```
/*
   Ben Davis
   2/23/24
   EE 371
   Lab 5 Task 1 and 2

   This module implements the audio_codec interface to the
   Labsland FPGA board. I did not create this document, I
   only edited parts of it and it is highlighted in the
   comments under the text "Task 2" and "Task 1."
*/

module part1 (CLOCK_50, CLOCK2_50, KEY, SW, FPGA_I2C_SCLK, FPGA_I2C_SDAT, AUD_XCK,
              AUD_DACLRCK, AUD_ADCLRCK, AUD_BCLK, AUD_ADCDAT, AUD_DACDAT);

   input CLOCK_50, CLOCK2_50;
   input [0:0] KEY;
   input [9:9] SW;
   // I2C Audio/Video config interface
   output FPGA_I2C_SCLK;
   inout FPGA_I2C_SDAT;
   // Audio CODEC
   output AUD_XCK;
   input AUD_DACLRCK, AUD_ADCLRCK, AUD_BCLK;
   input AUD_ADCDAT;
   output AUD_DACDAT;

   // Local wires.
   wire read_ready, write_ready, read, write;
   wire [23:0] readdata_left, readdata_right;
   wire [23:0] writedata_left, writedata_right;
   wire reset = ~KEY[0];

   /////////////////////////////////
   // Your code goes here
   /////////////////////////////////

   //                                      START OF TASK 2

   wire [23:0] q; //intermediate value for the rom content
   reg [15:0] addr; //interm value for the rom address

   //rom instantiation
   rom_B4 rn (.address(addr), .clock(CLOCK_50), .q(q));

   //if the sw9 is on, then increment rom address
   //only updates when the read_ready signal is true
   always @(posedge read_ready) begin
      if(SW[9]) begin
         addr <= addr + 1;
      end else begin
         addr <= 16'b0;
      end
   end

   //
   //if sw9 is on, writedata is from the rom
   //otherwise it is from the readdata of the mp3
   assign writedata_left = (SW[9]) ? q : readdata_left;
   assign writedata_right = (SW[9]) ? q : readdata_right;
   assign read = read_ready;
   assign write = write_ready;
   //                                      END OF TASK 2


   /*                                   TASK 1
   assign writedata_left = readdata_left;
   assign writedata_right = readdata_right;
   assign read = read_ready;
   assign write = write_ready;
   */


```

```
74  ////////////////////////////////////////////////////////////////////////////
75  // Audio CODEC interface.
76  //
77  // The interface consists of the following wires:
78  // read_ready, write_ready - CODEC ready for read/write operation
79  // readdata_left, readdata_right - left and right channel data from the CODEC
80  // read - send data from the CODEC (both channels)
81  // writedata_left, writedata_right - left and right channel data to the CODEC
82  // write - send data to the CODEC (both channels)
83  // AUD_* - should connect to top-level entity I/O of the same name.
84  //         These signals go directly to the Audio CODEC
85  // I2C_* - should connect to top-level entity I/O of the same name.
86  //         These signals go directly to the Audio/Video Config module
87  ////////////////////////////////////////////////////////////////////////////
88     clock_generator my_clock_gen(
89        // inputs
90        CLOCK2_50,
91        reset,
92
93        // outputs
94        AUD_XCK
95     );
96
97     audio_and_video_config cfg(
98        // Inputs
99        CLOCK_50,
100       reset,
101
102       // Bidirectionals
103       FPGA_I2C_SDAT,
104       FPGA_I2C_SCLK
105    );
106
107    audio_codec codec(
108       // Inputs
109       CLOCK_50,
110       reset,
111
112       read, write,
113       writedata_left, writedata_right,
114
115       AUD_ADCDAT,
116
117       // Bidirectionals
118       AUD_BCLK,
119       AUD_ADCLRCK,
120       AUD_DACLRCK,
121
122       // Outputs
123       read_ready, write_ready,
124       readdata_left, readdata_right,
125       AUD_DACDAT
126    );
127
128 endmodule
129
130
131
```