P3. Control de un estèreo CODEC.

1. Objectius

Els objectius d'aquesta sessió són aprofundir en el disseny de circuits lògics complexos mitjançant llenguatges de descripció maquinari. El llenguatge que s'ha seleccionat és VHDL, encara que també podria realitzar-se amb Verilog.

Per a tal objectiu, es realitzarà el control d'un estèreo còdec clàssic com el PCM3003 de Texas Instruments®, generant els senyals de sincronisme correctes perquè es pugui reproduir àudio estèreo mitjançant aquest còdec.

2. Previ.

2.1. Descripció del estèreo CODEC.

El PCM3003 és un estèreo còdec amb dos convertidors analògic-digital (A/D) i dos digital-analògic (D/A) que utilitzen en tots ells moduladors Sigma-Delta amb un sobre mostreig de 64 vegades la freqüència de mostreig. L'esquema simplificat de funcionament es mostra en la figura 1.

Per a utilitzar aquest còdec, s'ha aprofitat una placa on ja venen alguns circuits lògics i un oscil·lador que faciliten l'obtenció d'alguns senyals necessaris per al correcte funcionament d'aquest. Aquesta placa se sol utilitzar habitualment en aplicacions per a processadors DSP, però s'ha adaptat perquè permeti la seva connexió a la placa SPARTAN 3E de Digilent®, que incorpora una FPGA de Xilinx de 500k portes lògiques. Aquesta placa soluciona tota la part de disseny analògic, incorporant tots els elements per a la captació correcta dels senyals analògics per part del convertidor A/D, i per a la generació del senyal per part del

D/A. Així mateix, conté la circuiteria necessària per a alimentar correctament el còdec. A més té dos micròfons, que en el nostre cas no utilitzarem.

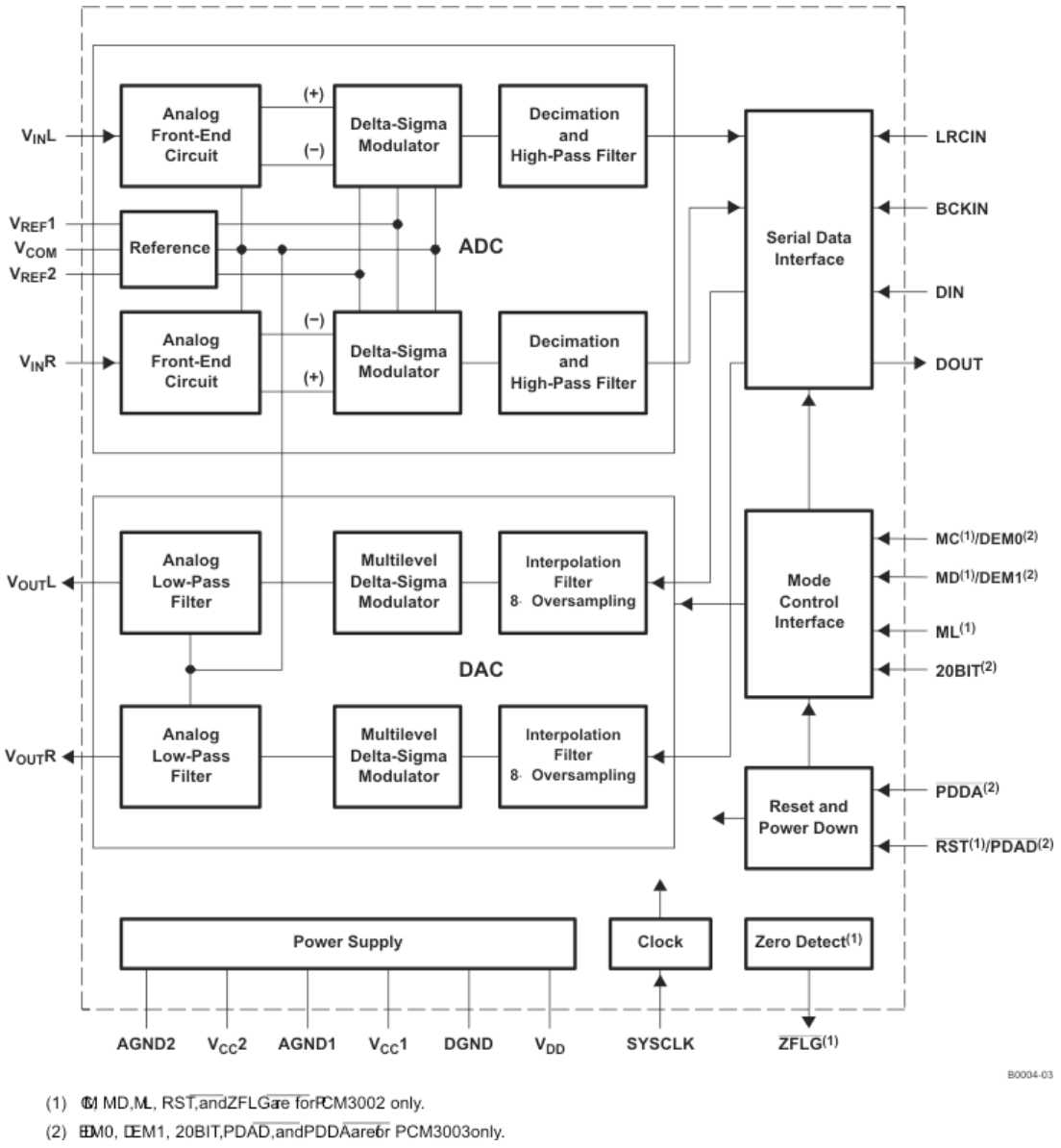

Figura 1. Diagrama de blocs del estèreo CODEC PCM.3003.

Afegeix a més tota una circuiteria lògica amb l'objectiu de generar els senyals lògics per al correcte funcionament del còdec.

