|  |  |  |
| --- | --- | --- |
|  | **Universidad Politécnica de Madrid**  **ETSI de Telecomunicación**  **Departamento de Ingeniería Electrónica** |  |

**Circuitos Electrónicos (CELT)**

**Plantilla de la memoria**

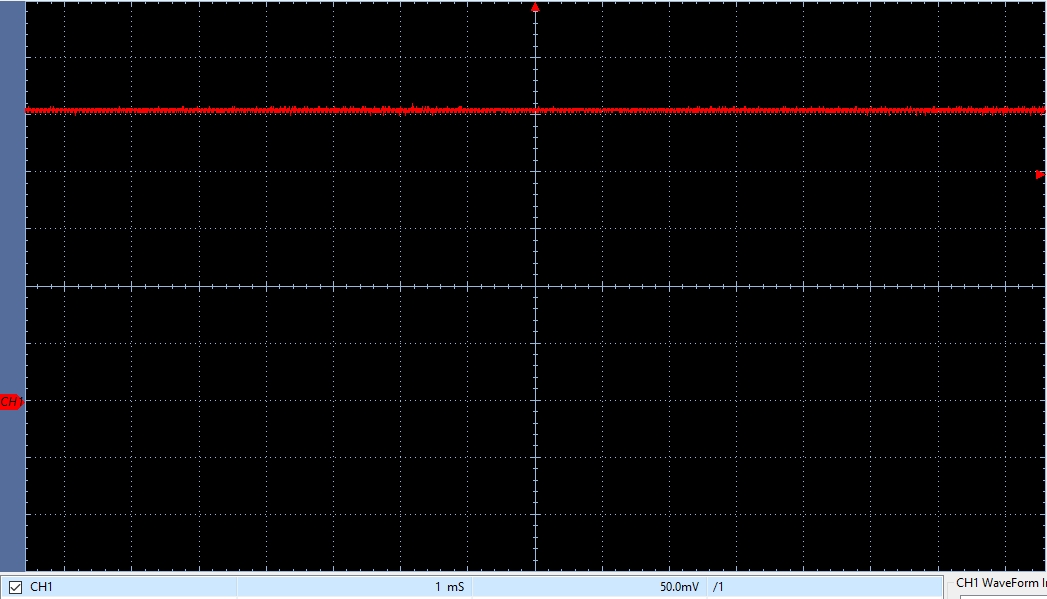
Curso 2023-2024

**Termómetro digital**

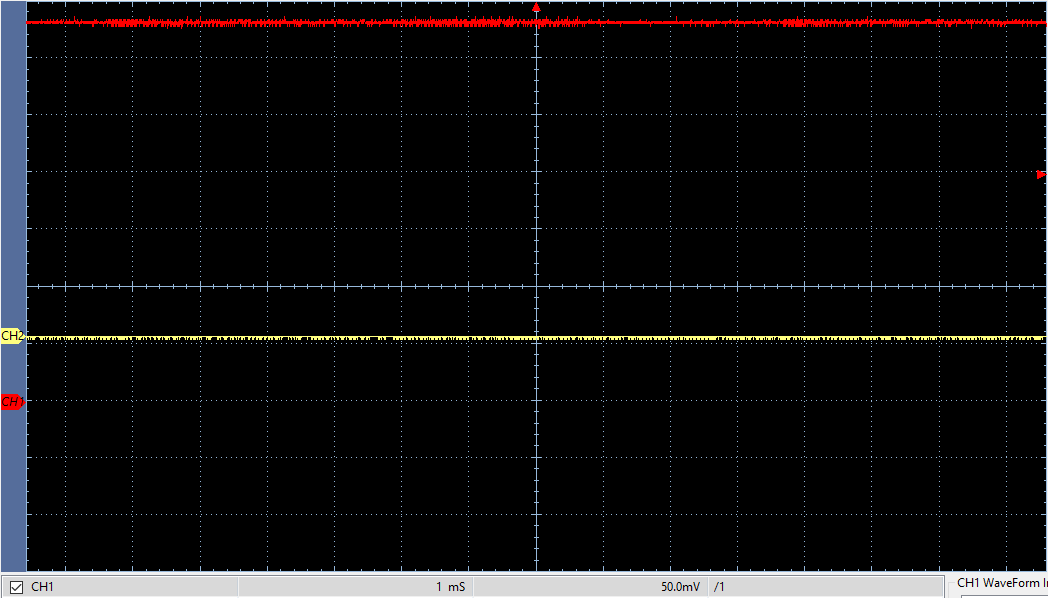
|  |  |
| --- | --- |
| Alumno/a 2 | Carlos Navarro Juan |
| Turno | MT15 |

MÓDULO1: El sensor de temperatura LM35

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 (Vm) de la tensión en la pantalla del osciloscopio).



