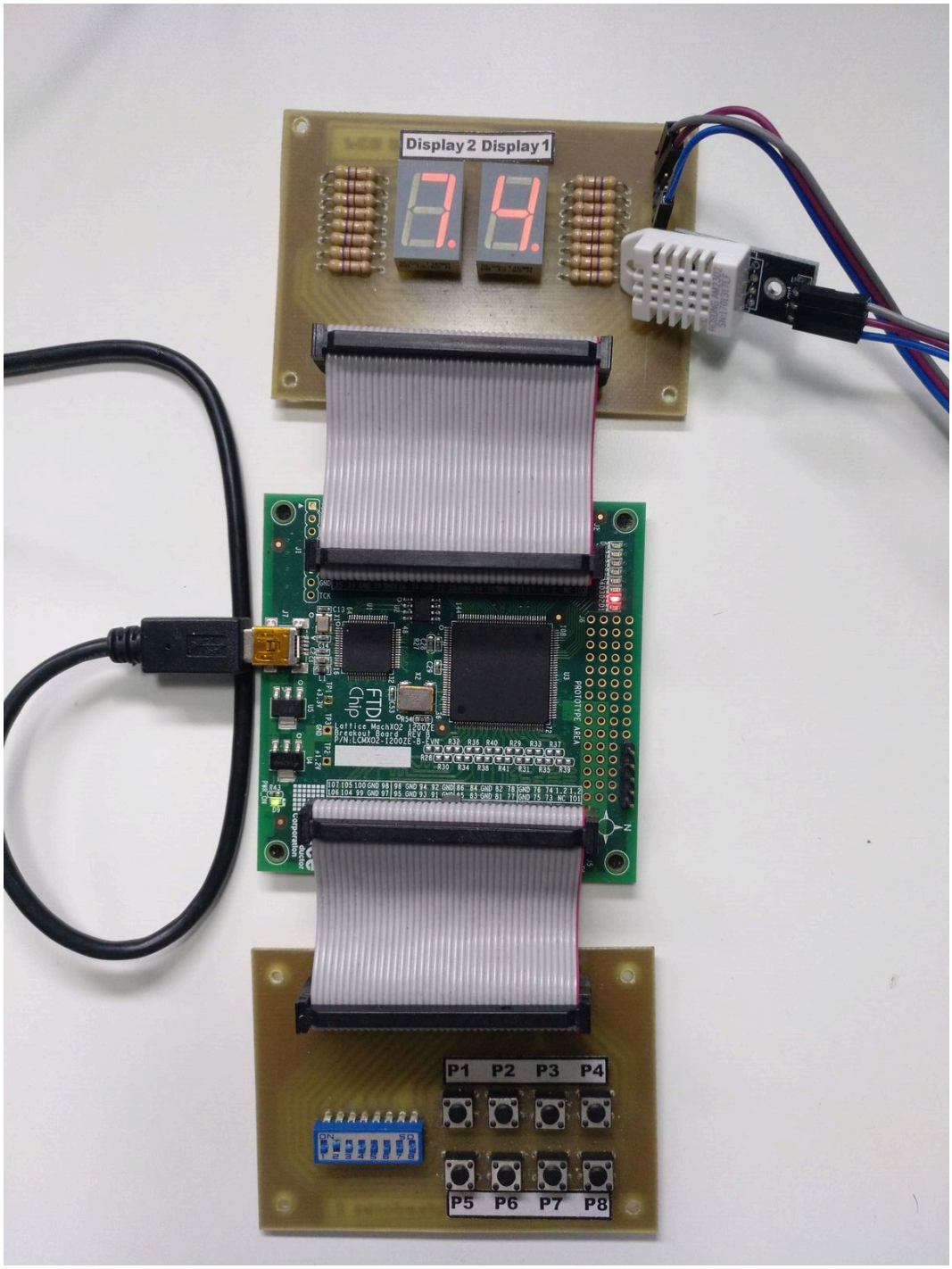


Figura 7.0.3: Funcionamiento Temperatura

Según las Figuras 7.0.2 y 7.0.3, la lectura de humedad relativa sería del 63.7 % y la de temperatura de 27.4ºC.

Capítulo 8

Conclusiones y Futuras Líneas de Desarrollo

Las posibles aplicaciones del dispositivo, no deberían estar determinadas ya que se trata de un medio para proporcionar ciertas variables, de una forma cómoda y configurable de cara al usuario final. siendo este dispositivo un medio para la obtención de un bien superior.

El dispositivo podrá tener distintas aplicaciones: como estación meteorológica, para poder conocer el tiempo antes de salir al exterior; aplicaciones de domótica, o en aplicaciones distribuidas SCADA; donde según como sean las condiciones climatologías se podrán realizar acciones con un control de nivel superior, en centros comerciales, grandes almacenes, hospitales, fábricas ...

A continuación se procederá a exponer las conclusiones obtenidas como consecuencia de la elaboración de este TFG y de los estudios realizados con respecto a la FPGA y con el sensor utilizado para las mediciones.

- Se ha realizado el desarrollo, simulación e implementación del sistema digital, en la FPGA, de la familia MACHOX2 de Lattice, pudiendo desarrollar la lectura de los datos de forma sencilla y muy visual.
- Se han utilizado técnicas de división del trabajo para llevar a cabo la aplicación de una forma conjunta, fortaleciendo las capacidades de trabajo en equipo y desarrollo colaborativo.

- Se ha realizado un estudio un estudio en el mercado de las posibles FPGAs utilizables para seleccionar la más adecuada para la aplicación en cuestión.

- Se ha realizado un estudio de las alternativas del mercado para la selección del sensor de forma que se pueda elegir el sensor más adecuado a las necesidades tanto económicas como

- Se ha realizado un estudio de los recursos empleados en la FPGA de forma que se puede ver el grado de ocupación de la aplicación dentro de la FPGA de cara a futuras líneas de desarrollo y su desarrollo comercial, ya que puede ser más adecuada para una FPGA de mayor o menor capacidad.

- El diseño se ha realizado de forma jerárquica y estructurada, empleando técnicas de desarrollo modernas. El desarrollo de bloques se ha generado utilizando esquemas, descripciones VHDL, bloques generados mediante IPexpress y máquinas de estado generadas mediante Qfsm.

- Se ha desarrollado la posibilidad de dividir los bloques funcionales para poder realizar la trasmisión de datos y completar la ejecución de la aplicación en otro dispositivo.

- Se ha generado la implantación del sistema completo en un prototipo, comprobando su completa funcionalidad y viabilidad.

- Se ha implementado funciones dentro de los bloques funcionales para almacenar valores máximos y mínimos y poder seleccionar la información a mostrar con entradas de la FPGA.

A Continuación se procederá a plantear las posibles líneas de desarrollo de cara a futuras mejoras de la aplicación:

- Optimización de los bloques funcionales debido a que seguramente existan formas con un empleo de recursos inferior, ya que la prioridad en esta aplicación no es la velocidad, ya que no es una apliación que precise un reloj y un tiempo de procesamiento muy altos.

- Añadir más grupo de sensores como el BMP 180 que mide la presión atmosférica y la temperatura para obtener una medida más fiable se pueden comparar.
- Posibilidad de incorporar un lector LCD para la visualización de la información de cara a facilitar la interpretación por parte del usuario.
- Optimización del diseño para integrarlo dentro de una una PCB, optimizando el espacio que ocupa el dispositivo.
- Desarrollo de una embolvente para mantener el dispositivo protegido frente a golpes ya que se tratará un dispositvo de exterior, es conveniente que cuente por lo menos con un tipo protección IP20 o superior.

Parte V

Bibliografia

- https://es.wikipedia.org/wiki/Field-programmable_gate_array

Consultada última vez: 22/06/18

- https://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf

Consultada última vez: 16/05/18

- https://www.xilinx.com/

Consultada última vez: 03/06/18

- https://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

Consultada última vez: 03/06/18

- https://www.waveshare.com/open3s500e-standard.htm

Consultada última vez: 04/07/18

- http://www.latticesemi.com/

Consultada última vez: 03/06/18

- http://www.latticesemi.com/en/Products/FPGAandCPLD/MachXO2#_3D24D0EE

Consultada última vez: 03/06/18

- http://statics3.seeedstudio.com/assets/file/bazaar/product/HC-12_english_datashee

Consultada última vez: 18/05/18

- https://www.dfrobot.com/wiki/index.php/Weather_Station_with_Anemometer/Wi

Consultada última vez: 18/05/18

- http://www.stcmcu.com/datasheet/stc/stc-ad-pdf/stc12c5a60s2-english.pdf

Consultada última vez: 20/05/18

- http://docplayer.es/10812141-Estado-del-arte-de-la-tecnologia-fpga.html

Consultada última vez: 13/06/18

- https://programarfacil.com/podcast/82-escoger-mejor-sensor-temperatura-arduino/

Consultada última vez: 07/05/18

- https://es.rs-online.com/web/

Consultada última vez: 29/06/18

- https://www.mouser.es/

Consultada última vez: 29/06/18

- https://es.dhgate.com/wholesale/electronic+humidity+sensor.html

Consultada última vez: 15/05/18

Parte VI

Anejos

Capítulo 9

Anejo 1: Estudio Económico y Estudio de utilización de Recursos.

A continuación se procederá a realizar una estimación del plano económico para la realización del dispositivo.

La selección de la FPGA sería similar en cuanto a capacidades internas de LUTS, pero el producto comercial no necesita tantos pines y como existen versiones con una distribución inferior de pines, estas serían las seleccionadas para el producto.

En la realización del producto no se tendrían en cuenta los precios unitarios sino los precios unitarios de la compra al por mayor, ya que interesa una producción alta de productos, para generar beneficios con un coste asequible.

La FPGA MachXO2 1200 ZE que es la misma que se ha utilizado en el prototipo salvando la diferencia de 19 I/O con tecnología SMD. La unidad tiene un coste 3,71€ con la compra de 5000 unidades.

El sensor DHT 22 igual al utilizado en el prototipo tiene un precio unitario de 2,20€ *sin contar con la resistencia de pull-up que habría que añadir 2 centimos al precio unitario con pedidos de 10000 unidades.*

El módulo bluetooth HC-12 tiene un coste unitario de 2,43€ y permite tener un alcance hasta 1000m con otro dispositivo bluetooth de las mismas características.

La PCB de ensamble tiene aproximadamente un precio unitario de unos 0,30€ para las placas.

componente	precio unitario
Machxo2	3,71€
DHT 22	2,20€
Resistencia	0,02€
HC-12	2,43€
PCB + ensamble	0,30€
TOTAL	10,56€

El coste total por dispositivo son 10,56€ un precio comercial podría ser de aproximadamente 11,99€ dejando unos beneficios del 12 % para cubrir los gastos de ingeniería y comercialización del dispositivo.

A continuación se procederá a realizar el estudio de recursos, tanto de los ya consumidos como son horas de ingeniería, como los consumidos con la FPGA. Se ha tomado como base la FPGA 1200ZE, que se ha utilizado para el prototipo, de forma que se podrá ver si es el dispositivo idóneo o por el contrarío para el diseño final será necesario un cambio de dispositivo.

Bloque funcional	Ocupación FPGA (SLices)	Reloj PLL	PIO
reloj	0	1	0
Control Global	<1 %	0	0
States Moore	1 %	0	0
Barrel Shifter	3 %	0	0
Suma Check	21 %	0	0
Bin 2 BCD	34 %	0	0
BCD 7Seg	1 %	0	0
TOTAL	61 %	1	38 %

La utilización de pines unicamente se tiene en cuenta en el diseño final, ya que este es el único que necesita las entradas y salidas hacia el exterior de la placa. Las unidades lógicas tenidas en cuenta son Slices que estos contienen 2 LUT4.

Como la ocupación en la FPGA es mayor del 50 % no se puede optar por una FPGA inferior, ya que el diseño no entraría por utilización de unidades lógicas. Una FPGA de mayor capacidad no sería necesario ya que incrementa el precio y provoca que desaprovechen recursos. Para futuros diseños y ampliaciones de funciones se podrá necesitar un dispositivo de características superiores, pero actualmente no es necesario ya que todavía queda casi un 40 % de la FPGA

disponible para mejoras.

La parte correspodiente al trabajo de desarrollo del dispositivo corresponde principalmente a horas de ingeniería ya que para realizar el dispositivo ha necesitado un desarrollo para la parte de programación del dispositivo. Se estiman unas 400h de desarrollo por parte del departamento de ingeniería.

Capítulo 10

Anejo 2: Resumen TFG comunicación de FPGAs mediante Bluetooth

El funcionamiento de este tfg no es completo ya que obliga a una lectura de los datos de forma directa, lo cual a veces no es posible, ya sea por la climatología o el lugar donde esta la propia estación de medida. Por lo tanto, sería muy interesante poder comunicar dicha información hacia otro dispositivo utilizando el espectro electromagnético, ya que es el más común y más fácil de implementar.

El desarrollo del tfg descrito se complementa perfectamente con otro tfg basado en comunicación mediante un módulo bluetooth de dos dispositivos FPGAs. Al generar un canal de trasmisión de los datos a través de la distancia, ya no es necesario tener una limitación física como sería el bus de comunicación para la transmisión de dicha información.

El tfg de estación meteorológica permite el envío de información en varios puntos ideales para ello, ya sea por la condición de estas que poseen una propia señal de fin de lectura de datos, o fin de procesamiento de estos. Los momentos idóneos para la trasmisión serían una vez la información es procesada y obtenida del módulo de forma que en tal caso enviaríamos cadenas de 40 bits donde sería necesario un post procesamiento para comprobar la coherencia de los datos. Otro momento ideal para la transmisión de estos es justo después del primer procesado de los datos donde ya se conoce si los datos son o no correctos.

La primera forma propuesta envía siempre los datos que interpreta del sensor y esto puede provocar que se produzcan errores ya que estos se pueden producir tanto en la lectura del módulo como en transmisión, ya que, a mayor número de bits transmitidos, mayor serán la probabilidad de que suceda un error durante la transmisión.

