AS-BUILT

Digitado por:	Professor:	Versão	Ano
Ricardo Kim 201100597	Renato Giacomini	1.0	2017

ÍNDICE

1. Introdução

O projeto se trata do desenvolvimento de hardware com a função de gerar eco e reverberação, através da FPGA da Altera modelo DE1, Cyclone II modelo EP2C20F484C7N.

1.1. Tema

Eco e Reverberação são definições de percepção de como os ouvidos percebem as informações sonoras.

- Eco: essa percepção ocorre quando o som é recebido com um intervalo no mínimo de 100 milissegundos.
- Reverberação: é a deflexão da onda em vários obstáculos, sobrepondo a informação sonora.

1.2. Objetivo

O projeto tem como objetivo realizar a reprodução de ondas sonoras através dos efeitos eco e reverberação. São usados LEDs verdes para representar a captação do áudio, LEDs vermelhos para representar a reprodução do áudio a ser somado com o áudio do momento, e também a utilização de 4 displays de 7 segmentos, sendo 2 indicativo das posições de armazenamento, e 2 indicativos das instruções da interface bluetooth.

1.3. Delimitação do Problema

Verificar a distancia do dispositivo de captação do som com o emissor, pois uma distância insuficiente pode acarretar em reentrada do dispositivo causando microfonia, o dispositivo é portátil, respeitando a fonte de alimentação de 7,5 volts para a FPGA, e uma alimentação dedicada para o dispositivo de amplificação de som, sendo que a distância de alcance fica dependente da caixa amplificadora, e para controle necessário o dispositivo smartphone android 3.0 ou superior.

1.4. Motivação acadêmica

Analisando a importância de unir o conhecimento teórico com pratica, optou-se em desenvolver um projeto que melhor simulasse o uso de todos os conhecimentos adquiridos do uso da ferramenta com as práticas de mercado.

Desta maneira estabelecendo a capacidade do aluno cursando a disciplina de Projetos Eletrônicos Digitais através dos respectivos critérios acadêmicos.

O trabalho possui grande importância no mercado, pois disponibiliza uma ferramenta que possa auxiliar profissionais de outras áreas a atingirem seus objetivos.

1.5. Método de trabalho

O método usado para esse trabalho foi utilizada programação VHDL, e Verilog para a construção dos ligamentos do hardware da FPGA através da IDE Quartus na versão 13.0, foi utilizada a linguagem de programação orientada a objetos C# com o framework de desenvolvimento de plataformas mobile Xamarin pela IDE Visual Studio 2017 para o desenvolvimento da interface de controle do aplicativo para a comunicação sem fio pelo módulo Bluetooth de baixo consumo, e também foi utilizada a linguagem de programação estruturada C, para definir as configurações do módulo Bluetooth através da comunicação serial da IDE Arduino, assim podendo definir nome do dispositivo e senha.

O projeto está dividido no desenvolvimento físico do Hardware tanto do circuito, e a integração do dispositivo de entrada e saída, integração do dispositivo de comunicação sem fio bluetooth HC-05, e a interface de controle por smartphone dispositivo Android.

1.6. Organização do Trabalho

Esse documento está dividido em introdução, descrição, e requisitos de sistema.

1.7. Glossário

- FPGA – Field Programmable Gate Array
- VHDL – VHSIC Hardware Description Language
- Smartphone – Dispositivo portátil com um Sistema operacional nativo.
- Android – Sistema operacional do Smartphone usado para desenvolvimento
- Eco – Uma reflexão de som que chega ao ouvinte pouco tempo depois do som direto.
- Reverberação – Similar ao eco, porém se trata de mais reflexões chegarem ao ouvinte que ele não possa distinguir uma das outras.
- C# – Linguagem de programação proprietária da Microsoft.
- C – Linguagem de programação estruturada criada por Dennis Ritchie por volta de 1973.
- Xamarin – Empresa independente que foi comprada pela Microsoft e hoje disponibiliza seus framworks junto com a IDE Visual Studio.
- IDE – Ambientes de desenvolvimento integrado (Integrated Development Enviroment).
- Arduino – IDE utilizada para desenvolvimento de instruções de micro controladores.

2. Descrição

O sistema busca a integração de diferentes plataformas complementares, sendo assim o sistema esta particionado a placa FPGA como o Hardware de controle, a interface de configuração como o Software, e o Microfone, as Caixas de som e o módulo Bluetooth como os periféricos do sistema.

A FPGA é a placa de desenvolvimento da Altera, onde nela será armazenado o áudio em formato digital na memória SRAM integrada na placa.

O Software foi desenvolvido para o envio das instruções operacionais para o hardware assim personalizando os efeitos desejados.

Os periféricos são os dispositivos que permitem a captação, a reprodução e a comunicação do Hardware e Software.

2.1. Descrição do problema

A voz humana é um instrumento poderoso que todos tocamos, é um que pode iniciar uma guerra ou fidelizar acordos, e mesmo assim podemos passar uma vida inteira sem ser ouvidos, e como podemos falar poderosamente para sermos ouvidos.

O sistema possibilita o usuário a se manifestar de forma diferenciada, entre outras aplicações também voltadas para o vocalista em shows de músicas proporcionando efeitos diferenciados.

O desenvolvedor não se responsabiliza por mau uso e ou, imperícia no manejo do equipamento, todos os efeitos produzidos estão delimitados pelo hardware.

O limite de armazenamento de dados esta relacionado ao limite da memória SRAM vinculado à placa, e também ao tempo necessário para não desfigurar o efeito sonoro.

A comunicação do módulo sem fio Bluetooth está configurada de forma padrão com baud rate de 9600bits.

2.2. Principais envolvidos e suas características

2.2.1. Os usuários do sistema

Esse projeto tem como foco usuários como banda de música, palestrantes, ou oradores, o eco é uma característica que fixa a atenção e até enfatiza, podendo ser utilizado para criar efeitos em discursos.

2.2.2. Desenvolvedores do sistema

O desenvolvedor desse sistema é um estudante de engenharia de computação da Universidade São Judas que ficou sem grupo, portanto é o único desenvolvedor do projeto.

Cabendo a ele a responsabilidade da criação da documentação, do Sistema, do software usado na plataforma Android.

Exceto pelo controle do sistema de conversão analógico-digital, codec, e conversão digital-analógico que foi provido pelo professor responsável pela proposta do projeto.

2.3. Regras de negócio

O eco e a reverberação são excelentes meios de conseguir atenção, para eventos, shows, manifestações, e discursos.

O disposto é modular permitindo a utilização de um microfone ou uma caixa amplificadora de preferencia.

O utilizador necessitará de um dispositivo Android versão 3.0 ou superior com comunicação Bluetooth podendo ser o modelo HC-05 ou HC 06 para a interface de controle e permitindo assim a personalização tanto o tempo do efeito e o tipo do efeito.

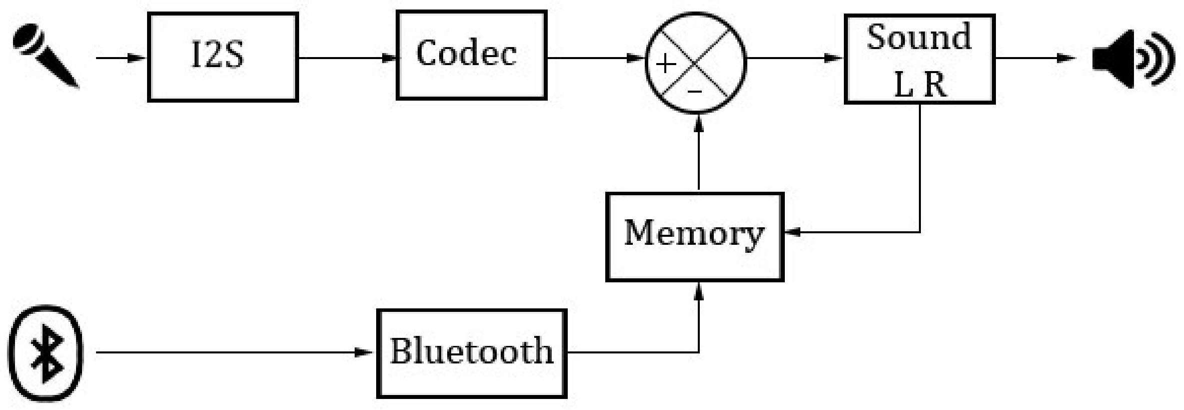
Vide responsabilidade de o utilizador respeitar as tensões de alimentação especificadas nesse documento.

3. Requisitos de sistema

3.1. Diagrama de blocos

3.1.1. Sistema de Controle

A figura abaixo, mostra o sistema de controle da placa FPGA



3.2. Requisitos Não-Funcionais

O Eco é uma forma de onda onde o alvo recebe uma informação repetidamente igual, com o mínimo de 100 milissegundos e o comprimento máximo sem descaracterizar no sistema foi definido em 743 milissegundos.

$$\frac{2^{16}}{88{,}201 \cdot 10^{3}} \rightarrow 743 \cdot 10^{-3}\ [s]$$

A reverberação ocorre quando os alvos recebem a mesma informação de forma sobreposta o efeito é mais bem percebido com 185 milissegundos.

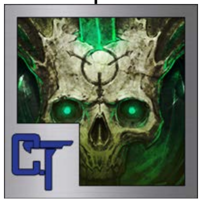
$$\frac{2^{14}}{88{,}201 \cdot 10^{3}} \rightarrow 185 \cdot 10^{-3} [s]$$

3.3. Protótipo

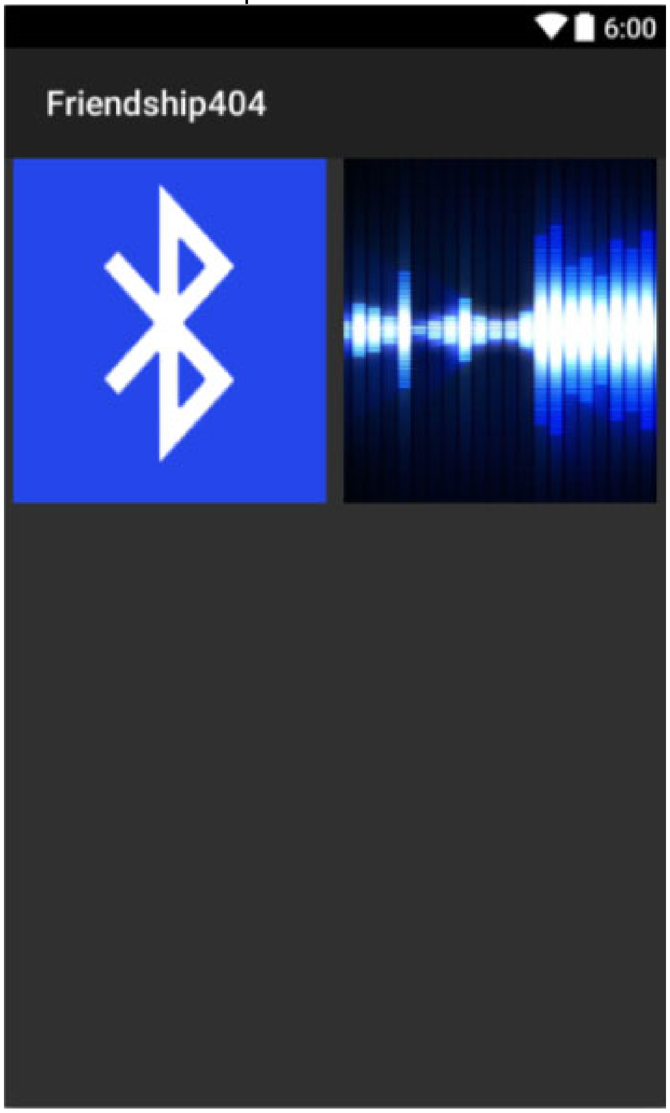
3.3.1. Diagrama de Navegação

3.3.1.1. APP

A figura abaixo representa o ícone do aplicativo que faz o controle da placa.



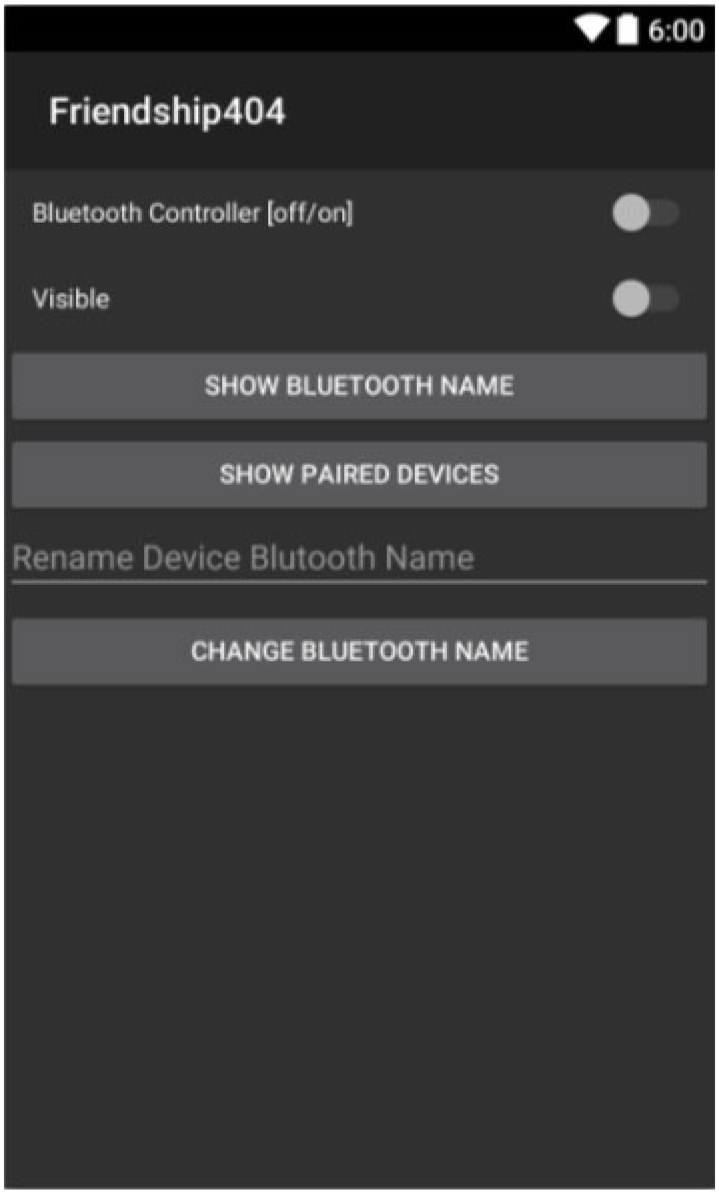
A figura abaixo representa a interface do aplicativo



Nesta figura é possível tomar 2 direções, para o controle do Bluetooth do dispositivo e o controlador da FPGA.