1. 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

1. Detalle el cálculo de la función de transferencia del filtro paso bajo.

H(jf)= se sustituye para el caso que debe cumplir de fp=fc=10 Hz

En este caso al ser un filtro paso bajo la función de transferencia será H(j)=

1. Detalle el cálculo de los valores R y C.

Se sustituye para el caso que debe cumplir de fp=fc=10 Hz se sustituye en la función de transferencia y se aproxima a los siguientes valores que son los que se aplican al circuito(Se debe tener en cuenta que la resistencia y condensador se han aproximado de acuerdo a lo que hay en el mercado)

Los valores calculados son 15 kilohmios y 330 nanofaradios

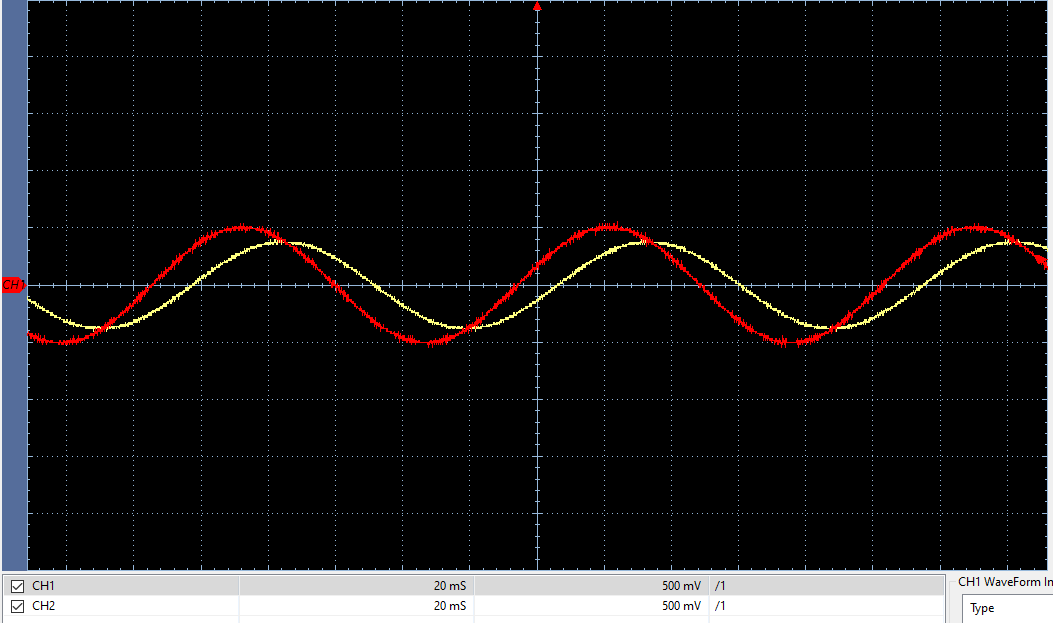
1. Dibuje los diagramas asintóticos de Bode en módulo y fase del filtro empleando las plantillas de las páginas siguientes. Indique la frecuencia del polo, la pendiente de la atenuación, el máximo valor de la ganancia y la pendiente de la fase.
2. Rellene la tabla para la caracterización del filtro e inclúyala.

(siguiente hoja)

1. 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
2. Indique el valor de la frecuencia de corte (*fc*) medida con precisión.

Valor de la frecuencia de corte(fc) 9,19Hz

1. 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 *vs*) 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 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 0.01 | 10.8 |
| 5 | 1 | 0.98 | 0.98 | -0.174 | 0.012 | 21.6 |
| 7 | 1 | 0.96 | 0.96 | -0.354 | 0.0124 | 31.2 |
| 8 | 1 | 0.93 | 0.93 | -0.630 | 0.013 | 37.44 |
| 9 | 1 | 0.84 | 0.84 | -1.514 | 0.0136 | 44.064 |
| 10 | 1 | 0.8 | 0.8 | -1.938 | 0.014 | 50.4 |
| 11 | 1 | 0.76 | 0.76 | -2.38 | 0.0124 | 49.1 |
| 12 | 1 | 0.7 | 0.7 | -3.08 | 0.0116 | 50.1 |
| 13 | 1 | 0.65 | 0.65 | -3.74 | 0.0108 | 50.5 |
| 14 | 1 | 0.62 | 0.62 | -4.16 | 0.010 | 50.4 |
| 15 | 1 | 0.6 | 0.6 | -4.44 | 0.0096 | 51.8 |
| 17 | 1 | 0.54 | 0.54 | -5.34 | 0.00925 | 56.61 |
| 20 | 1 | 0.45 | 0.45 | -6.92 | 0.009 | 64.8 |
| 30 | 1 | 0.36 | 0.36 | -8.88 | 0.0071 | 76.68 |
| 50 | 1 | 0.22 | 0.22 | -13.36 | 0.0045 | 81 |
| 70 | 1 | 0.15 | 0.15 | -16.48 | 0.0034 | 85.68 |
| 100 | 1 | 0.11 | 0.11 | -19.1 | 0.0024 | 86.4 |
| 200 | 1 | 0.05 | 0.05 | -26 | 0.00116 | 83.52 |
| 300 | 1 | 0.033 | 0.033 | -29.6 | 0.00084 | 90.72 |
| 500 | 1 | 0.02 | 0.02 | -33.97 | 0.0004 | 72 |
| 700 | 1 | 0.015 | 0.015 | -36.47 | 0.00034 | 85.68 |
| 1.000 | 1 | 0.01 | 0.01 | -40 | 0.00028 | 100.8 |