Si la transmisión se produce después del evitaría transmisiones innecesarias de todo el conjunto de bits ya que esto supone una pérdida de energía y tiempo de procesamiento por parte del módulo emisor y receptor.

Para la transmisión de los datos a través de dispositivos se ha optado por el uso de un módulo bluetooth con frecuencias de transmisión del orden de 433MHz, esta frecuencia no esta tomada al azar, ya que el espectro electromagnético es responsabilidad del estado y la transmisión a otro rango de frecuencias no es recomendable, ya que otras frecuencias son utilizadas por las ondas de radio, televisión, redes móviles. El uso de estas frecuencias provocará que recibamos informaciones erróneas al solaparse con transmisiones ajenas a nuestras emisiones.

Para llevar a cabo la trasmisión de los datos se ha implementado un bloque que envía la información al módulo bluetooth y este trabajando en modo trasparente la reenvía al otro módulo de forma que este de igual manera lo recibe y lo pasa a la otra FPGA a la cual está conectado. Para la emisión de los datos se ha seleccionado una frecuencia de 9600 baudios, esta frecuencia no esta disponible como frecuencia de reloj por lo que se ha de implementar un contador que reduzca la frecuencia del reloj de 20uS.

Una vez trasmitidos los datos la interpretación puede realizarse de forma idéntica a la planteada en el trabajo de Estación Meteorológica, mostrando a través de los de la FPGA y los displays, aunque para el trabajo de la comunicación bluetooth se ha implementado una aplicación movil de forma que permite visualizar los datos de una forma visual y permitiendo visualizar un histórico con valores leídos por parte del sensor.

Capítulo 11

Anejo 3: FPGA MACHXO2

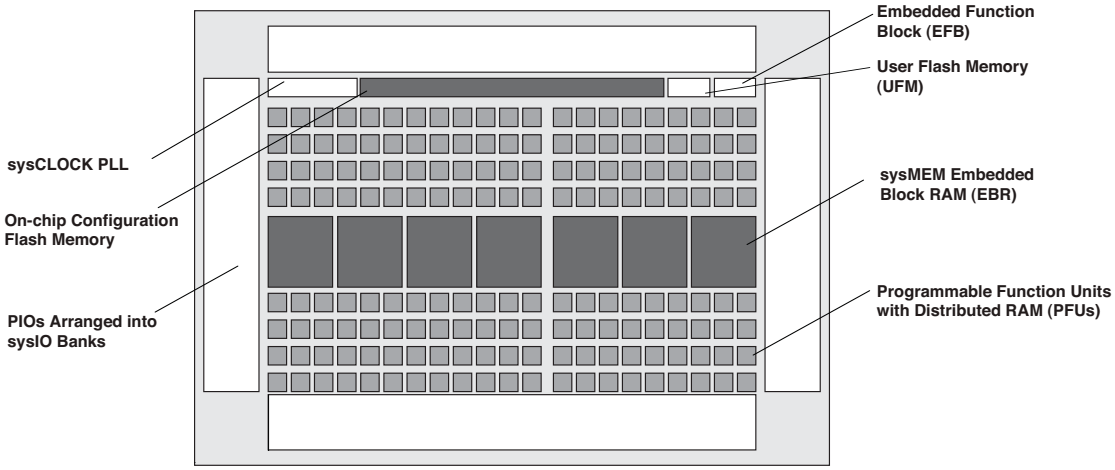
MachXO2 Family Data Sheet Architecture

March 2016 Data Sheet DS1035

Architecture Overview

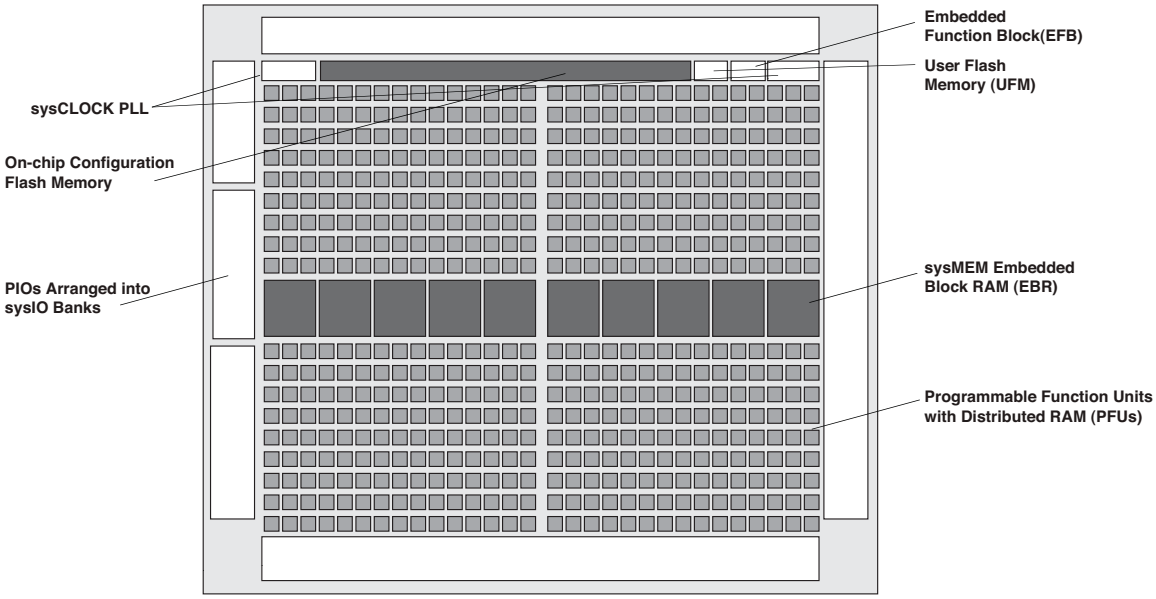
The MachXO2 family architecture contains an array of logic blocks surrounded by Programmable I/O (PIO). The larger logic density devices in this family have sysCLOCK™ PLLs and blocks of sysMEM Embedded Block RAM (EBRs). Figure 2-1 and Figure 2-2 show the block diagrams of the various family members.

Figure 2-1. Top View of the MachXO2-1200 Device



Note: MachXO2-256, and MachXO2-640/U are similar to MachXO2-1200. MachXO2-256 has a lower LUT count and no PLL or EBR blocks. MachXO2-640 has no PLL, a lower LUT count and two EBR blocks. MachXO2-640U has a lower LUT count, one PLL and seven EBR blocks.

Figure 2-2. Top View of the MachXO2-4000 Device



Note: MachXO2-1200U, MachXO2-2000/U and MachXO2-7000 are similar to MachXO2-4000. MachXO2-1200U and MachXO2-2000 have a lower LUT count, one PLL, and eight EBR blocks. MachXO2-2000U has a lower LUT count, two PLLs, and 10 EBR blocks. MachXO2-7000 has a higher LUT count, two PLLs, and 26 EBR blocks.

The logic blocks, Programmable Functional Unit (PFU) and sysMEM EBR blocks, are arranged in a two-dimensional grid with rows and columns. Each row has either the logic blocks or the EBR blocks. The PIO cells are located at the periphery of the device, arranged in banks. The PFU contains the building blocks for logic, arithmetic, RAM, ROM, and register functions. The PIOs utilize a flexible I/O buffer referred to as a sysIO buffer that supports operation with a variety of interface standards. The blocks are connected with many vertical and horizontal routing channel resources. The place and route software tool automatically allocates these routing resources.

In the MachXO2 family, the number of sysIO banks varies by device. There are different types of I/O buffers on the different banks. Refer to the details in later sections of this document. The sysMEM EBRs are large, dedicated fast memory blocks; these blocks are found in MachXO2-640/U and larger devices. These blocks can be configured as RAM, ROM or FIFO. FIFO support includes dedicated FIFO pointer and flag “hard” control logic to minimize LUT usage.

The MachXO2 registers in PFU and sysI/O can be configured to be SET or RESET. After power up and device is configured, the device enters into user mode with these registers SET/RESET according to the configuration setting, allowing device entering to a known state for predictable system function.

The MachXO2 architecture also provides up to two sysCLOCK Phase Locked Loop (PLL) blocks on MachXO2-640U, MachXO2-1200/U and larger devices. These blocks are located at the ends of the on-chip Flash block. The PLLs have multiply, divide, and phase shifting capabilities that are used to manage the frequency and phase relationships of the clocks.

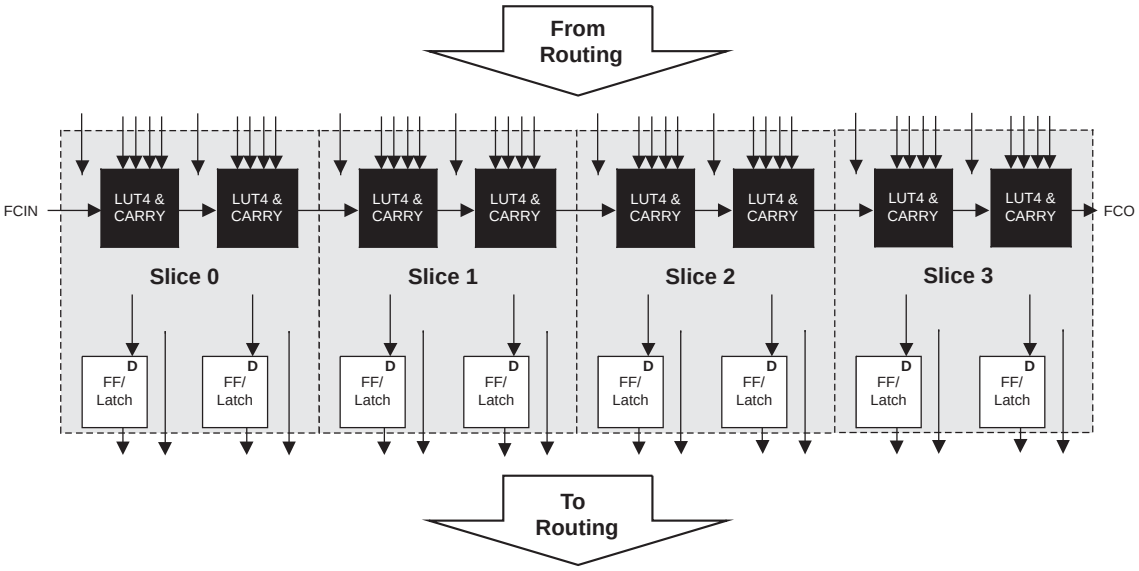
MachXO2 devices provide commonly used hardened functions such as SPI controller, I^2C controller and timer/counter. MachXO2-640/U and higher density devices also provide User Flash Memory (UFM). These hardened functions and the UFM interface to the core logic and routing through a WISHBONE interface. The UFM can also be accessed through the SPI, I^2C and JTAG ports.

Every device in the family has a JTAG port that supports programming and configuration of the device as well as access to the user logic. The MachXO2 devices are available for operation from 3.3 V, 2.5 V and 1.2 V power supplies, providing easy integration into the overall system.

PFU Blocks

The core of the MachXO2 device consists of PFU blocks, which can be programmed to perform logic, arithmetic, distributed RAM and distributed ROM functions. Each PFU block consists of four interconnected slices numbered 0 to 3 as shown in Figure 2-3. Each slice contains two LUTs and two registers. There are 53 inputs and 25 outputs associated with each PFU block.

Figure 2-3. PFU Block Diagram



Slices

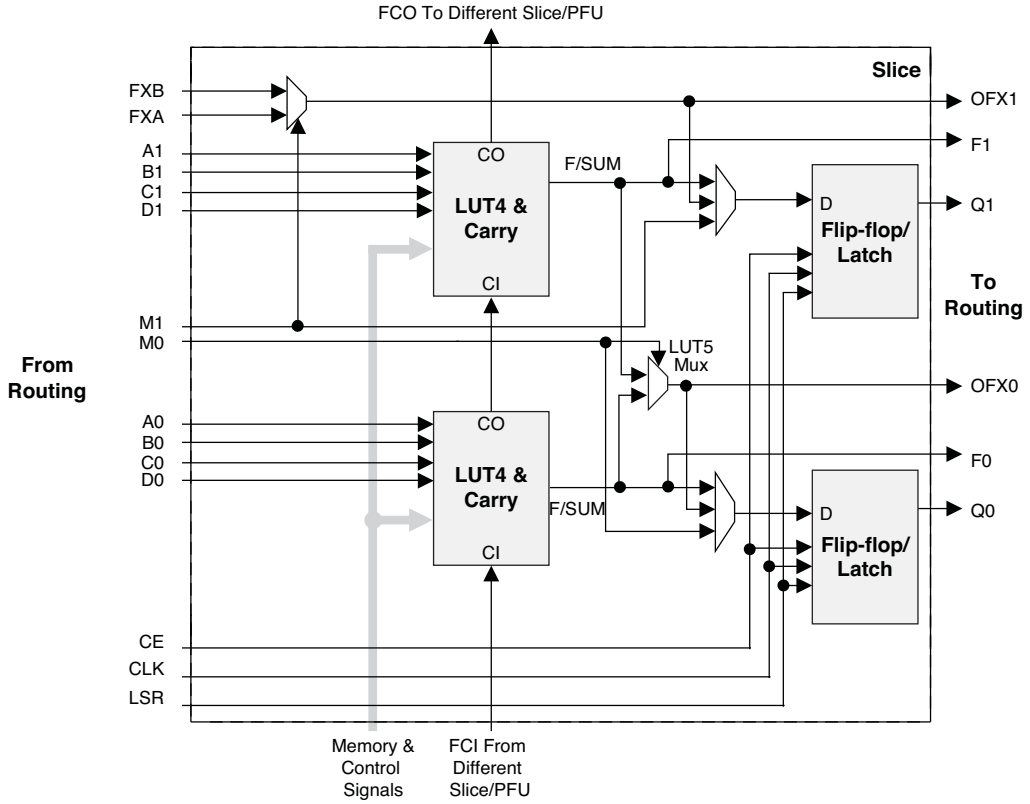
Slices 0-3 contain two LUT4s feeding two registers. Slices 0-2 can be configured as distributed memory. Table 2-1 shows the capability of the slices in PFU blocks along with the operation modes they enable. In addition, each PFU contains logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7 and LUT8. The control logic performs set/reset functions (programmable as synchronous/ asynchronous), clock select, chip-select and wider RAM/ROM functions.

