Musical FPGA Instrument

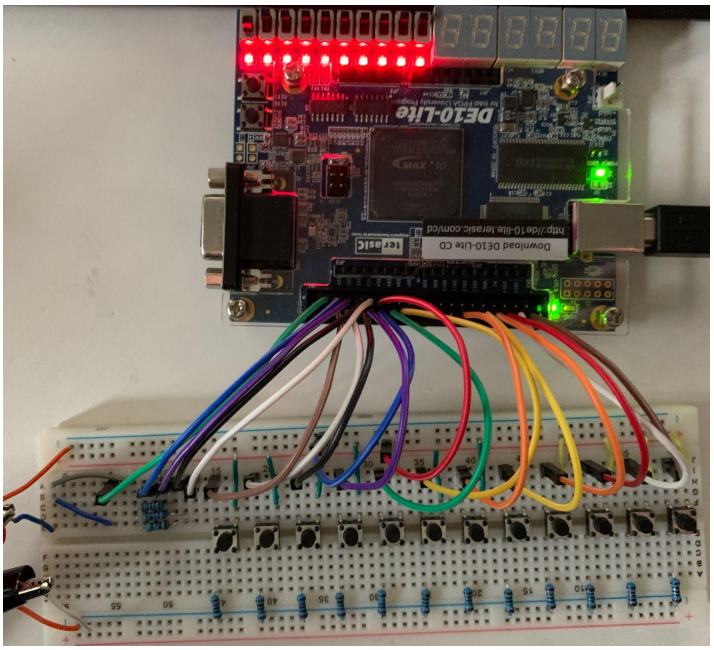
An EECS 3216 FPGA Project

Solomon Ukwosah (215672629)
Michael Sandrin (217144692)
Usman Elahi (215103682)

Table of Contents:

Introduction & Description:

The **Musical FPGA Instrument** is a *Verilog* based device that utilises features such as the components from the DE-10 Lite Board (Buttons, Switches, LED's, Seven-Segment Displays, System and ADC Clock) and external components (Breadboard, Speaker, 12-Buttons, Resistors).



Picture of the System (excluding the speaker)

Due to the use of an analog signal being produced by the DE-10 Lite board and VHDL (Verilog Hardware Description Language), the project would fall under **project type A**, as we will be converting the digital value inputted by the button array to then be translated into an analog signal to then be outputted via the speaker. Further into the report, the ways the signal is modified is shown below.

Using the arrange buttons, we can translate their positions to one of a piano, in which each respective key changes the octave of the sound wave (modifies the analog signal) and plays the sound accordingly. The change of notes is instantaneous and changes the tone of the sound coming out of the speaker. This creates an instrument for the user to play and make music with. In addition, a switch on the DE-10 Lite board can be used as a hard on/off switch to power the keyboard.

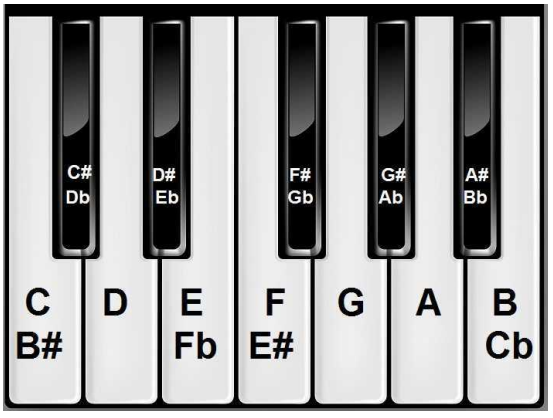


Diagram of Piano Notes

Unlike a piano, the device will allow the user to hold notes for as long as they feel, as well as combine the keys together to find an average pitch between all the keys pressed, to give the user even more options for possible instrumental sounds.

Hardware & Functions:

NOTE: DE-10 Lite Board is **upside down** in terms of the instruments orientation

Inputs:

Sound Buttons (12 Buttons):

The 12 buttons on the breadboard act as the playable notes for the instrument in which from left to right, each switch decreases in octave/pitch (a value that modifies the overall sound frequency). This design takes inspiration from a piano's 12 note scale design. The keys will be described later in the report.

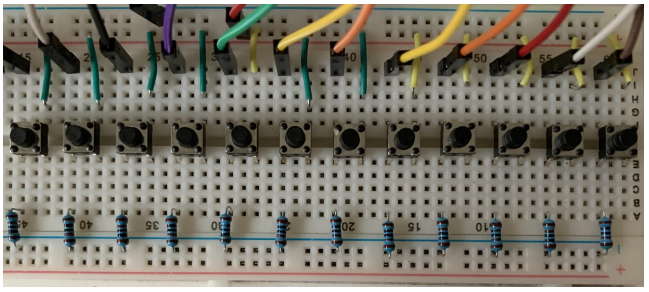


Image of 12 Playable Physical Keys

DE-10 Lite Buttons (Key 0 and 1):

The volume can be set with the two keys on the DE-10 Lite Board (Key 1: Increments Volume, Key 0: Decrements Volume), allowing the user to choose their volume, which chooses the correct resistor to output a sound signal to the speaker (least resistance = highest value).

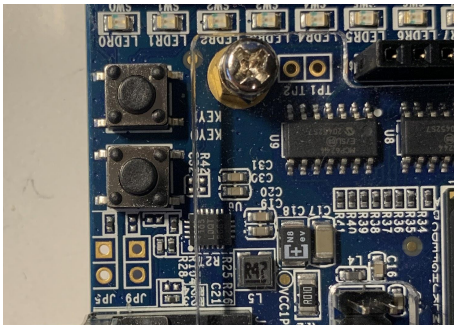


Image of Volume Keys

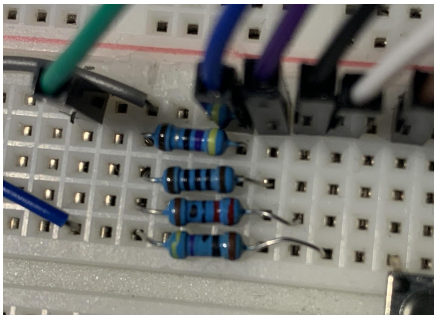


Image of Volume Control Resistors

DE-10 Lite Switch Array (4 Switches Used):

The switch array utilises **four** of the ten switches on the board, these being switches ***0***, ***1***, ***8***, and ***9***. *Switch 0* is used to turn the system on and off, *switch 1* is used to change between a *low* and *high* pitch sound on the piano. *Switch 8* allows the system to play a preset tune while *switch 9* swaps between the two presets. The specifics of their uses will be discussed later in the software section.

