

 <p><b>UNIVERSIDAD POLITÉCNICA DE MADRID</b></p> <p><i>ETSIS de Telecomunicación</i></p>				<b>APELLIDOS:</b>					
				<b>NOMBRE:</b>				<b>DNI:</b>	
				<b>ASIGNATURA:</b> DISEÑO DIGITAL I				<b>Bloque:</b> II (Escrito)	
				<b>TITULACIÓN:</b> <input type="checkbox"/> Electrónica de Comunic. <input type="checkbox"/> Sistemas de Telecom. <input type="checkbox"/> Sonido e Imagen <input type="checkbox"/> Telemática					
<b>Fecha</b>			<b>Curso</b>	<b>Calificación de los ejercicios</b>					<b>Nota Final</b>
28	06	2017	TERCERO						

## 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

<b>Ejercicio 1</b>	<b>Subsistemas digitales</b>		
		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

<b>Ejercicio 2</b>	<b>Subsistemas Digitales</b>		
		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;

```



<b>Ejercicio 3</b>	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)**
  
  
  
  
  
  
  
  
  
  
2. ¿Qué subsistema se modela en el código comprendido entre las etiquetas “Inicio FRAGMENTO 2” y “Final FRAGMENTO 2”? **(0.2 puntos)**
  
  
  
  
  
  
  
  
  
  
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)**



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

<b>Ejercicio 4</b>	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)

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

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)**

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;
```

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)**

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.