Table 2-1. Resources and Modes Available per Slice

Slice	PFU Block	
	Resources	Modes
Slice 0	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM
Slice 1	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM
Slice 2	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM
Slice 3	2 LUT4s and 2 Registers	Logic, Ripple, ROM

Figure 2-4 shows an overview of the internal logic of the slice. The registers in the slice can be configured for positive/negative and edge triggered or level sensitive clocks. All slices have 15 inputs from routing and one from the carry-chain (from the adjacent slice or PFU). There are seven outputs: six for routing and one to carry-chain (to the adjacent PFU). Table 2-2 lists the signals associated with Slices 0-3.

Figure 2-4. Slice Diagram



For Slices 0 and 1, memory control signals are generated from Slice 2 as follows:

- WCK is CLK
- WRE is from LSR
- DI[3:2] for Slice 1 and DI[1:0] for Slice 0 data from Slice 2
- WAD [A:D] is a 4-bit address from slice 2 LUT input

Table 2-2. Slice Signal Descriptions

Function	Type	Signal Names	Description
Input	Data signal	A0, B0, C0, D0	Inputs to LUT4
Input	Data signal	A1, B1, C1, D1	Inputs to LUT4
Input	Multi-purpose	M0/M1	Multi-purpose input
Input	Control signal	CE	Clock enable
Input	Control signal	LSR	Local set/reset
Input	Control signal	CLK	System clock
Input	Inter-PFU signal	FCIN	Fast carry in[1]
Output	Data signals	F0, F1	LUT4 output register bypass signals
Output	Data signals	Q0, Q1	Register outputs
Output	Data signals	OFX0	Output of a LUT5 MUX
Output	Data signals	OFX1	Output of a LUT6, LUT7, LUT8[2] MUX depending on the slice
Output	Inter-PFU signal	FCO	Fast carry out[1]

1. See Figure 2-3 for connection details.
2. Requires two PFUs.

Modes of Operation

Each slice has up to four potential modes of operation: Logic, Ripple, RAM and ROM.

Logic Mode
In this mode, the LUTs in each slice are configured as 4-input combinatorial lookup tables. A LUT4 can have 16 possible input combinations. Any four input logic functions can be generated by programming this lookup table. Since there are two LUT4s per slice, a LUT5 can be constructed within one slice. Larger look-up tables such as LUT6, LUT7 and LUT8 can be constructed by concatenating other slices. Note LUT8 requires more than four slices.

Ripple Mode
Ripple mode supports the efficient implementation of small arithmetic functions. In Ripple mode, the following functions can be implemented by each slice:

- Addition 2-bit
- Subtraction 2-bit
- Add/subtract 2-bit using dynamic control
- Up counter 2-bit
- Down counter 2-bit
- Up/down counter with asynchronous clear
- Up/down counter with preload (sync)
- Ripple mode multiplier building block
- Multiplier support
- Comparator functions of A and B inputs
  — A greater-than-or-equal-to B
  — A not-equal-to B
  — A less-than-or-equal-to B

Ripple mode includes an optional configuration that performs arithmetic using fast carry chain methods. In this configuration (also referred to as CCU2 mode) two additional signals, Carry Generate and Carry Propagate, are generated on a per-slice basis to allow fast arithmetic functions to be constructed by concatenating slices.

RAM Mode

In this mode, a 16x4-bit distributed single port RAM (SPR) can be constructed by using each LUT block in Slice 0 and Slice 1 as a 16x1-bit memory. Slice 2 is used to provide memory address and control signals.

MachXO2 devices support distributed memory initialization.

The Lattice design tools support the creation of a variety of different size memories. Where appropriate, the software will construct these using distributed memory primitives that represent the capabilities of the PFU. Table 2-3 shows the number of slices required to implement different distributed RAM primitives. For more information about using RAM in MachXO2 devices, please see TN1201, Memory Usage Guide for MachXO2 Devices.

Table 2-3. Number of Slices Required For Implementing Distributed RAM

	SPR 16x4	PDPR 16x4
Number of slices	3	3

Note: SPR = Single Port RAM, PDPR = Pseudo Dual Port RAM

ROM Mode

ROM mode uses the LUT logic; hence, slices 0-3 can be used in ROM mode. Preloading is accomplished through the programming interface during PFU configuration.

For more information on the RAM and ROM modes, please refer to TN1201, Memory Usage Guide for MachXO2 Devices.

Routing

There are many resources provided in the MachXO2 devices to route signals individually or as buses with related control signals. The routing resources consist of switching circuitry, buffers and metal interconnect (routing) segments.

The inter-PFU connections are made with three different types of routing resources: x1 (spans two PFUs), x2 (spans three PFUs) and x6 (spans seven PFUs). The x1, x2, and x6 connections provide fast and efficient connections in the horizontal and vertical directions.

The design tools take the output of the synthesis tool and places and routes the design. Generally, the place and route tool is completely automatic, although an interactive routing editor is available to optimize the design.

Clock/Control Distribution Network

Each MachXO2 device has eight clock inputs (PCLK [T, C] [Banknum]_[2..0]) – three pins on the left side, two pins each on the bottom and top sides and one pin on the right side. These clock inputs drive the clock nets. These eight inputs can be differential or single-ended and may be used as general purpose I/O if they are not used to drive the clock nets. When using a single ended clock input, only the PCLKT input can drive the clock tree directly.

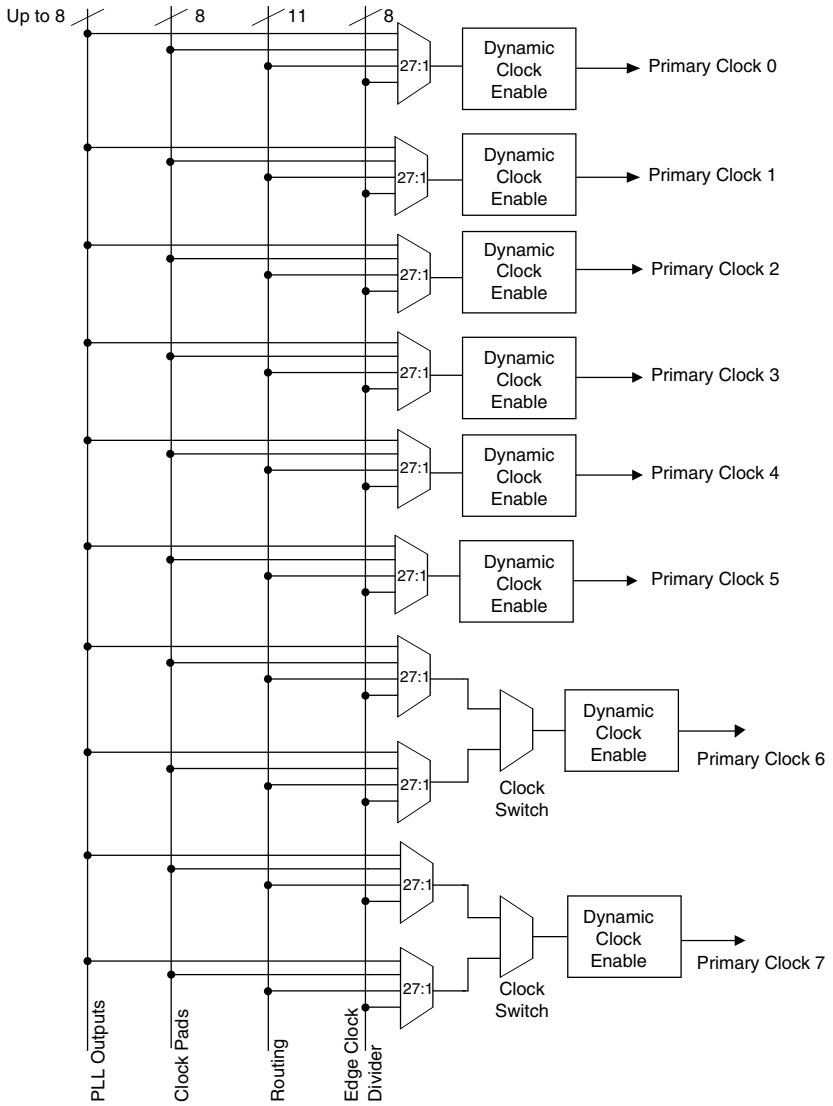
The MachXO2 architecture has three types of clocking resources: edge clocks, primary clocks and secondary high fanout nets. MachXO2-640U, MachXO2-1200/U and higher density devices have two edge clocks each on the top and bottom edges. Lower density devices have no edge clocks. Edge clocks are used to clock I/O registers and have low injection time and skew. Edge clock inputs are from PLL outputs, primary clock pads, edge clock bridge outputs and CIB sources.

The eight primary clock lines in the primary clock network drive throughout the entire device and can provide clocks for all resources within the device including PFUs, EBRs and PICs. In addition to the primary clock signals, MachXO2 devices also have eight secondary high fanout signals which can be used for global control signals, such as clock enables, synchronous or asynchronous clears, presets, output enables, etc. Internal logic can drive the global clock network for internally-generated global clocks and control signals.

The maximum frequency for the primary clock network is shown in the MachXO2 External Switching Characteristics table.

The primary clock signals for the MachXO2-256 and MachXO2-640 are generated from eight 17:1 muxes The available clock sources include eight I/O sources and 9 routing inputs. Primary clock signals for the MachXO2-640U, MachXO2-1200/U and larger devices are generated from eight 27:1 muxes The available clock sources include eight I/O sources, 11 routing inputs, eight clock divider inputs and up to eight sysCLOCK PLL outputs.

Figure 2-5. Primary Clocks for MachXO2 Devices

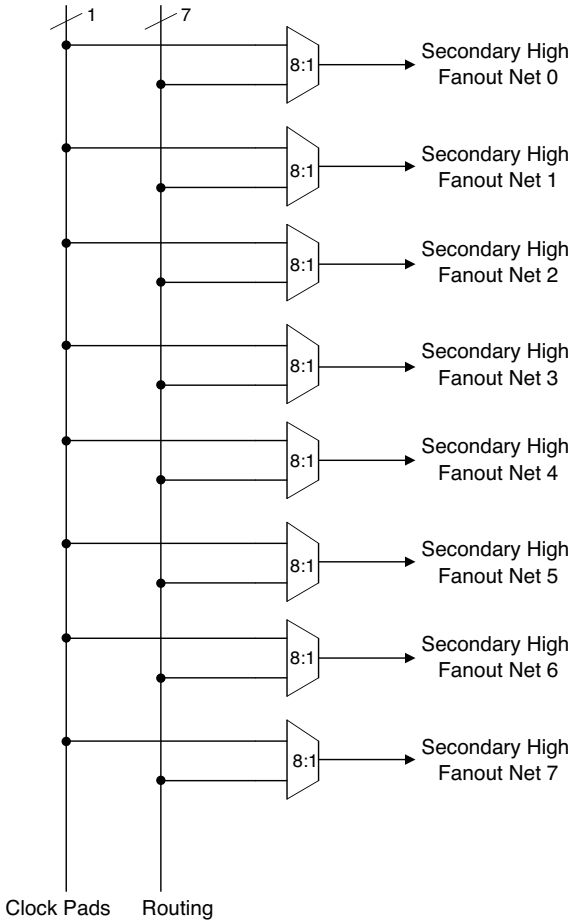


Primary clocks for MachXO2-640U, MachXO2-1200/U and larger devices.

Note: MachXO2-640 and smaller devices do not have inputs from the Edge Clock Divider or PLL and fewer routing inputs. These devices have 17:1 muxes instead of 27:1 muxes.

Eight secondary high fanout nets are generated from eight 8:1 muxes as shown in Figure 2-6. One of the eight inputs to the secondary high fanout net input mux comes from dual function clock pins and the remaining seven come from internal routing. The maximum frequency for the secondary clock network is shown in MachXO2 External Switching Characteristics table.

Figure 2-6. Secondary High Fanout Nets for MachXO2 Devices



sysCLOCK Phase Locked Loops (PLLs)

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. The MachXO2-640U, MachXO2-1200/U and larger devices have one or more sysCLOCK PLL. CLKI is the reference frequency input to the PLL and its source can come from an external I/O pin or from internal routing. CLKFB is the feedback signal to the PLL which can come from internal routing or an external I/O pin. The feedback divider is used to multiply the reference frequency and thus synthesize a higher frequency clock output.

The MachXO2 sysCLOCK PLLs support high resolution (16-bit) fractional-N synthesis. Fractional-N frequency synthesis allows the user to generate an output clock which is a non-integer multiple of the input frequency. For more information about using the PLL with Fractional-N synthesis, please see TN1199, MachXO2 sysCLOCK PLL Design and Usage Guide.

Each output has its own output divider, thus allowing the PLL to generate different frequencies for each output. The output dividers can have a value from 1 to 128. The output dividers may also be cascaded together to generate low frequency clocks. The CLKOP, CLKOS, CLKOS2, and CLKOS3 outputs can all be used to drive the MachXO2 clock distribution network directly or general purpose routing resources can be used.

The LOCK signal is asserted when the PLL determines it has achieved lock and de-asserted if a loss of lock is detected. A block diagram of the PLL is shown in Figure 2-7.

The setup and hold times of the device can be improved by programming a phase shift into the CLKOS, CLKOS2, and CLKOS3 output clocks which will advance or delay the output clock with reference to the CLKOP output clock.

This phase shift can be either programmed during configuration or can be adjusted dynamically. In dynamic mode, the PLL may lose lock after a phase adjustment on the output used as the feedback source and not relock until the t_{LOCK} parameter has been satisfied.

The MachXO2 also has a feature that allows the user to select between two different reference clock sources dynamically. This feature is implemented using the PLLREFCS primitive. The timing parameters for the PLL are shown in the sysCLOCK PLL Timing table.