|  |  |
| --- | --- |
| Frecuencia de corte(*fc*) exacta medida | **9.19** |

**DIAGRAMAS ASINTÓTICOS DE BODE Y MEDIDAS, FILTRO PASO BAJO**

Máximo valor de ganancia 0dB

Pendiente de la atenuación es de -20dB/octava

Frecuencia del polo es de 10Hz

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Frecuencia del polo es de 10Hz

Pendiente de fase es de -45º/octava

Gráfico, Gráfico de líneas

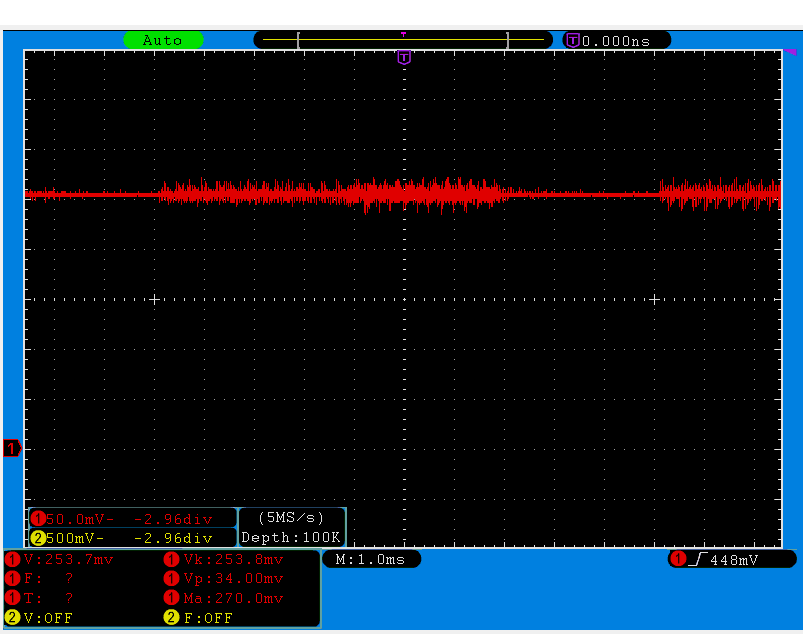
Descripción generada automáticamente

MÓDULO 3: Amplificador

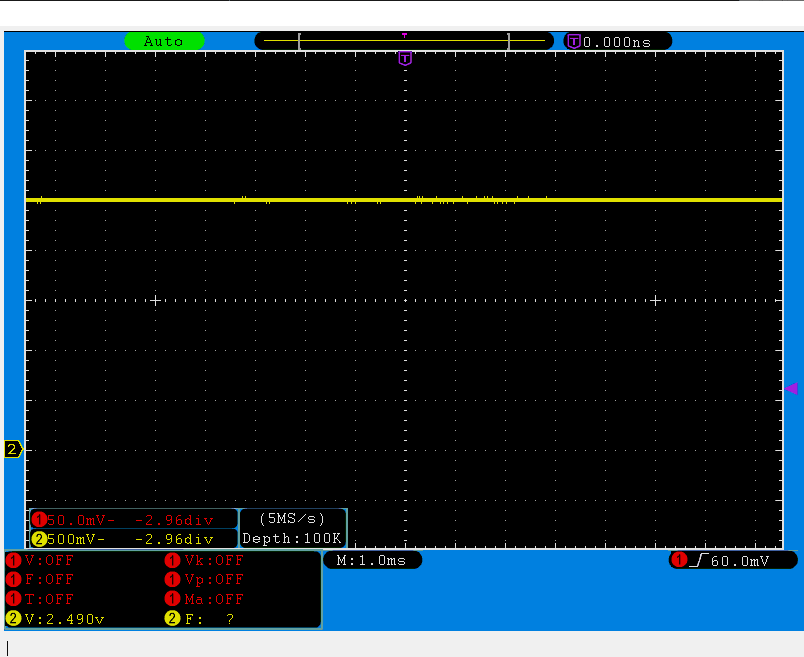
1. Detalle el cálculo de las resistencias R1 y R2.

Resolviendo el valor de las resistencias se obtiene R1=2500Ω R2=17500Ω

1. 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).



1. 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

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.

