Universidad Politécnica de Madrid
ETSI de Telecomunicación

Departamento de Ingeniería Electrónica

Circuitos Electrónicos (CELT)

Enunciado del proyecto

Curso 2023-2024

Termómetro digital

Álvaro de Guzmán Fernández

CALENDARIO DE LA ASIGNATURA

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	
	05-sep	06-sep	07-sep	08-sep	
	SELECCIÓN DE TURNO		Y PAREJA		
11-sep **LT1**	12-sep **MT1**	13-sep **XT1**	14-sep **JT1**	15-sep **VM1 / VT1**	
18-sep **LT2**	19-sep **MT2**	20-sep **XT2**	21-sep **JT2**	22-sep **VM2 / VT2**	
25-sep	26-sep **MT3**	27-sep **XT3**	28-sep **JT3**	29-sep **VM3 / VT3**	
02-oct **LT3**	03-oct **MT4**	04-oct **XT4**	05-oct **JT4**	06-oct **VM4 / VT4**	
09-oct **LT4**	10-oct **JT5**	11-oct **XT5**	12-oct	13-oct	10/10: Día de jueves
16-oct **LT5**	17-oct **MT5**	18-oct **XT6**	19-oct **JT6**	20-oct **VM5 / VT5**	
SEMANA DE EXÁMENES					
	31-oct **MT6**	01-nov	02-nov **JT7**	03-nov **VM6 / VT6**	
06-nov **LT6**	07-nov **MT7**	08-nov **XT7**	09-nov	10-nov **VM7 / VT7**	
13-nov **LT7**	14-nov **MT8**	15-nov **XT8**	16-nov **JT8**	17-nov **VM8 / VT8**	
20-nov **LT8**	21-nov **VM9 / VT9**	22-nov **XT9**	23-nov **JT9**	24-nov **VM10 / VT10**	21/11: Día de viernes
27-nov **LT9**	28-nov **MT9**	29-nov **XT10**	30-nov **JT10**	01-dic **VM11 / VT11**	Entrega de memorias
04-dic **LT10**	05-dic **MT10**	06-dic	07-dic	08-dic	
11-dic **LT11**	12-dic **MT11**	13-dic **XT11**	14-dic **JT11**	15-dic **VM12 / VT12**	Pruebas de validacion
18-dic **LT12**	19-dic **MT12**	20-dic **XT12**	21-dic **JT12**	22-dic	22/12: Día de ajuste

- **Los montajes deben de hacerse siempre en casa. El tiempo de laboratorio sólo debería utilizarse para medir, ajustar y buscar errores.**
- **Igualmente el código VHDL deberá prepararse y simularse en casa. Aproveche el tiempo de laboratorio para medir y buscar fallos.**
- **En esta asignatura no hay una distribución obligatoria de módulos por sesiones. En todo caso se sugiere el siguiente plan de trabajo**.

SESIÓN	MÓDULOS
1	Clase introductoria
2	Familiarización con los equipos
3	**Módulos 1 y 2**
4	**Módulos 3 y 4**
5	**Módulo 5**
6	**Módulos 6 y 7**
7	**Módulos 8 y 9**
8	**Módulo 10**
9	Sesión adicional
10	Sesión adicional
11	Entrega memoria
12	Prueba de validación

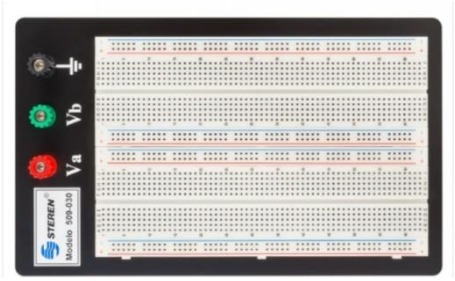
Evaluación

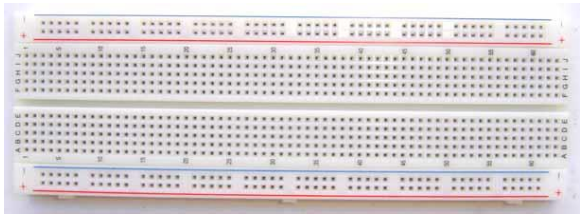
- En la sesión 11 (23:59 horas) se deberá entregar una memoria del trabajo realizado según el formato publicado. Esta memoria cuenta 20% de la nota. **Es una actividad NO RECUPERABLE**.
- En la sesión 12 se realizará la prueba de validación del circuito que cuenta un 30% de la nota. **Nota mínima: 4/10. Es una actividad NO RECUPERABLE.**
- Además se realizará un examen escrito sobre el funcionamiento de la práctica el día de la convocatoria oficial en enero, que cuenta el 50% de la nota. **Nota mínima 2/10 puntos**.

MATERIAL NECESARIO

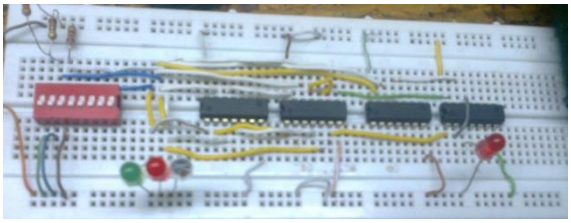
Para realizar esta práctica necesitará los siguientes materiales:

- 1 tablero de inserción (protoboard), a ser posible con bananas incluidas, (vea el anexo II):





- Cables rígidos de colores para las conexiones de los componentes (no sirven los cables llamados Arduino). Se sugiere rojo, negro y azul para la alimentación, y amarillo, blanco y verde para las señales:

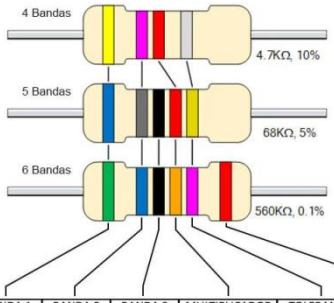
- Pinzas, alicates de puntas, destornillador plano pequeño y pelacables:




- Componentes electrónicos que se irán describiendo a lo largo de este documento.

- Para la primera sesión:

  - 2 LEDs (uno rojo y otro verde)
  - 2 resistencias de 1 kΩ
  - 2 condensadores electrolíticos de 100 µF
  - 2 condensadores de plástico de 100 nF
  - 2 condensadores cerámicos de 100 pF



CÓDIGO DE COLORES DE RESISTENCIAS

COLOR	BANDA 1	BANDA 2	BANDA 3	MULTIPLICADOR	TOLERANCIA	TCR(ppm/k)
NEGRO	0	0	0	1	1% (F)	100
MARRON	1	1	1	10	2% (G)	50
ROJO	2	2	2	100		15
NARANJA	3	3	3	1K		25
AMARILLO	4	4	4	10K		
VERDE	5	5	5	100K	0.5% (D)	
AZUL	6	6	6	1M	0.25%(C)	10
VIOLETA	7	7	7	10M	0.1%(B)	5
GRIS	8	8	8	100M	0.05%(A)	
BLANCO	9	9	9	1G		
ORO				0.1	5% (J)	
PLATA				0.001	10% (K)	
NADA						

EN EL LABORATORIO NO EXISTE MATERIAL DISPONIBLE PARA EL PRÉSTAMO, POR LO QUE SE RECOMIENDA LLEVAR EL MATERIAL DESDE LA PRIMERA SESIÓN.

NORMAS OBLIGATORIAS PARA TODAS LAS SESIONES

En toda la parte analógica emplee siempre:

1. **Condensadores de desacoplo** para asegurar la estabilidad de la alimentación y problemas debidos a ruidos producidos por los transitorios de conmutación de los elementos digitales. Vea la sesión 1 y los vídeos explicativos.
2. **Alimentación entre +5 y −5 V**.
3. **Resistencias y condensadores de la serie E12**, cuyos multiplicadores son:

   1 - 1,2 - 1,5 - 1,8 - 2,2 - 2,7 - 3,3 - 3,9 - 4,7 - 5,6 - 6,8 - 8,2

4. **Valores de resistencias entre 100 Ω y 100 kΩ**. Valores más pequeños de 100 Ω dan lugar a corrientes altas que los operacionales no pueden suministrar. Valores mayores de 100 kΩ dan lugar a corrientes muy bajas que pueden confundirse con el ruido en el circuito.
5. **Valores de condensadores entre 100 pF y 1 µF**. Valores más pequeños de 100 pF empiezan a ser comparables a las capacidades existentes entre las pistas del circuito y valores mayores de 1 µF se emplean generalmente en sistemas de alimentación no siendo generalmente necesarios en circuitos de baja potencia.
6. **Para realizar las capturas de pantalla del osciloscopio** utilice la aplicación PROMAX disponible en el escritorio del ordenador que tiene en su puesto. Deberán verse claramente las indicaciones de las escalas en ambos ejes.

En la parte digital emplee siempre:

1. **Lenguaje VHDL para describir los circuitos digitales**.
2. **Diseño digital síncrono para los circuitos secuenciales (vea el manual de referencia de la tarjeta BASYS 2, página 14)**.
3. **Simulación mediante ISIM** (simulador integrado en el ISE Webpack).
4. **Síntesis sobre la tarjeta BASYS2 disponible en el laboratorio.**
5. **RESPETE EN TODO MOMENTO LOS NOMBRES DE ARCHIVO QUE SE INDICAN EN EL ENUNCIADO. EN OTRO CASO SU CÓDIGO NO PODRÁ SER EVALUADO**.

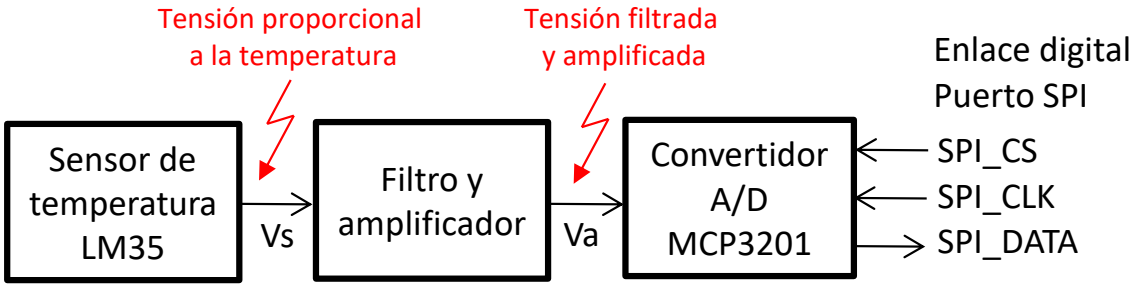
DESCRIPCIÓN GENERAL

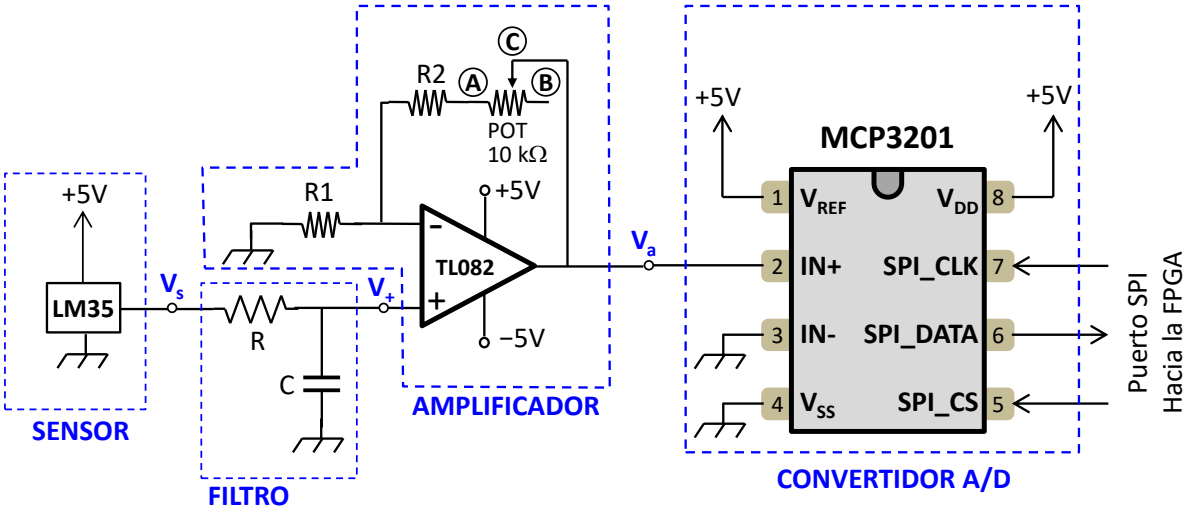
En este proyecto se construirá un circuito para realizar un termómetro digital con precisión de décimas de grado Celsius. Básicamente se empleará un sensor analógico de temperatura, que suministra una tensión proporcional a la temperatura medida, un circuito de filtrado y amplificación, y un convertidor analógico-digital. Posteriormente, la señal digital será procesada para obtener el valor de la temperatura en BCD, de modo que pueda ser presentada en unos displays de 7 segmentos.

1. Esquema general del circuito

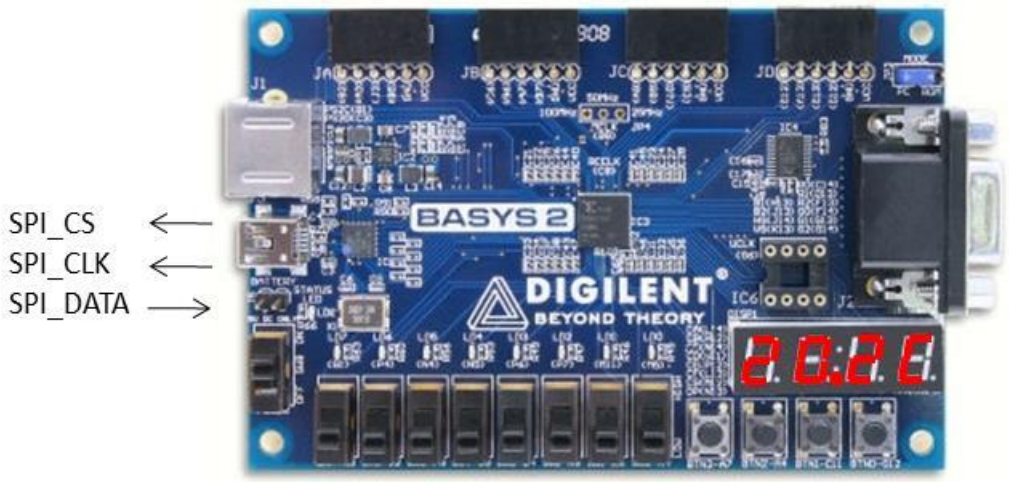
El proyecto se compone de dos partes:

1. **Un circuito analógico** que proporciona una tensión proporcional a la temperatura debidamente filtrada y estabilizada. La tensión es convertida en un valor digital empleando un convertidor analógico/digital con una salida digital en serie de tipo puerto SPI (este puerto se describirá más adelante en este documento).





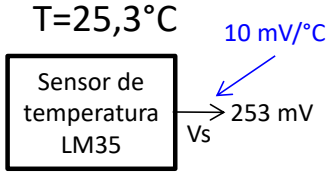
2. **Un circuito digital** descrito en VHDL que lee la señal digital obtenida del circuito anterior mediante el puerto SPI y calcula el valor de la temperatura, obtiene su representación en BCD con la precisión de décimas de grado y genera 3 dígitos BCD: Decenas, Unidades y Décimas de grado Celsius. Estos dígitos se presentarán en los 3 primeros displays de la tarjeta BASYS2 del laboratorio, el punto decimal, y una C en el display de la derecha indicando grados Celsius.



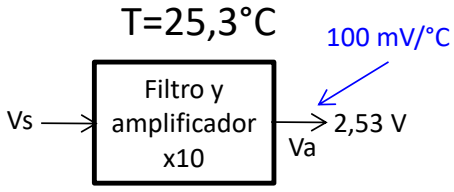
2. Circuito analógico

El circuito analógico genera una tensión proporcional a la temperatura empleando un circuito integrado **LM35**. Esta tensión es débil y muy sensible al ruido, por lo tanto debe ser filtrada y amplificada para adecuar su valor al necesario para la entrada del convertidor analógico/digital MCP3201. Las etapas que componen este circuito son las siguientes:

Sensor de temperatura: El LM35 es un circuito integrado que proporciona a su salida una tensión proporcional a la temperatura ambiente (expresada en grados Celsius) con una precisión de **10 mV/ºC.** Es decir, que si la tensión ambiente es de 25,3ºC, la tensión proporcionada será de 253 mV.



Filtro y amplificador: Para tener precisión de una décima de grado debemos ser capaces de leer tensiones con precisión de 1 mV, que es una tensión muy débil. Esto hace que cualquier ruido (con esa amplitud o mayor) que pueda añadirse a la señal, dará un error en la lectura de la temperatura. Para estabilizar dicha tensión se realizará un filtro paso bajo con una frecuencia de corte muy baja (f_c =10 Hz), que debería atenuar fluctuaciones rápidas de la tensión. Como consecuencia de la baja frecuencia de corte, la señal tarda aproximadamente $1/f_c$ =0,1 s en estabilizarse (ya que los condensadores dentro del filtro necesitan un tiempo para cargarse). **Generalmente, se toman muestras con un tiempo de muestreo de al menos 10 veces el tiempo de estabilización, por lo tanto se tomarán muestras de temperatura cada segundo para asegurar la estabilidad de la señal leída.** Además también es necesario amplificar la señal para aprovechar al máximo las características del convertidor A/D. La ganancia en tensión de este amplificador será de 10 V/V. De este modo, aumentamos el factor de conversión del LM35 a **100 mV/°C**.



Convertidor A/D: Emplearemos un convertidor de aproximaciones sucesivas MCP3201 de 12 bits con una tensión de referencia (V_{REF}) de 5V. Esto quiere decir que el convertidor proporciona 2^{12} valores binarios proporcionales a la tensión de entrada desde "000000000000" para 0V hasta "111111111111" para $\frac{2^{12}-1}{2^{12}} \cdot V_{REF}$ ~5V. El escalón mínimo de tensión que puede apreciar es, por tanto, $V_{REF}/2^{12}$= 1,22 mV. Si no amplificásemos la señal del LM35, esto no sería suficiente para apreciar la décima de grado, pero al amplificar la tensión por un factor 10, la precisión aumenta a 100 mV/ºC y la décima de grado equivale a un cambio de 10 mV, que puede ser perfectamente detectado por el convertidor. Si el valor

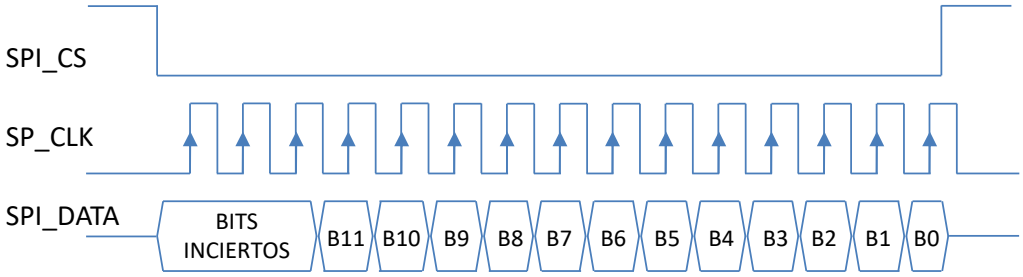
binario obtenido del convertidor es DATOS_ADC (12 bits), considerando su valor en base 10 ($DATOS\_ADC_{(10)}$), la tensión analógica correspondiente será:

$$V_a = \frac{DATOS\_ADC_{(10)}}{2^{12}} \cdot V_{REF} = \frac{DATOS\_ADC_{(10)}}{4096} \cdot 5V$$

3. Circuito digital

El circuito digital tiene que leer el valor binario proporcionado por el convertidor A/D. Esto se lleva a cabo a través de un interfaz llamado puerto SPI. Este puerto permite leer los bits correspondientes en serie (uno tras otro en el tiempo). Desde el convertidor A/D, el puerto SPI tiene dos entradas (SPI_CS y SPI_CLK) y una salida (SPI_DATA). La lectura de los datos desde la FPGA responde al siguiente protocolo:

1. En reposo la línea SPI_CS debe mantenerse a '1' y la línea SPI_CLK debe mantenerse a '0'.
2. Para iniciar una conversión, la línea SPI_CS debe forzarse a '0'.
3. A continuación se deben generar 15 pulsos en la línea SPI_CLK. El MCP3201 emplea esta señal para generar los bits correspondientes al valor binario de forma secuencial en la línea SPI_DATA. Dichos bits deben capturarse en cada flanco de subida de SPI_CLK. Los 3 primeros son inciertos (debido al tiempo que necesita el MCP3201 para convertir los datos), y los siguientes 12 bits son los correspondientes al valor convertido empezando por el más significativo (MSB).
4. Finalmente se debe poner a '1' la línea SPI_CS y luego se debe poner a '0' la línea SPI_CLK (en este orden). Vea la siguiente figura:



El circuito digital deberá generar las señales SPI_CS y SPI_CLK capturando los bits correspondientes al valor digital (SPI_DATA) en un registro de desplazamiento. Una vez leído el valor binario de 12 bits (DATOS_ADC), el circuito digital tiene que calcular el valor de la temperatura en grados Celsius y generar los tres dígitos BCD (decenas, unidades y décimas) y presentarlos en los displays.