A figura abaixo representa o controlador Bluetooth:

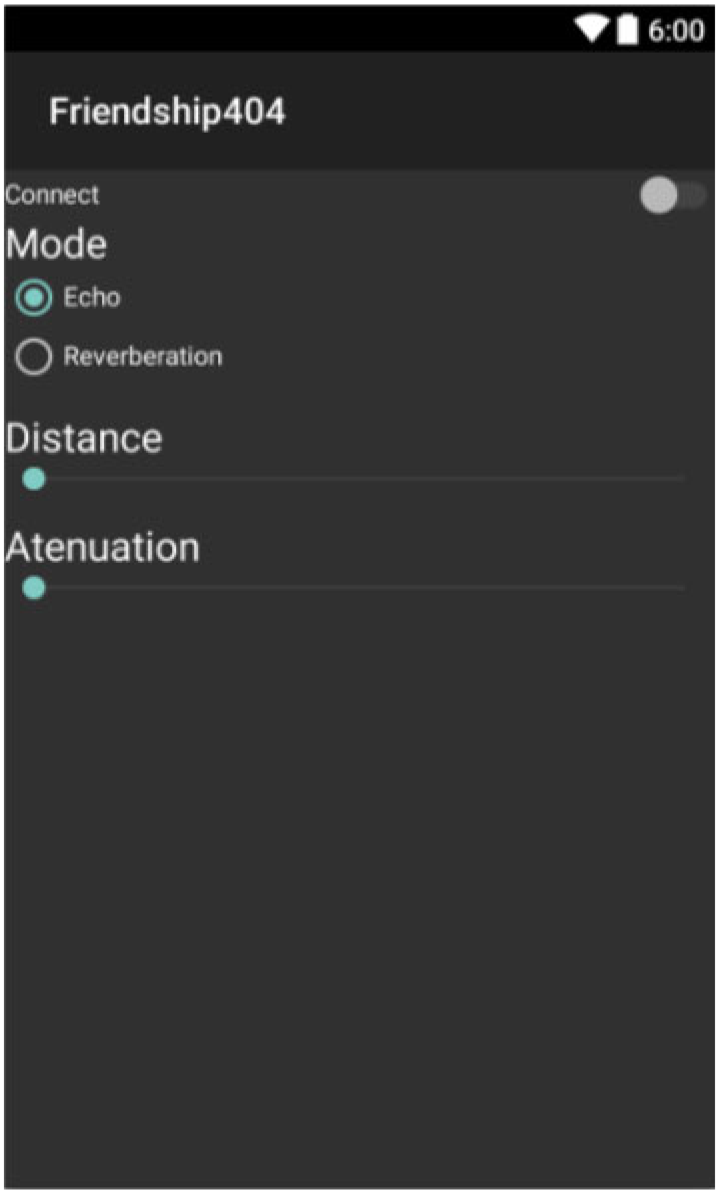
1. Disponibiliza o controle de ligar e desligar a função Bluetooth
2. Deixa o dispositivo visível (aplicável caso o modulo Bluetooth estiver em modo máster)
3. Mostra a ID do Bluetooth no dispositivo Android.
4. Mostra uma lista com os dispositivos pareados.
5. Campo que permite renomear a ID (funciona somente se o Bluetooth estiver ligado).
6. Salva o novo nome do ID do dispositivo



A figura abaixo representa a interface de controle:

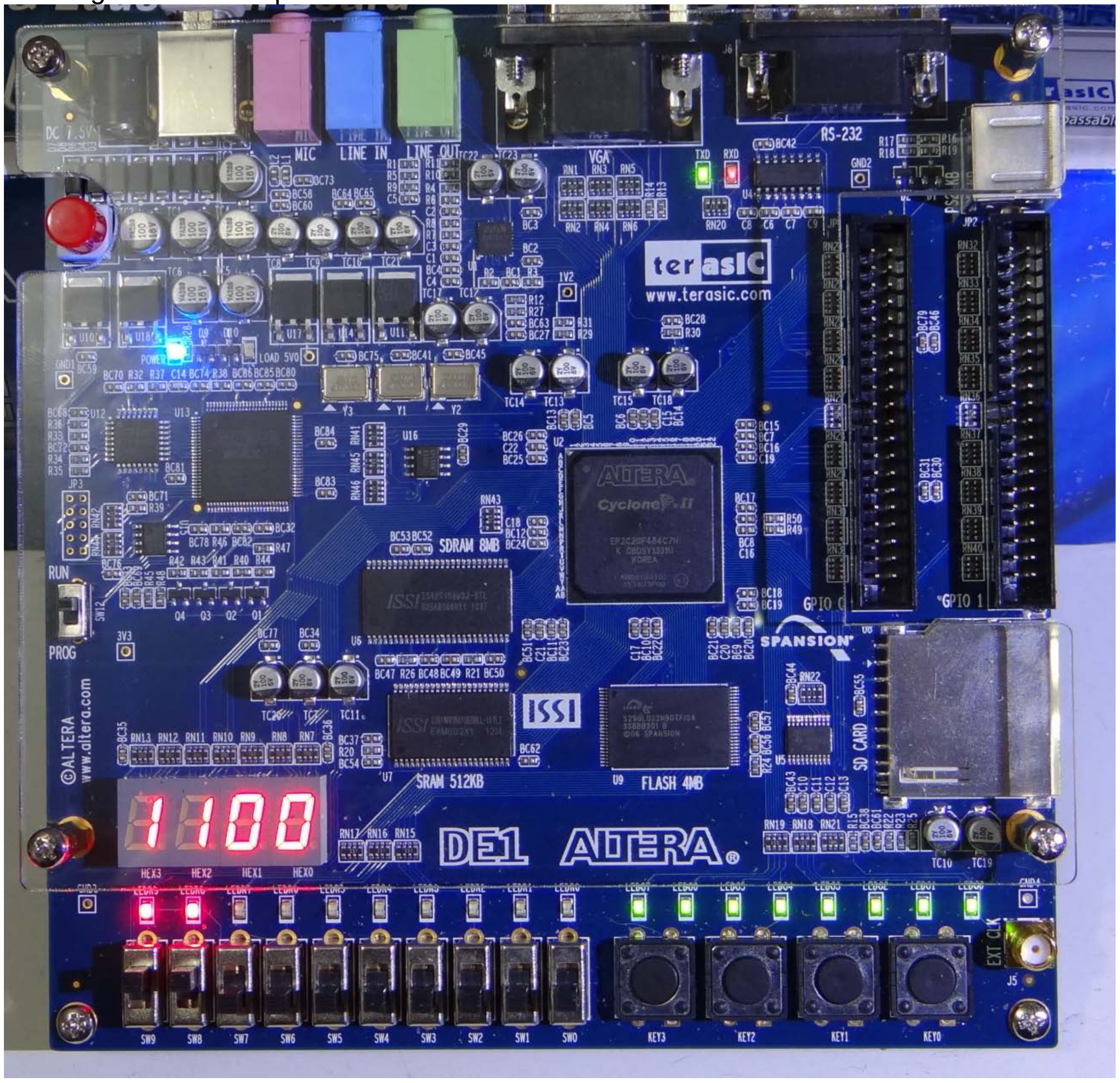
1. Faz a conexão com o módulo
2. Define o modo de uso se Eco ou Reverberação
3. Distancia: é correspondente a distancia que o som irá percorrer até retornar
4. Atenuação: o quando o áudio será reduzido a cada repetição

Caso o modo seja definido como Reverberação os controles da distancia e atenuação fica desabilitada.

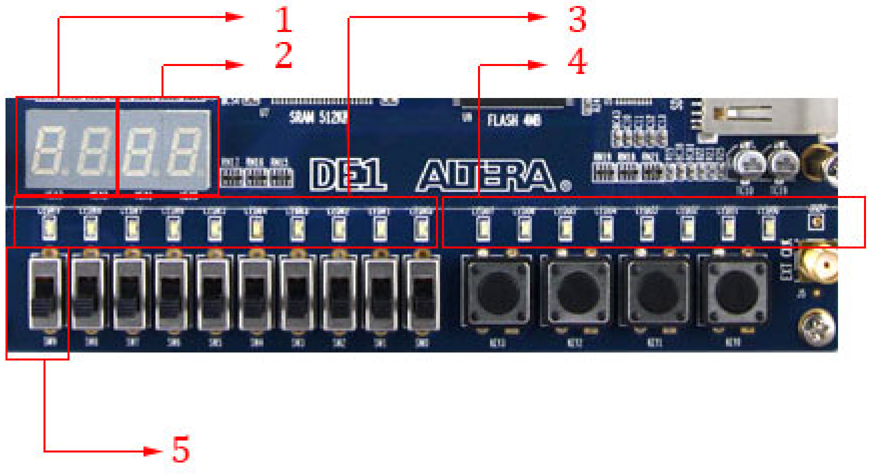


3.3.1.2. Interface da FPGA

Essa figura mostra a placa de desenvolvimento



A figura abaixo representa a interface de informações da FPGA



Sendo:

1. Endereçamentos acessados (corresponde também ao limite de tempo da gravação)
2. Informação passada pela interface Bluetooth
3. Informação do áudio em formato digital armazenado
4. Informação do áudio em formato digital sendo capturado
5. Chave de liga e desliga do modo (manualmente pode ser definido somente como amplificador)

3.3.1.3. Modulo de comunicação HC-05

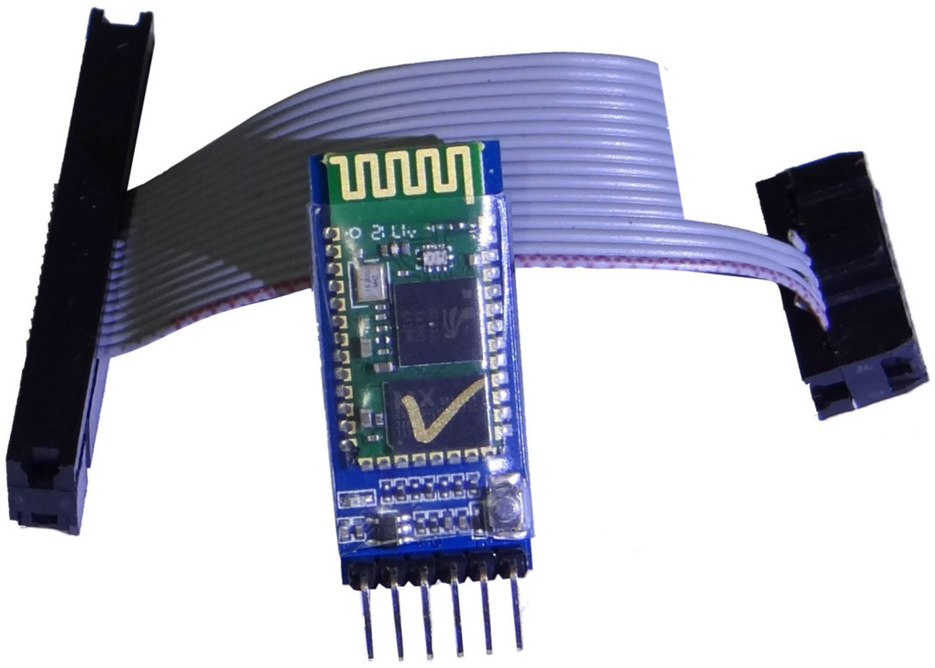
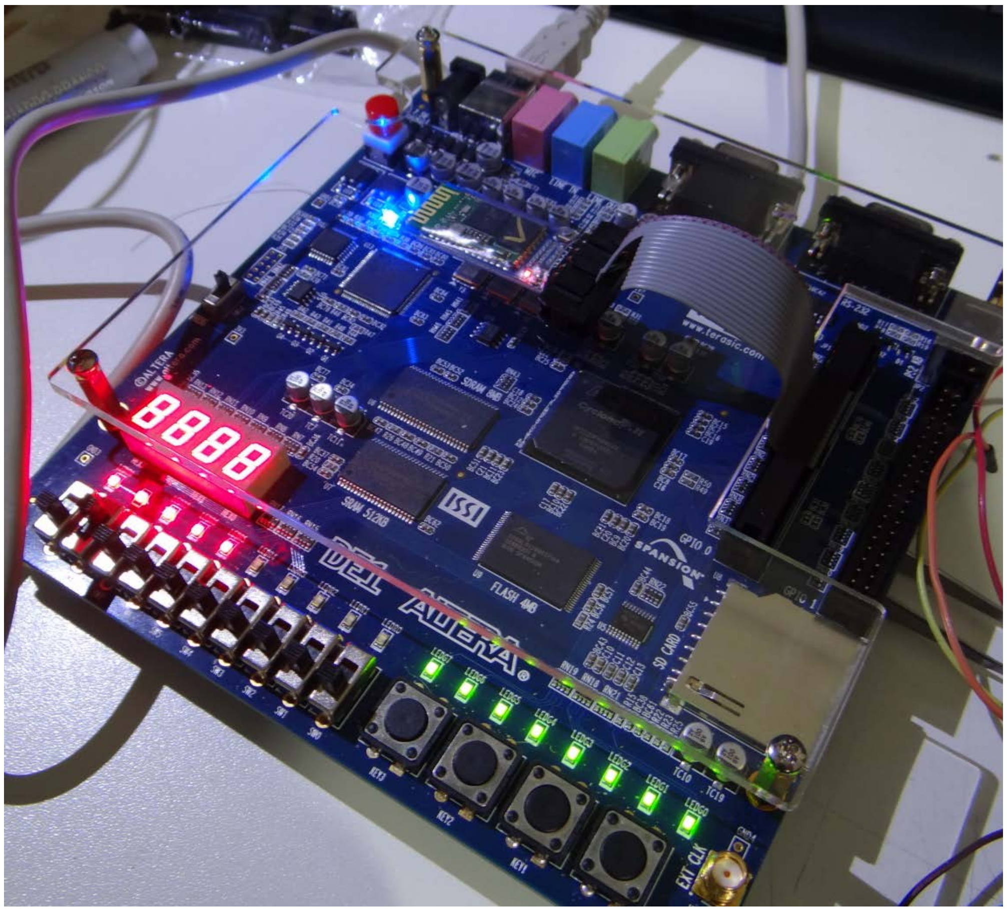


Figura do módulo HC-05 com o cabo de conexão

Deve ser feito a conexão no Header de expansão da esquerda.


A figura representa a comunicação do modulo bluetooth com a placa pelo cabo

3.4. Métricas e Cronogramas

Entendimento da plataforma a ser utilizada.
Desenvolvimento da comunicação Bluetooth.
Desenvolvimento de aplicativo de interface
Entendimento da utilização da memória externa
Soma de sinais de onda.
Testes de utilização do dispositivo e utilização.
Finalização da documentação.
Finalização do projeto.

4. Código

4.1. FPGA

4.1.1. Bloco de controle e integração

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

entity part1 is
   port (
            -- clock availble
            clock_50                 :     in    std_logic;
            clock_27                 :     in    bit_vector (0 to 1);
            clock_24                 :     in    bit_vector (0 to 1);

            -- audio control interface
            i2c_sdat                 :     inout std_logic;
            i2c_sclk                 :     out   std_logic;
            aud_xck                  :     out   std_logic;

            -- digital audio interface
            aud_daclrck        :     in    std_logic;
      aud_adclrck        :     in    std_logic;
            aud_bclk                 :     in    std_logic;
            aud_adcdat         :     in    std_logic;
            aud_dacdat         :     out   std_logic;

            --comunication
            gpio_0             :     in bit_vector (0 to 39);
            gpio_1             :     out bit_vector (0 to 39);

            --memory
            sram_addr    :     out   std_logic_vector (17 downto 0);
            sram_dq            :     inout std_logic_vector (15 downto 0);
            sram_we_n    :     out   bit;
            sram_oe_n    :     out   bit;
            sram_ub_n    :     out   bit;
            sram_lb_n    :     out   bit;
            sram_ce_n    :     out   bit;
```

```
        --interface
        hex3  :     out         bit_vector  (0 to 6);
        hex2  :     out         bit_vector  (0 to 6);
        hex1  :     out         bit_vector  (0 to 6);
        hex0  :     out         bit_vector  (0 to 6);
        --ledg      :     buffer      bit_vector  (0 to 7);
        ledg  :     buffer      std_logic_vector  (7 downto 0);
        ledr  :     buffer      std_logic_vector  (9 downto 0);
        sw          :     in                bit_vector  (9 downto 0);
        key   :     in                bit_vector  (0 to 3)


      );
end part1;

architecture behavior of part1 is
      -- code customization init
      component bluetooth_module
            port(
                  bluetoothData           :     inout       std_logic_vector(7
downto 0);
                  writedata_left,
                  writedata_right   :     in          std_logic_vector(23 downto
0);
                  read_s,
                  write_s                 :     in
      std_logic;
                  clock_24                :     in
      bit_vector (0 to 1);
                  gpio_0                  :     in
      bit_vector (0 to 39)--;
                  --ledg                        :     out
      std_logic_vector (7 downto 0)
            );
      end component;

      component SRAM
            port(
                  bluetoothData           :     inout       std_logic_vector(7
downto 0);
                  writedata_left,
                  writedata_right   :     inout       std_logic_vector(23 downto
0);
                  readdata_left,
                  readdata_right    :     in          std_logic_vector(23 downto
0);
                  read_s,
                  write_s                 :     in
      std_logic;
                  ledr                    :     buffer
      std_logic_vector  (9 downto 0);
                  ledg                    :     buffer
      std_logic_vector  (7 downto 0);
                  clock_50                :     in          std_logic;
                  clock_27                :     in          bit_vector (0
to 1);
```