Com pot veure's en la figura 1, des del punt de vista lògic els senyals d'interès són:

- SYSCLK: Aquest senyal és el rellotge del sistema. Aquesta freqüència és un múltiple de la freqüència de mostreig del senyal. El múltiple ve determinat per la freqüència present en el pin LRCIN. Per a obtenir un rellotge correcte, la placa incorpora un oscil·lador fix de 12,288 MHz, el qual és 256 vegades major de 48.000 kHz, que és una freqüència de mostreig àmpliament utilitzada en àudio.
- BCKIN: Aquest senyal determina la freqüència a la qual es transmeten els bits individuals per les línies de dades DIN i DOUT.
- LRCIN: Aquesta línia té una freqüència d'oscil·lació igual a la freqüència de mostreig del senyal analògic, i determina, segons el seu valor, que canal (esquerre o dret) es rep o transmet pels pins DIN i DOUT respectivament.
- DEM0, DEM1: Control del de-èmfasi per a corregir la distorsió freqüencial produïda pel circuit de mostreig.
- 20BIT: Determina si s'utilitzaran mostres de 16 o 20 bits.
- DIN: Dades digitals que s'envien a l'estèreo còdec perquè els converteixi a senyals analògics.
- DOUT: Dades provinents de la conversió A/D de tots dos canals, esquerre i dret.

D'entre tots aquests senyals, només les necessàries per a obtenir les dades i el seu sincronisme s'han connectat als pins de la FPGA. En la taula 1 es mostren aquests senyals amb els pins corresponents a la FPGA.

Senyal	PIN FPGA
BKCIN	A10
LRCIN	C5
DIN	B4
DOUT	A4
CLK	C9

Tabla 1. Senyals accessibles des de la FPGA i els seus terminals corresponents.

Per a controlar aquests senyals la placa incorpora la circuiteria mostrada en la figura 2, i per a donar flexibilitat d'ús a la placa, s'han incorporat uns ponts configurables, que permeten canviar entre diferents valors per als senyals enumerats anteriorment.

Els ponts d'interès en aquesta placa i les seves connexions són:

- JP4: Configura el nombre de bits de les mostres, així com l'activació dels filtres de de-èmfasis. Connectats els pins 1-2.
- JP5: Selecciona si el SYSCLK del còdec serà generat per l'oscil·lador de la placa, o directament per la FPGA. Connectats els pins 3-4.
- JP6: Permet seleccionar entre diferents formes de sincronisme de les paraules. Connectats els pins 3-4.
- JP11: Configura la freqüència de la línia BCKIN, és a dir la velocitat de transmissió dels bits. Connectats els pins 1-2.
- JP12: Determina la freqüència de mostreig, és a dir controla la freqüència del senyal LRCIN. Connectats els pins 1-2.

La resta de ponts configuren la connexió dels micròfons i les tensions d'alimentació.

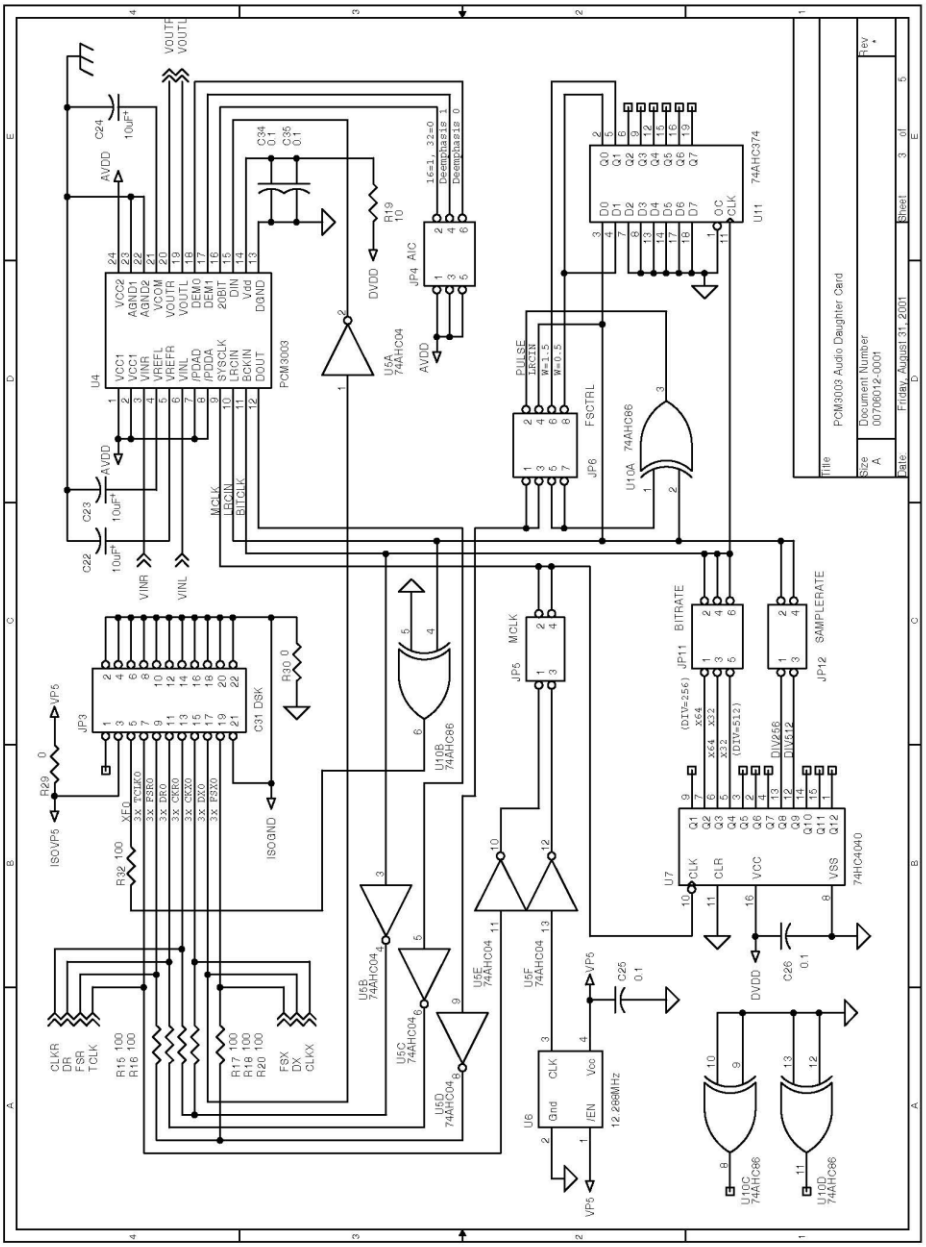

Figura 2. Circuiteria lògica que incorpora la placa del CODEC PCM3003.