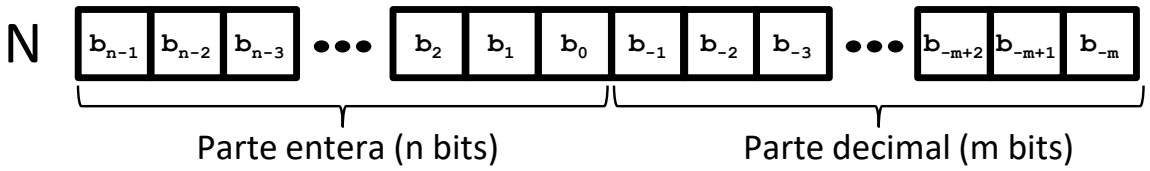
Conversión del valor leído en temperatura:

El valor leído DATOS_ADC es un número binario. Su valor expresado en base 10 ($DATOS\_ADC_{(10)}$), es proporcional a la tensión a la salida del amplificador según la expresión indicada anteriormente [$V_a = DATOS\_ADC_{(10)}/4096 \cdot 5V$]. Además sabemos que en este punto del circuito, la precisión del sensor es de 100 mV/ºC. Por tanto, para calcular la temperatura deberíamos realizar la siguiente operación:

$$TEMP = V_a \cdot \frac{1}{100mV/℃}\ [℃]$$

$$TEMP = \frac{DATOS\_ADC_{(10)}}{2^{12}} \cdot 5V \cdot \frac{1}{100mV/℃} = \frac{DATOS\_ADC_{(10)} \cdot 50}{4096}\ \ [℃]$$

Puesto que DATOS_ADC es un valor binario, estas operaciones hay que realizarlas en binario. Para ello vamos a emplear aritmética de punto fijo. Recuerde que en aritmética de punto fijo parte de los bits del vector (los más significativos) constituyen la parte entera mientras que el resto constituyen la parte decimal. En general, el valor en base 10 de un número binario (N) con n bits en la parte entera y m en la parte decimal se obtiene según la expresión:



$$N(base\ 10) = \sum_{i=-m}^{n-1} b_i \cdot 2^i$$

Donde b_i son los valores de los diferentes bits (0 o 1).

En este proyecto vamos a emplear **lógica interna de 32 bits para evitar desbordamientos en los cálculos y finalmente aproximaremos el resultado a un número de 16 bits despreciando 10 bits en la parte entera y 6 en la decimal)**. Esto nos dará un valor de temperatura con precisión suficiente para apreciar la décima de grado. Vamos a mostrar todo esto con un ejemplo:

Pongamos como ejemplo que el valor leído es DATOS_ADC="100000011011". Este valor se corresponde con 2075 en base 10, que a su vez nos daría el valor TEMP=25, 32958984375 ºC. Al realizar la operación en binario tenemos:

DATOS_ADC (12 bits)

1	0	0	0	0	0	0	1	1	0	1	1

Trabajamos ahora con lógica interna de 32 bits. Las operaciones a realizar son dos: multiplicar este valor por 50 y a continuación dividirlo por 2^{12}=4096.

DATOS_ADC x 50

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	1	1	0

DATOS_ADC x 50 / 4096 (dividir entre 4096="1000000000000" (en base 2) consiste en mover la coma 12 bits hacia la izquierda)

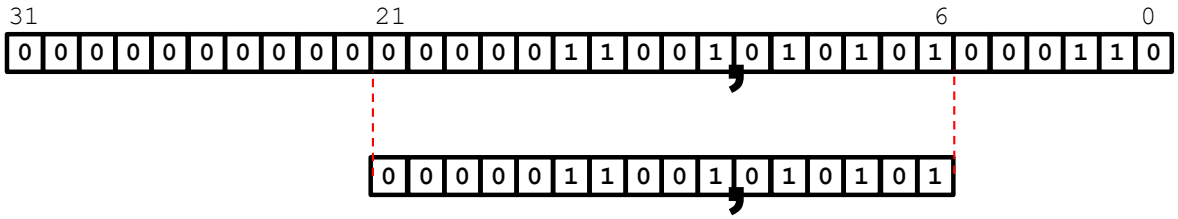
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	,	0	1	0	1	0	1	0	0	0	1	1	0

Coma decimal

Como se puede ver, la parte entera del valor obtenido es "11001" correspondiente a 25 en base 10, y la parte decimal es "010101000110" correspondiente a $\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\frac{1}{1024}+$ $+\frac{1}{2048}$ =0.32958984375 en base 10.

En este proyecto nosotros emplearemos aritmética de punto fijo de 16 bits, con **10 bits para la parte entera y 6 bits para la parte decimal**, ya que es suficiente para la precisión que necesitamos en nuestra aplicación (décimas de grado)**.** Para ello sencillamente vamos a seleccionar los 16 bits (10 enteros y 6 decimales que nos interesan) despreciando los 10 bits más significativos y los 6 bits menos significativos del valor de 32 bits. Haciendo esto, el error cometido no es muy grande puesto que en ese caso tendríamos:

Valor leído del ADC y ya multiplicado por 50



Valor de 16 bits con 10 bits en la parte entera y 6 en la parte decimal

Donde la parte decimal es: "010101" correspondiente a $\frac{1}{4}+\frac{1}{16}+\frac{1}{64}$ =0,328125 (suficiente)

El valor entero más grande expresable con 10 bits es 1023, y el valor decimal más pequeño expresable con 6 bits es: 1/64=0,015625, lo que es de sobra suficiente para nuestra aplicación.

Representación en BCD:

El valor binario obtenido no es adecuado para poder representarse en un display, donde aparecerán las cifras decimales correspondientes a las decenas, unidades y décimas de grado. Para ello es necesario convertir este valor binario a representación BCD. La conversión se realiza mediante un módulo combinacional que opera del siguiente modo:

- Primero separa la parte entera (ENT) de la parte decimal (DEC).
- A continuación realiza la siguiente lógica:

DECENAS	**UNIDADES**	**DÉCIMAS**
Si ENT>=90 DECENAS=9 en otro caso	Si ENT>=90 UNIDADES=ENT-90 en otro caso	Si DEC<6 DECIMAS=0 en otro caso
Si ENT>=80 DECENAS=8 en otro caso	Si ENT>=80 UNIDADES=ENT-80 en otro caso	Si DEC<12 DECIMAS=1 en otro caso
Si ENT>=70 DECENAS=7 en otro caso	Si ENT>=70 UNIDADES=ENT-70 en otro caso	Si DEC<18 DECIMAS=2 en otro caso
Si ENT>=60 DECENAS=6 en otro caso	Si ENT>=60 UNIDADES=ENT-60 en otro caso	Si DEC<24 DECIMAS=3 en otro caso
Si ENT>=50 DECENAS=5 en otro caso	Si ENT>=50 UNIDADES=ENT-50 en otro caso	Si DEC<30 DECIMAS=4 en otro caso
Si ENT>=40 DECENAS=4 en otro caso	Si ENT>=40 UNIDADES=ENT-40 en otro caso	Si DEC<36 DECIMAS=5 en otro caso
Si ENT>=30 DECENAS=3 en otro caso	Si ENT>=30 UNIDADES=ENT-30 en otro caso	Si DEC<42 DECIMAS=6 en otro caso
Si ENT>=20 DECENAS=2 en otro caso	Si ENT>=20 UNIDADES=ENT-20 en otro caso	Si DEC<48 DECIMAS=7 en otro caso
Si ENT>=10 DECENAS=1 en otro caso	Si ENT>=10 UNIDADES=ENT-10 en otro caso	Si DEC<54 DECIMAS=8 en otro caso
DECENAS=0	UNIDADES=ENT	DECIMAS=9

En el caso de la parte entera básicamente se comprueba la decena en la que se encuentra el valor binario y se deciden la DECENA y la UNIDAD.

En el caso de la parte decimal tenemos 6 bits, correspondientes a 64 valores. Dividiendo 64 entre 10 (que son las cifras decimales en BCD) tenemos un valor de 6,4 cuya parte entera es 6. Por tanto, en función del valor DEC decidimos la cifra BCD (0 entre 0 y 5, 1 entre 6 y 11, 2 entre 12 y 17, 3 entre 18 y 23 etc…).

Por ejemplo, en el caso anterior: **000011001,010101**

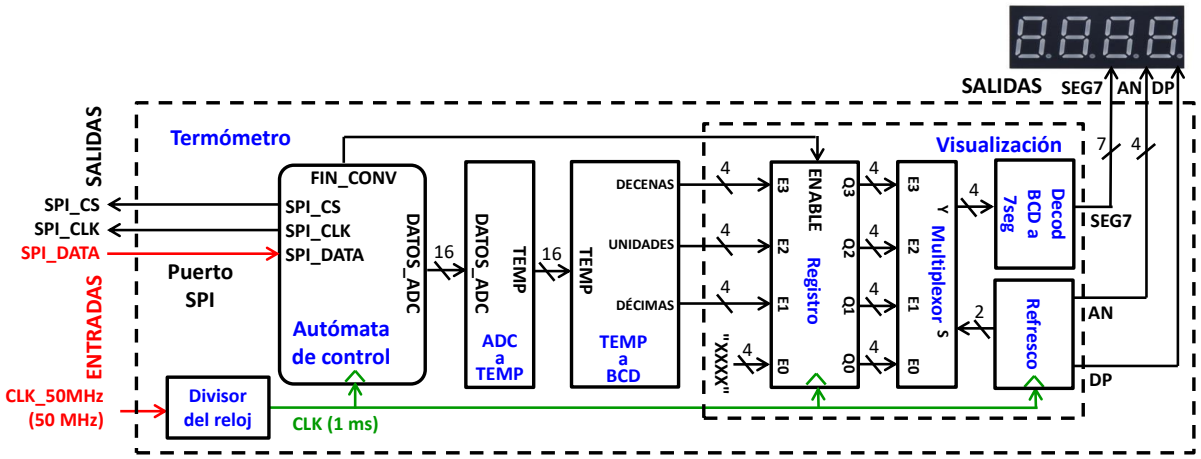
Parte entera: $000011001=25_{(10)}$, es mayor que 20, luego DECENAS=2, UNIDADES=25-20=5.

Parte decimal: $010101 = 21_{(10)}$, está entre 18 y 23, luego DECIMAS=3.

Se correspondería con el valor TEMP = 25,3 °C

Circuito digital completo:

Este circuito (para realizar en VHDL sobre la FPGA) responde al siguiente esquema:



Las entradas (en rojo) son: **CLK_50Mhz** (Señal de reloj de 50 MHz de la FPGA) y **SPI_DATA** (señal digital que permite la comunicación con el puerto SPI del MCP3201). Las salidas (en negro) son: **SPI_CS, SPI_CLK** (señales digitales que permiten la comunicación con el puerto SPI del MCP3201), **SEG7, AN y DP** (señales para excitar los displays de 7 segmentos y el punto decimal).

Los módulos que lo componen son los siguientes:

Divisor de reloj: Toda la lógica interna del sistema estará sincronizada con un reloj de 1 kHz de frecuencia (1 ms de periodo). Por tanto es necesario desarrollar un divisor de frecuencia que obtenga un reloj de 1 kHz a partir del reloj de la FPGA (50 MHz).

Autómata de control: Realiza la lectura del valor digital del convertidor A/D cada segundo. Establece la comunicación mediante el protocolo del puerto SPI y almacena el valor digital leído en un registro interno de desplazamiento. Además, una vez leído el valor digital de la

temperatura activa la señal FIN_CONV (haciendo que el registro en el módulo de visualización capture el valor de temperatura en BCD).

ADC a TEMP: Se trata de un módulo combinacional que realiza la conversión entre el valor digital leído (DATOS_ADC) y la temperatura en grados Celsius (TEMP). Realiza la lógica explicada anteriormente generando un número decimal en punto fijo con 10 bits enteros y 6 decimales empleado lógica interna de 32 bits.

TEMP a BCD: La temperatura TEMP está expresada en aritmética de punto fijo en binario. Para poder representarla en un display debe ser convertida a BCD. En este módulo combinacional se lleva a cabo la conversión según se explica en las páginas anteriores.

Módulo de visualización: (**Para entender este módulo deberá leer los ejemplos 3 y 4 del "Manual de referencia de la tarjeta BASYS 2"**). Este módulo toma los valores BCD de DECENAS, UNIDADES y DECIMAS cada vez que se activa la señal FIN_CONV (cada segundo) y presenta sus valores en los displays. Este módulo está compuesto de los siguientes submódulos:

- Registro: Cada vez que se valida la lectura de una temperatura (FIN_CONV) se activa la señal enable. Esto hace que los valores de DECENAS, UNIDADES y DECIMAS se almacenen en el registro y permanezcan estables hasta la siguiente lectura.
- Refresco: El circuito de refresco actualiza periódicamente (de forma muy rápida) el carácter que se visualiza en cada uno de los displays. Para ello controla el multiplexor que selecciona el carácter para visualizar y genera una salida (**AN**) de activación de display en sincronía. También activa la señal que enciende el punto decimal (**DP**)
- Multiplexor: Conecta una de las entradas con la salida en función de la señal de control (**S**).
- Decodificador de BCD a 7 segmentos: Convierte el código BCD expresado con 4 bits a la configuración de segmentos que deben encenderse para visualizar ese carácter. Los valores superiores a 9 los representa como cifras hexadecimales (ABCDEF).

Termómetro: Es el módulo general que agrupa e interconecta todos los elementos del circuito digital.

ANTES DE LA PRIMERA SESIÓN

Antes de acudir a la primera sesión del laboratorio deberá realizar las siguientes actividades:

1. Si su placa de inserción protoboard no es nueva, asegúrese de que tiene sus conexiones internas en buen estado. Para ello, si es posible, abra la parte trasera, apriete bien todos los peines metálicos y fíjese si falta alguno. Escoja los lugares de la protoboard que estén en mejor estado para realizar su montaje.
2. Visione los vídeos en MOODLE:
    - Descripción y uso de la placa de inserción protoboard.
    - Instalación de los condensadores de desacoplo.
    - Instalación de la alimentación y LEDs para la detección de cortocircuitos.
    - Realización de un diseño sobre la placa protoboard.

Consejos de diseño:

1. Emplee siempre cables cortos y ajustados a la distancia, no realice arcos que puedan soltarse fácilmente en el transporte diario de la placa.
2. Recorte las patillas de las resistencias y condensadores antes de insertarlos. Así evitará cortocircuitos entre componentes.
3. Cuando tenga que seleccionar un producto de valores puede emplear la siguiente tabla que es muy útil para ajustar los componentes a los valores de la serie E12. La tabla muestra los productos entre los valores de la serie. **Ejemplo:** si le sale en los cálculos un producto cuya mantisa es RC=5,62, los valores más cercanos en esta tabla serían 4,7 y 1,2 o bien 5,6 y 1,0 en los componentes (con sus correspondientes exponentes).

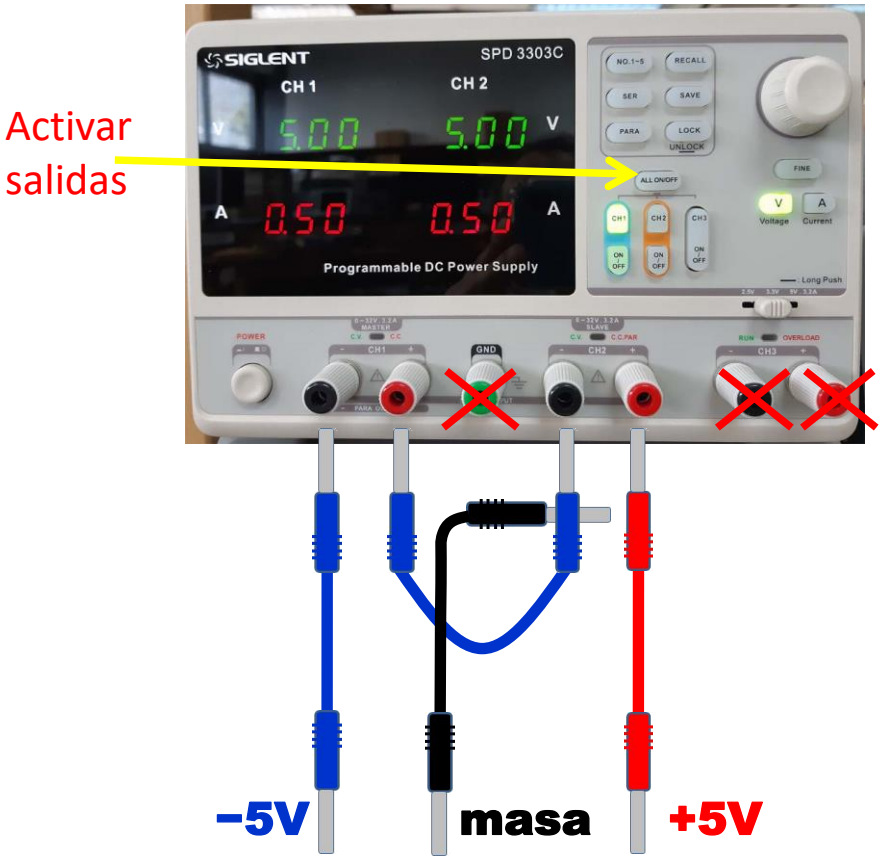
PRODUCTOS ENTRE LOS VALORES DE LA SERIE E12

	1	**1,2**	**1,5**	**1,8**	**2,2**	**2,7**	**3,3**	**3,9**	**4,7**	**5,6**	**6,8**	**8,2**
1	1	1,2	1,5	1,8	2,2	2,7	3,3	3,9	4,7	5,6	6,8	8,2
1,2		1,44	1,8	2,16	2,64	3,24	3,96	4,68	5,64	6,72	8,16	9,84
1,5			2,25	2,7	3,3	4,05	4,95	5,85	7,05	8,4	10,2	12,3
1,8				3,24	3,96	4,86	5,94	7,02	8,46	10,08	12,24	14,76
2,2					4,84	5,94	7,26	8,58	10,34	12,32	14,96	18,04
2,7						7,29	8,91	10,53	12,69	15,12	18,36	22,14
3,3							10,89	12,87	15,51	18,48	22,44	27,06
3,9								15,21	18,33	21,84	26,52	31,98
4,7									22,09	26,32	31,96	38,54
5,6										31,36	38,08	45,92
6,8											46,24	55,76
8,2												67,24

SESIÓN 1: Familiarización con los instrumentos del laboratorio y preparación de la protoboard.

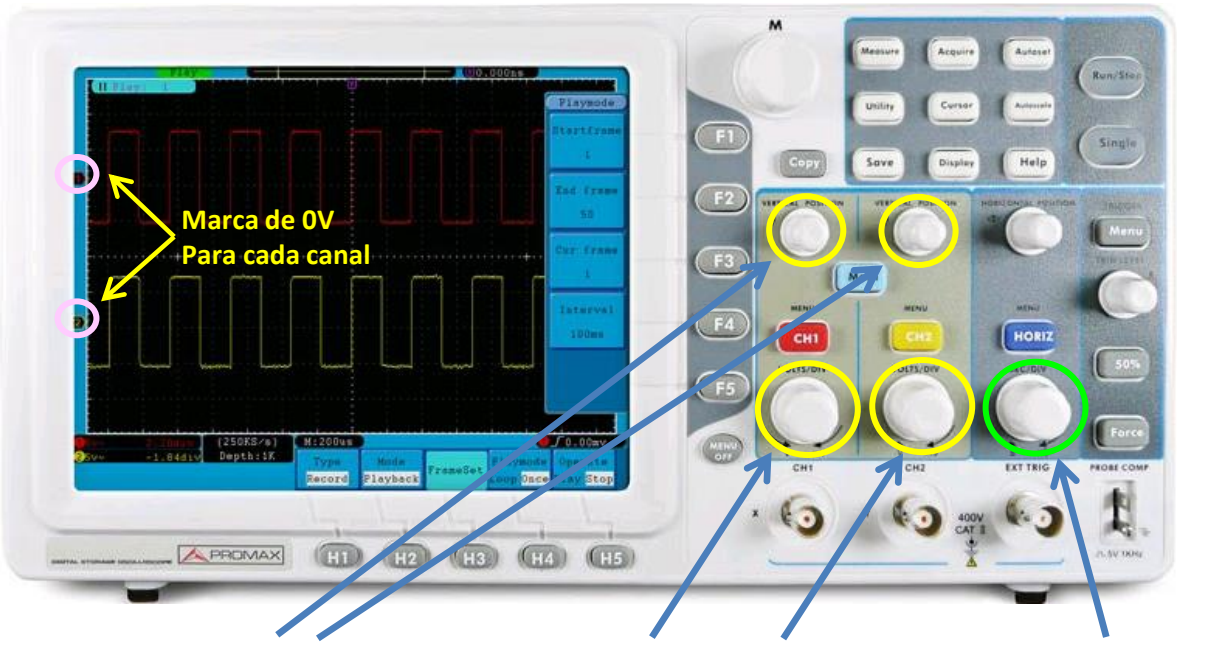
Instrumentación:

FUENTE DE ALIMENTACIÓN



GENERADOR DE FUNCIONES

El osciloscopio, manejo y utilidades:



El osciloscopio tiene dos canales para poder representar dos señales simultáneamente. Una conectada al canal 1 (CH1) y otra conectada al canal 2 (CH2). Ambas señales se presentan con el eje vertical en voltios y el eje horizontal en tiempo. La escala vertical de cada canal se puede controlar independientemente. El tiempo es común para ambas señales.

