

ÍNDICE

Índice	1
Descripción de la práctica.....	2
Características del FPGA	2
Características del Modulo de Audio	2
Descripción del código de programa	3
Máquina de estados	3
ROM de Notas	3
ROM de Tiempos	3
ROM de Volúmenes	3
ROM de Melodías	3
Ordinograma al nivel de transferencia de registros	4
Cronograma	5
Frecuencia de las notas musicales	6
Frecuencia del módulo de audio	7
Código de programa optimizado	8

DESCRIPCIÓN DE LA PRÁCTICA

Se pretende controlar un módulo de audio mediante un dispositivo programable del tipo FPGA. Se ha escogido un fragmento del Himno a la alegría de la Novena Sinfonía de Ludwig van Beethoven para implementar el control del módulo de audio.

Características del FPGA

El modelo de FPGA utilizado es un EPF8282A de fabricante ALTERA. Este componente es de la familia FLEX 8000, proporciona 282 registros, 208 células lógicas y 64 pines de I/O.

Características del Modulo de Audio

Los puertos codifican la siguiente informacion:

Puerto A = PA[7..0]. Frecuencia tal que:

PA[7..0] = 0xFF => 400 Hz.

PA[7..0] = 0x00 => 20000 Hz.

Entonces el incremento de un bit equivale a $\frac{20000 - 400}{2^8} = 76.6525Hz$

Puerto B = PB[7..0]. Amplitud, Timbre y Control tal que:

PB[7..4] = Amplitud:

PB[7..4] = 0xF => Máximo volumen.

PB[7..4] = 0x0 => Mínimo volumen.

PB[3,2] = Timbre:

PB[3,2] = 0b00 => Señal sinusoidal.

PB[3,2] = 0b01 => Señal Triangular.

PB[3,2] = 0b10 => Señal cuadrada.

PB[3,2] = 0b11 => Señal externa.

PB[1,0] = Control:

PB[1,0] = 0b00 => Control de frecuencia.

PB[1,0] = 0b01 => Filtro paso-bajo.

PB[1,0] = 0b10 => Filtro paso-alto.

PB[1,0] = 0b11 => Indefinido.

DESCRIPCIÓN DEL CÓDIGO DE PROGRAMA

El programa está desarrollado en VHDL con vistas a la implementación sobre un EPF8282A. Con el código de programa optimizado se han utilizado 202 de las 208 células lógicas.

Está constituido por los siguientes procesos:

Máquina de estados

Realiza el control del módulo de audio implementa con cuatro estados:

1. Actualización de la ROM Melodía, ésta a su vez provoca la actualización de las ROMs de Notas, Tiempos y Volúmenes
2. Operaciones sobre el Puerto B: El Volumen el la parte alta y el Timbre en la baja.
3. Actualización de los datos en los Puerto A y B.
4. Retardo en función del Tiempo de la nota.

ROM de Notas

Para la codificación de la melodía propuesta sólo ha sido necesario la utilización de una octava de SOL6 a SOL7 sin sostenidos, lo que equivale a 8 elementos y por lo tanto 3 bits.

ROM de Tiempos

Los tiempos necesarios han sido 4 (blanca, negra con punto, negra y corchea), por lo tanto con 2 bits

ROM de Volúmenes

En nuestro caso todas las notas tienen el mismo volumen (piano), con un bit es suficiente

