ECEN302 Lab 8 - Sound and light show

Joshua Benfell - 300433229

October 3, 2020

1 Introduction

This lab builds on the previous labs by using the created IP block as a peripheral to the Microblaze microcontroller. The microcontroller will be used to send the digital signal, that was previously done as analog switches on the fpga, to the audio IP block so that it can play a sequence of sounds. This will be done by programming the microcontroller with the C programming language.

2 Methodology

To begin this project the microcontroller setup in lab 6 was opened and the audio block created in lab 7 was included along with ports and an axi_gpio bus. This will setup all the peripherals needed for the microcontroller to produce sound. In addition to these peripherals there is another axi_gpio bus that is used to communicate with the boards LEDs.

This was uploaded to the FPGA and the Vitis editor opened with the imported xsa file. The first program put onto the microcontroller was a test one that turned on every light and every switch into the audio block. This proved that what was made was functioning. How this works is by first initialising a gpio device, in this ase the axi buses. The direction is then set to output (0x0) on the selected channel, the lights had 2 channels as the switches were also connected to this bus, but the audio had one. All bits in the channel are then set to a low output so that nothing shows. To write a value the XGpio_DiscreteWrite function is called passing through a 32 bit number of which only the 9 LSBs are used as they are all the device will accept.

$$\text{value} = \frac{\text{F\_CLK}}{1024 \times f} \tag{1}$$

$$\text{length} = \frac{\text{noteDur} \times 3000000}{\text{bpm}} \tag{2}$$

Now that the functionality has been verified, it was time to produce some proper music. To do this the code was set up so that we could pass through a frequency and have it converted to the appropriate writable value. This was done with eq. (1), where the value 1024 comes from the frequency of the PWM signal. This was then played for eq. (2) duration where teh 3000000 was found through trial and error and it was something that sounded alright. From there it was found that a gap between notes needed to exist, so no sound was played for 800000 clock cycles as without this, there was nothing to be heard but the last note.

Now that this was all setup, a song was created in the form of a 2D array with notes and the duration to play them. This was then looped through and each tone was played in sequence. This data is from a github repo that I made with Sam Griffen in Year 13 at high school.

3 Code

3.1 Song Data

```
const int fn = 240; //Duration of 4 beats
const int hn = 120; //Duration of 2 beats
const int qn = 60; //Duration of 1 beat
const int en = 30;
const int sn = 15;
//const int bpm = 180;

// Define the notes
const int c[9]   = {16, 33, 65, 131, 262, 523, 1047, 2093, 4186};
const int cs[9]  = {17, 35, 69, 139, 277, 554, 1109, 2217, 4435};
const int d[9]   = {18, 37, 73, 147, 294, 587, 1175, 2349, 4699};
const int eb[9]  = {19, 39, 78, 156, 311, 622, 1245, 2489, 4978};
const int e[9]   = {21, 41, 82, 165, 330, 659, 1319, 2637, 5274};
const int f[9]   = {22, 44, 87, 175, 349, 699, 1397, 2794, 5588};
const int fs[9]  = {23, 46, 93, 185, 370, 740, 1480, 2960, 5920};
const int g[9]   = {25, 49, 98, 196, 392, 784, 1568, 3136, 6272};
const int gs[9]  = {26, 52, 104, 208, 415, 831, 1661, 3322, 6645};
const int a[9]   = {28, 55, 110, 220, 440, 880, 1760, 3520, 7040};
const int bb[9]  = {29, 58, 117, 233, 466, 932, 1865, 3729, 7459};
const int b[9]   = {31, 62, 124, 247, 494, 988, 1976, 3951, 7902};
const int r = 25000;

long bpm = 140;
long song[][2] ={
  {b[4]    , qn + en},//2st bar start  | Darude Start * 3
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},     //2st bar end
  {e[5]    , qn + en},//3nd bar start
  {e[5]    , qn + en},
  {d[5]    , en},
  {d[5]    , qn + en},
  {d[5]    , qn + en},
  {a[4]    , en},     //3nd bar end
  {b[4]    , qn + en},//4rd bar start
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},     //4rd bar end
  {b[4]    , qn + en},//5th bar start
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},   //5th bar end
  {b[4]    , qn + en},//6th bar start
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},      //6th bar end
  {e[5]    , qn + en},//6th bar start
  {e[5]    , qn + en},
  {d[5]    , en},
  {d[5]    , qn + en},
  {d[5]    , qn + en},
  {a[4]    , en},      //6th bar end
  {b[4]    , qn + en},//7th bar start
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},      //7th bar end
  {b[4]    , qn + en},//8th bar start
  {b[4]    , qn + en},
  {b[4]    , en},
  {b[4]    , qn + en},
  {b[4]    , qn + en},
  {e[5]    , en},    //8th bar end  | Darude Start end
  {b[3]    , sn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {r       , fn + en + qn},
  {d[4]    , qn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {r       , fn + en + qn + fn},
  {b[3]    , sn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {b[3]    , sn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {b[3]    , sn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {b[3]    , sn},//duh_duh_duh_duh_da Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},//duh_duh_duh_duh_da end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
```

```
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {b[3]    , sn},//duh_duh_duh_duh Start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},//duh_duh_duh_duh end
  {d[4]    , qn},
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end

  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end

  {b[3]    , sn},//darude_up_duh_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},//darude_up_duh_da End
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//darude_up_duh_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},//darude_up_duh_da End
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//Darude up
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {e[4]    , sn},
  {e[4]    , sn},//Darude up end
  {e[4]    , sn},//Darude Down
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , sn},
  {e[4]    , en},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , sn},
  {d[4]    , en},
  {a[3]    , sn},
  {a[3]    , sn},//Darude Down end
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//darude_up_da_da start
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , en},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {b[3]    , sn},
  {d[4]    , en},
  {d[4]    , en},//darude_up_da_da end
  {b[3]    , sn},//duh_duh_duh_da_da start
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},
  {d[4]    , en},//duh_duh_duh_da_da end
  {b[3]    , sn},//duh_duh_duh_da_da start
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},
  {d[4]    , en},//duh_duh_duh_da_da end
  {b[3]    , sn},//duh_duh_duh_da_da start
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},
  {d[4]    , en},//duh_duh_duh_da_da end
  {b[3]    , sn},//duh_duh_duh_da_da start
  {b[3]    , sn},
  {b[3]    , en},
  {d[4]    , en},
  {d[4]    , en},//duh_duh_duh_da_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {b[3]    , sn},//duh_duh_da start
  {b[3]    , sn},
  {d[4]    , en},//duh_duh_da end
  {e[5]    , fn}
};
```