Computadora de escritorio sobre superficie de madera

Descripción generada automáticamente con confianza baja

1. 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.

Un hombre sentado frente a una computadora

Descripción generada automáticamente con confianza baja

MÓDULO 5: Módulo de visualización

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

Con respecto a la entrada del registro E0 a la salida queremos que aparezca fija la letra C, por lo que ponemos el valor 1100 que es la equivalencia de C (hexadecimal) en binario.

1. 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.**

Registro.vhd:

process (CLK)  
    begin  
      if (CLK'event and CLK='1') then   -- con cada flanco activo  
            if (ENABLE = '1') then  
            QS0 <= E0;  
            QS1 <= E1;  
            QS2 <= E2;  
            QS3 <= E3;  
            end if;  
            -- Realizar el proceso de captura  
  
        end if;  
  end process;  
  
     Q0 <= QS0;  
     Q1 <= QS1;  
     Q2 <= QS2;  
     Q3 <= QS3;

En la primera parte hacemos un if para que solo ejecute el código cuando la señal de reloj esté en el flanco de subida. Además, añadimos como condición que el ENABLE esté activo. Si las dos condiciones anteriores se dan entonces los valores que teníamos en la entrada se almacenan asignándolos a las señales QSX. Luego fuera de los ifs asignamos las señales QSX a las salidas para que aunque las entradas cambien solo varíen las salidas cuando se cumplan las situaciones anteriores.

MUX4x4.vhdl:

with S select Y <=  
 E0 when "00",  
 E1 when "01",  
 E2 when "10",  
 E3 when "11",  
 '0' when others;

En este caso, con el recurso with select, hacemos que Y sea una de las entradas dependiendo del valor de S de dos bits.

Refresco.vhdl:

 process (CLK)  
    begin  
     if (CLK'event and CLK='1') then  
  
 QS<=QS+1;  
  
  
        -- Parte secuencial, completar.  
  
     end if;  
     end process;  
  
 -- Parte combinacional  
 S<=STD\_LOGIC\_VECTOR (QS);  
AN<="0111" when QS="00" else  
 "1011" when QS="01" else  
 "1101" when QS="10" else  
 "1110" when QS="11";  
 DP <= '1' when QS="01" else  
 '0';  
 -- Completar cableado de:  
  
 -- S  
 -- AN  
 -- DP

Primero hacemos que la señal incremente 1 en cada flanco de subida de la señal de reloj, después asignamos QS a S. AN activa cada uno de los displays, que se activarán según la señal QS. DP se trata del display del punto decimal que en este caso solo lo queremos activo cuando se activa el segundo display por la izquierda es decir cuando QS=01.AN se utiliza para seleccionar cada uno de los 4 displays de siete segmentos dependiendo del valor que tenga QS. Según donde este situado el 0 en la señal AN se utiliza un display u otro.

Visualización.vhdl:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

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 dcimas

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)); -- Activacin individual

end visualizacion;

architecture a\_visualizacion of visualizacion is

-- COMPONENTES

component decodBCDa7s

Port ( BCD : in STD\_LOGIC\_VECTOR (3 downto 0);

SEGMENTOS : out STD\_LOGIC\_VECTOR (0 to 6));

end component;

-- Definir otros componentes necesarios

component 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 component;

component 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); -- Seal de control

Y : out STD\_LOGIC\_VECTOR (3 downto 0)); -- Salida

end component;

component 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 component;

-- SEALES

-- Definir posibles seales necesarias

signal qe3,qe2,qe1,qe0,y: STD\_LOGIC\_VECTOR (3 downto 0); -- creamos las seales qe que unen las salidas 'Q' del registro con las entradas 'E' del multiplexor e 'y' que une la salida 'Y' y la entrada del decodificador

signal s: STD\_LOGIC\_VECTOR(1 downto 0); -- creamos la seal 's' que une la salida de refresco y la seal de entrada de control del multiplexor

begin

-- esta parte es el cableado de todos los componentes

U1 : decodBCDa7s port map (y,SEG7); -- establecemos a la entrada la seal 'y' y la salida a la salida de visualizacion 'SEG7'.

U2: registro port map(CLK,ENABLE,DECENAS,UNIDADES,DECIMAS,"1100",qe3,qe2,qe1,qe0); -- establecemos a las entradas las seales de entrada 'CLK','ENABLE','DECENAS','UNIDADES','DECIMAS', a la entrada 'E3' la letra C y a las salidas las seales 'qe'.

U3: mux4x4 port map(qe3,qe2,qe1,qe0,s,y); -- asignamos a las entradas las seales 'qe' y las salidas 's' e 'y'

U4: refresco port map(CLK,s,DP,AN); -- asignamos a la entrada la seal de entrada 'CLK' y la seal 's' y a la salida las entradas de salida 'DP' y 'AN'

end a\_visualizacion;

Este módulo consiste en juntar todos los módulos anteriores, por lo que hay que establecer cuales van a ser los componentes antes del begin y luego indicamos cuales son las salidas y las entradas utilizando en algunos casos señales.