The MachXO2 PLL contains a WISHBONE port feature that allows the PLL settings, including divider values, to be dynamically changed from the user logic. When using this feature the EFB block must also be instantiated in the design to allow access to the WISHBONE ports. Similar to the dynamic phase adjustment, when PLL settings are updated through the WISHBONE port the PLL may lose lock and not relock until the t_{LOCK} parameter has been satisfied. The timing parameters for the PLL are shown in the sysCLOCK PLL Timing table.

For more details on the PLL and the WISHBONE interface, see TN1199, MachXO2 sysCLOCK PLL Design and Usage Guide.

Figure 2-7. PLL Diagram

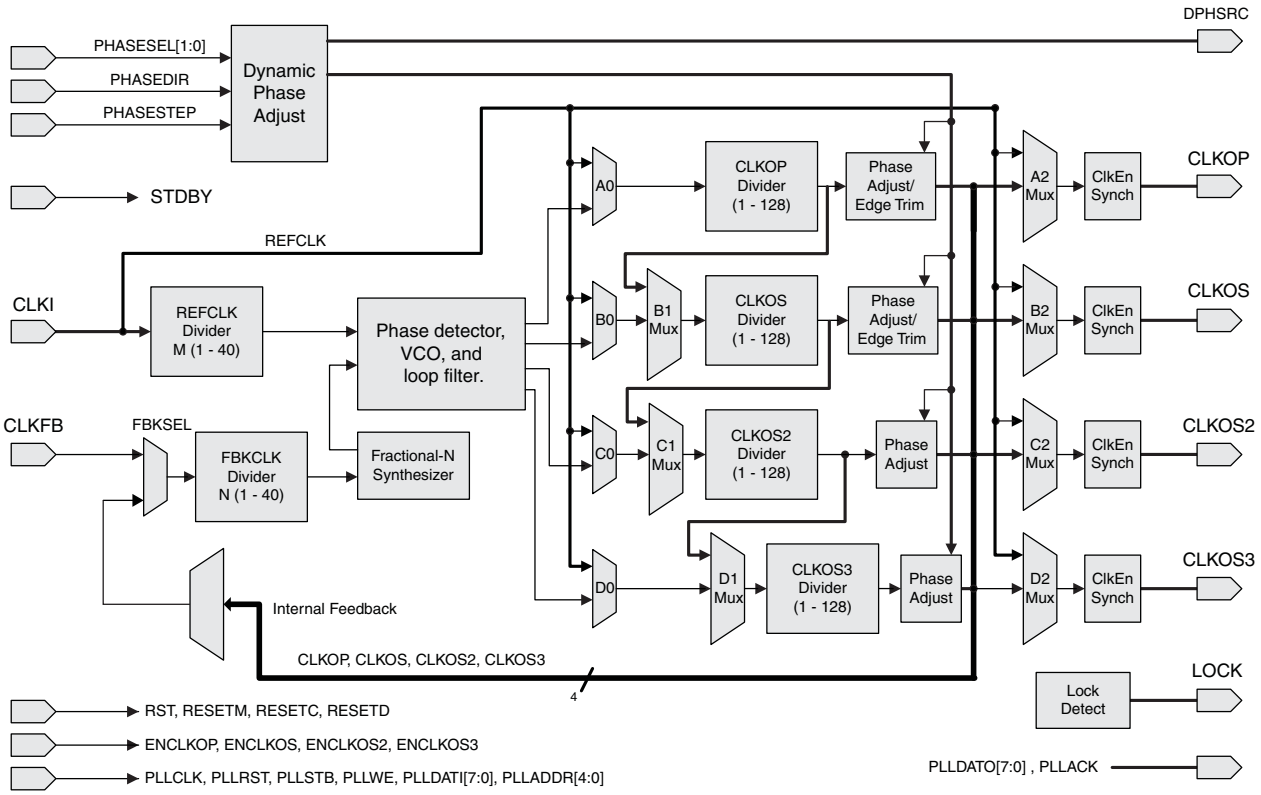


Table 2-4 provides signal descriptions of the PLL block.

Table 2-4. PLL Signal Descriptions

Port Name	I/O	Description
CLKI	I	Input clock to PLL
CLKFB	I	Feedback clock
PHASESEL[1:0]	I	Select which output is affected by Dynamic Phase adjustment ports
PHASEDIR	I	Dynamic Phase adjustment direction
PHASESTEP	I	Dynamic Phase step – toggle shifts VCO phase adjust by one step.

Table 2-4. PLL Signal Descriptions (Continued)

Port Name	I/O	Description
CLKOP	O	Primary PLL output clock (with phase shift adjustment)
CLKOS	O	Secondary PLL output clock (with phase shift adjust)
CLKOS2	O	Secondary PLL output clock2 (with phase shift adjust)
CLKOS3	O	Secondary PLL output clock3 (with phase shift adjust)
LOCK	O	PLL LOCK, asynchronous signal. Active high indicates PLL is locked to input and feedback signals.
DPHSRC	O	Dynamic Phase source – ports or WISHBONE is active
STDBY	I	Standby signal to power down the PLL
RST	I	PLL reset without resetting the M-divider. Active high reset.
RESETM	I	PLL reset - includes resetting the M-divider. Active high reset.
RESETC	I	Reset for CLKOS2 output divider only. Active high reset.
RESETD	I	Reset for CLKOS3 output divider only. Active high reset.
ENCLKOP	I	Enable PLL output CLKOP
ENCLKOS	I	Enable PLL output CLKOS when port is active
ENCLKOS2	I	Enable PLL output CLKOS2 when port is active
ENCLKOS3	I	Enable PLL output CLKOS3 when port is active
PLLCLK	I	PLL data bus clock input signal
PLLRST	I	PLL data bus reset. This resets only the data bus not any register values.
PLLSTB	I	PLL data bus strobe signal
PLLWE	I	PLL data bus write enable signal
PLLADDR [4:0]	I	PLL data bus address
PLLDATI [7:0]	I	PLL data bus data input
PLLDATO [7:0]	O	PLL data bus data output
PLLACK	O	PLL data bus acknowledge signal

sysMEM Embedded Block RAM Memory

The MachXO2-640/U and larger devices contain sysMEM Embedded Block RAMs (EBRs). The EBR consists of a 9-kbit RAM, with dedicated input and output registers. This memory can be used for a wide variety of purposes including data buffering, PROM for the soft processor and FIFO.

sysMEM Memory Block

The sysMEM block can implement single port, dual port, pseudo dual port, or FIFO memories. Each block can be used in a variety of depths and widths as shown in Table 2-5.

Table 2-5. sysMEM Block Configurations

Memory Mode	Configurations
Single Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9
True Dual Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9
Pseudo Dual Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9 512 x 18
FIFO	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9 512 x 18

Bus Size Matching

All of the multi-port memory modes support different widths on each of the ports. The RAM bits are mapped LSB word 0 to MSB word 0, LSB word 1 to MSB word 1, and so on. Although the word size and number of words for each port varies, this mapping scheme applies to each port.

RAM Initialization and ROM Operation

If desired, the contents of the RAM can be pre-loaded during device configuration. EBR initialization data can be loaded from the UFM. To maximize the number of UFM bits, initialize the EBRs used in your design to an all-zero pattern. Initializing to an all-zero pattern does not use up UFM bits. MachXO2 devices have been designed such that multiple EBRs share the same initialization memory space if they are initialized to the same pattern.

By preloading the RAM block during the chip configuration cycle and disabling the write controls, the sysMEM block can also be utilized as a ROM.

Memory Cascading

Larger and deeper blocks of RAM can be created using EBR sysMEM Blocks. Typically, the Lattice design tools cascade memory transparently, based on specific design inputs.

Single, Dual, Pseudo-Dual Port and FIFO Modes

Figure 2-8 shows the five basic memory configurations and their input/output names. In all the sysMEM RAM modes, the input data and addresses for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the memory array output.

Figure 2-8. sysMEM Memory Primitives

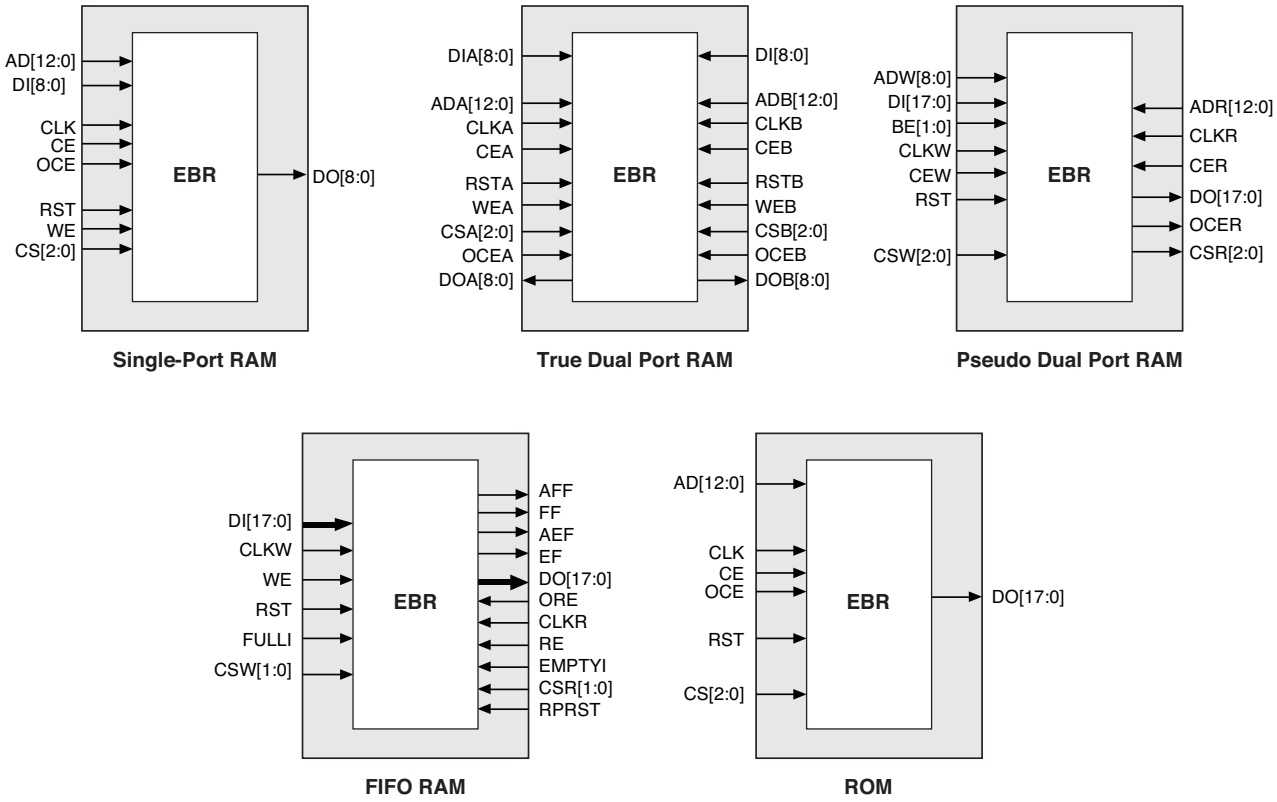


Table 2-6. EBR Signal Descriptions

Port Name	Description	Active State
CLK	Clock	Rising Clock Edge
CE	Clock Enable	Active High
OCE[1]	Output Clock Enable	Active High
RST	Reset	Active High
BE[1]	Byte Enable	Active High
WE	Write Enable	Active High
AD	Address Bus	—
DI	Data In	—
DO	Data Out	—
CS	Chip Select	Active High
AFF	FIFO RAM Almost Full Flag	—
FF	FIFO RAM Full Flag	—
AEF	FIFO RAM Almost Empty Flag	—
EF	FIFO RAM Empty Flag	—
RPRST	FIFO RAM Read Pointer Reset	—

1. Optional signals.
2. For dual port EBR primitives a trailing ‘A’ or ‘B’ in the signal name specifies the EBR port A or port B respectively.
3. For FIFO RAM mode primitive, a trailing ‘R’ or ‘W’ in the signal name specifies the FIFO read port or write port respectively.
4. For FIFO RAM mode primitive FULLI has the same function as CSW(2) and EMPTYI has the same function as CSR(2).
5. In FIFO mode, CLKW is the write port clock, CSW is the write port chip select, CLKR is the read port clock, CSR is the read port chip select, ORE is the output read enable.

The EBR memory supports three forms of write behavior for single or dual port operation:

1. **Normal** – Data on the output appears only during the read cycle. During a write cycle, the data (at the current address) does not appear on the output. This mode is supported for all data widths.
2. **Write Through** – A copy of the input data appears at the output of the same port. This mode is supported for all data widths.
3. **Read-Before-Write** – When new data is being written, the old contents of the address appears at the output.

FIFO Configuration

The FIFO has a write port with data-in, CEW, WE and CLKW signals. There is a separate read port with data-out, RCE, RE and CLKR signals. The FIFO internally generates Almost Full, Full, Almost Empty and Empty Flags. The Full and Almost Full flags are registered with CLKW. The Empty and Almost Empty flags are registered with CLKR. Table 2-7 shows the range of programming values for these flags.

Table 2-7. Programmable FIFO Flag Ranges

Flag Name	Programming Range
Full (FF)	1 to max (up to 2^N-1)
Almost Full (AF)	1 to Full-1
Almost Empty (AE)	1 to Full-1
Empty (EF)	0

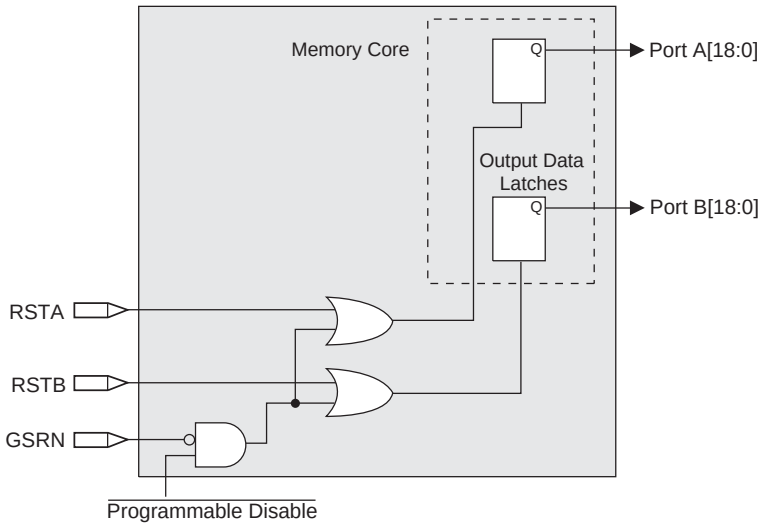
N = Address bit width.

