My First Piano

-Final project report-

Part I. Specification

My First Piano, features an immediate-responding animation user interfaces, simulates the piano by using keypad as the keyboard of real piano and buzzer to make sounds. Besides, My First Piano has another function which is to record the sequence of pressed keys and play them again. Not only as a simulator, My First Piano is a music box as well. It can play the music we have written in. With features above, we want to give you a game like experience. Have fun!

Manual:

Key 1-D is for notes to play

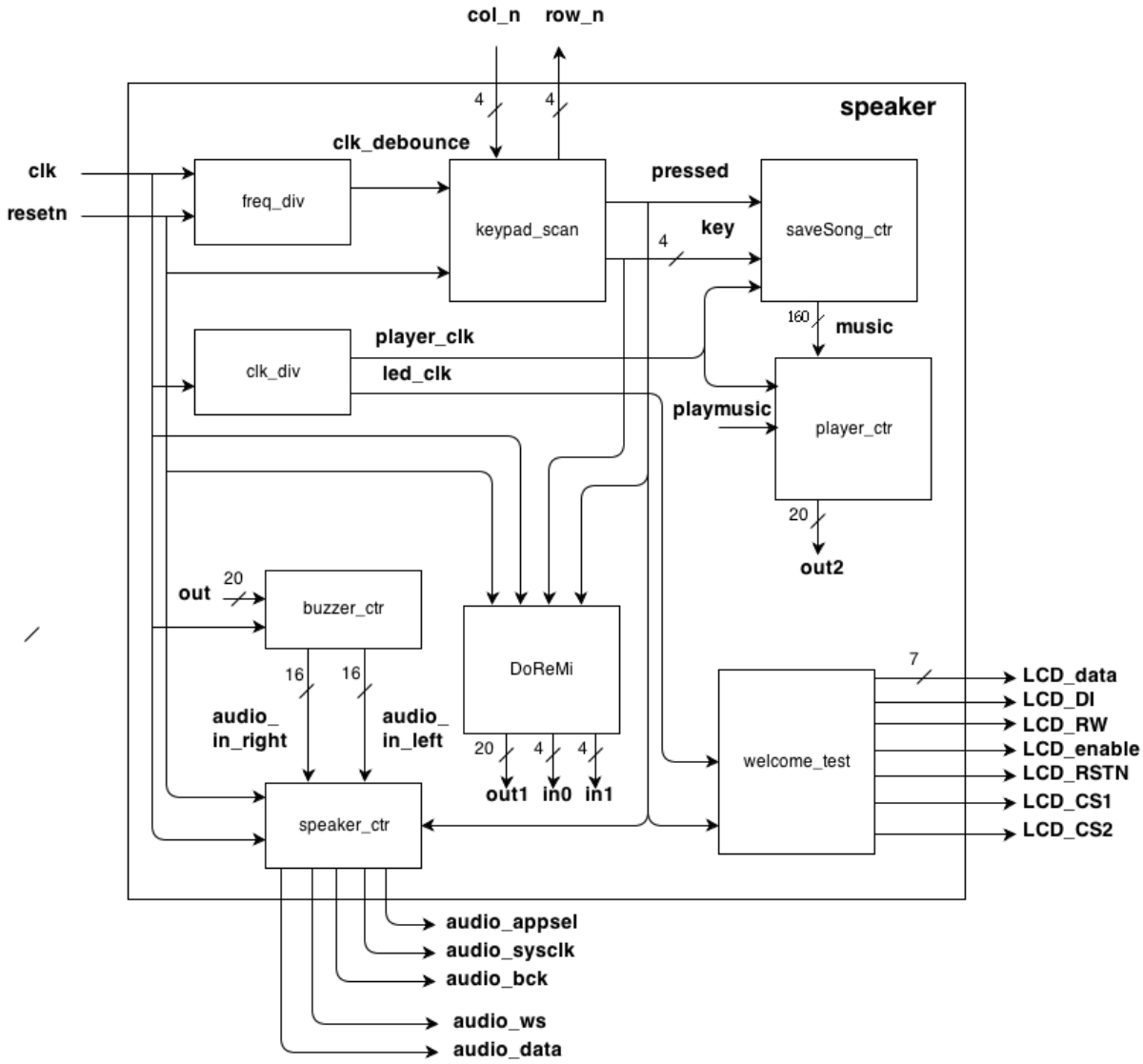
Key 0 is for starting to record

Key F is for playing the notes that have be recorded so far

Key E is for deleting the notes that have be recorded so far

Part II. Implementation details

1. Top block diagram




```
114
115  assign music =(!first) ?welcome_music : music_s;
116
```

```
128  wire playmusic;
129  assign playmusic = ((pressed==1 && key==4'hF)||!first)? 1 : 0;
```