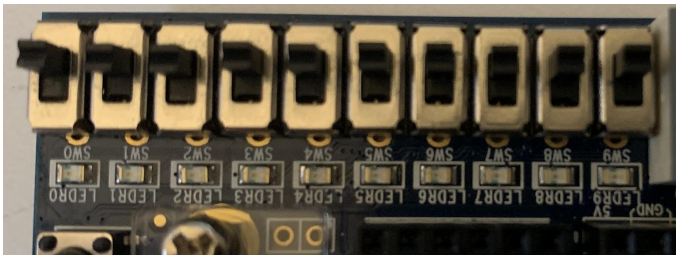


Image of Switch Array (Only uses SW0,1,8, &9)

Outputs:

Speaker:

The speaker is simply a device that takes the generated signal from the DE-10 Lite board based on the user's input. As mentioned previously, the speaker can output both high and low pitch sounds for the user to hear. The speaker used in the assignment is large but can be easily interchanged with smaller or larger speakers/buzzers as well.

DE-10 Lite LED Array (All 10 Used):

The LED-Array lights up according to the set volume using the ***two DE-10 Lite buttons*** (0% = 0 LED's on, 20% = 2 LED's on, 40% = 4 LED's on, 60% = 6 LED's on, 80% = 8 LED's on, 100% = 10 LED's on).

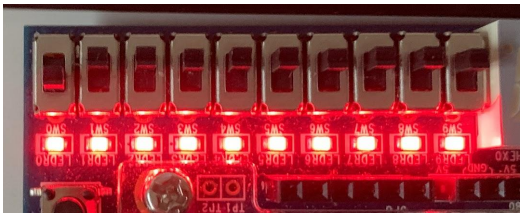


Image of LED Array at 100% Volume

DE-10 Lite Seven Segment Displays:

The Seven-Segment displays are used to notify the user of what the instrument is currently doing. The dot segment of the top right seven segment display turns on when the system on/off (SW0) switch is ***on***. The two left seven segments tell the user which pitch mode they have currently set (**high pitch = HI, low pitch = LO**). The rightmost seven segment displays the current note the user is playing, and if the top left LED of that seven segment display is on, the note is a *sharp note*.

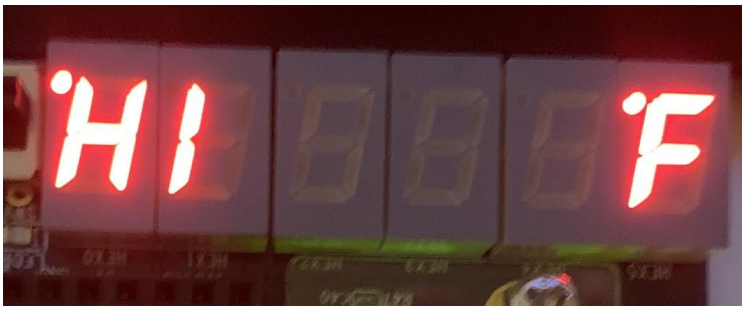


Image of 7-Seg Display when High Pitch F-Sharp is being played

GPIO Pins:

The GPIO (General Purpose Input/Output) pins are simply how the DE-10 Lite board works with the external devices on the breadboard. This includes the following:

- 12 Playable Notes/Keys and their 10k Ω resistors
- Volume Controlling Resistors
- Positive Contact (3.3V DC)
- Ground (GND)

With the use of the many GPIO pins for input and output connections, we decided to rotate the DE-10 Lite in our system to organise and improve the system's look and easy use.

Due to the versatility of the pins, we did not need to use any other external pins/connections to create the device.

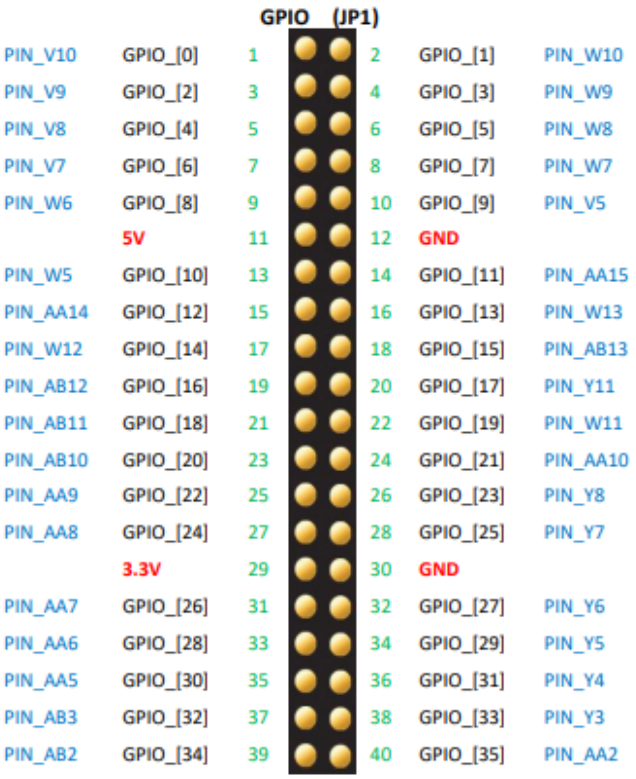


Diagram of GPIO on DE-10 Lite with Pin Locations

Software:

Code with Comments:

```
/**
Musical FPGA Instrument

By:
Michael Sandrin (217144692)
Solomon Ukwosah (215672629)
Usman Elahi (216103682)

*/

//Main Module
module Piano(clkHighPitch, clkLowPitch, pitchSw, onOffSw, tuneChoice, tuneOnOff, volumeControl,
speakerVolume, keys, LEDArray, pwrDisp, notesDisp, hiLoDispL, hiLoDispR);

//50 MHz Clock
input clkHighPitch;

//10 MHz Clock
input clkLowPitch;

//Switch to choose Pitch
input pitchSw;

//Clock to System
reg clk;

//Soft on/off switch (SW0)
input onOffSw;

input tuneOnOff;

input tuneChoice;

//Volume Control Buttons ([0] = Key0, [1] = Key1)
input [1:0] volumeControl;

//Register to allow volume changes one button press at a time
reg nextIncrement = 1'b0;

//Controls volume level (which volume resistor selected)
reg [2:0] volume = 3'b000;

//Each volume path ([0] being lowest volume, [4] being highest volume)
output [4:0] speakerVolume;
```

```
//The Corresponding LED Array above the switches
output reg [9:0] LEDArray = 10'b0000000000;

//Register for the selected octave
reg [8:0] octave;

//12 Input Keys (attached to the breadboard)
input [11:0] keys;

//Output via Seven Segment Displays
output pwrDisp;
output [7:0] notesDisp;
output [6:0] hiLoDispL;
output [6:0] hiLoDispR;

//Register Values for Seven Segment Displays
reg pwrSevenSegDot = 1'b1;
reg [7:0] noteSevenSeg = 8'b11111111;
reg [6:0] hiLoSevenSegL = 7'b1111111;
reg [6:0] hiLoSevenSegR = 7'b1111111;



//For the user to chose the pitch they want based on the clock chosen
//If onOffSw is on
always @(onOffSw) begin

 //If the Pitch is chosen to be on High Pitch (SW1 = 0) or a tune is being played
 if(~pitchSw || tuneOnOff) begin

        //Clock register value is the High Pitch Clock (50 MHz Clock)
        clk <= clkHighPitch;

 end else begin

 //Clock register value is the High Pitch Clock (50 MHz Clock)
 clk <= clkLowPitch;

 end

end

//Based on a clock cycle of the 50 MHz Clock
always @(posedge clk) begin

//If the On/Off soft switch is on
if(onOffSw) begin

//Seven Segment Dot turns on to signify the system is on
```

```
pwrSevenSegDot = 1'b0;

 //If the Pitch is chosen to be on High Pitch (SW1 = 0) or a tune is being played
 if(~pitchSw || tuneOnOff) begin

  //Sets the Pitch Display to output High Pitch (HI)
        hiLoSevenSegL = 7'b0001001;
        hiLoSevenSegR = 7'b1111001;

        end else begin

        //Sets the Pitch Display to output Low Pitch (LO)
        hiLoSevenSegL = 7'b1111000;
        hiLoSevenSegR = 7'b1000000;

        end

//If the Volume Control Increment Button is Pressed and the Volume is less than its maximum (Volume =
5 or 100%)
if(~volumeControl[1] & (volume < 3'b101) & (~nextIncrement)) begin

        //Volume Increments by 1
        volume = volume + 3'b001;

        //Sets the Value of next increment to 1 until the button is let go of
        nextIncrement = 1'b1;

//If the Volume Control Decrement Button is Pressed and the Volume is greater than its minimum
(Volume = 0 or 0%)
end else if(~volumeControl[0] & (volume > 3'b000) & (~nextIncrement)) begin

        //Volume Decrements by 1
        volume = volume - 3'b001;

        //Sets the Value of next increment to 1 until the button is let go of
        nextIncrement = 1'b1;


end else if (volumeControl[1] & volumeControl[0]) begin

        //Sets the Value of next increment to 0 as the button is let go of
  nextIncrement = 1'b0;

end

//For the when the keys are pressed
case(keys)

  //First Key is pressed, the octave is 361 (equivalent to D#Eb)
  12'b100000000000: begin
```

```
octave = 361; // D#/Eb

//Displays the Note
noteSevenSeg = 8'b00001100;

end

//Second Key is pressed, the octave is 341 (equivalent to E)
12'b010000000000: begin

octave = 341; // E

//Displays the Note
noteSevenSeg = 8'b10110000;

end

//Third Key is pressed, the octave is 322 (equivalent to F)
12'b001000000000: begin

octave = 322; // F

//Displays the Note
noteSevenSeg = 8'b10110001;

end

//Fourth Key is pressed, the octave is 303 (equivalent to F#Gb)
12'b000100000000: begin

octave = 303; // F#/Gb

//Displays the Note
noteSevenSeg = 8'b00110001;

end

//Fifth Key is pressed, the octave is 286 (equivalent to G)
12'b000010000000: begin

octave = 286; // G

 //Displays the Note
noteSevenSeg = 8'b11010000;

end

//Sixth Key is pressed, the octave is 270 (equivalent to G#/Ab)
12'b000001000000: begin
```

```
octave = 270; // G#/Ab

//Displays the Note
noteSevenSeg = 8'b01010000;

end

//Seventh Key is pressed, the octave is 511 (equivalent to A)
12'b000000100000: begin

octave = 511; // A

//Displays the Note
noteSevenSeg = 8'b10000001;

end

//Eigth Key is pressed, the octave is 482 (equivalent to A#/Bb)
12'b000000010000: begin

        octave = 482; // A#/Bb

        //Displays the Note
        noteSevenSeg = 8'b00000001;

        end

//Ninth Key is pressed, the octave is 455 (equivalent to B)
12'b000000001000: begin

octave = 455; // B

//Displays the Note
noteSevenSeg = 8'b10011000;

end

//Tenth Key is pressed, the octave is 430 (equivalent to C)
12'b000000000100: begin

octave = 430; // C

//Displays the Note
noteSevenSeg = 8'b11110000;

end

//Eleventh Key is pressed, the octave is 405 (equivalent to C#/Db)
12'b000000000010: begin
```

```
 octave = 405; // C#/Db

 //Displays the Note
 noteSevenSeg = 8'b01110000;

 end

 //Twelfth  is pressed, the octave is 383 (equivalent to D)
 12'b000000000001: begin

 octave = 383; // D

 //Displays the Note
 noteSevenSeg = 8'b10001100;

 end

 //Default sets octave to 0, producing no sound when used on music module
default begin

octave = 0; //No octave set, no sound played

//Displays the no Note
noteSevenSeg = 8'b11111111;

end

endcase

 //For the when the volumes are set
case(volume)
 //When Volume is set to 0%. all LED's are off
 3'b000: LEDArray = 10'b0000000000;

 //When Volume is set to 20%. the LED's 8 and 9 are on
 3'b001: LEDArray = 10'b1100000000;

 //When Volume is set to 40%. the LED's 6 to 9 are on
 3'b010: LEDArray = 10'b1111000000;

 //When Volume is set to 60%. the LED's 4 to 9 are on
 3'b011: LEDArray = 10'b1111110000;

 //When Volume is set to 80%. the LED's 2 to 9 are on
 3'b100: LEDArray = 10'b1111111100;

 //When Volume is set to 100%. all LED's are on
 3'b101: LEDArray = 10'b1111111111;
```

```
//When the system is off, the default would be all LED's off
default LEDArray = 10'b0000000000;

endcase

//When the on/off soft switch is set to off
end else begin

//LED's are off when the system is off
LEDArray = 10'b0000000000;

//Octave is set to 0 so no audio is played when the system is off
octave = 0;

//Sets all Seven Segments Off
pwrSevenSegDot = 1'b1;
noteSevenSeg = 8'b11111111;
hiLoSevenSegL = 7'b1111111;
hiLoSevenSegR = 7'b1111111;

end

end



 //Calls on the music module (for notes)
 keys notes(clk, onOffSw, octave, volume, tuneOnOff, tuneChoice, speakerVolume);

 //Displays the power seven segment dot according to the register value
 assign pwrDisp = pwrSevenSegDot;

 //Calls on the notes display module
 notesDisplay notedisp(noteSevenSeg, notesDisp);

 //Calls on the pitch display module
 hiLoDisplay hiLo(hiLoSevenSegL, hiLoSevenSegR, hiLoDispL, hiLoDispR);

endmodule



//Notes Display Module
module notesDisplay (noteSevenSeg, disp);

//Notes Seven Segment Values
input [7:0] noteSevenSeg;
```

```
//Output Seven Segment Display
output [7:0] disp;

//Sets Value for Segment 0 for the notes display
assign disp[0] = noteSevenSeg[0];

//Sets Value for Segment 1 for the notes display
assign disp[1] = noteSevenSeg[1];

//Sets Value for Segment 2 for the notes display
assign disp[2] = noteSevenSeg[2];

//Sets Value for Segment 3 for the notes display
assign disp[3] = noteSevenSeg[3];

//Sets Value for Segment 4 for the notes display
assign disp[4] = noteSevenSeg[4];

//Sets Value for Segment 5 for the notes display
assign disp[5] = noteSevenSeg[5];

//Sets Value for Segment 6 for the notes display
