

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 case 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.

$$\text{value} = \frac{F_CLK}{1024 \times f} \quad (1)$$

$$\text{length} = \frac{\text{noteDur} \times 3000000}{\text{bpm}} \quad (2)$$

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 the 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

```
1 const int fn = 240; //Duration of 4 beats
2 const int hn = 120; //Duration of 2 beats
3 const int qn = 60; //Duration of 1 beat
4 const int en = 30;
5 const int sn = 15;
6 //const int bpm = 180;
7
8 // Define the notes
9 const int c[9] = {16, 33, 65, 131, 262, 523, 1047, 2093, 4186};
10 const int cs[9] = {17, 35, 69, 139, 277, 554, 1109, 2217, 4435};
11 const int d[9] = {18, 37, 73, 147, 294, 587, 1175, 2349, 4699};
12 const int eb[9] = {19, 39, 78, 156, 311, 622, 1245, 2489, 4978};
13 const int e[9] = {21, 41, 82, 165, 330, 659, 1319, 2637, 5274};
14 const int f[9] = {22, 44, 87, 175, 349, 699, 1397, 2794, 5588};
15 const int fs[9] = {23, 46, 93, 185, 370, 740, 1480, 2960, 5920};
16 const int g[9] = {25, 49, 98, 196, 392, 784, 1568, 3136, 6272};
17 const int gs[9] = {26, 52, 104, 208, 415, 831, 1661, 3322, 6645};
18 const int a[9] = {28, 55, 110, 220, 440, 880, 1760, 3520, 7040};
19 const int bb[9] = {29, 58, 117, 233, 466, 932, 1865, 3729, 7459};
20 const int b[9] = {31, 62, 124, 247, 494, 988, 1976, 3951, 7902};
21 const int r = 25000;
22
23 long bpm = 140;
24 long song[][2] = {
25     {b[4], qn + en}, //2st bar start | Darude Start * 3
26     {b[4], qn + en},
27     {b[4], en},
28     {b[4], qn + en},
29     {b[4], qn + en},
30     {e[5], en}, //2st bar end
31     {e[5], qn + en}, //3nd bar start
32     {e[5], qn + en},
33     {d[5], en},
34     {d[5], qn + en},
35     {d[5], qn + en},
36     {a[4], en}, //3nd bar end
37     {b[4], qn + en}, //4rd bar start
38     {b[4], qn + en},
39     {b[4], en},
40     {b[4], qn + en},
41     {b[4], qn + en},
42     {e[5], en}, //4rd bar end
43     {b[4], qn + en}, //5th bar start
44     {b[4], qn + en},
45     {b[4], en},
46     {b[4], qn + en},
47     {b[4], qn + en},
48     {e[5], en}, //5th bar end
49     {b[4], qn + en}, //6th bar start
50     {b[4], qn + en},
```

```

51 {b[4] , en},
52 {b[4] , qn + en},
53 {b[4] , qn + en},
54 {e[5] , en}, //6th bar end
55 {e[5] , qn + en}, //6th bar start
56 {e[5] , qn + en},
57 {d[5] , en},
58 {d[5] , qn + en},
59 {d[5] , qn + en},
60 {a[4] , en}, //6th bar end
61 {b[4] , qn + en}, //7th bar start
62 {b[4] , qn + en},
63 {b[4] , en},
64 {b[4] , qn + en},
65 {b[4] , qn + en},
66 {e[5] , en}, //7th bar end
67 {b[4] , qn + en}, //8th bar start
68 {b[4] , qn + en},
69 {b[4] , en},
70 {b[4] , qn + en},
71 {b[4] , qn + en},
72 {e[5] , en}, //8th bar end | Darude Start end
73 {b[3] , sn}, //duh-duh-duh-duh-da Start
74 {b[3] , sn},
75 {b[3] , sn},
76 {b[3] , sn},
77 {b[3] , en}, //duh-duh-duh-duh-da end
78 {r , fn + en + qn},
79 {d[4] , qn}, //duh-duh-duh-duh-da Start
80 {b[3] , sn},
81 {b[3] , sn},
82 {b[3] , sn},
83 {b[3] , sn},
84 {b[3] , en}, //duh-duh-duh-duh-da end
85 {r , fn + en + qn + fn},
86 {b[3] , sn}, //duh-duh-duh-duh-da Start
87 {b[3] , sn},
88 {b[3] , sn},
89 {b[3] , sn},
90 {b[3] , en}, //duh-duh-duh-duh-da end
91 {b[3] , sn}, //duh-duh-duh-duh-da Start
92 {b[3] , sn},
93 {b[3] , sn},
94 {b[3] , sn},
95 {b[3] , en}, //duh-duh-duh-duh-da end
96 {b[3] , sn}, //duh-duh-duh-duh-da Start
97 {b[3] , sn},
98 {b[3] , sn},
99 {b[3] , sn},
100 {b[3] , en}, //duh-duh-duh-duh-da end
101 {b[3] , sn}, //duh-duh-duh-duh-da Start
102 {b[3] , sn},
103 {b[3] , sn},
104 {b[3] , sn},
105 {b[3] , en}, //duh-duh-duh-duh-da end
106 {b[3] , sn}, //duh-duh-duh-duh Start
107 {b[3] , sn},
108 {b[3] , sn},
109 {b[3] , sn}, //duh-duh-duh-duh end