The FIFO state machine supports two types of reset signals: RST and RPRST. The RST signal is a global reset that clears the contents of the FIFO by resetting the read/write pointer and puts the FIFO flags in their initial reset state. The RPRST signal is used to reset the read pointer. The purpose of this reset is to retransmit the data that is in the FIFO. In these applications it is important to keep careful track of when a packet is written into or read from the FIFO.

Memory Core Reset

The memory core contains data output latches for ports A and B. These are simple latches that can be reset synchronously or asynchronously. RSTA and RSTB are local signals, which reset the output latches associated with port A and port B respectively. The Global Reset (GSRN) signal resets both ports. The output data latches and associated resets for both ports are as shown in Figure 2-9.

Figure 2-9. Memory Core Reset

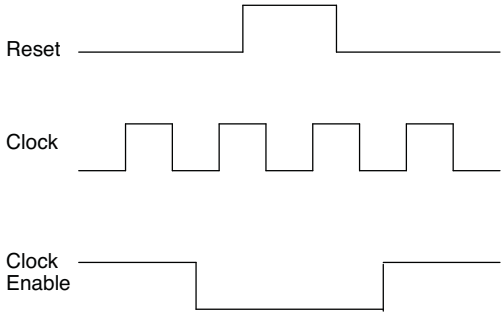


For further information on the sysMEM EBR block, please refer to TN1201, Memory Usage Guide for MachXO2 Devices.

EBR Asynchronous Reset

EBR asynchronous reset or GSR (if used) can only be applied if all clock enables are low for a clock cycle before the reset is applied and released a clock cycle after the reset is released, as shown in Figure 2-10. The GSR input to the EBR is always asynchronous.

Figure 2-10. EBR Asynchronous Reset (Including GSR) Timing Diagram



If all clock enables remain enabled, the EBR asynchronous reset or GSR may only be applied and released after the EBR read and write clock inputs are in a steady state condition for a minimum of $1/f_{MAX}$ (EBR clock). The reset release must adhere to the EBR synchronous reset setup time before the next active read or write clock edge.

If an EBR is pre-loaded during configuration, the GSR input must be disabled or the release of the GSR during device wake up must occur before the release of the device I/Os becoming active.

These instructions apply to all EBR RAM, ROM and FIFO implementations. For the EBR FIFO mode, the GSR signal is always enabled and the WE and RE signals act like the clock enable signals in Figure 2-10. The reset timing rules apply to the RPReset input versus the RE input and the RST input versus the WE and RE inputs. Both RST and RPReset are always asynchronous EBR inputs. For more details refer to TN1201, Memory Usage Guide for MachXO2 Devices.

Note that there are no reset restrictions if the EBR synchronous reset is used and the EBR GSR input is disabled.

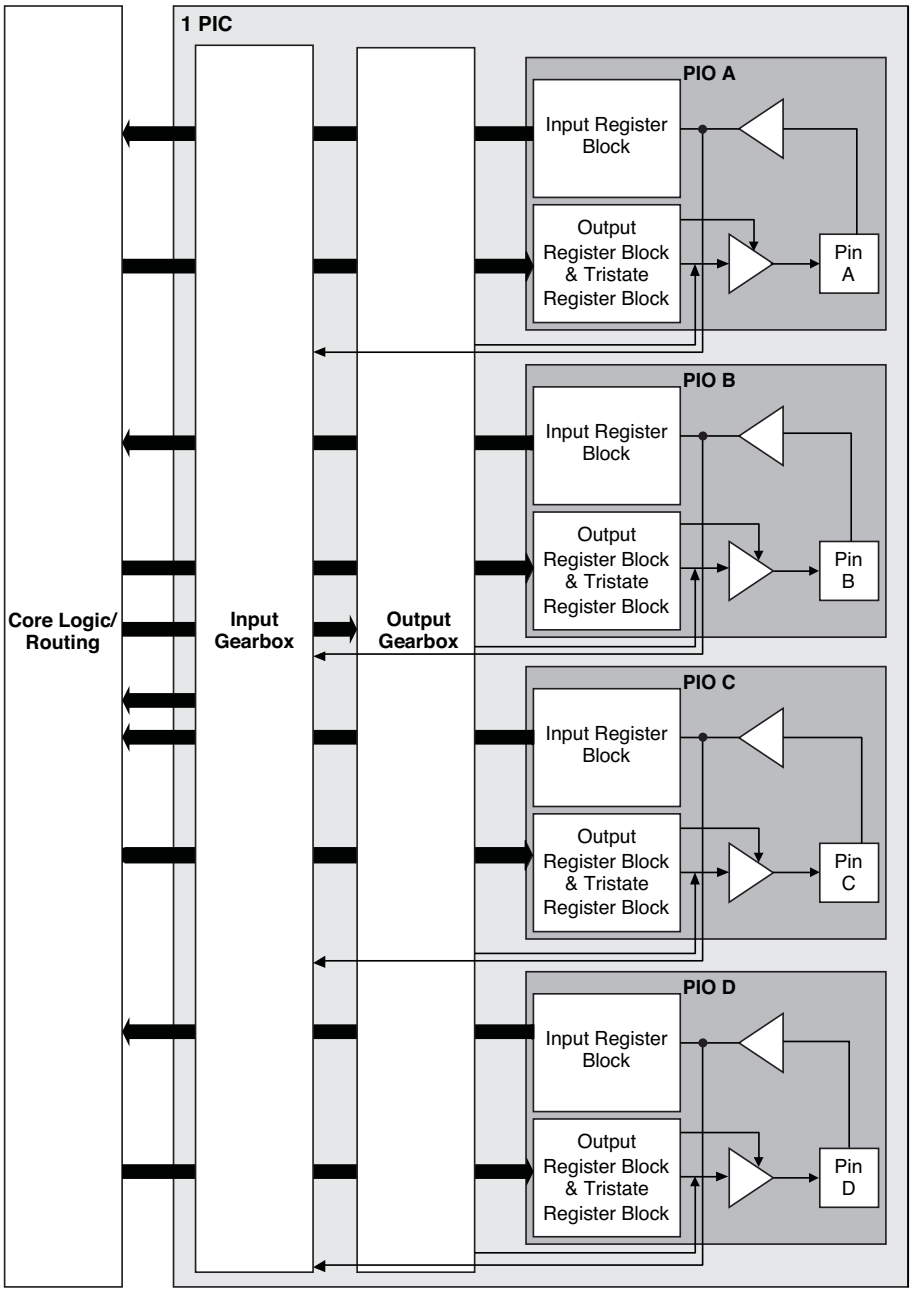
Programmable I/O Cells (PIC)

The programmable logic associated with an I/O is called a PIO. The individual PIO are connected to their respective sysIO buffers and pads. On the MachXO2 devices, the PIO cells are assembled into groups of four PIO cells called a Programmable I/O Cell or PIC. The PICs are placed on all four sides of the device.

On all the MachXO2 devices, two adjacent PIOs can be combined to provide a complementary output driver pair.

The MachXO2-640U, MachXO2-1200/U and higher density devices contain enhanced I/O capability. All PIO pairs on these larger devices can implement differential receivers. Half of the PIO pairs on the top edge of these devices can be configured as true LVDS transmit pairs. The PIO pairs on the bottom edge of these higher density devices have on-chip differential termination and also provide PCI support.

Figure 2-11. Group of Four Programmable I/O Cells



Notes:
1. Input gearbox is available only in PIC on the bottom edge of MachXO2-640U, MachXO2-1200/U and larger devices.
2. Output gearbox is available only in PIC on the top edge of MachXO2-640U, MachXO2-1200/U and larger devices.

PIO

The PIO contains three blocks: an input register block, output register block and tri-state register block. These blocks contain registers for operating in a variety of modes along with the necessary clock and selection logic.

Table 2-8. PIO Signal List

Pin Name	I/O Type	Description
CE	Input	Clock Enable
D	Input	Pin input from sysIO buffer.
INDD	Output	Register bypassed input.
INCK	Output	Clock input
Q0	Output	DDR positive edge input
Q1	Output	Registered input/DDR negative edge input
D0	Input	Output signal from the core (SDR and DDR)
D1	Input	Output signal from the core (DDR)
TD	Input	Tri-state signal from the core
Q	Output	Data output signals to sysIO Buffer
TQ	Output	Tri-state output signals to sysIO Buffer
DQSR90[1]	Input	DQS shift 90-degree read clock
DQSW90[1]	Input	DQS shift 90-degree write clock
DDRCLKPOL[1]	Input	DDR input register polarity control signal from DQS
SCLK	Input	System clock for input and output/tri-state blocks.
RST	Input	Local set reset signal

1. Available in PIO on right edge only.

Input Register Block

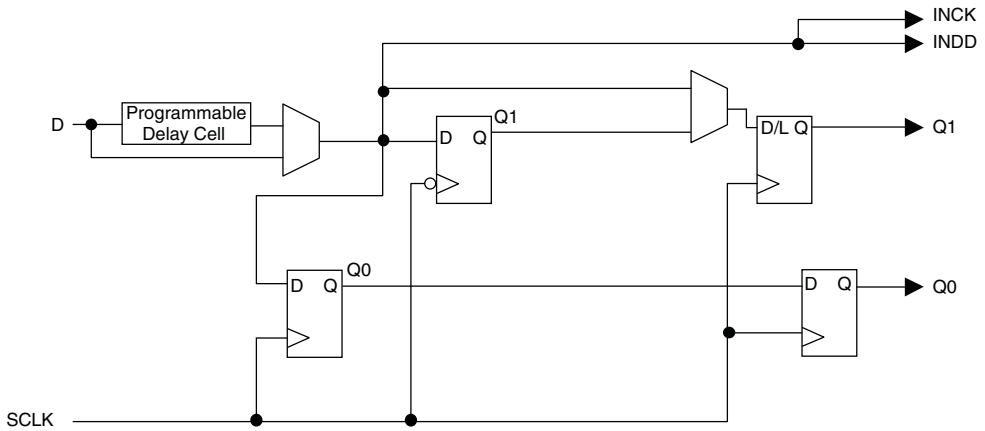
The input register blocks for the PIOs on all edges contain delay elements and registers that can be used to condition high-speed interface signals before they are passed to the device core. In addition to this functionality, the input register blocks for the PIOs on the right edge include built-in logic to interface to DDR memory.

Figure 2-12 shows the input register block for the PIOs located on the left, top and bottom edges. Figure 2-13 shows the input register block for the PIOs on the right edge.

Left, Top, Bottom Edges

Input signals are fed from the sysIO buffer to the input register block (as signal D). If desired, the input signal can bypass the register and delay elements and be used directly as a combinatorial signal (INDD), and a clock (INCK). If an input delay is desired, users can select a fixed delay. I/Os on the bottom edge also have a dynamic delay, DEL[4:0]. The delay, if selected, reduces input register hold time requirements when using a global clock. The input block allows two modes of operation. In single data rate (SDR) the data is registered with the system clock (SCLK) by one of the registers in the single data rate sync register block. In Generic DDR mode, two registers are used to sample the data on the positive and negative edges of the system clock (SCLK) signal, creating two data streams.

Figure 2-12. MachXO2 Input Register Block Diagram (PIO on Left, Top and Bottom Edges)



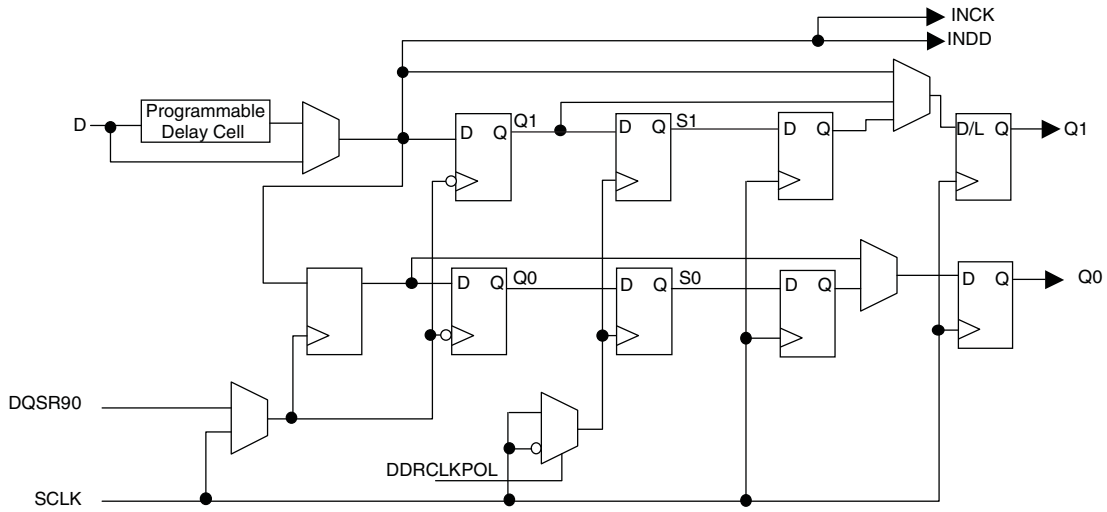
Right Edge

The input register block on the right edge is a superset of the same block on the top, bottom, and left edges. In addition to the modes described above, the input register block on the right edge also supports DDR memory mode.