ROM de Melodías

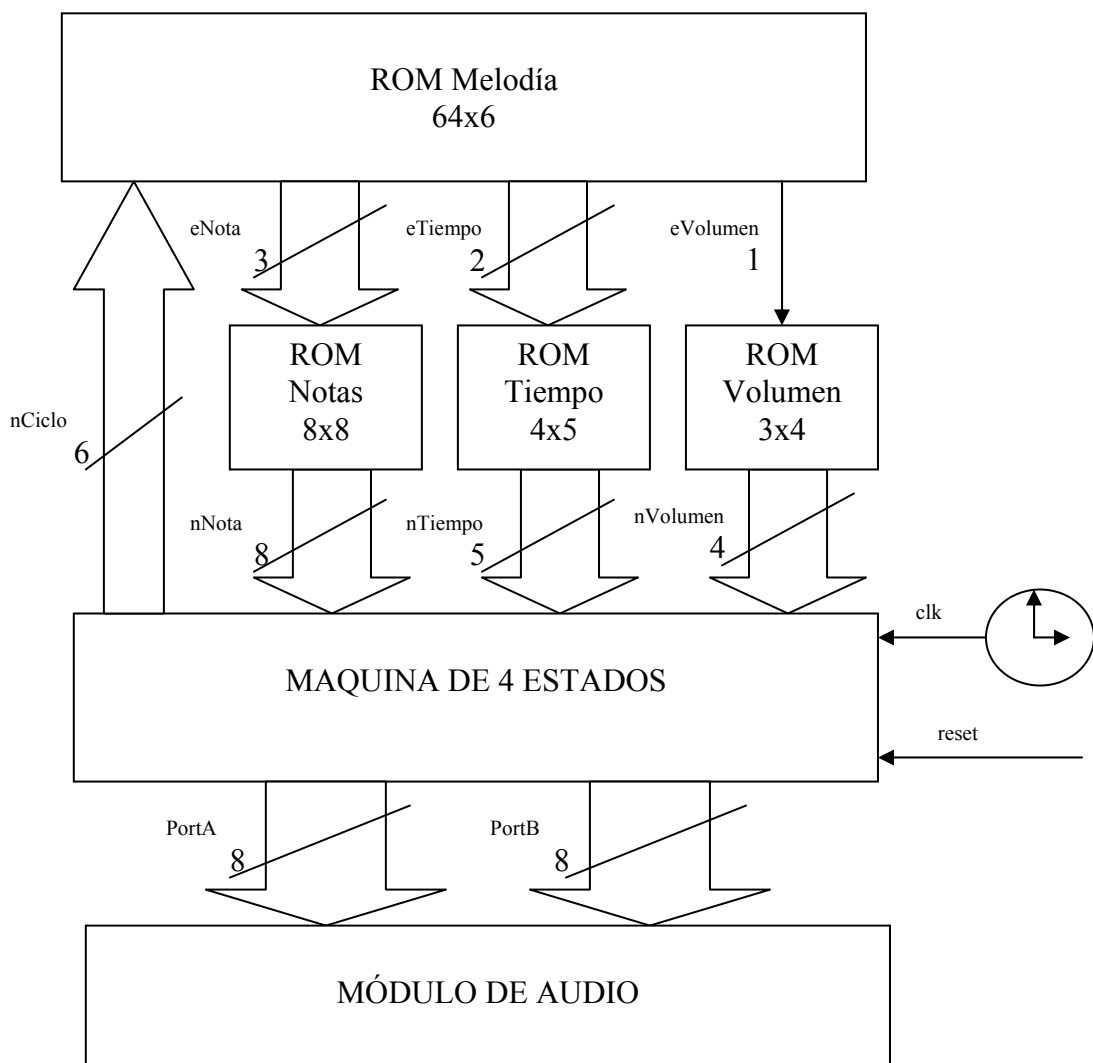
Codifica la partitura. El método para realizarlo es casi directo ya que gracias a las tres ROMs anteriores se definen los elementos característicos de la escritura musical.

Una primera codificación de las notas podría ser la de los puertos del módulo de audio, pero entonces se utilizan 16 bits para cada nota y el método de codificación es muy austero.

Con el esquema de codificación propuesto se utilizan 6 bits para cada nota y el método de codificación es mucho más cercano a la forma del pensamiento musical. Este método es tanto mejor cuanto mayor sea el tamaño de la melodía, ya que las 3 primeras ROMs provocan la utilización de una memoria mínima fija.

Datos de la melodía: se organizan en cuatro memorias ROM. Tres ROMs codifican las notas, los volúmenes y el tiempos respectivamente; y la Rom restante codifica la melodía.