assign disp[6] = noteSevenSeg[6];

//Sets Value for Segment 7 for the notes display
assign disp[7] = noteSevenSeg[7];

endmodule




//Pitch Display Module
module hiLoDisplay (inL, inR, dispL, dispR);

//Left Seven Segment Values
input [6:0] inL;

//Right Seven Segment Values
input [6:0] inR;

//Left Output Seven Segment Display
output [6:0] dispL;

//Right Output Seven Segment Display
output [6:0] dispR;

//Sets Value for Segment 0 for the Left Pitch display
assign dispL[0] = inL[0];
```

```
//Sets Value for Segment 1 for the Left Pitch display
assign dispL[1] = inL[1];

//Sets Value for Segment 2 for the Left Pitch display
assign dispL[2] = inL[2];

//Sets Value for Segment 3 for the Left Pitch display
assign dispL[3] = inL[3];

//Sets Value for Segment 4 for the Left Pitch display
assign dispL[4] = inL[4];

//Sets Value for Segment 5 for the Left Pitch display
assign dispL[5] = inL[5];

//Sets Value for Segment 6 for the Left Pitch display
assign dispL[6] = inL[6];

//Sets Value for Segment 0 for the Right Pitch display
assign dispR[0] = inR[0];

//Sets Value for Segment 1 for the Right Pitch display
assign dispR[1] = inR[1];

//Sets Value for Segment 2 for the Right Pitch display
assign dispR[2] = inR[2];

//Sets Value for Segment 3 for the Right Pitch display
assign dispR[3] = inR[3];

//Sets Value for Segment 4 for the Right Pitch display
assign dispR[4] = inR[4];

//Sets Value for Segment 5 for the Right Pitch display
assign dispR[5] = inR[5];

//Sets Value for Segment 6 for the Right Pitch display
assign dispR[6] = inR[6];

endmodule



//Piano Keys Module
module keys(clk, onOffSw, octave, volume, tuneOnOff, sw, speakerVolume);

//50 MHz Clock
input clk;
```

```
input onOffSw;

//Register for the selected octave
input [8:0] octave;

//Controls volume level (which volume resistor selected)
input [2:0] volume;

//Each volume path ([0] being lowest volume, [4] being highest volume)
output reg [4:0] speakerVolume;

//Register for modifiable speaker value
reg speaker;

//Parameter used for the values of the inputs

//Register for divided clock
reg [24:0] clkdivider = 24'd0;

//Register to evenly count for the clock cycle in the wave
reg [14:0] counter;

//Input if tunes will be played
input tuneOnOff;

//Chooses which song
input sw;

//Register for the sound wave of the chosen sound
reg [30:0] sound;

//Tune values for the tunes chosen by the user
wire [7:0] tunes;

//Calls on the module to obtain the notes
music getnotes(.clk(clk), .sw(sw), .address(sound[29:22]), .notes(tunes));

//Wire to connect the octave for the current sound of the tunes
wire [2:0] tuneOctave;

//Wire to connect the notes for the current sound of the tunes
wire [3:0] notes;

//Calls on the module to obtain the proper octaves for the notes
divide_12 getnotes_octave(.num(tunes[5:0]), .quotient(tuneOctave), .rem(notes));

//Register value to divide the clock properly, which changes the sound of the current sound
reg [8:0] clkdiv;

always @*
```

```
case(notes)

            0: clkdiv = 9'd511;//A
            1: clkdiv = 9'd482;// A#/Bb
            2: clkdiv = 9'd455;//B
            3: clkdiv = 9'd430;//C
            4: clkdiv = 9'd405;// C#/Db
            5: clkdiv = 9'd383;//D
            6: clkdiv = 9'd361;// D#/Eb
            7: clkdiv = 9'd341;//E
            8: clkdiv = 9'd322;//F
            9: clkdiv = 9'd303;// F#/Gb
           10: clkdiv = 9'd286;//G
           11: clkdiv = 9'd270;// G#/Ab
           default: clkdiv = 9'd0;
endcase

reg [8:0] notes_count;
reg [7:0] octave_counter;


//Based on a clock cycle of the 50 MHz Clock
always @(posedge clk) begin

sound <= sound+31'd1;

if(~tuneOnOff) begin

            //If Octave is equal to 430 (C note)
  if(octave == (430)) begin

            //Clock Divider for the correct octave
            clkdivider = 25000000/430/2;

            //If Octave is equal to 430 (C#/Db note)
            end else if(octave == (405)) begin

            //Clock Divider for the correct octave
            clkdivider = 25000000/405/2;

            //If Octave is equal to 430 (D note)
            end else if(octave == (383)) begin

            //Clock Divider for the correct octave
            clkdivider = 25000000/383/2;

            //If Octave is equal to 430 (D#/Eb note)
            end else if(octave == (361)) begin

            //Clock Divider for the correct octave
```

```
clkdivider = 25000000/361/2;

//If Octave is equal to 430 (E note)
end else if(octave == (341)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/341/2;

//If Octave is equal to 430 (F note)
end else if(octave == (322)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/322/2;

//If Octave is equal to 430 (F#/Gb note)
end else if(octave == (303)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/303/2;

//If Octave is equal to 430 (G note)
end else if(octave == (286)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/286/2;

//If Octave is equal to 430 (G#/Ab note)
end else if(octave == (270)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/270/2;

//If Octave is equal to 430 (A note)
end else if(octave == (511)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/511/2;

//If Octave is equal to 430 (A#/Bb note)
end else if(octave == (482)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/482/2;

//If Octave is equal to 430 (B note)
end else if(octave == (455)) begin

//Clock Divider for the correct octave
clkdivider = 25000000/455/2;
```

```
		//No octave is given
		end else begin

		//Clock Divider to set the speaker to output no sound
		clkdivider = 0;

		end

		//Divides the clock into a proper signal based on the timing of the counter
		if (clkdivider != 0) begin

			//Changes the counter value to the previous divided clock value correspondingly
			if(counter==0) counter <= clkdivider-1; else counter <= counter-1;

			//Inverts the value of the speaker if counter is equal to 0
			if(counter==0) speaker <= ~speaker;

		end

end

//If the tune Switch is set to be on as well as the system is on
else if (tuneOnOff & onOffSw) begin

/**

Start of FPGA4FUN Sound Segment
https://sites.google.com/site/tgptechnologies/fpga-project/music-box

*/

if(notes_count==0)
		begin
			notes_count <= clkdiv;

				if(octave_counter==0)
				begin

					octave_counter<=8'd255 >> tuneOctave;
				end

				else
				begin
					octave_counter <= octave_counter-8'd1;
				end

		end

		else
		begin
```

```
                notes_count <= notes_count-9'd1;
        end



        if(notes_count==0 && octave_counter==0)
        begin
                if(tunes!=0 && sound[21:18]!=0)
                begin
                speaker <= ~speaker;
                end
        end

end

/**

END of FPGA4FUN Sound Segment
https://sites.google.com/site/tgptechnologies/fpga-project/music-box

https://www.fpga4fun.com/MusicBox.html

*/



//For the when the volumes are set
case(volume)

//For the when the volume is 0%, no path of resistance is chosen
3'b000: begin

speakerVolume[0] = 0;
speakerVolume[1] = 0;
speakerVolume[2] = 0;
speakerVolume[3] = 0;
speakerVolume[4] = 0;

end

//For the when the volume is 20%, First path of resistance is chosen (Highest Resistance)
3'b001: begin

speakerVolume[0] = speaker;
speakerVolume[1] = 0;
speakerVolume[2] = 0;
speakerVolume[3] = 0;
speakerVolume[4] = 0;

end
```

```
//For the when the volume is 40%, Second path of resistance is chosen
3'b010: begin

speakerVolume[0] = 0;
speakerVolume[1] = speaker;
speakerVolume[2] = 0;
speakerVolume[3] = 0;
speakerVolume[4] = 0;

end

//For the when the volume is 60%, Third path of resistance is chosen
3'b011: begin

speakerVolume[0] = 0;
speakerVolume[1] = 0;
speakerVolume[2] = speaker;
speakerVolume[3] = 0;
speakerVolume[4] = 0;

end

//For the when the volume is 80%, Fourth path of resistance is chosen
3'b100: begin

speakerVolume[0] = 0;
speakerVolume[1] = 0;
speakerVolume[2] = 0;
speakerVolume[3] = speaker;
speakerVolume[4] = 0;