```

```

110 {b[3] , sn}, //duh-duh-duh-duh Start
111 {b[3] , sn},
112 {b[3] , sn},
113 {b[3] , sn}, //duh-duh-duh-duh end
114 {b[3] , sn}, //duh-duh-duh-duh Start
115 {b[3] , sn},
116 {b[3] , sn},
117 {b[3] , sn}, //duh-duh-duh-duh end
118 {b[3] , sn}, //duh-duh-duh-duh Start
119 {b[3] , sn},
120 {b[3] , sn},
121 {b[3] , sn}, //duh-duh-duh-duh end
122 {b[3] , sn}, //duh-duh-duh-duh Start
123 {b[3] , sn},
124 {b[3] , sn},
125 {b[3] , sn}, //duh-duh-duh-duh end
126 {b[3] , sn}, //duh-duh-duh-duh Start
127 {b[3] , sn},
128 {b[3] , sn},
129 {b[3] , sn}, //duh-duh-duh-duh end
130 {b[3] , sn}, //duh-duh-duh-duh Start
131 {b[3] , sn},
132 {b[3] , sn},
133 {b[3] , sn}, //duh-duh-duh-duh end
134 {d[4] , qn},
135 {b[3] , sn}, //Darude up
136 {b[3] , sn},
137 {b[3] , sn},
138 {b[3] , sn},
139 {b[3] , en},
140 {b[3] , sn},
141 {b[3] , sn},
142 {b[3] , sn},
143 {b[3] , sn},
144 {b[3] , sn},
145 {b[3] , sn},
146 {b[3] , en},
147 {e[4] , sn},
148 {e[4] , sn}, //Darude up end
149 {e[4] , sn}, //Darude Down
150 {e[4] , sn},
151 {e[4] , sn},
152 {e[4] , sn},
153 {e[4] , en},
154 {d[4] , sn},
155 {d[4] , sn},
156 {d[4] , sn},
157 {d[4] , sn},
158 {d[4] , sn},
159 {d[4] , sn},
160 {d[4] , en},
161 {a[3] , sn},
162 {a[3] , sn}, //Darude Down end
163 {b[3] , sn}, //Darude up
164 {b[3] , sn},
165 {b[3] , sn},
166 {b[3] , sn},
167 {b[3] , en},
168 {b[3] , sn},