code/song.h

3.2 Music Player

```
/*
 *
 * Xilinx, Inc.
 * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A
 * COURTESY TO YOU.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS
 * ONE POSSIBLE   IMPLEMENTATION OF THIS FEATURE, APPLICATION OR
 * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION
 * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE
 * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION
 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO
 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO
 * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE
 * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.
 */

/*
 *
 *
 * This file is a generated sample test application.
 *
 * This application is intended to test and/or illustrate some
 * functionality of your system.  The contents of this file may
 * vary depending on the IP in your system and may use existing
 * IP driver functions.  These drivers will be generated in your
 * SDK application project when you run the "Generate Libraries" menu item.
 *
 */

#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xgpio.h"
#include "gpio_header.h"
#include "song.h"

XGpio GpioOutput; /* The driver instance for GPIO Device configured as O/P */

#define F_CLK 100000000.0
#define CONV 100000000.0 / 1024.0
#define DELAY 3000000

int setupAudio() {
    int Status;

    /*
     * Initialize the GPIO driver so that it's ready to use,
     * specify the device ID that is generated in xparameters.h
     */
     Status = XGpio_Initialize(&GpioOutput, XPAR_AXI_GPIO_1_DEVICE_ID);
     if (Status != XST_SUCCESS)  {
          return XST_FAILURE;
     }

     /* Set the direction for all signals to be outputs */
     XGpio_SetDataDirection(&GpioOutput, 1, 0x0);

     /* Set the GPIO outputs to low */
     XGpio_DiscreteWrite(&GpioOutput, 1, 0x0);
     return XST_SUCCESS;
}

void tone(int freq, int length) {
    long note = F_CLK / (1024 * freq);
    XGpio_DiscreteWrite(&GpioOutput, 1, note);
    int timeDelay = (length * DELAY) / bpm;
    for(int Delay = 0; Delay < timeDelay; Delay++);
    XGpio_DiscreteClear(&GpioOutput, 1, 0x0);
    for(int Delay = 0; Delay < 800000; Delay++);
}
```

```


int main ()
{
     Xil_ICacheEnable();
     Xil_DCacheEnable();
     print("---Entering main---\n\r");




     setupAudio();



    /*
     * Peripheral SelfTest will not be run for axi_uartlite_0
     * because it has been selected as the STDOUT device
     */
     for (int i = 0; i < (sizeof(song)/sizeof(song[0])); i++){
         tone(song[i][0], song[i][1]);
     }



    print("---Exiting main---\n\r");
    Xil_DCacheDisable();
    Xil_ICacheDisable();
    return 0;
}
```

code/testperiph.c