2.2. Estudi del esquema presentat.

Important. No és necessari portar-lo fet per la primera sessió de laboratori.

Abans de procedir a realitzar el disseny de la interfície de l'estèreo còdec, és necessari conèixer les formes d'ona de cadascuna dels senyals i el seu sincronisme. A més, per a la realització d'un disseny eficient, es recomana realitzar est en diferents blocs, atenent la seva funcionalitat. Per a això, *VHDL permet mitjançant la descripció jeràrquica la separació del disseny en diferents blocs

2.2.1. Estudi dels senyals del CODEC.

Per a començar, segons la posició dels ponts de la placa, identificar i obtenir els següents valors:

- Número de bits utilitzats en la conversió.
- Tipus de de-èmfasi configurat.
- Font del senyal de rellotge utilitzada per al senyal CLK.
- Freqüència del senyal BCKIN.
- Freqüència del senyal LRCIN.

Una vegada identificats aquests paràmetres, i utilitzant les dades proporcionades pel fabricant:

- Identificar que format, dels disponibles, utilitza l'estèreo còdec per a enviar les dades a través de la línia **dout**, i el format en el qual llegeix les dades per a rebre'ls mitjançant la línia **din**.
- Dibuixar els senyals **lrcin**, **bckin**, **dout** i **din** durant un cicle complet de **lrcin**, parant esment a les seves temporitzacions, i especialment al moment en què les dades canvien en el senyal **dout**, respecte als flancs de **bckin**.

En el gràfic anterior, indicar els períodes de cada senyal, i en les dades, el temps actiu de cada bit.

2.2.2. Definint els blocs VHDL del disseny.

Per a realitzar un disseny eficient, és recomanable seguir un procés "top-down", de tal manera que es comença per un diagrama de blocs global del sistema, i es va desdoblegant en sub-blocs de menor grandària, i amb un major detall.

La divisió en sub-blocs que es proposa per a aquest disseny és la mostrada en la figura 3. Per a major claredat, s'han obviat alguns senyals que són comuns a la majoria de blocs, senyals com per exemple SYSCLK. De qualsevol de les maneres, aquesta és una proposta i quan es dissenya existeixen diverses solucions. S'insta a trobar altres particions d'aquest disseny.

Per a comprendre el diagrama de blocs, ha de tenir-se en compte que les dades arriben a la FPGA per una sola línia **dout** en forma d'una trama sèrie. Per a poder ser processats (per un filtre, per exemple) han de ser separats les dades del canal esquerre i les dades del canal dret.

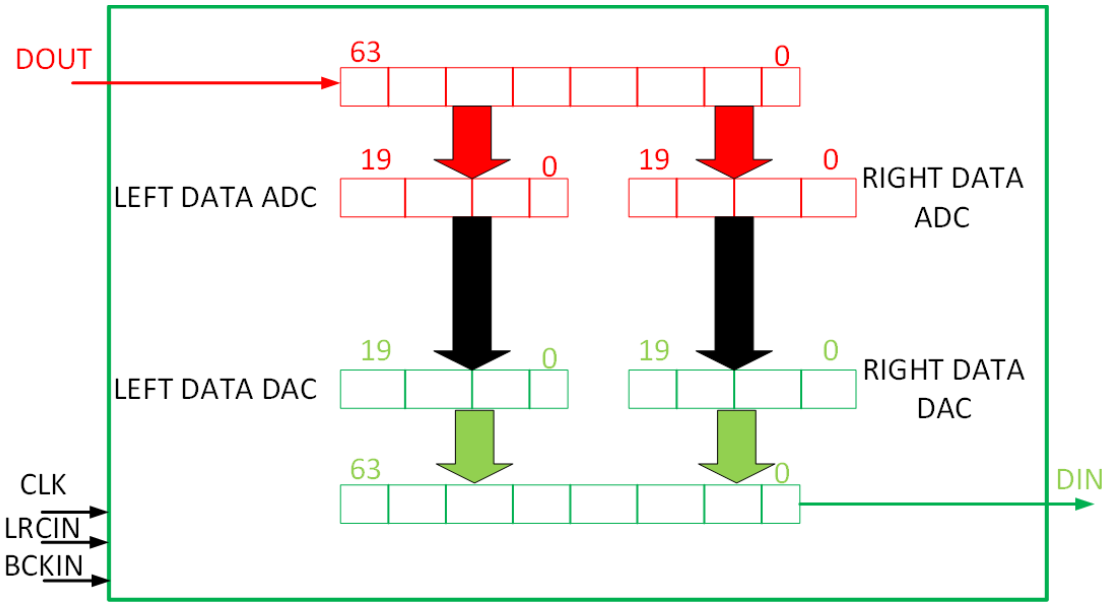

Figura 3. Proposta de la estructura de per la interfície amb el còdec estèreo.

Per això, el primer pas que s'ha de realitzar és una conversió serie-paral·lel per a tots dos canals. El bloc de conversió ha d'incorporar un senyal d'habilitació o **enable** que activi la conversió en el moment adequat que arribi la cadena de bits corresponent al canal que es vol capturar. A aquest bloc ha d'afegir-se dos registres addicionals per a mantenir el valor capturat durant tot el període del senyal.