```
                    clock_24                        :     in          bit_vector (0
to 1);
                    sram_addr                   :     out         std_logic_vector (17
downto 0);
                    sram_dq                         :     inout
      std_logic_vector (15 downto 0);
                    sram_we_n                   :     out         bit;
                    sram_oe_n                   :     out         bit;
                    sram_ub_n                   :     out         bit;
                    sram_lb_n                   :     out         bit;
                    sram_ce_n                   :     out         bit;
                    aud_daclrck                 :     in          std_logic;
                    aud_adclrck                 :     in          std_logic;
                    key                             :     in
      bit_vector (0 to 3);
                    sw                                  :     in
      bit_vector (9 downto 0);
                    hex3                            :     out         bit_vector (0
to 6);
                    hex2                            :     out         bit_vector (0
to 6);
                    hex1                            :     out         bit_vector (0
to 6);
                    hex0                            :     out         bit_vector (0
to 6)
            );
      end component;
      -- code customization end

   component clock_generator
      port(
                    clock_27 : in std_logic;
         reset     : in std_logic;
         aud_xck   : out std_logic
            );
   end component;

   component audio_and_video_config
      port(
                    clock_50,
                    reset                   : in    std_logic;
         i2c_sdat       : inout std_logic;
         i2c_sclk       : out   std_logic
            );
   end component;

   component audio_codec
      port(
                    clock_50,
                    reset,
                    read_s,
                    write_s           : in  std_logic;

                    writedata_left,
                    writedata_right   : in  std_logic_vector(23 downto 0);

         aud_adcdat,
                    aud_bclk,
                  aud_adclrck,
                  aud_daclrck              : in  std_logic;

         read_ready,
                  write_ready              : out std_logic;

                  readdata_left,
                  readdata_right     : out std_logic_vector(23 downto 0);

         aud_dacdat          : out std_logic
            );
   end component;

      signal bluetoothData    :     std_logic_vector(7 downto 0);

   signal
            clock2_50,
            read_ready,
            write_ready,
            read_s,
            write_s                      : std_logic;

   signal
            readdata_left,
            readdata_right    : std_logic_vector(23 downto 0);

   signal
            writedata_left,
            writedata_right   : std_logic_vector(23 downto 0);

   signal reset         : std_logic;
begin
   --reset <= not(to_stdulogic(key(0)));
      reset <= '0';
      clock2_50 <= clock_50;
   --your code goes here
   --writedata_left <= readdata_left when key(1)='0' else readdata_right ;
   --writedata_right <= readdata_right when key(1)='0' else
"000000000000000000000000";
   read_s <= read_ready;
   write_s <= write_ready and read_ready;

      gpio_1(0) <= to_bit(write_s);
      gpio_1(1) <= to_bit(read_s);

      my_clock_gen: clock_generator
      port map (
            clock2_50,
            reset,
            aud_xck
      );

   cfg: audio_and_video_config
      port map (
            clock_50,
            reset,
            i2c_sdat,
            i2c_sclk
      );

   codec: audio_codec
      port map (
            clock_50,
            reset,
            read_s,
            write_s,
            writedata_left,
            writedata_right,
            aud_adcdat,
            aud_bclk,
            aud_adclrck,
            aud_daclrck,
            read_ready,
            write_ready,
            readdata_left,
            readdata_right,
            aud_dacdat
      );

      -- customization init
      bluetooth : bluetooth_module
      port map(
            bluetoothData           => bluetoothData,
            writedata_left    => writedata_left,
            writedata_right   => writedata_right,
            read_s                        =>    read_s,
            write_s                       => write_s,
            clock_24                => clock_24,
            gpio_0                        =>    gpio_0--,
            --ledg                              => ledg
      );

      memory      :     SRAM
      port map(
            bluetoothData           => bluetoothData,
            writedata_left    => writedata_left,
            writedata_right   => writedata_right,
            readdata_left           => readdata_left,
            readdata_right    => readdata_right,
            read_s                        =>    read_s,
            write_s                       =>    write_s,
            ledr                          =>    ledr,
            ledg                          =>    ledg,
            clock_50                      => clock_50,
            clock_27                      => clock_27,
            clock_24                      => clock_24,
            sram_addr               => sram_addr,
            sram_dq                       => sram_dq,
            sram_we_n               => sram_we_n,
            sram_oe_n               => sram_oe_n,
            sram_ub_n               => sram_ub_n,
            sram_lb_n               => sram_lb_n,
            sram_ce_n               => sram_ce_n,
            aud_daclrck             => aud_daclrck,
            aud_adclrck             => aud_adclrck,
            key                           => key,
            sw                                  => sw,
            hex3                            => hex3,
            hex2                            => hex2,
            hex1                            => hex1,
            hex0                            => hex0
        );
        -- customization end

end behavior;
```

4.1.2. Componente de comunicação Bluetooth

```
-- projeto_base -> bluetooth_module
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.Std_Logic_Unsigned.all;

entity bluetooth_module is
      port(
            -- custom port
            bluetoothData           :     inout std_logic_vector(7 downto 0);

            -- audio
            writedata_left,
            writedata_right: in std_logic_vector(23 downto 0);
            read_s,
            write_s                 :     in          std_logic;
            -- clock
            clock_24 :  in bit_vector     (0 to 1);

            -- Comunication
            gpio_0      :     in bit_vector     (0 to 39)--;

            -- Interface
            -- ledg             :     out std_logic_vector    (7 downto 0)

      );
end bluetooth_module;

architecture bluetooth of bluetooth_module is

      -- clock reducer
      signal clockCounter48KHz      :     integer range 0 to 250 := 0;
      signal clock_48KHz                  :     std_ulogic := '0';

      -- rx of module
      signal rxCounter                    :     integer range 0 to 63 := 0;
      signal rxStream                     :     bit_vector (0 to 7);
      signal rxStandby                    :     bit := '1';
      signal startBit                     :     bit := '0';
      constant rxPin                            :     integer := 10;

      begin
            --interface
            --ledg <= bluetoothData;
            bluetoothData(7) <= to_stdulogic(rxStream(7));
            bluetoothData(6) <= to_stdulogic(rxStream(6));
            bluetoothData(5) <= to_stdulogic(rxStream(5));
            bluetoothData(4) <= to_stdulogic(rxStream(4));
            bluetoothData(3) <= to_stdulogic(rxStream(3));
            bluetoothData(2) <= to_stdulogic(rxStream(2));
            bluetoothData(1) <= to_stdulogic(rxStream(1));
            bluetoothData(0) <= to_stdulogic(rxStream(0));

            -- clock reducer;
            process (clock_24(0))
            begin
                if falling_edge(to_stdulogic(clock_24(0))) then

                    -- 48KHz
                    if (clockCounter48KHz < 250) THEN
                        clockCounter48KHz <= clockCounter48KHz + 1;
                    else
                        clockCounter48KHz <= 0;
                        clock_48KHz <= not clock_48KHz;
                    end if;

                end if;
        end process;


        -- RX read
        process (clock_48KHz)
            begin
                if falling_edge (clock_48KHz) then
                    if rxStandby = '0' then --received the start bit
                        if rxCounter < 48 then
                            rxCounter <= rxCounter + 1;
                        else
                            rxCounter <= 0;
                        end if;

                        case rxCounter is
                            when  5 =>
                                rxStream(0) <= gpio_0(rxPin);
                            when  8 =>
                                rxStream(1) <= gpio_0(rxPin);
                            when 13 =>
                                rxStream(2) <= gpio_0(rxPin);
                            when 18 =>
                                rxStream(3) <= gpio_0(rxPin);
                            when 23 =>
                                rxStream(4) <= gpio_0(rxPin);
                            when 28 =>
                                rxStream(5) <= gpio_0(rxPin);
                            when 33 =>
                                rxStream(6) <= gpio_0(rxPin);
                            when 38 =>
                                rxStream(7) <= gpio_0(rxPin);
                            when 43 =>
                                -- end bit sender;
                            when 47 =>
                                startBit <= '1';
                                rxStandby <= '1';

                            when others =>

                        end case;
                    else
                        if startBit = '0' and rxStandby = '1' then
                            rxStandby <= '0';
                        else
                            startBit <= gpio_0(rxPin);
                        end if;
```

```
                              end if;
                        end if;
            end process;
end bluetooth;
```

4.1.3. Modulo da memória SRAM

```
-- projeto_base -> memory_SRAM
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.Std_Logic_Unsigned.all;

entity SRAM is
      port(
            -- custom port
            bluetoothData           :     inout       std_logic_vector(7 downto
0);

            -- audio
            writedata_left,
            writedata_right   :     inout       std_logic_vector(23 downto 0);
            readdata_left,
            readdata_right    :     in          std_logic_vector(23 downto 0);
            read_s,
            write_s                 :     in                std_logic;
            ledr                    :     buffer      std_logic_vector   (9
downto 0);
            ledg                    :     buffer      std_logic_vector   (7
downto 0);

            -- clock availble
            clock_50                :     in          std_logic;
            clock_27                :     in          bit_vector (0 to 1);
            clock_24                :     in          bit_vector (0 to 1);

            --memory
            sram_addr               :     out         std_logic_vector (17
downto 0);
            sram_dq                 :     inout       std_logic_vector (15
downto 0);
            sram_we_n               :     out         bit;
            sram_oe_n               :     out         bit;
            sram_ub_n               :     out         bit;
            sram_lb_n               :     out         bit;
            sram_ce_n               :     out         bit;

            -- Audio Moment
            aud_daclrck             :     in          std_logic;
            aud_adclrck             :     in          std_logic;

            --interface
            key                     :     in                bit_vector (0
to 3);
            sw                            :     in
      bit_vector (9 downto 0);
            hex3                    :     out         bit_vector (0 to 6);
            hex2                    :     out         bit_vector (0 to 6);
            hex1                    :     out         bit_vector (0 to 6);
            hex0                    :     out         bit_vector (0 to 6)
      );
end SRAM;
```

```
architecture Memory_SRAM of SRAM is
      -- memory addressing
      signal      addressing17Bits  :     std_logic_vector  (17 downto 0) :=
"000000000000000000";
      signal      addressing17Max   :     std_logic_vector  (17 downto 0) :=
"111111111111111111";
      constant    oneBit                        :     std_logic_vector  (17
downto 0) := "000000000000000001";
      constant    addressingStart   :     std_logic_vector  (17 downto 0) :=
"000000000000000000";

      -- memory data
      constant ioLocker                 :     std_logic_vector  (15 downto 0)
:= "ZZZZZZZZZZZZZZZZ";

      -- memory controller
      signal      writeEnable           :     bit := '1';
      signal      readEnable            :     bit := '1';

      --memory Operation
      signal      mode                        :     integer range 0 to 7 := 0;
      signal      readyStats            :     bit := '0';
      constant memRec                   :     integer := 0;
      constant memRec1                  :     integer := 1;
      constant memRead                  :     integer := 2;
      constant memRead1                 :     integer := 3;
      constant memNULL                  :     integer := 4;
      constant memNULL1                 :     integer := 5;

      type decimal is array (0 to 15) of bit_vector (0 to 6);
      constant b2h : decimal := (
            0 =>  "0000001",
            1 =>  "1001111",
            2 =>  "0010010",
            3 =>  "0000110",
            4 =>  "1001100",
            5 =>  "0100100",
            6 =>  "0100000",
            7 =>  "0001101",
            8 =>  "0000000",
            9 =>  "0000100",
            10=>  "0001000",
            11=>  "1100000",
            12=>  "0110001",
            13=>  "1000010",
            14=>  "0110000",
            15=>  "0111000"
      );

      signal soundBuffer            :     std_logic_vector (23 downto 0);

      constant maxSRAM              :     integer := 567; -- 'magic' number do
not change i'm joking this is 50 mHz / 88.2048 kHz 567
      signal stateSRAM              :     integer range 0 to maxSRAM:=0;

      constant clockCounter   :     integer := 135;
      signal clockReducer           :     integer range 0 to clockCounter:=0;
```

```
    signal clockSignal              :     bit := '0';

    signal appMode                        :     integer range 0 to 7 := 0;
    signal appSRA                         :     integer range 1 to 4 := 1;


    begin
            -- display the address
            hex3 <= b2h(to_integer(ieee.numeric_std.unsigned(addressing17Bits(15
downto 12))));
            hex2 <= b2h(to_integer(ieee.numeric_std.unsigned(addressing17Bits(11
downto  9))));
            hex1 <= b2h(to_integer(ieee.numeric_std.unsigned(bluetoothData( 7
downto  4))));
            hex0 <= b2h(to_integer(ieee.numeric_std.unsigned(bluetoothData( 3
downto  0))));
            -- hex1 <=
b2h(to_integer(ieee.numeric_std.unsigned(addressing17Bits( 9 downto  6))));
            -- hex0 <=
b2h(to_integer(ieee.numeric_std.unsigned(addressing17Bits( 5 downto  2))));
            sram_addr <= addressing17Bits;
            --ledg <= bluetoothData;

            -- memory
            sram_ub_n <= '0';
            sram_lb_n <= '0';
            sram_ce_n <= '0';
            writedata_left    <= (to_stdlogicvector((to_bitvector(soundBuffer)
sra appSRA))      + to_stdlogicvector(to_bitvector(readdata_left)  sra 1));
            writedata_right   <= (to_stdlogicvector((to_bitvector(soundBuffer)
sra appSRA))      + to_stdlogicvector(to_bitVector(readdata_right) sra 1));

            -- clock reducer;
            reduce_clock:process (clock_50)
                  begin
                        -- ===================================
                        -- Bluetooth Setup App Interface
                        -- ===================================
                        -- Echo (in decimal is 111 to 148)
                        -- ===================================
                        if (  bluetoothData = x"6F" or bluetoothData = x"FF"
    )then
                              addressing17Max <= "00110000110011111111";--1
                              appMode <= 1;
                        elsif bluetoothData = x"70" then
                              addressing17Max <= "00110011101011011";--2
                              appSRA <= 1;
                        elsif bluetoothData = x"71" then
                              addressing17Max <= "00110101101111001";--3
                              appSRA <= 1;
                        elsif bluetoothData = x"72" then
                              addressing17Max <= "00110111110010011";--4
                              appSRA <= 1;
                        elsif bluetoothData = x"73" then
                              addressing17Max <= "00111001110101101";--5
                              appSRA <= 1;
                        elsif bluetoothData = x"74" then
                              addressing17Max <= "00111011111001001";--6
```

```
        appSRA <= 1;
elsif bluetoothData = x"75" then
        addressing17Max <= "00111011111001001";--7
        appSRA <= 1;
elsif bluetoothData = x"76" then
        addressing17Max <= "00111111111111111";--8
        appSRA <= 1;

elsif bluetoothData = x"79" then
        addressing17Max <= "00110001100111111";--1
        appSRA <= 2;
elsif bluetoothData = x"7A" then
        addressing17Max <= "00110011101011101";--2
        appSRA <= 2;
elsif bluetoothData = x"7B" then
        addressing17Max <= "00110101101111000";--3
        appSRA <= 2;
elsif bluetoothData = x"7C" then
        addressing17Max <= "00110111100100111";--4
        appSRA <= 2;
elsif bluetoothData = x"7D" then
        addressing17Max <= "00111001110101110";--5
        appSRA <= 2;
elsif bluetoothData = x"7E" then
        addressing17Max <= "00111011111001001";--6
        appSRA <= 2;
elsif bluetoothData = x"7F" then
        addressing17Max <= "00111101111100100";--7
        appSRA <= 2;
elsif bluetoothData = x"80" then
        addressing17Max <= "00111111111111111";--8
        appSRA <= 2;

elsif bluetoothData = x"83" then
        addressing17Max <= "00110001100111111";--1
        appSRA <= 3;
elsif bluetoothData = x"84" then
        addressing17Max <= "00110011101011101";--2
        appSRA <= 3;
elsif bluetoothData = x"85" then
        addressing17Max <= "00110101101111000";--3
        appSRA <= 3;
elsif bluetoothData = x"86" then
        addressing17Max <= "00110111100100111";--4
        appSRA <= 3;
elsif bluetoothData = x"87" then
        addressing17Max <= "00111001110101110";--5
        appSRA <= 3;
elsif bluetoothData = x"88" then
        addressing17Max <= "00111011111001001";--6
        appSRA <= 3;
elsif bluetoothData = x"89" then
        addressing17Max <= "00111101111100100";--7
        appSRA <= 3;
elsif bluetoothData = x"8A" then
        addressing17Max <= "00111111111111111";--8
        appSRA <= 3;
```

```
                                    elsif bluetoothData = x"8D" then
                                          addressing17Max <= "001100011001111111";--1
                                          appSRA <= 4;
                                    elsif bluetoothData = x"8E" then
                                          addressing17Max <= "001100111010111011";--2
                                          appSRA <= 4;
                                    elsif bluetoothData = x"8F" then
                                          addressing17Max <= "001101011011110001";--3
                                          appSRA <= 4;
                                    elsif bluetoothData = x"90" then
                                          addressing17Max <= "001101111100100111";--4
                                          appSRA <= 4;
                                    elsif bluetoothData = x"91" then
                                          addressing17Max <= "001110011101011101";--5
                                          appSRA <= 4;
                                    elsif bluetoothData = x"92" then
                                          addressing17Max <= "001110111110010011";--6
                                          appSRA <= 4;
                                    elsif bluetoothData = x"93" then
                                          addressing17Max <= "001111011111001001";--7
                                          appSRA <= 4;
                                    elsif bluetoothData = x"94" then
                                          addressing17Max <= "001111111111111111";--8
                                          appSRA <= 4;

                                    -- ===================================
                                    -- Reverberation
                                    -- ===================================
                                    elsif bluetoothData = x"64" then

                                          -- bluetoothDecimal = 100
                                          addressing17Max <= "000011111111111111";--0
                                          appSRA <= 1;
                                    else
                                    end if;
                                    -- ===================================

                                    if key(3) = '1' then -- this will force to start
                                          if mode = memRec then
                                                sram_oe_n   <=    '0';
                                                sram_we_n   <=    '0';
                                          elsif mode = memRec1 then
                                                sram_oe_n   <=    '0';
                                                sram_we_n   <=    '0';
                                                if key(1) = '1' then
                                                      if aud_daclrck = '1' then
                                                            if appMode = 1 or appMode = 0
then
                                                                  sram_dq     <=
(to_stdlogicvector(to_bitvector(writedata_left(23 downto 8))      sra 0));
                                                            elsif appMode = 2 then
                                                                  sram_dq     <=
(to_stdlogicvector(to_bitvector(writedata_left(23 downto 8))      sra 0));
                                                            end if;
                                                            --sram_dq <= readdata_left(23
downto 8);
                                                            ledg <= readdata_left(23 downto
16);
```