In DDR memory mode, two registers are used to sample the data on the positive and negative edges of the modified DQS (DQSR90) in the DDR Memory mode creating two data streams. Before entering the core, these two data streams are synchronized to the system clock to generate two data streams.

The signal DDRCLKPOL controls the polarity of the clock used in the synchronization registers. It ensures adequate timing when data is transferred to the system clock domain from the DQS domain. The DQSR90 and DDRCLKPOL signals are generated in the DQS read-write block.

Figure 2-13. MachXO2 Input Register Block Diagram (PIO on Right Edge)



Capítulo 12

Anejo 4: Sensor DHT 22

AOSONG

Temperature and humidity module

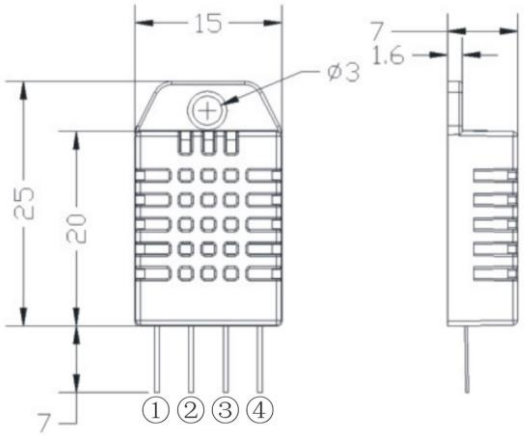
AM2302 Product Manual

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

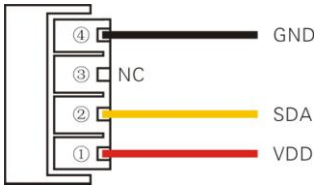
Ultra-low power, the transmission distance, fully automated calibration, the use of capacitive humidity sensor, completely interchangeable, standard digital single-bus output, excellent long-term stability, high accuracy temperature measurement devices.

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）

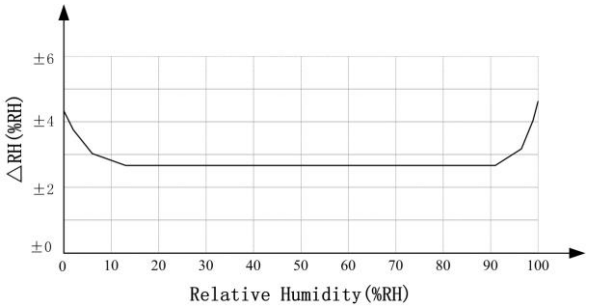
SDA pin is tri structure for reading, writing sensor data. Specific communication timing, see the detailed description of the communication protocol.

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

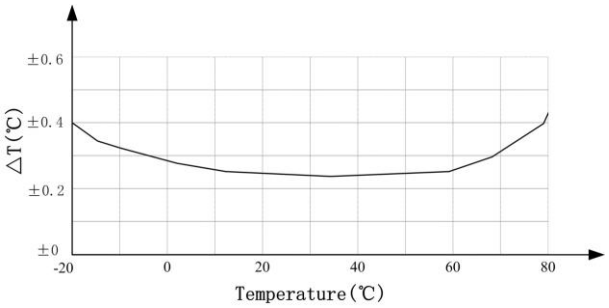


Pic2: At25℃ The error of relative humidity

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℃ 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.

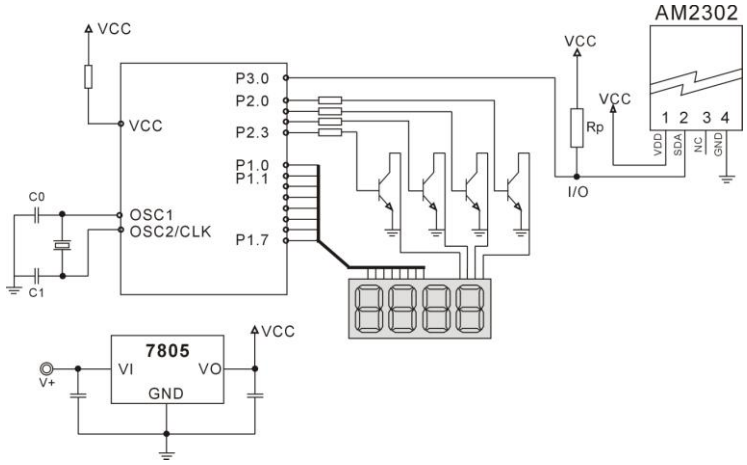
7、Single-bus communication（ONE-WIRE）

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

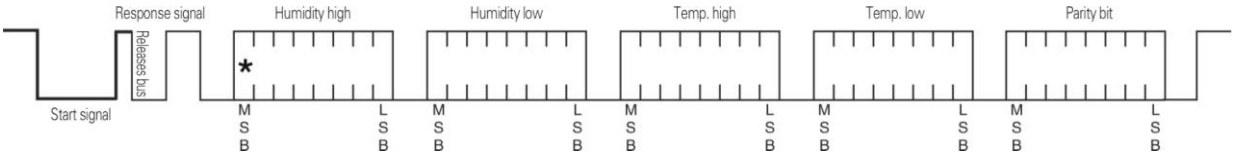
7.2、Single-bus communication protocol

◎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; 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; 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：

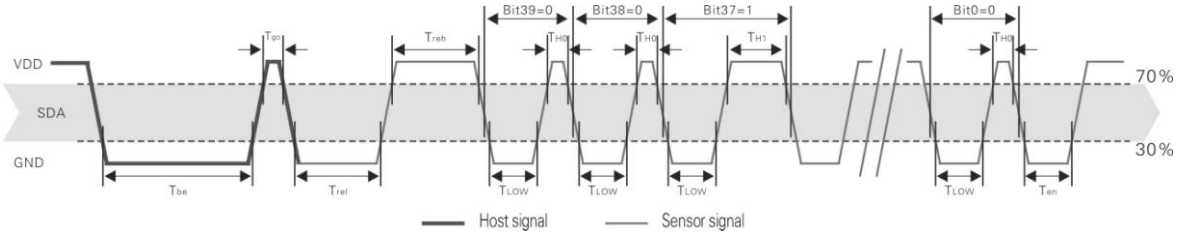
0000 0010+1001 0010 +0000 0001+0000 1101= 1010 0010 ≠ 1011 0010 （Validation error）

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

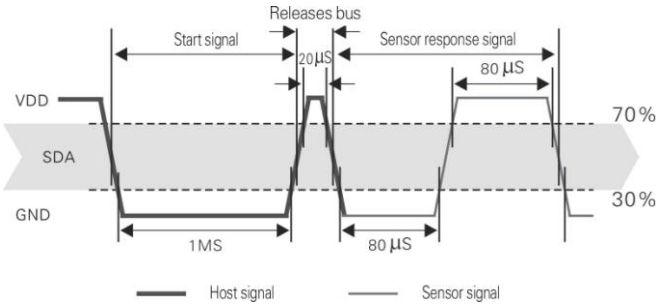
Communication between the host and the sensor can read data through the following three steps to complete.

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

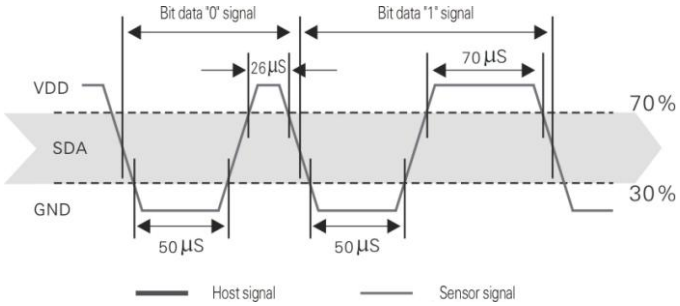
Step 3

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:

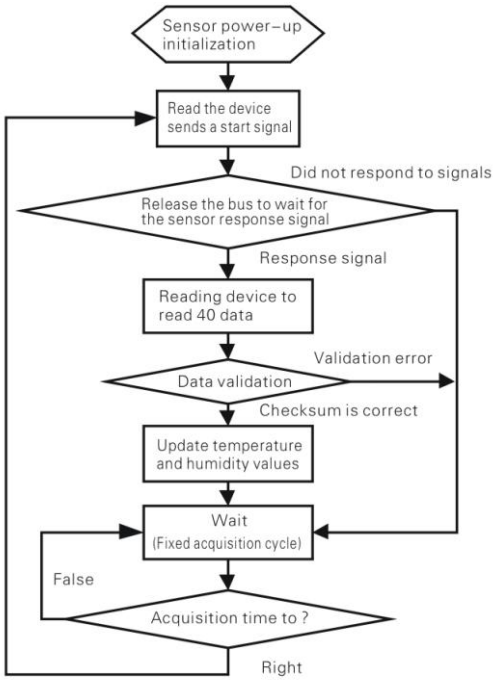


Pic 8： The single bus break down the timing diagram

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.

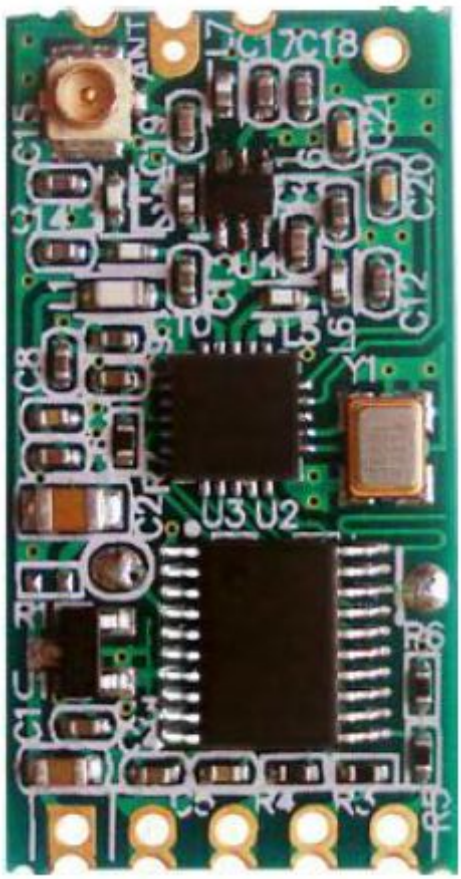
The Company is only responsible for those used in the occasion of the technical condition of the product defective product. Without any guarantee, warranty or written statement of its products used in special applications. Company for its products applied to the reliability of the product or circuit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

Capítulo 13

Anejo 5: Módulo HC-12

HC-12 Wireless Serial Port Communication Module

User Manual V1.18



Product Application

- Wireless sensor
- Community building security
- Robot wireless control
- Industrial remote control and telemetering
- Automatic data acquisition
- Container information management
- POS system
- Wireless acquisition of gas meter data
- Vehicle keyless entry system
- PC wireless networking

……

Document: 2012/10

Product Features

- Long-distance wireless transmission (1,000m in open space/baud rate 5,000bps in the air)
- Working frequency range (433.4-473.0MHz, up to 100 communication channels)
- Maximum 100mW (20dBm) transmitting power (8 gears of power can be set)
- Three working modes, adapting to different application situations
- Built-in MCU, performing communication with external device through serial port
- The number of bytes transmitted unlimited to one time
- Update software version through serial port

Product Introduction

HC-12 wireless serial port communication module is a new-generation multichannel embedded wireless data transmission module. Its wireless working frequency band is 433.4-473.0MHz, multiple channels can be set, with the stepping of 400 KHz, and there are totally 100 channels. The maximum transmitting power of module is 100mW (20dBm), the receiving sensitivity is -117dBm at baud rate of 5,000bps in the air, and the communication distance is 1,000m in open space.

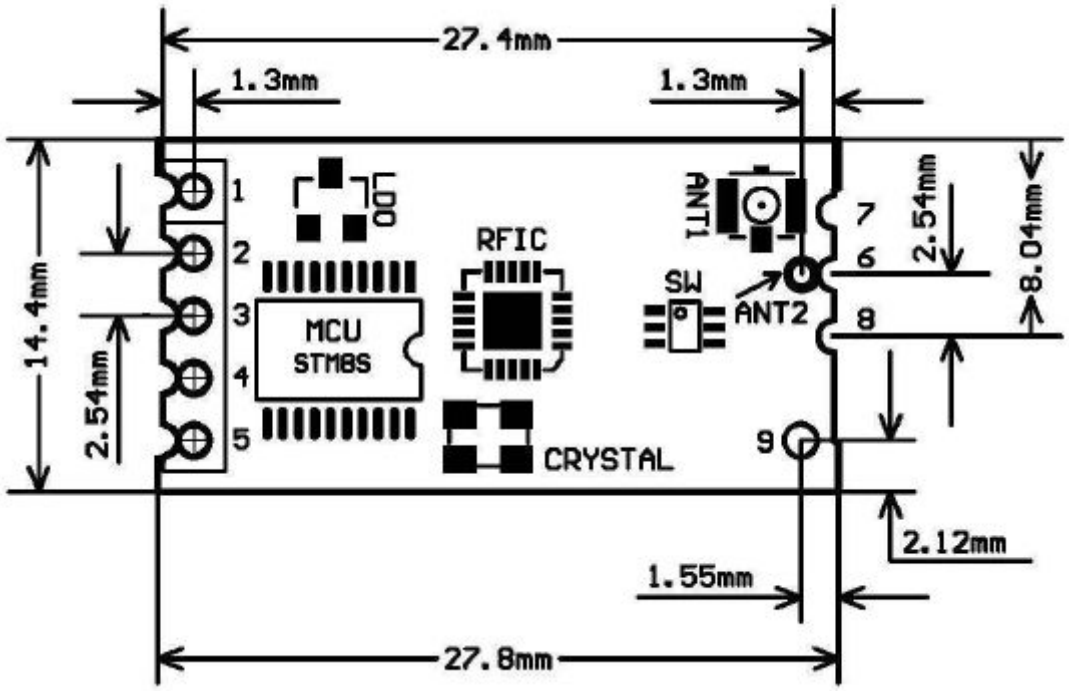
The module is encapsulated with stamp hole, can adopt patch welding, and its dimension is 27.8mm × 14.4mm × 4mm (including antenna cap, excluding spring antenna), so it is very convenient for customers to go into application system. There is a PCB antenna pedestal ANT1 on the module, and user can use external antenna of 433M frequency band through coaxial cable; there is also an antenna solder eye ANT2 in the module, and it is convenient for user to weld spring antenna. User could select one of these antennas according to use requirements.

