<table>
<tr><td colspan="3" rowspan="4"> 
UNIVERSIDAD POLITÉCNICA DE MADRID

ETSIS de Telecomunicación</td><td colspan="6">APELLIDOS:</td></tr>
<tr><td colspan="4">NOMBRE:</td><td colspan="2">DNI:</td></tr>
<tr><td colspan="4">ASIGNATURA: DISEÑO DIGITAL I</td><td colspan="2">Bloque: II (Escrito)</td></tr>
<tr><td colspan="6">TITULACIÓN: ☐ Electrónica de Comunic. ☐ Sonido e Imagen
☐ Sistemas de Telecom. ☐ Telemática</td></tr>
<tr><td colspan="3">Fecha</td><td>Curso</td><td colspan="5">Calificación de los ejercicios</td><td>Nota Final</td></tr>
<tr><td>28</td><td>06</td><td>2017</td><td>TERCERO</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

ADVERTENCIAS PARA LA REALIZACIÓN DE LA PRIMERA PARTE DEL EXAMEN (Ejercicios 1, 2, 3 y 4)

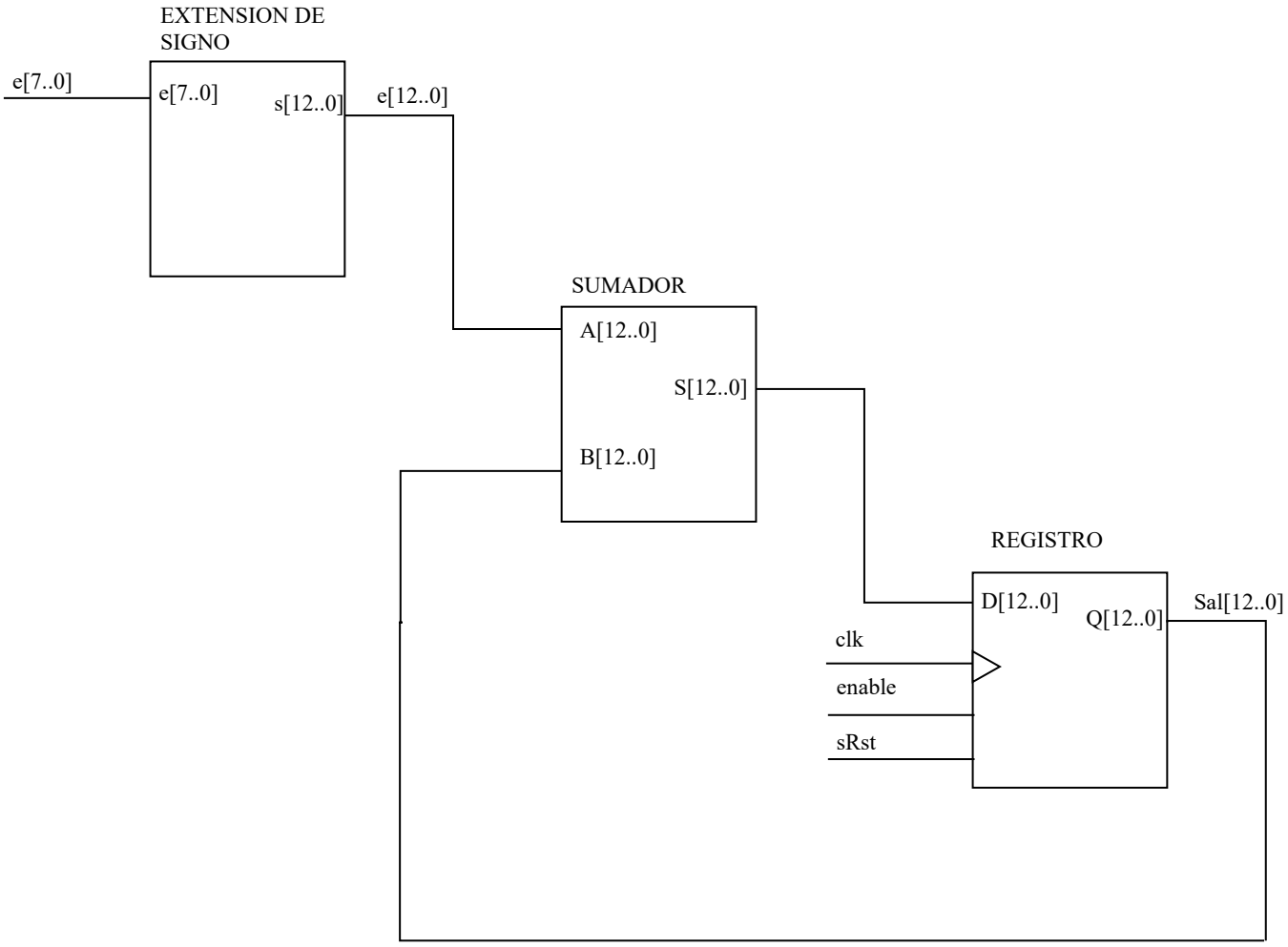
- Rellene **AHORA** los datos personales que deben figurar en esta hoja.
- Mientras dure el examen deberá exponer su D.N.I. encima de la mesa.
- **NO SE ADMITIRÁN** exámenes escritos a lapicero ni con tinta roja o verde.
- **COMPRUEBE** que su ejemplar del examen consta de **4** ejercicios en **15** 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 **110 minutos**.