```
                                        else
                                              if appMode = 1 or appMode = 0
then
                                                    sram_dq     <=
(to_stdlogicvector(to_bitvector(writedata_right(23 downto 8))     sra 0));
                                              elsif appMode = 2 then
                                                    sram_dq     <=
(to_stdlogicvector(to_bitvector(writedata_right(23 downto 8))     sra 0));
                                              end if;
                                              --sram_dq <= readdata_right(23
downto 8);
                                              ledg <= readdata_right(23 downto
16);
                                        end if;
                                  else
                                        sram_dq <= ("0000000000000000");
                                  end if;
                                  -- (19 downto 4);
                                  -- (23 downto 8);
                            elsif mode = memRead then
                                  sram_oe_n   <=    '0';
                                  sram_we_n   <=    '1';
                            elsif mode = memRead1 then
                                  sram_oe_n   <=    '0';
                                  sram_we_n   <=    '1';
                                  soundBuffer <= (""&sram_dq&"00000000");
                                  ledr <=     sram_dq(15 downto 6);
                            else
                                  sram_oe_n   <=    '1';
                                  sram_we_n   <=    '1';
                                  sram_dq           <=    ioLocker;
                            end if;
                      end if;

                      if write_s = '0' then
                            if falling_edge(clock_50) and sw(9) = '0'  then
                                  if stateSRAM < maxSRAM then
                                        stateSRAM <= stateSRAM + 1;
                                  end if;


                                  if          94 < stateSRAM and stateSRAM <=
189 then
                                        mode <= memRead; -- prepare to read
                                  elsif 189 < stateSRAM and stateSRAM <= 283
then
                                        mode <= memRead1; -- read
                                  elsif 283 < stateSRAM and stateSRAM <= 378
then
                                        mode <= memRec; -- prepare to Record
                                  elsif 378 < stateSRAM and stateSRAM <= 472
then
                                        mode <= memRec1; -- record
                                  else
                                        mode <= memNULL; -- waiting to change
the address
                                  end if;
```

```
                        end if;
                else
                        stateSRAM <= 0;
                end if;
        end process;

        -- write_s velocidade aproximadamente de 88.8889 kHz
        -- clock exato de 88.2048 kHz
        store:process (write_s)
            begin
                if falling_edge(to_stdulogic(to_bit(write_s))) and sw(9)
= '0' then
                        if addressing17Bits < addressing17Max then
                            addressing17Bits <= addressing17Bits +
oneBit;
                        else
                            addressing17Bits <= addressingStart;
                        end if;
                end if;
                if key(2) = '0' then
                        addressing17Bits <= addressingStart;
                end if;
        end process;

end Memory_SRAM;
```

4.1.4. Configuração de áudio e vídeo

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State *
 *  of California and by the laws of the United States of America.          *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module sends and receives data from the audio's and TV in's         *
 *  control registers for the chips on Altera's DE1 board. Plus, it can     *
 *  send and receive data from the TRDB_DC2 and TRDB_LCM add-on modules.    *
 *                                                                           *
 *****************************************************************************/

module audio_and_video_config (
	// Inputs
	CLOCK_50,
	reset,

	// Bidirectionals
	I2C_SDAT,
	I2C_SCLK
);

/*****************************************************************************
 *                           Parameter Declarations                          *
 *****************************************************************************/

parameter I2C_BUS_MODE           = 1'b0;
parameter CFG_TYPE                     = 8'h01;

parameter MIN_ROM_ADDRESS        = 6'h00;
parameter MAX_ROM_ADDRESS        = 6'h32;

parameter AUD_LINE_IN_LC         = 9'h01A;
parameter AUD_LINE_IN_RC         = 9'h01A;
parameter AUD_LINE_OUT_LC        = 9'h07B;
parameter AUD_LINE_OUT_RC        = 9'h07B;
parameter AUD_ADC_PATH           = 9'd149;
parameter AUD_DAC_PATH           = 9'h006;
parameter AUD_POWER                    = 9'h000;
parameter AUD_DATA_FORMAT        = 9'd73;
parameter AUD_SAMPLE_CTRL        = 9'd0;
parameter AUD_SET_ACTIVE         = 9'h001;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/
// Inputs
input                   CLOCK_50;
input                   reset;

// Bidirectionals
inout                   I2C_SDAT;                          //    I2C Data
output                  I2C_SCLK;                          //    I2C Clock
/*****************************************************************************
 *                           Constant Declarations                           *
 *****************************************************************************/


/*****************************************************************************
 *                 Internal wires and registers Declarations                 *
 *****************************************************************************/
// Internal Wires
wire                    clk_400KHz;
wire                    start_and_stop_en;
wire                    change_output_bit_en;

wire                    enable_clk;

wire                    send_start_bit;
wire                    send_stop_bit;

wire            [7:0] auto_init_data;
wire                    auto_init_transfer_data;
wire                    auto_init_start_bit;
wire                    auto_init_stop_bit;
wire                    auto_init_complete;
wire                    auto_init_error;
wire                    transfer_data;
wire                    transfer_complete;

wire                    i2c_ack;
wire        [7:0] i2c_received_data;

// Internal Registers
reg               [7:0] data_to_transfer;
reg               [2:0] num_bits_to_transfer;

/*****************************************************************************
 *                          Finite State Machine(s)                          *
 *****************************************************************************/


/*****************************************************************************
 *                              Sequential logic                             *
 *****************************************************************************/

always @(posedge CLOCK_50)
begin
      if (reset)
      begin
            data_to_transfer        <= 8'h00;
            num_bits_to_transfer    <= 3'h0;
      end
      else
            if (auto_init_complete == 1'b0)
            begin
                  data_to_transfer        <= auto_init_data;
                  num_bits_to_transfer    <= 3'h7;
            end
end

/*****************************************************************************
 *                            Combinational logic                            *
 *****************************************************************************/

assign transfer_data = auto_init_transfer_data;

assign send_start_bit = auto_init_start_bit;

assign send_stop_bit = auto_init_stop_bit;

/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/

Altera_UP_Slow_Clock_Generator Clock_Generator_400KHz (
      // Inputs
      .clk                          (CLOCK_50),
      .reset                              (reset),

      .enable_clk                   (enable_clk),

      // Bidirectionals

      // Outputs
	.new_clk			(clk_400KHz),

	.rising_edge			(),
	.falling_edge			(),

	.middle_of_high_level		(start_and_stop_en),
	.middle_of_low_level		(change_output_bit_en)
);
defparam
	Clock_Generator_400KHz.COUNTER_BITS = 10, // 4, //
	Clock_Generator_400KHz.COUNTER_INC  = 10'h001; // 4'h1; //

Altera_UP_I2C_AV_Auto_Initialize Auto_Initialize (
	// Inputs
	.clk			(CLOCK_50),
	.reset				(reset),

	.clear_error		(1'b1),

	.ack			(i2c_ack),
	.transfer_complete	(transfer_complete),

	// Bidirectionals

	// Outputs
	.data_out		(auto_init_data),
	.transfer_data		(auto_init_transfer_data),
	.send_start_bit		(auto_init_start_bit),
	.send_stop_bit		(auto_init_stop_bit),

	.auto_init_complete	(auto_init_complete),
	.auto_init_error	(auto_init_error)
);
defparam
	Auto_Initialize.MIN_ROM_ADDRESS	= MIN_ROM_ADDRESS,
	Auto_Initialize.MAX_ROM_ADDRESS	= MAX_ROM_ADDRESS,

	Auto_Initialize.AUD_LINE_IN_LC	= AUD_LINE_IN_LC,
	Auto_Initialize.AUD_LINE_IN_RC	= AUD_LINE_IN_RC,
	Auto_Initialize.AUD_LINE_OUT_LC	= AUD_LINE_OUT_LC,
	Auto_Initialize.AUD_LINE_OUT_RC	= AUD_LINE_OUT_RC,
	Auto_Initialize.AUD_ADC_PATH	= AUD_ADC_PATH,
	Auto_Initialize.AUD_DAC_PATH	= AUD_DAC_PATH,
	Auto_Initialize.AUD_POWER		= AUD_POWER,
	Auto_Initialize.AUD_DATA_FORMAT	= AUD_DATA_FORMAT,
	Auto_Initialize.AUD_SAMPLE_CTRL	= AUD_SAMPLE_CTRL,
	Auto_Initialize.AUD_SET_ACTIVE	= AUD_SET_ACTIVE;

Altera_UP_I2C I2C_Controller (
	// Inputs
	.clk				(CLOCK_50),
	.reset					(reset),

	.clear_ack			(1'b1),

	.clk_400KHz			(clk_400KHz),
	.start_and_stop_en		(start_and_stop_en),
	.change_output_bit_en	(change_output_bit_en),
      .send_start_bit               (send_start_bit),
      .send_stop_bit                (send_stop_bit),

      .data_in                      (data_to_transfer),
      .transfer_data                (transfer_data),
      .read_byte                    (1'b0),
      .num_bits_to_transfer   (num_bits_to_transfer),

      // Bidirectionals
      .i2c_sdata                    (I2C_SDAT),

      // Outputs
      .i2c_sclk                     (I2C_SCLK),
      .i2c_scen                     (),

      .enable_clk                   (enable_clk),

      .ack                          (i2c_ack),
      .data_from_i2c                (i2c_received_data),
      .transfer_complete            (transfer_complete)
);
defparam
      I2C_Controller.I2C_BUS_MODE   = I2C_BUS_MODE;

endmodule
```

4.1.4.1. Modulo I2C da configuração de áudio

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner   *
 *  under Altera's Megafunction Partnership Program may be used only to       *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other  *
 *  use of such megafunction design, net list, support information, device    *
 *  programming or simulation file, or any other related documentation or     *
 *  information is prohibited for any other purpose, including, but not       *
 *  limited to modification, reverse engineering, de-compiling, or use with   *
 *  any other silicon devices, unless such use is explicitly licensed under   *
 *  a separate agreement with Altera or a megafunction partner.  Title to     *
 *  the intellectual property, including patents, copyrights, trademarks,     *
 *  trade secrets, or maskworks, embodied in any such megafunction design,    *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a    *
 *  megafunction partner, remains with Altera, the megafunction partner, or   *
 *  their respective licensors.  No other licenses, including any licenses    *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice   *
 *  is attached violates this copyright.                                      *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR    *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,   *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL    *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING    *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 ******************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module loads data into the Audio and Video chips' control            *
 *  registers after system reset.                                            *
 *                                                                           *
 ******************************************************************************/

module Altera_UP_I2C_AV_Auto_Initialize (
	// Inputs
	clk,
	reset,

	clear_error,

	ack,
	transfer_complete,
	// Bidirectionals

	// Outputs
	data_out,
	transfer_data,
	send_start_bit,
	send_stop_bit,

	auto_init_complete,
	auto_init_error
);

/*****************************************************************************
 *                           Parameter Declarations                          *
 *****************************************************************************/

parameter MIN_ROM_ADDRESS	= 6'h00;
parameter MAX_ROM_ADDRESS	= 6'h32;

parameter AUD_LINE_IN_LC	= 9'h01A;
parameter AUD_LINE_IN_RC	= 9'h01A;
parameter AUD_LINE_OUT_LC	= 9'h07B;
parameter AUD_LINE_OUT_RC	= 9'h07B;
parameter AUD_ADC_PATH		= 9'h0F8;
parameter AUD_DAC_PATH		= 9'h006;
parameter AUD_POWER			= 9'h000;
parameter AUD_DATA_FORMAT	= 9'h001;
parameter AUD_SAMPLE_CTRL	= 9'h002;
parameter AUD_SET_ACTIVE	= 9'h001;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/
// Inputs
input				clk;
input				reset;

input				clear_error;

input				ack;
input				transfer_complete;

// Bidirectionals

// Outputs
output		reg	[7:0]	data_out;
output		reg			transfer_data;
output		reg			send_start_bit;
output		reg			send_stop_bit;


output					auto_init_complete;
output		reg			auto_init_error;

/*****************************************************************************
 *                           Constant Declarations                           *
 *****************************************************************************/
```

```
// States
localparam  AUTO_STATE_0_CHECK_STATUS           = 3'h0,
            AUTO_STATE_1_SEND_START_BIT           = 3'h1,
            AUTO_STATE_2_TRANSFER_BYTE_1  = 3'h2,
            AUTO_STATE_3_TRANSFER_BYTE_2  = 3'h3,
            AUTO_STATE_4_WAIT                     = 3'h4,
            AUTO_STATE_5_SEND_STOP_BIT            = 3'h5,
            AUTO_STATE_6_INCREASE_COUNTER = 3'h6,
            AUTO_STATE_7_DONE                     = 3'h7;

/*****************************************************************************
 *                  Internal wires and registers Declarations                *
 *****************************************************************************/
// Internal Wires
wire                        change_state;

wire                        finished_auto_init;

// Internal Registers
reg                 [5:0] rom_address_counter;
reg                 [25:0]      rom_data;

// State Machine Registers
reg                 [2:0] ns_i2c_auto_init;
reg                 [2:0] s_i2c_auto_init;

/*****************************************************************************
 *                          Finite State Machine(s)                          *
 *****************************************************************************/

always @(posedge clk)
begin
      if (reset == 1'b1)
            s_i2c_auto_init <= AUTO_STATE_0_CHECK_STATUS;
      else
            s_i2c_auto_init <= ns_i2c_auto_init;
end

always @(*)
begin
      // Defaults
      ns_i2c_auto_init = AUTO_STATE_0_CHECK_STATUS;

    case (s_i2c_auto_init)
      AUTO_STATE_0_CHECK_STATUS:
            begin
                  if (finished_auto_init == 1'b1)
                        ns_i2c_auto_init = AUTO_STATE_7_DONE;
                  else if (rom_data[25] == 1'b1)
                        ns_i2c_auto_init = AUTO_STATE_1_SEND_START_BIT;
                  else
                        ns_i2c_auto_init = AUTO_STATE_3_TRANSFER_BYTE_2;
            end
      AUTO_STATE_1_SEND_START_BIT:
            begin
                  if (change_state == 1'b1)
                        ns_i2c_auto_init = AUTO_STATE_2_TRANSFER_BYTE_1;
                  else
```

```
                        ns_i2c_auto_init = AUTO_STATE_1_SEND_START_BIT;
                end
        AUTO_STATE_2_TRANSFER_BYTE_1:
                begin
                        if (change_state == 1'b1)
                                ns_i2c_auto_init = AUTO_STATE_3_TRANSFER_BYTE_2;
                        else
                                ns_i2c_auto_init = AUTO_STATE_2_TRANSFER_BYTE_1;
                end
        AUTO_STATE_3_TRANSFER_BYTE_2:
                begin
                        if ((change_state == 1'b1) && (rom_data[24] == 1'b1))
                                ns_i2c_auto_init = AUTO_STATE_4_WAIT;
                        else if (change_state == 1'b1)
                                ns_i2c_auto_init = AUTO_STATE_6_INCREASE_COUNTER;
                        else
                                ns_i2c_auto_init = AUTO_STATE_3_TRANSFER_BYTE_2;
                end
        AUTO_STATE_4_WAIT:
                begin
                        if (transfer_complete == 1'b0)
                                ns_i2c_auto_init = AUTO_STATE_5_SEND_STOP_BIT;
                        else
                                ns_i2c_auto_init = AUTO_STATE_4_WAIT;
                end
        AUTO_STATE_5_SEND_STOP_BIT:
                begin
                        if (transfer_complete == 1'b1)
                                ns_i2c_auto_init = AUTO_STATE_6_INCREASE_COUNTER;
                        else
                                ns_i2c_auto_init = AUTO_STATE_5_SEND_STOP_BIT;
                end
        AUTO_STATE_6_INCREASE_COUNTER:
                begin
                        ns_i2c_auto_init = AUTO_STATE_0_CHECK_STATUS;
                end
        AUTO_STATE_7_DONE:
                begin
                        ns_i2c_auto_init = AUTO_STATE_7_DONE;
                end
        default:
                begin
                        ns_i2c_auto_init = AUTO_STATE_0_CHECK_STATUS;
                end
        endcase
end

/*****************************************************************************
 *                             Sequential logic                              *
 *****************************************************************************/

// Output Registers
always @(posedge clk)
begin
        if (reset == 1'b1)
                data_out <= 8'h00;
        else if (s_i2c_auto_init == AUTO_STATE_1_SEND_START_BIT)
                data_out <= rom_data[23:16];
```

