UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

ANÁLISIS DE TEMPERATURA EN FPGAs

Mónica Torre Albarsanz

Marzo 2015

ANÁLISIS DE TEMPERATURA EN FPGAs

AUTOR: Mónica Torre Albarsanz

TUTOR: Carlos Minchola Guardia
PONENTE: Eduardo Boemo Scalvinoni

Digital SystemLab
Dpto. de Tecnología Electrónica y Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Marzo de 2015

Resumen

Este proyecto se ha basado fundamentalmente en el diseño, desarrollo y caracterización de osciladores anillo en una FPGA, modelo Virtex-5 para obtener la temperatura de la misma a través de la frecuencia de oscilación de los sensores.

Para ello primero se diseñaron los osciladores anillo en VHDL y una lógica de control que permitiera obtener la frecuencia de oscilación y controlase la habilitación/deshabilitación de los sensores.

Los datos de la frecuencia obtenidos fueron enviados a través de la interfaz RS232 (UART) al ordenador. Con el fin de almacenar y procesar esos datos se crea una interfaz en Matlab. Tras obtener los datos se realizaron distintas representaciones gráficas para ayudar a la interpretación de los resultados.

Con el fin de obtener la temperatura de la placa y poder así realizar la calibración de los sensores anillo, se monitorizó el diodo interno de temperatura pre-calibrado que posee el modelo de FPGA utilizado. Por tal razón, el módulo System Monitor tuvo que ser considerado como parte de nuestro diseño obteniendo aproximadamente la temperatura del FPGA como su respectivo voltaje de núcleo

Una vez definido el oscilador e interconectarlo con Matlab, se procedió a realizar tres experimentos diferentes:

El primero consistió en identificar el número de inversores que son necesarios para que los datos recogidos del sensor anillo sean los más fieles posibles a la realidad.

En el segundo se realizó la calibración de dos sensores próximos al diodo calibrado interno que posee el FPGA para estudiar cual es la variación de la frecuencia de oscilación con respecto el voltaje de núcleo, la temperatura y la posición de los anillos.

Y el tercero consistió en colocar 48 sensores distribuidos por toda la FPGA y obtener la frecuencia de oscilación de cada uno de ellos, procesando un número determinado de muestras por cada sensor.

En resumen, se han realizado 12 versiones de circuitos sobre los cuales se han hecho 21.504 medidas. Y el código contiene unas 8.700 líneas divididas en 6 ficheros diferentes.

Palabras clave

FPGA, oscilador anillo, sensor, System Monitor, UART, VHDL, equilibrio térmico, inversor, desviación estándar media, mapa térmico, temperatura, voltaje de núcleo, frecuencia de oscilación.

Abstract

The aim of this project tackles the design, development and analysis of ring oscillators, which comprises of a series of inverters, implemented over an FPGA-Virtex-5 in order to sense the temperature gradient of hot-spots in FPGAs.

Therefore, our proposal consists of designing an array of ring oscillators which are monitored by a control central unit which actives the period required to conduct several stages of our design in order to read the counter generated by ring oscillator.

The data obtained are computationally processed carrying out varies operations such as frequency oscillation calculation, median and standard deviation as well as several graphs depicting profiles thermal. Likewise, a PC connected via UART to the FPGA in charges of receiving the data from our design.

In order to get the absolute temperature of the tested FPGA and be able to perform calibration of the sensor ring the System Monitor module was used. It's worth mentioning that this Xilinx FPGA contains a pre-calibrated built-in thermal diode which is sensed by the module indicated earlier.

Several experiments were carried out evaluating the ring oscillator design. It can be mentioned the following:

As a first experiment, it was examined the number of inverters utilized in each sensor by means of several design combinations and evaluated their respective sensor performance.

As second one, it was calibrated two ring oscillator sensors closets to the position of the built-in thermal diode in order to evaluate the variation of the tuned frequency related to absolute temperature, voltage core and relative placement within FPGA.

Lastly, it was tested the frequency oscillation of each sensor considering an array of 48 thermal sensors distributed properly over the FPGA.

In summary, there have been made 12 circuit versions and 21.504 measurements. And the code contains 8.700 lines divided into 6 different files.

Key words

FPGA, ring oscillator, sensor, System Monitor, UART, VHDL, thermal equilibrium, inverter, average standard deviation, thermal mapping, temperature, core voltage, frequency oscillation.

Agradecimientos

Quiero agradecer a mi tutor y a mi ponente, por la ayuda que me han prestado durante todo el proceso, desde la elección del tema hasta el análisis de las conclusiones finales una vez terminados los diferentes experimentos.

También quiere agradecer a todos aquellos, compañeros, amigos, familiares que me han ayudado durante todos los años de estudio, para llegar hasta esta difícil meta.

Y por último a ti, tú ya sabes a quien me refiero, por estar a mi lado animándome en los momentos difíciles y celebrándolo cuando todo salía bien.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

La frecuencia de operación y densidad de puertas de FPGAs actuales (2015) convierte la medición de temperatura en una técnica crucial para garantizar funcionamiento y fiabilidad. [1]

Las técnicas para medir temperatura en FPGA son:

1. Diodos embebidos en el DIE.
2. Diodo Clamping de un IOB.
3. System Monitor (V5).
4. Sensores externos.
5. Cámaras IR.
6. Osciladores en anillo.

Los sensores de temperatura empleados en la FPGA deben ser digitales ya que, salvo algunas excepciones, éstas son chips digitales. Pero la temperatura es una magnitud analógica. Por tanto, como la única magnitud analógica que puede medirse mediante un circuito digital es la frecuencia, se utilizan osciladores anillo.

Los osciladores en anillo son ideales para ser empleados en los circuitos reconfigurables, pues se pueden construir solo con los recursos de la FPGA, sin tener que recurrir a circuitos analógicos externos. Estos circuitos están compuestos por un lazo cerrado que incluye un número impar de inversores. De este modo, se produce el cambio de fase necesario para mantener la oscilación. El periodo resultante es dos veces la suma de los retardos de todos los elementos que componen el lazo. Los inversores pueden mapearse utilizando las tablas de look-up de los bloques lógicos configurables (CLBs), o los propios inversores que incluyen los bloques de E/S (IOBs) de las FPGAs. En cualquiera de los casos, es útil insertar una puerta AND para abrir el lazo y evitar el autocalentamiento, así como un buffer a la salida, con el objeto de prevenir variaciones de frecuencia debido a variaciones en la carga de la señal fout.

1.2 Objetivos

Este trabajo se centra en el estudio e implementación de sensores de temperatura en una FPGA (Virtex-5). El sensor está basado en una matriz de osciladores anillos distribuidos sobre el dispositivo mencionado.

Se implementará una metodología para evaluar el rendimiento de los sensores de temperatura. En este punto se consideran cálculos de frecuencias de oscilación, patrones estadísticos, y comparaciones.

Se diseñarán, simularán e implementarán diferentes osciladores en anillo, los cuales se analizarán a diferentes temperaturas por medio de un horno de temperatura controlada. Se diseñará además la interfaz para adquirir los datos y un método de representación gráfica en tiempo real de las medidas de temperatura

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción, motivación y objetivos del proyecto.
- Capítulo 2: Estado del arte.
- Capítulo 3: Diseño.
- Capítulo 4: Integración, pruebas y resultados.
- Capítulo 5: Conclusiones y trabajo futuro.

2 Estado del arte

En este capítulo se van a repasar las técnicas más destacables relacionadas con este proyecto, como por ejemplo, el funcionamiento de los osciladores anillo. Pero también los conceptos básicos de calor y temperatura.

2.1 Conceptos básicos del calor

Para poder explicar los conceptos de calor y temperatura, antes hay que entender otros dos conceptos [13]:

- Contacto térmico es cuando dos objetos se pueden intercambiar energía debido a una diferencia de temperatura, aunque no estén en contacto físico entre ellos y pasen por un tercer objeto o medio.
- Equilibrio térmico es por tanto, cuando estos dos objetos no intercambian nada de energía por calor o radiación electromagnética si se ponen en contacto térmico.
- Principio cero de la termodinámica: Si dos objetos están en equilibrio térmico con un tercero, entonces están en equilibrio entre sí.

Una vez explicados estos términos, se puede definir el concepto de temperatura como la propiedad que determine si un objeto está en equilibrio térmico con otros objetos. Por tanto si dos objetos están en equilibrio térmico están a la misma temperatura.

Entonces según las definiciones anteriores, se puede definir el calor como la transferencia de energía a través de la frontera de un sistema debido a la diferencia de temperatura entre el sistema y su entorno.

En conclusión, el calor es la energía que se transfiere y la temperatura es una medida de ella.

Hay varias formas de transferir el calor, conducción, convección y radiación:

- Conducción: la energía se transmite en forma de calor como consecuencia de las interacciones entre átomos o moléculas, aunque no exista un transporte de los propios átomos o moléculas. Por ejemplo cuando se calienta una barra por un extremo, el calor se extiende a toda ella aunque no se esté calentando esa parte.
- Convección: el calor se transfiere mediante un transporte directo de masa. Por ejemplo, cuando sube el aire caliente de una habitación hacia arriba, además del aire sube la energía calorífica.
- Radiación: la energía térmica se transporta a través del espacio en forma de ondas electromagnéticas que se mueven a la velocidad de la luz. Por ejemplo cualquier tipo de onda como las luminosas, las de radio, televisión…

En todos los mecanismos de transmisión de calor, la velocidad de enfriamiento de un cuerpo es aproximadamente proporcional a la diferencia de temperatura que existe entre el cuerpo y el medio que la rodea [17].

2.2 Osciladores anillo

El oscilador anillo es una de las muchas técnicas que existen para obtener la temperatura en FPGAs.

Estos se construyen a partir de puertas inversoras, las cuales se colocan formando un lazo cerrado, y siempre siendo un número impar de ellas. Con ellas se obtiene la frecuencia de oscilación. Gracias a esta arquitectura, estos pueden ser instanciados en la posición que más le convenga al usuario y tantos como desee [3].

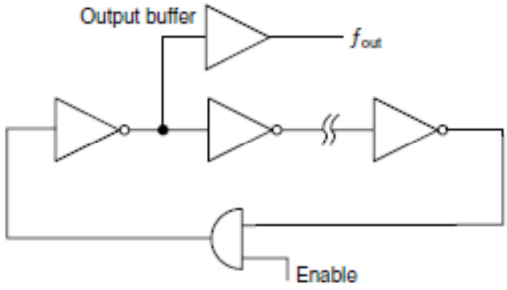

Figura 2-1: Esquema de un sensor anillo. Extraído de [4].

Estos osciladores pueden tener múltiples funciones pero en este caso se han utilizado para obtener la temperatura, la cual se obtiene de forma indirecta realizando primero una recogida de los datos de la frecuencia y posteriormente calibrando el sensor. Además, si se desea obtener dicha temperatura, se tendrá que hacer una caracterización de cada sensor de la respuesta respecto a voltaje, temperatura y localización del mismo. Se debe hacer una calibración por sensor, ya que el ruteado es diferente y por tanto la calibración también lo será.

Por tanto, un oscilador convencional es el formado por N etapas de inversores, como el que se muestra en la Figura 2-2. Asumiendo que las capacitancias parásitas (Cg) entre etapas son iguales entonces la frecuencia de oscilación es representada como:

$$f_{osc} = \frac{1}{2Nt} \text{ donde t es el delay por cada etapa.}$$

A su vez t es representada con la expresión:

$$t = \frac{V_{nucleo} * C_g}{I_{control}}$$

Reemplazando en la expresión anterior obtenemos lo siguiente:

$$f_{osc} = \frac{I_{control}}{2 * N * V_{nucleo} * C_g}$$

Lo que demuestra que la oscilación depende directamente de la corriente y del voltaje de núcleo.

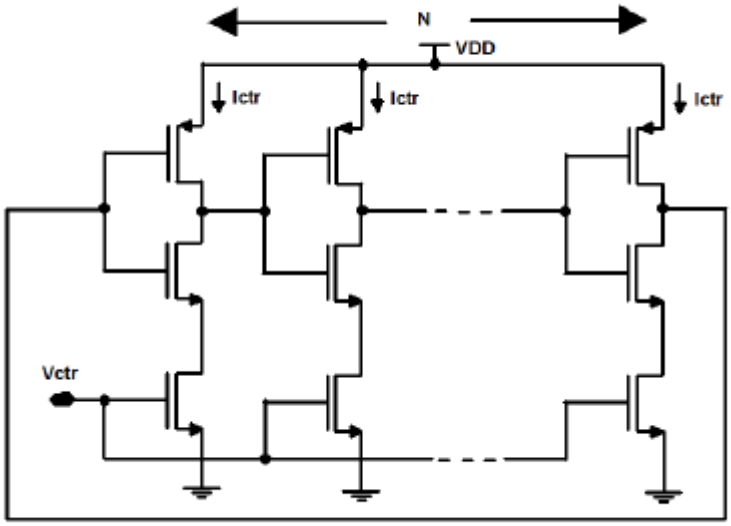

Figura 2-2: Oscilador convencional. Extraído de [25].

Estos tiempos de propagación dependen de la temperatura y del voltaje de núcleo y por tanto varían con ellos. Los fabricantes de circuitos proporcionan tablas y gráficos informando de estas variaciones como por ejemplo la que se muestra en la siguiente tabla que son datos basados en procesadores de 180 nm [16].

Tabla 2-1: Tabla de "deration" en función de la temperatura y el voltaje. Indica el coeficiente por el cual hay que multiplicar un retardo nominal para cada (V,T) Extraído de [16]

Vcc	**Temperatura (ºC)**								
	120	**100**	**80**	**60**	**40**	**32**	**20**	**0**	**-20**
2.6	1.39	1.36	1.32	1.27	1.24	1.20	1.19	1.15	1.12
2.8	1.29	1.25	1.22	1.18	1.14	1.11	1.10	1.07	1.03

3.0	1.23	1.20	1.17	1.12	1.09	1.06	1.05	1.02	0.99
3.2	1.18	1.15	1.12	1.08	1.05	1.02	1.01	0.98	0.95
3.3	1.16	1.13	1.10	1.06	1.03	1.00	0.99	0.96	0.93
3.4	1.14	1.11	1.08	1.04	1.01	0.98	0.97	0.94	0.91

Según estos datos, la frecuencia de oscilación del sensor de anillo será dependiente de estos dos factores. Pero también varía según los parámetros de fabricación, por lo que no solo habrá diferencias dentro del mismo chip, sino también entre diferentes chips [1].

Estas relaciones entre temperatura, voltaje y frecuencia de oscilación se han demostrado solo en FPGAs de más de 65 nm, ya que en circuitos con un ancho de canal de 45 nm existe una inversión de temperatura y este comportamiento cambia [5].

Este sensor basado en un oscilador anillo tiene tres partes diferenciadas [1]:

1. Un lazo cerrado de inversores con una puerta AND para habilitar o deshabilitar el oscilador y evitar el auto calentamiento.
2. Una lógica de control, la cual controla la puerta AND anterior y el tiempo de captura.
3. Un FDC (Frequency to digital convert), el cual se encarga de capturar el número de oscilaciones durante un intervalo de tiempo dado por la lógica de control.

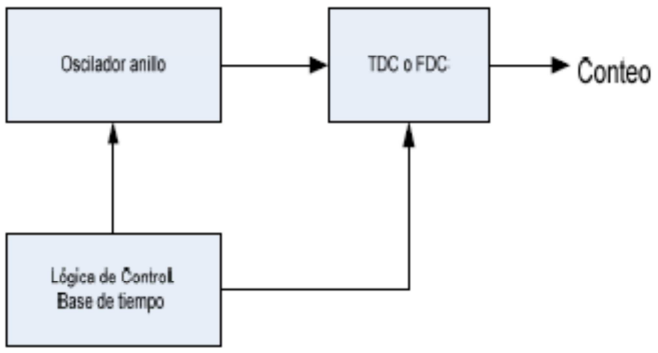

Figura 2-3: Diagrama de bloques de un sensor de temperatura basado en oscilador anillo. Extraído de [1]

3 Diseño

3.1 Elementos físicos para la obtención de datos

Para poder obtener los datos de los experimentos realizados en la FPGA se han utilizado los siguientes elementos:

- Plataforma prototipo FPGA Virtex-5 XC5VLX30-FF676.
- Osciladores anillo.
- Interfaz RS232 (UART).
- Horno Pselecta.
- Fuente de alimentación Agilent N6705A.
- Osciloscopio Agilent MSO-X 3014ª.
- Placa de medición de temperatura EMC140X/2X

En conjunto todos ellos forman el siguiente esquema y se irán explicando más detenidamente a continuación:

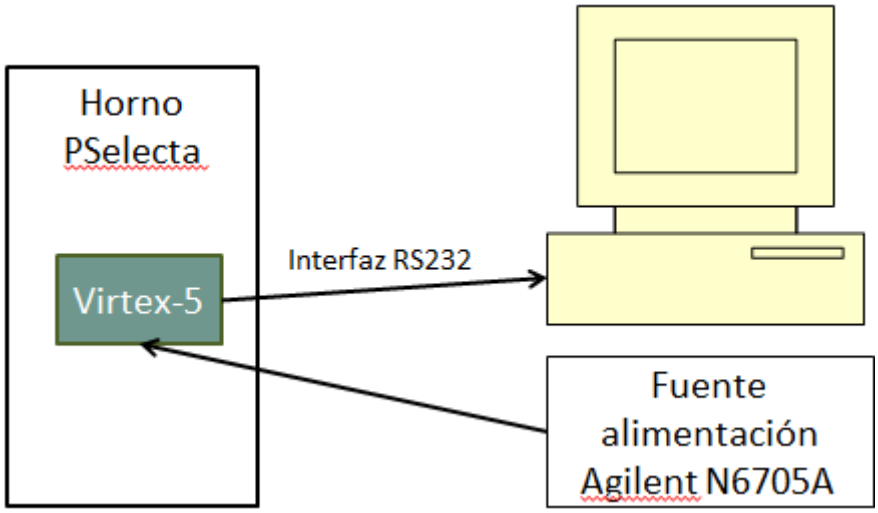

Figura 3-1: Diagrama de la posición de los equipos para recoger las muestras.

3.1.1 Plataforma prototipo FPGA Virtex-5

La placa de pruebas que se ha elegido para este proyecto es una plataformaVirtex-5 XC5VLX30-FF676. Los motivos de ésta elección son:

- Con ésta FPGA se puede regular muy fácilmente y con bastante exactitud el voltaje de núcleo, el cual es un factor muy importante en la variación de la frecuencia de oscilación de los sensores anillo.

- Se cuenta con el System Monitor, el cual es un sensor que se encuentra en el centro de la FPGA que mide los parámetros de funcionamiento físicos como tensiones de alimentación y temperaturas los cuales se han utilizado para calibrar los sensores.
- El sistema se completa con una interfaz para enviar los datos al ordenador y poder trabajar con ellos. Para ello se cuenta con una interfaz externa RS232 que se explicará en uno de los puntos siguientes.

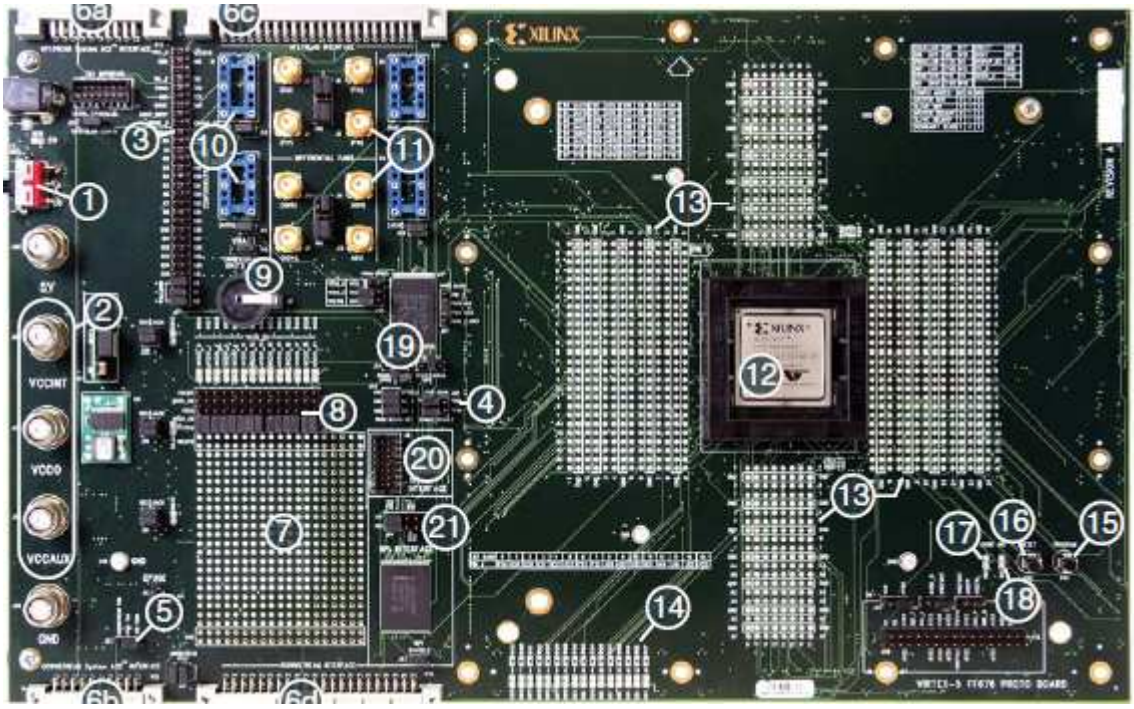

Figura 3-2: Representación de la FPGA que se ha utilizado.

3.1.1.1 System Monitor

Como ya se ha indicado anteriormente, esta FPGA cuenta con un diodo interno localizado en el centro de la placa (marcado en rojo y rodeado en la Figura 3-3) el cual es monitoreado por un componente denominado System Monitor. Este elemento tiene un conversor analógico digital (ADC) en torno a 10-bit y 200-ksps (kilomuestras por segundo). Este ADC puede medir voltaje y temperatura internos pero también se cuenta con otras 16 entradas externas que pueden ser elegidas por el usuario a partir de las cuales se pueden controlar otros parámetros, como establecer limitaciones a la tensión y la temperatura o activar y desactivar alarmas.