```

```

169 {b[3]      , sn},
170 {b[3]      , sn},
171 {b[3]      , sn},
172 {b[3]      , sn},
173 {b[3]      , sn},
174 {b[3]      , en},
175 {e[4]      , sn},
176 {e[4]      , sn}, //Darude up end
177 {b[3]      , sn}, //Darude up
178 {b[3]      , sn},
179 {b[3]      , sn},
180 {b[3]      , sn},
181 {b[3]      , en},
182 {b[3]      , sn},
183 {b[3]      , sn},
184 {b[3]      , sn},
185 {b[3]      , sn},
186 {b[3]      , sn},
187 {b[3]      , sn},
188 {b[3]      , en},
189 {e[4]      , sn},
190 {e[4]      , sn}, //Darude up end
191 {b[3]      , sn}, //Darude up
192 {b[3]      , sn},
193 {b[3]      , sn},
194 {b[3]      , sn},
195 {b[3]      , en},
196 {b[3]      , sn},
197 {b[3]      , sn},
198 {b[3]      , sn},
199 {b[3]      , sn},
200 {b[3]      , sn},
201 {b[3]      , sn},
202 {b[3]      , en},
203 {e[4]      , sn},
204 {e[4]      , sn}, //Darude up end
205 {e[4]      , sn}, //Darude Down
206 {e[4]      , sn},
207 {e[4]      , sn},
208 {e[4]      , sn},
209 {e[4]      , en},
210 {d[4]      , sn},
211 {d[4]      , sn},
212 {d[4]      , sn},
213 {d[4]      , sn},
214 {d[4]      , sn},
215 {d[4]      , sn},
216 {d[4]      , en},
217 {a[3]      , sn},
218 {a[3]      , sn}, //Darude Down end
219 {b[3]      , sn}, //Darude up
220 {b[3]      , sn},
221 {b[3]      , sn},
222 {b[3]      , sn},
223 {b[3]      , en},
224 {b[3]      , sn},
225 {b[3]      , sn},
226 {b[3]      , sn},
227 {b[3]      , sn},

```

```

228 {b[3] , sn} ,
229 {b[3] , sn} ,
230 {b[3] , en} ,
231 {e[4] , sn} ,
232 {e[4] , sn} , //Darude up end
233 {b[3] , sn} , //Darude up
234 {b[3] , sn} ,
235 {b[3] , sn} ,
236 {b[3] , sn} ,
237 {b[3] , en} ,
238 {b[3] , sn} ,
239 {b[3] , sn} ,
240 {b[3] , sn} ,
241 {b[3] , sn} ,
242 {b[3] , sn} ,
243 {b[3] , sn} ,
244 {b[3] , en} ,
245 {e[4] , sn} ,
246 {e[4] , sn} , //Darude up end
247 {b[3] , sn} , //Darude up
248 {b[3] , sn} ,
249 {b[3] , sn} ,
250 {b[3] , sn} ,
251 {b[3] , en} ,
252 {b[3] , sn} ,
253 {b[3] , sn} ,
254 {b[3] , sn} ,
255 {b[3] , sn} ,
256 {b[3] , sn} ,
257 {b[3] , sn} ,
258 {b[3] , en} ,
259 {e[4] , sn} ,
260 {e[4] , sn} , //Darude up end
261
262 {e[4] , sn} , //Darude Down
263 {e[4] , sn} ,
264 {e[4] , sn} ,
265 {e[4] , sn} ,
266 {e[4] , en} ,
267 {d[4] , sn} ,
268 {d[4] , sn} ,
269 {d[4] , sn} ,
270 {d[4] , sn} ,
271 {d[4] , sn} ,
272 {d[4] , sn} ,
273 {d[4] , en} ,
274 {a[3] , sn} ,
275 {a[3] , sn} , //Darude Down end
276 {b[3] , sn} , //Darude up
277 {b[3] , sn} ,
278 {b[3] , sn} ,
279 {b[3] , sn} ,
280 {b[3] , en} ,
281 {b[3] , sn} ,
282 {b[3] , sn} ,
283 {b[3] , sn} ,
284 {b[3] , sn} ,
285 {b[3] , sn} ,
286 {b[3] , sn} ,

