PUESTO Nº:	

UNIVERSIDAD POLITÉCNICA DE MADRID			**APELLIDOS:**						
			NOMBRE:					**DNI:**	
			ASIGNATURA: DISEÑO DIGITAL I					**Bloque:** I	
ETSIS de Telecomunicación			**TITULACIÓN:** ☐ Electrónica de Comunic. ☐ Sonido e Imagen ☐ Sistemas de Telecom. ☐ Telemática						
Fecha			***Curso***	**Calificaciones Parciales**					**Nota Final**
7	*11*	*2022*	*TERCERO*						

SEGUNDA PARTE

- Indique **AHORA en la esquina superior derecha** el número del puesto que ocupa.
- 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.
- **Cuando le indiquen que puede abrir el examen, COMPRUEBE** que su ejemplar del examen consta de **5** 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 **90 minutos**.
- Cree una carpeta *EXAMEN* y, dentro de ella, un proyecto *Modelsim* que debe utilizar para resolver todos los ejercicios. Al final del examen realice un comprimido (.zip) de esta carpeta y súbala a Moodle.

Esta hoja se ha dejado en blanco intencionadamente

Ejercicio 1			
		2 puntos	20 minutos

a) Realice un modelo VHDL de un circuito combinacional que tiene una entrada de control, *ctrl_in*, y una entrada de datos, *E*, ambas de un bit. Asimismo, tiene dos salidas, también de un bit. En una de las salidas, *ctrl_out*, se entrega el valor de la entrada de control; en la otra salida, *S*, se entrega el valor de la entrada de datos cuando la entrada de control vale 0 o el valor negado de la entrada de datos cuando la entrada de control vale 1. (0,5 puntos)
Nota: Llame al circuito diseñado ***ej1_a.vhd***. Este circuito debe estar ubicado dentro de la carpeta *EXAMEN*.

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

entity cnot is
port(ctrl_in:  in     std_logic;
     E:        in     std_logic;
     ctrl_out: buffer std_logic;
     S:        buffer std_logic);

end entity;

architecture rtl of cnot is
begin
  S <= not E when ctrl_in = '1' else
       E;

  ctrl_out <= ctrl_in;

end rtl;
```

b) El circuito combinacional del apartado anterior es una puerta CNOT. La función de esta puerta es invertible: esto quiere decir que observando la combinación de los bits en la salida del circuito puede determinarse la combinación de los bits de entrada. Es además autoinvertible, lo que quiere decir que, si se conecta a las salidas de una puerta CNOT las entradas de otra CNOT, se obtiene como salida de ésta la entrada de la primera (la segunda CNOT invierte la operación de la primera).

Teniendo esto en cuenta, realice el modelo VHDL sintetizable de un circuito combinacional que cuenta con las mismas entradas y salidas que una puerta CNOT y está dotado de una entrada adicional, *par_impar*. Cuando esta entrada vale 0, el funcionamiento del circuito debe equivaler a la conexión en cascada de un número par de puertas CNOT y cuando vale 1 a un número impar (1,5 puntos)

Nota: Llame al circuito diseñado ***ej1_b.vhd***. Este circuito debe estar ubicado dentro de la carpeta *EXAMEN*.

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

entity cascada_cnot is
port(par_impar: in     std_logic;
     ctrl_in:   in     std_logic;
     E:         in     std_logic;
     ctrl_out:  buffer std_logic;
     S:         buffer std_logic);

end entity;

architecture rtl of cascada_cnot is
begin
  S <= not E when ctrl_in = '1' and par_impar = '1' else
       E;

  ctrl_out <= ctrl_in;

end rtl;
```

Ejercicio 2			
		2,5 puntos	30 minutos

El producto de dos números complejos *a + ib* y *c + id* es *(ac-bd) + i(ad + bc)*. Teniendo en cuenta esto, realice el modelo de un circuito combinacional que tiene como entradas dos números complejos, X e Y, y entrega en su salida P el producto de ambos. Considere que los coeficientes *a*, *b*, *c* y *d* solo pueden tomar los valores -1, 0 y +1.

