THE AWESOME GUITAR GAME

Project Report

Embedded System Design
CSEE 4840

Spring 2012 Semester

Academic supervisor: Professor Stephen Edwards

Imré Frotier de la Messelière (imf2108)

Front page picture source: imf2108.

Acknowledgements

I wish to thank Professor Stephen Edwards for all his help throughout this project and for making this work an instructive and enjoyable experience.

Contents

Introduction

This report gives an account of my work during the Embedded System Design class of the Spring 2012 semester. It focuses on the description of my class project: the Awesome Guitar Game, aka TAGG. I have worked on this project all semester long, beginning with the Project Proposal and the Project Design documents, then learning the necessary skills to realize it throughout the laboratory assignments of the class, and finishing by implementing the project itself.

The Awesome Guitar Game is based on the same principle as in the famous "Guitar Hero" video game series[32]. TAGG is an interactive game where the user can play a game guitar to match up the notes as displayed on the screen. The display continuously reflects that the note has been or has not been correctly played. Most of the technical aspects of this project have been realized based on Professor Stephen Edwards' class documentation and tutorials[23].

In section 1 of the report, I shall present an overview of the project and its objectives. Then, in section 2, I will give more information about the project design document. Section 3 will be dedicated to the milestones of the project and section 4 to a discussion of the architectural and timing design. Section 5 focuses on the hardware used in this project. Then section 6 will be related to the experiences and issues in implementation, while section 7 will be a summary of the project, including the lessons that I learned. Finally, section 8 contains a listing of all source code.

1 Overview of the project and its objectives

In this section, I shall present an overview of my project, including a revised version of my project proposal.

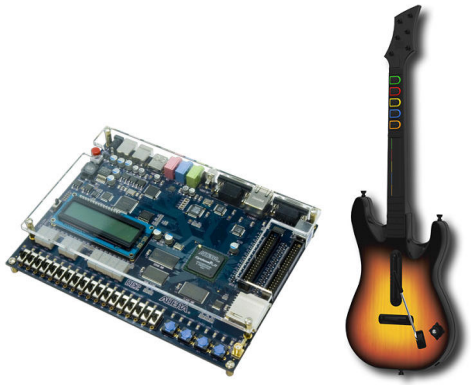
1.1 Revised version of the Project proposal

1.1.1 Project Introduction

In this project, I implement an interactive game where the user can play a game guitar to match up the notes as displayed on the screen. The game principle is the same as in the famous "Guitar Hero" video game series[32].

1.1.2 Project Design

I implement the project using an Altera DE2 board, a game guitar, a VGA display and a pair of speakers.



Altera DE2 Development and Education board and Game Guitar

(altera.com/education/univ/images/boards/de2.jpg; i14.ebayimg.com/01/i/001/22/73/0b2412.JPG)

When the user starts the game, the screen displays a stream of notes. If the user presses the correct button at the correct instant, his score is increased. To make the user even more involved, the display will reflect that the note has been correctly played. The implementation is done on VHDL and C. I have five buttons on the game guitar which are connected to the GPIOs of the FPGA.

1.1.3 Milestones

Please refer to section 3: Milestones of the Project.

1.2 Objectives of the project

The global objectives of the project are:

- building the hardware for the Game Guitar,
- developing the VHDL and C software,

- having the game running.

More precisely, these objectives can be described as follows:

- Build the hardware for the Game Guitar
- Detect the input of the game guitar
- Extract the beats of a song
- Ask for the correct key presses, based on the beats
- Analyze the correctness of the key presses of the player
- Keep track of the score

2 Project Design Document

In this section, I present the detailed project design documents, which are a revised version of the project design document submitted earlier during the year.

2.1 Abstract

This document describes my preliminary project design implementation based on my research at the time when this document was due. In this project, I implement an interactive game where the user can play a game guitar to match up the notes as displayed on a screen. The expected music will be played in the background and the score will grow higher whenever the user presses the correct key. For this project, I will use an Altera DE2 board, a NIOS II processor, a game guitar, a VGA display and a pair of speakers.

2.2 Project Design Introduction

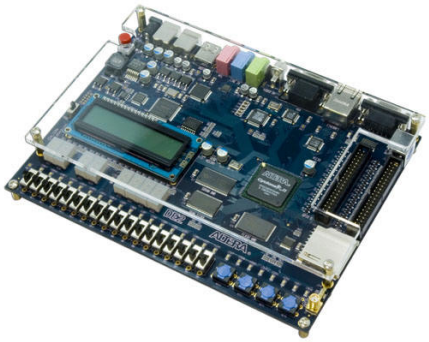
This game is inspired by the Guitar Hero video game series [32]. I use a Guitar Hero controller for PlayStation 2 to serve as the "game guitar", it has 5 colored buttons as well as a button to simulate the action on the guitar string. When the user starts the game, he will hear the music. The screen will display a stream of notes. If the user presses the correct button at the correct instant, his score is increased. To make the user even more involved, the display will reflect that the note has been correctly played.

The implementation of this project will be done on an Altera DE2 board. I will use a NIOS II software processor as the central element of my design, and will write programs in C for this processor. The processor will interact with modules implemented in VHDL.

2.3 Hardware Implementation

2.3.1 Altera DE2 FPGA

The Altera DE2 FPGA will be used to include a NIOS II processor, that will interact with several controllers.



Altera DE2 Development and Education board

(altera.com/education/univ/images/boards/de2.jpg)

It is the central point of my design as it:

- gets the user input through the game guitar,
- contains all the information needed for the game.

The NIOS interacts through an Avalon Bus with the controllers.

2.3.2 Game Guitar and input controller

Game Guitar

(i14.ebayimg.com/01/i/001/22/73/0b2412.JPG)

As said before, I use a Guitar Hero controller for PlayStation 2 as my game guitar. The game guitar is used to get the user input in an enjoyable way for the player. When this document was first released, I didn't know at that point how the guitar would be connected to the FPGA board. So I would open it and figure that out during the first milestone.

The controller will take the input from the the 5 buttons and when the string button is pressed it updates a memory location assigned to it with the color button pressed and sends an interruption to the processor. The interruption is caught and the click is processed. The processor then clears the memory location and the interruption. I will filter for the button bouncing effect at the controller level, not at software level.

The connection from the guitar to the board will be one of following solutions:

1. Using a custom cable that maps every button on a single wire using GPIOs
2. Reusing whatever I could find in the guitar: USB, RS 232 or whatever I will find inside the guitar

2.3.3 Audio format and memory dimensioning

WARNING: This part did not make it to the final version of the project. However, it has involved a consequent amount of work. This is why I wish to keep mentioning it here.

The song is stored in the SDRAM in Binary format. It is sampled down from a sampling rate of 44.1Khz to 8 Khz in order to reduce the memory requirements for

each song. I tested that the song still sounds good at a sampling rate of 8Khz. Every sample is represented on a 8 bits scale. The total memory requirements of a 3 minutes song are 1.44 MB(3*60*1*8000 bytes). This order of magnitude suggests that we use the SDRAM (I need more than the SRAM and less than the SDCARD.). I will therefore be able to store a few songs.

I will use the SRAM to store the sprites and the beats. I will have at maximum 5 sprites of 32x32x8 bit (monochromatic sprites): 5kbytes. The size of the beats binary stream is not fixed as of now. More details are given at the end of this document in the section "Representation of the notes".

2.3.4 Video controller

WARNING: This part did not make it to the final version of the project. However, it has involved a consequent amount of work. This is why I wish to keep mentioning it here.

The video controller is in charge of displaying the sprites on the screen. It is controlled by the processor with the following commands:

- Add a sprite with as parameter the color and the id
- Clear the sprite list

The video controller has a dedicated SRAM for the sprite. I will take as a model the video controller in the Nintendo NES. I aim at a sprite stack of size 4 (4 sprites in the same line maximum). I will have 4 custom sprites for the project: for the note, the note pressed, the note rightly pressed and the background. I will have sprites for the text and numbers in the same fashion than LAB2.

2.4 Software Implementation

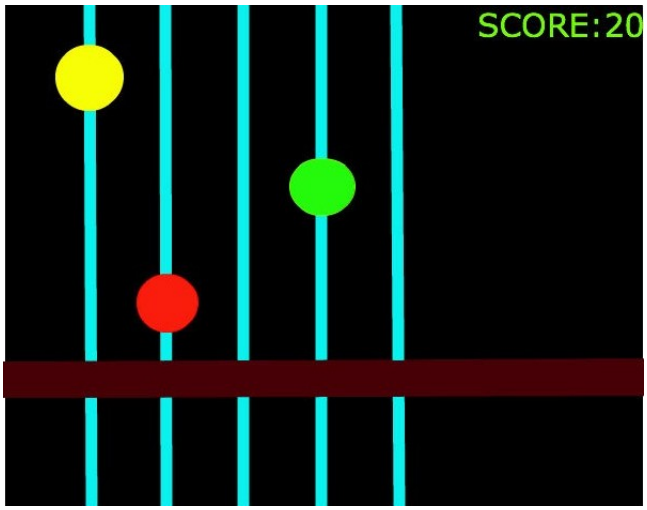
2.4.1 The NIOS program: game loop and event handling

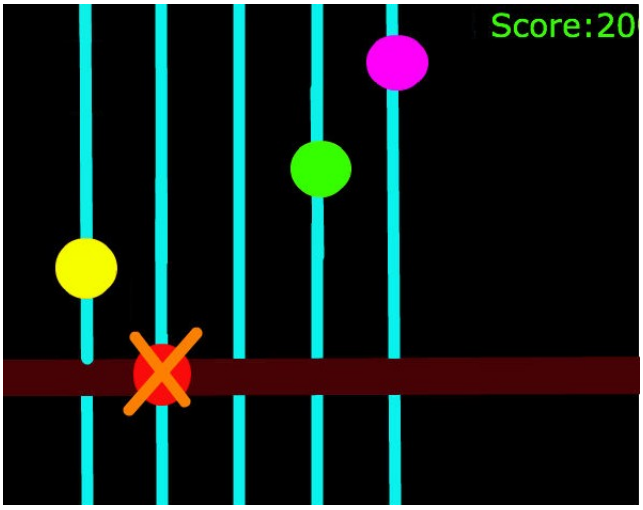
The program that runs on the NIOS processor will be in charge of the Main Game loop, which is a concept that is generally used in video games. It will rely on a timer either as an Avalon Peripheral or a component of the NIOS processor if available to ensure a constant frame rate.

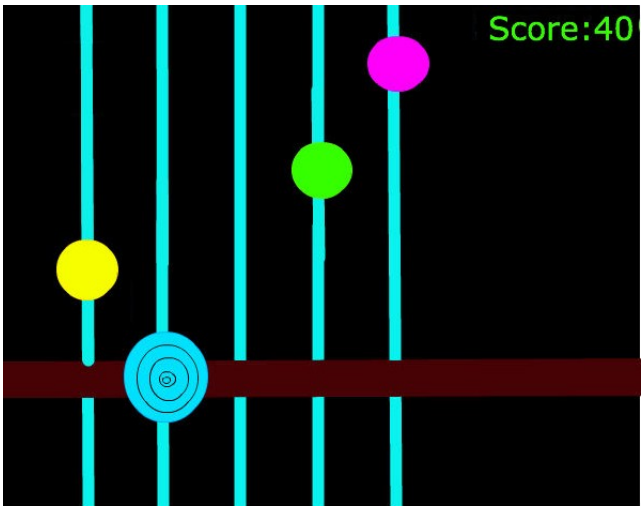
Meanwhile, the program could be interrupted to deal with key presses. Every time a key is pressed, a visual feedback is given for the few next frames and the new score is immediately computed.

WARNING: The following part did not make it to the final version of the project. However, it has involved a consequent amount of work. This is why I wish to keep mentioning it here.

The graphical interface will look like the following:


SCORE:20


Score:20


Score:40

Every iteration of the main loop, I will update the display by computing the new positions of the falling notes.

2.4.2 Beat tracking

The instants at which the notes appear on the screen are not random events.They are synchronized with the beats in the song so as to give a more natural effect.The beats of the song are extracted by a Beat Tracking Algorithm I implemented in Matlab. [35] These beat instances are saved as a .txt file for further usage. They are extracted once and for all so that we do not need to use Matlab when running my game.

2.4.3 Representation of the notes

WARNING: This part did not make it to the final version of the project. However, it has involved a consequent amount of work. This is why I wish to keep mentioning it here.

After having run the beat tracking algorithm on Matlab we obtain a text file that contains n notes n_i, with i in $[0; n-1]$. Each note is characterized by a color and a timestamp when it is played. To transfer the beats in the FPGA, we will convert this text file into a binary format.
I built it inspired by domain specific binary format such as ILDA for laser shows [31].

The binary "stream" associated with a song is made of a header (metadata) and the notes. The header contains:

- the id of the song on 1 byte,
- the number of notes in the song on 2 bytes. 2 bytes seem reasonable based on the number of beats we detect per song in matlab (roughly 400 so 256 = 1 byte is too low).

Following the header we have the notes. Each note is characterized by:

- its "color" on 1 byte. I take more than 3 bits (which is the minimal bound to represent the 5 colors of my guitar's buttons) since I might add new information later. I also go for it to remain consistent with the rest: every field is made of one or more bytes
- its timestamp on m bytes

I haven't yet determined m but to do it I will do some experiments. I will start by finding u integer such as $t/2^u < t_{min} < t/2^u - 1$ with t_{min} the minimum space in seconds between two following beats in the note list and t a constant which represents the maximum length of a song we permit in seconds. I will then choose m such as the multiple of 8 (a byte) 8*m that comes after u. If it happens not to be precise enough in practice, I will increase the value of m.

To compute the binary stream from the output of the Matlab algorithm:

- I set the song id and write it in one byte.
- I discretise all the timestamps to the closer multiple of $t/2^3m$ and store these values.

- Once it is done, I know the number of notes; I write it in the binary stream in two bytes.
- I write the notes one by one in the binary stream following the right format.

2.5 Milestones

Please see section 3: Milestones of the Project.

3 Milestones of the Project

In this section, I provide the details of the three milestones of the project.

3.1 Milestone 1: March 27th

- I will buy and construct the game guitar.
- I shall detect key inputs with the game guitar.
- I will play a song (raw sound format) from a SD card in the FPGA.
- I will develop a program to build a script of a given song. This means, to produce a file that contains the notes and their corresponding positions for this particular song.
- I shall finally make a prototype of the base game engine in Java.

3.2 Milestone 2: April 10th

- I will work on the sprites and study how to do graphics and how to encode the sound efficiently (how many bits, how much information I can store...).
- The Java game prototype will integrate the work on scripts from the first milestone.
- I shall have designed the game internal functioning to ensure a constant frame rate (on paper).
- I will have started implementing the game.

3.3 Milestone 3: April 24th

- I shall finalize the game.
- I will develop an algorithm to compute the score.
- I shall improve the performance of the game and work on the graphics.

3.4 Final run, part one: May 23rd

- I finished coding the VHDL files.
- I finished coding the C files.

3.5 Final run, part two: September 30th

- I finished writing the project report.
- I finished creating the project presentation.

4 Discussion of the architectural and timing design

In this section, I present the final version of the Awesome Guitar Game.

4.1 Architectural design

4.1.1 MATLAB code

This code extracts the beats out of a song. It comes from LabRosa. It is run and stored beforehand in the project. More precisely, given a mp3 song, the MATLAB code extracts the beats of the song and stores them in a text file. This text file is then being processed by a PYTHON code that converts it to a format recognizable in C.

Here is an excerpt of the README for LabROSA-coversongID.

"See http://labrosa.ee.columbia.edu/projects/coversongs/ for more info."

"Key functions:

- t = tempo(d,sr); estimates the tempo in BPM of audio waveform d at sample rate sr
- b = beat(d,sr); estimates the beat times (in sec) of audio waveform d at sample rate sr
- qlist = calclistftrs(querylistfilename); calculates beat-synchronous chroma feature matrices for all the wav or mp3 files listed, one per line, in the named file, returning a list of calculated feature files, then...
- R = coverTestLists(qlist); compares each feature file named in the qlist against every item and returns R as a square matrix of distance between each pair."

4.1.2 PYTHON code

This code stores into a new format the songs that have been preprocessed through MATLAB so that they may be readable inside the VHDL project:

- toHexArray.py
- encode.py
- randomize.py
- shortest_time_dist.py

4.1.3 NIOS II C code

The main code of the NIOS II[12][11] C program is the following:

- hello_world.c

It enables to control the beats and to keep track of the score of the user. The related console serves as a visual display for the game where the player can see which note he is supposed to play and what his current score is.

I set the displayed key each time val is incremented.

```
     static void irqhandler (void * context, alt_u32 id)
val ++;
key = 1;
IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0); // reset request
}
```

This is where I store the beats I extracted earlier with MATLAB:

```
     int beats[] = {0x00b4,0x00c8,0x00f0,0x0116,0x0141,0x016a,0x0193,0x01bb,
...
0x4b71,0x4b98,0x4bc0,0x4be6,0x4c0d,0x4c33,0x4c5d,0x4c82,0x4cad,0x4cd7};
```

This is the interruption detection function:

```
     static void input_isr ( void* context, alt_u32 id){
valz++;
IOWR_16DIRECT(INPUTCONTROLLER_INST_BASE, 0,0);
return;
}
```

The following is part of the main():

```
     int main(){
```

This is how I register the interruptions:

```
     IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0);

     alt_irq_register( INPUTCONTROLLER_INST_IRQ, NULL,(void*)irqhandler ); // register the irq
```

This is the beats counter:

```
     int counter = 0;
```

This indicates the success or failure to press the correct key on time:

```
     int flag = 0;
```

This stores the value of the current key:

```
     int val_buffer = val;
int key_buffer = floor(rand(5));
```

This keeps track of the score:

```
     int score = 0;
```

We have now reached the core of the game:

```
     while(counter<540){
```

If the correct key still hasn't been pressed:

```
     if (flag!=1){
if (val>val_buffer){
```

This is the case of a successful key press:

```
     if (key==key_buffer){
flag = 1;
score = score+1;
printf("SUCCESS: SCORE %d",score);}}}


     if (beat_counter>=(beats[counter])*3){
```

That is the case of a failure to match a note in time:

```
     if (flag == 0){

printf("FAIL: SCORE %d",score);}
```

I then reset the variables:

```
     flag = 0;
val_buffer = val;
```

The game asks for a new specific key press:

```
    key_buffer = floor(1+rand()%5);

    printf("PRESS %d",key_buffer);
counter=counter+1;
}
}
```

The game is finished once all the beats have been read:

```
    printf("FINISHED!");

    return 0; }
```

4.1.4 QUARTUS VHDL code

Here I present the different sets of modules in my VHDL project[30][29][36][3] on QUARTUS[4][13][14] [10][16][25][26]

.