Esta hoja se ha dejado en blanco intencionadamente

Ejercicio 1	Subsistemas digitales		
		0.5 puntos	10 minutos

Dibuje el diagrama de bloques de un de un acumulador para datos de 8 bits con signo. El acumulador debe ser capaz de ofrecer una salida en el rango 4095 a -4096. Deberá disponer de una entrada de habilitación y un *reset* síncrono, activos ambos a nivel alto.

Nota: realice el conexionado utilizando etiquetas



Ejercicio 2	Subsistemas Digitales		
		0,7 puntos	20 minutos

El siguiente el código VHDL modela el funcionamiento de un subsistema complejo. Determine el tipo de subsistema, indicando sus características específicas. Además, explique detalladamente su funcionamiento. El reloj del circuito tiene una frecuencia de 1 MHz.

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

entity sistema is
port(clk:    in     std_logic;
     nRst:   in     std_logic;
     ent_A:  in     std_logic;
     ent_B:  in     std_logic_vector(2 downto 0);
     salida: buffer std_logic);
end entity;

architecture rtl of sistema is
  signal cuenta1: std_logic_vector(13 downto 0);
  signal cuenta2: std_logic_vector(2 downto 0);
  signal tic:  std_logic;

begin
  process(clk, nRst)
  begin
    if nRst = '0' then
      cuenta1 <= (0 => '1', others => '0');

    elsif clk'event and clk = '1' then
      if tic = '1' or ent_A = '1' then
          cuenta1 <= (0 => '1', others => '0');
      else
          cuenta1 <= cuenta1 + 1;
      end if;
    end if;
  end process;

  tic <= '1' when cuenta1 = 10000 else '0';

  process(clk, nRst)
  begin
    if nRst = '0' then
      cuenta2 <= "111";

    elsif clk'event and clk = '1' then
      if ent_A ='1' then
       cuenta2 <= (others => '0');

      elsif tic = '1' then
       if salida = '1' then
          cuenta2 <= cuenta2 + 1;

       else
          cuenta2 <= (others => '1');

