

KringleCon 4: Calling Birds!

13) FPGA Programming

1. Click **Map** (Destinations) icon and then click **Frost Tower Rooftop**
2. Click **FPGA Programming**

EE/CS 302 - Exercise #4

Hello, students! In exercise #4, we continue our FPGA journey, documenting the creation of the sound chip for this holiday season's new *Kurse 'em Out Karen* doll. Our goal is to make the doll say its [trademark phrase](#). But, as I always tell you in class, we must walk before we run.

Before the doll can say anything, we must first have it make noise. In this exercise you will design an FPGA module that creates a square wave tone at a variable frequency.

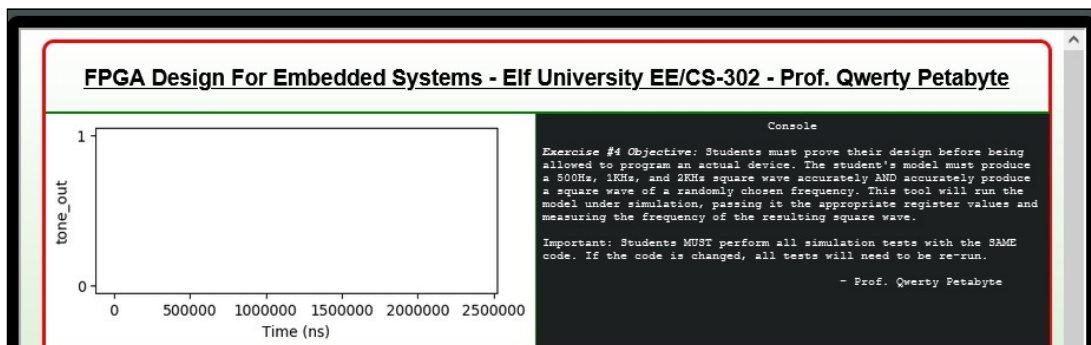
Creating a square wave output takes our clock signal (which is also a square wave) and uses a counter to divide the clock to match the desired frequency. One tricky problem that we'll encounter is that Verilog (v1364-2005) doesn't have a built-in mechanism to *round* real numbers to integers, so you'll need to devise a means to do that correctly if you want your module to match frequencies accurately.

Good luck and always remember:

If $\$toi(real_no * 10) - (\$rtoi(real_no) * 10) > 4$, add 1

- Prof. Qwerty Petabyte

3. Close the **EE/CS 302 - Exercise #4** window



4. Scroll **Down** the window

```

1 // Note: For this Lab, we will be working with QRP Corporation's CQC-11 FPGA.
2 // The CQC-11 operates with a 125MHz clock.
3 // Your design for a tone generator must support the following
4 // inputs/outputs:
5 // (NOTE: DO NOT CHANGE THE NAMES. OUR AUTOMATED GRADING TOOL
6 // REQUIRES THE USE OF THESE NAMES!)
7 // input clk - this will be connected to the 125MHz system clock
8 // input rst - this will be connected to the system board's reset bus
9 // input freq - a 32 bit integer indicating the required frequency
10 // (0 = 9999.99Hz) formatted as follows:
11 // 32'hf1206 or 32'o987654 = 9876.54Hz
12 // output wave_out - a square wave output of the desired frequency
13 // you can create whatever other variables you need, but remember
14 // to initialize them to something!
15
16 `timescale 1ns/1ns
17 module tone_generator (
18     input clk,
19     input rst,
20     input [31:0] freq,
21     output wave_out
22 );
23 // ---- DO NOT CHANGE THE CODE ABOVE THIS LINE ----
24 // ---- IT IS NECESSARY FOR AUTOMATED ANALYSIS ----
25 // TODO: Add your code below.
26 // Remove the following line and add your own implementation.
27 // Note: It's silly, but it compiles...
28 assign wave_out = (clk | rst | (freq > 0));
29 endmodule

```

Load

Save

Simulate 500Hz

Simulate 1KHz

Simulate 2KHz

Simulate Random

Frequency:

Simulate Frequency

Play Sound

Program Device

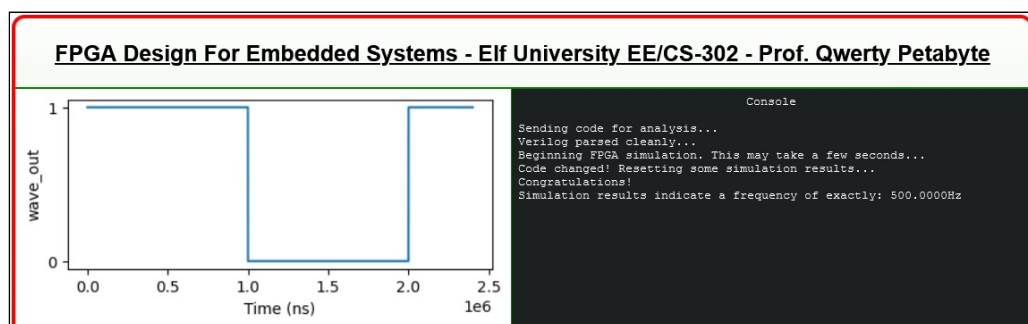
5. Scroll to the **Top** of the window
6. Replace line 28 onwards with the following code

```
reg sq_wave;
integer counter;

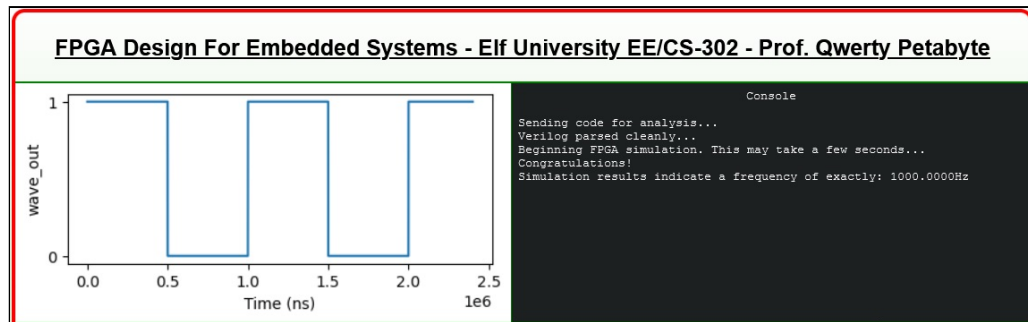
assign wave_out = sq_wave;

always @(posedge clk or posedge rst)
begin
    if(rst==1)
        begin
            counter <= 0;
            sq_wave <= 0;
        end
    else
        begin
            if(counter <= 0)
                begin
                    counter <= ((125000000 / $rtoi(freq / 100)) / 2) - 1;
                    if ($rtoi(freq * 10) - ($rtoi(freq) * 10) > 4)
                        begin
                            counter <= counter + 2;
                        end
                    sq_wave <= sq_wave ^ 1'b1;
                end
            else
                begin
                    counter <= counter - 1;
                end
            end
        end
    end
endmodule
```

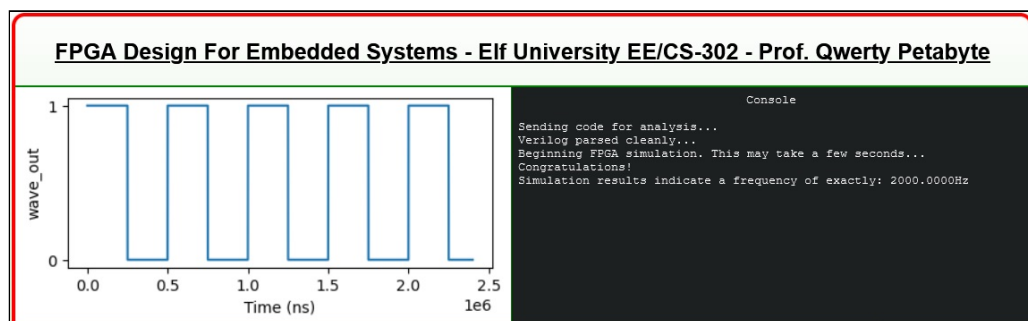
7. Click the **Simulate 500Hz** button



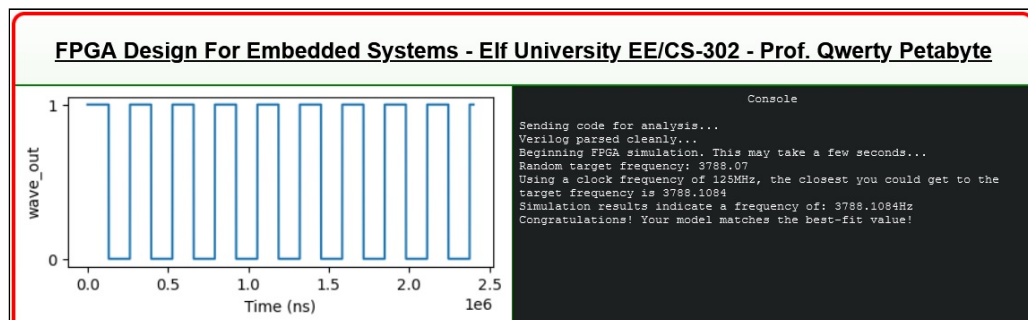
8. Click the **Simulate 1KHz** button



9. Click the **Simulate 2KHz** button



10. Click the **Simulate Random** button



11. Click the **Program Device** button

12. Click the **Close** button

13. Click **Objectives** (tick icon)

14. Scroll **Down**

15. Click **13) FPGA Programming**

✓ **13) FPGA Programming**

Difficulty: 🚩🚩🚩🚩🚩

Write your first FPGA program to make a doll sing.
You might get some suggestions from Grody Goiterson,
near Jack's elevator.