These are the architecture files of the project[15][5][24]:

- nios_system.sopc
- nios_system.vhd

This is the main module of the project:

- guitar_top.vhd

Here is a closer look at its contents:

```
    entity guitar_top is

    port (
signal CLOCK_50 :  in std_logic;

    SRAM_DQ : inout std_logic_vector(15 downto 0);
SRAM_ADDR : out std_logic_vector(17 downto 0);
```

This stores the current key:

```
KEY : in std_logic_vector(3 downto 0);

    SRAM_UB_N,
SRAM_LB_N,
SRAM_WE_N,
SRAM_CE_N,
SRAM_OE_N : out std_logic ;
```

These are the links to the Game Guitar:

```
GPIO_0, - GPIO Connection 0
GPIO_1 :  inout std_logic_vector(35 downto 0) - GPIO Connection 1
);

    end guitar_top;

    architecture datapath of guitar_top is

    signal reset_n :std_logic;
signal new_res:std_logic;
signal audio_clock :  unsigned(1 downto 0) := "00";
signal counter :  unsigned(15 downto 0);

    begin

    process (CLOCK_50)
begin
if rising_edge(CLOCK_50) then
if counter = x"ffff" then
reset_n <= '1';
else
reset_n <= '0';
counter <= counter + 1;
end if;
end if;
end process;

    process (CLOCK_50)
begin
if rising_edge(CLOCK_50) then
audio_clock <= audio_clock + "1";
end if;
end process;

    nios :  entity work.nios_system port map (
clk => CLOCK_50,
reset_n => reset_n,
SRAM_ADDR_from_the_sram => SRAM_ADDR,
SRAM_CE_N_from_the_sram => SRAM_CE_N,
SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
SRAM_LB_N_from_the_sram => SRAM_LB_N,
SRAM_OE_N_from_the_sram => SRAM_OE_N,
SRAM_UB_N_from_the_sram => SRAM_UB_N,
SRAM_WE_N_from_the_sram => SRAM_WE_N,
```

These are the links to Game Guitar key presses:

```
SWITCH_1_to_the_InputController_inst => GPIO_0(0),
SWITCH_2_to_the_InputController_inst => GPIO_0(1) ,
SWITCH_3_to_the_InputController_inst => GPIO_0(2) ,
SWITCH_4_to_the_InputController_inst => GPIO_0(3),
SWITCH_5_to_the_InputController_inst => GPIO_0(4)
);

    end datapath;
```

These modules enable to get the key presses from the user:

- counter.vhd
- DebounceCounter.vhd
- debouncer.vhd
- pulser.vhd

These are the key features of the file counter.vhd:

```
    entity counter is

    port
(
-Synchronous reset
reset :  in std_logic;
clk:  in std_logic;
do:  out unsigned(31 downto 0);
di :  in unsigned (31 downto 0)
);

end counter;


architecture arl of counter is
signal value :  unsigned(31 downto 0);
begin
process(clk)
begin
if(rising_edge(clk)) then
```

The value of the counter is modified at each time click:

```
if( reset='1' )
then
value <= di;
else
value <= value -1;
end if;
end if;
end process;
do <= value; -copy value to the output
end arl;
```

Now, I present the key features of debouncer.vhd:

```
    entity debouncer is

    port (
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic
);

    end debouncer;

    architecture arl of debouncer is
```

"states" defines the different configurations that describe the current state of a button and its next state:

```
    type states is (ZERO, ZERO_TO_ONE, ONE, ONE_TO_ZERO);
```

"states" thus enables me to define “state” and “next_state”:

```
    signal state, next_state :  states;
signal do :  unsigned(18 downto 0);
signal reset_counter:  std_logic; -to reset the counter
begin
```

```
     C0:  entity work.DebounceCounter port map (
clk => clk,
reset => reset_counter,
do => do);

     -FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
state <= ZERO;
else
state <= next_state;
end if;
end if;
end process;
```

This is the computation of the next state:

```
     process (state,key_in,do)

begin

next_state <= state;

case state is
```

This is the case of the ZERO state:

```
     -Signal generated:  reset => 1, key_out => 0

when ZERO =>

reset_counter <= '1';

key_out <= '1';

if (key_in = '0')

then

next_state <= ZERO_TO_ONE;

end if;
```

This is the case of the ZERO_TO_ONE state:

```
     -Signal generated:  reset => 0, key_out => 0

when ZERO_TO_ONE =>

reset_counter <= '0';

key_out <= '1';

if (do >= 500_000)

then

next_state <= ONE;

end if;
```

This is the case of the ONE state:

```
     -Signal generated:  reset => 1, key_out => 1

when ONE =>

reset_counter <= '1';

key_out <= '0';

if (key_in = '1')

then

next_state <= ONE_TO_ZERO;

end if;
```

This is the case of the ONE_TO_ZERO state:

```
     -Signal generated:  reset => 0, key_out => 1
when ONE_TO_ZERO =>
reset_counter <= '0';
key_out <= '0';
if (do >= 500_000)
then
next_state <= ZERO;
end if;
end case;
end process;

     end ar1;
```

I now present the main features of DebounceCounter.vhd:

```
     entity DebounceCounter is

     port(
-Synchronous reset
reset :  in std_logic;
clk:  in std_logic;
do:  out unsigned(18 downto 0)
);

     end DebounceCounter;
architecture ar1 of DebounceCounter is
signal value :  unsigned(18 downto 0);
begin
process(clk)
begin
if(rising_edge(clk)) then
```

The value of the counter is modified at each time click:

```
if( reset='1' )
then
value <= (others => '0');
else
value <= value + 1;
end if;
end if;
end process;
do <= value; -copy value to the output
end ar1;
```

Finally, these are the key features of pulser.vhd. When the input signal switches to 1, it generates a pulse:

```
     entity pulser is

     port
(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic

     );

     end pulser ;

     architecture A1 of pulser is
```

"states" defines the different configurations that describe the pulse:

```
     type states is (ZERO, ONE_PULSE, ONE_STANDBY);
```

"states" thus enables me to define “state” and “next_state”:

```
     signal state, next_state :  states;
begin

     -FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
state <= ZERO;
else
state <= next_state;
end if;
end if;
end process;
```

This is the computation of the next state:

```
     process (state,key_in)

begin

next_state <= state;

case state is
```

This is the case of the ZERO state:

```
     -Signal generated 0
when ZERO =>
key_out <= '1';
if (key_in = '0')
then
next_state <= ONE_PULSE;
end if;
```

This is the case of the ONE_PULSE state:

```
     -Signal generated 1

when ONE_PULSE =>

key_out <= '0';

next_state <= ONE_STANDBY;
```

This is the case of the ONE_STANDBY state:

```
     -Signal generated:  -
when ONE_STANDBY =>
key_out <= '1';
if (key_in = '1')
then
next_state <= ZERO;
end if;
end case;
end process;
end A1;
```

The complete links between the uses of all these files are described more precisely in the following paragraphs, which present the input controllers.

Moreover, these modules convert the key presses from the user to interruptions:

- InputController.vhd
- InputController_inst.vhd
- InputController2_inst.vhd
- InputController3_inst.vhd
- InputController4_inst.vhd
- InputController5_inst.vhd

I use five instances of the InputController vhdl file. I created them following the tutorial[1] from the class website. There is one input controller per button of the Game Guitar. In each of these files the counter, debouncer and pulser vhdl files are called in oreder to detect key presses and thus send an interruption signal.

```
    entity InputController is

port (
-Avalon
clk :  in std_logic;
reset_n :  in std_logic;
read :  in std_logic;
write :  in std_logic;
chipselect :  in std_logic;
address :  in unsigned(9 downto 0);
readdata :  out unsigned(15 downto 0);
writedata :  in unsigned(15 downto 0);
inter :  out std_logic;
```

These are the 5 switches of the Game Guitar

```
    SWITCH_1 :  in std_logic;
SWITCH_2 :  in std_logic;
SWITCH_3 :  in std_logic;
SWITCH_4 :  in std_logic;
SWITCH_5 :  in std_logic
);
end InputController;

architecture rtl of InputController is
```

"states" defines the different configurations of the input controller:

```
    type states is (IDLE, PRESSED,WAITST);
```

"states" thus enables me to define “state” and “next_state”:

```
    signal state, next_state :  states;
signal we:std_Logic;
signal reset:std_logic;
signal counter:  integer;
```

The following is a bunch of signals for the button. The button clicks are first debounced and then pulsed.

The naming follows this convention:

– Initial SWITCH_1: this is the original signal.
Here is a first signal example: _____–_-_————–_–_____

– Debounced DSWITCH_1: The debouncing has been applied.
Here is the new version of our example: _____————_____

– Pulsed PSWITCH_1: The pulser has been applied in order to avoid redundant interruption.

This is the final version of our example: _____-_____

```
-Switch 1
signal DSWITCH_1:std_logic;
signal PSWITCH_1:std_logic;
-Switch 2
signal DSWITCH_2:std_logic;
signal PSWITCH_2:std_logic;
-Switch 3
signal DSWITCH_3:std_logic;
signal PSWITCH_3:std_logic;
-Switch 4
signal DSWITCH_4:std_logic;
signal PSWITCH_4:std_logic;
--Switch 5
signal DSWITCH_5:std_logic;
signal PSWITCH_5:std_logic;
```

This is the debouncer:

```
    component debouncer is port(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic
);
end component debouncer;
```

This is the pulser:

```
    component pulser is port(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic
);
end component pulser;
begin
-Reset
reset <= not reset_n;
-Write enable
we <= '1' when chipselect ='1' and write='1' else '0';
```

These are the mappings of the debouncer:

```
    D1:  debouncer port map(clk,SWITCH_1,DSWITCH_1,reset);
D2:  debouncer port map(clk,SWITCH_2,DSWITCH_2,reset);
```

```
D3:  debouncer port map(clk,SWITCH_3,DSWITCH_3,reset);
D4:  debouncer port map(clk,SWITCH_4,DSWITCH_4,reset);
D5:  debouncer port map(clk,SWITCH_5,DSWITCH_5,reset);
```

"states" thus enables me to define “state” and “next_state”:

```
     P1:  pulser port map(clk,DSWITCH_1,PSWITCH_1,reset);
P2:  pulser port map(clk,DSWITCH_2,PSWITCH_2,reset);
P3:  pulser port map(clk,DSWITCH_3,PSWITCH_3,reset);
P4:  pulser port map(clk,DSWITCH_4,PSWITCH_4,reset);
P5:  pulser port map(clk,DSWITCH_5,PSWITCH_5,reset);

-FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
if( reset_n='0' )
then
state <= IDLE;
else
state <= next_state;
if (state = PRESSED) then
counter <=0;
else counter <=counter +1;
end if;

end if;
end if;
end process;

-Combinational process
process (state,SWITCH_1,we)
begin
-By default reset the interuption signal
inter <= '0';
next_state <= state;
case state is
```

Here I am waiting for a button click:

```
     when IDLE =>
inter <='0';
if (PSWITCH_1 = '0' or PSWITCH_2 = '0' or PSWITCH_3 = '0' or PSWITCH_4 = '0' or PSWITCH_5 = '0' )
then
next_state <= PRESSED;
end if;
```

A button has been pressed. I stay into this PRESSED state until the interuption has been cleared.

```
     when PRESSED =>
inter <='1';
if (we='1')
```

```
then

next_state <= WAITST;

end if;


when WAITST =>

inter <='0';

if (counter > 30000)

then

next_state <= IDLE;

end if;


end case;


end process;


end rtl;
```

This is the main organization of all the instances of InputController:

```
     entity InputController_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController_inst;

     architecture europa of InputController_inst is
component InputController is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController, which is an e_instance
the_InputController :  InputController2
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
```

```
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

The last sets of files are mostly pre-written files already available in the class labs.

Here are the files for the sram[34]:

- de2_sram_controller.vhd
- sram.vhd

And here is the file for the video display[33] and the audio:

- de2_i2c_av_config.v
- de2_i2c_controller.v
- de2_wm8731_audio.vhd
- jtag_uart.vhd

These are other files, which are related to the cpu:

- cpu.vhd
- cpu_jtag_debug_module.vhd
- cpu_jtag_debug_module_wrapper.vhd
- cpu_test_bench.vhd

This is an additional file related to my early tests of the audio of the game:

- lab3_audio.vhd

4.2 Timing design

In this subsection, I present the Timing design of the project[27].

4.2.1 Extraction of the beats

The extraction of the beats is done using LabRosa[35]. It is run and stored beforehand in the project. More precisely, given a mp3 song, the MATLAB code extracts the beats of the song and stores them in a text file. This text file is then being processed by a PYTHON code that converts it to a format recognizable in C.

Here is an excerpt of the README for LabROSA-coversongID.

"See http://labrosa.ee.columbia.edu/projects/coversongs/ for more info."

"Key functions:

- t = tempo(d,sr); estimates the tempo in BPM of audio waveform d at sample rate sr
- b = beat(d,sr); estimates the beat times (in sec) of audio waveform d at sample rate sr
- qlist = calclistftrs(querylistfilename); calculates beat-synchronous chroma feature matrices for all the wav or mp3 files listed, one per line, in the named file, returning a list of calculated feature files, then...
- R = coverTestLists(qlist); compares each feature file named in the qlist against every item and returns R as a square matrix of distance between each pair."

4.3 Correct timing of the beats

The beats are stored inside the main C program:

```
     int beats[] = {0x00b4,0x00c8,0x00f0,0x0116,0x0141,0x016a,0x0193,0x01bb,
...
0x4b71,0x4b98,0x4bc0,0x4be6,0x4c0d,0x4c33,0x4c5d,0x4c82,0x4cad,0x4cd7};
```

They are then accessed at each incrementation of the counter:

```
int main(){

     int counter = 0;

     int flag = 0;
int val_buffer = val;
int key_buffer = floor(rand(5));

     int score = 0;

     while(counter<540){

     if (flag!=1){
if (val>val_buffer){
if (key==key_buffer){
flag = 1;
score = score+1;
printf("SUCCESS: SCORE %d",score);}}}

     if (beat_counter>=(beats[counter])*3){

if (flag == 0){
printf("FAIL: SCORE %d",score);}

flag = 0;
val_buffer = val;
key_buffer = floor(1+rand()%5);

     printf("PRESS %d",key_buffer);
counter=counter+1;
}
```

```
}
printf("FINISHED!");

     return 0; }
```

4.3.1 Use of interruptions to indicate that the guitar has been pressed

The interruptions implementation tutorial by Professor Stephen Edwards is available at [1].

```
     // I set the displayed key each time val is incremented.

     static void irqhandler (void * context, alt_u32 id)
val ++;
key = 1;
IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0); // reset request
}

     static void input_isr ( void* context, alt_u32 id){
valz++;
IOWR_16DIRECT(INPUTCONTROLLER_INST_BASE, 0,0);
return;
}
```

The interruptions are then exploited inside the main() by using "alt_irq_register":

```
     int main(){

     IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0);

     alt_irq_register( INPUTCONTROLLER_INST_IRQ, NULL,(void*)irqhandler ); // register the irq

     int counter = 0;

     int flag = 0;
int val_buffer = val;
int key_buffer = floor(rand(5));

     int score = 0;

     while(counter<540){

     if (flag!=1){
if (val>val_buffer){
if (key==key_buffer){
flag = 1;
score = score+1;
printf("SUCCESS: SCORE %d",score);}}}

     if (beat_counter>=(beats[counter])*3){

if (flag == 0){
printf("FAIL: SCORE %d",score);}

flag = 0;
val_buffer = val;
key_buffer = floor(1+rand()%5);

     printf("PRESS %d",key_buffer);
counter=counter+1;
}
}
printf("FINISHED!");