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

  salida <= '1' when cuenta2 < ent_B else '0';
end rtl;
```

Se trata de un monoestable redisparable con anchura de impulso programable. Con un periodo monoestable comprendido entre 10 ms y 70 ms.

Se ha implementado mediante un divisor de frecuencia para obtener un tic de 10ms, dividiendo la señal de reloj de 1MHz entre 10000, el cual se utiliza para habilitar la cuenta de un contador de módulo 8. La señal de salida de pulso se toma en la salida de un comparador.

El funcionamiento es el siguiente:
cuando se produce un pulso en la entrada ent_A, que es la señal de disparo, el contador del tic se inicializa para sincronizar su cuenta con la medida exacta de 10ms. El contador del monoestable también se inicializa a cero. Desde ese momento, a cada uno de los tics de este contador se incrementa hasta tomar el valor igual que la entrada ent_B, en ese momento la cuenta pasa a valer 7, valor que se mantendrá hasta que el disparo vuelva a producirse. La salida permanece a nivel alto mientras el contador está contando.

Ejercicio 3	Análisis de modelos. Diseño		
		2 puntos	35 minutos

El siguiente código VHDL modela el funcionamiento de un circuito aritmético. Analice el código presentado y responda a las preguntas que se plantean.

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

entity sumador is
port(clk:         in    std_logic;
     nRst:        in    std_logic;
     WE:          in    std_logic;
     Dir_WR:      in    std_logic_vector(1 downto 0);
     Din:         in    std_logic_vector(7 downto 0);
     Dir_RD:      in    std_logic_vector(1 downto 0);
     inicio:      in    std_logic;  -- Pulso a nivel alto de un ciclo de
                                    -- reloj de duración
     Dout:        buffer std_logic_vector(7 downto 0);
     busy:        buffer std_logic);
end entity;

architecture rtl of sumador is
  signal segunda:       std_logic;
  signal wr_addr:       std_logic_vector(1 downto 0);
  signal oper_A:        std_logic_vector(7 downto 0);
  signal oper_B:        std_logic_vector(7 downto 0);
  signal reg0:          std_logic_vector(7 downto 0);
  signal reg1:          std_logic_vector(7 downto 0);
  signal reg2:          std_logic_vector(7 downto 0);
  signal reg3:          std_logic_vector(7 downto 0);
  signal Dato_reg3:     std_logic_vector(7 downto 0);

begin
  -- Inicio FRAGMENTO 1
  process(clk, nRst)
  begin
    if nRst = '0' then
      reg0 <= (others => '0');
      reg1 <= (others => '0');
      reg2 <= (others => '0');
      reg3 <= (others => '0');
    elsif clk'event and clk = '1' then
      if WE = '1' or busy = '1' then
        case wr_addr is
            when "00" =>
              reg0 <= Din;
            when "01" =>
              reg1 <= Din;
            when "10" =>
              reg2 <= Din;
            when "11" =>
              reg3 <= Dato_reg3;
            when others =>
              null ;
        end case;
      end if;
    end if;
  end process;
```

```
 process(reg0, reg1, reg2, reg3, Dir_RD)
 begin
   case DIr_RD is
     when "00" =>
       Dout <= reg0;
     when "01" =>
       Dout <= reg1;
     when "10" =>
       Dout <= reg2;
     when others =>
       Dout <= reg3;
   end case;
 end process;

 -- Final FRAGMENTO 1


 -- Inicio FRAGMENTO 2

 process(clk, nRst)
 begin
   if nRst = '0' then
     segunda <= '0';

   elsif clk'event and clk = '1' then
     segunda <= inicio;

   end if;
 end process;

 -- Final FRAGMENTO 2

 oper_a <= reg0  when segunda = '0' else
           reg2;

 oper_b <= reg1  when segunda = '0' else
           reg3;

 wr_addr <= "11"    when inicio = '1' or segunda = '1' else
             Dir_WR;

 Dato_reg3 <= oper_a + oper_b when inicio = '1' or segunda = '1' else
              Din;

 busy <=  '1' when inicio = '1' or segunda = '1' else
          '0';

end rtl;
```

1. ¿Qué subsistema se modela en el código comprendido entre las etiquetas “Inicio FRAGMENTO 1” y “Final FRAGMENTO 1? **(0.3 puntos)**

Se trata de un banco de cuatro registros de 8 bits, con puerto de escritura y puerto de lectura. La peculiaridad de este circuito sobre un banco de registro convencional es que la escritura del registro que ocupa la tercera dirección no se escribe desde el exterior sino que se escribe a partir de la información de una señal interna.