end

//For the when the volume is 100%, Fifth path of resistance is chosen (Least Resistance)
3'b101: begin

speakerVolume[0] = 0;
speakerVolume[1] = 0;
speakerVolume[2] = 0;
speakerVolume[3] = 0;
speakerVolume[4] = speaker;


end

endcase

end
```

```
endmodule


/**

START of FPGA4FUN  Music Box Segment
https://www.fpga4fun.com/MusicBox.html

*/


module divide_12(input [5:0] num,  /** value to be divided by 12 */output reg [2:0] quotient, output [3:0]
rem);

reg [1:0] rem_32;
always @(num[5:2])
begin
        quotient=(num[5:2])/3;
        rem_32 = (num[5:2])%3;

end

assign rem[1:0] = num[1:0];  // the first 2 bits will be copied exactly
assign rem[3:2] = rem_32;  // last 2 bits will come from the case statement
endmodule


module music(input clk, input sw,input [7:0] address,output reg [7:0] notes);

always @(posedge clk)
if (sw)
begin
        // Jingle bells tune
        case(address)
                0: notes<= 8'd22;
                        1: notes<= 8'd22;
                        2: notes<= 8'd22;
                        3: notes<= 8'd22;
                        4: notes<= 8'd22;
                        5: notes<= 8'd22;
                        6: notes<= 8'd22;
                        7: notes<= 8'd22;
                        8: notes<= 8'd00;
                        9: notes<= 8'd22;
                        10: notes<= 8'd22;
                        11: notes<= 8'd22;
                        12: notes<= 8'd22;
                        13: notes<= 8'd22;
                        14: notes<= 8'd22;
                        15: notes<= 8'd22;
```

```
            16: notes<= 8'd22;
            17: notes<= 8'd00;
            18: notes<= 8'd22;
        19: notes<= 8'd22;
            20: notes<= 8'd25;
            21: notes<= 8'd25;
            22: notes<= 8'd18;
            23: notes<= 8'd18;
            24: notes<= 8'd20;
            25: notes<= 8'd00;
            26: notes<= 8'd00;
            27: notes<= 8'd22;
            28: notes<= 8'd22;
            29: notes<= 8'd22;
            30: notes<= 8'd22;
            31: notes<= 8'd22;
            32: notes<= 8'd22;
            33: notes<= 8'd22;
            34: notes<= 8'd22;
            35: notes<= 8'd00;
            36: notes<= 8'd23;
            37: notes<= 8'd23;
            38: notes<= 8'd23;
            39: notes<= 8'd23;
            40: notes<= 8'd23;
            41: notes<= 8'd23;
            42: notes<= 8'd23;
            43: notes<= 8'd00;
            44: notes<= 8'd00;
            45: notes<= 8'd23;
            46: notes<= 8'd23;
            47: notes<= 8'd22;
            48: notes<= 8'd22;
            49: notes<= 8'd22;
            50: notes<= 8'd22;
            51: notes<= 8'd22;
            52: notes<= 8'd22;
            53: notes<= 8'd22;
            54: notes<= 8'd00;
            55: notes<= 8'd22;
            56: notes<= 8'd22;
            57: notes<= 8'd20;
            58: notes<= 8'd20;
            59: notes<= 8'd20;
            60: notes<= 8'd20;
            61: notes<= 8'd22;
            62: notes<= 8'd22;
            63: notes<= 8'd00;
            64: notes<= 8'd20;
            65: notes<= 8'd20;
```

```
66: notes<= 8'd20;
67: notes<= 8'd20;
68: notes<= 8'd20;
69: notes<= 8'd25;
70: notes<= 8'd25;
71: notes<= 8'd25;
72: notes<= 8'd25;
73: notes<= 8'd00;
74: notes<= 8'd22;
75: notes<= 8'd22;
76: notes<= 8'd22;
77: notes<= 8'd22;
78: notes<= 8'd22;
79: notes<= 8'd22;
80: notes<= 8'd22;
81: notes<= 8'd22;
82: notes<= 8'd00;
83: notes<= 8'd22;
84: notes<= 8'd22;
85: notes<= 8'd22;
86: notes<= 8'd22;
87: notes<= 8'd22;
88: notes<= 8'd22;
89: notes<= 8'd22;
90: notes<= 8'd22;
91: notes<= 8'd22;
92: notes<= 8'd22;
93: notes<= 8'd00;
94: notes<= 8'd22;
95: notes<= 8'd22;
96: notes<= 8'd25;
97: notes<= 8'd25;
98: notes<= 8'd18;
99: notes<= 8'd18;
100: notes<= 8'd20;
101: notes<= 8'd00;
102: notes<= 8'd00;
103: notes<= 8'd22;
104: notes<= 8'd22;
105: notes<= 8'd22;
106: notes<= 8'd22;
107: notes<= 8'd22;
108: notes<= 8'd22;
109: notes<= 8'd22;
110: notes<= 8'd22;
111: notes<= 8'd00;
112: notes<= 8'd23;
113: notes<= 8'd23;
114: notes<= 8'd23;
115: notes<= 8'd23;
```

```
                    116: notes<= 8'd23;
                    117: notes<= 8'd23;
                    118: notes<= 8'd23;
                    119: notes<= 8'd23;
                    120: notes<= 8'd00;
                    121: notes<= 8'd23;
                    122: notes<= 8'd23;
                    123: notes<= 8'd22;
                    124: notes<= 8'd22;
                    125: notes<= 8'd22;
                    126: notes<= 8'd22;
                    127: notes<= 8'd22;
                    128: notes<= 8'd22;
                    129: notes<= 8'd22;
                    130: notes<= 8'd00;
                    131: notes<= 8'd25;
                    132: notes<= 8'd25;
                    133: notes<= 8'd25;
                    134: notes<= 8'd25;
                    135: notes<= 8'd23;
                    136: notes<= 8'd23;
                    137: notes<= 8'd20;
                    138: notes<= 8'd20;
                    139: notes<= 8'd00;
                    140: notes<= 8'd18;
                    141: notes<= 8'd18;
                    142: notes<= 8'd18;
                    143: notes<= 8'd18;
                    144: notes<= 8'd18;
                    145: notes<= 8'd18;
                    146: notes<= 8'd18;
                    147: notes<= 8'd18;
                    148: notes<= 8'd00;
                    default: notes <= 8'd0;

endcase
end


else
begin
        case(address)
         0: notes<= 8'd18;
         1: notes<= 8'd18;
         2: notes<= 8'd18;
         3: notes<= 8'd18;
         4: notes<= 8'd25;
         5: notes<= 8'd25;
         6: notes<= 8'd25;
         7: notes<= 8'd25;
```

```
8: notes<= 8'd00;
9: notes<= 8'd27;
10: notes<= 8'd27;
11: notes<= 8'd27;
12: notes<= 8'd27;
13: notes<= 8'd25;
14: notes<= 8'd25;
15: notes<= 8'd25;
16: notes<= 8'd25;
17: notes<= 8'd00;
18: notes<= 8'd23;
19: notes<= 8'd23;
20: notes<= 8'd23;
21: notes<= 8'd23;
22: notes<= 8'd22;
23: notes<= 8'd22;
24: notes<= 8'd22;
25: notes<= 8'd22;
26: notes<= 8'd00;
27: notes<= 8'd20;
28: notes<= 8'd20;
29: notes<= 8'd20;
30: notes<= 8'd20;
31: notes<= 8'd18;
32: notes<= 8'd18;
33: notes<= 8'd18;
34: notes<= 8'd18;
35: notes<= 8'd00;
36: notes<= 8'd25;
37: notes<= 8'd25;
38: notes<= 8'd25;
39: notes<= 8'd25;
40: notes<= 8'd23;
41: notes<= 8'd23;
42: notes<= 8'd23;
43: notes<= 8'd23;
44: notes<= 8'd00;
45: notes<= 8'd22;
46: notes<= 8'd22;
47: notes<= 8'd22;
48: notes<= 8'd22;
49: notes<= 8'd20;
50: notes<= 8'd20;
51: notes<= 8'd20;
52: notes<= 8'd20;
53: notes<= 8'd00;
54: notes<= 8'd25;
55: notes<= 8'd25;
56: notes<= 8'd25;
57: notes<= 8'd25;
```

```
 58: notes<= 8'd23;
 59: notes<= 8'd23;
 60: notes<= 8'd23;
 61: notes<= 8'd23;
 62: notes<= 8'd00;
 63: notes<= 8'd22;
 64: notes<= 8'd22;
 65: notes<= 8'd22;
 66: notes<= 8'd22;
 67: notes<= 8'd20;
 68: notes<= 8'd20;
 69: notes<= 8'd20;
 70: notes<= 8'd20;
 71: notes<= 8'd00;
 72: notes<= 8'd18;
 73: notes<= 8'd18;
 74: notes<= 8'd18;
 75: notes<= 8'd18;
 76: notes<= 8'd25;
 77: notes<= 8'd25;
 78: notes<= 8'd25;
 79: notes<= 8'd25;
 80: notes<= 8'd00;
 81: notes<= 8'd27;
 82: notes<= 8'd27;
 83: notes<= 8'd27;
 84: notes<= 8'd27;
 85: notes<= 8'd25;
 86: notes<= 8'd25;
 87: notes<= 8'd25;
 88: notes<= 8'd25;
 89: notes<= 8'd00;
 90: notes<= 8'd23;
 91: notes<= 8'd23;
 92: notes<= 8'd23;
 93: notes<= 8'd23;
 94: notes<= 8'd22;
 95: notes<= 8'd22;
 96: notes<= 8'd22;
 97: notes<= 8'd22;
 98: notes<= 8'd00;
 99: notes<= 8'd20;
100: notes<= 8'd20;
101: notes<= 8'd20;
102: notes<= 8'd20;
103: notes<= 8'd18;
104: notes<= 8'd18;
105: notes<= 8'd18;
106: notes<= 8'd18;
107: notes<= 8'd00;
```

```
                default: notes <= 8'd0;
                        endcase

        end

/**

END of FPGA4FUN  Music Box Segment
https://www.fpga4fun.com/MusicBox.html

*/

endmodule
```