Cada número complejo de entrada o salida se ingresa o entrega por dos puertos, cada uno de los cuales codifica el coeficiente de su parte real o imaginaria: a X corresponden los puertos de *Re_X* e *Im_X*; a Y corresponden los puertos *Re_Y* e *Im_Y*; y a P corresponden los puertos *Re_P* e *Im_P*.

> **Nota**: Llame al circuito diseñado ***ej2.vhd***. Este circuito debe estar ubicado dentro de la carpeta *EXAMEN*.

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

entity mult_Im is
port(Re_X: in     std_logic_vector(1 downto 0);
     Im_X: in     std_logic_vector(1 downto 0);
     Re_Y: in     std_logic_vector(1 downto 0);
     Im_Y: in     std_logic_vector(1 downto 0);
     Re_P: buffer std_logic_vector(2 downto 0);
     Im_P: buffer std_logic_vector(2 downto 0));

end entity;

architecture rtl of mult_Im is
  signal ac: std_logic_vector(1 downto 0);
  signal ad: std_logic_vector(1 downto 0);
  signal bc: std_logic_vector(1 downto 0);
  signal bd: std_logic_vector(1 downto 0);

begin

-- Re_X x Re_Y
 ac <= "00" when Re_X = 0 or Re_Y = 0 else
       "01" when Re_X(1) = Re_Y(1)    else
       "11";

-- Re_X x Im_Y
 ad <= "00" when Re_X = 0 or Im_Y = 0 else
       "01" when Re_X(1) = Im_Y(1)    else
       "11";

-- Im_X x Re_Y
 bc <= "00" when Im_X = 0 or Re_Y = 0 else
       "01" when Im_X(1) = Re_Y(1)    else
       "11";

-- Im_X x Im_Y
 bd <= "00" when Im_X = 0 or Im_Y = 0 else
       "01" when Im_X(1) = Im_Y(1)    else
       "11";

 Re_P <= (ac(1)&ac) - bd;
 Im_P <= (ad(1)&ad) + bc;