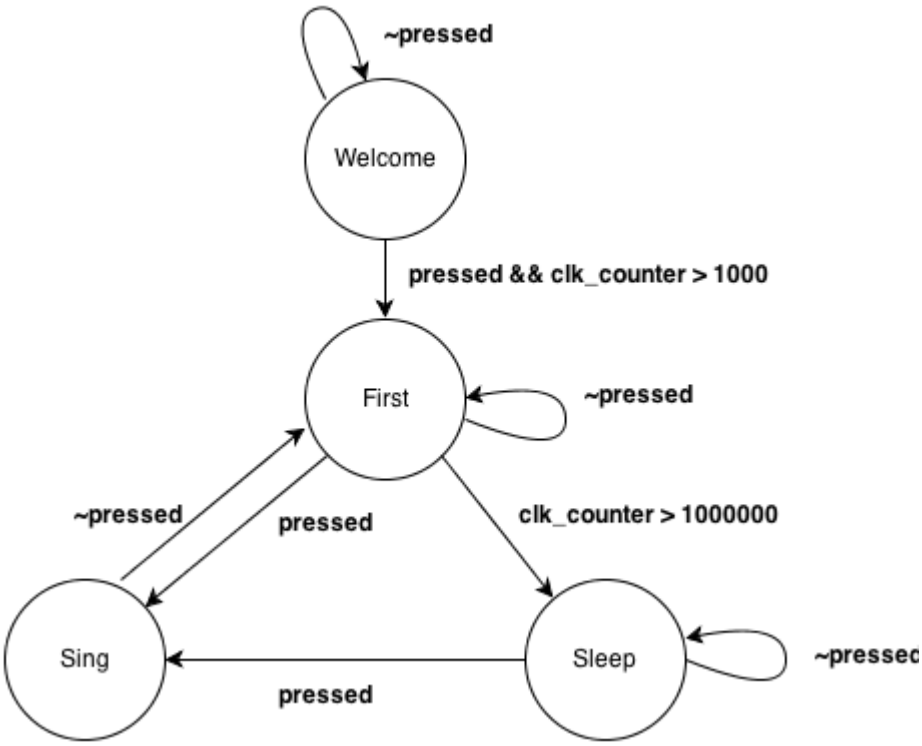
```
134  assign out = (out2==20'b0)?out1:out2;
```

We have to select the correct source music to play; if it's in welcome state, the welcome music has to be play; if it's not in welcome state, the music user

recording must be play.

In additional, We still have to decide which note has to play. There are two candidate: note user is playing and the note that player_control outputs . If player_control is output nothing, we play the note that user is playing.

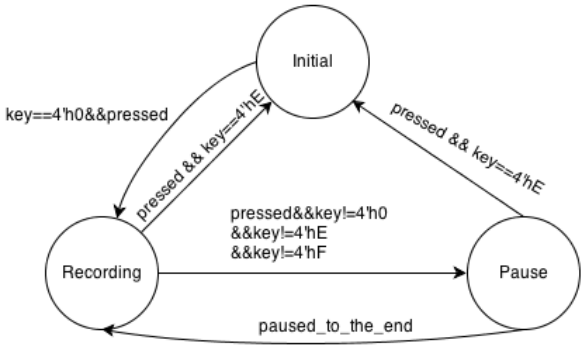
2. Image state transition diagram



Welcome	First
	
Sing	**Sleep**
	

3. Save songs

State diagram:



The saveSongs_control has three states; the default state is Initial, it will turn to Recording when user presses “0”, and start to record what buttons on keyboard user has pressed.

The music is recorded in reg [159:0] and four bit is for one note, so it can record maximum 40 notes.

When it is in Recording state, music_next = {key,music[159:4]}, the data structure of it is implemented as queue. The tail entry is music[159:156] and the head entry is music[3:0].

Also, it needs to turn to Pause state if the music is updated because we have to avoid repeating recording the same button for many times. We set a parameter ”p_delay” and a counter “pause” which is be added 1 every clock cycle time, paused_to_the_end is 1 when pause==p_delay. And the music is as output connected to player_control for whenever user want to know what notes he/she has recorded.