```

```

287 {b[3] , en},
288 {e[4] , sn},
289 {e[4] , sn}, //Darude up end
290 {b[3] , sn}, //Darude up
291 {b[3] , sn},
292 {b[3] , sn},
293 {b[3] , sn},
294 {b[3] , en},
295 {b[3] , sn},
296 {b[3] , sn},
297 {b[3] , sn},
298 {b[3] , sn},
299 {b[3] , sn},
300 {b[3] , sn},
301 {b[3] , en},
302 {e[4] , sn},
303 {e[4] , sn}, //Darude up end
304 {b[3] , sn}, //Darude up
305 {b[3] , sn},
306 {b[3] , sn},
307 {b[3] , sn},
308 {b[3] , en},
309 {b[3] , sn},
310 {b[3] , sn},
311 {b[3] , sn},
312 {b[3] , sn},
313 {b[3] , sn},
314 {b[3] , sn},
315 {b[3] , en},
316 {e[4] , sn},
317 {e[4] , sn}, //Darude up end
318 {e[4] , sn}, //Darude Down
319 {e[4] , sn},
320 {e[4] , sn},
321 {e[4] , sn},
322 {e[4] , en},
323 {d[4] , sn},
324 {d[4] , sn},
325 {d[4] , sn},
326 {d[4] , sn},
327 {d[4] , sn},
328 {d[4] , sn},
329 {d[4] , en},
330 {a[3] , sn},
331 {a[3] , sn}, //Darude Down end
332 {b[3] , sn}, //Darude up
333 {b[3] , sn},
334 {b[3] , sn},
335 {b[3] , sn},
336 {b[3] , en},
337 {b[3] , sn},
338 {b[3] , sn},
339 {b[3] , sn},
340 {b[3] , sn},
341 {b[3] , sn},
342 {b[3] , sn},
343 {b[3] , en},
344 {e[4] , sn},
345 {e[4] , sn}, //Darude up end

```

```

346 {b[3] , sn} , //Darude up
347 {b[3] , sn} ,
348 {b[3] , sn} ,
349 {b[3] , sn} ,
350 {b[3] , en} ,
351 {b[3] , sn} ,
352 {b[3] , sn} ,
353 {b[3] , sn} ,
354 {b[3] , sn} ,
355 {b[3] , sn} ,
356 {b[3] , sn} ,
357 {b[3] , en} ,
358 {e[4] , sn} ,
359 {e[4] , sn} , //Darude up end
360
361 {b[3] , sn} , //darude_up_duh_da start
362 {b[3] , sn} ,
363 {b[3] , sn} ,
364 {b[3] , sn} ,
365 {b[3] , en} ,
366 {b[3] , sn} ,
367 {b[3] , sn} ,
368 {b[3] , sn} ,
369 {b[3] , sn} ,
370 {b[3] , sn} ,
371 {b[3] , sn} ,
372 {b[3] , en} ,
373 {d[4] , en} , //darude_up_duh_da End
374 {b[3] , sn} , //darude_up_da-da start
375 {b[3] , sn} ,
376 {b[3] , sn} ,
377 {b[3] , sn} ,
378 {b[3] , en} ,
379 {b[3] , sn} ,
380 {b[3] , sn} ,
381 {b[3] , sn} ,
382 {b[3] , sn} ,
383 {b[3] , sn} ,
384 {b[3] , sn} ,
385 {d[4] , en} ,
386 {d[4] , en} , //darude_up_da-da end
387 {b[3] , sn} , //Darude up
388 {b[3] , sn} ,
389 {b[3] , sn} ,
390 {b[3] , sn} ,
391 {b[3] , en} ,
392 {b[3] , sn} ,
393 {b[3] , sn} ,
394 {b[3] , sn} ,
395 {b[3] , sn} ,
396 {b[3] , sn} ,
397 {b[3] , sn} ,
398 {b[3] , en} ,
399 {e[4] , sn} ,
400 {e[4] , sn} , //Darude up end
401 {e[4] , sn} , //Darude Down
402 {e[4] , sn} ,
403 {e[4] , sn} ,
404 {e[4] , sn} ,