2. ¿Qué subsistema se modela en el código comprendido entre las etiquetas “Inicio FRAGMENTO 2” y “Final FRAGMENTO 2? **(0.2 puntos)**

Se trata de un flipflop tipo D con entrada de reset asíncrono.

3. Explique detalladamente qué operación aritmética realiza el circuito. Especificando cuáles son los operandos y cómo podemos escribirlos, y dónde se almacena el resultado de la operación y cómo podemos leerlo. **(0.4 puntos)**

Realiza la suma aritmética de tres datos de ocho bits, los contenidos en los registros reg0, reg1 y reg2. Para escribir estos operandos debemos escribirlos primero, mediante la activación de la señal WE, poniendo el dato que queremos escribir en la señal Din y la dirección correspondiente al registro en la entrada Dir_WR. El resultado de la suma se almacena en reg3, de donde puede ser leído poniendo la dirección 3 en la señal Dir_RD.

4. Explique detalladamente la funcionalidad de las señales “inicio”, “busy” y “segunda”. **(0.5 puntos)**

***inicio**: marca mediante un pulso a nivel alto de un ciclo de reloj de duración el inicio de la operación de suma.*

***busy**: informa de que el circuito se encuentra realizando la operación de suma y que el dato aún no está disponible en la salida. Asimismo, se utiliza esta señal para habilitar la escritura en el registro reg3, con el resultado parcial y el final.*

***segunda**: indica que se está realizando la suma del resultado de la primera operación, reg0+reg1, almacenado en reg3, con el contenido de reg2.*

5. Como puede observar, el circuito no tiene en cuenta ningún tipo de error en la operación que se realiza. Si consideramos que los datos con los que opera el circuito están codificados en *binario natural*, describa las modificaciones que habría que efectuar sobre el código para modelar una salida adicional del circuito, **overflow**, que indique la ocurrencia de tal error. **(0.6 puntos)**

Habría que realizar las siguientes modificaciones.

*Añadimos una salida más a la interfaz: **overflow***

Realizar un sumador con acarreo para lo cual modificamos el ancho de Dato_reg3 y modificamos el sumador.

*Creamos un fliflop, con la señal **overflow**, que se borraría al inicio de operación y se pondría a uno cuando se produjera un acarreo en la primera o en la segunda de las sumas.*

Ejercicio 4	Análisis de modelos. Diseño		
		2,8 puntos	45 minutos

El siguiente el código VHDL modela el funcionamiento de un circuito que se utiliza para generar números aleatorios de 8 bits, en BCD en el rango 0 a 99, los cuales pueden obtenerse en la salida **numero**. Este circuito dispone, además, de una entrada, **gen**, mediante la cual se ordena la generación del número aleatorio. Otra entrada, **pre**, se utiliza para preparar al sistema para la generación de un nuevo número. Estas dos últimas entradas son activas a nivel alto.

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

entity gen_secuencia is
  port (clk:        in     std_logic;
        nRst:       in     std_logic;
        gen:        in     std_logic;
        pre:        in     std_logic;
        numero:     buffer std_logic_vector(7 downto 0)
              );
end gen_secuencia;


architecture rtl of gen_secuencia is
  signal c_u:       std_logic;
  signal c_d:       std_logic;
  signal u:         std_logic_vector(3 downto 0);
  signal d:         std_logic_vector(3 downto 0);
  signal gen_reg:   std_logic_vector(1 downto 0);
  signal pre_reg:   std_logic_vector(1 downto 0);
  signal genera:    std_logic;
  signal prepara:   std_logic;
  signal cuenta:    std_logic;