     return 0; }
```

4.4 Additional tracks of study

4.4.1 Alternative beats handling system

This next set of files did not make it to the final version of the project. However, It has been a consequent track of study which I wish to mention. The timer files were a custom timing function I had design to keep track of the occurences of the beats directly inside the VHDL code. The sixth instance of InputController was to send a notification when a beat was appearing. However, I decided to deal with the beats directly in the NIOS II C code in the end. The files related to this particular track of study are the following:

- InputController6_inst.vhd

- timer.vhd
- timer.vhdl

4.4.2 Storage of the song

Concerning the storage of the song, I first used the SDRAM[28]. The beats controller and the music controller still had to be synchronized. Then, I tried to use the flash memory[2] instead. The beats controller and the music controller still had to be synchronizedas well.

4.4.3 Playing the song

For this part, I reused my work of the lab3 assignment where I dealt with playing audio sounds.

4.4.4 Video display using sprites

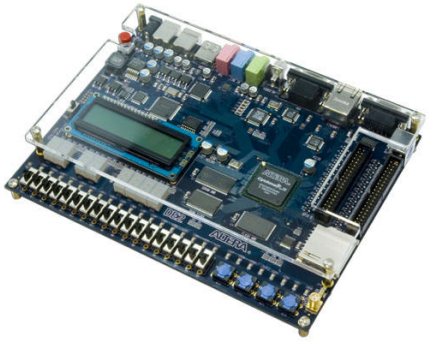
There was some early work done on the display of sprites. I included the resluting files in the "ADDITIONAL TRACKS OF STUDY" folder of the final project.

5 Hardware

In this section, I give a closer look at the hardware that was used in the project.

5.1 FPGA

The main hardware of the project is the FPGA[7][8][9][6].



Altera DE2 Development and Education board

(altera.com/education/univ/images/boards/de2.jpg)

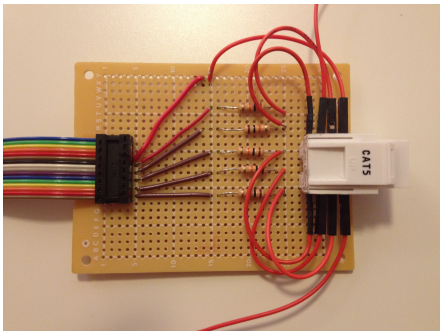
5.2 Game Guitar

I have customized a Guitar Hero controller to become the Game Guitar.

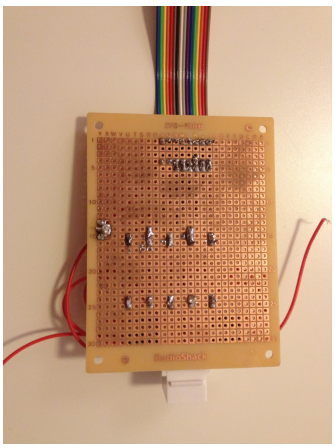
Game Guitar

(i14.ebayimg.com/01/i/001/22/73/0b2412.JPG)

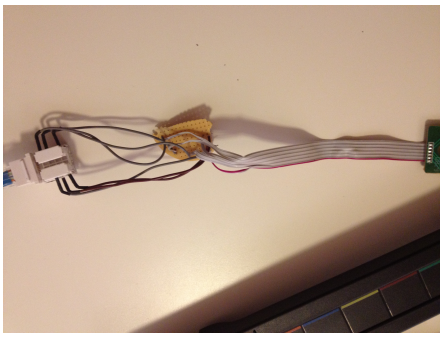
Connexion between the two band wires of the FPGA
(imf2108)



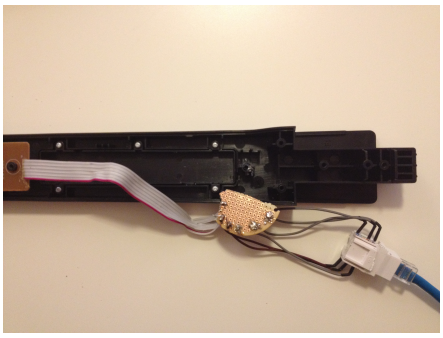
Front view of the connexion between the second band wire of the FPGA and the ethernet port of the Game Guitar
(imf2108)



Back view of the connexion between the second band wire of the FPGA and the ethernet port of the Game Guitar
(imf2108)



Front view of the connexion of the ethernet cable to the Game Guitar
(imf2108)



Back view of the connexion of the ethernet cable to the Game Guitar
(imf2108)

5.3 VGA display

The VGA dispaly I used for this project is one of the displays that are available in the Embedded Systems laboratory. This is where I use the console of the NIOS II interface.

VGA display
(cclonline.com/images/MediaPool/PRQ1d111fvkKq4fqdrRmYg-3d-3d_LargeProductImage.jpg)

6 Experiences and issues in implementation

In this section, I present the experiences and issues in implementation, including the difficult parts.

6.1 Playing the song

At the beginning of the project, I ran tests with 1 second samples form a song in the VHDL code of lab3. However, it was hard to make oneself a sure opinion of the result as the playtime was so short.

I had thought about several ways to store the song:

- Store it in the SDRAM
- Store it in the Flash memory
- Use a raw sound from an external source

After trying all three possibilities, I decided to use the last one. This decision was based on project time constraints.

6.2 Storage using the SDRAM

WHen implementing the SDRAM, I had to work on a step by step basis. First of all, I followed the tutorial for the SDRAM setup. Once the SDRAM had been added to the project, I still had write on it and read it. These two parts were the trickiest ones. I had to carefully study of the memory locations where indexed so that each element of the song would fit inside the SDRAM without any risk of overlapping or gaps. I thus had to run tests ot understand which address I was accessing and how many bits I was reading or writing when I was using a specific address.

6.3 Bugs solving

It was generally difficult to trace the source of a crash, especially in VHDL. The debugging information was scarce so it required a lot of time to locate the source of a bug. It often involved backtracking the source of the crash through multiple files.

Debugging was easier using C. However, missing libraries were often an issue, and finding these libraries was not always a solution to some of the problems that I wanted to solve. This was especially true when I was dealing with the time element of this project.

6.4 Merging several parts of the project

Merging several parts of the project was always tricky. Generally, I was working on two aspects of the project at the same time. So when both of them were done, it was time to join them so that the project could continue to go on. What I did was to take the more complex of the two works, and progressively add the elements of the other one inside it.

I first added all the new files. Then I modifiedd the files that were common to both these work by adding the lines of code of the second work that were absent from the

first one. This often implied a consequent debugging phase, due to the apparition of bugs that were specific to the coexistence of these two works in the same project.

7 Summary including lessons learned

In this section, I provide a summary of the project as well as a listing of what lessons I learned during this project. I also give some advice for future projects.

7.1 Summary of the project

The global objectives of the project have been fulfilled:

- building the hardware for the Game Guitar,
- developing the VHDL and C software,
- having the game running.

More precisely, these features are available in the final version of the project:

- Hardware of the Game Guitar
- Detection of the input of the game guitar
- Extraction of the beats of a song
- Asking for the correct key presses, based on the beats
- Analysis of the correctness of the key presses of the player
- Keeping track of the score

7.2 Lessons learned

- Start the project as early as possible.
- Do not venture into too many directions at once; it is more efficient to stay focused on a limited number of tasks.
- Code and debug small step per small step.
- When a specific track of study progressively seems to be a dead end, find a new solution as early as possible.
- Synchronize all the parts of the project as soon as possible.
- Carefully plan ahead when to come to the Embedded Systems laboratory in order to have an available workstation and always keep in mind that other students need these workstations as well so it is imperative to share ressources. That way, all projects make progress and the mood in the laboratory is much more enjoyable.
- Make regular copies of the global projects in order to be able to go back to a previous version in case the current one get corrupted by a bug too difficult to find quickly.
- Do not hesitate to ask the teaching Assistants for information when starting to study a new programming language.

- Be very careful with the SOPC builder.
- The most important lesson of all: enjoy your project!

Enjoy the Awesome Guitar Game!
(imf2108)

7.3 Some advice for future projects

- If it is your first time coding with VHDL or C, be very careful in the first days of the project and regularly test your work.
- As I said previously, future students should keep in mind that other students need the workstations of the Embedded Systems laboratory as well so it is imperative to share ressources. That way, all projects make progress and the mood in the laboratory is much more enjoyable.
- Do not hesitate to refer repeatedly to the previous years projects and projects reports [17][18][19][20][21][22]. They contain very valuable information from students who already saw an Embedded System Design project through.

8 Listing of all source code

Here is a complete listings of every file I wrote for the project. It includes C source, VHDL source, and things such as .mhs files. I did not include any file that was generated automatically. I mad sure of that by applying "rm -rf db incremental_db .rpt .done .summary .smsg .pin .qdf .pof" beforehand. I did not include the .bak files either.

8.1 MATLAB source code

- beatavg.m
- beat.m
- calclistftrs.m
- chromagram_E.m
- chromagram_IF.m
- chromagram_P.m
- chrombeatftrs.m
- chromnorm.m
- chrompwr.m
- chromrot.m
- chromxcorr.m
- coverDistMxLists.m
- coverFtrExLists.m
- coverTestLists.m
- distmatrixwrite.m
- fexist.m
- fft2chromamx.m
- fft2melmx.m
- history-bragg-autoco.m
- history-golddust-xcorr.m
- hz2octs.m
- ifgram.m
- ifptrack.m
- listfileread.m
- listfilewrite.m

- localmax.m
- mkblips.m
- mp3read.m
- mymkdir.m
- octs2hz.m
- tempo.m
- testlist.m
- test.m

8.2 PYTHON source code

- toHexArray.py
- encode.py
- randomize.py
- shortest_time_dist.py

8.3 C source code

- hello_world.c

8.4 VHDL source code

- AWESOME_GUITAR.qpf
- AWESOME_GUITAR.qws
- AWESOME_GUITAR_TOP.dpf
- AWESOME_GUITAR_TOP.jdi
- AWESOME_GUITAR_TOP.qsf
- AWESOME_GUITAR_TOP.sof
- counter.vhd
- cpu.ocp
- cpu.vhd
- cpu_jtag_debug_module.vhd
- cpu_jtag_debug_module_wrapper.vhd
- cpu_ociram_default_contents.mif
- cpu_rf_ram.mif

- cpu_test_bench.vhd
- DebounceCounter.vhd
- debouncer.vhd
- de2_i2c_av_config.v
- de2_i2c_controller.v
- de2_sram_controller.vhd
- de2_sram_controller_hw.tcl
- de2_wm8731_audio.vhd
- guitar_top.vhd
- InputController.vhd
- InputController_hw.tcl
- InputController_inst.vhd
- InputController2_inst.vhd
- InputController3_inst.vhd
- InputController4_inst.vhd
- InputController5_inst.vhd
- InputController6_inst.vhd
- jtag_uart.vhd
- nios_system.bsf
- nios_system.ptf
- nios_system.qip
- nios_system.sopc
- nios_system.vhd
- nios_system_generation_script
- nios_system_log.txt
- nios_system.ptf.pre_generation_ptf
- nios_system_setup_quartus.tcl
- pulser.vhd
- sopc_builder_log.txt
- sram.vhd

- timer.vhd
- timer.vhdl
- timer_hw.tcl
- timer_inst.vhd

Conclusion

Once again, I wish to thank Professor Stephen Edwards for all his help throughout this entire project and for making this class a very enjoyable experience. It has been most instructive and gave me more insight in dealing with some specific types of hardware.

In the end, I have designed a video game that enables the user to match the notes that are asked from him on a VGA display using a game guitar while his score is being modified based on how successful he is. Coding and debuging this software has been a really entertaining activity. It enabled me to discover VHDL, reinforce my skills in C, and continue to use MATLAB and PYTHON.

Using, modifying and building hardware by myself has also been a very enjoyable experience. This gave a "hands-on" feel to the project that enabled to see the physical effects of my work. As a matter of fact, it was the first time I was welding again since middle school.

Right now, this project enables the player to match the required notes using the game guitar I designed, all the while following instructions and his score using a VGA display. For students who will take this class in the future and who wish to do do a project based on this one, here are some hints on what to work on, using my work as a basis. The first element worth implementing is a more developed visual interface for the game, using sprites. It would also be nice to have several different songs available instead of just one. Finally, another cool feature would be to introduce a multiplayer mode.

Bibliography

[1] Tutorial: Creating peripherals with interrupts in altera's sopc builder author =.

[2] S29al032d; 32 megabit cmos 3.0 volt-only flash memory; 4 m x 8-bit uniform sector; 4 m x 8-bit/2 m x 16-bit boot sector; data sheet preliminary. (S29AL032D_00; Revision A; Amendment 5), Sep 2005.

[3] Avalon memory-mapped interface specification. (3.3), May 2007.

[4] Quartus ii version 7.2 handbook. (Volume 3: Verification), May 2007.

[5] Pong P. Chu. Embedded sopc design with nios ii processor and vhdl examples. 2011.

[6] Altera Corporation. Cyclone ii device handbook. (Volume 1).

[7] Altera Corporation. De2 development and education board.

[8] Altera Corporation. De2 development and education board. (3.3).

[9] Altera Corporation. De2 development and education board; user manual.

[10] Altera Corporation. Introduction to the quartus® ii software. (Version 7.2).

[11] Altera Corporation. Nios ii processor reference handbook.

[12] Altera Corporation. Nios ii software developer's handbook.

[13] Altera Corporation. Quartus ii version 7.2 handbook. (Volume 2: Design Implementation and Optimization).

[14] Altera Corporation. Quartus ii version 7.2 handbook. (Volume 1: Design and Synthesis).

[15] Altera Corporation. Quartus ii version 7.2 handbook. (Volume 4: SOPC Builder).

[16] Altera Corporation. Quick start guide for quartus ii software.

[17] cs.columbia.edu/ sedwards/classes/2006/4840/index.html.

[18] cs.columbia.edu/ sedwards/classes/2007/4840/index.html.

[19] cs.columbia.edu/ sedwards/classes/2008/4840/index.html.

[20] cs.columbia.edu/ sedwards/classes/2009/4840/index.html.

[21] cs.columbia.edu/ sedwards/classes/2010/4840/index.html.

[22] cs.columbia.edu/ sedwards/classes/2011/4840/index.html.

[23] cs.columbia.edu/ sedwards/classes/2012/4840/index.html.

[24] Stephen A. Edwards. Introduction to the altera sopc builder.

[25] Stephen A. Edwards. Quartus ii introduction using vhdl design.

[26] Stephen A. Edwards. Quartus ii simulation with vhdl designs.

[27] Stephen A. Edwards. Timing considerations with vhdl-based designs.

[28] Stephen A. Edwards. Using the sdram memory on altera's de2 board with vhdl design.

[29] Stephen A. Edwards. Writing vhdl for rtl synthesis. *Columbia University*, January 2011.

[30] esd.cs.ucr.edu/labs/tutorial/.

[31] http://www.laserist.org/.

[32] hub.guitarhero.com/.

[33] Texas Instruments. 9900; tms9918a/tms9928a/tms9929a; video display processors. 1982.

[34] Inc. Integrated Circuit Solution. Is42s8800/is42s8800l is42s16400/is42s16400l; 2(1)m words x 8(16) bits x 4 banks (64-mbit); synchronous dynamic ram.

[35] labrosa.ee.columbia.edu/projects/coversongs/.

[36] Mark Zwolinski. Digital system design with vhdl. *Pearson/Prentice-Hall*, January 2004.

Annexes

Here are the C and VHDL source files I wrote or modified. I did not include any automatically-generated files in this report. I mad sure of that by applying "rm -rf db incremental_db .rpt .done .summary .smsg .pin .qdf .pof" beforehand. I did not include the .bak files either.

MATLAB source code

beatavg.m

```
     function X = beatavg(Y,bts)
% X = beatavg(Y,bys)
% Calculate average of columns of Y according to grid defined
% (real-valued) column indices in vector bts.
% For folding spectrograms down into beat-sync features.
% 2006-09-26 dpwe@ee.columbia.edu

     % beat-based segments
%bts = beattrack(d,sr);
nbts = length(bts);
bttime = mean(diff(bts));
% map beats to specgram slices
ncols = size(Y,2);
coltimes = [0:(ncols-1)];
cols2beats = zeros(nbts, ncols);
btse = [bts,max(coltimes)];
for b = 1:nbts
cols2beats(b,:)  = ((coltimes >= btse(b)) & (coltimes < btse(b+1)))*1/(btse(b+1)-btse(b));
end

     % The actual desired output
X = Y * cols2beats';
```

beat.m

```
     function [b,onsetenv,D,cumscore] = beat(d,sr,startbpm,tightness,doplot)
% [b,onsetenv,D,cumscore] = beat(d,sr,startbpm,tightness,doplot)
% b returns the times (in sec) of the beats in the waveform d, samplerate sr.
% startbpm specifies the target tempo.  If it is a two-element
% vector, it is taken as the mode of a tempo search window, with
% the second envelope being the spread (in octaves) of the
% search, and the best tempo is calculated (with tempo.m).
% tightness controls how tightly the start tempo is enforced
% within the beat (default 6, larger = more rigid); if it is a
% two-element vector the second parameter is alpha, the strength
% of transition costs relative to local match (0..1, default 0.7).
% doplot enables diagnostic plots; if it has two elements, they
% are the time range (in sec) for the diagnostic plots.
% onsetenv returns the raw onset detection envelope
% D returns the mel-spectrogram,
% cumscore returns the per-frame cumulated dynamic-programming score.
% 2006-08-25 dpwe@ee.columbia.edu
% uses:  localmax

     % Copyright (c) 2006 Columbia University.
%
% This file is part of LabROSA-coversongID
%
% LabROSA-coversongID is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License version 2 as
% published by the Free Software Foundation.
%
% LabROSA-coversongID is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with LabROSA-coversongID; if not, write to the Free Software
% Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
% 02110-1301 USA
%
% See the file "COPYING" for the text of the license.

     if nargin < 3; startbpm = 0; end
if nargin < 4; tightness = 0; end
if nargin < 5; doplot = 0; end

     if length(startbpm) == 2
temposd = startbpm(2);
startbpm = startbpm(1);
else
temposd = 0;
end
if length(tightness) == 2
alpha = tightness(2);
```

```
tightness = tightness(1);
else
alpha = 0.8
end
if tightness == 0; tightness = 6; end

     % Have we been given an envelope (nonnegative waveform)
if min(d) >= 0
onsetenv = d;
sgsrate = sr;
disp(['beat:  treating input as onset strength envelope']);
else
onsetenv = [];
end

     % debug/plotting options
plotlims = [];
if length(doplot) > 1
% specify zoom-in limits too
plotlims = doplot;
doplot = 1;
end
if doplot > 0; debug = 1; else debug = 0; end

     b = [];

     % Select tempo search either with startbpm = 0 (means use defaults)
% or startbpm > 0 but temposd > 0 too (means search around startbpm)
% If onsetenv is empty, have to run tempo too to convert waveform
% to onsetenv, but we might not use the tempo it picks.
if startbpm == 0 | temposd > 0 | length(onsetenv) == 0

     if startbpm == 0
tempomean = 120;
else
tempomean = startbpm;
end

     if temposd == 0
temposd = 0.7;
end

     % Subfunction estimates global BPM; returns 'onset strength'
% waveform onsetenv
% If we were given an onsetenv as input, will use that
t,xcr,D,onsetenv,sgsrate = tempo(d,sr,tempomean,temposd,onsetenv,debug);

     % tempo.m returns the top-2 BPM estimates; use faster one for
% beat tracking
if (startbpm == 0 | temposd > 0)
startbpm = max(t([1 2]));
end

     if debug == 1
% plot the mel-specgram
tt = [1:length(onsetenv)]/sgsrate;
subplot(411)
imagesc(tt,[1 40],D); axis xy
subplot(412)
plot(tt,onsetenv);
end

     end

     % convert startbpm to startpd
startpd = (60*sgsrate)/startbpm;
%disp(['startpd=',num2str(startpd)]);

     pd = startpd;

     % Smooth beat events
templt = exp(-0.5*((-pd:pd/(pd/32)).^2));
localscore = conv(templt,onsetenv);
localscore = localscore(round(length(templt)/2)+[1:length(onsetenv)]);

     % DP version:
% backlink(time) is index of best preceding time for this point
% cumscore(time) is total cumulated score to this point

     backlink = zeros(1,length(localscore));
cumscore = zeros(1,length(localscore));

     % search range for previous beat
prange = round(-2*pd):-round(pd/2);

     % Skewed window
txwt = exp(-0.5*((tightness*log(prange/-pd)).^2));

     starting = 1;
for i = 1:length(localscore)

     timerange = i + prange;

     % Are we reaching back before time zero?
zpad = max(0, min(1-timerange(1),length(prange)));

     % Search over all possible predecessors and apply transition
% weighting
scorecands = txwt .* [zeros(1,zpad),cumscore(timerange(zpad+1:end))];
```

```
% Find best predecessor beat
vv,xx = max(scorecands);
% Add on local score
cumscore(i) = alpha*vv + (1-alpha)*localscore(i);

     % special case to catch first onset
% if starting == 1 & localscore(i) > 100*abs(vv)
if starting == 1 & localscore(i) < 0.01*max(localscore);
backlink(i) = -1;
else
backlink(i) = timerange(xx);
% prevent it from resetting, even through a stretch of silence
starting = 0;
end

     end

     %%%% Backtrace

     % Cumulated score is stabilized to lie in constant range,
% so just look for one near the end that has a reasonable score
medscore = median(cumscore(localmax(cumscore)));
bestendx = max(find(cumscore .* localmax(cumscore) > 0.5*medscore));

     b = bestendx;

     while backlink(b(end)) > 0
b = [b,backlink(b(end))];
end

     b = fliplr(b);

     % return beat times in secs
b = b / sgsrate;

     % Debug visualization
if doplot == 1
subplot(411)
hold on;
plot([b;b],[0;40]*ones(1,length(b)),'w');
hold off;
subplot(412)
hold on;
plot([b;b],[-5;20]*ones(1,length(b)),'g');
hold off;

     % redo 3rd pane as xcorr with templt
subplot(413)
tt = [1:length(localscore)]/sgsrate;
plot(tt,localscore);
hold on; plot([b;b],[min(localscore);max(localscore)]*ones(1,length(b)),'g'); hold off

     if length(plotlims) > 0
for i = 1:3;
subplot(4,1,i)
ax = axis;
ax([1 2]) = plotlims;
axis(ax);
end
end

     end
```

calclistftrs.m

This is a file that I did not modify from its original version.

chromagram_E.m

```
     function Y = chromagram_E(d,sr,fftlen,nbin,f_ctr,f_sd)
% Y = chromagram_E(d,sr,fftlen,nbin)
% Calculate a "chromagram" of the sound in d (at sampling rate sr)
% Use windows of fftlen points, hopped by ffthop points
% Divide the octave into nbin steps
% Weight with center frequency f_ctr (in Hz) and gaussian SD f_sd (in octaves)
% 2006-09-26 dpwe@ee.columbia.edu

     if nargin < 3; fftlen = 2048; end
if nargin < 4; nbin = 12; end
if nargin < 5; f_ctr = 1000; end
if nargin < 6; f_sd = 1; end

     fftwin = fftlen/2;
ffthop = fftlen/4; % always for this

     D = abs(specgram(d,fftlen,sr,fftwin,(fftwin-ffthop)));

     A0 = 27.5; % Hz
A440 = 440; % Hz

     f_ctr_log = log(f_ctr/A0) / log(2);
```

```
    CM = fft2chromamx(fftlen, nbin, sr, A440, f_ctr_log, f_sd);
% Chop extra dims
CM = CM(:,1:(fftlen/2)+1);


    Y = CM*D;
```

chromagram_IF.m

This is a file that I did not modify from its original version.

chromagram_P.m

```
    function Y = chromagram_P(d,sr,fftlen,nbin,f_ctr,f_sd)
% Y = chromagram_E(d,sr,fftlen,nbin)
% Calculate a "chromagram" of the sound in d (at sampling rate sr)
% Use windows of fftlen points, hopped by ffthop points
% Divide the octave into nbin steps
% Weight with center frequency f_ctr (in Hz) and gaussian SD f_sd (in octaves)
% 2006-09-26 dpwe@ee.columbia.edu

    if nargin < 3; fftlen = 2048; end
if nargin < 4; nbin = 12; end
if nargin < 5; f_ctr = 1000; end
if nargin < 6; f_sd = 1; end

    fftwin = fftlen/2;
ffthop = fftlen/4; % always for this

    D = abs(specgram(d,fftlen,sr,fftwin,(fftwin-ffthop)));

    [nr,nc] = size(D);

    A0 = 27.5; % Hz
A440 = 440; % Hz

    f_ctr_log = log(f_ctr/A0) / log(2);

    CM = fft2chromamx(fftlen, nbin, sr, A440, f_ctr_log, f_sd);
% Chop extra dims
CM = CM(:,1:(fftlen/2)+1);

    % Keep only local maxes in freq
Dm = (D > D([1,[1:nr-1]],:))  & (D >= D([[2:nr],nr],:));
Y = CM*(D.*Dm);
```

chrombeatftrs.m

```
    function [F,bts] = chrombeatftrs(d,sr,f_ctr,f_sd,type)
% [F,bts] = chrombeatftrs(D,SR,F_CTR,F_SD,TYPE)
% F returns a feature vector of beat-level chroma features (12
% rows x n time step columns).  bts returns the times of all the
% beats.
% New version separates out chroma calculation
% TYPE selects chroma calculation type; 1 (default) uses IF;
% 2 uses all FFT bins, 3 uses only local peaks (a bit like Emilia).
% 2006-07-14 dpwe@ee.columbia.edu

    % Copyright (c) 2006 Columbia University.
% % This file is part of LabROSA-coversongID
% % LabROSA-coversongID is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License version 2 as
% published by the Free Software Foundation.
% % LabROSA-coversongID is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
% % You should have received a copy of the GNU General Public License
% along with LabROSA-coversongID; if not, write to the Free Software
% Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
% 02110-1301 USA
% % See the file "COPYING" for the text of the license.

    if nargin < 3; f_ctr = 1000; end
if nargin < 4; f_sd = 1; end
if nargin < 5; type = 1; end

    tempomean = 240;
temposd = 1.5;

    % Try beat tracking now for quick answer
bts = beat(d,sr,[tempomean temposd],[6 0.8],0);

    % Calculate frame-rate chromagram
fftlen = 2 ^(round(log(sr*(2048/22050))/log(2)));
nbin = 12;

    if type == 2
Y = chromagram_E(d,sr,fftlen,nbin,f_ctr,f_sd);
elseif type == 3
```

```
Y = chromagram_P(d,sr,fftlen,nbin,f_ctr,f_sd);
else
Y = chromagram_IF(d,sr,fftlen,nbin,f_ctr,f_sd);
end

     ffthop = fftlen/4;
sgsrate = sr/ffthop;

     F = beatavg(Y,bts*sgsrate);
```

chromnorm.m

```
     function [N,S] = chromnorm(F)
% [N,S] = chromnorm(F)
% Normalize each column of a chroma ftrvec to unit norm
% so cross-correlation will give cosine distance
% S returns the per-column original norms, for reconstruction
% 2006-07-14 dpwe@ee.columbia.edu

     [nchr, nbts] = size(F);

     S = sqrt(sum(F.^2));


     N = F./repmat(S+(S==0), nchr, 1);
```

chrompwr.m

```
     function Y = chrompwr(X,P)
% Y = chrompwr(X,P) raise chroma columns to a power, preserving norm
% 2006-07-12 dpwe@ee.columbia.edu

     [nbins,nframes] = size(X);

