ELEX 7660: Lab 3

Tone Generator

Nicholas Huttemann	30/01/2019	Set: T

Table of Contents

Table of Figures

1 Questions

1.1 How many registers are used in the tonegen module?

Two – ‘count’ and ‘freq’

1.2 What are the possible values that can be loaded into each register?

Count can be loaded with ‘fclk’ and freq can be loaded with the tone frequency from ‘writedata’

1.3 What are the conditions under which each value is loaded?

Count: The value gets loaded during a reset or when count <= 0.
Freq: The value gets loaded when ‘write’ is high.

2 Code

2.1 Tonegen.sv

```
// Nicholas Huttemann 2018-01-30
// tonegen.sv - tone generator for ELEX 7660 Lab 3
// Creates a square wave on the spkr (speaker) output at a
// frequency given by the 'freq' control register.

module tonegen
  #(logic [31:0] fclk)           // clock frequency, Hz
   (input logic [31:0] writedata, // Avalon MM bus, data
    input logic write,            // " write strobe
    output logic spkr,            // on/off output for audio
    input logic reset, clk ) ;

int count;
logic [31:0] freq;

always_ff @ (posedge clk, negedge reset) begin
      if (reset) begin
            freq = 0;
            count = fclk;
            spkr = 0;
      end
      if (write) begin             // Get the tone freq from the data bus
            freq = writedata;
      end

      // The half period of the tone freq has been reached; toggle the speaker
      if (count <= 0) begin
            spkr ^= 1;
            count = fclk;
      end
      // Subtract twice the tone frequency every clock tick
      count = count - 2*freq;
end
endmodule
```

2.2 Toneplayer.c

```
// toneplayer.c - play a tune using hardware tone generator ELEX 7660 201710 Lab 3
// outline by Ed. Casas 2017-1-22
// modified by: Nicholas Huttemann 2019-1-27

#include "unistd.h"   /* for usleep() */

#define N 72

int freq [N] = { 330, 392, 440, 494, 523, 494, 440, 370, 294,
    330, 370, 392, 330, 330, 311, 330, 370, 311, 247, 330, 392,
    440, 494, 523, 494, 440, 370, 294, 330, 370, 392, 370, 330,
    311, 277, 294, 330, 330, 587, 587, 554, 494, 440, 370, 294,
    330, 370, 392, 330, 330, 311, 330, 370, 311, 247, 587, 587,
    554, 494, 440, 370, 294, 330, 370, 392, 370, 330, 311, 277,
    294, 330, 330} ;

int duration [N] = { 2, 4, 2, 3, 1, 2, 4, 2, 3, 1, 2, 4, 2, 3, 1, 2,
    4, 2, 4, 2, 4, 2, 3, 1, 2, 4, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2,
    6, 4, 6, 3, 1, 2, 4, 2, 3, 1, 2, 4, 2, 3, 1, 2, 4, 2, 6, 6,
    3, 1, 2, 4, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 6, 4} ;

#if 0

#include "system.h"   /* peripheral base addresses */
#define SETFREQ(x) (*(int*)TONEGEN_0_BASE) = (x)

#else

#define SETFREQ(x) (printf("%d Hz",x))
#define usleep(x)  (printf(" for %d ms\n",x/1000))

#endif

int main()
{

   for (int i = 0; i <= N; i++) // Play all the tones in freq
   {
     SETFREQ(freq[i]);
     usleep(duration[i]*150000);
   }

   SETFREQ(0) ;

   return 0;
}
```

2.2.1 Output

```
392 Hz for 600 ms
440 Hz for 300 ms
494 Hz for 450 ms
523 Hz for 150 ms
494 Hz for 300 ms
440 Hz for 600 ms
370 Hz for 300 ms
294 Hz for 450 ms
330 Hz for 150 ms
370 Hz for 300 ms
```

Figure 1: tongeplayer.c Output

3 Testbench Waveforms

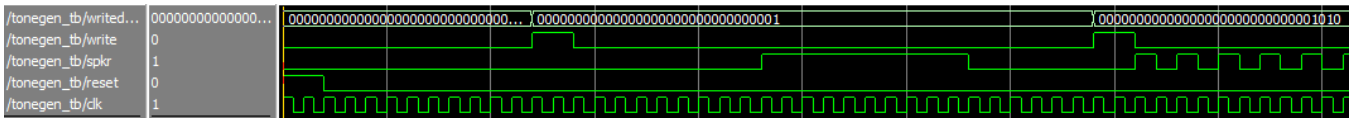

Figure 2: tonegen_tb.sv