Ordinograma al nivel de transferencia de registros



FRECUENCIA DE LAS NOTAS MUSICALES

	3	4	5	6	7	8	9	10
DO	130,8	261,6	523,2	1046,4	2093	4186	8372	16744
DO#	138,6	277,2	554,4	1108,8	2217,46	4434,92	8869,84	17739,68
RE	146,84	293,68	587,36	1174,72	2349,32	4698,64	9397,28	18794,56
RE#	155,56	311,12	622,24	1244,48	2489,02	4978,04	9956,08	19912,16
MI	164,8	329,6	659,2	1318,4	2637,02	5274,04	10548,08	21096,16
FA	174,6	349,2	698,4	1396,8	2793,83	5587,66	11175,32	22350,64
FA#	185	370	740	1480	2959,96	5919,92	11839,84	23679,68
SOL	196	392	784	1568	3135,96	6271,92	12543,84	25087,68
SOL#	247,64	495,28	990,56	1981,12	3962,24	7924,48	15848,96	31697,92
LA	220	440	880	1760	3520	7040	14080	28160
LA#	233,08	466,16	932,32	1864,64	3729,31	7458,62	14917,24	29834,48
SI	246,96	493,92	987,84	1975,68	3951,07	7902,14	15804,28	31608,56
	3	4	5	6	7	8	9	10

FRECUENCIA DEL MÓDULO DE AUDIO

	$400+(76,5625*AX)$	REDONDEO	MÓDULO	NOTA	ERROR [HZ]
0	400	400	255		
1	476,5625	480	254		
2	553,125	550	253		
3	629,6875	630	252		
4	706,25	710	251		
5	782,8125	780	250		
6	859,375	860	249		
7	935,9375	940	248		
8	1012,5	1010	247		
9	1089,0625	1090	246		
10	1165,625	1170	245		
11	1242,1875	1240	244		
12	1318,75	1320	243		
13	1395,3125	1400	242		
14	1471,875	1470	241	SOL6	18
15	1548,4375	1550	240		
16	1625	1630	239		
17	1701,5625	1700	238		
18	1778,125	1780	237	LA6	-20
19	1854,6875	1850	236		
20	1931,25	1930	235		
21	2007,8125	2010	234	SI6	-35
22	2084,375	2080	233	DO7	13
23	2160,9375	2160	232		
24	2237,5	2240	231		
25	2314,0625	2310	230		
26	2390,625	2390	229	RE7	-4
27	2467,1875	2470	228		
28	2543,75	2540	227		
29	2620,3125	2620	226	MI7	17
30	2696,875	2700	225		
31	2773,4375	2770	224	FA7	23
32	2850	2850	223		
33	2926,5625	2930	222		
34	3003,125	3000	221		
35	3079,6875	3080	220		
36	3156,25	3160	219	SOL7	-25
37	3232,8125	3230	218		
38	3309,375	3310	217		

