Cuestión 1 (25 min, 3 ptos)

Dado el siguiente código VHDL:

```
1   entity question is
2     port (Clk:          in std_logic;
3           Reset:        in std_logic;
4           Enable:       in std_logic;
5           Up_down:      in std_logic;
6           Count:     out std_logic_vector(3 downto 0));
7   end question;
8
9   architecture a of cuestion is
10    signal sCount: std_logic_vector(3 downto 0));
11    signal a,b: std_logic;
12  begin
13    process(…)
14    begin
15      if Reset = ‘1’ then
16          sCount <= “0000”;
17          a <= ‘0’;
18      elsif clk'event and clk = ‘1’ then
19          if Enable = ‘1’ then
20            if Up_down = ‘1’ then
21              if sCount = “1001” then
22                sCount <= “0000”;
23                a <=’1’;
24              else
25                sCount <= sCount + '1';
26                a <=’0’;
27              end if;
28            else
29              if sCount = “0000” then
30                sCount = “1001”;
31                a <=’1’;
32              else
33                sCount = sCount – ‘1’;
34                a <=’0’;
35            end if;
36          end if;
37      end if;
38    end process;
39
40    Count <= sCount;
41
42    process(…)
43    begin
44      if sCount = “0111” then
45        b <=’1’;
46      else
47        b <=’0’;
48      end if;
49    end process;
50
51  end a;
```

a) Rellene las listas de sensibilidad de los procesos con las señales adecuadas.
b) Dibuje el circuito que sería sintetizado con este código.
c) Modifique el código para obtener el mismo circuito con tamaño genérico, con las siguientes especificaciones:
    1. Count es una señal de n bits.
    2. El último valor posible de cuenta es: “100…0001” (empieza y acaba con ‘1’, y en el medio todo ceros.).
    3. b se activa cuando el valor de cuenta es “0000….0011” (acaba con dos unos y el resto son ceros).

Nota: Para c) no es necesario reescribir todo el código. Escriba únicamente aquellas líneas con cambios, indicando el número de línea.

Problema 1 (1h 15 min, 7 ptos)

Se quiere diseñar un receptor de infrarrojos de acuerdo con el protocolo RC5 de Philips. El código RC5 consiste en el envío de palabras de 14 bits, véase la Figura 1. Los dos primeros bits son bits de inicio y deben valer '1'. El siguiente bit es un bit de control, cuyo valor se invierte cada vez que un nuevo código se envía desde el transmisor. Esto permite distinguir por ejemplo el caso en el que se pulsa dos veces un botón del mando a distancia, del caso en el que se ha pulsado sólo una vez. Los siguientes 5 bits de la trama se utilizan para identificar el sistema al que va dirigido la señal (televisión, video, etc.). Los últimos 6 bits identifican el comando.

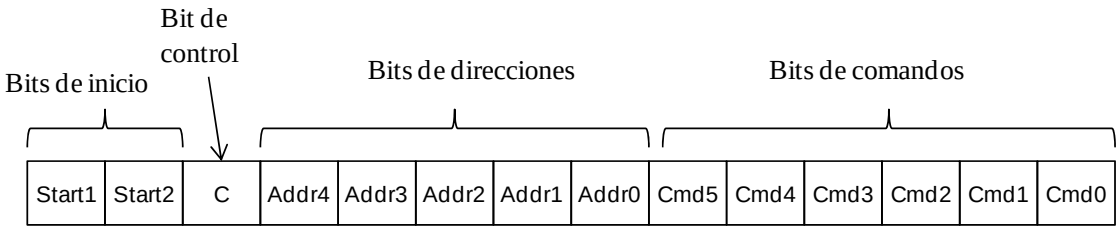

Figura 1. Formato de una trama RC5

Los bits se transmiten utilizando un código Manchester de forma que el valor de cada bit se indica con un flanco de señal, como se muestra en la Figura 2.

Figura 2. Código Manchester

Cada bit tiene una duración de 1,778 ms por lo que cada trama dura 24,892 ms. El reloj del sistema es de 50MHz. Diseñe un circuito capaz de decodificar una trama RC5. La interfaz del circuito es la siguiente:

- Entradas:
  - Clk: señal de reloj
  - Reset: señal de inicialización asíncrona activa a nivel alto.
  - Sin: señal digital de 1 bit generada por un detector de infrarrojos a partir de la señal enviada por el mando a distancia
- Salidas:
  - TV: activa a nivel alto sólo cuando el comando enviado va dirigido al sistema con dirección = 0.
  - Cmd: 6 bits. Debe mostrar el último comando pulsado una vez decodificado.
  - Error: activo a nivel alto cuando hay un fallo en la transmisión bien porque no se han recibido correctamente los dos bits de inicio o bien porque la comunicación se interrumpe (dejan de recibirse flancos antes de finalizar la trama).

Se proporciona un contador con la siguiente entidad que se puede utilizar como componente en el diseño:

```
Entity contador is
Generic( n: integer );
Port(
  Clk     : in std_logic;--señal de reloj
  Reset   : in std_logic;-- inicialización asíncrona activa a alta
  Ena     : in std_logic;--habilitación
  Clr     : in std_logic; inicialización síncrona activa a alta
  Cuenta: out integer range 0 to 2**n-1);
End contador;
```

Se pide:

a) Para muestrear cada bit se necesita un detector de flancos capaz de detectar los flancos que se producen a la mitad de la transmisión de un bit, evitando aquellos que puedan aparecer en los límites entre dos bits. Diseñe el hardware necesario para detectar flancos en la entrada *Sin*, sólo cuando una señal de enable está activada (*EnaFlanco* = 1). Dicho hardware debe generar un pulso de un ciclo de duración (*Flanco*) cada vez que se detecte un flanco (de cualquier tipo y siempre que *EnaFlanco* = 1), y una señal que indique el valor del bit muestreado (*BitLeido*) y que como se observa en la Figura 2 depende del tipo de flanco.

b) Puesto que los bits llegan en serie se utilizará un registro de desplazamiento para ir almacenando los bits de la trama. Se pide diseñar un registro de desplazamiento que tenga como entrada *BitLeido* y una señal de habilitación (*EnaReg*) para controlar cuando cargar un nuevo bit.

c) Especifique cuántos contadores considera necesarios y escriba la/s instanciación/es correspondiente/s. Indique qué función realiza/n.

d) Diseñe una máquina de estados que controle los distintos elementos del circuito y genere las salidas que sean necesarias.

e) Dibuje un diagrama de bloques a nivel RT (transferencia de registros) del circuito indicando las interconexiones necesarias.

f) Describa en VHDL el circuito total. Puede dejar indicado dónde insertar código desarrollado en los apartados anteriores con comentarios.

g) Describa en VHDL un banco de pruebas que compruebe el correcto funcionamiento del diseño. Realice comprobaciones automáticas al menos para los siguientes casos:

    a. Cuando no se reciban correctamente los dos bits de inicio envíe un mensaje indicándolo pero no pare la simulación.

    b. Cuando se detecte un error por una interrupción en la transmisión indíquelo y detenga la simulación.

Se valorará positivamente que el diseño sea lo más óptimo posible a nivel RT y que el código sea sencillo de entender.