<table>
<tr><td colspan="3" rowspan="2"> 
UNIVERSIDAD POLITÉCNICA DE MADRID
E.T.S.I.S. Telecomunicación</td><td colspan="5">APELLIDOS:</td></tr>
<tr><td colspan="3">NOMBRE:</td><td colspan="2">DNI:</td></tr>
<tr><td colspan="3">ASIGNATURA: Diseño Digital 2</td><td colspan="2">Bloque 1</td></tr>
<tr><td colspan="3"></td><td colspan="5"></td></tr>
<tr><td colspan="3">Fecha</td><td>Curso</td><td>Grupo</td><td colspan="4">Notas Parciales</td><td>Nota Final</td></tr>
<tr><td>2</td><td>7</td><td>2021</td><td>Tercero</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

ADVERTENCIAS PARA LA REALIZACIÓN DEL EXAMEN

- Rellene **AHORA** los datos personales que deben figurar en esta hoja.
- Mientras dure el examen deberá exponer su D.N.I. encima de la mesa.
- **NO SE ADMITIRÁN** exámenes escritos a lapicero ni con tinta roja o verde.
- **COMPRUEBE** que su ejemplar del examen consta de **3** ejercicios en **5** páginas numeradas.
- En este examen **NO PUEDEN UTILIZARSE CALCULADORAS, LIBROS, APUNTES NI DISPOSITIVOS DE TELECOMUNICACIÓN**. Retírelos ahora de la mesa.
- La duración del examen es de **60 minutos**.

Esta hoja se ha dejado en blanco intencionadamente

Ejercicio 1 (3 puntos, 15 min). El código del fichero *procesador_medida.vhd*, ubicado en la carpeta *ejercicio_1* corresponde al modelo del módulo *procesador_medida* utilizado en el prototipo de medida de temperatura y humedad realizado en la actividad 4 del bloque 1.

Califique las siguientes afirmaciones, que se refieren a dicho módulo como verdaderas (V) o falsas (F). Utilice el cajetín al lado del número de cada pregunta; si se equivoca, táchelo y escriba la respuesta correcta a la izquierda. Las respuestas correctas se calificarán con +0.3 puntos, las incorrectas con -0.3 puntos y las no contestadas con 0 puntos.

Nota: En la carpeta *docu* dispone, entre otros, de un documento con el modelo de registros del Periférico I2C.

F 1.- El módulo *procesador_medida* realiza una medida de temperatura y humedad cada 0.25 segundos.

V 2.- El módulo *procesador_medida* está basado en un autómata de Mealy con salidas registradas.

F 3.- Las transacciones I2C se configuran para que se utilice el modo de "lectura de todos los bytes". Pero si no se utilizara este modo, entonces la asignación de la línea 83 tendría que ser: numb_bytes <= "000", manteniéndose igual la asignación de la línea 87.

F 4.- En todos los casos, durante la ejecución del código de la línea 112, el bus de direcciones *add* tiene la dirección 3 y la señal *we* está a '0'.

V 5.- En el módulo *procesador_medida*, los datos de temperatura y humedad se leen utilizando un registro de desplazamiento que desplaza hacia la izquierda los bytes que se van leyendo.

F 6.- La sentencia de la línea 159 multiplica el valor en *temperatura* por la constante 164.

F 7.- La sentencia de la línea 167 divide el valor en *humedad* entre la constante 65535.

F 8.- En la actividad no presencial en la que se realizó el diseño del prototipo de medida de temperatura y humedad se calculó un valor para el genérico del temporizador para realizar la simulación a escala. El valor calculado debía proporcionar el tiempo suficiente entre dos tics para que se llevaran a cabo las transacciones I2C más largas (sin escalar). Si el *procesador_medida* se rediseñara para no utilizar el modo de "lectura de todos los bytes" entonces las transacciones serían más cortas y por tanto se podría utilizar un valor más pequeño para el genérico antes mencionado.