```
      else if (s_i2c_auto_init == AUTO_STATE_0_CHECK_STATUS)
            data_out <= rom_data[15: 8];
      else if (s_i2c_auto_init == AUTO_STATE_2_TRANSFER_BYTE_1)
            data_out <= rom_data[15: 8];
      else if (s_i2c_auto_init == AUTO_STATE_3_TRANSFER_BYTE_2)
            data_out <= rom_data[ 7: 0];
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            transfer_data <= 1'b0;
      else if (transfer_complete == 1'b1)
            transfer_data <= 1'b0;
      else if (s_i2c_auto_init == AUTO_STATE_1_SEND_START_BIT)
            transfer_data <= 1'b1;
      else if (s_i2c_auto_init == AUTO_STATE_2_TRANSFER_BYTE_1)
            transfer_data <= 1'b1;
      else if (s_i2c_auto_init == AUTO_STATE_3_TRANSFER_BYTE_2)
            transfer_data <= 1'b1;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            send_start_bit <= 1'b0;
      else if (transfer_complete == 1'b1)
            send_start_bit <= 1'b0;
      else if (s_i2c_auto_init == AUTO_STATE_1_SEND_START_BIT)
            send_start_bit <= 1'b1;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            send_stop_bit <= 1'b0;
      else if (transfer_complete == 1'b1)
            send_stop_bit <= 1'b0;
      else if (s_i2c_auto_init == AUTO_STATE_5_SEND_STOP_BIT)
            send_stop_bit <= 1'b1;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            auto_init_error <= 1'b0;
      else if (clear_error == 1'b1)
            auto_init_error <= 1'b0;
      else if ((s_i2c_auto_init == AUTO_STATE_6_INCREASE_COUNTER) & ack)
            auto_init_error <= 1'b1;
end

// Internal Registers
always @(posedge clk)
begin
      if (reset == 1'b1)
            rom_address_counter <= MIN_ROM_ADDRESS;
      else if (s_i2c_auto_init == AUTO_STATE_6_INCREASE_COUNTER)
```

```
            rom_address_counter <= rom_address_counter + 6'h01;
end

/*****************************************************************************
 *                              Combinational logic                          *
 *****************************************************************************/
// Output Assignments
assign auto_init_complete = (s_i2c_auto_init == AUTO_STATE_7_DONE);

// Internals Assignments
assign change_state     = transfer_complete & transfer_data;

assign finished_auto_init = (rom_address_counter == MAX_ROM_ADDRESS);

always @(*)
begin
      case (rom_address_counter)
      //    Audio Config Data
      0           :      rom_data    <=    {10'h334, 7'h0, AUD_LINE_IN_LC};
      1           :      rom_data    <=    {10'h334, 7'h1, AUD_LINE_IN_RC};
      2           :      rom_data    <=    {10'h334, 7'h2, AUD_LINE_OUT_LC};
      3           :      rom_data    <=    {10'h334, 7'h3, AUD_LINE_OUT_RC};
      4           :      rom_data    <=    {10'h334, 7'h4, AUD_ADC_PATH};
      5           :      rom_data    <=    {10'h334, 7'h5, AUD_DAC_PATH};
      6           :      rom_data    <=    {10'h334, 7'h6, AUD_POWER};
      7           :      rom_data    <=    {10'h334, 7'h7, AUD_DATA_FORMAT};
      8           :      rom_data    <=    {10'h334, 7'h8, AUD_SAMPLE_CTRL};
      9           :      rom_data    <=    {10'h334, 7'h9, AUD_SET_ACTIVE};
      //    Video Config Data
      10          :      rom_data    <=    26'h3401500;
      11          :      rom_data    <=    26'h3401741;
      12          :      rom_data    <=    26'h3403a16;
      13          :      rom_data    <=    26'h3405004;
      14          :      rom_data    <=    26'h340c305;
      15          :      rom_data    <=    26'h340c480;
      16          :      rom_data    <=    26'h3400e80;
      17          :      rom_data    <=    26'h3405020;
      18          :      rom_data    <=    26'h3405218;
      19          :      rom_data    <=    26'h34058ed;
      20          :      rom_data    <=    26'h34077c5;
      21          :      rom_data    <=    26'h3407c93;
      22          :      rom_data    <=    26'h3407d00;
      23          :      rom_data    <=    26'h340d048;
      24          :      rom_data    <=    26'h340d5a0;
      25          :      rom_data    <=    26'h340d7ea;
      26          :      rom_data    <=    26'h340e43e;
      27          :      rom_data    <=    26'h340ea0f;
      28          :      rom_data    <=    26'h3403112;
      29          :      rom_data    <=    26'h3403281;
      30          :      rom_data    <=    26'h3403384;
      31          :      rom_data    <=    26'h34037A0;
      32          :      rom_data    <=    26'h340e580;
      33          :      rom_data    <=    26'h340e603;
      34          :      rom_data    <=    26'h340e785;
      35          :      rom_data    <=    26'h3405000;
      36          :      rom_data    <=    26'h3405100;
      37          :      rom_data    <=    26'h3400070;
      38          :      rom_data    <=    26'h3401010;
```

```
    39          :     rom_data    <=    26'h3400482;
    40          :     rom_data    <=    26'h3400860;
    41          :     rom_data    <=    26'h3400a18;
    42          :     rom_data    <=    26'h3401100;
    43          :     rom_data    <=    26'h3402b00;
    44          :     rom_data    <=    26'h3402c8c;
    45          :     rom_data    <=    26'h3402df2;
    46          :     rom_data    <=    26'h3402eee;
    47          :     rom_data    <=    26'h3402ff4;
    48          :     rom_data    <=    26'h34030d2;
    49          :     rom_data    <=    26'h3400e05;
    default     :     rom_data    <=    26'h1000000;
    endcase
end

/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/


endmodule
```

4.1.4.2. Modulo Redutor de clock da configuração de áudio

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not     *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module can create clock signals that have a frequency lower         *
 *  than those a PLL can generate.                                           *
 *                                                                           *
 * Revision: 1.1                                                             *
 *                                                                           *
 * Used in IP Cores:                                                         *
 *   Altera UP Avalon Audio and Video Config                                 *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_Slow_Clock_Generator (
	// Inputs
	clk,
	reset,
	enable_clk,

	// Bidirectionals

	// Outputs
	new_clk,

	rising_edge,
	falling_edge,

	middle_of_high_level,
	middle_of_low_level
);

/*****************************************************************************
 *                           Parameter Declarations                          *
 *****************************************************************************/

parameter COUNTER_BITS  = 10;
parameter COUNTER_INC   = 10'h001;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/

// Inputs
input                       clk;
input                       reset;

input                       enable_clk;

// Bidirectionals

// Outputs
output      reg                 new_clk;

output      reg                 rising_edge;
output      reg                 falling_edge;

output      reg                 middle_of_high_level;
output      reg                 middle_of_low_level;

/*****************************************************************************
 *                           Constant Declarations                           *
 *****************************************************************************/

/*****************************************************************************
 *                 Internal wires and registers Declarations                 *
 *****************************************************************************/

// Internal Wires

// Internal Registers
reg                 [COUNTER_BITS:1]  clk_counter;

// State Machine Registers
```

```
/****************************************************************************
 *                         Finite State Machine(s)                          *
 ****************************************************************************/


/****************************************************************************
 *                            Sequential logic                              *
 ****************************************************************************/

always @(posedge clk)
begin
      if (reset == 1'b1)
            clk_counter <= {COUNTER_BITS{1'b0}};
      else if (enable_clk == 1'b1)
            clk_counter <= clk_counter + COUNTER_INC;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            new_clk     <= 1'b0;
      else
            new_clk     <= clk_counter[COUNTER_BITS];
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            rising_edge <= 1'b0;
      else
            rising_edge <= (clk_counter[COUNTER_BITS] ^ new_clk) & ~new_clk;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            falling_edge <= 1'b0;
      else
            falling_edge <= (clk_counter[COUNTER_BITS] ^ new_clk) & new_clk;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            middle_of_high_level <= 1'b0;
      else
            middle_of_high_level <=
                  clk_counter[COUNTER_BITS] &
                  ~clk_counter[(COUNTER_BITS - 1)] &
                  (&(clk_counter[(COUNTER_BITS - 2):1]));
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            middle_of_low_level <= 1'b0;
      else
            middle_of_low_level <=
                  ~clk_counter[COUNTER_BITS] &
                  ~clk_counter[(COUNTER_BITS - 1)] &
                  (&(clk_counter[(COUNTER_BITS - 2):1]));
end


/*****************************************************************************
 *                          Combinational logic                              *
 *****************************************************************************/

// Output Assignments

// Internal Assignments

/*****************************************************************************
 *                            Internal Modules                               *
 *****************************************************************************/

endmodule
```

4.1.4.3. Controle I2C do modulo de áudio

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module sends and receives data to/from the DE1's audio and TV        *
 *  peripherals' control registers.                                          *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_I2C (
	// Inputs
	clk,
	reset,

	clear_ack,

	clk_400KHz,
	start_and_stop_en,
	change_output_bit_en,

	send_start_bit,
	send_stop_bit,

	data_in,
	transfer_data,
	read_byte,
	num_bits_to_transfer,

	// Bidirectionals
	i2c_sdata,

	// Outputs
	i2c_sclk,
	i2c_scen,

	enable_clk,

	ack,
	data_from_i2c,
	transfer_complete
);

/*****************************************************************************
 *                          Parameter Declarations                          *
 *****************************************************************************/

parameter I2C_BUS_MODE = 1'b0;

/*****************************************************************************
 *                            Port Declarations                             *
 *****************************************************************************/
// Inputs
input                   clk;
input                   reset;

input                   clear_ack;

input                   clk_400KHz;
input                   start_and_stop_en;
input                   change_output_bit_en;

input                   send_start_bit;
input                   send_stop_bit;

input       [7:0]  data_in;
input                   transfer_data;
input                   read_byte;
input       [2:0]  num_bits_to_transfer;

// Bidirectionals
inout                   i2c_sdata;                    //    I2C Data

// Outputs
output                        i2c_sclk;                     //    I2C
Clock
output      reg               i2c_scen;
output                          enable_clk;

output      reg                 ack;
output      reg     [7:0] data_from_i2c;
output                          transfer_complete;

/****************************************************************************
 *                          Constant Declarations                           *
 ****************************************************************************/
// states
localparam  I2C_STATE_0_IDLE              = 3'h0,
                  I2C_STATE_1_PRE_START         = 3'h1,
                  I2C_STATE_2_START_BIT         = 3'h2,
                  I2C_STATE_3_TRANSFER_BYTE     = 3'h3,
                  I2C_STATE_4_TRANSFER_ACK      = 3'h4,
                  I2C_STATE_5_STOP_BIT          = 3'h5,
                  I2C_STATE_6_COMPLETE          = 3'h6;

/****************************************************************************
 *                Internal wires and registers Declarations                 *
 ****************************************************************************/
// Internal Wires

// Internal Registers
reg                 [2:0] current_bit;
reg                 [7:0] current_byte;

// State Machine Registers
reg                 [2:0] ns_i2c_transceiver;
reg                 [2:0] s_i2c_transceiver;

/****************************************************************************
 *                          Finite State Machine(s)                         *
 ****************************************************************************/

always @(posedge clk)
begin
      if (reset == 1'b1)
            s_i2c_transceiver <= I2C_STATE_0_IDLE;
      else
            s_i2c_transceiver <= ns_i2c_transceiver;
end

always @(*)
begin
      // Defaults
      ns_i2c_transceiver = I2C_STATE_0_IDLE;

    case (s_i2c_transceiver)
      I2C_STATE_0_IDLE:
            begin
                  if ((send_start_bit == 1'b1) && (clk_400KHz == 1'b0))
                        ns_i2c_transceiver = I2C_STATE_1_PRE_START;
                  else if (send_start_bit == 1'b1)
                        ns_i2c_transceiver = I2C_STATE_2_START_BIT;
                  else if (send_stop_bit == 1'b1)
                        ns_i2c_transceiver = I2C_STATE_5_STOP_BIT;
				else if (transfer_data == 1'b1)
					ns_i2c_transceiver = I2C_STATE_3_TRANSFER_BYTE;
				else
					ns_i2c_transceiver = I2C_STATE_0_IDLE;
			end
		I2C_STATE_1_PRE_START:
			begin
				if (start_and_stop_en == 1'b1)
					ns_i2c_transceiver = I2C_STATE_2_START_BIT;
				else
					ns_i2c_transceiver = I2C_STATE_1_PRE_START;
			end
		I2C_STATE_2_START_BIT:
			begin
				if (change_output_bit_en == 1'b1)
				begin
					if ((transfer_data == 1'b1) && (I2C_BUS_MODE == 1'b0))
						ns_i2c_transceiver = I2C_STATE_3_TRANSFER_BYTE;
					else
						ns_i2c_transceiver = I2C_STATE_6_COMPLETE;
				end
				else
					ns_i2c_transceiver = I2C_STATE_2_START_BIT;
			end
		I2C_STATE_3_TRANSFER_BYTE:
			begin
				if ((current_bit == 3'h0) && (change_output_bit_en == 1'b1))
				begin
					if ((I2C_BUS_MODE == 1'b0) || (num_bits_to_transfer ==
3'h6))
						ns_i2c_transceiver = I2C_STATE_4_TRANSFER_ACK;
					else
						ns_i2c_transceiver = I2C_STATE_6_COMPLETE;
				end
				else
					ns_i2c_transceiver = I2C_STATE_3_TRANSFER_BYTE;
			end
		I2C_STATE_4_TRANSFER_ACK:
			begin
				if (change_output_bit_en == 1'b1)
					ns_i2c_transceiver = I2C_STATE_6_COMPLETE;
				else
					ns_i2c_transceiver = I2C_STATE_4_TRANSFER_ACK;
			end
		I2C_STATE_5_STOP_BIT:
			begin
				if (start_and_stop_en == 1'b1)
					ns_i2c_transceiver = I2C_STATE_6_COMPLETE;
				else
					ns_i2c_transceiver = I2C_STATE_5_STOP_BIT;
			end
		I2C_STATE_6_COMPLETE:
			begin
				if (transfer_data == 1'b0)
					ns_i2c_transceiver = I2C_STATE_0_IDLE;
				else
					ns_i2c_transceiver = I2C_STATE_6_COMPLETE;
			end
```

```
      default:
            begin
                  ns_i2c_transceiver = I2C_STATE_0_IDLE;
            end
      endcase
end

/*****************************************************************************
 *                              Sequential logic                             *
 *****************************************************************************/

// Output Registers
always @(posedge clk)
begin
      if (reset == 1'b1)
            i2c_scen <= 1'b1;
      else if (change_output_bit_en & (s_i2c_transceiver ==
I2C_STATE_2_START_BIT))
            i2c_scen <= 1'b0;
      else if (s_i2c_transceiver == I2C_STATE_5_STOP_BIT)
            i2c_scen <= 1'b1;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            ack <= 1'b0;
      else if (clear_ack == 1'b1)
            ack <= 1'b0;
      else if (start_and_stop_en & (s_i2c_transceiver ==
I2C_STATE_4_TRANSFER_ACK))
            ack <= i2c_sdata ^ I2C_BUS_MODE;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            data_from_i2c <= 8'h00;
      else if (start_and_stop_en & (s_i2c_transceiver ==
I2C_STATE_3_TRANSFER_BYTE))
            data_from_i2c <= {data_from_i2c[6:0], i2c_sdata};
end


// Internal Registers
always @(posedge clk)
begin
      if (reset == 1'b1)
            current_bit <= 3'h0;
      else if ((s_i2c_transceiver == I2C_STATE_3_TRANSFER_BYTE) &&
                  (change_output_bit_en == 1'b1))
            current_bit <= current_bit - 3'h1;
      else if (s_i2c_transceiver != I2C_STATE_3_TRANSFER_BYTE)
            current_bit <= num_bits_to_transfer;
end

always @(posedge clk)
begin
```

```
      if (reset == 1'b1)
            current_byte <= 8'h00;
      else if ((s_i2c_transceiver == I2C_STATE_0_IDLE) ||
                   (s_i2c_transceiver == I2C_STATE_2_START_BIT))
            current_byte <= data_in;
end

/*****************************************************************************
 *                            Combinational logic                            *
 *****************************************************************************/

assign i2c_sclk         = (I2C_BUS_MODE == 1'b0) ?
                                    clk_400KHz :
                                    ((s_i2c_transceiver ==
I2C_STATE_3_TRANSFER_BYTE) |
                                    (s_i2c_transceiver ==
I2C_STATE_4_TRANSFER_ACK)) ?
                                          clk_400KHz :
                                          1'b0;

assign i2c_sdata  =
      (s_i2c_transceiver == I2C_STATE_2_START_BIT) ? 1'b0 :
      (s_i2c_transceiver == I2C_STATE_5_STOP_BIT) ? 1'b0 :
      ((s_i2c_transceiver == I2C_STATE_4_TRANSFER_ACK) & read_byte) ? 1'b0 :
      ((s_i2c_transceiver == I2C_STATE_3_TRANSFER_BYTE) & ~read_byte) ?
            current_byte[current_bit]
            : 1'bz;

assign enable_clk = ~(s_i2c_transceiver == I2C_STATE_0_IDLE) &&
                  ~(s_i2c_transceiver == I2C_STATE_6_COMPLETE);

assign transfer_complete =
            (s_i2c_transceiver == I2C_STATE_6_COMPLETE) ? 1'b1 : 1'b0;