A continuació, s'implementa la conversió d'una trama paral·lel a una trama seriï de les dades de sortida per al convertidor digital-analògic (**din**). Aquest ha de ser realitzat en un només bloc, ja que comparteix sortida.

Finalment, hi ha un bloc de control que s'encarregarà de generar tots els senyals de sincronisme dels elements del camí de dades.

La manera clàssica de capturar un flanc d'un senyal, és fer passar aquesta per un “flip-flop”, i obtenir la funció XOR del senyal amb la sortida del “flip-flop”. Aquest esquema produirà un valor “1” que servirà per a inicialitzar el comptador de bits al valor inicial en cada canvi de valor de **Ircin**.

CONSIDERAR PER ALS BLOCS SÈRIE-PARAL·LEL I PARAL·LEL-SÈRIE ELS FLANCS DE RELLOTGE QUE SIGUIN MÉS ADEQUATS PER A ASSEGURAR LA CORRECTA CAPTURA DE LA TRAMA DE BITS.

3. Desenvolupament en el laboratori.

Està pràctica va estar programada per a dues sessions de laboratori. En la primera sessió de laboratori, es realitza l'apartat 3.1, mentre que en la segona sessió, es finalitzaran els apartats 3.2, 3.3 i 3.4.

3.1. Conversor sèrie paral·lel.

El còdec envia 32 bits d'informació, tant per al canal esquerre com per al canal dret, com a resultat de la conversió ADC realitzada anteriorment pel còdec. D'aquests 32 bits d'informació que envia el còdec, només són aprofitables els 20 bits de més pes, tal com es mostra en la figura 4.

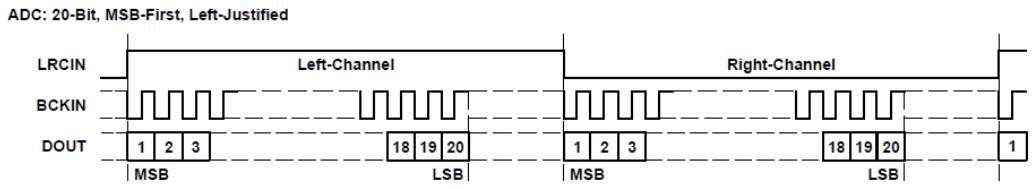

Figura 4. Trama de 1 i 0 que envia el còdec d'àudio.

L'objectiu d'aquesta primera sessió de la pràctica 3 és capturar aquests 64 bit que envia el còdec. Per dur a terme aquest propòsit, s'implementaran dos blocs digitals programats amb VHDL.

3.1.1. Conversor sèrie paral·lel. Crear entitat S2P i fer la descripció.

En convertidor sèrie paral·lel rep la trama sèrie, **dout**, que prové del còdec d'àudio.

El nom de l'entitat ha de ser "P3_S2P".

Aquest bloc digital té 4 entrades digitals:

- El senyal de rellotge, **clk**, de 50 MHz.
- El senyal de **reset**. El **reset** està actiu per nivell alt, és a dir, per un 1 lògic.
- El senyal d'habilitació, **enable_s2p**.
- El senyal que prové del còdec d'àudio, **dout**.

I té una sortida digital, **data_s2p**, un bus de 64 bits (figura 5).

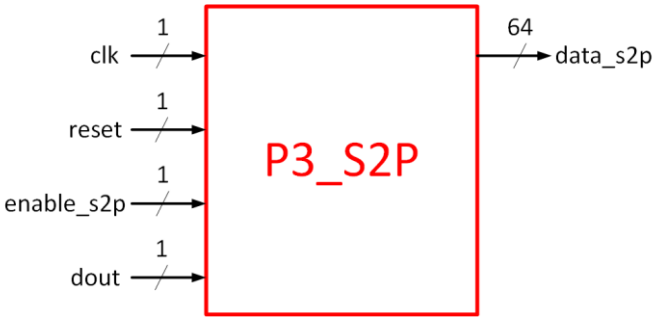

Figura 5. Bloc S2P per a llegir les dades del ADC.

El comportament ("behavior") d'aquest bloc és:

- Quan el senyal de **reset** estigui a nivell alt, la sortida data_s2p presa el valor de 0, és a dir, tots els bits a 0.
- Quan el senyal de **reset** estigui a nivell baix i hi hagi un flanc ascendent del senyal de rellotge i el senyal d'habilitació **enable_s2p** estigui a nivell alt, es desplaça el contingut de la sortida l'esquerra. En el bit de menys pes s'introdueix la dada **dout** que prové del còdec.

Un codi que pot realitzar aquest comportament és el següent:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity P3_S2P is
    Port ( clk : in  STD_LOGIC;
           dout : in  STD_LOGIC;
           enable_s2p : in  STD_LOGIC;
           reset : in STD_LOGIC;
           data_s2p: out  STD_LOGIC_VECTOR (63 downto 0));
end P3_S2P;

architecture P3_S2P_beh of P3_S2P is

       signal reg_int : std_logic_vector(63 downto 0);

begin

  process(clk, reset)
  begin

    if reset = '1' then
      reg_int <= (others => '0');
    else
              if (clk'event and clk='1') then
                      if enable_s2p = '1' then
                             reg_int <= reg_int(62 downto 0) & dout;
                      end if;
              end if;
       end if;

  end process;

  data_s2p <= reg_int;