Ejercicio 2 (3 puntos, 15 min). La carpeta *docu* contiene el *datasheet* del sensor de temperatura y humedad HC1000 empleado en la realización del MEDTH, los esquemas de la tarjeta DECA MAX10 y el *device overview* de la familia MAX10. Consúltelos, si lo considera necesario, para indicar si las siguientes afirmaciones son verdaderas (V) o falsas (F). Utilice el cajetín al lado del número de cada pregunta; si se equivoca, táchelo y escriba la respuesta correcta a la izquierda. Las respuestas correctas se calificarán con +0.3 puntos, las incorrectas con -0.3 puntos y las no contestadas con 0 puntos.

F 1.- Una transferencia I2C compuesta por los bytes x80 x02 x00 x15 configuraría al sensor de temperatura en modo de captura secuencial de temperatura y humedad con una resolución de 11 bits para ambas magnitudes.

V 2.- Una transferencia I2C sobre el sensor de temperatura compuesta por los bytes x80 x01 situaría el puntero del sensor de temperatura en la dirección correspondiente al registro de humedad.

F 3.- Si fuera necesario, el MEDTH podría realizar medidas de temperatura y humedad cada 5 ms, en lugar de cada 500 ms como se hace en el diseño actual, manteniendo el resto de parámetros de configuración de la medida.

V 4.- Los pines *SCL* y *SDA* de la interfaz I2C del MEDTH están conectadas a sendas resistencias de pull-up con un valor nominal de 2200 Ω.

F 5.- Los pines SCL y SDA se configuran en el MEDTH con tecnología Schmitt Trigger para reducir el nivel de ruido que generan cuando funcionan como salidas (mientras el nivel de ruido se mantenga por debajo de la tensión de histéresis no produce ningún efecto).

V 6.- Para que el LED4 de la DECA luzca es necesario activar la salida C4 de la FPGA con un nivel bajo.

V 7.- Los condensadores de la página 9 de los esquemas de la DECA son todos para desacoplo de las alimentaciones de la FPGA.

V 8.- Los circuitos U24 y U33 son cambiadores de nivel para generar relojes de 50 MHz con niveles de 2.5 V y 1.5 V respectivamente a partir de un reloj a 3.3V.

Ejercicio 3 (4 puntos, 30 min). La carpeta *ejercicio_3* contiene el modelo del módulo gen_SCL que se utilizó para el diseño del interfaz_i2C en la actividad 3, así como su correspondiente test.

Nota: Para resolver este ejercicio, cree un proyecto ModelSim e incluya en él los ficheros que se le proporcionan.

Analice el código del test *test_gen_SCL*. Observe las líneas 27 a 36 en las que se definen una serie de constantes para la verificación de tiempos. Después podrá encontrar el proceso que genera el reloj, la instanciación del DUT, el pull-up y el resto de estímulos del test. Finalmente, a partir de la línea 115, se incluye el código de autoverificación.

1.- El proceso que comienza a partir de la línea 121 chequea que la duración de los niveles bajo y alto de SCL se corresponde con los límites especificados en el estándar I2C (FAST). Usted debe completar el proceso para que, además, chequee que la frecuencia del reloj I2C es menor o igual que la máxima, 400 KHz (1.5 puntos)

2.- A continuación, a partir de la línea 147, deberá escribir un monitor que compruebe que, mientras el reset asíncrono esté deshabilitado, los niveles en SCL sean '0' o 'H'. Hágalo utilizando solo una sentencia assert concurrente (2.5 puntos)

Una vez haya terminado suba a *Moodle* la nueva versión de test_gen_SCL.