     % norms of each input col
CMn = repmat(sqrt(sum(X.^2)),nbins,1);
CMn(CMn == 0) = 1;

     % normalize each input col, raise to power
CMp = (X./CMn).^P;

     % norms of each resultant column
CMpn = repmat(sqrt(sum(CMp.^2)),nbins,1);
CMpn(CMpn == 0) = 1;

     % rescale cols so norm of output cols match norms of input cols
Y = CMn.*(CMp./CMpn);
```

chromrot.m

```
     function Y = chromrot(X,N)
% Y = chromrot(X,N)
% Rotate each column of chroma feature matrix X down by N
% semitones.
% 2006-07-15 dpwe@ee.columbia.edu

     [nr,nc] = size(X);


     Y = X(1+rem([0:(nr-1)]+N+nr,nr),:);
```

chromxcorr.m

```
     function r = chromxcorr(A,F,L)
% r = chromxcorr(A,F,L)
% Cross-correlate two chroma ftr vecs in both time and
% transposition
% Both A and F can be long, result is full convolution
% (length(A) + length(F) - 1 columns, in F order).
% L is the maximum lag to search to - default 100.
% of shorter, 2 = by length of longer
% Optimized version.
% 2006-07-14 dpwe@ee.columbia.edu

     % Copyright (c) 2006 Columbia University.
% % This file is part of LabROSA-coversongID
% % LabROSA-coversongID is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License version 2 as
% published by the Free Software Foundation.
% % LabROSA-coversongID is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
% % You should have received a copy of the GNU General Public License
% along with LabROSA-coversongID; if not, write to the Free Software
% Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
% 02110-1301 USA
```

```
% % See the file "COPYING" for the text of the license.

     if nargin < 3; L = 100; end

     [nchr,nbts1] = size(A);
nchr2,nbts2 = size(F);

     if nchr  = nchr2
error('chroma sizes dont match');
end

     r = zeros(nchr, 2*L+1);

     for i = 1:nchr
rr = 0;
for j = 1:nchr
rr = rr + xcorr(F(1+rem(j+i-2,nchr),:),A(j,:),L);
end
r(i,:)  = rr;
end


     % Normalize by shorter vector so max poss val is 1

r = r/min(nbts1,nbts2);
```

coverDistMxLists.m

This is a file that I did not modify from its original version.

coverTestLists.m

This is a file that I did not modify from its original version.

distmatrixwrite.m

```
     function distmatrixwrite(matrix,rownames,colnames,filename,tag)
% distmatrixwrite(matrix,rownames,colnames,filename,tag)
% Write a distance matrix
% 2006-08-06 dpwe@ee.columbia.edu

     if nargin < 5
tag = 'dpweDefault';
end

     fid = fopen(filename, 'w');

     fprintf(fid, '%s n', tag);
     usecolnames = zeros(length(colnames));

     nn = 0;

     for i = 1:length(rownames);
nn = nn+1;
fprintf(fid, '%d t%s n', nn, rownamesi);
% if this name also occurs as a col, remember we don't need it
usecolnames(strcmp(colnames, rownames(i))) = nn;
end

     for i = 1:length(colnames);
if usecolnames(i) == 0
nn = nn+1;
fprintf(fid, '%d t%s n', nn, colnamesi);
usecolnames(i) = nn;
end
end

     % Matrix heading
fprintf(fid, 'Q/R');
for i = 1:length(usecolnames);
fprintf(fid, '%d', usecolnames(i));
end
fprintf(fid, ' n');

     % Rows of matrix
for i = 1:size(matrix,1);
fprintf(fid, '%d', i);
fprintf(fid, ' t%f', matrix(i,:));
fprintf(fid, ' n');
end
```

fexist.m

```
     function E = fexist(F)
% E = fexist(F) returns 1 if file F exists, else 0
% 2006-08-06 dpwe@ee.columbia.edu
```

```
    x = dir(F);

    E = length(x);
```

fft2chromamx.m

This is a file that I did not modify from its original version.

fft2melmx.m

This is a file that I did not modify from its original version.

history-bragg-autoco.m

This is a file that I did not modify from its original version.

history-golddust-xcorr.m

This is a file that I did not modify from its original version.

hz2octs.m

```
    function octs = hz2octs(freq, A440)
% octs = hz2octs(freq, A440)
% Convert a frequency in Hz into a real number counting
% the octaves above A0.  So hz2octs(440) = 4.0
% Optional A440 specifies the Hz to be treated as middle A (default 440).
% 2006-06-29 dpwe@ee.columbia.edu for fft2chromamx

    if nargin < 2; A440 = 440; end

    % A4 = A440 = 440 Hz, so A0 = 440/16 Hz
octs = log(freq./(A440/16))./log(2);
```

ifgram.m

This is a file that I did not modify from its original version.

ifptrack.m

This is a file that I did not modify from its original version.

listfileread.m

```
    function [L,N] = listfileread(F)
% [L,N] = listfileread(F) Read a list of per-line items
% F is a file containing a list of items, one per line.
% Return L as a cell array, with one item per line, N as the
% number of items.
% If F is not found, return empty L and N == -1 (instead of 0).
% 2006-08-06 dpwe@ee.columbia.edu for MIREX 06

    N = -1;
L = [];

    if fexist(F) == 1

    fid = fopen(F);

    nitems = 0;

    while 1
tline = fgetl(fid);
if  ischar(tline), break, end

    nitems = nitems+1;
Lnitems = tline;
end
fclose(fid);
N = nitems;


    end
```

listfilewrite.m

```
     function N = listfilewrite(L, F)
% N = listfilewrite(L, F) Write a list of items to a file
% L is a cell array of strings.  Write to file F, one per line.
% N returns the number of items successfully written.
% 2006-08-06 dpwe@ee.columbia.edu for MIREX 06

     fid = fopen(F, 'w');

     nit = length(L);

     for i = 1:nit
fprintf(fid, '%s n', Li);
end

     fclose(fid);
```

localmax.m

```
     function m = localmax(x) % return 1 where there are local maxima in x (columnwise).
% don't include first point, maybe last point

     [nr,nc] = size(x);

     if nr == 1
lx = nc;
elseif nc == 1
lx = nr;
x = x';
else
lx = nr;
end

     if (nr == 1) || (nc == 1)

     m = (x > [x(1),x(1:(lx-1))]) & (x >= [x(2:lx),1+x(lx)]);

     if nc == 1
% retranspose
m = m';
end

     else
% matrix
lx = nr;
m = (x > [x(1,:);x(1:(lx-1),:)])  & (x >= [x(2:lx,:);1+x(lx,:)]);


     end
```

mkblips.m

```
     function d = mkblips(b,sr,l)
% d = mkblips(b,sr,l)
% Make a blip track at the set of times in b (in sec).  Output
% waveform has sampling rate sr (default 8000) and length l
% samples (default:  long enough to contain all blips).
% 2006-05-02, 2006-09-30 dpwe@ee.columbia.edu

     if nargin < 2
sr = 8000;
end
if nargin < 3
l = 0;
end

     % 100ms pip @ 2khz
tdur = 0.1;
fblip = 2000;
tt = (0:round(tdur*sr))';
x = tt.*exp(-tt/((tdur*sr)/10)).*cos(2*pi*tt/sr*fblip)/200;

     lx = length(x);

     bsamp = round(b*sr);

     % remove beats that would run off end
if l > 0
bsamp = bsamp(bsamp < (l-lx));
else
l = max(bsamp)+lx;
end

     d = zeros(1,l);


     for bb = bsamp

d(bb+[1:lx]) = d(bb+[1:lx]) + x;

end
```

mp3read.m

```
     function [Y,FS,NBITS,OPTS] = mp3read(FILE,N,MONO,DOWNSAMP,DELAY)
% MP3READ Read MP3 audio file via use of external binaries.
% Y = MP3READ(FILE) reads an mp3-encoded audio file into the
% vector Y just like wavread reads a wav-encoded file (one channel
% per column).  Extension ".mp3" is added if FILE has none.
% Also accepts other formats of wavread, such as
% Y = MP3READ(FILE,N) to read just the first N sample frames (N
% scalar), or the frames from N(1) to N(2) if N is a two-element vector.
% Y = MP3READ(FILE,FMT) or Y = mp3read(FILE,N,FMT)
% with FMT as 'native' returns int16 samples instead of doubles;
% FMT can be 'double' for default behavior (to exactly mirror the
% syntax of wavread).
%
% [Y,FS,NBITS,OPTS] = MP3READ(FILE...)  returns extra information:
% FS is the sampling rate, NBITS is the bit depth (always 16),
% OPTS.fmt is a format info string; OPTS has multiple other
% fields, see WAVREAD.
%
% SIZ = MP3READ(FILE,'size') returns the size of the audio data contained
% in the file in place of the actual audio data, returning the
% 2-element vector SIZ=[samples channels].
%
% [Y...]  = MP3READ(FILE,N,MONO,DOWNSAMP,DELAY) extends the
% WAVREAD syntax to allow access to special features of the
% mpg123 engine:  MONO = 1 forces output to be mono (by
% averaging stereo channels); DOWNSAMP = 2 or 4 downsamples by
% a factor of 2 or 4 (thus FS returns as 22050 or 11025
% respectively for a 44 kHz mp3 file);
% To accommodate a bug in mpg123-0.59, DELAY controls how many
% "warm up" samples to drop at the start of the file; the
% default value of 2257 makes an mp3write/mp3read loop for a 44
% kHz mp3 file be as close as possible to being temporally
% aligned; specify as 0 to prevent discard of initial samples.
% For later versions of mpg123 (e.g.  1.9.0) this is not needed;
% a flag in mp3read.m makes the default DELAY zero in this case.
%
% [Y...]  = MP3READ(URL...)  uses the built-in network
% functionality of mpg123 to read an MP3 file across the
% network.  URL must be of the form 'http://...'  or
% 'ftp://...'.  'size' and OPTS are not available in this mode.
%
% Example:
% To read an mp3 file as doubles at its original width and sampling rate:
% [Y,FS] = mp3read('piano.mp3');
% To read the first 1 second of the same file, downsampled by a
% factor of 4, cast to mono, using the default filename
% extension:
% [Y,FS4] = mp3read('piano', FS/4, 1, 4);
%
% Note:  Because the mp3 format encodes samples in blocks of 26 ms (at
% 44 kHz), and because of the "warm up" period of the encoder,
% the file length may not be exactly what you expect, depending
% on your version of mpg123 (recent versions fix warmup).
%
% Note:  requires external binaries mpg123 and mp3info; you
% can find binaries for several platforms at:
% http://labrosa.ee.columbia.edu/matlab/mp3read.html
%
% See also mp3write, wavread.

     % $Header:  /Users/dpwe/matlab/columbiafns/RCS/mp3read.m,v 1.6 2009/12/08 16:35:23 dpwe Exp dpwe $

     % 2003-07-20 dpwe@ee.columbia.edu This version calls mpg123.
% 2004-08-31 Fixed to read whole files correctly
% 2004-09-08 Uses mp3info to get info about mp3 files too
% 2004-09-18 Reports all mp3info fields in OPTS.fmt; handles MPG2LSF sizes
% + added MONO, DOWNSAMP flags, changed default behavior.
% 2005-09-28 Fixed bug reading full-rate stereo as 1ch (thx bjoerns@vjk.dk)
% 2006-09-17 Chop off initial 2257 sample delay (for 44.1 kHz mp3)
% so read-write loop doesn't get progressively delayed.
% You can suppress this with a 5th argument of 0.
% 2007-02-04 Added support for FMT argument to match wavread
% Added automatic selection of binary etc.  to allow it
% to work cross-platform without editing prior to
% submitting to Matlab File Exchange
% 2007-07-23 Tweaks to 'size' mode so it exactly agrees with read data.
% 2009-03-15 Added fixes so 'http://...'  file URLs will work.
% 2009-03-26 Added filename length check to http:  test (thx fabricio guzman)

     % find our baseline directory
path = fileparts(which('mp3read'));

     % %%%%% Directory for temporary file (if needed)
% % Try to read from environment, or use /tmp if it exists, or use CWD
tmpdir = getenv('TMPDIR');
if isempty(tmpdir) || exist(tmpdir,'file')==0
tmpdir = '/tmp';
end
if exist(tmpdir,'file')==0
tmpdir = '';
end
% ensure it exists
%if length(tmpdir) > 0 && exist(tmpdir,'file')==0
% mkdir(tmpdir);
%end
```

```
     %%%%%% Command to delete temporary file (if needed)
rmcmd = 'rm';

     %%%%%% Location of the binaries - attempt to choose automatically
%%%%%% (or edit to be hard-coded for your installation)
ext = lower(computer);
if ispc
ext = 'exe';
rmcmd = 'del';
end
% mpg123-0.59 inserts silence at the start of decoded files, which
% we compensate.  However, this is fixed in mpg123-1.9.0, so
% make this flag 1 only if you have mpg123-0.5.9
MPG123059 = 0;
mpg123 = fullfile(path,['mpg123.',ext]);
mp3info = fullfile(path,['mp3info.',ext]);

     %%%%% Check for network mode
if length(FILE) > 6 && (strcmp(lower(FILE(1:7)),'http://') == 1 ...
|| strcmp(lower(FILE(1:6)),'ftp://'))
% mp3info not available over network
OVERNET = 1;
else
OVERNET = 0;
end

     %%%%% Process input arguments
if nargin < 2
N = 0;
end

     % Check for FMT spec (per wavread)
FMT = 'double';
if ischar(N)
FMT = lower(N);
N = 0;
end

     if length(N) == 1
% Specified N was upper limit
N = [1 N];
end
if nargin < 3
forcemono = 0;
else
% Check for 3rd arg as FMT
if ischar(MONO)
FMT = lower(MONO);
MONO = 0;
end
forcemono = (MONO  = 0);
end
if nargin < 4
downsamp = 1;
else
downsamp = DOWNSAMP;
end
if downsamp  = 1 && downsamp  = 2 && downsamp  = 4
error('DOWNSAMP can only be 1, 2, or 4');
end

     % process DELAY option (nargin 5) after we've read the SR

     if strcmp(FMT,'native') == 0 && strcmp(FMT,'double') == 0 && ...
strcmp(FMT,'size') == 0
error(['FMT must be "native" or "double" (or "size"), not "',FMT,""]);
end

     %%%%%% Constants
NBITS=16;

     %%%%% add extension if none (like wavread)
path,file,ext = fileparts(FILE);
if isempty(ext)
FILE = [FILE, '.mp3'];
end

     if  OVERNET
%%%%%% Probe file to find format, size, etc.  using "mp3info" utility
cmd = ['"',mp3info, '" -r m -p "%Q %u %b %r %v * %C %e %E %L %O %o %p" "', FILE,'"'];
% Q = samprate, u = #frames, b = #badframes (needed to get right answer from %u)
% r = bitrate, v = mpeg version (1/2/2.5)
% C = Copyright, e = emph, E = CRC, L = layer, O = orig, o = mono, p = pad
w = mysystem(cmd);
% Break into numerical and ascii parts by finding the delimiter we put in
starpos = findstr(w,'*');
nums = str2num(w(1:(starpos - 2)));
strs = tokenize(w((starpos+2):end));

     SR = nums(1);
nframes = nums(2);
nchans = 2 - strcmp(strs6, 'mono');
layer = length(strs4);
bitrate = nums(4)*1000;
mpgv = nums(5);
% Figure samples per frame, after
% http://board.mp3-tech.org/view.php3?bn=agora_mp3techorg&key=1019510889
if layer == 1
smpspfrm = 384;
```

```
elseif SR < 32000 && layer ==3
smpspfrm = 576;
if mpgv == 1
error('SR < 32000 but mpeg version = 1');
end
else
smpspfrm = 1152;
end

     OPTS.fmt.mpgBitrate = bitrate;
OPTS.fmt.mpgVersion = mpgv;
% fields from wavread's OPTS
OPTS.fmt.nAvgBytesPerSec = bitrate/8;
OPTS.fmt.nSamplesPerSec = SR;
OPTS.fmt.nChannels = nchans;
OPTS.fmt.nBlockAlign = smpspfrm/SR*bitrate/8;
OPTS.fmt.nBitsPerSample = NBITS;
OPTS.fmt.mpgNFrames = nframes;
OPTS.fmt.mpgCopyright = strs1;
OPTS.fmt.mpgEmphasis = strs2;
OPTS.fmt.mpgCRC = strs3;
OPTS.fmt.mpgLayer = strs4;
OPTS.fmt.mpgOriginal = strs5;
OPTS.fmt.mpgChanmode = strs6;
OPTS.fmt.mpgPad = strs7;
OPTS.fmt.mpgSampsPerFrame = smpspfrm;
else
% OVERNET mode
OPTS = [];
% guesses
smpspfrm = 1152;
SR = 44100;
nframes = 0;
end

     if SR == 16000 && downsamp == 4
error('mpg123 will not downsample 16 kHz files by 4 (only 2)');
end

     % process or set delay
if nargin < 5

     if MPG123059 mpg123delay44kHz = 2257; % empirical delay of lame/mpg123 loop
mpg123delay16kHz = 1105; % empirical delay of lame/mpg123 loop for 16 kHz sampling
if SR == 16000
rawdelay = mpg123delay16kHz;
else
rawdelay = mpg123delay44kHz; % until we know better
end
delay = round(rawdelay/downsamp);
else
% seems like predelay is fixed in mpg123-1.9.0
delay = 0;
end
else
delay = DELAY;
end

     if downsamp == 1
downsampstr = '';
else
downsampstr = [' -',num2str(downsamp)];
end
FS = SR/downsamp;

     if forcemono == 1
nchans = 1;
chansstr = ' -m';
else
chansstr = '';
end

     % Size-reading version
if strcmp(FMT,'size') == 1
Y = [floor(smpspfrm*nframes/downsamp)-delay, nchans];
else

     % Temporary file to use
tmpfile = fullfile(tmpdir, ['tmp',num2str(round(1000*rand(1))),'.wav']);

     skipx = 0;
skipblks = 0;
skipstr = '';
sttfrm = N(1)-1;

     % chop off transcoding delay?
%sttfrm = sttfrm + delay; % empirically measured
% no, we want to *decode* those samples, then drop them
% so delay gets added to skipx instead

     if sttfrm > 0
skipblks = floor(sttfrm*downsamp/smpspfrm);
skipx = sttfrm - (skipblks*smpspfrm/downsamp);
skipstr = [' -k ', num2str(skipblks)];
end
skipx = skipx + delay;

     lenstr = '';
endfrm = -1;
```

```
decblk = 0;
if length(N) > 1
endfrm = N(2);
if endfrm > sttfrm
decblk = ceil((endfrm+delay)*downsamp/smpspfrm) - skipblks + 10;
% we read 10 extra blks (+10) to cover the case where up to 10 bad
% blocks are included in the part we are trying to read (it happened)
lenstr = [' -n ', num2str(decblk)];
% This generates a spurious "Warn:  requested..." if reading right
% to the last sample by index (or bad blks), but no matter.
end
end

     % Run the decode
cmd=['"',mpg123,'"', downsampstr, chansstr, skipstr, lenstr, ...
'  -q -w "', tmpfile,'" "',FILE,'"'];
%w =
mysystem(cmd);

     % Load the data
Y = wavread(tmpfile);

     % % pad delay on to end, just in case
% Y = [Y; zeros(delay,size(Y,2))];
% % no, the saved file is just longer

     if decblk > 0 && length(Y) < decblk*smpspfrm/downsamp
% This will happen if the selected block range includes >1 bad block
disp(['Warn:  requested ', num2str(decblk*smpspfrm/downsamp),' frames, returned ',num2str(length(Y))]);
end

     % Delete tmp file
mysystem([rmcmd,' "', tmpfile,'"']);

     % debug
% disp(['sttfrm=',num2str(sttfrm),' endfrm=',num2str(endfrm),' skipx=',num2str(skipx),' delay=',num2str(delay),' len=',num2str(length(Y))]);

     % Select the desired part
if skipx+endfrm-sttfrm > length(Y)
endfrm = length(Y)+sttfrm-skipx;
end

     if endfrm > sttfrm
Y = Y(skipx+(1:(endfrm-sttfrm)),:);
elseif skipx > 0
Y = Y((skipx+1):end,:);
end

     % Convert to int if format = 'native'
if strcmp(FMT,'native')
Y = int16((2Î5)*Y);
end

     end

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = mysystem(cmd)
% Run system command; report error; strip all but last line
s,w = system(cmd);
if s  = 0
error(['unable to execute ',cmd,' (',w,')']);
end
% Keep just final line
w = w((1+max([0,findstr(w,10)])):end);
% Debug
%disp([cmd,' -> ','*',w,'*']);