end P3_S2P_beh;
```

Una vegada realitzat l'arxiu VHDL que descrigui el funcionament del convertidor sèrie-paral·lel, es procedeix a realitzar la simulació funcional.

3.1.2. Conversor serie paralelo. Simulación.

Carregar l'arxiu de simulació P3_S2P_tb i comprovar el correcte funcionament, és a dir, que dugui a terme la descripció anterior. Realitzar una simulació de 20 µs en total i realitzar una captura de la simulació realitzada.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY P3_S2P_tb IS
END P3_S2P_tb;

ARCHITECTURE behavior OF P3_S2P_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT P3_S2P
    PORT(
         clk : IN  std_logic;
         dout : IN  std_logic;
         enable_s2p : IN  std_logic;
         reset : IN  std_logic;
         data_s2p : OUT  std_logic_vector(63 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal clk : std_logic := '0';
   signal dout : std_logic := '0';
   signal enable_s2p : std_logic := '0';
   signal reset : std_logic := '0';

       --Outputs
   signal data_s2p : std_logic_vector(63 downto 0);

   -- Clock period definitions
   constant clk_period : time := 20 ns;

BEGIN

       -- Instantiate the Unit Under Test (UUT)
   uut: P3_S2P PORT MAP (
          clk => clk,
          dout => dout,
```

```
          ce => ce,
          reset => reset,
          data_s2p => data_s2p
        );

   -- Clock process definitions
   clk_process :process
   begin
            clk <= '0';
            wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
   end process;

       dout_process :process
   begin
            dout <= '0';
            wait for 16*clk_period;
            dout <= '1';
            wait for 16*clk_period;
   end process;

       ce_process :process
   begin
            enable_s2p <= '1';
            wait for clk_period;
            enable_s2p <= '0';
            wait for 15*clk_period;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
            reset<='1';
      wait for 100 ns;

      wait for clk_period*10;
            reset<='0';

      -- insert stimulus here

      wait;
   end process;

END;
```

3.1.3. Conversor sèrie paral·lel buffer. Crear entitat S2P_buffer i fer la descripció.

Una vegada es tenen els 64 bits rebuts, s'han de recuperar les dades d'àudio, tant del canal esquerre com del canal dret. La manera de dur-lo a terme és agafar

una part dels 64 bits rebuts. La dada del canal esquerre està entre els 20 bits de més pes de **data_s2p**, mentre que la dada del canal dret està en des del bit 31 al 12 de **data_s2p**.

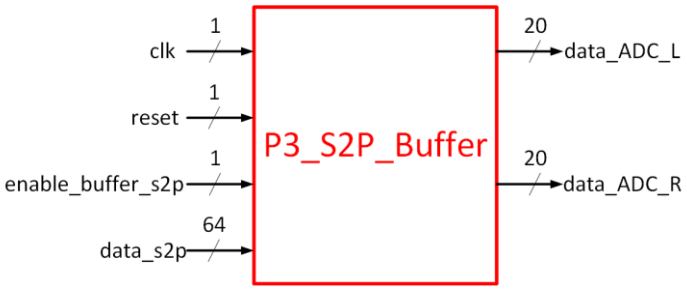

Figura 6. Bloc S2P_Buffer

El nom de l'entitat ha d'ésser "***S2P_buffer***".

Aquest bloc digital té 4 entrades digitals:

- El senyal de rellotge, **clk**, de 50 MHz.
- El senyal de **reset**. El **reset** està actiu per nivell alt, és a dir, per un 1 lògic.
- El senyal d'habilitació, **enable_buffer_s2p**.
- El senyal de 64 bits, **data_s2p**.

I té dues sortides digitals, **data_ADC_L** i **data_ADC_R**, dos bus de 20 bits (figura 6) que contindran les dades digitals del convertidor analògic-digital.

El comportament ("behavior") d'aquest bloc és:

- Quan el senyal de **reset** estigui a nivell alt, les sortides **data_ADC_L** i **data_ADC_R** prenen el valor de 0, és a dir, tots els bits a 0.
- Quan el senyal de **reset** estigui a nivell baix i hi hagi un flanc ascendent del senyal de rellotge i el senyal d'habilitació **enable_buffer_s2p** estigui a nivell alt, es carrega els 20 bits de més pes en la sortida **data_ADC_L** i els bits 31 a 12 en la sortida **data_ADC_R**.

Un codi que pot realitzar aquest comportament és el següent:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity S2P_buffer is
    Port ( data_s2p : in  STD_LOGIC_VECTOR (63 downto 0);
           enable_buffer_s2p : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           data_ADC_L : out  STD_LOGIC_VECTOR (19 downto 0);
           data_ADC_R : out STD_LOGIC_VECTOR (19 downto 0));
end S2P_buffer;

architecture S2P_buffer_Behavioral of S2P_buffer is

signal data_L_temp : STD_LOGIC_VECTOR (19 DOWNTO 0);
signal data_R_temp : STD_LOGIC_VECTOR (19 DOWNTO 0);

begin

       process(clk,reset)
       begin

              if (reset='1') then
                     data_L_temp<=(others => '0');
                     data_R_temp<=(others => '0');
              else
                     if (clk'event and clk='1') then
                            if (enable_buffer_s2p='1') then
                                   data_L_temp<=data_s2p(63 downto 44);
                                   data_R_temp<=data_s2p(31 downto 12);
                            end if;
                     end if;
              end if;
       end process;

       data_ADC_L<=data_L_temp;
       data_ADC_R<=data_R_temp;