Pin Assignments & Registers:

Variable Name:	**Variable Type:**	**Pin Location:**	**Function:**
keys[0]	Input [0] of [0:11]	PIN_W10	Plays a D note on the instrument
keys[1]	Input [1] of [0:11]	PIN_W9	Plays a C# note on the instrument
keys[2]	Input [2] of [0:11]	PIN_W8	Plays a C note on the instrument
keys[3]	Input [3] of [0:11]	PIN_W7	Plays a B note on the instrument
keys[4]	Input [4] of [0:11]	PIN_V5	Plays a A# note on the instrument
keys[5]	Input [5] of [0:11]	PIN_AA15	Plays a A note on the instrument
keys[6]	Input [6] of [0:11]	PIN_Y8	Plays a G# note on the instrument
keys[7]	Input [7] of [0:11]	PIN_Y7	Plays a G note on the instrument
keys[8]	Input [8] of [0:11]	PIN_Y6	Plays a F# note on the instrument
keys[9]	Input [9] of [0:11]	PIN_Y5	Plays a F note on the instrument
keys[10]	Input [10] of [0:11]	PIN_Y4	Plays a E note on the instrument
keys[11]	Input [11] of [0:11]	PIN_Y3	Plays a D# note on the instrument
volumeControl[0]	Input [0] of [0:1]	PIN_B8	Decrements the volume by 20%
volumeControl[1]	Input [1] of [0:1]	PIN_A7	Increments the volume by 20%
onOffSw	Input	PIN_C10	Soft On/Off switch
pitchSw	Input	PIN_C11	Switch to change pitch (SW1 off =

