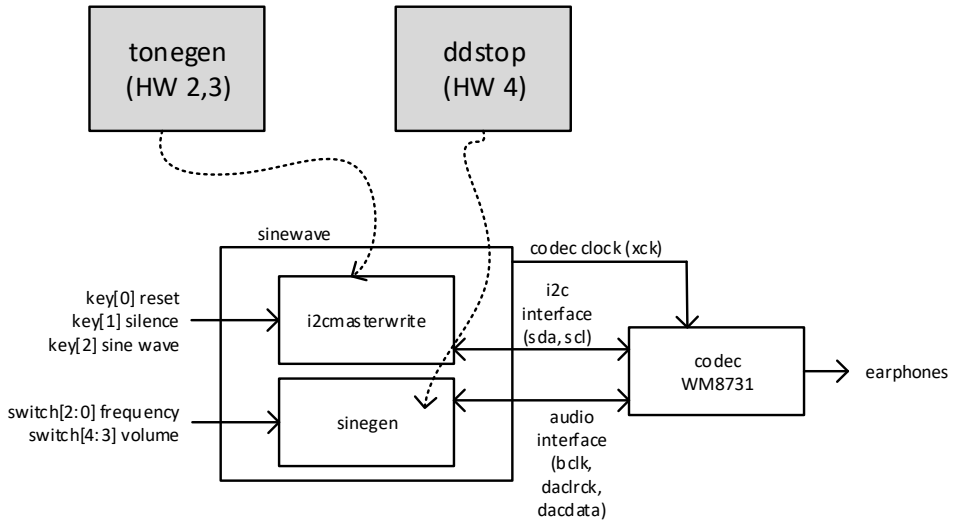
ECE 4514 Fall 2019: Homework 5

Assignment posted on March 5
Assignment due on March 21 11:59 PM

Homework weight: 100 points

- Homework 5 integrates the key components developed in the previous assignments (homework 2, 3 and 4). You will develop a sine wave generator which drives the CODEC with a sine wave of selectable frequency and volume. You will receive the solutions for the previous homework (2, 3, 4), but you need to complete the integration of the generator.

Sine wave generator



The sine wave generator contains two modules, including an I2C master interface and a DDS cordic-based sine generation module. Three keys of the DE1-SoC board are used to reset the design (key[0]), silence the design (key[1]) and enable the sine generation (key[2]). The I2C settings are identical to those of tonegen: 48KHz, 16-bit, left-aligned.

The switches of the DE1-SoC board are used to set the frequency and volume.

- Switch sw[2:0] is used to select the frequency from 1KHz (000) to 8 KHz (111) in steps of 1 KHz. For example, the setting 110 should result in a sine wave of 6KHz.
- Switch sw[4:3] set one of four volume levels. The setting 00 to 11 select an attenuation of 2^0, 2^2, 2^4 or 2^8. The 'full volume' level corresponds to samples of type fix<16,14> being send to the CODEC. Thus, +1.0 corresponds to 0x4000 and -1.0 corresponds to 0xC000. An attenuation of K is obtained by arithmetic right-shift on these values. For example, the switch setting 10, corresponding to an attenuation of $K=2^4$, is obtained by arithmetic right-shift over 4 bits. So in this case, +1.0 corresponds to 0x0400 and -1.0 corresponds to 0xFC00.

Starting the design

After downloading the repository, you will find the following directories:

- `tonegen`: A sample solution for homework 3. This version allows you to set the values of the top-4 bits of the high-value and the low-value of the 1KHz square wave send to the codec. You will need to refer to the tonegen solution to construct sinegen. The tonegen contains the I2C programming modules, along with the correct I2C settings to put the codec into the proper mode. The tonegen also contains a module audiogen that you can use as a blueprint for sinegen. It shows how to create the proper timing for BCLK, DACLRCK, DACDAT.
  Compile and download the tonegen design to the board as follows.

  ```
  quartus_sh --flow compile tonegen
  quartus_pgm -m jtag -o "p;tonegen.sof@2"
  ```

- `cordicdds`: A sample solution for homework 4. This version is written for a datapath with an internal precision of 20 bit. Furthermore, the design is synthesizable. You can simulate the solution in modelsim as follows.

  ```
  vlib work
  vlog accum.v
  vlog cordic.v
  vlog cordicconst.v
  vlog encoder.v
  vlog ddstop.v
  vlog ddstoptb.v
  vsim -c ddstoptb -do "run -all; exit"
  ```