MODULO 6: Divisor del reloj

1. Detalle el cálculo de la constante XXXX.

El valor XXXX lo hemos calculado dividiendo la frecuencia de la señal de entrada entre la de salida y dividiendo entre dos que es la distancia entre el flanco de suida y el de bajada. Dando lugar al valor 25000-1 para poder tener una frecuencia de 1000Hz.

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

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

--

-- Divisor de la frecuencia del reloj a 1 KHz.

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

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 : unsigned (31 downto 0);

signal frec\_div : STD\_LOGIC;

begin

process(CLK\_50MHz)

begin

if (CLK\_50MHz'event and CLK\_50MHz='1') then

contador<=contador+1;

if (contador=25000-1) then -- semiperiodo de 500 us

contador<=(others=>'0');

frec\_div<=not frec\_div;

end if;

end if;

end process;

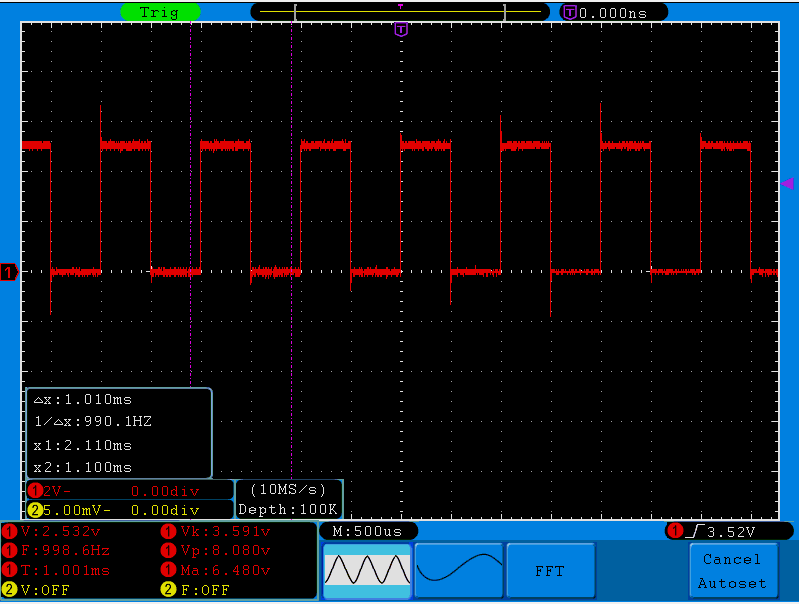
CLK<=frec\_div;

end a\_div\_reloj;

**EXPLICACIÓN:**

Este programa tiene una señal de entrada CLK\_50 y una señal de salida CLK y lo que se intenta es cambiar la frecuencia de la señal de entrada a una frecuencia de 1000Hz. En el código se crean señales internas como son contandor y frec\_div. Se inicia el proceso CLK\_50MHz donde Podemos encontrar un if que lo que hace es activarse en el flanco de subida para después incrementar el contador cuando se llegue a 25000-1 se pone el contandor a cero y se invierte la salida, por último a la señal de CLK se le pasa la señal interna frec\_div. Con esto se consigue cambiar la frecuencia a la señal de entrada.

1. 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

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

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 conversin A/D

DATOS\_ADC : out STD\_LOGIC\_VECTOR (15 downto 0)); -- Dato ledo 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); -- indicamos los diferentes estados que puede tener el autmata

signal ST : STATE\_TYPE := ESPERA\_1S; -- seal de tipo 'STATE\_TYPE' con valor inicial 'ESPERA\_1S'

signal cont : unsigned (15 downto 0):=(others=>'0'); -- seal que utilizamos a modo de contador

signal data : unsigned (15 downto 0):=(others=>'0'); -- seal que posteriormente se asignarar a la salida 'DATOS\_ADC'

begin

process (CLK) -- iniciamos el proceso de 'CLK'

begin

if (CLK'event and CLK='1') then -- En cada flanco de subida del reloj

case ST is -- sentencia case, es decir que ejecuta el cdigo en funcin de lo que valga 'ST'

when ACTIVAR\_CS => -- cuando 'ST' vale 'ACTIVAR\_ST'

cont<=(others=>'0'); -- ponemos el contador a 0

ST<=CLK0; -- 'ST' pasa a ser 'CLK0'

when CLK0 => -- cuando 'ST' sea 'CLK0'

data <= data(14 downto 0) & SPI\_DATA; -- concatenamos los primeros 15 valores de 'data' con 'SPI\_DATA' para que los valores de 'SPI\_DATA' vaya desplazandose a la izquierda

cont <= cont+1; -- sumamos 1 al contador

ST<=CLK1; -- 'ST' pasa a ser 'CLK1'