CÓDIGO DE PROGRAMA OPTIMIZADO

```

--
--          SISTEMAS ELECTRONICOS DIGITALES
--
-- Archivo: ModuloV5.vhd
-- Control del módulo de audio
-- Se programa en una FPGA FLEX8000 de Altera
-- Musica: HIMNO A LA ALEGRIA
-- Fragmento de la Novena Sinfonia
-- Ludwig van Beethoven

library ieee;
use ieee.std_logic_1164.all;

entity ModuloV5 is
    port(
        rst      : in    bit;          -- Reset: 1 bit
        clk      : in    bit;          -- Reloj: 1 bit
        PortA    : out   integer range 0 to 255; -- Puerto A: 8 bits
        PortB    : out   integer range 0 to 255; -- Puerto B: 8 bits
    end;

architecture Musica of ModuloV5 is
    -- Las puertos codifican la siguiente informacion:
    -- PortA = PA[7..0]. Frecuencia tal que
    -- PA[7..0] = 0xFF => 400 Hz
    -- PA[7..0] = 0x00 => 20000 Hz
    -- PortB = PB[7..0]. Amplitud, Timbre y Control tal que
    -- PB[7..4] = Amplitud
    -- PB[7..4] = 0xF => Max volumen
    -- PB[7..4] = 0x0 => Min volumen
    -- PB[3,2] = Timbre
    -- PB[3,2] = 0b00 => Señal sinusoidal
    -- PB[3,2] = 0b01 => Señal Triangular
    -- PB[3,2] = 0b10 => Señal cuadrada
    -- PB[3,2] = 0b11 => Señal externa
    -- PB[1,0] = Control
    -- PB[1,0] = 0b00 => Control de frecuencia
    -- PB[1,0] = 0b01 => Filtro paso-bajo
    -- PB[1,0] = 0b10 => Filtro paso-alto
    -- PB[1,0] = 0b11 => Indefinido
    constant CICLOS : integer range 0 to 63 := 63; -- Numero de CICLOS (Notas)
    signal nCiclo : integer range 0 to 63; -- Ciclo actual
    signal nNota : integer range 0 to 255; -- Frecuencia: 8 bits
    signal nTiempo : integer range 0 to 31; -- Duracion: 5 bits
    signal nVolumen : integer range 0 to 15; -- Volumen: 4 bits
    signal nTimbre : integer range 0 to 15; -- Timbre 4 bits
    type Notas is (SOL1,LAL,SI1,DO2,RE2,MI2,FA2,SOL2); -- Algunas notas 3 bits
    signal eNota : Notas;
    type Tiempos is (blanca,negraP,negra,corchea); -- Algunos tiempos 2 bits
    signal eTiempo : Tiempos;
    type Volumen is (piano); -- Un Volumen 1 bit
    signal eVolumen : Volumen;

begin
    Estado : process (clk, rst)
        variable nEst : integer range 0 to 3; -- Estado
        variable nCnt : integer range 0 to 31; -- Ciclos de una nota
        variable nPB : integer range 0 to 255; -- Operaciones del puerto B
    begin
        if (rst = '0') then
            nCiclo <= CICLOS;
            PortA <= 0;
            PortB <= 0;
            nCnt := 0;
            nEst := 0;
            nPB := 0;
            nTimbre <= 8; -- Inicializacion del timbre deseado
        else
            if (clk'EVENT and clk = '1') then
                case nEst is
                    when 0 => -- Actualizacion ROM_Melodia =>
                        -- actualizacion de eTiempo, eNota, eVolumen y
                        -- sus correspondientes ROMs =>
                        -- actualizacion de nTiempo, nNota, nVolumen
                        if (nCiclo = CICLOS) then
                            nCiclo <= 0;
                        else
                            nCiclo <= nCiclo + 1;
                        end if;
                        nEst := 1;
                end case;
            end if;
        end process;
    end;
end;

```

```

when 1 =>      -- Preparo el puerto B
              nPB := nVolumen * 16;
              nPB := nPB + nTimbre ;
              nEst := 2;
when 2 =>      -- Carga datos en puertos
              PortA <= nNota;-- Frecuencia
              PortB <= nPB; -- Volumen y Timbre
              nCnt := nTiempo;
              nEst := 3;
when 3 =>      -- Duracion de la nota
              if (nCnt = 0) then
                  nEst := 0;
              else
                  nCnt := nCnt - 1;
              end if;
            end case;
          end if;
        end if;
      end process;

ROM_Notas : process(eNota, rst)
begin
  if (rst = '0') then
    nNota <= 0;
  else
    case eNota is -- OCTAVAS 6 Y 7
      when SOL1 => nNota <= 240;
      when LA1 => nNota <= 237;
      when SI1 => nNota <= 235;
      when DO2 => nNota <= 233;
      when RE2 => nNota <= 229;
      when MI2 => nNota <= 226;
      when FA2 => nNota <= 224;
      when SOL2 => nNota <= 219;
    end case;
  end if;
end process;

ROM_Tiempos : process(eTiempo, rst)
begin
  if (rst = '0') then
    nTiempo <= 0;
  else
    case eTiempo is
      when blanca => nTiempo <= 20;
      when negraP => nTiempo <= 15;
      when negra => nTiempo <= 10;
      when corchea => nTiempo <= 5;
    end case;
  end if;
end process;

ROM_Volumenes : process(eVolumen, rst)
begin
  if (rst = '0') then
    nVolumen <= 0;
  else
    nVolumen <= 8;
  end if;
end process;

ROM_Melodia : process(nCiclo, rst)
begin
  if (rst = '0') then
    eNota <= MI2;
    eTiempo <= blanca;
    eVolumen <= piano;
  else
    case nCiclo is -- Allegretto
      -- 1
      when 0 => eNota <= MI2; eTiempo <= blanca; eVolumen <= piano;
      when 1 => eNota <= FA2; eTiempo <= negra; eVolumen <= piano;
      when 2 => eNota <= SOL2; eTiempo <= negra; eVolumen <= piano;
      -- 2
      when 3 => eNota <= SOL2; eTiempo <= negra; eVolumen <= piano;
      when 4 => eNota <= FA2; eTiempo <= negra; eVolumen <= piano;
      when 5 => eNota <= MI2; eTiempo <= negra; eVolumen <= piano;
      when 6 => eNota <= RE2; eTiempo <= negra; eVolumen <= piano;
      -- 3
      when 7 => eNota <= DO2; eTiempo <= negra; eVolumen <= piano;
      when 8 => eNota <= DO2; eTiempo <= negra; eVolumen <= piano;
      when 9 => eNota <= RE2; eTiempo <= negra; eVolumen <= piano;
      when 10 => eNota <= MI2; eTiempo <= negra; eVolumen <= piano;
      -- 4
      when 11 => eNota <= MI2; eTiempo <= negraP; eVolumen <= piano;
      when 12 => eNota <= RE2; eTiempo <= corchea; eVolumen <= piano;
      when 13 => eNota <= RE2; eTiempo <= blanca; eVolumen <= piano;
    end case;
  end if;
end process;