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a = tokenize(s,t)
% Break space-separated string into cell array of strings.
% Optional second arg gives alternate separator (default ' ')
% 2004-09-18 dpwe@ee.columbia.edu
if nargin < 2; t = ' '; end
a = [];
p = 1;
n = 1;
l = length(s);
nss = findstr([s(p:end),t],t);
for ns = nss
% Skip initial spaces (separators)
if ns == p
p = p+1;
else
if p <= l
an = s(p:(ns-1));
n = n+1;
p = ns+1;
end
end
end
```

mymkdir.m

```
     function r = mymkdir(dir)
% r = mymkdir(dir)
% Ensure that dir exists by creating all its parents as needed.
```

```
% 2006-08-06 dpwe@ee.columbia.edu


     [x,m,i] = fileattrib(dir);

if x == 0

pdir,nn,ee,vv = fileparts(dir);

mymkdir(pdir);

disp(['creating ',dir,' ...']);

mkdir(pdir, nn);

end
```

octs2hz.m

```
     function hz = octs2hz(octs,A440)
% hz = octs2hz(octs,A440)
% Convert a real-number octave
% into a frequency in Hzfrequency in Hz into a real number counting
% the octaves above A0.  So hz2octs(440) = 4.0.
% Optional A440 specifies the Hz to be treated as middle A (default 440).
% 2006-06-29 dpwe@ee.columbia.edu for fft2chromamx

     if nargin < 2; A440 = 440; end

     % A4 = A440 = 440 Hz, so A0 = 440/16 Hz

     hz = (A440/16).*(2.ôcts);
```

tempo.m

```
     function [t,xcr,D,onsetenv,sgsrate] = tempo(d,sr,tmean,tsd,onsetenv,debug)
% [t,xcr,D,onsetenv,sgsrate] = tempo(d,sr,tmean,tsd,onsetenv,debug)
% Estimate the overall tempo of a track for the MIREX McKinney
% contest.
% d is the input audio at sampling rate sr.  tmean is the mode
% for BPM weighting (in bpm) and tsd is its spread (in octaves).
% onsetenv is an already-calculated onset envelope (so d is
% ignored).  debug causes a debugging plot.
% Output t(1) is the lower BPM estimate, t(2) is the faster,
% t(3) is the relative weight for t(1) compared to t(2).
% xcr is the windowed autocorrelation from which the BPM peaks were picked.
% D is the mel-freq spectrogram
% onsetenv is the "onset strength waveform", used for beat tracking
% sgsrate is the sampling rate of onsetenv and D.
% % 2006-08-25 dpwe@ee.columbia.edu
% uses:  localmax, fft2melmx

     % Copyright (c) 2006 Columbia University.
% % This file is part of LabROSA-coversongID
% % LabROSA-coversongID is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License version 2 as
% published by the Free Software Foundation.
% % LabROSA-coversongID is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
% % You should have received a copy of the GNU General Public License
% along with LabROSA-coversongID; if not, write to the Free Software
% Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
% 02110-1301 USA
% % See the file "COPYING" for the text of the license.

     if nargin < 3; tmean = 120; end
if nargin < 4; tsd = 3.0; end
if nargin < 5; onsetenv = []; end
if nargin < 6; debug = 0; end

     sro = 8000;
% specgram:  256 bin @ 8kHz = 32 ms / 4 ms hop
swin = 256;
shop = 32;
% mel channels
nmel = 40;
% sample rate for specgram frames (granularity for rest of processing)
sgsrate = sro/shop;
% autoco out to 4 s
acmax = round(4*sgsrate);

     D = 0;

     if length(onsetenv) == 0
% no onsetenv provided - have to calculate it

     % resample to 8 kHz
if (sr  = sro)
gg = gcd(sro,sr);
d = resample(d,sro/gg,sr/gg);
sr = sro;
```

```
end

     D = specgram(d,swin,sr,swin,swin-shop);

     % Construct db-magnitude-mel-spectrogram
mlmx = fft2melmx(swin,sr,nmel);
D = 20*log10(max(1e-10,mlmx(:,1:(swin/2+1))*abs(D)));

     % Only look at the top 80 dB
D = max(D, max(max(D))-80);

     % The raw onset decision waveform
mm = (mean(max(0,diff(D')')));
eelen = length(mm);

     % dc-removed mm
onsetenv = filter([1 -1], [1 -.99],mm);

     end % of onsetenv calc block

     % Find rough global period
% Only use the 1st 90 sec to estimate global pd (avoid glitches?)

     maxdur = 90; % sec
maxcol = min(round(maxdur*sgsrate),length(onsetenv));

     xcr = xcorr(onsetenv(1:maxcol),onsetenv(1:maxcol),acmax);

     % find local max in the global ac
rawxcr = xcr(acmax+1+[0:acmax]);

     % window it around default bpm
xcrwin = exp(-.5*((log((60*sgsrate./([0:acmax]+0.1)/tmean))/log(2)*tsd).^2));
xcr = rawxcr.*xcrwin;

     xpks = localmax(xcr);
% will not include any peaks in first down slope (before goes below
% zero for the first time)
xpks(1:min(find(xcr<0))) = 0;
% largest local max away from zero
maxpk = max(xcr(xpks));

     % ??  then period is shortest period with a peak that approaches the max
%maxpkthr = 0.4;
%startpd = -1 + min(find( (xpks.*xcr) > maxpkthr*maxpk ) );
%startpd = -1 + (find( (xpks.*xcr) > maxpkthr*maxpk ) );

     % no, just largest peak after windowing
startpd = -1 + find((xpks.*xcr) == max(xpks.*xcr));

     % ??Choose acceptable peak closest to 120 bpm
%[vv,spix] = min(abs(60./(startpd/sgsrate) - 120));
%startpd = startpd(spix);
% No, just choose shortest acceptable peak
startpd = startpd(1);

     t = 60/(startpd/sgsrate);

     % Choose best peak out of .33 .5 2 3 x this period
candpds = round([.33 .5 2 3]*startpd);
candpds = candpds(candpds < acmax);

     [vv,xx] = max(xcr(1+candpds));

     startpd2 = candpds(xx);
vvm = xcr(1+startpd);
pratio = vvm/(vvm+vv);

     t = [60/(startpd/sgsrate) 60/(startpd2/sgsrate) pratio];

     % ensure results are lowest-first
if t(2) < t(1)
t([1 2]) = t([2 1]);
t(3) = 1-t(3);
end

     if debug > 0

     % Report results and plot weighted autocorrelation with picked peaks
disp(['Global bt pd = ',num2str(t(1)),' @ ',num2str(t(3)),' / ',num2str(t(2)),' bpm']);

     subplot(414)
plot([0:acmax],xcr,'-b', ...
0:acmax,xcrwin*maxpk,'-r', ...
startpd startpd, [min(xcr) max(xcr)], '-g', ...
startpd2 startpd2, [min(xcr) max(xcr)], '-c');
grid;

     end


     % Read in all the tempo settings

% for i = 1:20; f = fopen(['mirex-beattrack/train/train',num2str(i),'-tempo.txt']); r(i,:)  = fscanf(f, '
```

testlist.m

```
     function [R,S,T] = testlist(queryflist,testflist,pwr,norm,metric,xcr,verb);
% [R,S,T] = testlist(queryflist,testflist,pwr,norm,metric,xcr,verb);
% Takes a list of query files and compares each one to all
% of a list of test files.
% R returns a matrix of score values, each row is a query, each
% column is one of the test elements.
% S is a local normalization index; T is the best alignment time skew.
% pwr is power to raise chroma vectors to (dflt 1).
% norm is the normalization mode for the xcorr (0 = none, 1 = by shorter)
% metric is the metric used (1 = peak xcorr, 2 = peak filtered xcorr)
% xcr = 1 means fast cross-correlation (default 0)
% verb > 0 means provide progress update
% 2006-07-27 dpwe@ee.columbia.edu

     if nargin < 3; pwr = 1; end
if nargin < 4; norm = 1; end
if nargin < 5; metric = 1; end
if nargin < 6; xcr = 0; end
if nargin < 7; verb = 0; end

     [tsongs,ntsongs] = listfileread(testflist);
qsongs,nqsongs = listfileread(queryflist);

     % Now run through the queries

     for q = 1:nqsongs

     qline = qsongsq;

     if (verb > 0)
disp([datestr(rem(now,1),'HH:MM:SS'), ' ', 'doing song ',num2str(q),' ', qline]);
end

     Q = load(qline);
Q.F = chromnorm(chromapwr(Q.F,pwr));

     maxlag = 800;

     for i = 1:ntsongs
% if (verb > 0)
% disp(['..versus ', tsongsi]);
% end

     P = load(tsongsi);
P.F = chromnorm(chromapwr(P.F,pwr));

     if xcr == 0
r = chromxcorr2(Q.F, P.F, norm);
else
% fast version of xcor
r = chromxcorr2fast(Q.F, P.F, norm, maxlag);
end

     mmr = max(max(r));
bestchrom = find(max(r') == mmr);
besttime = find(max(r) == mmr);

     if metric == 1
R(q,i) = mmr;
S(q,i) = mean(mean(r(:,max(besttime-100,1):min(besttime+100,size(r,2)))));

     elseif metric == 2

     % Look for rapid variation - do HPF along time of best chrom
fxc = filter([1 -1], [1 -.9], r(bestchrom,:)-mean(r(bestchrom,:)));
% chop off first bit - onset transient for
% start-in-the-middle
fxc(1:50) = min(fxc);
%
R(q,i) = max(fxc);
if (xcr == 0)
refpt = length(Q.F);
else
refpt = maxlag;
end
besttime = find(fxc == max(fxc))-refpt;
T(q,i) = besttime;
if verb > 0
disp([datestr(rem(now,1),'HH:MM:SS'), ' ..versus ', tsongsi,' ',num2str(max(fxc)),' @ ',num2str(besttime)])
end

     S(q,i) = sqrt(mean(fxc(max(besttime+refpt-100,1):min(besttime+refpt+100,length(fxc))).^2));
end
end
end


     if (0) % not wanted for submission version - self included in lists

% scoring - find largest entry in each row (query)

vv,xx = max(R');

tt = 1:length(xx);

aa = (xx == tt);
```

```
disp(['Simple max:  acc = ',num2str(mean(aa),3),' ',num2str(aa)]);

vv,xx = max((R./S)');

aa = (xx == tt);

disp(['Ratio max:  acc = ',num2str(mean(aa),3),' ',num2str(aa)]);

end
```

test.m

```
     function [] = test (B,d6,sr6)

     figure();
hold

     for i = 1:length(B)

     counter = floor(4*rand(1));

     if (counter ==0)

     'PRESS A'

     elseif (counter == 1)

     'PRESS B'

     elseif (counter == 2)

     'PRESS C'

     elseif (counter == 3)

     'PRESS D'

     end
plot([mod(i,2)+i,mod(i+1,2)+i+1],[mod(i,2),mod(i+1,2)]);
sound(d6((B(i)*sr6):  ((B(i)+1)*sr6)),sr6)
end

     end
```

PYTHON source code

toHexArray.py

```
     #Encode a song into binary format
from struct import *
import sys


#Usage notice
if len(sys.argv)==1:
print """usage:
python encode.py file1 file2
file1 => name of the input file
file1 => name of the output file
"""
exit()

#Arguments acquiring
filein = sys.argv[1]
fileout = sys.argv[2]

#open the file that contains the notes
f = open(filein)
lines = f.readlines()
f.close()


for i in range(len(lines)):
lines[i] = hex(int(lines[i]))+str("")

num = len(lines)

#open the binary file for write
g = open(fileout,"w")
g.write(",".join(lines))
#close binary stream
g.close()
```

encode.py

```
     #Encode a song into binary format
from struct import *
import sys


#Usage notice
if len(sys.argv)==1:
print """usage:
python encode.py file1 file2 id

file1 => name of the input file
file2 => name of the output binary file
id => id of the song
"""
exit()

#Arguments acquiring
filein = sys.argv[1]
fileout = sys.argv[2]
id_song = sys.argv[3]

#Resolution computing
NBYTE = 2
MAX_LENGTH = 60.0*4 #4 minutes
res = int( (2**(NBYTE*8)) / MAX_LENGTH )

#open the file that contains the notes
f = open(filein)
lines = f.readlines()
f.close()

#get the number of notes
num = len(lines)

#open the binary file for write
g = open(fileout,"w")
#write the id of the song, the number of notes
# c = character(1 byte), h = short(2 bytes)
id_song = int(id_song)
g.write(pack("B", id_song))
g.write(pack("h", num))


#Iterating through the beats
for i in range(len(lines)):
color = int(lines[i].split()[0])
time = float(lines[i].split()[1])
val = time*res #timestamp value
g.write(pack("Bh",color,val))


#close binary stream
g.close()
```

randomize.py

```
     #Insert random notes in front of every line
import sys
import random

#Acquire the notes
f = open(sys.argv[1])
lines = f.readlines()
f.close()

#Add the random notes
for i in range(len(lines)):
lines[i]=str(random.randrange(0,4))+" "+lines[i]


#Write the resulting list
f = open(sys.argv[1],"w")
f.write("".join(lines))
f.close()
```

shortest_time_dist.py

```
     #Compute the shortest time length
#between two consecutive notes in
#a song
#

import sys
import random

#Acquire the notes
f = open(sys.argv[1])
lines = f.readlines()
f.close()
```

```
dist = 87439873298473298 #a lot

for i in range(len(lines)-1):
cur = float(lines[i].split()[1].rstrip()) #current note
nex = float(lines[i+1].split()[1].rstrip()) #next note
if nex - cur < dist:
dist = nex - cur

print "shortest distance "+str(dist)
```

C source code

Hello World.c

```
    /*
* "Hello World" example.
*
* This example prints 'Hello from Nios II' to the STDOUT stream.  It runs on
* the Nios II 'standard', 'full_featured', 'fast', and 'low_cost' example
* designs.  It runs with or without the MicroC/OS-II RTOS and requires a STDOUT
* device in your system's hardware.
* The memory footprint of this hosted application is  69 kbytes by default
* using the standard reference design.
*
* For a reduced footprint version of this template, and an explanation of how
* to reduce the memory footprint for a given application, see the
* "small_hello_world" template.
*
*/
#include <system.h>
#include <stdio.h>
#include <io.h>
#include <sys/alt_irq.h>
#include <alt_types.h>
#include <time.h>
#include <unistd.h>
#ifdef __unix__
# include <unistd.h>
#elif defined _WIN32
# include <windows.h>
#define sleep(x) Sleep(1000 * x)
#endif

volatile int val = 0;
volatile int key = 0;

// for beats
volatile int beat_counter = 0;

// I set the displayed key each time val is incremented.

static void irqhandler (void * context, alt_u32 id)
{
val ++;
key = 1;
IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0); // reset request
}

static void irqhandler2 (void * context, alt_u32 id)
{
val ++;
key = 2;
IOWR_8DIRECT(INPUTCONTROLLER2_INST_BASE, 0, 0); // reset request
}

static void irqhandler3 (void * context, alt_u32 id)
{
val ++;
key = 3;
IOWR_8DIRECT(INPUTCONTROLLER3_INST_BASE, 0, 0); // reset request
}

static void irqhandler4 (void * context, alt_u32 id)
{
val ++;
key = 4;
IOWR_8DIRECT(INPUTCONTROLLER4_INST_BASE, 0, 0); // reset request
}

static void irqhandler5 (void * context, alt_u32 id)
{
val ++;
key = 5;
IOWR_8DIRECT(INPUTCONTROLLER5_INST_BASE, 0, 0); // reset request
}

static void irqhandler6 (void * context, alt_u32 id)
{
beat_counter = beat_counter+2;//2 coups de clock
IOWR_8DIRECT(INPUTCONTROLLER6_INST_BASE, 0, 0); // reset request
}
```

```
volatile int valz = 0;

typedef struct note{
int color ;
int timestamp;
}note;

int beats[] = {
0x00b4,0x00c8,0x00f0,0x0116,0x0141,0x016a,0x0193,0x01bb,0x01e3,0x020a,0x0232,0x025a,0x0281,0x02a8,0x02d1,
0x02fa,0x0321,0x0348,0x0370,0x039a,0x03c1,0x03eb,0x0414,0x043a,0x0463,0x048a,0x04b3,0x04da,0x0503,0x052b,
0x0553,0x057a,0x05a2,0x05ca,0x05f2,0x061a,0x0642,0x066a,0x0692,0x06ba,0x06e2,0x0709,0x0731,0x0759,0x0781,
0x07a8,0x07d0,0x07f8,0x0820,0x0848,0x0872,0x0898,0x08bd,0x08e4,0x090c,0x0933,0x095a,0x0981,0x09ac,0x09d7,
0x0a00,0x0a25,0x0a4c,0x0a75,0x0a9c,0x0ac4,0x0aef,0x0b17,0x0b40,0x0b68,0x0b8f,0x0bb6,0x0bde,0x0c06,0x0c2f,
0x0c56,0x0c7e,0x0ca6,0x0cce,0x0cf6,0x0d1e,0x0d45,0x0d70,0x0d98,0x0dc0,0x0de4,0x0e0c,0x0e34,0x0e5c,0x0e85,
0x0eac,0x0ed4,0x0efb,0x0f24,0x0f4b,0x0f71,0x0f98,0x0fc2,0x0feb,0x1014,0x103c,0x1064,0x108b,0x10b3,0x10db,
0x1103,0x112b,0x1153,0x117b,0x11a2,0x11cb,0x11f2,0x121a,0x1243,0x126c,0x1292,0x12bb,0x12e2,0x130a,0x1332,
0x135a,0x1382,0x13aa,0x13d1,0x13f9,0x1421,0x144a,0x1471,0x1499,0x14c2,0x14e8,0x1510,0x153a,0x1560,0x1588,
0x15b0,0x15d9,0x15ff,0x1628,0x164f,0x1677,0x169f,0x16c7,0x16ee,0x1717,0x173f,0x1768,0x178f,0x17b9,0x17df,
0x1808,0x182e,0x1856,0x187e,0x18a6,0x18cd,0x18f6,0x191d,0x1945,0x196d,0x1995,0x19bd,0x19e6,0x1a0d,0x1a36,
0x1a5e,0x1a85,0x1aad,0x1ad4,0x1afc,0x1b23,0x1b4c,0x1b73,0x1b9c,0x1bc4,0x1bec,0x1c14,0x1c3b,0x1c64,0x1c8c,
0x1cb3,0x1cdb,0x1d03,0x1d2b,0x1d53,0x1d78,0x1d9c,0x1dbf,0x1de2,0x1e0a,0x1e32,0x1e5b,0x1e81,0x1eaa,0x1ed4,
0x1f01,0x1f31,0x1f5a,0x1f83,0x1faa,0x1fd1,0x1ff9,0x2022,0x204a,0x2071,0x2099,0x20c1,0x20e9,0x2111,0x2139,
0x2160,0x2188,0x21b0,0x21d9,0x2200,0x2228,0x2250,0x2279,0x22a2,0x22ca,0x22f2,0x2318,0x2340,0x2366,0x238b,
0x23b4,0x23e0,0x2407,0x2430,0x2458,0x2480,0x24a8,0x24d0,0x24f7,0x251e,0x2546,0x256e,0x2596,0x25bd,0x25e2,
0x260c,0x2635,0x265d,0x2685,0x26ac,0x26d4,0x26fd,0x2725,0x274d,0x2775,0x279d,0x27c4,0x27ed,0x2814,0x283c,
0x2864,0x288d,0x28b4,0x28dd,0x2904,0x292d,0x2955,0x297c,0x29a3,0x29ca,0x29f3,0x2a1c,0x2a43,0x2a6b,0x2a91,
0x2ab9,0x2ae2,0x2b0b,0x2b32,0x2b5c,0x2b82,0x2ba9,0x2bd0,0x2bfa,0x2c20,0x2c4a,0x2c72,0x2c9b,0x2cc1,0x2cea,
0x2d10,0x2d38,0x2d61,0x2d8a,0x2db0,0x2dd9,0x2e00,0x2e29,0x2e50,0x2e78,0x2ea0,0x2ec8,0x2ef0,0x2f18,0x2f40,
0x2f68,0x2f8f,0x2fb7,0x2fde,0x3007,0x3030,0x3058,0x307e,0x30a5,0x30cd,0x30f5,0x311d,0x3144,0x316c,0x3194,
0x31bb,0x31e4,0x320d,0x3235,0x325d,0x3286,0x32ae,0x32d7,0x32fd,0x3323,0x334c,0x3373,0x339c,0x33c4,0x33ec,
0x3416,0x343c,0x3463,0x348c,0x34b4,0x34dc,0x3504,0x352d,0x3554,0x357c,0x35a2,0x35ca,0x35f1,0x361b,0x3645,
0x366a,0x3692,0x36ba,0x36e2,0x370a,0x3733,0x375a,0x3782,0x37ab,0x37d3,0x37f9,0x3820,0x3847,0x386e,0x3896,
0x38c0,0x38e9,0x3912,0x393a,0x3963,0x398a,0x39b1,0x39db,0x3a02,0x3a2a,0x3a52,0x3a79,0x3aa2,0x3aca,0x3af2,
0x3b1c,0x3b48,0x3b6c,0x3b91,0x3bb8,0x3be0,0x3c08,0x3c31,0x3c59,0x3c80,0x3ca8,0x3cd0,0x3cf8,0x3d1f,0x3d49,
0x3d70,0x3d96,0x3dbe,0x3de5,0x3e0e,0x3e36,0x3e5e,0x3e86,0x3eae,0x3ed6,0x3efe,0x3f26,0x3f4e,0x3f76,0x3f9f,
0x3fc7,0x3fee,0x4015,0x403d,0x4065,0x408d,0x40b4,0x40dc,0x4104,0x412c,0x4154,0x417d,0x41a5,0x41cc,0x41f3,
0x421b,0x4242,0x426b,0x4293,0x42bb,0x42e3,0x430b,0x4333,0x435a,0x4381,0x43aa,0x43d2,0x43f9,0x4422,0x444a,
0x4472,0x449a,0x44c2,0x44ea,0x4512,0x4539,0x4562,0x458a,0x45b1,0x45d8,0x4601,0x4629,0x4654,0x467b,0x46a1,
0x46c9,0x46f0,0x4718,0x4740,0x4768,0x4791,0x47ba,0x47e1,0x4808,0x482f,0x4857,0x487e,0x48a7,0x48d0,0x48f8,
0x4920,0x4947,0x496e,0x4996,0x49be,0x49e6,0x4a0c,0x4a35,0x4a5d,0x4a84,0x4aac,0x4ad5,0x4afd,0x4b23,0x4b49,
0x4b71,0x4b98,0x4bc0,0x4be6,0x4c0d,0x4c33,0x4c5d,0x4c82,0x4cad,0x4cd7};