Los mandos de control principales se muestran en la figura. Si se selecciona por ejemplo una escala vertical de 50 mV/div eso significa que cada división vertical de la pantalla se corresponde con 50 mV. Lo mismo sucede en el eje de tiempos. Una selección de 200 ms/div significa que cada división horizontal de la pantalla representa 200 ms. Esto nos permite medir amplitudes, periodos y frecuencias de señales, así como diferencias de tiempos o de tensiones.

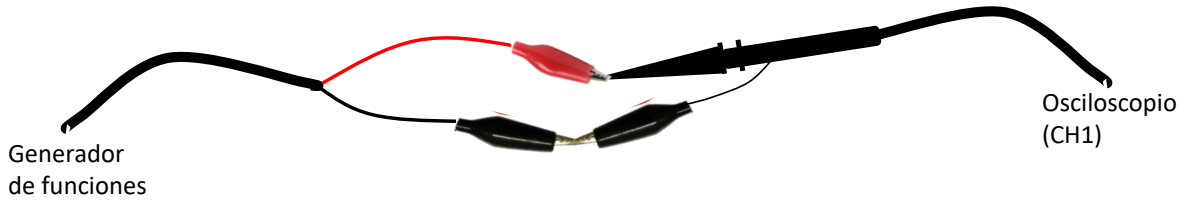
El botón **Autoset** (arriba a la derecha), permite sincronizar el osciloscopio inicialmente. Éste selecciona escalas de tensión y de tiempo para poder observar las señales de la forma más optimizada. En todo caso, en ocasiones es necesario modificar manualmente las escalas para ver las señales con precisión.

El botón **Measure** (arriba a la derecha), permite presentar en pantalla medidas que el osciloscopio realiza sobre las señales. Siguiendo los menús puede seleccionar medidas de periodo, frecuencia, tensión pico a pico (V_p), tensión media (V_m), etc.

Para medir con precisión, emplear cursores y medir diferencias de tiempo y tensión emplee la utilidad PROMAX en el ordenador del puesto.

Ejercicios sobre el manejo de los equipos de instrumentación:

Conecte el generador de funciones al osciloscopio (CH1). Cable rojo a la punta de la sonda y cable negro a la masa de la sonda:



Ejercicio 1:

1. Programe el generador de funciones para que genere una señal senoidal de 1 Vpp y 100 Hz.
2. Presione **Autoset** en el osciloscopio.
3. Modifique la escala horizontal y vertical hasta que observe un único periodo de la señal en la pantalla con la mayor amplitud que pueda conseguir.
4. A partir de esos datos obtenga la medida de amplitud, periodo y frecuencia de la señal.
5. Capture la señal con el programa PROMAX. Sitúe los cursores tanto en tensión como en tiempo para medir la amplitud y la frecuencia de la señal

Ejercicio 2:

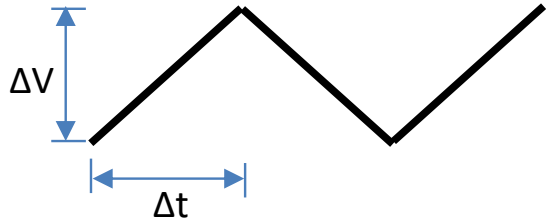
1. Programe el generador de funciones para que genere una señal senoidal de 1,5 Vpp y 10 Hz.
2. Presione **Autoset** en el osciloscopio.
3. Modifique la escala horizontal y vertical hasta que observe un único periodo de la señal en la pantalla con la mayor amplitud que pueda conseguir.
4. A partir de esos datos obtenga la medida de amplitud, periodo y frecuencia de la señal.
5. Capture la señal con el programa PROMAX. Sitúe los cursores tanto en tensión como en tiempo para medir la amplitud y la frecuencia de la señal.

Ejercicio 3:

1. Programe el generador de funciones para que genere una señal senoidal de 1 Vpp, 1 kHz y un offset de 1V.
2. Presione **Autoset** en el osciloscopio.
3. Presione **Measure** en el osciloscopio y siga los menús para que el instrumento indique en la pantalla la tensión media (V_m) y la frecuencia de la señal.
4. Compruebe que los valores medidos son correctos modificando las escalas horizontal y vertical. La tensión media V_m deberá ser 1V.
5. Capture la señal con el programa PROMAX. Sitúe los cursores tanto en tensión como en tiempo para medir la amplitud, tensión media y la frecuencia de la señal

Ejercicio 4:

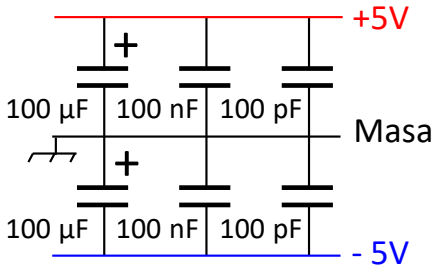
1. Programe el generador de funciones para que genere una señal triangular de 1 Vpp de amplitud, 10 kHz y SIN OFFSET.
2. Presione **Autoset** en el osciloscopio.
3. Modifique la escala horizontal y vertical hasta que observe un único periodo de la señal en la pantalla con la mayor amplitud que pueda conseguir.
4. Capture la señal con el programa PROMAX. Sitúe los cursores tanto en tensión como en tiempo para medir la diferencia de tensión entre los dos extremos de la rampa de subida y la diferencia de tiempo también entre dichos extremos. A partir de estos datos obtenga la pendiente de la rampa en V/s.



$$pte = \frac{\Delta V}{\Delta t}$$

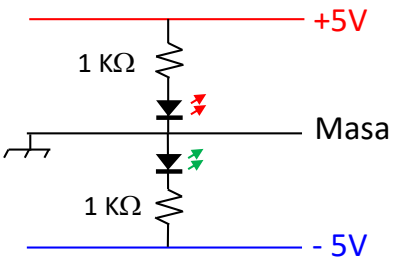
Instalación de condensadores de desacoplo:

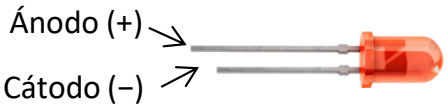
Las fuentes de alimentación deben suministrar una tensión continua estable de +5 y -5 V. Sin embargo, a través de estas líneas se introducen ruidos procedentes principalmente de los circuitos de conmutación (dichos circuitos absorben picos de corriente en los flancos activos del reloj produciendo fluctuaciones en la alimentación durante dichos flancos debidas a la componente inductiva de los cables y las conexiones). Estos ruidos deben ser filtrados para obtener una tensión continua limpia y estable. Para ello se emplean los llamados condensadores de desacoplo. Un condensador es un dispositivo capaz de proporcionar corriente con una variación muy pequeña de su tensión en bornas. Se colocarán tres condensadores de **100 µF** (electrólitico)**, 100 nF** (de plástico) **y 100 pF** (cerámico) en paralelo. Cada uno de ellos apropiado para un intervalo diferente de frecuencias. Tenga cuidado con la polaridad de los condensadores electrolíticos.



Instalación de LEDs para detección de posibles cortocircuitos:

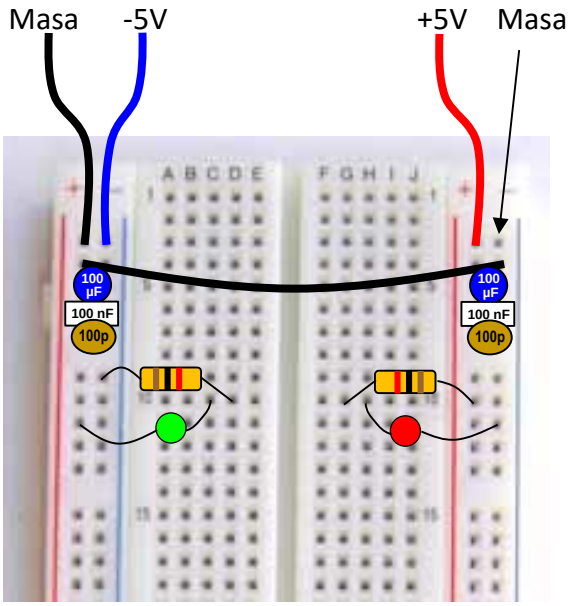
Es conveniente situar dos LEDs, uno alimentado a +5 V (rojo) y el otro alimentado a −5 V (verde) para detectar la presencia de cortocircuitos. En caso de producirse alguno, se apagará el LED correspondiente a la alimentación cortocircuitada alertando de este hecho. Tenga en cuenta que en un LED el terminal más largo es el ánodo (terminal positivo).

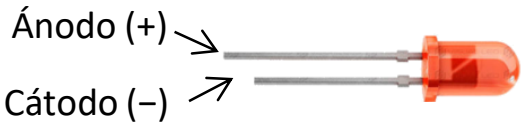




Ejemplo de posible montaje:

El siguiente montaje es un ejemplo de colocación de los condensadores de desacoplo y los LEDs de detección de cortocircuitos.





Trate de sacar provecho de esta sesión para entender los ejemplos y familiarizarse lo más posible con el entorno de trabajo que va a utilizar durante el curso. Puede emplear también tiempo de las próximas sesiones.

Iniciación a la parte digital:

La parte digital se realizará mediante síntesis en la FPGA Spartan 3E de la tarjeta BASYS2 que encontrará en el laboratorio. Todo el hardware digital se describirá obligatoriamente en VHDL empleando el entorno ISE Webpack 14.7 instalado en los ordenadores del laboratorio.

El tutorial: "**Manual de referencia de la tarjeta BASYS2**" es muy útil para familiarizarse con el entorno del laboratorio. Presenta una serie de ejemplos que muestran el funcionamiento de los diferentes recursos de la tarjeta. Dedique tiempo a leer y entender los ejemplos.

- Arranque el entorno ISE, compile y sintetice sobre la FPGA el ejemplo 1 contenido en el "Manual de referencia de la tarjeta BASYS 2".
- Realice los ejercicios propuestos.
- Compile y sintetice sobre la FPGA el ejemplo 2.
- Realice los ejercicios propuestos.
- Compile y sintetice sobre la FPGA el ejemplo 3.
- Realice los ejercicios propuestos.
- Compile y sintetice sobre la FPGA el ejemplo 4.
- Realice los ejercicios propuestos.

Continuación del entrenamiento con la placa de desarrollo BASYS 2:

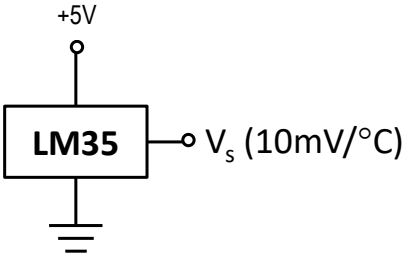
Vamos a centrarnos ahora en el diseño digital síncrono y en el uso de las entradas y salidas externas de la tarjeta BASYS2.

- Lea y entienda el capítulo "Diseño de dircuitos digitales síncronos con VHDL", página 14 del Manual de Referencia. **Todos los circuitos digitales de esta práctica deberán realizarse con diseño síncrono**.
- Lea y entienda el capítulo dedicado a la utilización de las entradas y salidas externas del entorno de desarrollo, página 33 del Manual de Referencia.
- Arranque el entorno ISE, compile y sintetice sobre la FPGA el ejemplo 5 contenido en el "Manual de referencia de la tarjeta BASYS 2".
- Realice los ejercicios propuestos.
- Compile y sintetice sobre la FPGA el ejemplo 6.
- Realice los ejercicios propuestos.

Trate de sacar provecho de esta sesión para entender los ejemplos y familiarizarse lo más posible con el entorno de trabajo que va a utilizar durante el curso. Puede emplear también tiempo de las próximas sesiones para entender bien estos ejemplos.

MÓDULO 1: El sensor de temperatura LM35

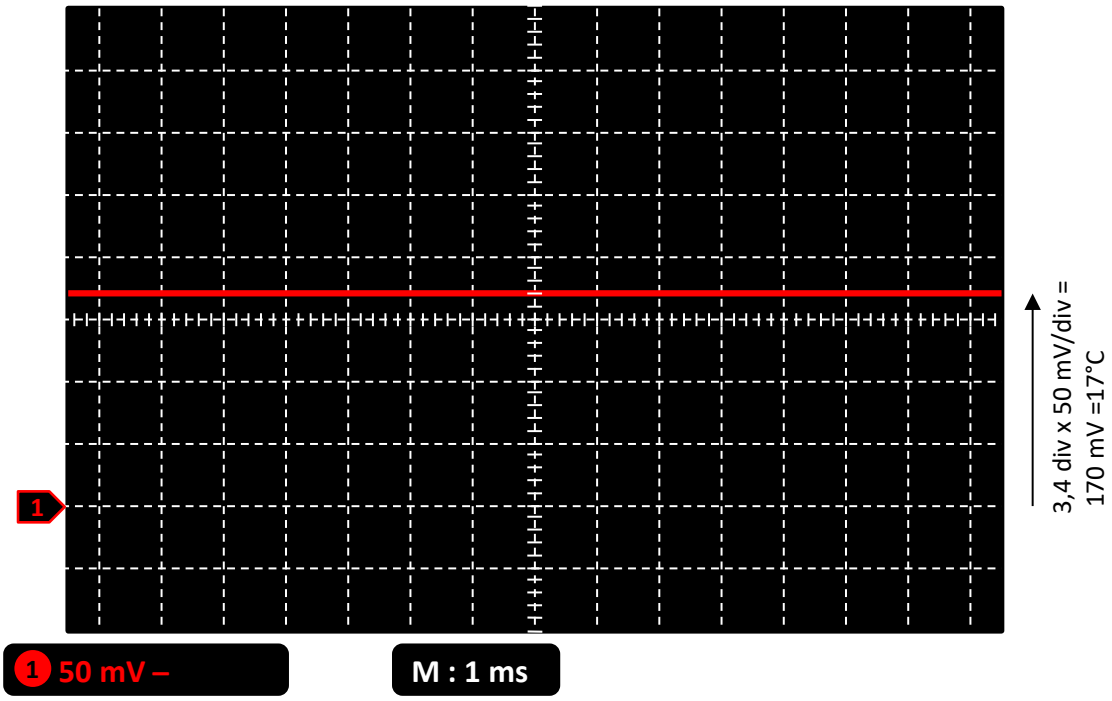
El sensor de temperatura LM35 es un circuito integrado de 3 patillas. En dos de ellas se debe introducir la alimentación (masa y Vcc) y en la tercera se genera una tensión proporcional a la temperatura con una precisión de 10 mV/°C. El integrado admite tensiones de alimentación entre 3 y 5,5V. Nosotros emplearemos Vcc=+5V.



Ajuste de la etapa y medidas:

1. Monte el LM35 en la placa protoboard y conecte los terminales de alimentación a +5V y masa (GND).

2. Utilice el osciloscopio y mida la tensión en el terminal de salida (terminal central). Emplee una base de tiempos rápida (1 ms/div), una escala vertical de 50mV/div y un nivel de 0 tres divisiones por debajo de la línea horizontal central. (Puede emplear además las herramientas de medida de tensión del osciloscopio. En el menú "Measure" puede hacer que aparezca el valor medio de la tensión en la pantalla del osciloscopio). Debería leerse una tensión proporcional a la temperatura del laboratorio con la precisión de 10mV/°C. En el siguiente ejemplo se leería una tensión de 170 mV indicando que hay 17 °C de temperatura ambiente.



3. Coloque los dedos sobre el integrado de modo que se caliente. Observe que la tensión leída en el osciloscopio aumenta y espere a que se estabilice. Retire los dedos y observe que la tensión vuelve a bajar al valor leído en el punto 2.

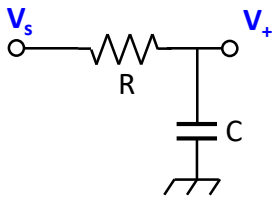
DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

1. Adjunte una captura de pantalla del osciloscopio donde se lea la tensión medida correspondiente a la temperatura de la sala. (También puede emplear las herramientas de medida de tensión del osciloscopio. En el menú "Measure" puede hacer que aparezca el valor medio (V_m) de la tensión en la pantalla del osciloscopio).
2. Adjunte una captura de pantalla del osciloscopio donde se lea la tensión máxima obtenida una vez calentado el sensor con los dedos.

Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

MÓDULO 2: Filtro paso bajo

Este filtro se emplea para atenuar las posibles componentes de ruido que puedan aparecer debido a fuentes externas como interferencias de los equipos del laboratorio, radiación de la red eléctrica, etc. Se diseñará de modo que su frecuencia de corte (f_c) sea muy baja (10 Hz) ya que la magnitud que queremos medir (la temperatura) no tiene grandes variaciones con el tiempo:



Diseño:

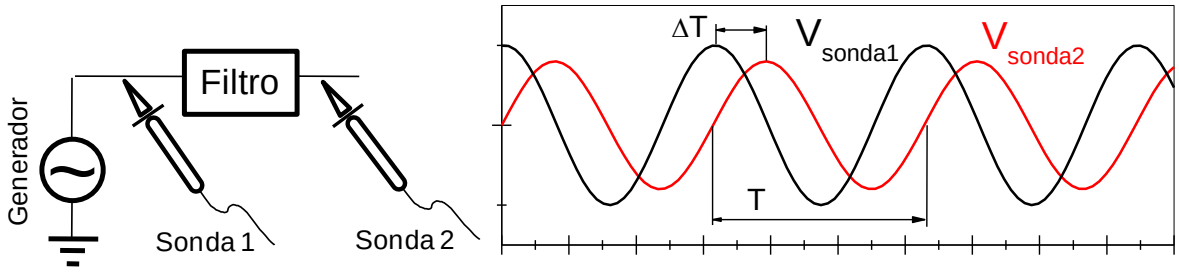
1. Obtenga la expresión de la función de transferencia para este filtro y exprésela en la forma siguiente:

$$H(jf) = \frac{v_+}{v_s} = \frac{1}{\left(1 + j\frac{f}{f_p}\right)}$$

2. En un filtro de orden 1 como éste, la frecuencia del polo (f_p) coincide con la frecuencia de corte (f_c). Diseñe los valores de R y C para que $f_p = f_c = 10$ Hz. Debe elegir un valor para uno de los componentes y calcular el otro. No utilice resistencias mayores de 47 kΩ ni condensadores mayores de 10 µF.
3. Dibuje el diagrama asintótico de Bode en módulo y fase teniendo en cuenta la frecuencia del polo. Emplee para ello las plantillas que se adjuntan en la página 26 o en la memoria. Tenga en cuenta que la fase del filtro debe ser siempre negativa para que el filtro sea causal. (**Tiene disponible en MOODLE un documento explicativo sobre el trazado de diagramas de Bode**).