<table>
<tr><td colspan="3" rowspan="2"> 
UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S.I.S. de Telecomunicación</td><td colspan="6">APELLIDOS:</td></tr>
<tr><td colspan="4">NOMBRE:</td><td colspan="2">DNI:</td></tr>
<tr><td colspan="3">ASIGNATURA: Diseño Digital 2</td><td colspan="3">Diseño (escrito)</td></tr>
<tr><td colspan="6">TITULACIÓN: ELECTRÓNICA DE COMUNICACIONES</td></tr>
<tr><td colspan="3">Fecha</td><td>Curso</td><td>Grupo</td><td colspan="4">Notas Parciales</td><td>Nota Final</td></tr>
<tr><td>02</td><td>07</td><td>2021</td><td>Tercero</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

ADVERTENCIAS PARA LA REALIZACIÓN DE ESTA PARTE DEL EXAMEN

- Rellene **AHORA** los datos personales que deben figurar en esta hoja.
- Mientras dure el examen deberá exponer su D.N.I. encima de la mesa.
- **NO SE ADMITIRÁN** exámenes escritos a lapicero ni con tinta roja o verde.
- **COMPRUEBE** que su ejemplar del examen consta de 2 ejercicios en 10 páginas numeradas.
- En este examen **NO PUEDEN UTILIZARSE CALCULADORAS, LIBROS, APUNTES NI DISPOSITIVOS DE TELECOMUNICACIÓN**. Retírelos ahora de la mesa.
- La duración de esta parte del examen es de **60 minutos**.
- La calificación de esta parte del examen tendrá un peso del 50% sobre la calificación total de BT2.

Esta hoja se ha dejado en blanco intencionadamente

Ejercicio 1	Controlador de Teclado		
		7 puntos	40 minutos

El código adjunto corresponde al modelo de un circuito que gestiona un teclado de 8 **filas** y 8 **columnas**. Proporcionando en su salida la identificación de la tecla pulsada, **cod_tecla**, y una señal que se mantiene a nivel alto mientras se está pulsando cualquiera de las teclas, **pulso_on**. Dispone de dos señales adicionales de entrada que funcionan a modo de sendos tics utilizados en la temporización del controlador de teclado, **tic_1** y **tic_2**.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ctrl_teclado is
port(nRst:      in      std_logic;
     clk:       in      std_logic;
     tic_1:     in      std_logic;
     tic_2:     in      std_logic;
     columnas: in      std_logic_vector(7 downto 0);
     filas:     buffer  std_logic_vector(7 downto 0);
     cod_tecla:buffer  std_logic_vector(X downto 0);
     pulso_on: buffer  std_logic);

end entity;

architecture rtl of ctrl_teclado is
  signal col_0, col_1: std_logic_vector(7 downto 0);

begin
  -- Fragmento 1

  process(clk, nRst)
  begin
    if nRst = '0' then
      col_0 <= (others => '1');
      col_1 <= (others => '1');

    elsif clk'event and clk = '1' then
      col_0 <= columnas;
      if tic_1 = '1' then
        col_1 <= col_0;

      end if;
    end if;
  end process;
  -- Fragmento 1

  -- Fragmento 2

  process(clk, nRst)
  begin
    if nRst = '0' then
      cod_tecla(X downto 3) <= (others => '0');

    elsif clk'event and clk = '1' then
      if tic_2 = '1' and col_0 = X"FF" and pulso_on = '0' then
        if cod_tecla(X downto 3) /= 7 then
          cod_tecla(X downto 3) <= cod_tecla(X downto 3) + 1;

        else
          cod_tecla(X downto 3) <= (others => '0');

        end if;
      end if;
    end if;
  end process;

  -- Fragmento 2
```

```
  -- Fragmento 3
  filas <= X"FE" when ...

  cod_tecla(2 downto 0) <= "000" when ...