end S2P_buffer_Behavioral;
```

Una vegada realitzat l'arxiu VHDL que descrigui el funcionament del buffer S2P_Buffer, es procedeix a realitzar la simulació funcional.

3.1.4. Conversor sèrie paral·lel buffer. Simulació.

Carregar l'arxiu de simulació P3_S2P_buffer_tb i comproveu el seu correcte funcionament, és a dir, que dugui a terme la descripció anterior. Realitzar una simulació de 30 µs en total i realitzar una captura de la simulació realitzada.

3.1.5. Unió dels blocs anteriors.

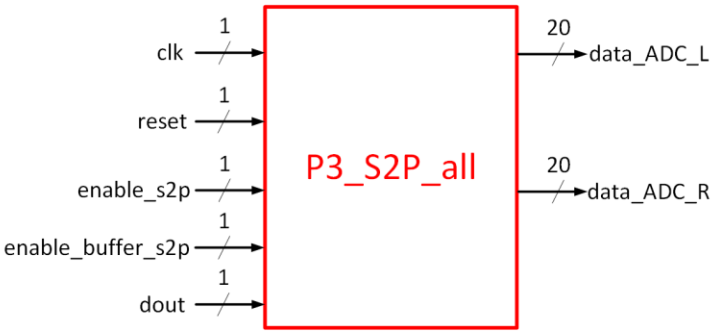

Figura 7. Unió dels blocs anteriors.

Realitzar la unió dels dos blocs anteriors, que la sortida de l'entitat P3_S2P sigui l'entrada de l'entitat P3_S2P_buffer.

El nom de l'entitat ha de ser "P3_S2P_all".

Aquest bloc digital té 4 entrades digitals:

- El senyal de rellotge, **clk**, de 50 MHz.
- El senyal de **reset**. El **reset** està actiu per nivell alt, és a dir, per un 1 lògic.

- El senyal d'habilitació, **enable_s2p**.
- El senyal d'habilitació, **enable_buffer_s2p**.
- El senyal que prové del còdec d'àudio, **dout**.

I té dues sortides digitals, **data_ADC_L** i **data_ADC_R**, dos bus de 20 bits (figura 6) que contindran les dades digitals del convertidor analògic-digital.

3.2. Conversor paral·lel sèrie.

El còdec d'àudio ha de rebre la informació en forma d'una trama seriï, però les dades que s'han d'enviar estan emmagatzemats en un registre de 20 bits. Per tant, s'ha de transformar les dades a una transmissió seriï, tal com es mostra en la figura 8.

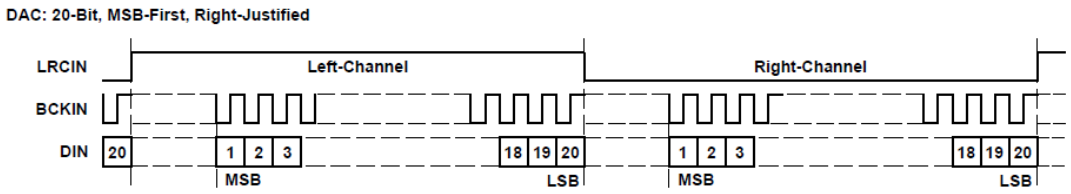

Figura 8. Transmisió sèrie de la FPGA al còdec d'àudio para enviar la paraula digital a convertir a senyal analògica.

3.2.1. Conversor paral·lel sèrie buffer. Crear entitat P2S_buffer i fer la descripció.

Amb les dades de 20 bits, **data_DAC_L** i **data_DAC_R** s'han d'ajuntar i després anar enviant des del bit més significatiu, un a un, en forma de transmissió sèrie.

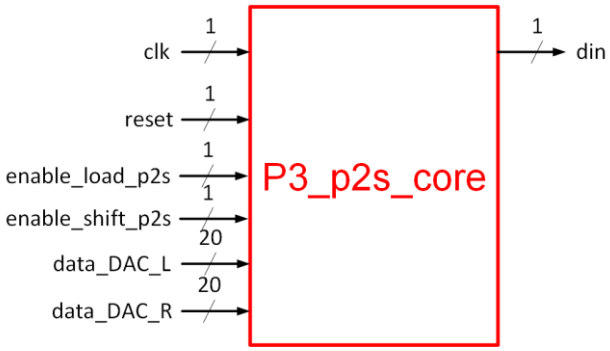

Figura 9. Bloc p2s_core

El nom de l'entitat ha de ser "P3_p2s_core".

Aquest bloc digital té 5 entrades digitals (Figura 9):

- El senyal de rellotge, **clk**, de 50 MHz.
- El senyal de **reset**. El **reset** està actiu per nivell alt, és a dir, per un 1 lògic.
- El senyal d'habilitació, **enable_load_p2s**.
- El senyal d'habilitació, **enable_shift_p2s**.
- El senyal de 20 bits, **data_DAC_L**.
- El senyal de 20 bits, **data_DAC_R**.

I té una sortida digital, **din**, que contindrà la paraula digital per a convertir a un senyal de tensió analògica.

El comportament ("*behavior") d'aquest bloc és:

- Quan el senyal de **reset** estigui a nivell alt, la sortida **din** prendrà el valor de 0.

- Quan el senyal de **reset** estigui a nivell baix i hi hagi un flanc ascendent del senyal de rellotge i el senyal d'habilitació **enable_load_p2s** estigui a nivell alt, es forma la paraula de 64 bits a enviar.
- De la mateixa manera, quan el senyal de **reset** estigui a nivell baix i hi hagi un flanc ascendent del rellotge i el senyal d'habilitació **enable_shift_p2s**, es desplaça a l'esquerra la paraula de 64 bits. En el bit menys significatiu es pot posar un 0 o un 1. És indiferent.
- El senyal **din**, sempre prendrà el valor del bit més significatiu de la paraula de 64 bits.

Del bit més significatiu al bit menys significatiu s'envia segons es representa en la figura 10. S'envien 12 bits, que poden ser 0 o 1. A continuació s'envia la paraula **data_DAC_L**. Posteriorment, 12 bits, que de nou poden ser 0 o 1 i, finalment, la paraula **data_DAC_R**.

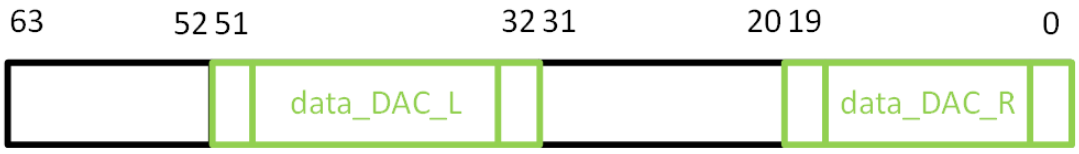

Figura 10. Trama del DAC.

Un codi que pot realitzar aquest comportament és el següent:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity P3_p2s_core is
    Port (
              clk : in STD_LOGIC;
              reset : in  STD_LOGIC;
              enable_load_p2s : in STD_LOGIC;
              enable_shift_p2s : in STD_LOGIC;
              data_DAC_L : in  STD_LOGIC_VECTOR (19 downto 0);
              data_DAC_R : in STD_LOGIC_VECTOR (19 downto 0);
              din : out  STD_LOGIC);
end P3_p2s_core;

architecture Behavioral of P3_p2s_core is

signal signalbuffer : STD_LOGIC_VECTOR (63 downto 0);

begin

process (clk,reset)

       begin

       if (reset='1') then
              signalbuffer<=(others=>'0');
       else
              if (clk'event and clk='1') then
                     if (enable_load_p2s = '1') then
                            signalbuffer<=    "XXXXXXXXXXXX"    &    data_DAC_L    &
"XXXXXXXXXXXX" & data_DAC_R;
                     elsif (enable_shift_p2s='1') then
                            signalbuffer<= signalbuffer(62 downto 0) & 'X';
                     end if;
              end if;
       end if;

end process;

din<=signalbuffer(63);

end Behavioral;
```