```



```

405 {e[4] , en},
406 {d[4] , sn},
407 {d[4] , sn},
408 {d[4] , sn},
409 {d[4] , sn},
410 {d[4] , sn},
411 {d[4] , sn},
412 {d[4] , en},
413 {a[3] , sn},
414 {a[3] , sn}, //Darude Down end
415 {b[3] , sn}, //darude_up-duh-da start
416 {b[3] , sn},
417 {b[3] , sn},
418 {b[3] , sn},
419 {b[3] , en},
420 {b[3] , sn},
421 {b[3] , sn},
422 {b[3] , sn},
423 {b[3] , sn},
424 {b[3] , sn},
425 {b[3] , sn},
426 {b[3] , en},
427 {d[4] , en}, //darude_up-duh-da End
428 {b[3] , sn}, //darude_up-da-da start
429 {b[3] , sn},
430 {b[3] , sn},
431 {b[3] , sn},
432 {b[3] , en},
433 {b[3] , sn},
434 {b[3] , sn},
435 {b[3] , sn},
436 {b[3] , sn},
437 {b[3] , sn},
438 {b[3] , sn},
439 {d[4] , en},
440 {d[4] , en}, //darude_up-da-da end
441 {b[3] , sn}, //Darude up
442 {b[3] , sn},
443 {b[3] , sn},
444 {b[3] , sn},
445 {b[3] , en},
446 {b[3] , sn},
447 {b[3] , sn},
448 {b[3] , sn},
449 {b[3] , sn},
450 {b[3] , sn},
451 {b[3] , sn},
452 {b[3] , en},
453 {e[4] , sn},
454 {e[4] , sn}, //Darude up end
455 {e[4] , sn}, //Darude Down
456 {e[4] , sn},
457 {e[4] , sn},
458 {e[4] , sn},
459 {e[4] , en},
460 {d[4] , sn},
461 {d[4] , sn},
462 {d[4] , sn},
463 {d[4] , sn},

```

```

464 {d[4] , sn},
465 {d[4] , sn},
466 {d[4] , en},
467 {a[3] , sn},
468 {a[3] , sn}, //Darude Down end
469 {b[3] , sn}, //darude_up_da_da start
470 {b[3] , sn},
471 {b[3] , sn},
472 {b[3] , sn},
473 {b[3] , en},
474 {b[3] , sn},
475 {b[3] , sn},
476 {b[3] , sn},
477 {b[3] , sn},
478 {b[3] , sn},
479 {b[3] , sn},
480 {d[4] , en},
481 {d[4] , en}, //darude_up_da_da end
482 {b[3] , sn}, //darude_up_da_da start
483 {b[3] , sn},
484 {b[3] , sn},
485 {b[3] , sn},
486 {b[3] , en},
487 {b[3] , sn},
488 {b[3] , sn},
489 {b[3] , sn},
490 {b[3] , sn},
491 {b[3] , sn},
492 {b[3] , sn},
493 {d[4] , en},
494 {d[4] , en}, //darude_up_da_da end
495 {b[3] , sn}, //darude_up_da_da start
496 {b[3] , sn},
497 {b[3] , sn},
498 {b[3] , sn},
499 {b[3] , en},
500 {b[3] , sn},
501 {b[3] , sn},
502 {b[3] , sn},
503 {b[3] , sn},
504 {b[3] , sn},
505 {b[3] , sn},
506 {d[4] , en},
507 {d[4] , en}, //darude_up_da_da end
508 {b[3] , sn}, //darude_up_da_da start
509 {b[3] , sn},
510 {b[3] , sn},
511 {b[3] , sn},
512 {b[3] , en},
513 {b[3] , sn},
514 {b[3] , sn},
515 {b[3] , sn},
516 {b[3] , sn},
517 {b[3] , sn},
518 {b[3] , sn},
519 {d[4] , en},
520 {d[4] , en}, //darude_up_da_da end
521 {b[3] , sn}, //duh-duh-duh-da-da start
522 {b[3] , sn},