  pulso_on <= ...
  -- Fragmento 3
end rtl;
```

1.- Conteste a las siguientes cuestiones:

a) Determine el número de bits que debe tener la salida "cod_tecla". Justifique su respuesta. (0.5 puntos)

Se necesita detectar hasta 64 códigos de tecla diferentes, ya que el teclado dispone de 8 filas y 8 columnas. Así pues, esta salida deberá constar de 6 bits.

b) Identifique los subsistemas modelados en el fragmento 1 del código. Deduzca la función que realizan en el contexto del circuito controlador de teclado. (0.8 puntos)

Se trata de dos circuitos. Por una parte, un registro de 8 bits utilizado para la sincronización de las entradas asíncronas correspondientes a las columnas del teclado. Por otra parte, un registro de 8 bits con habilitación, cuya función es, por un lado, la sincronización de la referida entrada y por otro lado la de suprimir los rebotes.

c) Identifique el subsistema modelado en el fragmento 2 del código. Deduzca la función que realiza en el contexto del circuito controlador de teclado. (0.8 puntos)

Se trata de un contador de módulo ocho con habilitación. Utilizado para efectuar el barrido de las ocho salidas de fila para el teclado.

d) Complete la sentencia concurrente de asignación a la señal **filas**, dentro del fragmento 3 de código. (0.8 puntos)

```
filas <= X"FE" when cod_tecla(5 downto 3) = 0 else
         X"FD" when cod_tecla(5 downto 3) = 1 else
         X"FB" when cod_tecla(5 downto 3) = 2 else
         X"F7" when cod_tecla(5 downto 3) = 3 else
         X"EF" when cod_tecla(5 downto 3) = 4 else
         X"DF" when cod_tecla(5 downto 3) = 5 else
         X"BF" when cod_tecla(5 downto 3) = 6 else
         X"7F" when cod_tecla(5 downto 3) = 7 else
         X"FF";
```

e) Complete la sentencia concurrente de asignación a la señal **cod_tecla(2 downto 0)**, dentro del fragmento 3 de código. (0.8 puntos)

```
cod_tecla(2 downto 0) <= "000" when col_1 = X"FE" else
                         "001" when col_1 = X"FD" else
                         "010" when col_1 = X"FB" else
                         "011" when col_1 = X"F7" else
                         "100" when col_1 = X"EF" else
                         "101" when col_1 = X"DF" else
                         "110" when col_1 = X"BF" else
                         "111" when col_1 = X"7F" else
                        "000";
```

f) Complete la sentencia concurrente de asignación a la señal **pulso_on**, dentro del fragmento 3 de código. (0.8 puntos)

```
pulso_on <= '0' when col_1 = X"FF" else '1';
```

g) Obtenga la frecuencia de los tics 1 y 2 suponiendo que los rebotes duran menos de 200 us y que se desea realizar 100 barridos completos de todas las filas cada segundo. Justifique su respuesta. (1.5 puntos)

El tic_1 está involucrado en la implementación del circuito anti-rebote. Por lo tanto, su frecuencia debe ser:

1/200us = 5 KHz

El tic_2 está involucrado en la implementación del circuito de barrido de filas. Por lo tanto, su frecuencia debe ser:

*100 barridos * 8 filas = 800 Hz*

h) Modele una salida, **tipo**, que indique si la tecla pulsada es un número, una letra, otro tipo de carácter imprimible o un carácter de control, sabiendo que: (1 punto)

- ✓ Las teclas numéricas están conectadas a la fila 0 completa y a las 2 primeras columnas de la fila 1.
- ✓ Las teclas que representan letras son 27 y están conectadas a las 6 columnas restantes de la fila 1, a las filas 2 y 3 completas y a las 5 primeras columnas de la fila 4.
- ✓ El resto de las teclas de la fila 4 y todas las de las filas 5 y 6 son caracteres imprimibles.
- ✓ Las de la fila 7, son teclas de control.