  type   t_estado is (inicio, preparado);
  signal estado: t_estado;

begin
       -- Inicio Fragmento 1
       process (clk, nRst)
       begin
         if nRst = '0' then
              gen_reg <= (others => '0');
              pre_reg <= (others => '0');

         elsif clk'event and clk = '1' then
              gen_reg(1) <= gen;
              pre_reg(1) <= pre;
              gen_reg(0) <= gen_reg(1);
              pre_reg(0) <= pre_reg(1);

         end if;
       end process ;

       genera <= gen_reg(1) and (not gen_reg(0));
       prepara <= pre_reg(1) and (not pre_reg(0));

       -- Final Fragmento 1
```

```
-- Inicio Fragmento 2

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

  elsif clk'event and clk = '1' then
       if (c_u = '1' and cuenta = '1') or prepara = '1' then
         u <= (others => '0');

       elsif cuenta = '1' then
         u <= u + 1;

       end if;
  end if;
end process ;

c_u <= '1' when u = 9;

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

  elsif clk'event and clk = '1' then
       if (c_d = '1' and cuenta = '1') or prepara = '1' then
         d <= (others => '0');

       elsif c_u = '1' and cuenta = '1' then
         d <= d + 1;

       end if;
  end if;
end process;

c_d <= '1' when d = 9 and c_u = '1';

-- Final Fragmento 2

-- Inicio Fragmento 3
process(clk, nRst)
begin
       if nRst = '0' then
              estado <= inicio;

       elsif clk'event and clk = '1' then
              case estado is
                     when inicio =>
                            if prepara = '1' then
                                   estado <= preparado;
                            end if;

                     when preparado =>
                            if genera = '1' then
                                   estado <= inicio;
                            end if;

              end case;
       end if;
end process;

cuenta <= '1' when estado = preparado else '0';
-- Final Fragmento 3
```

```
	-- Inicio Fragmento 4
	process(clk, nRst)
	begin
		if nRst = '0' then
			numero <= (others => '0');

		elsif clk'event and clk = '1' then
			if estado = preparado and genera = '1' then
				numero <= d&u;
			end if;
		end if;
	end process;
	-- Inicio Fragmento 4

end architecture rtl;
```

El circuito generador de números aleatorios basa su aleatoriedad en la selección del número mediante un procedimiento *no determinista*. Concretamente, se utilizarán las señales provenientes de sendos pulsadores para generar las señales **gen** y **pre**. De esta forma, mediante la combinación de una frecuencia suficientemente elevada para el conteo y la utilización de un evento asíncrono exterior al sistema para capturar un valor, se consigue que el usuario sea incapaz de predecir, a priori, el valor que será seleccionado por el generador de números. Otra característica esencial que debe poseer el generador es la *equiprobabilidad* de los números, esto es, todos los valores pertenecientes al rango deben tener la misma probabilidad de ser elegidos. Si analiza el código anteriormente mostrado observará que modela un circuito que cumple ambas características

1. Identifique los subsistemas que componen el sistema de generación de números aleatorios, detallando completamente las características de cada uno de ellos. (**0.4 puntos**)

***Fragmento 1**: Conformador de pulso*
Está duplicado para la señal gen y la señal pre. Sincroniza estas señales mediante sendos registros de dos bits y crea las señales genera y prepara que proporcionan un pulso a nivel alto de un ciclo de reloj de duración en cada flanco e subida de la señal correspondiente

***Fragmento 2**: Contador BCD de dos dígitos.*
*Dispone de entrada de habilitación, **cuenta**; y reset síncrono, **prepara**.*

***Fragmento 3**: Autómata de dos estados: **inicio** y **preparado**.*

***Fragmento 4**: Registro de ocho bits.*
Con señal de habilitación y reset asíncrono.

2. En el **fragmento 2** del código del modelo hay infracciones a las reglas de modelado de sistemas digitales simples.
    a. Identifique y explique las infracciones cometidas (**0.2 puntos**)

No se determina la salida para todas las combinaciones de entrada en el modelado de un circuito combinacional