			High, SW1 on = Low)
tuneOnOff	Input	PIN_B14	Switch to turn on preset tunes instead of the instrument
tuneChoice	Input	PIN_F15	Switches between preset tunes
clkHighPitch	Input	PIN_P11	50 MHz Clock Signal (Produces High Pitch Signal)
clkLowPitch	Input	PIN_N5	10 MHz Clock Signal (Produces Low Pitch Signal)
clk	reg	**NA**	Register of clock chosen by user
nextIncrement	reg	**NA**	Register to allow volume changes one button press at a time
volume[0]	reg[0] of [0:2]	**NA**	Controls volume level based on the resistors located on the breadboard
volume[1]	reg[1] of [0:2]	**NA**	Controls volume level based on the resistors located on the breadboard
volume[2]	reg[2] of [0:2]	**NA**	Controls volume level based on the resistors located on the breadboard
octave[0]	reg[0] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[1]	reg[1] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[2]	reg[2] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[3]	reg[3] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[4]	reg[4] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[5]	reg[5] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[6]	reg[6] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play

octave[7]	reg[7] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
octave[8]	reg[8] of [0:8]	**NA**	Register to save the octave value of the note the user wants to play
pwrSevenSegDot	reg	**NA**	Controls the Seven Segment Dot on the First Display to signify the system is on
noteSevenSeg[0]	reg[0] of [0:7]	**NA**	Seven Segment value of the Segment 0, on the very last display which displays the current note being played
noteSevenSeg[1]	reg[1] of [0:7]	**NA**	Seven Segment value of the Segment 1, on the very last display which displays the current note being played
noteSevenSeg[2]	reg[2] of [0:7]	**NA**	Seven Segment value of the Segment 2, on the very last display which displays the current note being played
noteSevenSeg[3]	reg[3] of [0:7]	**NA**	Seven Segment value of the Segment 3, on the very last display which displays the current note being played
noteSevenSeg[4]	reg[4] of [0:7]	**NA**	Seven Segment value of the Segment 4, on the very last display which displays the current note being played
noteSevenSeg[5]	reg[5] of [0:7]	**NA**	Seven Segment value of the Segment 5, on the very last display which displays the current note being played
noteSevenSeg[6]	reg[6] of [0:7]	**NA**	Seven Segment value of the Segment 6, on the very last display which displays the current note being played
noteSevenSeg[7]	reg[7] of [0:7]	**NA**	Seven Segment value of the Segment 7, on the very last display which displays if the current note is a sharp note or not

hiLoSevenSegL[0]	reg[0] of [0:6]	**NA**	Left Seven Segment display value for Segment 0, for High or Low Pitches
hiLoSevenSegL[1]	reg[1] of [0:6]	**NA**	Left Seven Segment display value for Segment 1, for High or Low Pitches
hiLoSevenSegL[2]	reg[2] of [0:6]	**NA**	Left Seven Segment display value for Segment 2, for High or Low Pitches
hiLoSevenSegL[3]	reg[3] of [0:6]	**NA**	Left Seven Segment display value for Segment 3, for High or Low Pitches
hiLoSevenSegL[4]	reg[4] of [0:6]	**NA**	Left Seven Segment display value for Segment 4, for High or Low Pitches
hiLoSevenSegL[5]	reg[5] of [0:6]	**NA**	Left Seven Segment display value for Segment 5, for High or Low Pitches
hiLoSevenSegL[6]	reg[6] of [0:6]	**NA**	Left Seven Segment display value for Segment 6, for High or Low Pitches
hiLoSevenSegR[0]	reg[0] of [0:6]	**NA**	Right Seven Segment display value for Segment 0, for High or Low Pitches
hiLoSevenSegR[1]	reg[1] of [0:6]	**NA**	Right Seven Segment display value for Segment 1, for High or Low Pitches
hiLoSevenSegR[2]	reg[2] of [0:6]	**NA**	Right Seven Segment display value for Segment 2, for High or Low Pitches
hiLoSevenSegR[3]	reg[3] of [0:6]	**NA**	Right Seven Segment display value for Segment 3, for High or Low Pitches
hiLoSevenSegR[4]	reg[4] of [0:6]	**NA**	Right Seven Segment display value for Segment 4, for High or Low Pitches
hiLoSevenSegR[5]	reg[5] of [0:6]	**NA**	Right Seven Segment display value for Segment 5, for High or Low