end rtl;
```

Ejercicio 3			
		5,5 puntos	40 minutos

a) Observe el subcircuito 1 en la figura 1. Este subcircuito contiene un acumulador que debe totalizar la suma del producto de dos números en complemento a 2 de dos bits, *V_i* y *A_i*, durante dos periodos consecutivos de reloj, es decir:

V_i(T-1) x A_i(T-1) + V_i(T) x A_i(T))

Ambos números solo pueden tener coeficientes de valor -1, 0 y +1

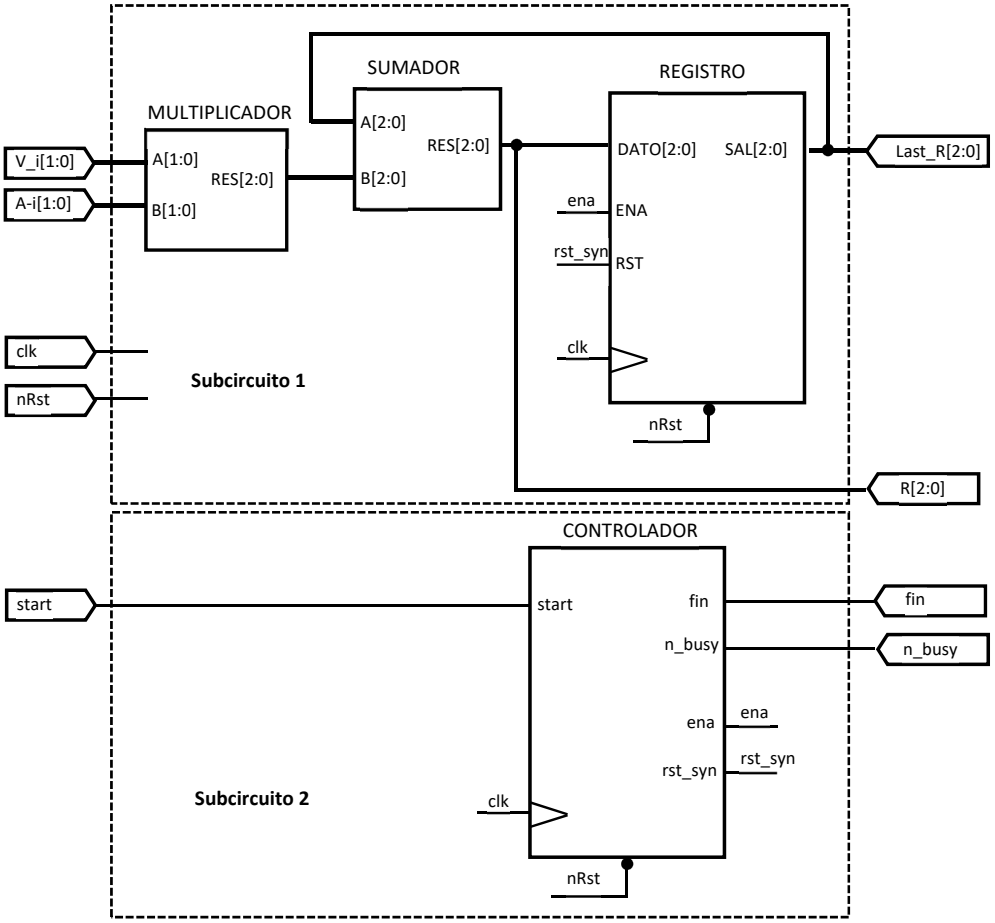

Figura 1

Realice el modelo VHDL de este subcircuito. (2,5 puntos)(20 minutos)

Nota: Para realizar el modelo de este subcircuito considere que *ena* (habilitación) y *rst_syn* (reset síncrono independiente de habilitación) son puertos de entrada

Nota2: Llame al circuito diseñado ***ej3_a.vhd***. Este circuito debe estar ubicado dentro de la carpeta *EXAMEN*.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all; -- o unsigned

entity subcircuito_1 is
port(clk:       in     std_logic;
     nRst:      in     std_logic;
     ena:       in     std_logic;
     rst_syn:   in     std_logic;
     V_i:       in     std_logic_vector(1 downto 0);
     A_i:       in     std_logic_vector(1 downto 0);
     R:         buffer std_logic_vector(2 downto 0);
     Last_R:    buffer std_logic_vector(2 downto 0));

end entity;

architecture rtl of subcircuito_1 is
begin
  -- Multiplicador (de -1, 0 o +1 por -1, 0, +1) + sumador + reset

  R <= Last_R + 0 when V_i = 0 or A_i = 0  else
       Last_R + 1 when V_i(1) = A_i(1)     else
       Last_R - 1;

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

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

      elsif ena = '1' then
        Last_R <= R;

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

b) Observe ahora el subcircuito 2 en la figura 1. Este subcircuito corresponde a un autómata que controla la operación del subcircuito 1. Funciona de la forma ilustrada en la figura 2:

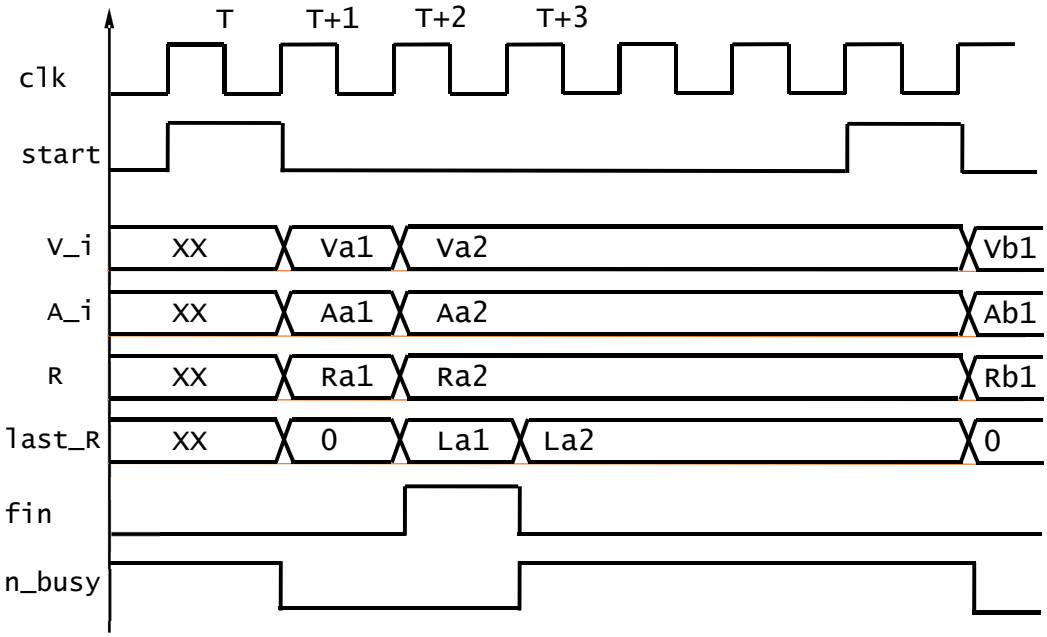

Figura 2

1. La activación de la señal *start*, que dura un ciclo de reloj, ordena la realización de la operación descrita en el enunciado del apartado anterior, es decir: la suma del producto de los dos números que hay en las entradas V_i y A_i en los dos instantes de tiempo que siguen a la activación de *start*. Es decir, si *start* se activa en el instante *T*, el circuito calcula:

   V_i(T+1) x A_i(T+1) + V_i(T+2) x A_i(T+2))

2. La salida *fin* se activa, durante un periodo de reloj, en el instante de tiempo en que está disponible el resultado en la salida R, en *T+2*