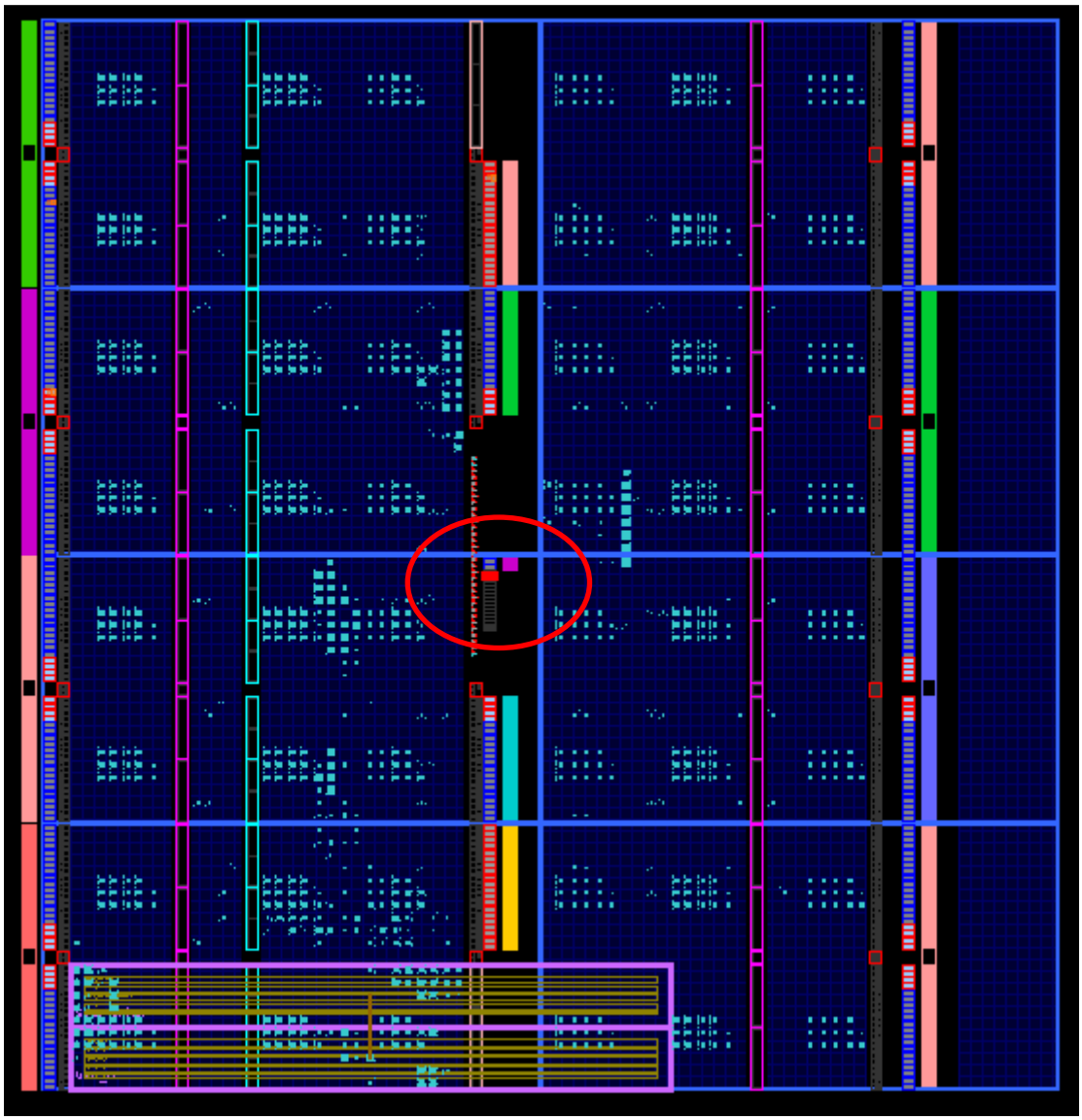

Figura 3-3: Localización del sensor del System Monitor.

Por tanto, el sensor de temperatura produce una salida de tensión que es proporcional a la temperatura del chip.

La tensión de salida de este sensor es digitalizada por el ADC para producir una salida digital de 10 bits. La Figura 3-4 ilustra la función de transferencia de salida digital para este sensor de temperatura.

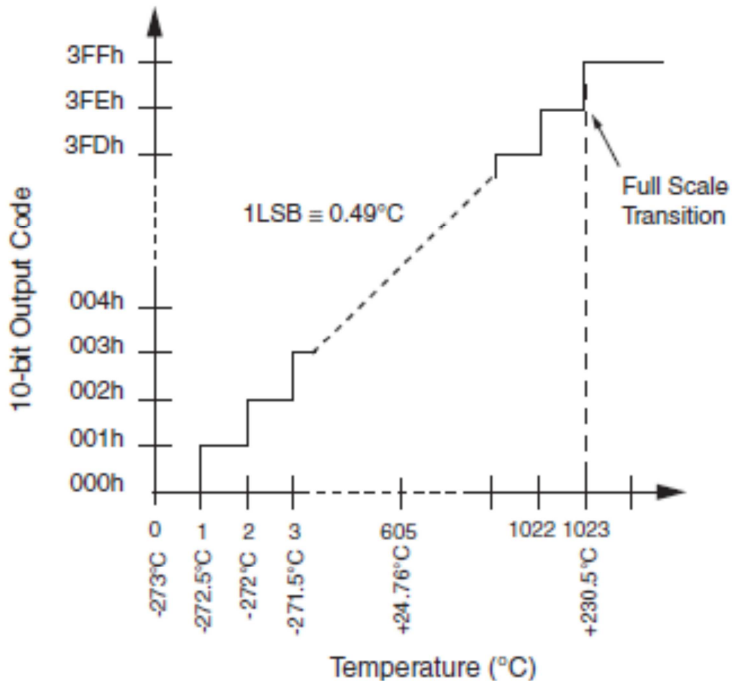

Figura 3-4: Función de transferencia de temperatura del sensor. Extraída de [6].

Por tanto, se obtiene la siguiente fórmula para obtener la temperatura a partir del valor proporcionado con el System monitor:

$$temperatura(ºC) = \frac{ADC * 503.975}{1024} - 273.15$$

Como en este proyecto se ha obtenido el valor de ADC con 16 bits en lugar de 10 bits, se cambia el 1024 por 65536 que serían 2^{16} en lugar de 2^{10}. Con este método se puede obtener un error de medición máximo de ± 4 ℃ .

Para obtener el valor del voltaje interno se utiliza una función de transferencia muy parecida.

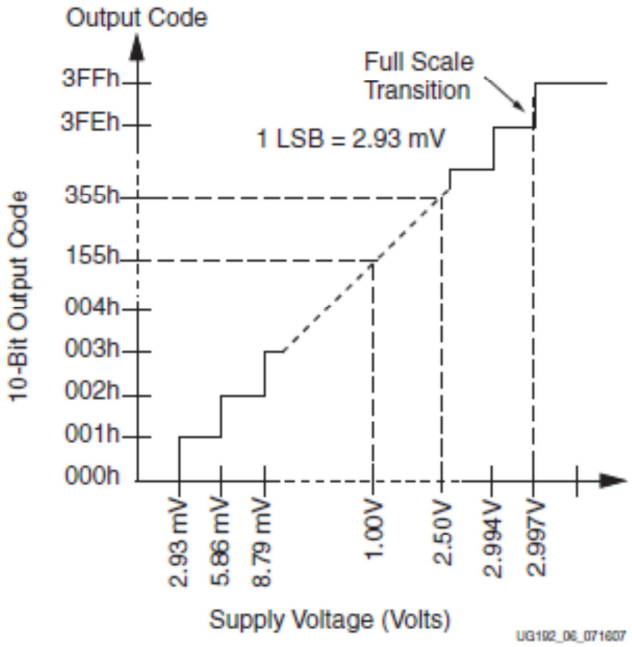

Figura 3-5: Función de transferencia de la fuente de alimentación. Extraída de [6].

Por tanto se obtiene la siguiente fórmula para obtener el voltaje a partir del ADC:

$$Voltaje(V) = \frac{ADC}{1024} * 3V$$

Igual que en el caso de la temperatura, como el tamaño del ADC que se obtiene es de 16 bits, hay que modificar el 1024 por 65536 [6].

3.1.2 Diseño del oscilador anillo

Los osciladores anillo que se han implementado para este proyecto han sido desarrollados en el entorno ISE Design Suite 14.2. La estructura es similar a la usada en [2] y se ha diseñado para que pueda instanciarse en cualquier fichero top[1] fácilmente. En la Figura 3-6 se muestra un sensor anillo:

[1] Fichero top: Fichero que se utiliza para juntar todos los elementos de un proyecto que se han ido diseñando en ficheros independientes.

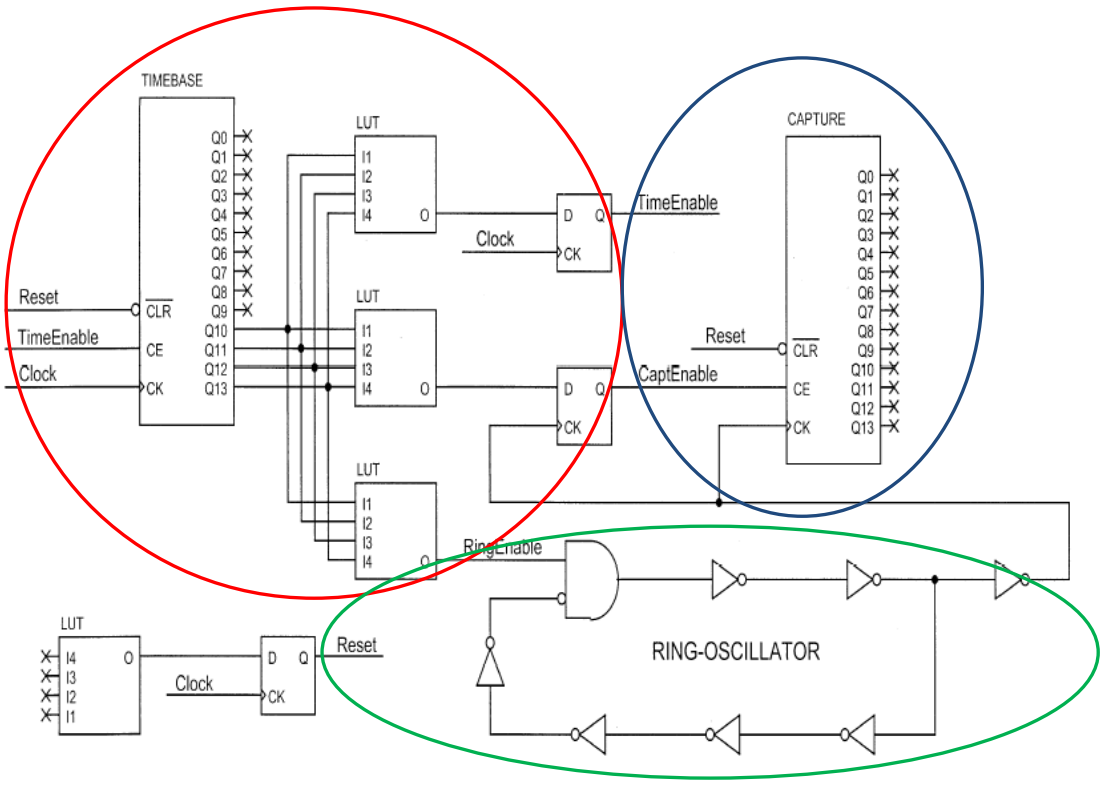

Figura 3-6: Implementación de un sensor de temperatura basado en oscilador anillo para FPGA. Extraída de [2].

Este esquema consta de tres partes diferenciadas:

- La lógica de control (rodeada en rojo): es el módulo de TIMEBASE que es el encargado de la habilitación del sensor y controlar el tiempo de habilitación y captura.
- El sensor de temperatura (rodeada en verde): que corresponde al RING-OSCILLATOR compuesto por seis inversores (aunque este número en los experimentos variará), una puerta NAND y un buffer de salida.
- Un último bloque indicado en el apartado Osciladores anillo2.2 como FDC (rodeada en azul): que es el contador CAPTURA el cual va a proporcionar la cuenta del oscilador con la cual se obtendrá la frecuencia de oscilación y el retardo.

Esta implementación, con el fin de hacer un código lo más simple posible, está dividido en dos partes. La primera parte en la que solo se incluye el RING-OSCILLATOR está en el anexo A. Y la segunda parte en la que se incluye el módulo de control y el FDC está en el anexo B. Ésta segunda parte se expone para la implementación de 48 sensores pero está diseñado para que dicho número sea fácilmente variable con solo modificar una constante y la posición en la que se quieran instanciar dichos sensores.

Al implementar uno de estos sensores con VHDL en el programa ISE Design como se explicará más adelante, se obtiene el siguiente reporte de síntesis:

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice LUTs	53	19,200	1%	
Number used as logic	53	19,200	1%	
Number using O6 output only	53			
Number of occupied Slices	14	4,800	1%	
Number of LUT Flip Flop pairs used	53			
Number with an unused Flip Flop	53	53	100%	
Number with an unused LUT	0	53	0%	
Number of fully used LUT-FF pairs	0	53	0%	
Number of slice register sites lost to control set restrictions	0	19,200	0%	
Number of bonded IOBs	2	400	1%	
Number of RPM macros	1			
Average Fanout of Non-Clock Nets	1.02			

Figura 3-7: Reporte de síntesis de un sensor sin contador.

3.1.2.1 Diseño de la red de sensores

Para poder estudiar cómo se distribuye el calor a lo largo de toda la FPGA, se ha diseñado una red de sensores que se localizan por toda la superficie del chip. Se ha realizado con 48 sensores, divididos en 8 filas por 6 columnas, porque se vio que eran suficientes para la FPGA. Pero se pueden variar fácilmente tanto el número de sensores como las posiciones en las que se encuentran. La Figura 3-8 muestra cómo se han distribuido los 48 sensores (marcados en blanco) y la lógica de control (marcado en azul):

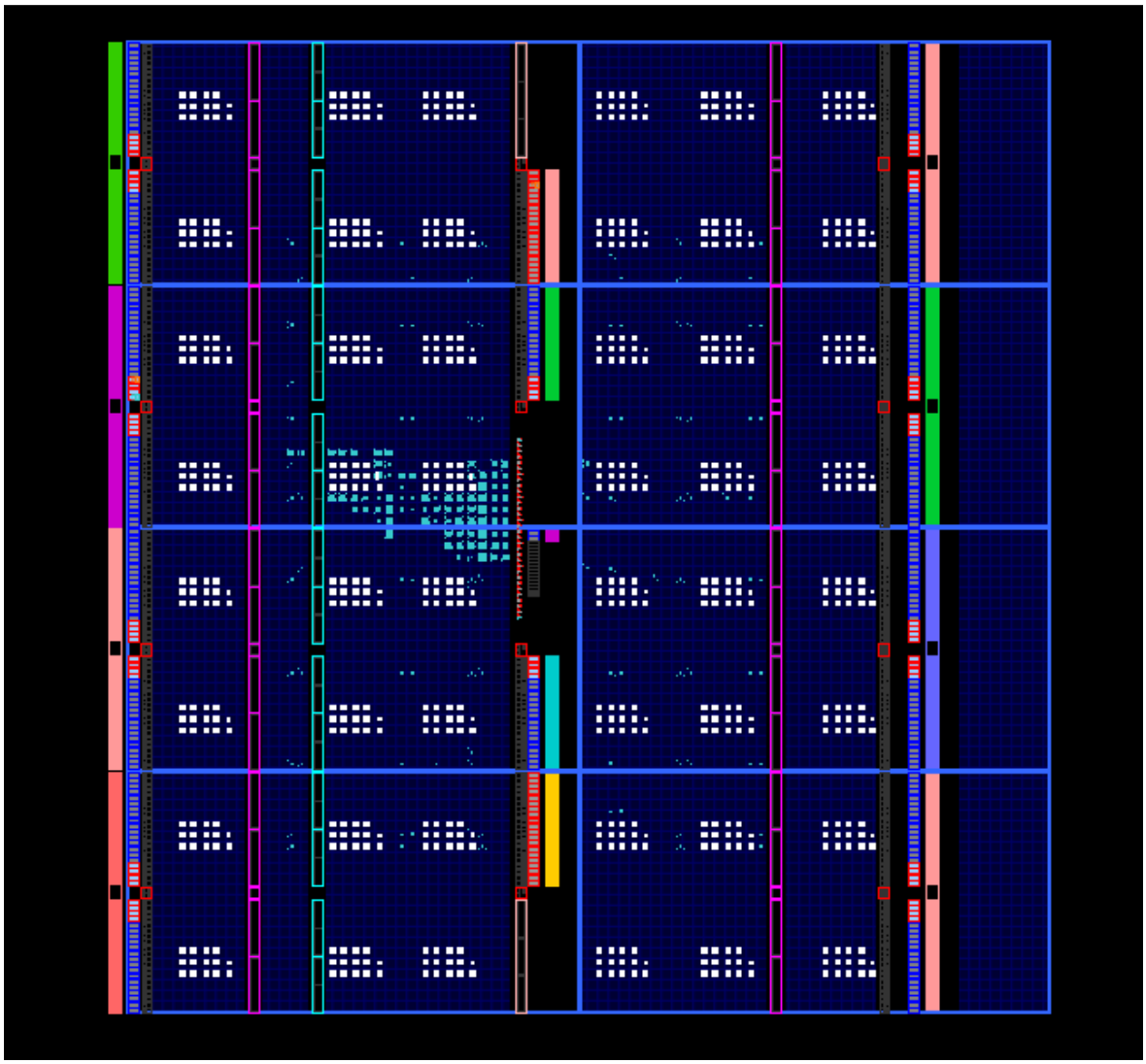

Figura 3-8: Distribución de los 48 sensores utilizados para las pruebas y su lógica de control.

Con el fin de controlar la puesta en marcha de los sensores y obtener la frecuencia de oscilación de cada uno, se ha creado una lógica de control (incluida también en el anexo B). Ésta consta básicamente de una máquina de estados controlada por el reloj del sistema, de 25 MHz y el reset. Por tanto, cuando está el reset a '0' el programa no funciona pero cuando éste pasa a ser '1' el programa empieza a funcionar con la frecuencia del reloj principal y activa el primer sensor con el cual se obtiene otro reloj secundario con el que se obtiene la frecuencia de oscilación. En un primer lugar se cuenta un tiempo offset de inicio "delay_initial" para asegurar que la sintonía del oscilador se estabilice. En segundo lugar hay un tiempo de habilitación del FDC "time_enable", es decir del contador de CAPTURA, durante el cual se obtiene un número de cuenta que indicará la frecuencia de oscilación del anillo. Por último, se deja un tiempo de captura "time_capt" que es el tiempo que tarda el UART (interfaz RS232) en enviar el dato del número de cuenta de ese sensor. En la Figura 3-9 se muestra un esquema de éste proceso:

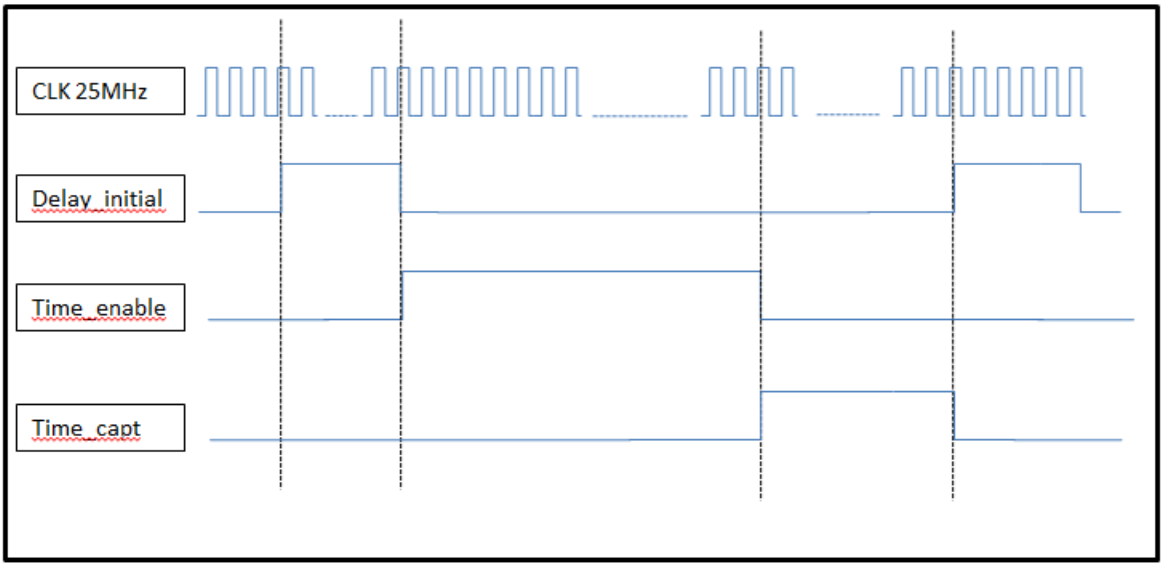

Figura 3-9: Esquema de los tiempos de la lógica de control.

Los tiempos que se han elegido para la realización de este proyecto son 625 pulsos para el "delay_initial" y 30.000 pulsos para "time_enable", ambos dependientes del reloj principal de 25 MHz. El tiempo de captura es el necesario para que la UART envíe los datos, que dependerá de la cantidad de datos que se quieran enviar.

Cuando se ha realizado todo éste proceso se desactiva el primer sensor para que no tenga sobrecalentamiento innecesario y se activa el siguiente. Cuando todos los sensores han sido activados y enviada su cuenta, es decir, se ha realizado una vuelta entera, se comienza otra vez desde el primero para tener distintas muestras. Para este proyecto se han considerado suficientes con 256 muestras enviadas de cada sensor.

Para activar y desactivar los sensores se usa una señal de habilitación llamada "ring_enable" que es un vector de 48 posiciones en la que cada una de ellas corresponde con un sensor. Por ejemplo, al darle el valor ring_enable(7)='1', el sensor 7 comienza a funcionar y se recogen los valores del mismo.