when CLK1 => -- cuando 'ST' sea 'CLK1'

if cont = 15 then -- si el contador es igual a 15

ST <= DESACT\_CS; -- ST pasa a ser 'DESACT\_CS'

else -- en otro caso

ST <=CLK0; -- 'ST' vuelve a ser 'CLK0'

end if; -- fin del if

when DESACT\_CS => -- cuando 'ST' vale 'DESACT\_CS'

cont <=(others=>'0'); -- se reinicia el contador

ST <= ESPERA\_1S; -- 'ST' pasa a ser 'ESPERA\_1S'

when ESPERA\_1S => -- cuando 'ST' sea 'ESPERA\_1S'

cont <= cont + 1; -- se suma 1 al contador

data <= (others=>'0'); -- 'data' se reinicia

if cont = 1000 then -- si el contador es igual a 1000, es decir que tienen que ocurrir 1000 flancos de subida de reloj para que pase a 'ACTIVAR\_CS'

ST <= ACTIVAR\_CS; -- 'ST' pasa a ser 'ACTIVAR\_CS'

else -- en otro caso

ST <=ESPERA\_1S; -- 'ST' sigue siendo 'ESPERA\_1S'

end if; -- fin del if

end case; -- fin del case

end if; -- fin del if

end process; -- fin del proceso

-- PARTE COMBINACIONAL (CABLEADO DE LAS SALIDAS)

SPI\_CS<='1' when (ST=ESPERA\_1S or ST=DESACT\_CS) else '0'; -- 'SPI\_CS' estar activo a nivel alto si 'ST' es igual a 'ESPERA\_1S' o a 'DESACT\_CS'

DATOS\_ADC<=std\_logic\_vector(data) ; -- asignamos a la salida 'DATOS\_ADC' la seal 'data' en formato 'STD\_LOGIC\_VECTOR'

FIN\_CONV<='1' when ST = DESACT\_CS else '0'; -- 'FIN\_CONV' ser '1' cuando 'ST' sea 'DESACT\_CS'

SPI\_CLK<= '1' when (ST = CLK1 or ST=DESACT\_CS) else '0'; -- 'SPI\_CLK' ser '1' cuando 'ST' sea 'CLK1' o 'DESACT\_CS'

end a\_aut\_control;

EXPLICACIÓN:

El Código anterior diseña un automata que controla la lectura de datos desde un ADC, en este caso se utiliza un diseño de máquina de estados finitos (en los comentarios del Código se indica de manera más precisa que hace cada cosa)

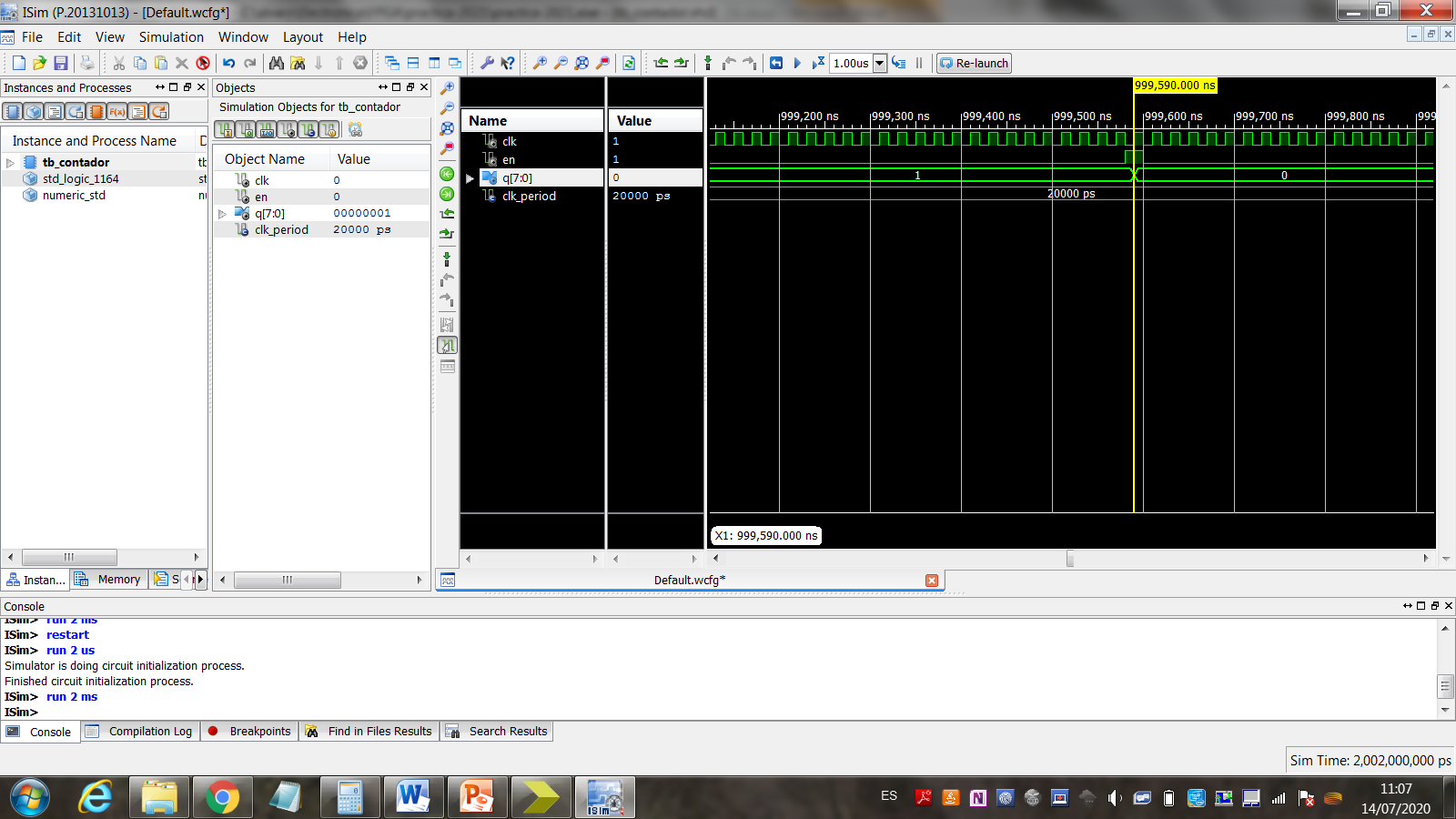
1. Incluya una captura de pantalla de la simulación donde pueda verse con precisión el tiempo entre 995 y 1035 ms y donde puedan apreciarse las transiciones de las señales de salida. Coloque dos cursores de tiempo: uno en el flanco de bajada de la salida SPI\_CS y otro en el flanco de subida de la salida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