/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/

endmodule
```

4.1.5. Modulo do Codec de Audio

```
/******************************************************************************
 * License Agreement                                                          *
 *                                                                            *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.     *
 * All rights reserved.                                                       *
 *                                                                            *
 * Any megafunction design, and related net list (encrypted or decrypted),    *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner   *
 *  under Altera's Megafunction Partnership Program may be used only to       *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other  *
 *  use of such megafunction design, net list, support information, device    *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not       *
 *  limited to modification, reverse engineering, de-compiling, or use with   *
 *  any other silicon devices, unless such use is explicitly licensed under   *
 *  a separate agreement with Altera or a megafunction partner.  Title to     *
 *  the intellectual property, including patents, copyrights, trademarks,     *
 *  trade secrets, or maskworks, embodied in any such megafunction design,    *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a    *
 *  megafunction partner, remains with Altera, the megafunction partner, or   *
 *  their respective licensors.  No other licenses, including any licenses    *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice   *
 *  is attached violates this copyright.                                      *
 *                                                                            *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR    *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,   *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL    *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING    *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                              *
 *                                                                            *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                            *
 ******************************************************************************/

/******************************************************************************
 *                                                                            *
 * This module reads and writes data to the Audio chip on Altera's DE1        *
 *  Development and Education Board. The audio chip must be in master mode    *
 *  and the digital format must be left justified.                            *
 *                                                                            *
 ******************************************************************************/

module audio_codec (
// Inputs
CLOCK_50,
reset,

read_s,     write_s,
writedata_left, writedata_right,
AUD_ADCDAT,

// Bidirectionals
AUD_BCLK,
AUD_ADCLRCK,
AUD_DACLRCK,

// Outputs
read_ready, write_ready,
readdata_left, readdata_right,
AUD_DACDAT
);

/*****************************************************************************
 *                           Parameter Declarations                          *
 *****************************************************************************/

parameter AUDIO_DATA_WIDTH    = 24;
parameter BIT_COUNTER_INIT    = 5'd23;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/
// Inputs
input                   CLOCK_50;
input                   reset;

input                         read_s;
input                         write_s;
input [AUDIO_DATA_WIDTH-1:0]  writedata_left;
input [AUDIO_DATA_WIDTH-1:0]  writedata_right;

input                   AUD_ADCDAT;
input                   AUD_BCLK;
input                   AUD_ADCLRCK;
input                   AUD_DACLRCK;

// Outputs
output                  read_ready, write_ready;
output      [AUDIO_DATA_WIDTH-1:0]  readdata_left;
output      [AUDIO_DATA_WIDTH-1:0]  readdata_right;


output                  AUD_DACDAT;

/*****************************************************************************
 *                 Internal wires and registers Declarations                 *
 *****************************************************************************/

// Internal Wires
wire                    bclk_rising_edge;
wire                    bclk_falling_edge;

wire                    adc_lrclk_rising_edge;
wire                    adc_lrclk_falling_edge;

wire        [AUDIO_DATA_WIDTH:1] new_left_channel_audio;
wire        [AUDIO_DATA_WIDTH:1] new_right_channel_audio;
wire        [7:0] left_channel_read_available;
wire        [7:0] right_channel_read_available;
wire                    dac_lrclk_rising_edge;
wire                    dac_lrclk_falling_edge;

wire        [7:0] left_channel_write_space;
wire        [7:0] right_channel_write_space;

// Internal Registers
reg                           done_adc_channel_sync;
reg                           done_dac_channel_sync;

// State Machine Registers


/*****************************************************************************
 *                           Finite State Machine(s)                         *
 *****************************************************************************/


/*****************************************************************************
 *                               Sequential logic                            *
 *****************************************************************************/

always @ (posedge CLOCK_50)
begin
      if (reset == 1'b1)
            done_adc_channel_sync <= 1'b0;
      else if (adc_lrclk_rising_edge == 1'b1)
            done_adc_channel_sync <= 1'b1;
end

always @ (posedge CLOCK_50)
begin
      if (reset == 1'b1)
            done_dac_channel_sync <= 1'b0;
      else if (dac_lrclk_falling_edge == 1'b1)
            done_dac_channel_sync <= 1'b1;
end

/*****************************************************************************
 *                              Combinational logic                          *
 *****************************************************************************/

assign read_ready = (left_channel_read_available != 8'd0) &
(right_channel_read_available != 8'd0);
assign write_ready = (left_channel_write_space != 8'd0) &
(right_channel_write_space != 8'd0);
assign readdata_left = new_left_channel_audio;
assign readdata_right = new_right_channel_audio;

/*****************************************************************************
 *                                Internal Modules                           *
 *****************************************************************************/

Altera_UP_Clock_Edge Bit_Clock_Edges (
      // Inputs
      .clk                (CLOCK_50),
      .reset              (reset),

      .test_clk           (AUD_BCLK),

      // Bidirectionals

      // Outputs
      .rising_edge        (bclk_rising_edge),
      .falling_edge       (bclk_falling_edge)
);

Altera_UP_Clock_Edge ADC_Left_Right_Clock_Edges (
      // Inputs
      .clk                (CLOCK_50),
      .reset              (reset),

      .test_clk           (AUD_ADCLRCK),

      // Bidirectionals

      // Outputs
      .rising_edge        (adc_lrclk_rising_edge),
      .falling_edge       (adc_lrclk_falling_edge)
);

Altera_UP_Clock_Edge DAC_Left_Right_Clock_Edges (
      // Inputs
      .clk                (CLOCK_50),
      .reset              (reset),

      .test_clk           (AUD_DACLRCK),

      // Bidirectionals

      // Outputs
      .rising_edge        (dac_lrclk_rising_edge),
      .falling_edge       (dac_lrclk_falling_edge)
);



Altera_UP_Audio_In_Deserializer Audio_In_Deserializer (
      // Inputs
      .clk                                                (CLOCK_50),
      .reset                                              (reset),

      .bit_clk_rising_edge                    (bclk_rising_edge),
      .bit_clk_falling_edge             (bclk_falling_edge),
      .left_right_clk_rising_edge       (adc_lrclk_rising_edge),
      .left_right_clk_falling_edge      (adc_lrclk_falling_edge),

      .done_channel_sync                      (done_adc_channel_sync),

      .serial_audio_in_data             (AUD_ADCDAT),

      .read_left_audio_data_en          (read_s &
(left_channel_read_available != 8'd0)),
	.read_right_audio_data_en				(read_s &
(right_channel_read_available != 8'd0)),

	// Bidirectionals

	// Outputs
	.left_audio_fifo_read_space		(left_channel_read_available),
	.right_audio_fifo_read_space		(right_channel_read_available),

	.left_channel_data					(new_left_channel_audio),
	.right_channel_data					(new_right_channel_audio)
);
defparam
	Audio_In_Deserializer.AUDIO_DATA_WIDTH = AUDIO_DATA_WIDTH,
	Audio_In_Deserializer.BIT_COUNTER_INIT = BIT_COUNTER_INIT;


Altera_UP_Audio_Out_Serializer Audio_Out_Serializer (
	// Inputs
	.clk								(CLOCK_50),
	.reset								(reset),

	.bit_clk_rising_edge					(bclk_rising_edge),
	.bit_clk_falling_edge				(bclk_falling_edge),
	.left_right_clk_rising_edge		(done_dac_channel_sync &
dac_lrclk_rising_edge),
	.left_right_clk_falling_edge		(done_dac_channel_sync &
dac_lrclk_falling_edge),

	.left_channel_data					(writedata_left),
	.left_channel_data_en				(write_s &
(left_channel_write_space != 8'd0)),

	.right_channel_data					(writedata_right),
	.right_channel_data_en				(write_s &
(right_channel_write_space != 8'd0)),

	// Bidirectionals

	// Outputs
	.left_channel_fifo_write_space		(left_channel_write_space),
	.right_channel_fifo_write_space	(right_channel_write_space),

	.serial_audio_out_data			(AUD_DACDAT)
);
defparam
	Audio_Out_Serializer.AUDIO_DATA_WIDTH = AUDIO_DATA_WIDTH;

endmodule
```

4.1.5.1. Controle de Canal de Audio

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device  *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module finds clock edges of one clock at the frequency of           *
 *  another clock.                                                           *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_Clock_Edge (
	// Inputs
	clk,
	reset,

	test_clk,

	// Bidirectionals
```

```
      // Outputs
      rising_edge,
      falling_edge
);

/*****************************************************************************
 *                            Parameter Declarations                         *
 *****************************************************************************/


/*****************************************************************************
 *                              Port Declarations                            *
 *****************************************************************************/

// Inputs
input                   clk;
input                   reset;

input                   test_clk;

// Bidirectionals

// Outputs
output                        rising_edge;
output                        falling_edge;

/*****************************************************************************
 *                            Constant Declarations                          *
 *****************************************************************************/

/*****************************************************************************
 *                  Internal wires and registers Declarations                *
 *****************************************************************************/

// Internal Wires
wire                    found_edge;

// Internal Registers
reg                           cur_test_clk;
reg                           last_test_clk;

// State Machine Registers

/*****************************************************************************
 *                          Finite State Machine(s)                          *
 *****************************************************************************/


/*****************************************************************************
 *                              Sequential logic                             *
 *****************************************************************************/

always @(posedge clk)
      cur_test_clk      <= test_clk;

always @(posedge clk)
      last_test_clk     <= cur_test_clk;
```

```
/******************************************************************************
 *                             Combinational logic                            *
 ******************************************************************************/

// Output Assignments
assign rising_edge      = found_edge & cur_test_clk;
assign falling_edge     = found_edge & last_test_clk;

// Internal Assignments
assign found_edge = last_test_clk ^ cur_test_clk;

/******************************************************************************
 *                               Internal Modules                             *
 ******************************************************************************/

endmodule
```

4.1.5.2. Deserializador de Audio de entrada

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not     *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module read data from the Audio ADC on the Altera DE1 board.         *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_Audio_In_Deserializer (
	// Inputs
	clk,
	reset,

	bit_clk_rising_edge,
	bit_clk_falling_edge,
	left_right_clk_rising_edge,
	left_right_clk_falling_edge,
      done_channel_sync,

      serial_audio_in_data,

      read_left_audio_data_en,
      read_right_audio_data_en,

      // Bidirectionals

      // Outputs
      left_audio_fifo_read_space,
      right_audio_fifo_read_space,

      left_channel_data,
      right_channel_data
);

/*****************************************************************************
 *                           Parameter Declarations                          *
 *****************************************************************************/

parameter AUDIO_DATA_WIDTH    = 16;
parameter BIT_COUNTER_INIT    = 5'h0F;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/
// Inputs
input                   clk;
input                   reset;

input                   bit_clk_rising_edge;
input                   bit_clk_falling_edge;
input                   left_right_clk_rising_edge;
input                   left_right_clk_falling_edge;

input                   done_channel_sync;

input                   serial_audio_in_data;

input                   read_left_audio_data_en;
input                   read_right_audio_data_en;

// Bidirectionals

// Outputs
output      reg   [7:0] left_audio_fifo_read_space;
output      reg   [7:0] right_audio_fifo_read_space;

output            [AUDIO_DATA_WIDTH:1]    left_channel_data;
output            [AUDIO_DATA_WIDTH:1]    right_channel_data;

/*****************************************************************************
 *                 Internal wires and registers Declarations                 *
 *****************************************************************************/
// Internal Wires
wire                    valid_audio_input;
wire                    left_channel_fifo_is_empty;
wire                    right_channel_fifo_is_empty;

wire                    left_channel_fifo_is_full;
wire                    right_channel_fifo_is_full;

wire        [6:0] left_channel_fifo_used;
wire        [6:0] right_channel_fifo_used;

// Internal Registers
reg               [AUDIO_DATA_WIDTH:1]    data_in_shift_reg;

// State Machine Registers


/*****************************************************************************
 *                         Finite State Machine(s)                           *
 *****************************************************************************/


/*****************************************************************************
 *                             Sequential logic                              *
 *****************************************************************************/

always @(posedge clk)
begin
      if (reset == 1'b1)
            left_audio_fifo_read_space                <= 8'h00;
      else
      begin
            left_audio_fifo_read_space[7]       <= left_channel_fifo_is_full;
            left_audio_fifo_read_space[6:0]           <= left_channel_fifo_used;
      end
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            right_audio_fifo_read_space               <= 8'h00;
      else
      begin
            right_audio_fifo_read_space[7]            <=
right_channel_fifo_is_full;
            right_audio_fifo_read_space[6:0]    <= right_channel_fifo_used;
      end
end



always @(posedge clk)
begin
      if (reset == 1'b1)
            data_in_shift_reg <= {AUDIO_DATA_WIDTH{1'b0}};
      else if (bit_clk_rising_edge & valid_audio_input)
            data_in_shift_reg <=
                  {data_in_shift_reg[(AUDIO_DATA_WIDTH - 1):1],
                   serial_audio_in_data};
```

```
end

/*****************************************************************************
 *                             Combinational logic                           *
 *****************************************************************************/


/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/

Altera_UP_Audio_Bit_Counter Audio_Out_Bit_Counter (
      // Inputs
      .clk                                    (clk),
      .reset                                        (reset),

      .bit_clk_rising_edge                (bit_clk_rising_edge),
      .bit_clk_falling_edge               (bit_clk_falling_edge),
      .left_right_clk_rising_edge         (left_right_clk_rising_edge),
      .left_right_clk_falling_edge  (left_right_clk_falling_edge),

      // Bidirectionals

      // Outputs
      .counting                               (valid_audio_input)
);
defparam
      Audio_Out_Bit_Counter.BIT_COUNTER_INIT    = BIT_COUNTER_INIT;

Altera_UP_SYNC_FIFO Audio_In_Left_Channel_FIFO(
      // Inputs
      .clk                  (clk),
      .reset                      (reset),

      .write_en             (left_right_clk_falling_edge &
~left_channel_fifo_is_full & done_channel_sync),
      .write_data           (data_in_shift_reg),

      .read_en              (read_left_audio_data_en & ~left_channel_fifo_is_empty),

      // Bidirectionals

      // Outputs
      .fifo_is_empty        (left_channel_fifo_is_empty),
      .fifo_is_full         (left_channel_fifo_is_full),
      .words_used           (left_channel_fifo_used),

      .read_data            (left_channel_data)
);
defparam
      Audio_In_Left_Channel_FIFO.DATA_WIDTH     = AUDIO_DATA_WIDTH,
      Audio_In_Left_Channel_FIFO.DATA_DEPTH     = 128,
      Audio_In_Left_Channel_FIFO.ADDR_WIDTH     = 7;

Altera_UP_SYNC_FIFO Audio_In_Right_Channel_FIFO(
      // Inputs
      .clk                  (clk),
      .reset                      (reset),
      .write_en         (left_right_clk_rising_edge &
~right_channel_fifo_is_full & done_channel_sync),
      .write_data       (data_in_shift_reg),

      .read_en          (read_right_audio_data_en &
~right_channel_fifo_is_empty),

      // Bidirectionals

      // Outputs
      .fifo_is_empty    (right_channel_fifo_is_empty),
      .fifo_is_full     (right_channel_fifo_is_full),
      .words_used       (right_channel_fifo_used),

      .read_data        (right_channel_data)
);
defparam
      Audio_In_Right_Channel_FIFO.DATA_WIDTH    = AUDIO_DATA_WIDTH,
      Audio_In_Right_Channel_FIFO.DATA_DEPTH    = 128,
      Audio_In_Right_Channel_FIFO.ADDR_WIDTH    = 7;

endmodule
```

4.1.5.2.1. Fila do Audio Esquerdo e Direito

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other  *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not     *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or  *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR    *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,   *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL    *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING    *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module is a FIFO with same clock for both reads and writes.          *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_SYNC_FIFO (
	// Inputs
	clk,
	reset,

	write_en,
	write_data,

	read_en,
      // Bidirectionals

      // Outputs
      fifo_is_empty,
      fifo_is_full,
      words_used,

      read_data
);

/*****************************************************************************
 *                          Parameter Declarations                           *
 *****************************************************************************/

parameter   DATA_WIDTH  = 32;
parameter   DATA_DEPTH  = 128;
parameter   ADDR_WIDTH  = 7;

/*****************************************************************************
 *                            Port Declarations                              *
 *****************************************************************************/

// Inputs
input                   clk;
input                   reset;

input                   write_en;
input       [DATA_WIDTH:1]    write_data;

input                   read_en;

// Bidirectionals

// Outputs
output                        fifo_is_empty;
output                        fifo_is_full;
output            [ADDR_WIDTH:1]    words_used;

output            [DATA_WIDTH:1]    read_data;

/*****************************************************************************
 *                 Internal wires and registers Declarations                 *
 *****************************************************************************/

// Internal Wires

// Internal Registers

// State Machine Registers

/*****************************************************************************
 *                         Finite State Machine(s)                           *
 *****************************************************************************/


/*****************************************************************************
 *                            Sequential logic                               *
 *****************************************************************************/
```

```
/*****************************************************************************
 *                              Combinational logic                          *
 *****************************************************************************/


/*****************************************************************************
 *                                Internal Modules                           *
 *****************************************************************************/


scfifo      Sync_FIFO (
      // Inputs
      .clock                  (clk),
      .sclr             (reset),

      .data             (write_data),
      .wrreq                  (write_en),

      .rdreq                  (read_en),

      // Bidirectionals

      // Outputs
      .empty                  (fifo_is_empty),
      .full             (fifo_is_full),
      .usedw                  (words_used),

      .q                      (read_data)

      // Unused
      // synopsys translate_off
      ,
      .aclr             (),
      .almost_empty     (),
      .almost_full      ()
      // synopsys translate_on
);
defparam
      Sync_FIFO.add_ram_output_register   = "OFF",
      Sync_FIFO.intended_device_family    = "Cyclone II",
      Sync_FIFO.lpm_numwords              = DATA_DEPTH,
      Sync_FIFO.lpm_showahead             = "ON",
      Sync_FIFO.lpm_type                        = "scfifo",
      Sync_FIFO.lpm_width                       = DATA_WIDTH,
      Sync_FIFO.lpm_widthu                = ADDR_WIDTH,
      Sync_FIFO.overflow_checking         = "OFF",
      Sync_FIFO.underflow_checking  = "OFF",
      Sync_FIFO.use_eab                   = "ON";

endmodule
```