Una vegada realitzat l'arxiu VHDL que descrigui el funcionament del buffer P2S_core, es procedeix a realitzar la simulació funcional.

3.2.2. Conversor paral·lel sèrie buffer. Simulació.

Carregar l'arxiu de simulació P3_p2s_core_tb i comproveu el seu correcte funcionament, és a dir, que dugui a terme la descripció anterior. Realitzar una simulació de 40 µs en total i realitzar una captura de la simulació realitzada.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY P3_p2s_core_tb IS
END P3_p2s_core_tb;

ARCHITECTURE behavior OF P3_p2s_core_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT P3_p2s_core
    PORT(
         clk : IN  std_logic;
         reset : IN  std_logic;
         enable_load_p2s : IN  std_logic;
         enable_shift_p2s : IN  std_logic;
         data_DAC_L : IN  std_logic_vector(19 downto 0);
         data_DAC_R : IN  std_logic_vector(19 downto 0);
         din : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal clk : std_logic := '0';
   signal reset : std_logic := '0';
   signal enable_load_p2s : std_logic := '0';
   signal enable_shift_p2s : std_logic := '0';
   signal data_DAC_L : std_logic_vector(19 downto 0) := (others => '0');
   signal data_DAC_R : std_logic_vector(19 downto 0) := (others => '0');

       --Outputs
   signal din : std_logic;

   -- Clock period definitions
   constant clk_period : time := 20 ns;

BEGIN

       -- Instantiate the Unit Under Test (UUT)
   uut: P3_p2s_core PORT MAP (
          clk => clk,
          reset => reset,
```

```
          enable_load_p2s => enable_load_p2s,
          enable_shift_p2s => enable_shift_p2s,
          data_DAC_L => data_DAC_L,
          data_DAC_R => data_DAC_R,
          din => din
        );

   -- Clock process definitions
   clk_process :process
   begin
            clk <= '0';
            wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
   end process;

       load_p2s_process :process
   begin
            enable_load_p2s <= '1';
            wait for clk_period*1;
            enable_load_p2s <= '0';
            wait for clk_period*1042;
   end process;

       shift_p2s_process :process
   begin
            enable_shift_p2s <= '1';
            wait for clk_period;
            enable_shift_p2s <= '0';
            wait for clk_period*15;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
            reset<='1';
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for clk_period*10;
            reset<='0';

            data_DAC_L<=x"00000";
            data_DAC_R<=x"0F0F0";

            wait for 21 us;

            data_DAC_L<=x"0F0F0";
            data_DAC_R<=x"00000";
      -- insert stimulus here

      wait;
   end process;