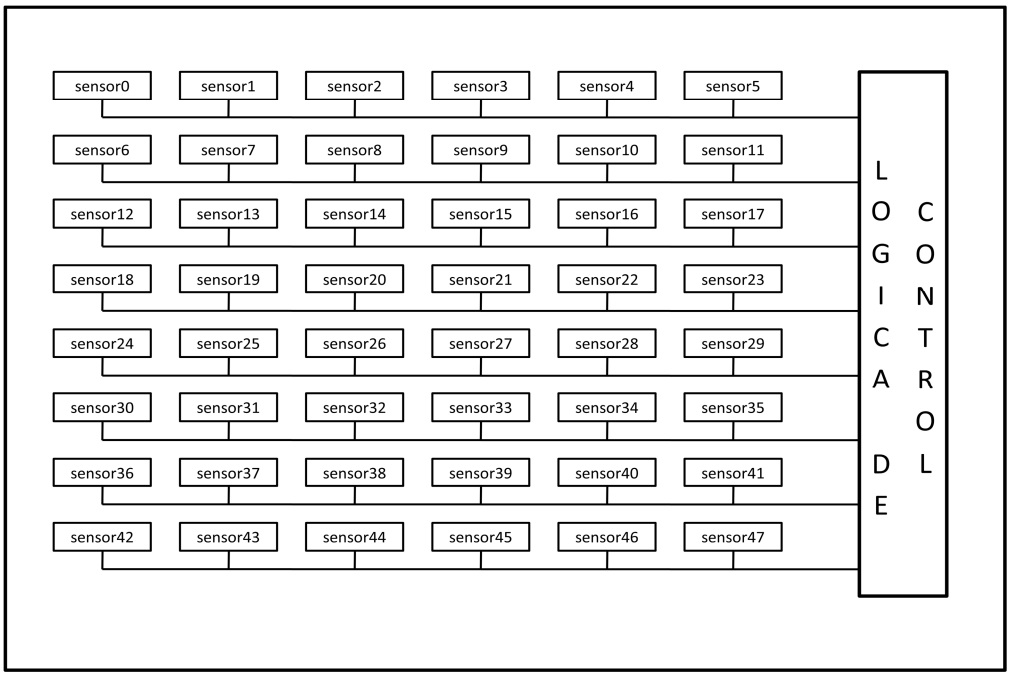

Figura 3-10: Red de sensores con lógica de control.

3.1.3 Uso de la interfaz RS232 como elemento de envío de datos

Como se ha indicado en apartados anteriores para realizar el envío de datos al ordenador se utiliza la interfaz RS232 (UART). Puesto que la plataforma de evaluación Virtex-5 no dispone de ella se ha realizado un circuito para adaptar los voltajes entre plataforma y ordenador. El chip que se ha utilizado es el MAX 232 IC [8] y se representa en la Figura 3-11:

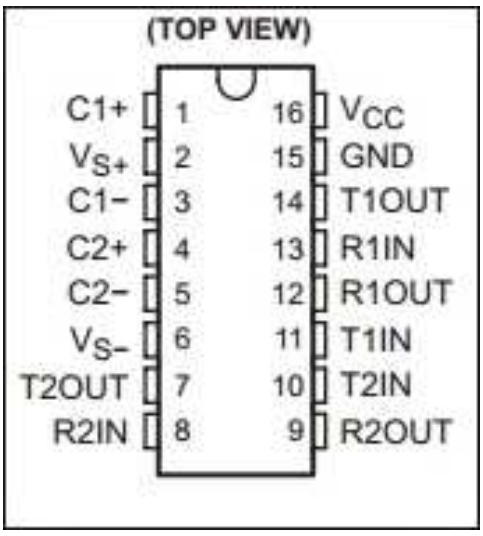

Figura 3-11: Esquema del circuito de MAX 232 IC. Extraído de [8].

Este circuito convierte las señales de un puerto serie RS232 en señales compatibles con los circuitos lógicos. Es alimentado con 5V gracias a la fuente de alimentación Agilent que se comentará más adelante y el pin de transmisión es conectado con un pin I/O de la Virtex-5. Según hoja de especificaciones [9] el pin B22 de la FPGA se corresponde con esas características.

Respecto a la implementación en ISE se ha realizado un código de envío y recepción de la UART que se encuentra en el anexo C. El control de cuándo y que enviar se encuentra también en el anexo B.

El control de envío consta de una máquina de estados en el que se registran los valores que se quieren enviar y posteriormente son enviados uno a uno. Puesto que los datos que se quieren enviar son de 16 bits y el UART solo envía datos de 8 bits, éstos se tratan y se parten en dos bloques y por tanto hay que tenerlo en cuenta en la recepción.

Puesto que se envían datos de los 48 sensores pero también de la temperatura y el voltaje que se obtienen con el System Monitor se ha decidido el siguiente orden. Primero se hace un envío de todos los sensores y después se envía una única vez el valor de la temperatura y el voltaje puesto que son los datos de un único punto. Y así sucesivamente hasta que se hallan enviado las 256 muestras de todos los sensores. Por tanto, se recibirán un total de 12.800 muestras tal y como se indica en la Tabla 3-1 y en la Figura 3-12:

Tabla 3-1: Nº total de muestras recogidas.

Elemento	Nº elementos	Nº de muestras	Total
Sensores	48	256	12.288
Temperatura System Monitor	1	256	256
Voltaje núcleo System Monitor	1	256	256
Total	**50**	**768**	**12.800**

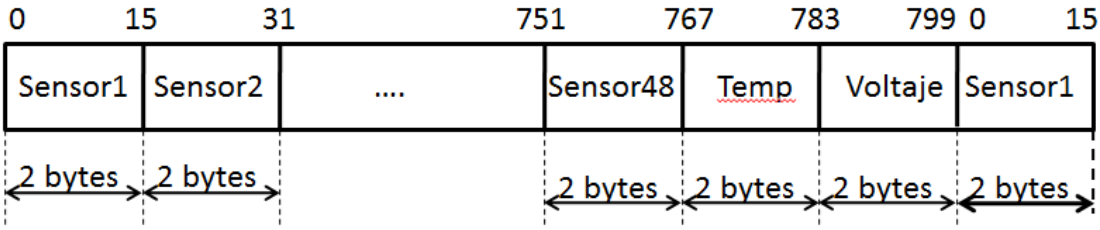

Figura 3-12: Esquema de la trama de datos de una iteración.

3.1.4 Otros materiales externos

Para realizar este proyecto se han utilizado otros materiales que son:

- Un horno Pselecta donde se ha introducido la Virtex-5 y se ha variado la temperatura, desde los 40 ºC hasta los 90 ºC, para obtener una curva de calibración de los sensores. Esto sólo se ha realizado para un número limitado de ellos que se encuentran cercanos al System Monitor. A continuación se ponía la temperatura deseada y se dejaba durante 45 minutos para que se alcanzase la estabilización térmica. Posteriormente se recogieron las muestras deseadas.

Figura 3-13: Representación del horno.

- Fuente de alimentación Agilent N6705A con la cual se alimenta la FPGA y por tanto, se controla el voltaje del núcleo. También se alimenta el UART. Esta fuente de alimentación es muy precisa [10] y gracias a ello se ha podido controlar la alimentación de la FPGA con la exactitud suficiente para hacer las pruebas deseadas.

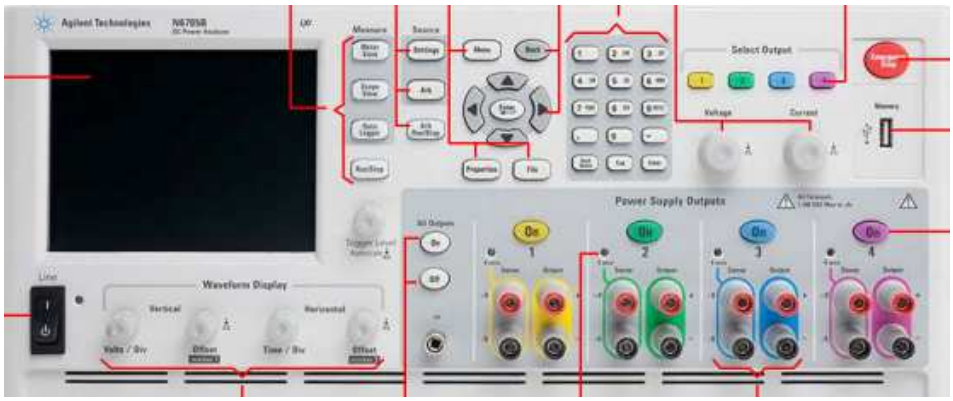

Figura 3-14: Representación de la fuente de alimentación.

- Osciloscopio Agilent MSO-X 3014A de 100MHz y 4 GSa/s. Este instrumento ha sido muy útil para observar las señales que se obtenían del oscilador anillo y ver si eran como se esperaban.

Figura 3-15: Representación del osciloscopio.

- Placa de medición de temperatura EMC140X/2X. Es un instrumento para medir la temperatura de un transistor externo. En primer lugar se pensó utilizar este instrumento para realizar la curva de calibración y obtener la temperatura interna de la FPGA pero luego se vio que el System Monitor podía ser más útil para obtener ese dato ya que proporcionaba también el valor del voltaje interno del núcleo, por lo que se descartó su utilización [18].

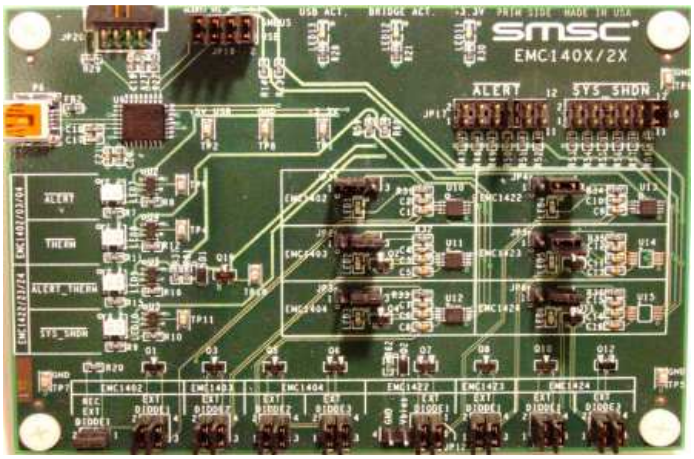

Figura 3-16: Representación del sensor de temperatura. Extraído de [18].

3.2 Aplicaciones

3.2.1 ISE Design Suite 14.2

ISE Design Suite 14.2 ha sido la herramienta de programación fundamental de este proyecto. Con ella se han diseñado, implementado y/o simulado todos los elementos necesarios para obtener los datos suficientes para crear un mapa térmico o una curva de calibración. El lenguaje utilizado ha sido el VHDL, que es utilizado para describir la totalidad del diseño.

Dentro de este programa existen otras herramientas que también han sido utilizadas como por ejemplo:

- PlanAhead [12] te da la ubicación de todos los elementos que se han implementado en la FPGA y que slice se utilizan para cada caso. Ver figuras:

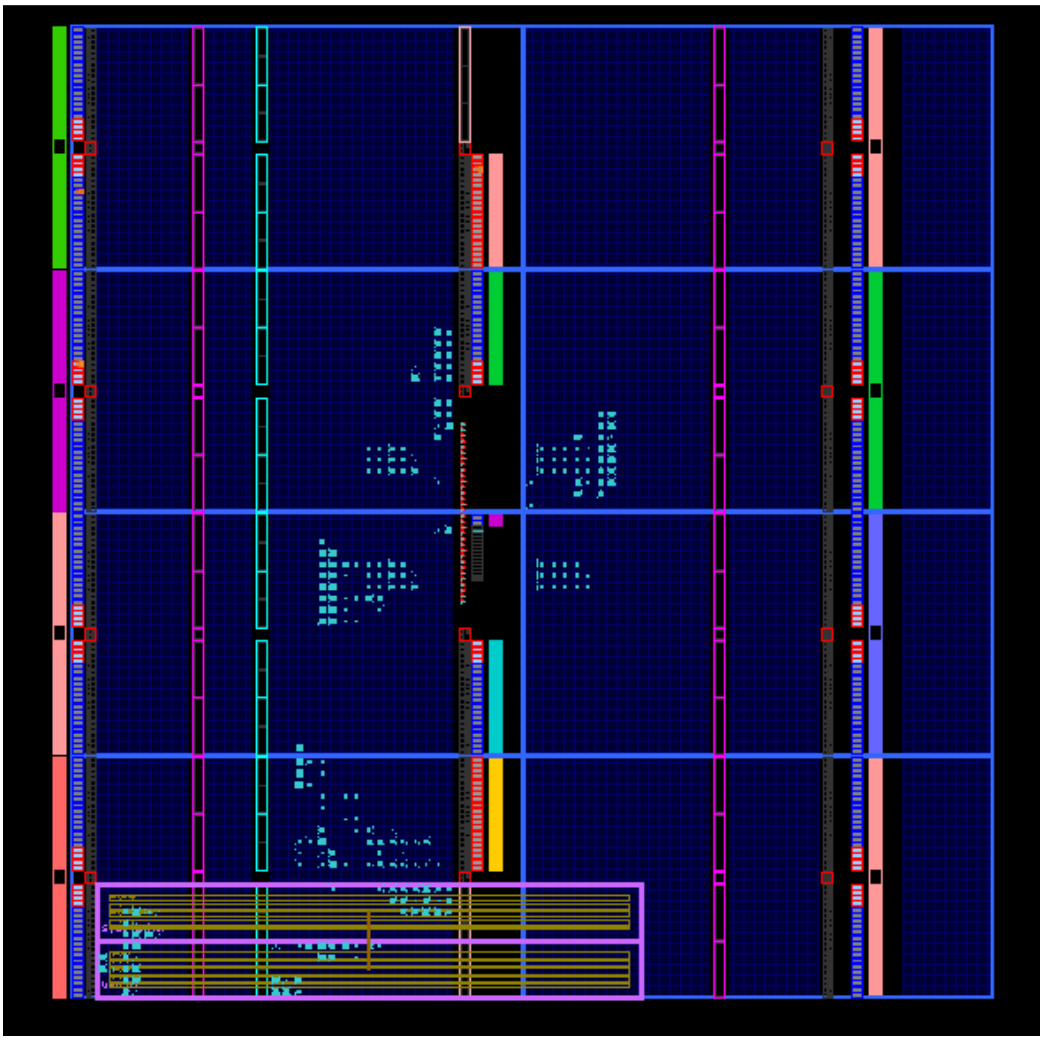

Figura 3-17: Localización de los elementos en la Virtex-5.

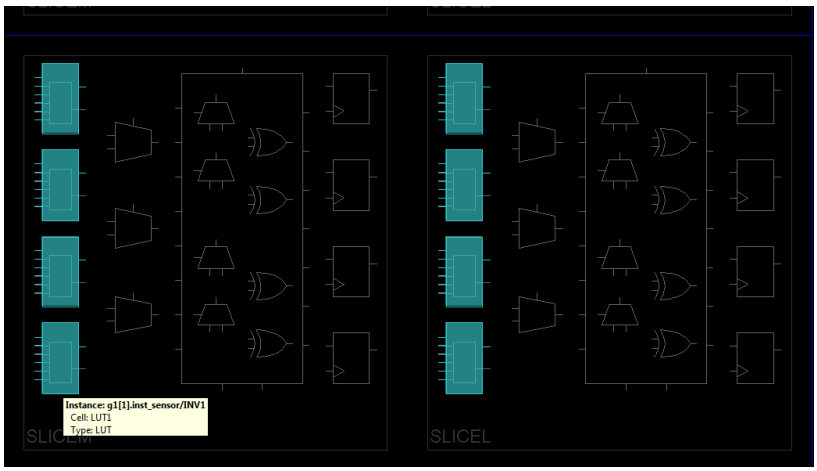

Figura 3-18: Ampliación de la Figura 3-17.

También se puede obtener el esquemático de cada señal o elemento, como por ejemplo el que se muestra en la Figura 3-19:

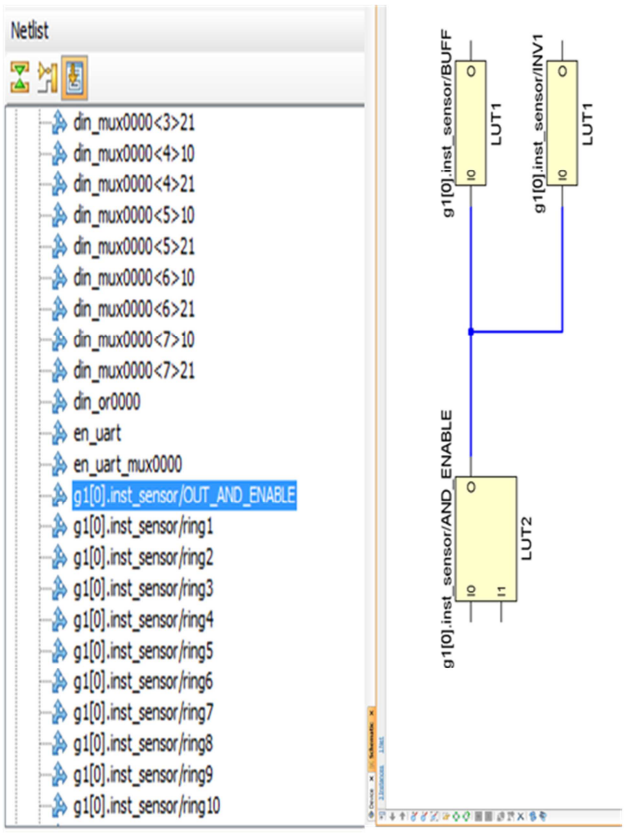

Figura 3-19: Ruta de la señal OUT_AND_ENABLE con el PlanAhead.

En este caso se puede observar cual es la ruta que hace la señal OUT_AND_ENABLE que va desde la AND al buffer y el primer inversor del sensor anillo. Con esto se puede ver con claridad que se hace con las señales.

- FPGA Editor [11]. Esta herramienta se ha utilizado para que el ruteado de los osciladores anillo sea siempre el mismo en las diferentes compilaciones y no influya en la variación de la temperatura. También se ha utilizado para la localización de las instanciaciones de la UART y el control del System Monitor.

  El ruteado de los sensores de temperatura se realiza de la siguiente manera:

  1. Se abre el programa desde el ISE y se entra en la ventana Tools =>DirectRoutingConstrains.

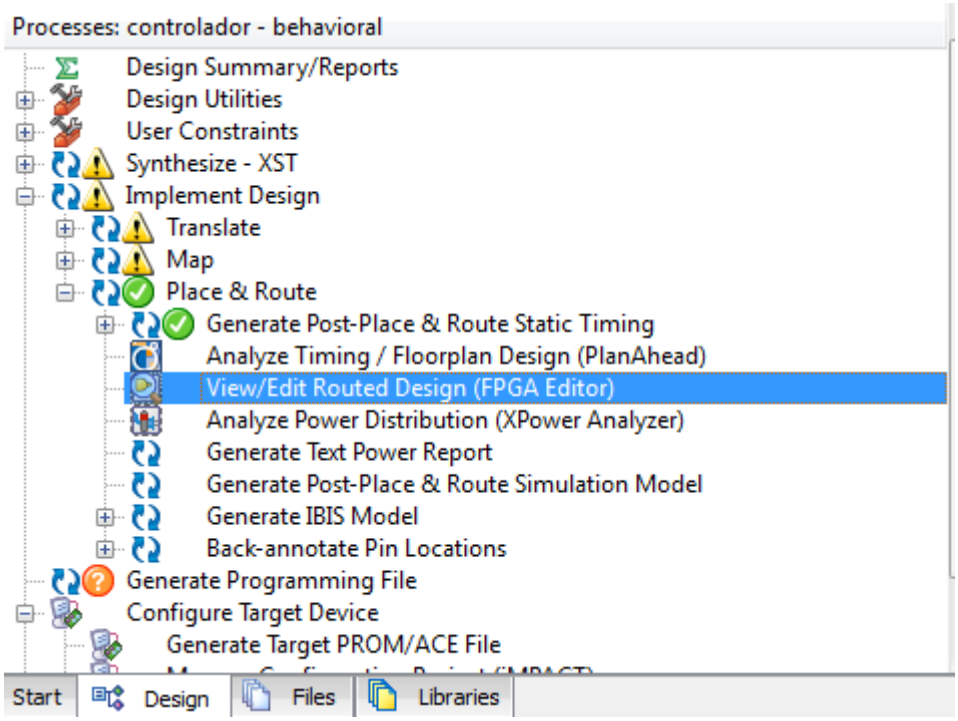

Figura 3-20: Paso 1 en la utilización FPGA Editor.

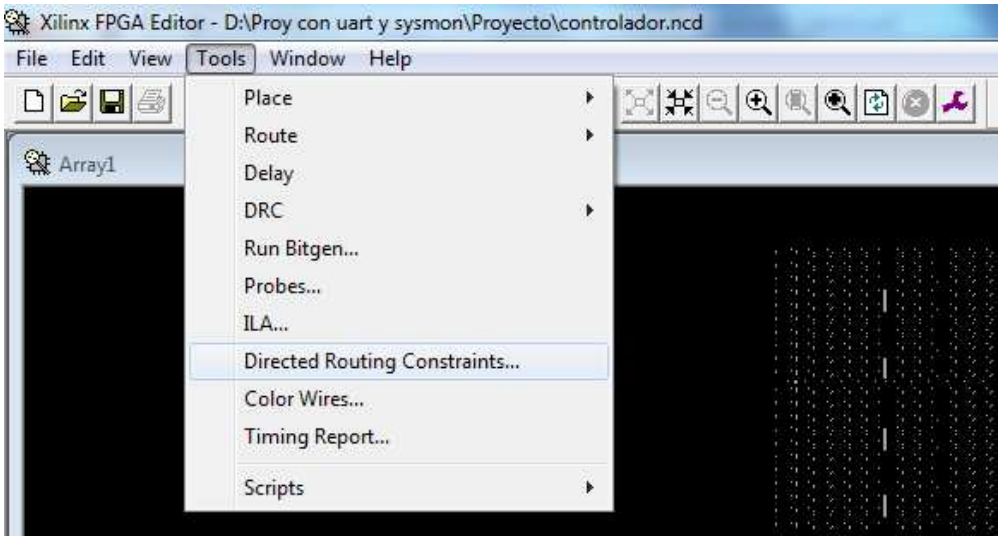

Figura 3-21: Paso 2 en la utilización FPGA Editor.

2. En esa ventana se escogen las señales que se quieran rutear (en este caso todos los sensores), se pinchan las opciones que se muestran en la Figura 3-22, se elige la ruta y el fichero en el que se quieran grabar los datos y se da Apply y Close.