static void input_isr ( void* context, alt_u32 id)
{
valz++;
IOWR_16DIRECT(INPUTCONTROLLER_INST_BASE, 0,0);
return;
}

int main()
{

IOWR_8DIRECT(INPUTCONTROLLER_INST_BASE, 0, 0);
IOWR_8DIRECT(INPUTCONTROLLER2_INST_BASE, 0, 0);
IOWR_8DIRECT(INPUTCONTROLLER3_INST_BASE, 0, 0);
IOWR_8DIRECT(INPUTCONTROLLER4_INST_BASE, 0, 0);
IOWR_8DIRECT(INPUTCONTROLLER5_INST_BASE, 0, 0);
IOWR_8DIRECT(INPUTCONTROLLER6_INST_BASE, 0, 0);

printf("main() started ");

alt_irq_register( INPUTCONTROLLER_INST_IRQ, NULL,(void*)irqhandler ); // register the irq

alt_irq_register( INPUTCONTROLLER2_INST_IRQ, NULL,(void*)irqhandler2 ); // register the irq

alt_irq_register( INPUTCONTROLLER3_INST_IRQ, NULL,(void*)irqhandler3 ); // register the irq

alt_irq_register( INPUTCONTROLLER4_INST_IRQ, NULL,(void*)irqhandler4 ); // register the irq

alt_irq_register( INPUTCONTROLLER5_INST_IRQ, NULL,(void*)irqhandler5 ); // register the irq

alt_irq_register( INPUTCONTROLLER6_INST_IRQ, NULL,(void*)irqhandler6 ); // register the irq

// value from the buffer may be displayed, no worries
printf("Hello from Nios II!");

int counter = 0;
printf("GO");

int flag = 0;
int val_buffer = val;
int key_buffer = floor(rand(5));

int score = 0;

while(counter<540){

if (flag!=1){
if (val>val_buffer){
if (key==key_buffer){
flag = 1;
score = score+1;
printf("SUCCESS: SCORE %d",score);}}}
// rajouter booleen pour afficher succes qu'une seule fois

if (beat_counter>=(beats[counter])*3){

if (flag == 0){
```

```
printf("FAIL: SCORE %d",score);}

flag = 0;
val_buffer = val;
key_buffer = floor(1+rand()%5);

printf("PRESS %d",key_buffer);
counter=counter+1;
}
}
printf("FINISHED!");


while(counter<541){

if(counter>0){
set_timer(0,beats[counter]*40 - beats[counter-1]*40 );}
else{
set_timer(0,beats[counter]*40);
}

launch_timer();
while(1){
if(timer_expired() ==1)
break;

}


if (flag == 0){
printf("FAIL");}


printf("[%d] %d ",counter, beats[counter]*40 );
counter=counter+1;
} return 0; }
```

VHDL source code

AWESOME_GUITAR.qpf

```
     # Copyright (C) 1991-2007 Altera Corporation
# Your use of Altera Corporation's design tools, logic functions
# and other software and tools, and its AMPP partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Altera Program License
# Subscription Agreement, Altera MegaCore Function License
# Agreement, or other applicable license agreement, including,
# without limitation, that your use is for the sole purpose of
# programming logic devices manufactured by Altera and sold by
# Altera or its authorized distributors.  Please refer to the
# applicable agreement for further details.

     QUARTUS_VERSION = "7.2"
DATE = "15:48:30 April 05, 2012"

     # Revisions

     PROJECT_REVISION = "AWESOME_GUITAR_TOP"
```

AWESOME_GUITAR.qws

```
     [ProjectWorkspace]
ptn_Child1=Frames

     [ProjectWorkspace.Frames]
ptn_Child1=ChildFrames

     [ProjectWorkspace.Frames.ChildFrames]
ptn_Child1=Document-0
ptn_Child2=Document-1
ptn_Child3=Document-2
ptn_Child4=Document-3

     [ProjectWorkspace.Frames.ChildFrames.Document-3]
ptn_Child1=ViewFrame-0

     [ProjectWorkspace.Frames.ChildFrames.Document-3.ViewFrame-0]
DocPathName=guitar_top.vhd
DocumentCLSID={ca385d57-a4c7-11d1-a098-0020affa43f2}
IsChildFrameDetached=False
IsActiveChildFrame=False
ptn_Child1=StateMap


     [ProjectWorkspace.Frames.ChildFrames.Document-3.ViewFrame-0.StateMap]

AFC_IN_REPORT=False
```

AWESOME_GUITAR_TOP.dpf

```
<?xml version="1.0" encoding="UTF-8"?>


<pin_planner>

<pin_info>

</pin_info>

<buses>

</buses>

<group_file_association>

</group_file_association>

<pin_planner_file_specifies>

</pin_planner_file_specifies>

</pin_planner>
```

AWESOME_GUITAR_TOP.jdi

This file has not been modified from its original version.

AWESOME_GUITAR_TOP.qsf

This file has not been modified from its original version.

AWESOME_GUITAR_TOP.sof

This file has not been modified from its original version.

counter.vhd

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is

port
(
-Synchronous reset
reset :  in std_logic;
clk:  in std_logic;
do:  out unsigned(31 downto 0);
di :  in unsigned (31 downto 0)
);

end counter;


architecture ar1 of counter is
signal value :  unsigned(31 downto 0);
begin
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
value <= di;
else
value <= value -1;
end if;
end if;
end process;
do <= value; -copy value to the output
end ar1;
```

cpu.ocp

This file has not been modified from its original version.

cpu.vhd

This file has not been modified from its original version.

cpu_jtag_debug_module.vhd

This file has not been modified from its original version.

cpu_jtag_debug_module_wrapper.vhd

This file has not been modified from its original version.

cpu_ociram_default_contents.mif

This file has not been modified from its original version.

cpu_rf_ram.mif

This file has not been modified from its original version.

cpu_test_bench.vhd

This file has not been modified from its original version.

DebounceCounter.vhd

```
     -19 bit counter
-19 bits since it needs to count to 20 ms
- 20 ms = 500 000 * tclock *2
- and 2**19 = 524 288

     library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

     entity DebounceCounter is

     port
( -Synchronous reset
reset :  in std_logic;
clk:  in std_logic;
do:  out unsigned(18 downto 0)

     );

     end DebounceCounter;
architecture ar1 of DebounceCounter is
signal value :  unsigned(18 downto 0);
begin
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
value <= (others => '0');
else
value <= value + 1;
end if;
end if;
end process;
do <= value; -copy value to the output
end ar1;
```

debouncer.vhd

```
     -Debouncer module
-TODO include reset
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

     entity debouncer is

     port
(
```

```
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic

     );

     end debouncer;

     architecture ar1 of debouncer is
type states is (ZERO, ZERO_TO_ONE, ONE, ONE_TO_ZERO);
signal state, next_state :  states;
signal do :  unsigned(18 downto 0);
signal reset_counter:  std_logic; -to reset the counter
begin

     C0:  entity work.DebounceCounter port map (
clk => clk,
reset => reset_counter,
do => do);

     -FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
state <= ZERO;
else
state <= next_state;
end if;
end if;
end process;

     -Computation of the next state
process (state,key_in,do)
begin
next_state <= state;
case state is
-State ZERO
-Signal generated:  reset => 1, key_out => 0
when ZERO =>
reset_counter <= '1';
key_out <= '1';
if (key_in = '0')
then
next_state <= ZERO_TO_ONE;
end if;

     -State ZERO_TO_ONE
-Signal generated:  reset => 0, key_out => 0
when ZERO_TO_ONE =>
reset_counter <= '0';
key_out <= '1';
if (do >= 500_000)
then
next_state <= ONE;
end if;
-State ONE
-Signal generated:  reset => 1, key_out => 1
when ONE =>
reset_counter <= '1';
key_out <= '0';
if (key_in = '1')
then
next_state <= ONE_TO_ZERO;
end if;
-State ONE_TO_ZERO
-Signal generated:  reset => 0, key_out => 1
when ONE_TO_ZERO =>
reset_counter <= '0';
key_out <= '0';
if (do >= 500_000)
then
next_state <= ZERO;
end if;
end case;
end process;

     end ar1;
```

de2_i2c_av_config.v

```
     /*
* I2C bus control for initializing the audio and video chips on the DE2 board
*
* Adapted by Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
*/

     module de2_i2c_av_config( iCLK, iRST_N, I2C_SCLK, I2C_SDAT );

     // From host
```

```
     input iCLK;
input iRST_N;

     // I2C bus

     output I2C_SCLK;
inout I2C_SDAT;

     // Internal Registers/Wires
reg [15:0] mI2C_CLK_DIV;
reg [23:0] mI2C_DATA;
reg mI2C_CTRL_CLK;
reg mI2C_GO;
wire mI2C_END;
wire mI2C_ACK;
reg [15:0] LUT_DATA;
reg [5:0] LUT_INDEX;
reg [3:0] mSetup_ST;

     // Clock frequencies
parameter CLK_Freq = 50000000; // 50 MHz
parameter I2C_Freq = 20000; // 20 kHz

     parameter LUT_SIZE = 50;

     // Audio Data Index
parameter SET_LIN_L = 0;
parameter SET_LIN_R = 1;
parameter SET_HEAD_L = 2;
parameter SET_HEAD_R = 3;
parameter A_PATH_CTRL = 4;
parameter D_PATH_CTRL = 5;
parameter POWER_ON = 6;
parameter SET_FORMAT = 7;
parameter SAMPLE_CTRL = 8;
parameter SET_ACTIVE = 9;
// Video Data Index
parameter SET_VIDEO = 10;

     // I2C Control Clock
always (posedge iCLK or negedge iRST_N)
begin
if (!iRST_N)
begin
mI2C_CTRL_CLK <= 0;
mI2C_CLK_DIV <= 0;
end
else
begin
if ( mI2C_CLK_DIV < (CLK_Freq/I2C_Freq) )
mI2C_CLK_DIV <= mI2C_CLK_DIV+1;
else
begin
mI2C_CLK_DIV <= 0;
mI2C_CTRL_CLK <=  mI2C_CTRL_CLK;
end
end
end

     de2_i2c_controller u0 (
.CLOCK(mI2C_CTRL_CLK), // Controller Work Clock
.I2C_SCLK(I2C_SCLK), // I2C CLOCK
.I2C_SDAT(I2C_SDAT), // I2C DATA
.I2C_DATA(mI2C_DATA), // DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
.GO(mI2C_GO), // GO transfor
.END(mI2C_END), // END transfor
.ACK(mI2C_ACK), // ACK
.RESET(iRST_N)
);

     // Configuration control
always (posedge mI2C_CTRL_CLK or negedge iRST_N)
begin
if(!iRST_N)
begin
LUT_INDEX <= 0;
mSetup_ST <= 0;
mI2C_GO <= 0;
end
else
begin
if (LUT_INDEX<LUT_SIZE)
begin
case(mSetup_ST)
0:  begin
if(LUT_INDEX<SET_VIDEO)
mI2C_DATA <= 8'h34,LUT_DATA;
else
mI2C_DATA <= 8'h40,LUT_DATA;
mI2C_GO <= 1;
mSetup_ST <= 1;
end
1:  begin
if (mI2C_END)
begin
if (!mI2C_ACK)
mSetup_ST <= 2;
else
mSetup_ST <= 0;
```

```
mI2C_GO <= 0;
end
end
2:  begin
LUT_INDEX <= LUT_INDEX+1;
mSetup_ST <= 0;
end
endcase
end
end
end

     // Configuration data LUT
always
begin
case (LUT_INDEX)
// Audio Config Data
SET_LIN_L : LUT_DATA <= 16'h001A;
SET_LIN_R : LUT_DATA <= 16'h021A;
SET_HEAD_L : LUT_DATA <= 16'h047B;
SET_HEAD_R : LUT_DATA <= 16'h067B;
A_PATH_CTRL : LUT_DATA <= 16'h08F8;
D_PATH_CTRL : LUT_DATA <= 16'h0A06;
POWER_ON : LUT_DATA <= 16'h0C00;
SET_FORMAT : LUT_DATA <= 16'h0E01;
SAMPLE_CTRL : LUT_DATA <= 16'h1002;
SET_ACTIVE : LUT_DATA <= 16'h1201;
// Video Config Data
SET_VIDEO+0 :  LUT_DATA <= 16'h1500;
SET_VIDEO+1 :  LUT_DATA <= 16'h1741;
SET_VIDEO+2 :  LUT_DATA <= 16'h3a16;
SET_VIDEO+3 :  LUT_DATA <= 16'h5004;
SET_VIDEO+4 :  LUT_DATA <= 16'hc305;
SET_VIDEO+5 :  LUT_DATA <= 16'hc480;
SET_VIDEO+6 :  LUT_DATA <= 16'h0e80;
SET_VIDEO+7 :  LUT_DATA <= 16'h5020;
SET_VIDEO+8 :  LUT_DATA <= 16'h5218;
SET_VIDEO+9 :  LUT_DATA <= 16'h58ed;
SET_VIDEO+10 :  LUT_DATA <= 16'h77c5;
SET_VIDEO+11 :  LUT_DATA <= 16'h7c93;
SET_VIDEO+12 :  LUT_DATA <= 16'h7d00;
SET_VIDEO+13 :  LUT_DATA <= 16'hd048;
SET_VIDEO+14 :  LUT_DATA <= 16'hd5a0;
SET_VIDEO+15 :  LUT_DATA <= 16'hd7ea;
SET_VIDEO+16 :  LUT_DATA <= 16'he43e;
SET_VIDEO+17 :  LUT_DATA <= 16'hea0f;
SET_VIDEO+18 :  LUT_DATA <= 16'h3112;
SET_VIDEO+19 :  LUT_DATA <= 16'h3281;
SET_VIDEO+20 :  LUT_DATA <= 16'h3384;
SET_VIDEO+21 :  LUT_DATA <= 16'h37A0;
SET_VIDEO+22 :  LUT_DATA <= 16'he580;
SET_VIDEO+23 :  LUT_DATA <= 16'he603;
SET_VIDEO+24 :  LUT_DATA <= 16'he785;
SET_VIDEO+25 :  LUT_DATA <= 16'h5000;
SET_VIDEO+26 :  LUT_DATA <= 16'h5100;
SET_VIDEO+27 :  LUT_DATA <= 16'h0050;
SET_VIDEO+28 :  LUT_DATA <= 16'h1000;
SET_VIDEO+29 :  LUT_DATA <= 16'h0402;
SET_VIDEO+30 :  LUT_DATA <= 16'h0b00;
SET_VIDEO+31 :  LUT_DATA <= 16'h0a20;
SET_VIDEO+32 :  LUT_DATA <= 16'h1100;
SET_VIDEO+33 :  LUT_DATA <= 16'h2b00;
SET_VIDEO+34 :  LUT_DATA <= 16'h2c8c;
SET_VIDEO+35 :  LUT_DATA <= 16'h2df2;
SET_VIDEO+36 :  LUT_DATA <= 16'h2eee;
SET_VIDEO+37 :  LUT_DATA <= 16'h2ff4;
SET_VIDEO+38 :  LUT_DATA <= 16'h30d2;
SET_VIDEO+39 :  LUT_DATA <= 16'h0e05;
default :  LUT_DATA <= 16'hxxxx;
endcase
end

     endmodule
```

de2_i2c_controller.v

```
     // --------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------
//
// Permission:
//
// Terasic grants permission to use and modify this code for use
// in synthesis for all Terasic Development Boards and Altrea Development
// Kits made by Terasic.  Other use of this code, including the selling,
// duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
// This VHDL or Verilog source code is intended as a design reference
// which illustrates how these types of functions can be implemented.
// It is the user's responsibility to verify their design for
// consistency and functionality through the use of formal
// verification methods.  Terasic provides no warranty regarding the use
// or functionality of this code.
```

```
//
// --------------------------------------------
//
// Terasic Technologies Inc
// 356 Fu-Shin E. Rd Sec.  1.  JhuBei City,
// HsinChu County, Taiwan
// 302
//
// web:  http://www.terasic.com/
// email:  support@terasic.com
//
// --------------------------------------------
//
// Major Functions:i2c controller
//
// --------------------------------------------
//
// Revision History :
// --------------------------------------------
// Ver :| Author :| Mod.  Date :| Changes Made:
// V1.0 :| Joe Yang :| 05/07/10 :| Initial Revision
// --------------------------------------------
module de2_i2c_controller (
CLOCK,
I2C_SCLK, // I2C CLOCK
I2C_SDAT, // I2C DATA
I2C_DATA, // DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
GO, // GO transfor
END, // END transfor
W_R, // W_R
ACK, // ACK
RESET,
// TEST
SD_COUNTER,
SDO
);

     input CLOCK;
input [23:0] I2C_DATA;
input GO;
input RESET;
input W_R;
inout I2C_SDAT;
output I2C_SCLK;
output END;
output ACK;

     // TEST
output [5:0] SD_COUNTER;
output SDO;

     reg SDO;
reg SCLK;
reg END;
reg [23:0] SD;
reg [5:0] SD_COUNTER;

     wire I2C_SCLK = SCLK | (((SD_COUNTER >= 4) & (SD_COUNTER <= 30))?   CLOCK : 0);
wire I2C_SDAT = SDO ? 1'bz :  0;

     reg ACK1, ACK2, ACK3;
wire ACK = ACK1 | ACK2 | ACK3;

     //-I2C COUNTER

     always (negedge RESET or posedge CLOCK)
begin
if (!RESET)
SD_COUNTER = 6'b111111;
else
begin
if (GO == 0)
SD_COUNTER = 0;
else
if (SD_COUNTER < 6'b111111)
SD_COUNTER = SD_COUNTER + 1;
end
end

     always @(negedge RESET or posedge CLOCK )
begin
if (!RESET)
begin
SCLK = 1;
SDO = 1;
ACK1 = 0;
ACK2 = 0;
ACK3 = 0;
END = 1;
end
else
case (SD_COUNTER)
6'd0 :  begin ACK1 = 0; ACK2 = 0; ACK3 = 0; END = 0; SDO = 1; SCLK = 1; end

     // Start
6'd1 :  begin SD = I2C_DATA; SDO = 0; end
6'd2 :  SCLK = 0;
```

```
     // Slave Address
6'd3 :  SDO = SD[23];
6'd4 :  SDO = SD[22];
6'd5 :  SDO = SD[21];
6'd6 :  SDO = SD[20];
6'd7 :  SDO = SD[19];
6'd8 :  SDO = SD[18];
6'd9 :  SDO = SD[17];
6'd10 :  SDO = SD[16];
6'd11 :  SDO = 1'b1; //ACK

     // Sub-address
6'd12 :  begin SDO = SD[15]; ACK1 = I2C_SDAT; end
6'd13 :  SDO = SD[14];
6'd14 :  SDO = SD[13];
6'd15 :  SDO = SD[12];
6'd16 :  SDO = SD[11];
6'd17 :  SDO = SD[10];
6'd18 :  SDO = SD[9];
6'd19 :  SDO = SD[8];
6'd20 :  SDO = 1'b1; // ACK

     // Data
6'd21 :  begin SDO = SD[7]; ACK2 = I2C_SDAT; end
6'd22 :  SDO = SD[6];
6'd23 :  SDO = SD[5];
6'd24 :  SDO = SD[4];
6'd25 :  SDO = SD[3];
6'd26 :  SDO = SD[2];
6'd27 :  SDO = SD[1];
6'd28 :  SDO = SD[0];
6'd29 :  SDO = 1'b1; // ACK

     // Stop
6'd30 :  begin SDO = 1'b0; SCLK = 1'b0; ACK3 = I2C_SDAT; end
6'd31 :  SCLK = 1'b1;
6'd32 :  begin SDO = 1'b1; END = 1; end
endcase
end

     endmodule
```

de2_sram_controller.vhd

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

     entity de2_sram_controller is

     port (
signal chipselect :  in std_logic;
signal write, read :  in std_logic;
signal address :  in std_logic_vector(17 downto 0);
signal readdata :  out std_logic_vector(15 downto 0);
signal writedata :  in std_logic_vector(15 downto 0);
signal byteenable :  in std_logic_vector(1 downto 0);

     signal SRAM_DQ : inout std_logic_vector(15 downto 0);
signal SRAM_ADDR : out std_logic_vector(17 downto 0);
signal SRAM_UB_N, SRAM_LB_N : out std_logic;
signal SRAM_WE_N, SRAM_CE_N : out std_logic;
signal SRAM_OE_N : out std_logic
);

     end de2_sram_controller;

     architecture dp of de2_sram_controller is
begin

     SRAM_DQ <= writedata when write = '1' else (others => 'Z');
readdata <= SRAM_DQ;
SRAM_ADDR <= address;
SRAM_UB_N <= not byteenable(1);
SRAM_LB_N <= not byteenable(0);
SRAM_WE_N <= not write;
SRAM_CE_N <= not chipselect;
SRAM_OE_N <= not read;

     end dp;
```

de2_sram_controller_hw.tcl

```
     # TCL File Generated by Component Editor 7.2 on:
# Thu Apr 05 15:54:15 EDT 2012
# DO NOT MODIFY

     set_source_file "de2_sram_controller.vhd"
set_module "de2_sram_controller"
set_module_description ""
set_module_property "className" "de2_sram_controller"
set_module_property "group" ""
set_module_property "libraries" [ list "ieee.std_logic_1164.all" "std.standard.all" ]
set_module_property "synthesisFiles" "de2_sram_controller.vhd"
```

