DISEÑO DIGITAL 2

MEMORIA DEL DISEÑO:

MEDT: MEDidor de Temperatura

Autor: Ignacio Blasco Hernandez
Curso 2019-2020.

Control de versiones

Versión	Fecha	Autor	Cambios realizados
0.0	20/04/2020	DTE	Inicial
1.0	23/05/2020	Ignacio Blasco	Completo

Tabla de contenido

1 Especificación del diseño. 4
1.1 Introducción 4
1.2 Interfaces 4
1.2.1 Interfaz con el sensor de temperatura y humedad 4
1.2.2 Interfaz con los pulsadores 5
1.2.3 Interfaz con la barra de displays de 7 segmentos 5
1.3 Especificaciones 5
2 Diseño jerárquico 6
2.1 Bloque (Reloj) 6
2.2 Bloque (Interfaz SPI) 7
2.3 Bloque (Procesado y Conversiones) 7
2.4 Bloque (Presentación Temperatura) 8
3 Diseño detallado 9
4 Pruebas de verificación funcional de MEDT 12
4.1 Test nº 1 12
4.2 Test nº 2 12
4.3 Test nº 3 13
5 Diseño físico 14
5.1 Asignación de pines 14
5.2 Restricciones de la síntesis 15
5.3 Recursos utilizados 16
5.4 Frecuencia máxima de reloj 16
6 Bibliografía 16

1 Especificación del diseño.

1.1 Introducción

El Medidor de Temperatura (MEDT) permite la lectura de la temperatura utilizando el sensor LM71 (disponible en la tarjeta DECA) y su representación en los displays en grados centígrados, Farenheit o en kelvin a elección del usuario. El MEDT utiliza también los dos pulsadores disponibles en la tarjeta DECA; el izquierdo (P1) para cambiar la representación de la temperatura entre las tres unidades de medida posibles cada vez que se detecte su activación y el derecho (P2) para cambiar el periodo de actualización de la medida de temperatura.

1.2 Interfaces

El diagrama de bloques general del diseño se muestra en la siguiente figura.

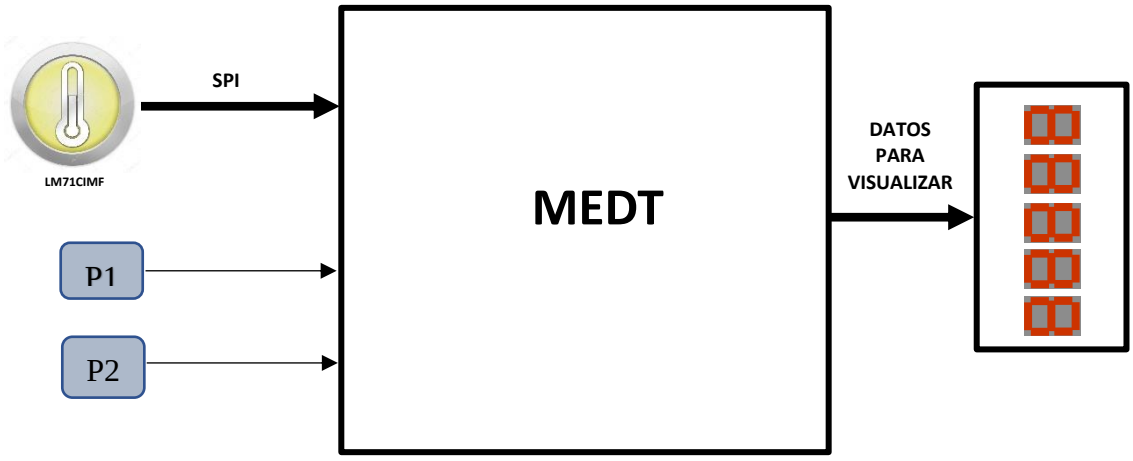

Fig. 1. Interconexión del MEDT al sensor de temperatura, pulsadores y barra de displays de 7 segmentos.

1.2.1 Interfaz con el sensor de temperatura y humedad

El sistema se comunica con el sensor de temperatura y humedad por medio de una interfaz SPI. Las señales de esta interfaz son las siguientes:

Señal	Dirección	Descripción
CS	Salida	Habilita la generación de CL
CL	Salida	Señal de sincronismo entre sensor y la interfaz.
SDATA	Entrada	Datos enviados desde el sensor (escritos en la línea durante el flanco de bajada) a la interfaz (leídos durante el ciclo de subida).

1.2.2 Interfaz con los pulsadores

El sistema se controla con dos pulsadores, uno para seleccionar las unidades con las que se visualiza la temperatura y otro para controlar su período de actualización:

Señal	Dirección	Descripción
Switch_P1	Entrada	Permite alternar la representación de la temperatura entre Celsius, Kelvin y Fahrenheit.
Switch_P2	Entrada	Permite alternar el tiempo de adquisición de la medida entre 4s, 6s, 8s, 10s, y 12s.

1.2.3 Interfaz con la barra de displays de 7 segmentos

El sistema realiza la presentación de los datos utilizando una barra de displays de 7 segmentos del tipo cátodo común. La interfaz es la siguiente:

Señal	Dirección	Descripción
seg [7..0]	salida	seg [0]: segmento g seg [1]: segmento f seg [2]: segmento e seg [3]: segmento d seg [4]: segmento c seg [5]: segmento b seg [6]: segmento a seg [7]: segmento punto
mux_disp [4..0]	salida	mux_disp [0]: cátodo del display 0 (LSD) mux_disp [1]: cátodo del display 1 mux_disp [2]: cátodo del display 2 mux_disp [3]: cátodo del display 3 mux_disp [4]: cátodo del display 4 (MSD)

La interfaz permite iluminar solo un display a la vez. El display se selecciona activando (a nivel bajo) el cátodo correspondiente. El display activo se ilumina de acuerdo con el código de 7 segmentos y punto decimal introducido (nivel alto).

1.3 Especificaciones

Las especificaciones funcionales y no funcionales se detallan en el documento [1].

2 Diseño jerárquico

El diagrama de la Fig. 2 representa el primer nivel de la jerarquía del diseño[1]:

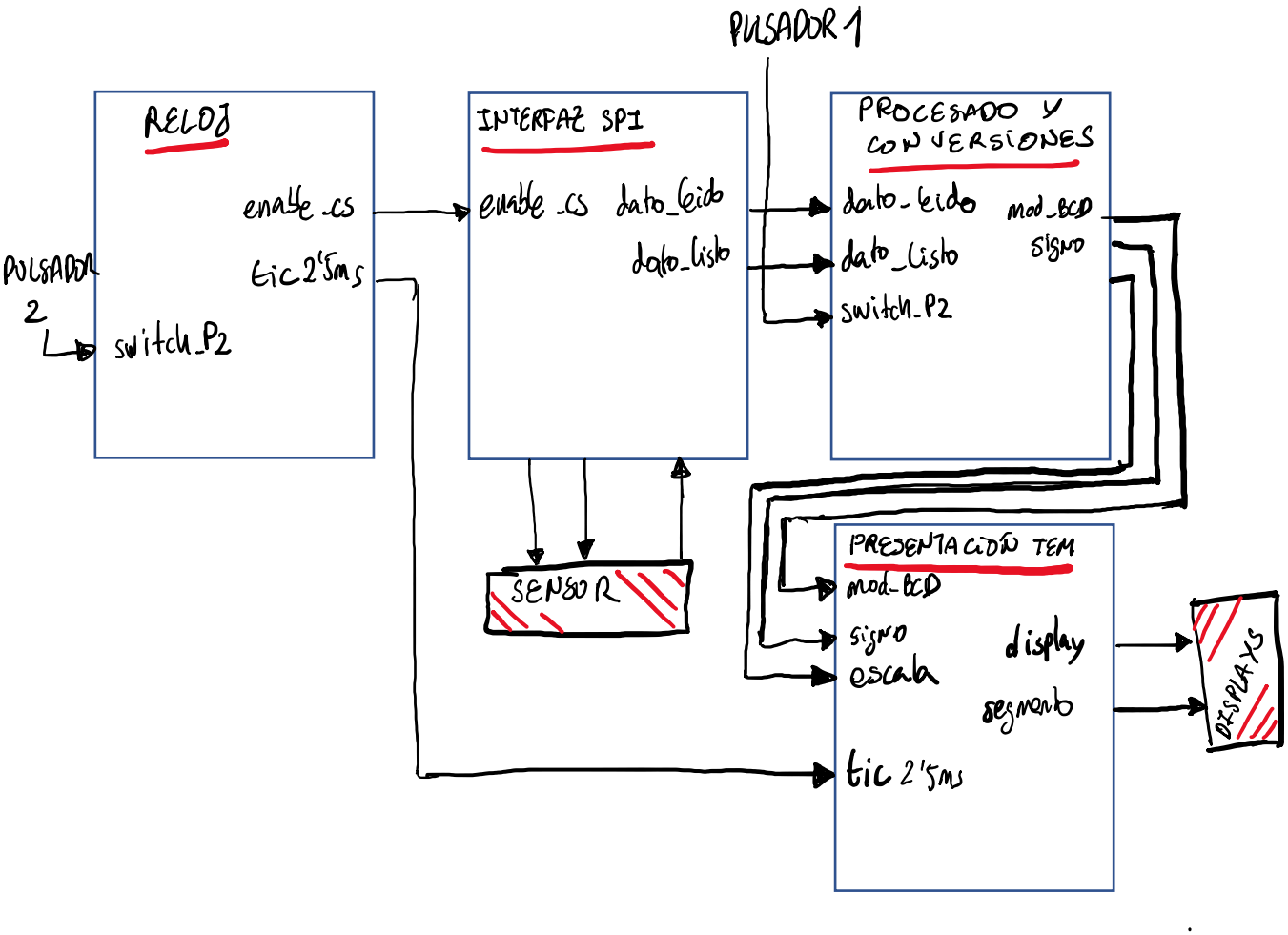

Fig. 2. Diagrama de bloques del primer nivel de la jerarquía de MEDT.

En los siguientes subapartados se describe la interfaz y la función de cada uno de estos bloques.

2.1 Bloque (Reloj)

Este bloque es el encargado de generar la señal que indica que se puede realizar una nueva medida, es decir, que han pasado x segundos (x dependerá de cuantas veces se haya pulsado P2) desde que se realizó la anterior medida. Además, genera un tic de 2.5ms que le indica al módulo de presentación que se ha de cambiar el display multiplexado.

[1] Todos los bloques tienen un reloj común, clk, y una entrada de reset asíncrono, rst_n, activa a nivel bajo. Estas señales no se incluyen en las interfaces por simplicidad.

Señal	Dirección	Descripción
Ena_CS	Salida	Indica que se puede hacer una nueva medida
Tic	Salida	(tic2’5ms) Indica que han pasado 2.5ms desde el anterior tic.
Switch_P2	Entrada	Permite alternar el tiempo de adquisición de la medida entre 4s, 6s, 8s, 10s, y 12s.

2.2 Bloque (Interfaz SPI)

Este bloque es el encargado de obtener la medida del sensor. Inicia una nueva medida cada vez que ena_cs se pone a 1 y pone envía al siguiente modulo la medida tomada junto con una señal que indica que la transferencia ha terminado.

Señal	Dirección	Descripción
Ena_cs	Entrada	Indica que hay que realizar una nueva medida.
Dato_out	Salida	(dato leído) Contiene el valor de la temperatura en c2 que se ha leído.
Flag_data_read	Salida	(dato listo) Indica que se ha completado la lectura.
CS	Salida	Habilita la generación de CL.
CL	Salida	Señal de sincronismo entre sensor y la interfaz.
SDAT	Entrada	Datos enviados desde el sensor (escritos en la línea durante el flanco de bajada) a la interfaz (leídos durante el ciclo de subida).

2.3 Bloque (Procesado y Conversiones)

Este módulo se encarga de realizar la conversión en función de la escala definida (que variará según se realicen pulsaciones en P2) cada vez que se indique que la transferencia ha terminado, y, de enviar al siguiente modulo, el módulo en BCD de la medida (según la escala) y el signo de la medida (según la escala)

Señal	Dirección	Descripción
Dato_out	Entrada	(dato_leido) Contiene el valor de la temperatura en c2 que se ha leído.
Flag_data_read	Entrada	(dato_listo) Indica que se puede realizar la conversión.
Switch_P1	Entrada	Permite alternar la representación de la temperatura entre Celsius, kelvin y Fahrenheit.
Mod_BCD	Salida	Módulo de la temperatura en la escala seleccionada codificado en BCD.
Sgn	Salida	(Signo) Signo de la temperatura en la escala seleccionada, 1 en caso de negativo y 0 en caso de positivo.
Escala	Salida	Escala en la que se representa la temperatura 1 si es Celsius, 2 si es kelvin y 3 si es Fahrenheit.

2.4 Bloque (Presentación Temperatura)

Por último, este bloque se encarga mostrar la temperatura mediante la multiplexación de los distintos segmentos.

Señal	Dirección	Descripción
Mod_BCD	Entrada	Módulo de la temperatura en la escala seleccionada codificado en BCD.
Sgn	Entrada	(Signo) Signo de la temperatura en la escala seleccionada, 1 en caso de negativo y 0 en caso de positivo.
Escala	Entrada	Escala en la que se representa la temperatura 1 si es Celsius, 2 si es kelvin y 3 si es Fahrenheit.
tic	Entrada	(tic2’5ms) Cuando se activa, cambia el display multiplexado.
mux_disp	Salida	(Display) Indica que segmento encender.
seg	Salida	(Seg) Enciende los leds correspondientes del segmento indicado por display

3 Diseño detallado

El proyecto está almacenado en la carpeta MEDT, que a su vez contiene las carpetas hdl, modelsim y quartus. La carpeta *hdl* contiene los ficheros RTL y estructural del diseño. La carpeta *modelsim* contiene el proyecto de simulación (MEDT.mpf), y los ficheros donde se definen los diferentes test-benches. Finalmente, *quartus* contiene el proyecto para el diseño físico y los ficheros relacionados con éste.
En cuanto a los ficheros RTL encontramos los siguientes:
Nota: Por simplicidad se entiende que toda la lógica secuencial síncrona tiene una entrada de reloj y una entrada de reset asíncrono activo a nivel bajo.

Timer.vhd, en este fichero se modela la lógica necesaria para generar un tic de un pulso de reloj cada 2.5 ms y otro cada 4s, 6s, 8s, 10s o 12s, según las veces que se haya pulsado el botón P2. En cuanto al hardware que modela, encontramos:

1) Un contador con salida de fin de cuenta que permite generar un tic cada 2.5ms, el módulo de este contador se obtiene con el siguiente cálculo, si el reloj del circuito es de 50MHz, esto quiere decir que generará un tic cada 20ns, mediante la siguiente relación podemos obtener el módulo del contador (esto se aplica a todos) $2.5ms = Mod * 20ns \rightarrow Mod = \frac{2.5ms}{20ns} = 125000$.
   Este contador contará de 1 a 125000 incluido, cuando llegue al final mediante una sentencia concurrente que modela un comparador, pondrá la señal de tic 1 durante un ciclo de reloj.
2) Un registro que permite registrar las pulsaciones de P2, de manera que, mediante el uso de dos señales, switch_p2_T1 y switch_p2_T2, se puedan conocer las transiciones que realiza el pulsador.
   Mediante una sentencia concurrente la señal pulsación se pondrá a 1 durante un ciclo de reloj cada vez que el pulsador haga una transición de 1 a 0, es decir, cada vez que se pulse.
3) Un contador de módulo 5 que permite modificar la tasa de refresco de la medida, este contador tiene una entrada de habilitación que corresponde a la señal pulsación, de esta manera, cada vez que se pulsa el botón P2, el contador sumará 1, hasta que llegue a 4, momento en el que pone la cuenta a 0.
4) Una sentencia concurrente que permite seleccionar el fin de cuenta del contador que se explicará en el siguiente punto. Este fin de cuenta (señal fin_cnt) dependerá de la señal selección_tiempo, la siguiente tabla indica los valores que toma en función de seleccion_tiempo:

Estado de selección_tiempo	Estado de fin_cnt	De donde sale el valor
0	1600	$\frac{4s}{2.5ms}$
1	2400	$\frac{6s}{2.5ms}$
2	3200	$\frac{8s}{2.5ms}$

3	4000	$\frac{10s}{2.5ms}$
4	4800	$\frac{12s}{2.5ms}$

5) Por último, se modela un contador de modulo programable, con entrada de habilitación (tic, permite reducir en gran medida el módulo del contador) y con salida de fin de cuenta, el módulo del contador dependerá de la señal fin_cnt. Cada vez que se llega al final de cuenta, se pone a uno la señal ena_CS durante un ciclo de reloj, esto le indica al bloque Interfaz_SPI que se puede realizar una nueva medida.

Interfaz_spi.vhd, en este fichero se modela la lógica que permite comunicarse con el sensor, encontramos el siguiente hardware:

1) Un registro que pondrá a cero la línea CS, que va directamente al sensor, cada vez que ena_cs valga 1, este registro cuenta con una entrada de preset síncrono, activo a nivel alto, que pondrá la línea CS a 1, cada vez que la señal data_readed valga 1.

2) Para generar la señal CL, encargada de la sincronización entre la interfaz y el sensor, se modela un contador de modulo 50 (reloj de 1 MHz) con entrada de habilitación activa a nivel bajo (CS). Mediante un comparador, generará el reloj CL en una señal auxiliar cl_aux que está adelantada un ciclo de reloj, para posteriormente filtrarlo registrando la señal cl_aux en CL que estará libre de glitches de cara al sensor. Además, mediante otro comparador, la señal ena_data_read, indicará al registro de desplazamiento cuando realizar una lectura. Esta señal se pone a 1 cuando el contador llega a 25, que condice con el flanco de subida de CL.

3) Por último, se modela un registro de desplazamiento con entrada de habilitación activa a nivel alto (ena_data_read), este registro almacena 9 bits recibidos por el sensor a través de la line SDAT en la salida data_out. Cuando se leen los 9 bits, la señal data_readed, se pondrá a nivel alto durante un ciclo de reloj, poniendo a 1 CS (como se explicó en el punto 1) y poniendo 1 durante un ciclo de reloj también, la salida flag_data_readed, que indica al módulo encargado del procesado y conversión del dato que el dato leído está listo.

4)

DataToDisplay.vhd, en este fichero se modela la lógica que procesa el dato de temperatura obtenido por el sensor haciendo la conversión que corresponda. Encontramos el siguiente hardware:

1) Un registro que permite registrar las pulsaciones de P1, de manera que, mediante el uso de dos señales, switch_p1_T1 y switch_p1_T2, se puedan conocer las transiciones que realiza el pulsador.
   Mediante una sentencia concurrente la señal pulsación se pondrá a 1 durante un ciclo de reloj cada vez que el pulsador haga una transición de 1 a 0, es decir, cada vez que se pulse.
2) Un contador de módulo 3 y con entrada de habilitación (señal pulsación) cuyo estado de cuenta (señal escala) indica la escala en la que se va a convertir el

dato. Un 1 indica que la conversión se hará en Celsius, un 2 en kelvin y un 3 en Fahrenheit.

3) Un registro que registra la señal de data_out (explicado en el bloque anterior) en la señal data_cels, que tendrá el valor de la temperatura en complemento a 2.
4) Para realizar las conversiones a kelvin y Fahrenheit, se hace lo siguiente:
    1) Para hacer la conversión a kelvin (data_kelvin), se extiende con el signo 3 bits a la izquierda la señal data_cels y se suma 273.
    2) Para hacer la conversión a Fahrenheit(data_fheit), se multiplica la señal data_cels por 1.8125 y se suma 32 o 33.
    Para multiplicar por 1.8125 es necesario extender el signo de data_cels 3 bits por la izquierda y añadir 4 bits ("0000") a la derecha, la siguiente tabla indica como queda la multiplicación por 1.8125:
    En la tabla data_cels es una señal de 9 bits, que por espacio se nombrará con la letra a, y data_fheit_aux como b.

Multiplicador	Parte entera												Parte decimal			
	b(15)	b(14)	b(13)	b(12)	b(11)	b(10)	b(9)	b(8)	b(7)	b(6)	b(5)	b(4)	b(3)	b(2)	b(1)	b(0)
$2^0(1)$	a(8)	a(8)	a(8)	a(8)	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	0	0	0	0
$2^{-1}(0.5)$	a(8)	a(8)	a(8)	a(8)	a(8)	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	0	0	0
$2^{-2}(0.25)$	a(8)	a(8)	a(8)	a(8)	a(8)	a(8)	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	0	0
$2^{-4}(0.0625)$	a(8)	a(8)	a(8)	a(8)	a(8)	a(8)	a(8)	a(8)	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)

Tras sumar el resultado de lo anterior en la señal data_fheit_aux, obtendremos una señal de 16 bits, donde los 4 de menor peso se corresponden a la parte decimal. Para completar la conversión se eliminará la parte decimal (data_fheit_aux(15 downto 4) y se tendrá que sumar 32 cuando b(3) sea '0' y 33 cuando b(3) sea '1'. b(3) es el bit de mayor peso de la parte decimal y por tanto cuando este a 1, se redondeará al entero superior. El resultado de esto será la señal data_fheit, de 12 bits y en c2.

5) Para obtener le bit de signo (señal sgn) basta con, aprovechando que las tres medidas están en complemento a 2, bastará con comparar el bit de mayor peso de la señal que corresponda según la escala para poner la señal sgn a 1 o a 0. Esto se hace mediante una sentencia concurrente.
6) A continuación, se pasa la medida que corresponda según la escala a binario natural, esto se hace de la siguiente manera. Si el bit de mayor peso es 0, esto indica que la señal es positiva y se pone a la salida (en num_bin) tal cual, por otro lado, cuando el bit de mayor peso es 1, se resta uno a la señal que corresponda y se invierte.
7) Por último, mediante lógica combinacional se obtiene el módulo de la temperatura codificada en BCD (mod_BCD).

Las señales escala, sgn, y mod, de salida en este fichero, serán entradas del siguiente bloque.

Por último, el fichero presentacion_temperatura, este contiene la lógica combinacional y secuencial necesaria para representar en los displays de 7 segmentos la temperatura obtenida en la escala solicitada.

4 Pruebas de verificación funcional de MEDT

El plan de pruebas diseñado consiste en comprobar, en cada fase, que se cumple con las especificaciones requeridas y que el circuito se comporta según lo esperado en cada momento. Por ello se ha hecho un test por cada hito.

4.1 Test nº 1

En el hito 1, el test consiste en comprobar que el sistema lee la temperatura del sensor cada 4 segundos (escalado) y presenta en C2.

Ubicación de los ficheros del test	En / /modelsim/test_interfaz_spi.vhd	
Simulación escalada	SI	
Ficheros	/hdl/Timer.vhd	Pone a 1 ena_sc cada 4 segundos (escalado).
	/hdl/interfaz_spi	Comunicación con el sensor y obtención del dato.
Descripción del test	Las comprobaciones que se han hecho en este test son, comprobar que las señales CS y CL cumplen con el estándar SPI definido, y se comportan de manera correcta, que el dato leído en cada flanco de subida se desplaza correctamente y, por último, que se genera la señal que indica que la transacción ha llegado al final y el dato en Celsius en C2 es correcto, desde los -40ºC a los 150ºC.	

4.2 Test nº 2

En el hito 2, el test consiste en comprobar que las conversiones de la temperatura se hacen de manera correcta, que se cambia de escala cuando se produce una pulsación y que la obtención del signo y del módulo en BCD son correctos.

Ubicación de los ficheros del test	En / /modelsim/test_interfaz_spi.vhd	
Simulación escalada	SI	
Ficheros	/hdl/Timer.vhd	Pone a 1 ena_sc cada 4 segundos (escalado).
	/hdl/interfaz_spi	Comunicación con el sensor y obtención del dato.

	/hdl/dataToDisplay.vhd	Procesado de la temperatura y obtención del signo y del módulo en BCD.
	/hdl/MEDT.chd	Estructural
Descripción del test	Las comprobaciones que se han hecho en este test son, comprobar que las pulsaciones se registran correctamente y que la escala cambia en consecuencia. Comprobar que las conversiones se hacen de manera correcta comprobando cada valor y, por último, que se obtiene el módulo en BCD y el signo de manera adecuada para todos los valores en las tres escalas. Para ello se han ejecutado 3 simulaciones, una para cada escala.	

4.3 Test nº 3

En el hito 3, el test consiste en comprobar que se general correctamente el tic de 2.5ms, que la selección del tiempo de medida cambia con cada pulsación y que la generación del tick de ena_cs varía en función de dicha selección.

Ubicación de los ficheros del test	En / /modelsim/test_interfaz_spi.vhd	
Simulación escalada	SI	
Ficheros	/hdl/Timer.vhd	Genera la señal tic cada 2.5ms y ena_cs según las veces que se haya pulsado.
	/hdl/interfaz_spi	Comunicación con el sensor y obtención del dato.
	/hdl/dataToDisplay	Procesado de la temperatura y obtención del signo y del módulo en BCD.
Descripción del test	Las comprobaciones que se han hecho en este test son, comprobar que el tic de 2.5 se genera correctamente (observando el progreso de la cuenta, está escalado a 10 el módulo), que con cada pulsación la señal fin_cnt varía según la señal selección tiempo, y se ha comprobado la generación de ena_cs (observando el estado del contador y el fin de cuenta). Se han hecho 5 simulaciones, una para cada selección de tiempo. Además, se ha comprobado que se mantiene el funcionamiento de las fases anteriores.	

5 Diseño físico

En este apartado se documentan los detalles básicos relacionados con el diseño físico del circuito: la asignación de pines, las restricciones de la síntesis y los informes que proporciona Quartus Prime sobre los recursos de la FPGA utilizados y la frecuencia máxima de reloj obtenida.

5.1 Asignación de pines

En la siguiente tabla se detalla la asignación de los pines de la interfaz de MEDT a los pines de la FPGA especificando para cada caso el tipo de pin, el número de pin de la FPGA que se corresponde con el pin del diseño, el banco al que corresponde y el estándar de entrada/salida que utiliza.

	dirección	Pin FPGA	I/O bank	I/O standard
clk	Input	M8	2	2.5-V
rst_n	Input	H21	6	1.5-V Scmitt Trigger
mux_disp [4]	Output	R11	3	3.3-V LVTTL
mux_disp [3]	Output	AB8	3	3.3-V LVTTL
mux_disp [2]	Output	W8	3	3.3-V LVTTL
mux_disp [1]	Output	W6	3	3.3-V LVTTL
mux_disp [0]	Output	Y5	3	3.3-V LVTTL
seg [6]	Output	W16	4	3.3-V LVTTL
seg [5]	Output	AB11	4	3.3-V LVTTL
seg [4]	Output	W15	4	3.3-V LVTTL
seg [3]	Output	AB10	4	3.3-V LVTTL
seg [2]	Output	AA15	4	3.3-V LVTTL
seg [1]	Output	W12	4	3.3-V LVTTL
seg [0]	Output	AA13	4	3.3-V LVTTL
switch_p1	Input	H22	6	1.5 V Schmitt Trigger
switch_P2	Input	H21	6	1.5 V Schmitt Trigger
CS	Output	AB4	3	3.3-V LVTTL
CL	Output	AA1	3	3.3-V LVTTL
SDAT	Input	Y2	3	3.3-V LVTTL

5.2 Restricciones de la síntesis

Se ha utilizado el siguiente fichero sdc : MEDT.out.sdc, en el cual se destacan las siguientes líneas:

```
##
## DEVICE “10M50DAF484C6GES"
##

#**************************************************************
# Time Information
#**************************************************************

set_time_format -unit ns -decimal_places 3



#**************************************************************
# Create Clock
#**************************************************************

create_clock -name {clk} -period 20.000 -waveform {0.000 10.000 } [get_ports {clk}]

#**************************************************************

# Set Clock Uncertainty

#**************************************************************


set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020

set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020

set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020

set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020


#**************************************************************

# Set False Path

#**************************************************************


set_false_path -from [get_ports {CL CS SDAT mux_disp [0] mux_disp [1]
mux_disp[2] mux_disp[3] mux_disp[4] nRst seg[0] seg[1] seg[2] seg[3] seg[4] seg[5]
seg[6] switch_P2 switch_p1}]
```

5.3 Recursos utilizados

La siguiente imagen muestra los recursos utilizados:

Total logic elements	366 / 49,760 (< 1 %)
Total registers	74
Total pins	19 / 360 (5 %)

5.4 Frecuencia máxima de reloj

Tras completar el diseño físico, obtenemos en la pestaña de TimeQuest Timing Analyzer que la frecuencia máxima del reloj es de 175.87MHz a 85ºC.

	Fmax	Restricted Fmax	Clock Name
1	175.87 MHz	175.87 MHz	clk

6 Bibliografía

[1] Especificación del diseño: MEDIDOR DE TEMPERATURA (MEDT) [moodle DD2]
[2] Tarjeta DECA-MAX10 (página web del fabricante). [online] https://www.arrow.com/es-mx/products/deca/arrow-development-tools
[3] Tarjeta XDECA. Manual de usuario. [moodle DD2-documentacion técnica]
[4] Sensor LM71CIMF. Datasheet [moodle DD2 en BT2: Actividad 1]