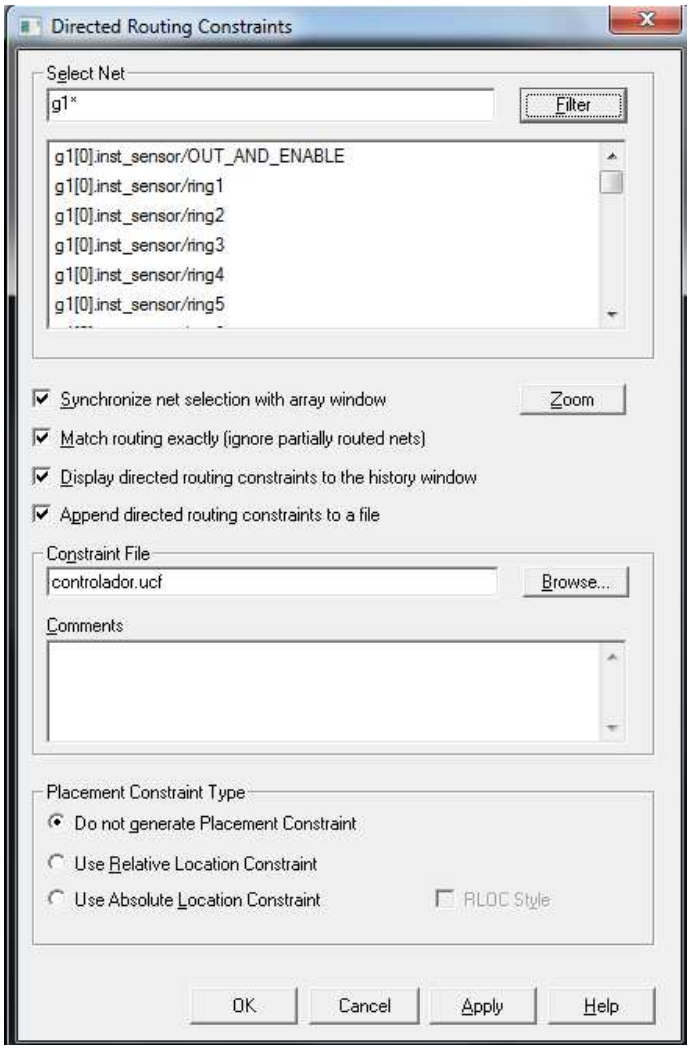

Figura 3-22: Paso 3 en la utilización FPGA Editor.

Siguiendo estos pasos y copiando lo que salga del fichero que se halla elegido dentro del .ucf que se use en el proyecto, se obtendrá un ruteado fijo. En el anexo E se expone un ejemplo de esto con un sensor anillo.
Además para colocar los elementos como la UART en un lugar concreto, se puede incluir en el fichero .ucf el siguiente comando, variando lo que está marcado en azul para cada caso:

```
INST "uart_inst*" AREA_GROUP = "uart_tx";
AREA_GROUP "uart_tx" RANGE=SLICE_X0Y5: SLICE_X37Y9;
```

3.2.2 Matlab

Para el diseño de la interfaz de obtención de datos y tratamiento de los mismos, se ha utilizado Matlab.

En primer lugar, indicándole que los datos provienen de un puerto serie, se crea un vector que recoge todos los datos que vienen del UART. Hay que tener en cuenta que los parámetros que vienen por el puerto serie deben ser los mismos que se hayan puesto en el programa de la FPGA para que no haya ningún problema de sincronización y los datos que recibamos sean erróneos. Dicho código se encuentra en el anexo D.

Dicho vector tiene los datos tal y como se explicó en el apartado 3.1.3, es decir, primero las muestras de los osciladores anillo y luego los de la temperatura y el voltaje obtenidos del System Monitor. Por tanto, primero hay que separarlos para poder tratarlos y hay que tener en cuenta que cada dos bytes se obtiene el valor total. Para obtener el valor en el sistema decimal hay que realizar el siguiente cálculo:

$$valor\ decimal = byte1 * 256 + byte0$$

Finalmente, una vez obtenido el valor en decimal hay que diferenciar entre la cuenta del sensor del anillo, la temperatura y el voltaje para obtener los valores que se desean.

Para el caso de la temperatura y el voltaje hay que aplicar las fórmulas indicadas en el punto 3.1.1.1, teniendo en cuenta que el valor de “ADC” es el que aquí se indica como “valor decimal”.

Para el caso de la cuenta del oscilador anillo, lo que queremos obtener es la frecuencia de oscilación por tanto hay que realizar los siguientes cálculos:

1. Para obtener el tiempo en nanosegundos de lo que tarda en realizarse una sola oscilación, hay que tener en cuenta el número de oscilaciones que se realizan, que son las que obtenemos en el resultado de “valor decimal” indicado arriba, y el tiempo de captura de esas oscilaciones (“time_enable” como se indica en apartados anteriores). Como en el programa se ha especificado que el “time_enable” sea de 30.000 pulsos con un reloj de 25 MHz (40 ns), se obtiene un tiempo total de 1.200.000 ns. Por tanto:

$$t_{oscilacion} = \frac{1200000}{valor\_decimal}\ (ns)$$

2. Por último, para hallar la frecuencia de oscilación, solamente hay que obtener la inversa de dicho tiempo y multiplicarla por 1.000 para obtener el valor en MHz:

$$f_{oscilacion} = \frac{1000}{t_{oscilacion}}\ (MHz)$$

Una vez se tienen estos valores solo hay que colocarlos como se quieran presentar para obtener las gráficas que se mostraran en los siguientes apartados.

4 Integración, pruebas y resultados

Para probar el funcionamiento y la aplicabilidad de los osciladores anillo que se han diseñado e implementado para este proyecto, se realizan una serie de pruebas que se exponen a continuación.

En primer lugar se realiza un experimento para determinar cuántos inversores son necesarios para que la frecuencia de oscilación obtenida de los sensores anillo sea lo más fiable posible. A continuación, se realiza una curva de calibración con osciladores anillo cercanos al sensor de System Monitor. Y finalmente se obtendrá la frecuencia en 48 sensores colocados por toda la FPGA.

Todos los datos recogidos en el experimento se encuentran en un DVD anexo al proyecto.

4.1 Primer experimento

Con el fin de saber cuál es el número adecuado de inversores que hay que poner para que el oscilador anillo sea lo más preciso posible se hizo una primera serie de experimentos en los que se fue variando el número de inversores del sensor.

Para ello se instanciaron dos sensores en las posiciones X20Y33 (sensor 1) y X29Y33 (sensor 2) y se fueron variando desde 5 inversores hasta 51. De este experimento se recogió únicamente la cuenta del oscilador anillo durante el tiempo de captura y a partir de ahí se obtuvo la frecuencia (Para este experimento no se usó el System Monitor por tanto no se recogen resultados). Se repitió el experimento 256 veces y se obtuvo la desviación estándar promedio. Todo esto se realizó a una temperatura ambiente, 21 ºC y con un voltaje de núcleo de 1V. En las siguientes tablas y figuras se muestran los resultados obtenidos:

Tabla 4-1: Desviación estándar de ambos sensores.

Nº inversores	Des. Estándar sensor 1 (Hz)	Des. Estándar sensor 2 (Hz)
5	36483.45	21426.07
11	17358.15	12487.15
15	13661.69	12678.53
21	7926.61	6906.87
25	5303.83	5157.87
31	3813.85	4465.65
35	3083.03	4300.92
41	2961.13	3081.91
45	2162.50	3572.86
51	2771.60	3300.53

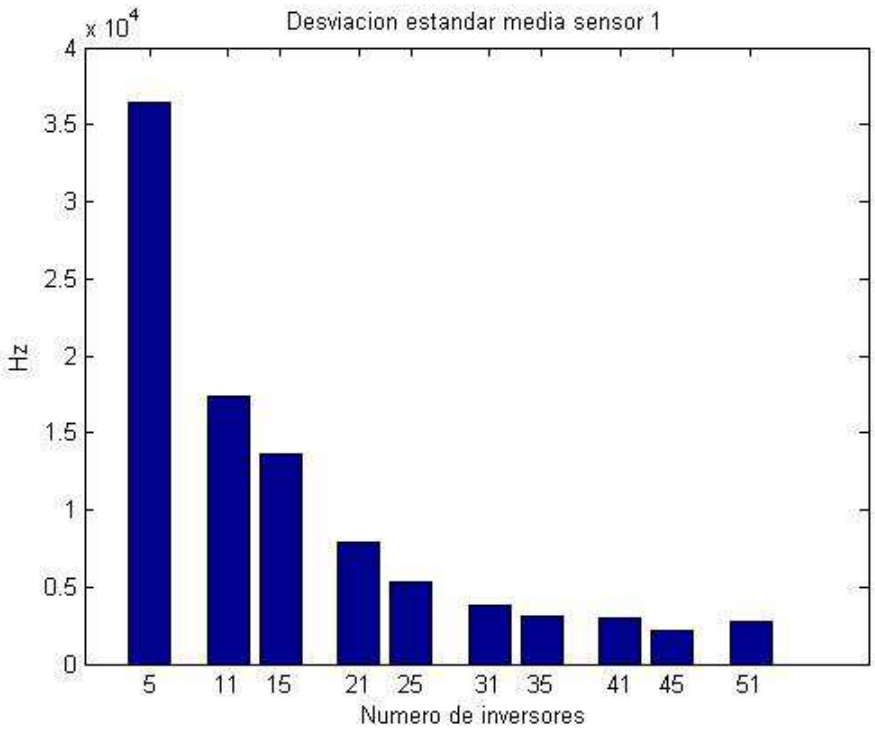

Figura 4-1: Desviación estándar media del sensor 1 a temperatura ambiente, 21 ºC y voltaje constante de 1V.

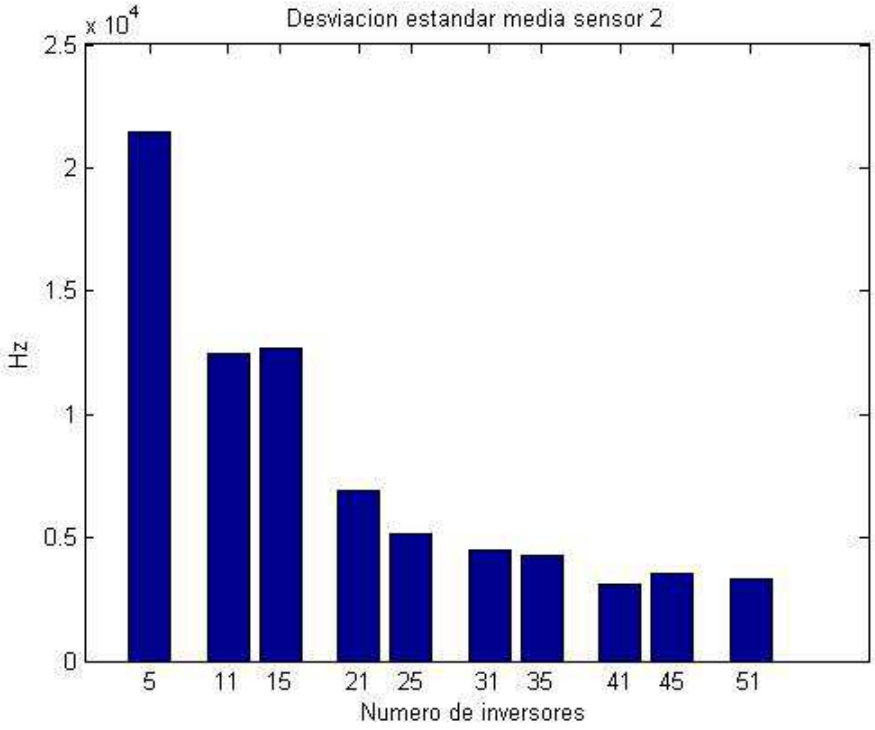

Figura 4-2: Desviación estándar media del sensor 2 a temperatura ambiente, 21 ºC y voltaje constante de 1V.

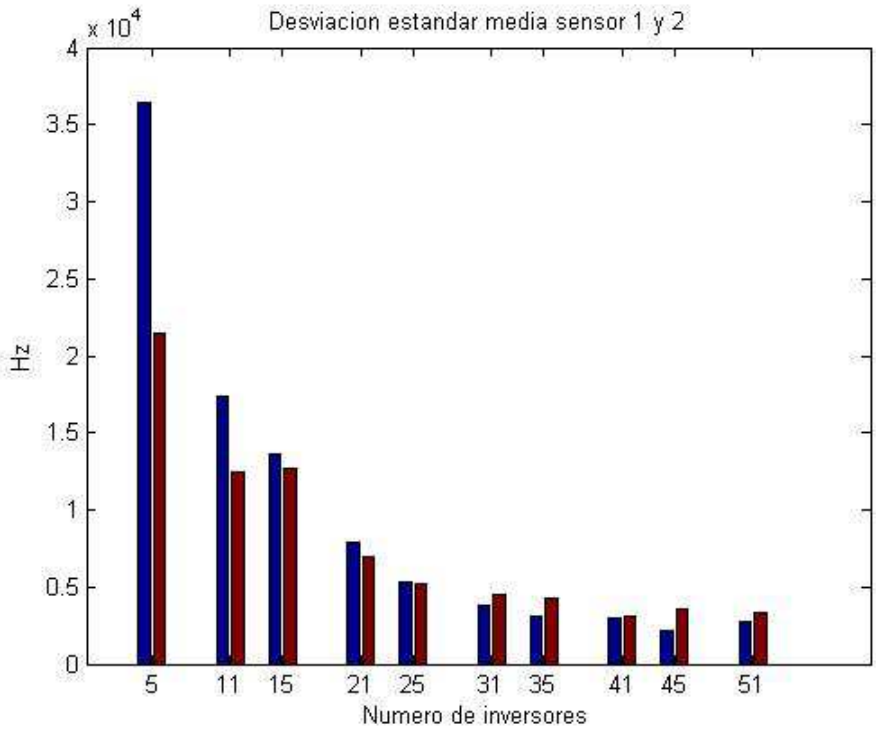

Figura 4-3: Comparación de la desviación estándar media de ambos sensores. A temperatura ambiente de 21ºC y voltaje constante de 1V.

A partir de estos experimentos se puede observar que desde los 31 inversores la curva tiende a estabilizarse lo que indica que a partir de dicho número se obtienen aproximadamente los mismos resultados sin que influya el número de inversores. Esto coincide aproximadamente con los resultados expuestos en [1].

A raíz de los experimentos anteriores se ha determinado que se usarán 51 inversores ya que aunque desde los 31 inversores se ve que la curva tiende a estabilizarse el error sigue descendiendo progresivamente. Por tanto, se ha considerado que 51 no es un número muy elevado y se consigue unos resultados más exactos.

4.2 Segundo experimento

Una vez establecido que el número de inversores será 51 se han realizado otra serie de experimentos para obtener la curva de calibración de dos de ellos.

Para ello, se han colocado 2 osciladores anillo alrededor del sensor del System Monitor en las posiciones X20Y33 (sensor 1) y X29Y33 (sensor 2). Y los elementos pertenecientes a la UART y el control del System Monitor en la parte inferior izquierda del DIE tal y como se observa en la Figura 4-4. Lo marcado en blanco son los sensores. Posteriormente se introdujo la FPGA en el horno.

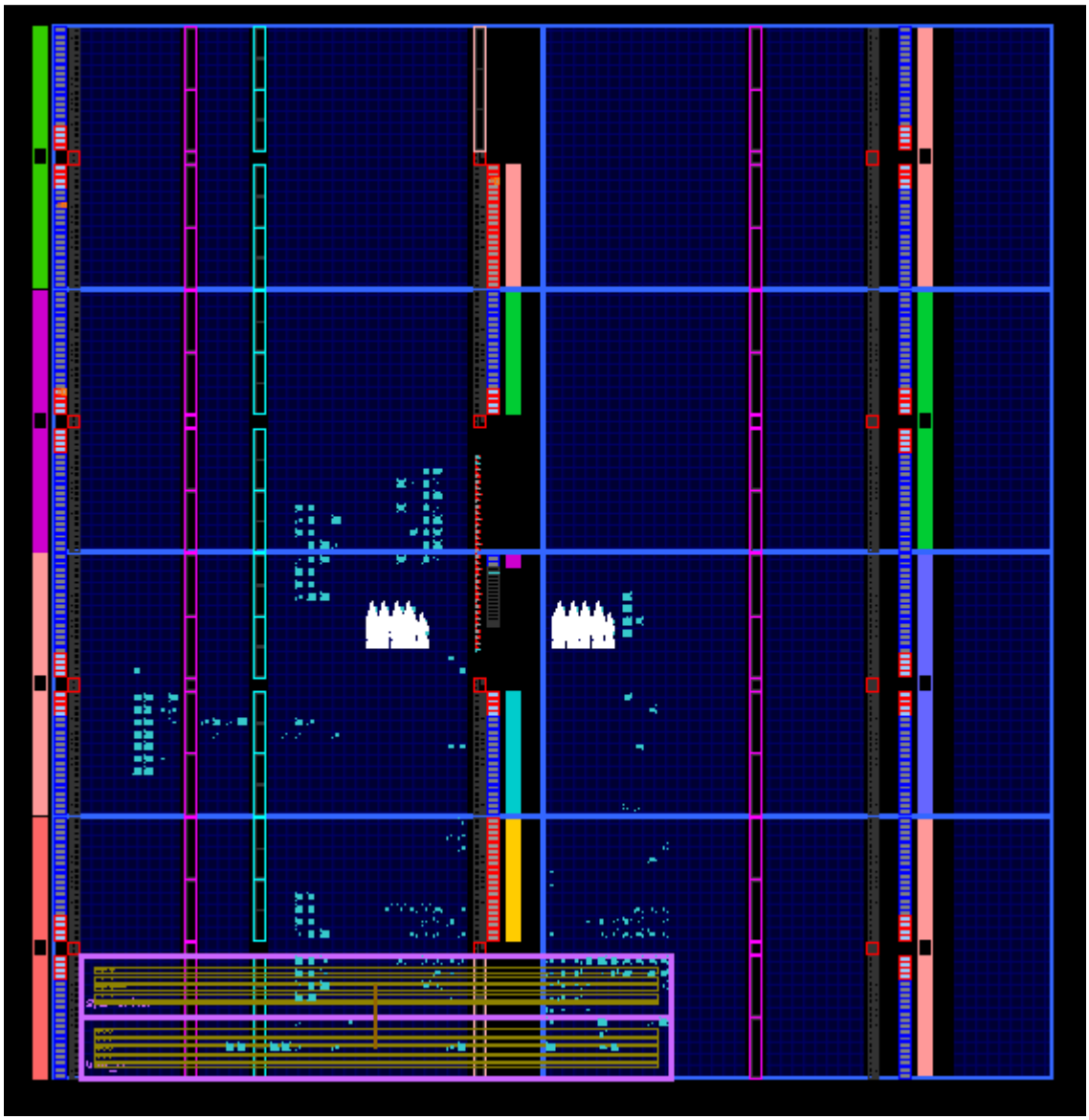

Figura 4-4: Colocación de los elementos en el segundo experimento.

A continuación se fue variando la temperatura desde los 40ºC hasta los 90ºC en pasos de 10ºC. Cuando se alcanzaba la estabilidad térmica se recogían 256 muestras tal y como se ha explicado en apartados anteriores por cada punto de temperatura. Finalmente se promediaron los 256 datos recibidos por cada temperatura y por cada sensor. El resultado que se observa es el siguiente:

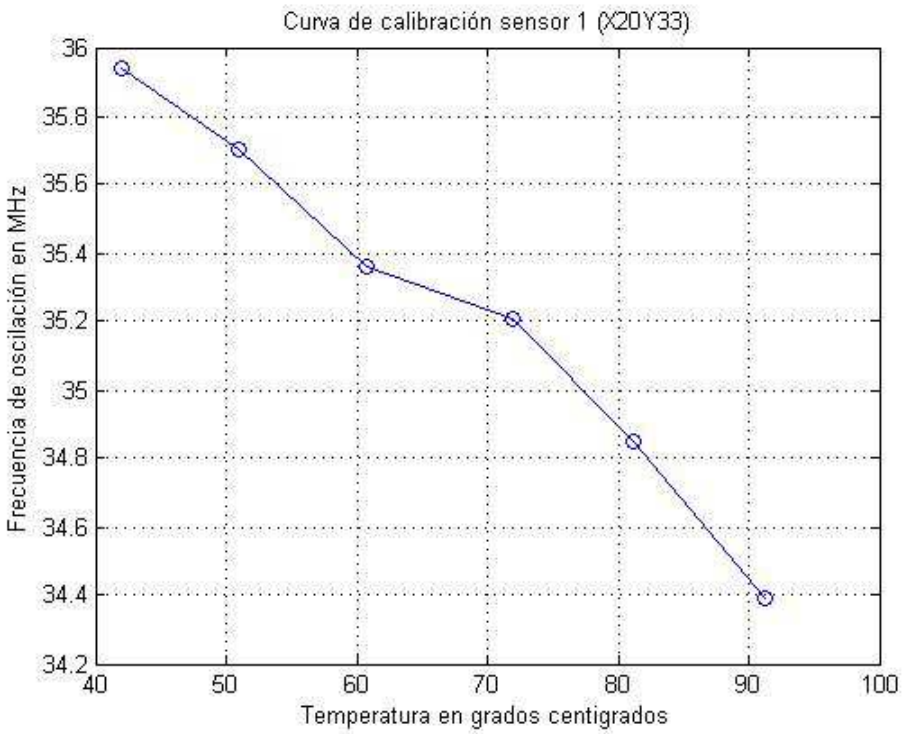

Figura 4-5: Curva de calibración del sensor 1 con voltaje constante de un 1V. El eje de la temperatura son los valores obtenidos del system monitor.