```
     # Module parameters

     # Interface export_0
add_interface "export_0" "conduit" "start" "asynchronous"
# Ports in interface export_0
add_port_to_interface "export_0" "SRAM_DQ" "export"
add_port_to_interface "export_0" "SRAM_ADDR" "export"
add_port_to_interface "export_0" "SRAM_UB_N" "export"
add_port_to_interface "export_0" "SRAM_LB_N" "export"
add_port_to_interface "export_0" "SRAM_WE_N" "export"
add_port_to_interface "export_0" "SRAM_OE_N" "export"
add_port_to_interface "export_0" "SRAM_CE_N" "export"


     # Interface avalon_slave_0

add_interface "avalon_slave_0" "avalon" "slave" "asynchronous"

set_interface_property "avalon_slave_0" "isNonVolatileStorage" "false"

set_interface_property "avalon_slave_0" "burstOnBurstBoundariesOnly" "false"

set_interface_property "avalon_slave_0" "readLatency" "0"

set_interface_property "avalon_slave_0" "holdTime" "0"

set_interface_property "avalon_slave_0" "printableDevice" "false"

set_interface_property "avalon_slave_0" "readWaitTime" "1"

set_interface_property "avalon_slave_0" "setupTime" "0"

set_interface_property "avalon_slave_0" "addressAlignment" "DYNAMIC"

set_interface_property "avalon_slave_0" "writeWaitTime" "0"

set_interface_property "avalon_slave_0" "timingUnits" "Cycles"

set_interface_property "avalon_slave_0" "minimumUninterruptedRunLength" "1"

set_interface_property "avalon_slave_0" "isMemoryDevice" "true"

set_interface_property "avalon_slave_0" "linewrapBursts" "false"

set_interface_property "avalon_slave_0" "maximumPendingReadTransactions" "0"

# Ports in interface avalon_slave_0 add_port_to_interface "avalon_slave_0" "chipselect" "chipselect"

add_port_to_interface "avalon_slave_0" "write" "write"

add_port_to_interface "avalon_slave_0" "read" "read"

add_port_to_interface "avalon_slave_0" "address" "address"

add_port_to_interface "avalon_slave_0" "readdata" "readdata"

add_port_to_interface "avalon_slave_0" "writedata" "writedata"

add_port_to_interface "avalon_slave_0" "byteenable" "byteenable"
```

de2_wm8731_audio.vhd

This file has not been modified from its original version.

guitar_top.vhd

```
     -
- DE2 top-level module
-
- Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
-
- From an original by Terasic Technology, Inc.
- (DE2_TOP.v, part of the DE2 system board CD supplied by Altera)
-

     library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

     entity guitar_top is

     port (
signal CLOCK_50 :  in std_logic;

     SRAM_DQ : inout std_logic_vector(15 downto 0);
SRAM_ADDR : out std_logic_vector(17 downto 0);
KEY : in std_logic_vector(3 downto 0);
SRAM_UB_N,
SRAM_LB_N,
SRAM_WE_N,
```

```
SRAM_CE_N,
SRAM_OE_N : out std_logic ;

     GPIO_0, - GPIO Connection 0
GPIO_1 :  inout std_logic_vector(35 downto 0) - GPIO Connection 1

     );

     end guitar_top;

     architecture datapath of guitar_top is

     signal reset_n :std_logic;
signal new_res:std_logic;
signal audio_clock :  unsigned(1 downto 0) := "00";
signal counter :  unsigned(15 downto 0);

     begin

     process (CLOCK_50)
begin
if rising_edge(CLOCK_50) then
if counter = x"ffff" then
reset_n <= '1';
else
reset_n <= '0';
counter <= counter + 1;
end if;
end if;
end process;

     process (CLOCK_50)
begin
if rising_edge(CLOCK_50) then
audio_clock <= audio_clock + "1";
end if;
end process;

     nios :  entity work.nios_system port map (
clk => CLOCK_50,
reset_n => reset_n,
SRAM_ADDR_from_the_sram => SRAM_ADDR,
SRAM_CE_N_from_the_sram => SRAM_CE_N,
SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
SRAM_LB_N_from_the_sram => SRAM_LB_N,
SRAM_OE_N_from_the_sram => SRAM_OE_N,
SRAM_UB_N_from_the_sram => SRAM_UB_N,
SRAM_WE_N_from_the_sram => SRAM_WE_N,
-SWITCH_1_to_the_InputController_inst => GPIO_0(0),
-SWITCH_2_to_the_InputController_inst => GPIO_0(1) ,
-SWITCH_3_to_the_InputController_inst => GPIO_0(2) ,
-SWITCH_4_to_the_InputController_inst => GPIO_0(3),
-SWITCH_5_to_the_InputController_inst => GPIO_0(4)
SWITCH_1_to_the_InputController_inst => KEY(0),
SWITCH_2_to_the_InputController_inst => '1' ,
SWITCH_3_to_the_InputController_inst => '1',
SWITCH_4_to_the_InputController_inst => '1',
SWITCH_5_to_the_InputController_inst => '1',
-SWITCH_1_to_the_InputController2_inst => GPIO_0(0),
-SWITCH_2_to_the_InputController2_inst => GPIO_0(1) ,
-SWITCH_3_to_the_InputController2_inst => GPIO_0(2) ,
-SWITCH_4_to_the_InputController2_inst => GPIO_0(3),
-SWITCH_5_to_the_InputController2_inst => GPIO_0(4)
SWITCH_1_to_the_InputController2_inst => '1' ,
SWITCH_2_to_the_InputController2_inst => KEY(1) ,
SWITCH_3_to_the_InputController2_inst => '1' ,
SWITCH_4_to_the_InputController2_inst => '1' ,
SWITCH_5_to_the_InputController2_inst => '1',
-SWITCH_1_to_the_InputController3_inst => GPIO_0(0),
-SWITCH_2_to_the_InputController3_inst => GPIO_0(1) ,
-SWITCH_3_to_the_InputController3_inst => GPIO_0(2) ,
-SWITCH_4_to_the_InputController3_inst => GPIO_0(3),
-SWITCH_5_to_the_InputController3_inst => GPIO_0(4)
SWITCH_1_to_the_InputController3_inst => '1' ,
SWITCH_2_to_the_InputController3_inst => '1' ,
SWITCH_3_to_the_InputController3_inst => KEY(2) ,
SWITCH_4_to_the_InputController3_inst => '1' ,
SWITCH_5_to_the_InputController3_inst => '1',
-SWITCH_1_to_the_InputController4_inst => GPIO_0(0),
-SWITCH_2_to_the_InputController4_inst => GPIO_0(1) ,
-SWITCH_3_to_the_InputController4_inst => GPIO_0(2) ,
-SWITCH_4_to_the_InputController4_inst => GPIO_0(3),
-SWITCH_5_to_the_InputController4_inst => GPIO_0(4)
SWITCH_1_to_the_InputController4_inst => '1' ,
SWITCH_2_to_the_InputController4_inst => '1' ,
SWITCH_3_to_the_InputController4_inst => '1' ,
SWITCH_4_to_the_InputController4_inst => KEY(3) ,
SWITCH_5_to_the_InputController4_inst => '1',
-SWITCH_1_to_the_InputController5_inst => GPIO_0(0),
-SWITCH_2_to_the_InputController5_inst => GPIO_0(1) ,
-SWITCH_3_to_the_InputController5_inst => GPIO_0(2) ,
-SWITCH_4_to_the_InputController5_inst => GPIO_0(3),
-SWITCH_5_to_the_InputController5_inst => GPIO_0(4)
SWITCH_1_to_the_InputController5_inst => '1' ,
SWITCH_2_to_the_InputController5_inst => '1' ,
SWITCH_3_to_the_InputController5_inst => '1' ,
SWITCH_4_to_the_InputController5_inst => '1' ,
SWITCH_5_to_the_InputController5_inst => '1'
```

```
);
     end datapath;
```

InputController.vhd

```
     library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity InputController is

port (
-Avalon
clk :  in std_logic;
reset_n :  in std_logic;
read :  in std_logic;
write :  in std_logic;
chipselect :  in std_logic;
address :  in unsigned(9 downto 0);
readdata :  out unsigned(15 downto 0);
writedata :  in unsigned(15 downto 0);
inter :  out std_logic;
-To switches
SWITCH_1 :  in std_logic;
SWITCH_2 :  in std_logic;
SWITCH_3 :  in std_logic;
SWITCH_4 :  in std_logic;
SWITCH_5 :  in std_logic
);
end InputController;

architecture rtl of InputController is

type states is (IDLE, PRESSED,WAITST);
signal state, next_state :  states;
signal we:std_Logic;
signal reset:std_logic;
signal counter:  integer;

- A bunch of signals for the button
- The button clicks are first debounced and then
- Pulsed
- The naming follow this convention:
- Initial SWITCH_1 ______-_-_--------_-_________
- Debounced DSWITCH_1 ___________----------_________
- Pulsed PSWITCH_1 ___________-_____________________

-Switch 1
signal DSWITCH_1:std_logic;
signal PSWITCH_1:std_logic;
```

```
-Switch 2
signal DSWITCH_2:std_logic;
signal PSWITCH_2:std_logic;
-Switch 3
signal DSWITCH_3:std_logic;
signal PSWITCH_3:std_logic;
-Switch 4
signal DSWITCH_4:std_logic;
signal PSWITCH_4:std_logic;
--Switch 5
signal DSWITCH_5:std_logic;
signal PSWITCH_5:std_logic;
-Debouncer
component debouncer is port(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic
);
end component debouncer;
-Pulser
component pulser is port(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic
);
end component pulser;
begin
-Reset
reset <= not reset_n;
-Write enable
we <= '1' when chipselect ='1' and write='1' else '0';
-Debouncers
D1:  debouncer port map(clk,SWITCH_1,DSWITCH_1,reset);
D2:  debouncer port map(clk,SWITCH_2,DSWITCH_2,reset);
D3:  debouncer port map(clk,SWITCH_3,DSWITCH_3,reset);
D4:  debouncer port map(clk,SWITCH_4,DSWITCH_4,reset);
D5:  debouncer port map(clk,SWITCH_5,DSWITCH_5,reset);
-Pulsers
P1:  pulser port map(clk,DSWITCH_1,PSWITCH_1,reset);
P2:  pulser port map(clk,DSWITCH_2,PSWITCH_2,reset);
P3:  pulser port map(clk,DSWITCH_3,PSWITCH_3,reset);
P4:  pulser port map(clk,DSWITCH_4,PSWITCH_4,reset);
P5:  pulser port map(clk,DSWITCH_5,PSWITCH_5,reset);

-FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
```

```
if( reset_n='0' )
then
state <= IDLE;
else
state <= next_state;
if (state = PRESSED) then
counter <=0;
else counter <=counter +1;
end if;

end if;
end if;
end process;

-Combinational process
process (state,SWITCH_1,we)
begin
-By default reset the interuption signal
inter <= '0';
next_state <= state;
case state is

-Waiting for a button click
when IDLE =>
inter <='0';
if (PSWITCH_1 = '0' or PSWITCH_2 = '0' or PSWITCH_3 = '0' or PSWITCH_4 = '0' or PSWITCH_5 = '0' )
then
next_state <= PRESSED;
end if;

-A button has been pressed
-Stay into this state until the interuption has been cleared
when PRESSED =>
inter <='1';
if (we='1')
then
next_state <= WAITST;
end if;

when WAITST =>
inter <='0';
if (counter > 30000)
then
next_state <= IDLE;
end if;

end case;

end process;
```

```
end rtl;
```

InputController_hw.tcl

This file has not been modified from its original version.

InputController_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController_inst;

     architecture europa of InputController_inst is
component InputController is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController, which is an e_instance
the_InputController :  InputController2
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
```

```
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

InputController2_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController2_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController2_inst;

     architecture europa of InputController2_inst is
component InputController2 is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController2;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin
```

```
     -the_InputController2, which is an e_instance
the_InputController2 :  InputController2
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

InputController3_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController3_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController3_inst;

     architecture europa of InputController2_inst is
component InputController3 is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
```

```
end component InputController3;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController3, which is an e_instance
the_InputController3 :  InputController3
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

InputController4_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController4_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController4_inst;

     architecture europa of InputController4_inst is
component InputController4 is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
```

```
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController4;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController4, which is an e_instance
the_InputController4 :  InputController4
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

InputController5_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController5_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController5_inst;

     architecture europa of InputController5_inst is
component InputController5 is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
```

```
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController5;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController5, which is an e_instance
the_InputController5 :  InputController5
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

InputController6_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity InputController6_inst is
port (
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity InputController6_inst;

     architecture europa of InputController6_inst is
component InputController6 is
port (
```

```
- inputs:
signal SWITCH_1 :  IN STD_LOGIC;
signal SWITCH_2 :  IN STD_LOGIC;
signal SWITCH_3 :  IN STD_LOGIC;
signal SWITCH_4 :  IN STD_LOGIC;
signal SWITCH_5 :  IN STD_LOGIC;
signal address :  IN STD_LOGIC_VECTOR (9 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal inter :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component InputController6;

     signal internal_inter :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_InputController6, which is an e_instance
the_InputController6 :  InputController2
port map(
inter => internal_inter,
readdata => internal_readdata,
SWITCH_1 => SWITCH_1,
SWITCH_2 => SWITCH_2,
SWITCH_3 => SWITCH_3,
SWITCH_4 => SWITCH_4,
SWITCH_5 => SWITCH_5,
address => address,
chipselect => chipselect,
clk => clk,
read => read,
reset_n => reset_n,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
inter <= internal_inter;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

jtag_uart.vhd

This file has not been modified from its original version.

nios_system.bsf

This file has not been modified from its original version.

nios_system.ptf

This file has not been modified from its original version.

nios_system.qip

```
     set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path)

/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/de2_sram_controller_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path)

/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/de2_sram_controller.vhd]

set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path)

/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/InputController_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path)

/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/InputController.vhd]

set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController2_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController2.vhd]
```

```
set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController3_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController3.vhd]

set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController4_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController4.vhd]

set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController5_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController5.vhd]

set_global_assignment -name SOURCE_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController6_hw.tcl]

set_global_assignment -name VHDL_FILE [file join $::quartus(qip_path) /home/user3/spring12/imf2108/Desktop/InputController6.vhd]
```

nios_system.sopc

This file has not been modified from its original version.

nios_system_generation_script

This file has not been modified from its original version.

nios_system_log.txt

```
     Altera SOPC Builder Version 7.20 Build 151
Copyright (c) 1999-2007 Altera Corporation.  All rights reserved.

     # 2012.05.23 22:23:07 (*) mk_custom_sdk starting
# 2012.05.23 22:23:07 (*) Reading project /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system.ptf.

     # 2012.05.23 22:23:07 (*) Finding all CPUs
# 2012.05.23 22:23:07 (*) Finding all available components
# 2012.05.23 22:23:07 (*) Reading /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/.sopc_builder/install.ptf

     # 2012.05.23 22:23:08 (*) Found 67 components

     # 2012.05.23 22:23:08 (*) Finding all peripherals

     # 2012.05.23 22:23:08 (*) Finding software components

     # 2012.05.23 22:23:09 (*) (Legacy SDK Generation Skipped)
# 2012.05.23 22:23:09 (*) (All TCL Script Generation Skipped)
# 2012.05.23 22:23:09 (*) (No Libraries Built)
# 2012.05.23 22:23:09 (*) (Contents Generation Skipped)
# 2012.05.23 22:23:09 (*) mk_custom_sdk finishing
# 2012.05.23 22:23:09 (*) Starting generation for system:  nios_system.

     .
.