Interfaz de usuario gráfica

Descripción generada automáticamente

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

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.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

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 punto fijo con 6 bits decimales

end ADC\_a\_TEMP;

architecture a\_ADC\_a\_TEMP of ADC\_a\_TEMP is

constant FILTRO : unsigned (15 downto 0):= "0000111111111000"; -- creamos una constante llamada FILTRO para filtrar la seal de entrada datos

signal datos : unsigned (15 downto 0):=(others=>'0'); -- Datos ledos del conversor 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

signal datosaux : unsigned(31 downto 0):=(others=>'0');

begin

datos<= unsigned(DATOS\_ADC) and FILTRO; -- datos de entrada filtrados

-- Para pasar a C se trata de multiplicar por 50 y dividir entre 4096

-- Para multiplicar datos\*50, hacemos: datos\*32+datos\*16+datos\*2

datosaux <= "0000000000000000"&datos; -- hacemos que datos sea de 32 bits en vez de 16

datosx32 <= datosaux sll 5; -- desplazamos la seal a la izquierda 5 posiciones, lo que equivale a multiplicar por 32(2^5)

datosx16 <= datosaux sll 4; -- desplazamos la seal a la izquierda 4 posiciones, lo que equivale a multiplicar por 16(2^4)

datosx2 <= datosaux sll 1; -- desplazamos la seal a la izquierda 1 posicin, lo que equivale a multiplicar por 2(2^1)

datosx50<= (datosx32 + datosx16 + datosx2); -- sumamos todas las multiplicaciones

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

TEMP <= STD\_LOGIC\_VECTOR(datosx50(21 downto 6));

end a\_ADC\_a\_TEMP;

**EXPLICACIÓN:**

En este caso se diseña un sistema que lo hace es pasar los datos del ADC a la señal de salida TEMP(En los comentarios del código se indica cada proceso y lo que hace cada señal)

1. Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.

Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente**MODULO 9: TEMP a BCD**

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

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)); -- Dcimas 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<= unsigned(TEMP(15 downto 6)); -- asignamos a ENT la parte entera de TEMP

DEC<= unsigned(TEMP(5 downto 0)); -- asignamos a DEC la parte decimal de DEC

-- Las decenas se determinan en funcin del intervalo de valores donde se

-- encuentre la temperatura tal y como indica la tabla utilizando el recurso when else

DECENAS<= "1001" when ENT >=90 else

"1000" when ENT >=80 else

"0111" when ENT >=70 else

"0110" when ENT >=60 else

"0101" when ENT >=50 else

"0100" when ENT >=40 else

"0011" when ENT >=30 else

"0010" when ENT >=20 else

"0001" when ENT >=10 else

"0000";

-- La seal s\_unid se determinan restando las decenas al valor de la temperatura tal y como indica la tabla utilizando el recurso when else

s\_unid<= ENT - 90 when ENT >=90 else

ENT - 80 when ENT >=80 else

ENT - 70 when ENT >=70 else

ENT - 60 when ENT >=60 else

ENT - 50 when ENT >=50 else

ENT - 40 when ENT >=40 else

ENT - 30 when ENT >=30 else

ENT - 20 when ENT >=20 else

ENT - 10 when ENT >=10 else

ENT;