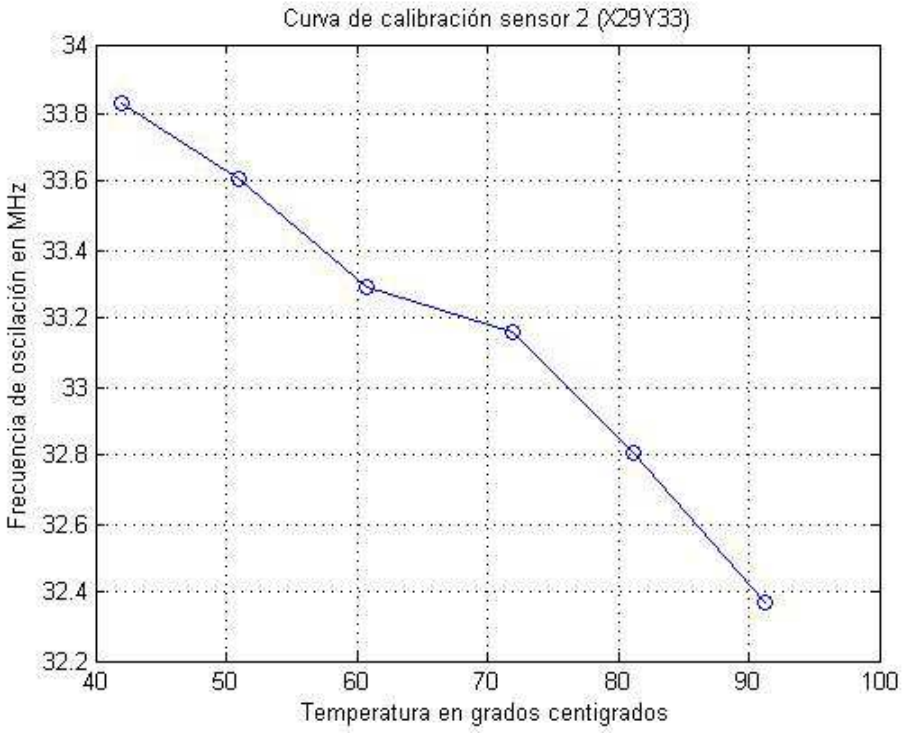

Figura 4-6: Curva de calibración del sensor 2 con voltaje constante de un 1V. El eje de la temperatura son los valores obtenidos del system monitor.

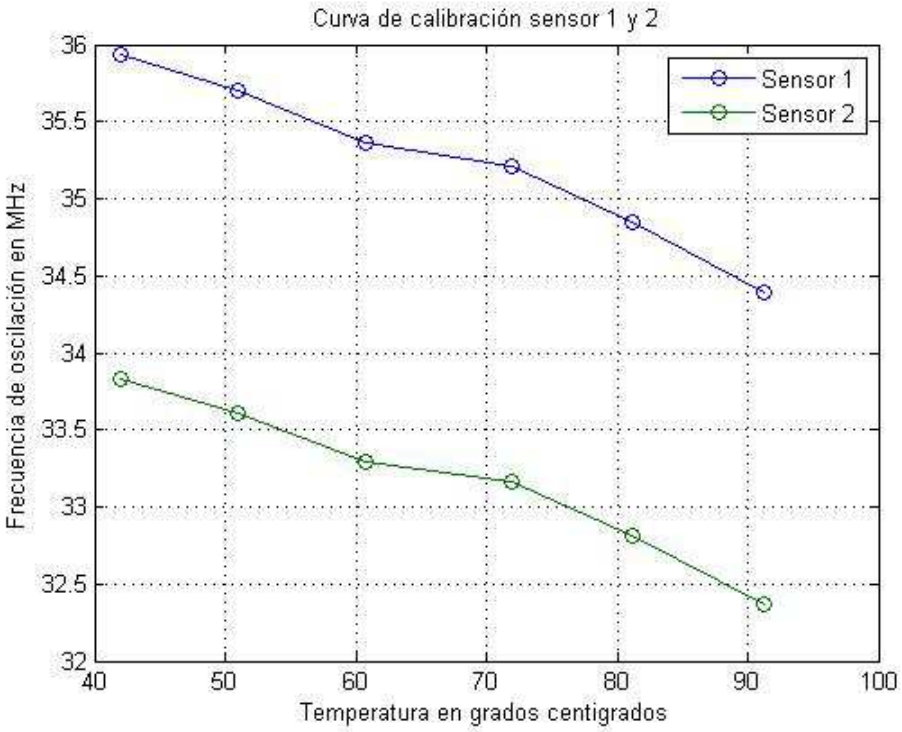

Figura 4-7: Comparación curva de calibración del sensor 1 y 2 con voltaje constante de un 1V. El eje de la temperatura son los valores obtenidos del system monitor.

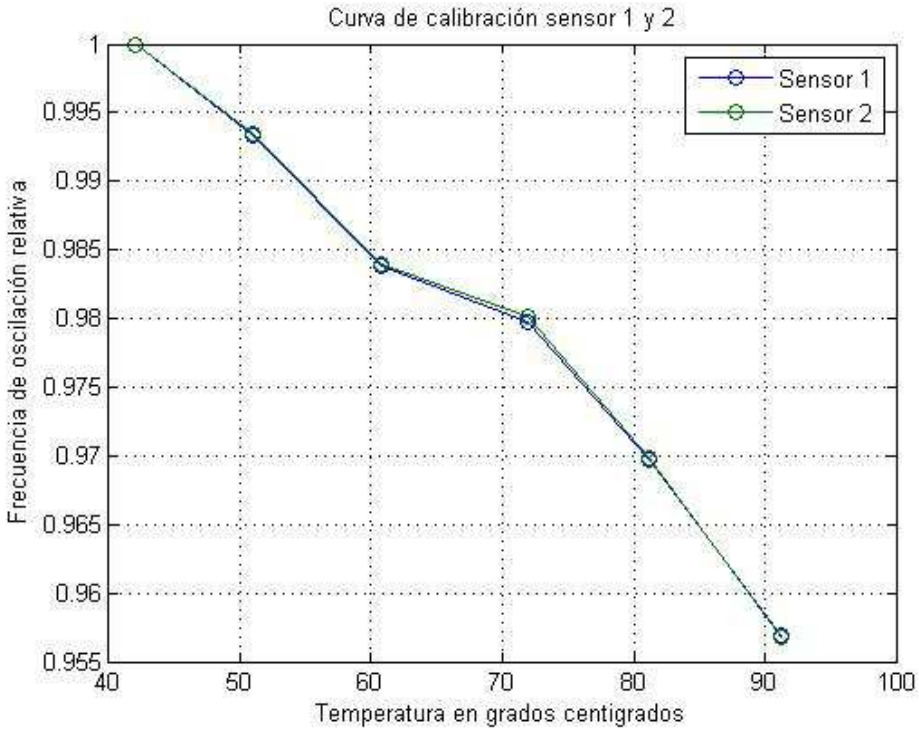

Figura 4-8: Comparación curva de calibración de la frecuencia relativa del sensor 1 y 2 con voltaje constante de un 1V. El eje de la temperatura son los valores obtenidos del system monitor.

Se puede observar que la temperatura de la medición obtenida con el System Monitor está un poco por encima de la del horno. Esto es debido a que se recogen los datos de la temperatura interna de la placa la cual es un poco más elevada.

Tras observar la Figura 4-8 se puede ver que la frecuencia relativa de ambos sensores es prácticamente la misma por lo que se puede decir que la variación de la frecuencia de oscilación es la misma aunque su frecuencia absoluta dependa de otros parámetros.

Como se ha visto en otros documentos [1] y [3], y tal y como se puede observar en la Figura 4-5 y Figura 4-6, la frecuencia de oscilación varía con la temperatura de una manera no lineal pero siempre se observa un descenso de la frecuencia según aumenta la temperatura.

También se observa que hay una diferencia entre las frecuencias obtenidas y la variación de la misma a iguales temperaturas de diferentes sensores, por lo que para poder obtener la temperatura exacta de cada sensor y cómo varía, habría que hacer una calibración para cada oscilador en su posición exacta ya que ésta también es muy importante.

A continuación y tras observar el efecto de la temperatura sobre la frecuencia en dos sensores diferentes, se procedió a variar el voltaje de núcleo, tal y como se ha indicado antes, desde 0.90 hasta 1.05 en pasos de 0.05 en un único sensor situado en la posición X20Y33. En cada punto de voltaje también se varió la temperatura nuevamente de 40 a 90ºC con pasos de 10 ºC. De cada punto se recogieron también 256 muestras y se promediaron. Se obtuvieron los siguientes resultados:

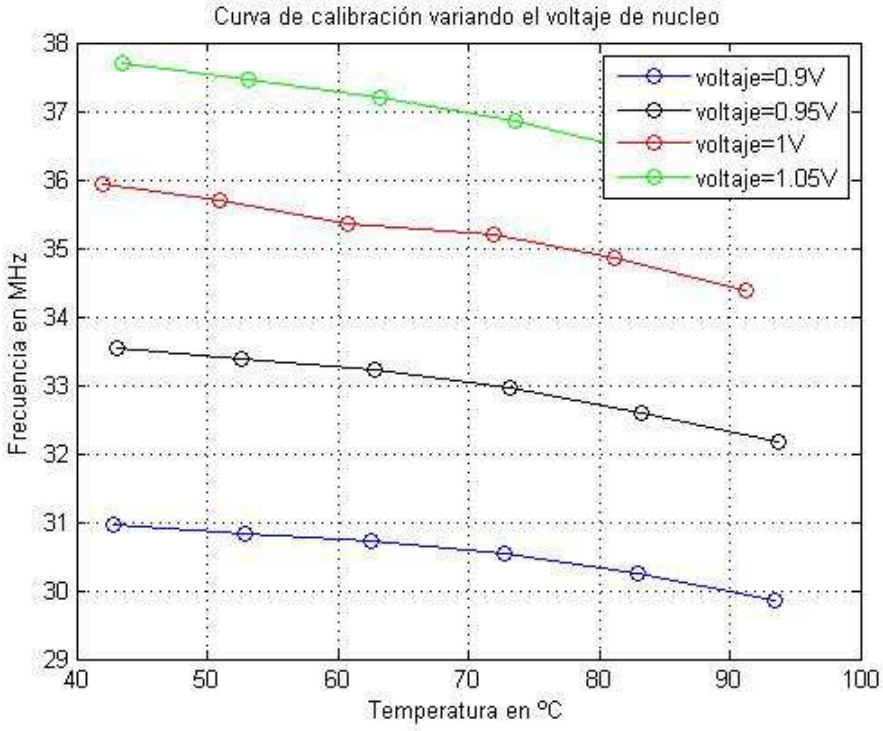

Figura 4-9: Curva de calibración del sensor 1 variando el voltaje de núcleo. Frecuencia en función de la temperatura. Perspectiva 1.

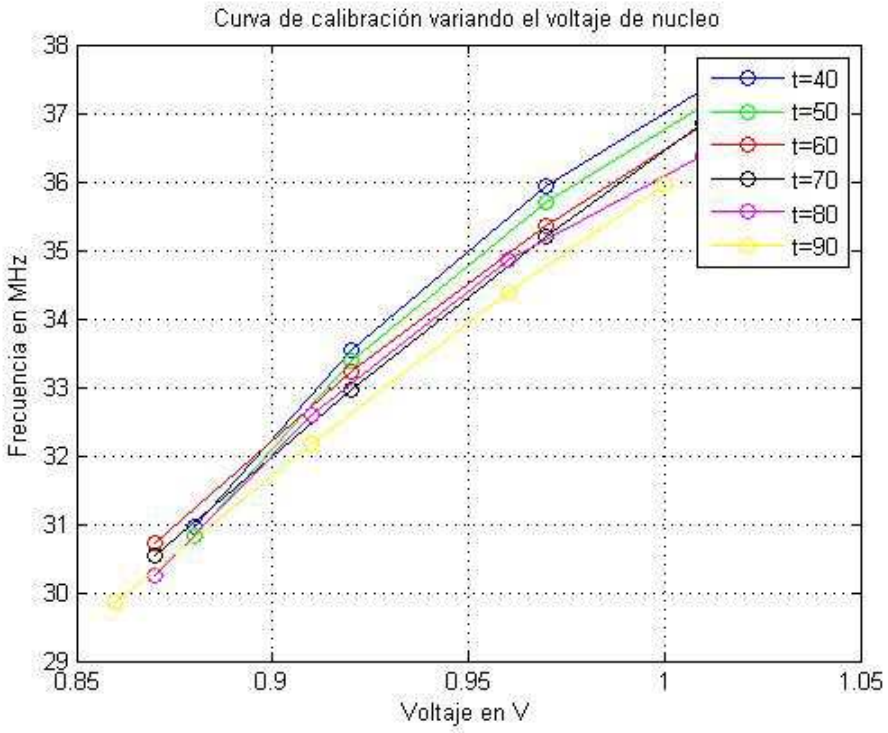

Figura 4-10: Curva de calibración del sensor 1 variando el voltaje de núcleo. Frecuencia en función del voltaje. Perspectiva 2.

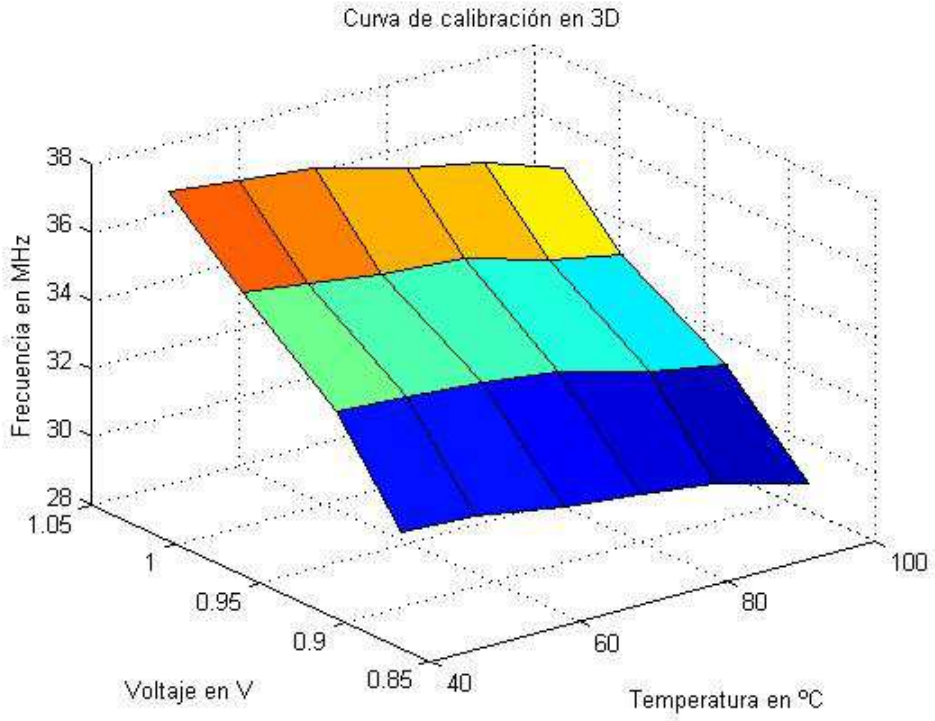

Figura 4-11: Curva de calibración en 3D del sensor 1. Perspectiva 3.

Según se observa arriba, la frecuencia de oscilación es más sensible al voltaje de núcleo que a la temperatura ya que una pequeña variación del primero consigue una variación mayor en la frecuencia que la temperatura.

Esto implica que hay que ser muy riguroso a la hora de alimentar el núcleo de la FPGA ya que una pequeña variación puede implicar un error muy grande en las medidas correspondientes e incluso en el funcionamiento de la misma. También se debe tener en cuenta la caída de tensión en la distribución de la alimentación, un aspecto difícil de conocer a nivel de usuario.

4.3 Tercer experimento

Para realizar este último experimento se han implementado los 48 osciladores anillo por toda la FPGA para poder obtener un mapa térmico en un futuro trabajo continuación de este PFC. Se han colocado formando 8 filas por 6 columnas y en la parte de inferior izquierda del DIE se han colocado los elementos de la UART y control del System Monitor tal y como se muestra en la Figura 4-12. Los datos se han recogido a una temperatura ambiente de 21 ºC y un voltaje de núcleo de 1V.

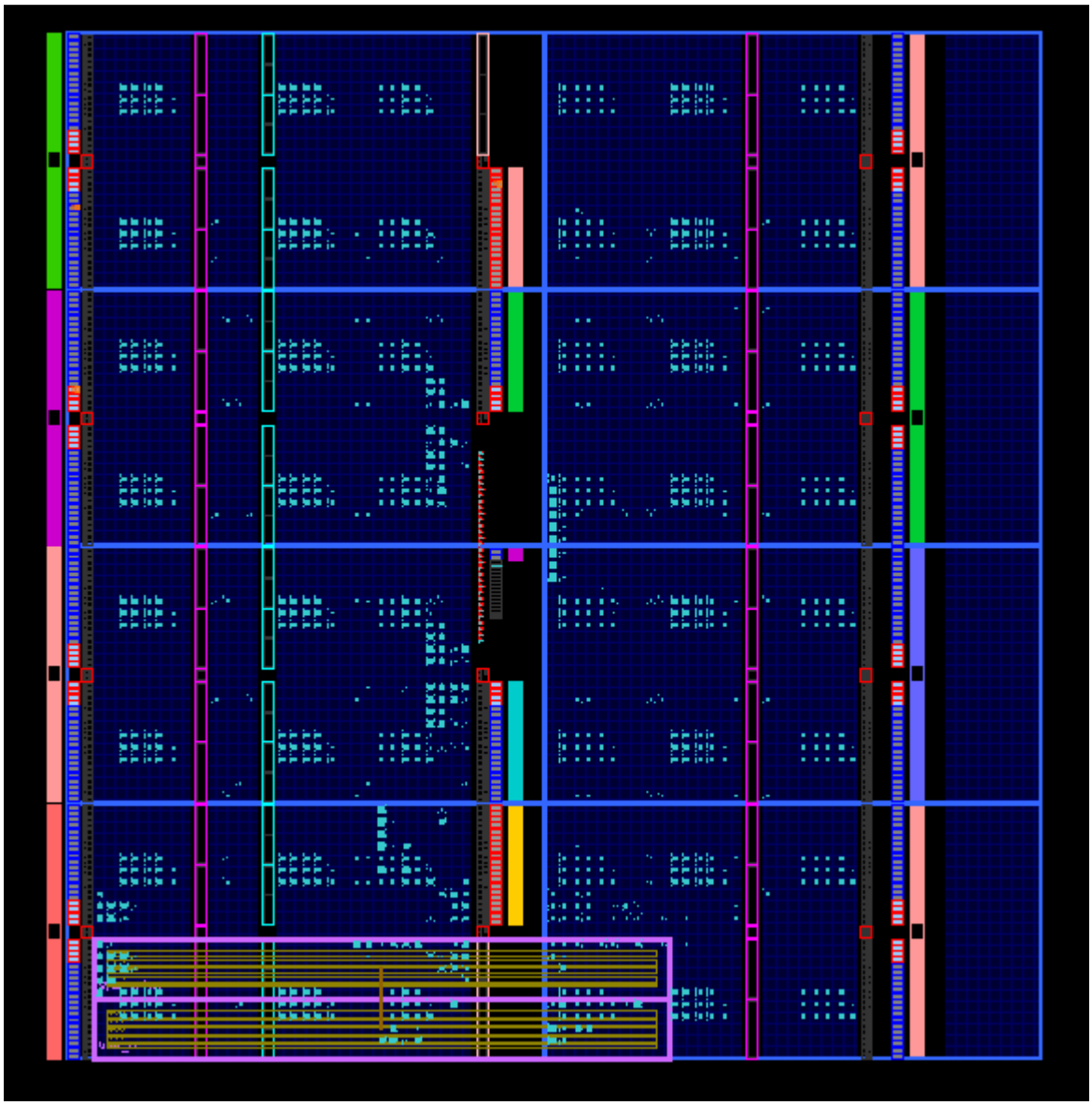

Figura 4-12: Ubicación de todos los elementos para el tercer experimento.

Con estos sensores se obtienen las diferentes frecuencias de oscilación en los diferentes puntos para los 48 sensores. Se han recogido 256 muestras en las que se puede observar que según va pasando el tiempo va subiendo la temperatura de la placa y por tanto va bajando la frecuencia de oscilación de los anillos, debido al funcionamiento de los mismos, tal y como se muestra en las siguientes figuras:

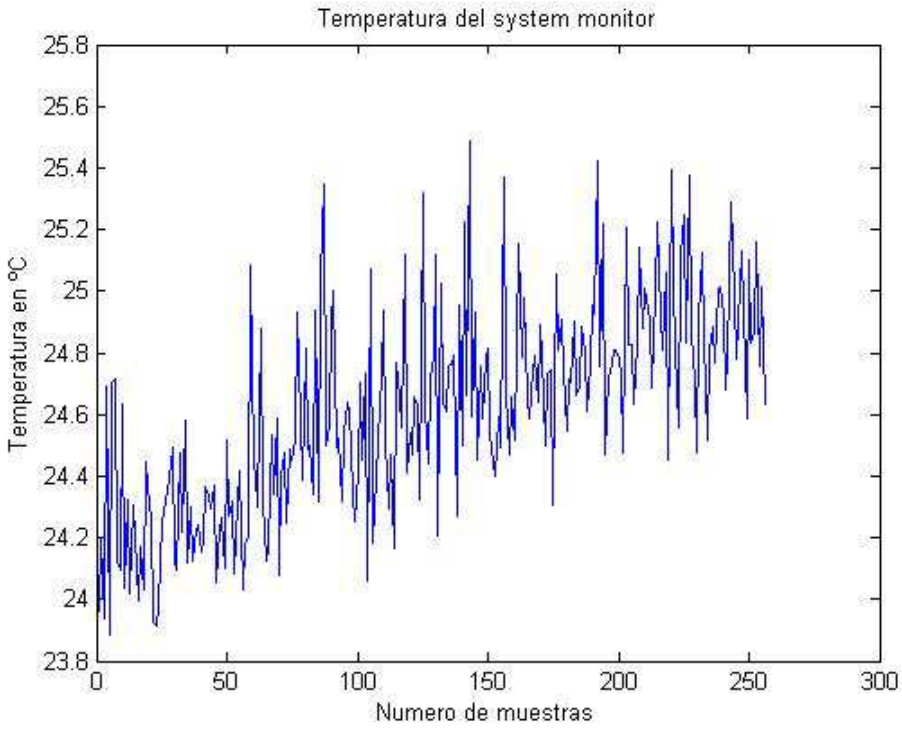

Figura 4-13: 256 muestras de la temperatura obtenida con el System Monitor mientras funcionan los sensores.