There is MCU inside the module, and user don't need to program the module separately, and all transparent transmission mode is only responsible for receiving and sending serial port data, so it is convenient to use. The module adopts multiple serial port transparent transmission modes, and user could select them by AT command according to use requirements. The average working current of three modes FU1, FU2 and FU3 in idle state is 80μa, 3.6mA an 16mA respectively, and the maximum working current is 100mA (in transmitting state).

Product Configuration

Standard configuration of HC-12 module only contains one 433MHz-frequency-band wireless communication module with IPEX20279-001E-03 standard RF socket. The optional accessories are 433MHz-frequency-band spring antenna, IPEX-to-BNC coaxial cable and matching 433MHz-frequency-band omni-directional rubber antenna of BNC connector. User could purchase them according to use requirements.

Product Dimension



Definition of Pins

HC-12 module can adopt patch welding, or weld 2.54mm-spacing pin header, and directly insert it onto user's PCB. The module totally has nine pins and one RF antenna pedestal ANT1, and their definitions are as shown in the table below:

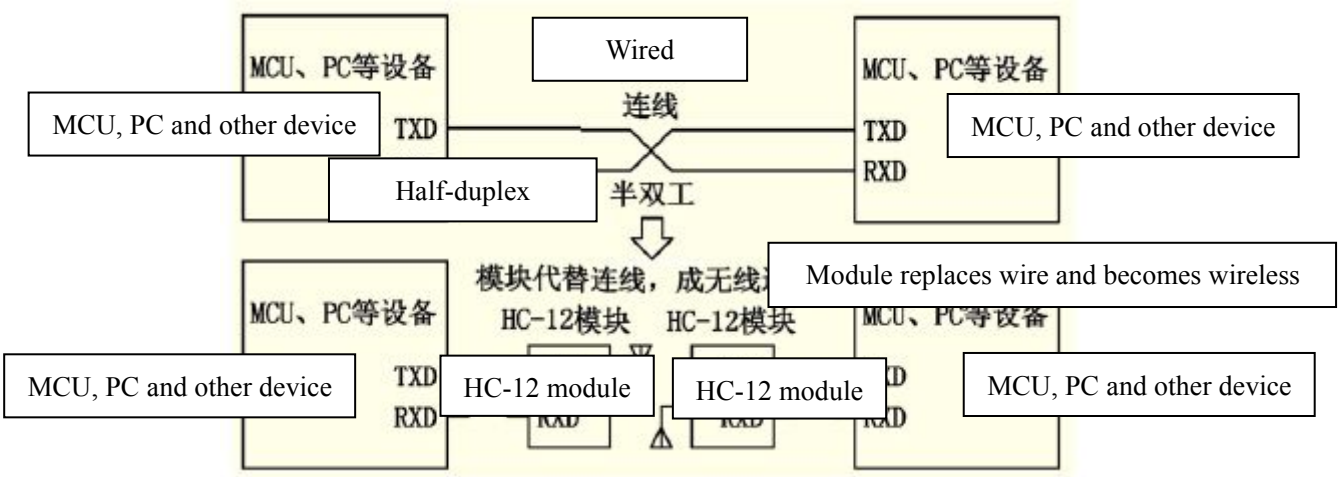
Pin	Definition	I/O direction	Note
1	VCC		Power supply input, DC3.2V-5.5V, with load capacity not less than 200mA. (Note: If the module is working in transmitting state for a long time, it is suggested that one 1N4007 diode should be connected in series when the power voltage is greater than 4.5V, to avoid heating of built-in LDO of module.)
2	GND		Common ground

3	RXD	Input, weak pull-up	URAT input port, TTL level; 1k resistance has been connected in series inside
4	TXD	Output	URAT output port, TTL level; 1k resistance has been connected in series inside
5	SET	Input, internal 10k pull-up resistance	Parameter setting control pin, valid for low level; 1k resistance has been connected in series inside
6	ANT	Input/output	433MHz antenna pin
7	GND		Common ground
8	GND		Common ground
9	NC		No connection, used in fixing, compatible with HC-11 module pin position
ANT1	ANT	Input/output	IPEX20279-001E-03 antenna socket
ANT2	ANT	Input/output	433MHz spring antenna solder eye

Pints 1-6 have two bonding pads respectively, and the outer half-hole bonding pad is used in patch welding. When the inner bonding pad ANT2 of Pin 6 is used in patch welding, the spring antenna can be welded with hands. The inner round-hole bonding pads of Pins 1-5 are used to weld 2.54mm-spacing pin header, and can be directly inserted onto user's PCB socket.

Wireless serial port transparent transmission

(1) Simple introduction of working p principle



As shown in the above figure, HC-12 module is used in physical wiring when replacing half duplex communication. The left device sends serial port data to module, and after RXD port of left module receives the serial port data, it will automatically send the data into the air via radio wave. The right module can automatically receive the data, and restore, from TXD, the serial port data originally sent by the left device. It is the same from right to left. Only half duplex state is available between modules, and they cannot receive and send data at the same time.

(2) Serial port transparent transmission

HC-12 module has three serial port transparent transmission modes, expressed with FU1, FU2 and FU3 respectively. In the use, all modes are only responsible for receiving and sending serial port data rather than wireless transmission. The default working mode of system is in FU3 full-speed mode, and in this mode, the baud rate in this air can be automatically adjusted according to baud rate of serial port, and the communication distance will be the farthest at the low baud rate. Different modes cannot transmit data to each other, and user could select the optimal mode according to practical circumstances.

The modules are usually used in pairs, and transmit data by means of half duplex. Meanwhile, the transparent transmission mode, serial port baud rate, and wireless communication channel of two paired modules shall be set to be the same. The default setting is FU3, 9,600bps (8-dibit data, no check, one stop bit), CH001 (433.4MHz).

Use the number of bytes continuously sent to serial port of module unlimited to one time. However, considering ambient interference and other factors, if thousands of data size is sent continuously at a time, some number of bytes may be lost. Therefore, the upper computer shall have response and resending mechanism, to avoid information loss.

(3) Three serial port transparent transmission modes

When HC-12 module leaves the factory, its default serial port transparent transmission mode is FU3. Then the module is in full-speed state, and the idle current is about

16mA. In this mode, the module can automatically adjust the baud rate of wireless transmission in the air according to serial port baud rate, and the corresponding relationship is as shown in the table below:

Serial port baud rate	1,200bps	2,400bps	4,800bps	9,600bps	19,200bps	38,400bps	57,600bps	115,200bps
Baud rate in the air	5,000bps		15,000bps		58,000bps		236,000bps	

To get the farthest communication distance, the serial port baud rate can be set to be low. For short-time transmission of mass data, set the serial port baud rate to be high, but the communication distance will be reduced accordingly.

The receiving sensitivity of module at different baud rates in the air is as shown in the table below:

Baud rate in the air	5,000bps	15,000bps	58,000bps	236,000bps
Wireless receiving sensitivity	-117dBm	-112dBm	-107dBm	-100dBm

Generally, every time the receiving sensitivity is reduced by 6dB, the communication distance will be reduced by half.

When “SET” pin of module is at low level, the serial port transparent transmission mode can be set through AT command (see the introduction in the following chapter for details).

FU1 mode is relatively power saving mode, and the idle working current of this mode is about 3.6mA. In this mode, the module can also set eight types of serial port baud rate as shown in the above table, but the baud rate in the air is uniform, 250,000bps.
FU2 mode is power saving mode, and the idle working current of this mode is about 80μA. In this mode, the module only supports baud rates of 1,200bps, 2,400bps and 4,800bps, and the baud rate in the air is uniform, 250,000bps. If the module is set to be other serial port baud rate, the module cannot conduct communication normally.

Meanwhile, when the module is set to be FU2 mode in FU1 and FU3 mode, the baud rate exceeding 4,800bps will be automatically reduced to be 4,800bps. In FU2 mode, the sending time interval of data package cannot be too short; otherwise, the data will be lost. It is suggested that the sending time interval of data package should not be less than 1sec.

The following gives some characteristics reference values of various modes:

Mode	FU1	FU2	FU3	Remark
Idle current	3.6mA	80μA	16mA	Average value
Transmission time delay	15-25mS	500mS	4-80mS	Sending one byte
Loopback test time delay 1	31mS			Serial port baud rate 9,600, sending one byte
Loopback test time delay 2	31mS			Serial port baud rate 9,600, sending ten bytes

Note: Loopback test time delay means the duration from the time of, after conducting short circuit on TX and RX pins of one module and sending serial port data to the other module, starting to send serial port data to the other module to the time that the returned data appear at TX pin of the other module.

Module Parameter Setting AT Command

AT command is used to set the module parameters and switch the module functions, and after setting, it will be valid only after exiting from setting state. Meanwhile, modification of parameters and functions will not be lost in case of power failure.

(1) Command mode entering

The first way to enter: in normal use (energized), put Pin 5 “SET” in low level;

The second way to enter: disconnect power supply, first put Pin 5 “Set” in low level, and then energize it;

Either of the above two ways can make the module enter AT command mode; release it (not put pin “SET” in low level), and exit from the command mode. If the module function is changed after exiting from command mode, it will be switched to corresponding functional status.

In the second way, the module enters AT in the serial port format of 9,600, N, 1 constantly.

(2) Command instruction

①. AT

Test command.

e.g.:
Send “AT” command to module, and the module returns “OK”.

②AT+Bxxxx
Change the serial port baud rate. The baud rate can be set to be 1,200bps, 2,400bps, 4,800bps, 9,600bps, 19,200bps, 38,400bps, 57,600bps, and 115,200bps. The default value is 9,600bps.
e.g.: To set serial port baud rate of module to be 19,200bps, first send “AT+B19200” command to module, and the module returns “OK+B19200”.

③AT+Cxxxx
Change wireless communication channel, optional from 001 to 127 (for the wireless channel exceeding 100, the communication distance cannot be ensured). The default value of wireless channel is 001, and the working frequency is 433.4MHz. The channel stepping is 400KHz, and the working frequency of Channel 100 is 473.0MHz.
e.g.:
To set the module to work at Channel 21, first send “AT+C021” command to the module, and the module returns “COK+C021”. After exiting from the command mode, the module will work at Channel 21, and the working frequency is 441.4MHz.
Note: As the wireless receiving sensitivity of HC-12 module is relatively high, when the serial port baud rate is greater than 9,600bps, five adjacent channels shall be staggered to use. When the serial port baud rate is not greater than 9,600bps, in short-distance (within 10m) communication, also five adjacent channels shall be staggered to use.

④AT+FUx
Change serial port transparent transmission mode of module and three modes are available, namely, FU1, FU2 and FU3. The default mode of module is FU3, and only when serial port transparent transmission mode of two modules is set to be the same, can normal communication be available. For detailed introduction, please see the above “wireless serial port transparent transmission”.
e.g.:
Send “AT+FU1” to module, and the module returns “AT+0K”.

⑤AT+Px
Set transmitting power of module, x is optional from 1 to 8, and the corresponding transmitting power of module is as shown below:

x value	1	2	3	4	5	6	7	8
Transmitting power of module (dBm)	-1	2	5	8	11	14	17	20

The default value is 8, and the higher the transmitting power is, the farther the communication distance is. When the transmitting power level is set to be 1, the transmitting power is the minimum. Generally speaking, every time the transmitting power is reduced by 6dB, the communication distance will be reduced by half.
e.g.:
Send “AT+P5” command to module, and the module returns “OK+P5”. After exiting from the command code, the transmitting power of module is +11dBm.

⑥AT+Ry
Obtain single parameter of module, y is any letter among B, C, F and P, respectively representing: baud rate, communication channel, serial port transparent transmission mode, and transmitting power.
Example 1:
Send “AT+RB” to module, and if the module returns “OK+B9600”, it is inquired that the serial port baud rate of module is 9,600bps.
Example 2:
Send “AT+RC” command to module, and if the module returns “OK+RC001”, it is inquired that the communication channel of module is 001.
Example 3:
Send “AT+RF” command to module, and if the module returns “OK+FU3”, it is inquired that the module is working in serial port transparent transmission mode 3.
Example 4:
Send “AT+RP” command to module, and if the module returns “OK+RP: +20dBm”, it is inquired that the transmitting power of module is +20dBm.

⑦AT+RX
Obtain all parameters of module. Return serial port transparent transmission mode, serial port baud rate, communication channel, and transmitting power in order.
e.g.:
Send “AT+RX” command to module, and the module returns “OK+FU3\r\n OK+B9600\r\n OK+C001\r\n OK+RP: +20dBm\r\n”. (“\r\n” means return\newline)

⑧AT+Uxxx
Set data bits, check bit and stop bit of serial port communication. For check bit, N means no check, O means odd check, and E means even check. For stop bit, 1 means one stop bit, 2 means two stop bits, and 3 means 1.5 stop bits.
e.g.:
To send serial port format to be eight data bits, odd check, and one stop bit, please Send “AT+U8O1” to module, and the module returns “OK+U8O1”.

⑨AT+V
Inquire firmware version information of module.
e.g.:
Send “AT+V” command to module, and the module returns “HC-12_V1.1”.

⑩AT+SLEEP

After receiving the command, the module enters sleep mode after exiting from AT, the working current is about 22μA, and this mode doesn't allow serial port data transmission. Then enter AT setting state again, and the module will exit from sleep mode automatically.

e.g.:

When wireless data transmission is not needed, to save power, send "AT+SLEEP" command to module, and the module returns "OK+SLEEP".

⑪AT+DEFAULT

Set serial port baud rate, communication channel, and serial port transparent transmission mode to be default value.

e.g.:

Send "AT+DEFAULT" to module, and the module returns "OK+DEFAULT", and the default vale is restored. The serial port baud rate is 9,600bps, communication channel is C001, and serial port transparent transmission mode is FU3.