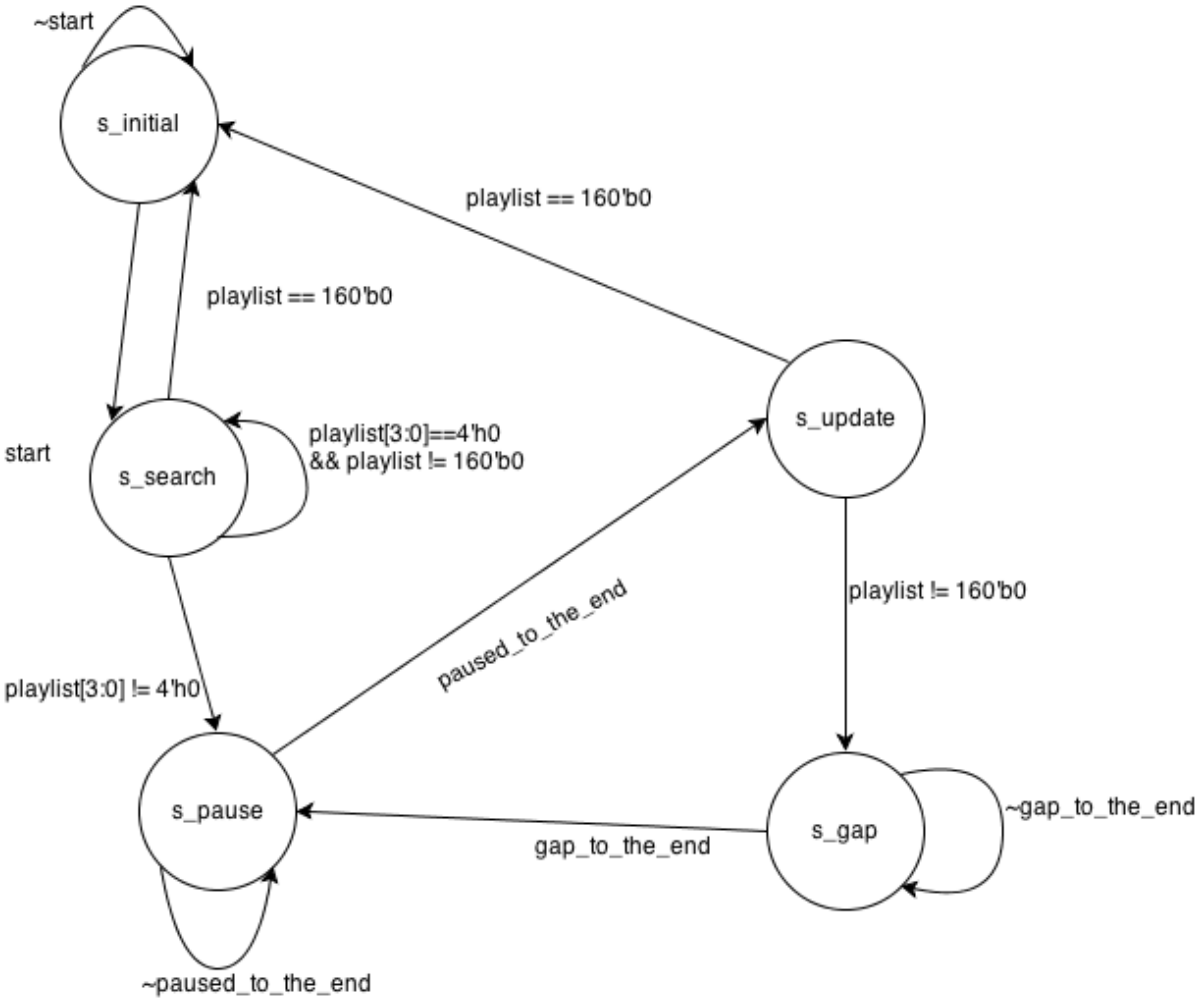
4. DoReMi

```
36  always@(pressed)
37  begin
38  if(pressed)
39  begin
40  case(in)
41     4'h0: out = 0;/*begin out = 20'd90909; in0=4'd13; end*/
42     4'h0: out = 0; /* begin out = 20'd81632; in0=4'd12; end*/
43     4'h1: out = 20'd76628;
44     4'h2: out = 20'd68259;
45     4'h3: out = 20'd60606;
46     4'h4: out = 20'd57306;
47     4'h5: out = 20'd51020;
48     4'h6: out = 20'd45454;
49     4'h7: out = 20'd40485;
50     4'h8: out = 20'd38167;
51     4'h9: out = 20'd34013;
52     4'hA: out = 20'd30303;
53     4'hB: out = 20'd28653;
54     4'hC: out = 20'd25510;
55     4'hD: out = 20'd22727;
56     4'hE: out = 20'd20242;
57
58     default: out = 20'd0;
59  endcase
60  end else out = 20'd0;
61
62  end
```