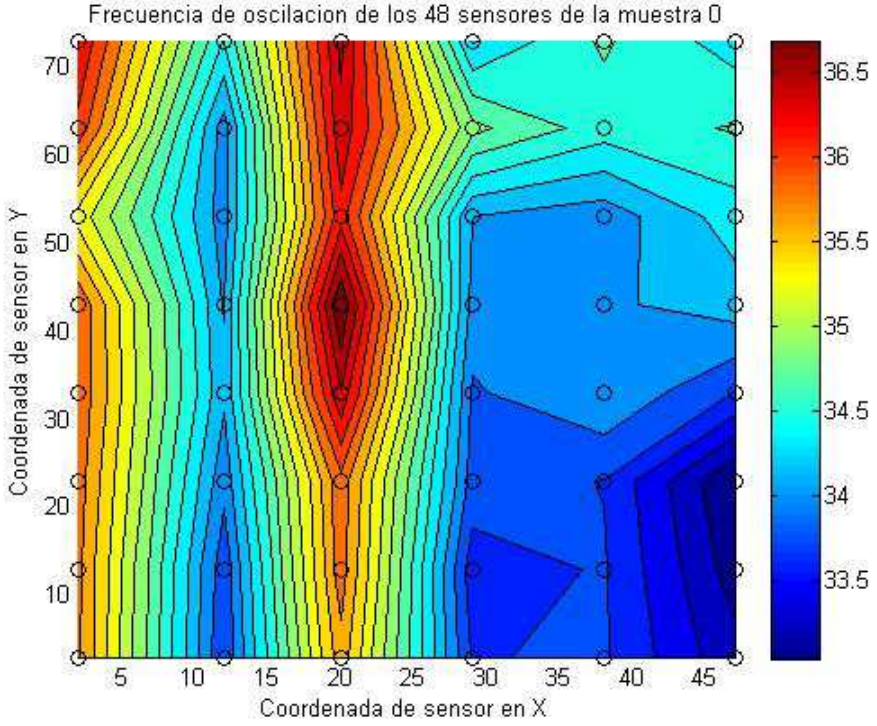

Figura 4-14: Frecuencia de oscilación de los 48 sensores de la primera muestra.

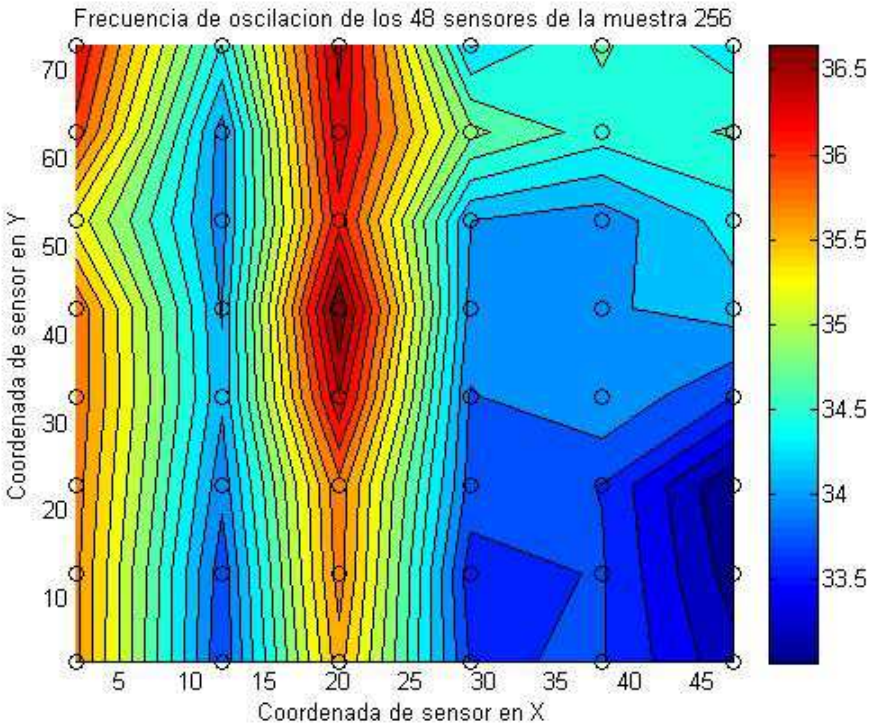

Figura 4-15: Frecuencia de oscilación de los 48 sensores de la 256ª muestra

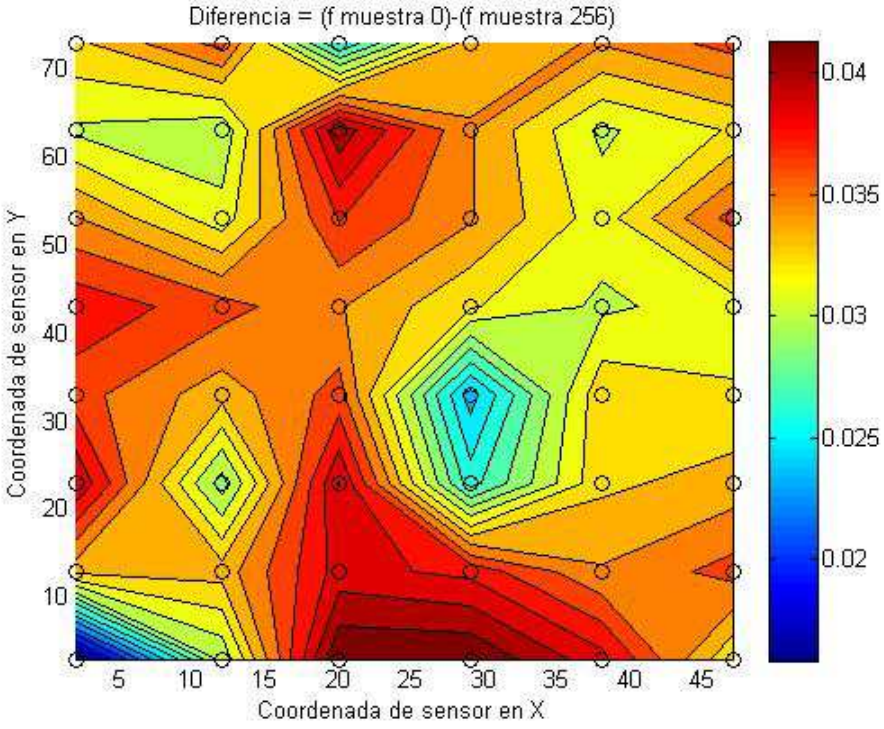

Figura 4-16: Diferencia de la frecuencia de oscilación en diferentes tiempos.

Las figuras anteriores representan dos cosas diferentes. La primera es la colocación de los sensores anillo que son los círculos negros y la segunda es la frecuencia de oscilación de los mismos en la Figura 4-14 y Figura 4-15 y la diferencia en la Figura 4-16. La Figura 4-13 es la variación de la temperatura del sensor del System Monitor según va pasando el tiempo en la recogida de muestras.

En la Figura 4-16 se puede observar que ha habido una variación en la frecuencia de oscilación por lo que ha habido una variación de temperatura. Pero para poder obtener la temperatura y cuanto ha variado concretamente en cada punto habría que realizar una calibración de cada uno de los sensores en su posición, tal y como se ha indicado durante el experimento 2.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Este proyecto ha consistido en diseñar, implementar y caracterizar osciladores anillo para sensar la temperatura de una FPGA de modelo Virtex-5 a través de la frecuencia de oscilación de los anillos. También se ha determinado los parámetros más importantes por los cuales varía dicha frecuencia. Y se ha llegado a las siguientes conclusiones:

- Para obtener un valor fiable del oscilador anillo hay que poner como mínimo 31 inversores ya que es el valor a partir del cual la desviación media estándar se estabiliza y no se obtienen valores significativamente diferentes. Para este proyecto se decidió poner 51 inversores ya que a partir de este número se comprobó que la desviación seguía disminuyendo de forma despreciable con respecto al incremento de elementos utilizados del FPGA (según el reporte de la síntesis)

- Es recomendable mantener un ruteado y localización fijos para todos los sensores anillo, ya que la variación de dichos parámetros hace que la frecuencia de oscilación puede ser muy diferente y variar de otra manera.

- Se observa que la frecuencia de oscilación de los sensores varía de forma inversamente proporcional a la temperatura de equilibrio de la FPGA. Por tal motivo y para que los resultados sean los más fiables posibles, es crucial que nuestro sistema entre en funcionamiento cuando el ambiente alcance el equilibrio térmico.

- La frecuencia de oscilación es muy sensible al voltaje de alimentación del núcleo. Tras realizar un cambio mínimo en dicho voltaje se puede observar una variación muy grande en la frecuencia. Por lo que es recomendable disponer de una fuente de alimentación muy precisa.

- Para poder realizar un mapa térmico completo con más de un sensor, hay que realizar la calibración de cada uno de ellos poniendo mucho cuidado en los parámetros indicados anteriormente ya que la más mínima variación en los mismos podría producir mapas térmicos incorrectos.

- Uno de los sensores diseñados reveló un coste en área del 1% LUTs. Lo que permitiría como máximo distribuir 100 sensores por toda la FPGA.

Finalmente comentar que los resultados obtenidos son coherentes con los modelos teóricos que definen el comportamiento de las FPGA's con la temperatura y gracias a ello se ha podido profundizar en el estudio de los siguientes conceptos:

- La tecnología de los sensores anillo.
- El funcionamiento de una FPGA y su lenguaje de programación, VHDL.
- El funcionamiento de los elementos externos como el osciloscopio y el horno.
- Realizar una interfaz de recogida de datos en Matlab a través de una UART.

5.2 Trabajo futuro

Los posibles trabajos futuros que se pueden realizar a partir de este proyecto son:

- Para que los datos obtenidos de la frecuencia de oscilación sean más exactos habría que hacer ruteado fijo para todas las señales de control que intervengan en el proceso. En este proyecto sólo se ha realizado este tipo de ruteado para las señales de los sensores que son las más sensibles.

- Para que no haya problemas de calentamiento innecesario en la zona donde están los sensores y por tanto afecte a la toma de datos, sería necesario ubicar todos los elementos que no pertenezcan a los sensores en una zona separada de los mismos e incluso en otra FPGA para que no interfiera.

- Para poder obtener un mapa térmico completo y poder observar cuales son las zonas calientes y como se van calentando, se debe hacer la calibración de cada uno de los 48 sensores. Tras este proceso se podrá ver cómo se calienta la placa.

- Con el fin de calentar la FPGA en una zona concreta, se podría implementar una aplicación que consuma muchos recursos y que funcione a máxima frecuencia de trabajo y por tanto caliente rápidamente la placa. Ésta se ubicaría en la zona que se quiera calentar y se obtendría el mapa térmico diseñado anteriormente.

- Otro experimento que se podría realizar es configurar el FPGA con un determinado número de una aplicación robusta en diferentes lugares de la placa y activarlas una por una. Las aplicaciones mencionadas son desactivadas solamente cuando superan un cierto umbral de variación de temperatura. Seguidamente se activa la siguiente aplicación con lo cual se podrían evitar puntos calientes muy elevados en varias zonas del FPGA.

- Debido a que los osciladores anillos presentan diferentes retardos en diferentes aplicaciones, se podría mejorar el diseño propuesto y crear un módulo con la finalidad de identificar físicamente funciones y evitar clonaciones.

Referencias

[1] John Jairo “Estudio sobre el comportamiento de sensores térmicos embarcados en FPGAs de 65 nm basados en osciladores anillo”, Tesis de Master, UAM.

[2] S. Lopez-Buedo; J. Garrido; E. Boemo."Dynamically Inserting, Operating, and Eliminating Thermal Sensors of FPGA-based Systems", IEEE Transactions on Components and Packaging Technologies (CPM), Vol.25, No4, pp.561-566, Diciembre 2002.

[3] Eduardo Boemo ; Sergio Lopez-Buedo. "Themal Monitoring on FPGAs using Ring-Oscillators", Lecture Notes in Computer Science, No.1304, pp.69-78, Berlin: Springer-Verlag, 1997.

[4] Sergio Lopez-Buedo; Javier Garrido; Eduardo Boemo. “Thermal Testing on Reconfigurable Computers”, IEEE Design & Test of Computers, vol.17, No.1, pp..84-91, Enero 2000.

[5] Kumar, R.; Kursun, V. “Reversed Temperature-Dependent Propagation Delay Characteristics in Nanometer CMOS Circuits”, Circuits and Systems II: Express Briefs, IEEE Transactions on. Volume 53, Issue 10, Oct. 2006 Page(s):1078 –1082.

[6] Xilinx, Inc., “Virtex-5 FPGA System Monitor User Guide.”.

[7] http://forums.xilinx.com/t5/Spartan-Family-FPGAs/Spartan-3-FPGA-and-Ethernet-Port-Hardware-Connection/td-p/164882

[8] http://www.ti.com/lit/ds/symlink/max232.pdf

[9] Xilinx Inc.Virtex-5 LX Prototype Platform User guide.

[10] http://application-notes.digchip.com/018/18-23898.pdf

[11] http://www.xilinx.com/support/sw_manuals/2_1i/download/fpedit.pdf

[12] http://www.xilinx.com/tools/planahead.htm

[13] Física para ciencias e ingenierías. Volumen 1. Raymond A. Serway y John W. Jewett Jr. Thomson.

[14] https://courses.cs.washington.edu/courses/cse467/03wi/FPGA.pdf

[15] http://www.xilinx.com/fpga/

[16] Jean-Pierre Deschamps; Gustavo D. Sutter; Enrique Cantó. “Guide to FPGA Implementation of Arithmetic Functions”, Springer.

[17] Física para la ciencia y la tecnología. Volumen 1. Tipler Mosca. Editorial Reverté 5º edición.

[18] SMSC Inc, EVB-EMC14XX user manual.

[19] Xilinx Inc. Virtex-5 Family Overview.

[20] Xilinx Inc. Virtex-5 Libraries Guide for VHDL design.

[21] Sergio Lopez-Buedo and Eduardo Boemo, "Making Visible the Thermal Behaviour of Embedded Microprocessors on FPGAs. A Progress Report", Proc. ACM FPGA 2004, Feb 22-24, Monterrey, USA. ACM Press: 2004.

[22] E. Boemo and S. Lopez-Buedo, "Thermal Verification on FPGAs", Invitated Talk, Proc. 23rd Norchip Conference, pp. 48-53, Oulu Finland, November 2005, IEEE Press, 2005.

[23] S. Lopez-Buedo, J. Garrido and E. Boemo, "Thermal Testing on Programmable Logic Devices", Proc. 1998 IEEE ISCAS (Int. Symp. on Circuits and Systems), Vol.II, pp.240-243, Monterey, June 1998.

[24] S. Lopez-Buedo and E. Boemo, "A Method for Temperature Measurement on Reconfigurable Systems", Proc. XII DCIS Conference (Design of Circuit and Integrated Systems), pp.727-730, Universidad de Sevilla: November 1997.

[25] Jamel Nebhem, StéphaneMeillère and Mohamed Masmoudi. “A Temperature Compensated CMOS Ring Oscillator for Wireless Sensing Applications”, New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International, pp. 37-40, Junio 2012.

Glosario

FPGA	Field Programmable Gate Array
CMOS	Complementary Metal Oxide Conductor
IOB	Input/Output Block
V5	Virtex-5
IR	Infrarrojos
E/S	Entrada/Salida
CLB	Configurable Logic Block
VHDL	Acrónimo que representa la combinación de VHSIC y HDL
VHSIC	Very High Speed Integrated Circuit
HDL	Hardware Description Language
CPLD	Complex Programmable Logic Device
FDC	Frequency to digital convert
TDC	Time to digital convert
UART	Universal Asynchronous Receiver-Transmitter
ADC	Analog-to-Digital Conversion
LUT	Look-Up Table
PFC	Proyecto Fin de Carrera

Anexos

A Anexo A

Código del sensor para 51 inversores:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
useUNISIM.VComponents.all;

entity sensor0 is
generic ( XPosSensor:integer:=0;  YPosSensor:integer:=0);
PORT ( RING_EN : IN  STD_LOGIC;
                CLK_RING : OUT  STD_LOGIC
 );
attribute RLOC_ORIGIN: string;
ATTRIBUTE RLOC_ORIGIN of sensor0: entity is "X" &integer'image(XPosSensor)& "Y"
      &integer'image(YPosSensor); --"X0Y0";
end sensor0;

architecture structural of sensor0 is

signal OUT_AND_ENABLE: std_logic;
signal ring1: std_logic;
signal ring2: std_logic;
signal ring3: std_logic;
signal ring4: std_logic;
signal ring5: std_logic;
signal ring6: std_logic;
signal ring7: std_logic;
signal ring8: std_logic;
signal ring9: std_logic;
signal ring10: std_logic;
signal ring11: std_logic;
signal ring12: std_logic;
signal ring13: std_logic;
signal ring14: std_logic;
signal ring15: std_logic;
signal ring16: std_logic;
signal ring17: std_logic;
signal ring18: std_logic;
signal ring19: std_logic;
signal ring20: std_logic;
signal ring21: std_logic;
signal ring22: std_logic;
signal ring23: std_logic;
signal ring24: std_logic;
signal ring25: std_logic;
signal ring26: std_logic;
signal ring27: std_logic;
```

```
signal ring28: std_logic;
signal ring29: std_logic;
signal ring30: std_logic;
signal ring31: std_logic;
signal ring32: std_logic;
signal ring33: std_logic;
signal ring34: std_logic;
signal ring35: std_logic;
signal ring36: std_logic;
signal ring37: std_logic;
signal ring38: std_logic;
signal ring39: std_logic;
signal ring40: std_logic;
signal ring41: std_logic;
signal ring42: std_logic;
signal ring43: std_logic;
signal ring44: std_logic;
signal ring45: std_logic;
signal ring46: std_logic;
signal ring47: std_logic;
signal ring48: std_logic;
signal ring49: std_logic;
signal ring50: std_logic;
signal ring51: std_logic;

attribute keep: string;
attribute keep of ring1: signal is "TRUE";
attribute keep of ring2: signal is "TRUE";
attribute keep of ring3: signal is "TRUE";
attribute keep of ring4: signal is "TRUE";
attribute keep of ring5: signal is "TRUE";
attribute keep of ring6: signal is "TRUE";
attribute keep of ring7: signal is "TRUE";
attribute keep of ring8: signal is "TRUE";
attribute keep of ring9: signal is "TRUE";
attribute keep of ring10: signal is "TRUE";
attribute keep of ring11: signal is "TRUE";
attribute keep of ring12: signal is "TRUE";
attribute keep of ring13: signal is "TRUE";
attribute keep of ring14: signal is "TRUE";
attribute keep of ring15: signal is "TRUE";
attribute keep of ring16: signal is "TRUE";
attribute keep of ring17: signal is "TRUE";
attribute keep of ring18: signal is "TRUE";
attribute keep of ring19: signal is "TRUE";
attribute keep of ring20: signal is "TRUE";
attribute keep of ring21: signal is "TRUE";
attribute keep of ring22: signal is "TRUE";
attribute keep of ring23: signal is "TRUE";
attribute keep of ring24: signal is "TRUE";
attribute keep of ring25: signal is "TRUE";
attribute keep of ring26: signal is "TRUE";
attribute keep of ring27: signal is "TRUE";
attribute keep of ring28: signal is "TRUE";
attribute keep of ring29: signal is "TRUE";
attribute keep of ring30: signal is "TRUE";
attribute keep of ring31: signal is "TRUE";
```

```
attribute keep of ring32: signal is "TRUE";
attribute keep of ring33: signal is "TRUE";
attribute keep of ring34: signal is "TRUE";
attribute keep of ring35: signal is "TRUE";
attribute keep of ring36: signal is "TRUE";
attribute keep of ring37: signal is "TRUE";
attribute keep of ring38: signal is "TRUE";
attribute keep of ring39: signal is "TRUE";
attribute keep of ring40: signal is "TRUE";
attribute keep of ring41: signal is "TRUE";
attribute keep of ring42: signal is "TRUE";
attribute keep of ring43: signal is "TRUE";
attribute keep of ring44: signal is "TRUE";
attribute keep of ring45: signal is "TRUE";
attribute keep of ring46: signal is "TRUE";
attribute keep of ring47: signal is "TRUE";
attribute keep of ring48: signal is "TRUE";
attribute keep of ring49: signal is "TRUE";
attribute keep of ring50: signal is "TRUE";
attribute keep of ring51: signal is "TRUE";
--attribute keep of ring_enable: signal is "TRUE";
attribute keep of OUT_AND_ENABLE: signal is "TRUE";

attribute RLOC: string;
attribute RLOC of INV1:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV2:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV3:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV4:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV5:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV6:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV7:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV8:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV9:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV10:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV11:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV12:label is "X" &integer'image(XPosSensor)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV13:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV14:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV15:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV16:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV17:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV18:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV19:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV20:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV21:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV22:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV23:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV24:label is "X" &integer'image(XPosSensor + 1)& "Y" &integer'image(YPosSensor + 2);

attribute RLOC of INV25:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV26:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV27:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor);

attribute RLOC of INV28:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV29:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV30:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV31:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 1);
```

```
attribute RLOC of INV32:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV33:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV34:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV35:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV36:label is "X" &integer'image(XPosSensor + 2)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV37:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV38:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV39:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV40:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV41:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV42:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV43:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV44:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 1);
attribute RLOC of INV45:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV46:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV47:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV48:label is "X" &integer'image(XPosSensor + 3)& "Y" &integer'image(YPosSensor + 2);
attribute RLOC of INV49:label is "X" &integer'image(XPosSensor + 4)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV50:label is "X" &integer'image(XPosSensor + 4)& "Y" &integer'image(YPosSensor);
attribute RLOC of INV51:label is "X" &integer'image(XPosSensor + 4)& "Y" &integer'image(YPosSensor);
attribute RLOC of AND_ENABLE: label is "X" &integer'image(XPosSensor + 4)& "Y"
    &integer'image(YPosSensor);
attribute RLOC of BUFF: label is "X" &integer'image(XPosSensor + 4)& "Y" &integer'image(YPosSensor + 1);

begin

-----------------------------------------------------------------
---------------------- RING OSCILLATOR ---------------------------
-----------------------------------------------------------------
INV1 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring1,                                  -- LUT general output
            I0 => OUT_AND_ENABLE                         -- LUT input);
INV2 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring2,                                  -- LUT general output
            I0 => ring1                                  -- LUT input);
INV3 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring3,                                  -- LUT general output
            I0 => ring2                                  -- LUT input);
INV4 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring4,                                  -- LUT general output
            I0 => ring3                                  -- LUT input);
INV5 : LUT1
    generic map (
            INIT => "01")
            port map (
```