⑫AT+UPDATE

Put the module in the status of waiting for software update.

After sending the command, the module will not respond to command any more, until it is re-energized.

After sending the command, please close the serial port assistant, and turn on HC-1X updater to update the software. For detailed operating method, please refer to the following "software update" introduction.

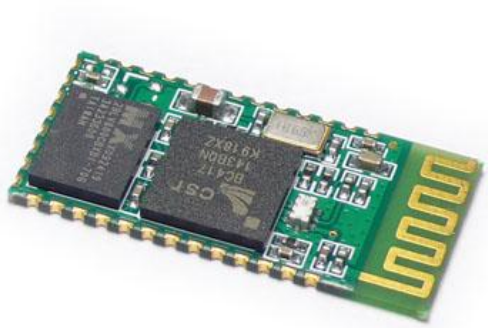
Capítulo 14

Anejo 5: Módulo HC-05

HC-05

-Bluetooth to Serial Port Module

Overview



HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

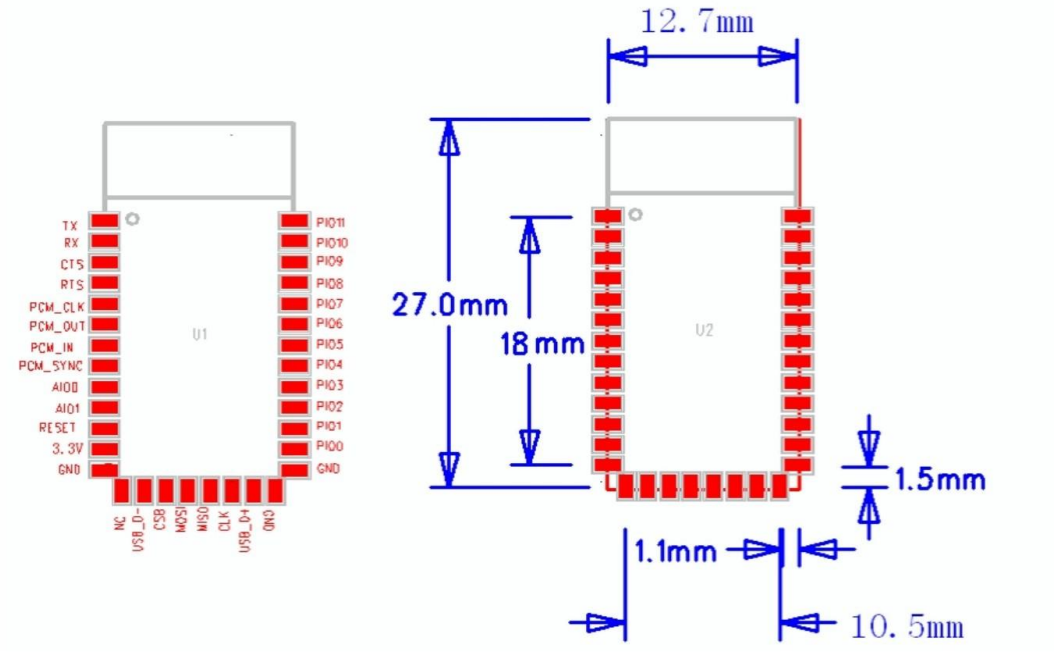
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

RESETB	11	CMOS input with weak internal pull-up	Reset if low.input debouncde so must be low for >5MS to cause a reset	
UART_RTS	4	CMOS output, tri-stable with weak internal pull-up	UART request to send, active low	
UART_CTS	3	CMOS input with weak internal pull-down	UART clear to send, active low	
UART_RX	2	CMOS input with weak internal pull-down	UART Data input	
UART_TX	1	CMOS output, Tri-stable with weak internal pull-up	UART Data output	
SPI_MOSI	17	CMOS input with weak internal pull-down	Serial peripheral interface data input	
SPI_CSB	16	CMOS input with weak internal pull-up	Chip select for serial peripheral interface, active low	
SPI_CLK	19	CMOS input with weak internal pull-down	Serial peripheral interface clock	
SPI_MISO	18	CMOS input with weak internal pull-down	Serial peripheral interface data Output	
USB_-	15	Bi-Directional		

USB_+	20	Bi-Directional		
NC	14			
PCM_CLK	5	Bi-Directional	Synchronous PCM data clock	
PCM_OUT	6	CMOS output	Synchronous PCM data output	
PCM_IN	7	CMOS Input	Synchronous PCM data input	
PCM_SYNC	8	Bi-Directional	Synchronous PCM data strobe	

AT command Default:

How to set the mode to server (master):
1. Connect PIO11 to high level.
2. Power on, module into command state.
3. Using baud rate 38400, sent the “AT+ROLE=1\r\n” to module, with “OK\r\n” means setting successes.
4. Connect the PIO11 to low level, repower the module, the module work as server (master).

AT commands: (all end with \r\n)

1. Test command:

Command	Respond	Parameter
AT	OK	-

2. Reset

Command	Respond	Parameter
AT+RESET	OK	-

3. Get firmware version

Command	Respond	Parameter
AT+VERSION?	+VERSION:<Param> OK	Param : firmware version

Example:
AT+VERSION?\r\n
+VERSION:2.0-20100601
OK

4. Restore default

Command	Respond	Parameter
AT+ORGL	OK	-

Default state:

Slave mode, pin code :1234, device name: H-C-2010-06-01 ,Baud 38400bits/s.

5. Get module address

Command	Respond	Parameter
AT+ADDR?	+ADDR:<Param> OK	Param: address of Bluetooth module

Bluetooth address: NAP: UAP : LAP

Example:

AT+ADDR?\r\n

+ADDR:1234:56:abcdef

OK

6. Set/Check module name:

Command	Respond	Parameter
AT+NAME=<Param>	OK	Param: Bluetooth module name (Default :HC-05)
AT+NAME?	+NAME:<Param> OK (/FAIL)	

Example:

AT+NAME=HC-05\r\n set the module name to “HC-05”

OK

AT+NAME=ITeadStudio\r\n

OK

AT+NAME?\r\n

+NAME: ITeadStudio

OK

7. Get the Bluetooth device name:

Command	Respond	Parameter
AT+RNAME?<Param1>	1. +NAME:<Param2> OK 2. FAIL	Param1,Param 2 : the address of Bluetooth device

Example: (Device address 00:02:72:od:22:24，name：ITead)

AT+RNAME? 0002，72，od2224\r\n

+RNAME:ITead

OK

8. Set/Check module mode:

Command	Respond	Parameter
AT+ROLE=<Param>	OK	Param: 0- Slave
AT+ ROLE?	+ROLE:<Param>	

	OK	1-Master 2-Slave-Loop

9. Set/Check device class

Command	Respond	Parameter
AT+CLASS=<Param>	OK	Param: Device Class
AT+ CLASS?	1. +CLASS:<Param> OK 2. FAIL	

10. Set/Check GIAC (General Inquire Access Code)

Command	Respond	Parameter
AT+IAC=<Param>	1.OK 2. FAIL	Param: GIAC (Default : 9e8b33)
AT+IAC	+IAC:<Param> OK	

Example:

AT+IAC=9e8b3f\r\n

OK

AT+IAC?\r\n

+IAC: 9e8b3f

OK

11. Set/Check -- Query access patterns

Command	Respond	Parameter
AT+INQM=<Param>,<Param2>,<Param3>	1.OK 2. FAIL	Param: 0——inquiry_mode_standard 1——inquiry_mode_rssi Param2: Maximum number of Bluetooth devices to respond to Param3: Timeout (1-48 : 1.28s to 61.44s)
AT+ INQM?	+INQM： <Param>,<Param2>,<Param3> OK	

Example:

AT+INQM=1,9,48\r\n

OK

AT+INQM\r\n

+INQM:1, 9, 48

OK

12. Set/Check PIN code:

Command	Respond	Parameter
AT+PSWD=<Param>	OK	Param: PIN code (Default 1234)
AT+ PSWD?	+ PSWD ： <Param> OK	

13. Set/Check serial parameter:

Command	Respond	Parameter
AT+UART=<Param>,<Param2>,<Param3>	OK	Param1: Baud Param2: Stop bit Param3: Parity
AT+ UART?	+UART=<Param>,<Param2>,<Param3> OK	

Example:

AT+UART=115200，1,2,\r\n

OK

AT+UART?

+UART:115200,1,2

OK

14. Set/Check connect mode:

Command	Respond	Parameter
AT+CMODE=<Param>	OK	Param: 0 - connect fixed address 1 - connect any address 2 - slave-Loop
AT+ CMODE?	+ CMODE:<Param> OK	

15. Set/Check fixed address:

Command	Respond	Parameter
AT+BIND=<Param>	OK	Param: Fixed address (Default 00:00:00:00:00:00)
AT+ BIND?	+ BIND:<Param> OK	

Example:

AT+BIND=1234，56，abcdef\r\n

OK

AT+BIND?\r\n

+BIND:1234:56:abcdef

OK

16. Set/Check LED I/O

Command	Respond	Parameter
AT+POLAR=<Param1,<Param2>	OK	Param1: 0- PIO8 low drive LED 1- PIO8 high drive LED
AT+ POLAR?	+ POLAR=<Param1>,<Param2> OK	

		Param2: 0- PIO9 low drive LED 1- PIO9 high drive LED

17. Set PIO output

Command	Respond	Parameter
AT+PIO=<Param1>,<Param2>	OK	Param1: PIO number Param2: PIO level 0- low 1- high

Example:

1. PIO10 output high level

AT+PI0=10， 1\r\n

OK

18. Set/Check – scan parameter

Command	Respond	Parameter
AT+IPSCAN=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Query time interval Param2：Query duration Param3：Paging interval Param4：Call duration
AT+IPSCAN?	+IPSCAN:<Param1>,<Param2>,<Param3>,<Param4> OK	

Example:

AT+IPSCAN =1234,500,1200,250\r\n

OK

AT+IPSCAN?

+IPSCAN:1234,500,1200,250

19. Set/Check – SHIFF parameter

Command	Respond	Parameter
AT+SNIFF=<Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: Max time Param2: Min time Param3: Retry time Param4: Time out
AT+ SNIFF?	+SNIFF:<Param1>,<Param2>,<Param3>,<Param4> OK	

20. Set/Check security mode

Command	Respond	Parameter
AT+SENM=<Param1>,<Param2>	1. OK 2. FAIL	Param1: 0——sec_mode0+off 1——sec_mode1+non_se
AT+ SENM?	+ SENM:<Param1>,<Param2>	

	OK	cure 2——sec_mode2_service 3——sec_mode3_link 4——sec_mode_unknown Param2: 0——hci_enc_mode_off 1——hci_enc_mode_pt_to_pt 2——hci_enc_mode_pt_to_pt_and_bcast

21. Delete Authenticated Device

Command	Respond	Parameter
AT+PMSAD=<Param>	OK	Param: Authenticated Device Address

Example:

AT+PMSAD =1234,56,abcdef\r\n

OK

22. Delete All Authenticated Device

Command	Respond	Parameter
AT+ RMAAD	OK	-

23. Search Authenticated Device

Command	Respond	Parameter
AT+FSAD=<Param>	1. OK 2. FAIL	Param: Device address

24. Get Authenticated Device Count

Command	Respond	Parameter
AT+ADCN?	+ADCN：<Param> OK	Param: Device Count

25. Most Recently Used Authenticated Device

Command	Respond	Parameter
AT+MRAD?	+ MRAD：<Param> OK	Param: Recently Authenticated Device Address

26. Get the module working state

Command	Respond	Parameter

AT+ STATE?	+ STATE：<Param> OK	Param: “INITIALIZED” “READY” “PAIRABLE” “PAIRED” “INQUIRING” “CONNECTING” “CONNECTED” “DISCONNECTED” “NUKNOW”

27. Initialize the SPP profile lib

Command	Respond	Parameter
AT+INIT	1. OK 2. FAIL	-

28. Inquiry Bluetooth Device

Command	Respond	Parameter
AT+INQ	+INQ: <Param1>， <Param2>，<Param3> …. OK	Param1：Address Param2：Device Class Param3：RSSI Signal strength

Example:

AT+INIT\r\n

OK

AT+IAC=9e8b33\r\n

OK

AT+CLASS=0\r\n

AT+INQM=1,9,48\r\n

At+INQ\r\n

+INQ:2:72:D2224,3E0104,FFBC

+INQ:1234:56:0,1F1F,FFC1

+INQ:1234:56:0,1F1F,FFC0

+INQ:1234:56:0,1F1F,FFC1

+INQ:2:72:D2224,3F0104,FFAD

+INQ:1234:56:0,1F1F,FFBE

+INQ:1234:56:0,1F1F,FFC2

+INQ:1234:56:0,1F1F,FFBE

+INQ:2:72:D2224,3F0104,FFBC

OK

28. Cancel Inquiring Bluetooth Device

Command	Respond	Parameter
AT+ INQC	OK	-

29. Equipment Matching

Command	Respond	Parameter
AT+PAIR=<Param1>,<Param2>	1. OK 2. FAIL	Param1： Device Address Param2： Time out

30. Connect Device

Command	Respond	Parameter
AT+LINK=<Param>	1. OK 2. FAIL	Param： Device Address

Example:

AT+FSAD=1234,56,abcdef\r\n

OK

AT+LINK=1234,56,abcdef\r\n

OK

31. Disconnect

Command	Respond	Parameter
AT+DISC	1. +DISC:SUCCESS OK 2. +DISC:LINK_LOSS OK 3. +DISC:NO_SLC OK 4. +DISC:TIMEOUT OK 5. +DISC:ERROR OK	Param： Device Address

32. Energy-saving mode

Command	Respond	Parameter
AT+ENSNIFF=<Param>	OK	Param： Device Address

33. Exerts Energy-saving mode

Command	Respond	Parameter
AT+ EXSNIFF =<Param>	OK	Param： Device Address

Revision History

Rev.	Description	Release date
v1.0	Initial version	7/18/2010