4.1.5.2.2. Contador de bit de saídas

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner   *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other  *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 *   This module counts which bits for serial audio transfers. The module    *
 * assume that the data format is I2S, as it is described in the audio       *
 * chip's datasheet.                                                         *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_Audio_Bit_Counter (
	// Inputs
	clk,
	reset,

	bit_clk_rising_edge,
	bit_clk_falling_edge,
	left_right_clk_rising_edge,
      left_right_clk_falling_edge,

      // Bidirectionals

      // Outputs
      counting
);

/*****************************************************************************
 *                             Parameter Declarations                        *
 *****************************************************************************/

parameter BIT_COUNTER_INIT    = 5'h0F;

/*****************************************************************************
 *                               Port Declarations                           *
 *****************************************************************************/

// Inputs
input                   clk;
input                   reset;

input                   bit_clk_rising_edge;
input                   bit_clk_falling_edge;
input                   left_right_clk_rising_edge;
input                   left_right_clk_falling_edge;

// Bidirectionals

// Outputs
output      reg               counting;

/*****************************************************************************
 *                             Constant Declarations                         *
 *****************************************************************************/


/*****************************************************************************
 *                    Internal wires and registers Declarations              *
 *****************************************************************************/

// Internal Wires
wire                    reset_bit_counter;

// Internal Registers
reg               [4:0] bit_counter;

// State Machine Registers


/*****************************************************************************
 *                           Finite State Machine(s)                         *
 *****************************************************************************/


/*****************************************************************************
 *                               Sequential logic                            *
 *****************************************************************************/
```

```
always @(posedge clk)
begin
      if (reset == 1'b1)
            bit_counter <= 5'h00;
      else if (reset_bit_counter == 1'b1)
            bit_counter <= BIT_COUNTER_INIT;
      else if ((bit_clk_falling_edge == 1'b1) && (bit_counter != 5'h00))
            bit_counter <= bit_counter - 5'h01;
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            counting <= 1'b0;
      else if (reset_bit_counter == 1'b1)
            counting <= 1'b1;
      else if ((bit_clk_falling_edge == 1'b1) && (bit_counter == 5'h00))
            counting <= 1'b0;
end

/*****************************************************************************
 *                             Combinational logic                           *
 *****************************************************************************/

assign reset_bit_counter = left_right_clk_rising_edge |
                                          left_right_clk_falling_edge;

/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/

endmodule
```

4.1.5.3. Modulo Serializador de saída

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS  *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State  *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 * This module writes data to the Audio DAC on the Altera DE1 board.         *
 *                                                                           *
 *****************************************************************************/

module Altera_UP_Audio_Out_Serializer (
	// Inputs
	clk,
	reset,

	bit_clk_rising_edge,
	bit_clk_falling_edge,
	left_right_clk_rising_edge,
	left_right_clk_falling_edge,
      left_channel_data,
      left_channel_data_en,

      right_channel_data,
      right_channel_data_en,

      // Bidirectionals

      // Outputs
      left_channel_fifo_write_space,
      right_channel_fifo_write_space,

      serial_audio_out_data
);

/*****************************************************************************
 *                             Parameter Declarations                        *
 *****************************************************************************/

parameter AUDIO_DATA_WIDTH    = 16;

/*****************************************************************************
 *                             Port Declarations                             *
 *****************************************************************************/
// Inputs
input                        clk;
input                        reset;

input                        bit_clk_rising_edge;
input                        bit_clk_falling_edge;
input                        left_right_clk_rising_edge;
input                        left_right_clk_falling_edge;

input        [AUDIO_DATA_WIDTH:1]          left_channel_data;
input                        left_channel_data_en;

input        [AUDIO_DATA_WIDTH:1]          right_channel_data;
input                        right_channel_data_en;

// Bidirectionals

// Outputs
output       reg    [7:0] left_channel_fifo_write_space;
output       reg    [7:0] right_channel_fifo_write_space;

output       reg                serial_audio_out_data;


/*****************************************************************************
 *                   Internal wires and registers Declarations               *
 *****************************************************************************/

// Internal Wires
wire                         read_left_channel;
wire                         read_right_channel;

wire                         left_channel_fifo_is_empty;
wire                         right_channel_fifo_is_empty;
wire                        left_channel_fifo_is_full;
wire                        right_channel_fifo_is_full;

wire            [6:0] left_channel_fifo_used;
wire            [6:0] right_channel_fifo_used;

wire            [AUDIO_DATA_WIDTH:1]          left_channel_from_fifo;
wire            [AUDIO_DATA_WIDTH:1]          right_channel_from_fifo;

// Internal Registers
reg                                 left_channel_was_read;
reg                   [AUDIO_DATA_WIDTH:1]    data_out_shift_reg;

// State Machine Registers

/*****************************************************************************
 *                           Finite State Machine(s)                         *
 *****************************************************************************/


/*****************************************************************************
 *                              Sequential logic                             *
 *****************************************************************************/

always @(posedge clk)
begin
      if (reset == 1'b1)
            left_channel_fifo_write_space <= 8'h00;
      else
            left_channel_fifo_write_space <= 8'h80 -
{left_channel_fifo_is_full,left_channel_fifo_used};
end

always @(posedge clk)
begin
      if (reset == 1'b1)
            right_channel_fifo_write_space <= 8'h00;
      else
            right_channel_fifo_write_space <= 8'h80 -
{right_channel_fifo_is_full,right_channel_fifo_used};
end


always @(posedge clk)
begin
      if (reset == 1'b1)
            serial_audio_out_data <= 1'b0;
      else
            serial_audio_out_data <= data_out_shift_reg[AUDIO_DATA_WIDTH];
end


always @(posedge clk)
begin
      if (reset == 1'b1)
            left_channel_was_read <= 1'b0;
      else if (read_left_channel)
```

```
            left_channel_was_read <=1'b1;
      else if (read_right_channel)
            left_channel_was_read <=1'b0;
end


always @(posedge clk)
begin
      if (reset == 1'b1)
            data_out_shift_reg      <= {AUDIO_DATA_WIDTH{1'b0}};
      else if (read_left_channel)
            data_out_shift_reg      <= left_channel_from_fifo;
      else if (read_right_channel)
            data_out_shift_reg      <= right_channel_from_fifo;
      else if (left_right_clk_rising_edge | left_right_clk_falling_edge)
            data_out_shift_reg      <= {AUDIO_DATA_WIDTH{1'b0}};
      else if (bit_clk_falling_edge)
            data_out_shift_reg      <=
                  {data_out_shift_reg[(AUDIO_DATA_WIDTH - 1):1], 1'b0};
end

/*****************************************************************************
 *                             Combinational logic                           *
 *****************************************************************************/

assign read_left_channel      = left_right_clk_rising_edge &
                                 ~left_channel_fifo_is_empty &
                                 ~right_channel_fifo_is_empty;
assign read_right_channel     = left_right_clk_falling_edge &
                                left_channel_was_read;

/*****************************************************************************
 *                              Internal Modules                             *
 *****************************************************************************/

Altera_UP_SYNC_FIFO Audio_Out_Left_Channel_FIFO(
      // Inputs
      .clk                (clk),
      .reset                    (reset),

      .write_en           (left_channel_data_en & ~left_channel_fifo_is_full),
      .write_data         (left_channel_data),

      .read_en            (read_left_channel),

      // Bidirectionals

      // Outputs
      .fifo_is_empty      (left_channel_fifo_is_empty),
      .fifo_is_full       (left_channel_fifo_is_full),
      .words_used         (left_channel_fifo_used),

      .read_data          (left_channel_from_fifo)
);
defparam
      Audio_Out_Left_Channel_FIFO.DATA_WIDTH    = AUDIO_DATA_WIDTH,
      Audio_Out_Left_Channel_FIFO.DATA_DEPTH    = 128,
      Audio_Out_Left_Channel_FIFO.ADDR_WIDTH    = 7;
```

```
Altera_UP_SYNC_FIFO Audio_Out_Right_Channel_FIFO(
	// Inputs
	.clk			(clk),
	.reset				(reset),

	.write_en		(right_channel_data_en & ~right_channel_fifo_is_full),
	.write_data		(right_channel_data),

	.read_en		(read_right_channel),

	// Bidirectionals

	// Outputs
	.fifo_is_empty	(right_channel_fifo_is_empty),
	.fifo_is_full	(right_channel_fifo_is_full),
	.words_used		(right_channel_fifo_used),

	.read_data		(right_channel_from_fifo)
);
defparam
	Audio_Out_Right_Channel_FIFO.DATA_WIDTH	= AUDIO_DATA_WIDTH,
	Audio_Out_Right_Channel_FIFO.DATA_DEPTH	= 128,
	Audio_Out_Right_Channel_FIFO.ADDR_WIDTH	= 7;

endmodule
```

4.1.5.4. Gerador de clocks

```
/*****************************************************************************
 * License Agreement                                                         *
 *                                                                           *
 * Copyright (c) 1991-2009 Altera Corporation, San Jose, California, USA.    *
 * All rights reserved.                                                      *
 *                                                                           *
 * Any megafunction design, and related net list (encrypted or decrypted),   *
 *  support information, device programming or simulation file, and any other *
 *  associated documentation or information provided by Altera or a partner  *
 *  under Altera's Megafunction Partnership Program may be used only to      *
 *  program PLD devices (but not masked PLD devices) from Altera.  Any other *
 *  use of such megafunction design, net list, support information, device   *
 *  programming or simulation file, or any other related documentation or    *
 *  information is prohibited for any other purpose, including, but not      *
 *  limited to modification, reverse engineering, de-compiling, or use with  *
 *  any other silicon devices, unless such use is explicitly licensed under  *
 *  a separate agreement with Altera or a megafunction partner.  Title to    *
 *  the intellectual property, including patents, copyrights, trademarks,    *
 *  trade secrets, or maskworks, embodied in any such megafunction design,   *
 *  net list, support information, device programming or simulation file, or *
 *  any other related documentation or information provided by Altera or a   *
 *  megafunction partner, remains with Altera, the megafunction partner, or  *
 *  their respective licensors.  No other licenses, including any licenses   *
 *  needed under any third party's intellectual property, are provided herein.*
 *  Copying or modifying any file, or portion thereof, to which this notice  *
 *  is attached violates this copyright.                                     *
 *                                                                           *
 * THIS FILE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR   *
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  *
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL   *
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER *
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING   *
 * FROM, OUT OF OR IN CONNECTION WITH THIS FILE OR THE USE OR OTHER DEALINGS *
 * IN THIS FILE.                                                             *
 *                                                                           *
 * This agreement shall be governed in all respects by the laws of the State *
 *  of California and by the laws of the United States of America.           *
 *                                                                           *
 *****************************************************************************/

/*****************************************************************************
 *                                                                           *
 *   This module generates the clocks needed for the I/O devices on          *
 * Altera's DE1 and DE1 Boards.                                              *
 *                                                                           *
 *****************************************************************************/

module clock_generator (
      // inputs
      CLOCK_27,
      reset,

      // outputs
      AUD_XCK
);
```

```
/*****************************************************************************
 *                             Parameter Declarations                        *
 *****************************************************************************/

parameter AUD_CLK_MULT  = 14;
parameter AUD_CLK_DIV   = 31;

/*****************************************************************************
 *                               Port Declarations                           *
 *****************************************************************************/
// Inputs
input                   [1:1] CLOCK_27;
input                   reset;

// Outputs
output                        AUD_XCK;

/*****************************************************************************
 *                             Constant Declarations                         *
 *****************************************************************************/


/*****************************************************************************
 *                  Internal wires and registers Declarations                *
 *****************************************************************************/
// Internal Wires
wire                    audio_clk_locked;

// Internal Registers

// State Machine Registers

/*****************************************************************************
 *                           Finite State Machine(s)                         *
 *****************************************************************************/


/*****************************************************************************
 *                               Sequential logic                            *
 *****************************************************************************/


/*****************************************************************************
 *                             Combinational logic                           *
 *****************************************************************************/


/*****************************************************************************
 *                                Internal Modules                           *
 *****************************************************************************/

altpll DE_Clock_Generator_Audio (
      .inclk            ({1'b0, CLOCK_27[1]}),
      .clk        (AUD_XCK),
      .locked           (audio_clk_locked),
      .activeclock      (),
      .areset           (reset),
      .clkbad           (),
      .clkena              ({6{1'b1}}),
      .clkloss             (),
      .clkswitch           (1'b0),
      .enable0             (),
      .enable1             (),
      .extclk              (),
      .extclkena           ({4{1'b1}}),
      .fbin                (1'b1),
      .pfdena              (1'b1),
      .pllena              (1'b1),
      .scanaclr            (1'b0),
      .scanclk             (1'b0),
      .scandata            (1'b0),
      .scandataout         (),
      .scandone            (),
      .scanread            (1'b0),
      .scanwrite           (1'b0),
      .sclkout0            (),
      .sclkout1            ()
);
defparam
      DE_Clock_Generator_Audio.clk0_divide_by                 = AUD_CLK_DIV,
      DE_Clock_Generator_Audio.clk0_duty_cycle                = 50,
      DE_Clock_Generator_Audio.clk0_multiply_by               = AUD_CLK_MULT,
      DE_Clock_Generator_Audio.clk0_phase_shift               = "0",
      DE_Clock_Generator_Audio.compensate_clock               = "CLK0",
      DE_Clock_Generator_Audio.gate_lock_signal               = "NO",
      DE_Clock_Generator_Audio.inclk0_input_frequency         = 37037,
      DE_Clock_Generator_Audio.intended_device_family         = "Cyclone II",
      DE_Clock_Generator_Audio.invalid_lock_multiplier        = 5,
      DE_Clock_Generator_Audio.lpm_type                       = "altpll",
      DE_Clock_Generator_Audio.operation_mode                 = "NORMAL",
      DE_Clock_Generator_Audio.pll_type                       = "FAST",
      DE_Clock_Generator_Audio.port_activeclock               = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_areset                    = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkbad0                   = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkbad1                   = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkloss                         =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkswitch                       =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_fbin                            =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_inclk0                          = "PORT_USED",
      DE_Clock_Generator_Audio.port_inclk1                          =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_locked                          = "PORT_USED",
      DE_Clock_Generator_Audio.port_pfdena                          =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_pllena                          =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scanaclr                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scanclk                         =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scandata                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scandataout               = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scandone                    =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scanread                    =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_scanwrite                   =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clk0                        = "PORT_USED",
      DE_Clock_Generator_Audio.port_clk1                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clk2                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clk3                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clk4                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clk5                        =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena0                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena1                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena2                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena3                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena4                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_clkena5                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_enable0                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_enable1                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclk0                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclk1                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclk2                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclk3                     =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclkena0            = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclkena1            = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclkena2            = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_extclkena3            = "PORT_UNUSED",
      DE_Clock_Generator_Audio.port_sclkout0                    =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.port_sclkout1                    =
"PORT_UNUSED",
      DE_Clock_Generator_Audio.valid_lock_multiplier      = 1;

endmodule
```