```
                O => ring5,                     -- LUT general output
                I0 => ring4                     -- LUT input);
INV6 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring6,                     -- LUT general output
                I0 => ring5                     -- LUT input);
INV7 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring7,                     -- LUT general output
                I0 => ring6                     -- LUT input);
INV8 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring8,                     -- LUT general output
                I0 => ring7                     -- LUT input);
INV9 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring9,                     -- LUT general output
                I0 => ring8                     -- LUT input);
INV10 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring10,                    -- LUT general output
                I0 => ring9                     -- LUT input);
INV11 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring11,                    -- LUT general output
                I0 => ring10                    -- LUT input);
INV12 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring12,                    -- LUT general output
                I0 => ring11                    -- LUT input);
INV13 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring13,                    -- LUT general output
                I0 => ring12                    -- LUT input);
INV14 : LUT1
    generic map (
                INIT => "01")
                port map (
                O => ring14,                    -- LUT general output
                I0 => ring13                    -- LUT input);
INV15 : LUT1
```

```
    generic map (
            INIT => "01")
            port map (
            O => ring15,                    -- LUT general output
            I0 => ring14                    -- LUT input);
INV16 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring16,                    -- LUT general output
            I0 => ring15                    -- LUT input);
INV17 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring17,                    -- LUT general output
            I0 => ring16                    -- LUT input);
INV18 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring18,                    -- LUT general output
            I0 => ring17                    -- LUT input);
INV19 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring19,                    -- LUT general output
            I0 => ring18                    -- LUT input);
INV20 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring20,                    -- LUT general output
            I0 => ring19                    -- LUT input);
INV21 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring21,                    -- LUT general output
            I0 => ring20                    -- LUT input);
INV22 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring22,                    -- LUT general output
            I0 => ring21                    -- LUT input);
INV23 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring23,                    -- LUT general output
            I0 => ring22                    -- LUT input);
INV24 : LUT1
    generic map (
            INIT => "01")
            port map (
```

```
            O => ring24,                    -- LUT general output
            I0 => ring23                    -- LUT input);
INV25 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring25,                    -- LUT general output
            I0 => ring24                    -- LUT input);
INV26 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring26,                    -- LUT general output
            I0 => ring25                    -- LUT input);
INV27 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring27,                    -- LUT general output
            I0 => ring26                    -- LUT input);
INV28 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring28,                    -- LUT general output
            I0 => ring27                    -- LUT input);
INV29 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring29,                    -- LUT general output
            I0 => ring28                    -- LUT input);
INV30 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring30,                    -- LUT general output
            I0 => ring29                    -- LUT input);
INV31 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring31,                    -- LUT general output
            I0 => ring30                    -- LUT input);
INV32 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring32,                    -- LUT general output
            I0 => ring31                    -- LUT input);
INV33 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring33,                    -- LUT general output
            I0 => ring32                    -- LUT input);
INV34 : LUT1
```

```
    generic map (
            INIT => "01")
            port map (
            O => ring34,                    -- LUT general output
            I0 => ring33                    -- LUT input);
INV35 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring35,                    -- LUT general output
            I0 => ring34                    -- LUT input);
INV36 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring36,                    -- LUT general output
            I0 => ring35                    -- LUT input);
INV37 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring37,                    -- LUT general output
            I0 => ring36                    -- LUT input);
INV38 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring38,                    -- LUT general output
            I0 => ring37                    -- LUT input);
INV39 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring39,                    -- LUT general output
            I0 => ring38                    -- LUT input);
INV40 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring40,                    -- LUT general output
            I0 => ring39                    -- LUT input);
INV41 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring41,                    -- LUT general output
            I0 => ring40                    -- LUT input);
INV42 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring42,                    -- LUT general output
            I0 => ring41                    -- LUT input);
INV43 : LUT1
    generic map (
            INIT => "01")
            port map (
```

```
            O => ring43,                    -- LUT general output
            I0 => ring42                    -- LUT input);
INV44 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring44,                     -- LUT general output
            I0 => ring43                    -- LUT input);
INV45 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring45,                     -- LUT general output
            I0 => ring44                    -- LUT input);
INV46 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring46,                     -- LUT general output
            I0 => ring45                    -- LUT input);
INV47 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring47,                    -- LUT general output
            I0 => ring46                     -- LUT input);
INV48 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring48,                     -- LUT general output
            I0 => ring47                     -- LUT input);
INV49 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring49,                     -- LUT general output
            I0 => ring48                    -- LUT input);
INV50 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring50,                    -- LUT general output
            I0 => ring49                    -- LUT input);
INV51 : LUT1
    generic map (
            INIT => "01")
            port map (
            O => ring51,                    -- LUT general output
            I0 => ring50                    -- LUT input);
AND_ENABLE: LUT2
    generic map (
            INIT => X"8")
    port map (
            O => OUT_AND_ENABLE,            -- LUT general output
            I0 => RING_EN,                  -- LUT input
            I1 => ring51                    -- LUT input);
```

```
BUFF : LUT1
      generic map (
                  INIT => "10")
                  port map (
                  O =>clk_ring,                          -- LUT general output
                  I0 => OUT_AND_ENABLE                   -- LUT input);
endstructural;
```

B Anexo B

Código para la lógica de control, el UART y el System Monitor. También se instancia la red de sensores:

```
libraryieee;
use ieee.std_logic_1164.all;
useieee.std_logic_arith.all;
useieee.std_logic_unsigned.all;
libraryunisim;
useunisim.vcomponents.all;

entitycontrolador is
	port(
		clk: in std_logic;
		reset: in std_logic;
		tx: out std_logic
		 );
attributerloc_origin: string;
attributerloc_origin of controlador: entity is "x0y0";
endcontrolador;

architecture behavioral of controlador is

component sensor0
generic ( xpossensor:integer:=0;  ypossensor:integer:=0);
port ( ring_en : in  std_logic;
clk_ring : out  std_logic
		 );
end component;

componentuart_tx
generic(	dbit: natural:=8; -- # data bits
	sb_tick: natural:=16;  -- # ticks for stop bits
	sys_clk: natural:=50e6;
	baud_rate: natural:=19200);
port(clk, reset: in std_logic;
	tx_start: in std_logic;
	din: in std_logic_vector(dbit-1 downto 0);
	tx_done_tick: out std_logic;
	tx: out std_logic );
end component;

componentsysmonitor
port(DCLK		: in  STD_LOGIC;
	RESET		: in  STD_LOGIC;
	MEASURED_TEMP	: out  STD_LOGIC_VECTOR (15 downto 0);
	MEASURED_VCCINT	: out  STD_LOGIC_VECTOR (15 downto 0));
end component;

constantNum_sensors: natural:= 48;
constantdelay_initial: natural:= 16;
constanttime_enable: natural:= 30000;
constantsys_clk: natural:= 25e6;
constanttotal_samples: natural:= 255;
```

```
signalstate_uart: std_logic_vector(4 downto 0):= "00000";
signalstate_ctrl: std_logic_vector(3 downto 0):= "0000";
signalstate_ring: std_logic_vector(2 downto 0):= "000";

typeposiciones is array (Num_sensors-1 downto 0) of integer;
constantposiciones_x : posiciones:= (2, 12, 20, 29, 38, 47, 2, 12, 20, 29, 38, 47, 2, 12, 20, 29, 38, 47, 2, 12, 20, 29,
      38, 47, 2, 12, 20, 29, 38, 47, 2, 12, 20, 29, 38, 47, 2, 12, 20, 29, 38, 47, 2, 12, 20, 29, 38, 47);
constantposiciones_y : posiciones:= (3, 3, 3, 3, 3, 3, 13, 13, 13, 13, 13, 13, 23, 23, 23, 23, 23, 23, 33, 33, 33, 33, 33,
      33, 43, 43, 43, 43, 43, 43, 53, 53, 53, 53, 53, 53, 63, 63, 63, 63, 63, 63, 73, 73, 73, 73, 73, 73);

signalring_enable, clk_anillo: std_logic_vector(0 to Num_sensors-1):=(others=>'0');
signalcount_ring_reg, count_ring: std_logic_vector(15 downto 0);
signal din, byte0_count, byte1_count, byte0_temp, byte1_temp, byte0_voltaje, byte1_voltaje: std_logic_vector(7
      downto 0);
signaltx_start, tx_done, tx_total, start_count, clk_ring, en_uart: std_logic;
signaltemp, voltaje : std_logic_vector(15 downto 0);
signalnum_samples, nsensor, cnt, vuelta: natural;

begin

--    proceso de control principal
control: process(clk, reset)
begin
                if reset='0' then
                state_ctrl<= "0000";
                ring_enable<= (others=>'0');
                num_samples<= 0;
                cnt<=0;
                nsensor<= 0;
      elsifrising_edge(clk) then
                casestate_ctrl is
                        when "0000" =>
                                state_ctrl<= "0001";
                                ifnsensor = Num_sensors then
                                        num_samples<= num_samples +1;
                                        nsensor<= 0;
                                        ifnum_samples = total_samples then
                                                state_ctrl<= "1101";
                                        end if;
                                end if;
                        when "0001" =>
                                ring_enable(nsensor) <= '1';
                                state_ctrl<= "0010";
                        when "0010" =>
                                cnt<= cnt+1;
                                ifcnt = delay_initial then
                                        state_ctrl<= "0011";
                                        start_count<= '1';
                                        cnt<=0;
                                end if;
                        when "0011" =>
                                cnt<= cnt+1;
                                ifcnt = time_enable then
                                        state_ctrl<= "0100";
                                        start_count<= '0';
                                        cnt<= 0;
```

```
                    end if;
                when "0100" =>
                    cnt<= cnt+1;
                    ifcnt = time_capt then
                        state_ctrl<= "0101";
                        en_uart<= '1';
                    end if;
                when "0101" =>
                    en_uart<= '0';
                    state_ctrl<= "0110";
                    ring_enable(nsensor) <= '0';
                    nsensor<= nsensor + 1;
                when "0110" =>
                    iftx_total = '1' then
                        state_ctrl<= "0000";
                    end if;
                when others =>

            end case;
        end if;
end process;

clk_ring<= clk_anillo(nsensor);

cont_ring_osci: process(clk_ring, start_count)
begin
        ifstart_count='0' then
            count_ring<= x"0000";
        elsifrising_edge(clk_ring) then
            count_ring<= count_ring + 1;
            count_ring_reg<= count_ring;
        end if;
end process;

g1:fori in 0 to (Num_sensors-1) generate
begin
        inst_sensor: sensor0 generic map(xpossensor =>posiciones_x(i), ypossensor =>posiciones_y(i))
        port map( clk_ring =>clk_anillo(i), ring_en =>ring_enable(i));
end generate;

uart_inst: uart_tx generic map(dbit => 8, sb_tick =>16, sys_clk =>sys_clk, baud_rate => 19200)
        port map( clk =>clk, reset =>reset,tx_start =>tx_start, din => din, tx_done_tick =>tx_done, tx =>tx);

sysmon_inst: sysmonitor
        port map (DCLK =>clk, RESET => reset, MEASURED_TEMP => temp, MEASURED_VCCINT =>voltaje);

tx_uart: process(clk, reset, state_ring)
        begin
            if reset='0' then
                tx_start<= '0';
                 byte0_count <=(others=>'0');
                 byte1_count <=(others=>'0');
                 byte0_temp <=(others=>'0');
                 byte1_temp <=(others=>'0');
                 byte0_voltaje <=(others=>'0');
                 byte1_voltaje <=(others=>'0');
                state_uart<= "00000";
```

```
		tx_total<= '0';
		vuelta<= 0;
elsifrising_edge(clk) then
		casestate_uart is
			when "00000" =>
				tx_total<= '0';
				ifen_uart = '1' then
					state_uart<= "00001";
					 byte0_count <= count_ring_reg(7 downto 0);
					 byte1_count <= count_ring_reg(15 downto 8);
					 byte0_temp <= temp(7 downto 0);
					 byte1_temp <= temp(15 downto 8);
					byte0_voltaje <= voltaje(7 downto 0);
					 byte1_voltaje <= voltaje(15 downto 8);
				end if;
			when "00001"	=>
				din<= byte1_count;
				state_uart<= "00010";
			when "00010" =>
				tx_start<= '1';
				state_uart<= "00011";
			when "00011" =>
				tx_start<= '0';
				iftx_done = '1' then
					state_uart<= "00100";
				end if;
			when "00100"	=>
				din<= byte0_count;
				state_uart<= "00101";
			when "00101" =>
				tx_start<= '1';
				state_uart<= "00110";
			when "00110" =>
				tx_start<= '0';
				iftx_done = '1' then
					ifvuelta = 47 then
						state_uart<= "00111";
						vuelta<= 0;
					else
						state_uart<= "00000";
						vuelta<= vuelta + 1;
						tx_total<= '1';
					end if;
				end if;
			when "00111"	=>
				din<= byte1_temp;
				state_uart<= "01000";
			when "01000" =>
				tx_start<= '1';
				state_uart<= "01001";
			when "01001" =>
				tx_start<= '0';
				iftx_done = '1' then
					state_uart<= "01010";
				end if;
			when "01010"	=>
				din<= byte0_temp;
```

```
                        state_uart<= "01011";
                when "01011" =>
                        tx_start<= '1';
                        state_uart<= "01100";
                when "01100" =>
                        tx_start<= '0';
                        iftx_done = '1' then
                                state_uart<= "01101";
                        end if;
                when "01101"        =>
                        din<= byte1_voltaje;
                        state_uart<= "01110";
                when "01110" =>
                        tx_start<= '1';
                        state_uart<= "01111";
                when "01111" =>
                        tx_start<= '0';
                        iftx_done = '1' then
                                state_uart<= "10000";
                        end if;
                when "10000"        =>
                        din<= byte0_voltaje;
                        state_uart<= "10001";
                when "10001" =>
                        tx_start<= '1';
                        state_uart<= "10010";
                when others =>
                        tx_start<= '0';
                        iftx_done = '1' then
                                state_uart<= "00000";
                                tx_total<= '1';
                        end if;
        end case;
    end if;

  end process;
endbehavioral;
```

C Anexo C

Código para el envío de la UART

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
---USE WORK.MY_PACKAGE.ALL;
LIBRARY UNISIM;
USE UNISIM.VCOMPONENTS.ALL;
ENTITY UART_TX IS
GENERIC( DBIT: NATURAL:=8; -- # DATA BITS
                    SB_TICK: NATURAL:=16;  -- # TICKS FOR STOP BITS
                    SYS_CLK: NATURAL:=50E6;
                    BAUD_RATE: NATURAL:=19200);
PORT(  CLK, RESET: IN STD_LOGIC;
                  TX_START: IN STD_LOGIC;
                  DIN: IN STD_LOGIC_VECTOR(DBIT-1 DOWNTO 0);
                  TX_DONE_TICK: OUT STD_LOGIC;
                  TX: OUT STD_LOGIC );
END UART_TX ;

ARCHITECTURE ARCH OF UART_TX IS
  CONSTANT DVSR: NATURAL:= (SYS_CLK/(SB_TICK*BAUD_RATE));
  TYPE STATE_TYPE IS (IDLE, START, DATA, STOP);
  SIGNAL STATE_REG: STATE_TYPE;
  SIGNAL S_REG: STD_LOGIC_VECTOR(3 DOWNTO 0);
  SIGNAL N_REG: STD_LOGIC_VECTOR(2 DOWNTO 0);----2 = DBIT =8,3 DBIT=16
  SIGNAL B_REG: STD_LOGIC_VECTOR(DBIT-1 DOWNTO 0);
  SIGNAL TX_REG: STD_LOGIC;
  SIGNAL S_TICK: STD_LOGIC;
BEGIN
PROCESS(CLK,RESET)
  BEGIN
    IF RESET='0' THEN
                  STATE_REG <= IDLE;
                  S_REG <= (OTHERS=>'0');
                  N_REG <= (OTHERS=>'0');
                  B_REG <= (OTHERS=>'0');
                  TX_REG <= '1';
                  TX_DONE_TICK <= '0';
   ELSIF RISING_EDGE(CLK)  THEN
                  CASE STATE_REG IS
                        WHEN IDLE =>
                              TX_REG <= '1';
                              TX_DONE_TICK <= '0';
                              IF TX_START='1' THEN
                                    STATE_REG <= START;
                                    S_REG <= (OTHERS=>'0');
                                    B_REG <= DIN;
                              END IF;
                        WHEN START =>
                              TX_REG <= '0';
                              IF (S_TICK = '1') THEN
                                    IF S_REG=(SB_TICK-1) THEN
                                          STATE_REG <= DATA;
                                          S_REG <=(OTHERS=>'0');
                                          N_REG <=(OTHERS=>'0');
                                    ELSE
                                          S_REG <= S_REG + 1;
                                    END IF;
                              END IF;
```

```
                        WHEN DATA =>
                                TX_REG <= B_REG(0);
                                IF S_TICK='1' THEN
                                        IF S_REG=(SB_TICK-1) THEN
                                                S_REG <= (OTHERS=>'0');
                                                B_REG <= '0' & B_REG(DBIT-1 DOWNTO 1) ;
                                                IF N_REG=(DBIT-1) THEN
                                                        STATE_REG <= STOP;
                                                        N_REG <=(OTHERS=>'0');
                                                ELSE
                                                        N_REG <= N_REG + 1;
                                                END IF;
                                        ELSE
                                                S_REG <= S_REG + 1;
                                        END IF;
                                END IF;
                        WHEN STOP =>
                                TX_REG <= '1';
                                IF (S_TICK = '1') THEN
                                        IF S_REG=(SB_TICK-1) THEN
                                                STATE_REG <= IDLE;
                                                TX_DONE_TICK <= '1';
                                        ELSE
                                                S_REG <= S_REG + 1;
                                        END IF;
                                END IF;
                END CASE;
    END IF;
      END PROCESS;
      TX <= TX_REG;
PROCESS(CLK,RESET)
                VARIABLE CLK_CNT: STD_LOGIC_VECTOR(10 DOWNTO 0);
       BEGIN
                IF RESET ='0' THEN
                        S_TICK<='0'; CLK_CNT := (OTHERS=>'0');
                ELSIF RISING_EDGE(CLK) THEN
                        CLK_CNT := CLK_CNT + 1;
                        IF CLK_CNT = DVSR THEN
                                S_TICK <= '1';
                        ELSIF CLK_CNT = DVSR+1 THEN
                                S_TICK <= '0'; CLK_CNT := (OTHERS=>'0');
                        END IF;
                END IF;
      END PROCESS;
END ARCH;
```

D Anexo D

Interfaz de recogida de datos en Matlab:

```
closeall;clearall;clc;
delete(instrfind(('Port'),('COM3')));      %
pserial = serial('COM3','TimeOut', 10,'BaudRate', 19200,'DataBits',
   8,'Parity', 'none','StopBit', 1,'FlowControl', 'none');
fopen(pserial);
raw=[];
n_raw=0;
i=0;
    j=0;
m_cont=[];
    k=0;

while (k<51000)
byte=fread(pserial,1,'uint8');
        k=k+1;
raw=[raw byte];

end
fclose(pserial);
delete(pserial);
savewalk1.matraw;
```

E Anexo E

Ejemplo de un ruteado fijo de un sensor:

```
NET "g1[0].inst_sensor/OUT_AND_ENABLE"
ROUTE="{3;1;5vlx30ff676;9ca740c2!-1;56624;109992;S!0;-843;-824!1;-5613;"
"1515!1;-3310;2688!2;-2691;-1371!3;3310;1016!4;843;328;L!5;843;288;L!}";
NET "g1[0].inst_sensor/ring1"
ROUTE="{3;1;5vlx30ff676;109354c!-1;48320;109680;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring2"
ROUTE="{3;1;5vlx30ff676;39357018!-1;48320;109784;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring3"
ROUTE="{3;1;5vlx30ff676;8fbcd3af!-1;48320;109888;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring4"
ROUTE="{3;1;5vlx30ff676;25def279!-1;48320;109992;S!0;-843;-824!1;-5581;"
"2777!2;5581;551!3;843;352;L!}";
NET "g1[0].inst_sensor/ring5"
ROUTE="{3;1;5vlx30ff676;bdb5867f!-1;48320;112880;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring6"
ROUTE="{3;1;5vlx30ff676;3d1afcaa!-1;48320;112984;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring7"
ROUTE="{3;1;5vlx30ff676;29e2c9e8!-1;48320;113088;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring8"
ROUTE="{3;1;5vlx30ff676;6c98a1b6!-1;48320;113192;S!0;-843;-824!1;-5581;"
"2777!2;5581;551!3;843;352;L!}";
NET "g1[0].inst_sensor/ring9"
ROUTE="{3;1;5vlx30ff676;cab16a5a!-1;48320;116080;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring10"
ROUTE="{3;1;5vlx30ff676;aacebc5a!-1;48320;116184;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring11"
ROUTE="{3;1;5vlx30ff676;89531487!-1;48320;116288;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring12"
ROUTE="{3;1;5vlx30ff676;9757cf57!-1;48320;116392;S!0;-843;-824!1;-1798;"
"-2376!2;4505;-2521!3;1445;-1143!4;683;-488;L!}";
NET "g1[0].inst_sensor/ring13"
ROUTE="{3;1;5vlx30ff676;534a28cb!-1;52312;109112;S!0;-928;-843!1;245;"
"1123!2;683;-208;L!}";
NET "g1[0].inst_sensor/ring14"
ROUTE="{3;1;5vlx30ff676;5d81ecd6!-1;52312;109216;S!0;-683;-32!1;0;368!2;"
"683;-288;L!}";
NET "g1[0].inst_sensor/ring15"
ROUTE="{3;1;5vlx30ff676;c7c90080!-1;52312;109320;S!0;-683;-128!1;0;272!2;"
"683;-80;L!}";
NET "g1[0].inst_sensor/ring16"
ROUTE="{3;1;5vlx30ff676;12c83437!-1;52312;109424;S!0;-968;1277!1;-5296;"
"1244!2;5581;743!3;683;-408;L!}";
NET "g1[0].inst_sensor/ring17"
```