```
c_u <= '1' when u = 9;
c_d <= '1' when d = 9 and c_u = '1';
```

    b. Corrija el error (escriba el código de sustitución) (**0.1 puntos**)

```
c_u <= '1' when u = 9 else '0';

c_d <= '1' when d = 9 and c_u = '1' else '0';
```

    c. Explique el efecto que dicha infracción tendría sobre el funcionamiento del modelo en una simulación (**0.1 puntos**)

Las señales de acarreo se ponen a uno la primera vez que se cumple la condición y ya no se vuelven a poner a cero, cuando la condición no se cumple.

3. Indique qué cambios realizaría en el código para que el rango de números aleatorios fuese entre 0 y 59. Escriba el código alternativo. (**0.3 puntos**)

```
c_d <= '1' when d = 5 and c_u = '1' else '0';
```

4. El siguiente código pretende sustituir al comprendido en el fragmento 3. ¿Modela este código un circuito con idéntico comportamiento al que sustituye? Justifique su respuesta. (**0.5 puntos**)

```
process(clk, nRst)
begin
	if nRst = '0' then
		cuenta <= '0';

	elsif clk'event and clk = '1' then
		if cuenta = ‘0’ and prepara = '1' then
			cuenta <= '1';

		elsif cuenta = ‘1’ and genera = '1' then
			cuenta <= '0';

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

Sí, en lo que respecta al propio fragmento3, ya que modela el funcionamiento de un flipflop con reset y preset síncrono, que es el circuito modelado por el autómata que contiene el fragmento 3. Sin embargo, hay que considerar que el fragmento 4 también se ve afectado al utilizar la señal “estado”.

5. Se desea realizar un circuito en el que solamente con un pulsador se dé la orden de generación del número aleatorio. Es decir, el sistema contaría solamente con una entrada, **gen**, proveniente de un pulsador, para dar la orden de preparación (con su flanco de subida, y de generación (con su flanco de bajada). Realice las modificaciones oportunas en el código para conseguir el nuevo comportamiento que se ha descrito, proponiendo el código alternativo. (**0.6 puntos**)

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

  elsif clk'event and clk = '1' then
       gen_reg(1) <= gen;
       gen_reg(0) <= gen_reg(1);

  end if;
end process ;

prepara <= gen_reg(1) and (not gen_reg(0));
genera <= (not gen_reg(1)) and gen_reg(0);
```

6. Explique los cambios (no debe escribir código) que habría que realizar para que el circuito generase números aleatorios entre -99 y 99. Deberá utilizar una codificación en BCD para la magnitud (0 a 99) y una señal adicional para el signo (0, positivo; 1, negativo) (**0.6 puntos**)

Nota: tenga en cuenta el deseado principio de equiprobabilidad, es decir, todos los números del rango deben tener la misma probabilidad de ser elegidos.

Añadir al contador de unidades una entrada síncrona para la puesta al valor "0001". EL contador de las decenas no requiere ninguna modificación. Añadir una salida en la interfaz para el signo. Esta señal deberá conmutar cada vez que el contador alcanza el valor 99. Para eso deberemos modelar un flipflop tipo T, cuya entrada será el final de la cuenta c_d = '1'.

Las inicializaciones síncronas deben estar controladas por el valor de magnitud de los contadores BCD y por el valor del signo. El funcionamiento será tal que cuando se llega al valor +99 el contador debe pasar al 1, al tiempo que el signo se vuelve negativo (signo = '1'), mientras que cuando el contador llega al -99, el contador debe pasar al valor 0 y el signo se volverá positivo (signo = '0').