```
tipo <= "00" when cod_tecla(5 downto 3) = 0 or
                 (cod_tecla(5 downto 3) = 1  and
                  cod_tecla(2 downto 0)) < 2 else  -- numero

        "01" when cod_tecla(5 downto 3) < 4 or
                 (cod_tecla(5 downto 3) = 4  and
                  cod_tecla(2 downto 0)) < 5 else  -- letra

        "10" when cod_tecla(5 downto 3) < 7  else  -- caracter

        "11";                                      -- control
```

Ejercicio 2	Detector de tecla pulsada		
		3 puntos	20 minutos

El código adjunto corresponde al modelo de un circuito que recibe las señales **pulso_on** y **cod_tecla,** provenientes del circuito del ejercicio 1, y que genera dos salidas, **s_p** y **l_p** que cualifican el pulso recibido por la entrada. Además, retarda en un ciclo de reloj el código de la tecla pulsada y la entrega en su salida **tecla**. Dispone además de una entrada adicional, **tic**, que se emplea en el modelo con funciones de temporización.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity detector is
port(nRst:      in     std_logic;
     clk:       in     std_logic;
     tic:       in     std_logic;
     cod_tecla: in     std_logic_vector(5 downto 0);
     pulso_on:  in     std_logic;
     tecla:     buffer std_logic_vector(5 downto 0);
     s_p:       buffer std_logic;
     l_p:       buffer std_logic);

end entity;

architecture rtl of detector is
  signal cnt: std_logic_vector(6 downto 0);

begin

  process(clk, nRst)
  begin
    if nRst = '0' then
      cnt <= (others => '0');
      tecla <= (others => '0');
      s_p <= '0';
      l_p <= '0';

    elsif clk'event and clk = '1' then
        tecla <= cod_tecla;
      if tic = '1' then
        if pulso_on = '1' and cnt /= 100 then
          cnt <= cnt + 1;

        elsif pulso_on = '0' and cnt = 100 then
          cnt <= (others => '0');
          l_p <= '1';

        elsif pulso_on = '0' and cnt > 0 and cnt < 100 then
          cnt <= (others => '0');
          s_p <= '1';

        else
          s_p <= '0';
          l_p <= '0';

        end if;
      end if;
    end if;
  end process;

end rtl;
```

1.- Analice el modelo para contestar a las siguientes cuestiones:

a) Explique cuál será la respuesta del circuito ante: (1.5 puntos)

-- Una pulsación con una duración de 10 tics

Activa la salida s_p

-- Una pulsación con una duración de 100 tics

Activa la salida l_p

-- Una pulsación de 5000 tics

Activa la salida l_p

b) Explique la utilidad de este circuito una vez conectado al circuito controlador de teclado del ejercicio 1. (0.5 puntos)

*Detecta si la pulsación en el teclado ha sido larga o corta, informando en la salida **s_p** o **l_p** de forma correspondiente. Informa del código identificador de la tecla pulsada, en su salida **tecla**.*

c) Indique la duración de los pulsos activos de **s_p** y **l_p**. Indique si habría que modificar, o no, el circuito en caso de que se desee que la duración de los pulsos sea de un ciclo de reloj. Si su respuesta es afirmativa, especifique la modificación que debería realizarse para conseguirlo, si no lo es, justifique su respuesta. (1 punto)

*Los pulsos en **s_p** y **l_p** siempre duran el tiempo entre dos pulsos consecutivos en la entrada **tic**.*

Para asegurar que siempre dura un ciclo de reloj, se proponen dos opciones:

a) Conformar las salidas
b) Realizar la puesta a '0' de ambas salidas fuera de la rama "if tic='1'". Cualificando mediante la condición de que pulso_on sea '0' y cnt sea 0.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ctrl_teclado is
port(nRst:      in     std_logic;
     clk:       in     std_logic;
     tic_1:     in     std_logic;
     tic_2:     in     std_logic;
     columnas:  in     std_logic_vector(7 downto 0);
     filas:     buffer std_logic_vector(3 downto 0);
     cod_tecla:buffer  std_logic_vector(4 downto 0);
     pulso_on:  buffer std_logic);

end entity;