  You need the ddstop module to develop sinegen.

  ```
  module ddstop(input wire clk,
                input wire       reset,
                input wire  [19:0] increment,
                input wire       update,
                output wire [15:0] q,
                output wire      ready);
  ```

  The 20-bit increment is a signal in <20,15> format. For example, the value pi / 6 = 0.5236 would be represented as the integer 17,157. You will need to determine the proper increment for each of the 8 frequencies, 1 KHz to 8KHz, keeping in mind that the codec sample rate is set at 48 KHz.

  The control interface of ddstop is a simple update – ready protocol. After asserting update to logic high for one clock cycle, ddstop computes the next cordic output and places the result in the q output as a <16,14> number, and asserts ready.

- `sinewave-generator`: This is the top-level of the module that you need to develop. You cannot make changes to the top-level file `sinewave.v`. Instead, you have to provide the implementation for two modules that are instantiated by this top-level. These two modules are `sinegen` and `i2cmasterwrite`. You may make use of the files and materials available in

tonegen and cordicdds. Keep in mind, however, that there is more work needed than simply copying over the files and recompiling!

Verilator

You must run all of the Verilog code that you design, with exception of the testbench, through the lint checker of verilator. To install Verilator, follow the instructions on https://www.veripool.org/projects/verilator/wiki/Installing

Installation under Cygwin is straightforward.

To run verilator in link checker mode, on e.g. ddstop.v, use the following command line:

```
verilator –lint-only ddstop.v
```

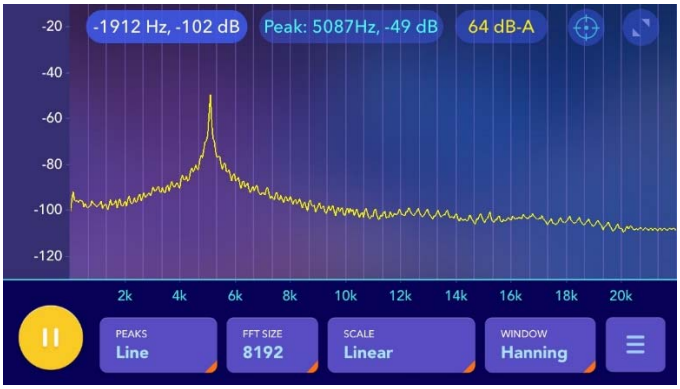
The tool generates warning messages when your code is not consistent. Here is an example of such a warning message:

```
$ verilator --lint-only ddstop.v
%Warning-WIDTH: ddstop.v:40: Operator ASSIGNW expects 17 bits on the
Assign RHS, but Assign RHS's COND generates 18 bits.
%Warning-WIDTH: Use "/* verilator lint_off WIDTH */" and lint_on
around source to disable this message.
%Error: Exiting due to 1 warning(s)
%Error: Command Failed /usr/local/bin/verilator_bin --lint-only
ddstop.v
```

To receive full grade on this assignment, none of the files that you turn in (with exception of the testbench `ddstoptb.v`) is allowed to generate a warning under verilator.

What to turn in

You need to complete the design sinewave-generator and test it. Plug in your earbuds. Use a spectrum analyzer app such as Spectrum Analyzer Pro to check the spectrum of the tones. In a good solution, you should find that harmonics are virtually absent. The following is an example for a 5KHz generator setting. Note that the frequency error, 5,087 Hz instead of 5,000Hz, is due to the frequency error on the system clock (12.5 MHz instead of 12.288MHz).



Coding Guidelines

Please refer to Chapter “RTL Coding Guidelines “ of the Reuse Methodology Manual (accessible for free on the Tech network: https://link.springer.com/book/10.1007%2Fb116360). These are generic guidelines for HDL coding; I expect you to aim to follow at least Chapter 5.2 and Chapter 5.5.

I value code clarity but I do not intend to penalize you if you don’t follow these guidelines to the letter. In the other hand, if you turn in messy code with obvious violations against the principles and main directives in this guideline, you will receive a penalty of up to 25% of the points (even if everything works perfectly).