END;
```

3.3. Unitat de control. VHDL.

Ja l'últim que queda crear és la unitat de control, que és el bloc encarregat de generar els següents senyals **enable_s2p**, **enable_buffer_s2p**, **enable_load_p2s** i **enable_shift_p2s**. Aquests senyals es generen a partir de **bckin** i **lrcin**, senyals que venen del propi còdec PCM3003.

3.3.1. Unitat de control. Crear entitat P3_uc i realitzar la descripció.

El nom de l'entitat ha de ser "P3_uc".

Aquest bloc digital té 5 entrades digitals (Figura 11):

- El senyal de rellotge, **clk**, de 50 MHz.
- El senyal de **reset**. El **reset** està actiu per nivell alt, és a dir, per un 1 lògic.
- El senyal de rellotge, **bckin**.
- El senyal de selecció de canal, **lrcin**.

I té quatre sortides digitals:

- Senyal d'habilitació **enable_s2p**. S'encarrega d'habilitar la càrrega del registre serïi paral·lel.
- Senyal d'habilitació **enable_buffer_s2p**. S'encarrega d'habilitar la càrrega del registre de 20 bits, tant del registre que contindrà la dada **data_ADC_L** com del registre **data_ADC_R**.
- Senyal d'habilitació **enable_load_p2s**, per a formar la paraula de 64 bit.
- Senyal d'habilitació **enable_shift_p2s**, per a desplaçar el senyal de 64 bits.

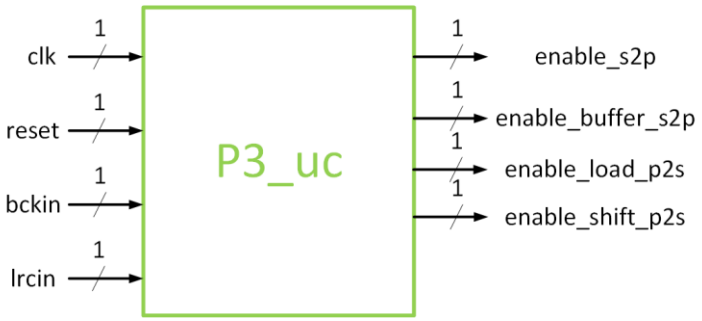

Figura 11. Unitat de control del còdec, entrades i sortides.

Aquests senyals es generaran a partir de dos blocs que s'estan penjats al campus Atenea: un detector de flanc ascendent i un detector de flanc descendent.

- El senyal **enable_s2p** es genera a partir de la detecció del flanc descendent del senyal **bckin**.
- El senyal **enable_buffer_s2p** es genera a partir de la detecció del flanc ascendent del senyal **lrcin**.
- El senyal **enable_load_p2s** es genera a partir de la detecció del flanc ascendent del senyal **lrcin**.
- El senyal de **enable_shift_p2s** es genera a partir de la detecció del flanc ascendent del senyal **bckin**.

3.4. Unió de tots els blocs.

Amb tots els blocs ja implementats, es procedeix a juntar tots els blocs.

3.4.1. Unió de tots els blocs. Creació de l'arxiu P3_codec.

S'ha de crear una entitat anomenada P3_còdec que haurà de tenir 5 entrades digitals d'1 bit i una sortida (Figura 12). Les entrades són:

- Entrada de rellotge, **clk**.
- Entrada de **reset**.
- Entrada **dout**, per on es reben les dades del conversor analògic-digital.
- Entrada **bckin**, rellotge que genera el còdec PCM3003.
- Entrada **lrcin**, senyal que indica que s'ha rebut la dada del canal esquerra o del canal dret.

I la sortida que és **din**, per on s'envien les dades al conversor digital-analògic.

Les entrades i sortides digital de la entitat P3_codec estan indicades en negreta i en vermell en la figura 13.

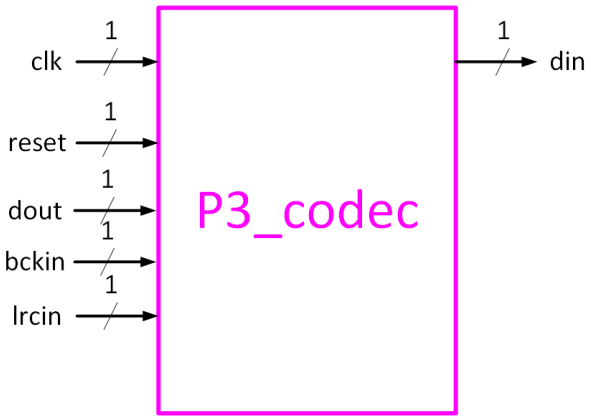

Figura 12. Entitat de la unió de tot.

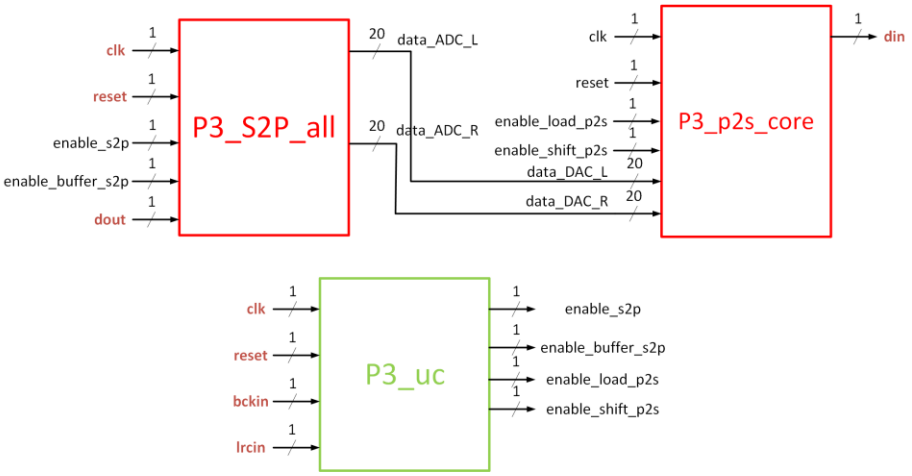

Figura 13. Unión de tots els blocs anteriors.

3.4.2. Unió de tots els blocs. Generació del arxiu .bit.

Una vegada realitzada la unió de tots els blocs, es procedeix a generar l'arxiu .bit. Prèviament, s'ha de fer realitzat l'assignació dels terminals d'entrada / sortida de la FPGA a entrades i sortides físiques de la placa de desenvolupament.

El projecte ha de finalitzar amb el disseny bolcat en la FPGA, i escoltant el senyal d'àudio estèreo en els altaveus, sense cap distorsió.

4. Conclusions

En aquesta pràctica s'ha dut a terme el disseny d'un sistema digital per a poder llegir les dades procedents del convertidor analògic – digital, tant del canal esquerre com del canal dret i s'han tornat a enviar les mateixes dades al convertidor digital – analògic.

Mitjançant el dispositiu programable FPGA es pot dur a terme aquest disseny, de la mateixa manera que es pot realitzar la implementació amb una DSP o un microcontrolador, com els de la família de Texas Instruments C2000 (anteriorment classificats per la pròpia empresa com DSP de gamma baixa). A partir d'aquesta pràctica i fins al final del curs, s'implementaran diferents sistemes digitals que permeten el processament del senyal, ja sigui aquesta de veu o de vídeo.

5. Entrega

En finalitzar cada sessió de laboratori, es lliuraran els fitxers *VHDL escrits, degudament comentats. En finalitzar l'última sessió de laboratori, es lliurarà el projecte generat per a fer la comprovació funcional en el laboratori, així com tots els arxius generats en aquest. És important que contingui l'arxiu .BIT.