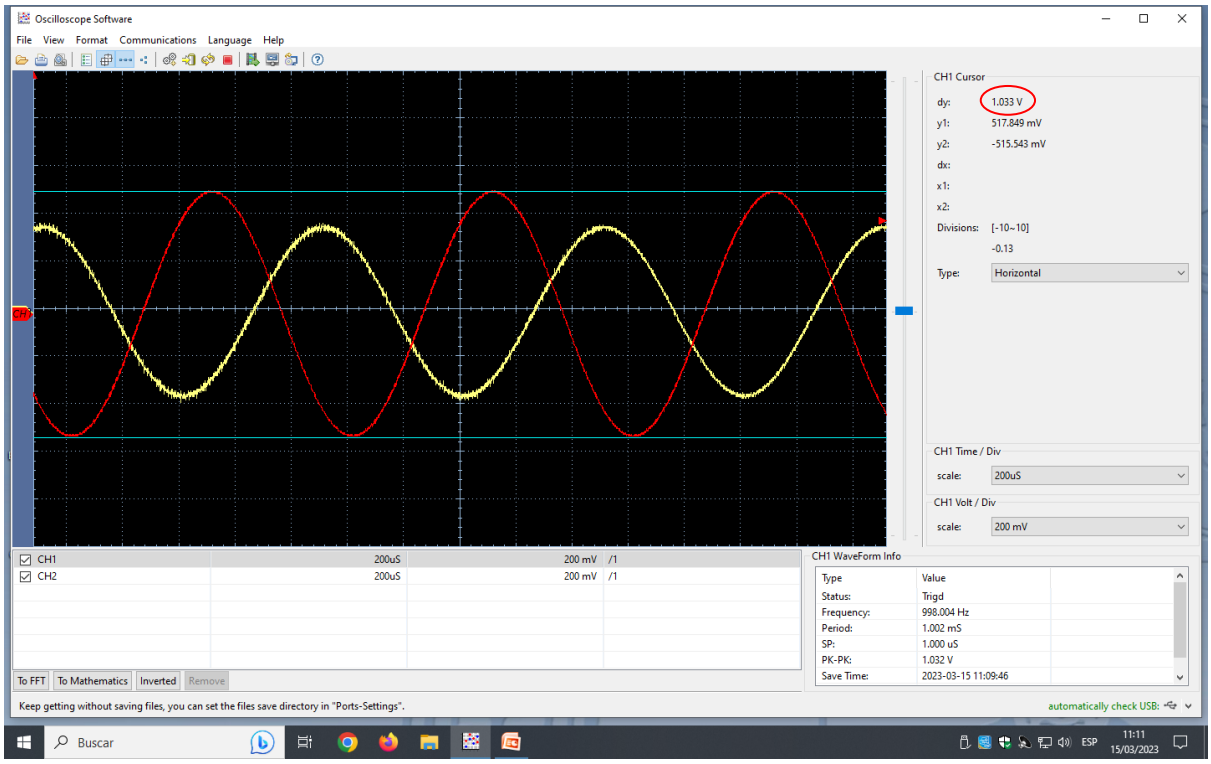
Ajuste de la etapa y medidas:

1. Monte la etapa en la placa protoboard. De momento no la conecte a la etapa anterior.
2. Debe caracterizar el filtro. Para ello mida su función de transferencia según se indica a continuación:
   - Asegúrese que la etapa está aislada (desconectada de las etapas anterior y posterior).
   - Mida la salida del generador de funciones en vacío (sin cargar con el filtro). Ajuste la amplitud para observar una sinusoide de 0,5 Vp de amplitud (1 Vpp).
   - Conecte el generador de funciones a la entrada del filtro (V_s) y vaya variando la frecuencia del generador (periodo T) para obtener la función de transferencia (módulo y fase) del filtro, según las expresiones que se indican a continuación.



$$|H(j\omega)| = \frac{V_{sonda2}}{V_{sonda1}} \qquad |H(j\omega)|[dB] = 20 \cdot log\left(\frac{V_{sonda2}}{V_{sonda1}}\right) \qquad \Phi[H(j\omega)] = -\frac{\Delta T}{T} \cdot 360°$$

Puede emplear la aplicación PROMAX en el ordenador, que permite capturar las formas de onda en la pantalla y hacer cálculos de retardos y amplitudes. Se muestra a continuación un ejemplo de medida la tensión pico a pico de la señal de entrada (rojo) empleando esta aplicación:



3. Mida todos los puntos indicados en la tabla de la página 25. (Para señales de salida muy débiles puede aumentar el nivel de entrada mientras no se sature la salida).
4. Mida con precisión la frecuencia de corte (f_c) del filtro (frecuencia donde la ganancia respecto a la banda de paso es igual a −3 dB). Para ello busque el valor de la frecuencia donde se cumple exactamente $20 \cdot \log(V_+/V_s) = -3$.
5. Conecte ahora esta etapa a la salida de la etapa anterior (LM35). Asegúrese de que las tensiones medidas a la entrada y a la salida de esta etapa son iguales.

DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

1. Detalle el cálculo de la función de transferencia del filtro paso bajo.
2. Detalle el cálculo de los valores R y C.
3. Dibuje los diagramas asintóticos de Bode en módulo y fase del filtro empleando las plantillas de las páginas siguientes. Indique la frecuencia del polo, la pendiente de la atenuación, el máximo valor de la ganancia y la pendiente de la fase.
4. Rellene la tabla para la caracterización del filtro e inclúyala.
5. Superponga los puntos de las medidas sobre el diagrama asintótico de Bode de modo que se vean las gráficas de función de transferencia de ganancia y fase.
6. Indique el valor de la frecuencia de corte (f_c) medida con precisión.
7. Incluya una captura de pantalla del osciloscopio donde se aprecie la señal a la salida del filtro paso bajo (punto v_+) en el canal 1 y a la entrada (punto v_s) en el canal 2 cuando éste se encuentra alimentado con la frecuencia de corte exacta.

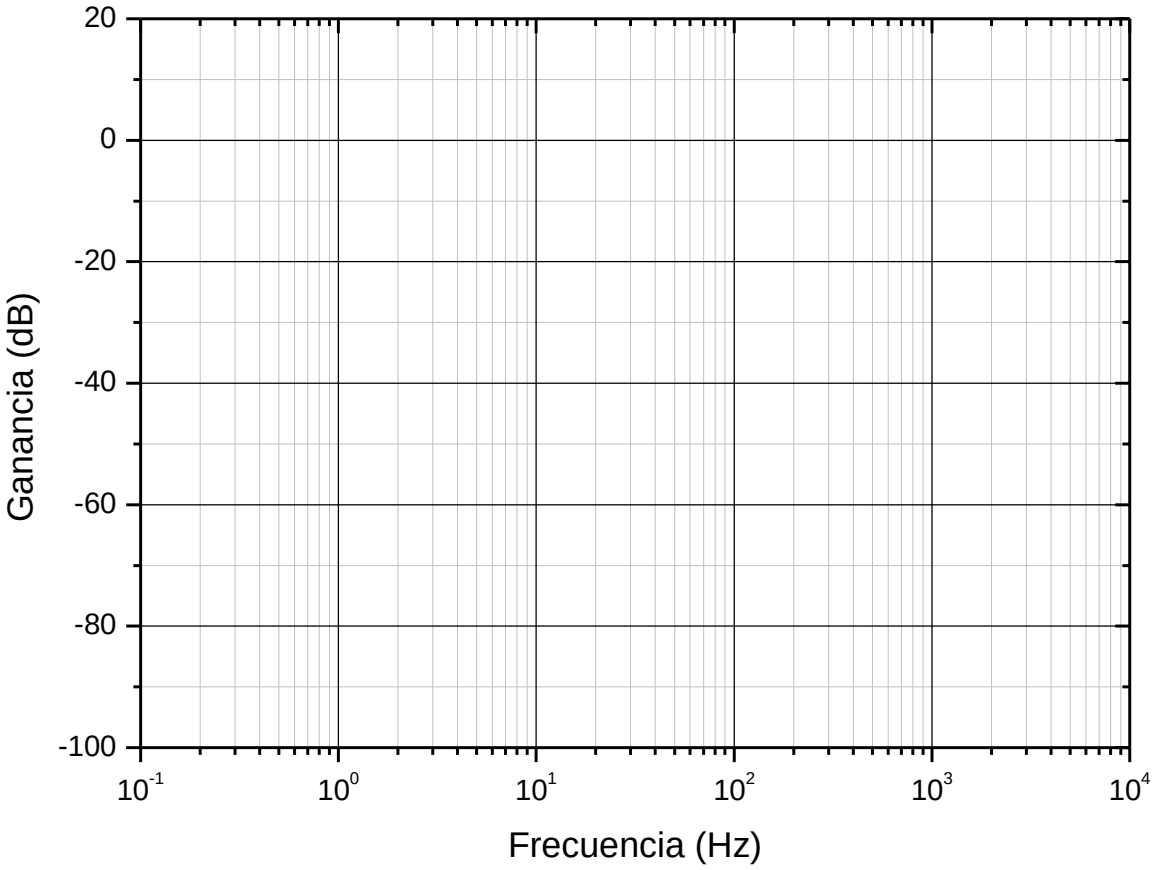
Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

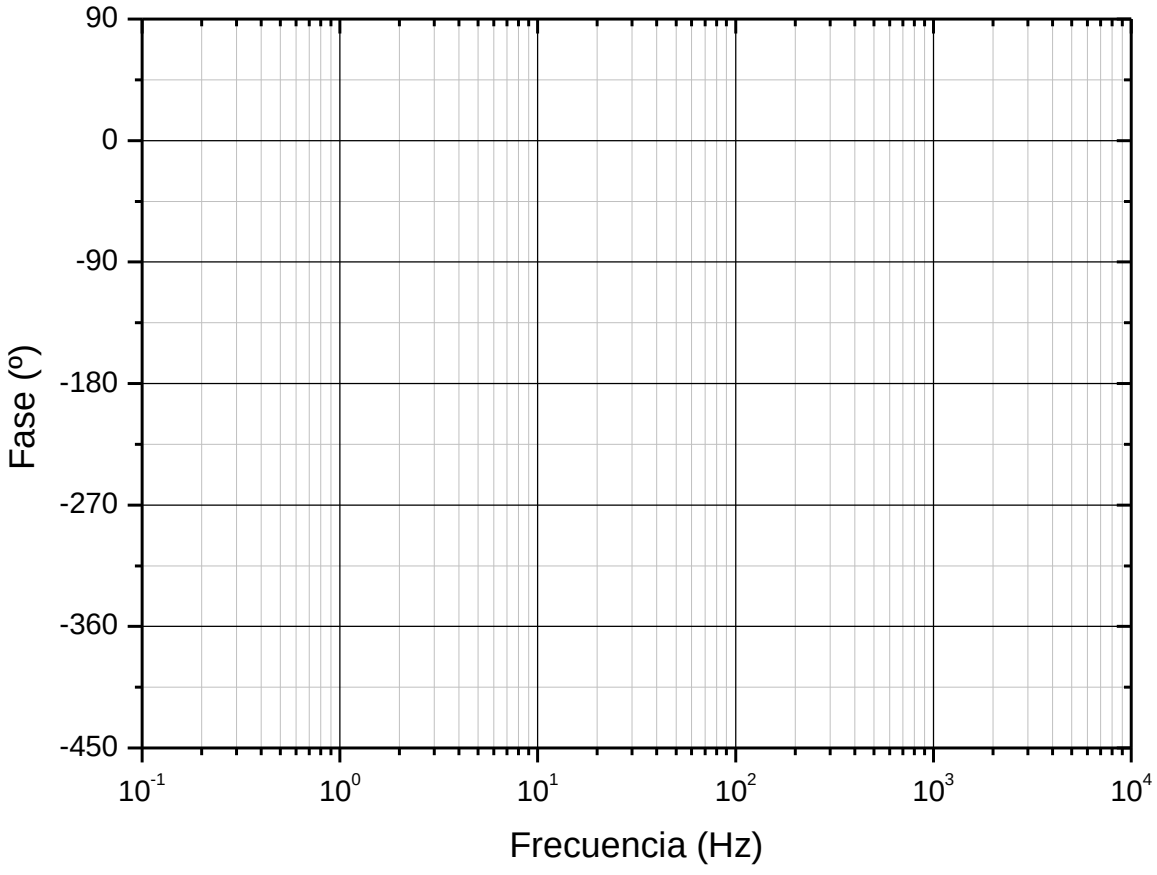
Tabla para la medida del filtro paso bajo

Frecuencia (Hz)	Entrada (Vpp)	Salida (Vpp)	Ganancia	Ganancia (dB)	ΔT (s)	Fase (º)
0,5						
1						
3						
5						
7						
8						
9						
10						
11						
12						
13						
14						
15						
17						
20						
30						
50						
70						
100						
200						
300						
500						
700						
1.000						

Frecuencia de corte (f_c) exacta medida	

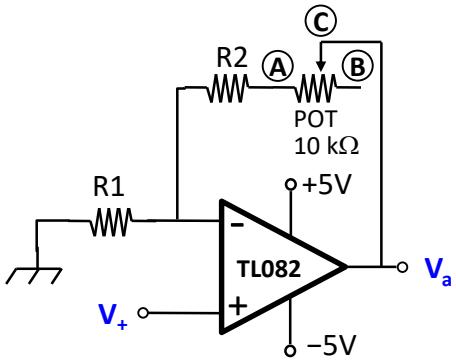
DIAGRAMAS ASINTÓTICOS DE BODE Y MEDIDAS, FILTRO PASO BAJO





MÓDULO 3: Amplificador

Vamos a construir un amplificador de ganancia 10. En este caso, necesitamos que la ganancia sea muy precisa para que el factor de conversión de nuestro termómetro sea exactamente de 100 mV/°C. Para ello emplearemos una configuración de amplificador no inversor con un potenciómetro de 10 kΩ que permita ajustar su ganancia. Consulte el apéndice I para entender el funcionamiento de un potenciómetro.



No conectado

Diseño:

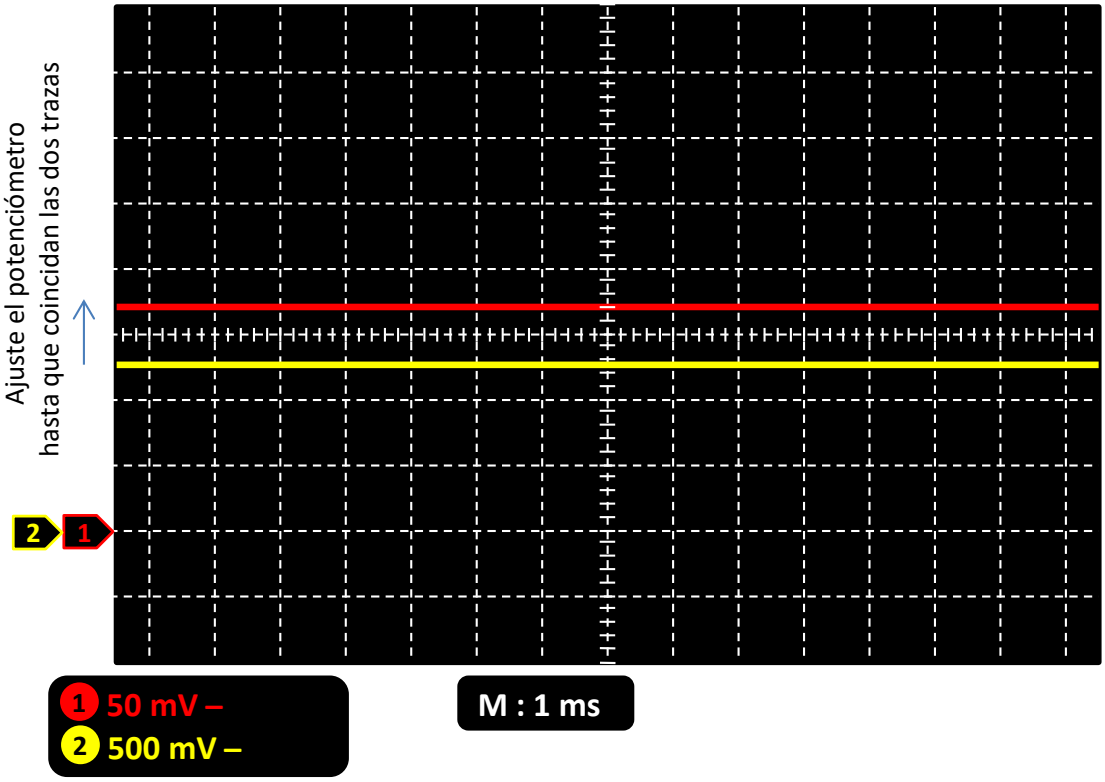
1. Queremos tener precisión en el ajuste, por lo tanto vamos a diseñar las resistencias R1 y R2 para que la ganancia pueda ajustarse entre 8 y 12 en los dos extremos del cursor del potenciómetro de 10 kΩ, que aquí actúa como resistencia variable. Esta ganancia se expresa como:

$$\frac{V_a}{V_+} = 1 + \frac{R2 + POT}{R1}$$

2. Teniendo en cuenta los dos valores de resistencia del potenciómetro para los extremos del cursor, calcule los valores de las resistencias R1 y R2 de modo que las ganancias mínima y máxima sean de 8 y 12 respectivamente.

Ajuste de la etapa:

1. Monte la etapa sobre la placa protoboard. Conéctela a la etapa anterior (filtro paso bajo) de modo que se encuentren conectadas las tres primeras etapas: sensor de temperatura (LM35), filtro paso bajo y amplificador.
2. Mida simultáneamente la tensión a la entrada y a la salida de este amplificador empleando los dos canales del osciloscopio (Canal 1: entrada y canal 2: salida). Utilice una base de tiempos rápida (1 ms/div), emplee una escala de 50 mV/div en el canal 1 y una escala de 500 mV/div en el canal 2. Sitúe los niveles de 0 de los dos canales a tres divisiones por debajo de la línea horizontal central.
3. **Ajuste el cursor del potenciómetro hasta que las dos trazas (roja y amarilla) coincidan en la pantalla**. En ese momento el nivel de tensión a la salida es 10 veces el nivel de tensión a la entrada.



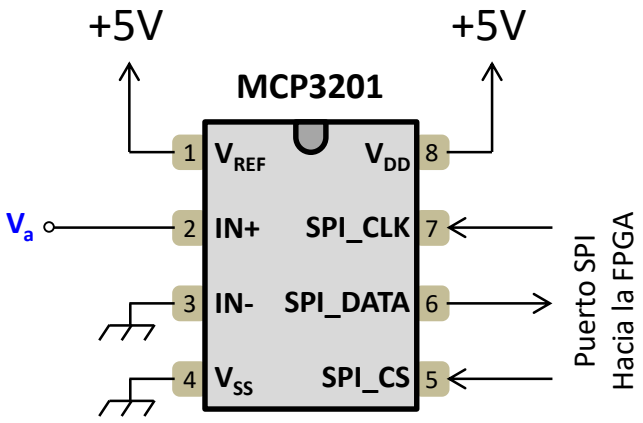
DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

1. Detalle el cálculo de las resistencias R1 y R2.
2. Incluya una captura de pantalla del osciloscopio donde se aprecie sólo la señal a la entrada del amplificador con la escala de 50 mV/div (Puede emplear las herramientas de medida de tensión del osciloscopio. En el menú "Measure" puede hacer que aparezca el valor medio de la tensión en la pantalla del osciloscopio).
3. Incluya una captura de pantalla del osciloscopio donde se aprecie sólo la señal a la salida del amplificador con la escala de 500 mV/div (Puede emplear las herramientas de medida de tensión del osciloscopio. En el menú "Measure" puede hacer que aparezca el valor medio de la tensión en la pantalla del osciloscopio).

Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

MÓDULO 4: Convertidor analógico-digital MCP3201

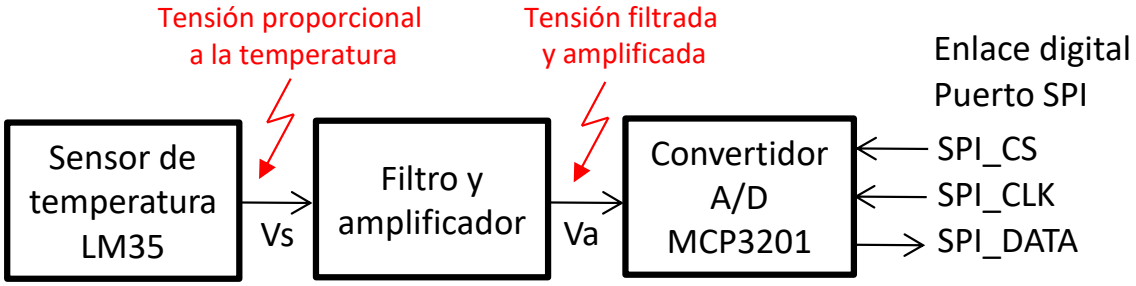
El convertidor analógico-digital es un circuito que produce un valor digital binario proporcional a la tensión que tiene a su entrada. En nuestro caso emplearemos un MCP3201 que permite ser alimentado mediante tensión asimétrica (0 y +5V) y emplearemos una tensión de referencia (V_{REF}) también de +5V.