ROUTE="{3;1;5vlx30ff676;3b6aead9!-1;52312;112312;S!0;-928;-843!1;245;"
"1123!2;683;-208;L!}";
NET "g1[0].inst_sensor/ring18"
ROUTE="{3;1;5vlx30ff676;6a44964d!-1;52312;112416;S!0;-683;-32!1;0;368!2;"
"683;-288;L!}";
NET "g1[0].inst_sensor/ring19"
ROUTE="{3;1;5vlx30ff676;8bbfc24d!-1;52312;112520;S!0;-683;-128!1;0;272!2;"
"683;-80;L!}";
NET "g1[0].inst_sensor/ring20"
ROUTE="{3;1;5vlx30ff676;214b91c3!-1;52312;112624;S!0;-968;1277!1;-5296;"
"1244!2;5581;743!3;683;-408;L!}";
NET "g1[0].inst_sensor/ring21"
ROUTE="{3;1;5vlx30ff676;a5b238f!-1;52312;115512;S!0;-928;-843!1;245;1123!"
"2;683;-208;L!}";
NET "g1[0].inst_sensor/ring22"
ROUTE="{3;1;5vlx30ff676;cabf9e9d!-1;52312;115616;S!0;-683;-32!1;0;368!2;"
"683;-288;L!}";
NET "g1[0].inst_sensor/ring23"
ROUTE="{3;1;5vlx30ff676;fdc78ca9!-1;52312;115720;S!0;-683;-128!1;0;272!2;"
"683;-80;L!}";
NET "g1[0].inst_sensor/ring24"
ROUTE="{3;1;5vlx30ff676;f04ba794!-1;52312;115824;S!0;-968;1277!1;-1513;"
"-3909!2;1798;-3864!3;843;288;L!}";
NET "g1[0].inst_sensor/ring25"
ROUTE="{3;1;5vlx30ff676;2cc8d920!-1;52472;109680;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring26"
ROUTE="{3;1;5vlx30ff676;efa46acb!-1;52472;109784;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring27"
ROUTE="{3;1;5vlx30ff676;99951b11!-1;52472;109888;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring28"
ROUTE="{3;1;5vlx30ff676;67715ab9!-1;52472;109992;S!0;-843;-824!1;-3310;"
"2688!2;3310;680!3;843;272;L!}";
NET "g1[0].inst_sensor/ring29"
ROUTE="{3;1;5vlx30ff676;72b70a1a!-1;52472;112880;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring30"
ROUTE="{3;1;5vlx30ff676;541c0848!-1;52472;112984;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring31"
ROUTE="{3;1;5vlx30ff676;79f1617e!-1;52472;113088;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring32"
ROUTE="{3;1;5vlx30ff676;6f74bb66!-1;52472;113192;S!0;-843;-824!1;-3310;"
"2688!2;3310;680!3;843;272;L!}";
NET "g1[0].inst_sensor/ring33"
ROUTE="{3;1;5vlx30ff676;5c3bcec9!-1;52472;116080;S!0;-843;-536!1;0;440!2;"
"843;168;L!}";
NET "g1[0].inst_sensor/ring34"
ROUTE="{3;1;5vlx30ff676;6343a431!-1;52472;116184;S!0;-843;-632!1;0;208!2;"
"843;472;L!}";
NET "g1[0].inst_sensor/ring35"
ROUTE="{3;1;5vlx30ff676;71154cd2!-1;52472;116288;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
NET "g1[0].inst_sensor/ring36"

```
    ROUTE="{3;1;5vlx30ff676;70201f85!-1;52472;116392;S!0;-843;-824!1;-1756;"
    "-2432!2;4471;-2471!3;1437;-1153!4;683;-456;L!}";
    NET "g1[0].inst_sensor/ring37"
    ROUTE="{3;1;5vlx30ff676;6b305ff2!-1;56464;109112;S!0;-928;-843!1;245;"
    "1123!2;683;-208;L!}";
    NET "g1[0].inst_sensor/ring38"
    ROUTE="{3;1;5vlx30ff676;1fe39145!-1;56464;109216;S!0;-683;-32!1;0;368!2;"
    "683;-288;L!}";
    NET "g1[0].inst_sensor/ring39"
    ROUTE="{3;1;5vlx30ff676;25e4a768!-1;56464;109320;S!0;-683;-128!1;0;272!2;"
    "683;-80;L!}";
    NET "g1[0].inst_sensor/ring40"
    ROUTE="{3;1;5vlx30ff676;d89a701b!-1;56464;109424;S!0;-968;1277!1;-5296;"
    "1244!2;5581;743!3;683;-408;L!}";
    NET "g1[0].inst_sensor/ring41"
    ROUTE="{3;1;5vlx30ff676;e8e3a34a!-1;56464;112312;S!0;-928;-843!1;245;"
    "1123!2;683;-208;L!}";
    NET "g1[0].inst_sensor/ring42"
    ROUTE="{3;1;5vlx30ff676;69fde879!-1;56464;112416;S!0;-683;-32!1;0;368!2;"
    "683;-288;L!}";
    NET "g1[0].inst_sensor/ring43"
    ROUTE="{3;1;5vlx30ff676;a4cdb3bd!-1;56464;112520;S!0;-683;-128!1;0;272!2;"
    "683;-80;L!}";
    NET "g1[0].inst_sensor/ring44"
    ROUTE="{3;1;5vlx30ff676;44807fd3!-1;56464;112624;S!0;-968;1277!1;-5296;"
    "1244!2;5581;743!3;683;-408;L!}";
    NET "g1[0].inst_sensor/ring45"
    ROUTE="{3;1;5vlx30ff676;b019420a!-1;56464;115512;S!0;-928;-843!1;245;"
    "1123!2;683;-208;L!}";
    NET "g1[0].inst_sensor/ring46"
    ROUTE="{3;1;5vlx30ff676;85be7c4a!-1;56464;115616;S!0;-683;-32!1;0;368!2;"
    "683;-288;L!}";
    NET "g1[0].inst_sensor/ring47"
    ROUTE="{3;1;5vlx30ff676;a19aded0!-1;56464;115720;S!0;-683;-128!1;0;272!2;"
    "683;-80;L!}";
    NET "g1[0].inst_sensor/ring48"
    ROUTE="{3;1;5vlx30ff676;dea084ff!-1;56464;115824;S!0;-968;1277!1;-1513;"
    "-3909!2;1798;-3864!3;843;288;L!}";
    NET "g1[0].inst_sensor/ring49"
    ROUTE="{3;1;5vlx30ff676;b5ffae66!-1;56624;109680;S!0;-843;-536!1;0;440!2;"
    "843;168;L!}";
    NET "g1[0].inst_sensor/ring50"
    ROUTE="{3;1;5vlx30ff676;e0677e38!-1;56624;109784;S!0;-843;-632!1;0;208!2;"
    "843;472;L!}";
    NET "g1[0].inst_sensor/ring51"
    ROUTE="{3;1;5vlx30ff676;fcfc000a!-1;56624;109888;S!0;-843;-728!1;0;496!2;"
"843;296;L!}";
```

F Anexo F

Código para la utilización del System Monitor:

```
libraryieee;
use ieee.STD_LOGIC_1164.all;
useieee.numeric_std.all;
Library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entitysysmonitor is
port
  (DCLK              : in  STD_LOGIC;
   RESET             : in  STD_LOGIC;
   MEASURED_TEMP     : out  STD_LOGIC_VECTOR (15 downto 0);
   MEASURED_VCCINT   : out  STD_LOGIC_VECTOR (15 downto 0)

);
end entity sysmonitor;


--  attribute CORE_GENERATION_INFO : string;
--  attribute CORE_GENERATION_INFO of xilinx : architecture is "sysmonitor_arch";
architecturertl of sysmonitor is


  SIGNAL nRESET                              : STD_LOGIC;
signalden_reg, dwe_reg: STD_LOGIC_VECTOR (1 downto 0);
signalalm_int       : STD_LOGIC_VECTOR (15 downto 0);
signalvauxp_active   : STD_LOGIC_VECTOR (15 downto 0);
signalvauxn_active   : STD_LOGIC_VECTOR (15 downto 0);
signaldaddr         : STD_LOGIC_VECTOR (6 downto 0);    -- Address bus for the dynamic reconfiguration port
signal den          : STD_LOGIC;                        -- Enable Signal for the dynamic reconfiguration port
signaldi_drp          : STD_LOGIC_VECTOR (15 downto 0);   -- Input data bus for the dynamic reconfiguration
    port
signaldwe           : STD_LOGIC;                        -- Write Enable for the dynamic reconfiguration port
signaldo_drp        : STD_LOGIC_VECTOR (15 downto 0);   -- Output data bus for dynamic reconfiguration port
signaldrdy          : STD_LOGIC;                        -- Data ready signal for the dynamic reconfiguration port
signaleoc_drp       : STD_LOGIC;
signaleos_drp       : STD_LOGIC;
signal busy         : STD_LOGIC;                        -- ADC Busy signal
signaldclk_bufg      : STD_LOGIC;
signal i2c_sclk_in    : STD_LOGIC;
signal i2c_sclk_ts    : STD_LOGIC;
signal i2c_sda_in     : STD_LOGIC;
signal i2c_sda_ts     : STD_LOGIC;
typestate_type is (init_read, read_waitdrdy,
write_waitdrdy,
                read_reg00,
                reg00_waitdrdy,
                read_reg01,
                reg01_waitdrdy

              );
signal state : state_type;
```

```
signal VCCAUX_ALARM_OUT, VCCINT_ALARM_OUT, USER_TEMP_ALARM_OUT, OT, EOC, EOS:
    std_logic;
signal CHANNEL: std_logic_vector(4 downto 0);


component sysmon_wiz_v2_1 is
port (
      DADDR_IN              : in  STD_LOGIC_VECTOR (6 downto 0);      -- Address bus for the dynamic
    reconfiguration port
      DCLK_IN          : in  STD_LOGIC;                   -- Clock input for the dynamic reconfiguration port
      DEN_IN           : in  STD_LOGIC;                   -- Enable Signal for the dynamic reconfiguration port
      DI_IN                 : in  STD_LOGIC_VECTOR (15 downto 0);     -- Input data bus for the dynamic
    reconfiguration port
      DWE_IN           : in  STD_LOGIC;                   -- Write Enable for the dynamic reconfiguration port
      RESET_IN         : in  STD_LOGIC;                   -- Reset signal for the System Monitor control logic
      BUSY_OUT         : out  STD_LOGIC;                  -- ADC Busy signal
      CHANNEL_OUT      : out  STD_LOGIC_VECTOR (4 downto 0);   -- Channel Selection Outputs
      DO_OUT                : out  STD_LOGIC_VECTOR (15 downto 0);   -- Output data bus for dynamic
    reconfiguration port
      DRDY_OUT         : out  STD_LOGIC;                  -- Data ready signal for the dynamic reconfiguration
    port
      EOC_OUT          : out  STD_LOGIC;                  -- End of Conversion Signal
      EOS_OUT          : out  STD_LOGIC;                  -- End of Sequence Signal
      OT_OUT           : out  STD_LOGIC;                  -- Over-Temperature alarm output
      VCCAUX_ALARM_OUT   : out  STD_LOGIC;                -- VCCAUX-sensor alarm output
      VCCINT_ALARM_OUT   : out  STD_LOGIC;                -- VCCINT-sensor alarm output
      USER_TEMP_ALARM_OUT : out  STD_LOGIC;               -- Temperature-sensor alarm output
      VP_IN            : in  STD_LOGIC;                   -- Dedicated Analog Input Pair
      VN_IN            : in  STD_LOGIC
);
end component;

begin

nRESET<= not RESET;
  U_BUFG : BUFG
port map(
     I => DCLK,
     O =>dclk_bufg);

    U0 : sysmon_wiz_v2_1 port map
              (
                    DADDR_IN          =>daddr,
                    DCLK_IN           =>dclk_bufg,
                    DEN_IN            =>den_reg(0),
                    DI_IN             =>di_drp,
                    DWE_IN            =>dwe_reg(0),
                    RESET_IN          =>nRESET,
                    BUSY_OUT          => busy,
                    CHANNEL_OUT       => CHANNEL,
                    DO_OUT            =>do_drp,
                    DRDY_OUT          =>drdy,
                    EOC_OUT           =>eoc_drp,
                    EOS_OUT           =>eos_drp,
                    OT_OUT            => OT,
                    VCCAUX_ALARM_OUT    => VCCAUX_ALARM_OUT,
                    VCCINT_ALARM_OUT    => VCCINT_ALARM_OUT,
```

```
                USER_TEMP_ALARM_OUT  => USER_TEMP_ALARM_OUT,
                VP_IN              => '0',
                VN_IN              => '0');

EOC <= eoc_drp;
 EOS <= eos_drp;


NEXT_STATE_DECODE: process (dclk_bufg, nRESET)
begin
        if (nRESET = '1') then
                        state<= init_read;
        elsif (dclk_bufg'event and dclk_bufg = '1') then
                case (state) is
                        wheninit_read =>
                                daddr<= "1000000";
                                den_reg<= "10";
                                dwe_reg<= "00";
                                state<= read_waitdrdy;
                        whenread_waitdrdy =>
                                ifeos_drp = '1' then
                                        state<= write_waitdrdy;
                                        di_drp<=  do_drp    AND  "0000001111111111";  --Clearing
AVG bits for Configreg0
                                        daddr<= "1000000";
                                        den_reg<= "10";
                                        dwe_reg<= "10"; -- performing write
                                else
                                        state<= read_waitdrdy;
                                        den_reg<= "0" &den_reg(1) ;
                                        dwe_reg<= "0" &dwe_reg(1) ;
                                end if;
                        whenwrite_waitdrdy =>
                                ifdrdy = '1' then
                                        state<= read_reg00;
                                        den_reg<= den_reg;
                                        dwe_reg<= dwe_reg; --performing write
                                else
                                        den_reg<= "0" &den_reg(1) ;
                                        dwe_reg<= "0" &dwe_reg(1) ;
                                        state<= write_waitdrdy;
                                end if;
                        when read_reg00 =>
                                        daddr<= "0000000";
                                        den_reg<= "10";
                                        state<= reg00_waitdrdy;
                        when reg00_waitdrdy =>
                                ifeos_drp = '1' then
                                         MEASURED_TEMP <= do_drp;
                                        den_reg<= den_reg;
                                        dwe_reg<= dwe_reg;
                                        state<= read_reg01;
                                else
                                        den_reg<= "0" &den_reg(1) ;
                                        dwe_reg<= "0" &dwe_reg(1) ;
                                        state<= reg00_waitdrdy;
                                end if;
```

```
                                when read_reg01 =>
                                        daddr<= "0000001";
                                        den_reg<= "10";
                                        state<= reg01_waitdrdy;
                                when reg01_waitdrdy =>
                                        ifdrdy = '1' then
                                                MEASURED_VCCINT <= do_drp;
                                                den_reg<= den_reg;
                                                state<= read_reg00;
                                        else
                                                den_reg<= "0" &den_reg(1) ;
                                                state<= reg01_waitdrdy ;
                                        end if;

                                when others =>
                                        state<= init_read ;
                        end case;
                end if;
        end process;


endrtl;


------------------------------------------------------------------------------------------------------------------------

libraryieee;
use ieee.std_logic_1164.all;
useieee.numeric_std.all;
Library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity sysmon_wiz_v2_1 is
port (
        DADDR_IN                : in   STD_LOGIC_VECTOR (6 downto 0);       -- Address bus for the dynamic
reconfiguration port
        DCLK_IN             : in  STD_LOGIC;                    -- Clock input for the dynamic reconfiguration port
        DEN_IN              : in  STD_LOGIC;                    -- Enable Signal for the dynamic reconfiguration port
        DI_IN                   : in   STD_LOGIC_VECTOR (15 downto 0);      -- Input data bus for the dynamic
reconfiguration port
        DWE_IN              : in  STD_LOGIC;                    -- Write Enable for the dynamic reconfiguration port
        RESET_IN            : in  STD_LOGIC;                    -- Reset signal for the System Monitor control logic
        BUSY_OUT            : out  STD_LOGIC;                   -- ADC Busy signal
        CHANNEL_OUT         : out  STD_LOGIC_VECTOR (4 downto 0);    -- Channel Selection Outputs
        DO_OUT                  : out   STD_LOGIC_VECTOR (15 downto 0);     -- Output data bus for dynamic
reconfiguration port
        DRDY_OUT            : out  STD_LOGIC;                   -- Data ready signal for the dynamic reconfiguration
port
        EOC_OUT             : out  STD_LOGIC;                   -- End of Conversion Signal
        EOS_OUT             : out  STD_LOGIC;                   -- End of Sequence Signal
        OT_OUT              : out  STD_LOGIC;                   -- Over-Temperature alarm output
        VCCAUX_ALARM_OUT    : out  STD_LOGIC;                   -- VCCAUX-sensor alarm output
        VCCINT_ALARM_OUT    : out  STD_LOGIC;                   -- VCCINT-sensor alarm output
        USER_TEMP_ALARM_OUT : out  STD_LOGIC;                   -- Temperature-sensor alarm output
        VP_IN               : in  STD_LOGIC;                    -- Dedicated Analog Input Pair
        VN_IN               : in  STD_LOGIC
);
```

```
end sysmon_wiz_v2_1;

architecturexilinx of sysmon_wiz_v2_1 is

attribute X_CORE_INFO : string;
attribute X_CORE_INFO of xilinx : architecture is "sysmon_wiz_v2_1, Coregen 12.4";

signalaux_channel_p : std_logic_vector (15 downto 0);
signalaux_channel_n : std_logic_vector (15 downto 0);

begin

aux_channel_p(0) <= '0';
aux_channel_n(0) <= '0';

aux_channel_p(1) <= '0';
aux_channel_n(1) <= '0';

aux_channel_p(2) <= '0';
aux_channel_n(2) <= '0';

aux_channel_p(3) <= '0';
aux_channel_n(3) <= '0';

aux_channel_p(4) <= '0';
aux_channel_n(4) <= '0';

aux_channel_p(5) <= '0';
aux_channel_n(5) <= '0';

aux_channel_p(6) <= '0';
aux_channel_n(6) <= '0';

aux_channel_p(7) <= '0';
aux_channel_n(7) <= '0';

aux_channel_p(8) <= '0';
aux_channel_n(8) <= '0';

aux_channel_p(9) <= '0';
aux_channel_n(9) <= '0';

aux_channel_p(10) <= '0';
aux_channel_n(10) <= '0';

aux_channel_p(11) <= '0';
aux_channel_n(11) <= '0';

aux_channel_p(12) <= '0';
aux_channel_n(12) <= '0';

aux_channel_p(13) <= '0';
aux_channel_n(13) <= '0';

aux_channel_p(14) <= '0';
aux_channel_n(14) <= '0';
```

```
aux_channel_p(15) <= '0';
aux_channel_n(15) <= '0';


 SYSMON_INST : SYSMON
generic map(
      INIT_40 => X"1000", -- configreg 0
      INIT_41 => X"20f0", -- configreg 1
      INIT_42 => X"0a00", -- configreg 2
      INIT_48 => X"0301", -- Sequencer channel selection
      INIT_49 => X"0000", -- Sequencer channel selection
      INIT_4A => X"0000", -- Sequencer Average selection
      INIT_4B => X"0000", -- Sequencer Average selection
      INIT_4C => X"0000", -- Sequencer Bipolar selection
      INIT_4D => X"0000", -- Sequencer Bipolar selection
      INIT_4E => X"0000", -- Sequencer Acq time selection
      INIT_4F => X"0000", -- Sequencer Acq time selection
      INIT_50 => X"b5ed", -- Temp alarm trigger
      INIT_51 => X"5999", -- Vccint upper alarm limit
      INIT_52 => X"e000", -- Vccaux upper alarm limit
      INIT_54 => X"a93a", -- Temp alarm reset
      INIT_55 => X"5111", -- Vccint lower alarm limit
      INIT_56 =>X"caaa", -- Vccaux lower alarm limit
      INIT_57 => X"ae4e",  -- Temp alarm OT reset
      SIM_MONITOR_FILE => "design.txt"
      )

port map (
      CONVST             => '0',
      CONVSTCLK          => '0',
DADDR(6 downto 0)   => DADDR_IN(6 downto 0),
      DCLK               => DCLK_IN,
      DEN                => DEN_IN,
DI(15 downto 0)     => DI_IN(15 downto 0),
      DWE                => DWE_IN,
      RESET              => RESET_IN,
VAUXN(15 downto 0)  =>aux_channel_n(15 downto 0),
VAUXP(15 downto 0)  =>aux_channel_p(15 downto 0),
ALM(2)             => VCCAUX_ALARM_OUT,
ALM(1)             => VCCINT_ALARM_OUT,
ALM(0)             => USER_TEMP_ALARM_OUT,
      BUSY               => BUSY_OUT,
CHANNEL(4 downto 0) => CHANNEL_OUT(4 downto 0),
DO(15 downto 0)     => DO_OUT(15 downto 0),
      DRDY               => DRDY_OUT,
      EOC                => EOC_OUT,
      EOS                => EOS_OUT,
      JTAGBUSY           => open,
      JTAGLOCKED         => open,
      JTAGMODIFIED       => open,
      OT                 => OT_OUT,
      VN                 => VN_IN,
      VP                 => VP_IN
);
endxilinx;
```

G PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido) ..... 1.200 €
- Placa de pruebas Virtex-5 ..... 2.500€
- Material de oficina ..... 20€
- Total de ejecución material ..... 2.720€

2) Gastos generales

- 16 % sobre Ejecución Material ..... 435,2 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 163,2 €

4) Honorarios Proyecto

- 420 horas a 15 € / hora ..... 6.300 €

5) Material fungible

- Gastos de impresión ..... 60 €
- Encuadernación ..... 20 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 9.698,4 €

7) I.V.A. aplicable

- 21 % Subtotal presupuesto ..... 2.036,66 €

8) Total presupuesto

- Total presupuesto ..... 11.735,06 €

Madrid, Marzo de 2015

El Ingeniero Jefe de Proyecto

Fdo.: Mónica Torre Albarsanz
Ingeniero de Telecomunicación

H PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un ANÁLISIS DE TEMPERATURA EN FPGAs. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.