-- Para la parte decimal vemos cul es la dcima (entre 10 valores de 0 a 9)

-- segn el valor binario de 6 bits entre 0 y 64 utilizando sentencias when else

DECIMAS<="0000" when DEC <6 else

"0001" when DEC <12 else

"0010" when DEC <18 else

"0011" when DEC <24 else

"0100" when DEC <30 else

"0101" when DEC <36 else

"0110" when DEC <42 else

"0111" when DEC <48 else

"1000" when DEC <54 else

"1001";

UNIDADES<= STD\_LOGIC\_VECTOR(s\_unid(3 downto 0));-- Las unidades son los bits menos significativos de s\_unid

end a\_TEMP\_a\_BCD;

**EXPLICACIÓN:**

En este caso, lo que se hace es pasar el valor de la temperatura de 6 bits decimales a códigos BCD (En los comentarios del código se indica que hace cada señal)

2.Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Gráfico

Descripción generada automáticamente con confianza media

Escala de tiempo

Descripción generada automáticamente con confianza baja

Gráfico

Descripción generada automáticamente con confianza media

MÓDULO10: Circuito digital completo

1.Detalle el código VHDL debidamente comentado.**Los comentarios son muy importantes. No se evalúa el código si no está comentado.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

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 seleccin 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

-- SE ESTABLECEN LOS COMPONENTES DE TERMOMETRO

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;

component 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 dcimas

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)); -- Activacin individual

end component;

component 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 punto fijo con 6 bits decimales

end component;

component 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)); -- Dcimas en BCD

end component;

component 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 conversin A/D

DATOS\_ADC : out STD\_LOGIC\_VECTOR (15 downto 0)); -- Dato ledo del ADC

end component;

-- POSIBLES SEALES NECESARIAS

signal CLK,enable : STD\_LOGIC; -- seales de 1 bit CLK, salida del div\_reloj y entrada en visualizacin y aut\_control y enable, salida del autmata de control y entrada de visualizacin

signal dece,ud,deci : STD\_LOGIC\_VECTOR(3 downto 0); -- seales de 4 bits

signal datos,temp : STD\_LOGIC\_VECTOR(15 downto 0);

begin

U1 : div\_reloj port map (CLK\_50MHz, CLK);

U2: visualizacion port map (enable,dece,ud,deci,CLK,SEG7,DP,AN);

U3: ADC\_a\_TEMP port map (datos,temp);

U4: TEMP\_A\_BCD port map (temp,dece,ud,deci);

U5: aut\_control port map (CLK,SPI\_DATA,SPI\_CLK,SPI\_CS,enable,datos);

end a\_termometro;

**EXPLICACIÓN:**

Este módulo lo que hace es unir los anteriores para crear el termómetro digital, en los comentarios del código se indica que hace cada señal.

2.Adjunte una fotografía del circuito conectado a la FPGA y donde se lea correctamente la temperatura ambiente.

Un circuito electrónico

Descripción generada automáticamente con confianza media

3.A continuación caliente el sensor con los dedos y adjunte otra fotografía donde se veael circuito conectado a la FPGA y se lea correctamente la temperatura.

Un circuito electrónico

Descripción generada automáticamente con confianza baja

4.Incluya el código del archivo de asociaciones empleado para la síntesis.

# Reloj principal del sistema  
NET "CLK\_50MHz" LOC = "M6"; # Señal de reloj del sistema de 50 MHz  
  
# Conexiones de los DISPLAYS  
NET "SEG7<0>" LOC = "L14"; # señal = CA  
NET "SEG7<1>" LOC = "H12"; # Señal = CB  
NET "SEG7<2>" LOC = "N14"; # Señal = CC  
NET "SEG7<3>" LOC = "N11"; # Señal = CD  
NET "SEG7<4>" LOC = "P12"; # Señal = CE  
NET "SEG7<5>" LOC = "L13"; # Señal = CF  
NET "SEG7<6>" LOC = "M12"; # Señal = CG  
NET "DP" LOC = "N13"; # Punto decimal  
  
# Señales de activación de los displays  
NET "AN<0>" LOC = "F12"; # Activación del display 0 = AN0  
NET "AN<1>" LOC = "J12"; # Activación del display 1 = AN1  
NET "AN<2>" LOC = "M13"; # Activación del display 2 = AN2  
NET "AN<3>" LOC = "K14"; # Activación del display 3 = AN3  
  
#Entradas y salidas  
NET "SPI\_DATA" LOC="B2"; # Entrada de datos  
NET "SPI\_CS" LOC="A9";  # Salida de Chip Select  
NET "SPI\_CLK" LOC="A10";   # Salida de reloj