architecture rtl of ctrl_teclado is
  signal col_0, col_1: std_logic_vector(7 downto 0);

begin

  process(clk, nRst)
  begin
    if nRst = '0' then
      col_0 <= (others => '1');
      col_1 <= (others => '1');

    elsif clk'event and clk = '1' then
      col_0 <= columnas;
      if tic_1 = '1' then
        col_1 <= col_0;

      end if;
    end if;
  end process;


  process(clk, nRst)
  begin
    if nRst = '0' then
      cod_tecla(4 downto 3) <= (others => '0');

    elsif clk'event and clk = '1' then
      if tic_2 = '1' and col_0 = X"FF" and pulso_on = '0' then
        if cod_tecla(4 downto 3) /= 3 then
          cod_tecla(4 downto 3) <= cod_tecla(4 downto 3) + 1;

        else
          cod_tecla(4 downto 3) <= (others => '0');

        end if;
      end if;
    end if;
  end process;


  filas <= X"E" when cod_tecla(4 downto 3) = 0 else
           X"D" when cod_tecla(4 downto 3) = 1 else
           X"B" when cod_tecla(4 downto 3) = 2 else
           X"7" when cod_tecla(4 downto 3) = 3 else
                   X"F";


  cod_tecla(2 downto 0) <= "000" when col_1 = X"FE" else
                           "001" when col_1 = X"FD" else
                           "010" when col_1 = X"FB" else
                           "011" when col_1 = X"F7" else
                           "100" when col_1 = X"EF" else
                           "101" when col_1 = X"DF" else
                           "110" when col_1 = X"BF" else
                           "111" when col_1 = X"7F" else
```

```
                              "000";

  pulso_on <= '0' when col_1 = X"FF" else '1';


end rtl;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity timers is port(
  clk:   in std_logic;
  nRst:  in std_logic;
  tic_1: buffer std_logic;
  tic_2: buffer std_logic
);
end entity;

architecture rtl of timers is
  signal cnt_tic1: std_logic_vector(15 downto 0);
  signal cnt_tic2: std_logic_vector(3 downto 0);

begin

-- contador de modulo 100.000.000/2.000 = 50.000
process(clk, nRst)
begin
  if nRst = '0' then
    cnt_tic1 <=(others => '0');

  elsif clk'event and clk = '1' then
    if tic_1 = '1' then
      cnt_tic1 <=(others => '0');

    else
      cnt_tic1 <= cnt_tic1 + 1;

    end if;
  end if;
end process;

tic_1 <= '1' when cnt_tic1 = 49999 else '0';

-- contador de modulo 100.000.000/500 = 200.000 = 50.000 x 4
process(clk, nRst)
begin
  if nRst = '0' then
    cnt_tic2 <=(others => '0');

  elsif clk'event and clk = '1' then
    if tic_1 = '1' then
      cnt_tic2 <= cnt_tic2 + 1;

    end if;
  end if;
end process;

tic_2 <= '1' when cnt_tic2 = 3 and tic1 = '1' else '0';

end rtl;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ctrl_operacion is port(
  clk:      in std_logic;
  nRst:     in std_logic;
  start_op: in std_logic;
  cod_op:   in std_logic_vector(1 downto 0);
  res_rdy:  buffer std_logic
);
end entity;

architecture rtl of ctrl_operacion is
  signal cnt : std_logic_vector(2 downto 0);

begin

process(clk, nRst)
begin
  if nRst <= '0' then
    cnt <= (others => '1');

  elsif clk'event and clk = '1' then
    if start_op = '1' then
      if cod_op = 3 then
        cnt <= "100";

      elsif cod_op = 2 then
        cnt <= "010";

      else
        cnt <= "000";

      end if;
    elsif cnt /= 0 then
      cnt <= cnt -1;

    end if;
  end if;
end process;

 res_rdy <= '1' when cnt = 0 else '0';

end rtl;
```