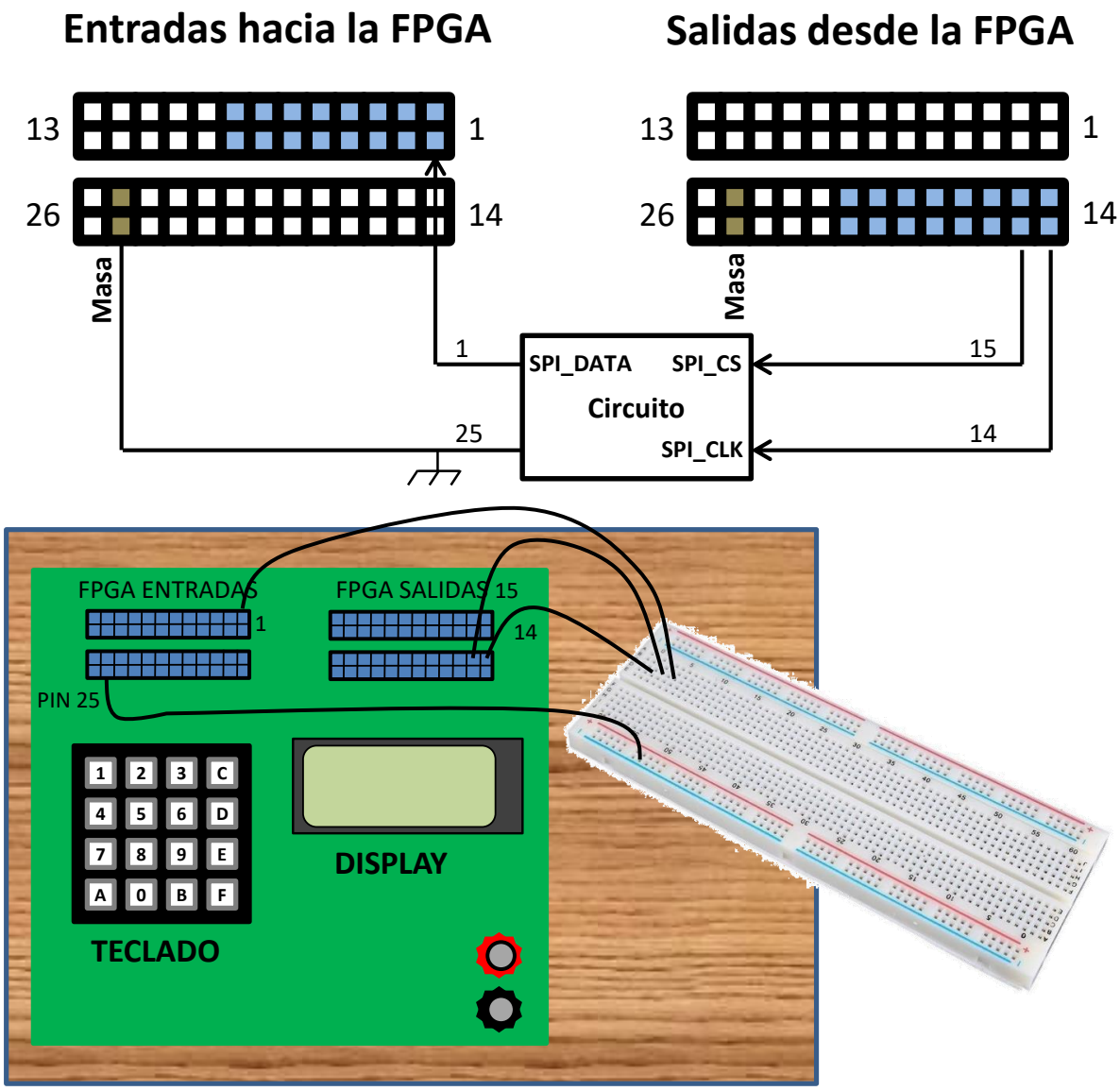
V_{DD} y V_{SS} son las tensiones de alimentación, V_{REF} es la tensión de referencia, IN+ e IN- permiten medir una tensión diferencial [V(IN+)-V(IN-)]. Nosotros conectaremos el terminal IN- a masa para medir la tensión Va respecto a masa.

Diseño:

1. Monte la etapa sobre la placa protoboard. Realice las conexiones de V_{DD} , V_{SS} , V_{REF} , IN+ e IN-. Conecte todos los módulos anteriores (LM35, filtro paso bajo y amplificador) y fíjese que IN+ debe conectarse a la salida de la etapa anterior (amplificador).



2. A continuación conecte los terminales del puerto SPI: SPI_CLK, SPI_DATA, SPI_CS y masa (4 cables) al tablero de madera del puesto según el siguiente esquema:



Ajuste de la etapa y medidas:

1. Ahora descargue desde MOODLE el archivo **termometro.bit.** Este archivo sintetiza un termómetro idéntico al que tiene que realizar y permite hacer pruebas sobre la placa protoboard. Cargue el archivo **termometro.bit** en la placa BASYS2 empleando el software ADEPT.
2. Si todo es correcto deberá aparecer el valor de la temperatura ambiente en los displays de la FPGA. Asegúrese de que la temperatura leída tiene sentido. Caliente el sensor con los dedos para ver que la temperatura aumenta y que todo funciona correctamente.

Los LEDs verdes de la placa BASYS2 parpadearán indicando que el hardware digital se trata de la versión de prueba proporcionada por los profesores.

DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

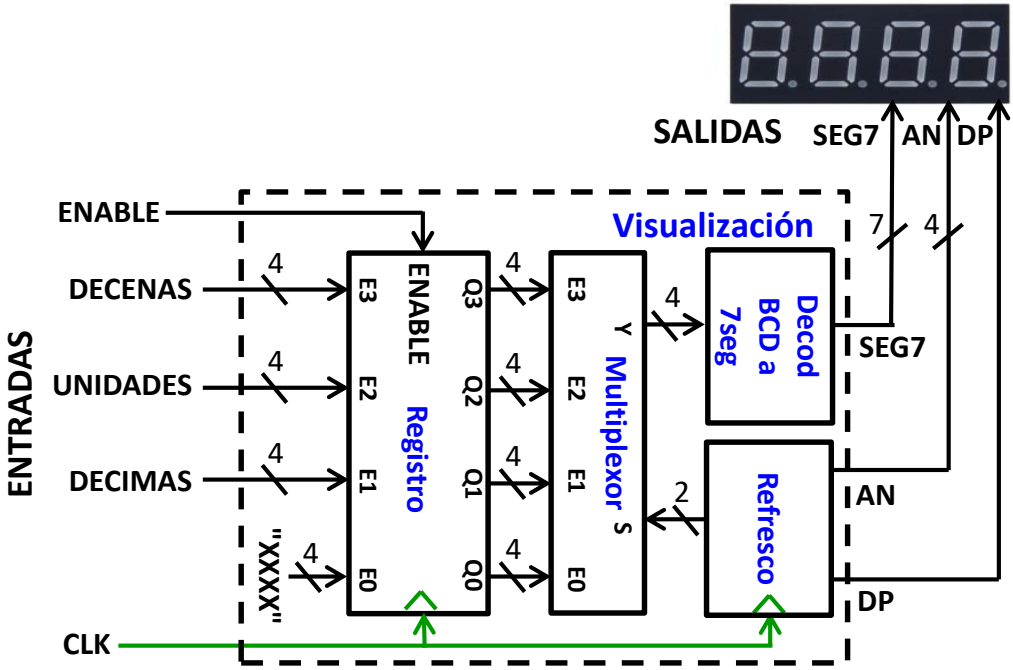
1. Incluya una fotografía del circuito conectado a la FPGA donde se vean las conexiones y se lea la temperatura ambiente indicada en los displays.
2. A continuación caliente el sensor con los dedos e incluya otra fotografía donde se vean las conexiones y se lea la temperatura en los displays.

MÓDULO 5: Módulo de visualización

El módulo de visualización debe capturar en un registro el valor BCD de las DECENAS. UNIDADES y DECIMAS de la temperatura cuando la señal ENABLE está activa a nivel alto. La captura se realiza en flanco de subida del reloj CLK. Además debe representar estas cifras en los tres primeros displays, activando el punto decimal en el segundo display de la izquierda y presentando una "C" en el display de más a la derecha (para indicar que se tata de grados Celsius).

Para implementar este módulo revise los ejemplos 3 y 4 del "Manual de referencia de la tarjeta BASYS2".

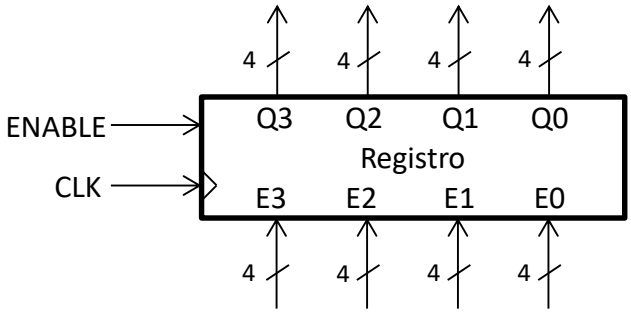
El esquema general de este módulo se puede ver en la figura siguiente:



5.1 Visualización: Registro

El registro permite capturar los valores en sus entradas E0, E1, E2 y E3 cuando la señal ENABLE es activa a nivel alto y en flanco de subida del reloj (CLK). Cuando ENABLE='0' el registro se encuentra en modo inactivo. Cuando ENABLE='1' el registro debe capturar los datos y presentarlos en en las salidas Q0, Q1, Q2 y Q3.

El esquema de este módulo y parte de su descripción en VHDL se muestran a continuación. Complete su descripción funcional.

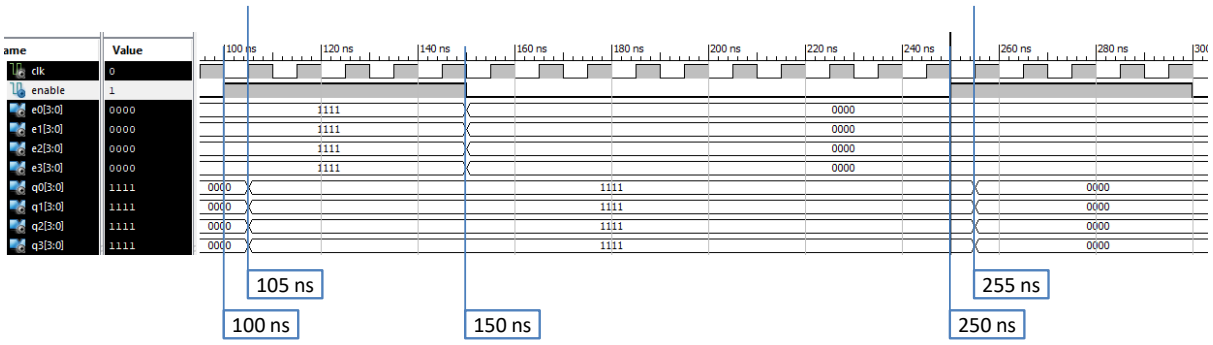


ARCHIVO: registro.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity registro is
    Port ( CLK     : in   STD_LOGIC;                        -- entrada de reloj
           ENABLE  : in   STD_LOGIC;                        -- enable
           E0      : in   STD_LOGIC_VECTOR (3 downto 0);    -- entrada E0
           E1      : in   STD_LOGIC_VECTOR (3 downto 0);    -- entrada E1
           E2      : in   STD_LOGIC_VECTOR (3 downto 0);    -- entrada E2
           E3      : in   STD_LOGIC_VECTOR (3 downto 0);    -- entrada E3
           Q0      : out  STD_LOGIC_VECTOR (3 downto 0);    -- salida Q0
           Q1      : out  STD_LOGIC_VECTOR (3 downto 0);    -- salida Q1
           Q2      : out  STD_LOGIC_VECTOR (3 downto 0);    -- salida Q2
           Q3      : out  STD_LOGIC_VECTOR (3 downto 0));   -- salida Q3
end registro;

architecture a_registro of registro is

signal QS0 : STD_LOGIC_VECTOR (3 downto 0) :="0000"; -- señal que almacena el valor de Q0
signal QS1 : STD_LOGIC_VECTOR (3 downto 0) :="0000"; -- señal que almacena el valor de Q1
signal QS2 : STD_LOGIC_VECTOR (3 downto 0) :="0000"; -- señal que almacena el valor de Q2
signal QS3 : STD_LOGIC_VECTOR (3 downto 0) :="0000"; -- señal que almacena el valor de Q3

begin
  process (CLK)
    begin
      if (CLK'event and CLK='1') then   -- con cada flanco activo

         -- Realizar el proceso de captura

      end if;
  end process;

  -- Cableado de las salidas

end a_registro;
```

Pruebas del registro:

1. Descargue el archivo de test_bench: **tb_registro.vhd** y realice la simulación. Las entradas y salidas deberán comportarse como se muestra en la gráfica de simulación de la página siguiente, cumpliendo los tiempos y los valores digitales en cada caso:



5.2 Visualización: Decodificador de BCD a 7 segmentos

Para visualizar la imagen correspondiente a cada código debe crearse un decodificador que asocie un código BCD expresado con 4 bits a un vector de 7 bits que indique los segmentos que deben encenderse. **Este módulo se proporciona ya codificado para ahorrar tiempo y evitar errores**, ya que no supone mayor dificultad. El módulo proporcionado también enciende los segmentos correspondientes a los caracteres "A" "B" "C" "D" "E" y "F" para los valores 1010, 1011, 1100, 1101, 1110 y 1111 respectivamente. De esta forma puede emplear el valor binario 1100 para representar la "C". Descargue el fichero **decodBCDa7s.vhd** que contiene dicho decodificador.

Las entradas y salidas asociadas al mismo se resumen a continuación:

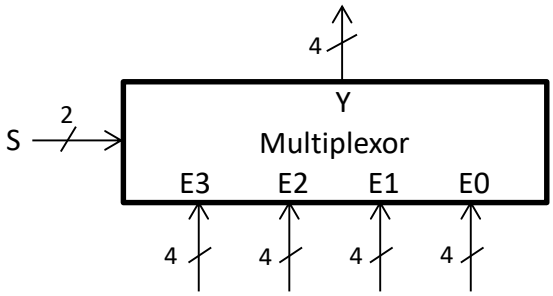
ARCHIVO: decodBCDa7s.vhd (Puede descargar este archivo desde MOODLE)

```
entity decodBCDa7s is
    Port ( BCD : in  STD_LOGIC_VECTOR (3 downto 0);   -- Entrada del valor BCD
           SEGMENTOS : out  STD_LOGIC_VECTOR (0 to 6));   -- Salidas al display
end decodBCDa7s;
```

5.3 Visualización: Multiplexor

Este es el elemento que permite visualizar caracteres diferentes en cada display. Tenga en cuenta que los 4 displays presentes en la BASYS2 comparten las mismas líneas CA,CB,CC,CD,CE,CF,CG, aunque poseen señales de activación diferentes (AN). Por tanto para visualizar cifras diferentes en cada uno, es necesario presentar cada una de ellas por separado de forma muy rápida para que el ojo no perciba el parpadeo. (Remitimos al lector a los ejemplos 3 y 4 del "Manual de referencia de la tarjeta BASYS 2" donde se puede encontrar una descripción detallada).

En este caso necesitamos un multiplexor de 4 entradas de datos de 4 bits, una entrada de control de 2 bits y una salida también de 4 bits. El esquema de este módulo y parte de su descripción en VHDL se muestran a continuación. Complete su descripción funcional.

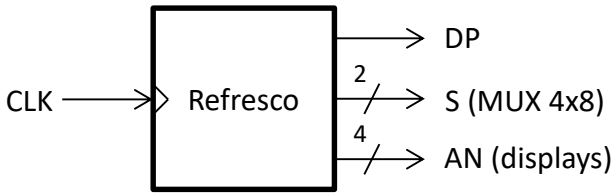


ARCHIVO: MUX4x4.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity MUX4x4 is
    Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 0
           E1 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 1
           E2 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 2
           E3 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 3
           S  : in  STD_LOGIC_VECTOR (1 downto 0);   -- Señal de control
           Y  : out  STD_LOGIC_VECTOR (3 downto 0)); -- Salida
end MUX4x4;
```

5.4 Visualización: Refresco

Este bloque controla la visualización sucesiva de cada carácter en cada display y el encendido del punto decimal. Para ello va alternando las entradas de control del multiplexor con la activación del display correspondiente. Cuando se encuentra seleccionado el segundo display por la izquierda también enciende el punto decimal. Como entrada toma la señal de reloj de 1 kHz (CLK) y como salidas: la señal de selección del multiplexor (S), la señal de selección del punto decimal (DP) y las señales de activación de los displays (AN). El esquema de este módulo y parte de su descripción en VHDL se muestran a continuación. Complete su descripción funcional (fíjese en los ejemplos 3 y 4 del "manual de referencia de la tarjeta BASYS 2").



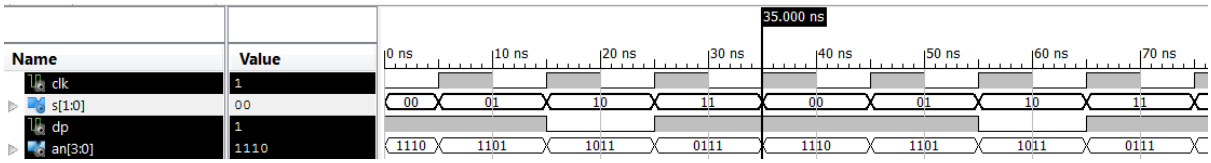
ARCHIVO: refresco.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity refresco is
    Port ( CLK : in  STD_LOGIC;                          -- reloj de refresco
           S   : out  STD_LOGIC_VECTOR (1 downto 0);     -- Control para el mux
           DP  : out STD_LOGIC;                          -- Control del punto decimal
           AN  : out  STD_LOGIC_VECTOR (3 downto 0)); -- Control displays individuales
end refresco;

architecture a_refresco of refresco is
signal QS : STD_LOGIC_VECTOR (1 downto 0):="00";
```

```
begin
  process (CLK)
    begin
        if (CLK'event and CLK='1') then

          -- Parte secuencial, completar.

        end if;
    end process;

-- Parte combinacional

-- completar cableado de:
--    S
--    DP
--    AN

end a_refresco;
```

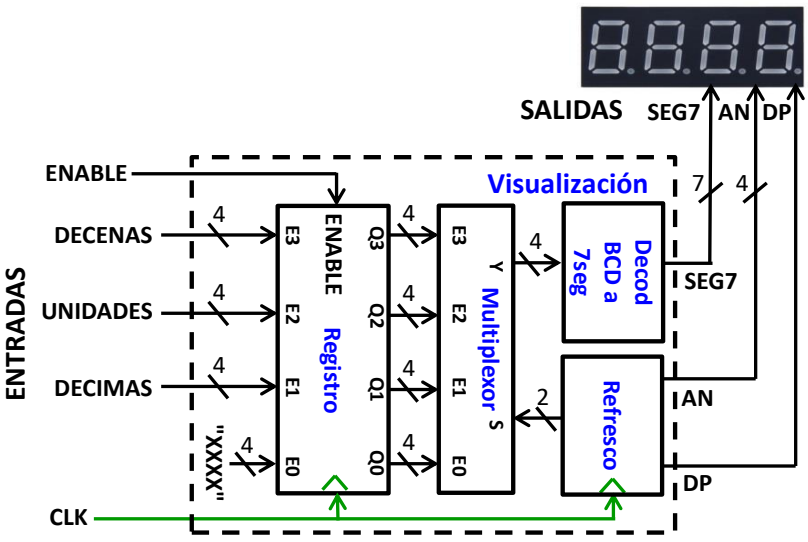
Pruebas del circuito de refresco:

1. Descargue el archivo de test_bench: **tb_refresco.vhd** y realice la simulación. Las salidas deberán comportarse como se muestra en la siguiente gráfica, cumpliendo los tiempos y los valores que se indican:



7.5 Visualización: Cableado final y pruebas

Finalmente es necesario cablear todos los elementos anteriores en un solo módulo (visualizacion.vhd). Para ello realizaremos un código VHDL con una descripción arquitectural de interconexión. Deberán declararse todos los módulos anteriores como **component** y realizar el cableado entre ellos.



Recuerde que el display de la derecha debe presentar una "C" y que esta representación debe hacerse dentro de este módulo. Fíjese en el esquema general de la página anterior y sustituya el valor XXXX por los dígitos binarios correctos.

A continuación se muestra parte del código VHDL que deberá escribir. Complete la descripción arquitectural de interconexión de la forma adecuada.

ARCHIVO: visualizacion.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity visualizacion is
    Port ( ENABLE : in STD_LOGIC;                          -- Entrada de ENABLE
           DECENAS : in  STD_LOGIC_VECTOR (3 downto 0);    -- Entrada decenas
           UNIDADES : in  STD_LOGIC_VECTOR (3 downto 0);   -- Entrada unidades
           DECIMAS : in  STD_LOGIC_VECTOR (3 downto 0);    -- Entrada décimas
           CLK     : in  STD_LOGIC;                       -- Entrada de reloj
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);         -- Salida para los displays
           DP   : out STD_LOGIC;                          -- Salida punto decimal
           AN   : out  STD_LOGIC_VECTOR (3 downto 0)); -- Activación individual
end visualizacion;

architecture a_visualizacion of visualizacion is

-- COMPONENTES

component decodBCDa7s is
    Port ( BCD : in  STD_LOGIC_VECTOR (7 downto 0);   -- Entrada del valor BCD
           SEGMENTOS : out  STD_LOGIC_VECTOR (0 to 6));   -- Salidas al display
end component;

            . . . DEFINIR OTROS COMPONENTES NECESARIOS

-- SEÑALES

            . . . POSIBLES SEÑALES NECESARIAS
begin

   U1 : decodBCDa7s port map ( . . . COMPLETE LA DESCRIPCION);

   . . . INTERCONEXIÓN ENTRE COMPONENTES

end a_visualizacion;
```

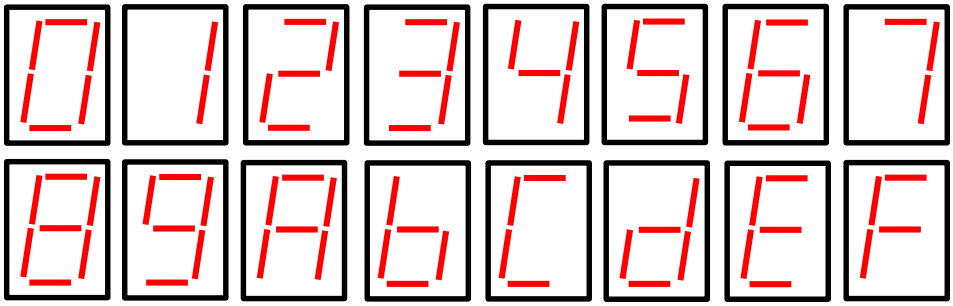
Pruebas del módulo de visualización:

1. Para probar el módulo de visualización descargue los archivos: **prueba_visualizacion.vhd y asoc_prueba_vis.ucf**. Estos archivos, compilados junto con los creados por los alumnos, permiten comprobar el funcionamiento del módulo.