     # 2012.05.23 22:23:10 (*) Running Generator Program for cpu

     # 2012.05.23 22:23:11 (*) Checking for plaintext license.
# 2012.05.23 22:23:11 (*) Couldn't query license setup in Quartus directory /opt/altera/altera7.2/quartus
# 2012.05.23 22:23:11 (*) Defaulting to contents of LM_LICENSE_FILE environment variable
# 2012.05.23 22:23:11 (*) Plaintext license not found.
# 2012.05.23 22:23:11 (*) Checking for encrypted license (non-evaluation).
# 2012.05.23 22:23:11 (*) Couldn't query license setup in Quartus directory /opt/altera/altera7.2/quartus
# 2012.05.23 22:23:11 (*) Defaulting to contents of LM_LICENSE_FILE environment variable
# 2012.05.23 22:23:11 (*) Encrypted license found.  SOF will not be time-limited.
# 2012.05.23 22:23:20 (*) Creating encrypted HDL

     # 2012.05.23 22:23:22 (*) Running Generator Program for jtag_uart

     # 2012.05.23 22:23:23 (*) Making arbitration and system (top) modules.

     # 2012.05.23 22:23:30 (*) Generating Quartus symbol for top level:  nios_system

     # 2012.05.23 22:23:30 (*) Generating Symbol /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system.bsf

     # 2012.05.23 22:23:30 (*) Creating command-line system-generation script:
/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system_generation_script

     # 2012.05.23 22:23:30 (*) Running setup for HDL simulator:  modelsim

     # 2012.05.23 22:23:30 (*) Setting up Quartus with nios_system_setup_quartus.tcl
/opt/altera/altera7.2/quartus/bin/quartus_sh -t nios_system_setup_quartus.tcl

     Info:  ************************************************************
Info:  Running Quartus II Shell
Info:  Version 7.2 Build 151 09/26/2007 SJ Full Version
Info:  Copyright (C) 1991-2007 Altera Corporation.  All rights reserved.
Info:  Your use of Altera Corporation's design tools, logic functions
Info:  and other software and tools, and its AMPP partner logic
Info:  functions, and any output files from any of the foregoing
Info:  (including device programming or simulation files), and any
Info:  associated documentation or information are expressly subject
```

```
Info:  to the terms and conditions of the Altera Program License
Info:  Subscription Agreement, Altera MegaCore Function License
Info:  Agreement, or other applicable license agreement, including,
Info:  without limitation, that your use is for the sole purpose of
Info:  programming logic devices manufactured by Altera and sold by
Info:  Altera or its authorized distributors.  Please refer to the
In
fo:  applicable agreement for further details.
Info:  Processing started:  Wed May 23 22:23:30 2012
Info:  Command:  quartus_sh -t nios_system_setup_quartus.tcl
Info:  Evaluation of Tcl script nios_system_setup_quartus.tcl was successful
Info:  Quartus II Shell was successful.  0 errors, 0 warnings
Info:  Processing ended:  Wed May 23 22:23:31 2012
Info:  Elapsed time:  00:00:01

     # 2012.05.23 22:23:31 (*) Completed generation for system:  nios_system.
# 2012.05.23 22:23:31 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
SOPC Builder database :  /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system.ptf
System HDL Model :  /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system.vhd
System Generation Script :  /home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/nios_system_generation_script

     # 2012.05.23 22:23:31 (*) SUCCESS: SYSTEM GENERATION COMPLETED.


     Press 'Exit' to exit.
```

nios_system.ptf.pre_generation_ptf

This file has not been modified from its original version.

nios_system_setup_quartus.tcl

```
     # Caution:  this file may be regenerated by SOPC Builder.  User edits will be lost.

project_open -current_revision "/home/user3/spring12/imf2108/embsyscolspr2012/WORK/AWESOME_GUITAR/AWESOME_GUITAR.qpf"

set_global_assignment -name VHDL_FILE altera_europa_support.vhd

project_close
```

pulser.vhd

```
     -Pulser generator
-When the input signal switches to 1
-it generates a pulse

     library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

     entity pulser is

     port
(
clk :  in std_logic;
key_in :  in std_logic;
key_out :  out std_logic;
reset :  in std_logic

     );

     end pulser ;

     architecture A1 of pulser is
type states is (ZERO, ONE_PULSE, ONE_STANDBY);
signal state, next_state :  states;
begin

     -FSM Strandard next_state => state
process(clk)
begin
if(rising_edge(clk)) then
if( reset='1' )
then
state <= ZERO;
else
state <= next_state;
end if;
end if;
end process;

     -Computation of the next state
process (state,key_in)
begin
next_state <= state;
case state is
-State ZERO
-Signal generated 0
when ZERO =>
```

```
key_out <= '1';
if (key_in = '0')
then
next_state <= ONE_PULSE;
end if;

     -State ONE_PULSE
-Signal generated 1
when ONE_PULSE =>
key_out <= '0';
next_state <= ONE_STANDBY;
-State ONE_STANDBY
-Signal generated:  -
when ONE_STANDBY =>
key_out <= '1';
if (key_in = '1')
then
next_state <= ZERO;
end if;
end case;
end process;
end A1;
```

sopc_builder_log.txt

This file has not been modified from its original version.

sram.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity sram is
port (
- inputs:
signal address :  IN STD_LOGIC_VECTOR (17 DOWNTO 0);
signal byteenable :  IN STD_LOGIC_VECTOR (1 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal SRAM_ADDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0);
signal SRAM_CE_N : OUT STD_LOGIC;
signal SRAM_DQ : INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
signal SRAM_LB_N : OUT STD_LOGIC;
signal SRAM_OE_N : OUT STD_LOGIC;
signal SRAM_UB_N : OUT STD_LOGIC;
signal SRAM_WE_N : OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity sram;

     architecture europa of sram is
component de2_sram_controller is
port (
- inputs:
signal address :  IN STD_LOGIC_VECTOR (17 DOWNTO 0);
signal byteenable :  IN STD_LOGIC_VECTOR (1 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal read :  IN STD_LOGIC;
signal write :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal SRAM_ADDR : OUT STD_LOGIC_VECTOR (17 DOWNTO 0);
signal SRAM_CE_N : OUT STD_LOGIC;
signal SRAM_DQ : INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
signal SRAM_LB_N : OUT STD_LOGIC;
signal SRAM_OE_N : OUT STD_LOGIC;
```

```
signal SRAM_UB_N : OUT STD_LOGIC;
signal SRAM_WE_N : OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component de2_sram_controller;

     signal internal_SRAM_ADDR : STD_LOGIC_VECTOR (17 DOWNTO 0);
signal internal_SRAM_CE_N : STD_LOGIC;
signal internal_SRAM_LB_N : STD_LOGIC;
signal internal_SRAM_OE_N : STD_LOGIC;
signal internal_SRAM_UB_N : STD_LOGIC;
signal internal_SRAM_WE_N : STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_de2_sram_controller, which is an e_instance
the_de2_sram_controller :  de2_sram_controller
port map(
SRAM_ADDR => internal_SRAM_ADDR,
SRAM_CE_N => internal_SRAM_CE_N,
SRAM_DQ => SRAM_DQ,
SRAM_LB_N => internal_SRAM_LB_N,
SRAM_OE_N => internal_SRAM_OE_N,
SRAM_UB_N => internal_SRAM_UB_N,
SRAM_WE_N => internal_SRAM_WE_N,
readdata => internal_readdata,
address => address,
byteenable => byteenable,
chipselect => chipselect,
read => read,
write => write,
writedata => writedata
);

     -vhdl renameroo for output signals
SRAM_ADDR <= internal_SRAM_ADDR;
-vhdl renameroo for output signals
SRAM_CE_N <= internal_SRAM_CE_N;
-vhdl renameroo for output signals
SRAM_LB_N <= internal_SRAM_LB_N;
-vhdl renameroo for output signals
SRAM_OE_N <= internal_SRAM_OE_N;
-vhdl renameroo for output signals
SRAM_UB_N <= internal_SRAM_UB_N;
-vhdl renameroo for output signals
SRAM_WE_N <= internal_SRAM_WE_N;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```

timer.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity timer is
port (
- inputs:
signal address :  IN STD_LOGIC_VECTOR (2 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal reset_n :  IN STD_LOGIC;
signal write_n :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal irq :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity timer;

     architecture europa of timer is
signal clk_en :  STD_LOGIC;
```

```
signal control_continuous :  STD_LOGIC;
signal control_interrupt_enable :  STD_LOGIC;
signal control_register :  STD_LOGIC_VECTOR (3 DOWNTO 0);
signal control_wr_strobe :  STD_LOGIC;
signal counter_is_running :  STD_LOGIC;
signal counter_is_zero :  STD_LOGIC;
signal counter_load_value :  STD_LOGIC_VECTOR (31 DOWNTO 0);
signal counter_snapshot :  STD_LOGIC_VECTOR (31 DOWNTO 0);
signal delayed_unxcounter_is_zeroxx0 :  STD_LOGIC;
signal do_start_counter :  STD_LOGIC;
signal do_stop_counter :  STD_LOGIC;
signal force_reload :  STD_LOGIC;
signal internal_counter :  STD_LOGIC_VECTOR (31 DOWNTO 0);
signal period_h_register :  STD_LOGIC_VECTOR (15 DOWNTO 0);
signal period_h_wr_strobe :  STD_LOGIC;
signal period_l_register :  STD_LOGIC_VECTOR (15 DOWNTO 0);
signal period_l_wr_strobe :  STD_LOGIC;
signal read_mux_out :  STD_LOGIC_VECTOR (15 DOWNTO 0);
signal snap_h_wr_strobe :  STD_LOGIC;
signal snap_l_wr_strobe :  STD_LOGIC;
signal snap_read_value :  STD_LOGIC_VECTOR (31 DOWNTO 0);
signal snap_strobe :  STD_LOGIC;
signal start_strobe :  STD_LOGIC;
signal status_wr_strobe :  STD_LOGIC;
signal stop_strobe :  STD_LOGIC;
signal timeout_event :  STD_LOGIC;
signal timeout_occurred :  STD_LOGIC;

     begin

     clk_en <= std_logic'('1');
process (clk, reset_n)
begin
if reset_n = '0' then
internal_counter <= std_logic_vector'("00000000000000000001100001101001");
elsif clk'event and clk = '1' then
if std_logic'((counter_is_running OR force_reload)) = '1' then
if std_logic'((counter_is_zero OR force_reload)) = '1' then
internal_counter <= counter_load_value;
else
internal_counter <= A_EXT (((std_logic_vector'("0") & (internal_counter))
- std_logic_vector'("00000000000000000000000000000001")), 32);
end if;
end if;
end if;

     end process;

     counter_is_zero <= to_std_logic((internal_counter =
std_logic_vector'("00000000000000000000000000000000")));
counter_load_value <= period_h_register & period_l_register;
process (clk, reset_n)
begin
if reset_n = '0' then
force_reload <= std_logic'('0');
elsif clk'event and clk = '1' then
if std_logic'(clk_en) = '1' then
force_reload <= period_h_wr_strobe OR period_l_wr_strobe;
end if;
end if;

     end process;

     do_start_counter <= start_strobe;
do_stop_counter <= ((stop_strobe) OR (force_reload)) OR
((counter_is_zero AND NOT control_continuous));
process (clk, reset_n)
begin
if reset_n = '0' then
counter_is_running <= std_logic'('0');
elsif clk'event and clk = '1' then
if std_logic'(clk_en) = '1' then
if std_logic'(do_start_counter) = '1' then
counter_is_running <= Vector_To_Std_Logic
(-SIGNED(std_logic_vector'("00000000000000000000000000000001")));
elsif std_logic'(do_stop_counter) = '1' then
counter_is_running <= std_logic'('0');
end if;
end if;
end if;

     end process;

     -delayed_unxcounter_is_zeroxx0, which is an e_register
process (clk, reset_n)
begin
if reset_n = '0' then
delayed_unxcounter_is_zeroxx0 <= std_logic'('0');
elsif clk'event and clk = '1' then
if std_logic'(clk_en) = '1' then
delayed_unxcounter_is_zeroxx0 <= counter_is_zero;
end if;
end if;

     end process;

     timeout_event <= (counter_is_zero) AND NOT (delayed_unxcounter_is_zeroxx0);
process (clk, reset_n)
begin
```

```
if reset_n = '0' then
timeout_occurred <= std_logic'('0');
elsif clk'event and clk = '1' then
if std_logic'(clk_en) = '1' then
if std_logic'(status_wr_strobe) = '1' then
timeout_occurred <= std_logic'('0');
elsif std_logic'(timeout_event) = '1' then
timeout_occurred <= Vector_To_Std_Logic
(-SIGNED(std_logic_vector'("00000000000000000000000000000001")));
end if;
end if;
end if;

     end process;

     irq <= timeout_occurred AND control_interrupt_enable;
-s1, which is an e_avalon_slave
read_mux_out <= ((((((A_REP(to_std_logic
((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'
("00000000000000000000000000000010")))), 16) AND period_l_register)) OR ((A_REP(to_std_logic
((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'
("00000000000000000000000000000011")))), 16) AND period_h_register))) OR ((A_REP(to_std_logic
((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000100")))), 16) AND snap_read_value(15 DOWNTO 0)))) OR
((A_REP(to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000101")))), 16) AND snap_read_value(31 DOWNTO 16)))) OR
((A_REP(to_std_logic
((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'
("00000000000000000000000000000001")))), 16) AND ( std_logic_vector'("000000000000")
& (control_register))))) OR ((A_REP(to_std_logic( (((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector' ("00000000000000000000000000000000")))), 16) AND
(std_logic_vector'("00000000000000")
& (Std_Logic_Vector'(A_ToStdLogicVector(counter_is_running)
& A_ToStdLogicVector(timeout_occurred))))));
process (clk, reset_n)
begin
if reset_n = '0' then
readdata <= std_logic_vector'("0000000000000000");
elsif clk'event and clk = '1' then
if std_logic'(clk_en) = '1' then
readdata <= read_mux_out;
end if;
end if;

     end process;

     period_l_wr_strobe <= (chipselect AND NOT write_n) AND to_std_logic
((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000010"))));
period_h_wr_strobe <= (chipselect AND NOT write_n) AND
to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000011"))));
process (clk, reset_n)
begin
if reset_n = '0' then
period_l_register <= std_logic_vector'("0001100001101001");
elsif clk'event and clk = '1' then
if std_logic'(period_l_wr_strobe) = '1' then
period_l_register <= writedata;
end if;
end if;

     end process;

     process (clk, reset_n)
begin
if reset_n = '0' then
period_h_register <= std_logic_vector'("0000000000000000");
elsif clk'event and clk = '1' then
if std_logic'(period_h_wr_strobe) = '1' then
period_h_register <= writedata;
end if;
end if;

     end process;

     snap_l_wr_strobe <= (chipselect AND NOT write_n) AND
to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000100"))));
snap_h_wr_strobe <= (chipselect AND NOT write_n) AND
to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000101"))));
snap_strobe <= snap_l_wr_strobe OR snap_h_wr_strobe;
process (clk, reset_n)
begin
if reset_n = '0' then
counter_snapshot <= std_logic_vector'("00000000000000000000000000000000");
elsif clk'event and clk = '1' then
if std_logic'(snap_strobe) = '1' then
counter_snapshot <= internal_counter;
end if;
end if;

     end process;
```

```
     snap_read_value <= counter_snapshot;
control_wr_strobe <= (chipselect AND NOT write_n) AND
to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000001"))));
process (clk, reset_n)
begin
if reset_n = '0' then
control_register <= std_logic_vector'("0000");
elsif clk'event and clk = '1' then
if std_logic'(control_wr_strobe) = '1' then
control_register <= writedata(3 DOWNTO 0);
end if;
end if;

     end process;

     stop_strobe <= writedata(3) AND control_wr_strobe;
start_strobe <= writedata(2) AND control_wr_strobe;
control_continuous <= control_register(1);
control_interrupt_enable <= control_register(0);
status_wr_strobe <= (chipselect AND NOT write_n) AND
to_std_logic((((std_logic_vector'("00000000000000000000000000000000")
& (address)) = std_logic_vector'("00000000000000000000000000000000"))));

     end europa;
```

timer.vhdl

```
     library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

     entity timer is

     port (
clk :  in std_logic;
reset_n :  in std_logic;
read :  in std_logic;
write :  in std_logic;
chipselect :  in std_logic;
address :  in unsigned(4 downto 0);
readdata :  out unsigned(15 downto 0);
writedata :  in unsigned(15 downto 0)
);
end timer;

     architecture rtl of timer is

     signal we,wl,wh,ws :  std_logic;
signal lo:  unsigned(15 downto 0);
signal hi:  unsigned(15 downto 0);
signal va :  unsigned (31 downto 0);
signal start:  std_logic;
signal re:std_logic;
signal count:std_logic;

     begin

     -Write signals
-Style inspired by http://www.amazon.com/Embedded-SoPC-Design-Processor-Examples/
dp/111800888X/ref=sr_1_3?s=books&ie=UTF8&qid=1332644132&sr=1-3
we <= '1' when chipselect = '1'and write ='1' else '0' ;
wl <= '1' when we ='1' and address = "00000" else '0' ;
wh <= '1' when we ='1' and address = "00001" else '0' ;
ws <= '1' when we ='1' and address = "00010" else '0' ;
re <= '1' when chipselect = '1' and read='1' else '0' ;
readdata <=
va(15 downto 0) when address="00000" and re ='1' else
va(31 downto 16) when address="00001" and re ='1' else
x"0000";

     process(clk)
begin
if rising_edge(clk) then

     va <= va;
start <= start;

     if reset_n='0' then
count <= '0' ;
start <='0';
end if;

     if wh = '1' then va(31 downto 16) <= writedata;
elsif wl = '1' then va(15 downto 0) <= writedata ;
elsif ws = '1' then start <= writedata(0);
else
if start ='1' then
count <= '1';
start <= '0' ;
end if;
if count ='1' then

     if va = x"00000000" then
count <= '0' ; start <= '0' ;
```

```
else
va <= va - 1;
end if;

     end if;
end if;
end if;
end process;

     end rtl;
```

timer_hw.tcl

```
     # TCL File Generated by Component Editor 7.2 on:
# Tue May 22 19:56:46 EDT 2012
# DO NOT MODIFY

     set_source_file "timer.vhd"
set_module "timer"
set_module_description ""
set_module_property "className" "timer"
set_module_property "group" ""
set_module_property "libraries" [ list "altera.altera_europa_support_lib.all" "ieee.std_logic_1164.all" "ieee.std_logic_arith.all"
"ieee.std_logic_unsigned.all" "std.standard.all" ]
set_module_property "synthesisFiles" "timer.vhd"

     # Module parameters

     # Interface clock
add_interface "clock" "clock" "sink" "asynchronous"
# Ports in interface clock
add_port_to_interface "clock" "clk" "clk"

     # Interface clock_source
add_interface "clock_source" "clock" "source" "clock"
# Ports in interface clock_source

     # Interface clock_sink
add_interface "clock_sink" "clock" "sink" "clock"
# Ports in interface clock_sink

     # Interface export_0
add_interface "export_0" "conduit" "start" "clock"
# Ports in interface export_0

     # Interface avalon_slave_0
add_interface "avalon_slave_0" "avalon" "slave" "clock"
set_interface_property "avalon_slave_0" "isNonVolatileStorage" "false"
set_interface_property "avalon_slave_0" "burstOnBurstBoundariesOnly" "false"
set_interface_property "avalon_slave_0" "readLatency" "0"
set_interface_property "avalon_slave_0" "holdTime" "0"
set_interface_property "avalon_slave_0" "printableDevice" "false"
set_interface_property "avalon_slave_0" "readWaitTime" "1"
set_interface_property "avalon_slave_0" "setupTime" "0"
set_interface_property "avalon_slave_0" "addressAlignment" "DYNAMIC"
set_interface_property "avalon_slave_0" "writeWaitTime" "0"
set_interface_property "avalon_slave_0" "timingUnits" "Cycles"
set_interface_property "avalon_slave_0" "minimumUninterruptedRunLength" "1"
set_interface_property "avalon_slave_0" "isMemoryDevice" "false"
set_interface_property "avalon_slave_0" "linewrapBursts" "false"
set_interface_property "avalon_slave_0" "maximumPendingReadTransactions" "0"
# Ports in interface avalon_slave_0
add_port_to_interface "avalon_slave_0" "address" "address"
add_port_to_interface "avalon_slave_0" "chipselect" "chipselect"
add_port_to_interface "avalon_slave_0" "write_n" "write_n"
add_port_to_interface "avalon_slave_0" "writedata" "writedata"
add_port_to_interface "avalon_slave_0" "readdata" "readdata"


     # Interface interrupt_sender

add_interface "interrupt_sender" "interrupt" "sender" "clock"

set_interface_property "interrupt_sender" "associatedAddressablePoint" "avalon_slave_0"

# Ports in interface interrupt_sender

add_port_to_interface "interrupt_sender" "irq" "irq"

add_port_to_interface "interrupt_sender" "reset_n" "irq_n"
```

timer_inst.vhd

```
     -Legal Notice:  (C)2007 Altera Corporation.  All rights reserved.  Your
-use of Altera Corporation's design tools, logic functions and other
-software and tools, and its AMPP partner logic functions, and any
-output files any of the foregoing (including device programming or
-simulation files), and any associated documentation or information are
-expressly subject to the terms and conditions of the Altera Program
-License Subscription Agreement or other applicable license agreement,
-including, without limitation, that your use is for the sole purpose
```

```
-of programming logic devices manufactured by Altera and sold by Altera
-or its authorized distributors.  Please refer to the applicable
-agreement for further details.

     - turn off superfluous VHDL processor warnings
- altera message_level Level1
- altera message_off 10034 10035 10036 10037 10230 10240 10030

     library altera;
use altera.altera_europa_support_lib.all;

     library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

     entity timer_inst is
port (
- inputs:
signal address :  IN STD_LOGIC_VECTOR (2 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal write_n :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal irq :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end entity timer_inst;

     architecture europa of timer_inst is
component timer is
port (
- inputs:
signal address :  IN STD_LOGIC_VECTOR (2 DOWNTO 0);
signal chipselect :  IN STD_LOGIC;
signal clk :  IN STD_LOGIC;
signal write_n :  IN STD_LOGIC;
signal writedata :  IN STD_LOGIC_VECTOR (15 DOWNTO 0);

     - outputs:
signal irq :  OUT STD_LOGIC;
signal readdata :  OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
);
end component timer;

     signal internal_irq :  STD_LOGIC;
signal internal_readdata :  STD_LOGIC_VECTOR (15 DOWNTO 0);

     begin

     -the_timer, which is an e_instance
the_timer :  timer
port map(
irq => internal_irq,
readdata => internal_readdata,
address => address,
chipselect => chipselect,
clk => clk,
write_n => write_n,
writedata => writedata
);

     -vhdl renameroo for output signals
irq <= internal_irq;
-vhdl renameroo for output signals
readdata <= internal_readdata;

     end europa;
```