3. La salida *n_busy* se activa durante los ciclos en los que se está realizando la operación y se desactiva en el instante que sigue a la activación de *fin*, momento en el que estará disponible el resultado en la salida *last_R*. Debe permanece desactivada hasta que se indique el comienzo de una nueva operación.

4. Las salidas *ena* y *rst_syn* del autómata controlan al acumulador para que el circuito ejecute correctamente la operación.

Realice el modelo VHDL del circuito completo de la figura 1; para ello copie los procesos que modelan el circuito del apartado anterior en el nuevo cuerpo de arquitectura junto a los procesos necesarios para modelar el autómata de control. (3 puntos)(35 minutos)

Nota: Tenga en cuenta que en el modelo del circuito anterior *rst_syn* y *ena* se modelaron como entradas, mientras que en el circuito completo son señales internas.

Nota2: Llame al circuito diseñado ***ej3_b.vhd***. Este circuito debe estar ubicado dentro de la carpeta *EXAMEN*.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all; -- o unsigned

entity circuito is
port(clk:       in     std_logic;
     nRst:      in     std_logic;
     start:     in     std_logic;
     V_i:       in     std_logic_vector(1 downto 0);
     A_i:       in     std_logic_vector(1 downto 0);
     R:         buffer std_logic_vector(2 downto 0);
     Last_R:    buffer std_logic_vector(2 downto 0);
     fin:       buffer std_logic;
     n_busy:    buffer std_logic);

end entity;

architecture rtl of circuito is
  type t_estado is (wait_start, T_mas_1, T_mas_2);
  signal estado: t_estado;

  signal reg_acum: std_logic_vector(2 downto 0);
  signal ena:      std_logic;
  signal rst_syn:  std_logic;

begin
-- Automata de control
  process(clk, nRst)
  begin
    if nRst = '0' then
      estado <= reposo;

    elsif clk'event and clk = '1' then
      case estado is

        when wait_start =>
          if start = '1' then
            estado <= T_mas_1;

          end if;

        when T_mas_1 =>
          estado <= T_mas_2;

        when T_mas_2 =>
          estado <= wait_start;

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

```
  fin  <= '1' when estado = T_mas_2 else
          '0';

  n_busy <= not ena;

  rst_syn <= start when estado = wait_start else
             '0';

  ena <= '1' when estado = T_mas_1 or estado = T_mas_2 else
             '0';

-- Multiplicador (de -1, 0 o +1 por -1, 0, +1) + sumador + reset

  R <= reg_acum + 0 when V_i = 0 or A_i = 0  else
       reg_acum + 1 when V_i(1) = A_i(1)     else
       reg_acum - 1;

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

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

      elsif ena = '1' then
        reg_acum <= R;

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