2. Compile su código junto con los archivos **prueba_visualizacion.vhd** y **asoc_prueba_vis.ucf** y genere el fichero de bitstream. Cargue dicho fichero sobre la FPGA.

Asegúrese que se cumplen todas las condiciones siguientes:

- Deben aparecer secuencialmente todos los valores “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “A”, “B”, “C”, “D”, “E”, “F” en los displays 1 a 3.
- El display de la derecha debe presentar permanentemente una “C”.
- El segundo display por la izquierda deberá tener su punto decimal encendido.



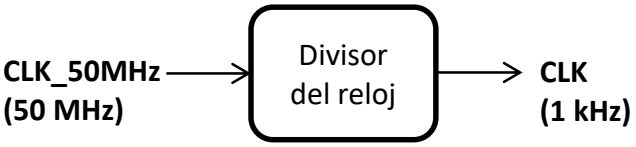
DATOS PARA LA MEMORIA (Emplee la plantilla de memoria publicada)

1. Explique razonadamente la elección de los valores para los bits “XXXX” en la entrada E0 del registro.

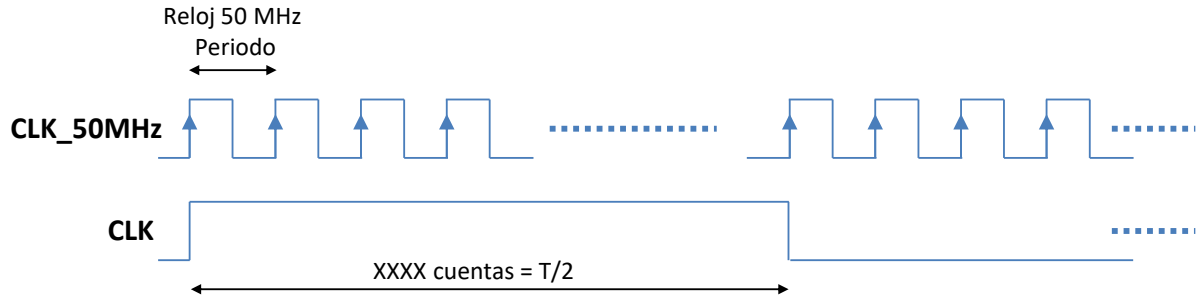
2. Detalle el código VHDL de los cuatro módulos (registro.vhd, MUX4x4.vhd, refresco.vhd, visualizacion.vhd) debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

MODULO 6: Divisor del reloj

El divisor del reloj tiene como entrada el reloj de la FPGA (CLK_50MHz), de 50 MHz de frecuencia, y como salida debe generar una señal cuadrada (**CLK**) de 1 kHz. Esto se lleva a cabo mediante división de frecuencia con un contador y una señal (**frec_div**) que cambia de valor lógico cuando la cuenta alcanza un valor determinado.



A continuación se muestra parte del código VHDL necesario. Repase el ejemplo 2 del "Manual de referencia sobre la tarjeta BASYS2" para entender su funcionamiento.



ARCHIVO: div_reloj.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity div_reloj is
    Port ( CLK_50MHz  : in  STD_LOGIC;      -- Entrada reloj de la FPGA 50 MHz
           CLK        : out  STD_LOGIC);    -- Salida reloj a 1 kHz
end div_reloj;

architecture a_div_reloj of div_reloj is

signal contador : STD_LOGIC_VECTOR  (15 downto 0);  -- contador
signal frec_div : STD_LOGIC;                -- señal de frecuencia dividida

begin

process(CLK_50MHz)
  begin
    if (CLK_50MHz'event and CLK_50MHz='1') then  -- En cada flanco de subida
      contador<=contador+1;                 -- Incrementar el contador
      if (contador>=XXXX) then              -- Cuando alcanza el valor máximo
        contador<=(others=>'0');            -- Poner el contador a 0
        frec_div<=not frec_div;             -- e intercambiar el valor de frec_div
      end if;
    end if;
  end process;

CLK<=frec_div;                              -- La salida es la señal frec_div

end a_div_reloj;
```

Pruebas del divisor de reloj:

1. Deberá calcular el valor de la constante XXXX para poder dividir la frecuencia del reloj de entrada al valor especificado (vea el ejemplo 2 del "Manual del referencia de la tarjeta BASYS2").

2. Para comprobar que este módulo funciona correctamente asocie la salida CLK a un terminal de salida de la FPGA y mida con el osciloscopio la frecuencia obtenida. Por ejemplo puede utilizar las siguientes asociaciones:

ARCHIVO: asoc_reloj.ucf (Puede descargar esta plantilla desde MOODLE)

```
# Reloj principal del sistema

NET "CLK_50MHz" LOC = "M6"; # Señal de reloj del sistema

# Salida externa donde se visualiza la señal de reloj dividida

NET "CLK" LOC = "A9"; # Salida terminal 14
```

3. Conecte el osciloscopio entre las patillas 14 (salida) y 25 (masa). Debería observar una señal cuadrada entre 0 y 5 V y 1 kHz de frecuencia.

DATOS PARA LA MEMORIA (Emplee la plantilla de memoria publicada)

1. Detalle el cálculo de la constante XXXX

2. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

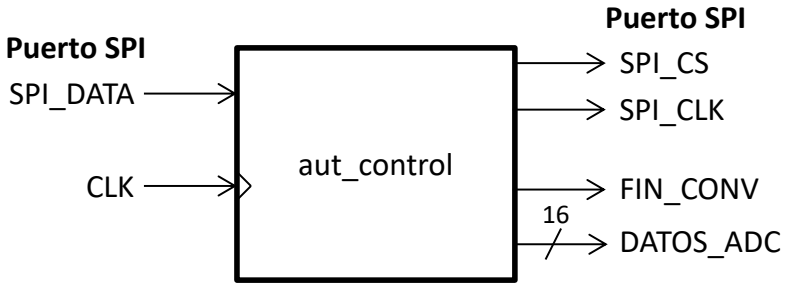
3. Incluya la captura de pantalla del osciloscopio donde pueda observarse la señal de reloj de 1 kHz con niveles de 0 y 5V.

Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

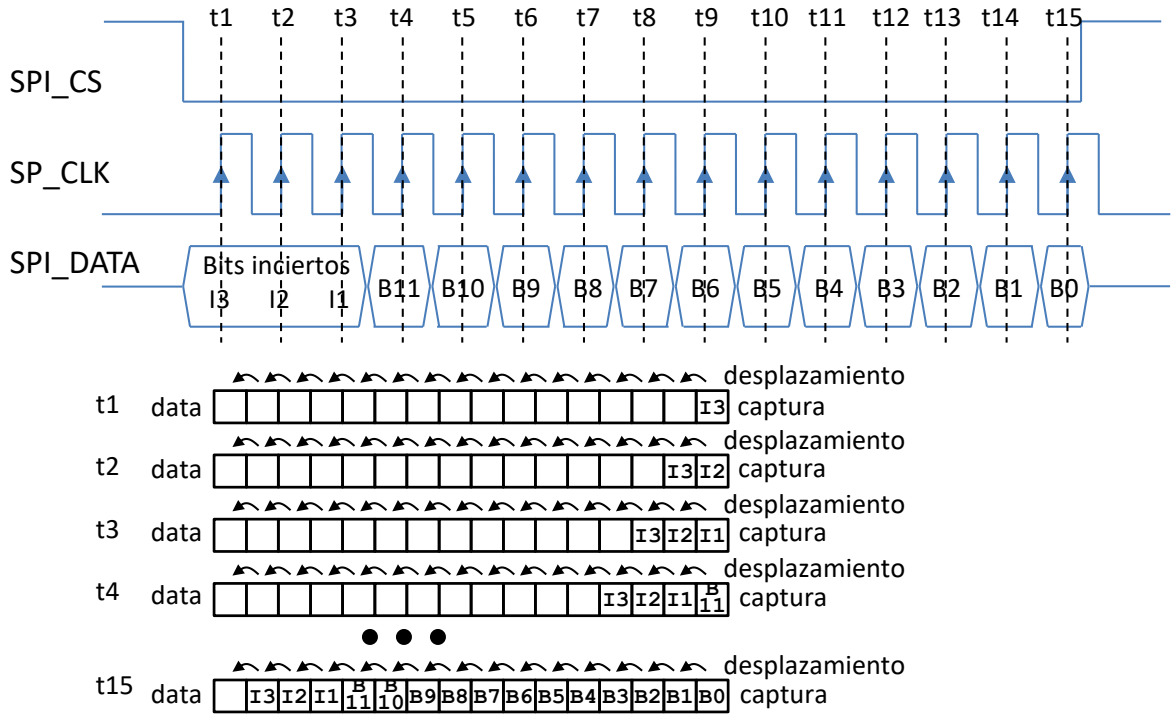
MÓDULO 7: Autómata de control

El autómata de control genera la secuencia necesaria para obtener los datos del puerto **SPI** (vea la figura de la página 8). También captura los bits correspondientes al código binario de 12 bits y lo almacena en un registro.

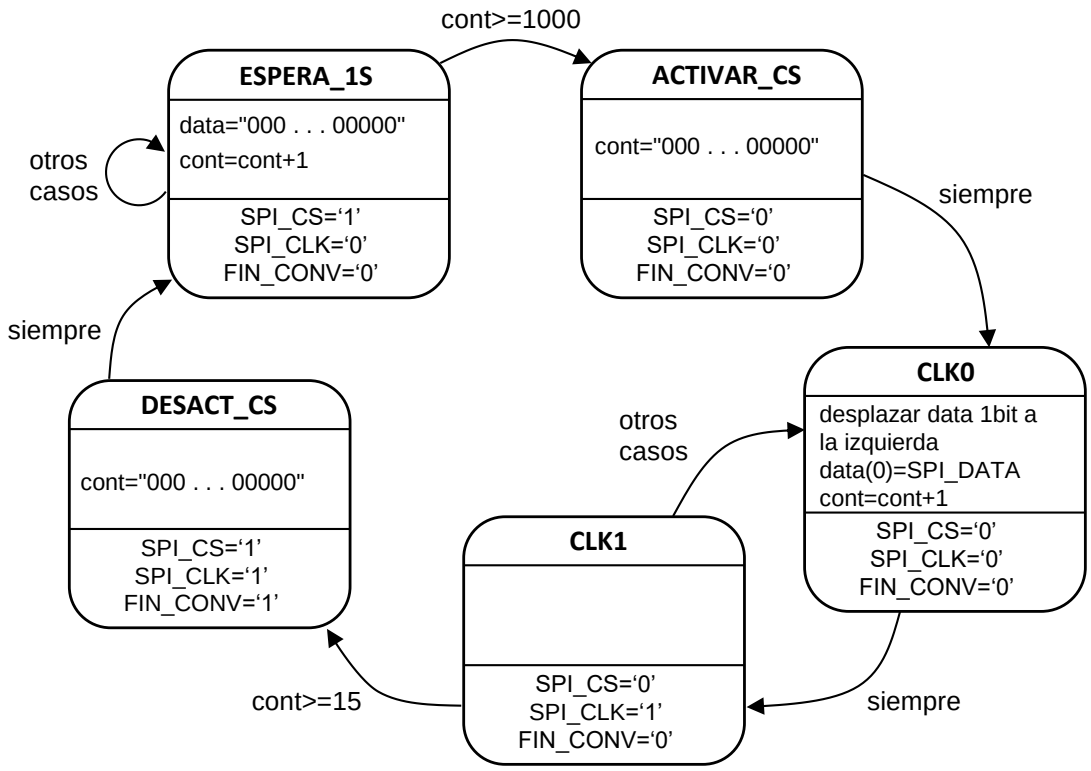
El autómata toma como entradas la señal de reloj de 1ms de periodo (**CLK**) y la señal **SPI_DATA** del puerto **SPI**. Sus salidas son: **SPI_CS** y **SPI_CLK** del puerto **SPI**, **FIN_CONV** (que activa a nivel alto cuándo está lista una nueva muestra) y **DATOS_ADC** (vector de 16 bits que contiene la muestra leída en los 12 bits menos significativos). El esquema de este módulo se muestra a continuación:



Este autómata comienza esperando un tiempo de 1s (tiempo entre medidas de temperatura). A continuación activa (a nivel bajo) la salida **SPI_CS** y genera 15 pulsos en la salida **SPI_CLK**. En cada flanco de subida de la señal **SPI_CLK** captura el dato que entra por **SPI_DATA** en el bit 0 de un registro de desplazamiento (**data**) que se va desplazando hacia la izquierda. De este modo, al final del proceso tendremos los 12 bits correspondientes al valor leído en los bits menos significativos del registro de desplazamiento. La figura siguiente resume este procedimiento:



El diagrama de estados de este autómata es el siguiente:



Descripción de los estados:

- **ESPERA_1S**: Se desactivan todas las salidas (**SPI_CLK, SPI_CS y FIN_CONV**) Teniendo en cuenta que cada estado dura 1 ciclo de reloj (1 ms), se incrementa el contador (cont) hasta que alcanza el valor 1000 (1 segundo). Cuando alcanza ese valor se pasa a **ACTIVAR_CS**. En otro caso se permanece en **ESPERA_1S**.
- **ACTIVAR_CS**: Pone el contador (cont) a 0 y activa la salida **SPI_CS** (a nivel bajo) para iniciar una nueva conversión de datos. Al ciclo siguiente pasa a **CLK0.**
- **CLK0**: Desplaza el registro de desplazamiento (**data**) 1 bit a la izquierda y captura el bit presente en **SPI_DATA** en la posición data(0). Pone la salida **SPI_CLK** a 0 y se incrementa el contador (**cont**) para contar el número de pulsos que se generan. Se pasa a **CLK1.**
- **CLK1**: Pone la salida **SPI_CLK** a 1. Si el contador ha llegado a 15 (se han generado 15 pulsos de reloj), pasa a **DESACT_CS**. En otro caso vuelve a **CLK0.**
- **DESACT_CS**: Desactiva la señal SPI_CS (a nivel alto) y pone el contador (**cont**) a 0. Se activa la señal de fin de conversión (**FIN_CONV**) indicando que hay una muestra lista para ser leída. Se pasa a **ESPERA_1S**.

Tenga en cuenta que cada estado dura 1 ciclo de reloj (1 ms), por lo tanto, el tiempo entre dos flancos de subida consecutivos de la señal **SPI_CLK** es de 2 ms. El contador **cont** se emplea tanto para medir el tiempo de espera de 1s como para contar el número de ciclos de reloj **SPI_CLK** que se generan.

El siguiente código muestra parte de su descripción en VHDL.

ARCHIVO: aut_control.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity aut_control is
    Port ( CLK        : in  STD_LOGIC;                    -- reloj del sistema
           SPI_DATA   : in  STD_LOGIC;      -- entrada de datos del puerto SPI
           SPI_CLK    : out  STD_LOGIC;     -- Salida de reloj del puerto SPI
           SPI_CS     : out  STD_LOGIC;     -- Chip Select del puerto SPI
           FIN_CONV   : out  STD_LOGIC;     -- Fin de conversión A/D
           DATOS_ADC  : out  STD_LOGIC_VECTOR (15 downto 0));  -- Dato leído del ADC
end aut_control;

architecture a_aut_control  of aut_control is

type STATE_TYPE is (ESPERA_1S,ACTIVAR_CS,CLK1,CLK0,DESACT_CS);

signal ST : STATE_TYPE := ESPERA_1S;
signal cont : unsigned (15 downto 0):=(others=>'0');
signal data : unsigned (15 downto 0):=(others=>'0');

begin

process (CLK)
  begin
  if (CLK'event and CLK='1') then   -- En cada flanco de subida del reloj
    case ST is
      when ACTIVAR_CS =>
            cont<=(others=>'0');
            ST<=CLK0;

      -- OTROS CASOS

    end case;
  end if;
  end process;

-- PARTE COMBINACIONAL (CABLEADO DE LAS SALIDAS)

SPI_CS<='0' when ... COMPLETAR;
DATOS_ADC<= ... COMPLETAR;
FIN_CONV<= ... COMPLETAR;
SPI_CLK<= ... COMPLETAR;

end a_aut_control;
```

Diseño:

1. Complete la descripción funcional del autómata según se describe en su diagrama de estados. **Realice el autómata con un solo proceso**. Asigne el valor de los registros **cont** y **data** dentro de los estados. **Los valores de las salidas deberán cablearse fuera del proceso**. Como puede ver, parte del código se muestra como ejemplo.

Pruebas del autómata de control: simulación:

1. Descargue el archivo **tb_aut_control.vhd** y realice la simulación.
2. En la consola del ISIM escriba **run 1.5 s** para ampliar el tiempo de simulación.