The out is connected to buzzer_contro(implement as clk_div)l as the divider of clock. Take 4’h6 for example, because the frequency of “La” is 440 Hz that is the divider of it must be (40M/440)/2 ~=. 45454

5. Player Control

player_control state diagram



The player_control has four states. The default state is s_initial. In s_initial, reg[159:0]playlist_next was set to input [159:0]music(generated by save songs or welcome music we writing in speaker module). When the input start is 1, the state will turn to s_search to find the first one note that is not 4'h0 from right to left. And when the note be found, the state will turn to s_pause from s_search to play the note at left most in playlist); when pause_to_end(It's implemented in the same way to paused_to_the_end in saveSongs_control) is 1, the s_pause will turn to s_update to update the playlist (dequeue). And when playlist is not empty, s_update will turn to s_gap to play nothing for a short time.(Or the music will be continuous and will sounds strange!) The state will turn to s_pause again from s_gap when gap_to_the_end is 1.

Part III. Issues and problems worthy of note and how you solved them

When key pressed, it takes some time (delay) for LCD to react to.

@ welcome_test module, which reponds for LCD display

```
30      reg [3:0]   IMAGE, IMAGE_NEXT;
31      reg [7:0]   PATTERN;
32      reg [9:0]   INDEX, INDEX_NEXT;
33      reg [14:0] PAUSE_TIME, PAUSE_TIME_NEXT;//15
34      reg [19:0] clk_counter;
```

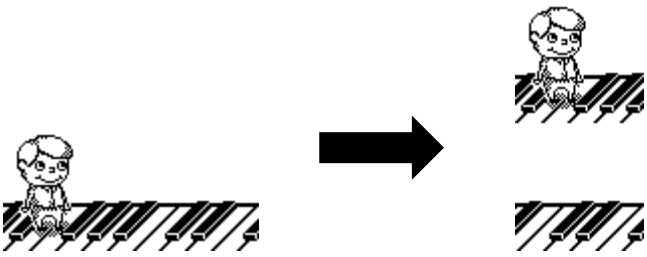
To address the delay, I shorten the length of *PAUSE_TIME* and *PAUSE_TIME_NEXT,* both 16 bit-long originally. However, it results that the change between two images faster (ex. The boy winks faster). The Issue becomes to find the balance compromising between reducing-delay and speed-up images changing.

After some tests, the length of 15 bits is the best one.

How to display full-screen images

Prepare:

I draw the image with painter and split the full screen image into two halves, like this



And using a program from the net, called “LCDAssistant”, to transfer the .bmp into .txt containing hex array as below

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0xC0, 0x40, 0xC0, 0x40, 0xC0, 0x40, 0x40, 0x40, 0x40, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

Then using the c# program read the text file and output a Verilog format, like this

```
10'd0 : PATTERN = 8'h00; 10'd1 : PATTERN = 8'h00; 10'd2 : PATTERN = 8'h00; 10'd3 : PATTERN = 8'h00;
'h00; 10'd40 : PATTERN = 8'h00; 10'd41 : PATTERN = 8'h00; 10'd42 : PATTERN = 8'h00; 10'd43 : PATTERN
10'd64 : PATTERN = 8'h00; 10'd65 : PATTERN = 8'h00; 10'd66 : PATTERN = 8'h00; 10'd67 : PATTERN = 8'h
: PATTERN = 8'h00; 10'd104 : PATTERN = 8'h00; 10'd105 : PATTERN = 8'h00; 10'd106 : PATTERN = 8'h00;
```

Verilog:

At the very beginning of *welcome_test,* the initial value of *LCD_CS1* and *LCD_CS2*

are both 1. And to toggle *LCD_CS1* and *LCD_CS2* I declare *LCD_CS1_NEXT* and *LCD_CS2_NEXT*.

```
16      output reg LCD_CS1=1;
17      output reg LCD_CS2=1;
```

```
23      reg LCD_CS1_NEXT, LCD_CS2_NEXT;
```

Every time get into *Copy_Image* state, let the left half screen light.

```
180             Copy_Image: begin // write image pattern into LCD RAM
181                 if (START) begin
182
183                     NEW_PAGE_NEXT = 1'b1;
184                     X_PAGE_NEXT = 0;  // image initial X address
185                     //Y_NEXT = (Y+1)%64; // image initial Y address
186                     PAGE_COUNTER_NEXT = 0;
187                     COL_COUNTER_NEXT = 0;
188
189                         LCD_CS1_NEXT = 1'b1;
190                         LCD_CS2_NEXT = 1'b0;
191                 end else if (NEW_PAGE) begin
```

Also at *Copy_Image* state, if

the page is the last page of the right half → the image is done

the page is the last page of the left half → toggle *LCD_CS1* and *LCD_CS2* to draw the rest right half

not above → draw next page

```
223                 if (PAGE_COUNTER == 7 && LCD_CS2 == 1) begin
224                     // last page of image
225                     STATE_NEXT = Pause;
226                     end else if(PAGE_COUNTER == 7 && LCD_CS1 == 1) be
227                     LCD_CS1_NEXT = 1'b0;
228                     LCD_CS2_NEXT = 1'b1;
229                     PAGE_COUNTER_NEXT = 0;
230                     COL_COUNTER_NEXT = 0;
231                     NEW_PAGE_NEXT = 1'b1;
232                     X_PAGE_NEXT = 0;
233                     end else begin
234                     // prepare to change page
235                     X_PAGE_NEXT = X_PAGE + 1;
236                     NEW_PAGE_NEXT = 1'b1;
237                     PAGE_COUNTER_NEXT = PAGE_COUNTER + 1;
238                     COL_COUNTER_NEXT = 0;
239                     end
240             end
```

When the program initialized on the board,
it comes a sudden rise of *pressed* signal

The sudden rise of *pressed* will result skipping the IMAGE_STATE “Welcome” and the welcome image won’t be displayed.

To resolve this question, I use a clock counter. Only when clock counter greater tha n 10000 can user press any key to start the game.

```
249             //image controller
250             case(IMAGE_STATE)
251                Welcome:begin // 4'b0000 - 4'b0001
252                      if(IMAGE == 4'b0000)IMAGE_NEXT = 4'b0001;
253                      else IMAGE_NEXT = 4'b0000;
254
255                      if(pressed && clk_counter > 10000)  IMAGE_STATE_NEXT =
256                      // prevent the sudden high of pressed
257                      else IMAGE_STATE_NEXT = Welcome;
258                end
```

Index not sufficient for full screen

The original length and hex representation of *INDEX* and *INDEX_NEXT* is not sufficient for full screen image. Thus, I use longer length and decimal for *INDEX* and *INDEX_NEXT*

```
29     reg [6:0]   INDEX, INDEX_NEXT;
```

```
32     reg [9:0]   INDEX, INDEX_NEXT;
```

And

```
case (INDEX)
  8'h00  :  PATTERN = 8'hC0; // upper half of image, wid = 32
  8'h01  :  PATTERN = 8'h20; // NOT SINGING
  8'h02  :  PATTERN = 8'h10; //1110_0000
  8'h03  :  PATTERN = 8'h08;
```

```
10'd0 : PATTERN = 8'h00; 10'd1 : PATTERN = 8'h00; 10'd2 : PATTERN
'h00; 10'd40 : PATTERN = 8'h00; 10'd41 : PATTERN = 8'h00; 10'd42
10'd64 : PATTERN = 8'h00; 10'd65 : PATTERN = 8'h00; 10'd66 : PAT
: PATTERN = 8'h00; 10'd104 : PATTERN = 8'h00; 10'd105 : PATTERN
```

Continuous Music sounds strange

We put gap between every two notes that makes it sounds more like music box!

When user only record few notes, play the record music will be delayed (because the most of notes in the right part of playlist are 0000)
To solve this problem, at first, we set a s_search state in Player Control to find the first note that is not 4'h0 .

It will save the same button many times that user only pressed once
Lengthen the delay time!