```

```

523 {b[3]    , en},
524 {d[4]    , en},
525 {d[4]    , en}, //duh-duh-duh-da-da end
526 {b[3]    , sn}, //duh-duh-duh-da-da start
527 {b[3]    , sn},
528 {b[3]    , en},
529 {d[4]    , en},
530 {d[4]    , en}, //duh-duh-duh-da-da end
531 {b[3]    , sn}, //duh-duh-duh-da-da start
532 {b[3]    , sn},
533 {b[3]    , en},
534 {d[4]    , en},
535 {d[4]    , en}, //duh-duh-duh-da-da end
536 {b[3]    , sn}, //duh-duh-duh-da-da start
537 {b[3]    , sn},
538 {b[3]    , en},
539 {d[4]    , en},
540 {d[4]    , en}, //duh-duh-duh-da-da end
541 {b[3]    , sn}, //duh-duh-da start
542 {b[3]    , sn},
543 {d[4]    , en}, //duh-duh-da end
544 {b[3]    , sn}, //duh-duh-da start
545 {b[3]    , sn},
546 {d[4]    , en}, //duh-duh-da end
547 {b[3]    , sn}, //duh-duh-da start
548 {b[3]    , sn},
549 {d[4]    , en}, //duh-duh-da end
550 {b[3]    , sn}, //duh-duh-da start
551 {b[3]    , sn},
552 {d[4]    , en}, //duh-duh-da end
553 {b[3]    , sn}, //duh-duh-da start
554 {b[3]    , sn},
555 {d[4]    , en}, //duh-duh-da end
556 {b[3]    , sn}, //duh-duh-da start
557 {b[3]    , sn},
558 {d[4]    , en}, //duh-duh-da end
559 {b[3]    , sn}, //duh-duh-da start
560 {b[3]    , sn},
561 {d[4]    , en}, //duh-duh-da end
562 {b[3]    , sn}, //duh-duh-da start
563 {b[3]    , sn},
564 {d[4]    , en}, //duh-duh-da end
565 {e[5]    , fn}
566 };

```

code/song.h

3.2 Music Player

```

1  /*
2  *
3  * Xilinx , Inc .
4  * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A
5  * COURTESY TO YOU. BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS
6  * ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION OR
7  * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION
8  * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE
9  * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION
10 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO
11 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO

```

```

12  * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE
13  * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY
14  * AND FITNESS FOR A PARTICULAR PURPOSE.
15  */
16
17 /*
18  *
19  *
20  * This file is a generated sample test application.
21  *
22  * This application is intended to test and/or illustrate some
23  * functionality of your system. The contents of this file may
24  * vary depending on the IP in your system and may use existing
25  * IP driver functions. These drivers will be generated in your
26  * SDK application project when you run the "Generate Libraries" menu item.
27  *
28  */
29
30 #include <stdio.h>
31 #include "xparameters.h"
32 #include "xil_cache.h"
33 #include "xgpio.h"
34 #include "gpio_header.h"
35 #include "song.h"
36
37 XGpio GpioOutput; /* The driver instance for GPIO Device configured as O/P */
38
39 #define F_CLK 100000000.0
40 #define CONV 100000000.0 / 1024.0
41 #define DELAY 3000000
42
43 int setupAudio() {
44     int Status;
45
46     /*
47      * Initialize the GPIO driver so that it's ready to use,
48      * specify the device ID that is generated in xparameters.h
49      */
50     Status = XGpio_Initialize(&GpioOutput, XPAR_AXI_GPIO_1_DEVICE_ID);
51     if (Status != XST_SUCCESS) {
52         return XST_FAILURE;
53     }
54
55     /* Set the direction for all signals to be outputs */
56     XGpio_SetDataDirection(&GpioOutput, 1, 0x0);
57
58     /* Set the GPIO outputs to low */
59     XGpio_DiscreteWrite(&GpioOutput, 1, 0x0);
60     return XST_SUCCESS;
61 }
62
63 void tone(int freq, int length) {
64     long note = F_CLK / (1024 * freq);
65     XGpio_DiscreteWrite(&GpioOutput, 1, note);
66     int timeDelay = (length * DELAY) / bpm;
67     for(int Delay = 0; Delay < timeDelay; Delay++);
68     XGpio_DiscreteClear(&GpioOutput, 1, 0x0);
69     for(int Delay = 0; Delay < 800000; Delay++);
70 }

```

```

71
72
73 int main ()
74 {
75     Xil_ICacheEnable();
76     Xil_DCacheEnable();
77     print("——Entering main——\n\r");
78
79
80
81
82     setupAudio();
83
84
85
86     /*
87     * Peripheral SelfTest will not be run for axi_uartlite_0
88     * because it has been selected as the STDOUT device
89     */
90     for (int i = 0; i < (sizeof(song)/sizeof(song[0])); i++){
91         tone(song[i][0], song[i][1]);
92     }
93
94
95
96     print("——Exiting main——\n\r");
97     Xil_DCacheDisable();
98     Xil_ICacheDisable();
99     return 0;
100 }

```

code/testperiph.c