4.2. Aplicativo Android

4.2.1. Views

4.2.1.1. Main Activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:weightSum="2"
        android:id="@+id/linearLayout1">
        <ImageButton
            android:layout_gravity="center"
            android:adjustViewBounds="true"
            android:scaleType="fitStart"
            android:layout_height="200dp"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="5dp"
            android:background="@drawable/bluetooth"
            android:padding="20dp"
            android:id="@+id/btBlutooth" />
        <ImageButton
            android:layout_gravity="center"
            android:adjustViewBounds="true"
            android:scaleType="fitStart"
            android:layout_height="200dp"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="5dp"
            android:background="@drawable/Equalizer"
            android:padding="20dp"
            android:id="@+id/btEqualizer" />
    </LinearLayout>
</LinearLayout>
```

4.2.1.2. Controlador Bluetooth

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Switch
        android:text="Bluetooth Controller [off/on]"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="15dp"
        android:id="@+id/swBluetoothController" />
    <Switch
        android:text="Visible"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="15dp"
        android:id="@+id/swBlutoothDicovery" />
    <Button
        android:id="@+id/showName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Show Bluetooth Name" />
    <Button
        android:id="@+id/pairD"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Show Paired Devices" />
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Rename Device Blutooth Name"
        android:layout_marginBottom="5dp"
        android:ems="10" />
    <Button
        android:id="@+id/nameBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Change Bluetooth Name" />
    <ListView
        android:id="@+id/list_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

4.2.1.3. Equalizador

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="25px"
    android:minHeight="25px">
    <Switch
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Connect"
        android:minWidth="25px"
        android:minHeight="25px"
        android:id="@+id/switchConnect" />
    <TextView
        android:text="Mode"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <RadioGroup
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/radioGroup1">
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="Echo"
            android:id="@+id/rbtEcho" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reverberation"
            android:id="@+id/rbtReverberation" />
    </RadioGroup>
    <TextView
        android:text="Distance"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="15dp"
        android:id="@+id/txDistance" />
    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="7"
        android:id="@+id/sbTimeController" />
    <TextView
        android:text="Atenuation"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="15dp"
        android:id="@+id/txAtenuation" />
    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="3"
        android:id="@+id/sbIntensityController" />
</LinearLayout>
```

4.2.2. ViewModel

4.2.2.1. Main Activity

```
using Android.App;
using Android.Widget;
using Android.OS;
using Android.Content;
using Android.Runtime;
using Android.Views;

using System;
using Android.Content.PM;

namespace Friendship404
{

    [Activity(ConfigurationChanges = ConfigChanges.Orientation,
ScreenOrientation = ScreenOrientation.Portrait, Label = "Action Selection",
MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
        ImageButton btBluetooth;
        ImageButton btEqualizer;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // Set our view from the "main" layout resource
            SetContentView (Resource.Layout.Main);

            btBluetooth = FindViewById<ImageButton>(Resource.Id.btBlutooth);
            btEqualizer = FindViewById<ImageButton>(Resource.Id.btEqualizer);

            btBluetooth.Click += BtBluetooth_Click;
            btEqualizer.Click += BtEqualizer_Click;
        }

        private void BtEqualizer_Click(object sender, EventArgs e)
        {
            Intent intent = new Intent(this, typeof(EqualizerController));
            this.StartActivity(intent);
            this.OverridePendingTransition(Android.Resource.Animation.FadeIn,
Android.Resource.Animation.FadeOut);
        }

        private void BtBluetooth_Click(object sender, EventArgs e)
        {
            Intent intent = new Intent(this, typeof(BluetoothController));
            this.StartActivity(intent);
            this.OverridePendingTransition(Android.Resource.Animation.FadeIn,
Android.Resource.Animation.FadeOut);
        }
    }
}
```

4.2.2.2. Bluetooth Controller

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Bluetooth;
using Android.Content.PM;

namespace Friendship404
{
    [Activity(ConfigurationChanges = ConfigChanges.Orientation,
ScreenOrientation = ScreenOrientation.Portrait, Label = "BluetoothController",
Icon = "@drawable/Bluetooth")]
    public class BluetoothController : Activity,
Android.Views.View.IOnClickListener
    {
        Button  listBtn, change_Bluetooth_Name, display_Name;
        //Button onBtn, offBtn,visibleBtn;
        BluetoothAdapter blue;              // Bluetooth adapter class variable

        ListView list_Of_Devices;           // list view for paired devices
        EditText bluetoothName;            // user bluetooth name edit text

        Switch swBluetoothController;
        Switch swBluetoothDiscovery;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.BluetoothControllers);

            // Create your application here
            blue = BluetoothAdapter.DefaultAdapter;
            initialize();
        }

        private void initialize()
        {
            //onBtn = (Button)FindViewById(Resource.Id.onB);
            //offBtn = (Button)FindViewById(Resource.Id.offB);
            //visibleBtn = (Button)FindViewById(Resource.Id.visibleD);
            listBtn = (Button)FindViewById(Resource.Id.pairD);
            bluetoothName = (EditText)FindViewById(Resource.Id.name);
            change_Bluetooth_Name = (Button)FindViewById(Resource.Id.nameBtn);
            display_Name = (Button)FindViewById(Resource.Id.showName);
            list_Of_Devices = (ListView)FindViewById(Resource.Id.list_devices);

            //switch
            swBluetoothController =
(Switch)FindViewById(Resource.Id.swBluetoothController);
            swBluetoothDiscovery =
(Switch)FindViewById(Resource.Id.swBlutoothDicovery);

            if (blue.IsEnabled)
            {
                swBluetoothController.Checked = true;
                if (blue.IsDiscovering == true)
                {
                    swBluetoothDiscovery.Checked = true;
                }
                else
                {
                    swBluetoothDiscovery.Checked = false;
                }
            }
            else
            {
                swBluetoothController.Checked = false;
                swBluetoothDiscovery.Checked = false;
            }

            //onBtn.SetOnClickListener(this);
            //offBtn.SetOnClickListener(this);
            //visibleBtn.SetOnClickListener(this);
            listBtn.SetOnClickListener(this);
            change_Bluetooth_Name.SetOnClickListener(this);
            display_Name.SetOnClickListener(this);
            //switch
            swBluetoothController.SetOnClickListener(this);
        }

        public void OnClick(View v)
        {
            switch (v.Id)
            {
                case Resource.Id.swBluetoothController:
                    if (blue.IsEnabled == false)
                    {
                        //Intent onBlue = new
Intent(BluetoothAdapter.ActionRequestEnable);
                        //StartActivityForResult(onBlue, 0);
                        blue.Enable();

                        Toast.MakeText(this, "Bluetooth Enabled",
ToastLength.Short).Show();

                    }
                    else
                    {
                        try
                        {
                            blue.Disable();
                        }
                        catch (Exception e)
                        {
                            Toast.MakeText(this, e.Message, 
ToastLength.Short).Show();
                            throw;
                        }

                        Toast.MakeText(this, "Bluetooth Disabled", 
ToastLength.Short).Show();

                    }

                    break;

                case Resource.Id.swBlutoothDicovery:
                    if (blue.IsDiscovering == false)
                    {
                        //Intent discovery = new 
Intent(BluetoothAdapter.ActionRequestDiscoverable);
                        //StartActivityForResult(discovery, 0);
                        blue.StartDiscovery();
                        Toast.MakeText(this, "Is Visible", 
ToastLength.Short).Show();
                        //swBluetoothDiscovery.Checked = true;
                    }
                    else
                    {
                        blue.CancelDiscovery();
                        Toast.MakeText(this, "Not Visible", 
ToastLength.Short).Show();
                        //swBluetoothDiscovery.Checked = false;
                    }

                    break;


                case Resource.Id.pairD:

                    ArrayList list = new ArrayList();
                    foreach (BluetoothDevice bt in blue.BondedDevices)
                    {
                        list.Add(bt.Name);
                    }

                    ArrayAdapter adapter = new ArrayAdapter(this, 
Android.Resource.Layout.SimpleListItem1, list);
                    list_Of_Devices.SetAdapter(adapter);
                    break;

                case Resource.Id.nameBtn:
                    if (!bluetoothName.Text.ToString().Equals(""))
                    {
                        String n = bluetoothName.Text.ToString();
                        blue.SetName(n);
                    }
                    else
                    {
                        Toast.MakeText(this, "Please enter name", 
ToastLength.Short).Show();
                    }
                    break;

                case Resource.Id.showName:
                    Toast.MakeText(this, blue.Name.ToString(),
ToastLength.Short).Show();
                    break;
            }
        }
    }
}
```

4.2.2.3. Equalizer Controller

```
using System;
using System.IO;
using System.Threading.Tasks;

using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Android.Bluetooth;
using Android.Content.PM;

using Java.Util;

namespace Friendship404
{
    [Activity(ConfigurationChanges = ConfigChanges.Orientation,
ScreenOrientation = ScreenOrientation.Portrait,Label =
"EqualizerController",Icon = "@drawable/Equalizer")]
    public class EqualizerController : Activity
    {

        //View
        RadioButton Echo, Reverberation;
        Switch BluetoothConnection;
        SeekBar SBTimeBar;
        SeekBar SBIntensive;
        TextView txAtenuation;
        TextView txDistance;
        //ToggleButton tgConnect;
        //TextView Result;

        private string outsideData = "";
        private Java.Lang.String dataToSend;
        private BluetoothAdapter mBluetoothAdapter = null;
        private BluetoothSocket btSocket = null;
        private Stream outStream = null;
        private static string address = "00:21:13:02:32:0E";
        private static UUID MY_UUID = UUID.FromString("00001101-0000-1000-8000-
00805F9B34FB");
        private Stream inStream = null;
        string resultText = "";

        int soundMode, soundAtenuation, soundDistance;
        char soundState;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Equalizer);

            // Create your application here
            Instance();
            CheckBt();
        }
        // Conversion tools
        private Java.Lang.String prepareData(char data)
        {
            return new Java.Lang.String(Convert.ToString(data));
        }

        // 1. Initialize the variables
        private void Instance()
        {
            Echo = (RadioButton)FindViewById(Resource.Id.rbtEcho);
            Reverberation =
(RadioButton)FindViewById(Resource.Id.rbtReverberation);
            BluetoothConnection =
(Switch)FindViewById(Resource.Id.switchConnect);
            SBTimeBar = (SeekBar)FindViewById(Resource.Id.sbTimeController);
            SBIntensive =
(SeekBar)FindViewById(Resource.Id.sbIntensityController);
            txDistance = (TextView)FindViewById(Resource.Id.txDistance);
            txAtenuation = (TextView)FindViewById(Resource.Id.txAtenuation);

            BluetoothConnection.Checked = false;

            Echo.CheckedChange += Echo_CheckedChange;
            Reverberation.CheckedChange += Reverberation_CheckedChange;
            BluetoothConnection.CheckedChange +=
BluetoothConnection_CheckedChange;

            SBTimeBar.ProgressChanged += SBTimeBar_ProgressChanged;
            SBIntensive.ProgressChanged += SBIntensive_ProgressChanged;

        }

        //2. Connect/Disconnect
        private void BluetoothConnection_CheckedChange(object sender,
CompoundButton.CheckedChangeEventArgs e)
        {
            if (BluetoothConnection.Checked)
            {
                Connect();
                soundMode = 100;
                soundAtenuation = 10;
                soundDistance = 1;
                soundState = (char)(soundMode + soundAtenuation +
soundDistance);
                dataToSend = prepareData(soundState);
                try
                {
                    writeData(dataToSend);
                }
                catch (Exception ex)
                {

                    Toast.MakeText(this, "Couldn't update the FPGA: " +
ex.Message, ToastLength.Long).Show();
                }

            }
            else
            {
                try
                {
                    btSocket.Close();
                }
                catch (Exception ex)
                {
                    Toast.MakeText(this, "Connect error: " + ex.Message,
ToastLength.Long).Show();
                }

            }

        }

        // 2.1.  Must connect before stream data
        public void Connect()
        {
            //vincula dispositivo
            BluetoothDevice device = mBluetoothAdapter.GetRemoteDevice(address);
            Toast.MakeText(
                      this,
                      "Conexão em Andamento: " + device,
                      ToastLength.Short
                      ).Show();
            mBluetoothAdapter.CancelDiscovery();

            //Inicia socket de conexão
            try
            {
                btSocket = device.CreateRfcommSocketToServiceRecord(MY_UUID);
                btSocket.Connect();
                Toast.MakeText(
                      this,
                      "Conexão estabelecida",
                      ToastLength.Short
                      ).Show();

            }
            catch (Exception e)
            {
                Toast.MakeText(
                      this,
                      "Conexão não estabilecida \nERRO: " + e.Message,
                      ToastLength.Short
                      ).Show();

                //Caso conexão não seja possível estabelecer fecha conexão
                try
                {
                    btSocket.Close();
                }
                catch (Exception)
                {
                    Toast.MakeText(
                      this,
                      "Não é possível conectar",
                      ToastLength.Short
                    ).Show();

                }

                Toast.MakeText(
                      this,
                      "Socket Encerrado",
                      ToastLength.Short
                      ).Show();
            }

        }

        // 3A. Event toogle, this will open conections send and receive
        private void Echo_CheckedChange(object sender,
CompoundButton.CheckedChangeEventArgs e)
        {
            if (e.IsChecked)
            {
                SBTimeBar.Enabled = true;
                txAtenuation.Enabled = true;
                SBTimeBar.Progress = 0;
                SBIntensive.Enabled = true;
                txDistance.Enabled = true;
                SBIntensive.Progress = 0;

                soundMode = 100;
                soundAtenuation = 10;
                soundDistance = 1;
                soundState = (char)(soundMode + soundAtenuation +
soundDistance);
                dataToSend = prepareData(soundState);
                writeData(dataToSend);

            }

        }

        // 3B. Event toogle, this will open conections send and receive
        private void Reverberation_CheckedChange(object sender,
CompoundButton.CheckedChangeEventArgs e)
        {
            if (e.IsChecked)
            {
                SBTimeBar.Enabled = false;
                txDistance.Enabled = false;
                SBIntensive.Enabled = false;
                txAtenuation.Enabled = false;


                soundState = (char)(100);
                dataToSend = prepareData(soundState);
                writeData(dataToSend);

            }

        }
        private void SBTimeBar_ProgressChanged(object sender,
SeekBar.ProgressChangedEventArgs e)
        {
            soundDistance = e.Progress + 1;
            soundState = (char)(soundMode + soundAtenuation + soundDistance);
            dataToSend = prepareData(soundState);
            writeData(dataToSend);

        }

        private void SBIntensive_ProgressChanged(object sender,
SeekBar.ProgressChangedEventArgs e)
        {
            soundAtenuation = (e.Progress + 1) * 10;
            soundState = (char)(soundMode + soundAtenuation + soundDistance);
            dataToSend = prepareData(soundState);
            writeData(dataToSend);
        }

        // 3Z.1 Recebe Dados
        public void beginListenForData()
        {
            //Inicia socket de entrada
            try
            {
                inStream = btSocket.InputStream;
            }
            catch (IOException ex)
            {
                Toast.MakeText(
                     this,
                     "Dados de entrada\nErro: " + ex.Message,
                     ToastLength.Short
                     ).Show();
            }

            //Inicia a Thread de Recepção
            Task.Factory.StartNew(() => {
                byte[] buffer = new byte[1024];
                int bytes;
                while (true)
                {
                    //Recebe os dados em Bytes
                    try
                    {
                        bytes = inStream.Read(buffer, 0, buffer.Length);
                        if (bytes > 0)
                        {
                            RunOnUiThread(() => {
                                string valor =
System.Text.Encoding.ASCII.GetString(buffer);
                                outsideData = outsideData + "\n" + valor;
                            });
                        }
                    }

                    //Caso não haja mais dados para se receber encerra o evento
de recepção
```

```
                catch (Java.IO.IOException)
                {
                    RunOnUiThread(() => {
                        outsideData = string.Empty;
                    });
                    break;
                }
            }
        });
    }

    //3X.1. Envia Dados
    private void writeData(Java.Lang.String data)
    {
        //inicia socket de envio
        try
        {
            outStream = btSocket.OutputStream;
        }
        catch (Exception e)
        {
            Toast.MakeText(
                 this,
                 "Socket de saida\nErro: " + e.Message,
                 ToastLength.Short
                 ).Show();
        }

        Java.Lang.String message = data;

        byte[] msgBuffer = message.GetBytes();

        try
        {
            outStream.Write(msgBuffer, 0, msgBuffer.Length);
        }
        catch (Exception e)
        {
            Toast.MakeText(
                 this,
                 "Envio de dados\nErro: " + e.Message,
                 ToastLength.Short
                 ).Show();
        }
    }

    // 4. Check Bluetooth Status
    private void CheckBt()
    {
        mBluetoothAdapter = BluetoothAdapter.DefaultAdapter;

        if (!mBluetoothAdapter.IsEnabled)
        {
            Toast.MakeText(this, "Bluetooth is desactivated",
                ToastLength.Short).Show();
        }

        if (mBluetoothAdapter == null)
```

```
            {
                Toast.MakeText(this,
                    "Device not compatible with Bluetooth Adapter",
ToastLength.Short)
                    .Show();
            }
        }

    }
}
```

4.3. Configuração do Arduino no modo AT Serial

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  Serial.begin(9600);
  pinMode(9,OUTPUT); digitalWrite(9,HIGH);
  Serial.println("Enter AT commands:");
  mySerial.begin(38400);
}

void loop(){
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());
}
```

4.4. Observações

O código está sujeito a melhorias futuras link para verificação com o desenvolvedor https://github.com/ctrigger