			Pitches
hiLoSevenSegR[6]	reg[6] of [0:6]	**NA**	Right Seven Segment display value for Segment 6, for High or Low Pitches
LEDArray[0]	Output reg [0] of [0:9]	PIN_A8	Outputs the current set volume level, LED 1 of the 10
LEDArray[1]	Output reg [1] of [0:9]	PIN_A9	Outputs the current set volume level, LED 2 of the 10
LEDArray[2]	Output reg [2] of [0:9]	PIN_A10	Outputs the current set volume level, LED 3 of the 10
LEDArray[3]	Output reg [3] of [0:9]	PIN_B10	Outputs the current set volume level, LED 4 of the 10
LEDArray[4]	Output reg [4] of [0:9]	PIN_D13	Outputs the current set volume level, LED 5 of the 10
LEDArray[5]	Output reg [5] of [0:9]	PIN_C13	Outputs the current set volume level, LED 6 of the 10
LEDArray[6]	Output reg [6] of [0:9]	PIN_E14	Outputs the current set volume level, LED 7 of the 10
LEDArray[7]	Output reg [7] of [0:9]	PIN_D14	Outputs the current set volume level, LED 8 of the 10
LEDArray[8]	Output reg [8] of [0:9]	PIN_A11	Outputs the current set volume level, LED 9 of the 10
LEDArray[9]	Output reg [9] of [0:9]	PIN_B11	Outputs the current set volume level, LED 10 of the 10
speakerVolume[0]	Output [0] of [0:4]	PIN_AA7	The path of 20% Volume for the speaker
speakerVolume[1]	Output [1] of [0:4]	PIN_AA6	The path of 40% Volume for the speaker
speakerVolume[2]	Output [2] of [0:4]	PIN_AA5	The path of 60% Volume for the speaker
speakerVolume[3]	Output [3] of [0:4]	PIN_AB3	The path of 80% Volume for the speaker
speakerVolume[4]	Output [4] of [0:4]	PIN_AB2	The path of 100% Volume for the speaker

pwrDisp	Output	PIN_D15	Outputs the Seven Segment Dot on the First Display to signify the system is on
notesDisp[0]	Output [0] of [0:7]	PIN_J20	Seven Segment Display of the Segment 0, on the very last display which shows the current note being played
notesDisp[1]	Output [1] of [0:7]	PIN_K20	Seven Segment Display of the Segment 1, on the very last display which shows the current note being played
notesDisp[2]	Output [2] of [0:7]	PIN_L18	Seven Segment Display of the Segment 2, on the very last display which shows the current note being played
notesDisp[3]	Output [3] of [0:7]	PIN_N18	Seven Segment Display of the Segment 3, on the very last display which shows the current note being played
notesDisp[4]	Output [4] of [0:7]	PIN_M20	Seven Segment Display of the Segment 4, on the very last display which shows the current note being played
notesDisp[5]	Output [5] of [0:7]	PIN_N19	Seven Segment Display of the Segment 5, on the very last display which shows the current note being played
notesDisp[6]	Output [6] of [0:7]	PIN_N20	Seven Segment Display of the Segment 6, on the very last display which shows the current note being played
notesDisp[7]	Output [7] of [0:7]	PIN_L19	Seven Segment Display of the Segment 7, on the very last display which shows the current note being played
hiLoSevenSegL[0]	Output [0] of [0:6]	PIN_C14	Left Seven Segment display for Segment 0, for High or Low Pitches
hiLoSevenSegL[1]	Output [1] of [0:6]	PIN_E15	Left Seven Segment display for Segment 1, for High or Low Pitches

hiLoSevenSegL[2]	Output [2] of [0:6]	PIN_C15	Left Seven Segment display for Segment 2, for High or Low Pitches
hiLoSevenSegL[3]	Output [3] of [0:6]	PIN_C16	Left Seven Segment display for Segment 3, for High or Low Pitches
hiLoSevenSegL[4]	Output [4] of [0:6]	PIN_E16	Left Seven Segment display for Segment 4, for High or Low Pitches
hiLoSevenSegL[5]	Output [5] of [0:6]	PIN_D17	Left Seven Segment display for Segment 5, for High or Low Pitches
hiLoSevenSegL[6]	Output [6] of [0:6]	PIN_C17	Left Seven Segment display for Segment 6, for High or Low Pitches
hiLoSevenSegR[0]	Output [0] of [0:6]	PIN_C18	Right Seven Segment display for Segment 0, for High or Low Pitches
hiLoSevenSegR[1]	Output [1] of [0:6]	PIN_D18	Right Seven Segment display for Segment 1, for High or Low Pitches
hiLoSevenSegR[2]	Output [2] of [0:6]	PIN_E18	Right Seven Segment display for Segment 2, for High or Low Pitches
hiLoSevenSegR[3]	Output [3] of [0:6]	PIN_B16	Right Seven Segment display for Segment 3, for High or Low Pitches
hiLoSevenSegR[4]	Output [4] of [0:6]	PIN_A17	Right Seven Segment display for Segment 4, for High or Low Pitches
hiLoSevenSegR[5]	Output [5] of [0:6]	PIN_A18	Right Seven Segment display for Segment 5, for High or Low Pitches
hiLoSevenSegR[6]	Output [6] of [0:6]	PIN_B17	Right Seven Segment display for Segment 6, for High or Low Pitches

Use of Work:

Our project used work/code from sources outside of the class material provided, this includes:

Basis of Note Playing: https://sites.google.com/site/tgptechnologies/fpga-project/music-box
Music Box Tunes: https://www.fpga4fun.com/MusicBox.html

Link to Video/Demo:

Video/Demo via YouTube: https://www.youtube.com/watch?v=MCiXtk6o_28