3. Las señales de salida: **SPI_CS, SPI_CLK, FIN_CONV y DATOS_ADC** deben cumplir los tiempos y valores que se resumen en la siguiente tabla:

Tiempo	SPI_CS	SPI_CLK	FIN_CONV	DATOS_ADC
Anterior	1	0	0	0000000000000000
1.000,5 ms	Flanco de bajada	0	0	0000000000000000
1.002,5 ms	0	Flanco de subida	0	0000000000000000
1.003,5 ms	0	Flanco de bajada	0	0000000000000000
1.004,5 ms	0	Flanco de subida	0	0000000000000000
1.005,5 ms	0	Flanco de bajada	0	0000000000000000
1.006,5 ms	0	Flanco de subida	0	0000000000000000
1.007,5 ms	0	Flanco de bajada	0	0000000000000000
1.008,5 ms	0	Flanco de subida	0	Cambio de valor
1.009,5 ms	0	Flanco de bajada	0	0000000000000001
1.010,5 ms	0	Flanco de subida	0	Cambio de valor
1.011,5 ms	0	Flanco de bajada	0	0000000000000010
1.012,5 ms	0	Flanco de subida	0	Cambio de valor
1.013,5 ms	0	Flanco de bajada	0	0000000000000101
1.014,5 ms	0	Flanco de subida	0	Cambio de valor
1.015,5 ms	0	Flanco de bajada	0	0000000000001010
1.016,5 ms	0	Flanco de subida	0	Cambio de valor
1.017,5 ms	0	Flanco de bajada	0	0000000000010101
1.018,5 ms	0	Flanco de subida	0	Cambio de valor
1.019,5 ms	0	Flanco de bajada	0	0000000000101010
1.020,5 ms	0	Flanco de subida	0	Cambio de valor
1.021,5 ms	0	Flanco de bajada	0	0000000001010101
1.022,5 ms	0	Flanco de subida	0	Cambio de valor
1.023,5 ms	0	Flanco de bajada	0	0000000010101010
1.024,5 ms	0	Flanco de subida	0	Cambio de valor
1.025,5 ms	0	Flanco de bajada	0	0000000101010101
1.026,5 ms	0	Flanco de subida	0	Cambio de valor
1.027,5 ms	0	Flanco de bajada	0	0000001010101010
1.028,5 ms	0	Flanco de subida	0	Cambio de valor
1.029,5 ms	0	Flanco de bajada	0	0000010101010101
1.030,5 ms	0	Flanco de subida	0	Cambio de valor
1.031,5 ms	Flanco de subida	1	Flanco de subida	0000101010101010
1.032,5 ms	1	Flanco de bajada	Flanco de bajada	0000101010101010
1.033,5 ms	1	0	0	Cambio de valor
Posterior	1	0	0	0000000000000000

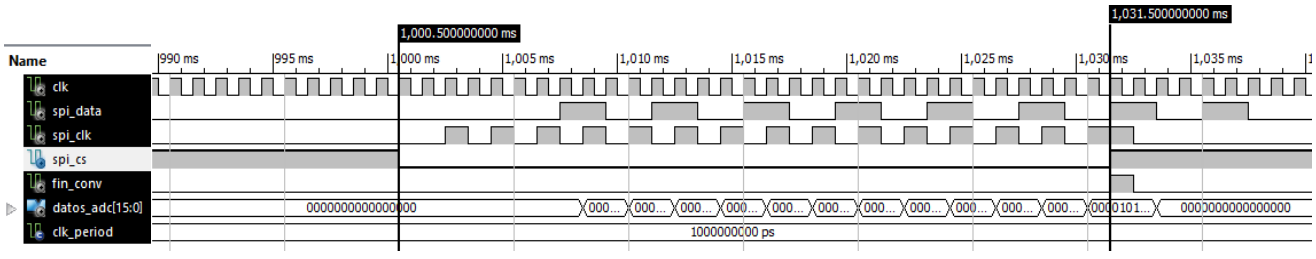


Tabla y fragmento del gráfico de simulación del autómata de control

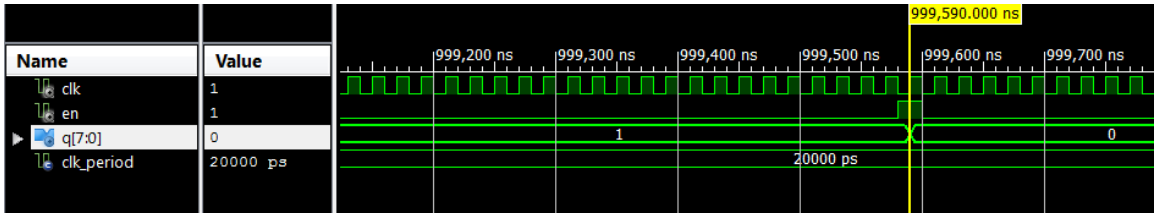
DATOS PARA LA MEMORIA (Emplee la plantilla de memoria publicada)

1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

2. Incluya una captura de pantalla de la simulación donde pueda verse con precisión el tiempo entre 995 y 1035 ms y donde puedan apreciarse las transiciones de las señales de salida. Coloque dos cursores de tiempo: uno en el flanco de bajada de la salida SPI_CS y otro en el flanco de subida de la salida SPI_CS.

Tenga en cuenta que debido al fondo negro de la pantalla del simulador es difícil que se vean las señales adecuadamente. Se recomienda utilizar la herramienta de recorte de Word y ampliar convenientemente la captura de pantalla para resaltar las regiones importantes.

La siguiente figura es un ejemplo de cómo se deben entregar las gráficas de simulación.



Ejemplo de gráfica de simulación con las lecturas visibles

Intente realizar la memoria de forma creativa, puede emplear diferentes colores para las diferentes señales y varios indicadores de tiempo. El objetivo es conseguir gráficas donde se aprecien con detalle los instantes temporales que se indican en cada señal.

MÓDULO 8: ADC a TEMP

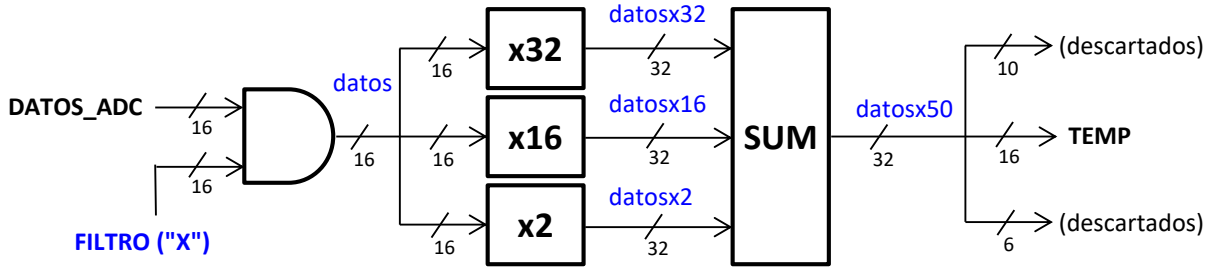
Este módulo convierte la tensión leída en el convertidor analógico/digital en el valor de temperatura correspondiente. Sabiendo que la tensión de salida del circuito analógico (V_a) tiene una precisión de 100 mV/°C y que la tensión de referencia (V_{REF}) del convertidor A/D es de 5V, este módulo combinacional debe realizar la siguiente operación en binario:

$$TEMP = V_a \cdot \frac{1}{100mV/°C} \ [°C]$$

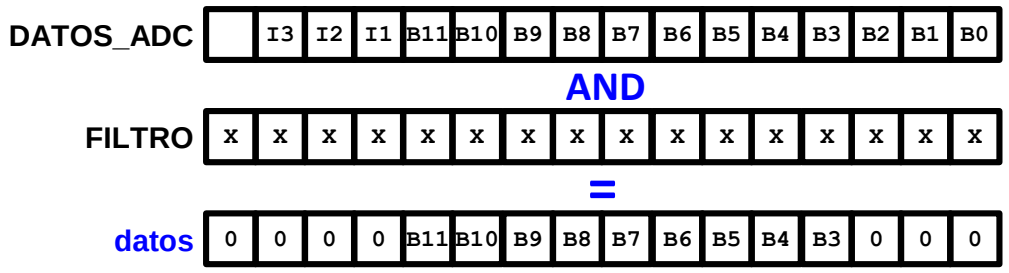
$$V_a = \frac{DATOS\_ADC_{(10)}}{2^{12}} \cdot V_{REF} \ \ [V]$$

$$TEMP = \frac{DATOS\_ADC_{(10)}}{2^{12}} \cdot 5V \cdot \frac{1}{100mV/°C} = \frac{DATOS\_ADC_{(10)} \cdot 50}{4096} \ \ [°C]$$

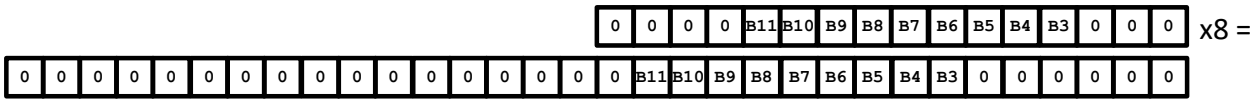
Resumiendo, este módulo debe tomar como entrada el valor **DATOS_ADC** leído del convertidor A/D (ya en binario), multiplicarlo por 50 (en binario) y dividirlo por 4096 (en binario), conservando 10 bits (parte entera) y 6 bits (parte decimal) tal como se explica en las páginas 9 y 10. El esquema general del circuito se muestra a continuación:



1. Inicialmente tomamos los datos de entrada (**DATOS_ADC**) y los filtramos poniendo a 0 los bits que no nos interesan (puerta AND). Recuerde que los 4 bits más significativos son inciertos. Además, para apreciar la décima de grado Celsius necesitamos apreciar 10 mV. Si cada escalón de tensión es de $5/2^{12}$=1,22 mV, los 3 bits menos significativos nos darían variaciones de tensión entre 0V (para 000) y 8,54 mV (para 111). Por lo tanto tampoco nos aportan información significativa. Debemos por tanto comenzar poniendo a 0 todos estos bits quedándonos únicamente con los necesarios. Esto lo haremos mediante una operación AND con un valor que deberá obtener:

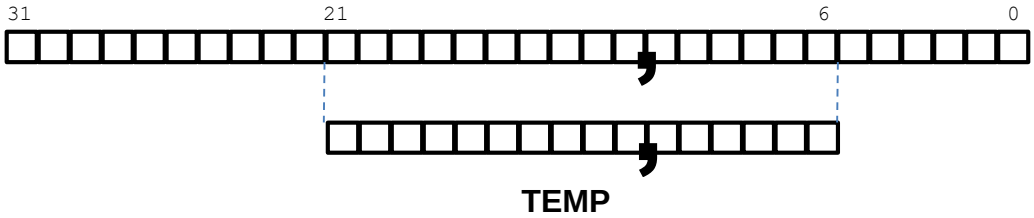


2. Para realizar la operación DATOS_ADC·50 (en binario) vamos a multiplicar por potencias de 2 que resulta más sencillo dado que, en binario, esto solamente implica desplazar los bits hacia la izquierda añadiendo '0' por la derecha. Concretamente: DATOS_ADC·50 = DATOS_ADC·(32 + 16 + 2). Para realizar estas 3 multiplicaciones (x32, x16 y x2) [en binario serían (x100000, x10000 y x10)] emplearemos vectores de 32 bits de modo que podamos desplazar los bits hacia la izquierda añadiendo '0' por la derecha (señales datosx32, datosx16 y datosx2). Por ejemplo:

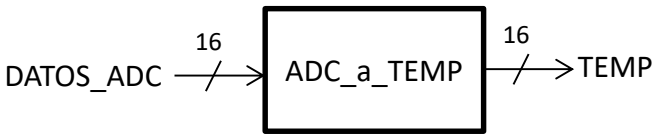


3. A continuación sumamos los tres registros donde hemos multiplicado por 2, 16 y 32 obteniendo de este modo la multiplicación por 50. El resultado también lo obtenemos en un registro de 32 bits (señal datosx50).

4. Por último debemos dividir por 4096 (1000000000000 en binario), lo que implica mover la coma 12 posiciones a la izquierda. Dado que tomaremos únicamente 10 bits para la parte entera y 6 para la parte decimal, descartaremos los bits no deseados, obteniendo de nuevo un vector de 16 bits. Tomaremos por tanto la parte del vector que nos interesa.



El esquema de este módulo combinacional y parte de su descripción en VHDL se muestran a continuación. Complete su descripción funcional.



ARCHIVO: ADC_a_TEMP.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity ADC_a_TEMP is
    Port ( DATOS_ADC : in  STD_LOGIC_VECTOR (15 downto 0);     -- Datos del convertidor A/D
           TEMP      : out  STD_LOGIC_VECTOR (15 downto 0));   -- Salida temperatura en
end ADC_a_TEMP;                                                -- punto fijo con 6 bits decimales

architecture a_ADC_a_TEMP of ADC_a_TEMP is

constant FILTRO  : unsigned (15 downto 0):= "XXXXXXXXXXXXXXXX";
signal datos     : unsigned (15 downto 0):=(others=>'0'); -- Datos leídos del convertidor y filtrados
signal datosx32  : unsigned (31 downto 0):=(others=>'0'); -- Datos multiplicados por 32
signal datosx16  : unsigned (31 downto 0):=(others=>'0'); -- Datos multiplicados por 16
signal datosx2   : unsigned (31 downto 0):=(others=>'0'); -- Datos multiplicados por 2
signal datosx50  : unsigned (31 downto 0):=(others=>'0'); -- Datos multiplicados por 50
```

```
begin

datos<= unsigned(DATOS_ADC) and FILTRO;  -- datos de entrada filtrados

-- Para pasar a ºC se trata de multiplicar por 50 y dividir entre 4096
-- Para multiplicar datos*50, hacemos: datos*32+datos*16+datos*2

datosx32 <= ... COMPLETE ESTA LÍNEA ;
datosx16 <= ... COMPLETE ESTA LÍNEA ;
datosx2  <= ... COMPLETE ESTA LÍNEA ;

datosx50<= ... SUMAMOS LOS TRES VECTORES ANTERIORES PARA OBTENER datos x 50;

-- Ahora tomamos los bits correspondientes a 10 enteros y 6 decimales

TEMP<=STD_LOGIC_VECTOR(--- TOME LOS BITS NECESARIOS DEL VECTOR );

end a_ADC_a_TEMP;
```

Diseño:

1. Obtenga el valor correcto de la constante FILTRO para filtrar los datos.
2. Complete las líneas de código VHDL indicadas teniendo en cuenta el funcionamiento del módulo descrito en la introducción.

Pruebas del módulo:

1. Descargue el archivo **tb_ADC_a_TEMP.vhd** y realice la simulación.
2. En la consola del ISIM escriba **run 45 us** para ampliar el tiempo de simulación. Coloque el cursor sobre la señal “datos_adc”, haga click en el botón derecho y seleccione la opción Radix->Unsigned Decimal.
3. El archivo de simulación recorre todos los valores binarios posibles en el vector de entrada (DATOS_ADC), desde “0000000000000000” hasta “1111111111111111”. La salida (TEMP) va cambiando según el valor de la entrada.
4. A continuación mostramos algunos de los valores en las transiciones de la salida para valores concretos de tiempo. Asegúrese de que se cumplen todos.

Tiempo transición	**Entrada (DATOS_ADC) Transición a:**	**Salida (TEMP) Transición a:**
…	…	…
20,080 µs	2008 (base 10)	0000011000100000 (24,5 en base 10)
21,280 µs	2128 (base 10)	0000011001111110 (25,96875 en base 10)
24,640 µs	2464 (base 10)	0000011110000101 (30,078125 en base 10)
28,240 µs	2824 (base 10)	0000100010011110 (34,46875 en base 10)
…	…	…

DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

1. Razone el valor elegido para la constante FILTRO.
2. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**
3. Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.

MODULO 9: TEMP a BCD

La representación de la temperatura en binario con 16 bits donde 10 forman la parte entera y 6 forman la parte decimal no es apropiada para su representación en un display. Para eso, tenemos que obtener la expresión en BCD con tres cifras: DECENAS, UNIDADES y DECIMAS.

Para realizar esta operación, inicialmente separamos la parte entera (ENT) de la parte decimal (DEC) empleando dos señales. Después, en función de estos valores realizamos la siguiente operación:

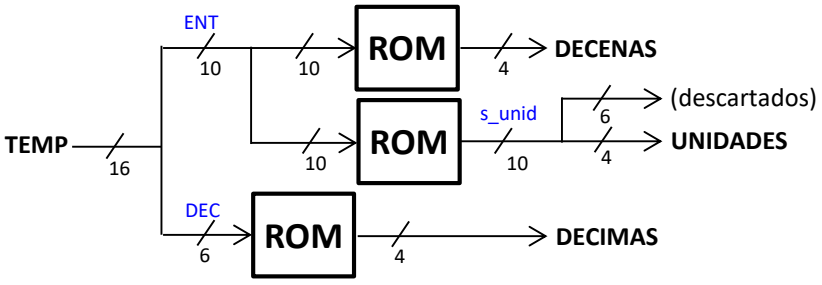
DECENAS	UNIDADES	DECIMAS
Si ENT>=90 DECENAS=9 en otro caso	Si ENT>=90 UNIDADES=ENT-90 en otro caso	Si DEC<6 DECIMAS=0 en otro caso
Si ENT>=80 DECENAS=8 en otro caso	Si ENT>=80 UNIDADES=ENT-80 en otro caso	Si DEC<12 DECIMAS=1 en otro caso
Si ENT>=70 DECENAS=7 en otro caso	Si ENT>=70 UNIDADES=ENT-70 en otro caso	Si DEC<18 DECIMAS=2 en otro caso
Si ENT>=60 DECENAS=6 en otro caso	Si ENT>=60 UNIDADES=ENT-60 en otro caso	Si DEC<24 DECIMAS=3 en otro caso
Si ENT>=50 DECENAS=5 en otro caso	Si ENT>=50 UNIDADES=ENT-50 en otro caso	Si DEC<30 DECIMAS=4 en otro caso
Si ENT>=40 DECENAS=4 en otro caso	Si ENT>=40 UNIDADES=ENT-40 en otro caso	Si DEC<36 DECIMAS=5 en otro caso
Si ENT>=30 DECENAS=3 en otro caso	Si ENT>=30 UNIDADES=ENT-30 en otro caso	Si DEC<42 DECIMAS=6 en otro caso
Si ENT>=20 DECENAS=2 en otro caso	Si ENT>=20 UNIDADES=ENT-20 en otro caso	Si DEC<48 DECIMAS=7 en otro caso
Si ENT>=10 DECENAS=1 en otro caso	Si ENT>=10 UNIDADES=ENT-10 en otro caso	Si DEC<54 DECIMAS=8 en otro caso
DECENAS=0	UNIDADES=ENT	DECIMAS=9

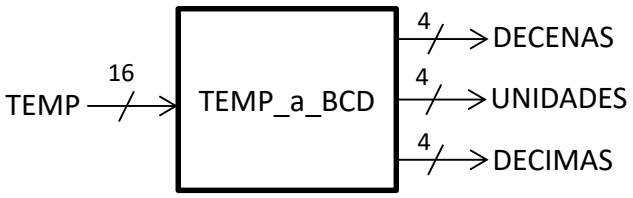
En el caso de la parte entera básicamente se comprueba la decena en la que se encuentra el valor binario y se deciden la DECENA y la UNIDAD.

En el caso de la parte decimal tenemos 6 bits, correspondientes a 64 valores. Dividiendo 64 entre 10 (que son las cifras decimales en BCD) tenemos un valor cuya parte entera es 6. Por tanto, en función del valor DEC decidimos la cifra BCD (0 entre 0 y 5, 1 entre 6 y 11, etc…). Hay un pequeño desajuste en el caso de las DECIMAS=9, donde el intervalo de valores es de 10 en lugar de 6. Esto no es demasiado relevante para el resultado.

Para esto vamos a emplear sentencias ***when*** … ***else*** que implementan físicamente una **ROM** que lleva a cabo las operaciones anteriores. Sus entradas serán las señales ENT y DEC y sus salidas serán las propias salidas del módulo: DECENAS, UNIDADES y DECIMAS. En el caso de las UNIDADES será necesario emplear una señal adicional (s_unid) para poder realizar la resta con 10 bits, descartando luego los más significativos.

El esquema de este módulo combinacional y parte de su descripción en VHDL se muestran a continuación. Complete su descripción funcional.





ARCHIVO: TEMP_a_BCD.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity TEMP_a_BCD is
    Port ( TEMP     : in  STD_LOGIC_VECTOR (15 downto 0);  -- Temperatura con 6 bits decimales
           DECENAS  : out  STD_LOGIC_VECTOR (3 downto 0);  -- Decenas en BCD
           UNIDADES : out  STD_LOGIC_VECTOR (3 downto 0);  -- Unidades en BCD
           DECIMAS  : out  STD_LOGIC_VECTOR (3 downto 0)); -- Décimas en BCD
end TEMP_a_BCD;

architecture a_TEMP_a_BCD of TEMP_a_BCD is

signal ENT : unsigned (9 downto 0):="0000000000";  -- Parte entera de la temperatura
signal DEC : unsigned (5 downto 0):="000000";      -- Parte decimal de la temperatura
signal s_unid : unsigned (9 downto 0):="0000000000";  -- unidades de la temperatura
begin

-- Separamos parte entera y parte decimal

ENT<= ... COMPLETE LA LÍNEA;
DEC<= ... COMPLETE LA LÍNEA;

-- Las decenas se determinan en función del intervalo de valores donde se
-- encuentre la temperatura

DECENAS<=  ... Complete este código asignando el valor adecuado según
               la década a la que pertenece la parte entera (ENT).
               Emplee sentencias “when else” ;

-- Las unidades se determinan restando las decenas al valor de la temperatura.

s_unid<= ... Complete este código seleccionando en s_unid el valor de las unidades
             Para ello reste el valor de la parte entera menos la década a la que pertenece.
             Emplee sentencias “when else”  ;

-- Para la parte decimal vemos cuál es la décima (entre 10 valores de 0 a 9)
-- según el valor binario de 6 bits entre 0 y 64

DECIMAS<= ... Complete este código seleccionando el valor de la décima en función de los
              Intervalos de la parte decimal (DEC) 0-6, 7-12, 13-18, etc...
              Emplee sentencias “when else”  ;

UNIDADES<= ... Las unidades son los bits menos significativos de s_unid ;

end a_TEMP_a_BCD;
```

Diseño:

1. Complete las líneas de código VHDL indicadas teniendo en cuenta el funcionamiento del módulo descrito en la introducción y siguiendo las instrucciones que se indican en los comentarios.