```

```

-- 5
when 14 =>      eNota <= MI2;      eTiempo <= blanca;      eVolumen <= piano;
when 15 =>      eNota <= FA2;      eTiempo <= negra;      eVolumen <= piano;
when 16 =>      eNota <= SOL2;     eTiempo <= negra;      eVolumen <= piano;
-- 6
when 17 =>      eNota <= SOL2;     eTiempo <= negra;      eVolumen <= piano;
when 18 =>      eNota <= FA2;      eTiempo <= negra;      eVolumen <= piano;
when 19 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 20 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
-- 7
when 21 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
when 22 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
when 23 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
when 24 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
-- 8
when 25 =>      eNota <= RE2;      eTiempo <= negraP;     eVolumen <= piano;
when 26 =>      eNota <= DO2;      eTiempo <= corchea;    eVolumen <= piano;
when 27 =>      eNota <= DO2;      eTiempo <= blanca;     eVolumen <= piano;
-- 9
when 28 =>      eNota <= RE2;      eTiempo <= blanca;     eVolumen <= piano;
when 29 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 30 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
-- 10
when 31 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
when 32 =>      eNota <= MI2;      eTiempo <= corchea;    eVolumen <= piano;
when 33 =>      eNota <= FA2;      eTiempo <= corchea;    eVolumen <= piano;
when 34 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 35 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
-- 11
when 36 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
when 37 =>      eNota <= MI2;      eTiempo <= corchea;    eVolumen <= piano;
when 38 =>      eNota <= FA2;      eTiempo <= corchea;    eVolumen <= piano;
when 39 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 40 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
-- 12
when 41 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
when 42 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
when 43 =>      eNota <= SOL1;     eTiempo <= negra;      eVolumen <= piano;
when 44 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
-- 13
when 45 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 46 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 47 =>      eNota <= FA2;      eTiempo <= negra;      eVolumen <= piano;
when 48 =>      eNota <= SOL2;     eTiempo <= negra;      eVolumen <= piano;
-- 14
when 49 =>      eNota <= SOL2;     eTiempo <= negra;      eVolumen <= piano;
when 50 =>      eNota <= FA2;      eTiempo <= negra;      eVolumen <= piano;
when 51 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 52 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
-- 15
when 53 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
when 54 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
when 55 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
when 56 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
-- 16
when 57 =>      eNota <= RE2;      eTiempo <= negraP;     eVolumen <= piano;
when 58 =>      eNota <= DO2;      eTiempo <= corchea;    eVolumen <= piano;
when 59 =>      eNota <= DO2;      eTiempo <= blanca;     eVolumen <= piano;
-- 17
when 60 =>      eNota <= RE2;      eTiempo <= blanca;     eVolumen <= piano;
when 61 =>      eNota <= MI2;      eTiempo <= negra;      eVolumen <= piano;
when 62 =>      eNota <= DO2;      eTiempo <= negra;      eVolumen <= piano;
-- 18
when 63 =>      eNota <= RE2;      eTiempo <= negra;      eVolumen <= piano;
end case;
end if;
end process;

end Musica;

```