Pruebas del módulo:

1. Descargue el archivo **tb_TEMP_a_BCD.vhd** y realice la simulación.
2. En la consola del ISIM escriba **run 65 us** para ampliar el tiempo de simulación.
3. Coloque el cursor sobre la señal “decenas”, haga click en el botón derecho y seleccione la opción Radix->Unsigned Decimal.
4. Coloque el cursor sobre la señal “unidades”, haga click en el botón derecho y seleccione la opción Radix->Unsigned Decimal.
5. Coloque el cursor sobre la señal “decimas”, haga click en el botón derecho y seleccione la opción Radix->Unsigned Decimal.
6. El archivo de simulación recorre todos los valores binarios posibles en el vector de entrada (TEMP), desde “0000000000000000” hasta “1111111111111111”. Las salidas (DECENAS, UNIDADES y DECIMAS) van cambiando según el valor de la entrada.
7. En la siguiente tabla se muestran algunos de los valores que deben tomar las salidas para algunos de los valores de la entradaen determinados instantes de tiempo. Asegúrese de que se cumplen todos.

tiempo	**TEMP (transición a)**	**DECENAS**	**UNIDADES**	**DECIMAS**
…	…	…	…	…
16.480 ns	0000011001110000	2	5	Transición a 8
…	…	…	…	…
19.500 ns	0000011110011110	3	0	Transición a 5
…	…	…	…	…
22.460 ns	0000100011000110	3	5	Transición a 1
…	…	…	…	…
36.840 ns	0000111001100100	5	7	Transición a 6
…	…	…	…	…

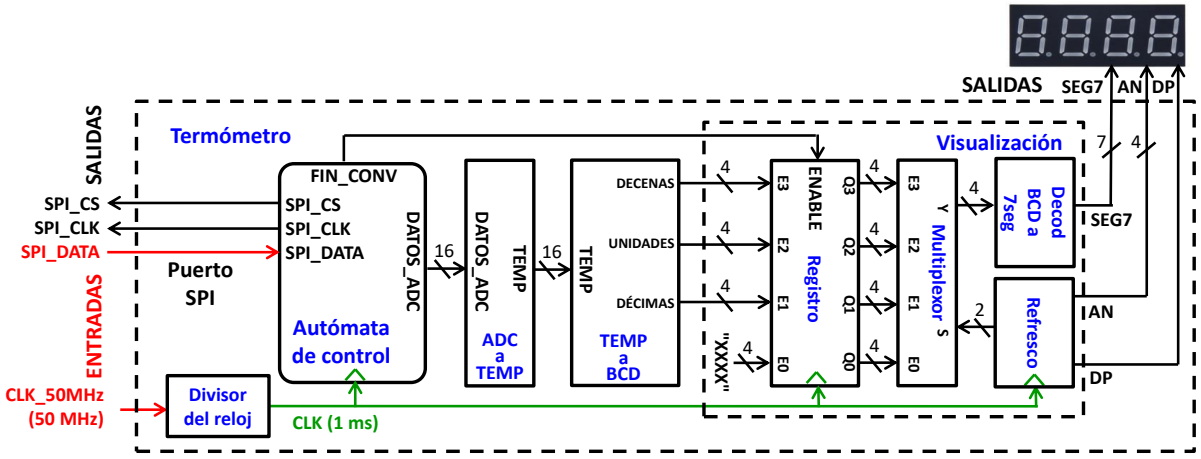
DATOS PARA LA MEMORIA **(Emplee la plantilla de memoria publicada)**

1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**
2. Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.

MÓDULO 10: Circuito digital completo

En este módulo deberá interconectar todos los elementos que componen el hardware digital. Para ello tiene que crear un nuevo archivo que describa de forma arquitectural (estructural) las conexiones entre los diferentes módulos que serán importados como componentes.

El esquema general del circuito digital es el que sigue:



Para realizar la interconexión entre los módulos debe declarar éstos como **component**, debe declarar también algunas señales para realizar los cableados internos y hacer las asignaciones adecuadas. Parte de la descripción en VHDL se muestra a continuación:

ARCHIVO: termometro.vhd (Puede descargar esta plantilla desde MOODLE)

```
entity termometro is
  Port ( CLK_50MHz : in  STD_LOGIC;          -- Reloj del sistema
         SPI_DATA : in  STD_LOGIC;           -- Entrada de datos del puerto SPI
         SPI_CLK : out  STD_LOGIC;           -- Salida de reloj del puerto SPI
         SPI_CS : out  STD_LOGIC;            -- Salida chip select del puerto SPI
         AN : out  STD_LOGIC_VECTOR (3 downto 0); -- Salida de selección de los displays
         SEG7 : out  STD_LOGIC_VECTOR (0 to 6); -- Salida para los segmentos de los displays
         DP : out  STD_LOGIC);              -- Salida para el punto decimal de los displays
end termometro;

architecture a_termometro of termometro is

component div_reloj is
    Port ( CLK_50MHz : in  STD_LOGIC;       -- Entrada reloj de la FPGA 50 MHz
           CLK       : out  STD_LOGIC);     -- Salida reloj a 1 KHz
end component;

     . . . OTROS COMPONENTES

     . . . SEÑALES NECESARIAS PARA LAS INTERCONEXIONES

begin

  U1 : div_reloj port map (. . . COMPLETE LA DESCRIPCIÓN);

     . . . OTRAS CONEXIONES

end a_termometro;
```

Pruebas:

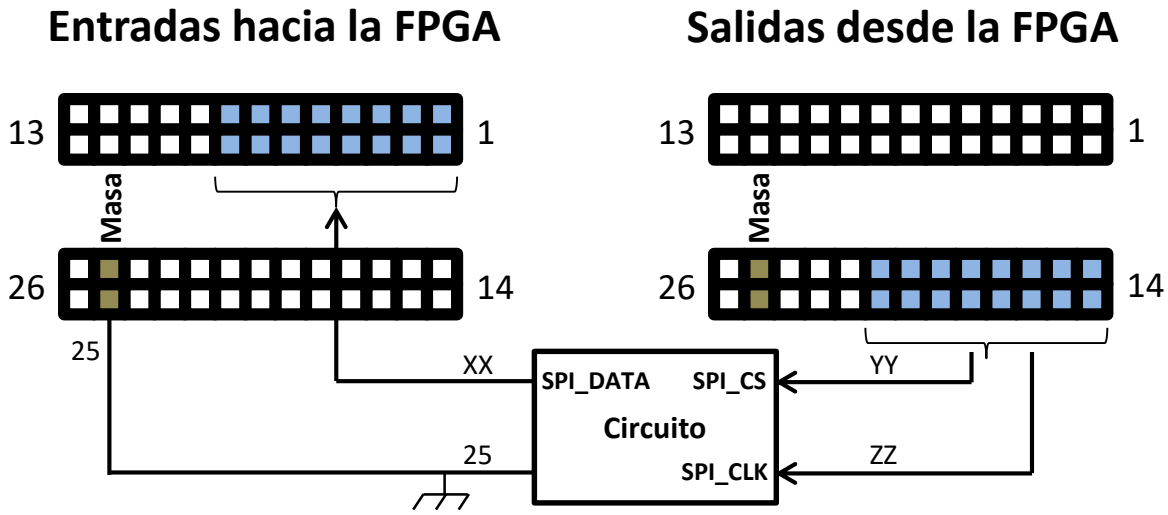
1. Complete su descripción estructural según se describe en los comentarios incluidos. Declare las señales que necesite para las conexiones internas y realice las conexiones entre los diferentes bloques y las entradas y salidas del sistema.

2. Escriba el archivo de asociaciones que permita conectar el hardware digital con los elementos de la FPGA. Se muestra un posible ejemplo a continuación donde XX, YY y ZZ son los pines de entrada y salida de la FPGA para el puerto SPI que puede elegir libremente:

ARCHIVO: asociaciones.ucf (Puede descargar esta plantilla desde MOODLE)

```
# Reloj principal del sistema
NET "CLK_50MHz" LOC = "M6"; # Entrada del reloj del sistema

# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "--";  # señal = CA
NET "SEG7<1>" LOC = "--";  # Señal = CB
NET "SEG7<2>" LOC = "--";  # Señal = CC
NET "SEG7<3>" LOC = "--";  # Señal = CD
NET "SEG7<4>" LOC = "--";  # Señal = CE
NET "SEG7<5>" LOC = "--";  # Señal = CF
NET "SEG7<6>" LOC = "--";  # Señal = CG
NET "DP"      LOC = "--";  # Punto decimal

# Señales de activación de los displays
NET "AN<0>" LOC = "--";  # Activación del display 0 = AN0
NET "AN<1>" LOC = "--";  # Activación del display 1 = AN1
NET "AN<2>" LOC = "--";  # Activación del display 2 = AN2
NET "AN<3>" LOC = "--";  # Activación del display 3 = AN3

# Entradas y salidas externas del puerto SPI
NET "SPI_DATA" LOC = "XX"; # Entrada de datos
NET "SPI_CS"   LOC = "YY"; # Salida de Chip Select
NET "SPI_CLK"  LOC = "ZZ"; # Salida de reloj
```

3. A continuación conecte los terminales del puerto SPI del MCP3201 (en la placa de inserción): SPI_CLK, SPI_DATA, SPI_CS y la masa (4 cables) al tablero de madera del puesto según los pines de salidas y entradas externas elegidos en el archivo de asociaciones (XX, YY y ZZ):



4. Si todo es correcto deberá aparecer la temperatura correcta en los displays de la izquierda, con un decimal. El punto decimal entre los displays 2 y 3 deberá estar encendido y deberá aparecer el carácter "C" en el display de la derecha. (Puede probar a calentar el sensor con los dedos para ver si la temperatura sube y baja adecuadamente).

DATOS PARA LA MEMORIA (Emplee la plantilla de memoria publicada)

1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes. No se evalúa el código si no está comentado.**
2. Adjunte una fotografía del circuito conectado a la FPGA y donde se lea correctamente la temperatura ambiente.
3. A continuación caliente el sensor con los dedos y adjunte otra fotografía donde se vea el circuito conectado a la FPGA y se lea correctamente la temperatura.
4. Incluya el código del archivo de asociaciones empleado para la síntesis.

MEMORIA y archivos VHDL: Fecha de entrega (sesión 11)

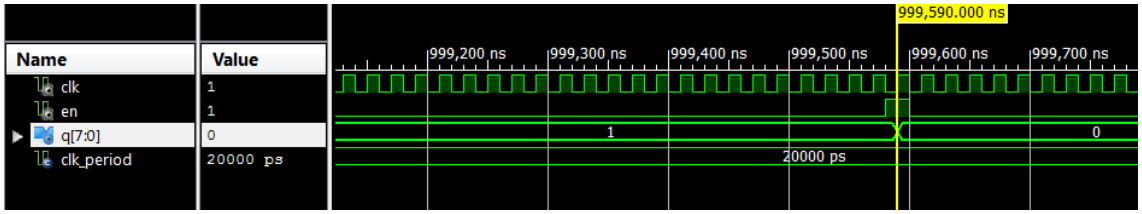
- **Realice la memoria empleando la plantilla publicada en MOODLE**.
- Respete todos los nombres de los archivos, entradas, salidas y señales que se indican en los módulos. EN OTRO CASO SU CÓDIGO NO PODRÁ SER EVALUADO.
- Comprima todos los ficheros *.VHD y *.UCF en un archivo ZIP de nombre **digital.zip** y entréguelo en el buzón de entrega. Estos archivos son los que se emplearán el día de la prueba de validación.
- Entregue también un documento con formato .DOC o .PDF conteniendo la memoria:

1. La memoria debe contener **todos** los elementos que se indican en las sesiones
2. Las expresiones matemáticas y fórmulas pueden entregarse realizadas a mano con letra clara y limpias. Deberá realizar una fotografía con bien expuesta y enfocada. También puede emplearse el editor de ecuaciones si se desea. La calidad de la presentación será tenida en cuenta en la evaluación.
3. El código VHDL deberá formatearse con tipo de letra **Consolas 10 o Courier New 9** para que aparezca debidamente alineado. Vea el siguiente ejemplo:

```
entity registro is
    Port ( CLK     : in  STD_LOGIC;                          -- Reloj
           ENABLE  : in  STD_LOGIC;                          -- Enable
           D       : in  STD_LOGIC_VECTOR (7 downto 0);      -- Entradas
           Q       : out  STD_LOGIC_VECTOR (7 downto 0));    -- Salidas
end registro;
```

4. Para realizar las capturas de pantalla del osciloscopio utilice la aplicación PROMAX disponible en el escritorio del ordenador que tiene en su puesto. Deberán verse claramente las indicaciones de las escalas en ambos ejes.
5. Respecto a las capturas de pantalla de las simulaciones VHDL, tenga en cuenta que debido al fondo negro de la pantalla del simulador es difícil que se vean las señales adecuadamente. Se recomienda utilizar la herramienta de recorte de Word y ampliar convenientemente la captura de pantalla para resaltar las regiones importantes.

La siguiente figura es un ejemplo de cómo se deben entregar las gráficas de simulación.



Referencias y bibliografía

[1] Sergio Franco, Diseño con Amplificadores Operacionales y Circuitos Integrados Analógicos, 3ª edición, McGraw-Hill, 2005.

[2] Norbert R. Malik, Circuitos Electrónicos: Análisis, Diseño y Simulación, Prentice-Hall, 1996.

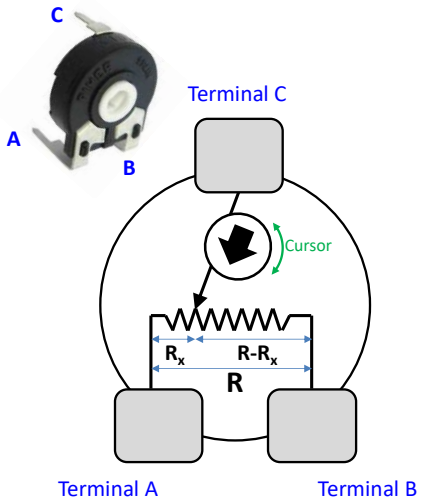
[3] John F. Wakerly, Digital Design (Principles and practices), 3ª y 4ª edición, Prentice Hall.

[4] Alan V. Oppenheim y Alan S. Willsky, Señales y Sistemas, 2ª edición, Prentice-Hall, 1998.

[5] A. Bruce Carlson, Communication systems: An Introduction to Signals and Noise in Electrical Communication, 3ª edidión, McGraw-Hill, 1986.

Anexo I : Funcionamiento de un potenciómetro

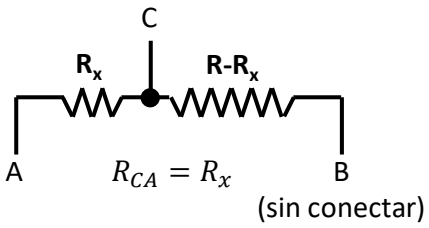
Un potenciómetro es una resistencia con un cursor que puede desplazarse a lo largo de su longitud. Tiene tres terminales: dos de ellos están conectados a los extremos de la resistencia y el tercero está conectado al cursor. Su esquema puede verse a continuación en la figura siguiente:



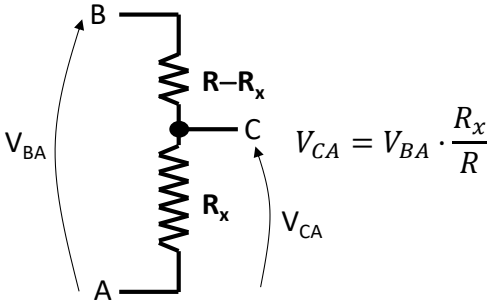
Según esta figura, si el potenciómetro tiene el valor de **R** ohmios, siempre se medirá una resistencia de **R** ohmios entre los terminales A y B. En cambio, si se mide la resistencia entre los terminales A y C, o B y C se leerá una resistencia variable en función de la posición del cursor. Necesariamente se cumple que si entre A y C se mide una resistencia de $\mathbf{R_x}$ ohmios, entonces entre B y C tiene que medirse una resistencia de $\mathbf{R-R_x}$ ohmios.

El potenciómetro tiene principalmente dos usos: **como resistencia variable**, o **como divisor de tensión variable**.

Para emplearlo como resistencia variable basta con conectar solamente dos de sus terminales (A y C o bien B y C), dejando el otro sin conectar. En este caso el valor de la resistencia será variable ($\mathbf{R_X}$) entre 0 y **R** ohmios en función de la posición del cursor. Vea la figura siguiente:

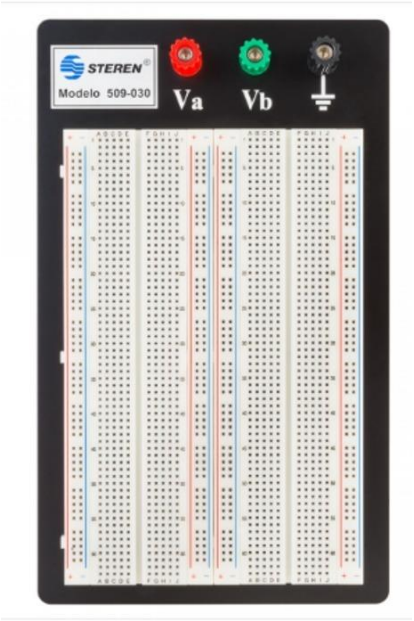


Para construir un divisor de tensión variable se conectan los tres terminales, de modo que la tensión entre los terminales A y C es una fracción variable de la tensión entre los terminales A y B, según la expresión de la figura siguiente:

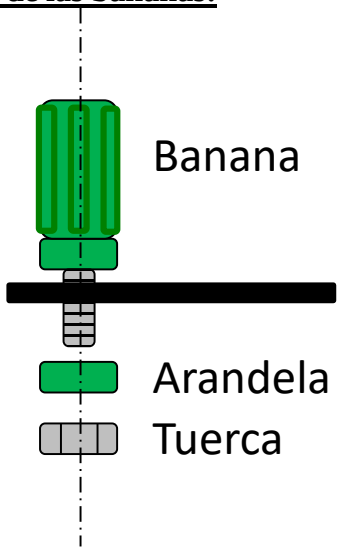


Anexo II : Tablero de inserción (protoboard) con las bananas incluidas

En un tablero de inserción con las bananas incluidas es más sencillo llevar la alimentación a la placa.



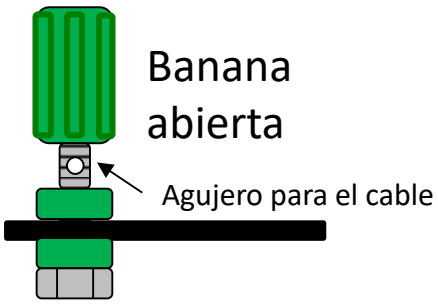
Montaje de las bananas:



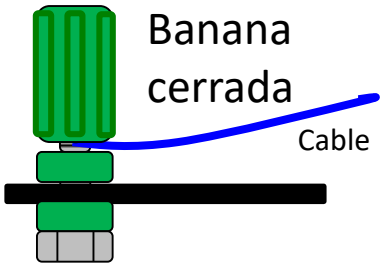
Apretar la tuerca para sujetar la banana a la placa.

Para conectar la alimentación:

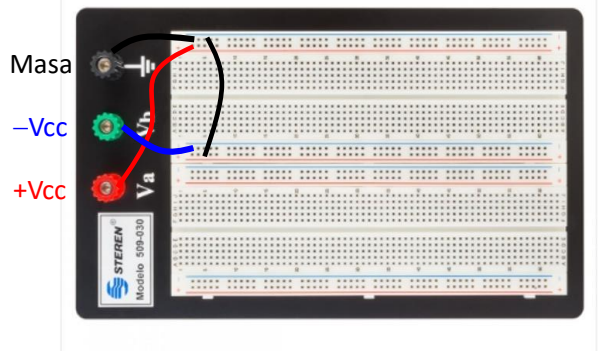
1. Abrir la banana:



2. Pelar el cable, utilizar el agujero para sujetarlo y cerrar la banana:



3. Conectar los cables al tablero:



4. Conectar las bananas a la fuente: